Fehlerschutz–Codierung 1: Block Codes

Inhaltsverzeichnis

Abbildungsverzeichnis

Fehlerschutz–Codierung 1: Block Codes

Nachdem Shannon im Jahre 1948 bewiesen hatte, daß es möglich ist, eine Datenrate, die kleiner als die Kanal–Kapazität ist, fehlerfrei zu übertragen, setzte eine intensive Suche nach geeigneten Codes ein. Leider hatte Shannen nur gezeigt, daß es möglich ist, aber nicht wie.

Es bildeten sich 2 Richtungen heraus. Die erste Richtung sah das Codierungsproblem als eine algebraische Aufgabe an. Hieraus entstanden die **Block–Codes**. Die zweite Richtung behandelte die Codierung stochastisch. Daraus entstanden die **Faltungs–Codes**.

Obwohl man heute mit der „Turbo–Codierung" der Shannon–Grenze schon sehr nahe gekommen ist, besteht durchaus die Möglichkeit, daß noch bessere und/oder günstiger zu implementierende Codes gefunden werden.

Die Notwendigkeit für eine Fehlerschutz–Codierung ergab sich nicht nur bei der Übertragung, sondern auch bei der Speicherung digitaler Daten. Gerade bei der Speicherung sind die enormen Fortschritte der Packungsdichte, die zu immer „größeren" Laufwerken (im PC) führen, ohne eine äußerst effektive Fehlerschutz–Codierung undenkbar. Schließlich würde ein einziger Bit–Fehler eine ausfürbare Datei abstürzen lassen. Hier sind also z.B. nur Fehler–Raten von BER $< 10^{-12}$ zulässig.

Ähnliche, wenn auch geringere Anforderungen stellen z.B. digitale Speicherverfahren für Audio– und Video–Files.

1 Fehlerschutz–Strategien

Die Art und Weise, wie die Verbindung zwischen Daten–Sender und Daten–Empfänger beschaffen ist, bestimmt die Strategie, die zum Fehlerschutz angewendet werden kann. Es sind je nach Anwendung folgende Verbindungsarten möglich, Bild 1.1.

Unidirektionale Verbindung: Übertragungsrichtung nur vom Sender zu (einem oder) mehreren Empfängern (Broadcast–Betrieb)

Simplex Verbindung: Übertragung zwischen zwei Terminals (jeweils Sender und Empfänger), wobei immer nur eine Richtung jeweils aktiv sein kann. Der Empfänger kann Rückmeldung geben, ob er die Daten fehlerfrei empfangen hat.

Duplex Verbindung: Zwischen zwei Terminals bestehen gleichzeitige Verbindungen in beiden Richtungen.

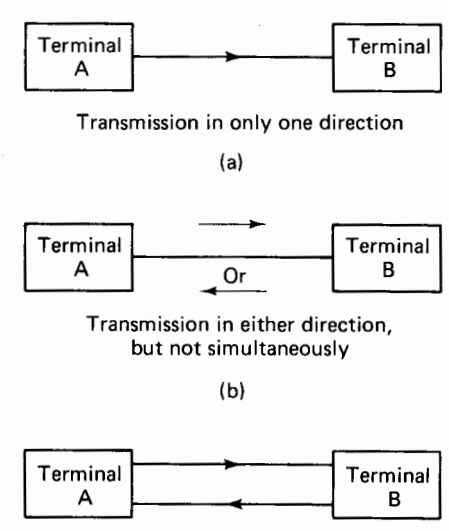

Bild 1.1: Verbindungsmöglichkeiten zwischen Daten–Sender und Daten–Empfänger

1.1 Fehlerschutz durch Rückfrage (ARQ)

Bei Simplex– oder Duplex–Verbindungen wird der Fehlerschutz ermöglicht durch automatische Rückmeldung über die empfangenen Daten und die Aufforderung, diese im Fehlerfall zu wiederholen (retransmission query). Im Fehlerfall erhält der Sender eine Automatic Repeat Request (ARQ). Es sind 3 Protokolle üblich. Im Fall a) wird jedes Bit vom Empfänger bestätigt (ACK, acknowledge) oder im Fehlerfall erneut angefordert (NAK, negative acknowledge). Es genügt eine Simplex–Verbindung. Eine solche Strategie führt naturgemäß auf einen geringen Datendurchsatz. Im Fall b) wird nicht auf ACK bzw. NAK gewartet, jedoch im Falle eines NAK (nach der durch die Übertragung bedingten Laufzeit) alles ab der fehlerhaften Stelle wiederholt. Dabei kann nunmehr ein Fehler bei einer zuvor richtig empfangenen Stelle auftreten (Duplex–Verbindung erforderlich). Im Fall c) werden die einmal richtig empfangenen Bits nicht verworfen, sondern nur die fehlerhaften wiederholt (selective repeat), die der Empfänger schließlich an der richtigen Stelle einordnen muß. Bild 1.2 zeigt diese Strategien.

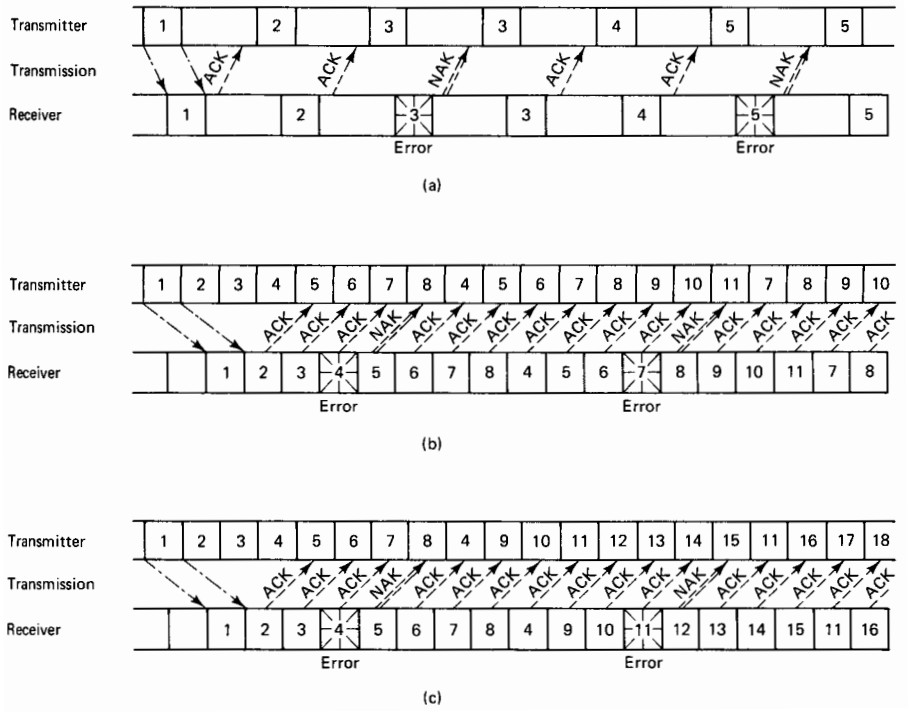

Bild 1.2: ARQ Strategien

1.2 Forward Error Correction (FEC)

Ist keine Simplex– oder Duplex–Verbindung möglich, muß der zu übertragende Datenstrom durch Hinzufügung von Fehlerschutz–Symbolen geschützt werden. Der Datenstrom wird damit codiert.

Die weiteren Abschnitte befassen sich mit dieser Themenstellung.

2 Grundlagen der Block–Codes

Wie der Name sagt, werden hier die Informationen zu Blöcken zusammengefaßt und entsprechend verarbeitet. Der Block–Codierer (encoder) verarbeitet k Informations–Bits und erzeugt daraus n Kanal–Bits für die Übertragung. Auf diese Weise entsteht ein (n, k) Code.

2.1 Code–Rate

Die Code–Rate eines (n, k) Codes ist definiert mit $R_c = \frac{k}{n}$.

$$\boxed{\textbf{Code Rate } R_c = \frac{k}{n} = \frac{\text{Anzahl der Nutzbits, („Netto–Bits“)}}{\text{Anzahl der übertragenen Bits, („Brutto–Bits“)}} < 1} \tag{2.1}$$

Pro Kanal–Bit werden somit R_c Informations–Bits übertragen, also Bruchteile eines Informations–Bits.

2.2 Hamming–Distanz und –Gewicht

Ein Block mit k Informations–Bits wird dargestellt als k–Vektor $\mathbf{m}$, wobei jedes m_j entweder 0 oder 1 ist.

$$\mathbf{m} = [m_0, m_1, m_2, \cdots, m_{k-1}] \tag{2.2}$$

Es gibt 2^k unterschiedliche Informations–Blöcke bzw. 2^k binäre k–Vektoren $\mathbf{m}$.

Für die Quelle sei dabei unterstellt, daß alle Informations–Blöcke $\mathbf{m}$ gleich wahrscheinlich sein sollen. Für deren Wahrscheinlichkeit $P(m)$ gilt dann:

$$P(m) = 2^{-k} \tag{2.3}$$

Der Block–Coder bildet sämtliche k–Eingangs–Vektoren $\mathbf{m}$ ein–eindeutig auf die zugehörigen k Stück Code–Vektoren $\mathbf{c}$ ab.

$$\mathbf{c} = [c_0, c_1, c_2, \cdots, c_{n-1}] \tag{2.4}$$

Jedes $\mathbf{c}$ ist ein Code–Wort oder Code–Vektor der Länge n.

Es gibt 2^n (mögliche) Code–Worte für 2^k (unterschiedliche) Informations–Blöcke. Da $k < n$ ist, werden nur 2^k Code–Worte verwendet. Diese sind so ausgesucht, daß sie sich in möglichst vielen Positionen unterscheiden. Der Anteil der verwendeten Code–Worte, bezogen auf alle theoretisch möglichen, beträgt:

$$\boxed{2^{k-n} = 2^{k(1-1/R_c)}} \tag{2.5}$$

Man sieht hieraus, daß für eine konstante Code–Rate R_c der Prozentsatz der verwendeten Code–Worte abnimmt, wenn die Länge k der Informations–Blöcke zunimmt. **Größere Block–Codes sind also besser**.

Die Fähigkeit eines Codes, Fehler zu korrigieren beruht darauf, daß nicht alle 2^n möglichen Code–Worte verwendet werden. Entsteht durch Fehler ein nicht verwendetes (und damit unzulässiges) Code–Wort $\mathbf{c_u}$, läßt sich im Empfänger dasjenige Code–Wort $\mathbf{c}$ identifizieren, das am ähnlichsten ist, d.h. welches sich an den wenigsten Positionen unterscheidet.

Zwei (zulässige) Code–Worte $\mathbf{c}$ und $\mathbf{c}'$ sollen sich in l Positionen unterscheiden. Dieser Unterschied zwischen den Code–Worten ist deren Hamming–Distanz $d_H(\mathbf{c}, \mathbf{c}') = l$.

Als **Hamming–Distanz** d_H eines Codes wird die minimal zwischen 2 (beliebigen zulässigen) Code–Worten auftretende Hamming–Distanz $d_{H_{\min}}$ bezeichnet. Allgemein findet man diese in dem kleinsten Abstand vom „Null“–Wort. Dieser Minimal–Abstand $d_{H_{\min}}$ wird auch als **Gewicht des Codes** w_H bezeichnet.

Sollen t **Fehler** von einem Code **korrigiert** werden können, muß dessen Hamming–Distanz d_H

$$\boxed{d_H = w_H \geq 2t + 1} \tag{2.6}$$

sein.

2.3 Kanal–Modell für Block–Codes

Bei der Betrachtung der Block–Codes wird der Digitale Modulator und der Digitale Demodulator einschließlich des Entscheiders in den Kanal einbezogen. Damit kann der Kanal als binär und symmetrisch dargestellt werden, Bild 2.1. Es bedeuten:

- $Pr(0|1) = p$ Wahrscheinlichkeit dafür, daß bei einer gesendeten 1 eine 0 detektiert wurde
- $Pr(1|0) = p$ Wahrscheinlichkeit dafür, daß bei einer gesendeten 0 eine 1 detektiert wurde
- $Pr(0|0) = 1 - p$ Wahrscheinlichkeit dafür, daß eine 0 richtig detektiert wurde
- $Pr(1|1) = 1 - p$ Wahrscheinlichkeit dafür, daß eine 1 richtig detektiert wurde

Die Fehlerwahrscheinlichkeiten für $0 \rightarrow 1$ und $1 \rightarrow 0$ sind gleich, wenn sich der Kanal symmetrisch verhält. Der Block–Decoder erhält Bits, die Ergebnis einer „harten“ Entscheidung (0 oder 1) sind.

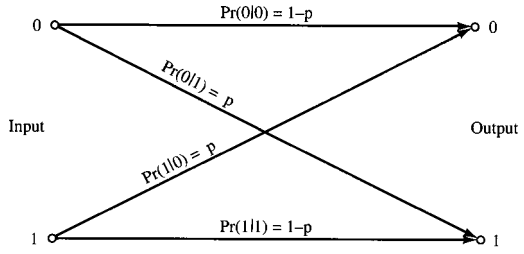

Bild 2.1: Übergangs–Wahrscheinlichkeiten des symmetrischen Binär–Kanals

2.4 Beispiele einfacher Block Codes

2.4.1 Wiederholungs–Code

Ein (relativ primitiver) Code für niederratige Übertragung mit guter Fehlerschutzeigenschaft ist der Wiederholungs–Code. Da hier die Hamming Distanz $d_H = 5$ ist, lassen sich bis zu 2 Fehler korrigieren.

$$0 \longrightarrow 00000$$
$$1 \longrightarrow 11111$$

Der Wiederholungs–Code läßt sich auch anders interpretieren. Tatsächlich handelt es sich eigentlich nur um eine langsamere Übertragung (ohne Codierung). Damit erhöht sich die pro Bit übertragene Energie hier um den Faktor $n = 5$. Also wird das Verhältnis E_b/N_0 um diesen Faktor n oder $10\log_{10}(5) = 6.99$ dB verbessert. Mit Matched–Filter–Empfang verbessert sich damit die Bitfehler–Rate z.B. für eine BPSK Übertragung entsprechend, wie sich aus den „Wasserfall–Kurven“ der Q–Funktion entnehmen läßt, siehe z.B. auch Bild 2.7 die durchgezogene Kurve.

2.4.2 Parity Check

Hierfür wird der Datenstrom segmentiert in Stücke à $k = 4$ (bis $k = 8$) Bits (k–Vektoren). Jeder dieser k–Vektoren („Netto–Bits“) wird dann um 1 Bit erweitert, so daß $(k + 1)$–Vektoren („Brutto–Bits“) entstehen. Das Parity–Bit wird so bestimmt, daß die Quersumme modulo 2

- 0 ergibt bei gerader Parität (even parity)
- 1 ergibt bei ungerader Parität (odd parity).

Es entsteht so ein $(k + 1, k)$ Code, der eine Code Rate $R_c = \frac{k}{k+1}$ hat.

In einem Beispiel mit $k = 4$, $R_c = \frac{4}{5}$ und Even Parity erhält man die folgende Tabelle. Die Parity Bits stehen rechts hinter dem Punkt.

$$\begin{array}{l} 0000 \longrightarrow 0000.0 \\ 0001 \longrightarrow 0001.1 \\ 0010 \longrightarrow 0010.1 \\ 0011 \longrightarrow 0011.0 \\ \quad\vdots \quad\longrightarrow\quad \vdots \;\; \vdots \\ 1111 \longrightarrow 1111.0 \end{array}$$

Parity Check Verfahren lassen eine Fehler–Erkennung zu. Es wird 1 Fehler (allgemein: eine ungerade Anzahl von Fehlern) erkannt. Dagegen wird eine gerade Anzahl von Fehlern nicht erkannt.

Die Code–Worte in dieser Tabelle unterscheiden sich in (mindestens) 2 Positionen. Die minimale Distanz zum „Null“–Wort und damit das Gewicht dieses Codes ist $d_{H_{\min}} = w_H = 2$. Parity Check Codes mit $d_H = w_H = 2$ können somit keine Fehler korrigieren.

2.4.3 Unentdeckte Fehler

Es sei angenommen, daß alle Bit–Fehler unabhängig von einander entstehen und gleich wahrscheinlich auftreten. Dann ist die Wahrscheinlichkeit P dafür, daß j Fehler in einem n–Vektor (oder Block von n Bits) auftreten mit

$$\boxed{P(j,n) = \binom{n}{j} p^j (1-p)^{n-j} = \frac{n!}{j!(n-j)!} \cdot p^j (1-p)^{n-j}} \tag{2.7}$$

gegeben, wobei p die Wahrscheinlichkeit dafür ist, daß ein Vektor fehlerhaft empfangen wurde. Im Beispiel können das $j = 2$ Fehler oder $j = 4$ Fehler pro Vektor sein, die dann unentdeckt bleiben.

Es sei angenommen, daß die Wahrscheinlichkeit für den Fehler eines Vektors $p = 10^{-3}$ betrage. Dann ergibt sich im vorherigen Beispiel mit $n = 5$ für die Wahrscheinlichkeit P_{nd} unentdeckter Doppel– und Vierfach–Fehler:

$$\begin{aligned} P_{nd} &= \binom{5}{2} p^2 (1-p)^3 + \binom{5}{4} p^4 (1-p)^1 && (2.8)\\ &= \underbrace{10 \cdot 10^{-6} - 10 \cdot 10^{-15}}_{\text{Doppelfehler}} + \underbrace{5 \cdot 10^{-12} - 5 \cdot 10^{-15}}_{\text{Vierfachfehler}} && (2.9)\\ &\approx 10^{-5} \qquad \text{Doppelfehler dominieren} && (2.10) \end{aligned}$$

Parity Check Verfahren sind wirkungsvoll, wenn die Wahrscheinlichkeit für Fehler eines n–Vektors klein ist.

2.4.4 Parity Check mit rechtwinkligen Codes

Die Bezeichnung „rechtwinkliger“ Code oder „Produkt–Code“ weist auf die Anordnung der k–Vektoren hin, die verwendet wird, um die Parity Bits zu gewinnen. Hierzu werden die k–Vektoren in einen rechteckig organisierten Speicher in M Reihen und N Spalten geschrieben. Danach werden Parity Bits für jede der M Reihen und N Spalten gebildet, so daß eine Anordnung von $M + 1$ Reihen und $N + 1$ Spalten entsteht. Auf den letzten verbliebenen Platz kommt das Parity Bit aus den anderen Parity Bits, Bild 2.2.

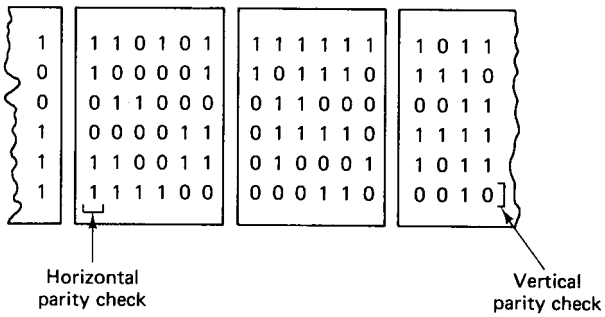

Bild 2.2: Parity Check in Reihen und Spalten

Die Code–Rate eines rechtwinkligen Codes wird damit:

$$\boxed{R_c = \frac{k}{n} = \frac{MN}{(M+1)(N+1)}} \tag{2.11}$$

In Bild 2.2 ist ein Beispiel dargestellt mit $M = 5$ und $N = 5$. Die horizontalen Parity Check Bits sind in dieser Darstellung links angeordnet. Jedes dieser Felder ist in der Darstellung von einem Rechteck eingerahmt. Es ist ein $(36, 25)$–Code mit einer Code–Rate $R_c = 25/36 = 0.6944\cdots$. Das rechteckförmige Schema erlaubt die Lokalisierung und Korrektur eines Einzelfehlers $t = 1$.

Man erkennt, daß sich diese Anordnung auch für eine parallele Übertragung (über 6 Sub–Channel in Blöcken à 6 Bit) eignen würde. Bei der Übertragung über einen Kanal werden die k–Vektoren z.B. horizontal

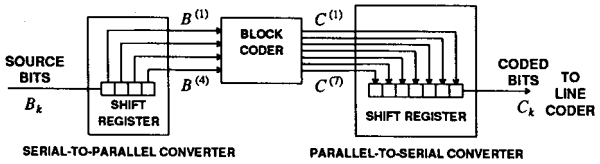

Bild 2.3: Ein– und Auslesen eines Block–Encoders (Beispiel für einen (7, 4) Code)

Reihe für Reihe eingelesen. Nachdem nun die Parity Bits berechnet sind, wird (mit erhöhter Taktrate um den Faktor $1/R_c$) wieder ausgelesen, Bild 2.3.

Um einen kontinuierlichen Datenstrom verarbeiten zu können, werden 2 derartige rechteckig organisierte Speicher benötigt, von denen der erste ausgelesen wird, während der zweite gefüllt wird.

2.4.5 Blockfehler–Wahrscheinlichkeit

Die Wahrscheinlichkeit dafür, daß j Fehler in einem Block von n Bits (Binär–Symbolen) auftreten ist bereits in Gleichung (2.7) gegeben. Es soll nun die Block–Fehlerwahrscheinlichkeit P_M für einen Code berechnet werden, der in der Lage ist, t Fehler zu korrigieren. Hierfür gilt, wenn p die Wahrscheinlichkeit dafür ist, daß ein n–Vektor (oder Block) fehlerhaft empfangen wurde:

$$\boxed{P_M = \sum_{j=t+1}^{n} \binom{n}{j} p^j (1-p)^{n-j}} \tag{2.12}$$

Mit $n = 36$ und $t = 1$ wird daraus:

$$P_M = \sum_{j=2}^{36} \binom{36}{j} p^j (1-p)^{36-j} \tag{2.13}$$

Falls p genügend klein ist, dominiert der größte Term in der Summe, womit näherungsweise gilt:

$$P_M = \binom{36}{2} p^2 (1-p)^{34} \tag{2.14}$$

Die sich daraus ergebende Bit–Fehlerwahrscheinlichkeit hängt ab von dem speziellen Code und von den Eigenschaften des Decoders.

2.4.6 Interleaver

Werden die Bits in einen rechteckig organisierten Speicher zeilenweise eingelesen und dann spaltenweise ausgelesen, Bild 2.4, erreicht man damit, daß auf dem Übertragungsweg eine andere Reihenfolge betsteht.

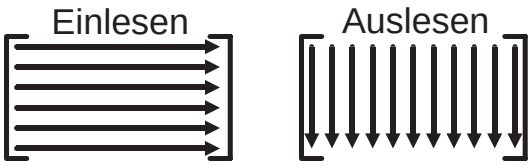

Bild 2.4: Ein– und Auslesevorgang in einem (rechteckförmigen) Block–Interleaver

Dieser Vorgang wird mit Interleaving (Verschachtelung) bezeichnet. Kommt es nun auf dem Übertragungsweg zu einem Burst–Fehler, so wird dieser nach dem empfangsseitigen De–Interleaving (spaltenweise einlesen und reihenweise auslesen) in Einzelfehler aufgelöst.

Wenn die Auflösung von Burstfehlern wirkungsvoll sein soll, muß die Größe des Interleavers ausreichend sein. Interleaven bedeutet für die Informations–Übertragung eine Sender– und Empfänger–seitige Verzögerung. Dies setzt praktische Grenzen für die Größe eines Interleavers.

2.5 (7, 4) Hamming–Code

Der (7, 4) Hamming–Code gehört zu einer Familie von (n, k) Block–Codes mit den Parametern

Block Länge	$n = 2^{\wp} - 1$
Informations Bits	$k = 2^{\wp} - \wp - 1$
Parity Bits	$\wp = n - k \geq 3$

Die Codierungs–Vorschrift für diesen (7, 4) Code (mit $\wp = 3$) ist in der folgenden Tabelle gegeben.

Nr.	Information	Code–Wort	Gewicht w_H	Nr.	Information	Code–Wort	Gewicht w_H
0	0000	000.0000	0	8	1000	110.1000	3
1	0001	101.0001	3	9	1001	011.1001	4
2	0010	111.0010	4	10	1010	001.1010	3
3	0011	010.0011	3	11	1011	100.1011	4
4	0100	011.0100	3	12	1100	101.1100	4
5	0101	110.0101	4	13	1101	000.1101	3
6	0110	100.0110	3	14	1110	010.1110	4
7	0111	001.0111	4	15	1111	111.1111	7

Da $n = 7$ ist, gibt es theoretisch $2^7 = 128$ mögliche Code–Worte. Davon wurden aber nur $2^4 = 16$ ausgewählt. Die minimale Hamming–Distanz oder das Gewicht des Codes ist $w_H = 3$. Daher ist er in der Lage, 1 Fehler zu korrigieren oder 2 Fehler zu erkennen. Dies gilt für alle Hamming–Codes.

2.5.1 Systematische und Lineare Codes

Der (7, 4) Hamming–Code ist systematisch und linear. Diese Eigenschaften sind wie folgt definiert:

Systematischer Code: Das Code–Wort besteht aus 2 Teilen: Parity–Wort & Informations–Wort.
(In der Tabelle durch einen Punkt getrennt.)

Linearer Code: Die Summe (modulo 2) zweier Code–Wort ergibt wieder ein Code–Wort. Diese Eigenschaft wird mit „Abgeschlossenheit“ bezeichnet.

2.5.2 Addition und Multiplikation modulo 2

Addition $\oplus$ und Multiplikation $\otimes$ modulo 2 sind gemaß folgenem Schema auszuführen.

$\oplus$	0	1	$\otimes$	0	1
0	0	1	0	0	0
1	1	0	1	0	1

Wie sofort erkennbar ist, entspricht bei modulo 2 die Addition $\oplus$ einer EXOR Verknüpfung und die Multiplikation $\otimes$ einer AND Verknüpfung.

2.6 Implementierung eines Hamming Encoders & Decoders

Die Parity–Bits beim (7, 4) Hamming–Code werden gemäß folgender Vorschrift gebildet, Bild 2.5.

$$\wp_1 = i_1 \oplus i_2 \oplus i_3 \tag{2.15}$$
$$\wp_2 = i_2 \oplus i_3 \oplus i_4 \tag{2.16}$$
$$\wp_3 = i_1 \oplus i_2 \oplus i_4 \tag{2.17}$$

Damit ergibt sich eine Encoder–Struktur nach Bild 2.5. In diesem Beispiel sind die Informations–Bits i_1, i_2, i_3, i_4 benannt und die Parity–Bits $\wp$ hinter den Informations–Bits angeordnet.

Der Hamming–Decoder, Bild 2.6, empfängt in diesem Fall ein 7-Bit Wort $(i'_1, i'_2, i'_3, i'_4, \wp'_1, \wp'_2, \wp'_3)$ (ggf. mit Fehler, daher mit $'$ versehen) und berechnet daraus (modulo 2):

$$s_1 = \wp_1' \oplus i_1' \oplus i_2' \oplus i_3' \tag{2.18}$$
$$s_2 = \wp_2' \oplus i_2' \oplus i_3' \oplus i_4' \tag{2.19}$$
$$s_3 = \wp_3' \oplus i_1' \oplus i_2' \oplus i_4' \tag{2.20}$$

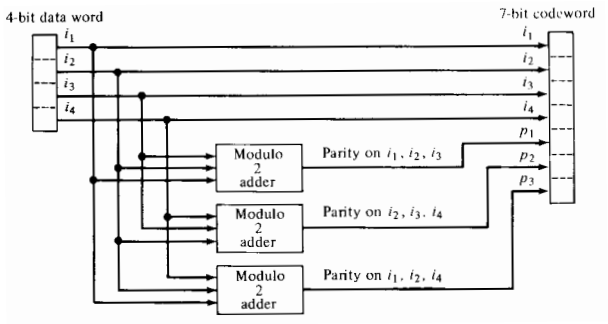

Bild 2.5: Encoder für den (7, 4) Hamming–Code

Das Bitmuster (s_1, s_2, s_3) wird **Syndrom** genannt und ist (000) falls kein Fehler aufgetreten ist. Das Syndrom ist unabhängig von den aktuellen Informations–Bits. Es ist nur abhängig vom Fehler–Muster. Da es 7 Fehler–Muster (für 1 Fehler) gibt, kann somit ein Fehler an jeder beliebigen Stelle korrigiert werden.

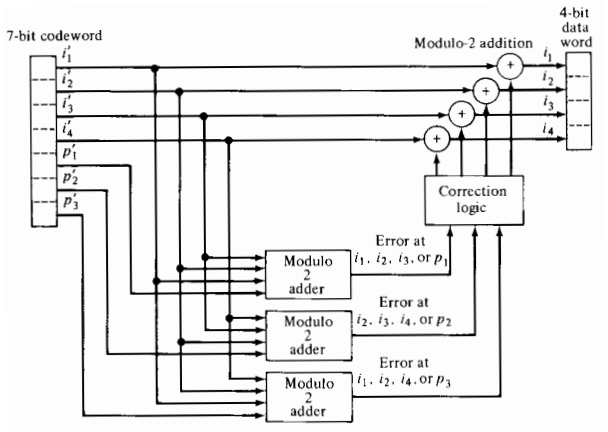

Bild 2.6: Decoder für den (7, 4) Hamming–Code

2.7 Vergleich zwischen codierter und nicht codierter Übertragung

Fehlerschutz–Codierung wird angewendet um ohne eine Erhöhung der Sende–Leistung zu einer geringeren Bitfehler–Rate zu gelangen. Bei Störungen durch weißes Gauß förmiges Rauschen ließe sich zwar die Bitfehler–Rate entsprechend zur Q–Funktion durch Erhöhung der Bit–Energie E_b beliebig herabsetzen. Aber es gibt einige wichtige Anwendungen, wo genau dies nicht geht.

- Störungen durch Echos im Funk–Kanal.

  Mit Erhöhung der Sendeleistung erhöht sich auch die Leistung der Echos.

- Störungen durch andere Teilnehmer in Funknetzen.

  Mit Erhöhung der Sendeleistung der einzelnen Handys erhöht sich auch die Störung in der Funkzelle.

- Satelliten–Systeme

  Eine Erhöhung der Sendeleistung erfordert den Transport größerer Massen. Dies ist teuerer als der durch die Fehlerschutz–Codierung bedingte Mehraufwand.

2.7.1 Auswirkungen der Codierung auf Bandbreite, Leistung und Daten–Rate

In Bild 2.7 sind die typischen „Wasserfall-Kurven“ (gemäß Q–Funktion) für eine uncodierte Übertragung (durchgezogene Linie) und eine codierte Übertragung (gestrichelte Linie) dargestellt.

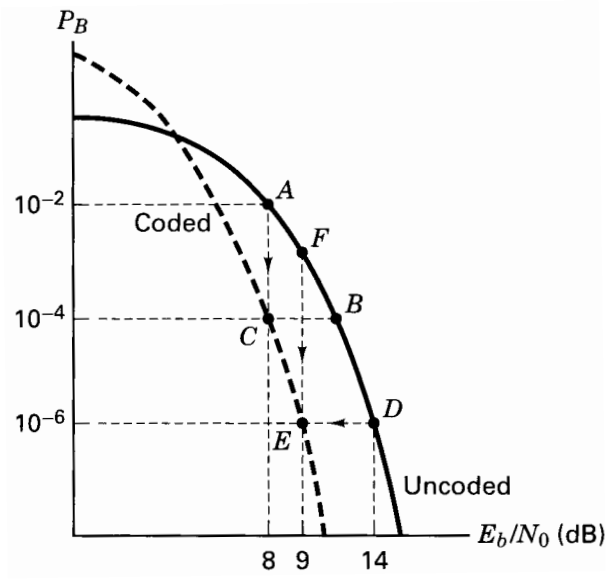

Bild 2.7: Vergleich von codierter und uncodierter Übertragung

Fehler–Rate und Bandbreite der Übertragung

Ein System habe 8 dB für E_b/N_0 zur Verfügung. Das führe bei uncodierter Übertragung auf ein BER $P_B = 10^{-2}$ (Punkt A). Da $P_B = 10^{-4}$ erforderlich sein sollen, muß Abhilfe geschaffen werden. Theoretisch könnte man die Sendeleistung erhöhen, womit man zum Punkt B käme, der uncodiert dann $P_B = 10^{-4}$ ergeben würde. Allerdings scheidet diese Lösung aus, wenn nur 8 dB für E_b/N_0 verfügbar sind.

Die nun erforderliche Codierung erhöht die Anzahl der insgesamt zu übertragenden Bits. Wenn die gesamte Übertragungszeit für die Nachricht dadurch nicht vergrößert werden darf, muß die Bitrate erhöht werden. Dies erfordert eine Vergrößerung der Bandbreite.

Leistung und Bandbreite der Übertragung

Ein System arbeite im Punkt D bei $E_b/N_0 = 14$ dB und erreiche damit $P_B = 10^{-6}$. Da der Aktions–Radius des Systems (bei konstanter Sendeleistung) vergrößert werden soll, wird eine Fehlerschutz–Codierung hinzugefügt, wodurch der Punkt E erreicht wird. Der Preis dafür ist wieder eine Erhöhung der Bandbreite.

Alternativ könnte man auch mit der alten Datenrate übertragen, wenn keine Echtzeitübertragung notwendig ist. Der Preis ist dann eine Verlangsamung der Informations–Übertragung.

Daten–Rate und Bandbreite

Ein System, das im Punkt D arbeitet, soll so geändert werden, daß die Datenrate R vergrößert wird. Da die Bitenergie $E_b = P_r \cdot T_b$ ist, also die Empfangs–Leistung P_r mal der Bit–Dauer T_b, folgt:

$$\frac{E_b}{N_0} = \frac{P_r \cdot T_b}{N_0} = \frac{P_r}{N_0} \cdot \frac{1}{R} \quad \text{mit} \quad R = 1/T_b \tag{2.21}$$

Wenn nun also die Datenrate R erhöht wird, erniedrigt sich die Bit–Energie E_b und der Arbeitspunkt verschiebt sich z.B. nach dem Punkt F, was im uncodierten Fall zu einer unzureichenden BER führt. Wird codiert übertragen, kommt man zum Punkt E und erreicht wieder die vorherige BER, obwohl durch die Codierung die Datenrate R zusätzlich weiter erhöht wird.

Der Preis für die erhöhte Datenrate ist auch hier eine Vergrößerung der Bandbreite.

2.7.2 Codierungs–Gewinn

Das vorige Beispiel eignet sich zur Definition des Codierungs–Gewinns: durch die Hinzufügung von zusätzlichen (Fehlerschutz–) Bits ergibt sich (ohne Ausnutzung der Redundanz) eine Verschlechterung der BER, denn das einzelne Bit enthält weniger Energie, weil die Datenrate größer wird. Und nur wenn die damit verbundene Redundanz bei der Decodierung ausgewertet wird und dabei auf einen besseren Wert für die BER führt, ergibt sich ein Codierungs–Gewinn, Bild 2.8.

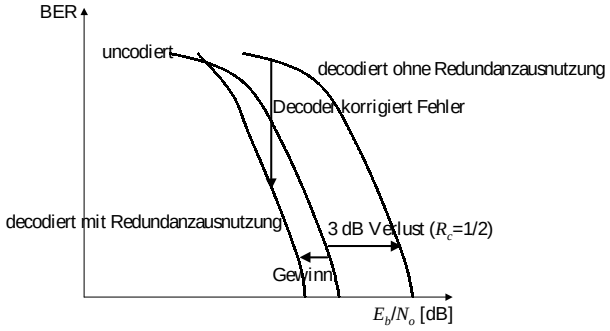

Bild 2.8: Zur Definition des Codierungs–Gewinns

Damit erklärt sich auch, weshalb die BER Kurve für den codierten Fall bei kleinen Werten von E_b/N_0 oberhalb der BER Kurve für den uncodierten Fall liegt. In diesem Fall kann der Verlust durch die zusätzlichen Bits nicht mehr ausgeglichen werden, da auch die Fehlerschutz–Bits selbst fehlerhaft empfangen werden. Der Decoder wird förmlich überschwemmt von fehlerhaften Bits. Dies zeigt sich im Falle einer BPSK Übertragung, für die BER P_B für den uncodierten Fall und für zwei verschieden lange Block–Codes dargestellt ist, Bild 2.9. Ein Gewinn ergibt sich erst oberhalb einer Schwelle von $E_b/N_0 \approx 5.5$ dB.

Beispiel für den Codierungs–Gewinn

Die Zusammenhänge lassen sich an einem Beispiel verdeutlichen. Es sei eine BPSK Übertragung mit einer Datenrate $R_u = 4800$ Bit/sec gegeben, die (nur) durch weißes Gauß–förmiges Rauschen gestört sei. Das empfangsseitige Signal–zu–Geräusch Verhältnis sei $S/N_0 = P_r/N_0 = 43.776$. Zur Codierung werde ein (15, 11) Hamming–Code angenommen, der 1 Fehler pro 15 Bit Block korrigieren kann.

Dann sind die Fehlerwahrscheinlichkeiten:

$$p_u = \sqrt{\frac{2E_b}{N_0}} \quad \text{uncodiert;} \quad p_c = \sqrt{\frac{2E_c}{N_0}} \quad \text{codiert} \tag{2.22}$$

Uncodiert ergibt sich:

$$\frac{E_b}{N_0} = \frac{S}{R_u N_0} = 9.12 \; (\equiv 9.6\, dB) \tag{2.23}$$

$$p_u = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) = Q(\sqrt{18.24}) = 1.02 \cdot 10^{-6} \tag{2.24}$$

Die Wahrscheinlichkeit P_M^u dafür, daß ein nicht codierter Block fehlerhaft empfangen wird ist 1 minus dem Produkt aller Wahrscheinlickeiten, daß jedes Bit fehlerfrei empfangen wurde.

$$P_M^u = 1 - (1 - p_u)^k = 1 - (1 - p_u)^{11} = 1.12 \cdot 10^{-6} \tag{2.25}$$

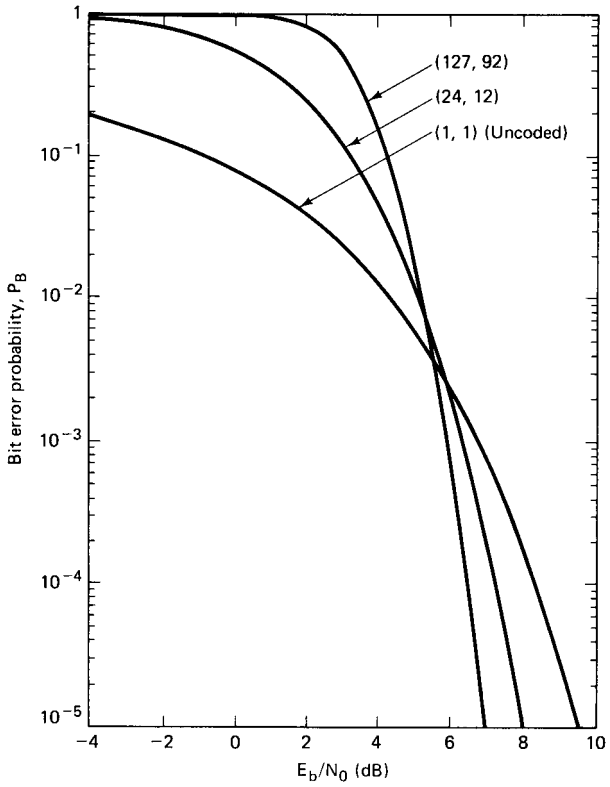

Bild 2.9: BER P_B für BPSK, uncodiert und (n, k) Block–Codes (24, 12) und (127, 92)

Mit Codierung erhöht sich die Bitrate um den Faktor $\frac{15}{11} = 1.36$ auf $R_c = 6545$ Bit/sec. Damit ergibt sich folgende Bit–Fehler–Wahrscheinlichkeit (gleiche Sendeleistung vorausgesetzt):

$$\frac{E_c}{N_0} = \frac{S}{R_c N_0} = 6.688\ (\equiv 8.25\,dB) \tag{2.26}$$

$$p_c = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) = Q(\sqrt{13.38}) = 1.36 \cdot 10^{-4} \tag{2.27}$$

Wie zu erwarten war, ergeben sich insgesamt mehr Bit–Fehler, da aufgrund der zusätzlichen Fehlerschutz–Bits die Energie pro Bit nun geringer ist.

Für die Wahrscheinlichkeit, daß ein codierter Block mehr als 1 Fehler (der ja korrigiert wird) aufweist, gewinnt man mit Gleichung (2.12). Damit ergibt sich:

$$P_M^c = \sum_{j=2}^{15} \binom{15}{j} p_c^j (1-p_c)^{15-j} \approx \binom{15}{2} p_c^2 (1-p_c)^{13} = 1.94 \cdot 10^{-5} \tag{2.28}$$

Damit verbessert sich durch diese Codierung die Fehlerwahrscheinlichkeit um den Faktor 58.

3 Leistungsfähigere Block–Codes

Aufgrund von Fehlerstatistiken ist bekannt, daß Fehler häufig in Form von Bursts auftreten. Hierzu sei folgendes Beispiel betrachtet.

Es sollen 1000 Bits im Fall a) mit einem (hypothetischen) $(2000, 1000)$ Code (Coderate $R_c = 1/2$) geschützt werden. Dieser sei in der Lage, Fehler bzw. Burstfehler bis zu (insgesamt) 100 Bits zu korrigieren.

Im Fall b) werden die Bits in 10 Blöcken à 100 Bits mit einem $(200, 100)$ Code (Coderate ebenfalls $R_c = 1/2$) geschützt. Der Code b) sei in der Lage, Fehler bzw. Burstfehler bis zu 10 Bits zu korrigieren.

Der Code im Fall b) ist nur dann gleichwertig, falls pro Datenblock tatsächlich nicht mehr als 10 Fehler auftreten, was aber als Sonderfall anzusehen ist. Dagegen ist die Verteilung von 100 Fehlern im Fall a) beliebig.

Für Block–Codes gilt somit (vergleiche auch Bild 2.9):

- Gute Codes haben große Block–Längen.
- Sehr gute Codes haben sehr große Block–Längen, da mit steigender Länge die prozentuale Anzahl der verwendeten Code–Worte abnimmt (bei konstanter Code–Rate R_c) und sich damit die Hamming–Distanz erhöht.

Ein Block–Code ist damit gekennzeichnet durch:

- Block–Länge n
- Informations–Länge k
- Hamming–Distanz d_H

3.1 Computer–Suche nach Codes

Die Problemstellung läßt sich eigentlich ganz einfach beschreiben.

- Gesucht seien die Code–Worte (Code–Vektoren) für einen (n, k) (Binär–) Code.
- Das sind 2^k Code–Worte mit insgesamt $n \cdot 2^k$ Binär–Symbolen.
- Dazu gibt es $2^{n \cdot 2^k}$ Möglichkeiten, diese Binär–Symbole auszuwählen.
- Somit gibt es (theoretisch) insgesamt $2^{n \cdot 2^k}$ (n, k) Codes.

Allerdings sind davon viele Code–Worte wertlos, z.B. weil sie identisch sind oder die Hamming–Distanz zu gering ist. Eine (planlose) Suche per Computer müßte jedoch alle Möglichkeiten durchprüfen, was auf eine astronomisch große Vielfalt führt. So führt ein $(n, k) = (40, 20)$ Code (mit mäßiger Leistungsfähigkeit) auf eine Zahl von ca. $10^{10^7} = 10^{10000000}$ zu prüfender Möglichkeiten, was erneut zeigt, daß eine planlose Suche witzlos ist.

Auch die Zahl der 2^k (zulässigen) Code–Worte (Code–Vektoren) ist bereits für diesen $(n, k) = (40, 20)$ Code sehr groß, nämlich 1 048 576 $\approx 10^6$. Diese lassen sich auch in einer Code–Tabelle nicht mehr vorrätig halten. Erst recht geht das nicht mehr bei größeren Codes, wie z.B. dem $(127, 92)$ Code mit $2^{92} \approx 5 \cdot 10^{27}$ zulässigen Code–Vektoren.

3.2 Lineare und systematische Codes

Durch die Beschränkung auf lineare und systematische Codes ergibt sich die Möglichkeit ohne riesige Codierungs–Tabellen auszukommen. Man betrachet hierfür zunächst die (Reihen–) Vektoren

$$\mathbf{m} = [m_0, m_1, \cdots, m_{k-1}] \quad \wp = [\wp_0, \wp_1, \cdots, \wp_{n-k-1}] \quad \mathbf{c} = [c_0, c_1, c_2, \cdots, c_{n-1}] \tag{3.1}$$

Die Vorgehensweise ist dabei ganz entsprechend zu der in den Bildern 2.5 und 2.6 gezeigten Anodnung, Bild 3.1, wird jedoch softwaremäßig ausgeführt.

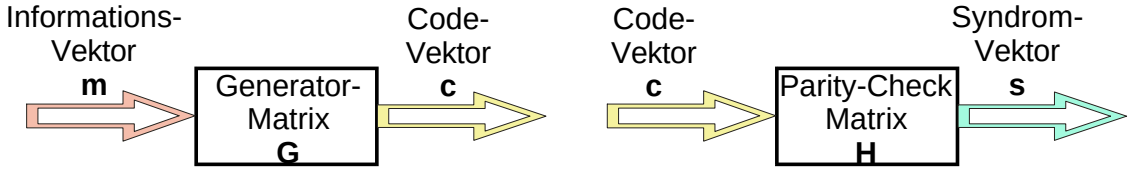

Bild 3.1: Gewinnung des Code–Wortes **c** und des Syndroms **s** mit Generator–Matrix **G** bzw. Parity–Check Matrix **H**

3.2.1 Die Generator–Matrix

Da der Code systematisch ist, besteht der Code–Vektor $\mathbf{c}$ aus dem Informations–Vektor $\mathbf{m}$ und dem Parity–Vektor $\wp$. Davon muß aber nur der Parity–Vektor $\wp$ berechnet werden, womit sich die Berechnung erheblich vereinfacht. In Matrizen–Notation wird

$$\wp = \mathbf{mP} \tag{3.2}$$

Hierbei ist $\mathbf{P}$ eine (für den jeweiligen Code charakteristische) $k \cdot (n-k)$ Koeffizienten–Matrix, deren Elemente p_{ij} 0 oder 1 sind.

$$\mathbf{P} = \begin{pmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{pmatrix} \tag{3.3}$$

Der Code–Vektor $\mathbf{c}$ läßt sich bei systematischen Codes in die Teile $\mathbf{m}$ und $\wp$ unterteilen.

$$\mathbf{c} = [\wp \vdots \mathbf{m}] \tag{3.4}$$

Mit Gleichung (3.2) wird daraus, wenn $\mathbf{m} = \mathbf{m} \cdot \mathbf{I_k}$ gesetzt wird:

$$\mathbf{c} = \mathbf{m}[\mathbf{P} \vdots \mathbf{I_k}] \tag{3.5}$$

$\mathbf{I_k}$ ist eine $k \cdot k$ Einheits–Matrix.

$$\mathbf{I_k} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{3.6}$$

Mittels Gleichung (3.5) läßt sich nun die $k \cdot n$ Generator–Matrix $\mathbf{G}$ definieren.

$$\mathbf{G} = [\mathbf{P} \vdots \mathbf{I_k}] \tag{3.7}$$

Damit läßt sich das Code–Wort $\mathbf{c}$ mit der Generator–Matrix $\mathbf{G}$ berechnen.

$$\mathbf{c} = \mathbf{mG} \tag{3.8}$$

3.2.2 Die Parity–Check Matrix

Die Parity–Check Matrix $\mathbf{H}$ ist eine $(n-k) \cdot n$ Matrix der folgenden Form:

$$\mathbf{H} = [\mathbf{I_{n-k}} \vdots \mathbf{P^T}] \tag{3.9}$$

$\mathbf{P^T}$ ist eine $(n-k) \cdot k$ Matrix und die Transponierte der Koeffizienten–Matrix $\mathbf{P}$ und $\mathbf{I_{n-k}}$ ist eine $(n-k) \cdot (n-k)$ Einheits–Matrix.

Die Multiplikation der Parity–Check Matrix mit der Transponierten der Generator Matrix liefert, da die Multiplikation mit einer Einheitsmatrix (geeigneter Dimension) die zu multiplizierende Matrix unverändert läßt:

$$\mathbf{HG^T} = [\mathbf{I_{n-k}} \vdots \mathbf{P^T}] \begin{bmatrix} \mathbf{P^T} \\ \cdots \\ \mathbf{I_k} \end{bmatrix} = \mathbf{P^T} \oplus \mathbf{P^T} = \mathbf{0} \tag{3.10}$$

$\mathbf{0}$ ist eine $(n-k) \cdot k$ Null–Matrix (d.h. alle Elemente sind 0).

Entsprechend gilt:

$$\mathbf{GH^T} = \mathbf{0} \tag{3.11}$$

Wird nun Gleichung (3.8) von rechts mit $\mathbf{H^T}$ multipliziert, folgt mit Gleichung (3.11):

$$\mathbf{c}\mathbf{H}^{\mathbf{T}} = \mathbf{m}\mathbf{G}\mathbf{H}^{\mathbf{T}} = \mathbf{0} \tag{3.12}$$

Gleichung (3.12) beschreibt die Parity–Check Kontrolle auf der Empfangsseite. Ist kein Fehler entstanden, ist der Syndrom–Vektor $\mathbf{s} = \mathbf{0}$ ein Null–Vektor der Länge k.

3.2.3 Fehlermuster und Syndrom

Ein fehlerhafter Empfangs–Vektor $\mathbf{r}$ kann als Summe des korrekten Vektors $\mathbf{c}$ und eines Fehlervektors $\mathbf{e}$ dargestellt werden.

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{e} \tag{3.13}$$

Der Fehler–Vektor hat an der Position eine 1 wo ein Fehler auftrat. Bei der Berechnung des Syndroms ergibt sich mit Gleichung (3.12):

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathbf{T}} = (\mathbf{c} \oplus \mathbf{e})\mathbf{H}^{\mathbf{T}} = \mathbf{c}\mathbf{H}^{\mathbf{T}} \oplus \mathbf{e}\mathbf{H}^{\mathbf{T}} = \mathbf{0} \oplus \mathbf{e}\mathbf{H}^{\mathbf{T}} = \mathbf{e}\mathbf{H}^{\mathbf{T}} \tag{3.14}$$

Das Syndrom $\mathbf{s}$ ist also nicht vom aktuellen Code–Wort abhängig, sondern nur von der Position des Fehlers. Es ist ein Vektor, der nur an der Position des Fehlers eine 1 enthält und sonst überall Nullen. Damit folgt für die Korrektur des Fehlers:

$$\mathbf{c} = \mathbf{r} \oplus \mathbf{s} \tag{3.15}$$

3.2.4 Charakteristische Matrizen für den (7, 4) Hamming–Code

Die Block–Codes sind durch ihre charakteristischen Matrizen definiert. Für den (7, 4) Hamming–Code sind dies:

$$\mathbf{G} = \left(\begin{array}{cccccc} 1 & 1 & 0 \vdots 1 & 0 & 0 & 0 \\ 0 & 1 & 1 \vdots 0 & 1 & 0 & 0 \\ 1 & 1 & 1 \vdots 0 & 0 & 1 & 0 \\ 1 & 0 & 1 \vdots 0 & 0 & 0 & 1 \end{array}\right) \quad \mathbf{H} = \left(\begin{array}{cccccc} 1 & 0 & 0 \vdots 1 & 0 & 1 & 1 \\ 0 & 1 & 0 \vdots 1 & 1 & 1 & 0 \\ 0 & 0 & 1 \vdots 0 & 1 & 1 & 1 \end{array}\right) \tag{3.16}$$

Literatur

[1] Sklar, B.: *Digital Communications, Fundamentals and Applications*, Prentice Hall, 1988

[2] Sklar, B.: *Digital Communications, Fundamentals and Applications*, Prentice Hall, 2nd ed. 2001

[3] Blahut, R.E.: *Theory and Practice of Error Control Codes*, Addison Wesley, 1983

[4] Blahut, R.E.: *Digital Transmission of Information*, Addison Wesley, 1990

[5] Haykin, S.: *Communication Systems*, Wiley, 2001

[6] Simon, M.K.; Hinedi, S.M.; Lindsey, W.C.: *Digital Communication Techniques, Signal Detection and Detection*, Prentice Hall, 1994

[7] Proakis, J.G.: *Digital Communications*, 2nd ed. 1989

[8] Proakis, J.G.; Salehi, M.: *Communication Systems Engineering*, Prentice Hall, 1994

[9] Ziemer, R.E.; Peterson, R.L.: *Introduction to Digital Communication*, McMillan, 1992

[10] Taub, H.; Schilling, D.L.: *Principles of Communication Systems*, 2nd ed. 1989

[11] Lee, E.A.; Messerschmitt, D.G.: *Digital Communication*, Kluver 1988