

Prof. Dr.-Ing. Dr. h. c. J. Becker

becker@kit.edu

Karlsruher Institut für Technologie (KIT)

Institut für Technik der Informationsverarbeitung (ITIV)

Digitaltechnik

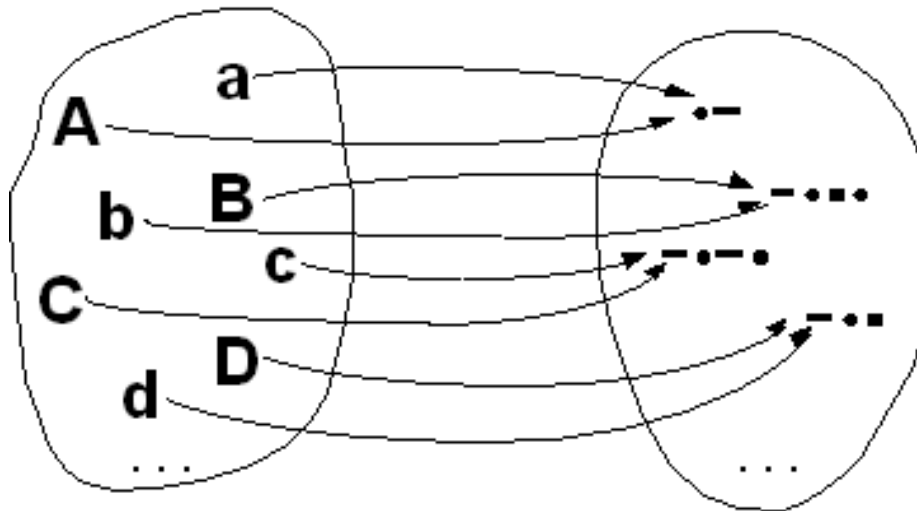
Codierung und Fehlerkorrektur

Allgemein: **Code** ist Vorschrift für **eindeutige Zuordnung** → **Codierung**
→ Zuordnung muss **nicht umkehrbar eindeutig** sein

Beispiel: **Zuordnung von Zeichen verschiedener Alphabete**

Zeichenvorrat (Urmenge)

anderer Zeichenvorrat (Bildmenge)



■ Digitaltechnik: **Abstraktion** der Werte “L” und “H” durch Symbole “0” und “1”

- **Codewörter** sind elementare Einheiten zur **Darstellung** von Informationen

Beispiel:

<u>Codewort</u>	<u>Anzahl Binärstellen</u>
10111010	m=8
10111	m=5

- **Anzahl** möglicher **Codewörter** mit m Binärstellen: 2^m
- Maximale **Anzahl** N von m -stelligen **Codewörtern**: $N \leq 2^m$
- **Anzahl** strukturierter m -stelliger **Codewörter** mit k Einsen: $\binom{m}{k} = \frac{m!}{k!(m-k)!}$

Allgemein gilt: Codes legen fest, wie Codewörter zu **interpretieren** sind

Beispiel: Strukturierter Code

→ Anzahl Wörter im (2 aus 5)-Code:
$$N = \binom{5}{2} = \frac{5!}{2!(5-2)!} = 10$$

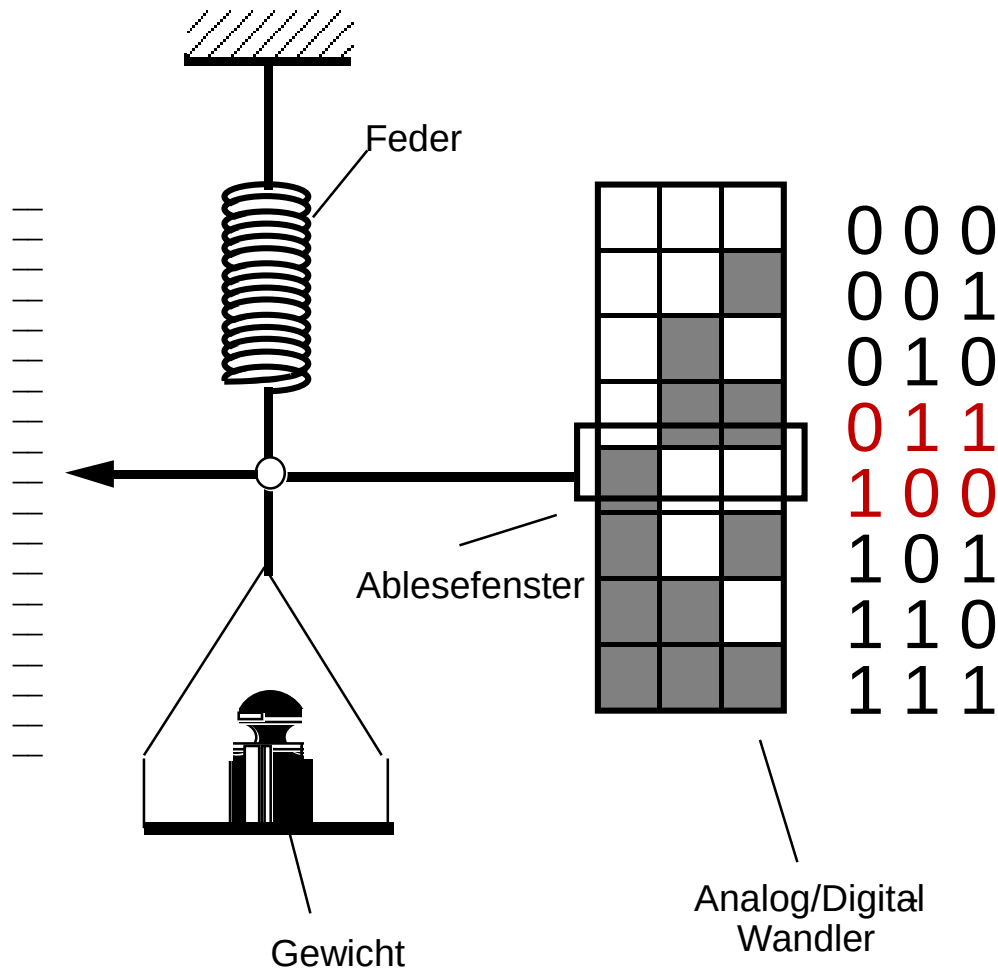
■ $m = 5$ $k = 2$

■ Folgende **Codewörter** sind möglich:

000 11	00 110	0 1010	10001	10100
00 101	0 1001	0 1100	10010	11000

■ Der **(2 aus 5)-Code** eignet sich mit seinen **10 Codewörtern** besonders zur Darstellung von **Dezimalziffern**

Wandlung von stetigen Signalen in zusammengesetzte Binärsignale



Aufgabe:

Festlegung einer
geeigneten Zuordnung:

Stetiger Wertebereich



Binärer Wertebereich

Wandlung: stetige (analoge) Signale in Binärsignale (Binärvektoren)

■ Problem: Übergänge, bei denen sich **mehr** als **eine Binärstelle** ändert

■ Beispiel:

0	1	1
↓	↓	↓
1	0	0

3 Binärstellen-
übergänge

■ Beim **Wechsel** müssen **nicht alle** Binärstellen **gleichzeitig** wechseln

■ Dadurch: **große Abweichungen** für Werte der **Binärsignale** möglich:

0	1	1
↓	↓	↓
111, 001, 010, 101, 110 oder 000		
↓	↓	↓
1	0	0

Fehlerhafte Über-
gänge in andere
Codewörter möglich

Eigenschaften von/zwischen Codewörtern

■ Definition: **Hammingdistanz HD** (auch **Hammingabstand** genannt)

Seien die Codewörter $CW_i, CW_j \in \{0, 1\}^n$,

HD_{ij} = Anzahl der Stellen, an denen sich CW_i und CW_j unterscheiden.

Dann heißt **Hd_{ij}** die **Hammingdistanz** von CW_i und CW_j .

■ Also: Die **Hammingdistanz HD** zwischen **zwei** gleich langen Codewörtern gibt die **Anzahl der unterschiedlichen Binärstellen** an.

Beispiele: **Bestimmung HD**

011	100	HD = 3
100	101	HD = 1
1100	0010	HD = 4

■ Definition: Minimale Hammingdistanz $HD_{\min}$

Sei $X \subseteq \{0, 1\}^n$ beliebig, und

$$Hd_{\min}(X) = \min\{ Hd_{ij} \mid CW_i, CW_j \in X \wedge CW_i \neq CW_j \}$$

Dann heißt $Hd_{\min}(X)$ **minimale Hammingdistanz** von X

■ Beispiel: Codewörter des **(2 aus 5)-Codes** haben folgende minimale Hammingdistanz: $Hd_{\min}(\text{2 aus 5}) = 2$

- Die **minimale Hammingdistanz** ist eine entscheidende Eigenschaft eines Codes, um **Übertragungsfehler erkennen** und **korrigieren** zu können

Begriff: **Einschrittige Codes**

- Codes, bei denen zwei **benachbarte Codewörter** immer eine **Hammingdistanz** von **eins** haben heißen **einschrittig**.

Spezielle Codes: Gray-Code

Code zur Analog/Digital-Wandlung

Einschrittige Codes: besondere Rolle bei der **Analog-Digital Wandlung**

Beispiel: Problem der digitalisierten Waageablesung

→ **Änderung einer Binärstelle** geschieht auf jeden Fall immer „gleichzeitig“

→ **Wandlungsfehler** von **höchstens 1** in kleinster Messeinheit

■ Wichtiger Vertreter der **einschrittigen Codes**: **Gray-Code**

linear, m=4

0: 0000
1: 0001
2: 0011
3: 0010
4: 0110
5: 0111
6: 0101
7: 0100
8: 1100
9: 1101

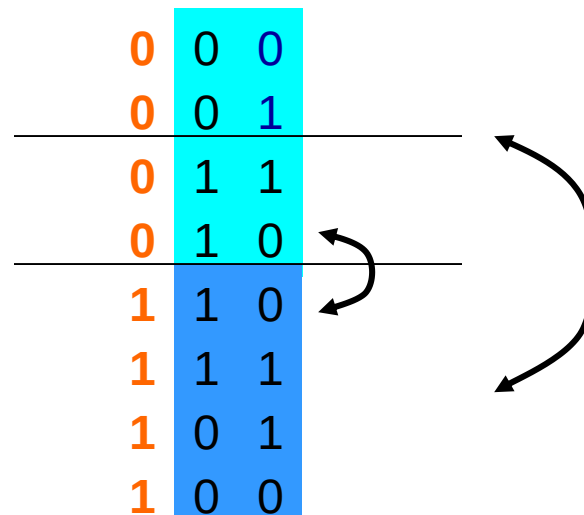
zyklisch, m=3

0: 000
1: 001
2: 011
3: 010
4: 110
5: 111
6: 101
7: 100
0: 000
...

Spezielle Codes: Gray-Code

Konstruktion des Gray-Codes

- Geg.: Gray-Code mit $m = x$ Binärstellen liegt vor
- Doppelt so langen **Gray-Code** mit $m_{\text{neu}} = x+1$ Binärstellen erzeugen:
 - in umgekehrter Reihenfolge an **gegebenen Code anhängen**
(**Spiegelung** an der **Horizontalen**)
 - und **zusätzliche Binärstelle** anfügen
(zuerst $2^{x+1}/2$ Nullen/Einsen dann $2^{x+1}/2$ Einsen/Nullen)



Spezielle Codes: Austauschcodes

Notwendig: **Datenaustausch** zwischen **digitalen Systemen**:

- > spezielle Codes werden benötigt
- > **Texte** aus Buchstaben, Ziffern, Satzzeichen, Sonderzeichen (*characters*) **übertragbar**

ASCII-Code

- Verbreitung: **ASCII-Code** (American Standard Code for Information Interchange)
→ kodiert mit **7 Bit 128 Zeichen**: reicht für englischen Sprachraum weitgehend aus
- **Bitgruppen** werden zusammengefasst -> spezielle Bedeutung / Namen:
 - **MSB** (Most Significant Bits) bezeichnet **3 höherwertigen** Bits
 - **LSB** (Least Significant Bits) bezeichnet **4 niederwertigen** Bits
- Zuordnung: **Zeichen** <-> **Codewort**
→ **MSB** teilt alle Zeichen in **Gruppen** zu je **16 Zeichen** ein
(lexikographische Anordnung von **Zeichen + Codewörtern**)

Spezielle Codes: ASCII-Code

LSB	MSB							
Binär	000	001	010	011	100	101	110	111
	Steuerzeichen				Großbuchstaben		Kleinbuchstaben	
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	„	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Spezielle Codes: Austauschcodes

Beispiel: ASCII-Codierung

■ Buchstabe "A":

$A = 41_H = 100\ 0001_B$
 / \
 MSB LSB

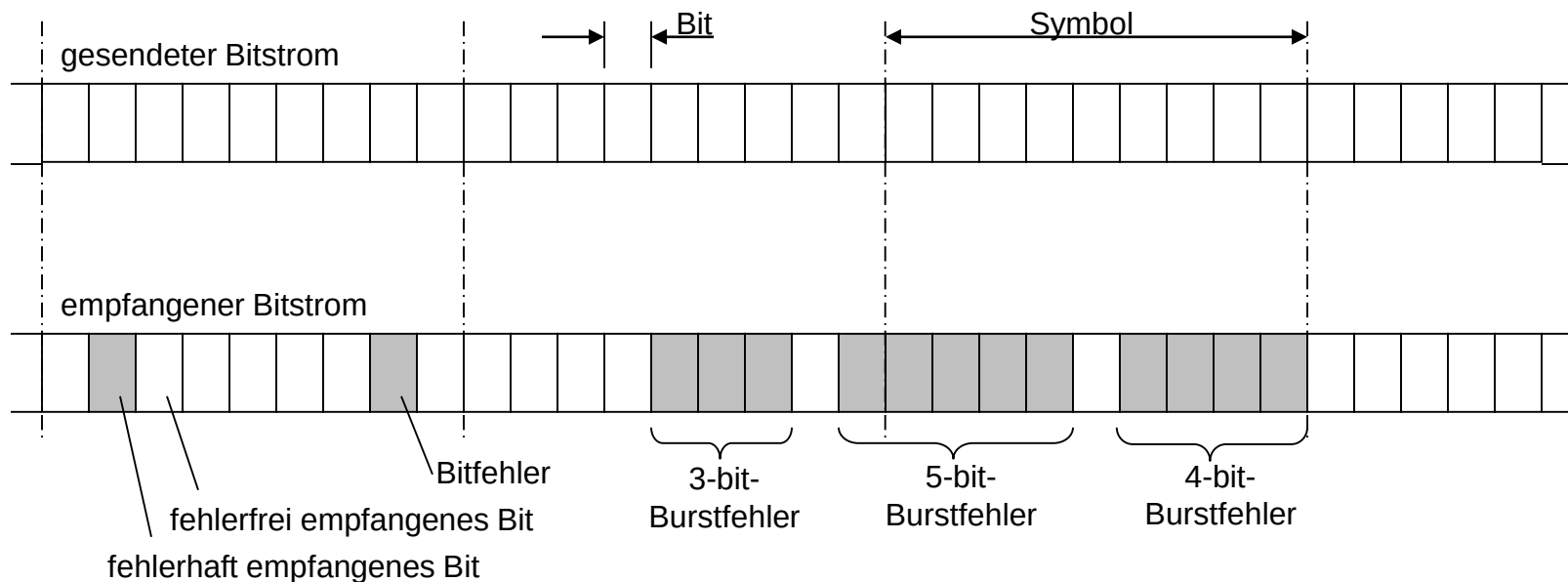
(Bitgruppen analog zu Hexadezimalsystem / Kapitel 4.6)

- Praxis: verschiedene 8, 16 und sogar 32 Bit Codes in Gebrauch
→ Spezialzeichen verschiedener Sprachen darstellbar

■ Weitere Beispiele:

- verbreitetster **16-Bit Austauschcode** ist der **UNICODE**
→ 65536 Zeichen darstellbar
→ ASCII-Code: erste 128 Zeichen des **UNICODES**
→ enthält Erweiterungen für viele Sprachen
- **7-Segment-Code**: digitale Ziffernanzeige, etc.
- **Punktmatrixanzeigen**: Drucker, etc.
- **OCR-Code** (Optical Character Recognition):
- **Blindenschrift**, etc.

- Fehlerschutz umfasst **Fehlererkennung** und **Fehlerkorrektur**
- **Schutzwirkung** wird durch **Codierung** realisiert
- Unterscheidung abhängig von Lage und Menge fehlerhaft empfangener Bits:
→ **Bitfehler**, **Burstfehler** und **Symbolfehler**
können innerhalb des übertragenen Bitstroms auftreten



Codes für Fehlererkennung

■ Problem in der Praxis: **Störeinflüsse** können bei der **Übertragung** oder **Speicherung** von **binär kodierten** Informationen den Wert der zur Darstellung verwendeten physikalischen Grösse **verfälschen**

■ Es gilt: **einzelne Bitfehler** (1 Bit „kippt“) sind **erkennbar** bei Codes mit **minimaler Hammingdistanz** $HD_{\min} = 2$

→ jeder denkbare **Bitfehler** führt zu **ungültigem** (d.h. unbenutztem) **Codewort**:

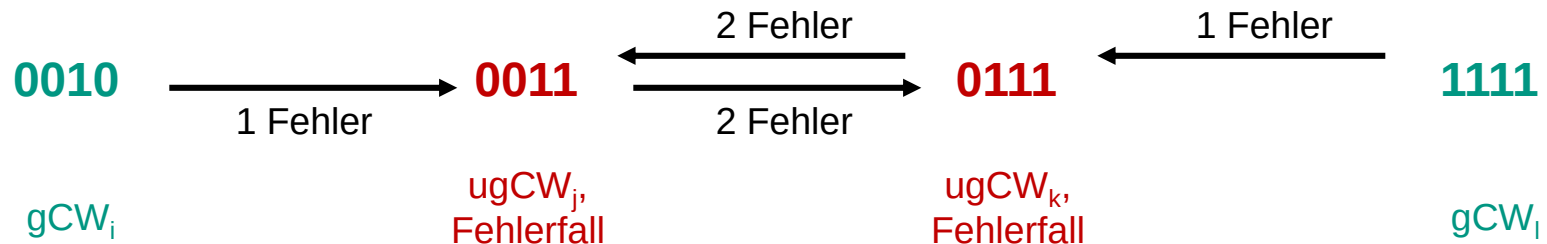


Codes für Fehlererkennung / -korrektur

Es gilt: Bei Codes mit $HD_{\min} = 3$:

→ **Zweifachfehler erkennbar** oder **Einfachfehler korrigierbar**

2-Fach Fehlererkennung:



1-Fach Fehlerkorrektur:



Allgemein gilt:

- **Geeignete Codes** können **Fehler erkennen** und sogar **korrigieren**
- **Notwendig:** Hinzufügen **zusätzlicher** (redundanter) **Informationen**
→ **zusätzlicher Darstellungsaufwand (Kosten!)**
- **Annahme:** Höchstanzahl **gleichzeitig** zu berücksichtigender **Fehler** ist fest
meistens: höchstens **1** oder **2 Bitfehler** gleichzeitig
- **(Teil-)Systematik** bei Festlegung des **Codes** notwendig
Möglichkeiten: **minimale Hammingdistanz, Paritätsbits, Blocksicherungsverfahren, Hamming-Codes, etc.**

Paritätsbit-Prüfverfahren:

- **Fehler** bei Code-Übertragung **erkennbar** (ohne Beachtung Hammingdistanz)
→ zusätzliches **Paritätsbit** anhängen:
 - im Binärwort **enthaltene Einsen** werden entweder auf **gerade** (**even parity**) oder **ungerade** (**odd parity**) Anzahl ergänzt → **Überprüfung beim Empfänger**

Fehlererkennung durch Parität

Beispiel: Ergänzung des Paritätsbit

Dezimal	Binär	gerade Parität	ungerade Parität
0	0000	0000 0	0000 1
1	0001	0001 1	0001 0
2	0010	0010 1	0010 0
3	0011	0011 0	0011 1
4	0100	0100 1	0100 0
5	0101	0101 0	0101 1
6	0110	0110 0	0110 1
7	0111	0111 1	0111 0
8	1000	1000 1	1000 0
9	1001	1001 0	1001 1

Fehler:

1001

10**1**1 **0**

10**1**1 **1**

- Das Prinzip der **Paritätssicherung** ist **zweidimensional** anwendbar
→ **Blocksicherungsverfahren** mit doppelter Quersummenergänzung
- **Nachricht** wird in **Blöcke** von n **Codewörtern mit Paritätsbit** eingeteilt
zusätzlich: am Ende jedes Blocks ein **weitere Codewort** einfügen
→ enthält alle **Paritätsbits** der **Spalten**
- Also: bei Auftreten von **Einfachfehlern** lassen sich **Spalte** und **Zeile** **eindeutig** ermitteln:
 - **Einfachfehler** sind damit **korrigierbar**
 - gleichzeitig sind noch **weitergehende Fehlererkennungsverfahren** möglich: **Bündelstörung** erkennbar
- **Bündelstörung:** zeitlich konzentrierte Fehler, d.h. über einen Zeitraum ist die Verbindung gestört, können erkannt bzw. behoben werden

Beispiel: Fehlerlokalisierung durch Blocksicherungsverfahren

Ziffer	Codewörter mit gerader Parität				
5	0	1	0	1	0
4	0	1	0	0	1
1	0	0	1	1	1
3	0	0	1	1	0
9	1	0	0	1	0
8	1	0	0	0	1
Prüfwort	0	0	1	0	1

↑
Spalte mit Fehler
(ungerade Parität)

←
Zeile mit Fehler
(ungerade Parität)

Fehlererkorrektur durch Blocksicherung

- Also: m Informationsbits werden k Fehlerschutzbits als Redundanz angehängt und übertragen
- Damit: Vergrößerung des Datenstroms auf Länge $n = m + k$
- Beispiel für Blocksicherung: zyklische Redundanzprüfung
[Cyclic Redundancy Check (CRC)]

