

Grundlagen der Informatik

Prof. Dr. J. Schmidt

Fakultät für Informatik

GDI – WS 2020/21

Codesicherung und Kanalcodierung
Fehlertolerante Codes

- Wie ist die Stellendistanz und die Hamming-Distanz eines Codes definiert?
- Was ist ein m-aus-n-Code?
- Welche charakteristischen Merkmale haben Codes mit Paritäts-Bits?



- Beim Übertragen von Nachrichten und Speichern bzw. Lesen von Daten können Fehler auftreten
- Suche nach **fehlertoleranten Codes**
 - ermöglichen es dem Empfänger zu erkennen, ob bei Übertragung ein Fehler aufgetreten ist
 - und wenn ja, diesen evtl. selbst zu korrigieren
- → gezieltes Hinzufügen von Redundanz
- **Fehlererkennende Codes**
 - Fehler kann erkannt werden
- **Fehlerkorrigierende Codes**
 - Empfänger kann erkannte Fehler korrigieren



● Stellendistanz d

- Anzahl der Binärstellen, an denen sich zwei gleich lange Codewörter x und y unterscheiden
- Für unterschiedlich lange Codewörter ist die Stellendistanz nicht definiert

● Hamming-Distanz h

- Minimale paarweise Stellendistanz eines Codes
- Maß für die Störsicherheit eines Codes



Hamming-Distanz eines Codes (2) – Beispiel

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

- Bestimmung der Hamming-Distanz durch Vergleich der einzelnen Codewörter

	a	b	c	d	e	f	g	h	i	j
a	0									
b	2	0								
c	2	2	0							
d	2	2	4	0						
e	2	4	2	2	0					
f	4	2	2	2	2	0				
g	2	2	4	2	4	4	0			
h	2	4	2	4	2	4	2	0		
i	4	2	2	4	4	2	2	2	0	
j	4	4	4	2	2	2	2	2	2	0

Hamming-Distanz = 2



Hamming-Distanz eines Codes (3)

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

- Für eine gegebene Hamming-Distanz h gilt
 - Sind maximal $h - 1$ Bit in einem Wort fehlerhaft, so kann dies erkannt werden
 - Sind maximal $(h - 1)/2$ Bit fehlerhaft, so können diese Fehler korrigiert werden

oder anders formuliert

- Hat ein Code die Hamming-Distanz h , können alle Fehler
 - erkannt werden, die weniger als h Bits betreffen ODER
 - korrigiert werden, die weniger als $h/2$ Bits betreffen



Hamming-Distanz eines Codes (4)

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

- $h=1$
 - Fehlerhafte Binärstellen können nicht erkannt werden (Bsp.: ASCII-Code)
- $h=2$
 - 1-Bit-Fehler können erkannt, aber nicht korrigiert werden
- $h=3$
 - 1-Bit-Fehler können korrigiert werden ODER
 - 1-Bit- und 2-Bit-Fehler können erkannt, aber nicht korrigiert werden
- $h=4$
 - 1-Bit-Fehler können korrigiert und 2-Bit-Fehler erkannt werden ODER
 - 1-Bit-, 2-Bit- und 3-Bit-Fehler können erkannt, aber nicht korrigiert werden



Hamming-Distanz eines Codes (5)

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

In Abhängigkeit von der Anzahl **n** von falsch übertragenen Bits, die bei einem Code automatisch erkannt bzw. korrigiert werden können, spricht man von einem

n-erkennenden bzw. **n-korrigierenden** Code.



Hamming-Distanz eines Codes (6) – Beispiel

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

- Bestimmung der Hamming-Distanz durch Vergleich der einzelnen Codewörter

	a	b	c	d	e	f	g	h	i	j
a	0									
b	2	0								
c	2	2	0							
d	2	2	4	0						
e	2	4	2	2	0					
f	4	2	2	2	2	0				
g	2	2	4	2	4	4	0			
h	2	4	2	4	2	4	2	0		
i	4	2	2	4	4	2	2	2	0	
j	4	4	4	2	2	2	2	2	2	0

Hamming-Distanz = 2

1-erkennender Code



Beispiel: Bestimmung Hamming-Distanz (1)

- Gegeben
 - Ziffern 1 bis 4 in direkter binärer Codierung
 - $1 = 001$, $2 = 010$, $3 = 011$, $4 = 100$

	001	010	011	100
001	-	-	-	-
010	2	-	-	-
011	1	1	-	-
100	2	2	3	-

- Hamming-Distanz = 1
→ Fehler nicht immer erkennbar



Beispiel: Bestimmung Hamming-Distanz (2)

- Gegeben
 - Ziffern 1 bis 4 in anderer binärer Codierung
 - $1 = 000$, $2 = 011$, $3 = 101$, $4 = 110$

	000	011	101	110
000	-	-	-	-
011	2	-	-	-
101	2	2	-	-
110	2	2	2	-

- Hamming-Distanz = 2
→ Erkennung von 1-Bit-Fehlern immer möglich



- Nur eine Teilmenge aller möglichen Codes wird verwendet
- **Block-Codes** mit einer **Wortlänge** von n
- In jedem Code-Wort kommen genau
 - m **Einsen** und
 - $n - m$ **Nullen** vor
- Spezialfall 1-aus-n-Code: „one-hot“ Codierung
- Bei gegebenem m und n gibt es genau

$$\binom{n}{m} \text{ Codewörter}$$



Beispiele von m-aus-n-Code

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

<u>Ziffer</u>	<u>2-aus-5-Code</u>	<u>1-aus-10-Code</u>
0	00011	0000000001
1	00101	0000000010
2	00110	0000000100
3	01001	0000001000
4	01010	0000010000
5	01100	0000100000
6	10001	0001000000
7	10010	0010000000
8	10100	0100000000
9	11000	1000000000



- Häufig verwendetes Verfahren zur Fehlererkennung und Fehlerkorrektur

Paritäts-Prüfung („parity check“)

- Vorgehen
 - Einführung eines Zusatz-Bits (Paritäts-Bit)
 - Anzahl der Einsen von Codewörtern werden
 - auf eine gerade Anzahl (gerade Parität, „even parity“)
 - oder ungerade Anzahl (ungerade Parität, „odd parity“)
- ergänzt.



Eindimensionale Paritäts-Prüfung

- Beispiel: 7 Bit-ASCII-Code mit Paritäts-Bit

A 10000010	G 10001110	M 10011010	S 10100110	Y 10110010
B 10000100	H 10010000	N 10011100	T 10101001	Z 10110100
C 10000111	I 10010011	O 10011111	U 10101010	
D 10001000	J 10010101	P 10100000	V 10101100	
E 10001011	K 10010110	Q 10100011	W 10101111	
F 10001101	L 10011001	R 10100101	X 10110001	



Eindimensionale Paritäts-Prüfung – Beispiel (2)

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

Einlesen eines Bitmusters und Identifizierung von einzelnen Zeichen

Geben Sie eine beliebig lange Folge von Nullen und Einsen ein:

100100001100001111011000110110001101111001000001101011111100101011001

10010000	H
11000011	a
11011000	l
11011000	l
11011110	o
01000001	(Leerzeichen)
10101111	W
11001010	e
11001000	← Fehler: ungerade Parität!



Zweidimensionale Paritäts-Prüfung

- Wird verwendet bei der Übertragung von Blöcken
- Erweiterung der eindimensionalen Paritäts-Prüfung
 - Für jedes einzelne Zeichen wird ein Paritäts-Bit verwendet
 - Nachdem der gesamte Block von Codewörtern übertragen wurde, wird noch ein weiteres Codewort übertragen, dass die Paritäts-Bits zu allen Spalten des übertragenen Blocks enthält



● Zweidimensionale Paritäts-Prüfung

1	1	0	0	0	0	1	1	a
1	1	0	0	0	1	0	1	b
1	1	0	0	0	1	1	0	c
1	1	0	0	1	0	0	1	d
1	1	0	0	1	0	1	0	e
1	1	0	0	1	1	0	0	f
1	1	0	0	1	1	1	1	g
1	1	0	1	0	0	0	1	h
0	0	0	1	0	0	0	1	

← Spalten-Paritäts-Bits

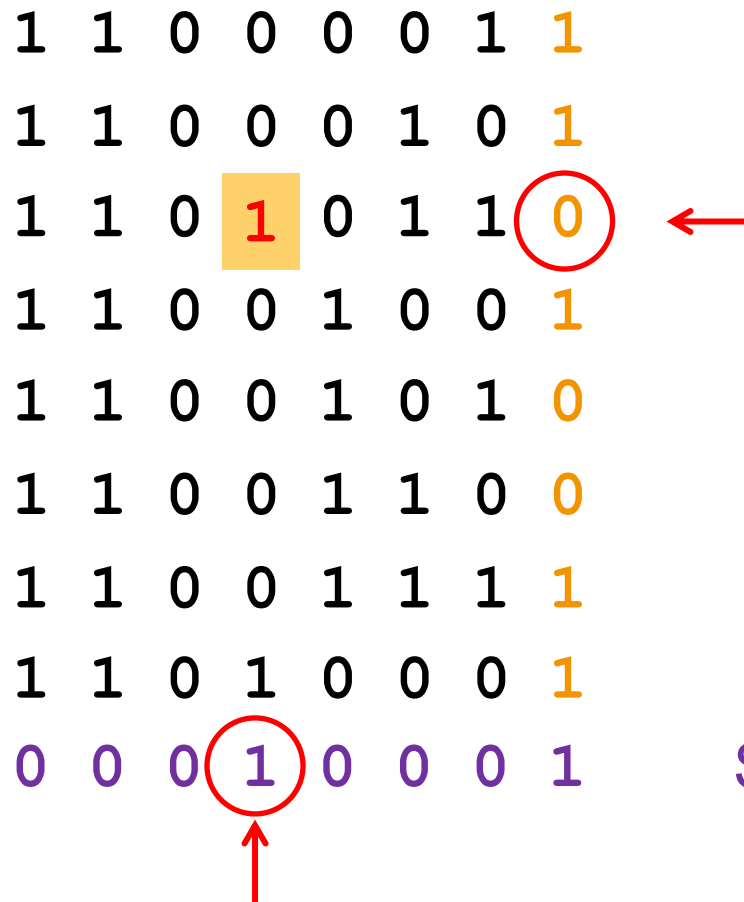


Zweidimensionale Paritäts-Prüfung

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

- Kippt bei der Übertragung **nur ein Bit**, so kann dieses bei der zweidimensionalen Paritäts-Prüfung korrigiert werden

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	0	1	0	1	0
1	1	0	0	1	1	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1



Spalten-Paritäts-Bits



Zweidimensionale Paritäts-Prüfung

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

- Kippt bei der Übertragung **nur ein Bit**, so kann dieses bei der zweidimensionalen Paritäts-Prüfung korrigiert werden

- Lokalisierung falscher Zeilen- und Spalten-Paritäts-Bits

- Spalten-Paritäts-Bits: 000**1**0001
- richtige Spalten-Paritäts-Bits: 000**0**0001

- Korrektur des 3. Wortes

- falsch: 11010110
- korrigiert: 110**0**0110 → c

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	0	1	0	1	0
1	1	0	0	1	1	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1



Zweidimensionale Paritäts-Prüfung

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

● Erkennen eines Doppelfehlers

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	0	1	1	1	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1

anhand 4
falscher
Paritäts-Bits

Korrektur nicht
möglich

Spalten-Paritäts-Bits



Zweidimensionale Paritäts-Prüfung

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

● Erkennen eines Doppelfehlers

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	0	1	0	1	0
1	1	0	1	1	1	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1

anhand 2
falscher
Paritäts-Bits

Korrektur nicht
möglich

Spalten-Paritäts-Bits



Zweidimensionale Paritäts-Prüfung

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

● Erkennen eines Dreierfehlers

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	0	1	1	1	0
1	1	0	0	1	1	0	0
1	1	1	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1

anhand 6
falscher
Paritäts-Bits

Korrektur nicht
möglich

Spalten-Paritäts-Bits



Zweidimensionale Paritäts-Prüfung

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes

● Erkennen eines Dreierfehlers

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	1	0	1	1	1	1	0
1	1	0	1	1	0	0	1
1	1	0	0	1	0	1	0
1	1	0	0	1	1	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1

anhand 2
falscher
Paritäts-Bits

Korrektur nicht
möglich

Sieht aus wie
ein Einzelfehler!

Spalten-Paritäts-Bits



● Erkennen eines Viererfehlers

1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	0	0	1	0	1	0
1	0	0	1	1	1	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	1
0	0	0	1	0	0	0	1

... ist nicht
garantiert!

Spalten-Paritäts-Bits



Code mit zweidimensionaler Paritäts-Prüfung ist

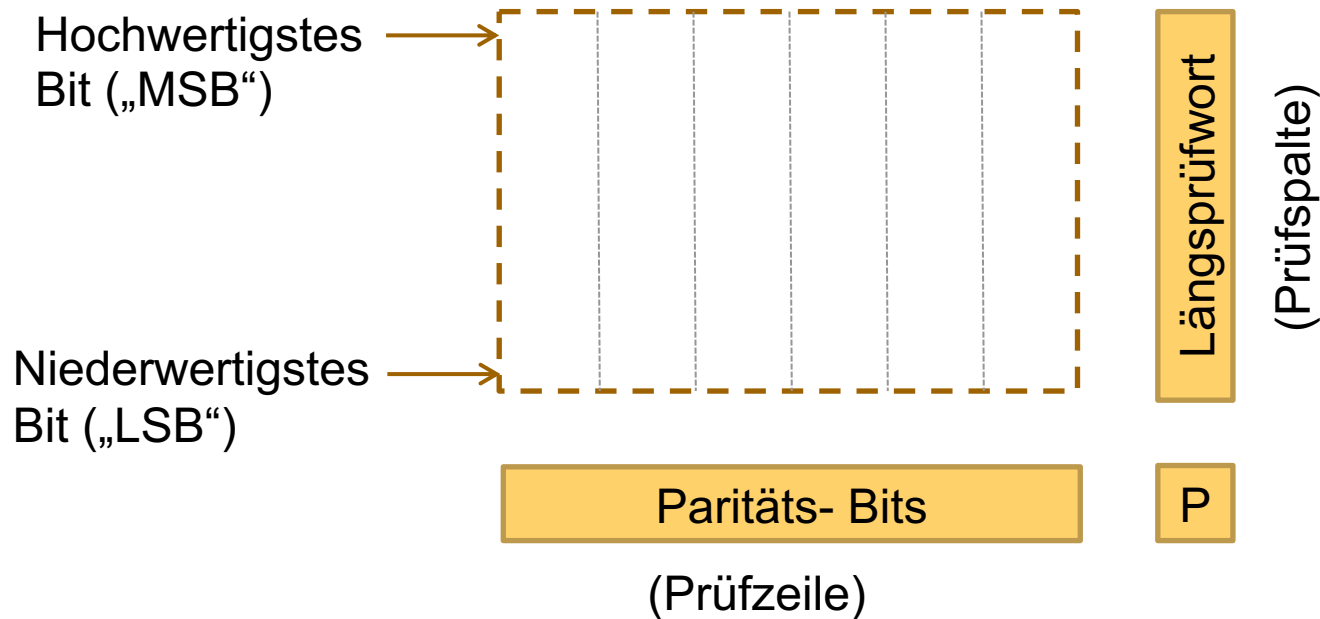
- 1-fehlerkorrigierend
 - Korrektur von Einzelfehlern und Erkennung von Doppelfehlern

ODER

- 3-fehlererkennend
 - Erkennung von Einzel-, Doppel- und Dreifachfehlern



● Andere Darstellung



● Prüfbit P

- wird üblicherweise so gesetzt, dass es die Anzahl der Einsen im gesamten Datenblock auf die gewünschte Parität ergänzt



Folgende Möglichkeiten können bei 1-Bit-Fehler auftreten

- Fehler tritt im Block auf
 - 1 Bit in Prüfzeile und Prüfspalte sind falsch
 - Position des fehlerhaften Bits bekannt
 - Korrektur: ermitteltes Bit wird invertiert
- Fehler tritt in einem der beiden Prüfwörter auf, nicht aber in Prüfbit P
 - Fehlerhaftes Paritäts-Bit
 - Keine Korrektur der Daten notwendig
- Fehler tritt in Prüfbit P auf
 - P selbst muss fehlerhaft sein
 - Keine Korrektur der Daten notwendig



Redundanz erzeugung

- Durch die Paritäts-Bits wird zusätzliche Redundanz eingeführt,
 - die von der Anzahl s der Bits pro Wort und
 - von der Anzahl k der Worte pro Block abhängt
- $R = (k + s + 1) / k$ [Bit/Wort]

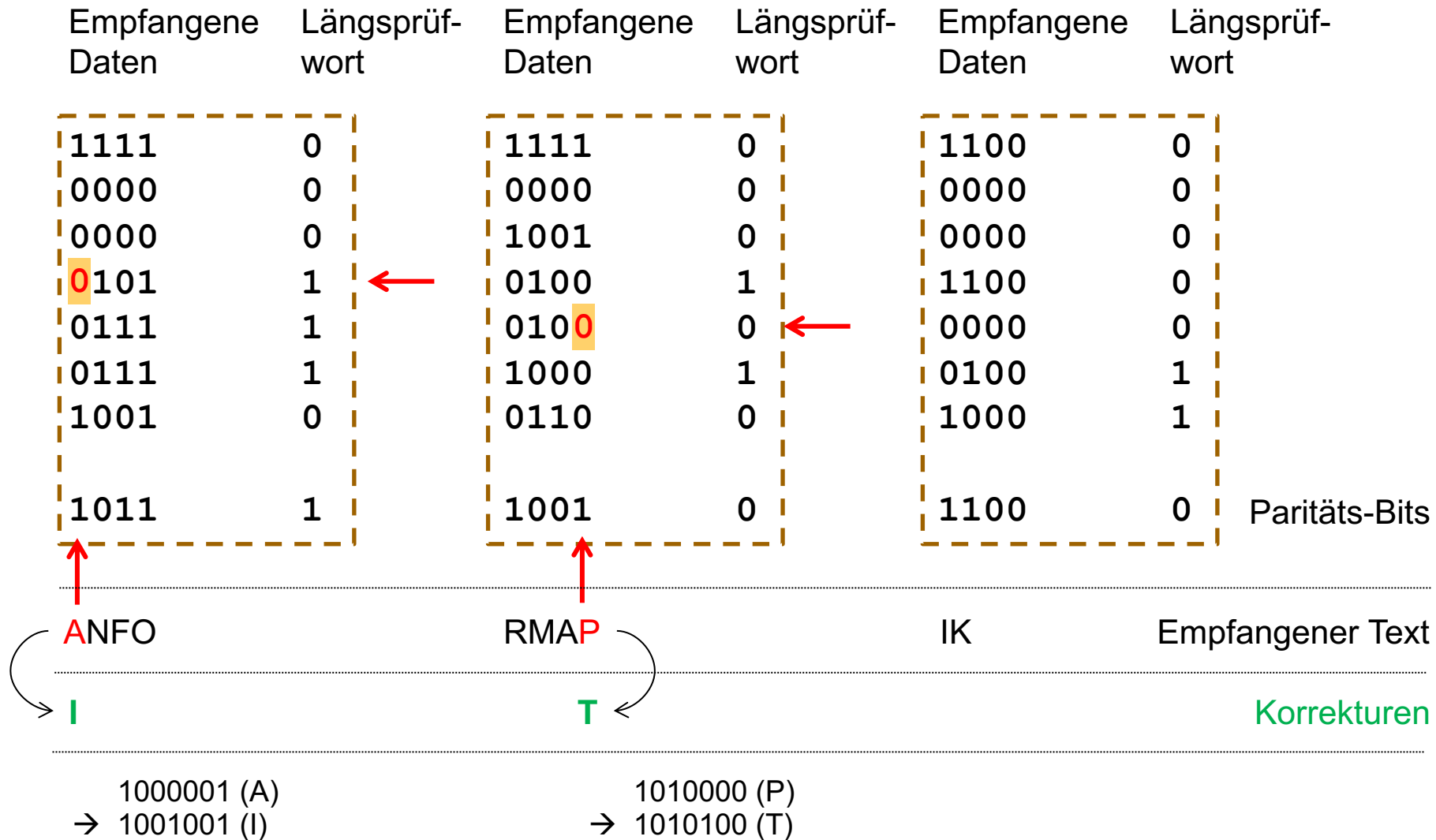


- Binäre Codierung des Worts **INFORMATIK** im ASCII-Code
 - die Anzahl der Einsen wird zu einer geraden Zahl in einem Paritäts-Bit ergänzt
 - die Anzahl der Einsen wird nach jedem vierten Wort in einem Längsprüfwort ergänzt
- Übertragung erfolgt in Blöcken
 - 1. und 2. Block enthalten 4 Zeichen
 - 3. Block enthält 2 Zeichen
- Bei Übertragung treten 1-Bit-Fehler auf und das Wort **ANFORMAPIK** wird empfangen



Beispiel (2)

Kapitel 4.1: Codesicherung und Kanalcodierung – Fehlertolerante Codes



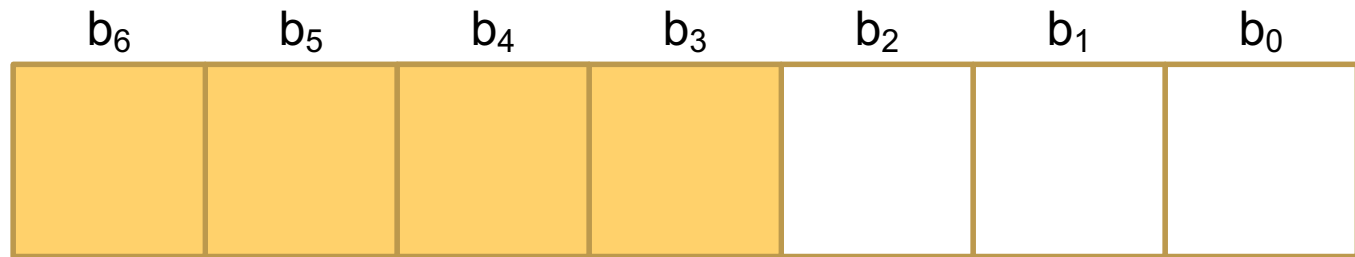
Tetraden mit drei Paritäts-Bits (1)

- Erweiterung des Konzepts der Paritäts-Bits
 - Für ein Codewort werden mehrere Paritäts-Bits zur Verfügung gestellt
- Vorteil
 - Jedes Wort kann für sich geprüft werden



Tetraden mit drei Paritäts-Bits (2)

Direkter binärer Tetraden-Code der Ziffern von 0 bis 9



- Mit 3 Paritäts-Bits
- Die vier höherwertigen Bits (b_6 bis b_3) sind direkt binär codierte Ziffern
- Niederwertige Bits (b_2 bis b_0) sind Paritäts-Bits
 - $b_2=1$, wenn Anzahl der Einsen in b_6, b_5, b_4 gerade
 - $b_1=1$, wenn Anzahl der Einsen in b_6, b_5, b_3 gerade
 - $b_0=1$, wenn Anzahl der Einsen in b_6, b_4, b_3 gerade



Tetraden mit drei Paritäts-Bits (3)

Direkter binärer Tetraden-Code der Ziffern von 0 bis 9 mit drei Paritäts-Bits

Ziffer Code

	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0
2	0	0	1	0	0	1	0
3	0	0	1	1	0	0	1
4	0	1	0	0	0	0	1

Ziffer Code

	b_6	b_5	b_4	b_3	b_2	b_1	b_0
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	1	1
8	1	0	0	0	0	0	0
9	1	0	0	1	0	1	1



Tetraden mit drei Paritäts-Bits (3)

- Annahme, dass alle Ziffern mit derselben Wahrscheinlichkeit $p = \frac{1}{10}$ auftreten
 - Entropie $H = \text{ld} \frac{1}{p} = \text{ld} 10 \approx 3.322 \frac{\text{Bit}}{\text{Zeichen}}$
 - Redundanz $R = L - H = 7 - 3.322 = 3.678 \frac{\text{Bit}}{\text{Zeichen}}$
- Hamming-Distanz der Tetraden alleine
 - $h = 1$
- Hamming-Distanz des Codes mit drei Paritäts-Bits
 - $h = 3$
 - 1-Bit-Fehler können erkannt und korrigiert werden ODER
 - 1-Bit- und 2-Bit-Fehler können erkannt, aber nicht korrigiert werden

