

6. Übung

□ Hammingcode

Hammingcodes in Arbeitsspeichern

- Der Hauptspeicher dient zur Speicherung von Daten und Programmen während des Rechenbetriebs.
- Erhöhung der Zuverlässigkeit gespeicherter Information im Hauptspeicher:

→ **Fehlertoleranz durch Informationsredundanz**
(ECC - error correcting codes)

- Einfache Verfahren (Quersummenbits) erlauben die Fehlererkennung.
- Aufwändigere Verfahren (Hammingcodes) die Fehlerkorrektur von Einzel- oder Mehrfachbitfehlern.

Fehlererkennung durch Quersummenbits

Jedem Datenwort, z. B. 8-Bit-Wort x_1, \dots, x_8 , wird ein neuntes Bit x_0 , das Prüfbit, angehängt, das sich durch Addition mod 2 oder Antivalenzbildung berechnet:

$$x_0 = x_1 \oplus \dots \oplus x_8$$

Damit wird jedes Codewort auf gerade Parität (Quersumme) ergänzt.

Der zusätzliche Speicheraufwand ermöglicht damit eine einfache Fehlerprüfung:

$$x_0 \oplus (x_1 \oplus \dots \oplus x_8) = 0$$

Einfache Bit-Fehlererkennung

$$x_0 \oplus (x_1 \oplus \dots \oplus x_8) = 0$$

Die Gleichung wird falsch, wenn ein **einzelnes** Bit oder eine **ungerade** Zahl von Bits fehlerhaft geworden ist.

→ **einfache Bitfehlererkennung**

Der Code kann jedoch **keine** Fehler erkennen, bei denen eine **gerade** Zahl von Bits fehlerhaft ist, und ermöglicht auch keine Fehlerkorrektur.

Fehlerkorrektur durch Hammingcodes

- ❑ Fehlerkorrekturcodes können ein fehlerfreies Codewort von einem fehlerhaften Codewort unterscheiden.
- ❑ Bei fehlerbehafteten Codewörtern sind die fehlerhaften Bits im Codewort innerhalb bestimmter Grenzen korrigierbar.
 - **Mehr als ein Prüfbit pro Datenwort notwendig**
- ❑ Um die Fehlerkorrektur vornehmen zu können, wird in den Korrekturbits eindeutig kodiert, ob und ggf. welche Bits im gesamten Codewort (Daten- und Korrekturbits) fehlerhaft sind.



Ein-Bit-Fehlerkorrektur

Ein-Bit-Fehlerkorrektur:

Notwendiger Aufwand für die sog. *Hammingcodes*:

$$2^k \geq m + k + 1$$

k : Zahl der zusätzlichen Korrekturbits

m : Zahl der Datenbits



Mindestaufwand: Ein-Bit-Fehlerkorrekturcodes

Zahl der Datenbits m	Zahl der Prüfbits k	Bits insgesamt: $m + k$
1	2	$m + 2$
2 bis 4	3	$m + 3$
5 bis 11	4	$m + 4$
12 bis 26	5	$m + 5$
27 bis 57	6	$m + 6$
...



Aufbauprinzipien

Unterschiedliche Aufbauprinzipien sind möglich.

Hier: Einfaches Realisierungsschema

Vorteil: beliebige Erweiterbarkeit auf größere Datenwortlängen.

Der Ansatz verwendet k verschiedene Prüfgleichungen für die Quersummen QS_0 bis QS_{k-1}



Ansatz

Die i -te Quersumme (QS_i) ergänzt alle Positionen im Codewort, die die Potenz 2^i in ihrer Ziffernwertigkeit enthalten ($i = 0, 1, \dots, k-1$) auf gerade.

$k_1 = QS_0$ über alle ungeraden Stellen **1, 3, 5, 7, ...**

$k_2 = QS_1$ über die Stellen **2, 3, 6, 7, 10, 11, ...**

$k_3 = QS_2$ über die Stellen **4, 5, 6, 7, 12, 13, 14, 15, ..**

$k_4 = QS_3$ über die Stellen **8, 9, 10, 11, 12, 13, 14, 15, ..**

```
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
...
```



Ansatz

- Es werden gerade Quersummen verwendet
- Datenwörter werden zum Einfügen der Prüfbits gespreizt, indem die Codewortposition 2^i für die i -te Quersummenprüfstelle k_{i+1} freigehalten wird (m_i bezeichnen die Datenbits).

Damit erhält man folgendes Schema:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m_{12}	k_5	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1	Stellen



Ansatz

- Beim Lesen eines evtl. fehlerhaften Codeworts kann durch **erneutes Bilden der Quersummen** ermittelt werden, ob und an welcher Position ein Fehler aufgetreten ist.
- Die Quersummen geben dabei, als Dualzahl interpretiert, die fehlerhafte Position im Codewort an.
- Liegt kein Fehler vor, dann sind alle Prüfgleichungen erfüllt.



Beispiel

Hammingcode mit 4 Datenbits und 3 Prüfbits, also 7-Bit-Codewörtern

Datenwort : 1 1 0 1

	7	6	5	4	3	2	1
→	1	1	0	-	1	-	-
	m_4	m_3	m_2	k_3	m_1	k_2	k_1

Bestimmung der Prüfbits:

$$k_1 = m_1 \oplus m_2 \oplus m_4 = 1 \oplus 0 \oplus 1 = 0$$

$$k_2 = m_1 \oplus m_3 \oplus m_4 = 1 \oplus 1 \oplus 1 = 1$$

$$k_3 = m_2 \oplus m_3 \oplus m_4 = 0 \oplus 1 \oplus 1 = 0$$

→ Codewort : 1 1 0 0 1 1 0



Beispiel

Bildet man erneut die Prüfbits bei fehlerfreiem Codewort, ergibt sich als Syndrom $k_3k_2k_1$ der Wert 000.

Codewort: 1 1 0 0 1 1 0

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 = 1 \oplus 1 \oplus 1 \oplus 1 = 0$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

Codewort: 1 1 0 0 0 1 0

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$k_3k_2k_1 = 011_2 = 3_{10} \quad \text{Fehler an 3. Position}$$



Beispiel

Ist ein **Prüfbit verfälscht**, z. B. k_3 , wird **nur** die betroffene Prüfgleichung beeinflusst, hier QS_2 .

Codewort: $m_4 \ m_3 \ m_2 \ k_3 \ m_1 \ k_2 \ k_1$
1 1 0 1 1 1 0

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 = 1 \oplus 1 \oplus 1 \oplus 1 = 0$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$k_3k_2k_1 = 100_2 = 4_{10} \quad \text{Fehler an 4. Position}$$



Aufgaben

➤ Hamming-Code

- Erkennen und korrigieren von 1-Bit-Fehlern
- Erkennen von 2-Bit-Fehlern

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m_{12}	k_5	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1	Stellen

Datenwort: 1 0 1 0 1 0 1 0

Datenwort ausspreizen:

12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	-	1	0	1	-	0	-	-
m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1

Bestimmung der Prüfbits:

$$k_1 = m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$k_2 = m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$k_3 = m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$k_4 = m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

➔ **Codewort:** 1 0 1 0 0 1 0 1 1 0 0 0

0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
...



Aufgabe 2

Der Hauptspeicher eines Rechners mit 8-Bit Datenwortbreite unterstützt eine Einzelbitfehler-Korrektur. Aus dem Speicher erhält man die beiden Codewörter

- Codewort 1: 101001000110
- Codewort 2: 100000000000

Prüfen Sie beide Codewörter auf Fehler, die beim Übertragen oder Speichern entstanden sein könnten und korrigieren Sie diese (falls notwendig). Geben Sie die zugehörigen Datenwörter an.



17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m ₁₂	k ₅	m ₁₁	m ₁₀	m ₉	m ₈	m ₇	m ₆	m ₅	k ₄	m ₄	m ₃	m ₂	k ₃	m ₁	k ₂	k ₁	Stellen

Hammingcode mit 8 Datenbits und 4 Prüfbits

12 11 10 9 8 7 6 5 4 3 2 1
Codewort 1 : 1 0 1 0 0 1 0 0 0 1 1 0
m₈ m₇ m₆ m₅ k₄ m₄ m₃ m₂ k₃ m₁ k₂ k₁

Bestimmung der Prüfbits:

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 0$$

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$k_4 = k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$k_4 k_3 k_2 k_1 = 0000 \rightarrow$ Codewort ist fehlerfrei

\rightarrow Datenwort: 1 0 1 0 1 0 0 1



17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Position
	4								3				2		1	0	i
m ₁₂	k ₅	m ₁₁	m ₁₀	m ₉	m ₈	m ₇	m ₆	m ₅	k ₄	m ₄	m ₃	m ₂	k ₃	m ₁	k ₂	k ₁	Stellen

12 11 10 9 8 7 6 5 4 3 2 1
Codewort 2 : 1 0 0 0 0 0 0 0 0 0 0 0
m₈ m₇ m₆ m₅ k₄ m₄ m₃ m₂ k₃ m₁ k₂ k₁

Bestimmung der Prüfbits:

$$k_1 = k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$$

$$k_2 = k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$$

$$k_3 = k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

$$k_4 = k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

$k_4 k_3 k_2 k_1 = 1100 = 12_{10} \rightarrow$ Fehler in der 12. Position

\rightarrow Datenwort: 0 0 0 0 0 0 0 0



Hamming-Codes

An diese Codewörter lässt sich ein weiteres Quersummenbit anfügen, so dass Zweibitfehler im Codewort erkennbar werden.

32-Bit Wortlänge werden durch 7 Prüfbits zur Einzelfehlerkorrektur und Doppelfehlererkennung ergänzt.

64 Bit lange Datenwörter erfordern 8 Korrekturbits.

Die Algebra endlicher Körper bietet über Generator- und Nullmatrix Verfahren, die zu jedem Datenwort ein Prüfbyte erzeugen.



Hamming-Code

• Fall 1: Kein Fehler

m_5 k_4 m_4 m_3 m_2 k_3 m_1 k_2 k_1 **PB**
0 0 1 1 0 0 1 1 0 0

$$k_4k_3k_2k_1 = 0000$$

• Fall 2: 1-Bit-Fehler

0 0 1 0 0 0 1 1 0 1

$$k_4k_3k_2k_1 = 0110$$

Fehler an 6. Position

• Fall 3: 2-Bitfehler

0 0 1 0 0 0 1 0 0

$$k_4k_3k_2k_1 = 0101$$

2-Bit-Fehler



Hamming-Code

- Die Betrachtungen gelten genau dann, wenn garantiert werden kann, dass tatsächlich **nur 1-Bitfehler** auftreten
- Der Hammingcode kann sehr wohl mehrfache Bitfehler, erkennen aber nur 1-Bitfehler korrigieren

Voraussetzung: Nur 1- oder 2-Bitfehler treten auf !

- Im Fall von 2-Bitfehlern werden diese vom Hammingcode erkannt, können aber nicht von 1-Bitfehlern unterschieden werden:

Bei dem Syndrom $k_4k_3k_2k_1 = 0101$ kann sowohl einen 1-Bitfehler an der 5. Position als auch einen 2-Bitfehler an zwei unbekannt Positionen vorliegen.



Hamming-Code

- Um 1-Bitfehler von 2-Bitfehlern unterscheiden zu können, fügt man dem Hammingcode ein weiteres Paritätsbit hinzu

1-Bitfehler:

- Erkennbar durch das Paritätsbit und
- Erkennbar und korrigierbar durch den Hammingcode

2-Bitfehler:

- Nicht erkennbar durch das Paritätsbit
- Erkennbar durch den Hammingcode, **aber** nicht korrigierbar, da die fehlerhaften Positionen nicht ermittelt werden können

n-Bitfehler (n>2): andere Verfahren



Vorlesungsbefragung

Besonders **schlecht** hat mir gefallen:

- Lokon
- Übungsblätter unpunktlich
- Bonussystem nicht transparent
- Tabellen zeilenweise erklären (langweilig, ...)
- Langes Behandeln einfacher Themen
- Schnelles Behandeln schwieriger Themen (Fliflops)
- Überflüssige Notizen auf den Folien (Vergewaltigung der Folien), Viele Farben

- Hörsaal (ohne WLAN), schlechter Beamer
- Nichts



Vorlesungsbefragung

Besonders **gut** hat mir gefallen:

- Bonussystem
 - Gutes Skript
 - Regelmäßiger Versuch einen Bezug zur Praxis herzustellen
 - Lustige Vorlesung, die den Stoff einfach vermittelt
 - Dozent interessiert, ob alles verstanden wurde
 - Dozent bringt den Stoff sehr verständlich rüber
 - Es wird Wert darauf gelegt, dass man auch die Zusammenhänge versteht.
 - Beste Vorlesung an der Uni KA bis jetzt
-
- Der Dozent, seine Motiviertheit, Dozent wirkt nie unvorbereitet
 - Lockere Art von Tamim, Dozent ist den Studenten nah
 - Dass Tamim stets gut gelaunt ist
 - „Du weißt doch, dass Du der Beste bist“
 - „Tamim für TI-2“
 - Ü, U-Umlaut, Das u-Umlaut-Kalkül

Colani



Ende der Vorlesung!

Eine erfolgreiche und erholsame
vorlesungsfreie Zeit wünscht Euch das
TI-Team:

- die Tutoren und
- Tamim