Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 12
Rechnerarchitektur im Sommersemester 2023

Zu dem Modul O

Besprechung: 10.07.2023 - 14.07.2023

Aufgabe 1: (T) Hamming Codes

(- Pkt.)

Übertragung von Daten über physische Kanäle (Kabel etc.) ist fehleranfällig. Indem man ein einzelnes Bit, das *Paritätsbit*, zu jedem Datenpaket hinzufügt, kann man Ein-Bit-Fehler entdecken. Mit einem einzelnen Paritätsbit ist es allerdings nicht möglich herauszufinden, welches Bit fehlerhaft ist. Wenn man also das fehlerhafte Bit erkennen möchte, benötigt man mehr Paritätsbits.
Das folgende Verfahren wurde mit dem Ziel Ein-Bit Datenfehler, die durch unzuverlässige Übertragungskanäle verursacht wurden, *zu erkennen und zu korrigieren*.
Der Trick besteht darin, zusätzliche Paritätsbits in die Datenpakete einzufügen und so ein *Hamming Codewort* zu bilden. Das Hamming Codewort besteht aus den eigentlichen Datenbits, die übertragen werden sollen, und einigen Paritätsbits, die an strategischen Punkten eingefügt wurden.
Die Anzahl der benötigten Paritätsbits ist abhängig von der Anzahl der Datenbits:

Daten Bits:	8	16	32	64	128
Paritäts-Bits:	4	5	6	7	8
Codewort:	12	21	38	71	136 Bits

Allgemein gilt: Für Daten den Länge 2^n Bits werden $n+1$ Paritätsbits eingefügt, um das Codewort zu bilden.
Algorithmus:

1. Die Bits des Codeworts werden von 1 an nummeriert. Bit 1 (das am weitesten links stehende) ist das *most significant bit*.

2. Die Paritätsbits sind die Bits des Codeworts, deren Nummer eine Potenz von 2 ist, also 1,2,4,8, ... Alle anderen Bits sind Datenbits.

3. Jedes Paritätsbit prüft mehrere genau festgelegte Bits des Codeworts, sich selbst eingeschlossen. Ein Bit des Codewortes kann von mehr als einem Paritätsbit geprüft werden.

   Ein Bit B wird von den Paritätsbits $P1, P2, P3 \ldots, Pk$ geprüft, wenn $B = P1 + P2 + \cdots + Pk$ ist.

   Beispiel: Bit 11 wird von den Paritätsbits 1,2 und 8 geprüft.

```
Parity Bit Getestete Bits
      1 :     1    3    5    7     9   11   13   15   17   19   21
      2 :     2    3    6    7    10   11   14   15   18   19
      4 :     4    5    6    7    12   13   14   15   20   21
      8 :     8    9   10   11    12   13   14   15
     16 :    16   17   18   19    20   21
```

4. Die Paritätsbits werden so gesetzt, dass die Summe der geprüften Bits (sich selbst eingeschlossen) gerade ist.

   Beispiel: Um das 8-Bit Datenwort 1011 0110 zu kodieren, benötigen wir vier Paritätsbits; das Codewort ist also 12 Bits lang und sieht folgendermaßen aus:

   P P D P D D D P D D D D (P = Paritätsbit, D = Datenbit)

```
                1  2  3  4  5  6  7  8  9 10 11 12
Codewort    :   ?  ?  1  ?  0  1  1  ?  0  1  1  0
Parity Bit 1:   ?     1     0     1     0     1     ? = 1 damit Summe gerade
Parity Bit 2:      ?  1        1  1        1  1     ? = 1 damit Summe gerade
Parity Bit 4:            ?  0  1  1              0  ? = 0 damit Summe gerade
Parity Bit 8:                        ?  0  1  1  0  ? = 0 damit Summe gerade
Codewort    :   1  1  1  0  0  1  1  0  0  1  1  0
```

Fehlererkennung:

a. Um einen Fehler zu erkennen und zu korrigieren, berechnet man die Checksumme für jedes Paritätsbit. Wenn alle Checksummen gerade (bzw. ungerade) sind, dann ist das Codewort fehlerfrei.

b. Identifiziere alle Paritätsbits mit fehlerhaften Checksummen. Bilde die Schnittmenge aller von nicht korrekten Paritätsbits geprüften Bits. Eliminiere alle Bits, die auch von korrekten Paritätsbits geprüft werden. Das fehlerhafte Bit bleibt übrig.

   Alternative Methode: Die Summe der Nummern der fehlerhaften Paritätsbits ergibt die Nummer des fehlerhaften Bits.

Beispiel: Finde und korrigiere einen eventuell vorhandenen Fehler in dem Codewort 1100 0110 0110. Dieses 12-Bit Codewort hat vier Paritätsbits, Bits Nummer 1,2,4 und 8.

```
                1   2   3   4   5   6   7   8   9  10  11  12
Codewort:       1   1   0   0   0   1   1   0   0   1   1   0
Parity Bit 1:   1       0       0       1       0       1     :3 X
Parity Bit 2:       1   0           1   1           1   1     :5 X
Parity Bit 4:               0   0   1   1                 0 :2
Parity Bit 8:                               0   0   1   1   0 :2
```

Die Prüfsummen der Bits 1 und 2 sind falsch. Die Schnittmenge der Bitlisten sind die Bits 3, 7 und 11, also ist eines von diesen fehlerhaft.
Bit 11 wird auch von Bit 8 geprüft, dessen Checksumme korrekt ist, also ist auch Bit 11 OK.
Bit 7 wird auch von Bit 4 geprüft, dessen Checksumme korrekt ist, also ist auch Bit 7 OK.
Bit 3 wird nur von den Bits 1 und 2 geprüft, also ist Bit 3 falsch.

```
Korrigiert:  1   1   1   0   0   1   1   0   0   1   1   0
8-Bit Daten:         1       0   1   1       0   1   1   0
```

Aufgaben:

a. Beantworten Sie zunächst die folgenden Fragen zum Hamming-Abstand

   (i) Was ist der Hamming-Abstand?

   (ii) Wie groß muss der Hamming-Abstand mindestens sein um d Einzelbitfehler erkennen zu können? Begründen Sie Ihre Antwort ausführlich!

   (iii) Wie groß muss der Hamming-Abstand mindestens sein um d Einzelbitfehler korrigieren zu können? Begründen Sie Ihre Antwort ausführlich!

b. Kodieren Sie die folgenden 8-Bit Daten in 12-Bit Hamming Codes:

   (i) 1010 1110
   (ii) 0101 0001

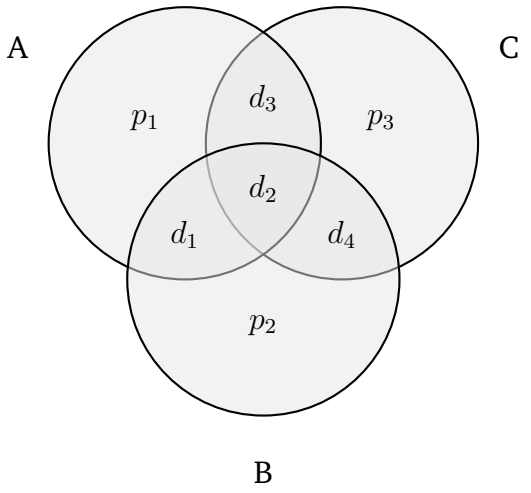
c. Dekodieren Sie die folgenden 12-Bit Codewörter. Wenn sie Fehler enthalten, identifizieren Sie das fehlerhafte Bit und korrigieren Sie den Fehler. Das Ergebnis muss ein 8-Bit Datenwort sein.

   (i) 1101 0110 0111
   (ii) 1101 1110 0111

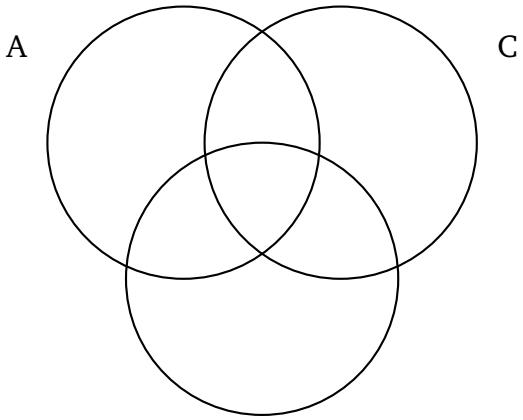
Aufgabe 2: (T) Fehlererkennungscodes

(- Pkt.)

Wir gehen von folgender Struktur der Code-Wörter $d_1d_2d_3d_4p_1p_2p_3$ aus. Wobei $d_i(i \in \{1,2,3,4\})$ für das jeweilige Datenbit und $p_j(j \in \{1,2,3\})$ für das jeweilige Prüf- bzw. Paritätsbit steht. Die Paritätsbits zur Fehlererkennung bzw. Fehlerkorrektur für ein Datenwort $d_1d_2d_3d_4$ können anschaulich mit Hilfe eines Venn-Diagramms berechnet werden, in welchem sich die Bits wie folgt anordnen:



a. Berechnen Sie unter Verwendung des folgenden Venn-Diagramms die Prüfbits für das Datenwort **1111**. Verwenden Sie dazu **gerade Parität**. Tragen Sie zunächst die Datenbits in die für die Berechnung sinnvollen (Schnitt-)Mengen ein.



b. Gehen Sie nun davon aus, dass Sie ein mit dem zuvor beschriebenen Code codiertes Code-Wort `0001010` empfangen haben. Es wurde **gerade Parität** verwendet. Handelt es sich um ein gültiges Codewort? Falls nein, treffen Sie eine Aussage darüber, an welcher/welchen Stelle/Stellen mutmaßlich (ein) Bitfehler aufgetreten ist/sind. Verwenden Sie zur Berechnung das folgende Venn-Diagramm. Korrigieren Sie (falls möglich/nötig) den/die Fehler **innerhalb** des Venn-Diagramms und geben Sie das (ggf. korrigierte) 4-Bit Datenwort an.