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

mobile and distributed systems group

Übungsblatt 12
Rechnerarchitektur im SS 22

Zum Modul O

Abgabetermin: 24.07.2022, 18:00 Uhr

Besprechung: T-Aufgaben: 18.07.22 - 22.07.22

Ankündigung: Am Montag, den **25. Juli 2022 findet von 18.00 – 20.00 Uhr ein Sondertutorium via Zoom statt** für alle statt, an dem nochmal gezielt Fragen zum Stoff gestellt werden können. In der Woche vom 25. – 29. Juli 2022 finden keine regulären Übungen und auch keine Vorlesung statt.
Die **Klausur** findet am **28. Juli 2022 von 14-17 Uhr** statt. Bitte melden Sie sich **bis spätestens 24. Juli 2022, 23:59 Uhr** zur Klausur ber Uni2work **an** bzw. **ab**. Wir wünschen Ihnen viel Erfolg!

Aufgabe 76: (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 Paraitätsbits, 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 eins 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

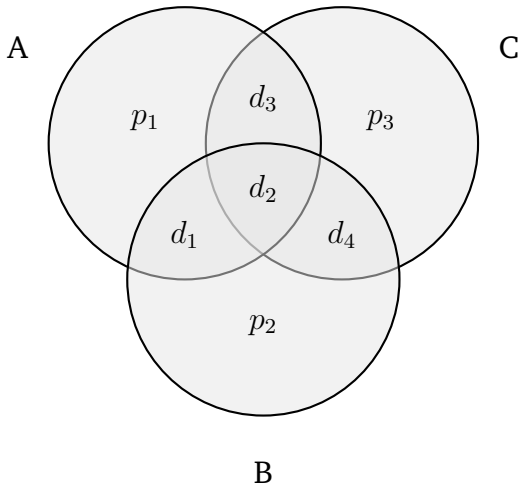
c. Dekodieren Sie die folgenden 12-Bit Codewörter. Wenn sie Fehler enthalten, identifizieren Sie das fehlerhafte Bit und korrigieren Sie den Fehler. Das Ergebnis muss ein 8-Bit Datenwort sein.

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

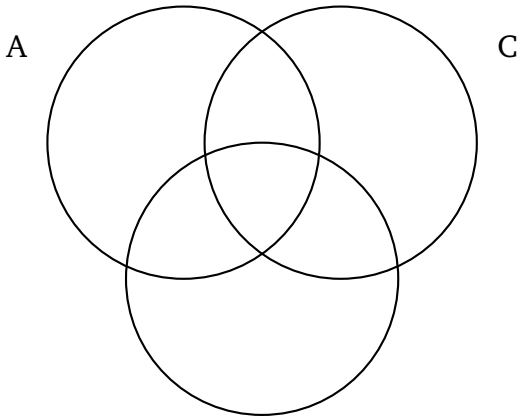
Aufgabe 77: (T) Fehlererkennungscodes

(- Pkt.)

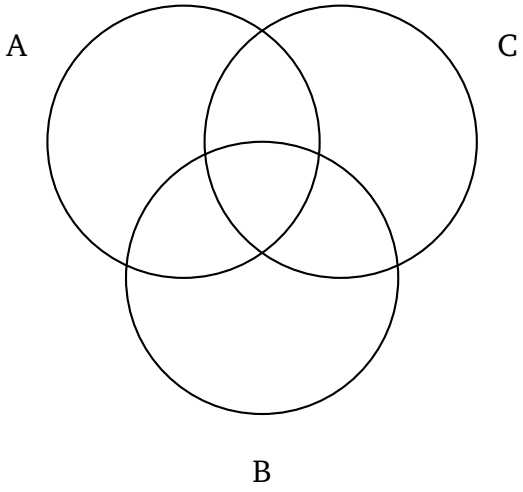
Wir gehen von folgender Struktur der Code-Wörter $d_1d_2d_3d_4p_1p_2p_3$ aus. Wobei $d_i(i \in \{1,2,3,4\})$ für das jeweilige Datenbit und $p_j(j \in \{1,2,3\})$ für das jeweilige Prüf- bzw. Paritätsbit steht. Die Paritätsbits zur Fehlererkennung bzw. Fehlerkorrektur für ein Datenwort $d_1d_2d_3d_4$ 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 folgenden Venn-Diagramm. Korrigieren Sie (falls möglich/nötig) den/die Fehler **innerhalb** des Venn-Diagramms und geben Sie das (ggf. korrigierte) 4-Bit Datenwort an.

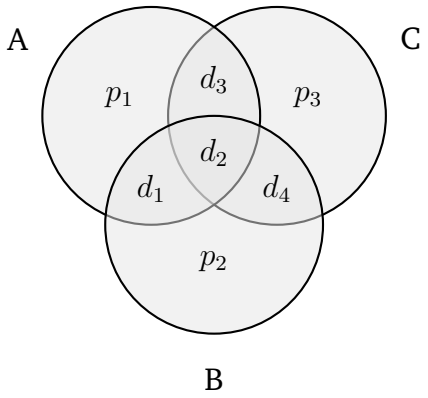


Aufgabe 78: (H) Fehlererkennung und -korrektur

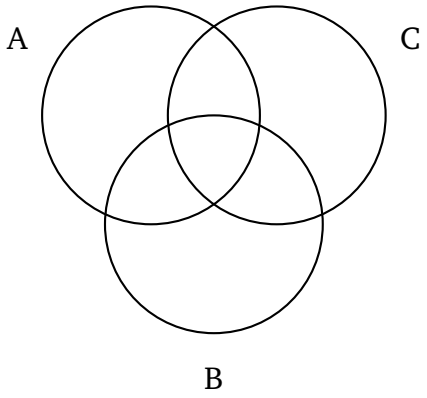
(13 Pkt.)

Zum Schutz vor Speicherfehlern werden sogenannte Codes zur Fehlererkennung und zur Fehlerkorrektur eingesetzt. Bearbeiten Sie die folgenden Teilaufgaben:

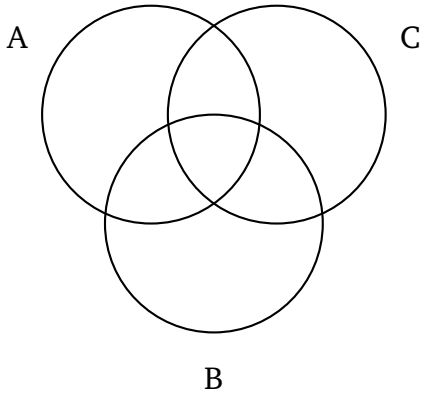
a. Wir gehen von folgender Struktur der Code-Wörter $d_1d_2d_3d_4p_1p_2p_3$ aus. Wobei $d_i(i \in \{1,2,3,4\})$ für das jeweilige Datenbit und $p_j(j \in \{1,2,3\})$ für das jeweilige Prüf- bzw. Paritätsbit steht. Die Paritätsbits zur Fehlererkennung bzw. Fehlerkorrektur für ein Datenwort $d_1d_2d_3d_4$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



(i) Berechnen Sie unter Verwendung des folgenden Venn-Diagramms die Prüfbits für das Datenwort `1110`. Verwenden Sie dazu **gerade Parität**. Tragen Sie zunächst die Datenbits in die für die Berechnung sinnvollen (Schnitt-)Mengen ein.



(ii) Gehen Sie nun davon aus, dass Sie ein mit dem zuvor beschriebenen Code codiertes Code-Wort `0011010` 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**.

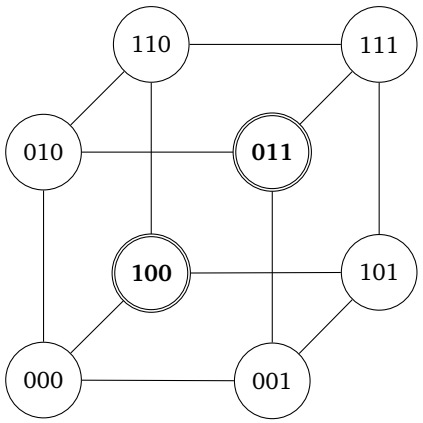


(iii) Wie ist beim vorliegenden Fehlererkennungs- bzw. Fehlerkorrekturcode der Overhead der Prüfbits bzgl. der Speicherbits in Prozent (%)?

b. In dieser Teilaufgabe sei ein Fehlerkorrekturcode gegeben, der nur aus **zwei gültigen Codewörtern** besteht:

- `100`
- `011`

In der Abbildung rechts sind alle möglichen Kombinationen von 3 Bit eingezeichnet und die gültigen Codewörter markiert.



Bearbeiten Sie dazu die folgenden Aufgaben:

(i) Wie viele Einzelbitfehler können mit dem vorliegenden Code in jedem Fall **erkannt** werden?

(ii) Wie viele Einzelbitfehler können mit dem vorliegenden Code in jedem Fall **korrigiert** werden?

Aufgabe 79: (H) Hamming Codes

(12 Pkt.)

Übertragung von Daten über physische Kanäle (Kabel etc.) ist fehleranfällig. Als Schutz vor solchen Fehlern setzen die meisten Speicher Codes für die Fehlererkennung und ggf. auch zur Fehlerkorrektur ein. Lesen Sie sich im Vorlesungsskript das Kapitel 14.4 zur „Fehlererkennung und -korrektur“ (S. 173-176) aufmerksam durch und bearbeiten Sie die folgenden Aufgaben:

a. Kodieren Sie die folgenden 16-Bit Daten in 21-Bit Hamming Code. Verwenden Sie dazu gerade Parität.

   (i) 1100 1010 0000 0100

b. Dekodieren Sie das folgende 21-Bit Codewort. Wenn Sie Fehler enthalten, identifizieren Sie das fehlerhafte Bit und korrigieren Sie den Fehler. Das Ergebnis muss ein 16-Bit Datenwort sein.

   (i) 0110 1011 0110 1100 1101 1

Aufgabe 80: (H) Einfachauswahlaufgabe: Darstellung von Speicherinhalten

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“).

a) Wie viele Bit stehen im ursprünglichen ASCII-Code zur Kodierung eines Zeichens zur Verfügung?

(i) `1`	(ii) `7`	(iii) `16`	(iv) `128`

b) Die Dezimalzahl 16.909.060 (01020304 Hexadezimal) soll als 32-Bit-Integer-Wert (Wortbreite) ab Speicheradresse 0000 gespeichert werden. Dabei kommt die Little Endian Byte-Anordnung zum Einsatz. Welche Antwort entspricht der resultierenden Speicherbelegung?

(i)

Adresse	Wert
0000	01
0001	02
0002	03
0003	04

(ii)

Adresse	Wert
0000	02
0001	01
0002	04
0003	03

(iii)

Adresse	Wert
0000	04
0001	03
0002	02
0003	01

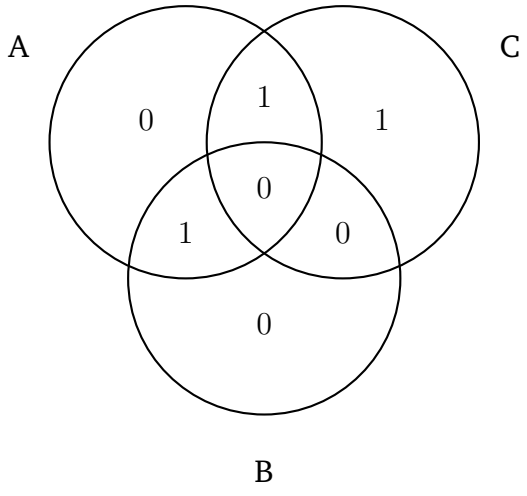
(iv)

Adresse	Wert
0000	03
0001	01
0002	02
0003	04

c) Welche Operation kann auf zwei gleichlange Codewörter angewendet werden, um durch Zählen der 1en im Ergebnis den Hamming-Abstand der Codewörter zu bestimmen?

(i) AND	(ii) OR	(iii) XOR	(iv) NOR

d) Gehen Sie nun davon aus, dass Sie folgendes Code-Wort **`1010001`** empfangen haben. Es wurde **gerade Parität** verwendet. Bei der Übertragung ist ein einzelner Bitfehler aufgetreten. Welches Paritätsbit ist betroffen?



(i) A	(ii) B	(iii) C	(iv) keins

e) Angenommen ein Speicherwort wird in einem kurzen Zeitintervall k mal gelesen oder geschrieben und befindet sich nach dem ersten Zugriff im Cache. Wie berechnet sich die Trefferrate (Hit Ratio) h?			
(i) $h = \frac{k-1}{k}$	(ii) $h = \frac{k}{k-1}$	(iii) $h = (k-1) \cdot (k)$	(iv) $h = \frac{k}{k}$

Aufgabe 81: (H) Nachgefragt

(Pkt.)

Diese Aufgabe dient dazu, sich nochmals gezielt Fragen über den Stoff zu überlegen!
Bitte formulieren Sie **auf freiwilliger Basis** Fragen, die Ihnen beim Durcharbeiten Ihrer Vorlesungsmitschriften (bzw. des Skripts) oder bei der Bearbeitung der Übungsblätter bisher unbeantwortet geblieben sind. Laden Sie Ihre Fragen bitte bis **spätestens 22.07.2022 23:59 Uhr** bei Uni2Work hoch. Dort wird es eine gesonderte Upload-Möglichkeit mit der Bezeichnung „Nachgefragt“ neben den üblichen Upload-Möglichkeiten für die Lösung der H-Aufgaben geben. Ihre eingereichten Fragen werden dann in einem zusätzlichen Sondertutorium beantwortet. Dieses findet satt am: **25. Juli 2022 von 18.00 - 20.00 Uhr Online über Zoom**.

Wir wünschen Ihnen viel Erfolg bei den Vorbereitungen auf die Klausur!