Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch
Holger Lange
Mathias Halbach (FG Rechnerarchitektur)

04.05.2006

Technische Grundlagen der Informatik II
1. Übung – Zeichencodierung
Sommersemester 2006

Aufgabe 1: Zahlendarstellung

a) Erstellen Sie eine Tabelle, in der die Dezimalzahlen von 0 bis 16, die zugehörige Binär- und Hexadezimaldarstellung, der BCD-Code in binärer Schreibweise und der Gray-Code aufgeführt sind.

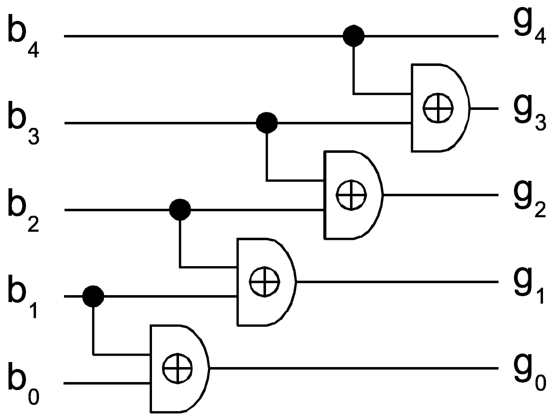
Lösung:

Dez.	Bin.	Hex.	BCD	Gray
0	00000	0	0000 0000	00000
1	00001	1	0000 0001	00001
2	00010	2	0000 0010	00011
3	00011	3	0000 0011	00010
4	00100	4	0000 0100	00110
5	00101	5	0000 0101	00111
6	00110	6	0000 0110	00101
7	00111	7	0000 0111	00100
8	01000	8	0000 1000	01100
9	01001	9	0000 1001	01101
10	01010	A	0001 0000	01111
11	01011	B	0001 0001	01110
12	01100	C	0001 0010	01010
13	01101	D	0001 0011	01011
14	01110	E	0001 0100	01001
15	01111	F	0001 0101	01000
16	10000	10	0001 0110	11000

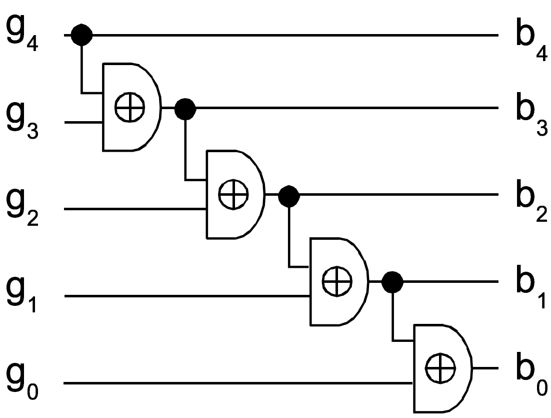
b) Erstellen Sie eine Schaltung, mit der aus der obigen Binärdarstellung der zugehörige Gray-Code generiert wird. Wie sieht die Schaltung für die andere Konvertierungsrichtung (Gray nach binär) aus?

Lösung:

Binär nach Gray:



Gray nach binär:



Aufgabe 2: Hamming-Distanz

a) Was ist die Hamming-Distanz zwischen den Codes

- 0110 und 0111,
- 1001 und 0111 sowie
- 1111 und 0000?

Lösung:

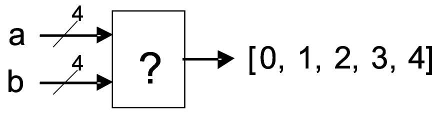
$d = 1$, $d = 3$ und $d = 4$

b) Welche binären Codes gibt es für die Hamming-Distanz von 3 für den Code 0100?

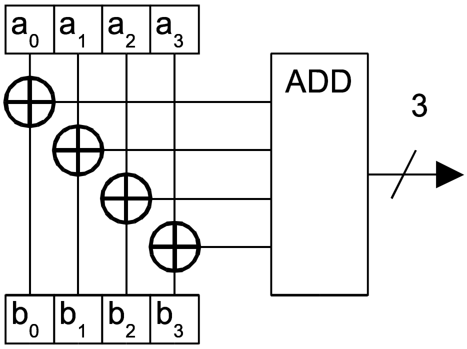
Lösung:

0011, 1111, 1001, 1010

c) Konstruieren Sie einen Mechanismus mit Logik-Gattern, um die Hamming-Distanz zwischen zwei 4-Bit-Zeichen zu errechnen.



Lösung:



d) Warum funktioniert die 1-Bit Fehlerkorrektur mit Codes der Hamming-Distanz 3 nicht zuverlässig?

Lösung:

Alle 1-Bit Fehler können zwar immer sicher korrigiert werden, falls aber ein 2-Bit Fehler auftritt, wird das System das falsche Codewort zur Korrektur auswählen, da die Hamming-Distanz des fehlerhaften Codes zum falschen Korrekturwort (1) geringer ist als die zum richtigen (2).

Beispiel:

Mögliche Codewörter: 1111, 1000, 0001

Gesendetes Wort: 1111

Empfangenes fehlerhaftes Wort:

- 1-Bit-Fehler $\rightarrow$ 1110. Hamming-Distanz $d(1110, 1000) = 2$, $d(1110, 0001) = 4$, $d(1110, 1111) = 1$ $\Rightarrow$ richtig korrigiertes Wort ist 1111.
- 2-Bit-Fehler $\rightarrow$ 1010. Hamming-Distanz $d(1010, 1000) = 1$, $d(1010, 0001) = 3$, $d(1010, 1111) = 2$ $\Rightarrow$ *falsch* korrigiertes Wort ist 1000.

Deshalb sollen Codes mit Hamming-Abstand 3 zur Fehlerkorrektur nur verwendet werden, wenn höchstens 1-Bit-Fehler auftreten können. Sollen jedoch nur Fehler erkannt, nicht aber korrigiert werden, ist die Verwendung dieser Codes zulässig.

Aufgabe 3: 4-Bit-CRC-Algorithmus

Erstellen Sie mit dem Generatorpolynom $G(x) = x^3 + x^2 + 1$ für den Datenstrom $D(x) = x^5 + x^4 + x^2 + 1$ (110101) eine Prüfsumme $R(x)$ mittels Polynomdivision. $R(x)$ soll mit '000' vorinitialisiert sein.

Lösung:

$$\begin{array}{l}
\overbrace{110101}^{D(x)}\overbrace{000}^{R(x)} : \overbrace{1101}^{G(x)}\\
\underline{1101}\\
00000\\
\underline{0000}\\
00001\\
\underline{0000}\\
0001\ 0\\
\underline{000\ 0}\\
001\ 00\\
\underline{00\ 00}\\
1\ 000\\
\underline{1\ 101}\\
0\ 101 = R(x)
\end{array}$$

Aufgabe 4: Shannon-Fano-Code

Gegeben ist das Alphabet $A = \{t, o, b, e, n, \_, r\}$.

a) Erstellen Sie für den Text "to_be_or_not_to_be" eine Wahrscheinlichkeitstabelle und daraus eine Shannon-Fano-Codierung, indem Sie die absteigend sortierte Liste halbieren, so dass in beiden Teilen die Summe der Wahrscheinlichkeiten annähernd gleich ist. Der oberen bzw. linken Hälfte wird eine 0 zugewiesen, der unteren bzw. rechten eine 1. Sind mehrere Zeichen in einer Hälfte, so wird die Teilung für die jeweilige Hälfte wiederholt und es ergibt sich die nächste Codeziffer.

Lösung:

z	Anzahl(z)	P(z)	Code	Lnge	$\sum$Bits
_	5	0,2778	00	2	10
o	4	0,2222	01	2	8
t	3	0,1667	100	3	9
e	2	0,1111	101	3	6
b	2	0,1111	110	3	6
r	1	0,0556	1110	4	4
n	1	0,0556	1111	4	4
$\sum$	18	1,0001			47

b) Wie viele Bits brauchen Sie für die Codierung?

Lösung:

47 Bits (siehe Tabelle)

c) Wie viele Bits bräuchten Sie mit einem minimalen Code konstanter Länge?

Lösung:

7 Zeichen: $\lceil \log_2 7 \rceil = 3 \Rightarrow$ 3 Bits * 18 Zeichen = 54 Bits