Fakultät IV Elektrotechnik und Informatik
Informatik 1 WS 2001/02, Jähnichen / Flik / Lechtchinsky / Neunast / Tutoren

Aufgabenblatt 8

letzte Aktualisierung: 16. Januar, 14:29

Ausgabe: *11.01.2002*
Abgabe: *21./22.1.2002* *Prozent:* *100*

Thema: *Codesicherung; Zahlendarstellung, Rechnen mit Dualzahlen*

1. Aufgabe (35 Prozent): Codesicherung

1.1. Paritätsbildung (Tut) Ermittelt die ASCII-Codierung des Wortes „STEIN“. Ergänzt die Codewörter um die gerade Querparität. Welche Fehler sind erkennbar bzw. korrigierbar?
Ergänzt die Codewörter nun um einen Block-Check-Character mit gerader Längsparität. Welche Fehler sind dann erkennbar bzw. korrigierbar?

Hinweis: ASCII-Code für 'A' = 41_{16}, 'B' = 42_{16} usw.

Lösung:
Die Paritätsbildung dient zur Fehlererkennung. Bei 7-Bit-Code (z.B. ASCII) kann das achte Bit als Paritätsbit verwendet werden. Bei gerader Parität wird das Paritätsbit immer so gesetzt, dass pro 8-Bit-Wort eine gerade Anzahl von Einsen vorhanden ist, bei ungerader Parität entsprechend für eine ungerade Anzahl.

Fehler erkennen heißt, falsche Codewörter von richtigen zu unterscheiden; Fehler korrigieren dagegen, die Position der fehlerhaften Bits zu bestimmen.

```
 | Querparitaet
 V
|0| 1 0 1 0 0 1 1
|1| 1 0 1 0 1 0 0
|1| 1 0 0 0 1 0 1
|1| 1 0 0 1 0 0 1
|0| 1 0 0 1 1 1 0
```

Somit sind 1-Bit-Fehler, sowie Mehr-Bit-Fehler in ungerader Anzahl erkennbar. Es sind aber keine Fehler korrigierbar.

```
 | Querparitaet
 V
|0| 1 0 1 0 0 1 1
|1| 1 0 1 0 1 0 0
|1| 1 0 0 0 1 0 1
|1| 1 0 0 1 0 0 1
|0| 1 0 0 1 1 1 0
 ----------------
  1  1 0 0 0 1 0 1 <- Laengsparitaet
```

Jetzt sind 1-Bit-Fehler korrigierbar. Mehr-Bit-Fehler sind in gewissen Umfang erkennbar, aber nicht korrigierbar.

1.2. Hamming-Distanz (Tut) Was versteht man unter Hamming-Distanz? Welche Fehler können erkannt bzw. korrigiert werden? Wie groß ist die Hamming-Distanz beim ASCII bzw. einem Code mit Paritätsbit?

Lösung:
Die Hamming-Distanz h gibt an, um wieviel Bits sich zwei Codewörter mindestens unterscheiden.

```
Beispiel:   0000 0000
            0000 0111
            0011 1000
            1100 0001
            0001 1110
```

Im Beispiel-Code unterscheiden sich alle Zeichen um mindestens drei Bitpositionen, die Hamming-Distanz beträgt also 3.

Hinweis: *Es ist möglich $h-1$ Fehler zu erkennen und $(h-1)/2$ (h ungerade) bzw. $h/2-1$ (h gerade) zu korrigieren.*

Bei ASCII ist $h = 1$, bei Parität $h = 2$.

1.3. Cyclic Redundancy Check (Tut) Erzeugt mit dem Generatorpolynom $G(x) = x^4 + x^3 + 1$ die CRC-Prüfbits für die Bitfolge 01110101.

Lösung:

1. *Die n Bits eines Datenstroms werden als Koeffizienten eines Polynoms $D(x)$ der Ordnung $n-1$ betrachtet:*

   Bitfolge 01110101 $\rightarrow$ *Polynom* $D(x) = x^6 + x^5 + x^4 + x^2 + 1$

2. *Es werden vier Nullen an die Bitfolge gehängt, was einer Multiplikation mit x^4 entspricht:*

   $011101010000 \rightarrow x^{10} + x^9 + x^8 + x^6 + x^4$

3. *Es erfolgt die Division durch das Generatorpolynom $G(x)$, wobei ein 4-Bit-Restpolynom $R(x)$ als Prüfinformation entsteht:*
   Hinweis: *Die für die Divison erforderliche Substraktion wird als Modulo-2-Substraktion ausgeführt (z.B. $3x^2 - x^2 \rightarrow 0x^2$ aber $2x^2 - x^2 \rightarrow x^2$).*

$$
\begin{array}{llllll}
(x^{10}+x^9 & +x^8 & & +x^6 & +x^4) : (x^4+x^3+1) = x^6+x^4-x^3+x^2-x+1 \\
-(x^{10}+x^9 & & & +x^6) & \\
\hline
 & +x^8 & & +x^4) & \\
 & -(x^8 & +x^7 & +x^4) & \\
\hline
 & & -x^7 & & \\
 & & -(-x^7 & -x^6 & -x^3) \\
\hline
 & & & +x^6+x^3 & \\
 & & & -(x^6 & +x^5+x^2) \\
\hline
 & & & & -x^5+x^3-x^2 \\
 & & & & -(-x^5-x^4-x) \\
\hline
 & & & & +x^4+x^3-x^2+x \\
 & & & & -(x^4+x^3+1) \\
\hline
 & & & & -x^2+x-1 \\
\hline\hline
\end{array}
$$

$R(x) = -x^2 + x - 1 \rightarrow 0111$

4. *$R(x)$ wird zu $D(x)$ addiert:*

   $x^{10} + x^9 + x^8 + x^6 + x^4 - x^2 + x - 1 \rightarrow 011101010111$

5. *Ein Decodierer auf der Empfängerseite dividiert $D(x) + R(x)$ durch $G(x)$ und erwartet bei fehlerfreier Übertragung den Rest Null.*

1.4. Cyclic Redundancy Check (35 Prozent) Überprüft die folgenden Codewörter mit Hilfe der Zyklischen Blockprüfung (4-Bit-Restpolynom, Generatorpolynom $G(x) = x^4 + x^3 + 1$): 110011111101, 10101110, 1101000100.
Welche Codewörter sind fehlerhaft?

Lösung:

$$
\begin{array}{llllllll}
 & (x^{11}+x^{10}+x^7 & & +x^6+x^5+x^4 & -x^3 & -x^2 & +1) : (x^4+x^3+1) = x^7+x^2+1 \\
- & (x^{11}+x^{10}+x^7) & & & & & & \\
\hline
 & & & +x^6+x^5+x^4 & +x^3 & +x^2 & +1 & \\
 & & & -(x^6+x^5 & & +x^2) & & \\
\hline
 & & & +x^4 & +x^3 & & +1 & \\
 & & & -(x^4 & +x^3 & & +1) & \\
\hline
 & & & & & & 0 & \\
\hline\hline
\end{array}
$$

Die Übertragung war korrekt, da der Rest Null beträgt.

$$
\begin{array}{llllllll}
 & (x^7 & & +x^5 & +x^3 & +x^2 & +x) : (x^4+x^3+1) = x^3-x^2 \\
- & (x^7 & +x^6 & & +x^3) & & & \\
\hline
 & & -x^6 & +x^5 & & +x^2 & +x & \\
 & - & (-x^6 & -x^5 & & -x^2) & & \\
\hline
 & & & & & & x & \\
\hline\hline
\end{array}
$$

Die Übertragung war fehlerhaft, da der Rest nicht Null beträgt.

$$
\begin{array}{llllll}
 & (x^9+x^8+x^6 & & & +x^2) : (x^4+x^3+1) = x^5+x^2 \\
- & (x^9+x^8 & & +x^5 & & \\
\hline
 & x^6 & & -x^5 & +x^2 & \\
 & -(x^6 & & +x^5 & +x^2) & \\
\hline
 & & & & 0 & \\
\hline\hline
\end{array}
$$

Die Übertragung war korrekt, da der Rest Null beträgt.

2. Aufgabe (30 Prozent): Dualzahldarstellung

2.1. Umwandlung (Tut) Konvertiert 7_{10} und 13_{10} in Dualzahlen.

Lösung:

$7_{10} \rightarrow$	$7/2 = 3$	*Rest* 1	
	$3/2 = 1$	*Rest* 1	
	$1/2 = 0$	*Rest* 1	$\rightarrow 0111_2$
$13_{10} \rightarrow$	$13/2 = 6$	*Rest* 1	
	$6/2 = 3$	*Rest* 0	
	$3/2 = 1$	*Rest* 1	
	$1/2 = 0$	*Rest* 1	$\rightarrow 1101_2$

2.2. Umwandlung (10 Prozent) Konvertiert 43_{10} und 19_{10} in Dualzahlen.

Lösung:

$19_{10} \rightarrow$	$19/2 = 9$	*Rest* 1	
	$9/2 = 4$	*Rest* 1	
	$4/2 = 2$	*Rest* 0	
	$2/2 = 1$	*Rest* 0	
	$1/2 = 0$	*Rest* 1	$\rightarrow 10011_2$
$43_{10} \rightarrow$	$43/2 = 21$	*Rest* 1	
	$21/2 = 10$	*Rest* 1	
	$10/2 = 5$	*Rest* 0	
	$5/2 = 2$	*Rest* 1	
	$2/2 = 1$	*Rest* 0	
	$1/2 = 0$	*Rest* 1	$\rightarrow 101011_2$

2.3. Addition (Tut) Addiert 01111_2 und 101_2. Das Ergebnis soll in eine Dezimalzahl umgewandelt werden.

Lösung:

$$
\begin{array}{llllll}
 & 0 & 1 & 1 & 1 & 1 \\
+ & 0 & 0 & 1 & 0 & 1 \\
 & \bullet & \bullet & \bullet & \bullet & \\
\hline
 & 1 & 0 & 1 & 0 & 0 \\
\hline\hline
\end{array}
$$

$10100_2 = 1 \cdot 2^4 + 1 \cdot 2^2 = 20_{10}$

2.4. Subtraktion (Tut) Subtrahiert 10110_2 von 01010_2. Interpretiert das Ergebnis. Welches Bedingungsbit wird gesetzt?

Lösung:

$$
\begin{array}{llllll}
 & 0 & 1 & 0 & 1 & 0 \\
- & 1 & 0 & 1 & 1 & 0 \\
\bullet & & \bullet & & & \\
\hline
 & 1 & 0 & 1 & 0 & 0 \\
\hline\hline
\end{array}
$$

Es tritt eine Bereichsüberschreitung (Übertrag) aus. Aus diesem Grund wird das Carry-Bit c gesetzt.

2.5. Addition (10 Prozent) Addiert 101000_2 und 10001_2. Das Ergebnis soll in eine Dezimalzahl umgewandelt werden.

Lösung:

```
  101000
+ 10001
---------
  111001
```

$111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^0 = 57_{10}$

2.6. Subtraktion (10 Prozent) Subtrahiert 101010_2 von 1011100_2 und 100101_2 von 100000_2.

Lösung:

```
  1011100
-  101010
----------
   110010
```

```
  100000
- 100101
---------
  111011
```

3. Aufgabe (35 Prozent): Negative Dualzahlen

3.1. Darstellung negativer Zahlen (Tut) Welche Möglichkeiten gibt es, negative Zahlen im Computer darzustellen und wie werden sie gebildet?

Lösung:

- *Vorzeichenzahlen:*
  höchstwertiges, n-tes Bit als Vorzeichenbit: 0 = pos., 1 = neg., restliche $n-1$ Bits = Dualzahl (Betrag)

  Beispiel: $-1_{10} = 10000001_2$.

- *1-Komplement-Zahlen:*
  Bitweises invertieren der positiven Dualzahl
  → höchstwertiges, n-tes Bit impliziert das Vorzeichen: 0 = pos., 1 = neg.

  Beispiel: $-1_{10} = 11111110_2$.

- *2-Komplement-Zahlen:*
  Bitweises invertieren der positiven Dualzahl und anschließendes Addieren von „1“
  → höchstwertiges, n-tes Bit impliziert das Vorzeichen: 0 = pos., 1 = neg.

  Beispiel: $-1_{10} = 11111111_2$, *denn*
  $-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = -1$.

Häufig wird die 2-Komplementdarstellung zur Repräsentation von negativen Zahlen verwendet. Dies hat den Vorteil, dass die Grundrechenarten zwischen negativen und positiven Zahlen ohne Probleme durchgeführt werden können. Es muss natürlich beachtet werden, dass das höchstwertiges Bit ein negatives Gewicht hat.

3.2. Umwandlung (Tut) Wandelt 10101010 in eine Dezimalzahl um und berechnet von -5 (Dezimalsystem) das zugehörige 2-Komplement (8 Stellen).

Lösung:

$10101010_2 \rightarrow$
$-2^7 + 2^5 + 2^3 + 2^1 = -128 + 32 + 8 + 2$
$\rightarrow -86_{10}$

$5_{10} \rightarrow 00000101_2$

```
                      00000101
Invertierung:         11111010
Addition von Eins: +          1
                      ---------
Ergebnis:             11111011
```

3.3. Umwandlung (10 Prozent) Wandelt 11100110 in eine Dezimalzahl um und berechnet von -13 (Dezimalsystem) das zugehörige 2-Komplement mit vier Stellen. Was fällt euch auf?

Lösung:

$11100110_2 \rightarrow$
$-2^7 + 2^6 + 2^5 + 2^2 + 2^1 = -128 + 64 + 32 + 4 + 2$
$\rightarrow -26_{10}$

$13_{10} \rightarrow$
$13/2 = 6$ *Rest* 1
$6/2 = 3$ *Rest* 0
$3/2 = 1$ *Rest* 1
$1/2 = 0$ *Rest* 1
$\rightarrow 1101_2$

```
                      1101
Invertieren:          0010
Addition von Eins:  +    1
                      -----
                      0011

Ergebnis: 0011 (4 Stellen)
```

Das Ergebnis kann natürlich nicht korrekt sein, da zur Darstellung der 13_2 (ohne Vorzeichenbit) schon 4 Bit notwendig sind. Für die 2-Komplement-Darstellung von -13 sind also mindestens 5 Bit erforderlich. Mit 4 Stellen repräsentiert die 2-Komplement-Zahl 1101 *also eine -3 und nicht* 13.

3.4. Zahlenbereich (10 Prozent) Welcher Zahlenbereich kann durch 10 bzw. 12 Bit 2-Komplement dargestellt werden? Gebt die kleinste und größte Zahl sowohl in Dezimal- als auch in Binärdarstellung an.

Lösung:

10 Bit:
Kleinste Zahl: $1000000000_2 = -2^9 = -512_{10}$
Größte Zahl: $0111111111_2 = 2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 == 2^9 - 1 = 511_{10}$

12 Bit:
Kleinste Zahl: $100000000000_2 = -2^{11} = -2048_{10}$
Größte Zahl: $011111111111_2 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 2^{11} - 1 = 2047_{10}$

3.5. Rechnen mit dem 2-Komplement (15 Prozent) Addiert und subtrahiert 01111010_2 und 10101100_2 (8 Bit 2-Komplement). Macht die Probe im Dezimalsystem. Treten Bereichsüberschreitungen auf? Welches Bedingungsbit wird gesetzt?

Lösung:

	0	*1*	*1*	*1*	*1*	*0*	*1*	*0*
+	*1*	*0*	*1*	*0*	*1*	*1*	*0*	*0*
	•	•	•	•				
	0	*0*	*1*	*0*	*0*	*1*	*1*	*0*

Das Ergebnis lautet: 00100110.

$01111010_2 = 2^6 + 2^5 + 2^4 + 2^3 + 2^1 = +122_{10}$
$10101100_2 = -2^7 + 2^5 + 2^3 + 2^2 = -84_{10}$

$+122 + (-84) = +38 = 00100110_2$
Das Ergebnis ist korrekt, es tritt keine Bereichsüberschreitung auf.

	0	*1*	*1*	*1*	*1*	*0*	*1*	*0*
-	*1*	*0*	*1*	*0*	*1*	*1*	*0*	*0*
•				•	•			
	1	*0*	*0*	*0*	*1*	*1*	*1*	*0*

$+122 - (-84) = +206 \neq 10001110_2$

Das Ergebnis ist nicht korrekt, es wird der Zahlenbereich der 8-Bit-2-Komplementzahlen überschritten (-128 bis +127). Das v-Bit wird gesetzt.