Zahlensysteme

Inhalt

1. Analoge und digitale Signale 2
Vor- und Nachteile digitaler Systeme 2

2. Codierung 3
Codierungsvorschriften 3

3. Zahlensysteme 6
Gewichteter Code 6
Binär ↔ Hex-Wandlung 7
Dezimal → Binär 7
Dezimal → Hex 7
Ganzzahlenarithmetik 8
Das Einerkomplement (EK) 8
Das Zweierkomplement (ZK) 8
Der Zahlenkreis 9
Addition/Subtraktion von Binärzahlen in Zweierkomplement-Darstellung 10
Weitere Zahlencodes 11
Sign Magnitude Darstellung 11
Offset Binary Darstellung 12
Darstellung von Gleitkommazahlen 12

Anhang 14
Zehner-Komplement (TK) 14

Quellen 15

1. Analoge und digitale Signale

Analoge Signale
Die Amplitudenwerte eines Signals können beliebige Werte annehmen. Das Signal gibt zu jedem beliebigen Zeitpunkt den exakten Wert wieder.
In der analogen Welt existieren Werte- und Zeitkontinuum.

Digitale Signale
Die Amplitudenwerte eines Signals sind nötigenfalls diskretisiert (digitalisiert). Die Zeit ist quantisiert. Innerhalb eines Zeitquants ist ein Signal konstant aufgefasst.
In der digitalen Welt existieren kein Werte- und kein Zeitkontinuum.

Vor- und Nachteile digitaler Systeme

+ Werte sind endlich abzählbar.
+ Ereignisse über eine Dauer lassen sich mit endlich vielen Zeitintervallen beschreiben.
– Signale sind nicht absolut genau bezüglich Werte und Zeiten (können jedoch mit entsprechendem Aufwand angenähert werden).
– Signale benötigen (in der Verarbeitung) mehr Zeit, da z.B. A/D- und D/A-Wandlung nötig ist.

Liste nicht abschliessend.

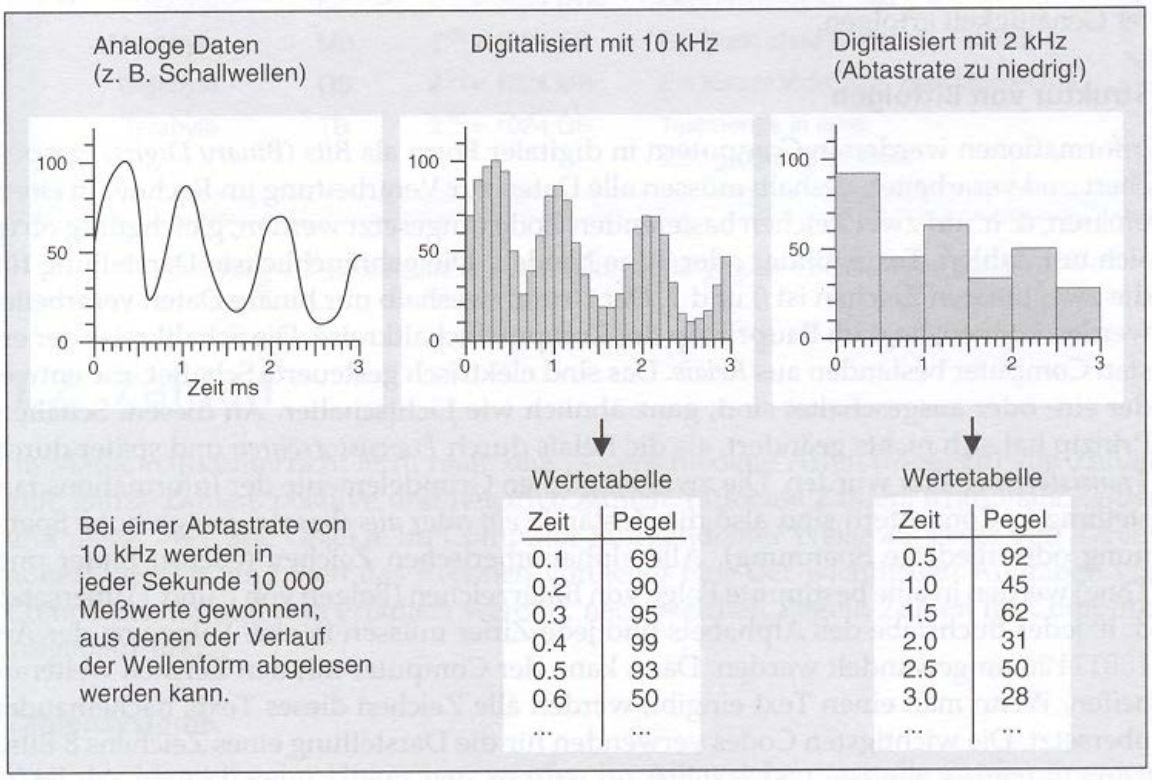


Abb. 1: Digitalisierung von Signalen. [1]

2. Codierung

Eine Codevorschrift (Codierung) bildet die Zeichen mit einem ersten Zeichenvorrat auf Zeichen mit einem zweiten Zeichenvorrat ab.
Es kann zwischen numerischen und nicht-numerischen Darstellungen unterschieden werden (z.B. ASCII-Zeichen).

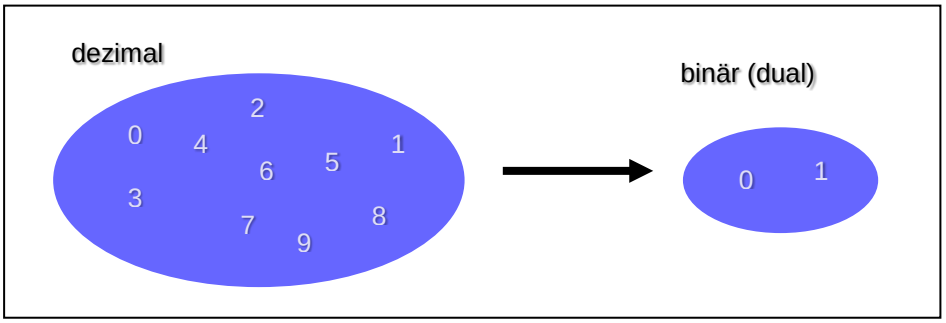


Abb. 2: Zeichenvorräte

Codierungsvorschriften
können folgende Punkte beinhalten:
- Reihenfolge der Zeichen wichtig
- Vorschrift basiert auf einem mathematischen Abbildungsgesetz
- Abbildungsliste
- etc.

Codes sollen dekodierbar sein, so dass wieder die ursprünglichen Zeichen gewonnen werden können.
Für unterschiedliche Anwendungen sind unterschiedliche Codes geeignet. Im Folgenden werden einige Codes vorgestellt.

Codes verfolgen unterschiedliche Ziele:
- (Vereinfachtes) Festhalten einer Situation
- Geheimhaltung
- Komprimierung von Daten
- etc.
- und Kombinationen davon

dez	dual	hex
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

dez	dual	hex
17	10001	11
...	...	...
30	11110	1E
31	11111	1F
32	100000	20
...	...	...
100	1100100	64
200	11001000	C8
...	...	...
254	11111110	FE
255	11111111	FF
256	100000000	100
257	100000001	101
...	...	...
1000	1111101000	3E8
1023	1111111111	3FF
...	...	...

Abb. 3: Codierung von Zahlen in unterschiedlichen Zahlensystemen

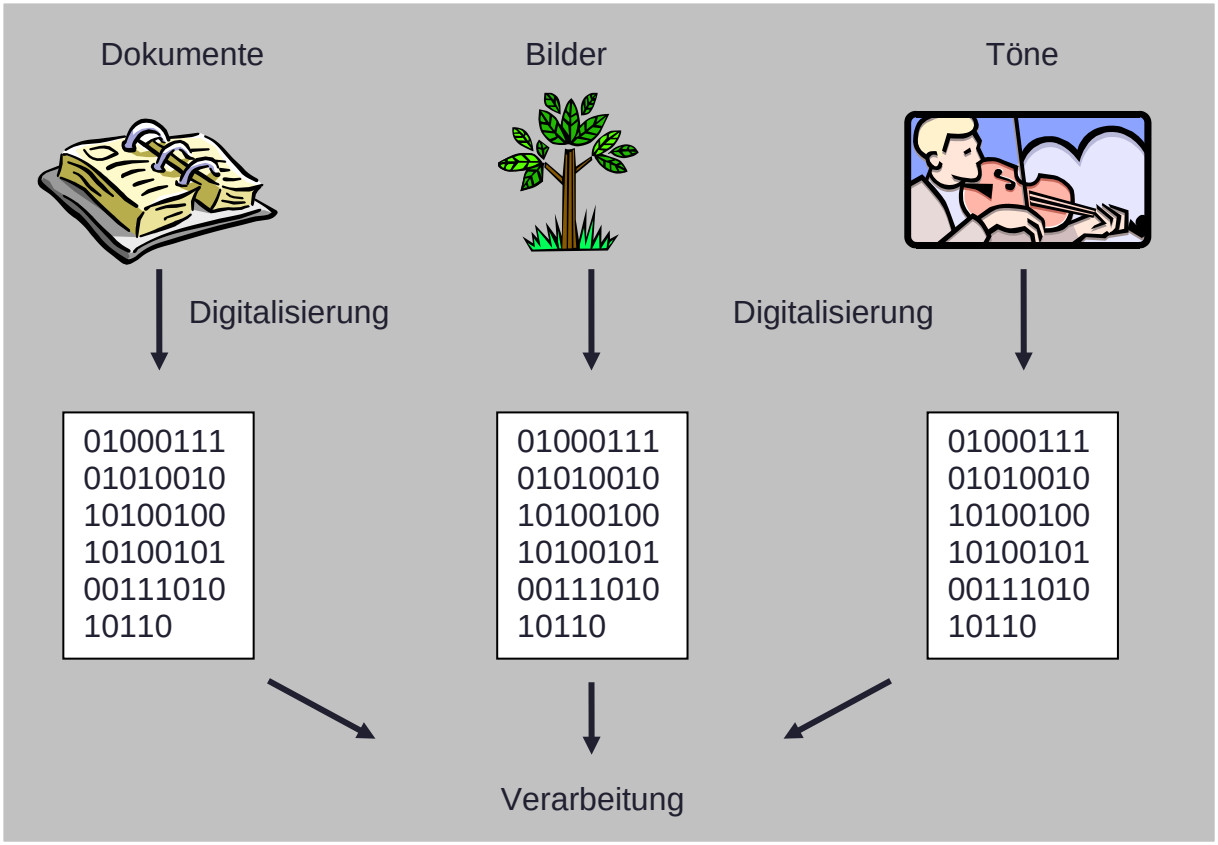


Abb. 4: Codierbeispiele unterschiedlicher Quellen. [1]

LS Nibble (LS Halfbyte)

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

← MS Nibble (MS Halfbyte)

LS: least significant
MS: most significant

Herauslesen der Hex-Codierung von ASCII-Zeichen am Beispiel von 'b':

MS Nibble LS Nibble
6 2

'b' → 0x62

Abb. 5: ASCII-Tabelle [2]

In der Digitaltechnik und in Rechnersystemen hat sich zumeist das Dual-System (binäre Darstellung) durchgesetzt. Wie weiter oben bereits gezeigt, wird mit Bitfolgen gearbeitet, d.h. Codierungen bestehend aus den Zeichen 0 und 1.

Zur Kennzeichnung von Bitfolgen werden oftmals Typen eingeführt:
- Datentypen
- Erweiterungen bei Dateinamen (Extension)
- Protokolltypen
- etc.

Typen lassen sich als 'Etikett' auffassen, die die Bitfolge charakterisieren.

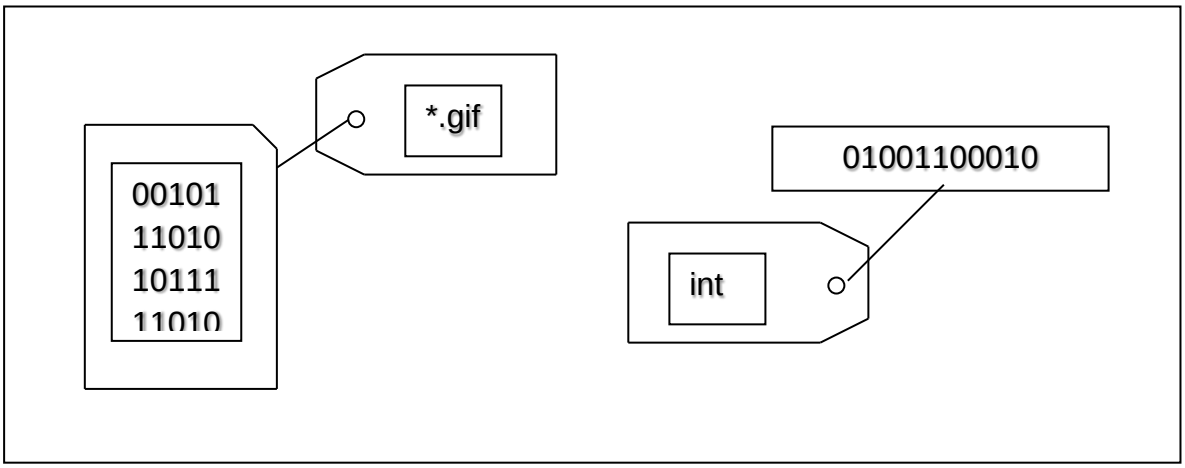


Abb. 6: Typisierung von Daten

3. Zahlensysteme

Eine Codierung, um die Menge abzählbarer Dinge festzuhalten.

Wie Julius und Brutus notierten:
VII
XCV
MLIV

Gewichteter Code

Symbole werden aneinander gereiht.
- Jedes Symbol repräsentiert eine Stelle.
- Jede Stelle hat ein Gewicht.
- Die Stelle ganz rechts hat niedrigstes Gewicht, jene ganz links höchstes Gewicht.
- Eine Stelle ohne eigenen Beitrag wird mit Null (als Stellenhalter) dargestellt.
- Die max. Anzahl Stellen einer Zahl (Symbolfolge) wird als Wortlänge bezeichnet (Im Alltag werden führende Nullen oft weg gelassen, die Wortlänge kann dann nicht genau bestimmt werden).

Die schwere Geburt der Null

Die Null wurde lange nach den andern Zahlen erfunden. Die Römer kannten sie überhaupt nicht, die Babylonier konnten nicht mit ihr umgehen, erst die Inder erkannten das Potential dieser bizarren Zahl, die alleine nichts ist, aber andern zur Grösse verhelfen kann.
...

Herbert Cerutti [3]

Abb. 7:
Die Null als Stellenhalter und Gewicht

Gewichteter Code am Beispiel einer Dezimalzahl:

Zahl (z): 1205
Zahlenbasis (b): 10
Ziffernvorrat (c_i): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Wortlänge (k): 4

$1205 = 1 * 10^3 + 2 * 10^2 + 0 * 10^1 + 5 * 10^0$

$z = c_{i3} * b^3 + c_{i2} * b^2 + c_{i1} * b^1 + c_{i0} * b^0$

Allgemein:

$$z = \sum_{j=0}^{j=k-1} c_{i\,j} * b^j$$

Hex	Dez	Bin
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Jedes gewichtete Zahlensystem hat so viele Symbole (Ziffern) nötig, wie die Basis gross ist. Das Hexadezimale Zahlensystem (b = 16) benützt die Ziffern 0 bis 9 und A bis F.

Binär ↔ Hex-Wandlung

- Eine Binärzahl wird vorerst in Nibbles (Gruppen à 4 Bits) eingeteilt.
- Danach wird jedem Nibble die Ziffer mit dem entsprechenden Wert in Hex zugeordnet.

Beispiel:

1001'0011'1101'1111

9 3 D F

Wandlungen sind grundsätzlich besonders einfach, wenn zwischen Zahlensysteme mit Basen der Form 2^n zu wandeln ist:
Binär: n = 1
Oktal: n = 3
Hex: n = 4

Vorgehen: Zahl in binärer Darstellung notieren, ggf. von rechts nach links in Gruppen mit n Ziffern unterteilen. Jede Gruppe individuell in die Ziffer der neuen Basis wandeln.

Aufgabe: Wandeln Sie die Zahl $B7_{16}$ in die gleichwertige Zahl zur Basis 8.

Dezimal → Binär

Mehr Aufwand verlangt eine Wandlung z.B. von Dezimal nach Binär:

Fortlaufende Ganzzahl-Division der Dezimalzahl mit 2 und "Sammeln" der Reste führt zum gesuchten Resultat. Abbruchkriterium: Resultat liefert 0 (bei Rest 1 oder 0).

An einem Beispiel:

55 : 2 = 27	Rest: 1LSB (Least Significant Bit)
27 : 2 = 13	Rest: 1
13 : 2 = 6	Rest: 1
6 : 2 = 3	Rest: 0
3 : 2 = 1	Rest: 1
1 : 2 = 0 (Abbruch)	Rest: 1MSB (Most Significant Bit)

Also: $55_{10} = 11'0111_2$

Kontrolle:
$1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 32 + 16 + 0 + 4 + 2 + 1 = 55_{10}$

Dezimal → Hex

Wandlung von Dezimal nach Hexadezimal:

$181_{10} : 16_{10} = 11_{10}$	Rest: $5_{10} = 5_{16}$
$11_{10} : 16_{10} = 0$ (Abbruch)	Rest: $11_{10} = B_{16}$

Also: $181_{10} = B5_{16}$

Kontrolle:
$11_{10} * 16^1 + 5_{10} * 16^0 = 176_{10} + 5_{10} = 181_{10}$

Ganzzahlenarithmetik binärer Zahlen

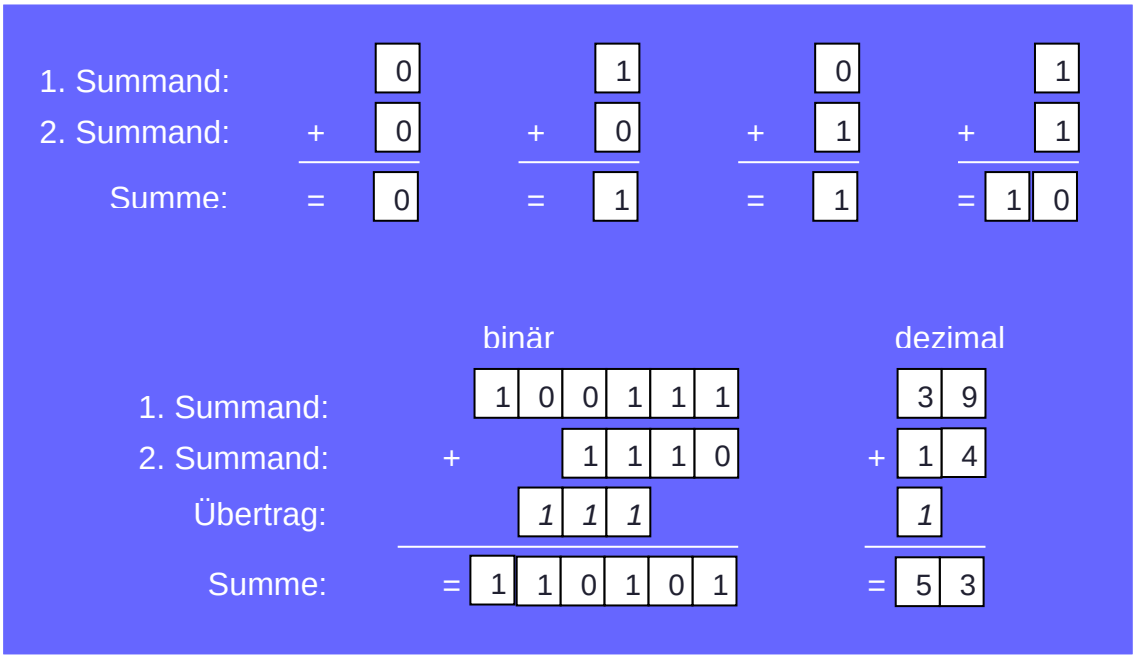


Abb. 8: Stellenaddition ohne/mit Uebertrag

Das Einerkomplement (EK), Ones' Complement

einer Binärzahl wird gebildet, indem stellenweise jede 1 durch eine 0, bzw. jede 0 durch eine 1 ersetzt wird.

Beispiel:
EK(1001'1011) = 0110'0100

Das Einerkomplement kann dazu verwendet werden, negative Ganzzahlen zu kodieren. Dazu muss vorerst die Wortlänge bekannt sein. Das MSB kennzeichnet das Vorzeichen (0: +, 1: -). Die Bits der *positiven* Zahl (also mit MSB = 0) kodieren den Wert in der oben vorgestellten Weise. Beispiel bei der Wortlänge 8 (8 Bits):

$EK(0110'0100_2) = 1001'1011_2$

$0110'0100_2 = 100_{10}$ $\quad 1001'1011_2 = -100_{10}$

Da die Null (bei Wortlänge 8) sowohl als $0000'0000_2$ wie auch als $1111'1111_2$ dargestellt werden kann, eignet sich diese Codierung schlecht für Addierschaltungen. Besser geeignet dazu ist:

Das Zweierkomplement (ZK), Two's Complement

Das Zweierkomplement (ZK) einer Binärzahl wird gebildet, indem vorerst das Einerkomplement gebildet und danach 1 addiert wird.

Beispiel:
ZK(1001'1011) = EK(1001'1011) + 1 =
0110'0100 + 1 = 0110'0101

Wie beim Einerkomplement muss die Stellenzahl bekannt sein, da sonst Fehlinterpretationen auftreten können.

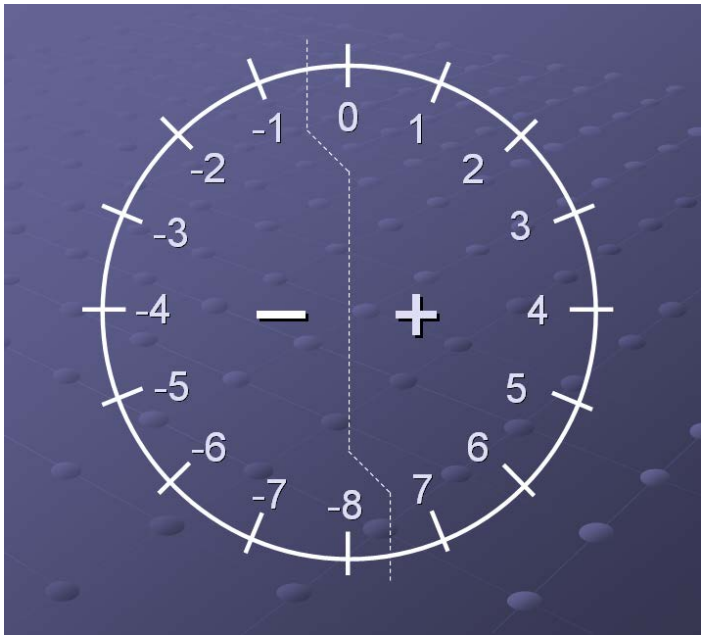
Beispiel:
Stellen Sie die Dezimalzahl 49 als Binärzahl mit der Stellenzahl k = 8 dar.

$49_{10} = 32_{10} + 16_{10} + 1_{10} = 0011'0001_2$

Ohne Angabe k = 8 ist die Anzahl führender 0 unklar!

Beispiel Fortsetzung:
Bilden Sie das ZK obiger Binärzahl
$ZK(0011'0001_2) = EK(0011'0001_2) + 1 =$
$1100'1110_2 + 1 =$
$1100'1111_2$

Der Zahlenkreis



Zahlenkreis für das Zweierkomplement

Statt binärer Zahlen (Wortlänge 4) sind nebenstehend dezimale Zahlen notiert.
Diese Darstellung zeigt die Spiegelung der Beträge, im linken Halbkreis mit negativem, im rechten Halbkreis mit positivem Vorzeichen.
Beachten Sie die Spezialfälle 0 und -8.

Unten:
Notation der binären Zahlen zur Hervorhebung des MSB = 1.

Abb. 9: Der Zahlenkreis in Zweierkomplement-Darstellung mit Dezimalzahlen beschriftet

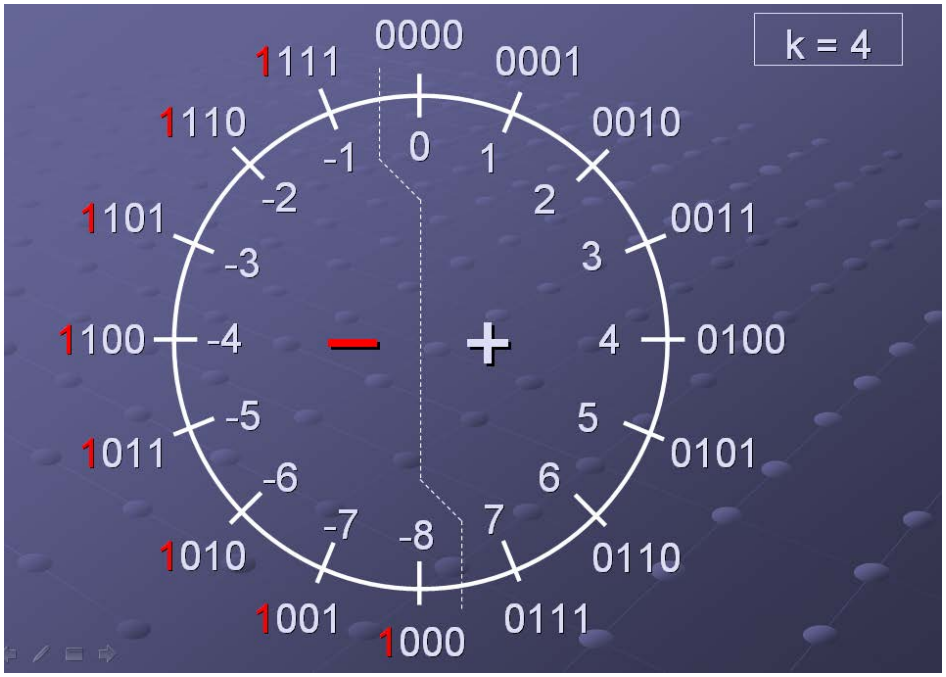


Abb. 10: Der Zahlenkreis in Zweierkomplement-Darstellung mit Dezimal- und Binärzahlen beschriftet.

Addition/Subtraktion von Binärzahlen in Zweierkomplement-Darstellung

Das Zweierkomplement lässt die Realisierung von Schaltungen zu, die eine Addition und Subtraktion auf einfache Weise zulässt. Der Zahlenkreis wiederum veranschaulicht auf einfache Weise, welchen Restriktionen die Operationen unterworfen sind.

Fallbetrachtungen für unterschiedliche Situationen werden nachfolgend dargestellt:

Fallbetrachtungen für **k = 4**:

a) 0100 + 0011 = 0111 pos + pos = pos √
4 + 3 = 7

b) 0100 + 0101 = 1001 pos + pos = neg f
4 + 5 = -7

c) 0100 + 1101 = ~~1~~0001 pos + neg = pos √
4 + -3 = 1

d) 0100 + 1010 = 1110 pos + neg = neg √
4 + -6 = -2

e) 1100 + 1101 = ~~1~~1001 neg + neg = neg √
-4 + -3 = -7

f) 1100 + 1011 = ~~1~~0111 neg + neg = pos f
-4 + -5 = 7

g) 1100 + 0011 = 1111 neg + pos = neg √
-4 + 3 = -1

h) 1100 + 0110 = ~~1~~0010 neg + pos = pos √
-4 + 6 = 2

Zusammenfassung:
Die Resultate bei der Addition/Subtraktion sind falsch, wenn es zu einer Bereichsüberschreitung kommt. Eine Bereichsüberschreitung findet (für k = 4) statt, wenn vom positiven in den negativen Halbkreis via 7
zu -8 oder umgekehrt gewechselt wird.

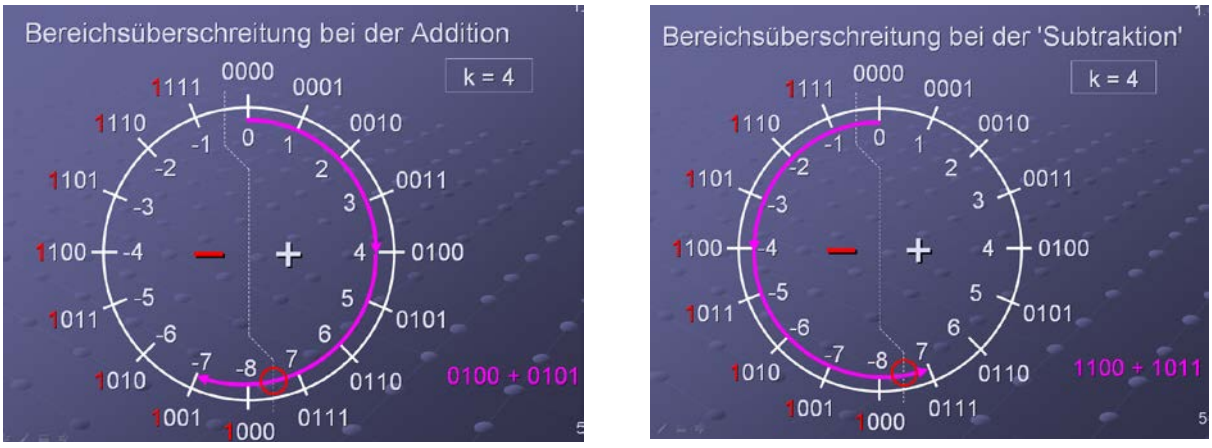


Abb. 11: Bereichsüberschreitungen am Zahlenkreis (k = 4) illustriert.

Alle gezeigten Fälle können nun leicht auf andere Wortlängen übertragen werden. Gängige Wortlängen sind k = 8, 16, 32, 64.

Aufgaben:
- Zeichnen Sie einen Zahlenkreis für k = 4 für nur positiv zu interpretierende Zahlen (unsigned int).
- Zeigen Sie im Kreis an einem Beispiel, wo es zu Bereichsüberschreitungen kommen kann.
- Begründen Sie, weshalb es sich um eine Bereichsüberschreitung handelt.

Weitere Zahlencodes

Gray-Code (*ein*schrittiger Code, ungewichteter Code). Eingesetzt bei Längen- und Winkelcodierungen, um die Ablesefehler zu verringern: bei einem Wechsel ändert nur 1 Bit.

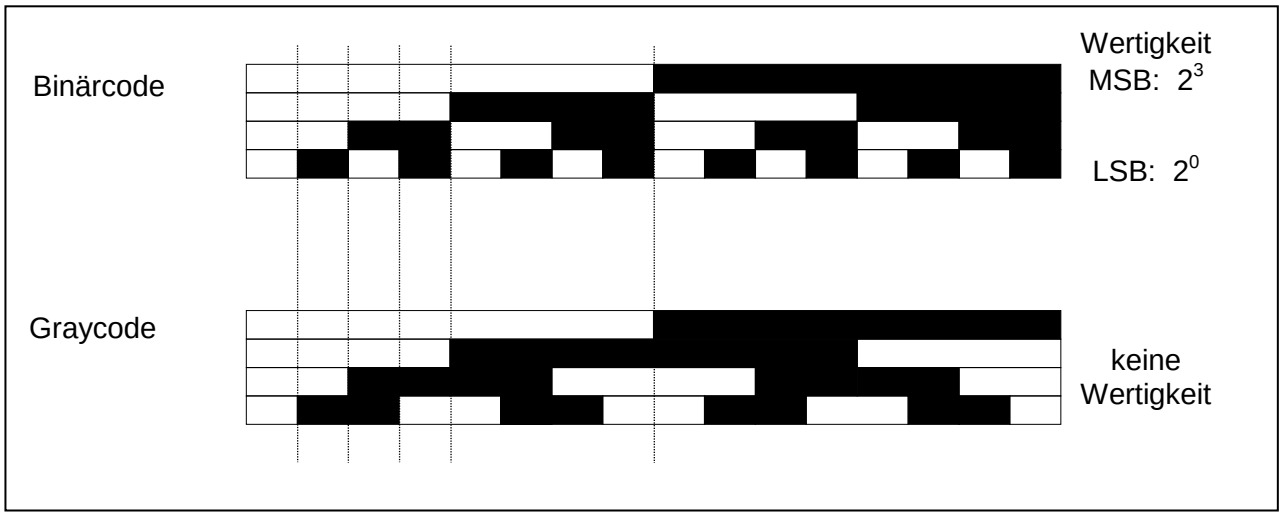


Abb. 12: Binär- vs. Graycode (k = 4).

Aufgabe:
Erstellen Sie die Tabelle des Graycodes für die Zahlen 0_{10} bis 15_{10} in der Zahlenbasis 2, k = 4.

Binary Coded Decimal (BCD) ist ein Code, der jede Dezimalziffer einzeln als 4-Bit-Zahl kodiert.
Beispiel:
$239_{10} = 0010'0011'1001_{BCD}$
Oftmals für Dezimalanzeigen eingesetzt.

Sign Magnitude Darstellung

Bei dieser Codierung übernimmt das MSB ausschliesslich die Rolle des Vorzeichenbits (Sign-Bit, SB), die restlichen Bits dienen zur Codierung des Betrags der Ganzzahl Z ($|Z|$) als Magnitude M bezeichnet.

Binary	Signed	Unsigned
0000'0000	+0	0
0000'0001	1	1
...	...	...
0111'1111	127	127
1000'0000	−0	128
1000'0001	−1	129
...	...	...
1111'1111	−127	255

Abb. 13: 8-Bit Signed Magnitude (mittlere Spalte), eingesetzt in einigen frühen Computern.

Offset Binary Darstellung (auch Excess-N Darstellung)

In dieser Darstellung wird ein Offset (Bias) verwendet, um negative Zahlen in den positiven Bereich zu verschieben. Ein praktisches Beispiel findet sich bei den Skalen, die Temperaturen in °C oder in Kelvin angeben. Diese Darstellung wird heute vorzugsweise bei der Codierung von Gleitkomma-Zahlen verwendet.

Bsp.

Dezimal	Binär	Dezimal	Binär	Dezimal	Binär	Dezimal	Binär
-3	0000	1	0100	5	1000	9	1100
-2	0001	2	0101	6	1001	10	1101
-1	0010	3	0110	7	1010	11	1110
0	0011	4	0111	8	1011	12	1111

Abb. 14: Excess-3 Binär-codierte Dezimalen.

Darstellung von Gleitkommazahlen (Floating point)

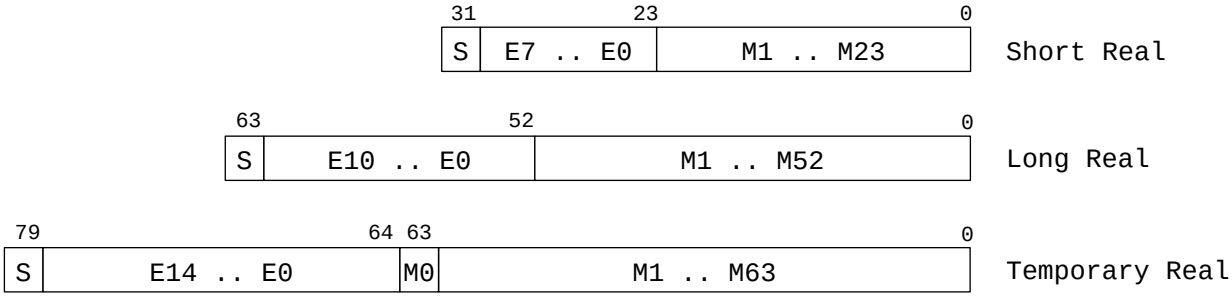


Abb. 15: Bsp. von Datentypen für Gleitkommazahlen am Beispiel des math. Coprozessors 8087. S: Vorzeichen, E: Exponent, M: Mantisse

Datentyp	Signifikante Dezimalstellen	Bereich
Word Integer	4	-32'768 .. 32767
Short Integer	9	$-2 * 10^{9}$ .. $2 * 10^{9}$
Long Integer	18	$-9 * 10^{18}$ .. $9 * 10^{18}$
Short Real	6 .. 7	$\pm 8.43 * 10^{-37}$ .. $\pm 3.37 * 10^{-38}$
Long Real	15 .. 16	$\pm 4.19 * 10^{-307}$ .. $\pm 1.67 * 10^{-308}$
Temporary Real	19	$\pm 3.4 * 10^{-4932}$ .. $\pm 1.2 * 10^{-4932}$
Packed BCD	18	-999999999999999999 .. 999999999999999999

Abb. 16: Alle Datentypen des math. Coprozessors 8087. Signifikante Dezimalstellen, Bereich.

Codierung:
Der Datentyp Real ist mit einem Bias versehen:

$Z = (-1)^{S} * (2^{E - BIAS}) *(M0 \bullet M1 \bullet M2 \bullet ...)$

Bias für verschiedene Datentypen:
- Short Real 127_{10}
- Long Real 1023_{10}
- Temporary Real 16383_{10}

Berechnungsbeispiele für Z im Format Short Real in binärer Darstellung:

Beispiel 1:
$Z_{10} = -1{,}875_{10}$

Vorzeichen S: $S = 1_2$

Exponent E: $1{,}875_{10} = 2^0 + 2^{-1} + 2^{-2} + 2^{-3}$ $\rightarrow$ höchster Exponent: 0
E - BIAS = 0
$E = 127_{10} = 0111'1111_2$

Mantisse M: 11'1000..., für
$M_1 = 1$ für 2_{-1}
$M_2 = 1$ für 2_{-2}
$M_3 = 1$ für 2_{-3}
Alle andern M_i sind 0. M_0 (= 1) wird nicht mitgegeben.

$Z_2 = 1\ \ 011'1111'1\ \ 111'0000\ \ 0000'0000\ \ 0000'0000_2$

Kontrolle:
$Z = (-1)^1 * (2^{127 - 127}) *(M0 \bullet 1 \bullet 1 \bullet 1 \bullet 0 \bullet 0 \bullet 0 \bullet ...) = -1 * 1 * (1 + 0{,}5 + 0{,}25 + 0{,}125) = -1{,}875_{10}$

Beispiel 2:
$Z_{10} = 18{,}75_{10}$

Vorzeichen S: $S = 0_2$

Exponent E: $18{,}75_{10} = 2^4 + 2^1 + 2^{-1} + 2^{-2}$ $\rightarrow$ höchster Exponent: 4_{10}
E - BIAS = 4_{10}
$E = 131_{10} = 1000'0011_2$

Mantisse M: 0010'1100..., für
$M_1 = 0$ für 2^3
$M_2 = 0$ für 2^2
$M_3 = 1$ für 2^1
$M_4 = 0$ für 2^0
$M_5 = 1$ für 2^{-1}
$M_6 = 1$ für 2^{-2}
Alle andern M_i sind 0. M_0 (= 1) wird nicht mitgegeben.

$Z_2 = 0\ \ 100'0001'1\ \ 001'0110\ \ 0000'0000\ \ 0000'0000_2$

Kontrolle:
Als Aufgabe.

Anhang

Zehner-Komplement (TK), Tens' Complement

Die günstige Komplementbildung kann grundsätzlich in jeder Zahlenbasis vorgenommen werden. Dies soll am Zehnerkomplement der Dezimalzahlen gezeigt werden.

Beispiele:

```
  873
 -218
 ----
  655

Wortlänge k = 3

  873
+ 781   (NK(218): Neunerkomplement
         ziffernweise Ergänzung zu 9)
+   1    Zehnerkomplement bilden
-----
 1655

-1000   Uebertrag entfernen
-----
  655
```

```
  034
 -  9
 ----
   25

Wortlänge k = 3

  034
+ 990

+   1
-----
 1025

-1000
-----
  025
```

Abb. 17: Neunerkomplement der Zahlen 218_{10} und 9_{10} der Wortlänge 3;

```
 -15
 -21
 ---
 -36   NK(36) = 63; TK(36) = 64

Wortlänge k = 2

  85   NK(15) = 84; TK(15) = 85
 +79   NK(21) = 78; TK(78) = 79
 ---
 164

-100   Uebertrag entfernen
   0   Zehnerkomplement bereits gebildet
 ---
  64   NK(64) = 35; TK(64) = 36; -> -36
```

Abb. 18: Addieren mit Zehnerkomplement

Quellen

[1] EDV-Grundwissen, M. Precht, N. Meier, J. Kleinlein, Addison-Wesley-Longman, 1998, ISBN 3-8273-1422-2

[2] http://ascii-table.com/ (Juli 2010)

[3] NZZ-Folio, Neue Zürcher Zeitung, Zürich, Februar 2002, ISSN 1420-5262