TECHNISCHE UNIVERSITÄT DRESDEN

VLSI-EDA

Fakultät Informatik, Institut für Technische Informatik, Professur VLSI-Entwurfssysteme, Diagnostik und Architektur

Rechnerstrukturen und -organisation

Informations-verarbeitung

Rainer G. Spallek

TU Dresden, 11.11.2020

Rainer.Spallek@tu-dresden.de http://vlsi-eda.inf.tu-dresden.de

Gliederung

1 Zahlensysteme (Stellenwertsysteme)

Polyadische Zahlensysteme

Ziffernfolge (Vor- und Nachkommastellen):

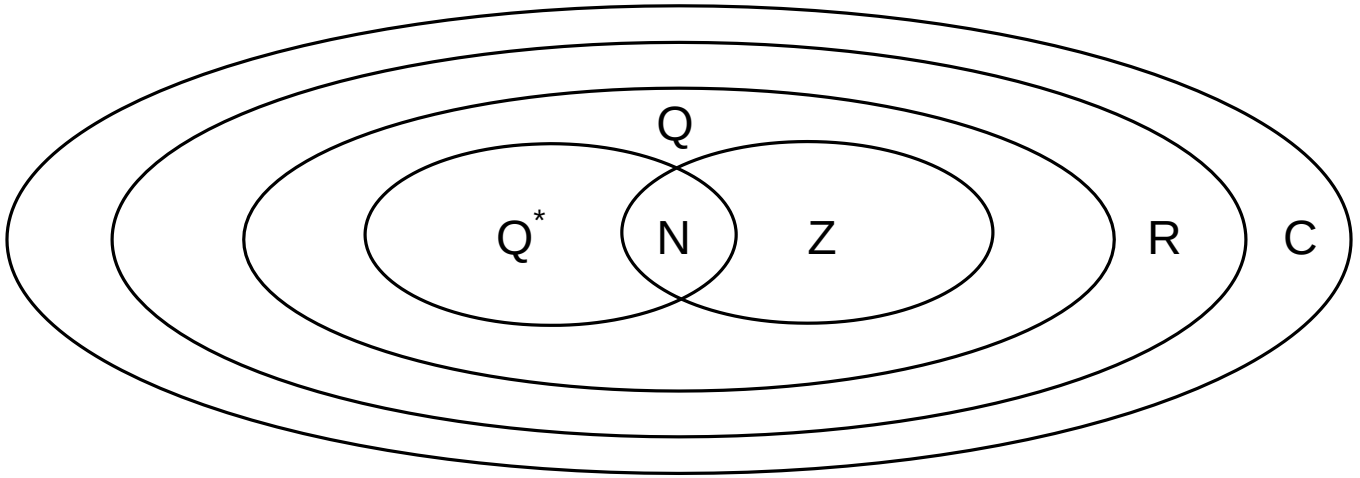
$$Z = \sum_{k=-m}^{n} a_k B^k \quad \text{mit} \quad B \geq 0,\ 0 \leq a_k < B,\ a_k \in N$$

Stellenwertsysteme sind gekennzeichnet durch:

- Darstellung einer Zahl erfolgt durch eine Ziffernfolge (Aneinanderreihung).
- Der Wert einer Ziffer hängt von ihrer Stellung innerhalb der Ziffernfolge ab.
- Der Ziffernwert wird zu einer einheitlichen Basis *B* gebildet.
- Die Basis ist die kleinste, nicht mehr durch eine Ziffer darstellbare Zahl.
- Der Wert der Zahl ergibt sich durch Aufsummierung ihrer Ziffernwerte.
- Dezimalsystem (B=10), Dualsystem (B=2), Hexadezimalsystem (B=16)

Zahlenbereiche

Menge von Zahlen, in der eine Ordnung erklärt ist und gewisse mathematische Verknüpfungen, Operationen uneingeschränkt ausführbar sind (keine natürliche Ordnung für komplexe Zahlen).



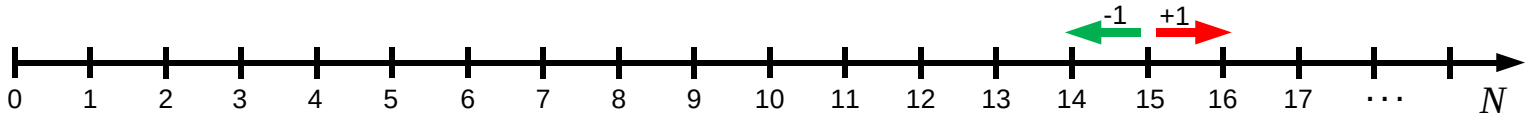
$N \subset Q^* \subset Q \subset R \subset C$ bzw. $N \subset Z \subset Q \subset R \subset C$

Zahlenbereiche

- Die Menge der natürlichen Zahlen N dient dem Abzählen.
- Die Menge der ganzen Zahlen Z entsteht aus N, indem man negative Zahlen als Inverse bzgl. der Addition konstruiert.
- Die Menge der nichtnegativen Brüche Q^* entsteht aus N, indem man Bruchzahlen als Inverse bzgl. der Multiplikation konstruiert.
- Die Menge der Brüche oder rationalen Zahlen Q entsteht aus Q^* durch Hinzunahme der Inversen bzgl. der Addition oder aus Z durch Hinzunahme der Inversen bzgl. der Multiplikation.
- Die Menge der reellen Zahlen R entsteht aus Q durch topologische Vervollständigung (z.B. Werte für e, π).
- Die Menge der komplexen Zahlen C entsteht aus Paaren reeller Zahlen (a,b) die in der Schreibweise $a + b \cdot i$ mit $i^2 = -1$ den üblichen Rechengesetzen genügen.

	Zahlenbereich	Definition	uneing. Operation
N	natürliche Zahlen	$N = \{0,1,2,...\}$	$+,*,<$
Z	ganze Zahlen	$Z = \{n-m,\ \ m,n \in N\}$	$+,-,*,<$
Q^*	gebrochene Zahlen	$Q^* = \{m/n,\ m,n \in N, n \neq 0\}$	$+,-,*,<$
Q	rationale Zahlen	$Q = \{a/b,\ a,b \in Z,\ b \neq 0\}$	$+,-,*,/,<$
R	reelle Zahlen	$a_0, a_1, a_2 ... \infty - Dezimalbruch$	$+,-,*,/,<,\lim$
C	komplexe Zahlen	$C = [a;b],\ a,b \in R\ (Paare)$	$+,-,*,/,\lim$

Zahlenstrahl natürlicher Zahlen (Ordnung: Nachfolger der Zahl n ist $n+1$).



Ganze Zahlen im Stellenwertsystem

Die Zifferndarstellung einer n-stelligen ganzen Zahl Z_B lautet im Stellenwertsystem der Basis *B*:

$$Z_B = \pm(z_{n-1}...z_1 z_0)_B.$$

Der Wert von Z_B bestimmt sich durch:

$$Z_B = \pm\sum_{i=0}^{n-1} z_i \cdot B^i = \pm(z_{n-1} \cdot B^{n-1} + \cdots + z_1 B^1 + z_0 B^0)$$

Beispiele:

$$Z_2 = 1010011010_2 = \sum_{i=0}^{9} z_i \cdot 2^i = 1\cdot 2^1 + 1\cdot 2^3 + 1\cdot 2^4 + 1\cdot 2^7 + 1\cdot 2^9 = 666_{10}$$

$$Z_8 = -4321_8 = \sum_{i=0}^{3} z_i \cdot 8^i = -(1\cdot 8^0 + 2\cdot 8^1 + 3\cdot 8^2 + 4\cdot 8^3) = -2257_{10}$$

Umrechnung ganzer Zahlen zwischen Stellenwertsystemen

$Z_D =$	$Z_B = \pm(z_{n-1} \cdot B^{n-1} + z_{n-2} \cdot B^{n-2} + \cdots + z_1 \cdot B^1 + z_0 \cdot B^0)$	Div.-Rest
$Q_0 =$	$Z_D = \pm(((\ldots(z_{n-1} \cdot B + z_{n-2}) \cdot B \ldots) \cdot B + z_1) \cdot B + z_0)$	
$Q_1 =$	$\frac{Q_0 - z_0}{B} = \pm((\ldots(z_{n-1} \cdot B + z_{n-2}) \cdot B \ldots) \cdot B + z_1)$	z_0
$Q_2 =$	$\frac{Q_1 - z_1}{B} = \pm(\ldots(z_{n-1} \cdot B + z_{n-2}) \cdot B \ldots)$	z_1
...		...
$Q_{n-1} =$	$\frac{Q_{n-2} - z_{n-2}}{B} = \pm z_{n-1}$	z_{n-2}
$Q_n =$	$\frac{Q_{n-1} - z_{n-1}}{B} = 0$	z_{n-1}

$$Z_B = \pm \sum_{i=0}^{n-1} z_i \cdot B^i = \pm(z_{n-1} \ldots z_1 z_0)_B$$

Fortlaufende ganzzahlige Division durch B bis $Q_n = 0 \longrightarrow$ Divisionsreste z_μ

Z_D - Ist-System, Z_B - Ziel-System, Q_V - ganzzahlige Quotienten

B-Komplement ganzer Zahlen im Stellenwertsystem

Das B-Komplement ${}^{(B)}Z_B$ einer Zahl Z_B im Stellenwertsystem der Basis B ist definiert durch (n Anzahl der Ziffern):

$$Z_B + {}^{(B)}Z_B = B^n \quad \text{mit} \quad Z_B = \sum_{i=0}^{n-1} z_i B^i \quad \text{und} \quad B^n = 1 + \sum_{i=0}^{n-1} (B-1) B^i$$

Der Wert dieses Komplements berechnet sich zu:

$${}^{(B)}Z_B = B^n - Z_B = 1 + \sum_{i=0}^{n-1} (B-1) B^i - \sum_{i=0}^{n-1} z_i B^i = 1 + \sum_{i=0}^{n-1} (B-1-z_i) B^i$$

Die beim B-Komplement auftretende Summe heißt $(B-1)$ -Komplement:

${}^{(B-1)}Z_B = \sum_{i=0}^{n-1} (B-1-z_i) B^i$ und das B-Komplement damit: ${}^{(B)}Z_B = {}^{(B-1)}Z_B + 1$

Beispiele zum B-Komplement:

Zehn-Komplement von $Z_{10} = 51_{10} \quad (n = 2)$:

$$^{(10)}Z_{10} = 10^2 - 51 = 100 - 51 = 49$$

Neun-Komplement von $Z_{10} = 51_{10} \quad (n = 2)$:

$$^{(9)}Z_{10} = 10^2 - 1 - 51 = 99 - 51 = 48$$

Zwei-Komplement von $Z_2 = 0101_2 \quad (n = 4)$:

$$^{(2)}Z_2 = 10000 - 0101 = 1011$$

(dezimal: $2^4 - 5 = 11$)

Eins-Komplement von $Z_2 = 0101_2 \quad (n = 4)$:

$$^{(1)}Z_2 = 10000 - 0001 - 0101 = 1111 - 0101 = 1010$$

(dezimal: $2^4 - 5 - 1 = 10$)

Echt gebrochene Zahlen im Stellenwertsystem

Die Zifferndarstellung einer gebrochenen Zahl R_B , die kleiner als 1 ist (Nachkommazahl), lautet im Stellenwert der Basis B :

$$R_B = \pm(0, z_{-1}, z_{-2}, \ldots, z_{-m})_B$$

Der Wert von R_B bestimmt sich durch:

$$R_B = \pm\sum_{i=1}^{m} z_{-i} \cdot B^{-i} = \pm(z_{-1} \cdot B^{-1} + z_{-2} \cdot B^{-2} + \ldots + z_{-m} \cdot B^{-m})$$

Beispiele zu gebrochenen Zahlen:

$$R_2 = 0{,}110_2 = \sum_{i=1}^{3} z_{-i} \cdot 2^{-i} = \pm(1 \cdot 2^{-1} + 1 \cdot 2^{-2}) = 0{,}75_{10}$$

$$R_{10} = -0{,}5362_{10} = -\sum_{i=1}^{4} z_{-i} \cdot 10^{-i} = -(5 \cdot 10^{-1} + 3 \cdot 10^{-2} + 6 \cdot 10^{-3} + 2 \cdot 10^{-4})$$

Umrechnung echt gebrochener Zahlen zwischen Stellenwertsystemen

$R_D =$	$R_B = \pm(z_{-1} \cdot B^{-1} + \ldots + z_{-m+1} \cdot B^{-m+1} + z_{-m} \cdot B^{-m})$	ganzz.
$P_0 =$	$R_D = \pm(((\ldots(z_{-m} \cdot B^{-1} + z_{-m+1}) \cdot B^{-1} \ldots) \cdot B^{-1} + z_{-1}) \cdot B^{-1})$	
$P_1 =$	$P_0 \cdot B - z_1 = \pm((\ldots(z_{-m} \cdot B^{-1} + z_{-m+1}) \cdot B^{-1} \ldots) \cdot B^{-1})$	z_{-1}
...		...
$P_{m-1} = P_{m-2} \cdot B - z_{m-1} = \pm z_{-m} \cdot B^{-1}$		z_{-m+1}
$P_m =$	$P_{m-1} \cdot B - z_m = 0$	z_{-m}

$$R_B = \pm \sum_{i=0}^{m} z_{-1} B^{-i} = \pm(0, z_{-1} z_{-2} \ldots z_{-m})_B$$

Fortlaufende Multiplikation mit B bis $P_m = 0 \longrightarrow$ ganzzahliger Anteil z_v

R_D - Ist-System, R_B - Ziel-System, P_V - echt gebrochene Produkte

Gebrochene Zahlen im Stellenwertsystem

Die Zifferndarstellung einer allgemeinen gebrochenen Zahl G_B lautet im Stellenwertsystem der Basis B (Festkomma-, Gleitkommadarstellung):

$$G_B = \pm(0, z_{-1}z_{-2} \ldots z_{-m})_B \cdot B^{(\pm e_k \ldots e_1 e_0)_B}, \quad M_B = \sum_{i=1}^{m} z_{-1}B^{-i} \quad , \quad E_B = \sum_{j=0}^{k-1} e_j B^j$$

Der Wert G_B bestimmt sich durch:

$$G_B = \pm\left(\sum_{i=1}^{m} z_{-i}B^{-i}\right) \cdot B^{\left(\pm\sum_{j=0}^{k-1} z_j B^j\right)} = \pm M_B \cdot B^{(\pm E_B)}$$

M_B - ist eine echt gebrochene Zahl (<1) und wird als Mantisse bezeichnet.

E_B - ist eine ganze Zahl und wird als Exponent bezeichnet.

Der Exponent gibt an, um wie viele Stellen das Komma nach rechts (positiv) bzw. nach links (negativ) verschoben werden muss.

Umrechnung gebrochener Zahlen zwischen Stellenwertsystemen

Zerlegung der gebrochenen Zahl in ganzzahligen und echt gebrochenen Anteil. Getrennte Umrechnung beider Teile und anschließende Addition.

$$G_B = \pm(z_{n-1} \ldots z_1 z_0, z_{-1} z_{n-2} \ldots z_{-m})_B = \pm((z_{n-1} \ldots z_1 z_0)_B + (0, z_{-1} z_{-2} \ldots z_{-m})_B)$$

Der Wert von G_B bestimmt sich durch:

$$G_B = \pm \sum_{i=-m}^{n-1} z_i \cdot B^i = \sum_{i=-m}^{-1} z_i \cdot B^i + \sum_{i=0}^{n-1} z_i \cdot B^i$$

$$G_B = \pm(((\ldots(z_{n-1} \cdot B + z_{n-2})B \ldots)B + z_1)B + z_0 \quad \leftarrow \text{ganzzahlig}$$

$$+ ((\ldots(z_{-m} \cdot B^{-1} + z_{-m+1}) \cdot B^{-1} \ldots) \cdot B^{-1} + z_{-1}) \cdot B^{-1}) \quad \leftarrow \text{ganzzahlig}$$

Stellenzahl bei Konvertierung zwischen Stellenwertsystemen bei gleicher Genauigkeit

Konvertierung einer Zahl aus dem Stellenwertsystem A in das Stellenwertsystem X bei gleichbleibender Genauigkeit:
(Alle Ziffern mit $B_A - 1$ bzw. $B_X - 1$ belegt, größter möglicher Wert)

$$Z_A = Z_X$$
$$B_A^{n_A} - 1 = B_X^{n_X} - 1$$
$$n_A \cdot \log B_A = n_X \cdot \log B_X$$
$$\frac{n_X}{n_A} = \frac{\log B_A}{\log B_X}$$

Z - Zahlenwert
n - Stellenzahl
B - Basis

	$\frac{n_X}{n_A}$	B_X 2	8	10	16
B_A	2	1	0,333	0,301	0,25
	8	3	1	0,903	0,75
	10	3,322	1,107	1	0,830
	16	4	1,333	1,204	1

2 Zahlendarstellung im Computer, Datenformate

Die Zahlendarstellung im Computer (Datenformat) erfolgt in der Regel binär codiert in Formaten fester Länge *n* (Blockcode).

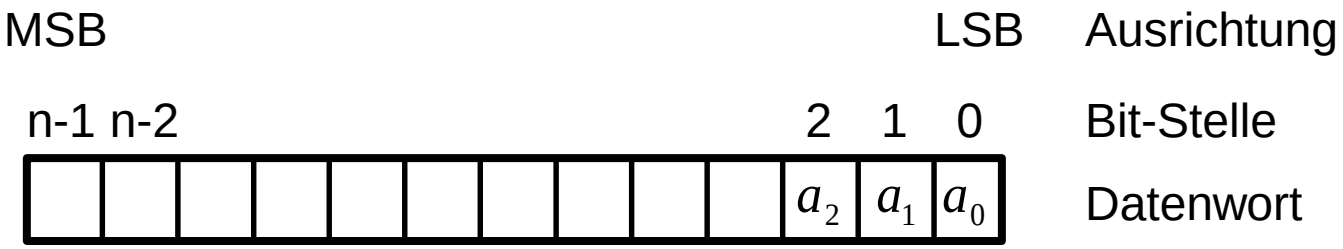
Es wird das Dualzahlen-Stellenwertsystem (*B*=2) verwendet.

Folgende Darstellungsvarianten werden allgemein unterschieden:

- Natürliche Zahlen (vorzeichenlose Dualzahl)
- Ganze Zahlen (vorzeichenbehaftete Dualzahl)
- Festkommazahlen (Festpunktzahlen)
- Gleitkommazahlen (Gleitpunktzahlen)

Durch die Darstellungsvariante selbst und die begrenzte Stellenzahl (Datenformate fester Länge *n*) sind die Gültigkeitsbereiche des Assoziativgesetztes und des Distributivgesetzes zu beachten.

n-bit Datenformat



Übliche Datenformate: 8-bit, 16-bit, 32-bit, 64-bit,…, $8*2^n$ bit für n = 0, 1, 2, …

- Die Datenwörter werden binär codiert und abgespeichert.
- Je nach Datenwort können einzelne Bit-Stellen spezielle Funktionen haben: z.B. Zahlenwert, Vorzeichen, Exponent, Mantisse, …
- Jede Darstellungsvariante hat ein gesondertes Datenformat.
- Aus der binären Codierung selbst ist die Art des Datenwortes, die Darstellungsvariante nicht erkennbar. Sie ist nur aus dem Programmkontext ermittelbar.

3 Natürliche Zahlen (Unsigned Integer), (Unsigned Binary Numbers)

Die Darstellung erfolgt als *n*-stellige Dualzahl (n-bit Codewort).
Eine Darstellung von negativen Zahlen ist nicht möglich.

Wertebereich: $0 \leq x \leq 2^n - 1$ x – unsigned integer, n-bit Codewort

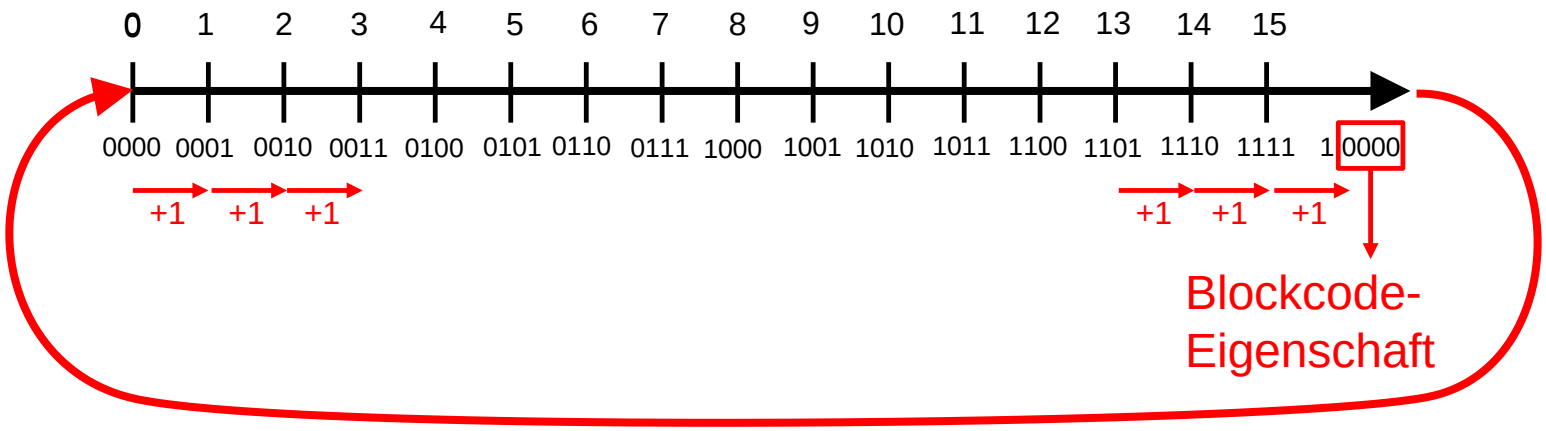
Beispiel für unsigned integer Darstellung (n=8):

dezimal		dual	dezimal		dual
+0	-	0000 0000	128	-	1000 0000
+3	-	0000 0011	131	-	1000 0011
+127	-	0111 1111	255	-	1111 1111

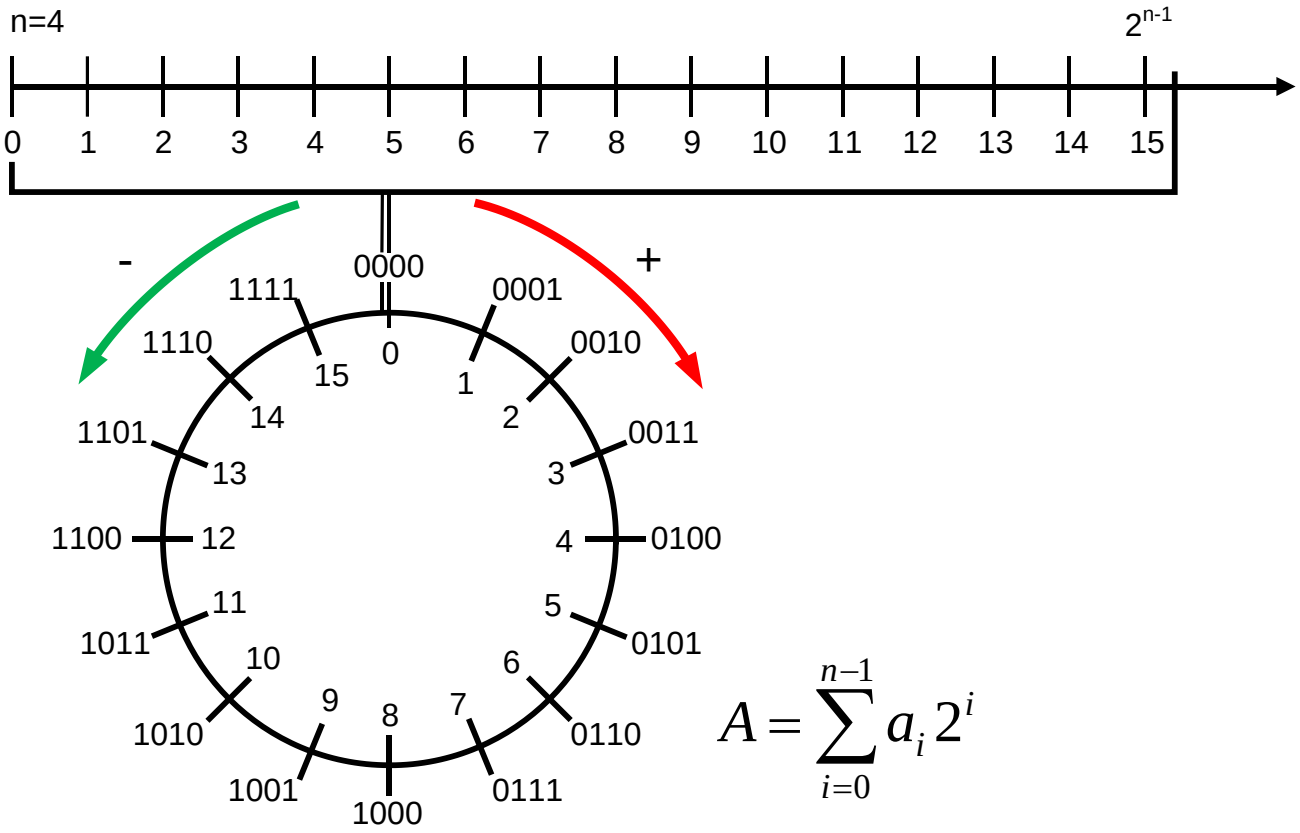
Anwendung : Adressmanipulation

Vom Zahlenstrahl zum Zahlenkreis

Beispiel: 4-bit unsigned integer



Zahlenkreis für 4-bit unsigned integer



$$A = \sum_{i=0}^{n-1} a_i 2^i$$

Addition

Beispiel:

	dezimal (n = 2)	dual (n = 8)
a	$5\ 9_{10}$	$0\ 0\ 1\ 1\ 1\ 0\ 1\ 1_2$
b	$+\ 9\ 1_{10}$	$+0\ 1\ 0\ 1\ 1\ 0\ 1\ 1_2$
s	$= 1^1\ 5^1\ 0_{10}$	$= 1^1\ 0^1 0^1 1^1 0^0 1^1 1^1\ 0_2$

Schreibweise: Summe$^{\text{Übertrag der vorherigen Summe}}$

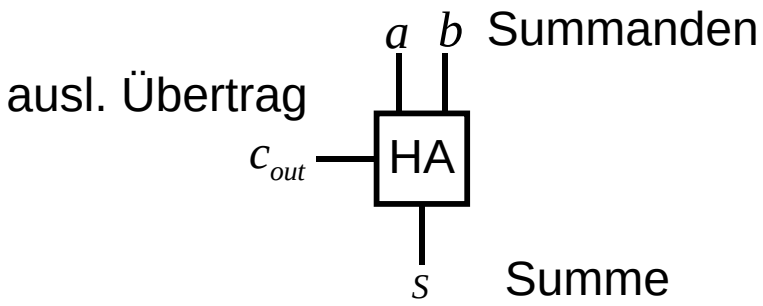
Sukzessive stellenweise Addition der Summanden mit dem LSB beginnend bis hin zum MSB $(a_0 + b_0, \ldots, a_{n-1} + b_{n-1})$ zur Summe $(s_0, \ldots, s_{n-1})$ mit Übertragstechnik. Die bei der stellenweise Addition auftretenden Überträge (carry) $(c_0, \ldots, c_{n-1})$ werden bei der Addition der jeweils nächstfolgenden Stelle mit dazu addiert. Der letzte Übertrag c_{n-1} heißt auslaufender Übertrag (carry out).

Hardware-Realisierung der Addition

1-bit Addition Halbaddierer

Halbaddierer (Half Adder (HA), ohne einlaufenden Übertrag)

Schaltsymbol



Modulo 2 Addition ohne Übertrag

Wertetabelle

a	b	S	c_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

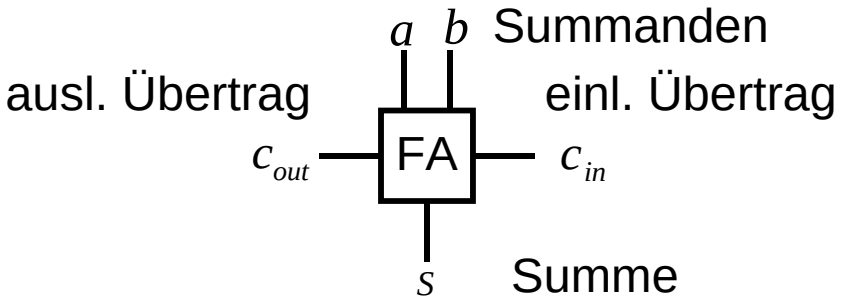
Schaltfunktion

$$s = a \wedge \bar{b} \vee \bar{a} \wedge b$$
$$= a \leftrightarrow b$$
$$c_{out} = a \wedge b$$

1-bit Addition Volladdierer

Volladdierer (Full Adder (FA), mit einlaufenden Übertrag)

Schaltsymbol



Modulo 2 Addition mit Übertrag

Wertetabelle

a	b	c_{in}	S	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Schaltfunktion

$$s = a \wedge \bar{b} \wedge \overline{c_{in}} \vee \bar{a} \wedge b \wedge \overline{c_{in}} \vee \bar{a} \wedge \bar{b} \wedge c_{in} \vee a \wedge b \wedge c_{in}$$
$$= a \leftrightarrow b \leftrightarrow c_{in}$$
$$c_{out} = a \wedge b \vee c_{in} \wedge b \vee a \wedge c_{in}$$

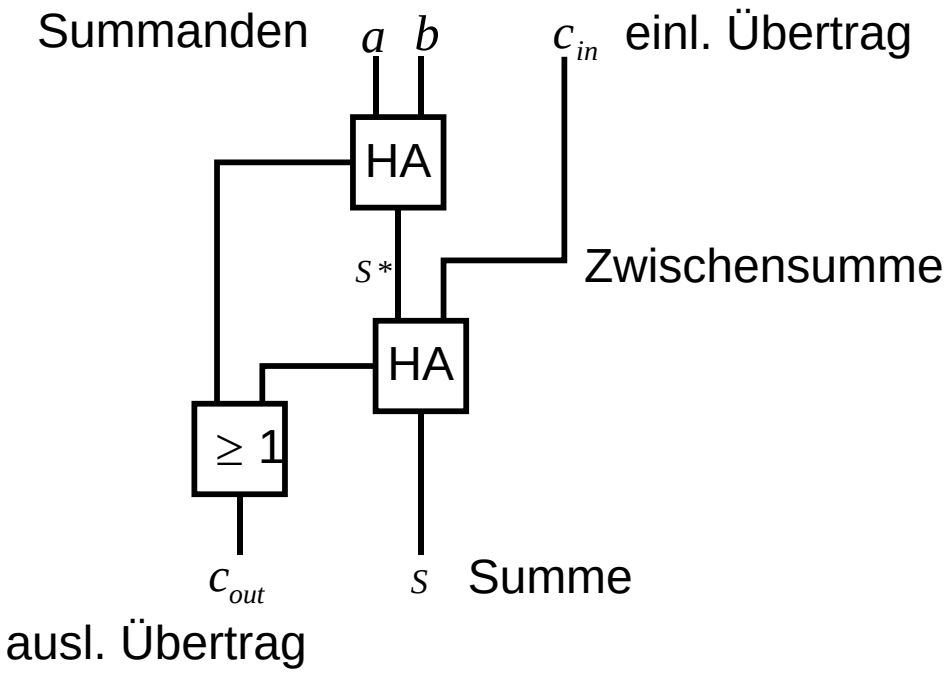
Übertrag beim Volladdierer

a	b	c_{out}		$c_{out} = g \vee p \wedge c_{in}$
0	0	0	kill	$k = \bar{a} \wedge \bar{b}$
0	1	c_{in}	propagate	$p = a \otimes b$
1	0	c_{in}	propagate	$= a \wedge \bar{b} \vee \bar{a} \wedge b$
1	1	1	generate	$g = a \wedge b$

Beim Volladdierer wird ein Übertrag $c_{out} = 1$ nur ausgegeben, wenn entweder einer generiert wird $g = 1$ oder der vom Eingang $c_{in} = 1$ propagiert wird $p = 1$. Bei $k = 1$ wird immer der Übertrag $c_{out} = 0$ ausgegeben.

Volladdierer realisiert durch Halbaddierer

Volladdierer, realisiert durch Halbaddierer



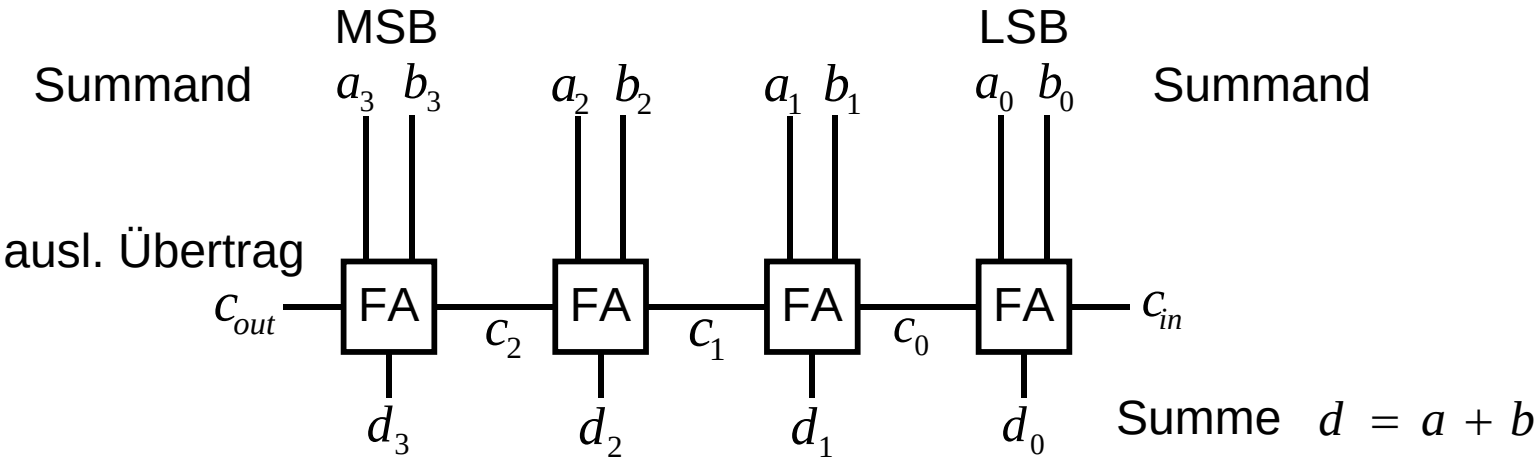
$$s* = \overline{a} \wedge b \vee a \wedge \overline{b} = a \leftrightarrow b$$

$$s = s* \wedge \overline{c_{in}} \vee \overline{s*} \wedge c_{in} = s* \leftrightarrow c_{in}$$

$$c_{out} = a \wedge b \vee s* \wedge c_{in}$$

n-bit Addition (Ripple-Carry-Adder, RCA)

Beispiel: 4-bit Addition



$$s_v = a_v \leftrightarrow b_v \leftrightarrow c_v \quad \text{für } v = 0,1,\ldots,n-1$$

$$c_v = a_v \wedge b_v \vee a_v \wedge c_v \vee c_v \wedge b_v$$

$$c_{in} = 0 \quad \text{kann bei } c_{-1} = 0 \text{ entfallen}$$

$$c_{out} = c_{n-1}$$

Subtraktion mit Borgetechnik

Beispiel:

		dezimal (n=2)	dual (n=8)
a		$9\ 1_{10}$	$0\ 1\ 0\ 1\ 1\ 0\ 1\ 1_2$
b	$-$	$5\ 9_{10}$	$-0\ 0\ 1\ 1\ 1\ 0\ 1\ 1_2$
s	$=$	$3^1\ 2_{10}$	$=0^0\,0^1\,1^0\,0^0\,0^0\,0^0\,0^0\,0_2$

Schreibweise: Differenz$^{\text{Übertrag (geborgt) der vorherigen Summe}}$

Sukzessive stellenweise Subtraktion mit dem LSB beginnend bis hin zum MSB mit Borgetechnik. Die bei der stellenweisen Subtraktion auftretenden (geborgten) Überträge (borrow) werden bei der Subtraktion der jeweils nächstfolgenden Stelle mit zum Subtrahenden addiert. Der letzte (geborgte) Übertrag c_{n-1} heißt auslaufender (geborgter) Übertrag (borrow out).

Subtraktion mit B-Komplement

Beispiel:

dezimal (n=2)

$$\begin{array}{rrr} a & & 9\ 1_{10} \\ b & (-\ & 5\ 9_{10}\ ) \\ {}^{(10)}b & +\ & 4\ 1_{10} \\ \hline s & =^{1}\ & 3^{0}\ 2_{10} \end{array}$$

dual (n=8)

$$\begin{array}{rrr} a & & 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1_{2} \\ b & (-\ & 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1_{2}\ ) \\ {}^{(2)}b & +\ & 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1_{2} \\ \hline & =^{1}\ & 0^{1}\ 0^{0}\ 1^{1}\ 0^{1}\ 0^{1}\ 0^{1}\ 0^{1}\ 0_{2} \end{array}$$

Schreibweise: Differenz$^{\text{Übertrag der vorherigen Summe}}$

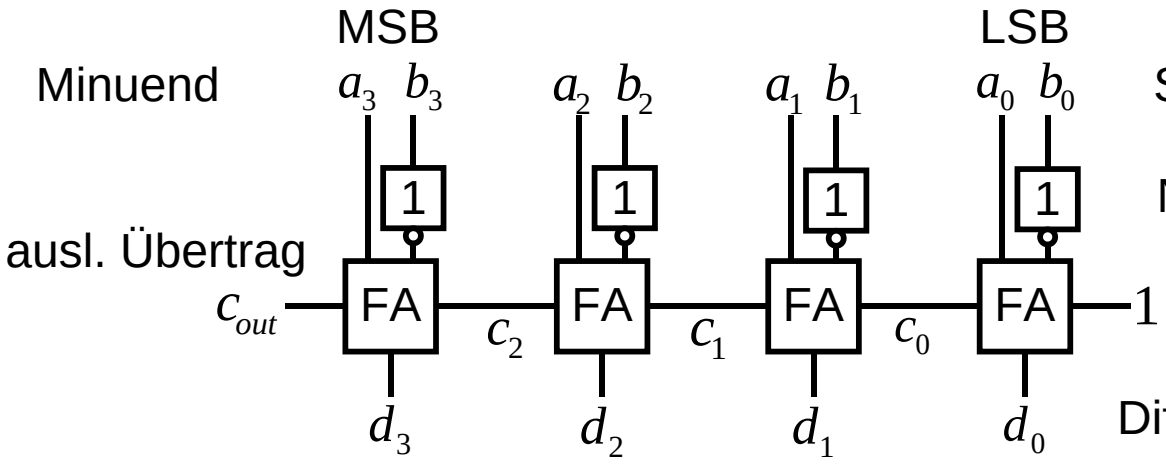
Realisierung der Subtraktion durch eine Addition mit dem B-Komplement des Subtrahenden bei fester Stellenzahl *n* (c_{n-1} wird dabei ignoriert).

$$d = a - b \quad \rightarrow \quad d = a +^{(2)} b$$

Die Subtraktion kann so auf eine Addition und eine 2-Komplement-bildung (bitweise Negation und Inkrement) zurückgeführt werden.

Hardware-Realisierung der Subtraktion

Beispiel: 4-bit Subtraktion (Ripple-Carry-Adder mit 2-Komplement)



$$s_v = a_v \leftrightarrow \overline{b_v} \leftrightarrow c_v \qquad \text{für } v = 0,1,\ldots,n-1$$

$$c_v = a_v \wedge \overline{b_v} \vee a_v \wedge c_v \vee c_v \wedge \overline{b_v}$$

$$c_{-1} = 1 \qquad \text{für 2-Komplement}$$

$$c_{out} = c_{n-1}$$

Multiplikation

Beispiel:

dezimal (n=2)

$$\begin{array}{rl} 1\ 5_{10} & \cdot\ 1\ 3_{10} \\ \hline & 4^1\ 5 \\ + & 1^0\ 5 \\ \hline = & 1^0\ 9^0\ 5_{10} \end{array}$$

dual (n=4)

$$\begin{array}{rl} 1\ 1\ 1\ 1_2 & 1\ 1\ 0\ 1_2 \\ \hline & 1\ 1\ 1\ 1 \\ + & 0\ 0\ 0\ 0 \\ + & 1\ 1\ 1\ 1 \\ + & 1\ 1\ 1\ 1 \\ \hline & 1\ 1^{10}0^{10}0^{10}0^{1}0\ 1\ 1_2 \end{array}$$

Sukzessive stellenweise Multiplikation des Multiplikator (0 oder 1) mit dem LSB beginnend bis hin zum MSB mit dem gesamten Multiplikanden und stellenrichtige Aufsummierung der Teilergebnisse (Linksverschiebung um n-1 Stellen): $(a \cdot b = a \cdot b_0 \cdot 2^0 + a \cdot b_1 \cdot 2^1 + ... + a \cdot b_{n-1} \cdot 2^{n-1})$.

Multiplikation Algorithmus, Stellenverschiebung

$$a \cdot b = p$$

$$b = b_0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + \ldots + b_{n-1} \cdot 2^{n-1}$$

$$p = a \cdot b_0 + a \cdot b_1 \cdot 2^1 + a \cdot b_2 \cdot 2^2 + \ldots + a \cdot b_{n-1} \cdot 2^{n-1}$$

$$b_v \in \{0, 1\}$$

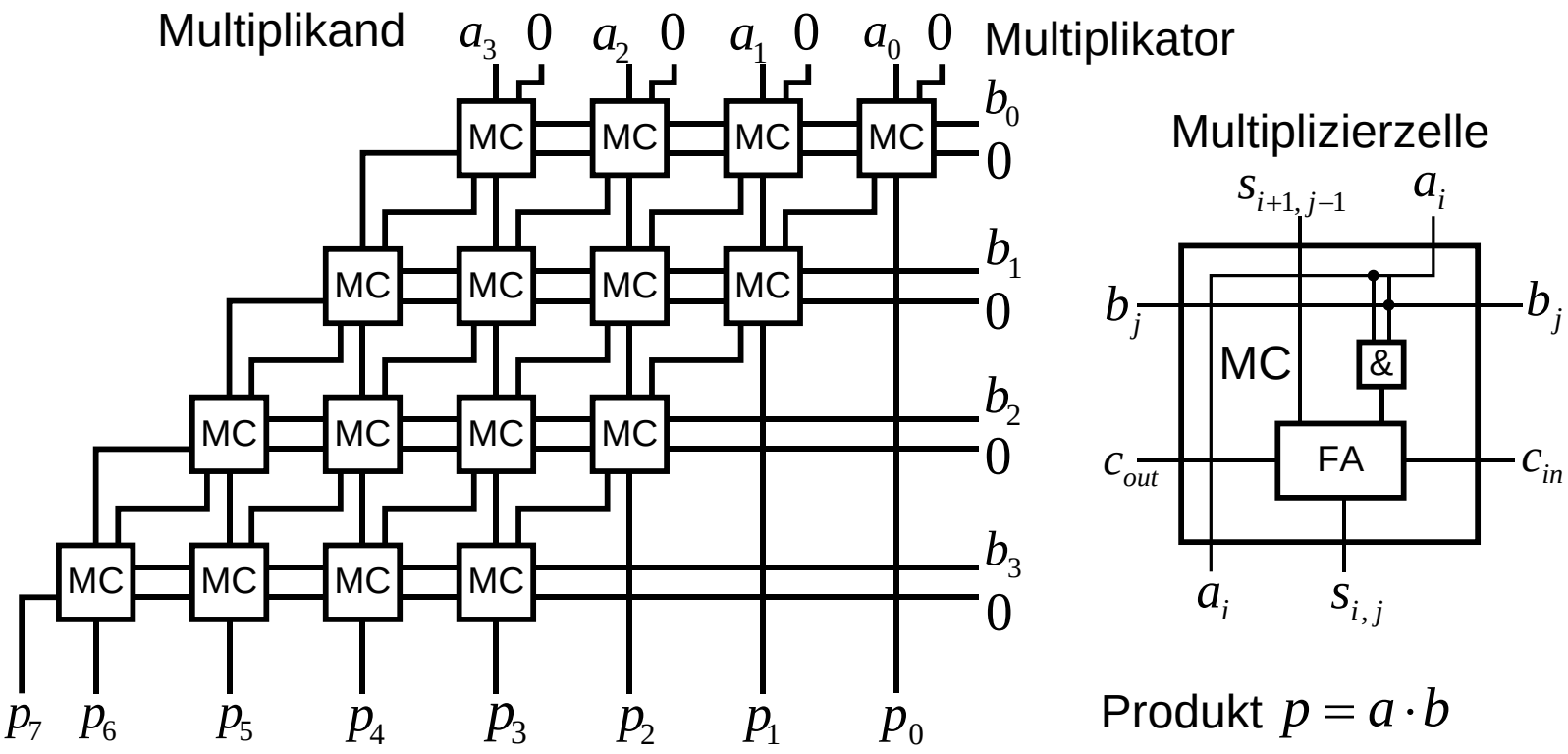
Die Multiplikation kann so auf eine fortlaufende Addition und mit einer entsprechenden Stellenverschiebung zurückgeführt werden.

Eine Multiplikation mit 2 bedeutet eine Verschiebung des Multiplikanden um eine Stelle nach links, wobei das LSB mit 0 aufgefüllt wird (analog für eine Multiplikation mit 2^k eine Verschiebung um *k* Stellen).

$$a \cdot 2^k \rightarrow (a_l = a_{l-k},\ \textit{für}\ \ l = n-1, \ldots, k)\ \ \textit{und}\ \ (a_l = 0,\ \textit{für}\ \ l = k-1, \ldots, 0)$$

Hardware-Realisierung der Multiplikation

Beispiel: 4-bit Multiplikation (Ripple-Carry-Addition)



Produkt $p = a \cdot b$

Ziffernitterative Division

Beispiel:

dezimal (n=2)

$$\begin{array}{rl} 59_{10} : 3_{10} = 19_{10} & \\ -\ 3 & \\ \hline 29 & \\ -\ 27 & \\ \hline 2 & \text{Rest} \end{array}$$

dual (n=6) ohne führende 0

$$\begin{array}{rll} 111011_2 : 11_2 = & & 10011_2 \\ -\ 11 & \rightarrow 11 \le 11 & \rightarrow 1 \uparrow \\ \hline 01011 & & \\ 11 & \rightarrow 11 > 01 & \rightarrow 0 \uparrow \\ 11 & \rightarrow 11 > 010 & \rightarrow 0 \uparrow \\ -\ 11 & \rightarrow 11 \le 0101 & \rightarrow 1 \uparrow \\ \hline 101 & & \\ -\ 11 & \rightarrow 11 \le 101 & \rightarrow 1 \uparrow \\ \hline 10 & \text{Rest} & \end{array}$$

Division Algorithmus

$$
\begin{aligned}
a:b &= q \\
q &= q_0 + q_1 \cdot 2^1 + q_2 \cdot 2^2 + ... + + q_{n-1} \cdot 2^{n-1} \\
a &= b \cdot q \\
a &= b \cdot q_0 + b \cdot q_1 \cdot 2^1 + b \cdot q_2 \cdot 2^2 + ... + b \cdot q_{n-1} \cdot 2^{n-1} \\
q_v &\in \{0, 1\}
\end{aligned}
$$

Bei der Division wird stellenweise mit MSB von q beginnend das Produkt $b \cdot [q_v] \cdot 2^v$ von a (durch Verschiebung) subtrahiert. Ist die Differenz positiv so ist $q_v = 1$, sonst 0. Mit der verbleibenden Differenz von a wird analog weiter verfahren bis alle Stellen abgearbeitet sind. Als letzte Differenz bleibt der Divisionsrest übrig.

Die Division kann so auf eine Subtraktion mit entsprechender Stellenverschiebung und einen Vergleich zurückgeführt werden.

Division - Stellenverschiebung

Eine Division durch 2 bedeutet eine Verschiebung des Dividenden um eine Stelle nach rechts, wobei das MSB mit 0 aufgefüllt (analog für eine Division durch 2^k eine Verschiebung um k Stellen).

$$\frac{a}{2^k} \rightarrow (a_{l-k} = a_l, \text{ für } l = k, \ldots, n-1) \text{ und } (a_l = 0, \; für\, l = n-1, \ldots, n-k-1)$$

Dieses Divisionsverfahren ist in Hardware aufwendiger zu realisieren als die Multiplikation und benötigt daher allgemein auch deutlich mehr Rechenzeit.

Eine Alternative zu diesem Divisionsverfahren stellen verschieden iterative Verfahren dar, die die Division auf Multiplikation und Subtraktion zurückführen.

Iterative Division mit dem Newton-Verfahren

Ansatz über die Kehrwertberechnung

q	$=$	$\frac{a}{b} = a \cdot \frac{1}{b}$	zu bestimmender Quotient
$f(x)$	$=$	$\frac{1}{x} - b$	Ansatzfunktion mit Nullstelle bei $x = \frac{1}{b}$
x_{i+1}	$=$	$x_i - \frac{f(x_i)}{f'(x_i)}$	Iterationsvorschrift zur Lösung von $f(x) = 0$
$f'(x_i)$	$=$	$\frac{-1}{x^2}$	Ableitung von $f(x)$
x_{i+1}	$=$	$x_i \cdot (2 - x_i \cdot b)$	Iterationsvorschrift, x_{i+1} - verbesserte Näherung
$\frac{a}{b}$	$=$	$a \cdot x_n$	bei Konvergenz $x_{i+1} \xrightarrow[i \to \infty]{} \frac{1}{b}$
x_0			Startwert

(Alternatives Verfahren: Iterative Division mit dem Goldschmidt-Algorithmus)

Beispiel Newton-Iteration

Beispiel: $x_q = \frac{a}{b} = \frac{15}{7} = 15 \cdot \frac{1}{7} = 2.142857142857...$

Startwert: $x_0 = 0.1$, Iteration zur Bestimmung von $x = \frac{1}{7}$

x_{i+1}	=	$x_i \cdot (2 - x_i \cdot b)$		Genauigkeit
x_1	=	0.1 · (2 - 0.1 · 7)	=0.13	10^{-1}
x_2	=	0.13 · (2 - 0.13 · 7)	=0.1417	10^{-2}
x_3	=	0.1417 · (2 - 0.1417 · 7)	=0.14284777	10^{-4}
x_4	=	...	=0.142857142242	10^{-8}
x_5	=	...	=0.142857142857	10^{-16}

Lösung: $q = 15 \cdot x_5 = 2.142857142857...$

Das Newton-Verfahren konvergiert quadratisch gegen die Lösung.

→ Verdopplung der Genauigkeit (Stellenzahl) in jedem Iterationsschritt.

Konvertierung Dual - Dezimal

Beispiel: Konvertierung einer Dualzahl $X_2 = 1\ 0011\ 1010_2$ in die wertgleiche Dezimalzahl X_{10}

$$X_2 = 1\ 0011\ 1010_2 = \sum_{i=0}^{n-1} z_i \cdot 2^i$$

$X_2 =$	1	0	0	1	1	0	0	1	0
$X_{10} =$	$1\cdot 2^8 +$	$0\cdot 2^7 +$	$0\cdot 2^6 +$	$1\cdot 2^5 +$	$1\cdot 2^4 +$	$0\cdot 2^3 +$	$0\cdot 2^2 +$	$1\cdot 2^1 +$	$0\cdot 2^0$

$$X_{10} = X_2 = 1\ 0011\ 1010_2 = 314_{10}$$

Konvertierung Dezimal - Dual

Beispiel: Konvertierung einer Dezimalzahl $X_{10} = 314$ in die wertgleiche Dualzahl X_2

Sukzessive Division der Dezimalzahl durch 2 und notieren des Restes.
Mit dem Ergebnis der Division ist solange in gleicher Weise weiter zu verfahren, bis das Resultat 0 erreicht ist.
Der jeweilige Rest, der nur die Werte 0 und 1 annehmen kann, bildet mit dem letzten Rest der Division als MSB beginnend, die äquivalente Dualzahl.

314	: 2 =	157	Rest 0
157	: 2 =	78	Rest 1
78	: 2 =	39	Rest 0
39	: 2 =	19	Rest 1
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

314_{10} = $1\,0011\,1010_2$

4 Ganze Zahlen (Signed Integer), (Signed Binary Numbers)

Die Darstellung erfolgt als n-stellige Dualzahl, n-bit Codewort, wobei das Vorzeichen durch verschiedene Verfahren realisiert wird:

- Vorzeichen-Wert-Darstellung
- Basiswert-Darstellung
- 1-Komplement-Darstellung
- 2-Komplement-Darstellung.

Die Wertebereiche der Darstellungen unterscheiden sich je nach der Realisierung des Vorzeichens.

Anwendung: Integer-Arithmetik

Vorzeichen-Wert-Darstellung

Das höchstwertige Bit (MSB) der Dualzahl wird für die Darstellung des Vorzeichens genutzt: (0-positiv, 1-negativ), die restlichen Stellen als n-1 Bit lange vorzeichenlose ganze Zahl.

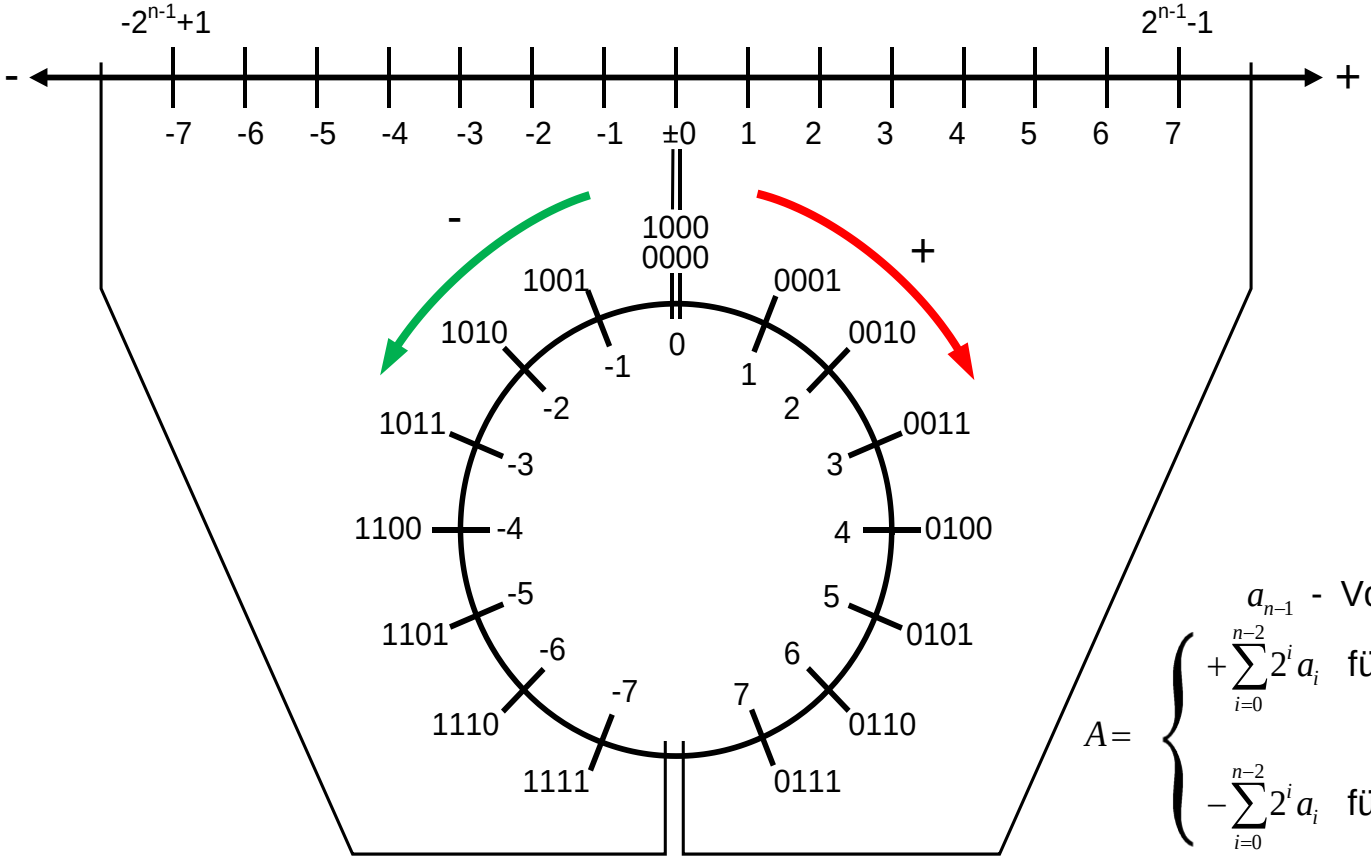
Wertebereich: $-2^{n-1}+1 \leq x \leq 2^{n-1}-1$ x – signed integer

- zwei verschiedene Darstellungen für die Zahl 0 : +0 und -0
- Wertebereich symmetrisch
- Vorzeichen im MSB (0 – positive Zahl, 1 – negative Zahl)

Beispiel für Vorzeichen-Wert-Darstellung (n=8):

dezimal		dual	dezimal		dual
+0	-	0000 0000	-0	-	1000 0000
+3	-	0000 0011	-3	-	1000 0011
+127	-	0111 1111	-127	-	1111 1111

Zahlenkreis für 4-bit signed integer in Vorzeichen-Wert-Darstellung



a_{n-1} - Vorzeichen

$$A = \begin{cases} +\sum_{i=0}^{n-2} 2^i a_i & \text{für } a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & \text{für } a_{n-1} = 1 \end{cases}$$

Basiswert-Darstellung (Biased)

Darstellung ganzer Zahlen x als positive Dualzahl d mit einer festen Basiswertverschiebung B.

$$x = d - B \quad \rightarrow \quad d = x + B \geq 0 \qquad x - \text{signed integer}$$

Wertebereich: $-B \leq x \leq 2^n - 1 - B$ (ca. symmetrisch für $B = 2^{n-1} - 1$)

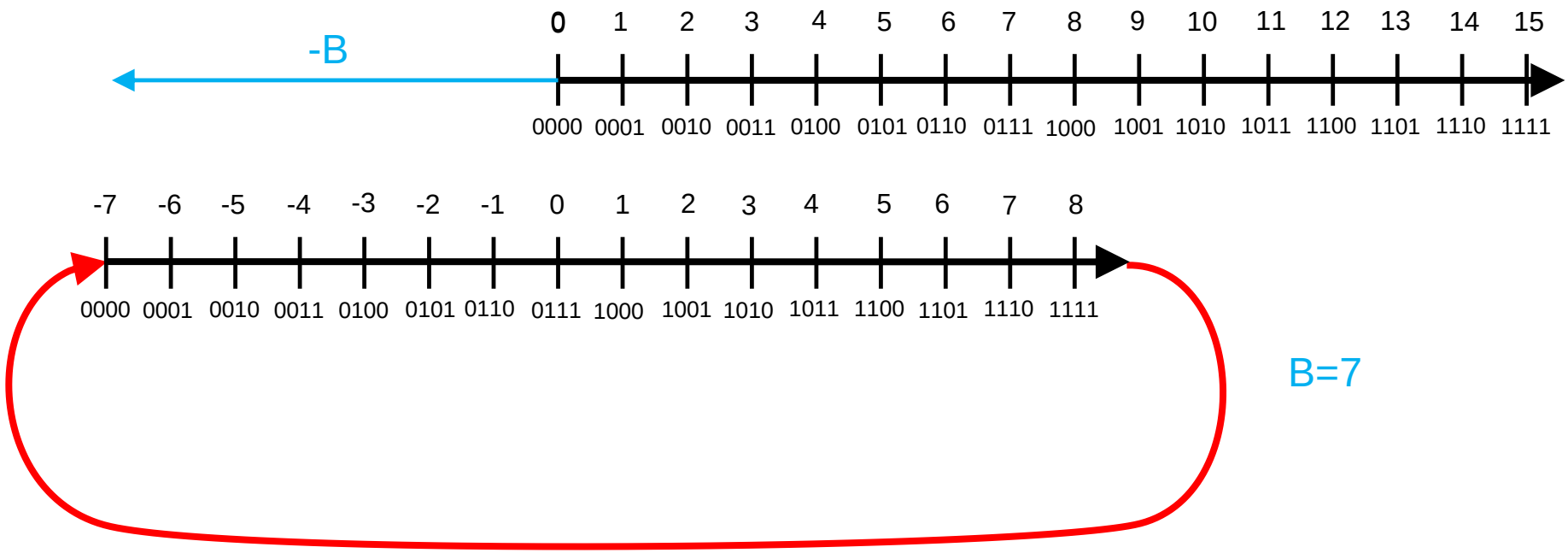
- eindeutige Darstellung der 0
- Wertebereich unsymmetrisch
- Vorzeichen für $B = 2^{n-1} - 1$ negativ bei MSB = 0, positiv bei MSB = 1

Beispiel für Basiswert-Darstellung (n=8, B = 127_{10} = 01111111_2):

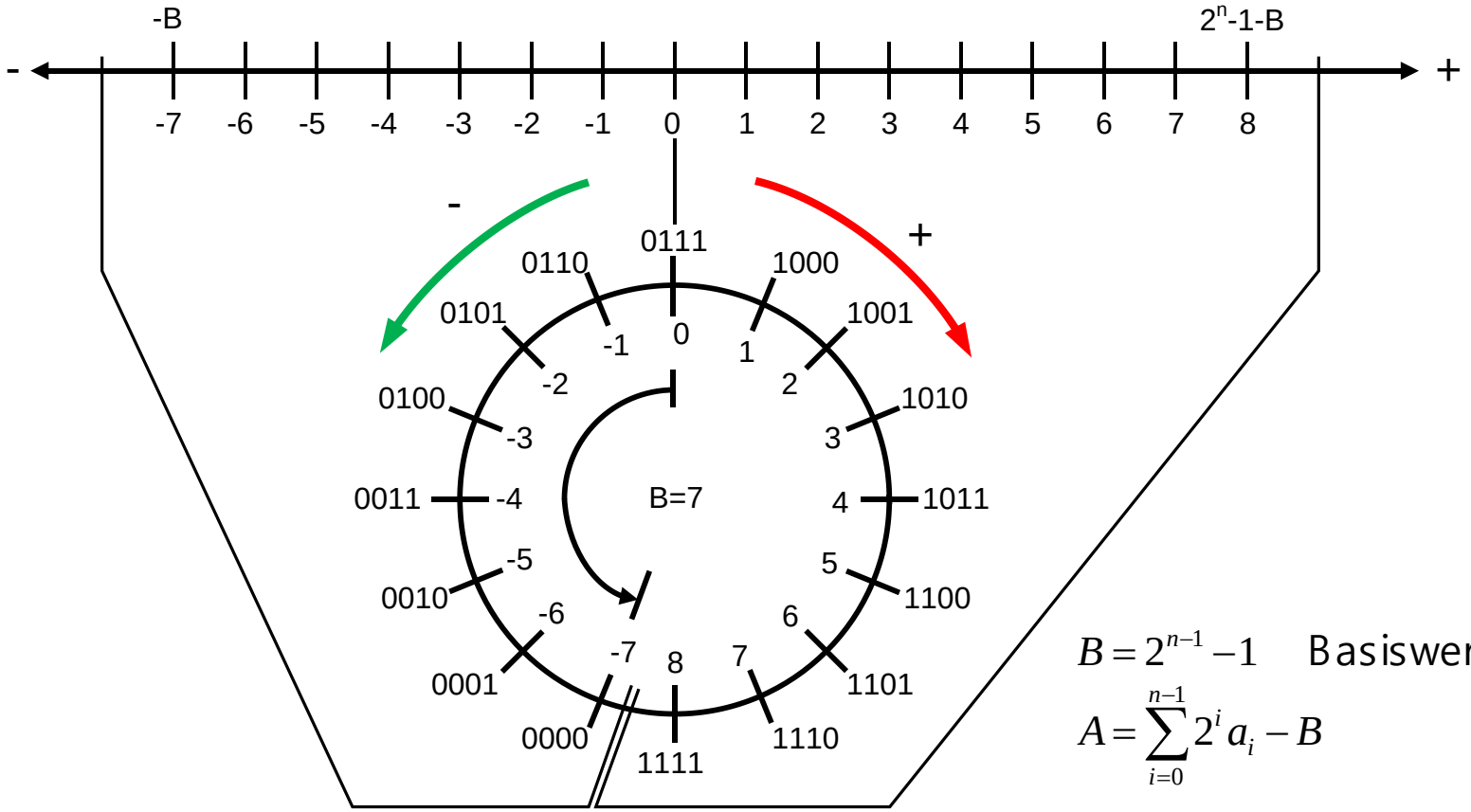
dezimal	dual	dezimal	dual
+0	0111 1111	-1	0111 1110
+3	1000 0010	-3	0111 1100
+128	1111 1111	-127	0000 0000

Zahlenstrahl mit Basiswert-Verschiebung

Beispiel: 4-bit signed integer



Zahlenkreis für 4-bit signed integer in Basiswert-Darstellung



$$B = 2^{n-1} - 1 \quad \text{Basiswert}$$

$$A = \sum_{i=0}^{n-1} 2^i a_i - B$$

1-Komplement-Darstellung

Darstellung negativer ganzer Zahlen durch das 1-Komplement $\left[-x\right]=^{(1)}x$.

$$^{(1)}x=(2^n-1)-x \quad \rightarrow \quad \left[-x\right]+x=^{(1)}x+x=(2^n-1)=0$$

Wertebereich: $2^{n-1}+1\leq x\leq 2^{n-1}-1$

- zwei unterschiedliche Darstellungen für die Zahl 0: +0 und -0
- Wertebereich symmetrisch
- Vorzeichen im MSB

Beispiel für 1-Komplement-Darstellung (n=8):

dezimal	dual	dezimal	dual
+0	0000 0000	-0	1111 1111
+3	0000 0011	-3	1111 1100
+127	0111 1111	-127	1000 0000

Bildungsvorschrift für das 1-Komplement (Stellenkomplement)

Bildung des 1-Komplement: $^{(1)}x = (2^n - 1) - x$

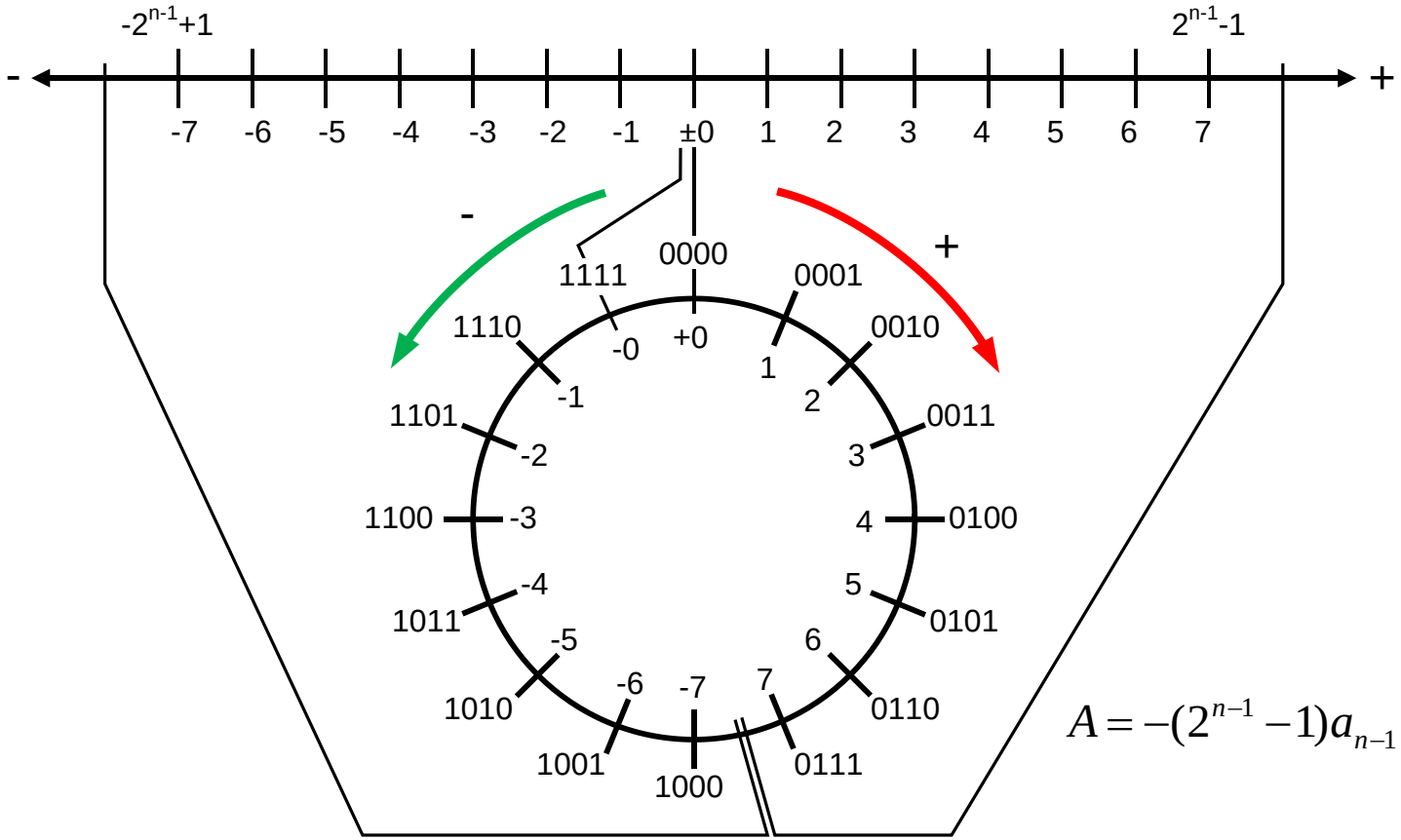
Bit-stellenweise Negation von x: $^{(1)}x \quad \rightarrow \quad (x_v \rightarrow \bar{x}_v \text{ für } v = 1 \ldots n-1)$

daraus folgt:

$$\begin{aligned} {}^{(1)}x + x &= x_{n-1}x_{n-2} \ldots x_0 + \bar{x}_{n-1}\bar{x}_{n-2} \ldots \bar{x}_0 \\ &= 1_{n-1}1_{n-2} \ldots 1_0 = 2^n - 1 \quad \text{mit} \quad x_v + \bar{x}_v = 1 \\ {}^{(1)}({}^{(1)}x) &= x \quad \rightarrow \quad -(-x) = x \end{aligned}$$

- Einfache Bildung des 1-Komplement durch bitstellenweise Negation.
- Hin- und Rücktransformation beim 1- Komplement sind identisch (positive Zahl ⇔ negative Zahl).
- Beim Rechnen mit dem 1-Komplement kann ein Fehler auftreten. Ist bei der Berechnung der Summe $s = a+b$ der auslaufende Übertrag 1, so ist die Summe wie folgt zu korrigieren: $s = (a+b) + 1$!

Zahlenkreis für 4-bit signed integer in 1-Komplement-Darstellung



$$A = -(2^{n-1}-1)a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

2-Komplement-Darstellung

Darstellung negativer ganzer Zahlen x durch das 2-Komplement ${}^{(2)}x$.

${}^{(2)}x = (2^n) - x \quad \rightarrow \quad {}^{(2)}x + x = 2^n = 0$

Wertebereich: $-2^{n-1} \leq x \leq 2^{n-1} - 1$

- Eindeutige Darstellung der Zahl 0
- Wertebereich unsymmetrisch
- Negative Zahlen sind durch eine 1 im MSB gekennzeichnet.

Beispiel für 2-Komplement-Darstellung (n=8):

dezimal		dual	dezimal		dual
+0	-	0000 0000	-1	-	1111 1111
+3	-	0000 0011	-3	-	1111 1101
+127	-	0111 1111	-128	-	1000 0000

Bildungsvorschrift für das 2-Komplement (echtes Komplement)

Stellenweise Negation von x (1-Komplement) und anschließendes Inkrementieren (1-Addition): ${}^{(2)}x = {}^{(1)}x + 1$

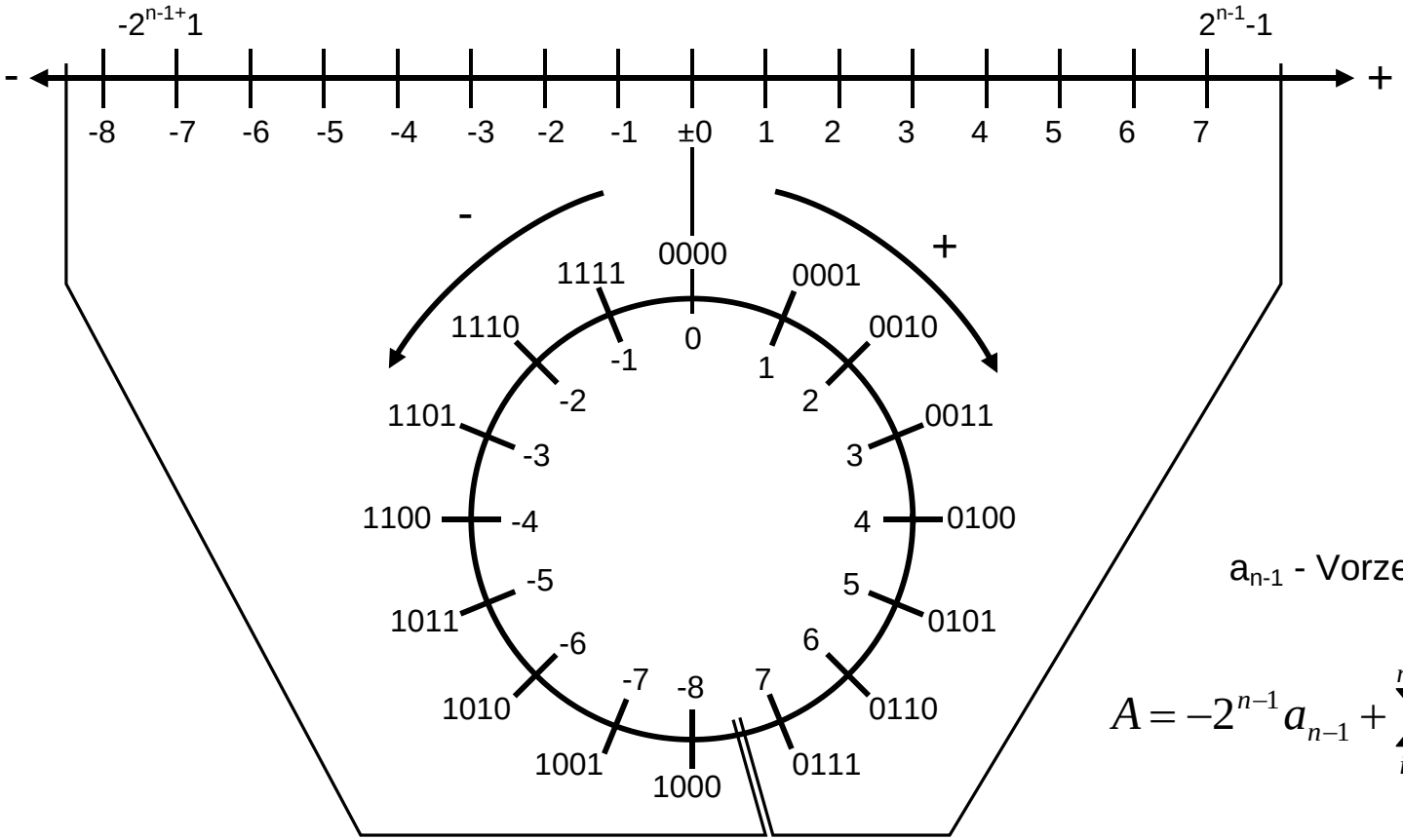
Daraus folgt:

$$\begin{aligned} {}^{(2)}x + x &= x_{n-1}x_{n-2}\ldots x_0 + \bar{x}_{n-1}\bar{x}_{n-2}\ldots\bar{x}_0 + 1 \\ &= 1_{n-1}1_{n-2}\ldots 1_0 + 1 = 2^n \\ x_v + \bar{x}_v &= 1 \\ {}^{(2)}({}^{(2)}x) = x &\rightarrow -(-x) = x \end{aligned}$$

Hin- und Rücktransformation (positive Zahl ⇔ negative Zahl) sind identisch.

Die 2-Komplementdarstellung für signed integer wird in der Computertechnik am häufigsten verwendet.

Zahlenkreis für 4-bit signed integer in 2-Komplement-Darstellung



a_{n-1} - Vorzeichen

$$A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

2-Komplement-Darstellung Addition/Subtraktion

- Die Darstellung negativer Dualzahlen erfolgt durch das 2-Komplement.
- Die Konvertierung positiver Dualzahlen in negative und umgekehrt erfolgt am einfachsten durch bitweise Negation der Dualzahl und anschließender Inkrementierung (1-Addtition).
- Negative Zahlen sind durch die 1 im MSB gekennzeichnet.
- Die Subtraktion entspricht einer Addition mit einer negativen Zahl.

$$a - b = a + (- b)$$

- Die Addition wird analog zur Addition vorzeichenloser ganzer Zahlen durchgeführt (bitweise modulo-2 Addition mit Übertrag).
- Ein Überlauf (Wertebereichsüberschreitung) liegt vor, wenn beim höchstwertigen Bit (MSB) der einlaufende und der auslaufende Übertrag unterschiedlich sind.

Beispiele 2-Komplement Addition

dezimal (n=2)

		1	6_{10}
+		3	2_{10}
=		4^0	8_{10}

		8	5_{10}
+		4	3_{10}
=	1^1	2^0	8_{10}

		6	5_{10}
+		-	3_{10}
=		6^0	2_{10}

dual (n=8)

		0	0	0	1	0	0	0	0_2
+		0	0	1	0	0	0	0	0_2
=	0^0	0^0	0^0	1^0	1^0	0^0	0^0	0^0	0_2

		0	1	0	1	0	1	0	1_2
+		0	0	1	0	1	0	1	1_2
=	0^0	1^1	0^1	0^1	0^1	0^1	0^1	0^1	0_2 **OV**

		0	1	0	0	0	0	0	1_2
+		1	1	1	1	1	1	0	1_2
=	1^1	0^1	0^0	1^0	1^0	1^0	1^0	1^1	0_2

OV – Überlauf (Wertebereichsüberschreitung, overflow)

Beispiele 2-Komplement Subtraktion

dezimal (n=2)

	-1	1	5_{10}
-		1	6_{10}
=	-1	3^1	1_{10}

	-1	1	5_{10}
-		1	3_{10}
=	-1^0	2^1	8_{10}

		1	6_{10}
-		3	2_{10}
=	-	1^1	6_{10}

dual (n=8)

		1	0	0	0	1	1	0	1_2	
+		1	1	1	1	0	0	0	0_2	
=	1^1	0^0	1^0	1^0	1^0	1^0	1^0	0^0	1_2	**OV**

		1	0	0	0	1	1	0	1_2	
+		1	1	1	1	0	0	1	1_2	
=	1^1	1^1	0^1	0^1	0^1	0^1	0^1	0^1	0_2	

		0	0	0	1	0	0	0	0_2	
+		1	1	1	0	0	0	0	0_2	
=	0^0	1^0	1^0	1^0	1^0	0^0	0^0	0^0	0_2	

OV – Überlauf (Wertebereichsüberschreitung, overflow)

2-Komplement-Darstellung Multiplikation/Division

Multiplikation und Division mit Konvertierung negativer Zahlen in die Vorzeichen-Wert-Darstellung

1. Überführung der 2-Komplement-Darstellung negativer Zahlen in die Vorzeichen-Wert-Darstellung.
2. Multiplikation bzw. Division der vorzeichenlosen Beträge analog zu vorzeichenlosen ganzen Zahlen (unsigned integer).
3. Gesonderte Bestimmung des Vorzeichens (gleiche Vorzeichen – positives Ergebnis, unterschiedliche Vorzeichen – negatives Ergebnis).
4. Überführung der Vorzeichen-Wert-Darstellung negativer Ergebnisse in die 2-Komplement-Darstellung.

Booth-Recording : Direkte Verarbeitung von Zweierkomplement-Darstellungen bei der Multiplikation.

5 Festkommazahlen (fixed-point numbers)

Darstellung von gebrochenen Zahlen

- Die Darstellung analog signed integer, ohne Komma.
- Das Komma steht implizit immer an der selben Stelle. Die Anzahl der Nachkommastellen *m* wird nicht mit in der Darstellung abgespeichert.
- Die Zahl der Vorkommastellen ergibt sich aus der Differenz von Gesamtstellenzahl *n* und Nachkommastellenzahl *m* (Vorzeichen im MSB).

***n*-Stellige Festkommazahl mit *m* Nachkommastellen:**

$$z = d, f \;\rightarrow\; d_{n-1} d_{n-2} \ldots d_{n-m} f_{m-1} f_{m-2} \ldots f_0 \qquad \left(\cdot 2^{-m}\right)$$

f fractional part

kleinste Zahl für $|z| : 2^{-m}$

größte Zahl für $|z| : (2^{n-1} - 1) \cdot 2^{-m}$

Anwendung : Real-Arithmetik

Konvertierung einer Festkommazahl in eine wertgleiche Dezimalzahl

Beispiel: Konvertierung einer Hexadezimalzahl $X_{16} = A45{,}8F$ (2 hexadezimale Nachkommastellen) in die wertgleiche Dezimalzahl X_{10}

$$X_{16} = A \quad 4 \quad 5, \quad 8 \quad F_{16}$$

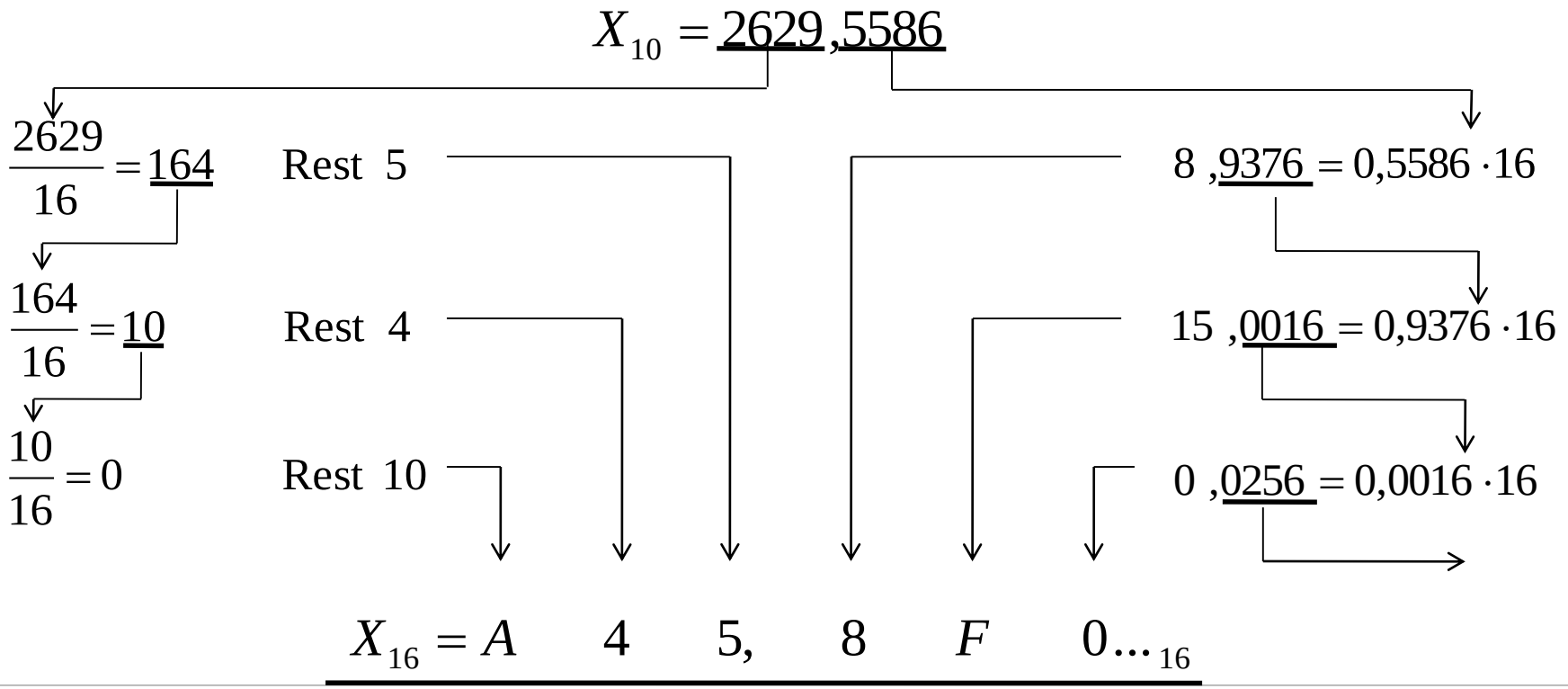
$$X_{10} = A_{16} \cdot 16^2 + 4_{16} \cdot 16^1 + 5_{16} \cdot 16^0 + 8_{16} \cdot 16^{-1} + F_{16} \cdot 16^{-2}$$

$$X_{10} = 10 \cdot 16^2 + 4 \cdot 16^1 + 5 \cdot 16^0 + 8 \cdot 16^{-1} + 15 \cdot 16^{-2}$$

$$X_{10} = 2629{,}5586\ldots_{10}$$

Konvertierung einer Dezimalzahl in eine wertgleiche Festkommazahl

Beispiel: Konvertierung einer Dezimalzahl $X_{10} = 2629{,}5586$ (4 hexadezimale Nachkommastellen) in die wertgleiche Hexadezimalzahl X_{16}



6 Gleitkommazahlen (floating-point numbers)

Halblogarithmische Darstellung reeller Zahlen durch Mantisse und Exponent:

$$z = M \cdot 2^E$$

- v - Vorzeichen der Mantisse (in der Mantisse enthalten)
- M - Mantisse (Signifikant)
- f - gebrochener Teil, Nachkommateil der Mantisse (fractional part)
- E - Exponent (Charakterisitik)

Vorteil : wesentlich größerer Wertebereich und flexibler Nachkommabereich

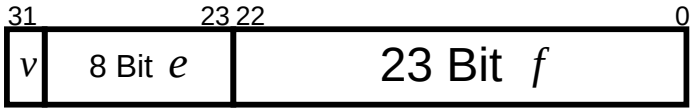
Nachteil : geringere relative Genauigkeit

Standard : ANSI / IEEE 754 von 1985
(Institutes for Electrical and Electronic Engineers)

Anwendung : Real-Arithmetik

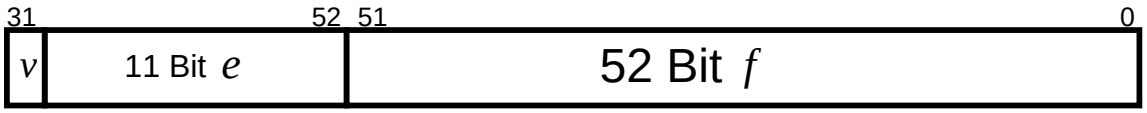
Datenformate für floating point

32-Bit Format (single)

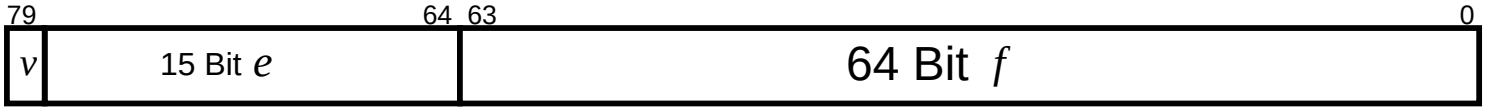


v - Vorzeichen
e - biased exponent
f - fractional part

64-Bit Format (double)



80-Bit Format (double-extended)



Mantissendarstellung normalisiert/denormalisiert

Darstellung der Mantisse *M* als Festkommazahl (signed integer in Vorzeichen-Wert-Darstellung) → zwei Darstellungsvarianten (Kennzeichnung in E):

- **normalisierte Form**: Die führende 1 und das darauf folgende Komma werden nicht explizit in der Darstellung gespeichert, sondern nur der gebrochene Anteil f (fractional part), (Ausnahme : double-extendet).
  → Vor der Verarbeitung sind die 1 und das darauf folgende Komma zu ergänzen (entpacken).
  $$M = (-1)^v \cdot 1, f$$
- **denormalisierte Form**: Die führende 1 wird durch eine 0 ersetzt. Die denormalisierte Form dient der Darstellung sehr kleiner Zahlen oder zur Exponentenanpassung bei Berechnungen
  → geringere Genauigkeit als normalisierte Darstellung.
  $$M = (-1)^v \cdot 0, f$$

Mantissendarstellung Wertebereich

Wertebereich normalisiert: $1, f: \quad 1{,}0 \le |M| < 2{,}0 \quad \text{mit} \quad 0{,}0 \le f < 1{,}0$

Wertebereich denormalisiert: $0, f: \quad 0{,}0 \le |M| < 1{,}0 \quad \text{mit} \quad 0{,}0 \le f < 1{,}0$

Genauigkeit der Mantisse (binär-dezmial):

Datenformat	Binärstellen	max. rel. Genauigkeit	Dezimalstellen
single	23	$2^{-34} = 5{,}96 \ldots 10^{-8}$	7
double	52	$2^{-53} = 5{,}96 \ldots 10^{-16}$	15
double-extended	64	$2^{-64} = 5{,}96 \ldots 10^{-20}$	19

Exponentendarstellung

Darstellung des Exponenten E als signed integer in Basiswert-Darstellung mit dem Basiswert B_e. Der Basiswert wird nicht in der Darstellung gespeichert.

$E = e - B_e$ bzw. $e = E + B_e$ e – biased exponent

Wertebereich (ohne Ausnahmesituationen):

Datenformat	b_e	B_e	Wertebereich
single	8	127	$-126 \leq E \leq 127$
double	11	1023	$-1022 \leq E \leq 1023$
double-extended	15	16383	$-16382 \leq E \leq 16383$

(b_e - Binärstellen (Exponentendarstellung), B_e - Basiswert (biased))

Ausnahmesituationen der Gleitkommadarstellung

Größter ($e = 2^{b_e} - 1$) und kleinster ($e = 0$) Exponentenwert sind für Ausnahmesituationen reserviert (→ 4 Darstellungsvarianten):

	Interpretation	V	e	f	Zahlendarstellung
	Null	v	0	0	$(-1)^v \cdot 0$
1.	denormalisiert	v	0		$(-1)^v \cdot 0, f \cdot 2^{(1-B_e)}$
2.	normalisiert	v	$0 < e < 2^{b_e} - 1$		$(-1)^v \cdot 1, f \cdot 2^{(e-B_e)}$
3.	unendlich	v	$2^{b_e} - 1$	0	$(-1)^v \cdot \infty$
4.	Not-a-Number	v	$2^{b_e} - 1$	$\neq 0$	NaN

- Zwei verschiedene Darstellungen der Null (+0 und -0) denormalisiert bei $e=f=0$.
- denormalisierte Zahlen bei $e=0$ werden mit $e=1$ interpretiert.
- Sonderdarstellungen für Unendlich, (- ∞ und + ∞) bei $e = 2^{b_e} - 1$ und $f = 0$
- Sonderdarstellung für Not-a-Number bei $e = 2^{b_e} - 1$ und $f \neq 0$

Beispiele zu Ausnahmesituationen der Gleitkommadarstellung (single precision)

v	e	f	Zahlenwert	Interpretation
0	0000 0000	00 … 00	+0	positive Null
1	0000 0000	00 … 00	-0	negative Null
0	0000 0000	00 … 01	$+0{,}00\ldots01\cdot2^{-126}$	denormalisiert
1	0000 0000	00 … 01	$-0{,}00\ldots01\cdot2^{-126}$	denormalisiert
0	0000 0001	00 … 00	$+1{,}00\ldots00\cdot2^{-126}$	normalisiert
1	0000 0001	00 … 00	$-1{,}00\ldots00\cdot2^{-126}$	normalisiert
0	1111 1110	11 …. 11	$+1{,}11\ldots11\cdot2^{+127}$	normalisiert
1	1111 1110	11 …. 11	$-1{,}11\ldots11\cdot2^{+127}$	normalisiert
0	1111 1111	00 … 00	$+\infty$	positiv unendlich
1	1111 1111	00 … 00	$-\infty$	negativ unendlich
0	1111 1111	00 … 01	NaN	Not-a-Number

Addition und Subtraktion von floating point

$$
\begin{aligned}
z_1 &= M_1 \cdot 2^{E_1} = (-1)^{v_1} \cdot 1, f_1 \cdot 2^{(e_1 - B_e)} \\
z_2 &= M_2 \cdot 2^{E_2} = (-1)^{v_2} \cdot 1, f_2 \cdot 2^{(e_2 - B_e)} \\
z_1 + z_2 &= (M_1 + M_2 \cdot 2^{-(E_1 - E_2)}) \cdot 2^{E_1} \\
&= ((-1)^{v_1} \cdot 1, f_1 + (-1)^{v_2} \cdot 1, f_2 \cdot 2^{-(e_1 - e_2)}) \cdot 2^{(e_1 - B_e)}
\end{aligned}
$$

Analog für denormalisierte Darstellungen

Bei ungleichen Exponenten e_1, e_2 muss die Zahl mit dem kleineren Exponenten denormalisiert werden (Erhöhung des Exponenten bei gleichzeitiger Rechtsverschiebung des Signifikanten).

Addition und Subtraktion von floating point

Algorithmus für vorzeichenbehaftete Addition $z_s = z_1 + z_2$:

$$z_s = (M_1 + M_2 \cdot 2^{-(e_1 - e_2)}) \cdot 2^{E_1}$$

1. Wenn $e_1 < e_2$,dann vertausche z_1 und z_2 .
2. Verschiebe M_2 um $e_1 - e_2$ Stellen nach rechts → Denormalisierung.
3. Addiere bzw. Subtrahiere M_1 und M_2 analog zu signed integer im Zweierkomplement. Bei negativem Ergebnis Konvertierung in die Vorzeichen-Wert-Darstellung.
4. Normalisierung und Rundung des Ergebnisses.

Beispiel Addition und Subtraktion von floating point (single precision)

Beispiel (single): (0 - Zusatzstelle wegen 2-Komplement Rechnung)

z	v	e	s	Bemerkung	Dez.
z_1	0	1000 0001	^{0} 1,0000...	normalisiert	4,0
z_2	0	0111 1111	^{0} 1,1000...	normalisiert	1,5
z_2	0	1000 0001	^{0} 0,0110...	denormalisiert	1,5
$z_1 + z_2$	0	1000 0001	^{0} 1,0110...	normalisiert	5,5
$-z_2$	1	1000 0001	^{1} 1,1010...	denormalisiert 2-Komplement	-1,5
$z_1 - z_2$	0	1000 0001	^{0} 0,1010…	denormalisiert	2,5
$z_1 - z_2$	0	1000 0000	^{0} 1,0100...	normalisiert	2,5

Multiplikation und Division von floating point

$$
\begin{aligned}
z_1 &= M_1 \cdot 2^{E_1} = (-1)^{v_1} \cdot 1, f_1 \cdot 2^{(e_1 - B_e)} \\
z_2 &= M_2 \cdot 2^{E_2} = (-1)^{v_2} \cdot 1, f_2 \cdot 2^{(e_2 - B_e)} \\
z_1 \cdot z_2 &= M_1 \cdot M_2 \cdot 2^{(E_1 + E_2)} \\
&= (-1)^{v_1} \cdot (-1)^{v_2} \cdot 1, f_1 \cdot 1, f_2 \cdot 2^{((e_1 + e_2 - B_e) - B_e)} \\
\frac{z_1}{z_2} &= \frac{M_1}{M_2} \cdot 2^{(E_1 - E_2)} \\
&= \frac{(-1)^{v_1}}{(-1)^{v_2}} \cdot \frac{1, f_1}{1, f_2} \cdot 2^{((e_1 - e_2 + B_e) - B_e)}
\end{aligned}
$$

Analog für denormalisierte Darstellungen

Multiplikation und Division von floating point

Algorithmus für $z_p = z_1 \cdot z_2$ **bzw.** $\frac{z_1}{z_2}$ **:**

$$z_p = M_1 \cdot M_2 \cdot 2^{((e_1+e_2-B_e)-B_e)} \quad \text{bzw.} \quad z_p = M_1 / M_2 \cdot 2^{((e_1-e_2+B_e)-B_e)}$$

1. Bestimmung des Vorzeichens ($v_1 \neq v_2 \rightarrow$ negativ, $v_1 = v_2 \rightarrow$ positiv).
2. Multiplikation bzw. Division von M_1 und M_2 analog zu unsigned integer.
3. Addition von e_1 und e_2 und Subtraktion von B_e bzw. Subtraktion von e_1 e_2 und Addition von B_e analog zu signed integer.
4. Normalisierung und Rundung des Ergebnisses.

Beispiel Multiplikation und Division von floting point (single precision)

Beispiel (single):

z	v	e	s	Bemerkung	Dez.
z_1	0	1000 0001	1,0000...	normalisiert	4,0
z_2	0	0111 1111	1,1000...	normalisiert	1,5
$z_1 \cdot z_2$	0	1000 0001	1,1000...	normalisiert	6,0
$\frac{z_1}{z_2}$	0	0111 1101	1,1000...	normalisiert	0,375

Rundung von Gleitkommazahlen

Der IEEE 754-Standart fordert:

Das Ergebnis einer arithmetischen Operation soll das dasselbe sein, als würde exakt gerechnet und anschließend gerundet (Runden zum nächsten Wert, die Hälfte auf den geraden Wert (round-to-even)).

Verschiedene Rundungsmodi:

- nach 0
- Nach $+\infty$ oder $-\infty$
- zum nächsten Wert, bei 0,5 auf den geraden Wert (round-to-even)

Relevante Bitstellen für die Rechnung (auf q Stellen, vom MSB gezählt):

- guard Bit g – Bitstelle auf die gerundet wird (q Stellen vom MSB)
- round Bit r – Rundungsbit (q+1 Stellen vom MSB)
- sticky Bit s – Zusatzstellen (q+2, q+3, … Stellen vom MSB bis zum LSB)

Rundung von Gleitkommazahlen, Schema

Beispiel (Rundung von $n = 8$-Bit Zahlen auf $q = 4$ Stellen):

Nr.	n-Bit Zahl			gerundet
1	1 , 0 0 0	1 0 0 0	⟶	1 , 0 0 0
2	1 , 0 0 0	1 0 0 1	⟶	1 , 0 0 1
3	1 , 0 0 1	1 0 0 0	⟶	1 , 0 1 0
4	1 , 0 0 1	1 1 1 1	⟶	1 , 0 1 0
5	1 , 1 1 1	1 0 0 0	⟶	1 0 , 0 0
	g	$r - s -$		

Ausschlaggebend für die Rundung ist, ob in den s-Bits (restliche Stellen) überhaupt eine 1 steht, nicht die Anzahl. Für eine genaue Rundung sind mehrere s-Bits erforderlich (z.B. s=3).

Rechnen mit 0, $\pm\infty$, NaN

Bei folgenden Berechnungen ist des Ergebnis NaN:

$\frac{\pm 0}{\pm 0}$, $\frac{\pm\infty}{\pm\infty}$, $\pm 0 \cdot \pm\infty$, $+\infty - \infty$, $\sqrt{b}$, $\log b$, für $b < 0$, ...

Ist bei einer Berechnung mindestens einer der Operanden NaN so ist das Ergebnis ebenfalls NaN.

Ergebnisse beim Rechnen mit 0, $\pm\infty$:

$0^0 = 1,\ \infty^0 = 1,\ 1^\infty = 1$

$\frac{a}{0} = sgn(a)\infty$ für $a \neq 0$, $\frac{a}{\infty} = 0$ für $a \neq \infty$

Probleme beim Rechnen mit Gleitkommazahlen

- Berechnungen mit denormalisierten Zahlen haben höhere Ungenauigkeit (begrenzte Darstellung betragsmäßig kleiner Werte).
- Überlauf Wertebereichsüberschreitungen ergeben $+\infty$ bzw. $-\infty$.
- Unterlauf (denormalisierte Darstellung) ergibt 0.
- Rundungsproblematik.

Assoziativ- und Distributivgesetz gelten nur eingeschränkt:

Ursachen für die Differenzen sind die endliche Stellenzahl zur Zahlendarstellung, die Rundung der Ergebnisse sowie Über- und Unterläufe.

$$a+(b+c) \neq (a+b)+c$$

$$a\cdot(b\cdot c) \neq (a\cdot b)\cdot c$$

$$(a+b)\cdot c \neq (a\cdot c)+(b\cdot c)$$

7 Zusammenfassung

- Stellenwertsystem zur Basis 2 als Grundlage für Zahlendarstellungen und Verwendung binärer Blockcode.
- Gesonderte Datenformate für Natürliche-, Ganze- und Gleitkommazahlen. Betrachtung der Grundrechenoperationen + - * /
- Unterschiedliche Datenformate für Ganze Zahlen (signed integer).
- Zweierkomplementdarstellung für signed integer am gebräuchlichsten.
- Verschiedene Datenformate für Gleitkommazahlen (floating point).
- Ausnahmesituationen bei floating point sind zu beachten.
- Konvertierung verschiedener Datenformate ineinander.
- Wertebereiche sind bei Zahlendarstellungen, Über- und Unterläufe sind bei Rechenoperationen immer zu beachten.

Beispiel: 32-bit Datenformate

32-bit unsigned integer

Bit 0 – 31 Wert der Dualzahl

32-bit signed integer

Vorzeichen-Wert

V Bit 31 Vorzeichen

Bit 0 – 30 Betrag der Dualzahl

V=1 : Vorzeichen negativ

Basiswert

Bit 0 – 31 Wert der Dualzahl - Basiswert B

Zweierkomplement

V Bit 31 Vorzeichen

V=0: Bit 0 – 31 Wert der Dualzahl

V=1: Bit 0 – 31 Betrag der Dualzahl im 2-Komplement (umwandeln), Vorzeichen negativ

32-bit floating point

V Bit 31 Vorzeichen

V=1 : Vorzeichen negativ

Bit 0 – 30 fractional part f

Bit 23 – 30 biased exponent e