OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Zahlendarstellung

Sebastian Breidel

Daniel Franke

Robert Oster

Martin Scherner

Paul Schnürer

Gliederung

- Zahlendefinition
- Darstellungsformen
- Effizientes Umformen
- Addierer
- Negative Binärzahlen
- Darstellung von Kommazahlen

Zahlendefinition

- Eine Zahl ist ein mathematisches Konstrukt, das zum Zählen und Vergleichen von Größen dient

- Vorsicht: verschiedene Darstellungsmöglichkeiten

$101|_{10}$ ↯ $101|_{2}$

Bekannte Darstellungsformen:

- Zählsysteme
  - Aufzählung einzelner Summationsterme
  - Beispiel römisches Zählsystem:
    - 22 = 10+10+1+1=DDII

- Stellenwertsysteme
  - Geordnete Sequenz von Stellen
  - Position bestimmt den Wert der Ziffer
  - Besteht aus Basis b und Potenzen dieser Basis

Zahlberechnung im Stellenwertsystem

- Basis b>1
- Der Wert an jeder Ziffernposition entspricht einer Potenz von b
- n-te Position = b^{n-1}
- Zahl x = $x_{n-1} \cdot b^{n-1} + x_{n-2} \cdot b^{n-2} + \ldots + x_1 \cdot b^1 + x_0 \cdot b^0$

Diese Stellenwertsysteme werden auch „b-adische“ Systeme genannt

Darstellungsformen

- Aufbau eines „b-adischen“ Systems:
  - Stellenwertsystem zur Basis b (ganze Zahl)
  - Menge von b Symbolen: 0,1,...,b-1 (Ziffern)
- Wichtige Stellenwertsysteme:
  - b=2 Binär/Dual - $x_i \in \{0,1\}$
  - b=8 Oktal - $x_i \in \{0,1,...,7\}$
  - b=10 Dezimal - $x_i \in \{0,1,...,9\}$
  - b=16 Hexadezimal - $x_i \in \{0,1,...,9,A,...,F\}$
- Darstellung und Umwandlung einer Zahl X:
  - $(X_{n-1}\ X_{n-2} \dots X_1\ X_0)_b$ mit $x_i \in \{0,1,...,b-1\}$ und $0 <= i < n$

$$= X_{n-1} \bullet b^{n-1} + X_{n-2} \bullet b^{n-2} + \dots + x_1 \bullet b^1 + x_0 \bullet b^0 = X_{dez}$$

→ viele Multiplikationen → viel Rechenaufwand

Horner-Schema + kaskadiertes Horner-Schema:

- Grundidee (x ausklammern):
  - $a_0 \bullet x^0 + \ldots + a_n \bullet x^n = a_0 + x \bullet (a_1 \bullet x^0 + \ldots + a_n \bullet x^{n-1}) \rightarrow$ weniger Multiplikationen
- Horner-Schema (b-adische Darstellung nach Dezimaldarstellung):
  - tabellarische Umsetzung der Grundidee
  - **Problem:** teilweise müssen große Zahlen multipliziert werden
  - Beispiel: $X = 30232|_4$

3	0	2	3	2
0	12	48	200	812
3	12	50	203	**814**

→ X im Dezimalsystem

„mal 4“ „mal 4“ „mal 4“ „mal 4“

Effizientes Umformen

Horner-Schema + kaskadiertes Horner-Schema:

- Verbesserung: kaskadiertes Horner-Schema
  - Idee: verwende „Einer“ zur Addition
    bearbeite „Zehner“ wie Übertrag
  - Ausgangszahl, Überträge und Zwischenergebnisse in Spalte eintragen

    → bessere Übersicht
  - Beispiel: $X = 30232|_4$

Überträge

				3
			1 3	0
		0 1	1 2	2
		2 5	0 0	3
	2	0 0	1 3	2
	8	1	4 »	

z.B. 10=2*4+2

→ X im Dezimalsystem

Zehnersystem zu beliebigem b-adischem System:

- Grundidee:
  - X = (X Div p) • p + X Mod p → rekursiv Anwenden bis maximal zerlegt
  - X = (X Div p) • p + X Mod p
  - = (((X Div p) Div p) • p + (X Div p) Mod p) • p + X Mod p = usw.
- Inverses Hornerschema (kaskadiert):
  - tabellarische Umsetzung der Grundidee
  - Beispiel: 814 ins 4er System

Reste (MOD)

						0	2	0	8
		0	1	0	5	2	0	0	1
0	3	1	2	1	0	0	3	1	4
3		**0**		**2**		**3**		**2**	

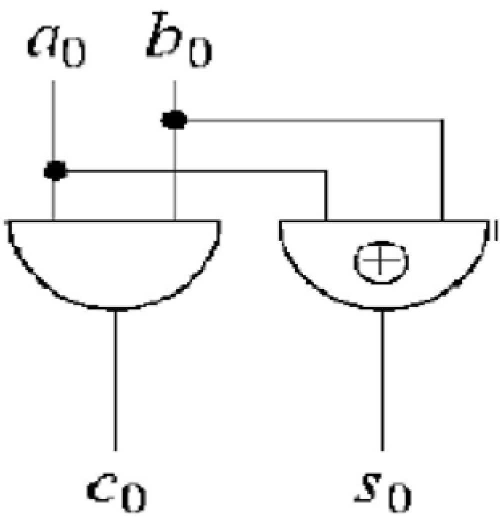
z.B. 2 = 08 DIV 4
0 = 08 MOD 4

→ X im Dezimalsystem

Addierer

- Halbaddierer
  - Berechnet die Summe aus zwei Binärzahlen
  - c(carry): gibt den möglicherweise anfallenden Übertrag aus
  - s(sum): gibt die Summe der beiden Zahlen aus

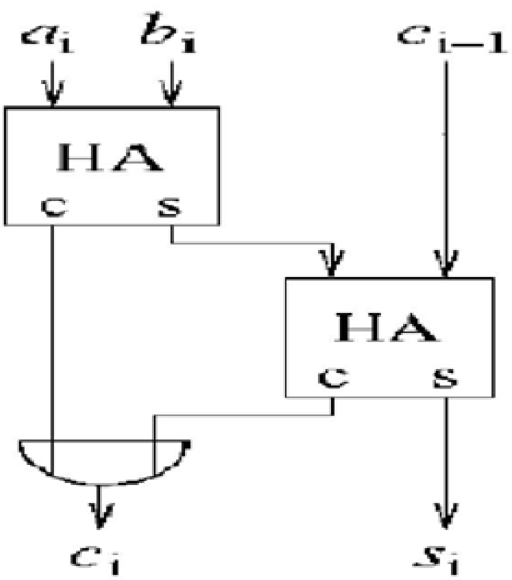
  - Schaltnetz eines Halbaddierers:



Addierer

- Volladdierer
  - Addiert zwei Binärzahlen
  - Kann den Übertrag aus vorhergehenden Additionen berücksichtigen

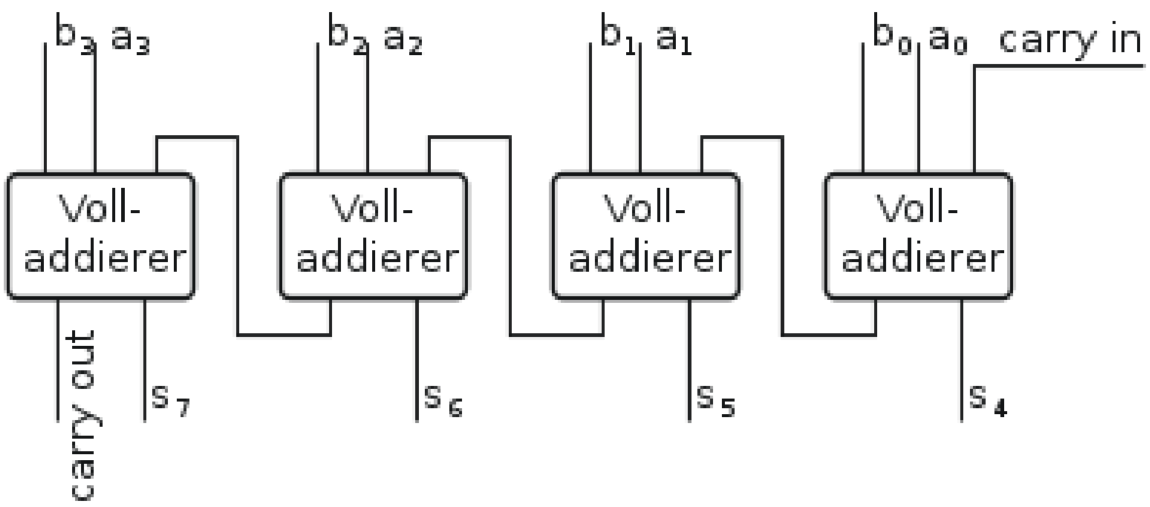
  - Aufbau eines Volladdierers aus zwei Halbaddierern und einem Oder-Gatter:



Addierer

- Carry Ripple Addierer
  - Besteht aus nacheinander geschalteten Volladdierern
  - Jeder Volladdierer erhält den Übertrag aus der vorherigen Stelle
  - Nachteil: Lange Laufzeit

  - Aus Volladdierern gebauter Carry Ripple Addierer:



Addierer

- Carry Look-Ahead Addierer
  - Idee: Überträge werden vorher berechnet
  - Generate: Übertrag an Stelle i wird erzeugt ($G_i = a_i * b_i$)
  - Propagate Übertrag an Stelle i wird propagiert ($P_i = a_i + b_i$)
  - $c_i = a_i * b_i + (a_i + b_i) * c_i - 1 := G_i + P_i * c_i - 1$
  - *Vorteil: geringere Laufzeit*
  - *Allerdings nur bei geringer Wortbreite wirklich effektiv*

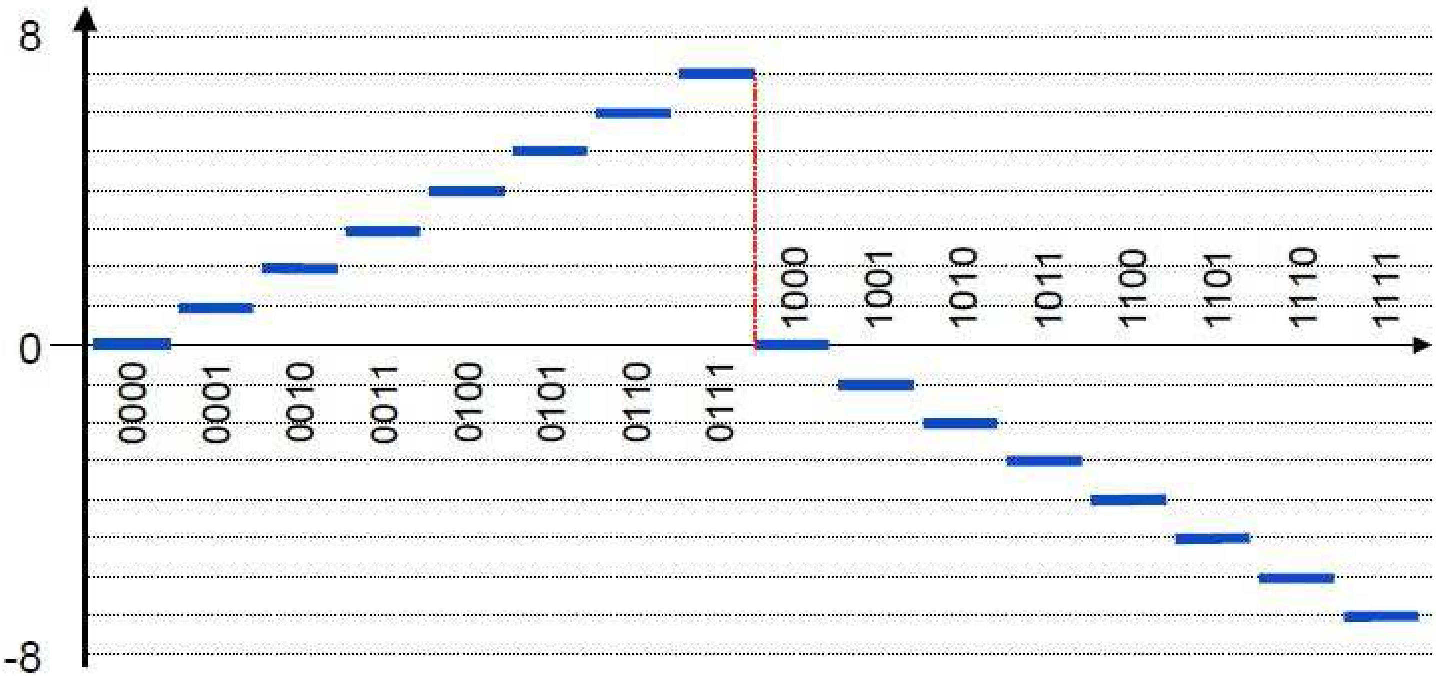
Negative Binärzahlen

Darstellung ganzer Zahlen:

- Zahlenbereich bei positiven Zahlen:
  - 8 Bit: 0 - 255
  - 16 Bit: 0 – 65‘535
  - 32 Bit: 4‘294‘967‘296
- Unterscheidung positiver und negativer Zahlen
  - Vorzeichenbit
    - Vorzeichenbit durch das erste Bit dargestellt
  - Einerkomplement
    - Negative Zahlen werden durch Invertierung gebildet
  - Zweierkomplement
    - Negative Zahlen: Invertieren und Addieren von ‚1‘

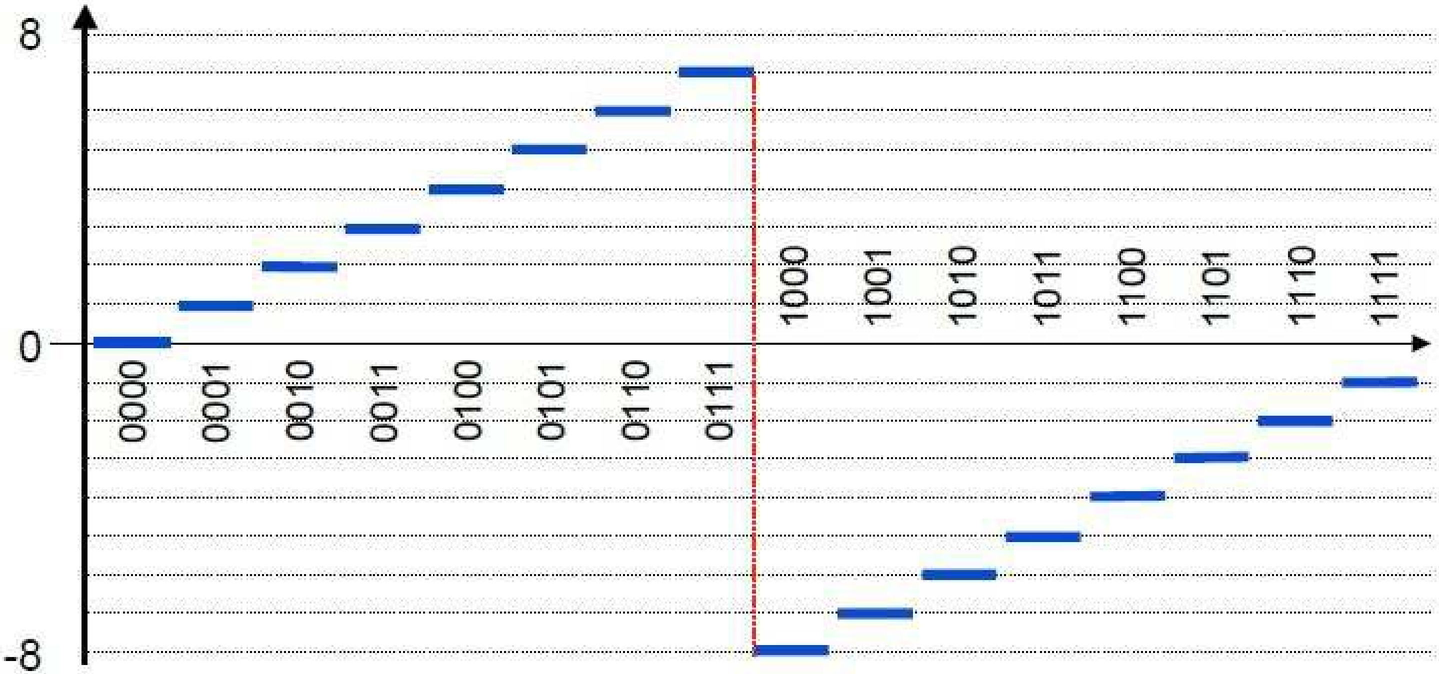
Darstellung mit Vorzeichenbit:

- Interpretation des ersten Bits als Vorzeichen
  - 0 = positive Zahl
  - 1 = negative Zahl



Zweierkomplement:

- Überführung ins Zweierkomplement durch
  - Invertierung wie im Einerkomplement
  - Addieren von 1



Wie stellt man Kommazahlen im Rechner richtig dar?

- Erweitertes Stellenwertsystem
  - Die Ziffer x_{-i} definiert die Anzahl des Bruches $b^{-i} = \frac{1}{b^i}$ in der dargestellten Zahl.

$$x = (x_{n-1} \; .... \; x_0 \; , \; x_{-1} \; .... \; x_{-k})_b$$

$$x = (x_{n-1} * b^{n-1} + \; .... \; + x_0 * b^0 + x_{-1} * b^{-1} + \; .... \; + x_{-k} * b^{-k})$$

Systematische Umwandlung: Dezimal -> b-adisch

- 1. Schritt: Ganzzahlanteil wird wie gewohnt umgewandelt
- 2. Schritt: Multiplikation des Nachkommateils mit b. Die erste Ziffer vor dem Komma des Ergebnisses dieser Multiplikation ist die nächste Stelle nach dem Komma der umgewandelten Zahl
- 3. Schritt: Streiche die Vorkommastelle und wiederhole Schritt 2.

- Spezielle Umwandlungen, wenn $b_1 = p^m$ und $b_2 = p^n$

Probleme mit Brüchen:

- Unendlich lange Sequenz zur Darstellung für endliche Zahlen häufig nötig, z.B:

$$\frac{1}{5_{[10]}} = 0{,}2_{[10]} = 0{,}\overline{0011}_{[2]}$$

- Daher wird im Computer eine etwas andere Darstellung gewählt:
  - Möglichkeit 1: Festkommazahlen (nicht praktikabel)
  - Möglichkeit 2 : Gleitkommazahlen

Gleitkommazahlen

- IEEE 754 Standard (single precision)
  - Allg.: $x = (-1)^S \times 1.\mathrm{f} \times 2^{e-b}$
  - S: Vorzeichen
  - f: gebrochener Anteil der normalisierten Mantisse
  - e: Exponent
  - b: Bias

1 Bit	8 Bit	23 Bit
S	Exponent mit Bias	Positiver, gebrochener Anteil der Mantisse

Rechnen mit Gleitkommazahlen

- Probleme
  - Auslöschung
  - Absorption
  - Ungültige Rechengesetze (Assoziativ- und Distributivgesetz
    gelten nicht mehr!)
  - Exakte Gleichheit

- Gleitkommaoperationen i.d.R. durch spezielle Rechenwerke

Ende

Vielen Dank für

Ihre

Aufmerksamkeit