

Grundlagen der Rechnerarchitektur und Betriebssysteme

Zahlen

Florian Knoch · Lehrstuhl für Praktische Informatik,
insbes. Systemnahe Programmierung, Universität Bamberg

Organisatorisches

- **online** verfügbar
- **Übungsblätter** werden im Laufe der Woche veröffentlicht
- **Lösungen** zu den Aufgaben eines Übungsblattes folgen, sobald das letzte Tutorium damit fertig ist
- **Fragen** gern per Mail

Grundlagen der Rechnerarchitektur und Betriebssysteme

Materialien zum Modul „Inf-GRABS-B“ an der Otto-Friedrich-Universität Bamberg.

Fotien

Übungsaufgaben

Übungsblätter

Über das Modul

Team

Lizenzen



Lizenziert unter [CC BY-SA 4.0](#),
sofern nicht anders angegeben.

[Datenschutz](#) · [Impressum](#)

GRABS » Übungsaufgaben » Logik und Schaltungen

Logik und Schaltungen

Kapitelübersicht

1. Gatter-Baukasten
2. Schaltungsanalyse
3. Mehrheitsentscheidung
4. Kinobesuch
5. Ein Gang, zwei Schalter
6. Addierer
7. Multiplexer
8. Sieben Segmente

[← Vorherige Seite](#)

[Nächste Seite →](#)

Addierer

Halbe Sachen?

AUFGABE 1A

Ein Halbaddierer verarbeitet zwei Eingangssignale X und Y zu einem Summensignal S und einem Übertrag C_{out} . Erstellen Sie eine Wertetabelle.

AUFGABE 1B

Rekonstruieren Sie den Schaltplan für einen Halbaddierer. Nutzen Sie die IEC-Notation.

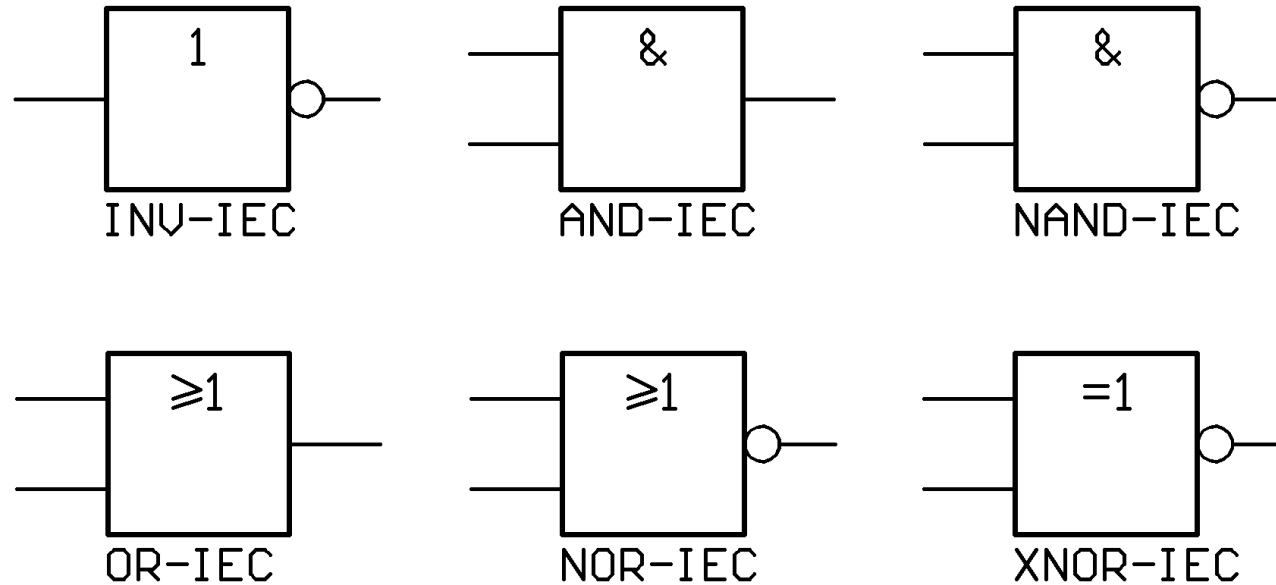


Abbildung 1: Logische Gatter in IEC-Notation

© Stefan50 / Wikimedia Commons / [CC BY-SA 3.0](#)

AUFGABE 2 A

Ein Volladdierer nimmt zusätzlich zu den Eingabebits X und Y den Übertrag C_{in} entgegen. Erstellen Sie eine Wertetabelle für dieses Bauteil.

AUFGABE 2 B

Rekonstruieren Sie den Schaltplan für einen Volladdierer.

Zahlendarstellung

Vorzeichenbehaftete Ganzzahlen

Komplementdarstellung



AUFGABE 3

Stellen Sie die Zahlen 23_{10} und -38_{10} als vorzeichenbehaftete 8-Bit-Ganzzahlen im Einerkomplement, Zweierkomplement und Exzess-Code ($b = 2^{l-1}$) dar.

Representations of floating-point data in the binary interchange formats are encoded in k bits in the following three fields ordered as shown in Figure 3.1:

- a) 1-bit sign S
- b) w -bit biased exponent $E = e + bias$
- c) $(t = p - 1)$ -bit trailing significand field digit string $T = d_1 d_2 \dots d_{p-1}$; the leading bit of the significand, d_0 , is implicitly encoded in the biased exponent E .

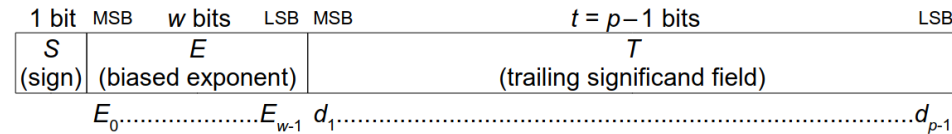


Figure 3.1—Binary interchange floating-point format

Abbildung 2: Auszug aus dem Standard IEEE 754-2008 [1]

Gleitkommazahlen

nach **Standard IEEE 754-2008**



Präzision	Gesamtlänge	Vorzeichen (v)	Exponent (e)	Mantisse (m)
single ¹	32	1	8	23
double	64	1	11	52

$$(-1)^v \times (1 + m) \times 2^{e-b}$$

¹Wir betrachten für den Rest dieses Foliensatzes ausschließlich dieses Format.

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1111	0000 0000 0000 0000 0000 000

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1111	0000 0000 0000 0000 0000 000

$$(-1)^0 \times (1 + 0) \times 2^{127-127}$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1111	0000 0000 0000 0000 0000 000

$$(-1)^0 \times (1 + 0) \times 2^{127-127}$$

$$= (-1)^0 \times (1 + 0) \times 2^0$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1111	0000 0000 0000 0000 0000 000

$$(-1)^0 \times (1 + 0) \times 2^{127-127}$$

$$= (-1)^0 \times (1 + 0) \times 2^0$$

$$= 1 \times 1 \times 1 = 1$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1110	0000 0000 0000 0000 0000 000

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1110	0000 0000 0000 0000 0000 000

$$(-1)^0 \times (1 + 0) \times 2^{126-127}$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1110	0000 0000 0000 0000 0000 000

$$(-1)^0 \times (1 + 0) \times 2^{126-127}$$

$$= (-1)^0 \times (1 + 0) \times 2^{-1}$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
0	0111 1110	0000 0000 0000 0000 0000 000

$$(-1)^0 \times (1 + 0) \times 2^{126-127}$$

$$= (-1)^0 \times (1 + 0) \times 2^{-1}$$

$$= 1 \times 0.5 \times 1 = 0.5$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
1	1000 0100	0101 0000 0000 0000 0000 000

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
1	1000 0100	0101 0000 0000 0000 0000 000

$$(-1)^1 \times (1 + 0.3125) \times 2^{132-127}$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
1	1000 0100	0101 0000 0000 0000 0000 000

$$(-1)^1 \times (1 + 0.3125) \times 2^{132-127}$$

$$= (-1)^1 \times (1 + 0.3125) \times 2^5$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m
1	1000 0100	0101 0000 0000 0000 0000 000

$$(-1)^1 \times (1 + 0.3125) \times 2^{132-127}$$

$$= (-1)^1 \times (1 + 0.3125) \times 2^5$$

$$= -1 \times 0.3125 \times 32 = -42$$

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

- gewisse Werte sind vordefiniert
- **NaN** steht für *Not a Number*
- **denormalisiert** bedeutet, dass m nicht implizit $+1$ entspricht

v	e	m	Bedeutung
0	$\bar{1}$	0	$+\infty$
1	$\bar{1}$	0	$-\infty$
v	$\bar{1}$	$\neq 0$	NaN
0	0	0	$+0$
1	0	0	-0
v	0	$\neq 0$	denormalisiert

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m	DENORMALISIERT ²
1	0000 0000	0010 0000 0000 0000 0000 000	

²da $e = 0$ und $m \neq 0$, es wird daher $e = 1$ angenommen

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m	DENORMALISIERT ³
1	0000 0000	0010 0000 0000 0000 0000 000	

$$(-1)^1 \times 0.125 \times 2^{1-127}$$

³da $e = 0$ und $m \neq 0$, es wird daher $e = 1$ angenommen

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m	DENORMALISIERT ⁴
1	0000 0000	0010 0000 0000 0000 0000 000	

$$(-1)^1 \times 0.125 \times 2^{1-127}$$

$$= (-1)^1 \times 1.4693... \times 10^{-39}$$

⁴da $e = 0$ und $m \neq 0$, es wird daher $e = 1$ angenommen

Gleitkommazahlen

nach **Standard IEEE 754-2008**



$$(-1)^v \times (1 + m) \times 2^{e-b}$$

v	e	m	DENORMALISIERT ⁵
1	0000 0000	0010 0000 0000 0000 0000 000	

$$\begin{aligned} & (-1)^1 \times 0.125 \times 2^{1-127} \\ &= (-1)^1 \times 1.4693... \times 10^{-39} \\ &\approx -1,46 \times 10^{-39} \end{aligned}$$

⁵da $e = 0$ und $m \neq 0$, es wird daher $e = 1$ angenommen



Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
float f = ... // Assume never NaN
if (f == -(-f))
    printf("true\n");
else
    printf("false\n");
```



Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
float f = ... // Assume never NaN
if (f == -(-f))
    printf("true\n");
else
    printf("false\n");
```

Garantiert wahr.

Die Negation beeinflusst lediglich das Vorzeichenbit und ist verlustfrei.

Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
float f = ... // Assume never NaN
if (f == -(-f))
    printf("true\n");
else
    printf("false\n");
```

```
float f, d = ... // Assume never NaN
if ((d + f) - d == f)
    printf("true\n");
else
    printf("false\n");
```

Garantiert wahr.

Die Negation beeinflusst lediglich das Vorzeichenbit und ist verlustfrei.

Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
float f = ... // Assume never NaN
if (f == -(-f))
    printf("true\n");
else
    printf("false\n");
```

Garantiert wahr.

Die Negation beeinflusst lediglich das Vorzeichenbit und ist verlustfrei.

```
float f, d = ... // Assume never NaN
if ((d + f) - d == f)
    printf("true\n");
else
    printf("false\n");
```

Möglicherweise falsch.

Der Ausdruck $d + f$ könnte ∞ ergeben.



Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
// Assume never NaN,  $f < \infty$ ,  $d > 0$   
float f, d = ...  
if ((f + d) > f)  
    printf("true\n");  
else  
    printf("false\n");
```

Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
// Assume never NaN,  $f < \infty$ ,  $d > 0$   
float f, d = ...  
if ((f + d) > f)  
    printf("true\n");  
else  
    printf("false\n");
```

Möglicherweise falsch.

Informationsverlust bei Addition, falls
Exponenten von f und d an unterschiedlichen
Enden des Wertebereiches.

Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
// Assume never NaN,  $f < \infty$ ,  $d > 0$   
float f, d = ...  
if ((f + d) > f)  
    printf("true\n");  
else  
    printf("false\n");
```

```
float f = ...  
if (f == f)  
    printf("true\n");  
else  
    printf("false\n");
```

Möglicherweise falsch.

Informationsverlust bei Addition, falls
Exponenten von f und d an unterschiedlichen
Enden des Wertebereiches.

Obacht: Wahr oder falsch?

Fallstricke bei Gleitkommazahlen [2]

```
// Assume never NaN,  $f < \infty$ ,  $d > 0$   
float f, d = ...  
if ((f + d) > f)  
    printf("true\n");  
else  
    printf("false\n");
```

Möglicherweise falsch.

Informationsverlust bei Addition, falls Exponenten von f und d an unterschiedlichen Enden des Wertebereiches.

```
float f = ...  
if (f == f)  
    printf("true\n");  
else  
    printf("false\n");
```

Möglicherweise falsch.

Der Ausdruck $f == f$ kann zu `false` ausgewertet werden, falls $f == \text{NaN}$.

Fragen?

- [1] „IEEE Standard for Floating-Point Arithmetic“, *IEEE Std 754-2008*, Bd. , Nr. , S. 1–70, 2008, doi: [10.1109/IEEESTD.2008.4610935](https://doi.org/10.1109/IEEESTD.2008.4610935).
- [2] D. Goldberg, „What every computer scientist should know about floating-point arithmetic“, *ACM Computing Surveys*, Bd. 23, Nr. 1, S. 5–48, März 1991, doi: [10.1145/103162.103163](https://doi.org/10.1145/103162.103163).

Kursmaterialien · **Quelltext**

Diese Präsentation wurde zuletzt am 14.07.2026 bearbeitet. Sie basiert auf den Folien der zugehörigen Vorlesung von Prof. Dr. Michael Engel. Sofern nicht anders angegeben, sind die Inhalte unter der **Lizenz CC BY-SA 4.0** verfügbar. Das Universitätslogo sowie die Schriftart UB Scala Sans sind Eigentum der Otto-Friedrich-Universität Bamberg.

Folgende **Open-Source-Komponenten** kommen in der Präsentation zum Einsatz:

circuiteria (Apache 2.0), **finite** (MIT), **fontawesome** (SIL), **pgf-tikz** (GPL 2.0), **typst** (Apache 2.0), **typst-ccicons** (MIT), **typst-polylux** (MIT).

