
Einführung in die Technische Informatik I

- ❑ Zahlendarstellung im Rechner
- ❑ Schaltnetze
- ❑ Schaltwerke
- ❑ Rechnerarithmetik



Zahlensysteme

Wert X_b der Zahl als Summe der Werte aller Einzelstellen $z_i b^i$:

$$X_b = z_n b^n + z_{n-1} b^{n-1} + \dots + z_1 b + z_0 + z_{-1} b^{-1} + \dots + z_{-m} b^{-m} = \sum_{i=-m}^n z_i b^i$$

Interessanteste Zahlensysteme in der Informatik:

b	Zahlensystem	Zahlenbezeichnung
2	Dualsystem	Dualzahl
8	Oktalsystem	Oktalzahl
10	Dezimalsystem	Dezimalzahl
16	Hexadezimalsystem	Hexadezimalzahl

Beispiel: Euklidischer Algorithmus

Umwandlung von $15741,233_{10}$ ins Hexadezimalsystem

1. $16^3 \leq 15741,233 < 16^4 \rightarrow$ höchste Potenz 16^3
2. $15741,233 : 16^3 = 3$ Rest $3453,233$
3. $3453,233 : 16^2 = D$ Rest $125,233$
4. $125,233 : 16 = 7$ Rest $13,233$
5. $13,233 : 1 = D$ Rest $0,233$
6. $0,233 : 16^{-1} = 3$ Rest $0,0455$
7. $0,0455 : 16^{-2} = B$ Rest $0,00253$
8. $0,00253 : 16^{-3} = A$ Rest $0,000088593$
9. $0,000088593 : 16^{-4} = 5$ Rest $0,000012299$
(\rightarrow Fehler)

$\rightarrow 15741,233_{10} \approx 3D7D,3BA5_{16}$

Zahlen-Umwandlung (2)

2. Methode: Abwandlung des Horner Schemas

Ganzzahligen und der gebrochenen Anteil getrennt betrachten

Umwandlung des ganzzahligen Anteils:

Eine ganze Zahl $X_b = \sum_{i=0}^n z_i b^i$ kann durch fortgesetztes Ausklammern auch in folgender Form geschrieben werden:

$$X_b = (((...(((z_n b + z_{n-1}) b + z_{n-2}) b + z_{n-3}) b \dots) b + z_1) b + z_0$$

Die gegebene Dezimalzahl wird sukzessive durch die Basis b dividiert

Horner Schema

Die jeweiligen ganzzahligen Reste ergeben die Ziffern der Zahl X_b in der Reihenfolge von der niedrigstwertigen zur höchstwertigen Stelle

Beispiel: 15741_{10} ins Hexadezimalsystem

$$15741_{10} : 16 = 983 \quad \text{Rest } 13 \quad (D_{16})$$

$$983_{10} : 16 = 61 \quad \text{Rest } 7 \quad (7_{16})$$

$$61_{10} : 16 = 3 \quad \text{Rest } 13 \quad (D_{16})$$

$$3_{10} : 16 = 0 \quad \text{Rest } 3 \quad (3_{16})$$

$$\rightarrow 15741_{10} = 3D7D_{16}$$

Umwandlung des Nachkommateils

Auch der gebrochene Anteil $\sum_{i=-m}^0 z_i b^i$ einer Zahl lässt sich entsprechend schreiben:

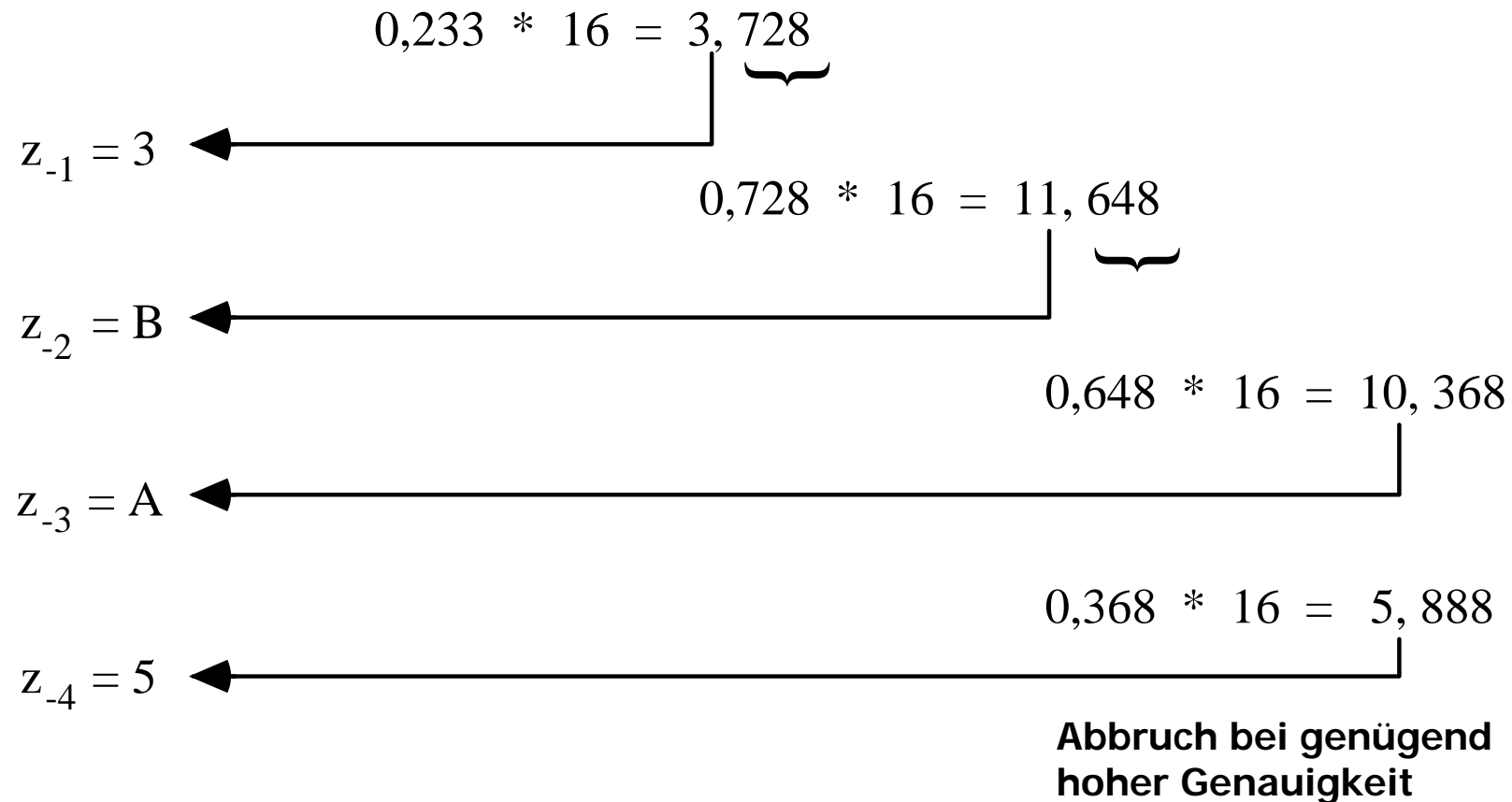
$$Y_b = (((...((y_{-m} b^{-1} + y_{-m+1}) b^{-1} + y_{-m+2}) b^{-1} + ... + y_{-2}) b^{-1} + y_{-1}) b^{-1}$$

Verfahren:

Sukzessive Multiplikation des Nachkommateils der Dezimalzahl mit der Basis b des Zielsystems → die y_{-i} in der Reihenfolge der höchstwertigen zur niedrigstwertigen Nachkommaziffer

Beispiel: Horner Schema

Umwandlung von $0,233_{10}$ ins Hexadezimalsystem:



→ $0,233_{10} \approx 0,3BA5_{16}$

Umwandlung: Basis $b \rightarrow$ Dezimalsystem

Werte der einzelnen Stellen nach nach der Stellenwertgleichung aufsummiert

Wert X_b der Zahl ergibt sich dann als Summe der Werte aller Einzelstellen $z_i b^i$:

$$X_b = z_n b^n + z_{n-1} b^{n-1} + \dots + z_1 b + z_0 + z_{-1} b^{-1} + \dots + z_{-m} b^{-m} = \sum_{i=-m}^n z_i b^i$$

Beispiel: Basis $b \rightarrow$ Dezimalsystem

Konvertiere **101101,1101**₂ ins Dezimalsystem

101101,1101

1	\rightarrow	$1 * 2^{-4} = 0,0625$
1	\rightarrow	$1 * 2^{-2} = 0,25$
1	\rightarrow	$1 * 2^{-1} = 0,5$
1	\rightarrow	$1 * 2^0 = 1$
0	\rightarrow	
1	\rightarrow	$1 * 2^2 = 4$
1	\rightarrow	$1 * 2^3 = 8$
1	\rightarrow	$1 * 2^5 = 32$

45,8125₁₀

Umwandlung beliebiger Stellenwertsysteme

- Zahl ins Dezimalsystem umwandeln
- Wandlung ins Zielsystem mit Methode 1 oder 2

Spezialfall:

Ist eine Basis eine Potenz der anderen Basis, können einfach mehrere Stellen zu einer Ziffer zusammengefasst werden oder eine Stelle kann durch eine Folge von Ziffern ersetzt werden.

Beispiel

Wandlung von **0110100,110101**₂ ins Hexadezimalsystem

$2^4 = 16 \Rightarrow$ **4 Dualstellen \rightarrow 1 Hexadezimalstelle**

dual

0110100,110101



00110100,11010100

Ergänzen von Nullen zur
Auffüllung auf Vierergruppen

hexadezimal

3 4 , D 4

Darstellung negativer Zahlen

Vier verschiedene Formate für die Darstellung negativer Zahlen im Rechner:

- ❑ Darstellung mit Betrag und Vorzeichen
- ❑ Stellenkomplement-Darstellung (Einerkomplement-Darstellung)
- ❑ Zweierkomplement-Darstellung
- ❑ Offset-Dual-Darstellung / Exzess-Darstellung

Darstellung mit Betrag und Vorzeichen

Eine Stelle wird als Vorzeichenbit benutzt.

Das ist das am weitesten links stehende Bit (MSB, most significant bit):

MSB = 0	➡	positive Zahl
MSB = 1	➡	negative Zahl

Beispiel:

0001 0010 = +18

1001 0010 = -18

Darstellung mit Betrag und Vorzeichen

Betrag durch 3 Bit:

0	000	=	0
0	001	=	1
0	010	=	2
0	011	=	3
0	100	=	4
0	101	=	5
0	110	=	6
0	111	=	7

**Symmetrischer
Zahlenbereich**

1	000	=	-0
1	001	=	-1
1	010	=	-2
1	011	=	-3
1	100	=	-4
1	101	=	-5
1	110	=	-6
1	111	=	-7

Darstellung mit Betrag und Vorzeichen

Nachteile:

- ❑ Darstellung ändert sich bei Bereichserweiterung
- ❑ Bei Addition und Subtraktion müssen die **Vorzeichen** der Operanden **gesondert** betrachtet werden → Bedingt geeignet zur Implementierung auf dem Rechner
- ❑ **Redundanz:** Es gibt zwei Repräsentationen der Null (+0 und -0)

Verbesserung: Einerkomplement-Darstellung



Einerkomplement-Darstellung

(Auch Stellenkomplement-Darstellung genannt)

Stellenkomplement der entsprechenden positiven Zahl.

Um das
Einerkomplement
zu bilden, wird
jedes Bit der
entsprechenden
positiven Zahl
negiert.

0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7

1000	=	-7
1001	=	-6
1010	=	-5
1011	=	-4
1100	=	-3
1101	=	-2
1110	=	-1
1111	=	-0

Einerkomplement-Darstellung

- Stellenkomplement der entsprechenden positiven Zahl entspricht dem Einerkomplement:

$$z_{ek} = (2^n - 1) - z$$

- Negative Zahlen sind wiederum durch ein gesetztes Bit in der ersten Stelle charakterisiert
- Bitfolge $z_n z_{n-1} \dots z_0$ hat den Wert:

$$Z = -z_n \cdot (2^n - 1) + z_{n-1} \cdot 2^{n-1} + \dots + z_0$$

Einerkomplement-Darstellung

- Symmetrischer Zahlenbereich
- Vorteil gegenüber der Darstellung mit Vorzeichenbit:
Erste Stelle bei Addition und Subtraktion muss **nicht** gesondert betrachtet werden.
- **Aber:** Es gibt weiterhin zwei Darstellungen der Null
- Beseitigung dieses Nachteils: Zweierkomplement

Zweierkomplement-Darstellung

Man addiert nach der Stellenkomplementierung noch eine 1

Man erhält so das Zweierkomplement:

$$Z_{zk} = 2^n - Z$$

Komplementbildung

$0 \dots 0 \rightarrow$ Einerkomplement $1 \dots 1$
 \rightarrow Zweierkomplement $0 \dots 0$

Zweierkomplement-Darstellung

0000 = 0

0001 = 1

0010 = 2

0011 = 3

0100 = 4

0101 = 5

0110 = 6

0111 = 7

**Unsymmetrischer
Zahlenbereich**

1000 = -8

1001 = -7

1010 = -6

1011 = -5

1100 = -4

1101 = -3

1110 = -2

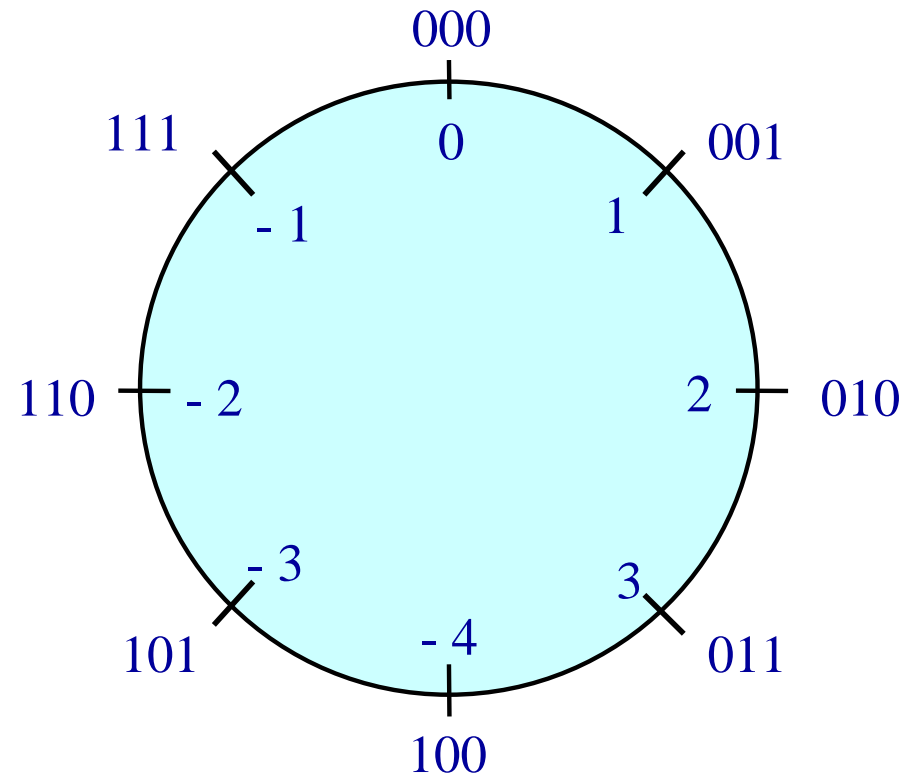
1111 = -1

Zweierkomplement-Darstellung

Nachteil:

Unsymmetrischer Zahlenbereich. Die kleinste negative Zahl ist betragsmäßig um 1 größer als die größte positive Zahl

3-Bit-ZK-Zahlen:



Zweierkomplement-Darstellung

- Alle anderen negativen Zahlen werden um 1 verschoben, das MSB (Most Significant Bit) bleibt aber gleich 1.
- Aus der ersten Stelle kann das Vorzeichen der Zahl abgelesen werden
- Bitfolge $z_n z_{n-1} \dots z_0$ hat den Dezimalwert:

$$Z = -z_n \cdot 2^n + z_{n-1} \cdot 2^{n-1} + \dots + z_0$$

$$1101 = -1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = -3$$

$$1111 \ 11101 = \dots = -3$$

Zweierkomplement-Darstellung

Beispiel (n = 31)

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010 = 2$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001 = 1$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 = 0$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 = -1$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110 = -2$$

$$\begin{aligned} 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 &= -2^{31} \\ &= -2147483648 \end{aligned}$$

$$\begin{aligned} 0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 &= 2^{31} - 1 \\ &= 2147483\ 647 \end{aligned}$$

Beispiel

Die Zahl -77_{10} soll mit 8 Bit dargestellt werden

$$77_{10} = 0100\ 1101_2$$

Mit Vorzeichenbit : $-77 = 1100\ 1101_2$

Einerkomplement : $-77 = 1011\ 0010_2$

Zweierkomplement : $-77 = 1011\ 0011_2$

Bitweise
Komplementierung

Addition von 1

Offset-Dual- (Exzess-) Darstellung

- ❑ Wird hauptsächlich bei der Exponenten-Darstellung von Gleitkommazahlen benutzt
- ❑ Die Darstellung einer Zahl erfolgt in Form ihrer **Charakteristik**
- ❑ Der gesamte Zahlenbereich wird durch Addition einer Konstanten (Exzess, Offset) so nach oben verschoben, dass die kleinste (negative) Zahl die Darstellung **0...0** erhält
- ❑ Bei **n** Stellen ist der Offset **2^{n-1}**
- ❑ Der Zahlenbereich ist hier auch asymmetrisch

Zusammenfassung der Möglichkeiten

Darstellung mit				
Dezimalzahl	Betrag + Vorzeichen	Einer- komplement	Zweier- komplement	Charakteristik
-4	- - -	- - -	1 0 0	0 0 0
-3	1 1 1	1 0 0	1 0 1	0 0 1
-2	1 1 0	1 0 1	1 1 0	0 1 0
-1	1 0 1	1 1 0	1 1 1	0 1 1
0	1 0 0/0 0 0	1 1 1/0 0 0	0 0 0	1 0 0
1	0 0 1	0 0 1	0 0 1	1 0 1
2	0 1 0	0 1 0	0 1 0	1 1 0
3	0 1 1	0 1 1	0 1 1	1 1 1

Fest- und Gleitkommazahlen

Zahlendarstellung auf dem Papier:

Ziffern	0 .. 9
Vorzeichen	+ -
Komma	,

Zahlendarstellung im Rechner:

Binärziffern	0, 1
--------------	------

→ spezielle Vereinbarungen für die Darstellung von Vorzeichen und Komma im Rechner sind erforderlich.

Fest- und Gleitkommazahlen

Darstellung des Vorzeichens:

wurde im vorigen Abschnitt behandelt

Darstellung des Kommas: 2 Möglichkeiten

- Festkommadarstellung
- Gleitkommadarstellung

Festkommazahlen

Vereinbarung:

- ❑ Das Komma sitzt innerhalb des Maschinenwortes, das eine Dualzahl enthalten soll, an einer festen Stelle
- ❑ Meist setzt man das Komma hinter die letzte Stelle
- ❑ Andere Zahlen können durch entsprechende Maßstabsfaktoren in die gewählte Darstellungsform überführt werden
- ❑ **Negative Zahlen:** meist Zweierkomplement-Darstellung

Festkommazahlen

- ❑ Datentyp "integer" (Ganzzahlen) ist ein spezielles Festkommaformat.
- ❑ Manche Programmiersprachen erlauben die Definition von Ganzzahlen unterschiedlicher Länge.

Beispiel "C":

"short int", "int", "long int", "unsigned"

Gleitkomma-Darstellung

- Zur Darstellung von Zahlen, die betragsmäßig sehr groß oder sehr klein sind, verwendet man die **Gleitkomma-Darstellung**.

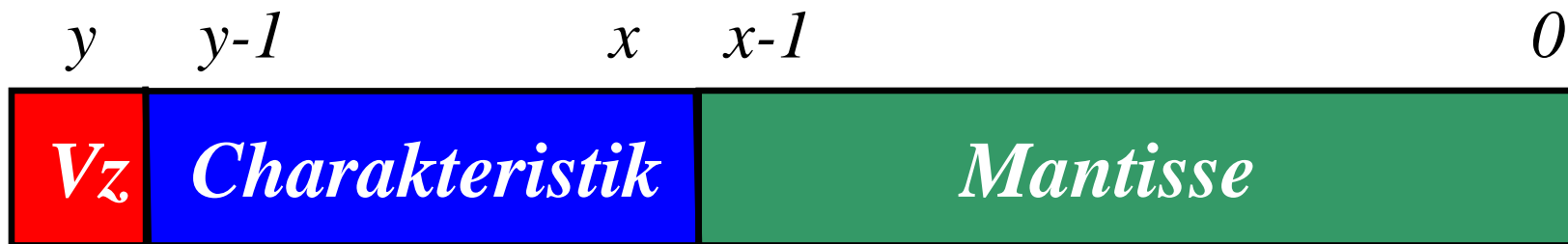
- Sie entspricht einer halblogarithmischen Form

$$X = \pm \textit{Mantisse} \cdot b^{\textit{Exponent}}$$

- Die Basis ***b*** ist für eine bestimmte Gleitkomma-Darstellung fest (meist 2 oder 16) und braucht damit nicht mehr explizit repräsentiert zu werden.
- Gleitkommazahlen werden meist **nicht** im Zweierkomplement, sondern in Betrag-Vorzeichen-Form dargestellt.

Gleitkomma-Maschinenformat

$$X = \pm \textit{Mantisse} \cdot b^{\textit{Exponent}}$$



$$X = (-1)^{V_z} * (0, \textit{Mantisse}) * b^{\textit{Exponent}}$$

$$\textit{Exponent} = \textit{Charakteristik} - b^{(y-1) - x}$$

Normalisierung

Eine Gleitkommazahl heisst **normalisiert**, wenn für den Wert der Mantisse gilt:

$$\frac{1}{b} \leq 0, \text{Mantisse} < 1$$

→ In dualer Darstellung ist die erste Stelle nach dem Komma gleich 1.

Ausnahme:

Bei der Zahl 0 sind alle Stellen der Mantisse gleich Null

Charakteristische Zahlen

Um verschiedene Gleitkomma-Darstellungen miteinander vergleichen zu können, definiert man drei charakteristische Zahlen:

- ❑ **maxreal** ist die größte darstellbare normalisierte positive Zahl
- ❑ **minreal** ist die kleinste darstellbare normalisierte positive Zahl
- ❑ **smallreal** ist die kleinste Zahl, die man zu 1 addieren kann, um einen von 1 verschiedenen Wert zu erhalten.

Beispiel

Das Assoziativgesetz $(x + y) + z = x + (y + z)$ gilt selbst dann nicht unbedingt, wenn kein *overflow* oder *underflow* auftritt.

Beispiel: $x = 1;$ $y = z = \textit{smallreal}/2$

$$\begin{aligned}(x + y) + z &= (1 + \textit{smallreal}/2) + \textit{smallreal}/2 \\ &= 1 + \textit{smallreal}/2 = 1\end{aligned}$$

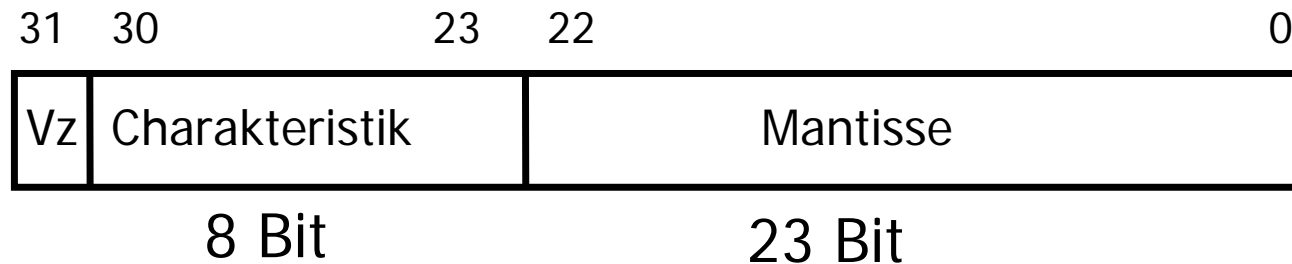
$$\begin{aligned}x + (y + z) &= 1 + (\textit{smallreal}/2 + \textit{smallreal}/2) \\ &= 1 + \textit{smallreal} \neq 1\end{aligned}$$

Problematik unterschiedlicher Definitionen

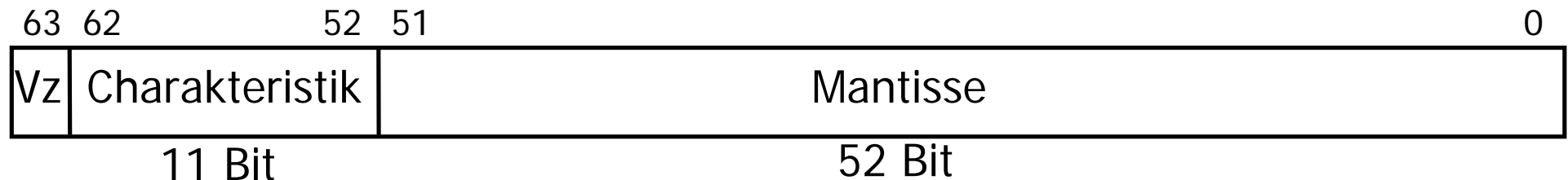
- ❑ Es existieren beliebig viele Möglichkeiten, selbst mit einer festen Wortbreite unterschiedliche Gleitkommaformate zu definieren:
 - unterschiedliche Basis b
 - Darstellung der Null
 - Anzahl der Stellen für Charakteristik und Mantisse
 - ❑ Es existierten (bis Mitte der 80er Jahre) viele verschiedene, herstellerabhängige Formate
 - ❑ Man konnte mit dem gleichen Programm auf unterschiedlichen Rechnern sehr unterschiedliche Ergebnisse erhalten
- ➔ Normierung erforderlich**

IEEE-P 754-Floating-Point-Standard

32-Bit Maschinenformate des IEEE-Standards:

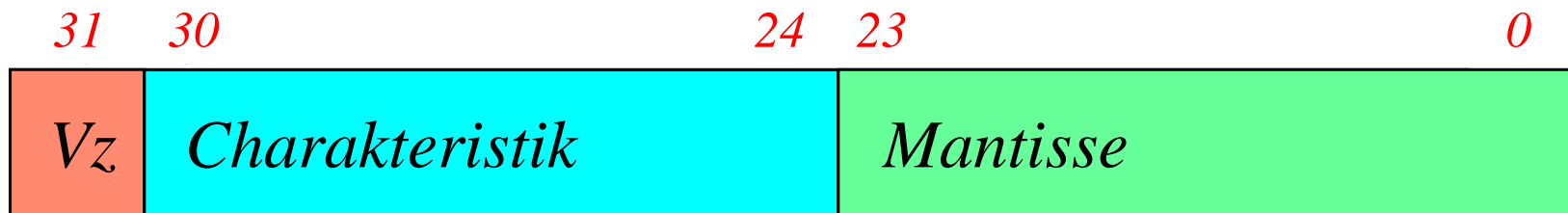


64-Bit Maschinenformate des IEEE-Standards:



IEEE-P 754 Standard (32-Bit-Format)

$$X = \pm 0, \text{Mantisse} \cdot b^{\text{Exponent}}$$



$$\text{Dezimalzahl} = (-1)^{V_z} * (\text{1}, \text{Mantisse})_2 * b^{\text{Exponent}}_{10}$$

$$\text{Exponent} = \text{Charakteristik} - b^{(y-1) - x}$$

$$\text{IEEE 754:} \quad b = 2$$

Literatur

- IEEE Computer Society:

IEEE Standard for Binary Floating-Point Arithmetic

ANSI/IEEE Standard 754-1985, SIGPLAN Notices,
Vol. 22, No. 2, pp 9-25, 1978

- D. Goldberg:

What every computer scientist should know about floating point arithmetic

ACM Computing Surveys, Vol. 13, No. 1, pp. 5-48, 1991

Schaltnetze

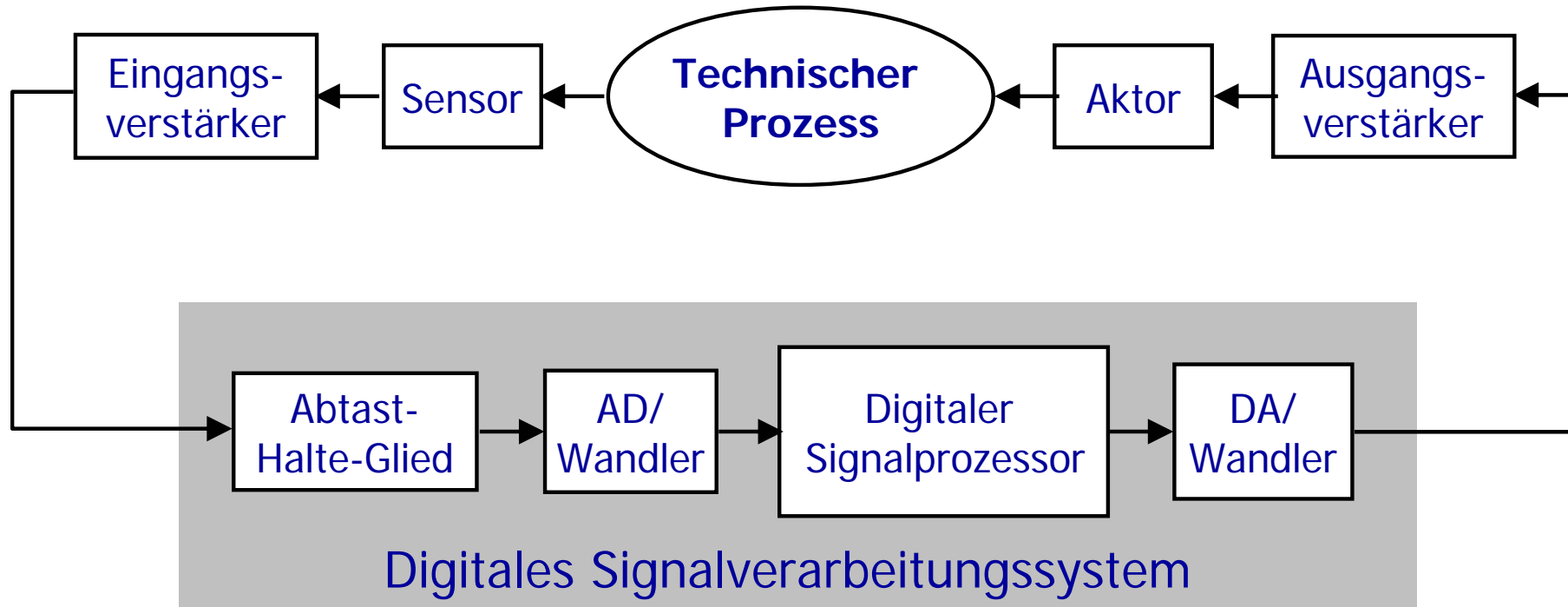
Schaltnetze:

- Rein kombinatorische logische Schaltungen
- Kein Speicherverhalten
- Realisierung logischer Funktionen

Beispiel: Licht-Aus Warnung im Kraftfahrzeug

„Motor aus“ und „Tür auf“ und „Licht an“ \Rightarrow Alarm

Aufbau eines digitalen Signalverarbeitungssystems



- Informationserfassung mittels Sensoren
- Sensordaten „digitalisieren“
- Algorithmen der digitalen Signalverarbeitung
- Verarbeitetes Signal wieder in ein analoges Signal umwandeln

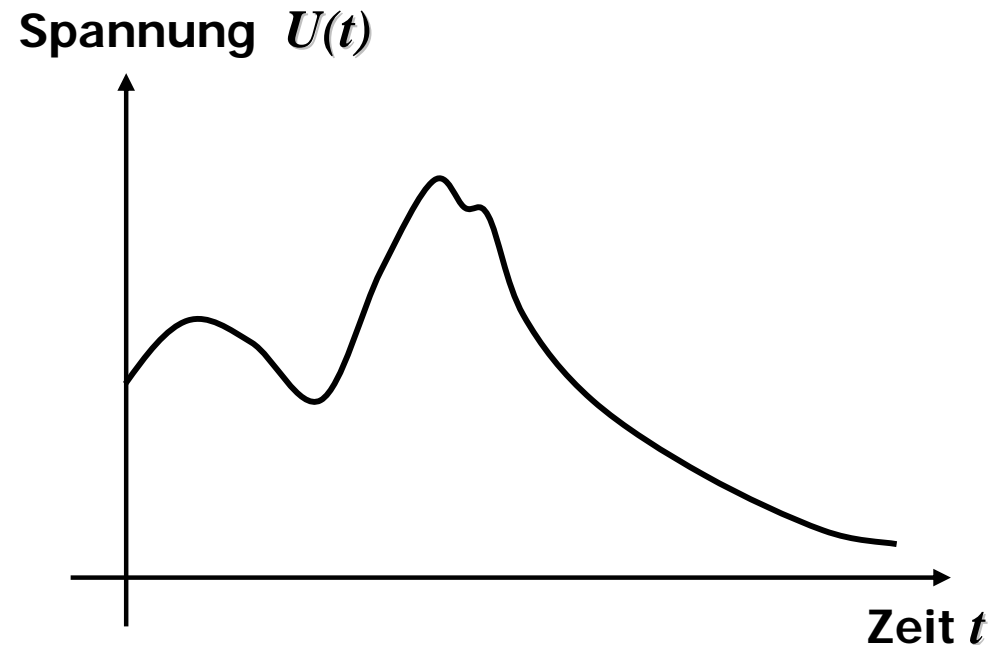
Aufbau eines digitalen Signalverarbeitungssystems

- ❑ Eingangsverstärker zur Verstärkung des Sensorsignals und dessen Umsetzung in den erforderlichen Spannungsbereich
- ❑ Abtast-Halte-Glied zur periodischen Abtastung des Eingangssignals. Der abgetastete Wert wird innerhalb einer Abtastperiode konstant gehalten
- ❑ Eingangsverstärker mit *Anti-aliasing*-Filter zur Beseitigung von hohen Störfrequenzen des Sensorsignals
- ❑ Das Ausgangsverstärker sorgt für die Glättung des vom DA/Wandler kommende Signal (Rekonstruktionsfilter)

Kontinuierliche und diskrete Signale (1)

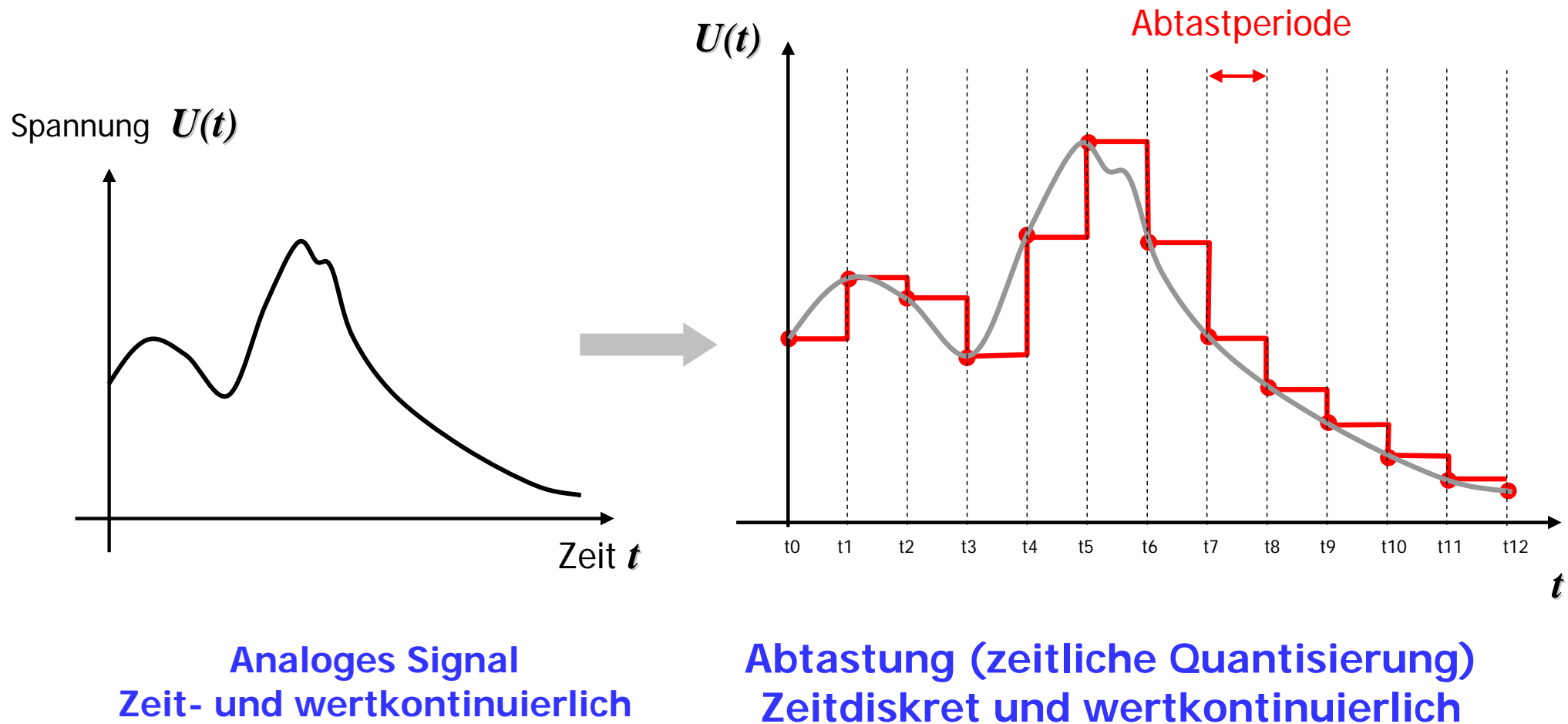
Signal als physikalischer Träger einer Information

- Ein Signal ist eine Funktion einer unabhängigen Variable t , die gewöhnlich die Zeit repräsentiert. Das Signal wird als $U(t)$ angegeben.
- Analoges Signal: $U(t)$ ist zu jedem Zeitpunkt definiert und kann jeden beliebigen Wert annehmen (Signal mit kontinuierlichen Werten).



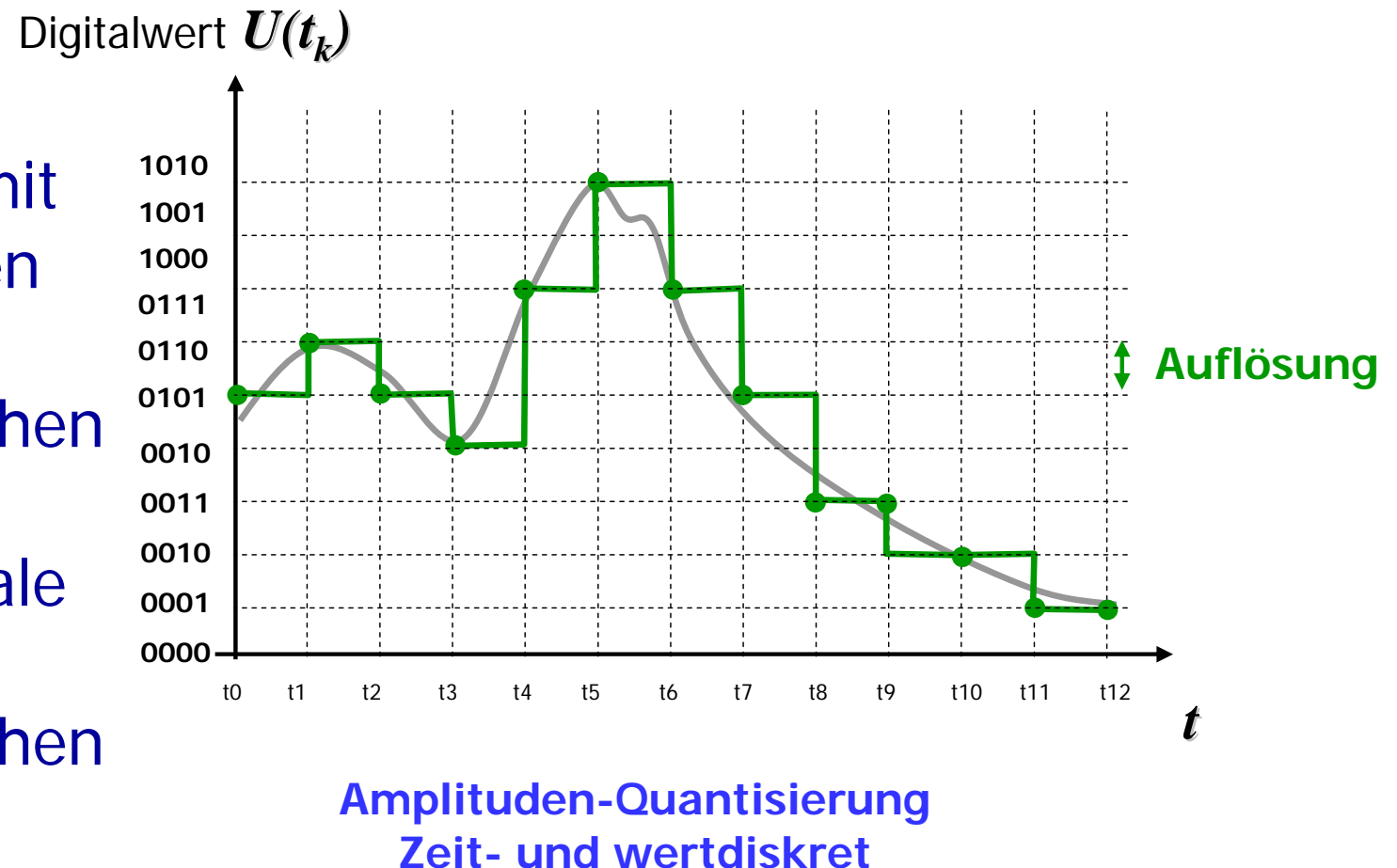
Analoges Signal
Zeit- und wertkontinuierlich

Kontinuierliche und diskrete Signale (2)

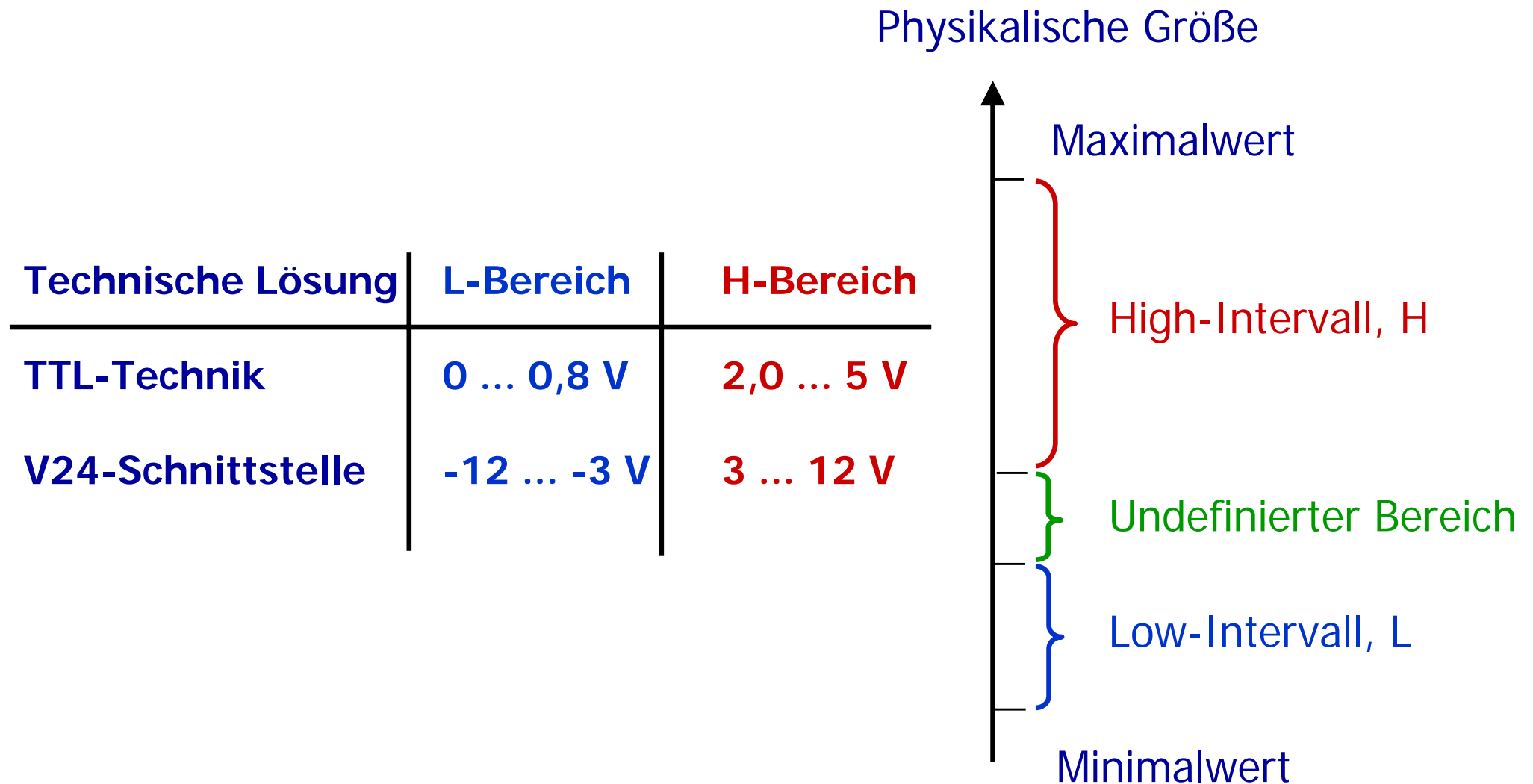


Kontinuierliche und diskrete Signale (3)

- Signal $U(t_k)$ mit einer endlichen Anzahl an unterschiedlichen Werten
- Wichtig: Signale mit zwei unterschiedlichen Werten



Binäre Signale



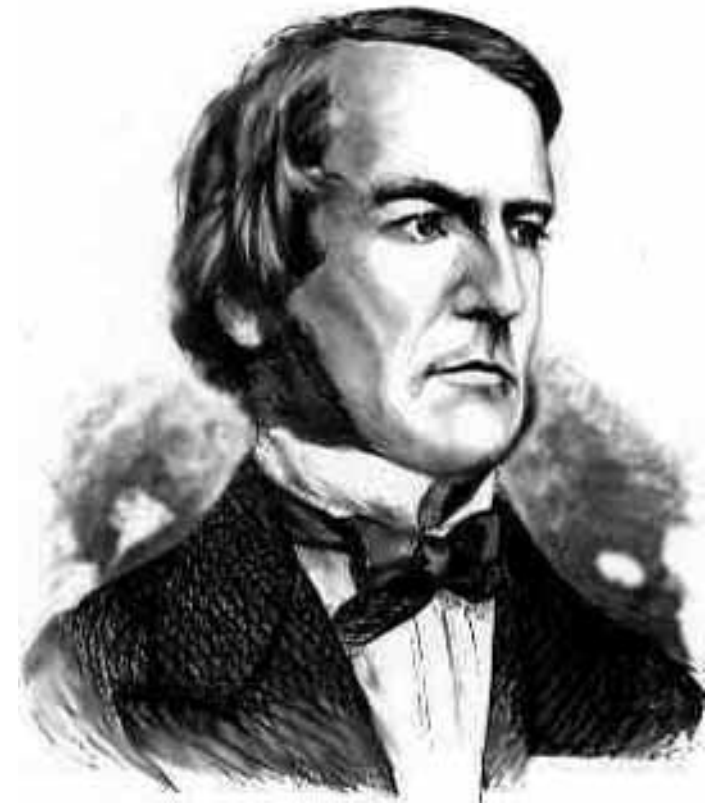
3.1 Formale Grundlagen

Zur Untersuchung und Beschreibung der Eigenschaften und des Verhaltens von logischen Funktionen ist die **Boolesche Algebra** hervorragend geeignet.

Entwickelt wurde sie von dem Mathematiker **George Boole** (1815 – 1864) als Algebra der Logik.

George Boole 1815-1864

- Englischer Mathematiker und Logiker.
- Mit 16 Jahren Lehrer. Später eröffnete er eine eigene Schule
- Boole konnte jedoch nicht an einer Universität studieren, da er das Einkommen aus seiner Schule für seine Eltern benötigte.
- Im Jahre 1849 erhielt Boole einen Mathematik-Lehrstuhl am Queens-College in Cork (Irland). Dort lehrte er als außergewöhnlicher und hingebungsvoller Lehrer für den Rest seines Lebens
- 1854 publizierte Boole: *An investigation into the Laws of Thought, on Which are founded the Mathematical Theories of Logic and Probabilities*: Beschreibung der Logik auf neue Weise mit nur einer Algebra (**Algebra der Logik**), die so die Logik mit der Mathematik verband.



Boolesche Algebra

Definition 2.1:

Als eine Boolesche Algebra bezeichnet man eine Menge $V = \{a, b, c, \dots\}$, auf der zwei zweistellige Operationen \oplus und \otimes derart erklärt sind, dass durch ihre Anwendung auf Elemente aus V wieder Elemente aus V entstehen (**Abgeschlossenheit**).

Verknüpfungsgebilde: $BA = [V, \oplus, \otimes]$

Weiterhin müssen die vier **Huntingtonschen Axiome (H1-H4)** gelten:

Huntingtonsche Axiome

H0. Abgeschlossenheit: Für alle $a, b \in V$ gilt:

$$a \otimes b \in V$$

$$a \oplus b \in V$$

H1. Kommutativgesetze $a \otimes b = b \otimes a$

$$a \oplus b = b \oplus a$$

H2. Distributivgesetze $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

$$a \oplus (b \otimes c) = (a \oplus b) \otimes (a \oplus c)$$

H3. Neutrale Elemente: Es existieren zwei Elemente $e, n \in V$, so dass gilt:

$$a \otimes e = a \quad (e \text{ wird } \underline{\text{Einselement}} \text{ genannt)}$$

$$a \oplus n = a \quad (n \text{ wird } \underline{\text{Nullelement}} \text{ genannt)}$$

H4. Inverse Elemente: Für alle $a \in V$ existiert ein Element \bar{a} , so dass gilt:

$$a \otimes \bar{a} = n$$

$$a \oplus \bar{a} = e$$

Schaltalgebra

Die Schaltalgebra ist eine spezielle Boolesche Algebra, die durch folgende Korrespondenztabelle definiert wird:

Boolesche Algebra	Schaltalgebra
V	$\{0,1\}$
\oplus	\vee ← Disjunktion
\otimes	\wedge ← Konjunktion
\mathbf{n}	0
\mathbf{e}	1
\overline{a}	\overline{a}

Schreibweise: $a + b$ für $a \vee b$
 $a \& b$ für $a \wedge b$

Schaltalgebra (2)

Schaltalgebra: $SA = [\{0,1\}, \wedge, \vee]$

\downarrow \searrow
Konjunktion Disjunktion

Die Verknüpfungen können leicht in Funktionstabellen dargestellt werden:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

a	\bar{a}
0	1
1	0

Schaltalgebra (3)

Huntingtonsche Axiome in der Schaltalgebra:

H0: **Abgeschlossenheit**

H1: $a \vee b = b \vee a$

$$a \wedge b = b \wedge a$$

H2: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

H3: $a \vee 0 = a$

$$a \wedge 1 = a$$

H4: $a \wedge \overline{a} = 0$

$$a \vee \overline{a} = 1$$

Schaltalgebra (4)

Aus den vier Huntington'schen Axiomen lassen sich weitere Sätze ableiten:

Assoziativgesetze:

$$(a \wedge b) \wedge c = a \wedge (b \wedge c)$$
$$(a \vee b) \vee c = a \vee (b \vee c)$$

Idempotenzgesetze:

$$a \wedge a = a$$
$$a \vee a = a$$

Absorptionsgesetze:

$$a \wedge (a \vee b) = a$$
$$a \vee (a \wedge b) = a$$

DeMorgan-Gesetze:

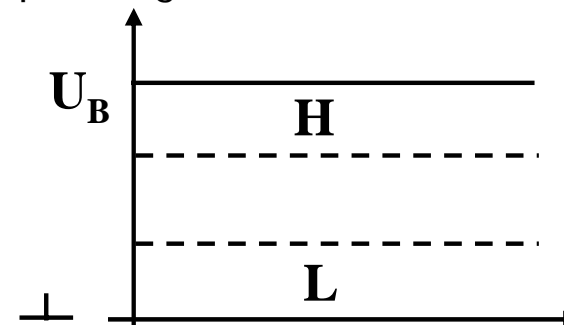
$$\overline{a \wedge b} = \bar{a} \vee \bar{b}$$
$$\overline{a \vee b} = \bar{a} \wedge \bar{b}$$

Schaltalgebra



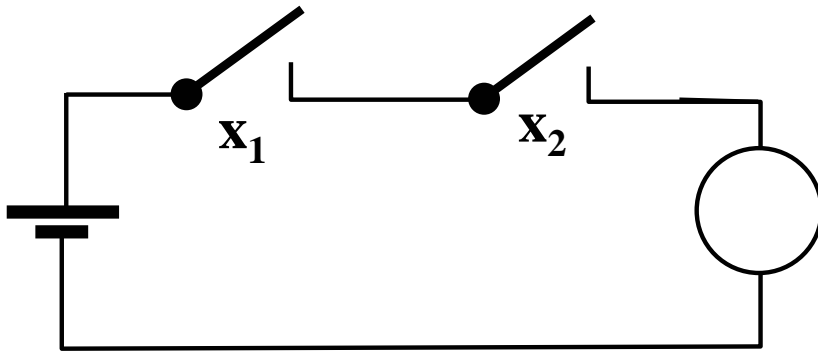
Zweiwertige Signaldarstellung:

physikalische Größe:
Spannung

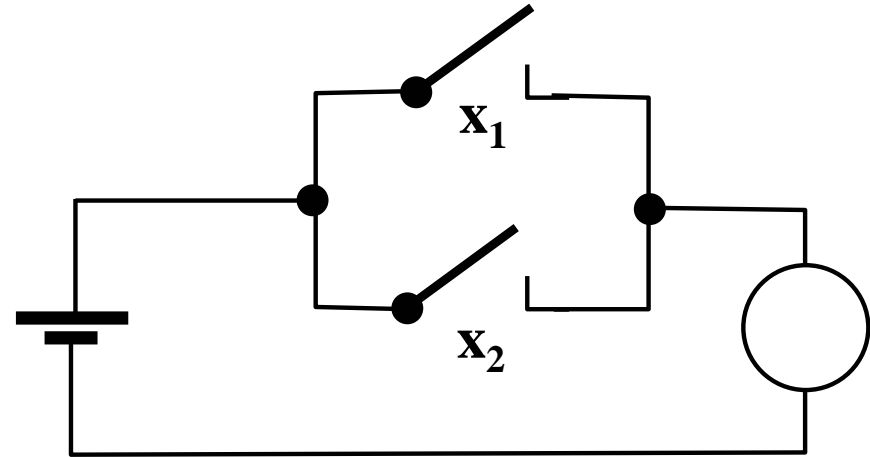


Schaltalgebra

Operationen:



Serienschaltung (ser)



Parallelschaltung (par)

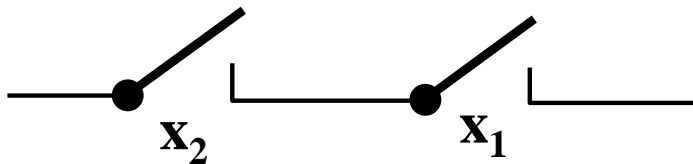
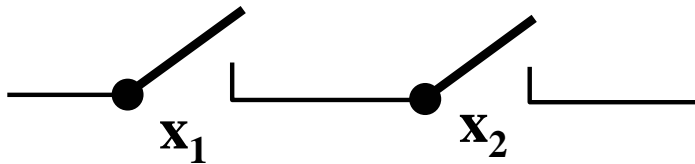
H0: Abgeschlossenheit

$$x_1 \text{ ser } x_2 \in \{L, H\}$$

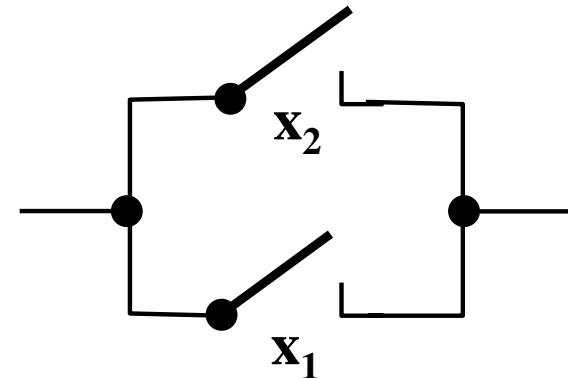
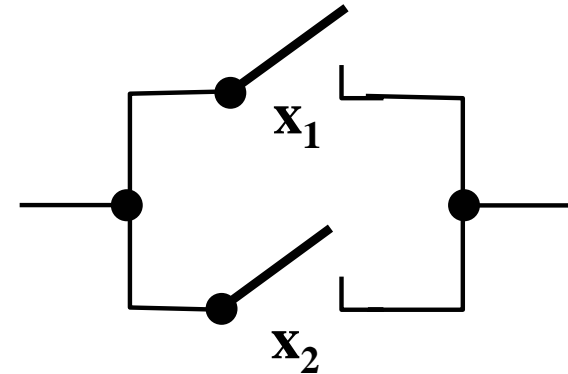
$$x_1 \text{ par } x_2 \in \{L, H\}$$

Schaltalgebra

H1: Kommutativgesetze



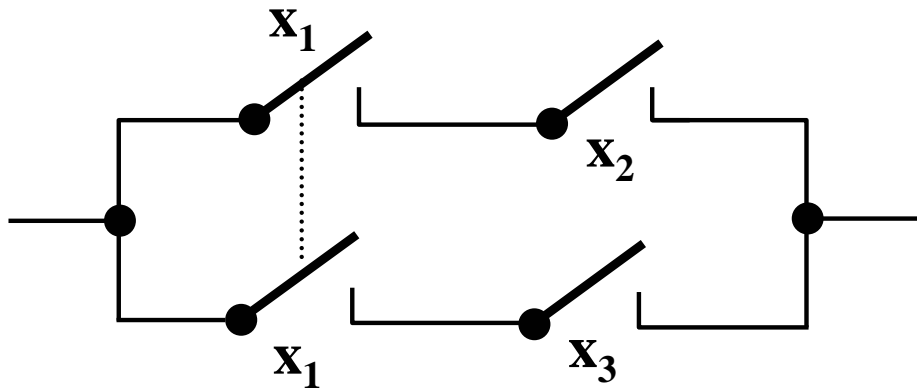
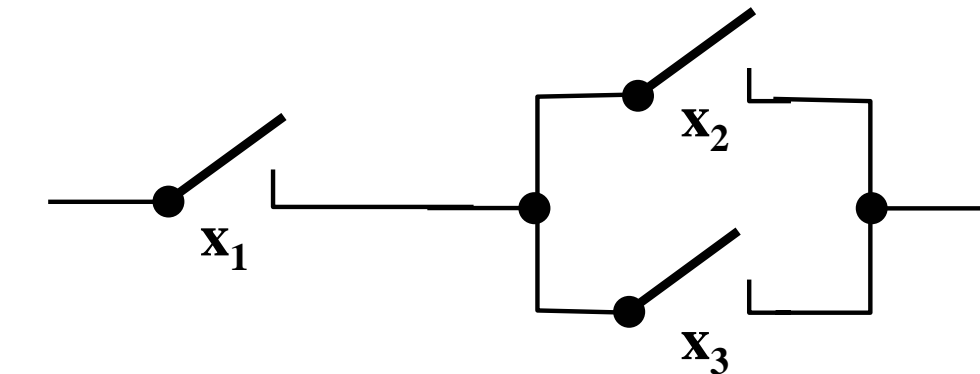
$$x_1 \text{ ser } x_2 = x_2 \text{ ser } x_1$$



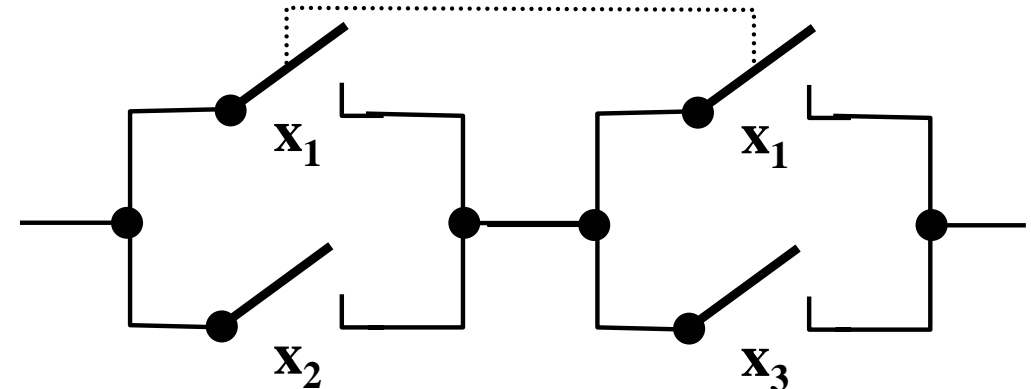
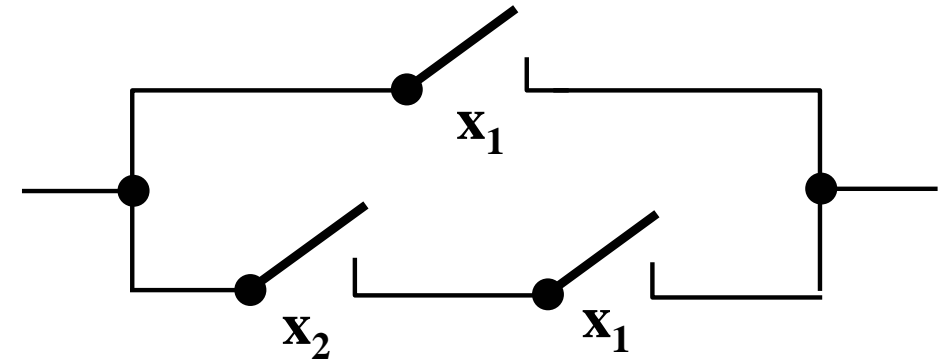
$$x_1 \text{ par } x_2 = x_2 \text{ par } x_1$$

Schaltalgebra

H2: Distributivgesetz



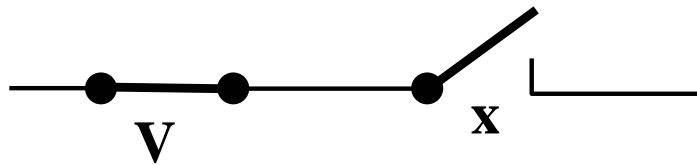
$$x_1 \text{ ser } (x_2 \text{ par } x_3) = \\ (x_1 \text{ ser } x_2) \text{ par } (x_1 \text{ ser } x_3)$$



$$x_1 \text{ par } (x_2 \text{ ser } x_3) = \\ (x_1 \text{ par } x_2) \text{ ser } (x_1 \text{ par } x_3)$$

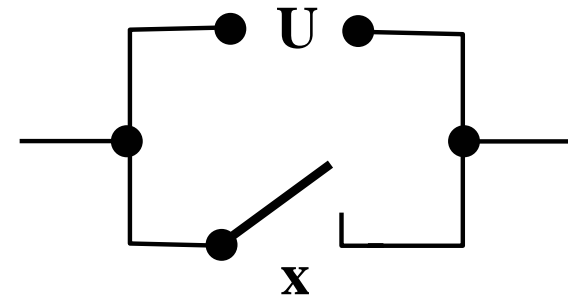
Schaltalgebra

H3: Neutrale Elemente



$$x \text{ ser } V = x$$

V: Dauernde **V**erbindung

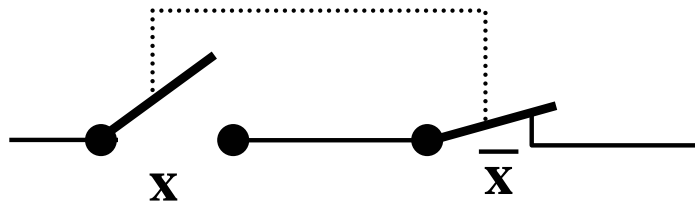


$$x \text{ par } U = x$$

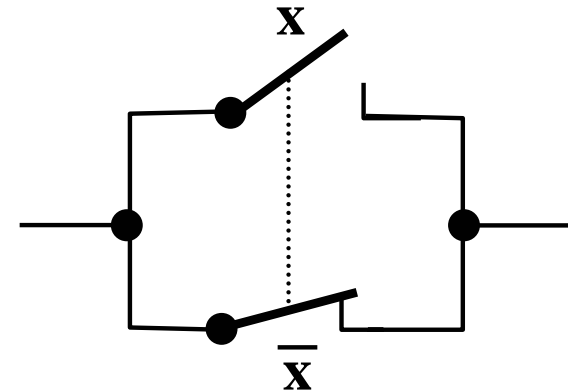
U: Dauernde **U**nterbrechung

Schaltalgebra

H4: Inverse Elemente



$$x \text{ ser } \bar{x} = U$$

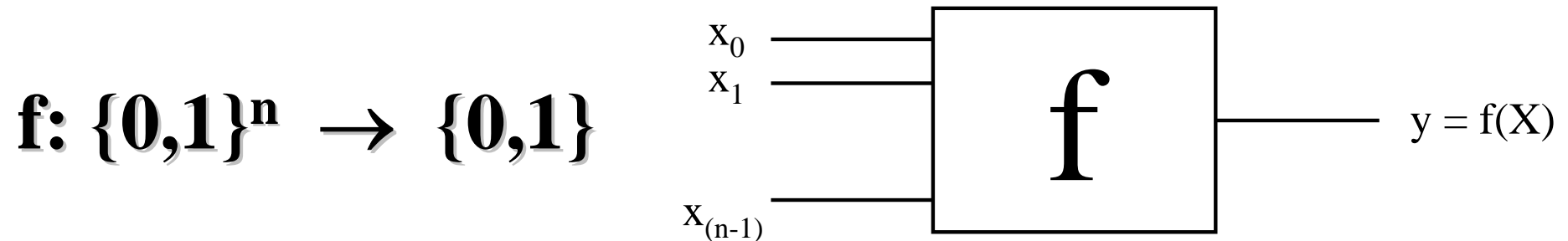


$$x \text{ par } \bar{x} = V$$

Voraussetzung: Ideale, gleichzeitiges Schalten

Boolesche Funktionen

Binäre Funktionen binärer Variablen



n unabhängige Eingangsvariablen: $x_i \in \{0,1\} \quad i = 0, \dots, n-1$

2^n spezielle Eingangsbelegungen: $B_i \in \{0,1\}^n \quad i = 0, \dots, 2^n-1$

1 abhängige Ausgangsvariable: $y \in \{0,1\}$

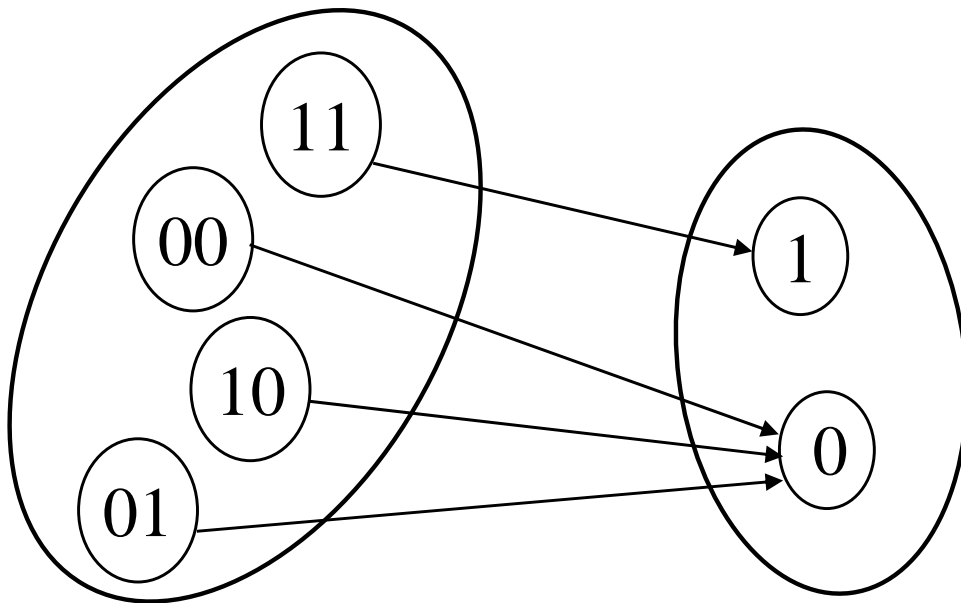
n Eingangsvariablen $\Rightarrow 2^{2^n}$ Funktionen

Beispiel: Konjunktion (UND)

$$y = f(x_1, x_0) = x_1 \wedge x_0$$

$n = 2 \Rightarrow 2^2 = 4$ mögliche Belegungen

Funktionsgraph



Funktionstabelle

x_1	x_0	y
0	0	0
0	1	0
1	0	0
1	1	1

Schaltalgebra

Die Verknüpfungen können leicht in Funktionstabellen dargestellt werden:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

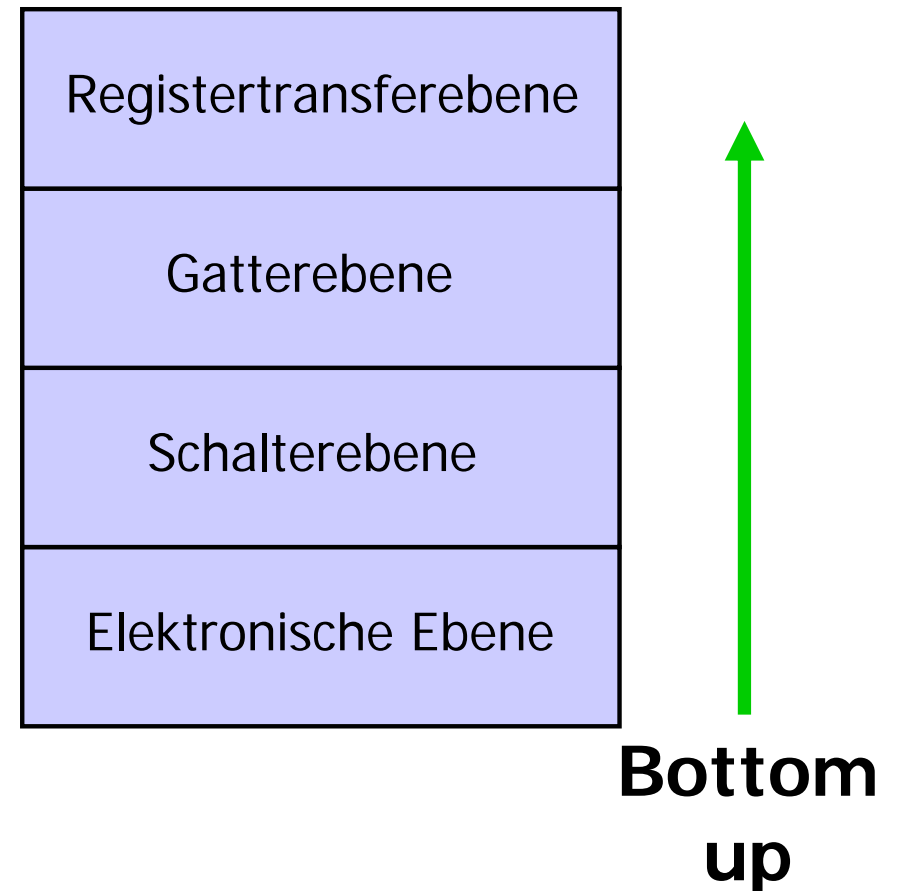
a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

a	\bar{a}
0	1
1	0

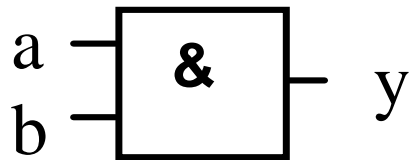
Realisierung von Schaltnetzen

Hierarchie:

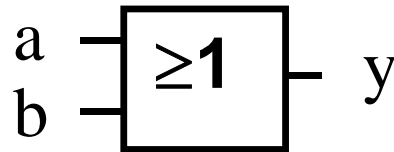
- Register-Transfer-Ebene:
logische Bausteine als
Grundelemente
- Gatterebene: logische Gatter:
UND, ODER, NAND,...
- Schalterebene: Transistoren
als Schalter,...
- Layoutebene:
Transistortechnologie



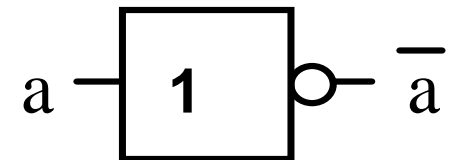
Schaltsymbole (DIN 40900 Teil 12)



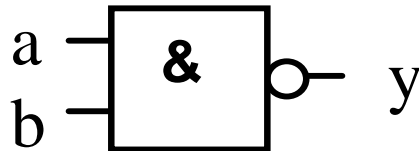
UND-Verknüpfung



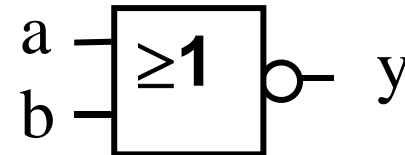
ODER-Verknüpfung



Negation



NAND-Verknüpfung

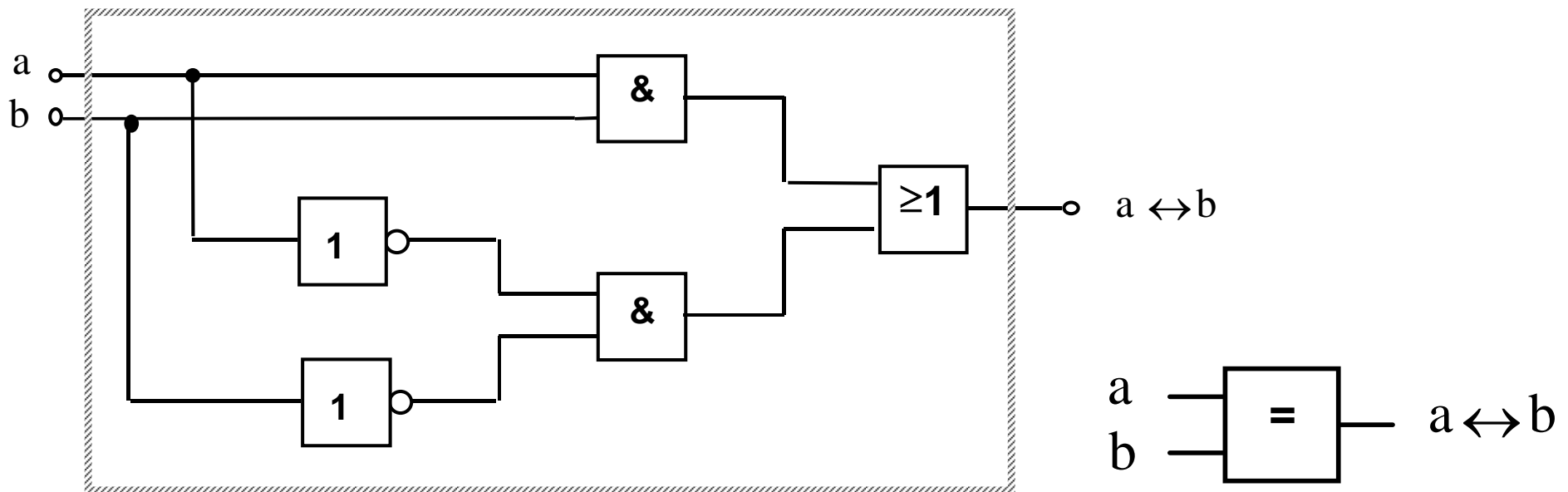


NOR-Verknüpfung

Verknüpfungsbausteine dieser Art werden **Gatter** genannt.

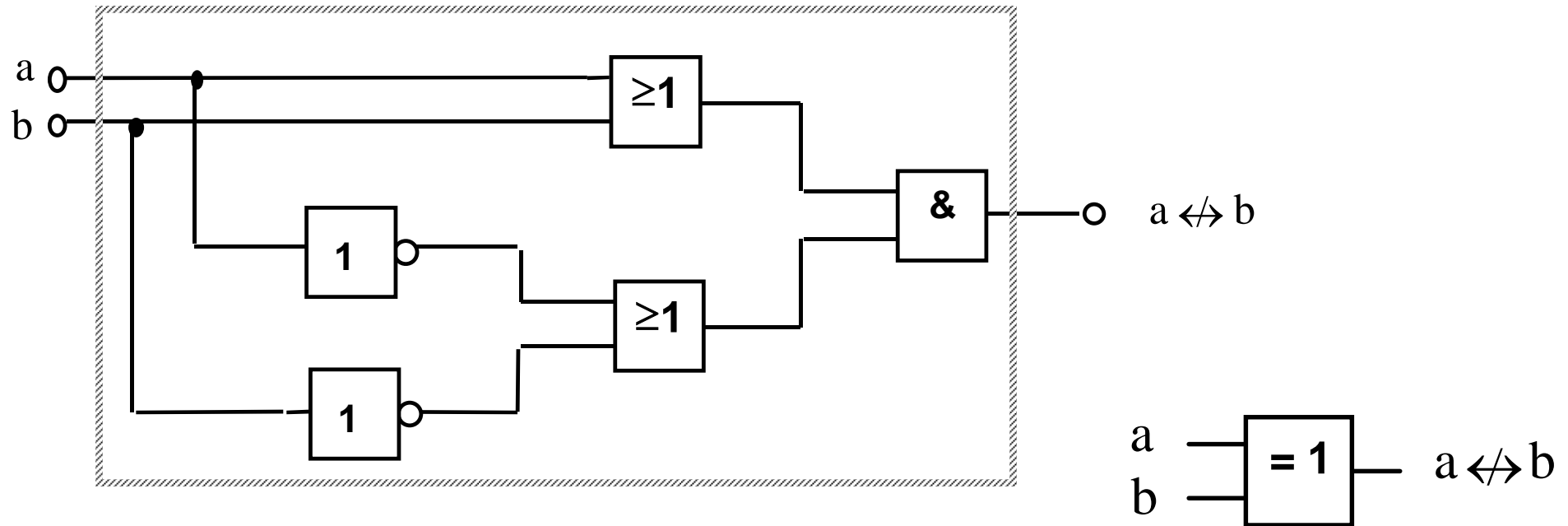
Äquivalenz-Gatter

Aus einfacheren Gattern lassen sich hierarchisch komplexere Gatter aufbauen, die teilweise eigene Symbole besitzen.



Äquivalenz als Komposition einfacherer Schaltglieder.

Antivalenz-Gatter



Antivalenz als Komposition einfacherer Schaltglieder.

Spezielle Strukturen

Anstelle aus logischen Gattern (UND, ODER, NICHT, NAND, NOR, ... usw.) lassen sich Schaltnetze auch mit komplexeren Standardbausteinen realisieren.

➡ Einfachere und oft flexiblere Realisierung

- Minimierung bedeutet hierbei dann nicht die Reduktion von Gattern.
- Es muss vielmehr die Anzahl und Größe komplexer Bausteine reduziert werden.

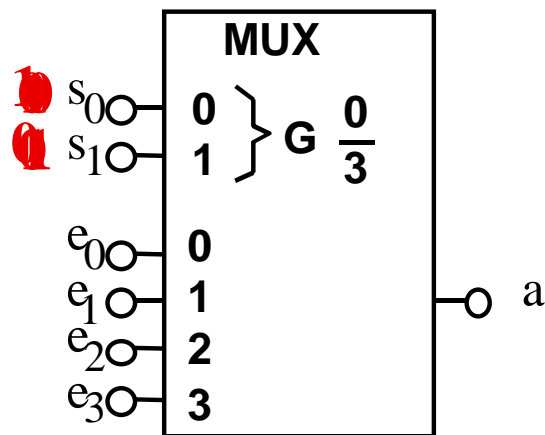
Multiplexer

Ein **Multiplexer** (Abk.: **MUX**) ist ein Baustein mit mehreren Eingängen und einem Ausgang.

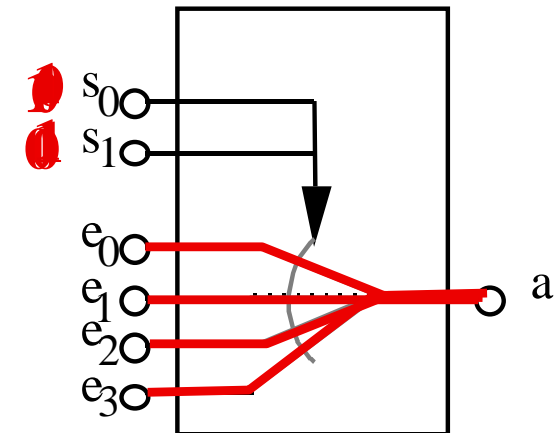
Über n Steuerleitungen wird einer der 2^n Eingänge auf den Ausgang geschaltet.

Multiplexer werden nach ihrer Größe als **2^n : 1 - Multiplexer** (alternativ als **1-aus- 2^n - Multiplexer**) klassifiziert.

Schaltbild und logisches Verhalten eines 1-aus-4-Multiplexers



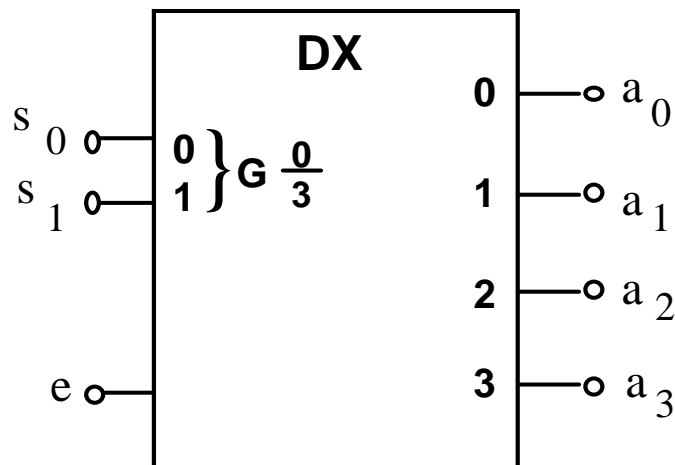
s_1	s_0	a
0	0	e_0
0	1	e_1
1	0	e_2
1	1	e_3



Demultiplexer / Dekoder

Der zum Multiplexer korrespondierende Baustein, der einen Eingang abhängig von **n** Steuerleitungen auf einen von **2ⁿ** Ausgängen schaltet, heißt **Demultiplexer**.

Beispiel:



s_1	s_0	a_0	a_1	a_2	a_3
0	0	e	0	0	0
0	1	0	e	0	0
1	0	0	0	e	0
1	1	0	0	0	e

Schaltbild und logisches Verhalten eines 1-auf-4-Demultiplexers

Realisierung mittels Speicherbausteinen

Bei den bisher behandelten Bausteinen (Gatter, Multiplexer, Dekoder) war die Funktion fest vorgegeben.

➡ festverdrahtete Logik

Höherintegrierte Verknüpfungsbausteine müssen die Flexibilität bieten, an viele verschiedene Anwendungen anpassbar zu sein. Diese Anpassung wird als **Personalisierung** oder als **Programmierung** bezeichnet.

➡ mikroprogrammierte Logik

Speichertypen

Je nachdem wie die Personalisierung des Speicherbausteins vonstatten geht, unterscheidet man verschiedene Speichertypen.

- **ROM (Read Only Memory)**
- **PROM (Programmable Read Only Memory)**
- **EPROM (Erasable Programmable Read Only Memory)**
- **RAM (Random Access Memory):**
 - Dynamische RAM-Bausteine (DRAM)
 - Statische RAM-Bausteine (SRAM)

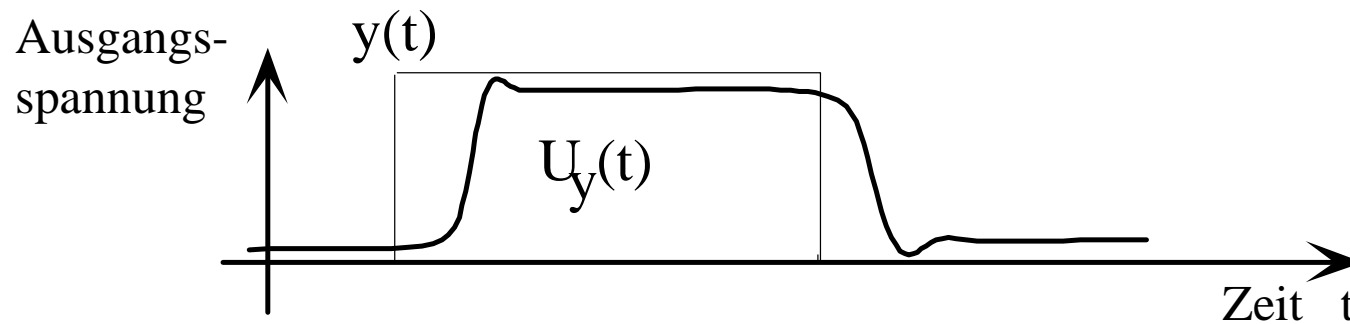
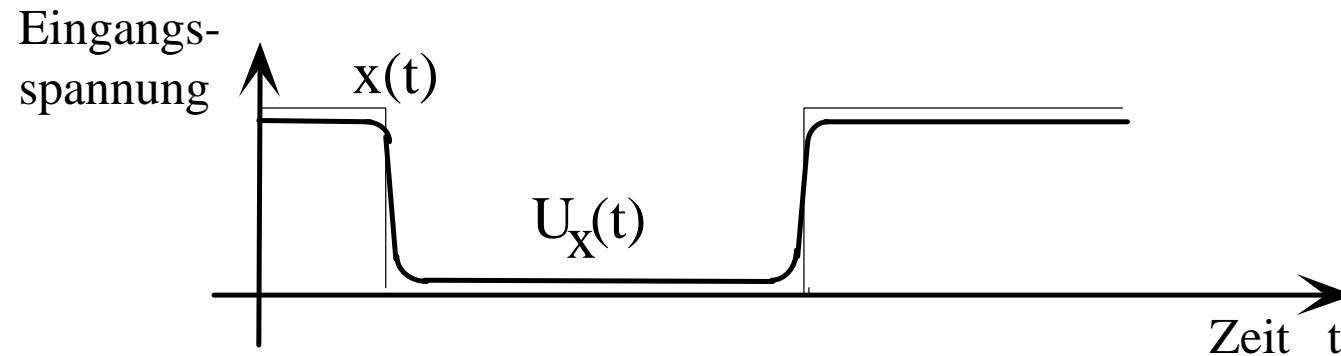
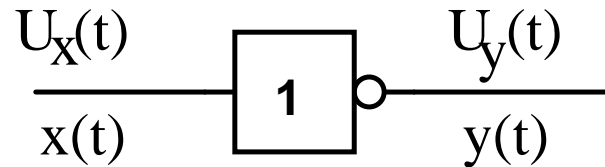
2.4 Laufzeiteffekte

Auf der Gatterebene wurden die Gatter bisher als ideale logische Verknüpfungen betrachtet.

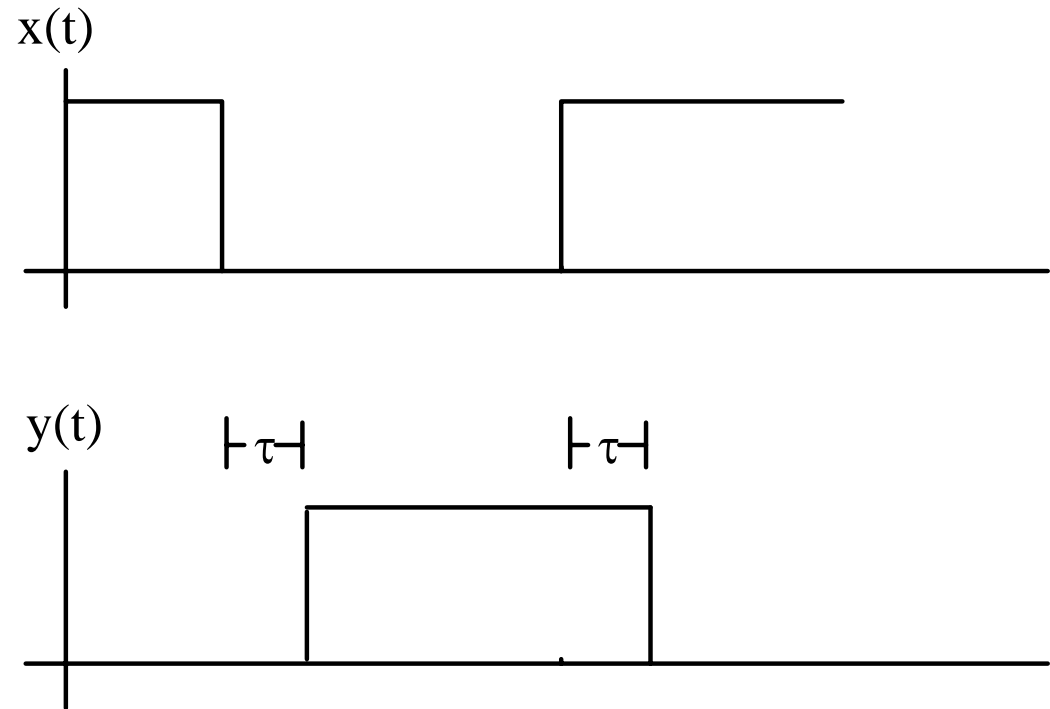
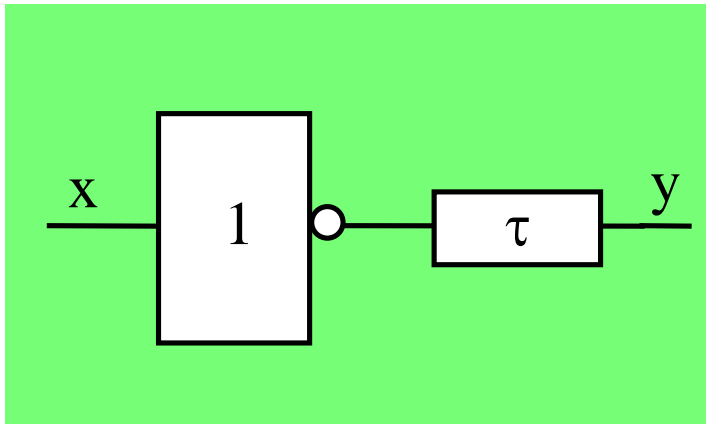
In der Realität werden Gatter jedoch mittels Transistoren, Widerstände, Kapazitäten, etc. realisiert (Layoutebene).

➡ **Der zeitliche Signal-Verlauf eines realen Gatters weicht vom Verlauf der idealen booleschen Größen ab.**

Realer und idealer Signalverlauf (Inverter)



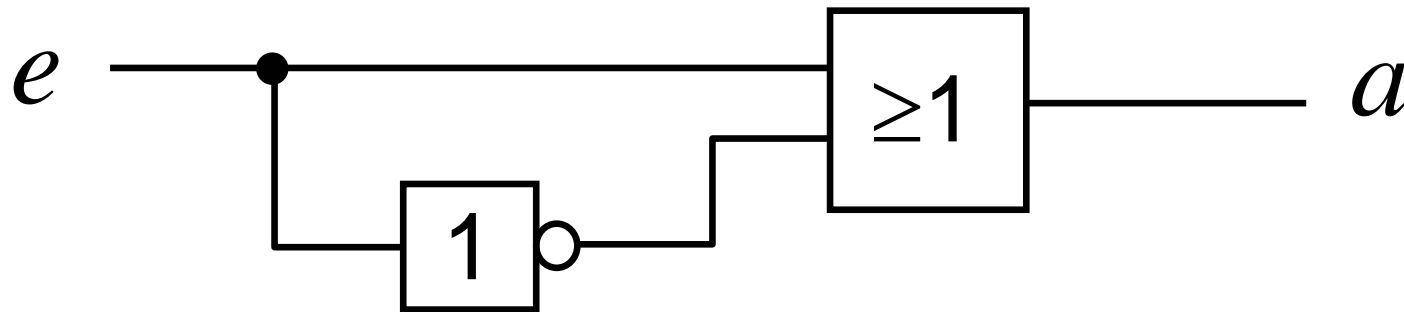
Beispiel: Totzeitmodell eines Inverters



Mit Hilfe dieses einfachen Modells lassen sich Laufzeiteffekte bereits sehr gut modellieren.

Beispiel: Inverteranwendung

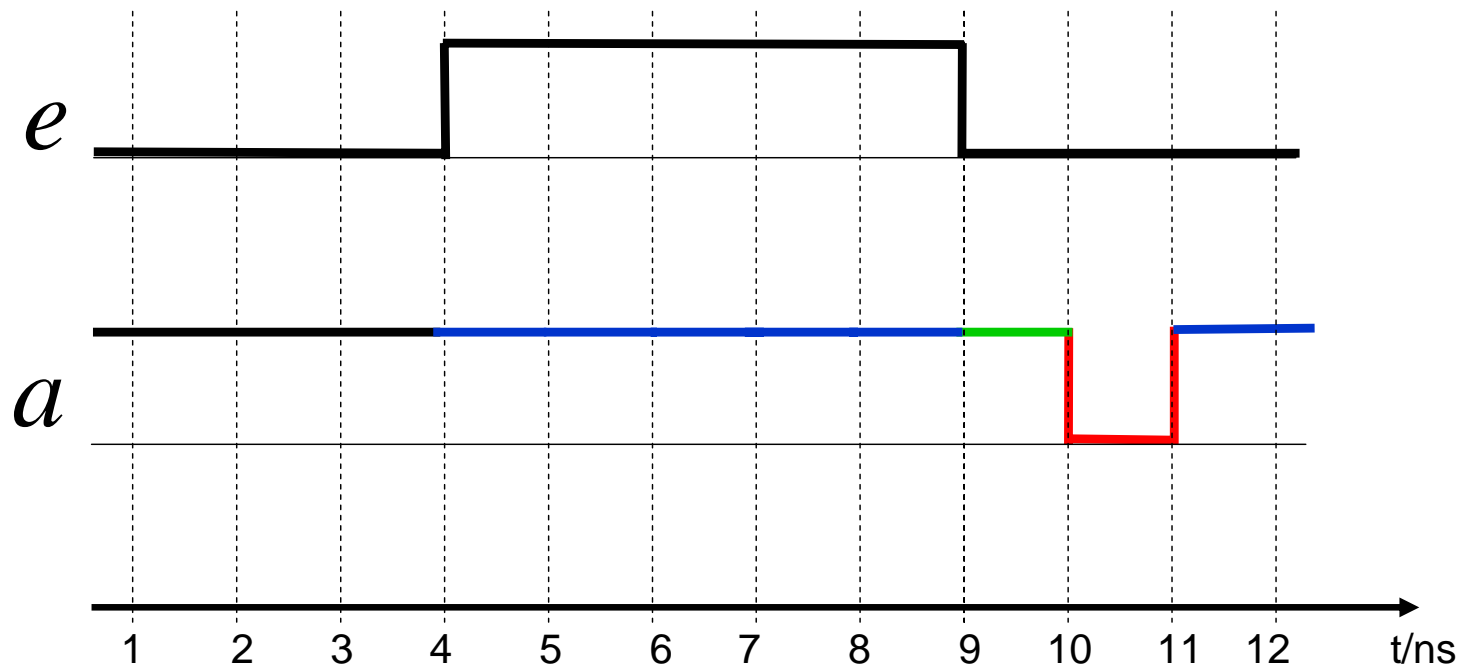
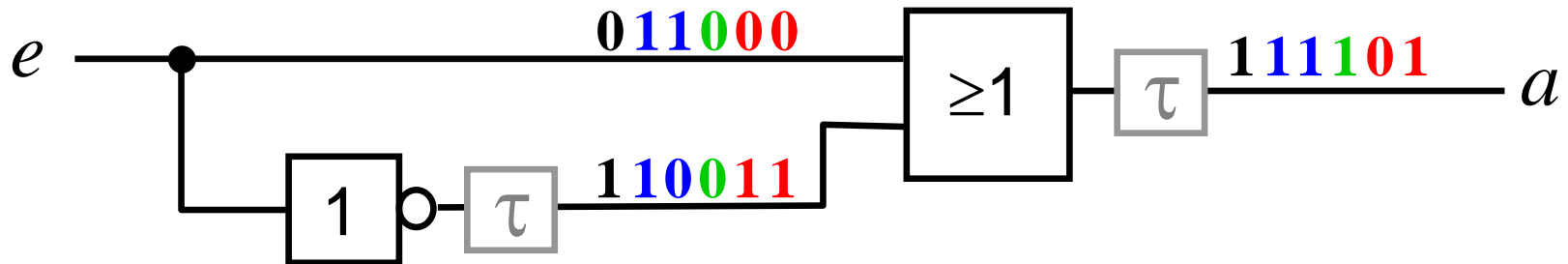
Gegeben:



$$a = e \vee \bar{e} = 1$$

Beide Gatter haben eine Verzögerungszeit von **1 ns**

Zeit-Diagramm



3. Schaltwerke

Schaltnetze (kombinatorische Schaltungen):

Die Ausgabe hängt lediglich von den Werten der Eingangsvariablen zum gleichen Zeitpunkt ab.

Schaltwerke (sequentielle Schaltungen):

Die Ausgabewerte hängen auch von Belegungen der Eingangsvariablen zu vergangenen Zeitpunkten ab.

Definition eines Automaten

Ein 6-Tupel $A = (E, A, Z, \delta, \omega, z_0)$ heißt **Automat**, wenn

□ **E, A und Z nichtleere Mengen** sind

- E ist die Menge der **Eingangsbelegungen** e ,
- A die Menge der **Ausgangsbelegungen** a und
- Z die Menge der **Zustände** z .

□ **Überföhrungsfunktion** $\delta: Z \times E \rightarrow Z$

δ ist eine auf der Menge $Z \times E$ definierte Funktion, deren Werte in Z liegen.

□ **Ausgabefunktion** $\omega: Z \times E \rightarrow A$

ω eine auf der Menge $Z \times E$ definierte Funktion, deren Werte in A liegen.

□ **Grundzustand** z_0 .

Einführung in die Automatentheorie

Die Zustandsmenge Z ermöglicht die **Speicherung** von Wissen über Eingangsbelegungen der Vergangenheit.

Die aktuelle Ausgabebelegung wird durch die Funktion ω , der neue Zustand durch die Funktion δ aus den **aktuellen Eingangsbelegungen und dem alten Zustand** erzeugt.

Realisierung von Automaten

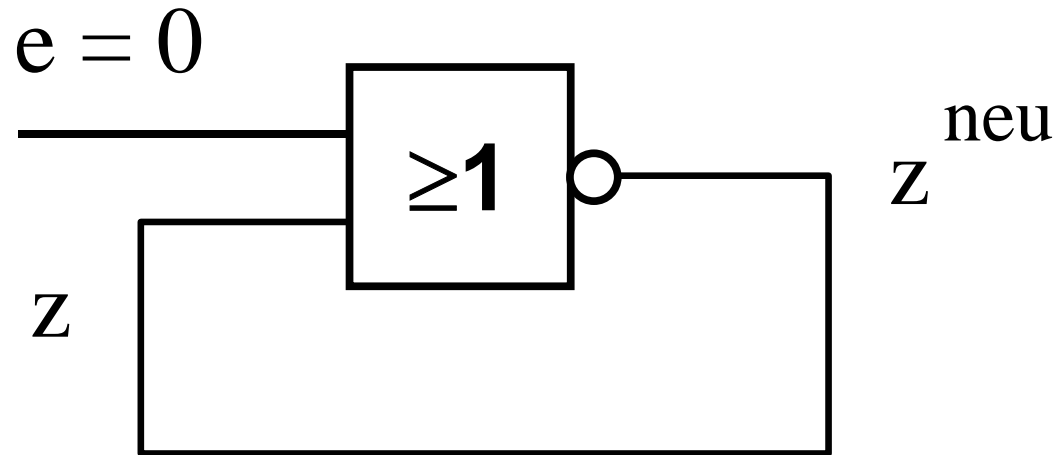
Zur Speicherung vergangener Informationen ist ein **Zustandsspeicher** erforderlich.

Einfachste Form dieses Zustandsspeichers:

➡ **Rückkopplung**

Durch die Rückkopplung lassen sich in den Eingangsvariablen nicht mehr repräsentierte Informationen wieder am Eingang zur Verfügung stellen.

Beispiel: Rückgekoppeltes NOR-Gatter



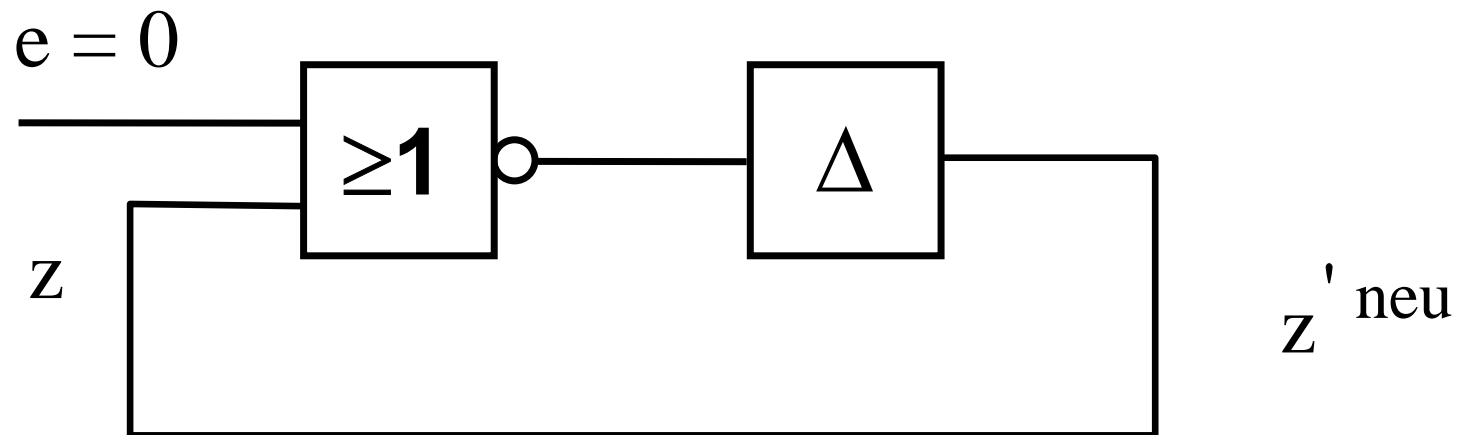
Als ideales Gatter betrachtet ist die Schaltung unzulässig, denn es müsste gleichzeitig gelten:

$$z^{\text{neu}} = \overline{z} \quad \text{und}$$

$$z = z^{\text{neu}}$$

Rückgekoppeltes NOR-Gatter

In der Realität hat jede Schaltung eine Verzögerungszeit größer 0 (Totzeitmodell).



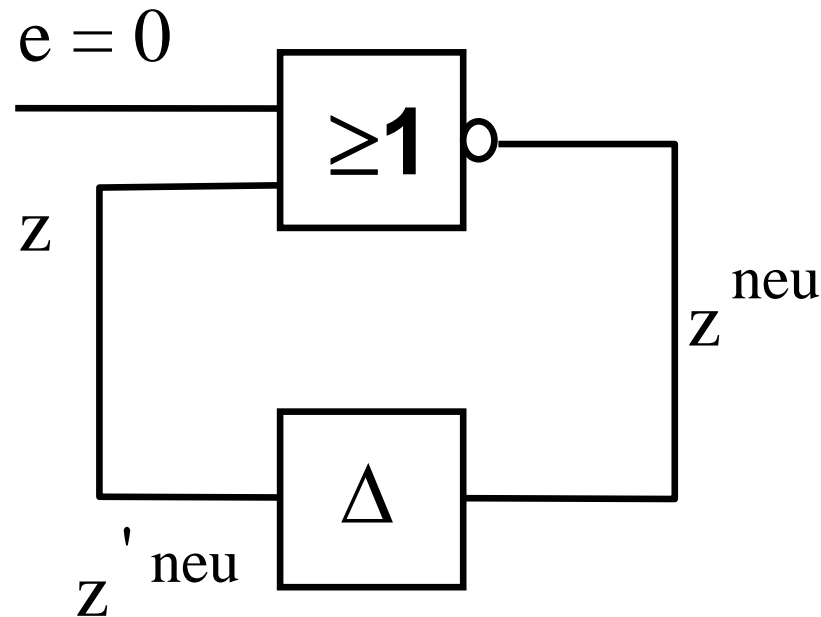
Mit dieser Verzögerung erhält man:

$$z'_{\text{neu}}(t+\Delta) = \bar{z}(t)$$

$$z(t+\Delta) = z'_{\text{neu}}(t+\Delta)$$

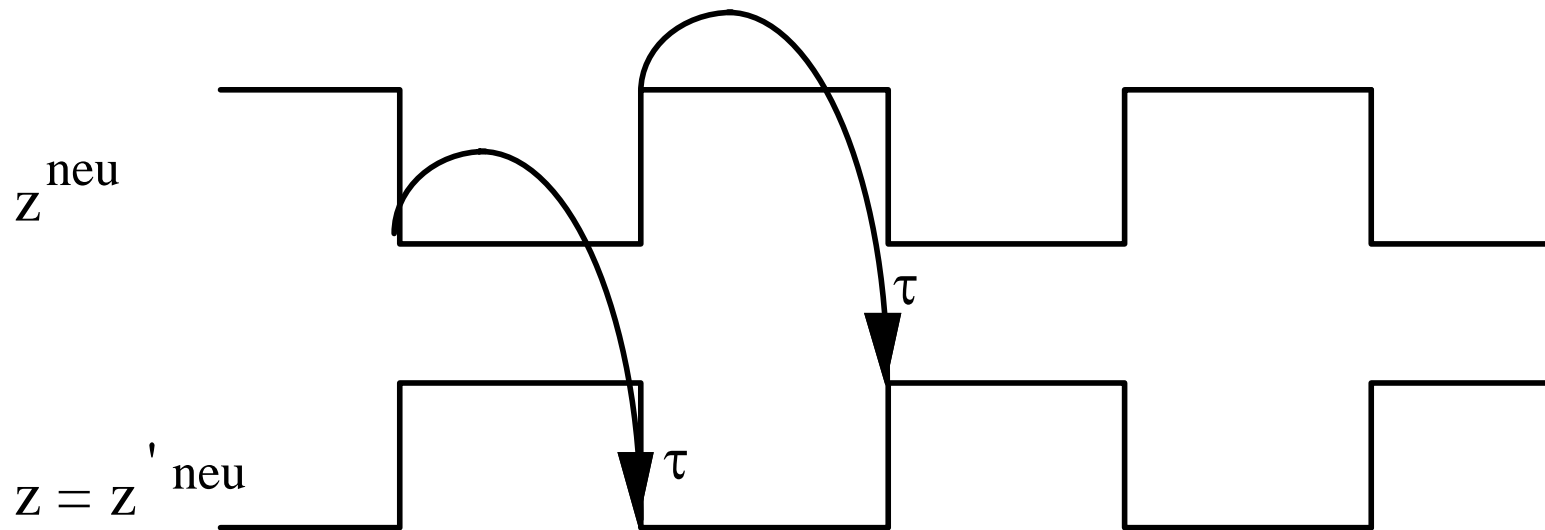
Rückgekoppeltes NOR-Gatter

Zeichnet man die Schaltung etwas anders, so sieht man, dass das **Δ -Verzögerungsglied** dem **Speicher** entspricht.



Rückgekoppeltes NOR-Gatter

Das Zeitverhalten der Schaltung im Zeitdiagramm:



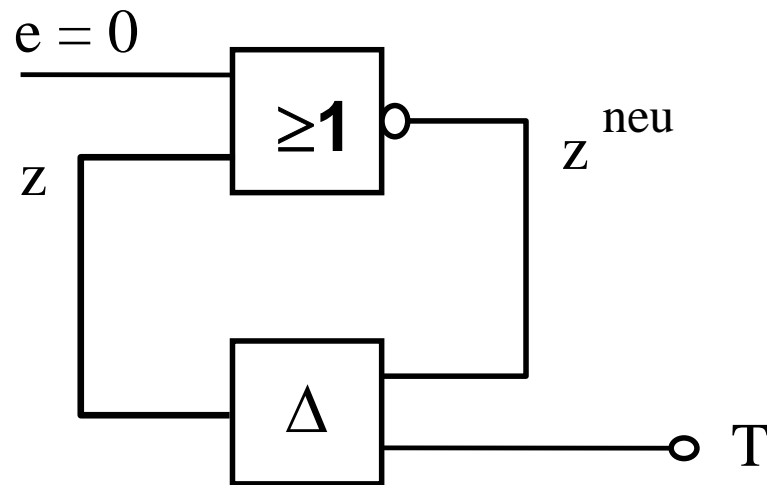
Das Verhalten der Schaltung ist stark abhängig von den Verzögerungszeiten.

Rückgekoppeltes NOR-Gatter

Von den Verzögerungszeiten unabhängiges Verhalten:

- ➡ Der Speicher wird so aufgebaut, dass z seinen alten Wert so lange beibehält, bis z explizit durch ein externes Signal T auf z^{neu} gesetzt wird.

Beispiel:



z behält seinen Wert solange, bis T von **0** auf **1** wechselt.

Dann wird z auf z^{neu} gesetzt und behält diesen Wert bis zum nächsten **0-1** Wechsel von T .

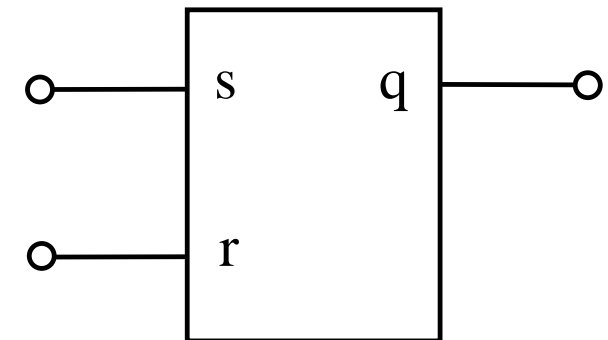
Definitionen

- Werden alle Zustandsspeicher von einem oder mehreren zentralen Synchronisationssignal(en) **T** (Takt) gesteuert, so spricht man von einem **synchronen Schaltwerk**.
- Anderenfalls spricht man von einem **asynchronen Schaltwerk**.
- Die Synchronisation über einen Takt kann **flankengesteuert** und **pegelgesteuert** sein.

Asynchrones RS-Flipflop

Einfacher Zustandsspeicher mit folgendem Verhalten:

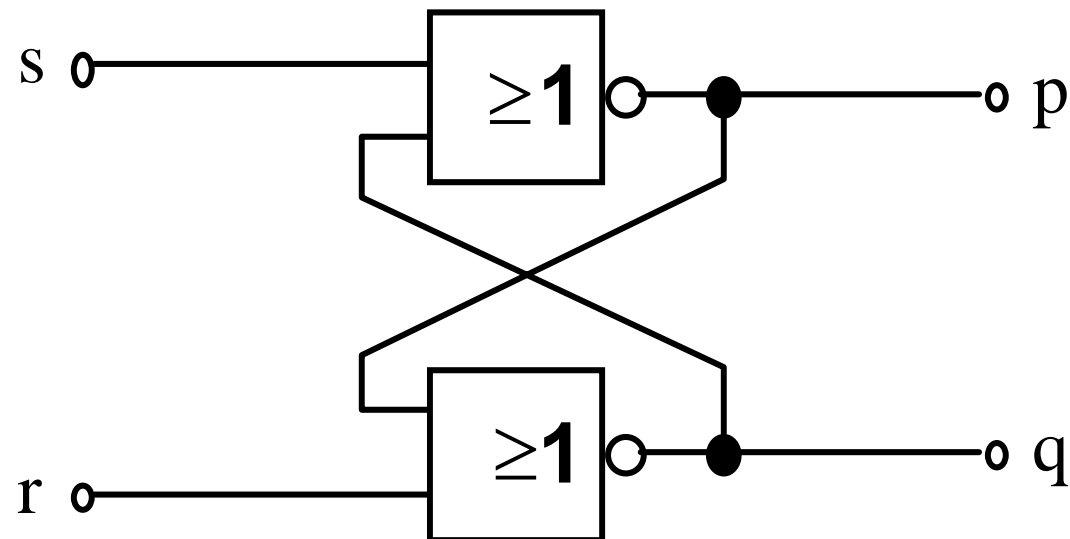
r	s	Funktion	
0	0	Speichern	$q^{t+1} = q^t$
0	1	Setzen	$q^{t+1} = 1$
1	0	Rücksetzen	$q^{t+1} = 0$



Asynchrones RS-Flipflop

Dieser Speicher ist ein Standardelement. Es wird als **asynchrones RS-Flipflop** bezeichnet.

Schaltbild aus NOR-Gatter:



RS-Flipflop

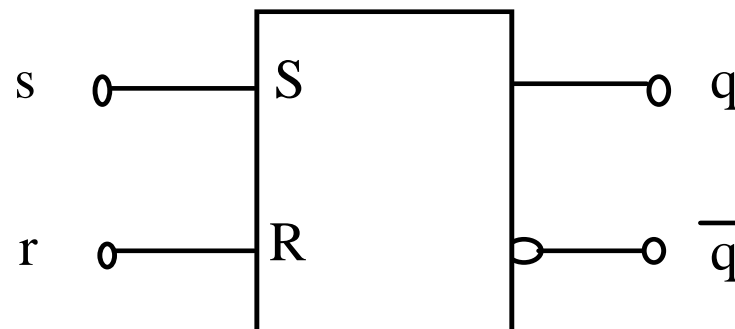
Dieses Flipflop haben wir bereits kennengelernt.

Es ist das einfachste aller Flipflops.

Sind r und s nicht gleichzeitig 1 (**verboten !**), so sind die Ausgänge p und q komplementär

→ **Die Zustandsvariable q und ihre Negation \bar{q} stehen am Ausgang zur Verfügung**

Schaltsymbol des asynchronen RS-Flipflops:

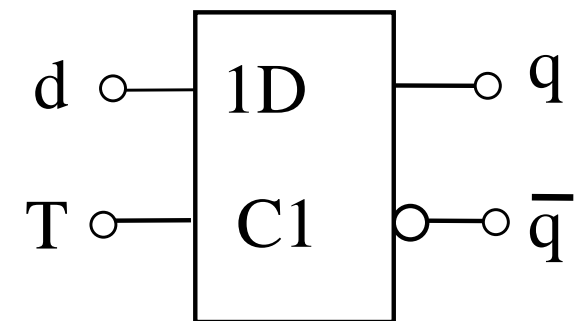
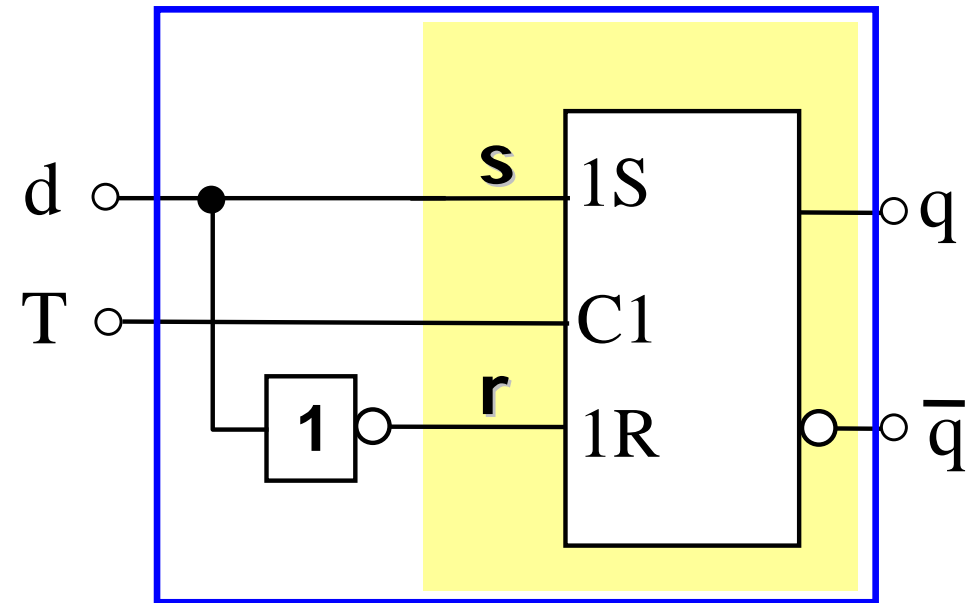


D-Flipflop

Bei einem RS-Flipflop ist stets die Nebenbedingung $(r \wedge s = 0)$ zu beachten.

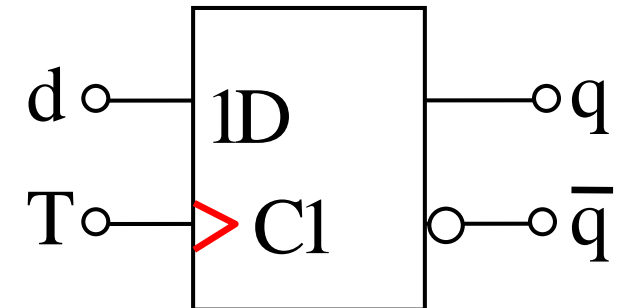
Führt man eine Eingangsvariable **d** **bejaht** zum **s**-Eingang und **negiert** zum **r**-Eingang, dann ist diese Bedingung stets erfüllt.

Damit erhält man ein sog. **D-Latch**

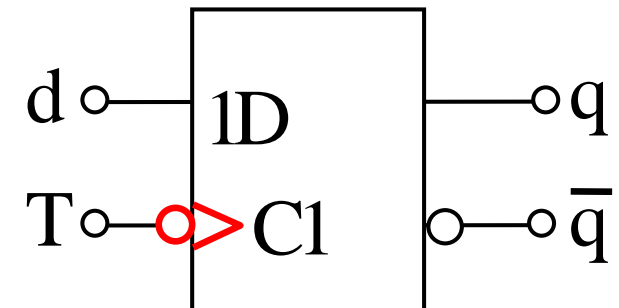


Anmerkungen

Im Schaltsymbol wird die **Taktflankensteuerung** durch ein Dreieck am Takteingang spezifiziert.



Bei einer Steuerung mit der **negativen Taktflanke** wird ein Negationszeichen vor das Dreieck gesetzt.



JK-Flipflop

Beim RS-Flipflop war die Eingangsvariablen-Kombination $r = s = 1$ verboten

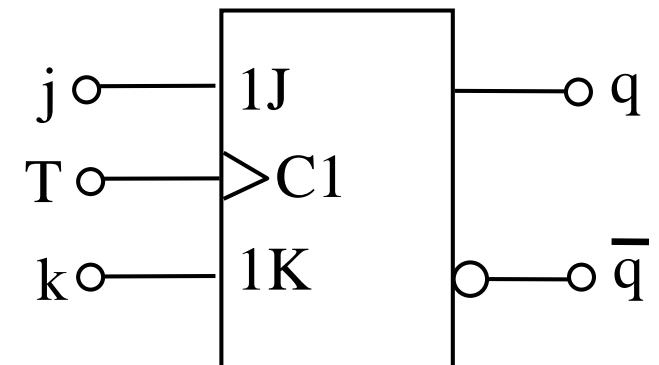
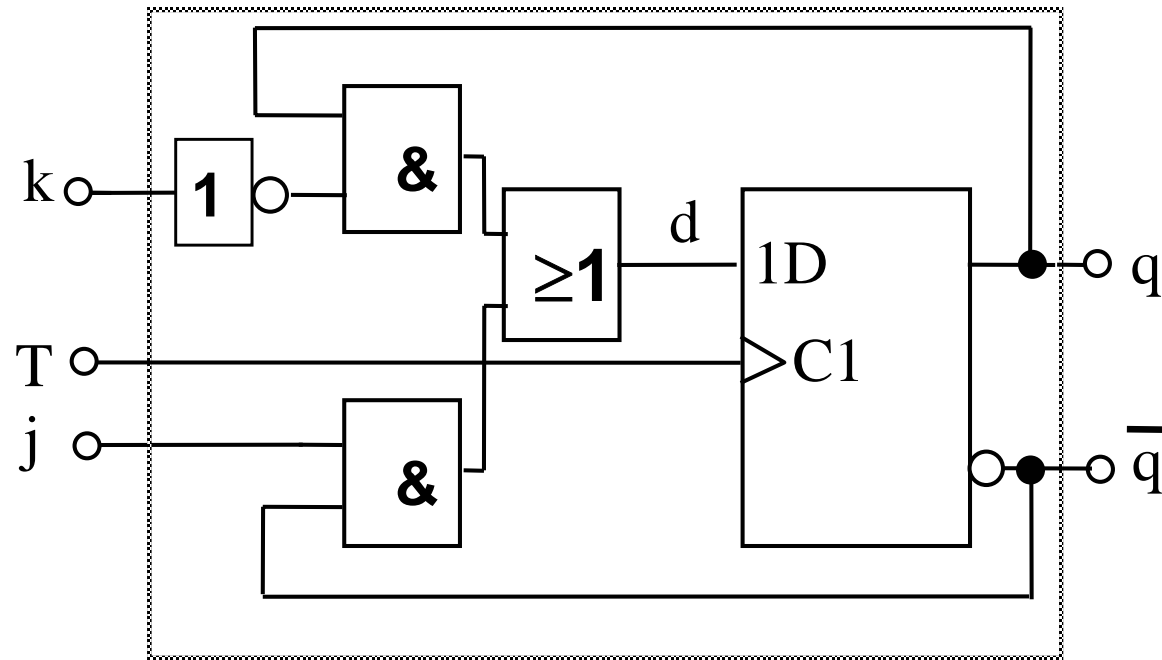
Ziel: Ein Flipflop entwerfen, welches diese Kombination nutzt.

Als vierte Funktion neben "speichern", "setzen" und "rücksetzen" soll bei Eingangskombination $r = s = 1$ der Flipflop-Inhalt komplementiert werden.

Bezeichnung: **j:** resultierender Setzeingang
 k: resultierender Rücksetzeingang
 ➔ **JK-Flipflop**

Dieses Verhalten lässt sich durch Zusatzbeschaltung schon bekannter Flipflops erreichen.

Schaltbild des synchronen JK-Flipflops



Funktionstabelle des JK-Flipflops

Verkürzte Funktionstabelle des JK-Flipflops:

j	k	q^{t+1}	Funktion
0	0	q^t	speichern
0	1	0	rücksetzen
1	0	1	setzen
1	1	$\overline{q^t}$	wechseln

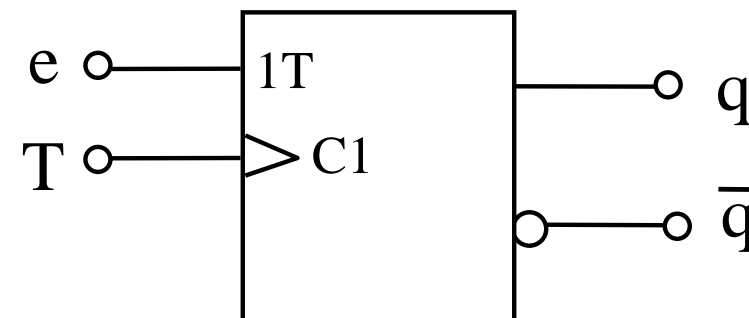
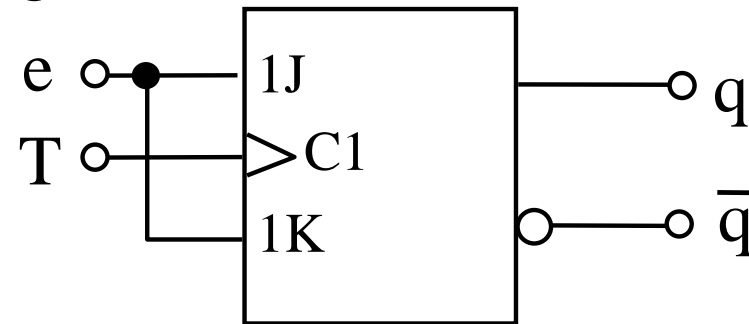
Das T-Flipflop

Ein **T-Flipflop** (*"to toggle", kippen*) hat nur einen Eingang.

Liegt an diesem Eingang eine "1", kippt das Flipflop mit jedem Taktimpuls in einen anderen Zustand, hat die Eingangsvariable den Wert "0", behält das Flipflop seinen alten Zustand bei.

T-Flipflop aus JK-Flipflop

Durch geeignete Eingangsbeschaltung eines JK-Flipflops lässt sich leicht das Verhalten eines T-Flipflops erzeugen.



Synchrones T-Flipflop

T-Flipflop: Verkürzte Funktionstabelle

Verkürzte Funktionstabelle des T-Flipflops

e	q^{t+1}	Funktion
0	q^t	speichern
1	$\overline{q^t}$	wechseln

Ein synchrones Setzen oder Rücksetzen des T-Flipflops ist mit dem Eingang e nicht möglich.

Um das Flipflop in einen definierten Grundzustand zu bringen, ist daher ein zusätzlicher Setz- oder Rücksetzeingang notwendig.

Spezielle Schaltwerkbausteine

Ähnlich wie bei den kombinatorischen Schaltungen (Schaltnetze) lassen sich auch bei sequentiellen Schaltungen (Schaltwerken) komplexe Bausteine aus elementaren Elementen zusammensetzen.

Register

Lineare Anordnung von Flipflops zur Speicherung mehrerer Bits (Bitvektors).

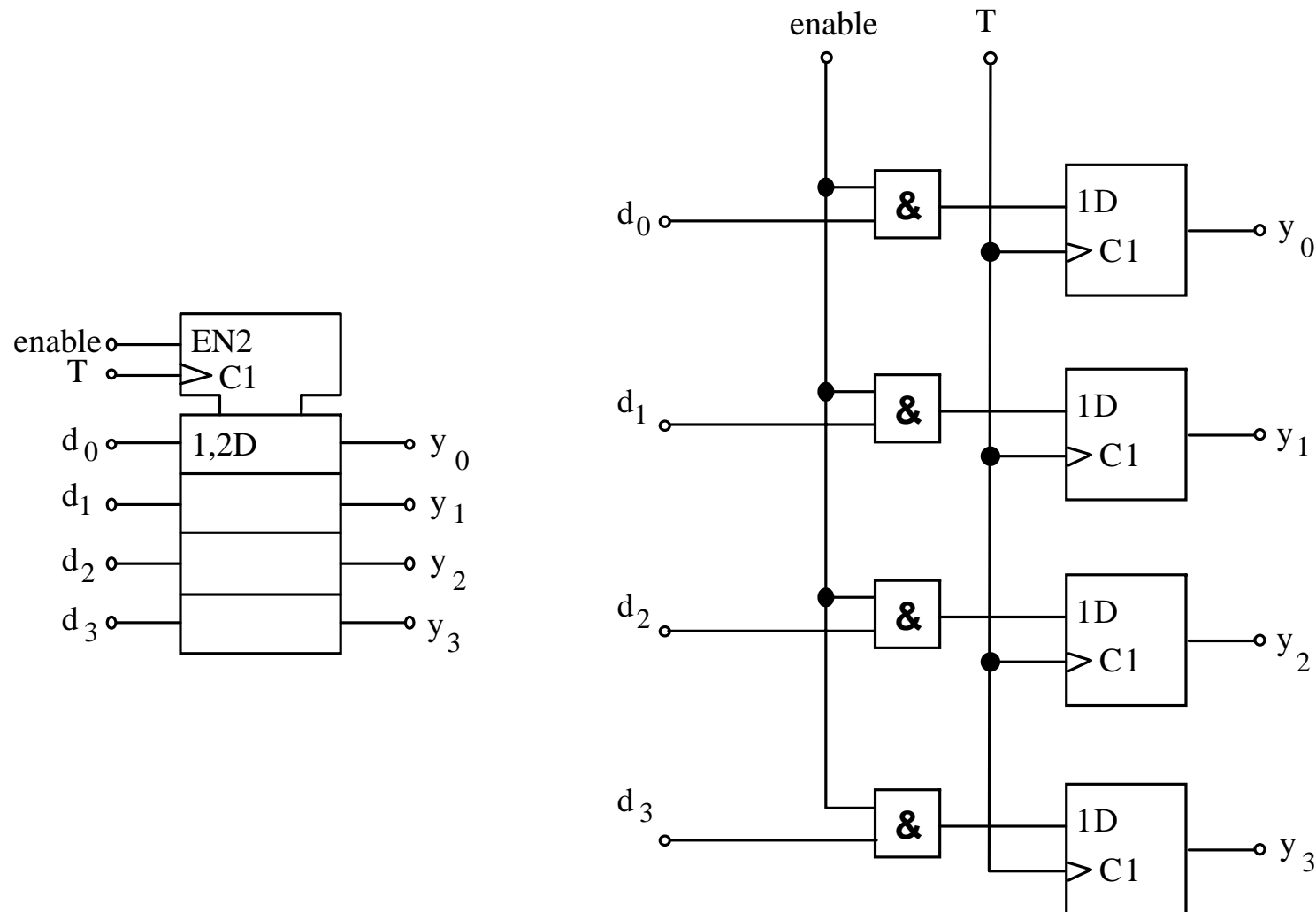
Die Flipflops werden mit einem gemeinsamen Takt angesteuert.

Einfachstes Register:

Unverkoppelt nebeneinandergesetzte D-Flipflops.

Im allgemeinen werden die Flipflops durch zusätzliche gemeinsame Steuersignale beeinflusst.

4-Bit-Registers aus D-Flipflops mit Freigabesignal

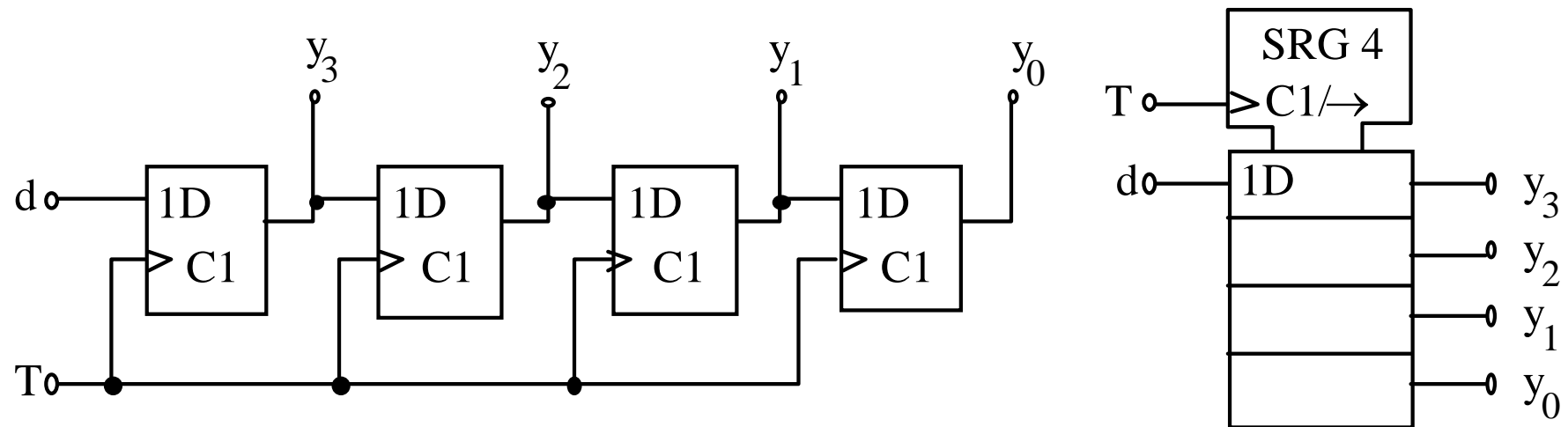


Nur wenn "enable" = 1 ist, werden die Daten übernommen.

Schieberegister

Kette von in Reihe geschalteten Registern oder D-Flipflops

Der Ausgang eines Speicherelements ist jeweils mit dem Eingang des nächsten verbunden.

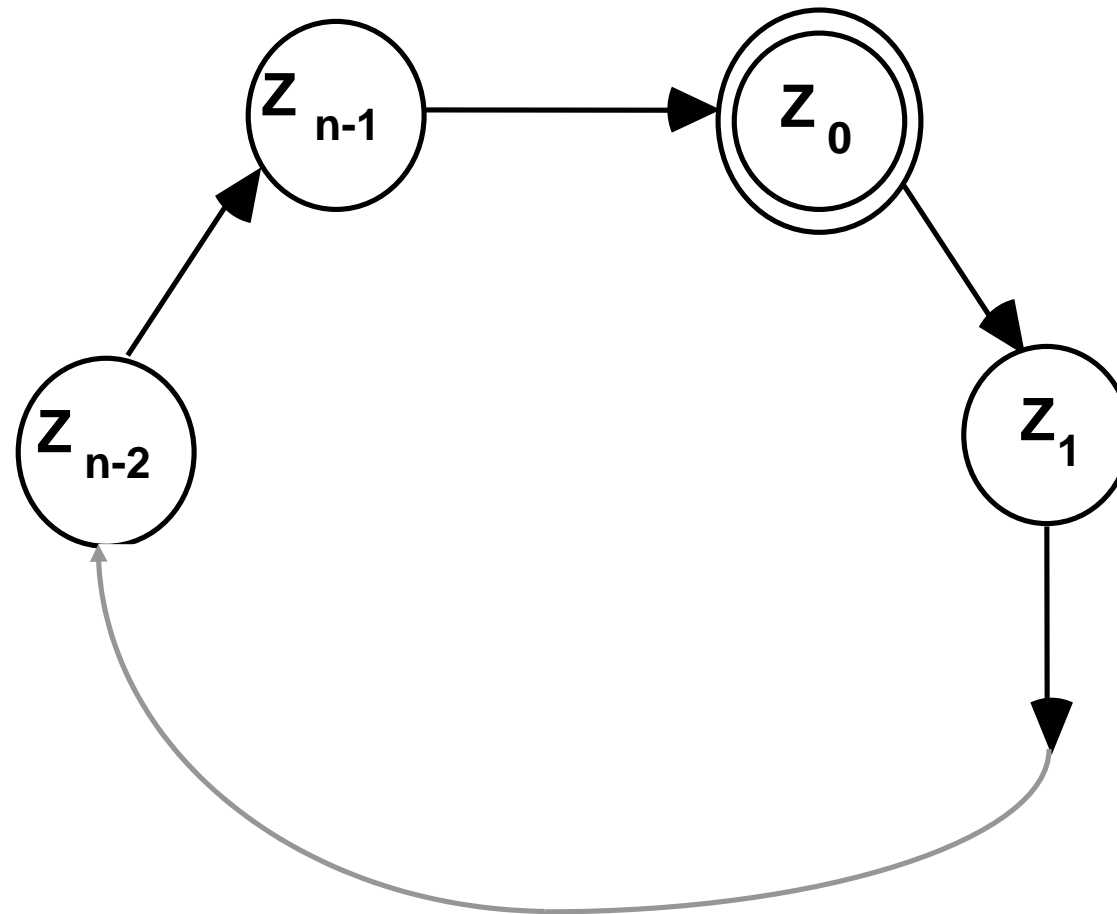


Zähler

Zähler erfüllen in digitalen Systemen mehrere Aufgaben:

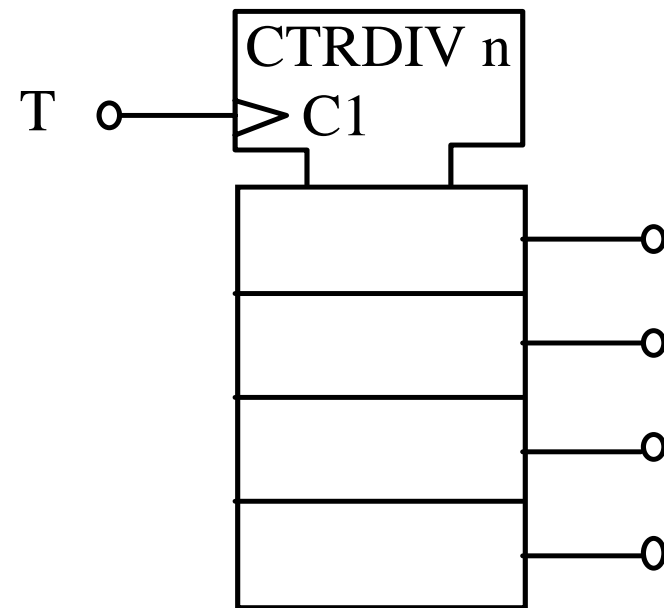
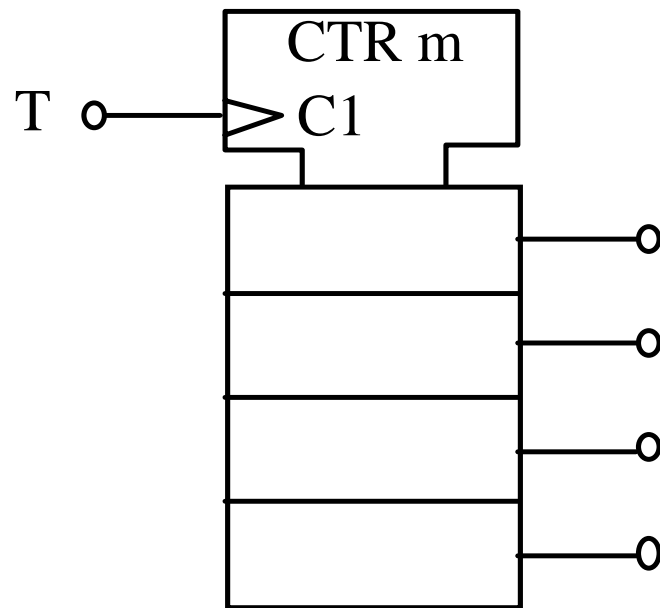
- Man kann Impulse abzählen.
- Man kann aufeinanderfolgende Adressen eines Speichers adressieren (z.B. bei Programmzählern) oder aufeinanderfolgende Arbeitsschritte kontrollieren (bei Steuerwerken).
- Eine vorgegebene Impulsfolge läßt sich in der Frequenz reduzieren, der Zähler wirkt als Frequenzteiler.
Dabei macht man sich die Tatsache zunutze, daß sich das Bit i einer Zahl $z_n \dots z_i \dots z_0$ nur 2^{-i} mal so oft ändert wie Bit 0, wenn diese Zahl fortlaufend inkrementiert wird.

Zähler



Grundlegendes Übergangsdiagramm

Schaltsymbole



Schaltsymbole für m-stellige und Modulo-n-Zähler

Programmierbare Bausteine

Ebenso wie für Schaltnetze können auch für Schaltwerke programmierbare Bausteine eingesetzt werden:

Beispiel:

die bereits erwähnten **maskenprogrammierbaren Gate-Arrays (MPGA)** lassen sich natürlich auch für Schaltwerke verwenden.

Rechnerarithmetik

- ❑ Verfahren und Implementierungsmöglichkeiten am Beispiel der vier Grundrechenarten
- ❑ Prinzipieller Aufbau einer arithmetisch-logischen Einheit, die mehrere Operationen integriert

Addition und Subtraktion

Schaltungen zur Addition von Festkomma-Dualzahlen:

Grundlage für die Durchführung aller arithmetischen Verknüpfungen

Denn:

- ❑ Subtraktion → Addition der negativen Zahl:

$$\mathbf{X - Y = X + (-Y)}$$

- ❑ Multiplikation und Division lassen sich ebenfalls auf die Addition zurückführen.

Addition und Subtraktion

Bei Gleitkommazahlen:

- ❑ Mantisse und Exponent werden separat verarbeitet.
- ❑ Hierbei bildet die Addition von Festkomma-Dualzahlen die Grundlage.

➔ Grundtypen von Addierern sind wichtig.

Volladdierer

Bei der Addition zweier Dualzahlen: Summe und Übertrag entstehen als Ergebnis

Funktionstabelle:

a	b	s	ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

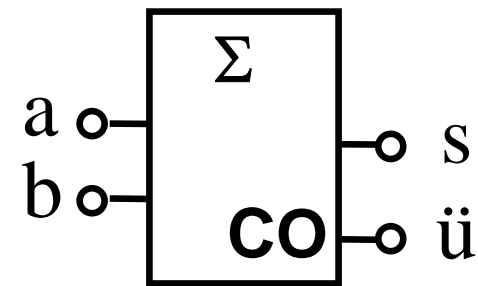
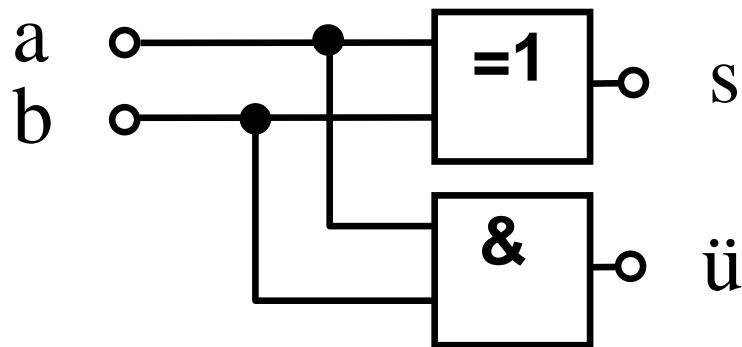
Man nennt dies einen **Halbaddierer**

Halbaddierer

Gleichungen: $s = a \bar{b} \vee \bar{a} b = a \oplus b$

$$\ddot{u} = a b$$

Das Schaltbild und das Schaltsymbol:



Schaltbild und Schaltsymbol eines 1-Bit-Halbaddierers

Mehrstellige Dualzahlen

Zusätzlicher Eingang für den Übertrag der vorhergehenden Stellen ist nötig.

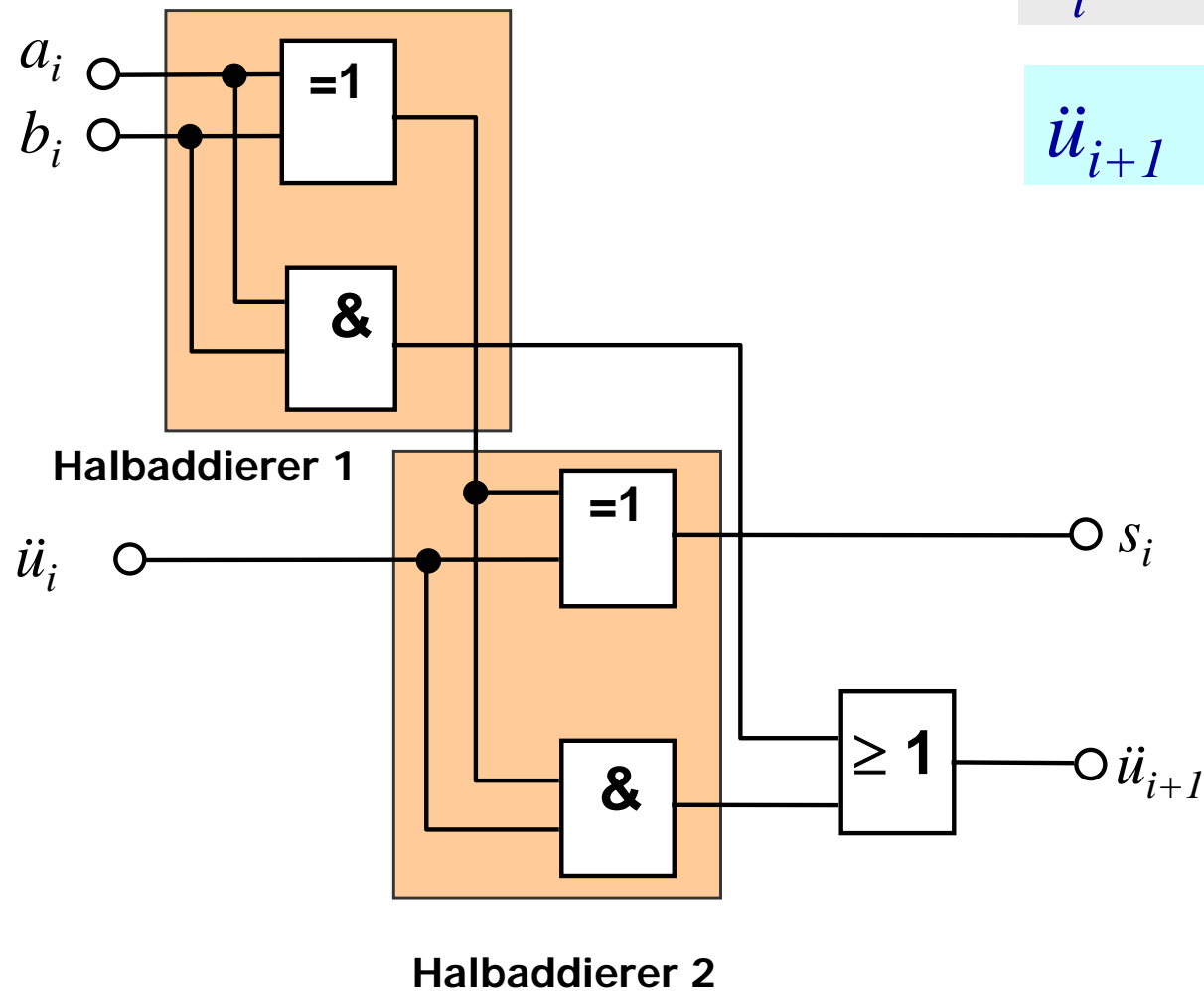
a_i	b_i	\ddot{u}_i	s_i	\ddot{u}_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Volladdierer

$$s_i = a_i \nleftrightarrow b_i \nleftrightarrow \ddot{u}_i$$

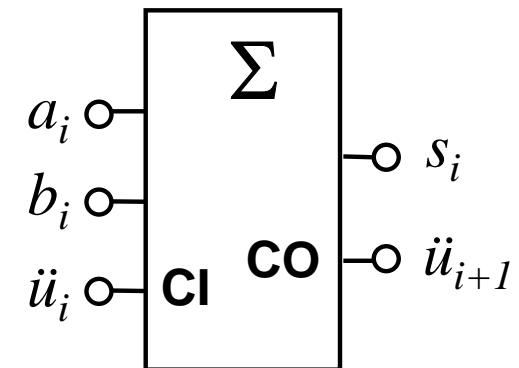
$$\begin{aligned}\ddot{u}_{i+1} &= a_i \ddot{u}_i \vee b_i \ddot{u}_i \vee a_i b_i \\ &= (a_i \nleftrightarrow b_i) \ddot{u}_i \vee a_i b_i\end{aligned}$$

Schaltnetz und Schaltsymbol

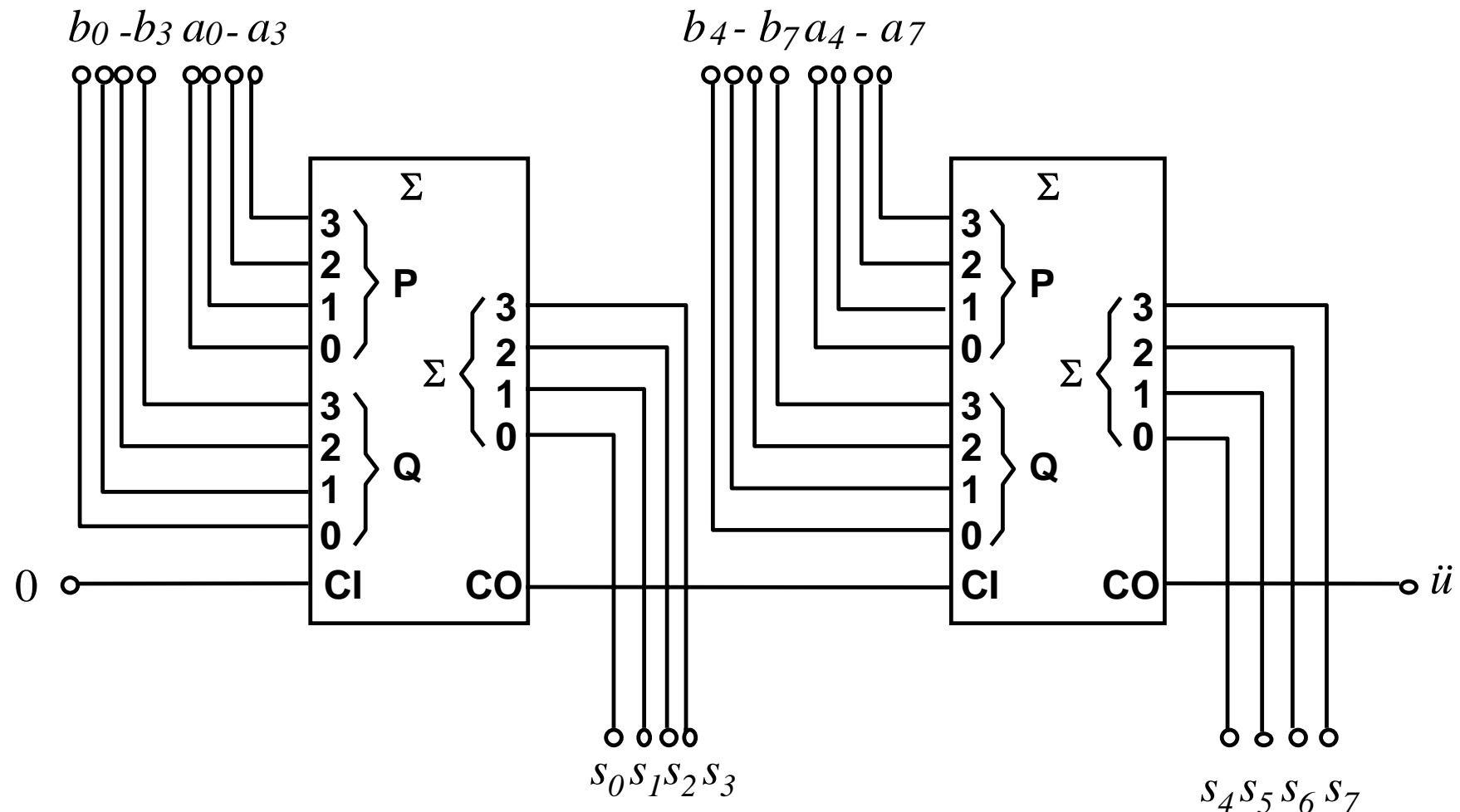


$$s_i = a_i \oplus b_i \oplus \ddot{u}_i$$

$$\ddot{u}_{i+1} = (a_i \oplus b_i) \ddot{u}_i \vee a_i b_i$$



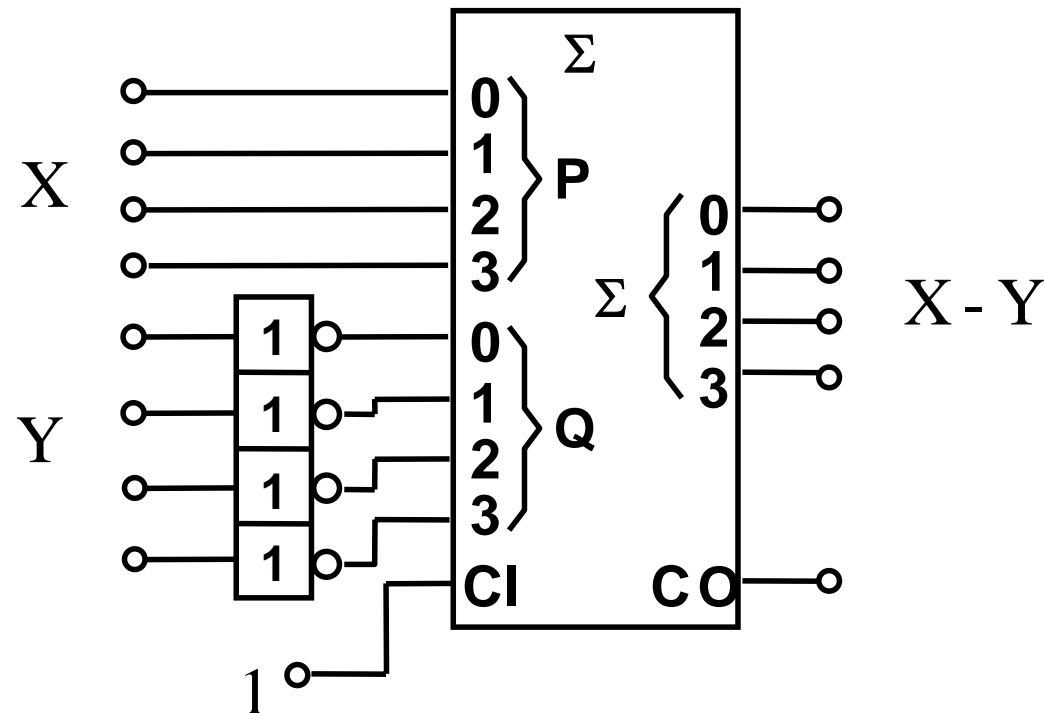
Carry-lookahead-Addierer



Kaskadierung zweier 4-Bit Carry-lookahead-Addierer

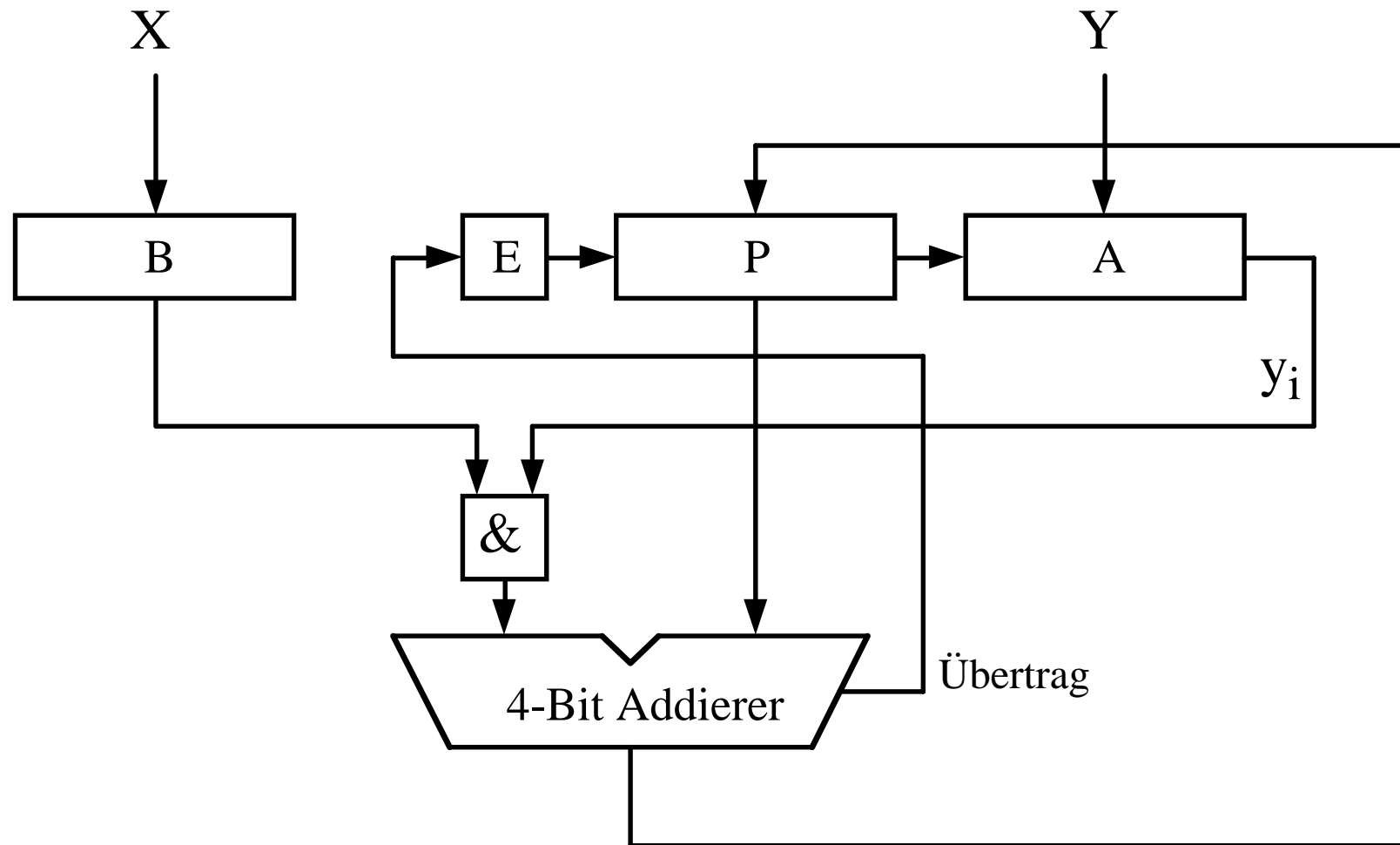
Subtraktion

Die beiden Additionen können mit einem Addierer vorgenommen werden, indem man den Übertragseingang ausnutzt.

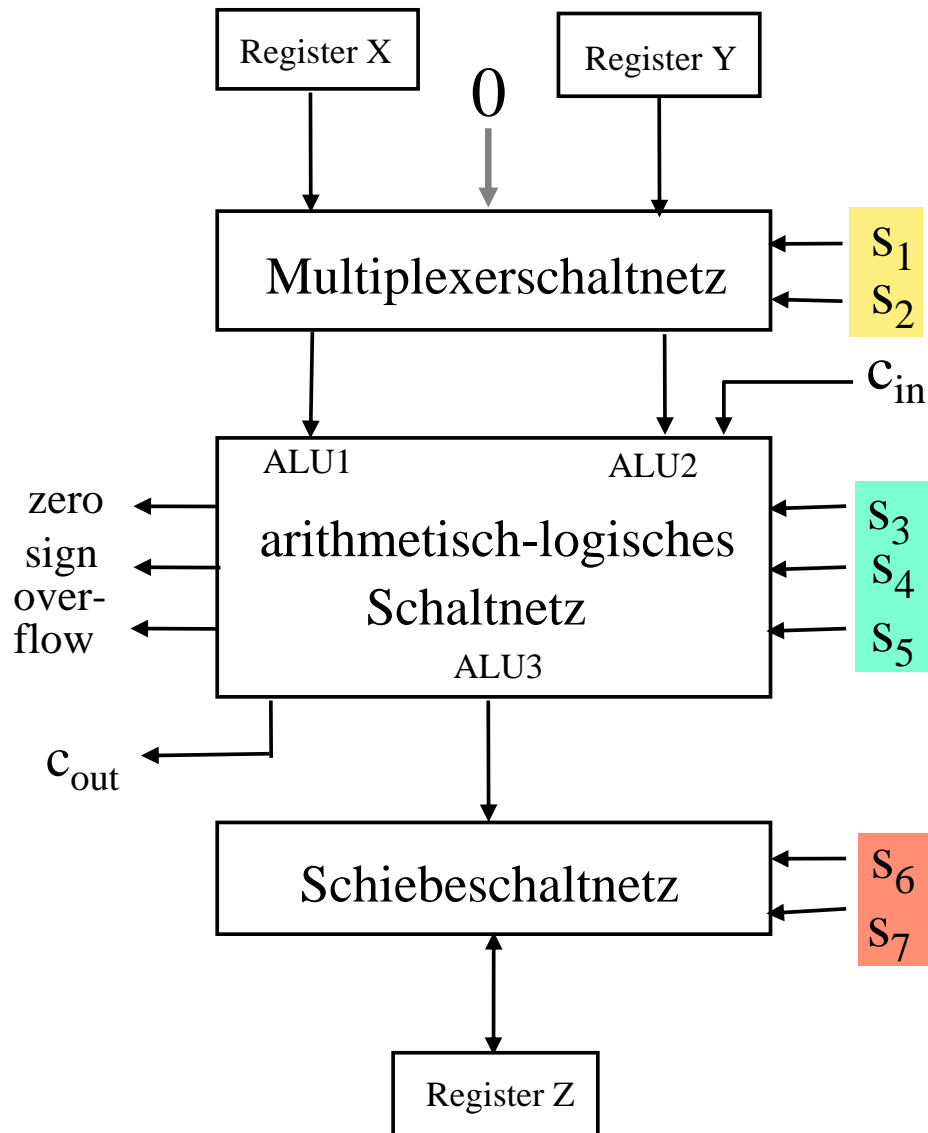


Subtraktion von Zweierkomplementzahlen

Datenfluß des seriellen Multiplizierers



Schema einer einfachen ALU



s_1	s_2	ALU1	ALU2
0	0	X	Y
0	1	X	0
1	0	Y	0
1	1	Y	X

s_3	s_4	s_5	ALU3
0	0	0	$ALU1 + ALU2 + c_{in}$
0	0	1	$ALU1 - ALU2 - \overline{c_{in}}$
0	1	0	$ALU2 - ALU1 - \overline{c_{in}}$
0	1	1	$ALU1 \vee ALU2$
1	0	0	$ALU1 \wedge ALU2$
1	0	1	$\overline{ALU1 \wedge ALU2}$
1	1	0	$ALU1 \nleftrightarrow ALU2$
1	1	1	$ALU1 \leftrightarrow ALU2$

s_6	s_7	Z
0	0	ALU3
0	1	$ALU3 \div 2$
1	0	$ALU3 \times 2$
1	1	Z speichern