

Grundlagen der Informationverarbeitung

Information wird im Computer binär repräsentiert. Die binär dargestellten Daten sollen im Computer verarbeitet werden, d.h. es müssen Rechnerschaltungen existieren, die aus einer binären Eingabe eine binäre Ausgabe erzeugen.

Dafür müssen **Regeln und Vorschriften zur Verarbeitung von Information** auf Bitebene existieren.

Boolesche Algebra - Schaltalgebra

Grundlage der Verarbeitung von Information auf Bitebene ist die **Boolesche Algebra**.

Ursprung: George Boole (1815-1864)

Zweck: Wahrheit und Falschheit von Aussagen zweifelsfrei festzustellen. Die Boolesche Algebra operiert mit den beiden Wahrheitswerten: wahr (*true*) und falsch (*false*).

Bemerkung: Die Boolesche Algebra steht in enger Beziehung zur Aussagenlogik.

Zwei Zustände *false* und *true* → zweiwertig (binär). Diese können auch durch 0 und 1 beschrieben werden.

In der Anwendung (digitalen Elektronik) wird die Boolesche Algebra auch als **Schaltalgebra** bezeichnet.

Wahr	W (von wahr)	T (von true)	H (von high)	1 (Bit gesetzt)
Falsch	F (von falsch)	F (von false)	L (von low)	0 (Bit nicht gesetzt)

Verknüpfungen von Wahrheitswerten

In der *Booleschen Algebra* behandelt man Verknüpfungen von Wahrheitswerten durch *logische Operatoren*, deren Ergebnisse wiederum Wahrheitswerte sind.

Logische Grundverknüpfungen:

Verknüpfung	Name	Schreibweise
nicht a	Negation	$\neg a$
a und b	Konjunktion	$a \wedge b$
a oder b	Disjunktion	$a \vee b$
wenn a dann b	Implikation	$a \Rightarrow b$
a genau dann wenn b	Äquivalenz	$a \Leftrightarrow b$

Boolesche Algebra - formale Definition

Eine *Algebra* wird durch eine Wertemenge, durch ihre Operationen auf dieser Wertemenge und durch die zugehörigen Rechengesetze bestimmt.

Die *Boolesche Algebra* basiert auf der Menge 0,1 und den Operationen \vee , \wedge und \neg . Wobei die Operationen folgendermaßen definiert sind:

Bit a	$\neg a$ (Negation)
1	0
0	1

Bit a	Bit b	$a \wedge b$ (UND)	$a \vee b$ (ODER)
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

Boolesche Algebra - formale Definition *cont.*

In der Booleschen Algebra gelten die folgenden Rechengesetze.

Kommutativgesetz:

$$a \vee b = b \vee a$$

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

Assoziativgesetz:

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

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

Distributivgesetz:

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

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

Absorptionsgesetz:

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

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

und die weiteren Gesetze:

$$0 \vee a = a$$

$$1 \wedge a = a$$

$$a \vee \neg a = 1$$

$$a \wedge \neg a = 0$$

Boolesche Algebra

Durch Kombination aller möglichen Zuordnungen von Argumenten und Ergebnissen ergeben sich die folgenden $2^2 = 4$ einstelligen Wahrheitsfunktionen und die $2^4 = 16$ zweistelligen Wahrheitsfunktionen.

Bit a	a (Negation)	a (Identität)	Konstante 0	Konstante 1
0	1	0	0	1
1	0	1	0	1

Bit a	0	0	1	1		
Bit b	0	1	0	1	Schreibweise	Bezeichnung
f1	0	0	0	0	0	Konstante 0
f2	0	0	0	1	$a \wedge b$	Konjunktion (AND)
f3	0	0	1	0	$\neg(a \Rightarrow b)$	Negation der Implikation
f4	0	0	1	1	a	Identität a
f5	0	1	0	0	$\neg(b \Rightarrow a)$	Negation der Implikation
f6	0	1	0	1	b	Identität b
f7	0	1	1	0	$\neg(a \Leftrightarrow b)$	Antivalenz (XOR)
f8	0	1	1	1	$a \vee b$	Disjunktion (OR)
f9	1	0	0	0	$\neg(a \vee b)$	Nicht-Oder (NOR)
f10	1	0	0	1	$a \Leftrightarrow b$	Äquivalenz
f11	1	0	1	0	$\neg b$	Negation von b
f12	1	0	1	1	$b \Rightarrow a$	Implikation
f13	1	1	0	0	$\neg a$	Negation von a
f14	1	1	0	1	$a \Rightarrow b$	Implikation
f15	1	1	1	0	$\neg(a \vee b)$	Nicht-Und (NAND)
f16	1	1	1	1	1	Konstante 1

Zweistellige logische Verknüpfungen

Durch Kombination der Operationen Konjunktion, Disjunktion und Negation lassen sich alle mögliche einstellige und zweistellige Verküfungen ausdrücken.

Beispiel: f1: $a \wedge \neg a = 0$

f6: $b \wedge 1 = b$

f14: $a \Rightarrow b = \neg a \vee b$

f16: $a \vee \neg a = 1$

Alle Regeln für das Rechnen mit logischen Verknüpfungen ergeben sich aus der Definition der *Booleschen Algebra*. Insbesondere lassen sich folgende Beziehungen herleiten:

$$\neg(\neg a) = a$$

Involutivgesetz

$$a \vee a = a$$

Idempotenzgesetz

$$a \wedge a = a$$

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

de Morgansche Gesetze

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

$$\neg a \vee a \wedge b = \neg a \vee b$$

$$a \vee \neg a \wedge b = a \vee b$$

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

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

Verschmelzungsregeln

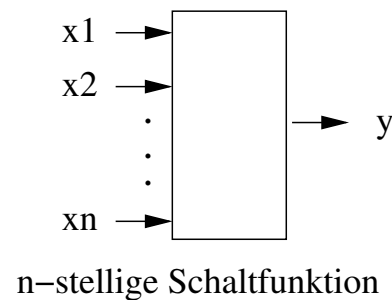
Schaltfunktionen

Bisher 1 Eingang (a) oder 2 Eingänge (a und b) und einen Ausgang.

Verallgemeinerung: n Eingänge 1 Ausgang.

Eine **Schaltfunktion** ordnet n binären Eingangsvariablen x_1, \dots, x_n eine binäre Ausgangsvariable y zu.

$$y = f(x_1, x_2, \dots, x_n)$$



Boolesche Normalform

Zweck:

Mit Hilfe des Theorems der *Boolesche Normalform* kann man aus einer Wahrheitstabelle der Schaltfunktion die Schaltfunktion konstruieren. Die Schaltfunktion wird dabei nur durch die 3 Basis-Operatoren (\neg , \vee , \wedge) ausgedrückt:

Beispiel:

Bit a	Bit b	Bit c	$f(a, b, c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$\text{ges.: } f(a, b, c) = (\neg a \wedge \neg b \wedge c) \vee \dots$$

Boolesche Normalform *cont.*

Benutzung der Identität:

$$f(x_1, x_2, x_3, \dots, x_n) = (\neg x_1 \wedge f(0, x_2, \dots, x_n)) \vee (x_1 \wedge f(1, x_2, \dots, x_n))$$

Führt zur *disjunktiven booleschen Normalform*:

$$\begin{aligned} f(x_1, x_2, x_3, \dots, x_n) = & \\ & (x_1 \wedge x_2 \wedge \dots x_n \wedge f(1, 1, \dots, 1)) \\ & \vee (\neg x_1 \wedge x_2 \wedge \dots x_n \wedge f(0, 1, \dots, 1)) \\ & \vee (x_1 \wedge \neg x_2 \wedge \dots x_n \wedge f(1, 0, \dots, 1)) \\ & \cdot \\ & \cdot \\ & \vee (\neg x_1 \wedge \neg x_2 \wedge \dots x_n \wedge f(0, 0, \dots, 1)) \\ & \vee (\neg x_1 \wedge \neg x_2 \wedge \dots \neg x_n \wedge f(0, 0, \dots, 0)) \end{aligned}$$

Die konjunktiv verknüpften Terme werden als *Minterme* bezeichnet. Für eine n-stellige Funktion gibt es maximal 2^n *Minterme*.

Beispiel

Bit a	Bit b	Bit c	$f(a, b, c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$f(a, b, c) = (\neg a \wedge \neg b \wedge c) \vee$$
$$(a \wedge \neg b \wedge \neg c) \vee$$
$$(a \wedge b \wedge c)$$

Konjunktive Normalform

Analog gibt es die *konjunktive Normalform*. Man erhält sie durch Vertauschen von \vee und \wedge :

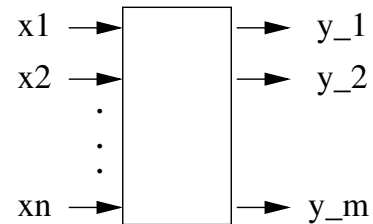
$$\begin{aligned} f(x_1, x_2, x_3, \dots, x_n) = & \\ & (x_1 \vee x_2 \vee \dots x_n \vee f(1, 1, \dots 1)) \\ & \wedge (\neg x_1 \vee x_2 \vee \dots x_n \vee f(0, 1, \dots 1)) \\ & \wedge (x_1 \vee \neg x_2 \vee \dots x_n \vee f(1, 0, \dots 1)) \\ & \cdot \\ & \cdot \\ & \wedge (\neg x_1 \vee \neg x_2 \vee \dots x_n \vee f(0, 0, \dots 1)) \\ & \wedge (\neg x_1 \vee \neg x_2 \vee \dots \neg x_n \vee f(0, 0, \dots 0)) \end{aligned}$$

Die disjunktiv verknüpften Terme werden als *Maxterme* bezeichnet.

Schaltnetze

Bisher: Schaltfunktionen, d.h. nur 1 Ausgang.

Schaltnetze ordnen einer am Eingang anliegenden Binärfolge eindeutig eine Binärfolge am Ausgang zu.

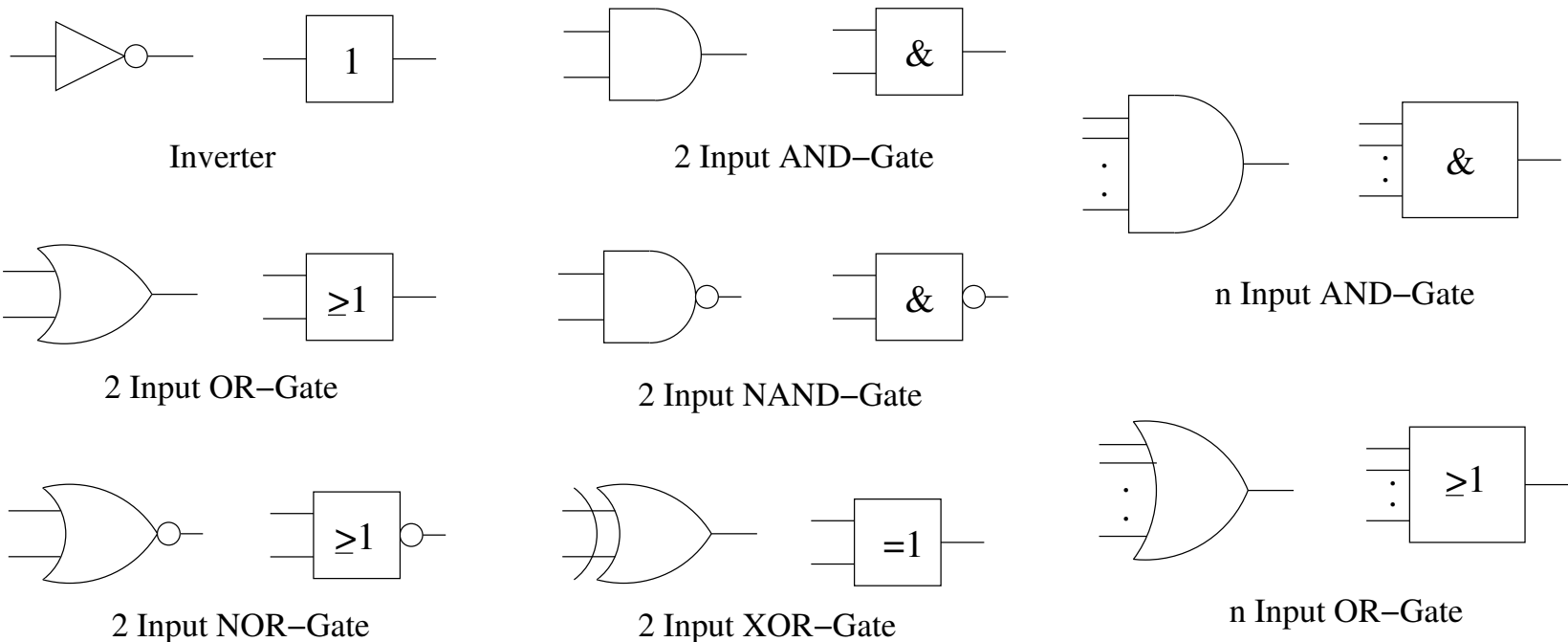


Schaltnetz
mit n Inputs
und m Outputs

Logische Gatter

Schaltnetze und Schaltfunktionen werden aus wenigen Grundbausteinen oder *logischen Gattern* (*logical gates*) aufgebaut.

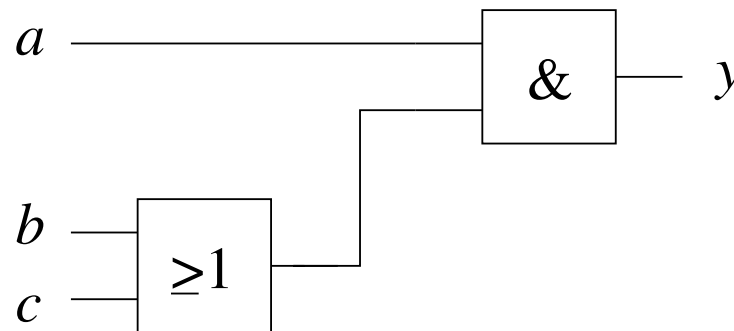
gebräuchliche logische Gatter sind (amerikanische Symbole und DIN 40700 [seit 1976]):



Logische Gatter - Beispiel

Es können aus den logischen Gatten beliebige Schaltfunktionen und Schaltnetze zusammengesetzt werden.

Beispiel: $y = a \wedge (b \vee c)$



Schaltnetze

In der Praxis erhält man Schaltnetze durch:

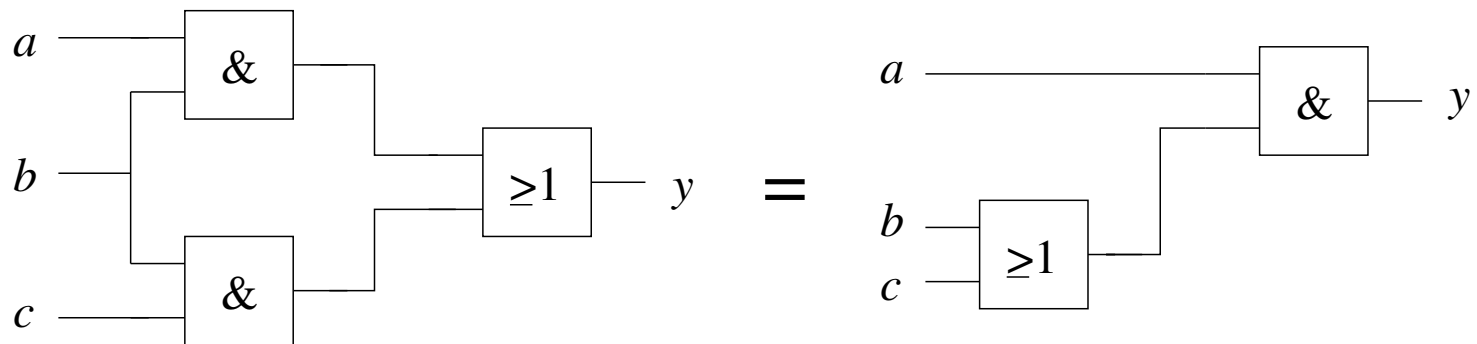
- Gewünschte Verhalten als Wahrheitstabelle darstellen.
- Aus der Wahrheitstabelle die Schaltfunktionen konstruieren (vgl. **boolesche Normalform**).
- Mit dem Gesetzen der booleschen Algebra die Schaltfunktionen umformen und vereinfachen. Hierfür gibt es auch systematische Verfahren (wie z.B. Karnaugh-Veitch-Diagramme).

Vereinfachung von Schaltnetzen

Beispiel:

Distributivgesetz:

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



Beispiel eines Schaltnetzes

1-aus-4-Decoder:

e_1	e_2	a_1	a_2	a_3	a_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

