Schaltalgebra

Eine Einführung

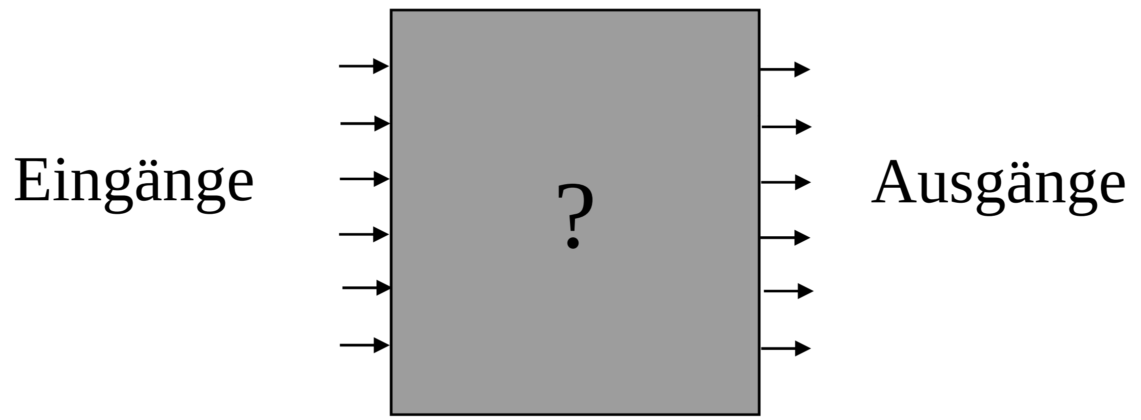
Prof. Dr. W. Küchlin
Informatik I

Digitale Logik und Boolesche Algebra

- **Digitale Logik**: UND- , ODER-, NOT- Gatter:
  Logik der digitalen Schaltungen
- Mathematisch modelliert durch **Boole’sche Algebra**
- **Wie** realisiert man eine binäre Addition?
- **Wie** schaltet man Datenpfade durch?
- **Wie** optimiert man digitale Schaltungen?

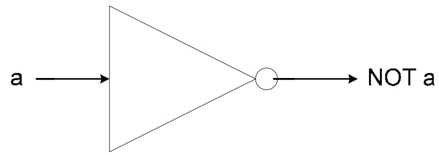
Digitale Logik und Boolesche Algebra

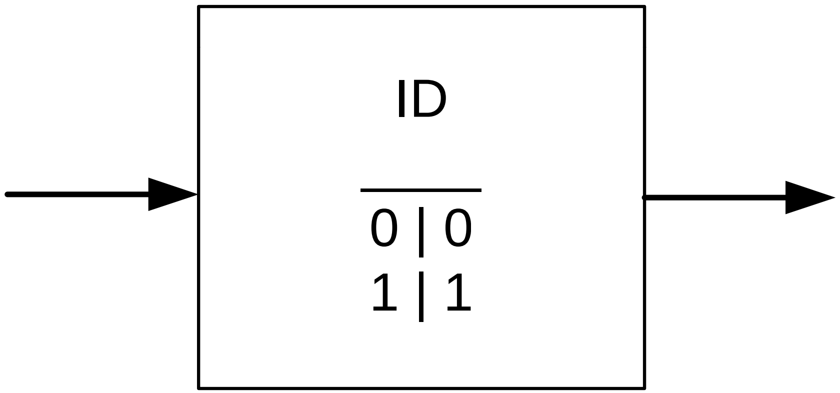
- Wie sind Schaltungen im Computer aufgebaut?
- Es kommen nur die Signale 0 und 1 (bzw. *low* und *high*) vor.
- Signale 0 und 1 auf den Eingängen müssen wieder in Signale 0 und 1 auf den Ausgängen abgebildet werden
- Abbildungen heißen **Schaltfunktionen** (*switching functions*)

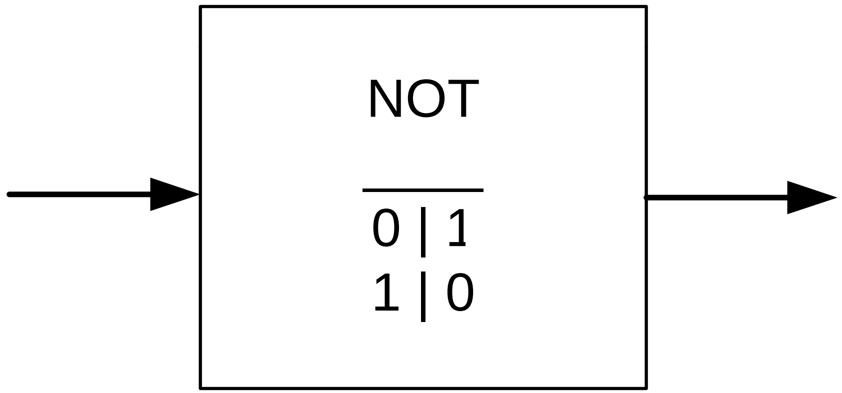


Digitale Logik und Boolesche Algebra

- Spezialfall Boole´sche *Funktionen*: nur ein Ausgang
- Einfachster Fall: ein Eingang, ein Ausgang
- 4 mögliche Schaltfunktionen, NUL, ONE, ID und NOT
- NUL immer 0, ONE immer 1, ID uninteressant
- **NOT** heißt auch Negation, Schaltsymbol:







Digitale Logik und Boolesche Algebra

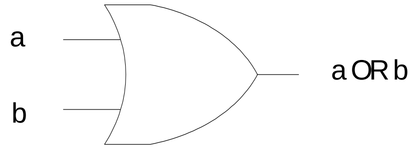
- Nächster Fall: 2 Eingänge, 1 Ausgang
- 4 mögliche Eingangskombinationen
- Je 2 Ausgangswerte möglich → $2^4 = 16$ mögl. Funktionen
- Einige weniger interessant: NUL, a, NOTa, b, NOTb, …
- Interessant: AND, OR, NAND, NOR, XOR, EQV, IMP

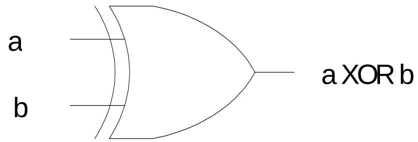
a, b	NUL	NOR		NOTa		NOTb	XOR	NAND	AND	EQV	b	IMP	a		OR	ONE
0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Digitale Logik und Boolesche Algebra

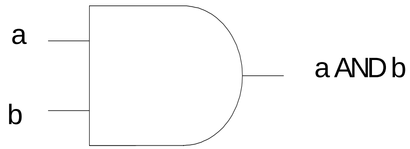
- Die wichtigsten (Schalt-)Gatter: AND, OR, NOT, (XOR, NAND, NOR)
- Schaltbilder nach IEEE Standard (anglo-amerikanisch); weitere: DIN alt und neu

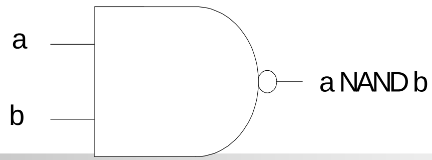
a,b	OR	XOR
0 0	0	0
0 1	1	1
1 0	1	1
1 1	1	0





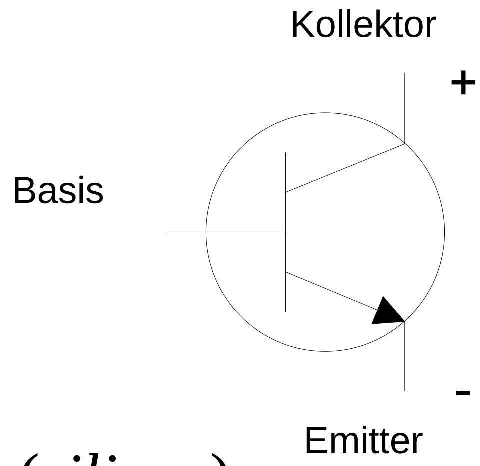
a,b	AND	NAND
0 0	0	1
0 1	0	1
1 0	0	1
1 1	1	0





Exkurs: Bausteine Digitaler Logik (Ebene der Elektrotechnik)

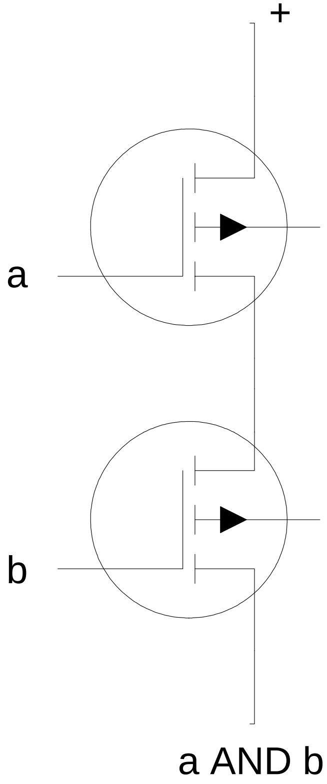
- Logik-Bausteine (**Gatter**, *gates*) heute durch **Transistoren** realisiert
- Transistor: elementarer elektronischer **Schalter**
  - schaltet Strom von **Kollektor** zu **Emitter** durch Spannung an **Basis**
  - **drain, source, (**transistor) **gate**
  - (Elektronenfluss umgekehrt zu Stromfluss)



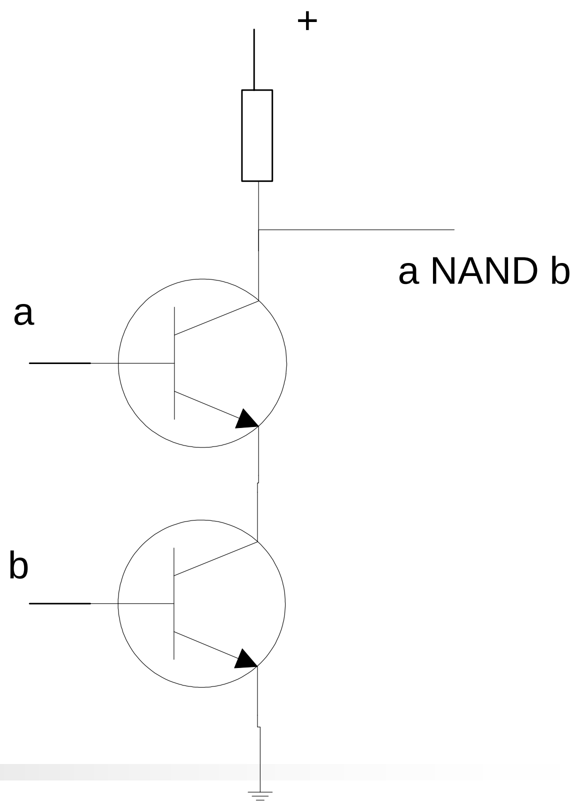
- Transistoren aus Halbleitermaterial Silizium (*silicon*)
- Halbleiter können isolieren oder leiten
- CMOS Feldeffekt-Transistoren induzieren Elektronen in Basis durch Kondensator-Effekt (verlustfrei)
- Bipolare Trans. bringen Elektronen durch (schwachen) Strom in Basis

Exkurs: Bausteine Digitaler Logik (Ebene der Elektrotechnik)

- Die wichtigsten Gatter: AND, OR, NOT, (XOR, NAND, NOR) auf *einfache* Weise realisierbar
- AND mit MOS-FET

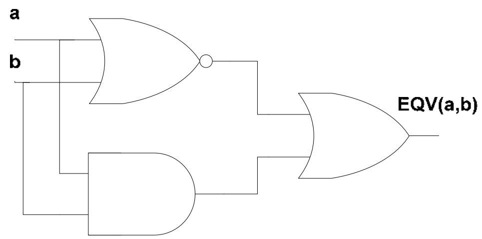


- NAND (Spezialfall: NOT) bipolar



Digitale Logik und Boolesche Algebra

- Verschiedene Schaltungen können die selbe Schaltfunktion realisieren. Beispiele mit AND, OR, NOT:
  - NAND(a,b) = NOT(AND(a,b))
  - NOR(a,b) = NOT(OR(a,b))
  - EQV(a,b) = OR(AND(a,b), NOR(a,b))
  - …



- Beweis durch Vergleich der Funktionstabellen

a,b	NOR	XOR	NAND	AND	EQV	OR
0 0	1	0	1	0	1	0
0 1	0	1	1	0	0	1
1 0	0	1	1	0	0	1
1 1	0	0	0	1	1	1

a,b	AND	NOR	OR(AND,NOR)	EQV
0 0	0	1	1	1
0 1	0	0	0	0
1 0	0	0	0	0
1 1	1	0	1	1

Digitale Logik und Boolesche Algebra

- Jede Schaltfunktion ist Kollektion Boole'scher Funktionen
- Jede Boole'sche Funktion *f* kann durch Kombination von AND, OR, NOT realisiert werden (alternativ: NAND bzw. NOR)
- Beweis durch Einsicht:
  - *f* ist vollständig charakterisiert dadurch, wo sie 1 wird
  - jede 1-Stelle durch AND/NOT-Ausdruck beschreibbar
  - *f* durch OR-Ausdruck über die 1-Stellen charakterisiert

a,b	EQV	AND(NOT(a),NOT(b))	AND	OR(AND(NOT(a),NOT(b), AND(a,b))
0 0	**1**	**1**	**0**	**1**
0 1	**0**	**0**	**0**	**0**
1 0	**0**	**0**	**0**	**0**
1 1	**1**	**0**	**1**	**1**

Digitale Logik und Boolesche Algebra

- Arithmetik mittels Schaltfunktionen realisierbar
- Intuition: endlich viele Ziffern → endlich viele Fälle
- Beispiel: Eine Spalte der Addition $c = a+b$
  - Übertrag von rechts: c_{in} (*carry in*); Übertrag nach links c_{out}
- Kompletter Addierer ist Reihe davon: *ripple carry adder*

a, b, Cin	Cout	c	((a XOR b) AND Cin) OR (a AND B)	(a XOR b) XOR Cin
0 0 0	0	0	0	0
0 1 0	0	1	0	1
1 0 0	0	1	0	1
1 1 0	1	0	1	0
0 0 1	0	1	0	1
0 1 1	1	0	1	0
1 0 1	1	0	1	0
1 1 1	1	1	1	1

Digitale Logik und Boolesche Algebra

Definition: Sei B eine Menge und $\vee$, $\wedge$ seien zwei Verknüpfungen auf B und $0, 1 \in B$ zwei feste Elemente. Es gelte:

1. $\vee$ und $\wedge$ sind assoziativ und kommutativ.
2. Es gelten die Distributivgesetze
   $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ und $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.
3. $x \wedge 0 = 0$, und $x \wedge 1 = x$ für alle x,
4. $x \vee 0 = x$, und $x \vee 1 = 1$ für alle x,
5. Zu jedem x gibt es genau ein x' mit $x \wedge x' = 0$ und $x \vee x' = 1$.
   ($\_'$ ist ein einstelliger Operator in Postfix-Schreibweise)

Dann heißt $[B; \vee, \wedge, ', 0, 1]$ (kürzer: **B**) eine **Boolesche Algebra**.

Digitale Logik und Boolesche Algebra

- Boole’sche Schaltfunktionen bilden eine Boole’sche Algebra

  Schaltalgebra [{0,1}; OR, AND, NOT, 0, 1]
- Beweis: rechne Axiome über Funktionstabellen nach
  - endliche Fallunterscheidung, da nur 0 und 1
  - Beispiel: x OR (y AND z) = (x OR y) AND (x OR z)

x,y,z	y AND z	x OR (y AND z)	x OR y	x OR z	(x OR y) AND (x OR z)
0 0 0	0	0	0	0	0
0 0 1	0	0	0	1	0
0 1 0	0	0	1	0	0
0 1 1	1	1	1	1	1
1 0 0	0	1	1	1	1
1 0 1	0	1	1	1	1
1 1 0	0	1	1	1	1
1 1 1	1	1	1	1	1

Weitere Gesetze (Sätze) der Boole´schen Algebra

1. **Doppelte Negation**: $(x')' = x$
2. **Idempotenz:** $x \lor x = x$, und $x \land x = x$
3. **Absorption:** $x \lor (x \land y) = x$ und $x \land (x \lor y) = x$
   $x \lor (y \land z) = (x \lor y) \land (x \lor z)$ und
   $x \land (y \lor z) = (x \land y) \lor (x \land z)$.
4. **Implikation:** $(x \Rightarrow y) = x' \lor y$ [in Schaltalgebra ist => die Funktion IMP]
5. **De Morgan** Regeln: $(x \lor y)' = (x' \land y')$
   $(x \land y)' = (x' \lor y')$

Digitale Logik und Boolesche Algebra

- Mit den Gesetzen der Boole´schen Algebra lassen sich Boole´sche Audrücke in gleichwertige (äquivalente) umformen
- Schreibe ≡ für Gleichheit bezüglich Boole'scher Algebra
- Gleichheit modulo BA ≡ der Ausdrücke bedeutet Gleichwertigkeit:

  Gleichwertige Ausdrücke stellen dieselbe Schaltfunktion dar
- Dadurch funktional gleichwertiges Ersetzen möglich:
  - **Vereinfachung**: weniger Logik-Gatter
  - **Beschleunigung**: schnellere Schaltungen
  - **Kosten**: billigere/kleinere Bauteile
- Boole´sche Operatoren AND, OR, NOT in Java: &&, ||, !
  - günstigen Java Ausdruck für gewünschte Funktion finden