

# Rechnerarchitektur

Kombinatorische Logik I

Univ.-Prof. Dr.-Ing. Rainer Böhme

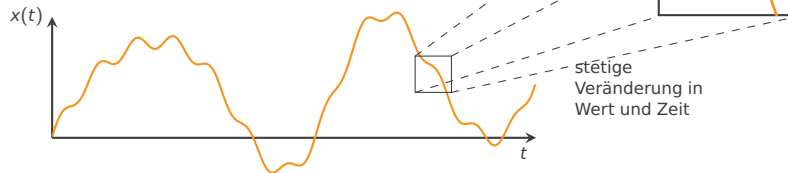
Wintersemester 2021/22 · 13. Oktober 2021

# Gliederung heute

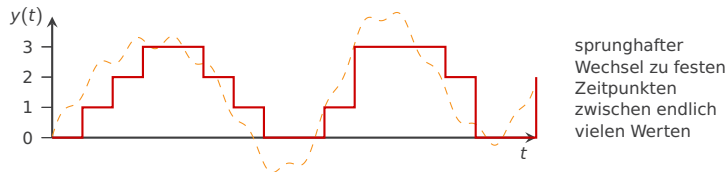
- 1. Grundlagen der Digitaltechnik**
2. Boolesche Algebra
3. Realisierung in Schaltungen

# Analoge und digitale Signale

## Analogtechnik: kontinuierliche Signale



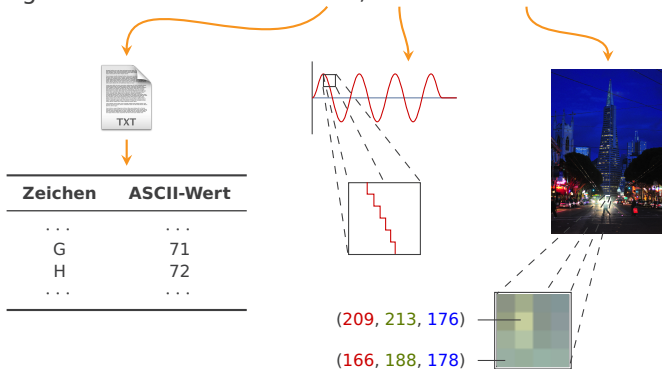
## Digitaltechnik: diskrete Signale (oft auch binär)



# Digitale Daten

Alle Arten von Daten werden digital als **diskrete Zahlen** gespeichert.

Zuordnungen existieren z. B. für Texte, Töne und Bilder.



Digitalrechner können nur digitale Daten verarbeiten.

# Vergleich von Analog- und Digitaltechnik

## Analogrechner

- + Multiplikation, Addition und Filter leicht realisierbar
- + geringer Flächenbedarf
- + sehr schnell
- nichtlineare Bauteile
- niedrige Genauigkeit
- temperaturabhängig
- Speicherung von Daten schwierig

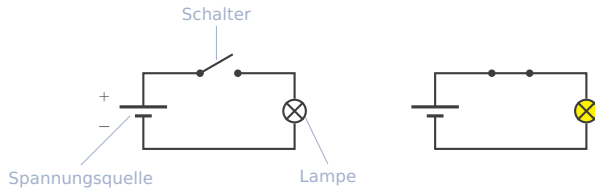
## Digitalrechner

- + weniger störanfällig (z. B. Rauschen)
- + beliebig hohe Genauigkeit erreichbar
- + exakte Reproduktion und Übertragung von Daten
- + einfacher, modularer Entwurf
- oft hoher Flächenbedarf
- hoher Energieverbrauch

Kompromiss zwischen Analog- und Digitaltechnik: Hybridrechner

# Digitaltechnik

Darstellung als elektrische Schaltung

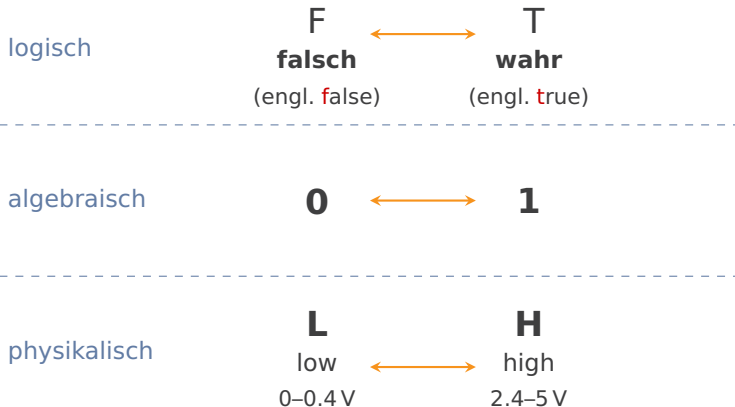


Zwei Zustände:

0. **Strom fließt nicht** (Lampe leuchtet nicht)
1. **Strom fließt** (Lampe leuchtet)

# Varianten der Binärdarstellung

## Interpretation der digitalen Zustände



Beispiel: positive Logik mit TTL-Ausgangspegeln

# Konventionen in der kombinatorischen Logik

- Präferenz der digitalen Zustandsmenge  $\{0, 1\}$
- Realisierung elementarer Operatoren durch **Gatter**
- Realisierung komplexer Funktionen durch Verschalten von Gattern
- Vektorschreibweise für mehrstellige digitale Zustände:

$$\mathbf{x} = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$$

- Die **Dimension**  $n$  ist dabei oft implizit.

## Vorsicht !

InformatikerInnen zählen mitunter auch von 0 bis  $n - 1$   
(angelehnt z. B. an Felder in den Programmiersprachen C und Java).



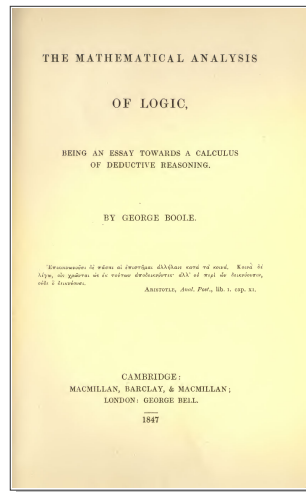
# Gliederung heute

1. Grundlagen der Digitaltechnik
2. **Boolesche Algebra**
3. Realisierung in Schaltungen

# Boolesche Algebra

nach **George Boole** (1815–1864)

- Verbindung von Philosophie (Logik) und Mathematik (Rechenregeln)
- Grundlage für heutige Rechner-Hardware
- Dient dem Entwurf, der Beschreibung, Berechnung und Vereinfachung von Schaltungen und Schaltwerken für die Verarbeitung binärer Größen
- gleichzeitig Grundlage der Theoretischen Informatik  
→ *Einführung in die Theoretische Informatik*



# Operatoren

- Gegeben sei die Boolesche Menge  $\mathbb{B}$  oft  $\mathbb{B} = \{0, 1\}$
- Definition von Operatoren auf Variablen  $x_1, x_2 \in \mathbb{B}$  z.B.  $+$ ,  $\cdot$ ,  $-$
- Vollständige Bestimmung durch **Wahrheitstabelle**
- Die Schreibweise der Operatoren kann variieren.

Achten Sie jedoch auf Konsistenz bei eigener Verwendung !

# Elementare Operatoren

## OR-Operator

logische Summe (ODER)

$+$   $\vee$

$x_1$	$x_2$	$x_1$ OR $x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Das Ergebnis ist 1, falls **mindestens ein** Operand den Wert 1 annimmt.

## AND-Operator

logisches Produkt (UND)

$\cdot$   $*$   $\wedge$

$x_1$	$x_2$	$x_1$ AND $x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Das Ergebnis ist 1, genau dann wenn **beide** Operanden den Wert 1 annehmen.

## NOT-Operator

Invertierung (NICHT)

$\bar{x}$   $\neg x$   $\neg$

$x$	NOT $x$
0	1
1	0

Das Ergebnis ist 1, genau dann wenn der Operand den Wert 0 annimmt.

# Boolesche Algebra

(im engeren Sinne)

## Definition

Die Kombination der Booleschen Menge  $\mathbb{B}$  mit den Operatoren OR, AND und NOT wird als **boolesche Algebra** bezeichnet.

## Schreibweisen

- $(\mathbb{B}, \text{AND}, \text{OR}, \text{NOT})$
- $(\mathbb{B}, \cdot, +, -) = (\{0, 1\}, \cdot, +, -)$
- $\mathbb{B}(\wedge, \vee, \neg)$  auch  $B(\wedge, \vee, \neg)$

Ähnlich wie in der Schulalgebra kann der Punkt-Operator (AND) bei Ausdrücken auch weggelassen werden:  $x_1 \cdot x_2 \Leftrightarrow x_1 x_2$

Es gilt Punkt vor Strich.

# Axiome

der Booleschen Algebra zur Umformung logischer Gleichungen

## Kommutativität

$$x_1 + x_2 = x_2 + x_1 \quad (1)$$

$$x_1 \cdot x_2 = x_2 \cdot x_1 \quad (2)$$

## Distributivität

$$x_1 \cdot (x_2 + x_3) = (x_1 \cdot x_2) + (x_1 \cdot x_3) \quad (3)$$

$$x_1 + (x_2 \cdot x_3) = (x_1 + x_2) \cdot (x_1 + x_3) \quad (4)$$

## Neutrale Elemente

$$0 + x = x \quad (5)$$

$$1 \cdot x = x \quad (6)$$

## Komplementäres Element

$$x + \bar{x} = 1 \quad (7)$$

$$x \cdot \bar{x} = 0 \quad (8)$$

# Sätze

abgeleitet aus den Axiomen

## Idempotenz

$$x + x = x \quad (9)$$

$$x \cdot x = x \quad (10)$$

## Assoziativität

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 \quad (11)$$

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3 \quad (12)$$

## Absorption

$$x_1 + (x_1 \cdot x_2) = x_1 \quad (13)$$

$$x_1 \cdot (x_1 + x_2) = x_1 \quad (14)$$

## Substitution

$$x + 1 = 1 \quad (15)$$

$$x \cdot 0 = 0 \quad (16)$$

## Doppelnegation

$$\overline{\overline{x}} = x \quad (17)$$

# Weitere Gesetzmäßigkeiten

Komplementäre Werte  $\overline{0} = 1$  und  $\overline{1} = 0$

## Abgeschlossenheit

Boolesche Operationen liefern nur boolesche Werte als Ergebnis.

## Dualität

Für jede aus Axiomen ableitbare Aussage existiert eine duale Aussage.

Diese erhält man durch Tausch der Operatoren  $+$  und  $\cdot$  sowie der Werte 0, 1.

De Morgansche Gesetze (folgende Folien)



# Die De Morganschen Gesetze

Das 1. De Morgansche Gesetz lautet:  $\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$

$x_1$	$x_2$	$x_1 \cdot x_2$	$\overline{x_1 \cdot x_2}$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_1} + \overline{x_2}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

# Die De Morganschen Gesetze (Forts.)

Das 2. De Morgansche Gesetz lautet:  $\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$

$x_1$	$x_2$	$x_1 + x_2$	$\overline{x_1 + x_2}$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_1} \cdot \overline{x_2}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

# Die De Morganschen Gesetze (Forts.)

## Negation mithilfe der De Morganschen Gesetze

Negation von Termen erfolgt durch Tausch der Operatoren  $+$  und  $\cdot$  sowie Komplementierung aller Variablen.

Beispiel:

$$\overline{(x_1 + x_2) \cdot \overline{x_3}} \\ = \\ (\overline{x_1} \cdot \overline{x_2}) + x_3$$

# Schaltfunktionen

## Definition

Eine Funktion  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  mit  $n, m \geq 1$  heißt **Schaltfunktion**.

## Spezialfall

Eine Schaltfunktion mit  $m = 1$  heißt  $n$ -stellige **Boolesche Funktion**.

Beschreibung Boolescher Funktionen:

1. eindeutig mit (sortierter) **Wahrheitstabelle**
2. kompakter, aber **nicht eindeutig** mit **Booleschem Ausdruck** (aus Booleschen Variablen und Operationen)

→ Jede **Schaltfunktion** kann durch  $m$  **Boolesche Funktionen** zusammengesetzt werden.

# Boolesche Funktionen

*Wie viele  $n$ -stellige Boolesche Funktionen gibt es?*

- Kombination aller  $2^n$   $n$ -Tupel aus  $\{0, 1\}$  der Argumente mit den Werten  $\{0, 1\}$ :  $2^{(2^n)}$
- Für  $n = 1$ :
  - $f_0(x) = 0$  Kontradiktion (0-stellig)
  - $f_1(x) = x$  Identität
  - $f_2(x) = \neg x$  Negation
  - $f_3(x) = 1$  Tautologie (0-stellig)
- Für  $n = 2$ : 16 zweistellige Boolesche Funktionen

Einige davon sind lediglich auf zwei Argumente erweiterte null- oder einstellige Boolesche Funktionen:  $f_0, f_3, f_5, f_{10}, f_{12}, f_{15}$  (folgende Folien)

# Zweistellige Boolesche Funktionen

$x_2 =$	0	1	0	1	Term	Bezeichnung	Sprechweise
$x_1 =$	0	0	1	1			
$f_0$	0	0	0	0	0	Nullfunktion	
$f_1$	0	0	0	1	$x_1 x_2$	Konjunktion	$x_1$ AND $x_2$
$f_2$	0	0	1	0	$x_1 \overline{x_2}$	1. Differenz	$x_1$ AND NOT $x_2$
$f_3$	0	0	1	1	$x_1$	1. Identität	
$f_4$	0	1	0	0	$\overline{x_1} x_2$	2. Differenz	NOT $x_1$ AND $x_2$
$f_5$	0	1	0	1	$x_2$	2. Identität	
$f_6$	0	1	1	0	$\overline{x_1} x_2 + x_1 \overline{x_2}$	Antivalenz	$x_1$ XOR $x_2$
$f_7$	0	1	1	1	$x_1 + x_2$	Disjunktion	$x_1$ OR $x_2$

# Zweistellige Boolesche Funktionen (Forts.)

$x_2 =$	0	1	0	1	Term	Bezeichnung	Sprechweise	
$x_1 =$	0	0	1	1				
$f_8$	1	0	0	0	$\overline{x_1} + \overline{x_2}$	Negatdisjunktion	$x_1 \text{ NOR } x_2$	„genau dann, wenn“
$f_9$	1	0	0	1	$(\overline{x_1} + x_2)(x_1 + \overline{x_2})$	Äquivalenz	$x_1 \Leftrightarrow x_2$	
$f_{10}$	1	0	1	0	$\overline{x_2}$	2. Negation	NOT $x_2$	„impliziert“
$f_{11}$	1	0	1	1	$x_1 + \overline{x_2}$	2. Implikation	$x_2 \Rightarrow x_1$	
$f_{12}$	1	1	0	0	$\overline{x_1}$	1. Negation	NOT $x_1$	
$f_{13}$	1	1	0	1	$\overline{x_1} + x_2$	1. Implikation	$x_1 \Rightarrow x_2$	
$f_{14}$	1	1	1	0	$\overline{x_1 x_2}$	Negatkonjunktion	$x_1 \text{ NAND } x_2$	
$f_{15}$	1	1	1	1	1	Einsfunktion		

# Vollständige Operatorensysteme

## 1. Alle Booleschen Funktionen können mithilfe der

- **Disjunktion** ( $+$ , OR),
- **Konjunktion** ( $\cdot$ , AND) und
- **Negation** ( $-$ , NOT)

dargestellt werden.

→ Boolesche Basis

## 2. Alle Booleschen Funktionen können

- **entweder** mithilfe der Negation und der Konjunktion
- **oder** mithilfe der Negation und der Disjunktion

dargestellt werden.

→ De Morgan-Basis

## 3. Alle Booleschen Funktionen können

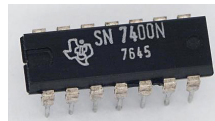
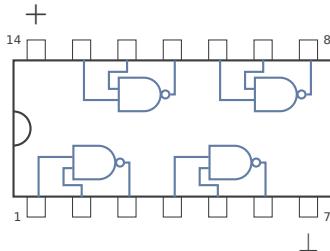
- entweder mithilfe der **NAND-Verknüpfung**
- oder mithilfe der **NOR-Verknüpfung**

dargestellt werden.



# Der 7400er-Chip

Chip der 7400er-Serie mit 4 NAND-Gattern:



Bildquelle: Wikimedia Commons

# Gliederung heute

1. Grundlagen der Digitaltechnik
2. Boolesche Algebra
3. **Realisierung in Schaltungen**

# Technische Realisierung digitaler Systeme

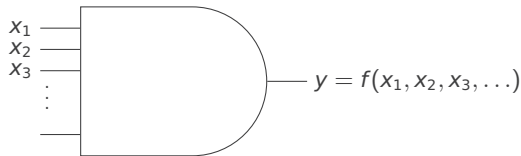
**Gatter** sind elektronische Schalter zur Verknüpfung binärer Argumente.

- Aufbau aus einfachen elektronischen Bauteilen: Widerständen, Dioden, Transistoren
- Verhalten realisiert Booleschen Funktionen mit  $n \geq 1$  **Eingängen**, je einer pro Argument  $(x_1, x_2, x_3, \dots) \in \{0, 1\}^n$ , und einem **Ausgang**  $y \in \{0, 1\}$

**Schaltsymbol** mit zwei Eingängen  
(hier: AND-Gatter)









Verallgemeinerung mit  $n$  Eingängen  
z. B.  $f(x_1, x_2, x_3) = f(x_1, f(x_2, x_3))$  usw.



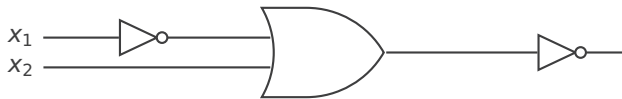
**Konvention:** Die Form des Schaltsymbols lässt auf dessen Funktion schließen.

# Wichtige Boolesche Funktionen als Gatter

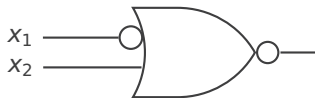
Eingabe		Ausgabe					
$x_1$	$x_2$	elementar			kombiniert		
		$x_1 + x_2$	$x_1 x_2$	$\overline{x_1}$	$\overline{x_1} x_2 + x_1 \overline{x_2}$	$\overline{x_1 + x_2}$	$\overline{x_1 x_2}$
0	0	0	0	1	0	1	1
0	1	1	0	1	1	0	1
1	0	1	0	0	1	0	1
1	1	1	1	0	0	0	0
Schaltsymbol							
Bezeichnung		OR	AND	NOT	XOR	NOR	NAND

# Darstellungsvarianten

Realisierung der Booleschen Funktion  $\overline{\overline{x_1} + x_2}$  mit Gattern:



explizite Darstellung



vereinfachte Darstellung

# Beispiel einer logischen Schaltung

**Gesucht** Schaltung, die 1 ausgibt, wenn einer oder zwei von drei Eingängen  $x_1, x_2, x_3$  den Wert 1 annehmen.

Wahrheitstabelle:

$x_1$	$x_2$	$x_3$	$y$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	0

# Beispiel einer logischen Schaltung

**Gesucht** Schaltung, die 1 ausgibt, wenn einer oder zwei von drei Eingängen  $x_1, x_2, x_3$  den Wert 1 annehmen.

Wahrheitstabelle, Boolesche Funktion:

$x_1$	$x_2$	$x_3$	$y$
0	0	0	0
1	0	0	1 ✓
0	1	0	1 ✓
1	1	0	1 ✓
0	0	1	1 ✓
1	0	1	1 ✓
0	1	1	1 ✓
1	1	1	0

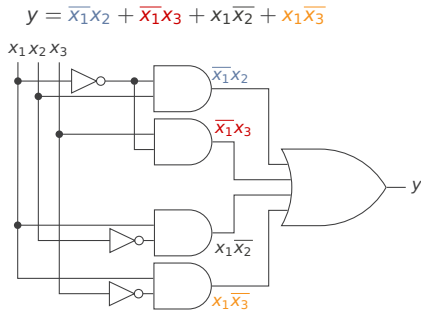
$$y = \overline{x_1}x_2 + \overline{x_1}x_3 + x_1\overline{x_2} + x_1\overline{x_3}$$

# Beispiel einer logischen Schaltung

**Gesucht** Schaltung, die 1 ausgibt, wenn einer oder zwei von drei Eingängen  $x_1, x_2, x_3$  den Wert 1 annehmen.

Wahrheitstabelle, Boolesche Funktion, Realisierung:

$x_1$	$x_2$	$x_3$	$y$
0	0	0	0
1	0	0	1 ✓
0	1	0	1 ✓
1	1	0	1 ✓
0	0	1	1 ✓
1	0	1	1 ✓
0	1	1	1 ✓
1	1	1	0





# Syllabus – Wintersemester 2021/22

06.10.21	1. Einführung
13.10.21	2. Kombinatorische Logik I
20.10.21	3. Kombinatorische Logik II
27.10.21	4. Sequenzielle Logik I
03.11.21	5. Sequenzielle Logik II
10.11.21	6. Arithmetik I
17.11.21	7. Arithmetik II
24.11.21	8. Befehlssatzarchitektur (ARM) I
01.12.21	9. Befehlssatzarchitektur (ARM) II
15.12.21	10. Ein-/Ausgabe
12.01.22	11. Prozessorarchitekturen
19.01.22	12. Speicher
26.01.22	13. Leistung
<b>02.02.22</b>	<b>Klausur (1. Termin)</b>