Jörg Keller, Wolfram Schiffmann

Computersysteme I

Kurseinheit 1-4

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Der Inhalt dieses Studienbriefs wird gedruckt auf Recyclingpapier (80 g/m², weiß), hergestellt aus 100 % Altpapier.

Inhaltsverzeichnis

Kapitel 1

Technische Informatik

Kapitelinhalt

Die Informatik beschäftigt sich mit der automatischen Verarbeitung von Informationen mit Computersystemen. Die Bezeichnung *Informatik* wurde von Karl Steinbuch eingeführt und ergibt sich aus der Kombination der Begriffe Information und Automatik.

Wir können im Wesentlichen drei Gebiete der Informatik unterscheiden:

- Theoretische Informatik,
- Praktische und Angewandte Informatik,
- Technische Informatik.

Die *Theoretische Informatik* widmet sich dabei den formalen und mathematischen Aspekten. Sie beantwortet grundlegende Fragen zu Logik und Deduktion, Formalen Sprachen, Automaten-, Berechenbarkeits- und Komplexitätstheorie sowie zur Algorithmentheorie.

Bei der *Praktischen Informatik* stehen keine abstrakten sondern konkrete Problemstellungen im Vordergrund. Es werden Methoden und Werkzeuge für die Erstellung von Softwarelösungen solcher Probleme entwickelt. Die wichtigsten Teilbereiche bilden die Softwaretechnik, Programmiersprachen, Datenstrukturen, Algorithmen, Datenbanken, Compilerbau und Betriebssysteme.

Zur *Angewandten Informatik* zählen die Teilbereiche Webbasierte Systeme, Mensch-Maschine-Schnittstellen, Multimediatechniken, Künstliche Intelligenz, Wissensbasierte Systeme, Bio-Informatik, Computergrafik, Bildverarbeitung und IT-Sicherheit.

Die *Technische Informatik* vereint Konzepte und Methoden zum Entwurf von Computersystemen, die für verschiedene Anwendungsbereiche optimiert werden. Im Folgenden werden wir – beginnend mit kleinen und wenig performanten Computersystemen – verschiedene Anwendungsbereiche vorstellen.

Wie bei vielen Klassifizierungen gilt auch hier, dass die Gebiete der Informatik nicht disjunkt sind. So gibt es beispielsweise starke Bezüge zwischen der Softwaretechnik und der theoretischen Informatik im Bereich automatischer Ableitung von Programmeigenschaften, die bei einer Code-Umstrukturierung erhalten bleiben sollen. Beim Entwurf eines Mikrochips werden die Schaltnetze heute durch Hardware-Beschreibungssprachen spezifiziert, so dass sich bei Chip-Entwurfssystemen beispielsweise auch umfangreiche softwaretechnische Fragestellungen ergeben.

Auch zur Einordnung einzelner Fachgebiete der Informatik finden sich in der Literatur Unterschiede. So findet sich die Bildverarbeitung teils in der Angewandten, teils in der Technischen Informatik.

1.1 Anwendungsbereiche der Technischen Informatik

Eingebettete Systeme sind in großer Zahl in Geräten der Haushalts- und Unterhaltungselektronik sowie in modernen Fahrzeugen zu finden. Mobile Geräte wie Smartphones oder Tablet-Computer verfügen zusätzlich über berührungsgesteuerte Bildschirme (touch screens) und Telekommunikationsschnittstellen. Desktop-Computer (häufig auch Personal Computer oder PC genannt) besitzen meist großformatige und hochauflösende Bildschirme, eine Tastatur, leistungsstarke Mehrkern-Prozessoren sowie größere Haupt- und Sekundärspeicher. Bei Server-Computern verzichtet man auf Bildschirm bzw. Tastatur und beschränkt sich auf den Zugriff über eine Netzwerk-Schnittstelle. Server sind auf maximale Performanz optimiert und verfügen daher meist über mehrere Mehrkern-Prozessoren sowie graphische Co-Prozessoren (Graphical Processing Unit, GPU) zur Beschleunigung von rechenintensiven Programmen. Um die Performanz weiter zu erhöhen, können mehrere Server über ein leistungsstarkes lokales Netzwerk zu einem Cluster (engl. für Anhäufung) zusammengeschaltet werden. Der Zusammenschluss mehrerer Cluster führt schließlich zum Grid– oder Cloud-Computing, bei dem die Nutzer über das Internet auf extrem performante Parallelrechnersysteme zugreifen können.

Je nach Anwendungsbereich stehen verschiedene Entwurfsziele im Vordergrund. So sollen z.B. mobile Systeme möglichst wenig Energie verbrauchen, um die Akkulaufzeiten zu verlängern. Im Gegensatz dazu haben PCs und Server einen deutlich höheren Energieverbrauch, da es hier vor allem auf eine hohe Performanz ankommt.

1.2 Teilbereiche der Technischen Informatik

Die Technische Informatik umfasst die Teilbereiche

- Hardware-Entwurf (engl. digital design),
- Rechnerarchitektur (engl. computer architecture),
- Messen, Modellieren und Bewerten,
- Echtzeit– und eingebettete Systeme,
- Betriebssysteme, und
- Rechnernetze und Verteilte Systeme.

wobei die Betriebssysteme in natürlicher Weise eine Schnittstelle zwischen Praktischer und Technischer Informatik bilden.

1.2.1 Hardware-Entwurf

Der *Hardware-Entwurf* befasst sich mit der computergestützten Erstellung von Hardware-Schaltungen zur Realisierung von Schaltnetzen, Schaltwerken und Prozessoren.

Die Bandbreite reicht von Überlegungen zur effizienten Realisierung von oft benötigten Schaltungen, zum Beispiel Rechnerarithmetik, über die Eingabe von Schaltungen mittels Hardware-Beschreibungssprachen bis zu Algorithmen zur automatischen Erstellung von Test-Eingabemustern für Mikrochips zur Erkennung von Fabrikationsfehlern.

1.2.2 Rechnerarchitektur

Die *Rechnerarchitektur* befasst sich mit Organisationsformen von Prozessoren, um verschiedene, teilweise in Konflikt stehende Entwurfskriterien wie Leistungsfähigkeit, Ressourcenverbrauch (Leistungsaufnahme, Anzahl Transistoren) und Zuverlässigkeit zu erfüllen.

Die Untersuchungen reichen von der Organisation von Zwischenspeichern zur Erhöhung des Datendurchsatzes über die Anzahl, Auswahl und Steuerung von Ausführungseinheiten für verschiedene Befehle bis zur Struktur von Instruktionssätzen.

1.2.3 Bewertung von Computersystemen

Bei der *Bewertung* von Computersystemen gibt es viele Ansätze auf der Basis von *Modellen* verschiedener Detailgrade sowie durch Ausführung von Anwendungen auf dem realen System. Hierbei darf die *Messung* der relevanten Parameter zur Laufzeit (wie Leistungsaufnahme, Datendurchsatz oder Speicherzugriffszeit) die Ausführung nicht so beeinflussen, dass die Messung verfälscht wird.

1.2.4 Echtzeit– und eingebettete Systeme

Eingebettete Systeme dienen oft der Steuerung von Geräten. Deshalb dürfen Reaktionen des Systems bestimmte Zeiten nicht überschreiten. Zum Beispiel wäre es fatal, wenn die Weitergabe des Kommandos zum Bewegen der Vorderräder beim Steer-by-Wire in Kraftfahrzeugen manchmal erst mit einer Verzögerung von einer halben Sekunde erfolgen würde. *Echtzeitsysteme* erfüllen solche Eigenschaften.

Weiterhin sind solche Systeme oft mobil und daher von einer Batterie oder Akku abhängig, so dass zum Beispiel eine Abwägung zwischen Leistungsfähigkeit und Leistungsaufnahme erfolgen muss.

1.2.5 Betriebssysteme

Ein *Betriebssystem* stellt eine Abstraktionsschicht zwischen einer konkreten Rechnerhardware und Anwendungsprogrammen zur Verfügung. Es ermöglicht beispielsweise, dass mehrere Anwendungsprogramme quasi-gleichzeitig ausgeführt werden, und jedes dabei so ausgeführt werden kann, als ob es alleine auf dem Rechner ausgeführt würde. Gleichzeitig koordiniert und beschränkt es den Zugriff von Anwendungen auf Hardware-Einheiten wie Festplatten oder Netzwerk-Karten.

1.2.6 Kommunikationsnetze und verteilte Systeme

Kommunikationsnetze sorgen dafür, dass verschiedene Rechner miteinander kommunizieren können. Hierbei gibt es eine Fülle von Technologien unterschiedlichster Bandbreite, Verzögerung, oder Übertragungsmedien. Arbeiten mehrere Rechner, die sich an verschiedenen Orten (und in der Regel unter verschiedener administrativer Kontrolle) befinden, an der Lösung eines Problems, so spricht man von einem verteilten System. Mit dem Aufkommen des Grid– und Cloud-Computing hat dieses Feld an Bedeutung zugenommen.

1.3 Einordnung des Moduls Computersysteme

Das Modul *Computersysteme* besteht aus den Kursen *Computersysteme 1* und *Computersysteme 2*.

Der Kurs Computersysteme 1 bietet eine Einführung in den Hardware-Entwurf. Behandelt werden Schaltnetze, Schaltwerke und komplexe Schaltwerke sowie die Grundlagen von Mikroprozessoren.

Der Kurs Computersysteme 2 bietet eine Einführung in die Rechnerarchitektur. Behandelt werden Befehlssatz-Architektur, Mikroarchitektur mit Pipelining, Superskalarität und Sprungvorhersage sowie die Speicherorganisation einschließlich Caches und Externspeichern.

Vielleicht fragen Sie sich, weshalb diese Themen in Ihrem Studiengang vorkommen, wenn Sie doch, wie viele Studierende, eine Tätigkeit in der Software-Entwicklung oder –Administration anstreben. Die Leistungsfähigkeit einer Software hängt jedoch entscheidend von deren Zusammenwirken mit der benutzten Hardware ab. Daher ist es wichtig, die grundlegenden Abläufe bei der Ausführung eines Programms auf der Hardware zu kennen. Daher gibt es sowohl Empfehlungen der Gesellschaft für Informatik als auch Anforderungen der Akkreditierungsagenturen, die ein entsprechendes Fachwissen für ein Informatikstudium vorsehen.

Ein großer Teil der heutigen Software läuft nicht auf PCs oder Servern, sondern auf mobilen oder eingebetteten Geräten. Die Software-Entwicklung für

solche Geräte verlangt damit auch Kenntnis von deren Besonderheiten (zum Beispiel reduzierte Speichergrößen), um diese berücksichtigen zu können.

Während dies lange als Spezialgebiet galt, führen die Verbreitung von mobilen Geräten (Beispiel App-Entwicklung für Smartphones) sowie die Vision vom *Internet der Dinge*, bei dem letztlich alle Alltagsgeräte computergesteuert sind und über (drahtlose) Netzwerke kommunizieren, dazu, dass immer mehr Software-Entwickler Anwendungen für solche Geräte programmieren.

Auch unabhängig vom Anwendungsfeld und trotz Unterstützung durch Softwareenwicklungs-Werkzeuge bedingt eine Performance-Optimierung von Anwendungen in der Regel Kenntnisse der zugrundeliegenden Architektur, zum Beispiel der Cachegröße und –Organisation.

Last but not least: auch wenn Sie derzeit eine konkrete Tätigkeit im Fokus haben, weiß niemand von uns mit Sicherheit, was er oder sie in 10 oder 20 Jahren beruflich machen wird. Hier ist eine gewisse Breite in der Grundlagenausbildung unabdingbar (womit wir wieder bei den Vorgaben der Akkreditierungsagenturen wären).

In diesem Sinne wünschen wir Ihnen viel Freude bei der Bearbeitung des Moduls *Computersysteme*.

Falls wir Sie jetzt sogar neugierig auf Technische Informatik gemacht haben, finden Sie im folgenden Abschnitt Informationen über weitere Module in diesem Bereich.

1.4 Weitere Kurse zur Technischen Informatik

Der Pflichtkurs *Betriebssysteme und Rechnernetze* bietet eine Einführung in die beiden genannten Bereiche. Zu jedem der Bereiche gibt es darüber hinaus ein Wahlpflichtmodul mit vertiefenden Inhalten. Zusätzlich gibt es ein Wahlpflichtmodul *Verteilte Systeme*.

Das Wahlpflichtmodul *PC-Technologie* stellt die Hardware sowie Betriebssystem– und Hardware-nahe Software-Aspekte in PCs vor, und bietet somit eine zu den vorigen Modulen orthogonale Herangehensweise.

Das Wahlpflichtmodul *Parallele Programmierung und Grid-Computing* bietet neben einer Einführung in die Programmierung von Rechnern mit mehreren Prozessoren und verschiedenen Speicherorganisationen auch eine Einführung ins Grid– und Cloud-Computing. Vertiefende Informationen zu diesen Bereichen bietet das Modul *Advanced Parallel Computing*.

Das Wahlpflichtmodul *Virtuelle Maschinen* befasst sich unter anderem mit den Anforderungen an Instruktionssätze und Kontrollmechanismen von Prozessoren, wenn nicht wie bei einem Betriebssystem mehrere Prozesse quasigleichzeitig mit der Illusion, den Rechner alleine zu nutzen, ausgeführt werden, sondern mehrere *virtuelle* Rechner, von denen jeder sein eigenes Betriebssystem

nutzen kann, auf einem physikalischen Rechner ausgeführt werden.

Das Wahlpflichtmodul *Anwendungsorientierte Mikroprozessoren* stellt die Architektur von Prozessoren vor, die nicht in PCs eingesetzt werden. Mikrocontroller dienen vorwiegend Steuerungsaufgaben und digitale Signalprozessoren dienen vorwiegend der Verarbeitung digitalisierter Signale, zum Beispiel Audio– und Videodaten.

Kapitel 2

Schaltnetze

Kapitelinhalt

Schaltnetze basieren auf elektronischen Schaltungen, die Spannungen als logische Variablen 0 oder 1 interpretieren. Mit geeigneten Schaltungen ist man in der Lage, alle grundlegenden Verknüpfungen der Boole'schen Algebra, auch Schaltalgebra genannt, zu realisieren.

Schaltnetze enthalten mehrere Verknüpfungsglieder und realisieren eine Schaltfunktion oder Vektorfunktion:

$$F : \{0,1\}^n \to \{0,1\}^m, \qquad Y = F(X) \tag{2.1}$$

Dabei ist X der Eingangsvektor mit den Eingangsvariablen $x_1, x_2, \ldots, x_n$, Y der Ausgangsvektor mit den Ausgangsvariablen $y_1, y_2, \ldots, y_m$, F die Zuordnungsvorschrift zwischen den Eingangs– und Ausgangsvariablen, gebildet durch die Operatoren $\vee$, $\wedge$, $^-$ und dargestellt durch die Schaltzeichen für die UND–, ODER–, NICHT–Verknüpfung.

Die Schreibweise in Komponentendarstellung lautet:

$$\begin{aligned} y_1 &= f_1(x_1, x_2, \ldots, x_n) \\ y_2 &= f_2(x_1, x_2, \ldots, x_n) \\ \vdots &\quad \vdots \\ y_m &= f_m(x_1, x_2, \ldots, x_n) \end{aligned} \tag{2.2}$$

Die Signallaufzeiten in den Verknüpfungsgliedern werden bei der Beschreibung der Schaltnetze nicht berücksichtigt. Unter dieser Voraussetzung gilt:

> Die Ausgangsvariablen zu irgend einem Zeitpunkt sind eindeutig bestimmt durch die Eingangsvariablen zum gleichen Zeitpunkt.

Die Definition der DIN–Norm (DIN44300/93) für ein Schaltnetz lautet deshalb:

> *Schaltnetz:* Ein Schaltwerk[1], dessen Wert am Ausgang zu irgend einem Zeitpunkt nur vom Wert am Eingang zu diesem Zeitpunkt abhängt.

Schaltnetze werden mit Begriffen und Schaltzeichen der Schaltalgebra beschrieben. Im ersten Abschnitt dieses Kapitels werden Begriffe und Gesetze der Schaltalgebra in einer Übersichtsform dargestellt. In der Analyse von Schaltnetzen wird ausgehend von einem Schaltplan die Funktionstabelle und die Funktionsgleichung erstellt. In der Synthese von Schaltnetzen gelangt man in umgekehrter Reihenfolge von einer Funktionstabelle oder Funktionsgleichung zum Schaltplan.

Als Beispiele für Schaltnetze werden Funktionseinheiten digitaler Rechensysteme dargestellt: Code–Umsetzer, arithmetische Schaltnetze, Multiplexer. Schaltnetze können nach zwei unterschiedlichen Methoden entworfen werden:

[1] eine Funktionseinheit zum Verarbeiten von Schaltvariablen

1. Realisierung durch einzelne Verknüpfungsglieder (UND, ODER, NICHT)

2. Realisierung durch adressierende Bausteine (Multiplexer, Festwertspeicher–ROM, PROM ...)

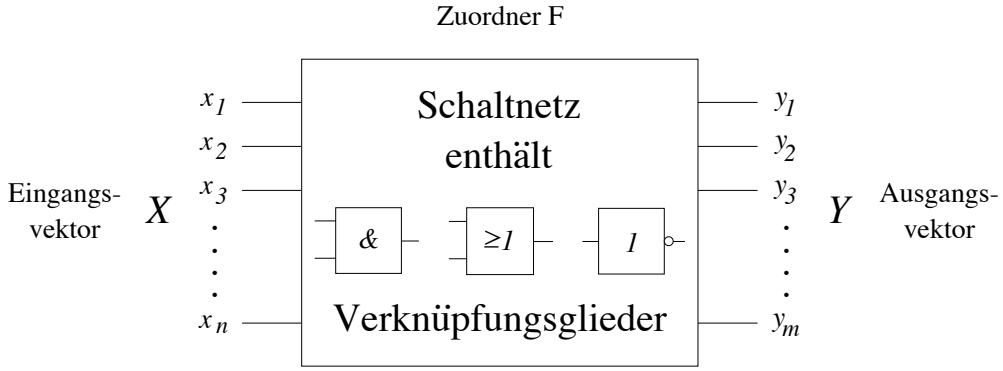


Abbildung 2.1: Blockschaltbild für ein Schaltnetz

In der DIN–Definition von Schaltnetzen werden Laufzeiteffekte nicht berücksichtigt. Reale Schaltnetze sind aus Verknüpfungsgliedern aufgebaut, bei denen Signallaufzeiten auftreten. Deshalb werden am Ende des Kapitels Laufzeiteffekte in Schaltnetzen (Hazards) besprochen.

2.1 Boole'sche Algebra

Die Zweiwertigkeit von Schalterzuständen – Schalter *ein*/Schalter *aus* – führte zur praktischen Anwendung der Booleschen Algebra, zur Schaltalgebra. Die Schaltalgebra ist eine Boolesche Algebra über der Menge $B = \{0, 1\}$. Die Realisierungsmöglichkeit der Schaltalgebra mit Schaltern (Verknüpfungsgliedern) kommt in den Begriffen Schaltvariable, Schaltfunktion und Schaltalgebra selbst zum Ausdruck. Die Schaltalgebra als Modell der Booleschen Algebra bildet die theoretische Grundlage für den Entwurf von Schaltnetzen. Man kann sagen: In digitalen Datenverarbeitungssystemen werden auf der physikalischen Ebene binäre Schaltvariablen mit elektronischen Schaltern (Verknüpfungsgliedern) nach den Gesetzen der Schaltalgebra verknüpft.

2.1.1 Definition der Booleschen Algebra

Die Boolesche Algebra ist eine algebraische Struktur. Sie wird charakterisiert durch folgende Eigenschaften:

1. Es existiert eine Menge $B = \{a, b, \ldots, n\}$
   $\wedge$ und $\vee$ sind eindeutige Verknüpfungen:

$$\begin{aligned} \wedge &: B \times B \longrightarrow B \\ \vee &: B \times B \longrightarrow B \end{aligned}$$

2. Für $a, b, c \in B$ gelten folgende Gesetze:

   2.1 Kommutativgesetze

$$\begin{aligned} a \wedge b &= b \wedge a \quad (K\wedge) \\ a \vee b &= b \vee a \quad (K\vee) \end{aligned}$$

   2.2 Assoziativgesetze

$$\begin{aligned} (a \wedge b) \wedge c &= a \wedge (b \wedge c) \quad (A\wedge) \\ (a \vee b) \vee c &= a \vee (b \vee c) \quad (A\vee) \end{aligned}$$

   2.3 Absorptionsgesetze

$$\begin{aligned} a \wedge (a \vee b) &= a \quad (Ab\wedge) \\ a \vee (a \wedge b) &= a \quad (Ab\vee) \end{aligned}$$

   2.4 Distributivgesetze

$$\begin{aligned} a \wedge (b \vee c) &= (a \wedge b) \vee (a \wedge c) \quad (D\wedge) \\ a \vee (b \wedge c) &= (a \vee b) \wedge (a \vee c) \quad (D\vee) \end{aligned}$$

   2.5 Neutrale Elemente
   Es gibt in B verschiedene Elemente e und $n \in B$, so dass für alle $a \in B$ gilt:

$$\begin{aligned} a \wedge e &= a \quad (N\wedge) \\ a \vee n &= a \quad (N\vee) \end{aligned}$$

   2.6 Komplementäres Element
   Zu jedem $a \in B$ existiert genau ein Element $\overline{a} \in B$ mit den Eigenschaften

$$\begin{aligned} a \wedge \overline{a} &= n \quad (C\wedge) \\ a \vee \overline{a} &= e \quad (C\vee) \end{aligned}$$

   2.7 Dualitätsprinzip
   Ist A eine Aussage der Booleschen Algebra, so auch $\overline{A}$, die man durch Vertauschen von $\wedge$ gegen $\vee$ und n gegen e erhält.

   2.8 De Morgansche Regeln:

$$\begin{aligned} \overline{a \wedge b} &= \overline{a} \vee \overline{b} \\ \overline{a \vee b} &= \overline{a} \wedge \overline{b} \end{aligned}$$

2.1.2 Schaltalgebra – ein Modell der Booleschen Algebra

Die *Schaltalgebra* ist ein Modell der Booleschen Algebra, die nur über zwei Elementen, den beiden Elementen 0 und 1 definiert ist, d.h. die Schaltalgebra ist eine Boolesche Algebra über der Menge $B = \{0, 1\}$. Sie wird durch folgende Eigenschaften charakterisiert

1. Es existiert eine Menge $B = \{0, 1\}$

2. Es existieren die Verknüpfungen (Operatoren) $\wedge, \vee, \bar{}$
   Für die Verknüpfungszeichen werden Schaltzeichen eingeführt (DIN 40700 Teil 14).

3. Es gelten die Gesetze der Booleschen Algebra.

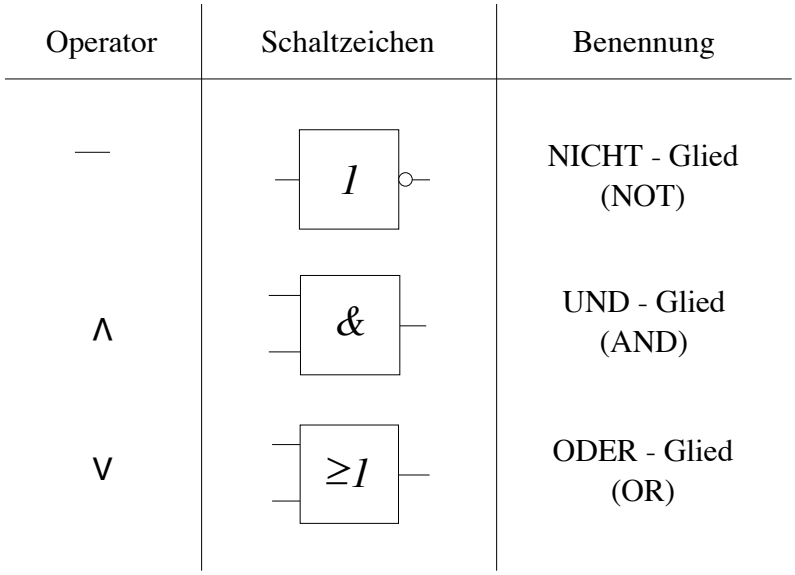


Abbildung 2.2: Operatoren und Schaltzeichen nach DIN 40700

2.2 Schaltfunktionen

Die *Schaltfunktionen* der Schaltalgebra sind vergleichbar mit den Funktionen der (allgemeinen) Algebra.

2.2.1 Definitionen

Eine *Schaltfunktion* ist eine Gleichung der Schaltalgebra, die die Abhängigkeit einer binären Schaltvariablen y (Ausgangsvariable) von einer (oder mehreren) unabhängigen binären Schaltvariablen $x_1, (x_2, ...x_n)$ (Argument– oder Eingangsvariablen) beschreibt (2.2). Die DIN–Formulierung lautet:

> Schaltfunktion– eine Funktion, bei der jede Argumentvariable und die Funktion selbst nur endlich viele Werte annehmen kann. (DIN 44300/87)

Eine Schaltvariable ist ein Symbol für die Elemente $\{0, 1\}$ der Schaltalgebra. Ist $B = \{0, 1\}$, dann bedeutet $x \in B$: die Variable x hat entweder den Wert 0 oder 1. DIN–Formulierung (44300/86): eine Variable, die nur endlich viele Werte annehmen kann (am häufigsten sind binäre Schaltvariablen).

Ist $x_1 \in B$ und $x_2 \in B$, dann hat die Produktmenge (kartesisches Kreuzprodukt) $B \times B = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ vier Elemente oder Wertekombinationen. Als Schreibweise für die Wertekombinationen verwendet man meist folgende Darstellung:

$$\begin{array}{lcll} (0,\,0) & \hat{=} & \overline{\mathrm{x_1}} & \overline{\mathrm{x_2}} \\ (0,\,1) & \hat{=} & \overline{\mathrm{x_1}} & \mathrm{x_2} \\ (1,\,0) & \hat{=} & \mathrm{x_1} & \overline{\mathrm{x_2}} \\ (1,\,1) & \hat{=} & \mathrm{x_1} & \mathrm{x_2} \end{array}$$

Mit n Variablen können 2^n Wertekombinationen gebildet werden. Für die Schaltfunktion gilt deshalb:

> Eine Schaltfunktion ist eine eindeutige Zuordnungsvorschrift, die jeder Wertekombination von Schaltvariablen einen Wert zuordnet.

Jeder der insgesamt 2^n Wertekombinationen der Variablen

$$x_1, x_2, \ldots, x_n \qquad x_i \in \{0, 1\} (i = 1, 2, \ldots, n)$$

wird durch die Zuordnungsvorschrift f eindeutig ein Funktionswert

$$f(x_1, x_2, \ldots, x_n) \quad \in \{0, 1\} \quad \text{zugeordnet.}$$

Man schreibt

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

und sagt

$$y \text{ ist eine Funktion von } x_1, x_2, \ldots, x_n.$$

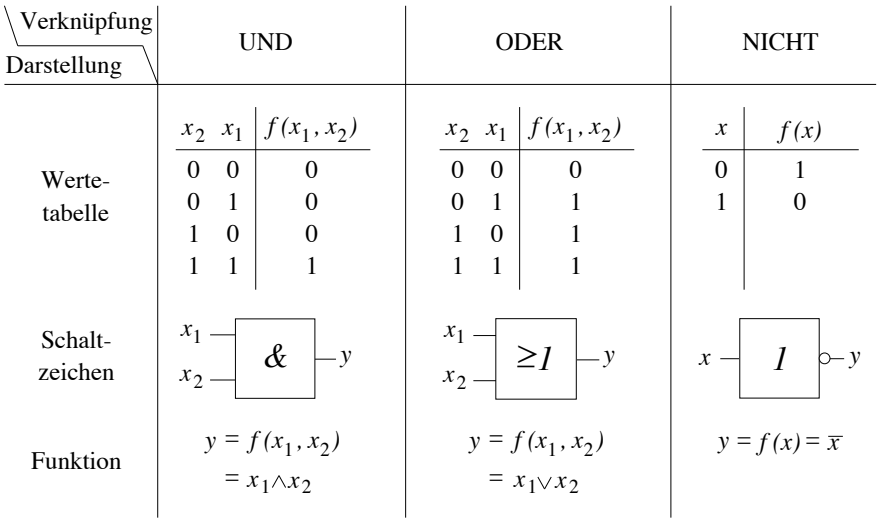


Abbildung 2.3: Grundverknüpfungen und ihre Darstellung

Der Ausdruck $y = f(x_1, x_2, \ldots, x_n)$ wird Schaltfunktion genannt.

Wird eine Schaltfunktion mit Hilfe eines Operationssymbols $(\wedge, \vee, ^-)$ dargestellt, dann heißt diese Schaltfunktion eine *Verknüpfung* (DIN 44300/87). Mit diesen Operationssymbolen werden drei *Grundverknüpfungen* gebildet. Mit diesen Grundverknüpfungen kann jede Schaltfunktion dargestellt werden, es handelt sich bei $(\wedge, \vee, ^-)$ um ein sog. *vollständiges Operatorensystem.*

Es gibt für jede Verknüpfung drei gleichwertige Darstellungen (Abb. 2.3): *Wertetabelle*, *Schaltzeichen* oder die *Angabe der Funktion.*
Weil mit n Eingangsvariablen 2^n Wertekombinationen gebildet werden können und die Ausgangsvariable zwei Werte $0, 1$ annehmen kann, gibt es zu n Eingangsvariablen insgesamt 2^{2^n} Ausgangsfunktionen. Für $y = f(x)$ gibt es vier Funktionen.

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	1	1
1	0	1	0	1

Bei f_1 und f_4 sind die Ausgangswerte unabhängig von den Eingangswerten 0 bzw. 1. Bei f_2 ist der Ausgangswert gleich x, f_3 bildet die Negation von x, also $f_3(x) = \overline{x}$.
Mit zwei Variablen $x_1 \in \{0, 1\}$ und $x_2 \in \{0, 1\}$ lassen sich insgesamt 16 unterschiedliche Verknüpfungen bilden, s. Tabelle 2.1.

Die wichtigsten Verknüpfungen, die als Digitalschaltungen realisiert sind, sind in Abb. 2.4 dargestellt. Mit den in Abb. 2.4 angegebenen Verknüpfungen

Benennung der Verknüpfung	Funktionswert $y = f(x_1, x_2)$ x_1 x_2	Schreibweise mit den Zeichen $\wedge \vee -$	Bemerkung
	$y = 0\ 1\ 0\ 1$ $x_1 = 0\ 1\ 0\ 1$ $x_2 = 0\ 0\ 1\ 1$		
Null	$y_0 = 0\ 0\ 0\ 0$	0	Null
Konjunktion	$y_1 = 0\ 0\ 0\ 1$	$x_1 \wedge x_2$	UND
Inhibition	$y_2 = 0\ 0\ 1\ 0$	$\overline{x}_1 \wedge x_2$	
Transfer	$y_3 = 0\ 0\ 1\ 1$	x_2	
Inhibition	$y_4 = 0\ 1\ 0\ 0$	$x_1 \wedge \overline{x}_2$	
Transfer	$y_5 = 0\ 1\ 0\ 1$	x_1	
Antivalenz	$y_6 = 0\ 1\ 1\ 0$	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$	Exclusiv–ODER
Disjunktion	$y_7 = 0\ 1\ 1\ 1$	$x_1 \vee x_2$	ODER
NOR–Verknüpfung	$y_8 = 1\ 0\ 0\ 0$	$\overline{x_1 \vee x_2}$	NICHT–ODER
Äquivalenz	$y_9 = 1\ 0\ 0\ 1$	$(x_1 \wedge x_2) \vee (\overline{x}_1 \wedge \overline{x}_2)$	
Komplement	$y_{10} = 1\ 0\ 1\ 0$	$\overline{x}_1$	
Implikation	$y_{11} = 1\ 0\ 1\ 1$	$\overline{x}_1 \vee x_2$	
Komplement	$y_{12} = 1\ 1\ 0\ 0$	$\overline{x}_2$	
Implikation	$y_{13} = 1\ 1\ 0\ 1$	$x_1 \vee \overline{x}_2$	
NAND–Verknüpfung	$y_{14} = 1\ 1\ 1\ 0$	$\overline{x_1 \wedge x_2}$	NICHT–UND
Eins	$y_{15} = 1\ 1\ 1\ 1$	1	Eins

Tabelle 2.1: Tabelle der möglichen Verknüpfungen mit zwei Variablen

werden mindestens zwei oder mehr Variablen miteinander verknüpft. Entsprechend haben die Schaltzeichen zwei oder mehr Eingänge. Die verbale Formulierung der Verknüpfung bedeutet

- UND–Verknüpfung: Die Ausgangsvariable y ist dann 1, wenn alle Eingangsvariablen $x_1, x_2, \cdots x_n$ gleich 1 sind.
- ODER–Verknüpfung: Die Ausgangsvariable y ist dann 1, wenn mindestens eine Eingangsvariable x_1 oder x_2 oder $\cdots x_n$ gleich 1 ist.
- Antivalenz (Exklusiv–ODER, XOR): Die Ausgangsvariable y ist dann 1, wenn *entweder* x_1 *oder* x_2 gleich 1 ist, aber nicht beide. Werden mehr als zwei Variablen verknüpft, dann erfolgt die Verknüpfung durch Kaskadierung von zweifach *XOR*–Gliedern $(((x_1 \oplus x_2) \oplus x_3) \oplus \cdots)$.

Selbsttestaufgabe 2.1 *In Tabelle 2.1 gibt es 16 Schaltfunktionen mit zwei Variablen. Wie viele Schaltfunktionen gibt es mit vier Variablen?*

Lösung auf Seite 77

2.2.2 Darstellung

Wie in Abb. 2.3 dargestellt, gibt es für die Grundverknüpfung drei gleichwertige Darstellungen. Ebenso gibt es für Schaltfunktionen verschiedene gleichwertige Darstellungsformen:

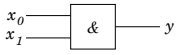
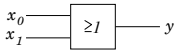
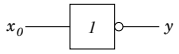
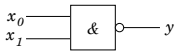
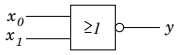
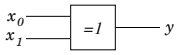
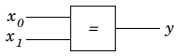
Verknüpfung	Operator-Schreibweise	Schaltzeichen	Wertetabelle
UND	$y = x_0 \wedge x_1$		x_1 x_0 | y 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
ODER	$y = x_0 \vee x_1$		x_1 x_0 | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NICHT	$y = \overline{x_0}$		x_0 | y 0 | 1 1 | 0
NAND	$y = \overline{x_0 \wedge x_1}$		x_1 x_0 | y 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR	$y = \overline{x_0 \vee x_1}$		x_1 x_0 | y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Antivalenz	$y = x_0 \not\equiv x_1$		x_1 x_0 | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Äquivalenz	$y = x_0 \equiv x_1$		x_1 x_0 | y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Abbildung 2.4: Wichtige Verknüpfungen der Digitalelektronik

- Funktionstabelle
- Funktionsgleichung
- KV–Diagramm
- Schaltzeichen
- Binäres Entscheidungsdiagramm

Darstellung als Funktionstabelle

Die Funktionstabelle ist eine Zusammenstellung aller möglichen Wertekombinationen der Eingangsvariablen und der zugehörigen Werte der Ausgangsvariablen (Tabelle 2.2). Sind für alle möglichen Wertekombinationen der Eingangsvariablen, (2^n Kombinationen bei n Variablen), Werte für die Ausgangsvariablen