1. Grundlagen

1.1 Bits

1.1.1 Bit und Signal

Das Bit ist die einfachste Informationsstruktur der Digitaltechnik. Es ist eine einzelne binäre Angabe, die jeweils mit einem von zwei Werten belegt werden kann. Der Begriff wird üblicherweise als Abkürzung von Binary Digit (binäre Ziffernstelle) erklärt.

Auch eine binäre Signalleitung kann einen von zwei Signalwerten führen. Deren Belegung zu einem bestimmten Zeitpunkt t entspricht somit einem Bit (ein Bit = eine Signalbelegung zu einer Zeit). Das Bit ist die abstrakte Informationsstruktur, das Signal deren Darstellung in der Schaltung.

Die beiden Werte bezeichnen
Im Grunde ist es beliebig; Hauptsache, es sind zwei verschiedene Namen. Typische Bezeichnungspaare beziehen sich auf

- die Ziffern des binären Zahlensystems: 0 und 1,
- die Wahrheitswerte der Aussagenlogik: wahr (W) und falsch (F) oder True (T) und False (F),
- die Signalpegel: niedrig (Low oder L) und hoch (High oder H),
- die Wirkungsweise: eingeschaltet, wirkend oder erregt (On, Active, Asserted) und ausgeschaltet, nicht wirkend oder nicht erregt (Off, Inactive, Not Asserted, Negated).

Alle Bezeichnungsweisen können ohne Schwierigkeit ineinander überführt werden. Die Wahl der jeweiligen Bezeichnungsweise hängt von der Zweckmäßigkeit ab. Im folgenden werden, wenn es um Allgemeines geht (also nicht um Wahrheitswerte, Signalpegel und Wirkungszusammenhänge), die beiden Signalwerte mit 0 und 1 bezeichnet.

1.1.2 Wertangaben

Betrachten wir den Ausdruck "10110". Wenn wir nicht genau wissen, worum es geht, würden wir ihn als Zahl "Zehntausendeinhundertzehn" ansehen. Es kann sich aber auch um eine Bitkette oder eine Binärzahl handeln. Wie also die Angaben unterscheiden? Konrad Zuse hatte seinerzeit einfach die binäre Eins auf den Kopf gestellt und durch ein "L" wiedergegeben (10110 binär entspricht also LOLLO und ist deshalb nicht mit Zehntausendeinhundertzehn zu verwechseln). Heutzutage hat man sich an die Schreibweise der Programmiersprachen angelehnt. Man schließt binäre Angaben beispielsweise in Hochkommas ein (Beispiel: '10110') oder stellt ein "B" nach (10110B; diese Bezeichnungsweise soll im folgenden beibehalten werden).

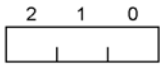
Oktal- und Hexadezimalzahlen

Das sind keine besonderen Datenstrukturen, sondern Darstellungen von Bitketten. Natürlich kann man diese als Folgen von Einsen und Nullen angeben, die man, wie eben gezeigt, beispielsweise mit einem nachgestellten "B" abschließt. Die Probleme:

- Solche Angaben sind lang und unübersichtlich (z. B. folgendes 16-Bit-Wort: 10010000 11001011B).
- Man kann sie kaum aussprechen.

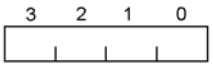
Der Ausweg liegt darin, mehrere Binärstellen in einem Zeichen zusammenzufassen. In einer Oktalzahl werden drei aufeinanderfolgende Bits zusammengefasst, in einer Hexadezimalzahl vier. Bei Oktalzahlen wird jedes Bitmuster mit einer Ziffer zwischen 0 und 7 gekennzeichnet, bei Hexadezimalzahlen mit einer der Ziffern 0...9 oder einem der Buchstaben A...F. Im soeben genannten Beispiel wird das 16-Bit-Wort oktal mit 110313 wiedergegeben und hexadezimal mit 90CB. Oktalzahlen stammen aus der Zeit, als Zeichen noch mit sechs Bits codiert wurden. Um Binärwerte bequem notieren und aussprechen sowie an Bedientafeln anzeigen und eingeben zu können, lag es nahe, sie in drei Bits lange Abschnitte zu zerlegen. Mit der Einführung des 8-Bit-Bytes (IBM System /360) setzte sich die Hexadezimaldarstellung nach und nach durch.

a) Oktaldarstellung
3 Bits werden zusammengefasst:



0	0	0	-	0
0	0	1	-	1
0	1	0	-	2
0	1	1	-	3
1	0	0	-	4
1	0	1	-	5
1	1	0	-	6
1	1	1	-	7

b) Hexadezimaldarstellung
4 Bits werden zusammengefasst (Halbbyte, Tetrade, Nibble):



0	0	0	0	-	0
0	0	0	1	-	1
0	0	1	0	-	2
0	0	1	1	-	3
0	1	0	0	-	4
0	1	0	1	-	5
0	1	1	0	-	6
0	1	1	1	-	7
1	0	0	0	-	8
1	0	0	1	-	9
1	0	1	0	-	A
1	0	1	1	-	B
1	1	0	0	-	C
1	1	0	1	-	D
1	1	1	0	-	E
1	1	1	1	-	F

Abb. 1.1 Oktal- und Hexadezimaldarstellung.

Die Zusammenfassung der Bits in Dreier- oder Vierergruppen beginnt immer mit der niedrigstwertigen Position (also von ganz rechts an). Längere Bitketten werden vom niedrigstwertigen Bit an in Abschnitte von drei oder vier Bits Länge zerlegt. Links außen fehlende Bitpositionen werden stets durch Nullen ergänzt. Beispiele: 11B = 0011B = 3H, 101B = 5H, 11 1010B = 3AH, 101 1101 = 5DH = 135_8.

Mit Binär-, Oktal- und Hexadezimalzahlen rechnen
Es gibt Taschenrechner, die entsprechende Funktionen haben. Auch der in Windows enthaltene Rechner ist brauchbar (Ansicht auf "wissenschaftlich" (XP) oder “Programmierer” umstellen).

Oktalzahlen sind in der heutigen Praxis bedeutungslos (man sollte sie aber kennen, um sich in älteren Handbüchern und Programmtexten zurechtzufinden). Im vorliegenden Text werden ausschließlich Hexadezimalzahlen verwendet. Sie werden durch ein nachgestelltes "H" gekennzeichnet.

Das Dezimaläquivalent
Manchmal wird die Bitkette einfach als vorzeichenlose Binärzahl aufgefasst, deren Wert als Dezimalzahl angegeben wird. Beispiele: 1101B = 13, 10110B = 22. Anwendungsbeispiele:

- Kurzdarstellung von Binärvektoren und von Einträgen in Wahrheitstabellen.
- Darstellung von Codewörtern. So entspricht der ASCII-Code 32 = 20H dem Leerzeichen, der Code 71 = 47H dem Zeichen “G” usw.
- IP-Adressen (Dotted Decimal Notation). Die herkömmliche IP-Adresse ist 32 Bits lang. Die 32-Bit-Angabe wird in vier Abschnitte zu je acht Bits (Octets) eingeteilt. Jedes Octet wird als natürliche (vorzeichenlose) Binärzahl aufgefasst, deren Wert ins Dezimale gewandelt wird. Der jeweils kleinste Wert: 00H = 0, der jeweils größte Wert: FFH = 255. Die vier Dezimalzahlen werden durch Punkte (Dots) voneinander getrennt. Beispiel: 169.254.61.151 = A9 FE 3D 97 = 1010 1001 1111 1110 0011 1101 1001 0111.

1.2 Binärzahlen

1.2.1 Dezimal- und Binärzahlen

Ein Stellenwertsystem zur Basis B beruht auf n Ziffernsymbolen mit den Werten 0, 1 usw. bis B-1. Der Wert der einzelnen Stelle ergibt sich zu

$$\text{Wert des Ziffernsymbols} \cdot \text{Basis}^{\text{Stellen-Index}}$$

Eine n-stellige natürliche Zahl zur Basis B mit den Ziffernstellen $z_n...z_1$ hat den Wert

$$\sum_{i=1}^{n} z_i \cdot B^{i-1}$$

Durch sinngemäßes Anfügen von Brüchen zur Basis B kann die Darstellung auf rationale Zahlen erweitert werden.

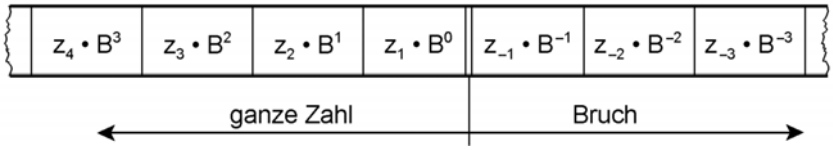


Abb. 1.2 Darstellung rationaler Zahlen in einem Stellenwertsystem zur Basis B.

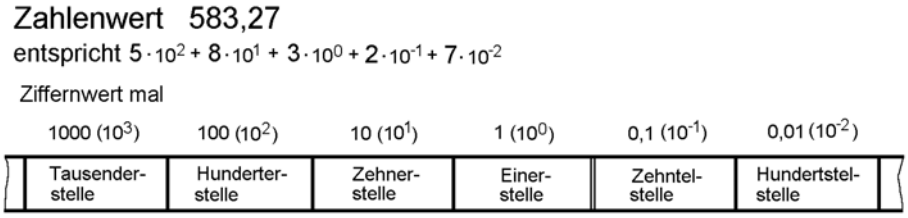


Abb. 1.3 Zahlendarstellung im Dezimalsystem.

Die kleinste Basis eines Stellenwertsystems ergibt sich dann, wenn man nur zwei Zahlzeichen verwendet – die Null und ein weiteres Ziffernsymbol für den Wert Eins. Hiermit kann man beliebigeZahlen in einem Stellenwertsystem zur Basis 2 (Binärsystem) darstellen. Derart dargestellte Zahlenwerte lassen sich mit technischen Mitteln besonders einfach übertragen, speichern und verarbeiten; die Informationswandlungen lassen sich mit den Mitteln der Booleschen Algebra beschreiben (Prinzip der Zweiwertigkeit).

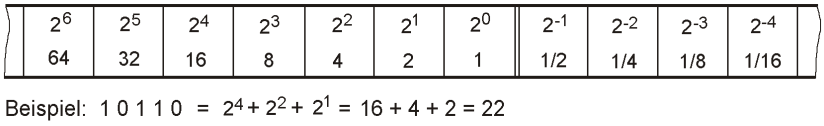


Abb. 1.4 Zahlendarstellung im Binärsystem.

1.2.2 Natürliche Binärzahlen

Die Binärzahlen werden hier nur soweit eingeführt, wie es notwendig ist, um den Zusammenhang mit der Aussagenlogik und den Booleschen Gleichungen zu veranschaulichen. Natürliche (vorzeichenlose) Zahlen bilden die Grundlage jeglicher Zahlenrechnung. Der niedrigste Wert ist Null. Der gesamte Wertebereich einer natürlichen Binärzahl x aus n Bits ist gegeben durch

$$0 \leq x \leq 2^n - 1$$

Der kleinste Wert entspricht einer Belegung mit Nullen, der größte einer Belegung mit Einsen (vgl. Abb. 1.9b). *Beispiel:* eine natürliche Binärzahl aus 16 Bits (n = 16):

- kleinster Wert = 0000 0000 0000 0000B = 0000H = 0,
- größter Wert = $2^{16} - 1$ = 1111 1111 1111 1111B = FFFFH = 65 535.

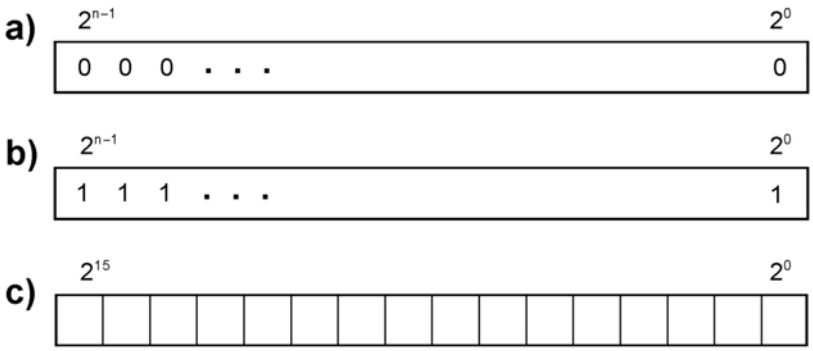


Abb. 1.5 Natürliche (vorzeichenlose) Binärzahlen. a) der kleinste Wert; b) der größte Wert; c) Beispiel (16 Bits).

1.2.3 Elementares Rechnen im Binären

Im Binären kann genauso schulmäßig gerechnet werden wie im Dezimalen. Die Rechenregeln sind vergleichsweise einfach.

Addieren
Addend + Augend = Summe. Die Rechenregeln der stellenweisen Addition:

- 0 + 0 = 0.
- 0 + 1 = 1.
- 1 + 0 = 1.
- 1 + 1 = 2, also binär 11. Die höherwertige Eins ist in der nächst-höheren Stelle als einlaufender Übertrag zu verrechnen.

Der in eine Ziffernstelle einlaufende Übertrag wird wie eine dritte Binärziffer addiert:

- 0 + 0 + 1 = 1, also binär 01 (kein Übertrag in die nächst-höhere Stelle).
- 0 + 1 + 1 = 2, also binär 10 (Übertrag in die nächst-höhere Stelle).
- 1 + 0 + 1 = 2, also binär 10 (Übertrag in die nächst-höhere Stelle).
- 1 + 1 + 1 = 3, also binär 11 (Übertrag in die nächst-höhere Stelle).

Subtrahieren
Minuend – Subtrahend= Differenz. Die Rechenregeln der stellenweisen Subtraktion:

- 0 – 0 = 0.
- 0 – 1 = – 1 = binär 11. Die höherwertige Eins ist in der nächst-höheren Stelle als einlaufender Übertrag zu verrechnen.
- 1 – 0 = 1.
- 1 – 1 = 0.

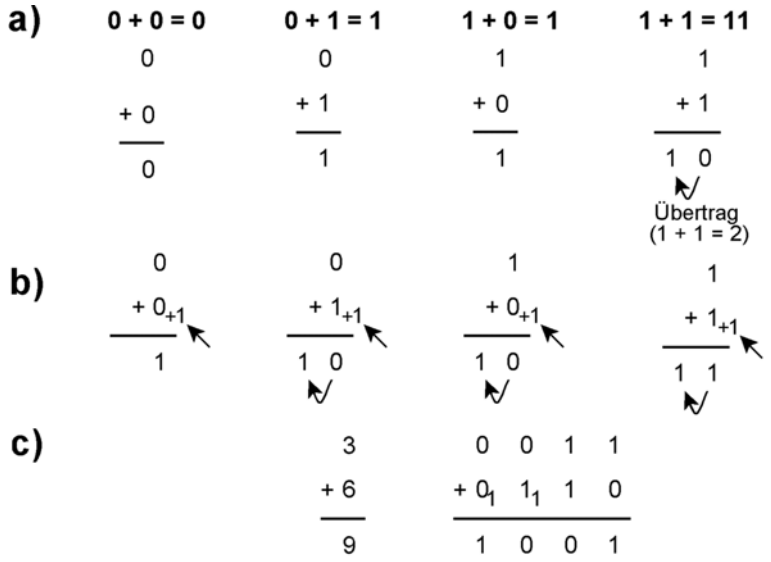


Abb. 1.6 Addition im Binären. a) stellenweises Addieren; b) Einrechnen des einlaufenden Übertrags; c) Rechenbeispiel.

Ergibt sich ein negatives Ergebnis, so wird von der nächst-höheren Stelle eine Eins geborgt, um die Minuendenstelle zu ergänzen. Die geborgte Eins ist in der nächst-höheren Stelle als einlaufender Übertrag zu verrechnen. Statt 0 – 1 wird also gerechnet 10 – 1 = 2 – 1 = 1 + Übertrag, also binär 11. Der Übertrag wird wie eine dritte Binärziffer subtrahiert:

- 0 – 0 – 1 = – 1, also binär 11 (1 + Übertrag in die nächst-höhere Stelle).
- 0 – 1 – 1 = – 2, also binär 10 (0 + Übertrag in die nächst-höhere Stelle).
- 1 – 0 – 1 = 0, also binär 01 (0; kein Übertrag in die nächst-höhere Stelle).
- 1 – 1 – 1 = –1, also binär 11 (1 + Übertrag in die nächst-höhere Stelle).

Der übliche schulmäßige Rechengang: der Übertrag wird zum Subtrahenden addiert; das Ergebnis wird vom Minuenden subtrahiert. Der alternative Rechengang: vor dem Subtrahieren des Subtrahenden wird der Übertrag vom Minuenden subtrahiert.

Multiplizieren

Multiplikand • Multiplikator = Produkt. Das kleine Einmaleins kann einfacher nicht sein:

- 0 • 0 = 0,
- 0 • 1 = 0,
- 1 • 0 = 0,
- 1 • 1 = 1.

Wie beim schriftlichen Multiplizieren im Dezimalen wird der Multiplikator Stellefür Stelle mit dem Multiplikanden multipliziert. Diese Teilprodukte werden – jeweils um eine Stelle verschoben – zueinander addiert.

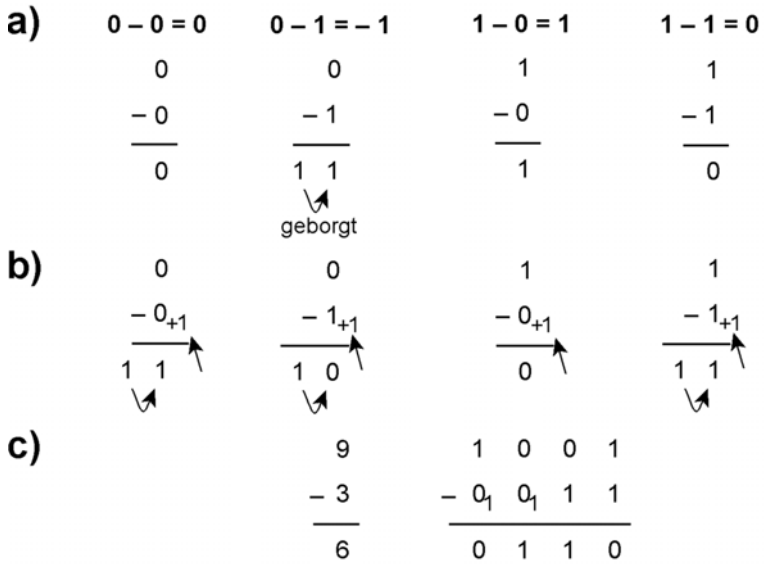


Abb. 1.7 Subtraktion im Binären. a) stellenweises Subtrahieren; b) Einrechnen der geborgten Eins (Übertrag); c) Rechenbeispiel.

Infolge der Einfachheit des Einmaleins kann jedes Teilprodukt nur einen von zwei möglichen Werten annehmen:

- Ist das zugehörige Multiplikatorbit = 0, so ist auch das Teilprodukt = 0.
- Ist das zugehörige Multiplikatorbit = 1, so entspricht das Teilprodukt dem gemäß der Position des Multiplikatorbits nach links verschobenen Multiplikanden.

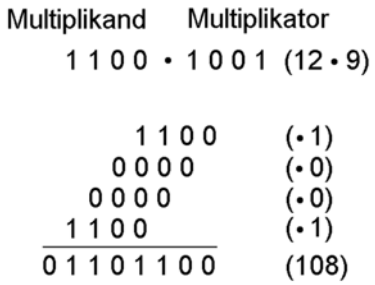


Abb. 1.8 Multiplizieren im Binären.

Dividieren

Dividend : Divisor = Quotient und Rest. Der Ablauf entspricht dem schriftlichen Dividieren im Dezimalen. Der Quotient wird Bitstelle für Bitstelle schrittweise gebildet. Hierzu wird der Divisor linksbündig unter den Dividenden gesetzt. Dann ist zu entscheiden, wie oft der Divisor in den darüber stehenden (gleich langen) Abschnitt des Dividenden hineinpasst. Im Binären ist die Entscheidung einfach: er passt entweder nur einmal hinein oder gar nicht. Um dies zu erkennen, wird der Divisor vom Dividendenabschnitt subtrahiert. Es gibt zwei Möglichkeiten:

- Differenz ≥ 0. Der Divisor ist gleich dem Abschnitt des Dividenden oder größer, passt also hinein. Das Quotientenbit ist dann = 1, und die Differenz wird als Rest verwendet (mit dem weitergerechnet wird).
- Differenz < 0. Der Divisor ist kleiner als der Abschnitt des Dividenden, passt also nicht hinein. Dann ergibt sich ein Quotientenbit = 0, und der Rest entspricht dem unveränderten Dividendenabschnitt.

Um das nächste Quotientenbit zu bilden, wird das Dividendenbit der nächst-niederwertigen Stelle rechts an den Rest angefügt. Von diesem verlängerten Rest wird wieder der Divisor subtrahiert usw. Das Divisionsergebnis aus Quotient und Rest liegt vor, nachdem alle Dividendenbits auf diese Weise ausgewertet wurden.

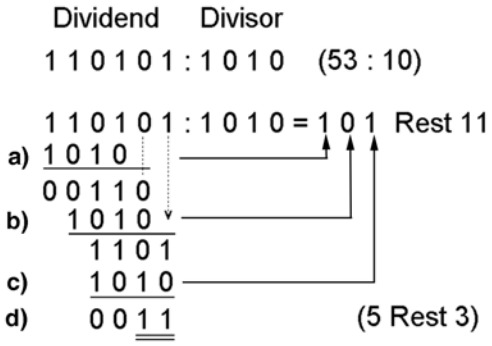


Abb. 1.9 Dividieren im Binären. a) erster Divisionsschritt. Der Divisor ist offensichtlich kleiner. Quotientenbit = 1, Rest = Differenz. b) zweiter Divisionsschritt. Das nächste Dividendenbit wird an den Rest angehängt; der Divisor daruntergesetzt. Er ist offensichtlich kleiner. Quotientenbit = 0; der Rest bleibt,wie er ist. c) dritter Divsionsschritt. Das letzte Dividendenbit wird an den Rest angehängt; der Divisor daruntergesetzt. Er ist offensichtlich größer. Quotientenbit = 1. d) die Differenz ergibt den verbleibenden Rest.

1.4 Aussagenlogik und Boolesche Algebra

Zu den Grundlagen der binären Informationsverarbeitung gibt es zwei Zugänge: die Aussagenlogik und die Boolesche Algebra. Natürlich könnte man Digitalschaltungen auch auf rein intuitiver Grundlage entwickeln. Dieses Vorgehen wird aber bald an Grenzen stoßen. Bereits die Pioniere der Rechentechnik hatten erkannt, dass es notwendig ist, abstrakte symbolische Beschreibungsmittel anzuwenden, denn es war praktisch nicht mehr möglich, die Entwürfe auf herkömmliche Weise (z. B. mit technischen Zeichnungen) zu veranschaulichen und in allen Einzelheiten zu dokumentieren.

1.4.1 Aussagenlogik

Traditionell ist die Logik – die Lehre vom vernunftgemäßen Denken und folgerichtigen Schließen – ein Teilgebiet der Philosophie. Die formale Logik befasst sich mit allgemeinen Gesetzen und Regeln des logischen Schließens. Die mathematische Logik beschreibt die

logischen Probleme mit symbolischen Ausdrucksmitteln (z. B. Formelzeichen und Gleichungen) und behandelt sie mit formalen Methoden, die denen der Mathematik ähnlich sind. Die Aussagenlogik wiederum bildet die Grundstufe der formalen Logik. Sie beschränkt sich auf die Untersuchung von Aussagen, die nur wahr oder falsch sein können. Hierbei wurde erkannt, dass sich beliebig komplizierte Verknüpfungen von Aussagen auf wenige grundlegende Verknüpfungen (Aussagefunktionen) zurückführen lassen.

Wahrheitswerte

Es gibt zwei Wahrheitswerte; wahr (W) und falsch (F). Wird eine Aussage bejaht, so wird sie als wahr beurteilt; ihr wird der Wahrheitswert W zugeordnet. Wird eine Aussage verneint, so wird sie als falsch beurteilt; ihr wird der Wahrheitswert F zugeordnet.

In der Aussagenlogik betrachtet man nicht die Inhalte der Aussagen (z. B. "es regnet"), sondern nur die Wahrheitswerte, die den Aussagen zugeordnet wurden. Wie diese Zuordnung zustande gekommen ist, spielt ebenfalls keine Rolle[1]. Gegenstand der formalen Logik ist allein das formale Schließen ohne jeden Bedeutungsbezug[2]. Die Bedeutungsanalyse ist Sache der Wissenschaft, in der die formale Logik jeweils zur Anwendung kommt. In der Digitaltechnik geht es hierbei u. a. um die richtige Formulierung des Entwurfsproblems und um das richtige Schalten der betreffenden Signale.

Einfachste Aussageverknüpfungen

Betrachtet man zwei Aussagen A, B, die jeweils bejaht oder verneint werden können, so ergeben sich folgende Kombinationen:

- A wird verneint und B wird verneint: A = F, B = F.
- A wird verneint) und B wird bejaht: A = F, B = W.
- A wird bejaht und B wird verneint: A = W, B = F.
- A wird bejaht und B wird bejaht: A = W, B = W.

Aussagefunktionen

Jeder der vier Kombinationen kann ein Wahrheitswert zugeordnet werden. Eine derartige Zuordnung ergibt eine Aussageverknüpfung oder Aussagefunktion. Die miteinander verknüpften Aussagen A, B sind die Variablen der Aussagefunktion f (A, B).

Die Wahrheitstabelle

Wahrheitstabellen (Truth Tables) dienen zur Darstellung von Aussagefunktionen. Eine solche Tabelleenthält alle Kombinationen der Wahrheitswerte der Variablen. In einer weiteren Tabellenspalte werden die der Funktion f (A, B) zugeordneten Wahrheitswerte angegeben.

1: Z. B. wird der Aussage "es regnet" der Wahrheitswert W dann zugeordnet, wenn die Beobachtung ergibt, dass es tatsächlich regnet.

2: Diese klare Trennung sorgt u. a. dafür, dass die Aussagenlogik auf nahezu alles angewendet werden kann, das durch Zweiwertigkeit gekennzeichnet ist.

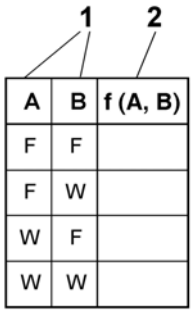


Abb. 1.10 Der grundsätzliche Aufbau einer Wahrheitstabelle am Beispiel einer Aussagefunktion mit zwei Variablen. 1 - die Wahrheitswerte der Variablen (alle Kombinationen); 2 - hier werden die Wahrheitswerte der Aussagefunktion eingetragen.

Elementare Aussagefunktionen

Aus dem umgangssprachlichen Gebrauch heraus unmittelbar einsichtig ist die Rückführung auf folgende drei elementare Funktionen: UND (Konjunktion), ODER (Disjunktion) und NICHT (Negation). Diese Aussagefunktionen kann man mit Wahrheitstabellen darstellen.

UND-Verknüpfung (Konjunktion)

Die Verknüpfung "Aussage A UND Aussage B" ist nur dann wahr, wenn beide Aussagen wahr sind. In allen anderen Fällen ist sie falsch.

ODER-Verknüpfung (Disjunktion)

Die Verknüpfung "Aussage A ODER Aussage B" ist dann wahr, wenn wenigstens eine der Aussagen wahr ist. Sie ist nur dann falsch, wenn beide Aussagen falsch sind.

NICHT-Funktion (Negation)

Die Negation einer Aussage ("NICHT Aussage A") ist nur dann wahr, wenn die Aussage falsch ist und umgekehrt.

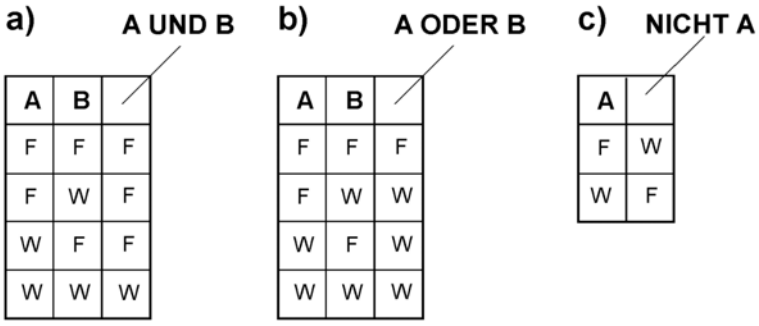


Abb. 1.11 Wahrheitstabellen elementarer Aussagefunktionen. a) Konjunktion; b) Disjunktion; c) Negation.

1.4.2 Boolesche Algebra

Am Anfang der Entwickung der mathematischen Logik stand der Wunsch, das spitzfindige logische Schließen durch ein formales, schematisches Rechnen zu ersetzen (Algebra der Logik). In Verfolgung dieses Ziels hatte George Boole in der Mitte des 19. Jahrhunderts eine der elementaren Mathematik entsprechende symbolische Darstellung logischer Funktionen angegeben, die Wahrheitswerte mit 0 und 1 bezeichnet und Zusammenhänge und Schlussweisen der Logik in Form von Gleichungen dargestellt. Ausgehend davon verwenden die Mathematiker Bezeichnungen wie Boolesche Algebra, Boolesche Variable, Boolescher Raum, Boolesche Funktion, Boolesche Gleichung usw. Die heutige Mathematik hat allerdings mit dem Inhalt der Originalarbeiten Booles nicht mehr viel zu tun. Aus den Anfängen der mathematischen Logik heraus haben sich mehrere Spezialgebiete entwickelt. Eines davon ist die Schaltalgebra (Switching Algebra). Deren Gegenstand ist die Anwendung der Booleschen Algebra beim Entwerfen, Analysieren, Optimieren, Implementieren und Prüfen von Digitalschaltungen.

Der Begriff "Boolesche Algebra" bezeichnet im Grunde zweierlei. Zum einen ist er eine pauschale Sammelbezeichnung für den hier in Rede stehenden Formalismus mit Nullen und Einsen. Zum anderen ist eine Algebra eine mathematische Struktur, die durch eine Menge und bestimmte Operationen über die Elemente dieser Menge definiert ist. Je nachdem, auf welche Menge man sich bezieht und welche Operationen über die Elemente dieser Menge vorgesehen werden, ergeben sich verschiedene Boolesche Algebren. Elementare Boolesche Algebren beruhen auf einer Menge B (Trägermenge), die nur zwei Elemente hat, nämlich 0 und 1: B = {0, 1}. U. a. gibt es eine Boolesche Algebra, die der oben skizzierten Aussagenlogik entspricht. In ihr sind die Operationen UND, ODER und NICHT definiert. Unter Verzicht auf Formelzeichen kann diese Boolesche Algebra wie folgt angegeben werden:

$$(\{0, 1\}, \text{UND}, \text{ODER}, \text{NICHT}) \qquad (1.1)$$

Boolesche Variable
Eine Boolesche Variable kann nur einen der beiden Wahrheitswerte (0 oder 1) annehmen.

Boolesche Funktion und Aussagefunktion
Im Grunde ist es das gleiche. Man spricht üblicherweise dann von einer Booleschen Funktion, wenn die Wahrheitswerte mit 0 und 1 bezeichnet werden.

Elementare Formelzeichen
Hier macht sich sofort eine Schwierigkeit bemerkbar: es gibt verschiedene Formelzeichen für jede der aussagenlogischen Verknüpfungen. Deshalb sei folgende Symbolik eingeführt:

- Boolesche Variable werden mit einzelnen Kleinbuchstaben bezeichnet (a, b usw.).
- Die UND-Verknüpfung (Konjunktion) wird durch einen Punkt dargestellt:
  a UND b = a • b.

- Die ODER-Verknüpfung (Disjunktion) wird durch das Zeichen $\vee$ dargestellt[3]: a ODER b = a $\vee$ b.
- Die Negation wird durch Überstreichen der zu negierenden Variablen dargestellt: NICHT a = $\overline{a}$.
  Betrifft die Negation einen Ausdruck aus mehreren Variablen, so wird der gesamte Ausdruck überstrichen:
  NICHT (a $\vee$ b) = $\overline{a \vee b}$.

Mit diesen Vereinbarungen kann die Boolesche Algebra (1.1) folgendermaßen angegeben werden:

$$\left(\{0,1\},\ \cdot,\ \vee,\ \overline{\ }\right) \qquad (1.2)$$

a)

a	b	a • b
0	0	0
0	1	0
1	0	0
1	1	1

b)

a	b	a v b
0	0	0
0	1	1
1	0	1
1	1	1

c)

a	$\overline{a}$
0	1
1	0

Abb. 1.12 Wahrheitstabellen elementarer Aussagefunktionen auf Grundlage der Wahrheitswerte 0 und 1. a) Konjunktion; b) Disjunktion; c) Negation.

Naheliegende Analogien

George Boole hat seine Symbole aus der gewohnten Mathematik entnommen und dabei naheliegende Analogien ausgenutzt:

Konjunktion und Multiplikation

Eine Verknüpfung a **op** b heißt multiplikativ, wenn sie stets den Wert des jeweils anderen Operanden ergibt, sofern einer der Operanden das Einselement: a • 1 = a. Es ist auf den ersten Blick zu erkennen, dass dies für die Konjunktion erfüllt ist und dass die Wahlrheitstabelle der Konjunktion dem kleinen Einmaleins der Binärzahlen entspricht.

Disjunktion und Addition

Eine Verknüpfung a **op** b heißt additiv, wenn sie stets den Wert des jeweils anderen Operanden ergibt, sofern einer der Operanden das Nullelement ist: a + 0 = a vgl. (1.4)). Die Analogie zwischen der Wahrheitstabelle der Disjunktion und der binären Addition gilt offensichtlich für die ersten drei Wertekombinationen: 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 0. Was die dritte Wertekombination angeht, so stoßen wir auf eine Spitzfindigkeit: George Boole

3: Von lat. vel = oder.

hatte die ODER-Verknüpfung analog zur Addition behandelt (und das Pluszeichen (+) als Symbol eingeführt). Diesem Ansatz hatte er aber das sog. ausschließende ODER (Exklusiv-ODER, Antivalenz) zugrunde gelegt. Diese Aussagenverknüpfung schließt den Fall aus, dass beide Aussagen gleichzeitig wahr sind (Beispiel: "der Verein gewinnt die Meisterschaft oder steigt ab" – beides kann zwar vorkommen, aber offensichtlich nicht zugleich). Somit gilt 1 + 1 = 0 (Addition modulo 2). Spätere Untersuchungen haben jedoch ergeben, dass es zweckmäßiger ist, der mathematischen Logik das sog. einschließende ODER zugrunde zu legen. Aber auch diese ODER-Verknüpfung ist additiv, denn $a \vee 0 = a$ ($0 \vee 0 = 0$; $1 \vee 0 = 1$).

Boolesche Räume

Einer Booleschen Variablen kann jeweils eines der Elemente der Menge B zugewiesen werden. Wir betrachten k Boolesche Variable a_1, a_2, ...$a_k \in \{0, 1\}$. Insgesamt können k Variable 2^k mögliche Belegungen annehmen. Die Menge dieser Belegungen bildet einen Booleschen Raum B^k. Ein solcher Raum ist ein abstraktes Gebilde, das aus 2^k einzelnen (diskreten) Punkten besteht (Punktraum).

Boolesche Funktionen

Eine Boolesche Funktion ist eine Abbildung $B^k \rightarrow B$, die jedem Punkt des Booleschen Raumes B^k einen der Werte 0 oder 1 zuweist. Da es bei k Variablen 2^k Punkte gibt und jedem Punkt wiederum einer von zwei Wahrheitswerten zugeordnet werden kann, gibt es insgesamt 2^{2^k} mögliche Boolesche Funktionen von k Binärvariablen. Die folgenden Tabellen enthalten alle Booleschen Funktionen mit einer und mit zwei Binärvariablen. Jede Tabellenzeile enthält die Wahrheitstabelle der betreffenden Funktion.

Funktionswerte bei		**Bezeichnung**
a = 1	**a = 0**	
0	0	Festwert 0
0	1	Negation ($\bar{a}$)
1	0	Identität (a)
1	1	Festwert 1

Tabelle 1.1 Alle 4 Booleschen Funktionen mit einer Binärvariablen.

Negation (Verneinung, Invertierung)

Der Wahrheitswert wird invertiert. Aus 0 wird 1, aus 1 wird 0.

Identität (Bejahung)

Der Wahrheitswert der Variablen wird unverändert beibehalten. In der Digitaltechnik entspricht diese (an sich triviale) Funktion dem Signalverstärker oder Treiber, der das Eingangssignal lediglich zum Ausgang weiterreicht.

Funktionswerte bei [a,b] =				Bezeichnung
1,1	**1,0**	**0,1**	**0,0**	
0	0	0	0	Festwert 0
0	0	0	1	NOR; $\overline{a \vee b}$
0	0	1	0	$\bar{a} \cdot b$ (Inhibition)
0	0	1	1	$\bar{a}$ (Negation)
0	1	0	0	$a \cdot \bar{b}$ (Inhibition)
0	1	0	1	$\bar{b}$ (Negation)
0	1	1	0	Exklusiv-ODER; $a \cdot \bar{b} \vee \bar{a} \cdot b = a \oplus b$ (Ungleichheit, Antivalenz, XOR)
0	1	1	1	NAND; $\overline{a \cdot b}$
1	0	0	0	UND; $a \cdot b$ (Konjunktion, AND)
1	0	0	1	Äquivalenz; $\bar{a} \cdot \bar{b} \vee a \cdot b = a \otimes b$ (Gleichheit, Koinzidenz (COIN), Exklusiv-NOR, XNOR)
1	0	1	0	b (Identität)
1	0	1	1	$\bar{a} \vee b = \overline{a \cdot \bar{b}}$ (Implikation $a \rightarrow b$)
1	1	0	0	a (Identität)
1	1	0	1	$a \vee \bar{b} = \overline{\bar{a} \cdot b}$ (Implikation $b \rightarrow a$)
1	1	1	0	ODER; $a \vee b$ (Disjunktion, OR)
1	1	1	1	Festwert 1

Tabelle 1.2 Alle 16 Booleschen Funktionen mit zwei Binärvariablen.

UND (Konjunktion, AND)
Die UND-Verknüpfung nimmt nur dann den Wahrheitswert 1 an, wenn beide Variablen den Wahrheitswert 1 haben. Der Wahrheitswert 0 ergibt sich bereits dann, wenn nur eine der Variablen den Wahrheitswert 0 hat.

ODER (Disjunktion, OR)
Die ODER-Verknüpfung nimmt dann den Wahrheitswert 1 an, wenn wenigstens eine der Variablen den Wahrheitswert 1 hat. Damit sich der Wahrheitswert 0 ergibt, müssen beide Variable den Wahrheitswert 0 haben.

NAND
Es ist eine negierte UND-Verknüpfung (NOT-AND). Sie nimmt nur dann den Wahrheitswert 0 an, wenn alle Variablen den Wahrheitswert 1 haben. Der Wahrheitswert 1 ergibt sich bereits dann, wenn nur eine der Variablen den Wahrheitswert 0 hat.

NOR
Es ist eine negierte ODER-Verknüpfung (NOT-OR). Sie nimmt dann den Wahrheitswert 0 an, wenn wenigstens eine der Variablen den Wahrheitswert 1 hat. Damit sich der Wahrheitswert 1 ergibt, müssen beide Variable den Wahrheitswert 0 haben.

Antivalenz (Exklusiv-ODER, XOR, Ungleichheit)
Diese Funktion hat immer dann den Wahrheitswert 1, wenn sich die Wahrheitswerte der beiden Variablen unterscheiden, das heißt, wenn entweder Variable a wahr und Variable b falsch ist oder umgekehrt. Auf Grund dieses Entweder-Oder-Verhaltens heißt die Funktion auch ausschließendes oder Exklusiv-ODER (XOR). Der Wahrheitswert 0 ergibt sich, wenn beide Variable den gleichen Wahrheitswert haben (0, 0 oder 1, 1).

Äquivalenz (Exklusiv-NOR, XNOR, Gleichheit; Koinzidenz (COIN))
Diese Funktion hat immer dann den Wahrheitswert 1, wenn die Wahrheitswerte der beiden Variablen gleich sind (0, 0 oder 1, 1). Der Wahrheitswert 0 ergibt sich, wenn beide Variable unterschiedliche Wahrheitswerte haben (0, 1 oder 1, 0). Diese Funktion ist die Negation des ausschließenden ODER; deshalb wird sie auch als Exklusiv-NOR (XNOR) bezeichnet.

Die Äquivalenz ist die Negation der Antivalenz und umgekehrt. Ausschließende ODER-Verknüpfungen können in der Aussagenlogik vorkommen. Die ODER-Verknüpfung ist dann ausschließend, wenn beide Aussagen nicht zugleich wahr sein können. In der Schaltalgebra sind vor allem folgenden Eigenschaften von Bedeutung:

- Die beiden ausschließenden Verknüpfungen (XOR und XNOR) drücken die Ungleichheit bzw. Gleichheit der Wahrheitswerte von a und b aus (die Antivalenz ergibt eine Eins bei Ungleichheit, die Äquivalenz bei Gleichheit).
- Die Antivalenzverknüpfung entspricht einer binären Addition modulo 2 oder – was das Gleiche bedeutet – der Summe in der niedrigstwertigen Binärstelle (0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0).

Inhibition
Die Inhibition (Verhinderung) ist eine UND-Verknüpfung, in der eine der Variablen negiert vorkommt. Hat die negierte Variable den Wahrheitswert 1, so hat die Funktion den Wahrheitswert 0. Diese Belegung verhindert also, dass die andere (nicht negierte) Variable den Wahrheitswert der Funktion beeinflusst.

Implikation
Die Implikation wird üblicherweise als "Wenn – So"-Verknüpfung beschrieben. Beispiel:

$\bar{a} \vee b = \overline{a \cdot \bar{b}}$ = wenn a, so b.

Auch deren Interpretation ist etwas spitzfindig. Die Implikation ist nur dann falsch, wenn das Vorderglied a wahr ist und das Hinterglied b falsch, wenn also aus Wahrem auf Falsches geschlossen wird[4]. Ist das Vorderglied falsch, so ist die Implikation immer wahr; auf das Hinterglied kommt es dann gar nicht an. Demgemäß gelten Aussageverknüpfungen wie "Wenn der Mond aus Käse besteht, so ist es am Tage finster." als wahr. Die Implikation ist aber für das formale Schließen (z. B. zum Beweisen mathematischer Sätze) von entscheidender Bedeutung.

Wichtige elementare Verknüpfungen

a	b	Negation*)	UND	ODER	NAND	NOR	Antivalenz	Äquivalenz
0	0	1	0	0	1	1	0	1
0	1	1	0	1	1	0	1	0
1	0	0	0	1	1	0	1	0
1	1	0	1	1	0	0	0	1

*): Nur Variable a.

Tabelle 1.3 Wahrheitstabellen wichtiger elementarer Verknüpfungen.

Elementare Verknüpfungen von mehr als zwei Variablen
Es liegt nahe, elementare Verknüpfungen auf mehr als zwei Variable zu erweitern. Manche Erweiterungen sind auf den ersten Blick sinnfällig, manche sind reine Definitionssache.

UND
Damit mehrere mit UND verbundene Aussagen insgesamt eine wahre Aussage ergeben, müssen offensichtlich alle Teilaussagen wahr sein (Beispiel: "der Angeklagte war am Tatort UND er war bewaffnet UND er hatte ein Tatmotiv"). Eine mehrstellige UND-Verknüpfung entspricht einer Hintereinanderausführung mehrerer zweistelliger UND-Verknüpfungen.

$$a \cdot b \cdot c = (a \cdot b) \cdot c \qquad (1.7)$$

Eine UND-Verknüpfung nimmt nur dann den Funktionswert 1 an, wenn alle Variablen den Wahrheitswert 1 haben. Der Funktionswert 0 ergibt sich bereits dann, wenn nur eine der Variablen den Wahrheitswert 0 hat.

4: Beispiel: "Wenn eine quadratische Gleichung zwei Lösungen hat, so erwärmt das CO_2 die Erde."

ODER (Disjunktion, OR)
Mehrere mit ODER verbundene Aussagen ergeben offensichtlich schon dann eine wahre Aussage, wenn wenigstens eine der Teilaussagen wahr ist (Beispiel: "der Schaltkreis ist defekt ODER die Spannung ist zu niedrig ODER die Belastung ist zu hoch"). Eine mehrstellige ODER-Verknüpfung entspricht einer Hintereinanderausführung mehrerer zweistelliger ODER-Verknüpfungen.

$$a \vee b \vee c = (a \vee b) \vee c \qquad (1.8)$$

Eine ODER-Verknüpfung nimmt dann den Funktionswert 1 an, wenn wenigstens eine der Variablen den Wahrheitswert 1 hat. Damit sich der Funktionswert 0 ergibt, müssen alle Variable den Wahrheitswert 0 haben.

a	b	c	UND	ODER	NAND	NOR	Antivalenz	Äquivalenz*
0	0	0	0	0	1	1	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	0	1	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	0	1	1	0	0	0
1	1	0	0	1	1	0	0	0
1	1	1	1	1	0	0	1	1

*: Definition gemäß (1.12). Vgl. auch (1.13).

Tabelle 1.4 Wahrheitstabellen dreistelliger Verknüpfungen.

Assoziativität
Dieser Begriff aus der Mathematik bezeichnet die Tatsache, dass man Verknüpfungen mehrerer Variablen auf die Hintereinanderausführung von Verknüpfungen mit weniger Variablen (im Grenzfall mit zwei Variablen) zurückführen kann. Die Verknüpfung der ersten beiden Variablen ergibt ein Zwischenergebnis, das mit der nächsten Variablen verknüpft wird usw. So kann man eine mehrstellige Verknüpfung auf eine Folge von zweistelligen zurückführen:

$$a \ \mathbf{op}\ b \ \mathbf{op}\ c = (a \ \mathbf{op}\ b) \ \mathbf{op}\ c \qquad (1.9)$$

UND und ODER sind offensichtlich assoziativ (vgl. (1.7) und (1.8)). Diese Eigenschaft gilt aber nicht für alle mehrstelligen Verknüpfungen.

a	b	c	d	UND	ODER	NAND	NOR	Antivalenz	Äquivalenz
0	0	0	0	0	0	1	1	0	1
0	0	0	1	0	1	1	0	1	0
0	0	1	0	0	1	1	0	1	0
0	0	1	1	0	1	1	0	0	1
0	1	0	0	0	1	1	0	1	0
0	1	0	1	0	1	1	0	0	1
0	1	1	0	0	1	1	0	0	1
0	1	1	1	0	1	1	0	1	0
1	0	0	0	0	1	1	0	1	0
1	0	0	1	0	1	1	0	0	1
1	0	1	0	0	1	1	0	0	1
1	0	1	1	0	1	1	0	1	0
1	1	0	0	0	1	1	0	0	1
1	1	0	1	0	1	1	0	1	0
1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	1	0	0	0	1

Tabelle 1.5 Wahrheitstabellen vierstelliger Verknüpfungen.

NAND

Eine mehrstellige NAND-Verknüpfung ist als Negation einer mehrstelligen UND-Verknüpfung definiert (NOT-AND). Sie nimmt nur dann den Funktionswert 0 an, wenn alle Variablen den Funktionswert 1 haben. Der Funktionswert 1 ergibt sich bereits dann, wenn nur eine der Variablen den Funktionswert 0 hat.

NOR

Eine mehrstellige NOR-Verknüpfung ist als Negation einer mehrstelligen ODER-Verknüpfung definiert (NOT-OR). Sie nimmt dann den Funktionswert 0 an, wenn wenigstens eine der Variablen den Funktionswert 1 hat. Damit sich der Funktionswert 1 ergibt, müssen alle Variable den Funktionswert 0 haben.

Mehrstellige NAND- und NOR-Verknüpfungen sind nicht assoziativ, können also nicht durch Hintereinanderausführen zweistelliger NAND- und NOR-Verknüpfungen oder durch Hintereinanderschalten von NAND-Gattern mit zwei Eingängen dargestellt werden.

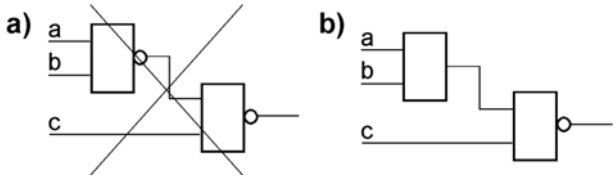


Abb. 1.13 NAND und NOR sind nicht assoziativ, sondern negierte mehrstellige UND- und ODER-Verknüpfungen. Also erst die jeweilige Grundverknüpfung (UND, ODER) ausführen, dann das Endergebnis invertieren. a) falsch; b) richtig.

Antivalenz (Exklusiv-ODER, XOR)
Als Aussagenverknüpfung ist ein ausschließendes ODER dem umgangssprachlichen Sinne nach nur dann wahr, wenn eine und nur eine der verknüpften Aussagen wahr ist (ENTWEDER a ODER b ODER c usw.). Die mehrstellige Antivalenzverknüpfung (XOR) ist jedoch nicht so, sondern als fortlaufende Addition modulo 2 definiert[5]:

- 1 + 1 + 1 = 3; 3 : 2 ergibt Rest 1. Allgemein: das Aufeinanderaddieren einer ungeraden Anzahl von Einsen ergibt eine ungerade Zahl, die bei Division durch 2 einen Rest = 1 hinterläßt.
- 1 + 1 + 1 + 1 = 4; 4 : 2 ergibt Rest 0. 1. Allgemein: das Aufeinanderaddieren einer geraden Anzahl von Einsen ergibt eine gerade Zahl, die ohne Rest durch 2 teilbar ist.

Der Funktionswert ergibt sich also folgendermaßen:

- 0, wenn die Anzahl der Einsen gerade ist,
- 1, wenn die Anzahl der Einsen ungerade ist.

Die mehrstellige Antivalenzverknüpfung ist assoziativ; sie entspricht einer Hintereinanderausführung mehrerer zweistelliger Antivalenzverknüpfungen:

$$a \oplus b \oplus c = (a \oplus b) \oplus c \tag{1.10}$$

Äquivalenz (Exklusiv-NOR, XNOR)
Für die mehrstellige Äquivalenzverknüpfung (XNOR) gibt es zwei Definitionen:

- Die Negation der XOR-Verknüpfung (XNOR = invertiertes XOR):

$$a \otimes b \otimes c = \overline{a \oplus b \oplus c} \tag{1.11}$$

5: "Modulo x" bedeutet den Rest beim Dividieren durch x. Eine ungerade Zahl durch 2 dividiert ergibt einen Rest = 1. Dividieren wir eine gerade Zahl durch 2, so geht es ohne Rest auf (Rest = 0).

- Die assoziative Äquivalenzverknüpfung, die einer Hintereinanderausführung mehrerer zweistelliger Äquivalenzverknüpfungen entspricht:

$$a \otimes b \otimes c = (a \otimes b) \otimes c \tag{1.12}$$

Der Funktionswert der assoziativen Verknüpfung ergibt sich folgendermaßen:

- 0, wenn die Anzahl der Nullen ungerade ist,
- 1, wenn die Anzahl der Nullen gerade ist.

Ist die Anzahl der Variablen ungerade, so muss, wenn sich der Funktionswert Eins ergibt, die Anzahl der Einsen ungerade sein. Die Äquivalenz hat dann den gleichen Verlauf der Funktionswerte wie die Antivalenz:

$$a \otimes b \otimes c = a \oplus b \oplus c \tag{1.13}$$

Ist die Anzahl der Variablen gerade, so muss, wenn sich der Funktionswert Eins ergibt, die Anzahl der Einsen gerade sein. Die Äquivalenz ist dann die Negation der Antivalenz:

$$a \otimes b \otimes c \otimes d = \overline{a \oplus b \oplus c \oplus d} \tag{1.14}$$

Boolesche Funktionen und zweiwertige Schaltungen

Die folgenden Abbildungen sollen die wechselseitige Überführbarkeit von Aussagefunktion und Schaltung veranschaulichen. Die Wirkungsweise der gezeigten Schaltungen kann durch elementare Aussagefunktionen oder Boolesche Funktionen beschrieben werden.

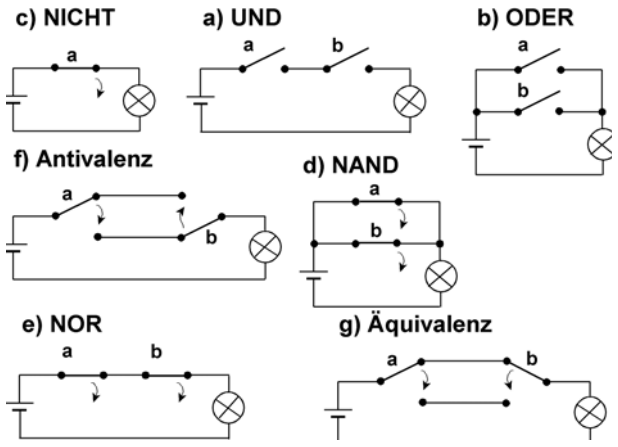


Abb. 1.14 Kontaktschaltungen als technische Entsprechung elementarer Boolescher Funktionen (Auswahl). Nicht betätigter Kontakt = nicht leuchtende Lampe = Wahrheitswert Falsch = 0; betätigter Kontakt = leuchtende Lampe = Wahrheitswert Wahr = 1. Alle Stromkreise sind im Ruhezustand dargestellt (alle Kontakte unbetätigt = Wahrheitswert Falsch = 0).

a) Konjunktion (UND-Verknüpfung). Um den Stromkreis zu schließen, sind beide Kontakte zu betätigen. Die Lampe leuchtet, wenn Kontakt a UND Kontakt b betätigt sind.

b) Disjunktion (ODER-Verknüpfung). Um den Stromkreis zu schließen, genügt es, einen der beiden Kontakte zu betätigen. Die Lampe leuchtet, wenn Kontakt a ODER Kontakt b oder beide Kontakte betätigt sind.

c) Negation (NICHT-Funktion). Der Stromkreis ist im Ruhezustand geschlossen; die Lampe leuchtet. Wird der Kontakt betätigt, so wird der Stromkreis aufgetrennt, und die Lampe verlischt (Ruhekontakt).

d) NAND-Verknüpfung. Um den Stromkreis zu trennen (so dass die Lampe verlischt), sind beide Kontakte (A UND B) zu betätigen.

e) NOR-Verknüpfung. Um den Stromkreis zu trennen (so dass die Lampe verlischt), genügt es, einen der beiden Kontakte (A ODER B) zu betätigen.

f) Antivalenz. Der Stromkreis wird nur dann geschlossen, wenn entweder nur der Kontakt a oder nur der Kontakt b betätigt wird. Die Lampe leuchtet, wenn Kontakt a betätigt ist und Kontakt b nicht ODER wenn Kontakt b betätigt ist und Kontakt a nicht.

g) Äquivalenz. Der Stromkreis ist immer dann geschlossen, wenn beide Kontakte zugleich betätigt oder nicht betätigt sind.

Die folgenden Abbildungen zeigen sinngemäß wirkende elektronische Schaltelemente. Solche elementaren Verknüpfungsschaltungen werden als Gatter (Gates) bezeichnet[6]. Die Wahrheitswerte werden durch unterschiedliche Signalpegel dargestellt. Für die folgenden Erläuterungen gilt: Signalpegel nahe 0 V (Massepotential) = Wahrheitswert Falsch = 0; Signalpegel nahe Betriebsspannung U_B = Wahrheitswert Wahr = 1.

Konjunktion (UND-Verknüpfung)
Liegen beide Eingänge E1, E1 auf 1-Pegel, so sind beide Dioden in Sperrichtung gepolt. Am Ausgang A ergibt sich über den Widerstand ein 1-Pegel. Wird einer der Eingänge mit einem 0-Pegel belegt , so liegt die jeweilige Diode in Flussrichtung. Über ihr fällt dann nur eine geringe Spannung ab (Flussspannung). Infolgedessen ergibt sich auch am Ausgang ein 0-Pegel.

Disjunktion (ODER-Verknüpfung)
Eigentlich müßte es genügen, alle Eingänge miteinander zu verbinden (“Lötstelle”). Hierdurch würde am Ausgang ein 1-Pegel auftreten, sofern auch nur einer der Eingänge mit einem1-Pegel belegt ist. Bei dieser Einfachlösung würden aber Rückströme von den mit 1 belegten Eingängen in die mit 0 belegten Eingänge fließen. Die Dioden dienen dazu, dies zu verhindern (Rückstromsperre).

6: Der Begriff leitet sich von der sog. Torschaltung her (Gate = Tor). Eine einfache Torschaltung hat einen Signaleingang, einen Signalausgang und einen Steuereingang. Sie leitet das Eingangssignal nur dann zum Ausgang weiter, wenn das Steuersignal aktiv ist.

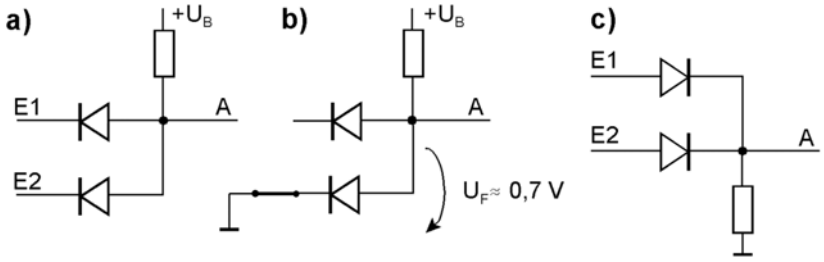


Abb. 1.15 Konjunktion und Disjunktion mit Dioden. a) UND-Gatter allgemein; b) UND-Gatter mit einem Eingang auf 0-Pegel; c) ODER-Gatter.

Negation (Invertierung, NICHT-Funktion)

Die Negation erfordert aktive Bauelemente, z. B. Transistoren. Eine Transistorschaltstufe wirkt ähnlich wie ein Relais mit Arbeitskontakt. Ist der Eingang E mit einem 0-Pegel belegt, so ist der Schalter geöffnet. Am Ausgang A ergibt sich dann über den Widerstand ein 1-Pegel. Ein 1-Pegel am Eingang E führt hingegen zum Schließen des Schalters (die Kollektor-Emitter-Strecke des Transistors wird leitend), so dass der Ausgang zum Massepotential hin durchgeschaltet und somit auf 0-Pegel gebracht wird.

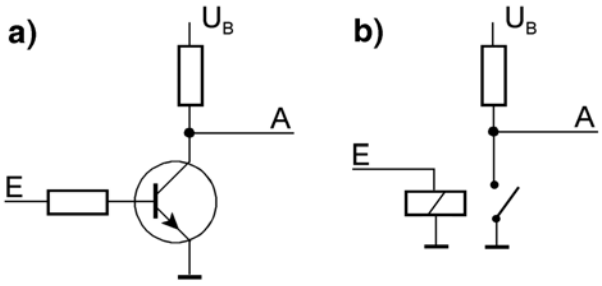


Abb. 1.16 Negation. a) Transistorschaltstufe (Negator, Inverter); b) Ersatzschaltung mit Relais und Arbeitskontakt (zur Funktionserklärung).

NAND

Sind zwei Transistorstufen in Reihe geschaltet, so kann der Ausgang A nur dann einen 0-Pegel führen ($\overline{A}$), wenn beide Transistoren leitend sind. Hierzu müssen alle Eingänge E1, E2 mit 1-Pegel belegt sein: $\overline{A} = E1 \cdot E2$; also $A = \overline{E1 \cdot E2}$.

NOR

Sind zwei Transistorstufen parallelgeschaltet, so führt der Ausgang A bereits dann einen 0-Pegel ($\overline{A}$), wenn nur einer der Transistoren leitend ist. Es genügt also die Erregung eines einzigen Eingangs: $\overline{A} = E1 \vee E2$; also $A = \overline{E1 \vee E2}$.

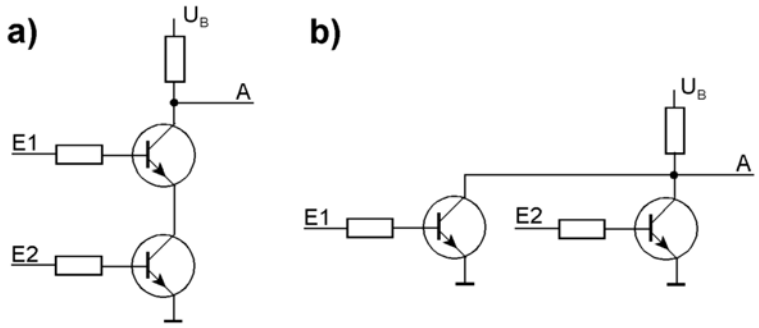


Abb. 1.17 Gatter mit Transistorschaltstufen. a) NAND; b) NOR.

Es ist offensichtlich, dass sich die Schaltungen auf mehr als zwei Eingänge erweitern lassen.

1.5 Einführung in die Schaltalgebra

1.5.1 Boolesche Räume

Binärvektoren

Ein Binärvektor ist nichts anderes als eine geordnete Reihe binärer Werte, wobei sich die Ordnung auf eine jeweils gegebene Folge Boolescher Variabler bezieht.

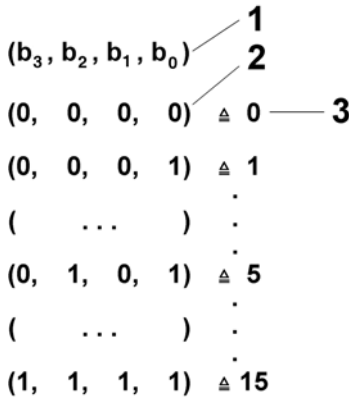


Abb. 1.18 Der Binärvektor. 1 - Anordnung der Booleschen Variablen; 2 - Belegungen mit verschiedenen Werten; 3 - Dezimaläquivalent.

Ist die Variablenreihenfolge bekannt, genügt es, bestimmte Belegungen einfach durch Folgen der zugeordneten Nullen und Einsen anzugeben. Diese Bitmuster kann man auch anders darstellen, z. B. hexadezimal oder als Dezimaläquivalent. Die Zahlenwerte 5 und 13 drücken dann beispielsweise aus, dassdie Variablen b_3, b_2, b_1, b_0 zum einen mit den Werten 0, 1, 0, 1 und zum anderen mit den Werten 1, 1, 0, 1 belegt sind. Offensichtlich kann jede Belegung von k Variablen als Binärvektor aus k Binärwerten dargestellt werden. Es gibt 2^k k-stellige Binärvektoren. Diese lassen sich in einem Booleschen Raum B^k anordnen. Jeder Binärvektor entspricht einem Punkt des Raumes. Ein BoolescherRaum B^k hat somit 2^k Raumpunkte. Die Anzahl k der Binärvariablen heißt auch Dimension des Booleschen Raums B^k.

Was unterscheidet den Booleschen Raum von der Liste aller 2^k Binärvektoren – oder, was das Gleiche bedeutet –, von einer gewöhnlichen Wahrheitstabelle? Zunächst eigentlich nur, dass solche Darstellungen dekorativer aussehen. Andererseits kann man sich beispielsweise einen Boolescher Raum der Dimension 10 (1024 Raumpunkte) wohl nur schwer vorstellen; ist doch schon bei Dimension 4 (16 Raumpunkte) die zeichnerische Darstellung kaum noch als überschaubar anzusprechen.Tatsächlich ist für den Schaltungsentwickler der Boolesche Raum zunächst nichts anderes als die Darstellung aller 2^k möglichen Belegungen von k Binärvariablen, also im Grunde auch nur eine Art Binärvektorliste oder Wahrheitstabelle. In der Theorie liegt die Bedeutung des Booleschen Raums B^k vor allem darin, dass er als metrischer Raum aufgefasst werden kann, dass es also möglich ist, Abstandsdefinitionen einzuführen. Das ist u. a. bei Untersuchungen zur Schaltungsminimierung sowie für Verfahren und Schaltungen zur Fehlererkennung und Fehlerkorrektur von Bedeutung.

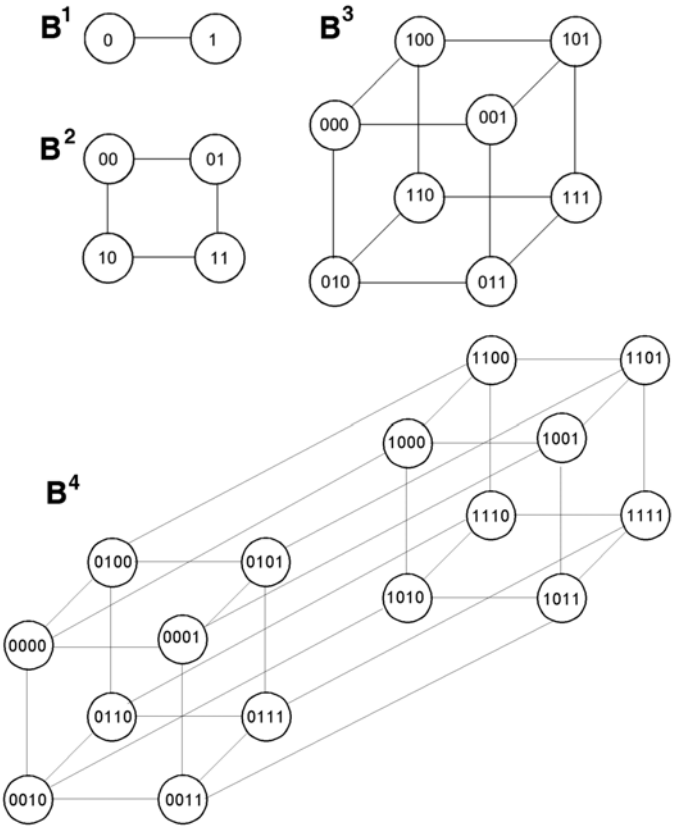


Abb. 1.19 Boolesche Räume mit 1, 2 , 3 und 4 Dimensionen. Die Kanten verbinden Raumpunkte, die voneinander den Abstand 1 haben.

Der Hamming-Abstand

Zwei Punkte des Raums haben voneinander einen Abstand n, wenn sich die zugehörigen Binärvektoren in n Variablenpositionen voneinander unterscheiden. Der Hamming-Abstand (Hamming Distance) d_H zweier Punkte x, y ergibt sich als Quersumme der bitweisen Antivalenzverknüpfung beider Binärvektoren:

$$d_H = \sum_i (x_i \oplus y_i) \tag{1.15}$$

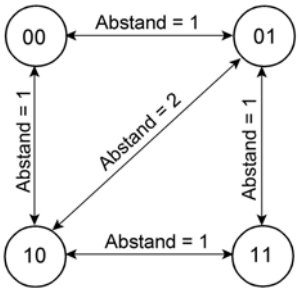


Abb. 1.20 Der Abstand im Booleschen Raum (Hamming-Abstand).

Boolesche Funktionen im Raum B^k

Bisher haben wir nur k Boolesche Variable betrachtet, denen 2^k verschiedene Belegungen aus Nullen und Einsen (also Bitmuster) zugeordnet werden können. Um eine bestimmte Boolesche Funktion von k Variablen anzugeben, ist es erforderlich, jeder dieser 2^k Belegungen entweder eine Null oder eine Eins zuzuweisen, also nicht nur die linken Spalten der Wahrheitstabelle (die vollständige Liste der Binärvektoren) hinzuschreiben, sondern auch die rechte Spalte (die zugewiesenen Funktionswerte) auszufüllen. Aus dieser Zuweisung ergeben sich zwei Mengen:

- Die Menge aller Punkte, denen der Funktionswert Null zugeordnet wird. Der typische Fachbegriff: Off-Set (im Folgenden: Null-Menge).
- Die Menge aller Punkte, denen der Funktionswert Eins zugeordnet wird. Der typische Fachbegriff: On-Set (im Folgenden: Eins-Menge).

Manchmal gibt es zudem eine Menge von Punkten, die für das betreffende Problem keine Bedeutung hat und somit außer Betracht bleiben kann (Don‘t-Care-Set; im folgenden: NB-Menge (NB = nicht beachten)).

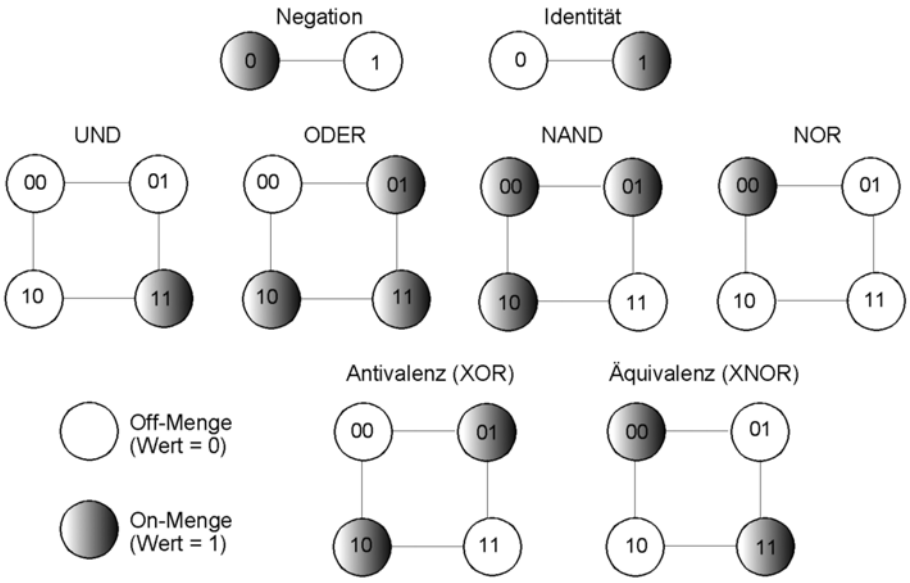


Abb. 1.21 Einige elementare Boolesche Funktionen im Booleschen Raum (B^1, B^2).

1.5.2 Darstellung von Schaltfunktionen

"Schaltfunktion" ist nur eine andere Bezeichnung für Boolesche Funktion. Jede Schaltfunktion bezieht sich auf bestimmte binäre Variable (Eingangsvariable). Sie ordnet jeder Belegung dieser Eingangsvariablen einen der Wahrheitswerte 0 oder 1 zu (Funktionswert). Diese Zuordnung kann u. a. in Form von Wahrheitstabellen, Belegungslisten, Schaltgleichungen, Schaltplänen und binären Entscheidungsdiagramme dargestellt werden.

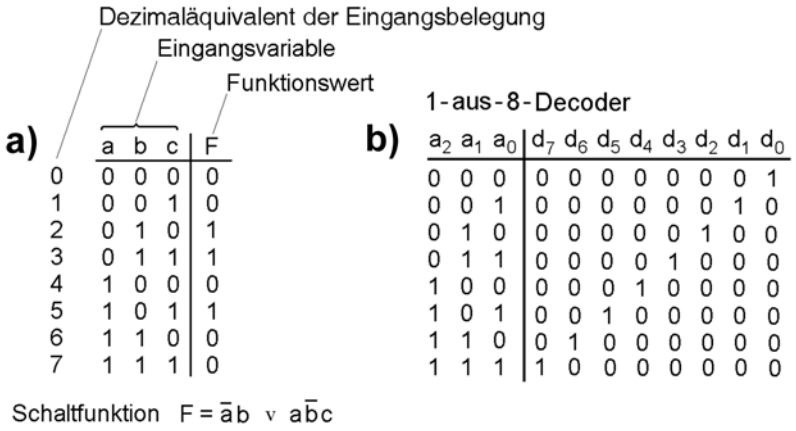
Variablenbezeichnungen

In der Theorie und beim Lernen ist Kürze von Vorteil. Deshalb werden Variable oft (so auch hier) mit einzelnen Buchstaben bezeichnet. In der Schaltungspraxis hat man es hingegen meistens mit vielen Variablen zu tun, und Eindeutigkeit ist das oberste Gebot. Anstelle von Buchstaben, wie a, b, usw. vergibt man deshalb Variablenbezeichnungen wie DATA7, ADRS31, CARRY_IN, ShiftLeft usw., also eine Kennzeichnung, welche Bedeutung die einzelne Variable hat[7], manchmal aber auch "sinnlose" Kombinationen aus Buchstaben, Ziffern und Sonderzeichen (dann handelt es sich meistens um Bezeichnungen, die von der Entwurfssoftware erzeugt wurden).

Wahrheitstabellen

In einer Wahrheitstabelle sind alle möglichen Kombinationen der Eingangsvariablen zusammen mit dem jeweiligen Funktionswert angegeben. Eine Wahrheitstabelle für n Eingangsvariable hat 2^n Einträge (Zeilen).

7: Vergleiche auch die Buchstabenrechnung der Mathematik und die Variablenbezeichner beim Programmieren.



	a	b	c	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

a_2	a_1	a_0	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Abb. 1.22 Wahrheitstabellen (Beispiele). a) grundsätzlicher Aufbau; b) Wahrheitstabelle mit mehreren Funktionen, die von den gleichen Eingangsvariablen abhängen.

Reihenfolge der Variablenbelegungen

Aus Gründen der Übersichtlichkeit sollen die Variablenbelegungen in regulärer Folge angegeben werden. Allgemein üblich ist die binäre Zählweise mit der Belegung 00...0 am Anfang und 11...1 am Ende, das heißt, das Bitmuster der Variablenbelegung in der i-ten Zeile (i von 1 bis 2^n) entspricht der Binärzahl i –1 bzw. deren Dezimaläquivalent. Wertebereich: von 0 bis 2^n – 1. Die folgende Abbildung zeigt, wie man die Belegungen der Eingangsvariablen schnell aufstellen kann: in die Spalte ganz rechts kommt die Ziffernfolge 0 – 1 – 0 – 1..., in die Spalte links daneben die Ziffernfolge 0 – 0 – 1 – 1– 0 – 0... in die nächste Spalte wiederum links daneben die Ziffernfolge 0 – 0 – 0 – 0 – 1 – 1 –1 – 1... usw.

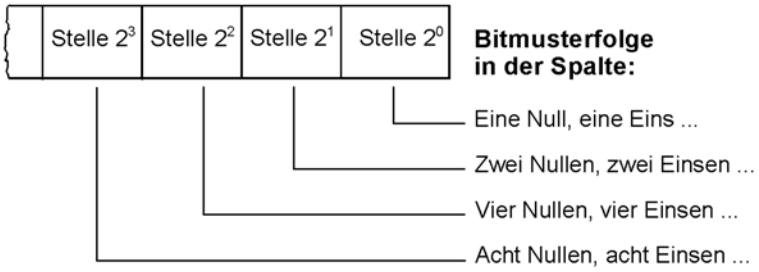


Abb. 1.23 Zum Aufstellen der Variablenbelegungen in Wahrheitstabellen.

Der Karnaugh-Plan

Der Karnaugh-Plan (auch Karnaugh-Veitch- oder KV-Diagramm) ist eine zweidimensional angelegte Wahrheitstabelle, wobei die Anordnung der Variablenbelegungen dem Gray-Code entspricht. In solchen Tabellen kann man Zusammenfassungs- und damit Vereinfachungsmöglichkeiten unmittelbar erkennen.

Binär- und Ternärvektorlisten

Die Wahrheitstabelle ist eine Listendarstellung aller 2^n Punkte des Booleschen Raums. Eine Alternative besteht darin, nur die Raumpunkte anzugeben, die zur Lösungsmenge gehören. Es werden nur jene Binärvektoren in eine Liste eingetragen, denen ein bestimmter Funktionswert (entweder 1 oder 0) zugeordnet ist. Eine solche Liste (Belegungsliste) ist im Grunde eine Aufzählung der Binärvektoren, die zur Eins-Menge oder zur Null-Menge gehören.

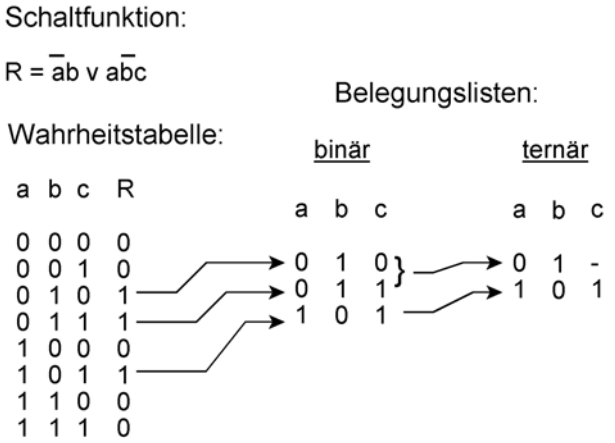


Abb. 1.24 Schaltfunktion, Wahrheitstabelle und Belegungslisten.Schaltfunktion und Wahrheitstabelle entsprechen Abb. 1.22a.

Es gibt zwei Arten solcher Listen:

- Die binäre (zweiwertige) Belegungsliste (Binärvektorliste). Sie entspricht unmittelbar dem Auszug aus der Wahrheitstabelle. Ihre Einträge können somit nur die Werte 0 und 1 annehmen.
- Die ternäre (dreiwertige) Belegungsliste (Ternärvektorliste). Jeder Eintrag kann einen von drei möglichen Werten annehmen: 0, 1 und – (“Strichelement”).

Das Strichelement

Der Strich (–; gelegentlich auch ein **x** oder *) steht als Abkürzung für die beiden binären Werte 0 und 1. Die beiden Belegungen (0 1 0) und (0 1 1) unterscheiden sich offensichtlich nur in einer Variablenposition (nämlich in c). Sie können somit zu (0 1 –) zusammengefasst werden. Eine ternäre Variablenbelegung mit n Strichelementen entspricht 2^n binären Variablenbelegungen.

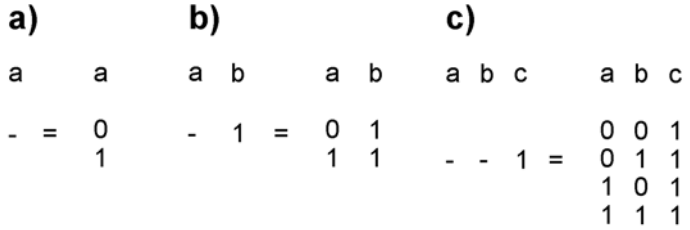


Abb. 1.25 Zur Bedeutung des Strichelements.

a) Nur eine Variable (a): Das Strichelement steht für beide Belegungen 0, 1.

b) Zwei Variable (a, b), ein Strichelement: Der ternäre Eintrag (– 1) steht für zwei binäre Einträge (0 1) und (1 1).

c) Drei Variable (a, b, c), zwei Strichelemente: Da jedes Strichelement für beide Belegungen einer Variablen steht, ergeben sich bei zwei mit Strichelementen belegtenVariablen $2^2 = 4$ binäre Einträge: (– – 1) = (0 0 1), (0 1 1), (1 0 1), (1 1 1).

Die naheliegende (aber nicht immer korrekte) Interpretation: eine mit einem Strich belegte Variable ist in der betreffenden Belegung bedeutungslos; sie muss deshalb gar nicht weiter beachtet werden (Don't Care). Anwendung: zum Vereinfachen (Minimieren) von Schaltfunktionen – eine Variable, die bedeutungslos ist, muss man auch bei der technischen Realisierung der Schaltfunktion nicht berücksichtigen.

Formelausdrücke und Schaltgleichungen (Boolesche Gleichungen)
In Formelausdrücken werden die Variablenbezeichner mit Symbolen verknüpft, die die elementaren logischen Funktionen repräsentieren.

Symbole in Theorie und Praxis
In der Schaltalgebra bevorzugt man eine der Mathematik ähnliche Symbolik:

- UND = • oder $\wedge$. Oft wird – in Analogie zur Multiplikation in der Arithmetik – das Symbol weggelassen: $a \cdot b = ab$. In dieser Schreibweise hat die Konjunktion Vorrang, sofern nicht durch Klammern eine andere Reihenfolge zum Ausdruck gebracht wird:

$$abc \vee def = (a \cdot b \cdot c) \vee (d \cdot e \cdot f)$$

- ODER = $\vee$ oder +.
- NICHT = Überstreichung. Eine lückenlose Überstreichung mehrerer Symbole entspricht einer Einklammerung:

$$\overline{a \vee b \vee c} = \neg(a \vee b \vee c)$$

Funktion	Symbole	Beispiele
Konjunktion (UND, AND)	&	a & b
	*	a * b
	·	a · b
	$\wedge$	a $\wedge$ b
	kein Symbol	ab

Funktion	Symbole	Beispiele
Disjunktion (ODER, OR)	v	a ∨ b
	+	a + b
	#	a # b
Antivalenz (Ungleichheit, Exclusiv-ODER, XOR)	⊕	a ⊕ b
	↚	a ↚ b
Äquivalenz (Gleichheit, XNOR)	≡	a ≡ b
	⇔	a ⇔ b
	⊙	a ⊙ b
Negation (NICHT, Invertierung, NOT)	/	/a; /(a ∨ b)
	¬	¬ a
	'	a'; (a ∨ b)'
	–	$\bar{a}$; $\overline{a \vee b}$
	!	!a; !(a ∨ b)
	#	a#

Tabelle 1.6 Symbole (Verknüpfungsoperatoren) in Formelausdrücken.

Schaltpläne

Schaltpläne sind zeichnerische Darstellungen, aus denen die Struktur der Schaltung hervorgeht. Negatoren (Inverter) und Gatter sind die einfachsten Schaltelemente. Negatoren verwirklichen die NICHT-Funktion, Gatter verwirklichen elementare aussagenlogische Verknüpfungen. Die einfachsten Gatter haben zwei Eingänge und einen Ausgang. Zusammengesetzte Schaltungen entstehen, indem solche Schaltsymbole ein- und ausgangsseitig so verbunden werden, wie dies die jeweilige Schaltfunktion erfordert. Jedem Verknüpfungsoperator entspricht ein Gatter, jeder Negation ein Negator oder Negationssymbol (Kreis), jeder Variablen eine Signalleitung.

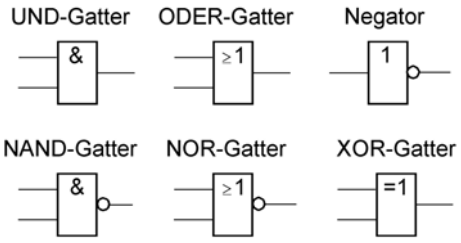


Abb. 1.26 Elementare Schaltsymbole. Zu alternativen Schaltsymbolen siehe den Anhang.

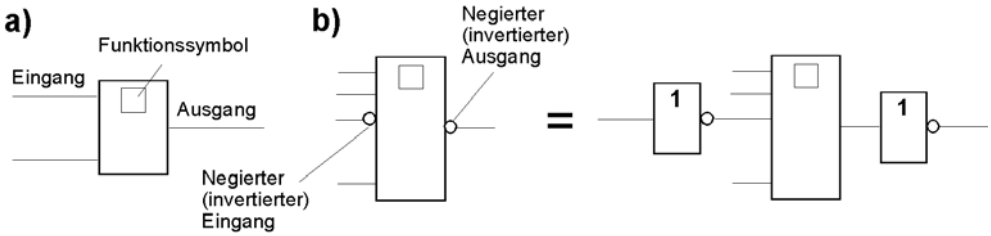


Abb. 1.27 Zur Gestaltung elementarer Schaltsymbole. a) die allgemeine Form; b) zur Symbolik der negierten (invertierten) Ein- und Ausgänge.

Von der Schaltgleichung zum Schaltplan:

- Jeder Operator mit zwei Operanden (a **op** b) ist ein Gatter des Typs **op**. Mehrere (n) gleichartige Operatoren hintereinander (a **op** b **op** c) sind ein Gatter mit n + 1 Eingängen.
- Steht eine Variable unmittelbar an einem Operator (a **op** b), so wird sie direkt an einen Eingang des Gatter angeschlossen.
- Steht an einem Operator keine einfache Variable, sondern ein Ausdruck ((a **op1** b) **op2** c), so wird der Ausgang des Gatters **op1**, der diesem Ausdruck entspricht, an einen Eingang des Gatters angeschlossen, der dem Operator **op2** entspricht.

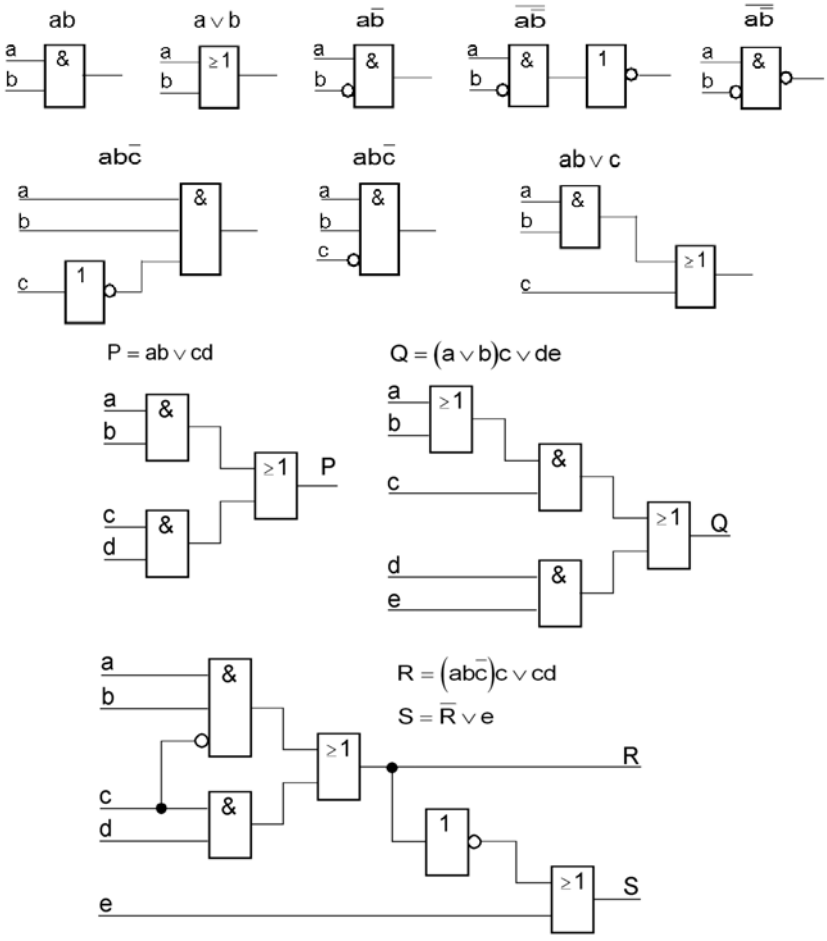


Abb. 1.28 Beispiele einfacher Schaltpläne.

Binäre Entscheidungsdiagramme

Ein binäres Entscheidungsdiagramm (Binary Decision Diagram BDD) ist ein gerichteter Graph, dessen Knoten Variable und dessen Kanten Variablenwerte repräsentieren. Diese Art der Darstellung wird ausschließlich für rechentechnische Zwecke (beispielsweise in Schaltungsentwicklungsprogrammen) und in der Grundlagenforschung eingesetzt; für das Hand- oder Kopfrechnen ist sie nicht geeignet.

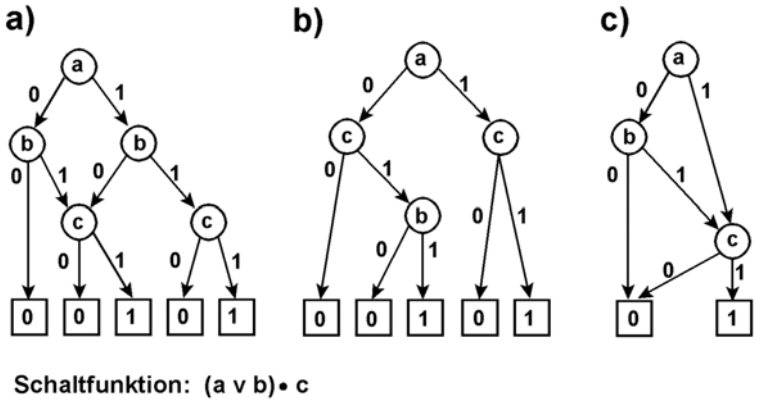


Abb. 1.29 Binäre Entscheidungsdiagramme (BDDs). a) Variablenreihenfolge a, b, c; b) Variablenreihenfolge a , c, b; c) reduziertes Diagramm (ROBDD) mit Variablenreihenfolge a, b, c.

Alle drei Entscheidungsdiagramme beschreiben die Schaltfunktion $(a \vee b) \cdot c$.

Ein solches Diagramm ermöglicht es, den Funktionswert aus den Belegungen der einzelnen Variablen zu bestimmen. Es entspricht einer Art Programmablaufplan, in dem jeder Knoten die Abfrage einer Variablen mit Verzweigung in zwei Richtungen kennzeichnet. Beispiel: wenn a = 0, so frage b ab; wenn b = 0, so ist der Funktionswert = 1, wenn b = 1, so frage c ab usw. Die Struktur eines solchen Diagramms hängt davon ab, in welcher Reihenfolge die Variablen abgefragt werden. Man spricht deshalb von geordneten Entscheidungsdiagrammen (Ordered Binary Decision Diagrams OBDDs). Für jede Variablenreihenfolge kann ein reduziertes geordnetes Entscheidungsdiagramm (Reduced Ordered Binary Decision Diagram ROBDD) angegeben werden, in dem alle überflüssigen Knoten beseitigt sind.

Anwendung:

- Zur rechnerinternen Darstellung von Schaltfunktionen (in Form von Listen- oder Zeigerstrukturen).
- Zur technischen Verwirklichung von Schaltfunktionen. Der Vorteil: alle Schaltfunktionen können auf Grundlage einer einzigen Elementarfunktion aufgebaut werden, nämlich der 1-aus-2-Auswahl (in Abhängigkeit von einem Auswahlsignal wird eines von zwei Eingangssignalen zum Ausgang durchgeschaltet). In der Grundlagenforschung arbeitet man daran, physikalische Effekte auszunutzen, um diese Elementarfunktion zu verwirklichen. Aber auch in herkömmlichen programmierbaren Schaltkreisen wird dieses Prinzip angewendet.

1.5.3 Boolesche Gleichungen

Jede Gleichung hat zwei Seiten. Das Gleichheitssysmbol drückt aus, dass beide Seiten einander gleich sind.

Die allgemeine Form einer Booleschen Gleichung

Zwei Boolesche Funktionen $f(\underline{x});\ g(\underline{y})$ werden einander gleichgesetzt:

$$f(\underline{x}) = g(\underline{y}) \tag{1.16}$$

Hierbei stehen $\underline{x}$ und $\underline{y}$ für beliebige Tupel Boolescher Variabler:

$$\underline{x} = \{x_1, x_2, \ldots x_n\}\ ;\quad \underline{y} = \{y_1, y_2, \ldots y_m\}$$

Mit Booleschen Gleichungen kann man ebenso umgehen wie mit den üblichen algebraischen Gleichungen: die Gleichung wird nicht verändert, wenn man auf beide Seiten die gleiche Operation anwendet.

Homogenen Gleichungen als Standardformen

Alle Booleschen Gleichungen können so umgeformt werden, dass sich auf der einen Seite lediglich eine Konstante befindet (homogene Gleichung). Da es nur zwei konstante Werte gibt, ergeben sich zwei Standardformen:

$$f(\underline{x}) = 0 \tag{1.17}$$

oder

$$g(\underline{x}) = 1 \tag{1.18}$$

Jede beliebige Boolesche Gleichung ((1.16)) läßt sich durch Antivalenzverknüpfung in eine Gleichung der Form (1.17) umwandeln (Homogenisierung). Hierzu wird auf beiden Seiten von (1.16) die gleiche Antivalenzverknüpfung ausgeführt:

$$f(\underline{x}) \oplus g(\underline{y}) = g(\underline{y}) \oplus g(\underline{y}) \tag{1.19}$$

Es gilt aber $g(\underline{y}) \oplus g(\underline{y}) = 0$. Also folgt:

$$f(\underline{x}) \oplus g(\underline{y}) = 0 \tag{1.20}$$

Lösungen

Eine Variablenbelegung, bei deren Einsetzen in die Boolesche Gleichung die Bedingung der Gleichheit erfüllt ist, ist eine Lösung der Booleschen Gleichung. Die Menge aller Punkte des Booleschen Raums B^n, die Lösungen der Booleschen Gleichung sind, heißt Lösungsmenge.

- Die Lösungsmenge einer Booleschen Gleichung $f(\underline{x}) = 0$ entspricht der Null-Menge der Booleschen Funktion $f(\underline{x})$.
- Die Lösungsmenge einer Booleschen Gleichung $g(\underline{x}) = 1$ entspricht der Eins-Menge der Booleschen Funktion $g(\underline{x})$.

Grundsätzlich gibt es folgende Möglichkeiten:

a) Keine einzige der 2^n Variablenbelegungen ist eine Lösung der Booleschen Gleichung (Kontradiktion). Im Falle von (1.17) ist die Null-Menge leer ($f(\underline{x})$ ergibt stets den Funktionswert Eins), im Falle von 1.18 die Eins-Menge ($g(\underline{x})$ ergibt stets den Funktionswert Null).
b) Alle 2^n Variablenbelegungen sind Lösungen der Booleschen Gleichung (Tautologie). Im Falle von (1.17) ist die Eins-Menge leer ($f(\underline{x})$ ergibt stets den Funktionswert Null), im Falle von (1.18) die Null-Menge ($g(\underline{x})$ ergibt stets den Funktionswert Eins).
c) Wenigstens eine Variablenbelegung ist eine Lösung der Boolesche Gleichung und wenigstens eine weitere Variablenbelegung ist keine Lösung.

In der Entwurfspraxis ist eine Boolesche Gleichung offensichtlich nur dann brauchbar, wenn sie "richtige" Lösungen (also gemäß Fall c)) hat. Eine solche Gleichung heißt “erfüllbar” (satisfiable). Die Aufgabe, zu erkennen, ob eine Boolesche Gleichung erfüllbar ist oder nicht, ist als SAT-Problem (SAT= Satisfiabilty) bekannt. Ist die Gleichung nicht erfüllbar, so ergibt sich als Lösung nur ein einziger Festwert. Dann kann man es sich ersparen, die Boolesche Funktion zu implementieren. Die Fälle a) und b) (Kontradiktion und Tautologie) sind in der formalen Logik von entscheidender Bedeutung, vor allem in logischen Beweisen. Ob beispielsweise eine Umformungsregel der Schaltalgebra zutrifft oder nicht, kann man überprüfen, indem man sie als Boolesche Gleichung in eine Standardform überführt und deren Lösungen ermittelt. Stellt sich eine Kontradiktion heraus, so liegt ein grundsätzlicher Widerspruch vor. Ergibt sich eine Tautologie, so ist die betreffende Umformungsregel als allgemeingültig nachgewiesen (weil sie dann für alle überhaupt möglichen Variablenbelegungen wahr ist).

Kontradiktion und Tautologie in der Entwurfspraxis

Solche Fälle können vorkommen. Es sind typischerweise Folgen von Entwurfsfehlern. Die Booleschen Gleichungen ergeben sich zunächst aus der Erfassung der Entwurfsaufgabe. Bereits hierbei können Fehler unterlaufen (irrtümliche Problemauffassung). Zudem kann man sich auf triviale Weise vertun (Tippfehler, Eingabefehler usw.) So kann es vorkommen, dass

man eine komplizierteGleichung formuliert oder eine verwickelte Schaltung entwirft, die aber in Wirklichkeit nicht mehr leistet, als beispielsweise den Festwert Eins auszugeben. Ein rechnerischer Nachweis – also die Lösung des SAT-Problems – gelingt allerdings nur in vergleichsweise einfachen Fällen (Rechenzeitbedarf).

Die Lösungsmenge einer Booleschen Gleichung bestimmen
Das Problem sieht einfach aus – alle 2^n Möglichkeiten durchprobieren und zusehen, was herauskommt. So könnte man auch erkennen, ob die Gleichung erfüllbar ist oder es sich um eine Kontradiktion oder um eine Tautologie handelt. Aber hier liegt auch die Schwierigkeit: der Rechenaufwand wächst offensichtlich exponentiell; das simple Durchprobieren hat eine Zeitkomplexität $O(2^n)$[8]. Es konnte zwar gezeigt werden, dass sich das SAT-Problem mit einer weniger als exponentiell wachsenden Zeitkomplexität bewältigen läßt. Trotzdem gehört das Bestimmen der Lösungsmengen Boolescher Gleichungen zu den härtesten Problemen der Rechentechnik.

Implizite Bejahung (Prinzip der Assertion)
In der Aussagenlogik bedeutet das bloße Hinschreiben oder Konstatieren einer Aussage im Grunde deren Bejahung. Die Aussage "der Mars ist ein Planet" besagt dasselbe wie die Aussage "die Aussage, dass der Mars ein Planet ist, ist wahr". In der Schaltalgebra verhält es sich sinngemäß; wenn man den Funktionsausdruck $a \cdot b$ hinschreibt, meint man damit eigentlich die Boolesche Gleichung $a \cdot b = 1$. Viele Boolesche Gleichungen werden von vornherein so aufgestellt, dass sich der Funktionswert 1 ergibt. Deshalb werden typischerweise Eins-Menge und Lösungsmenge als Synonyme gebraucht.

Boolesche Gleichungen und Schaltgleichungen
Eine Schaltgleichung ist eine Boolesche Gleichung, die das Verhalten einer binären ("logischen") Schaltung beschreibt. Eigentlich unterscheidet sie sich nicht von den Booleschen Gleichungen, wie sie in der formalen Aussagenlogik verwendet werden. Gelegentlich sind aber Spitzfindigkeiten zu beachten. Die Gleichung dient nicht dazu, eine Aussageverknüpfung als wahr zu kennzeichnen. Sie beschreibt vielmehr, dass bei Vorliegen bestimmter Wertekombinationen der Variablen ein bestimmter Funktionswert gebildet wird; mit anderen Worten: die Gleichung weist – wie beim Programmieren – einem Signal eine Belegung (Null oder Eins) zu. Die Form ist typischerweise nicht $f(\underline{x}) = 0$ oder $g(x) = 1$, sondern $\text{Signalbezeichner} = f(\underline{x})$. Ob das bezeichnete Signal bei Vorliegen der Wertekombinationen gleich Null oder gleich Eins sein soll, wird durch den Signalbezeichner ausgedrückt:

- $\text{Signalbezeichner} = f(\underline{x})$ bedeutet $f(\underline{x}) = 1$.
- $\overline{\text{Signalbezeichner}} = f(\underline{x})$ bedeutet $f(\underline{x}) = 0$.

8: Für eine Boolesche Gleichung mit vier Variablen wären 16 Wertekombintaionen durchzuprobieren, für acht Variable 256, für 16 Variable 65 536 usw.

In der Praxis wird die Negation zumeist mit einem Negationszeichen dargestellt. Die implizite Bejahung gilt bei Schaltgleichungen also nicht immer.

Praxisbeispiel:
!RAS# = (MEM_ACCESS * RAM_DECODE * RAS_PULSE) + (REFRESH_ACCESS * REFRESH_CYCLE)

Zweierlei Negationszeichen
Achtung – bei den Negationszeichen ist gelegentlich aufzupassen, denn es gibt zwei Interpretationen:

a) Es handelt sich lediglich um ein Negations*symbol* – also im Grunde nur um schmückendes Beiwerk (das die Funktion des Signals verdeutlichen soll).

b) Es handelt sich um einen Negations*operator*. Und der zeigt an, dass tatsächlich negiert (bzw. der Funktionswert Null zugewiesen) werden soll.

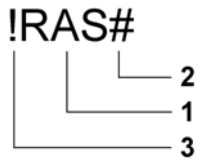


Abb. 1.30 Typische Negationsangaben. 1 - der eigentliche Signalname; 2 - Negationssymbol. Es besagt, dass das Signal aktiv (wirksam) ist, wenn es mit Null belegt ist. 3 - Negationsoperator. Weist an, die Schaltgleichung so zu implementieren, dass bei Erfüllung der Gleichung der Funktionswert Null gebildet wird.

Funktionswerte bestimmen

Gegeben sind eine Boolesche Funktion $f(\underline{x})$ und eine bestimmte Variablenbelegung mit Nullen und Einsen. Welcher Funktionswert ergibt sich? Wie der Funktionswert ermittelt wird, richtet sich danach, in welcher Form die betreffende Funktion dargestellt ist:

- Wahrheitstabelle. Es wird die Zeile aufgesucht, die der Variabenbelegung entspricht.
- Belegungsliste. Die Belegungsliste wird Zeile für Zeile mit der Variablenbelegung verglichen. Bei Gleichheit entspricht der Funktionswert dem Wert, für den die Belegungsliste aufgestellt ist, bei Ungleichheit dem invertierten Wert.
- Formelausdruck. Die Variablenbelegung wird in den Formelausdruck eingesetzt. Die Anwendung der jeweiligen Rechenregeln oder der Wahrheitstabellen der Verknüpfungsoperatoren ergibt den Funktionswert.
- Schaltplan. Die Variablenbelegung wird an die im Schaltplan dargestellten Schaltelemente angelegt. Stehen alle Eingangsbelegungen eines Schaltelements fest, kann dessen Ausgangsbelegung bestimmt werden. Am Ausgang des im Signalweg letzten Schaltelements ergibt sich der gesuchte Funktionswert.

- Binäres Entscheidungsdiagramm. Das Entscheidungsdiagramm wird gemäß den Werten der Variablenbelegung durchlaufen.

Probleme erfassen

Gegeben ist eine Entwurfsaufgabe. Aus binären Eingangssignalen (Variablen) sind binäre Ausgangssignale zu bilden. Diese Aufgabe soll mit Booleschen Gleichungen erledigt werden. Wie könnnen wir die umgangssprachlich oder in der Anschauung gegebene Aufgabenstellung in Boolesche Gleichungen umsetzen? Diese ursprüngliche Problemerfassung ist die wichtigste Phase im Entwurfsprozeß[9]. Alles weitere kann heutzutage maschinell erledigt werden. Zur Problemerfassung kommen vor allem die folgenden Darstellungsformen in Betracht:

- Wahrheitstabelle. Wir gehen alle überhaupt möglichen Variablenkombinationen daraufhin durch, ob jeweils eine Eins oder eine Null abgegeben werden soll. Das Verfahren hat den Vorteil, daß – hinreichende Gewissenhaftigkeit vorausgesetzt – wirklich alle Kombinationen betrachtet werden, so daß eigentlich nichts übersehen werden dürfte. Es ist aber nur für vergleichsweise wenige Variable durchführbar (von Hand bis ca. 6, bei maschineller Unterstützung bis ca. 10...12).
- Ternärvektorliste. Wir notieren alle Bitmuster, bei denen das Ausgangssignal aktiv werden soll, mit anderen Worten, für die die jeweilige Boolesche Gleichung erfüllt sein soll. Es wird typischerweise eine ternäre Liste sein, da man nicht immer alle Variaben betrachtet, sondern jeweils nur jene, die zum Ergebnis beitragen. Die jeweils anderen Variablen sind dann Don’t Cares.
- Boolesche Gleichung. Die ursprüngliche Booleschen Gleichungen ergeben sich, wenn man versucht, die Entwurfsaufgabe unter Nutzung von Verknüpfungen wie UND, ODER, NICHT, XOR usw. exakt zu beschreiben. Oftmals werden solche Gleichungen unstrukturiert, gleichsam wild aussehen (geschachtelte Klammerausdrücke, Negation von Ausdrücken anstatt von nur einzelnen Variablen usw.).
- Schaltbild. Manchmal (Übungssache) gelingt es, die Problemlösung sofort mit einem Schaltbild zu skizzieren. Es wird oftmals nicht gerade elegant aussehen und auch keine optimale Lösung darstellen.

1.5.4 Normalformen

Eine Normalform ist die Darstellung einer Booleschen Funktion, die nach bestimmten Regeln aufgebaut ist. Es gibt verschiedene Normalformen. Jede beliebige Boolesche Funktion ist in jeder Normalform darstellbar. Manche Normalformen sind eindeutig. Das heißt, für jede Boolesche Funktion gibt nur eine einzige nach den jeweiligen Regeln aufgebaute Normalformdarstellung (kanonische Normalformen). Andere Normalformen sind mehrdeutig; es gibt dann mehr als eine Möglichkeit, eine Boolesche Funktion gemäß den Regeln der

9: Fehler, die hierbei unterlaufen, kann oftmals auch die Simulation nicht finden. Das ist vor allem dann zu erwarten, wenn sowohl die Problemlösung als auch die Simulationsbeispiele auf der gleichen Grundlage – nämlich einer irrtümlichen Problemauffassung – ausgearbeitet wurden.

jeweiligen Normalform darzustellen. Die folgende Abbildung zeigt die Wahrheitstabelle einer Booleschen Funktion, die in den folgenden Erläuterungen als Beispiel verwendet wird. Die Einträge der üblichen Wahrheitstabellen kann man von Null an fortlaufend durchnumerieren. Bei bekannter Variablenzuordnung kann somit das Dezimaläquivalent die Angabe des Bitmusters oder des jeweils zugehörigen Booleschen Audrucks ersetzen.

dez.	a	b	c	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Abb. 1.31 Beispiel einer Booleschen Funktion. Dez. = Dezimaläquialent der Variablenbelegungen a, b, c.

Terme

Wir betrachten eine Boolesche Funktion, die von den Variablen x_1, x_2,...x_n abhängt. Ein Term (Fundamentalausdruck) ist eine konjunktive oder disjunktive Verknüpfung aller Variablen, wobei die Variablen entweder nicht negiert oder einzeln negiert oder gar nicht auftreten. Es gibt Konjunktionsterme (Minterme) und Disjunktionsterme (Maxterme). Die wichtigsten Normalformen bestehen aus Termen, die gemäß einer einzigen weiteren Booleschen Funktion miteinander verknüpft sind. Das allgemeine Schema einer solchen Normalform:

Term 1 **OP** Term 2 **OP** Term 3 ... (1.21)

Die Boolesche Funktion OP gibt der Normalform ihren Namen. So spricht man u. a. von disjunktiven und konjunktiven Normalformen.

Konjunktionsterm (Minterm):

$$K_i = x_1 \cdot x_2 \cdot \ldots \cdot x_n = x_1 x_2 \ldots x_n \qquad (1.22)$$

(x_1, x_2,...x_n einzeln negiert oder nicht negiert.)

Beispiel: $\overline{x_1} \cdot \overline{x_2} \cdot x_3$. Unzulässig: $\overline{x_1 \cdot x_2} \cdot x_3$.

Disjunktionsterm (Maxterm):

$$D_i = x_1 \vee x_2 \vee \ldots \vee x_n \qquad (1.23)$$

(x_1, x_2,...x_n einzeln negiert oder nicht negiert.)

Beispiel: $\overline{x_1} \vee \overline{x_2} \vee x_3$. Unzulässig: $\overline{x_1 \vee x_2} \vee x_3$.

Minterme und Maxterme
Die Bezeichnungen beziehen sich darauf, wieviele Terme den Funktionswert Eins aufweisen müssen, damit die gesamte – als Normalform ausgedrückte – Funktion den Wert Eins hat:

- Konjunktionsterme heißen auch Minterme, weil es in einer disjunktiven Normalform genügt, dass ein einziger Konjunktionsterm den Funktionswert Eins aufweist.
- Disjunktionsterme heißen auch Maxterme, weil es in einer konjunktiven Normalform erforderlich ist, dass alle Disjunktionsterme den Wert Eins aufweisen.

Implikanden
Das ist eine andere Bezeichnung für die Konjunktions- oder Disjunktionsterme.

Elementarkonjunktionen, Elementardisjunktionen
Das sind Implikanden, die *alle* Variablen (negiert oder nicht negiert) enthalten.

Die konjunktive Verknüpfung zweier Elementarkonjunktionen K_i, K_j ergibt stets den Funktionswert 0:

$$K_i \cdot K_j = 0 \quad \text{für alle i, j} \qquad (1.24)$$

Die disjunktive Verknüpfung zweier Elementardisjunktionen D_i, D_j ergibt stets den Funktionswert 1:

$$D_i \vee D_j = 1 \text{ für alle i, j} \qquad (1.25)$$

Kanonische Normalformen
Normalformen, die auf Konjunktions- und Disjunktionstermen beruhen, sind dann kanonisch, wenn jeder der Konjunktion- oder Disjunktionsterme alle Variablen (negiert oder nicht negiert) enthält, mit anderen Worten, wenn sie ausschließlich aus Elementarkonjunktionen oder aus Elementardisjunktionen aufgebaut sind.

Nichtkanonische (vereinfachte) Normalformen
In einer solchen Normalform ist wenigstens ein Implikand *keine* Elementarkonjunktion oder Elementardisjunktion; mit anderen Worten, in wenigstens einem der Terme fehlt wenigstes

eine Variable. Die kanonische Normalform entspricht einer binären, die vereinfachte Normalform einer ternären Belegungsliste (in der die Striche die fehlenden Variablen kennzeichnen).Alle Schaltfunktionen lassen sich als kanonische Normalformen darstellen, aber nicht alle Schaltfunktionen lassen sich vereinfachen. Bezeichnungen wie nichtkanonisch oder vereinfacht werden meistens weggelassen; man unterscheidet dann zwischen kanonischen und "gewöhnlichen" (nicht näher bezeichneten) Normalformen.

Bitmuster und Minterm

Die Variablenreihenfolge des Minterms entspricht der des Bitmusters in der Wahrheitstabelle. Die Variablen werden untereinander konjunktiv verknüpft. Ein Bitwert von Null entspricht einer negierten, ein Bitwert von Eins einer nicht negierten Variablen. In der Schaltung entspricht der Minterm einem UND-Gatter mit folgender Eingangsbelegung:

- Bitwert = 0 = Variable negiert = invertierter Anschluss.
- Bitwert = 1 = Variable nicht negiert = direkter Anschluss.

Bitmuster und Maxterm

Die Variablenreihenfolge des Maxterms entspricht der des Bitmusters in der Wahrheitstabelle. Die Variablen werden untereinander disjunktiv verknüpft. Ein Bitwert von Null entspricht einer nicht negierten, ein Bitwert von Eins einer negierten Variablen. In der Schaltung entspricht der Maxterm einem ODER-Gatter mit folgender Eingangsbelegung:

- Bitwert = 0 = Variable nicht negiert = direkter Anschluss,
- Bitwert = 1 = Variable negiert = invertierter Anschluss.

Darstellung durch das Dezimaläquivalent

Jede Elementarkonjunktion oder Elementardisjunktion entspricht einer Zeile der Wahrheitstabelle. Ist die Variablenreihenfolge festgelegt, kann somit anstelle des Booleschen Ausdrucks (1.22) oder (1.23) das Dezimaläquivalent der zugehörigen Belegung angegeben werden.

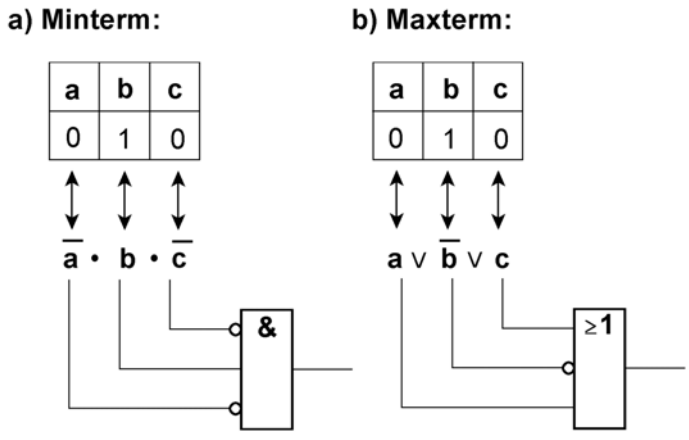


Abb. 1.32 Bitmuster und Terme.

dez.	a	b	c	Minterme	Maxterme
0	0	0	0	$m_0 = \bar{a} \cdot \bar{b} \cdot \bar{c}$	$M_0 = a \vee b \vee c$
1	0	0	1	$m_1 = \bar{a} \cdot \bar{b} \cdot c$	$M_1 = a \vee b \vee \bar{c}$
2	0	1	0	$m_2 = \bar{a} \cdot b \cdot \bar{c}$	$M_2 = a \vee \bar{b} \vee c$
3	0	1	1	$m_3 = \bar{a} \cdot b \cdot c$	$M_3 = a \vee \bar{b} \vee \bar{c}$
4	1	0	0	$m_4 = a \cdot \bar{b} \cdot \bar{c}$	$M_4 = \bar{a} \vee b \vee c$
5	1	0	1	$m_5 = a \cdot \bar{b} \cdot c$	$M_5 = \bar{a} \vee b \vee \bar{c}$
6	1	1	0	$m_6 = a \cdot b \cdot \bar{c}$	$M_6 = \bar{a} \vee \bar{b} \vee c$
7	1	1	1	$m_7 = a \cdot b \cdot c$	$M_7 = \bar{a} \vee \bar{b} \vee \bar{c}$

Abb. 1.33 Die Minterne und Maxterme einer Booleschen Funktion von drei Variablen.

Bei Mintermen entsprechen Bitmuster und Variable direkt (0 = negierte, 1 = nicht negierte Variable), bei Maxtermen invertiert (0 = nicht negierte, 1 = negierte Variable).

Beispiele (Bitmuster = 010):

- Minterm: $\bar{a} \cdot b \cdot \bar{c} = m_2$.
- Maxterm: $a \vee \bar{b} \vee c = M_2$.

Disjunktive Normalform

Eine disjunktive Normalform (DNF) besteht aus Konjunktionstermen (Mintermen) $K_1 ... K_n$, die disjunktiv verknüpft sind:

$$F = K_1 \vee K_2 \vee \ldots \vee K_n \qquad (1.26)$$

Eine kanonische disjunktiveNormalform (KDNF) liegt vor, wenn alle Konjunktionsterme Elementarkonjunktionen sind (mit anderen Worten, wenn in jedem Konjunktionsterm alle Variablen vorkommen). Beispiel:

$$F = \bar{a}b\bar{c} \vee \bar{a}bc \vee a\bar{b}c$$

Konjunktive Normalform

Eine konjunktive Normalform (KNF) besteht aus Disjunktionstermen (Maxtermen) $D_1 ... D_n$, die konjunktiv verknüpft sind:

$$F = D_1 \cdot D_2 \cdot \ldots \cdot D_n \qquad (1.27)$$

Eine kanonische konjunktive Normalform (KKNF) liegt vor, wenn alle Disjunktionsterme Elementardisjunktionen sind (mit anderen Worten, wenn in jedem Disjunktionsterm alle Variablen vorkommen). Beispiel:

$$F = (a \vee b \vee c) \cdot (a \vee b \vee \bar{c}) \cdot (\bar{a} \vee b \vee c) \cdot (\bar{a} \vee \bar{b} \vee c) \cdot (\bar{a} \vee \bar{b} \vee \bar{c})$$

Antivalenz-Normalform

Wenn eine kanonische disjunktive Normalform den Funktionswert Eins ergibt, kann nur eine der Elementarkonjunktionen $K_1 ... K_n$ den Funktionswert Eins aufweisen. Deshalb können die Konjunktionsterme auch antivalent verknüpft werden.

$$F = K_1 \oplus K_2 \oplus ... \oplus K_n \qquad (1.28)$$

Beispiel:

$$F = \bar{a}b\bar{c} \oplus \bar{a}bc \oplus a\bar{b}c$$

Äquivalenz-Normalform

Wenn eine kanonische konjunktive Normalform den Funktionswert Eins ergibt, so müssen alle Elementardisjunktionen den Funktionswert Eins aufweisen. Deshalb können die Disjunktionsterme $D_1 ... D_n$ auch äquivalent verknüpft werden..

$$F = D_1 \otimes D_2 \otimes ... \otimes D_n \qquad (1.29)$$

Beispiel):

$$F = (a \vee b \vee c) \otimes (a \vee b \vee \bar{c}) \otimes (\bar{a} \vee b \vee c) \otimes (\bar{a} \vee \bar{b} \vee c) \otimes (\bar{a} \vee \bar{b} \vee \bar{c})$$

Normalformen, die auf Antivalenz- oder Äquivalenzverknüpfungen beruhen, haben in der Theorie eine gewisse Bedeutung, nicht aber in der Schaltungspraxis, da diese Verknüpfungen schaltungstechnisch aufwendiger sind.

Negationstechnische Normalform

Eine negationstechnische Normalform liegt dann vor, wenn die Variablen nur einzeln negiert vorkommen, wenn es also keine insgesamt negierten Verknüpfungen gibt. Es ist im Grunde eine Zwischenstufe, die man erhält, wenn man in einem beliebigen Booleschen Formelausdruck die Negation von Verknüpfungen aufgelöst hat.

Beispielsweise ist $\bar{a} \cdot (b \vee \bar{c} \cdot d)$ eine negationstechnische Normalform; $a \cdot (b \vee \overline{c \cdot d})$ jedoch nicht.

Normalformen in der Entwurfspraxis

In der Praxis ist vor allem die disjunktive Normalform (DNF) von Bedeutung, gelegentlich auch die konjunktive Normalform (KNF). Die praktische Bedeutung disjunktiver Normalformen ergibt sich aus zwei Tatsachen:

- Sie hängen auf einfache Weise mit der gleichsam naiven Auffassung des jeweiligen Entwurfsproblems zusammen: die Booleschen Gleichungen werden typischerweise für den Funktionswert Eins aufgestellt (Prinzip der Assertion), und es liegt nahe, dem Bitwert Null die negierte und dem Bitwert Eins die nicht negierte Variable zuzuordnen.
- Sie entsprechen direkt einer zweistufigen UND-ODER-Schaltungsstruktur.

Eine UND-Verknüpfung aller n Variablen (negiert und nicht negiert) wählt im Grunde einen der 2^n Punkte des Booleschen Raums B^n aus und weist ihm den Funktionswert Eins zu.

So wird im Beispiel von Abb. 1.34 der Raumpunkt 010 dann ausgewählt, wenn die Variable a = 0 ist UND die Variable b = 1 ist UND die Variable c = 0 ist.

Erfordert es das Entwurfsproblem, k Punkten des Booleschen Raums den Funktionswert Eins zuzuweisen, so liegt es nahe, k UND-Gatter mit n Eingängen (negiert und nicht negiert) vorzusehen und deren Ausgänge disjunktiv zu verknüpfen.

Summen und Produkte

Gelegentlich bezeichnet man – mit Bezug auf naheliegende Analogien – die Disjunktion als Summe und die Konjunktion als Produkt (Disjunktion: $a + b$; Konjunktion: $a \bullet b$ oder ab).

Sum of Products (SOP oder S.O.P.) ist eine übliche Bezeichnung für disjunktive Normalformen (= disjunktive Verknüpfung von Konjunktionen).

Product of Sums (POS oder P.O.S.) ist eine übliche Bezeichnung für konjunktive Normalformen (= konjunktive Verknüpfung von Disjunktionen).

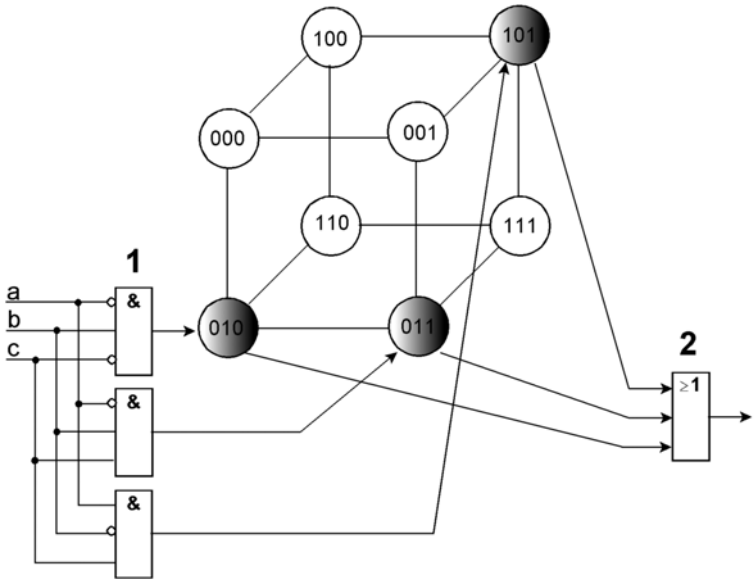


Abb. 1.34 Weshalb die UND-ODER-Struktur so beliebt ist ... Eine Boolesche Gleichung weist im Grunde bestimmten Punkten des Booleschen Raums einen der Funktionswerte Null oder Eins zu. Die Bevorzugung des Funktionswertes Eins ergibt sich aus der Tradition der Aussagenlogik – man will schließlich wahre Aussagen formulieren (Prinzip der Assertion). 1 - jedes der UND-Gatter spricht einen Punkt des Booleschen Raums an (mit dem Beispiel gemäß Abb. 1.33); 2 – die UND-Verknüpfungen werden disjunktiv zusammengefasst. Der Funktionswert Eins soll sich ergeben, wenn einer der Raumpunkte 010 ODER 011 ODER 101 ausgewählt worden ist.

Von der Wahrheitstabelle zur kanonischen Normalform

Die Wahrheitstabelle ist eine Zuordnung, aus der sowohl die Eins-Menge als auch die Null-Menge ersichtlich sind. Die Auswertung richtet sich danach, welche Boolesche Gleichung dargestellt werden soll. Die folgenden Angaben beziehen sich auf die Darstellung des Funktionswertes Eins (Boolesche Gleichung der Form $f(\underline{x}) = 1$). Ist der Funktionswert Null darzustellen (Boolesche Gleichung der Form $g(\underline{x}) = 0$), sind in den nachfolgenden Schritten Eins-Menge und Null-Menge gegeneinander auszutauschen.

Bestimmung der disjunktiven Normalform:

- Es werden alle Belegungen ausgewertet, denen der Funktionswert Eins zugeordnet ist (Eins-Menge).
- Für jede dieser Belegungen wird ein Konjunktionsterm gebildet, der aus allen Variablen besteht (Elementarkonjunktion).
- Die mit Null belegten Variablen werden negiert.
- Die so gewonnenen Minterme werden disjunktiv verknüpft.

Bestimmung der konjunktiven Normalform:

- Es werden alle Belegungen ausgewertet, denen der Funktionswert Null zugeordnet ist (Null-Menge).
- Für jede dieser Belegungen wird ein Disjunktionsterm gebildet, der aus allen Variablen besteht (Elementardisjunktion).
- Die mit Eins belegten Variablen werden negiert.
- Die so gewonnenen Maxterme werden konjunktiv verknüpft.

dez.	a	b	c	F	a) DNF	b) KNF
0	0	0	0	0		$M_0 = a \vee b \vee c$
1	0	0	1	0		$M_1 = a \vee b \vee \bar{c}$
2	0	1	0	1	$m_2 = \bar{a} \cdot b \cdot \bar{c}$	
3	0	1	1	1	$m_3 = \bar{a} \cdot b \cdot c$	
4	1	0	0	0		$M_4 = \bar{a} \vee b \vee c$
5	1	0	1	1	$m_5 = a \cdot \bar{b} \cdot c$	
6	1	1	0	0		$M_6 = \bar{a} \vee \bar{b} \vee c$
7	1	1	1	0		$M_7 = \bar{a} \vee \bar{b} \vee \bar{c}$

Abb. 1.35 Gewinnung der kanonischen Normalformen einer Booleschen Gleichung F(a, b, c) = 1. Die DNF besteht aus den disjunktiv zu verknüpfenden Mintermen, die KNF aus den konjunktiv zu verknüpfenden Maxtermen. Die Belegungen und Minterme der Eins-Menge sind grau hervorgehoben.

Zum Prinzip der konjunktiven Normalform
Die disjunktive Normalform ist – als UND-ODER-Struktur – intuitiv ohne weiteres klar. Hingegen bereitet es gelegentlich Schwierigkeiten, das Prinzip einer ODER-UND-Struktur – also der konjunktiven Normalform – zu verstehen. Der Funktionswert Eins ergibt sich dann, wenn keine der Variablenbelegungen vorliegt, die den Funktionswert Null ergeben (im Beispiel also nicht M_0 UND nicht M_1 UND nicht M_4 UND nicht M_6 UND nicht M_7). Eine Variablenbelegung $x_1x_2...x_n = 0$ liegt dann *nicht* vor, wenn gilt $\bar{x}_1 \vee \bar{x}_2 \vee ... \vee \bar{x}_n = 1$ ($\overline{x_1 \cdot x_2 \cdot ... \cdot x_n = 0}$ entspricht $\bar{x}_1 \vee \bar{x}_2 \vee ... \vee \bar{x}_n = 1$ (Negation nach DeMorgan)).

Kanonische Normalformen wechselseitig umwandeln
Gegeben ist eine Boolesche Funktion in Normalform A, gesucht ist die gleiche Funktion in Normalform B. Es gibt zwei Ansätze:

- Die Nutzung der jeweils komplementären Lösungsmenge. Wenn Normalform A der Eins-Menge entspricht, dann die Null-Menge nehmen und umgekehrt. Die Elemente dieser Menge werden als Terme der Normalform B dargestellt.
- Die rechnerische Umformung.

Von der disjunktiven zur konjunktiven Normalform	**Von der konjunktiven zur disjunktiven Normalform**
1. Funktion nach DeMorgan negieren. 2. Klammern auflösen; dabei redundante Konjunktionsterme streichen. 3. Funktion erneut nach DeMorgan negieren.	1. Klammern auflösen. 2. Redundante Konjunktionsterme streichen.

Tabelle 1.7 Schritte der rechnerischen Umformung von kanonischen Normalformen.

1.5.5 Rechenregeln der Schaltalgebra

Kommutativität

Bei gleichartigen Verknüpfungen ist die Reihenfolge der Variablen gleichgültig:

$$\begin{aligned} a \cdot b &= b \cdot a \\ a \vee b &= b \vee a \\ a \oplus b &= b \oplus a \\ a \otimes b &= b \otimes \cdot a \end{aligned} \tag{1.30}$$

Anwendung: die Reihenfolge der Belegung der Gattereingänge ist gleichgültig.

Assoziativität

Bei gleichartigen Verknüpfungen ist die Klammeranordnung gleichgültig:

$$\begin{aligned} a \cdot (b \cdot c) &= (a \cdot b) \cdot c = (a \cdot c) \cdot b = a \cdot b \cdot c \\ a \vee (b \vee c) &= (a \vee b) \vee c = (a \vee c) \vee b = a \vee b \vee c \\ a \oplus (b \oplus c) &= (a \oplus b) \oplus c = (a \oplus c) \oplus b = a \oplus b \oplus c \\ a \equiv (b \equiv c) &= (a \equiv b) \equiv c = (a \equiv c) \equiv b = a \equiv b \equiv c \end{aligned} \tag{1.31}$$

Anwendung: Gatter mit mehreren Eingängen können durch hintereinandergeschaltete gleichartige Gatter mit weniger Eingängen ersetzt werden (und umgekehrt), wobei die Belegung der Eingänge gleichgültig ist.

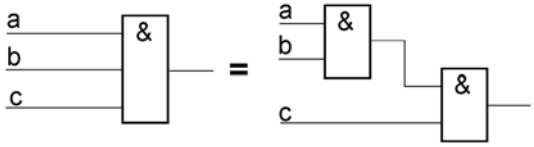


Abb. 1.36 Anwendungsbeispiel der Assoziativität.

Distributivität
Eine Funktion vor einer Klammer ist auf alle Variablen in der Klammer anzuwenden; die jeweiligen Ergebnisse sind gemäß der eingeklammerten Funktion zu verknüpfen. Diese Eigenschaft gilt nur für UND und ODER, nicht aber für Antivalenz (XOR) und Äquivalenz (XNOR)[10]:

$$a \cdot (b \vee c) = a \cdot b \vee a \cdot c \qquad (1.32)$$

$$a \vee (b \cdot c) = (a \vee b) \cdot (a \vee c) \qquad (1.33)$$

Für Antivalenz (XOR) und Äquivalenz (XNOR) gilt jedoch:

$$a \cdot (b \oplus c) = a \cdot b \oplus a \cdot c \qquad (1.34)$$

$$a \vee (b \otimes c) = (a \vee b) \otimes (a \vee c) \qquad (1.35)$$

Anwendung: Einklammerung bedeutet, dass die Verknüpfung außerhalb der Klammer nur einmal vorzusehen ist (Aufwandsersparnis). Ist ein solcher Klammerausdruck aufzulösen, so ist die auf den Klammerausdruck angewandte Verknüpfung für jede eingeklammerte Variable gesondert erforderlich.

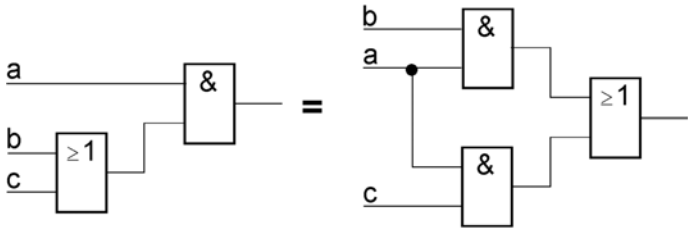


Abb. 1.37 Anwendungsbeispiel der Distributivität.

Doppelte Negation
Die Negation einer Negation ist eine Bejahung (Assertion, Identität).

$$\overline{\overline{a}} = a \qquad (1.36)$$

Anwendung: zwei hintereinandergeschaltete Negatoren ergeben eine 1:1-Treiberstufe.

Idempotenz
Die Verknüpfung einer Variablen mit sich selbst ergibt den Wert der Variablen. Diese Eigenschaft gilt nur für UND und ODER, nicht für Antivalenz (XOR) und Äquivalenz (XNOR):

10: Beispiel $a \oplus (b \cdot c) \neq (a \oplus b) \cdot (a \oplus c)$

$$a \cdot a = a \; ; \quad a \vee a = a \tag{1.37}$$

Anwendung: Es ist möglich, ein Signal an mehr als einen Eingang des gleichen Gatters anzuschließen (um ungenutzte Eingänge zu belegen).

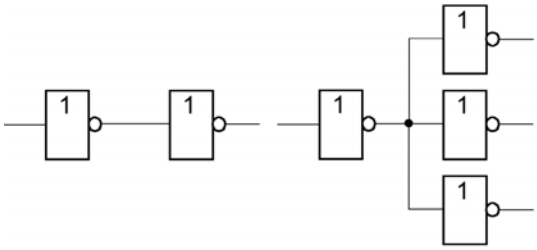


Abb. 1.38 Zwei Negatoren hintereinander ergeben eine Treiberstufe. Bei Bedarf nach höherer Treibfähigkeit können in der zweiten Reihe auch mehrere Negatoren parallel angeordnet werden.

Verknüpfungen mit Festwerten

$$1 \cdot a = a \tag{1.38}$$

$$0 \vee a = a \tag{1.39}$$

$$0 \oplus a = a \; ; \quad 1 \oplus a = \bar{a} \tag{1.40}$$

$$0 \equiv a = \bar{a} \; ; \quad 1 \equiv a = a \tag{1.41}$$

Anwendung: (1.38 und (1.39) zur Beschaltung ungenutzter Eingänge, (1.40) und (1.41) als gesteuerte Negation (das Signal a wird – abhängig von einem Steuersignal – entweder negiert oder nicht negiert weitergegeben).

Reduktion auf Festwerte

$$0 \cdot a = 0 \; ; \quad 1 \vee a = 1 \tag{1.42}$$

$$a \cdot \bar{a} = 0 \; ; \quad a \vee \bar{a} = 1 \tag{1.43}$$

$$a \oplus a = 0 \tag{1.44}$$

$$a \equiv a = 1 \tag{1.45}$$

Anwendung: DieseVerknüpfungen sind offensichtlich unbrauchbar (Kontradiktion oder Tautologie). Wenn sich im Lauf des Schaltungsentwurfs eine solche Verknüpfung ergibt, muss ein Fehler vorliegen.

Antivalenz- und Äquivalenz-Rechenregeln

$$a \oplus \bar{b} = \bar{a} \oplus b = \overline{a \oplus b} = a \otimes b \tag{1.46}$$

$$a \otimes \bar{b} = \bar{a} \otimes b = \overline{a \otimes b} = a \oplus b \tag{1.47}$$

$$\bar{a} \oplus \bar{b} = a \oplus b \; ; \quad \bar{a} \otimes \bar{b} = a \otimes b$$

Anwendung: Schaltungsoptimierung. Die Äquivalenz ist die Negation der Antivalenz und umgekehrt. Dabei ist es gleichgültig, ob man den Funktionswert negiert oder eine der Variablen. Die Negation beider Variablen hat keinen Einfluss (liegen nur negierte Variable vor, so kann man sie direkt – also ohne zusätzliche Negation – antivalent oder äquivalent verknüpfen).

Einsetzungsregel
Anstelle einer Variablen in einer Schaltfunktion kann eine beliebige Schaltfunktion eingesetzt (substituiert) werden:

$$f_1(a, b, c, \ldots) = f_1(a, f_2, c, \ldots) \tag{1.48}$$

Eine Schaltfunktion in einer Schaltfunktion heißt auch Teilschaltfunktion. Man kann Teilschaltfunktionen und einzelne Variable wechselseitig einsetzen. In (1.48) wurde anstelle der Variablen *b* die Schaltfunktion f_2 eingesetzt. f_2 kann sowohl von Variablen abhängen, die in f_1 vorkommen, als auch von weiteren Variablen.

Ersetzungsregel
Jede Teilschaltfunktion f_i in einer Schaltfunktion f kann durch eine äquivalente Teilschaltfunktion f_i^* ersetzt werden. Zwei Schaltfunktionen f_i und f_i^* sind äquivalent, wenn sie die gleiche Wahrheitstabelle haben.

$$f(a, b, f_1, f_2, f_3) = f(a, b, f_1^*, f_2^*, f_3^*) \text{ wenn für alle i gilt } f_i = f_i^* \tag{1.49}$$

DeMorgansche Regeln
Diese beschreiben die Dualität zwischen UND und ODER. Sie gelten sowohl für einzelne Variable als auch für Schaltfunktionen (vgl. die Einsetzungs- und die Ersetzungsregel):

1. DeMorgansche Regel:
Negation des UND = ODER der einzeln negierten Variablen.

$$\overline{a \cdot b} = \bar{a} \vee \bar{b} \tag{1.50}$$

2. DeMorgansche Regel:
Negation des ODER = UND der einzeln negierten Variablen.

$$\overline{a \vee b} = \bar{a} \cdot \bar{b} \tag{1.51}$$

Somit gilt auch:

$$\overline{a \cdot \bar{b}} = \bar{a} \vee b;\ \overline{\bar{a} \cdot b} = a \vee \bar{b};\ \overline{\bar{a} \cdot \bar{b}} = a \vee b \tag{1.52}$$

$$\overline{a \vee \bar{b}} = \bar{a} \cdot b;\ \overline{\bar{a} \vee b} = a \cdot \bar{b};\ \overline{\bar{a} \vee \bar{b}} = a \cdot b \tag{1.53}$$

Merksatz im Englischen: Break the line and change the sign.

Allgemeine Negation nach DeMorgan:

- Ist ein gemeinsames Negationszeichen vorhanden, so entfällt es. Ist es nicht vorhanden, wird es gesetzt.
- Die Variablen werden einzeln negiert (aus a wird $\bar{a}$, aus $\bar{a}$ wird a).
- Die Verknüpfungssymbole werden ausgewechselt (aus UND wird ODER, aus ODER wird UND ($\cdot \Rightarrow \vee$; $\vee \Rightarrow \cdot$).

$$\overline{f(\cdot;\ \vee;\ a;\ b;\ c;\ \ldots)} = f(\vee;\ \cdot;\ \bar{a};\ \bar{b};\ \bar{c};\ \ldots) \tag{1.54}$$

$$\overline{a \cdot b \cdot c \ldots} = \bar{a} \vee \bar{b} \vee \bar{c} \ldots \tag{1.55}$$

$$\overline{a \vee b \vee c \ldots} = \bar{a} \cdot \bar{b} \cdot \bar{c} \ldots \tag{1.56}$$

$$\overline{\bar{a} \vee \bar{b} \vee \bar{c} \ldots} = a \cdot b \cdot c \ldots \tag{1.57}$$

$$\overline{\bar{a} \cdot \bar{b} \cdot \bar{c} \ldots} = a \vee b \vee c \ldots \tag{1.58}$$

Erweiterungssatz nach Boole und Shannon
Eine Boolesche Funktion f(x) mit einer Variablen kann wie folgt dargestellt werden:

$$f(x) = \bar{x} \cdot f(0) \vee x \cdot f(1) \tag{1.59}$$

Das Prinzip kann sinngemäß auf zwei und mehr Variable erweitert werden. Der Erweiterungssatz (Expansion Theorem) ist u. a. die theoretische Grundlage der kanonischen

disjunktiven Normalform und damit der wechselseitigen 1:1-Entsprechung von Wahrheitstabelle und Funktionsausdruck. Beispielsweise erhält man für eine Boolesche Funktion mit drei Variablen a, b, c folgenden Ausdruck:

$$\begin{aligned} &f(a,b,c) = \\ &\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot f(0,0,0) \vee \bar{a}\cdot\bar{b}\cdot c\cdot f(0,0,1) \vee \bar{a}\cdot b\cdot\bar{c}\cdot f(0,1,0) \vee \bar{a}\cdot b\cdot c\cdot f(0,1,1) \vee \\ &a\cdot\bar{b}\cdot\bar{c}\cdot f(1,0,0) \vee a\cdot\bar{b}\cdot c\cdot f(1,0,1) \vee a\cdot b\cdot\bar{c}\cdot f(1,1,0) \vee a\cdot b\cdot c\cdot f(1,1,1) \end{aligned} \quad (1.60)$$

f(0, 0, 0), f(0, 0, 1) usw. sind die den Variablenbelegungen zugeordneten Funktionswerte aus der Wahrheitstabelle. Ist ein solcher Funktionswert = 1, so erscheint die betreffende Konjunktion im Formelausdruck der Funktion. Beispiel:

$$\begin{aligned} &f(a,b,c) = \\ &\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot 0 \vee \bar{a}\cdot\bar{b}\cdot c\cdot 0 \vee \bar{a}\cdot b\cdot\bar{c}\cdot 1 \vee \bar{a}\cdot b\cdot c\cdot 1 \vee \\ &a\cdot\bar{b}\cdot\bar{c}\cdot 0 \vee a\cdot\bar{b}\cdot c\cdot 1 \vee a\cdot b\cdot\bar{c}\cdot 0 \vee a\cdot b\cdot c\cdot 0 = \\ &\bar{a}\cdot b\cdot\bar{c} \vee \bar{a}\cdot b\cdot c\cdot \vee a\cdot\bar{b}\cdot c\cdot \end{aligned}$$

Universelle Schaltnetze

(1.60) ist ein Ausdruck, der alle überhaupt möglichen Produktterme umfasst, wobei jeder Produktterm um eine konjunktive Verknüpfung mit dem jeweiligen Funktionswert erweitert ist. Eine nach dem Schema von (1.60) aufgebaute Schaltungsanordnung kann jede beliebige Schaltfunktion darstellen. Hierzu ist es lediglich notwendig, anstelle der Funktionsausdrücke (f(0,0,0) usw.) Festwerte oder Funktionssteuersignale anzulegen.

1.6 Schaltungsvereinfachung

Das Ziel der Schaltungsvereinfachung oder Minimierung (Minimization) besteht darin, den zur Lösung einer Entwurfsaufgabe erforderlichen Schaltungsaufwand soweit wie möglich zu verringern. Betrachten wir den grundsätzlichen Lösungsansatz einer Entwurfsaufgabe als gegeben, so liegt es nahe, zu versuchen, die kombinatorischen Schaltungen mit möglichst wenigen Gattern aufzubauen. In der Praxis geht es jedoch letzten Endes nur darum, die Kosten zu senken und nicht darum, um jeden Preis Gatter einzusparen. Es ist offensichtlich, dass die Ansätze zur Kostensenkung wesentlich von der jeweiligen Schaltungstechnologie abhängen.

Die formalen Verfahren der Schaltungsminimierung betreffen aber vor allem Schaltfunktionen in disjunktiver oder konjunktiver Normalform oder – mit anderen Worten – zweistufige reguläre Schaltnetze. Das typische Ziel der Minimierung: ein Ausdruck mit der geringsten Anzahl an Variablenbezeichnern, gleichgültig in welcher Verknüpfung. $abcd \vee ef$ und $ab \vee cd \vee ef$ sind aus dieser Sicht gleichwertig; beide Ausdrücke haben jeweils sechs Variablenbezeichner. Gibt es in der betreffenden Baureihe aber keine Gatter mit vier

Eingängen, so erfordert der erste Ausdruck mehr Aufwand (Kaskadierung). Das wird von der elementaren Theorie nicht berücksichtigt. In der Praxis ist die Schaltungsminimierung deshalb eine Kombination schaltalgebraischer Verfahren mit trickreichen Ansätzen, die die Eigenheiten des jeweiligen Bauelementesortiments berücksichtigen. In der heutigen Entwurfspraxis betrifft das Minimieren von Hand zumeist kleine Probleme, wobei das jeweilige Gattersortiment zweckmäßig auszunutzen ist. Größere Vorhaben werden typischerweise mit programmierbaren Schaltkreisen verwirklicht. Dabei übernimmt die Entwicklungssoftware das Rechnen.

1.6.1 Kürzungsregeln

Kürzungsregeln sind Gleichungen der Schaltalgebra, die einen komplexeren Ausdruck einem einfacheren Ausdruck gleichsetzen. Die Variablenbezeichner a, b, c in den folgenden Gleichungen können Variable oder Teilschaltfunktionen sein. Anwendung: Die jeweils gegebenen Schaltfunktionen daraufhin ansehen, ob sie Ausdrücke enthalten, die denen auf den linken Seiten der nachfolgenden Gleichungen (1.63) bis (1.72) entsprechen. Dabei auch versuchen, ob sich Teilschaltfunktionen erkennen lassen (erkannte Teilschaltfunktionen kann man z. B. mit einem Hilfsnamen bezeichnen). Wurde ein solcher Ausdruck gefunden, so wird er durch einen Ausdruck ersetzt, der die gleiche Form hat wie der Ausdruck auf der rechten Seite der jeweiligen Gleichung.

1. Kürzungsregel (Absorptionsregel):

$$a \vee ab = a \tag{1.61}$$

$$a \cdot (a \vee b) = a \tag{1.62}$$

2. Kürzungsregel:

$$ab \vee a\bar{b} = a \tag{1.63}$$

$$\left(a \vee b\right) \cdot \left(a \vee \bar{b}\right) = a \tag{1.64}$$

3. Kürzungsregel:

$$a \vee \bar{a}b = a \vee b \tag{1.65}$$

$$\bar{a} \vee ab = \bar{a} \vee b \tag{1.66}$$

$$\bar{a} \vee \bar{a}b = \bar{a} \vee b \tag{1.67}$$

$$a \cdot \left(\bar{a} \vee b\right) = ab \tag{1.68}$$

4. Kürzungsregel (Konsensusregel):

$$ab \vee \bar{a}c \vee bc = ab \vee \bar{a}c \qquad (1.69)$$

$$(a \vee b) \cdot (\bar{a} \vee c) \cdot (b \vee c) = (a \vee b) \cdot (\bar{a} \vee c) \qquad (1.70)$$

Beispiele:

a) $x_1 \bar{y} \vee x_1 \bar{y} \bar{z} = H_1 \vee H_1 \bar{z} = H_1 = x_1 \bar{y}$

$x_1 \bar{y}$ ist als Teilschaltfunktion zu erkennen. Sie wird zeitweilig durch die Hilfsvariable H_1 ersetzt. Dies ergibt $H_1 \vee H_1 \bar{z}$. Dieser Ausdruck entspricht offensichtlich der linken Seite von (1.63) mit H_1 = a und $\bar{z}$ = b.

b) $y\,z\,(\overline{x_1} \vee \overline{x_2}) \vee y\,z\,x_1\,x_2 = y\,z$

Der Ausdruck entspricht (1.65). Er enthält zwei Teilschaltfunktionen: yz entspricht a, $\overline{x_1} \vee \overline{x_2}$ entspricht b, $x_1 x_2$ entspricht $\bar{b}$ (Negation nach DeMorgan).

1.6.2 Elementare Minimierungsansätze

Disjunktive und konjunktive Normalformen

Ist die Schaltfunktion als disjunktive oder konjunktive Normalform gegeben, so ergeben sich Vereinfachungsmöglichkeiten aus der Anwendung der zweiten Kürzungsregel. Unterscheiden sich zwei Terme nur ein einer Variablenposition, so können beide Terme durch einen einzigen Term ersetzt werden, in dem die betreffende Variable fehlt: Wenn $abc \vee ab\bar{c} = 1$, dann muss $ab = 1$ sein, und es ist offensichtlich gleichgültig, welchen Wert c hat. Also kann c weggelassen werden:

$$abc \vee ab\bar{c} = ab$$

Das gilt sinngemäß dann, wenn in vier ansonsten gleichen Termen zwei Variable in allen Kombinationen vorkommen, in acht ansonsten gleichen Termen drei Variable usw. (allgemein k Variabe in 2^k ansonsten gleichen Termen). Beispiel:

$$ab\bar{c}\bar{d} \vee ab\bar{c}d \vee abc\bar{d} \vee abcd = ab$$

Diese Verkürzung entspricht der Wandlung von der binären zur ternären Belegungsliste, das Weglassen einer Variablen entspricht dem Einfügen eines Strichelements. In der Praxis kommt es darauf an, solche Zusammenfassungsmöglichkeiten überhaupt zu erkennen. Handelt es sich nur um wenige Variable, so kann man die Wahrheitstabelle so gestalten, dass ein bloßes (allerdings genaues) Hinsehen genügt. Das ist das Prinzip des KV-Digramms. Ansonsten hilft nur ein systematisches Durchmustern von Belegungslisten. In der Praxis lohnt es sich nicht, von Hand zu rechnen – Programme können das viel besser.

Die folgende Abbildung veranschaulicht den Zusammenhang zwischen den Punkten des Booleschen Raums B^n, die zur Lösungsmenge gehören, und den Möglichkeiten der Zusammenfassung gemäß der zweiten Kürzungsregel. Es lassen sich offensichtlich nur Belegungen zusammenfassen, deren zugeordnete Punkte im Booleschen Raum benachbart sind, also einen Abstand von Eins haben. Zwei benachbarte Punkte bilden einen eindimensionalen Unterraum im Raum B^n. Eine Elementarkonjunktion oder der zugehörige binäre Eintrag der Belegungsliste wählen einen einzelnen Punkt im Raum B^n aus. Ein um eine Variable verkürzter Minterm oder ein ternärer Eintrag der Belegungsliste (mit einem Strichelement in der jeweiligen Variablenposition) sind im Grunde Kurzangaben für zwei benachbarte Raumpunkte, mit anderen Worten, sie beschreiben einen eindimensionalen Unterraum.

Beispiele:

- Der Term $\bar{a}b\bar{c}$ oder der Belegungslisteneintrag 010 beschreiben den Raumpunkt 010 des Booleschen Raums B^3.
- Der Term $\bar{a}b$ oder der Belegungslisteneintrag 01– beschreiben die beiden benachbarten Raumpunkte 010 und 011, die einen eindimensionalen Unterraum bilden.

Sinngemäß verhält es sich, wenn aus einem Minterm zwei Variable entfernt werden können. Eine solche Vereinfachung ist dann möglich, wenn vier Punkte des Booleschen Raums B^n, die zur Lösungsmenge gehören, untereinander einen Abstand von Eins haben (Abb. 1.46). Diese Raumpunkte bilden einen Unterraum der Dimension zwei. Beispielsweise beschreibt ein Term $\bar{a}$ oder ein Belegungslisteneintrag 0 – – die vier Raumpunkte 000, 001, 010 und 011. Mit anderen Worten, das Verkürzen von Mintermen läuft darauf hinaus, im Booleschen Raum B^n Unterräume von möglichst hoher Dimension aufzufinden, deren Punkte zur Lösungsmenge gehören. Das klingt einfach, ist es aber keineswegs ...

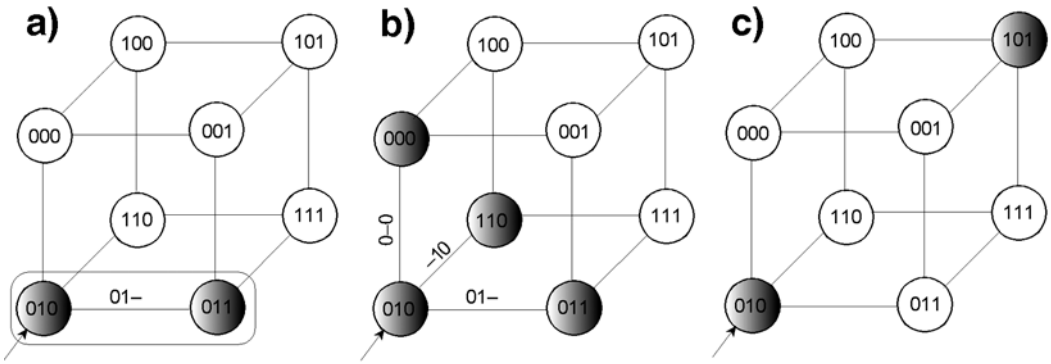


Abb. 1.39 Der Boolesche Raum und die zweite Kürzungsregel. Die Darstellung bezieht sich auf den Raumpunkt 010 (Pfeil). Wenn benachbarte Raumpunkte zur Lösungsmenge gehören, ist ein Zusammenfassen möglich. a) gehört beispielsweise der Raumpunkt 011 zur Lösungsmenge, so ergibt sich eine Zusammenfassung 01–. Eine solche ternäre Angabe beschreibt im Grunde einen aus zwei Punkten bestehenden eindimensionalen Unterraum. b) alle Raumpunkte, die vom Raumpunkt 010 den Abstand 1 haben. c) der Raumpunkt 101 hat einen zu großen Abstand und kann somit nicht in die Zusammenfassung einbezogen werden.

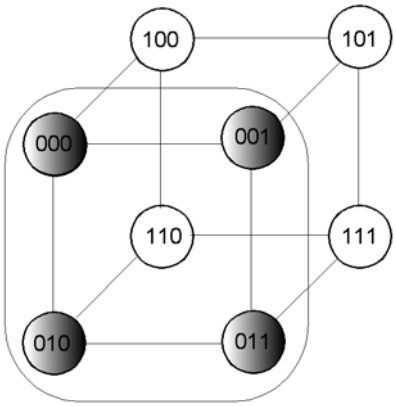


Abb. 1.40 Der Boolesche Raum und weitere Zusammenfassungen. Gehören die Raumpunkte 010, 000, 001 und 011 zur Lösungsmenge, so bilden sie einen zweidimensionalen Unterraum, der sich durch eine Angabe der Form 0 – – beschreiben lässt.

Die Mächtigkeit der Lösungsmenge

Das Grundproblem ist nicht die Mächtigkeit der Lösungsmenge an sich. Wichtig ist vielmehr, welche Möglichkeiten zum Vereinfachen gegeben sind, vor allem zum Zusammenfassen auf Grundlage der zweiten Kürzungsregel. Umfasst die Lösungsmenge nur einen einzigen Raumpunkt, so lässt sich gar nichts vereinfachen. Mit zunehmender Mächtigkeit der Lösungsmenge sollten sich mehr und mehr Gelegenheiten ergeben. Es kann durchaus sein, dass sich Schaltfunktionen, diezunächst extrem aufwendig aussehen, durch verblüffend kurze Ausdrücke darstellen und mit vergleichsweise wenig Aufwand realisieren lassen. Es kann aber auch vorkommen, dass harmlos aussehende Schaltfunktionen überhaupt keinen Ansatz zur Vereinfachung bieten. Die Vereinfachungsmöglichkeiten hängen davon ab, wie viele der

zur Lösungsmenge gehörenden Punkte des Booleschen Raums untereinander benachbart sind, sich also als Unterräume darstellen lassen.

Die Antivalenz als Beispiel einer harmlos aussehenden Schaltfunktion
Die Antivalenzverknüpfung von n Variablen ergibt immer dann eine Eins, wenn die Anzahl der Einsen der jeweiligen Variabenbelegung ungerade ist. Die Lösungsmenge einer Antivalenzverknüpfung von n Variablen umfasst 2^{n-1} Einsen. Sie belegt jeden zweiten Punkt des Booleschen Raums B^n. Somit ist keiner der zugehörigen Punkte einem anderen benachbart, und es lässt sich gar nichts verkürzen.

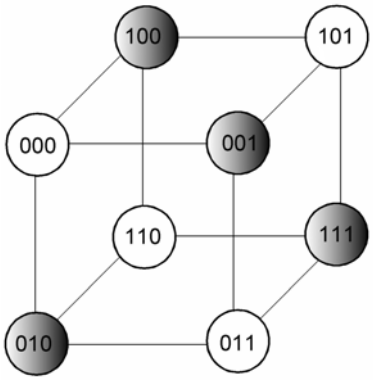


Abb. 1.41 Die Lösungsmenge der Antivalenz $a \oplus b \oplus c$ im Booleschen Raum B^3. Es gibt offensichtlich gar keine Gelegenheit, irgend etwas zusammenzufassen ...

Ob es mit der negierten Funktion besser klappt?
Hat einer Booleschen Gleichung mit n Variablen eine Lösungsmenge von mehr als 2^{n-1} Belegungen, so liegt es nahe, zu probieren, ob sich auf Grundlage der negierten Funktion oder der jeweils anderen Normalform(KNF statt DNF oder umgekehrt) bessere Möglichkeiten zum Vereinfachen und Zusammenfassen bieten. Hierbei wird die zur ursprünglichen Lösungsmenge komplementäre Punktmenge des Booleschen Raums B^n betrachtet. Hat sich die negierte Funktion als günstiger erwiesen, so wird sie implementiert, und es wird – falls erforderlich – eine zusätzliche Negation vorgesehen. Manche Entwicklungsprogramme tun dies automatisch. Manchmal lohnt sich aber auch der Versuch, die Schaltfunktion von Hand umzudrehen, denn es kann sein, dass es das Programm unterlässt, diese Variante zu berücksichtigen (Rechenzeitbedarf) oder dass es diese Möglichkeit gar nicht erkennt (z.B. wenn es sich um eine Teilfunktion in einer umfangreichen kombinatorischen Schaltung handelt).

Nebenbedingungen
Don't-Care-Belegungen sind funktionell bedeutungslose Belegungen. Es gibt zwei Arten:

a) Belegungen, die während des Betriebs grundsätzlich nicht auftreten können (Beispiel: die Werte AH bis FH am Ausgang eines Dezimalzählers).

b) Belegungen, die auftreten können, deren Funktionswert aber bedeutungslos ist.

Belegungen, für die a) oder b) gilt, schaden nichts. Sie müssen also in den Schaltfunktionen auch nicht ausdrücklich ausgeschlossen werden. Deshalb kann man sie der Schaltfunktion hinzufügen – in der Hoffnung, dass sich dadurch weitere Möglichkeiten zum Zusammenfassen ergeben. Das Hinzufügen solcher Belegungen bedeutet, dass der Lösungsmenge weitere Punkte des Booleschen Raums B^n zugeordnet werden. Hierdurch kann es sein, dass sich Unterräume mit mehr Dimensionen ergeben.

Beispiel:

Die ursprüngliche Schaltfunktion (vgl. Abb. 1.36): $F = \bar{a}b\bar{c} \vee \bar{a}bc \vee a\bar{b}c$

Der Ausdruck lässt sich offensichtlich nach der zweiten Kürzungsregel vereinfachen zu:

$$F = \bar{a}b \vee a\bar{b}c \qquad (1.71)$$

In der Anwendungsschaltung soll die Belegung 111 nie vorkommen. Somit können der Punkt 111 des Booleschen Raums B^3 zur Lösungsmenge und der Term abc zur Schaltfunktion hinzugefügt werden:

$$F = \bar{a}b \vee a\bar{b}c \vee abc$$

Auf den zweiten und dritten Term lässt sich offensichtlich die zweite Kürzungsregel anwenden. Damit ergibt sich ein gegenüber (1.71) weiter vereinfachter Ausdruck:

$$F = \bar{a}b \vee ac$$

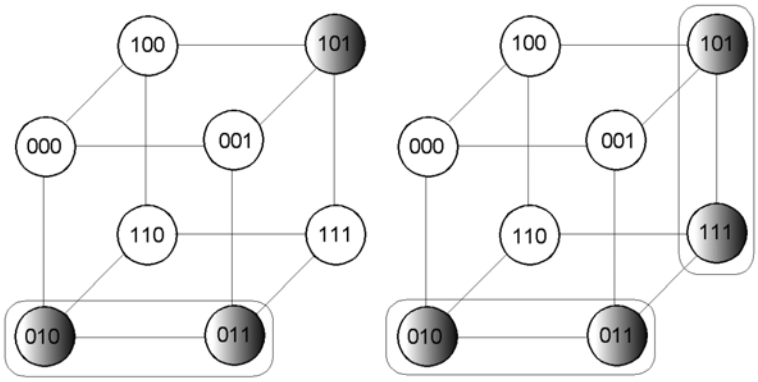


Abb. 1.42 Vereinfachung durch Hinzufügen von Nebenbedingungen. a) die ursprüngliche, b) die erweiterte Lösungsmenge.

1.6.3 Schaltungsvereinfachung mittels KV-Diagramm

Das KV-Diagramm (Karnaugh-Veitch-Diagramm, Karnaugh-Plan) ist eine Wahrheitstabelle in Form einer Matrix aus 2^n Feldern. Für jede Variablenbelegung ist ein Feld vorgesehen. Die Variablenbelegungen sind so angeordnet, dass Zusammenfassungsmöglichkeiten gemäß der zweiten Kürzungsregel durch bloßes Hinsehen zu erkennen sind. Damit dies gelingt, ist es erforderlich, dass sich die links und rechts sowie oben und unten benachbarten Felder in der Belegung jeweils nur einer Variablenposition unterscheiden. Da ein Feld nur vier solche Nachbarn haben kann, ist das gewöhnliche (zweidimensionale) KV-Diagramm auf vier Variable beschränkt. Die folgenden Abbildungen zeigen typische KV-Diagramme.

Wahrheitstabelle	KV-Diagramm
Das Bitmuster der Variablenbelegung wird als Binärzahl interpretiert. Listenanordnung. Die Reihenfolge der Bitmuster entspricht den Werten der jeweiligen Binärzahlen (Dezimaläquivalent)	Das Bitmuster der Variablenbelegung wird gemäß dem Gray-Code interpretiert. Zweidimensionale Anordnung. Die Variablen werden auf Zeilen und Spalten aufgeteilt.

Tabelle 1.8 Wahrheitstabelle und KV-Diagramm.

Vorgehensweise:

1. Wahrheitswerte eintragen. Üblicherweise trägt man nur den Wahrheitswert ein, der der jeweiligen Lösungsmenge entspricht. Kommt einem Feld der jeweils andere Wahrheitswert zu, läßt man es frei. Kein Feld vergessen! Fehlt in einem Term eine Variable, so entspricht dies einem Strichelement in der zugehörigen Belegungsliste. Solche Strichelemente sind aufzulösen. Beispiel: eine Schaltfunktion von drei Variablen a, b, c enthalte einen Produktterm a b. Dann müssen in die Felder a b c und a b $\bar{c}$ Einsen eingetragen werden.
2. Falls möglich, Don't-Care-Bedingungen eintragen (ergeben sich aus dem Anwendungsfall). Sie werden beispielsweise mit einem X bezeichnet, um sie von der eigentlichen Lösungsmenge unterscheiden zu können.
3. Nachsehen, was sich zusammenfassen lässt. Benachbarte Einträge und Don‘t Cares zu möglichst großen Blöcken zusammenfassen. Manchmal sind verschieden Zusammenfassungen möglich. Dann gibt es kein eindeutiges Ergebnis der Minimierung.
3. Ergebnis ablesen. Es entfallen alle Variable, die in einem zusammengefassten Block sowohl wahr als auch negiert vorkommen. Alle in der jeweiligen Zusammenfassung gleichbleibenden Variablen (nur wahr oder nur negiert) bilden einen Implikanden der Schaltfunktion (DNF: UND-Term, KNF: ODER-Term).

Zusammenfassungsregeln:

a) Benachbarte Einsen (bzw. Don´t Cares) lassen sich zusammenfassen.

b) Die Anzahl der zusammengefassten Einsen (bzw. Don´t Cares) muss eine Zweierpotenz sein (2, 4, 8 usw.).

c) An entgegesetzten Rändern angeordnete Einsen (bzw. Don´t Cares) lassen sich zusammenfassen.

d) Es können nur Belegungen zusammengefasst werden, die nebeneinander oder übereinander stehen (nur waagerecht und senkrecht vorgehen; nicht diagonal).

e) Blöcke, die nur Don´t Cares enthalten, werden nicht berücksichtigt.

f) Einsen (oder Don´t Cares) dürfen in mehreren Blöcken enthalten sein.

g) Das Zusammenfassen wird beendet, wenn alle Einsen berücksichtigt worden sind.

Keine überflüssigen Zusammenfassungen

Die Minimierung ist beendet, wenn alle irgendwie benachbarten Einsen in möglichst großen Blöcken zusammengefasst wurden. Es ist falsch, darüber hinaus noch alle weiteren möglichen Zusammenfassungen zu berücksichtigen. Prinzip: sind alle Einsen eines derartigen Blockes bereits in anderen Blöcken enthalten, so wird der betreffende Block nicht berücksichtigt. Ansonsten würde sich ein sog. nichtessentieller Primimplikand ergeben. Ein solcher Term ist aus Sicht der Funktionsweise nicht falsch, aber an sich unnötig, weil es bereits andere Terme gibt, die die gleichen Variablenbelegungen erfüllen. Diese Terme sind mit Kürzungsregeln nicht mehr wegzuschaffen.

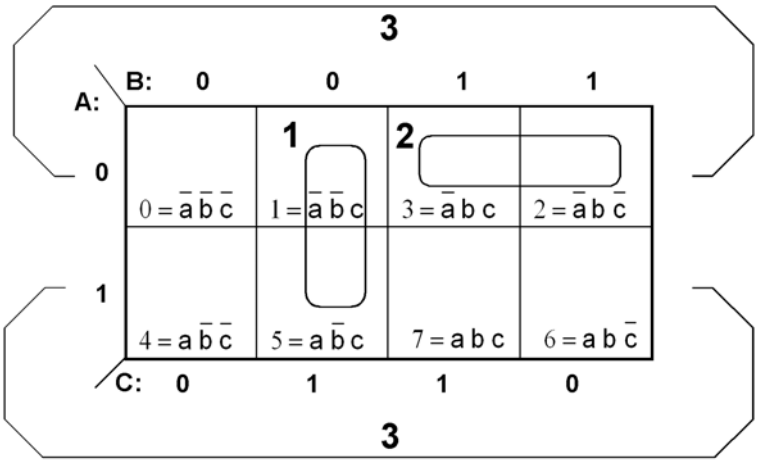
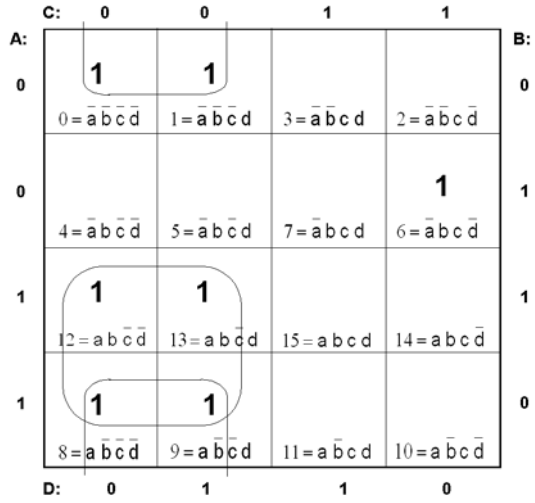


Abb. 1.43 Zusammenfassung von Feldern im KV-Diagramm. 1 - die Zusammenassung ergibt $\bar{b}\bar{c}$. 2 -die Zusammenfassung ergibt $\bar{a}\bar{b}$. 3 - auch die Felder an den Rändern gelten als benachbart.



Die ursprüngliche Schaltfunktion:

$$a\,\bar{b}\,\bar{c}\,\bar{d} \vee a\,b\,\bar{c}\,\bar{d} \vee a\,\bar{b}\,\bar{c}\,d \vee a\,b\,\bar{c}\,d \vee \bar{a}\,b\,c\,\bar{d} \vee \bar{a}\,\bar{b}\,\bar{c}\,d \vee \bar{a}\,\bar{b}\,\bar{c}\,\bar{d}$$

Ergebnis der Minimierung:

$$a\,\bar{c} \vee \bar{b}\,\bar{c} \vee \bar{a}\,b\,c\,\bar{d}$$

Abb. 1.44 Ein Minimierungsbeispiel. Terme, die sich nicht zusammenfassen lassen, werden unverändert übernommen.

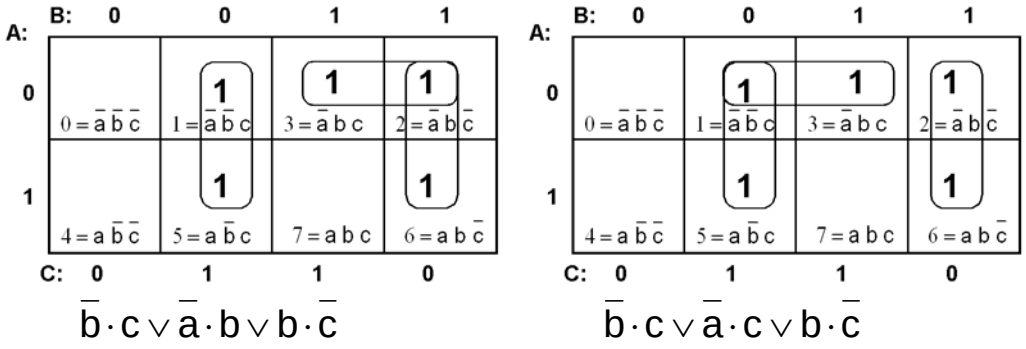


Abb. 1.45 Beispiel einer Schaltfunktion mit zwei Vereinfachungsmöglichkeiten.

KV-Diagramme für mehr als vier Variable

Bei n Variablen gibt es zu jeder beliebigen Variablenbelegung n weitere Belegungen, die sich nur in einer einzigen Variablenposition unterscheiden. Für n > 4 kann man die Variablenbelegungen in zwei Dimensionen nicht mehr so anordnen, dass sich die Zusammenfassungsmöglichkeiten auf den ersten Blick erkennen lassen. Man behilft sich u. a.mit mehreren Matrizen in einer perspektivisch-räumlichen Darstellung. In der Praxis dürfte sich der Aufwand kaum lohnen.

Anhang

1. Zweierpotenzen

Exponent n	2^n	2^{-n}
1	2	.5
2	4	.25
3	8	.125
4	16	.0625
5	32	.03125
6	64	.015625
7	128	.0078125
8	256	.00390625
9	512	.001953125
10	1024	.0009765625
11	2048	.00048828125
12	4096	.000244140625
13	8192	.0001220703125
14	16384	.00006103515625
15	32768	.000030517578125
16	65536	1.52587890625E-5
17	131072	7.62939453125E-6
18	262144	3.814697265625E-6
19	524288	1.9073486328125E-6
20	1048576	9.5367431640625E-7
21	2097152	4.76837158203125E-7
22	4194304	2.38418579101563E-7
23	8388608	1.19209289550781E-7
24	16777216	5.96046447753906E-8
25	33554432	2.98023223876953E-8
26	67108864	1.49011611938477E-8
27	134217728	7.45058059692383E-9
28	268435456	3.72529029846191E-9
29	536870912	1.86264514923096E-9
30	1073741824	9.31322574615479E-10
31	2147483648	4.65661287307739E-10
32	4294967296	2.3283064365387E-10

2. Schaltsymbole für Gatter

Abbildung 1 veranschaulicht die am weitesten verbreiteten Standards anhand typischer Beispiele.

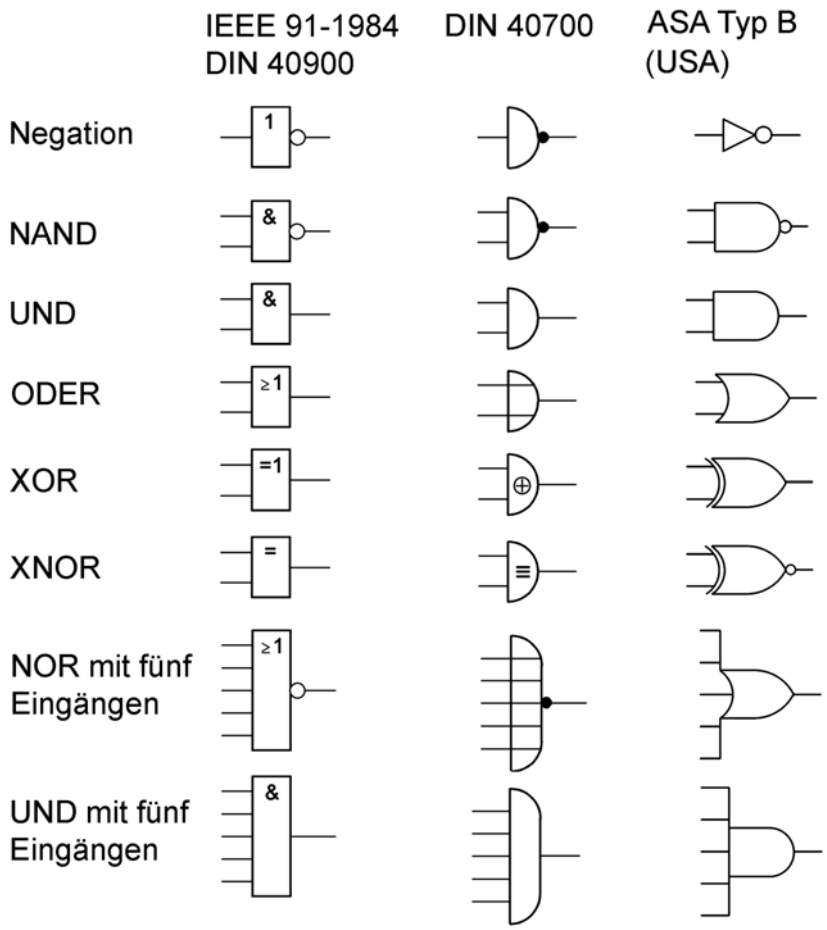


Abb. 1 Schaltsymbole für Gatter. DIN 40700 und ASA sind veraltet. Die ASA-Symbole werden aber noch oft verwendet.

3. Der Standard ANSI/IEEE 91-1984 (DIN 40900, Teil 12)

Es hat viele Jahre gedauert, bis dieser Standard ausgearbeitet war. Dabei hatte man sich das Ziel gestellt, nicht nur die ein- und ausgangsseitigen Signale anzugeben, sondern auch die Wirkungsweise der Funktionselemente mit symbolischen Mitteln zu beschreiben.

Auf naheliegende Weise verschaltete kombinatorische Funktionselemente (z. B. UND-ODER-Strukturen) kann man zusammenfassen (Abbildung 2). Gemeinsame Steuer- oder Ausgangsschaltungen (Common Control Blocks, Common Output Elements) können vom Datenteil abgesetzt werden (Abbildung 3). Besondere Symbole (Qualifying Symbols) kennzeichnen die Wirkung von Ein- und Ausgängen (Abbildung 4) sowie die funktionellen Abhängigkeiten (Abhängigkeitsnotation). Abbildung 5 zeigt einige Beispiele.

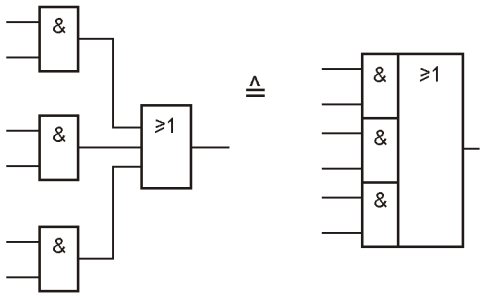


Abb. 2 Zusammenfassung kombinatorischer Funktionselemente.

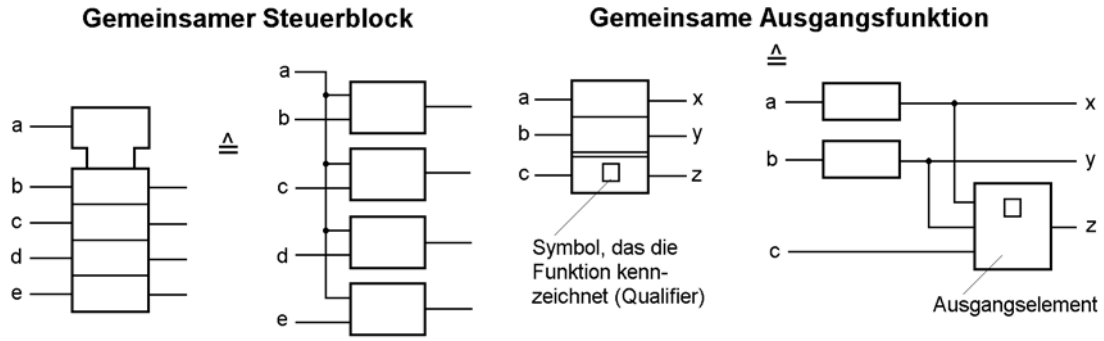


Abb. 3 Gemeinsame Steuer- und Ausgangssschaltungen.

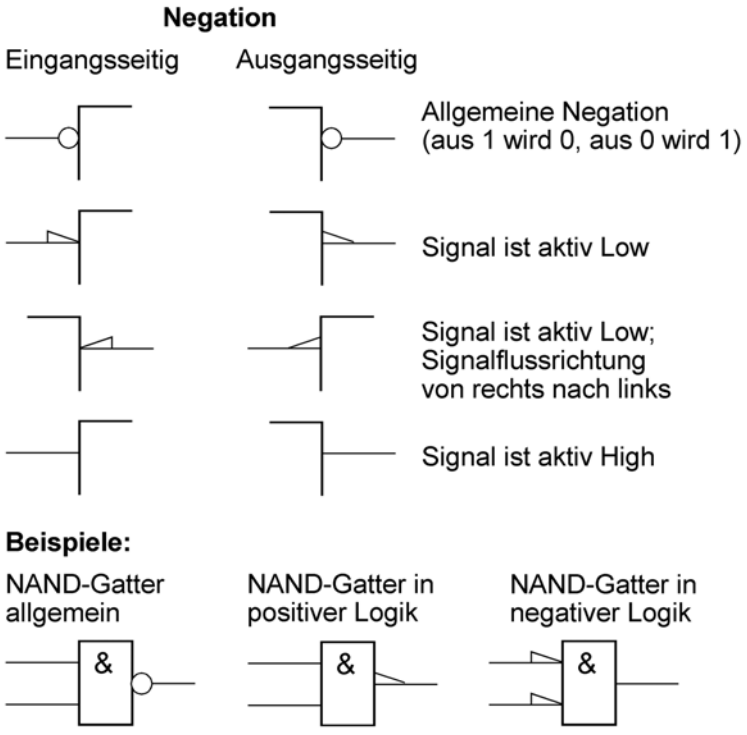


Abb. 4 Negationssymbole.

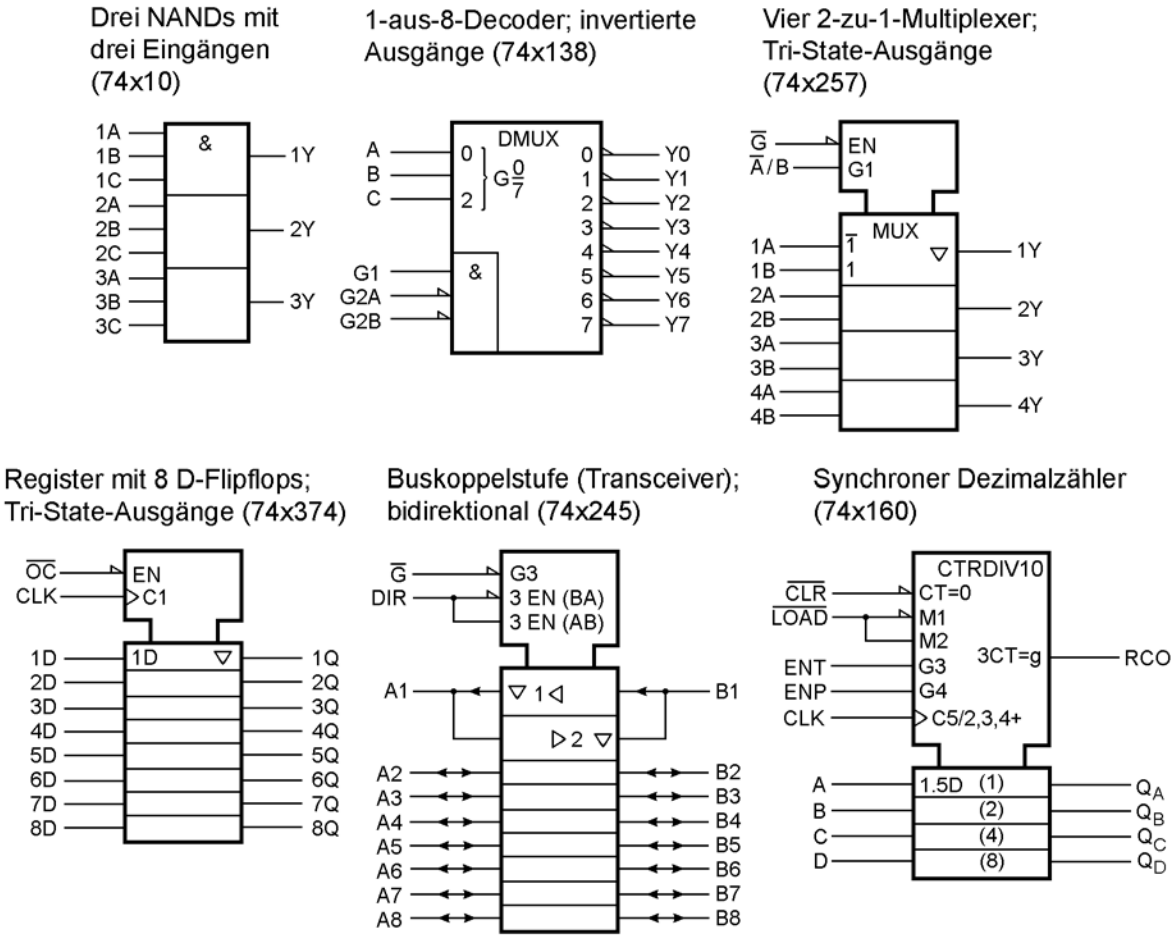


Abb. 5 Beispiele (nach Texas Instruments). Die Zahlenangaben *74x...* betreffen die Schaltkreistypen.

Die Praxis

Das Ziel, funktionelle Abhängigkeiten symbolisch *exakt* anzugeben, ist im Laufe der Zeit von der Entwicklung der Schaltkreistechnologie überholt worden. Die Symbolik gestattet nur die Wiedergabe vergleichsweise einfacher Funktionszusammenhänge, etwa im Falle eines Decoders, Zählers oder Schieberegisters. Wie will man aber die Wirkungsweise eines Prozessors, eines Videoschaltkreises oder eines Motherboard-Steuerschaltkreises darstellen? Jeder derartige Schaltkreis erfordert ein Handbuch von einigen hundert Seiten... Deshalb sind in vielen modernen Schaltplänen keine Schaltsymbole ähnlich Abbildung 5 zu finden. Höher integrierte Schaltkreise werden durch einfache Kästchen mit Ein- und Ausgängen dargestellt. Diese Vereinfachung wird oft auch auf Funktionselemente ausgedehnt, die eigentlich ohne Weiteres standardgemäß darstellbar wären, wie Register und Zähler. Nicht selten verwendet man auch in modernen Schaltplänen und in den Symbolbibliotheken der Entwicklungssysteme noch die alte Gattersymbolik nach ASA. Das hat den Vorteil, Gatter, Negatoren, Treiber usw. auf den ersten Blick von den komplexeren Funktionselementen unterscheiden zu können.

4. Die Kabelbaumdarstellung

Digitale Schaltungen sind oft durch komplizierte Verbindungen und durch viele einzelne Signalwege gekennzeichnet (Beispiel: ein Prozessor mit 64-Bit-Daten- und 32-Bit-Adreßbus). Solche Signalwege werden oft als "Kabelbaum" dargestellt (Abbildung 6). Der Kabelbaum selbst ist als dicke Linie gezeichnet, in die alle eingehenden Signale ein- und von der alle abgehenden ausmünden. Manchmal zeichnet man anstelle einer dicken – voll geschwärzten – Linie einen durch zwei dünne Linien begrenzten Schlauch. In manchen Schaltplänen hat man sich bemüht, die Anzahl der Leitungen durch die Dicke des Kabelbaums und das Aus- und Einmünden von Leitungen bildhaft zu veranschaulichen (Abbildung 6 rechts), in manchen gibt man sich hingegen mit einer mehr schematischen Darstellung zufrieden (Abbildung 6 links). Die Signale sind entweder durchnumeriert oder einzeln mit Signalnamen bezeichnet. Um den Signalfluß im Detail zu erkennen, müssen wir den Kabelbaum insgesamt sowie die jeweiligen Leitungsnummern verfolgen.

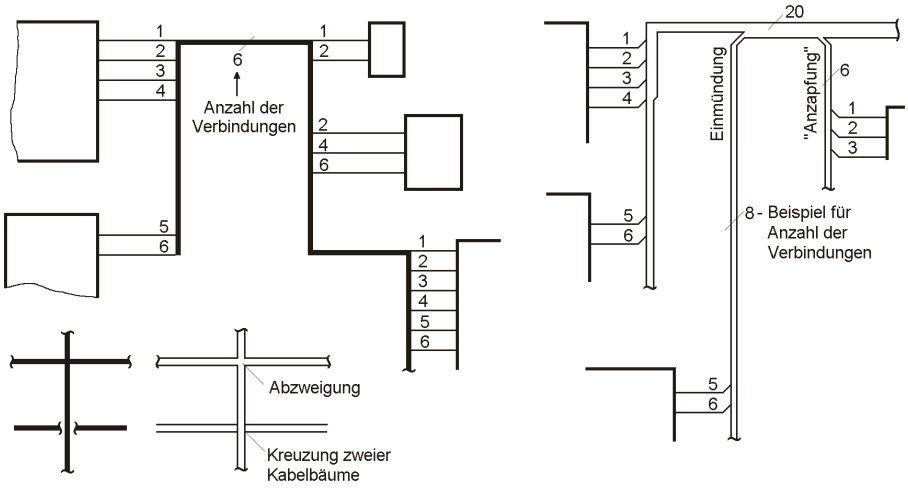


Abb. 6 Kabelbaumdarstellungen.

Achten Sie darauf, ob Kabelbäume miteinander verbunden sind oder nicht. In nicht verbundenen (unabhängigen) Kabelbäumen werden auch die einzelnen Leitungen unabhängig voneinander durchnumeriert. Sind beispielsweise 2 Kabelbäume A, B *nicht* miteinander verbunden, so hat die Leitung Nr. 1 in A nichts mit der Leitung Nr. 1 in B zu tun (Abb. 7).

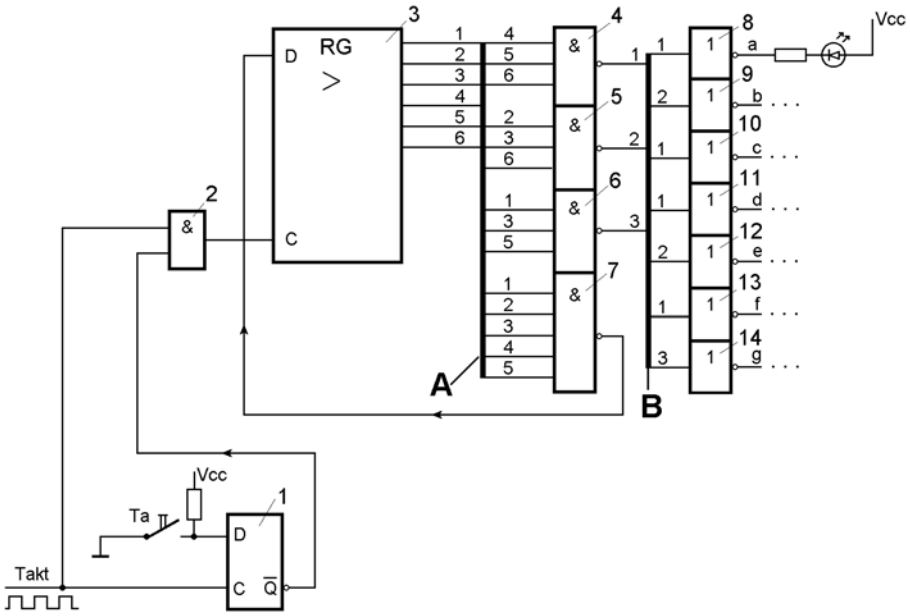


Abb. 7 Schaltplan-Beispiel mit 2 Kabelbäumen. In Kabelbaum A verbindet beispielsweise die Leitung Nr. 1 Funktionselement 3 mit den Funktionselementen 6 und 7; die Leitung Nr. 1 in Kabelbaum B verbindet hingegen Funktionselement 4 mit den Funktionselementen 8, 10, 11 und 13.