5 Rechnerarchitektur

Computer bestehen aus einer Zentraleinheit (engl. *Central Processing Unit*, kurz *CPU*), einem Arbeitsspeicher (engl. *Random Access Memory*, kurz *RAM*) und Peripheriegeräten. Alle diese Teile sind hochkomplexe Schaltkreise, die hauptsächlich aus *Transistoren* aufgebaut sind. Transistoren werden hier als als elektrisch gesteuerte Ein-Aus-Schalter eingesetzt. Durch geschickte Kombination vieler solcher Schalter entstehen Schaltkreise, die jedes gewünschte Verhalten realisieren können. Die *boolesche Algebra*, die wir in der ersten Hälfte dieses Kapitels kennen lernen, erlaubt es uns, zu einer beliebigen Schaltaufgabe einen entsprechenden Schaltkreis auszurechnen. Damit ausgerüstet zeigen wir, wie die wichtigsten Bauelemente eines Rechners, nämlich ALU (engl. *Arithmetic Logic Unit*, kurz *ALU*) und Speicher, aus einfacheren Schaltkreisen aufgebaut werden können. Aus diesen konstruieren wir danach eine mikroprogrammierte CPU und vollziehen damit den Übergang von der Hard- zur Software. Wir verfolgen diesen bis zum Maschinencode und Assembler und diskutieren anschließend noch RISC (engl. *Reduced Instruction Set Computer)* als alternative CPU-Architektur.

Dieses Kapitel erläutert also prinzipiell, wie durch geschickte Kombination von Transistoren ein komplexes Gerät wie ein PC entsteht. Wenn man wollte, könnte man Transistoren auch durch optische Schalter ersetzen und mit den gleichen Prinzipien einen optischen Computer konstruieren. Durch die schnelleren Umschaltzeiten optischer Bauteile darf man sich einen erheblichen Geschwindigkeitsgewinn erhoffen. Allerdings sind optische Schalter heute noch nicht so einfach zu realisieren wie Transistoren. Insbesondere ist eine technische Lösung für die Zusammenfassung (Integration) von Tausenden oder gar Millionen optischer Bauelemente auf einem Chip noch in weiter Ferne. Einige Aspekte der Herstellung elektronischer Chips wollen wir im ersten Unterkapitel beleuchten.

5.1 Vom Transistor zum Chip

Das für uns wichtigste elektronische Bauelement ist der so genannte *MOS-Transistor*. MOS ist die Abkürzung für den englischen Begriff *Metal-Oxide-Semiconductor* (Metalloxid-Halbleiter). Es gibt verschiedene Arten von MOS-Transistoren, alle sind aber, wie auch in der folgenden Abbildung zu sehen, aus mehreren Materialschichten aufgebaut.

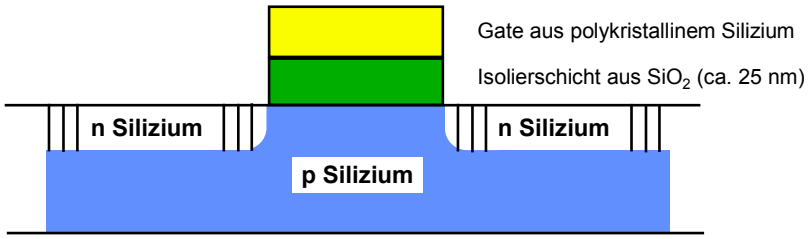


Abb. 5.1: *Ein MOS-Transistor*

Der oben gezeigte Transistor enthält bereits verschiedene Materialien: neben dem nicht gezeigten reinen Silizium zusätzlich p-Silizium, n-Silizium, Siliziumdioxid (SiO_2) und polykristallines Silizium. Bei p-Silizium bzw. n-Silizium handelt es sich um absichtlich verunreinigtes (*dotiertes*) Silizium, das zusätzliche positive bzw. negative Ladungsträger enthält. Hinzu kommen noch die ebenfalls nicht gezeigten metallischen Leitungsebenen, Abdichtungen, Kontakte etc. Diese verschiedenen Materialien und Strukturen werden beim Herstellungsprozess schrittweise erzeugt.

Für uns ist hier vor allem relevant, dass der Transistor wie ein Schalter wirkt. Dies soll in dem folgenden symbolischen Schaltbild zum Ausdruck kommen.

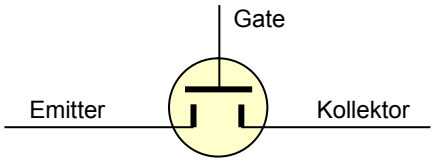


Abb. 5.2: *Ein Transistor als Schalter*

Der Transistor besitzt nach außen drei elektrische Anschlüsse, diese werden als *Emitter*, *Kollektor* und *Gate* bezeichnet. Ist auf dem *Gate* keine Ladung, ist der Schalter offen, es kann also kein Strom vom Emitter zum Kollektor fließen. Wenn sich auf dem Gate Ladungsträger befinden, ist der Schalter geschlossen.

5.1.1 Chips

Ein *Chip* ist ein dünnes Silizium-Scheibchen (daher Chip), auf das die Transistorschaltung beim Herstellungsprozess aufgebracht wird. Da auf einer daumennagelgroßen Fläche eine sehr große Anzahl von Schaltgliedern zu einem Schaltkreis zusammengefasst werden, nennt man das entstandene Bauteil auch *Integrated Circuit* (*IC*). Mit den Jahren wuchs die Anzahl der Bauelemente auf einem einzigen Chip um mehrere Größenordnungen, entsprechend wandelte sich auch der Name über LSI (large scale IC) zu VLSI (very large scale IC). Heutige CPU-Chips ent-

halten einige 100 000 000 Transistoren auf einer Fläche von weniger als 100 mm^2, Speicherchips können aufgrund ihrer regelmäßigeren Struktur noch höher integriert werden.

Die Dicke eines Chips beträgt nur etwa 1/10 mm, die der *aktiven Schicht* ist noch erheblich geringer. In der aktiven Schicht finden sich die Transistoren, Dioden, Widerstände und die Leitungen. Der Chip ist in ein Gehäuse aus Kunststoff oder Keramik eingebettet, das erheblich größer ist als das Silizium-Scheibchen. Die Verbindungen von dem inneren Silizium-Scheibchen zu den Außenkontakten des Chip-Gehäuses werden mithilfe hauchdünner Golddrähtchen hergestellt. Klassische Chips sind in einem rechteckigen Gehäuse mit zwei Reihen seitlich angebrachter Anschlussdrähte, den *Beinchen (*engl. *pin),* untergebracht. Die Anzahl der Außenverbindungen ist bei solchen Chips auf etwa 64 beschränkt. Chips mit mehr Anschlüssen (bis zu etwa 100) setzt man oft in ein quadratisches Gehäuse mit Anschlussdrähten an allen vier Seiten. Noch mehr Außenverbindungen schafft man durch Anbringung der Beinchen unter dem Chip. Durch diese *Pin Grid Array* (PGA) genannte Technik lassen sich Chips bauen, die mehrere hundert Verbindungen aufweisen können. Diese Technik wurde weiterentwickelt; heute üblich sind *Land Grid Arrays* (LGA). Die Anschlüsse sind auf einem *Sockel* angeordnet. Dieser hat federnde Kontaktstifte, das Prozessorgehäuse nurmehr Kontaktflächen, sogenannte Lands. Der erste Intel Pentium Prozessor hatte ein PGA mit 273 Pins, die ersten Versionen des Pentium-4 kamen auf 423 Pins. Der neueste Prozessor aus der Intel 8086 Serie, der Core 2 Duo E6700, hat ein LGA mit 775 Pins. Ein Core 2 Duo Prozessor ist auf S. 36 abgebildet.

Eine Leiterplatte ist in der Regel mit Chips unterschiedlicher Bauart bestückt und enthält zusätzlich einzelne klassische Bauelemente wie Kondensatoren oder Widerstände. Die Verdrahtung erfolgt meist in mehreren, mindestens jedoch zwei Verdrahtungsebenen. Diese stehen auf der Leiterplatte zur Verfügung, sind untereinander isoliert und haben Querverbindungen zu den anderen Ebenen. Werden mehrere Leiterplatten benötigt, sind diese meist senkrecht in eine Systemplatine (engl. *motherboard*) eingesteckt, die die Verbindungen enthält. Mit Anschlussbuchsen für genormte, mehrpolige Stecker kann ein Anschluss zu Netzteilen, externen Geräten etc. erfolgen.

In heutigen Computern finden sich meist eine oder mehrere Leiterplatten mit weniger als 50 Chips. In unmittelbarer Zukunft wird man durch höhere Integration die Anzahl der Chips in einem Rechner auf weniger als zehn reduzieren und zur selben Zeit die Leistung der Geräte um mehrere Größenordnungen steigern können.

5.1.2 Chipherstellung

Für die Herstellung eines Chips wird zunächst gereinigtes Silizium (Quarzsand) auf über tausend Grad erhitzt, bis es flüssig wird. Aus dieser Schmelze werden so genannte Einkristalle gezogen, die bis zu 2 m lang sein können und einen Durchmesser von etwa 20 cm haben. Sie werden nach dem Erkalten in dünne Scheiben gesägt und poliert. Diese Scheiben sind das Ausgangsmaterial für den Herstellungsprozess, in dem auf jeder dieser Scheiben einige hundert Chips in einem Arbeitsgang entstehen.

Komplexe Chips erfordern mehrere hundert Herstellungsschritte. Sie können viele Millionen individueller Transistoren enthalten. Für jeden Schritt kommt, in jeweils abgewandelter Form, ein fotolithografisches Grundverfahren zur Anwendung. Dabei wird jedesmal zunächst eine Materialschicht aufgetragen und mit Fotolack überzogen. Dieser wird mithilfe einer Maske, auf der die Chipstrukturen ausgespart sind, belichtet. Nach der Entwicklung werden die unbelichteten Stellen bearbeitet, das heißt entweder weggeätzt, dotiert oder mit Kontakten versehen. Dann wird der restliche Fotolack entfernt.

Um den in Abb. 5.1 gezeigten Transistor herzustellen, wird zunächst eine p-leitende Schicht erzeugt werden, darauf eine n-leitende Schicht, darauf eine Isolierschicht, dann eine polykristalline Schicht. Teile dieser Schichten werden jeweils fotolithografisch ausgespart, wieder verändert und schließlich mit metallischen Leitungen verdrahtet. Die wesentlichen Bearbeitungsvorgänge sind Oxidation, Diffusion zur Erzeugung von p- und n-leitenden Schichten, Ätzen und Metallisierung zur Erzeugung von leitenden Verbindungen und Kontakten.

Nach dem Aufbringen der Transistoren und Leiterbahnen entsteht auf den rechteckigen Siliziumscheiben, in einem durch die verwendeten kreisförmigen Masken definierten Gebiet, ein waffelartiges Muster einzelner Chips. Daher werden die Siliziumscheiben auch *Wafer* genannt. Sie werden zersägt, in die Gehäuse eingebaut und mit den Anschlussdrähten verbunden. Das Gehäuse wird endgültig verschlossen – und fertig ist der Chip.

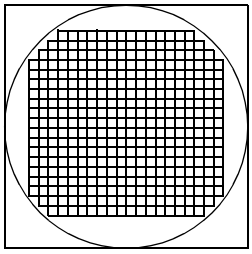


Abb. 5.3: *Ein Wafer*

Gegenwärtig ist Silizium der Rohstoff der Wahl für die Fertigung von Chips. Silizium ist billig und einfacher zu bearbeiten, als der alternative Rohstoff Galliumarsenid, aus dem man Chips mit erheblich kürzeren Schaltzeiten fertigen kann, die in Supercomputern und anderen kritischen Anwendungen gelegentlich eingesetzt werden.

5.1.3 Kleinste Chip-Strukturen

Ein wesentlicher Parameter bei der Chip-Herstellung ist die Größe der kleinsten Strukturen. Dabei handelt es sich um Leitungen, den Abstand zwischen zwei Leitungen oder um die Größe von Transistorzellen. Die kleinsten erzeugbaren Strukturen lagen bis vor kurzem im Bereich von etwa 1µ (1µ = 1 Mikrometer = 1/1000 mm). Lange Zeit lagen sie im Bereich von 100 nm bis 1µ (1nm = 1 Nanometer = 1/1000 µ). Zurzeit werden Strukturen von 65 nm bis 100 nm verwendet. Ein weiteres Absenken der kleinsten Strukturen auf 45 bzw. 32 nm ist für die nächsten Jahre geplant. Die Schwierigkeiten beim Verkleinern der Chip-Strukturen beste-

hen im Herstellen geeigneter Masken für die verschiedenen fotolithografischen Prozesse, in der exakten Positionierung der Masken, in Belichtungsproblemen, wenn die Wellenlänge des für die Belichtung verwendeten Lichts erreicht wird, und in mikroskopischen Ungenauigkeiten beim Ätzen, Beschichten etc.

Diese Schwierigkeiten konnten bisher immer wieder bewältigt werden. Meist waren dafür jedoch langwierige Forschungs- und Entwicklungsarbeiten erforderlich, so dass die kleinsten beherrschbaren Strukturen nur relativ langsam von 2 µ auf 1 µ und dann schrittweise auf 65 nm verkleinert werden konnten. Die weitere Verkleinerung auf Werte in der Größenordnung von 10 bis 50 nm wird weitere technische Innovationen erfordern. Als konsequente Fortsetzung der optischen Lithografie hin zu kürzeren Wellenlängen gilt z.B. die *EUV*-Lithografie (*Extreme Ultra Violet*). Dabei werden Wellenlängen im Bereich 13,5 nm genutzt, um Strukturen zwischen 45 nm und 32 nm und kleiner zu erzeugen.

5.1.4 Chipfläche und Anzahl der Transistoren

Die Herstellung von Chips ist ein langwieriger und fehleranfälliger Prozess. Der Anteil von funktionsfähigen Chips beträgt daher nur etwa 5 bis 50 %, bezogen auf die Gesamtproduktion, je nach der bereits gewonnenen Produktionserfahrung mit einem bestimmten Herstellungsprozess. Die Fehlerrate bei den einzelnen Chips ist von der Fläche des produzierten Chips abhängig. Um die Produktion wirtschaftlich zu machen, versucht man, die Chipfläche auf ein vertretbares Minimum zu reduzieren. Nur wenn es nicht anders geht, erhöht man die Chipfläche, um die Anzahl der Transistoren zu erhöhen. Gegenwärtig ändert sich die effektiv ausgenutzte Chipfläche von ca. 100 mm^2 nur wenig, da die Herstellungsprozesse so häufig verbessert werden, dass eine Vergrößerung der Chipfläche kaum notwendig ist.

Der Prozessor des Core 2 Duo E6700 verfügt über 291 Millionen Transistorfunktionen auf einer Fläche von 143 mm^2, beim Core 2 Duo E6400 sind es 167 Millionen Transistorfunktionen auf einer Fläche von nur 111 mm^2. Der wesentliche Unterschied beider Prozessoren ist die Größe des internen Cache, das größere Modell hat einen Cache von 4 MB, das kleinere halb so viel. Daraus kann man hochrechnen das der Prozessorkern nur vergleichsweise wenig Transistorfunktionen benötigt. Beide Modelle haben zwei Prozessorkerne, die vermutlich jeweils etwa 15 bis 20 Millionen Transistorfunktionen benötigen. Bei zukünftigen Generationen wird die Anzahl der Transistorfunktionen vermutlich weiter steigen – und für eine grössere Zahl von Prozessorkernen bzw. für noch mehr Cache-Speicher genutzt werden.

Gegenwärtig wird bei Speicher Chips mit ca. 100 mm^2 effektiver Nutzfläche in Anwendung einer 65 nm Technik eine Zahl von ca. 1 500 000 000 Transistoren erreicht. Über weitere Steigerungsmöglichkeiten kann man derzeit nur spekulieren.

5.1.5 Weitere Chip-Parameter

Je geringer die kleinsten Strukturen auf einem Chip sind, desto geringer sind die Schaltverzögerungen pro Transistor und der Energieverbrauch pro Schaltvorgang. Wenn dieser Energieverbrauch, der gegenwärtig ca. 1 pJ (Picojoule) beträgt, nicht um eine ganze Größenordnung

gesenkt werden könnte, wäre eine Erhöhung der Transistorzahl gar nicht möglich – die Chips würden zu heiß werden.

Während die Schaltverzögerung von NMOS-Transistoren etwa 0,8 ns (Nanosekunden) beträgt, ist dieser Wert bei neueren CMOS-Transistoren nur noch etwa 0,08 ns. Die Schnelligkeit einer ganzen Leiterplatte wird nicht nur durch die Geschwindigkeit der Transistoren in den Chips bestimmt, sondern auch durch die Zahl und die Länge der Verbindungen der verschiedenen Chips untereinander. Je mehr Transistoren in einem Chip untergebracht werden können, desto weniger Inter-Chip-Verbindungen sind erforderlich – um so schneller ist die Leiterplatte.

5.1.6 Speicherbausteine

Auch der Speicher eines Rechners ist aus Chips aufgebaut, den so genannten RAM-Chips. *RAM* ist die Abkürzung für den englischen Begriff *Random Access Memory* – zu deutsch: Speicher mit wahlfreiem Zugriff. Verwendet man die Ladung auf dem Gate eines Transistors zur Speicherung eines Bit, kommt man, zusammen mit der Adressierlogik, auf Speicherbausteine mit weniger als 1,5 Transistoren pro Bit. Allerdings verlieren diese *dynamischen* Speicherbausteine (*DRAM*) nach kurzer Zeit ihre Ladung wieder. Jedes Bit muss innerhalb einer bestimmten Zeit, die im Nanosekundenbereich liegt, wieder aufgefrischt, also gelesen und neu geschrieben werden. Eine Alternative ist die Verwendung *statischer* Speicherbausteine (*SRAM*). Diese müssen zwar nicht ständig aufgefrischt werden, benötigen aber mehrere Transistoren pro Bit. Sowohl dynamische als auch statische RAM-Chips verlieren die gespeicherte Information, wenn kein Strom vorhanden ist. Dies kann durch bestimmte, aufwändige Schaltungen oder durch Verwendung von Akku-Puffern verhindert werden. Heute werden dynamische RAM-Chips mit 256, 512 und 1024 MBit Speicherkapazität gefertigt, noch im Jahre 2007 werden 4 GBit RAM-Chips produziert werden – in absehbarer Zeit wird es bereits 8 GBit RAM-Chips geben. Die Entwicklungsgeschichte der Speicherbausteine illustriert Abbildung 5.4.

In den letzten Jahren sehr poulär geworden sind Flash-Speicher. Diese verwenden Speicherzellen, die die gespeicherte Information nicht verlieren, wenn kein Strom vorhanden ist. Diese beruhen auf dem *EEPROM* Prinzip (*Electrically Erasable Programmable Read-Only Memory*). Derartige Speicherbausteine können genauso einfach gelesen werden, wie normale Speicherzellen. Das Schreiben erfordert allerdings einen speziellen Mechanismus, das *Umprogrammieren* der Speicherzellen durch Anlegen einer relativ hohen Spannung. Das Schreiben erfolgt meist blockweise und ist sehr viel langsamer als das Schreiben in normale Speicherzellen. Flash-Speicher benutzen Bausteine mit spezieller EEPROM Technolgie und einer Blockgröße für Schreiboperationen von typischerweise 64 Bit. Derzeit sind preiswerte Flash-Speicher mit einer Kapazität von 512 MB bis 4 GB im Angebot. Bei höherer Kapazität ist der Preis derzeit noch überproportional größer. Derartige Speicherbausteine werden verwendet in MP3-Playern, als Speicherkarte für Kameras und in den mittlerweile ubiquitären USB-Sticks.

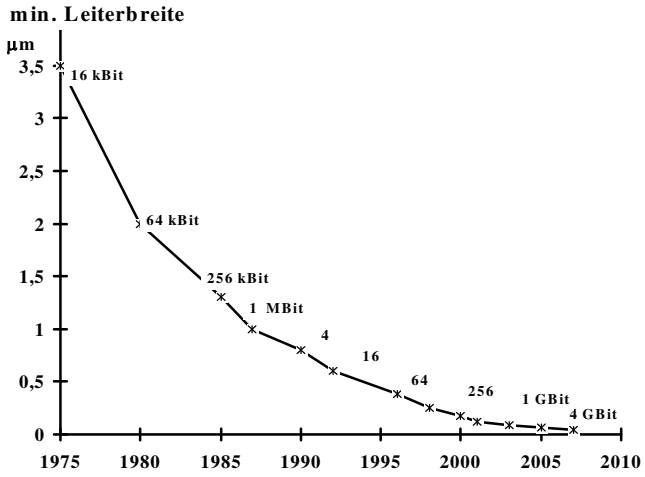


Abb. 5.4: *Entwicklung von Speicherchips*

5.1.7 Logikbausteine

Speicherbausteine bestehen aus vielen gleichartigen Speicherzellen und aus einer Lese- und Schreiblogik. Entsprechende Schaltpläne können in jeder Größe relativ rasch angefertigt werden. Daher verwundert es nicht, dass mit einem neuen Herstellungsprozess, der eine bestimmte Maximalzahl von Transistoren ermöglicht, als erstes Speicherbausteine gebaut werden können. Anders sieht es bei den *Logikbausteinen* aus. Hierbei handelt es sich um Mikroprozessoren oder um sonstige Spezialschaltungen. Die entsprechenden Schaltpläne sind nicht so einfach herzustellen wie die der Speicherbausteine. Sie bestehen nicht aus immer wieder kopierten Speicherzellen, sondern im Extremfall aus lauter unterschiedlichen Funktionsgruppen. Im Fall der ersten Mikroprozessoren, die nur über wenige Tausend Transistorfunktionen verfügten, konnten die entsprechenden Schaltpläne noch am Reißbrett entworfen werden. Spätere Mikroprozessoren, wie der Intel 8086 mit ca. 30 000 und der Motorola 68000 mit 68 000 Transistorfunktionen, konnten mit den damaligen Werkzeugen nur noch entwickelt werden, weil Teile des Mikroprozessors wiederum als Speicher ausgelegt waren – als *Mikroprogrammspeicher*. Der Übergang zu höher integrierten Schaltungen, wie z.B. der des 80286 mit 150 000 Transistoren, war nur mit computerunterstützten Methoden möglich. Heute stehen ausgereifte Werkzeuge zum Entwurf hochintegrierter Logikbausteine zur Verfügung, mit denen Mikroprozessoren, wie z.B. der Pentium-4-Prozessor mit seinen ca. 42 Millionen Transistorfunktionen und der neuere Core 2 Duo E6700 Prozessor mit seinen 291 Millionen Transistorfunktionen, entwickelt werden können.

5.1.8 Schaltungsentwurf

Die wesentlichen Hilfsmittel für den Entwurf hochintegrierter Schaltungen sind CAD-Systeme und Simulationsprogramme. CAD steht für *Computer Aided Design* – zu deutsch etwa:

Entwurfsverfahren mithilfe von Rechnern. Von Hand gezeichnete Schaltpläne für einige 100 000 Transistoren würden so groß wie Fußballfelder sein. Solche Schaltungen können daher nur noch mit elektronischen Entwurfssystemen beherrscht werden. Ähnliches gilt auch für das Testen der Schaltung. Früher wurde ein Prototyp hergestellt und dieser anschließend getestet. Die Herstellung von Chip-Prototypen ist jedoch sehr aufwändig. Es kann Monate dauern, bis ein Prototyp fertig ist, nur um dann nach Stunden wegen eines Fehlers verworfen zu werden.

Bei einem Chip mit einigen 1000 Transistoren kann die Schaltung im Elektroniklabor mithilfe von Oszillografen getestet werden. Wenn einige 100 000 oder sogar einige 100 000 000 Transistoren auf Funktionstüchtigkeit und hinsichtlich ihres Zusammenspiels getestet werden müssen, ist ein solches Verfahren nicht mehr möglich. Aus diesen Gründen verlagert man die Produktion der Prototypen von der Hardware in die Software. Statt eines physischen Probanden wird ein abstraktes Modell des zukünftigen Chips definiert und dessen Verhalten auf einem Rechner simuliert.

Dennoch verbleibt das Risiko von unentdeckten Entwurfsfehlern. Wie das Beispiel des „Pentium-FDIV-Bug“ zeigt, wurde ein Fehler in dem Pentium-Prozessor weder bei den Simulationen vor Produktionsbeginn noch beim Testen der ersten Serien von Prozessoren entdeckt. Erst anderthalb Jahre nach Beginn der Serienproduktion kam dieser Fehler mehr oder weniger zufällig ans Licht.

Der Entwurf hochintegrierter Transistorschaltungen ist ein ähnlich anspruchsvolles Problem wie der Entwurf und die Programmierung eines Softwaresystems, das aus mehreren hochkomplexen Programmen besteht. Es verwundert daher nicht, dass in beiden Fällen ähnliche Techniken angewendet werden.

Der modulare Entwurf: Ein komplexes System wird in mehrere einfachere Module mit klaren Schnittstellen zerlegt. Ein hochintegrierter Chip besteht häufig aus einzelnen Modulen, die über Schaltungskanäle verdrahtet sind.

Standardschaltungen: Für bestimmte wiederkehrende Aufgaben werden immer die gleichen Schaltungen verwendet, die in *Zellbibliotheken* verwaltet werden. Diese Zellbibliotheken bestehen aus logischen Standardschaltungen und deren Implementierung, jeweils in einem bestimmten Herstellungsprozess.

Nach wie vor ist die Struktur vieler heutiger Mikroprozessoren so, dass möglichst viele Funktionen in *Mikroprogramme* verlegt werden, die wiederum in Speicherzellen abgelegt werden. So haben heutige Mikroprozessoren einen Anteil von 20 bis 50 % an Transistoren mit Speicherfunktion.

Ein alternativer Weg zur Vereinfachung von Mikroprozessoren wird von den noch zu diskutierenden *RISC-Prozessoren* eingeschlagen. Bei RISC-Prozessoren wird der Befehlssatz so weit vereinfacht, dass man mit sehr wenigen Mikroprogrammen auskommt. Durch die konsequente Verwendung von *regulären Strukturen* erreicht man eine Vereinfachung des Schaltungsentwurfes und damit eine schnellere Anwendung eines moderneren Herstellungsprozesses auch für Logikschaltungen.

5.2 Boolesche Algebra

Die *boolesche Algebra* entstand aus den Arbeiten des Engländers *George Boole* (1815–1864), dessen eigentliches Ziel es war, die Logik formal zu begründen. Es ging darum, die Wahrheit oder Falschheit von Aussagen zweifelsfrei feststellen zu können, ähnlich wie man auch das Ergebnis einer Addition oder Multiplikation ausrechnen kann. Die Objekte, mit denen Boole operierte, waren *Wahrheitswerte* (*wahr* und *falsch*) doch kann man sie genauso gut als Bitwerte (**0** und **1**) oder Stromzustände (Strom fließt/Strom fließt nicht) interpretieren.

5.2.1 Serien-parallele Schaltungen

Information wird in einem Rechner letztlich durch eine Folge von Bits realisiert. Jedes Bit kann zwei Zustände haben, die wir mit **0** und **1** bezeichnen. Technisch können diese Zustände durch Spannungen realisiert werden, z.B. **0** durch eine Spannung zwischen 0,0 und 0,4 V und **1** durch eine Spannung zwischen 2,4 und 5,0 V.

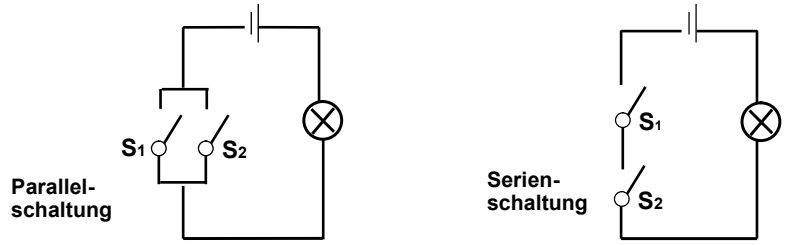
In einem einfachen Stromkreis, bestehend aus einer Batterie B, einem Widerstand R und einem Schalter S, können wir z.B. ein Bit durch die an dem Widerstand anliegende Spannung darstellen: Ist der Schalter geöffnet, so ist die anliegende Spannung 0, ist er geschlossen, so ist die Spannung ca. 5 V und stellt nach Vereinbarung das Bit **1** dar.

Abb. 5.5: *Schaltungen*

Vereinfachend könnte man auch den Widerstand durch ein Lämpchen ersetzen. Die Bitwerte **1** und **0** werden dann dadurch realisiert, dass das Lämpchen brennt oder erlischt (vorgehende Abbildung). Das Verhalten des Kreises kann man in einer Tabelle darstellen. Wenn wir die Stellungen des Schalters, offen bzw. geschlossen, mit **0** bzw. **1** bezeichnen, erhalten wir die folgende Tabelle für den Zustand der Lampe L:

S	L
0	0
1	1

Ersetzen wir den Schalter S durch zwei Schalter, S_1 und S_2, so ergeben sich zwei Kombinationsmöglichkeiten, die Parallelschaltung und die Serienschaltung:



Die zugehörigen Schalttabellen beschreiben alle möglichen Stellungen von S_1 und S_2 zusammen mit dem Ergebnis, das wir in Spalte L angeben:

Parallelschaltung

S_1	S_2	L
0	**0**	0
0	**1**	1
1	**0**	1
1	**1**	1

Serienschaltung

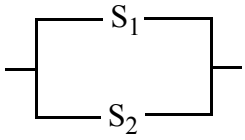
S_1	S_2	L
0	**0**	0
0	**1**	0
1	**0**	0
1	**1**	1

Abb. 5.6: *Parallelschaltung und Serienschaltung*

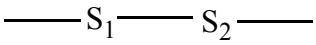
Wir können auf diese Weise fortfahren und weitere Kombinationen von Schaltern untersuchen. Dabei werden wir den von Boole entwickelten und heute nach ihm benannten mathematischen Kalkül, die *boolesche Algebra*, kennenlernen. Boole hat ihn in seinem Werk *An Investigation into the laws of thought* ursprünglich als eine *Algebra der Logik* entwickelt. Die boolesche Algebra ist aber auch in vielen anderen Bereichen anwendbar, insbesondere bei der Konstruktion digitaler Schaltkreise und in der Mengenalgebra.

5.2.2 Serien-parallele Schaltglieder

Wir untersuchen das Verhalten von Schaltgliedern, die aus einfacheren Schaltern zusammengebaut sind. Beginnend mit Elementarschaltern bauen wir neue Schaltglieder durch *Serien-* bzw. *Parallelschaltung*. Jedes Schaltglied hat einen Eingang und einen Ausgang. Sind S_1 und S_2 Schaltglieder, so erhält man durch Parallelschaltung das Schaltglied $S_1 + S_2$ und durch Serienschaltung das Schaltglied $S_1 * S_2$. Bei der Parallelschaltung genügt es, wenn einer der Schalter, S_1 *oder* S_2, eingeschaltet ist, damit die gesamte Schaltung Strom durchlässt, bei der Serienschaltung ist es notwendig, dass S_1 *und* S_2 eingeschaltet sind. Daher wird die Parallelschaltung auch als *Oder-Schaltung*, die Serienschaltung als *Und-Schaltung* bezeichnet. Durch fortgesetzte Kombination von Schaltkreisen durch Serien- oder durch Parallelschaltung erhalten wir beliebig komplexe Schaltkreise, die *Serien-Parallel-Kreise*.



Parallelschaltung



Serienschaltung

Das Verhalten von zusammengesetzten Schaltungen kann man in einer Tabelle beschreiben. Dabei interessiert uns lediglich, ob bei einer bestimmten Stellung der Elementarschalter das gesamte Schaltglied ein- oder ausgeschaltet ist. Mit der Abkürzung Ein = 1, Aus = 0 erhalten wir die Operationstafeln für $S_1 + S_2$ und $S_1 * S_2$:

+	0	1
0	0	1
1	1	1

*	0	1
0	0	0
1	0	1

Abb. 5.7: *Oder- und Und-Verknüpfung*

5.2.3 Boolesche Terme

Bezeichnet man die elementaren Ein-Aus-Schalter mit Variablen *x*, *y*, *z*, …, so lässt sich jeder serien-parallele Schaltkreis durch einen serien-parallel Term (kurz *SP-Term*) beschreiben. Diese sind induktiv folgendermaßen definiert:

Definition (SP-Terme):

(i) 0 und 1 sind SP-Terme
(ii) Jede Variable *x*, *y*, *z*, … ist ein SP-Term.
(iii) Sind t_1 und t_2 SP-Terme, so auch $t_1 + t_2$ und $t_1 * t_2$.

0 bzw. 1 stehen in dieser Definition für Schalter, die immer offen bzw. immer geschlossen sind. Die Operationszeichen + und * haben natürlich eine andere Bedeutung als gewohnt. Um eine Verwechslung auszuschließen, benutzt man statt ihrer auch die Zeichen $\vee$ und $\wedge$ oder schreibt sie aus als **OR** und **AND**. Zusätzlich erlauben wir Klammern, um die Entstehung eines Terms aufgrund der definierenden Regeln klarzustellen.

Das Schaltglied, das durch einen SP-Term beschrieben wird, lässt sich schrittweise ermitteln. Für den Term $x*(y + z)$ baut man zunächst den inneren Teilterm $(y + z)$ aus den Schaltern *y* und *z* zusammen und schaltet den erhaltenen Kreis in Serie mit dem Schalter *x*:

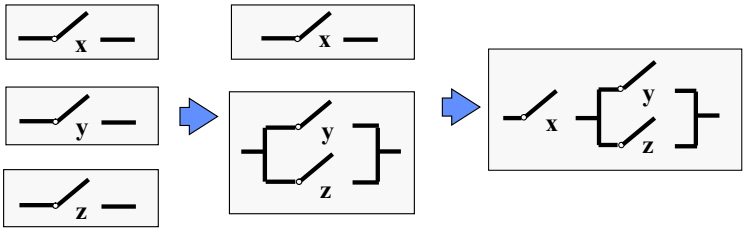


Abb. 5.8: *Aufbau des zum Term x*(y+z) gehörenden Schaltkreises*

Ähnlich erstellt man schrittweise die zugehörige Schalttabelle:

x y z	y + z	x*(y+z)
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	1	0
1 0 0	0	0
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

Abb. 5.9: *Zum Term x*(y+z) gehörende Schalttabelle*

5.2.4 Schaltfunktionen

Eine *Schaltfunktion* ist eine *n*-stellige Operation auf der Menge {0, 1}, also eine Abbildung $f: \{0,1\}^n \rightarrow \{0,1\}$. Jeder SP-Term beschreibt mittels seiner Schalttabelle eine Schaltfunktion. Der Term $x^*(y + z)$ realisiert z.B. die Funktion $f: \{0,1\}^3 \rightarrow \{0,1\}$ mit:

$$f(0,0,0) = 0 \quad f(0,0,1) = 0 \quad f(0,1,0) = 0 \quad f(0,1,1) = 0$$
$$f(1,0,0) = 0 \quad f(1,0,1) = 1 \quad f(1,1,0) = 1 \quad f(1,1,1) = 1$$

Verschiedene Terme können durchaus dieselbe Schaltfunktion beschreiben, wie zum Beispiel im Fall der Terme $t_1 = x^*(y + z)$ und $t_2 = (x^*y) + (x^*z)$. Dies erkennt man durch Tabellierung der zugehörigen Schaltfunktionen oder durch Analyse der zugehörigen Schaltkreise.

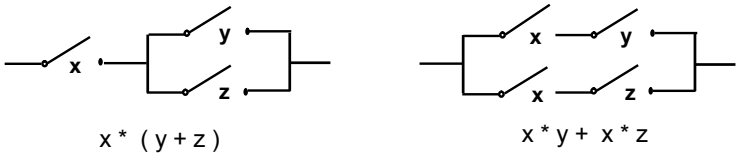


Abb. 5.10: *Zwei Schaltungen mit gleicher Schaltfunktion*

Man beachte, dass in einem Term, wie z.B. $x*y + x*z$, eine Variable mehrfach vorkommen kann. Für den entsprechenden Schaltkreis stellen wir uns vor, dass Schalter mit gleichem Namen so gekoppelt sind – mechanisch oder elektrisch –, so dass sie immer im gleichen Schaltzustand sind.

5.2.5 Gleichungen

Eine Gleichung $t_1 = t_2$ besteht aus zwei Termen, die dieselbe Schaltfunktion beschreiben. Um nachzuweisen, dass eine Gleichung $t_1 = t_2$ gilt, kann man daher die Schaltfunktionen der beiden Terme tabellieren und die Ergebnisse vergleichen. Während die Gleichung $x*(y + z) = x*y + x*z$ noch vertraut aussieht, überrascht vielleicht die folgende Gleichung:

$$x + (y*z) = (x + y)*(x + z).$$

Durch Tabellierung erhalten wir aber identische Ergebnisse für alle Belegungen der Variablen:

x y z	y * z	**x+(y*z)**	(x+y)	(x+z)	**(x+y)*(x+z)**
0 0 0	0	**0**	0	0	**0**
0 0 1	0	**0**	0	1	**0**
0 1 0	0	**0**	1	0	**0**
0 1 1	1	**1**	1	1	**1**
1 0 0	0	**1**	1	1	**1**
1 0 1	0	**1**	1	1	**1**
1 1 0	0	**1**	1	1	**1**
1 1 1	1	**1**	1	1	**1**

Abb. 5.11: *Vergleich der Schaltfunktionen x+(y*z) und (x+y)*(x+z)*

Es gibt eine recht überschaubare Menge von gültigen Gleichungen, aus denen sich alle anderen Gleichungen ableiten lassen. Eine Struktur, die diese Gleichungen erfüllt, heißt *distributiver Verband*.

$x + x = x$	Idempotenz	$x*x = x$
$x + y = y + x$	Kommutativität	$x*y = y*x$
$x + (y + z) = (x + y) + z$	Assoziativität	$x*(y*z) = (x*y)*z$
$x*(x + y) = x$	Absorption	$x + (x*y) = x$
$x*(y + z) = (x*y) + (x*z)$	Distributivität	$x + (y*z) = (x + y)*(x + z)$

Abb. 5.12: *Gleichungen eines distributiven Verbandes*

Es fällt sofort auf, dass sich die Gleichungen in der linken und in der rechten Spalte entsprechen, sofern man + durch * und * durch + ersetzt. Dieses Phänomen ist unter dem Namen *Dualität* bekannt. Die Terme 0 und 1, die den immer geöffneten, bzw. den immer geschlossenen Schaltkreis bezeichnen, werden durch die folgenden Gleichungen charakterisiert. Auch

x + 0 = x	x * 1 = x
x + 1 = 1	x * 0 = 0

hier entsprechen sich die Gleichungen, sofern man zusätzlich 0 mit 1 vertauscht. Vertauschen wir + und * sowie 0 und 1 in einem Term *t,* so erhalten wir den zu *t dualen Term*, den man mit t^d bezeichnet. Durch zweimaliges Dualisieren erhalten wir den alten Term zurück. Bei den Gleichungen haben wir bereits gesehen, dass mit jeder Gleichung $t_1 = t_2$ auch die duale Gleichung $t_1^d = t_2^d$ gilt. Dieses *Dualitätsprinzip* setzt sich auch auf alle Gleichungen fort, die wir aus den Basisgleichungen in Abb. 5.12 folgern können. Wir haben also immer $t^{dd} = t$, und zu jeder Gleichung $t_1 = t_2$ automatisch auch die duale Gleichung $t_1^d = t_2^d$.

5.2.6 SP-Schaltungen sind monoton

Wir wissen schon, dass jeder Term eine Schaltfunktion realisiert, haben aber noch nicht geklärt, ob wir auch jede denkbare Schaltfunktion durch einen Term beschreiben können. Ohne die *Negation*, also die Schaltfunktion *not* mit *not*(0) = 1 und *not*(1) = 0, wird dies noch nicht der Fall sein. Insbesondere kann man die Negation selber nicht durch eine SP-Schaltung realisieren. Dies erscheint physikalisch plausibel, aber wie können wir es nachweisen?

Wir werden diese Frage lösen, indem wir allgemeiner zeigen, dass die Schaltfunktion eines jeden SP-Schaltkreises *monoton* ist. Physikalisch kann man das so interpretieren, dass ein geschlossenes Schaltglied durch Schließen weiterer Schalter nicht geöffnet werden kann. Ist also $f: \{0,1\}^n \to \{0,1\}$ die durch ein SP-Schaltglied realisierte Schaltfunktion und gilt $f(b_1, \ldots, 0, \ldots, b_n) = 1$, so muss auch $f(b_1, \ldots, 1, \ldots, b_n) = 1$ sein.

Setzt man die natürliche Ordnung $0 \le 1$ komponentenweise auf $\{0,1\}^n$ fort durch

$$(b_1, \ldots, b_n) \le (c_1, \ldots, c_n) \quad :\Leftrightarrow \quad b_i \le c_i \text{ für alle } i \text{ mit } 1 \le i \le n,$$

so erkennen wir, dass jede durch einen SP-Schaltkreis realisierte Schaltfunktion f im folgenden Sinne *monoton* sein muss:

$$(b_1, \ldots, b_n) \le (c_1, \ldots, c_n) \Rightarrow f(b_1,\ldots, b_n) \le f(c_1,\ldots, c_n).$$

Diese intuitiv einleuchtende Tatsache wollen wir durch Induktion über den Aufbau von Termen nachweisen. Den Induktionsanfang bilden hierbei die Terme 0 und 1 sowie die Variablen x_i. Für den Induktionsschritt nehmen wir an, die Behauptung sei für t_1 und für t_2 schon bewiesen und zeigen, dass sie dann auch für $t_1 + t_2$ und für $t_1 * t_2$ gilt.

Für die konstanten Terme 0 und 1 ist die Behauptung in der Tat trivial. Die zu 0 gehörende konstante Schaltfunktion 0 z.B. erfüllt $0(b_1, \ldots, b_n) = 0 \le 0 = 0(c_1,\ldots, c_n)$.

Der Variablen x_i entspricht gerade die Schaltfunktion mit $x_i(b_1, \ldots, b_n) = b_i$. Sie ist offensichtlich monoton, denn aus $(b_1, \ldots, b_n) \le (c_1,\ldots, c_n)$ folgt nach Definition $b_i \le c_i$, was wiederum gleichbedeutend ist mit $x_i(b_1, \ldots, b_n) \le x_i(c_1,\ldots, c_n)$.

Sind t_1 und t_2 bereits als monoton nachgewiesen, und sei $(b_1, \ldots, b_n) \le (c_1,\ldots, c_n)$, dann gilt:

$$\begin{aligned}(t_1 + t_2)(b_1, \ldots, b_n) &= t_1(b_1, \ldots, b_n) + t_2(b_1, \ldots, b_n)\\ &\le t_1(c_1,\ldots, c_n) + t_2(c_1,\ldots, c_n)\\ &= (t_1 + t_2)(c_1,\ldots, c_n).\end{aligned}$$

Entsprechendes finden wir auch für $t_1{*}t_2$. Dabei haben wir die Monotonie von + und * ausgenutzt, die man leicht von den Operationstabellen abliest:

$$b_1 \le c_1,\ \ b_2 \le c_2 \Rightarrow (b_1 + b_2) \le (c_1 + c_2) \text{ und } (b_1{*}b_2) \le (c_1{*}c_2).$$

5.2.7 Negation

Ein einfaches Beispiel einer nützlichen Schaltung, die nicht mehr durch eine SP-Schaltung realisierbar ist, ist eine *Wechselschaltung*. Dabei soll eine Lampe L von zwei verschiedenen Schaltern unabhängig ein- und ausgeschaltet werden können. Seien die Schalter x und y, so ist die Funktionsweise durch eine der folgenden Tabellen gegeben. Nachdem wir den Wert von L für $x = y = 0$ festgelegt haben, ergeben sich die restlichen Einträge aus der Forderung, dass bei jeder Veränderung eines der Schalter sich der Zustand der Lampe verändern muss.

x	y	Lampe
0	**0**	0
0	**1**	1
1	**0**	1
1	**1**	0

x	y	Lampe
0	**0**	1
0	**1**	0
1	**0**	0
1	**1**	1

Abb. 5.13: *Mögliche Schaltfunktionen für Wechselschalter*

Beide Schaltfunktionen verletzen die Forderung der Monotonie, können daher nicht allein durch SP-Schaltglieder realisiert werden.[1]

Die einfachste nicht monotone Schaltfunktion ist durch die folgende Tabelle gegeben. Wenn der Schalter x offen ist, ist das Schaltglied geschlossen – und umgekehrt. Das entsprechende Schaltglied ist die *Negation*:

1. Elektriker benutzen für die Realisierung einer Wechselschaltung keine *Ein-Aus-Schalter*, sondern *Wechselschalter*, die eine Eingangsleitung mit einer von zwei Ausgangsleitungen verbinden.

x	x‘
0	1
1	0

Abb. 5.14: *Schalttabelle für Negation*

Definition: *Ist S ein Schaltglied, so sei S' dasjenige Schaltglied, das genau dann offen ist, wenn S geschlossen ist. S' heißt die* Negation *von S.*

Für die zweifache Negation gilt offensichtlich: $S'' = S$. Andere Namen für die Negation sind auch: *Komplement* oder *Inverses*.

In elektrischen Schaltkreisen lässt sich die Negation durch ein *Relais* realisieren: Fließt Strom durch *S*, so wird durch die Magnetwirkung einer Spule der Schalter *S'* geöffnet. Mit Transistoren gelingt die Realisierung der Negation einfacher und natürlicher, siehe Abschnitt 5.3.

5.2.8 Boolesche Terme

Ein Schaltkreis, in dem neben Serien- und Parallel-Schaltung auch noch die Negation verwendet werden darf, heißt *boolesche Schaltung*. Der einer booleschen Schaltung entsprechende Term heißt *boolescher Term*. Formal definieren wir:

Definition (*Boolesche Terme*)**:**

(i) 0 *und* 1 *sind boolesche Terme.*
(ii) *Jede Variable ist ein boolescher Term.*
(iii) *Sind t, t_1 und t_2 boolesche Terme, so auch: $t_1 + t_2$, $t_1 * t_2$* und *t'*.

Die Gleichheit boolescher Terme definiert man analog zu der Gleichheit von SP-Termen: Zwei Terme heißen gleich, wenn sie identische Schaltfunktionen besitzen. Als Beispiel einer booleschen Gleichung betrachten wir die *deMorgansche Regel*

$$(x + y)' = x' * y'$$

und ihre Herleitung durch Vergleich der entsprechenden Spalten der Schaltfunktionen.

x	y	x+y	(x+y)‘	x‘	y‘	x‘*y‘
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Abb. 5.15: *Ablesen der Gleichheit $(x+y)' = x' * y'$ aus der Wertetabelle*

Die wichtigsten Gleichungen, die das Verhalten der Negation bestimmen, sind:

$(x + y)' = x'*y'$	deMorgansche Regeln	$(x*y)' = x' + y'$
$x + x' = 1$	Komplementregeln	$x*x' = 0$
	$x'' = x$	

Eine algebraische Struktur, in der neben den Gleichungen eines distributiven Verbandes und den Gleichungen für 0 und 1 auch noch die obigen Komplementgleichungen gelten, heißt *boolesche Algebra*.

5.2.9 Dualität

Auch zu einem booleschen Term t gewinnt man den dualen Term t^d durch Vertauschen von + mit * und 0 mit 1. Der duale Term zu $t = x*(y + 0)'$ ist also $t^d = x + (y*1)'$. Wie hängen die Schaltfunktionen zu t und zu t^d zusammen?

Behauptung: *Für beliebige* $(b_1, \ldots, b_n) \in \{0,1\}^n$ *ist* $t(b_1, \ldots, b_n)' = t^d(b_1', \ldots, b_n')$.

Wir beweisen die Behauptung durch Induktion über den Aufbau boolescher Terme.

(i) $t = 0$ ($t = 1$ analog):

$0(b_1, \ldots, b_n)' = 0' = 1 = 1(b_1', \ldots, b_n') = 0^d(b_1', \ldots, b_n')$

(ii) $t = x_i$:

$x_i(b_1, \ldots, b_n)' = b_i' = x_i(b_1', \ldots, b_n') = x_i^d(b_1', \ldots, b_n')$

(iii) $t = t_1 + t_2$ ($t = t_1*t_2$ analog):

$$
\begin{aligned}
(t_1 + t_2)(b_1, \ldots, b_n)' &= (t_1(b_1, \ldots, b_n) + t_2(b_1, \ldots, b_n))' \\
&= t_1(b_1, \ldots, b_n)' * t_2(b_1, \ldots, b_n)' && \text{(deMorgan)} \\
&= t_1^d(b_1', \ldots, b_n') * t_2^d(b_1', \ldots, b_n') && \text{(Ind.-Annahme)} \\
&= (t_1^d * t_2^d)(b_1', \ldots, b_n') \\
&= (t_1 + t_2)^d(b_1', \ldots, b_n').
\end{aligned}
$$

Aus der gerade bewiesenen Behauptung ergibt sich sofort, dass mit einer Gleichung $t_1 = t_2$ auch immer die duale Gleichung $t_1^d = t_2^d$ richtig ist. Diese Folgerung hätten wir auch schon aus den definierenden Gleichungen einer booleschen Algebra ablesen können, denn offensichtlich enthalten diese zu jeder Gleichung auch die entsprechende duale Gleichung.

5.2.10 Realisierung von Schaltfunktionen

In der Praxis stellt sich häufig das Problem, zu einer gegebenen Schaltfunktion einen entsprechenden booleschen Term zu finden, der diese Schaltfunktion realisiert. Dazu betrachten wir zunächst spezielle boolesche Terme, so genannte *Literale* wie z.B. x, x', x_1, x_3' und *Monome*, wie z.B. $x'yz', xy'z$ oder $x_1'x_2'x_3x_4$. Wir haben hier, wie auch in der Arithmetik üblich, das Multiplikationszeichen weggelassen, d.h. wir schreiben kurz: t_1t_2 für t_1*t_2.

Definition: *Ein* Literal *ist eine Variable oder eine negierte Variable. Ein* Monom *ist ein Produkt von Literalen.*

Die Schaltfunktion eines Monoms kann nur dann 1 sein, wenn jedes darin vorkommende Literal 1 ist, d.h. wenn jede vorkommende Variable mit 1 und jede vorkommende negierte Variable mit 0 belegt ist. $x'yz'$ ist also nur dann 1, falls $x = z = 0$ und $y = 1$ sind. Der boolesche Term, der aus der Summe zweier Monome besteht, hat genau dort eine 1, wo mindestens eines der Monome eine 1 hat. So hat der Term $t = x'yz' + xy'z$ eine 1 für $x = z = 0$, $y = 1$ sowie für $x = z = 1$, $y = 0$. Durch Summierung von geeigneten Monomen kann man also an beliebigen Stellen einer Schaltfunktion eine 1 realisieren.

Als Beispiel sei eine Schaltfunktion gesucht, die es gestattet, eine Lampe von drei verschiedenen Schaltern x, y und z unabhängig ein- und auszuschalten. Die gesuchte Schaltfunktion $g(x, y, z)$ ist in der linken Tabelle spezifiziert:

x	y	z	g(x,y,z)
0	**0**	**0**	0
0	**0**	**1**	1
0	**1**	**0**	1
0	**1**	**1**	0
1	**0**	**0**	1
1	**0**	**1**	0
1	**1**	**0**	0
1	**1**	**1**	1

x	y	z	m_1	m_1	m_3	m_4	$m_1+m_2+m_3+m_4$
0	**0**	**0**	0	0	0	0	0
0	**0**	**1**	**1**	0	0	0	**1**
0	**1**	**0**	0	**1**	0	0	**1**
0	**1**	**1**	0	0	0	0	0
1	**0**	**0**	0	0	**1**	0	**1**
1	**0**	**1**	0	0	0	0	0
1	**1**	**0**	0	0	0	0	0
1	**1**	**1**	0	0	0	**1**	**1**

Abb. 5.16: *Schaltfunktion - Spezifikation und Realisierung als Summe von Monomen*

Diese Schaltfunktion liefert an vier Stellen den Wert 1, sie lässt sich also als Summe von vier Monomen m_1, m_2, m_3, m_4 schreiben. Die benötigten Monome sind:

$$m_1 = x'y'z,\ m_2 = x'yz',\ m_3 = xy'z' \text{ und } m_4 = xyz.$$

Der gesuchte boolesche Term ist daher:

$$g(x, y, z) = m_1 + m_2 + m_3 + m_4 = x'y'z + x'yz' + xy'z' + xyz.$$

Der durch die obige Vorgehensweise gebildete Term hat eine spezielle Form, die man auch als *disjunktive Normalform* (DNF) bezeichnet. Darunter versteht man eine Summe von Monomen, wobei verlangt ist, dass jede Variable in jedem Monom (entweder direkt oder negiert) vorkommt. Zu jeder Schaltfunktion gibt es dann genau eine disjunktive Normalform und diese lässt sich auf die oben beschriebene Weise gewinnen.

Für praktische Zwecke ist es sinnvoll, die gewonnene disjunktive Normalform noch zu vereinfachen. In dem obigen Beispiel erhalten wir durch Ausklammern:

$$g(x, y, z) = (x'y' + xy)z + (x'y + xy')z'.$$

Das Verfahren ist offensichtlich für jede Schaltfunktion durchführbar, so dass gilt:

Jede Schaltfunktion lässt sich durch einen booleschen Term realisieren.

5.2.11 Konjunktive Normalform

Die vorgestellte Methode liefert für jede Schaltfunktion einen booleschen Term, welcher um so komplizierter ist, je mehr 1-en die Schaltfunktion hat. Das Dualitätsprinzip deutet eine zweite Vorgehensweise an, die immer dann sinnvoll ist, wenn die Schaltfunktion mehr Einsen als Nullen hat.

Wir definieren eine *Elementarsumme* als Summe von Literalen. Die Schaltfunktion einer Elementarsumme ergibt genau für einen Input eine 0, sonst immer 1. Sind e_1 und e_2 Elementarsummen, so hat das Produkt e_1e_2 genau dort eine 0, wo e_1 oder e_2 eine 0 haben. Jede Schaltfunktion kann man als Produkt von Elementarsummen schreiben.

Beispiel: Gegeben sei die Schaltfunktion $h(x, y, z)$, durch die Werte in der vierten Spalte der folgenden Tabelle.

x y z	h(x,y,z)
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	1

x y z	e_1	e_2	e_3	$e_1 * e_2 * e_3$
0 0 0	**0**	1	1	**0**
0 0 1	1	1	1	1
0 1 0	1	1	1	1
0 1 1	1	**0**	1	**0**
1 0 0	1	1	1	1
1 0 1	1	1	**0**	**0**
1 1 0	1	1	1	1
1 1 1	1	1	1	1

Abb. 5.17: *Schaltfunktion und Darstellung als konjunktive Normalform*

Die drei Nullwerte geben Anlass für drei Elementarsummen $e_1 = x + y + z$, $e_2 = x + y' + z'$ und $e_3 = x' + y + z'$. Sie sind in der gleichen Tabelle dargestellt. Ihr Produkt ergibt den gesuchten booleschen Term:

$$h(x, y, z) = e_1e_2e_3 = (x + y + z)(x + y' + z')(x' + y + z').$$

Den so gewonnenen Term nennen wir auch *konjunktive Normalform*. Wiederum ist diese eindeutig, für praktische Zwecke ist es aber sinnvoll, die Darstellung zu vereinfachen.

5.2.12 Umwandlung in DNF oder KNF

Die Gesetze der booleschen Algebra erlauben auch eine direkte Umwandlung eines Terms in seine disjunktive bzw. seine konjunktive Normalform. Dies geschieht in drei Schritten, die wir an dem Beispielterm $t = (x * y)' * (y * z')' + (y' + (z + x))'$ nachvollziehen wollen:

– deMorgansche Regeln anwenden,
$t = (x * y)' * (y * z')' + (y' + (z + x))'$
$= (x' + y') * (y' + z'') + y'' * (z + x)'$
$= (x' + y') * (y' + z'') + y'' * z' * x'$

– doppelte Negationen entfernen,
$t = (x' + y') * (y' + z) + y * z' * x'$

– ausdistribuieren (für DNF mit $a*(b+c)=a*b+a*c$, für KNF mit $a+b*c=(a+b)*(a+c)$),
$t = (x' + y') * (y' + z) + y * z' * x'$
$= x' * y' + x' * z + y' * y' + y' * z + y * z' * x'$
$= x'y' + x'z + y'y' + y'z + y z'x'$

– gleiche Terme zusammenfassen und umordnen,
$t = x'y' + x'z + y' + y'z + x'y z'$

– verschmelzen (für DNF mit $a+(a*b) =a$, für KNF mit $a*(a+b)=a$).
$t = x'y' + x'z + y' + x'y z'$
$= x'z + y' + x'y z'$

Jetzt hat man den Term als Summe von Monomen (bzw. als Produkt von Elementarsummen) dargestellt. Meist lässt man diese Form schon als DNF (KNF) gelten. Genaugenommen müsste man aber noch jedes Monom (jede Elementarsumme) durch die Verwendung der Gleichungen $a = a*1 = a*(b+b') = a*b+a*b'$ (bzw. $a = (a+b)*(a+b')$) aufblähen, so dass es alle Variablen enthält. Im Beispiel:

$t = x'z + y' + x'y z'$
$= x'y z + x'y'z + y' + x'y z'$
$= x'y z + x'y'z + x y' + x'y' + x'y z'$
$= x'y z + x'y'z + x y'z + x y'z' + x'y'z + x'y'z' + x'y z'$
$= x'y z + x'y'z + x y'z + x y'z' + x'y'z' + x'y z'$.

5.2.13 Aussagenlogik

George Boole hatte die Absicht, die *Gesetze des Denkens* zu formalisieren. Er ging dazu von *Elementaraussagen* aus, von denen er lediglich verlangte, dass sie entweder wahr ($T = true$) oder falsch ($F = false$) sind. Beispiele solcher Elementaraussagen sind z.B.

„2 + 2 = 5“

„Microsoft ist eine Biersorte.“

„Blei ist schwerer als Wasser.“

„Jede ungerade Zahl größer als 3 ist Summe zweier Primzahlen.“

Durch Verknüpfung mit den logischen Operationen $\wedge$ (und), $\vee$ (oder) $\neg$(nicht) erhält man neue, zusammengesetzte *Aussagen*. Formal:

- *Jede Elementaraussage ist eine Aussage.*
- *Sind A, A_1 und A_2 Aussagen, so auch $A_1 \vee A_2$, $A_1 \wedge A_2$ und $\neg A$.*

Der *Wahrheitswert* einer zusammengesetzten Aussage berechnet sich aus den Wahrheitswerten der Teilaussagen anhand der *Wahrheitstabellen*. Die Wahrheitstabellen für $\vee$, $\wedge$ und $\neg$ ergeben sich aus den entsprechenden Tabellen für + , * und ' , indem man überall 0 durch *F* und 1 durch *T* ersetzt (siehe auch S. 15):

$\vee$	F	T
F	F	T
T	T	T

$\wedge$	F	T
F	F	F
T	F	T

$\neg$	
F	T
T	F

Abb. 5.18: *Verknüpfungstabellen der logischen Operatoren Oder, Und und Negation*

Für die Äquivalenz von Aussagen gelten genau die Gleichungen der booleschen Algebra. Man muss lediglich + , * , ' , 0 , 1 durch $\vee, \wedge, \neg, F, T$ ersetzen. Beispielsweise hat man: $A \vee (A \wedge B) = A$, d.h. eine Aussage der Form $A \vee (A \wedge B)$ ist genau dann wahr, wenn A wahr ist. Zusätzliche logische Verknüpfungen kann man als Kombination aus den vorhandenen definieren, zum Beispiel:

$$A \Rightarrow B \quad \text{durch} \quad \neg A \vee B.$$

5.2.14 Mengenalgebra

Ausgehend von einer festen Grundmenge M betrachten wir $\boldsymbol{P}(M)$, die Menge aller Teilmengen von M. Auf $\boldsymbol{P}(M)$ untersuchen wir die Operationen $\cup$, $\cap$, $^-$ (Vereinigung, Schnitt und Komplement). Hier gelten die Gleichungen der booleschen Algebra, wenn man + , * , ' , 0 , 1 durch $\cup$, $\cap$, $^-$,$\varnothing$ und M ersetzt. Beispielsweise gilt für beliebige $U, V \in \boldsymbol{P}(M)$ die deMorgansche Regel

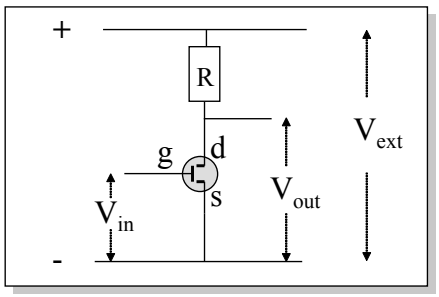
$$\overline{U \cap V} = \overline{U} \cup \overline{V}.$$

5.3 Digitale Logik

In der *digitalen Logik* geht es darum, Schaltfunktionen technisch zu realisieren. Als Elementarschalter werden in der Praxis *Transistoren* eingesetzt. Ein Transistor hat einen Eingang (*Source*), einen Ausgang (*Drain*) und einen Steuerungseingang (*Gate*). Legt man eine Spannung zwischen Source und Drain, so fließt nur dann Strom, falls auch eine Steuerspannung zwischen Source und Gate besteht.

In einem Stromkreis, bestehend aus einem Transistor und einem Widerstand *R* zwischen den Polen einer Spannungsquelle, kann man den Transistor als Schalter auffassen, der von einer Spannung zwischen *g* und *s* eingeschaltet wird.

Ist V_{ext} die externe elektrische Spannung (zwischen + und - in der obigen Abbildung), so kann man V_{out}, die Spannung zwischen *d* und *s,* in Abhängigkeit von V_{in}, der Spannung zwischen *g* und *s,* tabellieren. Man erhält, wenn man, wie üblich, die Spannung V_{ext} (die meist ca. 3 bis 5 V beträgt) mit 1 bezeichnet.



V_{in}	V_{out}
0	V_{ext}
V_{ext}	0

bzw.

V_{in}	V_{out}
0	1
1	0

Abb. 5.19: *Transistor als Schalter*

Betrachtet man stattdessen als Ausgangsspannung die am Widerstand *R* abfallende Spannung V_R, so hat man gerade das komplementäre Verhalten, denn $V_R = V_{ext} - V_{out}$.

Die folgenden beiden Schaltungen realisieren die Schaltfunktionen NAND und NOR.

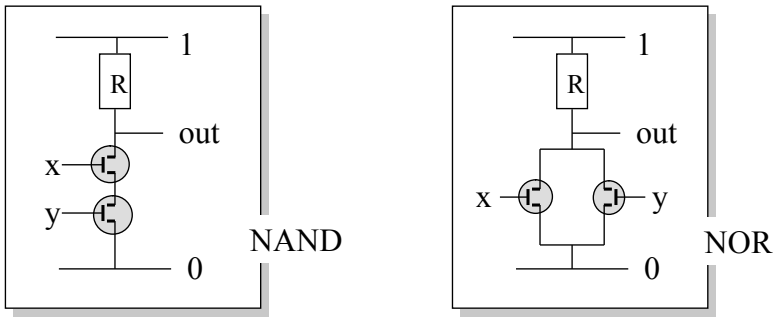


Abb. 5.20: *NAND und NOR*

Wir geben die Spannungswerte hier bezüglich des negativen Pols der Spannungsquelle an und erhalten die Tabellen: