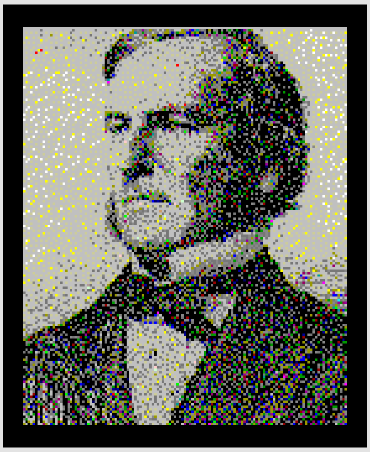
T=A∨B Boolesche Algebra

Georg Boole (1815-1846)



Born in the English industrial town of Lincoln, Boole was lucky enough to have a father who passed along his own love of math. Young George took to learning like a politician to a pay-rise and, by the age of eight, had outgrown his father's self-taught limits.

At the ripe old age of 24, George Boole published his first paper (Researches on the Theory of Analytical Transformations) in the Cambridge Mathematical Journal. Over the next ten years, his star rose as a steady stream of original articles began to push the limits of 'modern' mathematics.

Without a school to run, Boole began to delve deeper into his own work, concentrating on refining his Mathematical Analysis, and determined to find a way to encode logical arguments into an indicative language that could be manipulated and solved mathematically.

He came up with a type of linguistic algebra, the three most basic operations of which were (and still are) AND, OR and NOT. It was these three functions which formed the basis of his premise, and were the only operations necessary to perform comparisons or basic mathematical functions.

With George Boole's Mathematical Analysis and Investigation, Boolean algebra, sometimes known as Boolean logic, came into being.

Grundsätzliches

Bei den Formeln für die digitalen Grundfunktionen sind wir bereits der 'Booleschen Logik' begegnet. Da sich boolsche Ausdrücke von der herkömmlichen Mathematik unterscheiden, werden auch spezielle Operationszeichen verwendet.

Die Operationszeichen für INVERTER, AND sowie ODER werden mit unterschiedlichen Symbolen ausgedrückt. In der Programmiersprache C wird z.B. AND mit '&' und OR mit '|' dargestellt. Wir verwenden meistens die Symbole der Mengenlehre wie sie in den untenstehenden Beispielen dargestellt sind. Häufig wird auch das Zeichen '•' (mal) für die AND-Verknüpfung und das Zeichen '+' (plus) für die OR-Verknüpfung eingesetzt.

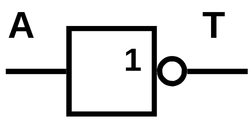
Oder - Operatoren:

T = A ∨ B	**T = A | B**	**T = A + B**

Und - Operatoren:

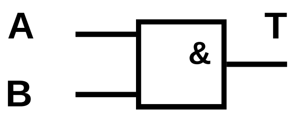
T = A ∧ B	**T = A & B**	**T = A * B**

INVERTER (NOT)



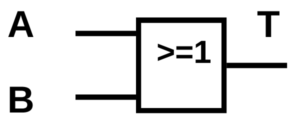
$T = \overline{A}$

UND (AND)



T = A ∧ B

ODER (OR)



T = A ∨ B

Wahrheitstabelle

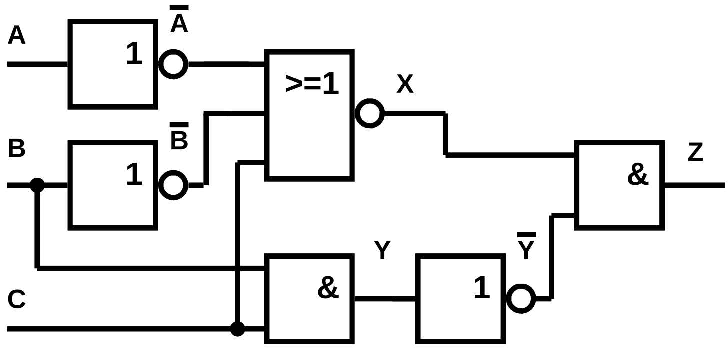
Wie für alle digitalen Grundschaltungen ist es auch möglich, für Schaltungen welche aus mehreren Gattern aufgebaut sind eine Wahrheitstabelle anzugeben.

Dazu werden zuerst alle möglichen Eingangskombinationen aufgelistet. Für jeden neuen Schaltungsteil wird in der Tabelle eine neue Spatle eingefügt, bis die gesamte Schaltung in Form einer Tabelle dargestellt werden kann.

Beispiel:

Ergänzen Sie für die nebenstehende Schaltung die Wahrheitstabelle.

Digitale Schaltung:



Wahrheitstabelle (A’ entspricht $\overline{A}$):

	C	B	A	A’	B’	X	Y	Y’	Z
0									
1									
2									
3									
4									
5									
6									
7									

Digitaltechnik und Computer: Bezeichnungen und Ausdrücke

Bit:
Binärziffer (Abkürzung für binary digit), kann nur 0 oder 1 sein. Kleinste Informationseinheit.

Byte:
Zusammenfassung von 8 Bits zu einer logischen Einheit; dies ermöglicht die Darstellung von 256 verschiedenen Zeichen (wie Ziffern, Buchstaben, Sonderzeichen). Die verbreitetsten Codes sind ASCII und EBCDIC. Kleinste adressierbare Speicherstelle bei einer Byte-Maschine (im Gegensatz zur Wortmaschine).

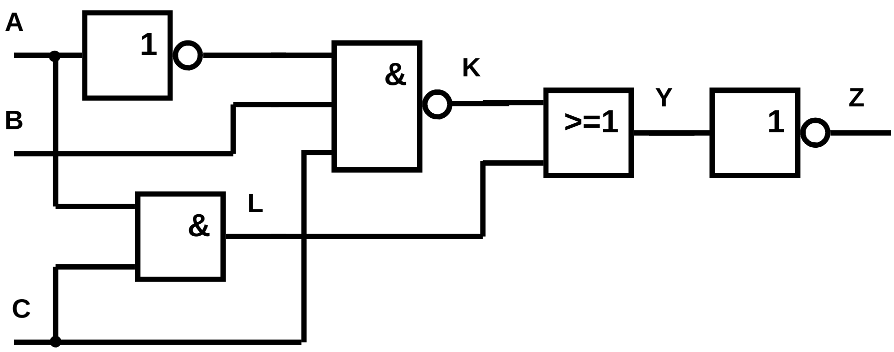
Bus:
Ein Leitungsbündel, an das mehrere Informations-Sender und -Empfänger angeschlossen sein können. Man unterscheidet z.B. Datenbus, Adressbus, Steuerbus.

Debugging:
Analysieren und Beseitigen von Fehlern. Am 9. September 1945 um 15.45 Uhr fand Grace Murry Hopper den allerersten "Bug" in der Geschichte der Informatik. Ein Käfer hatte ein Relais des Mark II Computers am Massachusetts Institute for Technology (MIT) in Harvard lahmgelegt. Sie klebte den Käfer in das Computer-Logbuch. Von nun an hiess es, immer wenn der Computer stillstand, was häufig vorkam, man sei am "debuggen”.

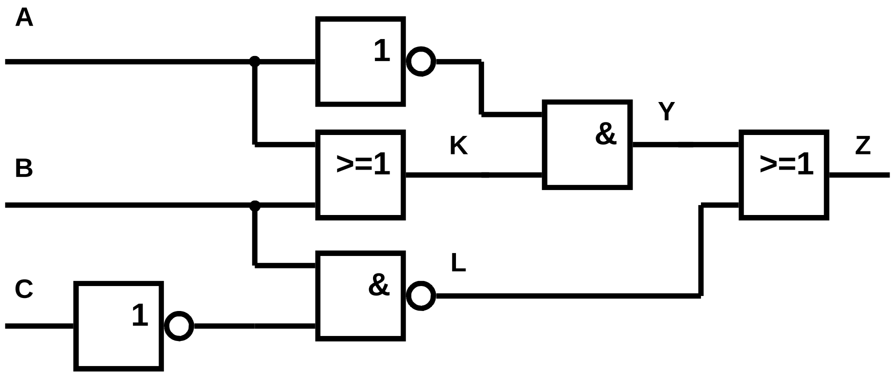
Funktionsgleichung

Auf der vorhergehenden Seite wurde dargestellt, wie eine digitale Schaltung durch eine Wahrheitstabelle ausgedrückt werden kann. Die einzelnen Schritte beim Erstellen der Wahrheitstabelle führen auch zu einer Gleichung für den Ausgang der Schaltung, in welcher nur die Eingangsgrössen und deren Negationen vorkommen. Da eine solche Gleichung die Funktion der ganzen Schaltung beschreibt, nennt man sie auch **Funktionsgleichung.**

Beispiele: Entwickeln Sie schrittweise die Funktionsgleichungen für die folgenden beiden Schaltungen

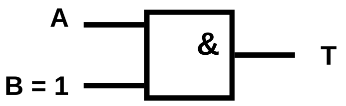


K =			
L =			
Y =		=	
Z =		=	



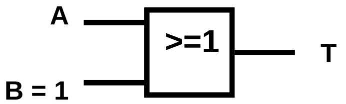
K =			
L =			
Y =		=	
Z =		=	

Grundgesetze der boolschen Logik



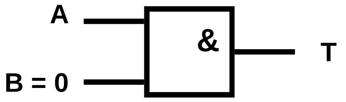
B	A	T
0	0	
0	1	
1	0	
1	1	

AND mit der Konstanten 1



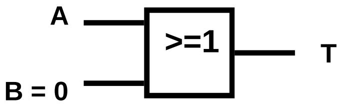
B	A	T
0	0	
0	1	
1	0	
1	1	

ODER mit der Konstanten 1



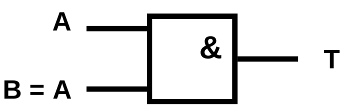
B	A	T
0	0	
0	1	
1	0	
1	1	

AND mit der Konstanten 0



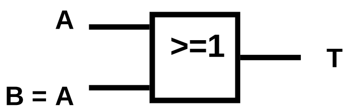
B	A	T
0	0	
0	1	
1	0	
1	1	

ODER mit der Konstanten 0



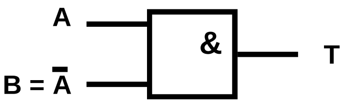
B	A	T
0	0	
0	1	
1	0	
1	1	

AND mit sich selbst



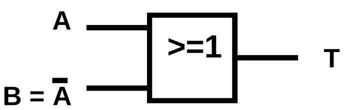
B	A	T
0	0	
0	1	
1	0	
1	1	

ODER mit sich selbst



B	A	T
0	0	
0	1	
1	0	
1	1	

AND mit dem Inversen von sich selbst

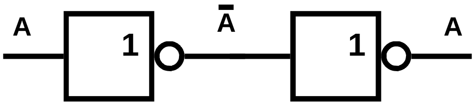


B	A	T
0	0	
0	1	
1	0	
1	1	

ODER mit dem Inversen von sich selbst

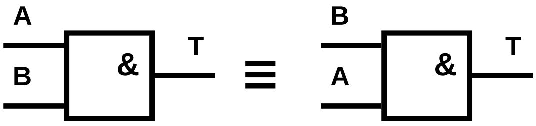
Rückbildungsgesetz (Involution)

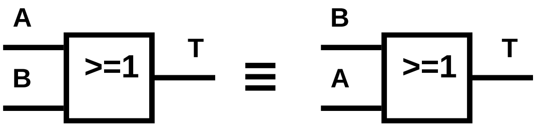
Das Rückbildungsgesetz sagt aus, dass sich eine gerade Anzahl Invertierungen aufheben.



Kommutativgesetz (Vertauschung)

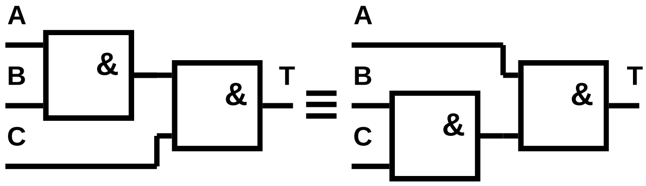
Das Kommutativgesetz sagt aus, dass es keine Rolle spielt, in welcher Reihenfolge Variablen der AND oder OR Verknüpfung angegeben werden.

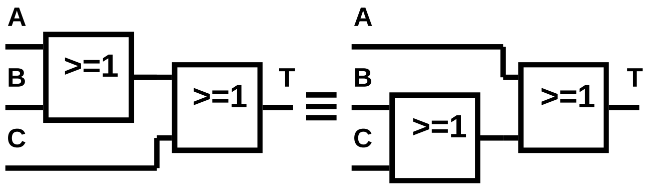




Assoziativgesetz (Verbindung)

Die Reihenfolge der Zuordnung der Variablenbei der AND- sowie der OR-Verknüpfung ist beliebig.





Integrierter Schaltkreis IC

engl. integrated cuircuit, Abk. IC, Halbleiterbauelement, bei dem mehrere elektronische Elemente (Transistoren, Kondensatoren etc.) in einem Stück, als sog. Chip, zusammengefasst sind. Integrierte Schaltungen werden danach klassifiziert, wie viele Funktionen sie auf kleinster Fläche beherbergen. Man unterscheidet:

***SSI** (Small Scale Integration): wenige integrierte Bauelemente*
***MSI** (Middle Scale Integration): einige hundert Bauelemente*
***LSI** (Large Scale Integration): mehr als tausend Bauelemente*
***VLSI** (Very Large Scale Integration): mehr als zehntausend Bauelemente.*

Priorität der Operatoren

In der normalen Mathematik hat die Multiplikation höhere Priorität als die Addition. (Punktoperation vor Strichoperation). Im folgenden Beispiel muss die Multiplikation vor der Addition ausgeführt werden!

$$5 + 3 * 8 = 5 + (3 * 8) = 29$$

In der Boolschen Algebra hat der AND-Operator höhere Priorität als der OR-Operator. Es gilt:

$$A \vee B \wedge E = A \vee (B \wedge E)$$

Wegen den Ähnlichkeiten zwischen den arithmetischen und logischen Operatoren wird der AND-Operator ($\wedge$) auch als logische Multiplikation und der OR-Operator als logische Addition bezeichnet. In vielen Fällen werden sogar die entsprechenden arithmetischen Operationszeichen eingesetzt:

$$\wedge \Rightarrow * \qquad \vee \Rightarrow +$$

Um Fehlinterpretationen zu vermeiden, empfiehlt es sich, häufig Klammern zu setzen.

1. Distributivgesetz (Verteilung)

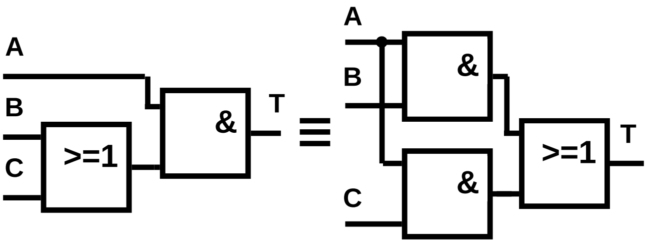
Von der normalen Algebra her, kennen wir die Regel für die Multiplikation einer Summe mit einem Faktor:

$$X * (Y + Z) = (X * Y) + (X * Z)$$

Diese Regel gilt sinngemäss auch für die boolsche Algebra:

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

1. Distributivgesetz als Schaltung:



2. Distributivgesetz (Verteilung)

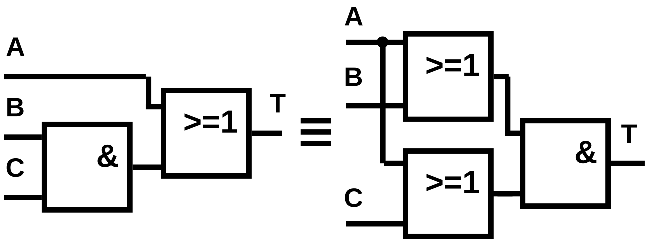
Das 2. Distributivgesetz der Boolschen Algebra ist in der normalen Algebra nicht möglich:

$$X + (Y * Z) <> (X + Y) * (X + Z)$$

Im Gegensatz zur normalen Algebra gilt jedoch bei der boolsche Algebra:

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

2. Distributivgesetz als Schaltung:



Aufgabe1:
Zeigen Sie, dass die beiden Distributivgesetze wirklich stimmen indem Sie für die beiden Schaltungen der 1. Distributivgesetzes und für die beiden Schaltungen des zweiten Distributivgsetzes die Wahrheitstabellen erstellen. Vergleichen Sie die Resultate!

Aufgabe 2:
Zeichnen Sie die Schaltungen des 1. und des 2. Distributivgesetzes mit elektromechanischen Schaltern.

Charles Babbage (1792 - 1871)

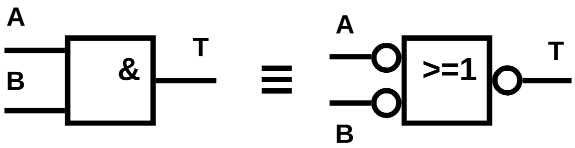
Charles Babbage was born in 1792 in Teignmouth, Devonshire, UK . Some people know him as the "Father of Computing" as a result of his contributions to the basic design of the computer. A major contribution was his Analytic Machine. Before he built this he produced the Difference Engine which operated on 6-digit numbers, and was designed to solve 2nd order difference equations.
However, only a few people knew who he was when he died in 1871 in London. But nowadays every person studying the history of the computer knows many things about him such as punch cards, chains and subassemblies. Ultimately the logical structure of the modern computer come from him. The analytical engine devised by Charles Babbage included 5 features crucial to future computers:
An input device, a storage facility to hold numbers for processing, a processor or number calculator, a control unit to direct tasks to be performed and an output device.
Unfortunately this engine never worked, because the technology of manufacturing exact technical parts was not developed far enough. This inaccuracies kept the machine from working.
But Charles Babbage did not only work in the field of computers. He also was responsible for other important inventions like dynamometer, standard railroad gauge, occulting lights for lighthouses, Greenwich time signals and many more.

Das Theorem von De Morgan (Umkehrgesetz)

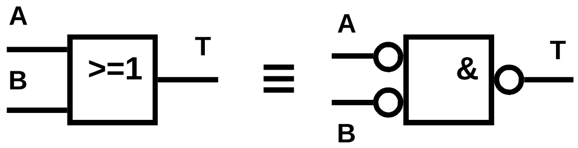
Das Theorem von De Morgan gibt die Bedingungen an, unter welchen eine AND- Funktion in eine OR-Funktion umgewandelt werden kann und umgekehrt.

In den IEC Normen für das Schemazeichnen wird neben der herkömmlichen Darstellung der AND-OR-Tore auch die nach De Morgan umgewandelte Form zugelassen. Dies bietet insbesondere bei inversen Signalen (z.B. CE' aktiv '0') bessere Lesbarkeit des Schemas.

Ein AND-Tor kann man durch ein OR-Tor ersetzen, wenn gleichzeitig alle Ein- und Ausgänge invertiert werden.



Ein OR-Tor kann man durch ein AND-Tor ersetzen, wenn gleichzeitig alle Ein- und Ausgänge invertiert werden.



Das Theorem von De Morgan gilt auch für Verknüpfungen von mehr als zwei Variablen (für Tore mit mehreren Eingängen):

Augustus De Morgan (1806 - 1871)

In 1828 De Morgan became the first professor of mathematics at University College, London. He gave his inaugural lecture on the study of mathematics.
In 1849 he published Trigonometry and double algebra in which he gave a geometric interpretation of complex numbers.
He recognised the purely symbolic nature of algebra and he was aware of the existence of algebras other than ordinary algebra. He introduced De Morgan's laws and his greatest contribution is as a reformer of mathematical logic.
De Morgan corresponded with Charles Babbage and gave private tuition to Lady Lovelace who, it is claimed, wrote the first computer program for Babbage.

Aufgabe zum Gesetz von De Morgan:

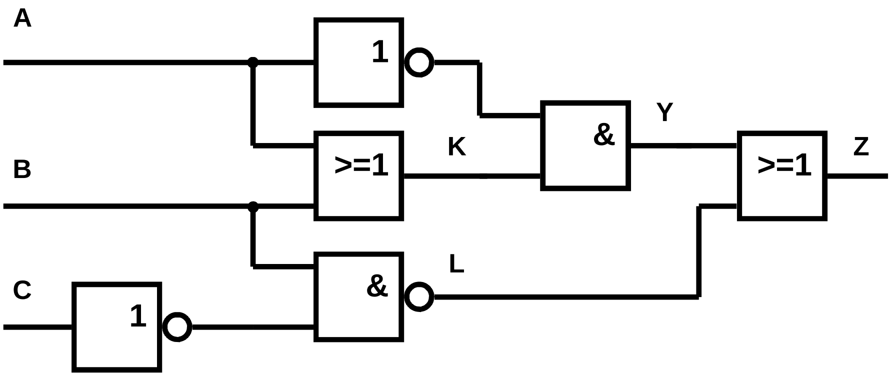
Zeichnen Sie die Tore der Grundfunktionen AND, OR, NAND und NOR. Geben Sie zu allen Schaltungen die Ersatzschaltung nach De Morgan mit der betreffenden Funktionsgleichung und der Wahrheitstabelle an.

<table>
<tr><th>Grundschaltung</th><th>De Morgan Ersatzschaltung</th><th>Wahrheitstabelle</th></tr>
<tr><td>AND:</td><td></td><td rowspan="2"><table><tr><th>B</th><th>A</th><th>T</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table></td></tr>
<tr><td>Funktionsgleichung:</td><td>Funktionsgleichung:</td></tr>
<tr><td>OR:</td><td></td><td rowspan="2"><table><tr><th>B</th><th>A</th><th>T</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table></td></tr>
<tr><td>Funktionsgleichung:</td><td>Funktionsgleichung:</td></tr>
<tr><td>NAND:</td><td></td><td rowspan="2"><table><tr><th>B</th><th>A</th><th>T</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table></td></tr>
<tr><td>Funktionsgleichung:</td><td>Funktionsgleichung:</td></tr>
<tr><td>NOR:</td><td></td><td rowspan="2"><table><tr><th>B</th><th>A</th><th>T</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table></td></tr>
<tr><td>Funktionsgleichung:</td><td>Funktionsgleichung:</td></tr>
</table>

Aufgabe zum Gesetz von De Morgan:

a Entwerfen Sie eine Schaltung welche ausschliesslich mit 2-Eingang NAND-Toren aufgebaut ist und die gleiche Funktion erfüllt, wie die untenstehende Schaltung.

b Entwerfen Sie eine Schaltung welche ausschliesslich mit 2-Eingang NOR-Toren aufgebaut ist und die gleiche Funktion erfüllt, wie die untenstehende Schaltung.



Minterme und Maxterme

Für jede Eingangskombination einer logischen Schaltung kann ein Minterm und ein Maxterm angegeben werden:

- Bei den **Mintermen (auch Vollkonjunktion)** werden die Eingangsvariablen respektive die inversen Eingangsvariablen jeder Zeile der Wahrheitstabelle derart UND verknüpft, dass das Resultat ‘1’ ergibt.

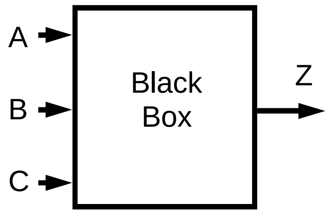
- Bei den **Maxtermen (auch Volldisjunktion)** werden die Eingangsvariablen respektive die inversen Eingangsvariablen jeder Zeile der Wahrheitstabelle derart ODER verknüpft, dass das Resultat ‘0’ ergibt.

Min- und Maxterme werden bei der Herleitung von Funktionsgleichungen aus Wahrheitstabellen angewendet. In der folgenden Tabelle sind die Min- und Maxterme für eine Wahrheitstabelle mit 3 Eingangsvariablen dargestellt. (A’ bedeutet A invers).

C	**B**	**A**	**Minterm**	**Maxterm**
0	0	0	$A' \wedge B' \wedge C' = 1$	$A \vee B \vee C = 0$
0	0	1	$A \wedge B' \wedge C' = 1$	$A' \vee B \vee C = 0$
0	1	0	$A' \wedge B \wedge C' = 1$	$A \vee B' \vee C = 0$
0	1	1	$A \wedge B \wedge C' = 1$	$A \vee B' \vee C' = 0$
1	0	0	$A' \wedge B' \wedge C = 1$	$A \vee B \vee C' = 0$
1	0	1	$A \wedge B' \wedge C = 1$	$A' \vee B \vee C' = 0$
1	1	0	$A' \wedge B \wedge C = 1$	$A \vee B' \vee C' = 0$
1	1	1	$A \wedge B \wedge C = 1$	$A' \vee B' \vee C' = 0$

Normalformen

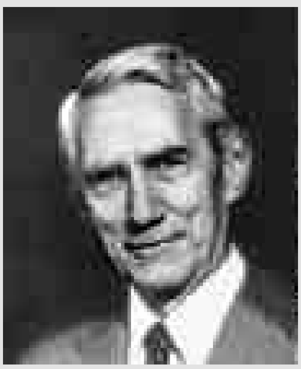
Bei der Entwicklung logischer Schaltungen geht man meistens von Wahrheitstabellen aus. Eine Schaltung wird vorerst als ‘Black Box’ betrachtet. In einem wieteren Schritt wird dann, ausgehend von der Wahrheitstabelle, eine Funktionsgleichung ermittelt. Dieser Schritt kann anhand der Normalformen durchgeführt werden. Man unterscheidet zwischen der disjunktiven (oder) Normalform und der konjunktiven (und) Normalform. Beide Formen ergeben gleichwertige, jedoch unterschiedliche Funktionsgleichungen.



Black-Box mit der entsprechenden Wahrheitstabelle

C	B	A	T
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Claude E. Shannon ***(Born: 30 April 1916, Gaylord, Michigan, USA)***



Claude Shannon *was a graduate of Michigan and went to MIT where he wrote a thesis on the use of Boole's algebra to analyse and optimise relay switching circuits. He joined Bell Telephones in 1941 as a research mathematician and remained there until 1972.*
He published A Mathematical Theory of Communication in in the Bell System Technical Journal (1948). His work founded the subject of information theory and he proposed a linear schematic model of a communications system. He gave a method of analysing a sequence of error terms in a signal to find their inherent variety, matching them to the designed variety of the control system.
In 1952 he devised an experiment illustrating the capabilities of telephone relays. Shannon was awarded the National Medal of Science in 1966.

Disjunktive Normalform (oder)

(Sum-of-Products)

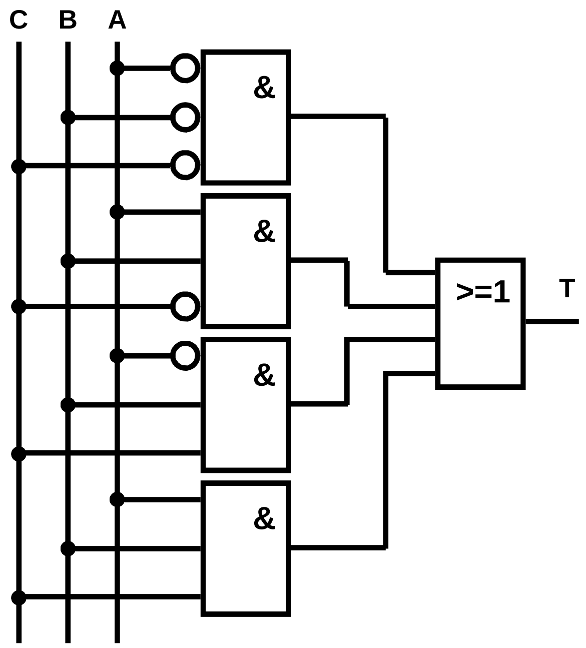
Mit der disjunktiven Normalform kann für eine Wahrheitstabelle eine Funktionsgleichung angegeben werden indem für alle Zeilen der Wahrheitstabelle welche den Ausgangszustand ‘1’ aufweisen die Minterme ermittelt werden. Alle so erhaltenen Minterme miteinander ODER verknüpft, ergeben die Funktionsgleichung (disjunktive Normalform oder Sum-of-Products).

UND (Minterm)

C	B	A	T	Minterme
0	**0**	**0**	**1**	$\mathbf{A' \wedge B' \wedge C'}$
0	0	1	0	
0	1	0	0	
0	**1**	**1**	**1**	$\mathbf{A \wedge B \wedge C'}$
1	0	0	0	
1	0	1	0	
1	**1**	**0**	**1**	$\mathbf{A' \wedge B \wedge C}$
1	**1**	**1**	**1**	$\mathbf{A \wedge B \wedge C}$

ODER

$$T = (A' \wedge B' \wedge C') \vee (A \wedge B \wedge C') \vee (A' \wedge B \wedge C) \vee (A \wedge B \wedge C)$$



Konjunktive Normalform (und)

(Product-of-Sums)

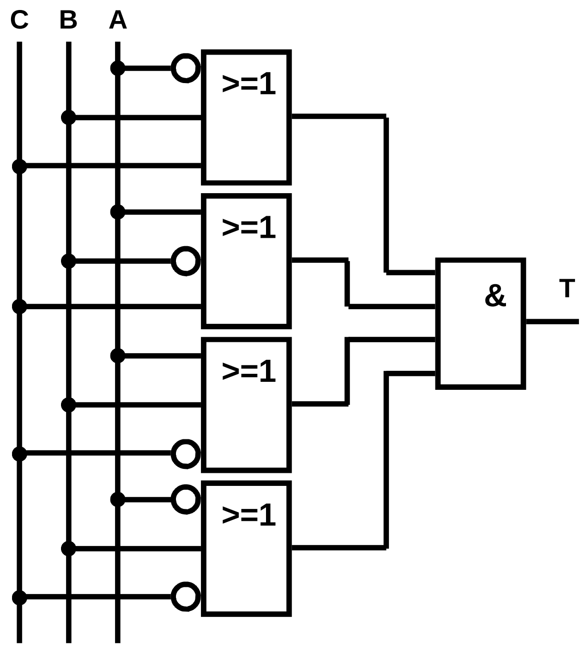
Mit der konjunktiven Normalform kann für eine Wahrheitstabelle eine Funktionsgleichung angegeben werden indem für alle Zeilen der Wahrheitstabelle welche den Ausgangszustand ‘0’ aufweisen die Maxterme ermittelt werden. Alle so erhaltenen Maxterme miteinander UND verknüpft, ergeben die Funktionsgleichung (konjunktive Normalform oder Product-of-Sums).

ODER (Maxterm)

C	B	A	T	Maxterme
0	0	0	1	
0	**0**	**1**	**0**	$\mathbf{A' \vee B \vee C}$
0	**1**	**0**	**0**	$\mathbf{A \vee B' \vee C}$
0	1	1	1	
1	**0**	**0**	**0**	$\mathbf{A \vee B \vee C'}$
1	**0**	**1**	**0**	$\mathbf{A' \vee B \vee C'}$
1	1	0	1	
1	1	1	1	

UND

$$T = (A' \vee B \vee C) \wedge (A \vee B' \vee C) \wedge (A \vee B \vee C') \wedge (A' \vee B \vee C')$$



Trotz den unteschiedlichen Funktionsgleichungen und Schaltungen sind beide Normalformen in ihrer Funktionsweise gleichwertig. Die konjunktive Normalform wird dann angewendet, wenn in der Wahrheitstabelle die ‘1’ Zustände am Ausgang überwiegen. Überwiegen am Ausgang die ‘0’ Zustände, wird mit Vorteil die konjunktive Normalform angewendet.

Aufgabe: Welche Möglichkeiten haben Sie, um zu beweisen, dass die beiden oben dargestellten Schaltungen identisch sind? Führen sie den Beweis durch.

Optimierung der Normalformen mit dem KV-Diagramm

Die mit der Wahrheitstabellen erhaltenen Normalformen geben häufig nicht die einfachste Schaltungsfunktion an. Gesucht ist jedoch meistens die kostengünstigste Schaltung, welche mit möglichst wenig Gattern aufgebaut werden kann. Um dies zu erreichen, müssen die Normalformen noch systematisch optimiert werden. Dies kann mit verschiedenen Verfahren erreicht werden. Hier soll das grafische Verfahren nach Karnaugh und Veitch (kurz KV-Diagramm) angewendet werden. Das KV-Diagramm eignet sich sehr gut für die Vereinfachung von logischen Schaltungen mit bis zu fünf Eingangs- und einer Ausgangsvariablen. Dabei wird die Wahrheitstabelle in ein Rechteck mit 2^n Feldern übergeführt (n = Anzahl der Eingangsvariablen).

KV-DIAGRAMM FÜR 2 VARIABLEN

Mit zwei Variablen erhalten wir ein quadratisches KV-Diagramm. Jedes der KV-Diagramm-Felder entspricht einer Zeile (einem Minterm) in der Wahrheitstabelle: Die Resultate der Minterme werden nun im KV-Diagramm eingetragen.

B	A	Minterm
0	0	**A’∧B’**
0	1	**A ∧B’**
1	0	**A’∧B**
1	1	**A ∧B**

	A	**A’**
B	**A ∧B**	**A’∧B**
B’	**A ∧B’**	**A’∧B’**

Regeln für die Vereinfachung (2 Variablen):

- Benachbarte Felder die eine 1 enthalten können zu 2er oder 4er Rechtecken zusammengefasst werden (gilt nur für Felder mit einer gemeinsamen Seite).
- Die Rechtecke sollen so gross wie möglich sein.
- Die Anzahl der Rechtecke soll so klein wie möglich sein, jedoch alle 1 enthalten.
- Für jedes Rechteck kann anhand der Koordinatenbezeichnungen eine Formel angegeben werden. Dabei werden Variablen welche **invers** und **nicht invers** auftreten, **weggelassen**.
- Die reduzierten Formel ergibt sich als ODER-Verknüpfung der einzelnen Rechteckformeln und der Einzelzellen die eine ‘1’ enthalten.

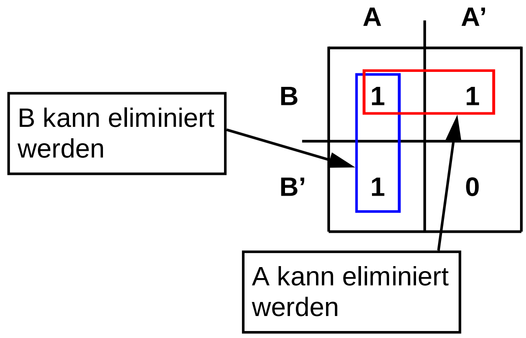
Beispiel: KV-Diagramm mit 2 Variablen

Erster Schritt: Wahrheitstabelle in das KV-Diagramm überführen:

B	A	T	
0	0	**0**	
0	1	**1**	**A ∧B’**
1	0	**1**	**A’∧B**
1	1	**1**	**A ∧B**

	A	**A’**
B	**1**	**1**
B’	**1**	**0**

Zweiter Schritt: Rechtecke bilden

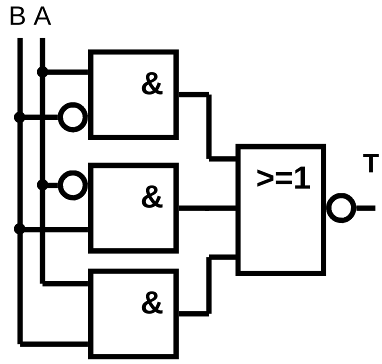


Dritter Schritt: Vereinfachte Funktionsgleichung erstellen:

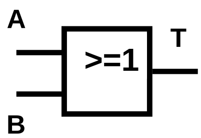
Im senkrechten Rechteck kommen die Variablen B und B’ gemeinsam vor und entfallen daher.
(A∧B) ∨ (A∧B’) = A ∧ (B∨B’) = A ∧ (1) = A

Im waagrechten Rechteck treten A und A’ gemeinsam auf und entfallen daher.
(A∧B) ∨ (A’∧B) = B ∧ (A∨A’) = B∧(1) = B

Die vereinfachte Formel lautet: **T = A ∨ B**



Schaltung nach Normalform
T = (A ∧B’)∨(A’∧B)∨(A ∧B)



Schaltung mit KV-Diagramm vereinfacht
T = A ∨ B

KV-DIAGRAMM FÜR 3 VARIABLEN

Mit drei Eingangsvariablen ergeben sich 8 Kombinationsmöglichkeiten, somit 8 Zeilen in der Wahrheitstabelle und dem entsprechend 8 Felder im KV-Diagramm.

KV-Diagramm mit 8 Feldern und den eingetragenen Mintermen:

	A		A’	
B	$A \wedge B \wedge C'$	$A \wedge B \wedge C$	$A' \wedge B \wedge C$	$A' \wedge B \wedge C'$
B’	$A \wedge B' \wedge C'$	$A \wedge B' \wedge C$	$A' \wedge B' \wedge C$	$A' \wedge B' \wedge C'$
	C’	C		C’

Regeln für die Vereinfachung (3 Variablen):

Neben den bereits angegebenen Regeln für 2 Variablen gelten zusätzlich folgende Regeln:

- Benachbarte Felder die eine 1 enthalten können zu 2er, 4er oder 8er Rechtecken zusammengefasst werden.
- Auch Felder an gegenüberliegenden Zeilen und Spalten gelten als benachbart.

Beispiel: KV-Diagramm mit 3 Variablen

- Geben Sie für die folgende Wahrheitstabelle die disjunktive Normalform an.
- Vereinfachen Sie die Normalform mit Hilfe der Schaltalgebra
- Tragen Sie die Minterme im KV-Diagramm ein und geben Sie die vereinfachte Schaltungsfunktion an
- Zeichnen Sie die Schaltung

C	B	A	T
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Normalform:

Vereinfachung mit Hilfe der Schaltalgebra:

Vereinfachung mit dem KV-Diagramm:

	A		A’	
B				
B’				
	C’	C		C’

Funktionsgleichung gemäss KV-Diagramm:

Schaltung:

KV-DIAGRAMM FÜR 4 VARIABLEN

Das KV-Diagramm für vier Eingangsvariablen besteht aus 16 Feldern.

	A		A'		
B	$A \wedge B \wedge C' \wedge D'$	$A \wedge B \wedge C \wedge D'$	$A' \wedge B \wedge C \wedge D'$	$A' \wedge B \wedge C' \wedge D'$	D'
B	$A \wedge B \wedge C' \wedge D$	$A \wedge B \wedge C \wedge D$	$A' \wedge B \wedge C \wedge D$	$A' \wedge B \wedge C' \wedge D$	D
B'	$A \wedge B' \wedge C' \wedge D$	$A \wedge B' \wedge C \wedge D$	$A' \wedge B' \wedge C \wedge D$	$A' \wedge B' \wedge C' \wedge D$	D
B'	$A \wedge B' \wedge C' \wedge D'$	$A \wedge B' \wedge C \wedge D'$	$A' \wedge B' \wedge C \wedge D'$	$A' \wedge B' \wedge C' \wedge$	D'
	C'	C	C	C'	

Regeln für die Vereinfachung (4 Variablen):

Neben den bereits angegebenen Regeln für 2 und 3 Variablen gilt zusätzlich:

- Die ‘1’en in den vier Ecken können zu einem Viererblock zusammengefasst werden.

Geben Sie für die folgenden KV-Diagramme die Funktionsgleichung an:

	A		A'		
B	1			1	D'
B					D
B'					D
B'	1	1	1	1	D'
	C'	C	C	C'	

	A		A'		
B	1		1	1	D'
B	1			1	D
B'					D
B'			1	1	D'
	C'	C	C	C'	

	A		A'		
B			1	1	D'
B	1	1			D
B'	1	1			D
B'			1	1	D'
	C'	C	C	C'	

	A		A'		
B	1	1			D'
B			1		D
B'			1	1	D
B'	1	1			D'
	C'	C	C	C'	

	A		A'		
B	1	1	1	1	D'
B					D
B'					D
B'	1	1	1	1	D'
	C'	C	C	C'	

	A		A'		
B	1			1	D'
B	1	1	1	1	D
B'	1			1	D
B'	1			1	D'
	C'	C	C	C'	

	A		A'		
B	1				D'
B	1				D
B'	1	1	1	1	D
B'	1				D'
	C'	C	C	C'	

	A		A'		
B	1			1	D'
B		1	1		D
B'		1	1		D
B'	1			1	D'
	C'	C	C	C'	

	A		A'		
B	1	1	1	1	D'
B	1			1	D
B'	1			1	D
B'	1	1	1	1	D'
	C'	C	C	C'	

UNBESTIMMTE FUNKTIONSWERTE

Unbestimmte Funktionswerte entstehen, wenn gewisse Eingangskombinationen nicht eintreten können, oder keine Rolle spielen.

Im Zusammenhang mit dem BCD-Code treten solche unbesttimmten Funktionswerte oft auf. Wollen wir Daten im Dezimalsystem darstellen, müssen wir für eine Stelle (Einer, Zehner, usw) einen Zahlenumfang von 0..9 abdecken. Im binären System berauchen wir für diese Darstellung 4 Bit. Mit diesen 4 Bit haben wir aber ingesamt 16 Kombinationsmöglichkeiten. Die Kombinationen von 10..15 wenden wir jedoch nie an. Diese 6 Werte können daher oft als unbestimmte Funktionswerte deklariert werden.

Binary Coded Decimal Code (BCD)

The Binary Coded Decimal, or BCD code, is used as the direct code to communicate decimal numbers using a binary code. Recall that the decimal numbers are 0 to 9. BCD code is used to directly translate these decimal numbers using a 4-bit code.
Here are the decimal equivalent 4-bit BCD codes:

Dezimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Notice that each single decimal digit has an equivalent 4-bit BCD code.

In einer Wahrheitstabelle und im KV-Diagramm werden die **unbestimmten Funktionswerte als ‘X’** eingetragen.

‘X’ können im KV-Diagramm sowohl **als ‘0’ oder als ‘1’** betrachtet werden. Dadurch lassen sich optimalere Blöcke bilden.

Beispiel:

Durch ein optimales Einbeziehen der unbestimmten Funktionswerte ‘X’ ergeben sich einfachere Funktionsgleichungen:

Bestimmen Sie die Funktionsgleichung **ohne** die unbestimmten Funktionswerrte X zu berücksichtigen:

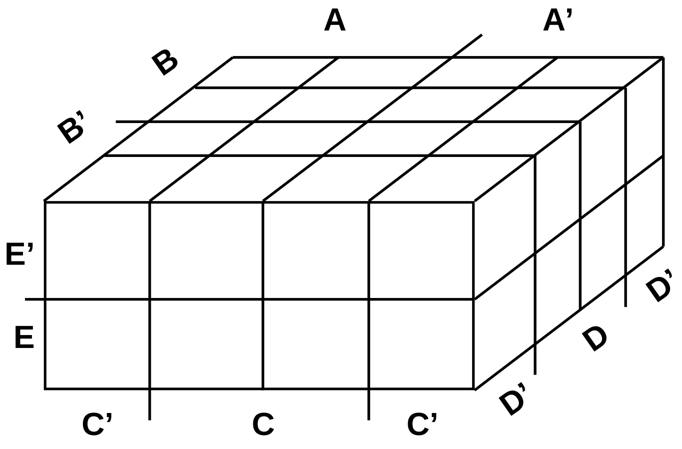
	A		A’		
B	0	1	0	X	D’
	0	X	0	0	D
B’	X	1	0	0	
	0	1	0	0	D’
	C’	C		C’	

Bestimmen Sie die Funktionsgleichung **unter** Berücksichtigung der unbestimmten Funktionswerte X:

	A		A’		
B	0	1	0	X	D’
	0	X	0	0	D
B’	X	1	0	0	
	0	1	0	0	D’
	C’	C		C’	

KV-DIAGRAMM FÜR 5 VARIABLEN

Ein KV-Diagramm für 5 Variablen muss 32 Plätze aufweisen. Zu diesem Zweck wird dem KV-Diagramm eine zusätzliche Etage angebaut. Dem unteren Stockwerk wird die Variable E zugeordnet, dem oberen Stockwerk die Variable E’.



Regeln für die Vereinfachung (5 Variablen):

Benachbarte Felder die eine 1 enthalten können zu 2er, 4er, 8er, 16er oder 32er Rechtecken (repektive Quader) zusammengefasst werden . Zusätzlich zu den bereits erwähnten Regeln gilt neu:

- Auch Felder die übereinander liegen, können zu Rechtecken oder Quadern

Zur Bearbeitung werden die beiden Stockwerke einzeln aufgezeichnet:

	A	A	A’	A’	
B			1		D’
B			1		D
B’					D
B’					D’
	C’	C	C	C’	E’

	A	A	A’	A’	
B			1		D’
B			1		D
B’			1		D
B’			1		D’
	C’	C	C	C’	E

Beispiel: Vereinfachen Sie das untenstehende KV-Diagramm für 5 Variablen

	A	A	A’	A’	
B	1			1	D’
B					D
B’	1	1	1	1	D
B’	1			1	D’
	C’	C	C	C’	E’

	A	A	A’	A’	
B	1			1	D’
B		1	1		D
B’					D
B’	1			1	D’
	C’	C	C	C’	E