Technische Universität Braunschweig

# Informatik für Ingenieure – VL 3

Prof. Dr. Andres Gomez

# 2. ENTWURF UND ANALYSE KOMBINATORISCHER SCHALTUNGEN

# Überblick



| | |
|---|---|
| Anwendungs-software | Programme |
| Betriebs-systeme | Gerätetreiber |
| **Architektur** | Befehle<br>Register |
| **Mikro-architektur** | Datenpfade<br>Steuerung |
| **Logik** | Addierer<br>Speicher |
| **Digital-schaltungen** | UND Gatter<br>Inverter |
| Analog-schaltungen | Verstärker<br>Filter |
| Bauteile | Transistoren<br>Dioden |
| Physik | Elektronen |

Ziele für Heute:

- Einleitung zu Logik und Digital Schaltungen
- Boolesche Algebra
  - Axiome, Sätze
  - Gleichungen
- Von Logik zu Gattern

Ein Teil des heutigen Vortrags basiert auf der Vorlesung von
Prof. H Michalik (TU Braunschweig)
Prof. S. Harris (TU Darmstadt) und
Prof. J Wawrzynek (UC Berkley)

# Definitionen



Ein digitales System realisiert in definierter Weise eine Zuordnung von binären Eingangswerten (oder digitalen Signalen) in binäre Ausgangswerte.

Digitale Systeme werden dabei üblicherweise mit drei Gruppen von Schaltungen und deren Kombinationen realisiert:

mit **Schaltnetzen**, d.h. kombinatorischen Schaltungen, die rein kombinatorisch zusammengesetzt sind und keinerlei Speicher enthalten.

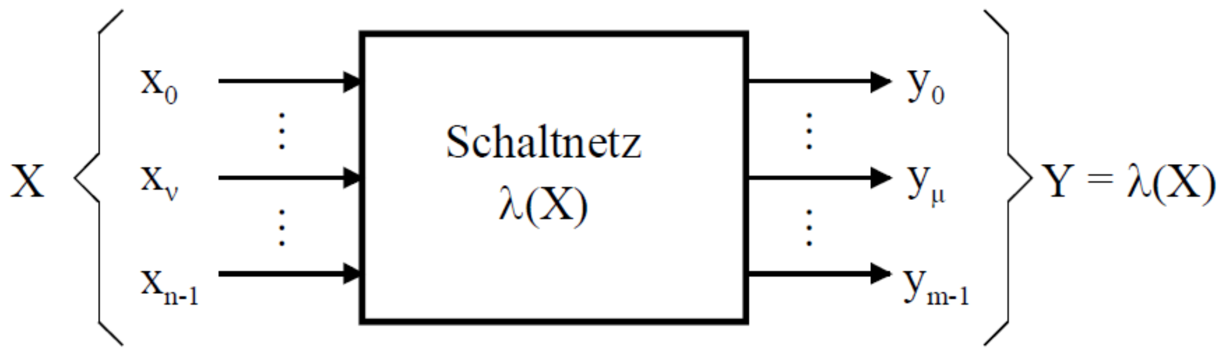
mit **Schaltwerken**, d.h. sequentiellen Schaltungen, die neben der Speicherung und Zeitverzögerung digitaler Signale auch die Impulserzeugung ermöglichen.

mit **Sonderschaltungen** z.B. zur Anpassung von Eingangs- und Ausgangssignalen, Pegelumsetzung und Leistungsverstärkung (Pegelwandler, Treiber, Sender-/ Empfängerbausteine).

# Definitionen



Eine kombinatorische Schaltung (auch Schaltnetz**)** ordnet
jeder Kombination von binären Eingangsvariablen ($x_0$, ... , $x_v$, ... ,$x_{n-1}$ )
eine Kombination von binären Ausgangsvariablen ($y_0$, ... ,$y_\mu$, ... ,$y_{m-1}$) zu.




Die Eingangsvariablen x sind als gegebene Signale, z.B. elektrische Spannungen von binären anzusehen.

Die Ausgangsvariablen y führen wiederum zu binären Elementen.

# Definitionen



Ein Eingangswort X wird gebildet aus einer Binärkombination der Komponenten $x_0$ ... $x_{n-1}$:

$$X_i = (x_{n-1}, x_{n-2}, \ldots, x_\nu, \ldots, x_0)$$

Der binäre Vektor $X_i$ ist ein bestimmtes Element der Menge aller Eingangskombinationen $X^k$, wobei der Exponent k die Mächtigkeit der Menge der Eingangskombinationen kennzeichnet.

Ein Ausgangswort oder binärer Vektor Y wird in gleicher Weise gebildet aus einer Binärkombination der Komponenten $y_0$ ... $y_{m-1}$:

$$Y_j = (y_{m-1}, y_{m-2}, \ldots, y_\mu, \ldots, y_0)$$

Im Folgenden wird der **Entwurf von kombinatorischen Schaltungen** bei gegebener und **zu realisierender logischer Funktion λ** betrachtet.

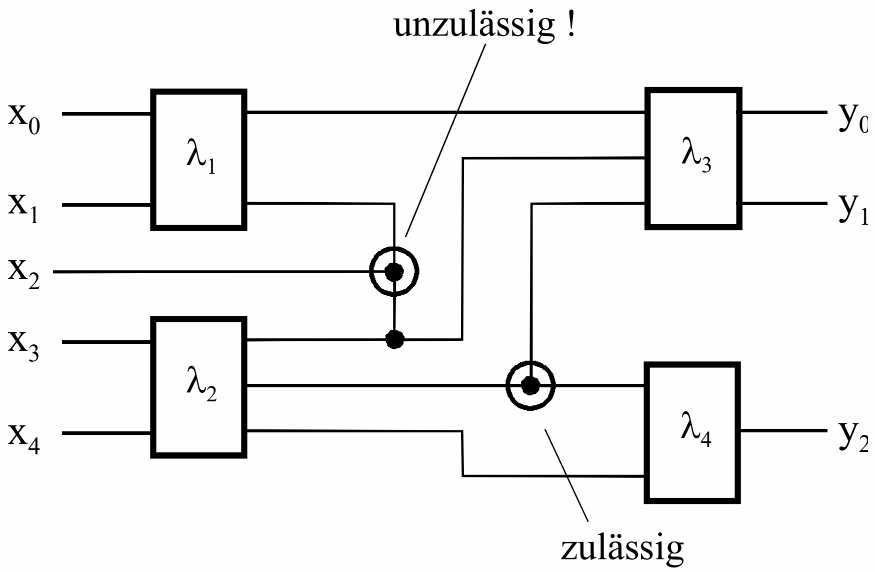
# Definitionen



Der logische Zustand einer Signalleitung muss immer eindeutig 0 oder 1 sein.

Zu beachten ist bei der Verkettung von kombinatorischen Modulen, dass keine Ausgänge von Teilnetzen direkt untereinander oder direkt mit Eingängen zusammen geschaltet werden.

Zulässig ist aber, ein Signal auf mehrere Eingänge im Netz zu verteilen.

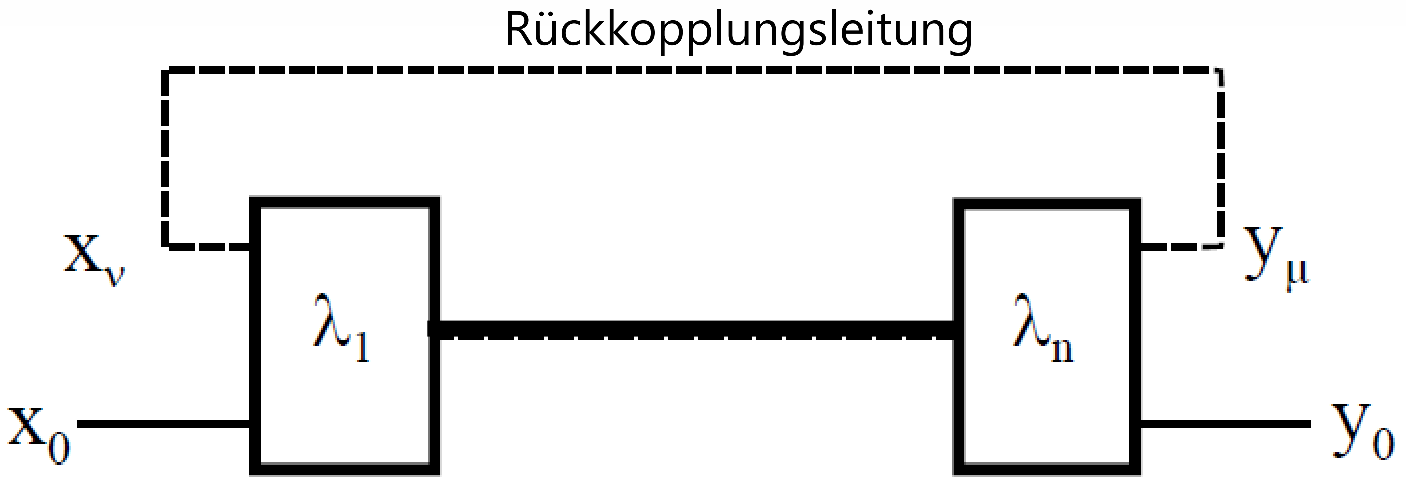


# Definitionen



Für die Schaltnetze gilt weiter, dass kein Ausgangssignal der Teilmodule auf in der Verkettung davor liegende Module gelegt werden darf (keine Rückkopplungszweige).

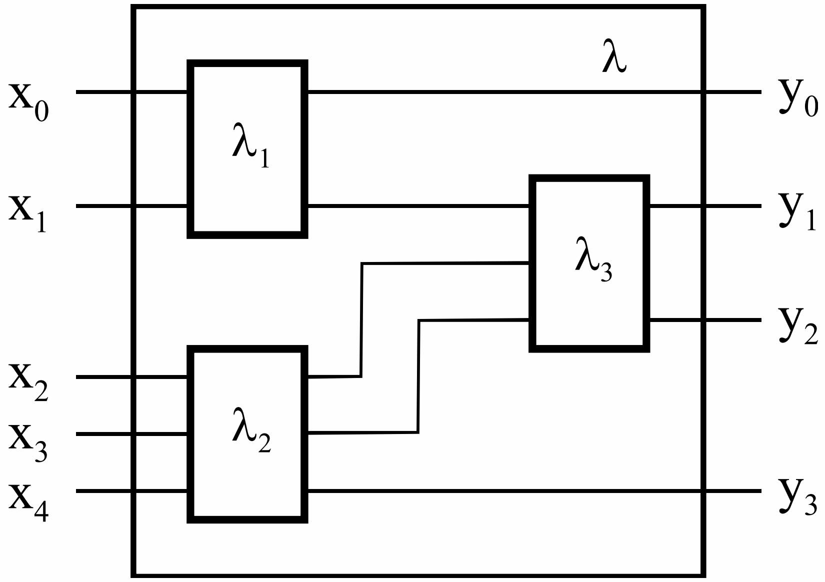
Durch eine Rückkopplung wird aus einer kombinatorischen Logikfunktion eine **sequenzielle Schaltung**, die in Kapitel 3 behandelt werden:



# Definitionen



Bei komplexen logischen Funktionen kann man auch den Entwurf strukturieren, in dem man das gesamte Schaltnetz der Funktion λ in eine Kombination von Teilnetzen (Modulen) $\lambda_p$ zerlegt:



# Beschreibungs- und Darstellungsformen



Mit Hilfe einer **Wahrheitstabelle** können die Zusammenhänge zwischen den Eingangsvariablen und den Ausgangsgrößen beschrieben werden.

1 Bit Volladdierer
Wahrheitstabelle

Drei Eingänge:
- Zwei 1-bit Zahlen
- 1-bit $\text{Carry}_{\text{in}}$

Zwei Ausgänge:
- 1-bit Summe
- 1-bit $\text{Carry}_{\text{out}}$

$2^3 = 8$

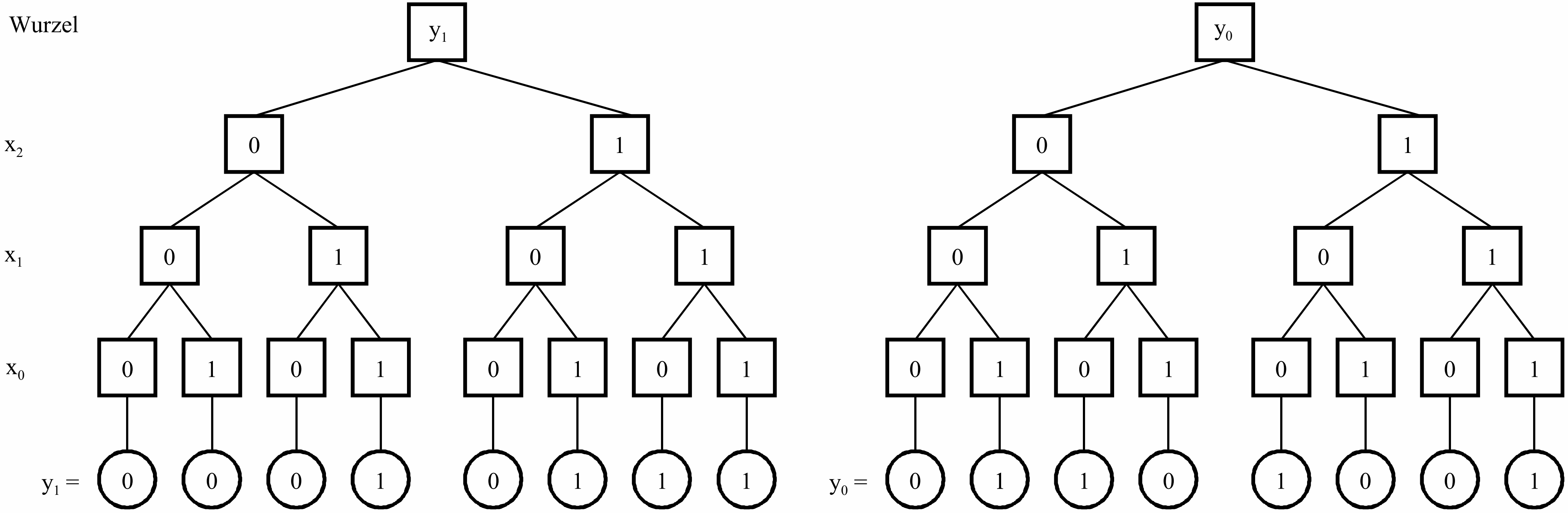
| Eingangsvariablen | | | Ausgangsvariablen | |
|---|---|---|---|---|
| $X_2$ | $X_1$ | $X_0$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

# Beschreibungs- und Darstellungsformen



Die Wahrheitstabelle des Volladdierers lässt sich auch in Form eines "Entscheidungsbaumes" beschreiben.

Das folgende Bild zeigt den sog. Shannon-Baum für das Beispiel des Volladdierers:



# Beschreibungs- und Darstellungsformen



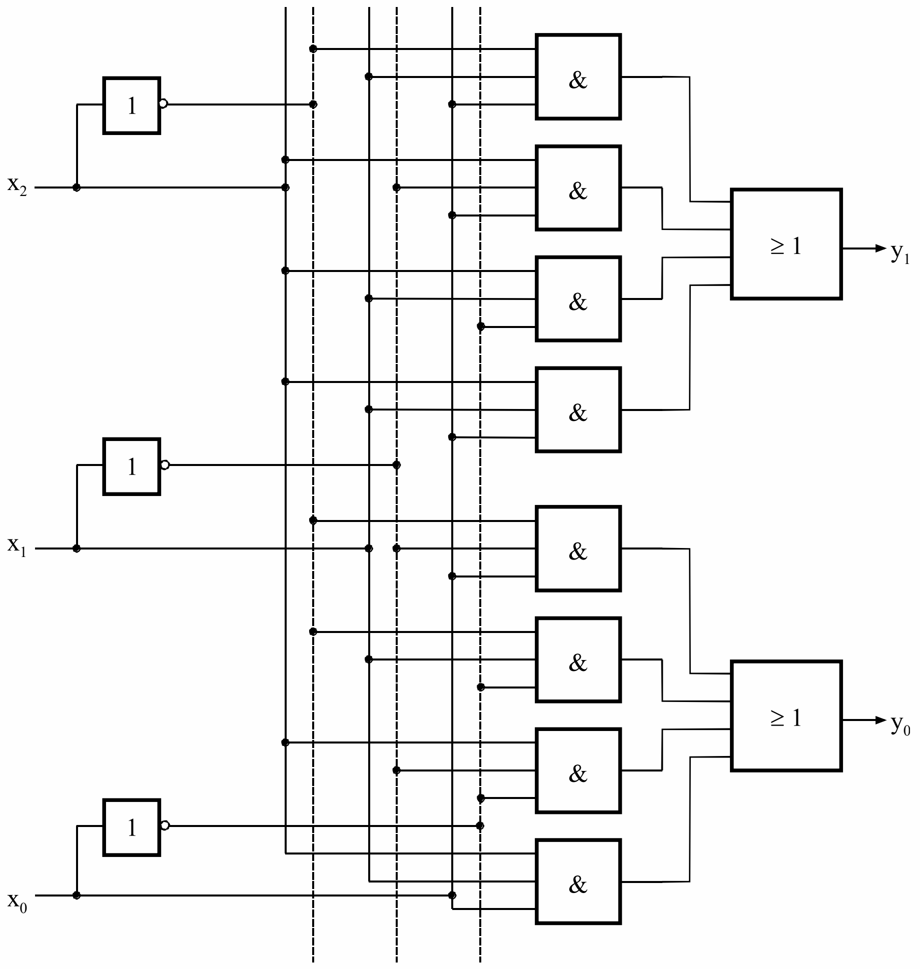
Jede logische Funktion kann auf **Verkettung von Elementarfunktionen** zurückführen.

Grosses Vorteil: nur eine begrenzte Zahl von Standardmodulvarianten sind nötig.

Beispiel:

Ein **Volladdierer** addiert zwei Binärvariablen und generiert einen Übertrag (zur Verkettung mehrerer Binärstellen).

Die Realisierung kommt mit drei Elementarfunktionen (1, &, ≥1) aus



# Beschreibungs- und Darstellungsformen



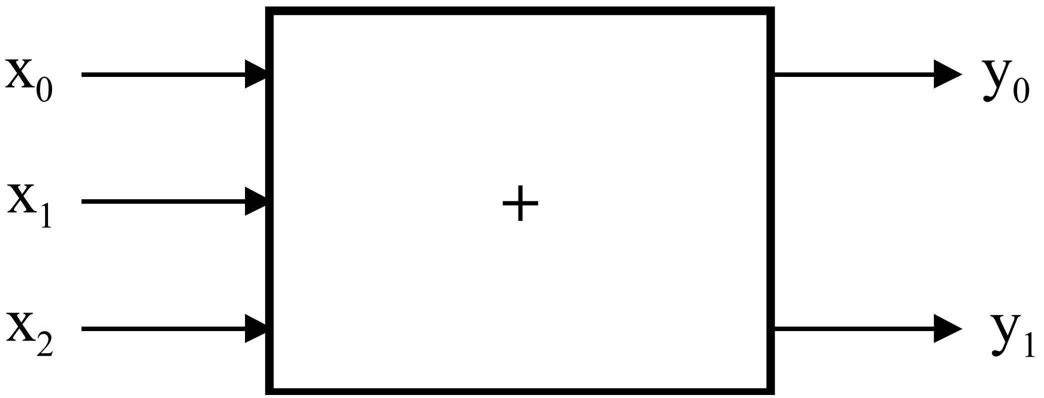
| | | |
|---|---|---|
| | Wahrheitstabelle | einzigartig |
| | ↕ | |
| geeignet für Manipulation | Boolescher Ausdruck | nicht einzigartig |
| | ↕ | |
| kurz vor der Implementierung | Gatterdarstellung | nicht einzigartig |

# Beschreibungs- und Darstellungsformen



Für den im Beispiel verwendeten Volladdierer kann das im Bild gezeichnete **Kurzsymbol** angegeben werden.

Neben den Eingangs- und Ausgangsvariablen wird die Funktion des Schaltnetzes durch das Zeichen "+" beschrieben.



# Beschreibungs- und Darstellungsformen



Der Zusammenhang zwischen Eingangs- und Ausgangsvariablen lässt sich auch durch eine algebraische Gleichung (**Schaltalgebra**) von binären Variablen mit den elementaren logischen Verknüpfungen angeben

Für das Beispiel des Volladdierers ergibt sich:

$$y_0 = \overline{x}_2\,\overline{x}_1\,x_0 \;\vee\; \overline{x}_2\,x_1\,\overline{x}_0 \;\vee\; x_2\,\overline{x}_1\,\overline{x}_0 \;\vee\; x_2\,x_1\,x_0$$

$$y_1 = \overline{x}_2\,x_1\,x_0 \;\vee\; x_2\,\overline{x}_1\,x_0 \;\vee\; x_2\,x_1\,\overline{x}_0 \;\vee\; x_2\,x_1\,x_0$$

Die dafür verwendeten logischen Verknüpfungen werden auf den nächsten Folien erläutert.

# Schaltfunktionen und Schaltalgebra



Eine Schaltfunktion ("logische Funktion") ist die Abhängigkeit einer Variablen y von binären Eingangsvariablen $x$.

***Logische Funktionen von einer Eingangsvariablen:***

Für eine Variable ($x$) gibt es $2^1$ Eingangskombinationen und $2^2$ mögliche Funktionen $y_i$

| Eingangs-kombination | Eingangs-variable | Ausgangsfunktion | | | | Bezeichnung |
|---|---|---|---|---|---|---|
| | $x_0$ | $y_0^1$ | $y_1^1$ | $y_2^1$ | $y_3^1$ | Algebraische Darstellung |
| | | 0 | $x_0$ | $\bar{x}_0$ | 1 | |
| $x^0$ | 0 | 0 | 0 | 1 | 1 | Funktionstabelle |
| $x^1$ | 1 | 0 | 1 | 0 | 1 | |
| | | Nullfunktion | Identität | Negation | Einsfunktion | |

Anm.: die Negation einer Variablen soll im Folgenden durch einen Überstrich der Variablen gekennzeichnet werden ($\bar{x}_0$, sprich: „$x_0$ quer“)

# Schaltfunktionen und Schaltalgebra



Mit logischen Funktionen von einer Variablen lassen sich noch keine verketteten Module zu Realisierung komplexer Schaltnetze aufbauen (min. 2 Eingänge).

Logischen Funktion von zwei Eingangsvariablen

Modulverkettungen aufbauen lassen

Enthalten minimal benötigten Satz von Elementarfunktionen

Allgemein gibt es für n Eingangsvariablen:

$K = 2^n$ Kombinationen und

$F = 2^K$ Funktionen

# Schaltfunktionen mit Zwei Variable



Zwei Variable → $2^4 = 16$ Funktionen $y_i^2$

| | | Algebra. Darstellung | Funktionstabelle | | | | Benennung | DIN EN 60617-12 Schaltsymbol |
|---|---|---|---|---|---|---|---|---|
| Ausgangsfunktion | $y_0^2$ | $0$ | 0 | 0 | 0 | 0 | Nullfunktion | 0 --------- |
| | $y_1^2$ | $x_1 \wedge x_0$ | 0 | 0 | 0 | 1 | AND (Konjunktion) | & |
| | $y_2^2$ | $x_1 \wedge \overline{x}_0$ | 0 | 0 | 1 | 0 | (UND) | $x_1$, $x_0$ & |
| | $y_3^2$ | $x_1$ | 0 | 0 | 1 | 1 | (Identität) | $x_1$ --------- |
| | $y_4^2$ | $\overline{x}_1 \wedge x_0$ | 0 | 1 | 0 | 0 | (UND) | $x_1$, $x_0$ & |
| | $y_5^2$ | $x_0$ | 0 | 1 | 0 | 1 | (Identität) | $x_0$ --------- |
| | $y_6^2$ | $x_1 \not\equiv x_0$ | 0 | 1 | 1 | 0 | XOR (Antivalenz) | =1 |
| | $y_7^2$ | $x_1 \vee x_0$ | 0 | 1 | 1 | 1 | OR (Disjunktion) | ≥1 |
| | $y_8^2$ | $\overline{x_1 \vee x_0}$ | 1 | 0 | 0 | 0 | NOR (Peirce-Fkt.) | ≥1 |
| | $y_9^2$ | $x_1 \equiv x_0$ | 1 | 0 | 0 | 1 | Äquivalenz | = |
| | $y_{10}^2$ | $\overline{x}_0$ | 1 | 0 | 1 | 0 | (Negation) | $x_0$ 1 |
| | $y_{11}^2$ | $x_1 \vee \overline{x}_0$ | 1 | 0 | 1 | 1 | Implikation | $x_1$, $x_0$ ≥1 |
| | $y_{12}^2$ | $\overline{x}_1$ | 1 | 1 | 0 | 0 | (Negation) | $x_1$ 1 |
| | $y_{13}^2$ | $\overline{x}_1 \vee x_0$ | 1 | 1 | 0 | 1 | Implikation | $x_1$, $x_0$ ≥1 |
| | $y_{14}^2$ | $\overline{x_1 \wedge x_0}$ | 1 | 1 | 1 | 0 | NAND (Sheffer-Fkt.) | & |
| | $y_{15}^2$ | $1$ | 1 | 1 | 1 | 1 | Einsfunktion | 1 --------- |
| Eingangs-variablen | | $x_1$ | 0 | 0 | 1 | 1 | | |
| | | $x_0$ | 0 | 1 | 0 | 1 | | |
| Kombination der Eingangsvariablen | | | $x^0$ | $x^1$ | $x^2$ | $x^3$ | | |

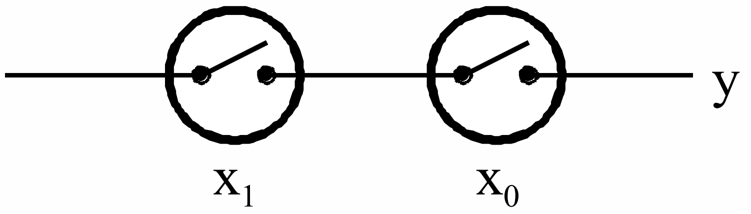
$2^2 = 4$ Eingangskombinationen $x^i$

# Logische Grundverknüpfungen von 2 Variablen



## 1. UND-Verknüpfung:

Ob in einer Leitung ein Strom fließt, hängt von einer bestimmten Wertekombination der Variablen $x_0$ und $x_1$ ab. Bei einer Reihenschaltung kann y nur dann 1 sein, wenn $x_0$ und $x_1$ gleichzeitig 1 sind, d.h. beide Schalter geschlossen sind.




| $x_1$ | $x_0$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Formale Beschreibung der Bedingung: $\mathbf{y = x_1 \wedge x_0 \; (= x_1 \cdot x_0 = x_1 x_0\,)}$

(Anm.: da UND im Ergebnis wie eine binäre Multiplikation ist, werden auch die Schreibweisen in der Klammer verwendet)

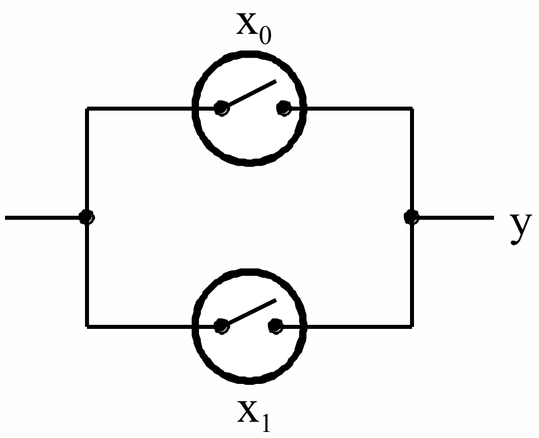
Es handelt sich dabei um eine **UND-Verknüpfung** (AND, Konjunktion)

# Logische Grundverknüpfungen von 2 Variablen



## 2. ODER-Verknüpfung

Schaltet man die Elemente parallel, ergibt sich eine ODER-Verknüpfung, welche die Schaltfunktion y = 1 immer dann erfüllt, wenn eine der beiden Variablen $x_0$ oder $x_1$ den Wert 1 hat.




| $x_1$ | $x_0$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

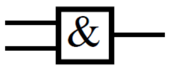
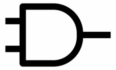
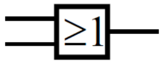
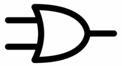
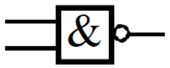
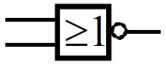
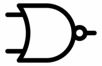
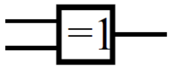
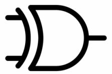
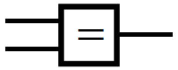
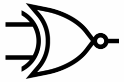
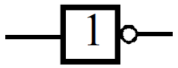
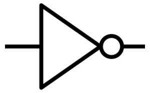
Formale Beschreibung der Bedingung: $\mathbf{y = x_1 \vee x_0 \; (= x_1 + x_0)}$

(Anm.: da OR im Ergebnis ***fast*** wie eine binäre Addition ist, werden auch die Schreibweisen in der Klammer verwendet)

Es handelt sich dabei um eine **ODER-Verknüpfung** (OR, Disjunktion)

# Gebräuchlichen Elementaren Funktionen

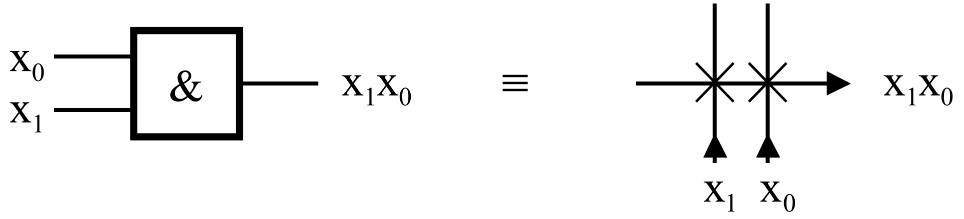
Alle Funktionen können durch Zusammenschaltung weniger Grundfunktionen realisiert werden.

| Benennung | Formel | Schaltsymbol DIN EN 60617-12 | US Symbol | |
|---|---|---|---|---|
| UND (AND) | $x_1 \wedge x_0$<br>$(x_1 x_0)$<br>$(x_1 \cdot x_0)$ | <br> |  | übliche Funktion |
| ODER (OR) | $x_1 \vee x_0$ | <br> |  | übliche Funktion |
| NAND | $\overline{x_1 \wedge x_0}$ | <br> |  | |
| NOR | $\overline{x_1 \vee x_0}$ | <br> |  | |
| XOR (Antivalenz) | $x_1 \not\equiv x_0$<br>$(x_1 \oplus x_0)$ | <br> |  | |
| Äquivalenz | $x_1 \equiv x_0$ | <br> |  | |
| Negation | $\overline{x}$ | <br> |  | übliche Funktion |

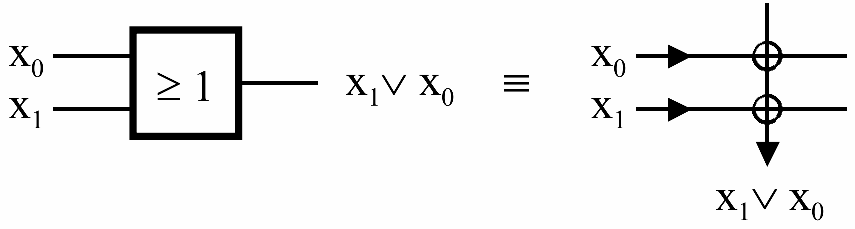
(Auch XNOR gennant)

# Mehr Graphischen Darstellungen

UND- und ODER-Schaltung in “verdrahteten Logik” Form




Bei der UND-schaltung sind die Ausgänge waagerecht gezeichnet




Bei der ODER-schaltung sind die Ausgänge senkrecht gezeichnet

Diese Symbolik ist häufig benutzt weil sie einfache, komprimierte Darstellungen logischer Verknüpfungen erlaubt und auch der technischen Realisierung moderner Integrationstechnik.

Richtungspfeile sind nötig wenn der Signalfluss nicht aus dem Kontext hervorgeht

# Boolesche Algebra



Logik: Die Untersuchung der Prinzipien des logischen Denkens.

George Boole, Mathematik im 19. Jahrhundert, entwickelte ein mathematisches System (Algebra) mit Logik, mit binären Variablen.

Später zeigte Claude Shannon (Vater der Informationstheorie, in seiner Magisterarbeit!), wie man die Boolesche Algebra auf digitale Schaltungen übertragen kann.

# Boolesche Algebra



**Axiome und Sätze**, hier zum Ziel der Vereinfachung Boolescher Gleichungen

- Wie die übliche Algebra
- Teilweise einfacher, da hier nur zwei Werte

Axiome und Sätze haben jeweils **duale Entsprechung**:

- Tausche AND/OR, tausche 0/1

# Axiome der Booleschen Algebra



| Nummer | Axiom | Name |
|---|---|---|
| A1 | $B = 0$ if $B \neq 1$ | Dualitätsgesetz |
| A2 | $\overline{0} = 1$ | NOT |
| A3 | $0 \bullet 0 = 0$ | AND/OR |
| A4 | $1 \bullet 1 = 1$ | AND/OR |
| A5 | $0 \bullet 1 = 1 \bullet 0 = 0$ | AND/OR |

# Axiome der Booleschen Algebra



| Nummer | Axiom | Name |
|---|---|---|
| A1 | $B = 0$ if $B \neq 1$ | Dualitätsgesetz |
| A2 | $\overline{0} = 1$ | NOT |
| A3 | $0 \cdot 0 = 0$ | AND/OR |
| A4 | $1 \cdot 1 = 1$ | AND/OR |
| A5 | $0 \cdot 1 = 1 \cdot 0 = 0$ | AND/OR |

**Dual:** Tausche: $\cdot$ mit $+$
0 mit 1

# Axiome der Booleschen Algebra



| Nummer | Axiom | Dual | Name |
|---|---|---|---|
| A1 | $B = 0$ if $B \neq 1$ | $B = 1$ if $B \neq 0$ | Dualitätsgesetz |
| A2 | $\neg 0 = 1$ | $\neg 1 = 0$ | NOT |
| A3 | $0 \bullet 0 = 0$ | $\boxed{1 + 1 = 1}$ | AND/OR |
| A4 | $1 \bullet 1 = 1$ | $0 + 0 = 0$ | AND/OR |
| A5 | $0 \bullet 1 = 1 \bullet 0 = 0$ | $1 + 0 = 0 + 1 = 1$ | AND/OR |

**Dual:** Tausche: • mit +
0 mit 1

# Gundlegende Definitionen



**Komplement:** Boolesche Variable mit einem Balken oder Strich (invertiert)

$A', B', C'$ (auch $\bar{A}, \bar{B}, \bar{C}$)

**Literal:** Variable oder ihr Komplement

$A, A, B, B, C, C$

**Implikant:** Produkt von Literalen

$ABC, AC, BC$

**Minterm:** Produkt (UND, Konjunktion) über alle Eingangsvariablen

$ABC, ABC, ABC$

**Maxterm:** Summe (ODER, Disjunktion) über alle Eingangsvariablen

$(A+B+C), (A+B+C), (A+B+C)$

# Sätze der Booleschen Algebra



| Nr. | Satz | Name |
|---|---|---|
| T1 | $B \cdot 1 = B$ | Neutralitätsgesetz |
| T2 | $B \cdot 0 = 0$ | Extremalgesetz |
| T3 | $B \cdot B = B$ | Idempotenzgesetz |
| T4 | $\overline{\overline{B}} = B$ | Involution |
| T5 | $B \cdot \overline{B} = 0$ | Komplementärgesetz |

# Sätze der Booleschen Algebra



| Nr. | Satz | Name |
|---|---|---|
| T1 | $B \bullet 1 = B$ | Neutralitätsgesetz |
| T2 | $B \bullet 0 = 0$ | Extremalgesetz |
| T3 | $B \bullet B = B$ | Idempotenzgesetz |
| T4 | $\overline{\overline{B}} = B$ | Involution |
| T5 | $B \bullet \overline{B} = 0$ | Komplementärgesetz |

**Dual:** Tausche: • mit +

0 mit 1

# Sätze der Booleschen Algebra



| Nr. | Satz | Dual | Name |
|---|---|---|---|
| T1 | $B \bullet 1 = B$ | $B + 0 = B$ | Neutralitätsgesetz |
| T2 | $B \bullet 0 = 0$ | $B + 1 = 1$ | Extremalgesetz |
| T3 | $B \bullet B = B$ | $B + B = B$ | Idempotenzgesetz |
| T4 | $\overline{\overline{B}} = B$ | | Involution |
| T5 | $B \bullet \overline{B} = 0$ | $B + \overline{B} = 1$ | Komplementärgesetz |

**Dual:** Tausche: $\bullet$ mit $+$
0 mit 1

# Sätze der Booleschen Algebra mit mehreren Variablen



| Nr. | Satz | Name |
|---|---|---|
| T6 | B•C = C•B | Kommutativgesetz |
| T7 | (B•C) • D = B • (C • D) | Assoziativgesetz |
| T8 | B • (C + D) = (B•C) + (B•D) | Distributivgesetz |
| T9 | B• (B+C) = B | Absorptionsgesetz |
| T10 | (B•C) + (B•C) = B | Zusammenfassen |
| T11 | (B•C) + (B•D) + (C•D) = (B•C) + (B•D) | Konsensusregeln |

# Sätze der Booleschen Algebra mit mehreren Variablen



| Nr. | Satz | Name |
|---|---|---|
| T6 | B•C = C•B | Kommutativgesetz |
| T7 | (B•C) • D = B • (C • D) | Assoziativgesetz |
| T8 | B • (C + D) = (B•C) + (B•D) | Distributivgesetz |
| T9 | B• (B+C) = B | Absorptionsgesetz |
| T10 | (B•C) + (B•C) = B | Zusammenfassen |
| T11 | (B•C) + (B•D) + (C•D) = (B•C) + (B•D) | Konsensusregeln |

**Dual:** Austauschen: • mit +
0 mit 1

# Sätze der Booleschen Algebra mit mehreren Variablen



| Nr. | Theorem | Dual | Name |
|---|---|---|---|
| T6 | B•C = C•B | B+C = C+B | Kommutativgesetz |
| T7 | (B•C) • D = B • (C•D) | (B + C) + D = B + (C + D) | Assoziativgesetz |
| T8 | B • (C + D) = (B•C) + (B•D) | B + (C•D) = (B+C) (B+D) | Distributivgesetz |
| T9 | B • (B+C) = B | B + (B•C) = B | Absorptionsgesetz |
| T10 | (B•C) + (B•C) = B | (B+C) • (B+C) = B | Zusammenfassen |
| T11 | (B•C) + (B•D) + (C•D) = (B•C) + (B•D) | (B+C) • (B+D) • (C+D) = (B+C) • (B+D) | Konsensusregeln |

**Dual:** Austauschen: • mit +
0 mit 1

# Sätze der Booleschen Algebra mit  mehreren Variablen



| Nr. | Theorem | Dual | Name |
|---|---|---|---|
| T6 | B•C = C•B | B+C = C+B | Kommutativgesetz |
| T7 | (B•C) • D = B • (C•D) | (B + C) + D = B + (C + D) | Assoziativgesetz |
| T8 | B • (C + D) = (B•C) + (B•D) | B + (C•D) = (B+C) (B+D) | Distributivgesetz |
| T9 | B • (B+C) = B | B + (B•C) = B | Absorptionsgesetz |
| T10 | (B•C) + (B•C) = B | (B+C) • (B+C) = B | Zusammenfassen |
| T11 | (B•C) + (B•D) + (C•D) = (B•C) + (B•D) | (B+C) • (B+D) • (C+D) = (B+C) • (B+D) | Konsensusregeln |

**Warnung:** T8’ ist mit normaler Algebra ungleich:
ODER (+) wird über UND (•) verteilt

# De Morgan'sches Gesetz



| Nr. | Theorem | Dual | Name |
|---|---|---|---|
| T12 | $\overline{B_0 \bullet B_1 \bullet B_i \bullet \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_i} + \ldots$ | $\overline{B_0 + B_1 + B_i + \cdots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_i} \bullet \ldots$ | DeMorgansche Gesetz |

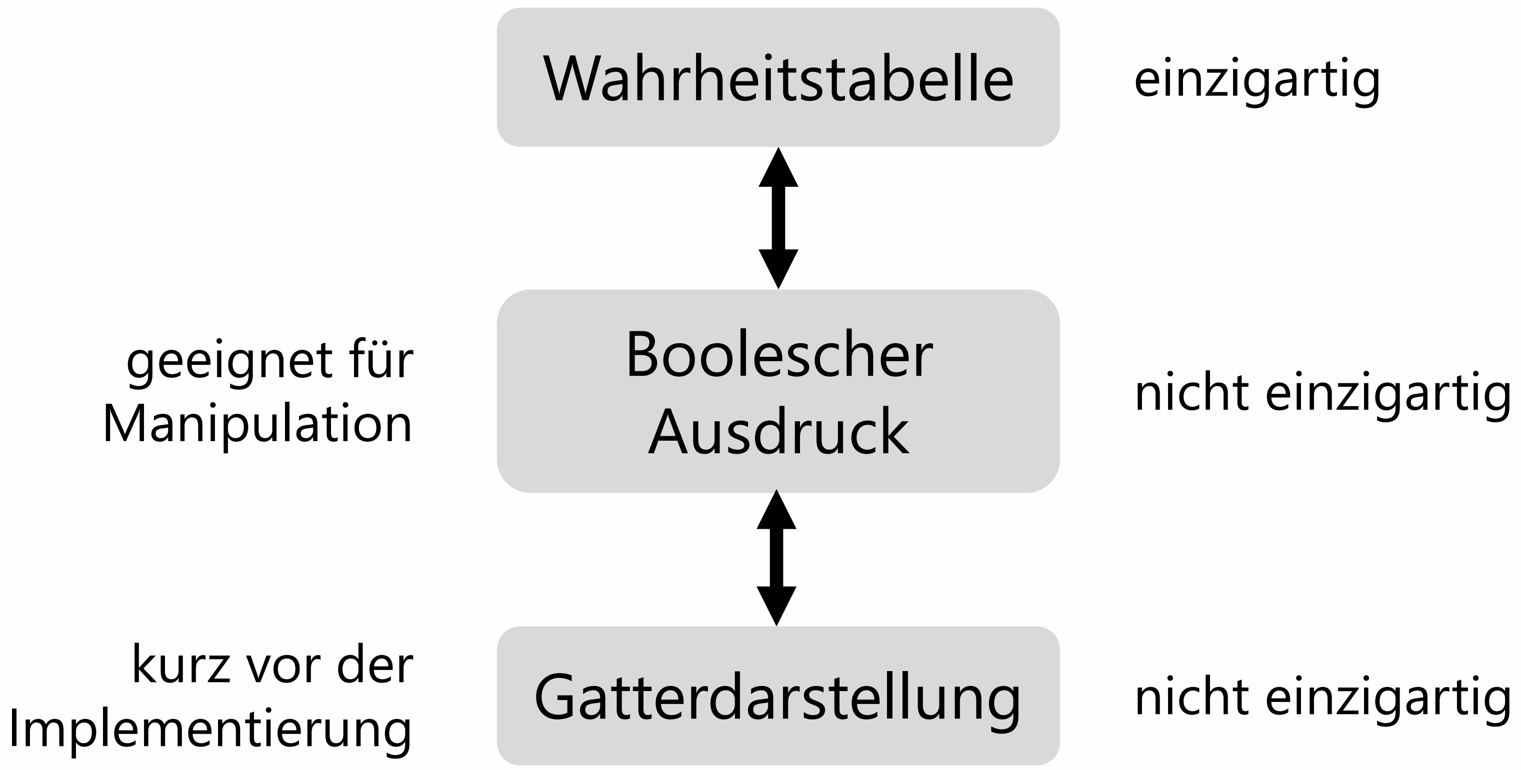
Das **Komplement** des **Produkts** ist die **Summe** der **Komplementen.**

Das **Komplement** der **Summe** ist das **Produkt** der **Komplementen**.

| $x_1$ | $x_0$ | $x_1 \wedge x_0$ | $\overline{x_1 \wedge x_0}$ | $\overline{x}_1$ | $\overline{x}_0$ | $\overline{x}_1 \vee \overline{x}_0$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

=

# Beschreibungs- und Darstellungsformen




*Wie konvertiert man vom einen zum anderen?*

# Kanonische Formen



Standardform für einen booleschen Ausdruck - eindeutiger algebraischer Ausdruck direkt aus einer Wahrheitstabllle Beschreibung.

Zwei Arten:

Summe der Produkte (SOP)

Produkt der Summen (POS)

# Disjunktive Normalform



Summe aus Produkte (*SOP*). Beispiel:

| **Minterms** | **a** | **b** | **c** | **f** | **f'** |
|---|---|---|---|---|---|
| a'b'c' | 0 | 0 | 0 | 0 | 1 |
| a'b'c | 0 | 0 | 1 | 0 | 1 |
| a'bc' | 0 | 1 | 0 | 0 | 1 |
| a'bc | 0 | 1 | 1 | 1 | 0 |
| ab'c' | 1 | 0 | 0 | 1 | 0 |
| ab'c | 1 | 0 | 1 | 1 | 0 |
| abc' | 1 | 1 | 0 | 1 | 0 |
| abc | 1 | 1 | 1 | 1 | 0 |

Ein Produkt (und) Term für jede ***1*** in **f** (minterm):

`f = a'bc + ab'c' + ab'c + abc' + abc`

`f' = a'b'c' + a'b'c + a'bc'`

Wie viel kostet das?

# Kanonische Formen sind in der Regel nicht minimal:



Wir können boolesche Formeln mithilfe von Axiomen und Theoremen minimieren

Beispiel:

$f = a'bc + ab'c' + ab'c + abc' + abc$ — **10x** UND2 + **4x** ODER2

$= a'bc + ab' + ab$

$= a'bc + a$

$= a + a'bc$ — siehe mehr gesetze

$= (a + a')(a + bc)$

$= a + bc$ — **1x** UND2 + **1x** ODER2

$f' = a'b'c' + a'b'c + a'bc'$ — **6x** UND2 + **2x** ODER2

$= (a'b'c' + a'b'c') + a'b'c + a'bc'$

$= a'b'(c'+c) + a'b'c' + a'bc'$

$= a'(b' + b'c' + bc')$

$= a'(b' + c'(b'+b))$

$= a'b' + a'c'$ — **2x** UND2 + **1x** ODER2

# Konjunktive Normalform



Produkte aus Summe (*POS*). Beispiel:

| Maxterms | a | b | c | f | f' |
|---|---|---|---|---|---|
| a+b+c | 0 | 0 | 0 | 0 | 1 |
| a+b+c' | 0 | 0 | 1 | 0 | 1 |
| a+b′+c | 0 | 1 | 0 | 0 | 1 |
| a+b′+c' | 0 | 1 | 1 | 1 | 0 |
| a′+b+c | 1 | 0 | 0 | 1 | 0 |
| a′+b+c' | 1 | 0 | 1 | 1 | 0 |
| a′+b′+c | 1 | 1 | 0 | 1 | 0 |
| a′+b′+c' | 1 | 1 | 1 | 1 | 0 |

Ein Summe (or) Term für jede ***0*** in **f** (maxterm):

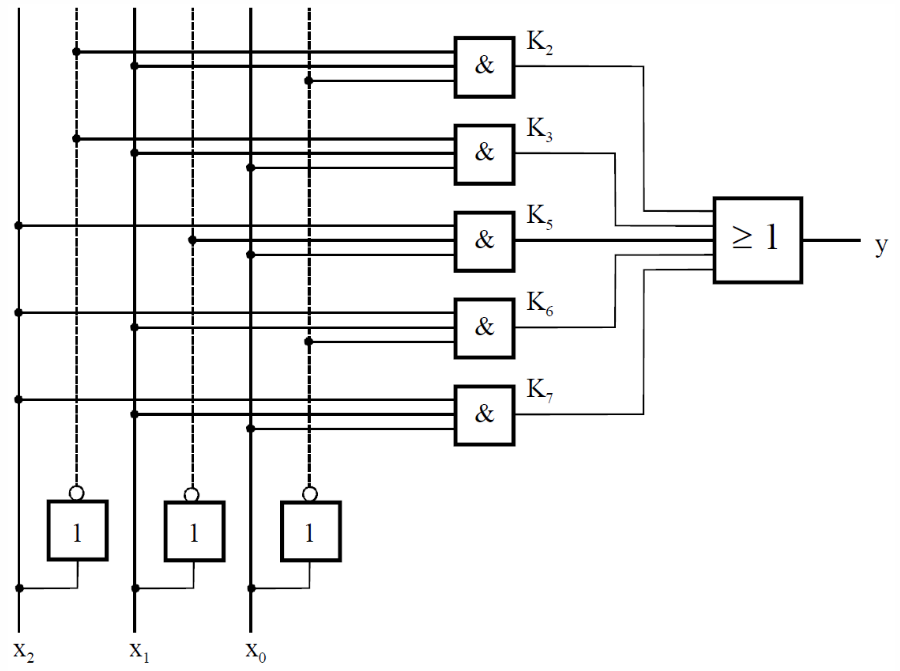
**f = (a+b+c) (a+b+c') (a+b'+c)**

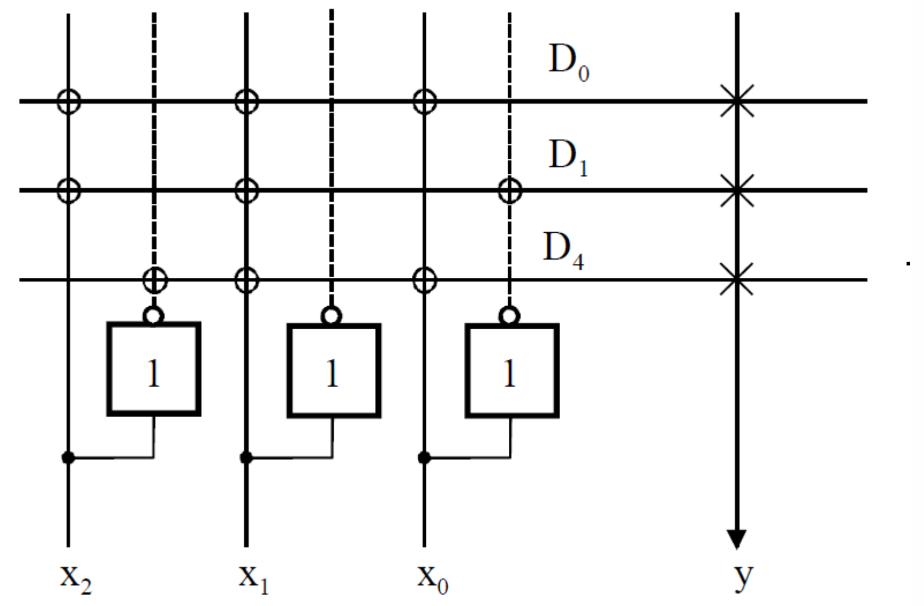
**f ' = (a+b'+c′)(a′+b+c)(a'+b+c′)(a'+b′+c)(a+b+c')**

# Graphische Darstellungen



Sind diese in DNF oder KNF?



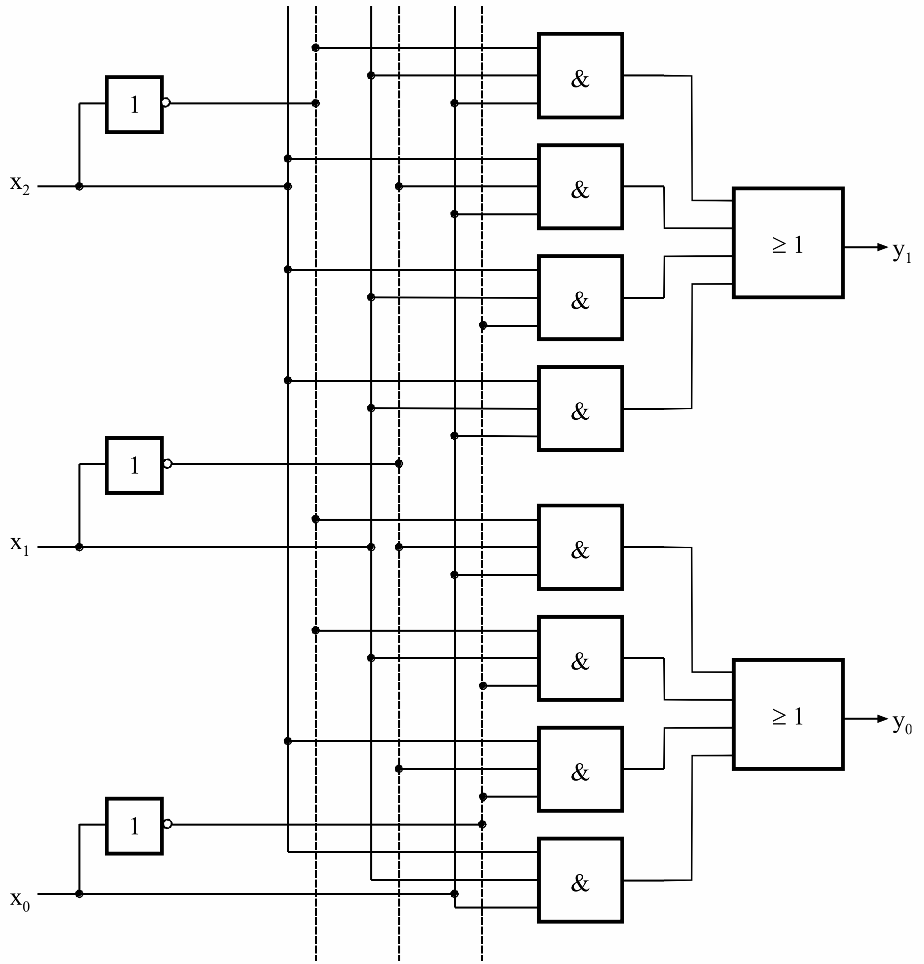
# Zurück zum Volladdierer



| Eingangsvariablen | | | Ausgangsvariablen | |
|---|---|---|---|---|
| $X_2$ | $X_1$ | $X_0$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$$Y_0 = \bar{X}_2\bar{X}_1X_0 + \bar{X}_2X_1\bar{X}_0 + X_2\bar{X}_1\bar{X}_0 + X_2X_1X_0$$

$$Y_1 = \bar{X}_2X_1X_0 + X_2\bar{X}_1X_0 + X_2X_1\bar{X}_0 + X_2X_1X_0$$

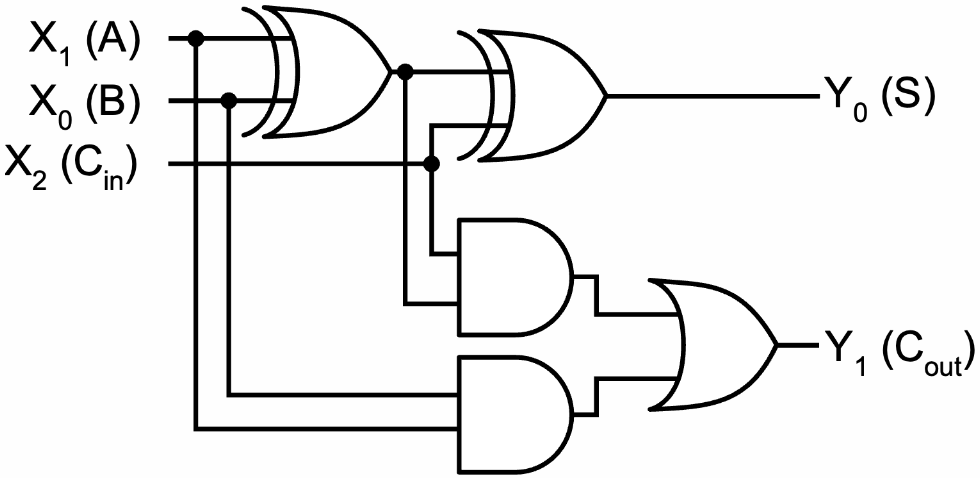


# Minimierung eines Volladdierers



$$
\begin{aligned}
C_{out} &= a'bc + ab'c + abc' + abc \\
&= a'bc + ab'c + abc' + abc + abc \\
&= a'bc + abc + ab'c + abc' + abc \\
&= (a' + a)bc + ab'c + abc' + abc \\
&= (1)bc + ab'c + abc' + abc \\
&= bc + ab'c + abc' + abc + abc \\
&= bc + ab'c + abc + abc' + abc \\
&= bc + a(b' + b)c + abc' + abc \\
&= bc + a(1)c + abc' + abc \\
&= bc + ac + ab(c' + c) \\
&= bc + ac + ab(1) \\
&= bc + ac + ab
\end{aligned}
$$

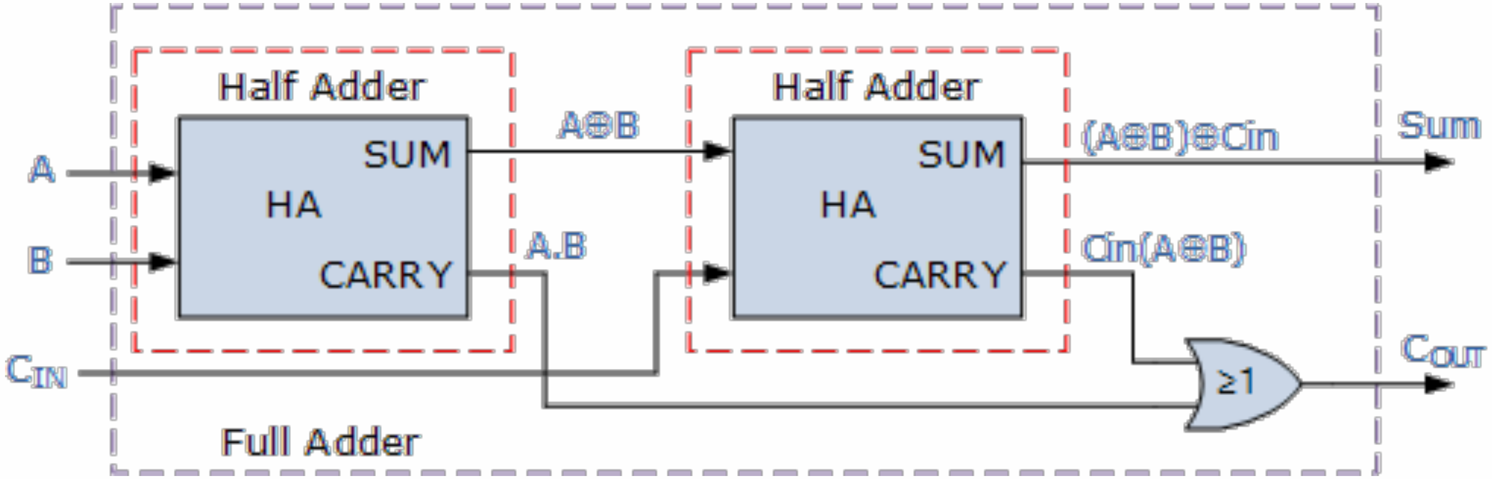
optimierte Volladdiererschaltung mit S und $C_{out}$




S selbst minimieren!

# 2 Halbaddierer = 1 Volladdierer

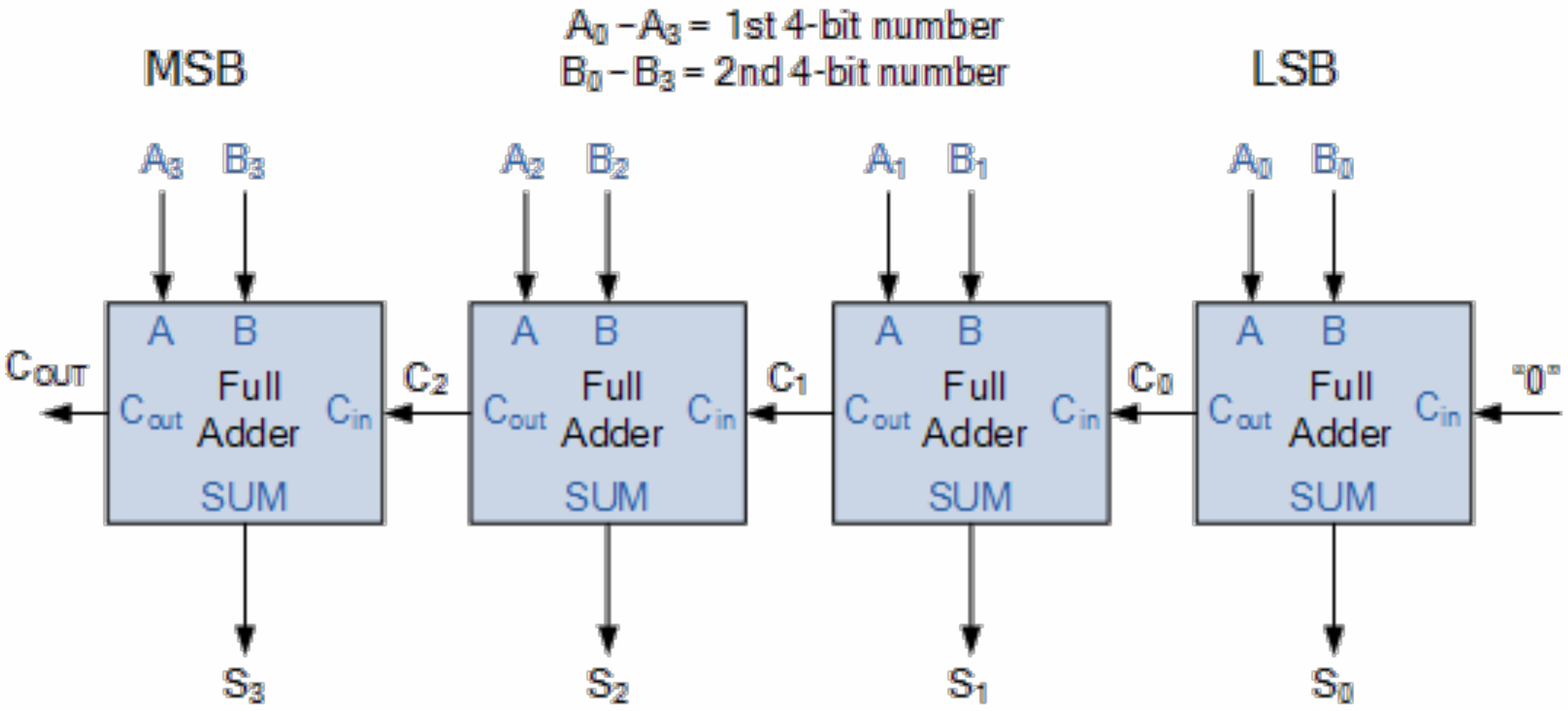





Quelle: ElectronicsTutorials

# Ripple-Carry Addierer






Quelle: ElectronicsTutorials