Digitaltechnik

Kapitel 4, Kombinatorische Schaltungen

Prof. Dr.-Ing. M. Winzker

(Stand: 20.03.2019)

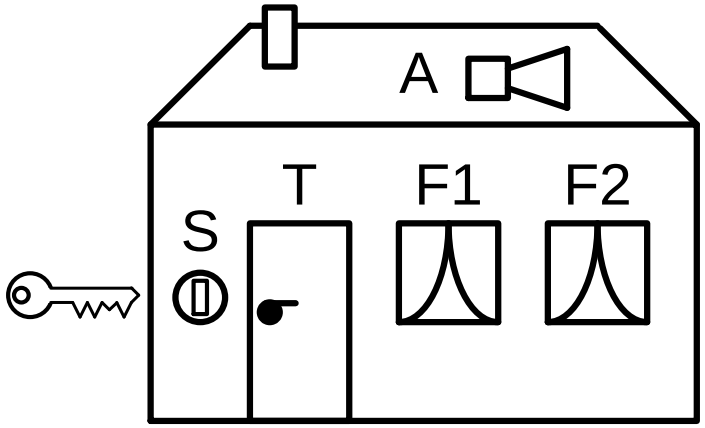
4.1 Schaltfunktionen

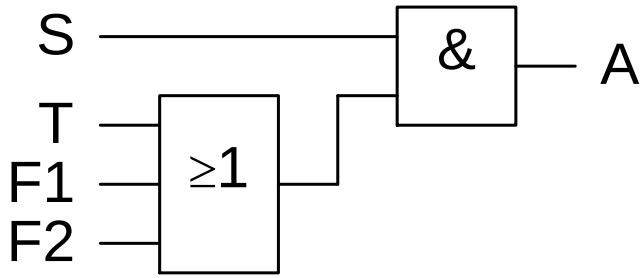
- Die Schaltalgebra verwendet Variablen und Konstanten (ähnlich wie Algebra)
  - In der **Algebra** gibt es **unendlich** viele Konstanten; Variablen können unendlich viele Werte annehmen
  - In der **Schaltalgebra** gibt es nur **zwei** Konstanten (0 und 1); Variable können nur zwei Werte annehmen (0 und 1)
- Funktionen, bei denen Eingangs- und Ausgangswerte nur die Werte 0 und 1 annehmen können, bezeichnet man als **binäre Schaltfunktionen**, **boolesche Schaltfunktionen** oder einfach **Schaltfunktionen**
- Das Adjektiv boolesch weist darauf hin, dass die Funktion nach der **Booleschen Algebra** berechnet wird (nach dem engl. Mathematiker G. Boole)
- Digitale Schaltungen führen in **Schaltgliedern** eine **boolesche Verknüpfung** von **Eingangsvariablen** zu einer **Ausgangsvariablen** durch
  - Schaltgliedern bezeichnet man auch **Verknüpfungsglieder** oder **Gatter**
  - Statt boolesch spricht man auch von **logisch** (z.B. logische Verknüpfung)

Die Schaltalgebra beschreibt die Rechenregeln zum Umgang mit den Werten 0 und 1; physikalische Eigenschaften, wie Spannungspegel, werden nicht beschrieben

Beispiel: Einfache Alarmanlage

- Eine Tür (T) und zwei Fenster (F1, F2) sollen überwacht werden
- Mit einem Schalter (S) wird die die Alarmanlage ein- oder ausgeschaltet
  - Binäre Werte also ‚0' oder ‚1'.
  - ‚1' bedeutet jeweils „aktiv“: Tür oder Fenster offen, Anlage eingeschaltet
- Ausgang A zeigt mit einer ‚1' einen Alarm an
  - An A ist eine Alarmhupe angebracht





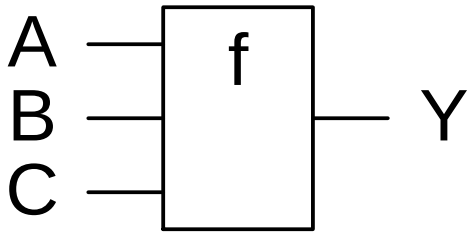
- Erstellung der Schaltung hier direkt aus der Aufgabenstellung
- Später wird systematische Vorgehensweise erläutert

Darstellung von Schaltfunktionen

- Im folgenden bezeichnen A, B, ... Eingangsvariablen und Y die Ausgangsvariable

- In einer Schaltfunktion ist die Ausgangsvariable von den Eingangsvariablen abhängig: Y = f (A, B)
- Eine boolesche Verknüpfung kann durch ein **Schaltzeichen** dargestellt werden

$$Y = f(A,B,C)$$



Funktionstabelle

- Die Funktionalität eines Schaltfunktion wird oft über eine **Funktionstabelle** (**Wahrheitstabelle**) angegeben
- Dabei wird für jede mögliche Kombination der Eingangsvariablen die Ausgangsvariable bestimmt
  - Bei n Eingangsvariablen sind 2^n Kombinationen möglich
  - Für häufig vorkommende Funktionen mit 2-4 Eingangsvariablen ist eine Funktionstabelle noch relativ übersichtlich

Beispiel Majoritätsschaltung

- Schaltung bildet die Mehrheit aus drei Eingangswerten
- Drei Eingänge A, B, C. Wenn zwei oder drei Eingänge ‚1' sind, soll auch der Ausgang Y ‚1' sein. Ansonsten ist der Ausgang ‚0'.
- Anwendung: Sicherheitsschaltung für redundante Systeme
  - Eine Fabrikhalle hat drei Rauchmelder und nur wenn zwei Rauchmelder auslösen, wird ein Alarm gemeldet und die Fabrikation gestoppt.
  - Ein Fehler in einem Rauchmelder kann also keinen Alarm auslösen.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Funktionstabelle mit Don't-Care

- Bei Funktionstabellen kann für eine oder mehrere Eingangskombinationen keine Ausgabe spezifiziert sein
  - Bestimmte Eingangskombinationen kommen laut Aufgabenstellung nicht vor
  - Das Ergebnis bestimmter Eingangskombinationen wird nicht verwendet
- Der nicht definierte Ausgang wird als „Don't-Care" bezeichnet
  - Kennzeichnung mit Strich ‚-' in der Funktionstabelle gekennzeichnet

Beispiel

- Primzahlerkennung für die Zahlen 0 bis 9
  - Zahlen als vierstellige Dualzahl $A(3:0)$
  - 6 von 16 Kombinationen unbenutzt

- Beim Schaltungsentwurf können die Don't-Care-Einträge benutzt werden, um eine möglichst kleine und damit kostengünstige Schaltung zu entwerfen

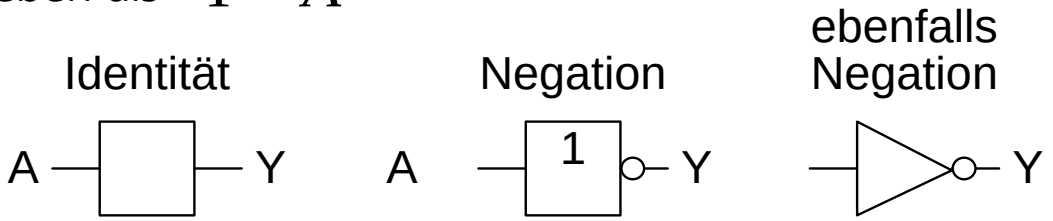
A(3:0)	Y	*Zahlenwert*
0 0 0 0	0	0
0 0 0 1	0	1
0 0 1 0	1	2
0 0 1 1	1	3
0 1 0 0	0	4
0 1 0 1	1	5
0 1 1 0	0	6
0 1 1 1	1	7
1 0 0 0	0	8
1 0 0 1	0	9
1 0 1 0	-	
1 0 1 1	-	
1 1 0 0	-	
1 1 0 1	-	
1 1 1 0	-	
1 1 1 1	-	

Verknüpfung mit einer Schaltvariable

- Frage: Wie viele verschiedene Schaltfunktionen mit einer Eingangsvariable gibt es?
- Antwort: 4

	Ausgang Y		
für:	A=0	A=1	Bezeichnung
	0	0	Konstante 0
	0	1	Identität
	1	0	Negation
	1	1	Konstante 1

- In der Praxis sinnvoll ist die Negation. Sie wird beschrieben als $Y = \overline{A}$ oder $Y = \neg A$ (sprich: „Y ist gleich nicht A“).
- Die Identität wird beschrieben als $Y = A$
- Die Schaltzeichen sind:



- Frage: Wie viele verschiedene Schaltfunktionen mit zwei Eingangsvariablen gibt es?
- Antwort: 16 (Aber einige dieser Schaltfunktionen sind trivial)

Grundverknüpfungen der Schaltalgebra

- Die Grundfunktionen der Schaltalgebra sind **UND-Verknüpfung**, **ODER-Verknüpfung** und **Negation**. Alle anderen Schaltfunktionen lassen sich aus Kombinationen dieser Grundfunktionen darstellen.

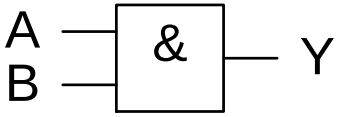
UND-Verknüpfung

- Bei einer UND-Verknüpfung (**Konjunktion**) ist die Ausgangsvariable nur dann 1, wenn Eingang A **und** Eingang B gleich 1 sind
- Die UND-Verknüpfung wird beschrieben als

$$Y = A \wedge B \quad \text{oder} \quad Y = A \& B$$

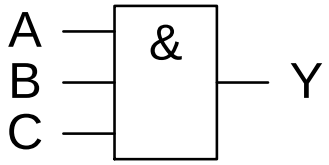
(sprich: „Y ist gleich A und B“)

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



- Mit einer UND-Verknüpfung können auch mehr als zwei Variablen verknüpft werden

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Grundverknüpfungen der Schaltalgebra (II)

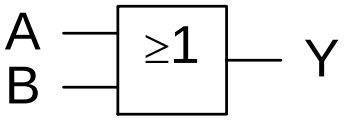
ODER-Verknüpfung

- Bei einer ODER-Verknüpfung (**Disjunktion**) ist die Ausgangsvariable dann 1, wenn Eingang A **oder** Eingang B gleich 1 sind
- Die ODER-Verknüpfung wird beschrieben als

$$Y = A \vee B$$

(sprich: „Y ist gleich A oder B“)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



- Mit einer ODER-Verknüpfung können auch mehr als zwei Variablen verknüpft werden

- Die ursprüngliche Schreibweise des Erfinders G. Boole war das Multiplikationszeichen ‚·‘ für UND sowie das Additionszeichen ‚+‘ für ODER
- Diese Bezeichnungen werden auch (vor allem im US-amerikanischen) verwendet
- Vermeiden Sie Verwechslungen: (‚+‘ = umgangsspr. „und“ = ODER-Verknüpfung)

Zusammengesetzte boolesche Verknüpfungen

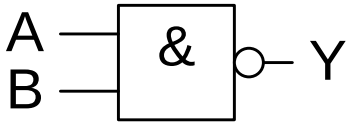
NAND-Verknüpfung

- Eine Reihenschaltung von UND-Verknüpfung mit der Negation ergibt die NAND-Verknüpfung (NAND = „not and“)
- Die NAND-Verknüpfung wird beschrieben als

$$Y = \overline{A \& B}$$ („Y ist gleich A nand B“)

- Die Negation wird im Schaltsymbol durch einen Kreis am Ausgang dargestellt

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

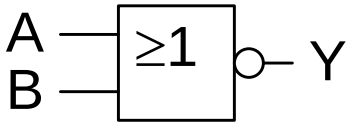


NOR-Verknüpfung

- Eine Reihenschaltung von ODER-Verknüpfung mit der Negation ergibt die NOR-Verknüpfung (NOR = „not or“)
- Die NOR-Verknüpfung wird beschrieben als

$$Y = \overline{A \vee B}$$ („Y ist gleich A nor B“)

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



- NAND und NOR können auch für mehr als zwei Variablen gebildet werden
- Durch Beschalten beider Eingänge mit dem **gleichen Signal** können NAND- und NOR-Logikgatter als **Inverter** benutzt werden

Zusammengesetzte boolesche Verknüpfungen (II)

Antivalenz – XOR

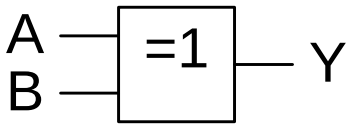
- Die Antivalenz ist nur für zwei Variablen definiert
- Der Ausgang ist 1, wenn die Eingänge **ungleich** sind
- Antivalenz wird meist als **Exklusiv-Oder** (kurz **XOR**) bezeichnet, denn der Ausgang ist 1, wenn A **oder** B gleich 1 sind, **nicht** aber wenn **beide** 1 sind
  - Die Funktion entspricht einem Flurlicht, mit zwei Schaltern
- Die Antivalenz wird beschrieben als

$$Y = A \oplus B = (\overline{A} \,\&\, B) \vee (A \,\&\, \overline{B})$$

(„Y ist gleich A xor B“)

- Das Schaltsymbol hat das Symbol ‚=1‘

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



Zusammengesetzte boolesche Verknüpfungen (III)

Äquivalenz – XNOR

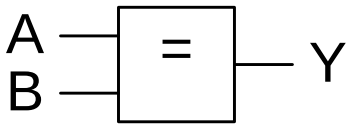
- Die Äquivalenz ist ebenfalls nur für zwei Variablen definiert
- Der Ausgang ist 1, wenn die Eingänge **gleich** sind
- Äquivalenz entspricht der negierten Antivalenz und wird auch als **Exklusiv-NOR** (kurz **XNOR**) bezeichnet
- Die Äquivalenz wird beschrieben als

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$$Y = A \leftrightarrow B = \overline{A \oplus B} = (A \,\&\, B) \vee (\overline{A} \,\&\, \overline{B})$$

(„Y ist gleich A äquivalent B“ oder „Y ist gleich A xnor B“)

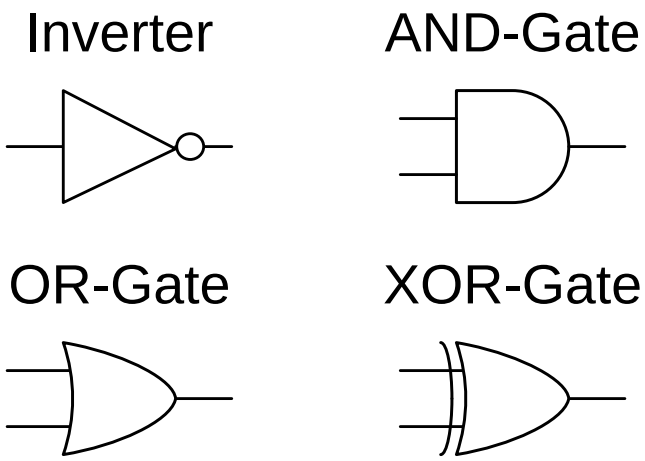
- Das Schaltsymbol hat das Symbol ‚=‘



- Es gibt noch weitere Verknüpfungen (Inhibition, Implikation)
  - Die Funktionen selbst sind sinnvoll
  - Die Begriffe werden in der Praxis jedoch nur selten verwendet

US-Amerikanische Logiksymbole

- In englischsprachiger Literatur und in Datenblättern finden Sie auch Logiksymbole in US-amerikanischer Darstellungsweise
- Durch einen Kreis am Ausgang werden die Varianten mit invertiertem Ausgang gekennzeichnet
  - Aus AND wird NAND, aus XOR wird XNOR.



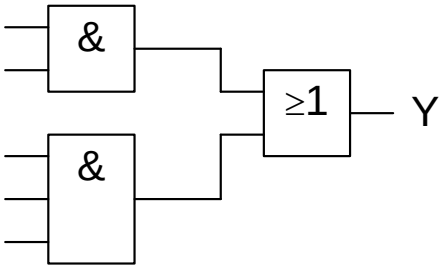
Tipps zum Merken:

- Gerade linke Kante des AND ähnlich wie vertikale Striche des A
- Gebogene linken Kante des OR ähnlich wie Rundungen des O
- Extra-Strich des XOR, wie Extra-Ergänzung des OR

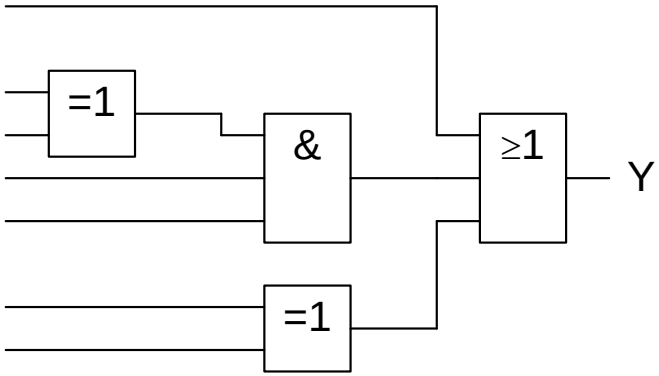
Bezeichnung der Schaltungselemente

- Allgemein werden die Schaltungselemente zu booleschen Verknüpfungen als **Schaltglieder** bezeichnet
- Logische Verknüpfungen werden als **Gatter** (engl. „**gate**“) bezeichnet
  - UND-Gatter, ODER-Gatter, XOR-Gatter
- Das Schaltglied zur Negation wird als **Inverter** (engl. „**inverter**“) bezeichnet
- Eine Schaltung aus mehreren (oder einem) Gattern ist ein **Schaltnetz** oder **kombinatorische Schaltung**
- Die von einem Signal maximal zu durchlaufenden Gatter werden als **Logik-Stufen** oder kurz **Stufen** eines Schaltnetzes bezeichnet
  - Man spricht z.B. von **zweistufiger Logik**
  - Inverter werden hierbei nicht gezählt

zweistufige Logik



dreistufige Logik

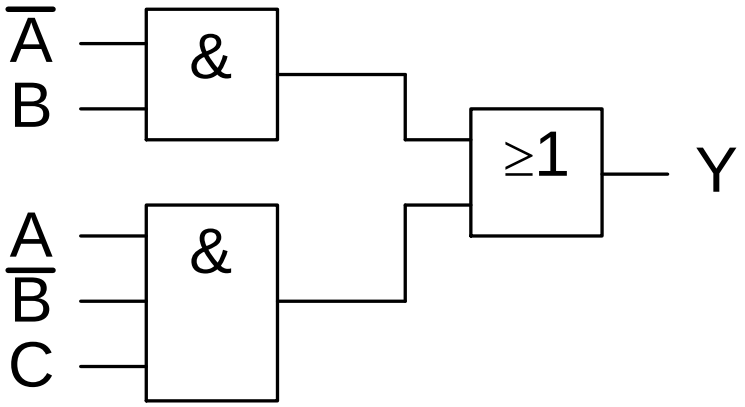


Beispiel: Zweistufige Logik

Gesucht ist die Funktionstabelle für die skizzierte Schaltung.

Wie viele Einträge hat die Funktionstabelle?

A	B	C	Y



Übung für zu Hause:

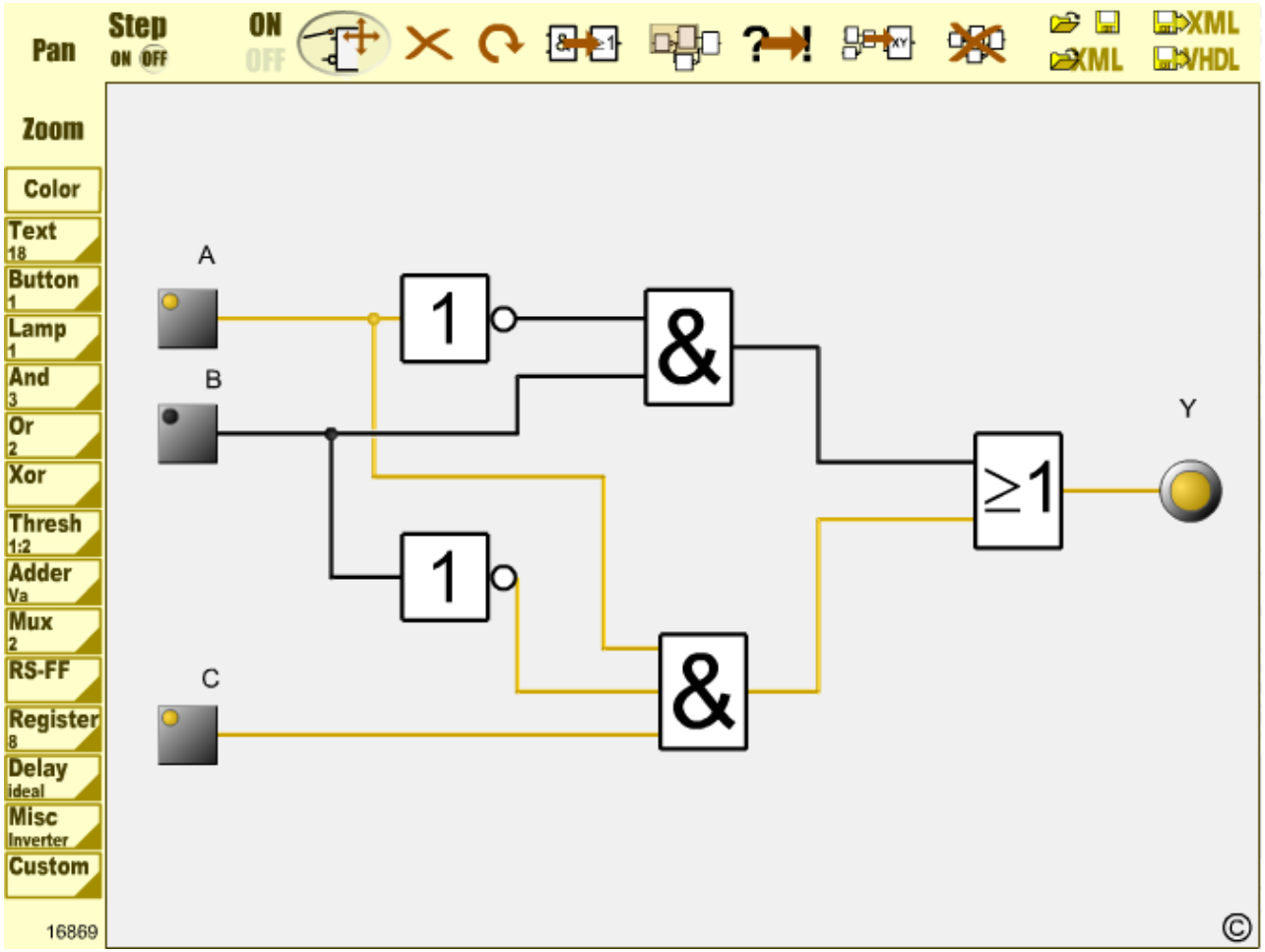
Es gibt verschiedene grafische Simulatoren für Digitalschaltungen:

- LogiFlash-Composer: **http://tiweb.hsu-hh.de/LogiFlash/index.html**
- CEDAR Logic Simulator: **http://sourceforge.net/projects/cedarlogic/**
- Logisim: **http://sourceforge.net/projects/circuit/**
- Logic Friday: Download z.B. über **http://download.cnet.com**

Erstellen Sie die Schaltung mit einem Simulator und simulieren Sie die Funktionstabelle

Die genannten Simulatoren werden zum Teil nicht mehr gepflegt. Die Industrie verwendet VHDL-Simulation (Kapitel 7).

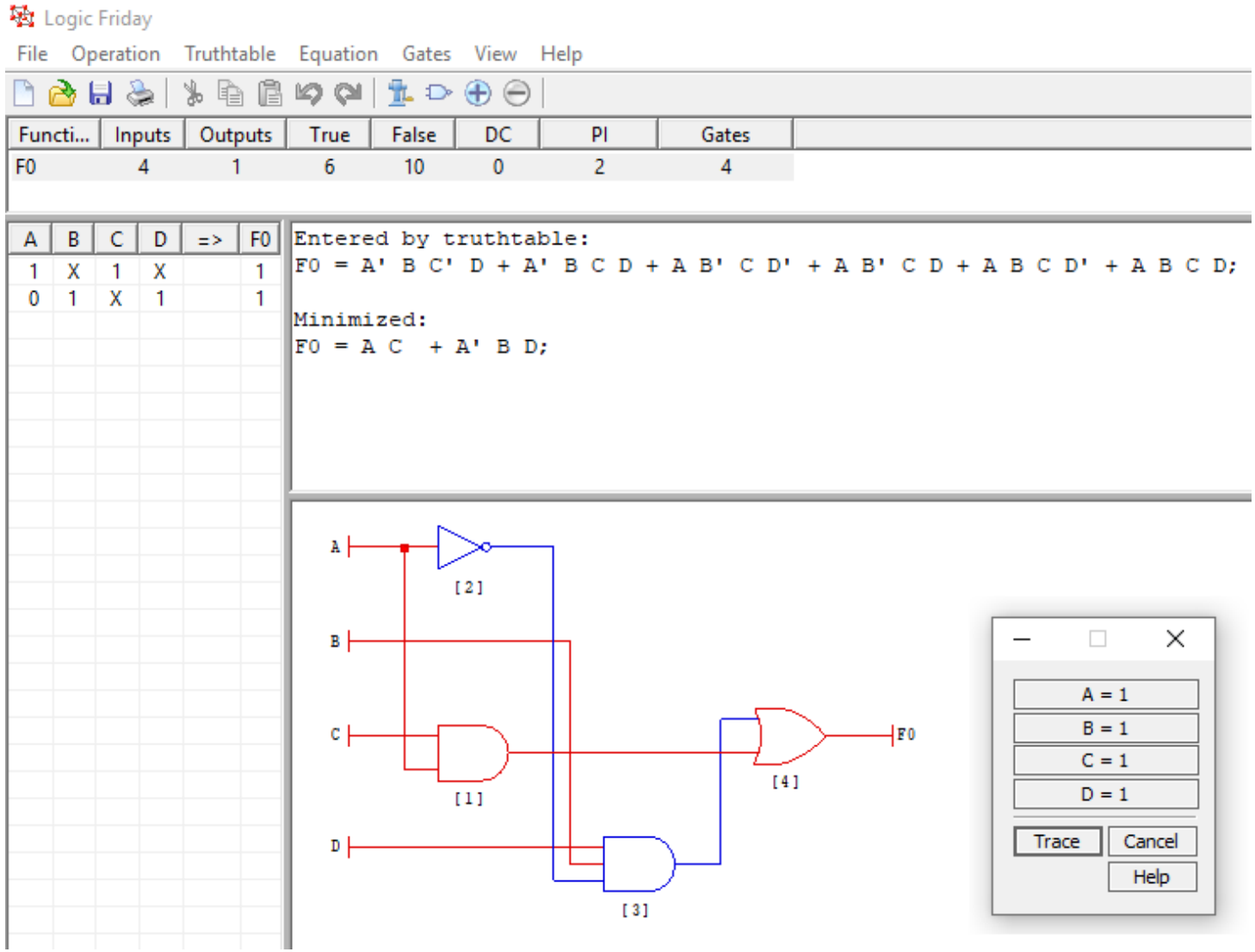
Beispiel: LogiFlash



http://tiweb.hsu-hh.de/LogiFlash/index.html

Tipps: - Der Inverter befindet sich in der Kategorie „Misc“
- Mit STRG wird ein Draht an einen bestehenden Draht angeschlossen

Beispiel: Logic Friday



4.2 Rechenregeln der Schaltalgebra

Für Schaltfunktionen gelten Rechenregeln der Schaltalgebra

- Dies ist vergleichbar zur herkömmlichen Algebra
- Die Rechenregeln gelten für UND sowie ODER, teilweise auch für XOR und XNOR
- Hier nur die wichtigsten Regeln, da sie in der Praxis wenig angewandt werden
  - ➔ Die kompletten Regeln finden sich in der Literatur

Kommutatives Gesetz (Vertauschungsgesetz)

- Variablen können in der Reihenfolge vertauscht werden

  Z.B.: A & B = B & A ; A v B = B v A

Assoziatives Gesetz (Verbindungsgesetz)

- Die Variablen können in beliebiger Reihenfolge zusammengefasst werden

  Z. B.: A & B & C = (A & B) & C = A & (B & C) = (A & C) & B

Distributives Gesetz (Verteilungsgesetz)

- Variablen können „ausmultipliziert“ und ausgeklammert werden

  Z.B.: A & (B v C) = (A & B) v (A & C)

Vorrangregeln

- Vorrang der Operationen (beginnend mit höchstem):
  - Negation
  - UND, ODER, NAND, NOR
  - Äquivalenz und Antivalenz
- Funktionen einer Ebene haben (eigentlich) gleichen Vorrang
- **Aber:** Oft wird ein Vorrang von UND gegenüber ODER verwendet
  - ➔ Schreibweise: (A & B) v (C & D) = A·B v C·D = A B v C D
  - ➔ Auch hier in der Lehrveranstaltung gilt, wenn nichts anderes genannt, ein Vorrang von UND gegenüber ODER
- In der Klausur dürfen Sie eine der üblichen Schreibweisen verwenden, wenn die Bedeutung klar wird
  - Also auch: (A & B) v (C & D) = A ∧ B v C ∧ D = A·B + C·D
- **Tipp für die Praxis:** Im Zweifelsfall Klammern setzen

De Morgansche Gesetze

Beschreibt die **Umwandlung** von UND-Verknüpfungen in ODER-Verknüpfungen und umgekehrt, mit Hilfe von Negationen

1. De Morgansche Gesetz: $\overline{A \& B} = \overline{A} \vee \overline{B}$

   Eine NAND-Verknüpfung ist gleich einer ODER-Verknüpfung der negierten Variablen

2. De Morgansche Gesetz: $\overline{A \vee B} = \overline{A} \& \overline{B}$

   Eine NOR-Verknüpfung ist gleich einer UND-Verknüpfung der negierten Variablen

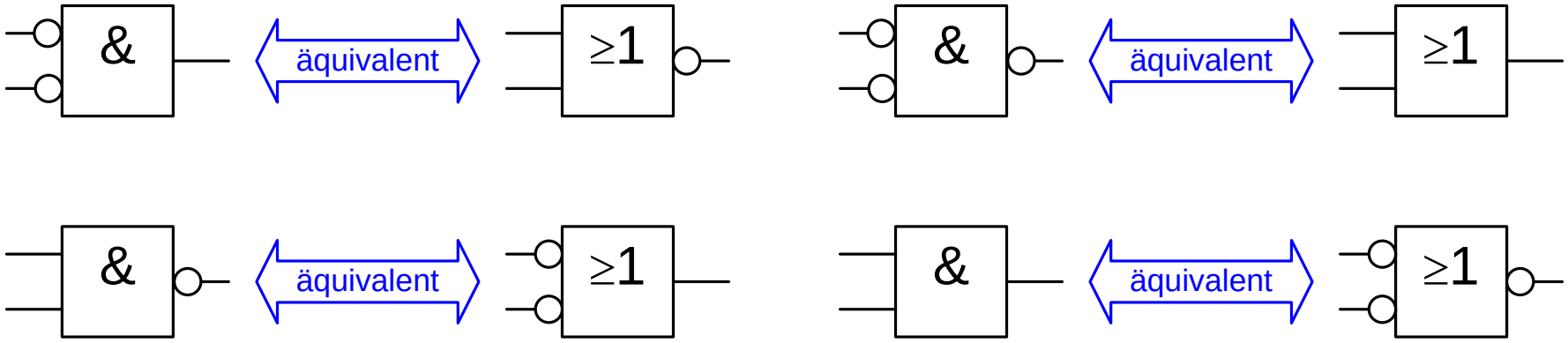
Die De Morganschen Gesetze gelten für beliebig viele Variablen

$$\overline{A \& B \& C \& D} = \overline{A} \vee \overline{B} \vee \overline{C} \vee \overline{D}$$

$$\overline{A \vee B \vee C \vee D} = \overline{A} \& \overline{B} \& \overline{C} \& \overline{D}$$

Anwendung der De Morganschen Gesetze

- Mit den De Morganschen Gesetzen lassen sich Schaltnetze einfach umformen
- Inverter an Eingängen und Ausgängen von Logikgattern werden durch Kreise dargestellt
- Die Äquivalenzen nach De Morgan erlauben die folgenden Umwandlungen

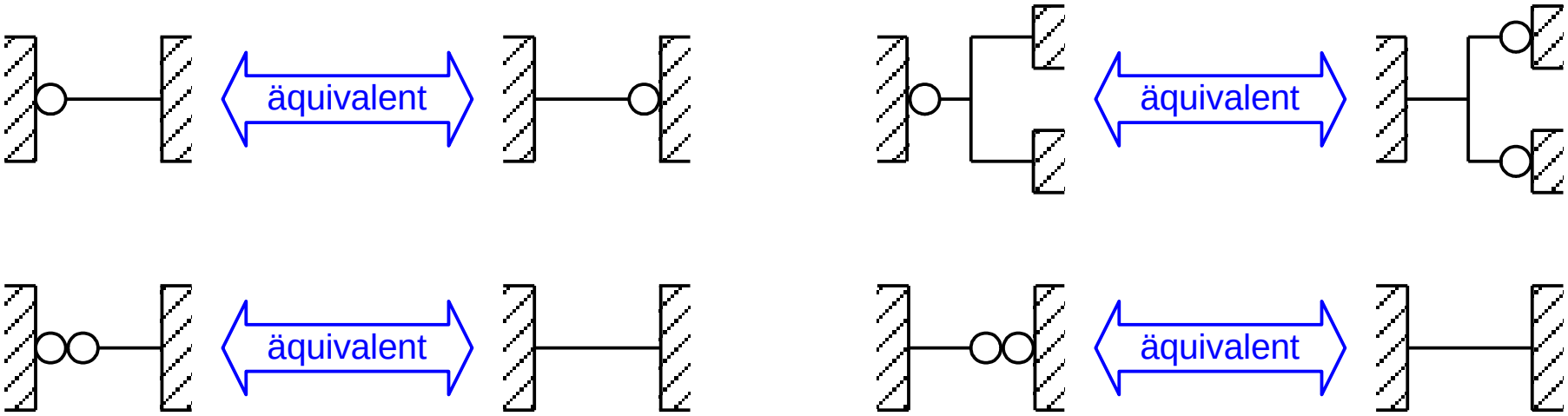


Anwendung der De Morganschen Gesetze (II)

- Inverter können an den Anfang oder das Ende eine Leitung verschoben werden

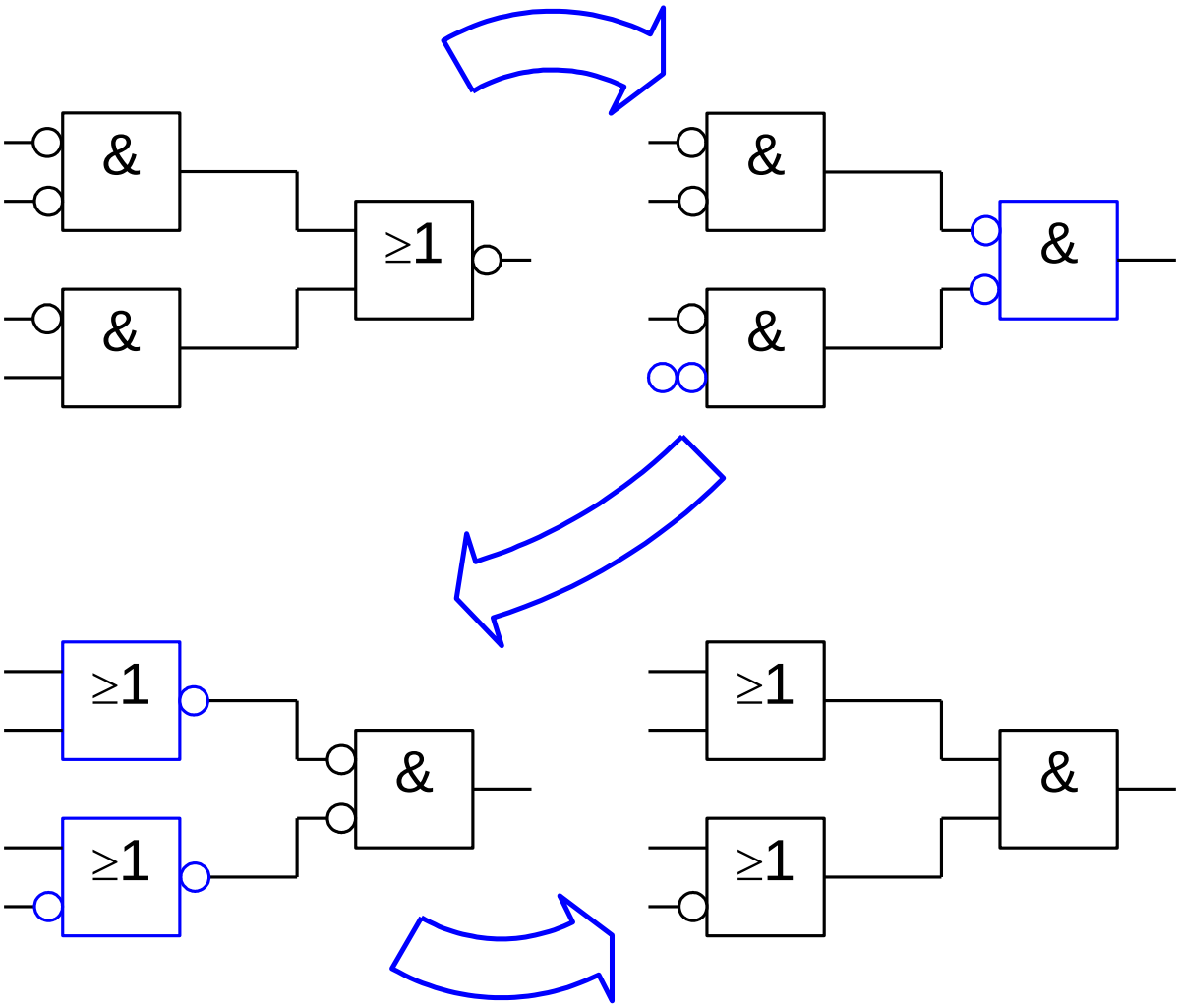
  Achtung: Wenn der Ausgang eines Logikgatters an mehrere Eingänge angeschlossen ist, müssen **alle** Eingänge invertiert werden

- Zwei Inverter heben sich auf



- Diese Umformungen werden auch als „**Bubble Pushing**“ bezeichnet
- **Ziel** ist es, vorhandene oder kostengünstige Logikgatter für ein Schaltnetz zu verwenden

Beispiel: Anwendung der De Morganschen Gesetze



Shannonsches Gesetz

- Die De Morganschen Gesetze lassen sich noch verallgemeinern
- Eine beliebige Schaltfunktion lässt sich negieren, indem
  - Alle Variablen negiert werden
  - Die Verknüpfungen UND in ODER sowie ODER in UND gewandelt werden
- Formal kann man schreiben:

$$\overline{f(A, B, C, \ldots, \overline{A}, \overline{B}, \overline{C}, \ldots, \&, \mathrm{v})} = f(\overline{A}, \overline{B}, \overline{C}, \ldots, A, B, C, \ldots, \mathrm{v}, \&)$$

4.3 Entwurfsverfahren nach Karnaugh

- Der Entwurf digitaler Schaltungen erfolgt heutzutage meist durch VHDL-Entwurf und EDA-Programme
  - EDA = Electronic Design Automation
- Kleinere Schaltungen mit bis zu 4 - 6 Eingangsvariablen können sehr einfach grafisch entworfen werden
  - Dies wird als Minimierung mit **Karnaugh-Diagramm** bezeichnet
  - Andere Bezeichnungen sind: Venn-Diagramm, KV-Diagramm (V=„Veitch“)

Weitere Verfahren

- Mit Minimierungsalgorithmen lassen sich Funktionen fast beliebiger Größe minimieren
- Am bekanntesten ist das Verfahren nach Quine-McCluskey
- Die Berechnung mit Papier und Bleistift ist möglich, es empfiehlt sich jedoch der Einsatz eines Computers
  - Einige Programme sind frei verfügbar

Normalformen

- Durch Anwendung der Regeln der Schaltalgebra lässt sich jede Schaltfunktion in eine **Normalform** umwandeln
- Eine Normalform ist eine besonders **anschauliche** und **übersichtliche** Form der Schaltfunktion
  - Es gibt zwei Normalformen, **disjunktive** und **konjunktive** Normalform
- In einer Normalform treten nur **UND**- sowie **ODER**-Verknüpfungen auf. Variablen werden **nicht negiert** und **negiert** verwendet.
- Die Normalformen benutzen Minterme und Maxterme

Minterm

- Ein Minterm ist eine **UND**-Verknüpfung die **jede Variable** genau einmal benutzt
- Die Variable kann **nicht negiert** oder **negiert** verwendet werden
- Ein Minterm ist bei genau einer Kombination der Eingangsvariablen 1
- Bei n Eingangsvariablen existieren 2^n verschiedene Minterme

Beispiele für 3 Eingangsvariable:

$$A \& B \& C; \qquad A \& \overline{B} \& C; \qquad \overline{A} \& \overline{B} \& C; \qquad \overline{A} \& \overline{B} \& \overline{C}$$

Normalformen (II)

Maxterm

- Ein Maxterm ist eine **ODER**-Verknüpfung die **jede Variable** genau einmal benutzt
- Die Variable kann **nicht negiert** oder **negiert** verwendet werden
- Ein Maxterm ist bei genau einer Kombination der Eingangsvariablen nicht 1
- Bei n Eingangsvariablen existieren 2^n verschiedene Maxterme

  Beispiele für 3 Eingangsvariable: $A \vee B \vee C$; $A \vee \overline{B} \vee C$

Beispiel: Funktionstabelle einiger Minterme und Maxterme

A	B	C	A&B&C	A&$\overline{B}$&C	$\overline{A}$&$\overline{B}$&$\overline{C}$	AvBvC	Av$\overline{B}$vC	$\overline{A}$v$\overline{B}$v$\overline{C}$
0	0	0	0	0	1	0	1	1
0	0	1	0	0	0	1	1	1
0	1	0	0	0	0	1	0	1
0	1	1	0	0	0	1	1	1
1	0	0	0	0	0	1	1	1
1	0	1	0	1	0	1	1	1
1	1	0	0	0	0	1	1	1
1	1	1	1	0	0	1	1	0

Disjunktive Normalform

- Die **disjunktive Normalform** (**DNF**) ist die **ODER**-Verknüpfung von **Mintermen**

  (Zur Erinnerung: Disjunktion meint die ODER-Verknüpfung)
- Jede Schaltfunktion kann in die disjunktive Normalform umgewandelt werden:
  - Für jede Kombination an Eingangsvariablen, die den Ausgangswert 1 ergibt, wird der Minterm gebildet
  - Die Minterme werden mit der ODER-Verknüpfung verbunden

Beispiel: Funktion laut Tabelle

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

3 Minterme:

$$\overline{A} \& B \& \overline{C}$$

$$\overline{A} \& B \& C$$

$$A \& \overline{B} \& C$$

Disjunktive Normalform:

$$Y = \overline{A}B\overline{C} \vee \overline{A}BC \vee A\overline{B}C$$

Konjunktive Normalform

- Die **konjunktive Normalform** (**KNF**) ist die **UND**-Verknüpfung von **Maxtermen**

  (Zur Erinnerung: Konjunktion meint die UND-Verknüpfung)
- Jede Schaltfunktion kann in die konjunktive Normalform umgewandelt werden:
  - Für jede Kombination an Eingangsvariablen, die den Ausgangswert 0 ergibt, wird der Maxterm gebildet
  - Die Maxterme werden mit der UND-Verknüpfung verbunden

Beispiel: Funktion laut Tabelle

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

3 Maxterme:

$$A \vee B \vee \overline{C}$$

$$\overline{A} \vee B \vee C$$

$$\overline{A} \vee \overline{B} \vee C$$

Konjunktive Normalform:

$$Y = \left(A \vee B \vee \overline{C}\right) \& \left(\overline{A} \vee B \vee C\right) \& \left(\overline{A} \vee \overline{B} \vee C\right)$$

Vergleich der Normalformen

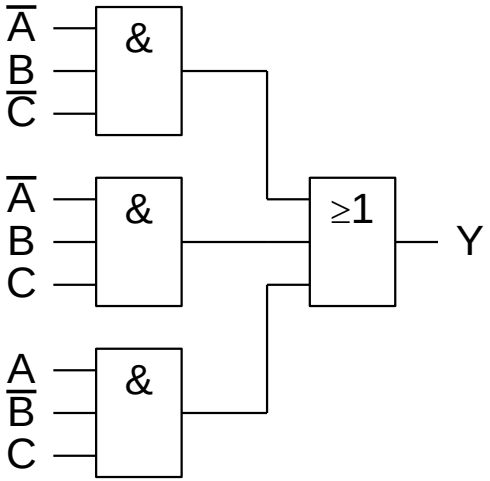
- Beide Normalformen sind gleich gut zur Schaltungsdarstellung geeignet
- Die Anzahl der Minterme bzw. Maxterme entspricht der Anzahl an Eingangskombination die eine 1 (DNF) bzw. 0 (KNF) ergeben.
  - Die disjunktive Normalform ist üblicherweise anschaulicher und wird meist verwendet
  - In der Vorlesung wird hauptsächlich die disjunktive Normalform (DNF) verwendet
  - Falls nur wenige Eingangskombinationen eine 0 ergeben, kann die konjunktive Normalform vorteilhaft sein

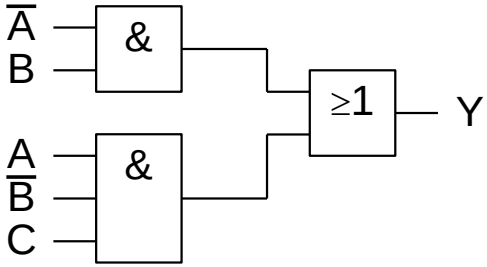
Hinweis: Die Normalformen werden manchmal auch als vollständige Normalformen oder kanonische Normalformen bezeichnet

Vereinfachung von Schaltfunktionen

- Eine **Schaltfunktion** kann entsprechend der Normalform direkt in ein **Schaltnetz** umgesetzt werden
- Oft kann die Schaltfunktion jedoch noch vereinfacht werden

  Beispiel:
  - Disjunktive Normalform: $Y = \overline{A}B\overline{C} \vee \overline{A}BC \vee A\overline{B}C$
  - Die ersten beiden Terme können zusammengefasst werden: $Y = \overline{A}B \vee A\overline{B}C$





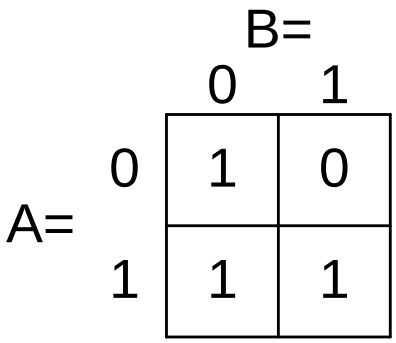
➔ Da nicht einfach ersichtlich ist, welche Terme zusammengefasst werden können, wird ein grafisches Verfahren verwendet

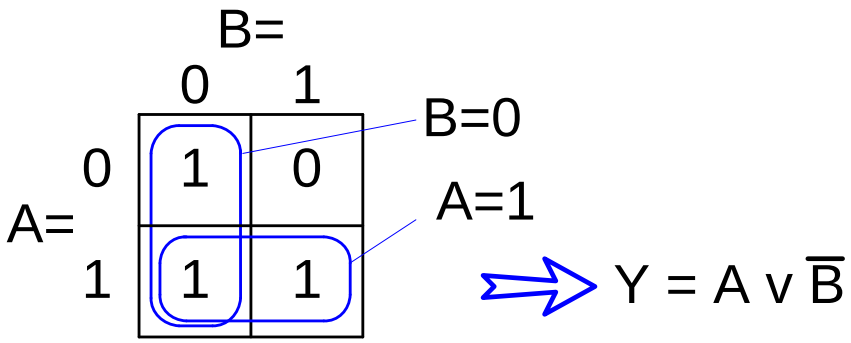
Logikminimierung mit Karnaugh-Diagramm

- In einem Karnaugh-Diagramm wird eine Funktionstabelle grafisch dargestellt
- Minterme oder Maxterme, die zusammengefasst werden können, liegen nebeneinander und können leicht identifiziert werden
  - Hier soll vorrangig das Karnaugh-Diagramm der disjunktiven Normalform mit Mintermen erläutert werden. Das Verfahren ist gleich gut für die konjunktive Normalform geeignet.
- Eine Schaltfunktion mit n Variablen wird in 2^n Feldern eingetragen
- Benachbarte Felder unterscheiden sich in nur einer Variable und können zusammengefasst werden

Karnaugh-Diagramm für zwei Variablen

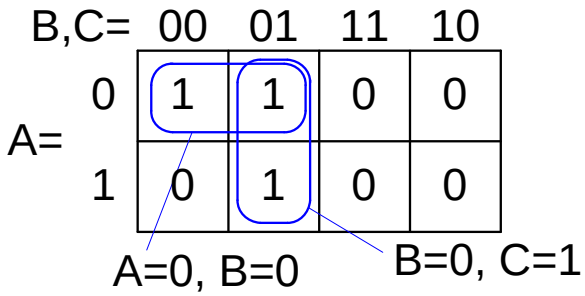
A	B	Y
0	0	1
0	1	0
1	0	1
1	1	1

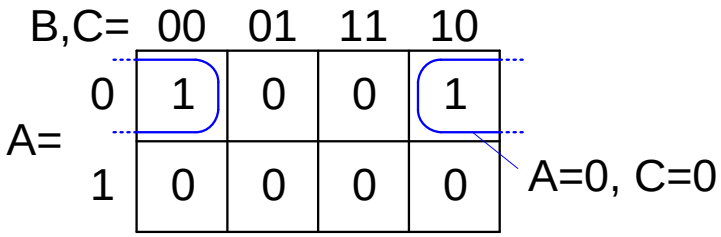


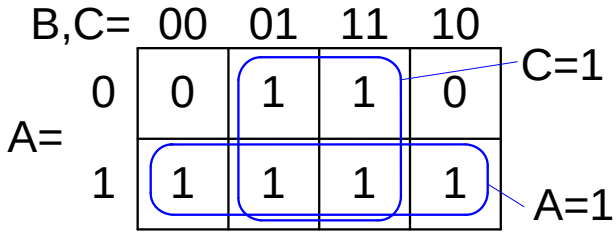


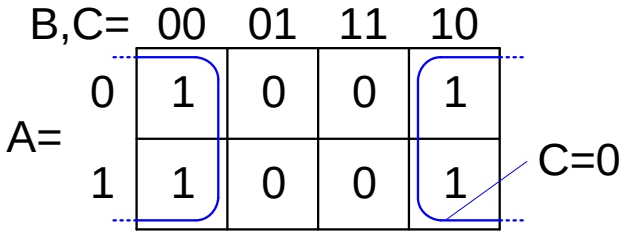
Karnaugh-Diagramm für drei Variablen

- Gegenüber dem Fall für zwei Variablen muss beachtet werden:
  - Erweiterung auf 8 Felder
  - An einer Kante werden zwei Variable angeordnet
  - Die Anordnung ist so, dass benachbarte Felder sich in nur einer Variable unterscheiden (Gray-Code)
  - In Richtung der zwei Variablen sind auch linker und rechter Rand benachbart
  - Auch Gruppen von vier Funktionswerten können zusammengefasst werden



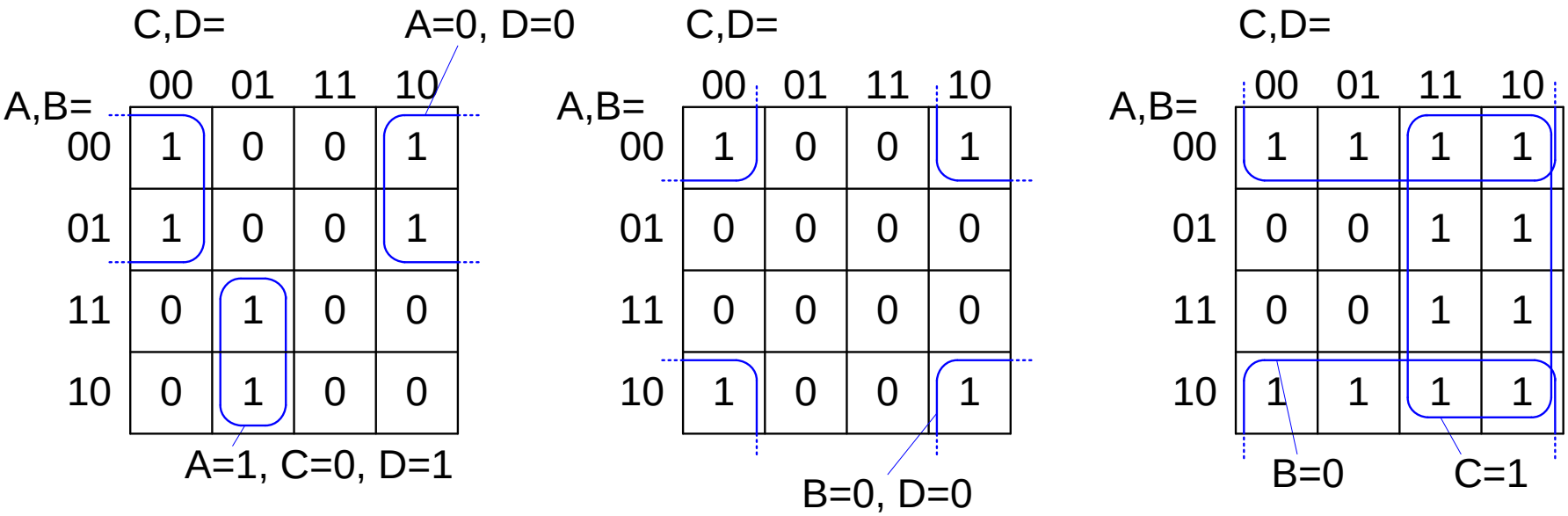






Karnaugh-Diagramm für vier Variablen

- Erweiterung auf 16 Felder
- An beiden Kanten werden zwei Variable angeordnet
- Auch Gruppen von acht Funktionswerten können zusammengefasst werden
  - Tipp: Auch die vier Ecken sind benachbart



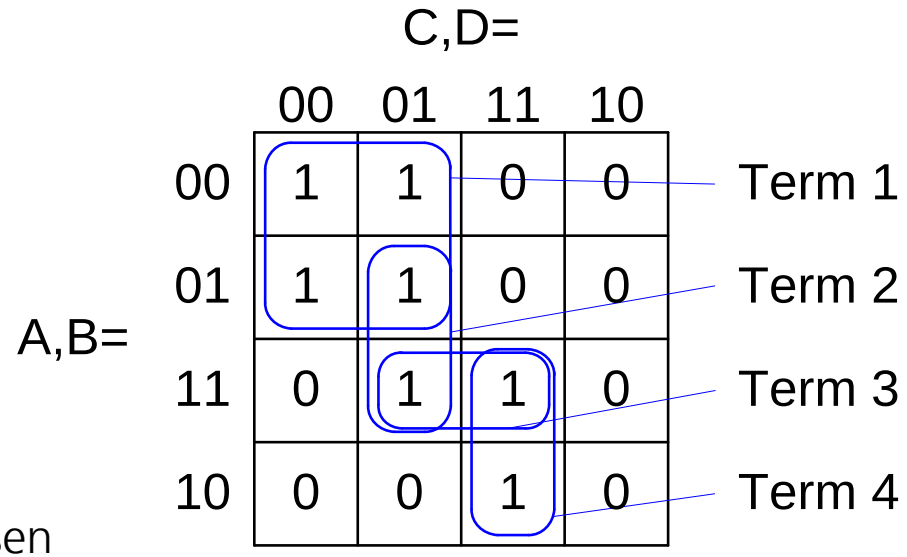
- Auch für 5 und 6 Variablen können Karnaugh-Diagramme aufgestellt werden
- Identifikation benachbarter Terme erfordert mehr Übung (trainiert aber das räumliche Vorstellungsvermögen)

Überdeckte Terme

- Mittels der Nachbarschaft in der grafischen Darstellung werden Minterme zu einfacheren Termen, den **Produkttermen**, zusammengefasst
- Die Produktterme werden durch eine ODER-Verknüpfung zusammengefasst
- Terme, die komplett von anderen Termen überdeckt sind, können möglicherweise weggelassen werden

Beispiel:

- Term 1 und Term 4 werden benötigt, da sie 1-Stellen umfassen, die von **keinem anderen Term** abgedeckt werden
- Term 2 und Term 3 umfassen nur 1-Stellen, die auch von anderen Termen abgedeckt sind
- Term 2 **oder** Term 3 können weggelassen werden
- **Achtung:** Werden Term 2 **und** Term 3 weggelassen, wäre eine 1-Stelle nicht abgedeckt Dies wäre ein **Fehler**



Formulierung der Minterme

- Aus der grafischen Darstellung müssen die Minterme abgelesen werden
  - Identifizieren Sie die Variablen, über die sich ein Term nur in einer Polarität (nicht negiert oder negiert) ausdehnt
  - Ein Minterm entspricht der UND-Verknüpfung dieser Variablen
- Zur besseren Identifikation, kann die Polarität der Variablen durch Balken eingetragen werden

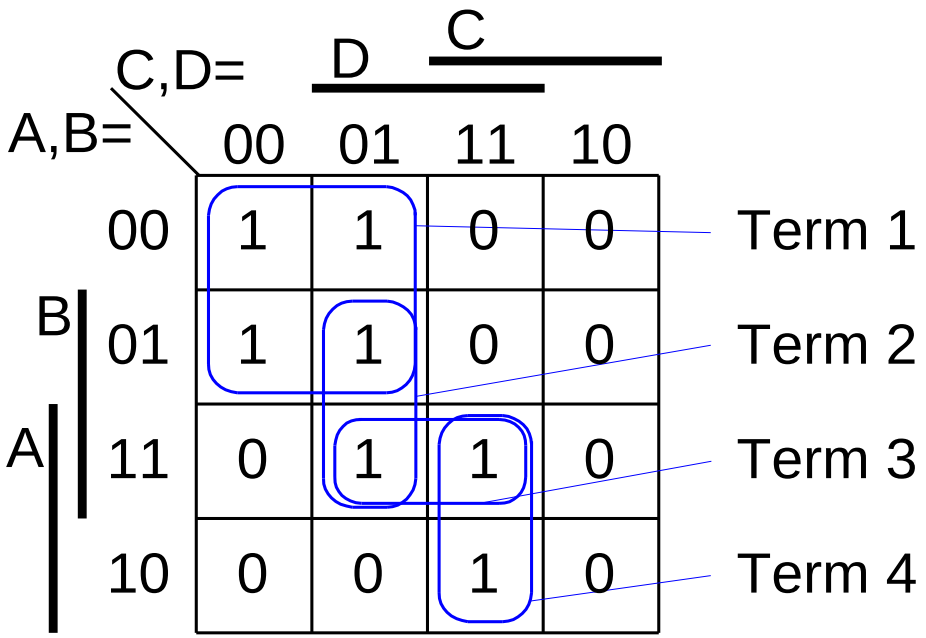
Beispiel:

Term 1: $\overline{A}\,\&\,\overline{C}$

Term 2: $B\,\&\,\overline{C}\,\&\,D$

Term 3: $A\,\&\,B\,\&\,D$

Term 4: $A\,\&\,C\,\&\,D$



„Don‘t Cares“

- Manche Schaltfunktionen sind nicht für alle Kombinationen an Eingangsvariablen definiert
- Die undefinierten Ausgangswerte werden als „don‘t care“ bezeichnet und erhalten das Symbol ‚-‘
- Bei der Minimierung werden „don‘t cares“ wie folgt benutzt:
  - Zur **Bildung** möglichst **großer Produktterme** werden „don‘t cares“ wie 1-Stellen verwendet
  - Zur **Auswahl** der **benötigten Produktterme** werden „don‘t cares“ nicht verwendet und wie 0-Stellen betrachtet

Beispiel:

- Mit Hilfe der „don‘t cares“ können vier Produktterme gebildet werden
- Term 3 wird nicht benötigt
- Die Schaltfunktion lautet:

$$Y = \overline{A}\,\overline{B}CD \vee BC\overline{D} \vee A\overline{C}$$