Aussagenlogik

Teil VII: Aussagenlogik

1. Einführung
2. Boolesche Funktionen
3. Boolesche Schaltungen



1. Einführung

- Sprachliche Aussagen
- Aussagenkombination
- Wahrheitstafeln/-tabellen



Sprachliche Aussagen

- Es geht darum sprachliche Aussagen durch logische Formeln darzustellen und ihnen den Wahrheitswert 1 (wahr) oder 0 (falsch) zuzuordnen.

Die Aussage

„Ulm liegt in Baden-Württemberg“

hat den Wahrheitswert **wahr** (oder 1)

Die Aussage

„Schwefel ist ein Metall“

hat den Wahrheitswert **falsch** (oder 0)

- Es sind nur Teilausschnitte modellierbar

So ist es nicht möglich

„Dieser Satz ist falsch“

da die Aussage **über sich selbst** redet

Aussagenkombination

- Es geht nun darum Aussagen zu kombinieren

„Wenn Ulm in Baden-Württemberg liegt und Schwefel ein Metall ist, dann können Pferde fliegen“

Diese Aussage ist **wahr**

- Ziel: Systematische Methoden um Wahrheitswerte von verknüpften Aussagen bestimmen zu können

Wahrheitstafeln/-tabellen

- Wir verwenden nun Platzhalter/Variable (A, B, C oder x, y, z) anstelle der Aussagen.

d. h. der Wahrheitswert seht noch nicht fest

- In Form von Wahrheitstafeln/-tabellen systematisch alle Möglichkeiten für Wahrheitswerte auflisten

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

Bei 3 Variablen (bzw. n Variablen) ergeben sich 8 Zeilen (bzw. 2^n Zeilen)

2. Boolesche Funktionen

- Boolesche Funktionen
- Anwendungsfelder
- Umformungsregeln für Boolesche Formeln
- Erfüllbarkeit / Tautologie
- Tautologie
- Wahrheitstafeln
- Disjunktive Normalform (DNF)
- Konjunktive Normalform (KNF)
- Resolutionskalkül
- Boolesche Schaltfunktionen










Boolesche Funktionen

- Es gibt nun mehrere Verbindungen von Aussagen (und, wenn, dann, oder, nicht)
- Diese stellen wir wie folgt dar:

wobei das ***einschließende*** **Oder** gemeint ist

A	B	(A ∨ B)
0	0	0
0	1	1
1	0	1
1	1	1

wobei die **Und-Operation** gemeint ist

A	B	(A ∧ B)
0	0	0
0	1	0
1	0	0
1	1	1

A	B	(A ⊕ B)
0	0	0
0	1	1
1	0	1
1	1	0

wobei das ***ausschließende*** **Oder** gemeint ist
(engl: exclusive-or, kurz: XOR)

Boolesche Funktionen (1)

wobei gemeint ist **aus A folgt B** bzw. **wenn A dann B**

A	B	(A → B)
0	0	1
0	1	1
1	0	0
1	1	1

A	B	(A ↔B)
0	0	1
0	1	0
1	0	0
1	1	1

wobei die **Nicht-Operation** gemeint ist

A	Ā bzw. ¬A
0	1
1	0

Bem.: Die „Umkehrung“ von A → B ist nicht die Aussage B → A.
Diese beiden Aussagen sind nicht dasselbe.
Zu A → B äquivalent ist
¬B → ¬A (kontraposition)

wobei gemeint ist
A genau dann, wenn B

Boolesche Funktionen (2)

Beispiel:

Ein Hundertjähriger wird gefragt nach seinem Geheimnis.
Der meint: „ Ich halte mich immer strickt an meine Diät“

(1) Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch

(1) $\neg B \rightarrow F$

(2) Wenn ich Fisch und Bier zu einer Mahlzeit habe, dann verzichte ich auf Eiscreme

(2) $(F \wedge B) \rightarrow \neg E$

(3) Wenn ich Eiscreme habe oder Bier meide, dann esse ich keinen Fisch

(3) $(E \vee \neg B) \rightarrow \neg F$

B	F	E	¬B	(1)	F ∧ B	¬E	(2)	E ∨ ¬B	¬F	(3)	(1) ∧ (2) ∧ (3)
0	0	0	1	0	0	1	1	1	1	1	0
0	0	1	1	0	0	0	1	1	1	1	0
0	1	0	1	1	0	1	1	1	0	0	0
0	1	1	1	1	0	0	1	1	0	0	0
1	0	0	0	1	0	1	1	0	1	1	1
1	0	1	0	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	1	0	1	1	0	0	1	0	0	0

Kompakt:
Zu jeder Mahlzeit B und nie F und E zusammen

Anwendungsfelder

- Künstliche Intelligenz/ Expertensysteme:
  - Wissensmodellierung
- Hardware/Logische Schaltungen
- Logische Programmiersprachen (PROLOG)
- Automatisches Beweisen

Umformungsregeln für Boolesche Formeln

- Distributivgesetze

  $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

  $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

  $x \wedge (y \oplus z) = (x \wedge y) \oplus (x \wedge z)$

- De Morgen

  $\neg(x \wedge y) = \neg x \vee \neg y$

  $\neg(x \vee y) = \neg x \wedge \neg y$

- Absorbtionsgesetze

  $x \wedge (x \vee y) = x$

  $x \vee (x \wedge y) = x$

Überprüfen von
$\neg(x \wedge y) = \neg x \vee \neg y$
mit Hilfe von Wahrheitstafel

x	y	$x \wedge y$	$\neg(x \wedge y)$	$\neg x$	$\neg y$	$\neg x \vee \neg y$
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

=

- Kontraposition

  $x \rightarrow y = \neg y \rightarrow x$

  $x \rightarrow y = \neg x \vee y$

  $x \leftrightarrow y = (x \rightarrow y) \wedge (y \rightarrow x) = (\neg x \vee y) \wedge (x \vee \neg y) = (x \wedge y) \vee (\neg x \wedge \neg y)$

Erfüllbarkeit / Tautologie

d. h. Wahrheitswerteverlauf von F in der Wahrheits-tabelle enthält eine 1

- Definition Erfüllbarkeit
  - Eine Boolesche Formel F (besteht aus den Aussagevariablen $A_1,\ldots A_n$) heißt **erfüllbar**, falls es eine Werte-Belegung für $A_1,\ldots,A_n$ gibt, so dass F den Wahrheitswert 1 erhält

$A_1,\ldots,A_n$	F
0 …… 0	0
.	.
.	1
.	.
1 ……1	0

F ist erfüllbar

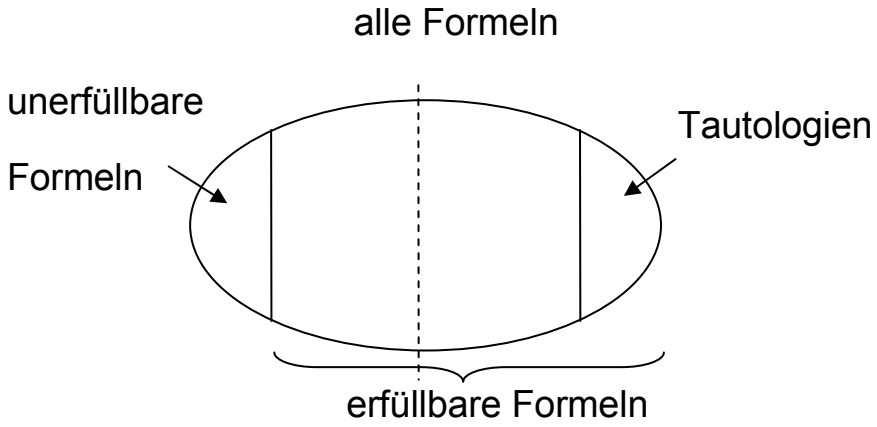
d. h. Wahrheitswerteverlauf von F besteht nur aus 1‘en

- Definition Tautologie
  - Formel F heißt **gültig** (oder Tautologie) falls für alle Werte-Belegungen für $A_1,\ldots,A_n$, die Formel F den Wahrheitswert 1 erhält.

$A_1,\ldots,A_n$	F
0 …… 0	1
.	.
.	1
.	.
1 ……1	1

F ist Tautologie

Tautologie



- Anwendung der Negation bedeutet Spiegelung an der gestrichelten Achse

Satz
F ist Tautologie ↔ ¬F ist unerfüllbar

F → ¬ F → | Erfüllbar-keitstest | → ja

→ nein → d. h. F ist Tautologie

Einige elementare Boolesche Funktionen

$\wedge$	und	Konjunktion	$\oplus$	XOR	Antivalenz
$\vee$	oder	Disjunktion	$\leftrightarrow$	genau dann wenn	Äquivalenz
$\neg$	nicht	Negation	nor	not-or	
$\rightarrow$	daraus folgt wenn…dann	Implikation	nand	not-and	

x	y	x ∧ y	x ∨ y	¬x	x→y	x ⊕ y	x ↔y	nor(x,y)	nand(x,y)
0	0	0	0	1	1	0	1	1	1
0	1	0	1	1	1	1	0	0	1
1	0	0	1	0	0	1	0	0	1
1	1	1	1	0	1	0	1	0	0

Wahrheitstafel mit n Booleschen Variablen
(auch: atomare Aussage) Besitzt 2^n Zeilen
Es gibt $2^{(2^n)}$ viele verschiedene n-stellige Boolesche Funktionen
(manche davon triviale Funktionen, z. B. Konstant)

Wahrheitstafeln

- Bisher:

  Gegeben Boolesche Formel, danach dann Wahrheitstafel aufstellen

- Jetzt:

  Gegeben Wahrheitstafel, finde dazu die Boolesche Formel

Disjunktive Normalform (DNF)

- Beispiel mit 3 Variablen

x	y	z	F	
0	0	0	1	→ (¬x ∧ ¬y ∧ ¬z)
0	0	1	1	→ (¬x ∧ ¬y ∧ z)
0	1	0	0	
0	1	1	0	
1	0	0	1	→ (x ∧ ¬y ∧ ¬z)
1	0	1	1	→ (x ∧ ¬y ∧ z)
1	1	0	1	→ (x ∧ y ∧ ¬z)
1	1	1	0	

„vollständig“= jeder Klammer-ausdruck enthält alle Variablen

⇒ (vereinfacht DNF)

- Dies ergibt die (vollständige) disjunktive Normalform:

$$F = (\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge \neg z)$$

$$= (\neg x \wedge \neg y) \vee (\neg y \wedge \neg z) \vee (x \wedge \neg y) \vee (x \wedge \neg z)$$

Konjunktive Normalform (KNF)

- Anhand der Zeilen der Wahrheitstafel mit 0.
  Wahrheitswert = 1 führt jetzt zur Negation der Variablen:

$$F = (x \vee \neg y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee \neg y \vee \neg z)$$
$$= (x \vee \neg y) \wedge (\neg y \vee \neg z)$$

Klammerausdrücke bei KNF heißen auch Klauseln

- 2 Klauseln, die sich im „Verzeichnis“ genau einer Variablen unterscheiden, können verschmelzen, wobei diese Variable dann weggelassen wird.
- Jede Boolesche Funktion besitzt eine Darstellung in DNF und KNF.

Konjunktive Normalform (KNF) (1)

- Da DNF (KNF) nur die Operatoren ∧, ∨, ¬ enthalten kann mit diesen Operatoren jede Boolsche Funktion ausgedrückt werden.

- d. h. Die Menge { ∧, ∨, ¬ } bildet eine vollständige Basis schon { ∧, ¬ } ist eine vollständige Basis,

  denn: $x \vee y = \neg(\neg x \wedge \neg y)$ (wegen de Morgan)

- Analog: { ∨, ¬ } ist vollständige Basis.

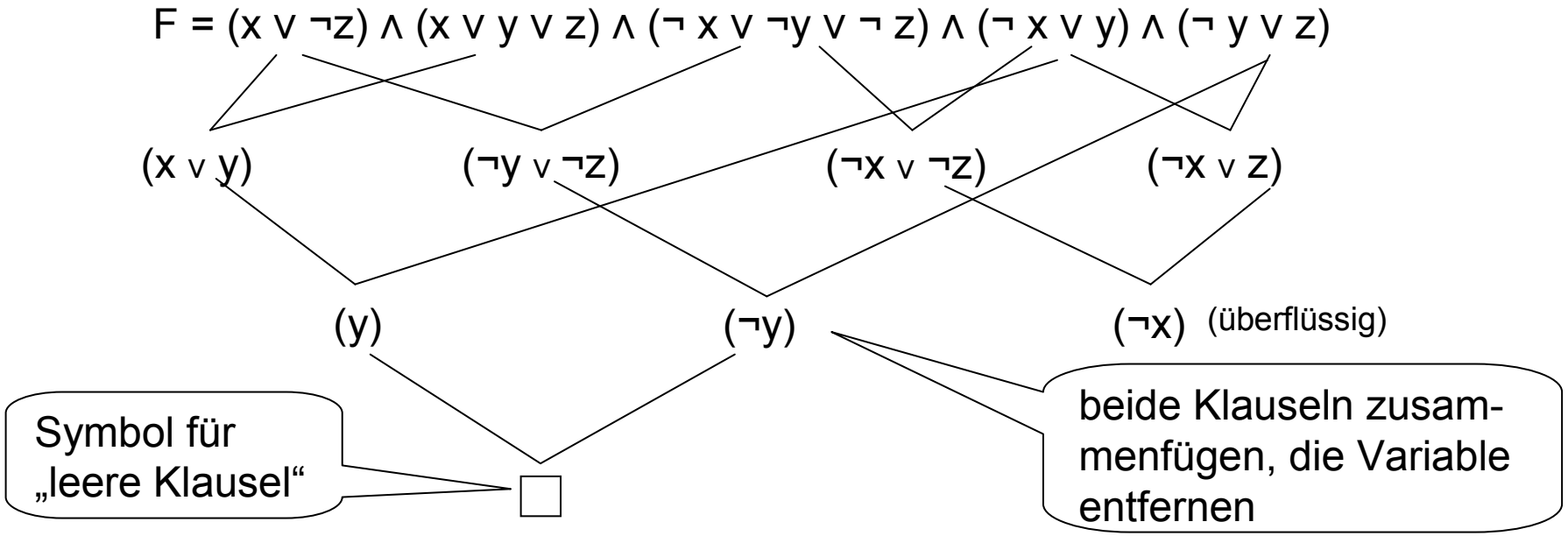
- Weitere vollständige Basen:

  { ⊕, ∧ }, { nor }, { nand }

Resolutionskalkül

- Gegeben sei Formel in KNF („Klauselformel“) (ggf. muss zunächst KNF hergestellt werden)
- Resolutionsregel:

  Wenn es bei 2 Klauseln genau eine Variable gibt, in der eine Klausel positiv, in der anderen negativ auftritt, dann darf ein Resolvent gebildet werden:



Resolutionskalkül (1)

- Satz:

  Eine Formel F in KNF ist unerfüllbar genau dann, wenn aus den Klauseln von F mittels der Resolution sich die leere Klausel ableiten lässt.

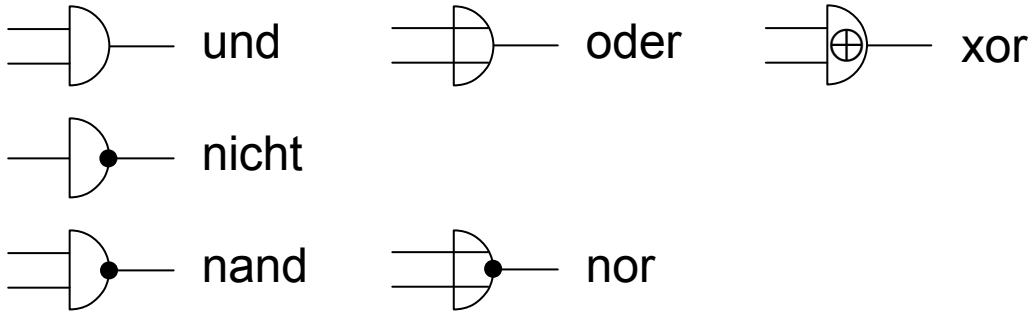
- Anwendung:

  Feststellung, ob F Tautologie ist:

  Erzeuge ¬F und forme in KNF um. Dann Resolution anwenden.

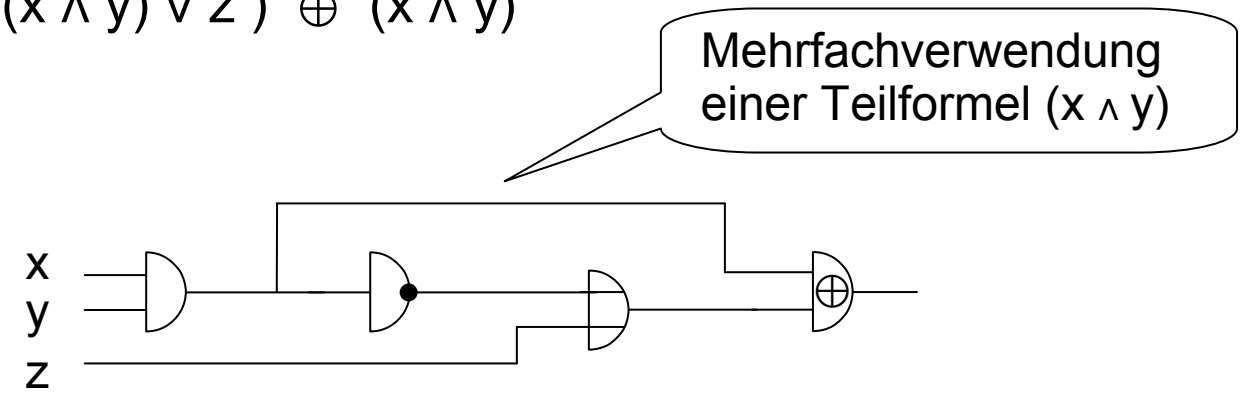
Boolesche Schaltfunktionen

- Hardware-Realisierung von Booleschen Formeln



- Aus jeder Formel kann eine Schaltung entstehen:

  $F = (\neg(x \wedge y) \vee z) \oplus (x \wedge y)$



3. Boolesche Schaltungen

- Einführung
- Boolesche Schaltungen
- Wahrheitstafel
- Disjunktive / Konjunktive Normalform (DNF / KNF)
- Flimmerschaltung
- Schaltungen mit Hilfe boolescher Basen
- XOR (Exklusives ODER)
- Halbaddierer
- Volladdierer
- Karnaugh-Diagramm










Einführung

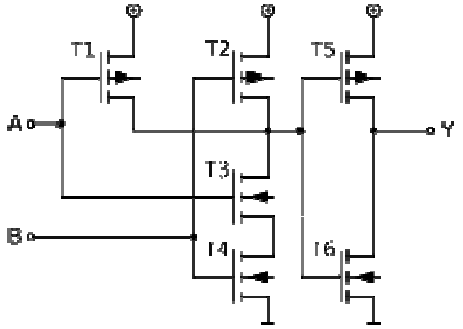
- (Schaltfunktionen, Logische Schaltungen, …)

- Boolesche Formeln oder Funktionen $F : \{ 0, 1 \}^n \rightarrow \{ 0, 1 \}$ können als Schaltungen realisiert werden.

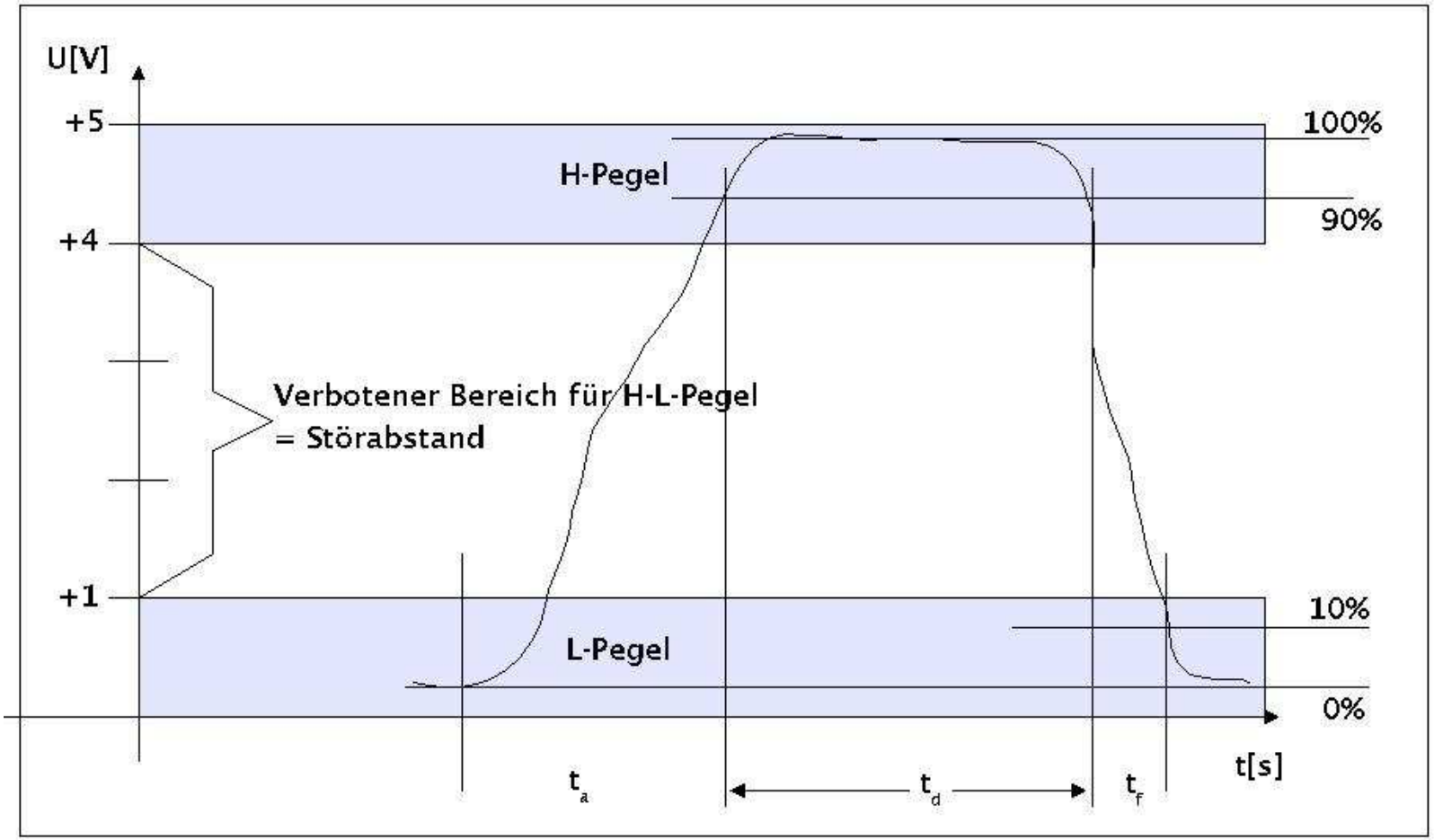
- Logische Werte 1, 0 werden umgesetzt in „Strom fliesst“, „Strom fliesst nicht“.

Einführung (1)

Schaltbild AND-Gatter:

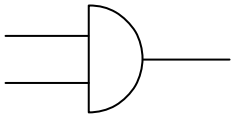


- Physikalische Umsetzung binärer Zustände („Strom fließt“ oder „Strom fließt nicht“)

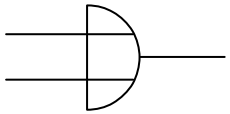


Einführung (2)

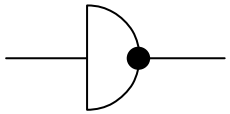
- Hardware-Realisierung von Booleschen Formeln



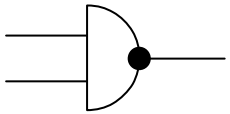
und



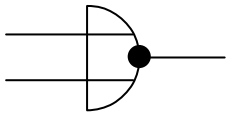
oder



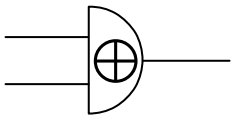
nicht



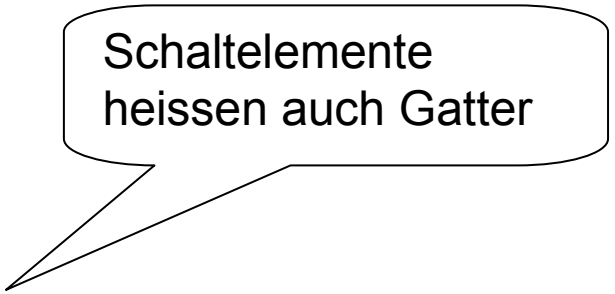
nand



nor



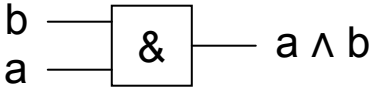
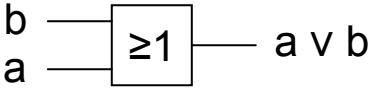
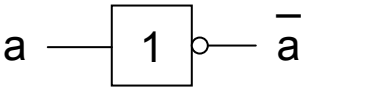
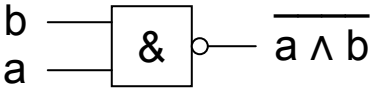
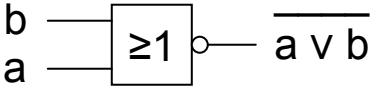
xor



(alternativ )

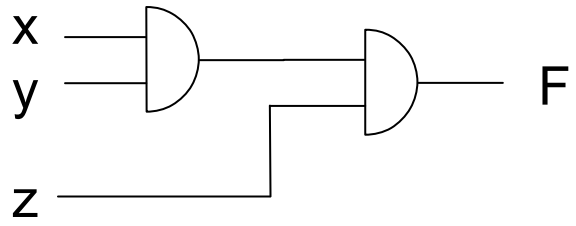
Alternative Darstellungen

➢ Einige Schaltzeichen nach DIN 40900 bzw. DIN EN 60617

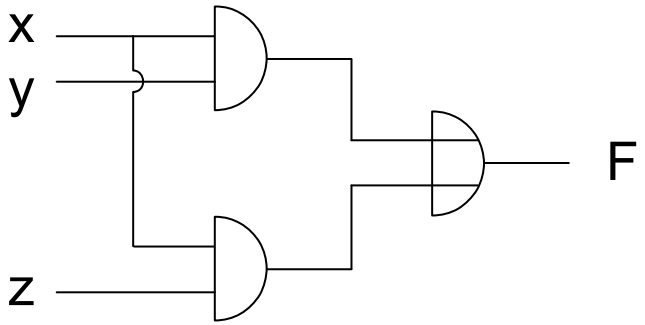
Verknüpfung	Schaltzeichen nach DIN 40900 bzw. DIN EN 60617	Benennung
$a \wedge b$		UND-Element (AND)
$a \vee b$		ODER-Element (OR)
$\overline{a}$		NICHT-Element (NOT)
$\overline{a \wedge b}$		UND-Element mit negiertem Ausgang (NAND)
$\overline{a \vee b}$		ODER-Element mit negiertem Ausgang (NOR)
		negierter Eingang
		negierter Ausgang

Boolesche Schaltungen

- $x \wedge y \wedge z \quad = \quad F \quad = \quad F(x, y, z)$



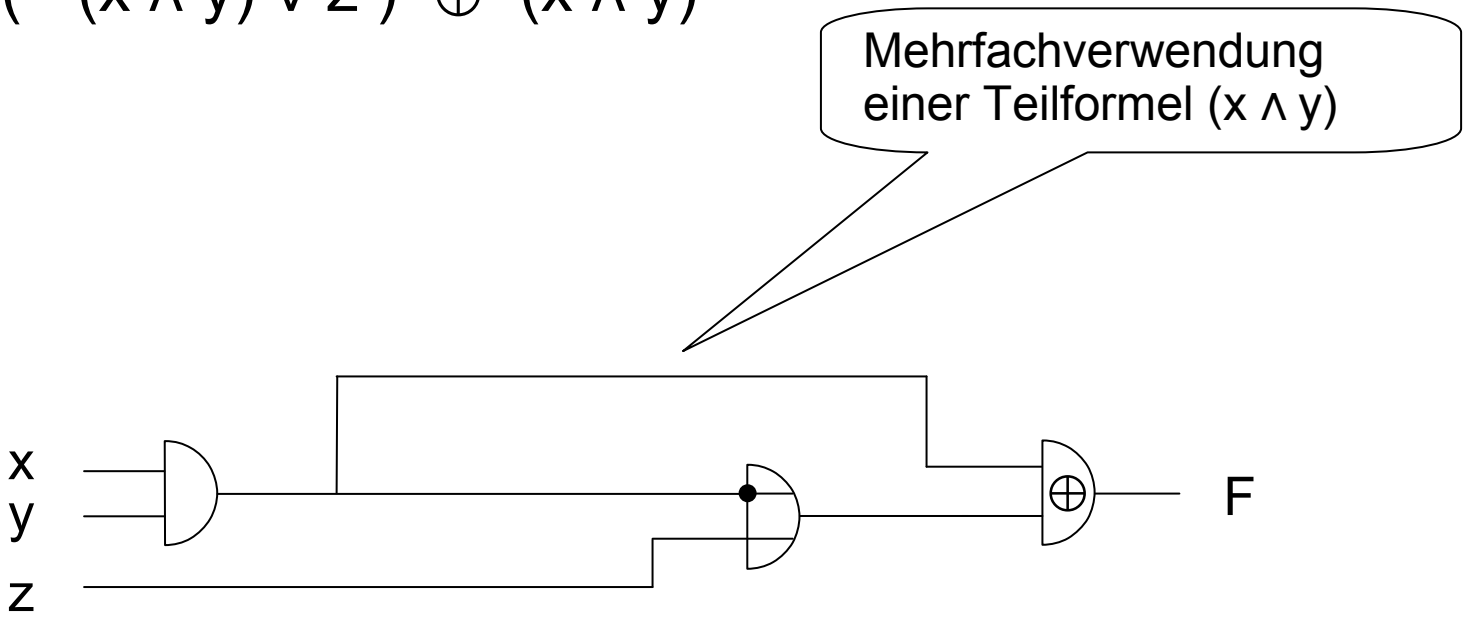
- $(x \wedge y) \vee (x \wedge z) \quad = \quad F \quad = \quad F(x, y, z)$



Mehrfachverwendung einer Variablen

Boolesche Schaltungen (1)

- $F = (\neg(x \wedge y) \vee z) \oplus (x \wedge y)$



Wahrheitstafel

➢ $F = (\neg(x \wedge y) \vee z) \oplus (x \wedge y)$

Wahrheitstafel:

x	y	z	$v = x \wedge y$	$w = \neg (x \wedge y) \vee z$	$F = v \oplus w$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	0

Disjunktive / Konjunktive Normalform (DNF / KNF)

- Disjunktive Normalform (beschreibt alle „1“ in Ergebnisspalte)

  $F = (\neg x \wedge \neg y \wedge \neg z) \vee \ldots$

  6 weitere Klammerausdrücke

- Konjunktive Normalform (beschreibt alle „0“ in Ergebnisspalte)

  $F = (\neg x \vee \neg y \vee \neg z) \wedge \ldots$

  keine weiteren Klauseln

  $= ((\neg x \vee \neg y) \vee \neg z) = ((\neg(x \wedge y) \vee \neg z) = (\neg w \vee \neg z)$

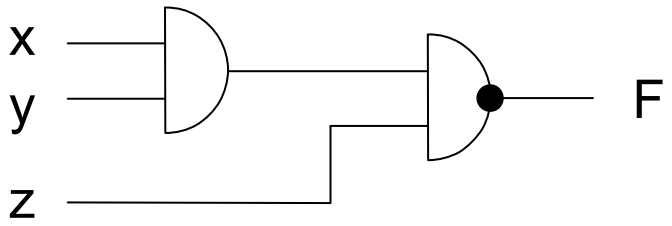
  $w = x \wedge y$

  $= \neg(w \wedge z) = \neg(x \wedge y \wedge z)$

Disjunktive / Konjunktive Normalform (1)

- Negation eines Klammerausdrucks

$$F = \neg(x \wedge y \wedge z)$$

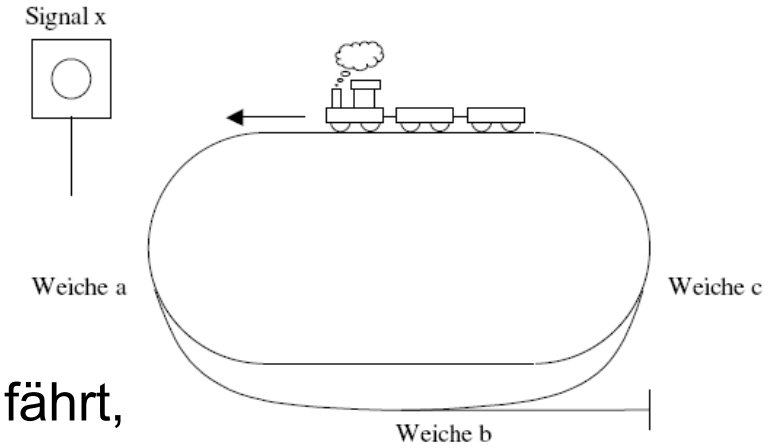


1 x UND

1 x NAND

keine Verzweigung

Disj. / Konj. Normalform (2)



- Beispiel:

  Signal soll rot (0) zeigen,
  wenn Zug über falsch gestellte Weiche fährt,
  ansonsten grün(1)

  Weiche nach innen/aussen gestellt 0/1

Weiche a	Weiche b	Weiche c	Signal x
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

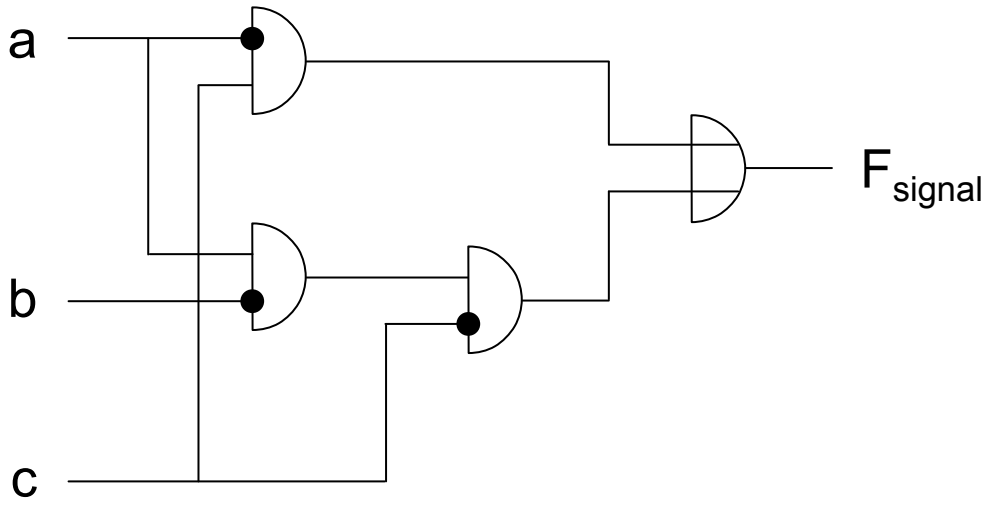
Disjunktive / Konjunktive Normalform (3)

- Schaltfunktion in DNF

$$
\begin{aligned}
F_{signal} &= [(\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c)] \vee (a \wedge \neg b \wedge c) \\
&= \left[\left(\underbrace{(\neg a \wedge c)}_{w} \wedge \neg b \right) \vee \left(\underbrace{(\neg a \wedge c)}_{w} \wedge b \right) \right] \vee (a \wedge \neg b \wedge \neg c) \\
&= (w \wedge \neg b) \vee (w \wedge b) \vee (a \wedge \neg b \wedge \neg c) \\
&= (w \wedge (\neg b \vee b)) \vee (a \wedge \neg b \wedge \neg c) \\
&= w \vee (a \wedge \neg b \wedge \neg c) \\
&= (\neg a \wedge c) \vee (a \wedge \neg b \wedge \neg c)
\end{aligned}
$$

Disjunktive / Konjunktive Normalform (4)

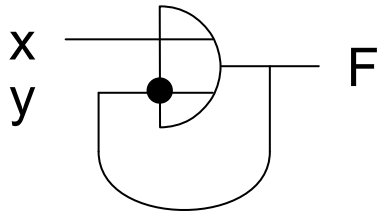
➢ Schaltung:



Flimmerschaltung

- Beispiel:

  Aus Schaltelementen kann folgende Schaltung aufgebaut werden:



x	y	F	
0	0	0	$y = 0 \rightarrow y = 1$
0	1	1	$y = 1 \rightarrow y = 0$
1	0	1	$y = 0 \rightarrow y = 1$
1	1	1	$y = 1 \rightarrow y = 0$

sog. Flimmerschaltung entspricht nicht irgendeiner Booleschen Funktion

Flimmerschaltung (1)

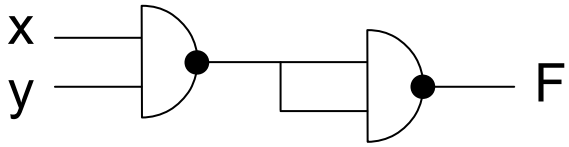
- Logische Schaltungen können als azyklische (kreisfreie) gerichtete Graphen aufgefasst werden (Kantenrichtung nicht gezeichnet)

- Logische Schaltung $\neq$ physikalischer Schaltkreis (ansonsten „kreisfreie Schaltkreise“)

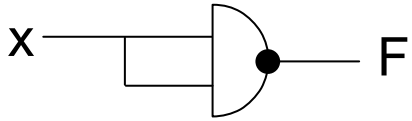
Schaltungen mit Hilfe boolescher Basen

- Mit NAND lassen sich alle Booleschen Funktionen realisieren
  ({ nand } ist vollständige Basis)

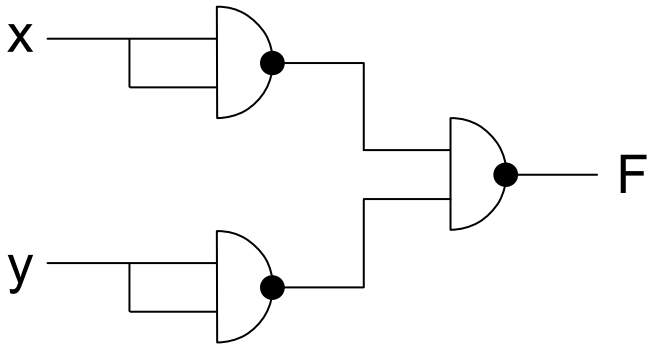
- Realisation von UND, NICHT, ODER:



$x \wedge y$



$\neg x$



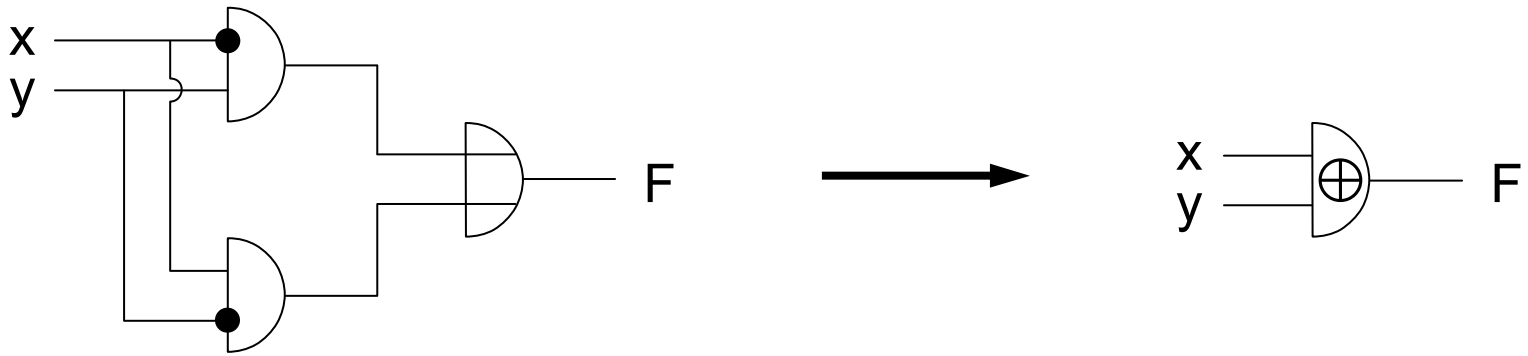
$x \vee y$

$\Rightarrow$ es genügt 1 Typ von Schaltelement

Analog: NOR genügt.

XOR (Exklusives ODER)

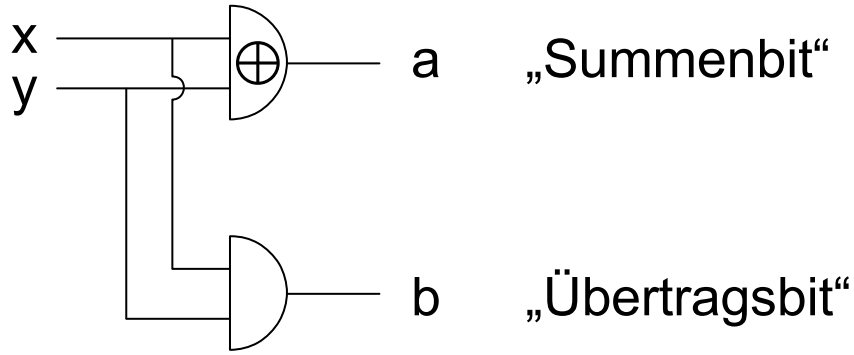
- XOR (Ausschließendes ODER / Exclusive OR)



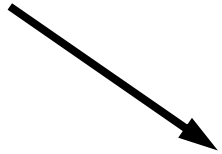
x	y	(¬x ∧ y)	(x ∧ ¬y)	F
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

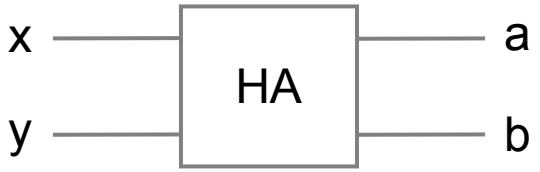
Halbaddierer

- Halbaddierer zweier Bits



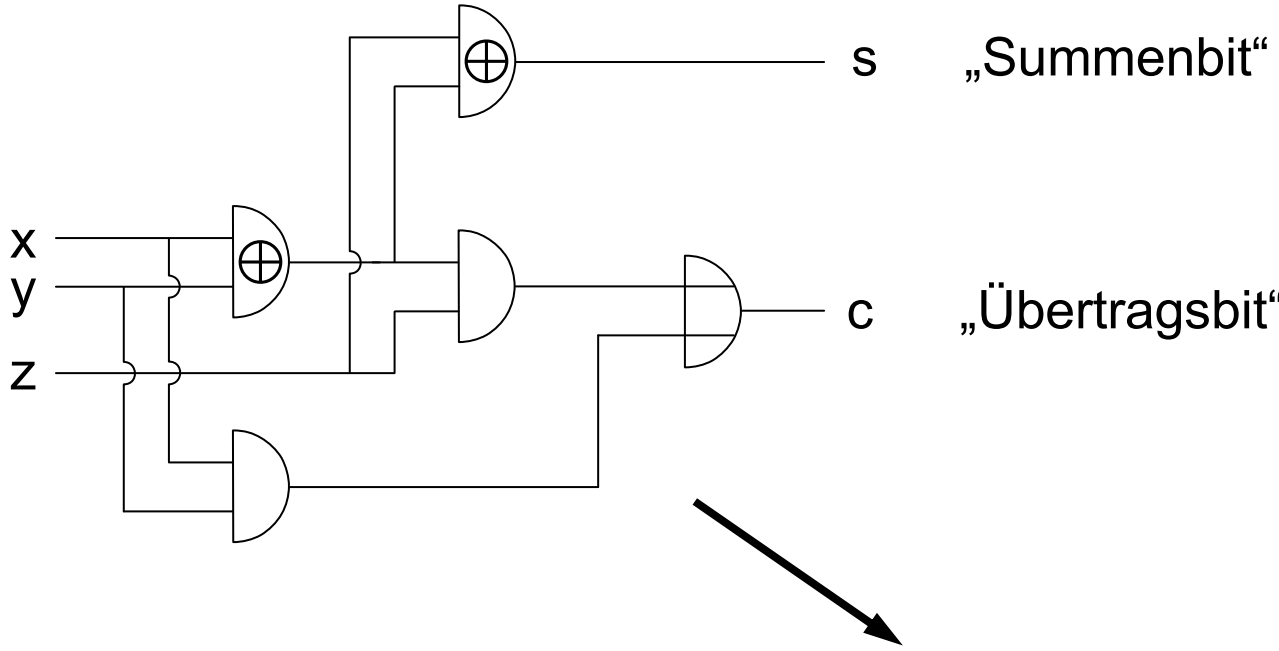
x	y	a = x ⊕ y	b = x ∧ y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

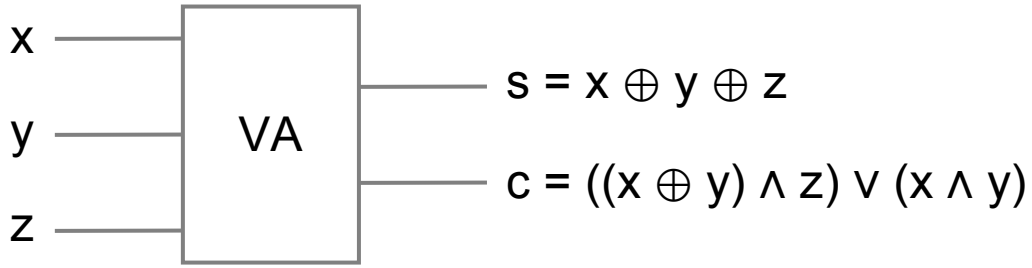




Volladdierer

- Volladdierer für 2 Bits und Übertrag aus niedrigerer Stelle





Volladdierer (1)

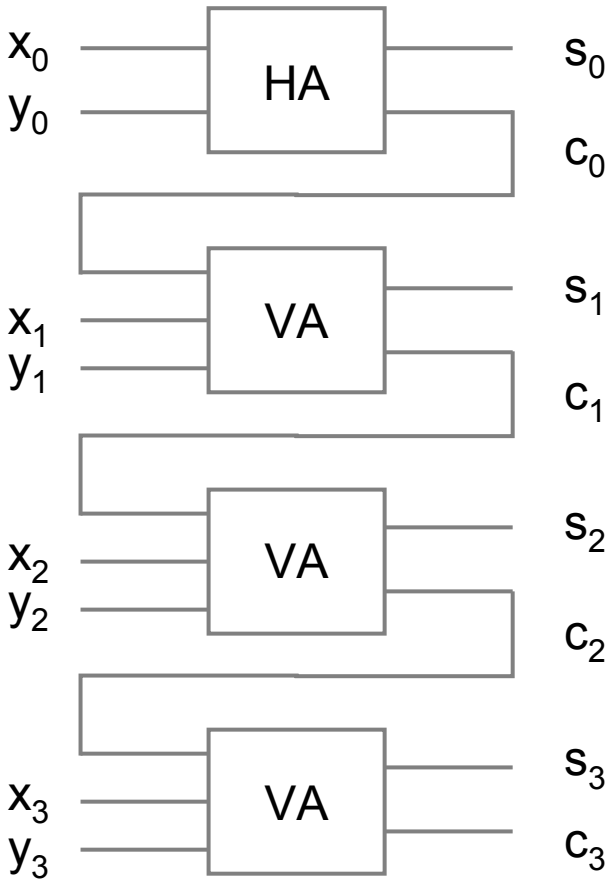
x y z	$x \oplus y$	$s = x \oplus y \oplus z$	$x \wedge y$	$(x \oplus y) \wedge z$	$c = ((x \oplus y) \wedge z) \vee (x \wedge y)$
0 0 0	0	0	0	0	0
0 0 1	0	1	0	0	0
0 1 0	1	1	0	0	0
0 1 1	1	0	0	1	1
1 0 0	1	1	0	0	0
1 0 1	1	0	0	1	1
1 1 0	0	0	1	0	1
1 1 1	0	1	1	0	1

Volladdierer (2)

- Addierwerk für zwei Binärzahlen mit 4 Bits

$x_3\ x_2\ x_1\ x_0$

$y_3\ y_2\ y_1\ y_0$



Ergebnis:
c3 s3 s2 s1 s0

Vereinfachung von Schaltungen mittels Karnaugh-Diagramm

- Zwei Grundideen:
  1. Umstellung der Wahrheitstafel in Rechteckform (Variablen horizontal und vertikal angeordnet)
  2. Suche nach Blöcken von 1 statt einzelnen 1

- DNF: $(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4) \vee (x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)$

  $= (\neg x_1 \wedge \neg x_3 \wedge x_4) \vee (x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)$

		x_1	0	0	1	1
		x_2	0	1	0	1
x_3	x_4					
0	0		0	0	0	0
0	1		1	1	0	0
1	0		0	0	1	0
1	1		0	0	0	0

Konstruktion von Karnaugh-Diagrammen (generell)

- Werte von Variablen werden so angeordnet, dass benachbarte Änderungen nur in 1 Bit bestehen

		x_1	0	0	**1**	**1**
		x_2	0	1	**1**	**0**
x_3	x_4					
0	0		0	0	0	0
0	1		1	1	0	0
1	**1**		0	0	0	0
1	**0**		0	0	0	1

Konstruktion von Karnaugh-Diagrammen (generell) (1)

- Auch bei mehr als vier Variablen kann die Wahrheitstafel so aufgebaut werden, dass sich benachbarte Kombinationen auf beiden Achsen nur um 1 Bit unterscheiden.

- Beispiel, fünf Variablen:

x_3 x_4 x_5	x_1 0 0 1 1
	x_2 0 1 1 0
0 0 0	
0 0 1	
0 1 1	
0 1 0	
1 1 0	
1 1 1	
1 0 1	
1 0 0	

Graycode

Konstruktion von Karnaugh-Diagrammen (generell) (2)

- Blöcke (Rechtecke) von 1 haben Größe von 2er Potenzen, dürfen überlappen und dürfen zyklisch („außen herum“) geschlossen sein

		x_1	0	0	1	1
		x_2	0	1	1	0
x_3	x_4					
0	0		0	0	0	0
0	1		1	0	0	1
1	1		1	0	1	1
1	0		0	0	0	0

1 x 4er Block
1 x 2er Block

Konstruktion von Karnaugh-Diagrammen (generell) (3)

- Für jeden Block identifiziert man die Variablen, die dort konstant sind. Die anderen entfallen.

x_3	x_4 \ x_1	0	0	1	1
	x_2	0	1	1	0
0	0	0	0	0	0
0	1	1	0	0	1
1	1	1	0	1	1
1	0	0	0	0	0

4er Block:

$x_1 = 0, x_1 = 1$	kommt vor	$\Rightarrow$	x_1 entfällt
$x_2 = 0$	konstant	$\Rightarrow$	$\neg x_2$ beschreibt Block
$x_3 = 0, x_3 = 1$	kommt vor	$\Rightarrow$	x_3 entfällt
$x_4 = 1$	konstant	$\Rightarrow$	x_4 beschreibt Block

$F_{\text{4er Block}} = \neg x_2 \wedge x_4$

2er Block:

$x_1 = 1$	konstant	$\Rightarrow$	x_1 beschreibt Block
$x_2 = 0, x_2 = 1$	kommt vor	$\Rightarrow$	x_2 entfällt
$x_3 = 1$	konstant	$\Rightarrow$	x_3 beschreibt Block
$x_4 = 1$	konstant	$\Rightarrow$	x_4 beschreibt Block

$F_{\text{2er Block}} = x_1 \wedge x_3 \wedge x_4$

Konstruktion von Karnaugh-Diagrammen (generell) (4)

- DNF:

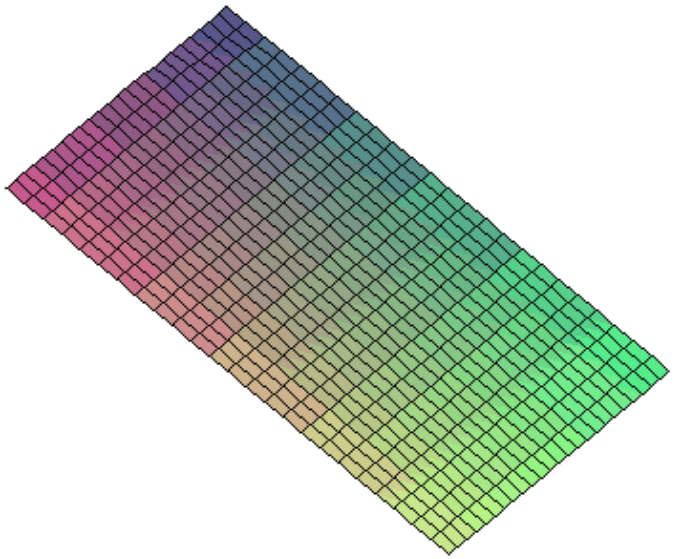
$\Rightarrow \quad F = F_{\text{4er Block}} \vee F_{\text{2er Block}} = (\neg x_2 \wedge x_4) \vee (x_1 \wedge x_3 \wedge x_4)$

Konstruktion von Karnaugh-Diagrammen (generell) (5)

➢ Blockbildung über 2 Ränder möglich!

		x_1	0	0	1	1
		x_2	0	1	1	0
x_3	x_4					
0	0		1	0	0	1
0	1		0	0	0	0
1	1		0	0	0	0
1	0		1	0	0	1

$$F = \neg x_2 \wedge \neg x_4$$



Anderes Beispiel, Torus-Animation:

http://math.arizona.edu/~ura/013/bethard.steven/torus.mov

oder

http://math.arizona.edu/~ura/013/bethard.steven/torus.avi