

Digitaltechnik Grundlagen (dtbasics)

Bit manipulationen - not, and, or, xor

- Da die binär Werte 0 und 1 die Kernwerte sind, wie Computer Daten kodieren, speichern und manipulieren, hat die Boolesche Algebra eine gewisse Bedeutung
- Die Boolesche Algebra definiert Operationen, die mit Werten von 0 und 1 arbeiten, z.B.

	NOT	AND	OR	XOR (excl. or)																																																			
Funktions gleichung	$y = \overline{x1}$	$y = x1 \wedge x2$	$y = x1 \vee x2$	$y = x1 \oplus x2$																																																			
C bit-level	y= ~x1;	y= x1 & x2;	y= x1 x2;	y = x1 ^ x2;																																																			
Wahrheitstabelle	<table><tr><th>x_1</th><th>y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x_1	y	0	1	1	0	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_2	x_1	y	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_2	x_1	y	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x_2	x_1	y	0	0	0	0	1	1	1	0	1	1	1	0
x_1	y																																																						
0	1																																																						
1	0																																																						
x_2	x_1	y																																																					
0	0	0																																																					
0	1	0																																																					
1	0	0																																																					
1	1	1																																																					
x_2	x_1	y																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	1																																																					
x_2	x_1	y																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	0																																																					

Bit manipulationen - not, and, or, xor - logische Operationen

- Logical operations = z.B: Verknüpfung von Bedingungen in if-abfragen

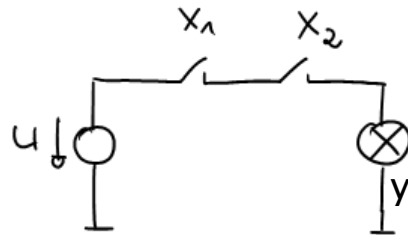
	Not \overline{x}	And $x \wedge y$	Or $x \vee y$	Xor (excl. or) $x \oplus y$																																																			
C logical (0=false; 1=true)	z= !x;	z= x && y;	z= x y;																																																				
Truth table describing the function	<table><tr><th>x_1</th><th>y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x_1	y	0	1	1	0	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_2	x_1	y	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_2	x_1	y	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x_2	x_1	y	0	0	0	0	1	1	1	0	1	1	1	0
	x_1	y																																																					
	0	1																																																					
	1	0																																																					
	x_2	x_1	y																																																				
0	0	0																																																					
0	1	0																																																					
1	0	0																																																					
1	1	1																																																					
x_2	x_1	y																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	1																																																					
x_2	x_1	y																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	0																																																					

Boolsche Algebra - Motivation

- Mit den Beziehungen **UND/AND** (\wedge), **ODER/OR** (\vee) und **NICHT/NOT** (\neg) lassen sich **ALLE BELIEBIGEN** Funktionen von zweiwertigen Variablen ausdrücken.
- Unter diesen Bedingungen spricht man von einer **Algebra**
- Diese Algebra wird auch als **Schaltalgebra** bezeichnet, die dazugehörigen Funktionen als **Schaltfunktionen**
 - Der Begriff Schaltalgebra bzw. Schaltfunktionen kommt von der ersten technischen Anwendung dieser zweiwertigen Boole'schen Algebra
 - 1938 wendete Claude E. Shannon die Algebra auf Serien- und Parallelschaltungen von Schaltern und Relais an
 - 0 bezeichnet einen offenen Schalter, 1 Bezeichnet einen geschlossenen Schalter
 - Sie ist ein Spezialfall einer **Booleschen Algebra** mit **Booleschen Funktionen**. Häufig werden die Begriffe aber auch synonym verwendet

Boolsche Algebra - Schaltfunktion - Beispiel AND

- Reihenschaltung von zwei Schaltern



x_2	x_1	y
0	0	0
0	1	0
1	0	0
1	1	1

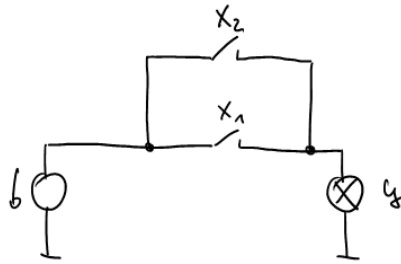
- Die Lampe brennt, wenn der erste Schalter geschlossen ist ($x_1 = 1$) UND wenn der zweite Schalter geschlossen ist ($x_2 = 1$)
- Man bezeichnet dies als UND (AND) - Verknüpfung und schreibt:

$$y = x_1 \wedge x_2$$

- Diese Verknüpfung ist eine **Schaltfunktion**. Sie wird **durch eine Wahrheitstabelle vollständig beschrieben**

Boolsche Algebra - Schaltfunktion - Beispiel OR

- Reihenschaltung von zwei Schaltern



x_2	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1

- Die Lampe brennt, wenn der erste Schalter geschlossen ist ($x_1 = 1$) ODER wenn der zweite Schalter geschlossen ist ($x_2 = 1$)
- Man bezeichnet dies als ODER (OR) - Verknüpfung und schreibt:

$$y = x_1 \vee x_2$$

Boolsche Algebra - Schaltfunktion - Beispiel NOT

- Kurzschluss der Lampe über einen Schalter



x_1	y
0	1
1	0

- Die Lampe brennt, wenn der Schalter geöffnet ist ($x_1 = 0$).
- Man bezeichnet dies als Negation (NOT) und schreibt:

$$y = \overline{x_1}$$

Boolesche Algebra - Unterschiedliche Schreibweisen

- Oft Verwendung anderer Operatorsymbole:
 - and/und: statt \wedge auch $*$ oder \cdot („mal“ oder Punkt) oder $\&\&$
 - or/oder: statt \vee auch $+$ oder \parallel
 - not/nicht: statt \neg auch $-$ („Überstrich“) oder $'$ („Strich“), oder $!$
 - Beispiel: es ist also: $\neg x$ dasselbe wie x' oder \bar{x} oder $!x$

Boolsche Algebra - Verkürzte Schreibweise

- Obwohl AND und OR der Booleschen Algebra auf das arithmetische MAL und PLUS nicht vollständig übertragbar sind, wird häufig eine verkürzte Schreibweise verwendet:

$$a \wedge b \rightarrow ab$$

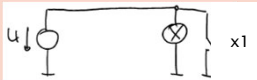
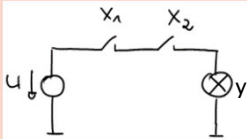
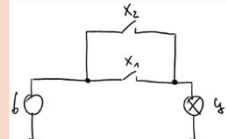
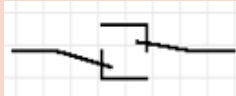

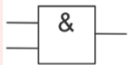
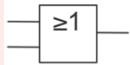
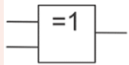
- Dies erleichtert das Lesen / Schreiben längerer Boolescher Ausdrücke enorm
- Beachten Sie aber stets den Unterschied zu den arithmetischen Operationen Addition und Multiplikation!

Boolsche Algebra - Rechenregeln

	\wedge	\vee
Kommutativgesetz	$a \wedge b = b \wedge a$	$a \vee b = b \vee a$
Assoziativgesetz	$(a \wedge b) \wedge c = a \wedge (b \wedge c)$	$(a \vee b) \vee c = a \vee (b \vee c)$
Distributivgesetz	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
Idempotenz	$a \wedge a = a$	$a \vee a = a$
Neutralität	$a \wedge 1 = a$	$a \vee 0 = a$
Extremalgesetz	$a \wedge 0 = 0$	$a \vee 1 = 1$
Komplement	$a \wedge \bar{a} = 0$	$a \vee \bar{a} = 1$
Absorptionsgesetz	$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$
de Morgansche Regeln	$\overline{a \wedge b} = \bar{a} \vee \bar{b}$	$\overline{a \vee b} = \bar{a} \wedge \bar{b}$

- Bindungsregeln: *not* vor *and* vor *or*

Realisierung Schaltfunktionen mit Gattern - Zusammenfassung

	NOT	AND	OR	XOR (excl. or)																																																			
Prinzipschaltung																																																							
Funktionsgleichung	$y = \overline{x_1}$	$y = x_1 \wedge x_2$	$y = x_1 \vee x_2$	$y = x_1 \oplus x_2$																																																			
Wahrheitstabelle	<table><tr><th>x_1</th><th>y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x_1	y	0	1	1	0	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_2	x_1	y	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_2	x_1	y	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>x_2</th><th>x_1</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x_2	x_1	y	0	0	0	0	1	1	1	0	1	1	1	0
x_1	y																																																						
0	1																																																						
1	0																																																						
x_2	x_1	y																																																					
0	0	0																																																					
0	1	0																																																					
1	0	0																																																					
1	1	1																																																					
x_2	x_1	y																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	1																																																					
x_2	x_1	y																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	0																																																					
Schaltzeichen																																																							

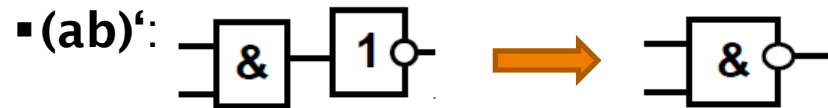
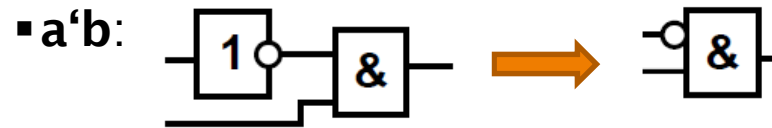
Realisierung Schaltfunktionen mit Gattern - Übersicht Schaltzeichen

Logiksymbole der Boole'schen
Algebra nach
DIN EN 60617 (links) und
ANSI/IEEE Std 91a-1991 (rechts)

	DIN- Symbole	US- Symbole
Eingangs- / Ausgangs- Verstärker		
NOT / Inverter		
UND and		
ODER or		
NAND		
NOR		
XOR		
XNOR		

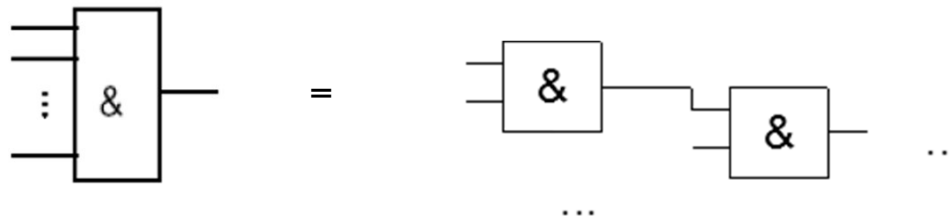
Realisierung Schaltfunktionen mit Gattern - Übersicht Schaltzeichen -erweiterte Darstellung 1/2 Negationskreis

- Zwecks kompakter Schreibweise wird die Darstellung so erweitert, dass invertierende Ein- oder Ausgänge durch einen Kreis gezeichnet werden, z.B.:

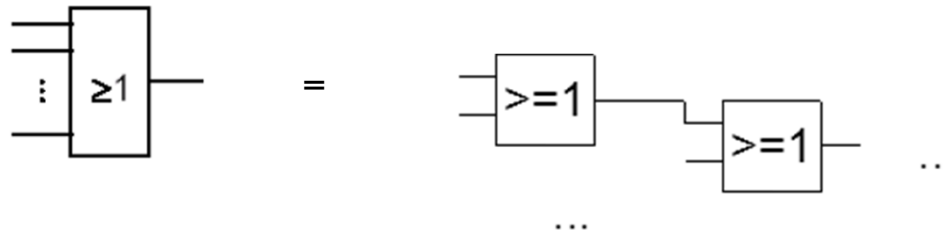


Realisierung Schaltfunktionen mit Gattern - Übersicht Schaltzeichen - erweiterte Darstellung 2/2 - Basisglieder mit mehreren Eingängen

- Zusätzlich arbeitet man zur Vereinfachung in komplexen Anwendungen mit n-stelligen Gattern, z.B.:
- AND mit n Eingängen:



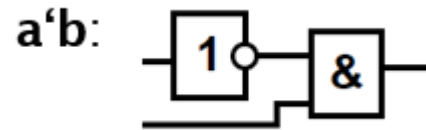
- OR mit n Eingängen :



Realisierung Schaltungsfunktionen aus Basisgattern

- Aus den Basiselementen NOT, AND und OR kann jede beliebige Schaltfunktion aufgebaut werden

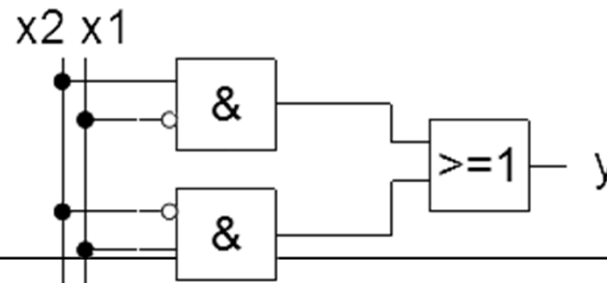
- Beispiel 1:



- Beispiel 2:

$$\text{XOR: } y = (x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2) = (x_1 \overline{x_2}) \vee (\overline{x_1} x_2)$$

// 2x NOT, 2x AND, 1x OR



Schaltungsentwurf

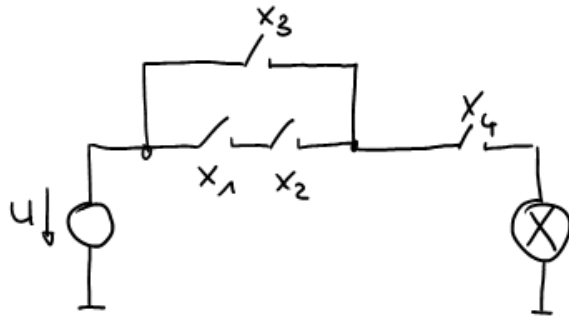
- Allgemeines Vorgehen:
 1. Exakte Funktionsbeschreibung der gesuchten Schaltung
 2. Festlegung der Eingangs- und Ausgangsvariablen
 - 3. Aufstellen der Wahrheitstabelle**
 - 4. Bestimmung des schaltalgebraischen Terms**
 - 5. Vereinfachung und ggf. Umformung des Terms**
 6. Aufbau der Schaltung aus Gattern gemäß dem Term
- -> Optimierung nötig (Ressourcenverbrauch, Berechnungszeit, ...)

Spezifikation und Wahrheitstabelle

- Beispiel Aufzugsteuerung
 - Aufzugsteuerung mit 3 Etagen, wobei die oberen beiden exklusiv genutzt werden sollen, d.h. wenn eine der oberen beiden angefragt werden, dann wird eine gleichzeitige Anfrage der untersten Etage verboten.
 - **Aufgabe:** Baue Schaltung, die gedrückte Knöpfe (x3,x2,x1) auf Korrektheit prüft
 - **Spezifikation:** Funktion f liefert auf der Ausgabe y = „Eingabe_ok“ eine 1, wenn entweder Knopf x2 oder Knopf x1 und nicht Knopf x3 gedrückt wurde.

Nr	x3	x2	x1	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Aufstellen der Wahrheitstabelle - Übung



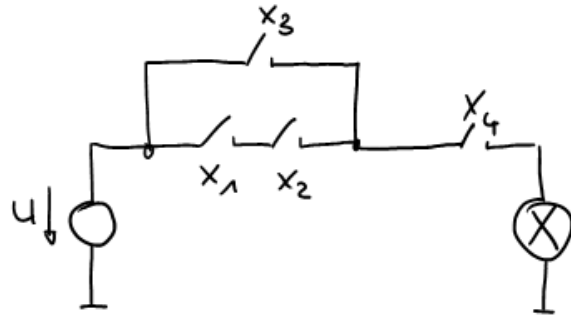
Nr	x4	x3	x2	x1	y
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

Termbestimmung - Normalformen 1/4 - DNF

- Von besonderer technischer und mathematischer Bedeutung sind sog. **Normalformen** boolescher Ausdrücke:
- **Disjunktive Normalform (DNF)**
 - ODER – Verknüpfung von **Mintermen**: $y = \vee m_i$
 - Minterm ist eine UND-Verknüpfung von Eingangsvariablen, die das Ergebnis „1“ haben : $m_i = \wedge x_k'$
 - $x_k' \in \{x_k, \overline{x_k}\}$
 - Bsp:
 - $m_0 = \overline{x_3} \wedge \overline{x_2} \wedge \overline{x_1}$
 - $m_1 = \overline{x_3} \wedge \overline{x_2} \wedge x_1$
 - $m_2 = \overline{x_3} \wedge x_2 \wedge \overline{x_1}$
 - $m_3 = \overline{x_3} \wedge x_2 \wedge x_1$
 - ... $m_7 = x_3 \wedge x_2 \wedge x_1$

Termbestimmung - Normalformen 2/4 - Beispiel DNF

Suchen der Normalformen aus Wahrheitstabelle



▪ DNF:

$$\begin{aligned}
 y &= (x_4 \wedge \overline{x_3} \wedge x_2 \wedge x_1) \vee \\
 &\quad (x_4 \wedge x_3 \wedge \overline{x_2} \wedge \overline{x_1}) \vee \\
 &\quad (x_4 \wedge x_3 \wedge \overline{x_2} \wedge x_1) \vee \\
 &\quad (x_4 \wedge x_3 \wedge x_2 \wedge \overline{x_1}) \vee \\
 &\quad (x_4 \wedge x_3 \wedge x_2 \wedge x_1) \\
 &= m_{11} \vee m_{12} \vee m_{13} \vee m_{14} \vee m_{15}
 \end{aligned}$$

Nr	x4	x3	x2	x1	y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Termbestimmung - Normalformen - DNF - Übung

Nr	x3	x2	x1	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

▪ DNF:

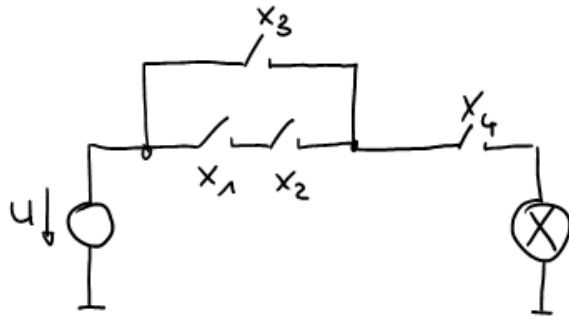
y=

Termbestimmung - Normalformen 3/4 - KNF

- **Konjunktive Normalform (KNF)**
 - UND – Verknüpfung von *Maxtermen*: $y = \bigwedge M_i$
 - Maxterm ist eine ODER – Verknüpfung von Eingangsvariablen, das Ergebnis „0“ haben : $M_i = \bigvee x_k'$
 - $x_k' \in \{x_k, \overline{x_k}\}$
 - Bsp:
 - $M_7 = \overline{x_3} \vee \overline{x_2} \vee \overline{x_1}$
 - $M_6 = \overline{x_3} \vee \overline{x_2} \vee x_1$
 - $M_5 = \overline{x_3} \vee x_2 \vee \overline{x_1}$
 - $M_4 = \overline{x_3} \vee x_2 \vee x_1$
 - ... $M_0 = x_3 \vee x_2 \vee x_1$

Termbestimmung - Normalformen 4/4 - Beispiel KNF

Suchen der Normalformen aus Wahrheitstabelle



▪ KNF:

$$\begin{aligned}
 y &= (x_4 \vee x_3 \vee x_2 \vee x_1) \wedge \\
 &\quad (x_4 \vee x_3 \vee \overline{x_2} \vee x_1) \dots \\
 &\quad \dots \wedge (x_4 \vee \overline{x_3} \vee x_2 \vee \overline{x_1}) \\
 &= M_0 \wedge M_1 \wedge M_2 \wedge M_3 \wedge M_4 \wedge M_5 \\
 &\quad \wedge M_6 \wedge M_7 \wedge M_8 \wedge M_9 \wedge M_{10}
 \end{aligned}$$

Nr	x4	x3	x2	x1	y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Optimierung

- Die Normalform beschreibt zwar die Funktion einer Schaltung, ist aber nicht der kürzest mögliche Term
- -> DMF (Disjunktive Minimal Form) aus Minimierung der DNF
- Dafür gibt es mehrere Verfahren:
 - **Karnaugh Veitch Diagramme (Graphisches Verfahren)**
 - Quine-McCluskey Verfahren
 - Binary Decision Diagramme (BDD)
 - Robinson Verfahren (Erweiterung von Quine-McCluskey)

Optimierung - KV Diagramme - Prinzip 1/2

- KV Diagramm = die graphische Darstellung der DNF
 - Ziel ist die DMF zu erreichen
 - Idee:
 - Betrachtung von möglichst großen Teilfunktionen (Blöcken), die die min-Terme mit ,1' er umfassen
 - Untersuchung dieser Teilfunktionen, ob diese eine Eingangsvariable in positiver Form und negierter Form enthalten -> wenn ja, kann diese Variable weggelassen werden (vgl. Komplementsgesetz $x1 \times 1' = 0$)
 - Die einzelnen minimierten Teilfunktionen (Blöcke) werden OR-verknüpft (vgl. Basis ist ja die DNF)
-

Optimierung - KV-Diagramme - Prinzip 2/2

- Algorithmus
- 1. Zeichne KV-Diagramm und füge alle „1“ der Minterme der Ergebnisvariablen (y) ein
- 2. Fasse benachbarte „1“ zu möglichst großen 2er-Potenz Blöcken zusammen
 - Starte mit dem größten Block
 - Überdeckungen sind erlaubt
 - Alle Einsen müssen abgedeckt werden

Beispiel

Nr	x2	x1	y	m
0	0	0	1	$x_2'x_1'$
1	0	1	0	$x_2'x_1$
2	1	0	1	x_2x_1'
3	1	1	1	x_2x_1

y:

	x2	x2'
x1	x_2x_1	$x_2'x_1$
x1'	x_2x_1'	$x_2'x_1'$

1.: y:

	x2	x2'
x1	1	
x1'	1	1

2.: y:

	x2	x2'
x1	1	
x1'	1	1

Diagram showing groupings: An orange circle groups the two '1's in the x1 column (x2 and x2'). A red circle groups the two '1's in the x1' row (x2 and x2').

-> $y = x_2 + x_1'$

Optimierung - KV-Diagramme - Beispiel 2 vars

Nr	x2	x1	y	m
0	0	0	1	$x_2'x_1'$
1	0	1	0	$x_2'x_1$
2	1	0	1	x_2x_1'
3	1	1	1	x_2x_1

y:

	x2	x2'
x1	x_2x_1	$x_2'x_1$
x1'	x_2x_1'	$x_2'x_1'$

y:

	x2	x2'
x1	1	
x1'	1	1

Diagram illustrating the Karnaugh map for the function y. The map shows the function y as a sum of products. The variables x2 and x1 are the inputs, and x2' and x1' are the complements. The function y is 1 for the combinations (x2, x1) = (0, 0), (1, 0), (1, 1), and (0, 1). The map is shown with the variables x2 and x1 on the axes, and the function value y in the cells. The cells (0, 0) and (1, 0) are grouped together, and the cells (1, 0) and (1, 1) are grouped together. The function y is then simplified to $y = x_2 + x_1'$.

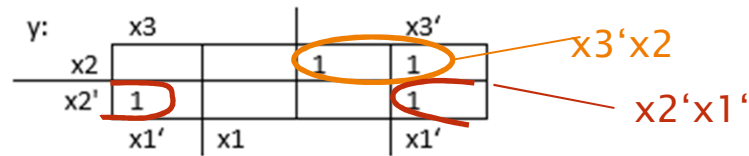
-> $y = x_2 + x_1'$

Mögliche Pakete zum Zusammenfassen:

- 2er senkrecht, 2er waagrecht
- 4er Block

Optimierung - KV-Diagramme - Beispiel 3 vars

Nr	x3	x2	x1	y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0



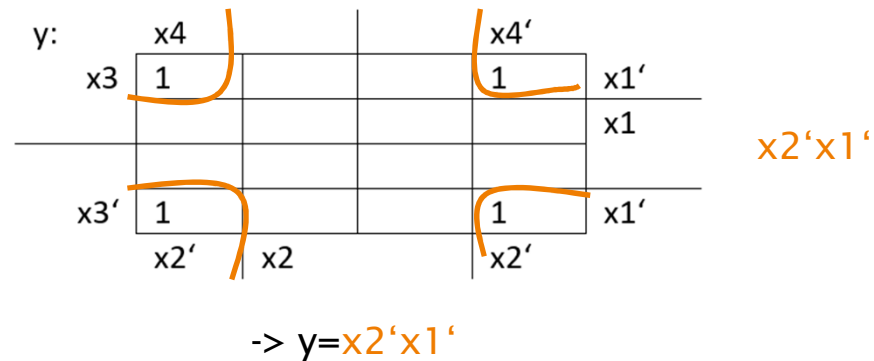
$$\rightarrow y = x3'x2 + x2'x1'$$

Mögliche Pakete zum Zusammenfassen:

- 2er senkrecht, 2er waagrecht
- 4er waagrecht, 4er Block
- 8er Block
- ! Achtung 3vars: über die seitlichen Grenzen hinweg darf zusammengefasst werden (aber nicht über die Grenzen oben/unten)
- Vorstellung: Zylinder

Optimierung - KV-Diagramme - Beispiel 4 vars

Nr	x4	x3	x2	x1	y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



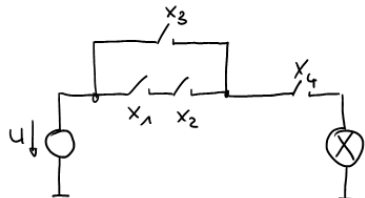
Mögliche Pakete zum Zusammenfassen:

- 2er senkrecht, 2er waagrecht
- 4er waagrecht, 4er senkrecht, 4er Block
- 8er waagrecht, 8er senkrecht
- 16er Block

-! Achtung 4vars: über die seitlichen UND oben/unten Grenzen hinweg darf zusammengefasst werden

--Vorstellung Kugel

Optimierung - KV-Diagramme - 4 vars - Übung



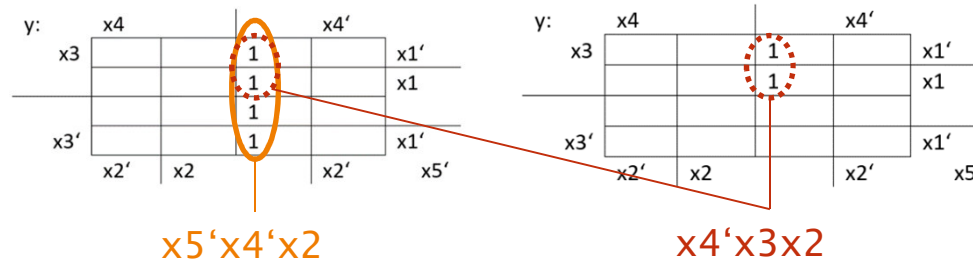
Nr	x4	x3	x2	x1	y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

y:	x4		x4'	
x3				x1'
				x1
x3'				x1'
	x2'	x2		x2'

-> y=?

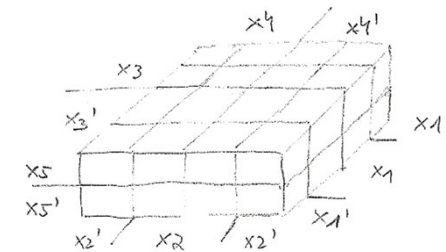
Nr	x5	x4	x3	x2	x1	y
0	0	0	0	0	0	0
1	0	0	0	0	1	0
2	0	0	0	1	0	1
3	0	0	0	1	1	1
4	0	0	1	0	0	0
5	0	0	1	0	1	0
6	0	0	1	1	0	1
7	0	0	1	1	1	1
8	0	1	0	0	0	0
9	0	1	0	0	1	0
10	0	1	0	1	0	0
11	0	1	0	1	1	0
12	0	1	1	0	0	0
13	0	1	1	0	1	0
14	0	1	1	1	0	0
15	0	1	1	1	1	0
16	1	0	0	0	0	0
17	1	0	0	0	1	0
18	1	0	0	1	0	0
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	0
22	1	0	1	1	0	1
23	1	0	1	1	1	1
24	1	1	0	0	0	0
25	1	1	0	0	1	0
26	1	1	0	1	0	0
27	1	1	0	1	1	0
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	0

- KV-Diagramme - Beispiel 5 vars



$$\rightarrow y = x5'x4'x2 + x4'x3x2 = (x4'x2)(x3 + x5')$$

Mögliche Pakete zum Zusammenfassen:
-2er, 4er, 8er, 16er, 32er Blöcke
-! Achtung 5vars: über die seitlichen und oben/unten Grenzen hinweg UND auch von Ebene zu Ebene darf zusammengefasst werden
--Vorstellung Quader mit zwei Ebenen



Optimierung - KV-Diagramme - mit „don't care“ Werten 1/2

- „don't care“ Werte:
 - Es kann vorkommen, dass bei bestimmten Anwendungen nicht für alle Kombinationen von Eingangsvariablen die Ausgangsvariable definiert ist
 - Grund:
 - Diese Kombinationen kommen nie vor
 - Es ist irrelevant, ob sie auf 0 oder 1 abgebildet werden
- Dies hat Auswirkungen auf die Optimierung:

▪ Beispiel:

Nr	x3	x2	x1	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	-
7	1	1	1	-

Optimierung - KV-Diagramme - mit „don't care“ Werten 2/2

- Vorgehen für die Optimierung mit KV-Diagramm
 - Wenn in einer Wertetabelle irrelevante Ausgabewerte (oft als „x“ oder „-“ markiert) vorhanden sind, können diese für das KV-Diagramm wahlweise mit einer 1 oder 0 (leer) belegt werden
 - Man wählt die Belegung so, dass sich die größten 1er Blöcke ergeben
- Beispiel:

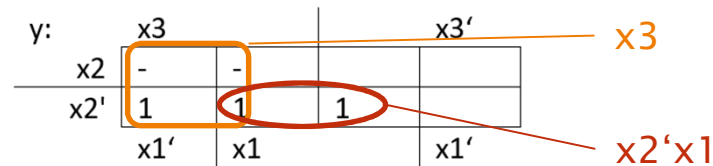
Nr	x3	x2	x1	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	-
7	1	1	1	-

„don't care“ -> 0



$$\rightarrow y = x3x2' + x2'x1$$

„don't care“ -> 1



$$\rightarrow y = x3 + x2'x1$$

Eine var gespart!

Optimierung - KV-Diagramme - mit mehrwertigen Funktionen

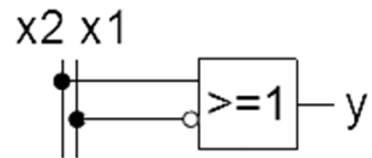
- Wertetabellen können Werte für mehr als eine Ausgangsvariable haben
- Die Optimierungsverfahren funktionieren aber nur für eine Ausgangsvariable
- -> Anwenden der Optimierung pro Ausgangsvariable; d.h. pro Ausgangsvariable 1 KV-Diagramm

Nr	x3	x2	x1	y1	y2
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	0	0
3	0	1	1	0	0
4	1	0	0	1	0
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	0	1

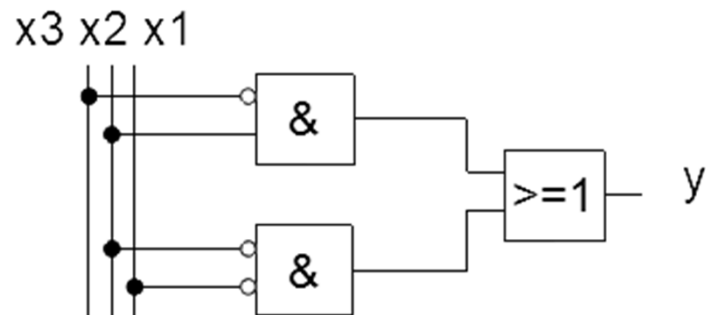
-> 1 KV für y1,
1 KV für y2

Schaltungsaufbau

$$y = x_2 + x_1'$$



$$y = x_3'x_2 + x_2'x_1'$$



Schaltungsaufbau - Übung

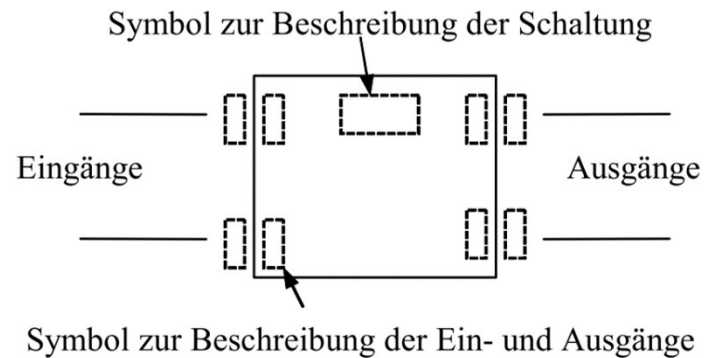
$y = x_4 x_3 + x_4 x_2 x_1 \rightarrow$ Gatterschaltplan?

$y = x_5' x_4' x_2 + x_4' x_3 x_2 \rightarrow$ Gatterschaltplan?

(Erweiterte) Schaltsymbole - Motivation

- Neben den bereits vorgestellten Schaltsymbolen gibt es noch erweiterte Schaltsymbole, die in der IEC 60617 / DIN EN 60617 / DIN 40900 definiert sind
- Da diese Symbole häufig in Schaltplänen verwendet werden, soll die Systematik hier kurz vorgestellt werden

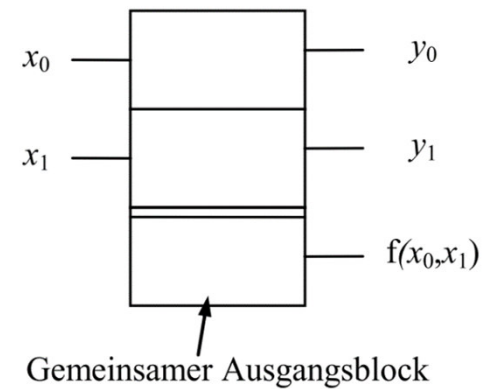
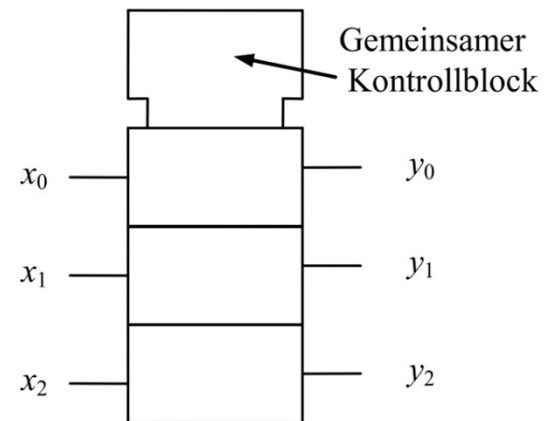
(Erweiterte) Schaltsymbole - Generelles Prinzip I/II



- Oben befindet sich die eigentliche Funktion (&, ≥ 1 , $=1$, 1 , ...)
- Eingänge sind in der Regel links, Ausgänge rechts
- Die rechteckig hervorgehobenen Bereiche können zusätzliche Angaben über die Eingänge und Ausgänge haben
- Die äußeren Bereiche wirken außen (z.B.: Negationskreis)
- Die inneren Rechteckbereiche wirken auf den inneren Zustand der Schaltung

(Erweiterte) Schaltsymbole - Generelles Prinzip II/II

- Daneben kann es eine gemeinsamen Kontrollblock sowie gemeinsamen Ausgangsblock geben

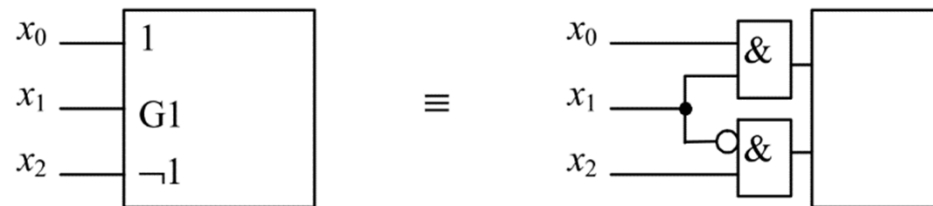


(Erweiterte) Schaltsymbole - Abhängigkeitsnotation

- Die Abhängigkeitsnotation beschreibt den Einfluss eines Eingangs (oder Ausgangs) auf andere Ein- und Ausgänge durch einen Buchstaben, der die Art des Einflusses näher beschreibt
- Dem Buchstaben folgt eine Zahl zur Identifikation
- Die gleiche Zahl findet man bei den Ein- und Ausgängen, auf die dieser Einfluss ausgeübt wird

(Erweiterte) Schaltsymbole - Abhängigkeitsnotation - UND-Abhängigkeit (G)

- Ein „G“ bezeichnet eine UND-Abhängigkeit
- Beispiel Eingang abhängig von Eingang:

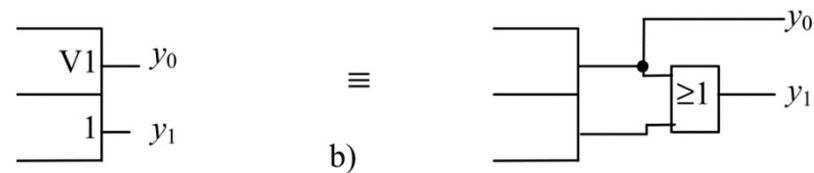
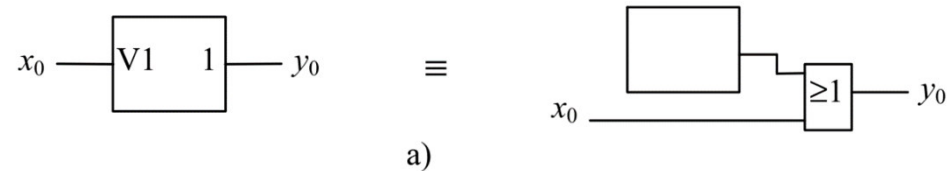


- x_1 legt eine G-Abhängigkeit fest, die sich auf x_0 und x_2 auswirkt
- Beispiel Eingang abhängig vom Ausgang:



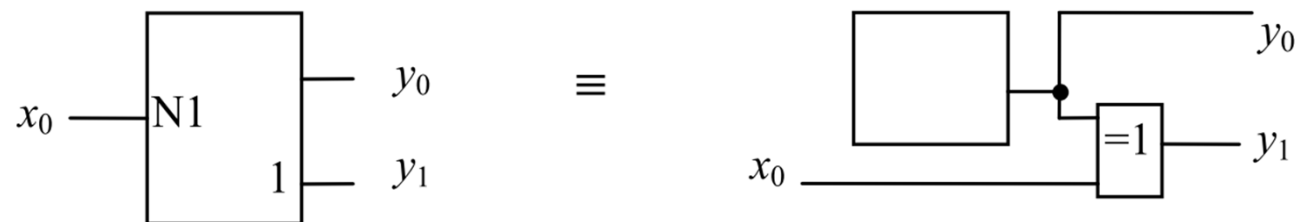
(Erweiterte) Schaltsymbole - Abhängigkeitsnotation - ODER-Abhängigkeit (V)

- Ein „V“ Bezeichnet eine ODER-Abhängigkeit
 - Wenn ein V<n>-Eingang bzw. Ausgang eine ‚1‘ hat, so haben alle Ein- und Ausgänge mit der Kennzeichnung <n> den Wert ‚1‘
 - Wenn ein V<n>- Eingang bzw. Ausgang den Wert ‚0‘ hat, so haben die davon abhängen Ein- und Ausgänge den normalen Wert
- Beispiel: a) Ausgang abhängig von Eingang;
b) Ausgang abhängig von Ausgang

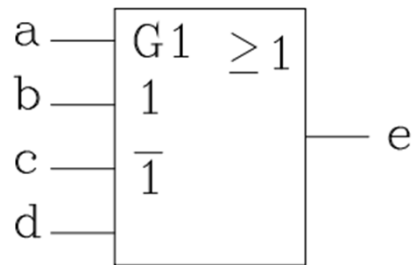


(Erweiterte) Schaltsymbole - Abhängigkeitsnotation - XOR-Abhängigkeit (N)

- Wenn ein N<n>-Eingang bzw. Ausgang eine ,1' hat, so haben alle Ein- und Ausgänge mit der Kennzeichnung <n> den negierten Wert
- Wenn ein N<n>-Eingang bzw. Ausgang den Wert ,0' hat, so haben die davon abhängen Ein- und Ausgänge den normalen Wert



Erweiterte) Schaltsymbole - Abhängigkeitsnotation - Übung



-> Äquivalenter
Schaltplan?,
äquivalente
boolsche
Gleichung für
e?