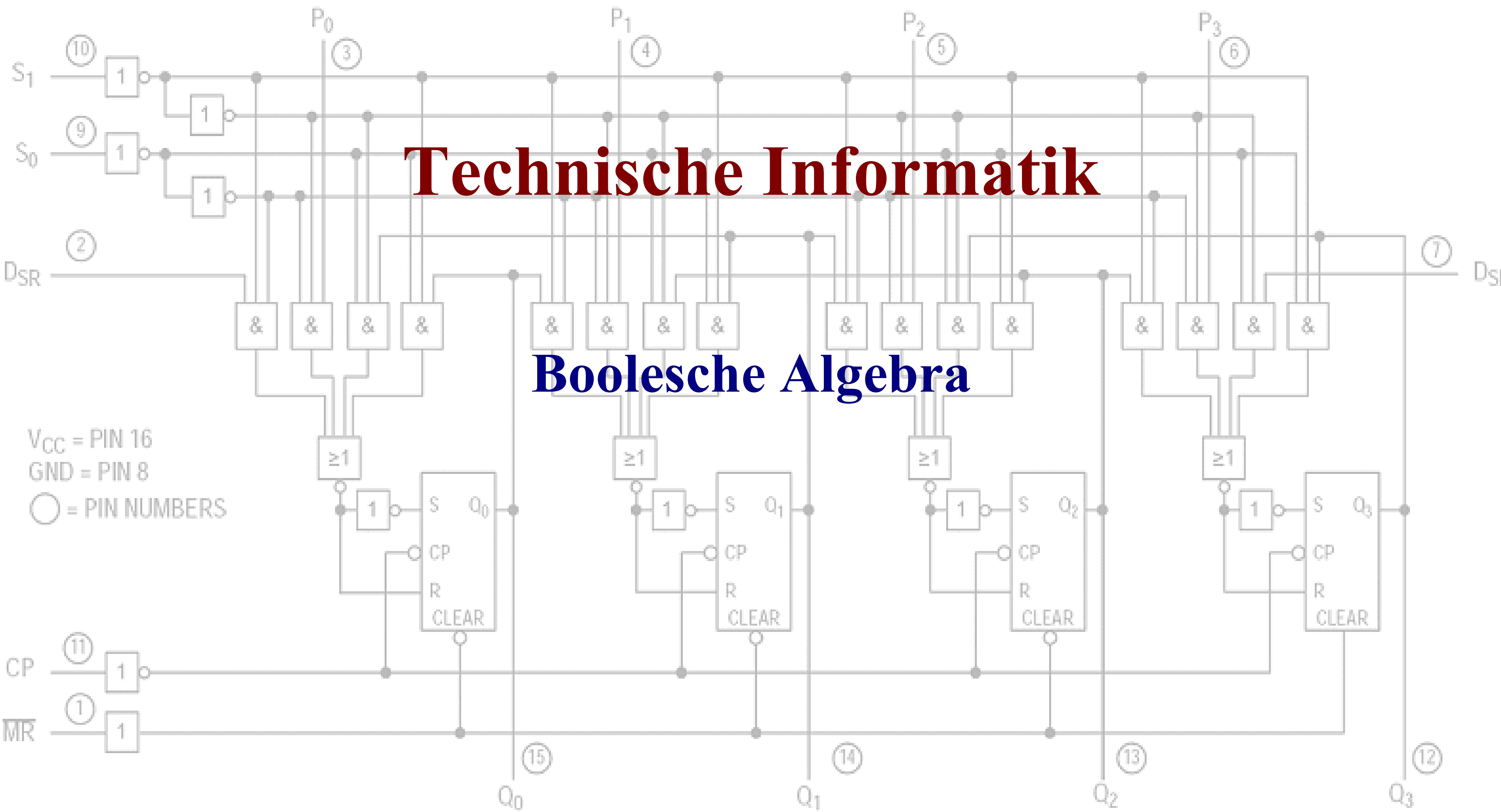
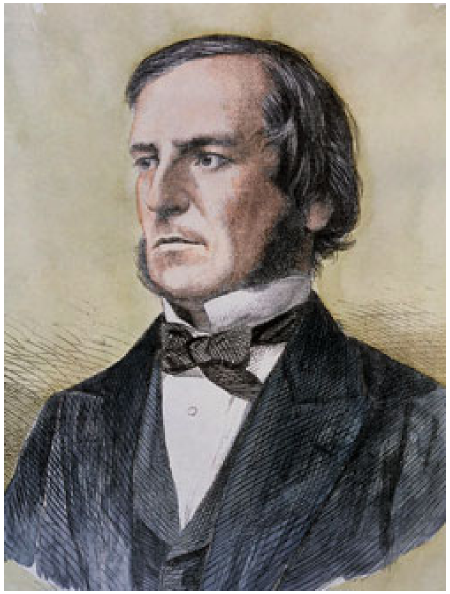
Technische Informatik

Boolesche Algebra



Historisches

- George Boole (1815-1864)
  - Mathematiker & Philosoph
  - Buch “An Investigation of the Laws of Thought” - eine Algebra der Logik



- Schaltalgebra mit den Grundoperationen UND, ODER oder die NEGATION wird als die *Boolesche Algebra* bezeichnet
- Gängige Rechner arbeiten nach den Regeln der Booleschen Algebra, sie verwenden (meistens) “binäre” d.h. zweiwertige Signale {0,1}

Axiome der Booleschen Algebra

- Boolesche Algebra = algebraische Struktur, die
  - eine Menge $\boldsymbol{B}$ mit 2 Elementen: $\boldsymbol{B} = \{0,1\}$
  - und folgende Operationen enthält:

Negation	(NICHT, NOT)	$\bar{a}$	$\neg a$
Disjunktion	(ODER, OR)	$a + b$	$a \vee b$
Konjunktion	(UND, AND)	$a \cdot b$	$a \wedge b$

- Die Booleschen Operationen werden durch *Wahrheitstafeln* definiert, die für jede mögliche Kombination der Eingangswerte die Ausgangswerte liefern

NOT	
$\boldsymbol{a}$	$\bar{\boldsymbol{a}}$
0	1
1	0

OR		
$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a + b}$
0	0	0
0	1	1
1	0	1
1	1	1

AND		
$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{a \cdot b}$
0	0	0
0	1	0
1	0	0
1	1	1

„Rechenregeln“ der Booleschen Algebra

Kommutativgesetz	$a \cdot b = b \cdot a$	$a+b = b+a$
Distributivgesetz	$a \cdot (b+c) = (a \cdot b)+(a \cdot c)$	$a+(b \cdot c) = (a+b) \cdot (a+c)$
Assoziativgesetz	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	$a+(b+c) = (a+b)+c$
Komplement	$a \cdot \overline{a} = 0$	$a+\overline{a} = 1$
Neutrale Elemente	$\overline{1} = 0$	$\overline{0} = 1$
Involution	$\overline{\overline{a}} = a$	
Identität	$a \cdot 1 = a$	$a+0 = a$
Elimination	$a \cdot 0 = 0$	$a+1 = 1$
Idempotenz	$a \cdot a = a$	$a+a = a$
Vereinigung	$(a+b) \cdot (a+\overline{b}) = a$	$a \cdot b+a \cdot \overline{b} = a$
De Morgan'sches Theorem	$\overline{(a \cdot b)} = \overline{a} + \overline{b}$	$\overline{(a+b)} = \overline{a} \cdot \overline{b}$
Absorption	$(a+b) \cdot a = a$	$(a \cdot b)+a = a$
	$(a \cdot \overline{b})+b = a+b$	$(a+\overline{b}) \cdot b = a \cdot b$
	$(a+b) \cdot (\overline{a}+c) = (a \cdot c)+(\overline{a} \cdot b)+(b \cdot c) = (a \cdot c)+(\overline{a} \cdot b)$	

„Rechenregeln" der Booleschen Algebra

- Ferner gilt in der Boole'schen Algebra das **Dualitätsprinzip**:

Zu jeder Aussage, die sich aus den vier Axiomen ableiten lässt, existiert eine dazu duale Aussage. Sie entsteht dadurch, dass man gleichzeitig die Operationen **+** und **·** sowie die neutralen Elemente ***1*** und ***0*** vertauscht.

- Beispiel:

$$y = (a \cdot b) + (\overline{a} \cdot c) \quad \Leftrightarrow \quad \overline{y} = (\overline{a} + \overline{b}) \cdot (a + \overline{c})$$

Boolesche Terme und logische Funktionen

- Aus einer Menge Boolescher Variablen $\{a_1, a_2, \ldots, a_N\}$ und den Konstanten ***0*** und ***1*** sowie den Operatoren NOT, OR und AND lassen sich *Boolesche Terme* konstruieren
- Eine Abbildung $f{:}B^n \rightarrow B$ heißt *Boolesche Funktion* (logische Funktion)
- Eine Boolesche Funktion kann durch einen Booleschen Term oder durch eine Wahrheitstafel beschrieben werden
- Beispiel:

Boolesche Funktion:

$$f(a,b,c) = a \cdot \overline{b} + c \cdot b$$

Auch hier gilt:

„Punktrechnung vor Strichrechnung"

Die AND-Operation hat eine höhere Priorität als die OR-Operation!

Wahrheitstabelle:

a	*b*	*c*	*f(a,b,c)*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Normalformen

- Seien $a_1, a_2, \ldots, a_n$ Boolesche Variablen, und $x_i = a_i$ oder $x_i = \overline{a_i}$ dann können daraus die Terme

AND = Konjunktion
OR = Disjunktion

$$(x_1 \cdot x_2 \cdot \ldots \cdot x_n) \qquad \textit{Konjunktionsterm}$$

oder

$$(x_1 + x_2 + \ldots + x_n) \qquad \textit{Disjunktionsterm}$$

gebildet werden, in denen die Eingangsvariablen a_i in negierter oder nicht-negierter Form auftreten

- Disjunktive Normalform (DNF)
  - logische Funktion, die aus Konjunktionstermen K_i gebildet wird

Disjunktion

$$f_{DNF} = K_1 + K_2 + \ldots + K_m \qquad \textit{Beispiel: } f_{DNF} = (a_1 \cdot \overline{a_2}) + (\overline{a_1})$$

- Konjunktive Normalform (KNF)
  - logische Funktion, die aus Disjunktionstermen D_i gebildet wird

Konjunktion

$$f_{KNF} = D_1 \cdot D_2 \cdot \ldots \cdot D_m \qquad \textit{Beispiel: } f_{KNF} = (\overline{a_1} + a_2) \cdot (a_1 + \overline{a_2})$$

Normalformen

- *Minterm* = Konjunktionsterm, der alle Variablen enthält
- *Maxterm* = Disjunktionsterm, der alle Variablen enthält
- Eine *kanonische Normalform* (= volldisjunkte bzw. vollkonjunkte Normalform) liegt vor, wenn alle Konjunktions- bzw. Disjunktionsterme alle Variablen enthalten
  - Eine voll<u>disjunkte</u> Normalform besteht aus der Booleschen <u>Summe</u> von <u>Min</u>termen

$$\textit{Beispiel:}\ \ f_{DNF} = (a_1 \cdot \overline{a_2} \cdot a_3) + (\overline{a_1} \cdot a_2 \cdot a_3)$$

  - Eine voll<u>konjunkte</u> Normalform besteht aus dem Booleschen <u>Produkt</u> von <u>Max</u>termen

$$\textit{Beispiel:}\ \ f_{KNF} = (\overline{a_1} + \overline{a_2} + a_3) \cdot (\overline{a_1} + a_2 + a_3)$$

- Bei n Eingangsvariablen gibt es insgesamt 2^n Min- bzw. Maxterme
- Jede logische Funktion lässt sich in Normalform (DNF oder KNF) darstellen

Disjunktive Normalform

- Erzeugung einer logischen Funktion in **disjunktiver** Normalform (DNF) aus einer Wahrheitstabelle:
  - Für alle Zeilen mit dem Funktionswert $f(a_1, a_2, \ldots, a_n) = 1$ jeweils einen **Minterm** aus allen Eingangsvariablen a_i so bilden, dass
    $\overline{a}_i$ eingesetzt wird, wenn $a_i = 0$
    a_i eingesetzt wird, wenn $a_i = 1$
  - Bilde abschließend die Disjunktion (ODER-Verknüpfung) aller Minterme
- Beispiel:

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

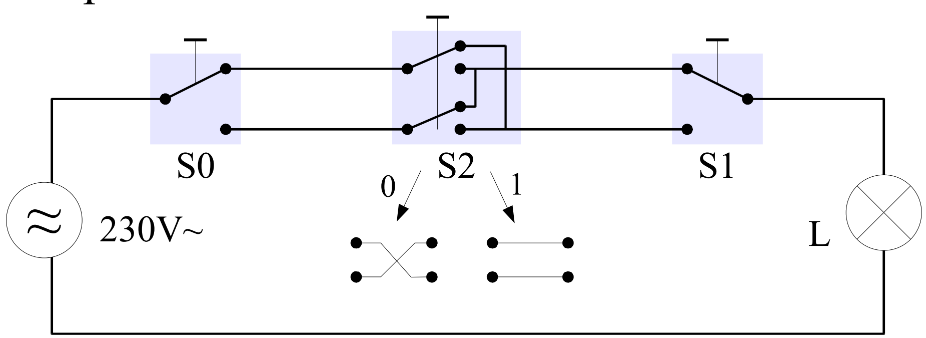
0 1 1   1 0 0   1 1 1

$$f(a,b,c) = (\overline{a} \cdot b \cdot c) + (a \cdot \overline{b} \cdot \overline{c}) + (a \cdot b \cdot c)$$

Beispiel Disjunktive Normalform

- Beispiel Treppenhausschaltung

Schaltplan:



Funktion:

Eingang			Ausgang
S2	S1	S0	L
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	1
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1

$$L = \overset{0\ 0\ 1}{\overline{S2}\cdot\overline{S1}\cdot S0} + \overset{0\ 1\ 0}{\overline{S2}\cdot S1\cdot\overline{S0}} + \overset{1\ 1\ 1}{S2\cdot S1\cdot S0} + \overset{1\ 0\ 0}{S2\cdot\overline{S1}\cdot\overline{S0}}$$

Konjunktive Normalform

- Erzeugung einer logischen Funktion in **konjunktiver** Normalform (KNF) aus einer Wahrheitstabelle:
  - Für alle Zeilen mit dem Funktionswert $f(a_1, a_2, \ldots, a_n) = 0$ jeweils einen **Maxterm** aus allen Eingangsvariablen a_i so bilden, dass

    a_i eingesetz wird, wenn $a_i = 0$

    $\overline{a}_i$ eingesetzt wird, wenn $a_i = 1$
  - Bilde abschließend die Konjunktion (UND-Verknüpfung) aller Maxterme

- Beispiel:

0 1 1 1 0 0 1 1 1

$$f(a,b,c) = (a+\overline{b}+\overline{c})\cdot(\overline{a}+b+c)\cdot(\overline{a}+\overline{b}+\overline{c})$$

a	b	c	f(a,b,c)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Disjunktive / Konjunktive Normalform

- Die Vorschriften zur Erzeugung der DNF und der KNF sind über das Dualitätsprinzip miteinander verknüpft
- Sind mehr Zeilen mit dem Ergebnis ***1*** als mit dem Ergebnis ***0*** vorhanden, ist die Erzeugung der KNF günstiger, da weniger Terme benötigt werden
- Mit Hilfe der Sätze der Booleschen Algebra können DNF und KNF ineinander überführt werden

Minimierungsverfahren

- Logische Funktionen in vollständiger Normalform enthalten Redundanzen, die mit Minimierungsverfahren beseitigt werden können
- Algebraische Verfahren
  - Anwendung der Booleschen Rechenregeln
- Karnaugh-Veitch-Diagramm (KV-Diagramm)
  - grafisches Verfahren
  - geeignet für Systeme mit bis zu 4 Variablen
- Tabellarische Verfahren
  - z.B. Quine – McCluskey
  - geeignet für komplexe Systeme mit vielen Variablen

Karnaugh-Veitch-Diagramm

- Verfahren:
  - Das KV-Diagramm ersetzt die Wahrheitstafel. Durch blockweise Zusammenfassung gleicher Funktionswerte werden Redundanzen vermieden
- Ausgangspunkt:
  - Logische Funktion mit *n* Variablen in **volldisjunktiver Normalform**
- Beispiel:

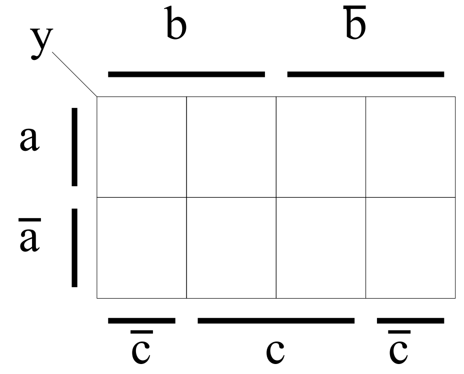
$$y = (\bar{a} \cdot b \cdot \bar{c}) + (a \cdot \bar{b} \cdot \bar{c}) + (a \cdot b \cdot \bar{c}) + (a \cdot b \cdot c)$$

Karnaugh-Veitch-Diagramm

- 1\. Schritt:
  - KV-Diagramm mit 2^n Feldern anlegen. Jedes Feld repräsentiert einen Minterm
  - Die Ränder werden mit den nicht-negierten und negierten Eingangsvariablen so beschriftet, dass sich beim Übergang zum benachbarten Feld nur eine Eingangsvariable ändert

- Beispiel:

$$y = (\overline{a} \cdot b \cdot \overline{c}) + (a \cdot \overline{b} \cdot \overline{c}) + (a \cdot b \cdot \overline{c}) + (a \cdot b \cdot c)$$



Karnaugh-Veitch-Diagramm

- 2. Schritt:
  - Für jeden Minterm, der in der logischen Funktion vorhanden ist, wird eine „1“ in das entsprechende Feld eingetragen

- Beispiel:

$$y = (\bar{a} \cdot b \cdot \bar{c}) + (a \cdot \bar{b} \cdot \bar{c}) + (a \cdot b \cdot \bar{c}) + (a \cdot b \cdot c)$$

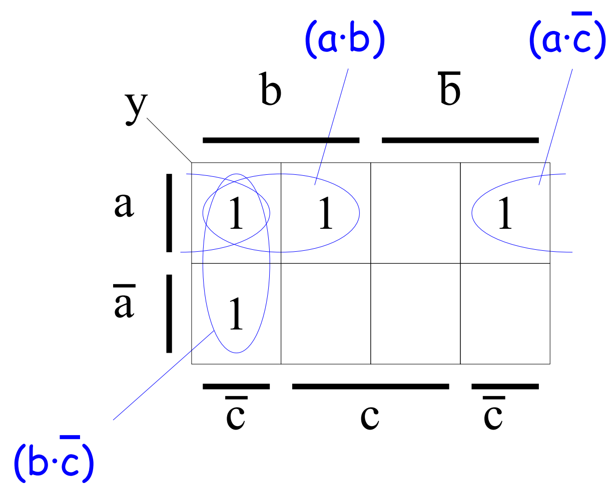
y	b	b	$\bar{b}$	$\bar{b}$
a	1	1		1
$\bar{a}$	1			
	$\bar{c}$	c	c	$\bar{c}$

Karnaugh-Veitch-Diagramm

- 3. Schritt:
  - Benachbarte Felder, die eine „1“ enthalten, werden zu möglichst großen rechteckigen Blöcken zusammengefasst
  - Die Blöcke können 2, 4, oder 8 Felder umfassen
  - Anfang und Ende einer Zeile bzw. Spalte sind benachbart, das Diagramm kann gewissermaßen zu einem Zylinder zusammengerollt werden. Blöcke dürfen also über den Rand hinaus angelegt werden
  - Ein Feld kann mehreren Blöcken zugeordnet werden

- Beispiel:

$$y = (\bar{a} \cdot b \cdot \bar{c}) + (a \cdot \bar{b} \cdot \bar{c}) + (a \cdot b \cdot \bar{c}) + (a \cdot b \cdot c)$$

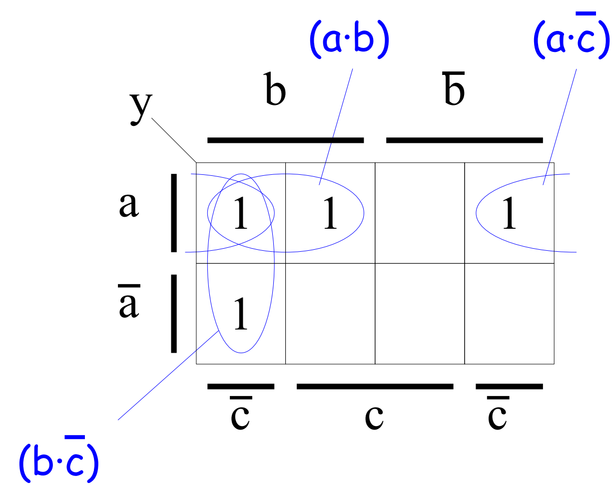


Karnaugh-Veitch-Diagramm

- 4. Schritt:
  - Jeder Block repräsentiert einen Konjunktionsterm der minimierten logischen Funktion
  - Je nach Blockgröße entfallen 1,2 oder 3 redundante Eingangsvariablen
  - Die minimierte logische Funktion ergibt sich aus der Disjunktion (ODER-Verknüpfung) aller Konjunktionsterme

- Beispiel:

$$y = (\bar{a} \cdot b \cdot \bar{c}) + (a \cdot \bar{b} \cdot \bar{c}) + (a \cdot b \cdot \bar{c}) + (a \cdot b \cdot c)$$



Minimierte Funktionsgleichung in DNF:

$$y = (b \cdot \bar{c}) + (a \cdot b) + (a \cdot \bar{c})$$