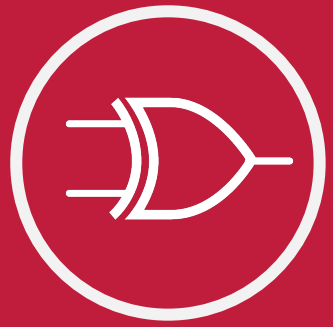
Technische Universität Braunschweig

Informatik für Ingenieure – VL 4

2. ENTWURF UND ANALYSE KOMBINATORISCHER SCHALTUNGEN



Übersicht

Letztes Mal:

Einleitung zu Logik und Digital Schaltungen

Boolesche Algebra

Axiome, Sätze

Boolesche Gleichungen

Heute:

Minimierung von Schaltfunktionen

Karnaugh-Veith Mappen

Quine-McCluskey

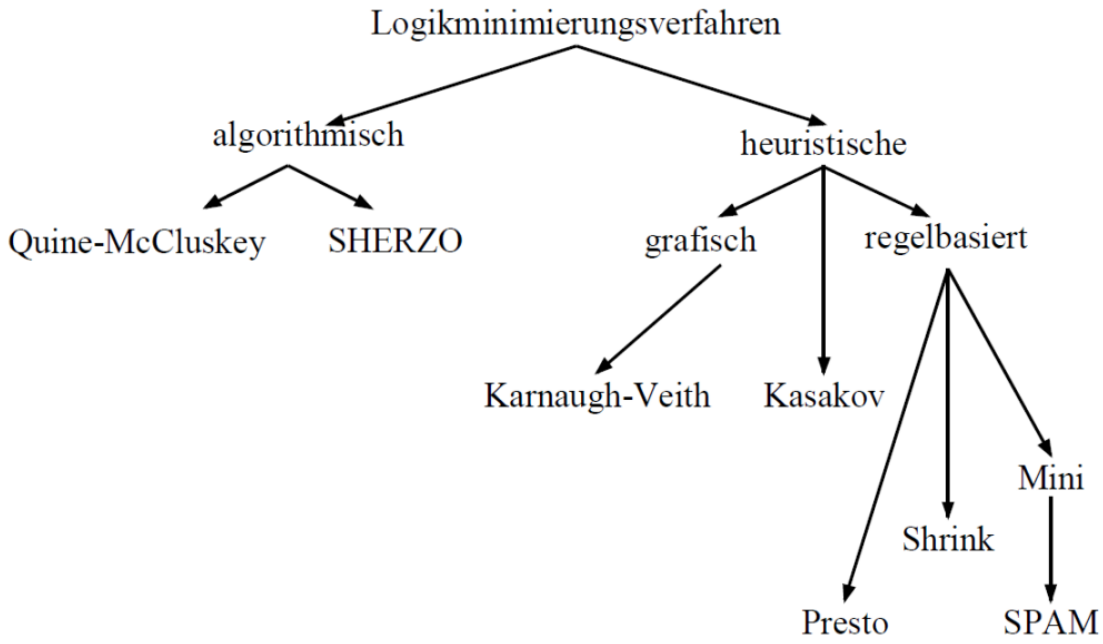
Einführung Schaltnetzrealisierung

Teile des heutigen Vortrags basiert auf der Vorlesungen von
Prof. H Michalik (TU Braunschweig)
Prof. S. Harris (TU Darmstadt) und
Prof. P. Fischer (UniHeidelberg)
Prof. J. Wawrzynek (UC Berkley)

Minimierung von Schaltfunktionen

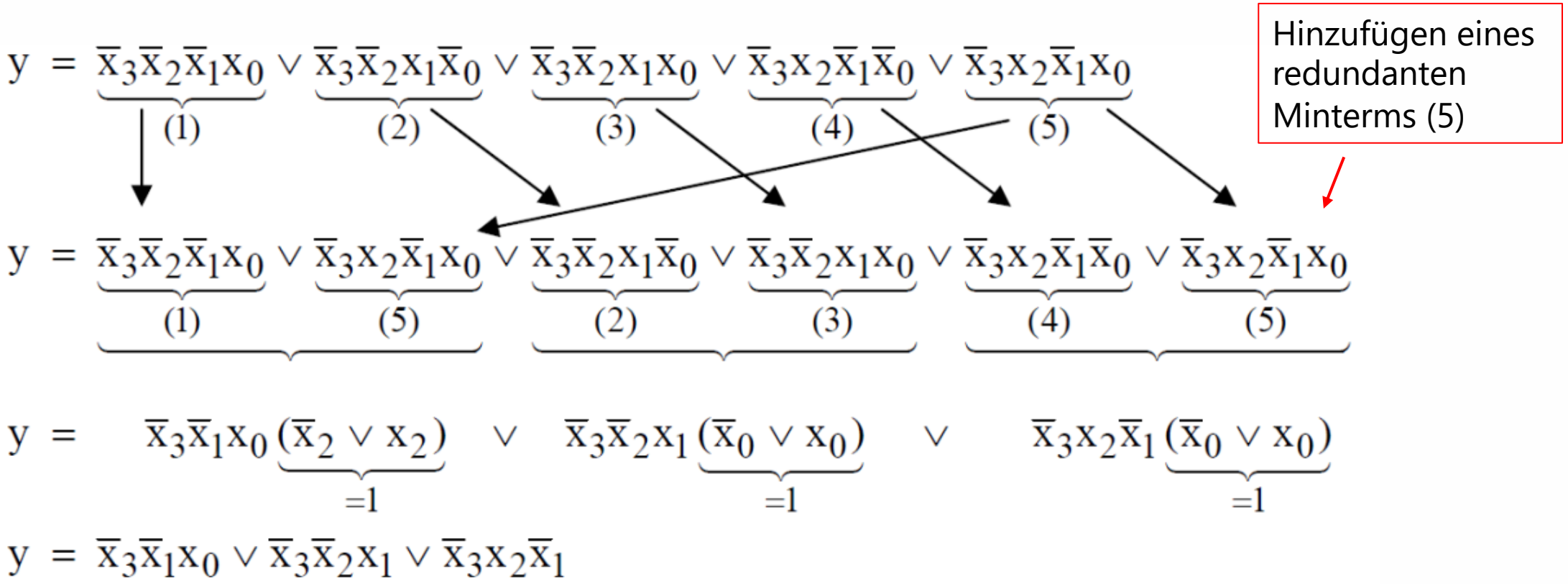
Zur Minimierung von Schaltfunktionen können für Problemstellungen geringer Komplexität graphische heuristische Verfahren eingesetzt werden. Für komplexe Schaltfunktionen gibt es sowohl heuristische als auch algorithmische Verfahren, wobei die algorithmischen Verfahren eine optimale Lösung im obigen Sinne liefern und rechnerunterstützt eingesetzt werden.

Wir werden für beide Gruppen jeweils ein Beispiel erläutern



Manuelle Verfahren

Mit den Rechenregeln der Booleschen Algebra können die algebraischen Ausdrücke einer Schaltfunktion minimiert werden.

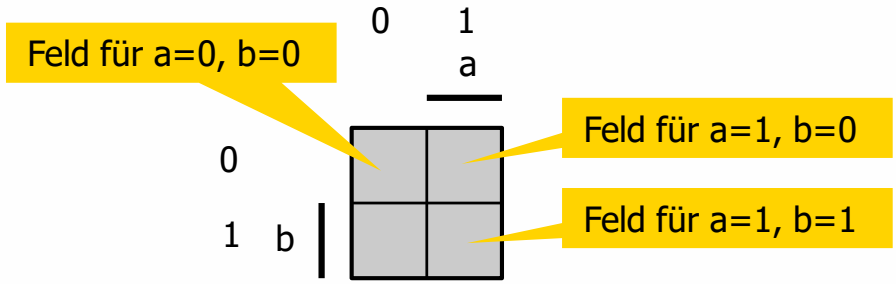


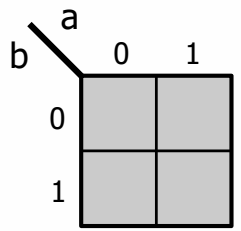
Karnaugh - Diagramme

Eine weitere Darstellungsform für Funktionen/Ausdrücke benutzt zweidimensionale **Tafeln**, die auch als **Karnaugh (-Veith) - Diagramme** (**KMAPs**) bezeichnet werden

Sie geben einen **grafischen Eindruck** der Funktion und
können zur **Logikminimierung** benutzt werden.

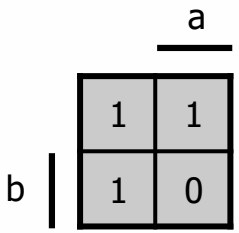
Für eine Funktion von 2 Variablen (4 mögliche Eingangskombinationen) zeichnet man:



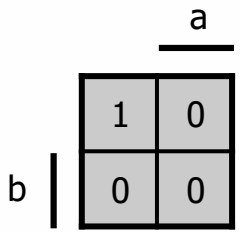


Manchmal auch
so geschrieben

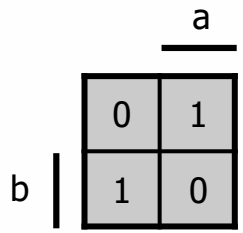
Einfache Beispiele:



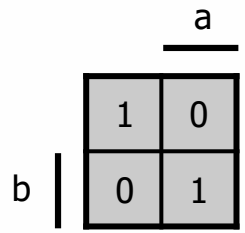
NAND2



NOR2



XOR2



XNOR2

Größere KMAPs

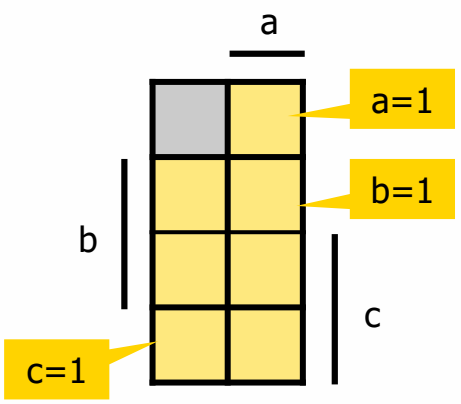
Für Funktionen mit >2 Variablen werden die Tafeln erweitert

Sie sind nützlich, wenn für jede Variable die Einsen in den Zeilen/Spalten **einen Block** bilden.

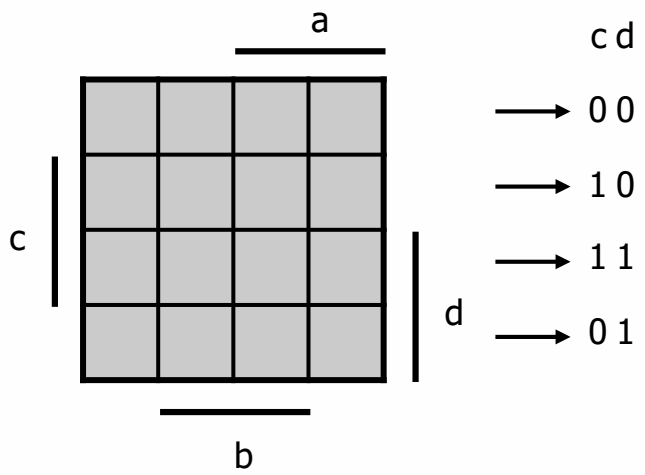
Dies kann nur für je 2 Variablen pro Dimension erfüllt werden. Sie sind dann Gray-codiert.

Für >4 Variablen müsste man in die dritte Dimension gehen, ab 7 Variablen wird's schwierig....

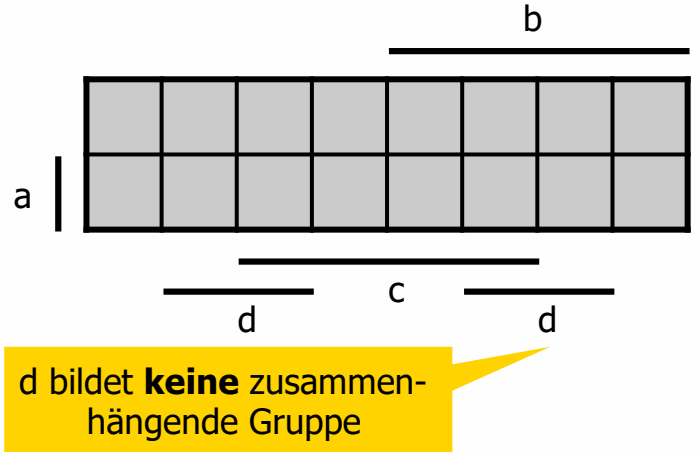
3 Variablen:



4 Variablen:



4 Variablen - Alternative:



Visualisierung der Nachbarschaft

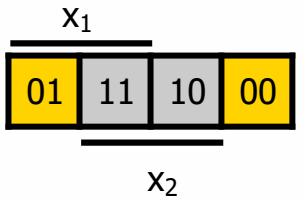
Die Nachbarschaft in N-stelligen Codes mit Hamming-Distanz H=1 wird in einer N-dimensionalen Darstellung besser verständlich:

Auf jeder Kante ändert sich ein Bit. Ein Code mit H=1 entsteht also beim 'Wandern auf Kanten‘

Auf jeder (N-1)-dimensionalen ‚Hyper‘-Ebene (senkrecht zu einer Achse) ist ein Bit konstant.

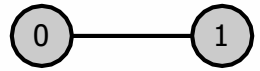
Das 'Aufklappen' des N-dimensionalen Würfels ergibt die KMAP.

Gegenüberliegende Randfelder sind benachbart!

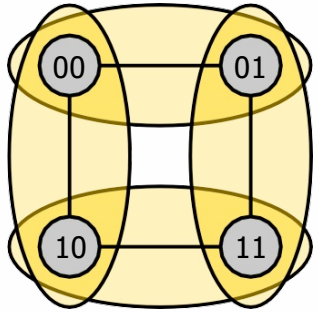


Die KMAP hat so viele Nachbarschaften, wie der Würfel Kanten hat.

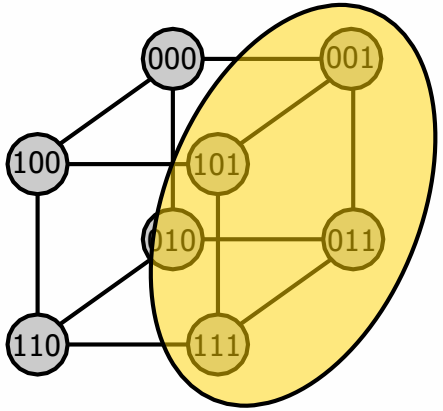
N=1:
Codewort a = 0 oder 1



N=2:
Codewort ba = 00,01,10,11

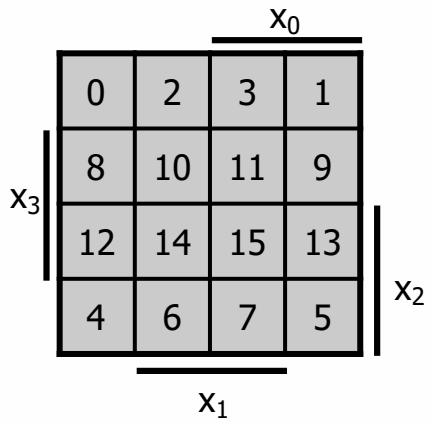


N=3:
Codewort cba = 000,001,...



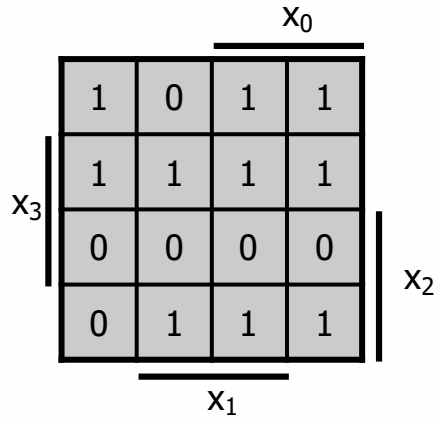
Nummerierung

Die Felder können nummeriert werden, wenn man die Eingangsvariable (hier $x_3\, x_2\, x_1\, x_0$) als Binärzahl auf fast



Eine beliebige Funktion der N Variablen wird durch das 0/1 Muster in der Tafel festgelegt.
(Für N=4 gibt es 2^{16} = 65536 verschiedene Funktionen)

In einer kompakten Schreibweise führt man die Eingangswerte auf, für die in der Tafel eine 1 steht und schreibt z.B. $F = \sum(0,1,3,5,6,7,8,9,10,11)$

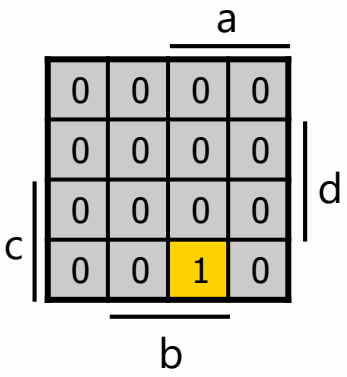


UND und ODER in KMAPS – N Variablen

Die UND-Verknüpfung aller **N Variablen** (direkt oder invertiert) selektiert **EIN Feld**.

Man nennt diese 16 möglichen Ausdrücke **MINTERME**, weil sie in der Tafel die minimale Fläche einnehmen.

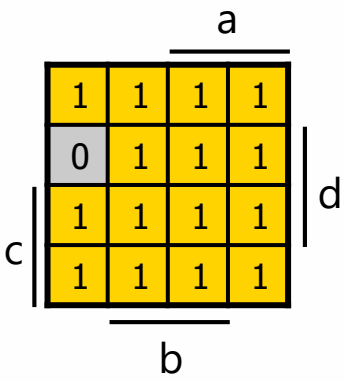
$$Y = abc\bar{d}$$



Die ODER-Verknüpfung der N Variablen selektiert **ALLE AUSSER EINEM Feld**.

Man nennt einen solchen Ausdruck **MAXTERM**, weil er in der Tafel die maximale Fläche einnimmt.

$$Y = a + b + c + \bar{d}$$

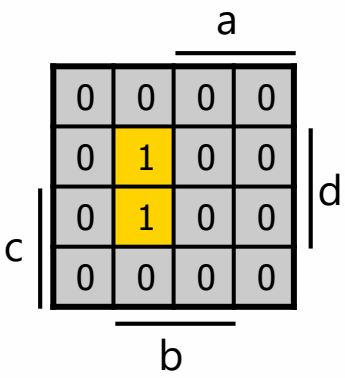


UND und ODER in KMAPS – N-x Variablen

Die UND-Verknüpfung von **N-1 Variablen** bilden einen **1x2 oder 2x1 Block**. (Dies gilt nur, wenn die Variablen Gray-codiert sind!).

Im rechten Beispiel ist. $Y = \bar{a}bcd + \bar{a}b\bar{c}d = \bar{a}bd(c + \bar{c}) = \bar{a}bd$

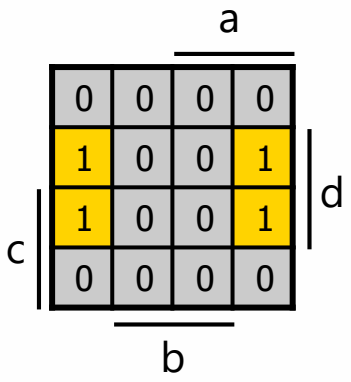
$Y = \bar{a}bd$



Die UND-Verknüpfung von **N-2 Variablen** bilden einen **Block aus 4 Zellen** (1x4, 4x1 oder 2x2), etc...

Blöcke können auch periodisch über den Rand hinaus gehen

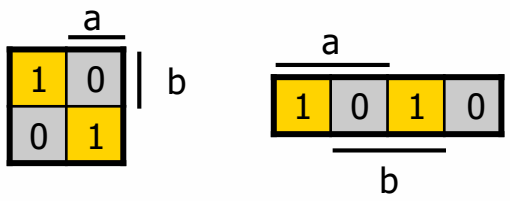
$Y = \bar{b}d$



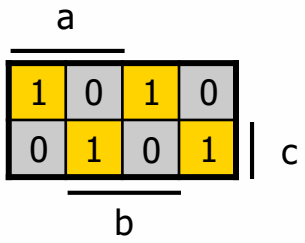
Entsprechendes gilt für das ODER von weniger als N Variablen (video)

XOR in KMAPS

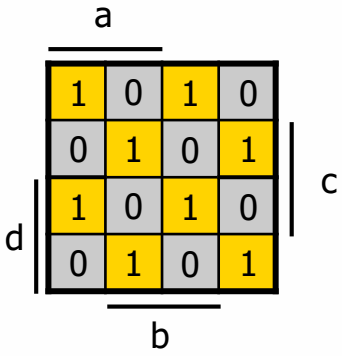
$a \oplus b = a\bar{b} + \bar{a}b$



$a \oplus b \oplus c = (a \oplus c)\bar{c} + \overline{(a \oplus b)}c$



$a \oplus b \oplus c \oplus d$



XOR und XNOR sind durch viele diagonale Einzelfelder gekennzeichnet.

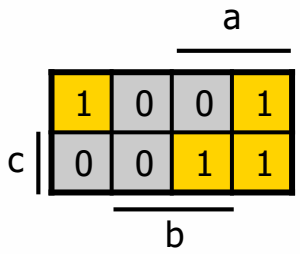
(a ⊕ b ⊕ c ⊕ d ist 1 bei 1000, 0100, 0010 und 0001 aber **auch** bei 1011,…)

Sie können daher nicht vereinfacht warden…

Auffinden des Komplements einer Funktion

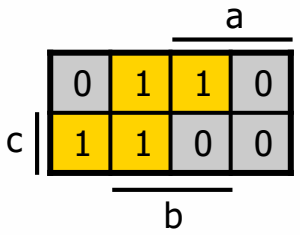
Gegeben sei die Funktion F = ac + b'c'

KMAP von F:



Die KMAP der inversen Funktion F' findet man direkt durch Vertauschen von 0 und 1:

KMAP von F':



F' = a'c + bc'

Mit den Regeln der Schaltalgebra ist das deutlich umständlicher:

$$F = ac + \bar{b}\bar{c}$$

$$\bar{F} = \overline{ac + \bar{b}\bar{c}} = \overline{ac}\,\overline{\bar{b}\bar{c}} = (\bar{a} + \bar{c})(b + c)$$

$$= \bar{a}b + \bar{a}c + \bar{c}b + \bar{c}c = \bar{a}b + \bar{a}c + \bar{c}b$$

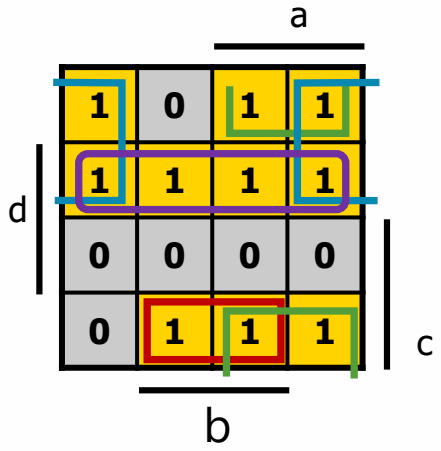
überflüssiger Term

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

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

Auffinden des einfachsten Ausdruck für eine Funktion

$F = \sum(0,1,3,5,6,7,8,9,10,11)$



$F = \bar{c}d + \bar{b}\bar{c} + a\bar{d} + bc\bar{d}$

Primimplikanten können nicht mit anderen Implikanten vereinfacht werden

(Implikant $a\bar{c}$ ist redundant)

Ist eine Funktion über ein Muster in der KMAP gegeben, so sucht man **eine vollständige Überdeckung** der Einsen mit **möglichst großen** rechteckigen Blöcken...

Die Funktion ist dann die ODER-Verknüpfung all dieser Implikanten

Es kann sein, dass eine Teilmenge der Implikante zur vollständigen Überdeckung ausreicht.

Die Implikanten in einer Minimalgleichung müssen alle Primimplikanten sein. Andernfalls könnten sie kombiniert werden, um die Anzahl der Literale zu verringern.

Binary Coded Decimals

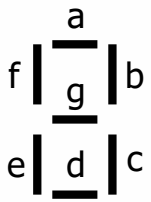
Es kommt oft vor, dass nicht alle Eingangskombinationen vorkommen können.

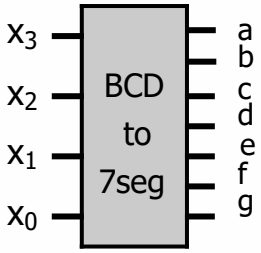
Dadurch bleiben Felder in der KMAP 'leer', sie werden meist mit ‚x' markiert.

Diese Felder können so mit 1 oder 0 belegt werden, dass die Funktion möglichst einfach wird, i.e. dass die Einsen mit möglichst wenigen, möglichst großen Blöcken überdeckt werden können.

	x_3	x_2	x_1	x_0	*e*
0	0	0	0	0	*1*
1	0	0	0	1	*0*
2	0	0	1	0	*1*
3	0	0	1	1	*0*
4	0	1	0	0	*0*
5	0	1	0	1	*0*
6	0	1	1	0	*1*
7	0	1	1	1	*0*
8	1	0	0	0	*1*
9	1	0	0	1	*0*
	1	0	1	0	-
	1	0	1	1	-
	1	1	0	0	-
	1	1	0	1	-
	1	1	1	0	-
	1	1	1	1	-

Ansteuerung einer Siebensegment-Anzeige mit eine BCD-Zahl:

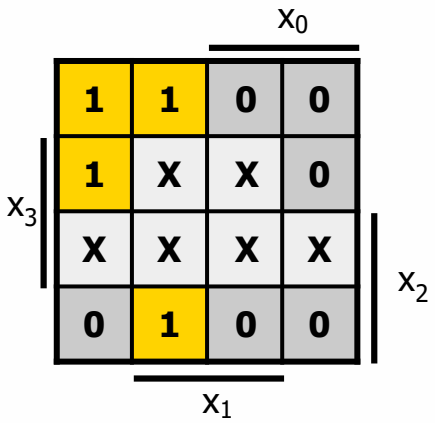




KMAPs mit leeren Feldern (don't care)

Man schreibt z.B. für **Segment *e***: $\mathrm{E} = \sum m(0,2,6,8) + \sum d(10,11,12,13,14,15)$

Dabei steht 'm' für Minterm und 'd' für *don't care*

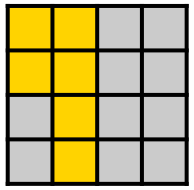


Abdeckung nur der Einsen erfordert 2 Felder a 1x2, d.h zwei Primterme mit 3 Termen:

$$E = X_0'X_1'X_2' + X_0'X_1X_3'$$

Ersetzt man zwei der don't care Felder durch Einsen, so kann man die Terme vereinfachen:

$$E = X_0'X_2' + X_0'X_1$$



Je mehr don't cares es gibt, desto stärkere Vereinfachungen sind möglich. Man sollte die Funktion also nicht unnötig einschränken.

KMAPs Zusammenfassung

Vorteile:

Karnaugh-Maps bieten eine einfache grafische Methode zur Minimierung boolescher Funktionen.

Sie können für bis zu sechs Variablen verwendet werden und eignen sich daher für kleine bis mittelgroße Funktionen.

Karnaugh Maps helfen bei der Visualisierung der logischen Beziehungen zwischen den Variablen, was zum Verständnis der Funktion und ihres Verhaltens beitragen kann.

Sie können in der Wahrheitstabelle mit "Don't Cares" (X) umgehen, wodurch die Funktion weiter vereinfacht werden kann.

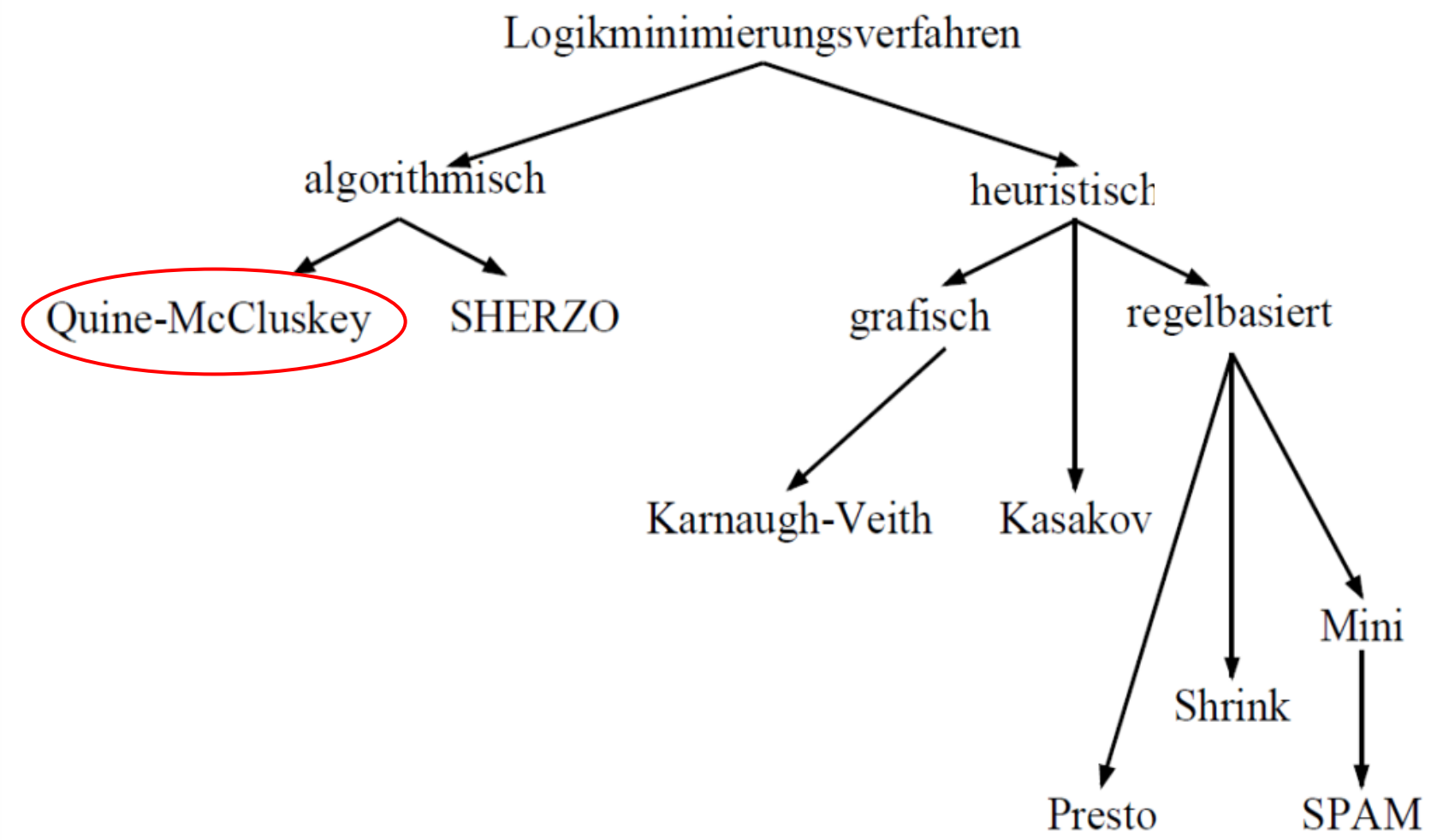
Nachteile:

Karnaugh-Maps können bei größeren Funktionen zeitaufwändig zu konstruieren und auszufüllen sein.

Bei der Verwendung von Karnaugh-Maps kann es mehrere gültige Lösungen geben, was zu Verwirrung und Ineffizienz führen kann.

Karnaugh-Maps eignen sich nicht für Funktionen mit sich überschneidenden Mintermen oder für Funktionen mit bestimmten Symmetrien.

Übersicht


Logikminimierungsverfahren
algorithmisch
heuristisch
Quine-McCluskey
SHERZO
grafisch
regelbasiert
Karnaugh-Veith
Kasakov
Mini
Shrink
Presto
SPAM

Minimierung nach Quine-McCluskey

Für mehr als 4 Eingangsvariable wird das grafische Verfahren der KMAPs unübersichtlich.

Ein Algorithmus zum Auffinden der einfachsten Summendarstellung mit Primtermen ist das Quine-McCluskey Verfahren.

Es unterteilt sich in 2 Schritte:

Auffinden aller Primterme

Bestimmung der minimalen Überdeckung (Verwerfen von redundanten Primtermen)

Auffinden aller Primterme:

1. Schritt:

- Schreibe die Binärdarstellung aller Einsen in die erste Spalte einer Tabelle.
- **Don't cares** werden zunächst **als Einsen** gewertet!
- Dabei werden Gruppen von Termen mit keiner Eins, einer 1, zwei Einsen etc. gebildet. (Benachbarte Gruppen enthalten also benachbarte Felder in der KMAP)

Minimierung nach Quine-McCluskey

2. Schritt:

- Für jeweils aufeinanderfolgenden Gruppenpaare werden alle Elemente paarweise verglichen. (→ exp. Aufwand!)
- Wenn ein Elementpaar sich nur in einer Stelle unterscheidet, existiert ein Term, der diese Stelle nicht enthält.
- Alle solche Terme werden Gruppenweise in der Form 10-0 etc. in die nächste Spalte notiert.
- Die Terme in der ersten Spalte, die eine Nachbarschaft aufweisen, werden mit einem Haken als nicht-prim markiert.
- Der 2. Schritt wird spaltenweise wiederholt, bis sich keine Paare mehr finden. Die '-' Zeichen müssen dabei beim Vergleich an der gleichen Stelle sitzen.

3. Schritt:

- Die Terme ohne Haken sind die Primterme.
- In ihrer Menge werden mit einer Tabelle die zur Überdeckung notwendigen Primterme gesucht.

Quine-McCluskey Beispiel

i	x_3	x_2	x_1	x_0	y
0	0	0	0	0	1
1	0	0	0	1	-
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
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	1
11	1	0	1	1	-
12	1	1	0	0	0
13	1	1	0	1	-
14	1	1	1	0	-
15	1	1	1	1	1

Definition:

$$Y = \sum m(0,2,5,8,10,15) + \sum d(1,11,13,14)$$

Zu binären Vektoren übersetzen

1's: (0, 0, 0, 0)
(0, 0, 1, 0)
(0, 1, 0, 1)
(1, 0, 0, 0)
(1, 0, 1, 0)
(1, 1, 1, 1)

d's: (0, 0, 0, 1)
(1, 0, 1, 1)
(1, 1, 0, 1)
(1, 1, 1, 0)

Darstellung mit Indexgruppen

Bei diesem Verfahren werden die Nachbarschaften durch die Indizes einer Eingangskombination ermittelt. Da die Indizes im Dezimalsystem angegeben werden, sind die Nachbarschaften nicht mehr unmittelbar zu erkennen, wie es in der linearen Vektorschreibweise möglich ist. Die möglichen Eingangs-kombinationen werden aufsteigend mit Indexnummern (z.B. I_j mit $0 \leq j \leq 15$ für 4 Variable) versehen.

Es gilt, dass benachbarte Belegungen sich in der Anzahl der Bits um genau 1 unterscheiden. Damit werden **Indexgruppen** gebildet mit gleicher "1"-Anzahl, hier z. B. für 4 Variablen:

Indexgruppe	Index	
0	0	(0, 0, 0, 0)
1	1, 2, 4, 8	.
2	3, 5, 6, 9, 10, 12	.
3	7, 11, 13, 14	.
4	15	(1, 1, 1, 1)

Darstellung mit Indexgruppen

Es gilt weiterhin, dass bei benachbarten Belegungen die Differenz der Indizes eine Potenz von 2 ist die Potenzzahl gibt dabei die Bitstelle an:

z. B. $I_{15} - I_7 = 8 = 2^3$

	x_3	x_2	x_1	x_0		
die Worte	1	1	1	1	$\hat{=}$	I_{15}
und	0	1	1	1	$\hat{=}$	I_7

unterscheiden sich in x_3

Zur Minimierung werden die in der Schaltfunktion gegebenen 1-Belegungen der Indizes in die Indexgruppe einsortiert und systematisch Differenzen innerhalb benachbarter Gruppen gesucht, die a) eine 2er Potenz ergeben und b) größer als Null sind, d. h. Differenzen aller Indizes k, l für die gilt:

$$k \in I_{j+1}, l \in I_j \quad k > l \quad \text{und} \quad k - l = 2^p$$

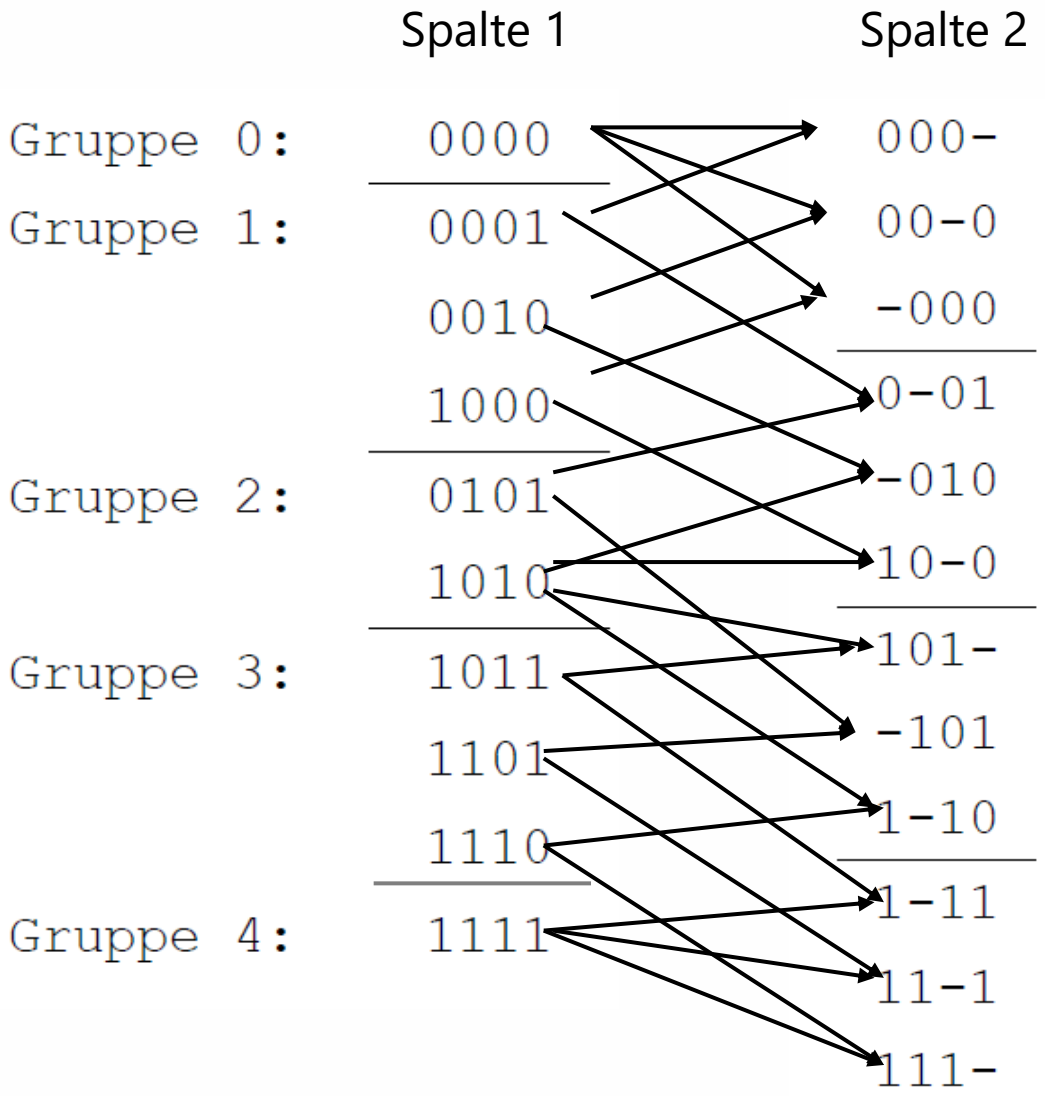
Diese werden in eine so genannte **erste Kürzungstabelle** eingetragen. Die Einträge entsprechen allen möglichen 2er-Schleifen im Karnaugh-Veith-Diagramm.

Zweite Schritt

Binären Vektoren in Gruppen zu sortieren, wobei jedes Element einer Gruppe dieselbe Anzahl von Einsen hat.

Die zweite Spalte enthält alle Kombinationen aus der ersten Spalte.

Die Terme aus der ersten Spalte, die nicht zu einem neuen Term kombiniert werden können, werden als nicht benutzt markiert (kommt hier nicht vor).



Zweite Schritt (noch einmal)

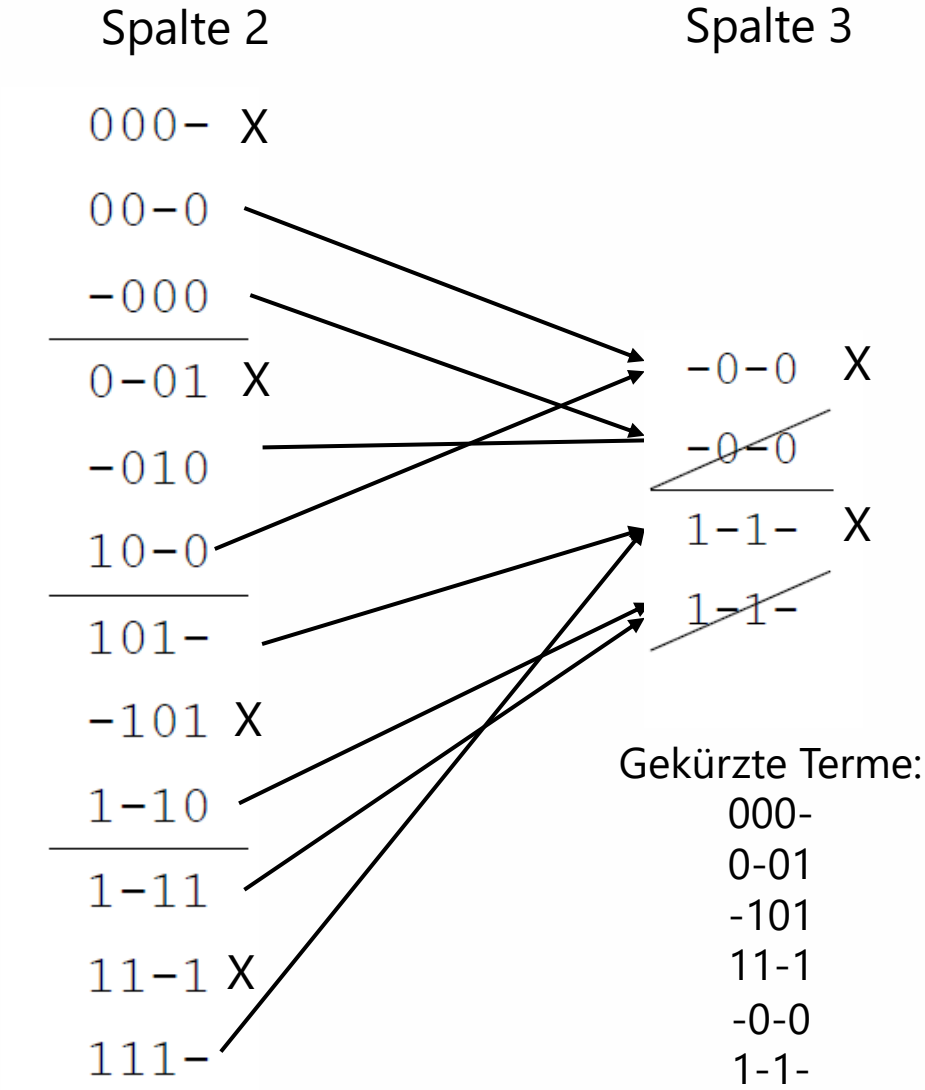
Vorherige Schritte werden wiederholt

In der ersten Spalte gibt es hier 4 Terme die nicht markiert sind:

000-, 0-01, -101 und 11-1.

Die Kürzung der Terme endet hier, da aus der zweiten Spalte keine weiteren Kombinationen gebildet werden können.

Die übrigbleibenden Terme sind die sogenannte Primimplikanten



Primimplikantentabelle

Der nächste Schritt ist es, herauszufinden welche Minterme durch welche Primimplikanten abgedeckt werden. Dazu wird eine sogenannte **Primimplikantentabelle** aufgestellt. Jeder notwendige Minterm (keine don´t cares) bekommt eine Spalte und jeder Primimplikant eine Zeile.

Aus der Kürzung:
Primimplikanten

000-
0-01
-101
11-1
-0-0
1-1-

	0000	0010	0101	1000	1010	1111
000-						
0-01						
-101						
11-1						
-0-0						
1-1-						

Minterm markieren

Es werden alle Stellen in der Tabelle markiert, an denen ein Primimplikant einen Minterm abdeckt.
die Striche im Primimplikanten bedeuten don't care

z. B. der Primimplikant -0-0 die Minterme 0000, 0010, 1000 und 1010 ab.

Ziel ist es mit möglichst wenig Primimplikanten alle Minterme abzudecken

	0000	0010	0101	1000	1010	1111
000-	X					
0-01			X			
-101			X			
11-1						X
-0-0	*X*	**X**		**X**	*X*	
1-1-					X	X

Suche nach wichtige Primimplikanten

Wird in der Tabelle ein Minterm nur durch einen ganz bestimmten Primimplikanten abgedeckt, dann spricht man von einem sogenannten Kernprimimplikanten. Dies ist erkennbar dadurch, dass in der Spalte nur ein Kreuz ist. Diese Kernprimimplikanten sind unverzichtbar. Hier gibt es nur einen Kernprimimplikanten -0-0, der für die Minterme 0010 und 1000 unverzichtbar ist.

	0000	0010	0101	1000	1010	1111
000-	X					
0-01			X			
-101			X			
11-1						X
-0-0	*X*	**X**		**X**	*X*	
1-1-					X	X

Suche nach wichtige Primimplikanten

Der Kernprimimplikant -0-0 deckt aber auch die Minterme 0000 und 1010 ab
→ der Primimplikant 000- redundant ist.

Für die Minterme 0101 und 1111 muss eine Auswahl getroffen werden.
Der Minterm 1111 wird sowohl vom Primimplikanten 11-1 als auch von 1-1- abgedeckt.

	0000	0010	0101	1000	1010	1111	
~~000-~~	X						redundant
0-01			X				
-101			X				
11-1						X	←
-0-0	X	X		X	X		
1-1-					X	X	←

Suche nach wichtige Primimplikanten

Deutet man die Knotenbedingung so, dass ein Minterm mit weniger Eingängen der Einfachere ist so fällt Entscheidung für 1-1-.

Die beiden Primimplikanten, die den Minterm 0101 abdecken, sind gleichberechtigt, so dass es hier zwei gleichberechtigte Lösungen gibt.

	0000	0010	0101	1000	1010	1111	
~~000~~	~~X~~						redundant
0-01			X				
-101			X				
~~11-1~~						~~X~~	redundant
-0-0	*X*	**X**		**X**	*X*		
1-1-					X	X	

Suche nach wichtige Primimplikanten

Die Minterme, die übrig bleiben, führen zu den folgenden beiden Lösungen:

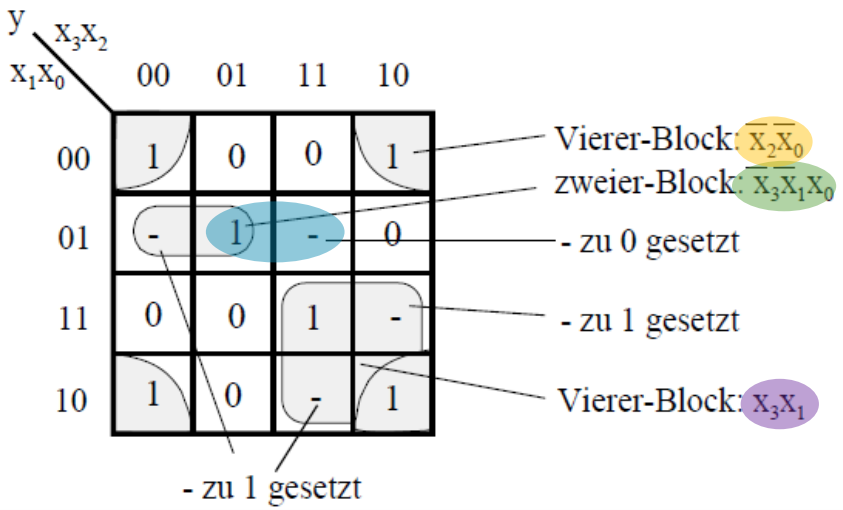
$$Y = \overline{X_3}\,\overline{X_1}X_0 \vee \overline{X_2}\,\overline{X_0} \vee X_3X_1$$

$$Y = X_2\overline{X_1}X_0 \vee \overline{X_2}\,\overline{X_0} \vee X_3X_1$$

	0000	0010	0101	1000	1010	1111		
000-	X						redundant	
0-01			X				wählbar	$\overline{x}_3\overline{x}_1x_0$
-101			X				wählbar	$x_2\overline{x}_1x_0$
11-1						X	eingeschränkt wählbar	
-0-0	*X*	**X**		**X**	*X*		unverzichtbar	$\overline{x}_2\overline{x}_0$
1-1-					X	X	eingeschränkt wählbar	x_3x_1

Vergleich mit Karnaugh-Veith

	0000	0010	0101	1000	1010	1111		
000-	X						redundant	
0-01			X				wählbar	$\bar{x}_3\bar{x}_1x_0$
-101			X				wählbar	$x_2\bar{x}_1x_0$
11-1						X	eingeschränkt wählbar	
-0-0	*X*	**X**		**X**	*X*		unverzichtbar	$\bar{x}_2\bar{x}_0$
1-1-					X	X	eingeschränkt wählbar	x_3x_1



Quine-McCluskey Zusammenfassung

Vorteile:

Der Quine-McCluskey-Algorithmus ist ein algorithmisches Verfahren zur Minimierung boolescher Funktionen, das heißt, er kann für Funktionen beliebiger Größe und Komplexität verwendet werden.

Es handelt sich um eine systematische und effiziente Methode, die das Auffinden der minimalen Summe von Produkten (oder des Produkts von Summen) garantiert.

Der Quine-McCluskey-Algorithmus kann mit "don't cares" (X) in der Wahrheitstabelle umgehen, wodurch die Funktion weiter vereinfacht werden kann.

Nachteile:

Der Quine-McCluskey-Algorithmus kann eine große Anzahl von Primzahlimplikanten erzeugen, was zu Ineffizienz bei der Auswahl der wesentlichen Primzahlimplikanten führen kann.

Der Algorithmus kann mehrere Primzahlimplikanten mit demselben Gültigkeitsbereich erzeugen, was zu Redundanz im endgültigen Ausdruck führt.

Andere Algorithmen können diese Probleme lösen (Video: Petrick's Methode)