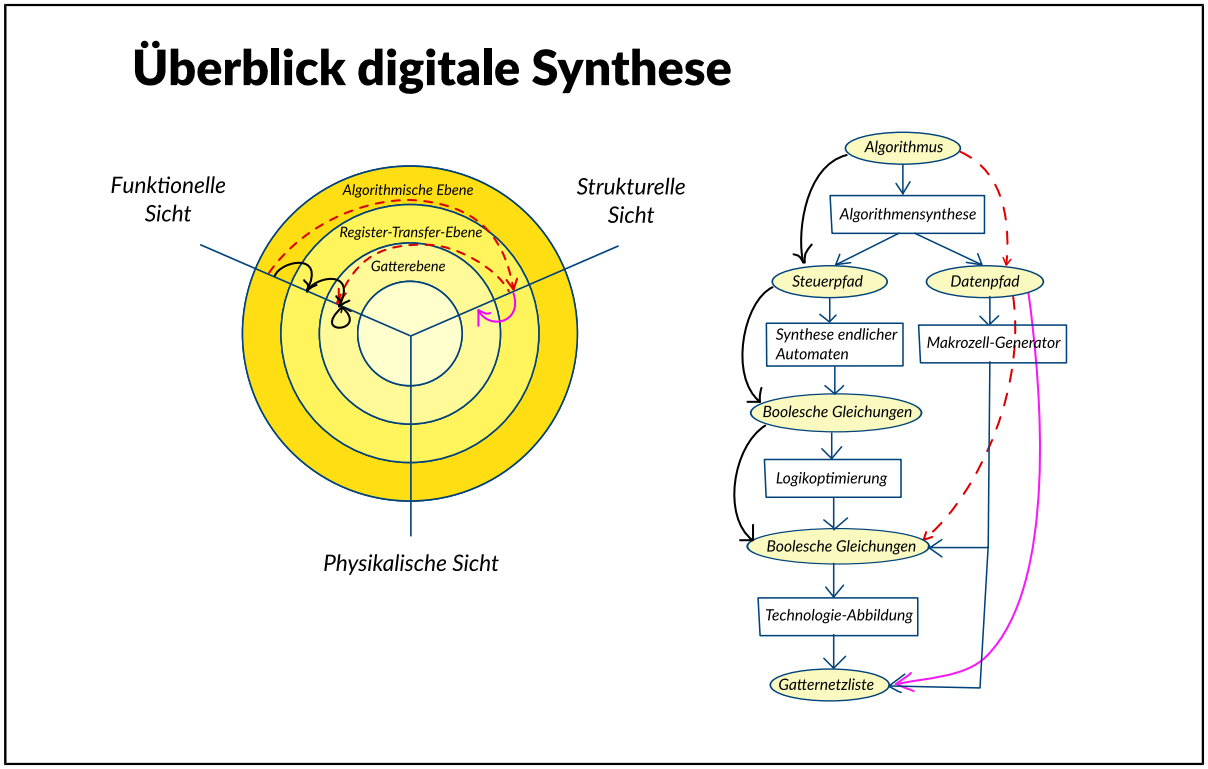
Electronic Design Automation (EDA)

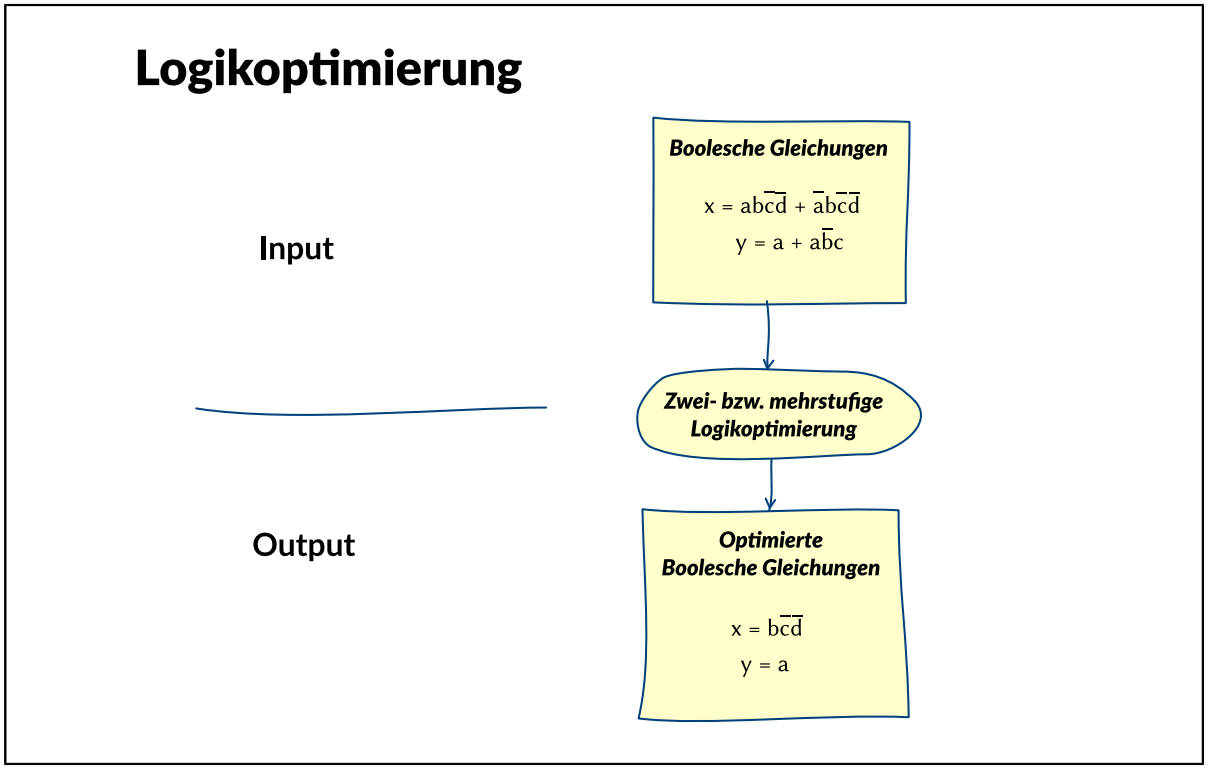
Logikoptimierung

Date of generation: 30.5.2023

Logikoptimierung: Überblick digitale Synthese



Logikoptimierung: Logikoptimierung

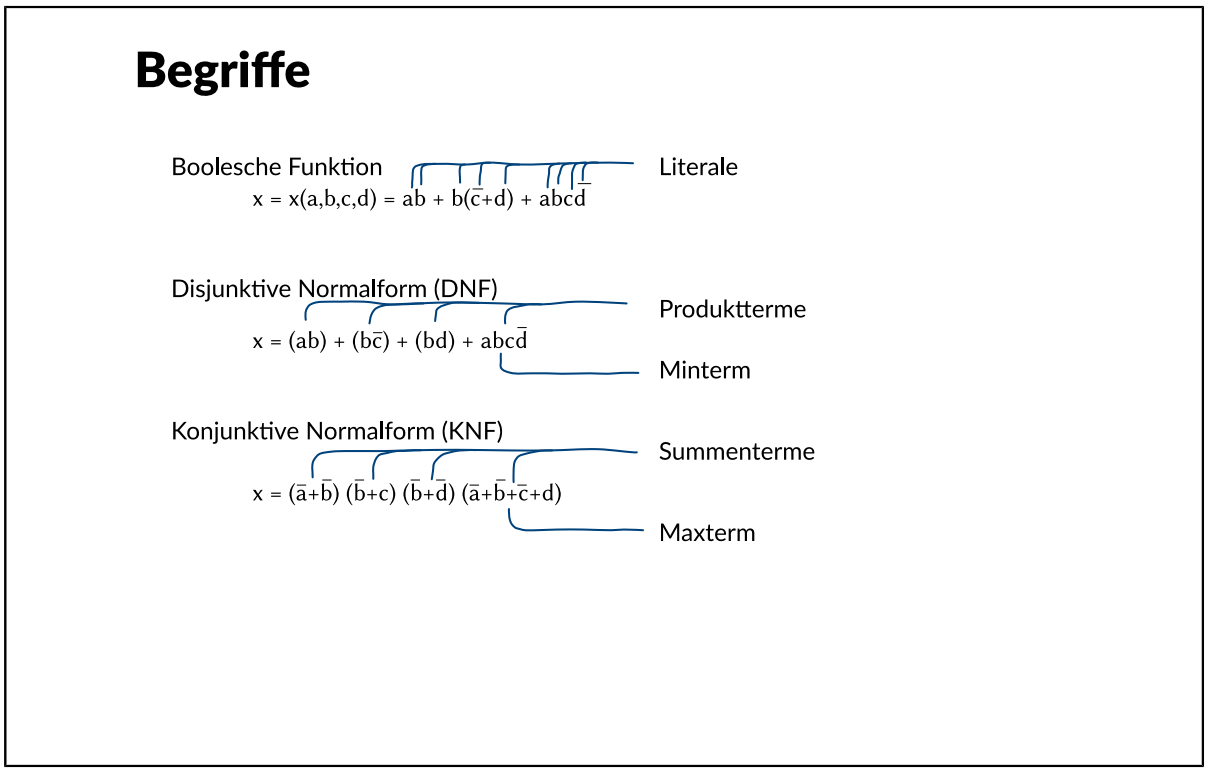


Bei der Logikoptimierung muss je nach Realisierungsform der Schaltung zwischen zwei Verfahren unterschieden werden:

- Mehrstufige Logikoptimierung für zellenbasierte Implementierungen wie Standardzellen, Gate Arrays oder FPGAs.
- Zweistufige Logikminimierung für zweistufige Implementierungen wie PLAs oder PLDs.

Wie stets heißen die Optimierungskriterien Kosten und Performance bzw. Fläche und Verzögerungszeit. Auf der Gatterebene wird die Fläche durch die Anzahl der auftretenden Literale (und damit später über die Anzahl und Größe der benötigten Gatter) und die Verzögerungszeit durch die Anzahl der logischen Stufen repräsentiert. Zweistufige Realisierungen sind deshalb auf dieser Abstraktionsebene die schnellst möglichen.

Logikoptimierung: Begriffe



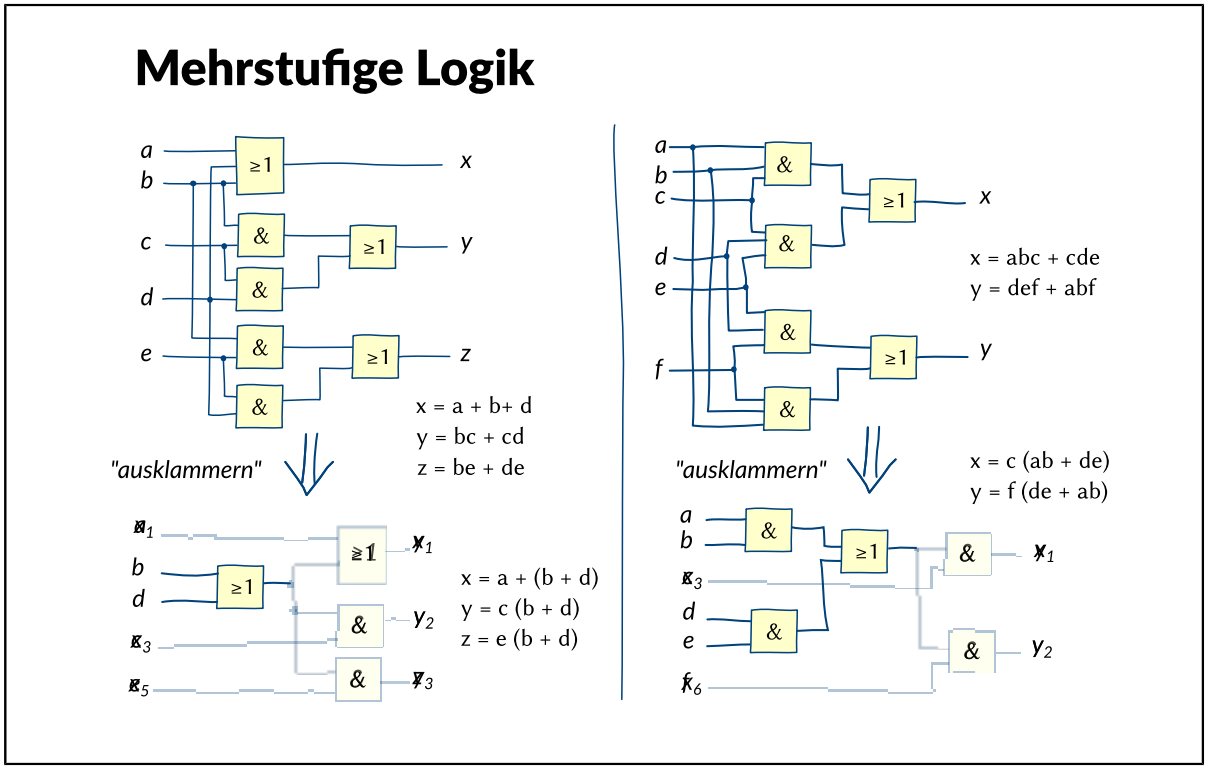
Produktterme bzw. Summenterme, die alle Variablen der booleschen Funktion invertiert oder nicht invertiert enthalten, heißen Minterme bzw. Maxterme. Enthält eine DNF bzw. KNF nur Minterme bzw. Maxterme, heißt sie kanonische DNF bzw. KNF.

Ein Produktterm p heißt Implikant für eine Funktion f, wenn gilt: p=1 => f=1. Ein Implikant heißt Primimplikant, wenn aus dem Produktterm kein Literal entfernt werden kann, ohne dass er seine Eigenschaft Implikant zu sein, verliert.

In der disjunktiven Normalform sind alle Produktterme Implikanten. In einer minimierten DNF sind alle Implikanten Primimplikanten.

Um die Schreibweise zu vereinfachen, wird im Folgenden ähnlich wie bei der Multiplikation das UND-Zeichen in den Gleichungen weggelassen.

Logikoptimierung: Mehrstufige Logik



Bei mehrstufiger Logik wird meist nach den Kosten (Anzahl der Gatter) optimiert, da vor der Technologie-Abbildung für die Performance noch keine genauen Informationen über das zeitliche Verhalten von Gattern vorliegen. Wie bereits erwähnt, wird stattdessen die Anzahl der Logikstufen zwischen Eingang und Ausgang des booleschen Netzwerkes als Maß für die Verzögerung verwendet.

Bei der Optimierung wird das System der booleschen Gleichungen umgeformt, um die Anzahl der Literale zu reduzieren und damit Fläche zu sparen. In erster Linie geht es um die Erkennung von gemeinsamen Unterfunktionen, die nur einmal implementiert werden müssen. Diese Unterfunktionen werden als Zwischenvariablen ("ausklammern") eingeführt. Die Abbildung auf der linken Seite zeigt die Gatterdarstellung folgenden Gleichungssystems vor und nach (unten) dem Einfügen einer Zwischenvariablen (b+d).

x = a+b+d= a+(b+d)

y = bc+cd = c(b+d)

z= be+de = e(b+d)

Dieser Schritt senkt die Kosten, da er Fläche spart (mehrfache Implementierungen werden vermieden), kann aber auch die Anzahl der Stufen erhöhen, wodurch die Schaltung langsamer und damit die Performance schlechter wird. Ein Beispiel dafür zeigt das folgende Gleichungssystem und die rechte Seite der Abbildung.

x = abc+cde = c(ab+de)

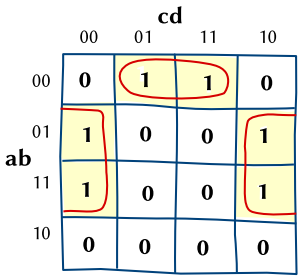
y = def+abf = f(ab+de)

Logikoptimierung: Zweistufige Logik: Exakte Verfahren

Zweistufige Logik: Exakte Verfahren

Exakte graphische Lösung mit Karnaugh Diagrammen

1) Disjunktive Normalform (DNF) graphisch darstellen.



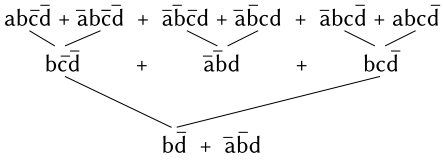
2) "Einsen zusammenfassen" (Absorptionsgesetz)

$\bar{a}\bar{b}d + b\bar{d}$

Exakte rechnerische Lösung mit Quine-McCluskey-Algorithmus

Wesentliche Schritte:
1) Berechnung aller Primimplikanten.
2) Extraktion der minimalen disjunktiven Form der Funktion

Zwei Implikanten lassen sich zusammenfassen, wenn sie sich an nur einer Position unterscheiden (wie beim Karnaugh-Diagramm).



Quine-McCluskey-Algorithmus hat exponentielle Komplexität $O(a^n)$!

Das Ziel ist, eine Summe von Produkttermen zu finden, die die gleiche Ausgangsfunktion beschreibt und möglichst wenig Implikanten und Literale enthält, um die Kosten zu minimieren.

Für die Minimierung zweistufiger Logik gibt es exakte Verfahren und Heuristiken.

Für wenige Eingänge lässt sich eine exakte Lösung mit dem Karnaugh-Diagramm finden. Dies ist eine 2-dimensionale Anordnung einer Funktionstabelle. Jedes Feld enthält genau einen Implikanten (Minterm). Bei benachbarten Feldern ändert sich die dazugehörige Eingangskombination nur bei einer Eingangsvariablen. Dadurch lassen sich benachbarte Felder zu einfacheren Implikanten zusammenfassen. Die Erklärung liefert das Absorptionsgesetz:

$$ab+a\bar{b}=a(b+\bar{b})=a$$

Das exakte Verfahren nach Quine und McCluskey lässt sich leichter in einem Programm implementieren als die Karnaugh-Diagramm-Minimierung. Es garantiert zwar ein Minimum, hat aber eine exponentielle Komplexität (O(n) = (3^n)/n) und ist daher nur für wenige Eingänge praktikabel. Im Wesentlichen basiert es auf zwei Schritten:

- Berechnung aller Primimplikanten: Zwei Implikanten lassen sich zusammenfassen, wenn sie sich an nur einer Position unterscheiden (wie beim Karnaugh-Diagramm).
- Extraktion der minimalen disjunktiven Form der Funktion.

Logikoptimierung: ... Heuristische Verfahren

Zweistufige Logik: Heuristische Verfahren

Bekannte exakte Verfahren sind zu langsam für komplexe Probleme, daher werden heuristische Optimierungsverfahren einsetzt (z.B. Espresso)

- Betrachtung von Untermengen von Implikanten
- Iteratives Vorgehen
- Rückschritte erlauben, um lokale Minima zu vermeiden

Das Espresso-Verfahren enthält drei elementare Schritte

- Implikanten expandieren
- Implikanten reduzieren
- Implikanten streichen

Aufgrund der Komplexität exakter Verfahren sind heuristische Verfahren gebräuchlich. Diese basieren zwar auf exakten Verfahren, betrachten jedoch nur eine Untermenge der Implikanten und optimieren iterativ. Das bekannteste Verfahren ist in dem Programm Espresso implementiert. Die Iterationen enthalten drei elementare Schritte: Implikanten expandieren, reduzieren oder streichen. Beim Expandieren und Streichen werden Literale entfernt und beim Reduzieren werden Literale hinzugefügt.

Logikoptimierung: ... Expansion/Reduktion

Zweistufige Logik: Expansion und Reduktion

Ausdruck: $a + \bar{a}\,b$

	a	b	$a + \bar{a}b$
	0	0	0
$\bar{a}\,b$	0	1	1
$a\,\bar{b}$	1	0	1
$a\,b$	1	1	1

Expansion: Hinzufügen von Mintermen, um dann zu vereinfachen.

Zum Ausdruck $a+\bar{a}b$ können die Minterme ab und/oder $a\bar{b}$ ohne Veränderung der Wahrheitstabelle hinzugefügt werden, da sie durch den Implikanten a bereits abgedeckt sind.

Durch hinzufügen von ab kann der Ausdruck vereinfacht werden:

$a + \bar{a}b + ab$
$a + b$

Reduktion ist der umgekehrte Vorgang. Es wird nicht vereinfacht, sondern erweitert, um lokale Minima zu vermeiden.

Expandieren

Durch das Expandieren einer Funktion durch weitere Implikanten ohne die Funktion und damit die Wahrheitstabelle zu verändern, kann sich die Möglichkeit ergeben Literale zu entfernen. Zum Beispiel kann durch Expansion folgender Ausdruck minimiert werden:

$a+\bar{a}b$

Die Wahrheitstabelle hierfür ist

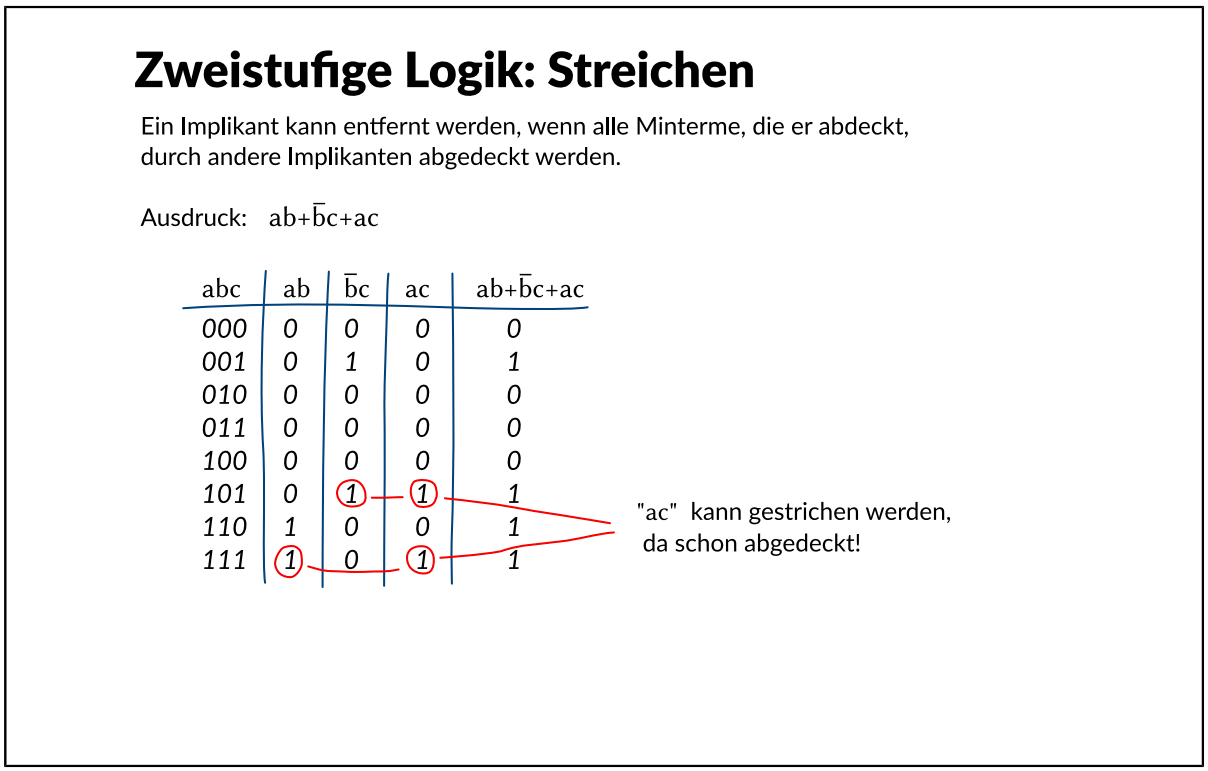
a	b	$a+\bar{a}b$
0	0	0
0	1	1
1	0	1
1	1	1

Zu diesem Ausdruck gehören also die Minterme $\bar{a}b$, $a\bar{b}$ und ab. Es können Minterme hinzugefügt werden, die durch vorhandene Implikanten bereits abgedeckt werden, z.B. können in dem obigen Ausdruck Minterme hinzugefügt werden, die durch den Implikanten a abgedeckt werden, wie z.B. ab oder $a\bar{b}$. Wird der Ausdruck mit ab expandiert, vereinfacht sich der Ausdruck wegen $\bar{a}b+ab=b$ zu $a+b$. Beide Ausdrücke stellen die gleiche Funktion dar.

Reduktion

Reduktion dagegen liefert vom Ausdruck $a+b$ den Ausdruck $a+\bar{a}b$. Die Reduktion erhöht zwar die Kosten der Funktion, indem sie Literale hinzufügt, kann aber notwendig sein, um lokale Minima bei der Optimierung verlassen zu können.

Logikoptimierung: ... Streichen



Streichen

Ein Implikant kann entfernt werden, wenn alle Minterme, die er abdeckt, durch andere Implikanten abgedeckt werden. Z. B. kann in dem folgenden Ausdruck der Implikant ac gestrichen werden:

$$ab+\bar{b}c+ac$$

Die Funktionstabelle dieses Ausdrucks verdeutlicht, wieso dies möglich ist:

a b c	ab	$\bar{b}c$	ac	$ab+\bar{b}c+ac$
0 0 0	0	0	0	0
0 0 1	0	1	0	1
0 1 1	0	0	0	0
1 0 0	0	0	0	0
1 0 1	0	1	1	1
1 1 0	1	0	0	1
1 1 1	1	0	1	1

Die Minterme, die durch ac abgedeckt werden ($a\bar{b}c$ und abc), werden bereits durch die Implikanten ab und $\bar{b}c$ abgedeckt.