15 NMOS-Gatter-Synthese

Im hier beginnenden Kapitel wird die Realisierung komplexer logischer Funktionen durch Schalternetze behandelt, wie beispielsweise durch Transistornetze (Bild 15.2 a). Die dabei durchlaufenen Abstraktionsebenen zeigt Fig. 15.1. Die Umsetzung einfacher logischer Funktionen in einzelne Gatter ist bereits in früheren Kapiteln (z. B. über Schaltkreisfamilien) dieses Textes gezeigt worden. Das hier beginnende Kapitel widmet sich insbesondere der optimierenden Synthese, wobei aus einer einzigen logischen Funktion ein aus mehreren Gattern bestehendes Schaltnetz synthetisiert wird. Bei dem in Abschnitt 15.1 eingeführten Verfahren (Bild 15.2 b) entsteht eine Gatter-Kaskade (siehe Bild 15.2 c).

Im folgenden wird der Entwurfsprozeß veranschaulicht über das Beispiel eines statischen nMOS-Schaltnetzes aus einer gegebenen logischen Funktion (Bild 15.2). Man kann folgende zwei Klassen von Entwurfsprozessen unterscheiden:

- heuristisch, d.h. durch sukzessive Äquivalenz-Transformation
- Serien-parallel-Analyse der logischen Funktion,
- algorithmisch, z.B. mit dem Algorithmus von Culiney-Muroga.

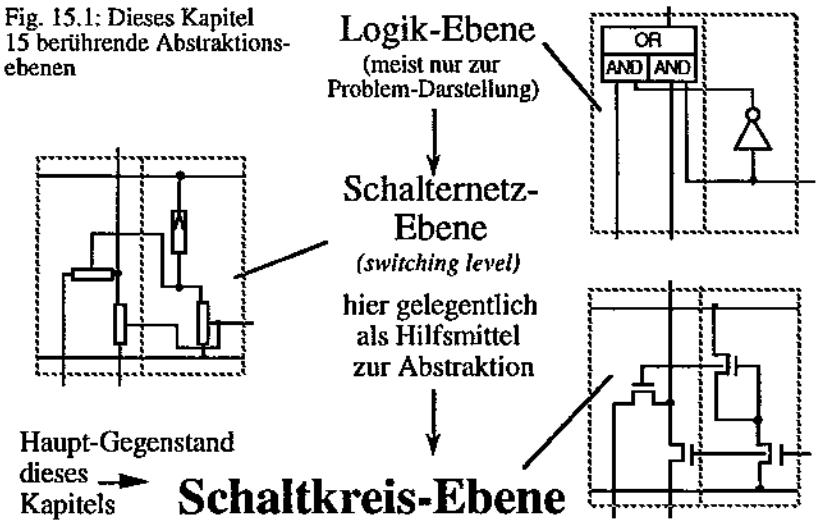


Fig. 15.1: Dieses Kapitel 15 berührende Abstraktionsebenen

Im folgenden Abschnitt 15.1 wird zu letzterer Kategorie der Culiney-Muroga-Algorithmus eingeführt, der aus einer Boole'schen Funktion ein Transistornetz in Verhältnislogik synthetisiert mit einer minimalen Anzahl von Gattern und ohne negierte Eingänge. Durch minimale Gatterzahl wird die Anzahl der Flächen-ineffizienten Pullups minimiert. Auch die Vermeidung zusätzlicher Inverter am Eingang (da keine negierten Eingänge) trägt zu dieser Minimierung bei. Wegen der Trivialität der Wandlung in eine statische CMOS-Schaltung kann dieses Verfahren leicht auch zur Synthese statischer CMOS-Logik erweitert werden.

15.1 Das Verfahren nach Culiney-Muroga

Nach dem schon erwähnten relativ heuristischen Verfahren zum Entwurf einer logischen Schaltung führen wir nun als Alternative eine stärker algorithmische Vorgehensweise ein. Es wird ein Entwurfsverfahren vorgestellt, das 1982 von Muroga [6] beschrieben wurde. Es liefert

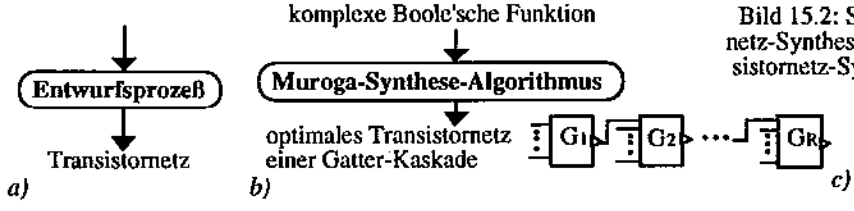


Bild 15.2: Schalternetz-Synthese (Transistornetz-Synthese)

V	w(V)	n(V)	#(V)
1101	3	4	13
0000000	0	7	0
111011	5	6	49
111111	6	6	63

Bild 15.3. Beispiele zu Definitionen 1 - 3.

als Endergebnis eine nMOS-Schaltung mit der minimalen Anzahl von Gattern ohne invertierte Eingänge. Aufgrund dieser Tatsache ist dieses Verfahren nützlich im Rahmen der nMOS-Schaltungsentwicklung. Der Beweis ist kompliziert, weshalb auf dessen Behandlung hier verzichtet wird.

Das Verfahren eignet sich auch für den Entwurf komplementärer CMOS-Schaltungen wenn eine zusätzliche Phase an das bekannte Verfahren angehängt wird. Der Algorithmus wird zunächst anhand eines Beispiels eingeführt.

15.1.1 Begriffs-Definitionen

Zuerst müssen jedoch einige Definitionen erfolgen, die zum Verständnis beitragen.

Definition 1: Ein **"Binärvektor"** oder **Vektor V** ist eine Wertekombination aus n Schaltvariablen mit $V = (x_1, x_2, \ldots x_n)$. Beispiele für $n = 4$ sind: 0110, 1010, 1110

Definition 2: Die Länge n(V) eines Vektors V gibt die Anzahl seiner Bits an.

Definition 3: Das **"Gewicht"** w(V) eines Vektors V gibt die Anzahl der Einsen im Vektor V an. Für w gilt: Der Maximalwert ist w = n(V) und der Minimalwert ist w = 0, wobei n(V) die Länge von V (Anzahl der Bits) angibt. Beispiele (Bild 15.3):

Definition 4: Der Dezimalkode #(V) eines Vektors V der Länge n(V) ergibt sich aus

$$\#(V) = \sum_{i=1}^{n} x_i 2^{n-i}$$

Definition 5: Die **"Hamming-Distanz"** HD(V1,V2) zweier Vektoren V1 und V2 gibt an, um wieviele Bit-Werte sich diese Vektoren unterscheiden. Alle Vektorpaare mit einer Hammingdistanz von Eins bezeichnet man als "direkte Hamming-Nachbarn". Beispiele sind in Bild 15.4 zu finden.

15.1.2 Die Phasen des Muroga-Algorithmus

Der Algorithmus von Culiney-Muroga wird in zwei Phasen (A und B) durchgeführt. Das Flußdiagramm von Bild 15.5 soll als Anschauungshilfe dienen.

Vektoren	Hamming-Distanz	Veranschaulichung
010101 110000	3	0 1 0 1 0 1 1 1 0 0 0 0
01010 01000	1	0 1 0 1 0 0 1 0 0 0
0101010 0101010	0	

direkte Hamming-Nachbarn

Bild 15.4. Beispiele zu Definition 5.

In Phase A wird als erstes aus der Funktionstabelle ein Gitterdiagramm ähnlich eines Hyperkubus aufgebaut und im Anschluß daran werden sogenannte Markierungsvektoren errechnet. In der folgenden Phase B werden in mehreren Schritten sogenannte Minimalvektoren (Labels) L(V) ermittelt, aus denen dann die einzelnen Gatter für das Schaltnetz abgeleitet werden.

Wir erklären im folgenden den Algorithmus mit Hilfe eines Beispiels mit n(V) = 4. Die logische Funktion, anhand derer der Algorithmus veranschaulicht wird, ist durch Funktionstabelle in Bild 15.6 gegeben. Diese Funktion soll per Hardware realisiert werden, d.h. es soll ein nMOS-Schaltnetz erzeugt werden, das diese Funktion realisiert.

15.1.3 Phase A: Gitterdiagramm mit Markierungsvektoren

In Phase A (s. a. Bild 15.5) wird als erstes die Funktionstabelle in eine graphische Darstellung umgewandelt: in ein Gitterdiagramm ähnlich eines Hyperkubus. Im Anschluß daran werden sogenannte Markierungsvektoren errechnet und in dieses Gitterdiagramm eingetragen.

15.1.3.1 Phase A 1: Erstellung eines Muroga-Gitterdiagramms

Der erste Schritt besteht darin, ein Gitterdiagramm nach Art eines ungerichteten Graphen aufzubauen (Bild 15.7) nach den wie folgt angegebenen Konstruktionsregeln. Dabei wird jeder Vektor V gemäß Funktionstabelle durch genau einen Knoten repräsentiert. Ein schwarzer Knoten für einen gegebenen Vektor V bedeutet f(V) = 0, ein weißer Knoten bedeutet f(V) = 1 und ein schraffierter Knoten zeigt, daß der Funktionswert des Binärvektors ein "don't care" ist (vgl. Bild 15.7). Die Punkte im Netz werden zeilenweise angeordnet, wobei in einer Zeile alle Vektoren mit gleichem Gewicht w angeordnet sind.

Die einzelnen Zeilen sind wieder nach fallendem Gewicht von oben nach unten geordnet. In der obersten Zeile findet man den Knoten mit dem Maximalgewicht, welches in unserem Beispiel w = 4 ist. Die Vektoren in jeder Zeile darunter haben immer ein um 1 vermindertes Gewicht w gegenüber der darüberliegenden Zeile. Innerhalb einer Zeile sind die Punkte nach dem dezimalen Wert von V in absteigender Folge von links nach rechts angeordnet, dabei werden die Vektoren als nicht negative binär kodierte ganze Zahlen interpretiert.

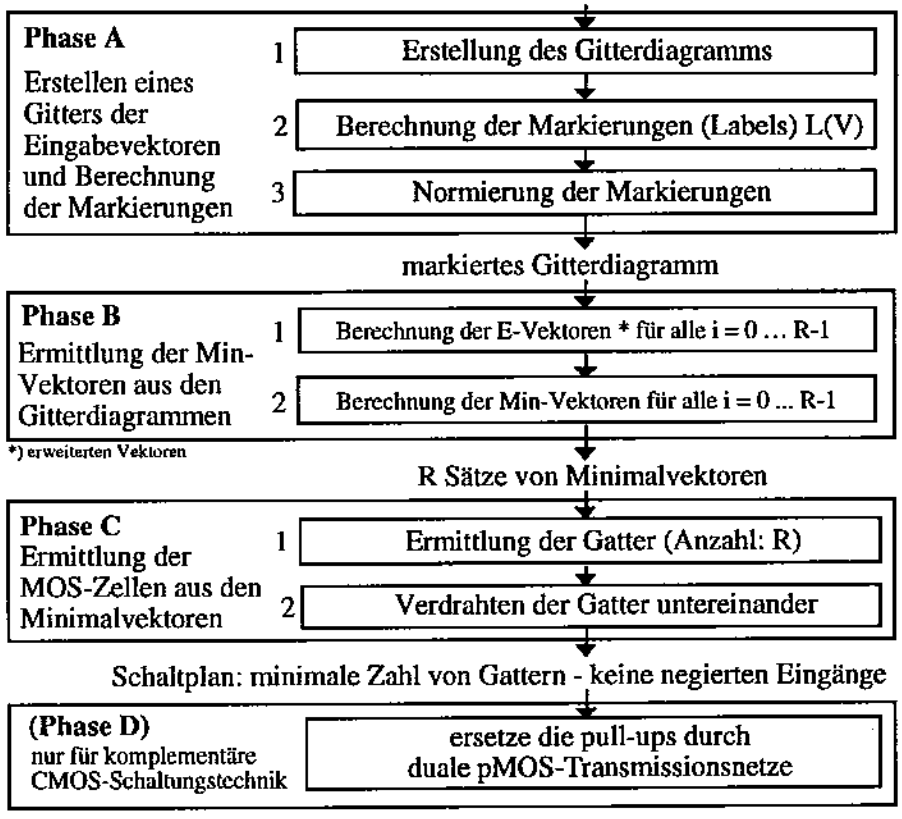


Bild 15.5: Flußdiagramm zum Culiney-Muroga-Verfahren

Die Kanten zwischen den einzelnen Knoten sind nach dem Kriterium einer Hamming-Distanz von 1 gelegt, d.h. zwei Knoten sind dann und nur dann miteinander verbunden, wenn die ihnen zugeordneten Vektoren sich in genau einem Bit voneinander unterscheiden. Beispiel in Bild 15.7: Die Vektoren #6 und #4 unterscheiden sich im dritten Bit von links (x_3) voneinander. Im Gitterdiagramm schreiben wir die Vektoren V in Klammern, damit wir diese deutlicher von den später zu ermittelnden Markierungen L(V) unterscheiden können.

15.1.3.2 Phase A 2: Ermittlungen der Markierungen L(V)

In Phase A 2 wird zu jedem Knoten und somit zu jedem Vektor V ein sogenannter Markierungsvektor (Label) L(V) ermittelt. Das dazugehörige Flußdiagramm, welches die Prozedur dieser Phase verdeutlicht, ist in Bild 15.8 abgebildet. Die Markierung des "schwersten" Vek-

Zeilen-#(V)	Vektor V = (x1, x2, x3, x4)	f (V)
0	0000	1
1	0001	1
2	0010	1
3	0011	1
4	0100	0
5	0101	0
6	0110	d
7	0111	1
8	1000	0
9	1001	1
10	1010	1
11	1011	0
12	1100	0
13	1101	1
14	1110	0
15	1111	1

Bild 15.6: Beispiel-Aufgabe zur Veranschaulichung des Muroga-Verfahrens

tors, des einzigen Vektors mit dem maximalen Gewicht, ist ein Trivialfall. Für diesen einzigen Vektor des höchsten Gewichtes (oben) ergibt sich L als ein einziges Bit, welches aus dem Funktionswert ermittelt wird. Für f(V)=1 ist L(V)=1 andernfalls ist L(V)=0 (vgl. Bild 15.8).

Definition 6: Vektor V' ist dann **"oberer Hamming-Nachbar"** des Vektors V, wenn w(V') = w(V)+1 und V und V' die Hamming-Distanz 1 haben, d.h. HD(V',V) = 1.

Nach Ermittlung der Markierung für den "schwersten" Vektor mit w = n wird bei den übrigen Gewichten w < n d.h. w = n-1, ...0 wie folgt verfahren. Zuerst wird das geringstwertige Bit (LSB) ermittelt. Dann werden eventuell dem LSB voranzustellende weitere Bits ermittelt.

Ermittlung des Markierungsvektor L(V)

Erster Schritt: das jeweilige LSB (Least Significant Bit) des zu ermittelnden Markierungsvektors ergibt sich ganz einfach aus dem Funktionswert des Knotens. Ist dieser Wert ein "don't care" (d), so wird das Maximum der Markierungsvektoren L(V') aller oberen Hamming-Nachbarn in den zu ermittelnden Markierungsvektor L(V) kopiert und der zweite Schritt entfällt.

Der zweite Schritt dieser Phase für die nach links folgenden Bits von L(V) ist etwas komplexer. Wir müssen dazu den Markierungsvektor L(V) jedes Knotens der aktuellen Zeile (Arbeitszeile) jeweils mit den Markierungen L(V') aller direkten Hamming-Nachbarn der darüberliegenden Zeile (Referenz-Zeile) vergleichen. Diesen Vergleich müssen wir in der Ar-

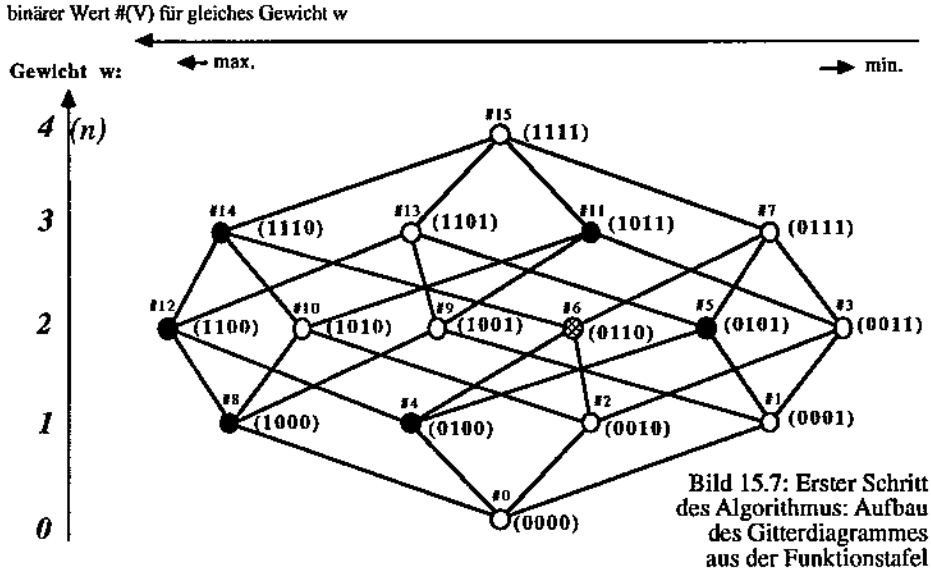
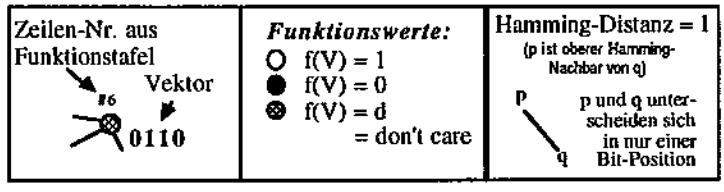


Bild 15.7: Erster Schritt des Algorithmus: Aufbau des Gitterdiagrammes aus der Funktionstafel



Aufstellen des Hamming-Graphen der Funktion

beitszeile mit dem zweithöchsten Gewicht beginnen und sukzessive bis zum untersten Knoten (mit w = 0) durchführen. Wir gehen Zeile für Zeile vor.

Gilt L(V) < L(V'), so muß L(V) solange durch Voranstellen weiterer Bits verlängert werden, bis das kleinste L(V) gefunden ist, für das gilt: L(V) ≥ L(V'). Bei Verlängerung von L(V) darf das LSB nicht verändert werden, d.h. die Verlängerung kann nur durch Vornanfügen einzelner Bits an die linke Seite von L(V) durchgeführt werden. Letztlich muß man für jeden direkten Hamming-Nachbarn L(V') die kleinste Markierung L(V) finden, für die L(V) ≥ L(V') gilt.

In Bild 15.9 ist diese Phase für unser Beispiel veranschaulicht. Der Vektor V steht in Klammern, während die Markierung L(V) nicht eingeklammert ist. Bild 15.9 a veranschaulicht die Ermittlung von L(V) für den "schwersten" Vektor mit V=(1111). Für diesen gilt eine Sonderregelung (Bild 15.9 a). Wir verwenden die Wahrheitstafel aus Bild 15.8 rechts oben. Der Knoten #15 hat den Funktionswert f(V)=1 und kann mit keinem oberen Hamming-Nachbarn verglichen werden. Also bekommt der Markierungsvektor den Wert 1 zugeordnet.

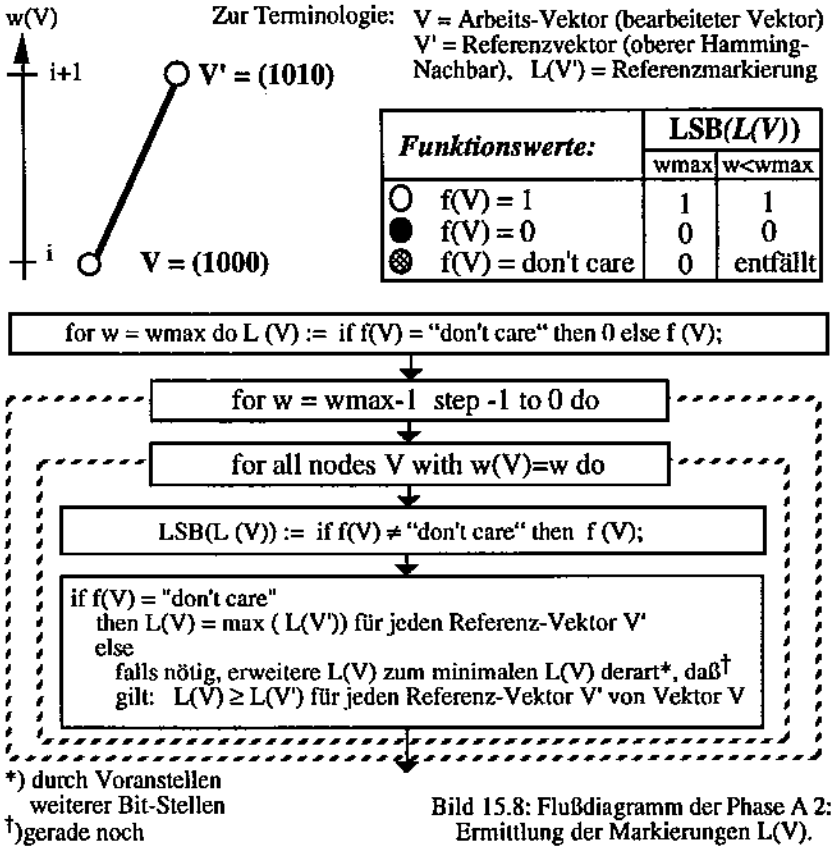


Bild 15.8: Flußdiagramm der Phase A 2:
Ermittlung der Markierungen L(V).

Im Beispiel nach Bild 15.9b hat die Arbeits-Zeile das Gewicht w=n-1=3 und die Knoten #14 (Bild a) und #11 (Bild b) haben ein L(V)<L(V') (0<1); deshalb muß L(V) zu 10 ergänzt werden. Bei Knoten #13 und #7 bleibt der Markierungsvektor unverändert. In Bild 15.9c) ergibt sich der Markierungsvektor von Knoten #6 aus dem größten Markierungsvektor aller oberen Hamming-Nachbarn, da der Funktionswert "don't care" ist (LSB wird nicht durch den Funktionswert bestimmt). Für die restlichen Knoten werden die Markierungen wie in a) und b) ermittelt. Bild 15.10 zeigt dann das resultierende Gitterdiagramm mit allen Markierungsvektoren.

15.1.3.3 Phase A 3: Normierung der Markierungen

Jetzt in Phase A3 werden die Markierungsvektoren "normiert", d.h. auf eine einheitliche Wortlänge R gebracht. Diese Markierungs-Länge R wird durch die Länge des Markierungsvektors

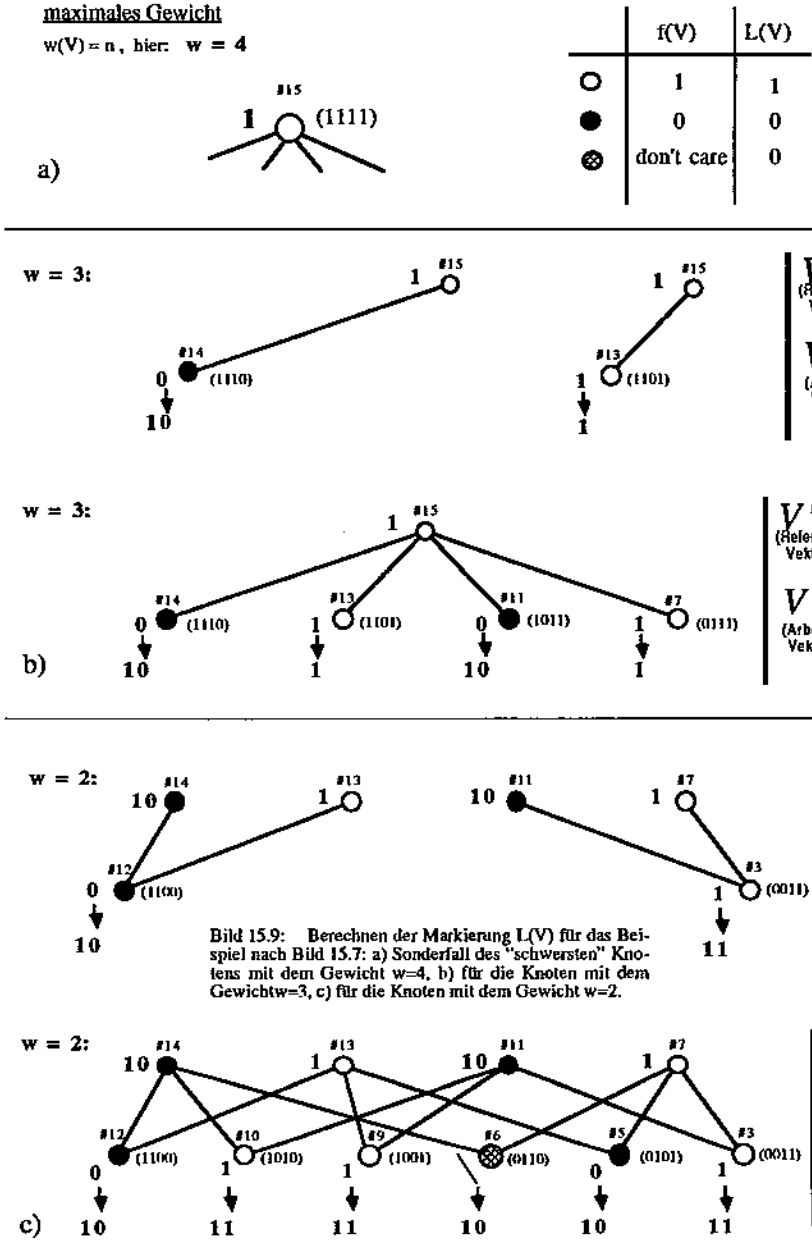


Bild 15.9: Berechnen der Markierung L(V) für das Beispiel nach Bild 15.7: a) Sonderfall des "schwersten" Knotens mit dem Gewicht w=4, b) für die Knoten mit dem Gewichtw=3, c) für die Knoten mit dem Gewicht w=2.

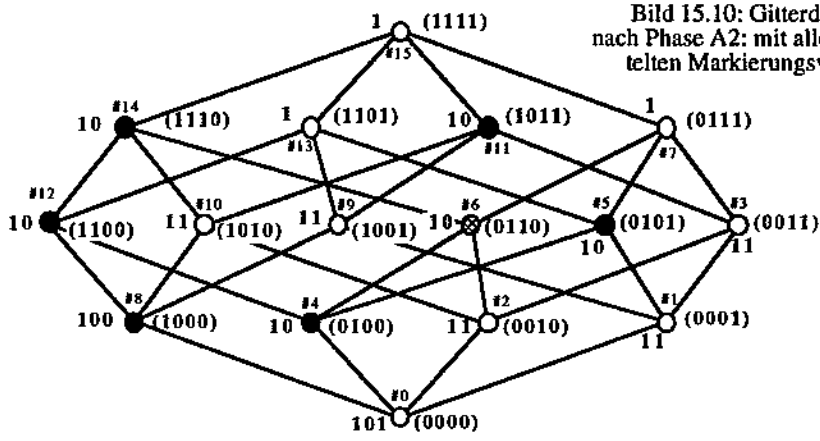


Bild 15.10: Gitterdiagramm nach Phase A2: mit allen ermittelten Markierungsvektoren.

des "leichtesten" Vektors V = (0000) vorgegeben (vgl. Bild 15.10). Alle L(V) deren Bitkombination weniger als R lang ist, werden durch führende Nullen auf die Länge von R erweitert. Für unser Beispiel ist dies in Bild 15.11 durchgeführt worden. An dieser Stelle kann schon gesagt werden, daß R die Anzahl der später erzeugten Gatter angibt.

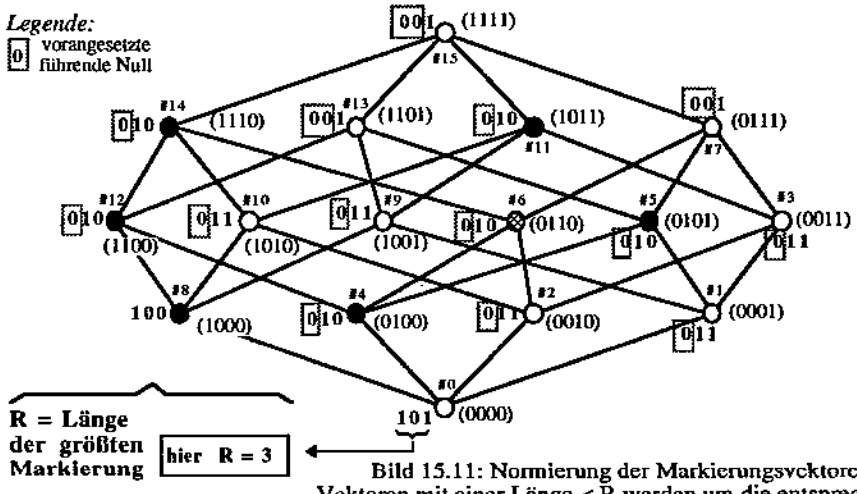


Bild 15.11: Normierung der Markierungsvektoren; alle Vektoren mit einer Länge < R werden um die entsprechende Anzahl führender Nullen verlängert auf R (die hinzugefügten Nullen sind jeweils durch ein Rechteck gekennzeichnet).

15.1.4 Phase B: Berechnung der Minimal-Vektoren

Phase A ist hiermit abgeschlossen. Aus dem in Phase A gewonnenen Gitterdiagramm werden nun R verschiedene neue Gitterdiagramme abgeleitet. Bevor wir jetzt die Phase B durchführen, müssen zuerst noch einmal einige Definition erfolgen. Es werden Dominanz-Relationen und spezielle "Vergleichs-Relationen" zwischen Vektoren $A = (a_1,...,a_k)$ und $B = (b_1,...,b_k)$ definiert, sowie ein spezieller "Minimalvektor" in einer Menge vergleichbarer Vektoren. Zur Präzisierung führen wir drei weitere Definitionen ein: $\geqslant$ für "dominant über" $\not\geqslant$ für "nicht dominant über" und $\lneqq$ für "unvergleichbar mit".

Definition 7: Eine **Dominanzrelation** $\geqslant$ ist gegeben durch $A \geqslant B$ (A ***dominiert*** über B), wenn für alle k gilt: $a_k \geq b_k$.

Wir sagen $A \lneqq B$ (A ist ***unvergleichbar*** mit B) wenn gilt: $A \not\geqslant B$ und $B \not\geqslant A$. Hierbei hat $A \not\geqslant B$ die Bedeutung: A ***dominiert nicht*** über B

Beispiel 1: für A = (001101) und B = (000101) gilt $A \geqslant B$.
Beispiel 2: für A = (001101) und B = (000111) gilt $A \lneqq B$.

Definition 8: Ein **"Minimalvektor"** ist gegeben, wenn: a) kein vergleichbarer anderer Vektor existiert, oder: b) alle anderen vergleichbaren Vektoren dominant über diesen Vektor sind.

Wegen der Sonderbehandlung von Knoten mit dem maximalen Gewicht $w = w_{max}$ wurde das Flußdiagramm in Bild 15.5 weiter verfeinert, wie Bild 15.16 zeigt. Die ursprüngliche Phase B1 wurde in B1a (für i = 0) und B1b (für i = 1, ... R-1) zerlegt, desgleichen die Phase B2 (für i = 0) in B2a und B2b (für i = 1, ... R-1).

15.1.4.1 Phase B.1: Berechnung der E-Vektoren (erweiterten Vektoren)

In der Phase B.1 werden zunächst *"erweiterte Vektoren"* **E(V)** ermittelt und aus diesen dann letztlich die *"Min-Vektoren"* **M(V)** (Minimalvektoren), die zur Ermittlung der Gattergleichungen in der Phase B2 benötigt werden.

Aus dem in Phase A gewonnen Gitternetz (Bild 15.11) werden nun im Phase B1 **in R Schritten (i = 0, 1, ... R-1)** R separate neue Gitternetze erstellt (Bild 15.13 (für i = 0), Bild 15.12 (für i = 1), Bild 15.15 (für i = 2 = R-1), aus denen in Phase B2 dann die sogenannten Minimalvektoren oder Min-Vektoren ermittelt werden (unterer Rand von Bild 15.13 bis Bild 15.15). Der erweiterte Vektor E ergibt sich durch Anhängen von i Stellen u_j an den Vektor V. Dabei ergibt sich $E = V$ für i = 0, $E = (V, u_1)$ für i = 1, und $E = (V, u_1, u_2)$ für i = 2.

Die neuen Gitterdiagramme haben die gleiche Form wie bisher, die den Knoten zugeordneten Werte sind jedoch andere als bisher. Zunächst werden aus den Vektoren V erweiterte Vektoren E durch (hinten) Anhängen weiterer Bitstellen gebildet. Aus jedem Vektor V entsteht ein er-

15.1.4 Phase B: Berechnung der Minimal-Vektoren

Phase A ist hiermit abgeschlossen. Aus dem in Phase A gewonnenen Gitterdiagramm werden nun R verschiedene neue Gitterdiagramme abgeleitet. Bevor wir jetzt die Phase B durchführen, müssen zuerst noch einmal einige Definition erfolgen. Es werden Dominanz-Relationen und spezielle "Vergleichs-Relationen" zwischen Vektoren $A = (a_1,...,a_k)$ und $B = (b_1,...,b_k)$ definiert, sowie ein spezieller "Minimalvektor" in einer Menge vergleichbarer Vektoren. Zur Präzisierung führen wir drei weitere Definitionen ein: $\trianglerighteq$ für "dominant über" $\not\trianglerighteq$ für "nicht dominant über" und $\trianglerighteq\!\!\!\!\!\!\not\quad\not\trianglerighteq$ für "unvergleichbar mit".

Definition 7: Eine **Dominanzrelation** $\trianglerighteq$ ist gegeben durch $A \trianglerighteq B$ (A ***dominiert*** über B), wenn für alle k gilt: $a_k \geq b_k$.

Wir sagen $A \trianglerighteq\!\!\!\!\!\!\not\quad\not\trianglerighteq B$ (A ist ***unvergleichbar*** mit B) wenn gilt: $A \not\trianglerighteq B$ und $B \not\trianglerighteq A$. Hierbei hat $A \not\trianglerighteq B$ die Bedeutung: A ***dominiert nicht*** über B

Beispiel 1: für A = (001101) und B = (000101) gilt $A \trianglerighteq B$.
Beispiel 2: für A = (001101) und B = (000111) gilt $A \trianglerighteq\!\!\!\!\!\!\not\quad\not\trianglerighteq B$.

Definition 8: Ein **"Minimalvektor"** ist gegeben, wenn: a) kein vergleichbarer anderer Vektor existiert, oder: b) alle anderen vergleichbaren Vektoren dominant über diesen Vektor sind.

Wegen der Sonderbehandlung von Knoten mit dem maximalen Gewicht $w = w_{max}$ wurde das Flußdiagramm in Bild 15.5 weiter verfeinert, wie Bild 15.16 zeigt. Die ursprüngliche Phase B1 wurde in B1a (für i = 0) und B1b (für i = 1, ... R-1) zerlegt, desgleichen die Phase B2 (für i = 0) in B2a und B2b (für i = 1, ... R-1).

15.1.4.1 Phase B.1: Berechnung der E-Vektoren (erweiterten Vektoren)

In der Phase B.1 werden zunächst *"erweiterte Vektoren"* **E(V)** ermittelt und aus diesen dann letztlich die *"Min-Vektoren"* **M(V)** (Minimalvektoren), die zur Ermittlung der Gattergleichungen in der Phase B2 benötigt werden.

Aus dem in Phase A gewonnen Gitternetz (Bild 15.11) werden nun im Phase B1 **in R Schritten (i = 0, 1, ... R-1)** R separate neue Gitternetze erstellt (Bild 15.13 (für i = 0), Bild 15.12 (für i = 1), Bild 15.15 (für i = 2 = R-1), aus denen in Phase B2 dann die sogenannten Minimalvektoren oder Min-Vektoren ermittelt werden (unterer Rand von Bild 15.13 bis Bild 15.15). Der erweiterte Vektor E ergibt sich durch Anhängen von i Stellen u_j an den Vektor V. Dabei ergibt sich $E = V$ für $i = 0$, $E = (V, u_1)$ für $i = 1$, und $E = (V, u_1, u_2)$ für $i = 2$.

Die neuen Gitterdiagramme haben die gleiche Form wie bisher, die den Knoten zugeordneten Werte sind jedoch andere als bisher. Zunächst werden aus den Vektoren V erweiterte Vektoren E durch (hinten) Anhängen weiterer Bitstellen gebildet. Aus jedem Vektor V entsteht ein er-

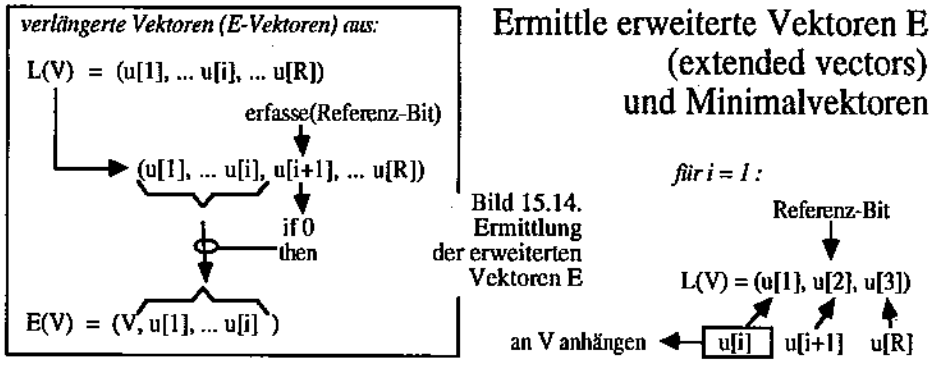


Bild 15.14. Ermittlung der erweiterten Vektoren E

weiterter Vektor E = (V, u_1,...,u_i). (Aus diesen erweiterten Vektoren E werden dann später in Phase B.2 die "Minimalvektoren" ermittelt.) Bild 15.14 veranschaulicht Formate des Verfahrens.

Der Fall i = 0 ist dabei ein Trivialfall, denn hier erfolgt keine eigentliche Verlängerung, sodaß hier stets E=V. Dieser Fall wird am Schluß dieses Abschnitts behandelt (siehe auch Bild 15.16).

1. Wir erstellen für jedes i ein neues Gitterdiagramm (Kopie des Gitterdiagrammes aus Phase A). Für i=1 erhalten wir das Beispiel nach Bild 15.12.

2. Für jedes i betrachten wir uns die Position i+1 ("Referenz-Bit") aller Markierungsvektoren L(V) = (u_1,...,u_R). Dies ist beispielsweise die Position 2 für i=1 (vgl. Bild 15.12). Hat das Bit dieser Position i+1 den Wert 0, wählen wir den dazugehörigen Variablenvektor V aus (vgl. z.B. den kurzen Pfeil links beim Knoten #15 in Bild 15.12) und hängen alle Bits des Markierungsvektors, die vor der Position i+1 plaziert sind (u_1,...,u_i), rechts an diesen Variablenvektor an und bilden somit einen neuen "erweiterten" Vektor E = (V, u_1, ...u_i). Bei Knoten #0 in Bild 15.12, z. B., wird V = (0000) durch Anhängen von "1" zu E = (00001) erweitert.

3. Alle Knoten, deren Markierungsvektoren an der Position i+1 keine 0 besitzen bleiben unberücksichtigt, wie z.B. bei der Markierung L(V) = (010) am Knoten #4 in Bild 15.12. Deren Vektoren werden aus der Liste gestrichen

4. Aus der Menge aller gekennzeichneten E-Vektoren werden später die Min-Vektoren ermittelt (z.B. für i = 1 aus den 5 E-Vektoren 11110 (bei #15), 11010 (bei #13), 01110 (bei #7), 10001 (bei #8) und 00001 (bei #0) in Bild 15.12). Dies wird im folgenden Abschnitt behandelt.

15.1.4.2 Phase B.2: Ermittlung der "Min-Vektoren" aus den E-Vektoren.

Für die verschiedenen i (i = 0...R-1) ergibt sich somit jeweils eine Menge von "erweiterten

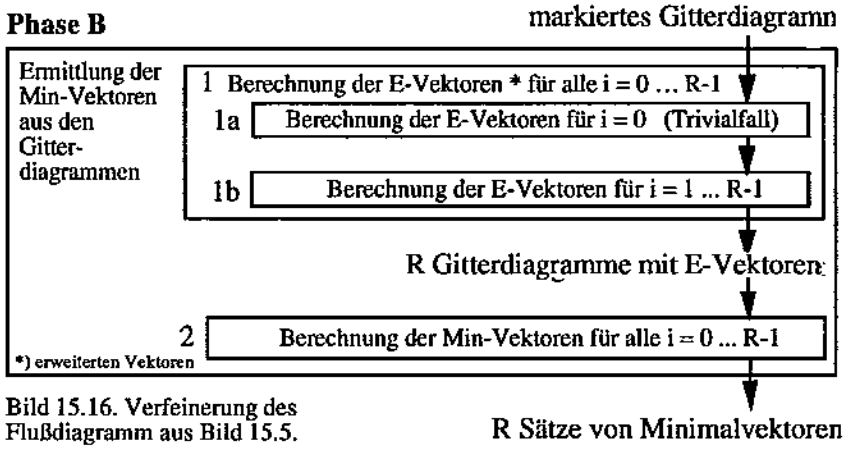


Bild 15.16. Verfeinerung des Flußdiagramm aus Bild 15.5.

Vektoren" (E-Vektoren). Aus jeder dieser Mengen werden dann die "Min-Vektoren" (Minimalvektoren) entsprechend der Definition 5 ermittelt. Diese sind z.B. für den Fall i = 2 die E-Vektoren 010001 (bei #11), 100010 (bei #8), 010001 (bei #4, vgl. Bild 15.15). Bild 15.13 ist die Darstellung des Falles der Minimalvektoren für i=0. Als Ergebnis erhalten wir die Vektoren 0100 (bei #4), 0010 (bei #2) und 0001 (bei #1, vgl. Bild 15.12).

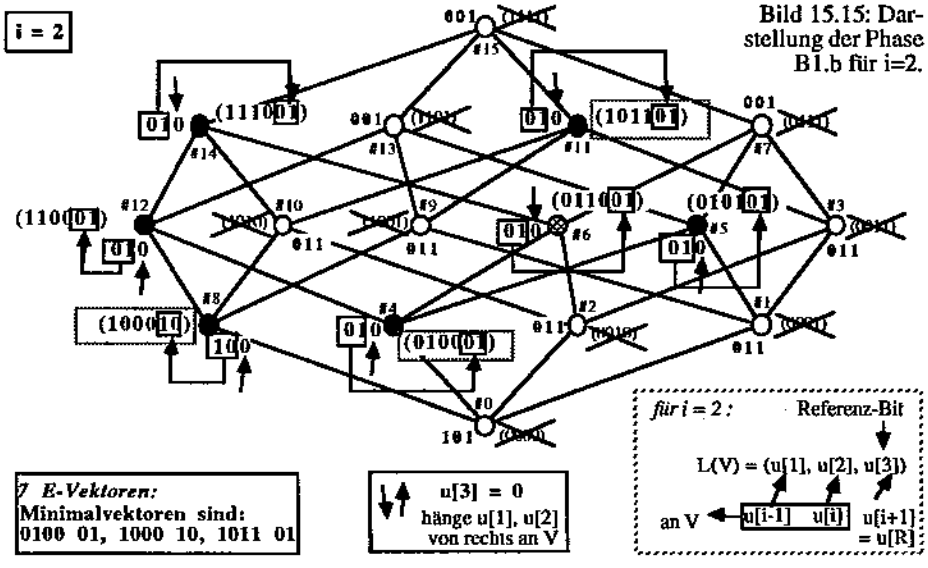


Bild 15.15: Darstellung der Phase B1.b für i=2.

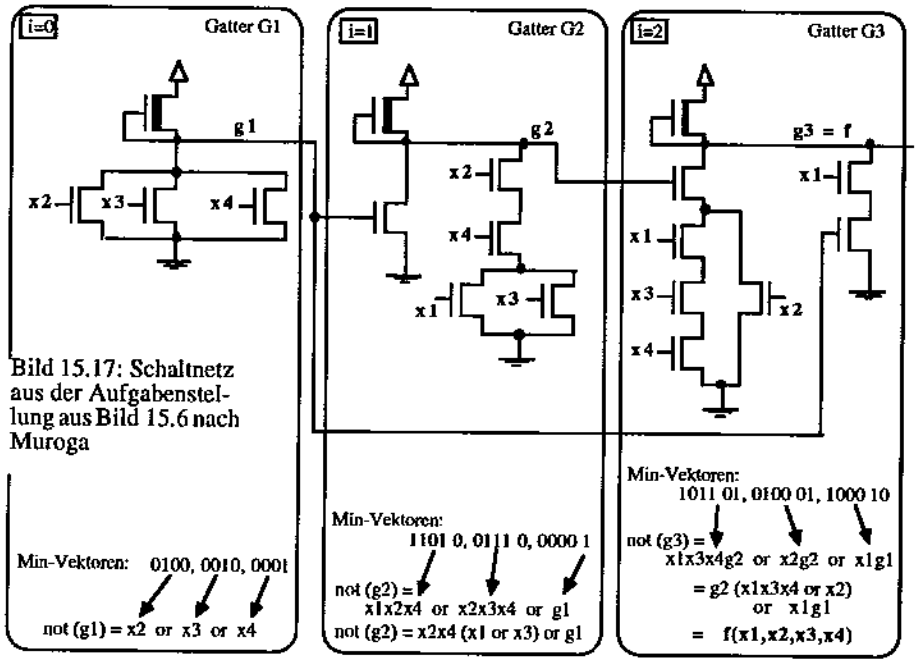


Bild 15.17: Schaltnetz aus der Aufgabenstellung aus Bild 15.6 nach Muroga

15.1.5 Phase C: Ermittlung der Gatter aus den Min-Vektoren.

In dieser Phase wird aus den in Phase B gefunden Min-Vektoren das Schaltnetz abgeleitet. Zuerst werden aus Min-Vektoren Boole'sche Gleichungen gewonnen. Die Min-Vektoren haben dabei die Form (x_1, ... , x_n, g_1, ... ,g_j). Auch hier gibt es wieder R Schritte, in denen aus den Min-Vektoren je ein Gatter-Schaltnetz entsteht. Der Algorithmus sieht folgendermaßen aus:

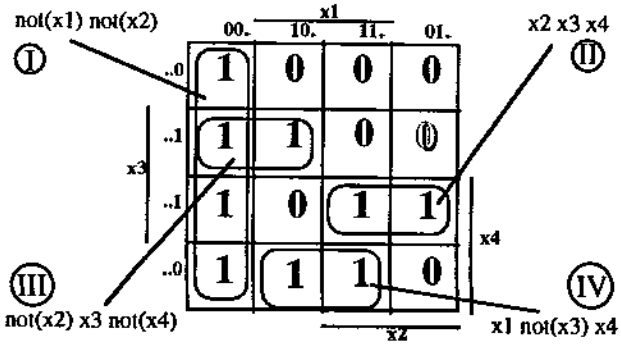


Bild 15.18: KV-Diagramm der Funktion aus Bild 15.6

for i = 0 to R-1 do

1. Bilde aus jedem Minimalvektor das Produkt der Variablen die eine 1 an der entsprechenden Position des Variablenvektors haben (z.B. ergibt sich für den Vektor V = 0100 in Bild 15.17 der Term x_2).

2. Bilde die Disjunktion der Produkte. Sie ergibt not(g[i+1]) (In Bild 15.17 ergibt sich für i=0 die Funktion **not**(g1) = x2 **or** x3 **or** x4).

3. Zeichne das Transistornetz zu dieser Boole'schen Funktion, wobei g[i+1] der Gatterausgang und gleichzeitig die Gatternummer ist (vgl. Bild 15.17).

Wenn dieser Algorithmus für alle i durchgeführt worden ist, muß aus den einzelnen Gattern das gesamte Schaltnetz aufgebaut werden. Im gesamten Bild 15.17, nachdem alle Gatter untereinander verbunden worden sind, kann man die resultierenden Gatter bzw. das gesamte Schaltnetz erkennen.

15.2 Vergleich der Muroga-Lösung mit einer "Geradeaus"-Lösung

Der Vorteil des Muroga-Algorithmus ist erst bei einem Vergleich mit einer "Geradeaus"-Lösung zur Erstellung des Schaltnetzes derselben Funktion zu sehen (Bild 15.6). Hier gewinnt man die Funktion mit Hilfe von Standardverfahren, die aus der Digitalen Logik bekannt sind (z.B.: KV-Diagramm Bild 15.18).

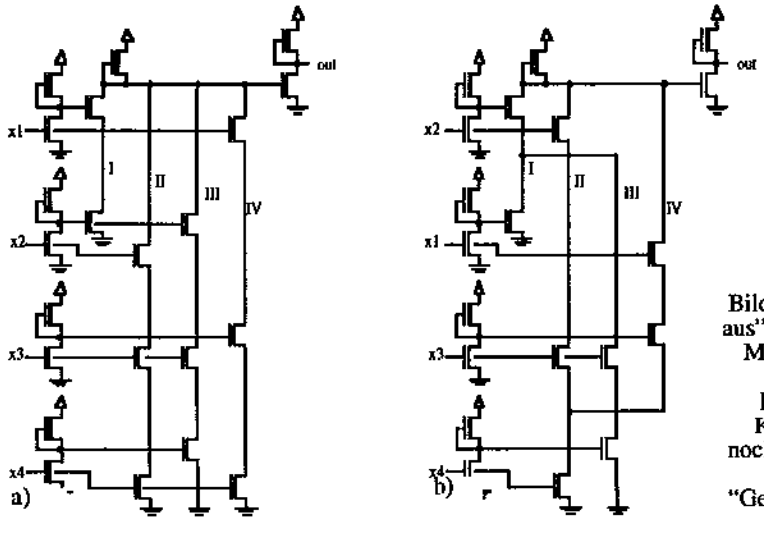


Bild 15.19: "Geradeaus"-Verfahren (ohne Muroga), a) direkte Umsetzung der Funktion aus dem KV-Diagramm, b) nochmals minimierte Version der "Geradeaus"-Lösung

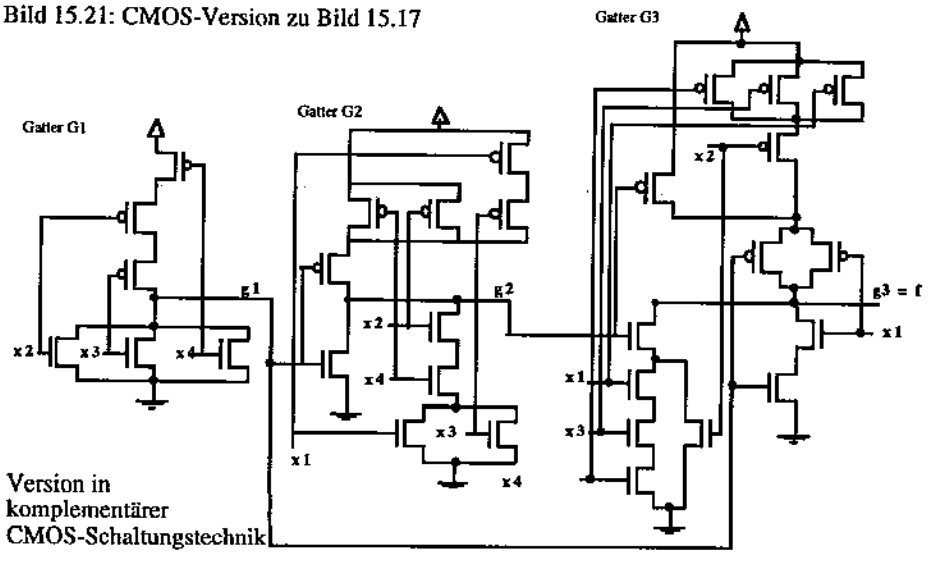
	Muroga-Verfahren	Standardlösung
Anz. der Gatter	3	6
Anz. der Transistoren (gesamt)	18	20
Anz. der pull-up-Transistoren	3	6
Anz. der pull-down-Transistoren	15	14

Bild 15.20: Vergleich zwischen Muroga- und Standardlösung

Aus der Funktion wird das Schaltnetz entworfen (Bild 15.19 a), wobei man bei geschickter Verdrahtung, d.h. daß man bei der gewonnen Funktion nochmals ausklammert, noch eine weitere Reduktion der Transistor-Anzahl ereichen kann (Bild 15.19 b).

Die Bild 15.20 zeigt für das Muroga-Verfahren und für die Standardlösung die Anzahl der Gatter und Transistoren. Auf den ersten Blick tritt der Vorteil des Muroga-Verfahrens nicht hervor, da die Gesamtzahl der Transistoren nur um 2 geringer ist, als bei der Standardlösung. Da beim Erstellen eines Layouts aber die Pullup-Transistoren deutlich mehr Platz benötigen, als die Pull-down-Transistoren ist das Muroga-Verfahren in dieser Hinsicht der Standardmethode bei weitem überlegen (für unser Beispiel benötigt der "Muroga" nur die Hälfte der Pullup-Transistoren; siehe Bild 15.20).

Bild 15.21: CMOS-Version zu Bild 15.17



Version in komplementärer CMOS-Schaltungstechnik

15.3 Anwendung auf komplementäre CMOS-Schaltungen (Phase D)

Die Ermittlung der für komplementäre CMOS-Schaltungen (Kapitel 16) notwendigen dualen Pullup-Netze ist unproblematisch. Die für den n-MOS-Fall ermittelten Pull-down-Netze können unverändert übernommen werden. Da diese sämtlich Serien/Parallel-zerlegbar sind, können aus diesen mit Leichtigkeit auf heuristische Weise über Betrachtung und direktes Zeichnen die dazu dualen Pullup-Netze konstruiert werden. Bild 15.21 zeigt die so entstandene komplementäre CMOS-Version zu Bild 15.17.

15.4 Literatur

[1] T. Ibaraki, S. Muroga: Synthesis of Networks with a Minimum Number of Negative Gates; IEEE TC-20, 1 (January 1971)

[2] H. C. Lai, S. Muroga: Automated Logic Design of NMOS Networks; in Vol. 9 of [7]

[3] T.-K. Liu: Synthesis Algorithms for 2-level NMOS-Networks; IEEE TC-24, 1 (Jan 1985)

[4] T.-K. Liu: Synthesis of Multilevel Feed-Forward NMOS Networks; IEEE TC-26,6 (June 1977)

[5] T.-K. Liu: Synthesis of Feed-Forward NMOS Networks with Cells of Similar complexities; IEEE TC-26, 8 (August 1977)

[6] S. Muroga: VLSI System Design; Wiley-Interscience 1982

[7] J. T. Tou: Advances in Information Systems Science; Plenum Press, New York 1985