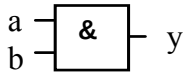
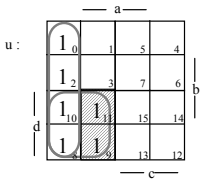


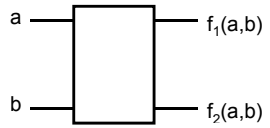
Kapitel 11: Schaltnetze



- Einführung in die formalen Grundlagen logischer Beschreibungen
  - Boolesche Algebra, Schaltalgebra
- Realisierung von Schaltnetzen auf Schalter- und Gatterebene
- Entwurf von Schaltnetzen
  - Logikminimierung, KV-Diagramme

Schaltnetze:

rein kombinatorische logische Schaltungen
kein Speicherverhalten
logische Funktionen



$f1(a,b) = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$
$f2(a,b) = (a \vee \bar{b}) \wedge (\bar{a} \vee b)$

Beispiel:

Licht-Aus Warnung im Kraftfahrzeug

„Motor aus“ und „Tür auf“ und „Licht an“ $\Rightarrow$ Alarm

Definition

Als eine Boolesche Algebra bezeichnet man eine Menge $V = \{a,b,c,...\}$, auf der zwei zweistellige Operationen $\oplus$ und $\otimes$ derart erklärt sind, dass durch ihre Anwendung auf Elemente aus V wieder Elemente aus V entstehen (Abgeschlossenheit).

Abgeschlossenheit: Für alle $a, b \in V$ gilt:

$$a \otimes b \in V$$
$$a \oplus b \in V$$

Weiterhin müssen die vier Huntingtonschen Axiome gelten.

H1 Kommutativgesetz

$a \otimes b = b \otimes a$
$a \oplus b = b \oplus a$

H2 Distributivgesetz

$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
$a \oplus (b \otimes c) = (a \oplus b) \otimes (a \oplus c)$

H3 Neutrales Element

Es existieren zwei Elemente $e, n \in V$, so dass gilt:

$a \otimes e = a$ (e wird Einselement genannt)
$a \oplus n = a$ (n wird Nullelement genannt)

H4 Inverses Element

Für alle $a \in V$ existiert ein Element $\bar{a} \in V$, so dass gilt:

$a \otimes \bar{a} = n$
$a \oplus \bar{a} = e$

Schaltalgebra

Die Schaltalgebra ist eine spezielle Boolesche Algebra, die durch folgende Korrespondenztabelle definiert wird:

Boolesche Algebra	Schaltalgebra
V	{0,1}
$\oplus$	$\vee$
$\otimes$	$\wedge$
n	0
e	1
$\bar{a}$	$\bar{a}$

Auch Schreibweise: a + b für $a \vee b$
a & b, ab ,a*b für $a \wedge b$

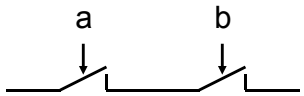
Schaltalgebra

Die Verknüpfungen können leicht in Funktionstabellen dargestellt werden:

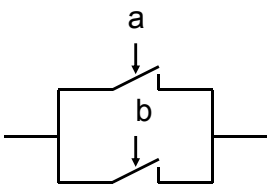
a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

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



Technische Realisierung mit Schaltern



Schaltalgebra

Huntingtonsche Axiome in der Schaltalgebra:

H1: $a \vee b = b \vee a$
$a \wedge b = b \wedge a$

H2: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

H3: $a \wedge 1 = a$
$a \vee 0 = a$

H4: $a \wedge \bar{a} = 0$
$a \vee \bar{a} = 1$

Schaltalgebra

Aus den vier Huntingtonschen Axiomen lassen sich weitere Sätze ableiten:

Assoziativgesetz: $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
$(a \vee b) \vee c = a \vee (b \vee c)$

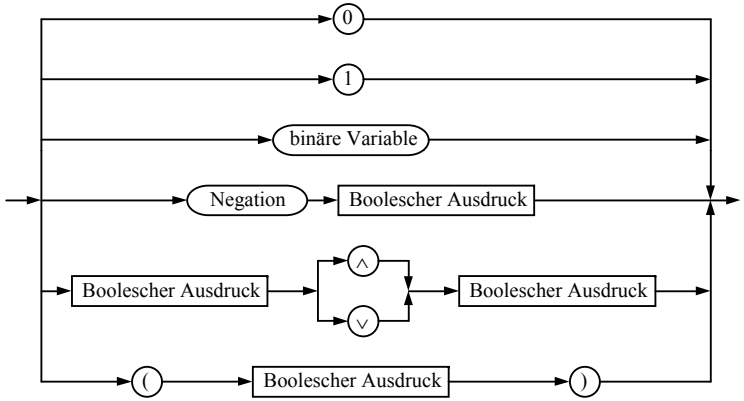
Idempotenzgesetz: $a \wedge a = a$
$a \vee a = a$

Absorptionsgesetz: $a \wedge (a \vee b) = a$
$a \vee (a \wedge b) = a$

DeMorgan-Gesetz: $\overline{a \wedge b} = \bar{a} \vee \bar{b}$
$\overline{a \vee b} = \bar{a} \wedge \bar{b}$

Boolescher Ausdruck

Ein Boolescher Ausdruck ist eine Zeichenfolge, die aus binären Variablen, den Operatoren $\wedge$ und $\vee$ und Klammern besteht und syntaktische Regeln erfüllt, die durch folgendes Syntaxdiagramm gegeben sind:



Boolescher Ausdruck

Beispiele:

syntaktisch korrekte Boolesche Ausdrücke: $a \vee b$, 0, $(\bar{a} \wedge b) \vee c$

keine Booleschen Ausdrücke, da syntaktisch nicht korrekt:

$a \vee \vee a$, 1 0, $( ) \vee c$

Für die Konstanten 0 und 1 verwendet man in der Schaltalgebra manchmal auch in Anlehnung an die Aussagenalgebra die Bezeichnung

Wahrheitswerte: 0 : falsch
1 : wahr

- Ein Boolescher Ausdruck hat in der Regel zunächst keinen Wahrheitswert, da er binäre Variable enthalten kann.
- Erst durch Belegung der binären Variablen mit Wahrheitswerten erhält der Boolesche Ausdruck einen Wahrheitswert.

Definitionen

- Die Belegung einer Menge von binären Variablen eines Booleschen Ausdrucks mit Wahrheitswerten bezeichnet man als Interpretation.
- Die Interpretation eines Booleschen Ausdrucks liefert eine Aussage, die entweder wahr oder falsch ist.
- Verschiedene Interpretationen eines Booleschen Ausdrucks können zu dem selben Wahrheitswert führen.
- Ein Boolescher Ausdruck, bei dem alle möglichen Interpretationen zum Wahrheitswert „wahr“ führen, heißt Tautologie.
  - Beispiel: $\bar{a} \vee a$ ist eine Tautologie.

Darstellung boolescher Funktionen

1. Durch eine Funktionstabelle
2. Durch einen algebraischen Ausdruck (symbolische Form)
3. Durch einen Graphen

Funktionstabelle

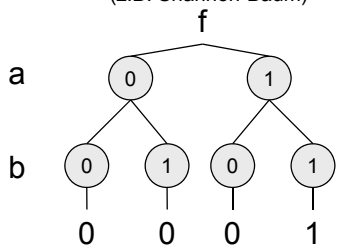
a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

symbolische Form

$f = a \wedge b$

Graph

(z.B. Shannon-Baum)



Boolesche Funktionen

Wie kommt man von der symbolischen Darstellung zur Funktionstabelle ?

Durch rekursive Auswertung des symbolischen Ausdrucks!

Konvention:

Negation vor Konjunktion und Konjunktion vor Disjunktion

Durch Klammern kann eine andere Reihenfolge der Auswertung festgelegt werden

$a \vee b \wedge \bar{c}$

$a \vee \boxed{b \wedge \bar{c}}$ Negation vor Konjunktion

$a \vee \boxed{b \wedge \bar{c}}$ Konjunktion vor Disjunktion

$a \vee b \wedge \bar{c}$

16 mögliche zweistellige boolesche Funktionen

x_1 0011 x_0 0101		verbale Form	symbolische Darstellung	Bezeichnung
f_0	0000	konstant 0	0	
f_1	0001	x_1 und x_0	$x_1 \wedge x_0$	Konjunktion
f_2	0010	nicht x_0, aber x_1	$x_1 \wedge \bar{x}_0$	Inhibition
f_3	0011	identisch x_1	x_1	Identität
f_4	0100	nicht x_1, aber x_0	$\bar{x}_1 \wedge x_0$	Inhibition
f_5	0101	identisch x_0	x_0	Identität
f_6	0110	x_1 ungleich x_0	$x_1 \not\leftrightarrow x_0$	Antivalenz, XOR
f_7	0111	x_1 oder x_0	$x_1 \vee x_0$	Disjunktion
f_8	1000	nicht (x_1 oder x_0)	$x_1 \bar{\vee} x_0$	NOR-Funktion, Peircescher Pfeil
f_9	1001	x_1 gleich x_0	$x_1 \leftrightarrow x_0$	Äquivalenz
f_{10}	1010	nicht x_0	$\bar{x}_0$	Negation
f_{11}	1011	wenn x_0, dann x_1	$x_0 \rightarrow x_1$	Implikation $\bar{x}_0 \vee x_1$
f_{12}	1100	nicht x_1	$\bar{x}_1$	Negation
f_{13}	1101	wenn x_1, dann x_0	$x_1 \rightarrow x_0$	Implikation $\bar{x}_1 \vee x_0$
f_{14}	1110	nicht (x_1 und x_0)	$x_1 \bar{\wedge} x_0$	NAND-Funktion, Shefferscher Strich
f_{15}	1111	konstant 1	1	Tautologie

Vollständige Operatorensysteme

Definition:

Ein System von Operatoren, mit dem alle booleschen Funktionen dargestellt werden können, heißt vollständiges Operatorensystem.

Die Operatoren ($\wedge$, $\vee$, $\bar{\ }$) bilden ein vollständiges Operatorensystem.

Operatoren-system	Darstellung der ...		
	Negation	Konjunktion	Disjunktion
($\wedge, \vee, \bar{\ }$)	$\bar{a}$	$a \wedge b$	$a \vee b$
($\wedge, \bar{\ }$)	$\bar{a}$	$a \wedge b$	$\overline{\bar{a} \wedge \bar{b}}$
($\vee, \bar{\ }$)	$\bar{a}$	$\overline{\bar{a} \vee \bar{b}}$	$a \vee b$
($\bar{\wedge}$)	$a \bar{\wedge} a$	$(a \bar{\wedge} b) \bar{\wedge} (a \bar{\wedge} b)$	$(a \bar{\wedge} a) \bar{\wedge} (b \bar{\wedge} b)$
($\bar{\vee}$)	$a \bar{\vee} a$	$(a \bar{\vee} a) \bar{\vee} (b \bar{\vee} b)$	$(a \bar{\vee} b) \bar{\vee} (a \bar{\vee} b)$
($\wedge, \not\leftrightarrow$)	$a \not\leftrightarrow 1$	$a \wedge b$	$a \not\leftrightarrow b \not\leftrightarrow (a \wedge b)$

a	y
0	1
1	0

a	b			y
0	0	1	1	0
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

a	b			y
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

Tautologie

- Wann repräsentieren zwei Ausdrücke A und B dieselbe Boolesche Funktion?
- Gleichbedeutend:
  Ist $A \leftrightarrow B$ eine Tautologie?
- Gegeben zwei Boolesche Funktionen:
  $f_1(a,b) = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$
  $f_2(a,b) = (a \vee \bar{b}) \wedge (\bar{a} \vee b)$
- Ist f_1 identisch mit f_2
  oder
  Ist $(a \wedge b) \vee (\bar{a} \wedge \bar{b}) \leftrightarrow (a \vee \bar{b}) \wedge (\bar{a} \vee b)$ eine Tautologie?

Tautologie

Beweis mit Hilfe von Funktionstabellen oder mittels Umformungen von Ausdrücken unter Verwendung der algebraischen Gesetze.

Zwei Ausdrücke sind äquivalent, falls die Ergebnisse ihrer Auswertung für alle möglichen Kombinationen von Variablenbelegungen identisch sind.

$X = f_1(a,b) = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$

$Y = f_2(a,b) = (a \vee \bar{b}) \wedge (\bar{a} \vee b)$

a b	$(a \wedge b) \vee (\bar{a} \wedge \bar{b})$	$(a \vee \bar{b}) \wedge (\bar{a} \vee b)$	$x \leftrightarrow y$
0 0	1	1	1
0 1	0	0	1
1 0	0	0	1
1 1	1	1	1

Tautologie

mittels algebraischer Umformung: $f_1(a,b) = f_2(a,b)$

$(a \wedge b) \vee (\bar{a} \wedge \bar{b}) = (a \vee \bar{b}) \wedge (\bar{a} \vee b)$

$(a \wedge b) \vee (\bar{a} \wedge \bar{b}) = [(a \wedge b) \vee \bar{a}] \wedge [(a \wedge b) \vee \bar{b}]$

(Distributivgesetz)

$= [\boxed{(a \vee \bar{a})} \wedge (b \vee \bar{a})] \wedge [(a \vee \bar{b}) \wedge \boxed{(b \vee \bar{b})}]$

(Distributivgesetz)

$= [\boxed{1} \wedge (b \vee \bar{a})] \wedge [(a \vee \bar{b}) \wedge \boxed{1}]$

(Inverses Element)

$= (b \vee \bar{a}) \wedge (a \vee \bar{b})$

(Neutrales Element)

$= (\bar{a} \vee b) \wedge (a \vee \bar{b})$

(Kommutativgesetz)

$(a \wedge b) \vee (\bar{a} \wedge \bar{b}) = (a \vee \bar{b}) \wedge (\bar{a} \vee b)$

KNF und DNF Normalformen

Eine boolesche Funktion kann durch verschiedene boolesche Ausdrücke beschrieben werden.

Beispiel: $f_1(a,b) = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$

$f_2(a,b) = (a \vee \bar{b}) \wedge (\bar{a} \vee b)$

Eine Standarddarstellung boolescher Funktionen im vollständigen Operatorensystem ($\wedge, \vee, \bar{\ }$) ist die konjunktive (KNF) und die disjunktive Normalform (DNF).

Definition:

Ein Literal L_i ist entweder eine Variable x_i oder ihre Negation $\bar{x}_i$

d.h. $L_i \in \{x_i, \bar{x}_i\}$

KNF und DNF Literale und Produktterme

Falls $L_h = x$, $L_j = x$, $h \neq j$ (mehrfach bejahtes Auftauchen)

$\Rightarrow L_h \wedge L_j = x$

$\Rightarrow K(x_1,\ldots,x_m) = \bigwedge_{i=1}^{m} L_i$

Mehrfaches Auftauchen von x kann nach Idempotenzgesetz: $a \wedge a = a$

$a \vee a = a$

gestrichen werden.

Falls $L_h = x$ und $L_j = \bar{x}$ (gemischtes bejahtes und negiertes Auftreten)

$\Rightarrow L_h \wedge L_j = 0$

$\Rightarrow K(x_1,\ldots,x_m) = 0$

Produktterm wird zu 0 nach Huntingtonschem Axiom H4: $a \wedge \bar{a} = 0$.

KNF und DNF
Definitionen

Ein Produkttherm $K(x_1,...,x_m)$ ist die Konjunktion von Literalen

$$\bigwedge_{i\in\{1,...,m\}} L_i = L_1 \wedge \ldots \wedge L_m$$

oder der Konstanten "0" oder "1"

Ein Produktterm $K(x_1,...,x_n)$ heißt Implikant einer Booleschen Funktion $y(x_1,...,x_n)$, wenn $K \rightarrow y$

Das heißt, für jede Belegung $B \in \{0,1\}^n$ gilt:
Wenn $K(B) = 1$, dann ist auch $y(B) = 1$

Ein Implikant einer Booleschen Funktion $y(x_1,...,x_n)$ heißt Minterm, wenn ein Literal jeder Variablen x_i der Funktion y im Implikanten genau einmal vorkommt.

Der Therm $D(x_1,...,x_m)$ ist eine Disjunktion von Literalen

$$\bigvee_{i\in\{1,...,m\}} L_i = L_1 \vee \ldots \vee L_m$$

oder der Konstanten "0" oder "1"

Der Term $D(x_1,...,x_m)$ heißt Implikat einer Booleschen Funktion $y(x_1,...,x_m)$, wenn $\bar{D} \rightarrow \bar{y}$

Das heißt für jede Belegung $B \in \{0,1\}^n$ gilt:
Wenn $D(B) = 0$, dann ist auch $y(B)=0$.

Ein Implikat einer Booleschen Funktion $y(x_1,...,x_m)$ heißt Maxterm, wenn ein Literal jeder Variable x_i der Funktion y im Implikaten genau einmal vorkommt.

KNF und DNF
Beispiel Min- und Maxterme

Minterme einer Booleschen Funktion $y(x_1,...,x_4)$:

$x_1 \wedge x_2 \wedge x_3 \wedge x_4$

$x_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_4$

Keine Minterme der Booleschen Funktion $y(x_1,...,x_4)$:

$x_1 \wedge x_2$

$x_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_3 \wedge x_4$

Maxterme einer Booleschen Funktion $y(x_1,...,x_3)$:

$x_1 \vee x_2 \vee x_3$

$x_1 \vee \bar{x}_2 \vee x_3$

KNF und DNF
Maxterm und Minterm

Der Maxterm einer KNF läßt sich für jede Zeile der Funktionstabelle bilden die den Funktionswert 0 liefert.

Herkunft der Bezeichnung Maxterm:

Funktionen aus genau einem Maxterm liefern für genau eine Belegung den Funktionswert 0, d.h. abgesehen von der trivialen Einsfunktion haben sie die maximale Anzahl an Einsen.

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

Der Minterm einer DNF läßt sich für jede Zeile der Funktionstabelle bilden die den Funktionswert 1 liefert.

Herkunft der Bezeichnung Minterm:

Funktionen aus genau einem Minterm liefern für genau eine Belegung den Funktionswert 1, d.h. abgesehen von der trivialen Nullfunktion haben sie eine minimale Anzahl an Einsen.

KNF und DNF
Definition

<u>Konjunktive Normalform KNF</u>

Es sei eine Boolesche Funktion $y(x_1,...,x_m)$ gegeben.
Ein Boolescher Ausdruck heißt konjunktive Normalform (KNF) der Funktion y, wenn er aus einer konjunktiven Verknüpfung aller Maxterme D_i besteht:

$$y = D_0 \wedge D_1 \wedge \ldots\ldots \wedge D_k \quad , k \leq 2^n-1$$

Es darf dabei keine zwei Disjunktionen D_i, D_j mit $i \neq j$ geben, die zueinander äquivalent sind.

<u>Disjunktive Normalform DNF</u>

Es sei eine Boolesche Funktion $y(x_1,...,x_n)$ gegeben.
Ein Boolescher Ausdruck heißt disjunktive Normalform (DNF) der Funktion y, wenn er aus einer disjunktiven Verknüpfung aller Minterme K_i besteht:

$$y = K_0 \vee K_1 \vee \ldots\ldots \vee K_k \quad , \quad k \leq 2^n-1$$

Es darf dabei keine zwei Konjunktionen K_i, K_j mit $i \neq j$ geben, die zueinander äquivalent sind.

KNF und DNF
Begriffe

Operatoren $\wedge, \vee, \overline{}$ Literale

Boolesche Funktion $y = f(a,b,c,d) = \overline{a} \wedge \overline{b} \vee b \wedge (\overline{c} \vee \overline{d}) \vee a \wedge b \wedge c \wedge \overline{d}$

Maxterme

KF Konjunktive Form $y = (\overline{a} \vee \overline{b}) \wedge (\overline{b} \vee c) \wedge (\overline{b} \vee \overline{d}) \wedge (\overline{a} \vee \overline{b} \vee \overline{c} \vee d)$

Implikat: Term, der für jede Belegung der Variablen, 0 ergibt, wenn auch die dazu gehörige boolsche Funktion 0 ergibt.

Maxterm: Zeile in der Funktionstabelle, wo das Resultat = 0

Produkttherme Minterme

DF Disjunktive Form $y = (a \wedge b) \vee (b \wedge \overline{c}) \vee (b \wedge d) \vee (a \wedge b \wedge c \wedge \overline{d})$

Produktterm: Konjunktion von Literalen

Implikant: Produktterm, wenn alle Variablen einer Funktion vorkommen.

Minterm: Zeile in der Funktionstabelle, wo das Resultat = 1

KNF und DNF
Funktionstabelle

	c b a	y	Minterme	Maxterme
	0 0 0	1	$\overline{a}\ \overline{b}\ \overline{c}$	
	0 0 1	0		$\overline{a} \vee b \vee c$
	0 1 0	0		$a \vee \overline{b} \vee c$
	0 1 1	1	$a\ b\ \overline{c}$	
	1 0 0	1	$\overline{a}\ \overline{b}\ c$	
	1 0 1	0		$\overline{a} \vee b \vee \overline{c}$
	1 1 0	0		$a \vee \overline{b} \vee \overline{c}$
	1 1 1	1	$a\ b\ c$	

DNF: $y = (\overline{a}\ \overline{b}\ \overline{c}) \vee (a\ b\ \overline{c}) \vee (\overline{a}\ \overline{b}\ c) \vee (a\ b\ c)$

KNF: $y = (\overline{a} \vee b \vee c)(a \vee \overline{b} \vee c)(\overline{a} \vee b \vee \overline{c})(a \vee \overline{b} \vee \overline{c})$

KNF und DNF
aus der Funktionstabelle

DNF:

Aus der Funktionstabelle einer Funktion erhält man die Minterme, indem man in allen Zeilen mit dem Funktionswert 1 jeweils alle Eingangsvariablen mit $\wedge$ verknüpft und dabei Eingangsvariablen mit dem Wert 0 negiert. Durch die disjunktive Verknüpfung dieser Minterme kann ein Boolescher Funktionsausdruck in DNF hergeleitet werden.

KNF:

Aus der Funktionstabelle einer Funktion erhält man die Maxterme, indem man in allen Zeilen mit dem Funktionswert 0 jeweils alle Eingangsvariablen mit $\vee$ verknüpft und dabei Eingangsvariablen mit dem Wert 1 negiert. Durch die konjunktive Verknüpfung dieser Maxterme kann ein Boolescher Funktionsausdruck in KNF hergeleitet werden.

KNF oder DNF
aus beliebiger Form

Um Funktionen aus der DF bzw. KF in die DNF bzw. KNF zu überführen, ist der Shannonsche Entwicklungssatz behilflich.

Entwicklung nach der Variablen x_i:

- die Variable wird in der Funktion auf den Wert 1 gesetzt,
- der entstehende Term konjunktiv mit x_i verknüpft,

$\vee$ -verknüpft mit:

- die Variable wird in der Funktion auf den Wert 0 gesetzt und
- der entstehende Term konjunktiv mit $\overline{x}_i$ verknüpft

$$y = f(x_1,\ldots,x_n)$$
$$= [\, x_i \wedge f(x_1,\ldots,x_{i-1}, \mathbf{1}, x_{i+1},\ldots,x_n)] \vee [\, \overline{x}_i \wedge f(x_1,\ldots,x_{i-1}, \mathbf{0}, x_{i+1},\ldots,x_n)]$$

KNF oder DNF aus beliebiger Form am Beispiel

$$y = a\,\bar{b}\,c \vee \bar{a}\,\bar{b} \vee b\,c$$

$$= a\,[1\,\bar{b}\,c \vee \bar{1}\,\bar{b} \vee b\,c] \vee \bar{a}\,[0\,\bar{b}\,c \vee \bar{0}\,\bar{b} \vee b\,c]$$

$$= a\,[\bar{b}\,c \vee b\,c] \vee \bar{a}\,[\bar{b} \vee b\,c]$$

$$= a\,[b\,(c) \vee \bar{b}\,(c)] \vee \bar{a}\,[b\,(c) \vee \bar{b}\,(1)]$$

$$= a\,[b\,(c) \vee \bar{b}\,(c)] \vee \bar{a}\,[b\,c \vee \bar{b}\,(c \vee \bar{c})]$$

$$= a\,b\,c \vee a\,\bar{b}\,c \vee \bar{a}\,b\,c \vee \bar{a}\,\bar{b}\,c \vee \bar{a}\,\bar{b}\,\bar{c}$$

DNF und KNF

Disjunktive und konjunktive Normalformen sind eindeutige Darstellungen!

bis auf Permutationen (z.B. abc, acb, bac, bca, cab, cba sind äquivalent)

Beispiel:

$$y = a\,\bar{b} \vee c$$

DNF: $y = \bar{a}\,\bar{b}\,c \vee \bar{a}\,b\,c \vee a\,\bar{b}\,\bar{c} \vee a\,\bar{b}\,c \vee a\,b\,c$

KNF: $y = (a \vee b \vee c) \wedge (a \vee \bar{b} \vee c) \wedge (\bar{a} \vee \bar{b} \vee c)$

DNF und KNF

In einer Funktion mit n Variablen können bis zu 2^n Minterme bzw. Maxterme auftreten. Für n = 3 sind diese:

	Minterm	Maxterm
0	$\bar{a}\,\bar{b}\,\bar{c}$	$a \vee b \vee c$
1	$a\,\bar{b}\,\bar{c}$	$\bar{a} \vee b \vee c$
2	$\bar{a}\,b\,\bar{c}$	$a \vee \bar{b} \vee c$
3	$a\,b\,\bar{c}$	$\bar{a} \vee \bar{b} \vee c$
4	$\bar{a}\,\bar{b}\,c$	$a \vee b \vee \bar{c}$
5	$a\,\bar{b}\,c$	$\bar{a} \vee b \vee \bar{c}$
6	$\bar{a}\,b\,c$	$a \vee \bar{b} \vee \bar{c}$
7	$a\,b\,c$	$\bar{a} \vee \bar{b} \vee \bar{c}$

DNF und KNF

Um eine Funktion zu beschreiben, reicht die Angabe aller Minterme oder aller Maxterme aus.

Nr.	a b c	y	Minterme	Maxterme
0	0 0 0	1	$\bar{a}\,\bar{b}\,\bar{c}$	
1	1 0 0	0		$\bar{a} \vee b \vee c$
2	0 1 0	0		$a \vee \bar{b} \vee c$
3	1 1 0	1	$a\,b\,\bar{c}$	
4	0 0 1	1	$\bar{a}\,\bar{b}\,c$	
5	1 0 1	0		$\bar{a} \vee b \vee \bar{c}$
6	0 1 1	0		$a \vee \bar{b} \vee \bar{c}$
7	1 1 1	1	$a\,b\,c$	

DNF: $y = (\bar{a}\,\bar{b}\,\bar{c}) \vee (a\,b\,\bar{c}) \vee (\bar{a}\,\bar{b}\,c) \vee (a\,b\,c)$

KNF: $y = (\bar{a} \vee b \vee c)(a \vee \bar{b} \vee c)(\bar{a} \vee b \vee \bar{c})(a \vee \bar{b} \vee \bar{c})$

DNF und KNF

Verkürzte Schreibweise:

nur die Indizes (Binärkodierungen der (c, b, a) - Belegung) der 1- oder 0-Stellen der Funktion

Voraussetzung:

Eindeutige Reihenfolge der Variablen.

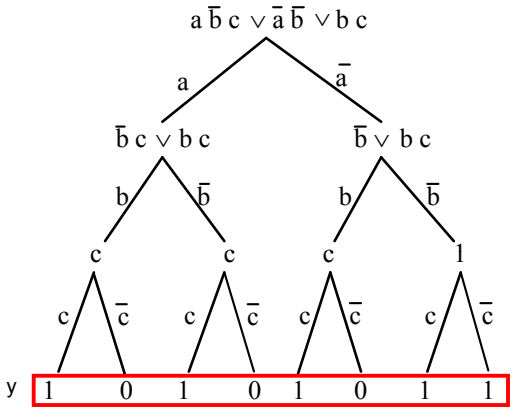
Beispiel (Reihenfolge der drei Variablen c, b, a)

y = MINt(0, 3, 4, 7)

y = MAXt(1, 2, 5, 6)

Nr.	a b c	y
0	0 0 0	1
1	1 0 0	0
2	0 1 0	0
3	1 1 0	1
4	0 0 1	1
5	1 0 1	0
6	0 1 1	0
7	1 1 1	1

Beispiel: Shannon-Baum



	a	b	c	y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Nachdem die Funktion nach allen Variablen entwickelt wurde, können die Minterme durch Verfolgen der Äste des Baums gefunden werden, die zu einer 1 führen.

DMF und KMF

Ziele:

„Möglichst kurze“ Boolesche Ausdrücke für eine gegebene Boolesche Funktion.

Technische Realisierung einer Schaltung mit möglichst geringen Kosten.

Ähnlich zum Aufbau der disjunktiven und konjunktiven Normalform gibt es eine disjunktive (DMF) und konjunktive (KMF) Minimalform.

DMF und KMF Kostenfunktion

Gegeben:

Eine Boolesche Funktion $y(x_1,\ldots,x_n)$
und die
Kostenfunktionen $\Psi_{UND}(k)$ und $\Psi_{ODER}(k)$,
die Realisierungskosten einer k-stelligen UND- bzw. ODER Verknüpfung (Kosten für die Negation einer Variablen seien vernachlässigt).

Ziel der Minimalform ist es die geringst möglichen Kosten zu erreichen.

DMF und KMF
Definition

Ein Ausdruck ist in disjunktiver Minimalform (Abk.: DMF), wenn er eine Disjunktion von Konjunktionen

$y(x_1,...,x_n) = (L_{11} \cdot ... \cdot L_{1k}) \vee ... \vee (L_{m1} \cdot ... \cdot L_{mj})$

mit den Literalen $L_{vw} \in \{ x_1,..., x_n, \bar{x}_1,..., \bar{x}_n\}$

darstellt und die Realisierungskosten

$Y_y = \Psi_{ODER}(m) + \Psi_{UND}(k) + ... + \Psi_{UND}(j)$

minimal sind.

Ein Ausdruck ist in konjunktiver Minimalform (Abk.: KMF), wenn er eine Konjunktion von Disjunktionen

$y(x_1,...,x_n) = (L_{11} \vee ... \vee L_{1k}) \cdot ... \cdot (L_{m1} \vee ... \vee L_{mj})$

mit den Literalen $L_{vw} \in \{ x_1 ,..., x_n, \bar{x}_1 ,..., \bar{x}_n\}$

darstellt und die Realisierungskosten

$Y_y = \Psi_{UND}(m) + \Psi_{ODER}(k) + ... + \Psi_{ODER}(j)$

minimal sind.

DMF und KMF
Beispiele

Kostenmaß: Anzahl der auftretenden Gatter

Es sei $\Psi_{UND}(k) = \Psi_{ODER}(k) = k$, k: Anzahl Gattereingänge

Beispiel DMF:

$y_1 = \bar{a}\, b \vee a\, \bar{b}$
ist in disjunktiver Minimalform,

$y_2 = \bar{a}\, b \vee a\, b$ jedoch nicht,

da y_2 auch kürzer durch

$y_2 = b$

ausgedrückt werden kann.

Beispiel KMF:

$y_1 = (\bar{a} \vee b) (a \vee c)$
ist in konjunktiver Minimalform,

$y_2 = \bar{a} (\bar{a} \vee c) (b \vee c)$ jedoch nicht,

da y_2 auch kürzer durch

$y_2 = \bar{a} (b \vee c)$

ausgedrückt werden kann.

DMF und KMF
Minimalformen sind nicht eindeutig

Es kann mehrere disjunktive und konjunktive Minimalformen für die gleiche Funktion geben.
Beispiel:

$y = a\, \bar{b} \vee b\, \bar{c} \vee \bar{a}\, c$ und

$y = a\, \bar{c} \vee \bar{b}\, c \vee \bar{a}\, b$

stellen dieselbe Funktion dar, beides sind disjunktive Minimalformen.

DMF und KMF
Unterschiedliche Kosten von

Die Kosten für eine disjunktive und eine konjunktive Minimalform derselben Funktion sind im Allgemeinen unterschiedlich.

Es sei $\Psi_{UND}(k) = \Psi_{ODER}(k) = 2 \cdot (k+1)$ und k ist die Anzahl der Gattereingänge

$y = a\, b\, c \vee \bar{a}\, \bar{b}\, \bar{c}$ disjunktive Minimalform

$y(x_1,...,x_n) = (L_{11} \cdot ... \cdot L_{1k}) \vee ... \vee (L_{m1} \cdot ... \cdot L_{mj})$

$\Psi y = \Psi_{ODER}(m) + \Psi_{UND}(k) + ... + \Psi_{UND}(j)$

$\Psi y = 6 + 8 + 8 = 22$

$y = (\bar{a} \vee b) (a \vee \bar{c}) (\bar{b} \vee c)$ konjunktive Minimalform

$y(x_1,...,x_n) = (L_{11} \vee ... \vee L_{1k}) \cdot ... \cdot (L_{m1} \vee ... \vee L_{mj})$

$\Psi y = \Psi_{UND}(m) + \Psi_{ODER}(k) + ... + \Psi_{ODER}(j)$

$\Psi y = 8 + 6 + 6 + 6 = 26$

NAND/NOR-Konversion

($\bar{\wedge}$)-System (NAND-System) und ($\bar{\vee}$)-System (NOR-System) sind vollständige Operatorensysteme

⇨ beliebige disjunktive und konjunktive Ausdrücke können mit NAND- und NOR-Verknüpfungen dargestellt werden.

Überführungen (vier Fälle):

1. Fall: Funktion in disjunktiver Form ⇒ ($\bar{\wedge}$)-System
2. Fall: Funktion in disjunktiver Form ⇒ ($\bar{\vee}$)-System
3. Fall: Funktion in konjunktiver Form ⇒ ($\bar{\vee}$)-System
4. Fall: Funktion in konjunktiver Form ⇒ ($\bar{\wedge}$)-System

Warum das alles?

⇨ Einfache Implementierung in Hardware!

NANDs/NORs sind sehr einfach in CMOS-Schaltungen realisierbar.

NAND-Konversion
1.Fall Funktion in disjunktiver Form ⇒ ($\bar{\wedge}$)-System

1. Fall: Funktion in disjunktiver Form ⇒ ($\bar{\wedge}$)-System

Gegeben: Boolsche Funktion in disjunktiver Form.

$$Y = \bar{a}\, b\, c \vee a\, \bar{b}\, c \vee a\, b\, \bar{c} \vee \bar{a}\, \bar{b}\, \bar{c}$$

Überführung:

1. Doppelte Negation und
2. anschließende Anwendung der DeMorganschen Regeln

Dann erhält man einen Ausdruck, der nur noch NAND als Operator enthält.

NAND-Konversion
1.Fall Funktion in disjunktiver Form ⇒ ($\bar{\wedge}$)-System

$$\begin{aligned} y &= \bar{a}\, b\, c \vee a\, \bar{b}\, c \vee a\, b\, \bar{c} \vee \bar{a}\, \bar{b}\, \bar{c} \\ &= \overline{\overline{\bar{a}\, b\, c \vee a\, \bar{b}\, c \vee a\, b\, \bar{c} \vee \bar{a}\, \bar{b}\, \bar{c}}} \\ &= \overline{\overline{\bar{a}\, b\, c} \wedge \overline{a\, \bar{b}\, c} \wedge \overline{a\, b\, \bar{c}} \wedge \overline{\bar{a}\, \bar{b}\, \bar{c}}} \\ &= \mathrm{NAND}_4(\mathrm{NAND}_3(\bar{a}, b, c), \mathrm{NAND}_3(a, \bar{b}, c), \\ &\qquad \mathrm{NAND}_3(a, b, \bar{c}), \mathrm{NAND}_3(\bar{a}, \bar{b}, \bar{c})) \end{aligned}$$

Dabei ist $\mathrm{NAND}_k(x_1,\ldots,x_k)$ eine k-stellige Funktion, für die gilt:

$$\mathrm{NAND}_k(x_1,\ldots,x_k) \begin{cases} 0 \text{ für } x_1 = \ldots = x_k = 1 \\ 1 \text{ sonst} \end{cases}$$

NAND-Konversion
1.Fall Funktion in disjunktiver Form ⇒ ($\bar{\wedge}$)-System

Darstellung der NAND_2-Funktion durch den $\bar{\wedge}$ Operator:

Problem: Die Operatoren $\bar{\wedge}$ und $\bar{\vee}$ sind nicht assoziativ.

$$(x_1 \bar{\wedge} x_2) \bar{\wedge} x_3 \neq x_1 \bar{\wedge} (x_2 \bar{\wedge} x_3)$$

$$(x_1 \bar{\vee} x_2) \bar{\vee} x_3 \neq x_1 \bar{\vee} (x_2 \bar{\vee} x_3)$$

$$\mathrm{NAND}_3(x_1, x_2, x_3) = \overline{x_1 \wedge x_2 \wedge x_3} = \overline{(x_1 \bar{\wedge} x_2)} \bar{\wedge} x_3$$

$$(x_1 \bar{\wedge} x_2) \bar{\wedge} x_3 = \overline{\overline{\overline{x_1 \wedge x_2} \wedge x_3}} \neq$$

$$x_1 \bar{\wedge} (x_2 \bar{\wedge} x_3) = \overline{x_1 \wedge \overline{\overline{x_2 \wedge x_3}}}$$

NAND-Konversion
1.Fall Funktion in disjunktiver Form $\Rightarrow$ ($\bar{\wedge}$)-System

Um die $NAND_k$-Funktion trotzdem für k > 2 durch $\bar{\wedge}$-Operatoren auszudrücken, geht man wie folgt vor:

1. Die Variablen x_1 bis x_k werden durch den $\bar{\wedge}$-Operator verknüpft: $x_1 \bar{\wedge} \ldots \bar{\wedge} x_k$
2. Auf die linke Seite des Terms werden k-2 offene Klammern "(" und hinter jede Variable außer x_1 und x_k wird jeweils eine geschlossene Klammer ")" gesetzt:
   $(\ldots(x_1 \bar{\wedge} x_2) \ldots \bar{\wedge} x_{k-1}) \bar{\wedge} x_k$
3. Schließlich wird jede Klammerebene negiert:
   $\overline{\overline{\ldots(x_1 \bar{\wedge} x_2) \ldots \bar{\wedge} x_{k-1}} \bar{\wedge} x_k}$

NAND-Konversion
1.Fall Funktion in disjunktiver Form $\Rightarrow$ ($\bar{\wedge}$)-System

$$\overline{(x_1 \bar{\wedge} x_2)} \bar{\wedge} x_3 = \overline{\overline{\overline{x_1 \wedge x_2} \wedge x_3}}$$
$$= \overline{x_1 \wedge x_2 \wedge x_3}$$
$$= NAND_3(x_1, x_2, x_3)$$

$$\overline{(\overline{(x_1 \bar{\wedge} x_2)} \bar{\wedge} x_3) \bar{\wedge} x_4} = \overline{\overline{\overline{\overline{x_1 \wedge x_2} \wedge x_3} \wedge x_4}}$$
$$= \overline{\overline{\overline{x_1 \wedge x_2 \wedge x_3} \wedge x_4}}$$
$$= \overline{x_1 \wedge x_2 \wedge x_3 \wedge x_4}$$
$$= NAND_4(x_1, x_2, x_3, x_4)$$

NOR-Konversion
2.Fall: Funktion in disjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

2. Fall: Funktion in disjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

Gegeben: Boolesche Funktion in disjunktiver Form.

$$y = \bar{a}\, b\, c \vee a\, \bar{b}\, c \vee a\, b\, \bar{c} \vee \bar{a}\, \bar{b}\, \bar{c}$$

Überführung:

1. Die Funktion wird zunächst negiert.
2. Anschließend werden die Konjunktionen doppelt negiert und
3. die DeMorganschen Regeln angewendet.
4. Das Ergebnis muss dann noch negiert werden.

NOR-Konversion
2.Fall: Funktion in disjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

$$y = \bar{a}\, b\, c \vee a\, \bar{b}\, c \vee a\, b\, \bar{c} \vee \bar{a}\, \bar{b}\, \bar{c}$$
$$\bar{y} = \overline{\bar{a}\, b\, c \vee a\, \bar{b}\, c \vee a\, b\, \bar{c} \vee \bar{a}\, \bar{b}\, \bar{c}}$$
$$= \overline{\overline{\bar{a}\, b\, c} \vee \overline{a\, \bar{b}\, c} \vee \overline{a\, b\, \bar{c}} \vee \overline{\bar{a}\, \bar{b}\, \bar{c}}}$$
$$= \overline{(a \vee \bar{b} \vee \bar{c}) \vee (\bar{a} \vee b \vee c) \vee (\bar{a} \vee \bar{b} \vee c) \vee (a \vee b \vee c)}$$
$$= NOR_4(NOR_3(a, \bar{b}, \bar{c}), NOR_3(\bar{a}, b, \bar{c}), NOR_3(\bar{a}, \bar{b}, c), NOR_3(a, b, c))$$

Die Negation von $\bar{y}$ zu y erhält man mit $y = \bar{y} \bar{\vee} \bar{y}$.

$$y = \overline{\bar{y} \vee \bar{y}} = NOR_2(\bar{y}, \bar{y})$$

NOR-Konversion
2.Fall: Funktion in disjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

Dabei ist $NOR_k(x_1,...,x_k)$ eine k-stellige Funktion, für die gilt:

$$NOR_k(x_1,...,x_k) \begin{cases} 1 \text{ für } x_1 = ... = x_k = 0 \\ 0 \text{ sonst} \end{cases}$$

Die NOR_k -Funktion lässt sich ebenfalls ausschließlich mit $\bar{\vee}$ als Operator darstellen.

NOR-Konversion
3.Fall: Funktion in konjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

3. Fall: Funktion in konjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

Gegeben: Boolesche Funktion in konjunktiver Form

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

Überführung:

1. Der Ausdruck wird doppelt negiert.
2. Anschließend wendet man die DeMorgan-Regeln an.

NOR-Konversion
3.Fall: Funktion in konjunktiver Form $\Rightarrow$ ($\bar{\vee}$)-System

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

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

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

$$= NOR_4(\, NOR_3(\bar{a}, b, c),\ NOR_3(a, \bar{b}, c),\ NOR_3(a, b, \bar{c}),\ NOR_3(\bar{a}, \bar{b}, \bar{c})\,)$$

NAND-Konversion
4.Fall: Funktion in konjunktiver Form $\Rightarrow$ ($\bar{\wedge}$)-System

4. Fall: Funktion in konjunktiver Form $\Rightarrow$ ($\bar{\wedge}$)-System

Gegeben: Boolesche Funktion in konjunktiver Form

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

Überführung:

1. Die Funktion wird negiert.
2. Anschließend werden die Disjunktionen doppelt negiert und
3. die DeMorgan-Regeln angewendet.
4. Das Ergebnis ist dann noch zu negieren.

NAND-Konversion

4.Fall: Funktion in konjunktiver Form $\Rightarrow$ ($\overline{\wedge}$)-System

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

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

$$= \overline{\overline{\overline{(\overline{a} \vee b \vee c)}} \wedge \overline{\overline{(a \vee \overline{b} \vee c)}} \wedge \overline{\overline{(a \vee b \vee \overline{c})}} \wedge \overline{\overline{(\overline{a} \vee \overline{b} \vee \overline{c})}}}$$

$$= NAND_4(NAND_3(a, \overline{b}, \overline{c}), NAND_3(\overline{a}, b, \overline{c}), NAND_3(\overline{a}, \overline{b}, c), NAND_3(a, b, c))$$

Die Negation von $\overline{y}$ zu y erhält man auch mit $y = \overline{y} \overline{\wedge} \overline{y}$.

$$y = \overline{\overline{y} \wedge \overline{y}} = NAND_2(\overline{y}, \overline{y})$$

Entwurf von Schaltnetzen

Beim praktischen Entwurf von Schaltnetzen muss beachtet werden, dass reale Gatter keine idealen Verknüpfungen sind, sondern z.B. Verzögerungszeiten besitzen, Wärme abgeben, Platz benötigen, ...

Technische Kriterien

- Leistungsaufnahme, Schaltzeit, Platzbedarf, Material, Lebensdauer, ...

Ökonomische Kriterien

- Kosten

Entwurf von Schaltnetzen

Technische Kriterien I

Korrekte Realisierung unter Beachtung des statischen und dynamischen Verhaltens der verwendeten Bauelemente

- z.B. Vermeidung von dynamischen Störeffekten, wie Hasards und Wettläufen

Berücksichtigung technischer Beschränkungen, z.B.

- begrenzte Anzahl von Eingängen der verwendeten Logikelemente
- begrenzte Belastbarkeit der Ausgänge der Elemente, Bauelemente-Bibliothek
- ...

Entwurf von Schaltnetzen

Technische Kriterien II

Gewährleistung einer hohen Systemzuverlässigkeit und Verfügbarkeit (leichte Testbarkeit, Selbsttest, Fehlertoleranz),

Berücksichtigung von Forderungen der Gebrauchseigenschaften (z.B. hoher Komfort, hohe Arbeitsgeschwindigkeit, großer Funktionsumfang),

Berücksichtigung technologischer Nebenbedingungen (z.B. Besonderheiten verschiedener Halbleitertechnologien),

Vermeidung von Störeinflüssen durch äußere elektromagnetische Felder.

Entwurf von Schaltnetzen
Ökonomische Kriterien

Geringe Kosten für den Entwurf (Entwurfsaufwand), z.B. Lohnkosten, Kosten für Rechnerbenutzung zur Entwurfsunterstützung,

Geringe Kosten für die Realisierung (Realisierungsaufwand), z.B. Kosten für die eingesetzten Bauelemente (bei der Herstellung integrierter Schaltkreise wird die Realisierung auf einer möglichst kleinen Chipfläche gefordert)

Geringe Kosten für die Inbetriebnahme und den laufenden Betrieb, z.B. Kosten für Test, Wartung und Energie.

Entwurf von Schaltnetzen

Grenzen zwischen technischen und ökonomischen Kriterien sind teilweise fließend.

Einzelne Kriterien stehen auch im Widerspruch zueinander.

z.B. Erhöhung der Zuverlässigkeit ⇒ Erhöhung der Kosten
geringere Realisierungskosten ⇒ höhere Entwurfskosten etc.

➔ Aufgabe des Entwerfers besteht darin, für eine bestimmte Problemstellung den günstigsten Kompromiss zu finden.

Ziel des Entwurfs: Unter Einhaltung bestimmter technischer Kriterien vor allem einen günstigen Kompromiss bezüglich der ökonomischen Kriterien anzustreben, um so zu einem Minimum an Gesamtkosten zu gelangen.

Minimierungsverfahren

Grundlage:

Realisierung der Schaltnetze in zweistufiger Form (DF, KF)

Darstellungsform, deren Realisierung die geringsten Kosten verursacht:

Minimalform

Den Vorgang der Erzeugung einer Minimalform bezeichnet man als Minimierung.

Minimierungsverfahren

Es gibt drei Arten von Minimierungsverfahren:

- Algebraische Verfahren
- Graphische Verfahren
- Tabellarische Verfahren

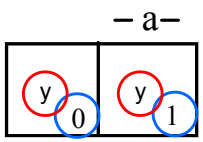
Algebraische und graphische Verfahren eignen sich nur für Funktionen mit bis zu 5 oder 6 Variablen, danach werden sie zu unübersichtlich.

Bei größerer Variablenzahl wendet man besser tabellarische Verfahren an.

Graphische Verfahren

Das KV-Diagramm (nach Karnaugh und Veitch)
auch oft nur Karnaugh map, Karnaugh-Diagramm

Ausgangspunkt ist ein Rechteck, dessen rechte Hälfte der Variablen a und dessen linke Hälfte $\bar{a}$ zugeordnet wird.



	a	y
0	0	0
1	1	1

	a	y
0	0	1
1	1	0

KV-Diagramm für eine Variable

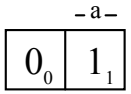
KV-Diagramm

Die Zahl in den Feldern gibt den Index der Variablenbelegung an
Index des Minterms, der dort den Wert 1 annimmt oder
Index des Maxterms, der dort den Wert 0 annimmt.

Durch Eintragen der Wahrheitswerte 0 oder 1 in die Felder des KV-Diagramms wird eine Boolesche Funktion charakterisiert.

Das KV-Diagramm ist eine weitere Darstellungsform Boolescher Funktionen (Alternative zur Funktionstabelle).

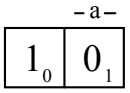
y:



y = a

	a	y
0	0	0
1	1	1

z:

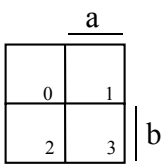


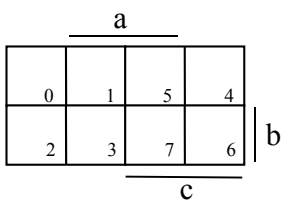
$z = \bar{a}$

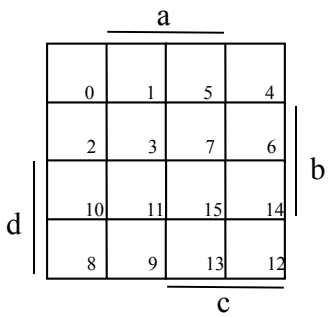
	a	z
0	0	1
1	1	0

KV-Diagramme

KV-Diagramme für mehrere Variable erhält man durch Spiegelung (für jede neue Variable verdoppelt sich die Anzahl der möglichen Belegungen)



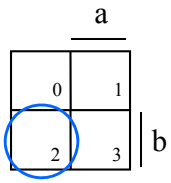




KV-Diagramme

Jedes Feld hat eine eindeutige Variablenzuordnung, die an den Rändern abgelesen werden kann: Feld 2 hat die Zuordnung $\bar{a}$ b.

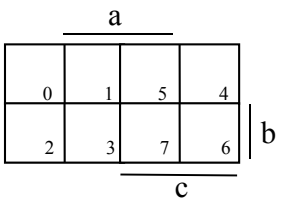
Der Index in den Feldern gibt den Index der zum Feld gehörenden Variablenbelegung an dabei die Variablen in umgekehrter alphabetischer Reihenfolge angetragen. Feld 2 = 10_2 = b $\bar{a}$



	b	a	y
0	0	0	
1	0	1	
2	1	0	?
3	1	1	

KV-Diagramme

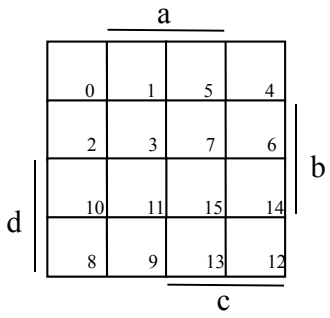
Feld 0 hat die Zuordnung: $\bar{a}\ \bar{b}\ \bar{c}$, Feld 0 = 000_0 = $\bar{c}\ \bar{b}\ \bar{a}$
Feld 1 hat die Zuordnung: $a\ \bar{b}\ \bar{c}$, Feld 1 = 001_1 = $\bar{c}\ \bar{b}\ a$
Feld 2 hat die Zuordnung: $\bar{a}\ b\ \bar{c}$, Feld 2 = 010_2 = $\bar{c}\ b\ \bar{a}$
Feld 3 hat die Zuordnung: $a\ b\ \bar{c}$, Feld 3 = 011_3 = $\bar{c}\ b\ a$
Feld 4 hat die Zuordnung: $\bar{a}\ \bar{b}\ c$, Feld 4 = 100_4 = $c\ \bar{b}\ \bar{a}$
Feld 5 hat die Zuordnung: $a\ \bar{b}\ c$, Feld 5 = 101_5 = $c\ \bar{b}\ a$
Feld 6 hat die Zuordnung: $\bar{a}\ b\ c$, Feld 6 = 110_6 = $c\ b\ \bar{a}$
Feld 7 hat die Zuordnung: $a\ b\ c$, Feld 7 = 111_7 = $c\ b\ a$



	c	b	a	y
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

KV-Diagramme

Feld 0 hat die Zuordnung: $\bar{a}\ \bar{b}\ \bar{c}\ \bar{d}$, Feld 0 = 0000_0 = $\bar{d}\ \bar{c}\ \bar{b}\ \bar{a}$
Feld 1 hat die Zuordnung: $a\ \bar{b}\ \bar{c}\ \bar{d}$, Feld 1 = 0001_1 = $\bar{d}\ \bar{c}\ \bar{b}\ a$
……
Feld 3 hat die Zuordnung: $a\ b\ \bar{c}\ \bar{d}$, Feld 3 = 0011_3 = $\bar{d}\ \bar{c}\ b\ a$
Feld 4 hat die Zuordnung: $\bar{a}\ \bar{b}\ c\ \bar{d}$, Feld 4 = 0100_4 = $\bar{d}\ c\ \bar{b}\ \bar{a}$
……
Feld 14 hat die Zuordnung: $\bar{a}\ b\ c\ d$, Feld 6 = 1110_{14} = $d\ c\ b\ \bar{a}$
Feld 15 hat die Zuordnung: $a\ b\ c\ d$, Feld 7 = 1111_{15} = $d\ c\ b\ a$



	d	c	b	a	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	

KV-Diagramme (Erstellung)

Funktion sei in Tabellenform gegeben:

Jede Zeile der Funktionstabelle entspricht einem Feld im KV-Diagramm.

⇒ Für jede Zeile der Funktionstabelle sucht man das zugehörige Feld im KV-Diagramm und trägt den Funktionswert ein.

Trick, um das Auffinden der Felder im KV-Diagramm zu erleichtern:

Man schreibt die Eingangsvariablen in „umgekehrter alphabetischer Reihenfolge“ in die Tabelle.

Dann kann das KV-Diagramm gemäß der Indizierung seiner Felder mit den Werten ausgefüllt werden, welche die Tabelle beim Durchlaufen von oben nach unten liefert.

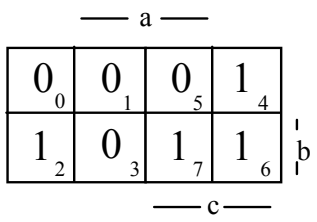
KV-Diagramme

Beispiel: $y = \bar{a}\ b \vee b\ c \vee \bar{a}\ \bar{b}\ c$

Funktionstabelle:

Index	c	b	a	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Damit ergibt sich das folgende KV-Diagramm:

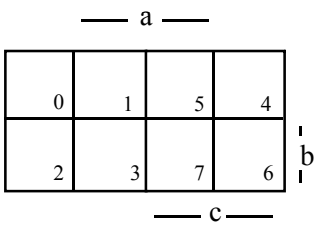


Eigenschaften der KV-Diagramme

Wesentliche Eigenschaft:

symmetrisch zu einer Achse liegende Minterme unterscheiden sich lediglich in einer Variablen.

Beispiel:



Minterm 0 = $\bar{c}\,\bar{b}\,\bar{a}$
Minterm 1 = $\bar{c}\,\bar{b}\,a$

oder Minterm 0 = $\bar{c}\,\bar{b}\,\bar{a}$
Minterm 4 = $c\,\bar{b}\,\bar{a}$

oder Minterm 1 = $\bar{c}\,\bar{b}\,a$
Minterm 3 = $\bar{c}\,b\,a$

Eigenschaften der KV-Diagramme

Nach den Regeln der Booleschen Algebra lassen sich Terme, die sich nur in einer Variablen unterscheiden, zusammenfassen:

Beispiel:

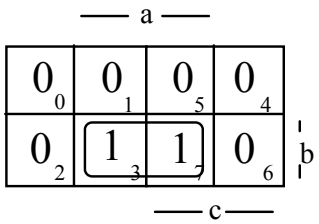
$$a\,b\,c \vee a\,b\,\bar{c} = a\,b\,(c \vee \bar{c}) = a\,b$$

Es entsteht ein Term ohne diese Variable.

Eigenschaften der KV-Diagramme

⇒ symmetrisch zu Achsen des KV-Diagramms liegende Minterme lassen sich zu einem einfacheren Term zusammenfassen.

$$a\,b\,c \vee a\,b\,\bar{c} = a\,b\,(c \vee \bar{c}) = a\,b$$



Herauslesen einer Funktion

Auffrischung:

Definition eines Implikanten:

f sei eine Funktion von $x_1, x_2, \ldots, x_n$ sowie g ein Produktterm aus Literalen dieser Variablen.

g ist Implikant von f, wenn g die Funktion f impliziert d.h. wenn die Menge aller Einsstellen von g in der Menge aller Einsstellen von f enthalten ist.

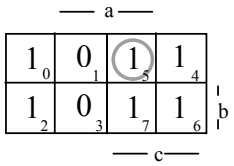
Herauslesen einer Funktion (1)

Beispiel: f = MINt (0,2,4,5,6,7)

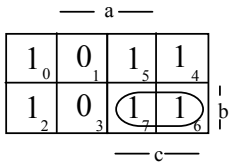
f:

1_0	0_1	1_5	1_4
1_2	0_3	1_7	1_6

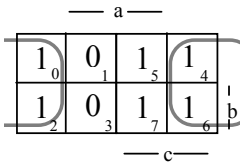
a, b, c



$g = a\,\bar{b}\,c$
ist Implikant



$g = b\,c$
ist Implikant



$g = \bar{a}$
ist Implikant

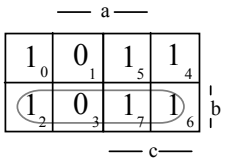
Herauslesen einer Funktion (2)

Beispiel: f = MINt (0,2,4,5,6,7)

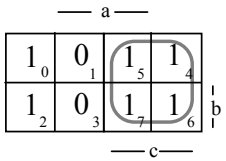
f:

1_0	0_1	1_5	1_4
1_2	0_3	1_7	1_6

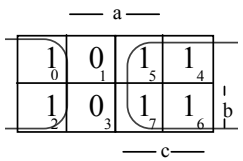
a, b, c



$g = b$
ist KEIN Implikant



$g = c$
ist Implikant



$g = \bar{a} \vee c$
ist KEIN Implikant

KV-Diagramme

Ein Implikant k-ter Ordnung umfasst 2^k Felder des KV-Diagramms

So erhält man

Implikanten 0. Ordnung: Minterme

Implikanten 1. Ordnung: Zusammenfassung von 2 Mintermen (z.B. b c im Beispiel)

Implikanten 2. Ordnung: Zusammenfassung zweier Implikanten 1. Ordnung (z.B. $\bar{a}$ oder c im Beispiel)

usw.

Definition: Primimplikant

Ein Implikant p ist Primimplikant, falls es keinen Implikanten $q \neq p$ gibt, der von p impliziert wird

$\forall q: \quad q \neq p \Leftrightarrow \neg(p \rightarrow q)$

d.h. p ist von größtmöglicher Ordnung (p umfasst einen maximal großen Einsblock).

Es gilt:
Jede Funktion ist als Disjunktion ihrer Primimplikanten darstellbar.

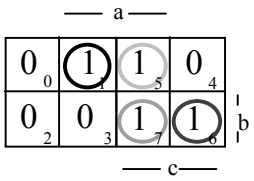
Kurz: Ist ein Implikant einer Boolschen Funktion in keinem anderen Implikanten vollständig enthalten, wird er als Primimplikant bezeichnet.

Herauslesen der Primimplikanten

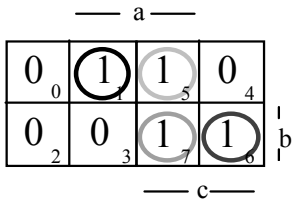
Herauslesen der Primimplikanten aus dem KV-Diagramm:
Man versucht, möglichst große Blöcke von Einsen im Diagramm zu finden, wobei jeder Einsblock 2^k Felder umfassen muss.

Beispiel:

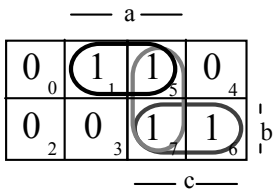
$g = \bar{a}\ b\ c \vee a\ c \vee a\ \bar{b}\ \bar{c}$



Beispiel: $g = \bar{a}\ b\ c \vee a\ c \vee a\ \bar{b}\ \bar{c}$



4 Minterme:
($a\ \bar{b}\ \bar{c}$, $a\ \bar{b}\ c$, $a\ b\ c$, $\bar{a}\ b\ c$)



3 Primimplikanten:
Implikanten erster Ordnung
($a\ \bar{b}$, $a\ c$, $b\ c$)

Aber: ($a\ \bar{b}$, $b\ c$) genügen eigentlich!

Minimierung einer zweistufigen Schaltfunktion

1.Schritt:
Bestimmung der Primimplikanten = Implikanten mit der geringsten möglichen Anzahl Literalen

⇒ Gatter mit der geringst möglichen Zahl an Eingängen

2. Schritt:
Auswahl einer minimalen Anzahl von Primimplikanten zur Überdeckung der Funktion.

⇒ minimale Anzahl von Gattern

Minimale Überdeckung

Bestimmung einer minimalen Überdeckung von Primimplikanten im KV-Diagramm:

Definition:
Ein Primimplikant ist ein Kernprimimplikant, wenn er einen Minterm der Funktion überdeckt, der von keinem anderen Primimplikanten überdeckt wird.

Kurz:
Enthält ein Primimplikant mindestens einen Min- oder Maxterm, der in keinem anderen Primimplikanten enthalten ist, bezeichnet man diesen als Kernprimimplikanten.

Kernprimimplikanten müssen in der disjunktiven Minimalform vorkommen.

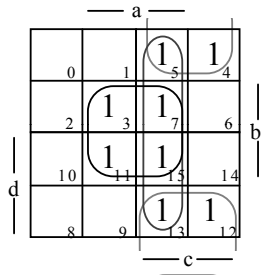
Disjunktive Minimalform

Die Funktion soll durch eine disjunktive Form aus Kernprimimplikanten und möglichst wenigen weiteren Primimplikanten überdeckt werden (irredundante Überdeckung).

Auswahl der Kernimplikanten und der geringstmöglichen Anzahl von weiteren Primimplikanten bei KV-Diagrammen: "durch Hinschauen"

⇒ disjunktive Minimalform

Beispiel: Disjunktive Minimalform



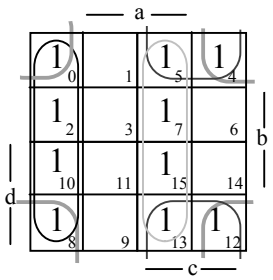
$f = a\,b \vee a\,c \vee \bar{b}\,c$

Primimplikanten : $a\,b$, $a\,c$, $\bar{b}\,c$

Kernprimimplikanten : $a\,b$, $\bar{b}\,c$

minimale Form : $f = a\,b \vee \bar{b}\,c$

Beispiel 2: Disjunktive Minimalform



$g = \bar{a}\,\bar{c} \vee a\,c \vee \bar{a}\,\bar{b}$

Primimplikanten : $\bar{a}\,\bar{c}$, $a\,c$, $\bar{b}\,c$, $\bar{a}\,\bar{b}$

Kernprimimplikanten : $\bar{a}\,\bar{c}$, $a\,c$

minimale Form : $g = \bar{a}\,\bar{c} \vee a\,c \vee \bar{b}\,c$

Bestimmung einer konjunktiven Minimalform aus dem KV-Diagramm

Im Prinzip wie bei der disjunktiven Minimalform

Es werden anstelle der Einsen die Nullen betrachtet.

Man sucht nun konjunktive Terme (Implikate) durch das Betrachten von Maxtermen und ihrer möglichen Zusammenfassungen.

Konjunktive Minimalform

Vorgehensweise:

1. Schritt:
   Zusammenfassung der größtmöglichen Blöcke von Nullen, wobei jeder Block 2^k Felder umfassen muss

⇒ Primimplikate

2. Schritt:
   Auswahl einer minimalen Anzahl von Primimplikaten, bestehend aus Kernprimimplikaten und weiteren Primimplikaten

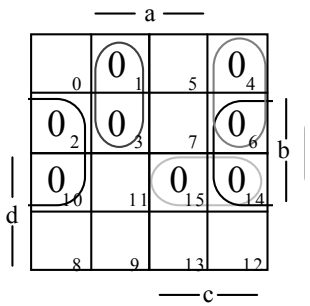
⇒ konjunktiven Minimalform

Konjunktive Minimalform

Besonderheit beim Ablesen der disjunktiven Terme:

Um den disjunktiven Term für einen 0-Block zu erhalten, verknüpft man die Variablen, die von dem Block *nicht* überdeckt werden, disjunktiv miteinander.

Konjunktive Minimalform



Maxterme 2, 6, 10, 14 ergeben $(a \vee \bar{b})$

Maxterme 1, 3 ergeben $(\bar{a} \vee c \vee d)$

Maxterme 4, 6 ergeben $(a \vee \bar{c} \vee d)$

Maxterme 14, 15 ergeben $(\bar{b} \vee \bar{c} \vee \bar{d})$

Damit ergibt sich die KMF für die Funktion:

$$g = (a \vee \bar{b})(\bar{a} \vee c \vee d)(a \vee \bar{c} \vee d)(\bar{b} \vee \bar{c} \vee \bar{d})$$

Maxterme 2, 3 ergeben $(\bar{b} \vee c \vee d)$ Diese sind jedoch entbehrlich!

Unvollständig definierte Funktionen

Bei den bisher betrachteten Funktionen war für jede mögliche Belegung der Eingangsvariablen ein Funktionswert definiert

➔ vollständig definierte Funktion

Es kommt jedoch vor, dass der Funktionswert nur für bestimmte Eingangsbelegungen definiert ist und die Funktionswerte der restlichen Belegungen frei wählbar sind

➔ unvollständig oder partiell definierte Funktion

Unvollständig definierte Funktionen

Die nicht verwendeten Eingangsbelegungen bezeichnet man als "don´t care"-Belegungen, man kann ihren Funktionswert beliebig zu 0 oder 1 verfügen.

FUNCTION TABLE

INPUTS					FUNCTION
SER	SRCLK	$\overline{SRCLR}$	RCLK	$\overline{OE}$	
X	X	X	X	H	Outputs Q_A–Q_H are disabled.
X	X	X	X	L	Outputs Q_A–Q_H are enabled.
X	X	L	X	X	Shift register is cleared.
L	↑	H	X	X	First stage of the shift register goes low. Other stages store the data of previous stage, respectively.
H	↑	H	X	X	First stage of the shift register goes high. Other stages store the data of previous stage, respectively.
X	X	X	↑	X	Shift-register data is stored in the storage register.

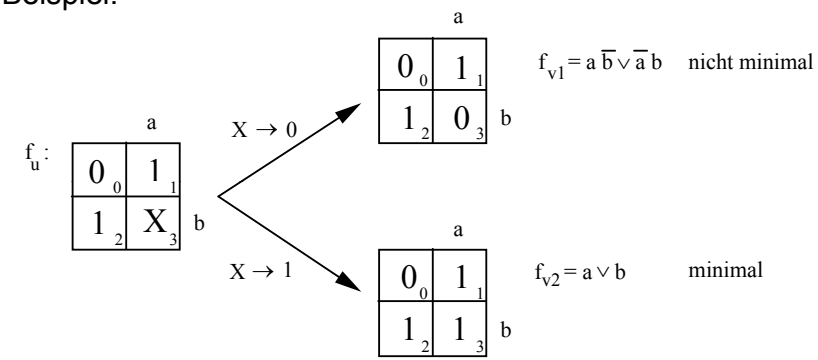
Ziel:

Vereinfachung des Funktionsausdrucks durch geschickte Wahl des Funktionswertes für "don´t care"-Belegungen.

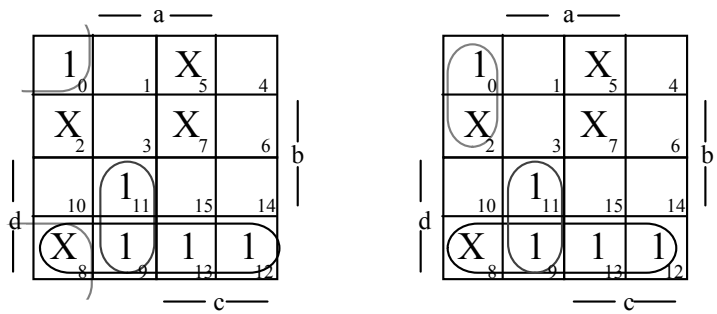
Um eine möglichst einfache DMF zu erhalten, muss man "don´t cares" so zu 0 oder 1 verfügen, dass möglichst große Blöcke von Primimplikanten entstehen.

Unvollständig definierte Funktionen

Beispiel:



Unvollständig definierte Funktionen



$f_u = \overline{b}\,c\,d \vee a\,\overline{c}\,d \vee \overline{a}\,\overline{b}\,\overline{c}\,\overline{d}$ (nur Einsstellen der unvollst. Def. Funktion f_u)

$f_{v1} = \overline{b}\,d \vee a\,\overline{c}\,d \vee \overline{a}\,\overline{b}\,\overline{c}$ (Feld 8 zu „1“ → vollst. def. Funktion f_{v1})

$f_{v2} = \overline{b}\,d \vee a\,\overline{c}\,d \vee \overline{a}\,\overline{c}\,\overline{d}$ (Felder 8 und 2 zu „1“ verfügt → f_{v2})

Zusammenfassung: Vorgehensweise beim Minimieren

Berechnung aller Primimplikanten (Primimplikate) der gegebenen Funktion

→ Gatter mit der geringst möglichen Zahl an Eingängen

Auswahl einer Menge von Primimplikanten (Primimplikate) zur Bildung der Minimalform:

Kernprimimplika(n)te(n) und möglichst wenigen Primimplika(n)ten zur Überdeckung der Funktion

→ Minimale Anzahl von Gattern

→ Minimierung der Chipfläche = Kostensenkung!

Einordnung und Ausblick

Aufgabenstellung	Primterme	Auswahl
Variablenanzahl ≤ 6	KV-Diagramm	KV-Diagramm Überdeckungstabelle
Geg. DF(KF) Ges. DMF(KMF)	Consensus Quine-McCluskey	Überdeckungstabelle
Geg. DF(KF) Ges. KMF(DMF)	Nelson	Überdeckungstabelle

Für erweiterte/leistungsfähigere Verfahren wird auf die Literatur verwiesen!

Allgemeine Problematik

Die Anzahl der Primimplikanten kann exponentiell mit der Anzahl der Eingabevariablen steigen ($3^n/n$ Primimplikanten)

Das Überdeckungsproblem ist NP-vollständig

- Es besteht wenig Hoffnung, einen Algorithmus zu finden, der das Problem in einer Zeit löst, die polynomial mit der Anzahl der Variablen wächst.
- Meist liegt exponentielles Wachstum vor.

Heuristische Verfahren

- Hierbei werden nicht notwendigerweise minimale Lösungen mit akzeptablem Zeit- und Speicherbedarf erzeugt.

Erweiterter Minimierungsansatz

Es wird versucht, mehrere Boolesche Funktionen gemeinsam zu minimieren

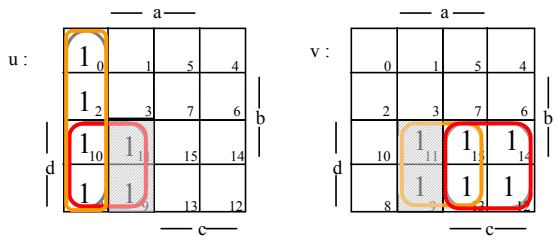
- Implikanten werden mehrfach ausgenutzt.
- Diese Implikanten müssen nicht notwendigerweise Primimplikanten der einzelnen Funktionen sein.

Bündelminimierung

- Prinzip im KV-Diagramm:
- Man sucht nach Implikanten, die in mehreren KV-Diagrammen vorkommen.
- Diese Koppelterme müssen für mehrere Funktionen nur einmal realisiert werden und sparen dadurch Kosten.

Bündelminimierung

Beispiel:



Ohne Koppelterme:

$u = \bar{a}\,\bar{c} \vee \bar{c}\,d$ $\qquad$ $v = a\,d \vee c\,d$

Mit Koppeltermen:

$u = \bar{a}\,\bar{c} \vee a\,\bar{c}\,d$ $\qquad$ $v = c\,d \vee a\,\bar{c}\,d$

Primkoppelterm

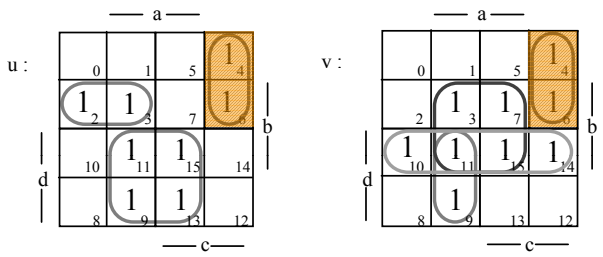
Koppelterme müssen keine Primimplikanten sein (siehe Beispiel).

Sie lassen sich im Allgemeinen weiter vereinfachen, jedoch für jede Funktion in unterschiedlicher Weise.

Koppelterme, die gleichzeitig Primimplikanten einer Funktion sind, heißen Primkoppelterme.

Primkoppelterm

Beispiel:



$u = a\,d \vee \bar{a}\,c\,\bar{d} \vee b\,\bar{c}\,\bar{d}$

$v = b\,d \vee \bar{a}\,c\,\bar{d} \vee a\,b \vee a\,\bar{c}\,d$ $\bar{a}\,c\,\bar{d}$ ist Primkoppelterm

Logikentwurf

Abbildung einer abstrakten logischen Form auf realisierbare Bauelemente.

Beispiel:

Erhält man als Ergebnis der Minimierung einen UND-Term mit 17 Variablen, so kann man diesen Term nicht einfach auf ein UND-Gatter mit 17 Eingängen abbilden.

Die technische Realisierbarkeit muss als Randbedingung beim Entwurf eingehalten werden.

Um auch diesen Schritt zu automatisieren, werden meist Systeme verwendet, die bestimmte logische Formen nach festen Regeln auf Bausteine einer vorgegebenen Bibliothek abbilden.

Logikentwurf

Bisher: zweistufige Realisierung von Schaltnetzen
erst UND-Gatter dann ODER-Gatter - oder umgekehrt (KF bzw. DF)

Kurze Signallaufzeiten (damit hoher Takt! – später dazu mehr)

Minimierungsverfahren:
1. möglichst wenig Gattereingänge sowie
2. möglichst wenig UND- und ODER-Gatter (bzw. NAND, NOR)

Unterschiedliche Bausteinformate (AND3, AND4, OR5) ergeben Probleme für die Automatisierung des Chip-Designs

Verwenden von komplexeren Standardbausteinen:
Realisierung logischer Funktionen durch Multiplexer/Demultiplexer/Decoder
Sowie programmierbare Logikbausteine PLA, FPLA, PAL, ...

Logikentwurf auf Basis spezieller Strukturen

Anstelle aus logischen Gattern (UND, ODER, NICHT, NAND, NOR, ... usw.) lassen sich Schaltnetze auch mit komplexeren Standardbausteinen realisieren.

Einfachere und oft flexiblere Realisierung

Minimierung bedeutet hierbei dann nicht die Reduktion von Gattern.

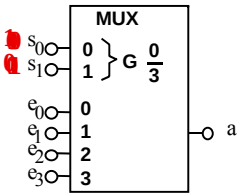
Es muss die Anzahl und Größe komplexer Bausteine reduziert werden.

Logische Funktionen mit Multiplexern
Multiplexer

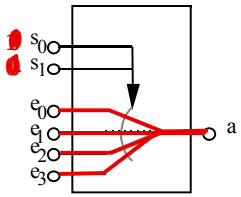
Ein Multiplexer (Abk.: MUX) ist ein Baustein mit mehreren Eingängen und einem Ausgang, wobei über n Steuerleitungen einer der 2^n Eingänge auf den Ausgang geschaltet wird.

Multiplexer werden nach ihrer Größe als 2^n: 1 - Multiplexer (alternativ als 1-aus-2^n- Multiplexer) klassifiziert.

Logische Funktionen mit Multiplexern
Verhalten eines 1-aus-4-Multiplexers



s_1	s_0	a
0	0	e_0
0	1	e_1
1	0	e_2
1	1	e_3



Ein Multiplexer kann nicht nur zur Steuerung von Datenflüssen sondern auch zur Realisierung logischer Funktionen verwendet werden.

Man kann mit einem 2n: 1 - Multiplexer eine logische Funktion mit n+1 Variablen implementieren.

Hierzu wird die sog. Implementierungstabelle verwendet.

Logische Funktionen mit Multiplexern
Implementierungstabelle

Die Tabelle besteht aus:

- 2^n Spalten für die möglichen Belegungen der n Steuereingänge
- 2 Zeilen für die negierte und nicht negierte (n+1)-te Variable

In die Tabelle werden die Funktionswerte in Abhängigkeit von den Variablen eingetragen.

Anschließend betrachtet man jede Spalte für sich und ordnet ihr eine einstellige Funktion $g \in \{0, 1, a, \bar{a}\}$ zu, mit der dann der Eingang belegt wird, der zu der entsprechenden Steuervariablenkombination gehört.

Logische Funktionen mit Multiplexern
Beispiel 1

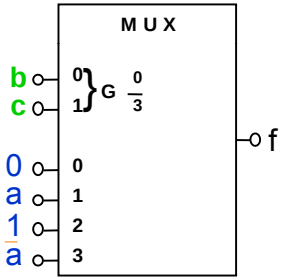
Realisierung einer Funktion mit Multiplexer:

$$f = \bar{a}\, c \vee \bar{b}\, c \vee a\, b\, \bar{c}$$

Implementierungstabelle bei Wahl von b und c als Steuereingänge:

c b	0 0	0 1	1 0	1 1
a = 0	0	0	1	1
a = 1	0	1	1	0
g =	0	a	1	$\bar{a}$

Realisierung



Logische Funktionen mit Multiplexern
Beispiel 2

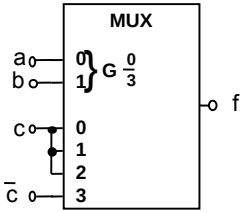
Realisierung einer Funktion mit Multiplexer:

$$f = \bar{a}\, c \vee \bar{b}\, c \vee a\, b\, \bar{c}$$

Implementierungstabelle bei Wahl von a und b als Steuereingänge :

b a	0 0	0 1	1 0	1 1
c = 0	0	0	0	1
c = 1	1	1	1	0
g =	c	c	c	$\bar{c}$

Realisierung :



Shannonscher Entwicklungssatz

Man kann auch Teilfunktionen mit einem Multiplexer realisieren.

Hierzu verwendet man den Shannonschen Entwicklungssatz.

Wdh.: Shannonentwicklung nach c:

$$f(a,b,c) = (c \wedge f(a,b,1)) \vee (\bar{c} \wedge f(a,b,0))$$

Vorgehensweise:

Die Funktion wird nach allen Steuervariablen des Multiplexers entwickelt, die verbleibenden Restfunktionen sind an die entsprechenden Eingänge des Multiplexers zu führen.

Beispiel

Die Funktion

$$f = \bar{a}\, c \vee \bar{b}\, c \vee a\, b\, \bar{c}$$

soll mit einem 2:1 - Multiplexer und Gattern entworfen werden.
Steuervariable des Multiplexers sei c

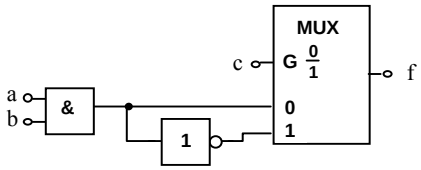
Entwicklung nach c :

$$f = c \cdot (\bar{a} \vee \bar{b}) \vee \bar{c} \cdot a\, b$$

$$f(c=1) = \bar{a} \vee \bar{b} = \overline{a\, b}$$

$$f(c=0) = a\, b$$

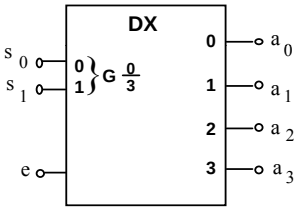
Realisierung:



Logische Funktionen mit Dekoder
Demultiplexer / Dekoder

Der zum Multiplexer korrespondierende Baustein, der einen Eingang abhängig von n Steuerleitungen auf einen von 2^n Ausgängen schaltet, heißt Demultiplexer.

Beispiel:



s_1	s_0	a_0	a_1	a_2	a_3
0	0	e	0	0	0
0	1	0	e	0	0
1	0	0	0	e	0
1	1	0	0	0	e

Schaltbild und logisches Verhalten eines 1-auf-4-Demultiplexers

Logische Funktionen mit Dekoder
Demultiplexer / Dekoder

Man beachte:

der Demultiplexer hat einen Enable-Eingang e sowie n Eingänge s_i für eine Dualzahl, die an den 2^n Ausgängen a_j dekodiert bereitgestellt wird.

Enable-Eingang e = 0, dann liegen alle Ausgänge auf 0

ansonsten wird eine 2-bit-Zahl dekodiert, z.B. wird bei Anlegen der Zahl 2 (s_1= 1, s_0= 0)
der Ausgang a_2= 1 und alle anderen Ausgänge bleiben 0.

Der Demultiplexer wird deshalb auch Dekoder genannt.

Logische Funktionen mit Dekoder
Realisierung logischer Funktionen

Man beachte: ein Dekoder erzeugt alle 2^n Minterme seiner n Steuervariablen.

Beispiel:

Berechnung der Anzahl Einsen: f(a,b,c) = a + b + c

Summe s $= a \leftrightarrow b \leftrightarrow c$
$= a\,\bar{b}\,\bar{c} \vee \bar{a}\,b\,\bar{c} \vee a\,b\,c \vee \bar{a}\,\bar{b}\,c$
= MINt(1,2,4,7)

Übertrag ü $= a\,b \vee a\,c \vee b\,c$
= MINt(3,5,6,7)

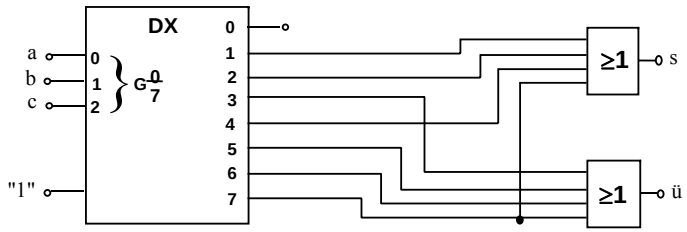
c	b	a	s	ü
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Funktion realisiert einen ein Volladdierer:
S = A + B + C
S – Summe einer 1Bit Addition von A und B und C
C – Übertrag aus der vorhergehenden Addition
Ü – Übertrag aus der Addition von A und B und C

Logische Funktionen mit Dekoder
Realisierung logischer Funktionen

$s = a \leftrightarrow b \leftrightarrow c$
$= a\,\bar{b}\,\bar{c} \vee \bar{a}\,b\,\bar{c} \vee a\,b\,c \vee \bar{a}\,\bar{b}\,c$
= MINt(1,2,4,7)

$ü = a\,b \vee a\,c \vee b\,c$
= MINt(3,5,6,7)



Logische Funktionen mit Speicher
Realisierung mittels Speicherbausteinen

Bei den bisher behandelten Bausteinen (Gatter, Multiplexer, Decoder) war die Funktion fest vorgegeben.

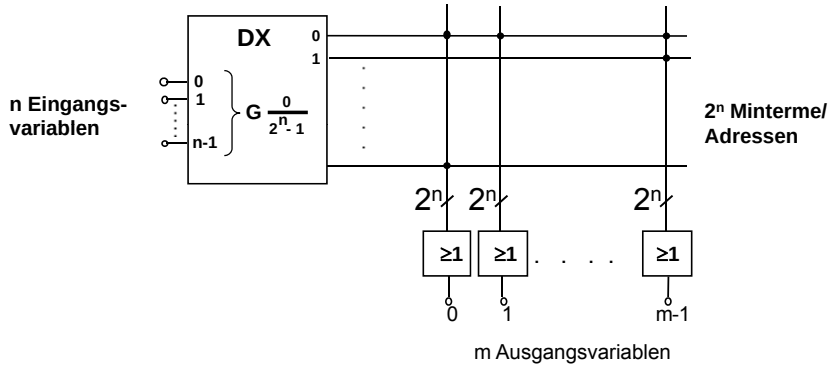
➔ festverdrahtete Logik

Höherintegrierte Verknüpfungsbausteine müssen die Flexibilität bieten, an viele verschiedene Anwendungen anpassbar zu sein. Diese Anpassung wird als Personalisierung oder als Programmierung bezeichnet.

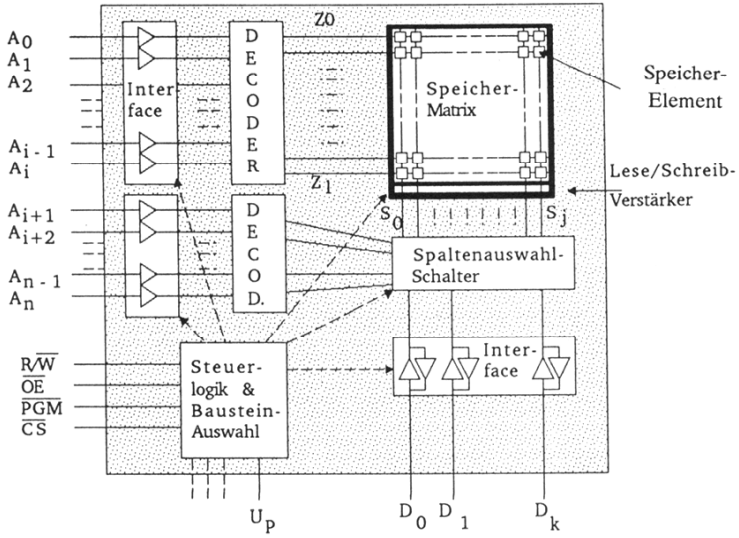
➔ mikroprogrammierte Logik
(siehe Rechnerorganisation für entsprechende CPUs)

Logische Funktionen mit Speicher
Aufbau eines Speicherbausteins

Speicheranordnung, in der beliebige Funktionstabellen abgelegt werden.



Logische Funktionen mit Speicher
Organisation von Speicherbausteinen



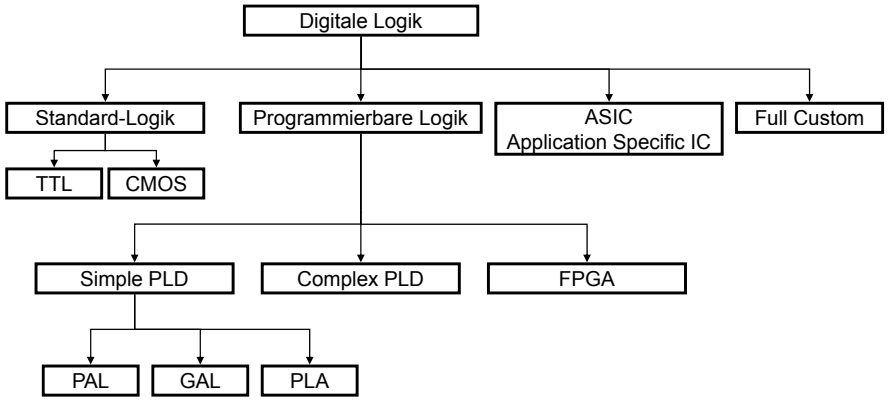
Logische Funktionen mit Speicher
Erläuterung zum Speicherbaustein

Durch das Anlegen von Eingangssignalen wird eine Speicherzelle ausgewählt (adressiert) und der dort gespeicherte Funktionswert an den Ausgängen zur Verfügung gestellt.

Die Leitungen, die den Dekoder verlassen, entsprechen also den Mintermen von n Eingangsvariablen, also den Zeilen der Funktionstabelle.

Das Speichern einer 1 für eine bestimmte Ausgangsvariable i bedeutet, dass dieser Minterm in die ODER-Verknüpfung am i-ten Ausgang einbezogen wird, eine 0 heißt, dass der Minterm nicht benutzt wird.

Logische Funktionen mit PLD
Programmierbarer Logik



Programmierbare Logik mit PLD
Begriffe

Bisher wurde die gesamte Funktionstabelle in einem Speicherbausteinen abgespeichert und die Funktion durch ihre DNF realisiert.

Verwendet man stattdessen die DMF, lassen sich Funktionen oft sehr viel kompakter darstellen.

Simple PLD (Programmable Logic Device) meist mit kleinem Funktionsumfang.

PLA (Programmable Logic Array): Funktion in DNF nachgebildet, AND- und OR-Terme freiprogrammierbar

PAL (Programmable Array Logic): PLA mit festverdrahteter OR-Logik, AND-Logik freiprogrammierbar, erweitert mit FlipFlop und Treiber

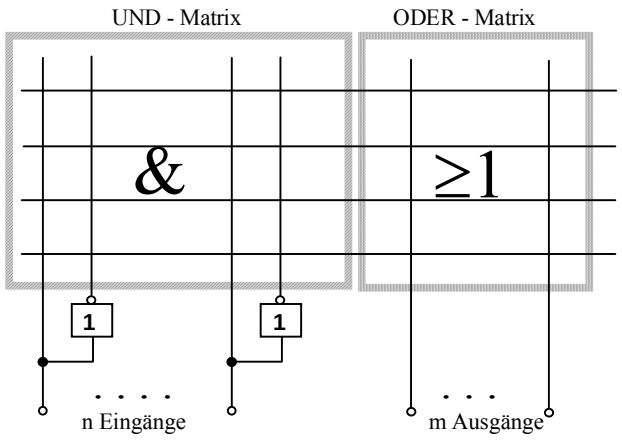
GAL (Gate Array Logic): bilden PAL nach, EEPROM, wiederprogrammierbar

PAL und GAL sind Sonderfälle der PLA da nur programmierbare AND-Logik und eine festverdrahtete OR-Logik integriert ist.

Complex PLD (Complex Programmable Logic Device) meist mit großem Funktionsumfang.

Heute werden vorrangig CPLD eingestetzt.

Programmierbare Logik mit PLD
Schematische Darstellung eines PLA

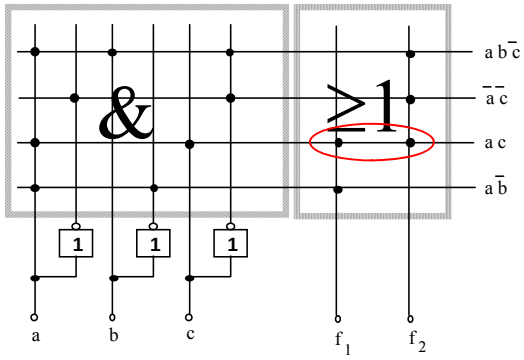


Programmierbare Logik mit PLD
Beispiel

Die (schon minimierten) Funktionen

$$f_1 = a\,\bar{b} \vee a\,c \quad \text{und} \quad f_2 = \bar{a}\,\bar{c} \vee a\,c \vee a\,\bar{b}\,c$$

sollen mit einem PLA realisiert werden.

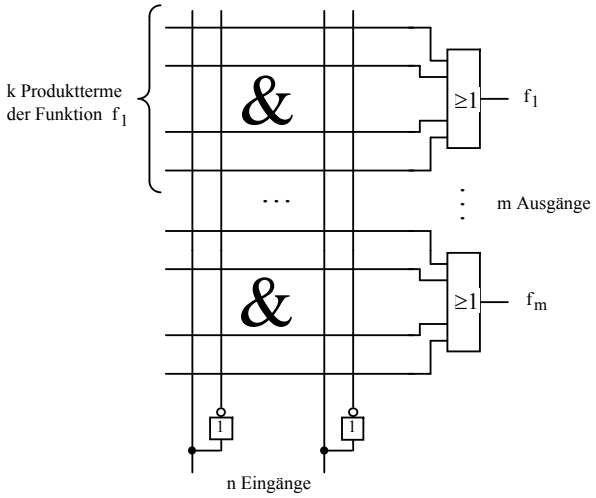


Programmierbare Logik mit PLD
Bündelminimierung

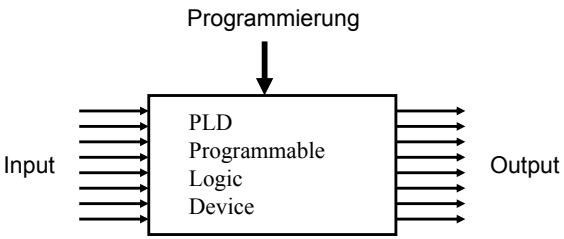
Der Term a c im letzten Beispiel konnte für beide Funktionen verwendet werden, wodurch sich die Anzahl der notwendigen Produktterme reduziert.

Allgemein sollte beim Minimieren mehrerer Funktionen (Funktionsbündeln) immer darauf geachtet werden, nicht jede Funktion für sich zu minimieren, sondern den Gesamtaufwand für alle Funktionen zu optimieren (Bündelminimierung).

Programmierbare Logik mit PLD
Schematische Darstellung eines PAL

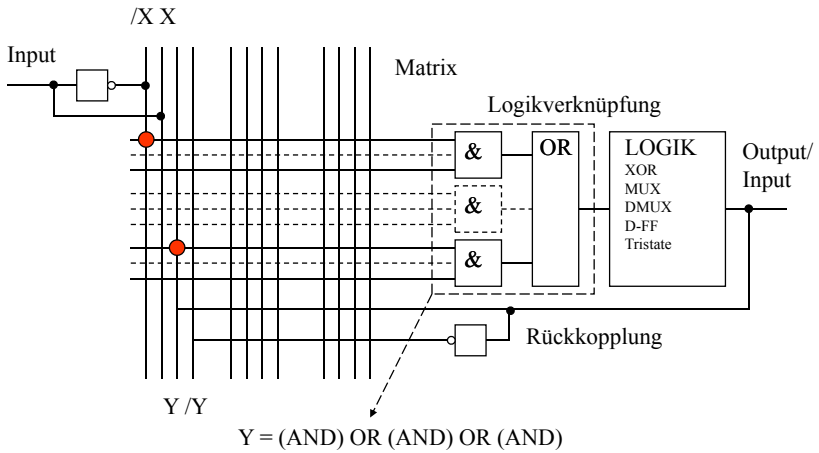


Programmierbare Logik mit PLD
Programmable Logic Device

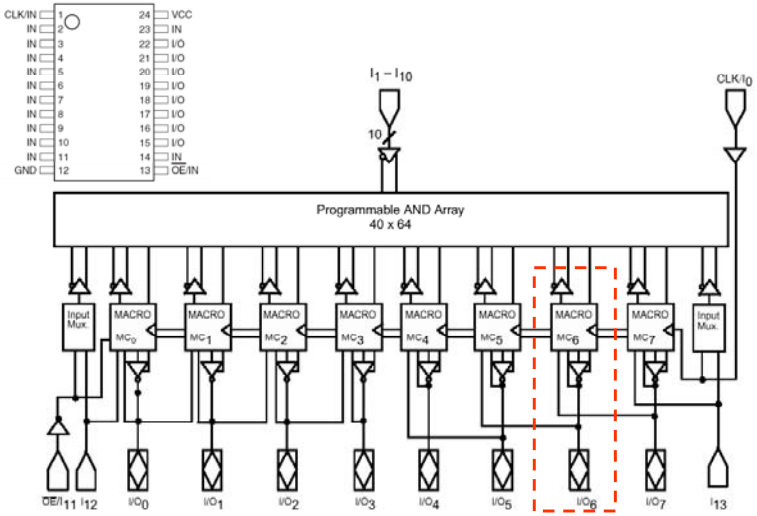


PLD bestehen aus folgenden Elementen:
- Programmierbare AND/OR- Matrix
- Programmierbare Rückkopplung
- Eingabeblock
- Ausgabeblock

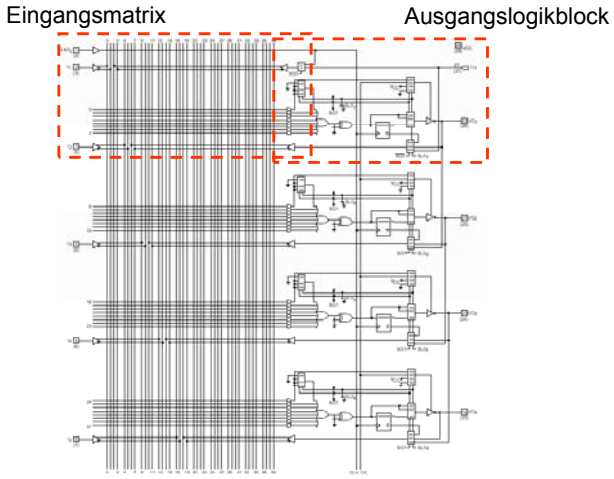
Logische Funktionen mit PLD
Grundgedanke



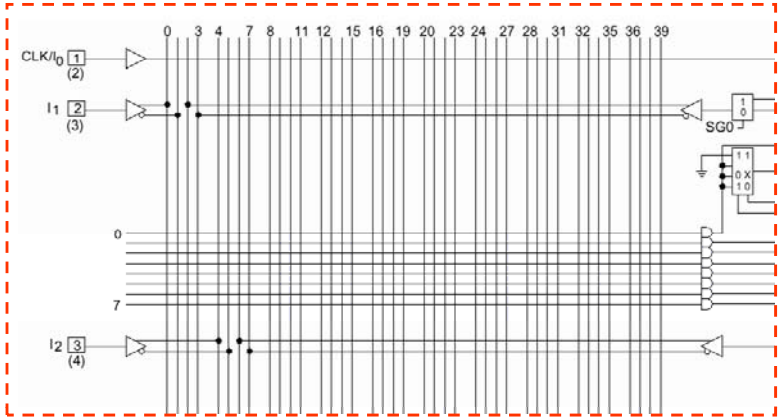
GAL20V8
Blockschaltbild



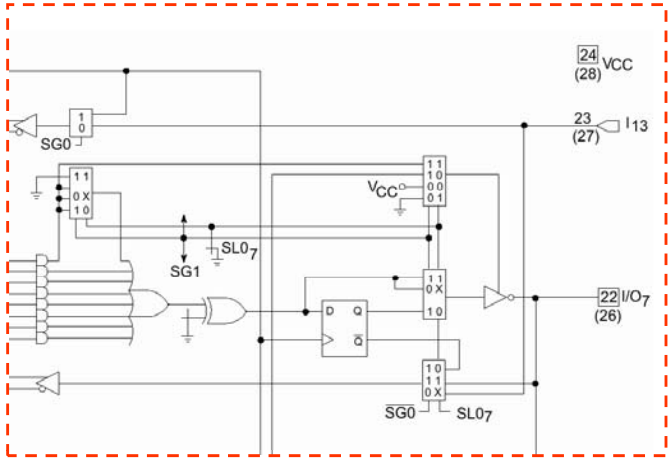
GAL20V8
Innenschaltung



GAL20V8
Eingangsmatrix



GAL20V8
Ausgangslogikblock

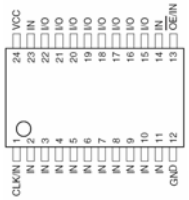


GAL20V8
Beispiel: Multiplexer mit Tristate Ausgang

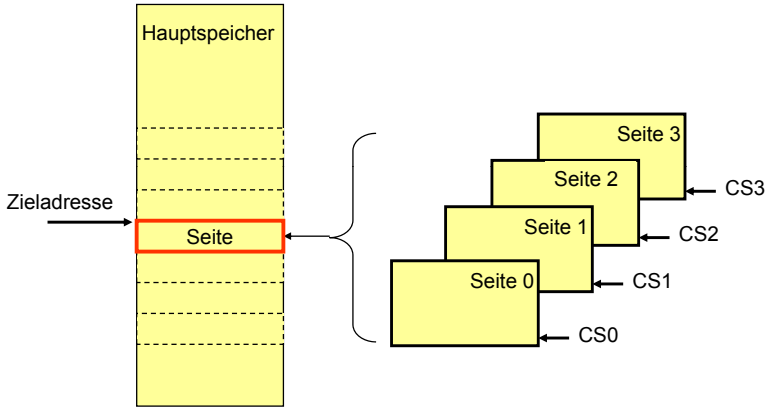
Start Quelltext | Name | Chiptype | Betriebsmode | PIN- und Variablen-festlegung

```
CHIP    MUX    GAL20V8A    COMPLEX_MODE

;PIN    1   2   3   4   5   6   7   8   9   10  11  12
        X0  X1  X2  X3  A0  A1  OE  NC  NC  NC  NC  GND     ;Belegung
        NC  NC  Y   NC  NC  NC  NC  NC  NC  NC  NC  VCC     ;Belegung
;PIN    13  14  15  16  17  18  19  20  21  22  23  24

Y =      X0 * /A0 * /A1
       + X1 * A0 * /A1
       + X2 * /A0 * A1
       + X3 * A0 * A1
Y.TRST = OE
```

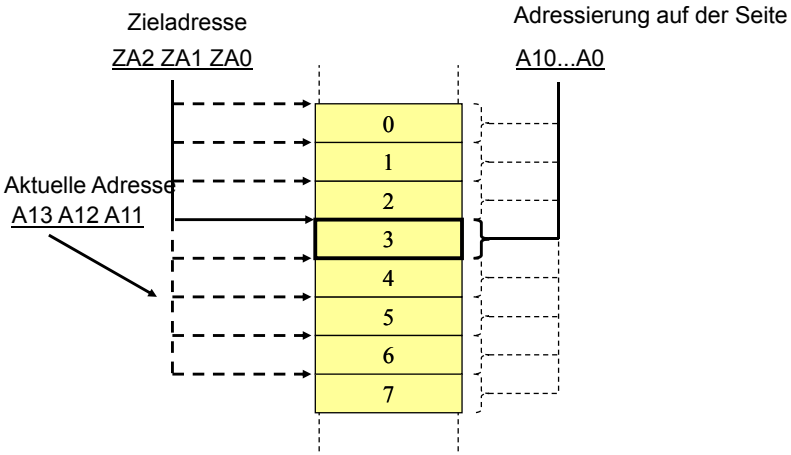
Logikausdruck



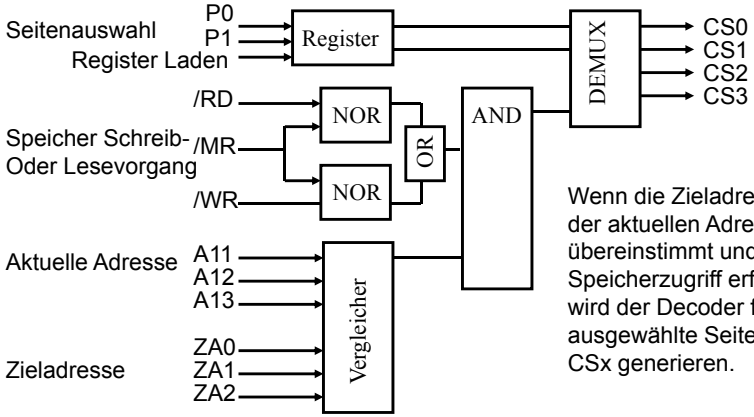
GAL20V8
Beispiel: Seiteneinblendung im Speicher



GAL20V8
Beispiel: Seiteneinblendung im Speicher



GAL20V8
Beispiel: Seiteneinblendung im Speicher



Wenn die Zieladresse mit der aktuellen Adresse übereinstimmt und ein Speicherzugriff erfolgt, wird der Decoder für die ausgewählte Seite das CSx generieren.

GAL20V8
Beispiel: Seiteneinblendung im Speicher

Speicherzugriff erfolgt bei RD=0 und MR=0 oder WR=0 und MR=0

$$M = \overline{RD} \cdot \overline{MR} + \overline{WR} \cdot \overline{MR}$$

Die Zieladresse wird aus dem Vergleich von ZA3...ZA0 (Zieladresse) im Speicherband und den aktuellen Adressbits A13..A11 gewonnen.

$$ZA = \left(ZA2 \cdot \overline{A13}\right) + \left(\overline{ZA2} \cdot A13\right) \cdot \left(ZA1 \cdot \overline{A12}\right) + \left(\overline{ZA1} \cdot A12\right) \cdot \left(ZA0 \cdot \overline{A11}\right) + \left(\overline{ZA0} \cdot A11\right)$$

Das Chipselect CS3..CS0 wird aus den Page Bits P1..P0, dem Speicherzugriff und den Zieladressen gewonnen.

$$CS0 = \overline{P1} \cdot \overline{P0} \cdot ZA \cdot M$$

$$CS1 = \overline{P1} \cdot P0 \cdot ZA \cdot M$$

$$CS2 = P1 \cdot \overline{P0} \cdot ZA \cdot M$$

$$CS3 = P1 \cdot P0 \cdot ZA \cdot M$$

GAL20V8
Beispiel: Seiteneinblendung im Speicher

```
CHIP      SPEICHER          GAL20V8A          COMPLEX_MODE

#DEFINE M /RD * /MR + /WR * /MR
#DEFINE A /ZA1 * /A11 + ZA1 * A11
#DEFINE B /ZA2 * /A12 + ZA2 * A12
#DEFINE C /ZA3 * /A13 + ZA3 * A13
 ;PIN     1    2    3    4    5    6    7    8    9    10   11   12
          CLK  P0   P1   RD   WR   MR   ZA1  ZA2  ZA3  A11  A12  GND  ;Belegung
          /OE  NC   PR0  PR1  NC   NC   CS0  CS1  CS2  CS3  A13  VCC  ;Belegung
;PIN      13   14   15   16   17   18   19   20   21   22   23   24
PR0 := P0
PR1 := P1
CS0 = /PR1 * /PR0 * A * B * C * M
CS1 = /PR1 *  PR0 * A * B * C * M
CS2 =  PR1 * /PR0 * A * B * C * M
CS3 =  PR1 *  PR0 * A * B * C * M
```