

64-040 Modul InfB-RS: Rechnerstrukturen

[https://tams.informatik.uni-hamburg.de/
lectures/2015ws/vorlesung/rs](https://tams.informatik.uni-hamburg.de/lectures/2015ws/vorlesung/rs)

– Kapitel 8 –

Norman Hendrich



Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2015/2016



Kapitel 8

Schaltfunktionen

Definition

Darstellung

Normalformen

Entscheidungsbäume und OBDDs

Realisierungsaufwand und Minimierung

Minimierung mit KV-Diagrammen

Literatur

Schaltfunktionen

- ▶ **Schaltfunktion:** eine eindeutige Zuordnungsvorschrift f , die jeder Wertekombination (b_1, b_2, \dots, b_n) von Schaltvariablen einen Wert zuweist:

$$y = f(b_1, b_2, \dots, b_n) \in \{0, 1\}$$

- ▶ **Schaltvariable:** eine Variable, die nur endlich viele Werte annehmen kann – typisch sind binäre Schaltvariablen
- ▶ **Ausgangvariable:** die Schaltvariable am Ausgang der Funktion, die den Wert y annimmt
- ▶ bereits bekannt: *elementare Schaltfunktionen* (AND, OR, usw.)
wir betrachten jetzt Funktionen von n Variablen

Beschreibung von Schaltfunktionen

- ▶ textuelle Beschreibungen
formale Notation, Schaltalgebra, Beschreibungssprachen

- ▶ tabellarische Beschreibungen
Funktionstabelle, KV-Diagramme, ...

- ▶ graphische Beschreibungen
Kantorovic-Baum (Datenflussgraph), Schaltbild, ...

- ▶ Verhaltensbeschreibungen \Rightarrow „was“
- ▶ Strukturbeschreibungen \Rightarrow „wie“

Funktionsstabelle

- ▶ Tabelle mit Eingängen x_i und Ausgangswert $y = f(x)$
- ▶ Zeilen im Binärcode sortiert
- ▶ zugehöriger Ausgangswert eingetragen

x_3	x_2	x_1	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Funktionstabelle (cont.)

- ▶ Kurzschreibweise: nur die Funktionswerte notiert

$$f(x_2, x_1, x_0) = \{0, 0, 1, 1, 0, 0, 1, 0\}$$

- ▶ n Eingänge: Funktionstabelle umfasst 2^n Einträge

- ▶ Speicherbedarf wächst exponentiell mit n

z.B.: 2^{33} Bit für 16-bit Addierer (16+16+1 Eingänge)

⇒ daher nur für kleine Funktionen geeignet

- ▶ Erweiterung auf *don't-care* Terme, s.u.



Verhaltensbeschreibung

- ▶ Beschreibung einer Funktion als Text über ihr Verhalten
- ▶ Problem: umgangssprachliche Formulierungen oft mehrdeutig
- ▶ logische Ausdrücke in Programmiersprachen
- ▶ Einsatz spezieller (Hardware-) Beschreibungssprachen
z.B.: Verilog, VHDL, SystemC

umgangssprachlich: Mehrdeutigkeit

„Das Schiebedach ist ok (y), wenn der Öffnungskontakt (x_0) **oder** der Schließkontakt (x_1) funktionieren **oder beide nicht** aktiv sind (Mittelstellung des Daches)“

K. Henke, H.-D. Wuttke: *Schaltsysteme* [WH03]

zwei mögliche Missverständnisse

- ▶ *oder*: als OR oder XOR?
- ▶ *beide nicht*: x_1 und x_0 nicht, oder x_1 nicht und x_0 nicht?

⇒ je nach Interpretation völlig unterschiedliche Schaltung



Strukturbeschreibung

- ▶ **Strukturbeschreibung:** eine Spezifikation der konkreten Realisierung einer Schaltfunktion

- ▶ vollständig geklammerte algebraische Ausdrücke

$$f = x_1 \oplus (x_2 \oplus x_3)$$
- ▶ Datenflussgraphen
- ▶ Schaltpläne mit Gattern (s.u.)
- ▶ PLA-Format für zweistufige AND-OR Schaltungen (s.u.)
- ▶ ...



Funktional vollständige Basismenge

- ▶ Menge M von Verknüpfungen über $GF(2)$ heißt **funktional vollständig**, wenn die Funktionen $f, g \in T_2$:

$$f(x_1, x_2) = x_1 \oplus x_2$$

$$g(x_1, x_2) = x_1 \wedge x_2$$

allein mit den in M enthaltenen Verknüpfungen geschrieben werden können

- ▶ Boole'sche Algebra: { AND, OR, NOT }
- ▶ Reed-Muller Form: { AND, XOR, 1 }
- ▶ technisch relevant: { NAND }, { NOR }



Normalformen

- ▶ Jede Funktion kann auf beliebig viele Arten beschrieben werden

Suche nach Standardformen

- ▶ in denen man alle Funktionen darstellen kann
- ▶ Darstellung mit universellen Eigenschaften
- ▶ eindeutige Repräsentation \Rightarrow einfache Überprüfung, ob gegebene Funktionen übereinstimmen
- ▶ Beispiel: Darstellung von reellen Funktionen als Potenzreihe

$$f(x) = \sum_{i=0}^{\infty} a_i x^i$$

Normalformen (cont.)

- ▶ Darstellung von reellen Funktionen als Potenzreihe

$$f(x) = \sum_{i=0}^{\infty} a_i x^i$$

Normalform einer Boole'schen Funktion

- ▶ analog zur Potenzreihe
- ▶ als Summe über Koeffizienten $\{0, 1\}$ und Basisfunktionen

$$f = \sum_{i=1}^{2^n} \hat{f}_i \hat{B}_i, \quad \hat{f}_i \in \text{GF}(2)$$

mit $\hat{B}_1, \dots, \hat{B}_{2^n}$ einer Basis des T^n

Definition: Normalform

- ▶ funktional vollständige Menge V der Verknüpfungen von $\{0, 1\}$
- ▶ Seien $\oplus, \otimes \in V$ und assoziativ

- ▶ Wenn sich alle $f \in T^n$ in der Form

$$f = (\hat{f}_1 \otimes \hat{B}_1) \oplus \cdots \oplus (\hat{f}_{2^n} \otimes \hat{B}_{2^n})$$

schreiben lassen, so wird die Form als **Normalform** und die Menge der \hat{B}_i als **Basis** bezeichnet.

- ▶ Menge von 2^n Basisfunktionen \hat{B}_i
 Menge von 2^{2^n} möglichen Funktionen f

Disjunktive Normalform (DNF)

- ▶ **Minterm:** die UND-Verknüpfung *aller* Schaltvariablen einer Schaltfunktion, die Variablen dürfen dabei negiert oder nicht negiert auftreten
- ▶ **Disjunktive Normalform:** die disjunktive Verknüpfung aller Minterme m mit dem Funktionswert 1

$$f = \bigvee_{i=1}^{2^n} \hat{f}_i \cdot m(i), \quad \text{mit } m(i) : \text{Minterm}(i)$$

auch: *kanonische disjunktive Normalform*
sum-of-products (SOP)



Disjunktive Normalform: Minterme

- ▶ Beispiel: alle 2^3 Minterme für drei Variablen
- ▶ jeder Minterm nimmt nur für eine Belegung der Eingangsvariablen den Wert 1 an

x_3	x_2	x_1	Minterme
0	0	0	$\overline{x_3} \wedge \overline{x_2} \wedge \overline{x_1}$
0	0	1	$\overline{x_3} \wedge \overline{x_2} \wedge x_1$
0	1	0	$\overline{x_3} \wedge x_2 \wedge \overline{x_1}$
0	1	1	$\overline{x_3} \wedge x_2 \wedge x_1$
1	0	0	$x_3 \wedge \overline{x_2} \wedge \overline{x_1}$
1	0	1	$x_3 \wedge \overline{x_2} \wedge x_1$
1	1	0	$x_3 \wedge x_2 \wedge \overline{x_1}$
1	1	1	$x_3 \wedge x_2 \wedge x_1$

Disjunktive Normalform: Beispiel

x_3	x_2	x_1	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- ▶ Funktionstabelle: Minterm $0 \equiv \bar{x}_i$ $1 \equiv x_i$
 - ▶ für f sind nur drei Koeffizienten der DNF gleich 1
- ⇒ DNF: $f(x) = (\bar{x}_3 \wedge x_2 \wedge \bar{x}_1) \vee (\bar{x}_3 \wedge x_2 \wedge x_1) \vee (x_3 \wedge x_2 \wedge \bar{x}_1)$



Allgemeine disjunktive Form

- ▶ **disjunktive Form** (sum-of-products): die disjunktive Verknüpfung (ODER) von Termen. Jeder Term besteht aus der UND-Verknüpfung von Schaltvariablen, die entweder direkt oder negiert auftreten können
- ▶ entspricht dem Zusammenfassen („Minimierung“) von Termen aus der disjunktiven Normalform
- ▶ disjunktive Form ist nicht eindeutig (keine Normalform)

▶ Beispiel

$$\text{DNF} \quad f(x) = (\overline{x_3} \wedge x_2 \wedge \overline{x_1}) \vee (\overline{x_3} \wedge x_2 \wedge x_1) \vee (x_3 \wedge x_2 \wedge \overline{x_1})$$

$$\text{minimierte disjunktive Form} \quad f(x) = (\overline{x_3} \wedge x_2) \vee (x_3 \wedge x_2 \wedge \overline{x_1})$$



Allgemeine disjunktive Form

- ▶ **disjunktive Form** (sum-of-products): die disjunktive Verknüpfung (ODER) von Termen. Jeder Term besteht aus der UND-Verknüpfung von Schaltvariablen, die entweder direkt oder negiert auftreten können
- ▶ entspricht dem Zusammenfassen („Minimierung“) von Termen aus der disjunktiven Normalform
- ▶ disjunktive Form ist nicht eindeutig (keine Normalform)

▶ Beispiel

$$\text{DNF} \quad f(x) = (\overline{x_3} \wedge x_2 \wedge \overline{x_1}) \vee (\overline{x_3} \wedge x_2 \wedge x_1) \vee (x_3 \wedge x_2 \wedge \overline{x_1})$$

$$\text{minimierte disjunktive Form} \quad f(x) = (\overline{x_3} \wedge x_2) \vee (x_3 \wedge x_2 \wedge \overline{x_1})$$

$$f(x) = (x_2 \wedge \overline{x_1}) \vee (\overline{x_3} \wedge x_2 \wedge x_1)$$



Konjunktive Normalform (KNF)

- ▶ **Maxterm:** die ODER-Verknüpfung *aller* Schaltvariablen einer Schaltfunktion, die Variablen dürfen dabei negiert oder nicht negiert auftreten
- ▶ **Konjunktive Normalform:** die konjunktive Verknüpfung aller Maxterme μ mit dem Funktionswert 0

$$f = \bigwedge_{i=1}^{2^n} \hat{f}_i \cdot \mu(i), \quad \text{mit } \mu(i) : \text{Maxterm}(i)$$

auch: *kanonische konjunktive Normalform*
product-of-sums (POS)



Konjunktive Normalform: Maxterme

- ▶ Beispiel: alle 2^3 Maxterme für drei Variablen
- ▶ jeder Maxterm nimmt nur für eine Belegung der Eingangsvariablen den Wert 0 an

x_3	x_2	x_1	Maxterme
0	0	0	$x_3 \vee x_2 \vee x_1$
0	0	1	$x_3 \vee x_2 \vee \overline{x_1}$
0	1	0	$x_3 \vee \overline{x_2} \vee x_1$
0	1	1	$x_3 \vee \overline{x_2} \vee \overline{x_1}$
1	0	0	$\overline{x_3} \vee x_2 \vee x_1$
1	0	1	$\overline{x_3} \vee x_2 \vee \overline{x_1}$
1	1	0	$\overline{x_3} \vee \overline{x_2} \vee x_1$
1	1	1	$\overline{x_3} \vee \overline{x_2} \vee \overline{x_1}$

Konjunktive Normalform: Beispiel

x_3	x_2	x_1	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- ▶ Funktionstabelle: Maxterm $0 \equiv x_i$ $1 \equiv \bar{x}_i$
 - ▶ für f sind fünf Koeffizienten der KNF gleich 0
- ⇒ KNF:
$$f(x) = (x_3 \vee x_2 \vee x_1) \wedge (x_3 \vee x_2 \vee \bar{x}_1) \wedge (\bar{x}_3 \vee x_2 \vee x_1) \wedge (\bar{x}_3 \vee x_2 \vee \bar{x}_1) \wedge (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1)$$



Allgemeine konjunktive Form

- ▶ **konjunktive Form** (product-of-sums): die konjunktive Verknüpfung (UND) von Termen. Jeder Term besteht aus der ODER-Verknüpfung von Schaltvariablen, die entweder direkt oder negiert auftreten können
- ▶ entspricht dem Zusammenfassen („Minimierung“) von Termen aus der konjunktiven Normalform
- ▶ konjunktive Form ist nicht eindeutig (keine Normalform)

▶ Beispiel

$$\text{KNF } f(x) = (x_3 \vee x_2 \vee x_1) \wedge (x_3 \vee x_2 \vee \bar{x}_1) \wedge (\bar{x}_3 \vee x_2 \vee x_1) \wedge (\bar{x}_3 \vee x_2 \vee \bar{x}_1) \wedge (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1)$$

minimierte konjunktive Form

$$f(x) = (x_3 \vee x_2) \wedge (x_2 \vee x_1) \wedge (\bar{x}_3 \vee \bar{x}_1)$$



Reed-Muller Form

- ▶ **Reed-Muller Form:** die additive Verknüpfung aller Reed-Muller-Terme mit dem Funktionswert 1

$$f = \bigoplus_{i=1}^{2^n} \hat{f}_i \cdot RM(i)$$

- ▶ mit den Reed-Muller Basisfunktionen $RM(i)$
- ▶ Erinnerung: Addition im $GF(2)$ ist die XOR-Operation



Reed-Muller Form: Basisfunktionen

- ▶ Basisfunktionen sind:

$\{1\}$,	(0 Variablen)
$\{1, x_1\}$,	(1 Variable)
$\{1, x_1, x_2, x_2x_1\}$,	(2 Variablen)
$\{1, x_1, x_2, x_2x_1, x_3, x_3x_1, x_3x_2, x_3x_2x_1\}$,	(3 Variablen)
...	
$\{RM(n-1), x_n \cdot RM(n-1)\}$	(n Variablen)

- ▶ rekursive Bildung: bei n bit alle Basisfunktionen von $(n-1)$ -bit und zusätzlich das Produkt von x_n mit den Basisfunktionen von $(n-1)$ -bit



Reed-Muller Form: Umrechnung

Umrechnung von gegebenem Ausdruck in Reed-Muller Form?

- ▶ Ersetzen der Negation: $\bar{a} = a \oplus 1$
- Ersetzen der Disjunktion: $a \vee b = a \oplus b \oplus ab$
- Ausnutzen von: $a \oplus a = 0$

▶ Beispiel

$$\begin{aligned}
 f(x_1, x_2, x_3) &= (\bar{x}_1 \vee x_2)x_3 \\
 &= (\bar{x}_1 \oplus x_2 \oplus \bar{x}_1x_2)x_3 \\
 &= ((1 \oplus x_1) \oplus x_2 \oplus (1 \oplus x_1)x_2)x_3 \\
 &= (1 \oplus x_1 \oplus x_2 \oplus x_2 \oplus x_1x_2)x_3 \\
 &= x_3 \oplus x_1x_3 \oplus x_1x_2x_3
 \end{aligned}$$



Reed-Muller Form: Transformationsmatrix

- ▶ lineare Umrechnung zwischen Funktion f , bzw. der Funktionstabelle (disjunktive Normalform), und RMF
- ▶ Transformationsmatrix A kann rekursiv definiert werden (wie die RMF-Basisfunktionen)
- ▶ Multiplikation von A mit f ergibt Koeffizientenvektor r der RMF

$$r = A \cdot f, \quad \text{und} \quad f = A \cdot r$$

- ▶ weitere Details in [Hei05]
 K. von der Heide: *Vorlesung: Technische Informatik T1*
- ▶ Hinweis: Beziehung zu Fraktalen (Sierpinski-Dreieck)



Reed-Muller Form: Transformationsmatrix (cont.)

► $r = A \cdot f$ (und $A \cdot A = I$, also $f = A \cdot r$ (!))

$$A_0 = (1)$$

$$A_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Reed-Muller Form: Transformationsmatrix (cont.)

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

...

$$A_n = \begin{pmatrix} A_{n-1} & 0 \\ A_{n-1} & A_{n-1} \end{pmatrix}$$

Reed-Muller Form: Beispiel

x_3	x_2	x_1	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- ▶ Berechnung durch Rechenregeln der Boole'schen Algebra oder Aufstellen von A_3 und Ausmultiplizieren: $f(x) = x_2 \oplus x_3x_2x_1$
- ▶ häufig kompaktere Darstellung als DNF oder KNF

Reed-Muller Form: Beispiel (cont.)

- ▶ $f(x_3, x_2, x_1) = \{0, 0, 1, 1, 0, 0, 1, 0\}$ (Funktionstabelle)
- ▶ Aufstellen von A_3 und Ausmultiplizieren

$$r = A_3 \cdot f = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

führt zur gesuchten RMF:

$$f(x_3, x_2, x_1) = r \cdot RM(3) = x_2 \oplus x_3 x_2 x_1$$

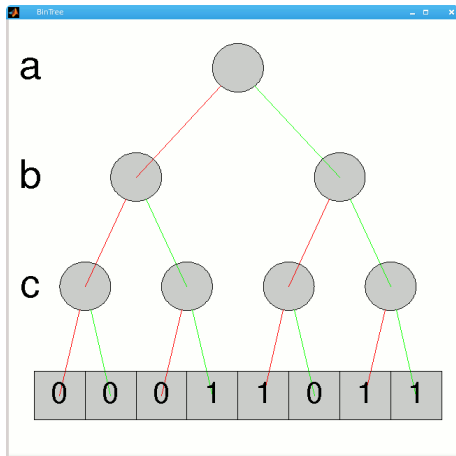


Grafische Darstellung: Entscheidungsbäume

- ▶ Darstellung einer Schaltfunktion als Baum/Graph
- ▶ jeder Knoten ist einer Variablen zugeordnet
jede Verzweigung entspricht einer *if-then-else*-Entscheidung
- ▶ vollständige Baum realisiert Funktionstabelle
- + einfaches Entfernen/Zusammenfassen redundanter Knoten
- ▶ Beispiel: Multiplexer

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

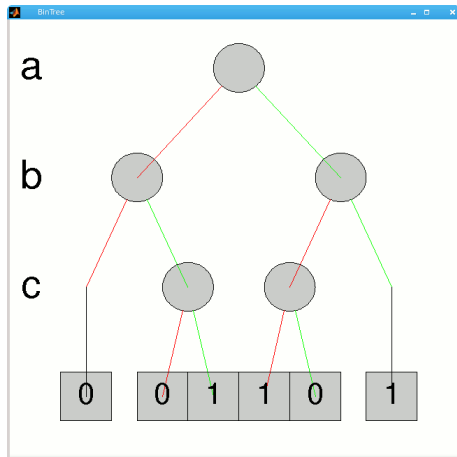
Entscheidungsbaum: Beispiel



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

- \blacktriangleright rot: 0-Zweig
- grün: 1-Zweig

Entscheidungsbaum: Beispiel (cont.)



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

\Rightarrow Knoten entfernt

\blacktriangleright rot: 0-Zweig
 grün: 1-Zweig

Reduced Ordered Binary-Decision Diagrams (ROBDD)

Binäres Entscheidungsdiagramm

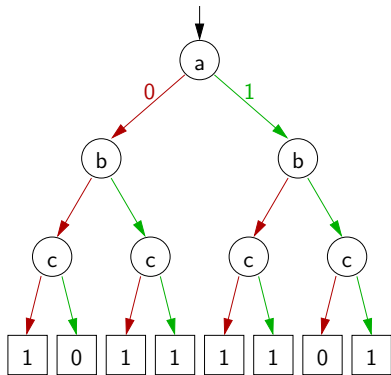
- ▶ Variante des Entscheidungsbaums
- ▶ vorab gewählte Variablenordnung (*ordered*)
- ▶ redundante Knoten werden entfernt (*reduced*)
- ▶ ein ROBDD ist eine Normalform für eine Funktion

- ▶ viele praxisrelevante Funktionen sehr kompakt darstellbar
 $O(n)..O(n^2)$ Knoten bei n Variablen
- ▶ wichtige Ausnahme: n -bit Multiplizierer ist $O(2^n)$
- ▶ derzeit das Standardverfahren zur Manipulation von
 (großen) Schaltfunktionen

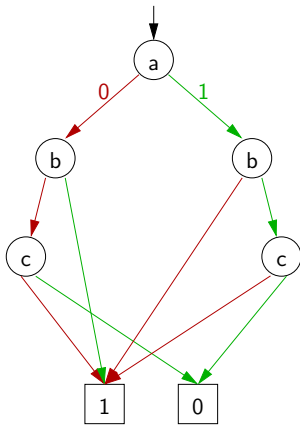
ROBDD vs. Entscheidungsbaum

Entscheidungsbaum

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

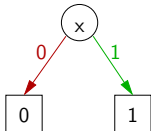


ROBDD

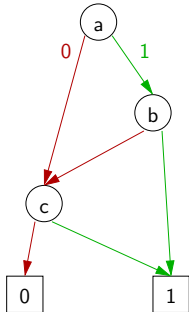


ROBDD: Beispiele

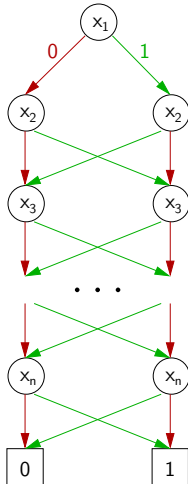
$$f(x) = x$$



$$g = (ab) \vee c$$



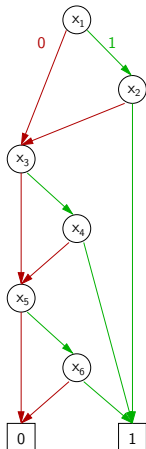
$$\text{Parität } p = x_1 \oplus x_2 \oplus \dots \oplus x_n$$



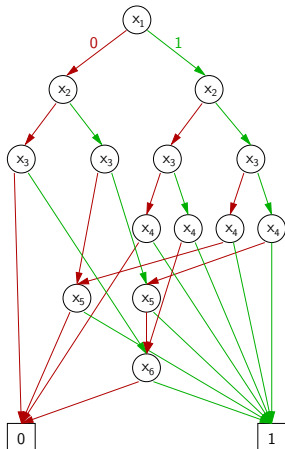
ROBDD: Problem der Variablenordnung

- Anzahl der Knoten oft stark abhängig von der Variablenordnung

$$f = x_1 x_2 \vee x_3 x_4 \vee x_5 x_6$$



$$g = x_1 x_4 \vee x_2 x_5 \vee x_3 x_6$$



Minimierung von Schaltfunktionen

- ▶ mehrere (beliebig viele) Varianten zur Realisierung einer gegebenen Schaltfunktion bzw. eines Schaltnetzes

Minimierung des Realisierungsaufwandes:

- ▶ diverse Kriterien, technologieabhängig
 - ▶ Hardwarekosten Anzahl der Gatter
 - ▶ Hardwareeffizienz z.B. NAND statt XOR
 - ▶ Geschwindigkeit Anzahl der Stufen, Laufzeiten
 - ▶ Testbarkeit Erkennung von Produktionsfehlern
 - ▶ Robustheit z.B. ionisierende Strahlung

Algebraische Minimierungsverfahren

- ▶ Vereinfachung der gegebenen Schaltfunktionen durch Anwendung der Gesetze der Boole'schen Algebra
- ▶ im Allgemeinen nur durch Ausprobieren
- ▶ ohne Rechner sehr mühsam
- ▶ keine allgemeingültigen Algorithmen bekannt
- ▶ Heuristische Verfahren
 - ▶ Suche nach *Primimplikanten* (= kürzeste Konjunktionsterme)
 - ▶ Quine-McCluskey-Verfahren und Erweiterungen



Algebraische Minimierung: Beispiel

- ▶ Ausgangsfunktion in DNF

$$\begin{aligned}
 y(x) = & \bar{x}_3 x_2 x_1 \bar{x}_0 \vee \bar{x}_3 x_2 x_1 x_0 \\
 & \vee x_3 \bar{x}_2 \bar{x}_1 x_0 \vee x_3 \bar{x}_2 x_1 \bar{x}_0 \\
 & \vee x_3 \bar{x}_2 x_1 x_0 \vee x_3 x_2 \bar{x}_1 x_0 \\
 & \vee x_3 x_2 x_1 \bar{x}_0 \vee x_3 x_2 x_1 x_0
 \end{aligned}$$

- ▶ Zusammenfassen benachbarter Terme liefert

$$y(x) = \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 x_0 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_0 \vee x_3 x_2 x_1$$

- ▶ aber bessere Lösung ist möglich (weiter Umformen)

$$y(x) = x_2 x_1 \vee x_3 x_0 \vee x_3 x_1$$



Grafische Minimierungsverfahren

- ▶ Darstellung einer Schaltfunktion im KV-Diagramm
- ▶ Interpretation als disjunktive Normalform (konjunktive NF)

- ▶ Zusammenfassen benachbarter Terme durch **Schleifen**
- ▶ alle 1-Terme mit möglichst wenigen Schleifen abdecken
 alle 0-Terme $\text{---} \equiv$ konjunktive Normalform
- ▶ Ablesen der minimierten Funktion, wenn keine weiteren Schleifen gebildet werden können

- ▶ beruht auf der menschlichen Fähigkeit, benachbarte Flächen auf einen Blick zu „sehen“
- ▶ bei mehr als 6 Variablen nicht mehr praktikabel

Erinnerung: Karnaugh-Veitch Diagramm

		x1 x0			
		00	01	11	10
x3 x2	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10

		x1 x0			
		00	01	11	10
x3 x2	00	0000	0001	0011	0010
	01	0100	0101	0111	0110
	11	1100	1101	1111	1110
	10	1000	1001	1011	1010

- ▶ 2D-Diagramm mit $2^n = 2^{n_y} \times 2^{n_x}$ Feldern
 - ▶ gängige Größen sind: 2×2 , 2×4 , 4×4
darüber hinaus: mehrere Diagramme der Größe 4×4
 - ▶ Anordnung der Indizes ist im Gray-Code (!)
- ⇒ benachbarte Felder unterscheiden sich gerade um 1 Bit

KV-Diagramme: 2...4 Variable (2×2, 2×4, 4×4)

		x0	
		0	1
x1	0	00	01
	1	10	11

		x1 x0			
		00	01	11	10
x3 x2	00	0000	0001	0011	0010
	01	0100	0101	0111	0110
	11	1100	1101	1111	1110
	10	1000	1001	1011	1010

		x1 x0			
		00	01	11	10
x2	0	000	001	011	010
	1	100	101	111	110



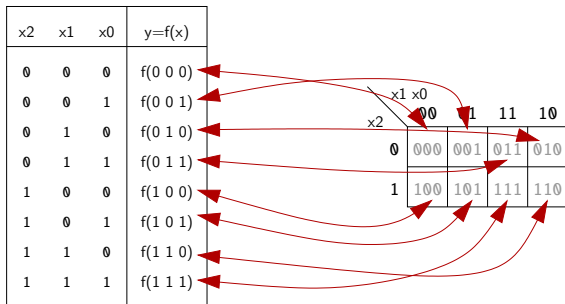
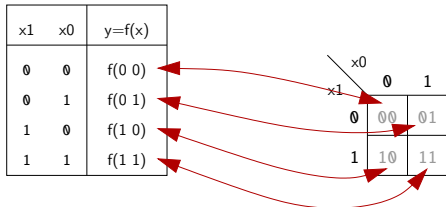
KV-Diagramm für Schaltfunktionen

- ▶ Funktionswerte in zugehöriges Feld im KV-Diagramm eintragen
- ▶ Werte 0 und 1
Don't-Care „*“ für nicht spezifizierte Werte wichtig!
- ▶ 2D-Äquivalent zur Funktionstabelle

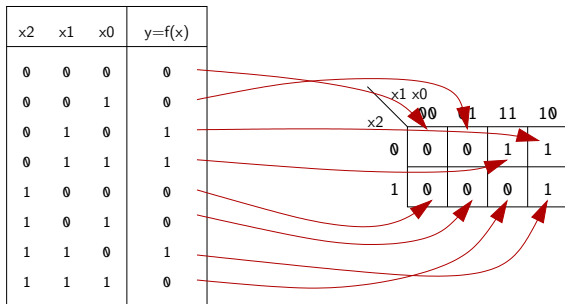
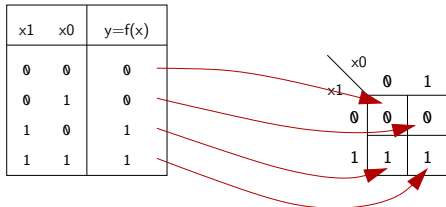
- ▶ praktikabel für 3..6 Eingänge
- ▶ fünf Eingänge: zwei Diagramme a 4×4 Felder
 sechs Eingänge: vier Diagramme a 4×4 Felder

- ▶ viele Strukturen „auf einen Blick“ erkennbar

KV-Diagramm: Zuordnung zur Funktionstabelle



KV-Diagramm: Eintragen aus Funktionstabelle



KV-Diagramm: Beispiel

		x1 x0			
		00	01	11	10
x3 x2	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10

		x1 x0			
		00	01	11	10
x3 x2	00	1	0	0	1
	01	0	0	0	0
	11	0	0	1	0
	10	0	0	1	0

- ▶ Beispielfunktion in DNF mit vier Termen:

$$f(x) = (\overline{x_3} \overline{x_2} \overline{x_1} \overline{x_0}) \vee (\overline{x_3} \overline{x_2} x_1 \overline{x_0}) \vee (x_3 \overline{x_2} x_1 x_0) \vee (x_3 x_2 x_1 x_0)$$
- ▶ Werte aus Funktionstabelle an entsprechender Stelle ins Diagramm eintragen

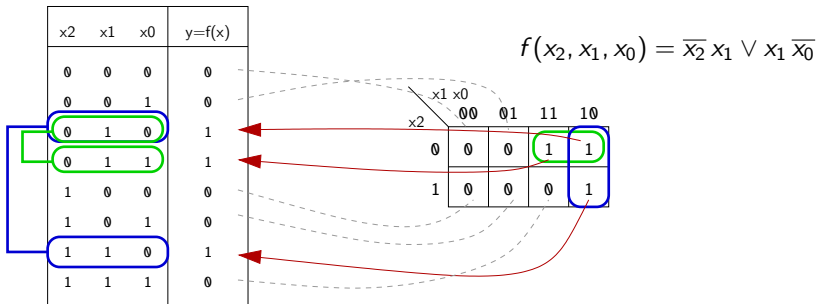
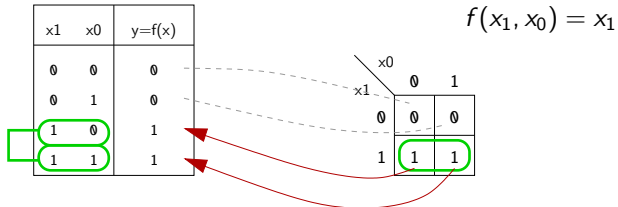
Schleifen: Zusammenfassen benachbarter Terme

- ▶ benachbarte Felder unterscheiden sich um 1-Bit
- ▶ falls benachbarte Terme beide 1 sind \Rightarrow Funktion hängt an dieser Stelle nicht von der betroffenen Variable ab
- ▶ zugehörige (Min-) Terme können zusammengefasst werden

- ▶ Erweiterung auf vier benachbarte Felder (4x1 1x4 2x2)
 —"–" auf acht —"–" (4x2 2x4) usw.
- ▶ aber keine Dreier- Fünfergruppen, usw. (Gruppengröße 2^i)

- ▶ Nachbarschaft auch „außen herum“
- ▶ mehrere Schleifen dürfen sich überlappen

Schleifen: Ablesen der Schleifen



Schleifen: Ablesen der Schleifen (cont.)

		x1 x0			
		00	01	11	10
x3 x2	00	1	0	0	1
	01	0	0	0	0
	11	0	0	1	0
	10	0	0	1	0

		x1 x0			
		00	01	11	10
x3 x2	00	1	0	0	1
	01	0	0	0	0
	11	0	0	1	0
	10	0	0	1	0

- ▶ insgesamt zwei Schleifen möglich
- ▶ grün entspricht $(\overline{x_3} \overline{x_2} \overline{x_0}) = (\overline{x_3} \overline{x_2} \overline{x_1} \overline{x_0}) \vee (\overline{x_3} \overline{x_2} x_1 \overline{x_0})$
 blau entspricht $(x_3 x_1 x_0) = (x_3 x_2 x_1 x_0) \vee (x_3 \overline{x_2} x_1 x_0)$
- ▶ minimierte disjunktive Form $f(x) = (\overline{x_3} \overline{x_2} \overline{x_0}) \vee (x_3 x_1 x_0)$

Schleifen: interaktive Demonstration

▶ Applet zur Minimierung mit KV-Diagrammen [HenKV]
tams.informatik.uni-hamburg.de/applets/kvd

1. Auswahl oder Eingabe einer Funktion (2..6 Variablen)
2. Interaktives Setzen und Erweitern von Schleifen:
 „click“, „shift+click“, „control+click“
3. Anzeige der zugehörigen Hardwarekosten und Schaltung

Achtung: andere Anordnung der Eingangsvariablen als im Skript

⇒ entsprechend andere Anordnung der Terme im KV-Diagramm
 Prinzip bleibt aber gleich

KV-Diagramm Applet: Screenshots

KV-Diagramm-Applet

File Option Info

Java Applet Window

Edit Function
 Show Indices
 Add Loop 1 (DNF)
 Add Loop 0 (KNF)

Function y1

		i0				
i3	1	0	1	0	i1	
	1	1	1	0		
	1	1	1	0		
	1	0	0	0		
		i2				

press <CTRL> + mouseclick to DELETE this loop
 press <build Loop> to commit the Loop
 press <build Loop> to commit the Loop

KV-Diagramm-Applet

File Option Info

Java Applet Window

Edit Function
 Show Indices
 Add Loop 1 (DNF)
 Add Loop 0 (KNF)

Function y1

		i0				
i3	1	0	1	0	i1	
	1	1	1	0		
	1	1	1	0		
	1	0	0	0		
		i2				

press <CTRL> + mouseclick to DELETE this loop
 press <build Loop> to commit the Loop
 press <build Loop> to commit the Loop

KV-Diagramm-Applet

File Option Info

Java Applet Window

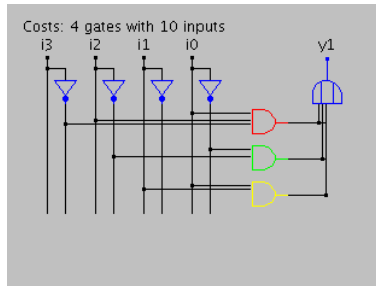
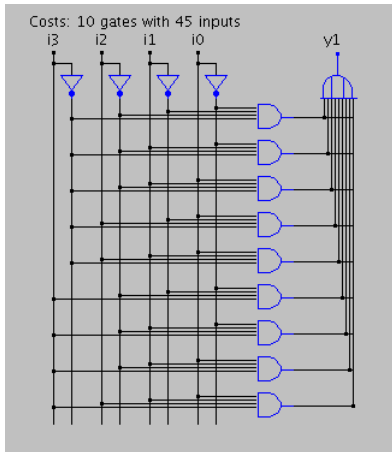
Edit Function
 Show Indices
 Add Loop 1 (DNF)
 Add Loop 0 (KNF)

Function y1

		i0				
i3	1	0	1	0	i1	
	1	1	1	0		
	1	1	1	0		
	1	0	0	0		
		i2				

press <CTRL> + mouseclick to DELETE this loop
 press <build Loop> to commit the Loop
 press <build Loop> to commit the Loop

KV-Diagramm Applet: zugehörige Hardwarekosten

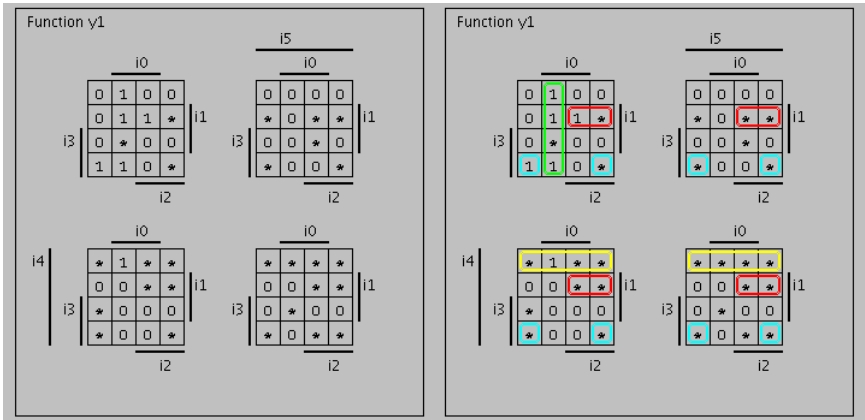


Don't-Care Terme

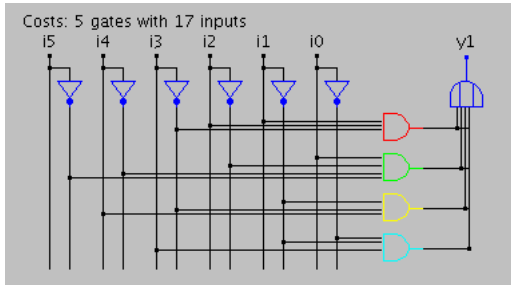
- ▶ in der Praxis: viele Schaltfunktionen unvollständig definiert weil bestimmte Eingangskombinationen nicht vorkommen
- ▶ zugehörige Terme als **Don't-Care** markieren
 typisch: Sternchen „*“ in Funktionstabelle/KV-Diagramm

- ▶ solche Terme bei Minimierung nach Wunsch auf 0/1 setzen
- ▶ Schleifen dürfen *Don't-Cares* enthalten
- ▶ Schleifen möglichst groß

KV-Diagramm Applet: 6 Variablen, *Don't-Cares*



KV-Diagramm Applet: 6 Variablen, *Don't-Cares* (cont.)



- ▶ Schaltung und Realisierungsaufwand (# Gatter, Eingänge) nach der Minimierung

Quine-McCluskey-Algorithmus

- ▶ Algorithmus zur Minimierung einer Schaltfunktion
- ▶ Notation der Terme in Tabellen, n Variablen
- ▶ Prinzip entspricht der Minimierung im KV-Diagramm aber auch geeignet für mehr als sechs Variablen
- ▶ Grundlage gängiger Minimierungsprogramme

- ▶ Sortieren der Terme nach Hamming-Abstand
- ▶ Erkennen der unverzichtbaren Terme („Primimplikanten“)
- ▶ Aufstellen von Gruppen benachbarter Terme (mit Distanz 1)
- ▶ Zusammenfassen geeigneter benachbarter Terme

Becker, Molitor: *Technische Informatik – eine einführende Darstellung* [BM08]

Schiffmann, Schmitz: *Technische Informatik I* [SS04]



Literatur

- [BM08] B. Becker, P. Molitor: *Technische Informatik – eine einführende Darstellung*. 2. Auflage, Oldenbourg, 2008. ISBN 978-3-486-58650-3
- [SS04] W. Schiffmann, R. Schmitz: *Technische Informatik 1 – Grundlagen der digitalen Elektronik*. 5. Auflage, Springer-Verlag, 2004. ISBN 978-3-540-40418-7
- [WH03] H.D. Wuttke, K. Henke: *Schaltsysteme – Eine automatenorientierte Einführung*. Pearson Studium, 2003. ISBN 978-3-8273-7035-8



Literatur (cont.)

- [Bry86] R.E. Bryant: *Graph-Based Algorithms for Boolean Function Manipulation*. in: *IEEE Trans. Computers* 35 (1986), Nr. 8, S. 677–691
- [HenKV] N. Hendrich: *KV-Diagram Simulation*.
 Universität Hamburg, FB Informatik, Lehrmaterial.
tams.informatik.uni-hamburg.de/applets/kvd
- [Hei05] K. von der Heide: *Vorlesung: Technische Informatik 1 — interaktives Skript*. Universität Hamburg, FB Informatik, 2005
tams.informatik.uni-hamburg.de/lectures/2004ws/vorlesung/t1