Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN-Fakultät
Fachbereich Informatik

Rechnerstrukturen und Betriebssysteme

64-040 Vorlesung RSB
Kapitel 8: Schaltfunktionen

Norman Hendrich

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Technische Aspekte Multimodaler Systeme

Wintersemester 2025/2026

Gliederung

Schaltfunktionen

- **Schaltvariable**: eine Variable, die nur endlich viele Werte annehmen kann
  hier sind das binäre Schaltvariablen: b_i
- **Schaltfunktion**: eine eindeutige Zuordnungsvorschrift f, die jeder Wertekombination $(b_1, b_2, \ldots, b_n)$ von Schaltvariablen einen Wert zuweist:

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

- **Ausgangsvariable**: die Schaltvariable am Ausgang der Funktion, für den Wert y

- 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“

Funktionstabelle

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

$\Rightarrow$ 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

```
f = x2 && !(x3 && x2 && x1)
```

- Einsatz spezieller (Hardware-) Beschreibungssprachen: Verilog, VHDL, SystemC

```
f <= (x2 and (x3 nand (x2 and x1))
```

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?

$\Rightarrow$ je nach Interpretation völlig unterschiedliche Schaltung

Strukturbeschreibung

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

- vollständig geklammerte algebraische Ausdrücke
  $f = x_1 \oplus (x_2 \vee x_3)$
- Datenflussgraphen
- Blockschaltbilder
- Schaltpläne mit Gattern (nächstes Kapitel)
- PLA-Format für zweistufige AND-OR Schaltungen
- ...

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 }

- 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 (mehrere) gegebene Funktionen übereinstimmen

- Beispiel: Darstellung reeller Polynome:

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

a_i: Koeffizienten
x^i: Basisfunktionen

Normalformen (cont.)

Normalform einer Boole'schen Funktion

- analog zur Potenzdarstellung von Polynomen
- als Summe über Koeffizienten {0,1} und Basisfunktionen

$$f = \sum_{i=1}^{2^n} a_i B_i, \quad a_i \in \{0, 1\}$$

mit $B_1, \dots, B_{2^n}$ 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

$$f = (a_1 \otimes B_1) \oplus \ldots \oplus (a_{2^n} \otimes B_{2^n})$$

schreiben lassen, so wird die Form als **Normalform** und die Menge der B_i als **Basis** bezeichnet

- Menge von 2^n Basisfunktionen 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

$$f = \bigvee_{i=1}^{2^n} a_i \cdot m_i, \quad \text{mit} \;\; a_i \in \{0,1\}, m_i : \;\; \text{Minterm}_i$$

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

- Minterm aus Funktionstabelle ablesen: $0 \equiv \overline{x_i} \quad 1 \equiv x_i$
- für f sind nur drei Koeffizienten der DNF gleich 1

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

Allgemeine disjunktive Form

- **Disjunktive Form** (sum-of-products)
  - die disjunktive Verknüpfung von Termen (ODER)
  - jeder Term besteht aus der UND-Verknüpfung von Schaltvariablen, direkt oder negiert
  - in Termen müssen nicht alle Schaltvariablen vorkommen (anders als bei Mintermen)
- entspricht dem Zusammenfassen von Termen aus der disjunktiven Normalform schaltalgebraische Minimierung

- Beispiel

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

  minimierte disjunktive Form

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

- disjunktive Form ist nicht eindeutig $\Rightarrow$ ist keine Normalform

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

$$f = \bigwedge_{i=1}^{2^n} a_i \cdot \mu_i, \quad \text{mit} \quad a_i \in \{0,1\}, \mu_i: \ \text{Maxterm}_i$$

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

- Maxterm aus Funktionstabelle ablesen: $0 \equiv x_i \quad 1 \equiv \overline{x_i}$
- KNF aus Funktionstabelle ablesen: $a_i = \neg f_i$ (alle $f = 0$ Terme sammeln)

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

Allgemeine konjunktive Form

- **Konjunktive Form** (product-of-sums)
  - die konjunktive Verknüpfung von Termen (UND)
  - jeder Term besteht aus der ODER-Verknüpfung von Schaltvariablen, direkt oder negiert
  - in Termen müssen nicht alle Schaltvariablen vorkommen (anders als bei Maxtermen)
- entspricht dem Zusammenfassen von Termen aus der konjunktiven Normalform schaltalgebraische Minimierung
- konjunktive Form ist nicht eindeutig $\Rightarrow$ ist keine Normalform

- Beispiel

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

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

multiplikative Kurzschreibweise

Anmerkung

- nach der Boole'schen Algebra sind Konjunktion $\wedge$ und Disjunktion $\vee$ gleichwertig!

$\Rightarrow$ es gibt also keine vorrangige Operation

- häufig wird folgende verkürzte multiplikative Schreibweise in schaltalgebraischen Ausdrücken benutzt
- ab statt $a \wedge b$

  $a\,b \vee c\,d$ statt $(a \wedge b) \vee (c \wedge d)$

$\Rightarrow$ der Operator $\wedge$ wird weggelassen oder durch $\cdot$ ersetzt

$\Rightarrow$ die Ausdrücke sind (implizit) geklammert $\Rightarrow$ $\wedge$ bindet stärker als $\vee$ (Konvention)

Reed-Muller Form

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

$$f = \bigoplus_{i=1}^{2^n} r_i \cdot RM_i, \quad r_i \in \{0,1\}$$

- mit den Reed-Muller Basisfunktionen $RM(i)$

- Erinnerung: Addition im GF(2) ist die XOR-Operation

Reed-Muller Form: Basisfunktionen

- Basisfunktionen sind:

$$
\begin{array}{lr}
\{1\}, & (0 \text{ Variablen}) \\
\{1,\ x_1\}, & (1 \text{ Variable }) \\
\{1,\ x_1,\ x_2,\ x_2x_1\}, & (2 \text{ Variablen}) \\
\{1,\ x_1,\ x_2,\ x_2x_1,\ x_3,\ x_3x_1,\ x_3x_2,\ x_3x_2x_1\}, & (3 \text{ Variablen}) \\
\ldots & \\
\{RM(n-1),\quad x_n \cdot RM(n-1)\} & (n \text{ Variablen})
\end{array}
$$

- 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: $\overline{a} = a \oplus 1$

  Ersetzen der Disjunktion: $a \vee b = a \oplus b \oplus ab$

  Ausnutzen von: $a \oplus a = 0$

- Beispiel

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

Reed-Muller Form: Transformationsmatrix

- lineare Umrechnung zwischen Funktion f, bzw. deren Funktionstabelle, und RMF
- Transformationsmatrix A kann rekursiv definiert werden (ähnlich den RMF-Basisfunktionen)
- Multiplikation von A mit f ergibt Koeffizientenvektor r der RMF

$$r = A \times f \quad \text{und} \quad f = A \times r$$

gilt wegen: $r = A \times f$ und $A \times A = I$, also $f = A \times r$!

$$A_0 = (1)$$

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

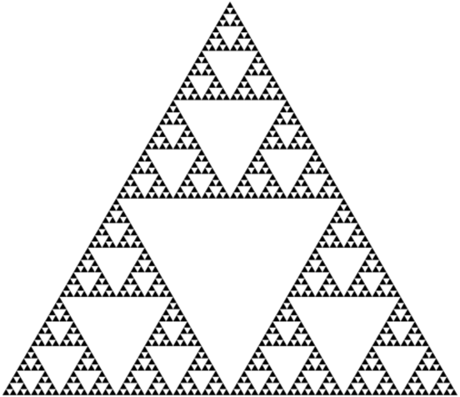
Reed-Muller Form: Transformationsmatrix

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

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

...

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



vgl. Sierpinski-Dreieck (Fraktal)

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: $f(x) = x_2 \oplus x_3 x_2 x_1$
- 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

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

Basisfunktionen: $\{1, x_1, x_2, x_2x_1, x_3, x_3x_1, x_3x_2, x_3x_2x_1\}$

führt zur gesuchten RMF: $f(x_3,x_2,x_1) = r \times RM(3) = x_2 \oplus x_3x_2x_1$

Grafische Darstellung: Entscheidungsbäume

- Darstellung einer Schaltfunktion als Baum/Graph
- jeder Knoten ist einer Variablen zugeordnet

  jede Verzweigung entspricht einer `if-then-else`-Entscheidung

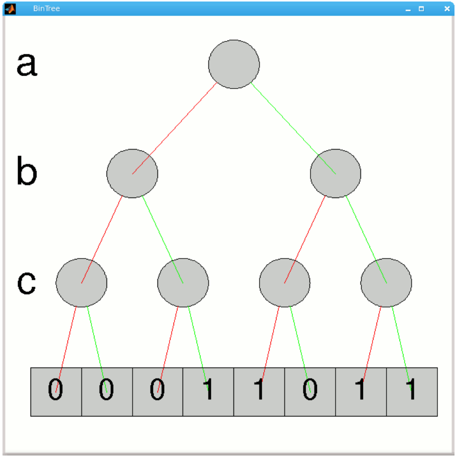
- vollständiger Baum realisiert Funktionstabelle

\+ einfaches Entfernen/Zusammenfassen redundanter Knoten

- Beispiel: Multiplexer

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

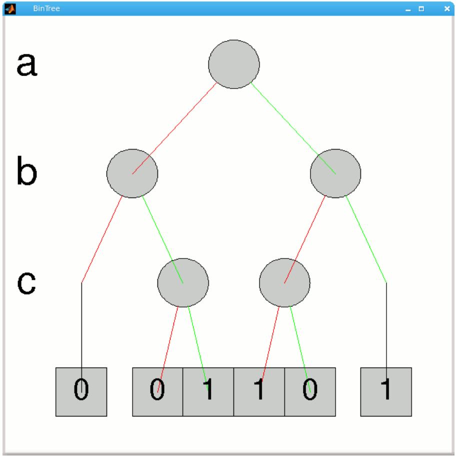
Entscheidungsbaum: Beispiel



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

- rot: 0-Zweig
  grün: 1-Zweig

Entscheidungsbaum: Beispiel (cont.)



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

$\Rightarrow$ redundante Knoten entfernt

- 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

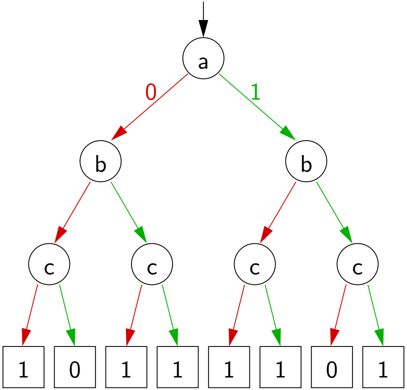
- Vorteil: viele praxisrelevante Funktionen sehr kompakt darstellbar
  $\mathcal{O}(n) \ldots \mathcal{O}(n^2)$ Knoten bei n Variablen statt $\mathcal{O}(2^n)$
- wichtige Ausnahme: n-bit Multiplizierer ist $\mathcal{O}(2^n)$
- derzeit das Standardverfahren zur Manipulation von (großen) Schaltfunktionen

R. E. Bryant: *Graph-Based Algorithms for Boolean Function Manipulation*, [Bry86]

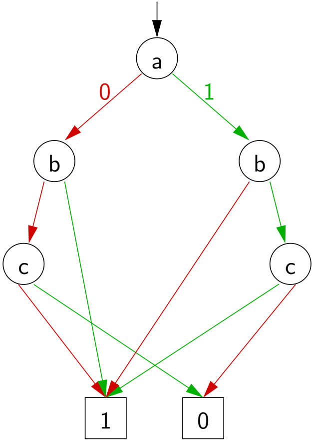
ROBDD vs. Entscheidungsbaum

Entscheidungsbaum

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

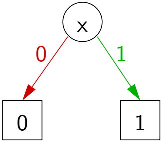


ROBDD

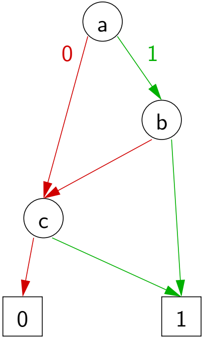


ROBDD: Beispiele

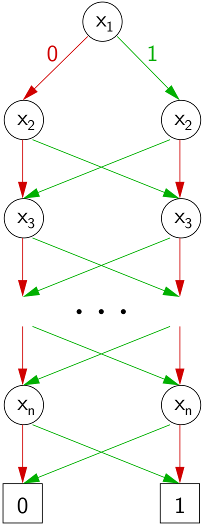
$f(x) = x$



$g = (a\,b) \vee c$



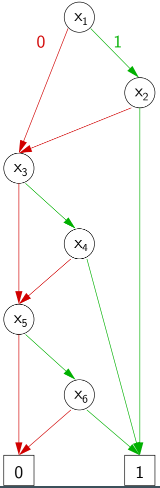
Parität $p = x_1 \oplus x_2 \oplus \ldots x_n$



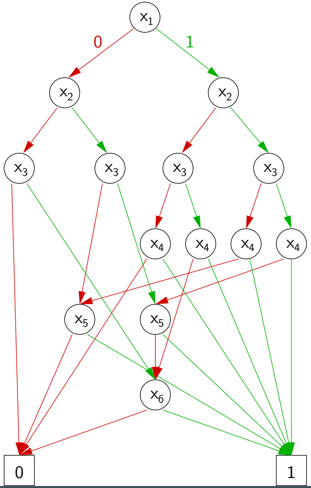
ROBDD: Problem der Variablenordnung

- 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 Elemente (Gatter / Chipfläche)
▶ Hardwareeffizienz	z.B. NAND statt XOR
▶ Geschwindigkeit	Anzahl der Stufen, Laufzeiten
▶ Leistungsaufnahme	statischer / dynamischer Stromverbrauch
▶ Testbarkeit	Erkennung von Produktionsfehlern
▶ Robustheit	z.B. Temperatur-, Spannungsschwankungen, 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

$$\begin{aligned} y(x) = \; & \overline{x_3}\, x_2\, x_1\, \overline{x_0} \vee \overline{x_3}\, x_2\, x_1\, x_0 \\ \vee \; & x_3\, \overline{x_2}\, \overline{x_1}\, x_0 \vee x_3\, \overline{x_2}\, x_1\, \overline{x_0} \\ \vee \; & x_3\, \overline{x_2}\, x_1\, x_0 \vee x_3\, x_2\, \overline{x_1}\, x_0 \\ \vee \; & x_3\, x_2\, x_1\, \overline{x_0} \vee x_3\, x_2\, x_1\, x_0 \end{aligned}$$

- Zusammenfassen von Termen liefert

$$y(x) = \overline{x_3}\, x_2\, x_1 \vee x_3\, \overline{x_2}\, x_0 \vee x_3\, \overline{x_2}\, x_1 \vee x_3\, x_2\, x_0 \vee x_3\, x_2\, x_1$$

Algebraische Minimierung: Beispiel

- Ausgangsfunktion in DNF

$$\begin{aligned} y(x) = {} & \overline{x_3}\, x_2\, x_1\, \overline{x_0} \vee \overline{x_3}\, x_2\, x_1\, x_0 \\ & \vee\ x_3\, \overline{x_2}\, \overline{x_1}\, x_0 \vee x_3\, \overline{x_2}\, x_1\, \overline{x_0} \\ & \vee\ x_3\, \overline{x_2}\, x_1\, x_0 \vee x_3\, x_2\, \overline{x_1}\, x_0 \\ & \vee\ x_3\, x_2\, x_1\, \overline{x_0} \vee x_3\, x_2\, x_1\, x_0 \end{aligned}$$

- Zusammenfassen von Termen liefert

$$y(x) = \overline{x_3}\, x_2\, x_1 \vee x_3\, \overline{x_2}\, x_0 \vee x_3\, \overline{x_2}\, x_1 \vee x_3\, x_2\, x_0 \vee x_3\, x_2\, x_1$$

- 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 / als konjunktive Normalform

- Zusammenfassen benachbarter Terme durch **Schleifen**
- alle 1-Terme mit möglichst wenigen Schleifen abdecken
  (alle 0-Terme –"– $\equiv$ konjunktive Normalform)
- minimierte Funktion ablesen, wenn keine weiteren Schleifen gebildet werden können

- nutzt Fähigkeit des Menschen, benachbarte Flächen auf einen Blick zu „sehen“
- bei mehr als 6 Variablen nicht mehr praktikabel

Erinnerung: Karnaugh-Veitch Diagramm

$x_3 x_2$ \ $x_1 x_0$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$x_3 x_2$ \ $x_1 x_0$	00	01	11	10
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 einschrittigen-Code / Gray-Code

$\Rightarrow$ benachbarte Felder unterscheiden sich gerade um 1 Bit

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

x_1 \ x_0	0	1
0	00	01
1	10	11

x_2 \ $x_1\,x_0$	00	01	11	10
0	000	001	011	010
1	100	101	111	110

$x_3\,x_2$ \ $x_1\,x_0$	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

KV-Diagramm für Schaltfunktionen

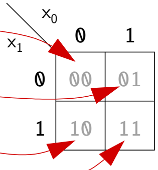
- Funktionswerte 0 und 1 in zugehöriges Feld im KV-Diagramm eintragen
- zusätzlich: *don't-care* „$*$“ für nicht spezifizierte Werte

- 2D-Äquivalent zur Funktionstabelle
- praktikabel für $3 \ldots 6$ Eingänge
- fünf Eingänge: zwei Diagramme à 4×4 Felder
  sechs Eingänge: vier Diagramme à 4×4 Felder

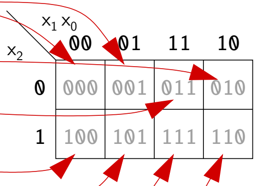
$\Rightarrow$ viele Strukturen „auf einen Blick“ erkennbar

KV-Diagramm: Zuordnung zur Funktionstabelle

x_1	x_0	y=f(x)
0	0	f(0 0)
0	1	f(0 1)
1	0	f(1 0)
1	1	f(1 1)



x_2	x_1	x_0	y=f(x)
0	0	0	f(0 0 0)
0	0	1	f(0 0 1)
0	1	0	f(0 1 0)
0	1	1	f(0 1 1)
1	0	0	f(1 0 0)
1	0	1	f(1 0 1)
1	1	0	f(1 1 0)
1	1	1	f(1 1 1)



KV-Diagramm: Eintragen aus Funktionstabelle

x_1	x_0	y=f(x)
0	0	0
0	1	0
1	0	1
1	1	1

x_1 \ x_0	0	1
0	0	0
1	1	1

x_2	x_1	x_0	y=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

x_2 \ x_1 x_0	00	01	11	10
0	0	0	1	1
1	0	0	0	1

KV-Diagramm: Beispiel

$x_3\,x_2$ \ $x_1\,x_0$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$x_3\,x_2$ \ $x_1\,x_0$	00	01	11	10
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
- die zugehörigen (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 im KV-Diagramm

x_1	x_0	y=f(x)
0	0	0
0	1	0
1	0	1
1	1	1

$$f(x_1, x_0) = x_1$$

x_1 \ x_0	0	1
0	0	0
1	1	1

x_2	x_1	x_0	y=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

$$f(x_2, x_1, x_0) = \overline{x_2}\, x_1 \vee x_1\, \overline{x_0}$$

x_2 \ $x_1 x_0$	00	01	11	10
0	0	0	1	1
1	0	0	0	1

Schleifen: Ablesen im KV-Diagramm (cont.)

$x_3\,x_2$ \ $x_1\,x_0$	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	1	0
10	0	0	1	0

$x_3\,x_2$ \ $x_1\,x_0$	00	01	11	10
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 $(\overline{x_3}\,\overline{x_2}\,\overline{x_0}) = (\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}) \vee (\overline{x_3}\,\overline{x_2}\,x_1\,\overline{x_0})$

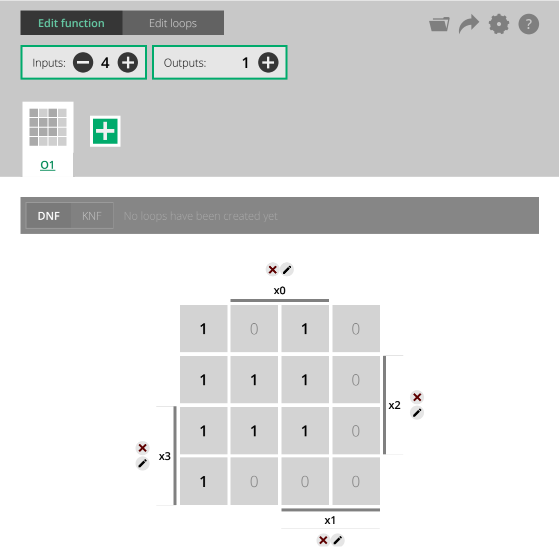
  blau entspricht $(x_3\,x_1\,x_0) = (x_3\,x_2\,x_1\,x_0) \vee (x_3\,\overline{x_2}\,x_1\,x_0)$
- minimierte disjunktive Form $f(x) = (\overline{x_3}\,\overline{x_2}\,\overline{x_0}) \vee (x_3\,x_1\,x_0)$

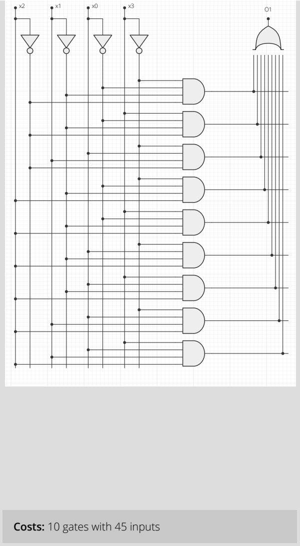
Schleifen: interaktive Demonstration

- Minimierung mit KV-Diagrammen [Kor16]
  `tams.informatik.uni-hamburg.de/research/software/tams-tools/kvd-editor.html`
  - Auswahl der Funktionalität: *Edit function*, *Edit loops*
  - Explizite Eingabe: *Open Diagram - From Expressions* Tipp!
  1. Funktion: Maustaste ändert Werte
  2. Schleifen: Auswahl und Aufziehen mit Maustaste
  - Anzeige des zugehörigen Hardwareaufwands und der Schaltung

- Applet zur Minimierung mit KV-Diagrammen [Hen96]
  `tams.informatik.uni-hamburg.de/applets/kvd`
  - Auswahl der Funktionalität: *Edit function*, *Add loop* ...
  - Ändern der Ein-/Ausgänge: *File - Examples - User define dialog*
  1. Funktion: Maustaste ändert Werte
  2. Schleifen: Maustaste, *shift*+Maus, *ctrl*+Maus
  - Anzeige des zugehörigen Hardwareaufwands und der Schaltung
  - Achtung: andere Anordnung der Eingangsvariablen als im Skript
    $\Rightarrow$ andere Anordnung der Terme im KV-Diagramm

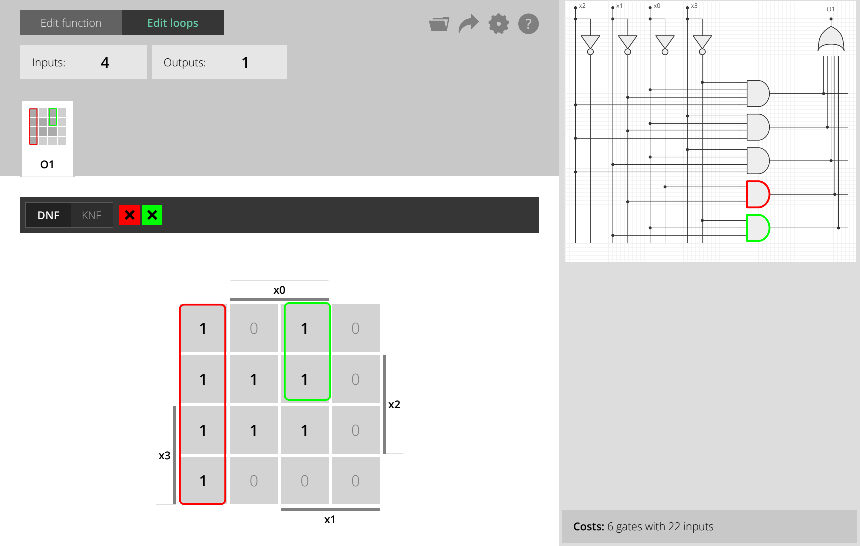
KV-Diagramm Editor: Screenshots





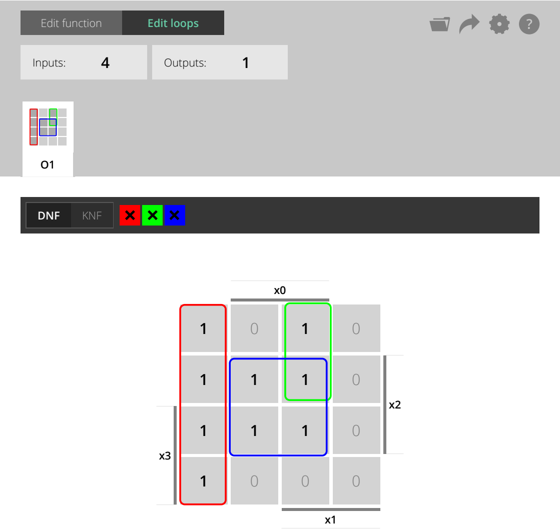
Eingabe der Schaltfunktion

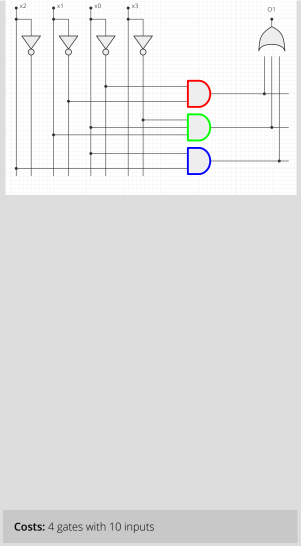
KV-Diagramm Editor: Screenshots (cont.)



Minimierung durch Schleifenbildung

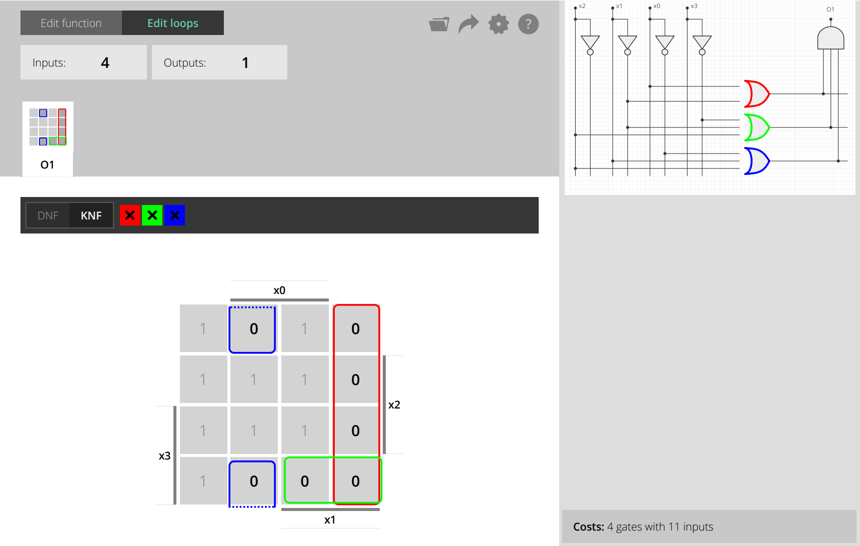
KV-Diagramm Editor: Screenshots (cont.)





Hardware-Kosten: # Gatter, Eingänge

KV-Diagramm Editor: Screenshots (cont.)



Konjunktive Form

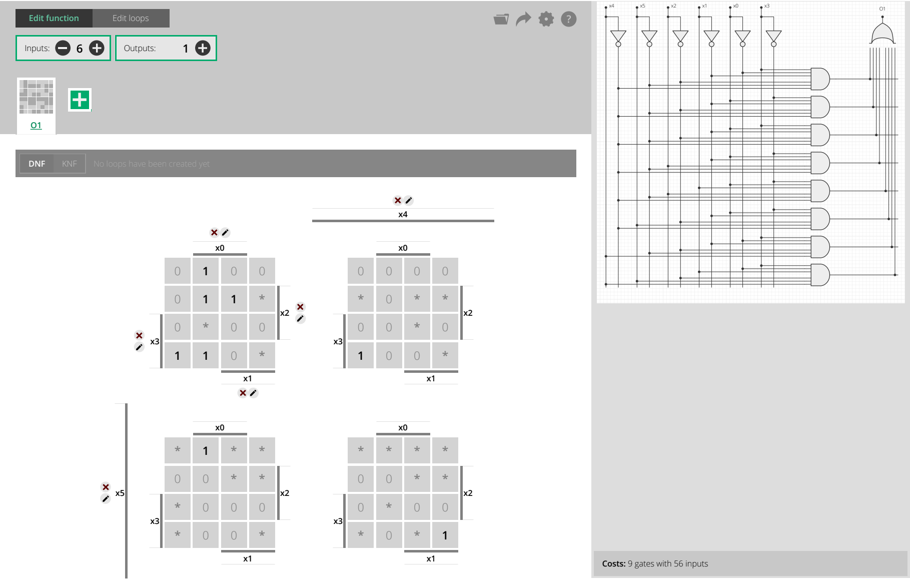
Don't-Care Terme

- in der Praxis sind viele Schaltfunktionen unvollständig definiert, weil bestimmte Eingangskombinationen nicht vorkommen
- zugehörige Terme werden in Funktionstabelle, bzw. im KV-Diagramm als ***Don't-Care*** markiert: „$*$“

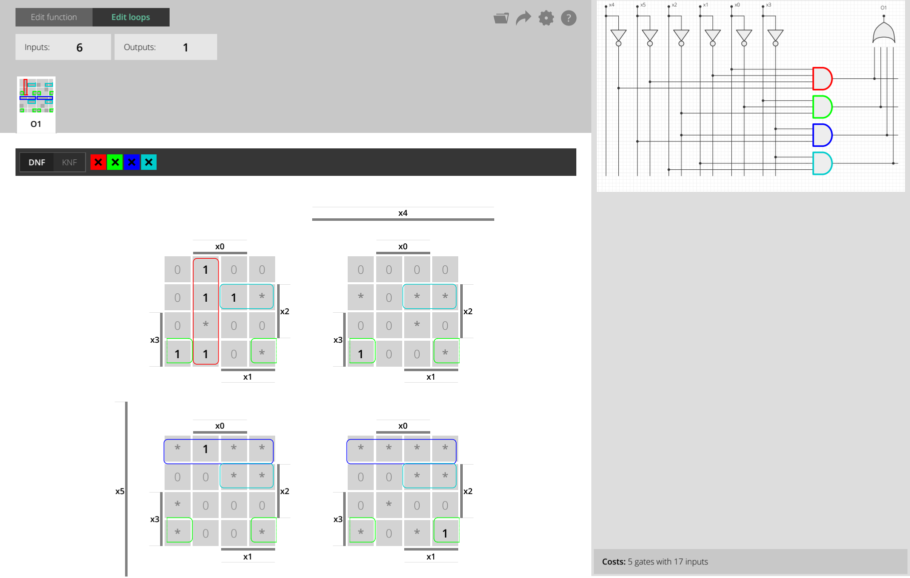
$\Rightarrow$ bei der Minimierung können *Don't-Care* Terme nach Wunsch als 0 oder 1 angenommen werden

$\Rightarrow$ *Don't-Cares* ausnutzen, um Schleifen möglichst groß zu machen

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



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



Literatur

[BM08] Bernd Becker and Paul Molitor.
Technische Informatik – eine einführende Darstellung.
Oldenbourg, München, second edition, 2008.

[Bry86] Randal E. Bryant.
Graph-based algorithms for boolean function manipulation.
IEEE Trans. Computers, 35(8):677–691, 1986.

[Hen96] Norman Hendrich.
Kv-diagram simulation.
Lehrmaterial, Universität Hamburg, Fachbereich Informatik, AB TAMS, 1996.

Literatur (cont.)

[Kor16] Laszlo Korte.
BSc Thesis: TAMS Tools for eLearning.
Bachelorarbeit, Universität Hamburg, Fachbereich Informatik, AB TAMS, 2016.

[SS04] Wolfram Schiffmann and Robert Schmitz.
Technische Informatik 1 – Grundlagen der digitalen Elektronik.
Springer-Verlag GmbH, Berlin, fifth edition, 2004.

[vdH05] Klaus von der Heide.
Vorlesung: Technische informatik 1 — interaktives skript.
Vorlesungsskript, Universität Hamburg, Fachbereich Informatik, AB TECH, 2005.

Literatur (cont.)

[WH03] Heinz-Dietrich Wuttke and Karsten Henke.
Schaltsysteme – Eine automatenorientierte Einführung.
Pearson Studium, München, Boston. . . , 2003.