GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Informatik II

SS 2005

Kapitel 2: Zahlen und Logik

Teil 2: Logik

Dr. Michael Ebner
Dipl.-Inf. René Soltwisch

Lehrstuhl für Telematik
Institut für Informatik

Überblick

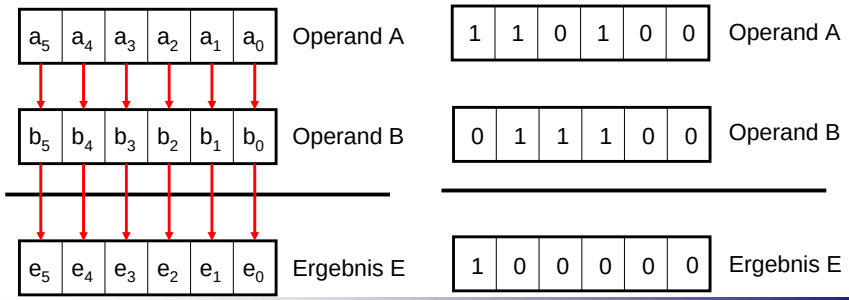
- Zahlen
  - Informationsdarstellung
  - Zahlensysteme
  - Rechnerarithmetik
- Logik
  - **Aussagenlogik und logische Gatter**
  - Prädikatenlogik

Ziel: Rechnerarithmetik implementieren

- Bitweise logische Operationen

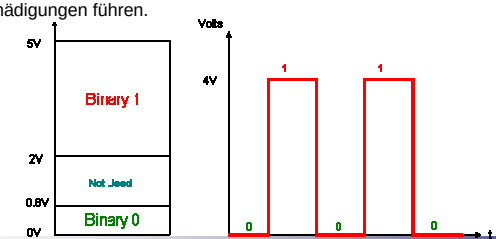
E = A ? B E = x"34" ? x"1C"

? ... beliebige logische Verknüpfung



Darstellung binärer Größen

- In digitalen Systemen werden Informationen in binärer Form dargestellt. Binäre Größen können von jedem Baustein dargestellt werden, der nur 2 Operationszustände besitzt.
  - Beispiel: Schalter. Zuweisung der Zustände 0 für offen und 1 für geschlossen ermöglicht die Darstellung binärer Werte.
  - Typischerweise werden folgende Spannungen zugewiesen:
    - Binäre 1: Eine beliebige Spannung zwischen 2V und 5V
    - Binäre 0: Eine beliebige Spannung zwischen 0V und 0.8V
    - Spannungen zwischen 0.8V und 2V werden nicht genutzt und können zu Beschädigungen führen.



Boolesche Algebra (1/2)

- Boolesche Algebra:
  - Die Booleschen Algebra (speziell die Aussagenlogik) ist ein Formalismus zur Beschreibung der Funktion digitaler Komponenten.
- Binäre oder Boolesche Variable (BV):
  - Variable, die nur zwei Werte annehmen können. Diese Werte werden mit
    L - H , wahr - falsch oder 0 - 1 bezeichnet. Boolesche Variable sind logische Variable.
- Boolesche Operatoren:
  - Auf Boolesche Variable können Boolesche Operatoren angewendet werden: NOT, OR, AND,XOR, ...
- Boolesche Ausdrücke, Boolesche Funktionen:
  - Boolesche Ausdrücke oder Boolesche Funktionen ergeben sich durch Verknüpfung von Boolesche Variable mit Booleschen Operatoren.

Boolesche Algebra (2/2)

- **Definition der Booleschen Algebra**
  - Die Booleschen Algebra ist eine algebraische Struktur. Sie besteht aus einer Menge M = {e1, e2, …}, den Operatoren ψ und ϕ, Axiomen und wichtigen Sätzen. Eine spezielle Boolesche Algebra ist die Aussagenlogik. Sie wird charakterisiert durch folgende Eigenschaften:

    M: {0,1}

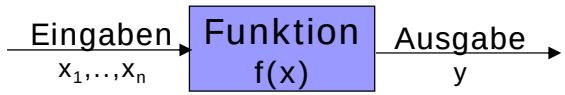
    e: 1 wahr

    n: 0 falsch

    ψ: $\wedge$ zweistellige Operation

    ϕ: $\vee$ zweistellige Operation
- Die Boolesche Algebra wurde von George Boole um 1850 entwickelt
- Claude Shannon verwendete um 1937 die Boolesche Algebra als erster für den Schaltungsentwurf

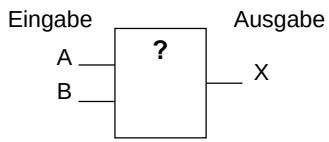
Operation



- Funktion von n Eingangsvariablen:
  - Nur vom gegenwärtigen Wert der Eingänge anhängig, kein "Gedächtnis"
  - $y = f(\mathbf{x}) = f(x_n, ... ,x_3,x_2,x_1)$, $x_i \in \{0,1\}$
  - Für n Eingangsvariablen gibt es 2^n Wertekombinationen.
  - Die Wertekombinationen heißen auch die Belegungen X_j der Eingangsvariablen
- Logische Operationen können auf zwei, sinngemäß gleiche Wege dargestellt werden:
  - Boolescher Ausdruck: endlich, aber nicht eindeutig
  - Wahrheitstabelle: endlich **und** eindeutig

Wahrheitstabellen

- Darstellung der Abhängigkeiten zwischen der Eingabe und Ausgabe
- Alle möglichen Kombinationen der Eingabe (A,B) werden mit der korrespondierenden Ausgabe (X) angegeben.
- Beispiel:



B	A	X
0	0	0
0	1	0
1	0	0
1	1	1

  - Die Ausgabe ist nur 1, wenn beide Eingaben 1 sind.

UND-Operator

Name UND (AND)

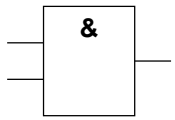
UND wird verwendet, um Bits gezielt auf 0 zu setzen. Dazu hat die Maske an allen Bitpositionen, die übernommen werden sollen, eine 1 und an den Stellen, die rückgesetzt werden sollen, eine 0.

Tabelle

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

Schriftlich $y = a \wedge b$ oder
$= ab$ oder
$= a * b$

Gatter



x“14“ = x"34" ∧ x"1C"

1	1	0	1	0	0	Operand A
0	1	1	1	0	0	Operand B
0	1	0	1	0	0	Ergebnis E

ODER-Operator

Name ODER (OR)

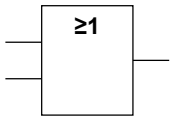
OR wird verwendet, um Bits gezielt auf 1 zu setzen. Dazu hat die Maske an allen Bitpositionen, die übernommen werden sollen, eine 0 und an den Stellen, die gesetzt werden sollen, eine 1.

Tabelle

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

Schriftlich $y = a \vee b$ oder
$= a + b$

Gatter



x“3C“ = x"34“ ∨ x"1C"

1	1	0	1	0	0	Operand A
0	1	1	1	0	0	Operand B
1	1	1	1	0	0	Ergebnis E

NICHT-Operator

Name NICHT (NOT)

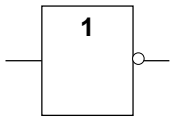
NICHT wird auch als Inverses oder Komplement bezeichnet.

Tabelle

a	y
0	1
1	0

Schriftlich $y = \bar{a}$

Gatter



x“0B“ = ~x"34"

1	1	0	1	0	0	Operand A
0	0	1	0	1	1	Ergebnis E

Gängige logische Operationen: XOR

Name XOR

XOR wird verwendet, um Bits gezielt zu invertieren. Dazu hat die Maske an allen Bitpositionen, die invertiert werden sollen, eine 1 und an allen sonstigen Stellen eine 0.

Tabelle

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

Schriftlich $y = a \leftrightarrow b$ oder
$= a \oplus b$

Sprache "C"
y = a ^ b

x“28“ = x"34" ↔ x"1C"

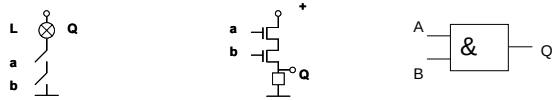
1	1	0	1	0	0	Operand A
0	1	1	1	0	0	Operand B
1	0	1	0	0	0	Ergebnis E

Gatter (1/2)

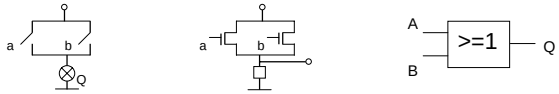
- Elektrische Realisierung mittels Schalter:
  - Relais, Knipser, Röhren, bipolare Transistoren, Feldeffekt – Transistoren (MOS-FET), etc.

Funktion | Realisierung: Knipser | MOS-FET | Gatter

AND



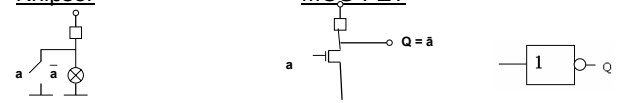
OR



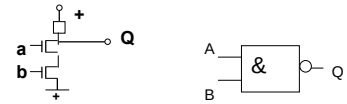
Gatter (2/2)

Funktion | Realisierung: Knipser | MOS-FET | Gatter

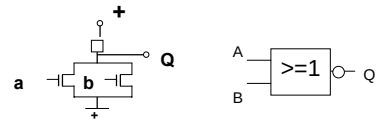
NOT



NAND



NOR



10 Regeln für logische Operationen

Boolesche Variablen a, b Elemente aus {0,1}	$a \vee 0 = a$	$a \wedge 0 = 0$	R1
	$a \vee 1 = 1$	$a \wedge 1 = a$	R2
	$a \vee a = a$	$a \wedge a = a$	R3
	$a \vee \overline{a} = 1$	$a \wedge \overline{a} = 0$	R4
	$\overline{\overline{a}} = a$		R5
Assoziativgesetz	$a \vee (b \vee c) = (a \vee b) \vee c = a \vee b \vee c$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c = a \wedge b \wedge c$	R6
Kommutativgesetz	$a \vee b = b \vee a$	$a \wedge b = b \wedge a$	R7
Distributivgesetz	$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	R8
Absorptionsgesetz	$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$	R9
De Morgan'sches Gesetz	$\overline{(a \vee b)} = \overline{a} \wedge \overline{b}$	$\overline{(a \wedge b)} = \overline{a} \vee \overline{b}$	R10

Anwendung der Regeln

Vereinfachung von Ausdrücken

Beispiel 1:

$$
\begin{aligned}
Y &= A \wedge B \wedge C \wedge D \vee \overline{A} \wedge B \vee \overline{(\overline{B} \vee (A \wedge \overline{A}))} && \text{R4} \\
&= A \wedge B \wedge C \wedge D \vee \overline{A} \wedge B \vee \overline{(\overline{B} \vee 0)} && \text{R1} \\
&= A \wedge B \wedge C \wedge D \vee \overline{A} \wedge B \vee B && \text{...} \\
&= A \wedge B \wedge C \wedge D \vee B \\
&= B
\end{aligned}
$$

Beispiel 2:

$$Y = \overline{(A \wedge B \wedge C) + (D \wedge E \wedge F)}$$

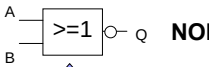
R5: $Y = \overline{(A \wedge B \wedge C)} + \overline{(D \wedge E \wedge F)}$

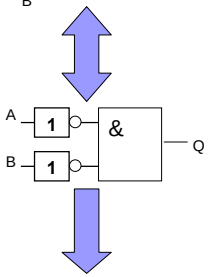
De Morgan: $Y = \overline{(A \wedge B \wedge C)} \wedge \overline{(D \wedge E \wedge F)}$

Implikationen aus dem De Morgan'schen Gesetz

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

$\overline{(a \vee b)}$ **NOR**

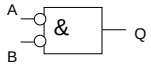




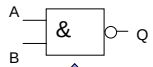
$\overline{a} \wedge \overline{b}$

Alternative NOR Darstellung

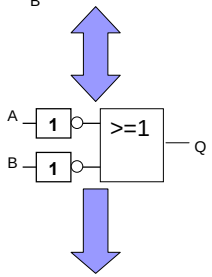
$\overline{a} \wedge \overline{b}$



NAND



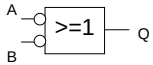
$\overline{(a \wedge b)}$



$\overline{a} \vee \overline{b}$

Alternative NAND Darstellung

$\overline{a} \vee \overline{b}$



NAND-Operator

Name NAND

Das NAND-Symbol ist abgesehen von dem kleinen Kreis am Ausgang identisch mit dem AND-Symbol. Der kleine Kreis stellt die Invertierung dar.

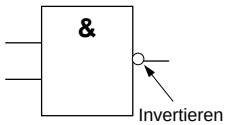
Tabelle

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

Schriftlich $y = a \overline{\wedge} b$

1	1	0	1	0	0	Operand A
0	1	1	1	0	0	Operand B
1	0	1	0	1	1	Ergebnis E

Gatter



NOR-Operator

Name NOR

Das NOR-Symbol ist abgesehen von dem kleinen Kreis am Ausgang identisch mit dem OR-Symbol. Der kleine Kreis stellt die Invertierung dar.

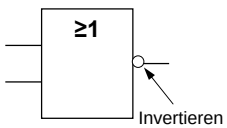
Tabelle

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

Schriftlich $y = a \overline{\vee} b$

1	1	0	1	0	0	Operand A
0	1	1	1	0	0	Operand B
0	0	0	0	1	1	Ergebnis E

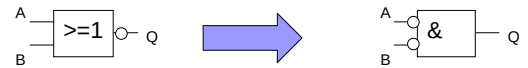
Gatter



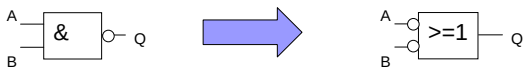
Alternative Darstellung von Gattern

- Für die Gatter UND, ODER, NICHT, NAND und NOR gibt es eine Alternative Darstellung
  - Invertiere jeden Eingang und Ausgang des Standardsymbols
  - Ändere das Operationssymbol von UND zu ODER, oder von ODER zu UND
- Die alternative Darstellung repräsentiert den gleichen physikalischen Baustein.
- Beispiele:

NOR



NAND



Vollständige Operatorensysteme

- Ein vollständiges Operatorensystem erlaubt die Darstellung aller möglichen Funktionen
  - Bei zweistelligen, booleschen Funktionen sind dies 16 Möglichkeiten
- Die folgenden Systeme sind u.a. vollständig:
  - UND, ODER, NICHT
  - UND, NICHT
  - ODER, NICHT
  - NAND
  - NOR
- Beispiel für NAND:
  - NICHT: $\overline{a \wedge a}$
  - UND: $\overline{(\overline{a \wedge b}) \wedge (\overline{a \wedge b})}$
  - ODER $\overline{(\overline{a \wedge a}) \wedge (\overline{b \wedge b})}$

Normalformen (1/2)

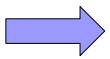
- Es gibt keinen eindeutigen booleschen Ausdruck für eine boolesche Funktion
  - Beschränkung auf ein vollständiges Operatorensystem reicht nicht für Eindeutigkeit
- Die Normalformen dienen dazu, anhand von Zustandstabellen Schaltungen zu entwickeln. Dabei werden logische Funktionen in Schaltkreise umgesetzt.
  - Randbedingungen:
    - die Zahl der Schaltkreise sollte möglichst gering sein
    - die einzelnen Bauelemente sollten möglichst gleich sein
- Standarddarstellung im Operatorensystem UND/ODER/NICHT

Normalform (2/2)

- **Disjunktive Normalform (DNF)**
  - disjunktive Verknüpfung von Konjunktionen
  - Konjunktionen müssen alle Variablen enthalten (bejaht oder negiert)
  - Konjunktionsterme werden **Minterme** genannt
  - In der DNF werden alle UND – Verknüpfungen (Konjunktionen) disjunktiv verknüpft (ODER- verknüpft), deren Ausgangsvariablen den Wert „1“ annehmen.
  - amerikanischer Ausdruck: “**sum of products**“ (SOP)
- **Konjunktive Normalform (KNF)**
  - Konjunktive Verknüpfung von Disjunktionen
  - Disjunktionen müssen alle Variablen enthalten (bejaht oder negiert)
  - Disjunktionsterme werden **Maxterme** genannt
  - In der KNF werden alle ODER – Verknüpfungen (Disjunktionen) konjunktiv verknüpft (UND- verknüpft), deren Ausgangsvariablen den Wert „0“ annehmen.

Übersetzungsregel für disjunktive Normalform

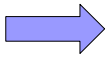
- Man geht von der **Eins**stellenmenge aus.
- Für jede Belegung ersetzt man
  - die 1 durch eine **bejahte** Variable
  - die 0 durch eine **negierte** Variable
- und bildet eine **UND** Verknüpfung (**Min**term).
- Die Maxterme werden **ODER** verknüpft



“**Dis**junktive Normalform“ **D**NF

Übersetzungsregel für konjunktive Normalform

- Man geht von der **Null**stellenmenge aus.
- Für jede Belegung ersetzt man
  - die 0 durch eine **bejahte** Variable
  - die 1 durch eine **negierte** Variable
- und bildet eine **ODER** Verknüpfung (**Max**term).
- Die Maxterme werden **UND** verknüpft



"**Kon**junktive Normalform" **K**NF

DNF Übersetzung

Oktal Index	(x4,x3,x2,x1)	f ->	y
0	0,0,0,0	->	0
1	0,0,0,1	->	0
2	0,0,1,0	->	0
3	0,0,1,1	->	1
4	0,1,0,0	->	0
5	0,1,0,1	->	0
6	0,1,1,0	->	1
7	0,1,1,1	->	0
10	1,0,0,0	->	0
11	1,0,0,1	->	1
12	1,0,1,0	->	-
13	1,0,1,1	->	-
14	1,1,0,0	->	-
15	1,1,0,1	->	-
16	1,1,1,0	->	-
17	1,1,1,1	->	-

Beispiel: Durch 3 teilbare BCD Ziffern

← Logische 1 für:
$\overline{x4} = 0$, x3 = 0, x2 = 1, x1 = 1
$\overline{x4}$ $\overline{x3}$ x2 x1

← Logische 1 für:
$\overline{x4} = 0$, x3 = $\overline{1}$, x2 = 1, x1 = 0
$\overline{x4}$ x3 x2 $\overline{x1}$

← Logische 1 für:
x4 = $\overline{1}$, $\overline{x3} = 0$, x2 = 0, x1 = 1
x4 $\overline{x3}$ $\overline{x2}$ x1

E = {3, 6, 11}

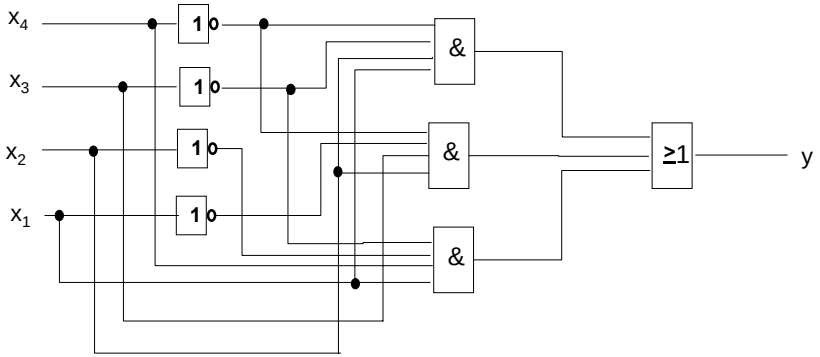
3 (0,0,1,1)
6 (0,1,1,0)
11 (1,0,0,1)

$y = \overline{x4}\ \overline{x3}\ x2\ \overline{x1} \vee x4\ \overline{x3}\ \overline{x2}\ x1 \vee x4\ x3\ x2\ x1$

Übersetzung DNF Gleichung in Hardware

Jede Gleichung läßt sich 1:1 in Hardware umsetzen

$y = \overline{x_4}\ \overline{x_3}\ x_2\ x_1 \vee \overline{x_4}\ x_3\ x_2\ \overline{x_1} \vee x_4\ \overline{x_3}\ \overline{x_2}\ x_1$



Beschreibung von logischen Schaltnetze

- Logische Schaltnetze können mit den grundlegenden logischen Verknüpfungen UND, ODER und NICHT beschrieben werden:
  1. x = AB ∨ C
  2. x = (A ∨ B) C
  3. x = (A ∨ B) C ∨ A
- Die Verwendung nur eines Gattertyps erleichtert die technische Realisierung, weswegen NAND oder NOR basierte Systeme oftmals verwendet werden.
- Logische Schaltnetze können durch die Minimierung der logischen Verknüpfung minimiert werden
  - In vielen Fällen sind einfachere Schaltungen schneller
  - Weniger Gatter reduziert die Kosten
  - Es wird weniger Energie benötigt

Minimierung

- DNF liefert relativ aufwendige Lösungen
  - Baut auf den Mintermen auf
  - Jedes UND Gatter enthält alle Eingangsvariablen
  - Nicht praktikabel in „echten“ Anwendungen
- Die Suche nach einer einfacheren Lösung ist Ziel der Optimierung (Minimierung).
- Voraussetzung jeder Optimierung: Kostenfunktion beschreibt Ziel.
  In unserem Fall: **Möglichst wenige, möglichst kleine Gatter**

  Genauer: Die Anzahl der Eingänge in alle Gatter soll minimal werden.
  In einer Gleichung ist dann die Anzahl der Variablen aller Terme plus die Anzahl der Terme (Länge) minimal.

  $Y = (D\ \overline{C}\ B\ \overline{A}) \vee (D\ C\ \overline{B}) \vee (E\ C) \Rightarrow L = 4+3+2+3 = 12$

- Wie erzielen wir möglichst kleine Gatter? Wir versuchen, durch Anwendung der Rechenregeln, die einzelnen Terme kürzer zu machen.

Minimierungsverfahren

- Es gibt 3 Arten von Minimierungsverfahren
  - algebraische Verfahren
  - graphische Verfahren (z.B. KV-Diagramme)
  - tabellarische/algorithmische Verfahren (z.B. Quine/McCluskey-Verfahren)
- Algebraische Verfahren werden praktisch nicht verwendet
- Graphische Verfahren sind für bis zu 6 Variablen handhabbar
- Tabellarische Verfahren sind auch für mehr als 6 Variablen geeignet

Algebraische Minimierung

Vereinfachung durch Ausklammern und Anwendung der Regeln:

Bsp: $y = \overline{c}\ b\ a \vee \overline{c}\ \overline{b}\ a = \overline{c}\ a\ (b \vee \overline{b}) = \overline{c}\ a\ 1 = \overline{c}\ a$

In Belegungsschreibweise

	c	b	a		y
$\overline{c}$ b a	0	1	1	->	1
$\overline{c}$ $\overline{b}$ a	0	0	1	->	1

Wir suchen Belegungen, die

Einsstellen sind und
sich nur in einer Komponente unterscheiden
(benachbart sind)

Blöcke

- Zwei Belegungen, die sich nur in einer Komponente unterscheiden, heißen benachbart.
- Die identischen Komponenten heißen "gebundene", die unterschiedlichen "freie" Komponenten.
- Die freien Komponenten werden durch einen "-" gekennzeichnet.
- Es entsteht ein (Belegungs-) Block.
- Ein Block mit r freien Komponenten enthält ("überdeckt") 2^r Belegungen.
- Beispiel: 4 (0,1,0,0)
  5 (0,1,0,1)
  → (0,1,0,-)

Minimierung über Nachbarschaftsbeziehungen

Oktal Index	(x3,x2,x1)	f ->	y
0	0,0,0	->	0
1	0,0,1	->	0
2	0,1,0	->	0
3	0,1,1	->	0
4	1,0,0	->	1
5	1,0,1	->	1
6	1,1,0	->	0
7	1,1,1	->	1

Zugehörige DNF:

$y = (x_3 \bar{x}_2 \bar{x}_1) \vee (x_3 x_2 x_1) \vee (x_3 x_2 x1)$

Benachbarte Belegungen:

4: 1, 0, 0
5: 1, 0, 1
Block: 1, 0, -

Minimierte Gleichung:

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

Bildung größerer Blöcke

Oktal Index	(x3,x2,x1)	f ->	y
0	0,0,0	->	0
1	0,0,1	->	0
2	0,1,0	->	0
3	0,1,1	->	0
4	1,0,0	->	1
5	1,0,1	->	1
6	1,1,0	->	1
7	1,1,1	->	1

Zugehörige DNF:

$y = (x_3 \bar{x}_2 \bar{x}_1) \vee (x_3 x_2 x_1) \vee (x_3 x_2 x_1) \vee (x_3 x_2 x_1)$

Benachbarte Belegungen:

4: 1, 0, 0
5: 1, 0, 1
Block: 1, 0, -

6: 1, 1, 0
7: 1, 1, 1
Block: 1, 1, -

1, 0, -
1, 1, -
Block: 1, -, -

Minimierte Gleichung:

$y = x_3$

Blockbildung bei komplizierteren Funktionen

Oktal Index	(x4,x3,x2,x1)	f ->	y
0	0,0,0,0	->	1
1	0,0,0,1	->	0
2	0,0,1,0	->	1
3	0,0,1,1	->	0
4	0,1,0,0	->	0
5	0,1,0,1	->	1
6	0,1,1,0	->	1
7	0,1,1,1	->	1
10	1,0,0,0	->	0
11	1,0,0,1	->	1
12	1,0,1,0	->	1
13	1,0,1,1	->	0
14	1,1,0,0	->	0
15	1,1,0,1	->	1
16	1,1,1,0	->	1
17	1,1,1,1	->	1

Mögliche Blöcke:

0/2: 00-0
2/6: 0-10
2/12: -010
5/7: 01-1
5/15: -101
6/7: 011-
6/16: -110
7/17: -111
11/15: 1-01
12/16: 1-10
15/17: 11-1
16/17: 111-

5/7/15/17: -1-1
6/7/16/17: -11-
2/12/6/16: --10

Minimierte Gleichung: $y = x_4 x_3 x_1 \vee x_4 x_2 x_1 \vee x_3 x_1 \vee x_2 x_1$

Zusammenfassung Blockbildung

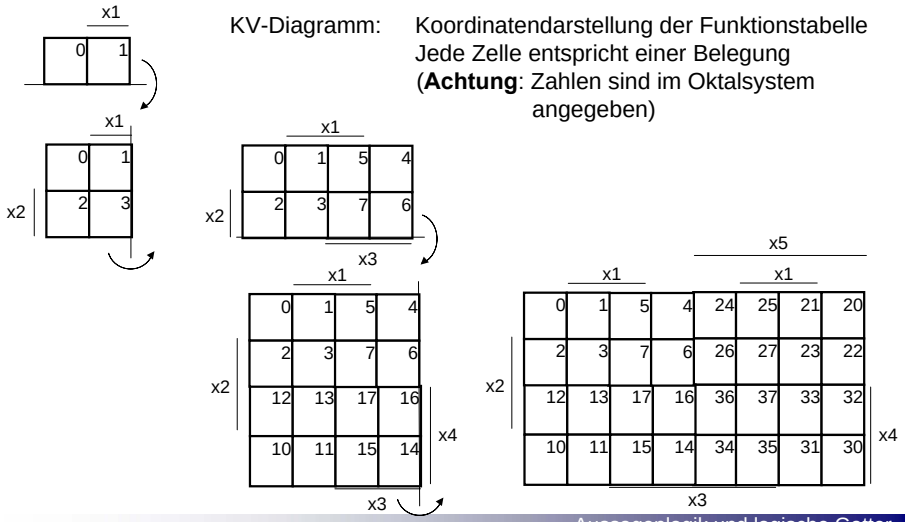
- Die Bildung von Blöcken kann zur Minimierung herangezogen werden.
  - Betrachtet werden nur die Einsstellen
  - Benachbarte Belegungen werden zu Blöcken zusammengefasst.
- Benachbarte Blöcke können zu größeren Blöcken zusammengefasst werden
  - es gelten dieselben Regeln.
  - Blöcke bestehen immer aus 2^r Belegungen (r = Anzahl der Striche)
  - Ein Block, der sich nicht mehr vergrößern lässt, heißt "**Primblock**"
- Gesucht ist die minimale Anzahl von möglichst großen Blöcken, die alle Einsstellen überdeckt.
- Übergang zur disjunktiven Form
  - Für Blöcke gelten dieselben Übersetzungsregeln, aber
  - freie Komponenten ("Striche") werden ignoriert.
  - Ein aus einem Primblock entstehender Term heißt "**Primimplikant**"

Karnaugh-Veitch-Diagramme

- KV-Diagramme bieten einen graphischen Weg zur Minimierung
- Es können minimale konjunktive oder disjunktive Formen gebildet werden

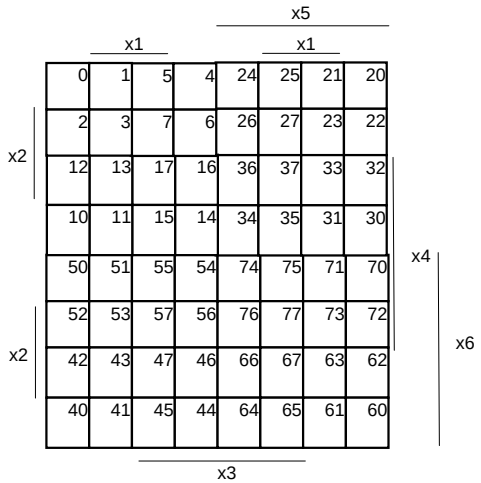
Minimierung per KV-Diagramm

Erstellung durch wechselweise horizontales und vertikales Spiegeln



KV-Diagramm: Koordinatendarstellung der Funktionstabelle
Jede Zelle entspricht einer Belegung
(**Achtung**: Zahlen sind im Oktalsystem angegeben)

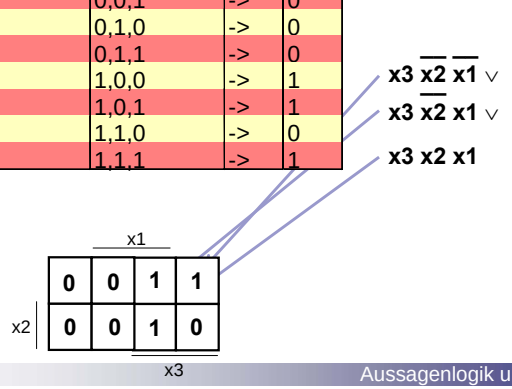
KV-Diagramm



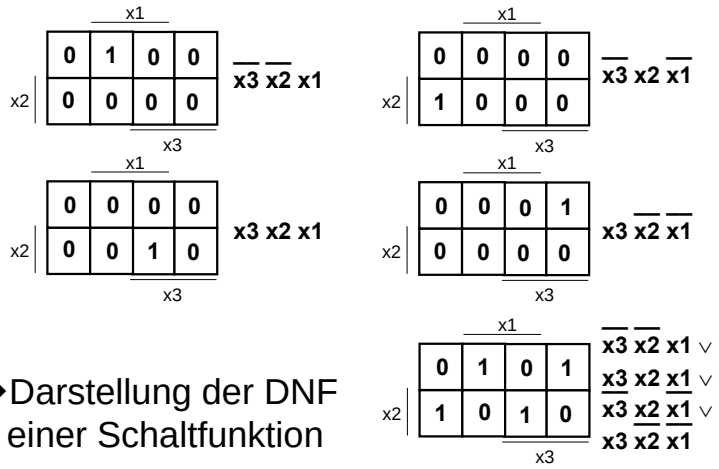
Minimierung per KV-Diagramm

Eintragung in das KV-Diagramm

Oktal Index	(x3,x2,x1)	f ->	y
0	0,0,0	->	0
1	0,0,1	->	0
2	0,1,0	->	0
3	0,1,1	->	0
4	1,0,0	->	1
5	1,0,1	->	1
6	1,1,0	->	0
7	1,1,1	->	1



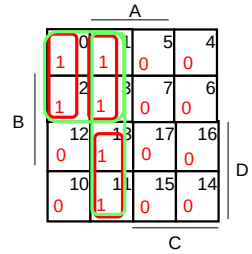
Interpretation eines KV-Diagramms



➔Darstellung der DNF einer Schaltfunktion

Blockbildung im KV-Diagramm

Fall	D	C	B	A	Z
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
10	1	0	0	0	0
11	1	0	0	1	1
12	1	0	1	0	0
13	1	0	1	1	1
14	1	1	0	0	0
15	1	1	0	1	0
16	1	1	1	0	0
17	1	1	1	1	0



DNF: $Z = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \vee A\,\overline{B}\,\overline{C}\,\overline{D} \vee \overline{A}\,B\,\overline{C}\,\overline{D} \vee A\,B\,\overline{C}\,\overline{D}$

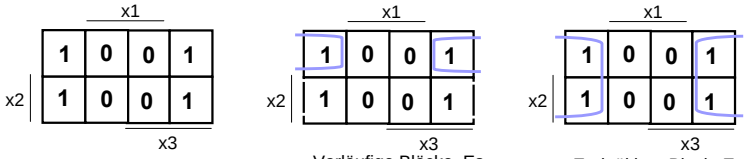
Minterme: 0: 0000 1: 0001 13: 1011
(Blöcke) 2: 0010 00-0 3: 0011 00-1 11: 1001 10-1

Minimierte Gleichung: $Z = \overline{A}\,\overline{C}\,\overline{D} \vee A\,\overline{C}\,\overline{D} \vee \overline{A}CD$

Primblöcke: 00-- -0-1

Minimierte Gleichung mit Primtermen: $Z = \overline{C}\,\overline{D} \vee A\,\overline{C}$

Beispiel für Blockbildung in KV-Diagrammen



Vorläufige Blöcke. Es entfällt je x3.

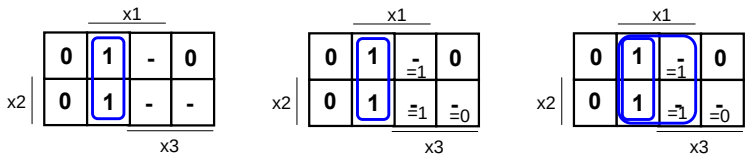
Endgültiger Block. Es entfällt x2.

Kann ein vorläufiger Block nicht mehr durch einen weiteren, symmetrisch gelegenen gleich großen Block vergrößert werden, so erhält man den maximal großen Block: **Primblock**
Der zugehörige Term heißt **Primterm**.

Jede vollständige Blocküberdeckung ist eine gültige Darstellung einer Funktion.
Vollständig: Jede Einsstelle wird von einem Block überdeckt.

Blockbildung bei unvollständig definierten Schaltfunktionen

Freistellen (**do not care** Stellen) können 0 oder 1 sein.
Festlegung derart, dass maximal große Blöcke entstehen.



Minimierung per KV-Diagramm

- Minimierungsverfahren
  1. Spezifikation der Funktion durch Terme
  2. Eintragung in das KV-Diagramm
  3. Bestimmen der Primblöcke
     - Sukzessive Bildung von Blöcken mit 2, dann 4, dann 8 Belegungen, usw.
     - Verfolgung aller möglichen Kombinationen
     - Wenn Blockbildung abbricht, sind alle Primblöcke gefunden.
  4. Auswahl der kleinsten Anzahl Primblöcke, die zur vollständigen Überdeckung nötig sind
     - Identifikation der Kerne: Markierung aller Primblöcke, welche alleine eine Funktionsstelle überdecken.
     - Falls diese bereits alle Stellen überdecken, ist minimale Lösung erreicht.
     - Reichen diese nicht zur Überdeckung aller Stellen aus, müssen weitere Primblöcke hinzugenommen werden.
  5. Bildung eines kürzesten Ausdrucks (DMF: disjunktive **Minimalform**)

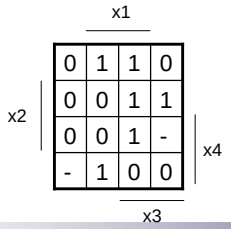
Minimierung per KV-Diagramm (1/3)

Minimierungsverfahren

Beispiel

Bereits erledigt:
1. Spezifikation der Funktion durch Terme
2. Eintragung in das KV-Diagramm

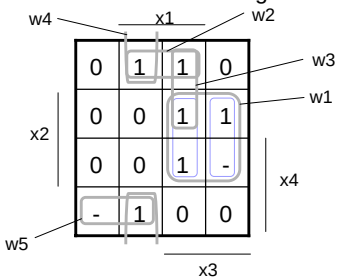
Ergebnis:



Minimierung per KV-Diagramm (2/3)

Beispiel

3. Bestimmen der Primblöcke
   - Sukzessive Bildung von Blöcken mit 2, dann 4, dann 8 Belegungen, usw.
   - Verfolgung aller möglichen Kombinationen
   - Wenn Blockbildung abbricht, sind alle Primblöcke gefunden.



=>

Primimplikanten:

$w1 = x3\ x2$

$w2 = \overline{x4}\ \overline{x2}\ x1$

$w3 = \overline{x4}\ x3\ x1$

$w4 = \overline{x3}\ \overline{x2}\ x1$

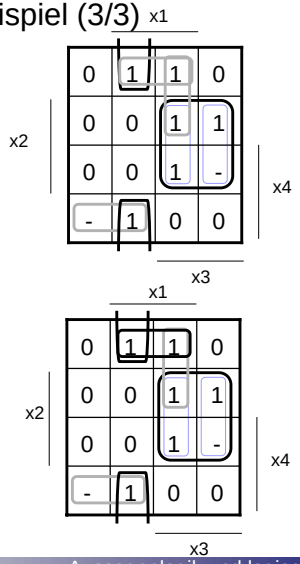
$w5 = x4\ \overline{x3}\ \overline{x2}$

Grau gekennzeichnete Blöcke: Primblöcke

Minimierung per KV-Diagramm Beispiel (3/3)

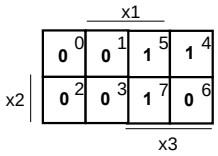
4. Auswahl der kleinsten Anzahl Primblöcke, die zur vollständigen Überdeckung nötig sind
   - Identifikation der Kerne: Markierung aller Primblöcke, welche alleine eine Funktionsstelle überdecken.
   - Falls diese bereits alle Stellen überdecken, ist minimale Lösung erreicht.
   - Reichen diese nicht zur Überdeckung aller Stellen aus, müssen weitere Primblöcke hinzugenommen werden.

5. Bildung eines kürzesten Ausdrucks (DMF: disjunktive Minimalform)



Minimierung mittels KV-Diagramm

Oktal Index	(x3,x2,x1)	f ->	y
0	0,0,0	->	0
1	0,0,1	->	0
2	0,1,0	->	0
3	0,1,1	->	0
4	1,0,0	->	1
5	1,0,1	->	1
6	1,1,0	->	0
7	1,1,1	->	1



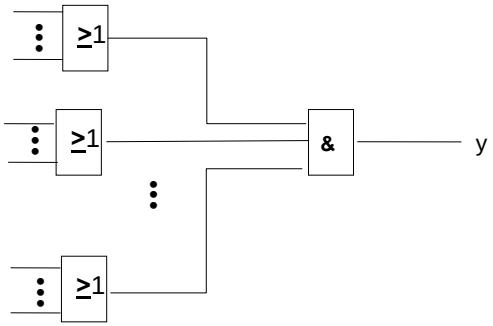
Blöcke:
1 0 –
1 - 1

Kern-Blöcke:
1 0 -
1 - 1

Minimierte Gleichung:

$y = x3\ \overline{x2} \vee x3\ x1$

Minimierung in der konjunktiven Form



z.B.

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

Beispiel

Durch 3 teilbare BCD Ziffern

Bei BCD kodierten Ziffern können nur Werte von 0 bis 9 vorkommen:

Das Verhalten der Schaltung bei Eingangswerten > 9 (11_8) ist gleichgültig.

R = { 12,13,14,15,16,17}
E = { 3,6,11}
N = {0,1,2,4,5,7,10}

Oktal Index	(x4,x3,x2,x1)	f ->	y
0	0,0,0,0	->	0
1	0,0,0,1	->	0
2	0,0,1,0	->	0
3	0,0,1,1	->	1
4	0,1,0,0	->	0
5	0,1,0,1	->	0
6	0,1,1,0	->	1
7	0,1,1,1	->	0
10	1,0,0,0	->	0
11	1,0,0,1	->	1
12	1,0,1,0	->	-
13	1,0,1,1	->	-
14	1,1,0,0	->	-
15	1,1,0,1	->	-
16	1,1,1,0	->	-
17	1,1,1,1	->	-

Ermittlung der Maxterme

Oktal Index	(x4,x3,x2,x1)	f ->	y
0	0,0,0,0	->	0
1	0,0,0,1	->	0
2	0,0,1,0	->	0
3	0,0,1,1	->	1
4	0,1,0,0	->	0
5	0,1,0,1	->	0
6	0,1,1,0	->	1
7	0,1,1,1	->	0
10	1,0,0,0	->	0
11	1,0,0,1	->	1
12	1,0,1,0	->	-
13	1,0,1,1	->	-
14	1,1,0,0	->	-
15	1,1,0,1	->	-
16	1,1,1,0	->	-
17	1,1,1,1	->	-

←Logische 0 für:

$x4 \vee x3 \vee x2 \vee x1 = 0$

←Logische 0 für:

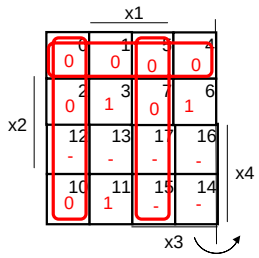
$\overline{x4} \vee x3 \vee x2 \vee x1 = 0$

KNF für Beispiel

N = {0,1,2,4,5,7,10} y =

0 (0,0,0,0)	$(x4 \vee x3 \vee x2 \vee x1)$
1 (0,0,0,1)	$\wedge (x4 \vee x3 \vee x2 \vee \overline{x1})$
2 (0,0,1,0)	$\wedge (x4 \vee x3 \vee \overline{x2} \vee x1)$
4 (0,1,0,0)	$\wedge (x4 \vee \overline{x3} \vee x2 \vee x1)$
5 (0,1,0,1)	$\wedge (x4 \vee \overline{x3} \vee x2 \vee \overline{x1})$
7 (0,1,1,1)	$\wedge (x4 \vee \overline{x3} \vee \overline{x2} \vee \overline{x1})$
10 (1,0,0,0)	$\wedge (\overline{x4} \vee x3 \vee x2 \vee x1)$

Minimierung mit KV-Diagramm



$y = (x2 \vee x4)$

$\wedge (x1 \vee x3)$

$\wedge (\overline{x3} \vee \overline{x1})$

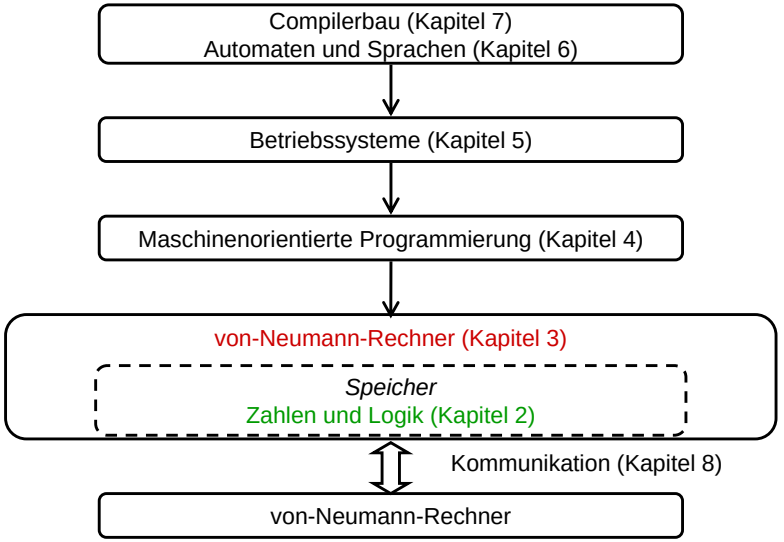
Überblick

- Zahlen
  - Informationsdarstellung
  - Zahlensysteme
  - Rechnerarithmetik
- Logik
  - Aussagenlogik und logische Gatter
  - **Prädikatenlogik**

Prädikatenlogik

- Siehe Tafelanschrieb

Ausblick


Compilerbau (Kapitel 7)
Automaten und Sprachen (Kapitel 6)
Betriebssysteme (Kapitel 5)
Maschinenorientierte Programmierung (Kapitel 4)
von-Neumann-Rechner (Kapitel 3)
Speicher
Zahlen und Logik (Kapitel 2)
Kommunikation (Kapitel 8)
von-Neumann-Rechner