

Digitaltechnik & Rechnersysteme

KV-Minimierung, Don't Cares, Spezielle Schaltnetze, Arithmetik

Martin Kumm



WiSe 2025/2026

Was bisher geschah...



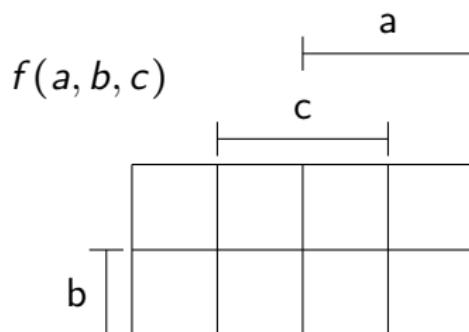
- Boolesche Algebra
- Abgeleitete Operatoren (NAND, NOR, XOR, Äquivalenz)
- Normalformen
 - Umrechnung in Normalformen (z.B. DNF in KDNF)
 - NAND/NOR Umformung
- KV-Diagramme
 - Aufbau
 - Ordnung von Termen

Vorlesungsaufgabe



a	b	c		$f(a, b, c, d)$
0	0	0		1
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		0
1	1	0		1
1	1	1		0

Tragen Sie die Funktion in das KV-Diagramm ein.



Darstellungsalternativen Schaltnetze



Darstellungsformen für Schaltfunktionen (gleichberechtigt)

- Wahrheitstabelle
- **KV-Diagramm**
- Boolesche Funktion (Polynomdarstellung)
- Schaltbild (grafische Darstellung durch Schaltsymbole)

Für diese gilt:

- Für eine Wahrheitstabelle (**KV-Diagramm**) existieren mehrere Boolesche Funktionen und **mehrere** Schaltbilder
- Für eine Boolesche Funktion existieren **mehrere** Schaltbilder aber **genau eine** Wahrheitstabelle (**KV-Diagramm**)
- Für ein Schaltbild existiert **genau eine** Boolesche Funktion und **genau eine** Wahrheitstabelle (**KV-Diagramm**)

Inhalte



- 1 Wrap-Up
- 2 Minimierung mit KV-Diagrammen
- 3 Don't Care
- 4 MUX
- 5 Decoder

Minimierung mit KV-Diagrammen

Diagramm eines KV-Diagramms für die Funktion $h(a, b, c, d)$. Die Variablen sind horizontal von links nach rechts: a , b , c , d . Die Werte der Funktion sind in den Kästen angegeben.

$h(a, b, c, d)$			
		b	d
		a	
0	1	1	0
1	0	1	1
2	1	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	0	0	1
7	1	0	1
8	1	0	0
9	0	1	0
10	1	1	1
11	0	0	1
12	0	0	0
13	0	1	0
14	1	0	1
15	1	1	1
16	0	0	0

Diagramm eines KV-Diagramms für die Funktion $h(a, b, c, d)$. Die Variablen sind horizontal von links nach rechts: a , b , c , d . Die Werte der Funktion sind in den Kästen angegeben. Der Minterm m_11 ist blau eingekreist.

$h(a, b, c, d)$			
		b	d
		a	
0	1	1	0
1	0	1	1
2	1	1	1
3	1	0	1
4	0	1	0
5	1	1	1
6	0	0	1
7	1	0	1
8	1	0	0
9	0	1	0
10	1	1	1
11	0	0	1
12	0	0	0
13	0	1	0
14	1	0	1
15	1	1	1
16	0	0	0

Jede »1« (»0«) repräsentiert einen Minterm (Maxterm)

Minterme (Maxterme), die sich in einer Variable unterscheiden liegen immer benachbart

Minterme sind Terme **nullter** Ordnung.

Diese lassen sich vereinfachen und man erhält Terme **erster Ordnung**: $\bar{a}\bar{b}cd + ab\bar{c}d = acd$

Minimierung mit KV-Diagrammen

h(a, b, c, d)

		b	
		d	
a	c		
0	1	1	0
1	0	1	0
2	1	1	1
3	0	0	1
4	1	1	1
5	0	0	0
6	1	1	0
7	0	0	0
8	1	0	0
9	0	0	0
10	1	1	1
11	0	0	1
12	1	0	0
13	0	0	0
14	1	1	0
15	0	0	0

Ausnahme: Ränder des Diagramms → umklappen

Der Term 1. Ordnung lautet: $a\bar{b}c\bar{d} + abc\bar{d} = ac\bar{d}$

Minimierung mit KV-Diagrammen

$h(a, b, c, d)$			
		b	
		d	
0	1	1	0
1	1	5	4
2	0	1	0
3	1	7	6
10	1	15	14
11	1	1	1
8	1	0	0
9	0	13	12

Zwei benachbarte Terme 1. Ordnung lassen sich zu einem Term 2. Ordnung zusammenfassen:

$$acd + ac\bar{d} = ac$$

Terme höchster Ordnung (= Primterme = Primimplikant) lassen sich direkt aus KV-Diagramm ablesen!

Primimplikant (Definition)



Primimplikant (Primterm): Term, der sich nicht weiter vereinfachen (zusammenfassen) lässt. (Ein Term mit maximaler Ordnung.) – Größtmögliche Zusammenfassung von 1, 2, 4, 8, etc. 1en (0en) im KV-Diagramm

1en (0en) können dabei mehrfach durch Primimplikanten überdeckt werden.

Das Ziel ist folglich möglichst wenige Primimplikanten zu verwenden.

Zur eindeutigen Minimierung müssen Primimplikanten weiter klassifiziert werden.

Klassifizierung Primimplikanten



Kernprimimplikant KPI (essentieller Primterm): Primimplikant, der zur Realisierung einer Funktion unbedingt erforderlich ist. Die Minterme aus denen er entstand, können nicht anders überdeckt werden. Diese werden zur Minimierung zwingend benötigt!

Absolut eliminierbarer Primimplikant API: Primimplikant, dessen Minterme (Maxterme) alle von Kernprimimplikanten überdeckt werden. Diese können zur Minimierung weggelassen werden.

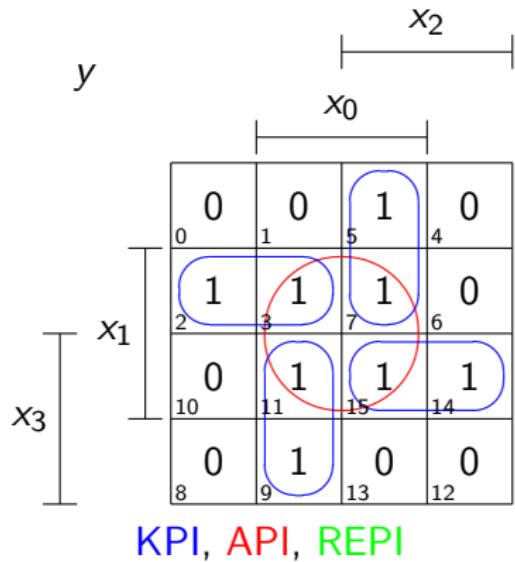
Alle weiteren Primimplikanten sind **relativ eliminierbare Primimplikanten** (REPI). Hier muss zur Minimierung eine Auszahl erfolgen!

Minimierung mit KV-Diagrammen

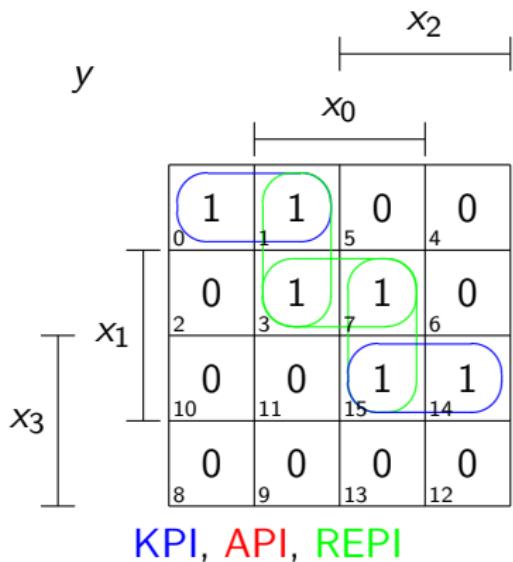


Zur Minimierung müssen **alle** KPI, **kein** API und eine minimale Anzahl **REPIs** verwendet werden.

Beispiel 1



Beispiel 2



Vorlesungsaufgabe

Markieren Sie alle Primimplikanten.

Diagram showing a Karnaugh map for the function $f(a, b, c, d)$. The variables are labeled a , b , c , and d .

The Karnaugh map is a 4x4 grid:

	$a=0, b=0$	$a=0, b=1$	$a=1, b=0$	$a=1, b=1$
$c=0, d=0$	0	1	1	1
$c=0, d=1$	0	1	1	0
$c=1, d=0$	1	1	0	0
$c=1, d=1$	1	1	0	1

Welche Primimplikanten sind zur Minimierung notwendig?

Don't Care Belegungen

Häufig tritt die Situation auf, dass nicht für jede Belegung x der Wert der Funktion $f(x)$ zugeordnet werden muss oder kann. Es kann vielmehr offen bleiben, ob $f(x) = 1$ oder $f(x) = 0$ gesetzt wird.

Man bezeichnet solche Zuordnungen mit **don't care** und spricht von einer **Redundanz** oder **Freistelle** der Funktion.

Statt einer 0 oder 1 wird häufig das Zeichen „–“ zugeordnet (oft auch: „d“ oder „*“).

Dies stellt aber keinen dritten Wert dar, sondern zeigt nur an, dass an dieser Stelle die Funktion wahlweise zu 0 oder zu 1 gesetzt werden kann. Das lässt sich bei der Minimierung ausnutzen!

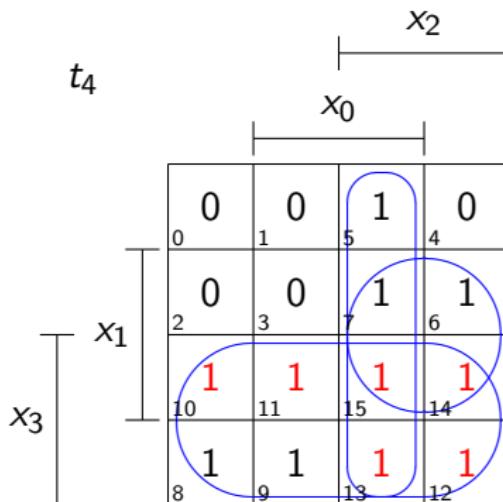
Beispiel für Don't Care

Thermometer-Code für die Ziffern 0 ... 9

x_3	x_2	x_1	x_0	t_8	t_7	t_6	t_5	t_4	t_3	t_2	t_1	t_0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	1	1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	1	1	1	1
0	1	0	1	0	0	0	0	1	1	1	1	1
0	1	1	0	0	0	0	1	1	1	1	1	1
0	1	1	1	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1
1	0	1	0	—	—	—	—	—	—	—	—	—
1	0	1	1	—	—	—	—	—	—	—	—	—
1	1	0	0	—	—	—	—	—	—	—	—	—
1	1	0	1	—	—	—	—	—	—	—	—	—
1	1	1	0	—	—	—	—	—	—	—	—	—
1	1	1	1	—	—	—	—	—	—	—	—	—

Beispiel für Don't Care

x_3	x_2	x_1	x_0	t_4
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

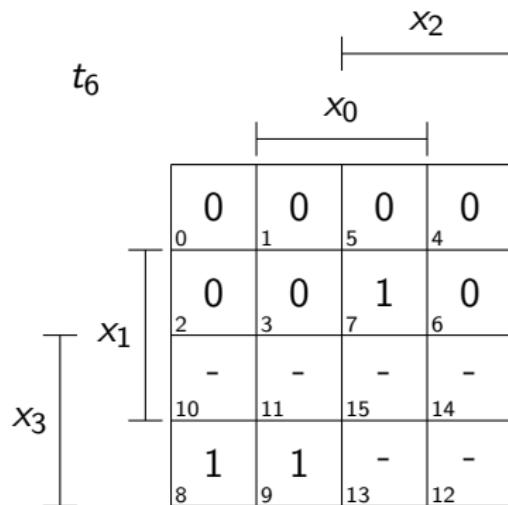


$$t_4 = x_3 + x_0x_2 + x_1x_2$$

Vorlesungsaufgabe

Bestimmen Sie die Primimplikanden für t_6 !

x_3	x_2	x_1	x_0	t_6
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	-
1	0	1	1	-
1	1	0	0	-
1	1	0	1	-
1	1	1	0	-
1	1	1	1	-



Hauptklassen von Schaltfunktionen

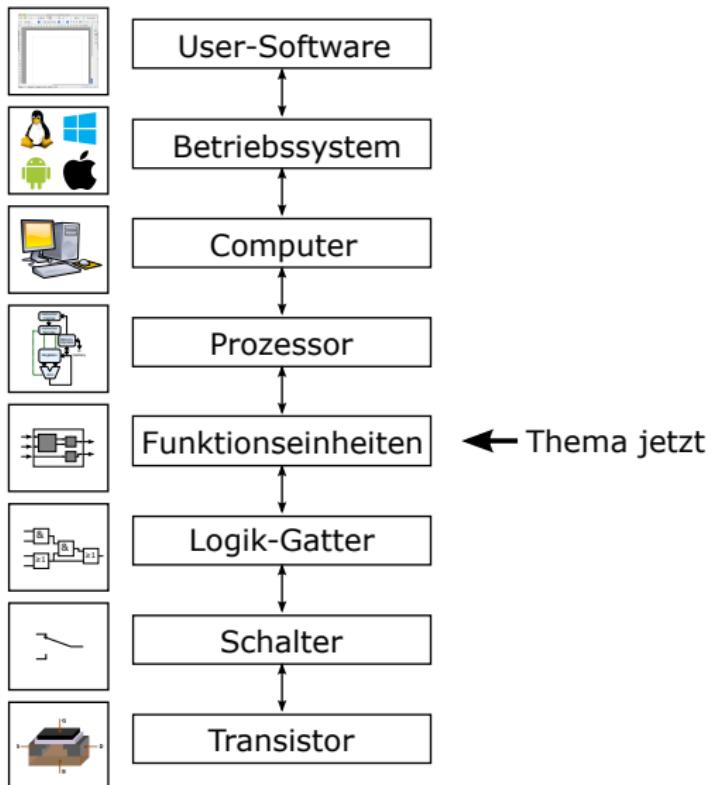


Man definiert zwei Hauptklassen von Schaltfunktionen: Eine Schaltfunktion heißt

- ① **vollständig** (definiert), wenn für alle Belegungen x ein Funktionswert $f(x) \in \{0, 1\}$ fest zugeordnet wird.
- ② **unvollständig** (definiert), wenn es mindestens eine Belegung x gibt, der kein Funktionswert $f(x) \in \{0, 1\}$ fest zugeordnet wird.

Wegen $|\{0, 1\}^n| = 2^n$ lässt sich bei unvollständigen Schaltfunktionen aus jeweils zwei Teilmengen die dritte bestimmen.

Die Macht der Abstraktion



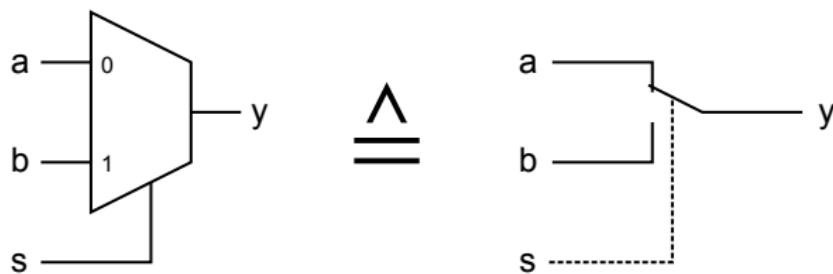
Multiplexer (MUX)



Ein 2 : 1 Multiplexer (MUX) kann 2 Eingänge auf einen Ausgang schalten

Gesteuert wird dies über einen Steuer-Eingang (*Select*)

Funktionsweise: Wenn der Select-Eingang $s = 0$, wird Eingang 0 durchgeschaltet, wenn $s = 1$, wird Eingang 1 durchgeschaltet.



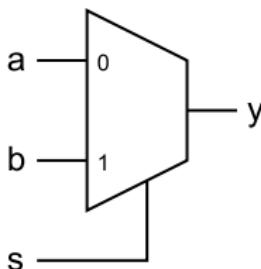
Anwendungsbereiche: Programmierbare (Rechner-)verbindungen

Multiplexer (MUX)



Funktionsweise: Wenn der Select-Eingang $s = 0$, wird Eingang 0 durchgeschaltet, wenn $s = 1$, wird Eingang 1 durchgeschaltet.

Schaltsymbol:



Wahrheitstabelle:

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

$$\text{KDNF: } y = \bar{s} a \bar{b} + \bar{s} a b + s \bar{a} b + s a b$$

Multiplexer



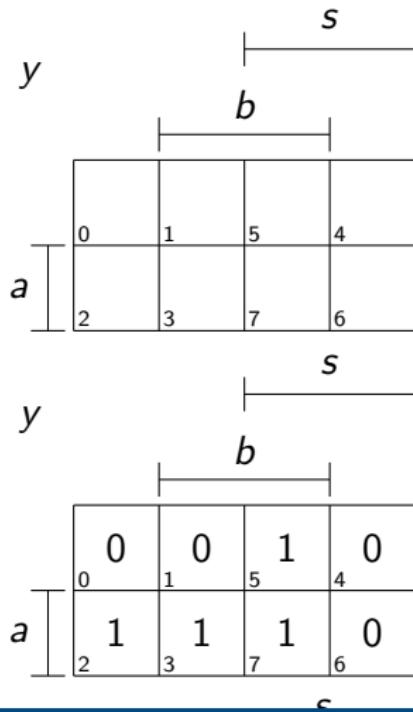
Vereinfachung:

$$\begin{aligned}y &= \bar{s} a \bar{b} + \bar{s} ab + s \bar{a} b + sab \\&= \bar{s} a (\bar{b} + b) + sb (\bar{a} + a) \\&= \bar{s} a \underbrace{(\bar{b} + b)}_{=1} + sb \underbrace{(\bar{a} + a)}_{=1} \\&= \bar{s} a + sb\end{aligned}$$

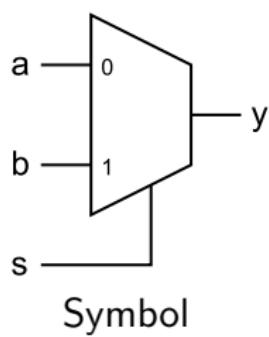
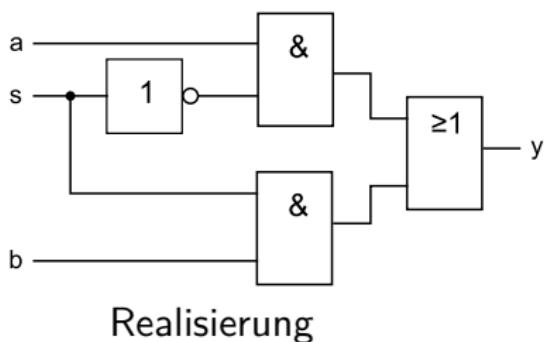
Multiplexer

KV-Diagramm:

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



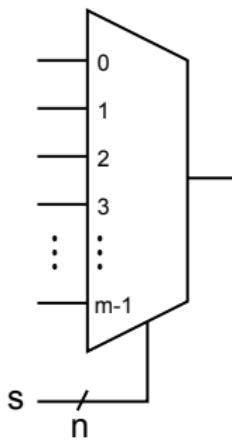
2:1 Multiplexer Symbol



Allgemeiner Multiplexer



Bei $m = 2^n$ Eingängen werden n Steuerleitungen für die Auswahl benötigt.



Die Ein/Ausgabe kann hier auch mehr Bits/Datenwörter umfassen.

Decoder



Schaltnetz, das n Eingänge auf 2^n Ausgänge abbildet.

Schaltet für jede Eingangskombination genau einen Ausgang auf 1
(auch *one hot code* genannt)

Allgemein können Ausgangsseitig weniger als 2^n Ausgänge vorgesehen sein: $m \leq 2^n$ bei n -zu- m -Decodern

Beispiel: 3 : 8 Decoder

Wahrheitstabelle zum 3 : 8 Decoder:

Index	a	b	c	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Ausgang i entspricht Minterm M_i .

3 : 8 Decoder Schaltbild & Symbol

