

Digitaltechnik

Wintersemester 2017/2018

5. Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT





1. Einleitung
2. Bubble Pushing
3. Logik-Realisierung mit Basis-Gattern
4. Karnaugh Diagramme
5. Zusammenfassung

Einleitung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

0100000101000111100010100010100111001111
0010010010010101100001101111010000000111
1011011110000100010101011111000100001101
0110101110000100011101110001110101110011
0010111000101100110110001101100100000111
1001100110000010100111110101111110001111
1111001100010111111101011011101100110001
1101101011011001000000111001100100011010
0100100000010001110100010000000010111110
1010011101011111100010100110000001111101
1101101101110101111110111010101010011000
0010101100000100101001011001010111000010
1111001110001010100011100100011110100101
1110101000001000000100011110110100111011
1001111100101101111110111001101100111001
1110111010000001000110101100100111001111



- ▶ Anmeldung zu Studienleistung und Fachprüfung offen
- ▶ mündliche Prüfung für Austauschstudenten

Rückblick auf die letzten Vorlesungen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Komplexität und (digitale) Abstraktion
- ▶ Zahlensysteme
- ▶ Logikgatter
- ▶ Physikalische Realisierung von Logikgattern
 - ▶ Logikpegel
 - ▶ Feldeffekt-Transistoren
 - ▶ CMOS-Gatter
 - ▶ Leistungsaufnahme
 - ▶ Moor'sches Gesetz
- ▶ Kombinatorische Logik
 - ▶ Boole'sche Gleichungen
 - ▶ Boole'sche Algebra

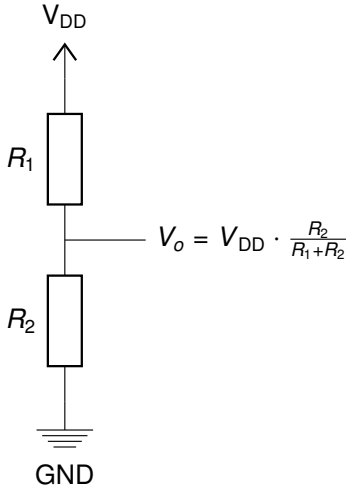


Harris 2013
Kapitel 1,2

Konzept des Spannungsteilers

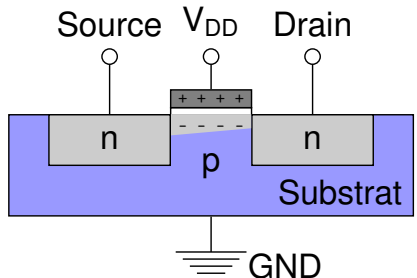
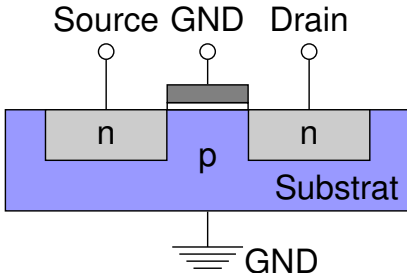


TECHNISCHE
UNIVERSITÄT
DARMSTADT



Wiederholung: nMOS

- ▶ Gate = 0, ausgeschaltet (R_{sd} sehr groß)
- ▶ Gate = 1, eingeschaltet (R_{sd} sehr klein)

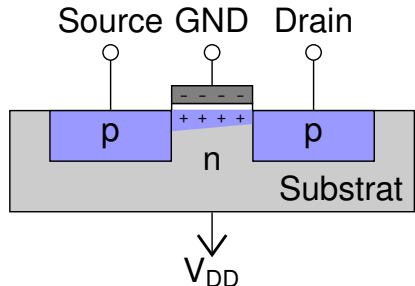
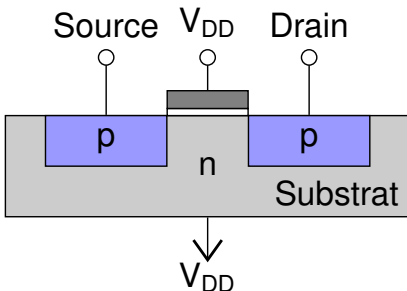


Wiederholung: pMOS

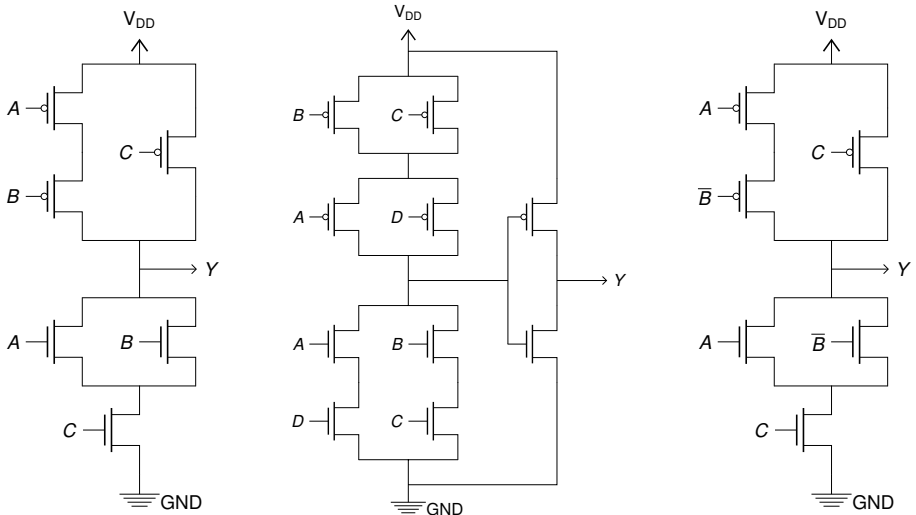


TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Gate = 1, ausgeschaltet (R_{sd} sehr groß)
- ▶ Gate = 0, eingeschaltet (R_{sd} sehr klein)



Wiederholung: CMOS



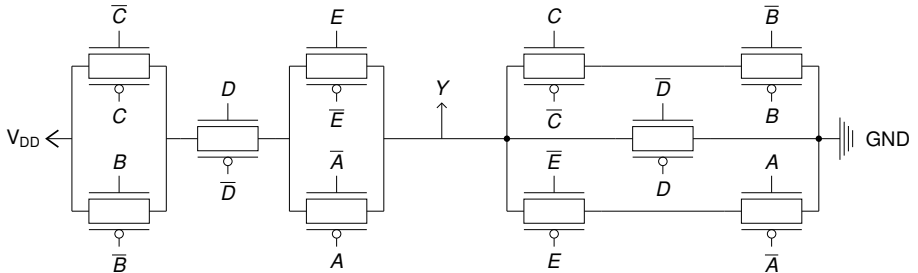
Konstruktion von CMOS-Schaltungen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Y enthält nur negierten Variablen (bspw. $Y = \overline{A} + \overline{B}$)
 - ▶ pMOS konstruieren (OR parallel, AND seriell)
 - ▶ nMOS komplementär
 - ▶ keine negierten Variablen an Gates
2. \overline{Y} enthält nur nicht-negierten Variablen (bspw. $\overline{Y} = A + B$)
 - ▶ nMOS konstruieren (OR parallel, AND seriell)
 - ▶ pMOS komplementär
 - ▶ keine negierten Variablen an Gates
3. Y enthält nur nicht-negierten Variablen (bspw. $Y = A + B$)
 - ▶ Konstruktion wie 2.
 - ▶ zusätzlicher Inverter am Ausgang
4. \overline{Y} enthält nur negierten Variablen (bspw. $\overline{Y} = \overline{A} + \overline{B}$)
 - ▶ Konstruktion wie 1.
 - ▶ zusätzlicher Inverter am Ausgang
5. sonst (bspw. $Y = \overline{A} + B$)
 - ▶ negierte Literale an Gates verwenden
 - ▶ bspw. $Y = \overline{A} + \overline{\overline{B}}$

Wiederholung: Transmissionsgatter



Theoreme der boole'schen Algebra



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Theorem	Duales Theorem	Bedeutung
T1 $A \cdot 1 = A$	T1' $A + 0 = A$	Neutralität
T2 $A \cdot 0 = 0$	T2' $A + 1 = 1$	Extremum
T3 $A \cdot A = A$	T3' $A + A = A$	Idempotenz
T4 $\overline{\overline{A}} = A$		Involution
T5 $A \cdot \overline{A} = 0$	T5' $A + \overline{A} = 1$	Komplement
T6 $A \cdot B = B \cdot A$	T6' $A + B = B + A$	Kommutativität
T7 $A \cdot (B \cdot C) = (A \cdot B) \cdot C$	T7' $A + (B + C) = (A + B) + C$	Assoziativität
T8 $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	T8' $A + (B \cdot C) = (A + B) \cdot (A + C)$	Distributivität
T9 $A \cdot (A + B) = A$	T9' $A + (A \cdot B) = A$	Absorption
T10 $(A \cdot B) + (A \cdot \overline{B}) = A$	T10' $(A + B) \cdot (A + \overline{B}) = A$	Zusammenfassen
T11 $(A \cdot B) + (\overline{A} \cdot C) + (B \cdot C) = (A \cdot B) + (\overline{A} \cdot C)$	T11' $(A + B) \cdot (\overline{A} + C) \cdot (B + C) = (A + B) \cdot (\overline{A} + C)$	Konsensus
T12 $\overline{A \cdot B \cdot C \dots} = \overline{A} + \overline{B} + \overline{C} \dots$	T12' $\overline{A + B + C \dots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \dots$	De Morgan

Beweis für Konsensus (T11) durch Anwendung von Axiomen und Theoremen

$A \cdot B + \bar{A} \cdot C + B \cdot C$	Neutralität
$= A \cdot B + \bar{A} \cdot C + 1 \cdot B \cdot C$	Komplement
$= A \cdot B + \bar{A} \cdot C + (A + \bar{A}) \cdot B \cdot C$	Distributivität
$= A \cdot B + \bar{A} \cdot C + A \cdot B \cdot C + \bar{A} \cdot B \cdot C$	Kommutativität
$= A \cdot B + A \cdot B \cdot C + \bar{A} \cdot C + \bar{A} \cdot C \cdot B$	Neutralität
$= A \cdot B \cdot 1 + A \cdot B \cdot C + \bar{A} \cdot C \cdot 1 + \bar{A} \cdot C \cdot B$	Distributivität
$= A \cdot B \cdot (1 + C) + \bar{A} \cdot C \cdot (1 + B)$	Extremum
$= A \cdot B \cdot 1 + \bar{A} \cdot C \cdot 1$	Neutralität
$= A \cdot B + \bar{A} \cdot C$	q.e.d.

Überblick der heutigen Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Kombinatorische Logik
 - ▶ Bubble Pushing
 - ▶ Logik-Realisierung mit Basis-Gattern
 - ▶ Karnaugh Diagramme



Harris 2013
Kap. 2.4,2.5,2.7,2.8

Bubble Pushing



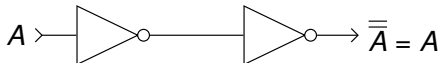
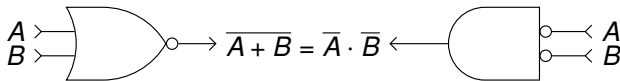
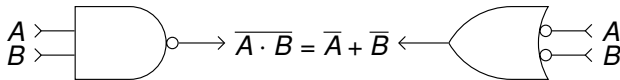
TECHNISCHE
UNIVERSITÄT
DARMSTADT

1001100000010110111110000101001000101110
1010100111010100100000010100011100110111
1101000011010111011010000001101111101111
0100000101101110010000001110100111100000
1000011000100011110000011111011111011001
1101000001000001111100101111010110101111
0001101010111001111000001100000010011111
1100010011100000000000000000001111000000111101
1110001001000011111001100000011100010111
1010100011011001110011101111011110010111
1100011111111010101111010111101000101011
0110101101111010001000100010010110000110
11101111110100100101010110100000100000000
1010100101001011001010111010111000101001
1011100111111110000101000110111001111011
011011111010011101010101010101010001111100

Graphische Umformung von Schaltungen nach De Morgan und Inversion



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Invertierungsblasen verschieben

Bubble Pushing



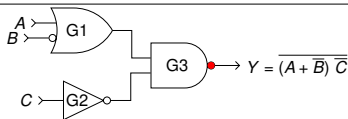
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ über Gatter (AND/OR/NOT/BUF) hinweg
 - ▶ vorwärts: Eingang \rightarrow Ausgang
 - ▶ rückwärts: Ausgang \rightarrow Eingang
 - ▶ Art des Gatters ändern: AND \leftrightarrow OR
 - ▶ Blasen an *allen* Eingängen ändern: vorhanden \leftrightarrow nicht vorhanden
 - ▶ Blase an Ausgang ändert: vorhanden \leftrightarrow nicht vorhanden

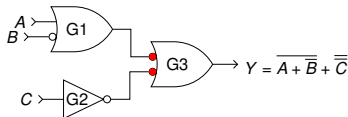
- ▶ zwischen Gattern
 - ▶ vorwärts: Treiber \rightarrow *alle* Empfänger
 - ▶ rückwärts: *alle* Empfänger \rightarrow Treiber
 - ▶ doppelte Blasen heben sich gegenseitig auf (Involution)

- ▶ verbleibende Buffer (vorher Inverter) können entfernt werden

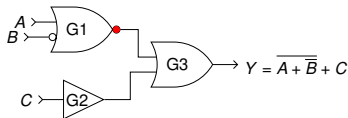
Beispiel: Invertierungsblasen rückwärts verschieben



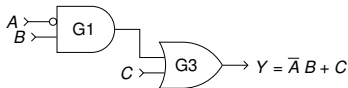
- ▶ De Morgan über G3
 - ▶ Blase am Ausgang \rightarrow Blase an beiden Eingängen
 - ▶ AND \rightarrow OR



- ▶ Blasen entlang Leitungen verschieben
 - ▶ G3 \rightarrow G1
 - ▶ G3 \rightarrow G2 (Doppelblase aufheben)



- ▶ De Morgan über G1
 - ▶ Blasen an Ein- und Ausgängen *invertieren*
 - ▶ OR \rightarrow AND
- ▶ Buffer G2 entfernen



- ▶ zwei Inverter weniger



- ▶ Schaltungen vereinfachen
 - ▶ weniger Invertierer
 - ▶ weniger Literale
 - ▶ weniger verschiedene Gatter-Arten → einfachere Zellbibliothek
- ▶ Komplementäre Schaltungen ableiten
 - ▶ bspw. für CMOS- oder Transmissionsgatter-Schaltung

Logik-Realisierung mit Basis-Gattern



TECHNISCHE
UNIVERSITÄT
DARMSTADT

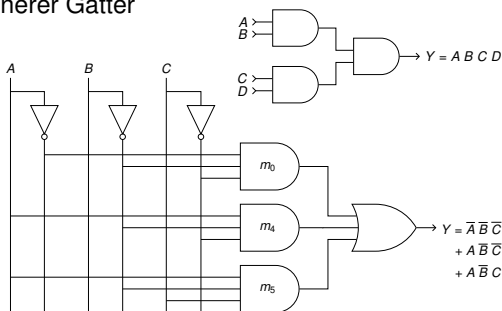
1100000001100000111001101000110001110111
1001001011110010100011111011111001111010
1001110000100001011101110010010100110011
0101100100011110111010100100000111110100
1110000100011010010001111111001100000110
0110001001110001101001111111110000011111
0010010010100110101111100110010110000100
100100101010001000001111100110110000001
0010010110001000100011000000101101010011
1100000100110010100100101000000001100011
0111010001100010101000001000101110101001
0010000001110000001111000010000000101111
0011100001000101000111110101100100010001
1000110000101000100011000010101010100010
1000001000011001011011110110001010101001
10111110110101010101110100011001111111100

- ▶ direkte (konstruktive) Umsetzung der disjunktiven Normalform
 - ▶ Eingangsliterale: ein Inverter pro Variable
 - ▶ Minterme: je ein „breites“ AND Gatter an passende Literale anschließen
 - ▶ Summe: alle Minterme an ein „breites“ OR Gatter anschließen

- ▶ „breite“ Gatter als Kaskaden kleinerer Gatter

⇒ jede boole'sche Funktion realisierbar mit Basisgattern

- ▶ AND2
- ▶ OR2
- ▶ NOT

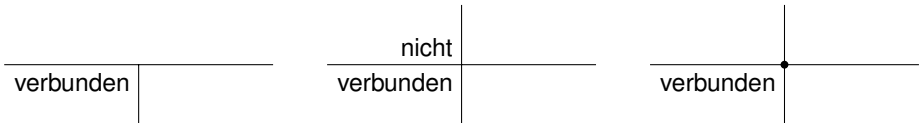


Konventionen für lesbare Schaltpläne



TECHNISCHE
UNIVERSITÄT
DARMSTADT

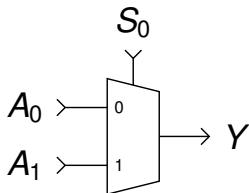
- ▶ Eingänge links (oder oberen)
- ▶ Ausgänge rechts (oder unteren)
- ▶ Gatter von links nach rechts (oben nach unten) angeordnet
- ▶ gerade (oder rechtwinklige) Verbindungen
- ⇒ keine Schrägen oder Kurven
- ▶ 3-armige Kreuzungen gelten implizit als verbunden
- ▶ 4-armige Kreuzungen gelten nur bei Markierung (Punkt) als verbunden



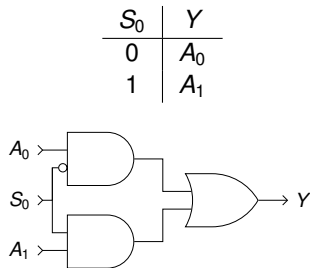


- ▶ zweistufige Logik
 - ▶ sehr mächtig
 - ▶ aufwändige Darstellung und Realisierung
 - ▶ realisiertes Verhalten nicht intuitiv ersichtlich
- ▶ weitere Basisgatter neben AND, OR, NOT:
 - ▶ XOR: Parität
 - ▶ Multiplexer: n zu 1 Auswahl
 - ▶ Dekodierer: n zu 2^n Auswahl („One-Hot“)

- ▶ verbindet einen von n Dateneingängen A_0, \dots, A_{n-1} mit Ausgang Y
- ▶ $k = \lceil \log_2 n \rceil$ Steuersignale S_0, \dots, S_{k-1}
- ▶ $Y = A_{u_{2,k}(S_{k-1} \dots S_0)}$



S_0	A_0	A_1	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



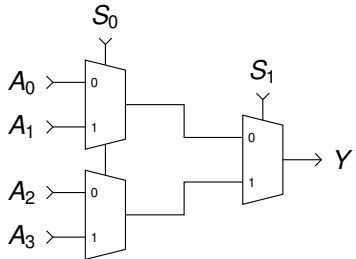
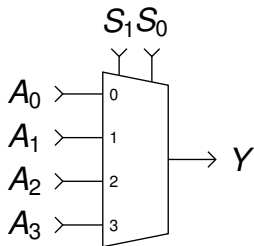
Multiplexer

MUX4 : $\mathbb{B}^6 \rightarrow \mathbb{B}$



TECHNISCHE
UNIVERSITÄT
DARMSTADT

S_1	S_0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3



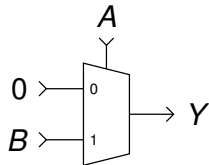
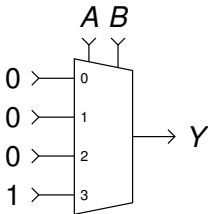
Logikrealisierung mit Multiplexern



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Variablen als Steuersignale verwenden
- ▶ Wahrheitswertetabelle als Konstanten an Dateneingängen
- ▶ entspricht adressiertem Speicherzugriff
 - ▶ Look-up Tabelle
 - ▶ ROM oder RAM → rekonfigurierbare Logik
- ▶ weitere funktionsspezifische Optimierungen möglich

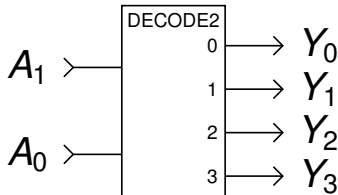
A	B	$Y = A B$
0	0	0
0	1	0
1	0	0
1	1	1





- ▶ n Eingänge A_0, \dots, A_{n-1}
- ▶ 2^n Ausgänge Y_0, \dots, Y_{2^n-1}
- ▶ „One-Hot“ Kodierung: $Y_i = u_{2,n}(A_{n-1} \dots A_0) == i ? 1 : 0$

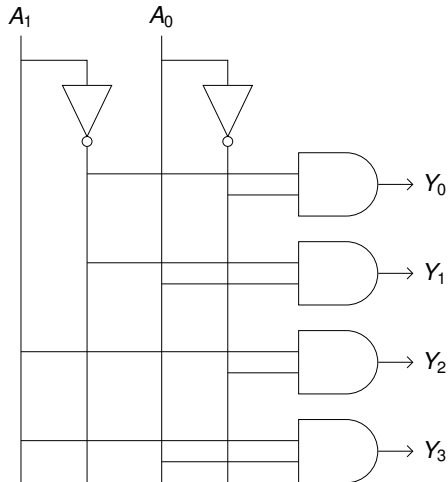
A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Implementierung von Dekodierern



TECHNISCHE
UNIVERSITÄT
DARMSTADT



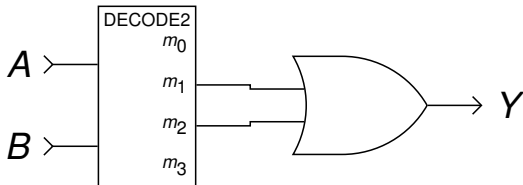
Logikrealisierung mit Decodern



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Summe über Minterme, auf denen Zielfunktion wahr ist
- ⇒ Decoder ersetzt erste Stufe der zweistufigen Logikrealisierung

<i>A</i>	<i>B</i>	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



Karnaugh Diagramme



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1100010111001101010101101110101100001100
0101010011110101101011001011101010011101
0110001110000010011010111110101010101101
1010111111001010000001001101100000100110
0001010110101100001001110000110011000000
1110111100011001010111100111111110100000
1100000110010001011101111100000101000001
1011001111111101100100111111110001010110
0100101011000100111111011000101010100111
1010101011001000010010110001110011000101
0010010000101011001111111010000000111011
0010011001011111100001100110111111100100
1100011100011001100100111010110100110110
1001111110110010011110000110011010011000
0010000100110011100001111010111101000101
1111101001100010010111011100111100010000

Maurice Karnaugh, 1924 -



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Bell Laboratory
- ▶ IBM Research
- ▶ Techniken und Methoden für den schnellen Entwurf informationstechnischer Systeme

⇒ Karnaugh(-Veitch) Diagramme



- ▶ Boole'sche Ausdrücke können durch Zusammenfassen von Mintermen minimiert werden
 - ▶ $Y = A B + A \bar{B} = A$
 - ▶ Karnaugh-Diagramme stellen Zusammenhänge graphisch dar
 - ▶ geschickte Anordnung der Wahrheitswertetabelle
 - ▶ benachbarte Einträge gehören zu gleichem Literal
- ⇒ Zusammenhängende Minterme besser erkennbar

A	B	Y	Minterm
0	0	0	$m_0 = \bar{A} \bar{B}$
0	1	0	$m_1 = \bar{A} B$
1	0	1	$m_2 = A \bar{B}$
1	1	1	$m_3 = A B$

Y:

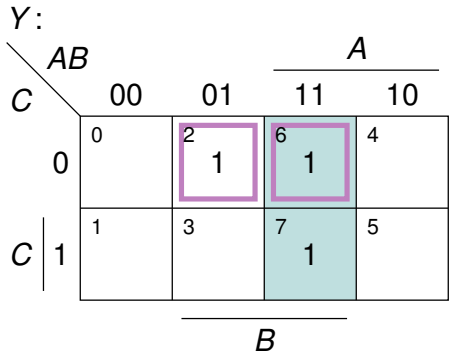
		A	
		0	1
B	0	0	2 1
	1	1	3 1

Karnaugh Diagramm für drei Eingänge



TECHNISCHE
UNIVERSITÄT
DARMSTADT

A	B	C	Y	Minterm
0	0	0	0	$m_0 = \bar{A} \bar{B} \bar{C}$
0	0	1	0	$m_1 = \bar{A} \bar{B} C$
0	1	0	1	$m_2 = \bar{A} B \bar{C}$
0	1	1	0	$m_3 = \bar{A} B C$
1	0	0	0	$m_4 = A \bar{B} \bar{C}$
1	0	1	0	$m_5 = A \bar{B} C$
1	1	0	1	$m_6 = A B \bar{C}$
1	1	1	1	$m_7 = A B C$



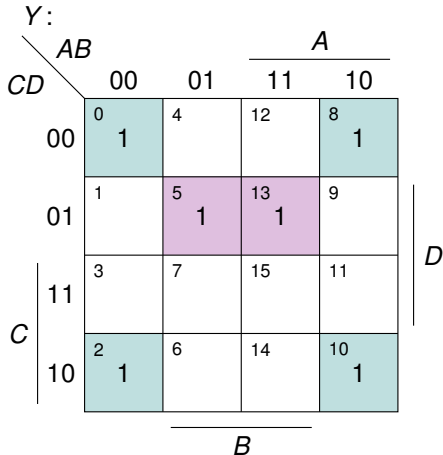
$$Y = AB + B\bar{C}$$

Karnaugh Diagramm für vier Eingänge



TECHNISCHE
UNIVERSITÄT
DARMSTADT

A	B	C	D	Y	Minterm
0	0	0	0	1	$m_0 = \bar{A}\bar{B}\bar{C}\bar{D}$
0	0	0	1	0	$m_1 = \bar{A}\bar{B}\bar{C}D$
0	0	1	0	1	$m_2 = \bar{A}\bar{B}C\bar{D}$
0	0	1	1	0	$m_3 = \bar{A}\bar{B}CD$
0	1	0	0	0	$m_4 = \bar{A}B\bar{C}\bar{D}$
0	1	0	1	1	$m_5 = \bar{A}B\bar{C}D$
0	1	1	0	0	$m_6 = \bar{A}BC\bar{D}$
0	1	1	1	0	$m_7 = \bar{A}BCD$
1	0	0	0	1	$m_8 = A\bar{B}\bar{C}\bar{D}$
1	0	0	1	0	$m_9 = A\bar{B}\bar{C}D$
1	0	1	0	1	$m_{10} = A\bar{B}C\bar{D}$
1	0	1	1	0	$m_{11} = A\bar{B}CD$
1	1	0	0	0	$m_{12} = AB\bar{C}\bar{D}$
1	1	0	1	1	$m_{13} = AB\bar{C}D$
1	1	1	0	0	$m_{14} = ABC\bar{D}$
1	1	1	1	0	$m_{15} = ABCD$



$$Y = \bar{B}\bar{D} + B\bar{C}D$$



- ▶ n Eingangsvariablen
- ▶ Implikant aus $k \leq n$ Literalen deckt 2^{n-k} Minterme ab
- ▶ Primimplikant
 - ▶ nicht vergrößerbare zusammenhängenden viereckigen Fläche im Karnaugh-Diagramm
 - ▶ *Achtung:* muss nicht größte Fläche sein

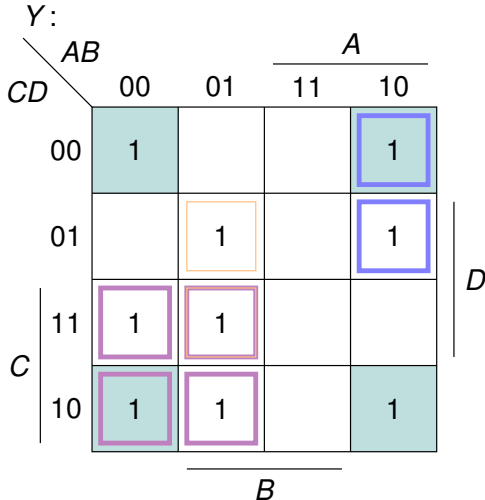


- ▶ Eintragen von Mintermen
 - ▶ Einsen aus Wahrheitstabelle
 - ▶ „Don't Cares“ (*) für ungültige Eingangskombinationen
- ▶ Markieren von Implikanten
 - ▶ markierte Bereiche dürfen 1 und * enthalten, aber keine 0
 - ▶ nur *Rechtecke* mit 2^k Einträgen erlaubt (keine L- oder Z-Formen)
 - ▶ Bereiche dürfen sich überschneiden
 - ▶ Bereiche dürfen um die Ränder des Diagrammes herum reichen (Torus)
 - ▶ Bereiche müssen so groß wie möglich sein (Primimplikanten)
- ▶ Ziel: Überdeckung aller Einsen mit möglichst wenigen Primimplikanten

Karnaugh-Diagramm mit vier Eingängen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

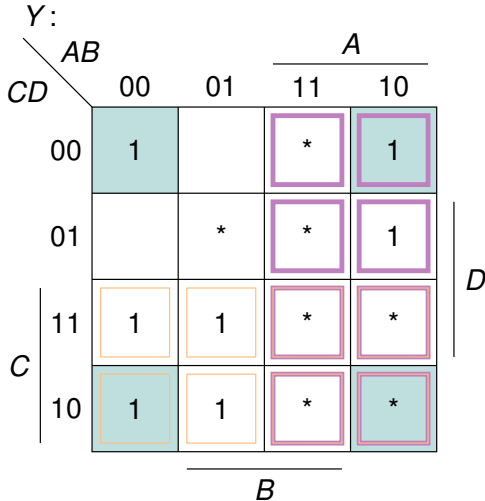


$$\begin{aligned} & \overline{B} \overline{D} \\ + & \overline{A} C \\ + & \overline{A} B D \\ + & A \overline{B} \overline{C} \end{aligned}$$

Karnaugh-Diagramm mit „Don't Cares“



TECHNISCHE
UNIVERSITÄT
DARMSTADT



$$\begin{aligned} & \overline{B} \overline{D} \\ + & A \\ + & C \end{aligned}$$

Zusammenfassung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

0001001000111010011000011100100000001100
1011001001011000010100110100000100011001
0010111011010100101001011000101011100001
0001100001010011101000010000111001110001
0011100010011100110001000011000011001111
1101101010110000100111011010100110000000
0000010101000100100110101000011101011000
0111001101110001110101101001100010100011
1011100111101110100110011110101000111110
0001101111100110010000011011001000101101
1010100010010001110000000010100011100001
1100000001111100001000010000111101111111
1100001111110011001110000011110110001110
1100011010110001110101100010100111100011
1110001101011001100100111100100111010101
0110100100111110100010000010000011000000



- ▶ Bubble Pushing
- ▶ Logik-Realisierung mit Basis-Gattern
- ▶ Karnaugh Diagramme

- ▶ Nächste Vorlesung behandelt
 - ▶ automatisierte Logikminimierung
 - ▶ Zeitverhalten von Schaltungen