

# Technische Grundlagen der Informatik – Kapitel 2



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Prof. Dr.-Ing. Andreas Koch  
Fachgebiet Eingebettete Systeme und ihre Anwendungen (ESA)  
Fachbereich Informatik

WS 12/13



# Kapitel 2: Kombinatorische Logik



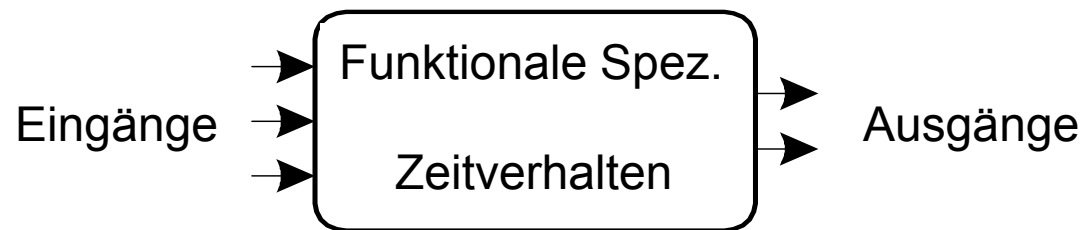
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- **Einleitung**
- **Boole'sche Gleichungen**
- **Boole'sche Algebra**
- **Von Logik zu Gattern**
- **Mehrstufige kombinatorische Logik**
- **X's und Z's**
- **Karnaugh Diagramme**
- **Kombinatorische Grundelemente**
- **Zeitverhalten**

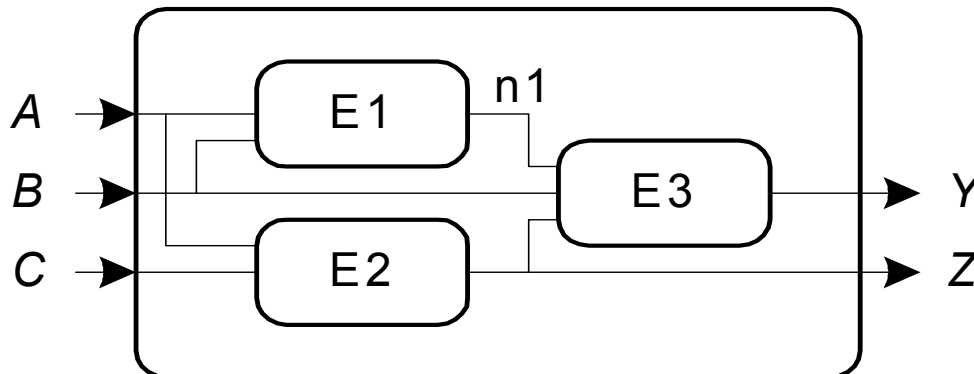
# Einleitung

Eine logische Schaltung ist **zusammengesetzt** aus

- Eingängen
- Ausgängen
- Spezifikation der Funktion
- Spezifikation des Zeitverhaltens



- Verbindungsknoten (*node*)
  - Eingangs-Terminals: *A*, *B*, *C*
  - Ausgangs-Terminals: *Y*, *Z*
  - Interne Knoten: *n1*
- Schaltungselemente
  - *E1*, *E2*, *E3*
  - Jedes wiederum eine **Schaltung** (Hierarchie!)



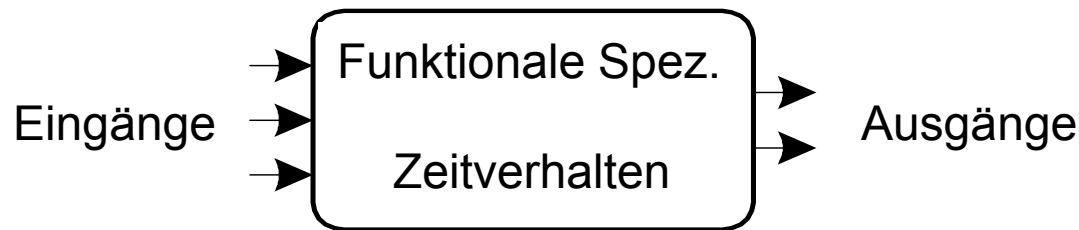
# Arten von logischen Schaltungen

- **Kombinatorische Logik**

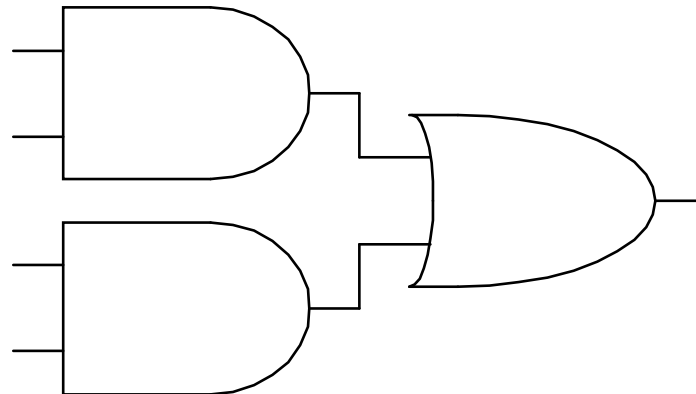
- Zustandslos
- Ausgänge hängen nur von aktuellen Eingangswerten ab

- **Sequentielle Logik**

- Speichert einen Zustand
- Ausgänge hängen ab von aktuellen Eingangswerten und **gespeichertem Zustand**
  - Also damit auch von vorherigen Eingangswerten



- Jedes Schaltungselement ist selbst **kombinatorisch**
- Jeder Verbindungsknoten der Schaltung ist **entweder**
  - ... ein **Eingang** in die Schaltung
  - ... oder an genau **ein** Ausgangsterminal eines Schaltungselements angeschlossen
- Die Schaltung enthält keine Zyklen
  - Jeder Pfad durch die Schaltung besucht jeden Verbindungsknoten maximal einmal
- **Beispiel**



# Boole'sche Gleichungen

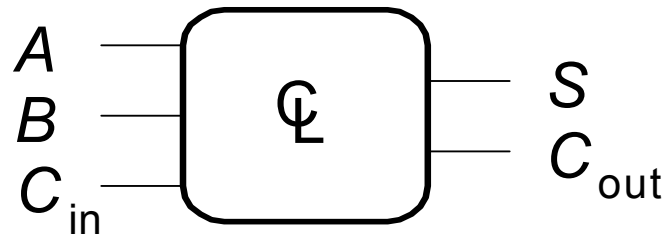


- Beschreiben Ausgänge als **Funktion** der Eingänge

- **Beispiel:**

$$S = F(A, B, C_{in})$$

$$C_{out} = F(A, B, C_{in})$$



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

# Grundlegende Definitionen

- Komplement: Boole'sche Variable mit einem Balken (invertiert)  
 $\bar{A}, \bar{B}, \bar{C}$
- Literal: Variable oder ihr Komplement  
 $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- Implikant: Produkt von Literalen  
 $ABC, A\bar{C}, BC$
- Minterm: Produkt (UND, Konjunktion) über alle Eingangsvariablen  
 $ABC, ABC\bar{C}, \bar{A}BC$
- Maxterm: Summe (ODER, Disjunktion) über alle Eingangsvariablen  
 $(A+\bar{B}+\bar{C}), (A+B+\bar{C}), (\bar{A}+\bar{B}+\bar{C})$



# Disjunktive Normalform (DNF)

- *Sum-of-products (SOP) form*
- Alle Boole'schen Funktionen können in DNF formuliert werden
- Jede **Zeile** der Wahrheitstabelle enthält einen **Minterm**
  - Jeder Minterm ist die **Konjunktion** (Produkt, UND) der Literale
- Der Minterm ist WAHR genau für **diese eine** Zeile
- Die Funktion wird beschrieben durch **Disjunktion** (Summe, ODER) der **Minterme**, die am Ausgang **WAHR** liefern
- Schema: **Summe** aus **Produkten** (SOP)

$A$	$B$	$Y$	minterm
0	0	0	$\overline{A} \overline{B}$
0	1	1	$\overline{A} B$
1	0	0	$A \overline{B}$
1	1	1	$A B$

$Y = F(A, B) =$

# Disjunktive Normalform (DNF)

- *Sum-of-products (SOP) form*
- Alle Boole'schen Funktionen können in DNF formuliert werden
- Jede **Zeile** der Wahrheitstabelle enthält einen **Minterm**
  - Jeder Minterm ist die **Konjunktion** (Produkt, UND) der Literale
- Der Minterm ist WAHR genau für **diese eine** Zeile
- Die Funktion wird beschrieben durch **Disjunktion** (Summe, ODER) der **Minterme**, die am Ausgang **WAHR** liefern
- Schema: **Summe** aus **Produkten** (SOP)

A	B	Y	minterm
0	0	0	$\overline{A} \overline{B}$
0	1	1	$\overline{A} B$
1	0	0	$A \overline{B}$
1	1	1	$A B$

$$Y = F(A, B) =$$

# Disjunktive Normalform (DNF)

- *Sum-of-products (SOP) form*
- Alle Boole'schen Funktionen können in DNF formuliert werden
- Jede **Zeile** der Wahrheitstabelle enthält einen **Minterm**
  - Jeder Minterm ist die **Konjunktion** (Produkt, UND) der Literale
- Der Minterm ist WAHR genau für **diese eine** Zeile
- Die Funktion wird beschrieben durch **Disjunktion** (Summe, ODER) der **Minterme**, die am Ausgang **WAHR** liefern
- Schema: **Summe** aus **Produkten** (SOP)

A	B	Y	minterm
0	0	0	$\bar{A} \bar{B}$
0	1	1	$\bar{A} B$
1	0	0	$A \bar{B}$
1	1	1	$A B$

$$Y = F(A, B) = \bar{A}B + AB$$

# Konjunktive Normalform (KNF)

- *Products-of-sums form (POS)*
- Alle Boole'schen Funktionen können in KNF formuliert werden
- Jede **Zeile** der Wahrheitstabelle enthält einen **Maxterm**
  - Jeder Maxterm ist die Disjunktion (Summe, ODER) von Literalen
- Der Maxterm ist FALSCH genau für **diese eine** Zeile
- Die Funktion wird beschrieben durch **Konjunktion** (Produkt, UND) der Maxterme, die am Ausgang **FALSCH** liefern
- Schema: **Produkt** aus **Summen** (POS)

A	B	Y	maxterm
0	0	0	$A + B$
0	1	1	$A + \bar{B}$
1	0	0	$\bar{A} + B$
1	1	1	$\bar{A} + \bar{B}$

$$Y = F(A, B) = (A + B)(\bar{A} + B)$$

# Beispiel für Boole'sche Funktion

- Sie prüfen das Mittagsangebot der Mensa
  - Sie werden dort **nicht** essen gehen ( $\overline{E}$ )
  - Wenn nicht mehr geöffnet ist ( $\overline{O}$ ) **oder**
  - Es nur Corned Beef-Variationen gibt ( $C$ )
- Stellen Sie eine Wahrheitstabelle auf, ob Sie in die Mensa gehen

$O$	$C$	$E$
0	0	
0	1	
1	0	
1	1	

# Beispiel für Boole'sche Funktion

- Sie prüfen das Mittagsangebot der Mensa
  - Sie werden dort **nicht** essen gehen ( $\overline{E}$ )
  - Wenn nicht mehr geöffnet ist ( $\overline{O}$ ) **oder**
  - Es nur Corned Beef-Variationen gibt ( $C$ )
- Stellen Sie eine **Wahrheitstabelle** auf, ob Sie in die Mensa gehen

$O$	$C$	$E$
0	0	0
0	1	0
1	0	1
1	1	0

# DNF (*SOP*) und KNF (*POS*) Formen

- DNF – Disjunktive Normalform (*sum-of-products, SOP*)

$O$	$C$	$E$	minterm
0	0	0	$\overline{O} \overline{C}$
0	1	0	$\overline{O} C$
1	0	1	$O \overline{C}$
1	1	0	$O C$

- KNF – Konjunktive Normalform (*product-of-sums, POS*)

$O$	$C$	$Y$	maxterm
0	0	0	$O + C$
0	1	0	$O + \overline{C}$
1	0	1	$\overline{O} + C$
1	1	0	$\overline{O} + \overline{C}$

# DNF (SOP) und KNF (POS) Formen

- DNF – Disjunktive Normalform (*sum-of-products, SOP*)

$O$	$C$	$E$	minterm
0	0	0	$\overline{O} \overline{C}$
0	1	0	$\overline{O} C$
1	0	1	$O \overline{C}$
1	1	0	$O C$

$$E = O\overline{C}$$

- KNF – Konjunktive Normalform (*product-of-sums, POS*)

$O$	$C$	$E$	maxterm
0	0	0	$O + C$
0	1	0	$O + \overline{C}$
1	0	1	$\overline{O} + C$
1	1	0	$\overline{O} + \overline{C}$

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$



# Boole'sche Algebra



- Axiome und Sätze, hier zum Ziel der Vereinfachung boole'scher Gleichungen
- Wie die übliche Algebra
  - Teilweise einfacher, da hier nur zwei Werte
- Axiome und Sätze haben jeweils duale Entsprechung:
  - Tausche AND/OR, tausche 0/1

# Axiome und Sätze der Boole'schen Algebra



	Axiom		Dual		Name
A1	$B = 0 \text{ if } B \neq 1$	A1'	$B = 1 \text{ if } B \neq 0$		<b>Dualitätsgesetz</b>
A2	$\bar{0} = 1$	A2'	$\bar{1} = 0$		NOT
A3	$0 \bullet 0 = 0$	A3'	$1 + 1 = 1$		AND/OR
A4	$1 \bullet 1 = 1$	A4'	$0 + 0 = 0$		AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5'	$1 + 0 = 0 + 1 = 1$		AND/OR

	Satz		Dual		Name
T1	$B \bullet 1 = B$	T1'	$B + 0 = B$		<b>Neutralitätsgesetz</b>
T2	$B \bullet 0 = 0$	T2'	$B + 1 = 1$		<b>Extremalgesetz</b>
T3	$B \bullet B = B$	T3'	$B + B = B$		<b>Idempotenzgesetz</b>
T4		$\overline{\overline{B}} = B$			<b>Involution</b>
T5	$B \bullet \bar{B} = 0$	T5'	$B + \bar{B} = 1$		<b>Komplementärgesetz</b>

# T1: Neutralitätsgesetz

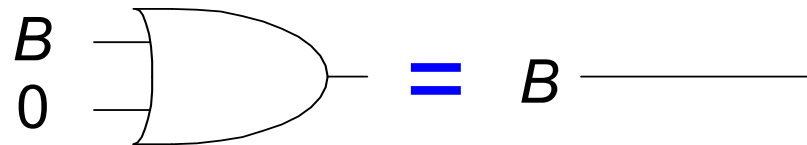
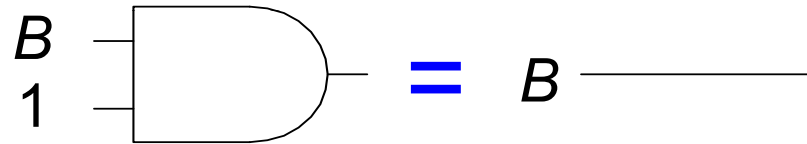


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- $B \bullet 1 =$
- $B + 0 =$

# T1: Neutralitätsgesetz

- $B \bullet 1 = B$
- $B + 0 = B$



# T2: Extremalgesetz

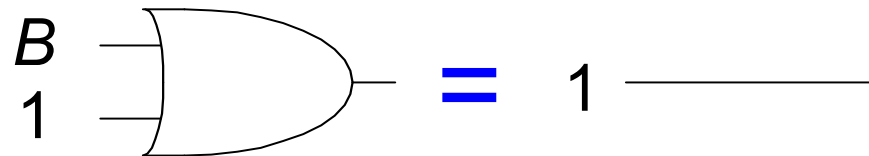
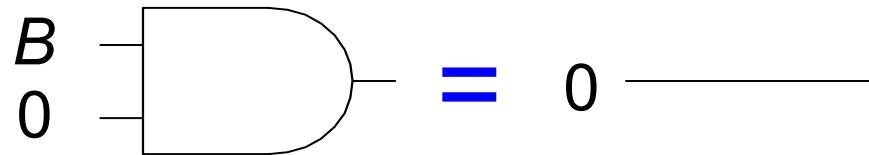


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- $B \bullet 0 =$
- $B + 1 =$

## T2: Extremalgesetz

- $B \bullet 0 = 0$
- $B + 1 = 1$

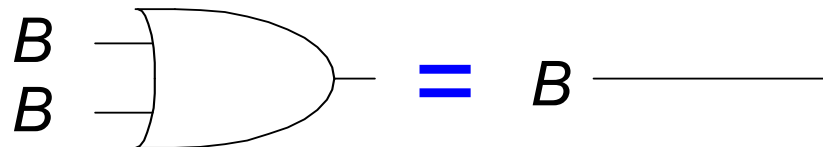
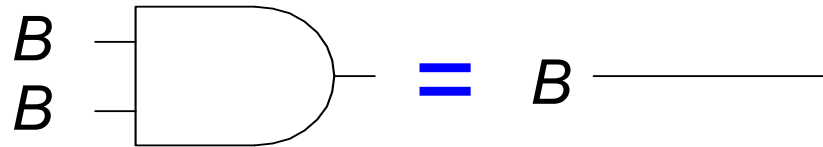


# T3: Idempotenzgesetz

- $B \bullet B =$
- $B + B =$

# T3: Idempotenzgesetz

- $B \bullet B = B$
- $B + B = B$





# T4: Involution (Selbstinversion)

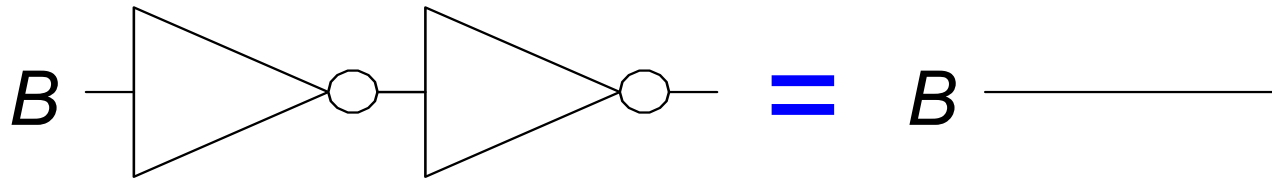


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- $\overline{\overline{B}} =$

# T4: Involution (Selbstinversion)

- $\overline{\overline{B}} = B$



# T5: Komplementärgesetz

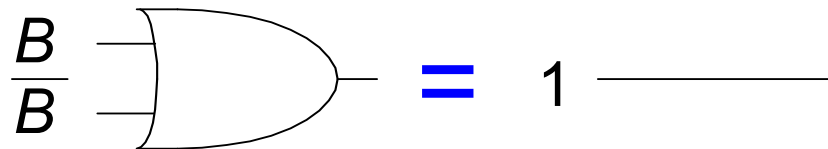
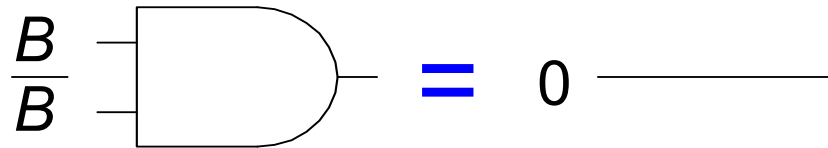


- $B \bullet \overline{B} =$
- $B + \overline{B} =$

# T5: Komplementärgesetz



- $B \bullet \overline{B} = 0$
- $B + \overline{B} = 1$



# Sätze der Boole'schen Algebra mit einer Variablen

	Satz		Dual	Name
T1	$B \cdot 1 = B$	T1'	$B + 0 = B$	Neutralitätsgesetz
T2	$B \cdot 0 = 0$	T2'	$B + 1 = 1$	Extremalgesetz
T3	$B \cdot B = B$	T3'	$B + B = B$	Idempotenzgesetz
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \cdot \overline{B} = 0$	T5'	$B + \overline{B} = 1$	Komplementärgesetz

# Sätze der Boole'schen Algebra mit mehreren Variablen

	Satz		Dual	Name
T6	$B \cdot C = C \cdot B$	T6'	$B + C = C + B$	Kommutativgesetz
T7	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	T7'	$(B + C) + D = B + (C + D)$	Assoziativgesetz
T8	$(B \cdot C) + B \cdot D = B \cdot (C + D)$	T8'	$(B + C) \cdot (B + D) = B + (C \cdot D)$	Distributivgesetz
T9	$B \cdot (B + C) = B$	T9'	$B + (B \cdot C) = B$	Absorptionsgesetz
T10	$(B \cdot C) + (B \cdot \overline{C}) = B$	T10'	$(B + C) \cdot (B + \overline{C}) = B$	Zusammenfassen
T11	$(B \cdot C) + (\overline{B} \cdot D) + (C \cdot D)$ $= B \cdot C + \overline{B} \cdot D$	T11'	$(B + C) \cdot (\overline{B} + D) \cdot (C + D)$ $= (B + C) \cdot (\overline{B} + D)$	Konsensusregeln
T12	$\overline{B_0 \cdot B_1 \cdot B_2 \dots}$ $= (\overline{B_0} + \overline{B_1} + \overline{B_2} \dots)$	T12'	$\overline{B_0 + B_1 + B_2 \dots}$ $= (\overline{B_0} \cdot \overline{B_1} \cdot \overline{B_2} \dots)$	De Morgansche Gesetze

# Beispiel 1: Vereinfachen von Boole'schen Ausdrücken



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- $Y = AB + \overline{A}B$

# Beispiel 1: Vereinfachen von Boole'schen Ausdrücken



■  $Y = AB + \overline{A}B$

$$= B(A + \overline{A})$$

T8      Distributivgesetz

$$= B(1)$$

T5'     Komplementärgesetz

$$= B$$

T1      Identitätsgesetz



## Beispiel 2: Vereinfachen von Boole'schen Ausdrücken



- $Y = A(AB + ABC)$

## Beispiel 2: Vereinfachen von Boole'schen Ausdrücken



$$\begin{aligned} \blacksquare Y &= A(AB + ABC) \\ &= A(AB(1 + C)) \\ &= A(AB(1)) \\ &= A(AB) \\ &= (AA)B \\ &= AB \end{aligned}$$

T8 Distributivgesetz

T2' Extremalgesetz

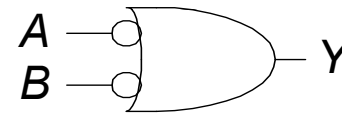
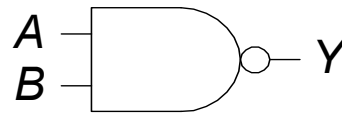
T1 Identitätsgesetz

T7 Assoziativgesetz

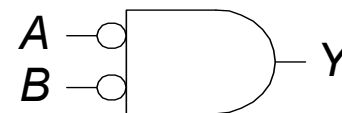
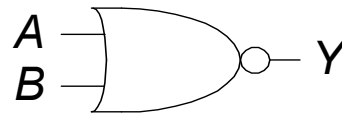
T3 Idempotenzgesetz

# De Morgan'sche Gesetze

- $Y = \overline{AB} = \overline{A} + \overline{B}$



- $Y = \overline{A + B} = \overline{A} \bullet \overline{B}$



# Invertierungsblasen verschieben (*bubble pushing*)

- Verschiebe Blasen **rückwärts** (vom Ausgang) oder **vorwärts** (vom Eingang)
- Art des Gatters ändert sich von **AND nach OR** (oder **umgekehrt**)
- Beim Verschieben rückwärts entstehen Blasen an **allen** Eingängen

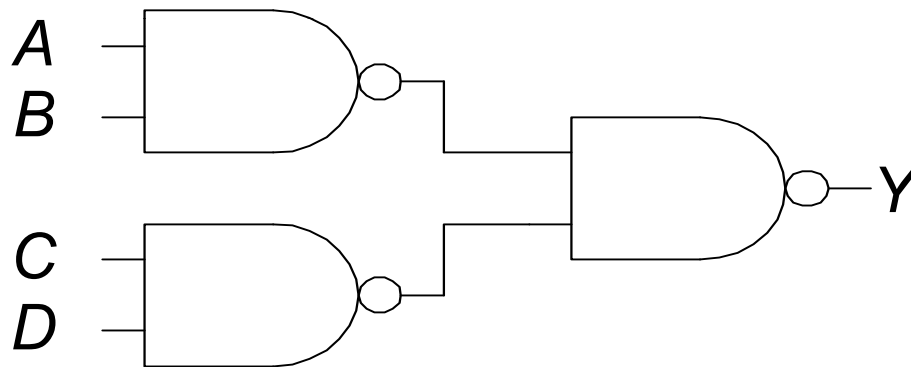


- Beim Verschieben vorwärts müssen Blasen an **allen** Eingängen gewesen sein



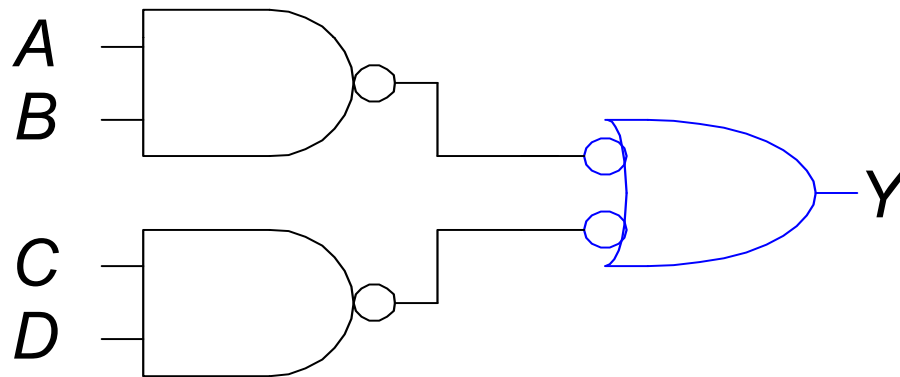
# Invertierungsblasen verschieben

- Was ist die boole'sche Funktion dieser Schaltung?



# Invertierungsblasen verschieben

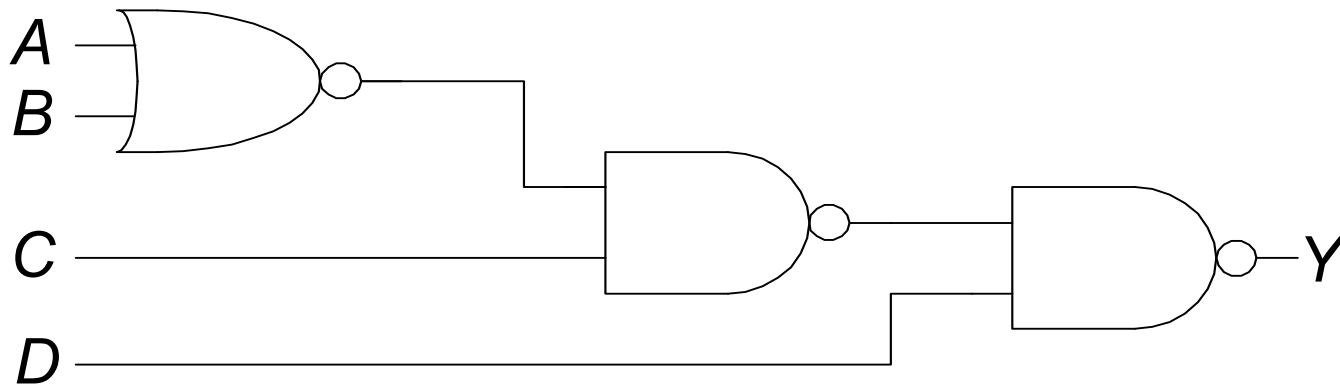
- Was ist die boole'sche Funktion dieser Schaltung?



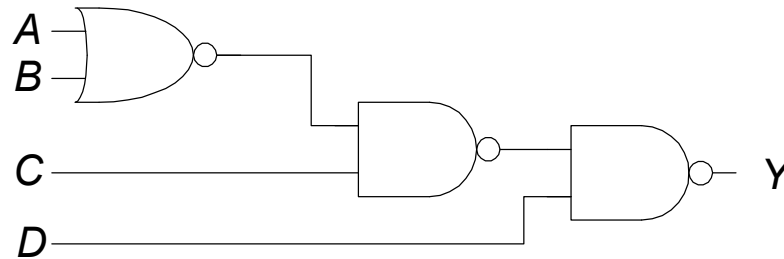
$$Y = AB + CD$$

# Regeln für das Verschieben von Invertierungsblasen

- Beginne am **Ausgang**, vorarbeiten Richtung **Eingänge**
- Schiebe Blasen am **Ausgang** Richtung **Eingang**
- Tausche **Art** des Gatters aus (AND/OR)
- Versuche Blasen **auszulöschen** (zwei Blasen auf einer Leitung)
  - Wenn **Eingang** Blase hat, versuche **Ausgang** mit Blase zu versehen
  - ... und umgekehrt

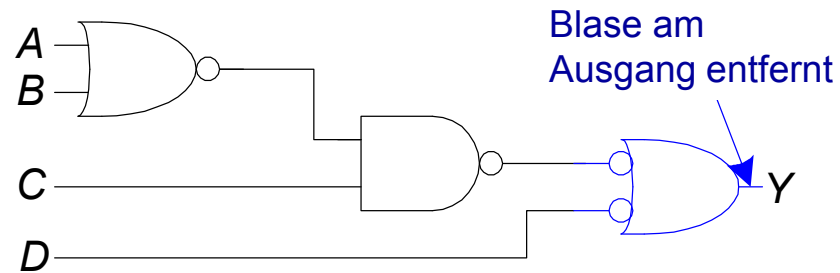


# Beispiel: Invertierungsblasen verschieben

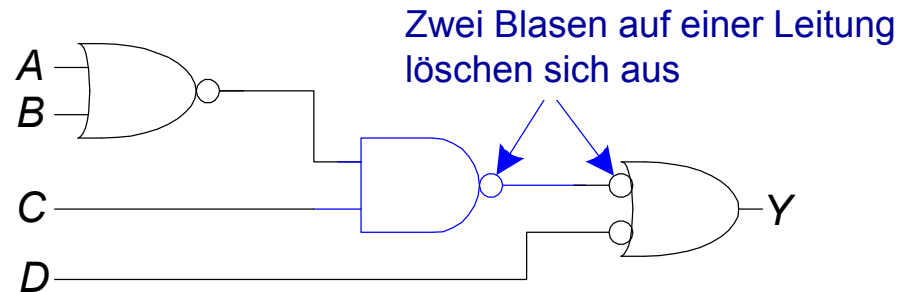
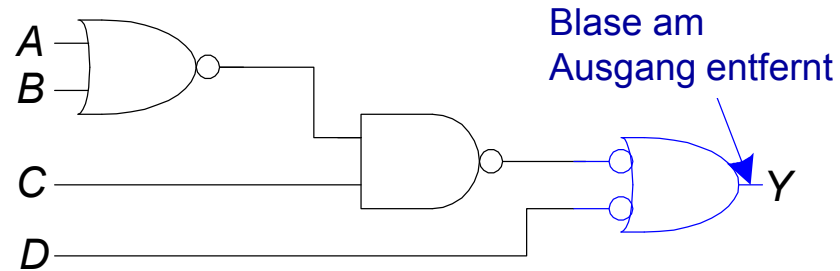




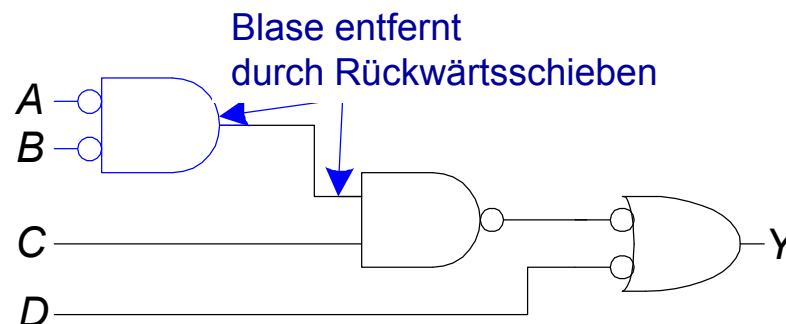
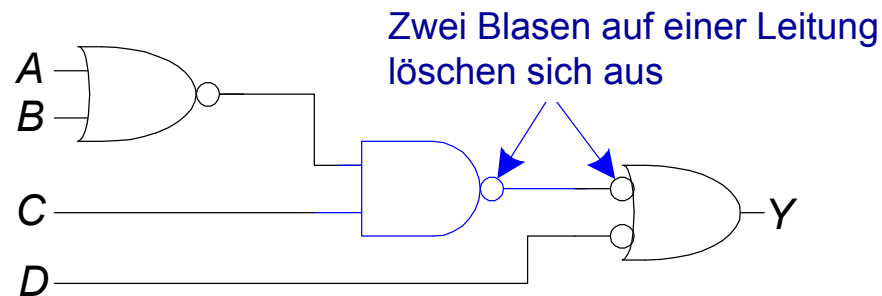
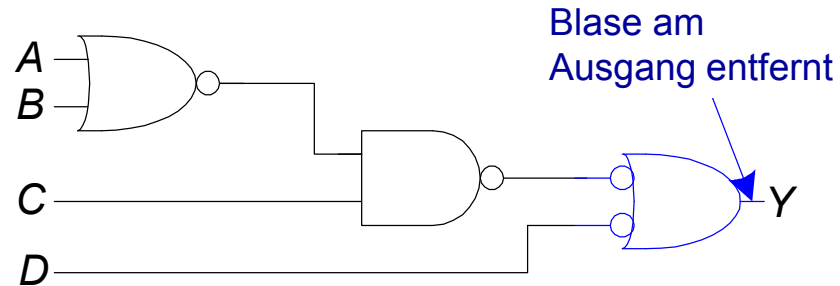
# Beispiel: Invertierungsblasen verschieben



# Beispiel: Invertierungsblasen verschieben



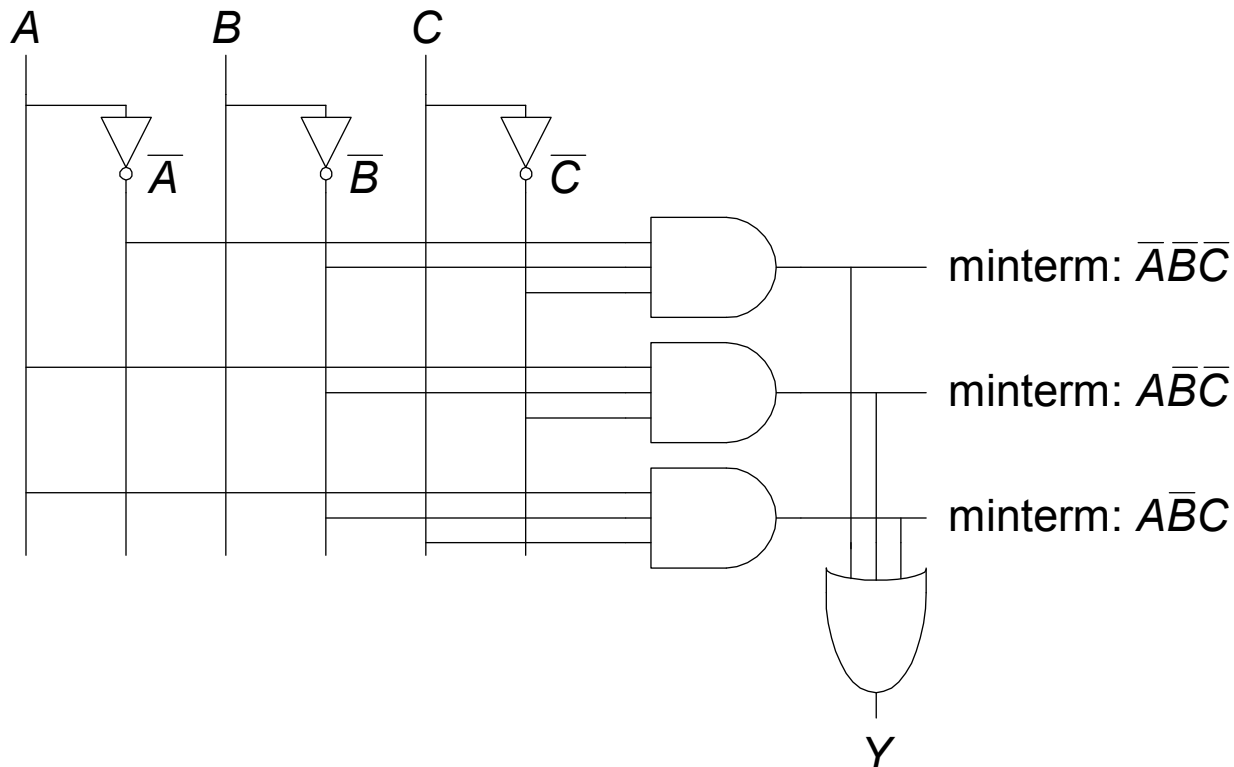
# Beispiel: Invertierungsblasen verschieben



$$Y = \overline{A}BC + \overline{D}$$

# Von Logik zu Gattern

- Zweistufige Logik: ANDs gefolgt von ORs
- Beispiel:  $Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$



# Lesbare Schaltpläne



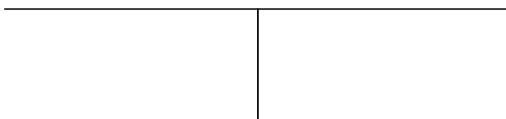
- Eingänge sind auf der **linken** (oder oberen) Seite des Schaltplans
- Ausgänge sind auf der **rechten** (oder unteren) Seite des Schaltplans
- Gatter sollten von **links nach rechts** angeordnet werden
  - In seltenen Fällen: Von oben nach unten
- **Gerade** Verbindungen sind leichter lesbar als abknickende
  - Gegebenenfalls gerade lange Verbindung statt kurzer abgeknickter

# Regeln für Schaltpläne

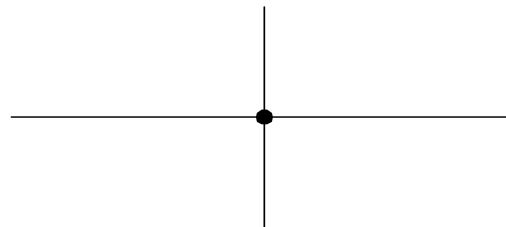


- Drähte an T-Kreuzung sind **verbunden**
- Sich **überkreuzende** Drähte werden durch **Punkt** als verbunden markiert
- Sich **überkreuzende** Drähte ohne Punkt sind **nicht** verbunden

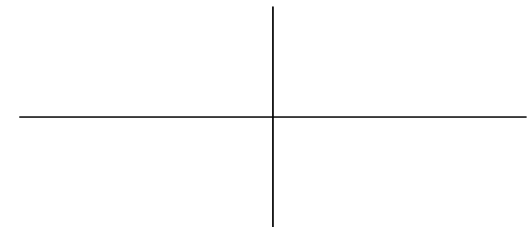
T-Kreuzung:  
**verbunden**



Überkreuzend:  
**verbunden**



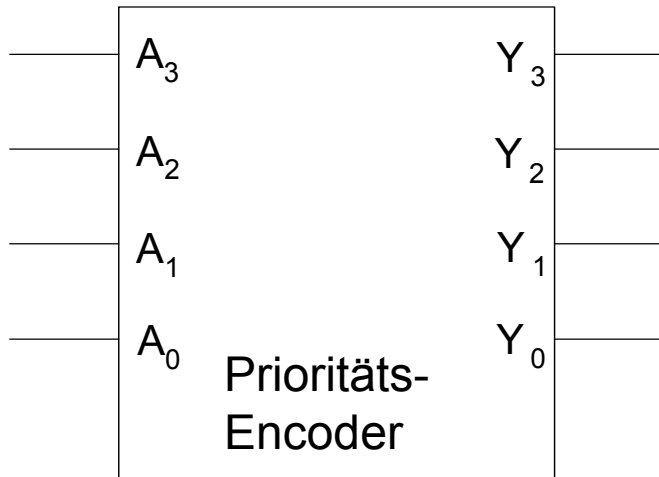
Überkreuzend:  
**Nicht** verbunden



# Schaltungen mit mehreren Ausgängen



- Ausgang entsprechend dem **höchstwertigen** gesetzten Eingangsbit wird auf TRUE gesetzt

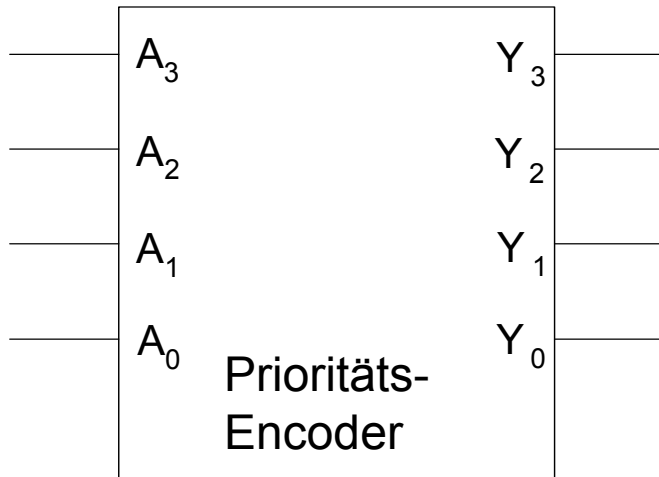


$A_3$	$A_2$	$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

# Schaltungen mit mehreren Ausgängen



- Ausgang entsprechend dem **höchstwertigen** gesetzten Eingangsbit wird auf TRUE gesetzt

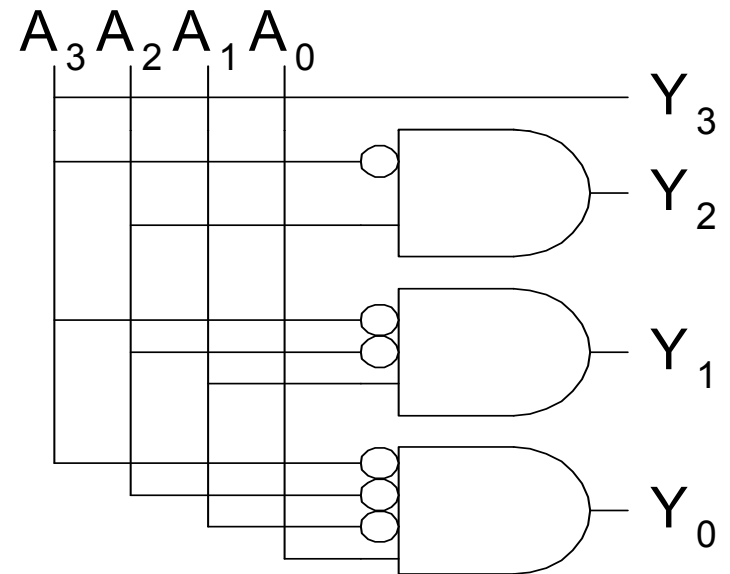


$A_3$	$A_2$	$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0



# Aufbau des Prioritäts-Encoders

$A_3$	$A_2$	$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0



# Ignorierbare Bits (“Don’t Cares”)

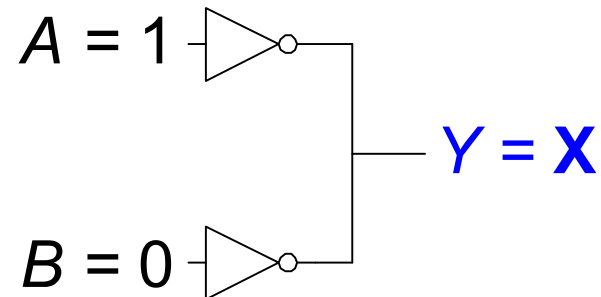
$A_3$	$A_2$	$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

$A_3$	$A_2$	$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

# Konkurrierende Treiber: X



- Konflikt: Schaltung treibt eine Leitung/Ausgang **gleichzeitig** auf 0 und 1
  - Analogwert liegt irgendwo dazwischen (**Spannungsteilung**)
  - Kann 0 oder 1 sein, oder im verbotenen Bereich liegen
  - Kann auch mit Betriebsspannung, Temperatur, Rauschen etc. **variieren**
  - Verursacht hohen **Energieverbrauch** (Kurzschluss)

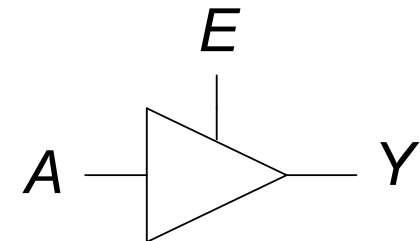


- Treiberkonflikt ist fast immer ein Entwurfsfehler
  - Beheben!
- **Vorsicht: X steht für “don’t care” und Treiberkonflikt**
  - Nicht das gleiche!
  - Kontext anschauen, um korrekte Bedeutung zu ermitteln

# Hochohmiger Ausgang: Z

- Auch genannt:
  - Offen, ungetrieben
  - *Floating, open, high-impedance*
- Kann 0 oder 1 sein, oder irgendwo dazwischen liegen
  - Leitung hat keinen aktiven Treiber

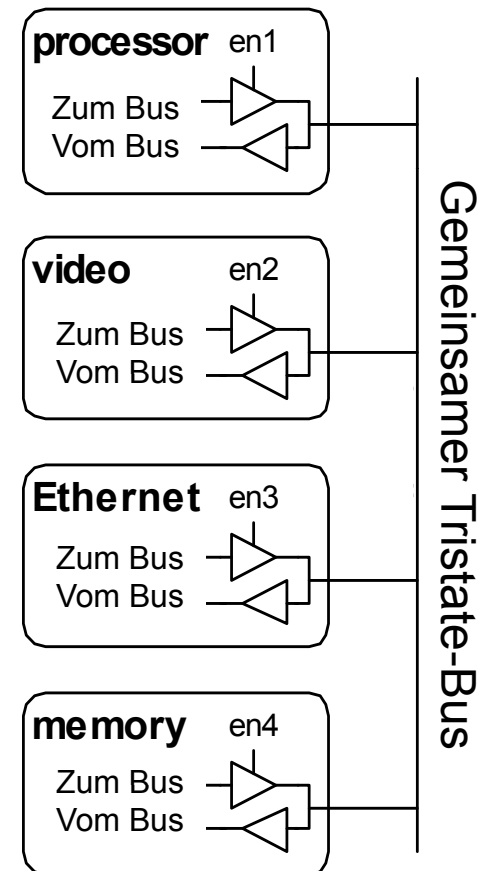
## Tristate Buffer



$E$	$A$	$Y$
0	0	Z
0	1	Z
1	0	0
1	1	1

# Tristate-Busse

- Hochohmige Knoten können zu Tristate-Bussen verschaltet werden
  - Viele **verschiedene** Treiber
  - Aber zu jedem Zeitpunkt ist **genau** einer aktiv
  - Der Rest ist **hochohmig** (Z)



# Karnaugh Diagramme (*Karnaugh maps*)

- Boole'sche Ausdrücke können durch Zusammenfassen **minimiert** werden
- Karnaugh-Diagramme stellen Zusammenhänge **graphisch** dar
  - Bilden **Ausgangspunkt** für eine Minimierung
- Idee:  $PA + \overline{PA} = P$

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

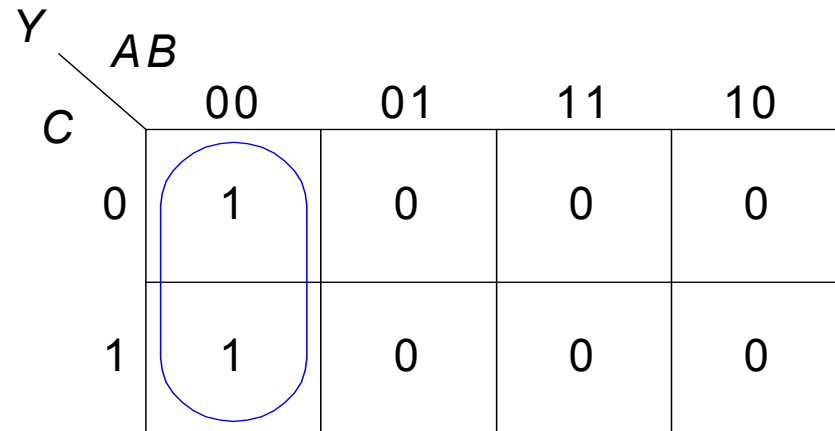
Y		AB			
		00	01	11	10
C	0	1	0	0	0
	1	1	0	0	0

Y		AB			
		00	01	11	10
C	0	$\overline{A}\overline{B}\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\overline{C}$
	1	$\overline{A}\overline{B}C$	$\overline{A}BC$	$ABC$	$A\overline{B}C$

# Minimierung mit Karnaugh Diagrammen

- Markiere 1en in **benachbarten** Plätzen und bilde **viereckigen** Bereich
  - Jeder Platz steht für einen Implikanten
- Lasse markierte Literale
  - ... die normal **und** als Komplement auftauchen, **weg**

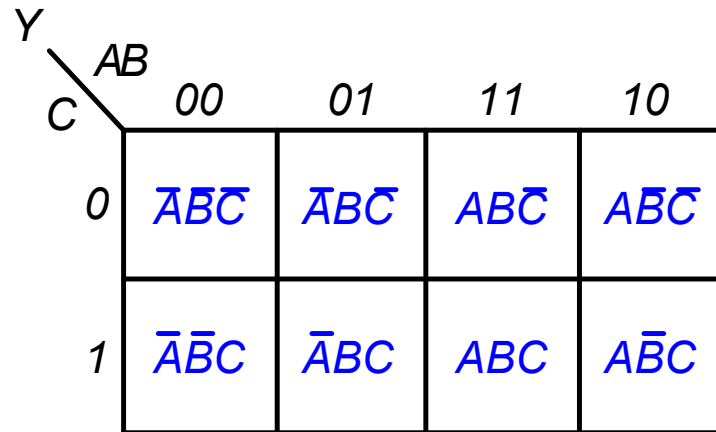
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



Y \ C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

$$Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C = \overline{A}\overline{B}$$

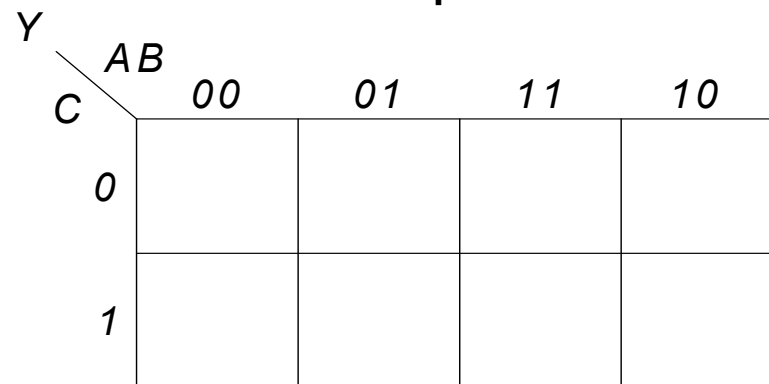
# Karnaugh Diagramm mit drei Eingängen



Truth Table

A	B	C	Y
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

K-Map





# Karnaugh Diagramm mit drei Eingängen



		Y AB			
		00	01	11	10
C	0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$A\bar{B}\bar{C}$	$AB\bar{C}$
	1	$\bar{A}\bar{B}C$	$\bar{A}BC$	$ABC$	$A\bar{B}C$

Truth Table

A	B	C	Y
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

K-Map

		Y AB			
		00	01	11	10
C	0	0	1	1	0
	1	0	1	0	0

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



# Karnaugh Diagramme: Definitionen

- Komplement: Variable mit Balken (invertierter Wert)

$\bar{A}, \bar{B}, \bar{C}$

- Literal: Variable oder ihr Komplement

$A, \bar{A}, B, \bar{B}, C, \bar{C}$

- Implikant: Produkt (UND) von Literalen

$ABC, A\bar{C}, \bar{B}C$

- **Primimplikant**

- Implikant der **größten zusammenhängenden viereckigen** Fläche im Karnaugh-Diagramm

# Minimierungsregeln für Karnaugh-Diagramme



- Jede 1 in einem K-Diagramm muss **mindestens** einmal markiert werden
  - Ist damit **Bestandteil** eines oder mehrerer viereckiger Bereiche
- Jeder viereckige Bereich hat als **Seitenlänge** eine Zweierpotenz an Flächen
  - 1,2,4,8,16,... Flächen Seitenlänge
  - Beide Seiten dürfen aber **unterschiedlich** lang sein
- Jeder Bereich muss so **groß** wie möglich sein (Primimplikant)
- Ein Bereich darf um die **Ränder** des K-Diagrammes herum reichen
- Ein “don't care” (X) **darf** markiert werden, wenn es die Fläche **größer** macht
- Ziel: Möglichst **wenige** Primimplikanten zur **Abdeckung** aller 1en

# Karnaugh-Diagramm mit vier Eingängen



A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
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	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

		Y			
		AB			
CD		00	01	11	10
	00				
	01				
	11				
	10				

# Karnaugh-Diagramm mit vier Eingängen

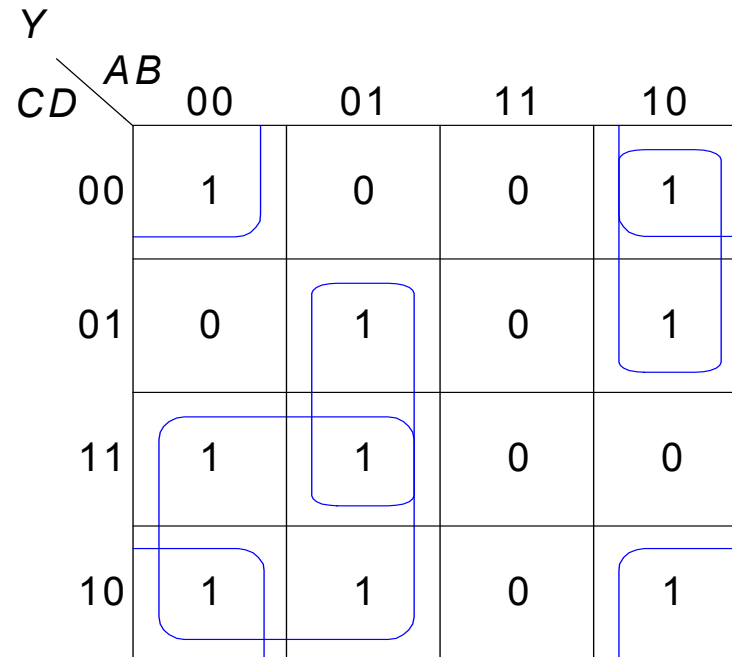


A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y		AB			
		00	01	11	10
CD	00	1	0	0	1
	01	0	1	0	1
	11	1	1	0	0
	10	1	1	0	1

# Karnaugh-Diagramm mit vier Eingängen

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
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	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



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

# Karnaugh-Diagramm mit “don't cares”



A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y		AB			
		00	01	11	10
CD	00				
	01				
	11				
	10				

# Karnaugh-Diagramm mit “don't cares”



A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

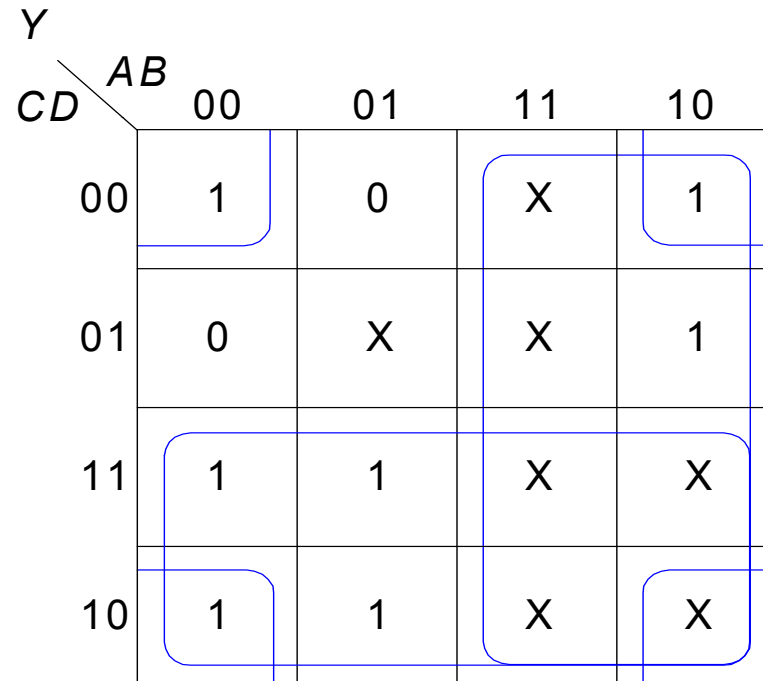
Y CD \ AB		AB			
		00	01	11	10
00	00	1	0	X	1
	01	0	X	X	1
11	11	1	1	X	X
	10	1	1	X	X



# Karnaugh-Diagramm mit “don't cares”



A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



$$Y = A + \overline{B}\overline{D} + C$$

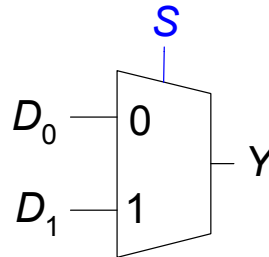
# Kombinatorische Grundelemente

- Multiplexer
- Dekodierer (*Decoders*)

# Multiplexer (Mux)



- Wählt einen von  $N$  Eingängen aus und verbindet ihn auf den Ausgang
- $\log_2 N$ -bit Selektor-Eingang (*select input*), Steuereingang
- Beispiel: **2:1 Mux**

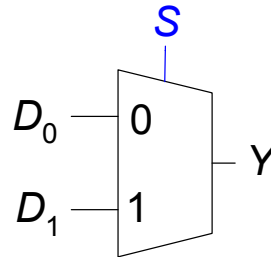


S	D <sub>1</sub>	D <sub>0</sub>	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

# Multiplexer (Mux)



- Wählt einen von  $N$  Eingängen aus und verbindet ihn auf den Ausgang
- $\log_2 N$ -bit Selektor-Eingang (*select input*), Steuereingang
- Beispiel: **2:1 Mux**



S	$D_1$	$D_0$	Y	S	Y
0	0	0	0	0	$D_0$
0	0	1	1	1	$D_1$
0	1	0	0		
0	1	1	1		
1	0	0	0		
1	0	1	0		
1	1	0	1		
1	1	1	1		

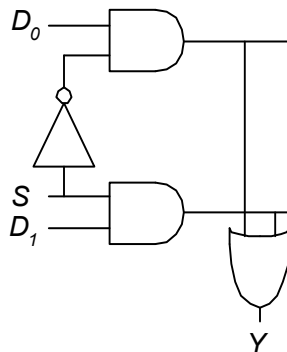
# Implementierung von Multiplexern

- Aus Logikgattern

- Disjunktive Normal Form (SOP)

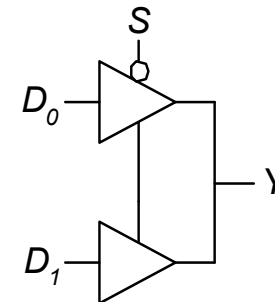
Y	$D_0 D_1$			
	$00$	$01$	$11$	$10$
S				
0	0	0	1	1
1	0	1	1	0

$$Y = D_0 \bar{S} + D_1 S$$



- Aus Tristate-Buffern

- Benutze  $N$  Tristates für  $N$ -Eingangs-Mux
- Schalte zu jeder Zeit genau einen Tristate-Buffer durch, Rest ist Z



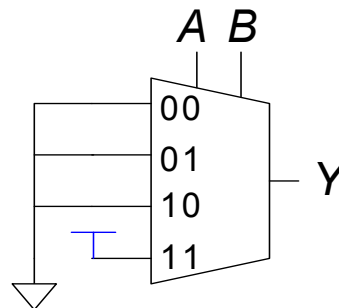
# Logikfunktionen aufgebaut aus Multiplexern



- Verwende Mux als Wertetabelle (*look-up table*)

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Y = AB$$



# Logikfunktionen aufgebaut aus Multiplexern

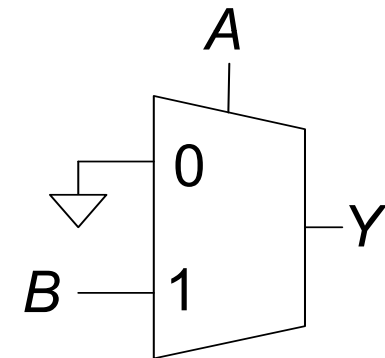
- Reduziere Größe des Multiplexers

$$Y = AB$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

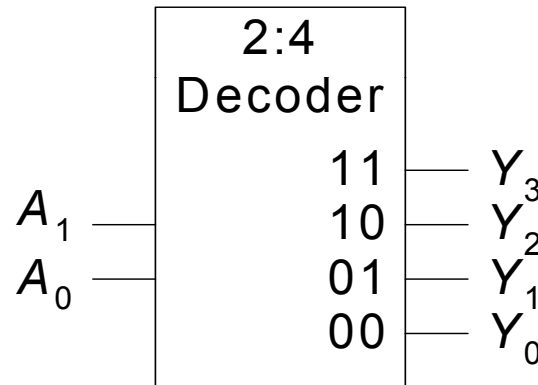
  

A	Y
0	0
1	B



# Dekodierer (*Decoder*)

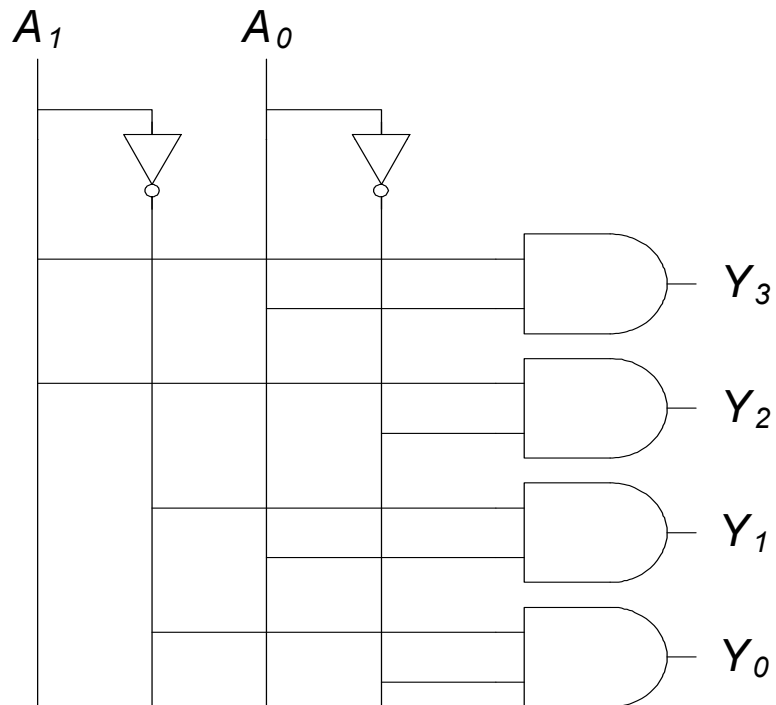
- $N$  Eingänge,  $2^N$  Ausgänge
- Ausgänge sind “one-hot”: Zu jedem Zeitpunkt ist **genau ein** Ausgang 1



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

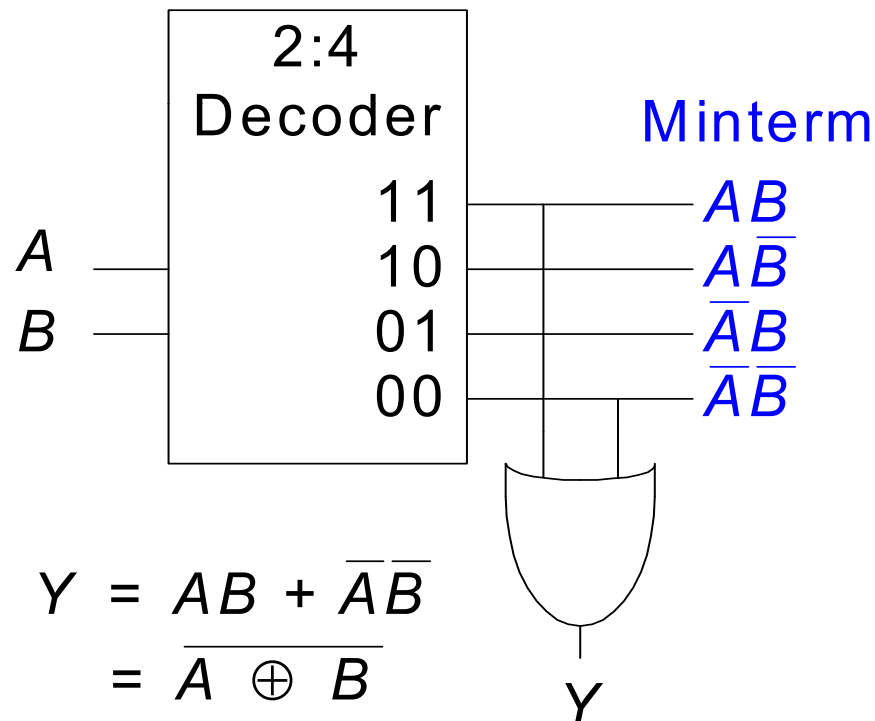


# Implementierung von Dekodierern



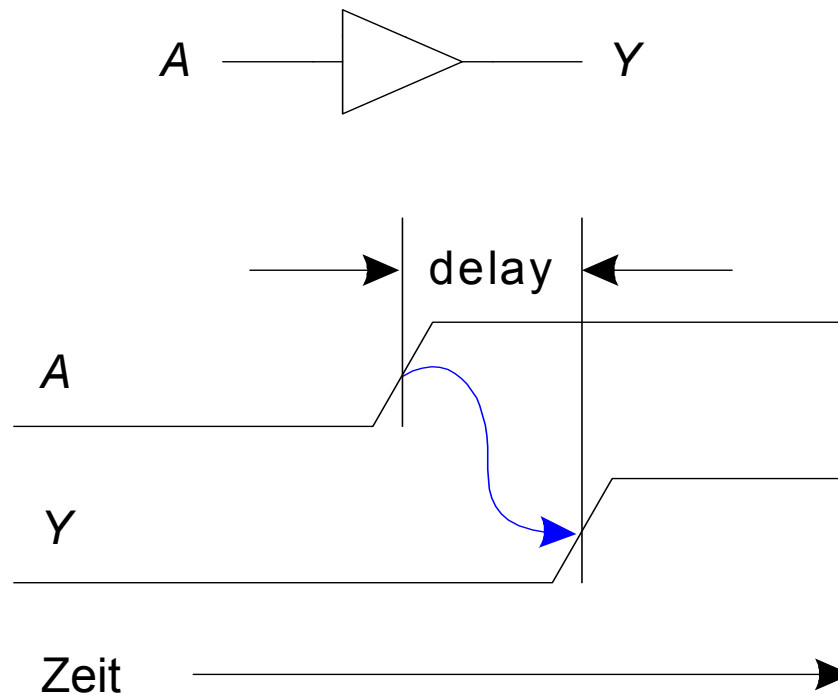
# Logik aufgebaut aus Dekodierern

- Verknüpfe **Minterme** mit ODER



# Zeitverhalten (*Timing*)

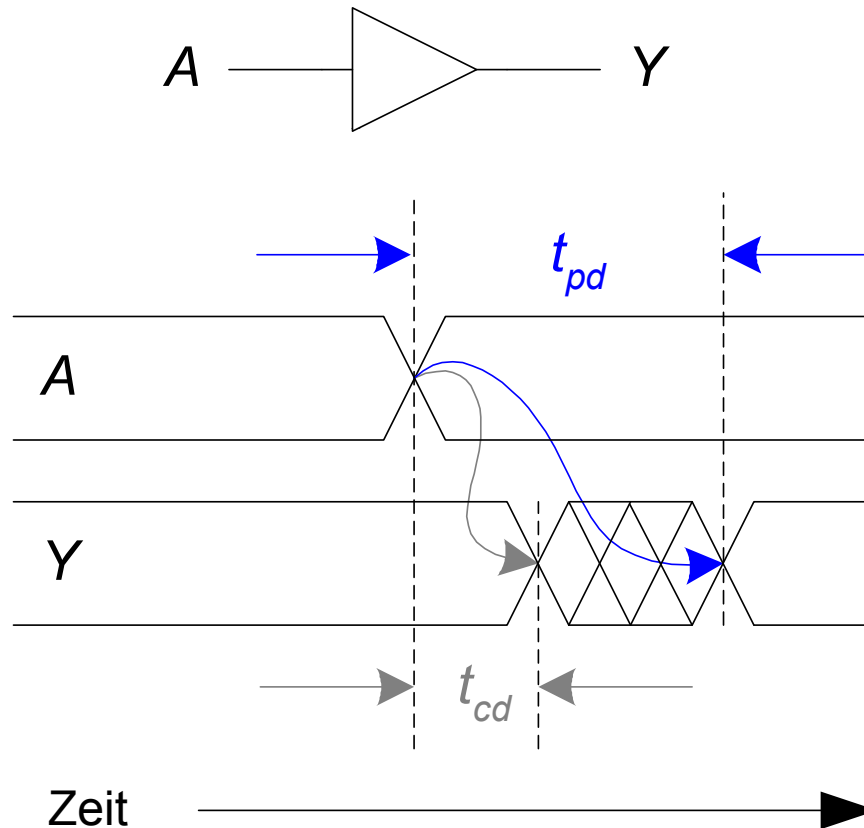
- **Verzögerung** (*delay*) zwischen Änderung am Eingang bis zur Änderung des Ausgangs
- Wie können **schnelle** Schaltungen aufgebaut werden?



# Ausbreitungs- und Kontaminationsverzögerung (*propagation*)                      (*contamination delay*)



- Ausbreitungsverzögerung:  $t_{pd}$  = max. Zeit vom Eingang zum Ausgang
- Kontaminationsverzögerung:  $t_{cd}$  = min. Zeit vom Eingang zum Ausgang

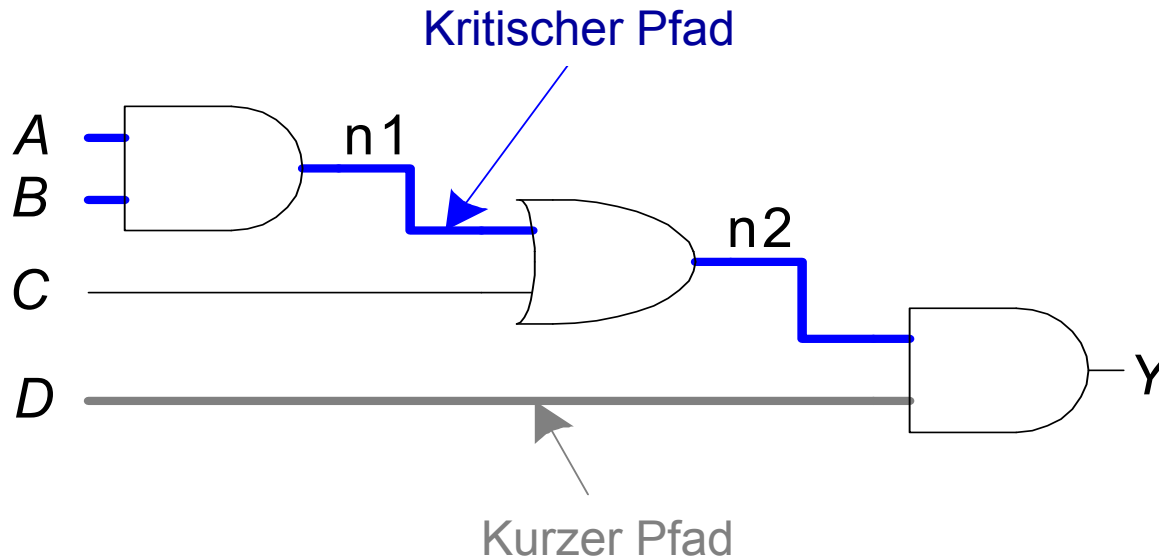


# Ausbreitungs- und Kontaminationsverzögerung



- Ursachen für **Verzögerung**
  - Kapazitäten, Induktivitäten und Widerstände in der Schaltung
  - Lichtgeschwindigkeit als maximale Ausbreitungsgeschwindigkeit
  
- Warum können  $t_{pd}$  und  $t_{cd}$  **unterschiedlich** sein?
  - Unterschiedliche Verzögerungen für steigende und fallende **Flanken**
  - **Mehrere** Ein- und Ausgänge
    - Mit unterschiedlich langen Verzögerungen
  - Schaltungen werden
    - ... **langsamer** bei Erwärmung
    - ... **schneller** bei Abkühlung

# Kritische (lange) und kurze Pfade



Kritischer (langer) Pfad:  $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$

Kurzer Pfad:  $t_{cd} = t_{cd\_AND}$

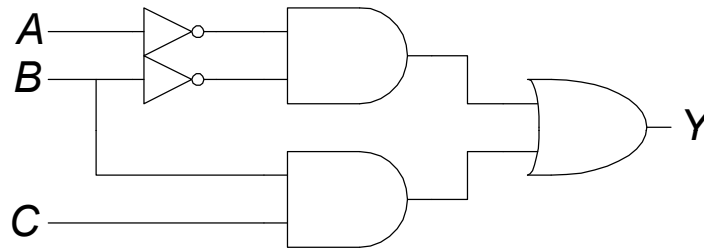
# Störimpulse (*glitches*)

## ▪ Störimpulse

- Eine Änderung eines Eingangs verursacht **mehrere** Änderungen des Ausgangs
- Können durch geeignete Entwurfsdisziplin **entschärft** werden
  - Können noch auftreten, richten aber **keinen Schaden** an
  - **Synchroner** Entwurf, kommt noch ...
    - Kann **Ausnahmen** geben
- Sollten aber im Vorfeld **erkannt** werden
  - Sichtbar im Timing-Diagramm

# Beispiel für Störimpulse

- Was passiert, wenn  $A = 0$ ,  $C = 1$ , und  $B$  fällt von  $1 \rightarrow 0$ ?

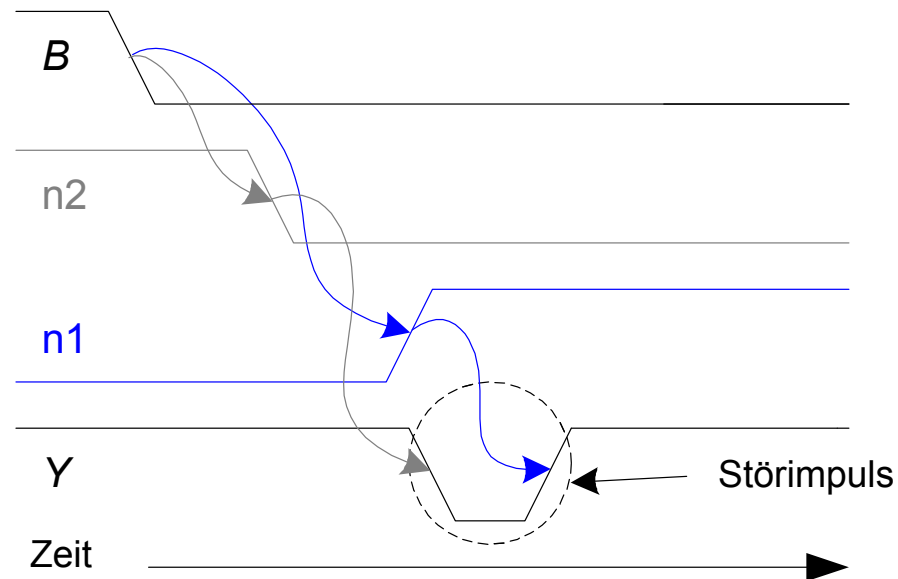
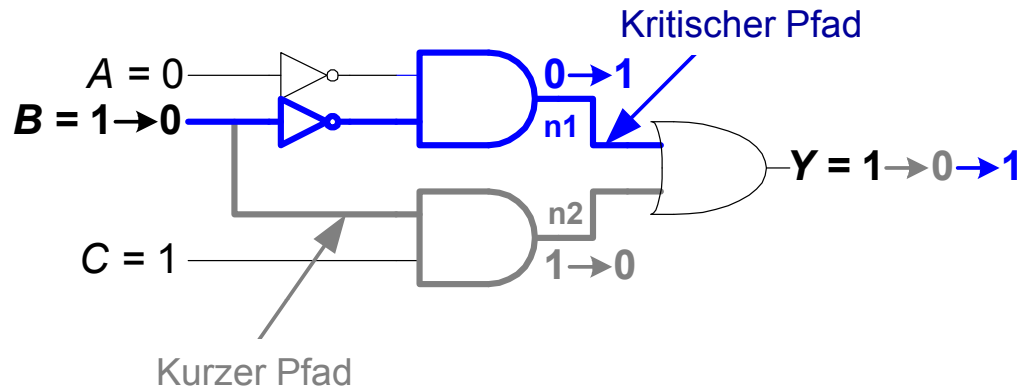


Y		AB			
		00	01	11	10
C	0	1	0	0	0
	1	1	1	1	0

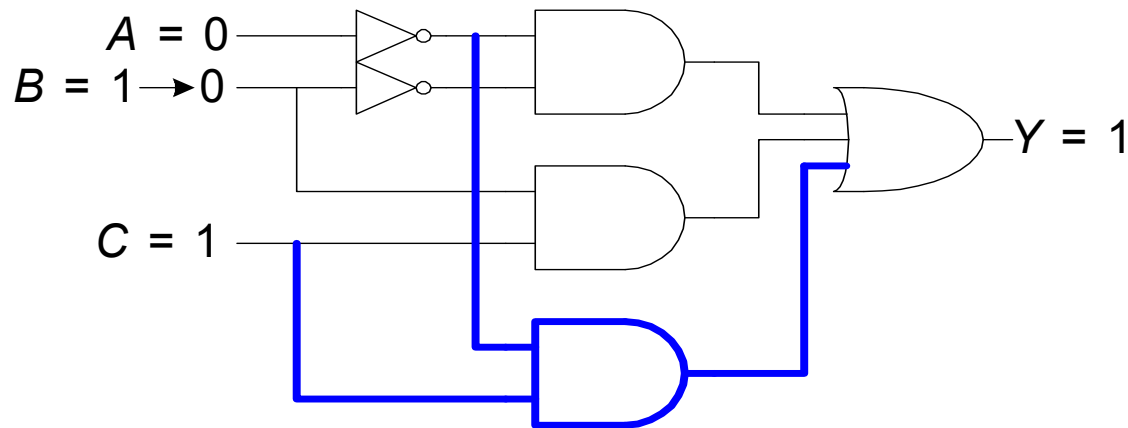
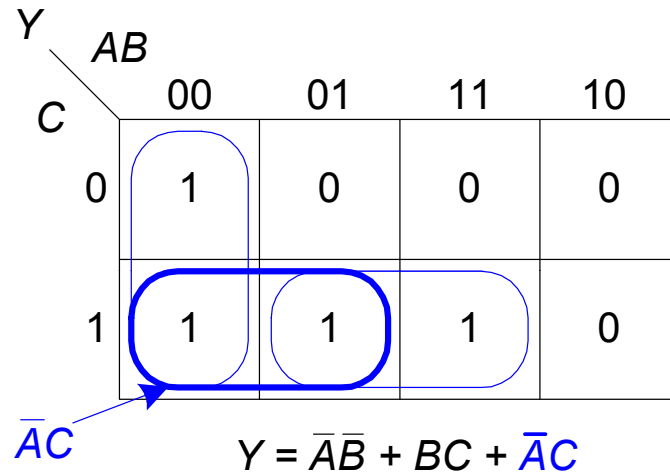
$$Y = \bar{A}\bar{B} + BC$$



# Beispiel für Störimpulse (Fortsetzung)



# Störimpuls beseitigen



# Warum Störimpulse beachten?

- Störimpulse verursachen keine Probleme bei **synchronem** Entwurf
  - In der Regel, auch da **Fehlerquellen**
  - → Kapitel 3
- Sollten aber **erkannt** werden
  - Beim Debugging einer Schaltung im Simulator oder mit dem Oszilloskop
- **Nicht** alle Störimpulse können beseitigt werden
  - Z.b. bei **gleichzeitigem** Schalten mehrerer Eingänge