Digitaltechnik – Kapitel 2

Prof. Sarah Harris, Ph.D.
Fachgebiet Eingebettete Systeme und ihre Anwendungen (ESA)
Fachbereich Informatik

WS 15/16

Kapitel 2: Kombinatorische Logik

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

Inhalte dieser Veranstaltung

Anwendungs-software	Programme
Betriebs-systeme	Gerätetreiber
Architektur	Befehle Register
Mikro-architektur	Datenpfade Steuerung
Logik	Addierer Speicher
Digital-schaltungen	UND Gatter Inverter
Analog-schaltungen	Verstärker Filter
Bauteile	Transistoren Dioden
Physik	Elektronen

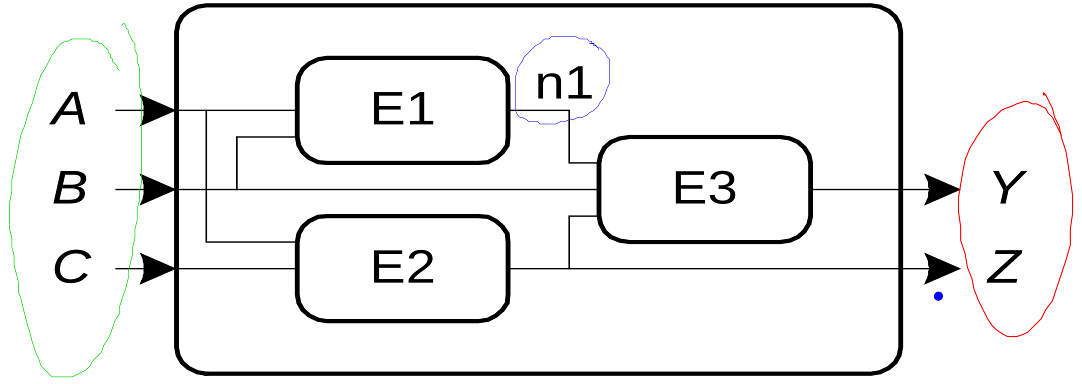
Einleitung

Eine logische Schaltung ist zusammengesetzt aus

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

Schaltungen

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Kombinatorische Logik**
  - Zustandslos
  - Ausgänge hängen nur von aktuellen Eingangswerten ab

2

- **Sequentielle Logik**
  - Speichert einen Zustand

Gedächtnis

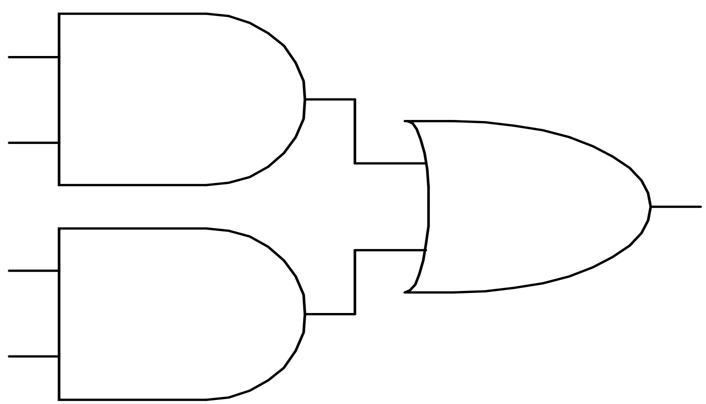
  - Ausgänge hängen ab von aktuellen Eingangswerten und gespeichertem Zustand
    - Also damit auch von vorherigen Eingangswerten

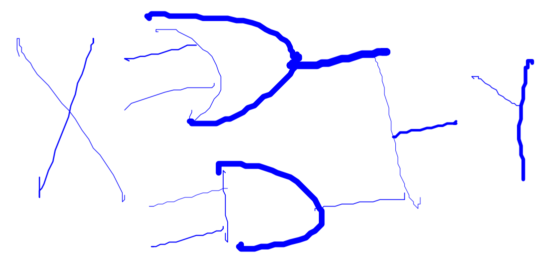
3

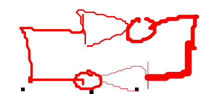
Embedded Systems & Applications

Regeln für kombinatorische Zusammensetzung

- 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

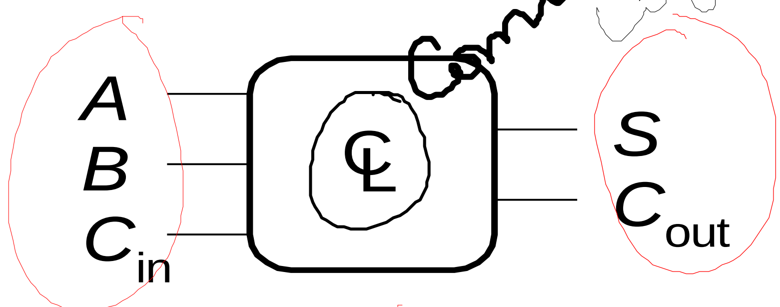
- Beschreiben Ausgänge als Funktion der Eingänge

- Beispiel:

Volladdierer

$$S = F_1(A, B, C_{in})$$
$$C_{out} = F_2(A, B, C_{in})$$

Combinational



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

1 Cin
0 A
+ 1 B
1 0 S
↑
Cout

Grundlegende Definitionen

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

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

Zwingen den Ausgang auf 1

A	B	Minterm	Y
0	0	$\bar{A} \cdot \bar{B}$	0
0	1	$\bar{A} \cdot B$	1
1	0	$A \cdot \bar{B}$	0
1	1	$A\,B$	1

$A = 0,\ B = 1$

$\bar{A} \cdot \bar{B}$

$1 \cdot 0 = 0$

Produkten

Summe

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

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

A	*B*	*Y*	**minterm**	**minterm name**
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

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**	**minterm name**
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

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	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

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

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**	**minterm name**
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

$$Y = F(A, B) = \overline{A}B + AB = \Sigma(1, 3)$$

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

A	B	Maxterm	Y
0	0	$(A + B)$	0
0	1	$A + \bar{B}$	1
1	0	$(\bar{A} + B)$	0
1	1	$\bar{A} + \bar{B}$	1

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

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

A	*B*	*Y*	**maxterm**	**maxterm name**
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

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**	**maxterm name**
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

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)

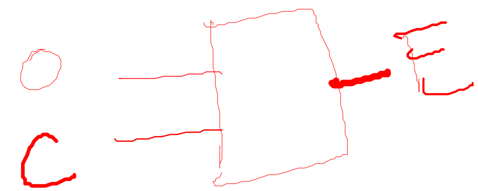
dem Ausgang auf $\boxed{0}$ zwingen

A	*B*	*Y*	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

$$Y = F(A, B) = (A + B)(A + \overline{B}) = \Pi(0, 2)$$

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)

KNF (POS)

Beispiel für Boole'sche Funktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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	$O\ C$

$E = O\overline{C}$

- **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	$\overline{O} + \overline{C}$

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

Embedded Systems & Applications

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

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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	$O\ C$

$$E = O\overline{C}$$
$$= \Sigma(2)$$

- **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	$\overline{O} + \overline{C}$

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$
$$= \Pi(0, 1, 3)$$

Embedded Systems & Applications

Boole'sche Algebra

- **Axiome und Sätze**, hier zum Ziel die Vereinfachung boole'scher Gleichungen
- Wie die übliche Algebra
  - Teilweise einfacher, da hier nur zwei Werte 0, 1

Embedded Systems & Applications

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 der Boole'schen Algebra

Nummer	Axiom	Name
A1	$B = 0$ if $B \neq 1$	Dualitätsgesetz
A2	$\overline{0} = 1$	NOT
A3	$0 \bullet 0 = 0$	AND/OR
A4	$1 \bullet 1 = 1$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	AND/OR

Axiome der Boole'schen Algebra

Nummer	Axiom	Name
A1	$B = 0$ if $B \neq 1$	Dualitätsgesetz
A2	$\overline{0} = 1$	NOT
A3	$0 \bullet 0 = 0$	AND/OR
A4	$1 \bullet 1 = 1$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	AND/OR

Dual: Tausche: $\bullet$ mit $+$
0 mit 1

Axiome der Boole'schen Algebra

Nummer	Axiom	Dual	Name
A1	$B = 0$ if $B \neq 1$	$B = 1$ if $B \neq 0$	Dualitätsgesetz
A2	$\overline{0} = 1$	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	$1 + 0 = 0 + 1 = 1$	AND/OR

Dual: Tausche: • mit +
0 mit 1

Sätze der Boole'schen Algebra

Theorem

Nr.	Satz	Name
T1	$B \bullet 1 = B$	Neutralitätsgesetz
T2	$B \bullet 0 = 0$	Extremalgesetz
T3	$B \bullet B = B$	Idempotenzgesetz
T4	$\overline{\overline{B}} = B$	Involution
T5	$B \bullet \overline{B} = 0$	Komplementärgesetz

Sätze der Boole'schen Algebra

Nr.	Satz	Name
T1	$B \bullet 1 = B$	Neutralitätsgesetz
T2	$B \bullet 0 = 0$	Extremalgesetz
T3	$B \bullet B = B$	Idempotenzgesetz
T4	$\overline{\overline{B}} = B$	Involution
T5	$B \bullet \overline{B} = 0$	Komplementärgesetz

Dual: Tausche: $\bullet$ mit $+$
0 mit 1

Sätze der Boole'schen Algebra

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

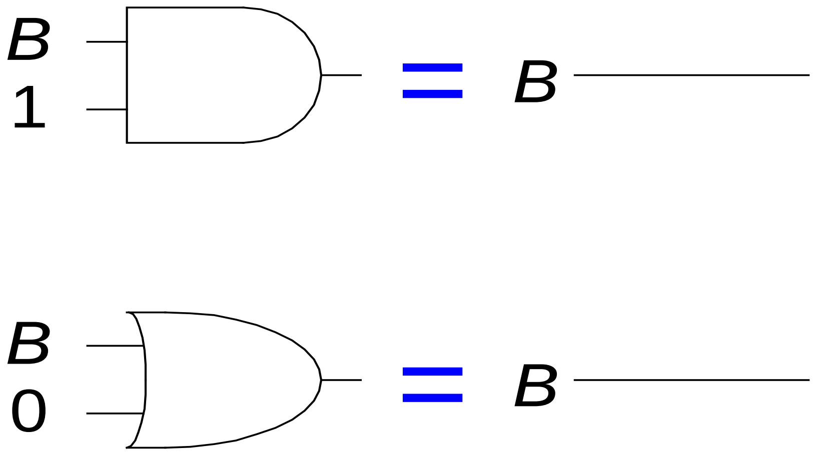
Dual: Tausche: • mit +
0 mit 1

T1: Neutralitätsgesetz

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

T1: Neutralitätsgesetz

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

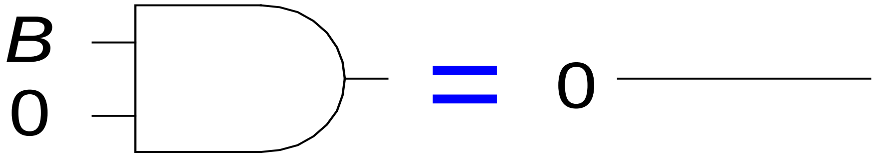


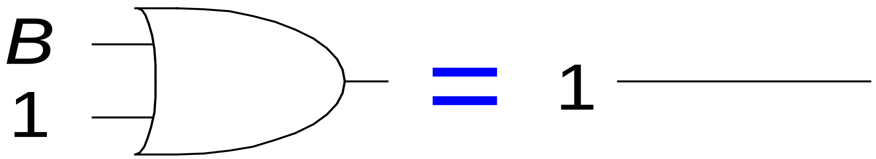
T2: Extremalgesetz

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

T2: Extremalgesetz

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



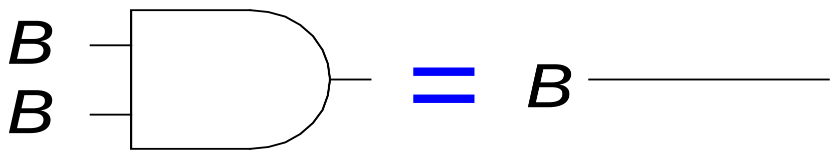


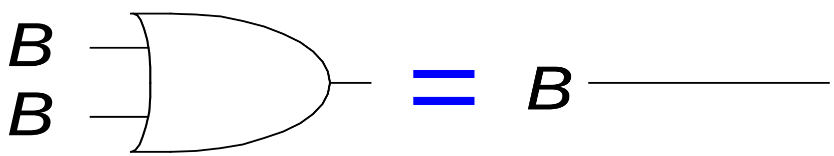
T3: Idempotenzgesetz

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

T3: Idempotenzgesetz

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





T4: Involution (Selbstinversion)

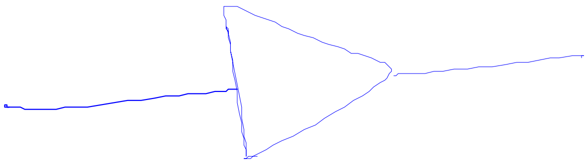
TECHNISCHE UNIVERSITÄT DARMSTADT

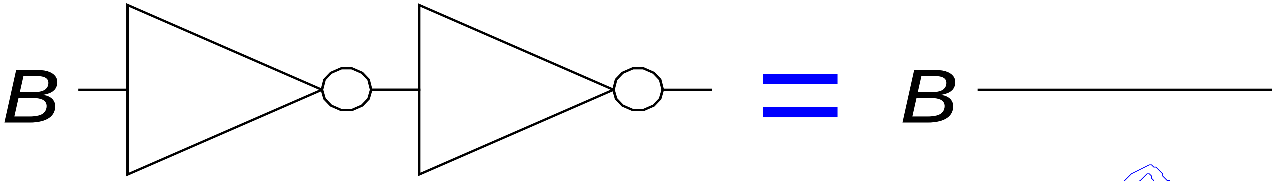
- $\overline{\overline{B}} =$

Embedded Systems & Applications

T4: Involution (Selbstinversion)

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



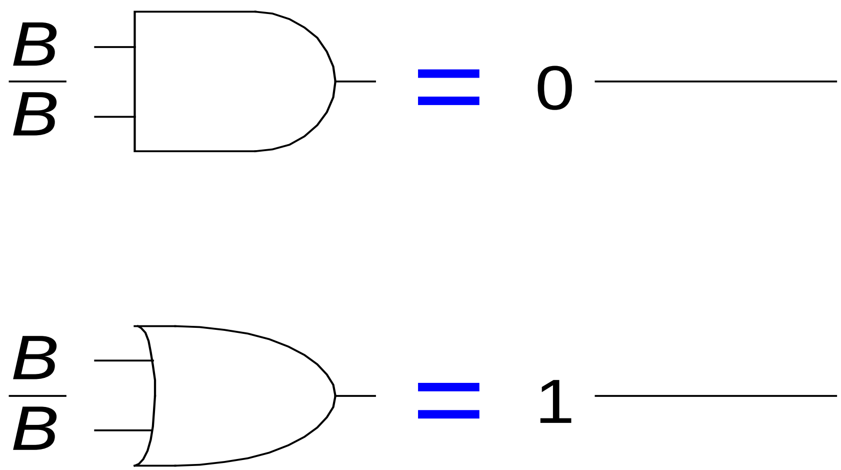


T5: Komplementärgesetz

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

T5: Komplementärgesetz

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



Sätze der Boole'schen Algebra mit einer Variablen

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Embedded Systems & Applications

Sätze der Boole'schen Algebra mit mehreren Variablen

TECHNISCHE UNIVERSITÄT DARMSTADT

Nr.	Satz	Name
T6	$B \bullet C = C \bullet B$	**Kommutativgesetz**
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	**Assoziativgesetz**
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	**Distributivgesetz**
T9	$B \bullet (B+C) = B$	**Absorptionsgesetz**
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	**Zusammenfassen**
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	**Konsensusregeln**

Embedded Systems & Applications

Sätze der Boole'schen Algebra mit mehreren Variablen

Nr.	Satz	Name
T6	B•C = C•B	**Kommutativgesetz**
T7	(B•C) • D = B • (C • D)	**Assoziativgesetz**
T8	B • (C + D) = (B•C) + (B•D)	**Distributivgesetz**
T9	B• (B+C) = B	**Absorptionsgesetz**
T10	(B•C) + (B•$\overline{C}$) = B	**Zusammenfassen**
T11	(B•C) + ($\overline{B}$•D) + (C•D) = (B•C) + ($\overline{B}$•D)	**Konsensusregeln**

Wie bestimmt man, ob die Aussagen wahr sind?

Wie beweisen wir die Sätze?

- **Methode 1:** überprüfen alle Möglichkeiten
- **Methode 2:** die Gleichung vereinfachen
  - andere Axiome und Sätze verwenden
  - bis beide Seiten gleich sind

Beispiel 1: überprüfen alle Möglichkeiten

Nr.	Satz	Name
T6	B•C = C•B	**Kommutativgesetz**

B	***C***	***BC***	***CB***
0	0		
0	1		
1	0		
1	1		

Beispiel 1: überprüfen alle Möglichkeiten

TECHNISCHE UNIVERSITÄT DARMSTADT

Nr.	Satz	Name
T6	B•C = C•B	**Kommutativgesetz**

B	***C***	***BC***	***CB***
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Beispiel 2: Absorptionsgesetz

Nr	Satz	Name
T9	$B \bullet (B+C) = B$	Absorptionsgesetz

Wahrheit prüfen durch:

- **Methode 1:** überprüfen alle Möglichkeiten
- **Methode 2:** die Gleichung vereinfachen

Beispiel 2: Absorptionsgesetz

Nr	Satz	Name
T9	B• (B+C) = B	Absorptionsgesetz

- **Methode 1:** überprüfen alle Möglichkeiten

B	***C***	***(B+C)***	***B(B+C)***
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Beispiel 2: Absorptionsgesetz

Nr	Satz	Name
T9	B• (B+C) = B	Absorptionsgesetz

- **Methode 1:** überprüfen alle Möglichkeiten

B	*C*	*(B+C)*	*B(B+C)*
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Beispiel 3: Absorptionsgesetz

Nr	Satz	Name
T9	B• (B+C) = B	Absorptionsgesetz

- **Methode 2:** die Gleichung vereinfachen

B•(B+C)

BB + BC

B + BC

B (1 + C)

B • 1

B

T8

T3

T8

T2

T1

Beispiel 3: Absorptionsgesetz

Nr	Satz	Name
T9	B• (B+C) = B	Absorptionsgesetz

- **Methode 2:** die Gleichung vereinfachen

B•(B+C)

= B•B + B•C	T8: Distributivgesetz
= **B** + B•C	T3: Idempotenzgesetz
= B•(1 + C)	T8: Distributivgesetz
= B•(**1**)	T2: Extremalgesetz
= B	T1: Neutralitätsgesetz

Beispiel 4: Vereinfachen von Boole'schen Ausdrücken

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

$= B\underbrace{(A + \overline{A})}_{1}$

$= B \cdot 1$

$= B$ ✓

DNF

$= \underbrace{ABC}\,F + \underbrace{ABC}\,\overline{F}$

$= ABC$

$BA + B\overline{A}$

$= B$

Beispiel 4: Vereinfachen von Boole'schen Ausdrücken

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

$= B(A + \overline{A})$	T8	Distributivgesetz
$= B(1)$	T5'	Komplementärgesetz
$= B$	T1	Neutralitätsgesetz

Beispiel 5: Vereinfachen von Boole'schen Ausdrücken

- $Y \quad = A(AB + ABC)$

Beispiel 5: Vereinfachen von Boole'schen Ausdrücken

TECHNISCHE UNIVERSITÄT DARMSTADT

$Y = A(AB + ABC)$	
$= A(AB(1 + C))$	T8 Distributivgesetz
$= A(AB(1))$	T2' Extremalgesetz
$= A(AB)$	T1 Neutralitätsgesetz
$= (AA)B$	T7 Assoziativgesetz
$= AB$	T3 Idempotenzgesetz

Embedded Systems & Applications

Sätze der Boole'schen Algebra mit mehreren Variablen

TECHNISCHE UNIVERSITÄT DARMSTADT

Nr.	Satz	Name
T6	$B \bullet C = C \bullet B$	**Kommutativgesetz**
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	**Assoziativgesetz**
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	**Distributivgesetz**
T9	$B \bullet (B+C) = B$	**Absorptionsgesetz**
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	**Zusammenfassen**
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	**Konsensusregeln**

Sätze der Boole'schen Algebra mit mehreren Variablen

Nr.	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Kommutativgesetz
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Assoziativgesetz
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivgesetz
T9	B • (B+C) = B	B + (B•C) = B	Absorptionsgesetz
T10	$(B\bullet C) + (B\bullet\overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Zusammenfassen
T11	$(B\bullet C) + (\overline{B}\bullet D) + (C\bullet D) =$ $(B\bullet C) + (\overline{B}\bullet D)$	$(B+C) \bullet (\overline{B}+D) \bullet (C+D) =$ $(B+C) \bullet (\overline{B}+D)$	Konsensusregeln

Dual: Austauschen: • mit +

0 mit 1

Sätze der Boole'schen Algebra mit mehreren Variablen

TECHNISCHE UNIVERSITÄT DARMSTADT

Nr.	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Kommutativgesetz
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Assoziativgesetz
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivgesetz
T9	B • (B+C) = B	B + (B•C) = B	Absorptionsgesetz
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Zusammenfassen
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	$(B+C) \bullet (\overline{B}+D) \bullet (C+D) =$ $(B+C) \bullet (\overline{B}+D)$	Konsensusregeln

Warnung: T8’ ist mit normaler Algebra ungleich:
ODER (+) wird über UND (•) verteilt

$B + (C \cdot D) = (B+C) \cdot (B+D)$

Embedded Systems & Applications

Vereinfachen von Gleichungen

- Mehrere Beispiele am Ende der Folien

De Morgan'sches Gesetz

Nr.	Satz	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	DeMorgansche Gesetz

Das **Komplement** des **Produkts**
ist die
Summe der **Komplementen.**

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

De Morgan'sches Gesetz

Nr.	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgansche Gesetz

De Morgan'sches Gesetz

Nr.	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgansche Gesetz

Das Komplement des Produkts
ist die
Summe der Komplementen.

De Morgan'sches Gesetz

$\overline{ABC} \neq \bar{A} \cdot \bar{B} \cdot \bar{C}$

TECHNISCHE UNIVERSITÄT DARMSTADT

Nr.	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgansche Gesetz

Das Komplement des Produkts
ist die
Summe der Komplementen.

Dual: Das **Komplement** der **Summe**
ist das
Produkt der **Komplementen**.

$\overline{A+B+C} = \bar{A} \cdot \bar{B} \cdot \bar{C}$

Embedded Systems & Applications

Beispiel 1: De Morgan'sche Gesetze

$$Y = \overline{(A+\overline{BD})\overline{C}}$$

von draußen → drinnen

$$= \overline{(A+\overline{BD})} + \overline{\overline{C}}$$

$$= (\overline{A} \cdot \overline{\overline{BD}}) + C$$

$$= (\overline{A} \cdot BD) + C$$

$$= \overline{A}BD + C$$

Beispiel 1: De Morgan'sche Gesetze

$$Y = \overline{(A+\overline{BD})\overline{C}}$$

$$= \overline{(A+\overline{BD})} + \overline{\overline{C}}$$

$$= (\overline{A} \bullet (\overline{\overline{BD}})) + C$$

$$= (\overline{A} \bullet (BD)) + C$$

$$= \overline{A}BD + C$$

Beispiel 2: De Morgan'sche Gesetze

$$Y = \overline{(\overline{AC}E+\overline{D}) + B}$$

Beispiel 2: De Morgan'sche Gesetze

$$Y = \overline{(\overline{AC}E+\overline{D}) + B}$$

$$= \overline{(\overline{AC}E+\overline{D})} \bullet \overline{B}$$

$$= (\overline{\overline{AC}E} \bullet \overline{\overline{D}}) \bullet \overline{B}$$

$$= ((\overline{\overline{AC}}+\overline{E}) \bullet D) \bullet \overline{B}$$

$$= ((AC+\overline{E}) \bullet D) \bullet \overline{B}$$

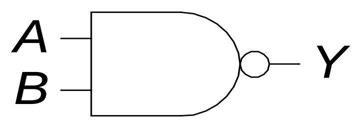
$$= (AC\overline{D} + D\overline{E}) \bullet \overline{B}$$

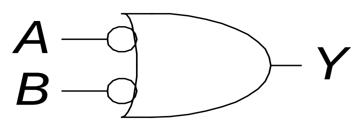
$$= A\overline{B}CD + \overline{B}D\overline{E}$$

De Morgan'sche Gesetze

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

NAND



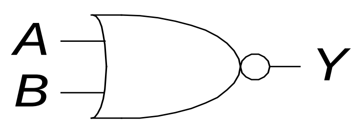


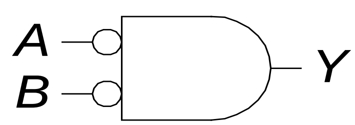
$Y = \overline{AB}$

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

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

NOR





Wichtige Mitteilungen

- die Klausur
- Selbsttest
- SystemVerilog Live Demo

Die Klausur

- Übungen selbst durcharbeiten (auf Papier!)
- an die Übungsgruppen teilnehmen
- keine Hilfsmittel
- Sprechstunden besuchen – jetzt (nicht warten bis zum Ende)
- in vorgehenden Jahren bis zu 50% durchgefallen

Vorbereitung für die Klausur

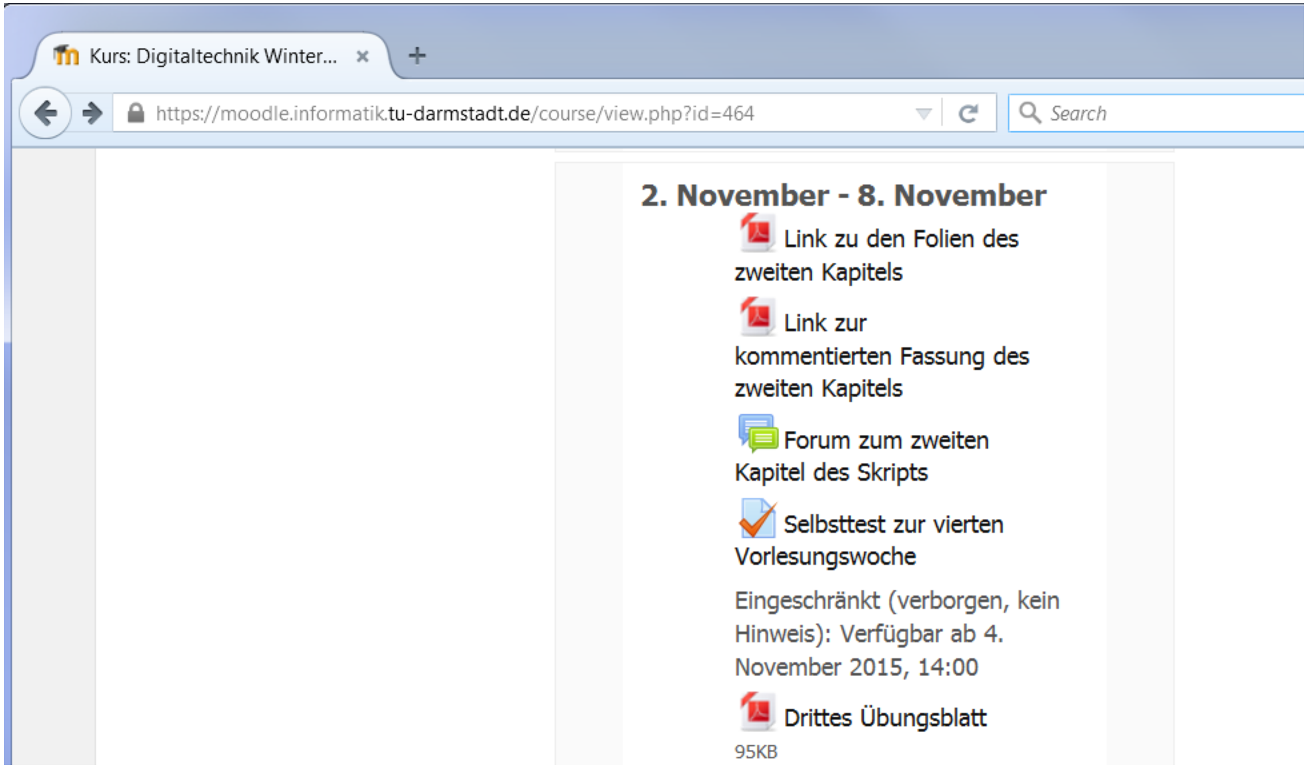
„Ich höre und ich vergesse.

Ich sehe und ich erinnere mich.

Ich tue und ich verstehe."

- Xunzi

Selbsttest bei Moodle



SystemVerilog Live Demo

- 10. Dezember 2015 (Donnerstag)
- 16:15 Uhr
- SystemVerilog:
  - Demo
  - Simulieren
  - Synthesis
  - u.s.w....

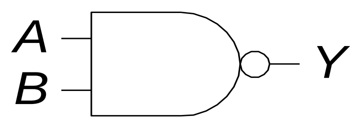
De Morgan'sche Gesetze

$\overline{AB} \neq \overline{A} \cdot \overline{B}$

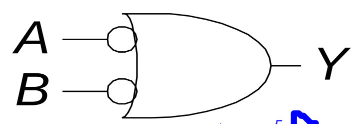
$\overline{A+B} \neq \overline{A} + \overline{B}$

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

NAND



$Y = \overline{A \cdot B}$

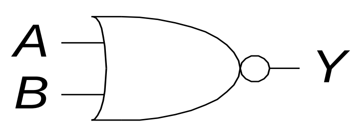


NAND

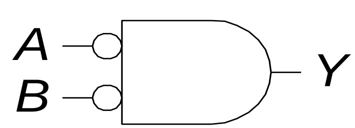
$Y = \overline{A} + \overline{B}$

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

NOR

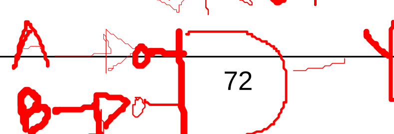


$Y = \overline{A+B}$



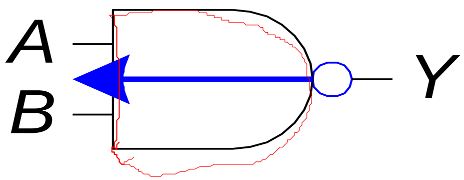
$Y = \overline{A} \cdot \overline{B}$

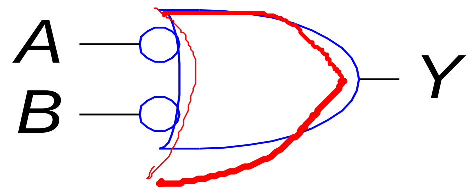
NOR

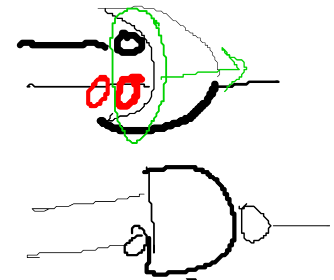


Invertierungsblasen verschieben (*bubble pushing*)

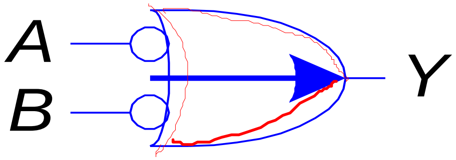
- **Blasen rückwärts Verschieben (vom Ausgang)**
  - Art des Gatters ändert sich: von AND nach OR (oder umgekehrt)
  - Blasen an allen Eingängen

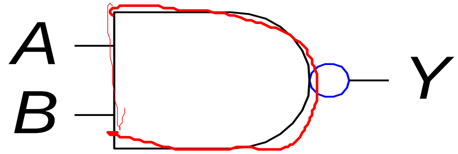






- **Blasen vorwärts Verschieben (vom Eingang)**
  - Art des Gatters ändert sich: von AND nach OR (oder umgekehrt)
  - Blasen an Ausgang
  - Müssen Blasen an allen Eingängen gewesen sein

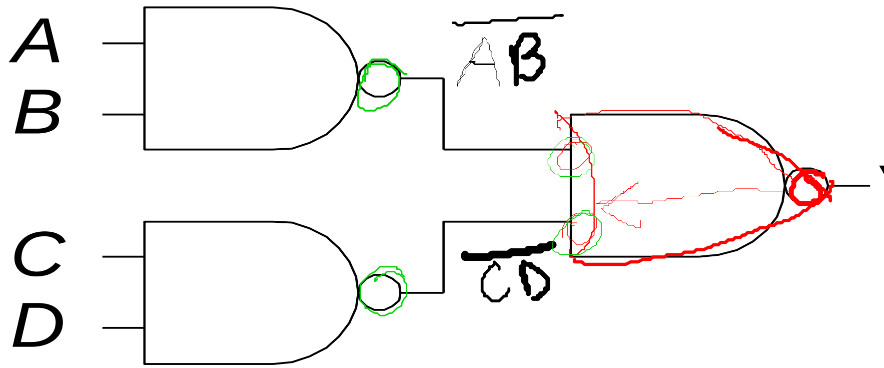




Invertierungsblasen verschieben

Was ist die boole'sche Funktion dieser Schaltung?

hätten wir das ... mal so tun können

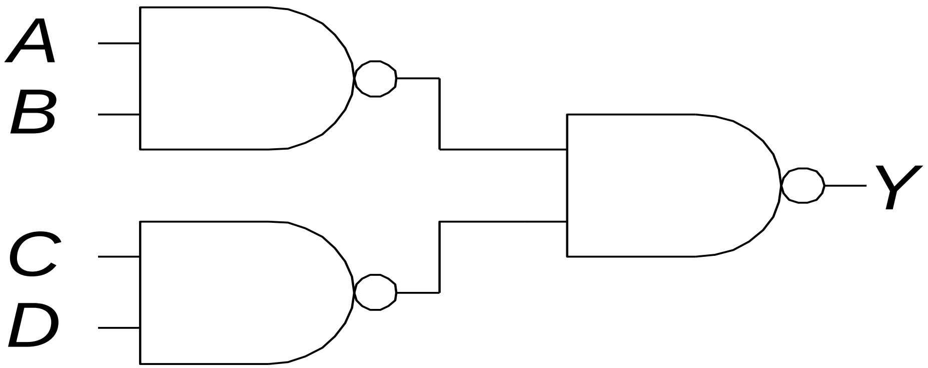


$Y = \overline{\overline{AB} \cdot \overline{CD}}$

$Y = AB + CD$

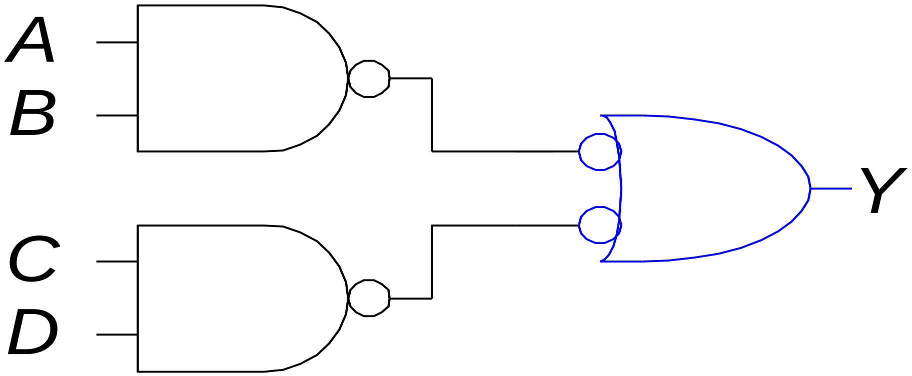
Invertierungsblasen verschieben

Was ist die boole'sche Funktion dieser Schaltung?



Invertierungsblasen verschieben

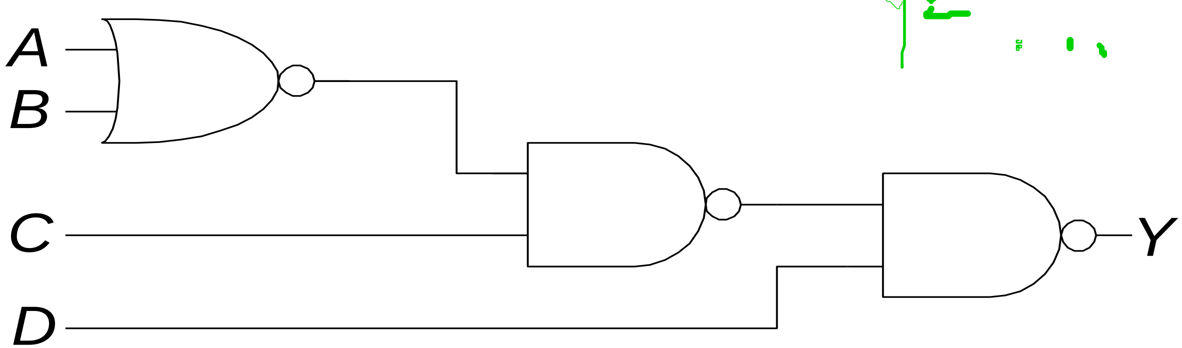
Was ist die boole'sche Funktion dieser Schaltung?



$Y = AB + CD$

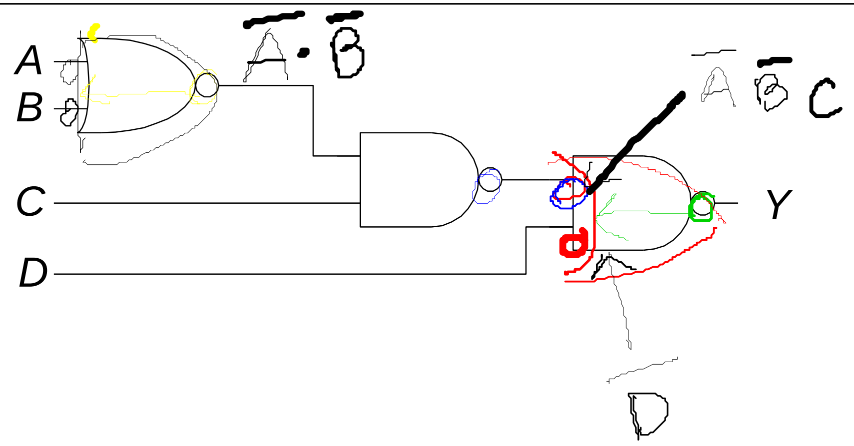
Regeln für das Verschieben von Invertierungsblasen

- 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



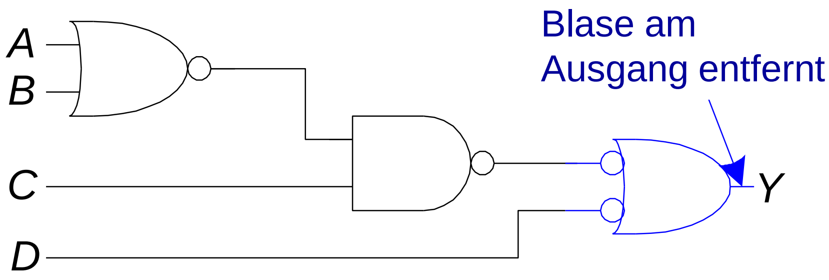
Beispiel: Invertierungsblasen verschieben

$\overline{\overline{A} \cdot \overline{B} + AB}$

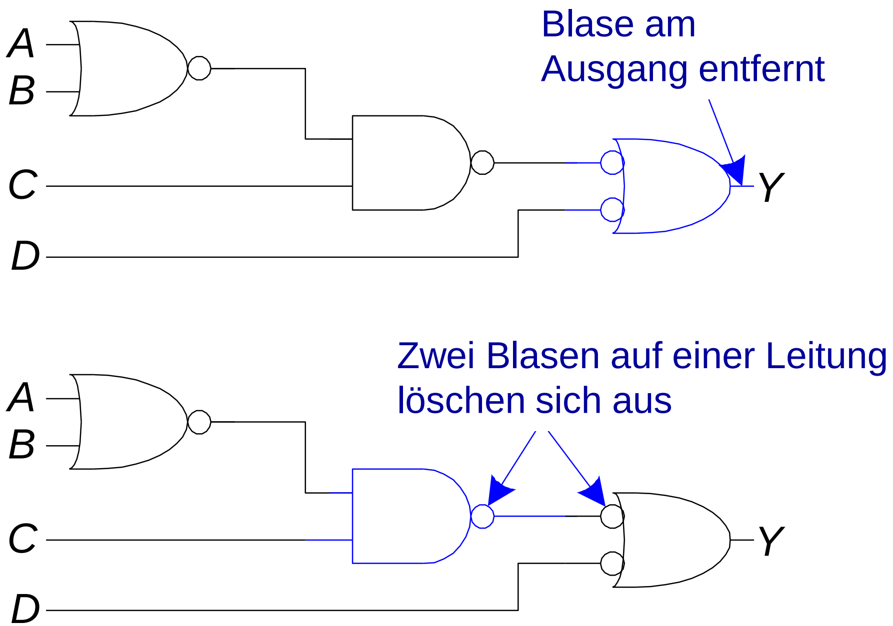


$Y = \overline{A}\,\overline{B}\,C + \overline{D}$

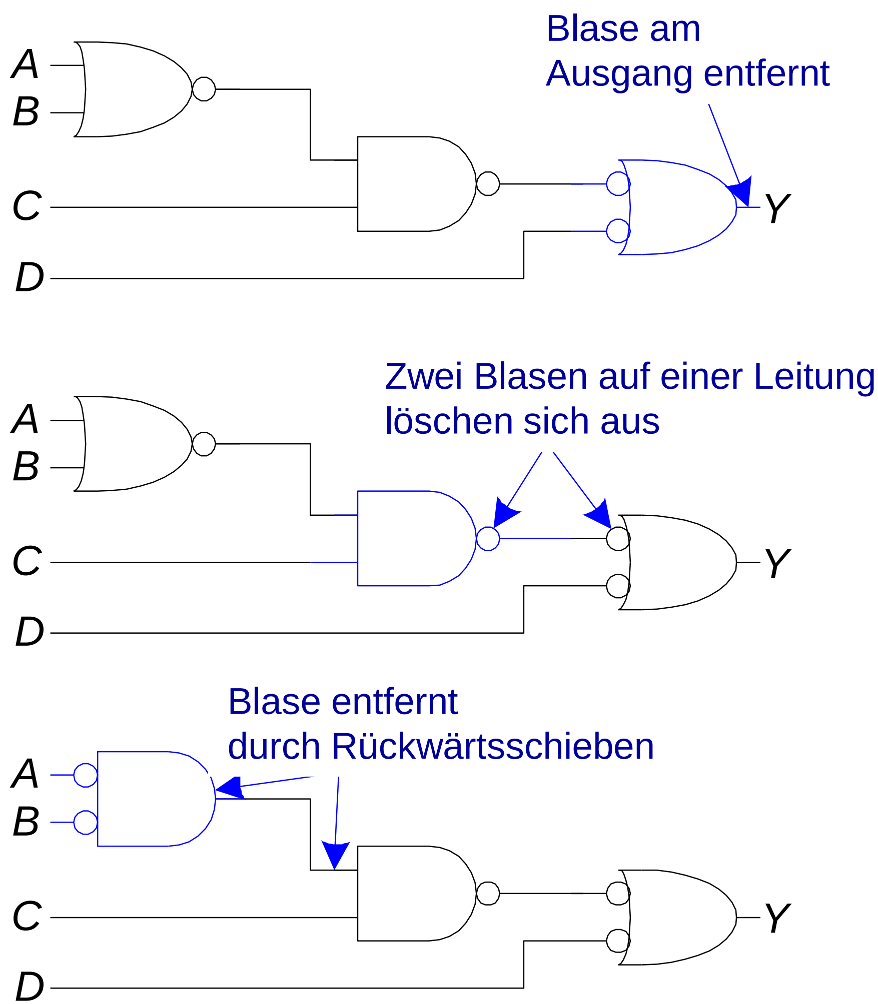
Beispiel: Invertierungsblasen verschieben



Beispiel: Invertierungsblasen verschieben



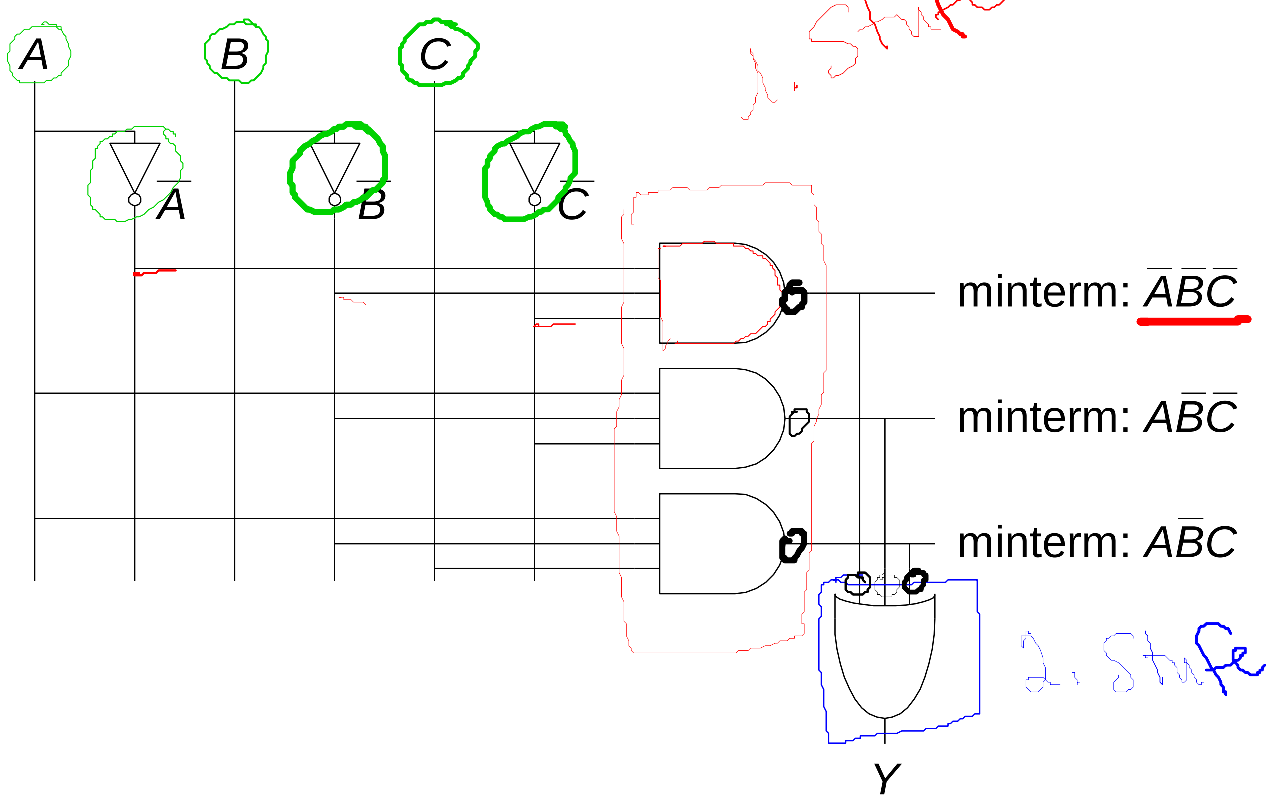
Beispiel: Invertierungsblasen verschieben



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

Von Logik zu Gattern

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



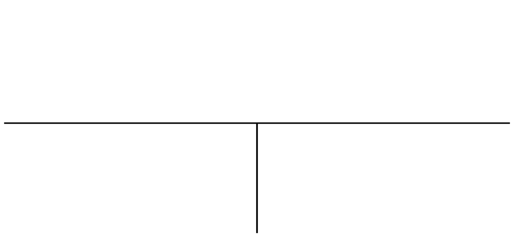
Lesbare Schaltpläne

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

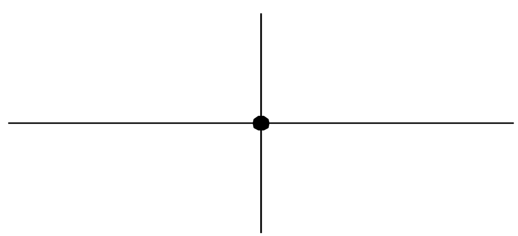
Regeln für Schaltpläne

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

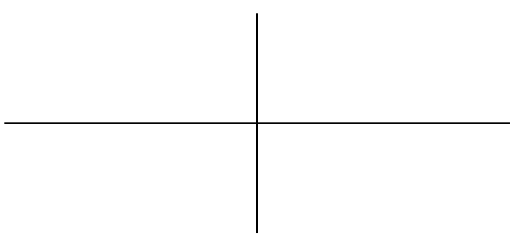
T-Kreuzung:
verbunden



Überkreuzend:
verbunden

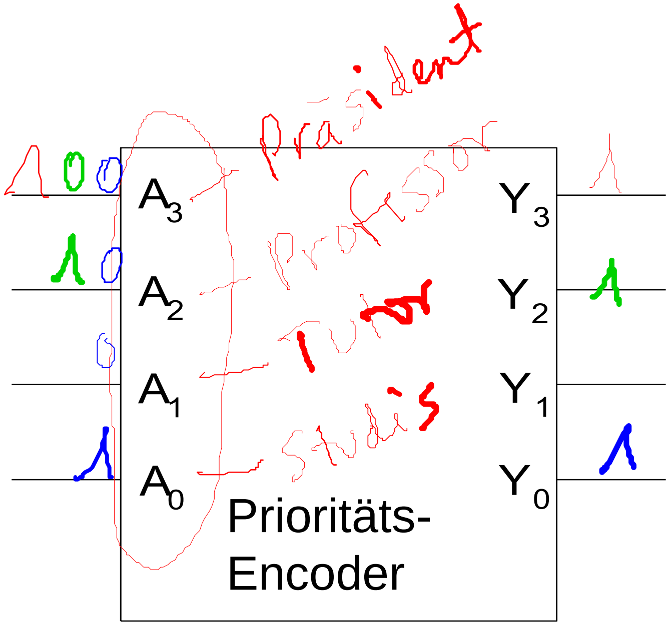


Überkreuzend:
Nicht verbunden



Schaltungen mit mehreren Ausgängen

- Ausgang entsprechend dem höchstwertigen gesetzten Eingangsbit wird auf TRUE gesetzt

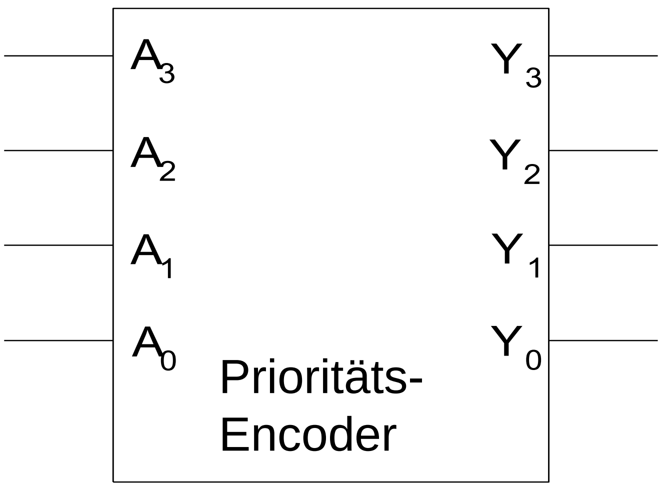


A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1				
0	1	1	0				
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1	1	0	0	0

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



Aufbau des Prioritäts-Encoders

TECHNISCHE UNIVERSITÄT DARMSTADT

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

$$Y_3 = A_3$$

$$Y_2 = \overline{A_3}\,A_2$$

$$Y_1 = \overline{A_3}\,\overline{A_2}\,A_1$$

$$Y_0 = \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,A_0$$

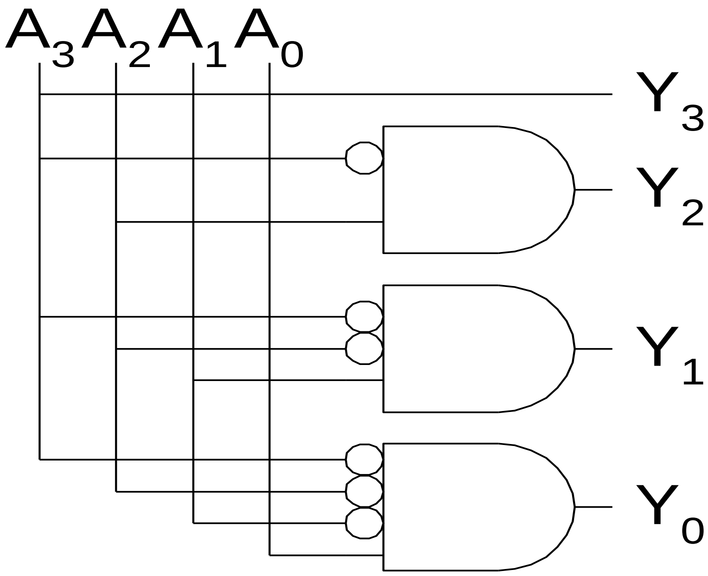
Embedded Systems & Applications

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

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**



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"*)

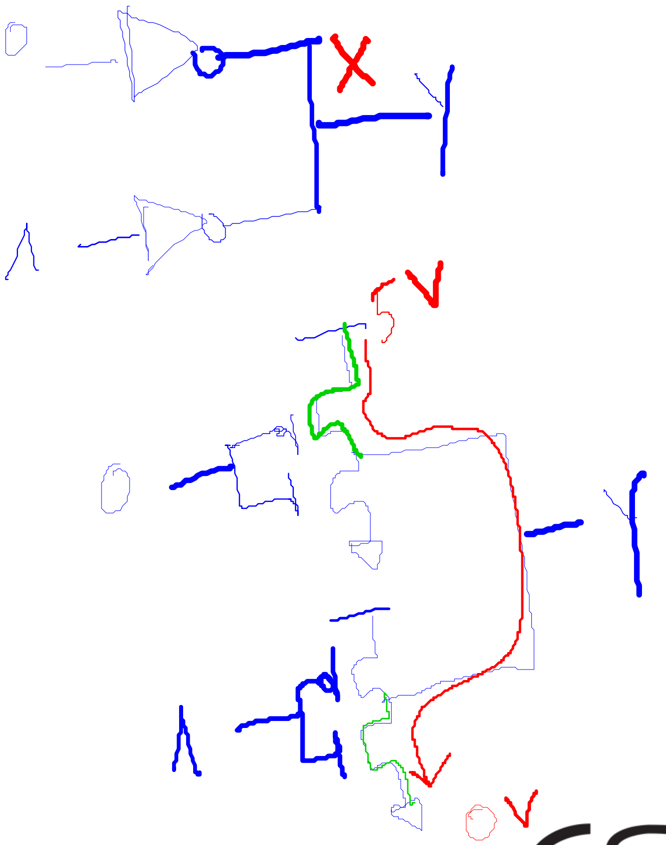
X

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

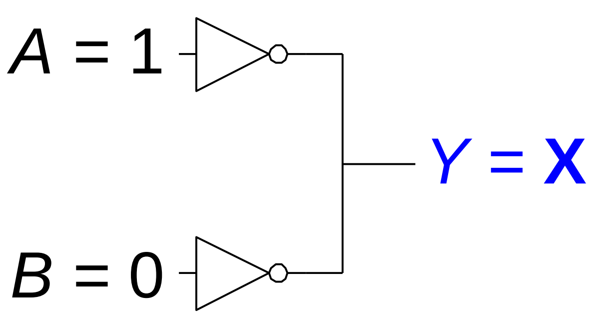
Konflikt: Schaltung treibt eine Leitung/Ausgang gleichzeitig auf 0 und 1



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)



Konkurrierende Treiber: X

Konflikt: Schaltung treibt eine Leitung/Ausgang gleichzeitig auf 0 und 1

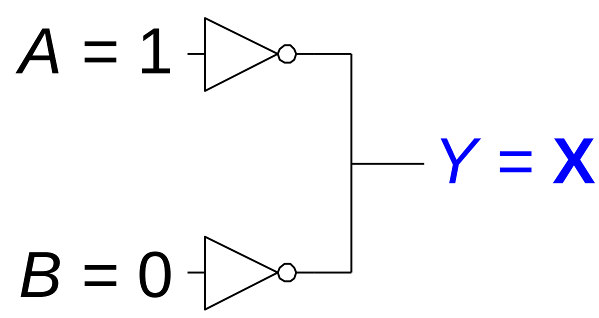
- Analogwert liegt irgendwo dazwischen (Spannungsteilung)
- Kann 0 oder 1 sein, oder im verbotenen Bereich liegen
- Kann auch mit Betriebsspannung, Temperatur, Rauschen etc. variieren
- Verursacht hohen Energieverbrauch (Kurzschluss)

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*

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

Hochohmiger Ausgang: Z

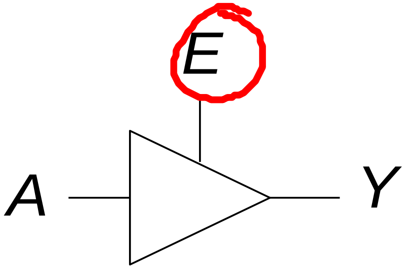
Auch genannt:

- Offen, ungetrieben
- *Floating, open, high-impedance*

Kann 0 oder 1 sein, oder irgendwo dazwischen liegen

- Leitung hat keinen aktiven Treiber

Tristate Buffer



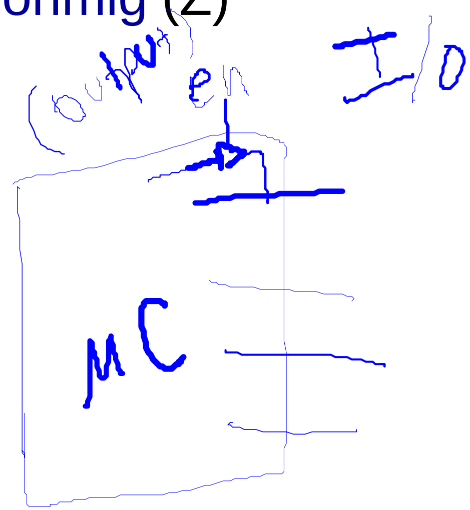
E	A	Y	
0	0	Z	1
0	1	Z	
1	0	0	2
1	1	1	3

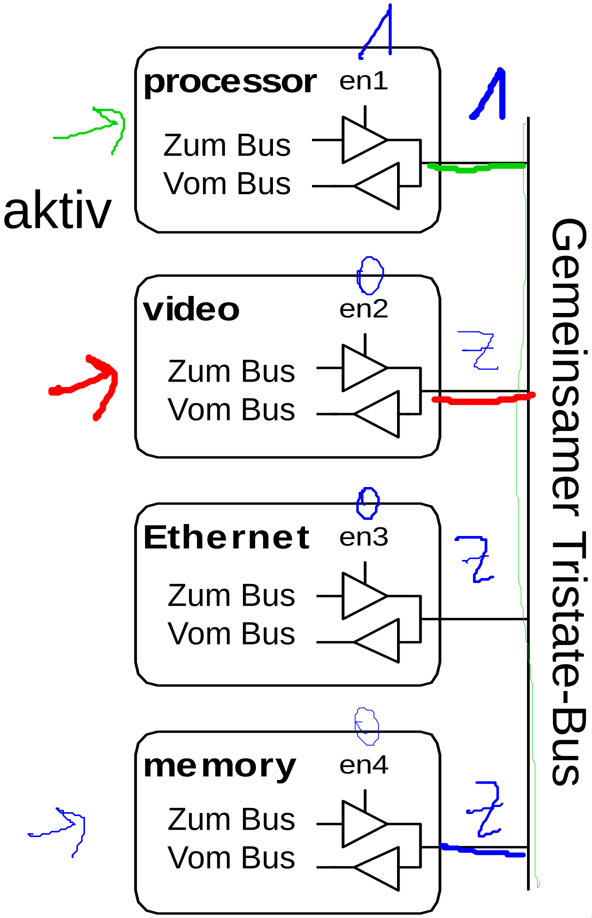
Embedded Systems & Applications

Tristate-Busse

TECHNISCHE UNIVERSITÄT DARMSTADT

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





Embedded Systems & Applications

Karnaugh Diagramme (*Karnaugh maps*)

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

$$A\underbrace{BCD}_{P} + \overline{A}\underbrace{BCD}_{P}$$

$$= \underbrace{BCD}$$

$$T = VK + V\overline{K}$$

$$= V$$

Karnaugh Diagramme (*Karnaugh maps*)

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

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

Karnaugh Diagramme (*Karnaugh maps*)

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

benachbarten Kästchen ändern sich in nur einer Stelle

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

Minimierung mit Karnaugh Diagrammen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Markiere 1en in benachbarten Plätzen und bilde viereckigen Bereich
  - Jeder Platz steht für einen Minterm
- Lasse markierte Literale
  - … die im Bereich normal und als Komplement auftauchen, im Produkt 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

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

Embedded Systems & Applications

Karnaugh Diagramm mit drei Eingängen

Y

C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
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

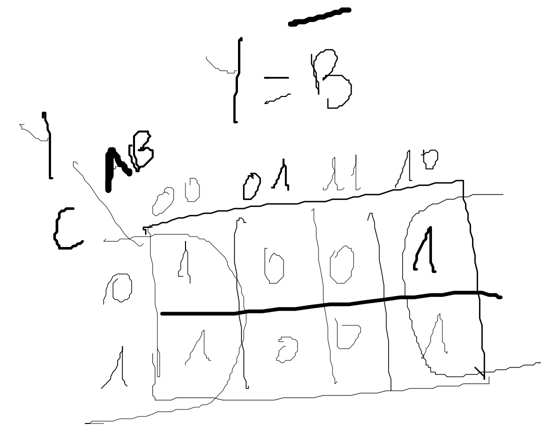
K-Map

Y

C \ AB	00	01	11	10
0				
1				

Karnaugh Diagramm mit drei Eingängen

C \ AB (Y)	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
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

C \ AB (Y)	00	01	11	10
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)

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

- **Literal:** Variable oder ihr Komplement

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

- **Implikant:** Produkt (UND) von Literalen

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

- **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,… 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

CD \ *AB*	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

$\bar{B}\bar{D}$

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

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

$\bar{A}BD$

$\bar{A}C$

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

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

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

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

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

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

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

Karnaugh-Diagramm mit “don’t cares”

TECHNISCHE UNIVERSITÄT DARMSTADT

$PA + P\bar{A} = P$

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

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

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

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

Embedded Systems & Applications

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

A

Y

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

C

D

B

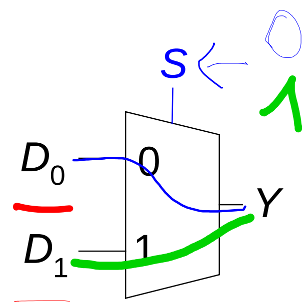
$$Y = A + \overline{B}\,\overline{D} + C$$

Kombinatorische Grundelemente

- Multiplexer
- Dekodierer (*Decoders*)

Multiplexer (Mux)

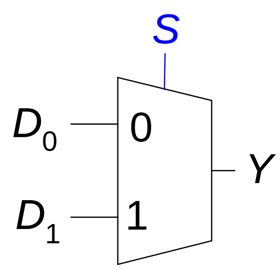
- Wählt einen von N Eingängen aus und verbindet ihn auf den Ausgang
- $\log_2 N$-bit Selektor-Eingang (*select input*), Steuereingang
- Beispiel: 2:1 Mux



S	D_1	D_0	Y
0	0	0	0
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	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



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

S	D_1	D_0	Y
0	0	0	0
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	1

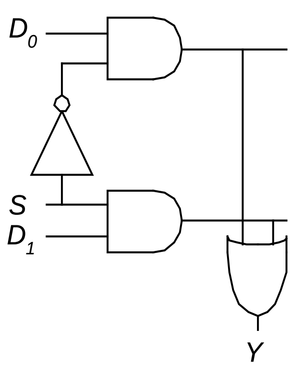
S	Y
0	D_0
1	D_1

Implementierung von Multiplexern

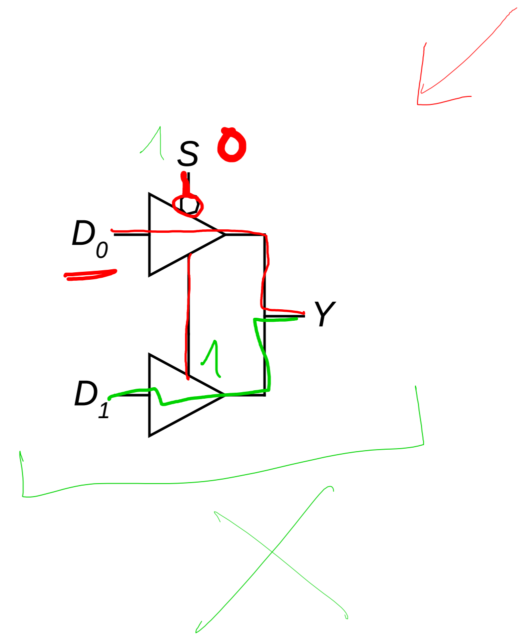
- Aus Logikgattern
  - Disjunktive Normal Form (SOP)

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

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



- 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

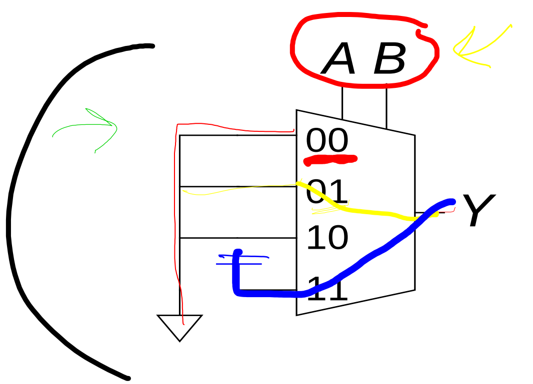
TECHNISCHE UNIVERSITÄT DARMSTADT

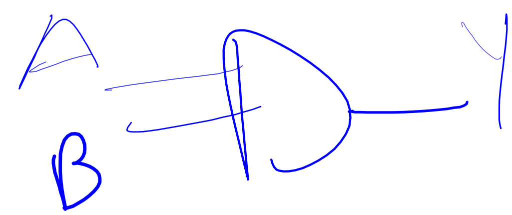
- Verwende Mux als Wertetabelle (*look-up table)*

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

AND

$$Y = AB$$





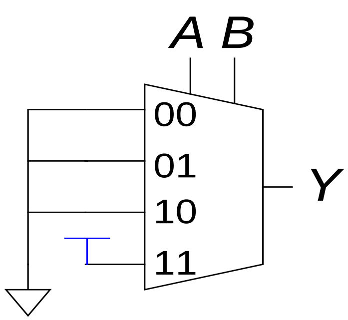
Embedded Systems & Applications

Logikfunktionen aufgebaut aus Multiplexern

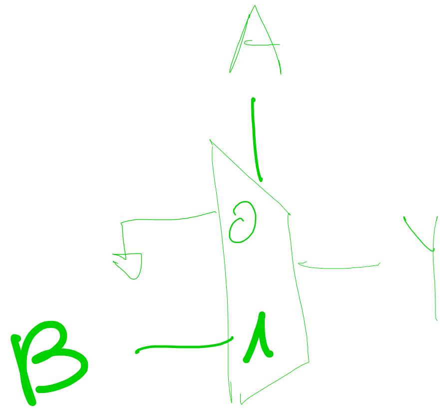
Reduziere Größe des Multiplexers

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

$Y = AB$



A	Y
0	0
1	B



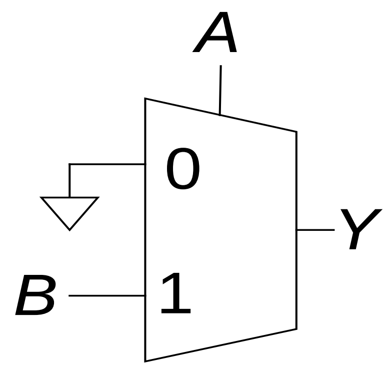
Logikfunktionen aufgebaut aus Multiplexern

Reduziere Größe des Multiplexers

$Y = AB$

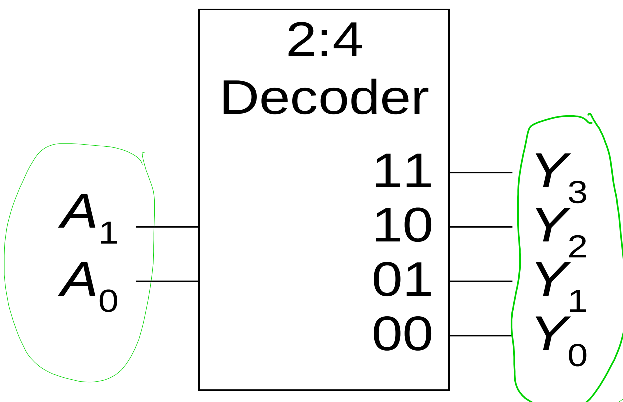
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B



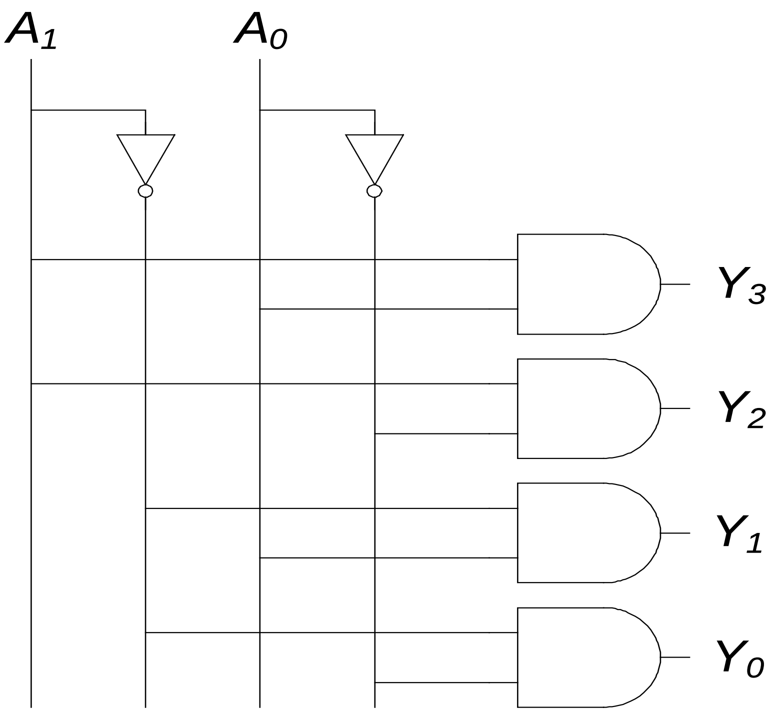
Dekodierer (*Decoder*)

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



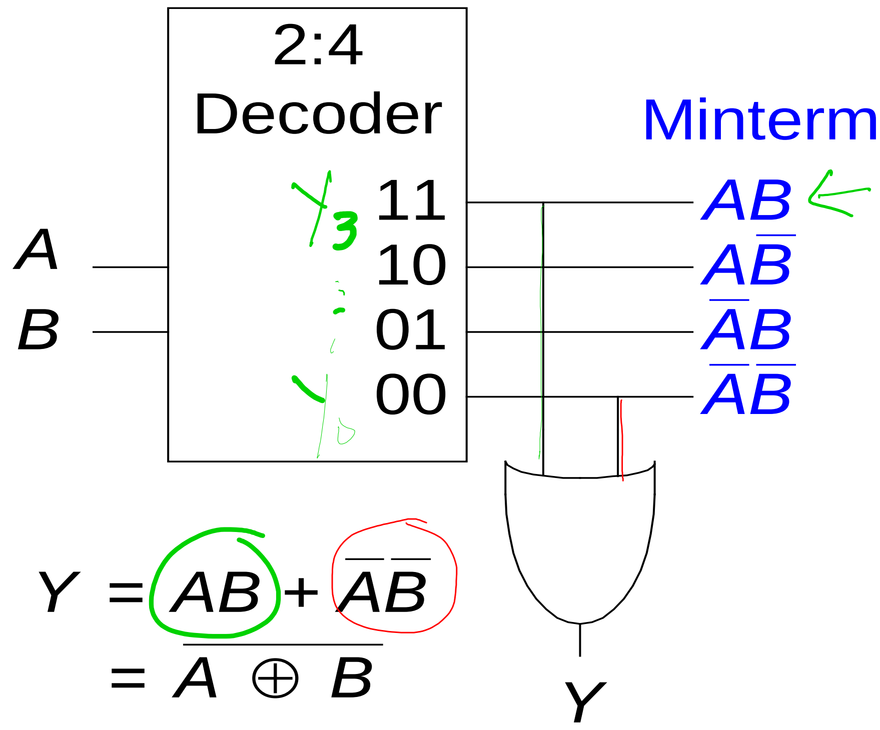
	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	1
1	0	1	0	0	1	0
2	1	0	0	1	0	0
3	1	1	1	0	0	0

Implementierung von Dekodierern



Logik aufgebaut aus Dekodierern

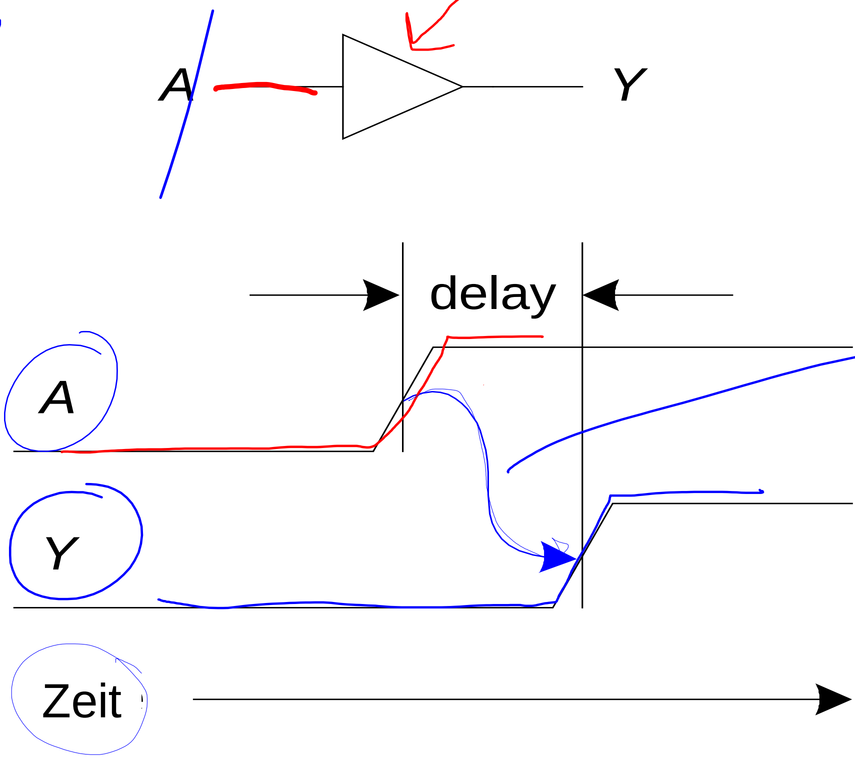
- Verknüpfe Minterme mit ODER



$$Y = AB + \overline{A}\,\overline{B}$$
$$= \overline{A \oplus B}$$

Zeitverhalten (*Timing*)

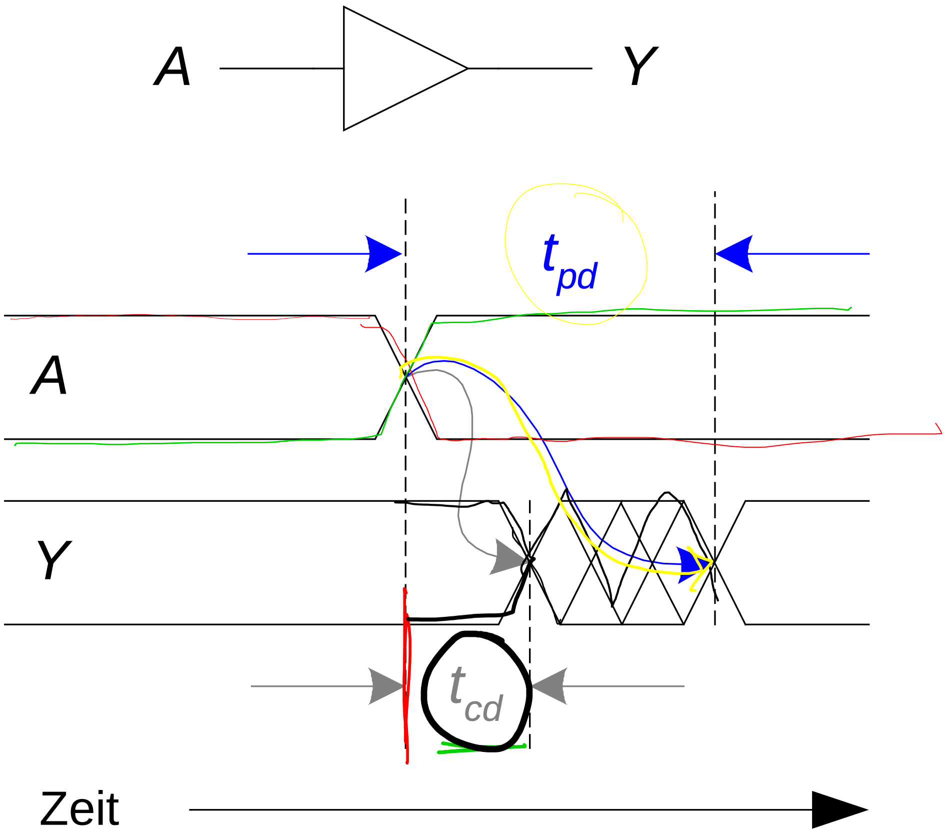
- Verzögerung (*delay*) zwischen Änderung am Eingang bis zur Änderung des Ausgangs
- Wie können schnelle Schaltungen aufgebaut werden?



Effektpfeile

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

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

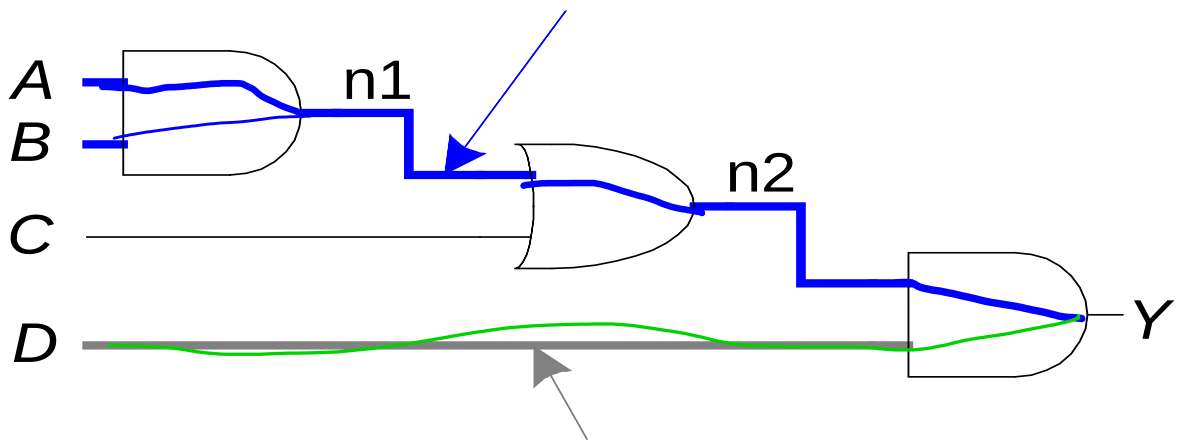


Ausbreitungs- und Kontaminationsverzögerung

- 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*)

Hazards

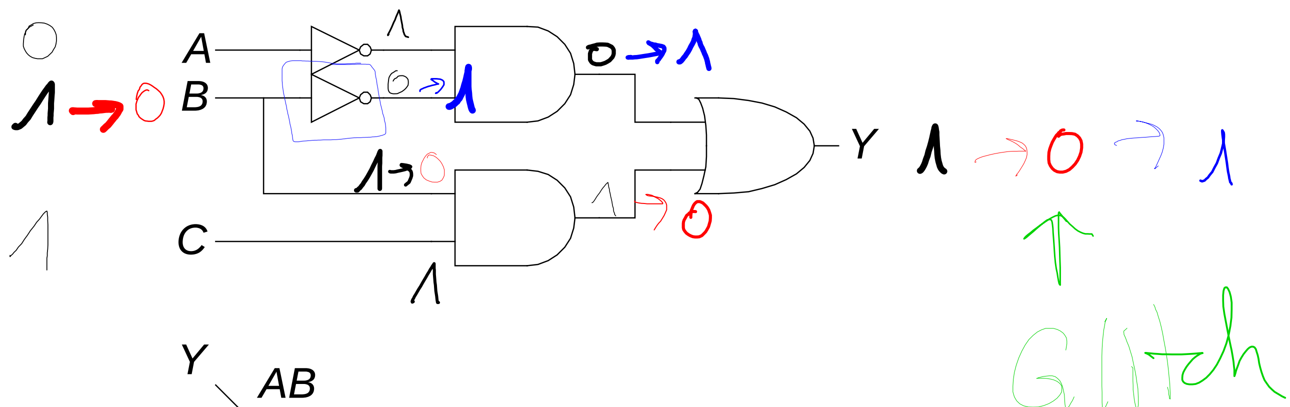
- **Störimpulse**
  - Eine Änderung eines Eingangs verursacht mehrere Änderungen des Ausgangs

- 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-Diagram

Beispiel für Störimpulse

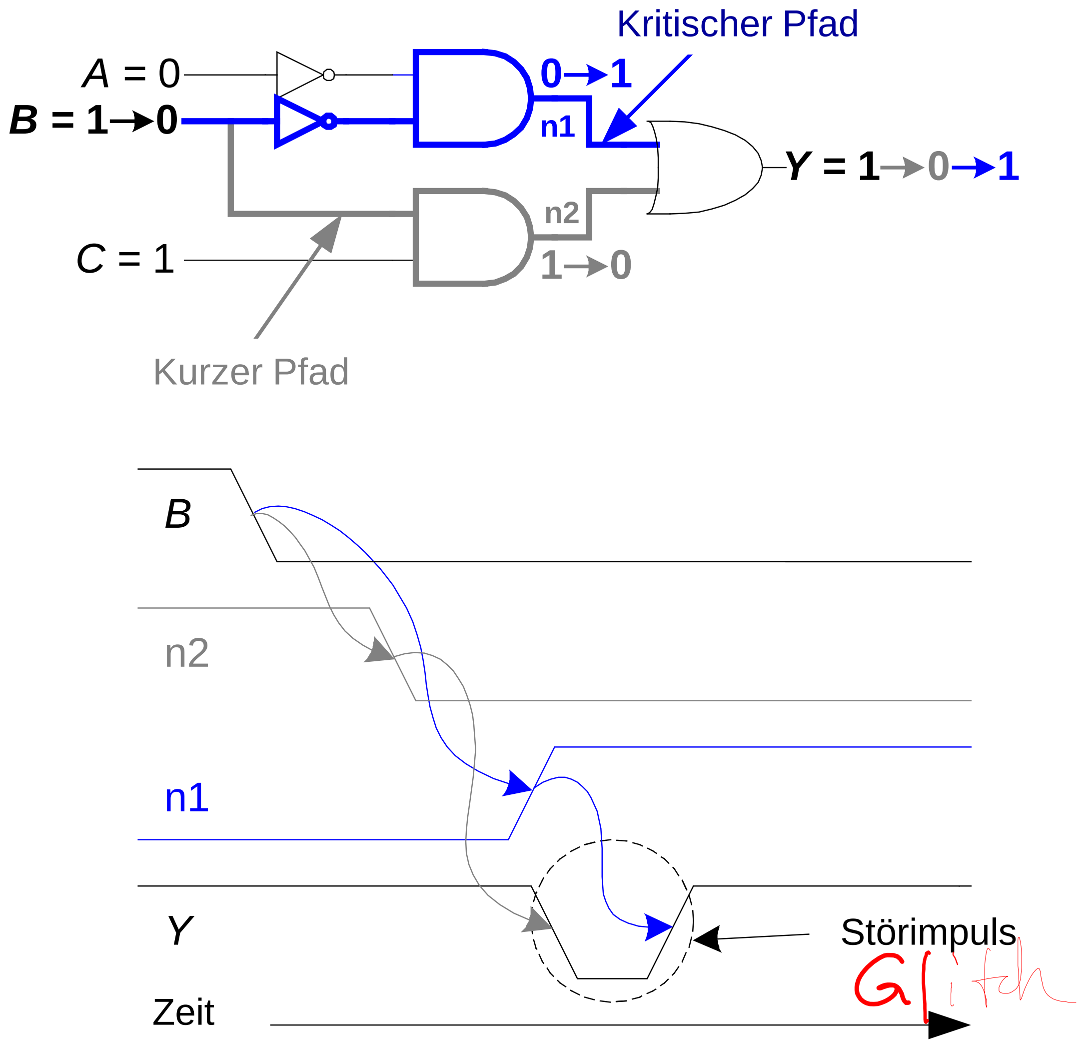
- Was passiert, wenn A = 0, C = 1, und B fällt von 1➔0?



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

$$Y = \overline{A}\,\overline{B} + BC$$

Beispiel für Störimpulse (Fortsetzung)



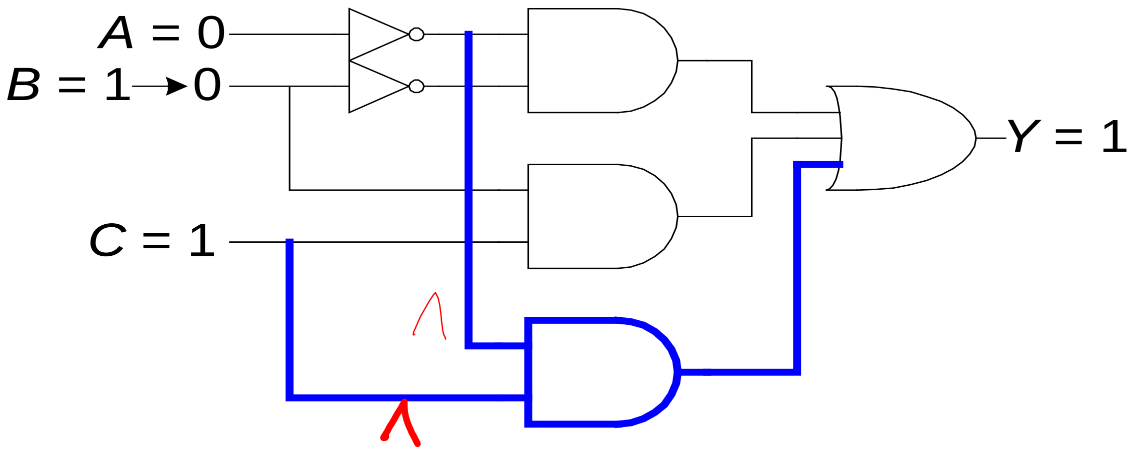
Störimpuls beseitigen

$C \backslash AB$	00	01	11	10
0	1	0	0	0
1	1	1	1	0

$\overline{A}C$

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

Konsensusregel



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

Zusätzliche Beispiele

Vereinfachen einer Formel

Ziel: eine Gleichung mit:

- die kleinste Anzahl von **Implikanten**
- jede Implikant hat die kleinste Anzahl von **Literalen**

Vereinfachen einer Formel

Ziel: eine Gleichung mit:

- die kleinste Anzahl von **Implikanten**
- jede Implikant hat die kleinste Anzahl von **Literalen**

Wiederholung:

– Implikant: Produkte von Literalen

$A\overline{B}C$, $\overline{A}\,\overline{C}$, BC

– Literal: Variable oder ihr Komplement

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

Methoden der Vereinfachung

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Distributivität (T8, T8')** $B\,(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Absorption (T9')** $A + AP = A$
- **Zusammenfassen (T10)** $PA + P\overline{A} = P$
- **Erweitern** $P = PA + P\overline{A}$
  $A = A + AP$
- **Duplizieren** $A = A + A$
- **"Vereinfachung" Satz** $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

Embedded Systems & Applications

Beweisung des Vereinfachungssatzes

$$PA + \overline{A} = P + \overline{A}$$

Beweisung Methode 1:

$PA + \overline{A}$	$= PA + (\mathbf{\overline{A} + \overline{A}P})$	**T9’**
	$= PA + \mathbf{P\overline{A}} + \overline{A}$	**T6**
	$= \mathbf{P(A + \overline{A})} + \overline{A}$	**T8**
	$= P\mathbf{(1)} + \overline{A}$	**T5’**
	$= P + \overline{A}$	**T1**

Beweisung des Vereinfachungssatzes

$$PA + \overline{A} = P + \overline{A}$$

Beweisung Methode 2:

$PA + \overline{A}$	$= (A + \overline{A})\,(P + \overline{A})$	**T8'**
	$= 1(\overline{A} + P)$	**T5'**
	$= \mathbf{\overline{A} + P}$	**T1**

Konsensusregel

Nr.	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = (B \bullet C) + (\overline{B} \bullet D)$	Konsensusregel

Beweisung:

$$B \bullet C + \overline{B} \bullet D + C \bullet D$$

$= BC + \overline{B}D + (CDB + CD\overline{B})$	**T10**
$= BC + \overline{B}D + BCD + \overline{B}CD$	**T6**
$= BC + BCD + \overline{B}D + \overline{B}CD$	**T6**
$= (BC + BCD) + (\overline{B}D + \overline{B}CD)$	**T7**
$= BC + \overline{B}D$	**T9'**

Methoden der Vereinfachung

• **Distributivität (T8, T8')**	$B\,(C+D) = BC + BD$
	$B + CD = (B+C)(B+D)$
• **Absorption (T9')**	$A + AP = A$
• **Zusammenfassen (T10)**	$PA + P\overline{A} = P$
• **Erweitern**	$P = PA + P\overline{A}$
	$A = A + AP$
• **Duplizieren**	$A = A + A$
• **"Vereinfachung" Satz**	$P\overline{A} + A = P + A$
	$PA + \overline{A} = P + \overline{A}$

Vereinfachen bei Zusammenfassen

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

$Y = A$ T10

oder:

$= A(B + \overline{B})$ T8

$= A(1)$ T5'

$= A$ T1

Methoden der Vereinfachung

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Distributivität (T8, T8')** — $B(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Absorption (T9')** — $A + AP = A$
- **Zusammenfassen (T10)** — $PA + P\overline{A} = P$
- **Erweitern** — $P = PA + P\overline{A}$
  $A = A + AP$
- **Duplizieren** — $A = A + A$
- **"Vereinfachung" Satz** — $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

Embedded Systems & Applications

Vereinfachung von Gleichungen

TECHNISCHE UNIVERSITÄT DARMSTADT

$$Y = A(AB + ABC)$$

$= A(AB(1 + C))$	T8
$= A(AB(1))$	T2’
$= A(AB)$	T1
$= (AA)B$	T7
$= AB$	T3

Methoden der Vereinfachung

- **Distributivität (T8, T8')** $B(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Absorption (T9')** $A + AP = A$
- **Zusammenfassen (T10)** $PA + P\overline{A} = P$
- **Erweitern** $P = PA + P\overline{A}$
  $A = A + AP$
- **Duplizieren** $A = A + A$
- **"Vereinfachung" Satz** $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

Vereinfachung von Gleichungen

$Y = A'BC + A'$	**Recall: $A' = \overline{A}$**
$= A'$	T9' Absorption: $X + XY = X$
oder	
$= A'(BC + 1)$	T8
$= A'(1)$	T2'
$= A'$	T1

Methoden der Vereinfachung

- **Distributivität (T8, T8')** — $B\,(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Absorption (T9')** — $A + AP = A$
- **Zusammenfassen (T10)** — $PA + P\overline{A} = P$
- **Erweitern** — $P = PA + P\overline{A}$
  $A = A + AP$
- **Duplizieren** — $A = A + A$
- **"Vereinfachung" Satz** — $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

Vereinfachung von Gleichungen

Y = AB’C + ABC + A’BC

= AB’C + **ABC + ABC** + A’BC	T3’
= (AB’C+ABC) + (ABC+A’BC)	T7’
= AC + BC	T10

Methoden der Vereinfachung

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Distributivität (T8, T8')** $B\,(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Absorption (T9')** $A + AP = A$
- **Zusammenfassen (T10)** $PA + P\overline{A} = P$
- **Erweitern** $P = PA + P\overline{A}$
  $A = A + AP$
- **Duplizieren** $A = A + A$
- **"Vereinfachung" Satz** $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

Embedded Systems & Applications

Vereinfachung von Gleichungen

TECHNISCHE UNIVERSITÄT DARMSTADT

$Y = AB + BC + B'D' + AC'D'$

Methode 1:

$Y = AB + BC + B'D' + (ABC'D' + AB'C'D')$	T10
$= (AB + ABC'D') + BC + (B'D' + AB'C'D')$	T6, T7
$= AB + BC + B'D'$	T9

Methode 2:

$Y = \mathbf{AB} + BC + \mathbf{B'D}' + AC'D' + \mathbf{AD'}$	T11
$= AB + BC + B'D' + AD'$	T9
$= AB + BC + B'D'$	T11

Methoden der Vereinfachung

- **Distributivität (T8, T8')** — $B\,(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Absorption (T9')** — $A + AP = A$
- **Zusammenfassen (T10)** — $PA + P\overline{A} = P$
- **Erweitern** — $P = PA + P\overline{A}$
  $A = A + AP$
- **Duplizieren** — $A = A + A$
- **"Vereinfachung" Satz** — $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

Vereinfachung von Gleichungen

TECHNISCHE UNIVERSITÄT DARMSTADT

$Y = (A + BC)(A + DE)$

T8' erst verwenden, wenn möglich: $W+XZ = (W+X)(W+Z)$

$X = BC$, $Z = DE$:

$$
\begin{aligned}
Y &= (A+X)(A+Z) \\
&= A + XZ && \text{T8'} \\
&= A + BCDE
\end{aligned}
$$

oder

$$
\begin{aligned}
Y &= AA + ADE + ABC + BCDE && \text{T8} \\
&= A + ADE + ABC + BCDE && \text{T3} \\
&= \mathbf{A + ADE} + ABC + BCDE \\
&= \mathbf{A} \qquad\qquad + ABC + BCDE && \text{T9'} \\
&= A + BCDE && \text{T9'}
\end{aligned}
$$

Embedded Systems & Applications

Vereinfachung von Gleichungen

TECHNISCHE UNIVERSITÄT DARMSTADT

$Y = (A + C + D + E)(A + B)$

T8' erst verwenden, wenn möglich: $W+XZ = (W+X)(W+Z)$

$X = (C+D+E)$, $Z = B$

$$
\begin{aligned}
Y &= (A+X)(A+Z) && \\
&= A + XZ && \text{T8'} \\
&= A + (C+D+E)B && \\
&= A + BC + BD + BE && \text{T8}
\end{aligned}
$$

oder

$$
\begin{aligned}
Y &= AA+AB+AC+BC+AD+BD+AE+BE && \text{T8} \\
&= \mathbf{A}+AB+AC+AD+AE+BC+BD+BE && \text{T3} \\
&= A + BC + BD + BE && \text{T9'}
\end{aligned}
$$