

# Digitaltechnik

Wintersemester 2024/2025

Vorlesung 6



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



**ENCRYPTO**  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

## 1 Karnaugh Diagramme

## 2 Algorithmische Logikminimierung

## 3 Zeitverhalten

## 4 Mehrwertige Logik



Harris 2016  
Kap. 2.6 - 2.10

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

Abgabefrist für Hausaufgabe B zu  
Vorlesungen 03 und 04 **diese** Woche  
Freitag 23:59!  
Wöchentliches Moodle-Quiz nicht vergessen!

## 1 Karnaugh Diagramme

## 2 Algorithmische Logikminimierung

## 3 Zeitverhalten

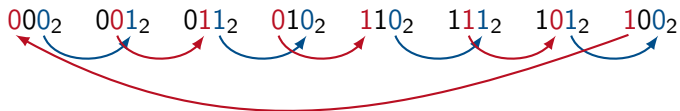
## 4 Mehrwertige Logik

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

- Bell Labs
  - IBM Research
  - Techniken und Methoden für den schnellen Entwurf informationstechnischer Systeme
- ⇒ Karnaugh(-Veitch) Diagramme



- Aufzählung von Binärzahlen einer festen Breite  $k$ , wobei sich (zyklisch) benachbarte Zahlen um nur ein Bit unterscheiden



- Konstruktion: Graycode für  $k + 1$  aus Graycode für  $k$  mit Prefix 0, dann umgekehrt Graycode für  $k$  mit Prefix 1.

$k = 1$ : 0<sub>2</sub> → 1<sub>2</sub>

$k = 2$ : 00<sub>2</sub> → 01<sub>2</sub> ← 11<sub>2</sub> → 10<sub>2</sub>

$k = 3$ : 000<sub>2</sub> → 001<sub>2</sub> → 011<sub>2</sub> → 010<sub>2</sub> ← 110<sub>2</sub> ← 111<sub>2</sub> ← 101<sub>2</sub> → 100<sub>2</sub>

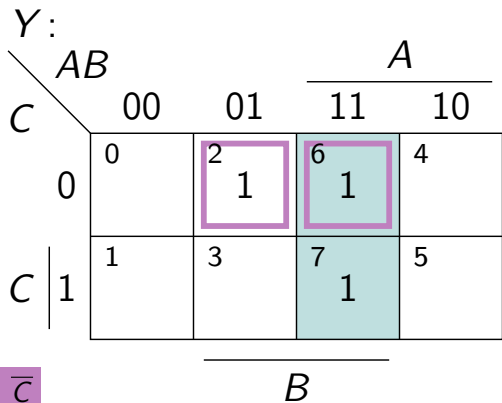
- boole'sche Ausdrücke können durch Zusammenfassen von Mintermen minimiert werden
  - $Y = A B + A \bar{B} = A$
- Karnaugh Diagramme stellen Zusammenhänge graphisch dar
  - Anordnung der Wahrheitstabelle via Graycode
  - ⇒ Zusammenhängende Minterme besser erkennbar

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

Y:

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

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



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



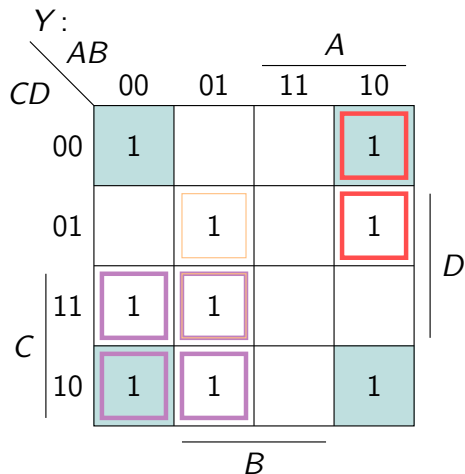


- $n$  Eingangsvariablen
- Implikant aus  $k \leq n$  Literalen deckt  $2^{n-k}$  Minterme ab
- Primimplikant
  - nicht vergrößerbare zusammenhängende viereckige Fläche im Karnaugh Diagramm
  - *Achtung:* Umbruch an Rändern beachten

# Karnaugh Diagram mit vier Eingängen



ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING



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

# Karnaugh Diagram mit „Don't Cares“

LQ4-4

RQ4-4



ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

Y:

CD \ AB	A			
	00	01	11	10
00	1		*	1
01		*	*	1
11	1	1	*	*
10	1	1	*	*

$D$

$B$

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












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

## 1 Karnaugh Diagramme

## 2 Algorithmische Logikminimierung

## 3 Zeitverhalten

## 4 Mehrwertige Logik

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

- Algebraisch:
  - Umformen nach Axiomen/Theoremen
- Grafisch:
  - Karnaugh Diagramme
- Algorithmisch
  - exakt: Quine-McCluskey
  - heuristisch: Espresso

⇒ Minimiere Anzahl der zur Darstellung einer Funktion notwendigen Implikanten



- Grafische Verfahren:
  - für viele ( $> 6$ ) Eingänge nicht mehr praktikabel
  - keine Optimierung zwischen Ausdrücken für mehrere Ausgänge
- Quine-McCluskey-Methode
  - berechnet zunächst *alle* möglichen Implikanten
  - ermittelt *danach* minimale Teilmenge für vollständige Überdeckung
  - ⇒ Rechenzeit steigt exponentiell in der Anzahl der Eingänge
- ⇒ für wirklich große Probleme ( $> 50$  Variablen) nur Heuristiken sinnvoll
  - geringere Laufzeitkomplexität
  - geringere Lösungsqualität



- in 1980er Jahren von IBM und UC Berkeley entwickelt
- unterstützt auch mehrere (zusammen optimierte) Ausgänge
- Details des Algorithmus hier nicht relevant (Buch v. Katz 2005, sowie Rudell 1986 “Multiple-Valued Logic Minimization for PLA Synthesis”)
- hier nur Anwendung einer konkreten Implementierung
  - <https://embedded.eecs.berkeley.edu/pubs/downloads/espresso>
  - spezielles Dateiformat für boole'sche Funktionen
  - erlaubt auch exakte Minimierung (als Referenz für Heuristik):  
`espresso -D exact input.esp > output.esp`  
`espresso -D ESPRESSO input.esp > output.esp`





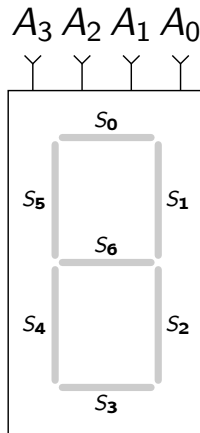
xor.esp

```
1 # Y = A xor B
2 .i    2    # Anzahl Eingänge
3 .o    1    # Anzahl Ausgänge
4 00 0      # Ausgang 0 optional
5 01 1
6 10 1
7 11 0      # Ausgang 0 optional
```



- jede Zeile beschreibt einen Implikanten mit  $n_i$  Zeichen ...
  - 0 Eingang negiert im Implikanten
  - 1 Eingang nicht-negiert im Implikanten
    - Eingang nicht im Implikanten (kein Minterm)
- ... und  $n_o$  Ausgangsfunktionen mit je einem Zeichen
  - 0 Implikant im off set des Ausgangs (optional)
  - 1 Implikant im on set des Ausgangs
    - Implikant im on set *oder* off set des Ausgangs (Don't Care)

- (Typ.) vier Eingänge für dargestellte Ziffer
  - Sieben *unabhängig* schaltbare Segmente  $S_0, \dots, S_6$
- ⇒ jedes Segment nur für bestimmte Zeichen aktiv

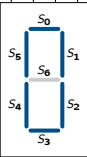
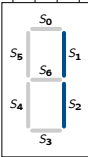

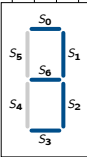
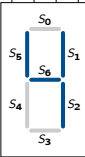



# 7-Segment Anzeige

## Wahrheitstabelle



**ENCRYPTO**  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

$A_3$	$A_2$	$A_1$	$A_0$	$S_0$	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	1	1	1	1	1	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0																																																																																																																																																																																																																																																																																																																																																																																																																																	
0	0	1	0	1	1	0	1	1	0	1	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	1	1	1	1	1	1	0	0	1	1																																																																																																																																																																																																																																																																																																																																																																																																																																																				

# 7-Segment Anzeige in Espresso Eingabedateien



ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

sevenseg/s0.esp

```
1 # S0 of 7-segment display
2 .i 4
3 .o 1
4 0000 1
5 0010 1
6 0011 1
7 0101 1
8 0110 1
9 0111 1
10 1000 1
11 1001 1
12 1010 -
13 1011 -
14 1100 -
15 1101 -
16 1110 -
17 1111 -
```

sevenseg/all.esp

```
1 # 7-segment display
2 .i 4
3 .o 7
4 0000 1111110
5 0001 0110000
6 0010 1101101
7 0011 1111001
8 0100 0110011
9 0101 1011011
10 0110 1011111
11 0111 1110000
12 1000 1111111
13 1001 1111011
14 1010 -----
15 1011 -----
16 1100 -----
17 1101 -----
18 1110 -----
19 1111 -----
```

espresso -D ESPRESSO sevenseg/s0.esp

```
1 # S0 of 7-segment display
2 .i 4
3 .o 1
4 .p 4
5 -0-0 1
6 1--- 1
7 --1- 1
8 -1-1 1
9 .e
```

$$S_0 = \overline{A_2} \overline{A_0} + A_3 + A_1 + A_2 A_0$$

espresso -D ESPRESSO sevenseg/all.esp










```
1 # 7-segment display
2 .i 4
3 .o 7
4 .p 9
5 -0-0 1001100
6 -0-1 0110000
7 --10 1001100
8 -01- 0101001
9 -1-0 0010011
10 --11 1110000
11 --00 0110010
12 -101 1011011
13 1--- 1001011
14 .e
```

1 Karnaugh Diagramme

2 Algorithmische Logikminimierung

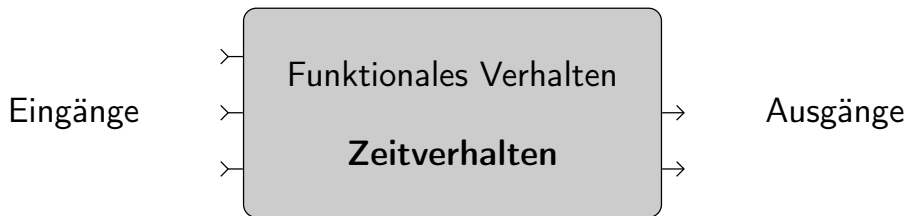
3 Zeitverhalten

4 Mehrwertige Logik

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



- Eingänge
- Ausgänge
- Spezifikation der realisierten (boole'schen) Funktion = Funktionales Verhalten
- Spezifikation des Zeitverhaltens

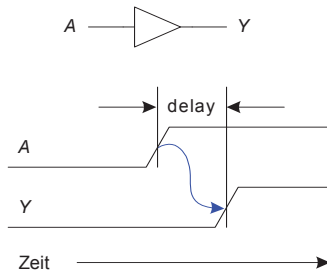




- Kombinatorisch: Werte der Ausgänge hängen nur von Werten an Eingängen ab
- reale Schaltungselemente benötigen aber endliche Zeit, um Änderung am Eingang auf Ausgang zu übertragen
  - z.B. für Umladen von MOSFET Gate-Kapazitäten

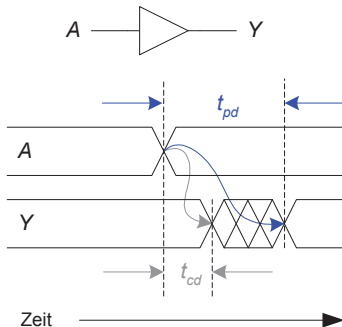
## ⇒ Zentrale Fragen

- Wann sind die Ausgänge stabil?
- Gibt es funktional äquivalente Schaltungen mit geringerer Verzögerung?
- Timing-Analyse anspruchsvoll, denn
  - Eingang kann Ausgang über verschiedene Pfade beeinflussen
  - Verzögerung kann für steigende/fallende Flanken unterschiedlich sein
  - Verzögerungen im (Sub-)Nanosekundenbereich



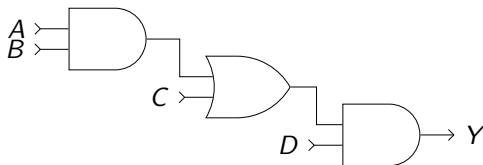
$t_{pd}$  maximale Zeit vom Eingang zum Ausgang (**Ausbreitungsverzögerung**, propagation delay)

$t_{cd}$  minimale Zeit vom Eingang zum Ausgang (**Kontaminationsverzögerung**, contamination delay)





- Ursachen für Verzögerung
  - Kapazitäten, Induktivitäten und Widerstände in der Schaltung
  - Lichtgeschwindigkeit als maximale Ausbreitungsgeschwindigkeit: 30 cm/ns
- Warum können  $t_{pd}$  und  $t_{cd}$  unterschiedlich sein?
  - mehrere Ein- und Ausgänge mit unterschiedlich langen Pfaden
  - unterschiedliche Verzögerungen für steigende ( $t_{pd,LH}$ ) und fallende ( $t_{pd,HL}$ ) Flanken
  - Schaltungen werden
    - langsamer bei Erwärmung (Hitze erhöht Widerstand des leitfähigen Materials)
    - schneller bei Abkühlung



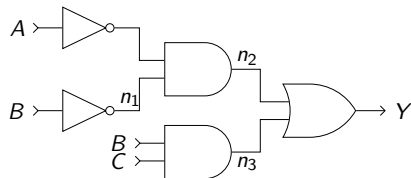
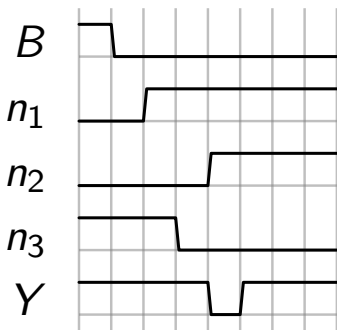
Kritischer Pfad  $t_{pd,Y} = \max(t_{pd,AND} + t_{pd,OR} + t_{pd,AND},$   
 $t_{pd,OR} + t_{pd,AND},$   
 $t_{pd,AND}) = 2t_{pd,AND} + t_{pd,OR}$

Kurzer Pfad  $t_{cd,Y} = \min(t_{cd,AND} + t_{cd,OR} + t_{cd,AND},$   
 $t_{cd,OR} + t_{cd,AND},$   
 $t_{cd,AND}) = t_{cd,AND}$



- eine Änderung eines Eingangs verursacht mehrere Änderungen des Ausgangs
- können durch geeignete Entwurfsdisziplin entschärft werden
  - Ausgänge nur zu bestimmten Zeiten auswerten (synchroner Entwurf)
  - Pfade modifizieren / hinzufügen
  - nicht alle Störimpulse können eliminiert werden  
(z.B. gleichzeitiges Schalten mehrerer Eingänge)
- können durch Timing- und Karnaugh-Diagramme analysiert werden

- Was passiert, wenn  $(A, B, C)$  von  $(0, 1, 1)$  nach  $(0, 0, 1)$  schaltet?
- $t_{pd,NOT/OR} = 1$ ,  $t_{pd,AND} = 2$

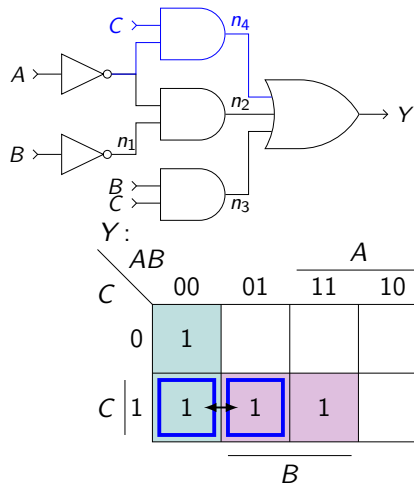
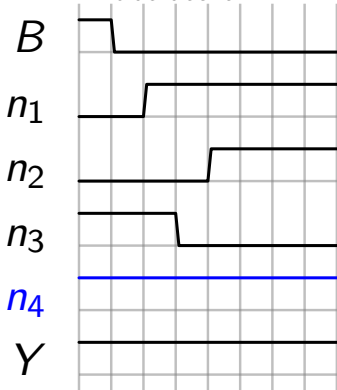


Y:

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

$B$

- Kritische Stelle im Karnaugh-Diagramm mit zusätzlichem Implikanten  $\bar{A} C$  überdecken





- Verzögerung in SystemVerilog: #Zeiteinheiten
  - **Kein synthetisierbarer Code**, eine Verzögerung lässt sich ohne weiteres nicht in Hardware übersetzen
- ⇒ **Nur für Simulation/Tests!**



# Beispiel: Störimpuls

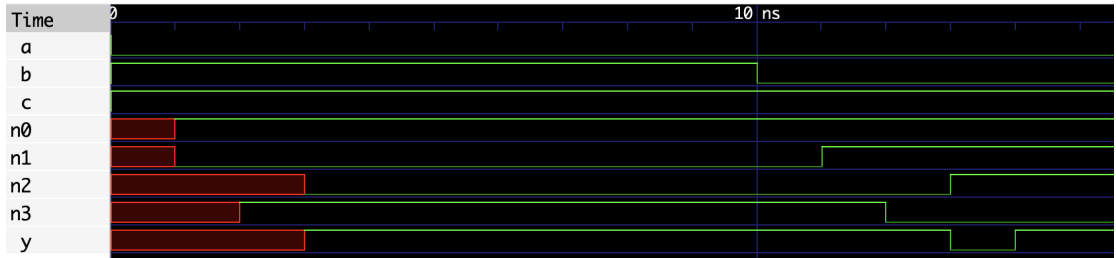


ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

example\_delay.sv

```
1 `timescale 1ns / 10ps // Zeiteinheit / Präzision f. Rundung
2 module example_delay(input logic a, b, c, output logic y);
3     logic n0, n1, n2, n3;
4     assign #1 n0 = ~a;          // Verz. 1 Einheit
5     assign #1 n1 = ~b;
6     assign #2 n2 = n0 & n1; // Verz. 2 Einheiten
7     assign #2 n3 = b & c;
8     assign #1 y = n2 | n3;
9 endmodule
```

Code  
synthetisiert  
nicht!



1 Karnaugh Diagramme

2 Algorithmische Logikminimierung

3 Zeitverhalten

4 Mehrwertige Logik

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



- bisher galt:

- jeder Schaltungsknoten (außer Eingänge) wird von *genau einem* Schaltungselement auf 0 oder 1 getrieben
- Axiome der boole'schen Algebra basieren auf  $\mathbb{B} = \{0, 1\}$

⇒ ignoriert wichtige Teile der Realität

- Wie breiten sich ungültige Spannungen in Schaltung aus?

⇒ Unterscheidung von zwei weiteren Logikwerten neben 0 und 1

X mehrfach getrieben: fehlerhaft

Z ungetrieben/hochohmig (high impedance): gezielt

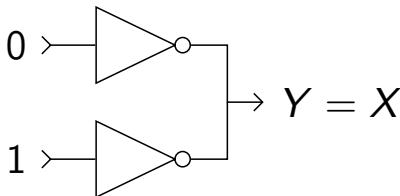
- *Achtung:*

- nicht mit „Don't Care“ (\*) verwechseln
- tatsächliche Spannung *kann* auch im 0- oder 1-Bereich liegen, das Schaltungsdesign stellt dies aber nicht sicher

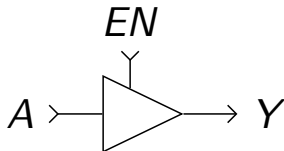
## X (mehrfach getrieben) bei konkurrierenden Ausgängen



- mehrere (unabhängige) Treiber für den selben Schaltungsknoten
  - Konflikt, sobald Treiber in entgegengesetzte Richtung ziehen
    - instabil: abhängig von Betriebsspannung, Temperatur, etc.
    - destruktiv: Kurzschluss verursacht hohen Energieverbrauch
  - fast immer ein Entwurfsfehler
    - z.B. doppelte Zuweisung in Hardwarebeschreibung (Unresolved net/uwire ... cannot have multiple drivers.)
- ⇒ Konflikt-Quelle muss in Simulation leicht nachvollziehbar sein

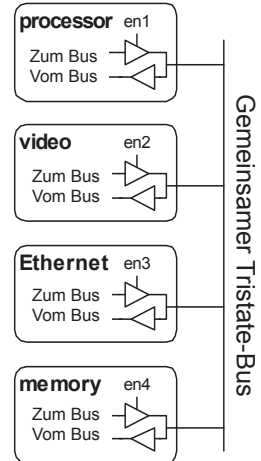


- zusätzliches Enable-Signal EN an Buffer
  - EN=1: Funktion wie normaler Buffer
  - EN=0: Ausgang hochohmig (offen, ungetrieben, floating, high-impedance) Z
- *Achtung:*  $Z \neq 0$

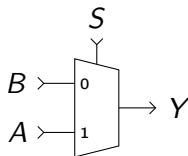
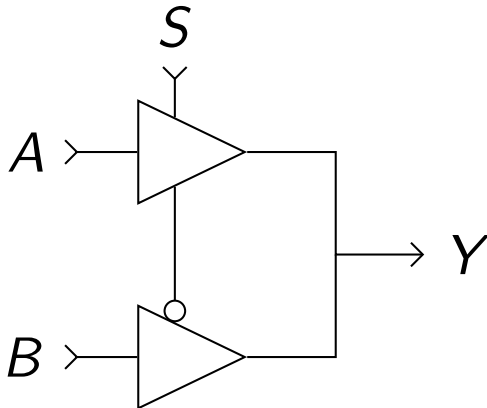


EN	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

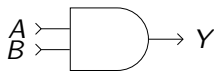
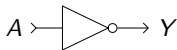
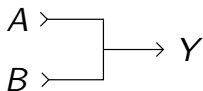
- mehrere Treiber an gemeinsamer Leitung
- zu jedem Zeitpunkt *genau ein* aktiver Treiber
- erlaubt Wechsel der Kommunikationsrichtung



$S$	$A$	$B$	$Y$
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



- Resolutionstabellen definieren Ausbreitung von X (mehrfach getrieben) und Z (hochohmig)
- mehr Konvention (für Simulator) als physikalische Realität
- z.B. IEEE 1164:



A/B	X	0	1	Z
X	X	X	X	X
0	X	0	X	0
1	X	X	1	1
Z	X	0	1	Z

A	Y
X	X
0	1
1	0
Z	X

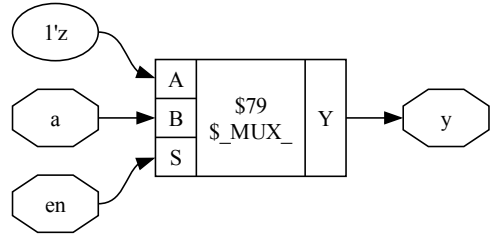
A/B	X	0	1	Z
X	X	0	X	X
0	0	0	0	0
1	X	0	1	X
Z	X	0	X	X

A/B	X	0	1	Z
X	X	X	1	X
0	X	0	1	X
1	1	1	1	1
Z	X	X	1	X



tristate.sv

```
1 module tristate
2   (input  logic a,
3    input  logic en,
4    output logic y);
5
6   assign y = en ? a : 1'bz;
7
8 endmodule
```












- 1 Karnaugh Diagramme
- 2 Algorithmische Logikminimierung
- 3 Zeitverhalten
- 4 Mehrwertige Logik

nächste Vorlesung beinhaltet

- SystemVerilog: Datentypen und kombinatorische Schaltungen
- Arithmetische Grundsaltungen

**Hausaufgabe B zu Vorlesungen 03 und 04 muss bis diese Woche Freitag 23:59 abgegeben werden.**  
**Wöchentliches Moodle-Quiz nicht vergessen!**

Anwendungssoftware		Programme
Betriebssysteme		Gerätetreiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital-schaltungen		UND Gatter Inverter
Analog-schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen