

# Digitaltechnik

## Kapitel 1



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

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

WS 15/16



# Wer bin ich?

- Stanford University  
Ph.D. (≈Doktor)  
Electrical and Computer Engineering  
(2005)
- Harvey Mudd College  
Assistant/Associate Professorin  
(2004-2014)
- University of Nevada, Las Vegas  
Associate Professorin  
(2014 - )



# Wer bin ich?

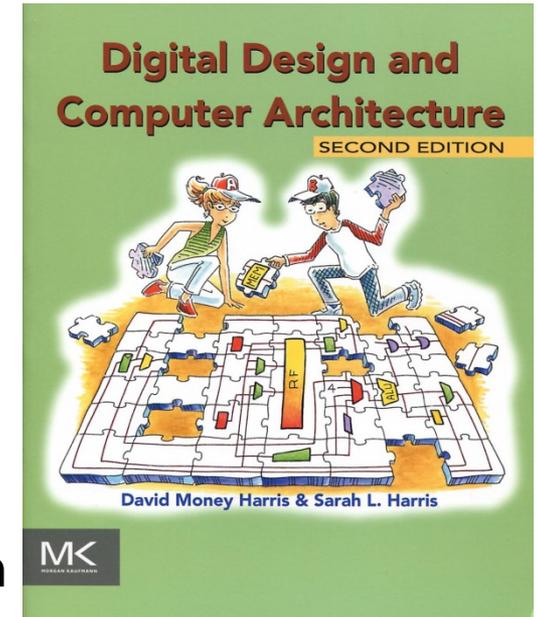
- Technische Universität Darmstadt  
2015 – 2016
- **Fachgebiet:** Eingebettete Systeme, Rechnerarchitekturen
- **Kontakt Informationen:**  
Sprechstunden: Mittwochs, 13:30 Uhr – 14:30 Uhr  
Email: [harris@esa.informatik.tu-darmstadt.de](mailto:harris@esa.informatik.tu-darmstadt.de)  
Büro: S2/02 (Piloty Gebäude), Raum E102

# Lehr- und Anschauungsmaterial

- Aus dem Lehrbuch

## *Digital Design and Computer Architecture, Zweite Auflage*

- Diese Folien nach englischen Originalvorlagen erstellt (Originale sind © 2007 Elsevier)
- Buch wird an Studierende **subventioniert** abgegeben
  - Organisiert durch Fachschaft Informatik
- Mehr **Hintergrundmaterial** auf Web-Seite zu Buch
- Winter Semester: **Kapitel 1 bis 5**



# Organisatorisches

- **Übungen:** Organisatorisches von dem Assistent jetzt erklärt...
  
- **Webseite:** [www.esa.informatik.tu-darmstadt.de](http://www.esa.informatik.tu-darmstadt.de)
  - unter Lehre → WS 2015/2016 → DigitalTechnik
  - Vorlesungsfolien, Aufzeichnungen der Vorlesungen, u.s.w.
  
- **Prüfung:** 1. März 2016

# Lehrphilosophie



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Wenn Sie **Fragen** haben, bitte stellen Sie die mal
- Ich werde Ihnen auch Fragen stellen während der Vorlesung



## Die Grundlagen der Digitaltechnik

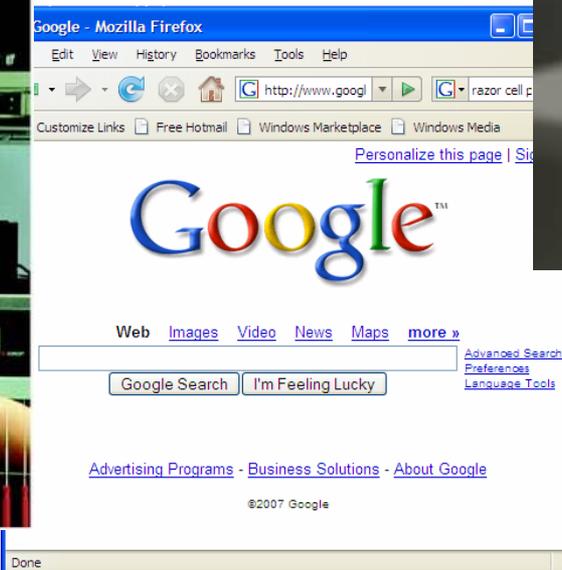
- Hintergrund
- Vorgehensweise
- Beherrschen von Komplexität
- Die digitale Abstraktion
- Zahlensysteme
- Logikgatter
- Darstellung als elektrische Spannungen
- CMOS Transistoren
- Elektrische Leistungsaufnahme

*Heute*

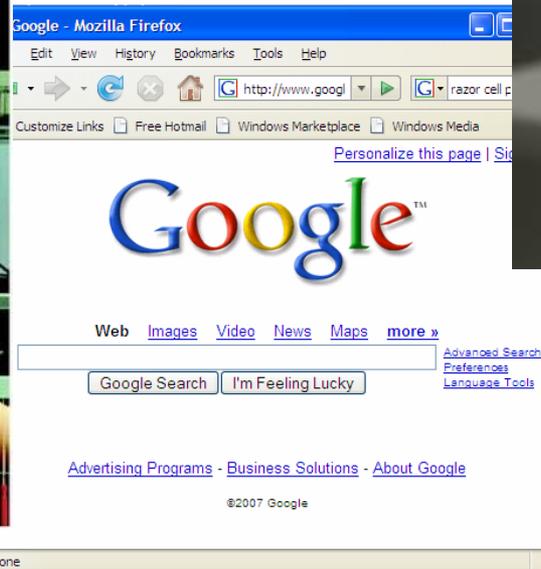


- Wie haben Digitalgeräte die Welt verändert?
- Wie haben Digitalgeräte Ihre Welt verändert?

- Mikroprozessoren haben die Welt verändert
  - Handys, Internet, Medizintechnik, Unterhaltung, ...
- Umsatzwachstum in der Halbleiterindustrie von \$21 Milliarden in 1985 auf \$323 Milliarden in 2015



Als Informatiker/in werden Sie Computer  
von der Pike auf verstehen!



# Themen dieser Veranstaltung

---

- Entwurf digitaler Logikschaltungen
- Systematische Fehlersuche in digitalen Logikschaltungen
- Interner Aufbau und Funktion eines Computers
- Entwurf und Realisierung eines Mikroprozessors (Sommer Semester)

# Beherrschen von Komplexität

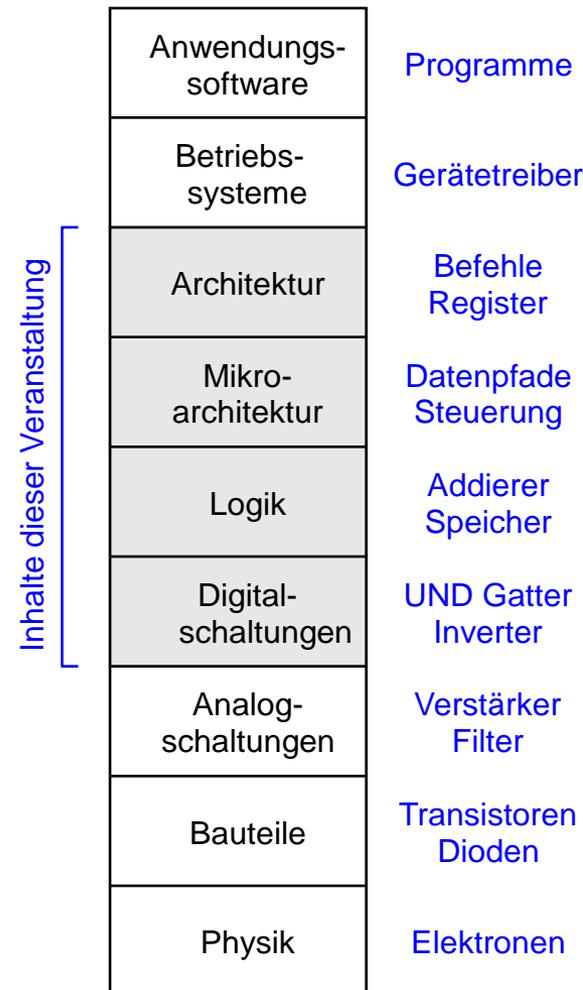


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Abstraktion
- Disziplin
- Wesentliche Techniken (die drei **Y**'s)
  - Hierarchie (*hierarchy*)
  - Modularität (*modularity*)
  - Regularität (*regularity*)

# Abstraktion

- Verstecken unnötiger Details
- „unnötig“
  - Für *diese* spezielle Aufgabe unnötig!
- Für alle Aufgaben hilfreich:
  - Verstehen der **anliegenden** Abstraktionsebenen



- **Wissentliche Beschränkung der Realisierungsmöglichkeiten**
  - Erlaubt produktivere Arbeit auf **höheren** Entwurfsebenen
- **Beispiel: Digitale Entwurfsdisziplin**
  - Arbeite mit **diskreten** statt mit stetigen Spannungspegeln
  - Digitalschaltungen sind **einfacher** zu entwerfen als analoge
    - Erlaubt den Entwurf komplexerer Schaltungen
  - Digitale Systeme **ersetzen** zunehmend analoge
    - Digitalkamera, digitales Fernsehen, moderne Handys, CD, DVD, ...

# Beherrschen von Komplexität



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Abstraktion
- Disziplin
- Wesentliche Techniken (die drei **Y**'s)
  - Hierarchie (*hierarchy*)
  - Modularität (*modularity*)
  - Regularität (*regularity*)

# Wesentliche Techniken (Die Drei-Y's)



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- **Hierarchie** (Hierarchy)
  - Aufteilen eines Systems in Module und Untermodule
- **Modularität** (Modularity)
  - Wohldefinierte Schnittstellen und Funktionen
- **Regularität** (Regularity)
  - Bevorzuge einheitliche Lösungen für einfachere Wiederverwendbarkeit

# Beispiel: Steinschlossgewehr

- Frühes Beispiel für Anwendungen der Drei-Y's
- **Komplexer** Gebrauchsgegenstand
- Entwicklung begann im 16. Jahrhundert
  - Aber noch sehr unzuverlässig
- Höhere Stückzahlen ab dem 17. Jahrhundert
  - Aber alles **Einzelanfertigungen** von Büchsenmachern
- Bis zum 19. Jahrhundert zunehmende Vereinheitlichung



# Hierarchie: Zerlegung in Module

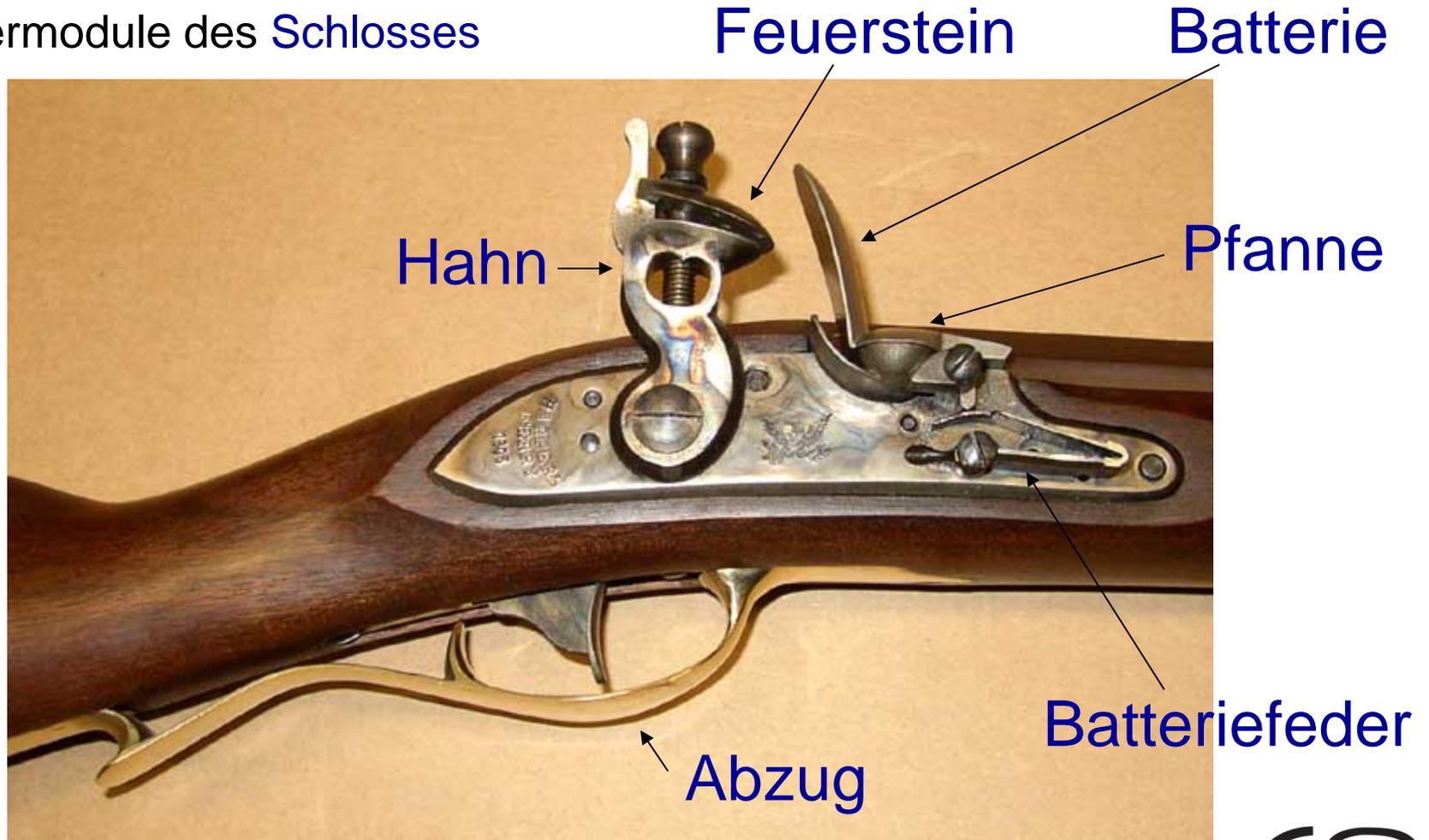


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



# Hierarchie: Zerlegung in Untermodule

- Untermodule des Schlosses



# Modularität: Schaft und Lauf

- Funktion des **Schafts**
  - Schloss und Lauf stabil zusammenfügen
- Funktion des **Laufes**
  - Projektil während Beschleunigung zu führen und mit Drall zu versehen
- Im Idealfall sind Funktionen **unabhängig** und beeinflussen sich nicht
- **Schnittstelle** zwischen Schaft und Lauf
  - Gemeinsame Haltevorrichtung



# Regularität: Austauschbare Teile



- **Gleiche** Schlösser in **unterschiedlichen** Schäften
  - Passender Ausschnitt in Schaft
- **Unterschiedliche** Läufe in **gleichen** Schäften
  - Passende Länge und Haltemechanismus
- Voraussetzung für industrielle **Massenproduktion**

# Beherrschen von Komplexität



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

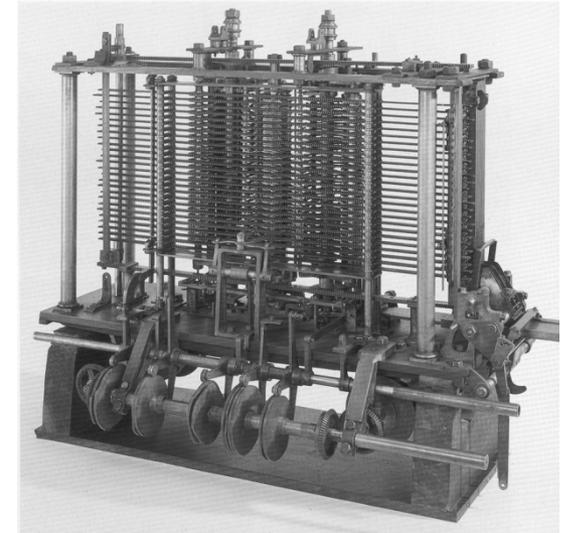
- **Abstraktion**
- Disziplin
- Wesentliche Techniken (die drei Y's)
  - Hierarchie (*hierarchy*)
  - Modularität (*modularity*)
  - Regularität (*regularity*)



- Die meisten **physikalischen** Größen haben **stetige** Werte
  - Elektrische Spannung auf einem Leiter
  - Frequenz einer Schwingung
  - Position einer Masse
- Berücksichtigen alle Werte der Größe (**unendlich** viele)
- Digitale Abstraktion: Berücksichtigt nur **endlich** viele Werte
  - **Untermenge** aus einem stetigen Wertebereich

# Analytische Maschine

- *Analytical engine*
- Entworfen durch **Charles Babbage** von 1834 – 1871
- **Erster Digitalrechner**
- Aufgebaut als **mechanischer** Rechner
  - Zahnstangen und –räder
  - Stellungen repräsentieren **Ziffern 0-9**
    - Genau 10 Stellungen je Zahnrad
- Babbage verstarb vor Fertigstellung
- Entwurf hätte aber **funktioniert**



- Digitale Disziplin
  - In der Regel Beschränkung auf nur **zwei** unterschiedliche Werte
    - Binärsystem
  - Können unterschiedlich heißen
    - **1**, WAHR, TRUE, HIGH, ...
    - **0**, FALSCH, FALSE, LOW, ...
- Unterschiedlichste **Darstellungen** der beiden Werte möglich
  - Spannungspegel, Zahnradstellungen, Flüssigkeitsstände, Quantenzustände, ...
- Digitalschaltungen verwenden üblicherweise unterschiedliche **Spannungspegel**
- **Bit** (**B**inary **dig**it): Maßeinheit für Information
  - 1 b = Eine Ja/Nein-Entscheidung



- Dezimalzahlen
- Binärzahlen
- Hexadezimal



## ▪ Dezimalzahlen

1's column  
10's column  
100's column  
1000's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands      three hundreds      seven tens      four ones

## ▪ Binärzahlen

1's column  
2's column  
4's column  
8's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight      one four      no two      one one

# Zweierpotenzen

▪  $2^0 =$

▪  $2^1 =$

▪  $2^2 =$

▪  $2^3 =$

▪  $2^4 =$

▪  $2^5 =$

▪  $2^6 =$

▪  $2^7 =$

▪  $2^8 =$

▪  $2^9 =$

▪  $2^{10} =$

▪  $2^{11} =$

▪  $2^{12} =$

▪  $2^{13} =$

▪  $2^{14} =$

▪  $2^{15} =$

# Zweierpotenzen



- $2^0 = 1$
  - $2^1 = 2$
  - $2^2 = 4$
  - $2^3 = 8$
  - $2^4 = 16$
  - $2^5 = 32$
  - $2^6 = 64$
  - $2^7 = 128$
  - $2^8 = 256$
  - $2^9 = 512$
  - $2^{10} = 1024$
  - $2^{11} = 2048$
  - $2^{12} = 4096$
  - $2^{13} = 8192$
  - $2^{14} = 16384$
  - $2^{15} = 32768$
- Sehr nützlich, wenigstens die **ersten 10** im **Kopf** zu haben

- **Binär** nach **dezimal** umrechnen:
  - Wandele  $10011_2$  ins Dezimalsystem um
  
- **Dezimal** nach **binär** umrechnen
  - Wandele  $47_{10}$  ins Binärsystem um



# Dezimal nach Binär Umrechnen

- **Auf zwei Arten möglich**
  - **Art 1:** Jeweils nach größter noch passender Zweierpotenz suchen
  - **Art 2:** Durch immer größer werdende Zweierpotenzen dividieren

# Dezimal nach Binär Umrechnen

**Art 1:** Jeweils nach größter noch passender Zweierpotenz suchen

$$53_{10}$$

**Method 2:** Durch immer größer werdende Zweierpotenzen dividieren

$$53_{10} =$$

# Dezimal nach Binär Umrechnen

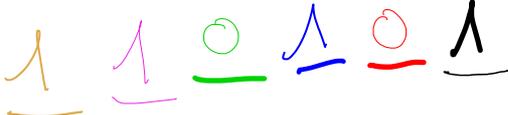
**Art 1:** Jeweils nach größter noch passender Zweierpotenz suchen

$$\begin{array}{r} 53_{10} \\ 53-32 = 21 \\ 21-16 = 5 \\ 5-4 = 1 \end{array} \quad \begin{array}{r} 32 \times 1 \\ 16 \times 1 \\ 4 \times 1 \\ 1 \times 1 \end{array}$$

$$= 110101_2$$

**Method 2:** Durch immer größer werdende Zweierpotenzen dividieren

$$\begin{array}{r} 53_{10} = \\ 53/2 = 26 \quad R 1 \\ 26/2 = 13 \quad R 0 \\ 13/2 = 6 \quad R 1 \\ 6/2 = 3 \quad R 0 \\ 3/2 = 1 \quad R 1 \\ 1/2 = 0 \quad R 1 \end{array}$$



1 1 0 1 0 1

# Dezimal nach Binär Umrechnen

**Art 1:** Jeweils nach größter noch passender Zweierpotenz suchen

$53_{10}$	$32 \times 1$
$53 - 32 = 21$	$16 \times 1$
$21 - 16 = 5$	$4 \times 1$
$5 - 4 = 1$	$1 \times 1$

$$= 110101_2$$

**Method 2:** Durch immer größer werdende Zweierpotenzen dividieren

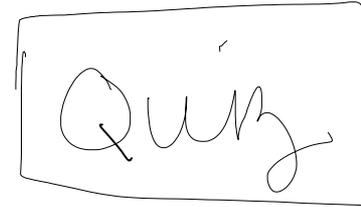
$53_{10} =$	$53/2 = 26$	R1
	$26/2 = 13$	R0
	$13/2 = 6$	R1
	$6/2 = 3$	R0
	$3/2 = 1$	R1
	$1/2 = 0$	R1

$$= 110101_2$$

# Dezimal nach Binär Umrechnen



Noch ein Beispiel:  $75_{10}$  ins Binär umrechnen



# Dezimal nach Binär Umrechnen

Noch ein Beispiel:  $75_{10}$  ins Binär umrechnen

$$75_{10} = 64 + 8 + 2 + 1 = 1001011_2$$

oder

$75/2$	$= 37$	R1
$37/2$	$= 18$	R1
$18/2$	$= 9$	R0
$9/2$	$= 4$	R1
$4/2$	$= 2$	R0
$2/2$	$= 1$	R0
$1/2$	$= 0$	R1

# Binärzahlen und Wertebereiche

## ▪ **N-stellige Dezimalzahl**

- Wie viele verschiedene Werte?  $10^N$
- Wertebereich?  $[0, 10^N - 1]$
- **Beispiel:** 3-stellige Dezimalzahl:
  - $10^3 = 1000$  mögliche Werte
  - Wertebereich:  $[0, 999]$

## ▪ **N-bit Binärzahl**

- Wie viele verschiedene Werte?  $2^N$
- Wertebereich?  $[0, 2^N - 1]$
- **Beispiel:** 3-bit Binärzahl
  - $2^3 = 8$  mögliche Werte
  - Wertebereich :  $[0, 7] = [000_2, 111_2]$

# Hexadezimale Zahlen



Hex-Ziffer	Entspricht Dezimal	Entspricht Binär
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

# Hexadezimale Zahlen



Hex-Ziffer	Entspricht Dezimal	Entspricht Binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

# Hexadezimalzahlen

- Schreibweise zur Basis 16
- Kürzere Darstellung für lange Binärzahlen

# Umwandeln von Hexadezimaldarstellung

- Umwandeln von **hexadezimal** nach **binär**:
  - Wandele  $4AF_{16}$  (auch geschrieben als  $0x4AF$ ) nach binär

$$0100 \quad 1010 \quad 1111 \rightarrow 010010101111$$

- Umwandeln von **hexadezimal** nach **dezimal**:
  - Wandele  $0x4AF$  nach dezimal

$$16^2 \quad 16^1 \quad 16^0$$

$$4 \times 16^2 + 10 \times 16^1 + 15 \times 16^0$$

# Umwandeln von Hexadezimaldarstellung



- Umwandeln von **hexadezimal** nach **binär**:
  - Wandele  $4AF_{16}$  (auch geschrieben als  $0x4AF$ ) nach binär
  - $0100\ 1010\ 1111_2$
- Umwandeln von **hexadezimal** nach **dezimal**:
  - Wandele  $0x4AF$  nach dezimal
  - $16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$



- Dezimal                      Basis 10
- Binär                        Basis 2
- Hexadezimal              Basis 16

Sonstige:

- Oktal
- Andere Basen

# Bits, Bytes, Nibbles...

- Bits (Einheit *b*)
  - Höchstwertiges Bit (*msb*)
  - Niedrigstwertiges Bit (*lsb*)
- Bytes (Einheit *B*) & Nibbles
- Bytes
  - Höchstwertiges Byte (*MSB*)
  - Niedrigstwertiges Byte (*LSB*)

10010110

most significant bit      least significant bit

byte

10010110

nibble

CEBF9AD7

most significant byte      least significant byte

# Zweierpotenzen und Präfixe

- $2^{10} = 1$  Kilo (K)  $\approx 1000$  (1024)
- $2^{20} = 1$  Mega (M)  $\approx 1$  Million (1,048,576)
- $2^{30} = 1$  Giga (G)  $\approx 1$  Milliarde (1,073,741,824)
  
- Beispiele
  - 4 GB: Maximal adressierbare Speichergröße für 32b-Prozessoren
  - 16M x 32b: erste GDDR5-Speicherchips für Grafikkarten
  
- Vorsicht Falle:
  - Deutsch  $10^9=1$  Milliarde
  - US English  $10^9=1$  *billion*

# Zweierpotenzen schnell schätzen

- Was ist der Wert von  $2^{24}$ ?
- Wie viele verschiedene Werte kann eine 32b Variable annehmen?

# Zweierpotenzen schnell schätzen

- Was ist der Wert von  $2^{24}$ ?

$$2^4 \times 2^{20} \approx 16 \text{ Millionen}$$

- Wie viele verschiedene Werte kann eine 32b Variable annehmen?

$$2^2 \times 2^{30} \approx 4 \text{ Milliarden}$$

Whiteboard und Quiz:

$$\begin{array}{l} 375 /_{10} = 37 \quad R \underline{5} \\ 37 /_{10} = 3 \quad R \underline{7} \\ 3 /_{10} = 0 \quad R \underline{3} \end{array}$$

$$\underline{3} \quad \underline{7} \quad | 5$$

Quiz 2: Wie viele Werte können wir mit 7 Bits darstellen?

Antwort:  $2^7 = 128$

# Addition

## ▪ Dezimal

$$\begin{array}{r}
 3734 \\
 + 5168 \\
 \hline
 8902
 \end{array}$$

Handwritten notes for decimal addition:  
 - A green '1' with an arrow points to the carry from the tens place to the hundreds place.  
 - A red '1' with an arrow points to the carry from the hundreds place to the thousands place.  
 - A calculation:  $4 + 8 = 12$  with '10' and '2' circled. Below it, '10' is written under '10' and '1' is written under '2'.  
 - A green '1' is written above the '9' in the result.

## ▪ Binär

$$\begin{array}{r}
 1011 \\
 + 0011 \\
 \hline
 1110
 \end{array}$$

Handwritten notes for binary addition:  
 - To the left:  $8 + 2 + 1 = 11 \rightarrow$  and  $\frac{3}{14} \rightarrow$   
 - To the right:  $1 + 1 = 10$  with '1' and '0' circled. Below it, '1' is written under '1' and '0' is written under '0'.  
 - Below that:  $1 + 1 + 1 = 11$  with '1' and '1' circled.  
 - A blue box highlights the result '1110'.

# Addition

- Dezimal

$$\begin{array}{r} 11 \leftarrow \text{Überträge} \\ 3734 \\ + 5168 \\ \hline 8902 \end{array}$$

- Binär

$$\begin{array}{r} 11 \leftarrow \text{Überträge} \\ 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

# Beispiele für Addition von Binärzahlen

- Addiere die 4-bit Binärzahlen

$$\begin{array}{r} \phantom{+} 1001 \\ + 0101 \\ \hline 1110 \end{array} \quad \begin{array}{r} 9 \\ 5 \\ \hline 14 \end{array}$$

- Addiere die 4-bit Binärzahlen

$$\begin{array}{r} \phantom{+} 1011 \\ + 0110 \\ \hline \end{array} \quad \begin{array}{r} 11_{10} \\ 6_{10} \\ \hline \end{array}$$

Überlauf →  $1 \underbrace{0001}_{4\text{-bit}}$

# Beispiele für Addition von Binärzahlen

- Addiere die 4-bit Binärzahlen

$$\begin{array}{r} 1 \\ 1001 \\ + 0101 \\ \hline 1110 \end{array}$$

- Addiere die 4-bit Binärzahlen

$$\begin{array}{r} 111 \\ 1011 \\ + 0110 \\ \hline 10001 \end{array}$$

Überlauf!

- Digitale Systeme arbeiten mit einer **festen** Anzahl an Bits
  - In der Regel, es gibt aber durchaus Ausnahmen!
- Eine Addition **läuft über**, wenn ihr Ergebnis nicht mehr in die verfügbare Anzahl von Bits hineinpasst
- Beispiel: 11+6, gerechnet mit 4b Breite

# Darstellung von Anzahlen



- Wir haben von positiven Anzahlen geredet.
- Wozu mit den negativen Zahlen?

# Vorzeichenbehaftete Binärzahlen



1 Bit  
↓  
- 3

- Darstellung als Vorzeichen und Betrag
- Zweierkomplement ←

# Darstellung als Vorzeichen und Betrag

- 1 Vorzeichenbit,  $N-1$  Bits für Betrag
- Vorzeichenbit ist höchstwertiges Bit (msb)
  - Positive Zahl: Vorzeichenbit = 0
  - Negative Zahl: Vorzeichenbit = 1

# Darstellung als Vorzeichen und Betrag

- 1 Vorzeichenbit,  $N-1$  Bits für Betrag
- Vorzeichenbit ist höchstwertiges Bit (msb)
  - Positive Zahl: Vorzeichenbit = 0
  - Negative Zahl: Vorzeichenbit = 1
- **Beispiel:** 4-bit Vorzeichen/Betrag-Darstellung von  $\pm 6$ :

$$+6 = \underline{0} \ 1 \ 1 \ 0$$

$$-6 = \underline{1} \ 1 \ 1 \ 0$$

# Darstellung als Vorzeichen und Betrag

- 1 Vorzeichenbit,  $N-1$  Bits für Betrag
- Vorzeichenbit ist höchstwertiges Bit (msb)
  - Positive Zahl: Vorzeichenbit = 0
  - Negative Zahl: Vorzeichenbit = 1
- **Beispiel:** 4-bit Vorzeichen/Betrag-Darstellung von  $\pm 6$ :

$$+6 = \mathbf{0110}$$

$$-6 = \mathbf{1110}$$

- Wertebereich einer Zahl in Vorzeichen/Betrag-Darstellung:

4-Bit:

$$[-7, 7]$$

$$\left[ -(2^{N-1}), 2^{N-1} - 1 \right]$$

# Darstellung als Vorzeichen und Betrag

- 1 Vorzeichenbit,  $N-1$  Bits für Betrag
- Vorzeichenbit ist höchstwertiges Bit (msb)
  - Positive Zahl: Vorzeichenbit = 0
  - Negative Zahl: Vorzeichenbit = 1
- **Beispiel:** 4-bit Vorzeichen/Betrag-Darstellung von  $\pm 6$ :
  - +6 = **0110**
  - 6 = **1110**
- Wertebereich einer Zahl in Vorzeichen/Betrag-Darstellung :  
 **$[-(2^{N-1}-1), 2^{N-1}-1]$**

# Darstellung als Vorzeichen und Betrag

- 1 Vorzeichenbit,  $N-1$  Bits für Betrag

- Vorzeichenbit ist höchstwertiges Bit (msb)

- Positive Zahl: Vorzeichenbit = 0
- Negative Zahl: Vorzeichenbit = 1

- **Beispiel:** 4-bit Vorzeichen/Betrag-Darstellung von  $\pm 6$ :

$$+6 = \mathbf{0110} \leftarrow$$

$$-6 = \mathbf{1110} \leftarrow$$

- Wertebereich einer Zahl in Vorzeichen/Betrag-Darstellung :

$$[-(2^{N-1}-1), 2^{N-1}-1]$$

$$\rightarrow A: \{ \overset{0}{a_{N-1}}, a_{N-2}, \dots, a_2, a_1, a_0 \}$$

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i 2^i$$

$$\left( (-1)^0 \left( - + 0 \cdot 2 + 1 \cdot 1 \right) \right)$$

$$\left( (-1)^1 \cdot \left( \dots \right) \right)$$

# Darstellung als Vorzeichen/Betrag: Probleme

- Addition schlägt **fehl**
  - Beispiel:  $-6 + 6$ :

$$\begin{array}{r} 1110 \\ + 0110 \\ \hline 10100 \text{ (falsch!)} \end{array}$$


- **Zwei** Darstellungen für Null ( $\pm 0$ ):

$$\begin{array}{ll} 1000 & (-0) \\ 0000 & (+0) \end{array}$$

# Zahldarstellung im **Zweierkomplement**

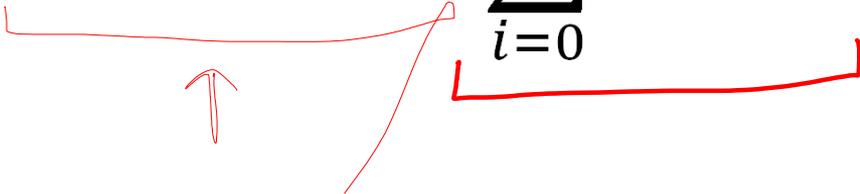
- **Behebt** Probleme der Vorzeichen/Betrag-Darstellung
  - Addition liefert wieder **korrekte** Ergebnisse
  - Nur **eine** Darstellung für Null

# Zahldarstellung im Zweierkomplement



- Wie **vorzeichenlose** Binärdarstellung, aber ...

- msb hat nun einen Wert von  $-2^{N-1}$

$$A = a_{N-1}(-2^{N-1}) \sum_{i=0}^{N-2} a_i 2^i$$


# Zahlendarstellung im Zweierkomplement

- Wie **vorzeichenlose** Binärdarstellung, aber ...

- msb hat nun einen Wert von  $-2^{N-1}$

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i 2^i$$

8 4 2 1

- Höchst** positive 4b Zahl

: 0111

8 4 2 1

- Niedrigste** negative 4b Zahl

: 1000

# Zahldarstellung im Zweierkomplement

- Wie **vorzeichenlose** Binärdarstellung, aber ...

- msb hat nun einen Wert von  $-2^{N-1}$

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i 2^i$$

- Höchst positive 4b Zahl : **0111** =  $2^2 + 2^1 + 2^0 = 7$
- Niedrigste negative 4b Zahl : **1000** =  $-2^3 = -8$
- msb gibt immer noch das **Vorzeichen** an

- 1=negativ, 0=positiv

$\underline{1}111 = -8 + 4 + 2 + 1 = -1$

# Zahldarstellung im Zweierkomplement

- Wie **vorzeichenlose** Binärdarstellung, aber ...

- msb hat nun einen Wert von  $-2^{N-1}$

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i 2^i$$

- Höchst positive 4b Zahl : **0111** =  $2^2 + 2^1 + 2^0 = 7$

- Niedrigste negative 4b Zahl : **1000** =  $-2^3 = -8$

- msb gibt immer noch das **Vorzeichen** an

- 1=negativ, 0=positiv

$$\begin{array}{cccc} a_3 & a_2 & a_1 & a_0 \\ 1 & 0 & 0 & 1 \\ \hline -8 & & & 1 \end{array} \quad 1(-2^3) + 1$$

$$-8 + 1$$

- Wertebereich einer  $N$ -bit Zweierkomplementzahl:

$$[-(2^{N-1}), 2^{N-1}-1]$$

$$[-2^{4-1}, 2^3-1]$$

$$[-8, 7]$$

# Darstellung im Zweierkomplement

- **Annahme:** Umzuwandelnde Zahlen liegen im Wertebereich
  - $N$  bit breites Zweierkomplement
  - Stelle Wert  $k$  im Zweierkomplement  $z$  dar

- **Positive Zahlen:**  $k \geq 0$

∩.

5 bit  
0 1 0 0 1

9<sub>10</sub>

- **Negative Zahlen:**  $k < 0$

5-Bits

-3

$$\begin{array}{r} 16 \\ - 3 \\ \hline 13 \end{array}$$

$$\begin{array}{r} 1 \quad 1 \quad 1 \quad 0 \quad 1 \\ \hline -16 \quad 8 \quad 4 \quad 2 \quad 1 \end{array}$$

# Darstellung im Zweierkomplement

- **Annahme:** Umzuwandelnde Zahlen liegen im Wertebereich
  - $N$  bit breites Zweierkomplement
  - Stelle Wert  $k$  im Zweierkomplement  $z$  dar
- **Positive Zahlen:**  $k \geq 0$ 
  - Normale Binärdarstellung, restliche Bits bis einschließlich **msb** mit 0 auffüllen
  - Beispiel:  $N = 5b$ ,  $k = 3_{10} \rightarrow z = 00011$
- **Negative Zahlen:**  $k < 0$

# Darstellung im Zweierkomplement

- **Annahme:** Umzuwandelnde Zahlen liegen im Wertebereich
  - $N$  bit breites Zweierkomplement
  - Stelle Wert  $k$  im Zweierkomplement  $z$  dar
- **Positive Zahlen:**  $k \geq 0$ 
  - Normale Binärdarstellung, restliche Bits bis einschließlich **msb** mit 0 auffüllen
  - Beispiel:  $N = 5b$ ,  $k = 3_{10} \rightarrow z = 00011$
- **Negative Zahlen:**  $k < 0$ 
  - **msb** auf 1 setzen, Wert soweit ist nun  $-2^{N-1}$
  - Nun muss aufaddiert werden, bis gewünschter Zielwert  $k$  erreicht
    - Differenz  $d = 2^{N-1} + k$
    - Diese Differenz binär in untere Bits eintragen (Beginn bei lsb)
    - Beispiel:  $N = 5b$ ,  $k = -3_{10} \rightarrow d = 2^4 - 3 = 16 - 3 = 13 \rightarrow z = 11101$

# Zweierkomplement arithmetisch bilden

- In **beide** Richtungen anwendbar

- Vorzeichenwechsel:  $k \xrightarrow{\leftarrow} -k$

- **Algorithmus**

1. Alle Bits invertieren ( $0 \rightarrow 1, 1 \rightarrow 0$ )
2. Dann 1 addieren

- **Beispiel:** Vorzeichenwechsel von  $3_{10} = 00011_2$

$$\begin{array}{r} 1. \quad \cancel{1} \cancel{0} \cancel{0} \cancel{0} \cancel{1} \cancel{1} \quad 00011 \\ 2. \quad + \quad 1 \\ \hline 11101 \end{array}$$

$$\begin{array}{r} k \\ -k \\ -3 \end{array}$$

- **Beispiel:** Vorzeichenwechsel von  $-3_{10} = 11101_2$

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ \hline = 1 \quad 0 \quad 8 \quad 4 \quad 2 \quad 1 \end{array}$$

$$\begin{array}{r} -k \\ k \\ 3 \end{array}$$

$$\begin{array}{r} 1. \quad 00010 \\ 2. \quad + \quad 1 \\ \hline 00011 \quad \checkmark \end{array}$$

# Zweierkomplement arithmetisch bilden



- In **beide** Richtungen anwendbar
  - Vorzeichenwechsel:  $k \rightarrow -k$
- **Algorithmus**
  1. Alle Bits invertieren ( $0 \rightarrow 1$ ,  $1 \rightarrow 0$ )
  2. Dann 1 addieren
- **Beispiel:** Vorzeichenwechsel von  $3_{10} = 00011_2$ 
  1.  **$11100_2$**
  2.  **$11101_2 = -3_{10}$**
- **Beispiel:** Vorzeichenwechsel von  $-3_{10} = 11101_2$ 
  1.  **$00010_2$**
  2.  **$00011_2 = 3_{10}$**

# Weitere Beispiele Zweierkomplement

- Bestimme Zweierkomplement von  $6_{10} = 0110_2$

$-6$  → 1010

$$\begin{array}{r} 1001 \\ + 1 \\ \hline 1010 \end{array}$$

- Was ist der Dezimalwert der Zweierkomplementzahl  $1001_2$ ?

$-7$  → 0111

$$\begin{array}{r} 10110 \\ + 1 \\ \hline 0111 \end{array}$$

# Weitere Beispiele Zweierkomplement

- Bestimme Zweierkomplement von  $6_{10} = 0110_2$

1.  $1001$   
2.  $+ 1$

---

$$1010_2 = -6_{10}$$

- Was ist der Dezimalwert der Zweierkomplementzahl  $1001_2$ ?

1.  $0110$   
2.  $+ 1$

---

$$0111_2 = 7_{10}, \text{ msb war vorher 1 also negativ: } 1001_2 = -7_{10}$$

# Addition im Zweierkomplement

- Addiere  $6 + (-6)$

$0 \Rightarrow \underline{0000}$

$$\begin{array}{r} 0111 \\ 0001 \\ \hline 1000 \end{array}$$

$$\begin{array}{r} 111 \\ 0110 \\ + 1010 \\ \hline 10000 \checkmark \end{array}$$

6  
-6

$$\begin{array}{r} 1 \\ 0101 \\ \hline 0110 \end{array} \checkmark$$

- / Addiere  $-2 + 3$

$$\begin{array}{r} 111 \\ 1110 \\ + 0011 \\ \hline 10001 \end{array}$$

# Addition im Zweierkomplement

- Addiere 6 + (-6)

$$\begin{array}{r}
 111 \\
 0110 \\
 + 1010 \\
 \hline
 10000
 \end{array}$$

4-bit

$$\begin{array}{r}
 1000 \\
 \hline
 -8 \quad 4 \quad 2 \quad 1
 \end{array}$$

$$\begin{array}{r}
 -7 \\
 -4 \\
 \hline
 \end{array}$$

Übertrag

Überlauf:

Ignorieren, wenn  
Positive und negative  
Zahlen gleicher  
Bitbreite addiert  
werden

- Addiere -2 + 3

$$\begin{array}{r}
 111 \\
 1110 \\
 + 0011 \\
 \hline
 10001
 \end{array}$$

$$\begin{array}{r}
 0 \dots 0 \quad 0 \quad 0 \quad 0 \quad 1 \\
 \hline
 31 \quad \quad \quad \quad \quad 4\text{-bit}
 \end{array}$$

# Erweitern von Zahlen auf höhere Bitbreite



- Verknüpfen von Zahlen unterschiedlicher Bitbreite?

- Verknüpfen von Zahlen **unterschiedlicher** Bitbreite?
- Anzahl Bits  $N$  der **schmaleren** Zahl erhöhen auf Breite  $M$  der anderen Zahl
- **Zwei Möglichkeiten**
  - Auffüllen mit führenden Nullen (*zero extension*)
  - Auffüllen mit dem bisherigen Vorzeichen (*sign extension*)

# Erweitern durch Auffüllen mit Vorzeichenbit

- Vorzeichenbit nach **links** kopieren bis gewünschte Breite erreicht
- Zahlenwert bleibt **unverändert**
  - Auch bei negativen Zahlen!

## ▪ **Beispiel 1:**

- 4-bit Darstellung von 3 =

- 8-bit aufgefüllt durch Vorzeichen: 00000011

0011



3

## ▪ **Beispiel 2:**

- 4-bit Darstellung von -5 =

- 8-bit aufgefüllt durch Vorzeichen: 11111011



1011

-5?

1. 00000100  
+ 00000101  
-----  
2. 00001001 ✓

# Erweitern durch Auffüllen mit Vorzeichenbit



- Vorzeichenbit nach **links** kopieren bis gewünschte Breite erreicht
- Zahlenwert bleibt **unverändert**
  - Auch bei negativen Zahlen!
- **Beispiel 1:**
  - 4-bit Darstellung von 3 = **0011**
  - 8-bit aufgefüllt durch Vorzeichen: **00000011**
- **Beispiel 2:**
  - 4-bit Darstellung von -5 = **1011**
  - 8-bit aufgefüllt durch Vorzeichen : **11111011**

# Erweitern durch Auffüllen mit Nullbits

- Nullen nach **links** anhängen bis gewünschte Breite erreicht
- **Zerstört** Wert von negativen Zahlen
  - Positive Zahlen bleiben **unverändert**
- **Beispiel 1:**
  - 4-bit Wert =  $0011 = 3_{10}$
  - 8-bit durch Auffüllen mit Nullbits:
- **Beispiel 2:**
  - 4-bit Wert =  $1011 = -5_{10}$
  - 8-bit durch Auffüllen mit Nullbits :

# Erweitern durch Auffüllen mit Nullbits

- Nullen nach **links** anhängen bis gewünschte Breite erreicht
- **Zerstört** Wert von negativen Zahlen
  - Positive Zahlen bleiben **unverändert**

## Beispiel 1:

- 4-bit Wert =  $0011 = 3_{10}$
- 8-bit durch Auffüllen mit Nullbits:  $00000011 = 3_{10}$

## Beispiel 2:

- 4-bit Wert =  $1011 = -5_{10}$
- 8-bit durch Auffüllen mit Nullbits :  $00001011 = 11_{10}$ , falsch!

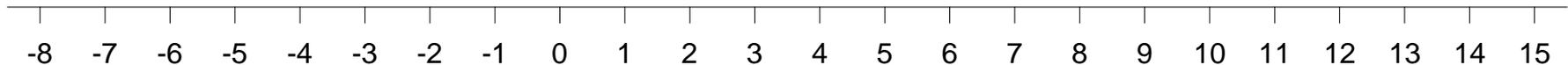
$8 + 2 + 1 = 11$

$\uparrow$

# Vergleich der Zahlensysteme

Zahlensystem	Wertebereich
Vorzeichenlos	$[0, 2^N-1]$
Vorzeichen/Betrag	$[-(2^{N-1}-1), 2^{N-1}-1]$
Zweierkomplement	$[-2^{N-1}, 2^{N-1}-1]$

Beispiel 4-bit breite Darstellung:



Vorzeichenlos

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

1000 1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110 0111

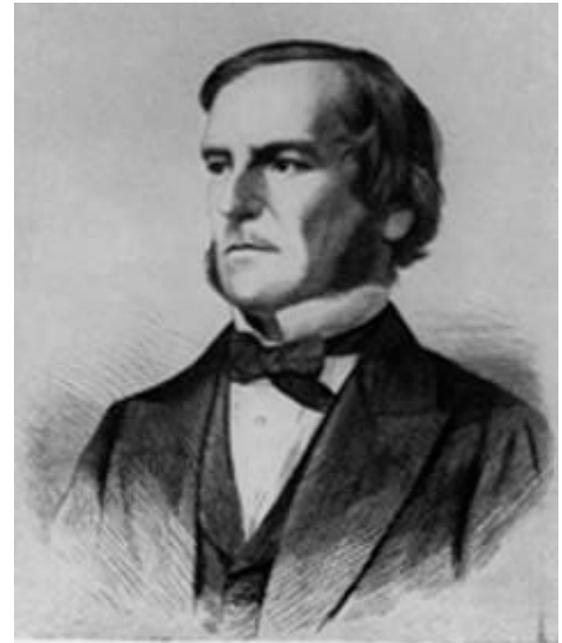
Zweierkomplement

1111 1110 1101 1100 1011 1010 1001 0000  
1000 0001 0010 0011 0100 0101 0110 0111

Vorzeichen/Betrag

# George Boole, 1815 - 1864

- In einfachen Verhältnissen geboren
- Brachte sich **selbst** Mathematik bei
- Später **Professor** am Queen's College in Irland
- Verfasste *An Investigation of the Laws of Thought*
  - 1854
- Einführung **binärer** Variablen

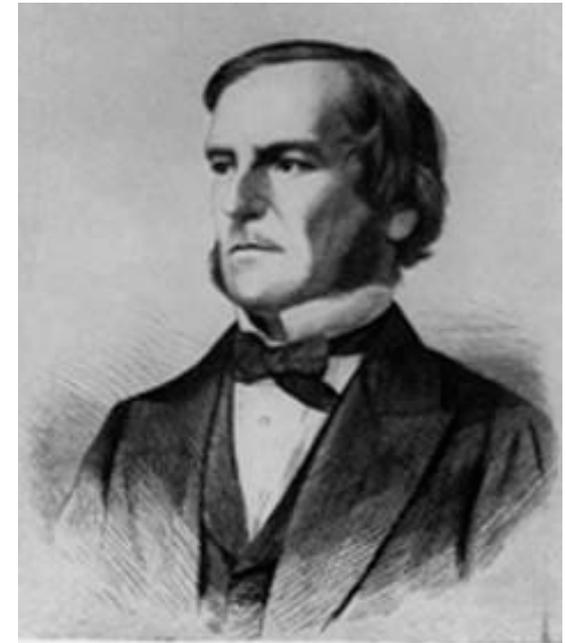


GEORGE BOOLE

Scanned at the American  
Institute of Physics

# George Boole, 1815 - 1864

- In einfachen Verhältnissen geboren
- Brachte sich **selbst** Mathematik bei
- Später **Professor** am Queen's College in Irland
- Verfasste *An Investigation of the Laws of Thought*
  - 1854
- Einführung **binärer** Variablen
- Einführung der drei grundlegenden **Logikoperationen**
  - UND (*AND*)
  - ODER (*OR*)
  - NICHT (*NOT*)
- Verknüpfen binäre Werte mit binärem Ergebnis



GEORGE BOOLE

Scanned at the American  
Institute of Physics

- Berechnen **logische** Funktionen:

- Inversion (NICHT), UND, ODER, ...
- NOT, AND, OR, NAND, NOR, ...

- **Ein Eingang:**

Nicht

- NOT Gatter, Puffer (*buffer*)

- **Zwei Eingänge:**

- AND, OR, XOR, NAND, NOR, XNOR

- **Viele Eingänge**

# Logikgatter mit einem Eingang

*Name*

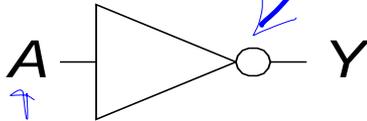
**NOT**

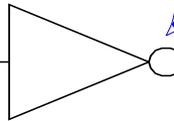
*Symbol*

*Formel*

*Wahrheitstabelle*

*Invertiert*

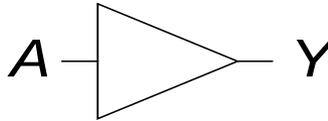


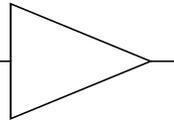
$A$  —  —  $Y$

$Y = \overline{A}$

$A$	$Y$
0	1
1	0

**BUF**



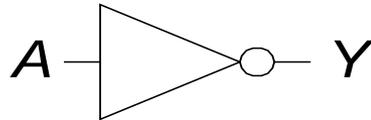
$A$  —  —  $Y$

$Y = A$

$A$	$Y$
0	0
1	1

# Logikgatter mit einem Eingang

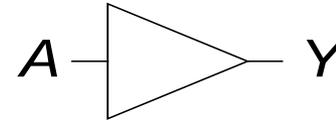
## NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

## BUF



$$Y = A$$

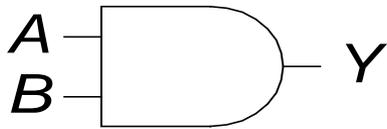
A	Y
0	0
1	1

Alternative Schreibweisen

$Y = !A$ ,  $Y = \sim A$ ,  $Y = \neg A$ ,  $Y = A'$

# Logikgatter mit zwei Eingängen

## AND

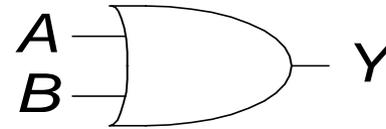


$$Y = AB$$

*Handwritten:  $Y = A \cdot B$*

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

## OR

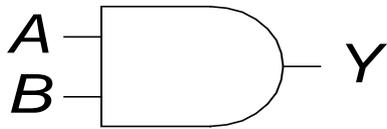


$$Y = A + B$$

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

# Logikgatter mit zwei Eingängen

## AND



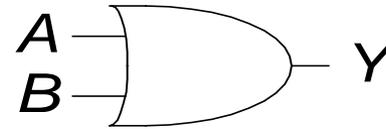
$$Y = AB$$

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

### Alternative Schreibweisen

$$Y = A \& B, Y = A * B, Y = A \cap B$$

## OR



$$Y = A + B$$

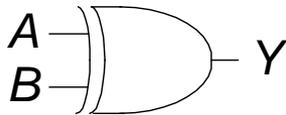
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

### Alternative Schreibweisen

$$Y = A | B, Y = A \cup B$$

# Weitere Logikgatter mit zwei Eingängen

**XOR**

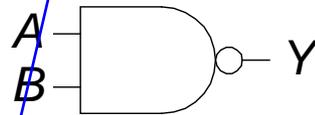


$$Y = A \oplus B$$

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

*Nicht-AND*

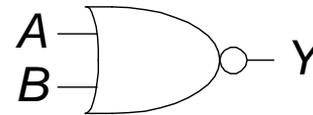
**NAND**



$$Y = \overline{AB}$$

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

**NOR**

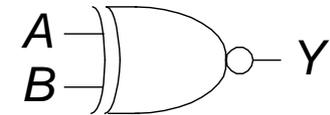


$$Y = \overline{A + B}$$

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

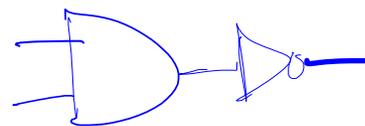
*NXOR*

**XNOR**

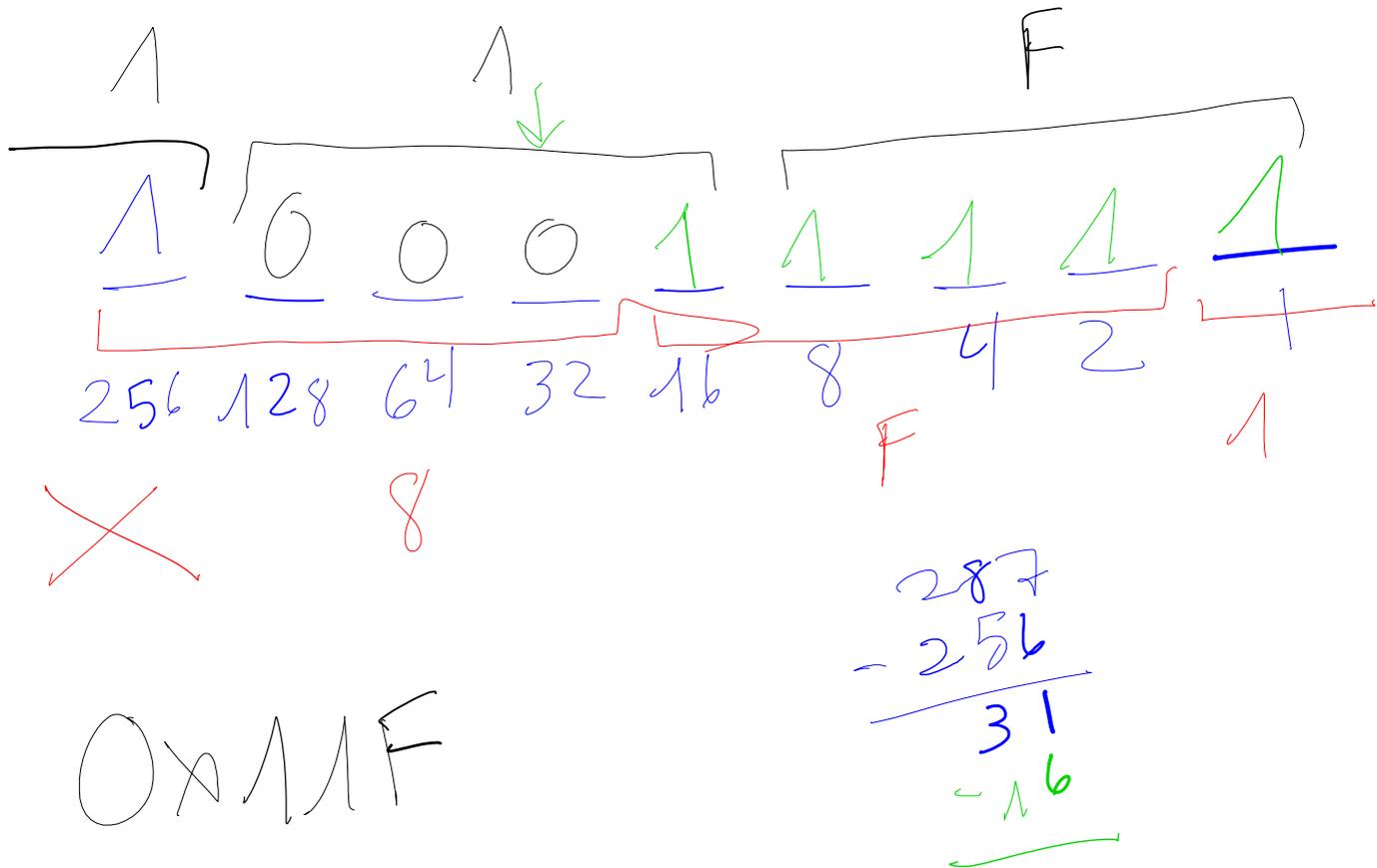


$$Y = \overline{A \oplus B}$$

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



Was ist die binrische Darstellung von der Dezimalzahl 287?



**Quiz 2:** Was ist das lsb von der Binrdarstellung der Dezimalzahl 42?

**Antwort:** 0 (ist eine gerade Zahl, also muss doch keine 1 in der eins Stelle (b.z.w. lsb Stelle) sein.)



1 0 0 0 0

-16



1. 1 1 1 1  
0 1 1 1 1

← einer Komplement

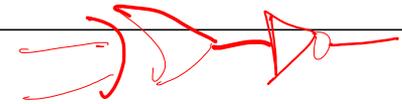
2. +  
-----  
1 0 0 0 0

0 0 , 0 0 1 1

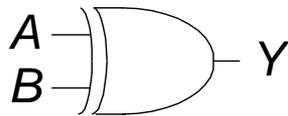
~~0 0 0 0~~ 1 1 1 1 - 1  
4-bit

1 1 1 1 1 1 1 1 - 1  
8-bit

# Weitere Logikgatter mit zwei Eingängen



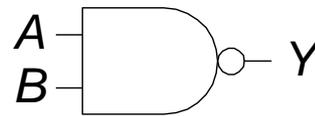
## XOR



$$Y = A \oplus B$$

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

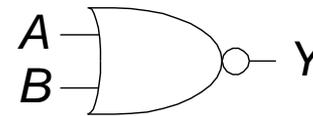
## NAND



$$Y = \overline{AB}$$

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

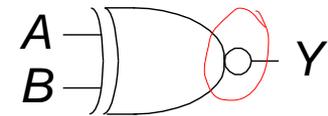
## NOR



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

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

## XNOR



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

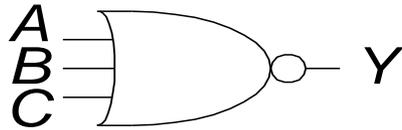
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Alternative  
Schreibweise

$$Y = A \wedge B$$

# Logikgatter mit mehr als zwei Eingängen

**NOR3**



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

A	B	C	Y
0	0	0	1
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	0



A+B+C

0

1

⋮

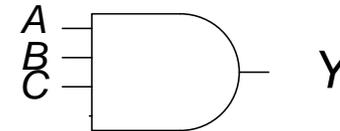
⋮

⋮

⋮

1

**AND3**

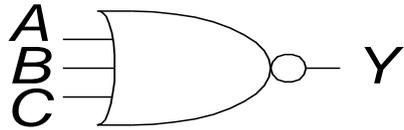


$$Y = ABC$$

A	B	C	Y
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

# Logikgatter mit mehr als zwei Eingängen

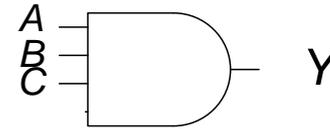
## NOR3



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

A	B	C	Y
0	0	0	1
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	0

## AND3

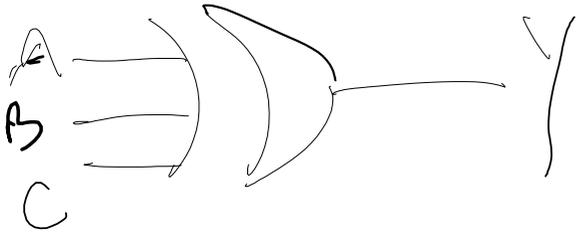


$$Y = ABC$$

A	B	C	Y
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

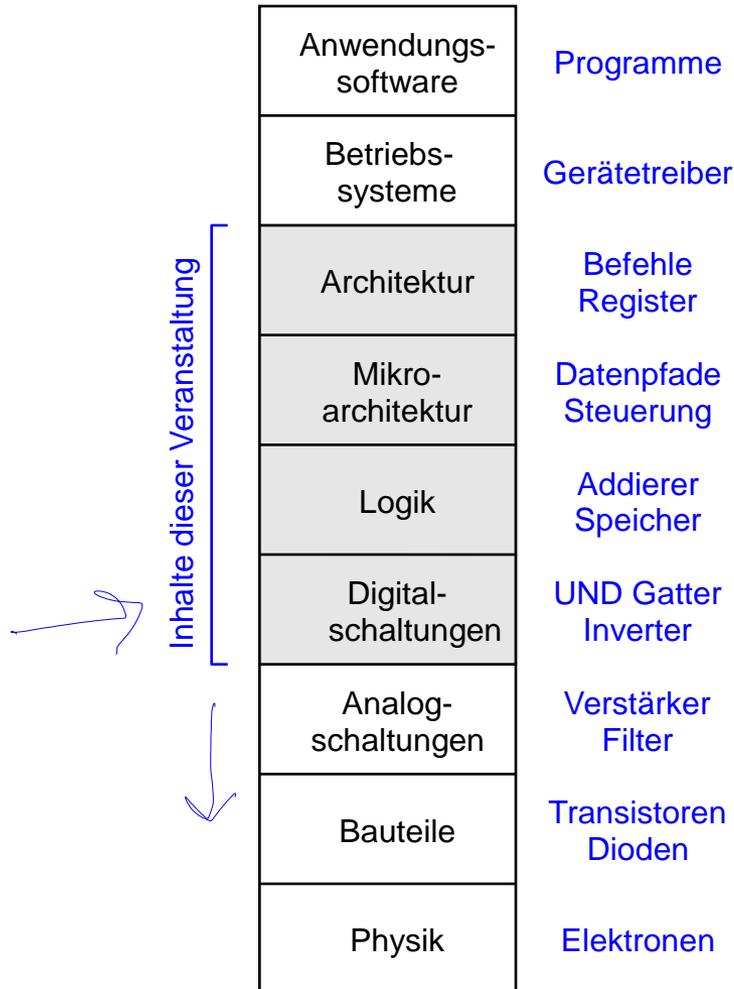
# XOR mit mehreren Eingängen

- Paritätsfunktion
  - Erkennt **gerade** oder **ungerade** Anzahl von Eingängen mit Wert 1
- XOR
  - **Ungerade** Paritätsfunktion
  - Liefert 1 am Ausgang, wenn **ungerade** Anzahl von Eingängen den Wert 1 haben



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

# Abstraktion



# Darstellung von Binärwerten durch Spannungen



- Definiere **Spannungspegel** für die Werte 0 und 1
  - Logikpegel (*logic levels*)



# Darstellung von Binärwerten durch Spannungen



- Definiere **Spannungspegel** für die Werte 0 und 1
  - Logikpegel (*logic levels*)
- **Beispiel:**
  - 0 Volt (Erde, **ground**) entspricht Binärwert 0
  - 5 Volt (Versorgungsspannung,  $V_{DD}$ ) entspricht Binärwert 1

4,99 V  $\rightarrow$  1  
3,2 V  $\rightarrow$  0? 1?



- Definiere **Spannungspegel** für die Werte 0 und 1
  - Logikpegel (*logic levels*)
- **Beispiel:**
  - 0 Volt (Erde, **ground**) entspricht Binärwert 0
  - 5 Volt (Versorgungsspannung,  $V_{DD}$ ) entspricht Binärwert 1
- **Probleme**
  - Wofür steht **4,99 V**? Den Wert 0 oder 1?
  - Wofür steht **3,2V**?
- Reale Schaltungen haben **keine** ganz exakten Spannungspegel
  - Teils sogar Umgebungsabhängig (Temperatur, Einstreuen, ...)
  - Solche Spannungsschwankungen werden **Rauschen** genannt

# Was ist Rauschen?

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

# Was ist Rauschen?

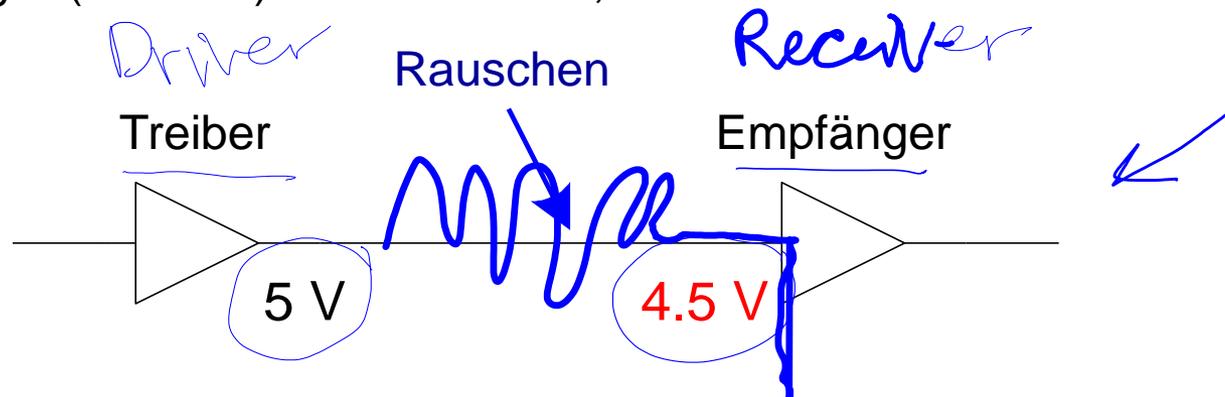
## ▪ Jede Störung der Nutzsignale

- Unerwünschte Widerstände, Kapazitäten und Induktivitäten
- Instabile Betriebsspannung
- Übersprechen von benachbarten Leitungen

...

## ▪ Beispiel

- Gatter gibt 5V aus (Treiber, *driver*)
- Lange Leitung hat hohen Widerstand (Spannungsabfall 0,5V)
- Am Empfänger (*receiver*) kommen nur 4,5V an



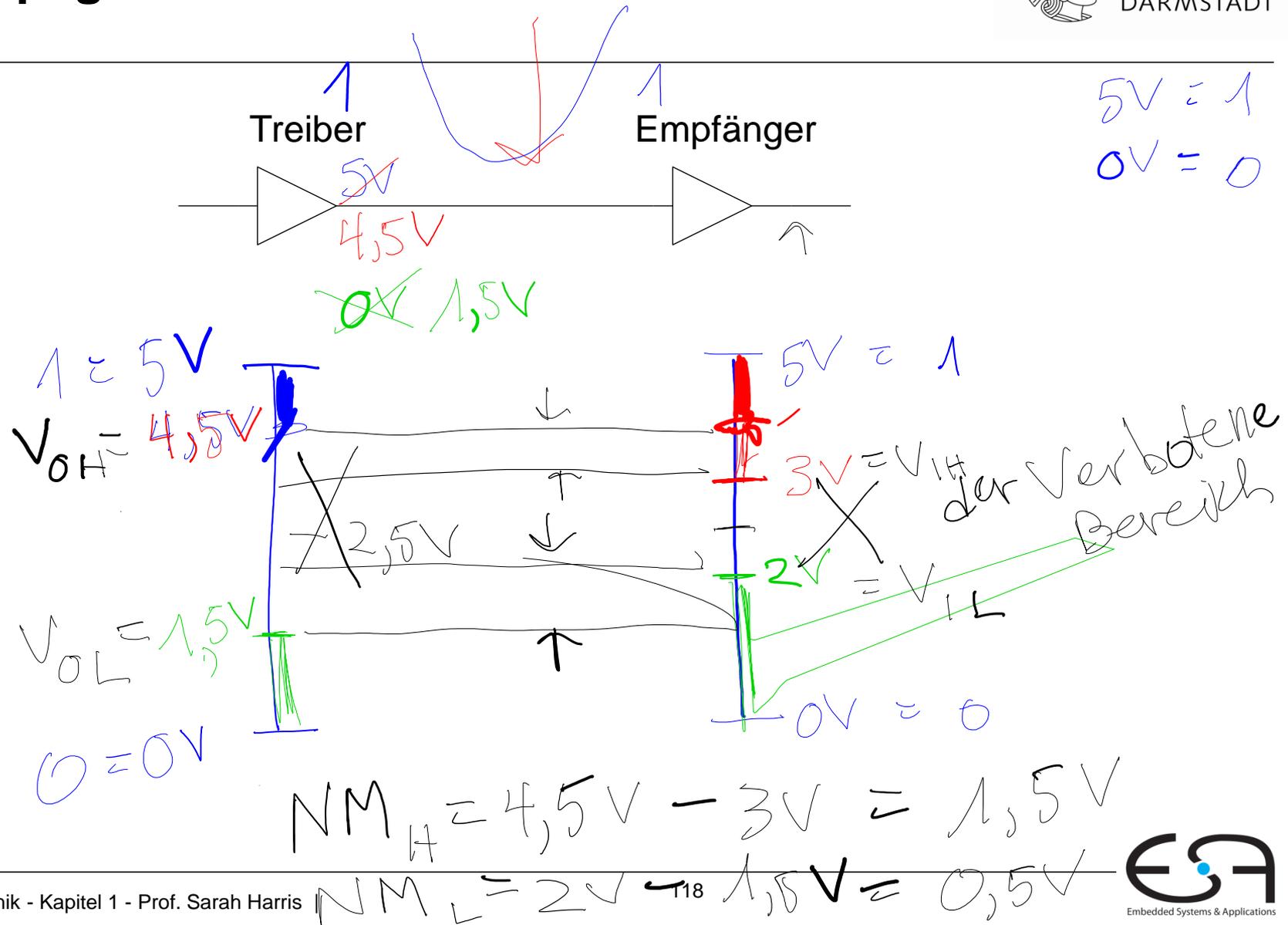


## ▪ Lösung

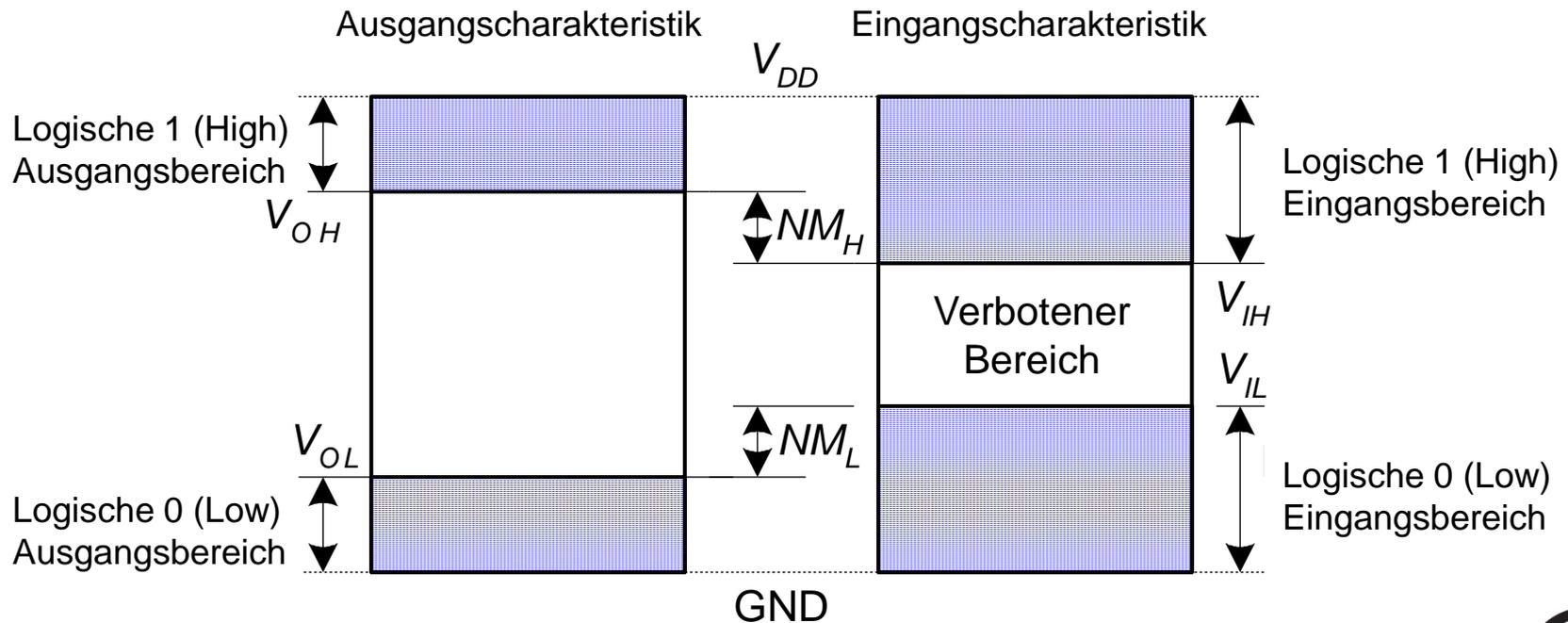
- Statt **einzelner Spannungspegel** für 0 und 1 ...
- ... verwende **Bereiche** von Spannungspegeln für 0 und 1
  
- Steigere Robustheit durch **unterschiedliche** Bereiche für
  - Eingänge
  - Ausgänge

- Jedes Schaltungselement muss bei **Eingabe** gültiger Logikpegel auch am **Ausgang** einen gültigen Logikpegel liefern
- Verwende nur **einen Satz** Spannungsbereiche für Logikpegel in gesamter Schaltung
  - Wird manchmal bewusst missachtet
    - Optimierung von Platz, Geschwindigkeit, Energiebedarf, Kosten, ...
    - ... bedarf aber großer **Vorsicht**

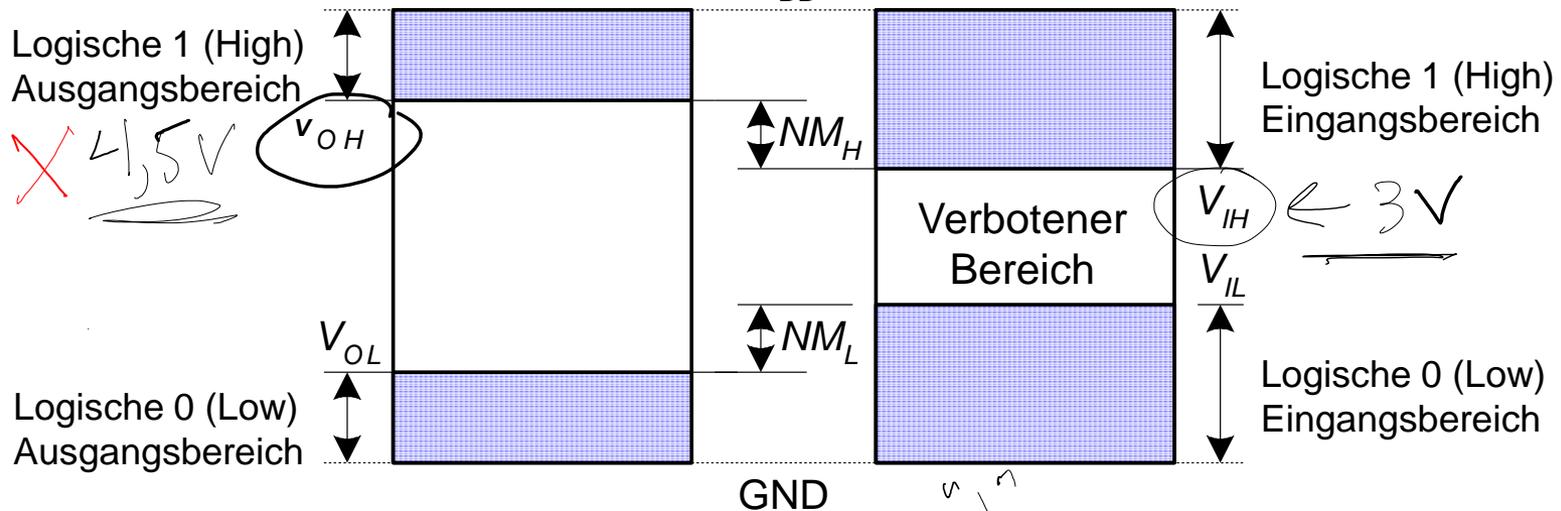
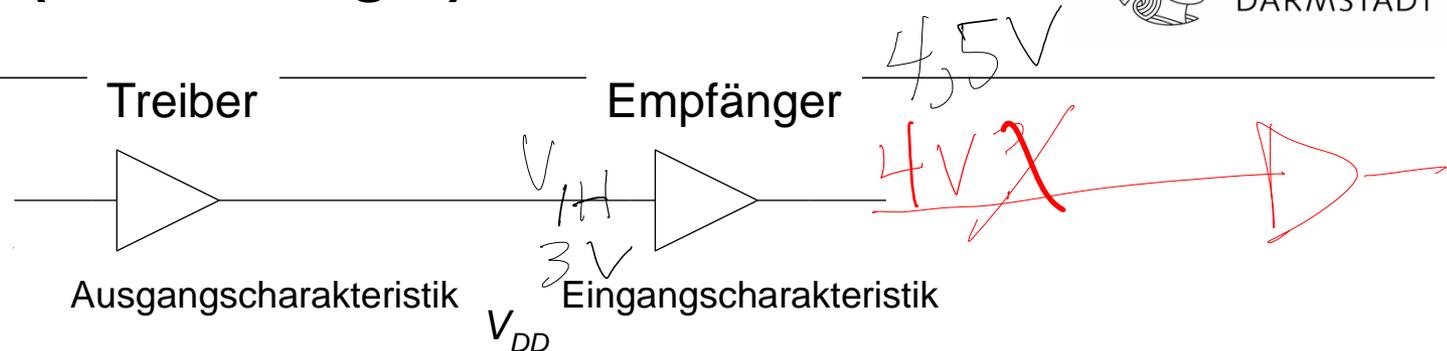
# Logikpegel



# Logikpegel



# Störabstand (*noise margin*)



**Hoch Störabstand:**  $NM_H = V_{OH} - V_{IH}$

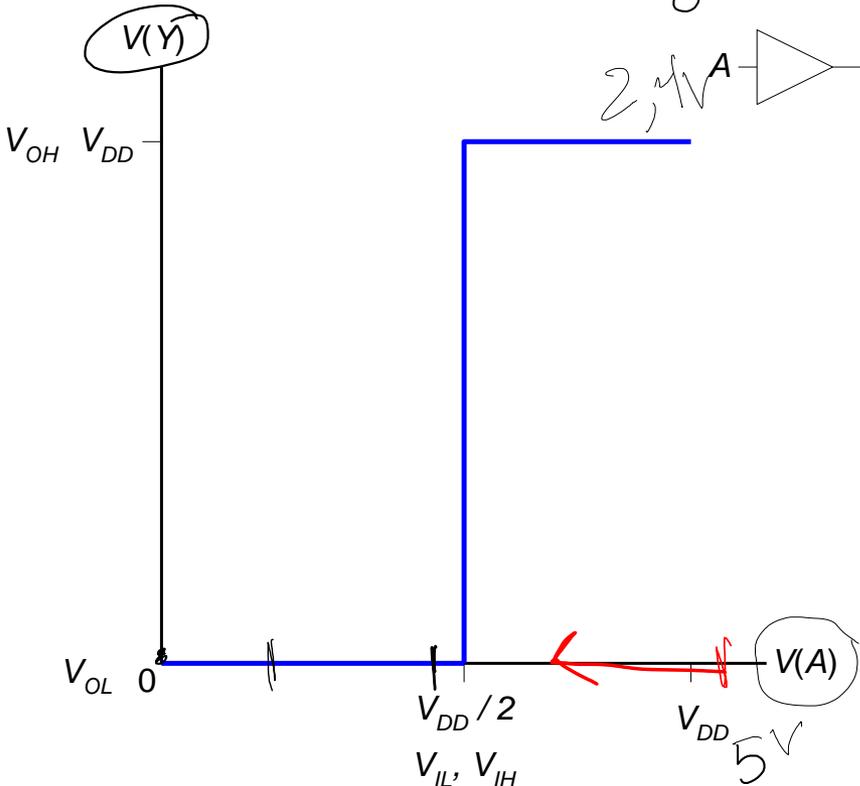
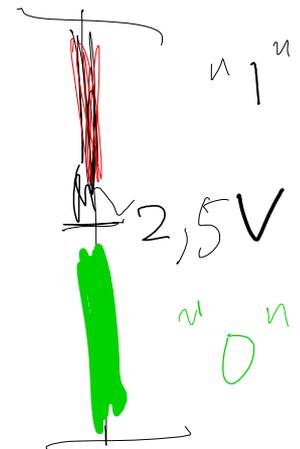
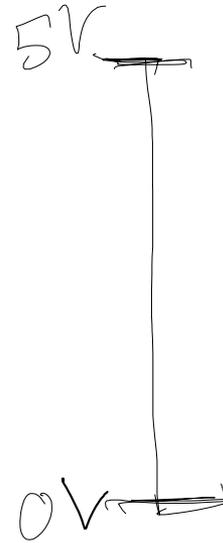
**Niedriger Störabstand:**  $NM_L = V_{IL} - V_{OL}$

# Gleichstrom-Transferkurve (DC transfer characteristics)

Idealer Buffer:

1V  
0V  
0V  
Ausgang

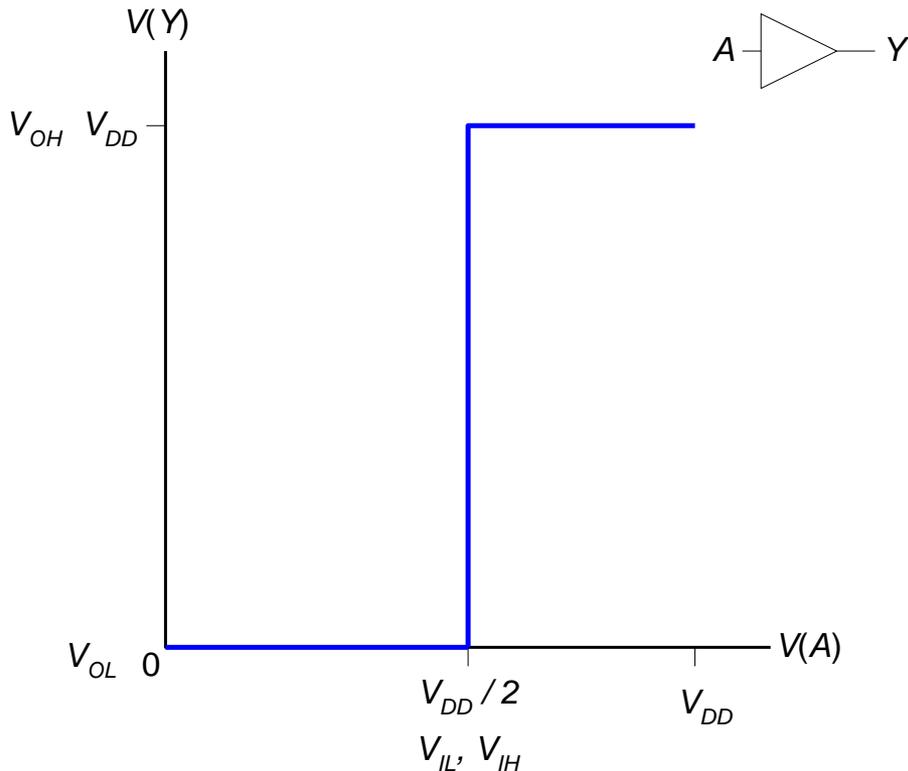
Eingang



$$NM_H = NM_L = V_{DD}/2$$

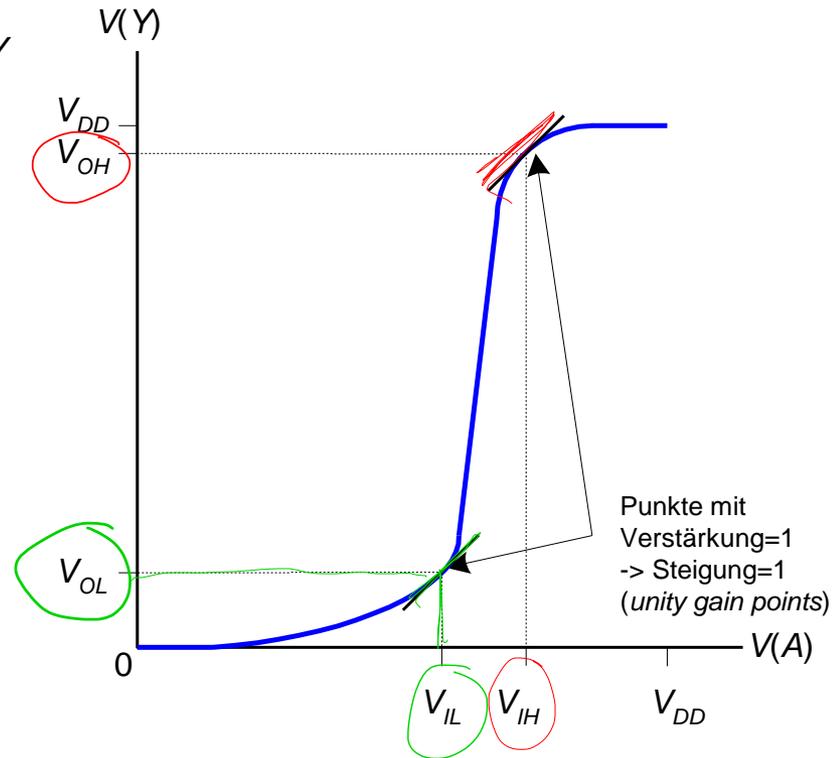
# Gleichstrom-Transferkurve (DC transfer characteristics)

## Idealer Buffer:



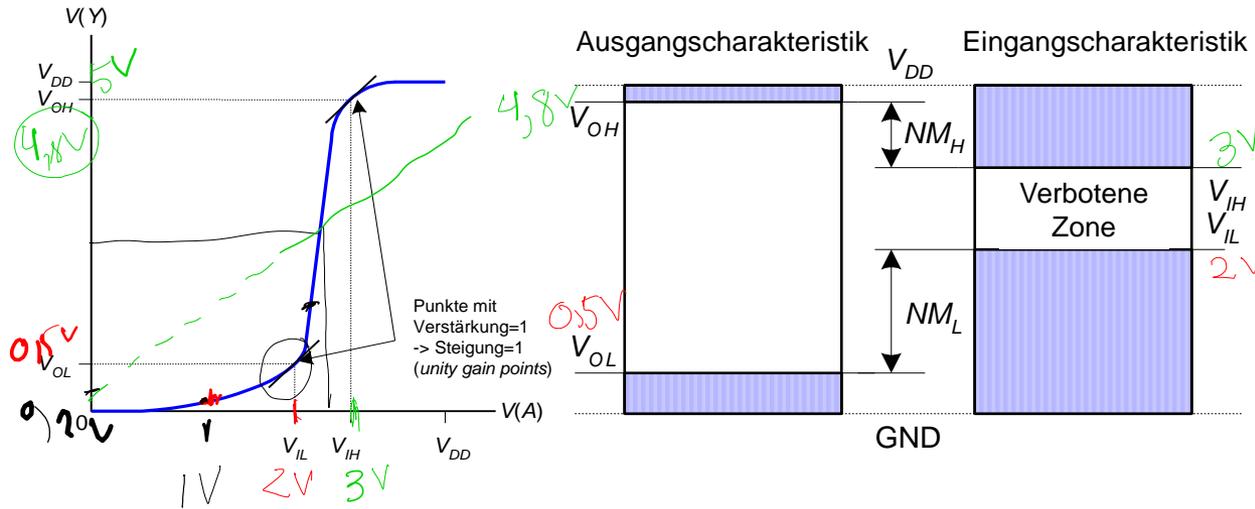
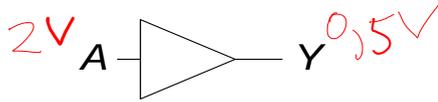
$$NM_H = NM_L = V_{DD}/2$$

## Realer Buffer:



$$NM_H, NM_L < V_{DD}/2$$

# Gleichstrom-Transferkurve



Handwritten notes:

$$V_{IL} - V_{OL} = 2V - 0,5V = 1,5V$$

Wenn wir einen anderen Punkt ausgewählt hätten  
z.B.  $V_{IL} = 1V$   
 $V_{OL} = 0,2V$

$NM_L = 1V - 0,2V = 0,8V$   
Störabstand wird kleiner

# Absenken der Versorgungsspannung $V_{DD}$

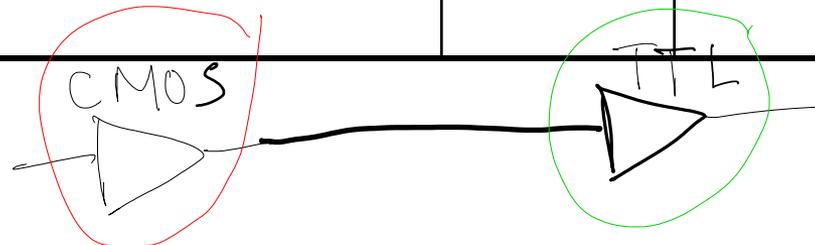
- **Versorgungsspannung** in den 70er-80er Jahren:  $V_{DD} = 5\text{ V}$
- Verbesserte Chip-Fertigungstechnologie erforderten **Absenkung** von  $V_{DD}$ 
  - Hohe Spannungen würden nun sehr kleine Transistoren **beschädigen**
  - **Energiebedarf** reduzieren
- 3.3 V, 2.5 V, 1.8 V, 1.5 V, 1.2 V, 1.0 V, ...
- Vorsicht beim Verbinden von Chips mit **unterschiedlichen** Versorgungsspannungen!



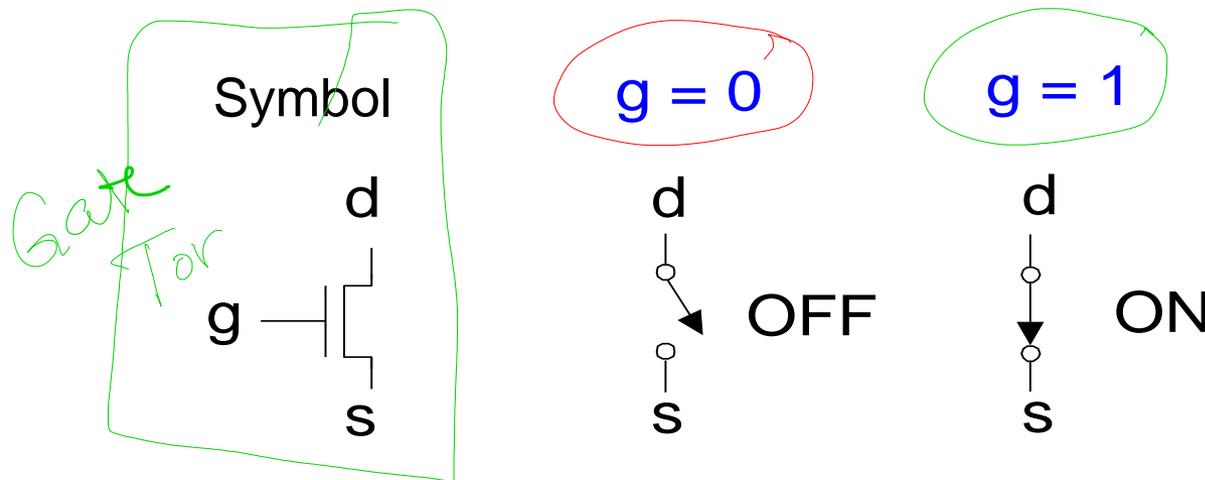
# Beispiele für Logikfamilien

- Bausteine mit **kompatiblen** Spannungspegeln

Logikfamilie	$V_{DD}$	$V_{IL}$	$V_{IH}$	$V_{OL}$	$V_{OH}$
TTL	5 (4.75 - 5.25)	0.8	2.0	0.4	2.4
CMOS	5 (4.5 - 6)	1.35	3.15	0.33	<del>3.84</del>
LVTTTL	3.3 (3 - 3.6)	0.8	2.0	0.4	2.4
LVC MOS	3.3 (3 - 3.6)	0.9	1.8	0.36	2.7

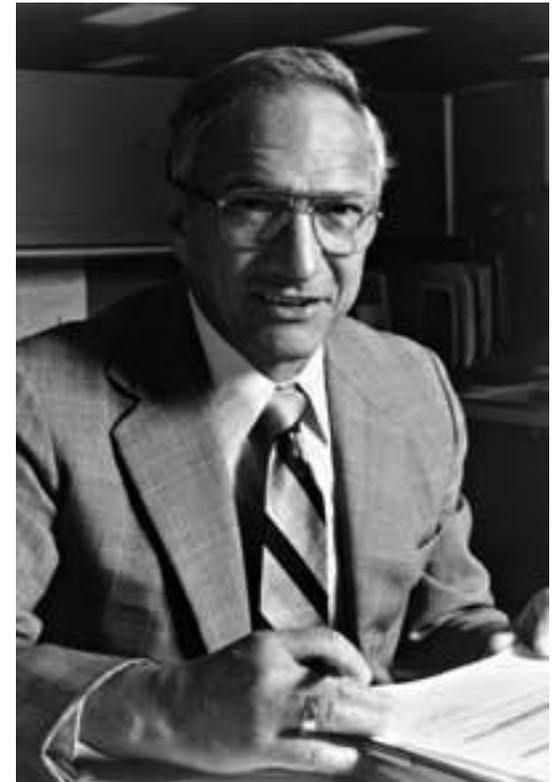


- Logikgatter werden üblicherweise aus **Transistoren** aufgebaut
  - Heute überwiegend **Feldeffekttransistoren** (FET)
  - Weiteres bezieht sich implizit auf **FETs**, nicht Bipolartransistoren
- Transistoren sind spannungsgesteuerte **Schalter**
  - Zwei Anschlüsse werden abhängig von Spannung an einem dritten **geschaltet**
    - Verbunden oder getrennt
  - **Beispiel:** Verbindung zwischen d,s verbunden wenn  $g=1$ , getrennt wenn  $g=0$

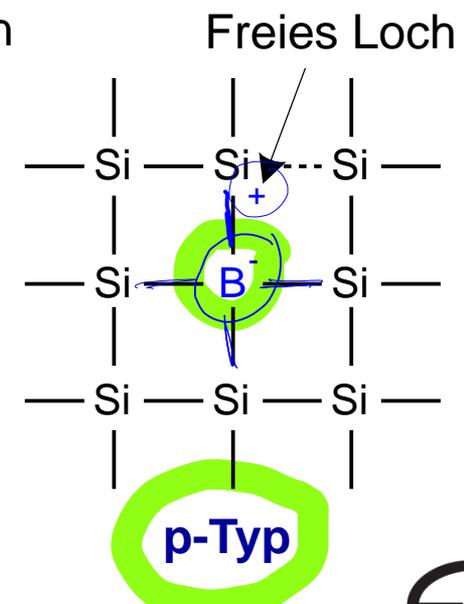
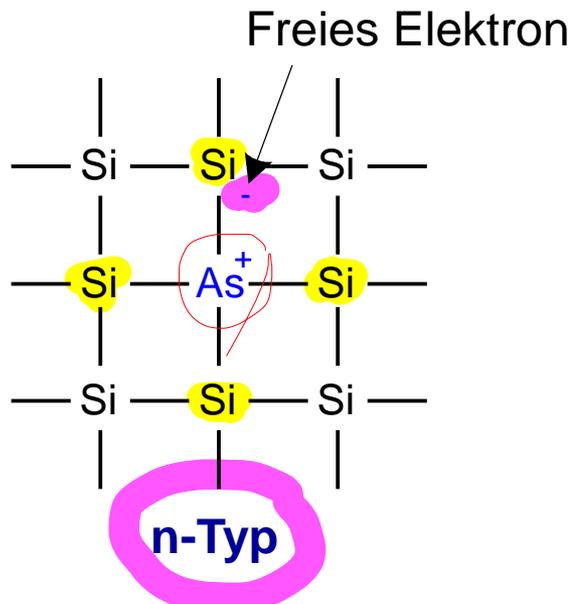
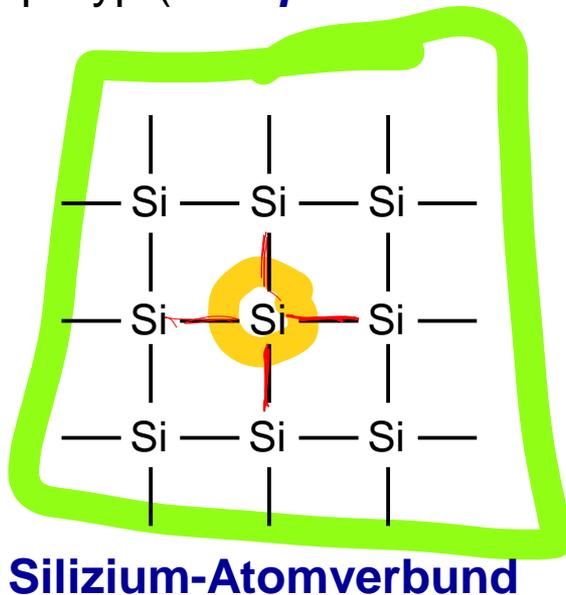


# Robert Noyce, 1927 - 1990

- Spitzname “Bürgermeister von Silicon Valley”
- Mitgründer von Fairchild Semiconductor in 1957
- Mitgründer von Intel in 1968
- Miterfinder der integrierten Schaltung

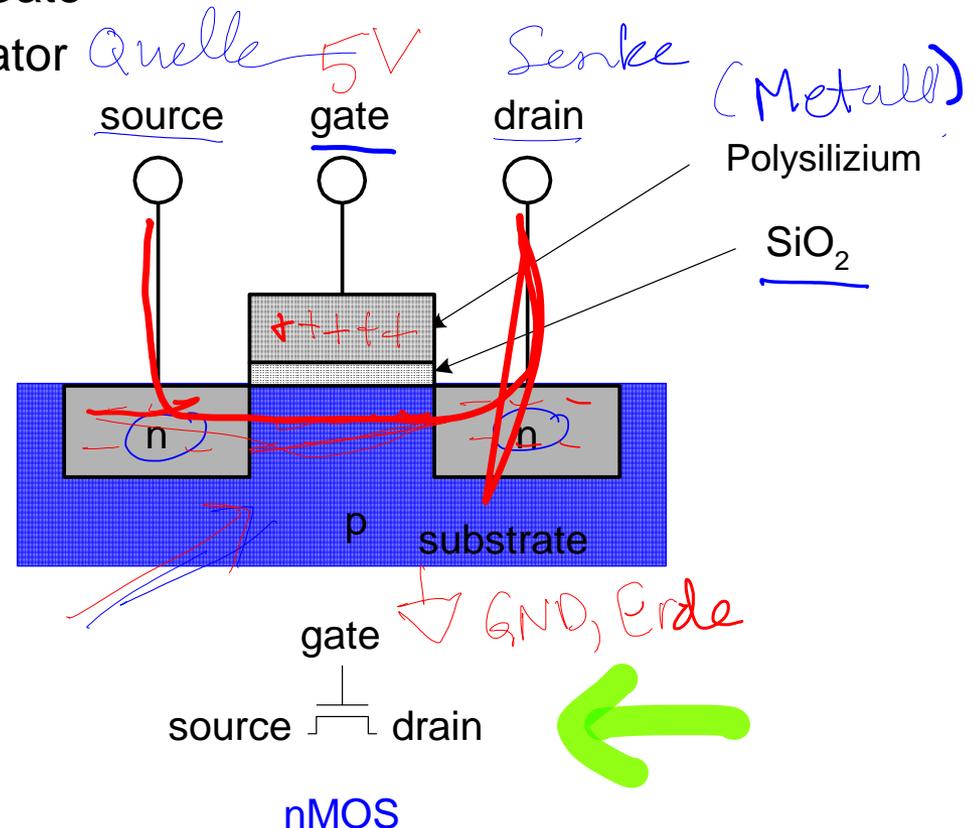
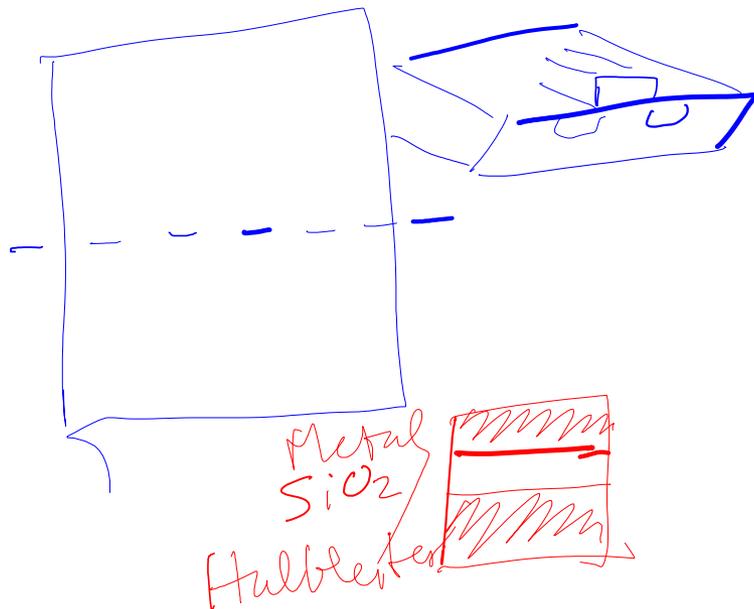


- Transistoren werden üblicherweise aus **Silizium** (Si, **Gruppe IV**) gefertigt
- Reines Silizium ist ein **schlechter** Leiter (keine freien Ladungsträger)
- Dotiertes Silizium ist ein **guter** Leiter (freie Ladungsträger)
  - n-Typ (freie **negative** Ladungsträger, Elektronen, dotiert mit **Arsen, Gruppe V**)
  - p-Typ (freie **positive** Ladungsträger, Löcher, dotiert mit **Bor, Gruppe III**)



# MOS Feldeffekttransistoren (MOSFETs)

- Metalloxid-Silizium (MOS) Transistoren
  - Polysilizium (früher **Metallschicht**) Gate
  - **Oxid** (Siliziumdioxid = Glas) als Isolator
  - Dotiertes Silizium

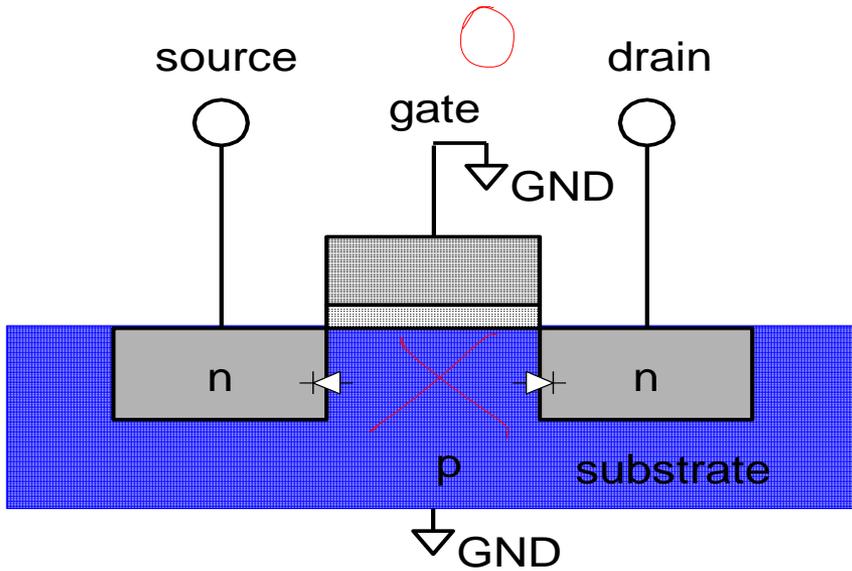


# Transistor: nMOS

**Gate = 0**, ausgeschaltet

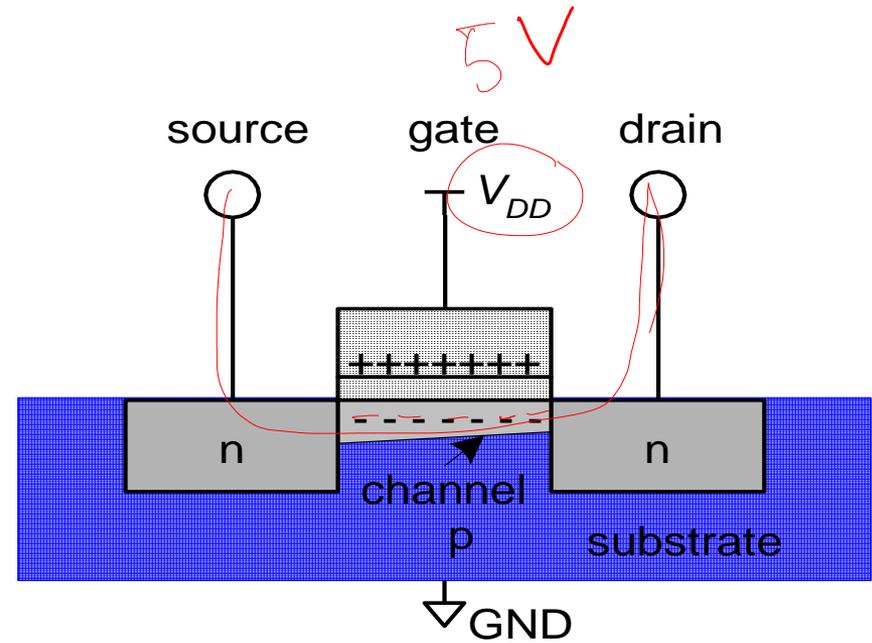
- keine Verbindung zwischen Source und Drain

*getrennt*



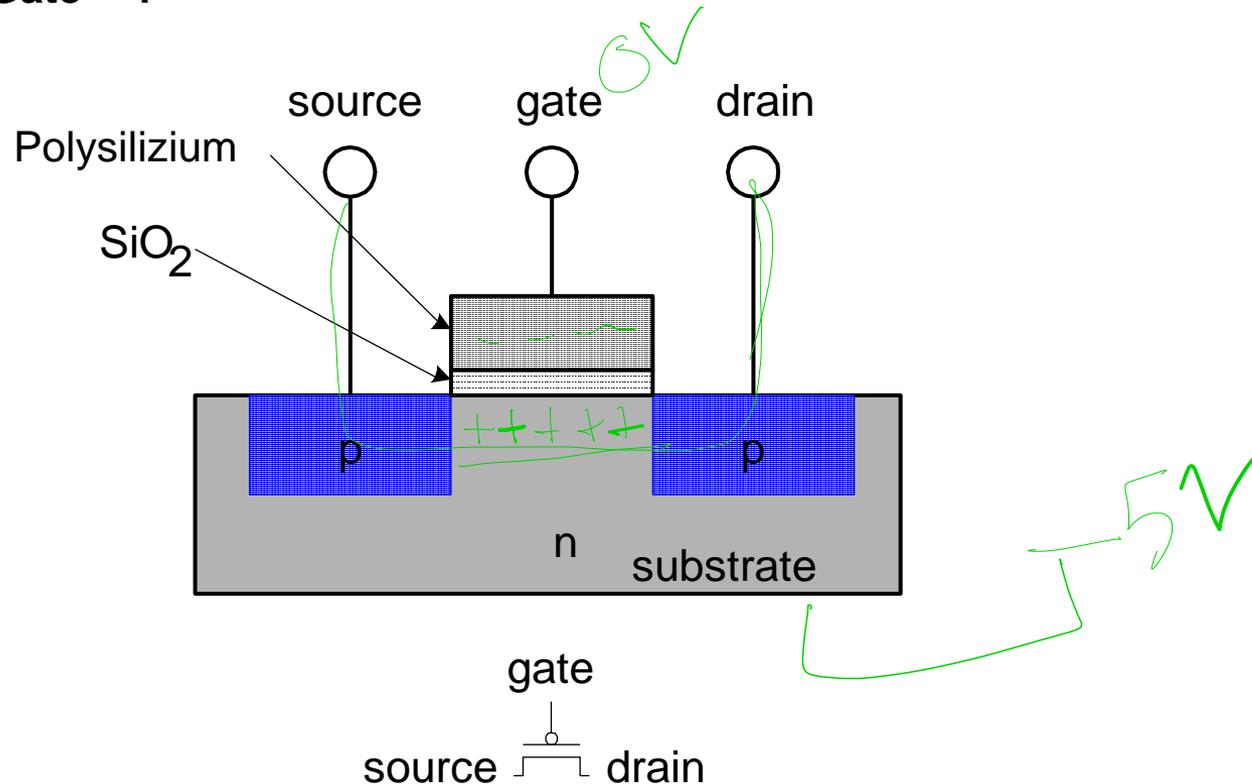
**Gate = 1**, eingeschaltet

- leitfähiger Kanal zwischen Source und Drain)



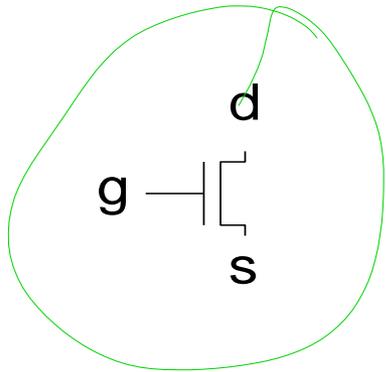
# Transistor: pMOS

- Verhalten von pMOS Transistor ist genau **umgekehrt**
  - **EIN** wenn **Gate = 0**
  - **AUS** wenn **Gate = 1**

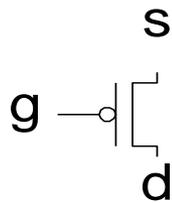


# Übersicht über Funktion von Transistoren

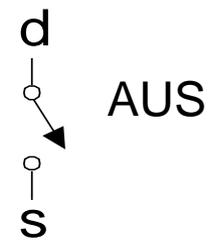
nMOS



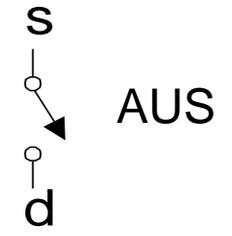
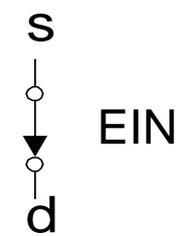
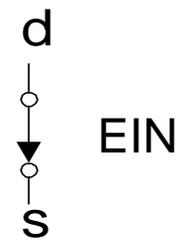
pMOS



$g = 0$



$g = 1$

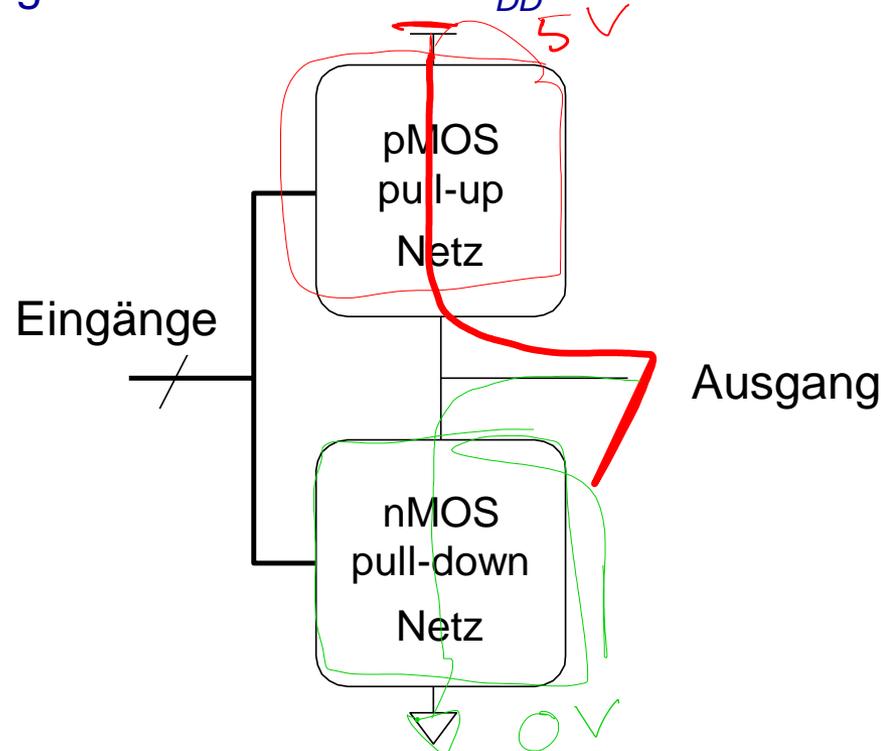


$g = 0$

$g = 1$

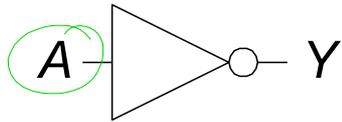
# Kombinieren von komplementären Transistoren (CMOS)

- nMOS Transistoren leiten 0'en gut zwischen S und D weiter
  - 1'en werden abgeschwächt → S an GND anschließen
- pMOS Transistoren leiten 1'en gut zwischen S und D weiter
  - 0'en werden abgeschwächt → S an  $V_{DD}$  anschließen



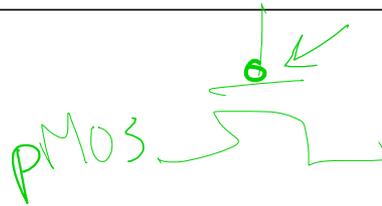
# CMOS Gatter: NOT

**NOT**

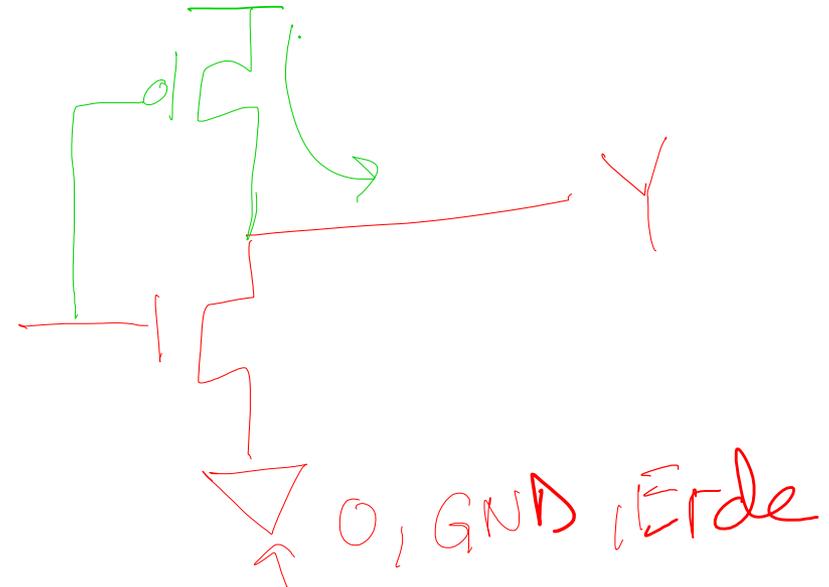


$$Y = \bar{A}$$

A	Y
0	1
1	0

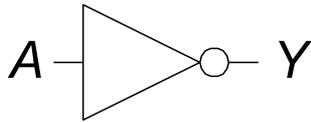


✓  $1 = A$   
 $0 = \bar{A}$



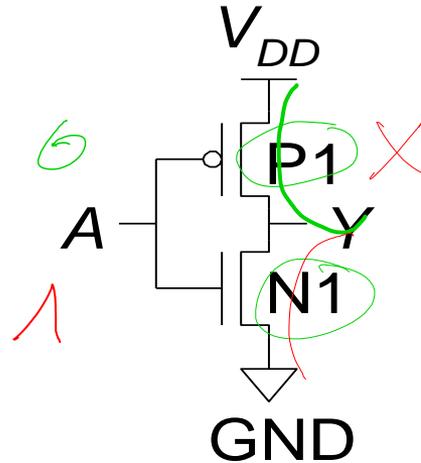
# CMOS Gatter: NOT

## NOT



$$Y = \bar{A}$$

A	Y
0	1
1	0



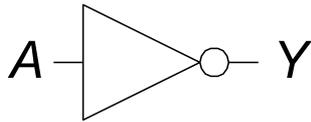
A	P1	N1	Y
0	EIN	AUS	1
1	AUS	EIN	0

→

✓

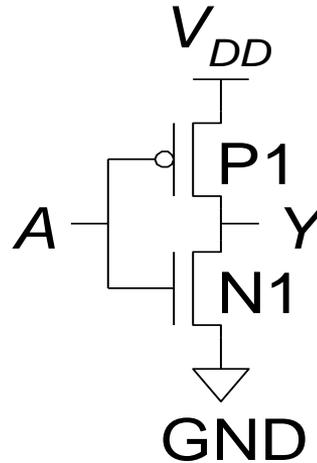
# CMOS Gatter: NOT

**NOT**



$$Y = \overline{A}$$

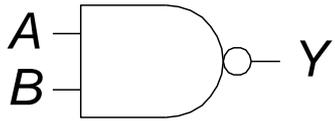
A	Y
0	1
1	0



A	P1	N1	Y
0	EIN	AUS	1
1	AUS	EIN	0

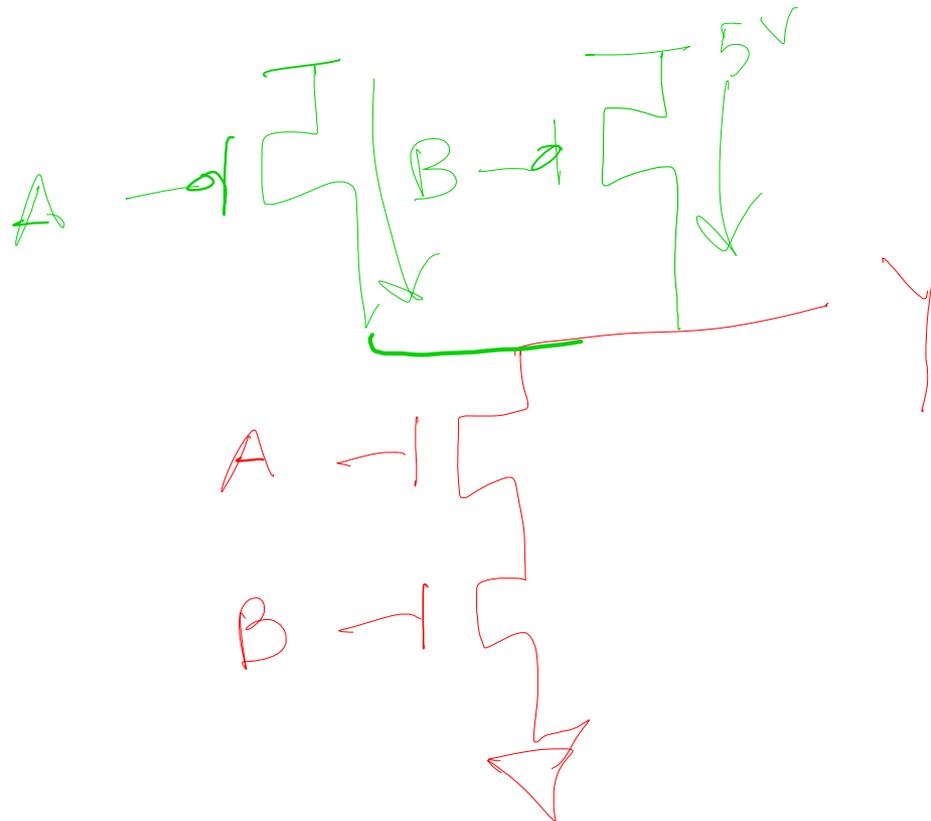
# CMOS Gates: NAND Gate

## NAND



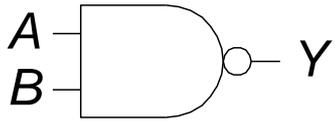
$$Y = \overline{AB}$$

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



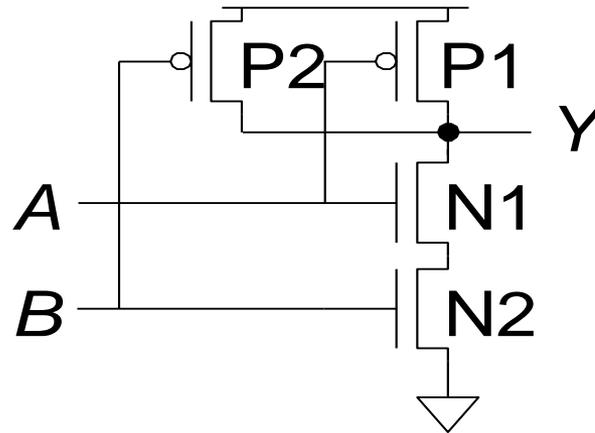
# CMOS Gates: NAND Gate

## NAND



$$Y = \overline{AB}$$

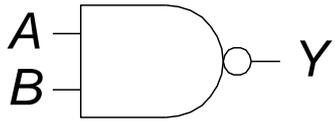
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A	B	P1	P2	N1	N2	Y
0	0					
0	1					
1	0					
1	1					

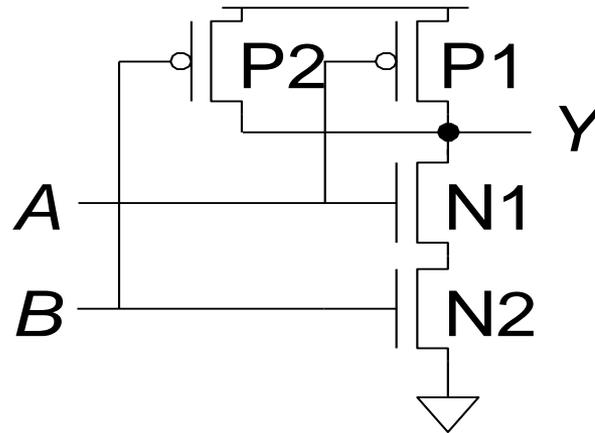
# CMOS Gates: NAND Gate

## NAND



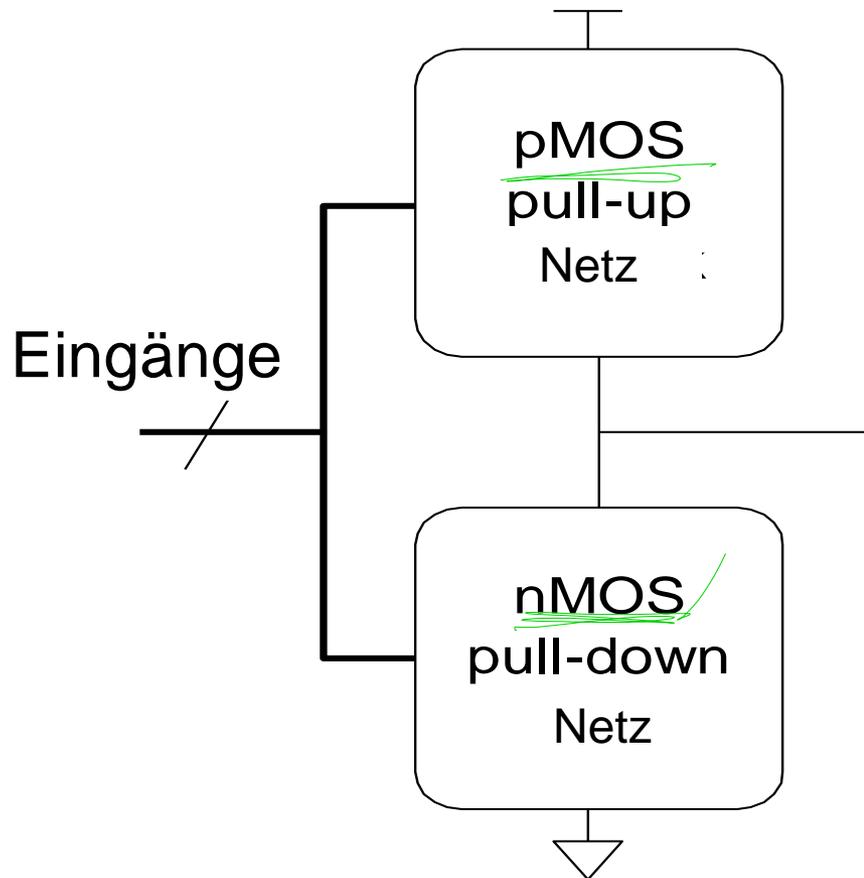
$$Y = \overline{AB}$$

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



A	B	P1	P2	N1	N2	Y
0	0	EIN	EIN	AUS	AUS	1
0	1	EIN	AUS	AUS	EIN	1
1	0	AUS	EIN	EIN	AUS	1
1	1	AUS	AUS	EIN	EIN	0

# Struktur eines CMOS Gatters

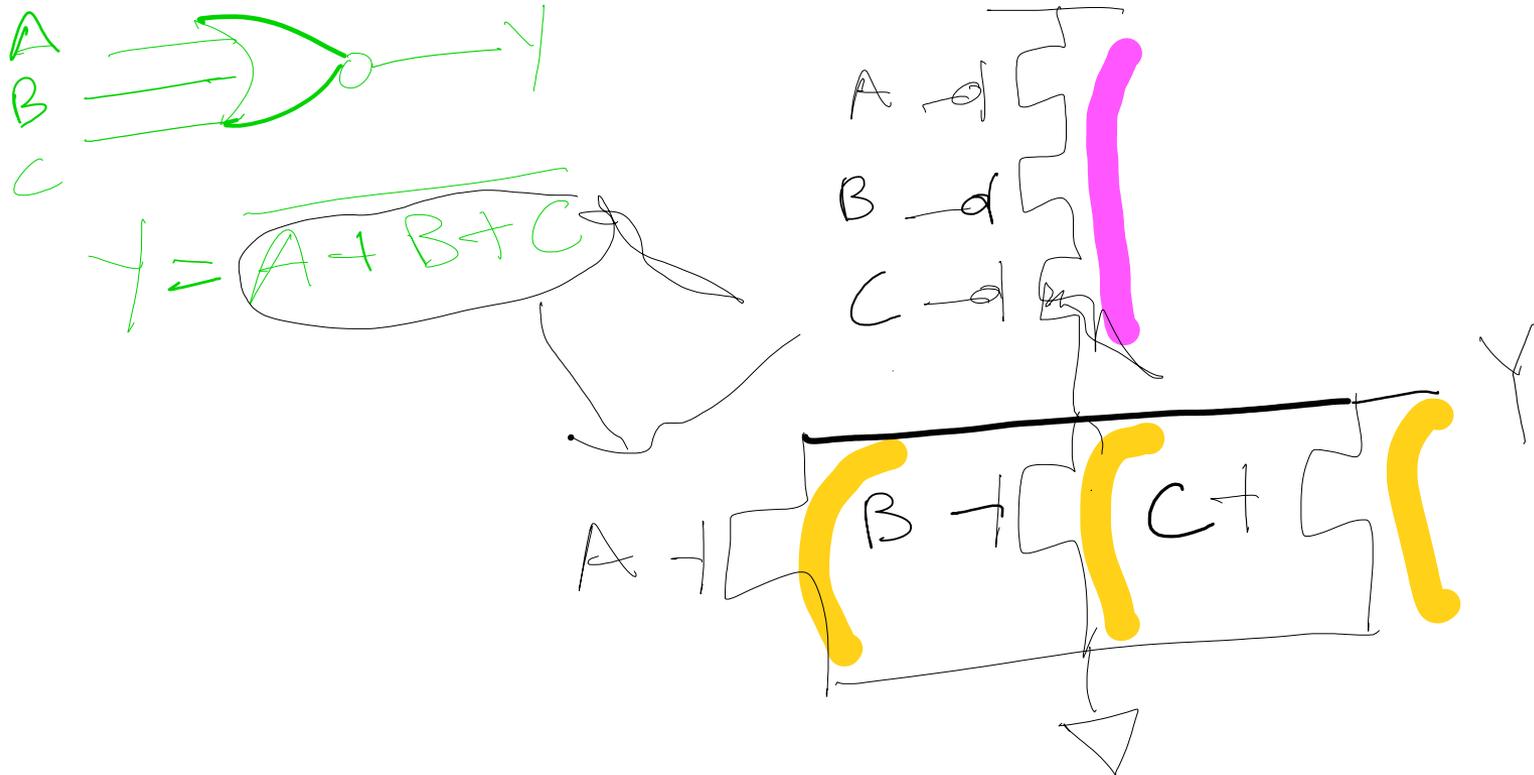


Wenn p-Transistoren in **Reihenschaltung**  
dann n-Transistoren in **Parallelschaltung**.

Ausgang

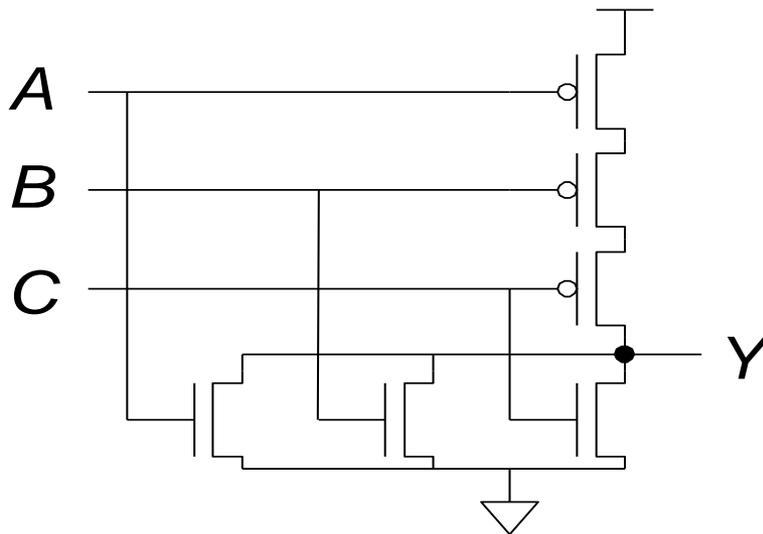
Wenn n-Transistoren in **Reihenschaltung**  
dann p-Transistoren in **Parallelschaltung**.

# Aufbau eines NOR-Gatters mit drei Eingängen



# Aufbau eines NOR-Gatters mit drei Eingängen

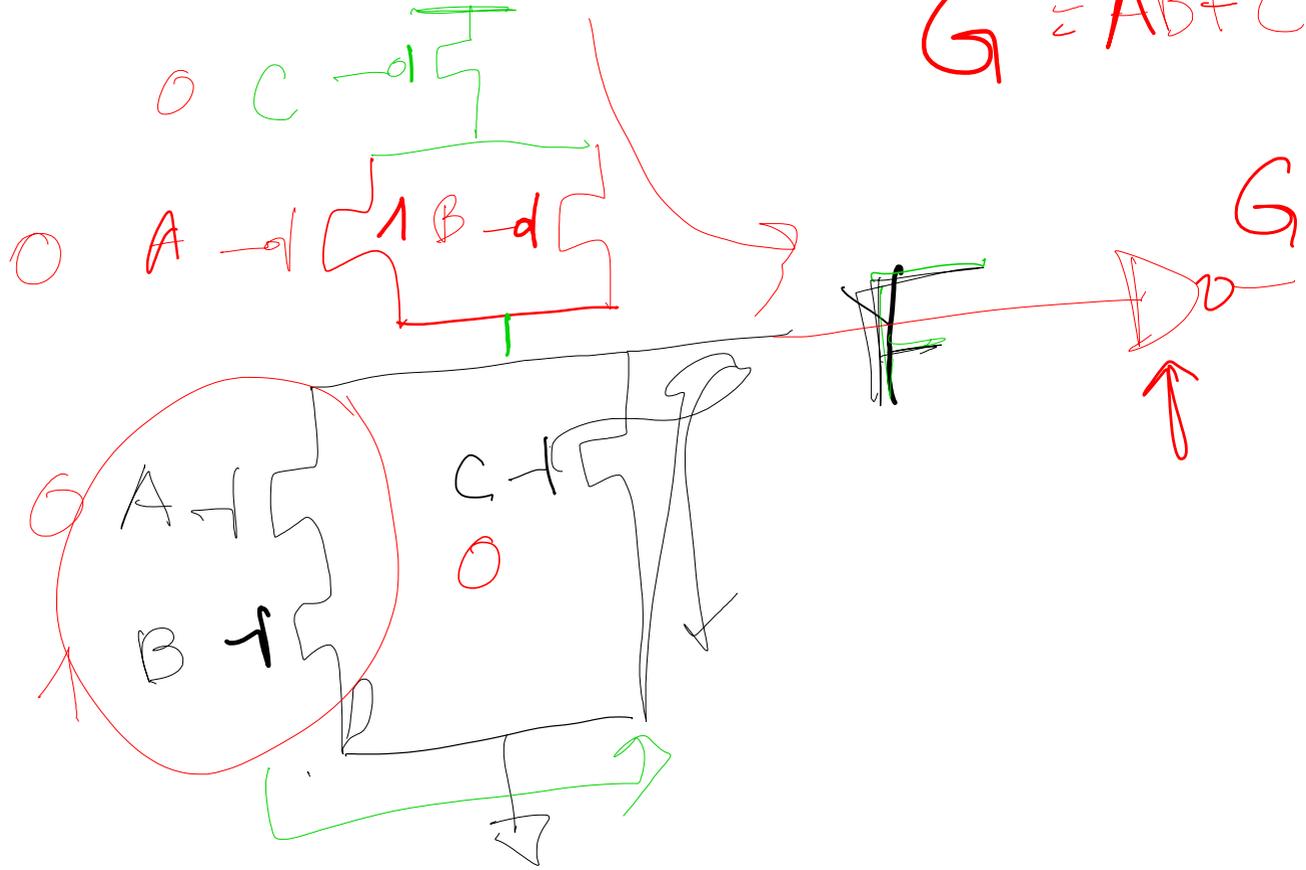
## NOR Gatter mit drei Eingängen



$$C = 1$$

$$F = AB + C$$

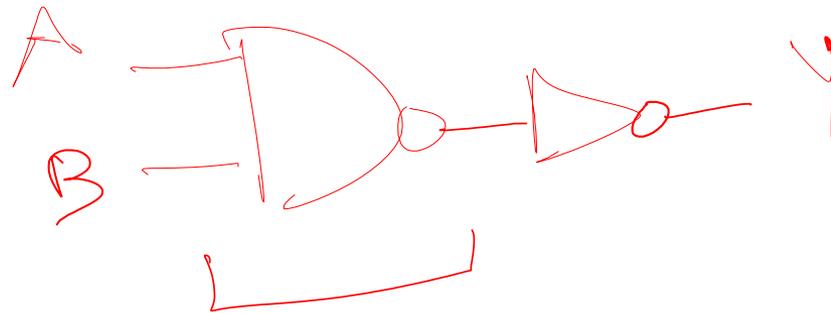
$$G = AB + C$$



# Aufbau eines AND-Gatters mit zwei Eingängen

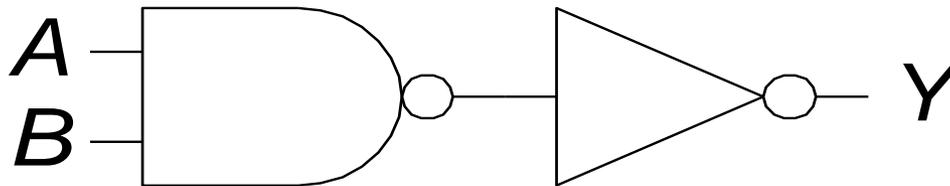


$$Y = AB$$



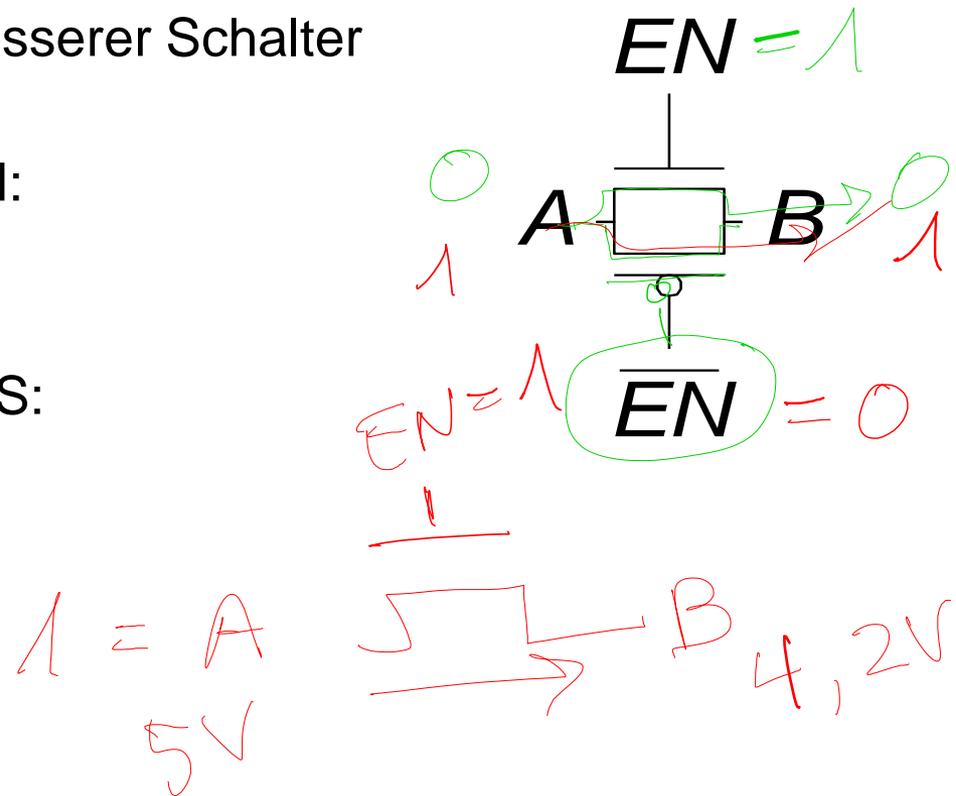
# Aufbau eines AND-Gatters mit zwei Eingängen

## AND Gatter mit zwei Eingängen



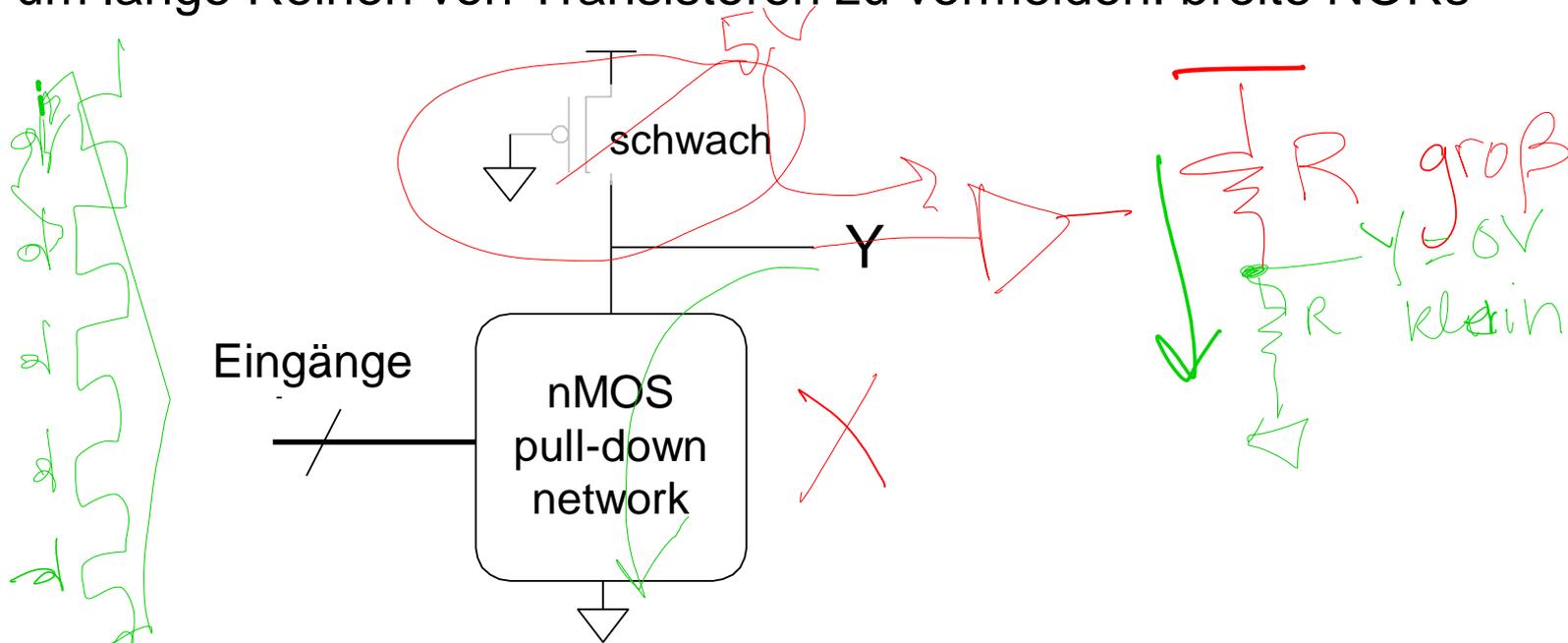
# Transmissionsgatter (*transmission gates*)

- nMOS leiten 1'en **schlecht** weiter
- pMOS leiten 0'en **schlecht** weiter
- **Transmissionsgatter** ist ein besserer Schalter
  - Leitet 0 und 1 gut weiter
- Wenn  $EN = 1$ , Schalter ist EIN:
  - $\overline{EN} = 0$
  - $A$  ist verbunden mit  $B$
- Wenn  $EN = 0$ , Schalter ist AUS:
  - $\overline{EN} = 1$
  - $A$  ist nicht verbunden mit  $B$



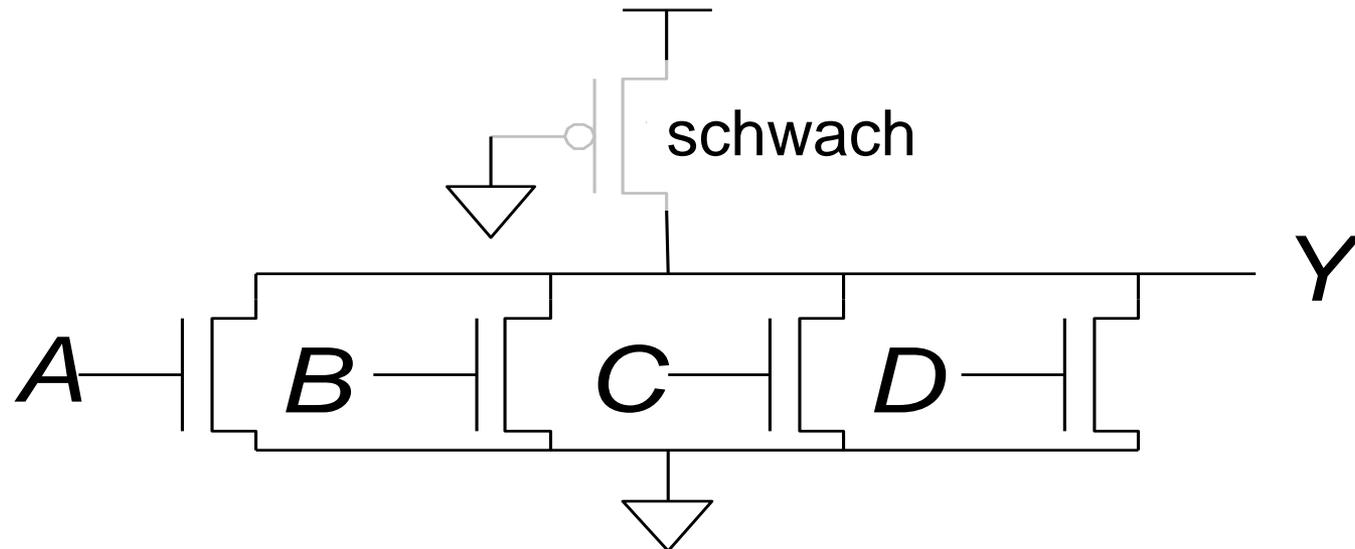
# Tricks: Pseudo-nMOS Gatter

- Pseudo-nMOS Gatter **ersetzen** das Pull-Up Netz
- Durch **schwachen immer eingeschalteten** pMOS Transistor
  - Schwach heißt: Seine 1 kann durch das Pull-Down Netz neutralisiert werden
- Nützlich um lange Reihen von Transistoren zu vermeiden: breite NORs



# Beispiel für Pseudo-nMOS Gatter

## Pseudo-nMOS NOR4



Verbraucht aber mehr Energie: Schwacher Dauerkurzschluss bei  $Y=0$

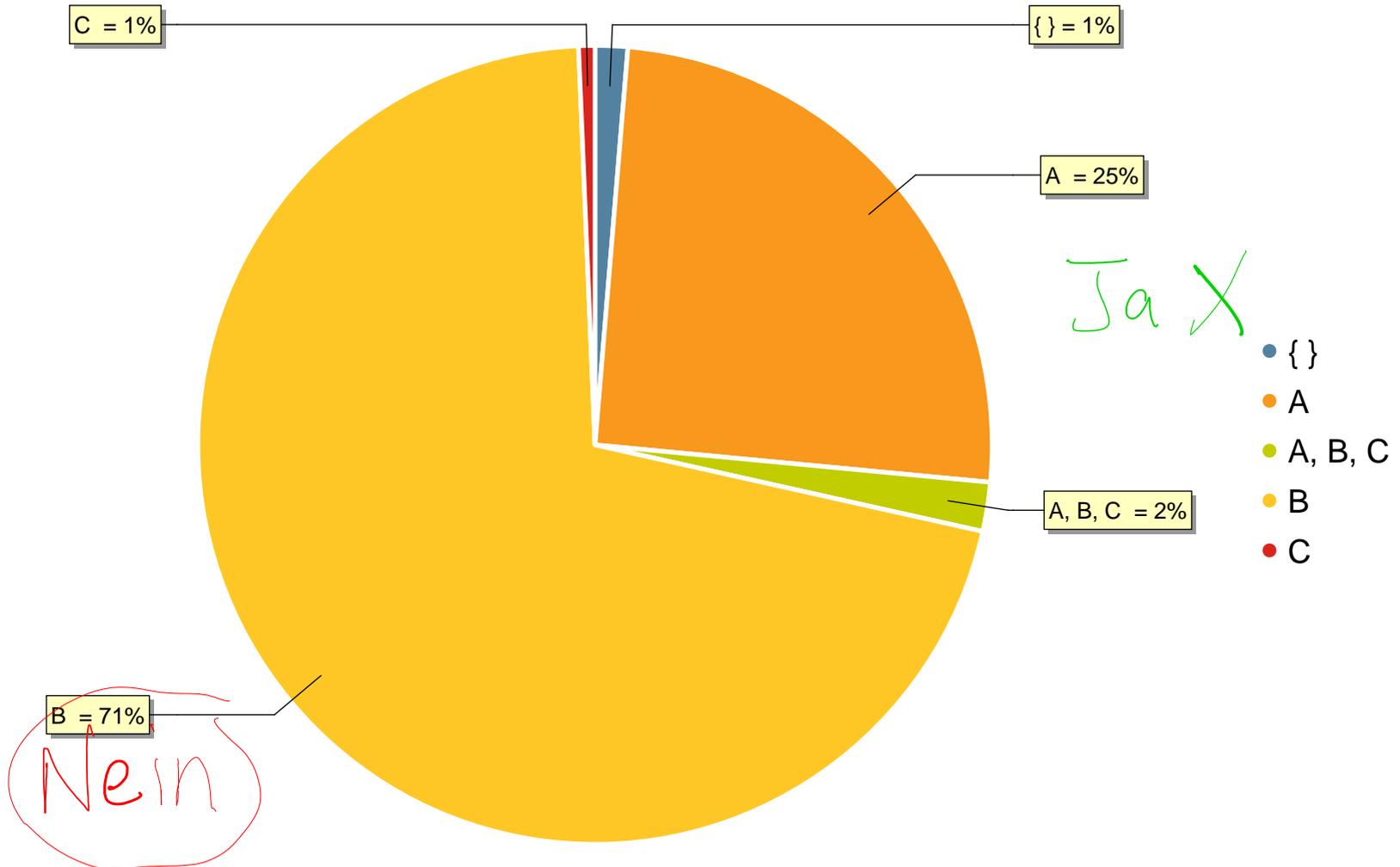
Kann man -18 mit einer 5-Bit Zweierkomplementzahl darstellen?

A) ja

B) nein

C) keine Ahnung

# Kann man -18 mit einer 5-Bit Zweierkomplementzahl darstellen?

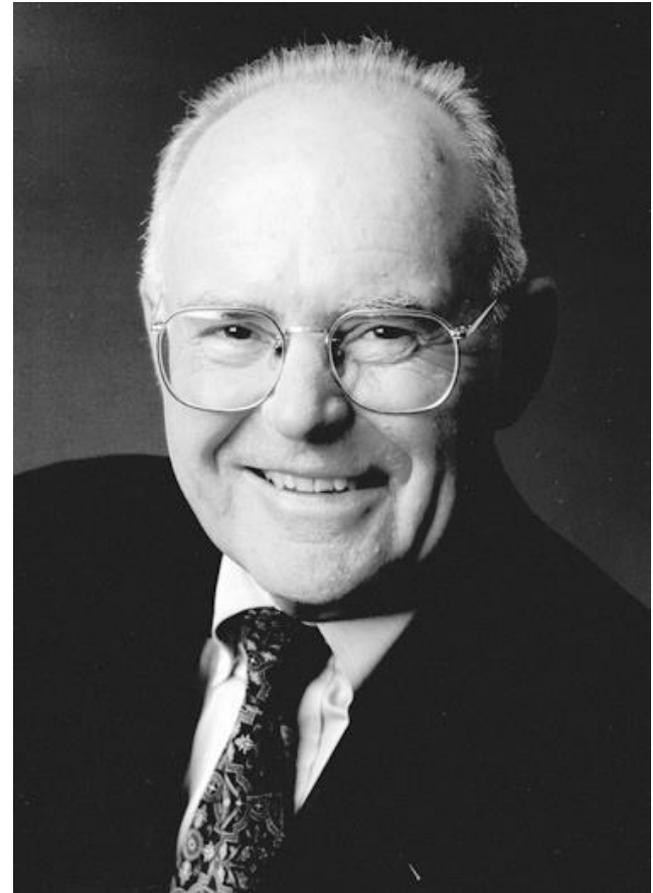


$$\begin{array}{r} \hline \hline \hline \hline \hline \\ = 16 \quad 8 \quad 4 \quad 2 \quad 1 \\ \\ 1 \quad 0 \quad 0 \quad 0 \quad 0 \end{array}$$

# Gordon Moore, 1929 -

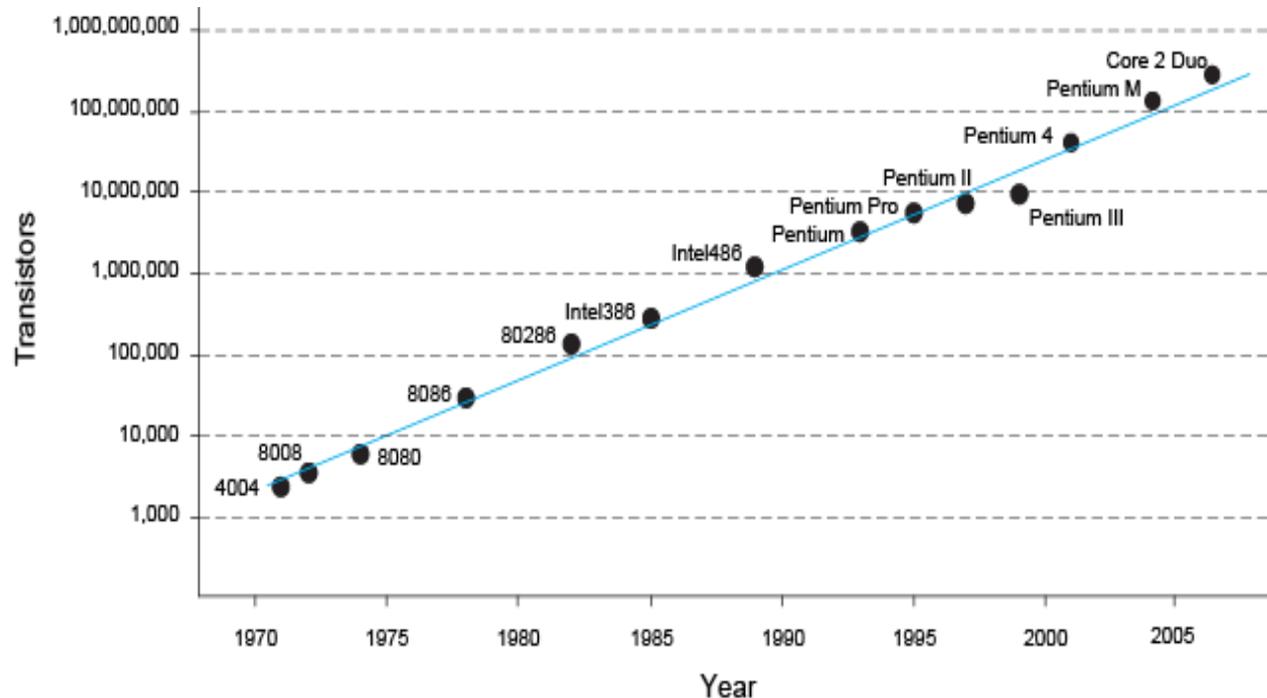


- Gründete Intel in 1968 zusammen mit Robert Noyce
- **Moores Gesetz:** Die Anzahl von Transistoren auf Chips verdoppelt sich
  - Jedes Jahr (1965)
  - Alle zwei Jahre (angepasst 1975)





# Moore's Law



- *“Wenn sich das Auto wie die Computer entwickelt hätte, würde ein Rolls-Royce heute \$100 kosten, 250 µl Benzin auf 100 km verbrauchen und einmal im Jahr explodieren ...”*

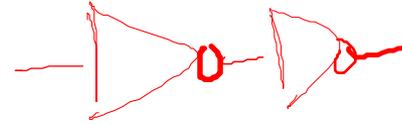
– Robert X. Cringely (Infoworld)

# Leistungsaufnahme

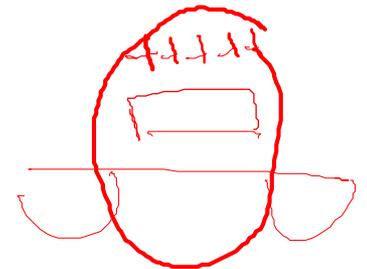
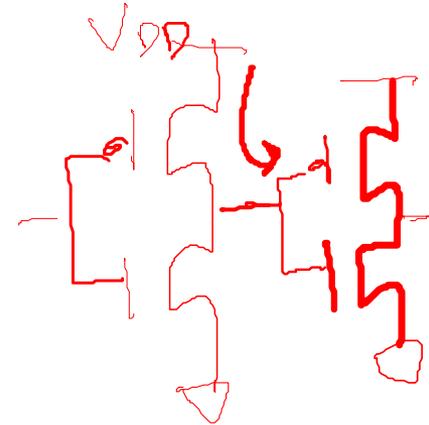


- **Leistung** = Energieverbrauch pro Zeiteinheit
- Zwei **Arten** der Leistungsaufnahme:
  - **Dynamische** Leistungsaufnahme
  - **Statische** Leistungsaufnahme

# Dynamische Leistungsaufnahme



- Leistung um Gates der Transistoren **umzuladen**
  - Wirken als **Kondensator**

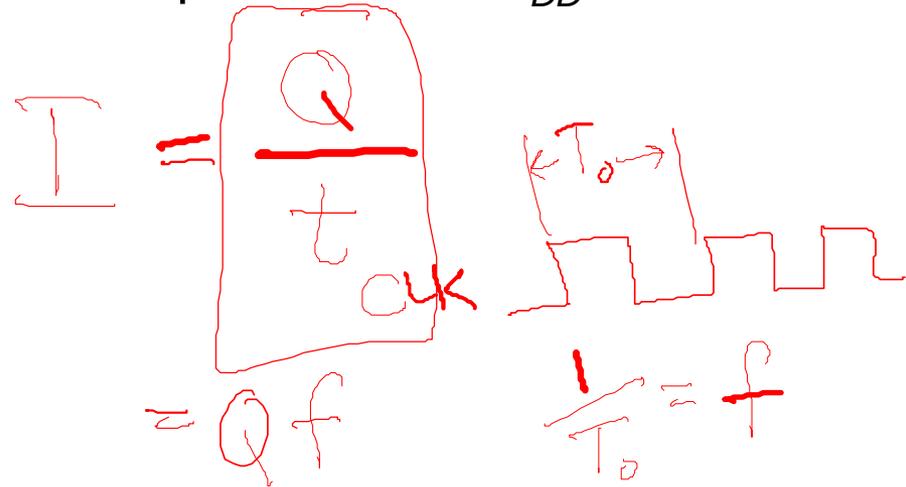


# Dynamische Leistungsaufnahme

- Leistung um Gates der Transistoren **umzuladen**
  - Wirken als **Kondensator**
  - **Ladungen** auf einen Kondensator der Kapazität  $C$  auf  $V_{DD}$  :

- $Q = CV_{DD}$

Ladungsträger



$I = \frac{Q}{t}$

$= Qf$

$\frac{1}{T_0} = f$

# Dynamische Leistungsaufnahme



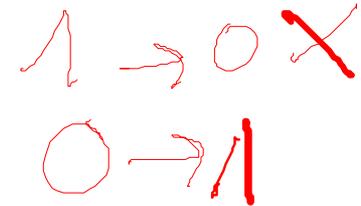
- Leistung um Gates der Transistoren **umzuladen**
  - Wirken als **Kondensator**
- **Ladungen** auf einen Kondensator der Kapazität  $C$  auf  $V_{DD}$  :
  - $Q = CV_{DD}$
- Schaltung wird mit **Frequenz  $f$**  betrieben
  - Transistoren schalten  $f$ -mal pro **Sekunde**
  - $I = Q/t = Qf = CV_{DD}f$

Power

$$P = I V_{DD}$$

# Dynamische Leistungsaufnahme

- Leistung um Gates der Transistoren **umzuladen**
  - Wirken als **Kondensator**
- **Ladungen** auf einen Kondensator der Kapazität  $C$  auf  $V_{DD}$  :
  - $Q = CV_{DD}$
- Schaltung wird mit **Frequenz  $f$**  betrieben
  - Transistoren schalten  $f$ -mal pro **Sekunde**
  - $I = Q/t = Qf = CV_{DD}f$
- Leistungsaufnahme ist:  $P = IV = \underbrace{( CV_{DD}f )}_I ( V_{DD} ) = \underbrace{CV_{DD}^2f}$



# Dynamische Leistungsaufnahme

- Leistung um Gates der Transistoren **umzuladen**
  - Wirken als **Kondensator**
- **Ladungen** auf einen Kondensator der Kapazität  $C$  auf  $V_{DD}$  :
  - $Q = CV_{DD}$
- Schaltung wird mit **Frequenz**  $f$  betrieben
  - Transistoren schalten  $f$ -mal pro **Sekunde**
  - $I = Q/t = Qf = CV_{DD}f$
- Leistungsaufnahme ist:  $P = IV = ( CV_{DD}f ) ( V_{DD} ) = CV_{DD}^2f$
- **Aber** aufladen kostet (0-1), entladen (1-0) kostet nichts
  - Hälfte der Zeit schalten die Gatter von 0 bis 1
- Die **dynamische** Leistungsaufnahme ist also:

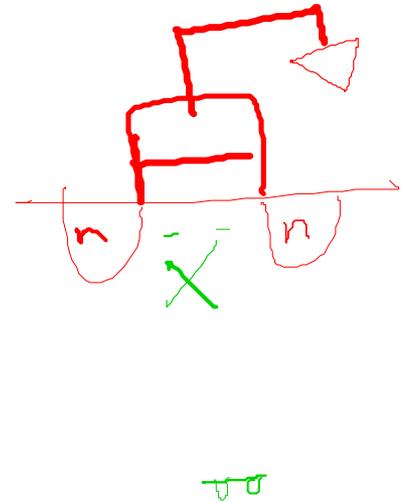
$$P_{dynamic} = \frac{1}{2} C V_{DD}^2 f$$

$\alpha = 0,5$

# Statische Leistungsaufnahme

- Leistungsbedarf wenn **kein** Gatter schaltet
- Wird verursacht durch den Leckstrom  $I_{DD}$ 
  - Immer **kleinere** Transistoren schalten nicht mehr **vollständig** ab
  - Pseudo-nMOS, ...
- **Statische** Leistungsaufnahme ist also

$$P_{static} = I_{DD} V_{DD}$$



# Beispielrechnung Leistungsaufnahme



- Abschätzen der Leistungsaufnahme für ein Netbook
- Parameter
  - Versorgungsspannung  $V_{DD} = 1,2 \text{ V}$
  - Transistorkapazität  $C = 20 \text{ nF}$
  - Taktfrequenz  $f = 1 \text{ GHz}$
  - Leckstrom  $I_{DD} = 20 \text{ mA}$

# Beispielrechnung Leistungsaufnahme

- Abschätzen der Leistungsaufnahme für ein Netbook
- Parameter
  - Versorgungsspannung  $V_{DD} = 1,2 \text{ V}$
  - Transistorkapazität  $C = 20 \text{ nF}$
  - Taktfrequenz  $f = 1 \text{ GHz}$
  - Leckstrom  $I_{DD} = 20 \text{ mA}$

$$\begin{aligned} P &= \frac{1}{2} C V_{DD}^2 f + I_{DD} V_{DD} \\ &= \frac{1}{2} (20 \text{ nF}) (1,2 \text{ V})^2 (1 \text{ GHz}) + (20 \text{ mA})(1,2 \text{ V}) \\ &= (14,4 + 0,024) \text{ W} \\ &\approx \mathbf{14,4 \text{ W}} \end{aligned}$$

# Quiz



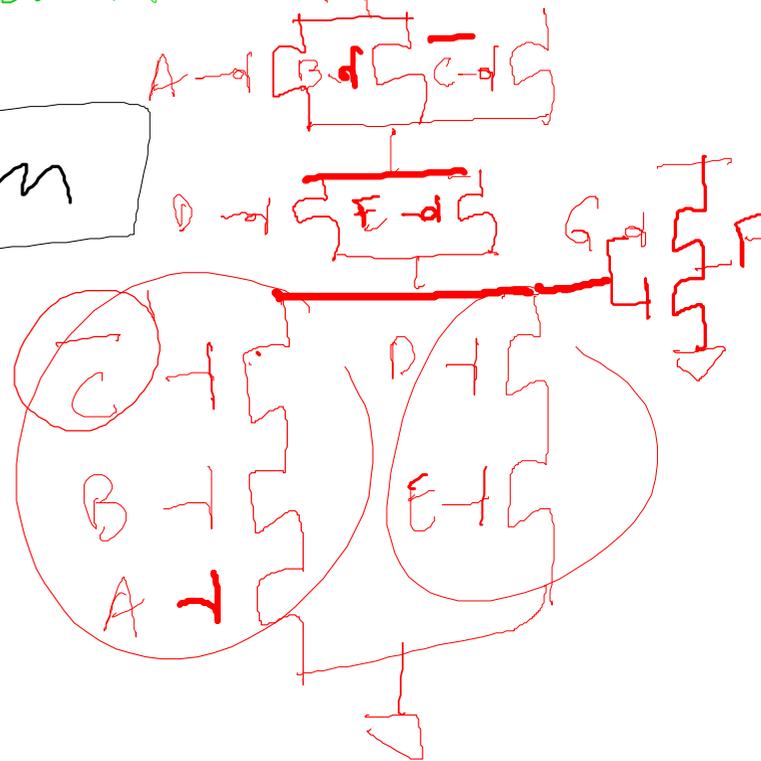
$$F = \underline{A} \underline{B} \overline{C} + \underline{D} \underline{E}$$

$$5 \times 2 = 10$$

Wieviele Transistoren?

12 Transistoren

$$\underline{G} = \overline{A} \overline{B} \overline{C} + \underline{D} \underline{E}$$



**Denkanstoß:** Wie würde man diese Funktion als CMOS Gatter bauen? Sollte man auch die kleinste Anzahl von Transistoren benutzen.

$$W = \overline{A} \overline{B} C + \overline{D} \overline{E}$$