Technische Grundlagen der Informatik – Kapitel 1

Prof. Dr. Andreas Koch
Fachbereich Informatik
TU Darmstadt

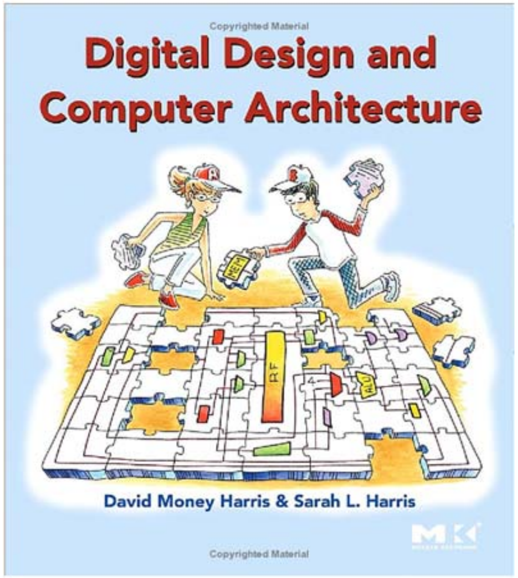
TECHNISCHE UNIVERSITÄT DARMSTADT

Lehr- und Anschauungsmaterial

- Aus dem Lehrbuch

  Digital Design and Computer Architecture

  von David M. Harris & Sarah L. Harris



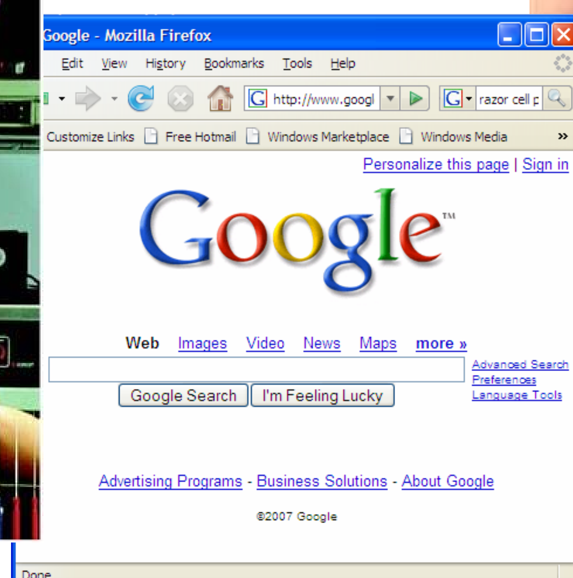
- Diese Folien nach englischen Originalvorlagen erstellt
  - Originale sind © 2007 Elsevier
- Buch wird an Studierende subventioniert abgegeben
  - Organisiert durch Fachschaft Informatik
- Mehr Hintergrundmaterial auf Web-Seite zu Buch

Kapitel 1: Von 0 nach 1

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

Hintergrund

- Mikroprozessoren haben die Welt verändert
  - Handys, Internet, Medizintechnik, Unterhaltung, …
- Umsatzwachstum in der Halbleiterindustrie von $21 Milliarden in 1985 auf $213 Milliarden in 2004



Themen dieser Veranstaltung

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

Beherrschen von Komplexität

- 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

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

Disziplin

- 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, …

Wesentliche Techniken (Die Drei-Y's)

- Hierarchie
  - Aufteilen eines Systems in Module und Untermodule
- Modularität
  - Wohldefinierte Schnittstellen und Funktionen
- Regularität
  - Bevorzuge einheitliche Lösungen für einfachere Wiederverwendbarkeit

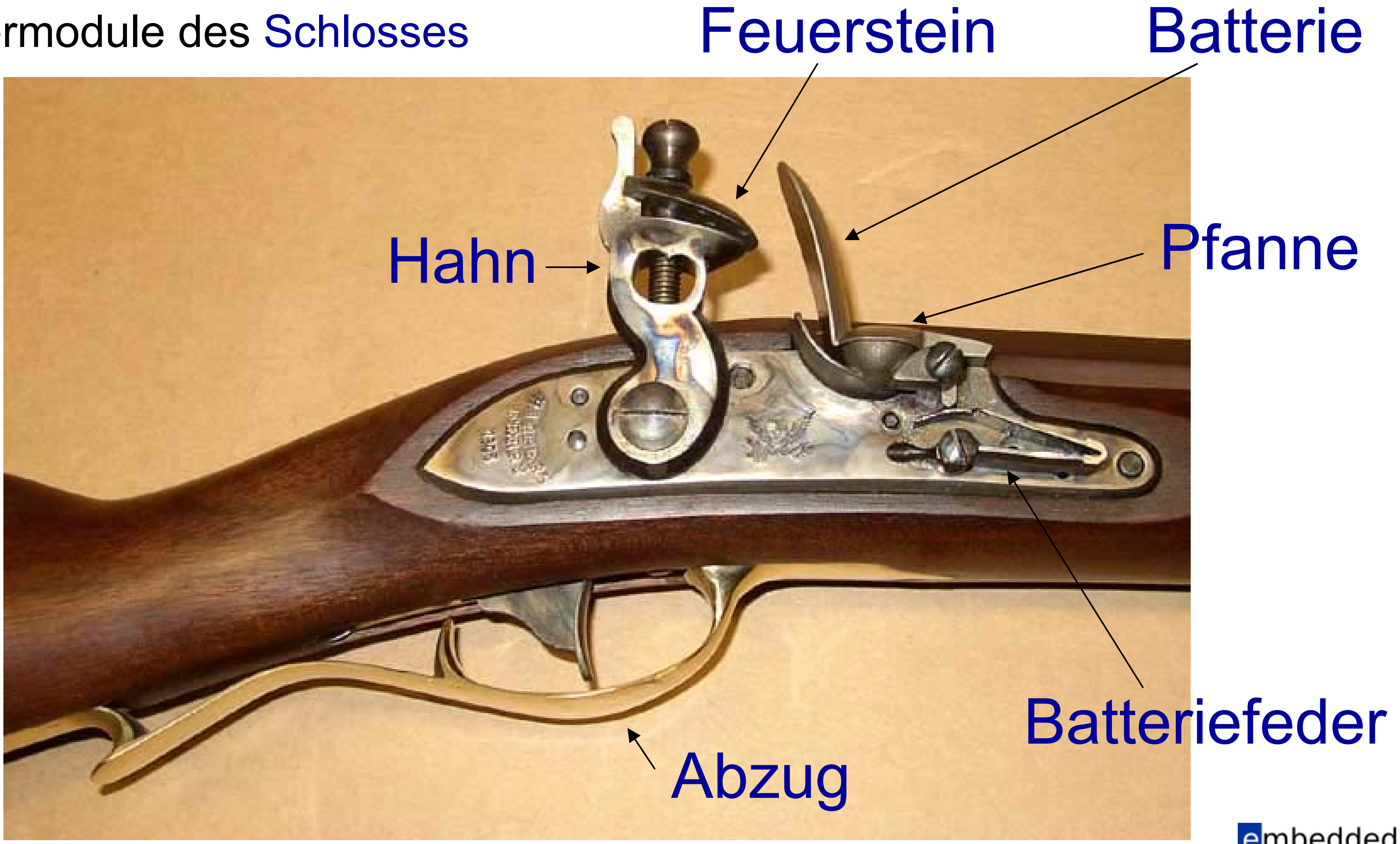
Beispiel: Steinschlossgewehr

- 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

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

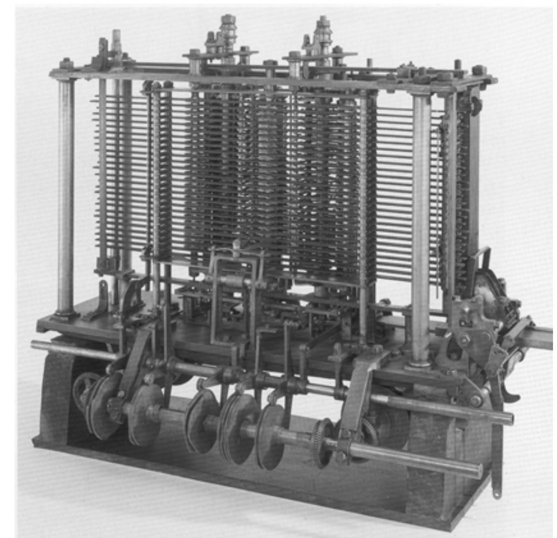
- Gleiche Schlösser in unterschiedlichen Schäften
  - Passender Ausschnitt in Schaft
- Unterschiedliche Läufe in gleichen Schäften
  - Passende Länge und Haltemechanismus
- Voraussetzung für industrielle Massenproduktion

Digitale Abstraktion

- Die meisten physikalischen Größen habe 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: Binärwerte

- Digitale Disziplin heute
  - 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

George Boole, 1815 - 1864

- In einfachen Verhältnissen geboren
- Brachte sich selbst Mathematik bei
- Später Professur 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

Scanned at the American Institute of Physics

Zahlensysteme

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dezimalzahlen

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

$$5374_{10} =$$

- Binärzahlen

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

$$1101_2 =$$

embedded systems + applications

Zahlensysteme

- Dezimalzahlen

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

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

Zahlenkonvertierung

- Binär nach dezimal umrechnen:
  - Wandele 10011_2 ins Dezimalsystem um

- Dezimal nach binär umrechnen
  - Wandele 47_{10} ins Binärsystem um

Zahlenkonvertierung

- Binär nach dezimal umrechnen:
  - Wandele 10011_2 ins Dezimalsystem um
  - **$16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$**

- Dezimal nach binär umrechnen
  - Wandele 47_{10} ins Binärsystem um
  - **$32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$**
  - Auf zwei Arten möglich
    - Jeweils nach größter noch passender Zweierpotenz suchen
    - Durch immer größer werdende Zweierpotenzen dividieren

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	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

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

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

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
  - $16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$

Bits, Bytes, Nibbles…

- Bits (Einheit *b*)
  - Höchstwertiges Bit (*msb*)
  - Niedrigstwertiges Bit (*lsb*)

10010110

most significant bit

least significant bit

- Bytes (Einheit *B*) & Nibbles

byte

10010110

nibble

- Bytes
  - Höchstwertiges Byte (*MSB*)
  - Niedrigstwertiges Byte (*LSB*)

CEBF9AD7

most significant byte

least significant byte

Zweierpotenzen und Präfixe

▪ 2^{10} = 1 Kilo	(K)	≈ 1000 (1024)
▪ 2^{20} = 1 Mega	(M)	≈ 1 Million (1,048,576)
▪ 2^{30} = 1 Giga	(G)	≈ 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 Millionen

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

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

Addition

- Dezimal

$$
\begin{array}{rr}
 & \color{blue}{11}\ \ \ \leftarrow \text{Überträge} \\
 & 3734 \\
+ & 5168 \\
\hline
 & 8902
\end{array}
$$

- Binär

$$
\begin{array}{rr}
 & \color{blue}{11}\ \ \ \leftarrow \text{Überträge} \\
 & 1011 \\
+ & 0011 \\
\hline
 & 1110
\end{array}
$$

Beispiele für Addition von Binärzahlen

- Addiere die 4-bit Binärzahlen

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

- Addiere die 4-bit Binärzahlen

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

Beispiele für Addition von Binärzahlen

- Addiere die 4-bit Binärzahlen

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

- Addiere die 4-bit Binärzahlen

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

Überlauf!

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

Vorzeichenbehaftete Binärzahlen

- Darstellung als Vorzeichen und Betrag
- Zweierkomplement

Darstellung als Vorzeichen und Betrag

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

$$A: \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

- Vorzeichenbit ist höchstwertiges Bit (msb)
  - Positive Zahl: Vorzeichenbit = 0
  - Negative Zahl: Vorzeichenbit = 1

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

- Beispiel: 4-bit Vorzeichen/Betrag-Darstellung von ± 6:

  +6 =

  - 6 =

- Wertebereich einer Zahl in Vorzeichen/Betrag-Darstellung :

Darstellung als Vorzeichen und Betrag

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

$$A:\{a_{N-1},a_{N-2},\cdots a_2,a_1,a_0\}$$

- Vorzeichenbit ist höchstwertiges Bit (msb)
  - Positive Zahl: Vorzeichenbit = 0
  - Negative Zahl: Vorzeichenbit = 1

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

- Beispiel: 4-bit Vorzeichen/Betrag-Darstellung von ± 6:

  +6 = **0110**

  - 6 = **1110**

- Wertebereich einer Zahl in Vorzeichen/Betrag-Darstellung :

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

Darstellung als Vorzeichen/Betrag: Probleme

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

```
   1110
 + 0110
 ------
  10100 (falsch!)
```

- Zwei Darstellungen für Null (± 0):

1000	(-0)
0000	(+0)

Zahlendarstellung im Zweierkomplement

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

Zahlendarstellung im Zweierkomplement

- Wie vorzeichenlose Binärdarstellung, aber …
  - msb hat nun einen Wert von -2^{N-1}

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

- Größte positive 4b Zahl :
- Kleinste negative 4b Zahl :
- msb gibt immer noch das Vorzeichen an
  - 1=negativ, 0=positiv

- Wertebereich einer *N*-bit Zweierkomplementzahl:

Zahlendarstellung im Zweierkomplement

- Wie vorzeichenlose Binärdarstellung, aber …
  - msb hat nun einen Wert von -2^{N-1}

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

- Größte positive 4b Zahl : **0111 = $2^2 + 2^1 + 2^0$ = 7**
- Kleinste negative 4b Zahl : **1000 = -2^3 = -8**
- msb gibt immer noch das Vorzeichen an
  - 1=negativ, 0=positiv

- Wertebereich einer *N*-bit Zweierkomplementzahl:

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

Darstellung im Zweierkomplement

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

- Positive Zahlen $k >= 0$
  - Normale Binärdarstellung, restliche Bits bis einschließlich msb mit 0 auffüllen
  - Beispiel: N = 5b, $k = 3_{10}$ ➔ 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 binär in untere Bits eintragen (Beginn bei lsb)
    - Beispiel: N = 5b, $k = -3_{10}$ ➔ $d = 2^4-3 = 16-3 = 13$ ➔ z = 11101

Zweierkomplement arithmetisch bilden

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

Zweierkomplement arithmetisch bilden

- In beide Richtungen anwendbar
  - Vorzeichenwechsel: *k* ➔ *-k*

- Algorithmus
  1. Alle Bits invertieren (0➔1, 1➔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 $6_{10} = 0110_2$

- Was ist der Dezimalwert der Zweierkomplementzahl 1001_2?

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

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

Addition im Zweierkomplement

- Addiere 6 + (-6)

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

- Addiere -2 + 3

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

Addition im Zweierkomplement

- Addiere 6 + (-6)

$$
\begin{array}{r}
{\color{blue}111} \\
0110 \\
+\quad 1010 \\
\hline
{\color{gray}1}0000
\end{array}
$$

- Addiere -2 + 3

$$
\begin{array}{r}
{\color{blue}111} \\
1110 \\
+\quad 0011 \\
\hline
{\color{gray}1}0001
\end{array}
$$

Überlauf:
Ignorieren, wenn Positive und negative Zahlen gleicher Bitbreite addiert werden

Erweitern von Zahlen auf höhere 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 = 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

TECHNISCHE UNIVERSITÄT DARMSTADT

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

- **Beispiel 1:**
  - 4-bit Wert = $0011_2 = 3_{10}$
  - 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!

embedded systems + applications

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:

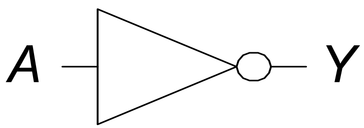
	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vorzeichenlos									0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Zweierkomplement	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111								
Vorzeichen/Betrag		1111	1110	1101	1100	1011	1010	1001	0000 1000	0001	0010	0011	0100	0101	0110	0111								

Logische Gatter

- Berechnen logische Funktionen:
  - Inversion (NICHT), UND, ODER, …
  - NOT, AND, OR, NAND, NOR, …
- Ein Eingang:
  - NOT Gatter, Puffer (*buffer*)
- Zwei Eingänge:
  - AND, OR, XOR, NAND, NOR, XNOR
- Viele Eingänge

Logikgatter mit einem Eingang

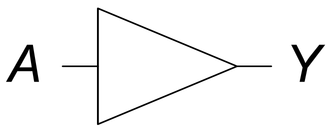
NOT



$$Y = \overline{A}$$

A	Y
0	
1	

BUF

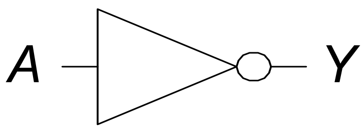


$$Y = A$$

A	Y
0	
1	

Logikgatter mit einem Eingang

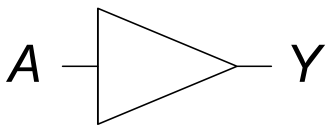
NOT



$Y = \overline{A}$

A	Y
0	1
1	0

BUF



$Y = A$

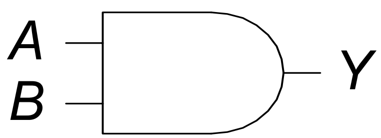
A	Y
0	0
1	1

Alternative Schreibweisen

Y=!A, Y=~A, Y=¬A, Y=A‘

Logikgatter mit zwei Eingängen

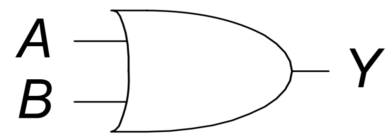
AND



$$Y = AB$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

OR

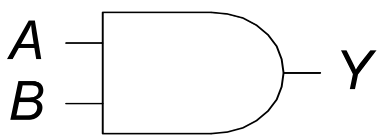


$$Y = A + B$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Logikgatter mit zwei Eingängen

AND



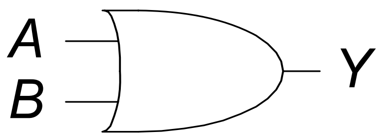
$$Y = AB$$

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 ∩ B

OR



$$Y = A + B$$

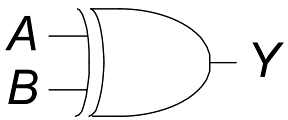
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Alternative Schreibweisen

Y= A | B, Y = A U B

Weitere Logikgatter mit zwei Eingängen

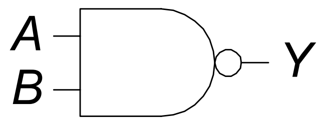
XOR



$$Y = A \oplus B$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

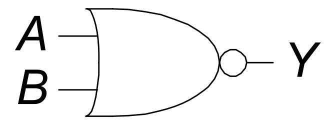
NAND



$$Y = \overline{AB}$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

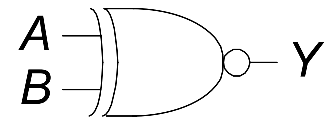
NOR



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

A	B	Y
0	0	
0	1	
1	0	
1	1	

XNOR

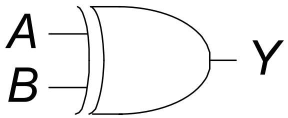


$$Y = \overline{A \oplus B}$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Weitere Logikgatter mit zwei Eingängen

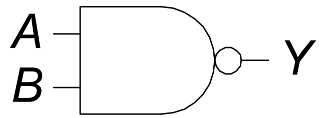
XOR



$$Y = A \oplus B$$

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

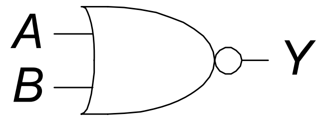
NAND



$$Y = \overline{AB}$$

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

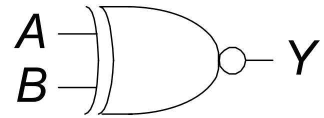
NOR



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

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

XNOR



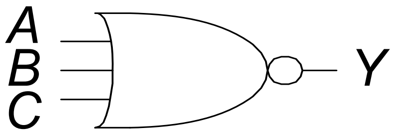
$$Y = \overline{A \oplus B}$$

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

Alternative
Schreibweise
Y= A ^ B

Logikgatter mit mehr als zwei Eingängen

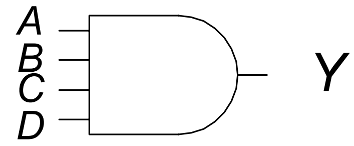
NOR3



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

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

AND4

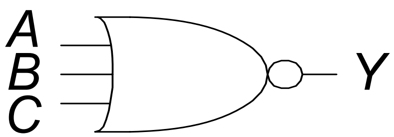


$$Y = ABCD$$

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

Logikgatter mit mehr als zwei Eingängen

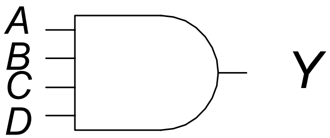
NOR3



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

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

AND4



$$Y = ABCD$$

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

Darstellung von Binärwerten durch Spannungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

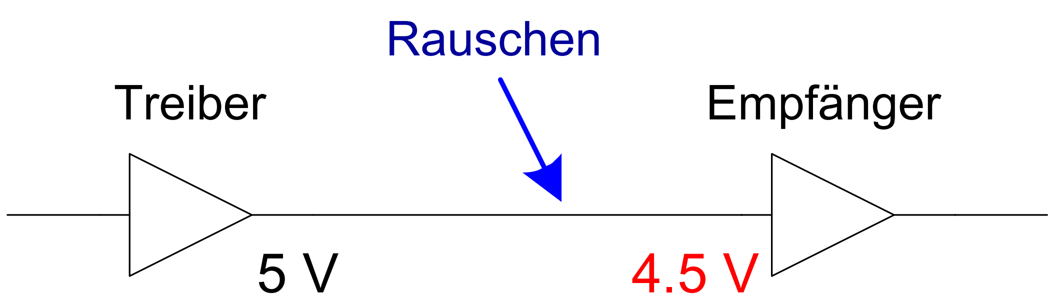
- Probleme
  - Wofür steht 4,99 V? Den Wert 0 oder 1?
  - Wofür steht 3,2V?

- Reale Schaltungen haben keine ganz exakten Spannungspegel
  - Teils sogar Umgebungsabhängig (Temperatur, Einstreuen, …)
  - Solche Spannungsschwankungen werden Rauschen genannt

embedded systems + applications

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



Darstellung von Binärwerten durch Spannungen

- Lösung
  - Statt einzelner Spannungspegel für 0 und 1 …
  - … verwende Bereiche von Spannungspegeln für 0 und 1

- Steigere Robustheit durch unterschiedliche Bereiche für
  - Eingänge
  - Ausgänge

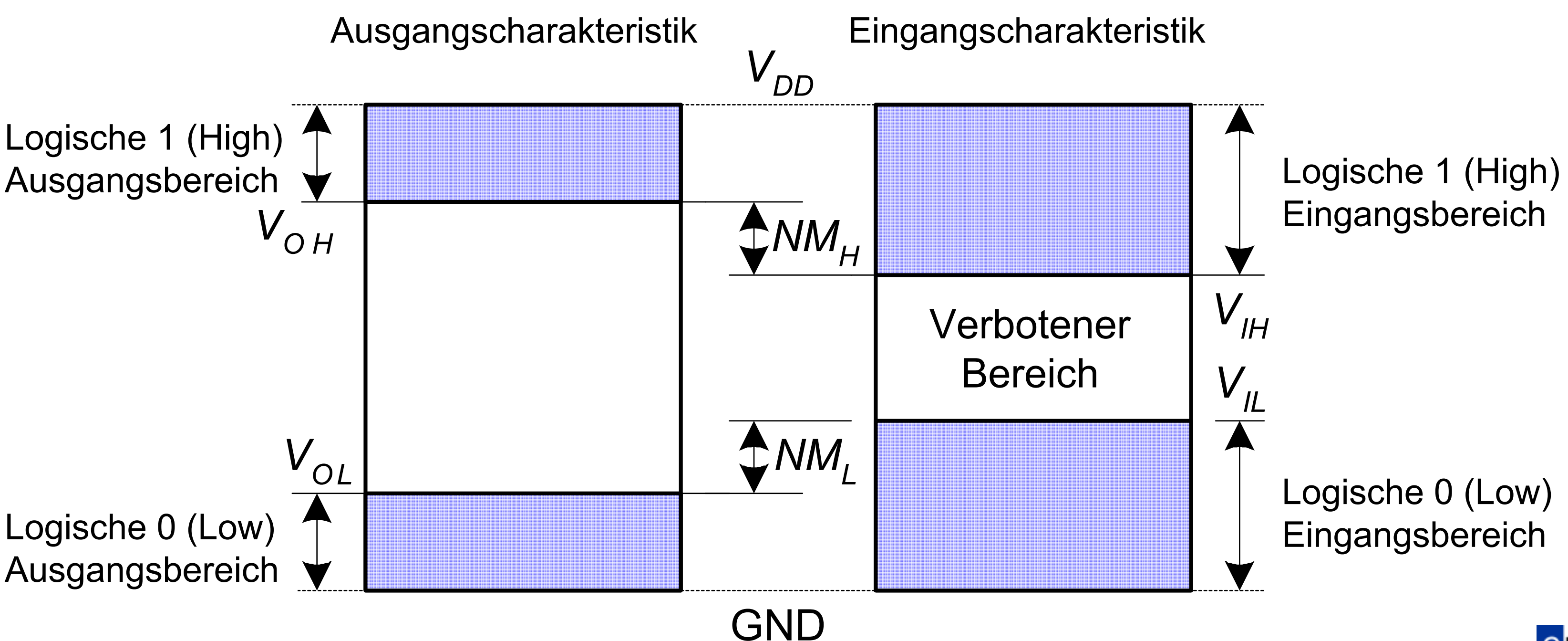
Statische Entwurfsdisziplin

- Jedes Schaltungselement muss bei Eingabe gültiger Logikpegel auch am Ausgang einen gültigen Logikpegel liefern

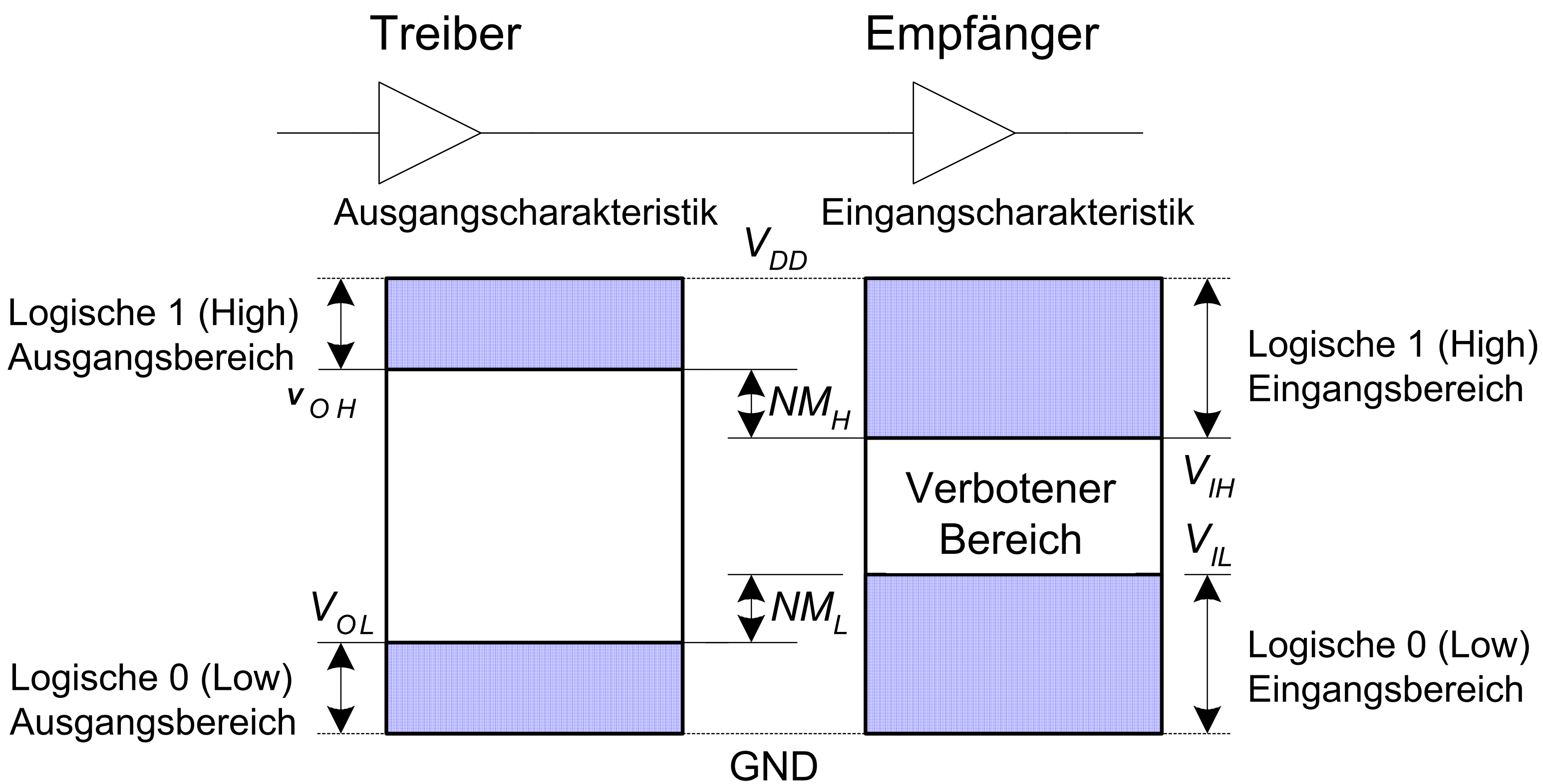
- Verwende nur einen Satz Spannungsbereiche für Logikpegel in gesamter Schaltung
  - Wird manchmal bewusst missachtet
    - Optimierung von Platz, Geschwindigkeit, Energiebedarf, Kosten, …
  - … bedarf aber großer Vorsicht

Logikpegel


Treiber
Empfänger


Ausgangscharakteristik
Eingangscharakteristik
V_{DD}
Logische 1 (High) Ausgangsbereich
V_{OH}
NM_H
Logische 1 (High) Eingangsbereich
V_{IH}
Verbotener Bereich
V_{IL}
V_{OL}
NM_L
Logische 0 (Low) Ausgangsbereich
Logische 0 (Low) Eingangsbereich
GND

Störabstand (*noise margin*)

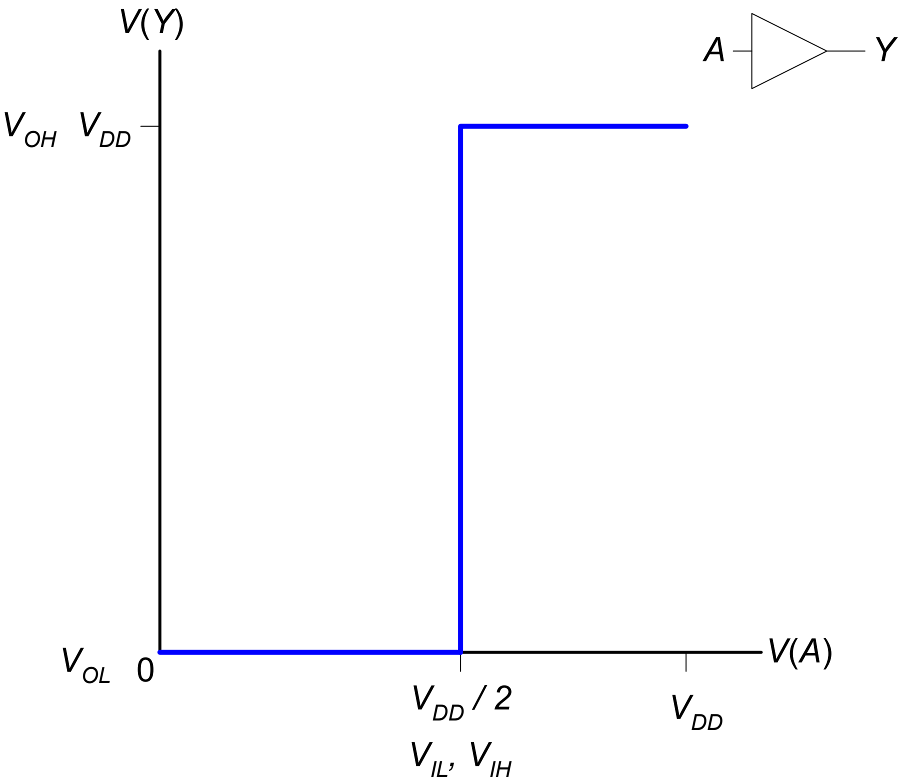


$$NM_H = V_{OH} - V_{IH}$$

$$NM_L = V_{IL} - V_{OL}$$

Gleichstrom-Transferkurve
(*DC transfer characteristics*)

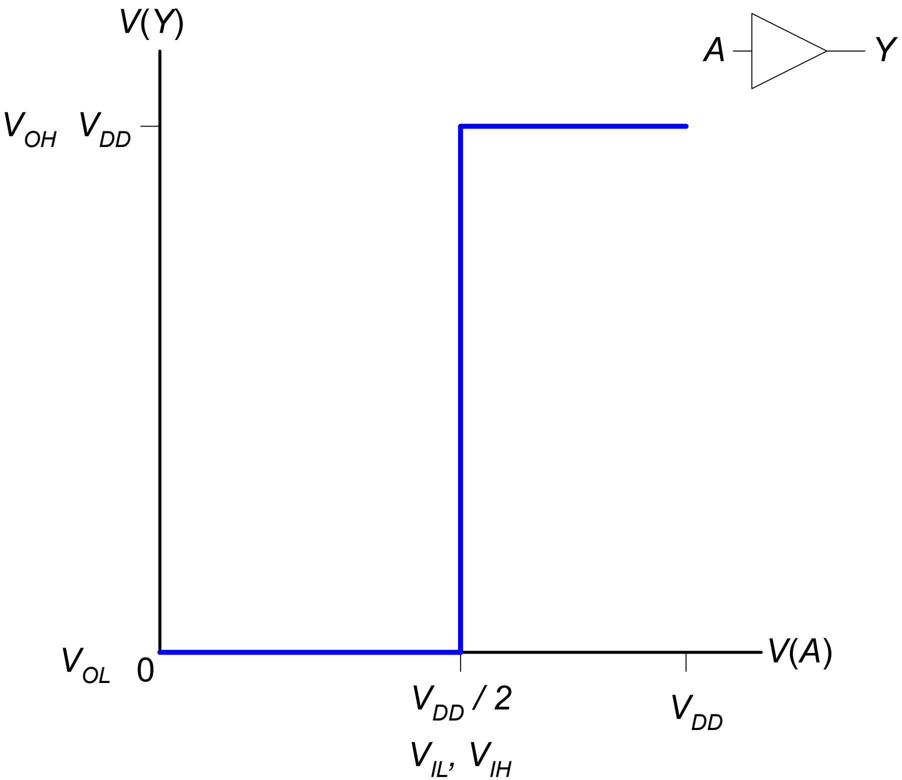
Idealer Buffer:



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

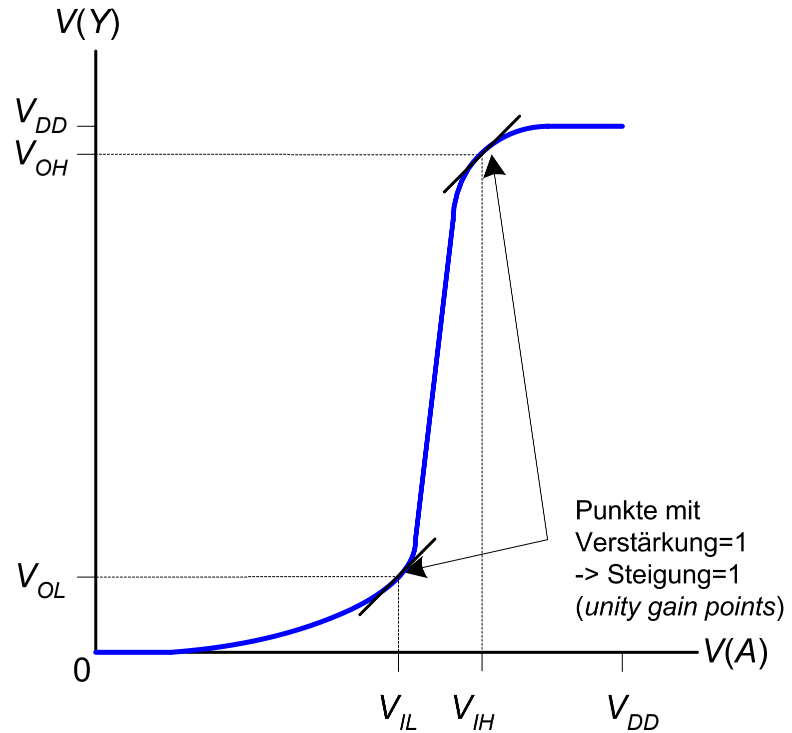
Gleichstrom-Transferkurve
(*DC transfer characteristics)*

Idealer Buffer:



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

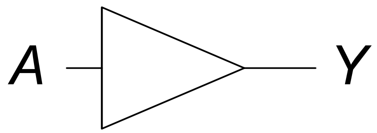
Realer Buffer:

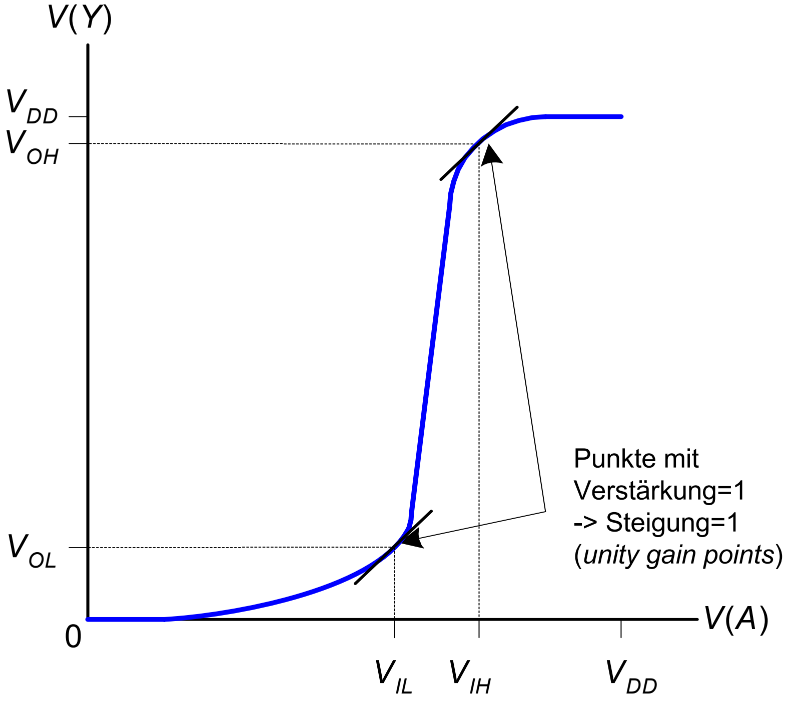


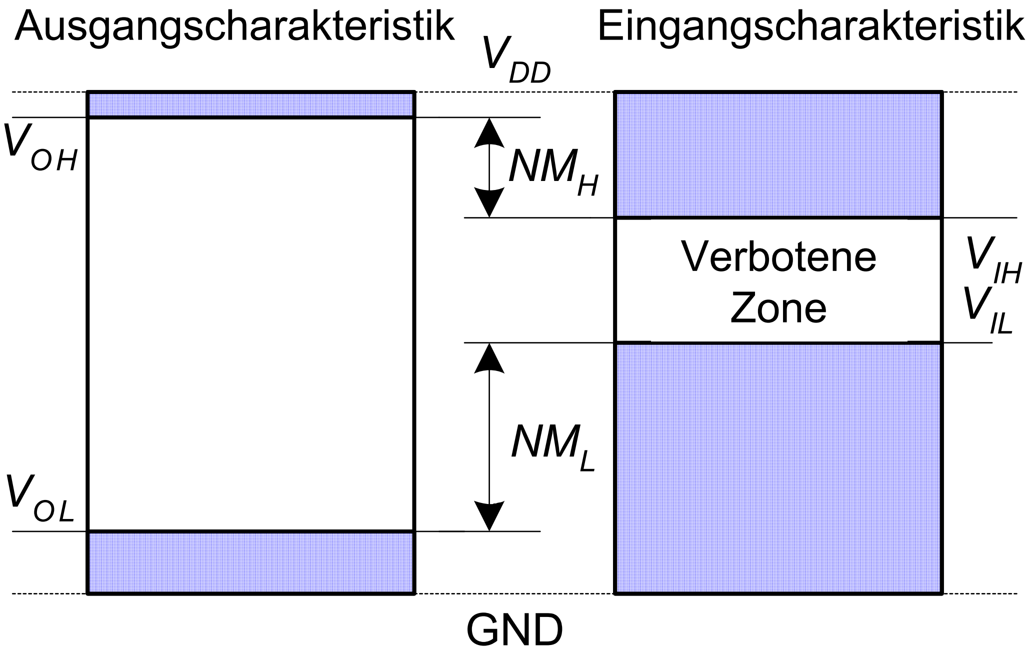
$NM_H, NM_L < V_{DD}/2$

Gleichstrom-Transferkurve

TECHNISCHE UNIVERSITÄT DARMSTADT







Absenken der Versorgungsspannung V_{DD}

- Versorgungsspannung in den 70er-80er Jahren: $V_{DD} = 5$ 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!

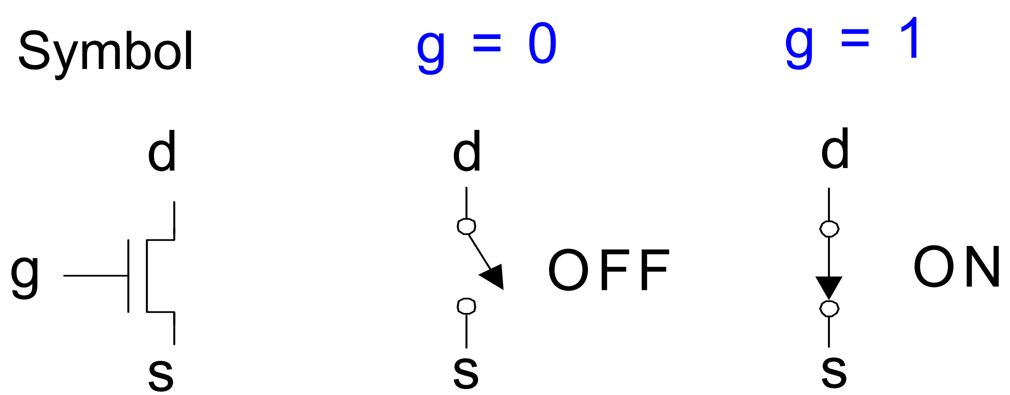
Beispiele für Logikfamilien

- 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	3.84
LVTTL	3.3 (3 - 3.6)	0.8	2.0	0.4	2.4
LVCMOS	3.3 (3 - 3.6)	0.9	1.8	0.36	2.7

Transistoren

- Logikgatter werden üblicherweise aus Transistoren aufgebaut
  - Heute überwiegend Feldeffekttransistoren (FET)
  - Weiteres bezieht sich implizit auf FETs, nicht Bipolartransistoren
- Transistoren sind spannungsgesteuerter 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

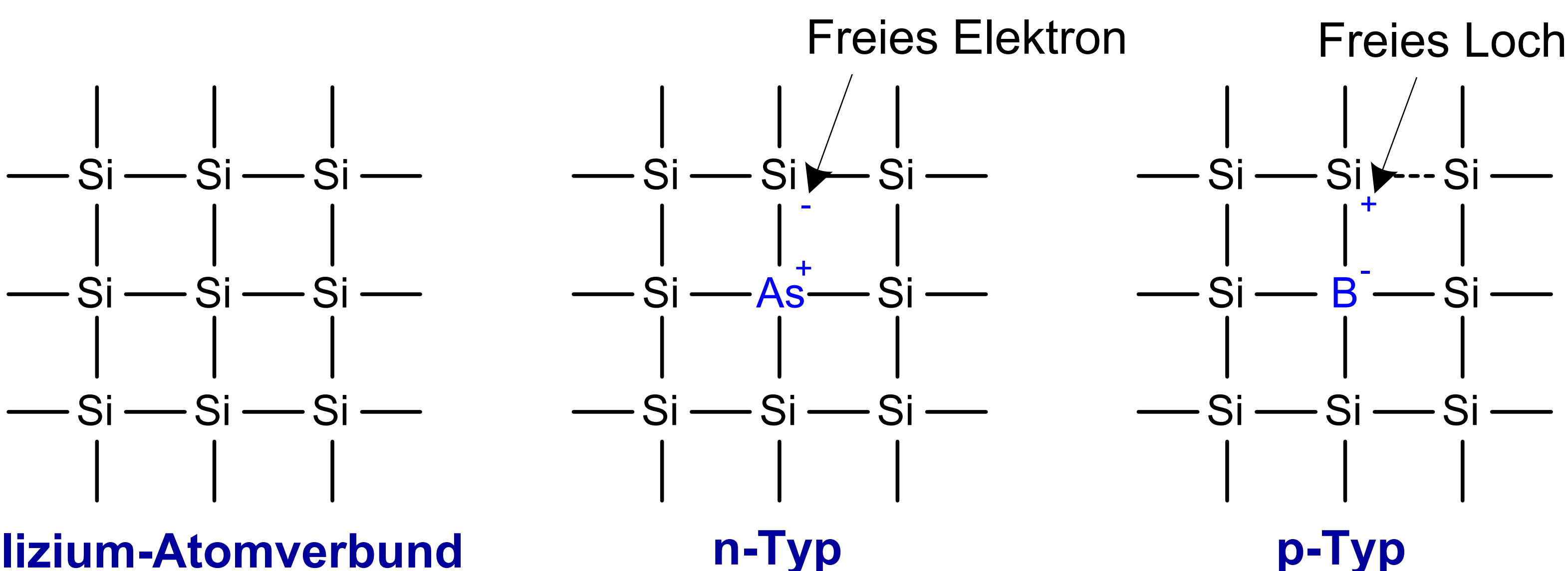


Robert Noyce, 1927 - 1990

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

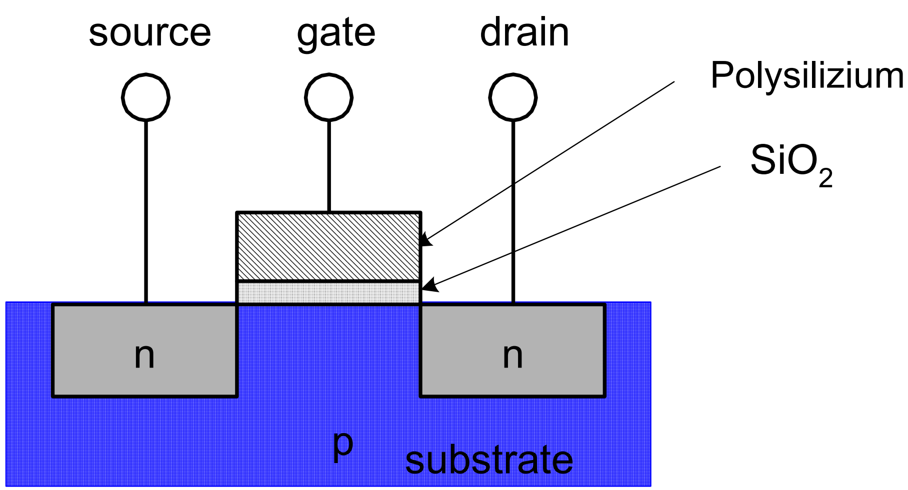
Silizium

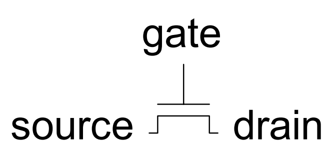
- Transistoren werden üblicherweise aus Silizium (Si, Gruppe IV) gefertigt
- Reines Silizium ist ein schlechter Leiter (keine freien Ladungsträger)
- Dotiertes Silizium ist ein guter Leiter (freie Ladungsträger)
  - n-Typ (freie *negative* Ladungsträger, Elektronen, dotiert mit Arsen, Gruppe V)
  - p-Typ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





nMOS

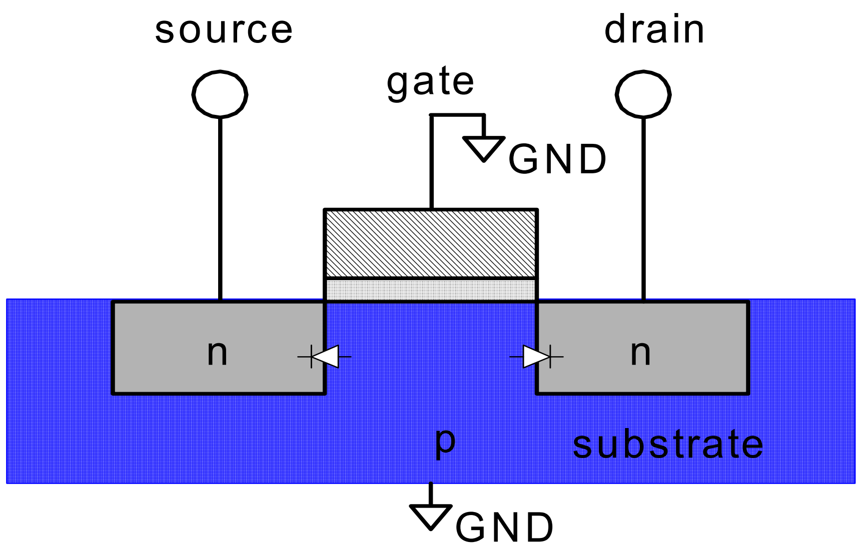
Transistor: nMOS

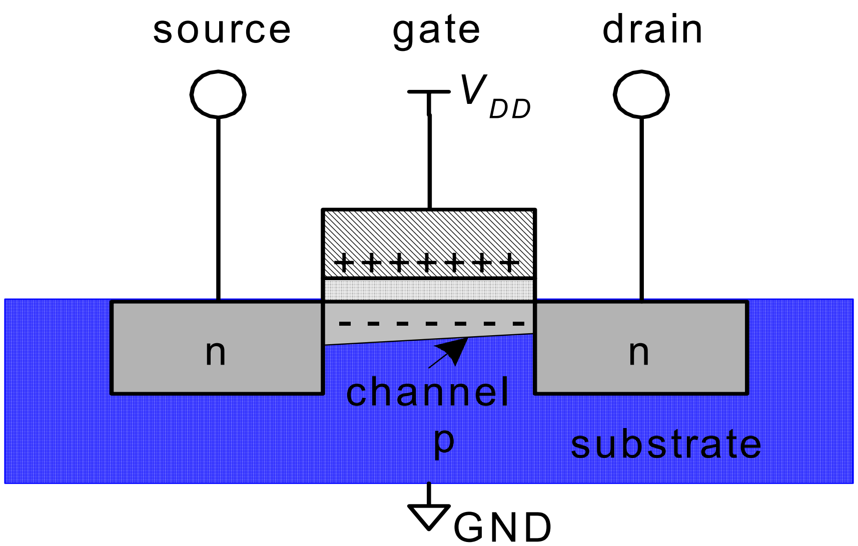
Gate = 0, ausgeschaltet

- keine Verbindung zwischen Source und Drain

Gate = 1, eingeschaltet

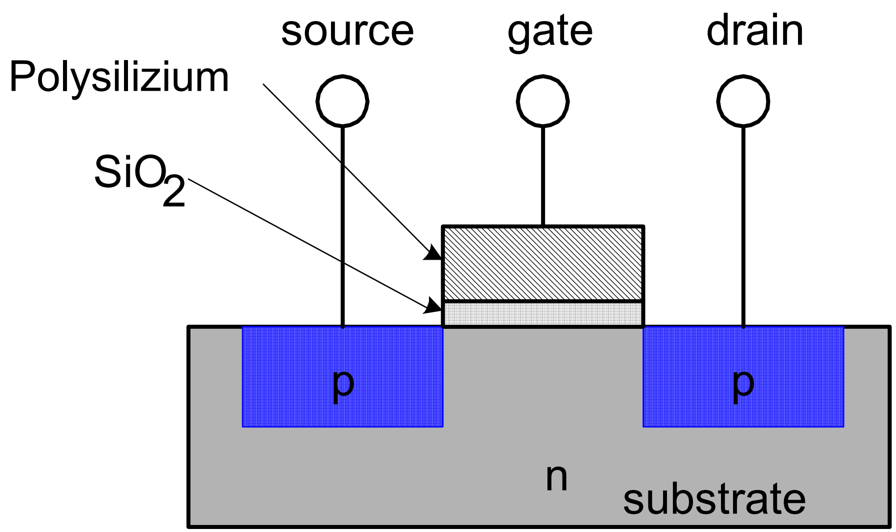
- leitfähiger Kanal zwischen Source und Drain)

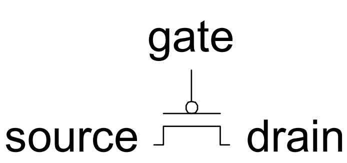




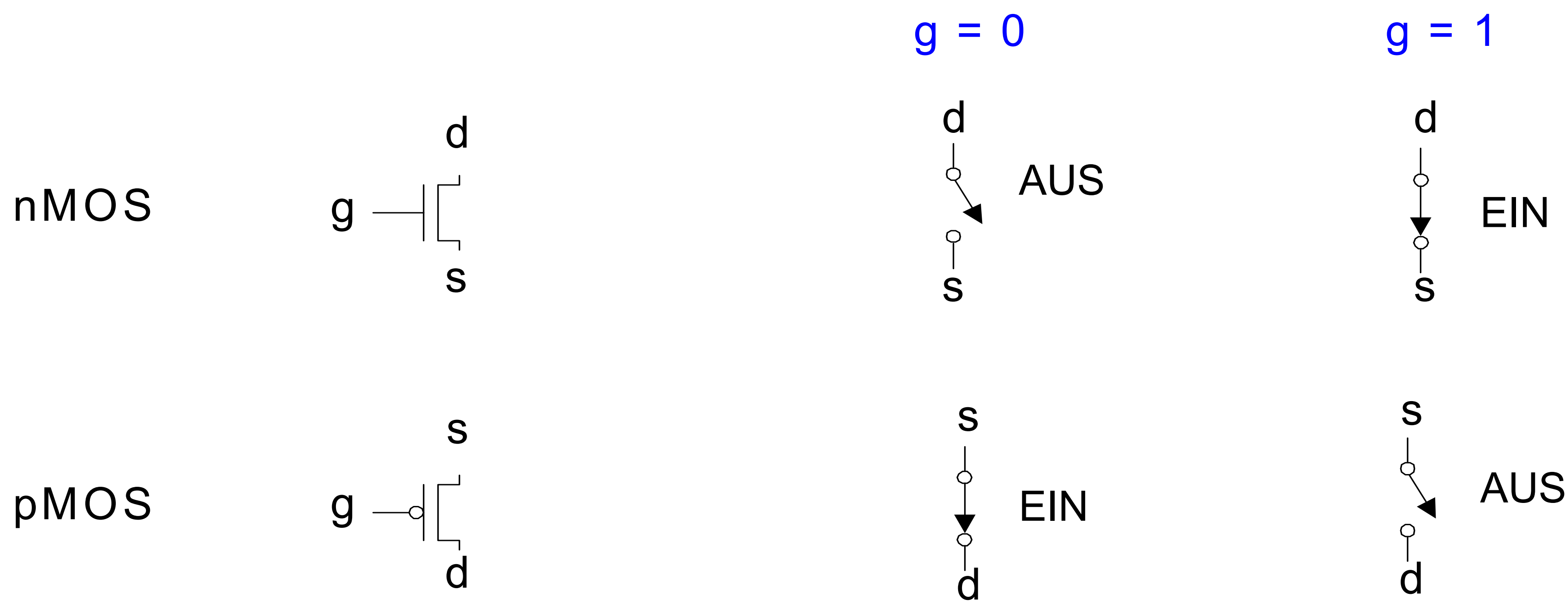
Transistor: pMOS

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



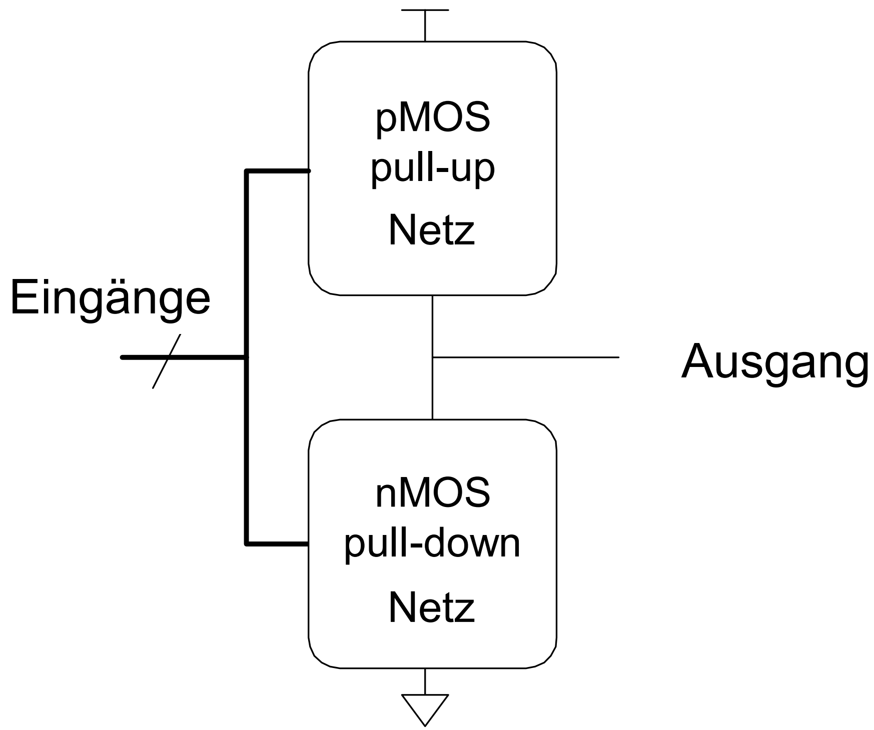


Übersicht über Funktion von Transistoren



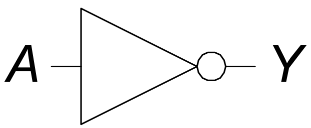
Kombinieren von Transistoren

- 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



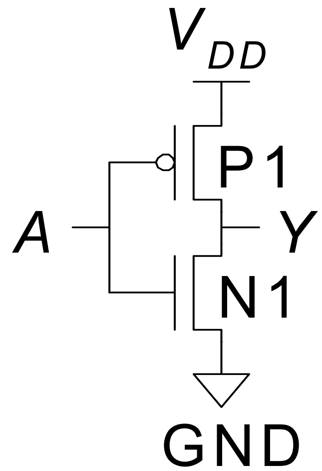
CMOS Gatter: NOT

NOT



$$Y = \overline{A}$$

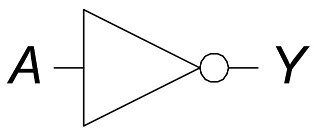
A	Y
0	1
1	0



A	P1	N1	Y
0			
1			

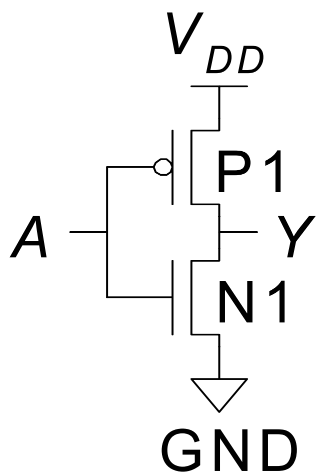
CMOS Gatter: NOT

NOT



$$Y = \overline{A}$$

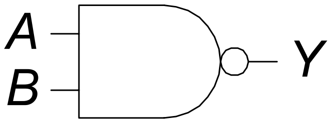
A	Y
0	1
1	0



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

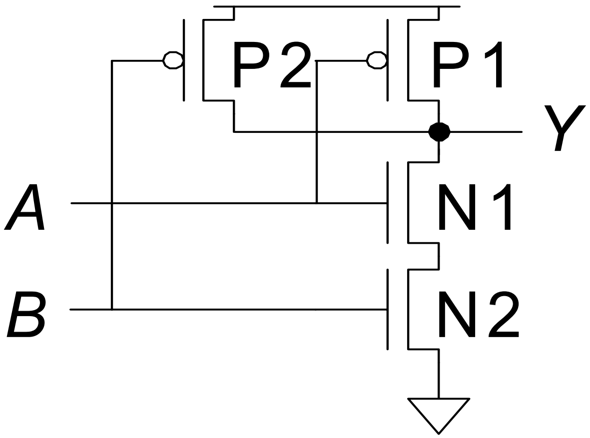
CMOS Gatter: NAND

NAND



$Y = \overline{AB}$

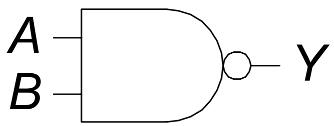
A	*B*	*Y*
0	0	1
0	1	1
1	0	1
1	1	0



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

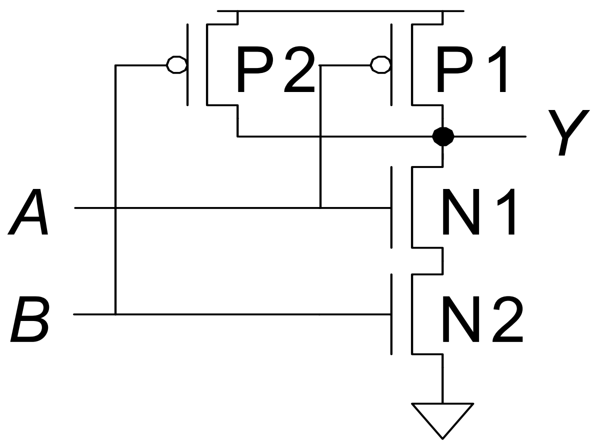
CMOS Gates: NAND Gate

NAND



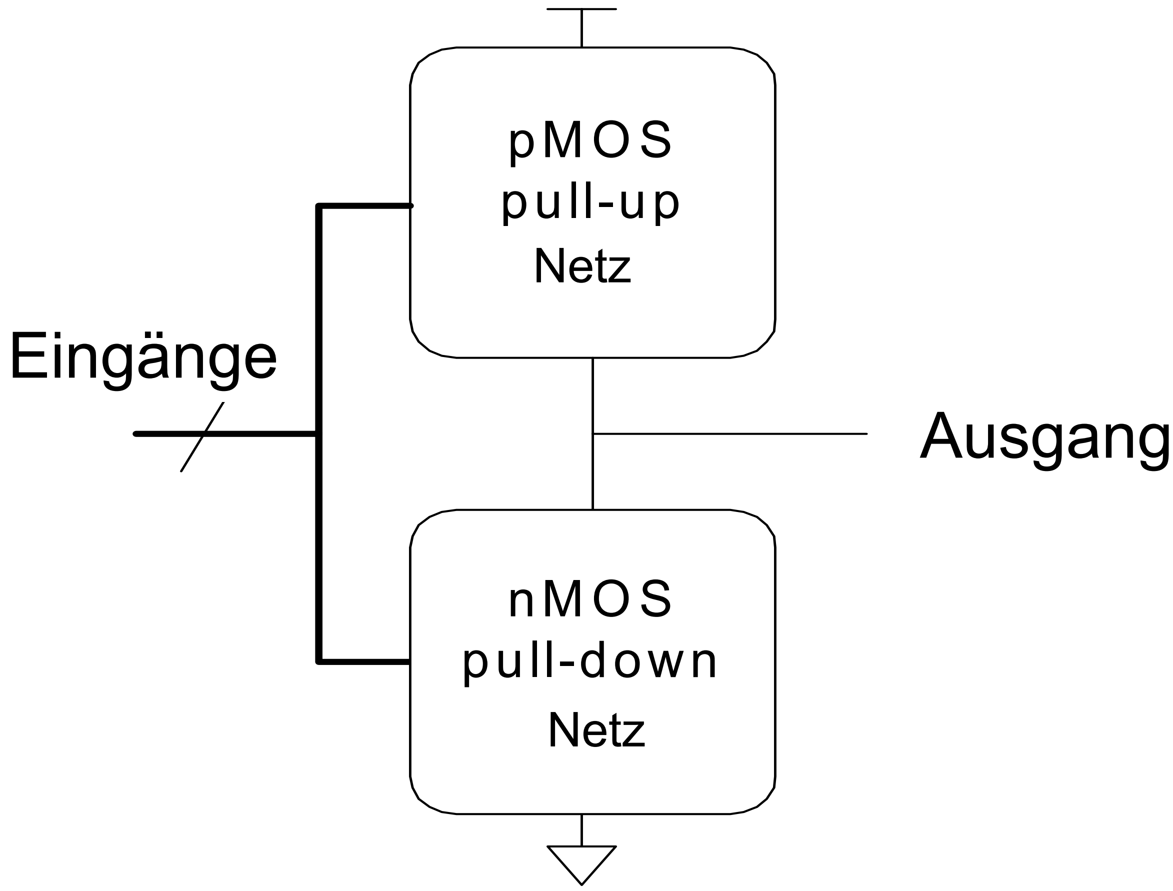
$Y = \overline{AB}$

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	OFF	1
1	1	AUS	AUS	EIN	EIN	0

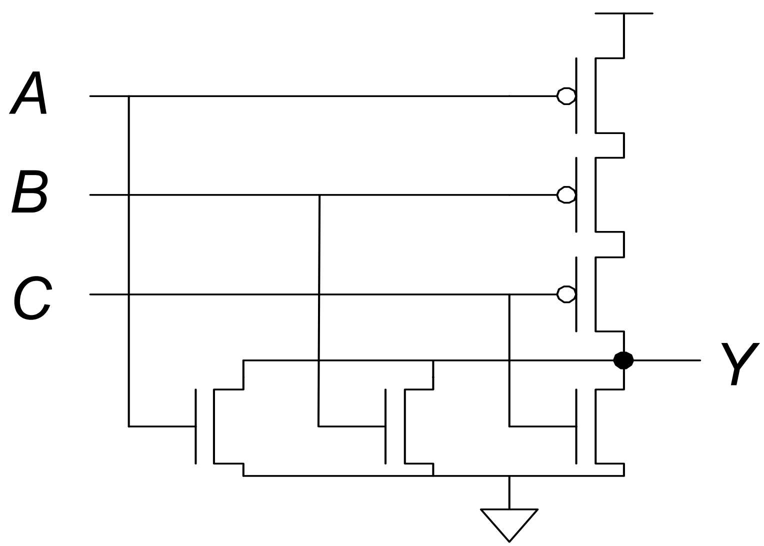
Struktur eines CMOS Gatters



Aufbau eines NOR-Gatters mit drei Eingängen

Aufbau eines NOR-Gatters mit drei Eingängen

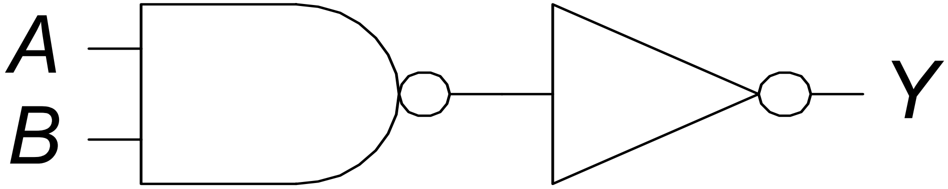
NOR Gatter mit drei Eingängen



Aufbau eines AND-Gatters mit zwei Eingängen

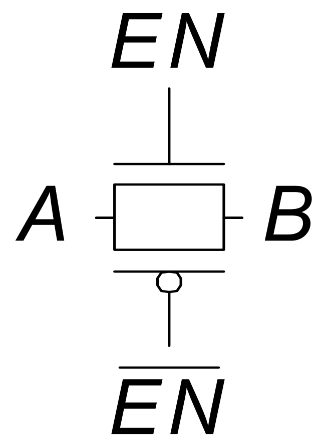
Aufbau eines AND-Gatters mit zwei Eingängen

AND Gatter mit zwei Eingängen



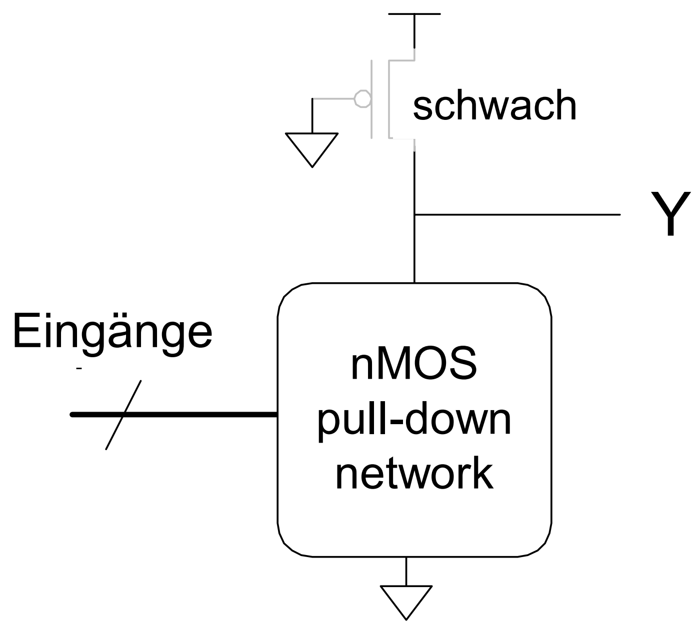
Transmissionsgatter (*transmission gates*)

- 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



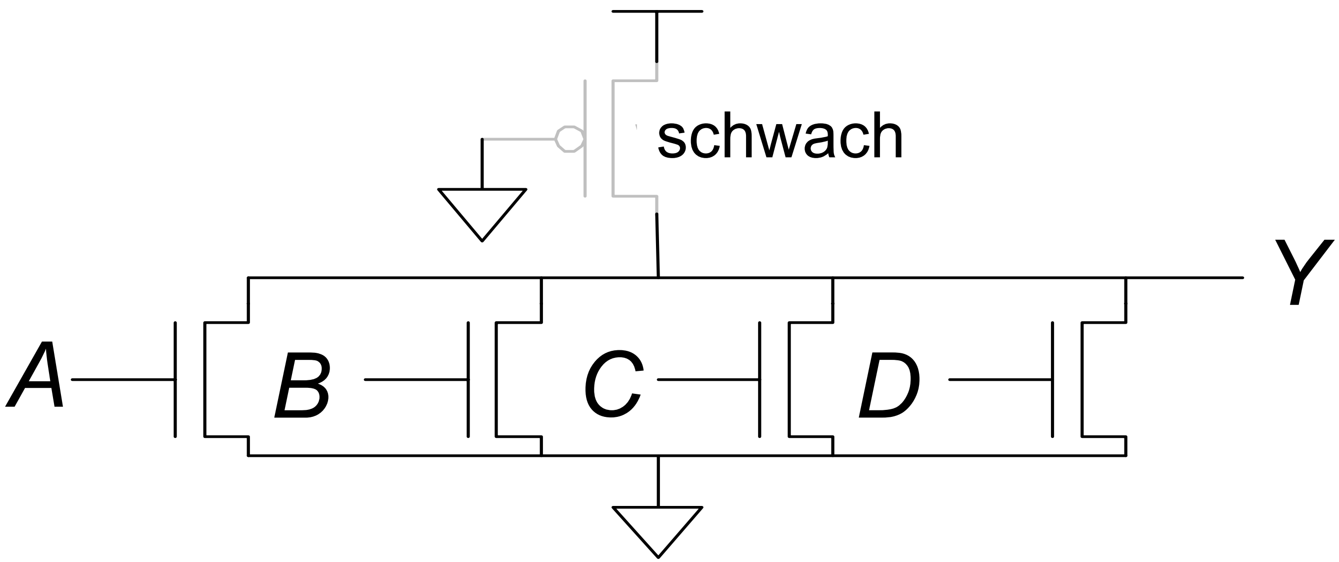
Tricks: Pseudo-nMOS Gatter

- 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

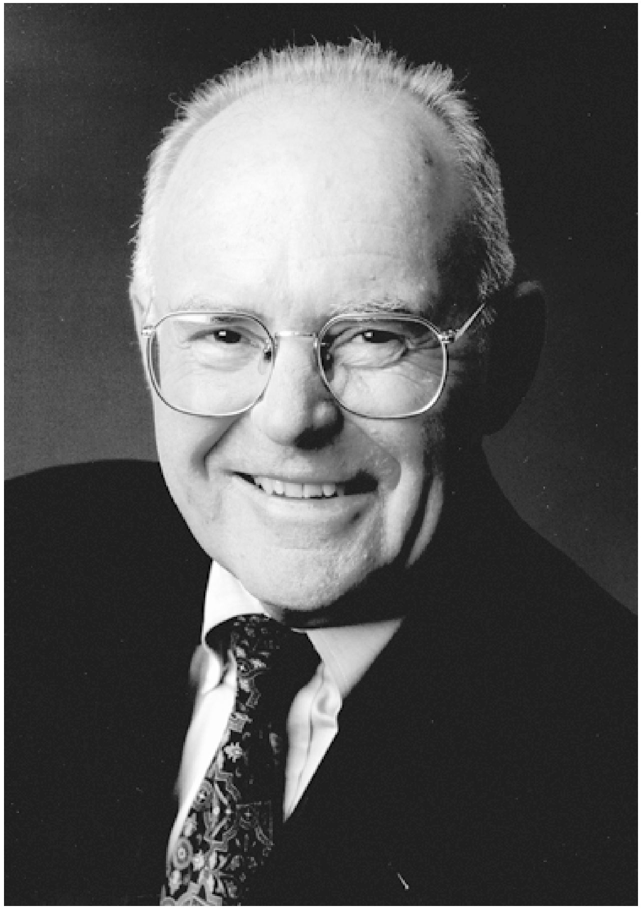


Verbraucht aber mehr Energie: Schwacher Dauerkurzschluss bei Y=0

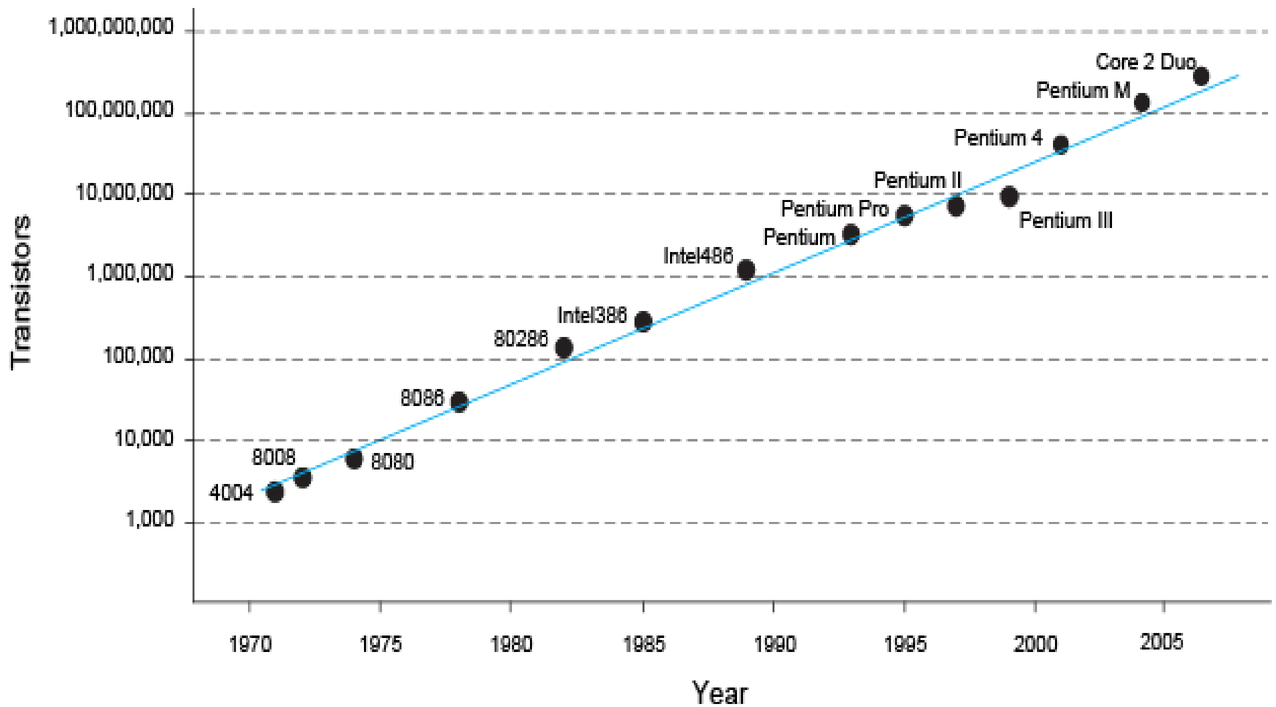
Gordon Moore, 1929 -

- Gründete Intel in 1968 zusammen mit Robert Noyce

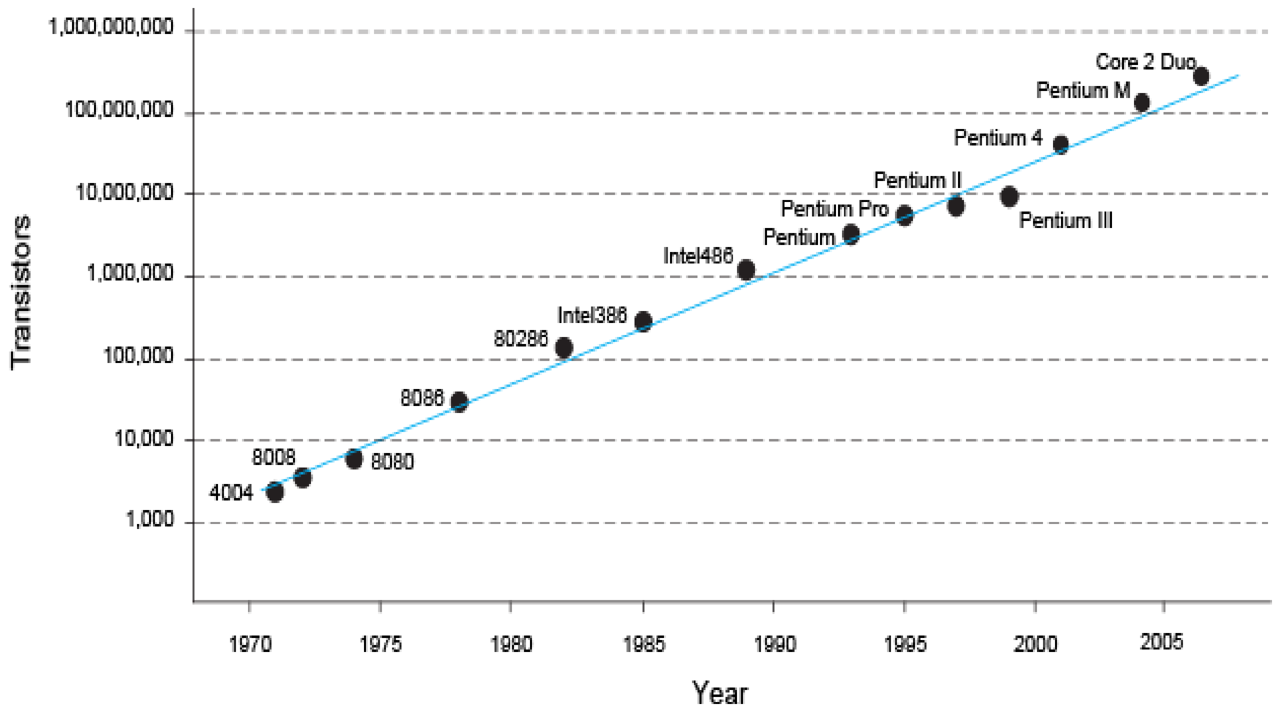
- **Moores Gesetz:** Die Anzahl von Transistoren auf Chips verdoppelt sich
  - Jedes Jahr (1965)
  - Alle zwei Jahre (angepasst 1975)



Moores Gesetz



Moores Gesetz



- *"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
- Energie um einen Kondensator der Kapazität *C* auf V_{DD} zu laden:
  - $E = ½\ Q\ V_{DD} = ½\ (C\ V_{DD})\ V_{DD} = ½\ C\ V_{DD}^2$
  - Ebenso beim Entladen ➔ Gesamtenergie: $C\ V_{DD}^2$
- Schaltung wird mit Frequenz *f* betrieben
  - Transistoren schalten *f*-mal pro Sekunde
  - Aber nicht alle Transistoren schalten jeden Takt um 0-1-0 um
  - Annahme: Jeden Takt nur Laden oder Entladen
    - Halbe Energieaufnahme (realistischer wäre 0,1)
- Die dynamische Leistungsaufnahme ist also:

$$\boxed{P_{dynamic} = ½\ C\ V_{DD}^2\ f}$$

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

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

Beispielrechnung Leistungsaufnahme

- Abschätzen der Leistungsaufnahme für modernen PDA
- Parameter
  - Versorgungsspannung V_{DD} = 1.2 V
  - Transistorkapazität C = 20 nF
  - Taktfrequenz f = 1 GHz
  - Leckstrom I_{DD} = 20 mA

Beispielrechnung Leistungsaufnahme

TECHNISCHE UNIVERSITÄT DARMSTADT

- Abschätzen der Leistungsaufnahme für ein Netbook
- Parameter
  - Versorgungsspannung V_{DD} = 1.2 V
  - Transistorkapazität C = 20 nF
  - Taktfrequenz f = 1 GHz
  - Leckstrom I_{DD} = 20 mA

$$P = \tfrac{1}{2}\, C\, V_{DD}^2 f + I_{DD} V_{DD}$$

$$= \tfrac{1}{2}\,(20\text{ nF})\,(1.2\text{ V})^2\,(1\text{ GHz}) + (20\text{ mA})(1.2\text{ V})$$

$$\mathbf{= 14{,}4\ W}$$

embedded systems + applications