

Technische Informatik



Digitaltechnik

- **Zahlen**
- **Zahlensysteme**
- **Aufbau von Zahlensystemen**
- **Wandlung von Zahlensystemen**
- **Rechnen mit Binärwerten**
- **Einführung und Grundbegriffe**
- **Codes**
- **Logische Verknüpfungen und ihre Darstellung**
- **Schaltalgebra**

- **Schaltungssynthese**
- **Minimierung mit Hilfe der Schaltalgebra**
- **Sequenzielle Schaltungen**
- **Halbleiter**

Literatur

- Elektronik 4: Digitaltechnik, K. Beuth, Vogel Fachbuch
- Digitaltechnik, K. Fricke, Springer Vieweg
- Digitaltechnik, R. Woitowitz, Springer
- Grundlagen der Digitaltechnik, G. W. Wöstenkühler, Hanser



Zahlen

Zahlen

- **Zahlen** sind abstrakte mathematische Objekte beziehungsweise Objekte des Denkens, die sich historisch aus Vorstellungen von Größe und Anzahl entwickeln.
- Durch eine Messung wird ein als Größe verstandener Aspekt einer Beobachtung mit einer Zahl in Verbindung gebracht, beispielsweise bei einer Zählung.
- Sie spielen daher für die empirischen Wissenschaften eine zentrale Rolle

- Eine Zahl ist ein Objekt, das man mit einem anderen Objekt der gleichen Art verknüpfen kann (z.B. addieren), so dass man wieder ein Objekt der gleichen Art bekommt.
- Um bestimmte Zahlen zu beschreiben, muss man daher zweierlei festlegen:
 - Welche Objekte gehören zu meiner Zahlenmenge?
 - Wie werden sie verknüpft?
- Zahlen haben üblicherweise auch noch einen physikalischen Bezug, d.h. man kann mit ihnen Objekte der realen Welt zählen oder messen.

- Verschiedene Zahlenmengen sind erweitert worden, indem man andere Zahlen *dazu nimmt*.
- Die Motivation kann sein, dass die ursprüngliche Zahlenmenge mit (sagen wir) der Addition abgeschlossen ist, aber nicht mehr mit ihrer Umkehrung der Subtraktion.
- Nehmen wir das Beispiel der natürlichen Zahlen als Ausgangsmenge, also die Zahlen:

0, 1, 2, 3, ...

- Wenn man zwei natürliche Zahlen addiert, bekommt man wieder eine natürliche, das nennt man Abgeschlossenheit.
- Jetzt definiert man aber auch das Gegenteil der Addition, die Subtraktion, und stellt fest, dass $x-y$ nur eine natürliche Zahl ist, wenn y nicht größer als x ist.
- Für diesen letzteren Fall kann man die natürlichen Zahlen um negative Zahlen erweitern und kommt zu den ganzen Zahlen.

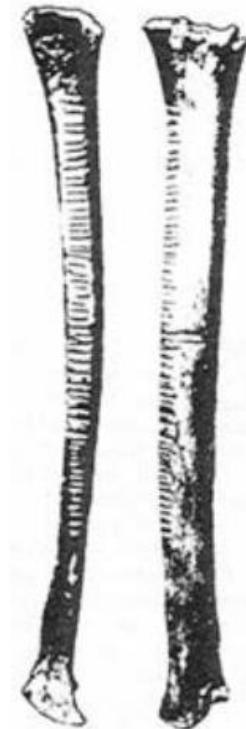
Der Unterschied zwischen Ziffer und Zahl

- Ziffern und Zahlen sind unentbehrliche Begleiter.
- Im Alltag werden sie häufig irrtümlicherweise synonymisch verwendet.
- Zwar drücken Ziffer und Zahl eine bestimmte Menge aus, identisch sind sie aber nicht.
- Ziffern sind einstellig: 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9. Zahlen können entweder einstellig sein, also nur aus einer Ziffer bestehen, oder aus zwei oder mehreren Ziffern zusammengeführt eine Zahl ergeben.
- Ziffern sind folglich Bestandteil von Zahlen.

- Für die Bildung von Zahlen kann man alle Ziffern verwenden und in beliebiger Reihenfolge (Positionsordnung) anordnen.
- Doch Ziffer ist nicht gleich Ziffer.
- So existieren beispielsweise Einerziffern, Zehnerziffern, Hunderterziffern, Tausenderziffern etc.
- Werden zwei Ziffern zu einer Zahl zusammengesetzt, erhält die erste Ziffer die Wertigkeit von Zehn.
- Die zweite Ziffer erhält die Wertigkeit von Eins.
- Die zweistellige Zahl 52 besteht aus der Zehnerziffer Fünf und der Einerziffer Zwei.
- Die dreistellige Zahl 427 besteht aus der Hunderterziffer Vier, der Zehnerziffer Zwei und der Einerziffer Sieben.

Geschichtliches zu Zahl und Zahldarstellung

- Erste menschliche Zahldarstellungen (Einkerbungen in Tierknochen, Hölzern u.ä.) sind aus der Steinzeit bekannt.
- Ihr Alter wird auf etwa 20.000–30.000 Jahre geschätzt (vgl. Wußing 2008).
- Einer der ältesten Funde ist ein im Jahre 1937 in Dolni Vésto-nice (Unter-Wisternitz), in der damaligen Tschechoslowakei (Mähren), gefundener Wolfsknochen mit 55 Einkerbungen in zwei Serien mit jeweils 25 und 30 Kerben.
- Sein Alter wird auf 12.000 bis 20.000 Jahre geschätzt.



Wolfsknochen mit
Einkerbungen



Zahlensysteme

Was ist ein Zahlensystem?

- Ein Zahlensystem ist eine Methode, um Zahlen darzustellen und mit ihnen zu arbeiten.
- Zahlensysteme basieren auf einer Basis (oder Radix), die bestimmt, wie viele Ziffern im System verwendet werden.
- Die gebräuchlichsten Zahlensysteme sind das Dezimalsystem (Basis 10), das Binärsystem (Basis 2) und das Hexadezimalsystem (Basis 16).

Vor allem werden drei Ziffernsysteme unterschieden:

➤ **Stellenwertsystem oder Positionssystem:**

(Beispiel: das Dezimalsystem) je nach der Position, an welcher sie steht. Wenn in diesem Zahlensystem eine Ziffer an

- Das Stellenwertsystem hat sich besonders bewährt und fast weltweit ausgebreitet.
- Der Wert einer Ziffer wird unterschiedlich gewichtet einer Stelle steht wie die 2 in der Zahl 20, wird ihr Wert mit dem Faktor zehn gewichtet (multipliziert), wenn sie an einer Stelle steht wie die 2 in der Zahl 200, dann mit dem Faktor hundert.
- Eine solche Zahl hat den Wert der Summe der gewichteten Ziffernwerte.

Zahlensysteme

➤ Additionssystem:

(Beispiel: die Römische Zahlschrift)

- Bei einem Additionssystem gibt es keine Gewichtung und kein Zeichen für die Null. Sechzehn Striche in einer Strichliste stehen für den Wert sechzehn, ebenso die drei Ziffern in der römischen Zahl XVI.

➤ „Hybridsystem“:

(Beispiel: die Japanische Zahlschrift)

- Ein Hybridsystem enthält neben Ziffern auch Gewichtungsfaktoren ähnlich dem schriftlichen Deutsch.
- Anstatt 3 000 000 bzw. 3'000'000 schreibt man 3 Millionen, in Naturwissenschaft und Technik auch $3e^6$.
- Im traditionellen japanischen Ziffernsystem schreibt man 三万六十 für dreimal zehntausend plus sechsmal zehn für 30 060 bzw. 30'060;
- die Gewichtungsfaktoren sind hier grün gekennzeichnet.

Praktische Anwendungen von Zahlensystemen

- **Dezimalsystem:**
 - Das Dezimalsystem wird im täglichen Leben und in der Mathematik verwendet.
 - Es ist das grundlegende System, das in der Schule gelehrt wird und das wir bei finanziellen Transaktionen, der Messung von Entfernungen und der Zeitmessung verwenden.
- **Binärsystem:**
 - Das Binärsystem ist die Grundlage für Computertechnologie.
 - Computer verwenden das Binärsystem, um Informationen zu speichern und zu verarbeiten, da es leichter ist, zwei Zustände (0 und 1) elektronisch darzustellen.

➤ Hexadezimalsystem:

- Das Hexadezimalsystem wird hauptsächlich in der Computertechnik verwendet, um binäre Daten kompakter und leichter lesbar darzustellen.
- Zum Beispiel wird es häufig verwendet, um Farben im Webdesign und Speicheradressen in der Programmierung darzustellen.
- Im Gegensatz zum dezimalen System, das auf 10 Zahlen basiert, nutzt das hexadezimale System 16 Zahlen: 0-9 und A-F.
- In vielen Programmiersprachen wird auch das Präfix „0x“ verwendet, um anzugeben, dass eine Zahl im hexadezimalen Format vorliegt

➤ Oktales Zahlensystem

- Es handelt sich dabei um ein Zahlensystem, das auf der Basis 8 aufgebaut ist.
- Im Gegensatz zum dezimalen System, das auf der Basis 10 beruht, hat das oktale System nur 8 Ziffern – von 0 bis 7.
- Es ist einfacher zu handhaben als das hexadezimale System (Basis 16), da es weniger Ziffern gibt.
- Zum anderen wird es in manchen Bereichen der Informatik noch immer genutzt – insbesondere bei der Arbeit mit älteren Betriebssystemen oder Hardware-Architekturen.

BCD-Zahlensystem

- Eine weitere interessante Art von Zahlensystemen, die für Programmierer von Bedeutung sein kann, ist das BCD-Zahlensystem.
- BCD steht für Binary Coded Decimal und bedeutet übersetzt binär codierte Dezimalzahlen.
- Anders als bei anderen Zahlensystemen wie dem Binär- oder Hexadezimalsystem, wird hierbei jede Ziffer einer Dezimalzahl in einem eigenen 4-Bit-Code dargestellt.
- Dadurch wird eine höhere Genauigkeit erreicht und es können präzise Rechnungen durchgeführt werden.

- Das BCD-Zahlensystem findet Anwendung in der Elektronik und insbesondere in der Programmierung von Mikrocontrollern und Mikroprozessoren, da diese oft mit dezimalen Eingaben arbeiten müssen.
- Ein Verständnis für das BCD-Zahlensystem kann daher bei der Entwicklung von Programmen für solche Geräte hilfreich sein.
- Hier sind 5 Beispiele für Zahlen im BCD-Zahlensystem:
 1. Dezimalzahl 0 entspricht im BCD der Binärzahl 0000
 2. Dezimalzahl 1 entspricht im BCD der Binärzahl 0001
 3. Dezimalzahl 2 entspricht im BCD der Binärzahl 0010
 4. Dezimalzahl 5 entspricht im BCD der Binärzahl 0101
 5. Dezimalzahl 9 entspricht im BCD der Binärzahl 1001

Gray-Code-Zahlensystem

- Das Gray-Code-Zahlensystem ist eine besondere Art der Binärdarstellung von Zahlen, bei der sich benachbarte Zahlen nur in einer einzigen Bitposition unterscheiden.
- Dadurch ist es besonders nützlich, wenn man sequentielle Schaltungen entwirft, da dadurch fehlerhafte Signale vermieden werden können.
- Auch in der Übertragung von Daten kann der Gray-Code Vorteile bieten, da er weniger anfällig für Störungen ist und somit eine höhere Fehlerkorrekturrate ermöglicht.

- Hier sind fünf Beispiele für Zahlen im Gray-Code-Zahlensystem:

1. Dezimalzahl 0 entspricht im Gray-Code der Binärzahl 0 0000
2. Dezimalzahl 1 entspricht im Gray-Code der Binärzahl 1 0001
3. Dezimalzahl 2 entspricht im Gray-Code der Binärzahl 3 0011
4. Dezimalzahl 3 entspricht im Gray-Code der Binärzahl 2 0010
5. Dezimalzahl 4 entspricht im Gray-Code der Binärzahl 6 0110

Zahlensysteme und der Computer

- Wenn Sie dem Computer irgendwas direkt sagen wollen, dann müssen Sie sich in Zahlen ausdrücken.
- Das nächste Problem ist, dass der Computer zwar grundsätzlich Zahlen versteht, es aber vorzieht, in anderen Zahlensystemen zu arbeiten als Sie.
- Während Sie im Alltag mit dem Dezimalsystem gut zuretkommen, verwendet Ihr Computer am liebsten das Binärsystem.
- Und damit es nicht zu einfach wird, lassen sich Zahlen im Binärsystem leider nicht allzu einfach in Zahlen des Dezimalsystems umwandeln.

- Nachdem das Binärsystem für menschliche Belange äußerst unangenehm zu handhaben ist, verwenden Programmierer lieber das Hexadezimalsystem.
- Denn vom Binärsystem ins Hexadezimalsystem sind es nur ein paar Schritte, und die Handhabung von Hexadezimalzahlen fällt doch wesentlich einfacher als die von Binärzahlen.
- Beachten Sie, dass moderne Programmiersprachen normalerweise ohne Probleme die Angabe von Dezimalzahlen ermöglichen, wenn an irgendeiner Stelle im Programm Zahlen angegeben werden müssen.

Warum werden in der Informatik unterschiedliche Zahlensysteme verwendet?

- Bei der Darstellung von großen Zahlen im binären wird die Zahl jedoch **sehr** lange.
- Man braucht mehrere Zeilen, um sie auszuschreiben.
- Deshalb greift man auf das Oktalsystem mit der Basis **8** oder das Hexadezimalsystem mit der Basis **16** zurück.
- Für das Hexadezimalsystem benötigt man aber mehr als zehn Ziffern.



Aufbau von Zahlensystemen

Aufbau von Zahlensystemen

- Jedes Zahlensystem besteht aus Nennwerten.
- Die Anzahl der Nennwerte ergibt sich aus der Basis.
- Der größte Nennwert entspricht der Basis minus (-) 1.
- Wird der größte Nennwert überschritten, entsteht aus dem Übertrag der nächst höhere Stellenwert.

Basis-Zahlendarstellung

$$zahl = \sum_{i=0}^n a_i \cdot b^i$$

- die Basis $b \geq 2$ ist aus den natürlichen Zahlen
- die Ziffer a_i ist aus den natürlichen Zahlen $0 \leq a_i \leq b-1$
- die Darstellung ist eindeutig
- Schreibweise:

$$zahl = (a_n \dots a_0)_b$$

Dezimalsystem

- Das von uns im Alltag verwendete Dezimalsystem kennt 10 Zahlensymbole:
 $\{0,1,2,3,4,5,6,7,8,9\}$
- Dabei ist die *absolute* Position der Zeichen wichtig, um den Zahlenwert zu ermitteln
Beispiel: $123 \neq 321$
Beispiel: $5124 = 5 \cdot 1000 + 1 \cdot 100 + 2 \cdot 10 + 4$

- Der Zahlenwert w berechnet sich also wie folgt aus den Ziffern z_{n-1}, \dots, z_1, z_0 einer Dezimalzahl:

$$\begin{aligned} w &= z_{n-1} \cdot \underbrace{10 \dots 0}_{n-1} + \dots + z_2 \cdot 100 + z_1 \cdot 10 + z_0 \cdot 1 = \\ &= \sum_{i=0}^{n-1} z_i \cdot 10^i \end{aligned}$$

- Bei dem Dezimalsystem handelt es sich um ein so genanntes *Stellenwertsystem*
Das Dezimalsystem ist ein Stellenwertsystem mit der Basis 10

Stellenwertsysteme: b-adische Darstellung

Sei $b > 1$ eine beliebige natürliche Zahl ($b \in \mathbb{N}$), dann heisst die Menge $\{0, \dots, b-1\}$ das Alphabet des b-adischen Zahlensystems (mit der Basis b)

- Dezimalsystem:
 $b=10 \rightarrow \{0,1,2,3,4,5,6,7,8,9\}$
- Dualsystem (Binärsystem):
 $b=2 \rightarrow \{0,1\}$
- Oktalsystem:
 $b=8 \rightarrow \{0,1,2,3,4,5,6,7\}$
- Hexadezimalsystem:
 $b=16 \rightarrow \{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f\}$

Stellenwertsysteme: b-adische Darstellung

- Wie schon festgestellt, berechnet sich ein Zahlenwert w im Dezimalsystem gemäß:

$$w = \sum_{i=0}^{n-1} z_i \cdot 10^i$$

- Dies lässt sich verallgemeinern. Für beliebige Basen $b > 1$, gilt:

$$w = \sum_{i=0}^{n-1} z_i \cdot b^i$$

die Anzahl der Stellen der Zahl w angibt

- Ziffernschreibweise:

$$w = (z_{n-1} \dots z_2 z_1 z_0)_b$$

Dualsystem (Binärsystem)

Der Basis $b=2$ mit einem Alphabet von zwei Zahlensymbolen, 0 und 1, kommt in der Informatik eine besondere Bedeutung zu, da sich zwei Symbole leicht elektrisch kodieren lassen:

- "kein Strom fließt" entspricht typischerweise der 0
- "Strom fließt" der 1
- Jede Stelle einer Binärzahl wird als "Bit" ("Binary Digit") bezeichnet
- 1 Bit ist demnach die kleinste Informationsmenge, die gespeichert werden kann
- 8 Bits werden als 1 Byte bezeichnet

Zahlensysteme

Die Bitfolge $(01001101)_2$ entspricht der Dezimalzahl $(77)_{10}$

$$\begin{aligned}(01001101)_2 &= 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 0 \cdot 128 + 1 \cdot 64 + 0 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= 64 + 8 + 4 + 1 \\ &= 77\end{aligned}$$

Oktal- und Hexadezimalsystem

- **Oktalsystem:**

Die Oktalzahl $(115)_8$ entspricht der Dezimalzahl $(77)_{10}$

$$(115)_8 = 1 \cdot 8^2 + 1 \cdot 8^1 + 5 \cdot 8^0 = 1 \cdot 64 + 1 \cdot 8 + 5 \cdot 1 = 77$$

- **Hexadezimalsystem:**

Die Zahl $(4D)_{16}$ entspricht der Dezimalzahl $(77)_{10}$

$$(4D)_{16} = 4 \cdot 16^1 + D \cdot 16^0 = 64 + 13 = 77$$

Dezimal-Präfixe

10^1	10^2	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}	10^{21}	10^{24}
Deka	Hekto	Kilo	Mega	Giga	Tera	Peta	Exa	Zetta	Yotta
da	h	k	M	G	T	P	E	Z	Y

10^{-1}	10^{-2}	10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}	10^{-21}	10^{-24}
Dezi	Zenti	Milli	Mikro	Nano	Piko	Femto	Atto	Zepto	Yokto
d	c	m	μ	n	p	f	a	z	y

Binär-Präfixe

Wenn in der Informatik von 1 kByte (Kilobyte) gesprochen wird, bezeichnet dies leider nicht immer 1000 Bytes, sondern häufig 1024 Bytes. Dies führt zu Verwirrungen.

- Daher sollten für die Potenzen von 1024 besser die folgenden Präfixe laut IEC verwendet werden:

Name	Präfix	Wert
kibi	Ki	$2^{10} = 1024^1 = 1.024$
mebi	Mi	$2^{20} = 1024^2 = 1.048.576$
gibi	Gi	$2^{30} = 1024^3 = 1.073.741.824$
tebi	Ti	$2^{40} = 1024^4 = 1.099.511.627.776$
pebi	Pi	$2^{50} = 1024^5 = 1.125.899.906.842.624$
exbi	Ei	$2^{60} = 1024^6 = 1.152.921.504.606.846.976$
zebi	Zi	$2^{70} = 1024^7 = 1.180.591.620.717.411.303.424$
yobi	Yi	$2^{80} = 1024^8 = 1.208.925.819.614.629.174.706.176$

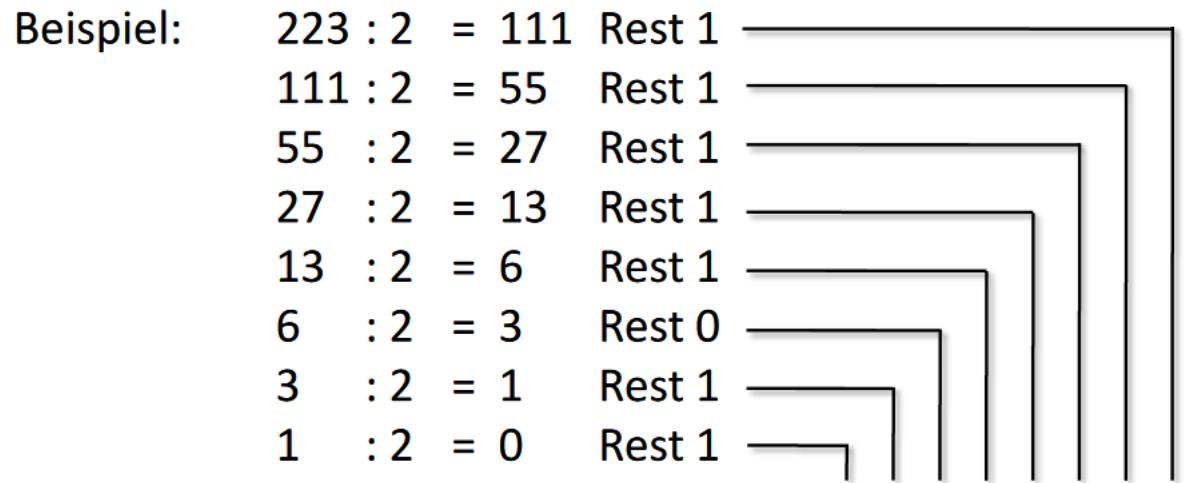


Wandlung von Zahlensystemen

Umrechnung Dezimal → Binär

Methode: Fortgesetzte Division durch zwei

Die umzuwandelnde Dezimalzahl wird fortlaufend durch zwei dividiert, bis null erreicht wird. Die dabei auftretenden Divisionsreste – in umgekehrter Reihenfolge ergeben die gesuchte Binärzahl



Die entsprechende Binärzahl lautet: 1 1 0 1 1 1 1

Umrechnung Dezimal → Binär

Als nächstes konvertieren wir den Dezimalbruch 0,6875 in einen binären Bruch durch sukzessive Multiplikation.

$$\begin{array}{rcl} 0,6875 * 2 = 1,375 & = 0,375 \text{ A1} \\ 0,375 * 2 = 0,75 & = 0,75 \text{ A0} \\ 0,75 * 2 = 1,50 & = 0,5 \text{ A1} \\ 0,5 * 2 = 1,00 & = 0,0 \text{ A1} \end{array}$$



Das binäre Äquivalent von $0,6875_{10}$ ist also: $0,1011_2$

Umrechnung Binär → Dezimal

Um eine Zahl vom **Binärsystem** ins Dezimalsystem umzurechnen, multipliziert man die Ziffer mit dem entsprechenden **Stellenwert** und addiert die Produkte.

$$32 + 0 + 8 + 0 + 2 + 1 = 43_d$$

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \ 1 \ 1_b \\ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0 \end{array}$$

Umrechnung Binär → Dezimal

$$0,5 + 0 + 0,125 + 0,0625 = 0,6875_d$$
$$\begin{array}{r} 0, \quad 1 \quad 0 \quad 1 \\ \hline 2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \end{array} \quad 1_b$$

Umrechnung Dezimal → Hexadezimal

Methode: Fortgesetzte Division durch 16

Die Dezimalzahl wird fortlaufend durch 16 dividiert, bis null erreicht wird. Die dabei auftretenden Divisionsreste – in umgekehrter Reihenfolge ergeben die gesuchte Hexadezimalzahl

Beispiel:

$$\begin{array}{r} 443 : 16 = 27 \text{ Rest } 11 \\ 27 : 16 = 1 \text{ Rest } 11 \\ 1 : 16 = 0 \text{ Rest } 1 \end{array}$$

Die entsprechende Hexadezimalzahl lautet: 1 B B

Hilfstabelle: Ziffern A...F im Hexadezimalsystem

A	B	C	D	E	F
10	11	12	13	14	15

Umrechnung Dezimal → Hexadezimal

Hilfstabelle: Vielfache von 16

n	$n \cdot 16$
1	16
2	32
3	48
4	64
5	80
6	96
7	112
8	128

n	$n \cdot 16$
9	144
10	160
11	176
12	192
13	208
14	224
15	240
16	256

Umrechnung Hexadezimal → Dezimal

E 7 A_h

16² 16¹ 16⁰

$$10 * 1 = \quad 10$$

$$7 * 16 = \quad + \quad 112$$

$$14 * 256 = \quad + \quad 3.584$$

$$3.706_d$$

Umrechnung Binär → Hexadezimal

Methode: 4er-Gruppen von Binärzahlen bilden

Die umzuandelnde Binärzahl wird von rechts nach links in 4er-Bündel von Binärziffern gruppiert.

Anschließend werden die Bündel in die entsprechenden Hexadezimalziffern umgewandelt.

Beispiel:

1010	1111	1111	1110
			
A	F	F	E

Hilfstabelle: 4er-Bündel von Binärziffern und Hexadezimalziffern

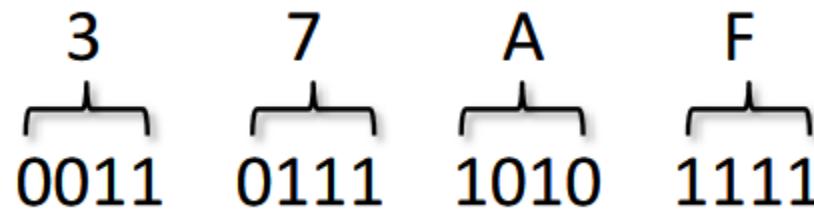
0000	0001	0010	0011	0100	0101	0110	0111
0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111
8	9	A	B	C	D	E	F

Umrechnung Hexadezimal → Binär

Methode: Hexadezimal in 4er-Bündel von Binärziffern „auflösen“

Die umzuwandelnde Hexadezimalzahl wird Ziffer für Ziffer in 4er - Bündel von Binärziffern „aufgelöst“.

Beispiel:



Führende Nullen zu Beginn der Binärzahl können weggelassen werden:

$$37AF_{16} = 11\ 0111\ 1010\ 1111_2$$



Rechnen mit Binärwerten

Addition von Binärzahlen

Es gibt folgende **vier Möglichkeiten** bei der Addition der zwei Ziffern 0 und 1:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10 \leftarrow \text{mit Übertrag}$$

Sollen wir folgende Addition durchführen: $1001 + 100$, so können wir wie gewohnt die schriftliche Addition verwenden:

$$\begin{array}{r} 1001 \\ + 100 \\ \hline = 1101 \end{array}$$

Subtraktion von Binärzahlen

Bei der Subtraktion von Binärzahlen gibt es sechs Möglichkeiten:

Rechenregeln für die Subtraktion von dualen Zahlen

$$0 - 0 = 0$$

$$0 - 1 = 1 + 1 \text{ Entlehnung}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 - 1 = 0 + 1 \text{ Entlehnung}$$

$$1 - 1 - 1 = 1 + 1 \text{ Entlehnung}$$

Subtraktion von Binärzahlen

Beispiel

$$\begin{array}{r} 1111000 \\ - 1001110 \\ \hline \end{array}$$

$$0101010 = 32 + 8 + 2 = 42$$

Wobei der Fall $0 - 1 = -1$ einen Übertrag auslöst,
sodass wir $0 - 1 = -1$ und einen Übertrag haben.

Schauen wir uns dies an einem Beispiel an:

Wir wollen $1100 - 1001$ errechnen.

$$\begin{array}{r} 1100 \\ - 1001 \\ \hline \textcolor{blue}{Ü: } 0110 \\ = 0011 \end{array}$$

Multiplikation von Binärzahlen

Multiplikation von Binärzahlen

Für die Multiplikation von Binärzahlen gilt:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

Im weiteren gehen wir genau so vor, wie wir es vom Dezimalsystem (schriftliche Multiplikation) kennen. Machen wir dies mit dem Beispiel $1111 \cdot 1001$. 0

$$\begin{array}{r} 1101 \cdot 1001 \\ \hline 1101 \\ + \quad 0000 \\ + \quad 0000 \\ + \quad 1101 \\ \hline \text{Übertrag} \quad 0010000 \\ \text{Produkt} \quad 1110101 \end{array}$$

Division von Binärzahlen

Die Division durch 0 ist wie im Dezimalsystem nicht definiert. Somit bleiben nur zwei Möglichkeiten für die Division von Binärzahlen:

$$\begin{aligned}0 : 1 &= 0 \\1 : 1 &= 1\end{aligned}$$

Schauen wir uns die Divison von Binärzahlen an und wählen dazu das Beispiel:

$$1000010 : 11 =$$

Wir gehen genau so vor,
wie wir es bei der schriftlichen
Division gelernt hab

$$\text{Lösung: } 1000010_2 : 11_2 = 10110_2$$

$$\begin{array}{r} 1000010 \\ \underline{- 0} \\ 100 \\ \underline{- 11} \\ 10 \\ \underline{- 0} \\ 100 \\ \underline{- 11} \\ 11 \\ \underline{- 11} \\ 00 \\ \underline{- 0} \\ 0 \end{array} \quad \begin{array}{l} 0 \\ 10110 \end{array}$$

Subtraktion von Dualzahlen mittels Zweierkomplementbildung

Für die Subtraktion von Dualzahlen gibt es in der Digitaltechnik keine logische Verknüpfung. Deshalb behilft man sich mit der Zweierkomplementbildung. Dabei werden die dualen Zahlen addiert, wobei das Ergebnis einer Subtraktion entspricht.

$$2 - 6 = -4 \quad 2 + (-6) = -4$$

Die Subtraktion ist eine Form der Addition, bei der ein Summand ein negatives Vorzeichen hat. Das negative Vorzeichen löst dabei die Addition auf.

Die Addition wird zur Subtraktion.

Das Ganze lässt sich auch wieder umkehren.

Bei der Subtraktion von Dualzahlen macht man sich das zu Nutze, indem man zuerst den Zweierkomplement der negativen Zahl bestimmt.

1. Zuerst muss der Zweierkomplement für die negative Zahl gebildet werden.
2. Anschließend kann eine Addition mit der negativen Zahl durchgeführt werden.

Beispiel:

$$10 - 110 = ?$$

Duale Zahlen subtrahieren $2 - 6 = -4$
 $2 + (-6) = -4$

$$0010 - 0110$$

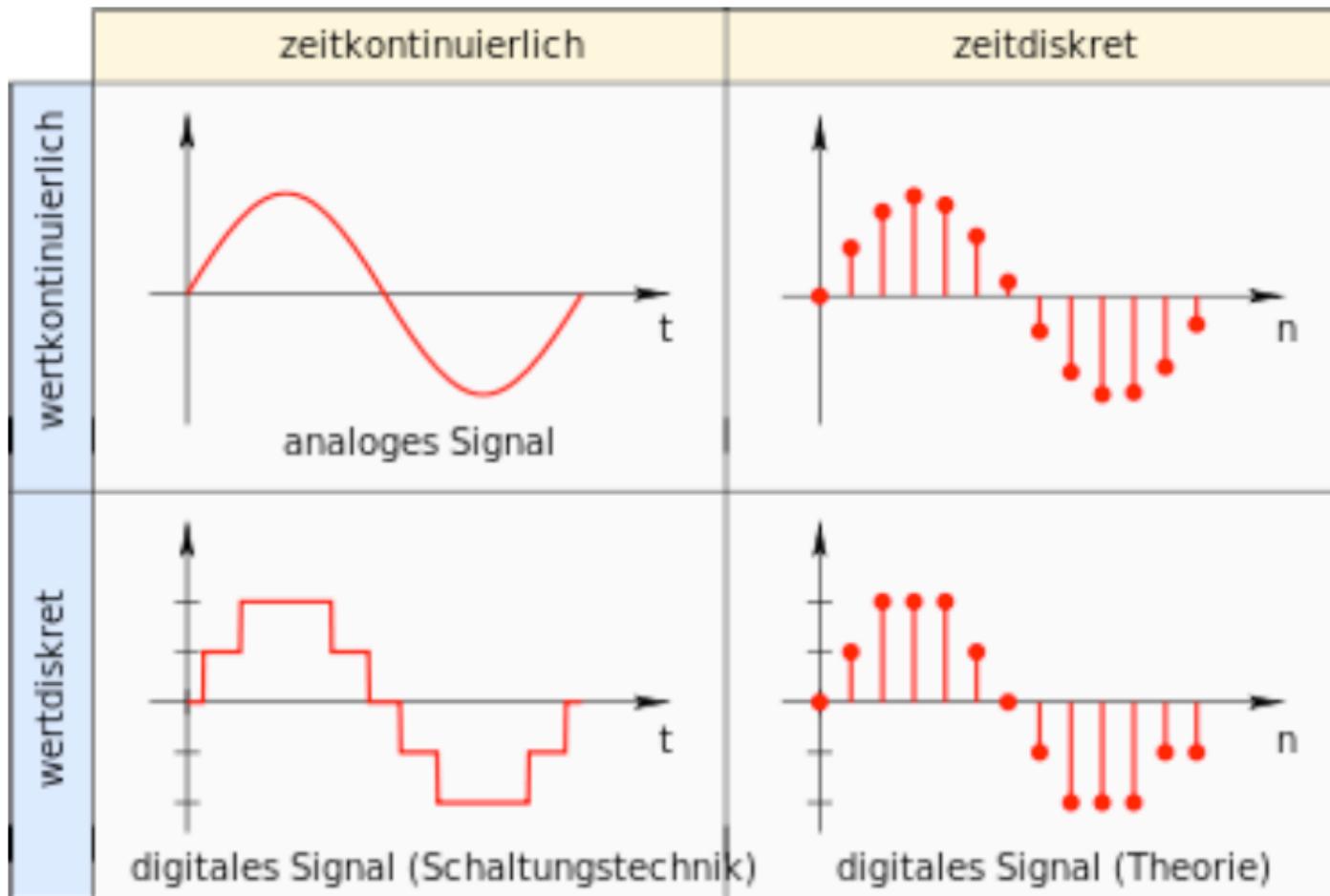
Zweierkomplement bilden $1001 + 0001 = 1010$

$$0010 + 1010 = \cancel{1}00$$



Einführung und Grundbegriffe

Einführung

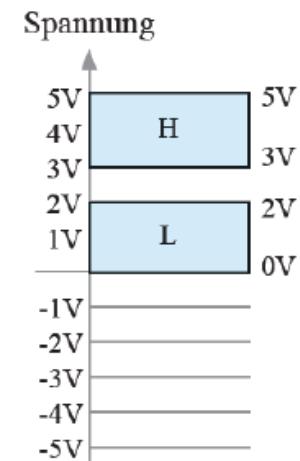
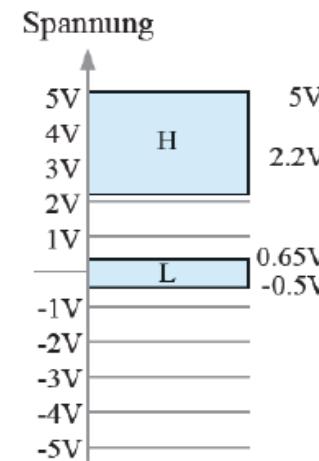
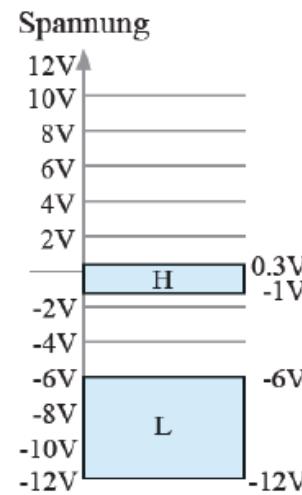
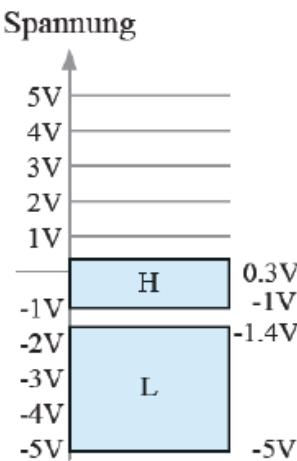
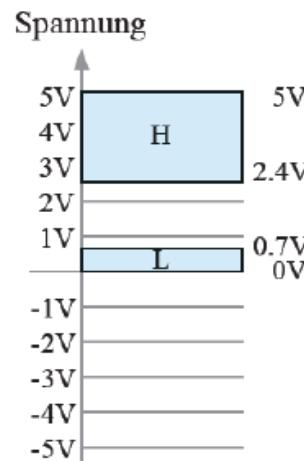




Zuordnung der Amplitudenwerte
zu den Codierungen (00, 01, etc.)

Bei einem digitalen System spricht man auch von diskreten Signalen, die nur die diskreten Werte = 0 und 1 annehmen. Stae 0/1 werden auch oft „OFF/ON“ oder „FALSE/TRUE“ oder „LOW/HIGH“ verwendet.

Einführung



TTL-
Technologie

ECL-
Technologie

PMOS-
Technologie

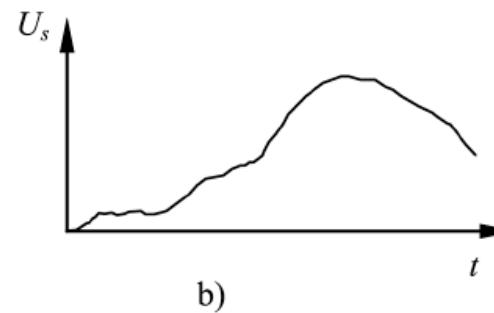
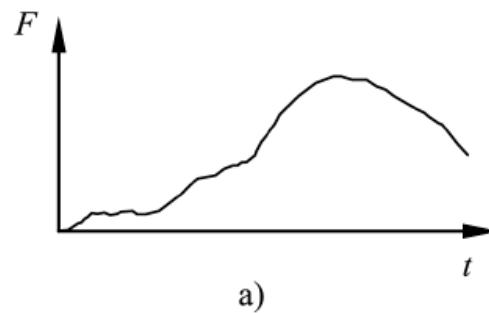
NMOS-
Technologie

CMOS-
Technologie

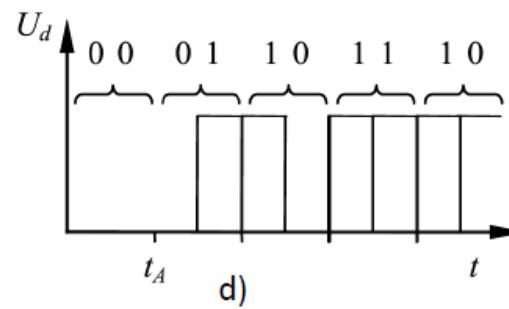
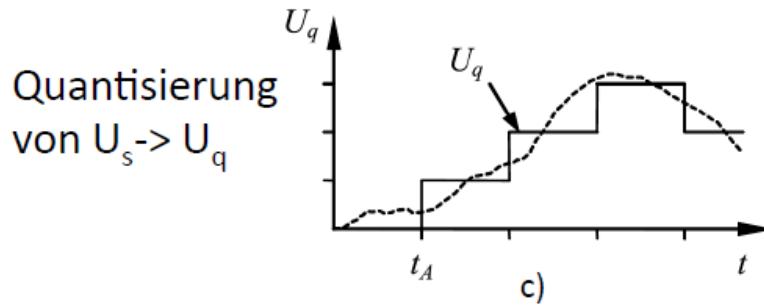
Positiv Logik: „0“ = LOW, „1“ = HIGH

Negativ Logik: „0“ = HIGH, „1“ = LOW

Digitaltechnik erlaubt es, **sehr komplexe Systeme** aufzubauen. Man erreicht dies, indem man sich auf **zwei Signalzustände (0 / 1)** beschränkt, die in logischen Schaltungen (aus sog. **Gattern**) ohne Fehlerfortpflanzung übertragen werden können. Durch diese Beschränkung gelingt es, eine Halbleiter-Technologie aufzubauen, die eine Realisierung von **über 107 logischen Gattern auf einem Chip** ermöglicht.



Analoge Messung



Codierung der
Amplitudenstufen

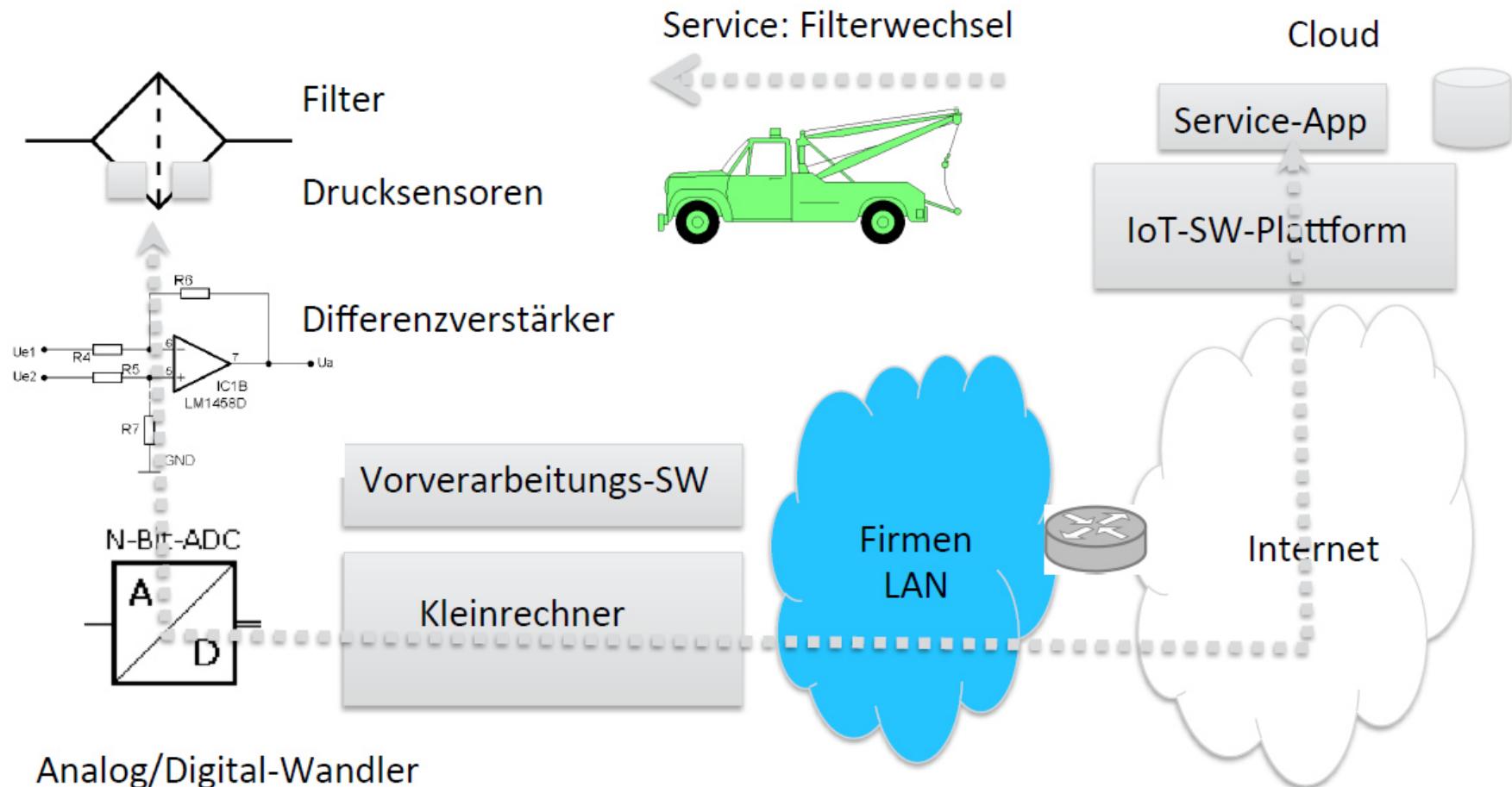
- **Zeitdiskrete Signale:** Änderung der Amplitude (des Wertes) nur zu bestimmten Zeitpunkten.
- **Zeitkontinuierliche Signale** können Ihre Amplitude zu einem beliebigen Zeitpunkt ändern.
- **Zeitdiskrete digitale Systeme:** Synchronisierung über ein Taktsignal (=**synchron**).

Vorteile digitaler Systeme:

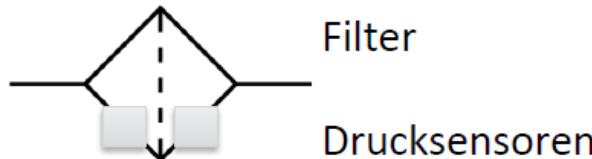
- **Keine Fehlerfortpflanzung**, dadurch sind fast beliebig komplexe Systeme wie zum Beispiel Mikroprozessoren realisierbar. Es können beliebig viele Bearbeitungsschritte nacheinander durchgeführt werden, ohne dass systematische Fehler auftauchen.
Bei Übertragung über weite Strecken ist diese Eigenschaft von Vorteil.

- Eine **hohe Verarbeitungsgeschwindigkeit** kann durch Parallelverarbeitung erzielt werden.
- **Leicht zu konstruieren:** boolesche Algebra stellt eine sehr einfache Beschreibung dar.
Die Entwicklung ist heute durch die Verwendung sehr leistungsfähiger Entwicklungswerkzeuge automatisierbar geworden.
- **Relativ einfach zu testen.**

Digitalisierung anhand des intelligenten Verschmutzungssensors

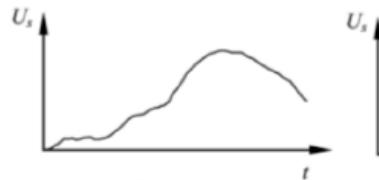


Einführung

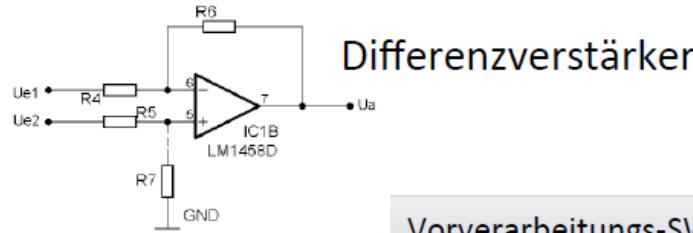


Filter

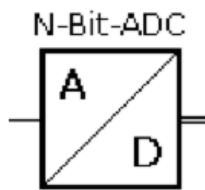
Drucksensoren



Ausgangsspannungen
der zwei Sensoren



Differenzverstärker



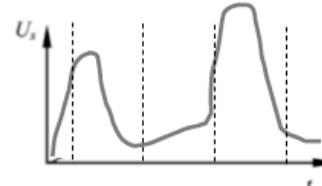
Vorverarbeitungs-SW

Kleinrechner

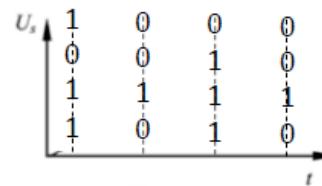
Analog/Digital-Wandler



Differenz der Spannungen
= Maß für Druckdifferenz
= Maß für Verschmutzungsgrad

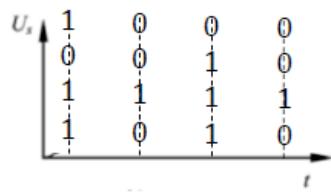
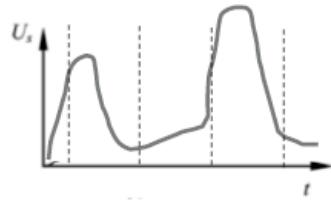


Verstärkung der Differenz
der Spannungen



Abtastung und Digitalisierung
z.B. in 4 Bit

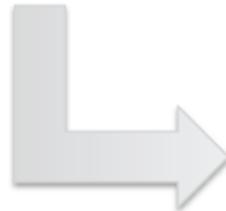
Einführung



Abtastung und Digitalisierung
z.B. in 4 Bit

- Abbildung digitale Eingangswerte -> Verschmutzungsgrad (Kennlinie des (Sensors))

- Empfindlichkeit (Umgang mit Kurzzeitigen Schwankungen / Fehlern)

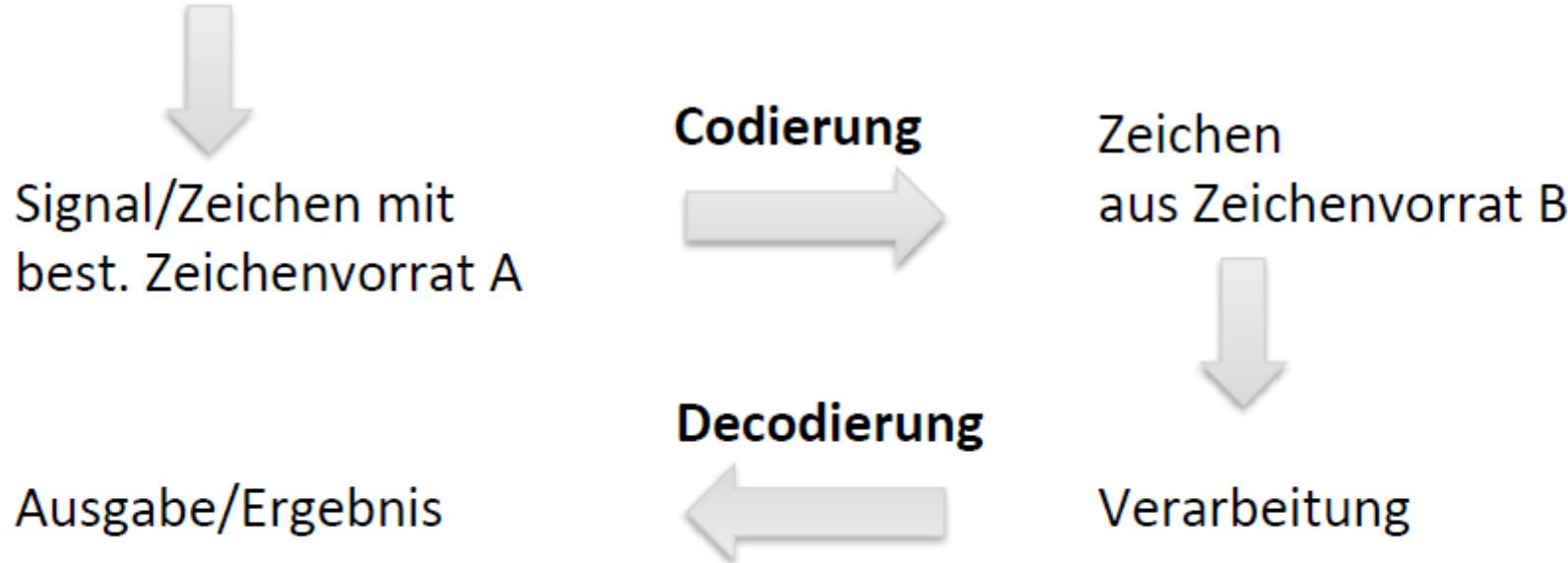


Vorverarbeitungs-SW

Kleinrechner

- Weitergabe der Daten an Host-Rechner

Anwendungsfall





Codes

Codierung und Decodierung

- Bei Computern ist das Codieren der Prozess des Übersetzens einer bestimmten Zeichenfolge (Buchstaben, Ziffern, Satzzeichen oder Symbole) in ein spezielles Format, damit diese effizienter übertragen oder gespeichert werden kann.
- Das Decodieren ist der entsprechende Gegenprozess – die Umwandlung eines codierten Formats zurück in die ursprüngliche Zeichenfolge.
- Codierung und Decodierung werden in der Daten-Kommunikation, in Netzwerken und bei Datenspeichern verwendet

Die Begriffe Codierung und Decodierung werden häufig für den Prozess der Konvertierung von analog zu digital und umgekehrt verwendet.

In diesem Sinne sind diese Begriffe auf jegliche Form von Daten, inklusive Text, Bilder, Video, Multimedia, Computer-Programme, Signale in Sensoren, Telemetrie und Steuerungssysteme anwendbar.

Codierung sollte dabei nicht mit Verschlüsselung verwechselt werden. Letztere ist ein Prozess, bei dem Daten bewusst verändert werden, um ihren Inhalt zu verbergen.

Eine Verschlüsselung kann erfolgen, ohne dass der jeweilige Code, in dem der jeweilige Inhalt vorliegt, geändert wird. Eine Codierung wiederum ist möglich, ohne damit bewusst den Inhalt zu verschleiern.

Codierung

- Verfahren, welches die Symbole einer Nachricht in eine andere Form bringt ohne den Informationsgehalt einzuschränken.
- Codierung wird dazu verwendet, die Informationen von der für Menschen verständliche Form in eine für Maschinen verarbeitbare und über Netzwerke kommunizierbare Form umzuwandeln und wieder zurück.
- Allgemeine Definition [nach DIN 44300 Teil 2 von 1988]:
 - Ein Code ist eine Abbildungsvorschrift, die jedem Zeichen eines Zeichenvorrats (Urbildmenge) eindeutig ein Zeichen oder eine Zeichenfolge aus einem möglicherweise anderen Zeichenvorrat (Bildmenge) zuordnet.

Grundlagen von Codierungen

- Ein Codewort ist eine Folge von Zeichen, die in einem bestimmten Zusammenhang als Einheit betrachtet wird. Ein Codewort kann auch aus nur einem Zeichen bestehen.
- Existiert eine Anordnung der Codeworte und somit eine Größer-Kleiner-Beziehung zwischen zwei beliebigen verschiedenen Codeworten, so nennt man den Codewortvorrat Codealphabet.
- Bei Binärcodes besteht jedes Codewort der Bildmenge aus einer Folge von Binärzeichen.

- Bei der technischen Verarbeitung von Informationen spielen Binärcodes eine überragende Rolle.
- Ein Gerät, das eine Codierung durchführt, nennt man Codierer, Coder oder Encoder.
- Ein Gerät, das eine Codierung rückgängig macht, nennt man Decodierer oder Decoder.
- Kann ein Gerät sowohl codieren als auch decodieren, so nennt man es Codec (Coder/Decoder).

Zweck einer Codierung ist die Umformung der Darstellung von Informationen, um deren

- Übertragbarkeit (hohe Sicherheit gegen Verfälschung, optimale Nutzung der physikalischen Eigenschaften des Übertragungskanals)
- Speicherbarkeit (kompakte Darstellung zur Minimierung des Speicherplatzes, optimale Nutzung der physikalischen Eigenschaften des Speichermediums)
- Verarbeitbarkeit (einfache und schnelle Durchführung von Operationen)
- Lesbarkeit (für Menschen oder Maschinen) zu verbessern. Um diese verschiedenen Zwecke zu erreichen, wurde eine unübersehbare Zahl von Codes entwickelt, insbesondere zur Darstellung von Zahlen (numerische Codes)
- beliebigen Schriftzeichen: Buchstaben, Sonderzeichen, Ziffern (alphanumerische Codes).

American Standard Code for Information Interchange

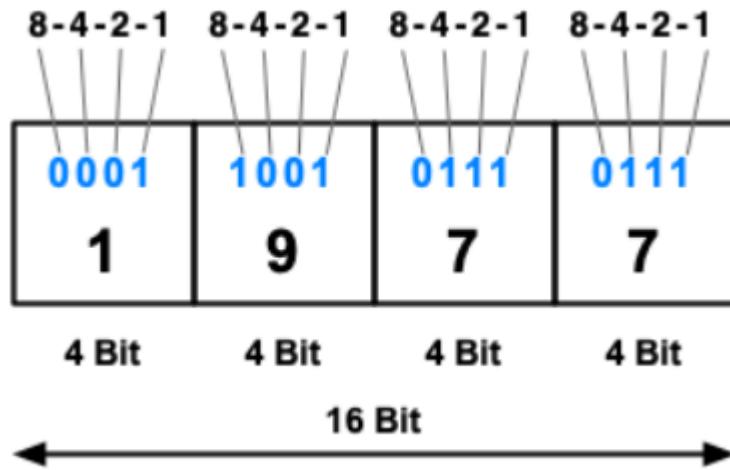
- Der **American Standard Code for Information Interchange (ASCII, alternativ US-ASCII, deutsch „Amerikanischer Standard-Code für den Informationsaustausch“)** ist eine 7-Bit-Zeichenkodierung.
- sie entspricht der US-Variante von ISO 646 und dient als Grundlage für spätere, auf mehr Bits basierende Kodierungen für Zeichensätze.
- Der ASCII-Code wurde zuerst am 17. Juni 1963 von der American Standards Association (ASA) als Standard ASA X3.4-1963 gebilligt und 1967/1968 wesentlich sowie zuletzt im Jahr 1986 (ANSI X3.4-1986) von ihren Nachfolgeinstitutionen aktualisiert und wird bis heute noch benutzt.
- Die Zeichenkodierung definiert 128 Zeichen, bestehend aus 33 nicht druckbaren sowie den folgenden 95 druckbaren Zeichen, beginnend mit dem Leerzeichen.

Tabelle: Zeichentabelle des ASCII mit Hexadezimal und Binärkode

BIT 5–7	0	1	2	3	4	5	6	7
1–4	000	001	010	011	100	101	110	111
0/ 0000	NUL	DLE	SP	0	@	P	‘	p
1/ 0001	SOH	DC1	!	1	A	Q	a	q
2/ 0010	STX	DC2	“	2	B	R	b	r
3/ 0011	ETX	DC3	#	3	C	S	c	s
4/ 0100	EOT	DC4	\$	4	D	T	d	t
5/ 0101	ENQ	NAK	%	5	E	U	e	u
6/ 0110	ACK	SYN	&	6	F	V	f	v
7/ 0111	BEL	ETB	,	7	G	W	g	w
8/ 1000	BS	CAN	(8	H	X	h	x
9/ 1001	HT	EM)	9	I	Y	i	y
A/ 1010	LF	SUB	*	:	J	Z	j	z
B/ 1011	VT	ESC	+	;	K	[k	{
C/ 1100	FF	FS	,	<	L	\	l	
D/ 1101	CR	GS	-	=	M]	m	}
E/ 1110	SO	RS	.	>	N	^	n	~
F/ 1111	SI	US	/	?	O	-	o	DEL

BCD-Code - Binary Coded Decimals

- **BCD-Code** steht im englischen für **Binary Coded Decimal**, also *dualkodierte Dezimalziffer*. Dabei wird jede dezimale Ziffer 0 bis 9 durch jeweils vier Bit dargestellt (0000 bis 1001, siehe Codetabelle), also in einem Halbbyte (Nibble).
- Eine andere Bezeichnung ist **8-4-2-1-BCD-Code**. Die Ziffernfolge 8-4-2-1 steht dabei für die Werte der Stellen in einer dualkodierten Dezimalziffer.



Codes

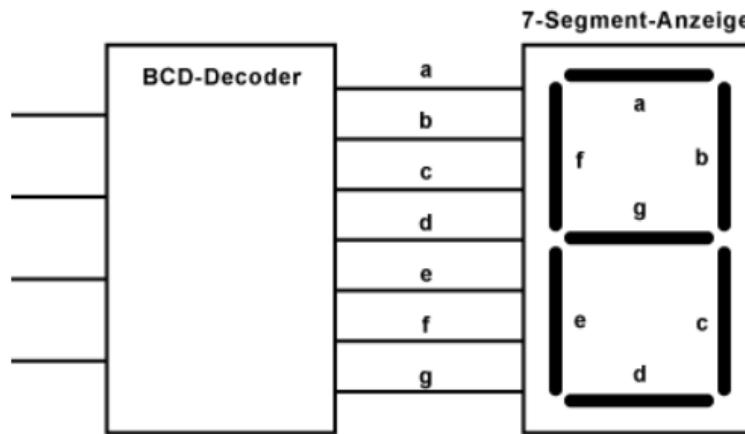
- Jede Dezimalziffer der Dezimalzahl wird durch eine eigene 4-Bit-Dualzahl bzw. binären Code ausgedrückt.
- Man nennt die 4 Bit eine Tetrade (griechisch: Vierergruppe).
- Für die 10 Dezimalziffer werden nur 10 Tetraden benötigt. Die 6 weiteren Tetraden werden Pseudotetraden genannt und gehören nicht mehr dazu.
- Sie entfallen bzw. haben keine Funktion. Sie treten im BCD-Code nicht auf bzw. dürfen nicht auftreten.

Dezimal	2^3	2^2	2^1	2^0	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	Tetraden
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
	1	0	1	0	
	1	0	1	1	
	1	1	0	0	Pseudotetraden
	1	1	0	1	
	1	1	1	0	
	1	1	1	1	

Codes

- Die BCD-Codierung wird in der Digitaltechnik und digitale Anzeigen eingesetzt. Zum Beispiel als 7-Segment-Anzeige.

7-Segment-Anzeige



- Digitaluhren laufen im Innern mit BCD-Code. Auch die Uhrzeit im DCF77-Signal ist als BCD-Code kodieren.

Die Hamming-Distanz

- Benannt ist sie nach dem amerikanischen Mathematiker Richard Wesley Hamming (1915-1998).
- Sie wird auch Hamming-Abstand oder Hamming-Gewicht genannt und ist ein Maß für die Unterschiedlichkeit von Codewörtern.
- Sie kann für sämtliche Alphabete und Zahlensysteme genutzt werden, am häufigsten ist jedoch die Nutzung am Binärsystem.
- Die Hamming-Distanz wird zur Fehlererkennung und zur Fehlerkorrektur benutzt, indem Dateneinheiten, die über eine Übertragungsstrecke empfangen werden, mit gültigen Zeichen verglichen werden.
- Eine etwaige Korrektur der Zeichen erfolgt nach dem Wahrscheinlichkeitsprinzip.

Um die Hamming-Distanz verstehst, codieren wir nun acht Codewörter, die den dezimalen Werten Null bis Sieben entsprechen.

Dezimaler Wert	
Dezimal	Binär
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- Die Anzahl der sich unterscheidenden Bits zweier Codewörter ist die Hamming-Distanz.
- Wir erweitern unsere Tabelle um eine neue Spalte, in die wir den Hamming-Abstand aller Codewörter zum ersten Codewort mit dem dezimalen Wert 0 eintragen.

Fangen wir mit der ersten Zeile an:

Das Codewort 0-0-0 unterscheidet sich in keinem Bit von dem Codewort 0-0-0.

- Die Hamming-Distanz eines Codewortes zu sich selbst ist also immer 0.
- Das zweite Codewort 0-0-1 unterscheidet sich nur in einem Bit von dem ersten Code-wort 0-0-0 – der Hamming-Abstand ist also 1.
- Genauso ist es beim dritten Codewort 0-1-0. Beim vierten Codewort 0-1-1 ist den Hamming-Abstand dementsprechend 2.
- Man musst also einfach nur die Anzahl der unterschiedlichen Bits zählen.

Dezimaler Wert Dezimal	Binärer Code Binär	Hamming- Distanz
0	000	(0)
1	001	1
2	010	1
3	011	2
4	100	1
5	101	2
6	110	2
7	111	3

Um jetzt bei zwei etwas längeren Codewörtern den Hamming-Abstand zu ermitteln muss man abzählen an wie vielen Stellen sich die Codes unterscheiden.

Code 1	1010	1111	0100
Code 2	0111	1001	1101

✗ ✗ ✓ ✗ ✓ ✗ ✗ ✓ ✗ ✓ ✓ ✗

→ 7 ✗ 5 ✓

→ Hamming-Distanz = 7

Der Hamming-Abstand wird zur Fehlererkennung und Fehlerkorrektur genutzt.

Bitfehler können zum Beispiel beim Übertragen von Codewörtern entstehen. Ob ein fehlerhaftes Codewort erkannt oder korrigiert wird, hängt von der Hamming-Distanz ab.

Dazu schauen wir die Tabelle mit den Hamming-Distanzen an und suchen den minimalen Wert zwischen zwei gültigen Codewörtern.

Bei einer minimalen Hamming-Distanz von d gilt:

- Es sind $d-1$ Fehler erkennbar.
- Es sind e gleich d minus 1 durch 2 Bitfehler korrigierbar.

Beispiel:

- Einen Getränkeautomat hat viele verschiedene Zustände.
- Der Zustand Null ist üblicherweise der Ruhezustand, also der Zustand in dem der Getränkeautomat zwar läuft aber niemand eine Taste gedrückt hat oder Geld eingeworfen hat.
- Damit beim Kauf einer Cola auch Cola rauskommt müssen die Bits stets überprüft werden.
- Der Zustand „Cola-Preis anzeigen“ hat nun das Codewort 0-0-1-1, das Codewort für den Zustand „Cola ausgeben“ ist 1-0-1-1.
- Bei einer Codierung sollen nun mindestens zwei Bitfehler korrigierbar sein, und mindestens drei gekippte Bits erkannt werden.

Da im Getränkeautomaten drei Bitfehler erkannt werden sollen, ist die minimal erlaubte Hamming-Distanz zwischen den Codewörtern 4.

Für die Korrektur von zwei Bits braucht die Codierung einen minimalen Hamming-Abstand von 5.

Letztendlich müssen die Codewörter zur Definition der Zustände also eine Hamming-Distanz von 5 haben und damit kann man sogar 4 Bitfehler erkennen.

Für eine Fehlererkennung muss der Hamming-Abstand also möglichst groß sein.

$$HD_{\min} = \text{FE} + 1 = 4$$

$$HD_{\min} = \text{FK} \cdot 2 + 1 = 5$$

$$\text{FE} = 3 \quad \text{FK} = 2$$

FE = Fehlererkennung FK = Fehlerkorrektur

Die Codierung der dezimalen Werte null bis sieben eignet sich bei einer minimalen Hamming-Distanz von eins also weder zur Fehlererkennung noch zu deren Korrektur.

- Wenn man annimmt, dass nur Einfachfehler auftreten und diese korrigieren möchte, werden Codewörter vereinbart, die jeweils einen Hamming-Abstand ≥ 3 haben, z. B. 01011, 01100, 10010, 10101.
- Wenn der Wert 01111 auftritt und angenommen wird dass ein Einfachfehler aufgetreten ist, dann kann 01111 nur aus dem gültigen Codewort 01011 entstanden sein, bei dem das mittlere Bit verändert wurde.
- Doppelfehler können ebenfalls erkannt werden. Da nur bestimmte Codewörter verwenden, die sich um mindestens drei Bit (Hamming-Abstand ≥ 3) unterscheiden, fällt auch ein Doppelfehler (nur zwei Bits geändert) auf, der aber mit den gesendeten Informationen nicht korrigiert werden kann.
- Dreifachfehler können nicht mehr erkannt werden, doch die Relevanz von mehrfachen Fehlern nimmt in technischen Systemen ab, da das gleichzeitige Auftreten mehrerer Fehler immer unwahrscheinlicher wird, je mehr Fehler zusammentreffen sollen.

1-aus-n-Code

- Ein **1-aus-n-Code**, auch **One-Hot-Kodierung**, stellt Zahlen binär dar, gewöhnlich für den Einsatz in der Digitaltechnik bzw. Computern.
- Eine dezimale Ziffer wird im 1-aus-n-Code durch n Bits dargestellt, wobei jeweils nur ein Bit auf 1 gesetzt ist, während die restlichen $n-1$ Bits 0 sind.

Beispiel für 1-aus-n-Code mit n=10

Dezimal-ziffer	1-aus-10-codiert	Binär-codiert
0	0000000001	0 0 0 0
1	0000000010	0 0 0 1
2	0000000100	0 0 1 0
3	0000001000	0 0 1 1
4	0000010000	0 1 0 0
5	0000100000	0 1 0 1
6	0001000000	0 1 1 0
7	0010000000	0 1 1 1
8	0100000000	1 0 0 0
9	1000000000	1 0 0 1

Codes

- Der Hamming-Abstand beträgt 2, weshalb 1-Bit-Fehler bemerkt (indem man feststellt, ob die Quersumme genau 1 ist), aber nicht korrigiert werden können.
- 2-Bit-Fehler können nicht zuverlässig entdeckt werden.
- Der Code ist sehr redundant, denn n Bit könnten bis zu 2^n verschiedene Zahlen kodieren.
- Der 1-aus-n-Code findet Anwendung insbesondere bei der Steuerung von Zustandsautomaten, der Speicheradressierung, beim maschinellen Lernen sowie in Tastenfeldern, Anzeigetafeln und Maschinensteuerungen

Beispiel für 1-aus-n-Code mit n=10

Dezimal-ziffer	1-aus-10-codiert	Binär-codiert
0	0000000001	0 0 0 0
1	0000000010	0 0 0 1
2	0000000100	0 0 1 0
3	0000001000	0 0 1 1
4	0000010000	0 1 0 0
5	0000100000	0 1 0 1
6	0001000000	0 1 1 0
7	0010000000	0 1 1 1
8	0100000000	1 0 0 0
9	1000000000	1 0 0 1

Gray Code

- Der Gray Code ist eine andere Darstellungsform des Binärcodes.
- Seine Grundlage besteht darin, dass sich zwei benachbarte Grayzahlen in nicht mehr als einem Bit unterscheiden dürfen.
- Der Gray Code ist somit ein einschrittiger Code
- Motivation für die Entwicklung dieses Codes ist das folgende Problem:
 - Auf mehreren Adern einer elektrischen Datenleitung sollen Daten parallel übertragen werden, die sich stetig (also immer nur um ein Digit) ändern, typisch dafür sind z. B. Signale eines Temperatursensors oder eines Drehwinkelgebers.

- Als Dualzahl übertragen, ändern sich die Bits bei einem neuen Messwert auf jeder betroffenen Leitung theoretisch exakt gleichzeitig, und zwar sowohl am Eingang der Leitung als auch am Ausgang.
- Tatsächlich aber ändern sich die Bits auf der Leitung nicht gleichzeitig. Das kann verschiedene Ursachen haben: Bauteilestreuung, Laufzeiten, Asymmetrien usw.
- Dadurch kommt es zu ungewollten Zwischenzuständen und kurzzeitig (zwischen den roten Linien) falsch empfangenen Werten



Codes

2-Bit-Gray-Code:

00
01
11
10

3-Bit-Gray-Code:

000
001
011
010
110
111
101
100

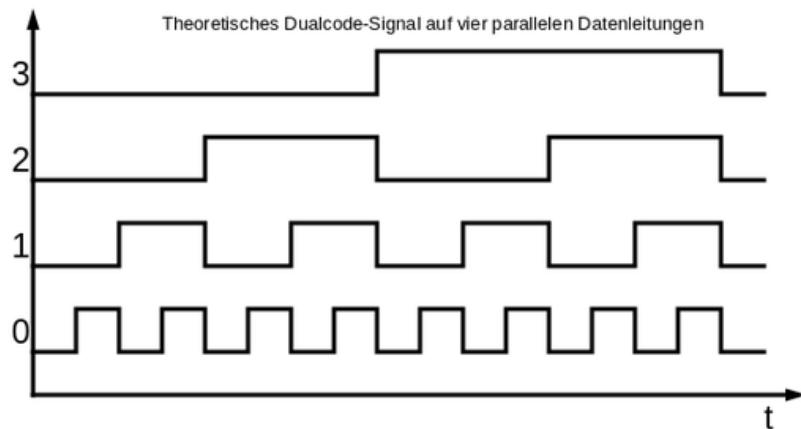
4-Bit-Gray-Code:

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

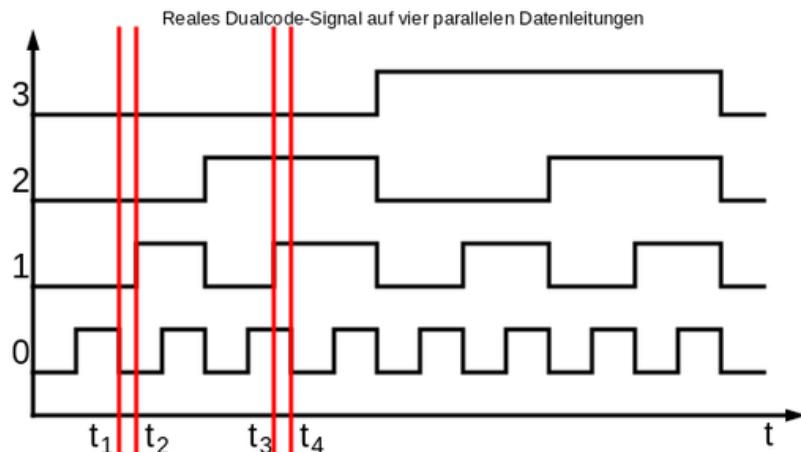
5-Bit-Gray-Code:

00000	11011
00001	11010
00011	11110
00010	11111
00110	11101
00111	11100
00101	10100
00100	10101
01100	10111
01101	
01111	
01110	
01010	
01011	
01001	
01000	
11000	

Problem bei Dualcode-Signalen



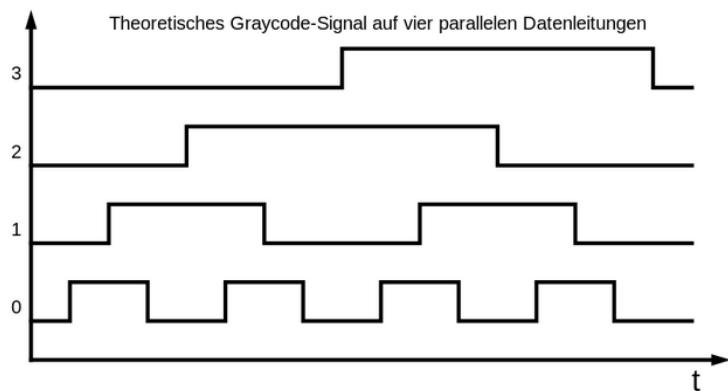
Während das theoretische Signal in der Reihenfolge
 $\{0000\}, \{0001\}, \{0010\}, \{0011\}, \{0100\},$
 $\{0101\}, \{0110\}, \{0111\}$ usw.
abgesendet wird,



kommen am Ausgang kurzzeitig andere
Signalzustände an:
 $\{0000\}, \{0001\}, \{\textbf{0000}\}, \{0010\}, \{0011\},$
 $\{0100\}, \{0101\}, \{\textbf{0111}\}, \{0110\}, \{0111\}$ usw.

Codes

Lösung mit Gray-Code

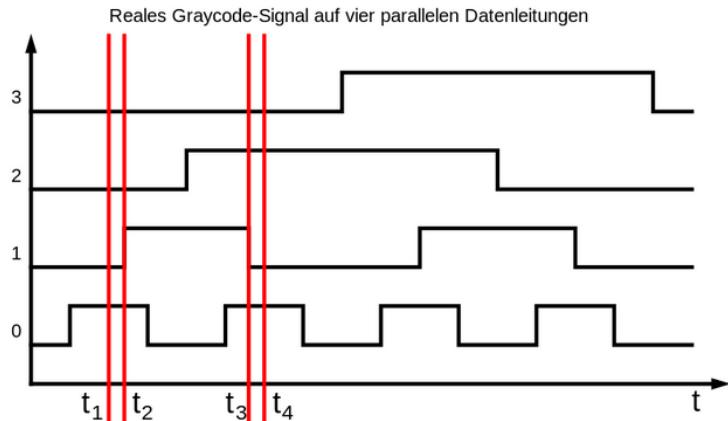


Um das zu vermeiden, werden die Steuersignalzustände mittels Gray-Code abgesendet, sodass sich immer nur ein Bit gleichzeitig ändert:

Abgesendete Sequenz: {0000}, {0001}, {0011}, {0010}, {0110}, {0111}, {0101}, {0100} usw.

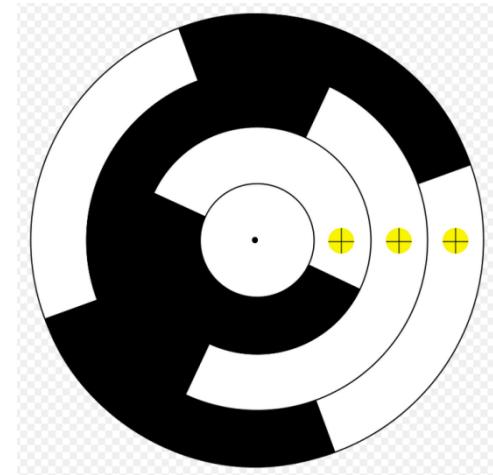
Ankommende Sequenz: {0000}, {0001}, {0011}, {0010}, {0110}, {0111}, {0101}, {0100} usw.

Hier kommt also am Ausgang auch dann die gleiche Sequenz wie am Eingang an, wenn beachtliche Zeitfehler (rote Linien) auftreten.

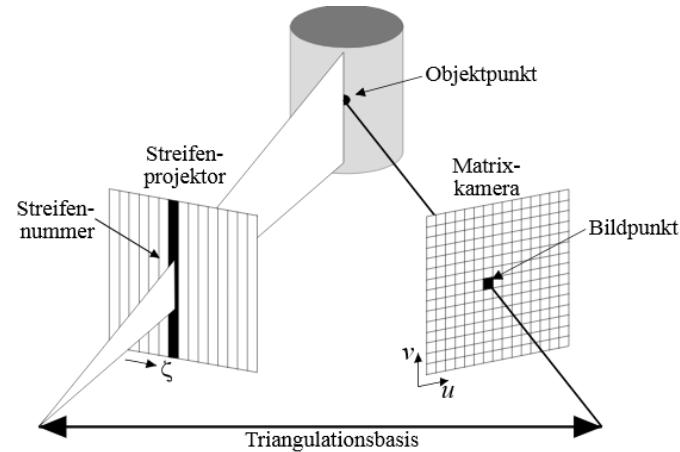


Anwendungen

Eine Anwendungsmöglichkeit ist die Bestimmung der absoluten Position einer Scheibe oder Leiste, die mit schwarzen und weißen Balken markiert ist, die mit Lichtschranken oder anderen Sensoren abgetastet werden. Diese Position wird dann zur Winkel- oder Drehgeschwindigkeitsmessung verwendet.



Eine weitere Anwendung ist die Streifenprojektion. Dort wird eine Folge von Mustern aus parallelen Streifen auf ein Objekt projiziert. Die Nummer der Streifen ist Gray-kodiert und kann von einer beobachtenden Kamera für jeden Bildpunkt berechnet werden.





Logische Verknüpfungen und ihre Darstellung

Logische Verknüpfungsglieder

- In Digitalschaltungen werden logische Steuersignale zusammengeführt und nach festgelegten Gesetzen der Booleschen Algebra ausgewertet.
- George Boole entwickelte um 1850 eine Algebra, die mit zwei Konstanten 0 und 1 und den Operatoren UND, ODER sowie NICHT auskommt.
- Jede Variable kann nur zwischen den Werten 0 und 1 wechseln.
- Das Ergebnis ist ebenfalls als Variable zu betrachten und kann daher auch nur die Werte 0 oder 1 annehmen.
- Die Schreibweise der Operatorsymbole ist nicht immer einheitlich.

Logische Verknüpfungen

Operator		Symbol	Beispiel	
UND	AND	\wedge	\cdot	$A \wedge B$ $A \cdot B$
ODER	OR	\vee	$+$	$A \vee B$ $A + B$
NICHT	NOT	\neg	$\overline{}$	$\neg A$ \overline{A}

Symbol	Verwendung	Interpretation	Artikel
\wedge	$A \wedge B$	Aussage A und Aussage B	<u>Konjunktion (Logik)</u>
\vee	$A \vee B$	Aussage A oder Aussage B (oder beide)	<u>Disjunktion</u>
\Leftrightarrow	$A \Leftrightarrow B$	Aussage A folgt aus Aussage B und umgekehrt	<u>Logische Äquivalenz</u>
\leftrightarrow	$A \leftrightarrow B$	aus Aussage A folgt Aussage B	<u>Implikation</u>
\Rightarrow	$A \Rightarrow B$		
\rightarrow	$A \rightarrow B$	entweder Aussage A oder Aussage B	<u>Kontravalenz/Antivalenz</u>
\approx	$A \approx B$		
\oplus	$A \oplus B$		
$\vee\!\vee$	$A \vee\!\vee B$		
$\dot{\vee}$	$A \dot{\vee} B$		
$\leftrightarrow\!\leftrightarrow$	$A \leftrightarrow\!\leftrightarrow B$		
\nparallel	$A \nparallel B$	nicht Aussage A	<u>Negation</u>
\neg	$\neg A$		
$\overline{}$	\overline{A}		

Logische Verknüpfungen

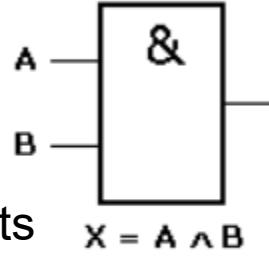
- Die Grundglieder der Digitaltechnik werden nachfolgend nur als Blockdiagramme dargestellt.
- Die interne elektronische Schaltung, die aus bipolaren Transistoren, MOSFET Transistoren, Diode und Widerständen bestehen kann, soll an dieser Stelle nicht weiter interessieren.
- Bei den Blockschaltbildern der Verknüpfungsglieder werden nur die Ein- und Ausgänge gezeichnet.
- Die logischen Eingangswerte werden nach den mathematischen Gesetzen der booleschen Algebra ausgewertet und ergeben einen logischen Ergebniswert.
- Die Gesetzmäßigkeiten der Funktionsgleichungen werden oft in Pegel- und Wahrheits- oder Funktionstabellen erfasst.

UND-Verknüpfung (AND) – Konjunktion

- Für eine UND-Verknüpfung, im englischen Sprachgebrauch AND, sind mindestens zwei Eingangsvariable ($n=2$) notwendig.
- Die Funktion liefert als Ergebnis eine Ausgangsvariable.
- Bei binären (2) Zuständen am Eingang sind $2^n = 4$ unterschiedliche Kombinationen möglich.
- Sie werden nach Eingangsvariablen getrennt in einer Arbeitstabelle mit Low und High oder Wahrheitstabelle mit 0 und 1 erfasst.
- In der letzten Spalte ist das Funktionsergebnis notiert.
- Bei einem logischen UND (AND) müssen alle Eingangswerte wahr oder 1 sein, damit das Ergebnis wahr oder 1 ergibt.

Logische Verknüpfungen

Stellt man sich die Eingangsvariablen als in Reihe liegende Schalter vor, so kann bei anliegender Spannung nur dann Strom fließen, wenn in diesem Beispiel beide Schalter gleichzeitig geschlossen sind.

- Die Reihenfolge der Eingangszustände in der Tabelle ist freigestellt.
-  $x = A \wedge B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1
- Schaltzeichen

Funktionsgleichung

elektromechanisches Schaltprinzip

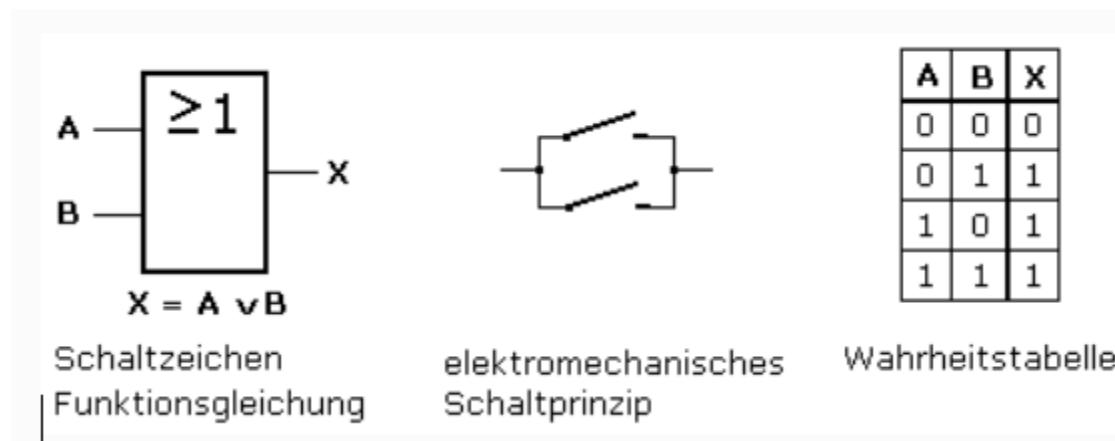
Wahrheitstabelle
- Ein reihenweise von rechts nach links im Sinn zunehmender Dualzahlen eingetragener Wechsel der Eingangsspegel hat sich als besonders übersichtlich erwiesen.
 - Das Ergebnis einer UND Funktion wird true, wenn alle Eingangsvariablen true sind.
 - Der Ausgangszustand eines UND-Glieds liefert 1, wenn alle Eingangszustände 1 sind.

ODER-Verknüpfung (OR) – Disjunktion

- Für eine ODER-Verknüpfung, im englischen Sprachgebrauch OR, sind mindestens zwei Eingangsvariable notwendig.
- Die Funktion liefert als Ergebnis eine Ausgangsvariable.
- Es gibt ebenso viele unterschiedliche Eingangskombinationen wie beim UND-Gatter.
- Die Ausgangsvariable ist immer dann wahr oder 1, wenn mindestens eine Eingangsvariable wahr oder 1 ist.

Logische Verknüpfungen

Im elektromechanischen Schaltprinzip kann das Verhalten durch zwei im Stromkreis parallel angeordnete Schalterelemente erreicht werden. Bei anliegender Spannung fließt Strom, wenn einer der Schalter geschlossen ist.

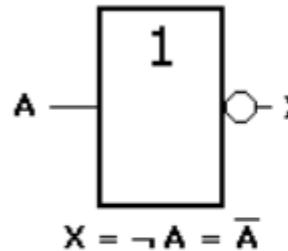


- Das Ergebnis einer ODER Funktion wird true, wenn eine Eingangsvariable true ist.
- Der Ausgangszustand eines ODER-Glieds liefert 1, wenn mindestens ein Eingangszustand 1 ist.

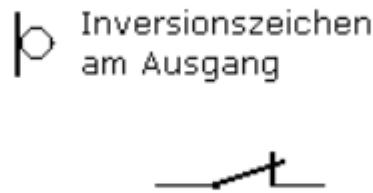
NICHT-Verknüpfung (NOT) – Negation

- Die dritte Grundverknüpfung ist die Negation.
- Für ein NICHT-Gatter, im englischen Sprachgebrauch NOT, ist nur eine Eingangsvariable notwendig.
- Das Ausgangssignal kehrt den Eingangszustand um. Das NICHT-Gatter funktioniert als Inverter.
- Im elektromechanischen Schaltprinzip liegt im Stromkreis ein Öffner, der ein genormtes Schaltzeichen hat.

Logische Verknüpfungen



Schaltzeichen
Funktionsgleichung



elektromechanisches
Schaltprinzip

A	X
0	1
1	0

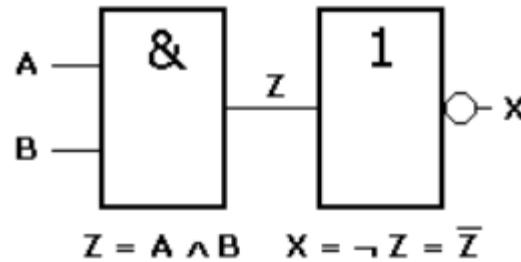
Wahrheitstabelle

- Das Ergebnis einer NICHT-Funktion wird true (false), wenn die Eingangsvariable false (true) ist.
- Der Ausgangszustand eines NICHT-Glieds ist entgegengesetzt (invers) zum Eingangszustand.
- Mit diesen drei Logikgattern lassen sich alle denkbaren logischen Funktionen darstellen. Sie werden daher als Grundgatter bezeichnet.
- Zur Verkleinerung des Schaltungsaufwands wurden weitere zusammengesetzte Gatter entwickelt, die nachfolgend vorgestellt werden.

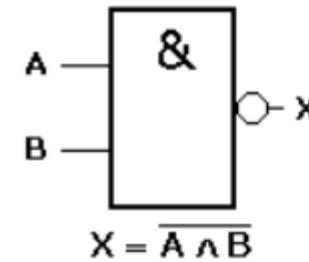
NAND-Verknüpfung – negiertes UND

- Eine Kombination aus einem UND-Gatter mit nachfolgendem NICHT-Gatter ergibt ein NAND-Gatter.
- Die Ausgangsvariable Z des UND-Gatters wird durch das NICHT-Gatter negiert und erzeugt die Ausgangsvariable X des NAND-Glieds.
- Viele logische Funktionen lassen sich durch den Einsatz von NAND-Gattern lösen.
- Oft steigt dadurch die Anzahl der notwendigen Gatter, mit dem wirtschaftlichen Vorteil nur einen Gattertyp zu verwenden.

Logische Verknüpfungen



UND-NICHT Schaltung



NAND genormtes
Schaltzeichen

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

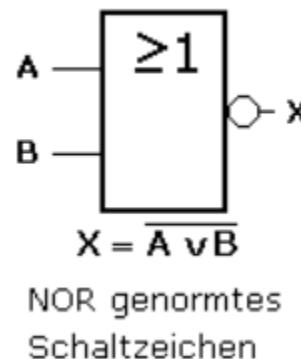
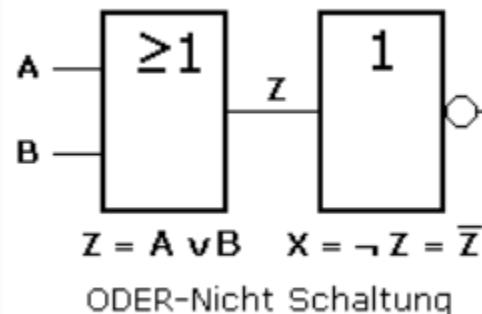
Wahrheitstabelle

Der Ausgangszustand eines NAND-Glieds ergibt 1, wenn nicht alle Eingangszustände 1 sind.

Logische Verknüpfungen

NOR-Verknüpfung – negiertes ODER

- Eine Kombination aus einem ODER-Gatter mit nachfolgendem NICHT-Gatter ergibt ein NOR-Gatter.
- Die Ausgangsvariable Z des ODER-Gatters wird durch das NICHT-Gatter negiert und erzeugt die Ausgangsvariable X des NOR-Glieds.
- NOR-Glieder haben in logischen Schaltungen die gleiche wichtige Bedeutung wie NAND-Glieder.
- Der Ausgangszustand eines NOR-Glieds ergibt 1, wenn alle Eingangszustände 0 sind.



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

Wahrheitstabelle

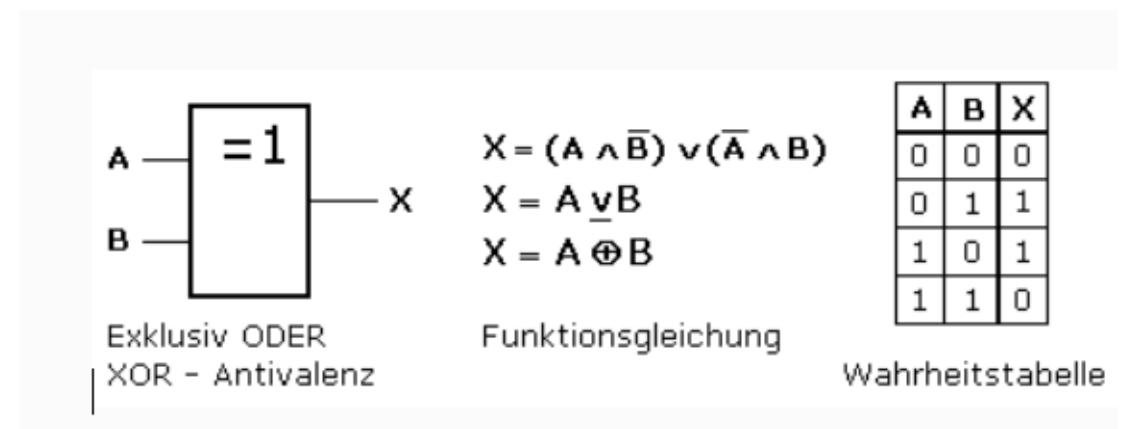
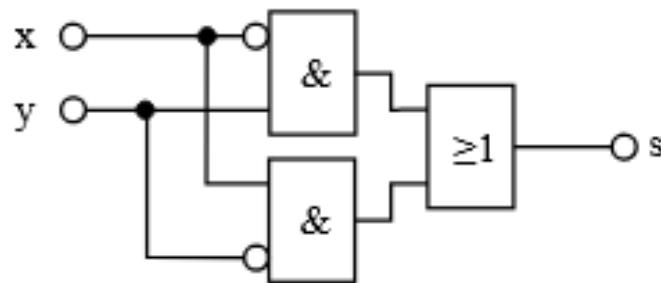
Logische Verknüpfungen

XOR-Verknüpfung – Exklusiv ODER – Antivalenz

- Die logische Verknüpfung des XOR-Gatters mit zwei Eingangsvariablen kann mit einem 'entweder - oder' umschrieben werden.
- Die Ausgangsvariable wird immer dann 'true' liefern, wenn die Eingangsvariablen unterschiedliche Zustände haben.
- Die Wahrheitstabelle des XOR-Gatters entspricht dem ODER-Gatter mit dem Ausschluss gleicher Eingangszustände, also exklusiv der Äquivalenz.
- Dieses Verhalten wird als Antivalenz bezeichnet. Es gibt ein nach IEC 60617-12 genormtes Schaltzeichen.

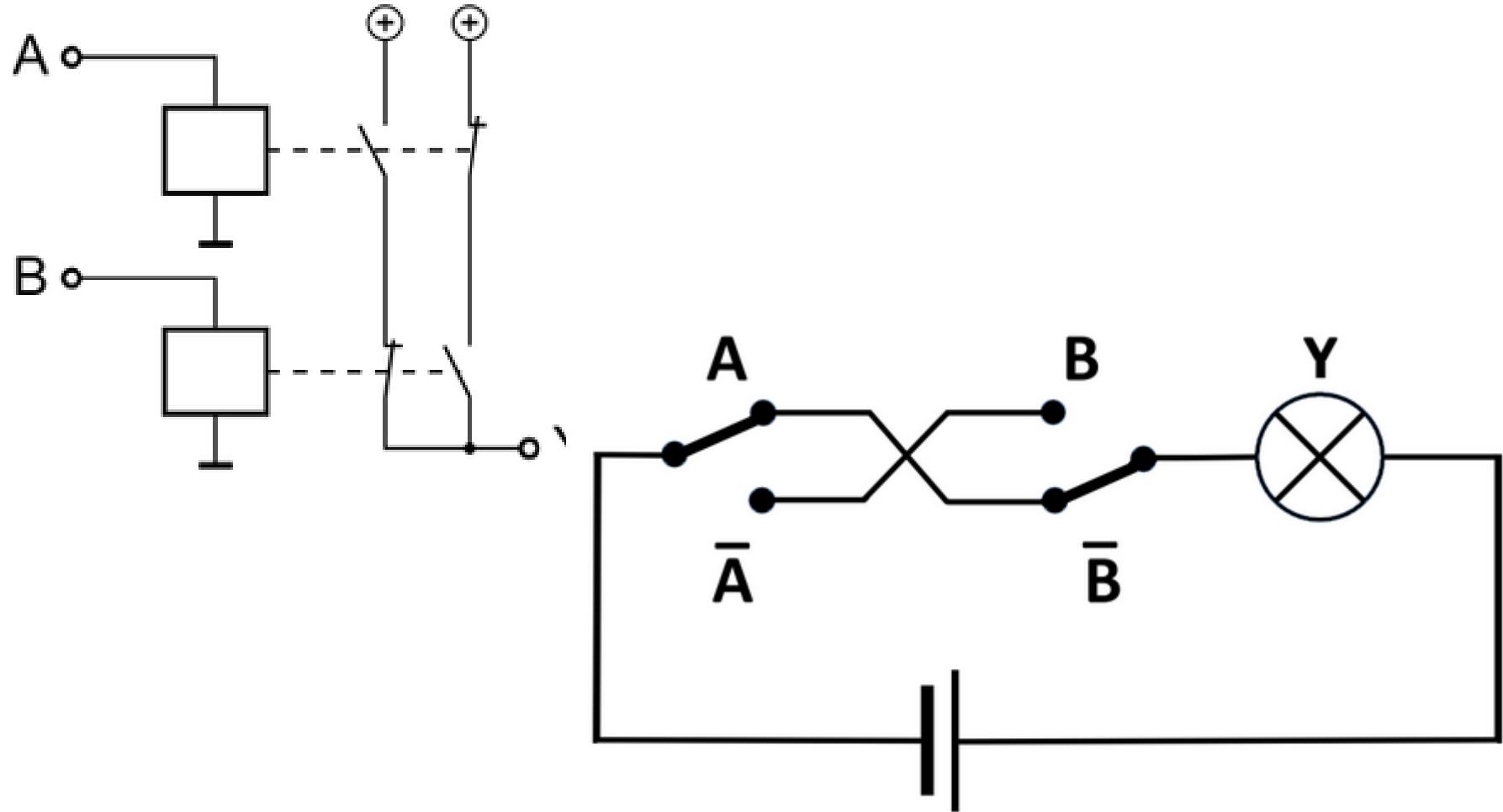
Logische Verknüpfungen

Wird in der Funktionsgleichung für die ODER Verknüpfung das v als Zeichen verwendet, kann die XOR Verknüpfung durch ein unterstrichenes v gekennzeichnet werden. Bei Verwendung des + Zeichens für ODER kennzeichnet ein Plus im Kreis \oplus für die XOR-Verknüpfung.



Der Ausgangszustand eines XOR-Glieds ergibt 1, wenn eine ungerade Zahl der Eingangszustände 1 und alle anderen Eingangszustände 0 sind.

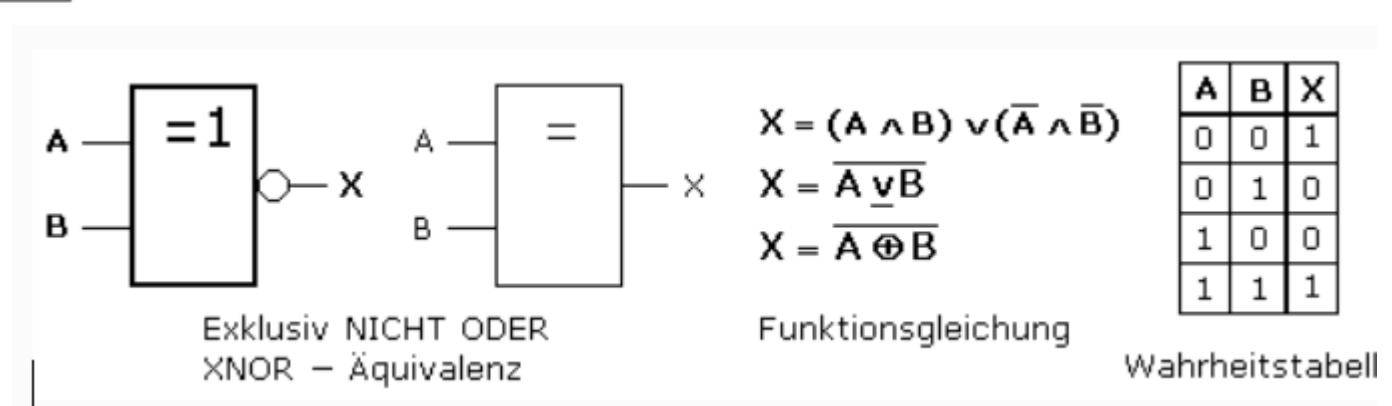
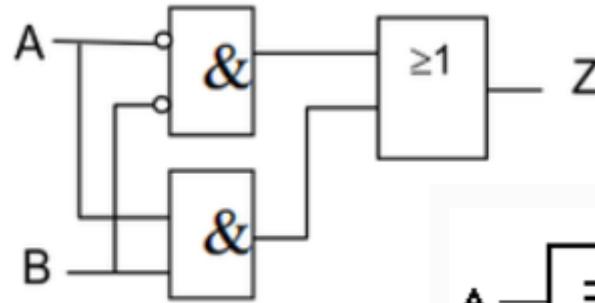
Logische Verknüpfungen



XNOR-Verknüpfung – Äquivalenz

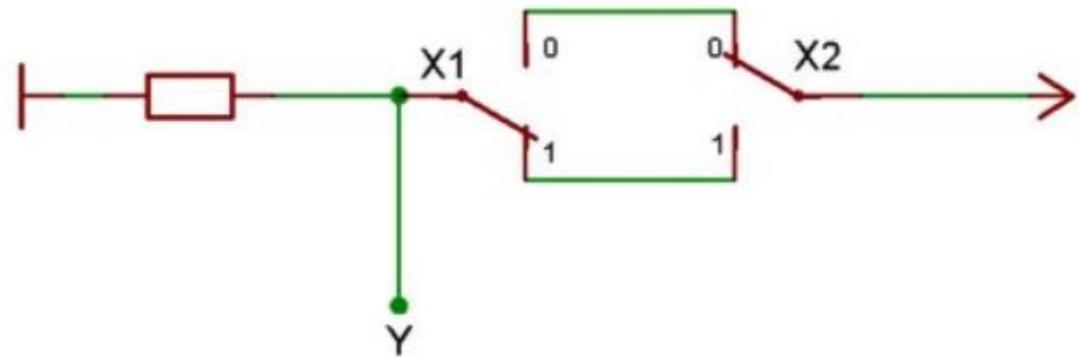
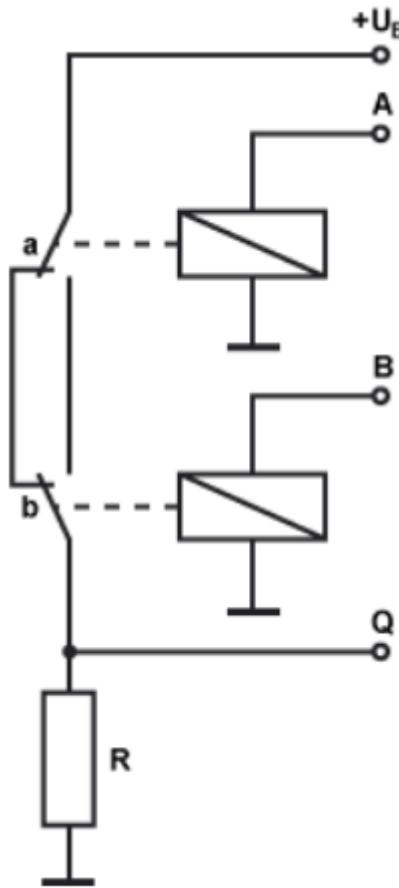
- Die Bezeichnung Äquivalenz bedeutet Gleichwertigkeit.
- Beim XNOR-Gatter mit zwei Eingangsvariablen ist der Zustand der Ausgangsvariable 1, wenn beide Eingangsvariablen den gleichen Zustand 0 oder 1 haben.
- Die Funktion kann durch eine Schaltung mit vier NOR-Gattern erreicht werden. Es sind zwei unterschiedliche Schaltzeichen zu finden.
- Nach IEC 60617-12 gilt dargestellte genormte Schaltzeichen des XOR-Gatters mit negiertem Ausgang.

Logische Verknüpfungen



Der Ausgangszustand eines XNOR-Glieds ergibt 1, wenn eine gerade Zahl der Eingangszustände 1 und alle anderen Eingangszustände 0 aufweisen oder alle 0 sind.

Logische Verknüpfungen





Schaltalgebra

Gesetze der Booleschen Algebra

- Boolesche Algebra verwendet eine Reihe von Gesetzen und Regeln, um den Betrieb einer digitalen Logikschaltung zu definieren
- Ebenso wie die Logiksymbole „0“ und „1“ einen digitalen Ein- oder Ausgang darstellen, können wir sie auch als Konstanten für einen permanenten „offenen“ bzw. „geschlossenen“ Stromkreis oder Kontakt verwenden.
- Das Regelwerk oder die Gesetze der Booleschen Algebra Formulierung wurden erfunden, um die Anzahl der Logikgatter zu reduzieren, die zur Durchführung einer bestimmten logischen Operation benötigt werden, was zu einer Liste von Funktionen oder Theoremen führt, die allgemein als die Gesetze der Booleschen Algebra bekannt sind.

Schaltalgebra

- Boolesche Algebra ist die Mathematik, mit der wir digitale Gatter und Schaltungen analysieren.
- Mit diesen „Booleschen Gesetzen“ können wir einen komplexen booleschen Ausdruck reduzieren und vereinfachen, um die Anzahl der benötigten Logikgatter zu reduzieren.
- Boolesche Algebra ist also ein auf Logik basierendes mathematisches System mit eigenen Regeln oder Gesetzen, die zur Definition und Reduzierung boolescher Ausdrücke verwendet werden.
- Die in der Booleschen Algebra verwendeten Variablen haben nur einen von zwei möglichen Werten, eine logische „0“ und eine logische „1“, aber ein Ausdruck kann eine unendliche Anzahl von Variablen haben, die alle einzeln beschriftet sind, um Eingänge zu dem Ausdruck darzustellen, z.B. Variablen A, B, C usw., was uns einen logischen Ausdruck $A + B = C$ gibt, aber jede Variable kann NUR eine 0 oder 1 sein.

Schaltalgebra

Grundgesetze

UND - Verknüpfungen mit einer Konstanten (Konstante: 0)

$$1 \wedge 0 = 0; \quad 0 \wedge 0 = 0$$

UND - Verknüpfungen mit einer Konstanten (Konstante: 1)

$$1 \wedge 1 = 1; \quad 0 \wedge 1 = 0$$

ODER - Verknüpfungen mit einer Konstanten (Konstante: 0)

$$1 \vee 0 = 1; \quad 0 \vee 0 = 0$$

ODER - Verknüpfungen mit einer Konstanten (Konstante: 1)

$$1 \vee 1 = 1; \quad 0 \vee 1 = 1$$

Rechenregeln der Schaltalgebra

a) Kommutativgesetz (Vertauschungsgesetz)

$$X = A \wedge B \wedge C = B \wedge A \wedge C = C \wedge B \wedge A$$

$$X = A \vee B \vee C = B \vee A \vee C = C \vee B \vee A$$

Die Reihenfolge, in der Variable der UND und ODER Verknüpfungen unterzogen werden, ist beliebig. Sie hat keinen Einfluss auf das Ergebnis.

b) Assoziativgesetz (Verbindungsgesetz)

$$X = A \wedge (B \wedge C) = (A \wedge B) \wedge C$$

$$X = A \vee (B \vee C) = (A \vee B) \vee C$$

Die Reihenfolge der Zuordnung der Variable bei der UND und ODER Verknüpfung ist beliebig. Sie hat keinen Einfluss auf das Ergebnis.

Schaltalgebra

c) Distributivgesetz (Verteilungsgesetz)

konjunktive Distributionsgesetz:

$$X = A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

disjunktive Distributionsgesetz:

$$X = A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

De Morgansche Gesetze

- De Morgansche Gesetze haben eine große praktische Bedeutung bei der Auflösung von Ausdrücken, die insgesamt negiert sind.
- Sie werden zur Umrechnung auf NAND oder auf NOR Verknüpfungen benötigt.

Erstes De Morgansches Gesetz:

$$X = \overline{A \wedge B} = \overline{A} \vee \overline{B}$$

Zweites De Morgansche Gesetz:

$$X = \overline{A \vee B} = \overline{A} \wedge \overline{B}$$

Schaltalgebra

Beispiel:

$$P = \overline{R \wedge S} \vee \overline{\overline{R} \wedge S} = \overline{R} \vee \overline{S} \vee \overline{\overline{R}} \vee \overline{S} = \overline{R} \vee \overline{S} \vee R \vee \overline{S}$$

Die Reihenfolge der Variablen kann geändert werden.

$$P = \overline{R} \vee R \vee \overline{S} \vee \overline{S}$$

$$\overline{A} \vee A = 1 \quad A \vee A = A \quad 1 \vee A = 1$$

$$P = 1 \vee \overline{S} = 1$$

Bindungsregel

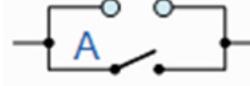
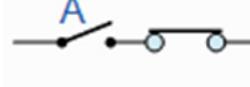
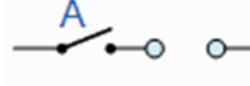
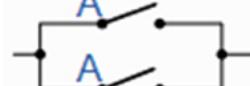
Die Verknüpfung mehrerer Variablen durch UND und ODER kann zu Mehrdeutigkeit führen. Es muß daher eine Priorität festgelegt werden.

Eine UND Verknüpfung bindet stärker als eine ODER Verknüpfung
Es können daher auch Klammern weggelassen werden.

zB: $X = A \vee B \wedge C = A \vee (B \wedge C)$

Schaltalgebra

Wahrheitstabellen für Boolesche Gesetze

Boolescher Ausdruck	Beschreibung	Gleichwertig Schaltkreis	Boolesche Algebra Gesetz oder Regel
$A + 1 = 1$	A parallel zu geschlossen = "geschlossen"		Aufhebung
$A + 0 = A$	A parallel zu offen = "A"		Identität
$A \cdot 1 = A$	A in <u>serie</u> zu geschlossen = "A"		Identität
$A \cdot 0 = 0$	A in <u>serie</u> zu offen = "OFFEN"		Aufhebung
$A + A = A$	A parallel zu A = "A"		Idempotent

Schaltalgebra

Boolescher Ausdruck

$$A \cdot A = A$$

Beschreibung

A in serie zu
A = "A"

Gleichwertig Schaltkreis



Boolsche Algebra
Gesetz oder Regel

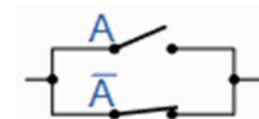
Idempotent

$$\text{NICHT } A = A \text{ NICHT NICHT } A \\ (\text{Doppel negativ}) = "A"$$

$$A + A = 1$$

A parallel zu
NICHT A = "geschlossen"

Doppel Negat



Komplement

$$A \cdot A = 0$$

A in serie zu
NICHT A = "OFFEN"



Komplement

$$A+B = B+A$$

A parallel zu B =
B parallel zu A



Kommutativ

$$A \cdot B = B \cdot A$$

A in serie zu B =
B in serie zu A



Kommutativ

$$A+B = A \cdot B$$

invertieren und ersetzen ODER zu
UND

de Morgan's
Theorem

$$A \cdot B = A+B$$

invertieren und ersetzen UND zu
ODER

de Morgan's
Theorem

Schaltalgebra

- Negation $\neg A$ (“nicht A”)
- Disjunktion $A \vee B$ (“A oder B”)
- B
- Äquivalenz $A \Leftrightarrow B$ (“A ist äquivalent zu B”)

- Konjunktion $A \wedge B$ (“A und B”)
- Implikation $A \Rightarrow B$ (“wenn A, dann B”)

$A : \Leftrightarrow \text{“Es regnet.”}$, $B : \Leftrightarrow \text{“Es ist Montag.”}$,

$\neg B \Leftrightarrow \text{“Es ist nicht Montag.”}$

$(A \wedge B) \Leftrightarrow \text{“Es regnet und es ist Montag.”}$

$(A \vee B) \Leftrightarrow \text{“Es regnet oder es ist Montag”}$

$(A \Rightarrow B) \Leftrightarrow \text{“Immer wenn es regnet, ist Montag.”}$

$(B \Rightarrow A) \Leftrightarrow \text{“Wenn Montag ist, dann regnet es.”}$

**$(B \Rightarrow (1 + 1 = 2))$ ist eine wahre Aussage, da $1 + 1 = 2$ wahr ist
(die Aussage ist also an jedem Wochentag wahr).**



Schaltungssynthese

Schaltungssynthese

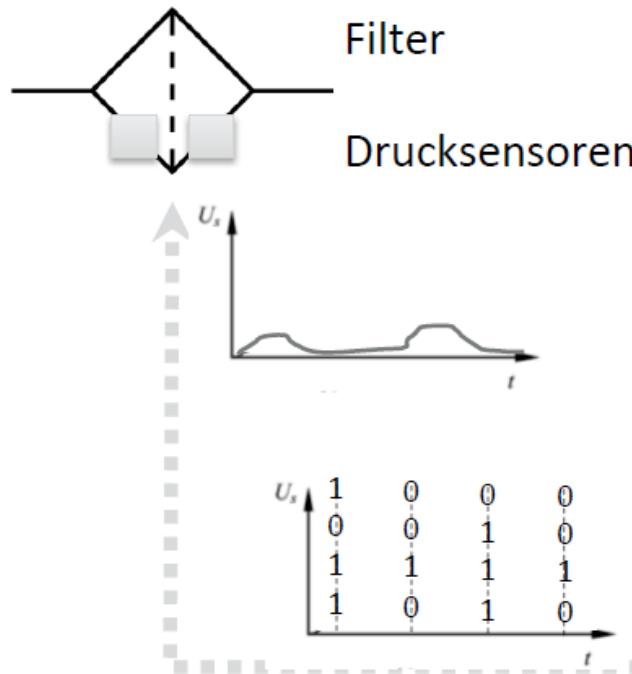
Normalformen

- Normalformen = beliebige **Schaltfunktionen in bestimmter, einheitlicher Form**
- Verwendung von **bestimmten, einfachen Strukturen von Gattern**
 - **Wichtigste Normalformen sind: disjunktive Normalformen** (Disjunktion = ODER)
 - **konjunktive Normalformen** (Konjunktion = UND)
- **Beliebig viele Eingangsvariablen** (Anzahl n = binäre Eingangsvariable) ergeben **eine Ausgangsvariable** unter alleiniger Verwendung von NICHT-, UND-und ODER-Verknüpfungen
- Schaltnetz = **zweistufig**(ohne NICHT)
- Jede binäre Eingangsvariable wird als Dualzahl interpretiert

Wahrheitstabelle - Wertetabelle

- Die möglichen Ausgangszustände eines Logikgatters können in Abhängigkeit von den Eingangszuständen in einer Wahrheitstabelle dargestellt werden.
- Sie listet alle möglichen Kombinationen der Eingangssignale auf und liefert die dazugehörigen Ausgangssignale.
- Aus dieser kann man logische Formeln relativ einfach herauslesen.
- Die einzelnen Zeilen mit denselben Ausgangswerten werden bei der disjunktiven Normalform (1 als Ergebnis) mit *logisch oder* und die einzelnen Eingänge mit *logisch und* verknüpft.
- Bei der konjunktiven Normalform (0 als Ergebnis) ist es umgekehrt.
- Um eine kompakte Formel zu erhalten, kann man ein KV-Diagramm verwenden.

Schaltungssynthese



Verschmutzung $y = 1$, wenn Eingang =

1	1	1	1
1	1	1	1
0	0	1	1
0	1	0	1

Digitale Schaltung



Lampe „brennt“ ($y = 1$),
wenn Filter verschmutzt

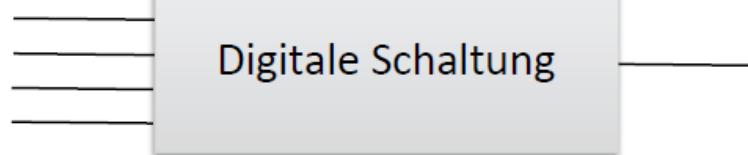
Schaltungssynthese

Beschreibung des Verhaltens der digitalen Schaltung in einer Wahrheitstabelle

Eingänge:
 x_1, x_2, x_3, x_4

Schaltfunktion:
 $y = f(x_1, x_2, x_3, x_4)$

Ausgang: y



Lampe „brennt“ ($y = 1$),
wenn Filter verschmutzt

x_4	x_3	x_2	x_1	y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
..
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Links werden für die Eingangsvariablen alle möglichen Kombinationen aufgelistet und rechts der zugehörige Funktionswert bzw. Wert der Ausgänge.

Schaltungssynthese

Kanonische disjunktive Normalform (KDNF)

- Eine **disjunktive Normalform** (DNF) ist eine **ODER-Verknüpfung von UND-Verknüpfungen**, die eine bestimmte Schaltfunktion realisiert.
- **Annahme:** alle Eingangsvariablen sind in normaler und negierter Form verfügbar.
- Zu einer Schaltfunktion existieren i. a. **mehrere verschiedene DNFs**.
- Eine DNF wird i. a. durch ein **zweistufiges Schaltnetz** (Negierer werden nicht mitgezählt) realisiert.

Schaltungssynthese

- Die kanonische disjunktive Normalform (KDNF) einer Schaltfunktion ist die **Disjunktion**(ODER-Verknüpfung) derjenigen **Minterme** der Funktion, für die die Schaltfunktion den Wert 1 liefert.
- Zu jeder Schaltfunktion gibt es **genau eine KDNF**.
- Als Vollkonjunktion (auch **Minterm** oder Elementarkonjunktion) bezeichnet man in der Aussagenlogik einen speziellen Konjunktionsterm, d. h. eine Anzahl von Literalen (booleschen Variablen), die alle durch ein logisches und (\wedge) verknüpft sind.

Schaltungssynthese

- Jede Formel der Aussagenlogik lässt sich in die disjunktive Normalform umwandeln, da sich auch jede Boolesche Funktion mit einer DNF darstellen lässt.
- Dazu genügt es, die Zeilen ihrer Wahrheitstabelle abzulesen. Für jede Zeile, die als Resultat eine 1 liefert, wird eine Konjunktion gebildet, die alle Variablen der Funktion (der Zeile) verknüpft. Variablen, die in der Zeile mit 1 belegt sind, werden dabei nicht negiert und Variablen, die mit 0 belegt sind, werden negiert.
- Diese Terme werden auch Minterme genannt.
- Durch disjuktive Verknüpfung der Minterme erhält man schließlich die disjunktive Normalform.
- Auf diese Weise erhält man allerdings in der Regel keine minimale Formel, das heißt eine Formel mit möglichst wenig Termen.

Schaltungssynthese

x_2	x_1	x_0	Dezimal	y
0	0	0	0	1
0	0	1	1	0
0	1	0	2	1
0	1	1	3	1
1	0	0	4	0
1	0	1	5	1
1	1	0	6	1
1	1	1	7	0

$$m_0 = \neg x_2 \neg x_1 \neg x_0$$

m_0 = NOT x_2 AND NOT x_1 AND NOT x_0

$$m_2 = \neg x_2 x_1 \neg x_0$$

$$m_3 = \neg x_2 x_1 x_0$$

m sind die sog. **Minterme**, sie enthalten alle Eingangsvariablen und ergeben für diesen Fall $y = 1$

$$m_5 = x_2 \neg x_1 x_0$$

$$m_6 = x_2 x_1 \neg x_0$$

$$y = \neg x_2 \neg x_1 \neg x_0 \vee \neg x_2 x_1 \neg x_0 \vee \neg x_2 x_1 x_0 \vee x_2 \neg x_1 x_0 \vee x_2 x_1 \neg x_0$$

KDNF ist die ODER-Verknupfung (Disjunktionon) aller Minterme, denn die Funktion soll den Wert 1 bekommen, wenn einer der Minterme gleich 1 wird.

Kanonische konjunktive Normalform (KKNF)

- Jede Formel der Aussagenlogik lässt sich in konjunktive Normalform umwandeln, da sich auch jede boolesche Funktion mit einer KNF darstellen lässt.
- Dazu genügt es, die Zeilen ihrer Wahrheitstabelle abzulesen.
- Für jede Zeile, die als Resultat eine 0 liefert, wird eine Klausel gebildet, die alle Variablen der Funktion disjunktiv mit der invertierten Belegung verknüpft.
- Die entstehenden Terme sind Maxterme.
- Deren konjunktive Verknüpfung liefert die kanonische konjunktive Normalform

Schaltungssynthese

- Eine **kanonische konjunktive Normalform** (KKNF) besteht aus paarweise verschiedenen Maxtermen.
- In jedem dieser Maxterme kommt jede Variable genau einmal vor.
- Jede Boolesche Funktion besitzt genau eine KKNF.
- Die KKNF wird auch vollständige **konjunktive Normalform** genannt.
- Als **Volldisjunktion** (auch: **Maxterm**) bezeichnet man in der Aussagenlogik einen speziellen Disjunktionsterm, d. h. eine Anzahl von Literalen, die alle durch ein logisches *Oder* (\vee) verknüpft sind.
- Dabei müssen alle n Variablen der betrachteten n -stelligen Booleschen Funktion im Disjunktionsterm vorkommen, um von einer Volldisjunktion sprechen zu können

Normalformen – Maxterm

- Maxterm auch: **Volldisjunktion**
- Ein Maxterm ist eine **ODER-Verknüpfung** aller (ggf. negierten) Eingangsvariablen einer Schaltfunktion.
- Ein Maxterm gibt bei allen Werten des Eingangsvektors **außer einem** den Wert 1, also **nur in einem einzigen Fall eine 0**.
- In einem Maxterm sind die zu diesem Term gehörigen **Eingangsvariablen**, die den Wert 1 haben, **invertiert**.

Schaltungssynthese

x_2	x_1	x_0	Dezimal	y
0	0	0	0	1
0	0	1	1	0
0	1	0	2	1
0	1	1	3	1
1	0	0	4	0
1	0	1	5	1
1	1	0	6	1
1	1	1	7	0

$$M_1 = x_2 \vee x_1 \vee \neg x_0$$

$$M_1 = x_2 \text{ OR } x_1 \text{ OR NOT } x_0$$

$$M_4 = \neg x_2 \vee x_1 \vee x_0$$

M sind die sog. **Maxterme**, sie enthalten alle Eingangsvariablen und ergeben für diesen Fall $y = 0$

$$M_7 = \neg x_2 \vee \neg x_1 \vee \neg x_0$$

$$y = (x_2 \vee x_1 \vee \neg x_0)(\neg x_2 \vee x_1 \vee x_0)(\neg x_2 \vee \neg x_1 \vee \neg x_0)$$

KKNF ist die UND-Verknüpfung (Konjunktion) aller Maxterme, denn die Funktion darf nur den Wert 0 bekommen, wenn mindestens einer der Maxterme gleich 0 wird

Schaltungssynthese

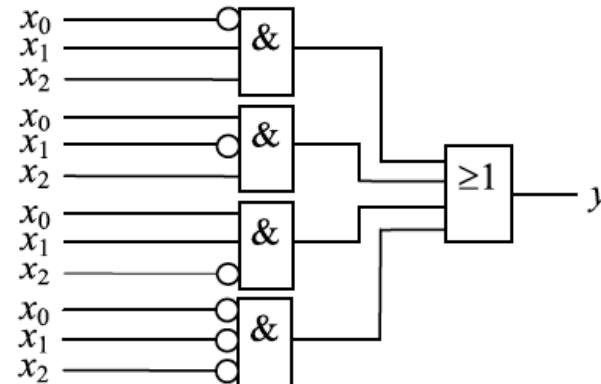
Beispiel für Schaltnetze für KDNF und KKNF

„Gerade Parität“ $y = f_p(x_2, x_1, x_0)$

x_2	x_1	x_0	Dezimal-äquivalent	y
0	0	0	0	1
0	0	1	1	0
0	1	0	2	0
0	1	1	3	1
1	0	0	4	0
1	0	1	5	1
1	1	0	6	1
1	1	1	7	0

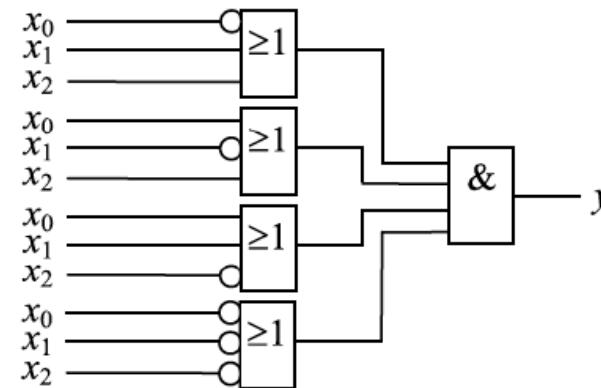
$$y = x_2 x_1 \neg x_0 \vee x_2 \neg x_1 x_0 \vee \neg x_2 x_1 x_0 \vee \neg x_2 \neg x_1 \neg x_0$$

KDNF



$$y = (x_2 \vee x_1 \vee \neg x_0)(x_2 \vee \neg x_1 \vee x_0)(\neg x_2 \vee x_1 \vee x_0)(\neg x_2 \vee \neg x_1 \vee \neg x_0)$$

KKNF



Schaltungssynthese

- Praktischen Nutzen bringen solche Normalformen bei großen Aussagensystemen
 - beispielsweise bei der logischen Beschreibung der Flugzeugelektrik mit 50 Eingabeparametern und Hunderten von Kombinationsmöglichkeiten.
- In einem weiteren Schritt erfolgt eine Vereinfachung des logischen Ausdrucks mittels Karnaugh-Veitch-Diagramm.
- Dabei werden logische Doppelungen entfernt und Überschneidungen berücksichtigt.
- Der letztendlich errechnete logische Ausdruck wird dann in die Steuer-software integriert bzw. hardwaremäßig in der Steuerelektronik umgesetzt.

Minimierung von Schaltfunktionen

Ziel: Darstellung der Schaltfunktion in einer Form, die zu minimalem Aufwand (Kosten) bei der technischen Realisierung (meist in Form einer integrierten elektronischen Schaltung, IC) führt.

Ausgegangen wird von einer **Schaltfunktion**, die **in einer beliebigen Beschreibungsform** gegeben ist.

Unterschiedliche **Minimierungsziele** üblich, z. B.

- minimale Gatterzahl
- minimale Anzahl von Eingängen der Gatter
- Abbildung auf vorhandene Gatterstrukturen, z. B. UND/ODER/NICHT, NANDs, NORs, Elemente in einer bestimmten Standardzell-Bibliothek,

Minimierung von Schaltfunktionen

Unterschiedliche **Minimierungsziele** üblich, z. B.

- minimale Kosten
- minimale Verlustleistung
- maximale Geschwindigkeit
- ...

Schaltungssynthese

- **Einfachstes Aufwandsmaß:**
Anzahl der notwendigen Verknüpfungsglieder (Gatter) und die Gesamtzahl der erforderlichen Eingänge aller Gatter.
- Minimierung mit Blick auf Grundgatter
- **Im folgenden:**
Minimierung der Zahl der Gatter und ihrer Eingänge.(Bildung des omplements nicht im Fokus)
- **Ergebnisse** sind hier stets zweistufige Gatteranordnungen:
als **disjunktive Normalform** (DNF) oder als **konjunktive Normalform** (KNF)
der Schaltfunktion realisieren.

Minimierung von Schaltfunktionen

- Grafisches Verfahren – KV-Diagramme
- **Grafisches Verfahren** = sehr **anschaulich** und „**selbsterklärend**“.
- In der **Praxis** nahezu **irrelevant**.
- Machen wir **trotzdem**, um die nicht grafischen Verfahren zu verstehen.
- Im folgenden **max. 4-Bit-Eingangsvariabel**
Ergebnis ist immer eine disjunktive oder konjunktive Normalform:
 - disjunktive **Minimalform(DMF)** oder
 - konjunktive **Minimalform(KMF)**.

Schaltungssynthese

- **Zusammenfassen von logisch benachbarten Feldern (HD = 1) zu Blöcken mit Kantenlängen, die 2er-Potenzen sind: 1, 2, 4**
- **Blöcke dürfen sich überlappen.**
- Bei **2er/4er/8er-Feldern entfallen 1/2/3 Variable** pro ehemaligem **Minterm/Maxterm**.
- Aus einer kanonischen Form wird eine Minimalform.
- Evtl. gibt es **mehrere verschiedene Minimalformen** eines Typs.
- Wenn man die Wahl hat, sollte man wegen des stabileren dynamischen Verhaltens überlappende statt nicht überlappende Blöcke wählen.

Schaltungssynthese

Disjunktive Minimalform (DMF):

- Zusammenfassung aller Felder mit „1“ zu Blöcken
- Beschreibung jedes einzelnen Blocks durch UND-Verknüpfung der notwendigen Eingangsvariablen (normal oder invertiert) Schaltfunktion = ODER-Verknüpfung der einzelnen UND-Verknüpfungen.
- Felder mit don't care-Symbolen („X“) dürfen zu Blöcken als „1“-Feld verwendet werden, falls sinnvoll -ansonsten als „0“.

Schaltungssynthese

Disjunktive Minimalform (DMF) -Regeln:

- Benachbarte Felder mit „1“ = Blöcke
- Alle „1“ müssen in Gruppen zusammengefasst werden.
- Benachbarte Felder mit Einsen werden zu Blöcke zusammengefasst.
- Blöcke müssen so groß wie möglich sein.
- So wenig Blöcke wie möglich
- Die Blöcke dürfen nur Größen haben, die Zweierpotenzen entsprechen.

Schaltungssynthese

- Die Blöcke müssen rechteckige Blöcke sein.
- Die Blöcke dürfen sich überlappen.
- Die Blöcke dürfen über die Ränder hinweggehen.
- Zwei Blöcke dürfen nicht exakt die gleichen Einsen umfassen.
- Es darf keine Gruppe vollständig von einer anderen Gruppe umschlossen werden.

Schaltungssynthese

KV-Diagramm...

für 2 Variablen:

	e_2	$\neg e_2$
e_1		
$\neg e_1$		

$$(\neg e_1 \wedge \neg e_2)$$

...für 3 Variablen:

	e_3	$\neg e_3$	
$\neg e_1$			$\neg e_2$
e_1			$\neg e_2$
e_1			e_2
$\neg e_1$			e_2

$$(e_1 \wedge e_2 \wedge \neg e_3)$$

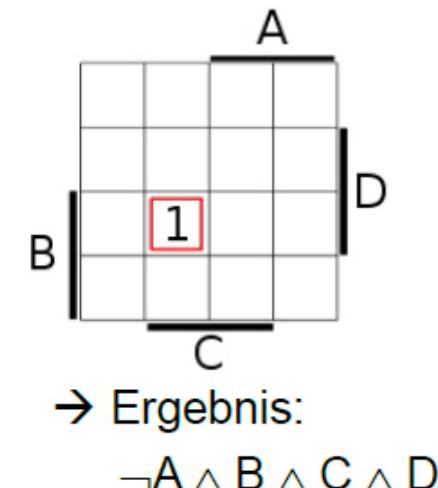
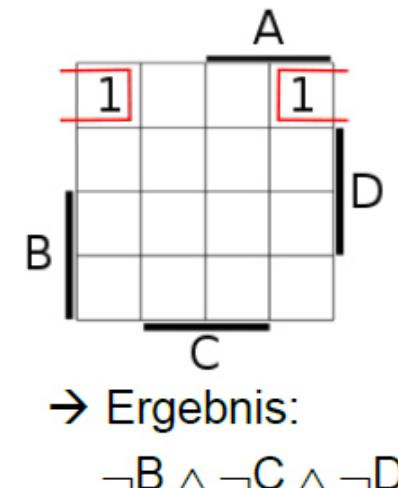
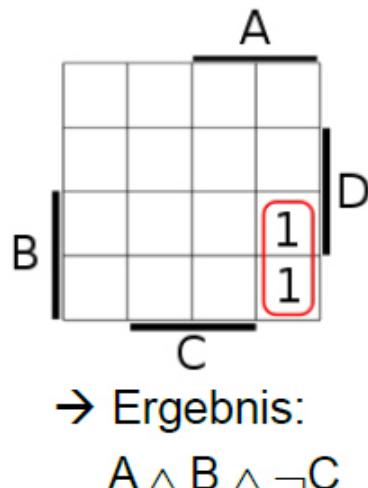
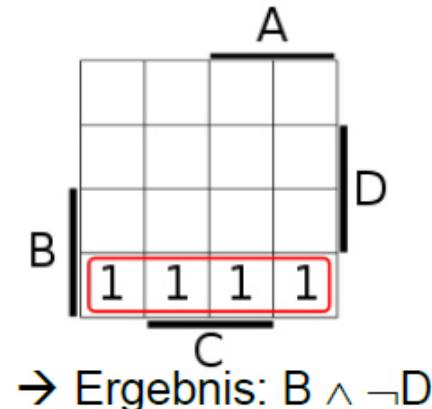
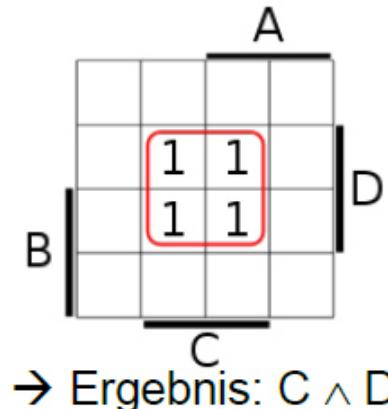
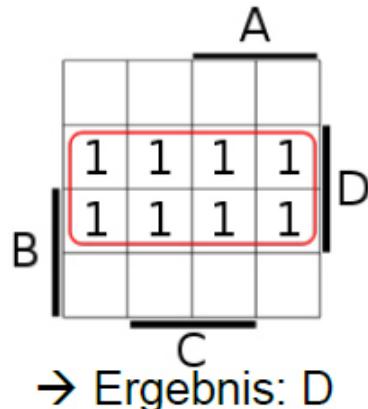
...für 4 Variablen:

	$\neg e_3$	e_3	e_3	$\neg e_3$		
$\neg e_1$					$\neg e_2$	
e_1					$\neg e_2$	
e_1					e_2	
$\neg e_1$					e_2	

$$(e_1 \wedge e_2 \wedge e_3 \wedge \neg e_4)$$

Schaltungssynthese

Disjunktive Minimalform (DMF) – Beispiele mit Eingangsvariablen ABCD:



Schaltungssynthese

Konjunktive Minimalform (DMF):

Zusammenfassung aller Felder mit „0“ zu Blöcken

Dann **zwei Varianten**:

1. Jeder einzelne Block wird wie bei DMF durch eine UND-Verknüpfung der Eingangsvariablen beschrieben.
- Komplementierte Schaltfunktion ergibt sich aus der ODER-Verknüpfung der einzelnen UND-Verknüpfungen.

Konjunktive Minimalform (DMF):

2. Für die Schaltfunktion selbst = Komplementierung beider Seiten
 - Jeder einzelne Block wird durch eine ODER-Verknüpfung der komplementierten zugehörigen Eingangsvariablen beschrieben.
 - Schaltfunktion = UND-Verknüpfung der einzelnen ODER-Verknüpfungen.

Felder mit don'tcare-Symbolen („X“) dürfen zu Blöcken als „0“-Feld verwendet werden, falls sinnvoll - ansonsten als „1“.



Minimierung mit Hilfe der Schaltalgebra

Schaltungssynthese

Der Shannonsche Satz:

Invertierung eines Ausdrucks durch Invertierung aller Variablen und Ersetzung der Operationen durch ihre dualen Operationen.

Für eine beliebige booleschen Funktion $y = f(x_0, x_1, \dots, x_n, \wedge, \vee, \leftrightarrow, \neg, 1, 0)$ gilt

$$\neg y = f(\neg x_0, \neg x_1, \dots, \neg x_n, \vee, \wedge, \leftrightarrow, \neg, 0, 1).$$

Beispiel: $y = (x_2 \vee x_1 \vee \neg x_0)(x_2 \vee \neg x_1 \vee x_0)$



$$\neg y = \neg x_2 \neg x_1 x_0 \vee \neg x_2 x_1 \neg x_0$$

- Ersetze \wedge durch \vee
- Ersetze \vee durch \wedge
- Ersetze x durch $\neg x$
- Erhalte $\neg y$

Schaltungssynthese

1. Aufstellung der booleschen Gleichungen über die KKNF oder KDNF – je nachdem, welche den geringeren Umfang (= geringere Komplexität des Schaltnetzes) hat. Bezuglich des Aufwandes an Gattern sind beide Formen für eine Realisierung aber nicht ideal.
2. Minimierung
3. Verschiedene Ansätze

Es gilt Folgendes: $x_0 x_1 \vee x_0 \neg x_1$

$$= x_0(x_1 \vee \neg x_1)$$

$$= x_0 \wedge 1$$

$$= x_0$$

$$x_0 x_1 \vee x_0 \neg x_1 = x_0$$

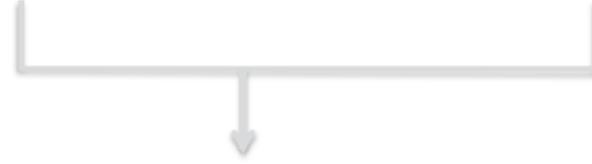
und die duale Regel:

$$(x_0 \vee x_1)(x_0 \vee \neg x_1) = x_0$$

Schaltungssynthese

Beispiel:

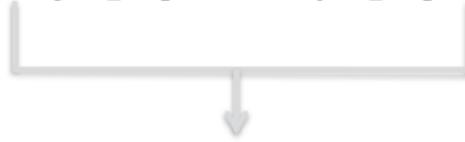
$$y = x_0 \neg x_1 x_2 x_3 \vee x_0 x_1 x_2 x_3 \vee x_0 x_1 \neg x_2 x_3 \vee \neg x_0 x_1 x_2 x_3 \vee \neg x_0 x_1 \neg x_2 x_3$$



$$y = x_0 x_2 x_3 \vee x_0 x_1 x_2 x_3 \vee x_0 x_1 \neg x_2 x_3 \vee \neg x_0 x_1 x_2 x_3 \vee \neg x_0 x_1 \neg x_2 x_3$$



$$y = x_0 x_2 x_3 \vee x_0 x_1 x_3 \vee \neg x_0 x_1 x_3$$



$$y = x_0 x_2 x_3 \vee x_1 x_3$$



Sequenzielle Schaltungen

Sequenzielle Schaltungen

Sequenzielle Schaltung, Schaltwerk:

- Schaltung, deren Ausgänge sowohl von den momentan anliegenden als auch von früheren Eingangsbelegungen abhängen, beispielsweise $y = f(x, y)$
- Wesentliche Elemente einer CPU wie Register, Zähler oder Schieberegister werden durch sequentielle Schaltungen realisiert

Flip-Flop (Basiseinheit einer sequenziellen Schaltung):

- Eine Box mit 2 oder 3 Eingängen und 2 Ausgängen, wobei der 2. Ausgang in der Regel das Komplement des 1. Ausgangs darstellt

Eigenschaften:

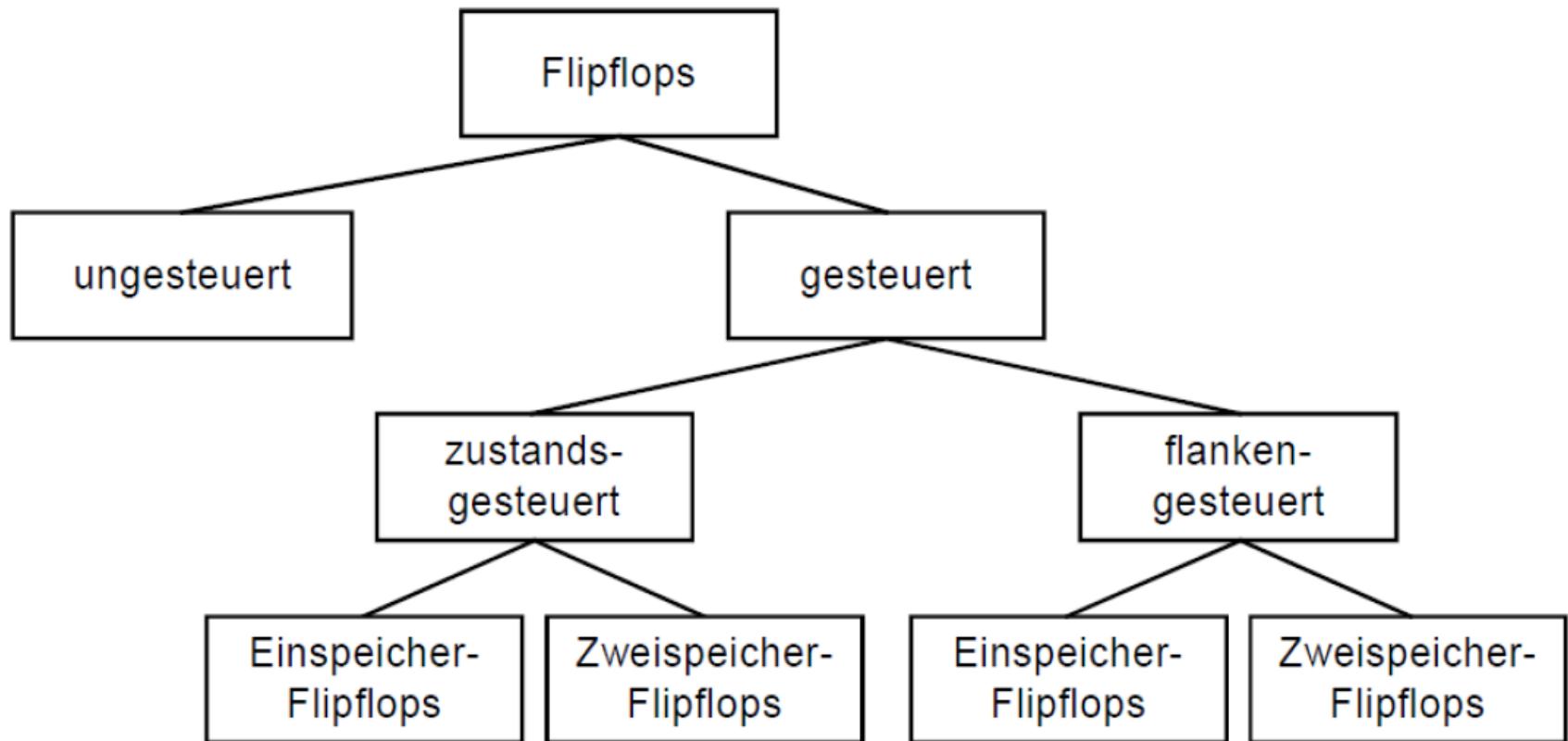
- sind bi-stabil, d. h. verfügen über zwei stabile Zustände (im Gegensatz zum Monoflop mit nur einem stabilen Zustand)
- haben ein „Gedächtnis“, da Ausgang nicht nur abhängig von aktuellen Eingangsbelegungen sondern auch von früheren Eingangsbelegungen
- kleinste Speicherzelle, speichern genau eine Informationseinheit (1 Bit)
- Ausdruck „Flip-Flop“ ist laut nachahmend. Er stammt aus einer Zeit, in der sie durch elektromagnetische Relais realisiert wurden (z.B. Relaisrechner Z3)

Kippstufen – Grundlagen und Klassifikation

- Kippstufendienen u.a. allgemein als **Speicher für Schaltwerke**.
- **Flipflops**, auch **bistabile Kippstufen** genannt, sind Schaltungen, deren Ausgangsgröße Q zwei stabile Zustände (1 = gesetzt und 0 = rückgesetzt) annehmen kann.
- Flipflops können also **ein Bit** Information **speichern**.
- Zusätzlich zum Ausgang Q ist bei Flipflops oft auch der invertierte Ausgang $\neg Q$ verfügbar.

Kippstufen – Grundlagen und Klassifikation

Klassifikation



Kippstufen – Grundlagen und Klassifikation

- Ungesteuerte Flipflops, auch Basis-Flipflops genannt, haben **keinen Steuereingang** und werten daher die Informationen an Ihren Dateneingängen ständig aus.
- Sie arbeiten also stets **asynchron**.
- Gesteuerte Flipflops **werten** die **Informationen** an Ihren Dateneingängen **nur** dann aus, wenn ein zusätzliches **Steuersignal C**(engl. control) einen bestimmten Wert bzw. eine bestimmte Veränderung des Wertes (Signalflanke) aufweist.
- An den Eingang C wird bei **synchronen** Schaltwerken meistens (aber nicht immer!) ein **Taktsignal**(engl. clock) gelegt.

Kippstufen –Grundlagen und Klassifikation

- Zustandsgesteuerte(engl. pulse triggered) Flipflops **übernehmen die Werte an Ihren Dateneingängen** während des Zeitintervalls, in dem das **Steuersignal C intern den Wert 1 hat** (engl.: Latch).
- Einen Eingang, der auf den Zustand des Eingangssignals reagiert, nennt man **„statisch“**.
- Flankengesteuerte(engl. edgetriggered) Flipflops **übernehmen die Werte an Ihren Dateneingängen** zu den Zeitpunkten, zu denen das Steuersignal C seinen Zustand auf definierte Weise ändert, der so genannten **aktiven Flanke**:
 - 0 nach 1 bei positiv flankengesteuerten (engl. positive edge triggered) Flipflops,
 - 1 nach 0 bei negativ flankengesteuerten (engl. negative edge triggered) Flipflops.
- Einen Eingang, der auf die Änderung des Zustandes des Eingangssignals reagiert, nennt man **„dynamisch“**.

Kippstufen – Grundlagen und Klassifikation

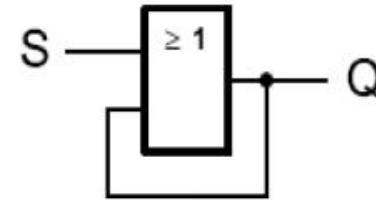
- Zweispeicher-Flipflops, auch Master-Slave-Flipflops genannt, enthalten **intern zwei** hintereinandergeschaltete **Speicherelemente**, die bei komplementären Zuständen bzw. Flanken des Steuersignals ihre Daten übernehmen.
- Hierdurch erscheint die Wirkung der Eingangssignale mit einer Verzögerung am Ausgang (retardierter Ausgang).

Kippstufen – Einspeicher-Flipflops – SR-Basis-Flipflop

Meist auch: RS-Flipflop

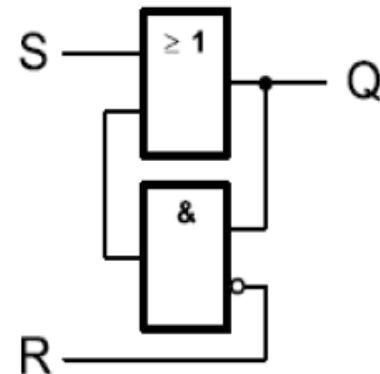
Herleitung:

1. Zunächst $Q = 0$; durch $S = 1$ wird $Q = 1$ und hält diesen Wert durch Rückführung = Speicher
→ dabei ist es egal, ob S wieder Null wird



Selbsthalteschaltung aus einem
rückgekoppelten ODER-Gatter

2. Durch $R = \text{Reset}$ kann mit $S = 0$ ein Zurücksetzen erzwungen werden



Selbsthalteschaltung mit
Erweiterung zum Rücksetzen

- Ein Grundelement aus zwei sich über Kreuz beeinflussenden Logikgattern möge einen Ruhezustand mit $R = S = 0$ haben.
- Mit einem Signal $S = 1$ am „Setz“-Eingang und gleichzeitig $R = 0$ wird der Ausgang Q des Flipflops auf „logisch 1“gesetzt.
- Mit der Zurücknahme dieser Anforderung durch $S = 0$ und gleichzeitig $R = 0$ verharrt das Flipflop infolge derRückkopplung des Ausgangs auf das Eingangsgatter in dem zuvor eingestellten Zustand; er wird also gespeichert.
- Erst wenn der „Rücksetz“-Eingang aktiviert wird mit $R = 1$ bei $S = 0$, wird das Flipflop zurückgesetzt: Am Ausgang entsteht $Q = 0$.
- Wiederum ändert sich mit der Zurücknahme der Anforderung der Zustand nicht.
- Das Ausgangssignal im Falle $R = S = 0$ ist ungewiss, wenn nicht der vorherige Verlauf bekannt ist.

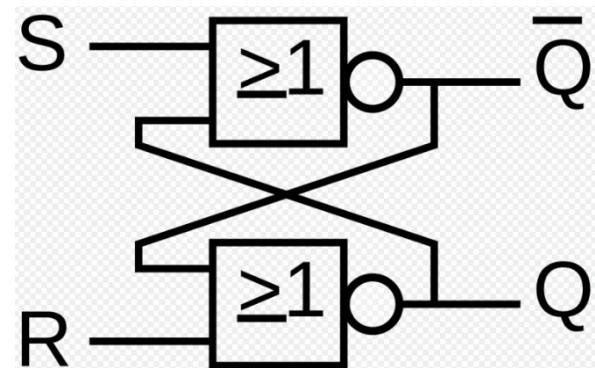
Sequenzielle Schaltungen

Die charakteristische Gleichung lautet (ausgehend von der nebenstehend gezeigten Schaltung mit NOR-Gattern und umgerechnet mit einer der Äquivalenzregeln)

$$Q = \overline{R \vee Q^*} = \overline{R} \wedge \overline{Q^*} = \overline{R} \wedge (S \vee Q)$$

Fall	t_n			t_{n+1}
	R	S	$Q_1 @$	
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	=
8	1	1	1	=

Speicherfälle
Setzfälle
Rücksetzfälle
verbotene Fälle



Sequenzielle Schaltungen

- Liegt am Setzeingang eine 0 und am Rücksetzeingang eine 1 -> wird am Ausgang eine 0 ausgegeben.
- Liegt am Setzeingang eine 1 und am Rücksetzeingang eine 0 -> wird am Ausgang eine 1 ausgegeben.
- Liegt am Setzeingang und am Rücksetzeingang eine 0 -> ändert sich am Ausgang nichts. Es wird der vorherige Zustand des Ausgangs übernommen.
- Liegt am Setzeingang und am Rücksetzeingang eine 1 -> entsteht der sogenannte Verbotene Zustand.
- Der negierte Ausgang nimmt immer das Gegenteil des „normalen“ Ausgangs an.

Fall	R	S	t_n		t_{n+1}
			$Q_1 @ t_n$	Q_1	
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	0	1	1
4	0	1	1	1	1
5	1	0	0	0	0
6	1	0	1	0	0
7	1	1	0	=	=
8	1	1	1	=	=

} Speicherfälle
} Setzfälle
} Rücksetzfälle
} verbotene Fälle

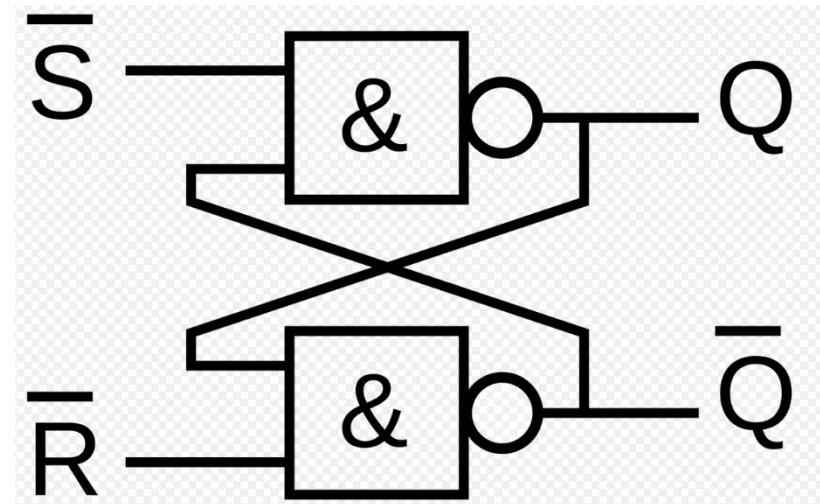
Widerspruchsverhalten

- Kritisch ist der Fall des Widerspruchs, wenn „Setzen“ und „Rücksetzen“ gleichzeitig angefordert werden mit $R = S = 1$ bzw. $R = S = 0$.
- Für diese Eingangsbelegung ist die Schaltung streng genommen kein RS-Flipflop.
- Dieser in sich widersprüchliche (deshalb oft als „verboten“ bezeichnete) Zustand führt dazu, dass beim RS-Flipflop aus NOR-Gattern an den *beiden* Ausgängen Q und $\neg Q$ eine 0 entsteht, dagegen beim RS-Flipflop aus NAND-Gattern an Q und $\neg Q$ eine 1.

Bei industriellen Steuerungssystemen sind Vorkehrungen zu treffen für den Fall, dass bei Betriebsstörungen der Widerspruch auftritt.

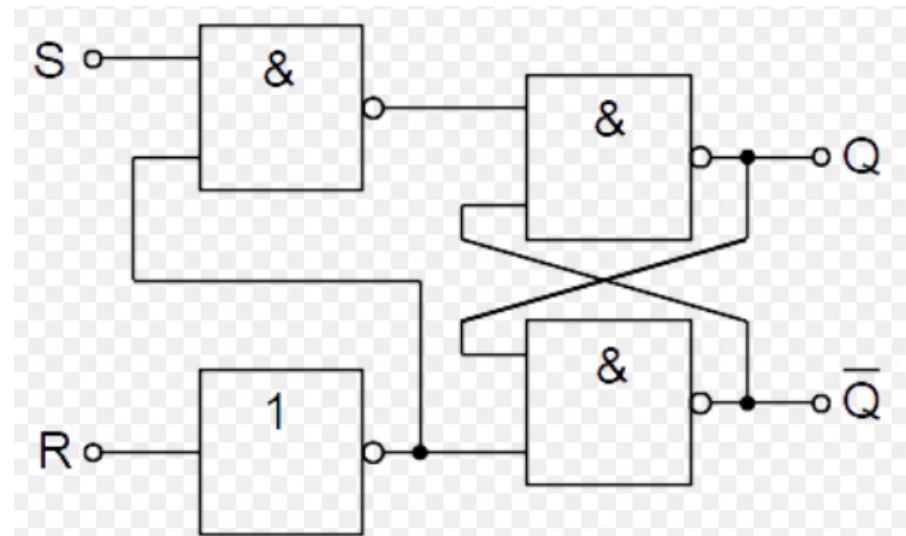
Dazu muss bedacht werden, welcher der beiden Anforderungen „Setzen“ und „Rücksetzen“ das System in einen sicheren Zustand führt, wer also Vorrang oder Dominanz haben soll.

Das Flipflop aus NAND-Gattern hat mit $Q = 1$ Setzvorrang. Das Flipflop aus NOR-Gattern hat mit $Q = 0$ Rücksetzvorrang.

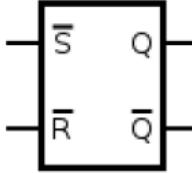
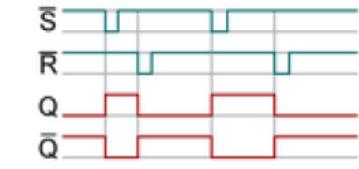
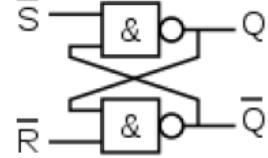
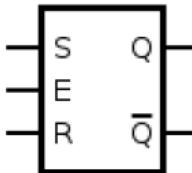
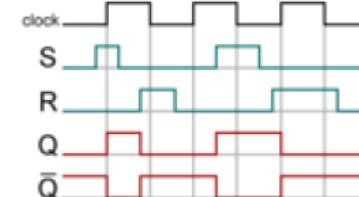
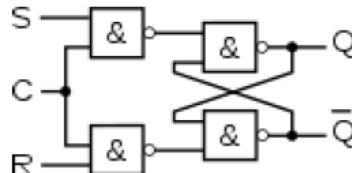


Sequenzielle Schaltungen

Eine Schaltung, die bei Widerspruch einen Vorrang realisiert ohne den Fehler, dass an den beiden Ausgängen Q und \bar{Q} gleiche Signale entstehen, zeigt das neben stehende Bild.

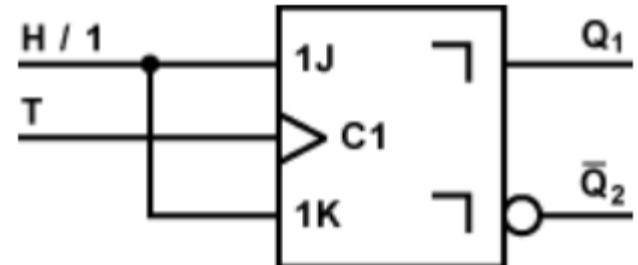


Sequenzielle Schaltungen

Name und Schaltzeichen	Signal-Zeit-Diagramm	Schaltplan	Funktionsabelle																								
Asynchrones RS-Flipflop 	Standardverhalten bei Ausstattung mit negierten Eingängen 	Logik-Schaltung eines RS-Flipflops aus zwei NAND-Gattern 	<table border="1"> <thead> <tr> <th>\bar{S}</th> <th>\bar{R}</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>0 oder 1 (je nach früherem Verlauf)</td> </tr> <tr> <td>0</td> <td>1</td> <td>1 (gesetzt)</td> </tr> <tr> <td>1</td> <td>0</td> <td>0 (zurückgesetzt)</td> </tr> <tr> <td>0</td> <td>0</td> <td>$Q=\bar{Q}=1$ (Fehler: widersprüchliche Eingabe; hier hat Q Setzvorrang)</td> </tr> </tbody> </table> <p>Die Eingänge \bar{S} und \bar{R} führen ihr (aktives) Setzen bzw. Rücksetzen mit 0 aus.</p>	\bar{S}	\bar{R}	Q	1	1	0 oder 1 (je nach früherem Verlauf)	0	1	1 (gesetzt)	1	0	0 (zurückgesetzt)	0	0	$Q=\bar{Q}=1$ (Fehler: widersprüchliche Eingabe; hier hat Q Setzvorrang)									
\bar{S}	\bar{R}	Q																									
1	1	0 oder 1 (je nach früherem Verlauf)																									
0	1	1 (gesetzt)																									
1	0	0 (zurückgesetzt)																									
0	0	$Q=\bar{Q}=1$ (Fehler: widersprüchliche Eingabe; hier hat Q Setzvorrang)																									
RS-Flipflop mit Taktpiegelsteuerung 	Verhalten mit Freigabe von R und S durch 1-Pegel an E bzw. C mit clock 	Logik-Schaltung eines getakteten RS-Flipflops aus vier NAND-Gattern 	<table border="1"> <thead> <tr> <th>C</th> <th>S</th> <th>R</th> <th>Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>X</td> <td>X</td> <td>unverändert</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>unverändert</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0 (zurückgesetzt)</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1 (gesetzt)</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>$Q=\bar{Q}=1$ (Widerspruch)</td> </tr> </tbody> </table> <p>X: beliebig (0 oder 1)</p>	C	S	R	Q	0	X	X	unverändert	1	0	0	unverändert	1	0	1	0 (zurückgesetzt)	1	1	0	1 (gesetzt)	1	1	1	$Q=\bar{Q}=1$ (Widerspruch)
C	S	R	Q																								
0	X	X	unverändert																								
1	0	0	unverändert																								
1	0	1	0 (zurückgesetzt)																								
1	1	0	1 (gesetzt)																								
1	1	1	$Q=\bar{Q}=1$ (Widerspruch)																								

T-Flip-Flop / Toggle-Flip-Flop

- Ein T-Flip-Flop wechselt mit jedem Taktimpuls seinen Ausgangszustand.
- Wobei das T nicht für Takt, sondern für Toggeln oder Toggle steht.
- Verbindet man die Eingänge eines JK-MS-Flip-Flop mit H-Pegel, so erhält man ein T-Flip-Flop. Es hat nur den Takteingang.
- Eine andere Variante ist das D-Flip-Flop bei dem man den negierten Ausgang Q mit dem Eingang D verbindet.
- Vergleicht man die Frequenzen von Eingangs- und Ausgangssignal, so ergibt sich eine Halbierung der Frequenz des Ausgangssignals.
- Damit eignet sich das T-Flip-Flop als Frequenzteiler

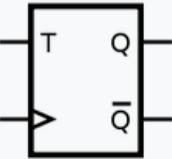
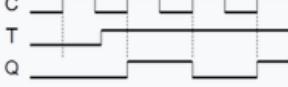


Synchrones T-Flipflop

- Das synchrone T-Flipflop besitzt neben dem dynamischen C-Takteingang einen T-Eingang.
- T steht dabei für *toggle* – hin- und herschalten.
- Es zeigt ein Wechselverhalten synchron zur aktiven Flanke immer dann und nur dann, wenn $T = 1$ ist.
- Er kann aus einem flankengesteuerten JK-Flipflop gebildet werden, indem J-und K-Eingang verbunden werden und gemeinsam als T-Eingang fungieren.

Sequenzielle Schaltungen

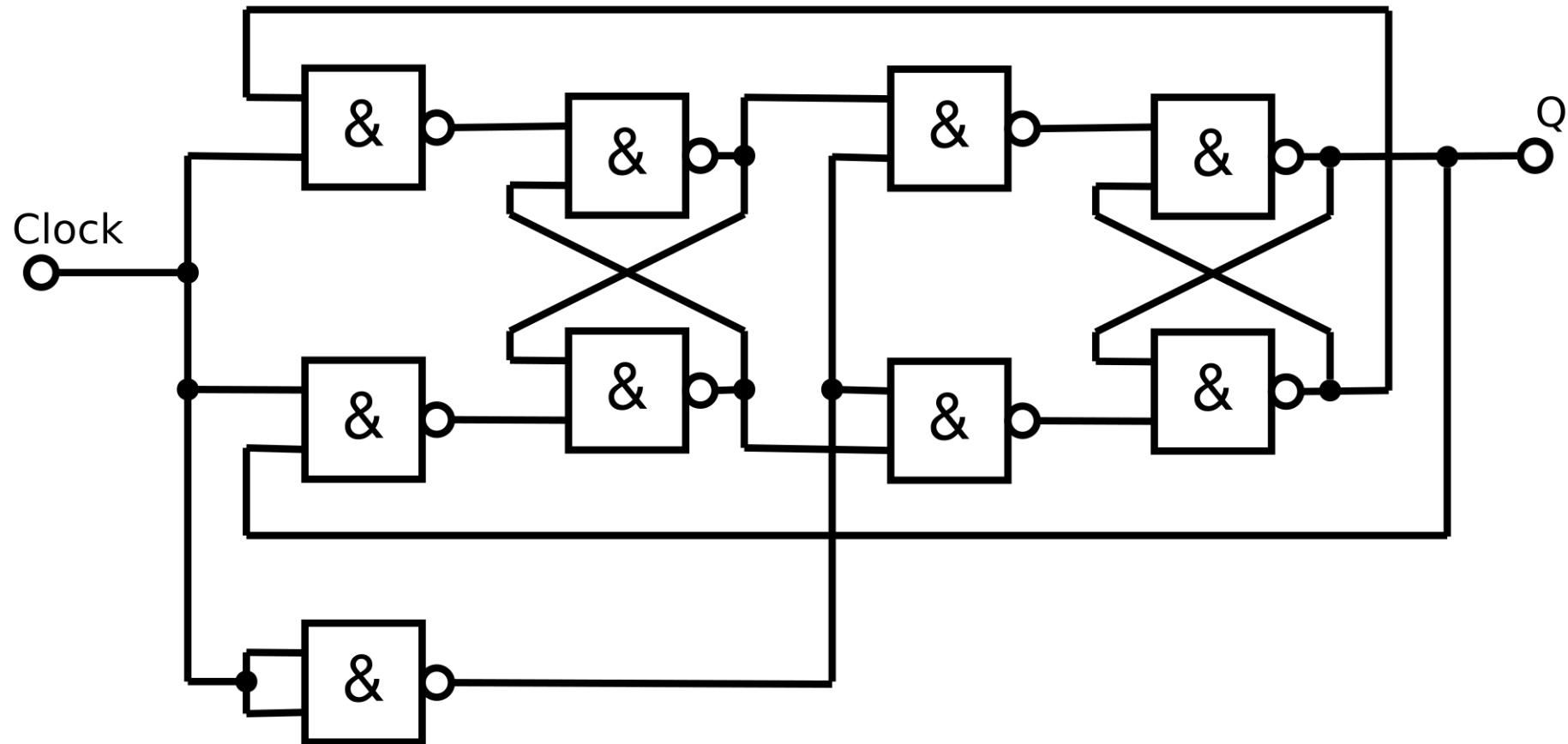
- Das synchrone T-Flipflop wird beispielsweise in Synchronzählern verwendet.
- Sein Verhalten wird durch die angegebene Tabelle beschrieben.
- Darin bedeutet Q_n den Zustand des Flipflops am Ausgang Q nach der n-ten aktiven Taktflanke.

Name und Schaltzeichen	Signal-Zeit-Diagramm	Schaltplan	Funktionstabelle										
Synchrones T-Flipflop 		Wie flankengesteuertes JK-Flipflop mit $J = K = T$	<table border="1"><thead><tr><th>bis zur</th><th>nach der</th></tr><tr><th colspan="2">... n-ten Taktflanke</th></tr><tr><th>T</th><th>Q_n</th></tr></thead><tbody><tr><td>0</td><td>Q_{n-1} (unverändert)</td></tr><tr><td>1</td><td>$\overline{Q_{n-1}}$ (gewechselt)</td></tr></tbody></table>	bis zur	nach der	... n-ten Taktflanke		T	Q_n	0	Q_{n-1} (unverändert)	1	$\overline{Q_{n-1}}$ (gewechselt)
bis zur	nach der												
... n-ten Taktflanke													
T	Q_n												
0	Q_{n-1} (unverändert)												
1	$\overline{Q_{n-1}}$ (gewechselt)												

Asynchrones T-Flipflop

- Wird der T-Eingang fest auf „1“ gelegt, so bekommt der bisherige Takteingang C die Funktion eines Signaleingangs.
- Da keine Anbindung an einen Takt gegeben ist, wird diese Ausführung als asynchrones T-Flipflop bezeichnet.
- Obwohl das Eingangssignal nicht periodisch auftreten muss, wird es teilweise ebenfalls Taktsignal genannt.
- Eine elektromechanische Realisierung eines Toggle-Flipflops ist der Stromstoßschalter.
- Er schaltet mit Hilfe eines von oft mehreren Tastern bei jedem Tastendruck zwischen den Zuständen *Ein* und *Aus* um.

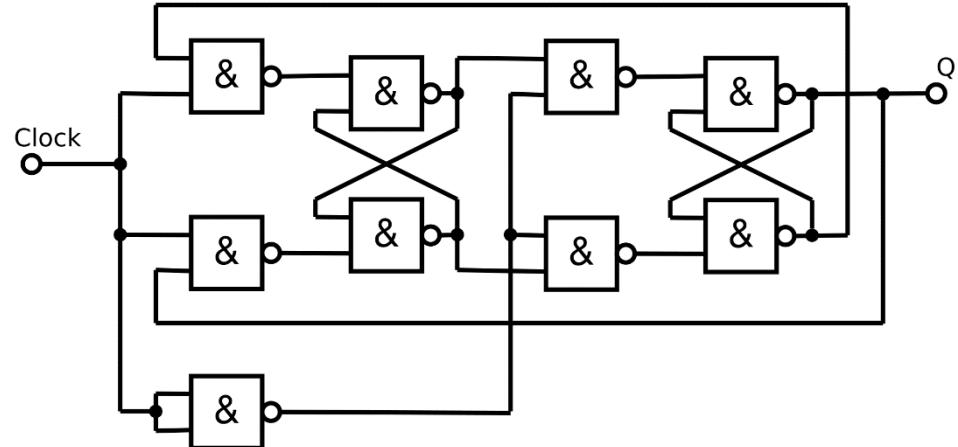
Sequenzielle Schaltungen



- Bei periodischem Eingangssignal erfährt das Ausgangssignal durch das Hin- und Herschalten eine Halbierung der Frequenz (Frequenzteilung durch 2);
- dementsprechend dienen diese Flipflops vor allem als Grundelement in asynchronen binären Zählern und in dezimalen Frequenzteilern und Frequenzzählern.
- Ferner werden sie verwendet, wenn ein Rechtecksignal mit einem Tastgrad von genau 50 % gewonnen werden soll, wenn nur ein unsymmetrisches Signal, aber von doppelter Frequenz, zur Verfügung steht.

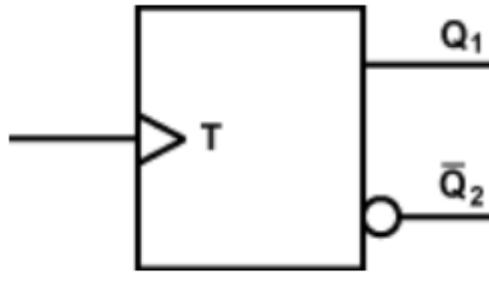
Sequenzielle Schaltungen

- Das asynchrone T-Flipflop kann durch ein taktflankengesteuertes D-Flipflop gebildet werden, wenn dessen Ausgang Q auf den Eingang D zurückgeführt wird.
- Eine Ausführung als Master-Slave-FF zeigt nebenstehendes Bild, in dem während $C = 1$ der Master (linkes RS-FF) mit dem zurückgeführten Ausgangssignal belegt wird;
- so lange ist der Eingang des Slaves (rechtes RS-FF) gesperrt. Mit $C = 0$ wird der Eingang des Masters gesperrt, und der Slave wird mit dem Zustand des Masters belegt.
- Das Signal an Q ändert sich bei jeder fallenden Flanke an C.



Schaltzeichen

Schaltzeichen eines einflankengesteuerten T-Flip-Flop, das bei ansteigender Flanke schaltet.



Stellen wir die Wahrheitstabelle auf:
 $/$ (slash) steht für die Positive Taktflanke
 \backslash (back slash) für die Negative Taktflanke

Wahrheitstabelle		
C	Q	Kommentar
0	0	Annahme
/	1	
\	1	
/	0	
\	0	wieder in der Ausgangssituation

Sequenzielle Schaltungen

Die negativen Taktflanken können entfallen, da das Flipflop nur auf positive Taktflanken regiert.

Wahrheitstabelle		
C	Q	Kommentar
0	0	Annahme
/	1	
/	0	

Was auch entfallen kann, ist die Annahme für den Start:

Wahrheitstabelle		
C	Q	Kommentar
/	1	
/	0	

Die Wahrheitstabelle lässt sich noch weiter vereinfachen.

Dazu führen wir eine Spalte Qn ein. Diese Spalte gibt den letzten Zustand des Flipflops an.

Wahrheitstabelle		
C	Qn	Q
/	0	1
/	1	0

Sequenzielle Schaltungen

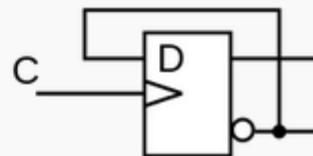
Betrachten wir die Abhängigkeit von Q_n zu Q :

Q_n ist immer das Gegenteil von Q . Wir können also sagen:

$$Q = \overline{Q_n}$$

Oder kurz: Bei jeder Taktflanke ändert sich das Ausgangssignal

Wenn wir jetzt nochmal das Schema betrachten, lässt sich das auch schon darin erkennen.



Rein der Form wegen fügen wir diese Gleichung wieder in die Tabelle ein:

Wahrheitstabelle	
c	Q
/	$\overline{Q_n}$

Die Wahrheitstabelle ist zwar jetzt aufs Minimum gekürzt, aber zugebenermaßen nicht einfacher interpretierbar.

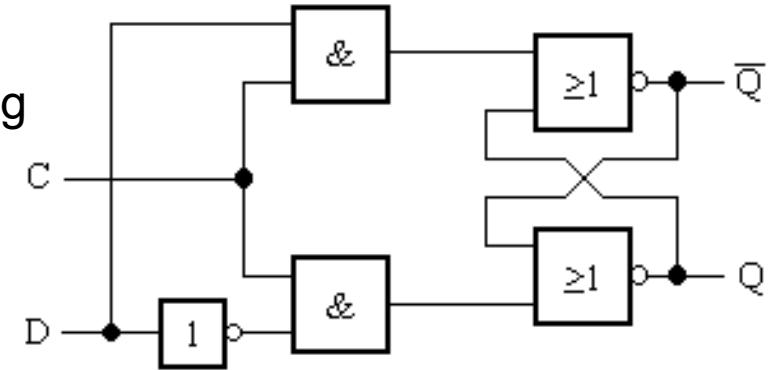
D-Flipflop

Taktflankengesteuertes D-Flipflop

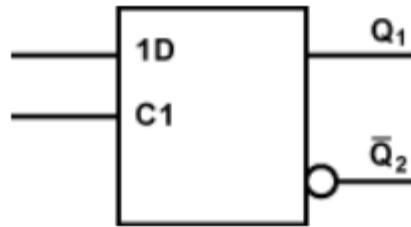
- Das D-Flipflop (abgekürzt für Data- oder Delay-Flipflop) dient zum Verzögern des Signals am Dateneingang bis zur Freigabe synchron zu einer Taktflanke.
- Es besitzt einen Dateneingang D und einen dynamischen Eingang C (Clock), der im Schaltzeichen mit \triangleright gekennzeichnet wird, wenn er auf steigende Flanken reagiert. (Wenn eine fallende Flanke die aktive ist, wird noch ein Negierungszeichen außerhalb der Symbolkontur davorgesetzt.)
- Dieses D-Flipflop realisiert die elementare charakteristische Funktion der taktgesteuerten direkten Übernahme des Dateneingangs zum Ausgang

$$Q' = D$$

- Darin gilt D für den Zustand bis zur Triggerung und Q' nach der Triggerung.
- Bis zur nächsten aktiven Taktflanke wird der aktuelle Zustand gehalten („verzögert“).
- Die gegenläufige Flanke hat keinen Einfluss. Dieses Verhalten führt auf zwei wichtige Anwendungen:
 - *Speicherung* eines Datenbits solange, bis der Takteingang eine neue Speicherung auslöst und
 - *Synchronisierung* paralleler, gleichzeitig begonnener Vorgänge, die je nach Anzahl und Art der durchlaufenden Bauelemente unterschiedlichen Laufzeitverzögerungen unterliegen.



Schaltzeichen

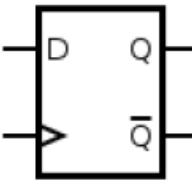
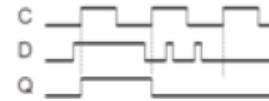


Wahrheitstabelle

E/1D	T/C1	Q_1	Funktion
0	0	n	Speichern
0	1	0	Rücksetzen
1	0	n	Speichern
1	1	1	Setzen

- Immer, wenn am Takteingang eine Null anliegt, wird egal welchen Pegel der Dateneingang hat, der vorhergehende Pegel am Ausgang gespeichert.
- Liegt am Takteingang ein High-Pegel und ein Low-Pegel am Dateneingang, so wird das Flip-Flop zurückgesetzt.
- Liegt am Takteingang ein High-Pegel und ein High-Pegel am Dateneingang, so wird das Flip-Flop gesetzt.

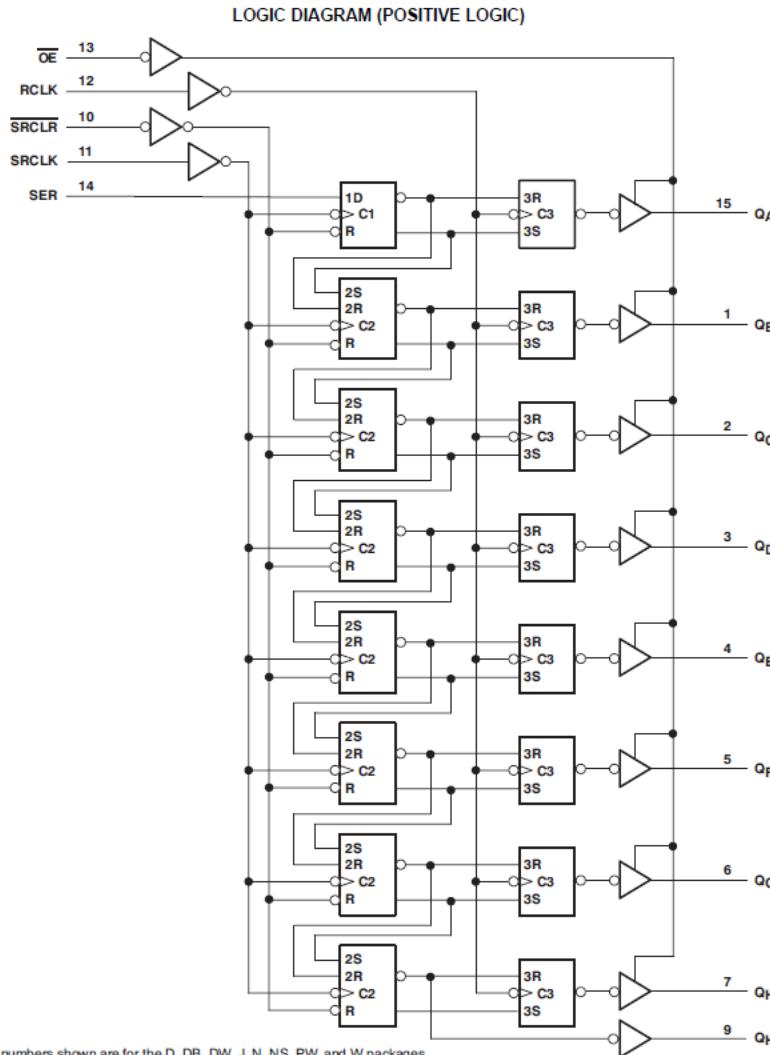
Sequenzielle Schaltungen

Name und Schaltzeichen	Signal-Zeit-Diagramm	Schaltplan	Funktionstabelle															
Flankengesteuertes D-Flipflop 	<p>Übernahme der Eingangsinformation bei steigender Flanke an C</p> 	<p>Wie flankengesteuertes JK-Flipflop mit $J = K = D$</p>	<table border="1"> <tr> <td>C</td> <td>D</td> <td>Q</td> </tr> <tr> <td>↑</td> <td>0</td> <td>0</td> </tr> <tr> <td>↑</td> <td>1</td> <td>1</td> </tr> <tr> <td>0, 1, ↓</td> <td>X</td> <td>unverändert</td> </tr> <tr> <td colspan="3">↑: steigende Flanke ↓: fallende Flanke X: beliebig (0 oder 1)</td> </tr> </table>	C	D	Q	↑	0	0	↑	1	1	0, 1, ↓	X	unverändert	↑: steigende Flanke ↓: fallende Flanke X: beliebig (0 oder 1)		
C	D	Q																
↑	0	0																
↑	1	1																
0, 1, ↓	X	unverändert																
↑: steigende Flanke ↓: fallende Flanke X: beliebig (0 oder 1)																		

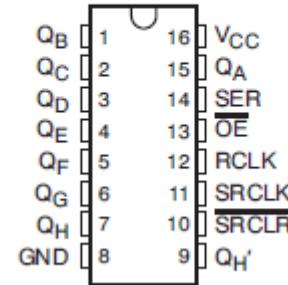
- Mit einem *Clock-Enable*-Eingang CE (im deutschsprachigen Raum „Vorbereitungs-eingang“ V) kann der Takteingang in seiner Funktion freigeschaltet werden (engl. *enable* = freigeben);
- ohne Freigabe bleibt der Zustand bis zu einer späteren Taktflanke unverändert. In dieser Ausstattung wird das Flipflop als **DV-Flipflop** bezeichnet.

- Weil sich alle Änderungen an D, die nach der aktiven Flanke eintreffen, erst zur nächsten aktiven Flanke auswirken, ist es *nichttransparent* und als elementares Flipflop direkt *rückkopplungsfähig*.
- Beispielsweise ist eine Verbindung von Ausgang $\neg Q$ zum Eingang D derselben Kippstufe zulässig, durch die sich das Ausgangssignal in sein Gegenteil ändert, aber immer erst zur nächsten aktiven Taktflanke.
- Damit eignet sich das D-Flipflop als Grundbaustein von Zählschaltungen.
- Durch Zusammenschluss mehrerer solcher Kipplieder und äußere Beschaltung lassen sich – wie mit weiteren taktflankengesteuerten Flipflops – umfangreiche Schaltungen wie Synchronzähler, Frequenzteiler oder Schieberegister realisieren.

Sequenzielle Schaltungen



SN54HC595...J OR W PACKAGE
SN74HC595...D, DB, DW, N, NS, OR PW PACKAGE
(TOP VIEW)



Q0 bis Q7 = Parallel geschaltete Ausgänge

Vcc = Plus-Anschluss der Versorgungsspannung

GND = Minus-Anschluss der Versorgungsspannung

OE= Output Enable zur Aktivierung der Ausgänge Q0 –Q7

DS = Data Signal (serieller Dateneingang)

SHCP = SHift Clock Pin ist der Clock-Eingang zur Übernahme
des Data Signals in das eigentliche Schieberegister (Shift Register).

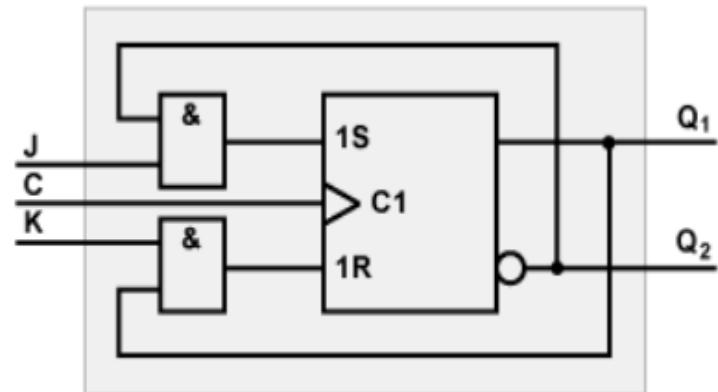
STCP = STore Clock Pin - Der Wechsel von Low auf High kopiert den
Inhalt des Shift Registers in das Ausgaberegister bzw. Speicherregister

MR= Master Reset für die Leerung des Shift Registers

Q7S = Überlauf für eine eventuelle Kaskadierung 2²

JK-Flipflop

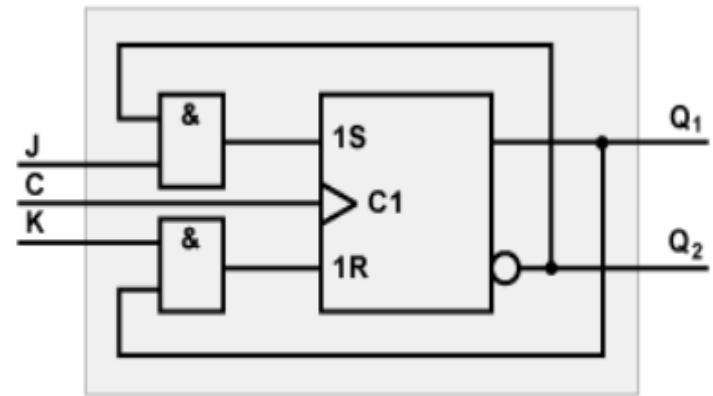
- JK-Flipflops wurden wahrscheinlich nach Jack Kilby benannt; gelegentlich werden sie Jump-/Kill-Flipflops genannt.
- Sie basieren auf dem asynchronen RS-Flipflop, sind aber flankengesteuert oder als Master-Slave-Flipflop ausgeführt.
- Mit dem Taktsignal und der Eingangsbelegung $J = 1$ und $K = 0$ wird am Ausgang eine 1 erzeugt und gespeichert, alternativ bei $K = 1$ und $J = 0$ eine 0.



$$Q' = (J \wedge \overline{Q}) \vee (\overline{K} \wedge Q)$$

Sequenzielle Schaltungen

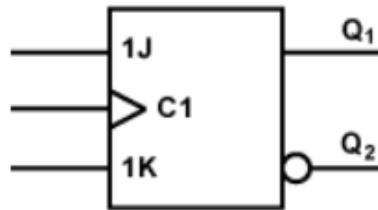
- Der Zustand $J = K = 1$ ist erlaubt; in diesem Fall wechselt der Ausgangspegel mit jeder wirksamen Flanke des Taktsignals.
- Dieses Verhalten lässt die Bezeichnung als Toggle-Flipflop zu. Für $J = K = 0$ bleibt der letzte Ausgangszustand erhalten.
- Die charakteristische Gleichung lautet (mit J, K, Q bis zur Flanke und Q' nach der Flanke).



$$Q' = (J \wedge \overline{Q}) \vee (\overline{K} \wedge Q)$$

- Bei der Realisierung des JK-Flipflops als taktzustandsgesteuertem Master-Slave-Flipflop muss als wesentliche Einschränkung beachtet werden, dass sich in der Transparenzphase des Masters die Zustände der beiden Eingänge J und K *nicht mehr* ändern dürfen.
- Damit liegt kein rein zustandsgesteuertes Flipflop vor.
- Dieser Nachteil ist ein Grund, warum sie als Master-Slave-Flipflops in komplizierteren Digitalschaltungen nur noch selten verwendet werden und durch flankengetriggerte Flipflops ersetzt werden, die diesen Nachteil nicht aufweisen.

Schaltzeichen (taktflankengesteuertes JK-Flip-Flop)



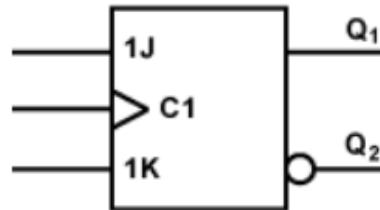
Wahrheitstabelle (taktflankengesteuertes JK-Flip-Flop)

C	K	J	Q ₁	Q ₂	Funktion
0 > 1	0	0	n	n	Speichern
0 > 1	0	1	1	0	Setzen
0 > 1	1	0	0	1	Rücksetzen
0 > 1	1	1	X	X	Wechseln (Toggeln)

n = Pegel abhängig von J und K (0 oder 1)

X = Pegel abhängig vom vorherigen Zustand (0 -> 1 und 1 -> 0)

Schaltzeichen (taktflankengesteuertes JK-Flip-Flop)



Wahrheitstabelle (taktflankengesteuertes JK-Flip-Flop)

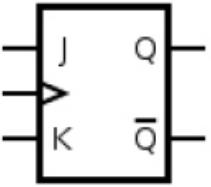
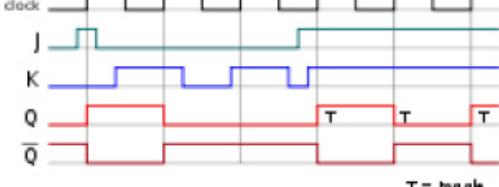
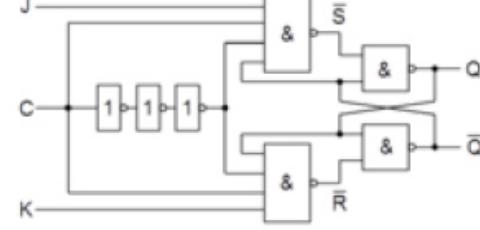
C	K	J	Q ₁	Q ₂	Funktion
0 > 1	0	0	n	n	Speichern
0 > 1	0	1	1	0	Setzen
0 > 1	1	0	0	1	Rücksetzen
0 > 1	1	1	X	X	Wechseln (Toggeln)

n = Pegel abhängig von J und K (0 oder 1)

X = Pegel abhängig vom vorherigen Zustand (0 -> 1 und 1 -> 0)

Sequenzielle Schaltungen

Bei der Realisierung des JK-Flipflops als taktflankengesteuertem Flipflop kann der Eingang C für steigende Flanken(Wechsel von 0 auf 1) oder für fallende Flanken (Wechsel von 1 auf 0) ausgelegt sein.

Name und Schaltzeichen	Signal-Zeit-Diagramm	Schaltplan ^[13]	Funktionstabelle																					
Flanken-gesteuertes JK-Flipflop 	Übernahme der Eingangsinformation durch steigende Flanke an C (clock) 	Nur solange eine an C aufgetretene Flanke \uparrow durch die 3 Nicht-Gatter läuft, kann S = 0 oder R = 0 werden. 	<table border="1"> <thead> <tr> <th colspan="2">bis zur</th> <th>nach der</th> </tr> <tr> <th colspan="2"></th> <th>... n-ten Taktflanke</th> </tr> <tr> <th>J</th> <th>K</th> <th>Q_n</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_{n-1} (unverändert)</td> </tr> <tr> <td>0</td> <td>1</td> <td>0 (zurückgesetzt)</td> </tr> <tr> <td>1</td> <td>0</td> <td>1 (gesetzt)</td> </tr> <tr> <td>1</td> <td>1</td> <td>$\overline{Q_{n-1}}$ (gewechselt)</td> </tr> </tbody> </table>	bis zur		nach der			... n-ten Taktflanke	J	K	Q _n	0	0	Q _{n-1} (unverändert)	0	1	0 (zurückgesetzt)	1	0	1 (gesetzt)	1	1	$\overline{Q_{n-1}}$ (gewechselt)
bis zur		nach der																						
		... n-ten Taktflanke																						
J	K	Q _n																						
0	0	Q _{n-1} (unverändert)																						
0	1	0 (zurückgesetzt)																						
1	0	1 (gesetzt)																						
1	1	$\overline{Q_{n-1}}$ (gewechselt)																						

Synchronzähler

- Ein **Synchronzähler** ist ein elektrisches Bauelement der Digitaltechnik, das eine Folge von Ereignissen zählt.
- Jede dabei entstehende Zahl wird bis zum nächsten Ereignis gespeichert.
- Der Zählerstand wird im einfachsten Fall in Zahlen des Dualsystems dargestellt.
- In diesem Fall sind bei n vorhandenen binären Speicherelementen die möglichen Zahlen auf $[0; 2^n - 1]$ beschränkt.

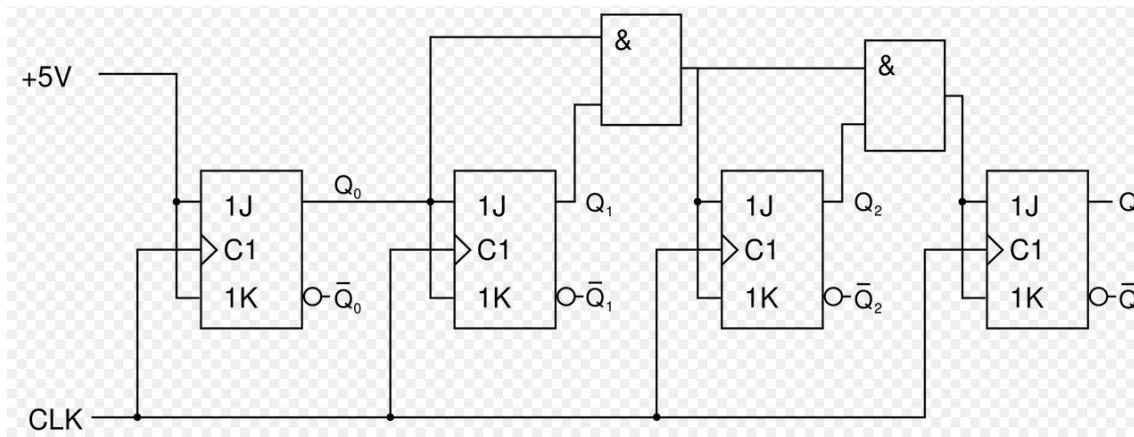
- Man spricht dann auch von *n-bit-Synchronzählern*.
- Durch geeignete Schaltung sind auch Zähler im Dezimalsystem möglich und verbreitet im Einsatz.
- Das zu zählende Eingangssignal wird bei periodischer Folge auch als Taktsignal bezeichnet.

Synchronzähler gehören zu den synchronen Schaltkreisen, weil sich alle Speicherelemente nur zu einer festgelegten (steigenden oder fallenden) Flanke eines gemeinsamen Signals ändern können.

Diese Gleichzeitigkeit über alle Speicherelemente ist dann erforderlich, wenn der Zählerstand von einer elektronischen Einrichtung (z. B. Mikroprozessor) kurz nach dem Ereignis (z. B. nach $\frac{1}{2}$ Periodendauer) übernommen werden soll.

- Ein Synchronzähler kann aus JK-Flipflops aufgebaut sein wie beispielsweise in der folgenden Schaltung.
- Der Triggereingang jedes Flipflops ist hier über den Eingang CLK mit dem Signal des zu zählenden Ereignisses verbunden.
- Diese Struktur lässt alle Flipflops untereinander synchron laufen und gibt der Schaltung den Namen (im Gegensatz zum Asynchronzähler).

4-bit-Synchronzähler aufgebaut aus JK-Flipflops



Sequenzielle Schaltungen

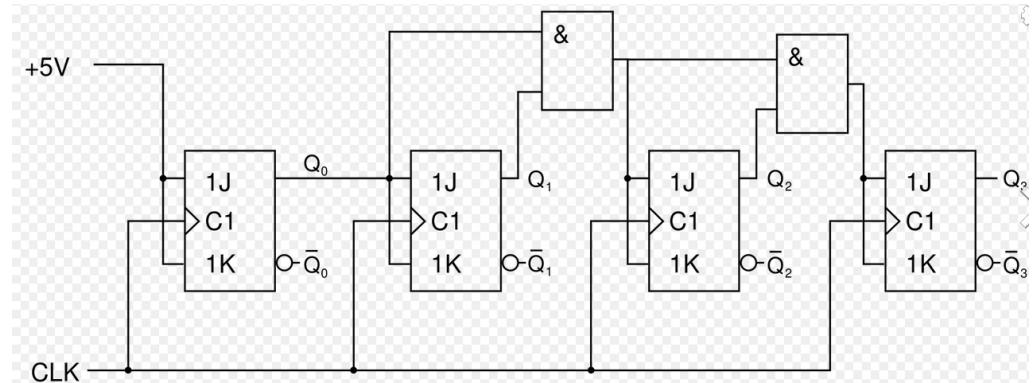
Im Folgenden wird ein *4-bit-Vorwärtszähler* mit den Ausgängen Q_0 bis Q_3 beschrieben, der die Zahlen von 0000_b ($=10_d$) bis 1111_b ($=15_d$) in natürlicher Reihenfolge zählt.

Q_3	Q_2	Q_1	Q_0	Binärzahl	Dezimalzahl
0	0	0	0	0000	0
0	0	0	1	0001	1
0	0	1	0	0010	2
0	0	1	1	0011	3
0	1	0	0	0100	4
0	1	0	1	0101	5
0	1	1	0	0110	6
0	1	1	1	0111	7

Q_3	Q_2	Q_1	Q_0	Binärzahl	Dezimalzahl
1	0	0	0	1000	8
1	0	0	1	1001	9
1	0	1	0	1010	10
1	0	1	1	1011	11
1	1	0	0	1100	12
1	1	0	1	1101	13
1	1	1	0	1110	14
1	1	1	1	1111	15

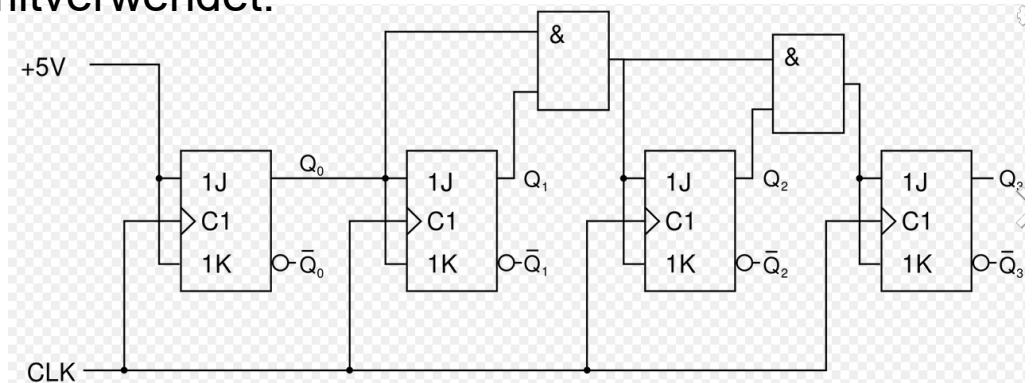
Daraus ergibt sich folgende Verschaltung:

- Das Bit Q_0 mit der niedrigsten Stellenwertigkeit ($2^0=1$) soll bei jeder steigenden Signalflanke wechseln und wird daher direkt mit dem Takteingang CLK verbunden.
- Das Bit Q_1 mit der nächsthöheren Stellenwertigkeit ($2^1=2$) soll nur wechseln, wenn der Ausgang Q_0 den Pegel 1 hat (also während des vorigen Takts eine 1 ausgegeben hat).
- Dies wird realisiert, indem der Ausgang mit den J- und K-Eingängen des zweiten Flipflops verbunden wird.



Sequenzielle Schaltungen

- Das nächste Bit Q_2 mit der nächsthöheren Wertigkeit ($2^2=4$) soll nur wechseln, wenn alle niedrigeren Bits (Q_0 und Q_1) den Pegel 1 haben (also während des vorigen Takt die Binärzahl 11 dargestellt haben).
- Dies wird mit einem Und-Gatter festgestellt.
- Das letzte Bit Q_3 mit der höchsten Wertigkeit ($2^3=8$) soll nur wechseln, wenn alle niedrigeren Bits (Q_0 , Q_1 und Q_2) den Pegel 1 haben (also im vorigen Takt die Binärzahl 111 dargestellt haben).
- Dieses kann mit einem zusätzlichen Und-Gatter realisiert werden, welches das Ergebnis aus dem vorigen Schritt mitverwendet.

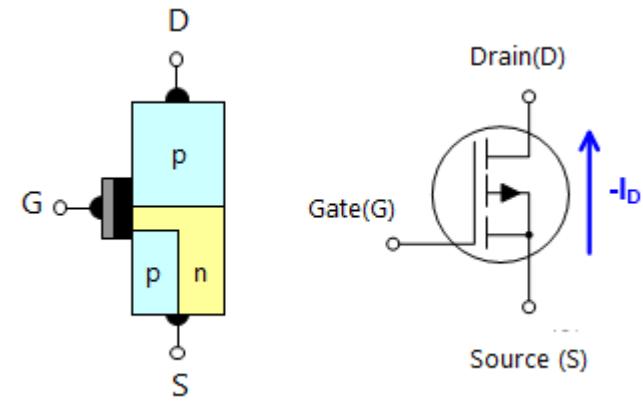




Halbleiter - MOS-Transistoren

Metall-Oxid-Halbleiter-Feldeffekttransistor

- Ein **Metall-Oxid-Halbleiter-Feldeffekttransistor** (englisch *metal-oxide-semiconductor field-effect transistor*, **MOSFET** auch **MOS-FET**, selten **MOST**) ist eine zu den Feldeffekttransistoren mit isoliertem Gate (IGFET) gehörende Bauform eines Transistors.
- In ihrer ursprünglichen und auch heute noch oft verwendeten Form sind sie durch einen Schichtstapel aus einer metallischen Gate-Elektrode, einem Halbleiter und dem dazwischen befindlichem oxidischen Dielektrikum bestimmt.
- Dies stellt eine Metall-Isolator-Halbleiter-Struktur dar, weshalb man verallgemeinert auch von Metall-Isolator-Halbleiter-Feldeffekttransistoren (**MISFET**) sprechen kann, die auch Varianten mit nicht-oxidischen Dielektrika umfassen.



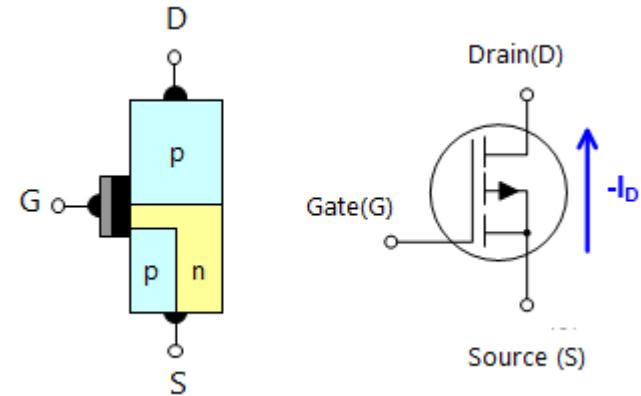
Unterschied: Bipolar Transistor vs. MOSFET

- Im Gegensatz zu bipolaren Transistoren werden MOSFETs **nicht über eine Stromzufuhr gesteuert**, sondern durch das Anlegen einer Spannung.
- Beim Anlegen einer gewissen Mindestspannung beginnt der MOSFET zu leiten.
- Er kann vereinfacht als ein **steuerbarer Schalter** beschrieben werden.
- Für besonders schnelles Schalten eignet sich ein PowerFET.
- Aufgrund ihrer hohen Leistungsfähigkeit werden sie in vielen verschiedenen Bereichen eingesetzt, in denen es darum geht, **elektrische Signale zu verstärken, zu schalten oder zu regeln**.
- MOSFETs werden häufig in Stromversorgungsschaltungen verwendet, um den Stromfluss zu steuern und die Effizienz der Stromversorgung zu verbessern.

Aufbau und Funktionsweise

- Ein MOSFET ist ein aktives Bauelement mit mindestens drei Anschlüssen (Elektroden):

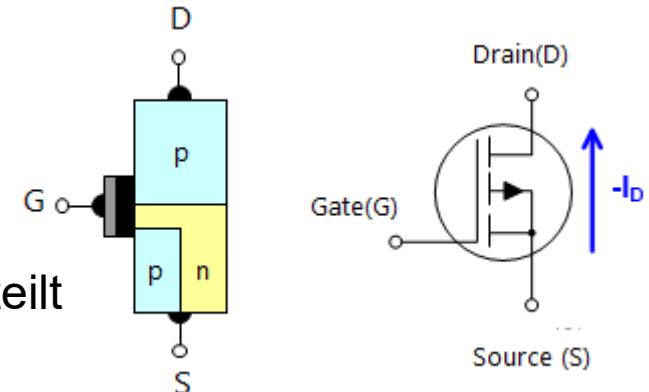
- G (gate, dt. *Steuerelektrode*),
- D (drain, dt. *Abfluss*),
- S (source, dt. *Quelle*).



- Bei einigen Bauformen wird ein zusätzlicher Anschluss B (bulk, Substrat) nach außen geführt, der mit der Chiprückseite verbunden ist.
- Da eine Spannung an der Chiprückseite zusätzliche elektrische Felder erzeugt, die auf den Kanal wirken, verschiebt sich, wenn man die Spannung am B-Anschluss ändert, die Threshold-Spannung des MOSFETs. Meistens ist das Substrat jedoch intern mit dem Source verbunden.

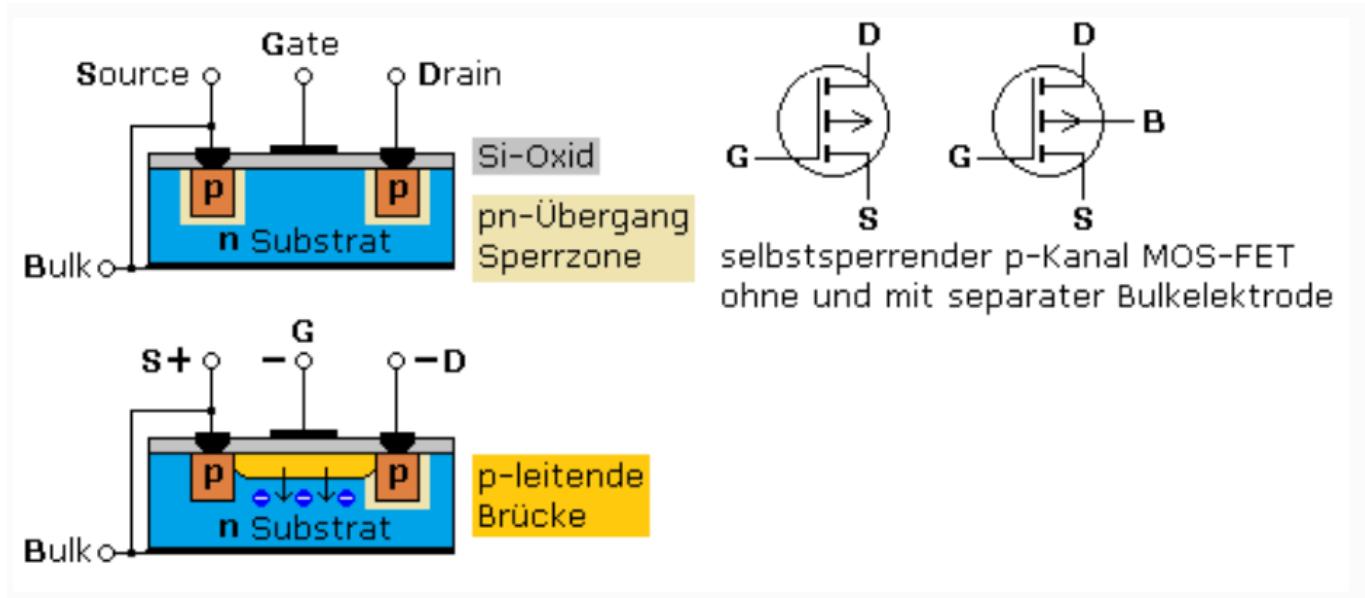
Grundtypen

- Ähnlich wie der Bipolartransistor kann auch der MOSFET in die zwei grundlegenden Varianten p-Typ (auch p-leitend, p-Kanal oder PMOS) und n-Typ (auch n-leitend, n-Kanal oder NMOS) eingeteilt werden.
- Werden, beispielsweise in integrierten Digitalschaltungen, beide Typen gemeinsam verwendet, spricht man von CMOS (engl.: complementary MOS).
- Vom Halbleitermaterial her unterscheidet man zwischen dem p-Kanal MOS-FET und n-Kanal MOS-FET.



- Der Verarmungstyp, englisch depletion, ist wie ein "normaler" FET selbstleitend und fast immer ein n-Kanal-Typ.
- Beide Arten werden für eine Schwellenspannung zwischen 0,8 ... 2 V, charakterisiert als Niedervolttechnik und 2,5 ... 4 V als Hochvolttechnik hergestellt.
- Innerhalb dieses Spannungsbereichs zwischen Gate und Source wird der Kanal gerade abgeschnürt oder gerade geöffnet.
- Die MOS-FET-Technologie hat eine besondere Bedeutung in der monolithischen Großintegration bei der Chipherstellung erlangt.
- Sie gestattet eine hohe Integrationsdichte bei sehr geringer Fehlerrate und einem kleinen Leistungsverbrauch, da die Ansteuerung stromlos erfolgt.

- Von der Bauart abhängig gibt es Anreicherungs- und Verarmungstransistoren.
- Der Anreicherungstyp, englisch enhancement, oder selbstsperrender MOS-FET ist ohne Ansteuerung nicht leitend und aus technologischen Gründen meistens ein p-Kanal-Transistor.

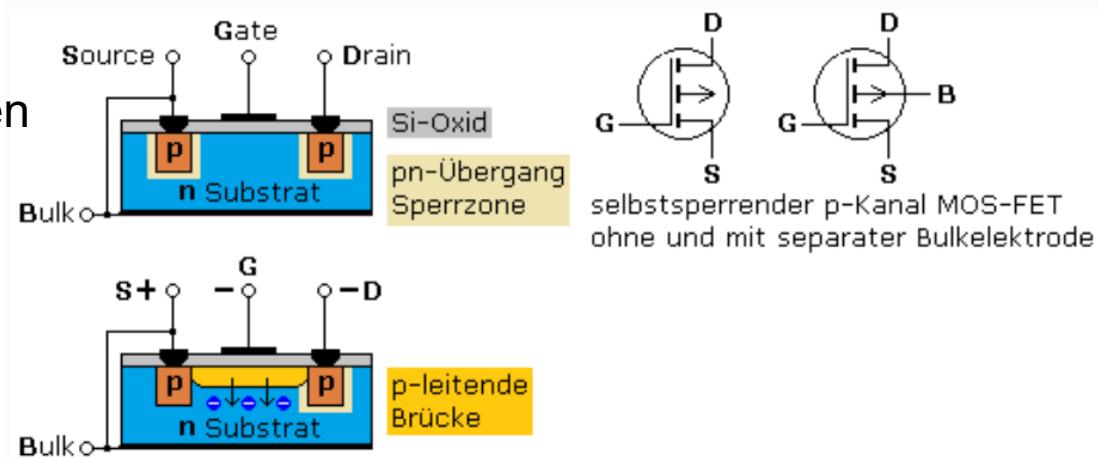


Selbstsperrender MOS-FET

- Beim selbstsperrenden p-Kanal MOS-FET befindet sich zwischen zwei p-dotierten Inseln ein n-dotiertes Substrat.

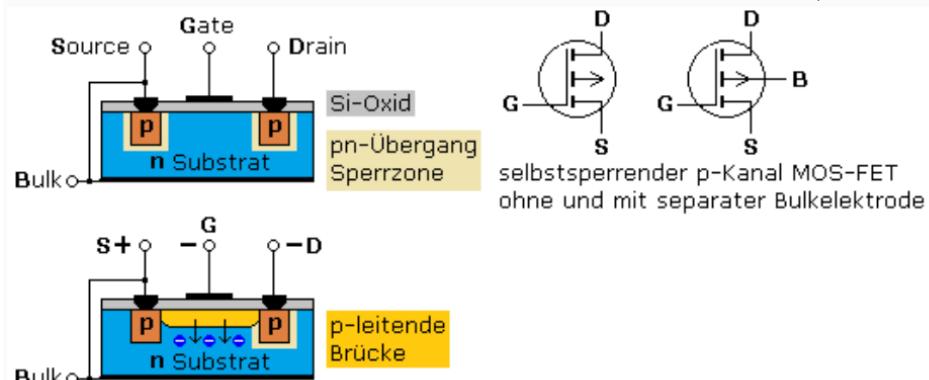
- Die p-Bereiche sind elektrisch leitend mit dem Drain- und Sourceanschluss verbunden.

- Durch Siliziumdioxid isoliert liegt dazwischen die Gate-Elektrode.



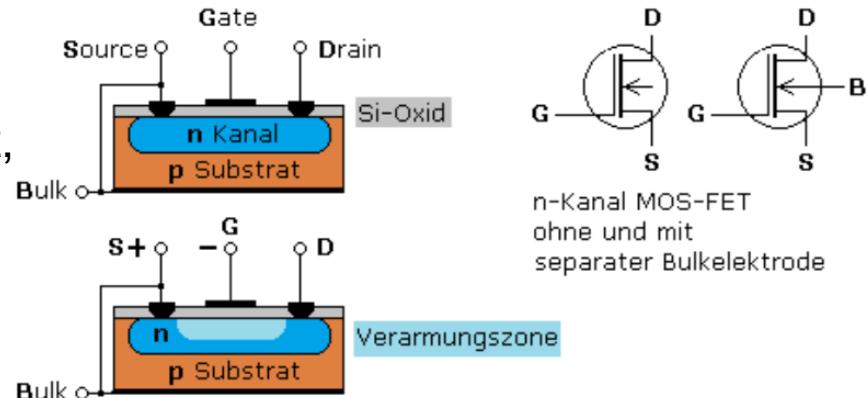
Selbstsperrender MOS-FET

- Die Drain-Source-Strecke bleibt weiterhin gesperrt.
- Eine negative Spannung zwischen Gate und Source oder Gate und Substrat, dem Bulk erzeugt ein elektrisches Feld unterhalb der Isolationsschicht.
- Durch die Feldstärke werden negative Ladungsträger vom Gate weg in das Substrat gedrängt.
- Es entsteht eine p-leitende Brücke zwischen Drain und Source.
- Der Drainstrom ist in diesem Fall ein Löcherstrom (Defektelektronen).



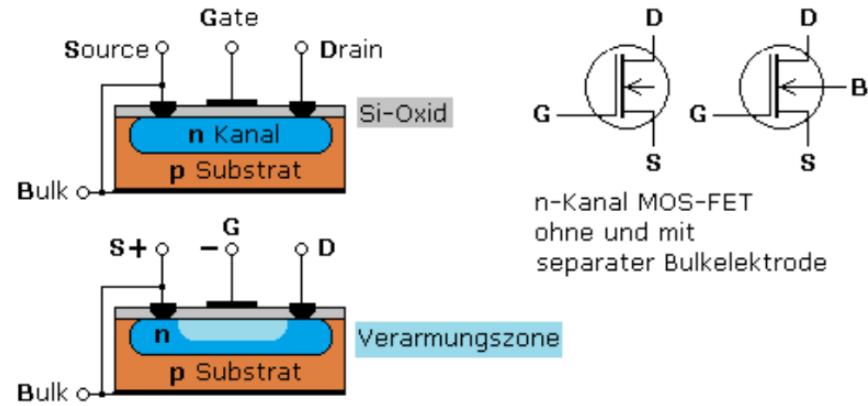
Selbstleitender MOS-FET

- Auf einem p-Trägermaterial, dem Substrat, englisch als Bulk bezeichnet, befindet sich eine schwach n-leitend dotierte Zone.
- Sie ist mit den Elektroden Drain und Source elektrisch leitend verbunden.
- Dazwischen liegt durch die Oxidschicht isoliert die Gateelektrode.
- Manchmal ist das Substrat intern mit Source leitend verbunden, oft ist dafür eine vierte Elektrode herausgeführt.
- Dieser MOS-FET verhält sich in vielen Eigenschaften wie ein FET, allerdings mit isoliertem **Gate** und wird daher auch als IG-FET bezeichnet.



Selbstleitender MOS-FET

- Im Schaltzeichen steht die Gateelektrode vergleichbar mit einer Kondensatorplatte dem Kanal gegenüber.
- Der Kanal veranschaulicht mit seiner durchgezogenen Linie die Selbstleitfähigkeit.
- Bei Ansteuerung entsteht zwischen der Gateelektrode und dem Kanal ein elektrisches Feld, das Elektronen aus dem n-Kanal in das p-Substrat verdrängt.
- Es bildet sich eine Verarmungszone, die den selbstleitenden n-Kanal bis zur Abschnürung verengt und hochohmiger werden lässt.



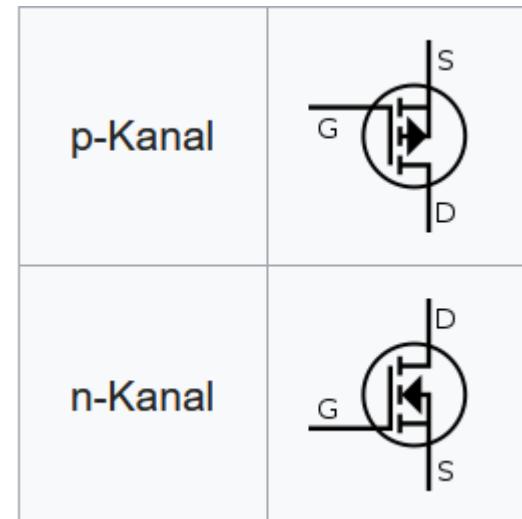
Unterschied N-Kanal / P-Kanal FET

- Bei MOSFETs mit **n**-Typ-Verstärkung schaltet eine positive Gatespannung den **Transistor** ein und bei Null-Gate-Spannung wird der **Transistor** ausgeschaltet.
- Bei einem **MOSFET** mit **p**-Kanal-Verstärkung schaltet eine negative Gate-spannung den **Transistor** ein und bei einer Null-Gate-Spannung ist der **Transistor** ausgeschaltet.
- Beim P-Kanal MOSFET wird der **Source-Pin an die positive Versorgungsspannung** angeschlossen.
- Beim N-Kanal MOSFET ist der Source-Pin an Masse angeschlossen

- **N-Kanal-MOSFETs:** Bei einem N-Kanal-MOSFET besteht der leitende Kanal aus einem N-Typ-Halbleiter, was bedeutet, dass er durch Elektronen leitet.
 - Im „EIN“-Zustand wird der N-Kanal-MOSFET durch eine positive Spannung am Gate aktiviert, was Elektronen anzieht und so den Stromfluss ermöglicht.
- **P-Kanal-MOSFETs:** Im Gegensatz dazu besteht der leitende Kanal eines P-Kanal-MOSFET aus einem P-Typ-Halbleiter, der durch „Löcher“ leitet – also Bereiche, in denen Elektronen fehlen.
 - Ein P-Kanal-MOSFET wird durch eine negative Spannung am Gate aktiviert, welche die notwendigen „Löcher“ erzeugt und den Stromfluss ermöglicht.

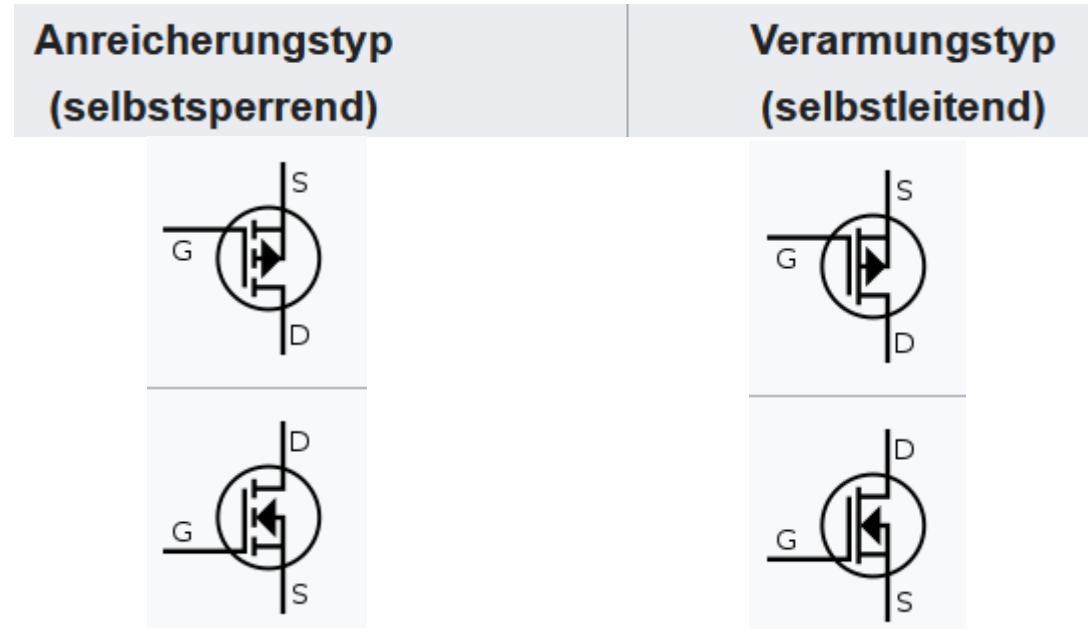
Schaltzeichen

- Als Schaltzeichen werden im deutschsprachigen Raum meist Darstellungen mit den vier Anschlüssen für Gate, Source, Drain und Body/Bulk (mittiger Anschluss mit Pfeil) genutzt.
- Dabei kennzeichnet die Richtung des Pfeils am Body/Bulk-Anschluss die Kanal-Art, das heißt die Majoritätsladungsträgerart.
- Hierbei kennzeichnet ein Pfeil zum Kanal einen n-Kanal- und ein Pfeil weg vom Kanal einen p-Kanal-Transistor.



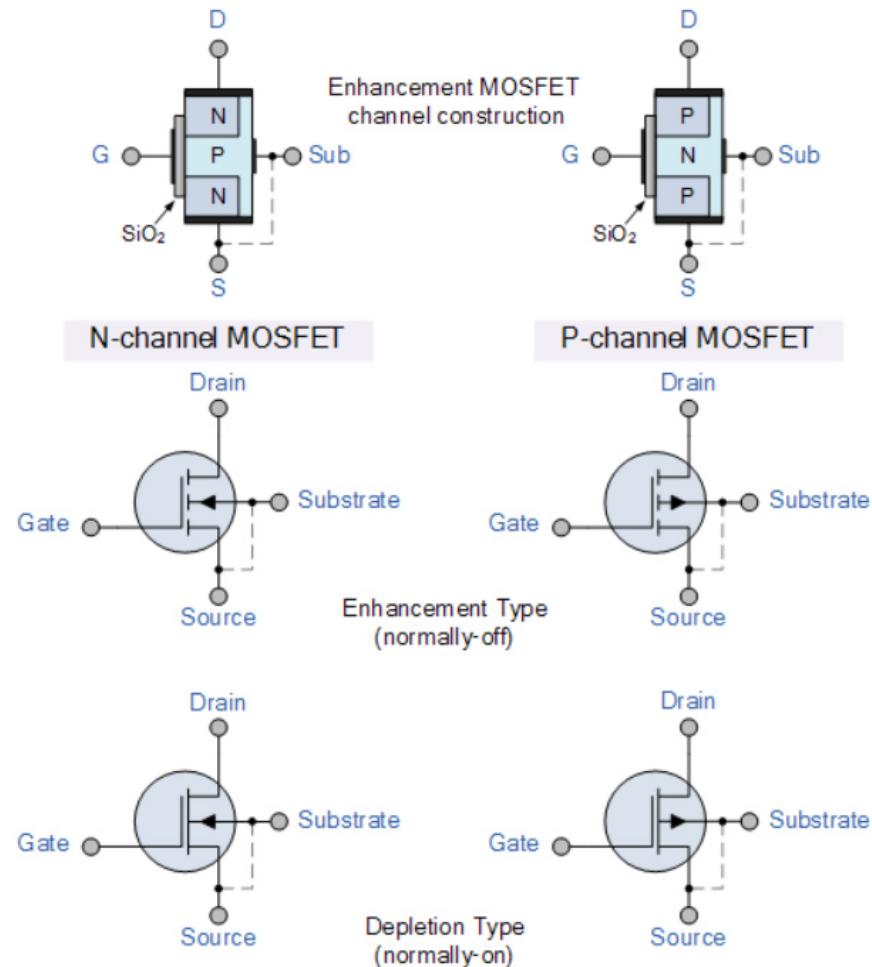
Schaltzeichen

- Ob der Transistor selbstsperrend oder selbstleitend ist, wird wiederum über eine gestrichelte („Kanal muss erst invertiert werden“ → Anreicherungstyp, selbstsperrend)
- bzw. eine durchgängige („Strom kann fließen“ → Verarmungstyp, selbstleitend) Kanallinie dargestellt.

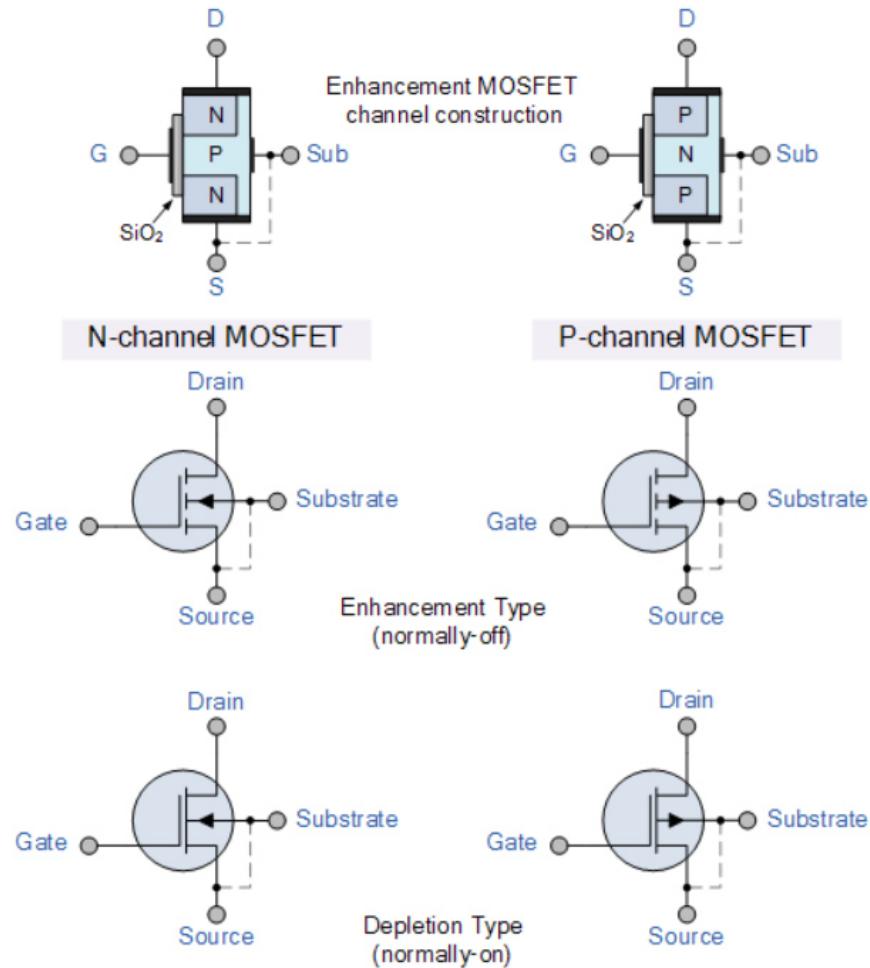


Halbleiter

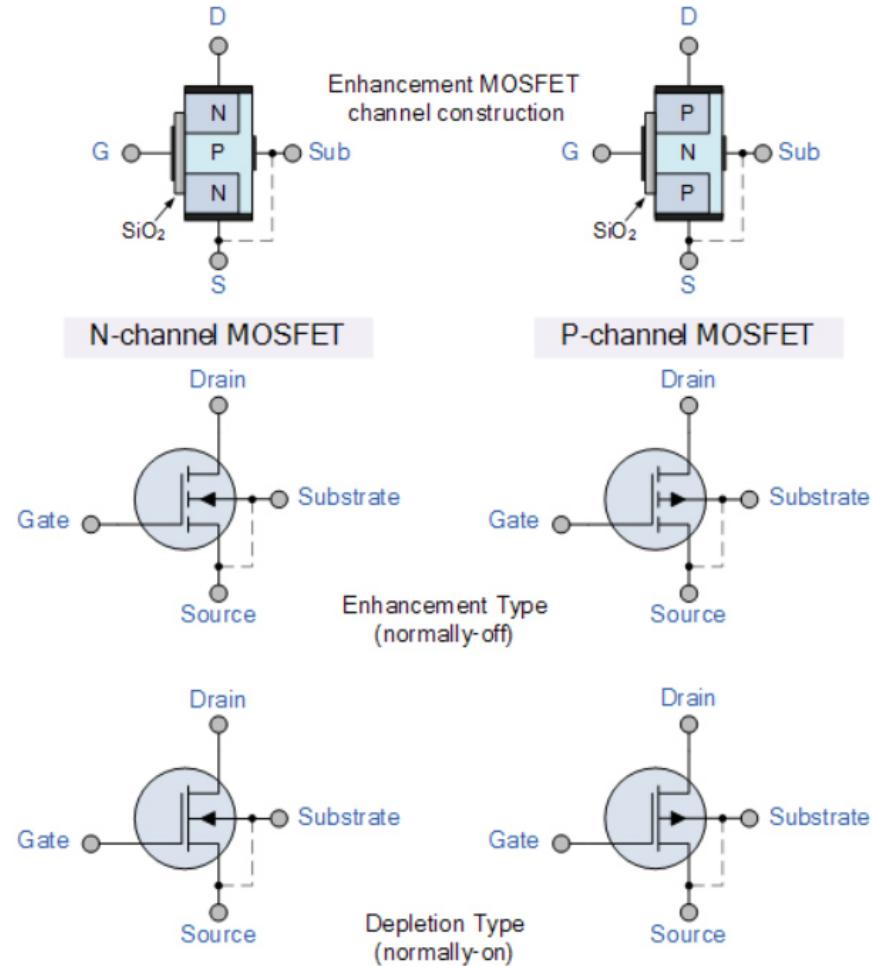
- Die Symbole und der prinzipielle Aufbau für beide MOSFET Konfigurationen sind nachfolgend dargestellt.
- Die vier MOSFET-Symbole oben zeigen einen zusätzlichen Anschluss namens Bulk (Substrate) und werden normalerweise nicht als Eingangs- oder Ausgangsanschluss verwendet, sondern dienen zur Erdung des „Substrate“.
- Es verbindet sich mit dem Haupt-Halbleiterkanal über eine Diodenverbindung mit dem Gehäuse oder der Metallplatte des MOSFETs.



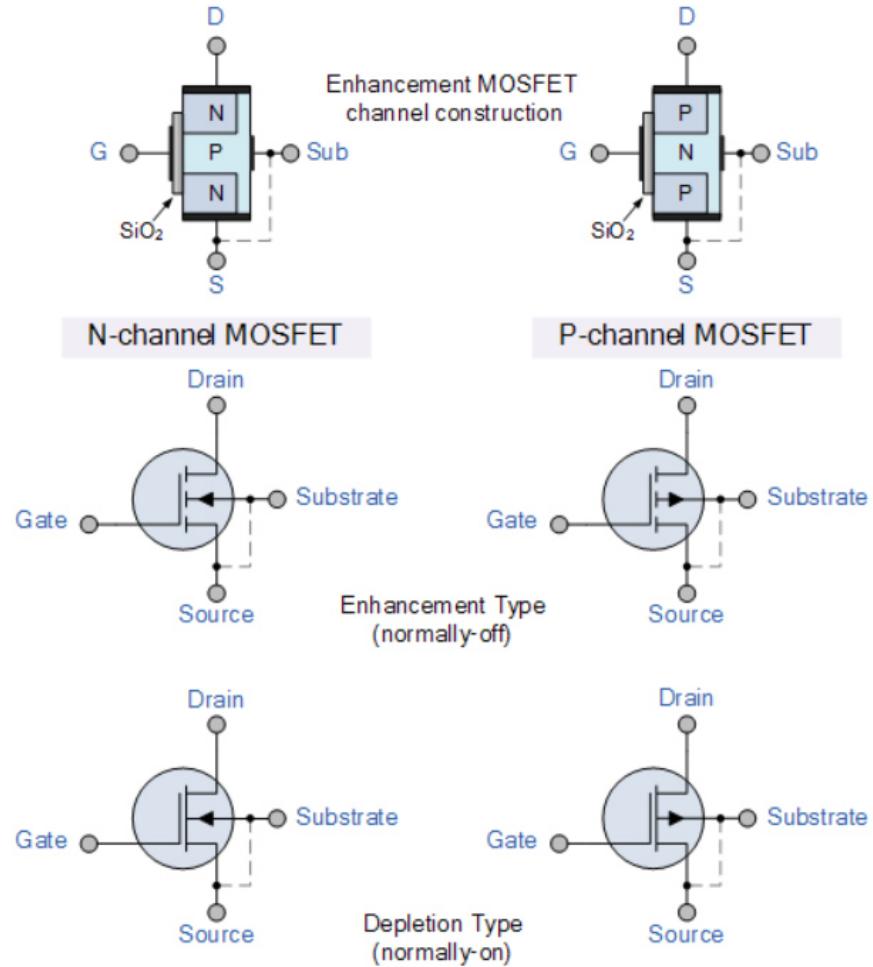
- In diskreten MOSFETs wird dieses Substratkabel normalerweise intern mit dem Quellanschluss verbunden.
- In diesem Fall wird es, wie bei den Erweiterungsarten, aus dem Symbol für die Klärung weggelassen.
- Die Linie zwischen den Anschlüssen Drain und Source stellt den halbleitenden Kanal dar.



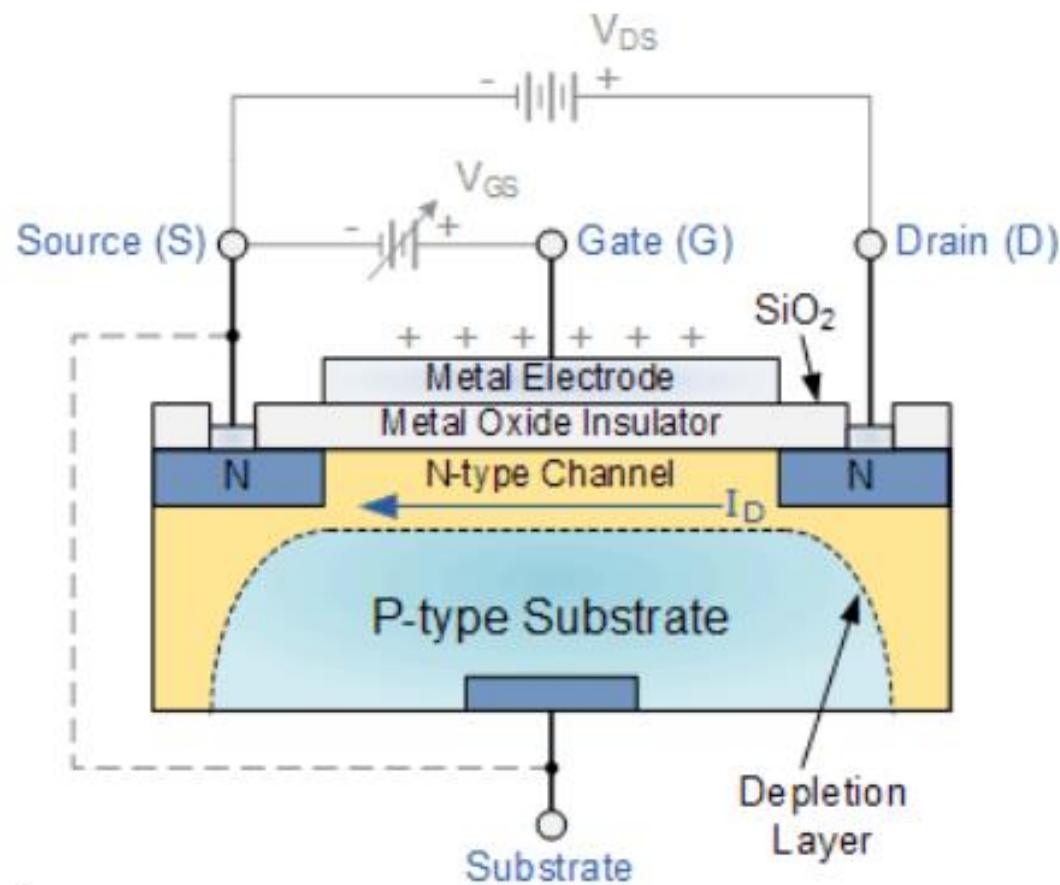
- Handelt es sich um eine durchgehend durchgezogene Linie, so handelt es sich um einen MOSFET vom „Verarmungstyp“ (normal-ON), da der Drainstrom mit Null-Gate-Potential fließen kann.
- Wenn die Kanallinie gestrichelt oder gebrochen dargestellt wird, handelt es sich um einen MOSFET vom „Anreicherungstyp“ (normal-OFF), da der Null-Drain-Strom mit Null-Gate-Potential fließt.



- Die Pfeilrichtung gibt an, ob es sich bei dem leitfähigen Kanal um ein p- oder n-Typ-Halbleiterbauelement handelt



Grundlegende MOSFET-Struktur und -Symbol



Vorteile des FET

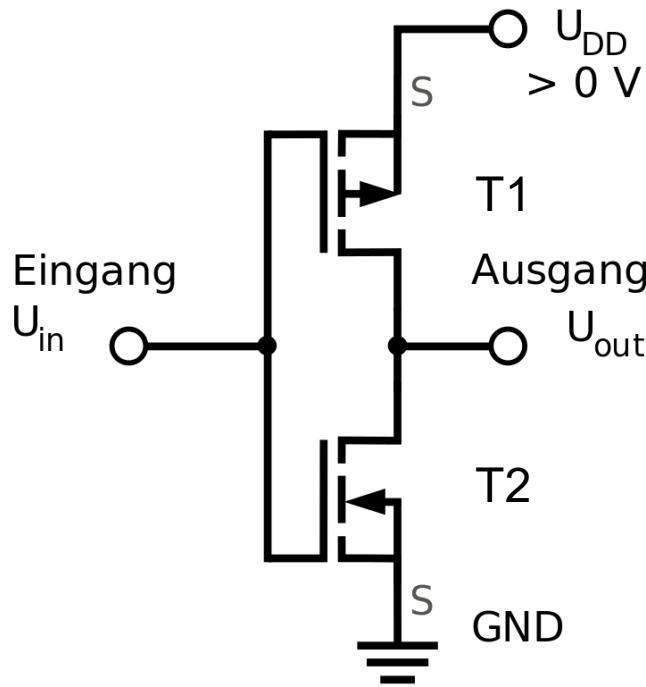
- Niedrigere Verluste als bei Bipolartransistoren.
- Sehr schnelles Schalten möglich, daher für sehr hohe Frequenzen geeignet (keine Speicherzeit wie beim BJT).
- Einfaches Parallelschalten im Schaltbetrieb, da Unterschiede im durch den positiven Temperaturkoeffizienten ausgeglichen werden.
- Leistungslose Ansteuerung im statischen Fall, jedoch hohe Umladeverluste am Gate!
- oft preiswerter als vergleichbare Bipolartransistoren (engl. **Bipolar Junction Transistor, BJT**)

Vorteile des FET

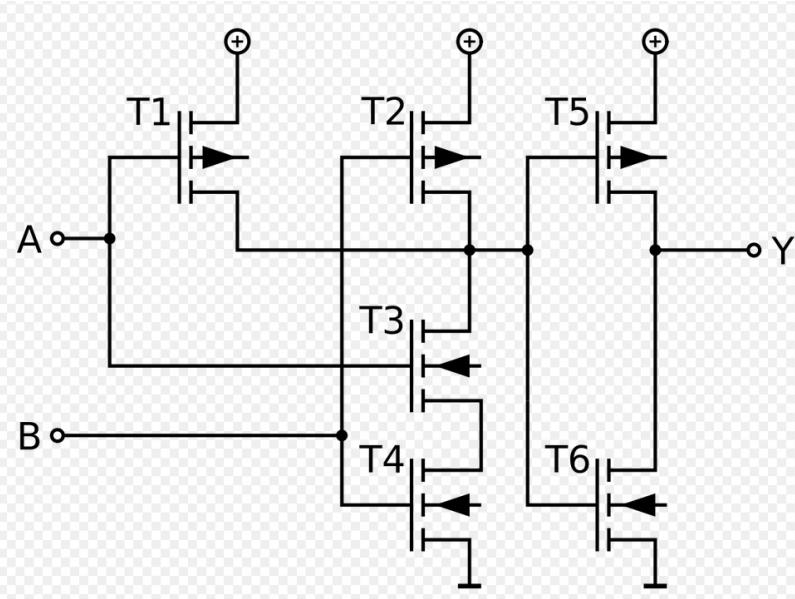
- Relativ unempfindlich gegen Überspannung zwischen Drain und Source.
- Bei Überschreitung der Maximalspannung zwischen Drain und Source findet ein sogenannter "Durchbruch" statt.
- Dies ist vergleichbar mit dem Zener-Effekt. Ist die Energiemenge begrenzt, ist dieser Durchbruch reversibel und der FET wird NICHT zerstört, im Gegensatz zum BJT.

Nachteile des FET

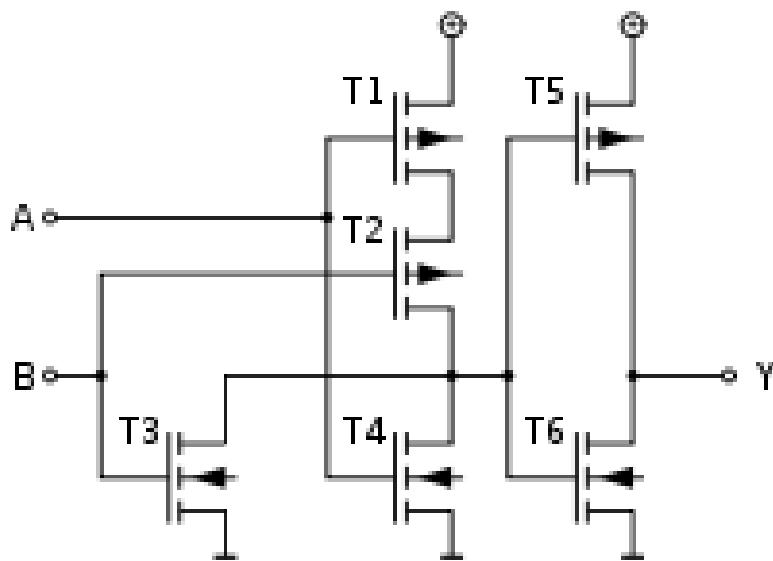
- Nur bedingt für hohe Spannungen geeignet, die ON-Verluste sind ab ca. 250V höher als bei einem IGBT.
- Parasitäre Diode parallel zur Drain-Source Strecke ist immer enthalten, das (Ab-) Schaltverhalten dieser Dioden ist meist schlechter als separate Dioden, was häufig zu unerwünschten Schwingungen führt.
- Empfindlicher gegen ESD am Gate als BJT
- Positiver Temperaturkoeffizient (TK), der ist stark temperaturabhängig und steigt von 25°C (Datenblattangabe) auf 150°C ungefähr um den Faktor 2.
- Dadurch steigen auch die Verluste und damit die Erwärmung des Bauteiles



- Das Bild links zeigt das Schaltbild eines Nicht-Gatters.
- Liegt am Eingang High-Potential an, dann sperrt T1 und am Ausgang liegt Low-Potential.
- Liegt am Eingang Low-Potential an, dann sperrt T2 und am Ausgang liegt High-Potential.

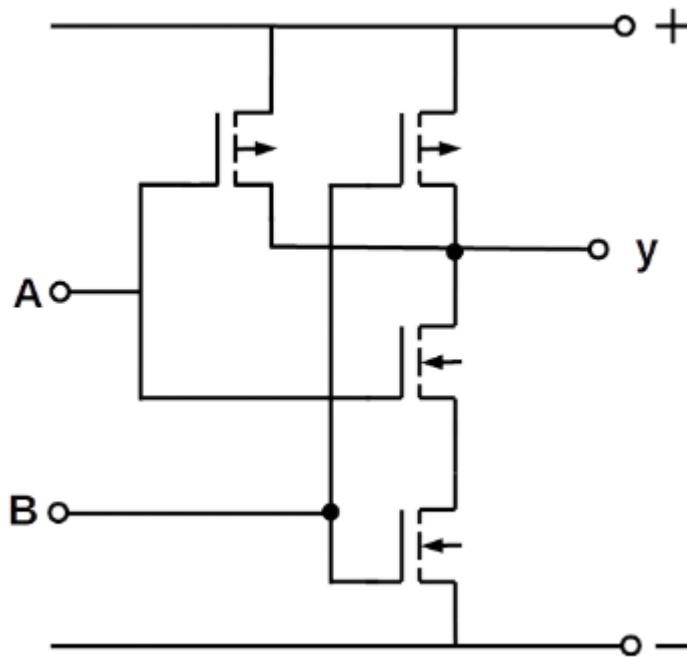


- Das Bild links zeigt das Schaltbild eines Und-Gatters.
- Liegt an den Eingängen A und B High-Potential an, dann leiten T3 und T4, wobei T1 und T2 sperren.
- Dadurch liegt an T5 und T6 Low-Potential an und T5 leitet und T6 sperrt, weswegen am Ausgang Y High-Potential anliegt.
- Bei allen anderen Eingangszuständen liegt Low-Potential am Ausgang, weil T6 leitet.

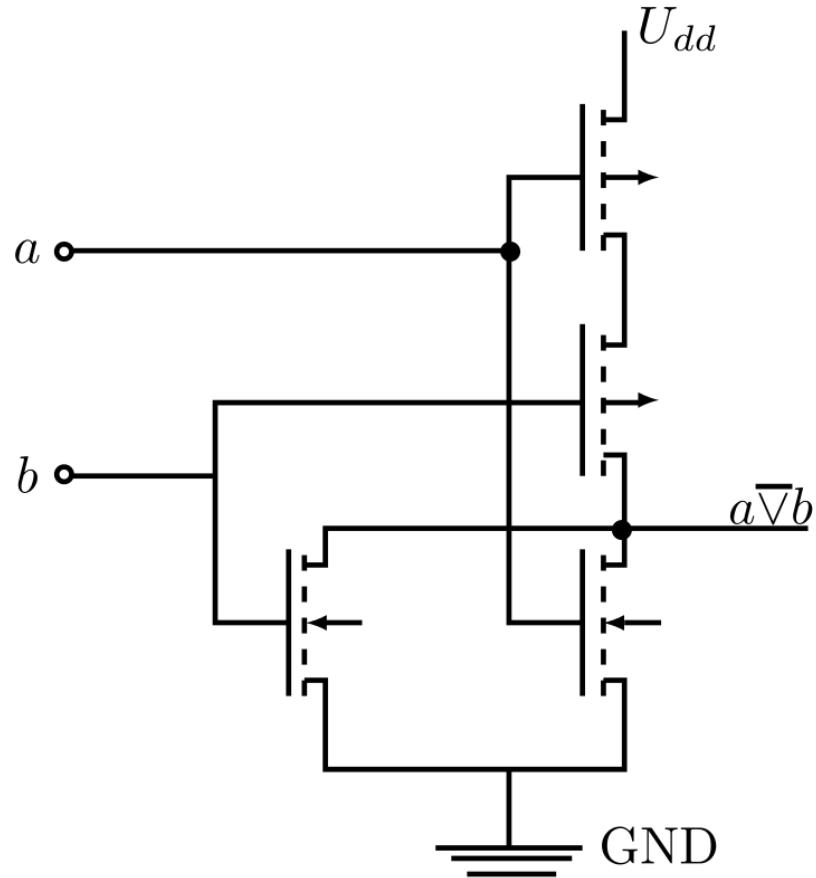


- Nebenstehende Abbildung zeigt das Schaltbild eines Oder-Gatters in CMOS-Technologie.
- Die Transistoren T1 bis T4 bilden zusammen ein NOR-Gatter.
- Liegt z. B. an Eingang A oder B High-Potential an, dann leitet T3 bzw. T4, wobei T1 bzw. T2 sperrt.
- Am Ausgang des NOR-Gatters realisieren die Transistoren T5 und T6 die Funktionalität eines NOT-Gatters.
- Ein negiertes NOR-Gatter ergibt das gewünschte OR-Gatter.
- Mit der erwähnten Eingangsbelegung an A und B liegt an T5 und T6 Low-Potential an und T5 leitet, aber T6 sperrt.
- Dadurch ergibt sich am Ausgang Y High-Potential.

NAND-GATT E R



- Das el. Potenzial an einem Punkt gibt die Spannung zwischen dem Punkt und einem Bezugspunkt an.
- Hier ist der Minuspol der Bezugspunkt.
- Liegt am Gate ein positives Potenzial, so schaltet der n-FET durch (geschlossener Schalter) und der p-FET sperrt (offener Schalter).
- Ist das Potenzial am Gate Null, so schaltet der p-FET durch (geschlossener Schalter) und der n-FET sperrt (offener Schalter).





E N D E