3 Kodierung von Informationen

Bevor ich Ihnen im nächsten Kapitel die einzelnen Bausteine einer Computeranlage vorstelle, möchte ich Ihnen noch kurz zeigen, wie Daten kodiert sein müssen, damit der Computer sie versteht.

3.1 Digitale Informationsverarbeitung

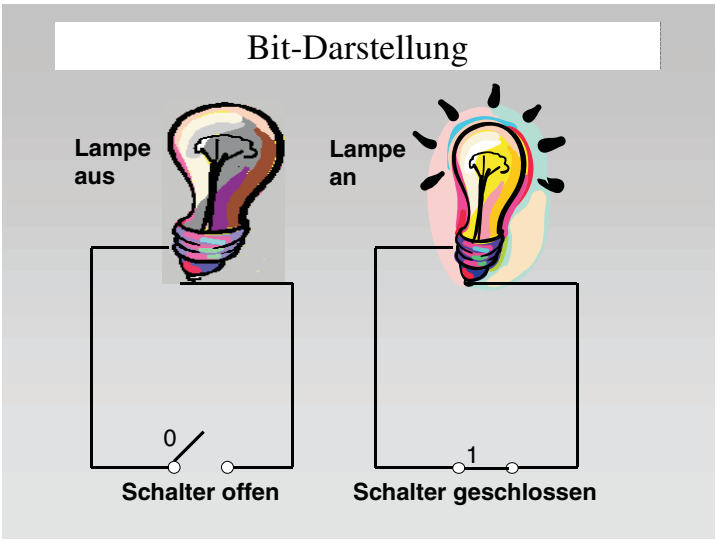
Daten werden innerhalb des Computers und seiner Peripherie in Form von Stromimpulsen weitergeleitet. Die kleinste und einfachste Informationseinheit besteht darin, dass entweder Strom fließt oder nicht. Dies ist eine zweiwertige Information.

Auch eine Glühbirne kann eine Informationseinheit darstellen. Bei einem geschlossenen Stromkreis fließt Strom, und die Birne brennt; ist der Stromkreis geöffnet, so bleibt sie dunkel.

Bit, Dualsystem

Ist die Glühbirne ausgeschaltet, kann dieser Zustand mit einer 0 kodiert werden; ist die Lampe an, schreiben wir eine 1. Auf diese Weise lassen sich mit einer Birne zwei verschiedene Zustände darstellen. Diese zwei Zustände ergeben eine Informationseinheit, die als *Bit* (Abkürzung für *binary digit*) bezeichnet wird; sie kann die Funktionszustände 0 und 1 annehmen. Ein Zahlensystem, das nur mit den Werten 0 und 1 arbeitet, ist das Dualsystem.

Bild 3.1:
Bit-Darstellung



3.2 Zahlensysteme

Sie wissen inzwischen, dass der Computer nur mit zwei verschiedenen Zuständen arbeiten kann, die durch ein Bit dargestellt werden. Alle Informationen, die wir in einen Computer eingeben, müssen in seine eigene Sprache, eben in Bits, übersetzt werden. Dies wird von speziellen Programmen erledigt, die sich des Dual- oder Sedezimalsystems und des ASCII-Codes bedienen. Obwohl Sie als normaler Computeranwender nur selten mit diesen Begriffen in Kontakt kommen, möchte ich sie Ihnen kurz vorstellen. So können Sie die Arbeitsweise des Computers besser verstehen.

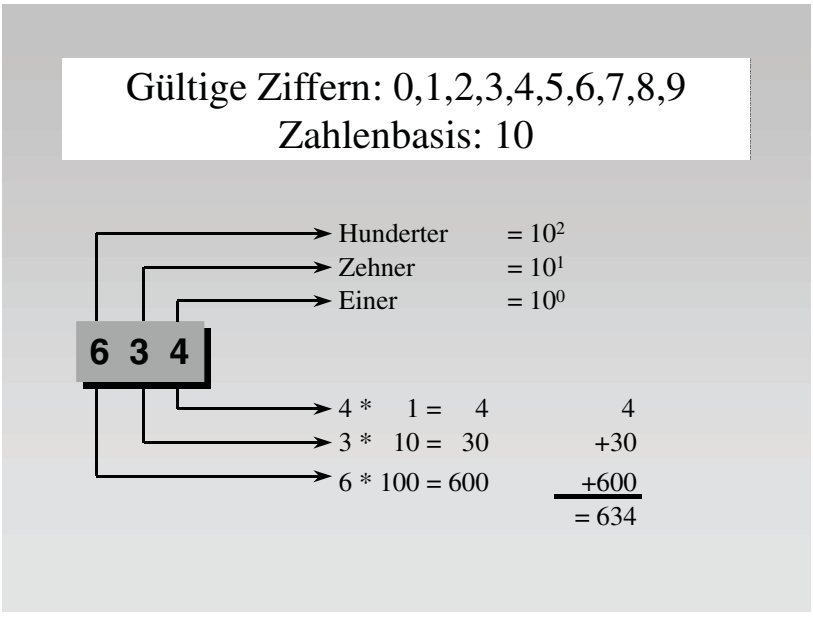
3.2.1 Das Dezimalsystem

Dezimalsystem

Schauen wir uns zunächst das Dezimalsystem an. Wenn wir die Zahl »634« sehen, lesen wir sofort »sechshundertvierunddreißig«. Wir unterstellen dabei automatisch, dass sich die Ziffern auf das Dezimalsystem beziehen. In diesem Zehnersystem wird mit zehn verschiedenen Zuständen gearbeitet, die durch die Ziffern 0 bis 9 dargestellt werden. Die Ziffern werden mit verschiedenen Potenzen zur Basis 10 multipliziert.

Wir lesen die Ziffern also nach folgendem Schema:

Bild 3.2:
Das Dezimalsystem



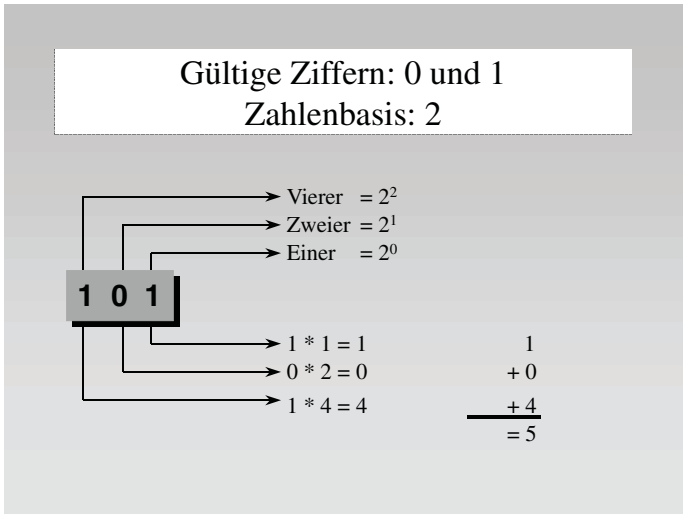
3.2.2 Das Dualsystem

Dualsystem

Im Dualsystem gibt es nur die Ziffern 0 und 1, die mit den verschiedenen Potenzen zur Zahlenbasis 2 multipliziert werden.

Betrachten wir dazu folgendes Schema zur Berechnung von Dualzahlen:

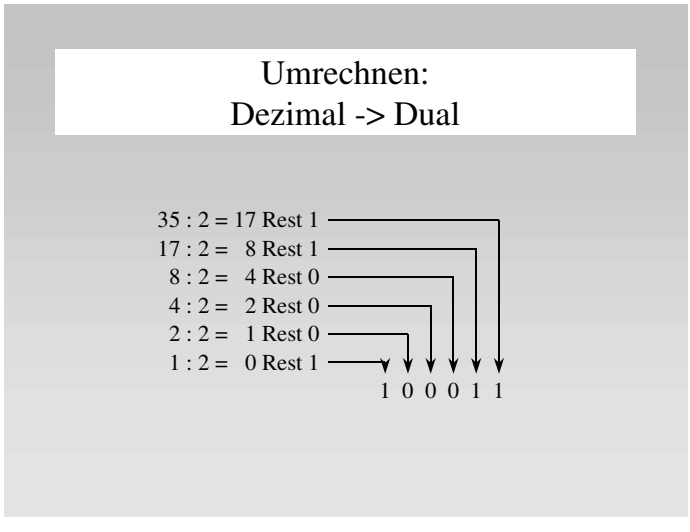
Bild 3.3: Das Dualsystem



Von der Dezimal- zur Dualzahl

Möchten Sie eine Dezimalzahl in eine Dualzahl umwandeln, muss die Dezimalzahl fortgesetzt durch 2 dividiert werden, wobei die Reste der Reihe von rechts nach links aufgeschrieben werden:

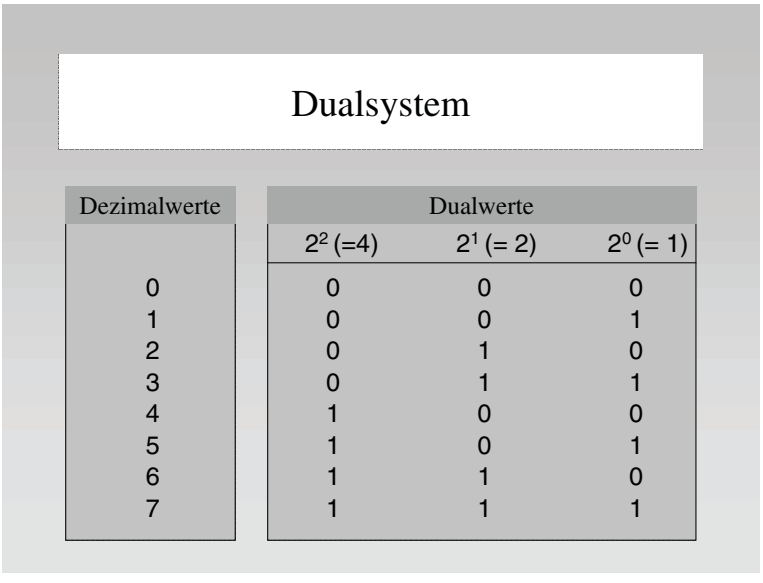
Bild 3.4: Umwandlungsverfahren: Von Dezimal nach Dual



Zusammenfassen von Bits

Nimmt man nun drei Informationseinheiten, also drei Bit, lassen sich 2x2x2 oder 2^3 verschiedene Dezimalzahlen darstellen:

Bild 3.5: Darstellung von Zahlen mit 3 Bit



Byte

Die Anzahl der kodierbaren Dezimalzahlen hängt von der Anzahl der zur Verfügung stehenden Bits ab. Mit jedem zusätzlich angefügten Bit wird die Anzahl der möglichen Bitkombinationen verdoppelt. Mit n Bit können 2^n verschiedene Zahlen kodiert werden. In der Datenverarbeitung hat man sich darauf geeinigt, 8 Bit zu einem Oktett zusammenzufassen, das allgemein als *Byte* bezeichnet wird. Mit einem Byte ergeben sich somit 256 Kombinationsmöglichkeiten.

3.2.3 Das Hexadezimalsystem

Ein Computer versteht Programme nur, wenn sie mit den Werten 0 und 1 kodiert sind. Für einen Programmierer ist es jedoch sehr schwierig, Programme in Dualzahlen zu schreiben bzw. zu lesen. Dies kann allerdings erleichtert werden.

Halbbyte

Ein Byte mit 8 Bit lässt sich in 2 Halbbyte mit jeweils 4 Bit aufteilen.

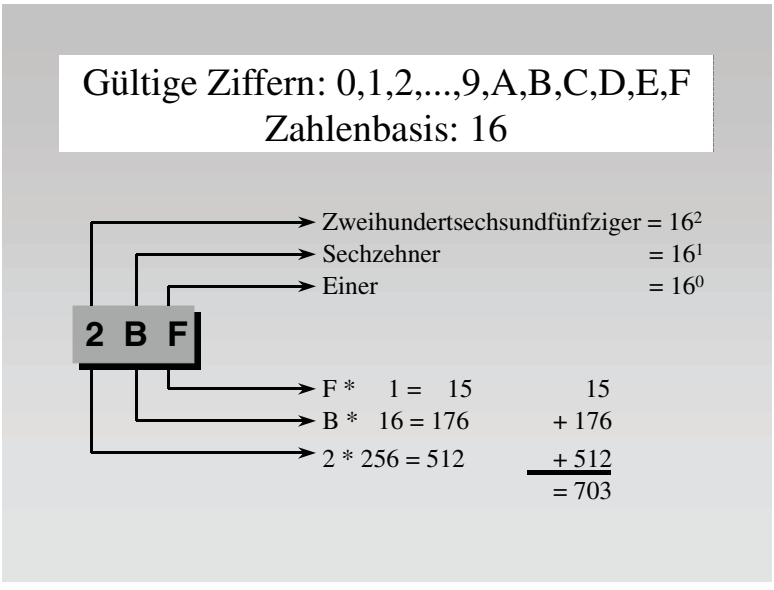
Tabelle 3.1: Aufteilung eines Byte in zwei Halbbyte

1 Byte (= 8 Bit)	
1 Halbbyte (= 4 Bit)	1 Halbbyte (= 4 Bit)

Sedezimalsystem

Mit 4 Bit sind 16 verschiedene Bitkombinationen möglich. Es gibt nun ein Zahlensystem, das mit 16 verschiedenen Zuständen arbeitet, das Sedezimal- oder Hexadezimalsystem. In diesem Zahlensystem sind die Ziffern 0 bis 9 um die Buchstaben A bis F erweitert.

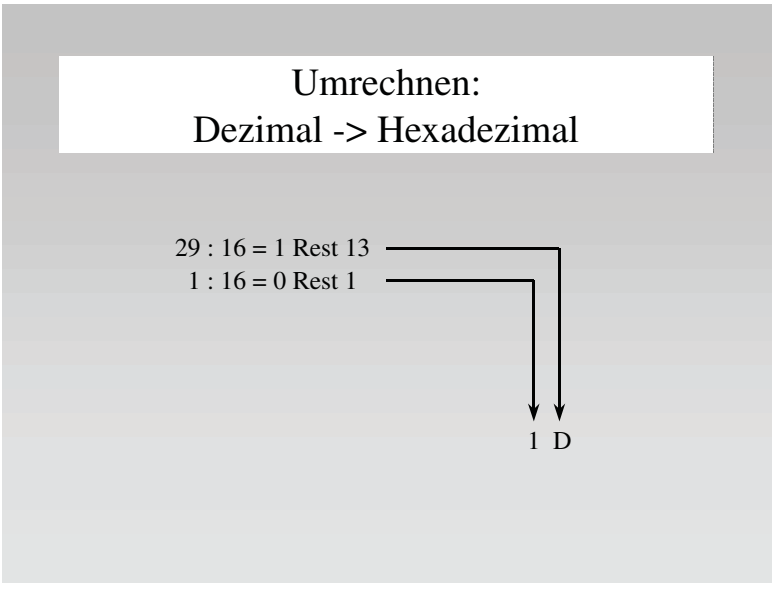
Bild 3.6: Das Hexadezimalsystem



Von der Dezimal- zur Hexadezimalzahl

Möchten Sie eine Dezimalzahl in eine Hexadezimalzahl umwandeln, muss die Dezimalzahl fortgesetzt durch 16 dividiert werden, wobei die Reste der Reihe nach von rechts nach links aufgeschrieben werden:

Bild 3.7: Umwandlungsverfahren: von Dezimal nach Hexadezimal

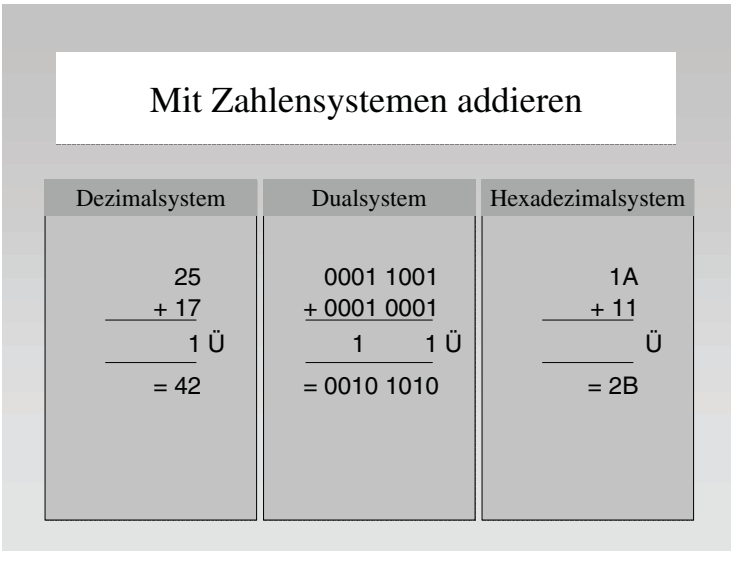


3.2.4 Rechnen mit den verschiedenen Zahlensystemen

Vom Dezimalsystem sind wir es gewohnt, mit den Zahlen zu rechnen. Um zwei oder mehrere Zahlen zu addieren, schreibt man sie untereinander, und unter einem Summenstrich wird das Ergebnis notiert. Ist z.B. die Summe der Einer größer als neun, wird nur der Einer des Ergebnisses unter den Strich geschrieben, während der Zehner, der so genannte Übertrag, »im Kopf behalten wird«. Er wird bei der Summierung der Zehner mitberücksichtigt.

Im Dual- wie im Sedezimalsystem wird nach dem gleichen Prinzip gerechnet. Im Dualsystem ergibt sich ein Übertrag aber bereits bei der Summierung von zwei Einsen. Im Sedezimalsystem muss der Übertrag berücksichtigt werden, wenn die Summe den Wert 15, der durch den Buchstaben 'F' dargestellt wird, überschreitet.

Bild 3.8: Addition im Dezimal-, Dual- und Sedezimalsystem



3.3 Kodierungsstandards

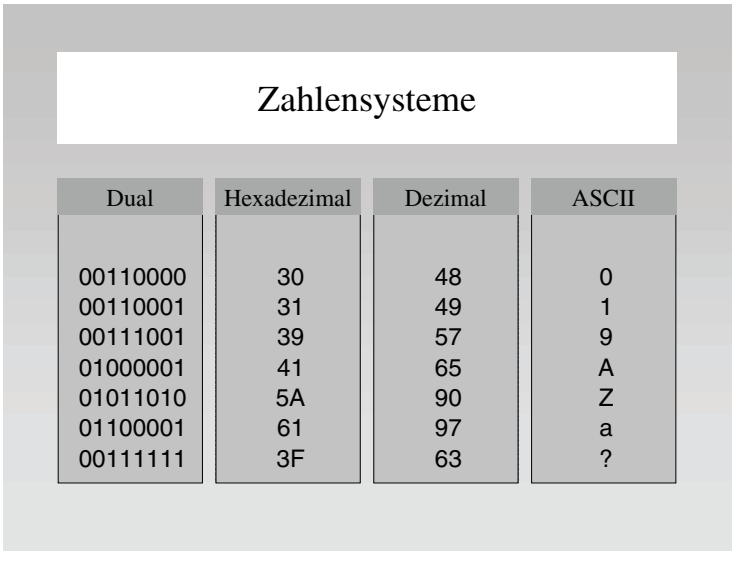
ASCII-Code

Ein Byte entspricht 8 Bit und ergibt somit 256 unterschiedliche Bitkombinationen. Diesen Kombinationen können nun Bedeutungen zugewiesen werden. Jedes Zeichen unserer Schrift (Buchstaben, Satzzeichen usw.) lässt sich einer bestimmten Bitkombination zuordnen. Die Zuordnung ist im so genannten ASCII-Code (American Standard Code of Information Interchange) einheitlich geregelt. Dieser Code bestand ursprünglich aus 7 Bits und einem Prüfbit, das zur Kontrolle diente, ob ein Zeichen auch richtig übermittelt wurde. Mit 7 Bits konnten 128 Zeichen kodiert werden. Als die

Speichertechnologie der Computer immer sicherer wurde, konnte man auf das Prüfbit verzichten und erweiterte den ASCII-Code auf 8 Bit bzw. 256 Zeichen.

Die folgende Abbildung zeigt eine Auswahl von kodierten Zeichen:

Bild 3.9: Kodierungsbeispiele



3.4 Kodierung von logischen Informationen

Logische Aussagen

Ein Computer kann nicht nur rechnen, er muss auch logische Entscheidungen treffen können. Logische Entscheidungen beruhen auf logischen Aussagen. Der Satz: »Wenn am Sonntag die Sonne scheint, gehe ich ins Freibad«, ist eine solche logische Aussage.

Derartige logische Aussagen kann man formalisiert folgendermaßen darstellen:

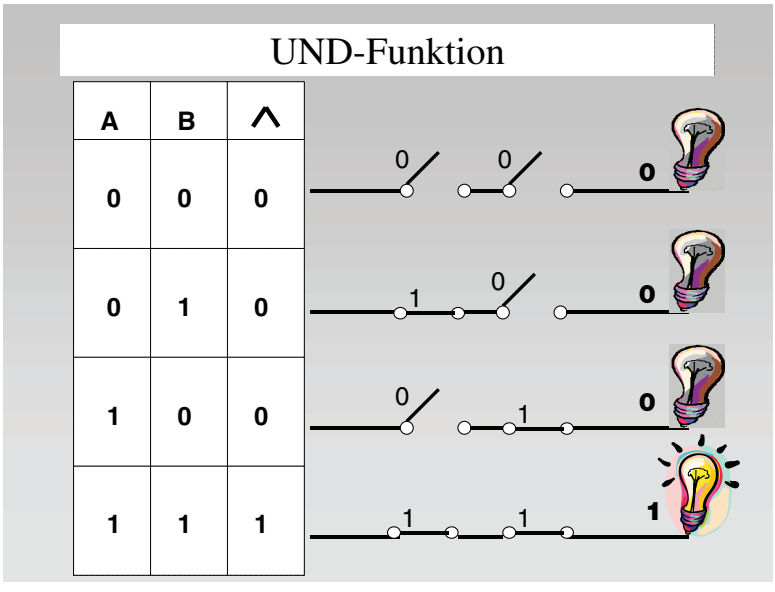
```
(A)   Es ist Sonntag
      und
(B)   es ist schönes Wetter
      dann
(C)   gehe ich ins Freibad.
```

UND-Funktion

Da die beiden Aussagen A und B durch UND miteinander verbunden sind, müssen beide Aussagen zutreffen, damit ich ins Freibad gehe. Statt "zutreffen" sagt man auch, die Aussagen sind wahr (oder falsch). In der Mathematik werden diese beiden Zustände durch 1 für wahr bzw. 0 für falsch kodiert.

Man kann logische Aussagen auch in Wahrheitstafeln darstellen. Dabei stellen die Buchstaben A und B jeweils Aussagen dar, die wahr oder falsch sein können. Die Verknüpfung von zwei Aussagen mit der UND-Funktion wird durch das /\-Zeichen dargestellt.

Bild 3.10:
UND-Funktion



Die UND-Funktion ist nicht die einzige Möglichkeit, Aussagen zu verbinden. Angenommen, ich habe Urlaub und kann jeden Tag ins Freibad gehen, dann wäre folgende Aussage möglich:

```
(A)   Es ist Sonntag
      oder
(B)   es ist schönes Wetter
      dann
(C)   gehe ich ins Freibad.
```

ODER-Funktion

In diesem Beispiel sind die beiden Bedingungen durch ODER verbunden, d.h., es muss nur eine Bedingung erfüllt sein, damit ich ins Freibad gehe. In der Wahrheitstafel wird das ODER durch das v-Zeichen symbolisiert.

NICHT-Funktion

In der Elektronik arbeitet man mit Schaltungen, die solche logischen Entscheidungen treffen können: die Gatter. Neben dem UND-Gatter und dem ODER-Gatter gibt es solche, die die NICHT-Funktion umsetzen können. Bei diesen Schaltungen wird ein Eingangssignal in das Gegenteil verkehrt, aus 1 wird 0 und aus 0 wird 1.

Bild 3.11:
ODER-Funktion

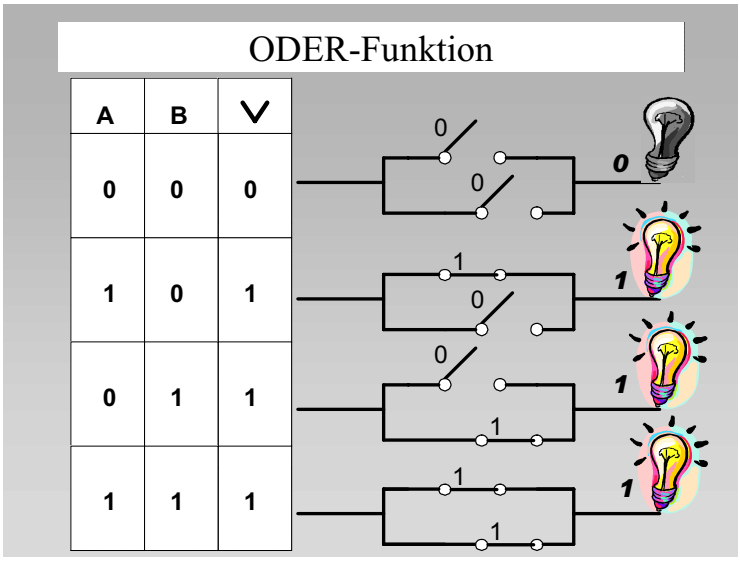
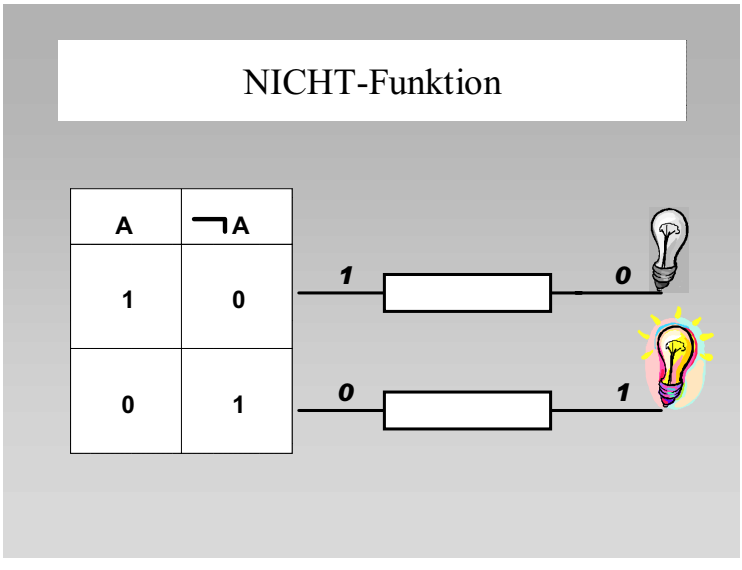


Bild 3.12:
NICHT-Funktion



Gatter Logische Gatter sind Grundbausteine, aus denen sich weitere wichtige Bausteine zusammensetzen lassen, die es z. B. ermöglichen, mit den Werten 1 und 0 zu rechnen oder die logischen Zustände 1 und 0 zu speichern. Diese Gatter sind letztendlich die Grundlage für die Entwicklung von hochintegrierten Bausteinen wie moderne Prozessoren und Speicherchips.