Informatik 5bi Schuljahr 2018/2019 als PDF exportieren

Informatik 5. Klasse - Schuljahr 2018/19

Lehrinhalte

- Lehrplaninhalte

Remote-Zugriff auf Schulserver

Kapitel

- 1) Zahlensysteme in der Informatik
- 2) Schaltalgebra

Leistungsbeurteilung

- **Test (SA)**
  - 2x Test (1h) pro Semester
- **Mitarbeit (MA)**
  - Aktive Mitarbeit im Unterricht (aMA)
  - Mündliche Stundenwiederholungen (mMA)
  - Schriftliche Stundenwiederholungen (sMA)
- **Praktische Arbeiten (PA)**
  - 1x praktischer Arbeitsauftrag pro Woche
- Aktueller Leistungsstand

Stoff für den 2. Test in Informatik – 5BI – ??.??.????

Stoff für den 1. Test in Informatik – 5BI – 17.10.2018

Was wird in der 5. Klasse gemacht?

Gesellschaftliche Aspekte der Informationstechnologie

Geschichte der Informatik

- Einen Überblick hinsichtlich der wichtigsten Entwicklungsstufen der Informatik gewinnen

Kommunikation und Kooperation

- Digitale Systeme zum Informationsaustausch, zur Unterstützung der Unterrichtsorganisation und zum Lernen auch in kommunikativen und kooperativen Formen verwenden können.
- Informationsmanagement und Lernorganisation für die eigene Lernarbeit und Weiterbildung mit geeigneter Software in der Praxis umsetzen und dabei vorhandene Informationsquellen erschließen und unterschiedliche Informationsdarstellungen ausgehend von den Vorkenntnissen anwenden

Verantwortung, Datenschutz und Datensicherheit

- Wesentliche Maßnahmen und rechtliche Grundlagen im Zusammenhang mit Datensicherheit, Datenschutz und Urheberrecht kennen lernen sowie die Auswirkungen des Technikeinsatzes auf die Einzelnen und die Gesellschaft nachvollziehen

Berufliches Spektrum der Informatik

- Einsatzmöglichkeiten der Informatik in verschiedenen Berufsfeldern kennen lernen und somit in ihrer Berufsorientierung Unterstützung finden

Informatiksysteme – Hardware, Betriebssysteme und Vernetzung

Technische Grundlagen und Funktionsweisen (Hardware)

- Den Aufbau von digitalen Endgeräten beschreiben und erklären können.
- Die Funktionsweise von Informatiksystemen erklären können.
- Die Komponenten eines Rechners und ihr Zusammenspiel kennen
- Die verschiedenen Formen von Datenträgern kennen und bewerten können
- Speichermedien konfigurieren können, Partitionierung von Festplatten erstellen und bearbeiten können

Betriebssysteme und Software (DOS und Windows, Netzwerk)

- Grundlagen von Betriebssystemen erklären, eine graphische Oberfläche und Dienstprogramme benutzen können.
- Grundlegende Konsolenkommandos kennen und anwenden können
- Mit graphische Oberflächen (Desktop, Modern UI-Oberfläche) umgehen können
- Batches und System-Scripts erstellen können
- Wesentliche Konfigurationsmöglichkeiten von Windows kennen
- Ein vernetztes Informationssystem für die individuelle Arbeit aufbauen und nutzen können

Algorithmik und Programmierung

Zahlensysteme und Schaltalgebra

- Die Zahlensysteme der Informatik kennen und damit umgehen können
- Rechenregeln der Schaltalgebra kennen und anwenden können
- Grundschaltungen und sowie Halbaddierer-, Volladdierer-Schaltung kennen und Schaltungssimulationen erstellen können

Algorithmen und Datenstrukturen

- Grundprinzipien von Automaten, Algorithmen, Datenstrukturen und Programmen erklären können
- Grundlegende Datentypen kennen
- Einfache Algorithmen erklären, entwerfen, darstellen können.
- Die Codierung einfacher Algorithmen kennen und an einfachen Beispielen anwenden können
- Wichtige und bekannte Algorithmen aus Mathematik und Informatik (z.B. aus Teilbarkeitslehre, Reihenfolgeprobleme, ...) kennen
- Möglichkeiten der Visualisierungen von Algorithmen kennen

Programmierung (Objektorientierte Programmiersprache)

- Algorithmen in einer Programmiersprache implementieren können
- Kontrollstrukturen (Verzweigungen, Schleifen) kennen und einsetzen können
- Unterprogramme kennen und einsetzen können

Mathematische Arbeitsumgebungen

- Mit mathematischen Arbeitsumgebungen umgehen können

Angewandte Informatik, Datenbanksysteme und Internet

Allgemein

- Einblicke in wesentliche Begriffe und Methoden der Informatik, ihre typischen Denk-und Arbeitsweisen, ihre historische Entwicklung sowie ihre technischen und theoretischen Grundlagen gewinnen und Grundprinzipien von Automaten, Algorithmen und Programmen kennen lernen
- Begriffe und Konzepte der Informatik verstehen und Methoden und Arbeitsweisen anwenden können

Grundlagen der Bild-, Ton- und Videobearbeitung

- Standardsoftware zur Bildbearbeitung kennen und einsetzen können.
- Standardsoftware zur Audiobearbeitung kennen und einsetzen können.
- Standardsoftware zur Videobearbeitung kennen und einsetzen können.

Textverarbeitungs- und Satzsysteme

- Grundlagen der Text- und Seitengestaltung kennen und anwenden können
- Techniken zur Bearbeitung großer Dokumente (wie z.B. VWA) kennen und anwenden können
- den sicheren Umgang mit Standardsoftware zur schriftlichen Korrespondenz, zur Dokumentation, zur Publikation von Arbeiten erreichen

Präsentationssysteme und Visualisierung

- Standardsoftware zur Kommunikation und Dokumentation sowie zur Erstellung, Publikation und multimedialen Präsentation eigener Arbeiten einsetzen können.
- den sicheren Umgang mit Standardsoftware zur multimedialen Präsentation sowie zur Kommunikation erreichen
- Inhalte systematisieren und strukturieren sowie Arbeitsergebnisse zusammenstellen und multimedial präsentieren können

Web-Techniken

- Grundlagen in HTML (Grundformatierung, Tabellen, Container, Einbau von Bild-, Ton- und Videoelementen) kennen und anwenden können
- Das Editieren in einem Content-Management-System beherrschen

---

```
=> 5. Klasse (3 Stunden,  2-3 Tests pro Semester)
```

1) Zahlensysteme in der Informatik

1.01) Dualsystem

Das **duale Zahlensystem - auch Dualsystem oder Binärsystem** genannt - besteht aus **2 Ziffern**, gekennzeichnet durch 0 und 1. Man benötigt dieses Zahlensystem in der Informatik, da sich mit technischen Bauteilen sehr leicht die Zustände AN und AUS erzeugen lassen können. Diese Zahlen können entsprechend unserem „normalen“ Dezimalsystem verwendet werden. Man kann sie addieren, subtrahieren, multiplizieren und dividieren. Da sie sich also kaum vom „normalen“ Rechnen unterscheiden, eignen sie sich hervorragend, um in der EDV eingesetzt zu werden.

Zählen im Dualsystem

Auch hier beginnen wir mit 0 und zählen dann 1. Leider haben wir nur 2 Zahlen, also gehen uns hier die Zahlen schnell aus. Wir machen es jetzt aber genau wie im Dezimalsystem und nehmen eine Stelle dazu. Nach 0 und 1 kommen dann also 10 und 11. Wieder reichen die Stellen nicht! Also noch eine dazu: 100, 101, 110, 111, usw.

Zählen von 0 bis 15 im Dezimal- und Dualsystem

Dezimal	Dual
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Eine andere Schreibweise

Man kann Zahlen auch anhand ihrer Basis darstellen. Im Dezimalsystem haben wir 10 Zahlen zur Verfügung, von 0 bis 9. Mit 2 Stellen können wir also 10 * 10 = 100 Zahlen darstellen. 100 Zahlen? Aber 100 hat doch drei Stellen! Dieser Einwand stimmt. Da wir jedoch mit der Zahl 0 beginnen, ist 0 die 1. Zahl, 1 die 2. Zahl, … 98 die 99. Zahl und 99 die 100. Zahl.

Mit 3 Stellen können wir 10 * 10 * 10 = 1000 Zahlen darstellen. Jede Stelle entspricht einer 10-er Potenz.

An einem einfachen Beispiel versuche ich diesen Sachverhalt zu erklären.

Wir nehmen dazu die Zahl 372 und schreiben sie als kleine Rechnung auf:

$372 = 3*100 + 7*10 + 2*1$.

Das kann man jetzt noch anders darstellen als:

$3*10^2 + 7*10^1 + 2*10^0$.

Auf diese Weise kann man jetzt alle anderen Zahlen auch darstellen:

$6574 = 6*10^3 + 5*10^2 + 7*10^1 + 4*10^0$
$12032 = 1*10^4 + 2*10^3 + 0*10^2 + 3*10^1 + 2*10^0$

Verwendung der Potenzschreibweise bei binären Zahlen

Wendet man die Potenzschreibweise bei binären Zahlen an, so muss man eine andere Basis wählen. Es gibt ja nur 2 verschiedene Ziffern, 0 und 1. Also nehmen wir als Basis 2.

Die Zahl 1011 schreibt sich dann als

$1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

Die Umrechnung von dezimalen in binäre Zahlen

Bei der Umrechnung der Dezimalzahlen verwenden wir die „Division mit Rest“ aus der Grundschule. Wir teilen die Zahl solange durch 2, bis als Ergebnis 0 herauskommt und merken uns dabei den Rest. Als Beispiel sollen die Zahlen 13 und 14 dienen.

13 / 2 = 6 Rest 1
6 / 2 = 3 Rest 0
3 / 2 = 1 Rest 1
1 / 2 = 0 Rest 1

Die Reste von unten nach oben aneinander gereiht ergeben dann die Dualzahl 1101.

14 / 2 = 7 Rest 0
7 / 2 = 3 Rest 1
3 / 2 = 1 Rest 1
1 / 2 = 0 Rest 1

Hieraus ergibt sich dann die Dualzahl 1110.

Addition von Dualzahlen

Einige erinnern sich vielleicht noch an die Addition von Zahlen wie wir sie in der Grundschule gelernt haben. Wir schreiben die Zahlen untereinander und addieren sie Stelle für Stelle. Dabei beginnen wir mit der letzten Stelle und arbeiten uns langsam nach vorne durch. Die gleiche Vorgehensweise

benutzen wir nun auch wenn wir Dualzahlen addieren wollen.

Die Addition funktioniert wie bei der Addition von Dezimalzahlen:

```
 0111
+0100
 ====
 1011
```

Addition mit Überlauf:

```
 1111
+0100
 ====
10011
```

Es gelten die Regeln 0+0=0, 1+0=1, 0+1=1, 1+1=0 Übertrag 1. Im Prinzip also nichts neues. Addiert man im Dezimalsystem 2 Zahlen so kommt bei 5+5 auch 0 Übertrag 1 heraus. Der Übertrag wird bei beiden System jeweils voran gestellt. Also gilt im Dualsystem 1+1=10.

Vorsicht Überlauf!

Die Addition ist im Prinzip problemlos. Es gibt allerdings einen Haken an der Sache: Die Addition funktioniert nur innerhalb eines bestimmten Wertebereiches. Woran liegt das? In der Realität können wir beliebig grosse Zahlen darstellen. Das geht leider nicht in der Informatik. Wir haben nur einen begrenzten Raum bzw. Speicherplatz zur Verfügung. Wir müssen aus diesem Grund den Speicherbereich einschränken (siehe was ist ein Byte), in unserem Beispiel nehmen wir als Speicherplatz ein Byte. Normalerweise verwendet man zur Addition von ganzen Zahlen, einen deutlich größeren Wertebereich, aber um einen überschaubaren Rahmen zu haben, begrenzen wir uns absichtlich auf ein Byte. Unsere größte darstellbare Zahl ist die 11111111, also dezimal 255. Was passiert nun, wenn wir eine 00000001, also 1, addieren? Das verrückte ist: es kommt 00000000, also 0 heraus. Da bekanntlich 1+1=0 Übertrag 1 gibt, bekommt man als Ergebnis in Wirklichkeit nicht 00000000 sondern 00000000 Übertrag 1. Dieser letzte Übertrag kann jedoch nicht mehr gespeichert werden und wird deshalb einfach ersatzlos gestrichen. Es ergibt sich daraus jedoch auch eine gewisse Logik. Durch meinen beschränkten Wertebereich komme ich irgendwann an meine obere Grenze. Bei der Addition von 1 fängt dann jedoch der Wertebereich wieder von vorne an, ich bin jetzt an der unteren Grenze, man durchläuft dann wieder den Bereich bis zur oberen Grenze, usw. Wenn wir also immer und immer wieder 1 addieren zählen wir unendlich oft von 0 bis 255, dann wieder 0 bis 255 usw.

Subktraktion von Dualzahlen

Drei Schritte zu Subtraktion

Hier kommen wir mit unserer normalen Schulmathematik nicht mehr weiter. Bevor wir uns mit dem komplizierten „Warum ist das denn so?“ beschäftigen, merken wir uns erst einmal den Mechanismus. Die Subtraktion von binären Zahlen wird durch die **Addition des Zweierkomplementes**

durchgeführt. Zur Erklärung beginnen wir im ersten Schritt mit dem **Einerkomplement**, dann schauen wir im zweiten Schritt was das **Zweierkomplement** ist und dann kommen wir im letzten Schritt zur **Subtraktion**.

Das Einerkomplement

Was ist das Komplement von Dualzahlen? Man bildet das sogenannte Einerkomplement, indem man jede Zahl durch ihr Gegenteil ersetzt, also die 0 durch die 1 und die 1 durch die 0.

01011010 wird zu 10100101 11101101 wird zu 00010010

Das Zweierkomplement

Das Zweierkomplement entspricht dem Einerkomplement, nur wird zusätzlich noch 00000001 addiert.

01011010 wird im Einerkomplement zu 10100101 im Zweierkomplement zu 10100110 11101101 wird im Einerkomplement zu 00010010 im Zweierkomplement zu 00010011

Die Subtraktion von Dualzahlen

Der Satz lautet: Die Subtraktion von 2 Zahlen erfolgt durch die Addition des Zweierkomplementes. Als konkretes Beispiel nehmen wir dazu die Rechnung 14-9=5.

!!WICHTIG:!!
Die restlichen Ziffern müssen immer mit 0 aufgefüllt werden.

9 ist im Dualsystem 00001001.

Das Einerkomplement zu 00001001 ist 11110110.

Das Zweierkomplement 11110111.

Dies addieren wir nun zu 14 also 00001110.

```
 00001110
+11110111
 ========
 00000101
```

Auch hier wäre die richtige Zahl eigentlich 00000101 Übertrag 1, da wir den Übertrag jedoch nicht speichern können, bleiben wir bei 00000101 was ja der Dezimalzahl 5 entspricht.

1.02) Hexadezimalsystem

Besonders **wichtig** ist in der **Informatik und Digitaltechnik** neben dem Binärsystem auch das **Hexadezimalsystem (Sedezimalsystem)**. Das Hexadezimalsystem verwendet die **Basis 16**, d.h. es gibt **16 verschiedene Ziffern**, **0 bis 9** und zusätzlich die Buchstaben **A bis F** (sog. Zahlzeichen; können auch als klein geschrieben werden: a-f).

Mit dem Hexadezimalsystem können auf einfachere und kürzere Weise Binärzahlen notiert werden. Mit einer 4-stelligen Binärzahl (auch als Halbbyte oder Nibble bezeichnet) lassen sich 16 (24 = 16) verschiedene Zahlen darstellen, und zwar 0 bis 15 (die Null zählt mit!). Da das Hexadezimalsystem die Basis 16 (= 24) verwendet, reicht eine (!) Hexadezimalzahl aus, um vier Bits (Binärziffern) darzustellen. Mit zwei Hexadezimalzahlen kann ein Byte (8 Bits) angeschrieben werden.

Gegenüberstellung Hexadezimal-, Binär- und Dezimalsystem

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

Um eindeutig darauf hinzuweisen, dass es sich um eine Hexadezimalzahl handelt, kann ebenso wie in anderen Zahlensystemen die Basis tiefgestellt dazu geschrieben werden, z.B. $\mathbf{3F_{16}}$ (= 63_{10} dezimal) oder $\mathbf{93_{16}}$ (= 147_{10} dezimal). Es sind aber auch andere Schreibweisen üblich:

a) Vorangestelltes **0x (Prefix), z.B. 0x93**. Diese Notation wird in Programmiersprachen mit C-ähnlicher-Syntax verwendet.

b) Nachgestelltes **h** (Postfix), z.B. **93h**. Letztere Schreibweise ist besonders in der Technik gebräuchlich.

Umrechnung vom Dezimal- ins Hexadezimalsystem

Die Umrechnung funktioniert ähnlich der Umrechnung von Dezimal- zu Binärzahlen (s.o.). Nun muss aber, statt durch 2, durch 16 dividiert werden. Die Reste werden genauso von rechts nach links angeschrieben und geben, wenn das Ergebnis der Ganzzahlendivision 0 ist, das Endergebnis.

Beispiel: Die Dezimalzahl **304_{10}** soll in eine Hexadezimalzahl umgewandelt werden.

```
304 / 16 = 19 => 0 Rest
 19 / 16 =  1 => 3 Rest
  1 / 16 =  0 => 1 Rest
```

Für das Endergebnis werden jetzt die Reste **von unten nach oben gelesen**.
Somit ergibt sich ein Endergebnis von 130_{16}, das entspricht der Dezimalzahl **304_{10}**.

Umrechnung vom Hexadezimal- ins Dezimalsystem

Die Umrechnung vom Hexadezimal- ins Dezimalsystem kann genauso wie oben von Binär→Dezimal demonstriert, erfolgen. Die einzelnen Ziffern werden mit dem jeweiligen Stellenwert (16^n, wobei n = 0, 1, 2, ...) multipliziert und die jeweiligen Ergebnisse aufsummiert. Das folgende Beispiel demonstriert dies anhand der Hexadezimalzahl 130_{16}:

```
0 * 16^0  =    0
3 * 16^1  =   48
1 * 16^2  =  256
---------------
          =  304
```

Als Ergebnis erhalten wir 304 dezimal, womit die Probe - zur vorigen Rechnung in die umgekehrte Richtung - erfolgreich war. 304_{10} entspricht 130_{16}. Diese Antwort hätte in der Praxis natürlich auch ein wissenschaftlicher Taschenrechner geliefert. 😀 Es reicht dazu sogar der **Windows-Rechner** (den Sie nur auf die wissenschaftliche Ansicht umstellen müssen) oder unter Linux Programme wie z.B. KCalc.

1.03) Oktalsystem

Das Oktalsystem, auch Achtersystem genannt, verwendet die **Basis 8 (acht)**. Um Zahlen darzustellen, stehen die **Ziffern 0 bis 7** zur Verfügung. Die Bedeutung in der Informatik/Digitaltechnik ergibt sich dadurch, dass sich mit einer **Oktalzahl drei Bits** darstellen lassen. 23 ist 8, somit lassen sich mit 3 Bits 8 verschiedene Möglichkeiten darstellen. Eine Oktalzahl reicht, um diese Information wiederzugeben.

Das Oktalsystem wird hier insbesondere deshalb erwähnt, weil in vielen Programmiersprachen Zahlen auch in Oktalform angegeben werden können. Meist, z.B. in PHP, wird dazu eine 0 (Null) vorangestellt, z.B. 077 für 77_8 (= 63_{10}).

Umrechnungen erfolgen genauso wie beim Hexadezimalsystem gezeigt. Um die Oktalzahl auszurechnen, die einer best. Dezimalzahl entspricht, dividieren Sie die Dezimalzahl fortlaufend durch 8 und schreiben die Reste von rechts nach links an. In umgekehrter Richtung - von Oktal nach Dezimal - multiplizieren Sie die einzelnen Ziffern mit dem Stellenwert (8^n für n = 0, 1, 2, ...) und addieren die Teilergebnisse.

Umrechnung vom Dezimal- ins Oktalsystem

Die Umrechnung funktioniert ähnlich der Umrechnung von Dezimal- zu Binärzahlen (s.o.). Nun muss aber, statt durch 2, durch 8 dividiert werden. Die Reste werden genauso von rechts nach links angeschrieben und geben, wenn das Ergebnis der Ganzzahlendivision 0 ist, das Endergebnis.

Beispiel: Die Dezimalzahl **304_{10}** soll in eine Hexadezimalzahl umgewandelt werden.

```
304 / 8 = 38 => 0 Rest
 38 / 8 =  4 => 6 Rest
  4 / 8 =  0 => 4 Rest
```

Für das Endergebnis werden jetzt die Reste **von unten nach oben gelesen**.
Somit ergibt sich ein Endergebnis von 460_8, das entspricht der Dezimalzahl **304_{10}**.

Umrechnung vom Oktal- ins Dezimalsystem

Die Umrechnung vom Oktal- ins Dezimalsystem kann genauso wie oben von Binär→Dezimal demonstriert, erfolgen. Die einzelnen Ziffern werden mit dem jeweiligen Stellenwert (8^n, wobei n = 0, 1, 2, ...) multipliziert und die jeweiligen Ergebnisse aufsummiert. Das folgende Beispiel demonstriert dies anhand der Oktalzahl 460_8:

```
0 * 8^0  =    0
6 * 8^1  =   48
4 * 8^2  =  256
---------------
         =  304
```

Als Ergebnis erhalten wir 304 dezimal, womit die Probe - zur vorigen Rechnung in die umgekehrte Richtung - erfolgreich war. 304_{10} entspricht 460_{8}. Diese Antwort hätte in der Praxis natürlich auch ein wissenschaftlicher Taschenrechner geliefert. Es reicht dazu sogar der **Windows-Rechner** (den Sie nur auf die wissenschaftliche Ansicht umstellen müssen) oder unter Linux Programme wie z.B. KCalc.

Umrechnungen

Prinzipiell kann man jede Zahl von einem Zahlensystem in ein anderes Zahlensystem umrechnen. Dazu muss man normalerweise die Zahl immer zuerst in das Dezimalsystem und dann in das Ziel-Zahlensystem umrechnen. Ausnahmen gibt es bei Umrechnungen vom Binärsystem in ein anderes Zahlensystem, hier braucht man nicht unbedingt das Dezimalsystem.

Binärsystem ⇔ Oktalsystem

Binärsystem ⇒ Oktalsystem

Wir wissen ja bereits, dass eine Ziffer im Oktalsystem die Ziffern 0-7 annehmen kann und dass man 3 Bits im Dualsystem braucht um 8 verschiedene Zahlenwerte darzustellen.

⇒ Immer 3 Stellen im Binärsystem ergeben eine Ziffer/Stelle im Oktalsystem

Beispiel

Umwandlung der Zahl 010110011111_2 ins Hexadezimalsystem:

```
010 110 011 111
 ⇓   ⇓   ⇓   ⇓
 2   6   3   7
```

Die Zahl 010110011111_2 entspricht somit der Zahl 2637_8.

Oktalsystem ⇒ Binärsystem

Selbiges gilt auch für die Umrechnung vom Oktalsystem in das Binärsystem.

⇒ Eine Stelle im Oktalsystem entspricht 3 Stellen im Binärsytem

Beispiel

Umwandlung der Zahl 672_8 ins Binärsystem:

```
 6   7   2
 ⇓   ⇓   ⇓
110 111 010
```

Die Zahl 672_8 entspricht somit der Zahl 110111010_2.

Binärsystem ⇔ Hexadezimalsystem

Binärsystem ⇒ Hexadezimalsystem

Wir wissen ja bereits, dass eine Ziffer im Hexadezimalsystem die Werte 0-15 annehmen kann und dass man 4 Bits im Dualsystem braucht um 16 verschiedene Zahlenwerte darzustellen.

⇒ Immer 4 Stellen im Binärsystem ergeben eine Ziffer/Stelle im Oktalsystem

Beispiel

Umwandlung der Zahl 010110011111_2 ins Oktalsystem:

```
0101 1001 1111
 ⇓    ⇓    ⇓
 5    9    F
```

Die Zahl 010110011111_2 entspricht somit der Zahl $59F_{16}$.

Hexadezimalsystem ⇒ Binärsystem

Selbiges gilt auch für die Umrechnung vom Hexadezimalsystem in das Binärsystem.

⇒ Eine Stelle im Hexadezimalsystem entspricht 4 Stellen im Binärsytem

Beispiel

Umwandlung der Zahl 672_{16} ins Binärsystem:

```
 6    7    2
 ⇓    ⇓    ⇓
0110 0111 0010
```

Die Zahl 672_{16} entspricht somit der Zahl 11001110010_2.

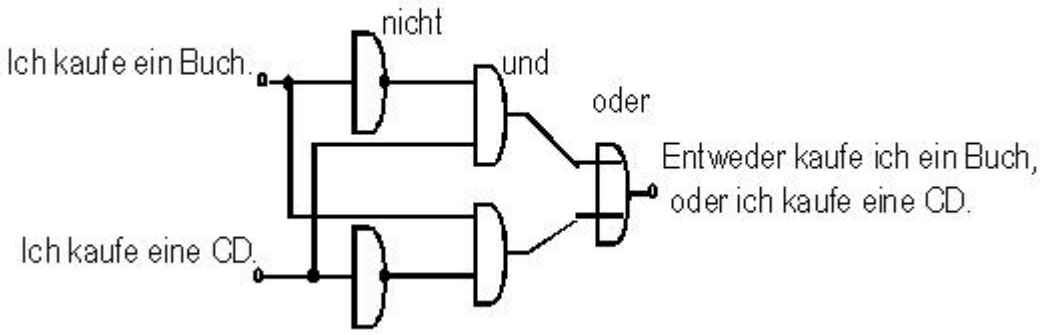
Oktalsystem ⇒ Hexadezimalsystem

Will man vom Oktalsystem ins Hexadezimalsystem umrechnen, rechnet man am besten zuerst ins Binärsystem und dann ins Hexadezimalsystem um.

Hexadezimalsystem ⇒ Oktalsystem

Will man vom Hexadezimalsystem ins Oktalsystem umrechnen, rechnet man am besten zuerst ins Binärsystem und dann ins Oktalsystem um.

2) Schaltalgebra



https://elektroniktutor.de/swfdatei/gatter.swf

Das Adobe Flash Plugin wird benötigt, um diesen Inhalt anzuzeigen.

2.01) Grundlagen

In der Digitaltechnik sind die Eingangsvariablen so miteinander zu verknüpfen, dass die Ausgangsvariable einen definierten Zustand annimmt, um damit einen Prozess zu steuern. So soll ein Fahrstuhl nur dann fahren, wenn eine Zieletage gewählt wurde, in der er zurzeit nicht steht, seine Tür vollständig geschlossen ist und von ihr nichts eingeklemmt wurde und die Kabine nicht überlastet ist. Man kann durch Kombinieren der beschriebenen digitalen Gatter und Ausprobieren bei einfacheren Aufgaben die eine oder andere Lösung finden. Das eigentliche Ziel ist es, eine wirtschaftliche Digitalschaltung zu entwickeln, die mit einem Minimum gleicher Gattertypen zum Ziel führt.

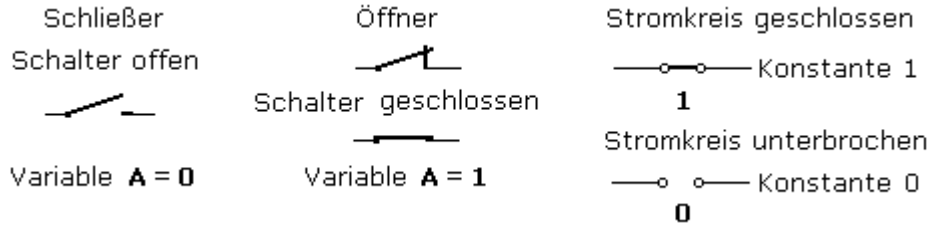
Der englische Mathematiker Georg Boole entwickelte eine Mengenalgebra, die in angepasster Weise als Boolesche Schaltalgebra bei der Problemlösung hilfreich ist. In der Schaltalgebra gibt es Variablen und Konstanten. Die binäre Digitaltechnik kommt mit zwei definierten logischen Zuständen, der 0 und 1 aus.

Konstante

In der Schaltalgebra gibt es nur die zwei konstanten Größen 0 und 1. In der elektronischen Schaltung entspricht eine dauerhaft geschaltete Leitung der Konstanten mit dem Zustand 1. Die dauerhafte Unterbrechung eines Stromkreises steht für die Konstante mit dem Wert 0.

Variable

Veränderbare, schaltbare Eingangsgrößen und davon abhängige veränderliche Ausgangswerte werden als Variable bezeichnet. In der binären Digitaltechnik nehmen sie entweder den Zustand 0 oder 1 an. In elektronischen Schaltungen können sie mit einem Schalter verglichen werden. Der geöffnete Schalter entspricht einer Variablen mit dem Wert 0. Bei geschlossenem Schalter nimmt die Variable den Wert 1 an.



Wir wissen, dass alle Grundrechnungsarten auf eine Addition zurückgeführt werden können. Deshalb ist das Ziel dieses Kapitels eine Maschine simulieren zu können, die addieren kann.

Informationseinheiten

BIT

Die Ziffern 0 und 1 werden physikalischen Zuständen zugeordnet:

```
0   Strom fließt nicht
1   Strom fließt
```

0 und 1 werden in der Informatik mit Bit bezeichnet. Ein Bit stellt die kleinste Informationseinheit dar (nur 2 Zustände – 0 und 1).

1 Bit ⇒ 2^1 ⇒ 2 Möglichkeiten
2 Bit ⇒ 2^2 ⇒ 4 Möglichkeiten
3 Bit ⇒ 2^3 ⇒ 8 Möglichkeiten
8 Bit = 1 Byte ⇒ 2^8 ⇒ 256 Möglichkeiten

BYTES

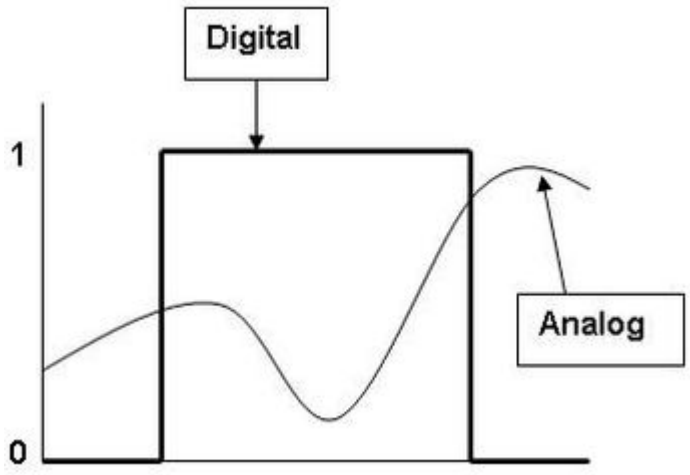
8 Bits sind ein Byte!

1 KB (Kilobyte) = 2^{10} = 1024 Byte
1 MB (Megabyte) = 2^{10} KB = 2^{10} x 2^{10} Byte = 2^{20} Byte = 1048576 Byte
1 GB (Gigabyte) = 2^{10} Byte
1 TB (Terrabyte) = 2^{10} Byte

Darstellung von 0 und 1

Ein Computer verarbeitet systembedingt Digitalsignale, da nur ausgewertet werden kann, ob eine Spannung anliegt oder nicht bzw. innerhalb eines Definitionsbereichs einen Wert über- oder unterschreitet. Diese zwei Zustände werden auch häufig bezeichnet als:

HIGH und **LOW**
TRUE und **FALSE**
AN und **AUS**
1 und **0**



2.02) Grundschaltungen

UND-Verknüpfung (=Serienschaltung)

Bei der UND-Verknüpfung führt der Ausgang das Signal 1, wenn an beiden Eingängen (a und b) das Signal 1 anliegt.

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)	Zeichen (Mathematik)
UND	AND	Konjunktion	X=A∧B	X=A*B

Schaltwerttabelle

a	b	a∧b
0	0	0
0	1	0
1	0	0
1	1	1

Schaltfunktion

f(a,b)=a∧b

Schaltsymbol

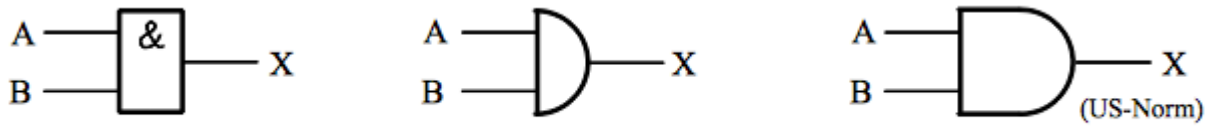


Bild 5-5: Schaltzeichen für UND-Gatter

elektronische Schaltung

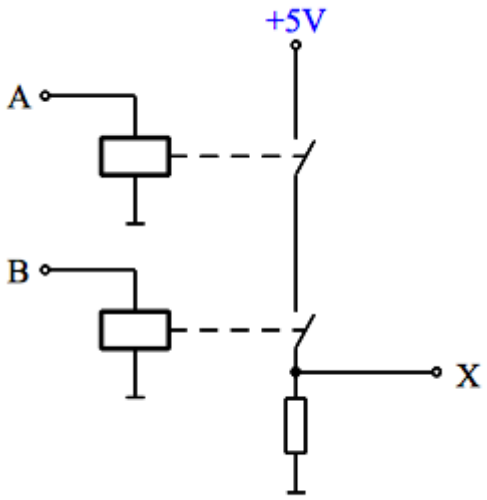


Bild 5-4: UND-Verknüpfung mit Relais

ODER-Verknüpfung (=Parallelschaltung)

Bei der ODER-Verknüpfung führt der Ausgang das Signal 1, wenn an einem der beiden Eingänge das Signal 1 anliegt.

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)	Zeichen (Mathematik)
ODER	OR	Disjunktion	X=A∨B	X=A+B

Schaltwerttabelle

a	b	a∧b
0	0	0
0	1	1
1	0	1
1	1	1

Schaltfunktion

f(a,b)=a∨b

Schaltsymbol

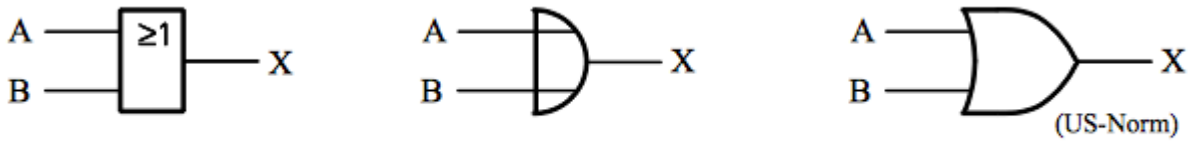


Bild 5-8: Schaltzeichen für ODER-Gatter

elektronische Schaltung

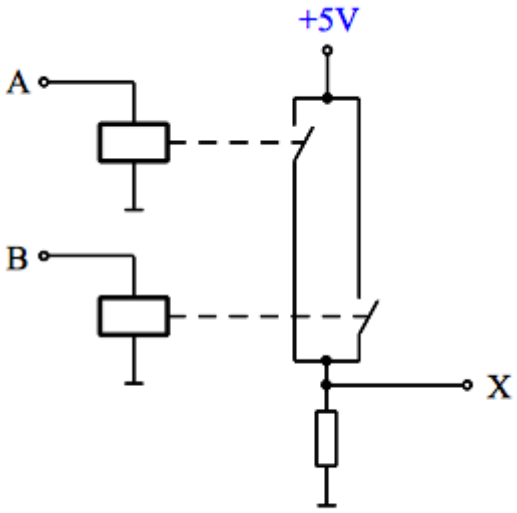


Bild 5-7: ODER-Verknüpfung mit Relais

NICHT-Verknüpfung (=NOT-Schaltung)

Bei einer NICHT-Verknüpfung wird der Ausgang logisch 1, wenn der Eingang logisch 0 ist, und umgekehrt.

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)	Zeichen (Mathematik)
NICHT	NOT	Negation	X=¬A	X=Ā

Schaltwerttabelle

a	¬a
0	1
1	0

Schaltfunktion

f(a)=¬a

Schaltsymbol

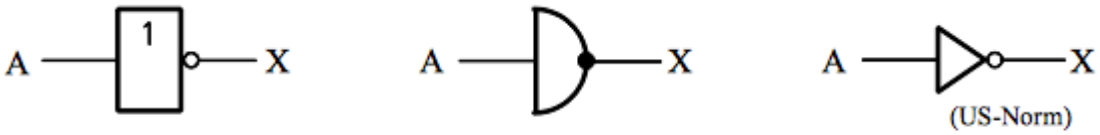


Bild 5-11: Schaltzeichen für NICHT-Gatter

elektronische Schaltung

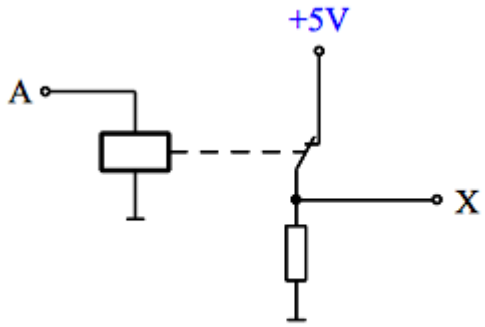


Bild 5-10: NICHT-Verknüpfung mit Relais

2.03) Zusammengesetzte Schaltungen

Die **Grundverknüpfungen** werden in der Regel miteinander **kombiniert**, sodass die Logik der Schaltung verändert wird. Beispielsweise kann man eine UND- mit einer NICHT-Verknüpfung kombinieren und erhält dadurch eine sogenannte NAND-Verknüpfung (NICHT-UND). Da solch zusammengesetzte Schaltfunktionen sehr häufig verwendet werden, haben sie **eigene Schaltzeichen** bekommen. Mit den folgenden Schaltfunktionen werden die Grundverknüpfungen erweitert.

- NAND
- NOR
- XOR
- XNOR

NAND-Verknüpfung

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.

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

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)
NEGIERTES UND	NOT AND	Negative Konjunktion	$Z = \overline{A \wedge B}$

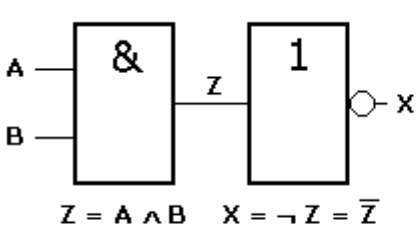
Schaltwerttabelle

a	b	a⊼b
0	0	1
0	1	1
1	0	1
1	1	0

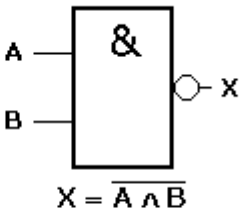
Schaltfunktion

f(a,b)=a⊼b

Schaltsymbol



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

Bild 5-13: Schaltzeichen für NAND-Gatter

NOR-Verknüpfung

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.

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)
NEGIERTES ODER	NOT OR	Negative Disjunktion	$Z = \overline{A \vee B}$

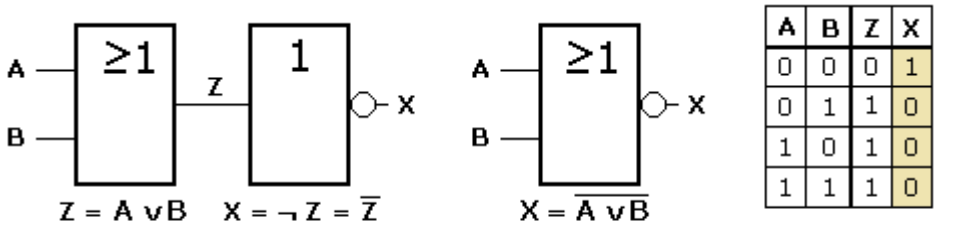
Schaltwerttabelle

a	b	a⊽b
0	0	1
0	1	0
1	0	0
1	1	0

Schaltfunktion

f(a,b)=a⊽b

Schaltsymbol



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

Wahrheitstabelle

Bild 5-15: Schaltzeichen für NOR-Gatter

XOR-Verknüpfung

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.

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

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)
Exklusives ODER (Antivalenz)	Exclusiv OR	Exklusive Disjunktion	Z=a⊻b

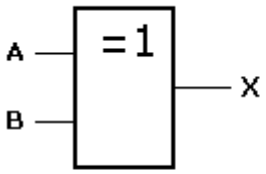
Schaltwerttabelle

a	b	a⊻b
0	0	0
0	1	1
1	0	1
1	1	0

Schaltfunktion

f(a,b)=a⊻b

Schaltsymbol



Exklusiv ODER
XOR - Antivalenz

$X = (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$
$X = A \underline{\vee} B$
$X = A \oplus B$

Funktionsgleichung

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

Wahrheitstabelle

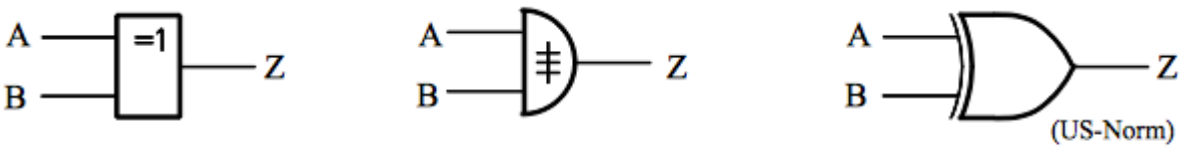


Bild 5-21: Schaltzeichen für ANTIVALENZ-Gatter

XNOR-Verknüpfung

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.

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

Zeichen

Deutsch	Englisch	Fachausdruck	Zeichen (boolsche Algebra)
Exklusives NICHT ODER (Äquivalenz)	Exclusiv NOT OR	Biimplikation	Z=a≡b

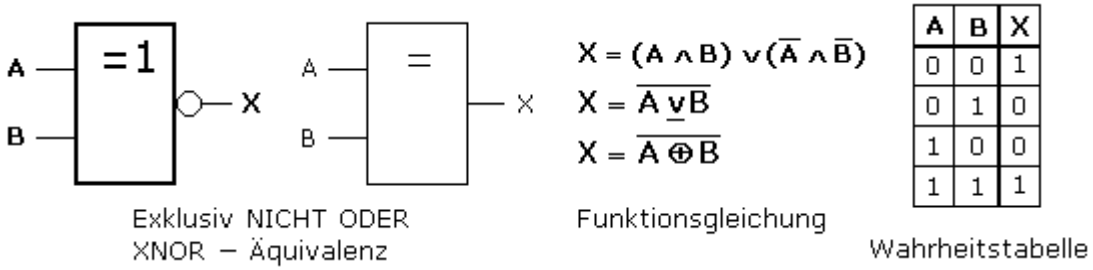
Schaltwerttabelle

a	b	a≡b
0	0	1
0	1	0
1	0	0
1	1	1

Schaltfunktion

f(a,b)=a≡b

Schaltsymbol



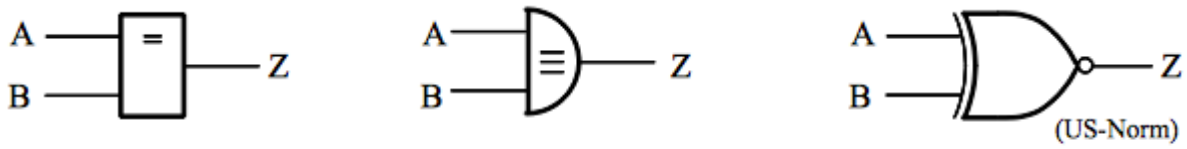


Bild 5-19: Schaltzeichen für ÄQUIVALENZ-Gatter

xor_nand.swf

Gesetze der Schaltalgebra

Vorrang- und Klammerregel

Wie in der Algebra ist auch in der Digitaltechnik auf eine bestimmte Reihenfolge der Operationen zu beachten. Die Negation sollte vor der Konjunktion (UND) und diese vor der Disjunktion (ODER) ausgeführt werden. Das ist vergleichbar mit der Aussage, dass die Multiplikation Vorrang vor der Addition hat.

Bei mehreren Variablen und unterschiedlichen Verknüpfungen kann eine Klammersetzung notwendig sein. Beachtet man die Vorrangregel, kann man auf viele Klammern verzichten. Ein Setzen zeigt sogleich eindeutig, welche der Variablen wie zu verknüpfen ist. Bei ODER sollte immer geklammert werden, ebenfalls beim Anwenden der De Morganschen Gesetze auf NAND- und NOR-Verknüpfungen.

$X = A \vee B \wedge C$

nach der Vorrangregel gilt

$\boxed{X = A \vee (B \wedge C)}$

festgelegte Abfolge der Verknüpfungen

$\boxed{Z = (A \vee B) \wedge C}$

A	B	C	B∧C	A∨(B∧C)	A∨B	(A∨B)∧C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	1	0
0	1	1	1	1	1	1
1	0	0	0	1	1	0
1	0	1	0	1	1	1
1	1	0	0	1	1	0
1	1	1	1	1	1	1

$A \vee (B \wedge C) \neq (A \vee B) \wedge C$

Bei drei Eingangsvariablen und binären Zuständen gibt es 2^3 = 8 Eingangskombinationen. Die Wahrheitstabelle zeigt bei definierter Klammersetzung oder Beachtung der Vorrangregel UND vor ODER unterschiedliche Ergebnisse.

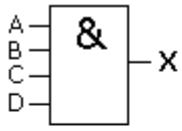
Kommutativgesetz

Das Kommutativ- oder Vertauschungsgesetz besagt, dass bei der UND sowie ODER Verknüpfung auch bei beliebiger Vertauschung der Reihenfolge der Eingangsvariablen das Ergebnis gleich bleibt.

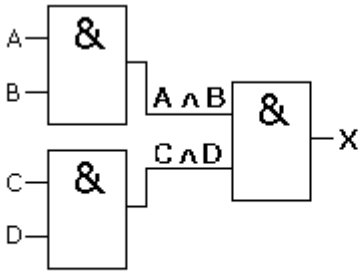
UND	$X = A \wedge B \wedge C = C \wedge A \wedge B = B \wedge C \wedge A$
ODER	$X = A \vee B \vee C = C \vee A \vee B = B \vee C \vee A$

Assoziativgesetz

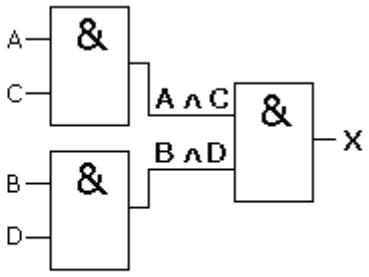
Das Assoziativ- oder Verbindungsgesetz besagt, dass bei einer UND- beziehungsweise ODER-Verknüpfung mit mehr als zwei Schaltvariablen die Verknüpfung auch stufenweise nacheinander in beliebiger Reihenfolge erfolgen kann.



A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



A∧B	C∧D	X
0	0	0
0	1	0
1	0	0
1	1	1



A∧C	B∧D	X
0	0	0
0	1	0
1	0	0
1	1	1

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

Beispiele zum Assoziativgesetz der UND-Verknüpfung

Anstelle des UND-Gatters mit vier Eingangsvariablen lassen sich zwei UND-Gatter mit zwei Eingängen verwenden. Die Variablen A, B, C, D werden einzeln beliebig mit den Eingängen verbunden. Die Ausgangsvariable der beiden UND-Gatter kann den Wert 0 oder 1 annehmen. Beide Ausgangsvariablen sind nochmals mit einem UND-Gatter zu verknüpfen, wobei sich wieder vier Eingangskombinationen ergeben. Nur wenn alle Eingangsvariablen 1 sind, wird der Ausgangszustand ebenfalls 1.

Das Assoziativgesetz gilt gleichermaßen für die ODER-Verknüpfung. Die Klammersetzung ist nicht notwendig und soll nur verschiedene Verteilungen besser erkennbar machen.

Distributivgesetz

Das Distributiv- oder Verteilungsgesetz wird zur Vereinfachung von Verknüpfungsgleichung angewendet. Es ist vergleichbar mit dem Ausmultiplizieren und Ausklammern von Variablen der normalen Algebra. Da die logische UND-Verknüpfung der algebraischen Multiplikation entspricht, während die ODER-Verknüpfung mit der Addition vergleichbar ist, gibt es zwei unterschiedliche Distributivgesetze.

Konjunktives Distributivgesetz

Eine Variable wird mit UND verknüpft und auf den Folgeausdruck verteilt. Die Vorgehensweise entspricht dem aus der Algebra bekannten Ausmultiplizieren eines Klammerausdrucks mit einem Faktor. Im umgekehrten Fall kann eine Variable, die mit mehreren anderen Variablen verknüpft ist, ausgeklammert werden. Das bedeutet für den Schaltungsaufbau eine Einsparung an Gattern.

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

⇨ Ausmultiplizieren
⇦ Ausklammern

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

		UND	ODER	
A	B	A∧B	A	X
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	1	1

Beim Ausklammern wird die Variable mit ihrem Verknüpfungszeichen, hier UND, vor die Klammer gesetzt. Die in der Klammer stehenden Variablen werden mit dem zuvor zwischen den Klammern stehenden ODER verknüpft.

Disjunktives Distributivgesetz

Eine Variable kann durch ein vergleichbares Ausmultiplizieren auf andere Ausdrücke verteilt werden oder bei mehrfachem Auftreten ausgeklammert werden. Beim Ausklammern wird das ODER Verknüpfungszeichen mit der Variablen vor die Klammer geschrieben. disjunktives Distributivgesetz

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

⇨ Ausmultiplizieren
⇦ Ausklammern

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

		ODER	UND	
A	B	A∨B	A	X
0	0	0	0	0
0	1	1	0	0
1	0	1	1	1
1	1	1	1	1

De Morgansche Gesetze

Die boolesche Algebra wurde vom englischen Mathematiker De Morgan weiter entwickelt. Für die Schaltalgebra gibt es zwei De Morgansche Gesetze. Sie machen Aussagen zur Negation einer Verknüpfung und der Umkehr von Verknüpfungszeichen. Mit den De Morganschen Gesetzen lassen sich bei der Entwicklung von Digitalschaltungen Gatter gegeneinander austauschen, Schaltungen verkleinern oder mit nur einem Gattertyp verwirklichen. Das erste Gesetz ist für die NAND-Verknüpfung und das zweite Gesetz entsprechend für die NOR-Verknüpfung definiert.

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

1. De Morgansche Gesetz

			ODER		
A	B	$\overline{A \wedge B}$	$\overline{A}$	$\overline{B}$	X
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

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

2. De Morgansche Gesetz

			UND		
A	B	$\overline{A \vee B}$	$\overline{A}$	$\overline{B}$	X
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

Ist eine Verknüpfung insgesamt negiert, so ist ihre Ausgangsvariable negiert. Die Negation kann auf die Einzelglieder aufgeteilt werden, wobei aus der Grundverknüpfung UND ein ODER beziehungsweise aus ODER ein UND wird. Eine doppelte Negation hebt sich auf. Getrennte Negationsstriche über

Variablen bedeuten, dass die Eingangsvariablen negiert sind.

Die oben zum 1. De Morganschen Gesetz gezeigte Verknüpfung kann schaltungstechnisch mit zwei NICHT- und einem ODER-Gatter verwirklicht werden. Wie zu erkennen ist, folgt das gleiche Ergebnis mit nur einem NAND-Gatter. Für das 2. De Morgansche Gesetz gilt die entsprechende Aussage. Zwei NICHT- und ein UND-Gatter reduzieren sich auf den Einsatz eines NOR-Gatters. Mit der doppelten Negation und den De Morganschen Gesetzen kann bei einer gegebenen Funktionsgleichung auf das Bestehen einer derartige Vereinfachung geprüft werden. Treten dabei mehrfache Negationen auf, so lassen sie sich von innen nach außen auflösen.

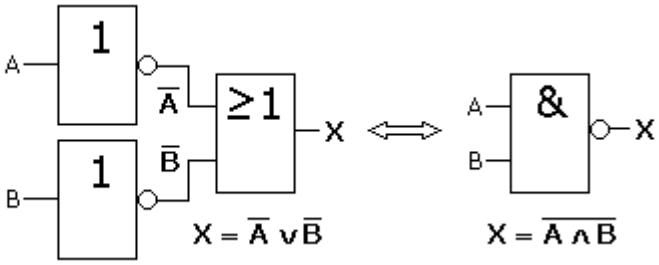
$$X = \overline{A} \vee \overline{B}$$

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

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

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

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



From:
http://elearn.bgamstetten.ac.at/wiki/ - **Wiki**

Permanent link:
http://elearn.bgamstetten.ac.at/wiki/doku.php?id=inf:inf5bi_201819:2:2_04

Last update: **2018/09/30 11:26**

Digitale Rechenschaltungen

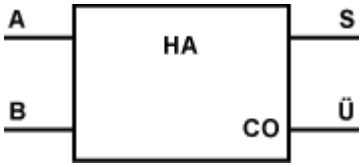
Aus logischen Verknüpfungen lassen sich digitale Schaltungen zusammenbauen, mit denen man Rechenvorgänge durchführen kann. Das heißt, diese Schaltungen haben zwischen ihren Eingängen eine Kombination aus logischen Verknüpfungen, die einem Rechenvorgang entspricht. In der Digitaltechnik kennt man Rechenschaltungen hauptsächlich für das duale Zahlensystem und den BCD-Code. Im Prinzip kann für jedes Zahlensystem eine Rechenschaltung aufgebaut werden.

Einige Rechenschaltungen der Digitaltechnik

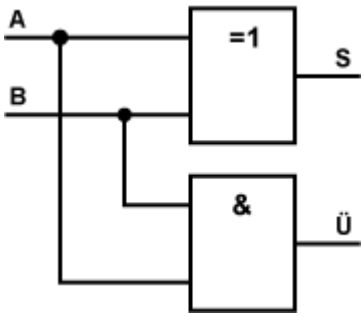
- Addiererschaltungen
- Subtrahiererschaltungen
- Addier-Subtrahier-Werke
- Multiplikationsschaltungen
- Arithmetisch-logische Einheit (ALU)

Stellvertretend für alle Rechenschaltungen dienen die folgenden Ausführungen zum Halbaddierer und dem Volladdierer.

Halbaddierer



Ein Halbaddierer ist die einfachste Rechenschaltung und kann zwei einstellige Dualziffern addieren. Der Eingang A des Halbaddierers ist der Summand A, der Eingang B ist der Summand B. Die Schaltung hat zwei Ausgänge. Der Ausgang S als Summenausgang (2^0) und der Ausgang Ü als Übertrag (2^1) für die nächsthöhere Stelle im dualen Zahlensystem.



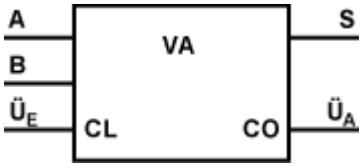
B	A	Ü	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Der Halbaddierer ist eine Schaltung aus den Verknüpfungsgliedern XOR und UND. Das XOR ist die Addier-Verknüpfung. Das UND stellt fest, ob ein Übertrag für die nächsthöhere Stelle vorgenommen

werden muss. Aus der Tabelle ist ersichtlich, dass das Ergebnis aus der Spalte Summe (S) einer Exklusiv-ODER-Verknüpfung (Antivalenz, XOR) entspricht. Das Ergebnis der Spalte Übertrag (Ü) entspricht einer UND-Verknüpfung. Die so entstandene Schaltung wird als Halbaddierer bezeichnet. Sie ist in der Lage zwei 1-Bit Summanden (einstellig) zu addieren.

Volladdierer

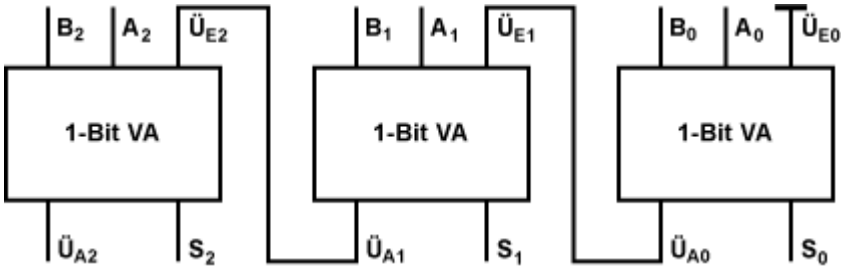
Um **mehrstellige Dualzahlen** addieren zu können benötigt man Schaltungen die auch einen Übertrag einer niederwertigen Stelle berücksichtigen. Man spricht vom **Übertragseingang ÜE**. Die Schaltung bezeichnet man als **Volladdierer (VA)**. Ein Volladdierer kann drei Dualzahlen addieren. Bzw. zwei Dualzahlen addieren und den Übertrag aus einer niederwertigen Stelle berücksichtigen.



Folgende Wahrheitstabelle ergibt sich:

$Ü_E$	B	A	$Ü_A$	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3-Bit-Volladdierer



Folgende Schaltung zeigt einen 3-Bit-Volladdierer (VA), der aus drei 1-Bit-Volladdierer (VA) realisiert wurde. Damit lassen sich zwei dreistellige Dualzahlen addieren. Der Eingang $Ü_{E0}$ liegt an 0 V (Masse), weil in der niederwertigsten Stelle kein Übertrag berücksichtigt werden muss.

Beispiel

```
 010 = A
+111 = B
```

```
---
1001 = S
```

0. Stelle

$Ü_{E0}$	A_0	B_0	$Ü_{A0}$	S_0
0	0	1	0	1

1. Stelle

$Ü_{E1}$	A_1	B_1	$Ü_{A1}$	S_1
0	1	1	1	0

2. Stelle

$Ü_{E2}$	A_2	B_2	$Ü_{A2}$	S_2
1	0	1	1	0

Ergebnis

$Ü_{A2}$ S_2 S_1 S_0

```
1  0  0  1
```

2.07) Übungen

1) Um ihren Swimmingpool schneller füllen zu können, hat Fam. Huber zwei Wasserleitungen (beide verfügen über einen Absperrhahn B und C verlegt, die direkt in den Pool führen. Diese Leitungen zweigen beide von einer Hauptleitung ab, die ebenfalls über einen Absperrhahn A verfügt. Demnach kann Wasser nur in den Pool fließen, wenn der Hauptabsperrhahn A und zumindest einer der anderen Absperrhähne offen sind.
Stelle die Situation mittels einer Wahrheitstabelle und einer Schaltfunktion (LogikSim, …) dar.

2) Frau Müller sagt zu ihrem Mann: „Den ganzen Sonntag hockst du auf dem Sofa, trinkst Bier und stopfst dir Kartoffelchips in den Schlund! Du solltest mal joggen!
Ihr Gatte erwidert: „Ich verspreche dir für nächsten Sonntag Folgendes:

- Wenn ich jogge, werde ich sogar auf Bier oder auf Chips verzichten.
- Wenn ich keine Chips esse, trinke ich kein Bier oder ich jogge nicht.
- Wenn ich aber jogge, brauche ich unbedingt hinterher ein Bier!
- Allerdings: Auch falls ich nicht jogge, werde ich auf jeden Fall Bier oder Chips zu mir nehmen.“

Falls Herr Müller seine Versprechen einhält, wie könnten seine Aktivitäten am nächsten Sonntag aussehen? Erstelle einen schaltalgebraischen Ausdruck und die Wahrheitstabelle zu diesen Aussagen.

3) Zeichne für folgende schaltalgebraischen Ausdrücke die zugehörige Schaltfunktion bzw. erstelle die jeweilige Wahrheitstabelle!

- f(A,B) = (¬A ∨ ¬B)
- f(A,B,C) = (¬A ∧ ¬B) ∧ C
- f(A,B) = (¬A ∨ ¬B) ∨ (¬A ∧ B)

4) Überprüfe das Gesetz von De Morgan mit Hilfe einer Leitwerttabelle und stelle die Schaltung im Digitalsimulator dar.

5) Überprüfe das Distributivgesetz mittels einer Leitwerttabelle und stelle die Schaltung im Digitalsimulator dar.

6) Vereinfache die Schaltfunktion f(a,b)= a ∧ (¬a v b) und baue die Schaltung im Digitalsimulator auf.

7) Vereinfache die Schaltfunktion f(a,b)= (a v b) ∧ (a v ¬b) ∧ (¬a v b)

8) Vereinfache die Schaltfunktion f(a,b)= (a ∧ b) v (a ∧ ¬b) v (¬a ∧ b)

9) Zur automatischen Brandbekämpfung wurden drei Sensoren in einem Raum angebracht. Melden mindestens zwei der drei Sensoren eine Rauchentwicklung, so schaltet sich die Sprinkleranlage ein. Entwirf eine Leitwerttabelle, eine Schaltfunktion und stelle diese mittels Digitalsimulator dar!