Inhalt

3 Hard- und Software eines Rechners (Teil I)

3.1 Algorithmus

Was macht eine elektronische Datenverarbeitungsanlage so revolutionär?
Sie ist eine Maschine, welche mittels *einfacher* Operationen (Basisoperationen) in *hoher* Geschwindigkeit *geistige Routinearbeit* durchführt.
Beispiel: Datenbankrecherchen.

Das Wort **Algorithmus** ist auf den persisch-arabischen Mathematiker und Astronom **MUHAMMAD IBN MUSA AL-HWÂRIZMÎ** (ca. 780 – 859, Bagdad) nach seinem Werk „*Kitab al-jabr w'al-muqabala*" um 825 zurückzuführen, „*al-jabr*" wurde später zu Algebra, **AL-HWÂRIZMÎ** zu Algorithmus.
Das Werk ist leider nicht mehr erhalten. Es behandelt algebraische Gleichungen, das indische Zahlensystem und das Rechnen in diesem. Eine lateinische Übersetzung aus dem 12. Jahrhundert „*Algorithmi de numero Inderum*" existiert noch.
Der Algorithmenbegriff wird seit der Jahrhundertwende 19./20. Jh. verstärkt diskutiert. Friedrich L. Bauer und Gerhard Goos[1] beschreiben diesen folgendermaßen:

Ein *Algorithmus* ist eine *präzise*, das heißt in einer festgelegten Sprache abgefasste, *endliche* Beschreibung eines *allgemeinen* Verfahrens unter Verwendung *ausführbarer* Verarbeitungsschritte zur Lösung einer Aufgabe.

Beispiel Problem ***r = ggT(m, n)***
Bestimmen des größten gemeinsamen Teilers *r* zweier natürlicher Zahlen *m* und *n*.
Euklidischer Algorithmus (nach dem griechischen Mathematiker **EUKLEIDES von ALEXANDREIA**, auch **EUKLID**, ca. 365 - 300 v. u. Z.):

ggT(130, 55):	**130** / **55** =	2	Rest	20
	55 / 20 =	2	Rest	15
	20 / 15 =	1	Rest	5
	15 / **5** =	3	Rest	0

⇒ **ggT(130, 55) = 5**

(Mathematischer Beweis in 3 Schritten: Algorithmus terminiert, liefert gT, liefert ggT)

Ein *Algorithmus* ist eine *Handlungsvorschrift*, keine Problembeschreibung.

Ein Algorithmus heißt **abbrechend** (**terminierend**), wenn er die gesuchte Größe nach endlich vielen Schritten liefert.

[1] BAUER, F. L. und GOOS, G. (1971):
Informatik - Eine einführende Übersicht, Springer Verlag. 2 Bde., 1. Aufl. 1971, 4. Aufl. 1991/92;
russische Übersetzung, Mir, Moskau, 1976;
polnische Übersetzung, Wydawnictwa Naukowo - Techniczna, Warschau, 1977.

3.2 Programm

Wie setzt man in einem Computer einen Algorithmus um?
Hauptkomponenten des Aufbaus eines Rechners

J. von Neumann – Architektur, 1947

Ein Computer ist mittels eines **Rechenwerks** in der Lage, *einfache Operationen* in hoher Geschwindigkeit auszuführen. Die *Reihenfolge* der auszuführenden Operationen und die notwendigen Operanden werden mit Hilfe eines **Steuerwerks** diesem zugeführt.

⇒ **Basisoperationen**: Das zu lösende Problem muss durch einfache Operationen beschreibbar sein.

⇒ **Arbeitsanweisungen**: Die Reihenfolge der auszuführenden Operationen muss bekannt sein, notwendige Daten müssen bereitstehen.

Ein *Programm* legt fest, welche *Arbeitsanweisungen* in welcher Reihenfolge zur Lösung eines Problems mit Hilfe der vorhandenen *Basisoperationen* notwendig sind.

Wir betrachten den folgenden Modellrechner:
Speicher

Als *Arbeitsspeicher* steht eine endliche Anzahl von Speicherzellen zur Verfügung.
Diese können genau eine Zahl einer nach oben beschränkten und damit endlichen Teilmenge der Menge der natürlichen Zahlen aufnehmen, $M \subset N$.

constant `0, 123`

Die einzelnen Zellen werden symbolisch mit einzelnen Kleinbuchstaben adressiert:

identifier `a, b, z`

Basisoperationen

Das *Rechenwerk* führt verschiedene Operationen aus. Operanden sind natürliche Zahlen oder die Inhalte von Speicherplätzen.
Rechenoperationen:
Addition +, Subtraktion −, Multiplikation `*`, ganzzahlige Division `/`, Rest `%`

expression `a % b`

Vergleichsoperationen: größer >, gleich == und ungleich `!=`

condition `a != 0`

Arbeitsanweisungen

1. *Wertzuweisungen* als *einfachste Arbeitsanweisungen*:
   Es wird einem Speicherplatz ein Wert zugewiesen. Links steht der Speicherplatz, dann folgt ein ‚=' und schließlich der zuzuweisende Speicherinhalt. Dieser wird direkt als natürliche Zahl angegeben, aus einem anderen Speicherplatz übernommen oder vom Rechenwerk berechnet. Abgeschlossen wird eine Wertzuweisung mit einem Semikolon:

   assignment
   ```
   a = 10;
   b = a;
   c = a * b;
   ```

2. *Zusammengesetzte Arbeitsanweisungen*:
   a. Mehrere Arbeitsanweisungen können *hintereinander* ausgeführt werden.
      sequence `a = m; b = n; c = a % b;`
   b. Arbeitsanweisungen können *wiederholt* ausgeführt werden.
      iteration `while( n != 0) { s = s + n; n = n - 1;}`
   c. Eine *Auswahl* der Arbeitsanweisungen soll ausgeführt werden.
      selection
      ```
      if( n > m) { c = m; m = n; n = c;}
      if( m > n) { x = m;} else { x = n;}
      ```

3. Abschluss: Nichts sonst ist erlaubt.

Beispiel Programm ***r = ggT(m, n)***

```
a = m;
b = n;
c = a % b;
while( c != 0)
{
  a = b;
  b = c;
  c = a % b;
}
r = b;
```

Protokoll (Speicherbelegungsänderung)

	m	*n*	*r*	*a*	*b*	*c*
Eingabe	**130**	**55**				
Verarbeitung				130	55	20
				55	20	15
				20	15	5
				15	5	0
Ausgabe			**5**			

Weitere Beispiele

Programm ***f = fak(n)***

```
a = n;
f = 1;
while( a != 1) { f = f * a; a = a - 1;}
```

Protokoll (Speicherbelegungsänderung)

	n	*f*	*a*
Eingabe	**3**		
Verarbeitung		1	3
		3	2
		6	1
Ausgabe		**6**	

Programm ***s = summeGauss(n)***

```
r = n % 2;
if( r == 0) { g = n; u = n + 1;} else { g = n + 1; u = n;}
s = g / 2; s = s * u;
```

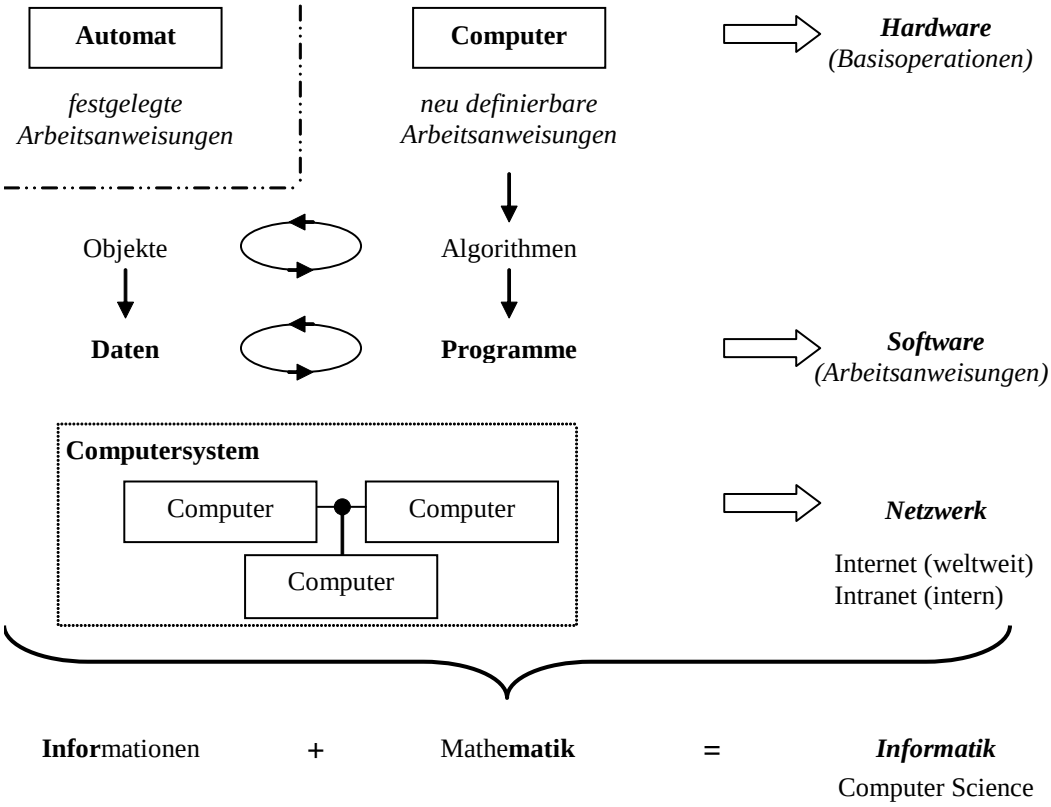
Wer führt die Arbeitsanweisungen aus?

Die konkrete Ausführung eines Algorithmus heißt **Prozess**.
Die Einheit, die den Prozess in der Reihenfolge der *Arbeitsanweisungen* ausführt, heißt **Prozessor**. Er beherrscht die *Basisoperationen*.
Ein **Zustand** eines Prozesses zu einem bestimmten Zeitpunkt ist der Stand der Ausführung zu diesem Zeitpunkt.

Problem	**Algorithmus**	**Basisoperationen**	**Prozessor**
Pullover stricken	Strickmuster	linke Maschen, rechte Maschen, ...	Strickmaschine
Lied singen	Notenblatt	Ton für jede Note (Tonhöhe und Tonlänge)	Spielorgel
Rechenaufgabe lösen	Formel	Grundrechenarten, ...	Rechner

computare (lat.: = rechnen) ⇒ Ein *Computer* ist ein Rechner, ein spezieller Prozessor.

Was unterscheidet einen Computer von einem Automaten?



Ein **Automat**, wie ein Getränkeautomat, kann nur *festgelegte Arbeitsanweisungen* ausführen. Im Unterschied dazu kann einem **Computer** die Vorschrift, nach der er arbeiten soll, jeweils *neu* vorgegeben werden.
So eine *Handlungsvorschrift* in Form eines Algorithmus *manipuliert* **Objekte** der realen Welt. Um dem Computer einen Algorithmus mitzuteilen, muss man diesen für ihn verständlich als **Programm** formulieren. Die zu manipulierenden Objekte müssen in ihm als **Daten** abgelegt werden.
Ein Computer setzt sich aus dem materiellen Teil, der **Hardware**, welche die *Basisoperationen* festlegt, und dem immateriellen Teil, der **Software**, welche es ermöglicht, *Arbeitsanweisungen* neu zu definieren, zusammen. Zur Software (*Systemsoftware*, *Anwendungssoftware*) gehören nicht nur die *Programme*, sondern auch die von diesen zu verarbeitenden *Daten*.
Computer können über Datenleitungen oder per Funk miteinander **vernetzt** sein, d.h. sie können untereinander Informationen austauschen. Man spricht dann von einem **Netzwerk**. **Intranet** ist die Vernetzung *innerhalb* einer Institution, **Internet** ist eine *weltweite* Vernetzung. Mehrere miteinander vernetzte Computer nennt man auch **Computersystem**.

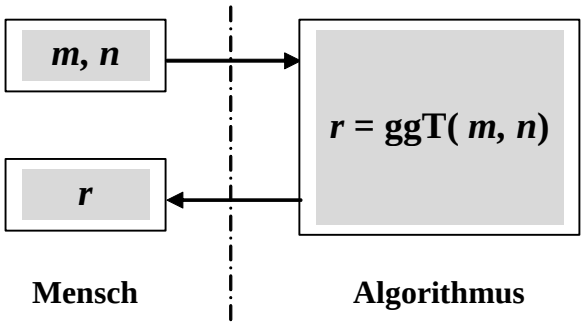
Informatik (***Infor***mation + Mathe***matik***, engl. **Computer Science**) ist seit 1960 eine eigenständige Wissenschaft. Sie beschäftigt sich mit der *theoretischen Analyse und Konzeption*, aber auch mit der *konkreten Realisierung von Computersystemen* in den Bereichen der *Hardware*, der *Software*, der *Organisationsstruktur* und der *Anwendung*.

Inhalt des Kurses ist die Programmierung solcher Computersysteme. Deshalb haben wir uns zunächst mit dem Algorithmenbegriff auseinandergesetzt.

3.3 Hardware – Der Körper eines Rechners

3.3.1 Aufbau eines Rechners

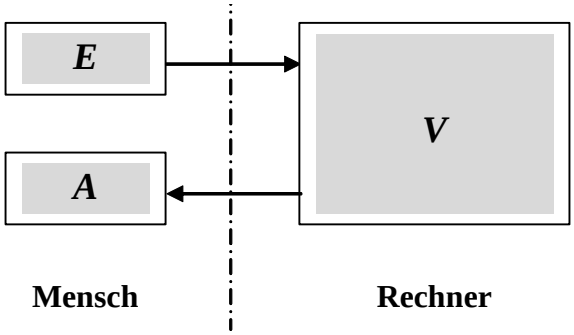
Dem euklidischen Algorithmus werden die beiden Startwerte ***m*** und ***n*** übergeben, nach seiner Ausführung liefert er das Ergebnis ***r*** zurück:



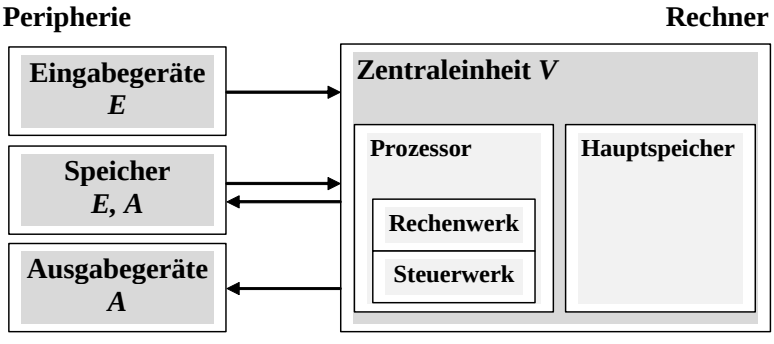
Ein Rechner soll mit gegebenen Werten einen Algorithmus ausführen und ein Ergebnis zurückgeben.

Wie muss er für diese Aufgabe aufgebaut sein?
Jeder Rechner ist ein ***informationsverarbeitendes System*** und arbeitet nach dem ***EVA-Prinzip***:

Eingabe von ***Informationen***
Verarbeitung von ***Informationen*** entsprechend vorgegebener Arbeitsanweisungen
Ausgabe von ***Informationen***



Hauptkomponenten des Aufbaus eines Rechners



J. von Neumann – Architektur, 1947

Zentraleinheit (intern):

Führt die **Arbeitsanweisungen** mit gegebenen **Basisoperationen** aus.

Prozessor (CPU – central processing unit)

- ***Rechenwerk*** (ALU – arithmetical and logical unit):
  Basisoperationen des Rechners, durch Schaltungen realisiert.
- ***Steuerwerk*** (CU – control unit)
  Koordinations-, Kontroll-, Organisationsaufgaben; übergibt die aktuelle *Arbeitsanweisung* an das Rechenwerk.

Hauptspeicher (CM – central memory, auch ***Arbeitsspeicher***)

Lädt *kurzfristig* Teile des Betriebssystems, Teile der *Arbeitsanweisungen* und die zu verarbeitenden *Daten* und stellt diese dem Prozessor zur Verfügung; schneller Zugriff.

- ***Nur-Lese-Speicher*** ROM (read only memory)
- ***Lese-Schreib-Speicher*** RAM (random access memory)

Peripherie (extern):

Speicher: Enthält das Betriebssystem, Arbeitsanweisungen des Algorithmus und zu verarbeitenden Daten; langsamer Zugriff, dient der längerfristigen Aufbewahrung.

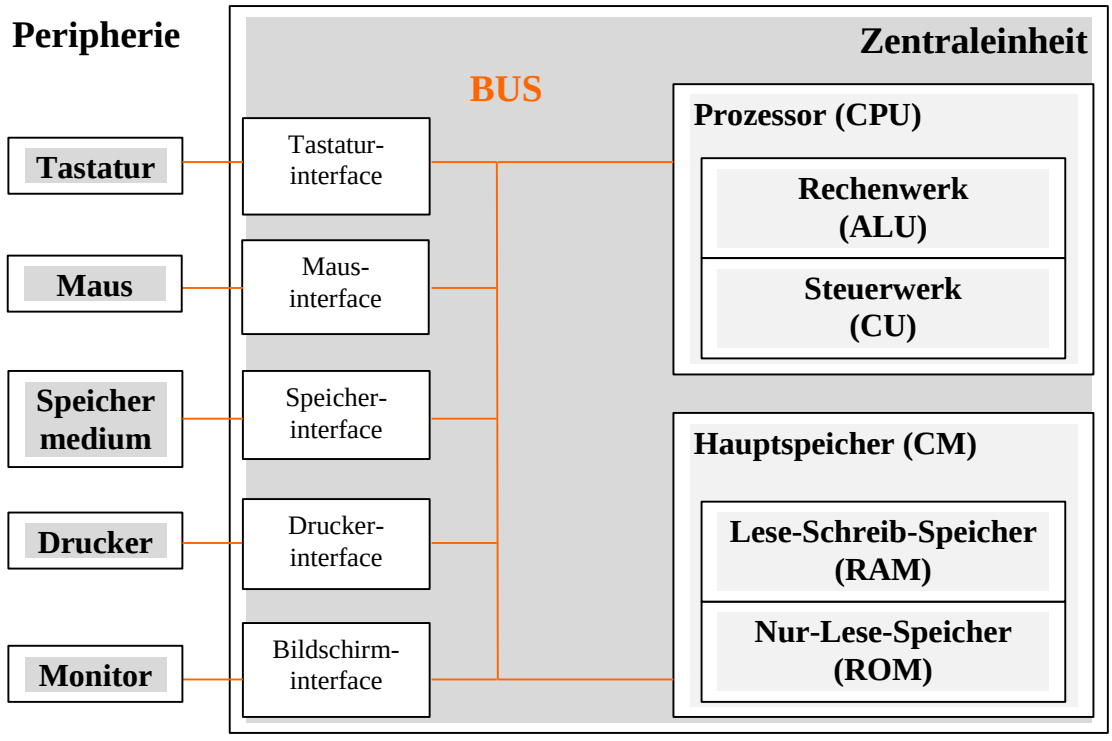
Speichermedien: Magnetband, Festplatte, Diskette, CD, DVD, Memorystick, ...

Ein-/Ausgabegeräte:

Kommunikation mit der Außenwelt, Übermittlung von Informationen zwischen Zentraleinheit und Nutzer.

Medien: Tastatur, Maus, ...; Monitor, Drucker, ...

Aufbau eines Personalcomputers (PC seit 1975)



Der Datenaustausch zwischen den Einheiten erfolgt über einen ***Bus***. Je nach Anzahl der gleichzeitig übertragbaren Bits („Sitzplätze“) unterscheidet man eine ***Bus-Breite*** von 4 (1971), 8, 16, 32 und 64 Bits.

3.3.2 Schaltalgebra

Wie wir in den obigen Abschnitten gesehen haben, wird ein Teil der Basisoperationen im Computer auf die Addition von Zahlen in Dualdarstellung zurückgeführt. Der Computer führt diese „Berechnungen“ mit Hilfe elektronischer Schaltungen durch.

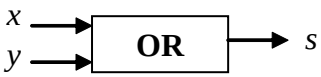
Die Theorie dazu, die ***Schaltalgebra***, baut auf die Gesetze der ***Booleschen Algebra*** (Mengenoperationen und ihre Analogie zur Aussagenlogik) auf, welche **1854** durch den englischen Mathematiker Georg Boole (1815 - 1864) veröffentlicht wurde.

Die Schaltungen im Computer beruhen auf **drei Grundschaltungen**, deren ***Schaltverhalten*** Funktionen der Aussagenlogik entsprechen, wobei die Werte wahr bzw. falsch hier wie folgt interpretiert werden:

1... Signal liegt an (wahr)

0... Signal liegt nicht an (falsch)

Diese ***logischen Schaltungen*** sind:

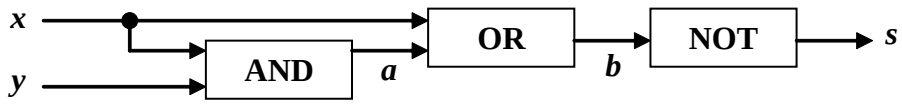
	Schaltung	logischer Ausdruck	
NOT – Schaltung (Negation)		$s = \bar{x}$	(nicht x)
OR – Schaltung (Alternative)		$s = x \vee y$	(x oder y)
AND – Schaltung (Konjunktion)		$s = x \wedge y$	(x und y)

Die folgende Tabelle zeigt das ***Schaltverhalten*** der drei Grundschaltungen in Analogie zu den logischen Funktionen:

x	y	**NOT** $\bar{x}$	**OR** $x \vee y$	**AND** $x \wedge y$
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

Der physikalische Aufbau dieser Grundschaltungen hat sich in den Jahren der Entwicklung der EDV wesentlich verändert. Ausgehend von dem *Röhrenaufbau* (1. Generation) über die Anwendung von *Transistoren* (2. Generation) bis hin zu den *Mikrochips* (3. und 4. Generation) sind die Grundschaltungen als solche geblieben.

Beispiel
Gegeben sei die folgende Schaltung:



Schaltverhalten
Ablesen des Schaltverhaltens bei gegebenen Eingangssignalen:

Wenn an den Eingängen *x* und *y* kein Signal anliegt, so liegt am Ausgang *a* kein Signal ($0 \wedge 0$), am Ausgang *b* kein Signal ($0 \vee 0$) und schließlich am Ausgang *s* ein Signal ($\overline{0}$) an.

Eine vollständige Beschreibung des Schaltverhaltens erhält man mit Hilfe einer **Wertetabelle**:

x	y	$a = x \wedge y$	$b = x \vee a$	$s = \overline{b}$
0	**0**	0	0	**1**
0	**1**	0	0	**1**
1	**0**	0	1	**0**
1	**1**	1	1	**0**

Die Schaltung repräsentiert den **logischen Ausdruck** $s = \overline{b} = \overline{x \vee a} = \overline{x \vee (x \wedge y)}$.

Aus der Tabelle wird ersichtlich, dass das Ausgangssignal *s* unabhängig von *y* nur vom Eingangssignal *x* beeinflusst wird, d.h. das Eingangssignal *y* ist überflüssig. Die folgende Schaltung zeigt das gleiche Schaltverhalten.

Beide Schaltungen sind *äquivalent*. Die letztere Schaltung ist aber optimaler, da weniger Bauelemente verbraucht werden.

Zu diesem Ergebnis kommt man auch durch geeignetes Umformen des logischen Ausdrucks.

$$s = \overline{x \vee (x \wedge y)} = \overline{x} \wedge \overline{(x \wedge y)} = \overline{x} \wedge (\overline{x} \vee \overline{y}) = \overline{x}$$

3.3.3 Volladdierer

Im Kapitel 2 haben wir festgestellt, dass ein Rechner nur addieren muss. Die Addition zweier natürlicher Zahlen setzt sich aus der Addition ihrer Ziffern gleicher Position zusammen. Außerdem ist noch ein evtl. vorhandener Übertrag zu berücksichtigen.

Betrachten wir nochmals eine Additionsaufgabe für natürliche Zahlen im Dualsystem:

$(110)_2$	$(a_n \cdots a_1 a_0)_2$
$+ (11)_2$	$+ (b_n \cdots b_1 b_0)_2$
Ü 1100	$\ddot{u}_{n+1}\ddot{u}_n \cdots \ddot{u}_1\ddot{u}_0$
$(1001)_2$	$(s_{n+1}s_n \cdots s_1 s_0)_2$

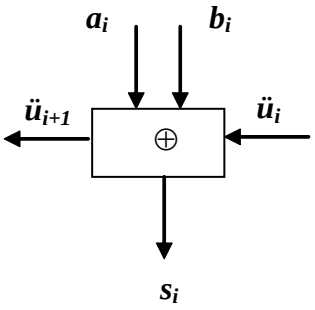
Additionstafel

+	**0**	**1**
0	0	1
1	1	10

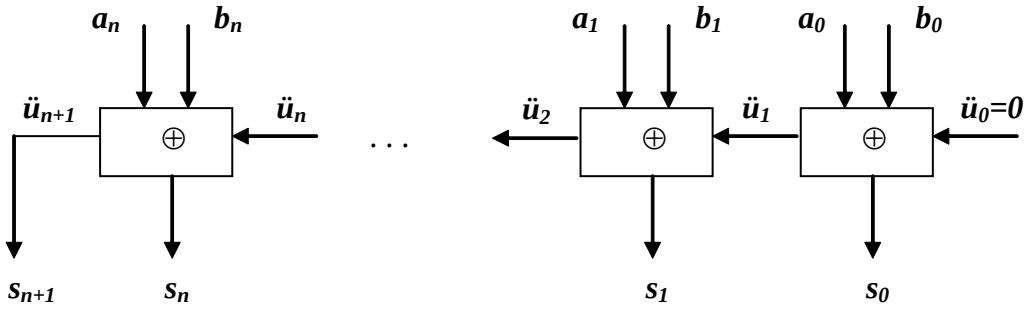
Der Additionsalgorithmus $(a_n \cdots a_1 a_0)_2 + (b_n \cdots b_1 b_0)_2 = (s_{n+1}s_n \cdots s_1 s_0)_2$ umfasst die folgenden drei Schritte:

1. Setze anfangs $\ddot{u}_0 = 0$.
2. Für jede Stelle $i = 0, 1, ..., n$ der Zahlen a und b gilt:
   Die $i.$ Stellen a_i und b_i der beiden Zahlen a und b werden mit dem $i.$ Übertrag $\ddot{u}_i$ addiert. Man erhält die $i.$ Stelle s_i des Ergebnisses s und den ($i+1.$) Übertrag $\ddot{u}_{i+1}$.
3. Abschließend wird noch $s_{n+1} = \ddot{u}_{n+1}$ gesetzt.

Benötigt wird eine Schaltung für die Addition zweier *Ziffern* mit Übertrag, als **Baustein** $\oplus$ realisiert. Diese Schaltung hat demnach drei Eingänge und zwei Ausgänge.

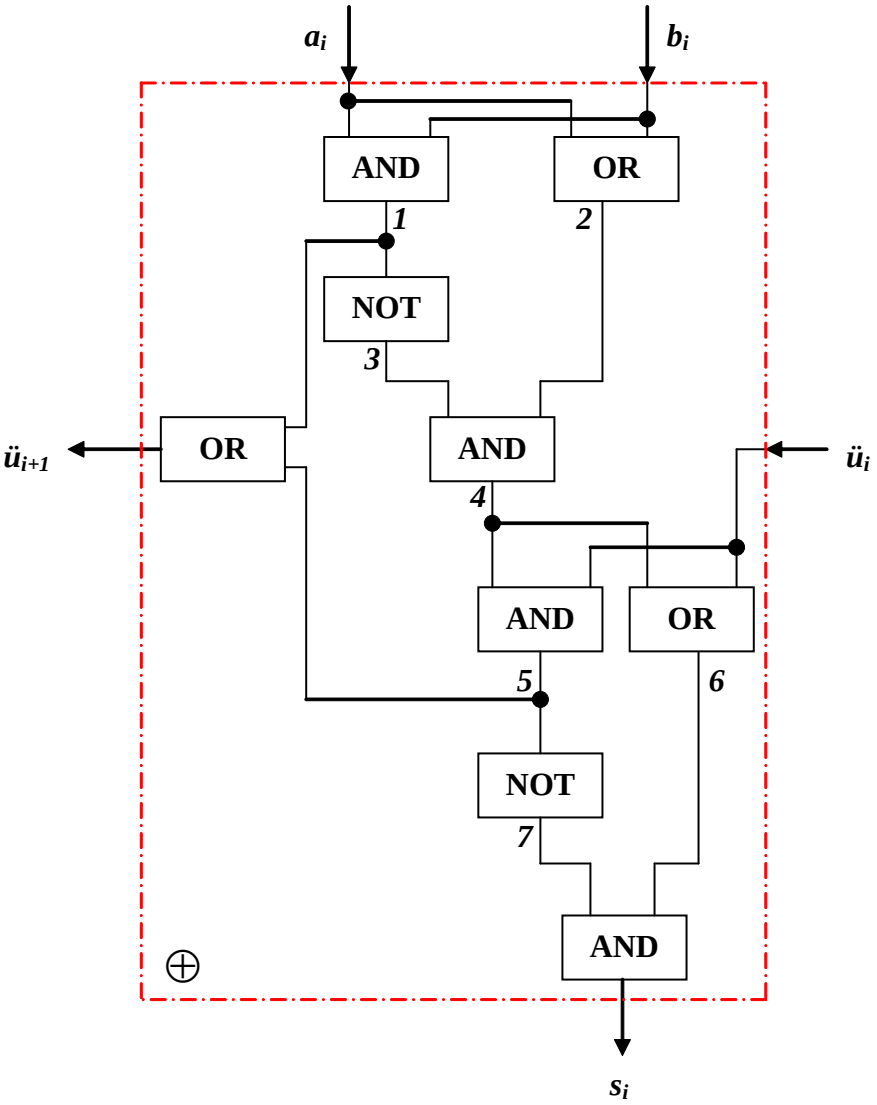


Die Addition zweier *Zahlen* kann dann mittels dieses Bausteins durch mehrmaliges Hintereinanderschalten aufgebaut werden. Allgemein ergibt sich folgendes Schema:



Der Baustein heißt ***Volladdierer*** und ist aus den drei Grundschaltungen aufgebaut:

Volladdierer



Schaltverhalten des Volladdierers:

a_i	b_i	$ü_i$	1	2	3	4	5	6	7	$ü_{i+1}$	s_i
0	0	0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	1
0	1	0	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1	0
1	0	0	0	1	1	1	0	1	1	0	1
1	0	1	0	1	1	1	1	1	0	1	0
1	1	0	1	1	0	0	0	0	1	1	0
1	1	1	1	1	0	0	0	1	1	1	1

Ein Vergleich mit der Additionstafel ergibt, dass der Volladdierer die Addition zweier Ziffern mit Übertrag realisiert.

Mit dieser Schaltung kann man *alle* Operationen aufbauen, welche auf eine *Addition* zurückzuführen sind.

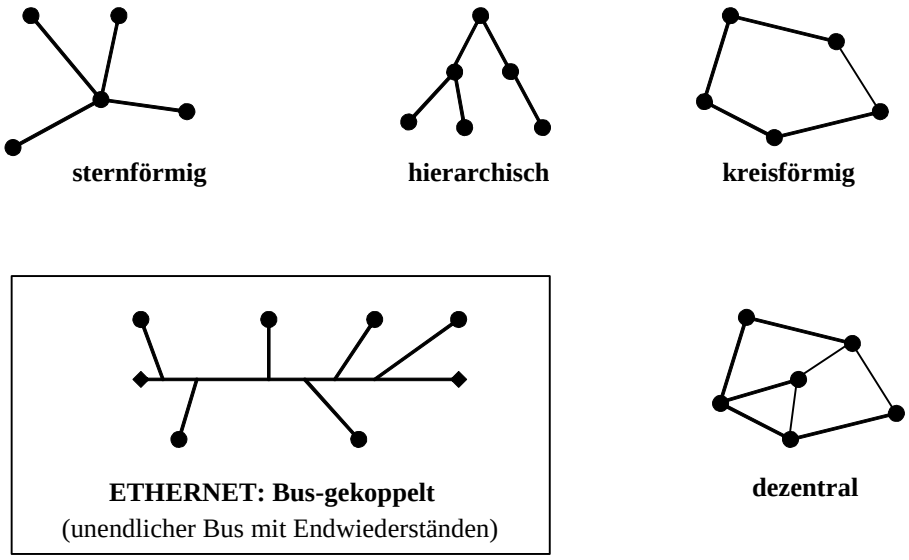
3.4 Rechnernetze

Die Kopplung oder auch Vernetzung mehrerer Rechner, die in ihrer Architektur und/oder ihrem Betriebssystem unterschiedlich sein können und i. a. räumlich getrennt stehen, nennt man ein ***Rechnernetz*** oder auch ein ***Rechnerpool*** (5. Generation). Rechnernetze werden für folgende Dienste konzipiert:

- ***Datenverbund***
  Gemeinsame Nutzung von Daten, die auf einzelnen Rechnern dezentral bereitstehen.
  Beispiel Kopplung von Bibliotheksrechnern.
- ***Betriebsmittelverbund***
  Gemeinsame Nutzung von Hard- oder/und Software, die einzelnen Rechnern zentral zur Verfügung gestellt wird.
  Beispiel Druckerrechner, Verwaltungsrechner.
- ***Lastverbund***
  Gleichmäßige Verteilung der benötigten Rechenleistung auf die im Netz angeschlossenen Rechner. Fällt ein Rechner aus, so übernimmt ein anderer dessen Aufgaben.
  Beispiel Raumfahrt.

Die Rechner können innerhalb des Rechnernetzes unterschiedlich angeordnet und verbunden werden. Man unterscheidet:

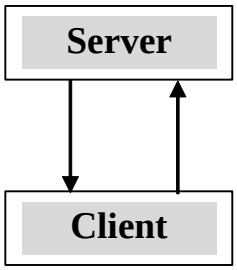
- ***Intranet*** Vernetzung innerhalb einer Firma
- ***Internet*** Vernetzung weltweit



Netztopologie von Rechnernetzen

Rechner im Netz bieten anderen Rechnern ***Dienste*** an bzw. nutzen angebotene Dienste. So ist es möglich, Dienste auf wenige Rechner zu installieren und mehreren Rechnern gleichzeitig zur Verfügung zu stellen. Die *Wartung* und *Pflege* dieser Dienste wird durch die Konzentration auf wenige Rechner vereinfacht.

Client - Server - Architektur



Server: Ein Rechner, der eine Reihe von Diensten (Service) bereitstellt und diese auf Wunsch ausführt.
Client: Ein Rechner, der die Dienste anderer Rechner in Anspruch nimmt.

Ein Rechner kann sowohl Server als auch Client sein.

Beispiele
In einem Rechnerpool erledigen ***Poolserver*** Verwaltungsaufgaben, wie

- Verwaltung von Software (Bereitstellen von Software für alle Rechner im Pool, Wartung dieser, ...)
- Verwaltung der Nutzer (Einrichten von Nutzern und Verwalten deren Verzeichnisse, regelmäßige Sicherung der Daten, Zugriffsrechte, ...).

Poolrechner sind die Clients, die die Kapazitäten des Poolservers nutzen. Deshalb sollte hier der Server technisch sehr gut ausgerüstet sein, während die Clients weniger Kapazitäten brauchen.

Ein ***Druckserver*** dient der Verwaltung von Druckaufträgen. Er gestattet es, von mehreren Rechnern aus, auf einen oder mehrere gemeinsam genutzte *Drucke*r zuzugreifen. Er erledigt meist nur diese Aufgabe. Man stellt deshalb wenige Ansprüche an dessen Ausstattung.

Auf einem ***Webserver*** werden Internet-Auftritte verwaltet. Dort werden Homepages abgelegt, dieser gestattet den Zugriff über das *Internet*. Das Internet ist ein weltweites Netzwerk, welches aus einer riesigen ständig wachsender Anzahl von Rechnern (z. Z. ca. 250 Millionen, 18 Millionen in D) besteht. Diese sind meist mit Standleitungen oder über Funk verbunden, die einen Datenaustausch untereinander ermöglichen.

Clients sind die Nutzer dieser Dienste. Zum Beispiel ein ***Poolrechner***, der von einem Server Software verwendet oder der einen Druck auslöst, aber auch ein Rechner, der mit Hilfe eines Browsers im Internet auf Homepages zugreift.

Laufwerkbezeichnungen im Netz:

A:	(Diskette)	***lokaler (naher) Rechner***
B:	(Diskette)	
C:	Festplatte	
D:	(Festplatte, CD, DVD, USB, ...)	
E:	(Festplatte, CD, DVD, USB, ...)	
	.	
	.	
X:		***ferner Rechner***
Y:		
Z:		