3.Vorlesung Grundlagen der Informatik

Christian Baun

Hochschule Darmstadt
Fachbereich Informatik
christian.baun@h-da.de

27.10.2011

Wiederholung vom letzten Mal

- Duales Rechnen

Heute

- Generationen von Computersystemen
- Boolesche Algebra

Generationen von Computersystemen

- Die **Geschichte der Informatik** wird in verschiedene Stationen unterteilt
- Maßgeblich sind dabei die zu dieser Zeit verfügbaren Rechenmaschinen (Rechner/Computer) und deren Fähigkeiten
- Man unterscheidet folgende Generationen:
  - **0.Generation (bis 1940)**
    (Elektro-)mechanische Rechenmaschinen
  - **1.Generation (1940 bis 1955)**
    Elektronenröhren, Relais und Klinkenfelder
  - **2.Generation (1955 bis 1965)**
    Transistoren und Stapelverarbeitung
  - **3.Generation (1965 bis 1980)**
    Integrierte Schaltungen und Dialogbetrieb
  - **4.Generation (1980 bis 2000)**
    Hoch-integrierte Schaltungen, Mikroprozessoren, PCs/Workstations
  - **5.Generation (2000 bis ????)**
    Verteilte Systeme, *Das Netz ist der Computer*, Virtualisierung

0.Generation (bis 1940)

- Mechanische/elektromechanische Rechenmaschinen
- Bekannte Beispiele:
  - Rechenmaschine von Wilhelm Schickard (1623)
    - Rechenmaschine für Addition und Subtraktion mit Zehnerübertragung
  - Rechenmaschine Pascaline von Blaise Pascal (1643)
    - Rechenmaschine für Addition und Subtraktion und bis zu achtstellige Zahlen mit Zehnerübertragung
  - Leibniz'sche Rechenmaschine von Gottfried Wilhelm Leibniz (1673)
    - Rechenmaschine für alle vier Grundrechenarten und bis zu sechsstelligen Zahlen. Vollständige Zehnerübertragung
  - Differenzmaschine Nr.1 zum Lösen polynomialer Funktionen von Charles Babbage (1832)
  - Hollerithmaschine von Herman Hollerith (1888)
    - Tabelliermaschine, Lochkartensortierer, Lochkartenlocher und Lochkartenleser
    - Einsatz 1890 bei der amerikanischen Volkszählung
    - Die Firma von Hollerith wurde 1924 zur International Business Machines Corporation (IBM) umbenannt

1.Generation (1940 bis 1955)

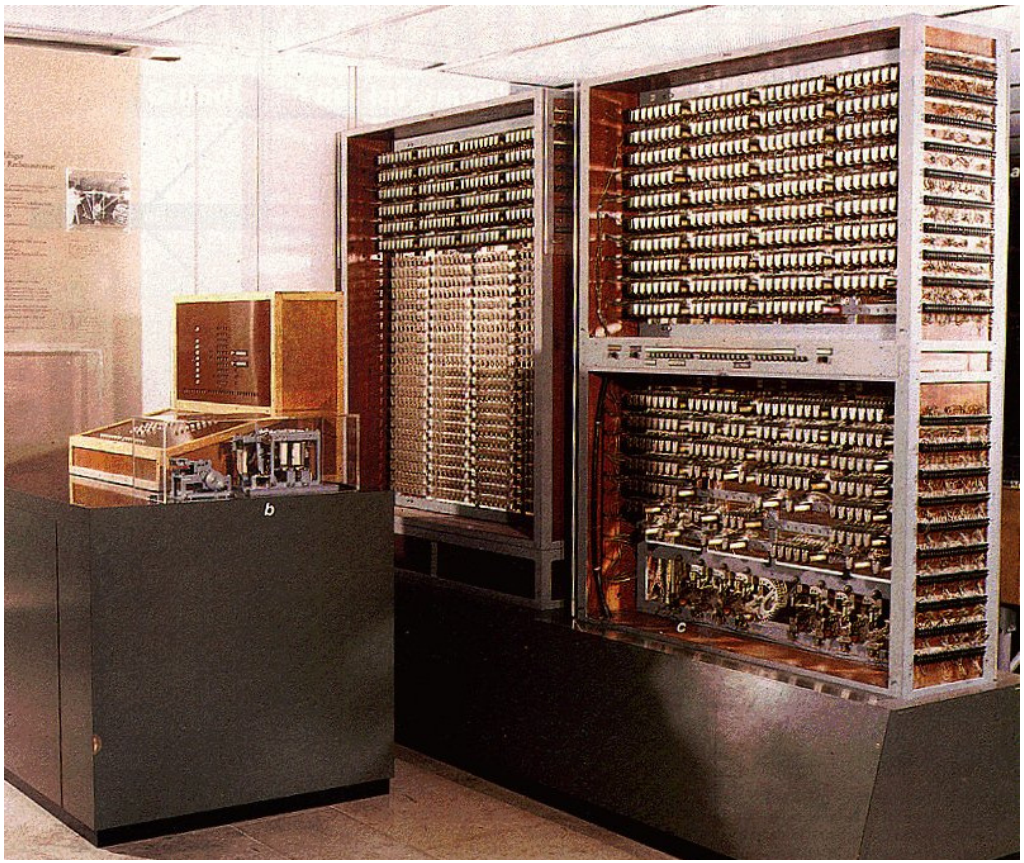
- Die Erfindung des Computers ist eine kollektive Errungenschaft über ein Jahrzehnt und zwei Kontinente
- Die erste Generation von Computersystemen entstand gegen Ende des 2.Weltkriegs $\Longrightarrow$ Konrad Zuse, John von Neumann
- Anforderungen an einen universellen Computer:
  - Gespeichertes Programm
  - Bedingte Sprünge
  - Trennung von Speicher und Prozessor
- Rechner waren Maschinen mit teilweise über 10.000 Röhren oder Relais, die langsam und fehleranfällig arbeiteten
- Betriebssysteme und Programmiersprachen waren unbekannt
  - Der Benutzer/Programmierer startet **ein** Programm, dass direkt auf die Hardware zugreift
- Programme wurden über Klinkenfelder gesteckt

Bekannte Vertreter der 1.Generation

Maschine	Entwicklung	Speicher/CPU getrennt	bedingte Sprünge	Program-mierung	interne Kodierung	Zahlen-darstellung
Z1 / Z3	1936-1941	ja	nein	SW	binär	Gleitkomma
ABC	1938-1942	ja	nein	HW	binär	Festkomma
Harvard Mark 1	1939-1944	nein	nein	SW	dezimal	Festkomma
ENIAC	1943-1945	nein	teilweise	HW	dezimal	Festkomma
Manchester Mark 1	1946-1948	ja	ja	SW	binär	Festkomma
EDSAC	1946-1948	ja	ja	SW	binär	Festkomma

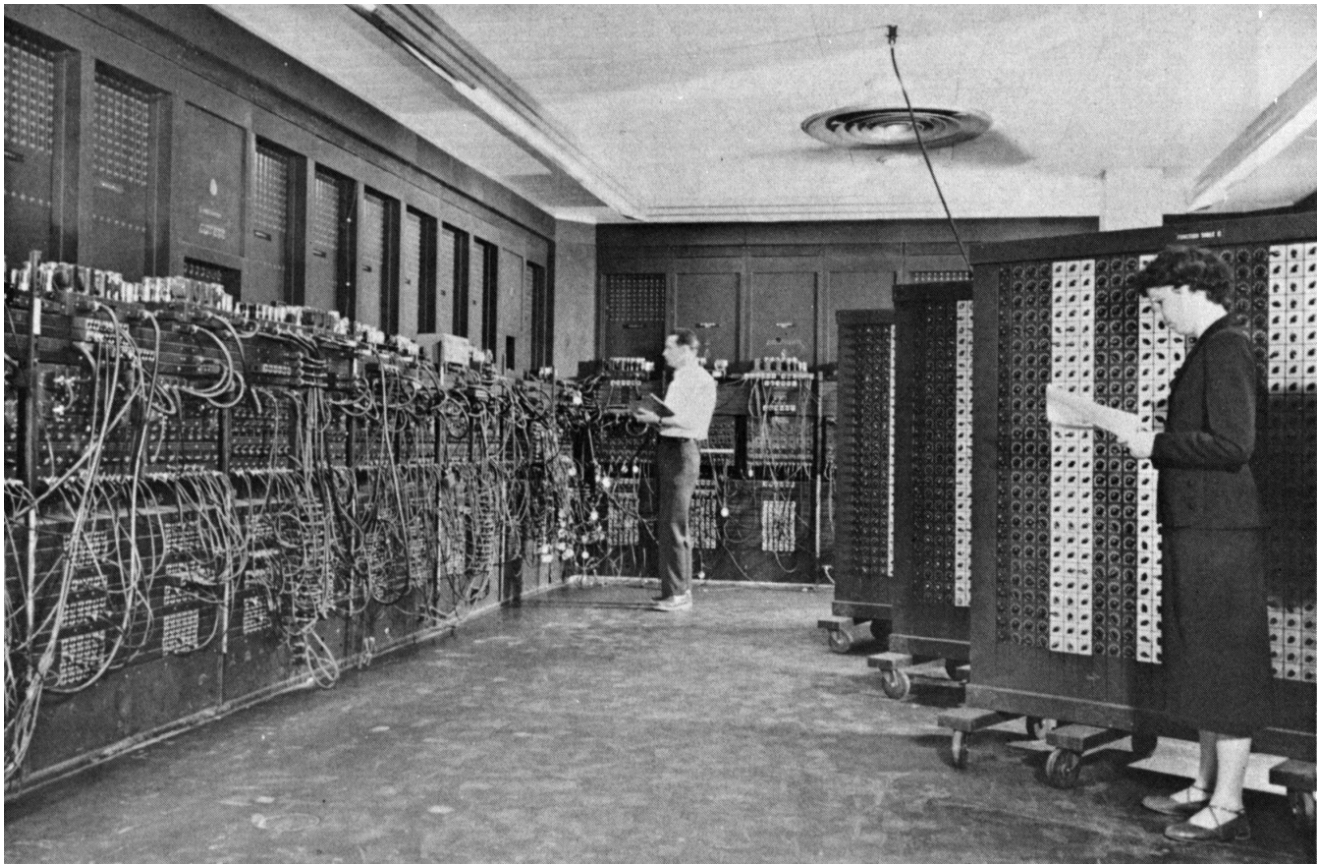
- Technologien
  - Mechanisch über Relais: Z1 und Z3
  - Elektronisch: Alle späteren

1.Generation: Zuse Z3 (1941)



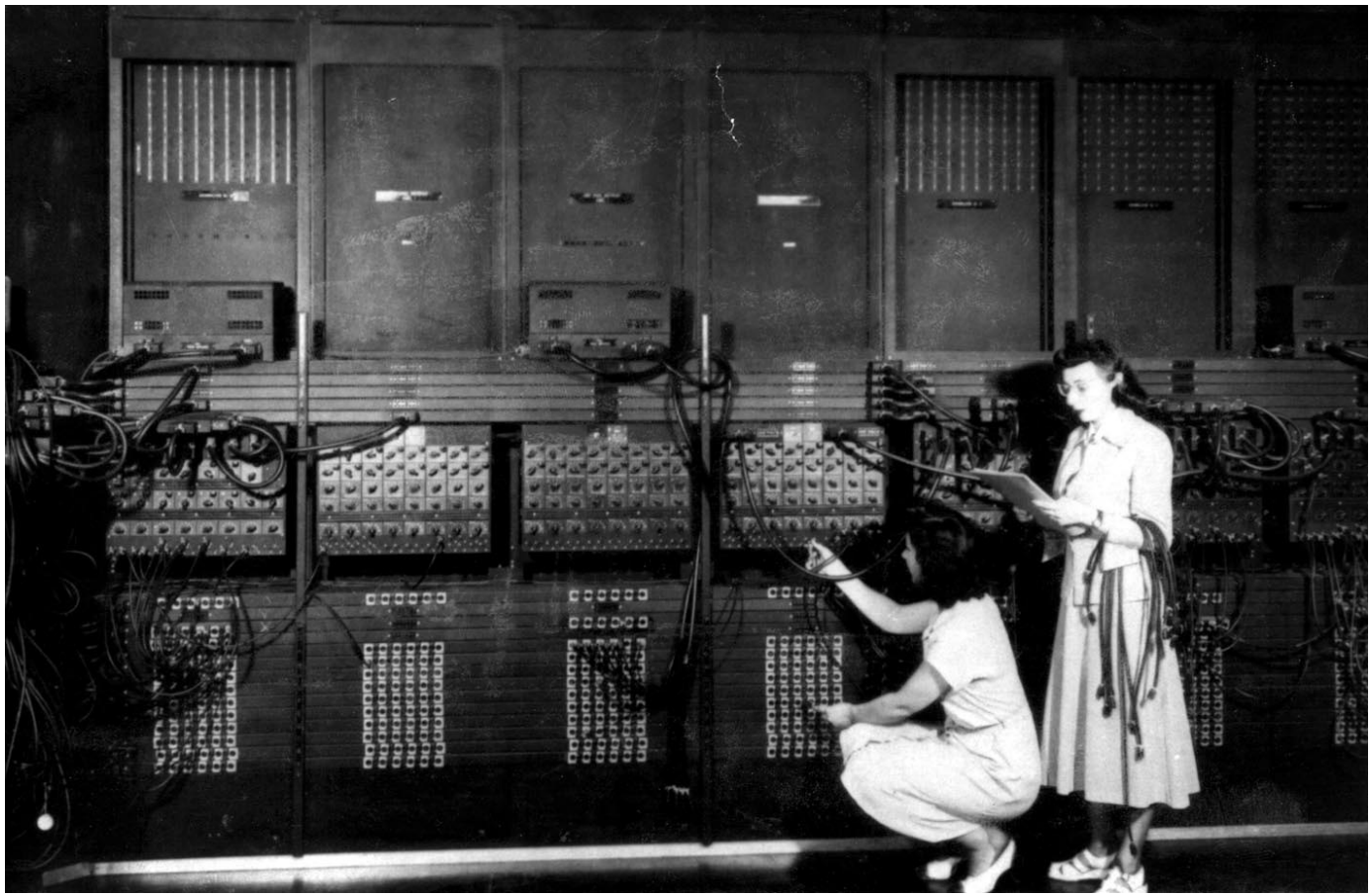
- Erster Computer der Welt (basiert auf Relaistechnik)
- Erstmals Verwendung des Dualsystems

1.Generation: ENIAC (1944) (1)



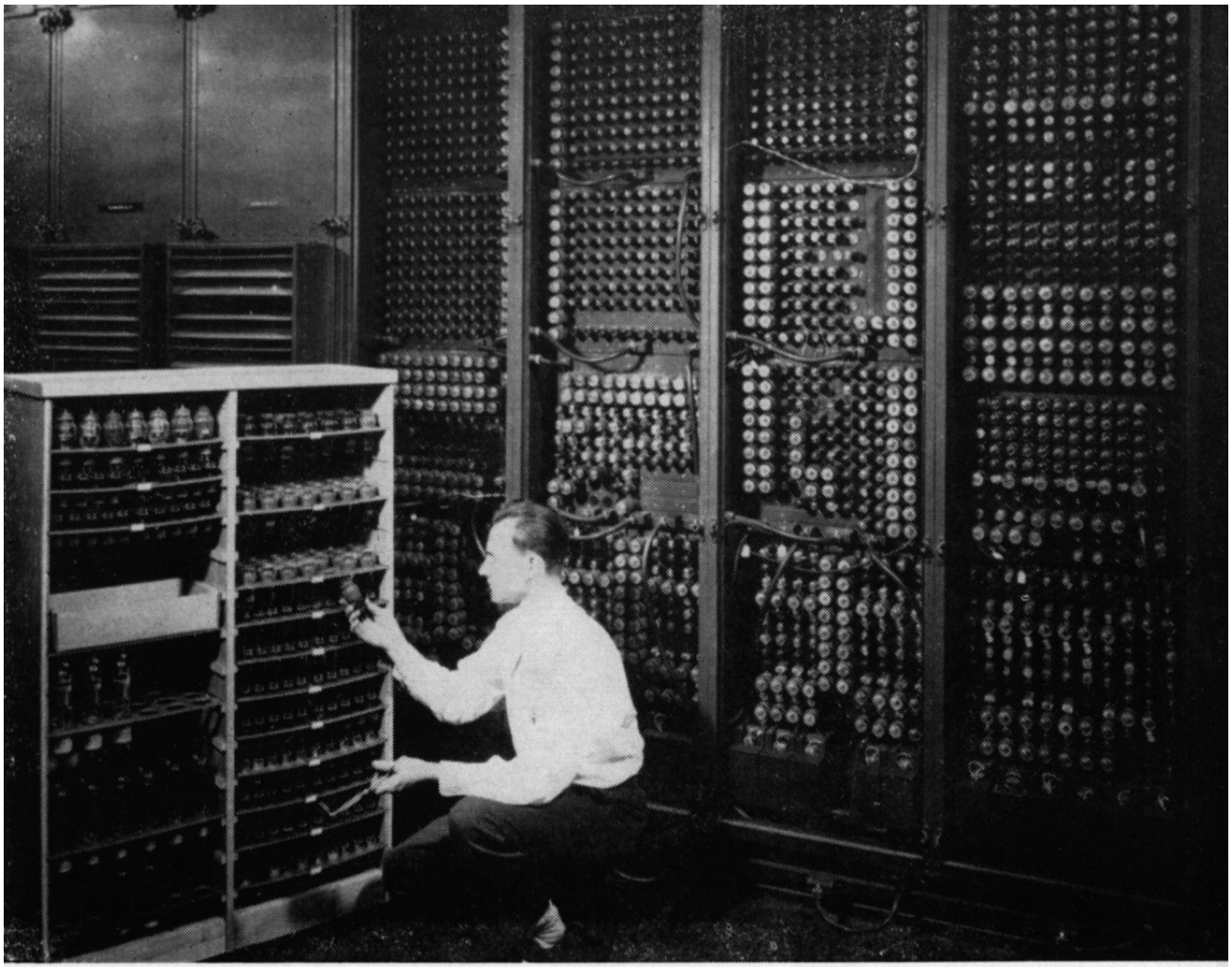
- Electronic Numerical Integrator and Computer (ENIAC)
- Erster elektronischer Universalrechner (mit Elektronenröhren)

1.Generation: ENIAC (1944) (2)



- 17.468 Röhren, 7.200 Dioden, 1.500 Relais, 70.000 Widerstände, 10.000 Kondensatoren

1.Generation: ENIAC (1944) (3)



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

2.Generation (1955 bis 1965)

- Einführung der Transistoren Mitte der 50er Jahre
  $\Longrightarrow$ Rechnersysteme wurden deutlich zuverlässiger
- Anfang der 50er Jahre wurde das Stecken der Klinkenfelder durch Lochkarten abgelöst
- FORTRAN- oder COBOL-Programme wurden:
  - vom Programmierer auf Formblätter aufgeschrieben,
  - vom Eingeber bzw. Codierer in Lochkarten gestanzt
  - und dem Operator übergeben
- Der Operator koordiniert die Reihenfolge der Programme (Jobs), bestückt den Rechner mit den entsprechenden Lochkarten, lädt den Compiler vom Magnetband, und übergibt abschließend das Rechenergebnis als Ausdruck
  $\Longrightarrow$ Sehr ineffiziente Arbeitsweise
- Später wurden aus Effizienzgründen die Programme gesammelt, auf Magnetbänder eingelesen und dann im Maschinenraum verarbeitet

Beispiel für die 2.Generation: IBM 7090 (1959) (3)

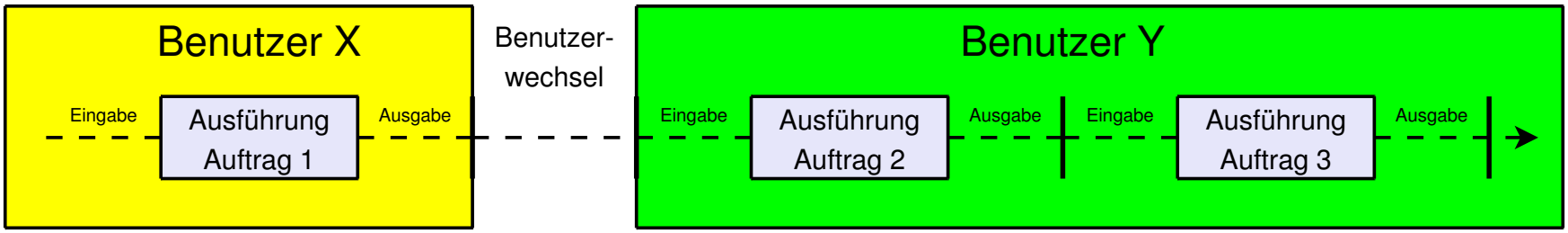
Bildquelle: `http://www.computer-history.info`

2.Generation: Stapelbetrieb bzw. Batchbetrieb (1)

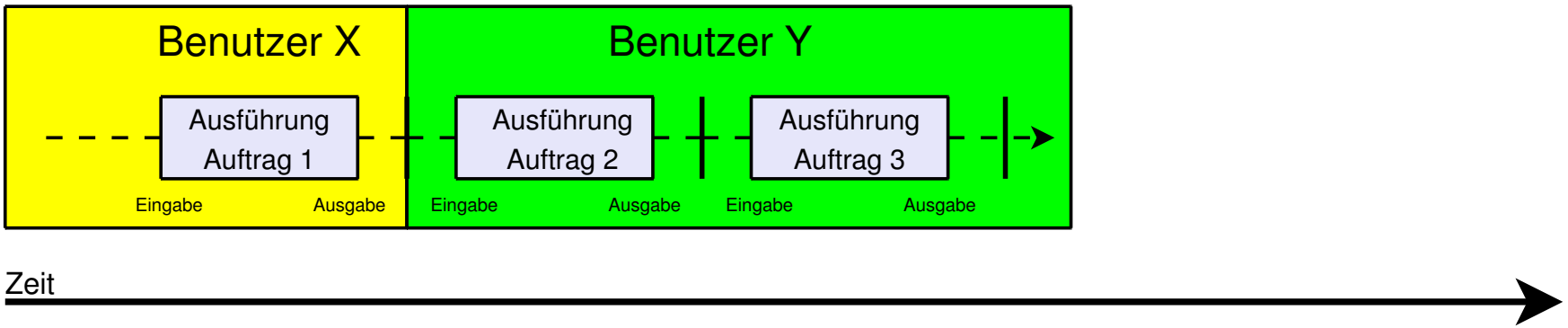
- Frühe Betriebssysteme waren **Stapelverarbeitungs-Betriebssysteme**
- Ziel: Maximierung der Prozessorausnutzung
- Bei Stapel- bzw. Batchbetrieb, muss eine Aufgabe aus einer Menge von Aufgaben vollständig gestellt sein, bevor ihre Abarbeitung beginnt
  - Üblicherweise interaktionslose Ausführung einer Folge von Aufträgen
- Programme wurden auf Lochkarten geschrieben und dem Operator gestapelt übergeben
  - Dieser bestücke den Rechner mit dem Stoß (*Batch*) von Lochkarten
- Stapelbetrieb eignet sich gut zur Ausführung von Routineauftragen
- Auch heutige Systeme besitzen die Möglichkeit, Progammabfolgen automatisch zu bearbeiten (z.B. Batch-Dateien, Shell-Skripte, usw.)
- Klassischer Stapelbetrieb ist **Einzelprogrammbetrieb** (**Singletasking**)
  - Betriebssystem erlaubt immer nur die Ausführung eines Programms
  - Start eines zweiten Programms geht erst nach Beendigung des Ersten

2.Generation: Stapelbetrieb bzw. Batchbetrieb (2)

Einbenutzerbetrieb mit Einzelauftragsbearbeitung (Single User Mode)

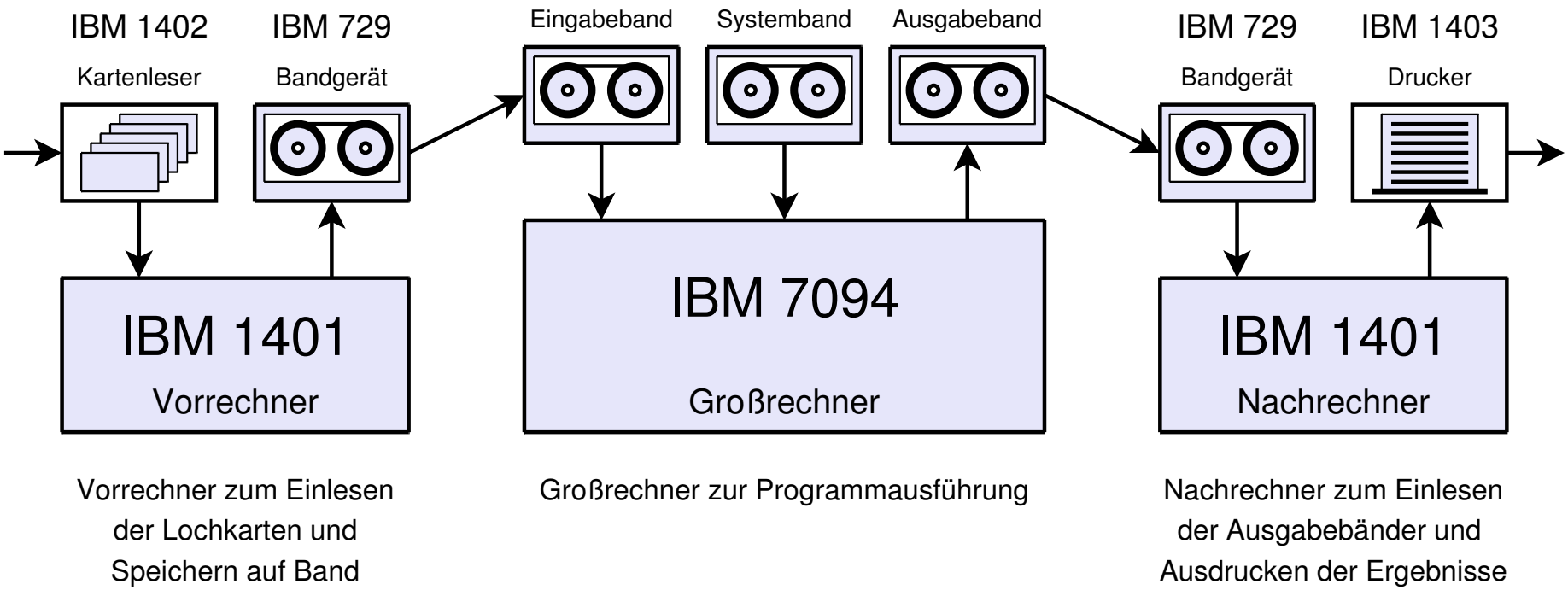


Stapelbetrieb (Batchbetrieb)



- Beschleunigung durch Automatisierung
- Problem: Beim Stapelbetrieb wird der Hauptprozessor nicht optimal ausgenutzt. Während der Ein-/Ausgabe liegt der Prozessor brach

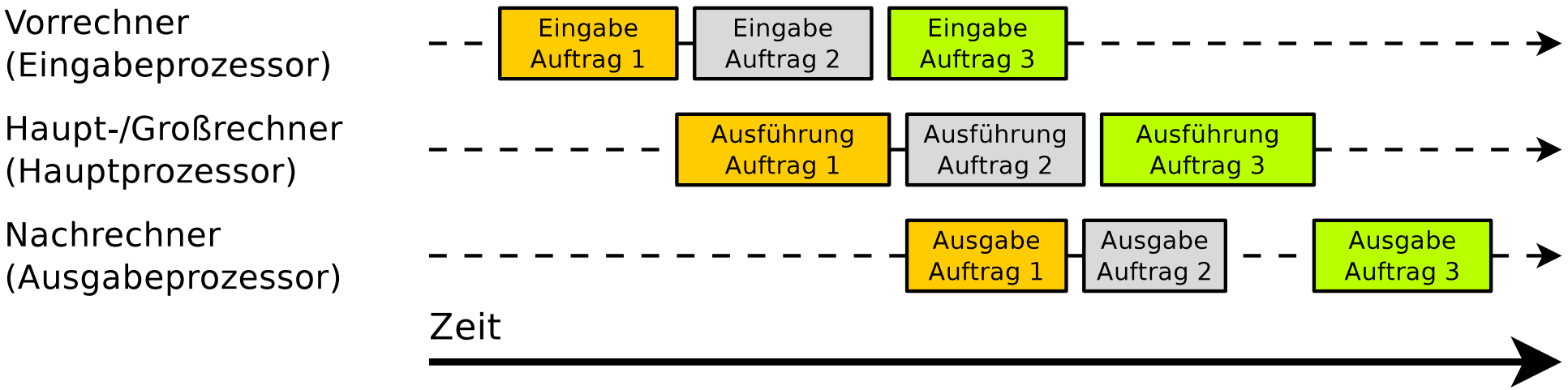
2.Generation: Stapelbetrieb bzw. Batchbetrieb (3)



- Vor-/Nachrechner befreien den Großrechner von langsamer I/O-Arbeit
- Von Band kann viel schneller eingelesen werden, als von Lochkarten und auf Band kann viel schneller Ausgegeben werden als auf Papier
- Ziel: Vermeidung ungenutzter Rechenleistung des Großrechners

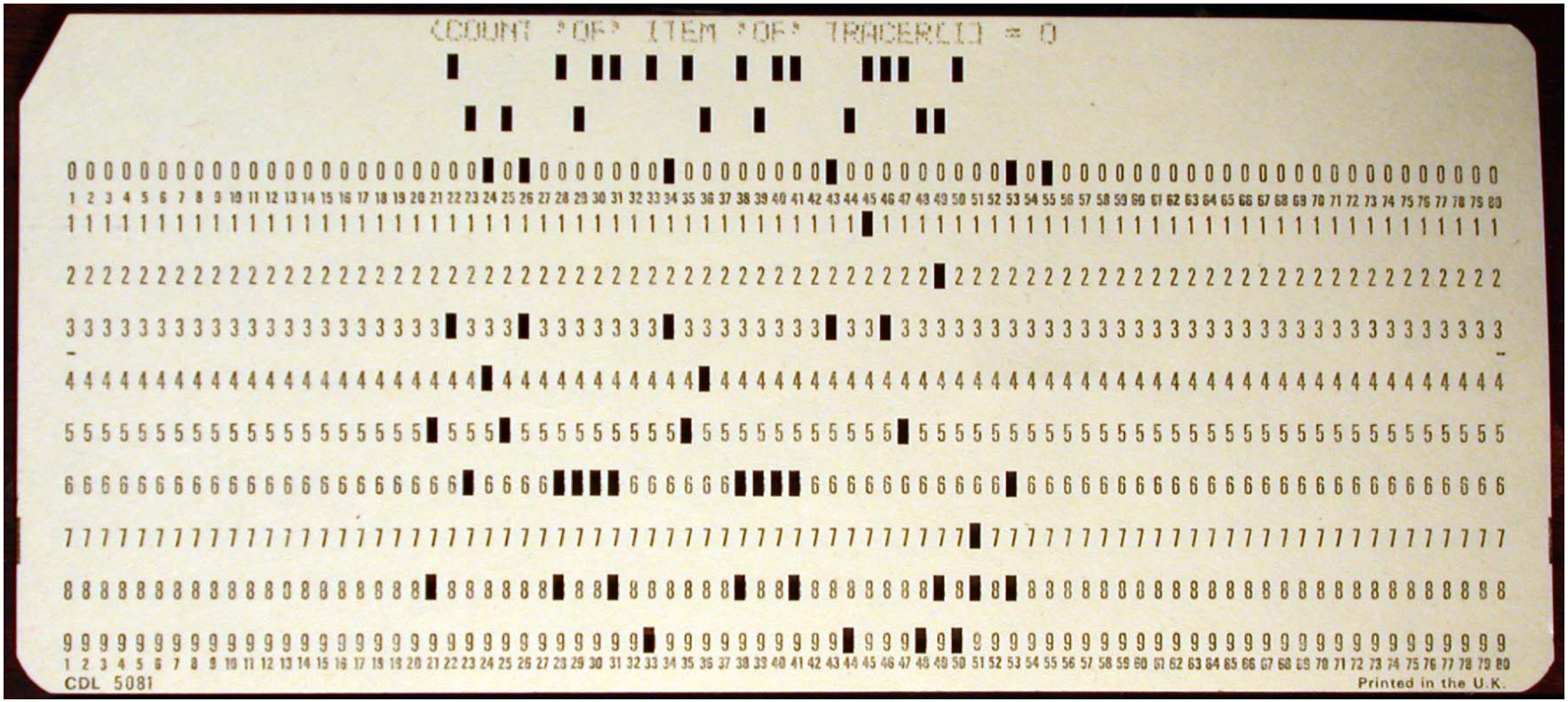
2.Generation: Stapelbetrieb bzw. Batchbetrieb (4)

Stapelbetrieb mit Vor- und Nachrechner (Spooling)



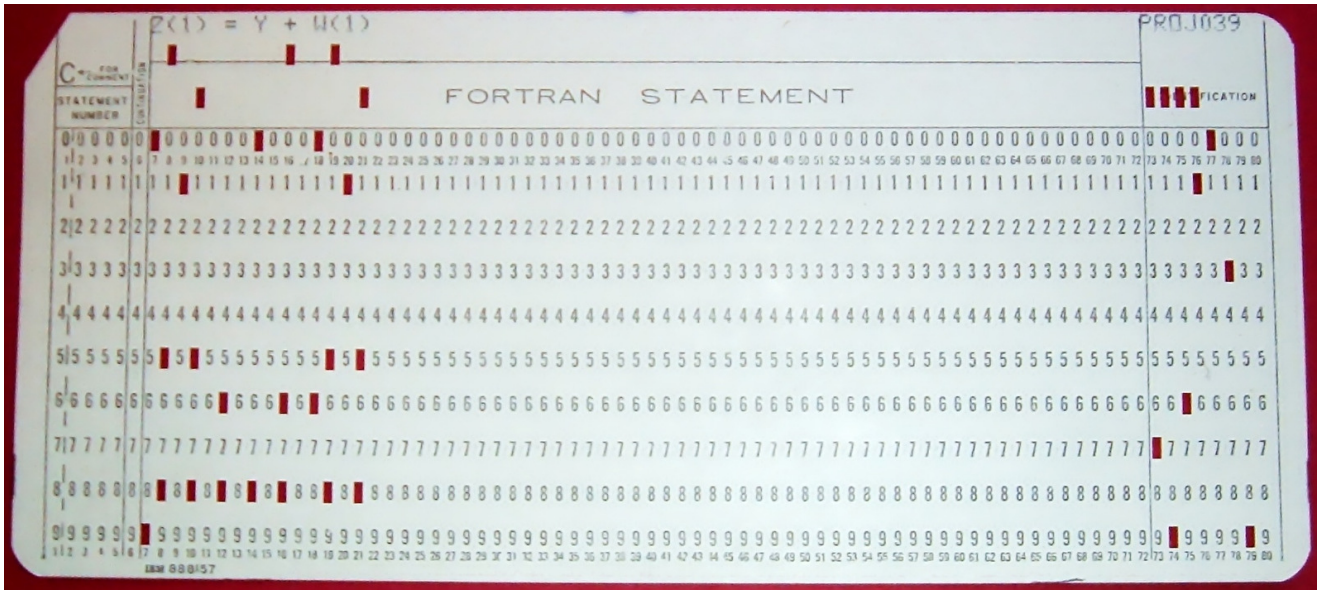
- **Spooling** ist die Entlastung des Hauptprozessors durch zusätzliche Hardware für Ein-/Ausgabeoperationen
  - Ein-/Ausgabe geschieht nebenläufig zur Bearbeitung anderer Aufträge
  - Heute haben Computer neben dem Hauptprozessor spezielle, DMA-fähige (*Direct Memory Access*) Ein-/Ausgabeprozessoren, die Aufträge direkt in den Hauptspeicher schreiben und Ergebnisse aus diesem holen
  - Spooling ist heute noch aktuell
    - z.B. Spoolingprozesse zum Drucken

2.Generation: Eine Lochkarte



- Jede Lochkarte stellt üblicherweise eine Zeile Programmtext mit 80 Zeichen oder entsprechend viele binäre Daten dar
- Das die Zeilenlänge von E-Mails und Textdateien heute noch typischerweise 80 Zeichen beträgt, geht auf die Lochkarte zurück

2.Generation: Noch eine Lochkarte (Bildquelle: Wikipedia)



- 12 Lochpositionen für die Kodierung jedes Zeichens
  - Ziffern kodiert man mit einem einzelnen Loch in der entsprechenden Zeile
  - Buchstaben und Sonderzeichen kodiert man, indem mehrere Löcher in die Spalte gestanzt werden

Stapelbetrieb ist heute nicht obsolet!

Rechenintensive Programme in verteilten Systemen sind häufig interaktionslose Batchprogramme
$\Longrightarrow$ Distributed Computing und sog. Number Crunching

3.Generation (1960 bis 1980)

- Frühe 60er Jahre: Integrierte Schaltungen setzen sich durch
  $\Longrightarrow$ Leistungsfähigere, kleinere und billigere Computer
- Anfang der 60er Jahre bildeten sich zwei Entwicklungsstränge heraus:
  - Weiterentwicklung der Stapelverarbeitungssysteme in Richtung gleichzeitig abzuarbeitende Jobs
  - Erste einfache Speicherverwaltung (*Fixed Partitions*)
- 70er Jahre: Aufkommen von Dialogbetrieb (*Time Sharing*) bzw. Zeitteilbetrieb
  - Eine Zentraleinheit, mehrere Terminals (Dialogstationen)
  - Jeder Benutzer erhält beim Anmelden einen Benutzerprozess

  $\Longrightarrow$ Ziel: Faire Verteilung der Rechenzeit
- Ende der 70er Jahre: Entwicklung des Mikroprozessors
  $\Longrightarrow$ Entwicklung des Home-Computer bzw. Personal Computer (PC)
  - 1977: Apple II. Erster Heimcomputer
  - 1981: IBM PC. Meist verkaufte Rechnerarchitektur (Intel 80x86)
- Auch diese Generation kennt noch keine verteilten Systeme

Bekannte Vertreter der 3.Generation

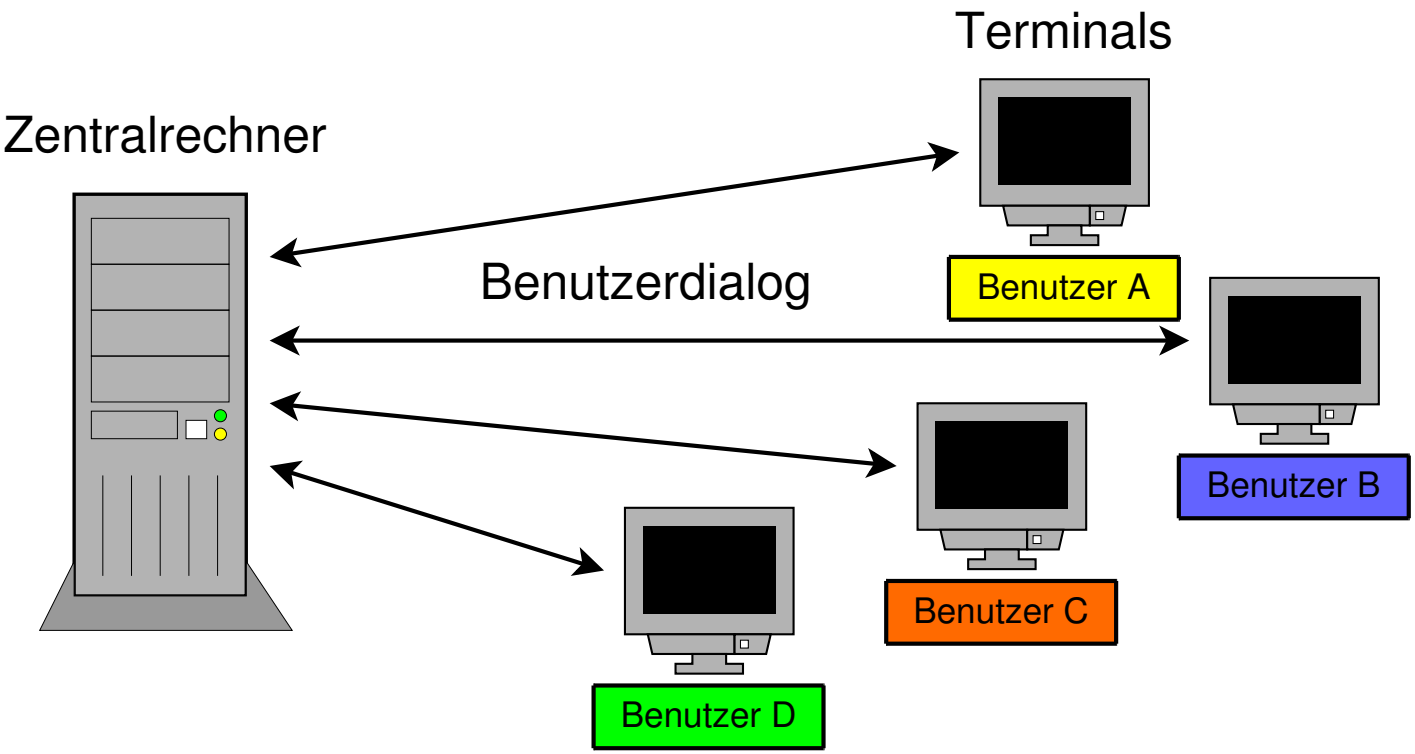
Maschine	Entwicklung	Besonderheiten
CDC 6600	1964	Erster Supercomputer
IBM System/360	1964	8-Bit Zeichengröße. Flexible Architektur. Optimierbar für verschiedene Anwendungen
PDP-8	1965	Erster kommerzieller Minicomputer von DEC
ILLIAC IV	1969	Erster Multiprozessor-Rechner
CRAY 1	1976	Supercomputer

Bildquelle: tecchannel

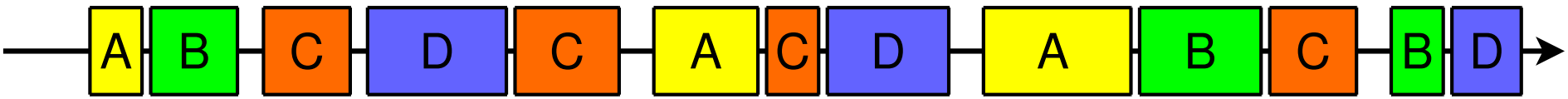
3.Generation: Dialogbetrieb – Timesharing (1)

- Mehrere Benutzer arbeiten an einem Rechner gleichzeitig und konkurrierend, indem sie sich die verfügbare Rechenzeit des Hauptprozessors teilen
- Verteilung der Rechenzeit durch Zeitscheiben (*Time Slices*)
  - Die Verteilung kann nach unterschiedlichen Strategien erfolgen
- Jeder Benutzer glaubt, dass er die gesamten Rechenleistung des Hauptprozessors stets für sich alleine zur Verfügung hat
- Erstmals können mehrere Benutzer gleichzeitig über Terminals an einem Computer interaktiv arbeiten und das Rechenverhalten beeinflussen
- Die Programme der Benutzer laufen voneinander unabhängig
- Die quasi-parallele Programm- bzw. Prozessausführung nennt man **Mehrprogrammbetrieb** oder **Multitasking**
- Ziel: Minimierung der Antwortzeit

3.Generation: Dialogbetrieb – Timesharing (2)



Mehrprogrammbetrieb (Multitasking)



3.Generation: Dialogbetrieb – Timesharing (3)

- Durch Dialogbetrieb kamen neue Arbeitsweisen in die IT, die heute selbstverständlich sind und neue Konzepte wurden notwendig:
  - **Scheduling** (*Zeitplanerstellung*): Automatische Erstellung eines Ablaufplanes (*schedule*), der Benutzern bzw. Prozessen zeitlich begrenzte Ressourcen zuteilt
  - **Swapping** (*Umlagerung*): Prozess des Ein- und Auslagerns von Speichersegmenten in den/vom Arbeitsspeicher vom/in den Hintergrundspeicher (Festplatten)
    - Swapping findet immer dann statt, wenn der Scheduler einen Prozess aktiviert (siehe Speicherpyramide)
  - **Dateisysteme**, die quasi-gleichzeitige Dateizugriffe erlauben
  - **Speicherschutz**: Der Arbeitsspeicher wird aufgeteilt und laufende Programme voneinander getrennt
    - So kann ein Programmierfehler oder Absturz eines einzelnen Programms nicht die Stabilität anderer Programme und des Gesamtsystems beeinträchtigen

4.Generation (1980 bis 2000)

- Aufkommen hoch-integrierter Schaltkreise und exponentiell wachsende Integrationsdichte der elektronischen Komponenten
  - Prozessoren werden immer leistungsfähiger und preiswerter
  - Speicherbausteine haben eine immer höhere Kapazität
- Hohe Rechenleistung kann an jedem Arbeitsplatz installiert werden
  - Workstations setzten sich durch
  - Immer größerer Erfolg von Heimcomputern und Personal Computern
- Hauptziel der Softwareindustrie (sollte es sein): Benutzerfreundliche (grafische) Lösungen für die Benutzer schaffen, die von der zu Grunde liegenden Hardware nichts wissen wollen
  - Auf Personal Computern: MS-DOS Version 1.0 (1981)
  - Auf Workstations (SUN, SGI,...): UNIX
- Aufkommen und Etablierung verteilter Systeme!
  - Insbesondere Client-Server-Architektur
  - Später auch Cluster, Peer-to-Peer, Grids, Clouds,...

5.Generation (2000 bis ????)

- Ein paar Schlagworte aus der 5.Generation:
  - *Das Netz ist der Computer*
  - Verteilte Systeme $\Longrightarrow$ Cluster-, Cloud-, Grid-, P2P-Computing
  - Multicore-Prozessoren und parallelisierte Anwendungen
  - Virtualisierung $\Longrightarrow$ VMware, XEN, KVM,...
  - Freie Software (OpenSource) $\Longrightarrow$ Linux, BSD,...
  - Kommunikation überall $\Longrightarrow$ mobile Systeme
  - Neue Arbeitsformen $\Longrightarrow$ e-Science, e-Learning, e-Business,...
  - Dienste und Services $\Longrightarrow$ Serviceorientierte Architekturen (SOA)
  - Ressourcen nach Bedarf mieten bzw. anfordern $\Longrightarrow$ On Demand
- Schlagworte für später:
  - Quantencomputer (wohl eher 7. oder 8.Generation)

Boolesche Algebra (Schaltalgebra)

- Für programmierbare Rechner existiert mit der **Booleschen Algebra** eine Formalisierung von Aussageverknüpfungen ($\Longrightarrow$ Aussagenlogik)

Spock – STAR TREK VI (1991)

„Logik ist der Anfang aller Weisheit, nicht das Ende.“

- Benannt nach dem englischen Mathematiker George Boole (1815-1864)
- 1937 (veröffentlicht 1938) benutzte der amerikanische Mathematiker und Elektrotechniker Claude Shannon (1916-2001) boolesche Algebren erstmals zur Beschreibung elektrischer Schaltungen
- Das Verständnis der Booleschen Algebra ist wichtig für den Bau effizienter Schaltungen zur Verarbeitung binärer Daten
  - Grundlage und mathematisches Werkzeug zum Bau von Rechnern
- In der Booleschen Algebra existieren nur zwei Wahrheitswerte:
  - 0 (*falsch* btw. *false*) und 1 (*wahr* btw. *true*)
  - Beide Werte kann man als Bitwerte oder Stromzustände interpretieren

Grundfunktionen der Booleschen Algebra

- Hat man 2 Variablen a und b aus der Menge B, so lassen sich 3 Grundfunktionen (Operatoren) der Booleschen Algebra definieren

a	$\bar{a}$
0	1
1	0

- **Negation** bzw. Invertierung (NOT-Operator)
- Wird geschrieben als $\neg a$, $-a$ oder $\bar{a}$

a	b	a $\vee$ b
0	0	0
0	1	1
1	0	1
1	1	1

- **Disjunktion** – logische Summe (OR-Operator)
- Wird geschrieben als $\vee$, oder $+$

a	b	a $\wedge$ b
0	0	0
0	1	0
1	0	0
1	1	1

- **Konjunktion** – logisches Produkt (AND-Operator)
- Wird geschrieben als $\wedge$, oder $*$

Eine Boolesche Algebra $(B, \neg, \vee, \wedge)$ besteht aus einer Booleschen Menge B und den 3 Operatoren

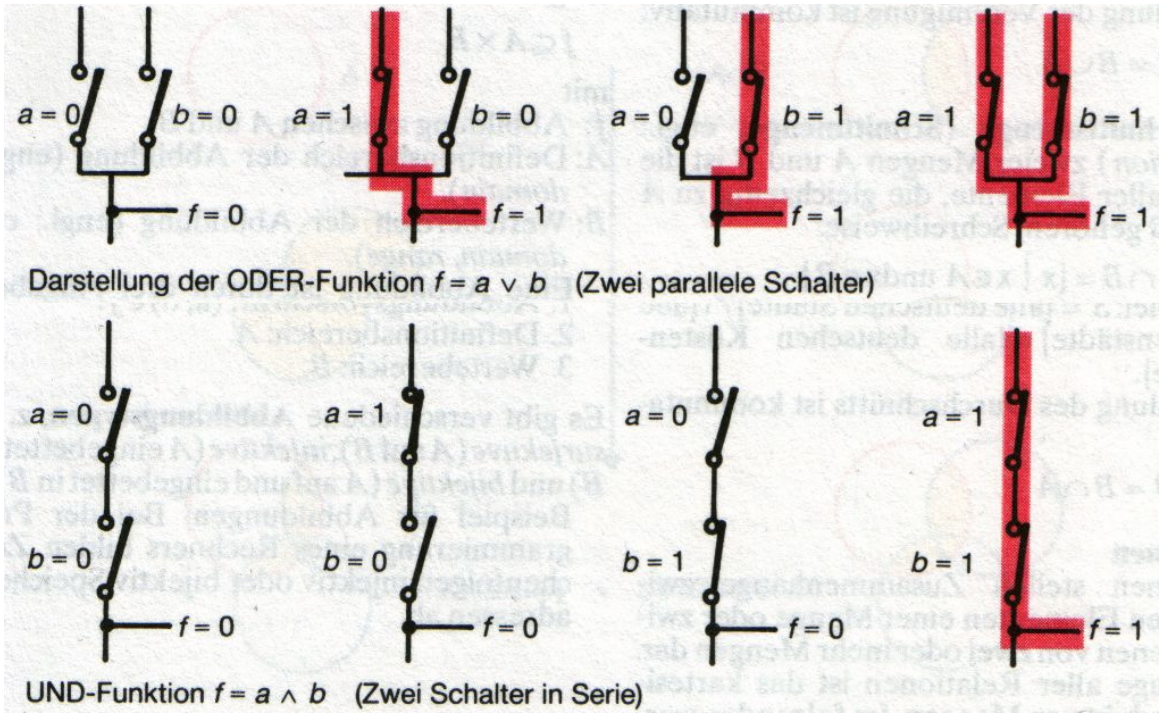
Auswertungsreihenfolge bei der Booleschen Algebra

1. Negation
2. Konjunktion (AND, $\wedge$, $*$)
3. Disjunktion (OR, $\vee$, $+$)

- Die Auswertungsreihenfolge folgt der *Punkt-vor-Strich-Regel*
- Komplizierte Ausdrücke werden entsprechend geklammert

Boolesche Schaltungen

- Verdeutlichen die booleschen Operatoren OR und AND
  - Parallelschaltung (OR-Operation): Strom fließt, wenn mindestens ein Schalter geschlossen ist
  - Serienschaltung (AND-Operation): Strom fließt, wenn beide Schalter geschlossen sind



Bildquelle: dtv-Atlas zur Informatik (1995)

Axiome der Booleschen Algebra

Axiom	**Formel**
Kommutativgesetze	$a \wedge b = b \wedge a$ $a \vee b = b \vee a$
Assoziativgesetze	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$ $a \vee (b \vee c) = (a \vee b) \vee c$
Idempotenzgesetze	$a \wedge a$ $a \vee a$
Distributivgesetze	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
Komplementärgesetze	$a \wedge \neg a = 0$ $a \vee \neg a = 1$
Neutralitätsgesetze (Identitätsgesetze)	$a \wedge 1 = a$ $a \vee 0 = a$
Extremalgesetze (Null-/Einsgesetze)	$a \wedge 0 = 0$ $a \vee 1 = 1$
Dualitätsgesetze	$\neg 0 = 1$ $\neg 1 = 0$
Doppeltes Negationsgesetz	$\neg(\neg a) = a$
Verschmelzungsgesetze (Absorptionsgesetze)	$a \vee (a \wedge b) = a$ $a \wedge (a \vee b) = a$
De Morgansche Gesetze	$\neg(a \wedge b) = \neg a \vee \neg b$ $\neg(a \vee b) = \neg a \wedge \neg b$

- In einer Booleschen Algebra gelten verschiedene Gesetze (Axiome)
- Der Beweis der Gültigkeit dieser Axiome erfolgt mit Hilfe von Wahrheitstabellen

Nachweis der Gültigkeit der Gesetze

- Beispiel: Distributivgesetze
  - $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
  - $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

a	**b**	**c**	$\mathbf{a \wedge (b \vee c)}$	$\mathbf{(a \wedge b) \vee (a \wedge c)}$	$\mathbf{a \vee (b \wedge c)}$	$\mathbf{(a \vee b) \wedge (a \vee c)}$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Funktionen der Booleschen Algebra (1/2)

- Eine Funktion $f : B^n \longrightarrow B$ heißt n-stellige boolesche Funktion
- Eine solche Funktion bildet die Menge aller möglichen n-Tupel aus $\{0, 1\}$ auf $\{0, 1\}$ ab und kann in Form einer Tabelle mit 2^{2^n} Werten dargestellt werden
- $n = 0$ ergibt die 2 Konstanten 1 (*wahr*) und 0 (*falsch*)
- $n = 1$ ergibt 4 einstellige (mit einer Variablen) boolesche Funktionen $y = f_0(x), f_1(x), f_2(x), f_3(x)$

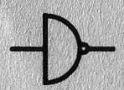
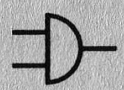
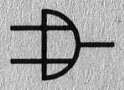
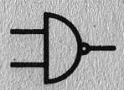
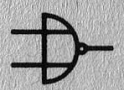
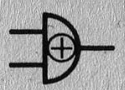
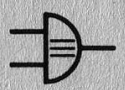
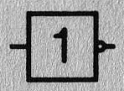
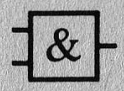
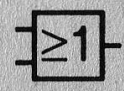
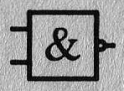
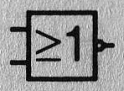
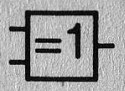
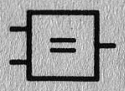
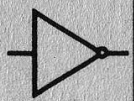
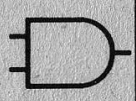
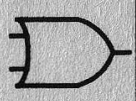
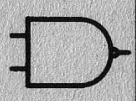
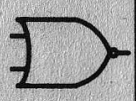
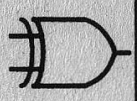
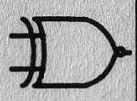
x =	**0**	**1**	**Funktion ($y =$)**	**Name**
f_0	0	0	0	Kontradiktion
f_1	0	1	x	Identität
f_2	1	0	$\neg x = \bar{x} = 1 - x$	Negation
f_3	1	1	1	Tautologie

- $n = 2$ ergibt 16 zweistellige boolesche Funktionen

Funktionen der Booleschen Algebra (2/2)

$x_1 =$ $x_2 =$	**0** **0**	**0** **1**	**1** **0**	**1** **1**	**Funktion**	**Name**	**Sprechweise**
f0	0	0	0	0	$x_1 \wedge \neg x_1$	Nullfunktion	
f1	0	0	0	1	$x_1 \wedge x_2$	Konjunktion	x_1 AND x_2
f2	0	0	1	0	$x_1 \wedge \neg x_2$	1. Differenz	x_1 AND NOT x_2
f3	0	0	1	1	x_1	Identität von x_1	
f4	0	1	0	0	$\neg x_1 \wedge x_2$	2. Differenz	NOT x_1 AND x_2
f5	0	1	0	1	x_2	Identität von x_2	
f6	0	1	1	0	$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$	Antivalenz	x_1 XOR x_2
f7	0	1	1	1	$x_1 \vee x_2$	Disjunktion	x_1 OR x_2
f8	1	0	0	0	$\neg(x_1 \vee x_2)$	Negatdisjunktion	x_1 NOR x_2
f9	1	0	0	1	$(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$	Äquivalenz	
f10	1	0	1	0	$\neg x_2$	Negation von x_2	NOT x_2
f11	1	0	1	1	$x_1 \vee \neg x_2$	2. Implikation	x_1 OR NOT x_2
f12	1	1	0	0	$\neg x_1$	Negation von x_1	NOT x_1
f13	1	1	0	1	$\neg x_1 \vee x_2$	1. Implikation	NOT x_1 OR x_2
f14	1	1	1	0	$\neg(x_1 \wedge x_2)$	Negatkonjunktion	x_1 NAND x_2
f15	1	1	1	1	$x_1 \vee \neg x_1$	Einsfunktion	

Schaltzeichen der wichtigsten Verknüpfungen

x_1	x_2	NOT	AND	OR	NAND	NOR	XOR	ÄQUIV	NAME
0	0	1	0	0	1	1	0	1	Werte-tabelle
0	1	1	0	1	1	0	1	0	
1	0	0	0	1	1	0	1	0	
1	1	0	1	1	0	0	0	1	
		$\overline{x_1}$	$x_1 \cdot x_2$	$x_1 + x_2$	$\overline{x_1 \cdot x_2}$	$\overline{x_1 + x_2}$	$x_1 \oplus x_2$	$x_1 \equiv x_2$	Funktion
									DIN (alt)
									DIN (neu)
									ASA

- Kombinationen aus OR-, AND- und NOT-Gattern reichen, um alle logischen Zusammenhänge darzustellen
- Gatter repräsentieren die Elemente der Schaltalgebra und werden nach deren Regeln zu Schaltnetzen zusammengeschaltet

Bildquelle: Bernd Schürmann. Skript zur Vorlesung Rechnersysteme (1998)

Beispiel für den Entwurf einer Digitalschaltung (1/3)

- Bei einer Spülmaschinensteuerung soll der Spülvorgang sofort unterbrochen werden, wenn entweder die Spülmaschinentür geöffnet wird oder wenn der Wasserstand bei eingeschalteter Heizung unter einen Mindeststand sinkt
- Diese Signale werden durch 3 Fühler überwacht:
  - Fühler a: Spülmaschine offen (1) zu (0)
  - Fühler b: Wasserstand zu niedrig (1) sonst (0)
  - Fühler c: Heizung an (1) aus (0)
- Die Unterbrechung des Spülvorgangs wird dadurch gesteuert, dass ein Stoppsignal auf 1 gesetzt wird
- Läuft die Maschine, steht das Stoppsignal auf 0
- Das Stoppsignal ist damit eine Funktion $f(a, b, c)$

Quelle: Frank Staab. Logik und Algebra. Oldenbourg Wissenschaftsverlag (2007)

Beispiel für den Entwurf einer Digitalschaltung (2/3)

- Schalttabelle zur Aufgabenstellung:

Spülmaschinentür a	**Wasserstand zu niedrig** b	**Heizung an** c	**Stoppsignal** $f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Fühler a: Spülmaschine offen (1) zu (0), Fühler b: Wasserstand zu niedrig (1) sonst (0), Fühler c: Heizung an (1) aus (0)

- Gesucht ist die Schaltfunktionen $f(a,b,c)$, welche die Schalttabelle realisiert

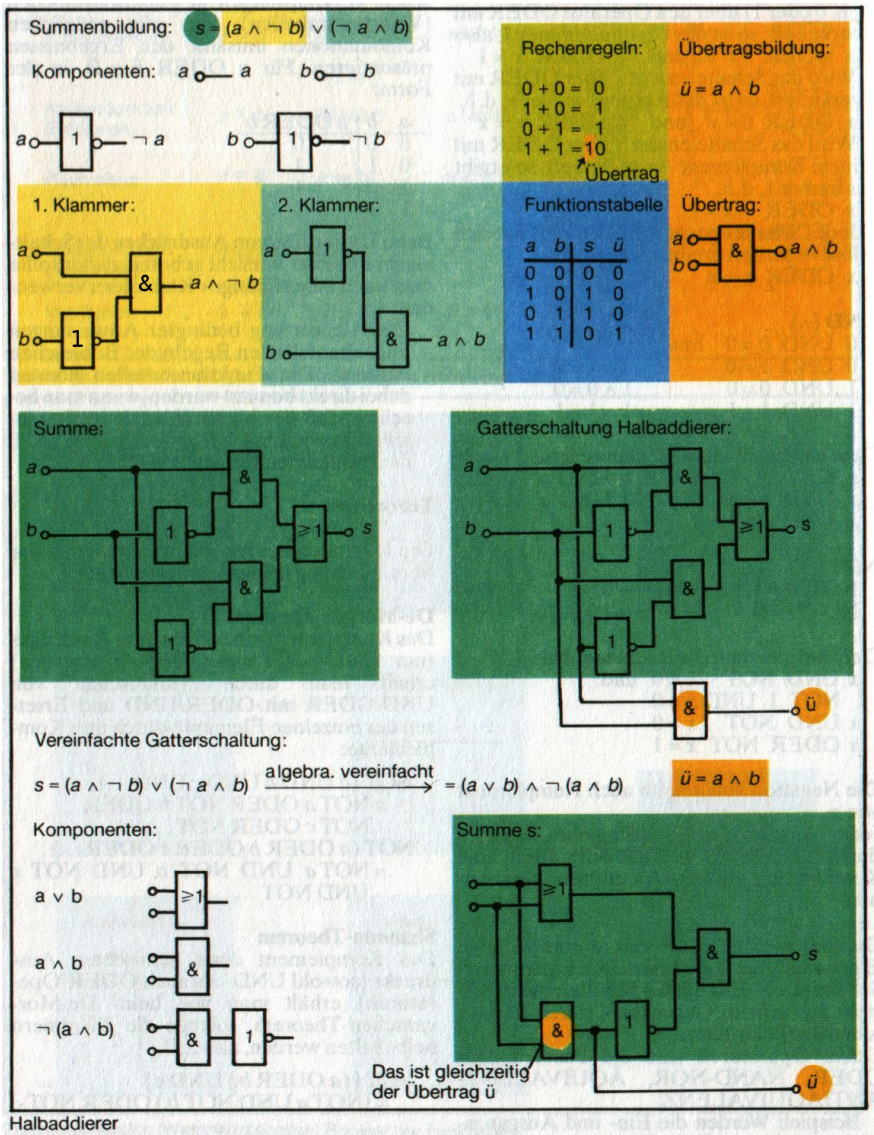
Quelle: Frank Staab. Logik und Algebra. Oldenbourg Wissenschaftsverlag (2007)

Beispiel für den Entwurf einer Digitalschaltung (3/3)

- Wegen der einfachen Aufgabenstellung ist die Lösung trivial
- Beim Öffnen der Spülmaschinentür ist der Signalzustand der anderen beiden Signale unerheblich
- Das Anlegen einer 1 an Signalleitung a führt zum sofortigen Stopp des Spülvorgangs
- Nun muss nur noch das gleichzeitige Anliegen einer 1 an Signalleitung b und c abgeprüft werden
- Darum ist die gesuchte Schaltfunktion: $f(a, b, c) = a \vee (b \wedge c)$
- Beachten Sie:
  - Dieses Beispiel ist recht einfach
  - Nicht bei allen Aufgabenstellungen ist die Lösung so einfach ablesbar
  - Es gibt zahlreiche weitere Funktionen, die das gleiche Ergebnis haben und somit auch korrekte Lösungen hier sind
    - Eine alternative Lösung: $f(a, b, c) = (a \vee b) \wedge (a \vee c)$

Quelle: Frank Staab. Logik und Algebra. Oldenbourg Wissenschaftsverlag (2007)

Beispiel zur Schaltalgebra (Halbaddierer)



- Der Halbaddierer berechnet die Summe zweier Dualziffern
- Der eventuell entstehende Übertrag (ü) wird von der Summenziffer (s) getrennt aufgeführt

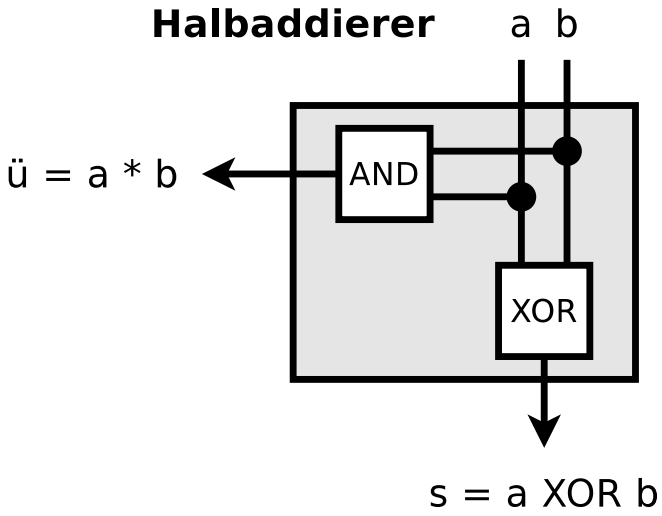
a	b	s	ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

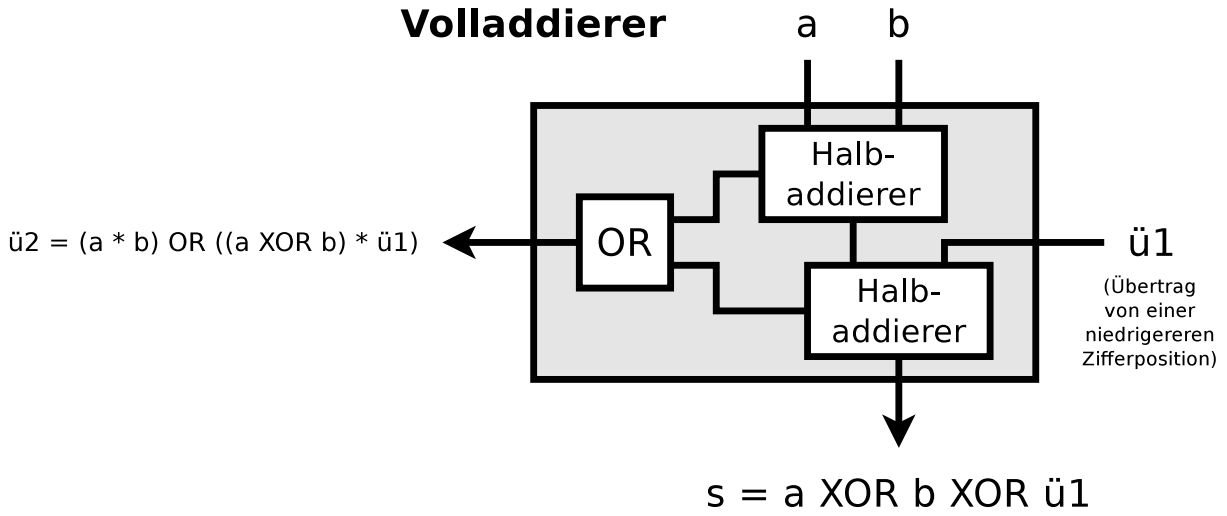
- Daraus lassen sich die folgenden Schaltfunktionen bestimmen
- $s = (a \wedge \neg b) \vee (\neg a \wedge b)$
- $ü = a \wedge b$
- 2 NOT-Gatter, 3 AND-Gatter und 1 OR-Gatter können in einem entsprechenden Schaltnetz beide Gleichungen darstellen

Quelle: dtv-Atlas zur Informatik (1995)

Addierwerk realisieren

- Halbaddierer: Addiert 2 Dualziffern, gibt Summe und Übertrag aus
- Volladdierer: Berücksichtigt zusätzlich einen von einer niedrigeren Zifferposition kommenden Übertrag
- Ein Volladdierer lässt sich aus 2 Halbaddierern und einem OR aufbauen





- Aus einer Kaskade von *n* Volladdierern und einem Halbaddierer kann man ein Addierwerk zusammenbauen

Nächste Vorlesung

Nächste Vorlesung:

3.11.2011