

HANSER



Leseprobe

zu

„Grundlagen der Technischen Informatik“

von Dirk W. Hoffmann

Print-ISBN 978-3-446-46314-1

E-Book-ISBN 978-3-446-46360-8

Weitere Informationen und Bestellungen unter
<http://www.hanser-fachbuch.de/978-3-446-46314-1>

sowie im Buchhandel

© Carl Hanser Verlag, München



Vorwort

Die Computertechnik hat in wenigen Jahrzehnten eine Entwicklung vollzogen, die in ihrer Geschwindigkeit und Intensität einzigartig ist. Setzen sich die ersten Computer noch aus vergleichsweise wenigen Schaltkreisen zusammen, so verrichten in jedem modernen Arbeitsplatzrechner, Tablet-PC oder Smartphone Abermillionen von Transistoren ihren Dienst und führen in jeder Sekunde Milliarden von Berechnungen aus. Doch so rasant die Entwicklung der letzten Jahrzehnte auch war: Vergleichen wir die Maschinen der Pionierzeit mit unseren modernen Rechenboliden, so lassen sich eine Reihe von Grundprinzipien identifizieren, die sich im Laufe der Zeit zwar weiterentwickelt, aber im Kern nicht verändert haben. Diese Grundprinzipien, zusammen mit ihren modernen Ausprägungen, formen das Gebiet der technischen Informatik und sind Gegenstand des vorliegenden Buchs.

Geschrieben habe ich das Buch für Bachelor-Studenten der Fachrichtungen Informatik, Elektrotechnik, Informationstechnik und verwandter Studiengänge. Inhaltlich habe ich mich dabei an den typischen Lehrinhalten orientiert, die im Grundstudium an Hochschulen und Universitäten vermittelt werden. Neben dem Grundlagenwissen aus den Gebieten der Halbleitertechnik, der Zahlendarstellung und der booleschen Algebra werden die Entwurfsprinzipien kombinatorischer und sequenzieller Hardware-Komponenten bis hin zur Beschreibung moderner Prozessor- und Speicherarchitekturen vermittelt. Damit spannt das Buch den Bogen von den mathematischen Grundlagen digitaler Schaltelemente bis hin zu den ausgefeilten Hardware-Optimierungen moderner Hochleistungscomputer.

Es ist mir ein besonderes Anliegen, den Stoff anwendungsorientiert und didaktisch ansprechend zu vermitteln. Damit das Buch sowohl vorlesungsbegleitend als auch zum Selbststudium eingesetzt werden kann, werden die Lehrinhalte aller Kapitel durch zahlreiche Übungsaufgaben komplementiert. Des Weiteren habe ich zahlreiche Anwendungsbezüge mit aufgenommen, um eine enge Verzahnung zwischen Theorie und Praxis zu erreichen.

Seit dem Erscheinen der letzten Auflage habe ich wieder zahlreiche Zuschriften erhalten, über die ich mich sehr gefreut habe. Namentlich bedanken möchte ich mich bei Herrn Martin Maltzahn, Prof. Dr. Joachim Melcher, Prof. Dr. Quirin Meyer und Prof. Dr. Martin Rumppler, die mich auf mehrere Fehler in der fünften Auflage hingewiesen haben. Inzwischen erscheinen die *Grundlagen der technischen Informatik* in der sechsten Auflage, und ich bin weiterhin jedem aufmerksamen Leser für Hinweise zu Verbesserungsmöglichkeiten oder Fehlern dankbar.

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einführung | 11 |
| 1.1 | Was ist technische Informatik? | 11 |
| 1.2 | Vom Abakus zum Supercomputer | 13 |
| 1.3 | Wohin geht die Reise? | 30 |
| 2 | Halbleitertechnik | 33 |
| 2.1 | Halbleiter | 34 |
| 2.1.1 | Atommodell von Bohr | 34 |
| 2.1.2 | Reine Halbleiter | 37 |
| 2.1.3 | Dotierte Halbleiter | 39 |
| 2.2 | Integrierte Schaltelemente | 41 |
| 2.2.1 | Halbleiterdioden | 41 |
| 2.2.2 | Bipolartransistoren | 42 |
| 2.2.3 | Feldeffekttransistoren | 46 |
| 2.3 | Chip-Fertigung | 51 |
| 2.3.1 | Produktion integrierter Schaltkreise | 51 |
| 2.3.2 | Integrationsdichte | 57 |
| 2.4 | Übungsaufgaben | 58 |
| 3 | Zahlendarstellung und Codes | 59 |
| 3.1 | Zahlensysteme | 60 |
| 3.2 | Rechnerinterne Zahlenformate | 67 |
| 3.2.1 | Darstellung natürlicher Zahlen | 67 |
| 3.2.2 | Darstellung rationaler Zahlen | 73 |
| 3.3 | Zahlencodes | 80 |
| 3.3.1 | Tetraden-Codes | 80 |
| 3.3.2 | Fehlererkennende Codes | 84 |
| 3.4 | Übungsaufgaben | 86 |
| 4 | Boolesche Algebra | 89 |
| 4.1 | Axiomatisierung nach Huntington | 90 |
| 4.1.1 | Mengenalgebra | 91 |
| 4.1.2 | Schaltalgebra | 93 |
| 4.2 | Boolesche Ausdrücke und Aussagen | 95 |
| 4.2.1 | Abgeleitete Operatoren | 97 |
| 4.2.2 | Erfüllbarkeit und Äquivalenz | 100 |
| 4.2.3 | Strukturelle Induktion | 102 |
| 4.2.4 | Dualitätsprinzip | 105 |

| | | |
|----------|---|------------|
| 4.3 | Rechnen in booleschen Algebren | 109 |
| 4.3.1 | Abgeleitete Umformungsregeln | 109 |
| 4.3.2 | Vereinfachung boolescher Ausdrücke | 111 |
| 4.3.3 | Vollständige Operatorensysteme | 117 |
| 4.4 | Normalformdarstellungen | 119 |
| 4.4.1 | Konjunktive und disjunktive Normalform | 119 |
| 4.4.2 | Reed-Muller-Normalform | 122 |
| 4.4.3 | Binäre Entscheidungsdiagramme | 125 |
| 4.5 | Übungsaufgaben | 133 |
| 5 | Schaltnetze | 139 |
| 5.1 | Grundlagen der Digitaltechnik | 140 |
| 5.1.1 | Schaltkreisfamilien | 140 |
| 5.1.2 | MOS-Schaltungstechnik | 145 |
| 5.1.3 | Lastfaktoren | 155 |
| 5.2 | Schaltungssynthese | 156 |
| 5.2.1 | Zweistufige Schaltungssynthese | 157 |
| 5.2.2 | BDD-basierte Schaltungssynthese | 158 |
| 5.2.3 | FDD-basierte Schaltungssynthese | 159 |
| 5.3 | Formelsynthese | 161 |
| 5.3.1 | Funktionale Formelsynthese | 161 |
| 5.3.2 | Relationale Formelsynthese | 163 |
| 5.3.3 | Definitorische Formelsynthese | 164 |
| 5.4 | Komplexitätsanalyse | 167 |
| 5.5 | Zeitverhalten digitaler Schaltungen | 169 |
| 5.5.1 | Signalausbreitung und -verzögerung | 169 |
| 5.5.2 | Störimpulse | 171 |
| 5.6 | Übungsaufgaben | 175 |
| 6 | Minimierung | 181 |
| 6.1 | Minimierungsziele | 182 |
| 6.2 | Karnaugh-Veitch-Diagramme | 186 |
| 6.2.1 | Minimierung partiell definierter Funktionen | 190 |
| 6.2.2 | Konstruktion Hazard-freier Schaltungen | 194 |
| 6.2.3 | Minimierung mehrstelliger Funktionen | 196 |
| 6.3 | Quine-McCluskey-Verfahren | 197 |
| 6.4 | Übungsaufgaben | 201 |
| 7 | Standardschaltnetze | 205 |
| 7.1 | Motivation | 206 |
| 7.2 | Multiplexer und Demultiplexer | 206 |
| 7.3 | Komparatoren | 213 |
| 7.4 | Präfix-Logik | 215 |

| | | |
|----------|---|------------|
| 7.5 | Addierer | 218 |
| 7.5.1 | Halb- und Volladdierer | 218 |
| 7.5.2 | Carry-ripple-Addierer | 220 |
| 7.5.3 | Carry-look-ahead-Addierer | 221 |
| 7.5.4 | Conditional-Sum-Addierer | 224 |
| 7.5.5 | Präfix-Addierer | 227 |
| 7.5.6 | Carry-save-Addierer | 229 |
| 7.6 | Inkrementierer | 232 |
| 7.7 | Subtrahierer | 233 |
| 7.8 | Multiplizierer | 234 |
| 7.8.1 | Matrixmultiplizierer | 235 |
| 7.8.2 | Carry-save-Multiplizierer | 238 |
| 7.8.3 | Wallace-Tree-Multiplizierer | 241 |
| 7.8.4 | Dadda-Tree-Multiplizierer | 246 |
| 7.9 | Barrel-Shifter | 249 |
| 7.10 | Arithmetisch-logische Einheit | 251 |
| 7.11 | Programmierbare Logikbausteine | 253 |
| 7.12 | Übungsaufgaben | 256 |
| 8 | Schaltwerke | 265 |
| 8.1 | Digitale Speicherelemente | 266 |
| 8.1.1 | Asynchrone Speicherelemente | 267 |
| 8.1.2 | Taktzustandsgesteuerte Speicherelemente | 271 |
| 8.1.3 | Taktflankengesteuerte Speicherelemente | 274 |
| 8.1.4 | Bevorrechtigte Eingänge | 281 |
| 8.1.5 | CMOS-Implementierung | 282 |
| 8.2 | Vom Flipflop zum Schaltwerk | 285 |
| 8.2.1 | Endliche Automaten | 286 |
| 8.2.2 | Schaltwerksynthese | 289 |
| 8.3 | Übungsaufgaben | 293 |
| 9 | Standardschaltwerke | 299 |
| 9.1 | Register | 300 |
| 9.1.1 | Auffangregister | 300 |
| 9.1.2 | Schieberegister | 302 |
| 9.1.3 | Universalregister | 304 |
| 9.1.4 | Akkumulatoren | 305 |
| 9.2 | Zähler | 308 |
| 9.2.1 | Synchrone Binärzähler | 309 |
| 9.2.2 | Asynchrone Binärzähler | 313 |
| 9.2.3 | Mischzähler | 314 |
| 9.2.4 | Instruktionszähler | 316 |

| | | |
|-----------|--|------------|
| 9.3 | Hauptspeicher | 318 |
| 9.3.1 | SRAM-Speicher | 318 |
| 9.3.2 | DRAM-Speicher | 320 |
| 9.3.3 | Fehlererkennung und -korrektur | 327 |
| 9.4 | Übungsaufgaben | 330 |
| 10 | Register-Transfer-Entwurf | 335 |
| 10.1 | Entwurf komplexer Systeme | 336 |
| 10.1.1 | Operationswerksynthese | 338 |
| 10.1.2 | Steuerwerksynthese | 340 |
| 10.2 | Mikroprogrammierung | 343 |
| 10.3 | Übungsaufgaben | 349 |
| 11 | Mikroprozessortechnik | 351 |
| 11.1 | Elemente eines Mikrorechners | 352 |
| 11.1.1 | Von-Neumann-Architektur | 352 |
| 11.1.2 | Aufbau der CPU | 356 |
| 11.2 | Ein einfacher Modellprozessor | 360 |
| 11.3 | Übungsaufgaben | 374 |
| 12 | Rechnerstrukturen | 377 |
| 12.1 | Rechnerklassifikation nach Flynn | 378 |
| 12.2 | Instruktionsarchitekturen | 379 |
| 12.2.1 | CISC-Prozessoren | 380 |
| 12.2.2 | RISC-Prozessoren | 384 |
| 12.3 | Methoden zur Leistungssteigerung | 388 |
| 12.3.1 | Pipelining | 388 |
| 12.3.2 | Cache-Speicher | 393 |
| 12.4 | Leistungsbewertung | 399 |
| 12.4.1 | Maßzahlen zur Leistungsbewertung | 399 |
| 12.4.2 | Benchmarks | 402 |
| 12.5 | Übungsaufgaben | 405 |
| A | Notationsverzeichnis | 411 |
| B | Abkürzungsverzeichnis | 413 |
| C | Glossar | 415 |
| | Literaturverzeichnis | 433 |
| | Namensverzeichnis | 437 |
| | Sachwortverzeichnis | 439 |

1 Einführung

„The first microprocessor only had 22 hundred transistors. We are looking at something a million times that complex in the next generations – a billion transistors. What that gives us in the way of flexibility to design products is phenomenal.“

Gordon E. Moore, Intel Corporation

1.1 Was ist technische Informatik?

Blicken wir auf die Entwicklung der letzten hundert Jahre zurück, so hat keine andere technische Innovation unser Leben mehr verändert als die Erfindung des Computers, wie wir ihn heute kennen. Die Geschwindigkeit, mit der die digitale Revolution immer größere Bereiche unseres täglichen Lebens erobert und umgestaltet hat, ist nicht nur in der Retrospektive atemberaubend. Die Auswirkungen sind heute tief bis in unser kulturelles und gesellschaftliches Leben zu spüren. Ob wir wollen oder nicht: Wir stehen heute an der Schwelle des *ubiquitären Computerzeitalters* und haben sie in manchen Bereichen auch schon überschritten. Mit der Fortsetzung der kontinuierlich voranschreitenden Miniaturisierung und der zunehmenden Vernetzung verschiedenster Geräte ist der Computer von morgen allgegenwärtig und in vielen Fällen nicht einmal mehr als solcher zu erkennen.

Hand in Hand mit der sich rasant entwickelnden Computertechnik wuchs gleichermaßen die Bedeutung der Informatik, die sich in kürzester Zeit von einer Nischendisziplin zu einer eigenständigen Wissenschaft entwickeln konnte (vgl. Abbildung 1.1). Eine ihrer Kernsäulen ist die *technische Informatik*, die sich grob gesprochen mit dem *Entwurf, der logischen Struktur und der technischen Realisierung von Computer-Hardware* beschäftigt.

Ausgehend von der elementaren Hardware-Komponente des *Logikgatters* beschäftigt sich die technische Informatik mit der Konstruktion

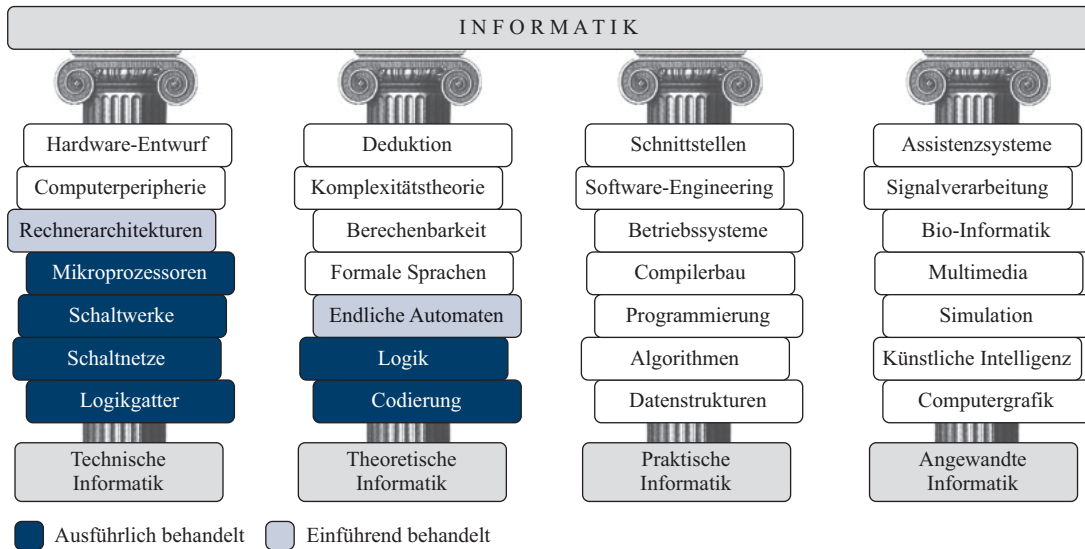


Abbildung 1.1: Die vier Säulen der Informatik

komplexer Digitalschaltungen. Hierzu gehören einfache *Schaltnetze* genauso wie komplexe, mit *Speicherelementen* angereicherte *Schaltwerke*. Durch das wechselseitige Zusammenspiel von Millionen von Schaltelementen sind wir heute in der Lage, Systeme mit einer Komplexität zu konstruieren, die noch vor nicht allzu langer Zeit als unmöglich erachtet wurde. Nichtsdestotrotz lässt sich selbst das komplexeste System stets auf die gleichen Grundprinzipien zurückführen. Die folgenden Kapitel werden diese in ein helleres Licht rücken und den Leser praxisnah in die technische Funktionsweise moderner Computersysteme einführen.

Die technische Informatik ist eng mit der theoretischen Informatik verzahnt. Viele der dort entwickelten Konzepte aus den Bereichen der Codierungstheorie, Logik und der endlichen Automaten dienen uns als das mathematische Fundament zur Beschreibung von Computer-Hardware. In entsprechender Weise werden wir uns auch in diesem Buch mit etlichen Teilaspekten dieser Disziplin beschäftigen. Doch bevor wir vollends in die Welt der Bits und Bytes eintauchen, wollen wir einen kurzen Streifzug durch die junge, aber bewegte Geschichte der Computertechnik wagen und uns mit der Frage beschäftigen, wohin die Reise in den nächsten Jahren führen wird.

1.2 Vom Abakus zum Supercomputer

Die ersten mechanischen Rechenhilfen

Wir beginnen unseren Streifzug durch die Geschichte im elften Jahrhundert vor Christus. Etwa zu dieser Zeit wird in China mit dem *Suan pan* die erste mechanische Rechenhilfe entwickelt – der sogenannte *Abakus*. Obwohl das auf den ersten Blick primitiv anmutende Rechenbrett nicht viel mit der heutigen Computertechnik verbindet, stellt der Abakus einen bedeutenden Schritt in Richtung des maschinellen Rechnens dar und ist mit seiner redundanten Zifferndarstellung ein willkommener Einstieg in die Thematik der Zahlensysteme.

Abbildung 1.2 zeigt das Bild eines chinesischen Abakus, wie er noch heute auf vielen fernöstlichen Warenmärkten die uns vertraute elektronische Kasse ersetzt. Aufgebaut ist der Suan pan aus einer Reihe von Stäben, auf denen jeweils 7 bewegliche Kugeln in zwei unterschiedlichen Segmenten aufgefädelt sind. Das obere Segment – der *Himmel* – enthält jeweils 2 und das untere Segment – die *Erde* – die restlichen fünf Kugeln. Zur Zahlendarstellung verwendet der Abakus das uns geläufige arabische System. Jeder Stab repräsentiert eine einzelne Ziffer, deren Wert sich aus der Stellung und den Wertigkeiten der einzelnen Kugeln bestimmt. In die Berechnung des Ziffernwerts gehen ausschließlich diejenigen Kugeln ein, die nach *innen*, d. h. in Richtung der mittleren Querstrebe, geschoben wurden. Jede Kugel aus dem oberen Segment erhöht den Ziffernwert dabei um 5 und jede Kugel aus dem unteren Segment um 1.

Abbildung 1.3 demonstriert die Darstellung der Zahl 10. Wie das Beispiel zeigt, ist die Zahlendarstellung im Gegensatz zum klassischen Dezimalsystem nicht eindeutig. Der Grund hierfür liegt in der Anzahl und der Wertigkeit der Kugeln, mit denen sich nicht nur die klassischen Dezimalziffern 0 bis 9, sondern auch die Werte 10 bis 15 darstellen lassen.

Der Aufbau des Abakus hat sich im Laufe der Zeit und durch den Einfluss verschiedener Kulturkreise in unterschiedliche Richtungen weiterentwickelt. So besitzt der japanische *Soroban* im Gegensatz zum chinesischen Suan pan nur noch 5 statt 7 Kugeln und der russische *Stschoty* kennt z. B. überhaupt keine Aufteilung in Himmel und Erde mehr.

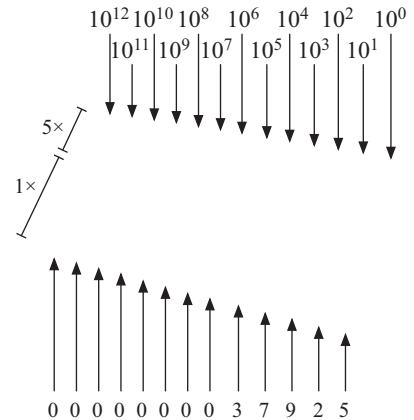
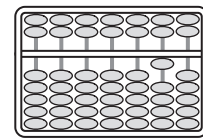
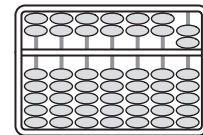


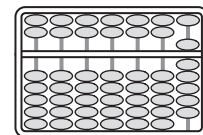
Abbildung 1.2: Der *Suan pan* (chinesischer Abakus)



$$10 = 1 \cdot 10$$



$$10 = 2 \cdot 5$$



$$10 = 1 \cdot 5 + 5 \cdot 1$$

Abbildung 1.3: Die Zahl 10 kann mit Hilfe des Abakus auf drei verschiedene Weisen dargestellt werden.

„Dasselbe, was Du auf rechnerischem Weg gemacht hast, habe ich kürzlich mechanisch versucht und eine aus 11 vollständigen und 6 verstümmelten Rädchen bestehende Maschine gebaut, welche gegebene Zahlen im Augenblick automatisch zusammenrechnet: addiert, subtrahiert, multipliziert und dividiert. Du würdest hell auflachen, wenn Du da wärest und sehen könntest, wie sie, so oft es über einen Zehner oder Hunderter weggeht, die Stellen zur Linken ganz von selbst erhöht oder ihnen beim Subtrahieren etwas wegnimmt.“



Wilhelm Schickard (1592 - 1635)

Abbildung 1.4: Schickard in einem Brief vom 20.9.1623 an Johannes Kepler



Abbildung 1.5: Die Schickard'sche Rechenuhr

Die Schickard'sche Rechenuhr

Mit dem Abakus lassen sich Zahlen *darstellen* und mit etwas Übung selbst komplexe arithmetische Operationen durchführen. Gleichwohl erfolgen alle Berechnungen stets manuell. Weit mehr als tausend Jahre vergingen, bis der deutsche Astronom, Mathematiker und Theologe Wilhelm Schickard das erste mechanische Rechenwerk ersann, mit dessen Hilfe sich zwei Zahlen zum einen vollautomatisch addieren und subtrahieren und zum anderen halbautomatisch multiplizieren und dividieren ließen. Schickard, der am 22.4.1592 im schwäbischen Herrenberg geboren wurde und 1619 einem Ruf an die Universität Tübingen folgte, verband eine lebenslange Freundschaft mit dem kaiserlichen Mathematiker und Astronomen Johannes Kepler, der für seine umfangreichen Berechnungen zur Planetenbewegung bis dato auf Papier und Bleistift angewiesen war.

Die Schickard'sche Rechenuhr besteht aus drei Teilen, die übereinander angeordnet sind (siehe Abbildung 1.5). Der obere Teil entspricht dem *Multiplikationswerk*, der mittlere dem *Additionswerk* und der untere einem *Speicher*, der die Funktion einer Merkhilfe für Zwischenergebnisse übernimmt. Die Funktionsweise des Multiplikationswerks entspricht im Wesentlichen dem Prinzip der *Napierstäbchen*, die auf den schottischen Mathematiker Lord John Napier of Merchiston zurückgehen [32]. Das Multiplikationswerk und das Additionswerk verfügen über keinerlei mechanische Verbindung. Die verschiedenen bei der Produktberechnung entstehenden Teilsummen mussten deshalb manuell abgelesen und per Hand in das Addierwerk übertragen werden. Aus diesem Grund war die Multiplikation mit Hilfe der Schickard'sche Rechenuhr nur halbautomatisch möglich.

Von der Schickard'schen Rechenuhr wurden nur zwei Exemplare überhaupt gebaut, die in den Wirren des Dreißigjährigen Kriegs für immer verloren gingen. Anhand von Skizzen und Aufzeichnungen aus den Nachlässen von Schickard und Kepler konnte die Rechenuhr jedoch rekonstruiert und die Funktionstüchtigkeit in mehreren Nachbauten nachträglich unter Beweis gestellt werden.

Es war die Pest, die das Leben von Wilhelm Schickard und seiner Familie im sechzehnten Jahr des Dreißigjährigen Kriegs ein tragisches Ende nehmen ließ. Zuerst rafft der schwarze Tod im Jahre 1634 seine Frau und seine drei Töchter dahin. Ein Jahr später, am 24. Oktober 1635, stirbt auch Wilhelm Schickard – zwei Tage, bevor sein neunjähriger Sohn ebenfalls der Seuche erliegt.

Die Rechenmaschinen des Charles Babbage

Weitere Meilensteine im Bereich des maschinellen Rechnens stellen die Rechenmaschinen des britischen Mathematikers und Ökonomen Charles Babbage dar, der am 26. Dezember 1791 in London das Licht der Welt erblickte [45]. Bereits mit 22 Jahren wird Babbage Mitglied in der Royal Society und nimmt 1823 unter Förderung der britischen Regierung die Arbeiten an der *Differenzenmaschine* auf. Im Gegensatz zur Schickard'schen Rechenuhr, die für die automatische bzw. halbautomatische Durchführung von primitiven arithmetischen Operationen konzipiert war, sollte die Differenzenmaschine in der Lage sein, ganze Wertetafeln komplexer Polynome selbstständig zu erzeugen.

Das Prinzip der Maschine beruhte auf der Newton'schen *Differenzenmethode*, mit der solche Berechnungen auf trickreiche Weise unter der ausschließlichen Verwendung von Additionen und Subtraktionen durchgeführt werden können. Um zum Beispiel das Polynom

$$y = x^2 - 2x + 3$$

mit Hilfe der Differenzenmethode an den Stützstellen $x = 0, 1, 2, 3, \dots$ auszuwerten, gehen wir in drei Schritten vor:

- Im ersten Schritt stellen wir die *initiale Differenzentabelle* auf. Für Polynome n -ten Grades tragen wir zunächst die ersten $n + 1$ manuell berechneten Funktionswerte $y(0)$ bis $y(n)$ ein. Die nächsten Spalten werden durch sukzessive Differenzberechnung erzeugt. Dabei enthält die zweite Spalte die Differenzen der Elemente der ersten Spalte, die dritte Spalte die Differenzen der Elemente der zweiten Spalte und so fort. Insgesamt entsteht auf diese Weise die in Abbildung 1.7 (oben) dargestellte Tabelle.
- In der dritten Spalte erhalten wir als Differenz zweiter Ordnung den Wert 2. Egal um wie viele Stützstellen Sie die Tabelle nach unten ergänzen – die Elemente der dritten Spalte sind für unser Beispiel stets gleich. Aus funktionsanalytischer Sicht ist dieses Ergebnis auch nicht weiter verwunderlich, schließlich berechnen wir durch die n -malige Differenzenbildung nichts anderes als die diskrete Ableitung n -ter Ordnung und diese ist für Polynome n -ten Grades stets konstant. Unseren Überlegungen folgend können wir somit die dritte Spalte, wie in Abbildung 1.7 (Mitte) dargestellt, im zweiten Schritt beliebig nach unten erweitern.
- Im dritten Schritt füllen wir fehlende Einträge der Differenzentabelle von rechts nach links auf und können die gesuchten Funktionswerte

„One evening I was sitting in the rooms of the Analytical Society, at Cambridge, my head leaning forward on the table in a kind of dreamy mood, with a table of logarithms laying open before me. Another member, coming into the room, and seeing me half asleep, called out, ‘Well Babbage, what are you dreaming about?’ to which I replied, ‘I am thinking that all these tables (pointing to the logarithms) might be calculated by machinery.’“ [95]



Charles Babbage (1791 – 1871)

Abbildung 1.6: Charles Babbage

■ Die initiale Differenzentabelle:

| | $y(i)$ | $\Delta(i)$ | $\Delta\Delta(i)$ |
|---------|--------|-------------|-------------------|
| $i = 0$ | 3 | | |
| $i = 1$ | 2 | 1 | |
| $i = 2$ | 3 | -1 | 2 |

■ Fortsetzen der letzte Spalte:

| | $y(i)$ | $\Delta(i)$ | $\Delta\Delta(i)$ |
|---------|--------|-------------|-------------------|
| $i = 0$ | 3 | | |
| $i = 1$ | 2 | 1 | 2 |
| $i = 2$ | 3 | -1 | 2 |
| | | | 2 |
| | | | 2 |
| | | | 2 |

■ Ableiten weiterer Stützstellen:

| | $y(i)$ | $\Delta(i)$ | $\Delta\Delta(i)$ |
|---------|--------|-------------|-------------------|
| $i = 0$ | 3 | | |
| $i = 1$ | 2 | 1 | |
| $i = 2$ | 3 | -1 | 2 |
| $i = 3$ | 6 | -3 | 2 |
| $i = 4$ | 11 | -5 | 2 |
| $i = 5$ | 18 | -7 | |

Abbildung 1.7: Stützstellenberechnung mit Hilfe der Differenzenmethode am Beispiel des Polynoms $y = x^2 - 2x + 3$

schließlich in der ersten Spalte ablesen. Die auf diese Weise vervollständigte Differenzentabelle ist für unser Beispiel in Abbildung 1.7 (unten) dargestellt.

Leider wird die Differenzenmaschine nie gebaut. Zum einen gestaltet sich die Fertigung vieler der geschätzten 25.000 Bauteile schwieriger als erwartet, zum anderen bringt Babbage stets neue Ideen und Verbesserungsvorschläge ein, die das Projekt zusätzlich verlangsamen. Als die Kosten schließlich vollends aus dem Ruder laufen, zieht sich die britische Regierung 1842 aus dem Projekt zurück.

In der Folgezeit ersann Babbage neben einer deutlich verbesserten *Differenzenmaschine 2* auch die *analytische Maschine*, die in ihrer Komplexität alle seine bisherigen Entwürfe nochmals weit übertrifft. Obwohl es sich bei der analytischen Maschine um eine vollständig mechanische Konstruktion handelt, finden sich in deren Entwurf viele der klassischen Elemente wieder, die auch heute noch die Grundstrukturen moderner Computer prägen. So verfügt die analytische Maschine über einen Speicher und ein getrenntes Rechenwerk (*Mill*). Zur Ausgabe der Ergebnisse sah Babbage einen speziell konstruierten Drucker vor. Programmiert werden sollte die Maschine mit Hilfe von *Lochkarten* – ein Konzept, das zu dieser Zeit bereits bekannt war und erstmals 1805 von dem französischen Buchbinder Joseph-Marie Jacquard zur Automatisierung der Webtechnik eingesetzt wurde. Die analytische Maschine war so allgemein konzipiert, dass sie in der Lage gewesen wäre, selbst komplexe Kontrollstrukturen wie Schleifen, Unterprogrammaufrufe und bedingte Sprünge abzubilden. Leider war der Entwurf nicht nur konzeptionell, sondern auch in seiner Komplexität der damaligen Zeit weit voraus und die analytische Maschine wurde ebenfalls nie vollendet.

Trotz seiner unbestrittenen Erfolge in verschiedenen Gebieten der Wissenschaft war Babbage am Ende seines Lebens tief von dem Scheitern seiner drei Großprojekte gezeichnet. Mit der britischen Regierung, die jeder weiteren finanziellen Förderung eine Absage erteilte, ging er noch zu Lebzeiten hart ins Gericht. Am 18. Oktober 1871 starb Babbage in seinem Haus in London als enttäuschter und verbitterter Mann im Alter von 79 Jahren. 150 Jahre nach ihrer Erfindung schaffte zumindest die Differenzenmaschine 2 dann doch noch den Sprung vom Reißbrett in die Realität. Am Londoner Science Museum wurde die Maschine nach Originalplänen rekonstruiert. 1991 wurden die Arbeiten an der funktionsfähigen Maschine beendet – pünktlich zum 200. Geburtstag ihres Erfinders.

Die elektrische Revolution

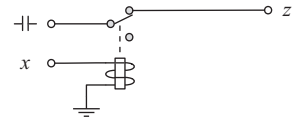
Die Verwendung elektromechanischer Bauteile stellt einen Quantensprung in der Computergeschichte dar. Mit der elektrischen Spannung als Informationsträger war auf einen Schlag die Konstruktion von Rechenmaschinen möglich, die in ihrer Komplexität und Zuverlässigkeit weit über das hinausgingen, was rein mechanisch arbeitende Apparaturen je zu leisten im Stande gewesen wären. Die ersten Rechenmaschinen dieser Art wurden Anfang der Vierzigerjahre gebaut und verwendeten zunächst *elektromechanische Relais* zur Daten- und Kontrollflusssteuerung. Das Relais, das schon kurz nach seiner Erfindung im Jahre 1835 durch Joseph Henry mit der Telegraphie das Kommunikationszeitalter einläutete, löste damit rund hundert Jahre später eine weitere technologische Revolution aus.

Zeitgleich mit dem Relais, das nur die zwei Zustände *offen* und *geschlossen* unterscheidet, hält das *Binärsystem* endgültig Einzug in die Computertechnik und bildet bis heute die Grundlage aller maßgebenden Computerarchitekturen. Abbildung 1.8 zeigt, wie sich die logischen Grundoperationen mit Hilfe konventioneller elektromechanischer Relais konstruieren lassen. In Kapitel 4 werden wir auf die verschiedenen Logikoperationen im Detail eingehen und in den darauf folgenden Kapiteln zeigen, wie sich unter alleiniger Verwendung der Elementarverknüpfungen NOT, AND und OR Hardware-Schaltungen mit nahezu beliebiger Komplexität in die Praxis umsetzen lassen.

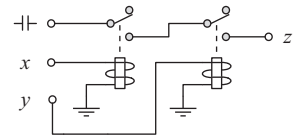
Die legendäre Z3 des Konrad Zuse

Konrad Zuse konstruierte 1941 mit der Z3 den ersten voll funktionsfähigen Rechner dieser neuen Generation [96]. Im Gegensatz zu der noch weitgehend mechanisch arbeitenden Z1 und dem Versuchsmodell Z2, die aufgrund technischer Mängel beide nur unzuverlässig ihren Dienst verrichteten, verwendete Zuse in der Z3 ausschließlich elektromechanische Relais. Die Maschine war aus ca. 2000 Einzel-Relais aufgebaut und konnte mit einer Taktfrequenz von 5 bis 10 Hertz betrieben werden. Insgesamt kam die Z3 auf ein Gesamtgewicht von ca. 1000 kg und besaß eine Leistungsaufnahme von 4000 Watt. Nicht nur aus der Sicht der Ingenieurkunst war die Z3 bis dato einzigartig – sie war vor allem auch aus konzeptioneller Sicht ein Meisterwerk. Viele Konzepte, die sich heute in modernen Computerarchitekturen wiederfinden, waren bereits in der Z3 vorhanden, wenngleich auch in deutlich primitiverer Form:

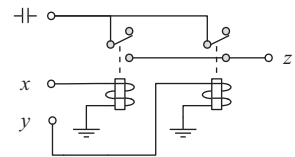
■ Die NOT-Verknüpfung:



■ Die AND-Verknüpfung:



■ Die OR-Verknüpfung:



■ Der Relais-basierte Zustandsspeicher:

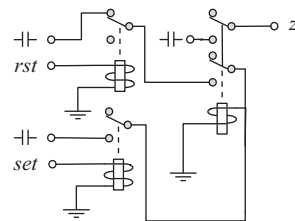


Abbildung 1.8: Die Basiskomponenten eines Relais-Rechners

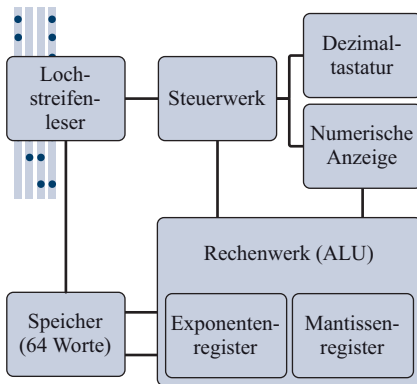


Abbildung 1.9: Blockschaltbild der Z3

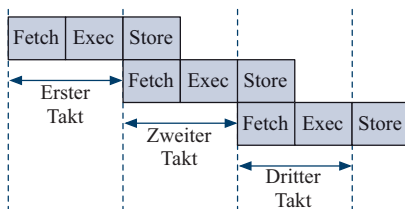


Abbildung 1.10: Die Pipeline-Architektur der Z3

| Befehl | Bedeutung | Codierung |
|-----------------------|----------------|-----------|
| Ein- und Ausgabe | | |
| Lu | Eingabe | 01110000 |
| Ld | Ausgabe | 01111000 |
| Speichertransfer | | |
| Pr z | Lesen | 11Adresse |
| Ps z | Schreiben | 10Adresse |
| Arithmetik | | |
| Lm | Multiplizieren | 01001000 |
| Li | Dividieren | 01010000 |
| Lw | Wurzelziehen | 01011000 |
| Ls₁ | Addieren | 01100000 |
| Ls₂ | Subtrahieren | 01101000 |

Tabelle 1.1: Der Befehlssatz der Z3

- Wie das Blockschaltbild in Abbildung 1.9 demonstriert, sind der *Speicher* und der *Prozessor* – bestehend aus *Steuer-* und *Rechenwerk* – klar voneinander getrennt. Diese Zweiteilung findet sich auch heute noch in den allermeisten Computerarchitekturen wieder.
- Intern verwendet die Z3 das *Gleitkommaformat* zur Darstellung rationaler Zahlen. Über die numerische Tastatur eingegebene Operanden wurden automatisch in das interne Gleitkommaformat übersetzt und die Rechenergebnisse für die Anzeige ebenfalls vollautomatisch in das Dezimalsystem zurückkonvertiert. Damit war die Erfindung von Zuse auch hier anderen Maschinen weit voraus, die mit dem *Festkommaformat* eine wesentlich primitivere Art der Darstellung einsetzten. In Kapitel 3 werden wir uns intensiv mit den Unterschieden des Gleit- und Festkommaformats beschäftigen.
- Das Prinzip der *Mikroprogrammierung*, das auch heute die Grundlage vieler moderner Mikroprozessoren bildet, findet sich bereits in der Z3 verwirklicht. Im Zentrum des Steuerwerks befindet sich hierzu der sogenannte *Mikrosequenzer*, mit dessen Hilfe sich komplexe Rechenoperationen in mehrere elementare Berechnungsschritte zerlegen und nacheinander ausführen ließen. Im Zusammenhang mit der CISC- und RISC-Architektur werden wir in Kapitel 12 auf dieses Prinzip der Ablaufsteuerung zurückkommen.
- Zur Geschwindigkeitssteigerung führt die Z3 den Befehlsstrom *überlappend* aus, d. h. während des Zurückschreibens des Ergebnisses ist der Rechner in der Lage, bereits die nächste Instruktion einzulesen. Wie in Abbildung 1.10 gezeigt, folgte die Z3 damit einer dreistufigen *Pipeline-Architektur*, die mit einem einstufigen Versatz ausgeführt wurde. In Kapitel 12 werden wir uns auch mit diesem Architekturprinzip genauer beschäftigen.
- Auch das verwendete Rechenwerk musste zu seiner Zeit keinen Vergleich scheuen. So setzte die Z3 zur Beschleunigung der Addition die *Carry-look-ahead-Technik* ein, die auch heute noch die Grundlage vieler Addierwerke bildet. In Kapitel 7 werden wir dieses Additionsprinzip im Detail kennen lernen.

Programmiert wurde die Z3 mit Hilfe eines 8-spurigen *Lochstreifens*. Jede Zeile codiert einen einzigen Befehl, zusammen mit den dazugehörigen Operanden. Wie die Befehlsübersicht in Tabelle 1.1 zeigt, lassen sich die insgesamt neun Befehle in die Kategorien *Ein- und Ausgabe*, *Speichertransfer* und *Arithmetik* unterteilen.

Mit Hilfe der Befehle **Lu** bzw. **Ld** wird eine Dezimalzahl über die numerische Tastatur eingelesen bzw. auf der numerischen Anzeige aus-

gegeben. Beide Befehle stoppen die Ausführung der Maschine, bis der Benutzer manuell die Fortsetzung veranlasst. Die Kommunikation mit dem Hauptspeicher wird über die Befehle **Pr** bzw. **Ps** gesteuert. Die anzusprechende Speicheradresse wird hierzu in den 8-Bit-Opcode der Befehle hineincodiert. Mit den 6 zur Verfügung stehenden Adressbits lassen sich insgesamt $2^6 = 64$ Speicherworte adressieren. Die restlichen Befehle dienen zur Durchführung der vier arithmetischen Grundrechenarten sowie der Berechnung der Quadratwurzel.

Betrachten wir den Befehlssatz der Z3 genauer, so fällt auf, dass mit den vorhandenen Befehlen weder eine Verzweigung noch eine Schleife programmiert werden kann. Der Programmierung sind hierdurch enge Grenzen gesetzt und die Z3 stellt damit keinen Universalrechner im eigentlichen Sinne dar.

Die Harvard Mark I

Fast zeitgleich mit dem Bau der Z3 wurde an den IBM Laboratories in New York unter der Regie von Howard H. Aiken der größte jemals fertiggestellte elektromechanische Rechner gebaut – der *Automatic Sequence Controlled Calculator* (ASCC) [17]. In einer feierlichen Zeremonie wurde die gigantische Maschine am 7. August 1944 an die Harvard University übergeben und offiziell in Betrieb genommen. Die Komplexität der Konstruktion, die fortan den Namen *Harvard Mark I* trug, lässt sich anhand weniger Kenndaten bereits erahnen. So bestand die gesamte Apparatur aus ca. 765.000 Einzelkomponenten, die über etliche Kilometer Kabel miteinander verbunden waren. Insgesamt füllte die Konstruktion mit 6 Metern Länge und über 2 Metern Höhe eine Werkhalle vollständig aus.

Die Harvard Mark I enthielt 72 *Akkumulatoren*, die jeder für sich eine dezimalcodierte Zahl mit einer Genauigkeit von 23 Ziffern speichern konnten. Neben einem weiteren Kurzzeit- sowie einem Konstantenspeicher verfügte der Rechner über separate Multiplikations- und Divisions-einheiten. Gesteuert wurde die Harvard Mark I über 24-spurige Lochstreifen, deren allgemeiner Aufbau in Abbildung 1.12 dargestellt ist. Jede Zeile codiert einen einzelnen Steuerbefehl, der sich aus drei Komponenten zusammensetzt. So definieren die Einstanzungen der ersten 8 Spuren die Quelladresse und die Einstanzungen der mittleren 8 Spuren die Zieladresse einer Operation. Die restlichen 8 Spuren definieren den Opcode des auszuführenden Befehls. Die Harvard Mark I konnte die so codierten Befehle ausschließlich sequenziell abarbeiten – Befehle für Schleifen, Verzweigungen und Unterprogrammaufrufe sah die Rechne-

Konrad Zuse (1910 – 1995)

Abbildung 1.11: Bronze-Statue in Hünfeld zum Gedenken an Konrad Zuse

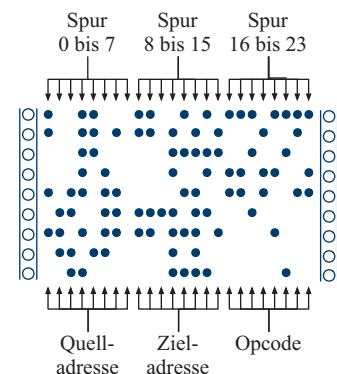
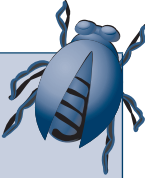


Abbildung 1.12: Aufbau des Lochstreifens zur Steuerung der Harvard Mark I



Dass wir heute im Software- und Hardware-Bereich Fehler gemeinhin als *Bugs* bezeichnen, wird in vielen Quellen in Zusammenhang mit der Harvard Mark I gebracht. Gerüchten zufolge geriet eine Motte in eines der zahlreichen elektrischen Relais des Rechners und brachte auf diese Weise die gesamte Maschine zum Erliegen. Die Motte wurde entfernt und der Begriff *Debugging* war geboren. In der Tat gehört die Geschichte zu den klassischen Mythen des Computerzeitalters. Die besagte Motte gab es wirklich, allerdings stammte sie nicht aus der Harvard Mark I, sondern von ihrem Nachfolger, der Harvard Mark II. Der Begriff des *Debuggings* wurde jedoch bereits viel früher geprägt, wie z. B. der folgende Ausschnitt aus einem Brief von Thomas Edison vom 18. November 1878 an Theodore Puskas verrät:

„It has been just so in all my inventions. The first step is an intuition – and comes with a burst, then difficulties arise. This thing gives out and then that – 'Bugs' – as such little faults and difficulties are called – show themselves and months of anxious watching, study and labor are requisite before commercial success – or failure – is certainly reached“ [47]

Nichtsdestotrotz war die Motte der Harvard Mark II wahrscheinlich der erste richtige *Bug* der Computergeschichte. Howard Aiken selbst bezeichnete sie als den „first actual case of bug being found“.

rarchitektur noch nicht vor. Damit ließen sich Endlosschleifen nur auf physikalischem Wege durch das Zusammenfügen des Lochstreifens zu einem endlosen Band realisieren.

Das bereits in der Harvard Mark I umgesetzte Konzept, Code und Daten in getrennten Speichern vorzuhalten, hat im Bereich moderner Computerarchitekturen in den letzten Jahren eine wahre Renaissance erlebt. So arbeiten z. B. moderne digitale Signalprozessoren fast ausschließlich nach diesem Prinzip, da sich im Vergleich zur klassischen Von-Neumann-Architektur deutlich höhere Übertragungsraten zwischen der Zentraleinheit und dem Speicher erreichen lassen. In Anlehnung an ihre historischen Wurzeln sprechen wir in diesem Zusammenhang heute immer noch von der *Harvard-Architektur*.

Die Harvard Mark I wurde erst 1959 außer Betrieb gestellt und anschließend zerlegt. Einige Komponenten sind heute noch im *Cabot Science Scenter* der Harvard University zu sehen, der Rest befindet sich im Besitz der IBM Laboratories in New York und des Smithsonian-Instituts in Washington, D.C.

Die ENIAC

Die Frage, ob wir in der Z3 und der in etwa zeitgleich entwickelten Harvard Mark I die ersten richtigen *Computer* vor uns sehen dürfen, ist immer wieder Gegenstand hitziger Diskussionen und in Fachkreisen umstritten. Einige Experten sehen in der freien Programmierbarkeit und der konzeptionellen Untergliederung der Maschinen in Speicher, Rechenwerk und Steuerwerk die wesentlichen Grundzüge eines Computers verwirklicht. Andere Fachleute sind der Meinung, dass die *Universalität* des zu Grunde liegenden Berechnungsmodells das wesentliche Charakteristikum des Computers darstellt. Dieser Interpretation folgend, gelten sowohl die Z3 als auch die Harvard Mark I zwar als die ersten universellen Rechenmaschinen, nicht jedoch als Computer im modernen Sinne.

Den Bau der ersten voll funktionsfähigen Rechenmaschine, die nahezu allen Definitionen des modernen Computer-Begriffs standhält und daher von vielen Experten als der erste wirkliche Computer der Welt angesehen wird, datieren wir auf das Jahr 1946 – das Jahr, in dem die ENIAC¹ der Öffentlichkeit vorgestellt wurde [36, 86]. Der Rechnerkoloss wurde an der Moore School of Electrical Engineering der University of Pennsylvania unter der Leitung von J. Presper Eckert und John

¹Electronic Numerical Integrator And Computer

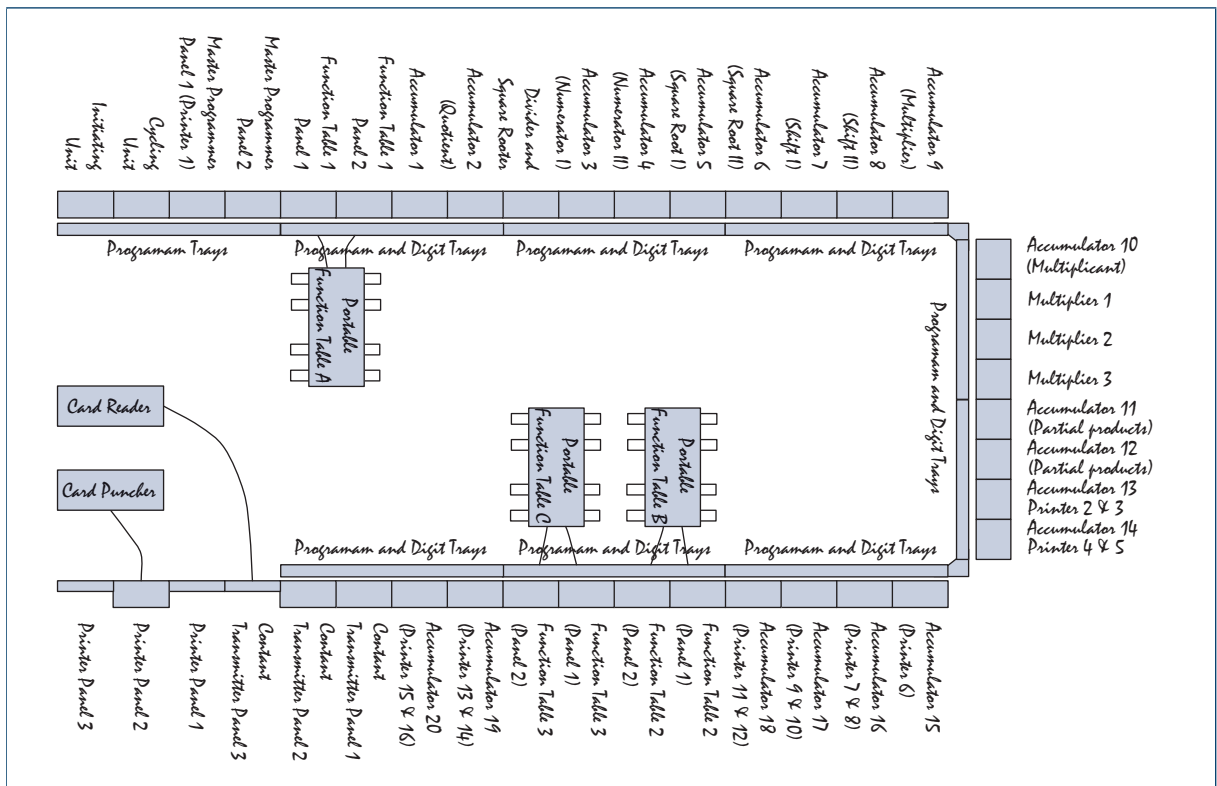


Abbildung 1.13: Der „Floorplan“ der ENIAC

W. Mauchly gebaut und beeindruckte schon aufgrund seiner schieren Größe. Wie in Abbildung 1.13 dargestellt, bestand die ENIAC aus insgesamt 30 Einheiten, die U-förmig über den gesamten Raum verteilt angeordnet waren. Die gesamte Konstruktion kam auf ein Gesamtgewicht von knapp 30 Tonnen (Abbildungen 1.15 bis 1.17).

Der Entschluss, einen Rechner dieses Ausmaßes überhaupt zu bauen, wurde aus der Not geboren. Die Intensität, mit der der Zweite Weltkrieg 1939 über die Welt hereinbrach, setzte die U.S.-Armee unter enormen Zeitdruck, sich auf den immer wahrscheinlicher werdenden Kriegseintritt vorzubereiten. Unter anderem experimentierten die U.S.-Streitkräfte mit großkalibrigen ballistischen Geschützen, die anhand von Trajektorien-Tabellen auf ihr Ziel ausgerichtet wurden. Das Aufstellen der Tabellen erforderte das Lösen von Differenzialgleichungen zweiter Ordnung und damit Fähigkeiten außerhalb des Leistungsspektrums damaliger Rechenmaschinen. Die ENIAC, mit deren Hilfe die

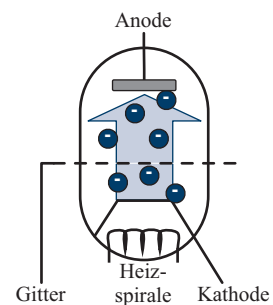


Abbildung 1.14: Die Trioden-Röhre als Schaltelement. Der durch die Heizspirale verursachte Stromfluss von der Kathode zur Anode kann durch das Anlegen einer Spannung an das Metallgitter blockiert oder verstärkt werden – ein einfaches Schaltelement entsteht.

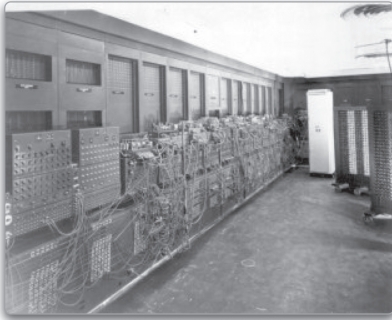


Abbildung 1.15: Teilansicht der ENIAC (U.S.-Armee-Foto)

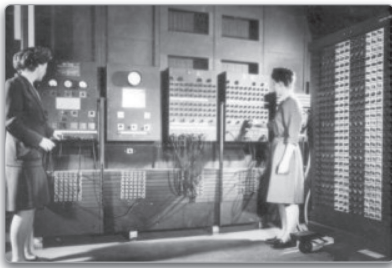


Abbildung 1.16: Das *Main Control Panel* der ENIAC (U.S.-Armee-Foto)

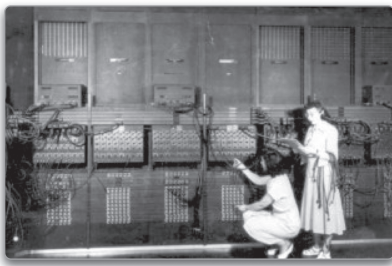


Abbildung 1.17: Programmiert wurde die ENIAC durch „Kabelstecken“ (U.S.-Armee-Foto)

automatisierte Berechnung der Trajektorien-Tabellen möglich werden sollte, war in zweierlei Hinsicht revolutionär:

- Im Gegensatz zur Z3 oder der Harvard Mark I war es möglich, Verzweigungen und Schleifen zu programmieren. Damit unterscheidet sich das formale Berechnungsmodell der ENIAC deutlich von ihren Vorgängern und ist insbesondere mit dem heutiger Computer vergleichbar, auch wenn die Programmierung des Kolosses im Vergleich zur heutiger Technik kaum unterschiedlicher sein könnte.
- In der ENIAC wurde auf die Verwendung elektromechanischer Relais verzichtet und die Schaltlogik stattdessen mit Hilfe von *Vakuumröhren* implementiert. Zwar ist die ENIAC nicht der erste Röhrenrechner der Welt, aber der erste Großrechner, der die Vakuumröhre in Form der *Triode* konsequent als Schaltelement verwendete. Im Vergleich zum Relais zeichnet sich die Röhre durch eine um den Faktor 1000 bis 2000 gesteigerte Schaltgeschwindigkeit aus, so dass die ENIAC mit einer für damalige Verhältnisse beeindruckenden Taktfrequenz von 100 Kilohertz betrieben werden konnte. Eine einzige Multiplikation berechnete die ENIAC in nur knapp 3 Millisekunden.

Auf der negativen Seite brachte die ENIAC konstruktionsbedingt auch gravierende Nachteile mit sich. Ein wesentliches Element der Röhrentriode ist, wie in Abbildung 1.14 skizziert, die Heizspirale. Wird sie zum Glühen gebracht, so beginnt die Kathode Elektronen zu emittieren, die von der Anode angezogen und aufgefangen werden. Konstruktionsbedingt sind Vakuumröhren damit äußerst stromhungrige Bauelemente. Die Leistungsaufnahme der ca. 18.000 in der ENIAC verbauten Vakuumröhren summierte sich auf sagenhafte 174.000 Watt. Des Weiteren besitzen Vakuumröhren im Vergleich zu Relais eine sehr begrenzte Lebensdauer und das langwierige Auffinden und Austauschen defekter Röhren gehörte zur täglichen Arbeit eines ENIAC-Ingenieurs.

Der wohl gravierendste Nachteil betrifft jedoch die Handhabung der Maschine, die nicht auf eine flexible Programmierung ausgelegt war. Der Datenpfad wurde mit Hilfe von Steckverbindungen fest verdrahtet und ein Programm für die ENIAC war damit nichts anderes als ein Verbindungsplan, der die benötigten Steckverbindungen beschrieb. Hierdurch war ein flexibler Wechsel zwischen verschiedenen Programmen von vornherein ausgeschlossen.

Die Limitierungen der ENIAC inspirierten den in Budapest geborenen und später in die USA immigrierten Wissenschaftler John von Neumann

Sachwortverzeichnis

Symbole

2-Bit-Prädiktion, 393
2-aus-5-Code, 84
2421-Code, 83
7-Segment-Anzeige, 80, 135
74210-Code, 84, 85
8421-Code, 81

A

Abakus, 13, 415
Abgeleiteter Operator, 97
Abhängige Variable, 94
Absolute Adressierung, 381
Absoluter Sprung, 317
Absorptionsgesetz, 109
Addierer, 218, 415
 Carry-look-ahead-, **221**, 263
 Carry-ripple-, **220**, 263
 Carry-save-, 229
 Conditional-Sum-, 224
 Präfix-, 227
 serieller, 333
Additionssystem, 60, 415
Adressbus, 356, 360, 415
Adressdecoder, 320
Adresse, 254
Adressierung
 absolute, 381
 indirekte, 381
 postinkrementierende, 386
 relative, 381
 speicherindirekte, 382
Adressierungsart, 381, 415
Adressierungsgranularität, 318
Adressmodifikator, 347
Adressmultiplexing, 323
Adresspin, 323
Äquivalenz, 100
 Äquivalenz-Operation, 99
Aiken-Code, 81, 83
Akkumulator, 305, 337, 415
Akzeptor, 40, 286, 415
Algorithmus
 Euklidischer, 375
Allgemeingültigkeit, 100
Allzweckregister, 415
Alphabet
 Ausgabe-, 286
 Eingabe-, 286
Alpha-Teilchen, 328, 334
Analytische Maschine, 16
Anstiegszeit, 170
Antivalenz-Operation, 99
Antivalenzfunktion, **125**, 157, 215
Arabisches System, 61
Architektur
 Big-Endian-, 68
 EM64T-, 380
 Harvard-, 20, **353**, 361
 IA-32-, 380
 IA-64-, 380
 Little-Endian-, 68
 Load-Store-, 385
 PowerPC-, 386
 Von-Neumann-, 352
 x86-, 379
Arithmetisch-logische Einheit, 251, 415
ASCC, 19
Assembler, 353, 415
Assembler-Sprache, 416
Assoziativer Cache, 396
Assoziativgesetz, 110
Asynchroner Zähler, 313, 330
Asynchrones RS-Latch, 267
Atom, 34
Auffangregister, 300, 416
Auflösungsgenauigkeit, 75, 77
Aufzugssteuerung, 343, 350

Ausdruck
 äquivalenter, 100
 allgemeingültiger, 100
 boolescher, 96
 erfüllbarer, 100
 tautologischer, 100
Ausführungsphase, 359
Ausgabealphabet, 286
Ausgabefunktion, 286
Ausgaberegister, 337
Ausgabeschaltnetz, 290
Ausgangslastfaktor, 155
Ausgangssignal, 140
Automat
 Akzeptor, 286
 endlicher, 286
 Mealy-, 288
 Moore-, 288
 Transduktor, 286
Axiome
 von Huntington, **90**
 von Robbins, 91

B

Back-end of line, 54
Backus-Naur-Form, 356
Bändermodell, 36
Bank, 324
Bank select, 326
Barrel-Shifter, **249**, 259, 416
Basis, 43, 61, 416
Basistechnologie, 140
 ECL, 141
 MOS, 141
 TTL, 140
Baumstruktur, 96
BCD-Code, **81**, 192, 259, 416
BDD, **125**, 158
Bedingter Sprung, 343, 363

Befehlsausführung
 spekulative, 392
 Befehlsdecoder, 251
 Befehlsholphase, 359
 Befehlssatz, 352, 416
 Benchmark, 402, 416
 Dhrystone-, 403
 Eispack-, 404
 Lapack-, 404
 Linpack-, 401, 404
 natürlicher, 403
 SPEC-, 404
 synthetischer, 403
 Whetstone-, 403
 Benchmark-Kollektion, 402
 Berechnungsmodell
 universelles, 352, 363
 Bevorrechtigte Eingänge, 416
 Bevorrechtigter Eingang, 281
 Bidirektionaler Zähler, 308
 Big-Endian, 416
 Big-Endian-Architektur, 68
 Bindungsenergie, 36, 416
 Bindungslücke, 38
 Bindungspriorität, 99
 Binärcode, 416
 Binärcodiertes Dezimalsystem, 80
 Binäres Entscheidungsdiagramm, **125**,
 158, 416
 geordnetes, 126
 reduziertes, 127
 Binärsystem, 62, 416
 Binärzähler, 308
 Bipolartransistor, 416
 Biquinär-Code, 84, 85
 Bit, 67, 416
 Blockbildung, 189
 inverse, 193
 Blockmultiplikation, 338, 349, 417
 Blue tape, 55
 Bohr'sches Atommodell, 34
 Boolesche Algebra, 90, 417
 Mengenalgebra, 91
 Schaltalgebra, 90, 93
 Boolesche Differenz, 130
 Boolesche Funktion, 94, 417
 Boolesche Konstanten, 417
 Boolescher Ausdruck, 96, 417

Branch prediction table, 393
 Brown'sche Molekularbewegung, 38
 Buffer, 170
 Bug, 20
 Bus, 417
 Bus-Knoten, 355
 Bus-Topologie, 355
 Byte, 67, 417

C

Cache, 393
 assoziativer, 396
 -Block, 394
 -Controller, 393
 direkt abgebildeter, 394
 -Hit, 393, 395
 Level-*n*-, 394
 -Miss, 393, 395
 -Speicher, 417
 vollassoziativer, 409
 Carry-Bit, 218, 358, 364
 Carry-look-ahead-Addierer, **221**, 263,
 417
 Carry-ripple-Addierer, **220**, 263, 417
 Carry-save-Addierer, 229, 417
 Carry-save-Format, **229**, 238, 417
 Carry-save-Multiplizierer, 238, 417
 Cell delay, 170
 Cell-Prozessor, 30
 Central processing unit, 27, 352, **356**
 Charakteristik, 75, 417
 Charakteristische Funktion, 163, 418
 Chipausbeute, 53
 Chip select, 326
 Church'sche These, 363
 CISC, 405, 418
 CISC-Prozessor, 380
 Clock enable, 281
 CMOS-Schaltung, 151, 282
 CMOS-Technik, 418
 Code, 418
 2-aus-5-, 84
 2421-, 83
 74210-, 84, 85
 8421-, 81
 Aiken-, 81, 83

BCD-, **81**, 192, 259
 Biquinär-, 84, 85
 einschränkter, 83
 Excess-3-, 81
 fehlererkennender, 84
 Glixon-, 88
 Gray-, 81, **83**, 187, 288, 291
 Hamming-, 329
 m-aus-*n*-, 85
 mehrschrittiger, 83
 One-Hot-, 84, **85**, 368
 progressiver, 83
 reflektierter Biquinär-, 84, 85
 Stibitz-, 81
 Walking-, 85
 Code-Distanz, 85
 Column address strobe, 323
 Compiler, 353
 Conditional-Sum-Addierer, 224, 418
 Core microarchitecture, 30
 CPI-Wert, 401
 CPU, 27, 352, **356**
 Current window pointer, 406

D

D-Flipflop, 276
 D-Latch, 272
 Dadda-Tree-Multiplizierer, 246, 418
 Data line, 320
 Datenbus, 355, 418
 Datenfluss, 356
 rückgekoppelter, 337
 Datenregister, 365
 Datenspeicher, 361
 Datenwort, 336
 Davio-Entwicklung
 negative, 130
 positive, 130
 DCTL, 142
 DDR-RAM, 326
 De Morgan'sche Regel, 107, 111
 erweiterte, 134
 Decode, 359, 388
 Decoder, 212, 346
 Decodierphase, 359
 Defektdichte, 53

Defektelektron, 38
 Definitorische Form, 418
 Delay
 Cell, 170
 Net, 171
 Demultiplexer, **211**, 257, 418
 Dezimalsystem, 61, 418
 binärcodiertes, 80
 Dhrystone-Benchmark, 403
 Dicing, 55
 Die, 55
 Differenzenmaschine, 15
 Differenzenmethode, 15
 Digitaler Signalprozessor, 74, 251
 Digitaltechnik, 140
 DIMM, 321
 DIN
 40900, 157
 66000, 97
 Diode, 41, 418
 Direkte Kommunikation, 354
 Direkter Sprung, 363
 Disjunktion, 93
 Disjunktions-Matrix, 253
 Disjunktive Minimalform, 418
 Disjunktive Normalform, 119, 212, 419
 Displacement, 382, 419
 Distanz
 Code-, 85
 Hamming-, 85, 329
 Distributivgesetz, 91
 Division, 259
 Don't-Care-Belegung, 192, 199, 419
 Donator, 39
 Doppelnegationsgesetz, 110
 Dot diagram, 244
 Dotierung, 39, 419
 Double-precision-Format, 77
 Drain, 47
 DRAM, 27, 320, 419
 DTL, 142
 Duale Gleichung, 107
 Dualer Operator, 108
 Dualitätsprinzip, 105, 107, 419
 Dynamische Sprungvorhersage, 393
 Dynamischer Hazard, 419
 Dynamischer Speicher, 320

E

E-Flipflop, 297
 ECL-Technik, 141
 Eigenleitung, 39
 Ein-/Ausgabebaustein, 354
 Einerkomplement, 70, 419
 Eingabealphabet, 286
 Eingaberegister, 337
 Eingang
 bevorrechtigter, 281
 -signal, 140
 -slastfaktor, 155
 Einkristall, 38
 Einschrittiger Code, 83
 Einsmenge, 120, 419
 Einzelkernprozessor, 352
 Eispack-Benchmark, 404
 Electronic Design Automation, 171
 Elektron
 Paarbildung, 38
 Rekombination, 38
 Elektronenloch, 38
 Elektronenmangelleiter, 40
 Elektronenstrom, 39
 Elektronenüberschussleiter, 39
 Elementaroperatoren, 419
 Eliminationsgesetz, 109
 ELSI, 25
 EM64T, 419
 EM64T-Architektur, 380
 Emitter, 43
 Emitter-Basis-Strecke, 43
 Emitter-Kollektor-Strecke, 43
 Enable-Eingang, 213
 Encoder, 346
 Endlicher Automat, 286, 419
 Endlosschleife, 368
 ENIAC, 20
 Entscheidungsdiagramm, 419
 Binäres, **125**, 158
 funktionales, **130**, 159
 Entwicklungssatz
 von Shannon, 127
 Erfüllbarkeit, 100
 Erholzeit, 323
 ESI, 386

Espresso, 199
 Euklidischer Algorithmus, 375
 Excess-3-Code, 81
 Execute, 359, 388
 Exponent, 74, 420
 Extended-precision-Format, 79
 Extra-large-scale integration, 25

F

Fallzeit, 170
 Fan-In, 155
 Fan-Out, 155
 FDD, **130**, 159
 FDIV-Bug, 74
 Feedback loop, 282
 Fehlererkennender Code, 84, 420
 Fehlererkennung, 327
 Fehlerkorrektur, 327
 Fehlerkorrigierender Code, 420
 Feldeffekttransistor, 47, 420
 Ferritkernspeicher, 24
 Festkommaformat, 18, 73, 420
 Festwertspeicher, 254
 FET, 47
 Fetch, 359, 388
 Finalzustand, 286
 Flächenbedarf, 184
 Flanke, 170
 Flankensteuerung, 274
 negative, 275
 positive, 275
 Flaschenhals
 Von-Neumann-, 356
 Fließbandprinzip, 388
 Fließkommazahl, 74
 Flip-Chip-Verfahren, 56, 425
 Flipflop, 274, 420
 D-, 276
 E-, 297
 JK-, 279
 Master-, 276
 RS-, 277
 Slave-, 276
 T-, 278
 Floating state, 145
 Flynn-Taxonomie, 378, 420

Folgeadresse, 347
 Formelsynthese, 161, 420
 definitorisches, 164
 funktionale, 161
 relationale, 163
 Freie Variable, 94
 Frequenzteiler, 310
 Front-end of line, 53
 Funktion
 charakteristische, 163
 partielle, 190
 Funktionales Entscheidungsdiagramm,
 130, 159, 420
 Funktionseinheit, 337
 Funktions-Hazard, 174, 421
 Funktionstabelle, 94

G

Gate, 47
 Gatter, 156, 421
 Gatternetzliste, 157
 Gauß'sche Summenformel, 104, 223
 Gesetz von Moore, 32
 ggT, 375
 Gibi, 64
 Giga-scale integration, 25
 Gleichheitstest, 69
 Gleichung
 duale, 107
 Gleitkommadivision, 74
 Gleitkommaformat, 18, 74, 421
 Glixon-Code, 88
 GPU, 74, 407
 Grafikprozessor, 74, 407
 Grammatik, 96
 Gray-Code, 81, **83**, 187, 288, 291
 GSI, 25

H

Halbaddierer, 218, 421
 Halbleiter, 34, 421
 dotierter, 39
 reiner, 37
 Hamming-Code, 329
 Hamming-Distanz, 85, 329, 421

Hamming-Würfel, 85
 Handshaking, 325
 Handshaking-Protokoll, 349
 Hardware-Schaltung, 140
 Hazard-freie, 194
 Stromverbrauch, 174, 184
 Zeitverhalten, 169
 Hardware-Schleife, 382
 Harvard-Architektur, 20, **353**, 361, 421
 Harvard Mark I, 19
 Hazard, 171, 194, 421
 -frei, 194
 dynamischer, 172
 funktionaler, 174
 Logik-, 194
 logischer, 172
 Pipeline-, 391
 statischer, 171
 Hexadezimalsystem, 63, 421
 High-Pegel, 142
 High-Pegelbereich, 142
 Hilfsregister, 357
 Hitzesensor, 384
 Hochintegration, 140
 Huffman-Normalform, 290, 421
 Huntington'sche Axiome, **90**, 421

I

I/O, 354
 IA-32-Architektur, 380, 422
 IA-64-Architektur, 380, 422
 Idempotenzgesetz, 109
 IEEE 754, 422
 IEEE-754, 77
 Implikationsoperator, 99, 118
 Indirekte Adressierung, 381
 Indirekte Kommunikation, 355
 Individualisierung, 253
 Induktion
 strukturelle, 102
 vollständige, 102
 Ingot, 51, 422
 Initialzustand, 286
 Inkrementierer, 232, 422
 Input/Output, 354
 Instabiler Zustand, 269

Instruktionsarchitektur, 379, 422
 CISC-, 380
 RISC-, 384
 Instruktionsdecoder, 356, **368**, 376, 422
 Instruktionsregister, 365
 Instruktionszähler, 316, **358**, 365, 422
 Integrationsdichte, **57**, 58, 422
 Integrierter Schaltkreis, 26
 Intel, 27
 Interleaving, 324
 Interrupt, 328
 Intervallgrenze, 75
 Inverses Element, 91
 Inversionszone, 49
 Ion, 35
 IPC-Wert, 401

J

JFET, 47
 JK-Flipflop, 279

K

Kanallänge, **57**, 58, 422
 Karnaugh-Veitch-Diagramm, 186, 422
 Kerbensystem, 60, 422
 kgV, 375
 Kibi, 64
 Koeffizienten-Matrix, 235
 Kofaktor, 422
 negativer, 128
 positiver, 128
 Kollektor, 43
 Kombinatorische Schaltung, 157
 Kommunikation
 direkte, 354
 indirekte, 355
 Kommutativgesetz, 91
 Komparator, 213, 259, 422
 Komplexgatter, 175
 Komplexitätsanalyse, 167
 Kondensator, 320, 327
 Konjunktion, 93
 Konjunktions-Matrix, 253
 Konjunktive Minimalform, 423
 Konjunktive Normalform, 119, 423

Konklusion, 99
 Konsistenzfunktion, 164
 Kontrollbus, 356
 Kontrollfluss, 356
 Korrektur-Tetrade, 82
 Kostenfunktion, 184, 423
 Kreuzprodukt, 407
 Kristallgitter, 37
 KV-Diagramm, 186
 dreidimensionales, 196

L

Label, 348
 Ladung, 320, 327
 Lapack-Benchmark, 404
 Large-scale integration, 25
 Lastfaktor, 155
 Latch, 274, 423
 D-, 272
 RS-, asynchrones, 267
 RS-, synchrones, 272
 Laufzeit, 184
 Layer, 54
 Leckstrom, 151, 321
 Leistungsbewertung, 399
 Leitungsband, 36
 Leitungselektron, 36
 Level-*n*-Cache, 394
 Linpack-Benchmark, 401, 404
 Literal, 119, 423
 Little-Endian, 423
 Little-Endian-Architektur, 68
 Load-Store-Architektur, 385, 423
 Löcherstrom, 39
 Lochkarte, 16
 Lochstreifen, 18
 Logik
 -ebene, 156
 -gatter, 156
 -polarität, 144
 -zelle, 156
 negative, 143
 positive, 143
 Logik-Hazard, 172, 194, 423
 Logikpolarität, 423
 Low-Pegel, 142

Low-Pegelbereich, 142
 LSI, 25

M

m-aus-n-Code, 85
 Mainframe, 25
 Majoritätsträger, 40
 Makro, 368
 Manchester Mark I, 23
 Mantisse, 73, 423
 Maschinenbefehl, 423
 Master-Flipflop, 276
 Master-Slave-Flipflop, 276, 423
 Matrix
 Disjunktions-, 253
 Koeffizienten-, 235
 Konjunktions-, 253
 Matrixmultiplizierer, 235, 261, 423
 Maxterm, **119**, 424
 Mealy-Automat, 288, 424
 Mebi, 64
 Medium-scale integration, 25
 Mehrkernprozessorsystem, 352
 Mehrphasentaktgeber, 365, 376
 Mehrschrittiger Code, 83
 Mengenalgebra, 91
 MFLOPS, 401
 Micromosaic, 25
 Mikroprogramm, 347, 384
 Mikroprogrammierung, 18, 343, 424
 Mikroprozessor, 27, 352, **356**, 424
 Mikrorechner, 352
 MIMD, 378
 Minicomputer, 26
 Minimierung, 424
 grafische, 186
 mehrstelliger Funktionen, 196
 partieller Funktionen, 190, 199
 tabellarische, 197
 Minimierungsziel, 182
 Minoritätsträger, 40
 Minterm, **119**, 212, 253, 424
 MIPS, 401
 Mischzähler, 314
 MISD, 378
 Mnemonic, 424

Mode
 Nibble, 324
 Page, 324
 Modell
 funktionales, 156
 Modellprozessor, 360
 Modulo-Operation, 375
 Moore's law, 32
 Moore'sches Gesetz, 32, 424
 Moore-Automat, 288, 424
 MOS-Schaltung, 145
 MOS-Technik, 141, 424
 MOSFET, 424
 n-Kanal-, 145
 p-Kanal-, 145
 MSI, 25
 Multiple Instruction
 Multiple Data, 378
 Single Data, 378
 Multiplexer, **207**, 256, 337, 424
 Multiplexing, 28
 Multiplikation, 259, 367
 Multiplizierer, 234, 424
 Carry-save-, 238
 Dadda-Tree-, 246
 Matrix-, 235, 261
 Wallace-Tree-, 241
 Multiprozessorsystem, 352, 395, 398
 MWIPS, 403

N

n-Kanal, 49
 n-Kanal-JFET, 47
 n-Leiter, 40
 Nachkommanormalisierung, 76
 NaN, 79
 NAND-Funktion, 99
 Napierstäbchen, 14
 Negation, 93, 252
 Negationskreis, 157
 Negationstheorem, 106
 Negative Logik, 143, 424
 Negative-Bit, 364
 Negative-Flag, 358
 Net delay, 171
 Netzliste, 157

Neunerkomplement, 82, 85
 Neutrales Element, 91
 Nibble mode, 324
 NMOS-Schaltung, 148
 NMOS-Technik, 425
 NOR-Funktion, 99
 NOR-Gatter, 268
 Normalform, 102, 425
 disjunktive, 119, 212
 konjunktive, 119
 Reed-Muller-, 122
 Normalisierung, 425
 Nachkomma-, 76
 Vorkomma-, 76
 Normalisierungsregel, 76
 Not a number, 79
 NP-hart, 199
 Nullmenge, 120, 425

O

ODER-Matrix, 376
 ODER-Verknüpfung, 93
 OFDD, **130**, 159
 Offsetzähler, 308
 O-Kalkül, 167, 425
 Oktalsystem, 62, 425
 One-Hot-Code, 84, **85**, 368, 425
 Opcode, 19
 Opcode-Feld, 387
 Operationswerk, 338, 425
 Operationswerksynthese, 338
 Operator
 abgeleiteter, 97
 dualer, 108
 Operatorensystem
 vollständiges, 97, 117

P

p-Kanal-JFET, 47
 p-Leiter, 40
 p-Wanne, 53
 Packaging, 55, 425
 Page, 322
 Page miss, 397
 Page mode, 324
 Parallele Präfix-Funktion, 216
 Parallelmultiplizierer, 425
 Paritätsbit, 125, 328
 Paritätscode, 125
 Paritätsfunktion, **125**, 157, 215
 Partialprodukt, 234, 239
 Partielle Funktion, 425
 Patriot-Abwehrsystem, 66
 Peirce-Funktion, 99
 Performance rating, 401
 Pipeline, 18, **388**, 426
 -Hazard, 391
 flush, 393
 Superpipelining, **390**, 408
 Pipeline-Hazard, 425
 Pixel-Shader, 407
 PLA, 253
 Planartechnik, 50, 52, 426
 PLD, 253
 PMOS-Schaltung, 145
 PMOS-Technik, 426
 pn-Übergang, 41, 426
 Polaritätsindikator, 144
 Positive Logik, 143, 426
 Postinkrement, 381
 Postinkrement-Adressierung, 386
 PowerPC-Architektur, 386
 Prädikatenlogik, 165
 Prädiktion
 2-Bit-, 393
 Präfix-Addierer, 227, 426
 Präfix-Funktion
 parallele, 216
 Präfix-Logik, 215, 426
 Präfix-Schreibweise, 63
 Prämisse, 99
 Primblock, 190
 Primimplikant, 190
 Primimplikantentafel, 198
 Program counter, 358
 Programmable logic array, 253
 Programmable logic device, 253
 Programmierbare Logik, 253, 258, 426
 Programmspeicher, 361
 Programmsteuerung, 352
 Programmzähler, 316
 Progressiver Code, 83
 Protected mode, 29

Prüfbit, 329
 Pseudo-Tetrad, 81
 Puffer, 170
 Punktdiagramm, 243

Q

Quantor, 426
 boolescher, 165
 Quine-McCluskey-Verfahren, 197, 426
 Quine'sche Tabelle, 197

R

Radioaktive Strahlung, 328
 Radix-4-SRT-Division, 74
 RAM, 318, 426
 Random access memory, 318
 Range gate, 66
 Rank, 326
 Rationale Zahl, 63
 Raumladungszone, 41
 Read-only memory, 254
 Read-Signal, 319
 Rechenregel, 102
 abgeleitete, 109
 Rechenwerk, 356, 426
 Rechner, 352
 Rechnerklassifikation
 nach Flynn, 378
 nach Instruktionsarchitektur, 379
 Rechteckschwingung, 271
 Reduktionszelle
 Carry-save-, 230, 239
 Reed-Muller-Normalform, 122, 426
 Reflektierter Biquinär-Code, 84, 85
 Refresh-Logik, 327
 erweiterte, 329
 Regel
 von De Morgan, 107, 111
 Register, 300, 337, 427
 Akkumulator-, 305
 Auffang-, 300
 Ausgabe-, 337
 Daten-, 365
 Eingabe-, 337
 Hilfs-, 357

Instruktions-, 365
 Schiebe-, 302
 Stapel-, 358
 Status-, 358, 363
 Umlauf-, 303
 Universal-, **304**, 357
 Register-Transfer-Ebene, 336, 427
 Register-Transfer-Entwurf, 427
 Registerbreite, 300, 360
 Registerfenster, 406
 Registersatz, 356
 Rekonfigurierbarer Speicher, 384
 Rekonvergenz, 161, 172, 427
 Relais, 17
 Relative Adressierung, 381
 Relativer Sprung, 317, 363
 Reset, 281, 331, 350, 427
 Resistor, 43
 Resistor-Transistor-Logik, 142
 Ringzähler, 303
 RISC, 405, 427
 RISC-Prozessor, 384
 Robbins-Algebra, 91
 ROBDD, **125**, 158
 Röhre, 22
 ROM, 254, 427
 Römische Zahlen, 60, 427
 Row address strobe, 323
 RS-Flipflop, 277
 RS-Latch
 asynchrones, 267
 synchrones, 272
 RTL, 142
 Rückkopplungsschleife, 282
 Rücksprungadresse, 358
 Rückwärtszähler, 308
 Rundungsfehler, 67

S

Schalenmodell, 34
 Schaltalgebra, 90, 93, 427
 Schaltfunktion, 94
 Schaltkreis, 427
 integrierter, 26
 Schaltkreisfamilie, 140, 427
 Schaltnetz, 157, 427

Ausgabe-, 290
 Hazard-freies, 194
 Übergangs-, 289
 zweistufiges, 158
 Schaltung
 kombinatorische, 157
 sequenzielle, 266
 Schaltungssynthese, 156, 428
 BDD-basierte, 158
 FDD-basierte, 159
 zweistufige, 157
 Schaltwerk, 266, 428
 Schaltwerksynthese, 285, **289**
 Schickard'sche Rechenuhr, 14
 Schiebeoperation, 234
 Schieberegister, 302, 428
 rotierendes, 250
 Schrankensteuerung, 350
 Schrittlänge, 308
 Schrittzählung, 317
 Schwebezustand, 145
 Scrubbing, 329
 SDR-RAM, 326
 SDRAM, 325
 Seite, 322
 Sequencer, 365, 428
 Sequenzielle Schaltung, 266
 Sequenzielles Element, 266
 Shader, 407
 Shannon'scher Entwicklungssatz, 127, 428
 Sheffer-Funktion, 99
 Sheffer-Stroke, 117
 Signal
 Ausgangs-, 140
 Eingangs-, 140
 Signalausbreitung, 169
 Signalfanke, 170
 Signalpfad, 336
 Signalverzögerung, 169
 SIMD, 378, 407
 SIMM, 321
 Single cristal, 38
 Single Instruction
 Multiple Data, 378
 Single Data, 378
 Single-precision-Format, 77
 SISD, 378

Slave-Flipflop, 276
 SLSI, 25
 Small-scale integration, 25
 Soft error, 328
 Soroban, 13
 Source, 47
 SPARC-Prozessor, 406
 SPEC-Benchmark, 404
 SpeedStep-Technologie, 384
 Speicher, **318**, 428
 Cache-, 393
 Daten-, 361
 DDR-, 326
 dynamischer, 320
 Programm-, 361
 rekonfigurierbarer, 384
 SD-, 325
 SDR-, 326
 statischer, 318
 virtueller, 397
 Speicherbank, 324
 Speicherchip, 321
 Speicherelement, **266**, 428
 asynchrones, 267
 D-Flipflop, 276
 D-Latch, 272
 einflankengesteuertes, 275
 flankengesteuertes, 274
 JK-Flipflop, 279
 RS-Flipflop, 277
 RS-Latch
 asynchrones, 267
 synchrones, 272
 T-Flipflop, 278
 taktzustandsgesteuertes, 271
 zweiflankengesteuertes, 275
 Speicherhierarchie, 394, 428
 Speicherindirekte Adressierung, 382
 Speichermatrix, 322
 Speicherordnung, 68, 428
 Speicherphase, 359
 Speicherschleife, 282
 Speicherseite, 322
 Speicherzeit, 169
 Spekulative Befehlsausführung, 392
 Spin-Coating-Verfahren, 52
 Sprung
 absoluter, 317

- bedingter, 343, 363
- direkter, 363
- relativer, 317, 363
- unbedingter, 363
- Sprungbedingung, 347
- Sprungbefehl, 358, 367
- Sprungvorhersage, 393, 428
 - dynamische, 393
- SRAM, 318, 428
- SRT-Division, 74
- SSI, 25
- Stabiler Zustand, 269
- Stack, 358
- Stapel, 358
- Stapelregister, 428
- Startzustand, 286
- Statischer Hazard, 428
- Statischer Speicher, 318
- Statusbit, 429
- Statusregister, 358, 363, 429
- Statusvariable, 338
- Stellensystem, 61
- Stellenwertsystem, 61, 429
- Stelligkeit, 94
- Stellvariable, 338
- Steuerbus, 429
- Steuerflussabhängigkeit, 392, 408, 429
- Steuerlogik, 337
- Steuersignal, 354
- Steuervariable, 338, 347
- Steuerwerk, 338, 356, 429
 - synthese, 340
 - adressmodifizierendes, 346
 - fest verdrahtetes, 343, 349
 - mikroprogrammiertes, 343, 349
- Stibitz-Code, 81
- Strahlung
 - radioaktive, 328
- Strichsystem, 60
- Stromverbrauch, 174, 184
- Strukturbreite, 57, 429
- Strukturelle Induktion, 102, 429
- Stschoty, 13
- Störimpuls, 171, 194
- Suan pan, 13
- Subtrahierer, 233, 429
- Subtraktionsregel, 60
- Suffix-Notation, 63

- Summenbit, 219
- Super-large-scale integration, 25
- Supercomputer, 404
- Superpipelining, **390**, 408, 429
- Superskalartechnik, **391**, 408, 429
- Swizzle-Maske, 407
- Synchroner Zähler, 309, 331
- Synchrones RS-Latch, 272
- Synergistic Processing Unit, 30
- Synthese
 - Operationswerk-, 338
 - Schaltwerk-, 285
 - Steuerwerk-, 340
- System/360, 25

T

- T-Flipflop, 278
- T-Glied, 154
- Takt, 372, 429
 - differenzieller, 326
- Taktflankensteuerung, 274
- Taktfrequenz, 271, 399
- Taktgeber
 - Mehrphasen-, 376
- Taktsignal, 271
- Taktzustandssteuerung, 271
- Tautologie, 100, 429
- Taxonomie
 - nach Flynn, 378
 - nach Instruktionsarchitektur, 379
- Teile-und-herrsche, 224
- Terminal, 27
- Tetrade, 81
- Tetraden-Code, 80
- Timing closure, 171
- Top-500-Liste, 404
- Transduktor, 286, 430
- Transistor, 23, 42, 156, 320, 430
- Transistorebene, 156
- Transition, 269
- Transmissionsglied, 154
- Triode, 22
- Trägermenge, 91
- TTL-Technik, 140
- Turing-Maschine, 363

U

- Überdeckung
 - minimale, 198
- Übergangsschaltnetz, 289
- Übergangstabelle, 273, 430
- Überhitzungsschutz, 384
- Übertragsadditionsregel, 70
- Übertragsbit, 218
 - Rekursionsschema, 221
- Übertragslogik, 314
- ULSI, 25
- Ultra-large-scale integration, 25
- Umladestrom, 151, 174
- Umlaufregister, 303, 330
- Unbedingter Sprung, 363
- UND-Verknüpfung, 93
- Universalregister, **304**, 357, 430
- Universelles Berechnungsmodell, 352, 363
- Unterprogrammaufruf, 358, 406

V

- Vakuumröhre, 22
- Valenzband, 37
- Valenzelektron, 36
- Variable
 - abhängige, 94
 - freie, 94
 - Status-, 338
 - Stell-, 338
 - Steuer-, 338, 347
- Variablenbelegung
 - benachbarte, 186
 - inkonsistente, 164
 - konsistente, 164
- Vektorrechner, 378
- Venn-Diagramm, 92
- Verdrängungsstrategie, 397
- Verlustleistung, 151
- Vertex-Shader, 407
- Very-large-scale integration, 25
- Verzögerung
 - RAS/CAS, 323
- Verzögerungsglied, 271
- Verzögerungslogik, 392

Verzögerungszeit, 169
Virtueller Speicher, 397, 430
VLIW-Prozessor, 391
VLSI, 25
Volladdierer, 218, 430
Vollassoziativer Cache, 409
Vollständige Induktion, 102, 430
Vollständiges Operatorensystem, 97, 117, 430
Von-Neumann-Architektur, 23, 352, 430
Von-Neumann-Flaschenhals, 356, 430
Von-Neumann-Rechner, 430
Vorkommanormalisierung, 76, 78
Vorwärtszähler, 308
Vorzeichenbitdarstellung, 69, 234, 431

W

Wafer, 51, 53, 431
Wahrheitstabelle, 94, 431
Wahrheitstafel, 94
Walking-Code, 85
Wallace-Tree-Multiplizierer, 241, 431
Wärmeentwicklung, 151
Wartezyklus, 392, 408, 431
Whetstone-Benchmark, 403
Widerstand
 spezifischer, 37
Wire-Bond-Verfahren, 56
Wiring layer, 54
Word line, 320
Write, 388
Write back, 359
Write enable, 371

Write-In-Strategie, 398
Write-Through-Strategie, 397

X

x86-Architektur, 379, 431
XLSI, 25
XOR-Verknüpfung, 99

Z

Z3, 17
Zahlencode, 80
Zahlendarstellung
 explizite, 77
 gepackte, 77
 implizite, 77
 ungepackte, 77
Zahlenformat, 431
 äquidistantes, 74
 Auflösungsgenauigkeit, 75
 Festkommadarstellung, 73
 Gleitkommadarstellung, 74
 rechnerinternes, 67
Zahlensystem, 60, 431
 b-adisches, 62
 binäres, 62
 eineindeutiges, 69
 hexadezimal, 63
 oktales, 62
 redundantes, 69
 symmetrisches, 69
 unäres, 60
Zähler, 308, 431

asynchroner, 313, 330
bidirektionaler, 308
Binär-, 308
gemischter, 314
Instruktions-, 316, **358**, 365
Offset-, 308
synchroner, 309, 331
Zeitdiagramm, 268, 431
Zeitverhalten, 169
Zellbibliotheken, 185
Zelle, 156
Zentraleinheit, 352
Zero-Bit, 358, 364
Ziffer, 60
Zugriffszeit
 CAS-, 323
 RAS-, 323
Zustand
 -smenge, 286
 -ssteuerung
 negative, 274
 positive, 274
 -svariable, 267
 -svektor, 285
 -sübergang, 269
 -sübergangsfunktion, 286
 -sübergangsgraph, 269
 instabiler, 269, 295
 stabiler, 269, 295
Zustandsvariable, 431
Zweierkomplement, **71**, 233, 252, 431
Zweistufiges Schaltnetz, 431
Zwischenglied, 140
Zykluszeit, 323
Zählrichtung, 308