Mikrocomputertechnik mit dem Prozessor 8085 A

Maschinenorientierte Programmierung
Grundlagen - Schaltungstechnik -
Anwendungen

von Prof. Dipl.-Ing. Günter Schmitt

6., verbesserte und erweiterte Auflage

Mit 400 Bildern und 17 Tabellen

R. Oldenbourg Verlag München Wien 1994

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Schmitt, Günter:
Mikrocomputertechnik mit dem Prozessor 8085 A :
maschinenorientierte Programmierung ; Grundlagen –
Schaltungstechnik – Anwendungen ; mit 17 Tabellen / von
Günter Schmitt. – 6., verb. und erw. Aufl. – München ; Wien :
Oldenbourg, 1994
ISBN 3-486-22802-1

Gesamtherstellung: R. Oldenbourg Graphische Betriebe GmbH, München

ISBN 3-486-22802-1

Inhaltsverzeichnis

Vorwort

Die rasche Entwicklung der Mikrocomputertechnik hat mich veranlaßt, meine beiden Bücher "Maschinenorientierte Programmierung für Mikroprozessoren " und "Grundlagen der Mikrocomputertechnik" zu überarbeiten. Hardware, Software und Anwendungen eines Prozessors werden jetzt in einem Band zusammengefaßt. Der erste Band für den Prozessor 8085A liegt hiermit vor, weitere Bände für die Prozessoren 6800/6802, 6809 und 68000 werden folgen.

Nach den ersten Lehrjahren gilt es nun, die Mikrocomputertechnik fest in die Ausbildung des technischen Nachwuchses einzubauen. Dieses Buch entstand aus und für meinen Unterricht im Pflichtfach "Mikrocomputertechnik" und in den weiterführenden Wahlpflichtfächern an der Fachhochschule Dieburg. In der Gliederung des Stoffes und in der Auswahl der Beispiele habe ich versucht, einen "Lehrbuchstil" zu finden, wie er sich z.B. auf dem Gebiet der Grundlagen der Elektrotechnik seit langem herausgebildet hat. Der Programmierteil beschäftigt sich ausschließlich mit der maschinenorientierten Programmierung auf Assemblerebene, wie sie für die Programmierung von technischen Anwendungen und in der Systemprogrammierung vorzugsweise verwendet wird.

Ich bedanke mich bei meinen Studenten für die vielen Fragen und Fehler, die viel zu einer besonders eingehenden Darstellung wichtiger und schwieriger Fragen beigetragen haben. Dem Oldenbourg Verlag danke ich für die gute Zusammenarbeit und bei meiner Familie entschuldige ich mich, daß ich "geistig abwesend" war, als dieses Buch entstand.

Groß-Umstadt, im Februar 1984, dem Jahr des Großen Bruders

Günter Schmitt

Vorwort zur 6. Auflage

Der Unterricht in der Mikrocomputertechnik wird durch das Vordringen des Personal Computers (PC) zunehmend schwieriger. Die in ihm verwendeten 32-bit- und 64-bit-Mikroprozessoren sind in der Grundausbildung wegen ihrer Komplexität "unlehrbar"; die hochintegrierten Steuer- und Multifunktionsbausteine machen die Schaltung völlig "undurchsichtig". Daher lassen sich die Grundprinzipien der Maschinenorientierten Programmierung und Schaltungstechnik mit einem PC als Übungsgerät nicht mehr lehren und für den Lernenden nicht mehr nachvollziehen. Wer würde es ohne einen ängstlichen Blick auf den Staatsanwalt wagen, im Unterricht an einer PC-Platine bei geöffnetem Gehäuse Messungen durchführen zu lassen?

Auf der anderen Seite wird der PC immer mehr als Rechner für Aufgaben in der Meß-, Steuerungs- und Regelungstechnik eingesetzt. Die Programmierung verwendet vorzugsweise die modernen Hochsprachen Pascal oder C; nur zeitkritische Teilaufgaben schreibt man noch im Assembler. Für diese Anwendungen muß die Mikrocomputertechnik zusätzlich Kenntnisse über die Schnittstellen des PC sowie über die DMA- und Interruptsteuerung liefern; jedoch sollten diese Bausteine gefahrlos für Messungen zugänglich sein.

In der vorliegenden 6. Auflage wurde diese neue Aufgabenstellung berücksichtigt. Der Abschnitt 3.9 beschreibt nun ein einfaches 8085-System mit den im PC üblicherweise verwendeten Peripheriebausteinen und einfachen Testprogrammen, die lediglich die richtige Adressierung der Bausteine zeigen können. Für weitergehende Untersuchungen sollte die im Kapitel 9 genannte ergänzende und weiterführende Literatur herangezogen werden. Das System wurde in lötfreier Stecktechnik aufgebaut und mit dem in Abschnitt 6.4 beschriebenen Monitor in Verbindung mit einem PC als Bedienungsterminal betrieben. Der Anhang enthält das in Pascal geschriebene Terminalprogramm. Die Testprogramme wurden mit einem ebenfalls in Pascal geschriebenen Cross-Assembler übersetzt und vom PC in das Testssystem heruntergeladen.

Groß-Umstadt, im August 1993

Günter Schmitt

1 Einführung

Dieser Abschnitt gibt Ihnen einen zusammenfassenden Überblick über die Anwendung, den Aufbau und die Programmierung von Mikrorechnern, neudeutsch auch Mikrocomputer genannt. In den Fällen, in denen die deutsche Fachsprache noch keine eigenen Ausdrücke gebildet hat, mußten die amerikanischen Bezeichnungen übernommen werden. Dabei wurde versucht, zusätzlich einen entsprechenden deutschen Ausdruck zu finden.

1.1 Anwendung von Mikrorechnern

Der Mikrorechner hat zwei Ahnen: die hochintegrierte Logikschaltung des Taschenrechners und die Großrechenanlage, Computer genannt. Auf einer Fläche von etwa 20 bis 50 Quadratmillimetern lassen sich heute mehr als 100 000 Schaltfunktionen unterbringen. Und dies in großen Stückzahlen zu niedrigen Preisen. Ähnlich wie bei einem Großrechner sind auch die Funktionen des Mikrorechners programmierbar. Was die Schaltung, die Hardware, tun soll, bestimmt ein Programm, die Software. Dadurch erst lassen sich die Bausteine universell einsetzen.

Heute unterscheidet man hauptsächlich zwei große Einsatzgebiete:

Mikrorechner in der technischen Anwendung steuern z.B. Drucker, elektronische Schreibmaschinen, Kopierautomaten, Telefonvermittlungen und Fertigungsanlagen. Durch den Einsatz von Mikrorechnern werden die Geräte kleiner und billiger und können mehr und "intelligentere" Funktionen übernehmen.

Mikrorechner werden in zunehmendem Maße als Klein-EDV-Anlagen eingesetzt und übernehmen damit Aufgaben ihres großen Bruders, des Großrechners. Die elektronische Datenverarbeitung, kurz EDV genannt, hält dadurch ihren Einzug als Personal-Computer oder Hobby-Computer in jeden Haushalt. Ob dies sinnvoll ist, darüber kann man geteilter Meinung sein; die Anwendung von Mikrorechnern zur Textverarbeitung oder Buchführung in Büros und kleineren Betrieben hat sich heute durchgesetzt. Der Text dieses Buches wurde mit Hilfe eines Mikrorechners am Bildschirm entworfen und korrigiert.

1.2 Aufbau und Bauformen von Mikrorechnern

Was ist allen Mikrorechnern in den verschiedenen Einsatzgebieten gemeinsam? Von der Funktion her gesehen sind es zunächst programmierbare Rechner. In einem Programmspeicher befindet sich eine Arbeitsvorschrift, das Programm. Bei einem Typenraddrucker z.B. gibt das Programm dem Hammer genau dann einen Ausgabebefehl, wenn der richtige Buchstabe des Rades am Papier vorbeikommt. Bei einem Abrechnungsprogramm z.B. enthält das Programm Rechenbefehle, die aus der Menge und dem Einzelpreis den Gesamtpreis berechnen. Der Datenspeicher enthält die zu verarbeitenden Daten. Im Beispiel eines Druckers sind es die auszugebenden Buchstaben und Ziffern, im Beispiel des Abrechnungsprogramms sind es Artikelbezeichnungen und Zahlen. In der Speichertechnik unterscheidet man Festwertspeicher und Schreib/Lesespeicher. Festwertspeicher behalten ihren Speicherinhalt unabhängig von der Versorgungsspannung. Sie können im Betrieb nur gelesen werden. Die Steuerprogramme für Geräte und kleine Anlagen werden hauptsächlich in Festwertspeichern untergebracht. Schreib/Lesespeicher verlieren ihren Speicherinhalt beim Abschalten der Versorgungsspannung; sie können aber während des Betriebes sowohl gelesen als auch neu beschrieben werden. Sie werden vorzugsweise für die Speicherung der Daten verwendet. Klein-EDV-Anlagen werden in den meisten Fällen mit magnetischen Speichern (Disketten- oder Floppy-Laufwerken) ausgerüstet, von denen man Anwendungsprogramme (z.B. Buchführungsprogramme) und Daten (z.B. Adressen der Kunden) in den Schreib/Lesespeicher lädt.

Der Mikroprozessor ist die Zentraleinheit, die das Programm ausführt und die Daten verarbeitet. Die Befehle werden in einer bestimmten Reihenfolge aus dem Programmspeicher in das Steuerwerk des Prozessors geholt. Das Rechenwerk verarbeitet die Daten, indem es z.B. den auszugebenden Buchstaben mit dem augenblicklichen Stand des Typenrades vergleicht oder bei einem Abrechnungsprogramm Zahlen addiert und subtrahiert.

Ein/Ausgabeschaltungen, auch Schnittstellen genannt, verbinden den Mikrorechner mit seiner Umwelt, der Peripherie. Im Beispiel des Typenraddruckers muß der Rechner, natürlich im richtigen Zeitpunkt, dem Magneten des Hammers einen Impuls geben. Bei einem Abrechnungsprogramm müssen z.B. von der Bedienungstastatur Zahlen eingelesen werden. Ein/Ausgabeschaltungen dienen hauptsächlich zur Übertragung von Daten.

Bild 1-1 zeigt zusammenfassend die wichtigsten Funktionseinheiten eines Mikrorechners.

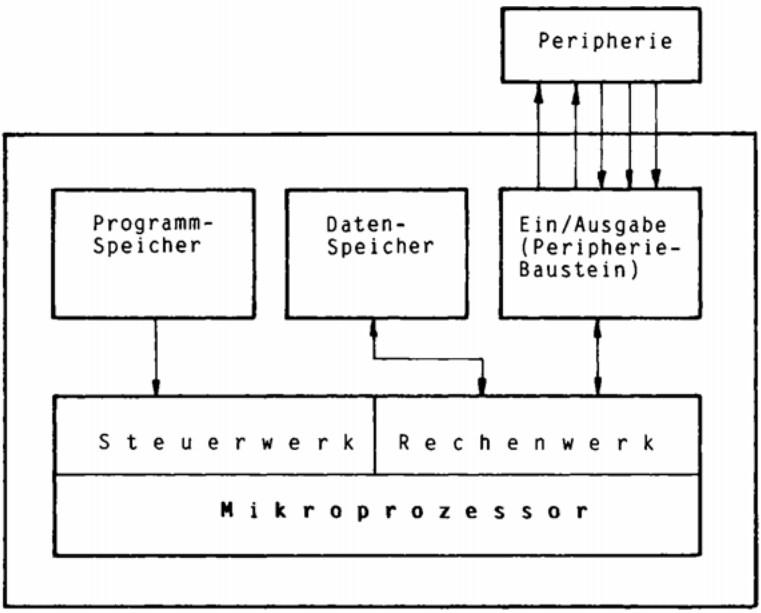


Bild 1-1: Aufbau eines Mikrorechners

Mikrorechner bestehen im wesentlichen aus dem Mikroprozessor, Programm- und Datenspeichern sowie Ein/Ausgabeschaltungen für die Verbindung zur Peripherie. Man unterscheidet folgende Bauformen:

Single-Chip-Mikrocomputer (Ein-Baustein-Mikrorechner) enthalten alle Funktionseinheiten (Prozessor, Speicher und Ein/Ausgabeschaltungen) auf einem Baustein der Größe 15 mal 50 mm. Das Programm besteht aus etwa 1000 Befehlen und befindet sich in einem Festwertspeicher auf dem Baustein. Der Schreib/Lesespeicher kann etwa 100 Daten (Zeichen oder Zahlen) aufnehmen. An den Anschlußbeinchen (ca. 40) stehen nur die Ein/Ausgabeleitungen für die Peripherie zur Verfügung. Ein derartiger Baustein kostet zwischen 10 und 100 DM. Der Ein-Baustein-Mikrorechner wird vorzugsweise für die Steuerung von kleineren Geräten (z.B. einfachen Druckern oder Meßgeräten) bei großen Stückzahlen eingesetzt, bei denen es auf geringe Abmessungen ankommt. Man kann ihn mehr als intelligenten Steuerbaustein denn als Rechner betrachten.

Single-Board-Mikrocomputer (Ein-Platinen-Mikrorechner) enthalten alle Funktionseinheiten eines Mikrorechners aufgebaut aus mehreren Bausteinen auf einer Leiterplatte. Im einfachsten Fall enthält eine Platine im Europaformat (100 x 160 mm) also einen Mikroprozessor z.B. vom Typ 8085, einen Festwert-Speicherbaustein mit dem Programm, einen Schreib/Lese-Speicherbaustein für die veränderlichen Daten und einige Ein/Ausgabebausteine für den Peripherie-

anschluß. Die Verbindungsleitungen, auf denen die Befehle und Daten zwischen den Bausteinen übertragen werden, bezeichnet man als Bus. Die Bausteine kosten zusammen ca. 50 DM, die Leiterplatte zwischen 50 und 200 DM. Dazu kommen die Kosten für das Programm und für die Peripherie. Das Haupteinsatzgebiet der Ein-Platinen-Mikrorechner liegt in der Steuerung von größeren Geräten wie z.B. elektronischen Schreibmaschinen oder Hobby-Computern. Die Entwicklung des Gerätes umfaßt den Entwurf des Rechners (Hardware), des Programms (Software) und die Anpassung an die Peripherie.

Bauplatten-Mikrocomputer bestehen aus mehreren Karten, meist im Europaformat, die in einem Rahmen zusammengesteckt werden. Hier teilt man den Mikrorechner auf in eine Prozessorkarte, Speicherkarten für Festwertspeicher, Speicherkarten für Schreib/Lesespeicher und Peripheriekarten für die Datenübertragung. Sein Haupteinsatzgebiet sind die Klein-EDV-Anlagen (Personal-Computer, Büro-Computer), die sich durch Einfügen neuer Karten leicht erweitern oder durch den Austausch defekter Karten schnell reparieren lassen. Für die Anwendung im technischen Bereich zur Steuerung von größeren Geräten und Anlagen werden von verschiedenen Herstellern Bauplattensysteme angeboten. Bei einem aus Bauplatten zusammengestellten Mikrorechner entfällt der größte Teil der Hardwareentwicklung, und die Entwicklung des Programms, der Software, kann sofort beginnen. Bei steigenden Stückzahlen kann es wirtschaftlich sein, bei unverändertem Programm aus einem Bauplatten-Mikrorechner einen maßgeschneiderten Ein-Karten-Mikrorechner zu entwickeln.

1.3 Die Programmierung von Mikrorechnern

Die sogenannten Maschinenbefehle im Programmspeicher des Mikrorechners sind binär verschlüsselt, d.h. sie bestehen nur aus Nullen und Einsen. In Programmlisten bedient man sich der kürzeren hexadezimalen Schreibweise, die jeweils vier Binärzeichen durch ein neues Zeichen ersetzt. Da diese Art der Programmdarstellung sehr unanschaulich ist, benutzt man beim Programmieren Sprachen, als wolle man mit dem Rechner "reden". Dabei unterscheidet man maschinennahe Sprachen, den Assembler, und aufgabennahe "höhere" Sprachen wie z.B. BASIC.

Bei der Programmierung von Problemen der Datenübertragung sind sehr genaue Kenntnisse über den Aufbau des Mikroprozessors und der Ein/Ausgabebausteine erforderlich. Hier bevorzugt man die maschinennahe Assemblersprache, die aus leicht merkbaren Abkürzungen besteht. Ein Assemblerbefehl entspricht einem Maschinenbefehl. Da jeder Mikroprozessor einen eigenen Befehls- und Registersatz hat, gibt es für jeden Prozessortyp eine eigene Assemblersprache, so daß sich im Assembler geschriebene Programme nicht zwischen verschiedenen Prozessortypen austauschen lassen. Die Programmierung im Assembler ist sehr zeitaufwendig, ergibt aber schnelle und kurze Programme.

anschluß. Die Verbindungsleitungen, auf denen die Befehle und Daten zwischen den Bausteinen übertragen werden, bezeichnet man als Bus. Die Bausteine kosten zusammen ca. 50 DM, die Leiterplatte zwischen 50 und 200 DM. Dazu kommen die Kosten für das Programm und für die Peripherie. Das Haupteinsatzgebiet der Ein-Platinen-Mikrorechner liegt in der Steuerung von größeren Geräten wie z.B. elektronischen Schreibmaschinen oder Hobby-Computern. Die Entwicklung des Gerätes umfaßt den Entwurf des Rechners (Hardware), des Programms (Software) und die Anpassung an die Peripherie.

Bauplatten-Mikrocomputer bestehen aus mehreren Karten, meist im Europaformat, die in einem Rahmen zusammengesteckt werden. Hier teilt man den Mikrorechner auf in eine Prozessorkarte, Speicherkarten für Festwertspeicher, Speicherkarten für Schreib/Lesespeicher und Peripheriekarten für die Datenübertragung. Sein Haupteinsatzgebiet sind die Klein-EDV-Anlagen (Personal-Computer, Büro-Computer), die sich durch Einfügen neuer Karten leicht erweitern oder durch den Austausch defekter Karten schnell reparieren lassen. Für die Anwendung im technischen Bereich zur Steuerung von größeren Geräten und Anlagen werden von verschiedenen Herstellern Bauplattensysteme angeboten. Bei einem aus Bauplatten zusammengestellten Mikrorechner entfällt der größte Teil der Hardwareentwicklung, und die Entwicklung des Programms, der Software, kann sofort beginnen. Bei steigenden Stückzahlen kann es wirtschaftlich sein, bei unverändertem Programm aus einem Bauplatten-Mikrorechner einen maßgeschneiderten Ein-Karten-Mikrorechner zu entwickeln.

1.3 Die Programmierung von Mikrorechnern

Die sogenannten Maschinenbefehle im Programmspeicher des Mikrorechners sind binär verschlüsselt, d.h. sie bestehen nur aus Nullen und Einsen. In Programmlisten bedient man sich der kürzeren hexadezimalen Schreibweise, die jeweils vier Binärzeichen durch ein neues Zeichen ersetzt. Da diese Art der Programmdarstellung sehr unanschaulich ist, benutzt man beim Programmieren Sprachen, als wolle man mit dem Rechner "reden". Dabei unterscheidet man maschinennahe Sprachen, den Assembler, und aufgabennahe "höhere" Sprachen wie z.B. BASIC.

Bei der Programmierung von Problemen der Datenübertragung sind sehr genaue Kenntnisse über den Aufbau des Mikroprozessors und der Ein/Ausgabebausteine erforderlich. Hier bevorzugt man die maschinennahe Assemblersprache, die aus leicht merkbaren Abkürzungen besteht. Ein Assemblerbefehl entspricht einem Maschinenbefehl. Da jeder Mikroprozessor einen eigenen Befehls- und Registersatz hat, gibt es für jeden Prozessortyp eine eigene Assemblersprache, so daß sich im Assembler geschriebene Programme nicht zwischen verschiedenen Prozessortypen austauschen lassen. Die Programmierung im Assembler ist sehr zeitaufwendig, ergibt aber schnelle und kurze Programme.

Bei der Programmierung von EDV-Problemen wie z.B. einer Adressenverwaltung bevorzugt man "höhere" Programmiersprachen, die z.T. der Formel- und Algorithmenschreibweise der Mathematik entsprechen. Ein Algorithmus ist die mathematische Beschreibung eines Lösungsverfahrens. Ein Übersetzungsprogramm (Interpretierer oder Compiler) wandelt einen Befehl in mehrere Maschinenbefehle um. Die Programmierung in einer höheren problemorientierten Sprache erfordert keine Kenntnisse über den Aufbau und die Funktion des Mikrorechners und seiner Bausteine, die Programme sind zwischen verschiedenen Rechnern austauschbar. Sie sind jedoch länger und langsamer als entsprechende Assemblerprogramme.

Bild 1-2 zeigt einige Beispiele für Befehle in verschiedenen Darstellungen. Sie sind jedoch nicht miteinander vergleichbar, da z.B. der BASIC-Befehl zum Wurzelziehen im Assembler nur mit sehr hohem Aufwand programmiert werden kann.

Befehl binär	00111010000100010100011 1
Befehl hexadezimal	3A 11 47
Assembler-Befehl	LDA WERT
BASIC-Befehl	LET WERT = 13

Bild 1-2: Beispiele für verschiedene Befehlsarten

Mikrorechner in der technischen Anwendung werden vorzugsweise im Assembler oder in besonders auf technische Probleme zugeschnittenen Sprachen programmiert. In der Anwendung als Klein-EDV-Anlage bevorzugt man problemnahe Sprachen wie z.B. BASIC und greift bei der Datenübertragung auf Systemprogramme zurück, die mit der Anlage vom Hersteller geliefert werden und im Assembler geschrieben sind. Die Systemprogramme, die zum Betrieb eines Rechners erforderlich sind, bezeichnet man auch als Betriebssystem.

Dieses Buch beschäftigt sich ausschließlich mit der maschinenorientierten Programmierung im Assembler des Prozessors 8085. **Bild 1-3** faßt die wichtigsten Erkenntnisse dieser Einführung zusammen.

Mikrorechner-Hardware	Mikrorechner-Anwendungen	Mikrorechner-Software
Ein-Baustein- oder Kleinstsystem	Kleingeräte Meßgeräte Drucker	Assembler
Ein-Karten-System	Größere Geräte Schreibmaschine Tischcomputer	Assembler BASIC PASCAL C
Bauplatten-System	Prozeßrechner Klein-EDV-Anlagen	Assembler PASCAL C

Bild 1-3: Anwendung, Bauformen und Programmierung von Mikrorechnern

Der im vorliegenden Buch behandelte Prozessor 8085A ist ein Universalprozessor, der früher in allen Anwendungsbereichen (Bild 1-3) eingesetzt wurde. Er wird heute zunehmend durch andere Bauformen abgelöst. Für die Steuerung von Kleingeräten (Meß- und Anzeigegeräten, Tastaturen, Druckern) verwendet man fast ausschließlich Single-Chip-Systeme. Die Personal Computer (PC) enthalten heute vorwiegend 16- und 32-Bit-Prozessoren z.T. mit Coprozessoren für arithmetische Befehle und mathematische Funktionen. In nachrichtentechnischen Anwendungen findet man vielfach Prozessoren, die auf bestimmte Anwendungen zugeschnitten sind. Ein Beispiel sind die "Digitalen Signalprozessoren" (DSP) zum Aufbau von digitalen Filtern und Regelungen im Echtzeitbetrieb.

2 Grundlagen

Dieser Abschnitt ist für Leser ohne Vorkenntnisse gedacht, die ohne begleitenden Unterricht arbeiten. Wer bereits mit den Grundlagen der Datenverarbeitung und Digitaltechnik vertraut ist, kann diesen Abschnitt überschlagen.

2.1 Darstellung der Daten im Mikrorechner

Daten sind Zahlen (z.B. Meßwerte), Zeichen (z.B. Buchstaben) oder analoge Signale (z.B. Spannungen). Sie werden im Rechner binär gespeichert und verarbeitet. Binär heißt zweiwertig, es sind also nur zwei Zustände entsprechend **Bild 2-1** erlaubt:

wahr	-	falsch
Schalter ein	-	Schalter aus
hohes Potential	-	niedriges Potential
HIGH-Potential	-	LOW-Potential

Bild 2-1: Binäre Zustände

Die Datenverarbeitung bezeichnet die beiden Zustände mit den Ziffern Null und Eins. In der Digitaltechnik wird ein niedriges Potential zwischen 0 und 0,8 Volt als **LOW** bezeichnet; ein hohes Potential zwischen 2 und 5 Volt heißt **HIGH** .

Eine Speicherstelle, die eines der beiden Binärzeichen enthält, nennt man ein Bit. Acht Bits, also acht Speicherstellen, bilden ein Byte. Weitere Einheiten entsprechend **Bild 2-2** sind das Kilobyte für 1024 Bytes und das Megabyte für 1024 Kilobytes.

Bit	=	Speicherstelle mit 0 oder 1
Byte	=	Speicherwort aus acht Bits
Kilobyte	=	1024 Bytes
Megabyte	=	1024 Kilobyte = 1 048 576 Bytes

Bild 2-2: Speichereinheiten

Für die Darstellung von Zahlen gibt es zwei Möglichkeiten: die BCD-Codierung und die duale Zahlendarstellung.

BCD bedeutet Binär Codierte Dezimalziffer. Jede Dezimalziffer wird entsprechend **Bild 2-3** binär verschlüsselt (codiert). Dabei bleibt jedoch die Zahl im dezimalen Zahlensystem erhalten.

Ziffer	Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Beispiel:

Dezimalzahl 123 = 000100100011

Ziffer 1 — 0001

Ziffer 2 — 0010

Ziffer 3 — 0011

Bild 2-3: BCD-Codierung

Das duale Zahlensystem kennt nur die beiden Ziffern 0 und 1. Die Dezimalzahl 123 lautet in dieser Darstellung 01111011. Da wir unsere Daten dezimal eingeben und auch die Ergebnisse dezimal erwarten, ist bei der dualen Speicherung von Zahlen im Rechner eine Umwandlung der Zahlen erforderlich.

Für die binäre Darstellung von Zeichen verwendet man Codes, mit denen man alle Buchstaben, Ziffern und Sonderzeichen der Schreibmaschinentastatur darstellen kann. **Bild 2-4** zeigt einen Ausschnitt aus dem in der Mikrorechnertechnik vorwiegend verwendeten ASCII-Code. Der Anhang enthält die vollständige Tabelle.

Buchstaben	Ziffern	Sonderzeichen
A = 01000001	0 = 00110000	! = 00100001
B = 01000010	1 = 00110001	" = 00100010
C = 01000011	2 = 00110010	# = 00100011
D = 01000100	3 = 00110011	$ = 00100100
E = 01000101	4 = 00110100	% = 00100101
F = 01000110	5 = 00110101	& = 00100110
.	.	.
.	.	.

Bild 2-4: ASCII-Codierung

ASCII ist eine Abkürzung aus dem Amerikanischen und bedeutet frei übersetzt: Amerikanischer Normcode für den Austausch von Nachrichten. Er wurde ursprünglich für den Fernschreibverkehr verwendet. Zu den sieben Bits des eigentlichen Zeichens fügt man oft ein achtes Bit als Kontrollbit hinzu. Damit kann ein Byte genau ein Zeichen speichern. Der Code ist regelmäßig aufgebaut. Die letzten vier Bits der Zifferncodierung z.B. entsprechen dem BCD-Code.

Analoge Signale (Spannungen und Ströme) werden durch Wandlerbausteine in binäre Werte umgesetzt. Sie finden besondere Anwendung in der Meßtechnik.

2.2 Zahlensysteme und Umrechnungsverfahren

Dezimal	Dual	Aufbau der Dualzahl
0	0000	0 + 0 + 0 + 0 = 0
1	0001	0 + 0 + 0 + 1 = 1
2	0010	0 + 0 + 2 + 0 = 2
3	0011	0 + 0 + 2 + 1 = 3
4	0100	0 + 4 + 0 + 0 = 4
5	0101	0 + 4 + 0 + 1 = 5
6	0110	0 + 4 + 2 + 0 = 6
7	0111	0 + 4 + 2 + 1 = 7
8	1000	8 + 0 + 0 + 0 = 8
9	1001	8 + 0 + 0 + 1 = 9

Bild 2-5: Die ersten neun Dualzahlen

Buchstaben	Ziffern	Sonderzeichen
A = 01000001	0 = 00110000	! = 00100001
B = 01000010	1 = 00110001	" = 00100010
C = 01000011	2 = 00110010	# = 00100011
D = 01000100	3 = 00110011	$ = 00100100
E = 01000101	4 = 00110100	% = 00100101
F = 01000110	5 = 00110101	& = 00100110
.	.	.
.	.	.

Bild 2-4: ASCII-Codierung

ASCII ist eine Abkürzung aus dem Amerikanischen und bedeutet frei übersetzt: Amerikanischer Normcode für den Austausch von Nachrichten. Er wurde ursprünglich für den Fernschreibverkehr verwendet. Zu den sieben Bits des eigentlichen Zeichens fügt man oft ein achtes Bit als Kontrollbit hinzu. Damit kann ein Byte genau ein Zeichen speichern. Der Code ist regelmäßig aufgebaut. Die letzten vier Bits der Zifferncodierung z.B. entsprechen dem BCD-Code.

Analoge Signale (Spannungen und Ströme) werden durch Wandlerbausteine in binäre Werte umgesetzt. Sie finden besondere Anwendung in der Meßtechnik.

2.2 Zahlensysteme und Umrechnungsverfahren

Dezimal	Dual	Aufbau der Dualzahl
0	0000	0 + 0 + 0 + 0 = 0
1	0001	0 + 0 + 0 + 1 = 1
2	0010	0 + 0 + 2 + 0 = 2
3	0011	0 + 0 + 2 + 1 = 3
4	0100	0 + 4 + 0 + 0 = 4
5	0101	0 + 4 + 0 + 1 = 5
6	0110	0 + 4 + 2 + 0 = 6
7	0111	0 + 4 + 2 + 1 = 7
8	1000	8 + 0 + 0 + 0 = 8
9	1001	8 + 0 + 0 + 1 = 9

Bild 2-5: Die ersten neun Dualzahlen

Der einfachste Weg zu den Dualzahlen führt über das Zählen. Die Zahlen 0 und 1 sind im dezimalen und im dualen Zahlensystem gleich. Da damit im Dualsystem der Wertevorrat erschöpft ist, rückt man eine Stelle nach links, und es ergibt sich die Dualzahl 10 für die dezimale 2. **Bild 2-5** zeigt die ersten neun Dualzahlen nach der Zählmethode. Sie entsprechen den BCD-Codierungen.

Im Dezimalsystem läßt man führende Nullen fort. Da die Rechentechnik mit einer festen Stellenzahl arbeitet, müssen führende Nullen entsprechend der Zahl der Stellen mitgeführt werden, denn den binären Wert "leer" gibt es nicht. Für die Umwandlung größerer Zahlen gibt es Verfahren, die sich aus dem Aufbau der Zahlensysteme herleiten lassen.

Das dezimale Zahlensystem verwendet die zehn Ziffern von 0 bis 9. Der Wert einer Ziffer hängt von der Stelle ab, an der sie steht. **Bild 2-6** zeigt als Beispiel den Aufbau der Dezimalzahl 123.

dezimal 123 = $1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

= 100 + 20 + 3

= 123

Bild 2-6: Aufbau der Dezimalzahl 123

Die Wertigkeiten der Stellen sind Potenzen zur Basis 10. Multipliziert man jede Ziffer mit ihrer Wertigkeit und addiert man die Produkte, so ergibt sich wieder die ursprüngliche Zahl.

Das duale Zahlensystem verwendet die beiden Ziffern 0 und 1. Daher sind die Wertigkeiten der Dualstellen Potenzen zur Basis 2. **Bild 2-7** zeigt als Beispiel die Umrechung der Dualzahl 01111011 in eine Dezimalzahl und die Umwandlungsregel.

dual 01111011 = $0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

= 0 + 64 + 32 + 16 + 8 + 0 + 2 + 1

= 123 **dezimal**

Umwandlungsregel

Die Dualstellen sind mit ihrer Wertigkeit zu multiplizieren. Die Produkte ergeben addiert die Dezimalzahl.

Bild 2-7: Umwandlung einer Dualzahl in eine Dezimalzahl

Will man umgekehrt eine Dezimalzahl in eine Dualzahl umrechnen, so ist sie in Zweierpotenzen zu zerlegen. **Bild 2-8** zeigt als Beispiel die Umwandlung der Dezimalzahl 123 in eine achtstellige Dualzahl und die Umwandlungsregel (Divisionsrestverfahren).

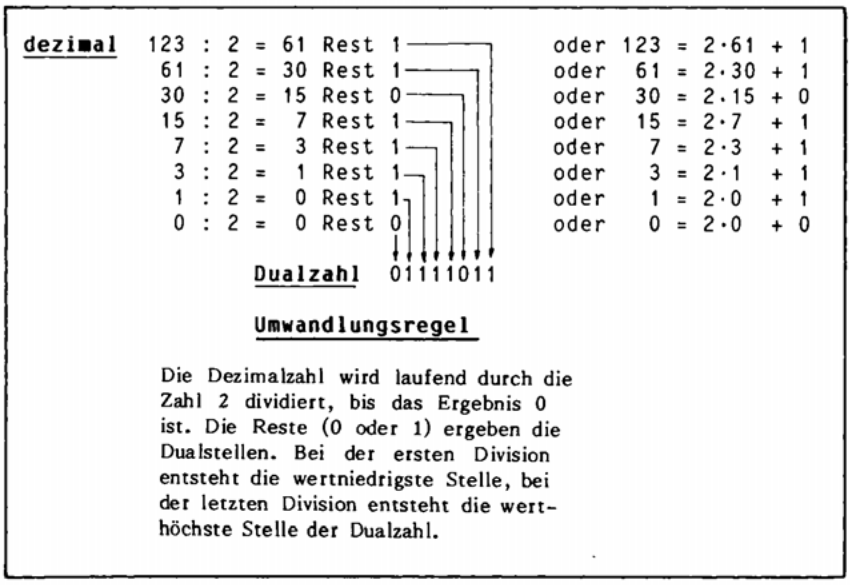


Bild 2-8: Umwandlung einer Dezimalzahl in eine Dualzahl

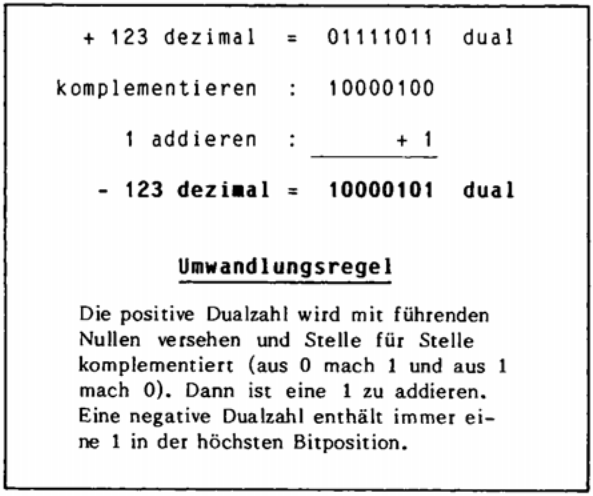


Bild 2-9: Bildung negativer Dualzahlen

Es lassen sich auch negative Dualzahlen bilden. Von den verschiedenen Möglichkeiten soll nur die bei Mikrorechnern gebräuchliche Darstellung im Zweier-Komplement erwähnt werden. Zur Bildung einer negativen Dualzahl wird der positive Wert Stelle für Stelle komplementiert (ergänzt), und es wird zusätzlich eine 1 addiert. **Bild 2-9** zeigt ein Beispiel und die Umwandlungsregel.

Aus einer negativen Dualzahl wird durch Rückkomplementieren nach dem gleichen Verfahren wieder eine positive Dualzahl, denn eine doppelte Verneinung hebt sich auf. Man beachte, daß bei vorzeichenbehafteten Dualzahlen die ganz links stehende Stelle nicht mehr Bestandteil der Zahl ist, sondern das Vorzeichen darstellt. Eine 0 bedeutet positiv, eine 1 negativ.

Beim Programmieren und bei der Eingabe und Ausgabe von Daten arbeitet man normalerweise im dezimalen Zahlensystem; zur Zahlenumwandlung gibt es fertige Systemprogramme oder Tabellen. Bei der Entwicklung von Hardware und bei der Fehlersuche in Programmen kann es jedoch vorkommen, daß sich der Entwickler mit Speicherinhalten beschäftigen muß. Diese werden in der Regel nicht binär, so wie sie im Speicher stehen, sondern hexadezimal ausgegeben. Das hexadezimale Zahlensystem entsteht durch Zusammenfassung von vier Dualstellen zu einem neuen Zeichen. Entsprechend **Bild 2-10** verwendet es die Ziffern 0 bis 9 und zusätzlich die Buchstaben A bis F.

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

Bild 2-10: Die 16 Hexadezimalziffern von 0 bis F

Die Dezimalzahl 123 lautet als achtstellige Dualzahl 01111011 und als zweistellige Hexadezimalzahl 7B. Das hexadezimale Zahlensystem hat 16 Ziffern; die Wertigkeiten der Stellen sind Potenzen zur Basis 16. In den Umrechnungsverfahren der Bilder 2-7 und 2-8 ist anstelle der 2 die 16 zu setzen. Auch binäre Speicherinhalte, die keine Dualzahlen sind wie z.B. Befehle oder Zeichencodierungen, werden ebenfalls kürzer hexadezimal ausgegeben. Der Buchstabe A ist entsprechend Bild 2-4 binär 01000001 hexadezimal 41.

2.3 Rechenschaltungen

Die Boolsche Algebra der Mathematik arbeitet mit den beiden (binären) logischen Zuständen "wahr" und "falsch". Die Digitaltechnik entwickelte daraus die Schaltalgebra. Sie bildet die Grundlage für das Rechnen mit Dualzahlen und binär codierten Daten. **Bild 2-11** faßt die wichtigsten logischen Funktionen zusammen.

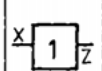
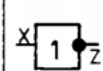
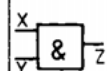
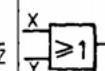
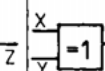
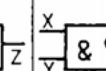
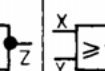
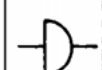
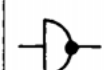
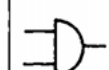
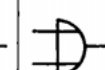
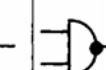
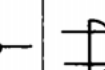
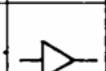
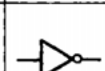
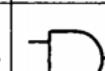
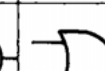
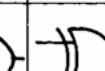
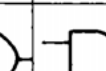
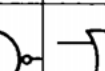
Name	JA	NICHT (NOT)	UND (AND)	ODER (OR)	EODER (XOR)	NICHT UND (NAND)	NICHT ODER (NOR)
Tabelle	X | Z 0 | 0 1 | 1	X | Z 0 | 1 1 | 0	X | Y | Z 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	X | Y | Z 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1	X | Y | Z 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	X | Y | Z 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	X | Y | Z 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0
neues Symbol							
altes Symbol							
amerik. Symbol							

Bild 2-11: Logische Grundfunktionen

Die **JA-Schaltung** zeigt an ihrem Ausgang immer den am Eingang anliegenden Zustand. Sie wird als Verstärker oder Treiber verwendet.

Die **NICHT-Schaltung** verneint oder negiert den am Eingang anliegenden Zustand. Sie wird zur Bildung des Komplementes bei der Darstellung negativer Zahlen verwendet.

Die **UND-Schaltung** hat nur dann am Ausgang eine 1, wenn beide Eingänge 1 sind. Dies gilt auch für UND-Schaltungen mit mehr als zwei Eingängen. Die UND-Schaltung bildet das logische Produkt nach dem kleinen Einmaleins der Dualzahlen.

Die **ODER-Schaltung** hat nur dann am Ausgang eine 0, wenn beide Eingänge 0 sind. Dies gilt auch für ODER-Schaltungen mit mehr als zwei Eingängen. Die ODER-Schaltung bildet die logische Summe, die jedoch für den Fall 1 + 1 = 10 korrigiert werden muß, da sich ein Übertrag auf die nächste Stelle ergibt.

Die **EODER-Schaltung** hat nur dann am Ausgang eine 0, wenn beide Eingänge gleich sind, also beide 0 oder beide 1. EODER bedeutet "Entweder oder, aber nicht alle beide ". Die EODER-Schaltung bildet die logische Differenz, bei der für den Fall 0 - 1 = 1 von der folgenden Stelle geborgt werden muß.

Die NICHT-UND- und die NICHT-ODER-Schaltungen entstehen durch eine zusätzliche Verneinung am Ausgang der UND- bzw. ODER-Schaltung. Die amerikanischen Bezeichnungen **NAND** für NOT-AND und **NOR** für NOT-OR sind bereits Bestandteil der deutschen Fachsprache geworden.

Für die Addition zweier Dualstellen ist eine Schaltung erforderlich, an deren Eingängen die beiden Dualstellen anliegen und an deren Ausgängen die einstellige Summe und der Übertrag auf die nächste Stelle erscheinen. **Bild 2-12** zeigt die Wertetabelle und die Schaltung eines Halbaddierers.

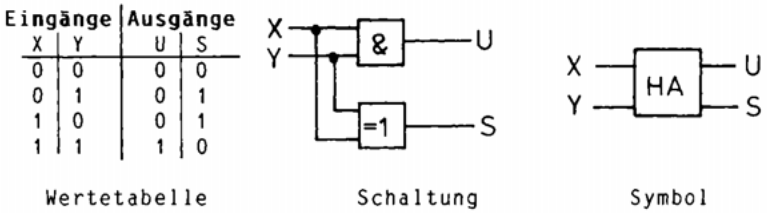


Bild 2-12: Halbaddierer

Ein Vergleich mit den Wertetabellen der logischen Grundfunktionen zeigt, daß die EODER-Schaltung die Summe und die UND-Schaltung den Übertrag bildet. Auf einen systematischen Entwurf einer logischen Schaltung aus der Wertetabelle kann an dieser Stelle nicht eingegangen werden. Der Halbaddierer eignet sich nur zur Addition der wertniedrigsten (letzten) Stellen zweier Dualzahlen, da bei allen folgenden Stellen ein Übertrag der vorhergehenden Stelle mit berücksichtigt werden muß. **Bild 2-13** zeigt die Wertetabelle und die Schaltung eines Volladdierers, der drei Eingänge und zwei Ausgänge hat.

Der erste Halbaddierer addiert die beiden Dualstellen X und Y. Die Zwischensumme läuft mit dem Übertrag UV der vorhergehenden Stelle über den zweiten Halbaddierer und bildet die Ergebnissumme S. Eine ODER-Schaltung addiert die beiden Teilüberträge der Halbaddierer zum Gesamtübertrag UN, der an den nächsten Volladdierer weiter zu reichen ist. **Bild 2-14** zeigt die Schaltung eines Addierwerkes aus acht Volladdierern, das zwei achtstellige Dualzahlen addieren kann, die parallel auf je acht Leitungen am Eingang ankommen.

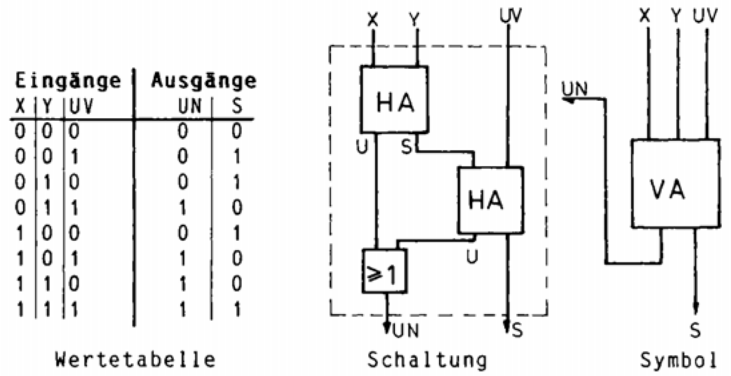


Bild 2-13: Volladdierer

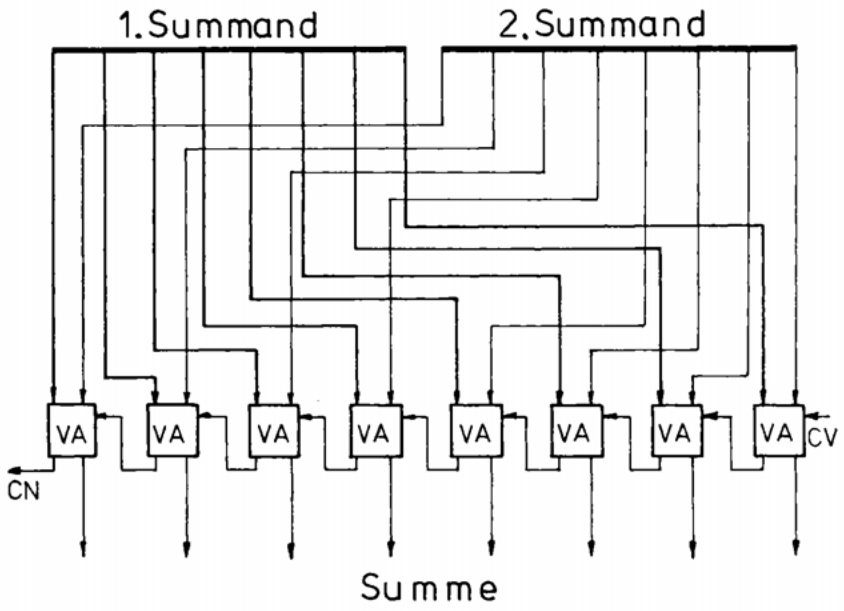


Bild 2-14: Achtstelliger Paralleladdierer

Der Übertragsausgang des werthöchsten Volladdierers wird mit C für Carry = Übertrag bezeichnet. Ist C = 1, so ist bei der Addition zweier achtstelliger Dualzahlen eine neunte Stelle entstanden. Dies kann als Fehleranzeige dienen, da der zulässige Zahlenbereich überschritten wurde. Der Addierer kann auch subtrahieren, wenn man die abzuziehende Dualzahl vorher mit einer NICHT-

Schaltung komplementiert und zusätzlich über den Übertragseingang des wertniedrigsten Volladdierers eine 1 addiert. Die Subtraktion wird auf die Addition der negativen Zahl zurückgeführt: A - B = A + (-B). **Bild 2-15** zeigt abschließend das Symbol einer Arithmetisch-logischen Einheit, die addiert, subtrahiert und logische Operationen ausführt.

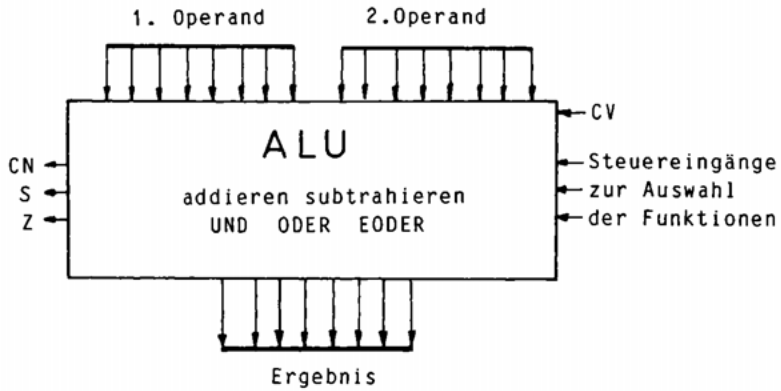


Bild 2-15: Arithmetisch-logische Einheit für acht Bit

ALU ist eine Abkürzung für Arithmetic-Logic Unit gleich Arithmetisch-logische Einheit. Sie ist Bestandteil des Rechenwerkes eines Mikroprozessors. Die ALU enthält zweimal acht Dateneingänge für die zu verknüpfenden Operanden und acht Datenausgänge für das Ergebnis. Der Übertragseingang CV addiert bei einer Subtraktion zusätzlich eine 1 zum Komplement oder kann bei einer Addition von mehr als acht bit langen Dualzahlen den Zwischenübertrag addieren. Daher ist der Übertrag des Übertragsausgangs C zwischen den Teiladditionen zu speichern. Der Ausgang S für Sign gleich Vorzeichen ist gleich dem werthöchsten Bit des Ergebnisses und enthält das Vorzeichen bei vorzeichenbehafteten Dualzahlen. Der Ausgang Z für Zero gleich Null zeigt über eine Logikschaltung (NOR mit acht Eingängen), ob das Ergebnis gleich Null ist. Über die Steuereingänge wird die gewünschte Operation der ALU ausgewählt, also Addition, Subtraktion, UND-Funktion, ODER-Funktion oder EODER-Funktion. An diesen Eingängen liegt beim Mikroprozessor der Funktionscode des Befehls.

2.4 Speicherschaltungen

Speicherschaltungen haben die Aufgabe, binäre Zustände (0 oder 1) aufzunehmen, zu speichern und auf Abruf wieder abzugeben. Die einfachste Speicherschaltung besteht aus zwei rückgekoppelten NAND-Schaltungen entsprechend **Bild 2-16** . Ein Flipflop kann auch aus zwei NOR-Schaltungen bestehen.

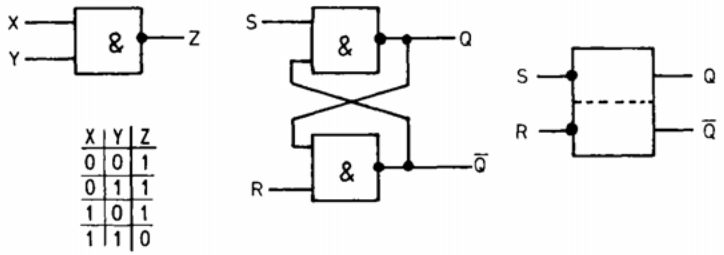


Bild 2-16: NAND-Flipflop

Das Wort "Flipflop" kommt aus der amerikanischen Laborsprache und bedeutet so viel wie "Klick-Klack". Die deutsche Bezeichnung "Bistabiler Multivibrator" hat sich nicht durchgesetzt. Die Schaltung hat zwei Eingänge und zwei Ausgänge. Mit dem S-Eingang kann man den Speicher auf 1 setzen, mit dem R-Eingang auf 0 rücksetzen. Der Ausgang Q ist gleich dem Speicherinhalt, der Ausgang $\overline{Q}$ ist durch einen Querstrich gekennzeichnet und enthält das Komplement (Verneinung) von Q.

Ruhe- oder Speicherzustand:
Bild 2-17 zeigt den Speicherzustand des NAND-Flipflops. Der linke Teil zeigt die Speicherung des Wertes Q = 0, der rechte den des Wertes Q = 1.

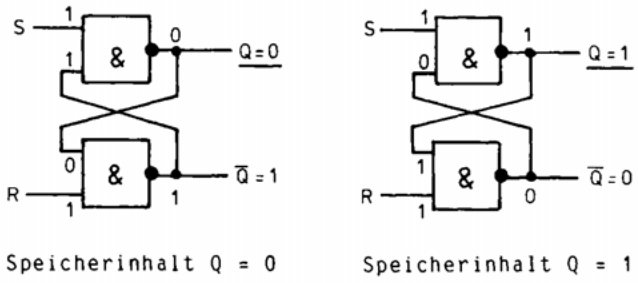


Bild 2-17: Speicherzustand des NAND-Flipflops

Ist ein Eingang der NAND-Schaltung 1, so hängt der Ausgang vom Zustand des anderen Eingangs ab. Ist der Speicherinhalt Q = 0 (linkes Bild), so liegt die 0 zusammen mit R = 1 am unteren NAND und ergibt am Ausgang $\overline{Q}$ = 1. Diese 1 wird auf das obere NAND zurückgeführt und ergibt zusammen mit S = 1 wieder den Ausgang Q = 0: die Schaltung speichert stabil den Inhalt 0.

Auch für den Speicherzustand Q = 1 (rechtes Bild) ergibt sich wieder ein stabiler Zustand, so daß also R = S = 1 auf jeden Fall einen der beiden Speicherzustände Q = 0 oder Q = 1 festhält (speichert).

Einschreiben einer 1 (Setzen):
Bringt man den Setzeingang S kurzzeitig auf 0, so ergibt sich immer der Ausgang und damit der Speicherinhalt Q = 1. R muß dabei auf 1 bleiben.

Einschreiben einer 0 (Rücksetzen):
Bringt man den Rücksetzeingang R kurzzeitig auf 0, so ergibt sich immer der Ausgang $\overline{Q}$ = 1 und damit der Speicherinhalt Q = 0. S muß dabei auf 1 bleiben.

Das NAND-Flipflop wird auch als RS-Flipflop bezeichnet. Es dient z.B. zum Entprellen von Schaltern und Tastern. Die Eingänge R und S schalten mit einem 0-Signal, sie sind also "aktiv LOW".

Das einfache RS-Flipflop kann entsprechend **Bild 2-18** zum D-Flipflop erweitert werden.

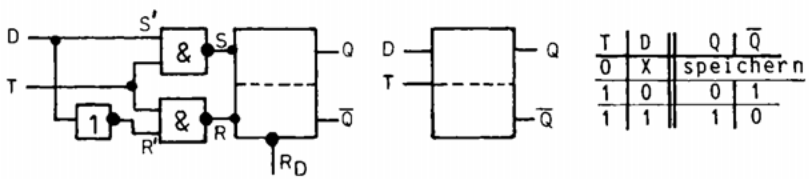


Bild 2-18: D-Flipflop

D bedeutet Delay wie Verzögerung. Dieses Flipflop hat nur noch einen einzigen Dateneingang D. Die NICHT-Schaltung sorgt dafür, daß die Eingänge R' und S' der beiden NAND-Schaltungen immer komplementär zueinander sind. Die NAND-Schaltungen kann man sich aufgeteilt denken in ein UND mit einem folgenden NICHT. Durch das UND werden die Daten nur dann weitergereicht, wenn der Takt T gleich 1 ist. Die NICHT-Schaltung komplementiert die Daten. Der D-Eingang ist jetzt "aktiv HIGH". D = 1 wird als Q = 1 gespeichert, D = 0 als Q = 0. **Bild 2-19** zeigt den zeitlichen Verlauf eines Schreibvorganges.

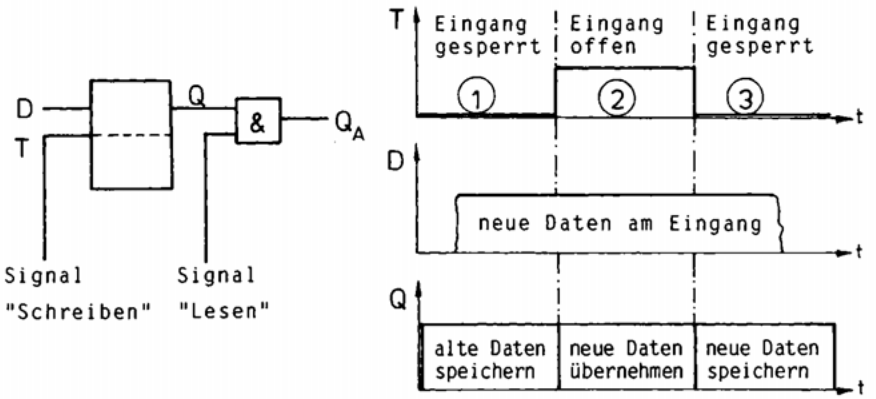


Bild 2-19: Schreiben eines D-Flipflops

Im Bereich 1 ist der Schreibtakt 0, damit sind die Eingänge R und S des RS-Flipflops unabhängig von den an D bzw. R' und S' anliegenden Daten immer 1. Der Eingang ist gesperrt, das Flipflop speichert.

Im Bereich 2 ist der Schreibtakt 1, damit werden die am Eingang D anliegenden Daten über R' und S' in das Flipflop übernommen und erscheinen am Ausgang Q. Während des Taktzustandes T = 1 dürfen sich die Daten nicht ändern. Das vorliegende Flipflop wird durch den Zustand des Taktes gesteuert. Daneben gibt es auch Flipflops, die durch eine Taktflanke gesteuert werden. Die Übernahme erfolgt dann nur zu dem Zeitpunkt, zu dem sich der Takt von 0 auf 1 (positive Flanke) oder von 1 auf 0 (negative Flanke) ändert.

Im Bereich 3 ist der Schreibtakt wieder 0. Damit ist die Datenübernahme wieder gesperrt, und das Flipflop bewahrt die Daten bis zum nächsten Schreibvorgang auf.

Der Ausgang Q des Flipflops kann jederzeit ohne Veränderung des Inhaltes gelesen werden. Dies kann durch das Anlegen eines Lesetaktes geschehen. D-Flipflops werden unter der Bezeichnung Latch - frei übersetzt Auffangspeicher - als Register für die Eingabe und Ausgabe von Daten verwendet.

Für rechentechnische Anwendungen ist es oft nötig, in einem Flipflop gleichzeitig einen alten Speicherinhalt im Hauptspeicher zu behalten und einen neuen Inhalt zunächst in einen Vorspeicher zu schreiben. **Bild 2-20** zeigt die Schaltung eines Master-Slave-Flipflops zusammen mit dem zeitlichen Verlauf der Datenübernahme.

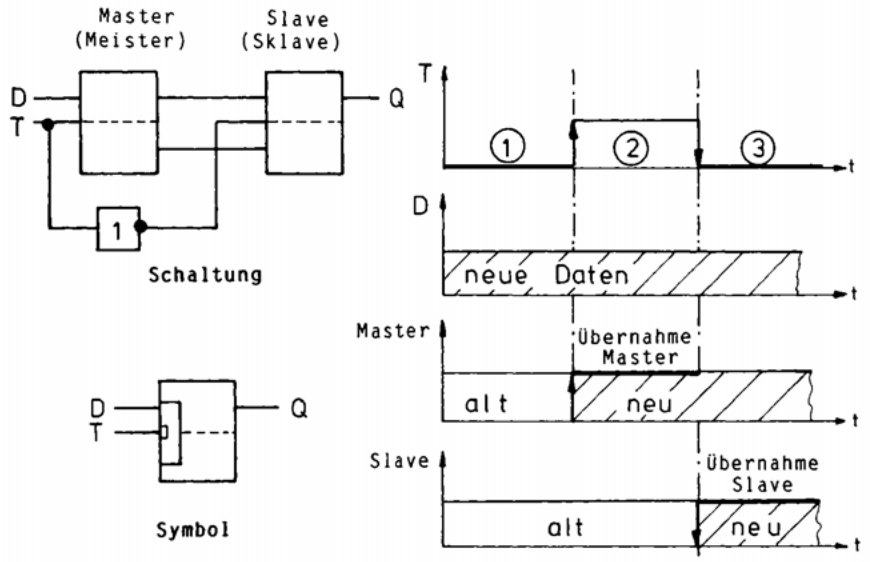


Bild 2-20: Master-Slave-Flipflop

Der Vorspeicher heißt Master gleich Meister, der Hauptspeicher Slave wie Sklave. Neue Daten liegen am Eingang des Meisters; der Sklave erhält seine Daten vom Meister. Die Übernahme erfolgt jedoch durch den negierten Takt des Sklaven zu unterschiedlichen Zeitpunkten.

Im Bereich 1 ist der Takt des Meisters 0. Dadurch ist sein Dateneingang gesperrt. Der Takt des Sklaven dagegen ist 1. Der Sklave übernimmt die Daten des Meisters; beide Flipflops haben den gleichen Inhalt.

Im Bereich 2 ist der Takt des Meisters 1, und er übernimmt die an seinem Eingang anliegenden Daten. Der Takt des Sklaven dagegen ist 0; daher behält dieser noch seinen alten Inhalt. Im Bereich 2 speichert also der Meister bereits die neuen Daten, während der Sklave noch die alten Werte festhält.

Im Bereich 3 übernimmt wie im Bereich 1 der Sklave die Daten des Meisters; beide Flipflops haben wieder den gleichen Inhalt.

Acht parallele Master-Slave-Flipflops entsprechend **Bild 2-21** bilden den Akkumulator, das wichtigste Datenregister im Rechenwerk eines Mikroprozessors.

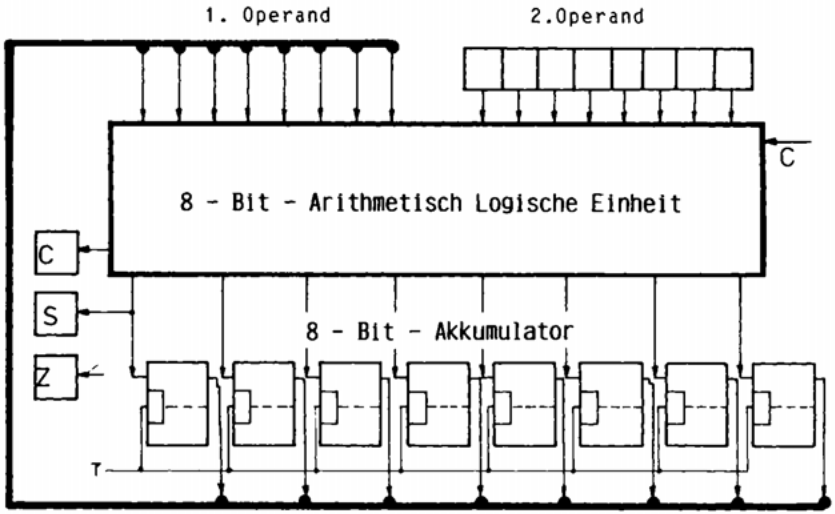


Bild 2-21: ALU und Akkumulator für acht Bit

Der Akkumulator gibt an seinen Ausgängen z.B. eine zu addierende Dualzahl an die arithmetisch-logische Einheit ab und nimmt an seinen Eingängen die Summe auf. Der zweite Summand kann z.B. aus einem aus D-Flipflops bestehenden Datenregister kommen. Akkumulator bedeutet Sammler. Er gibt seinen Speicherinhalt auf acht parallelen Leitungen gleichzeitig ab und nimmt an seinen acht Eingängen das Ergebnis auf acht parallelen Leitungen auf. Er wird daher auch als paralleles Schieberegister bezeichnet.

Eine Rückführung der Ausgänge eines Master-Slave-Flipflops auf die Eingänge entsprechend **Bild 2-22** liefert einen 2:1-Frequenzteiler, mit dem sich eine Zählerkette aufbauen läßt.

Jede fallende Taktflanke schaltet den Ausgang eines Zählelementes um, d.h. von 0 auf 1 oder von 1 auf 0. Die im Bild untereinander gezeichneten logischen Zustände der Takte bilden eine Dualzahl, die z.B. mit 0000 beginnend mit jedem Takt um 1 weitergezählt wird. Auf den größten Wert 1111 folgt wieder der Anfangswert 0000. Andere hier nicht behandelte Zähler lassen sich mit einem Anfangswert laden und wahlweise aufwärts oder abwärts zählen. Das Zählen geht schaltungstechnisch schneller und einfacher als die Addition oder Subtraktion der Zahl 1.

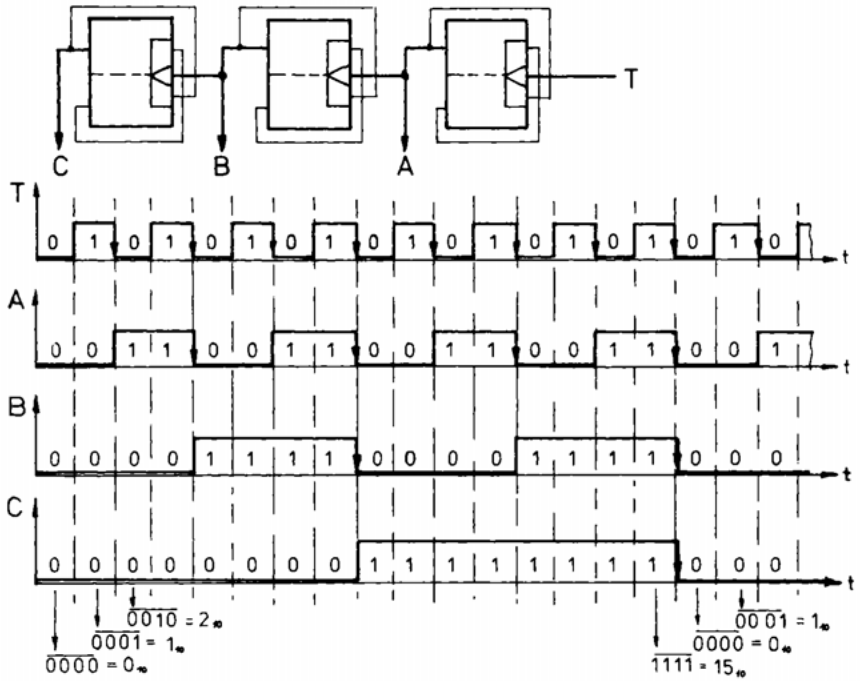


Bild 2-22: Vierstelliger Binärzähler

2.5 Aktive Zustände von Steuersignalen

Die Arbeitsgeschwindigkeit eines Mikrorechners hängt ab von den Schaltzeiten seiner Rechen- und Speicherschaltungen. Diese ergeben sich aus den Zeiten für den Aufbau und den Abbau von Ladungsträgern in den Halbleiterschichten. Schaltet man mehrere Logikbausteine (z.B. NICHT, UND, ODER) wie bei einem Volladdierer Bild 2-13 hintereinander, so addieren sich ihre Laufzeiten. Besonders zeitkritisch ist der achtstellige Paralleladdierer nach Bild 2-14, bei dem der Gesamtübertrag acht Volladdierer durchlaufen muß. Aus diesem Grunde werden derartige Schaltungen in der Praxis möglichst aus parallelen Logikelementen aufgebaut. Dadurch erhöht sich jedoch die Anzahl der Elemente und damit ihr Platz- und Leistungsbedarf. Die Bilder dieses Abschnitts zeigen nur die Funktionsweise der Rechen- und Speicherschaltungen, nicht jedoch ihren tatsächlichen Aufbau im Mikrorechner.

Der zeitliche Ablauf aller Funktionen eines Mikrorechners wird durch den Takt gesteuert. Dies ist ein von außen an den Mikroprozessor angelegtes Rechtecksignal von ca. 1 bis 10 MHz, für das es jedoch aus technologischen Gründen eine obere und auch eine untere Frequenzgrenze gibt. Innerhalb dieser aus den Datenblättern der Bausteine ersichtlichen Grenzen kann der Benutzer die Arbeitsgeschwindigkeit seines Mikrorechners durch den Takt selbst bestimmen. Aus dem Takt leitet das Steuerwerk des Mikroprozessors entsprechend **Bild 2-23** weitere Steuersignale ab.

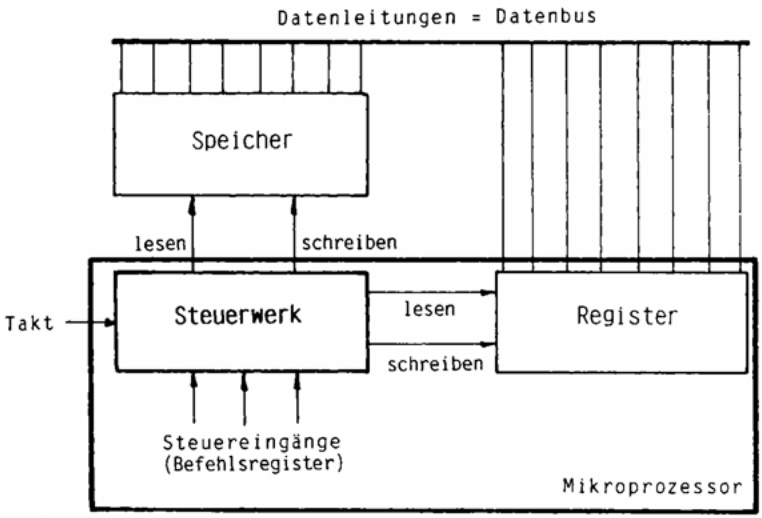


Bild 2-23: Steuerwerk und Steuersignale

Das Steuerwerk besteht aus Logik-, Speicher- und Zählschaltungen. Aus dem Takteingang und weiteren Steuereingängen werden die äußeren Steuersignale (z.B. Speicher Lesen und Schreiben) und die inneren Steuersignale (z.B. Takteingänge von Registern) abgeleitet. **Bild 2-24** zeigt als Beispiel den zeitlichen Verlauf der Signale "Lesen" und "Schreiben", die zu den Speicherbausteinen führen.

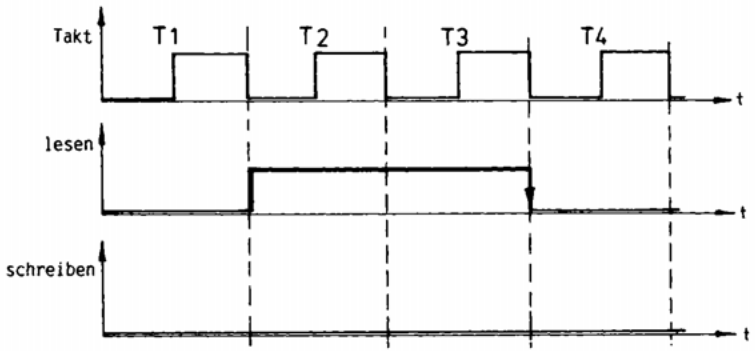


Bild 2-24: Zeitlicher Verlauf eines "aktiv-HIGH"-Lesesignals

Die beiden Steuersignale "Lesen" und "Schreiben" des Beispiels sind "aktiv HIGH". Ein hohes Potential bzw. eine logische 1 löst den gewünschten Vorgang aus.

Takt T1: Beide Steuersignale sind nicht aktiv.

Takt T2: Das Lesesignal ist aktiv, das Schreibsignal nicht.

Takt T3: Das Lesesignal ist aktiv, das Schreibsignal nicht.

Takt T4: Beide Steuersignale sind nicht aktiv.

Die Mikrorechnertechnik arbeitet jedoch vorzugsweise mit Steuersignalen, die "aktiv LOW" sind. Ein niedriges Potential bzw. eine logische 0 soll den gewünschten Vorgang auslösen. **Bild 2-25** zeigt wieder die Steuersignale "Lesen" und "Schreiben" jedoch für aktiv LOW.

Steuersignale, die aktiv LOW sind, werden meist durch einen Querstrich gekennzeichnet. Die Eingänge der Bausteine erhalten einen Punkt. Die Signale des Bildes 2-24 bzw. 2-25 könnten dazu dienen, die Übertragung von Daten zwischen dem Speicher und dem Mikroprozessor zu steuern. Im Takt T1 müssen die Daten noch verschiedene Schaltstufen zu den Speichern durchlaufen und sind noch nicht gültig. In den Takten T2 und T3 sind die Daten stabil und gültig und können von den Speicherschaltungen übernommen werden. Das Lesesignal legt nicht nur den Zeitpunkt, sondern auch die Richtung vom Speicher in den Mikroprozessor fest. Mit dem Schreibsignal werden Daten vom Mikro-

prozessor in den Speicher gebracht. Beide Signale bilden ein EODER. Entweder Lesen oder Schreiben, aber nicht beides gleichzeitig. Im Takt T4 müssen die Datenspeicher über Schaltstufen wieder abgeschaltet werden.

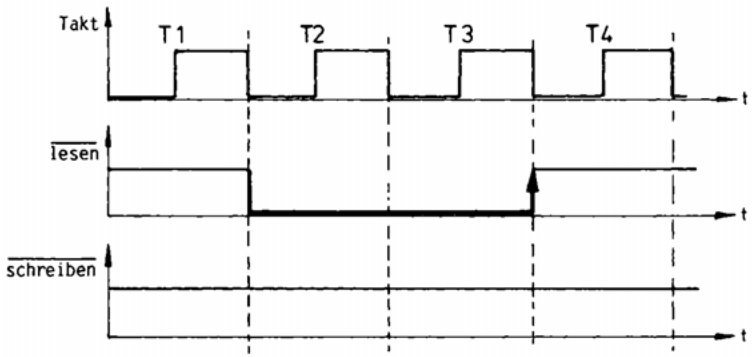


Bild 2-25: Zeitlicher Verlauf eines "aktiv-LOW"-Lesesignals

Die Logik von Steuersignalen läßt sich durch NICHT-Schaltungen leicht von aktiv LOW nach aktiv HIGH und umgekehrt umdrehen. Bei der logischen Verknüpfung mehrerer Steuersignale ist zu beachten, daß sich die UND- bzw. die ODER-Schaltung des Bildes 2-11 nur auf aktiv HIGH beziehen. Für die Verknüpfung von Steuersignalen, die aktiv LOW sind, ist entsprechend **Bild 2-26** die entgegengesetzte Logikfunktion zu wählen.

	logische **UND**-Verknüpfung	logische **ODER**-Verknüpfung
aktiv **HIGH**	X Y | Z 0 0 | 0 0 1 | 0 1 0 | 0 **1 1 | 1** X, Y → & → Z	X Y | Z 0 0 | 0 **0 1 | 1** **1 0 | 1** **1 1 | 1** X, Y → ≥1 → Z
aktiv **LOW**	X Y | Z **0 0 | 0** 0 1 | 1 1 0 | 1 1 1 | 1 X, Y → ≥1 → Z	X Y | Z **0 0 | 0** **0 1 | 0** **1 0 | 0** 1 1 | 1 X, Y → & → Z

Bild 2-26: Logische Verknüpfung von Steuersignalen

Für aktiv HIGH bildet die UND-Schaltung gleichzeitig auch die logische UND-Verknüpfung, denn der Ausgang ist nur dann 1, wenn beide Eingänge 1 sind. Entsprechendes gilt für das ODER. Auch hier stimmen Schaltung und Verknüp-

fung überein. Anders dagegen bei Signalen, die aktiv LOW sind. Die ODER-Schaltung hat nur dann am Ausgang eine 0 (aktiv LOW), wenn beide Eingänge 0 (aktiv LOW) sind. Entsprechend ist für eine logische UND-Verknüpfung zweier Signale aktiv LOW eine ODER-Schaltung einzusetzen. Die UND-Schaltung ist immer dann am Ausgang 0 (LOW), wenn mindestens einer der beiden Eingänge 0 (LOW) ist. Für eine logische ODER-Verknüpfung zweier aktiv LOW Signale ist also eine UND-Schaltung einzusetzen. **Bild 2-27** zeigt als Beispiel, wie aus den beiden Signalen Lesen und Schreiben ein neues Signal gewonnen wird, das dann aktiv LOW ist, wenn entweder gelesen oder geschrieben wird. Das Steuerwerk sorgt dafür, daß nicht beide Steuersignale gleichzeitig aktiv sein können.

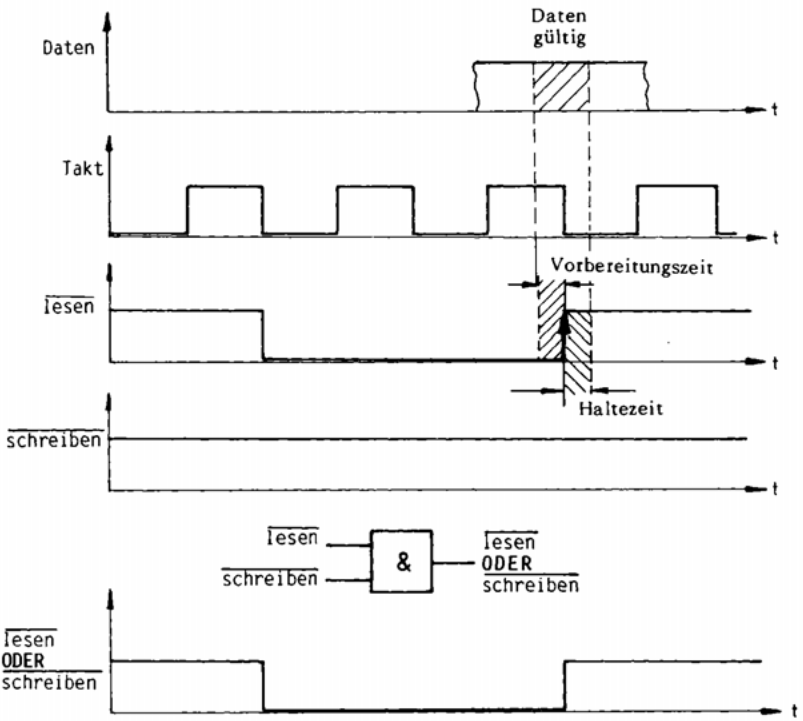


Bild 2-27: ODER-Verknüpfung bei aktiv LOW

Die Steuersignale "Lesen" und "Schreiben" wurden zunächst als zustandsgesteuert eingeführt. Die Datenübertragung erfolgt innerhalb der Zeit, in der das Signal im aktiven Zustand ist. Der genaue Zeitpunkt der Datenübernahme wird in vielen Fällen durch eine Taktflanke gesteuert. Bild 2-24 zeigt als Beispiel eine negative (fallende) Flanke, Bild 2-25 eine positive (steigende) Flanke. Die Daten müssen eine bestimmte Zeit vor der Flanke (Vorbereitungszeit) und eine bestimmte Zeit nach der Flanke (Haltezeit) stabil sein. Diese Zeiten wurden in das Bild 2-27 eingetragen.

Bild 2-28 zeigt am Beispiel eines handelsüblichen Flipflops (SN 7474) den Unterschied zwischen einer Steuerung durch einen Zustand und durch eine Flanke.

	Eingänge				Ausgänge		Funktion
	$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$	
Zustnd	0	1	X	X	1	0	setzen
	1	0	X	X	0	1	löschen
Flanke	1	1	↑	1	→ 1	0	setzen
	1	1	↑	0	→ 0	1	rücksetzen
keine	1	1	0	X	bleibt		speichern
Flanke	1	1	1	X	bleibt		speichern

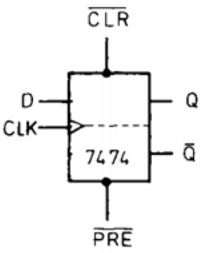


Bild 2-28: Flanken- und zustandsgesteuertes Flipflop

In der Tabelle könnten anstelle der logischen Bezeichnungen 0 und 1 auch die elektrischen Potentiale LOW und HIGH abgekürzt L und H stehen. Ein X bedeutet, daß der Zustand des Eingangs die Schaltung nicht beeinflußt.

Die beiden zustandsgesteuerten Eingänge $\overline{PRE}$ und $\overline{CLR}$ sind aktiv LOW und wirken wie ein RS-Flipflop entsprechend Bild 2-17 und 2-18. PRE bedeutet PRESET gleich setzen. Durch eine logische 0 (LOW-Potential) an diesem Eingang wird der Speicherzustand des Flipflops Q = 1 gesetzt. CLR bedeutet CLEAR gleich löschen. Durch eine logische 0 (LOW-Potential) an diesem Eingang wird der Speicherzustand des Flipflops auf Q = 0 gebracht. Sind beide Eingänge 1 (HIGH-Potential), so speichert das Flipflop seinen augenblicklichen Inhalt.

Der Dateneingang D wird durch eine positive (steigende) Flanke des Takteingangs CLK gesteuert. CLK bedeutet CLOCK gleich Taktgeber oder Uhr. Im Gegensatz zum D-Flipflop des Bildes 2-18 erfolgt die Datenübernahme durch die Taktflanke. Dabei müssen die Daten während der Vorbereitungszeit vor der Flanke und während der Haltezeit nach der Flanke stabil sein. Im Ruhezustand des Taktes speichert das Flipflop seinen augenblicklichen Inhalt.

Beim Aufbau eines Mikrorechners werden vorwiegend hochintegrierte Bausteine (Mikroprozessor, Speicher- und Ein/Ausgabebausteine) eingesetzt. Damit entfällt der Entwurf von Rechen- und Speicherschaltungen, da diese ja bereits in den Bausteinen vorhanden sind. Wichtig wird dagegen die logische und zeitliche Verknüpfung der Steuersignale, die die Datenübertragung zwischen den Bausteinen steuern. Dazu sind Grundkenntnisse der Digitaltechnik und der Arbeit mit TTL-Schaltungen unbedingt erforderlich. Für die Programmierung von Mikrorechnern genügt es, die Arbeitsweise der Schaltungen zu verstehen.

2.6 Speicherorganisation

Ein Mikrorechner kann über 500 000 binäre Speicherelemente in Form von Flipflops oder ähnlichen Schaltungen enthalten. Üblicherweise faßt man acht Bits zu einem Byte zusammen. Der Mikroprozessor enthält etwa 10 bis 20 Speicherbytes in Form von Registern. Der Befehls- und Datenspeicher eines Mikrorechners kann aus maximal 65 536 Bytes oder 64 Kilobytes bestehen. Jedes Byte erhält eine Adresse, mit der es eindeutig von allen anderen Bytes unterschieden werden kann.

Die Adresse eines Bytes wird wie sein Inhalt binär codiert und üblicherweise als Dualzahl angegeben. Der in **Bild 2-29** gezeigte Adreßdecoder ist eine Auswahlschaltung, die eine von vier Speicherstellen auswählt.

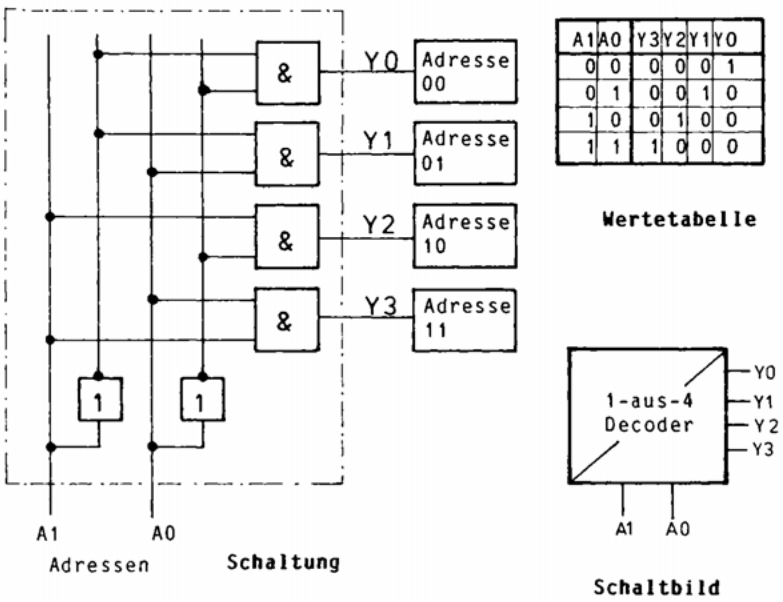


Bild 2-29: Adreßdecoder

Zur Auswahl von vier Speicherstellen sind als Adresse zwei Bits erforderlich, denn in zwei Bits lassen sich genau vier verschiedene Bitkombinationen darstellen. Dies sind die Dualzahlen 00, 01, 10 und 11 mit den dezimalen Werten 0, 1, 2 und 3. Zur Auswahl von acht Speicherstellen sind als Adresse drei Bits erforderlich; mit vier Bits lassen sich 16 Speicherstellen adressieren. Das