

Rechnerarchitekturen 2024

Dipl.-Ing. (FH) Andreas Fecht

Duale Hochschule Mannheim

1 Historisches	6
1.1 <i>Analog - Digital</i>	6
1.1.1 Analogrechner	6
1.1.2 Digitalrechner	6
1.2 <i>Mechanik</i>	7
1.3 <i>Elektromechanik</i>	8
1.4 <i>Röhren</i>	8
1.5 <i>Halbleiter</i>	9
1.6 <i>Chronologie</i>	10
1.6.1 Einfache Rechenmaschinen	10
1.6.2 Programmierbare Rechner.....	12
1.6.3 Software-Bugs	16
2 Systemkomponenten	17
2.1 <i>Klassifizierung von Computern</i>	17
2.2 <i>Klassifizierung von Betriebssystemen</i>	18
2.2.1 Bitbreite	18
2.2.2 64-Bit-Verwechslung.....	18
2.2.3 NX-Bit	18
2.3 <i>Standard Systemaufbau</i>	19
2.3.1 Überblick	19
2.3.2 Bus	19
2.3.3 Arbeitsspeicher/Cache	20
2.3.4 CPU	21
2.3.5 FPU.....	25
2.3.6 Massenspeicher.....	26
2.3.7 Redundanz	37
2.3.8 Power und Reset	39
2.4 <i>Architekturarten</i>	41
2.4.1 Von-Neumann	41
2.4.2 Harvard	42
2.5 <i>PC-Bussysteme</i>	43
2.5.1 ISA.....	43
2.5.2 PCI.....	43
2.5.3 PCI-Express	44
3 CPU.....	46
3.1 <i>Zahlendarstellung</i>	46
3.1.1 Gleitkomma (Floating Point).....	47
3.2 <i>ALU</i>	48
3.3 <i>Rechenwerk</i>	49
3.3.1 Addition	49
3.3.2 Subtraktion.....	55
3.3.3 Shift	56

3.3.4	Multiplikation	58
3.3.5	Division	60
3.3.6	Faktor 2 hoch x	61
3.3.7	Faktor 256 hoch x	61
3.3.8	MAC	62
3.3.9	SISD / SIMD	63
3.3.10	Sättigungsarithmetik (MMX)	63
3.4	<i>Steuerwerk</i>	64
3.4.1	Befehlszyklus	64
3.4.2	Befehlsaufbau	64
3.4.3	CISC/RISC	67
3.4.4	Pipelining	68
3.5	<i>Gleitkommawerk (FPU)</i>	71
3.5.1	Aufbau	71
3.5.2	Rechenweise	72
3.6	<i>Desktop CPUs von Intel</i>	73
3.6.1	Intel CPU Evolution	73
3.6.2	Multiplikationsperformance	74
3.6.3	Chipaufbau 80486	74
4	Busse	75
4.1	<i>Zusammenspiel aller Busse</i>	75
4.2	<i>Ein/Ausgabe (I/O)</i>	76
4.2.1	Memory Mapping	76
4.2.2	Polling	78
4.2.3	Interrupt	78
4.3	<i>Multiplexing</i>	79
4.4	<i>Burst Mode</i>	80
4.5	<i>DMA</i>	81
4.6	<i>MMU</i>	82
5	Speicher	83
5.1	<i>Speicherhierarchie</i>	83
5.2	<i>Flüchtige Speicher (RAM)</i>	84
5.2.1	Statisch (SRAM)	84
5.2.2	Dynamisch (DRAM)	85
5.3	<i>Nichtflüchtige Speicher (ROM)</i>	90
5.3.1	Allgemeines	90
5.3.2	EPROM	92
5.3.3	EEPROM	93
5.3.4	Flash	94
5.3.5	Modernere Speicherentwicklungen	98
5.4	<i>Fehlerkorrektur</i>	99
5.4.1	Softerror	99
5.4.2	Parity	100
5.4.3	ECC	100

6 Schnittstellen.....	101
6.1 <i>Master - Slave</i>	101
6.2 <i>Parallel</i>	101
6.2.1 Centronics.....	101
6.2.2 IDE/ATA	103
6.2.3 SCSI.....	105
6.3 <i>Seriell</i>	106
6.3.1 Allgemeines	106
6.3.2 RS232/RS485/RS422	110
6.3.3 SPI	115
6.3.4 I ² C	117
6.3.5 SATA.....	119
6.3.6 SAS.....	120
6.3.7 M.2.....	120
6.3.8 USB	121
6.3.9 Netzwerke.....	128
7 Displays	136
7.1 <i>Allgemeines</i>	136
7.2 <i>Einfache Displays</i>	136
7.2.1 Leuchtdioden	136
7.2.2 LC-Anzeigen	138
7.3 <i>Komplette Bildschirme</i>	141
7.3.1 Grundlegendes	141
7.3.2 VGA	142
7.3.3 DVI.....	143
7.3.4 HDMI.....	144
7.3.5 Display Port	144
7.4 <i>Grafiksysteme</i>	145
7.4.1 Ohne Intelligenz	145
7.4.2 GPUs.....	145
8 Aktueller CPU-Markt.....	149
8.1 <i>Hersteller von Desktop-CPUs</i>	149
8.2 <i>Hersteller ARM/Keil</i>	149
8.3 <i>CMSIS</i>	150
8.4 <i>SOCs (System On Chip)</i>	151
8.4.1 Klassifizierung.....	151
8.4.2 Hersteller von Signalprozessoren	151
8.4.3 Hersteller von Embedded Mikrocontrollern.....	151
8.4.4 Debugging & ISP.....	152
9 Beispielprojekt moderner Embedded Systementwurf.....	153
9.1 <i>Projektidee</i>	153
9.1.1 Beschreibung	153

9.1.2	Simulation.....	154
9.2	<i>Hardware</i>	155
9.2.1	Schaltung	155
9.2.2	Leiterplatte.....	156
9.3	<i>Software</i>	157
9.3.1	Beispiel der Firma STM mit STM32CubeMX.....	157
9.3.2	Blick auf die automatisch erzeugte Software	158
9.3.3	Anwenderprogramm	162
10	Abbildungsverzeichnis	166
11	Stichwortverzeichnis	171

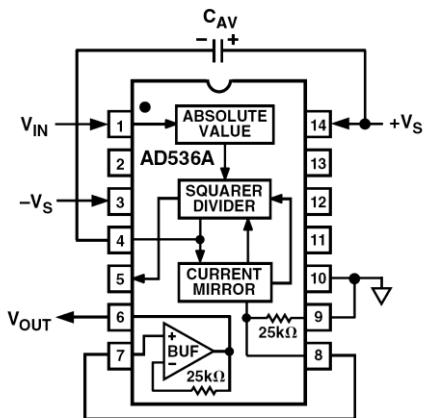
1 Historisches

1.1 Analog - Digital

1.1.1 Analogrechner

Analogrechner gibt es im Prinzip schon immer. Man könnte jede Anlage, die z.B. einen gemessenen Wert skaliert anzeigt, als Analogrechner bezeichnen.

Später wurden mit der Halbleitertechnik mit Operationsverstärkern einfache Analogrechner realisiert. Das wird heute in der Messtechnik teilweise immer noch gemacht, da eine Digitalisierung erst nach einer umfangreichen Messsignalaufbereitung erfolgen kann. Diese Aufbereitung könnte man als Analogrechner bezeichnen.



Beispiel Effektivwertgleichrichter mit Schaltkreis AD536 von Analog Devices:

$$U_{eff} = \sqrt{\frac{1}{T} \cdot \int_0^T U^2 dt}$$

Der Schaltkreis führt auf analogem Weg eine Berechnung des Effektivwertes einer Wechselspannung durch.

Abbildung 1 Effektivwertgleichrichter als Analogrechner

1.1.2 Digitalrechner

Die ersten Entwürfe Digitalrechner zu bauen, fingen schon im 17. Jahrhundert an. Es gab viele mechanische Rechenmaschinen, der Abakus ist der bekannteste und die chinesische Version Suanpan ist bis heute im asiatischen Raum immer noch in Gebrauch.

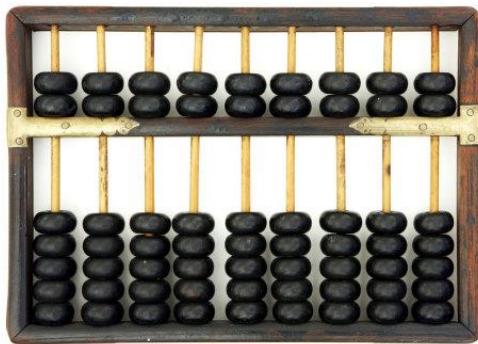


Abbildung 2 Suanpan

1.2 Mechanik

Rechenstäbe waren die ersten Rechenhilfen in der Antike und wurden in der Hauptsache zum Multiplizieren gebraucht.



Abbildung 3 Neperianische Rechenstäbchen

Zusammen mit dem Abakus hatte man damals für die Multiplikation und Addition bereits gute Werkzeuge.



Abbildung 4 Rechenstäbchen mit Abakus

Der Rechenschieber war bis in die 70er-Jahre das Standardwerkzeug der Ingenieure und ersetzte die vorher üblichen Logarithmentafeln.

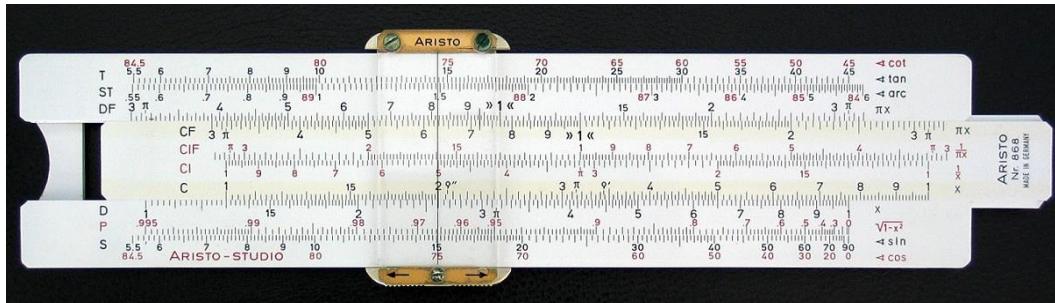


Abbildung 5 Rechenschieber

Gute Online-Simulation eines typischen Rechenschiebers:

<http://www.stefanv.com/calculators/aristo970/>

Die Maschinen, die den heutigen Computern nächsten sind, sind die Entwicklungen von Konrad Zuse. Seine Entwürfe ähneln sehr den heutigen Maschinen. Insbesondere der Aufbau mit Steuerwerk, Rechenwerk, Speicher und der grundlegende Entwurf mit binären Zahlen. Viele Registrierkassen bis in die 70er-Jahre arbeiteten fast immer noch rein mechanisch. Später gab es anstatt der Kurbel einen Elektromotor.

1.3 Elektromechanik

Das Grundelement für einen elektromechanischen Computer ist das Relais. Der erste voll funktionsfähige Digitalrechner war 1941 der Z3 von Konrad Zuse und bestand aus ca. 3000 Relais. Leistungsaufnahme ca. 4 kW, Gewicht 1 Tonne, Rechengenauigkeit 22 Bit.

1.4 Röhren

In den 40er und 50er-Jahren gab es einige Versuche mit Röhren einen digitalen Computer zu bauen. Bei ENIAC, dem bekanntesten Modell der US-Armee wurden ca. 18000 Röhren verbaut. Die Leistungsaufnahme lag bei etwa 174 kW. Da von den Röhren quasi immer welche defekt waren, war sie eine sehr unzuverlässige Maschine.

Der erste voll funktionsfähige Computer mit Röhren (Colossus) wurde in Großbritannien 1943 in Betrieb genommen. Es war eine sehr spezialisierte Maschine zur Dechiffrierung von geheimen Nachrichten. Aufbau mit 1500 Röhren, Leistungsaufnahme 4,5 kW.

1.5 Halbleiter

In den 1960er-Jahren hat es einige Computer aus diskreten Bauteilen gegeben. Große Stückzahlen wurden jedoch nie gebaut, weil der erste integrierte 4-Bit-Mikroprozessor von Intel (4004) schon ca. 1970 zur Verfügung stand. Erst einige Jahre später wurde mit dem 8080 von Intel und dem 6800 von Motorola die Massenproduktion möglich.

Ein neueres Projekt einen Computer ohne hochintegrierte Bausteine zu realisieren, wurde von Dennis Kuschel in Leben gerufen www.mycpu.eu

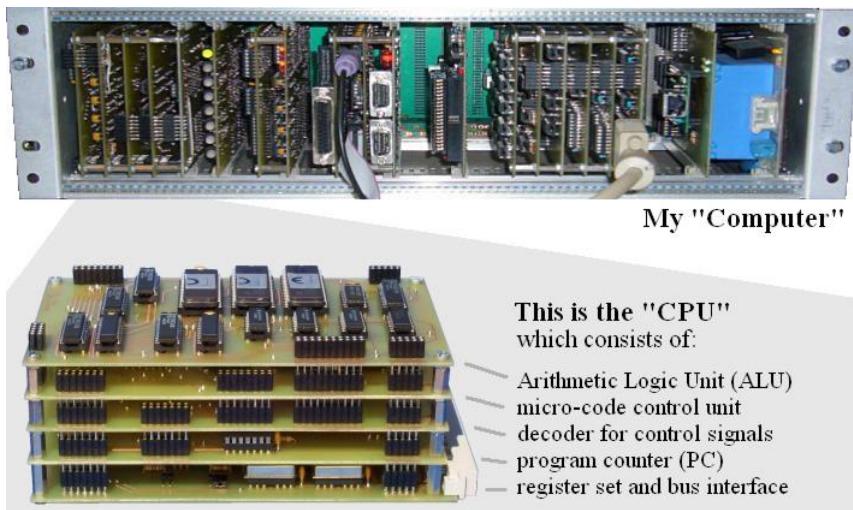


Abbildung 6 MyCPU

Eine weitere sehr interessante neuere Entwicklung von 2016 ist die Realisierung einer vollständigen 6502 CPU mit über 4000 Einzeltransistoren und über 300 LEDs.

www.monster6502.com



Abbildung 7 Monster 6502

Die maximale Taktfrequenz beträgt ca. 50 kHz.

1.6 Chronologie

1.6.1 Einfache Rechenmaschinen

- 1623 Erste Rechenmaschine von Professor Wilhelm Schickard



Abbildung 8 Rechenmaschine von Professor Wilhelm Schickard

- 1642 Rechenmaschine für sechsstellige Addition und Subtraktion Blaise Pascal

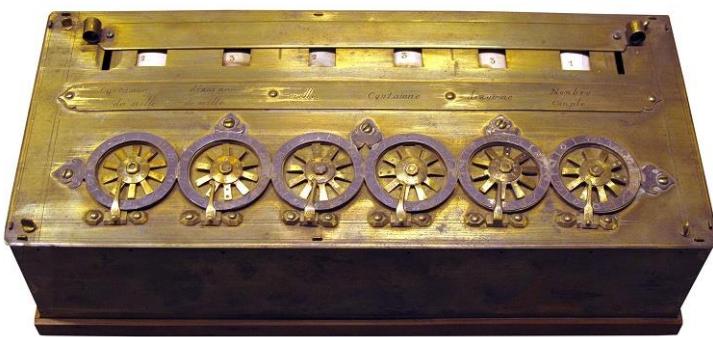


Abbildung 9 Rechenmaschine für sechsstellige Addition und Subtraktion Blaise Pascal

- 1673 Rechenmaschine von Leibniz



Abbildung 10 Rechenmaschine von Leibniz

- 1774 Rechenmaschine von Gottfried Wilhelm Hahn

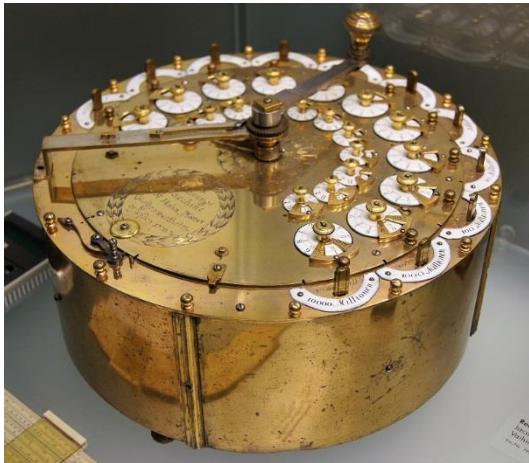


Abbildung 11 Rechenmaschine von Gottfried Wilhelm Hahn

- 1820 Arithmometer von Charles Xavier Thomas de Colmar



Abbildung 12 Arithmometer

- 1937 Taschenrechenmaschine Curta von Curt Herzstark



Abbildung 13 Curta von Curt Herzstark

1.6.2 Programmierbare Rechner

- 1801 erster mechanisch gesteuerter Webstuhl von Jacquard

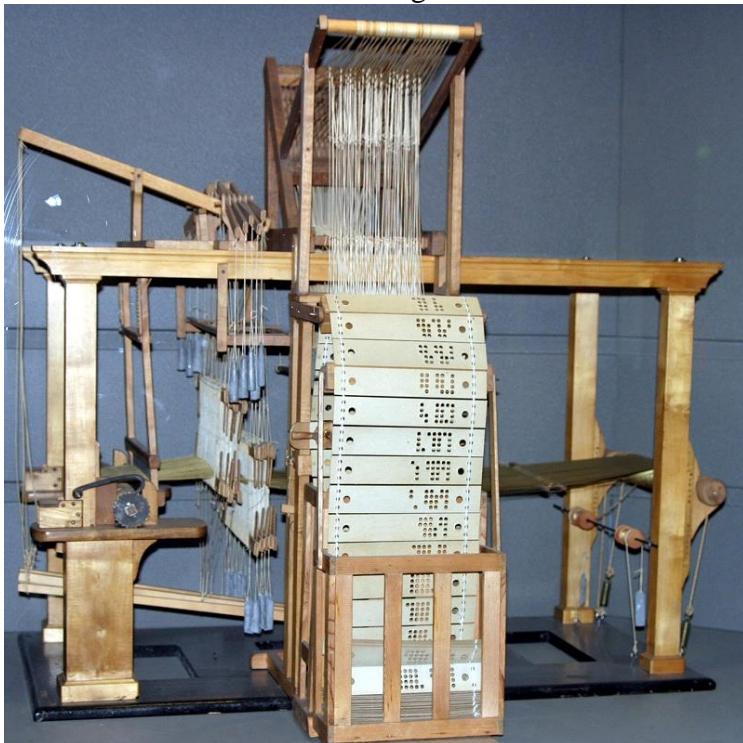


Abbildung 14 Webstuhl von Jacquard

- 1833 Analytical Engine von Charles Babbage

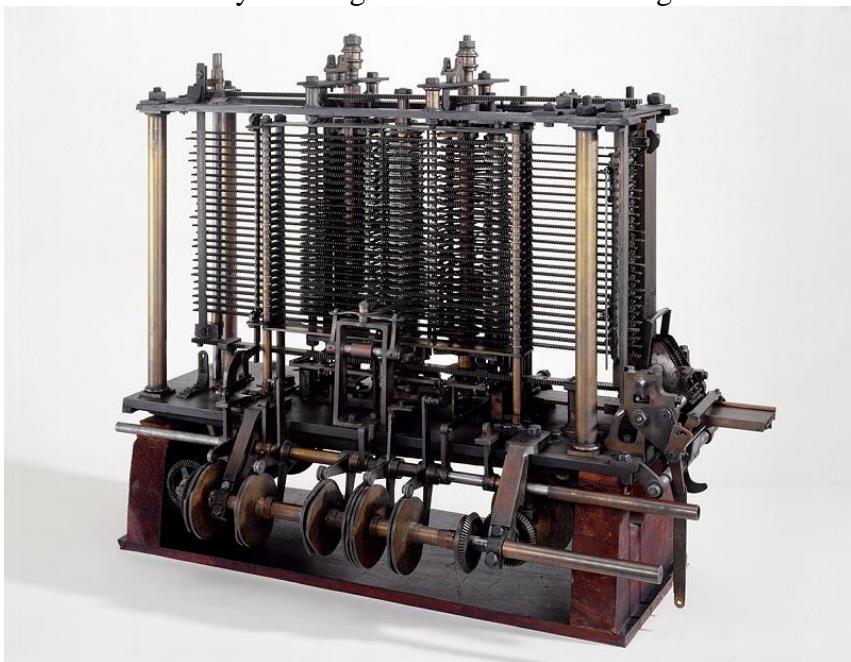


Abbildung 15 Analytical Engine

-
- 1886 Volkszählung in den USA mit Maschine von Hermann Hollerith



Abbildung 16 Maschine von Hermann Hollerith

- 1934 erster mechanischer binärer Computer Z1 von Konrad Zuse



Abbildung 17 Z1 von Konrad Zuse

- 1937 Erste Turing Maschine von Alan Turing

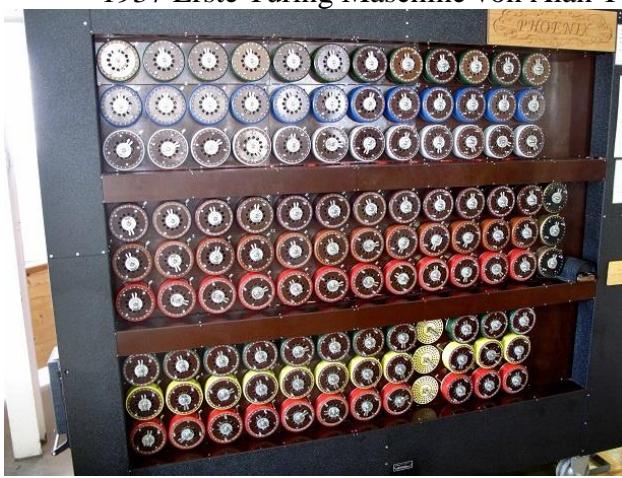


Abbildung 18 Turing Maschine von Alan Turing

- 1941 Relaiscomputer Z3 von Konrad Zuse

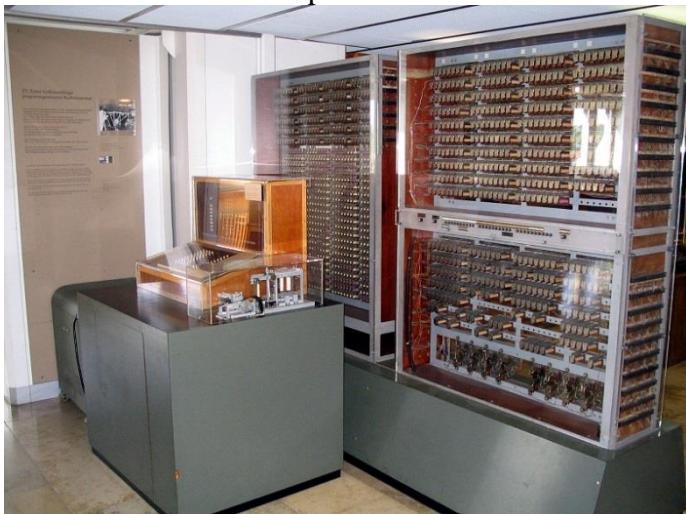


Abbildung 19 Z3 von Konrad Zuse

- 1946 Röhrencomputer ENIAC der US-Armee

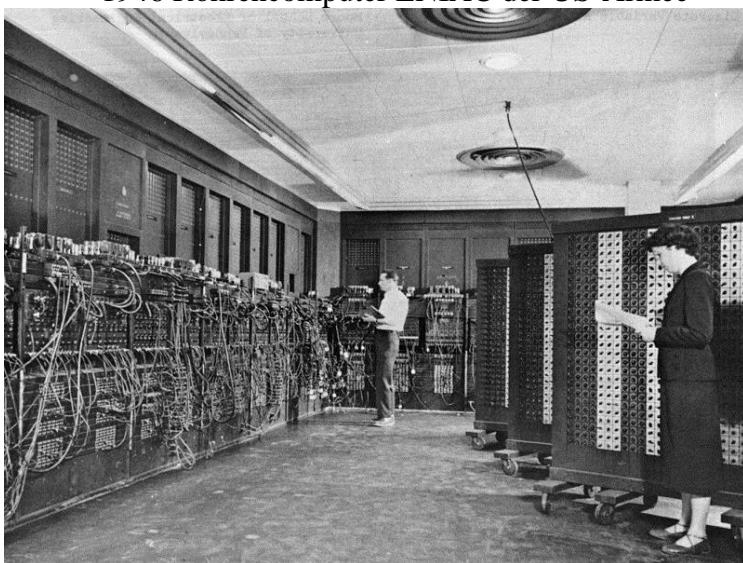


Abbildung 20 ENIAC

- 1949 Röhrencomputer EDSAC von M. V. Wilkes

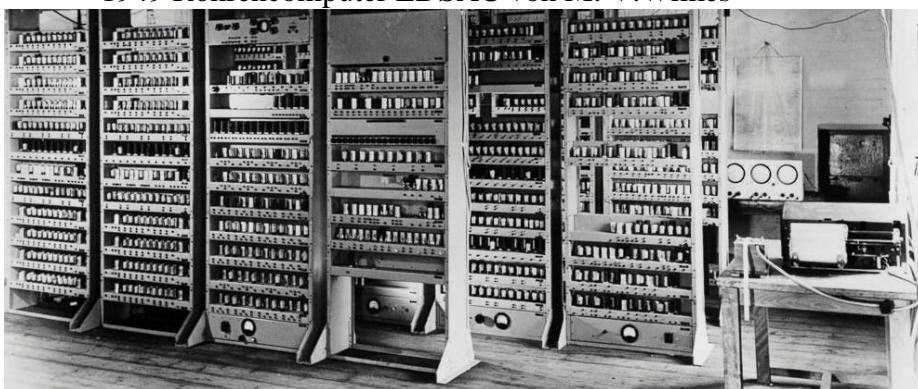


Abbildung 21 EDSAC

- 1959 erster Mikrocomputer von DEC, Auflage 50 Stück
- 1960 ALGOL erste Hochsprache
- 1965 erster Massen-Minicomputer, Auflage 50000 Stück
- 1971 erster 4-Bit-Mikroprozessor 4004 von Intel
- 1974 8080 8-Bit-Mikroprozessor von Intel
- 1976 erster Mikrocontroller, Prozessor, Speicher und I/O auf einem Chip
- 1981 erster IBM-PC

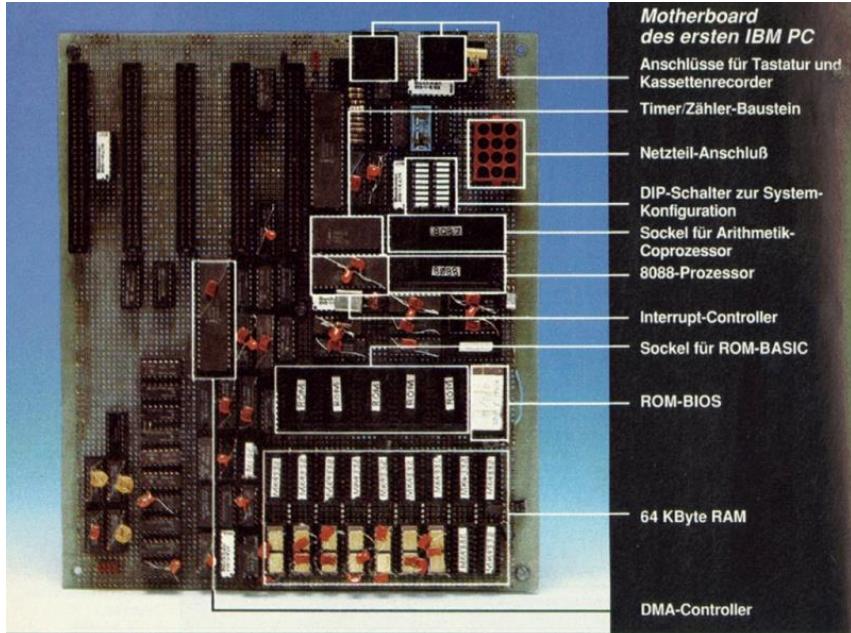


Abbildung 22 Erstes IBM-PC Motherboard

- 1985 32-Bit-Prozessorarchitektur einiger Hersteller (Intel Motorola AMD, ARM)
- 1990 diverse 32 Bit Prozessorarchitekturweiterentwicklungen vieler Hersteller
- 1995 ARM7TDMI 32-Bit-CPU mit Hardware-Multiplier und Debugger
- 2000 erste 64-Bit-Prozessorarchitekturen einiger Hersteller
- 2005 Beginn der 32-Bit-ARM-Cortex Architektur
- 2012 Erweiterung der ARM-Architektur auf 64 Bit
- 2018 Neue Prozessorarchitekturen 80x86 von Intel/AMD, ARM
- 2019 Erste Verbreitung der RISC-V Architektur
- 2020 Erste asymmetrische Multicore-Entwürfe (Apple M1, Intel 12. gen)

1.6.3 Software-Bugs

1.6.3.1 Historisches

Der Begriff Bug bei Computerprogrammen stammt vom „Mark II Aiken Relay Calculator“, als bei einer Fehlersuche 1947 innerhalb der Maschine eine Motte gefunden wurde. Seither werden Fehler in Computerprogrammen als Bugs bezeichnet.



Abbildung 23 Erster Software Bug

Der zuständige Service-Techniker hat den gefundenen „Bug“ ins Logbuch geklebt.

1.6.3.2 Klassifizierung

Bohr-Bug

Ein Bohr-Bug ist ein leicht zu findender Fehler, dessen Auswirkungen einfach und immer zu reproduzieren sind(deterministisches Verhalten).

Mandel-Bug oder Race-Condition

Ein Mandel-Bug entsteht oft durch einzelne miteinander kommunizierende Softwareteile, die sich manchmal je nach Laufzeit mit unterschiedlichen Ergebnissen unterschiedlich verhalten (nicht deterministisches Verhalten).

Andere Bezeichnungen: Monte-Carlo, Race-Condition oder auch Dead-Lock.

Heisen-Bug

Ein Heisen-Bug ist ein Phänomen, welches beim Einkreisen von Softwarefehlern passiert. Durch die Fehlersuche muss der Code verändert werden, wodurch sich auch das Fehlerverhalten ändern kann. Der Fehler wandert scheinbar immer von der Stelle weg, die man gerade untersucht.

Schroedin-Bug

Ein Schroedin-Bug ist ein Softwarefehler, der schon immer in einer Software vorhanden war, aber noch nie irgendeine schädliche Auswirkung hatte und nur durch Zufall im Quellcode gefunden wurde.

2 Systemkomponenten

2.1 Klassifizierung von Computern

Moderne Computer werden hauptsächlich durch die Bitbreite der Register in der CPU bzw. des Datenbusses unterschieden:

Gängige Werte heute sind:

- 8 Bit
- 16 Bit
- 32 Bit
- 64 Bit

In neuester Zeit zeichnet sich ein Trend zum 32-Bit-System ab. 16-Bit Systeme sind praktisch schon ausgestorben. 8-Bit-Systeme werden im Embedded-Bereich immer noch neu entwickelt und genutzt. 64-Bit-Systeme setzen sich langsam durch.

Manche Entwürfe haben unterschiedliche Datenbusbreiten innerhalb und außerhalb der CPU. Das führt manchmal zu Verwechslungen, was von den Marketingabteilungen durchaus beabsichtigt ist.

Früher hat man Computersysteme noch unterschieden in:

- Großrechner (Mainframe)
- Minirechner (Workstation)
- Mikrocomputer (PC)
- Mikrocontroller (8051, ARM)

Ein Mikrocontroller zeichnet sich dadurch aus, dass alle notwendigen Komponenten auf einem einzigen Chip enthalten sind. In neuester Zeit wird auch der Taktoszillator mit integriert.

Mittlerweile ergeben sich durch die Miniaturisierung neue Strukturen, die sich auch noch ständig verändern. Manche Mikrocontroller sind so leistungsfähig, dass man damit vollständige Desktopcomputer bauen kann -> Beispiel RASPI.

2.2 Klassifizierung von Betriebssystemen

2.2.1 Bitbreite

Die Bitbreite von Betriebssystemen wird hauptsächlich durch die Menge des adressierbaren Speichers unterschieden. Obwohl eine CPU theoretisch mehr Speicher adressieren könnte, weil sie einen mehr als 32 Bit breiten Adressbus besitzt, nutzen manche Betriebssysteme nur Teile davon.

Heutzutage sind folgende Systeme auf dem Markt.

- 16 Bit (max. 64 Kilobyte)
- 20 Bit (16 Bit + Segmentierung bis 1 Megabyte)
- 32 Bit (max. 4 Gigabyte)
- 64 Bit (max. 16 ExaByte = 18 Millionen Terabyte)

2.2.2 64-Bit-Verwechslung

Der aktuelle Hype auf 64-Bit-Betriebssysteme entstand in der Hauptsache durch eine Verwechslung der Bitbreite des Betriebssystems mit der Systembreite der CPU. Da bei einem 64-Bit-Betriebssystem jede Speicheradresse 64 Bit breit sein muss, dauert die Adressierung eines Datums immer doppelt so lange als bei einem 32-Bit-Betriebssystem, weil die Adresse eines Datums vorher immer über den Datenbus übertragen werden muss. Vorteile eines 64-Bit-Betriebssystems gibt es nur, wenn mehr als 4 Gigabyte Speicher im Spiel sind und eine Anwendung dies auch benötigt.

Des Weiteren gibt es zurzeit keine Prozessoren mit mehr als 45 echten Adressleitungen, d.h., 64-Bit-Betriebssysteme sind eigentlich nur max. 45-Bit-Betriebssysteme und können nicht mehr als 256 Terabyte echten Speicher ansprechen. Das Fehlen der untersten 3 Adressleitungen bei 64-Bit-Prozessoren trägt noch zur zusätzlichen Verwirrung bei (siehe Kapitel 2.3.4.3).

2.2.3 NX-Bit

Da die höchsten Bits der 64-Bit-Adresse wahrscheinlich nie benötigt werden, haben einige Prozessorhersteller damit angefangen Sonderbedeutungen einzubauen z.B. das höchste Bit der Adresse (Bit Nr. 63) wird benutzt, um zu kennzeichnen, ob an der entsprechenden Adresse Daten oder ausführbarer Programmcode abgelegt ist. Das dient dazu, es Schadprogrammen schwieriger zu machen Schadcode in den Speicher einzuschleusen. Dieses No-eXecute-Bit (NX-Bit) wurde von AMD 2003 mit dem Athlon64 eingeführt und wird zur Zeit von allen Betriebssystemen unterstützt.

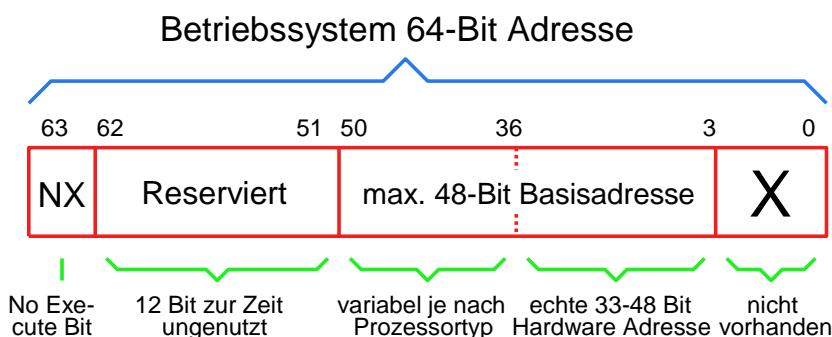


Abbildung 24 52+1-Bit Adresse

Diese Vorgehensweise verletzt die Von-Neumann-Struktur (siehe Kapitel 2.4.1) eines Computers, bei dem es im Prinzip nur eine Sorte Speicher für Daten und Programme geben darf.

2.3 Standard Systemaufbau

2.3.1 Überblick

Fast alle Computer sind nach einem ähnlichen Schema aufgebaut.

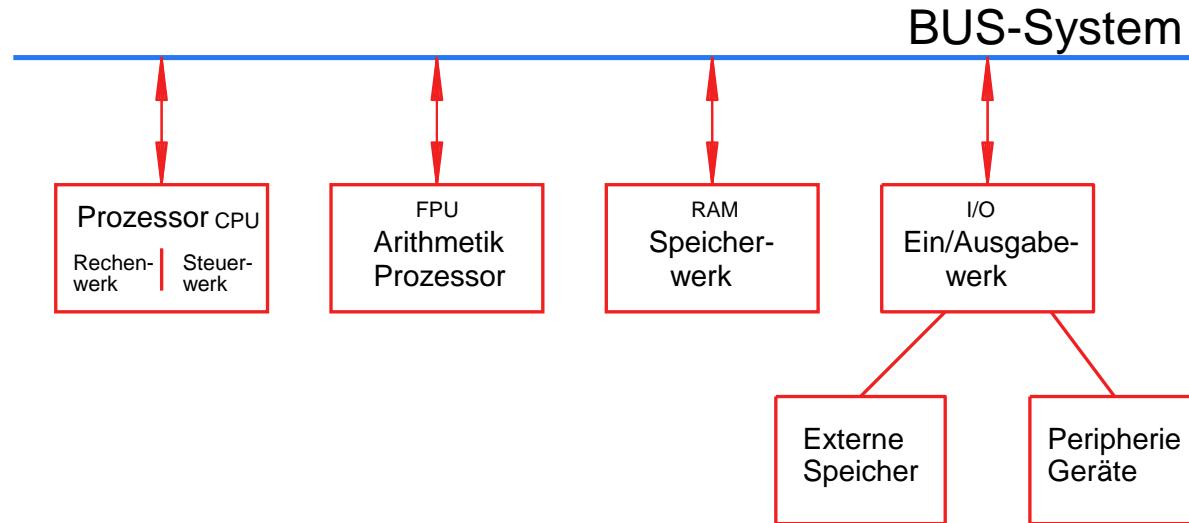


Abbildung 25 Systemaufbau

Von der früheren Klassifizierung sind quasi nur noch folgende übriggeblieben:

- Mikrocomputer, stationär (PC) oder portabel (Laptop, Mobiltelefone...)
- Mikrocontroller, in fast allen Embedded-Devices

2.3.2 Bus

Über den Bus wird sämtliche Kommunikation zwischen den einzelnen Komponenten abgewickelt. Er ist normalerweise in 3 Grundfunktionen unterteilt.

- Datenbus
- Adressbus
- Steuerbus

Der **Datenbus** gibt immer die Grundarchitektur des Systems wieder.
Es existieren in der Regel nur $8 \cdot 2^n$ -breite Bussysteme.

Über den **Adressbus** wählt die CPU aus woher oder wohin sie Daten übertragen möchte. Die Breite des Adressbusses ist nicht an die Grundarchitektur des Systems gebunden. Z.B. hatten im Prinzip alle 8-Bit-CPUs immer schon einen 16-Bit-Adressbus. Ein kommerzielles System mit einem Adressbus mit weniger als 16 Bit Busbreite hat es nie gegeben.

Der **Steuerbus** ist eine Ansammlung von Steuerleitungen für z.B. Lesen Schreiben I/O-Ausgaben, Reset-Leitung, Interrupt-Leitungen usw. Er ist in jedem Computersystem unterschiedlich.

2.3.3 Arbeitsspeicher/Cache

Anfangs wurden Standard-Prozessoren noch mit nur „einigen“ MHz Takt betrieben. Der „langsame“ dynamische Speicher war als Arbeitsspeicher dafür gerade noch schnell genug. Deshalb wurde der Speicher in Mikrocomputern bis in die 90er-Jahre als preiswertes dynamisches RAM (siehe Kapitel 5.2.2) ausgelegt.

Als die Taktfrequenzen 20 MHz überstiegen haben, wurde das immer mehr zum Problem, weil die CPU immer mehr auf den langsameren Speicher warten musste.

Das hat man ab den ersten 32 Bit Systemen versucht zu verbessern, indem man zwischen CPU und Speicher einen schnelleren Zwischenspeicher eingefügt hat. Dieser „Cache-Zwischen-Speicher“ wurde grundsätzlich als Statisches-RAM (siehe Kapitel 5.2.1) ausgeführt. Welche Daten im Cache stehen, hat der Cache-Controller verwaltet. Im Prinzip ein eigener kleiner zusätzlicher Computer, der sich nur um die Speicherverwaltung kümmert.



Abbildung 26 80386-CPU + 80387-FPU + 82385-Cachecontroller

Der Cache-Kontroller versucht mittels statistischer Methoden immer dafür zu sorgen, dass sich das von der CPU benötigte Datum bereits im Cache befindet (Cache-Hit), bevor sie diese anfordert. Dies gelingt leider nicht immer, deshalb muss die CPU manchmal doch wieder auf den langsamen DRAM-Hauptspeicher warten (Cache-Miss).

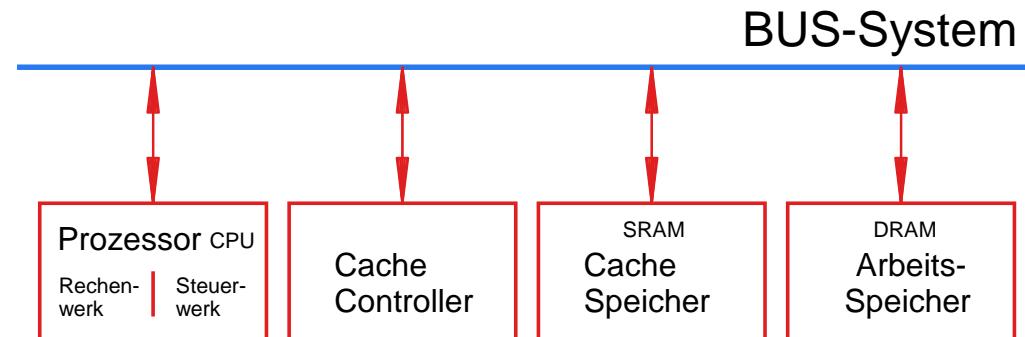


Abbildung 27 Cache

Die weiter ansteigenden Taktfrequenzen in der CPU-Entwicklung führten dazu, dass die CPU-Hersteller den Cache inklusive Cache-Controller mit in die CPU integriert haben (bei Intel ab i486). Allerdings waren hier anfangs nur sehr kleine Cache-Größen möglich <16 kByte. Deshalb wurde der externe Cache beibehalten und zum Second-Level-Cache umfunktioniert. Mittlerweile wurde auch dieser ab ca. 2000 wieder mit in die CPU integriert. Die neuesten Entwicklungen (ab 2018) integrieren einen Third-Level-Cache (L3-Cache) bis über 20 MByte direkt mit in die CPU.

2.3.4 CPU

2.3.4.1 Generischer CPU Aufbau

Die CPU (Central Processing Unit) ist das zentrale Element in einem Computer. Sie war früher das einzige aktive Element in einem Computer. Mittlerweile gibt es in modernen Computern mehrere auch unterschiedliche CPUs. Zum Beispiel in einem PC wird, wenn die CPU fehlt, trotzdem etwas passieren, da kleinere Mikrocontroller schon aktiv werden, bevor die eigentliche CPU gestartet wird.

Die CPU bedient alle Busse; Adressbus, Datenbus, Steuerbus und setzt verschiedene Vorgänge in Gang, welche teilweise dann von alleine weiterlaufen.

Der interne Aufbau einer CPU wird normalerweise mit Registern dargestellt. Ein Register ist eine Gruppierung von 1 Bit Speichern (Flip-Flops) mit 8-16-32 oder 64-Bit. Die Registerbreite einer CPU definiert auch gleichzeitig die Grundarchitektur eines Prozessors.

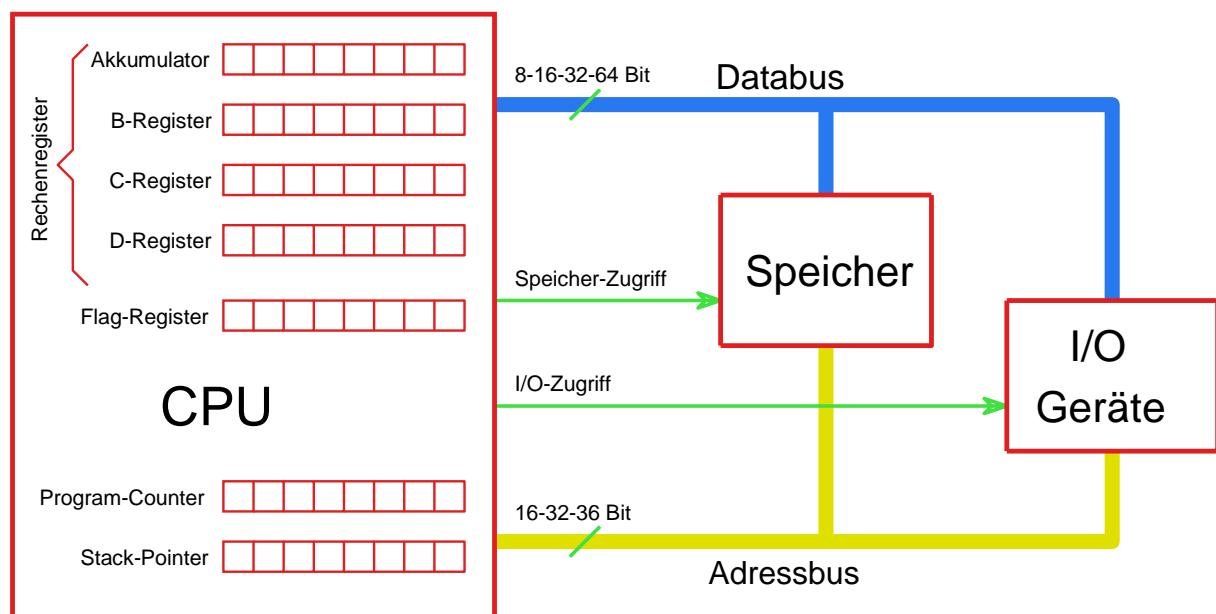


Abbildung 28 CPU intern

Bei früheren CPUs musste immer ein bestimmtes Register bei arithmetischen oder logischen Funktionen „dabei sein“. Dieses Register wurde als Akkumulator bezeichnet. Solche Prozessoren nannte man Akkumulatorrechner. Die Ergebnisse einer Operation landeten immer im Akkumulator. Heutzutage ist das bei modernen CPUs nur noch bei sehr speziellen Befehlen der Fall. Normalerweise können moderne CPUs heutzutage mit jedem Register alle Operationen durchführen.

2.3.4.2 Aktueller CPU Aufbau

Der langsame Speicherzugriff auf den DRAM-Hauptspeicher ist trotz mittlerweile 3 Cache-Ebenen immer noch ein Problem. Das wurde in den neunziger Jahren versucht zu verbessern, indem man bei 32-Bit-Prozessoren nach außen einen 64-Bit-Datenbus eingebaut hat. Dadurch wurde eine doppelt so schnelle Datenübertragung zwischen CPU und Hauptspeicher ermöglicht, da immer zwei 32-Bit-Worte auf einmal gelesen/geschrieben werden. Dies wurde zum ersten Mal von Intel mit dem Pentium 1993 eingeführt und im Prinzip bis heute beibehalten. Erst die neuen 64-Bit-Prozessoren haben dadurch keinen Vorteil mehr, da sie ja bereits intern einen 64 Bit breiten Datenbus haben.

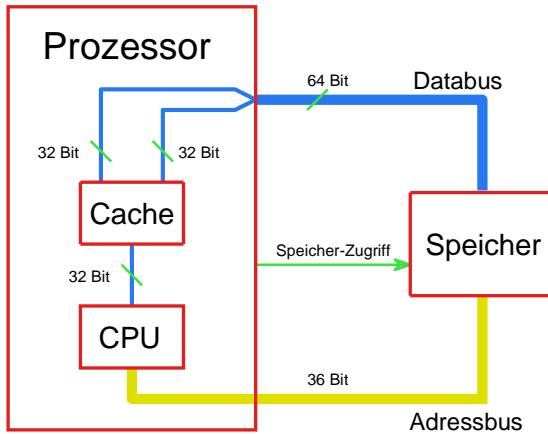


Abbildung 29 CPU 32/64 Bit Multiplexing

Bei modernen Prozessoren ab ca. 2010 gibt es die klassische Busstruktur nur noch innerhalb der CPU. Da die Datenübertragung zum Hauptspeicher immer noch zu langsam läuft, wurde das Speicherinterface noch einmal stark verändert.

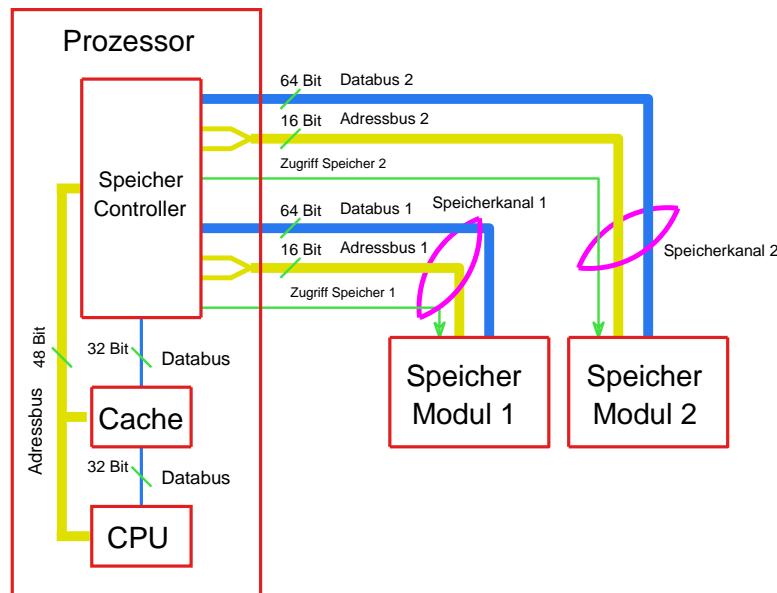


Abbildung 30 CPU mit 2 Speicherkanälen

Nach außen sind durch die mehrfachen Speicherkanäle jetzt mehrere Datenbusse und Adressbusse vorhanden. Die Adressleitungen sind jetzt gemultiplexed, da alle DRAM-Speicher ihre Adresse sowieso immer in 2 Teilen haben möchten (siehe Kapitel 5.2.2.1).

2.3.4.3 Fehlende Adressleitungen

Eine weitere Besonderheit bzgl. des Adressbusses gibt es ab den 32-Bit-Prozessoren. Sämtliche Speicheradressierung ist aus historischen Gründen immer byteweise organisiert. Da ein 32-Bit-Prozessor mit seinem 32-Bit-Datenbus immer gleich 4 Bytes auf einmal überträgt, ist es für die Speicherbausteine nicht möglich den gesamten Speicher einzeln byteweise anzusteuern.

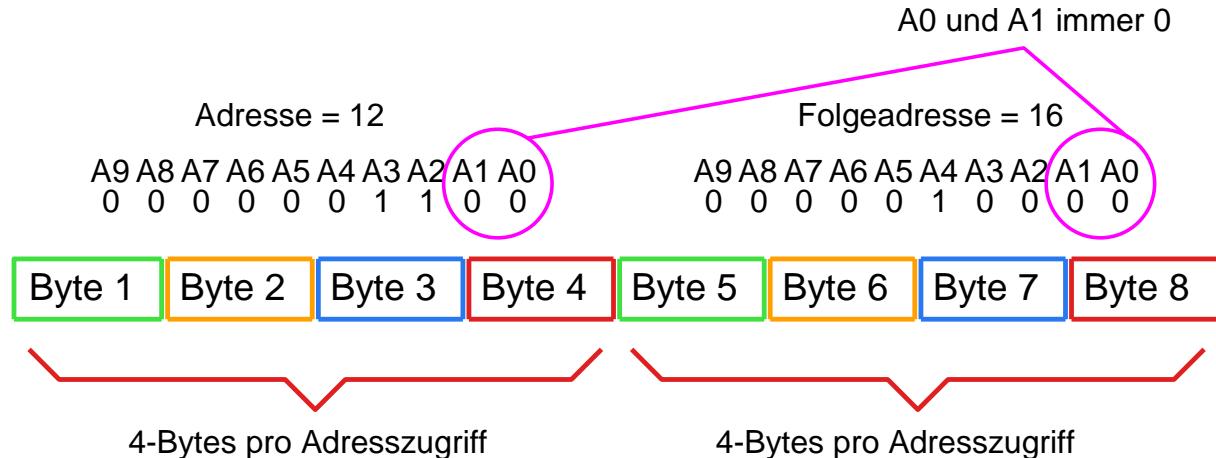


Abbildung 31 4 Bytes in einem 32-Bit-Wort

Das führt dazu, dass die beiden untersten Adressleitungen bei einem 32-Bit-Prozessor nicht mehr benötigt werden, da ja sowieso immer 3 Bytes übersprungen werden.

Intel386™ DX MICROPROCESSOR PIN DESCRIPTION TABLE		
Symbol	Type	Name and Function
CLK2	I	CLK2 provides the fundamental timing for the Intel386 DX.
D ₃₁ -D ₀	I/O	DATA BUS inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.
A ₃₁ -A ₂	O	ADDRESS BUS outputs physical memory or port I/O addresses.
BE0 # - BE3 #	O	BYTE ENABLES indicate which data bytes of the data bus take part in a bus cycle.
W/R #	O	WRITE/READ is a bus cycle definition pin that distinguishes write cycles from read cycles.
D/C #	O	DATA/CONTROL is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledgement, halt, and instruction fetch.

Abbildung 32 Auszug i386DX Datenblatt

Damit man trotzdem ein einzelnes Byte adressieren kann, gibt es jetzt 4 zusätzliche Busleitungen „Byte Enable“ BE0# - BE3#. Mit diesen Leitungen wird ausgewählt, welches der 4 Bytes gemeint ist.

Bei den Prozessoren mit 64-Bit-Datenbus wurde das konsequent weitergeführt. Hier gibt es auch die dritte Adressleitung A2 nicht mehr, dafür dann acht Byte Enable Leitungen BE0#-BE7#.

2.3.4.4 Takt und Timing

Bei den ersten Computern gab es genau einen Systemtakt. Dieser wurde von einem Taktoszillator meistens mit einem Quarzoszillator bereitgestellt.



Abbildung 33 Quarzoszillator

Als die CPU-Taktfrequenzen in den 1990ern ca. 50 MHz überstiegen, ist man dazu übergegangen, nur den Takt innerhalb der CPU zu erhöhen und schneller laufen zu lassen. Außerhalb hat sich der Takt von anfangs ca. 50 MHz auf einige 100 MHz weiterentwickelt.

Das liegt vor allem daran, dass auf einer Leiterplatte mit einer Ausdehnung von einigen 10 Zentimetern es schwierig wird, mit Taktfrequenzen zu hantieren, die einige 100 MHz übersteigen. Die Signalintegrität würde nicht mehr korrekt eingehalten. Bei einer Leiterbahn von 10 cm dauert es ca. 0,5 ns, bis das Signal auf der anderen Seite ankommt. Wenn die Leiterbahnen nicht alle gleich lang sind kommt es zu Fehlern (siehe Kapitel 2.5.3).

Um trotzdem schnellere CPUs nutzen zu können, wird nur innerhalb der CPUs mit einem höheren Takt gearbeitet. Die Taktvervielfachung wird meistens mittels einer PLL durchgeführt. Der Speichercontroller (bzw. Cache-Controller) in der CPU kümmert sich dann um die Entkopplung der beiden unterschiedlich schnellen Busse.

Die maximale Taktfrequenz von 200-300 MHz wurde schon ca. im Jahr 2000 erreicht und hat sich seitdem nicht mehr viel erhöht. Um trotzdem schneller Daten übertragen zu können, wurden zuerst während einer Taktperiode zwei anstatt eines Datenworts übertragen und später 4 Datenwörter pro Taktperiode.

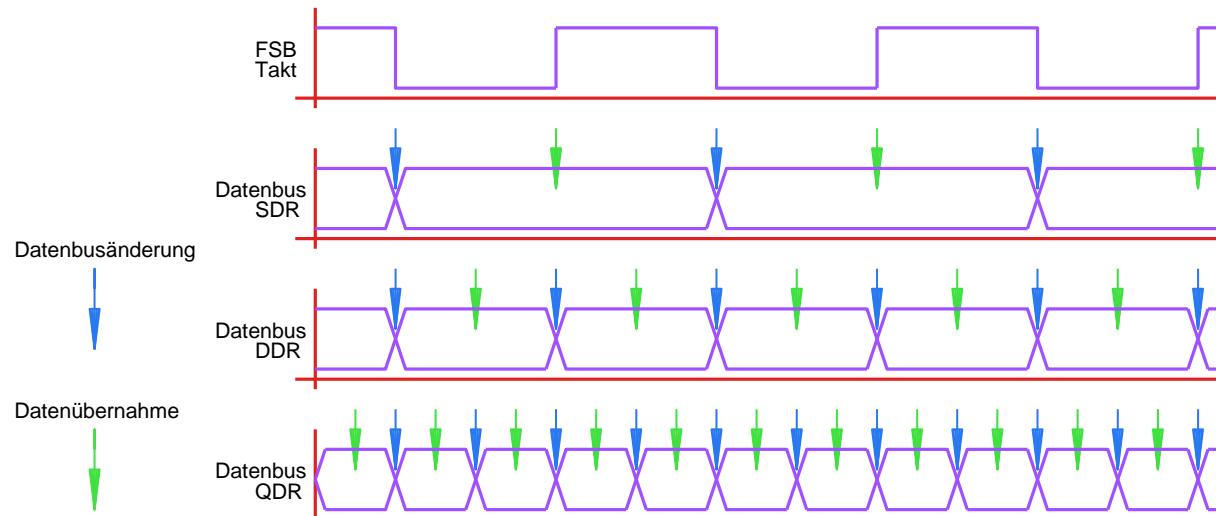


Abbildung 34 FSB SDR DDR QDR

2.3.5 FPU

Der Arithmetik-Prozessor wird heutzutage, im Gegensatz zum Embedded-Bereich, bei Mikrocomputern sehr oft mit in die CPU integriert. Früher musste der Arithmetik-Prozessor als Zubehör erworben werden. Bei Intel ist seit dem 486 die FPU mit in die CPU integriert. Wenn eine Software auf einem System ohne FPU Fließkomma-Befehle ausführen sollte, wurde dieses vom Betriebssystem emuliert. Die Arbeitgeschwindigkeit sank dabei etwa auf 1/100.

Die Arithmetik-Prozessoren vom Intel waren daran zu erkennen, dass die Typnummer auf 7 anstatt auf 6 endete.

Prozessor (CPU)	Koprozessor (FPU)
8086	8087
80286	80287
80386	80387
80486	---



Abbildung 35 Intel CPU 286 & FPU 287

Die FPU arbeitet im Gegensatz zur CPU als reines Rechenwerk und hat deshalb normalerweise keine Verbindung zum Adressbus. Alle Daten werden von der CPU über den Datenbus zur FPU transportiert.

Bei neueren FPUs, welche heutzutage direkt in die CPU mit integriert sind, kann die FPU auch teilweise selbstständig Daten transportieren, um die CPU hier zu entlasten.

Die FPU kann ausschließlich Fließkommazahlen verarbeiten. Die ersten 8087 FPUs von Intel arbeiteten intern schon im Jahr 1980 mit 80-Bit-Fließkommazahlen und konnten nach außen das 64-Bit-Zahlenformat verarbeiten.

Die grundsätzliche Arbeitsweise einer FPU funktioniert nach der **Umgekehrten Polnischen Notation** (UPN oder RPN) mit anfangs 8 Registern als Stack verwaltet (siehe Kapitel 3.5.1).

Bei späteren FPUs wurden mittels der Erweiterung MMX und SSE die Rechenregister breiter (128-Bit) und ihre Anzahl wurde später auf 16 verdoppelt (siehe Kapitel 3.3.9).

2.3.6 Massenspeicher

2.3.6.1 Historisch

In der Anfangszeit der Computer wurden Daten auf Lochkarten gespeichert. Diese wurden ab den 1960er-Jahren fast vollständig durch magnetische Speicher abgelöst. Das Grundprinzip ist teilweise heute noch in Gebrauch, da Lochkarten relativ fälschungssicher sind (Beispiel EG-Führerschein bis 1999).

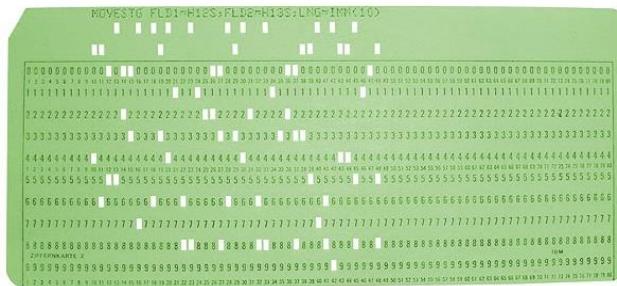


Abbildung 36 Lochkarte

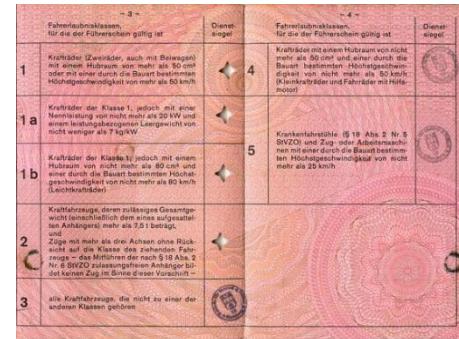


Abbildung 37 EG-Führerschein

2.3.6.2 Magnetisch

Band

Die ersten magnetischen Massenspeicher waren Magnetbänder. In der Anfangszeit wurden dafür echte Audio-Tonbänder zweckentfremdet. In der Homecomputer-Zeit (1980er) wurden dafür normale Audiokassettenrekorder mit Kompaktkassetten verwendet. Als Datenaufzeichnungsformat wurde meist frequenzmoduliert aufgezeichnet, weil das für die Audiotechnologie ideal passt.

Als Backup-Medium haben sich heutzutage immer noch die Magnetbänder als Streamer gehalten. Es gibt viele unterschiedliche Formate, die alle direkt digital auf Band aufzeichnen. Der stark sequentielle Zugriff ist bei Bandspeichern gleichzeitig Hauptvorteil und Nachteil. Die Zugriffszeit ist teilweise im Minutenbereich. Was bei einem Datenzugriff stört, sorgt gleichzeitig dafür, dass bei einem Virenbefall nicht alle Daten direkt erreichbar sind und zerstört werden können.

Aktuelle Streamer-Techniken basieren auf den LTO-Ultrium-Bandlaufwerken (**Linear Tape Open**), welche von einigen Herstellern gefertigt werden. Es gibt verschiedene Generationen von LTO1-LTO8, die teilweise rückwärtskompatibel sind.

Laufwerkstyp	Speichergröße
LTO-1	100 GB
LTO-2	200 GB
LTO-3	400 GB
LTO-4	800 GB
LTO-5	1.5 TB
LTO-6	2.5 TB
LTO-7	6 TB
LTO-8	12 TB



Abbildung 38 LTO-Band

Diskette

Die Diskette (auch Floppy-Disk) war in den 1980er-Jahren der mobile Massenspeicher für alle Computerarten. Man kann sie technologisch als Zwischenlösung zwischen Magnetbandspeicher und Magnetplatte ansehen.

Das Medium ähnelt vom mechanischen Verhalten dem Magnetband mit dem Vorteil des jetzt wahlfreien Zugriffs.

Bei Disketten besteht direkter Kontakt zwischen Schreiblesekopf und dem Medium. Disketten leiden deshalb unter starkem Verschleiß und geringer Lebensdauer.

Disketten haben sich über die Jahre hinweg zu immer kleinerer Größe und zu höherer Speicherdichte hin entwickelt.

Übliche Größen	Eingeführt	Speicher-Kapazität
8 Zoll	1970er	80 kByte bis 256 kByte
5,25 Zoll	1980er	360 kByte bis 1,2 MByte
3,5 Zoll	1980er	720 kByte bis 1,4 MByte

Abbildung 39 Diskettengrößen

Das Datenformat auf Disketten ist über verschiedene Computertypen hinweg nicht kompatibel.

Bei Disketten gab es Ende der 1990er-Jahre diverse Weiterentwicklungen, alle ungefähr im 100 MByte-Speicherbereich (ZIP-Drive, LS120). Mit dem Aufkommen von Flash-Speichern im USB-Stick-Format wurden ab ca. 2005 fast alle aufgegeben (Ausnahme Profi-Musiker Equipment).



Abbildung 40 5,25 Zoll und 2,5 Zoll Diskette

In den 1990ern gab es Streamerbandlaufwerke (Travan-QIC-Technologie), die sich hardwaremäßig mit gleichem Anschlussstecker wie ein Diskettenlaufwerk verhielten (Speicherkapazität 400 MByte).

Platte

Magnetplatten funktionierten prinzipiell genau wie Magnetbänder oder Disketten, mit dem Unterschied, dass der Schreib-/Lesekopf jetzt über der Platte schwebt. Der Luftwirbel, der durch die Rotation der Platte mit dem Bernoulli-Effekt entsteht, sorgt dafür, dass der Schreib-/Lesekopf die Platte im Datenbereich niemals berührt. Falls die Platte ausgeschaltet wird, und die Umdrehung stoppt, fährt der Schreib-/Lesekopf auf eine Landing-Zone, auf der keine Daten gespeichert sind.

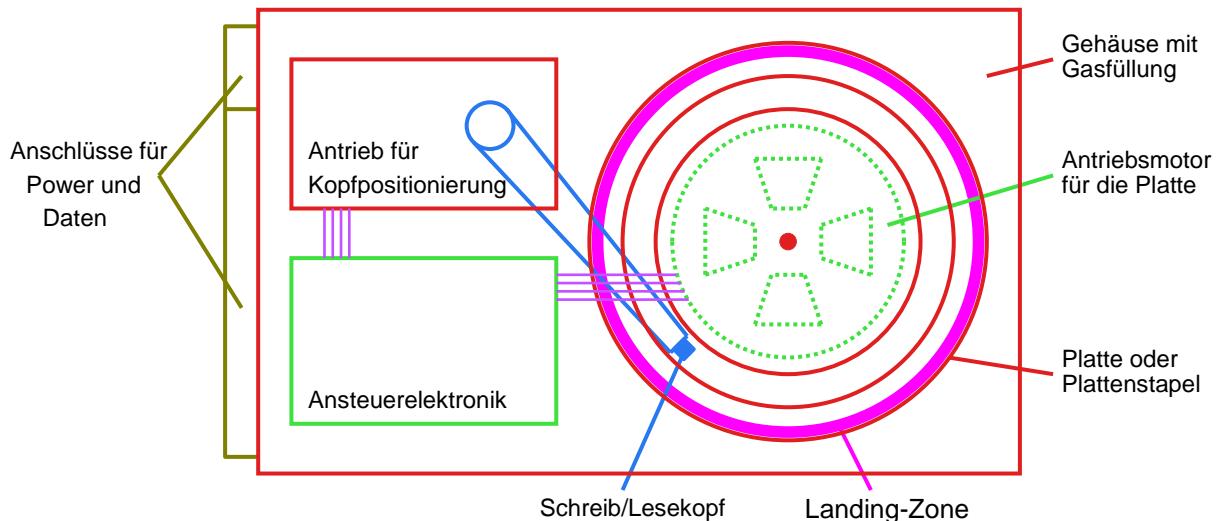


Abbildung 41 Aufbau Standard-Festplatte

Der wichtigste Vorteil liegt jetzt durch die hohe Rotationsrate in der sehr kurzen Zugriffszeit und der hohen Übertragungsrate. Die Hauptwartezzeit hängt hauptsächlich von der Umdrehungsgeschwindigkeit der Platte ab, weil der Lesekopf im Mittel eine halbe Umdrehung warten muss, bis die gesuchten Daten vorbeikommen.

Festplattenkapazitäten haben sich von anfangs einigen Megabytes auf heute über 10 Terabyte/Platte gesteigert. Die Schnittstellen für Festplatten haben ebenfalls eine lange Evolution hinter sich. Anfangs seriell (MFM), dann parallel (IDE/SCSI) mit 8 oder 16 Bit Busbreite wieder zurück zu seriell SATA und SAS (siehe Kapitel 6.2.2).

Der Innenraum einer Festplatte ist mit absolut staubfreiem Gas gefüllt. Ein Staubpartikel zwischen Lesekopf und Platte würde sofort zu einer Kollision von Kopf und Platte führen (Headcrash).

Bei der Gasfüllung sind manche Hersteller mittlerweile dazu übergegangen Helium anstatt Luft einzufüllen, weil die Gas-Reibung bei Helium entsprechend der kleineren Dichte geringer ist und zu weniger Erwärmung führt.

Die Speicherkapazitäten von Festplatten haben sich im Laufe der Jahre kontinuierlich weiterentwickelt. Bei genauerer Betrachtung fällt auf, dass die Entwicklung zwischen ca. 1990 und 1995 fast stehen geblieben ist. Das liegt daran, dass es nicht möglich war, den Lesekopf der Platte noch kleiner zu machen, und damit kleinen magnetisierten Zonen sicher getrennt voneinander lesen zu können.

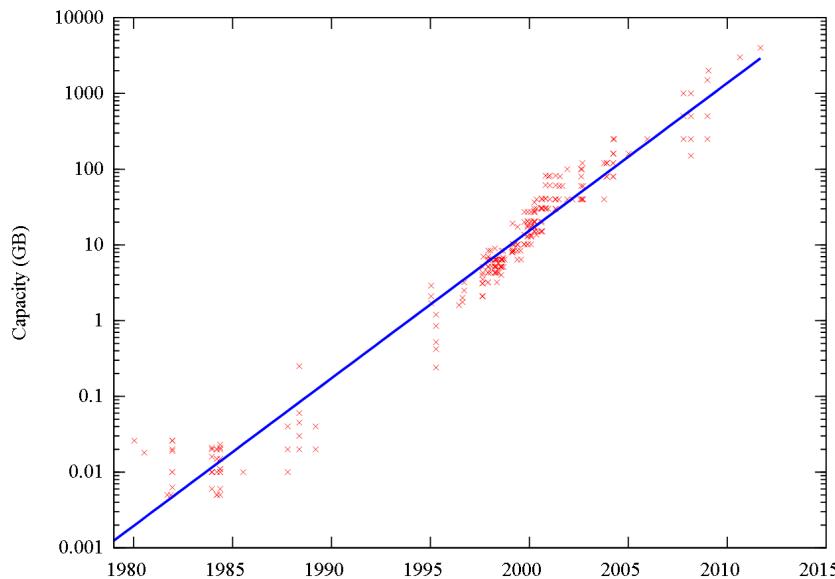


Abbildung 42 Festplattenentwicklung

Die erforderliche Physik, um hier weiterzukommen, wurde im Jahr 1988 entdeckt:

Der GMR-Effekt (Giant Magneto Resistance), Nobelpreis für Physik 2007. Hierbei handelt es sich um einen quantenmechanischen Effekt in ferromagnetischen Stoffen. Dieser Effekt wurde ausgenutzt, um einen wesentlich kleineren Lesekopf zu konstruieren. Der Schreibkopf blieb erst mal prinzipiell unverändert.

Es hat einige Jahre gedauert, bis diese Technologie bei Festplatten eingesetzt werden konnte. Ab dem Jahr 1995 ging die Entwicklung dann weiter, und die Speicherkapazitäten konnten weiter gesteigert werden.

Eine weitere Steigerung der Speicherkapazität wurde ca. 2008 durch eine Änderung der Magnetisierungsrichtung des Schreibkopfes erreicht.

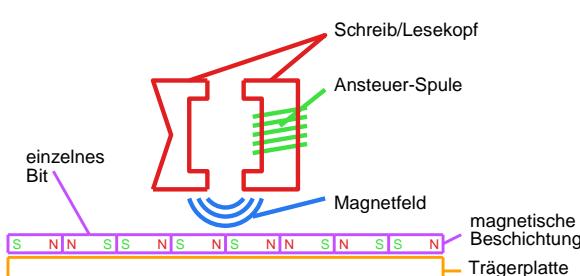


Abbildung 43 Longitudinale Magnetisierung

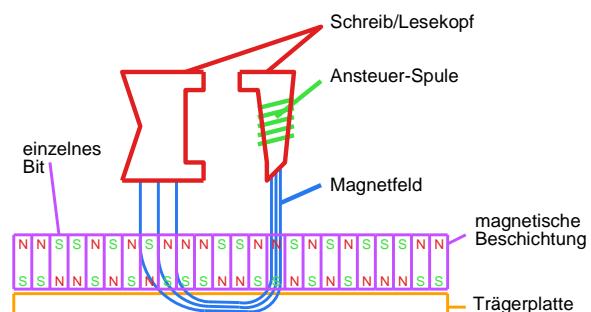


Abbildung 44 Perpendikuläre Magnetisierung

Um noch mehr magnetische Bits auf eine Platte zu bekommen, wurde ab ca. 2013 das SMR (Shingled Magnetic Recording) eingeführt. Bei der Magnetisierung von einzelnen Bits kann man die Ausdehnung des Magnetfeldes nicht beliebig klein halten. Deshalb wurde in der Vergangenheit immer ein Sicherheitsabstand zwischen den Nachbarspuren eingehalten.

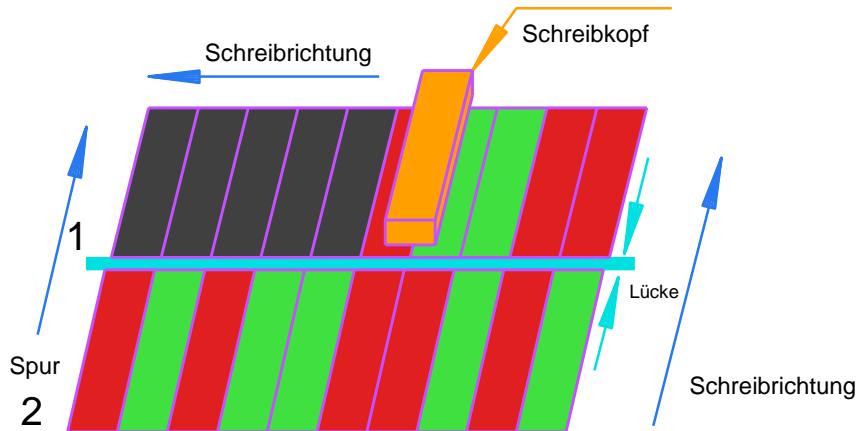


Abbildung 45 Herkömmliche Aufzeichnung

Dieser Abstand wurde bei Einführung von SMR bewusst unterschritten. Der Schreibkopf ist aus physikalischen Gründen größer als der Lesekopf. Beim SMR werden die vorhergeschriebenen Magnetzonen von der nachfolgenden Spur teilweise überschrieben. Wenn die zu lesenden magnetischen Zonen jetzt kleiner sind als vorher, macht das nichts aus, weil der jetzt wesentlich kleinere GMR-Lesekopf damit keine Probleme hat.

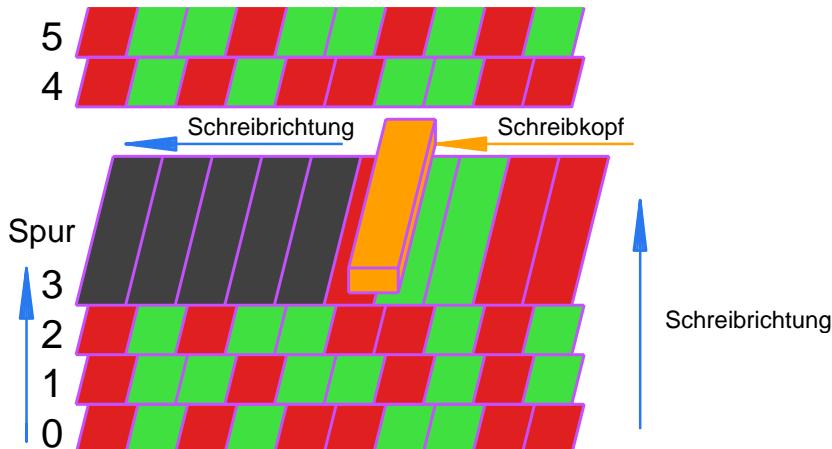


Abbildung 46 Shingled Magnetic Recording

Da die dahinterliegenden Spuren vollständig überschrieben werden, muss in gewissen Abständen eine Lücke freigelassen werden, damit nicht die ganze Platte überschrieben werden muss. Der Bereich bis zur nächsten Lücke muss jedoch vollständig neu geschrieben werden. Dieser zusätzliche Verwaltungsaufwand verlangsamt den Schreibvorgang etwas, da bei einer Änderung der gesamte Block bis zur nächsten Lücke zuerst ausgelesen werden muss und danach wieder komplett geschrieben wird. Aktuelle Festplatten bringen zur Zeit (2024) zwischen zwei Lücken ca. 40 MByte an Daten unter.

Ausblick auf die nächste Festplattengeneration:

Die unter Ausnutzung aller in der Vergangenheit eingeführten Technologien liefern zur Zeit (2024) Festplatten mit einer maximalen Größe von ca. 18-20 TeraByte. Um in Zukunft noch mehr speichern zu können, ist eine neue Technologie in Aussicht, die das ermöglichen soll.

Beim HAMR (**H**eat **A**sisted **M**agnetic **R**ecording) wird mit Hilfe eines Lasers die zu schreibende Stelle über die Curie-Temperatur des Materials erhitzt. Dadurch sinkt die für einen Schreibvorgang notwendige magnetische Feldstärke, und der Schreiberkopf kann dadurch noch kleiner gebaut werden.

Beim MAMR (**M**icrowave **A**sisted **M**agnetic **R**ecording) wird direkt beim Schreiberkopf mittels elektromagnetischer Mikrowellenstrahlung das Material sozusagen magnetisch aufgeweicht. Die Magnetisierung der Bits kann dann auch mit wesentlich weniger magnetischer Feldstärke erreicht werden. Auch dadurch kann der Schreiberkopf weiter verkleinert werden.

Kapazitäten über 20 TeraByte werden jedoch erst in einigen Jahren erwartet.

2.3.6.3 Solid State

Historisches

Der Begriff Solid-State ist historisch bedingt und wurde in der Übergangszeit von Röhren auf Halbleiter geprägt. Er hat sich bis heute gehalten als Bezeichnung für Bauteile, die auf Festkörperphysik basieren. Im Computerbereich ist damit üblicherweise ein Datenspeicher gemeint, der ohne Mechanik funktioniert.

Historisch angefangen haben damit die Ringkernspeicher, die auch heute noch in manchen militärischen Anwendungen immer noch in Gebrauch sind (Beispiel: Raketenkontrolle NASA, Space-Shuttle). Hauptvorteil hier ist das sehr robuste Verhalten gegenüber EMV-Störungen. Entfernt verwandt sind sie mit moderneren Speicherentwürfen; MRAMs und FRAMs (siehe Kapitel 5.3.5)

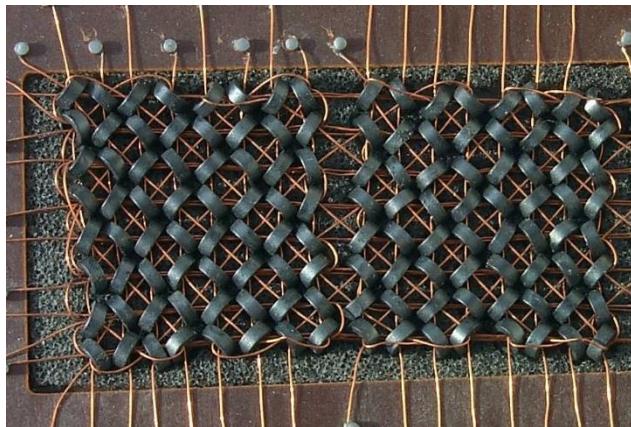


Abbildung 47 Ringkernspeicher

Flash

Im Gegensatz zum EEPROM (siehe Kapitel 5.3.3) sind bei Flash-Speicher (siehe Kapitel 5.3.4) die Speicherstellen nicht einzeln lösbar, sondern immer nur Blockweise. Die üblichen Blockgrößen liegen im Kilobyte-Bereich.

Um ein einziges Bit zu ändern, ist folgende Prozedur notwendig:

- kompletten Block vom Flash lesen und zwischenspeichern
- Flash-Block löschen
- Änderungen in gelesenem Block vornehmen
- kompletten geänderten Block in Flash zurück schreiben

Da sich die Löschprozedur über mehrere Millisekunden hinziehen kann, wird normalerweise ein bereits vorher gelöschter oder leerer Block für das Zurückschreiben benutzt.

SSDs

Die ersten Versionen von „Solid State Discs“, welche herkömmliche Festplatten vollständig ersetzen konnten, kamen nach dem Jahr 2000 auf den Markt. Es hat jedoch über 10 Jahre gedauert, bis die Preise für Standardcomputer in brauchbaren Regionen angekommen sind. Die Preise für SSD Speicher sind im Vergleich zu herkömmlichen Magnetplatten heute (2024) immer noch ca. Faktor 2 höher.

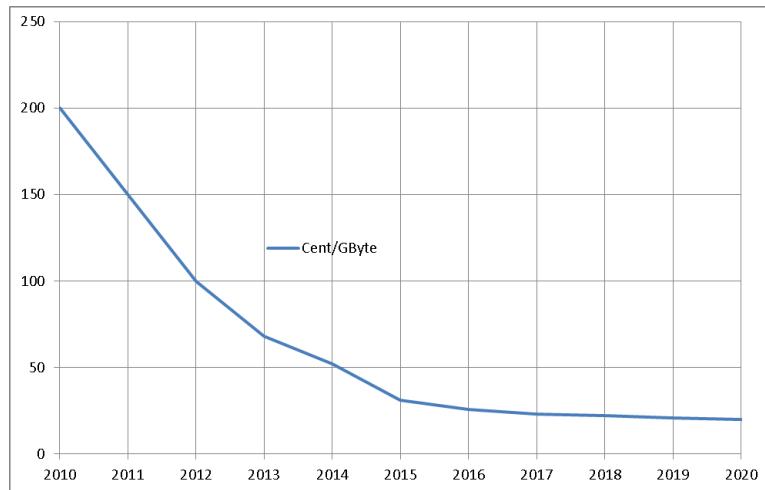


Abbildung 48 Preisentwicklung SSD

Die großen Vorteile von SSDs im Vergleich zu Magnetplatten sind folgende:

- deutlich kleinere Zugriffszeit
- höhere Datentransferrate
- erschütterungsunempfindlich
- geräuschlos

Bei den ersten Flashspeichern wurden anfangs nur 1 Bit pro Zelle gespeichert. Durch viele Weiterentwicklungen der Flashzellen werden aus Kostengründen zurzeit bis zu 4 Bit in einer Flash-Zelle gespeichert (siehe Kapitel 5.3.4). Der Datenerhalt ist stark temperaturabhängig und wird mit jedem zusätzlichen Bit immer schlechter.

Technologie	Max. Schreibvorgänge pro Zelle
SLC, 1 Bit/Zelle	100000
MLC, 2 Bit/Zelle	3000
TLC, 3 Bit/Zelle	300-1000
QLC, 4 Bit/Zelle	< 100 + Fehlerkorrektur

Abbildung 49 Aktuelle Daten (2018) von Flash-Speicherzellen

Wear Levelling

Um dem Verschleiß beim Speichern von SSDs insbesondere ab TLCs entgegenzuwirken, haben die Festplattenhersteller das Wear Levelling eingeführt. Dabei wird versucht, möglichst alle Speicherblöcke gleichmäßig oft zu löschen bzw. neu zu beschreiben. Das erfordert eine relativ aufwändige Verwaltung der Speicherstellen innerhalb des Speichermediums. Die Speicherdaten der Verwaltung selbst sind hierbei immer in extra SLC-Flashzellen gespeichert.

Es gibt trotzdem einige Hersteller von SSDs, die für professionelle bzw. industrielle Anwendungen immer noch reine (teure) SLC-SSDs herstellen.

Hybrid-SSDs

Um die Preise für schnelle Festplatten kleinzukriegen, kombinieren manche Hersteller die Vorteile beider Technologien (Magnet=preisgünstig, Solid State=schnell). In einem Festplattengehäuse wird eine herkömmliche Magnetplatte verbaut mit einem Cache-Zwischenspeicher auf SSD-Basis.

Beim Lesen versucht der integrierte Controller mittels statistischer Methoden immer dafür zu sorgen, dass sich die angeforderten Daten bereits im SSD-Speicherbereich befinden, bevor sie angefordert werden.

Beim Schreiben landen die Daten zuerst im schnellen SSD-Speicher und werden dann erst danach langsam auf die Magnetplatte gespeichert.

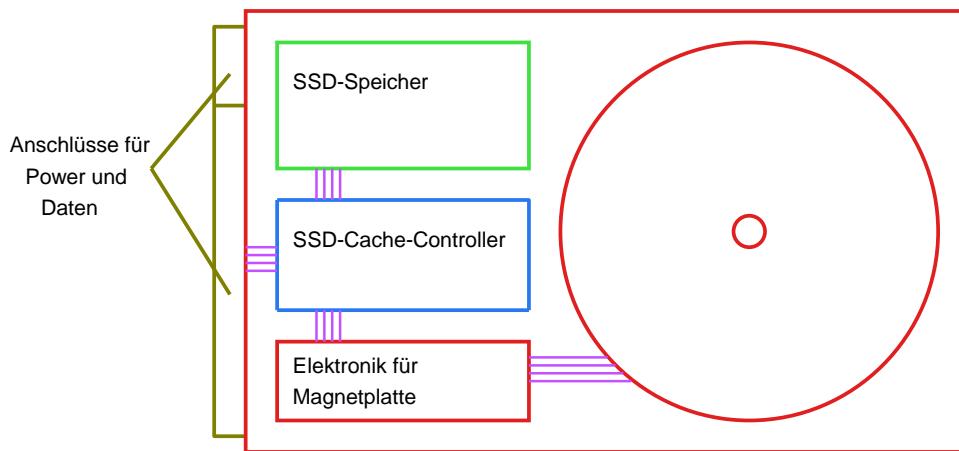


Abbildung 50 Hybrid-Festplatte

2.3.6.4 Optisch

CD-ROM

Die Daten auf einer CD werden optisch auf einer Polycarbonatscheibe gespeichert. Diese Art der Datenspeicherung wurde einige Jahre vorher mit der analogen Bildplatte (Laserdisc) seit 1970 erfolgreich erprobt.

Die Speicherung erfolgt durch unterschiedlich lange Vertiefungen (Pits und Lands) im Kunststoff. Die Informationsschicht befindet sich auf der Oberseite, die durch eine Aluminiumschicht und einen Schutzlack geschützt wird. Gelesen wird von der Unterseite durch die 1,2 mm dicke Polycarbonatscheibe hindurch.

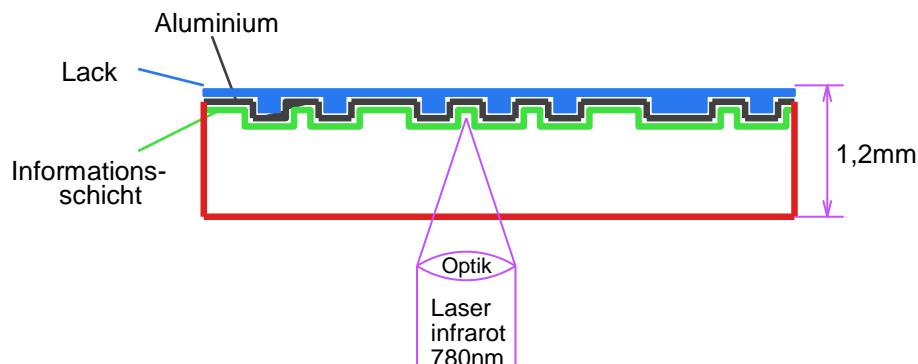


Abbildung 51 CD-ROM-Aufbau

Ursprünglich als Medium für Digital-Audio geplant, wurde die CD-ROM ca. 10 Jahre danach als Computermassenspeicher eingeführt.

Die Speicherkapazität liegt je nach Fehlerkorrekturmethode bei ca. 700-900MBYTE.

DVD

Um mehr Daten speichern zu können, wurde die optische CD weiterentwickelt, indem man zuerst die Größe der Vertiefungen ungefähr halbiert hat. Durch weitere Verbesserungen, insbesondere wurde der Leseabstand halbiert und die Laserwellenlänge verkleinert (jetzt Roter sichtbarer Laser anstatt Infrarot). Dadurch ergibt sich eine Speicherkapazität von 4,7 GByte. Als zusätzliche Möglichkeit wurden zwei zweilagige Discs mit halber Dicke zusammengeklebt, was im Vollausbau mit 4-Lagen dann ~17 GB ergibt.

Die Informationsschicht befindet sich jetzt in der Mitte zwischen den Scheiben, wodurch sie noch zusätzlich vor Beschädigungen geschützt ist.

Der Name DVD leitet sich ab aus **Digital Versatile Disk**.

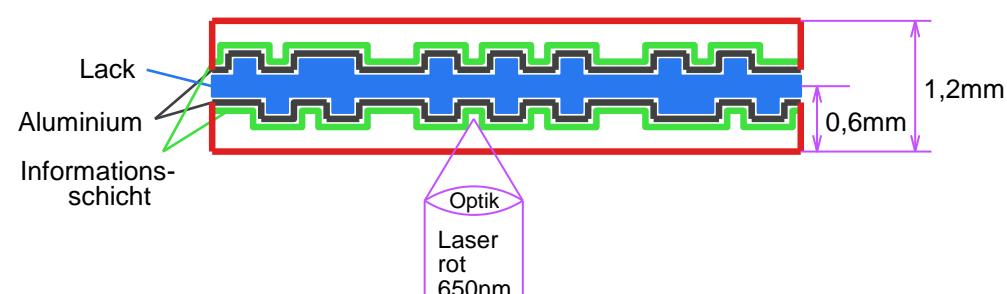


Abbildung 52 DVD Aufbau

BLU-RAY

Die Weiterentwicklung von der DVD zur Blu-Ray-Disk war etwas holpriger als der Übergang von der CD zur DVD. Es gab zwei Technologien, um mehr Daten als bei einer DVD zu speichern. Die eine bereits wieder ausgestorbene Idee war, dieselbe Technologie wie bei der DVD zu nutzen und einfach die Pits und Lands kleiner zu machen. Die Datenspeicherschicht war wie bei der DVD in der Mitte zwischen zwei 0,6 mm dicken Polycarbonatscheiben. Diese Technologie nannte sich HD-DVD und hat den Marketingkrieg ca. 2005-2008 gegen die BluRay-Disk verloren und wurde aufgegeben.

Bei der BluRay wurde der Leseabstand noch mal auf fast die Hälfte verringert und die Daten befinden sich nun auf der Unterseite der Polycarbonatscheibe. Die Speicherkapazität liegt bei 2-lagigen BluRay-CDs bei 50 GByte

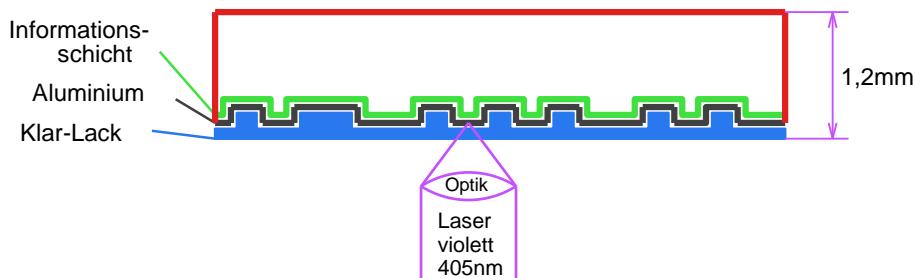


Abbildung 53 BluRay Aufbau

Ein großes Problem bei der Entwicklung der BluRay-Disk war, dass zwischen der Informationsschicht und der Laseroptik fast kein Schutzabstand mehr ist. Die Beständigkeit des Klarlacks auf der Unterseite gegenüber Kratzern, Schmutz und Fett, war eine der größten technischen Herausforderungen.

Bei der BluRay-Disk wurde auch wieder die Wellenlänge des Lasers verkürzt, und zwar auf 405 nm. Damit müsste sie eigentlich UV oder Violett-Ray-Disk heißen, da die Farbe des Lichtes mit 405 nm keineswegs blau aussieht, sondern eher violett, falls man überhaupt etwas sehen kann, denn der sichtbare Bereich des Lichtes endet beim menschlichen Auge individuell unterschiedlich bei ca. 400nm.

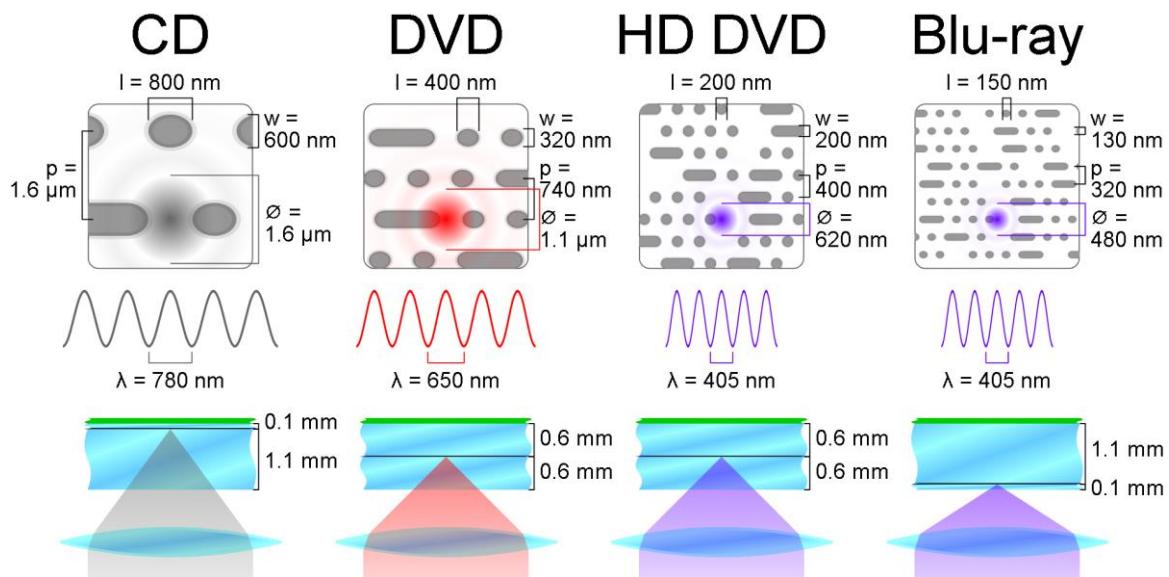


Abbildung 54 Vergleich CD DVD Blu-Ray

2.3.7 Redundanz

2.3.7.1 Allgemeines

Um Daten ausfallsicher zu speichern, kann es sinnvoll sein, sie mehrfach auf mehreren Speichermedien abzuspeichern. Dafür gibt es mehrere Möglichkeiten.

Das kann entweder als Back-up rhythmisch erfolgen oder dauerhaft in Echtzeit, wobei die mehrfache Echtzeitspeicherung das Back-up nicht ersetzen kann, da bei einer Zerstörung der Daten z.B. durch einen Virus die Kopie gleichzeitig auch zerstört wird.

Die Art und Weise, wie man mehrere Festplatten in Echtzeit als Verbund organisiert, wird in der Regel RAID (**R**edundant **A**rray of **I**ndependent **D**isks) genannt.

Die Organisation der Festplatte kann auf zwei verschiedene Arten erfolgen:

- Software RAID
- Hardware RAID

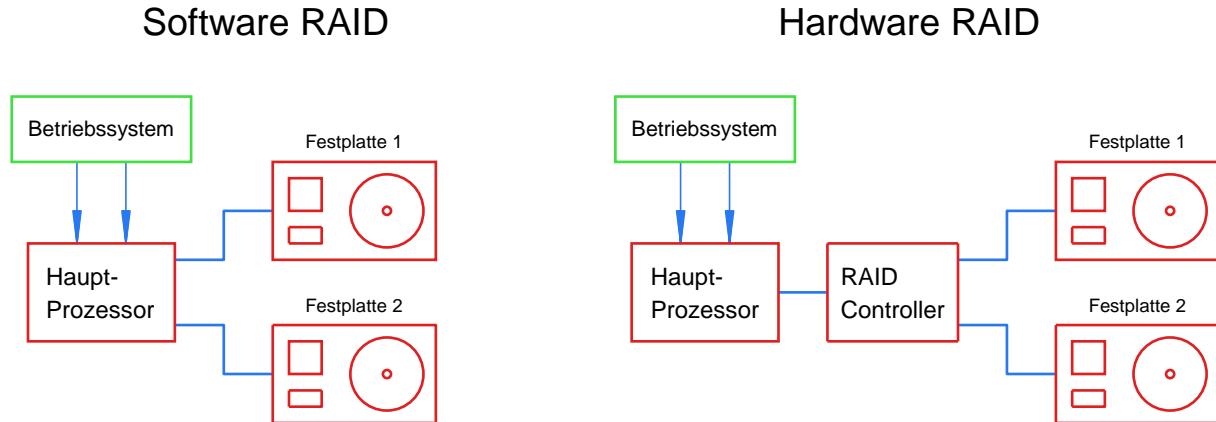


Abbildung 55 Software & Hardware RAID

Bei einer Software-RAID-Lösung wird die Organisation der Daten auf den angeschlossenen Festplatten vom Betriebssystem übernommen. Bei der Hardware-RAID-Lösung kriegt das Betriebssystem im Idealfall nichts davon mit, dass mehrere Festplatten im Spiel sind. Der Nachteil der Softwarelösung ist die stärkere Belastung des Systemprozessors bei Speichervorgängen.

Bei Einsatz von lauter gleichen Festplatten desselben Herstellers sollte man noch beachten, dass diese mit sehr hoher Wahrscheinlichkeit ähnliche Lebensdauern haben und relativ gleichzeitig ausfallen können.

Es gibt verschiedene RAID-Level, wie man die Festplatten verwalten kann:

2.3.7.2 RAID 1

Für RAID 1 benötigt man mindestens 2 identische Festplatten. Alle Daten werden synchron auf beiden Festplatten gespeichert. Beim Speichern gibt es keinen Geschwindigkeitsvorteil. Beim Lesen sinkt die Zugriffszeit etwas, da die Wahrscheinlichkeit größer wird, dass die gesuchten Daten früher unter einem der beiden Schreib-/Leseköpfe vorbei kommen. Als Nachteil kann man sagen, dass man immer doppelt so viel Speicherplatz benötigt. Beim Ausfall einer Festplatte sind alle Daten noch auf der anderen Platte vorhanden.

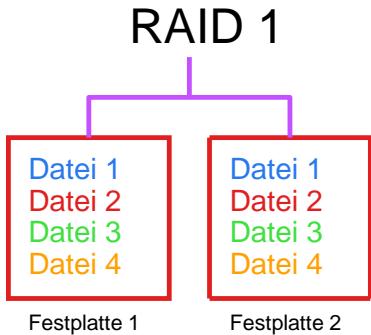


Abbildung 56 RAID 1

2.3.7.3 RAID 5

Bei RAID 5 werden die Daten auf alle Festplatten gleichmäßig verteilt. Es wird jedoch für jeden Datenblock eine Paritätsinformation zusätzlich gespeichert. Mit der Paritätsinformation kann bei einem Ausfall einer Festplatte der Inhalt zurückgerechnet werden. Man verliert bei einem RAID-5-Verbund nur den Speicherplatz einer Platte. Die Berechnung der Paritätsdaten erfolgt durch eine XOR-Verknüpfung. Da die Paritätsinformationen bei Lesen nicht benötigt werden, ist der RAID-5-Verbund schneller, da alle Platten parallel arbeiten.

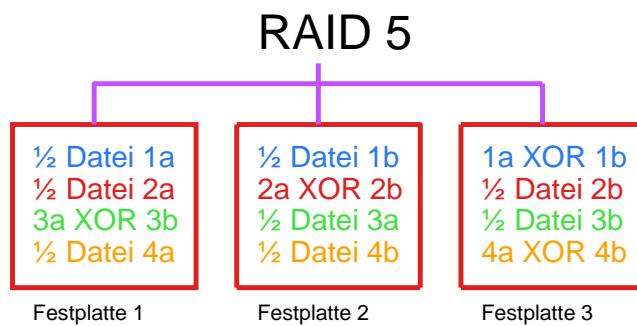


Abbildung 57 RAID 5

2.3.7.4 RAID 0

RAID 0 enthält eigentlich keine Redundanz und funktioniert in Prinzip wie RAID 5 nur ohne zusätzliche XOR-Daten. Bei einem Ausfall einer Platte sind alle Daten verloren. Vorteil ist die doppelte Schreib- und Lesegeschwindigkeit.

2.3.8 Power und Reset

2.3.8.1 Grundlegendes

Eine CPU kann nur korrekt arbeiten, wenn sich die Höhe der Versorgungsspannung in den für sie definierten Grenzen befindet. Wenn die Grenzen über- oder unterschritten werden, können seltsame Dinge passieren. Damit die CPU nicht bei fehlerhafter Versorgung startet, muss sichergestellt werden, dass die Versorgung vorher O.K. ist. Die Sicherstellung einer sauberen Versorgung kann über mehrere Methoden geschehen.

2.3.8.2 Power Good

Viele Computernetzteile verfügen über eine eigene Überwachungselektronik und melden dem System über eine eigene Leitung, dass die Versorgungsspannung jetzt bereit ist. Zum Beispiel bei ATX-Netzteilen wird diese Leitung Power-Good oder Power OK genannt.

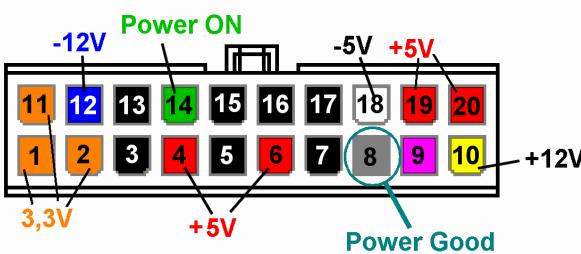


Abbildung 58 ATX-Power

2.3.8.3 Crowbar

Eine zusätzliche Sicherheitsmaßnahme bei manchen Netzteilen ist eine sogenannte Crowbar. Hierbei handelt es sich um eine Schaltung, die bei einer Überspannung einen Kurzschluss auslöst und somit die Sicherung zerstört.

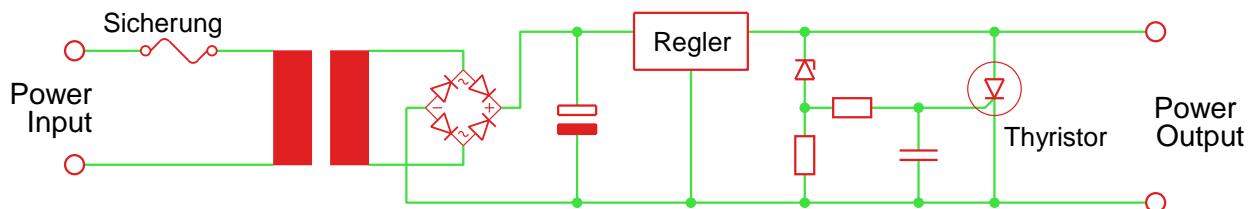


Abbildung 59 Crowbar

Diese Maßnahme dient dazu, größere Folgeschäden im Computersystem durch Überspannung zu verhindern.

2.3.8.4 Supply Supervisor

In vielen Computerarchitekturen erfolgt die Sicherstellung der Versorgung durch einen extra Baustein: der Supply-Supervisor. Dieser ist direkt mit der Reset-Leitung des Prozessors verbunden und überwacht die Versorgungsspannung. Falls die Grenzen über- oder unterschritten werden, wird der Prozessor solange über die Reset-Leitung angehalten, bis die Grenzen wieder eingehalten werden. Übliche Bausteine warten dann noch etwa eine halbe Sekunde, bis die Resetleitung wieder freigegeben wird, und das System wieder startet.

Manche CPUs reagieren sehr empfindlich auf eine langsam abfallende Versorgungsspannung. Bei CPUs mit Flash-Programmspeicher kann es vorkommen, dass sich das Programm bei langsam abfallender Versorgung selbst löscht -> Brown-Out.

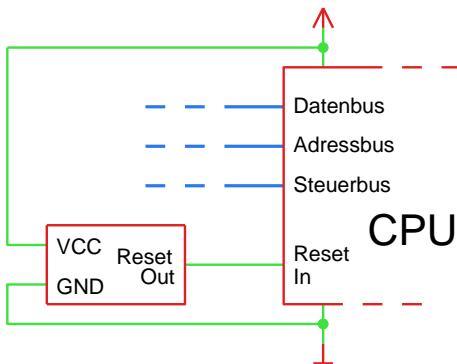


Abbildung 60 Reset

2.3.8.5 Watchdog

Bei manchen Computersystemen ist es sehr wichtig, dass es nach einem Softwareabsturz kontrolliert weitergeht. Dann ist die Verwendung eines Watchdog-Bausteins hilfreich. Dieser wartet auf einer bestimmten Adresse in einer vorgegebenen Zeit auf ein Triggersignal, wenn dieses ausbleibt, löst er einen Reset aus.

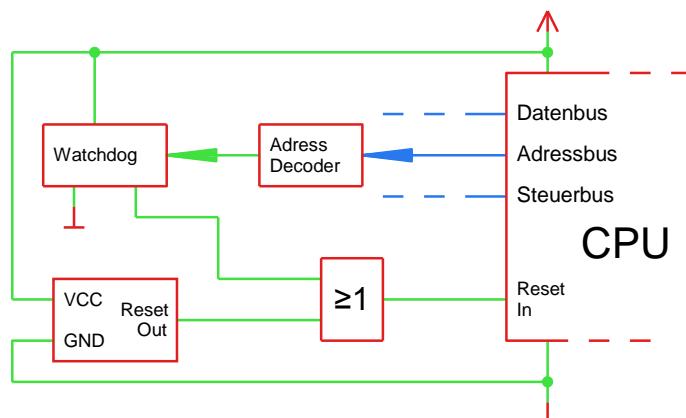


Abbildung 61 Watchdog

Bei vielen Mikrocontrollern ist oft ein Watchdog gleich mit eingebaut, der mittels erstmaligen Zugriff auf „seine“ Adresse aktiviert wird.

2.4 Architekturarten

2.4.1 Von-Neumann

Von-Neumann erläuterte bereits 1945, die nach ihm benannte Computerarchitektur. Fast alle modernen Computer sind nach der Von-Neumann-Architektur aufgebaut. Der erste Computer, der prinzipiell eine Von-Neumann-Struktur hatte, war schon 1936 der vollmechanische Z1 von Konrad Zuse.

Das Grundkonzept beruht darauf, dass es nur ein einziges Bussystem gibt, welches für alle möglichen Arten von Daten benutzt wird. Dieses können Daten sein, welches ein Programm für seine Verarbeitung benötigt, als auch die Programmbefehle selbst. Das hat den Nachteil, dass der Bus und auch der Speicher abwechselnd für Daten und Programme genutzt werden. Der große Vorteil ist, dass man nur eine „Sorte“ Arbeitsspeicher benötigt, der für alles zuständig ist.

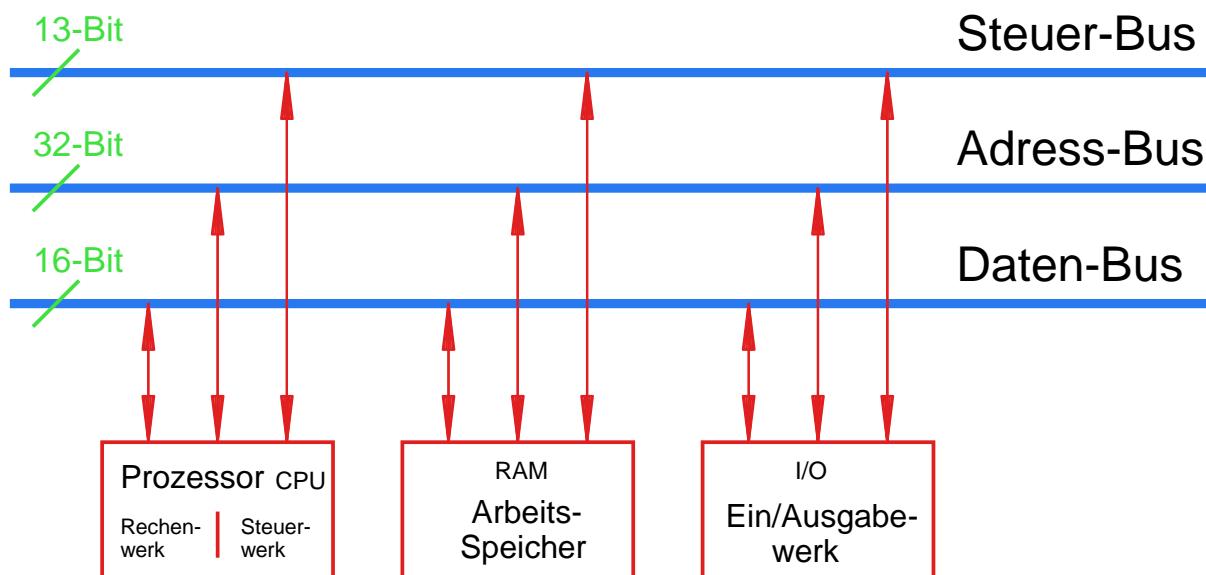


Abbildung 62 Von-Neumann

Als sich in der Speichertechnologie in den 90er-Jahren die Caches ausgebreitet haben (siehe Kapitel 2.3.3), wurde die Von-Neumann-Architektur zum Problem, weil sich Daten und Programmcode ständig gegenseitig aus dem Cache Speicher rausgeworfen haben (Cache-Trashing). Dieser Umstand wurde teilweise durch die Einführung von zwei getrennten Caches für Daten und Programmcode gelöst (bei Intel ab ca. 1993 mit dem Pentium). Diese Vorgehensweise führt innerhalb der CPU vom Prinzip her dann eher zu einer Harvard-Struktur (siehe Kapitel 2.4.2).

Ein großer Nachteil der Von-Neumann-Architektur ist die Möglichkeit, dass eine Software ihren eigenen Programmcode verändern kann, was für bösartige Software eine Möglichkeit darstellt, sich auszubreiten.

2.4.2 Harvard

Die Harvard Architektur wurde von IBM und der Harvard-Universität im Jahr 1944 entwickelt. Das wichtigste Merkmal besteht darin, dass das System physikalisch getrennte Speicher und Busse für Programmcode und Daten besitzt.

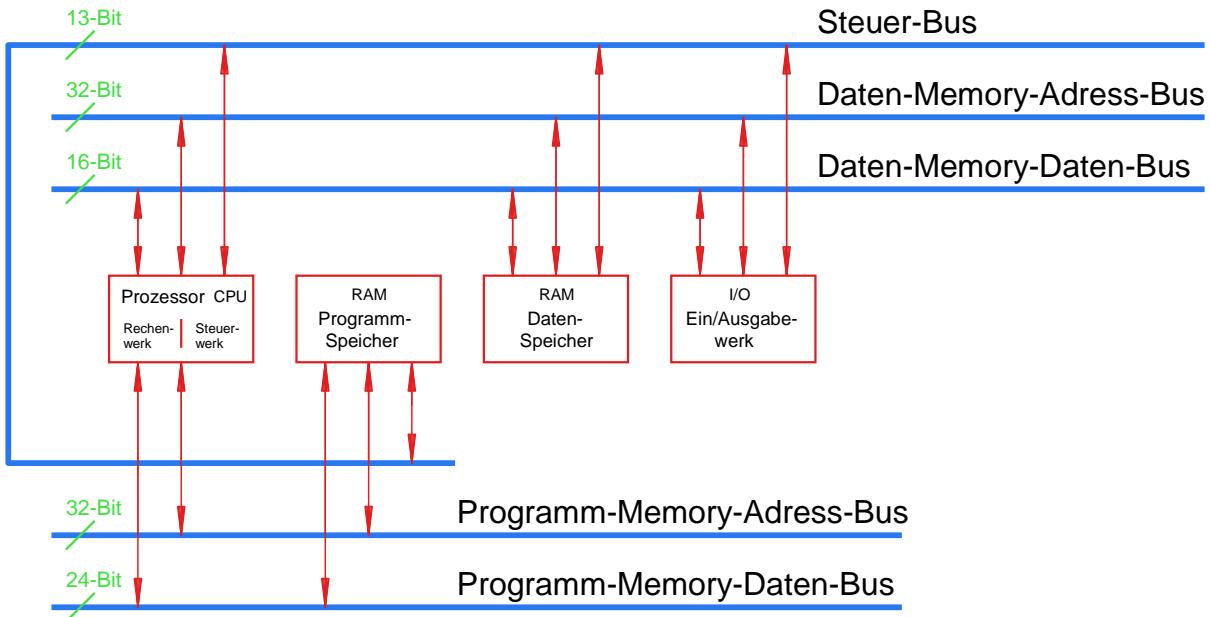


Abbildung 63 Full Harvard

Dadurch entstehen in der Hauptsache zwei Vorteile:

1. Der Zugriff auf den Speicher bei der Befehlsabarbeitung geht theoretisch doppelt so schnell.
2. Es ist nahezu unmöglich, dass ein laufendes Programm seinen eigenen Programmcode überschreibt.

Durch das Hemmnis des Überschreibens ist es auch nicht möglich, dass eine Softwaremanipulation durch ein bösartiges Programm sich in den Programmspeicher ausbreiten kann.

Der Programmspeicher ist bei normaler Programmausführung als Nur-Lese-Speicher ansprechbar und ist bei Embedded Systemen oft sogar in einem ROM-Speicher abgelegt.

Es gibt auch Harvard-Strukturen, welche das Harvard-Prinzip nur teilweise ausgeführt haben, was sich dann oft Modified-Harvard nennt. Auch die Caches in modernen Prozessoren sind oft Harvard-mäßig ausgelegt, um das gefürchtete Cache-Trashing (Daten und Programmcode werfen sich ständig gegenseitig aus dem Cache Speicher raus) zu verhindern.

2.5 PC-Bussysteme

2.5.1 ISA

Der ISA-BUS (Industry Standard Architecture) war das erste firmenübergreifende BUS-System für Standard PCs und wurde von IBM in den 1980er-Jahren entwickelt. Zuerst als XT-BUS mit 8-Bit-Datenbusbreite und 4,7 MHz und einige Jahre später mit einer Erweiterung auf 16 Bit mit einer Taktfrequenz von 8,33 MHz (Übertragungsrate 16,6 MByte/s). Beim ISA-Bus ist im Prinzip der gesamte Systembus des Computers direkt auf den ISA-Stecker geführt.

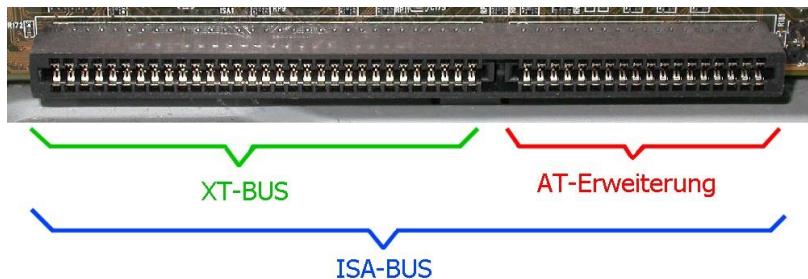


Abbildung 64 XT-AT-ISA-BUS

Später gab es noch einen Entwurf den ISA-BUS auf 32 Bit (EISA) zu erweitern, was sich jedoch nicht durchgesetzt hat.

Es werden immer noch Rechner für Industrieanwendungen mit einem ISA-Slot-System hergestellt, um bereits vorhandene (teure) ISA-Karten weiterverwenden zu können. Hierbei handelt es sich jedoch um Rechner mit einer zusätzlichen ISA-Hardware-Emulation, da es keinen Sinn macht, moderne Systeme mit nur ~8 MHz laufen zu lassen.

2.5.2 PCI

Mitte der 90er-Jahre wurde der PCI-BUS (Peripheral Component Interconnect) von einem Zusammenschluss einiger Firmen entworfen. Der Bus hat jetzt keine direkte Verbindung mehr zur CPU, sondern wird über den Chipsatz des Prozessors mit der CPU verbunden. Beim PCI-BUS sind Daten und Adressleitungen nicht getrennt herausgeführt, sondern sie werden gemultiplexed. Das liegt auch daran, dass ab Einführung des Burst-Modus (siehe Kapitel 4.4) für Busse die Adresse nicht jedes Mal neu übertragen werden muss. Adresse und Daten haben beide 32-Bit, die Taktfrequenz beträgt 33 MHz und erreicht somit eine Übertragungsrate von maximal 133 Mbyte/s im Burst-Modus.

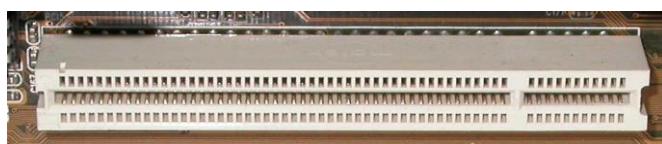


Abbildung 65 PCI-Bus

Vom parallelen PCI-Bus gab es später auch eine 64-Bit-Variante (PCI-X) mit fast doppelt so großem Stecker. Diese wurde in der Hauptsache für Serverboards verwendet und hat sich auf dem Massenmarkt nicht durchgesetzt. Hier wurde auch die Taktfrequenz erhöht und man erreichte mittels QDR in der letzten Ausbaustufe mit 1,5 V Signalpegel eine Übertragungsrate bis zu 4,2 GByte/s.

2.5.3 PCI-Express

Bei der Entwicklung paralleler Bussysteme entsteht das Problem, dass alle Signale auf Leitungen zwischen allen Geräten gleich lang unterwegs sein müssen (Flugzeit von Signalen auf Leiterplatten ca. 5 ns/m).

Da dies bei normalem Leiterplattenentwurf nicht möglich ist, werden manchmal Leiterbahnen künstlich mittels Mäandern verlängert.

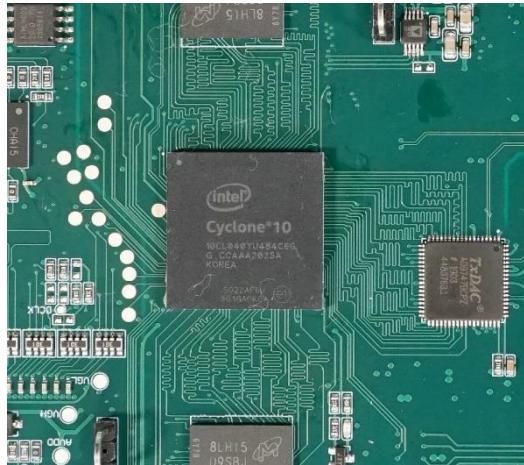


Abbildung 66 Mäander

Davon wurde in der Vergangenheit häufig Gebrauch gemacht. Wenn man den Datendurchsatz noch weiter erhöhen möchte, stößt man jedoch mit dieser Methode an gewisse physikalische Grenzen (Signallaufzeiten sind auf Innenlagen und Außenlagen von Multilayerleiterplatten unterschiedlich). Diese Grenzen waren mit dem parallelen PCI-Bus fast schon alle erreicht. Eine weitere Steigerung des Datendurchsatzes konnte nur erreicht werden, wenn man das parallele Konzept vollständig aufgibt und zu einer rein seriellen Übertragung wechselt. Zusätzlich wurde jede Leitung als differenzielles Paar doppelt ausgeführt, um eine störsichere Übertragung zu verbessern (siehe Kapitel 6.3.1.7).

Beim PCI-Express-Bus (ab ca. 2003) wurde der eigentliche BUS durch eine serielle Punkt-zu-Punkt-Verbindung (Lane) ersetzt. Die Taktfrequenz wurde (bei PCIe-1) mit ca. 2,5 GHz so hoch gewählt, dass ungefähr eine Verdopplung der Übertragungsrate von 32-Bit-PCI herauskommt. Der serielle Takt wird beim PCIe nicht mitübertragen, sondern wird aus dem Nutzdatensignal mittels PLL wieder zurückgewonnen (siehe Kapitel 6.3.1.8).

Des Weiteren können mehrere „Lanes“ zusammengeschaltet werden, um noch mehr Daten gleichzeitig zu übertragen (bis zu 16 Lanes).

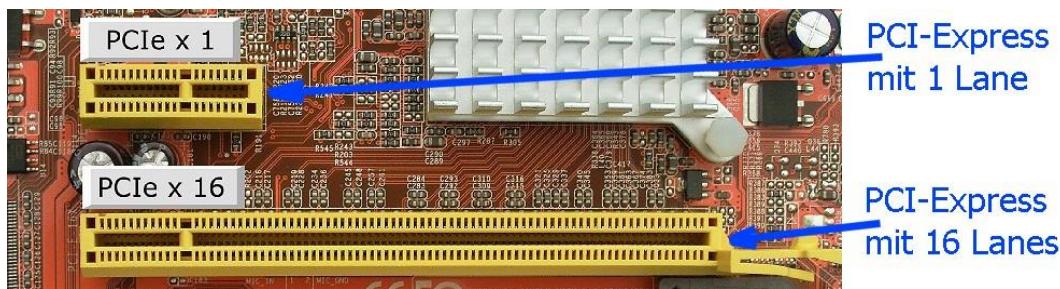


Abbildung 67 PCI-Express-Bus

Auch mit mehreren Lanes ergibt sich daraus keine neue parallele Übertragung. Die Signale der unterschiedlichen Lanes treffen gewöhnlicherweise nicht gleichzeitig ein und müssen auf der Empfängerseite wieder korrekt zusammengesetzt werden.

Der PCIe ist für die Software vollkommen unsichtbar, da die parallel-seriell-parallel-Wandlung direkt von der Hardware (im Chipsatz und auf der Erweiterungskarte) vollkommen transparent übernommen wird.

Das kann man auch daran erkennen, dass es einfache Umsetzerkarten zu kaufen gibt, die von PCI-Express nach PCI-Parallel konvertieren können. Diese können ohne Treiberunterstützung mit allen Betriebssystemen funktionieren.

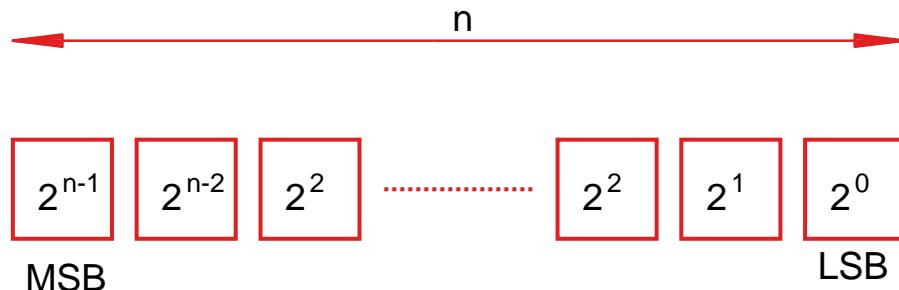


Abbildung 68 PCI-Express PCI-Parallel Adapter

3 CPU

3.1 Zahlendarstellung

Festkomma (Integer)



n entspricht Wortlänge

übliche Werte 8, 16, 32

LSB = Least significant Bit

MSB = Most significant Bit

Abbildung 69 Darstellung von Dualzahlen in Registern

Darstellbare Werte für vorzeichenlose Zahlen:

000...000 Wert 0

111...111 Wert $2^n - 1$

Darstellung negativer Werte als Komplemente

Rezept zur Bildung des Zweierkomplements aus negativen Zahlen:

1. Vorzeichen ignorieren
2. Alle Stellen invertieren (Einerkomplement)
3. Eins dazu addieren

Darstellbarer Zahlenbereich:

000...000 Wert 0

011...111 Wert $2^{n-1} - 1$

100...000 Wert -2^{n-1}

111...111 Wert -1

Negative Zahlen im Zweierkomplement kann man daran erkennen, dass das MSB=eins ist.

3.1.1 Gleitkomma (Floating Point)

Bei Gleitkommazahlen sind nach IEEE 754 folgende Formate in Gebrauch:

- Halbe Genauigkeit (16-Bit) (ca. seit 2008)
- Einfache Genauigkeit (32-Bit) (ca. seit 1980)
- doppelte Genauigkeit (64-Bit) (ca. seit 1980)
- 4-fache Genauigkeit (128-Bit) (ab ca. 2008)

Die am meisten verbreiteten Gleitkommiformate sind 32 und 64 Bit

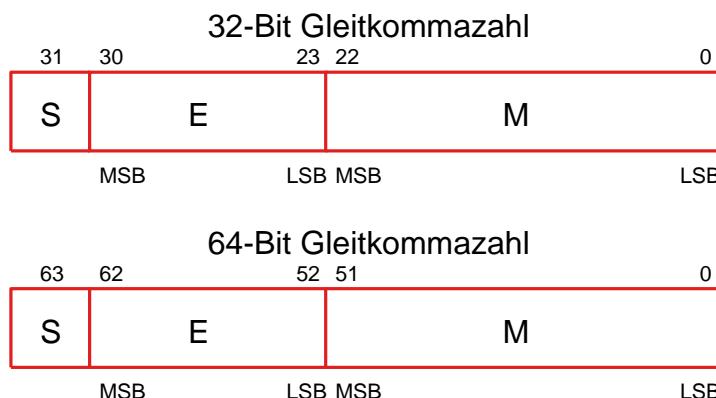


Abbildung 70 Gleitkommazahlen

	16 Bit	32 Bit	64 Bit	128 Bit
S=Vorzeichen	1 Bit	1 Bit	1 Bit	1 Bit
E=Exponent	5 Bit	8 Bit	11 Bit	15 Bit
M=Mantisse	10 Bit	23 Bit	52 Bit	112 Bit

Der Wert des Exponenten wird nicht direkt gespeichert, sondern es wird vorher ein Biaswert hinzugaddiert, dadurch werden negative Binärwerte beim Exponenten verhindert.

Wenn der Exponent dem Wert 0 entspricht, wurde für den Wert der Mantisse definiert, dass dieser dann immer zwischen 1 und 2 liegt. Das höchstwertigste Bit hätte damit immer den Wert 1. Deshalb wurde dieses Bit weggelassen und das MSB auf den Wert $\frac{1}{2}$ festgelegt. Dadurch erhöht sich die mögliche Rechenauflösung um ein zusätzliches Bit.

Wert 32 Bit: $N = (-1)^S \cdot (1,0+M) \cdot 2^{(E-127)}$

Wert 64 Bit: $N = (-1)^S \cdot (1,0+M) \cdot 2^{(E-1023)}$

Wertebereiche (Dezimal):

	Kleinste Zahl	Größte Zahl
16 Bit:	$\pm 0,00006104$	± 65504
32 Bit:	$\pm 1.175 \cdot 10^{-38}$	$\pm 3.4 \cdot 10^{38}$
64 Bit:	$\pm 2.225 \cdot 10^{-308}$	$\pm 1.798 \cdot 10^{308}$
128 Bit	$\pm 3.362 \cdot 10^{-4932}$	$\pm 1.190 \cdot 10^{4932}$

Früher gab es auch Gleitkommiformate im dezimalen BCD-Format, die jedoch heute fast keine Rolle mehr spielen.

3.2 ALU

Die Arithmetisch-Logische-Einheit ist eine Zusammenfassung aller Logik und Rechenfunktionen in einer CPU. Die Speicher innerhalb einer CPU, die direkt Zahlen verarbeiten können, nennt man Register. Sie sind eine Ansammlung von Flip-Flops. Die Rechenregister z.B. A/B/Y entsprechen immer der Systembreite der CPU.

In der Praxis hat sich folgendes Symbol durchgesetzt:

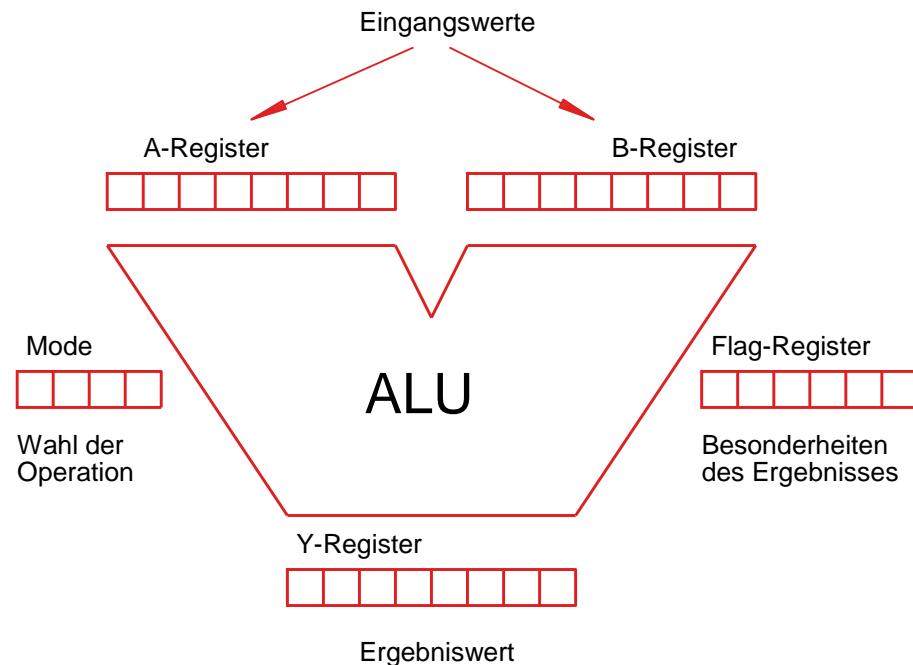


Abbildung 71 ALU

Beispiele für das Mode-Register:

- ADD Addition
- SUB Subtraktion
- MUL Multiplikation
- DIV Division
- AND Und-Verknüpfung
- OR Oder-Verknüpfung
- XOR Exklusiv-Oder-Verknüpfung
- ...

Beispiele für das Flag-Register:

- Carry Flag: Übertrag ist aufgetreten
- Overflow Flag: Ergebnis passt nicht ins Y-Register
- Zero Flag: Ergebnis ist Null
- Negation Flag: Ergebnis ist negativ
- ...

3.3 Rechenwerk

3.3.1 Addition

3.3.1.1 Halbaddierer

Aufgabe: Addition von einstelligen Dualzahlen

Rechenregeln:

Für die erste Stelle

$$\begin{array}{l} 0+0=0 \\ 0+1=1 \\ 1+0=1 \\ 1+1=0 \end{array}$$

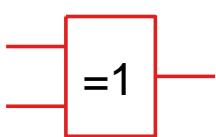


Abbildung 72 Exklusiv ODER

Für die zweite Stelle

$$\begin{array}{l} 0+0=0 \\ 0+1=0 \\ 1+0=0 \\ 1+1=1 \end{array}$$

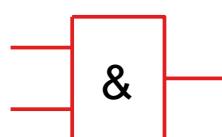


Abbildung 73 UND

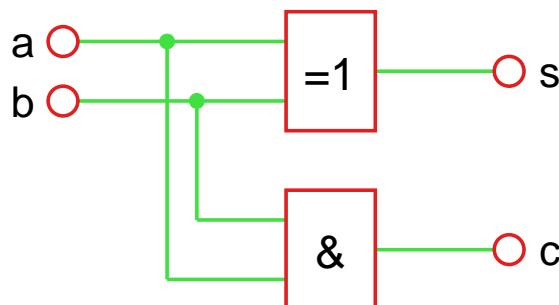


Abbildung 74 Halbaddierer

Wahrheitstabelle:

a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Realisierung mit Standardelementen

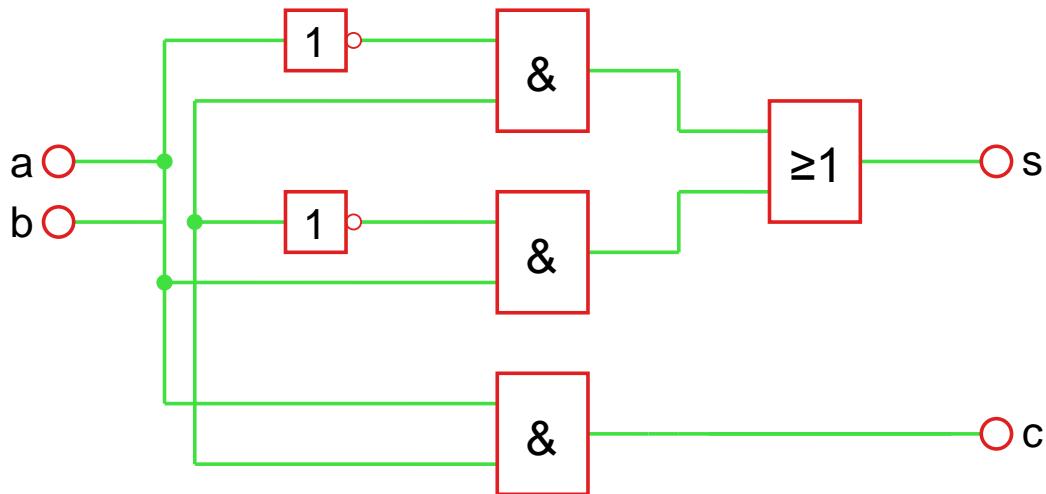


Abbildung 75 Halbaddierer mit Standardelementen

Formel für S: $s = (\bar{a} \wedge b) \vee (a \wedge \bar{b})$

Formel für C: $c = a \wedge b$

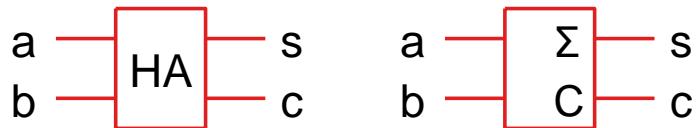


Abbildung 76 Halbaddierer Schaltsymbole

3.3.1.2 Volladdierer

Der Volladdierer berücksichtigt zusätzlich den von der nächstniedrigeren Stelle ankommenden Übertrag.

Wahrheitstabelle:

c_{i-1}	a_i	b_i	c_i	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Mögliche Realisierung durch 2 Halbaddierer + Oder-Gatter

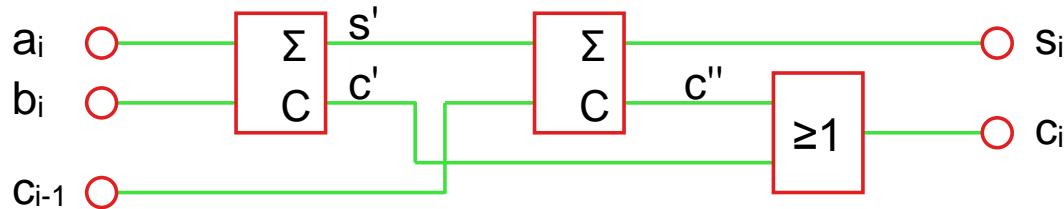


Abbildung 77 Volladdierer aus Halbaddierern

Wahrheitstabelle:

INPUT			HAI		HA2		ODER	
c_{i-1}	a_i	b_i	c'	s'	c''	s_i	c_i	
0	0	0	0	0	0	0	0	
0	0	1	0	1	0	1	0	
0	1	0	0	1	0	1	0	
0	1	1	1	0	0	0	1	
1	0	0	0	0	0	1	0	
1	0	1	0	1	1	0	1	
1	1	0	0	1	1	0	1	
1	1	1	1	0	0	1	1	

Formel für s_i : $s_i = (\overline{c_{i-1}} \wedge \overline{a_i} \wedge b_i) \vee (\overline{c_{i-1}} \wedge a_i \wedge \overline{b_i}) \vee (c_{i-1} \wedge \overline{a_i} \wedge \overline{b_i}) \vee (c_{i-1} \wedge a_i \wedge b_i)$

Formel für c_i : $c_i = (a_i \wedge b_i) \vee (c_i \wedge a_i) \vee (c_{i-1} \wedge b_i)$

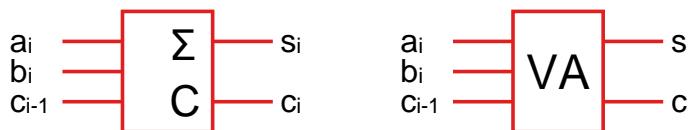


Abbildung 78 Volladdierer Schaltsymbole

3.3.1.3 Paralleladdierwerk

Um eine komplette Addition zweier Register durchzuführen, ist folgender Ablauf notwendig:
Die beiden Summanden stehen in zwei Registern mit der Bitbreite N. Das *Steuersignal Addition* aktiviert das Paralleladdierwerk. Mit der nächsten Taktflanke wird das Ergebnis dann in das A-Register übernommen.

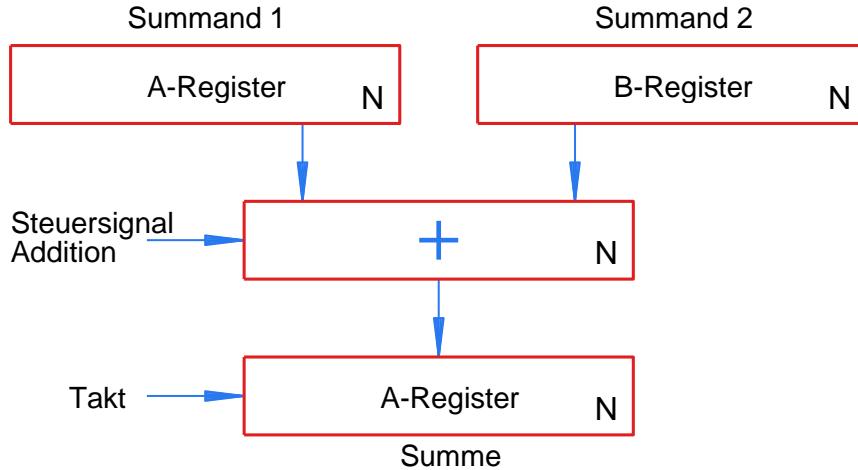


Abbildung 79 Paralleladdierwerk

Um eine Subtraktion durchzuführen, wird beim Subtrahend (B-Register) noch eine Zweierkomplement-Stufe dazwischen geschaltet. Siehe Kapitel 3.3.2

Ripple Carry

Um mehrstellige Zahlen zu addieren, wird pro Dualstelle ein Volladdierer benötigt.

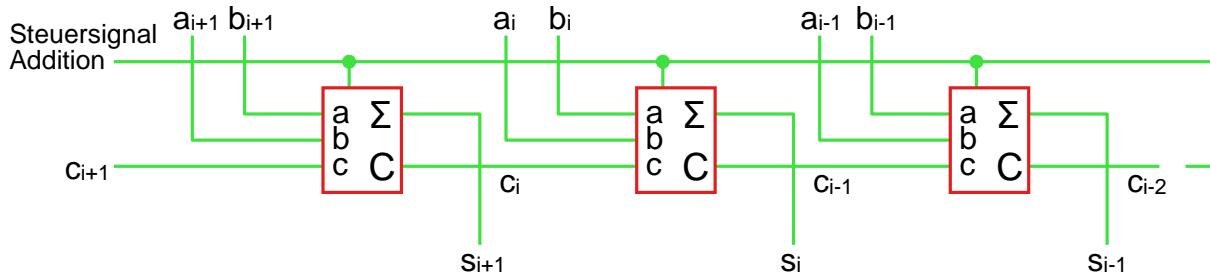


Abbildung 80 Paralleladdierer mit Volladdierern

Wenn an mehreren Stellen einer Addition ein Übertrag auftritt, kann es im ungünstigsten Fall dazu führen, dass der Übertrag durch alle Stellen hindurchlaufen muss. Das führt zu einer sehr starken Verlangsamung des Vorgangs, da man bei einer N-Bit-Addition N Durchlaufzyklen warten muss, bis das endgültige Ergebnis feststeht.

Carry Look Ahead

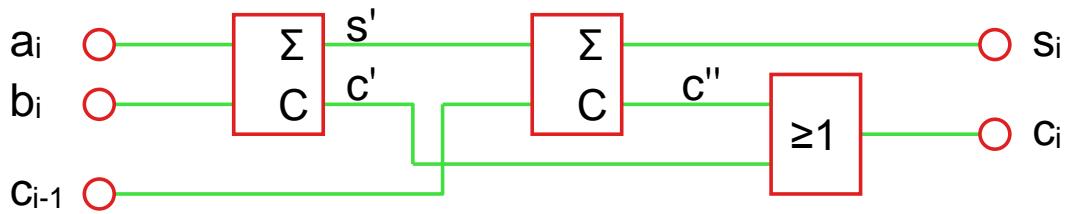


Abbildung 81 Übertragsbetrachtung

$$C_i = C_i' \vee C_i''$$

$$C_i = C_i' \vee (C_i'' \wedge S_i)$$

entstehender durchlaufender
Übertrag Übertrag

Um das Verzögerungs-Zeit-Problem des durchlaufenden Übertrags zu optimieren, ist es notwendig, nicht nur den Übertrag der vorherigen Stufe zu berücksichtigen, sondern den von allen Stufen.

Das Oder-Gatter des Volladdierers, welches bisher nur den Übertrag der zwei Halbaddierer berechnet hat, muss entsprechend erweitert werden, um alle möglichen Überträge sofort auswerten zu können.

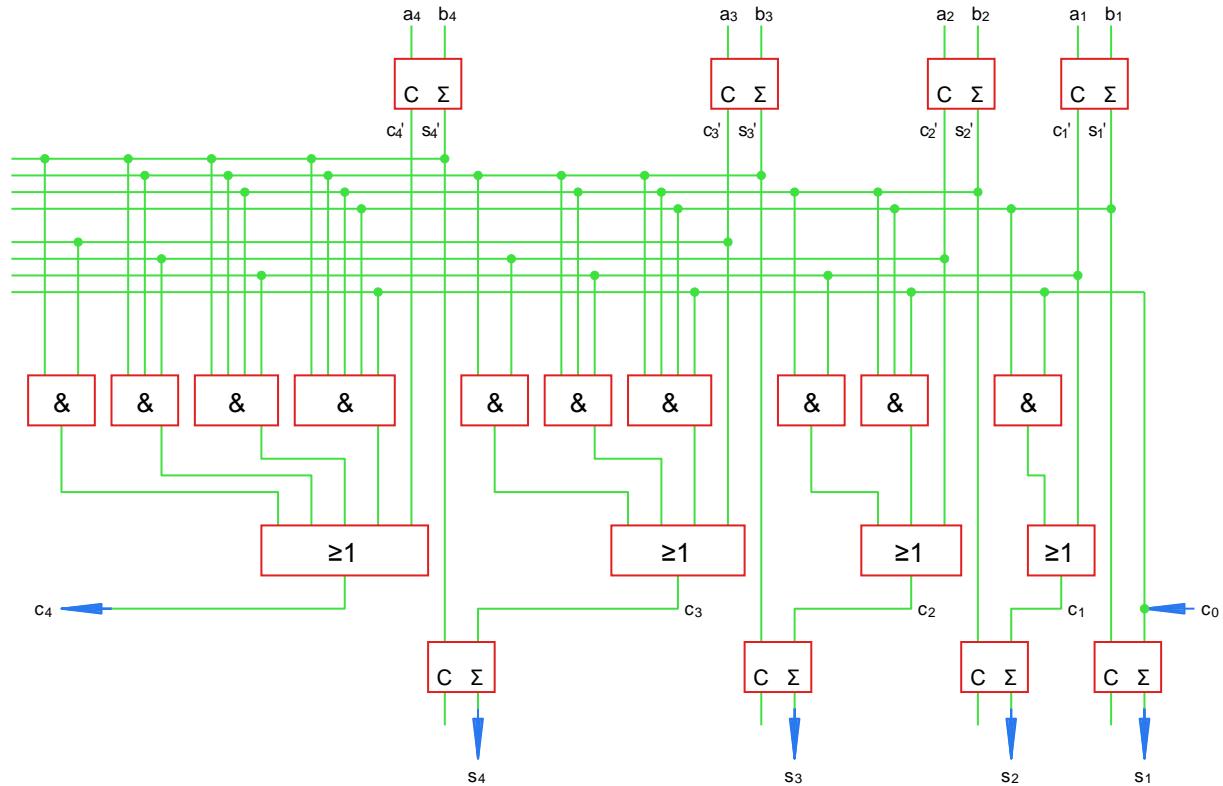


Abbildung 82 4-Bit Carry Look Ahead Addierer

3.3.1.4 Inkrement

Das Inkrementieren ist eine sehr häufige benötigte arithmetische Funktion, z.B. bei einer Schleifenberechnung ist das Inkrementieren der Zählvariablen eine typische Aktion.

Inkrementieren entspricht einer Addition mit 1, d.h. fast alle Stellen (bis auf das LSB) des einen Summanden sind null. Wenn an einem Eingang eines Volladdierers immer null anliegt, kann man die Schaltung entsprechend vereinfachen:

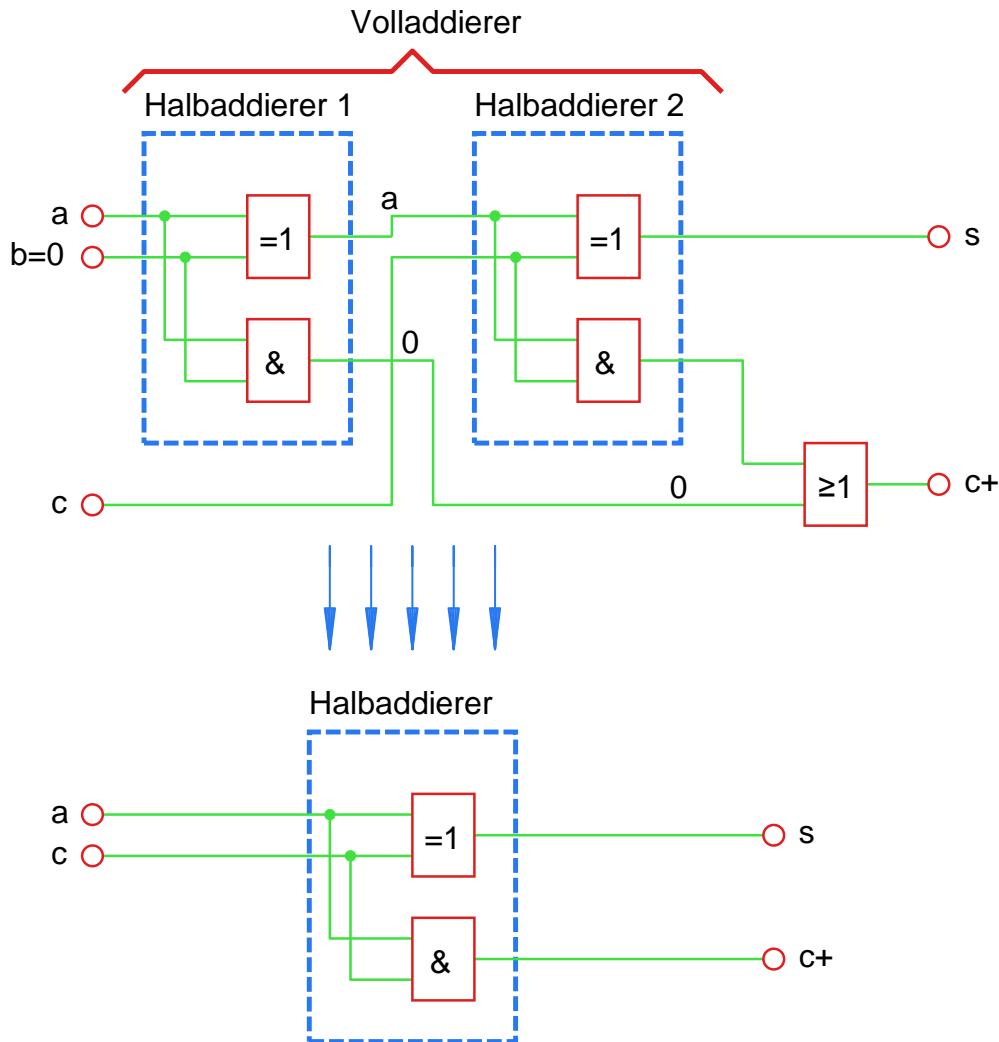


Abbildung 83 Inkrementator

Bis auf das LSB kann pro Binärstelle dann je ein Volladdierer durch einen Halbaddierer ersetzt werden. Damit kann man viele Logikelemente einsparen und die Durchlaufzeiten sind auch entsprechend kleiner.

3.3.2 Subtraktion

Eine Subtraktion wird vom Rechenwerk nicht anders behandelt als eine Addition. Um eine Subtraktion durchzuführen, wird vom Subtrahenden das Zweierkomplement gebildet, welches sich bei einer Addition wie eine negative Zahl verhält.

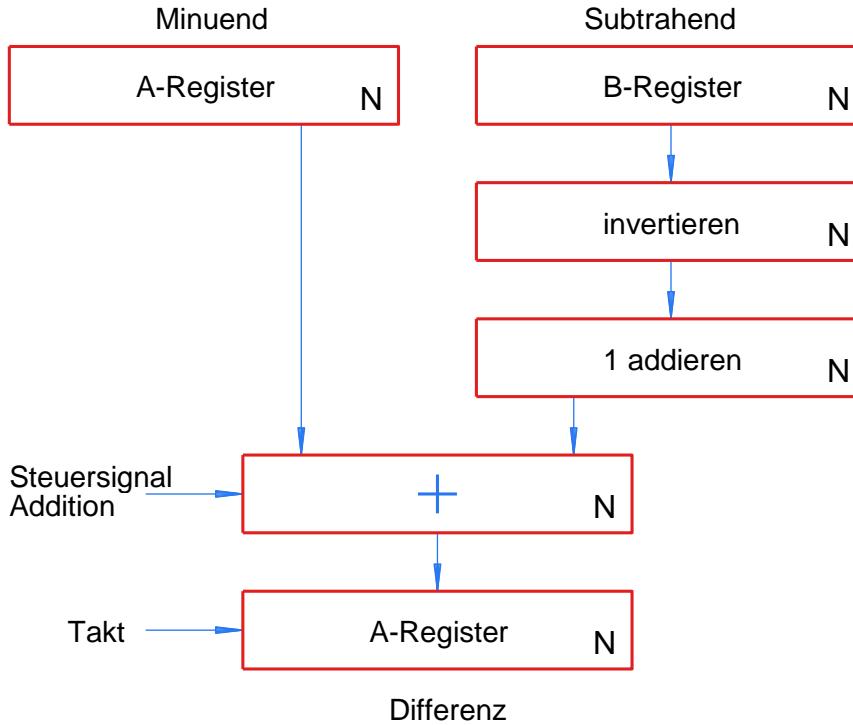


Abbildung 84 Parallele Subtraktion

Physikalisch gab es bei herkömmlichen Prozessoren nur ein Addierwerk, welches für eine Subtraktion dann 2 Mal benutzt wird. Eine Subtraktion dauerte deshalb etwas länger als eine Addition.

Spätere Prozessoren haben, um noch schneller rechnen zu können, jedoch mehrere Addierwerke. Insbesondere das Addieren von 1 kann mit einem vereinfachten Addierwerk erfolgen.

3.3.3 Shift

3.3.3.1 Standard Shift

Shift-Operationen (Schiebe-Operationen) sind nach der Addition und Subtraktion die dritte fundamentale Operation, die man benötigt, um eine vollwertige ALU zu erhalten. Bei einer Shift-Operation werden alle Bits eines Registers um eine Position nach rechts oder links verschoben. Bei einem Standard-Shift um mehrere Stellen wird pro Taktzyklus um eine Stelle weitergeschoben.

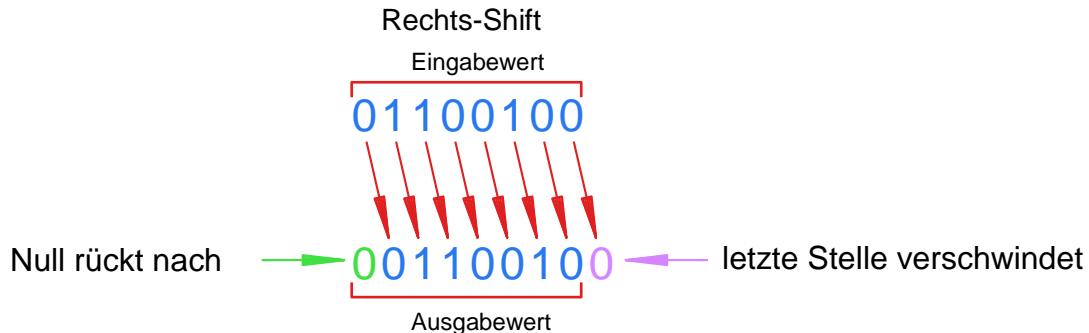


Abbildung 85 Shift

3.3.3.2 Shiftarten

Je nachdem was am rechten oder linken Ende des Registers passiert, wird zwischen zwei grundsätzlichen Shift-Arten unterschieden:

- Logischer Shift
- Arithmetischer Shift

Daraus kann man 4 Kombinationsmöglichkeiten ableiten:

Shift-Typ	Linkes Ende	Rechtes Ende
Logischer Rechts-Shift	0 rückt nach	Stelle verschwindet
Logischer Links-Shift	Stelle verschwindet	0 rückt nach
Arithmetischer Rechts-Shift	MSB bleibt stehen, 1 rückt rechts daneben nach	Stelle verschwindet
Arithmetischer Links-Shift	MSB bleibt stehen, Bit rechts daneben verschwindet	0 rückt nach

Shift-Operationen werden oft verwendet, um Zahlen mit dem Faktor 2 zu multiplizieren oder zu dividieren (siehe Kapitel 3.3.6). Bei einem arithmetischen Shift wird das MSB belassen und erst die nächstniedrigere Stelle wird verschoben. Damit wird bei Zahlen im Zweierkomplement das „Vorzeichen“ korrekt behandelt.

Eine Shift-Funktion wird in der Regel mit D-Flip-Flops realisiert, die zusätzlich parallele Ein- und Ausgänge wie R/S-Flip-Flops haben.

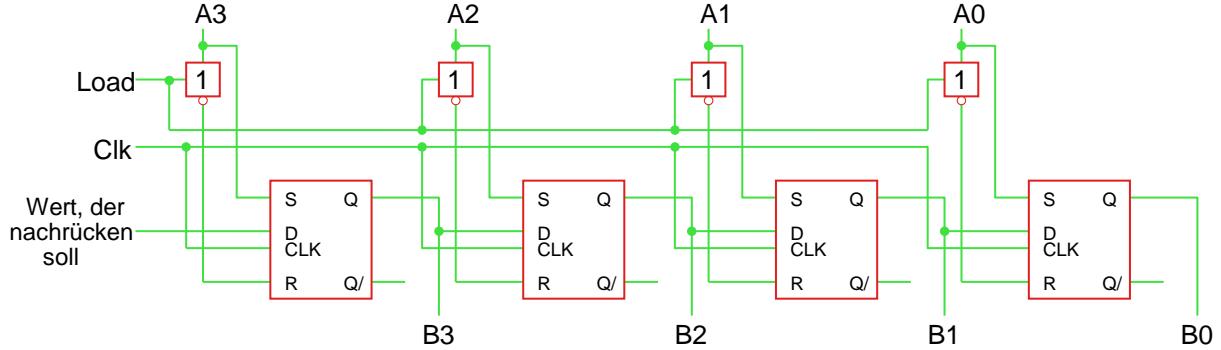


Abbildung 86 Shift mit Flip-Flops

An die A-Eingänge wird das einzugebende Datenwort angelegt und mit Load in die Flip-Flops gespeichert. Danach wird mit jedem Clock-Impuls um eine Stelle nach rechts weitergeschoben. Für einen Links-Shift ist das Gleiche noch mal aufzubauen, oder man kann die Q- und D-Leitungen mit einem Multiplexer je nach Anforderung nach rechts oder links verbinden.

3.3.3.3 Barrel-Shifter

Heutige Prozessoren haben gewöhnlicherweise einen Barrelshifter, bei dem ein mehrstelliger Shift in einem einzigen Taktzyklus durchgeführt wird. Damit dies möglich wird, muss man dafür eine Matrix aufbauen, die es ermöglicht, jede Stelle des Quellregisters mit jeder Stelle des Zielregisters zu verbinden.

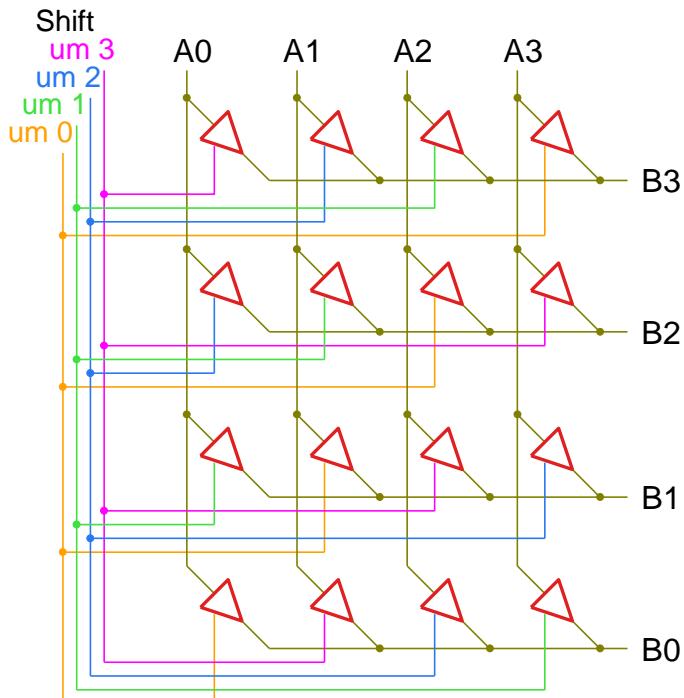


Abbildung 87 4-Bit-Barrel-Shifter

3.3.4 Multiplikation

3.3.4.1 Multiplikation mit Algorithmus

Ausgehend von der schriftlichen Multiplikation wird diese auf Binärzahlen übertragen.

$$\begin{array}{r}
 1011 \times 1101 \\
 \hline
 1011 \\
 0000 \\
 1011 \\
 1011 \\
 \hline
 10001111
 \end{array}$$

Abbildung 88 Multiplikation Schulmethode

Eine Multiplikation kann durch Verschiebe- und Additionsoperationen realisiert werden.

Für eine vollständige Multiplikation sind folgende Schritte notwendig:

- Vorzeichen der beiden Faktoren EXOR verknüpfen
- Multiplikationsalgorithmus nur mit den Beträgen

Multiplikand x Multiplikator

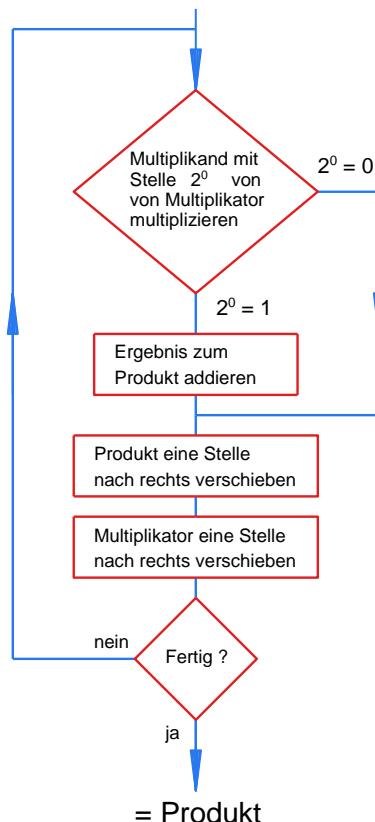


Abbildung 89 Multiplikationsalgorithmus

3.3.4.2 Multiplikation mit Hardware

Hardwarealgorithmus

Den Multiplikations-Algorithmus aus Kapitel 3.3.4.1 kann man auch direkt in Hardware abbilden. Dazu wird für jede Binärstelle ein Volladdierer mit der Bitbreite eines Multiplikationsfaktors benötigt (aus Performance-Gründen mit Carry-Look-Ahead, siehe Kapitel 3.3.1.3).

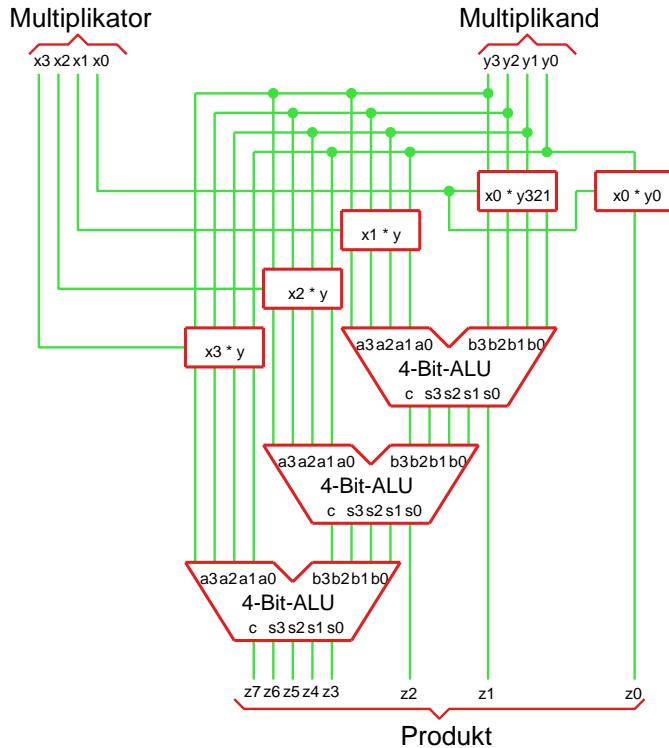


Abbildung 90 4-Bit-Hardware-Multiplikation mit Addierwerken

Diese Art der Multiplikation wurde bereits bei den 16-Bit-Prozessoren ca. Mitte der 1980er Jahren eingeführt (siehe Kapitel 3.6.2).

Für eine vollständige 32-Bit-Multiplikation ist ein immenser Aufwand nötig. Hier ein exemplarischer Entwurf für einen 32-Bit-Multiplizierer.

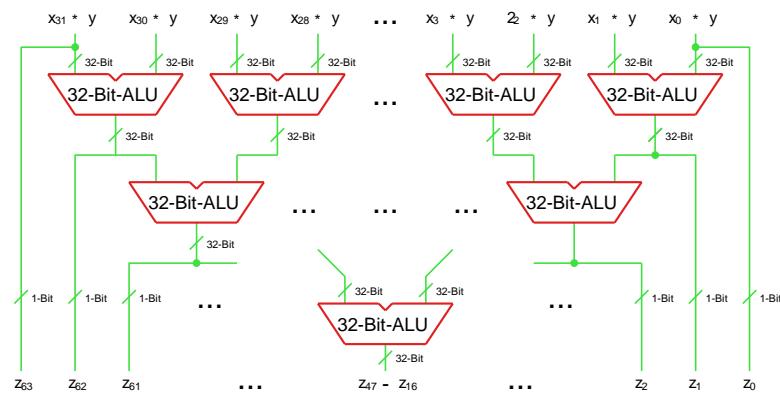


Abbildung 91 32-Bit-Hardwaredmultiplikation mit Addierwerken

Es existieren in der Praxis noch einige weitere Optimierungsmöglichkeiten, um die Anzahl der Addierwerke zu reduzieren. Die entsprechenden Begriffe für die heute üblichen Optimierungen lauten: „Wallace-Tree-Multiplizierer“ und „Dadda-Tree-Multiplizierer“.

Multiplikations-Lookuptable

Eine andere Art, um die Zeit für eine Multiplikation zu verkürzen, ist eine Look-Up-Table. Bei dieser wurden sämtliche Multiplikationsergebnisse schon vorher ausgerechnet.

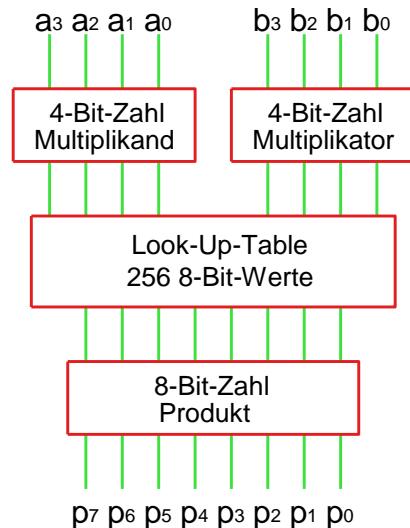


Abbildung 92 Hardware-Multiplikation mit Look-Up-Table

Der Speicherbedarf der Tabelle steigt quadratisch mit der Bitanzahl. Damit die Tabelle bei größeren Bitbreiten nicht zu groß wird, werden die beiden Methoden kombiniert, um eine ausreichend schnelle Multiplikation zu erreichen. Durch diese Kombination ist es heute möglich eine 64-Bit-Multiplikation in einem einzigen Prozessortakt durchzuführen.

3.3.5 Division

Ausgehend von der schriftlichen Division wird diese auf Binärzahlen übertragen.

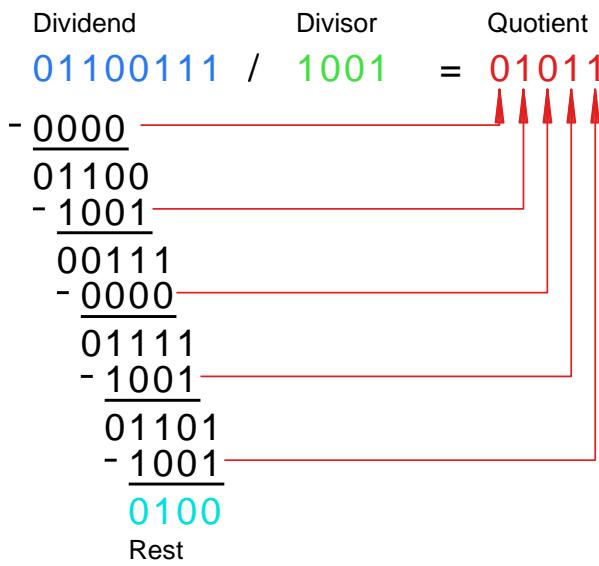


Abbildung 93 Division Schulmethode

Ähnlich wie bei der Multiplikation kann eine Division durch Verschiebe- und Subtraktionsoperationen realisiert werden.

3.3.6 Faktor 2^x

Eine Besonderheit soll hier noch betrachtet werden:

- Divisionen bzw. Multiplikationen mit dem Faktor 2^x

Eine Division bzw. Multiplikation mit Faktor 2 entspricht einer Shift-Operation um 1 Bit.

Links Shift	Binärwert	Dezimalwert
	11001000	200
	01100100	100
	00110010	50
	00011001	25
Rechts Shift		

Abbildung 94 Shift um 2

Viele Compiler erkennen den Faktor 2^x und ersetzen die Multiplikation automatisch durch einen Shift-Befehl, weil dieser schneller und einfacher durch die CPU zu erledigen ist. Die Operation dauert dann auch auf älteren CPUs üblicherweise nur 1 Taktzyklus.

3.3.7 Faktor 256 hoch x

Eine weitere Besonderheit stellt der Faktor 256 dar. Es entspräche einer Shift-Operation um 8-Bit. Fast alle modernen Compiler führen in diesem Fall gar keine Operation durch, sondern greifen auf die Adresse der zu multiplizierenden Variablen um 1 Byte versetzt zu.

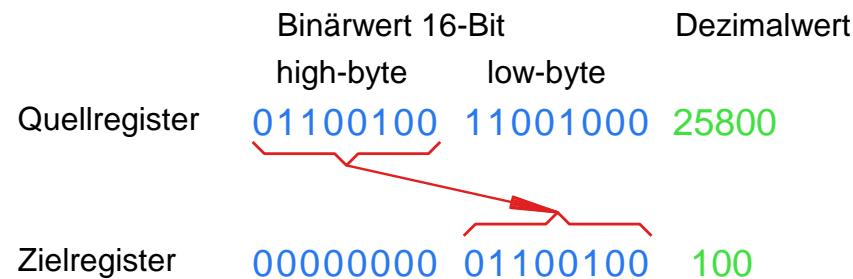


Abbildung 95 Faktor 256

Dadurch wird die Zahl automatisch ohne irgendeine Rechenoperation der ALU um Faktor 256 kleiner. Diese Rechnung kostet keine Zeit. Es erfolgt lediglich ein Umspeicher-Vorgang, der beim Abspeichern sowieso stattfindet. Durch modernes Pipelining erfolgt diese „Rechenoperation“ dann wirklich in 0 Taktzyklen.

Bei allen anderen Faktoren von 256^x wird das gleiche Prinzip angewendet, hier findet dann eine Verschiebung um x Byte statt.

3.3.8 MAC

Bei Signalverarbeitungsalgorithmen wird sehr oft eine Faltung oder eine Fourier-Transformation durchgeführt. Bei diesen Berechnungen besteht der Kern-Algorithmus aus vielen Multiplikationen und der anschließenden Integration der Ergebnisse.

$$z = z + (x \cdot y)$$

Damit diese Rechnung sehr schnell durchgeführt werden kann, haben Signalprozessoren eine MAC-Einheit (**M**ultiply **A**Ccumulate).

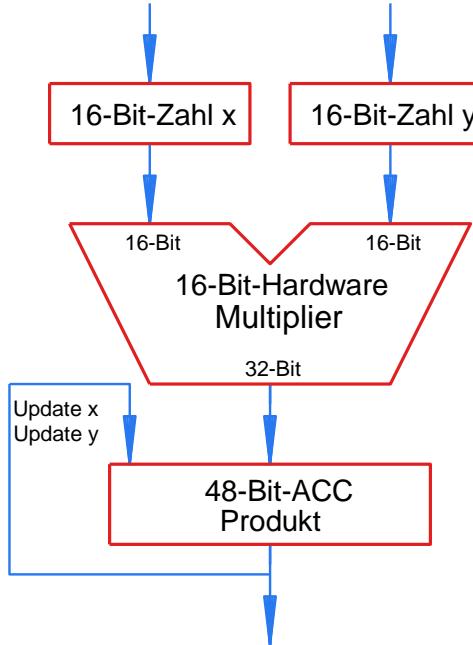


Abbildung 96 16-Bit MAC mit 48-Bit ACC

Das MAC-Ergebnisregister (ACC=Accumulation-Register) der Multiplikation ist oft breiter als die Ergebnisbreite des Multipliers; hier können deshalb viele Multiplikations-Ergebnisse aufintegriert werden.

Die Eingangswerte x und/oder y des Multipliers können bei den meisten Signalprozessoren mittels automatischen Zeigern auf den nächsten Multiplikator weitergeschaltet werden. Dabei ist es oft sogar möglich, einen Bit-Reversed-Zeiger zu benutzen. Dies ist sehr hilfreich bei der Umsetzung des Butterfly-Algorithmus für eine FFT (Fast-Fourier-Transformation) in Software.

FMA (Fused-Multiply-Add):

In neueren CPU-Designs (ab ca. 2014) wurde diese Art der Berechnungsmöglichkeit auch auf die FPU ausgedehnt. Hier wurde die Berechnung noch um eine weitere Operation erweitert, Das Zwischenergebnis nach der Fließkommamultiplikation wird nicht gerundet (unfused), es wird mit der vollen Auflösung (fused) weitergerechnet, was zu genaueren Endergebnissen führt.

3.3.9 SISD / SIMD

Normalerweise kann in einem Register einer CPU nur eine Operation auf den gesamten Registerinhalt angewandt werden. Diese traditionelle Arbeitsweise wird bei Prozessoren SISD (**Single Instruction Single Data**) genannt.

Für die meisten Berechnungen sind jedoch die heutigen 64 Bit breiten Register der modernen 64-Bit-Prozessoren schlicht nicht notwendig. Damit wäre z.B. bei 16-Bit-Operationen 75% des 64-Bit-Registers ungenutzt.

Um die Registerbreite der CPU dennoch voll ausnutzen zu können, werden diese heutzutage für den SIMD-Betrieb (**Single Instruction Multiple Data**) erweitert. Hier wird das Register in mehrere Teile geteilt, und die Einzelteile werden wie separate Register behandelt. Dadurch werden bei 64-Bit-Registern bis zu 8x 8-Bit-Berechnungen gleichzeitig durchgeführt.

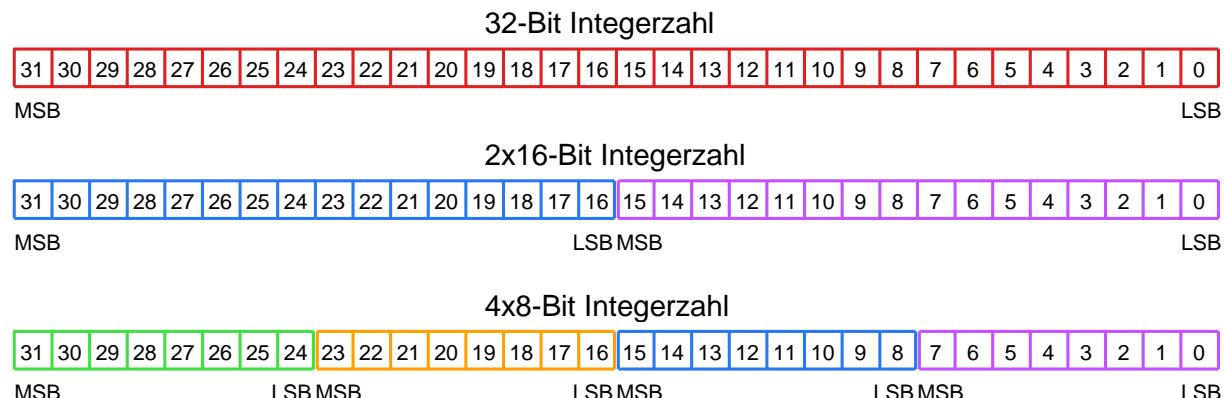


Abbildung 97 SIMD Beispiele bei 32-Bit-Registern

3.3.10 Sättigungsarithmetik (MMX)

Die zweite Erweiterung, die fast gleichzeitig mit SIMD eingeführt wurde, um bei Signalverarbeitungen schneller zu werden, ist die Sättigungsarithmetik.

Der bei den Grundrechenarten mögliche Über/Unterlauf wird durch die Sättigungsarithmetik verhindert, d.h., das Ergebnis im Register bleibt auf dem größtmöglichen Registerwert stehen. Bei Signalverarbeitungen musste bisher der Über/Unterlauf durch die Software nachfolgend abgefangen werden.

Diese Funktion wurde von Intel MMX (**M**atrix **M**ath **E**xtensions oder **M**ulti **M**edia **E**xensions) getauft und funktioniert optimal im Zusammenspiel mit SIMD.

Beispiel für 8-Bit-Sättigungsarithmetik:

$$\begin{array}{ll} \text{ohne Sättigung:} & 187 + 175 = 106 + \text{Carry-Bit} = 106 + 256 = 362 \\ \text{mit Sättigung:} & 187 + 175 = 255 \end{array}$$

Diese Funktionen wurden im nächsten Schritt auf die FPU ausgedehnt. Gleichzeitig wurde die Registerbreite der FPU auf 128 Bit erweitert. Die offizielle Bezeichnung von Intel für diese Erweiterung der FPU trägt den Namen SSE (**S**treaming **SIMD** **E**xtensions). Mit der Erweiterung der Registerbreite auf 256 Bit wurde der Begriff AVX eingeführt (**A**dvanced **V**ector **E**xtensions) bzw. AVX-512 bei der Erweiterung auf 512 Bit.

3.4 Steuerwerk

3.4.1 Befehlszyklus

Um ein Programm abzuarbeiten, sind für jeden einzelnen Assemblerbefehl folgende Einzel-Schritte notwendig:

- Befehl holen (vom Arbeitsspeicher)
- Befehl dekodieren
- Operanden holen, falls notwendig (vom Arbeitsspeicher)
- Befehl ausführen
- Auswerten der Steuersignale (Flags)
- Ergebnisse zurückschreiben (zum Arbeitsspeicher)
- Adressberechnungen
- Nächsten Befehl holen (vom Arbeitsspeicher)

Die gesamte Abfolge wird auch Von-Neumann-Zyklus genannt.

3.4.2 Befehlsaufbau



Abbildung 98 Opcode

Das Befehlswort kann größer sein als die Bit-Breite der Maschine selbst.

Opcode

- Ist immer im Befehlswort enthalten
- Enthält die auszuführende Operation
- Definiert die Bedeutung des Adressteils
- Definiert die Länge des Befehlswortes

Adressteile

- Enthält Registeradresse eines Operanden
- Enthält Speicheradresse eines Operanden
- Oder Direktoperand
- Oder Sprungziel (bei Sprungbefehlen)

Zusätze

- Enthält Angaben über die Art der Adressrechnung
- Mehrfachausführung eines Befehls

Befehlsarten

- **Transportbefehle**
Register -> Register
Register -> Speicher
Speicher -> Register
Speicher -> Speicher
- **Arithmetische Befehle**
Addition
Subtraktion
Multiplikation (nicht immer)
Division (nicht immer)
 Inkrementieren
 Dekrementieren
- **Arithmetische Gleitkomma-Befehle**
Grundrechenarten
Wurzel
Trigonometrie
- **Logische Befehle**
Und- Verknüpfung (Bitweise)
Oder Verknüpfung (Bitweise)
Exklusiv-Oder Verknüpfung (Bitweise)
- **Schiebebefehle**
Arithmetischer Shift
Logischer Shift
- **Vergleichsbefehle**
Werden als Subtraktion ausgeführt ohne Registeränderung
nur die Flags werden beeinflusst
- **Sprungbefehle**
Das Programm wird nicht mit dem auf den Sprungbefehl folgendem Befehl
ausgeführt, sondern mit dem Befehl an der Stelle des Adressteils im Sprungbefehls.
Unbedingter Sprung
Bedingter Sprung
- **Bitfeld- und Flag-Befehle**
Zur direkten Beeinflussung von Flags und einzelnen Bits eines Registers
- E/A-Befehle (in, out)
Input (von Peripherie)
Output (zur Peripherie)
- **Sonderbefehle**
Interrupt-Steuerung

Befehls-Optimierung

Eigentlich müsste jeder Befehl die folgenden 5 Angaben enthalten:

Auszuführende Operation (OPcode)
Adresse des ersten Operanden
Adresse des zweiten Operanden
Adresse an der das Ergebnis abzulegen ist
Adresse des nächsten Befehls

OPcode	Adressteil 1	Adressteil 2	Adressteil 3	Adressteil 4
--------	--------------	--------------	--------------	--------------

Abbildung 99 OPcode mit 4 Adressen

Dies kann man folgendermaßen vereinfachen:

- Aufeinanderfolgende Befehle eines Programms stehen in Speicherstellen mit aufeinanderfolgenden Adressen. Die Adresse des Folgebefehls kann aus der Adresse des aktuellen Befehls durch einen Zählvorgang ermittelt werden.
-> Adressteil 4 entfällt.
- Bei Operationen mit 2 Operanden wird einer der beiden durch das Ergebnis überschrieben
-> Adressteil 3 entfällt.

Man kommt somit zur sogenannten „Zweiadressmaschine“

OPcode	Adressteil 1	Adressteil 2
--------	--------------	--------------

Abbildung 100 OPcode mit 2 Adressen

- Bei Operationen mit 2 Operanden steht grundsätzlich einer der beiden in einem als Akkumulator bezeichneten Register. Dieses Register nimmt auch das Ergebnis auf.
-> Adressteil 2 entfällt.

Man kommt somit zur sogenannten „Einadressmaschine“

OPcode	Adressteil 1
--------	--------------

Abbildung 101 OPcode mit einer Adresse

Heutige Computer sind eine Mischung aus Einadressmaschine und Zweiadressmaschine.

3.4.3 CISC/RISC

Die CISC-Architektur eines Computers ist entstanden in Zeiten früher Compiler mit dem Ziel, dass die CPU-Maschinencode-Befehle sehr nahe an den Hochsprachen sein sollten. Jeder Einzelbefehl beschäftigte die CPU für eine längere Zeit. Ein Grund dafür lag auch bei den damals sehr langsamem Speichern. Wenn die CPU eine Zeit lang beschäftigt war, wurde der Speicher nicht so häufig angesprochen.

Die meisten RISC-Architekturen entwickelten sich durch viele kleine Schritte aus einer CISC-Maschine. Der Aufwand der CPU auf Chipebene als RISC-Maschine ist deutlich höher als bei einer eher komplexen CISC-Maschine, da die Softwarealgorithmen der CISC-Maschine bei der RISC Maschine durch Hardware abgebildet werden müssen. Ähnlich wie beim Hardwaremultiplizierer (siehe Kapitel 3.3.4.2).

Bei einer CISC-Maschine wurden Teile der Hardware in Software, sogenannte Mikro-Programme, ausgelagert.

CISC (Complex Instruction Set Computers):

- Relativ kompakte Kodierung von Software
- Unterschiedliche Befehlswortlänge
- Für jeden Befehl werden viele interne Taktzyklen benötigt
- Mikro-Programme für jeden Befehl notwendig
- Frühe Compiler nutzen nicht alle Befehle

RISC (Reduced Instruction Set Computers):

- Feste Befehlswortlänge
- Einfache Adressierungsarten
- Modernere Compiler schließen die Lücke zu den Hochsprachen durch Optimierung
- Rein in Hardware realisierbar (nur Gatter und Flip-Flops)
- Keine Mikro-Programme

Durch die feste RISC-Befehlswortlänge wird die Dekodierung eines Befehls sehr einheitlich und geht viel schneller. Da alle Befehle gleich lang sind, wird dadurch die Einführung einer Pipeline möglich (siehe Kapitel 3.4.4).

Viele Prozessorhersteller sind in den letzten Jahren dazu übergegangen keine reinen RISC Maschinen zu bauen, sondern zwischen dem eigentlichen RISC-Kern und dem Maschinencode, der vom Compiler geliefert wird, eine Art Interpreter zu schalten. Bei Intel ab ca. 1995 mit dem **Pentium Pro**. (Ab dem Pentium wurde kurz vorher 1993 bei dessen Einführung zusätzlich noch der Hauptteil der CPU (Rechenwerk, Steuerwerk) doppelt ausgeführt).

Dadurch ist jeder heutige Rechner im Kern zwar eine RISC-Maschine, nach außen betrachtet (aus Compiler-Sicht) sieht die CPU eher wie eine CISC-Maschine aus.

3.4.4 Pipelining

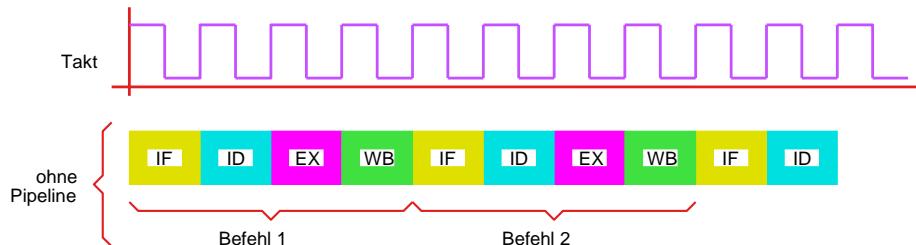
3.4.4.1 Grundprinzip

Prozessoren kann man in einige unterschiedliche Verarbeitungsklassen unterteilen:

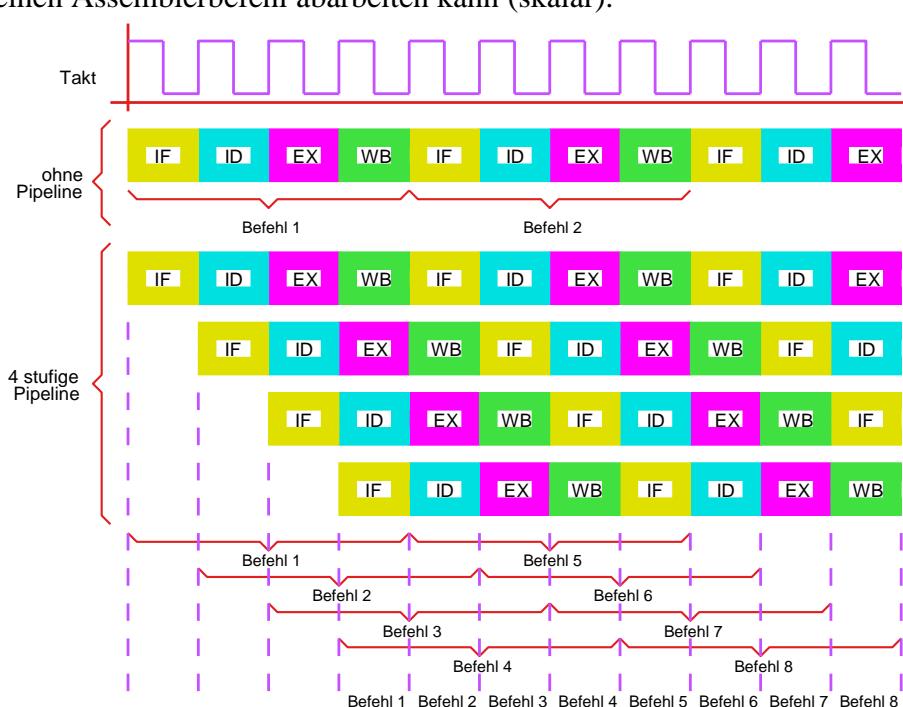
- nicht skalar weniger als 1 Assemblerbefehl pro Taktzyklus
- skalar 1 Assemblerbefehl pro Taktzyklus
- superskalar mehr als 1 Assemblerbefehl pro Taktzyklus

Um einen einzigen Assemblerbefehl abzuarbeiten, sind einige teilweise unterschiedlich viele Verarbeitungsschritte nacheinander durchzuführen (siehe Kapitel 3.4.1). Zum besseren Überblick werden hier immer nur 4 Verarbeitungsschritte pro Befehl aufgezeigt.

1. Befehl holen (**IF**, Instruction Fetch)
2. Befehl dekodieren (**ID**, Instruction Decoding)
3. Befehl ausführen (**EX**, EXecution)
4. Ergebnis wegschreiben (**WB**, Write Back)



Damit alle Verarbeitungsstufen immer was zu tun haben, kann man sie wie in einem Fließband (Pipeline) organisieren. Erst jetzt ist es möglich, dass der Prozessor pro Taktzyklus einen Assemblerbefehl abarbeiten kann (skalar).



3.4.4.2 Superskalar

Als weiteren Entwicklungsschritt kann man die gesamte Pipeline dann doppelt ausführen, um mindestens 2 Befehle in einem Takt zu schaffen (bei Intel ab 1993 mit dem **Pentium** -> U und V-Pipe).

Moderne Prozessoren verwenden heutzutage oft eine Kombination aus mehreren Pipelines und Ausführungseinheiten (Execution-Units).

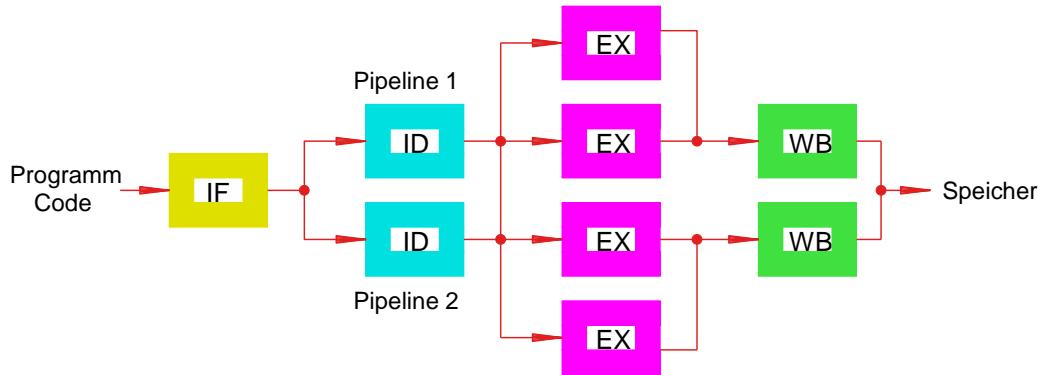


Abbildung 104 CPU Superskalar logischer Aufbau

3.4.4.3 Out Of Order Execution

Manche Befehlsabfolgen lassen sich schlecht mit einer oder mehreren Pipelines parallel verarbeiten. Das Vertauschen von Befehlsreihenfolgen kann manchmal helfen, um die Pipelines besser auszunutzen. Dazu werden einige Befehle in eine Warteschlange (IB Instruction Buffer) eingereiht, und der Befehlsdecoder prüft dann mittels Heuristik, ob er einen Befehl vorziehen kann, wenn gerade eine entsprechende Execution-Unit frei ist.

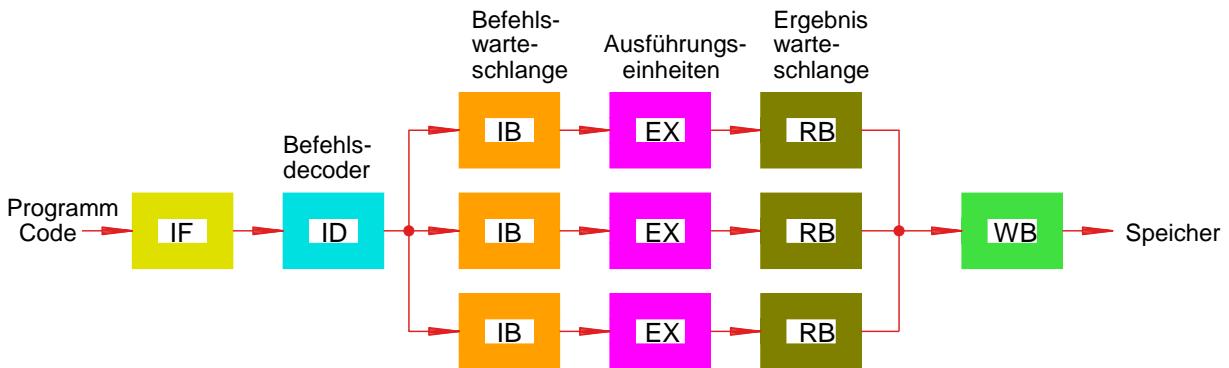


Abbildung 105 Out Of Order Execution logischer Aufbau

Nach der Ausführung werden die Ergebnisse in eine Ergebnis-Warteschlange (RB **Re-order Buffer** oder **Register Retirement Buffer**) eingereiht und in der ursprünglichen Reihenfolge dann wieder an den Speicher bzw. das Ergebnisregister übergeben.

3.4.4.4 Branch Prediction

Um den ursprünglichen Programmcode noch weiter zu optimieren, ist in modernen Prozessoren noch eine weitere Einheit dazugekommen: Die Sprungvorhersage-Einheit (Branch-Prediction-Unit).

Die Sprungvorhersage sorgt in Zusammenarbeit mit der Out-Of-Order-Unit dafür, dass bei einer Verzweigung im Programmcode der wahrscheinlichste Zweig schon im Voraus ausgeführt wird.

Bei der Sprungvorhersage wird deshalb spekulativer Code (Speculative Execution) ausgeführt, der bei einer falschen Vorhersage unbrauchbar ist und verworfen werden muss.

Die Hardware der Sprungvorhersage-Einheit zusammen mit der spekulativen Codeausführung ist eine sehr aufwendige Technologie, die bei den meisten CPUs den Teil ausmacht, der am meisten Energie verbraucht!

3.4.4.5 Fazit

Durch das Zusammenspiel von:

- mehrfachen Pipelines
- mehrfachen pipelineübergreifenden Execution-Units
- Out-Of-Order -Execution
- Branch-Prediction-Unit
- Speculative Execution

wird eine deutliche Steigerung der Performance einer CPU erreicht. Das Pipelining hat sich über die Jahre stark weiterentwickelt und die Pipelines wurden immer länger.

Diese Entwicklung lief bis ca. 2005 mit Pipelines mit über 30 Stufen in eine Sackgasse.

Beim Versagen der Sprungvorhersage musste der Inhalt der gesamten Pipeline verworfen werden. Bei einer 30-stufigen Pipeline dauert es dann 30 Takte, bis es dann weitergeht. Heutige Prozessoren haben deshalb keine riesigen Pipelines mehr, die mehr als 15 Stufen beinhalten.

Alle Optimierungen zusammen bilden eine sehr komplexe Maschine. Leider ist dadurch in neuester Zeit (ab 2018) eine Sicherheitslücke (Stichwörter: Spectre, Meltdown) in der Hardware fast aller CPU-Hersteller enthalten, für die bis ca. 2020 noch keine vollständigen Lösungen existierten.

3.5 Gleitkommawerk (FPU)

3.5.1 Aufbau

Die FPU ist eine Ansammlung von optimierten Rechenalgorithmen für Fließkommazahlen. Die Arbeitsweise einer FPU entspricht im Prinzip der UPN (Umgekehrte-Polnische-Notation). Der große Vorteil von UPN ist, dass man mit nur 4 Stack-Speichern alle Arten von Berechnungen durchführen kann. Die meisten FPUs enthalten mindestens 8 Register mit je 80 Bit Breite, die wie in einem Stack organisiert sind. Es kann immer nur das Register ST0 geändert werden. Alle weiteren Werte werden mittels Push nach unten geschoben.



Abbildung 106 FPU Register

Des Weiteren sind einige oft gebräuchliche mathematische Konstanten direkt in der FPU- gespeichert und ersparen somit den Speichertransfer:

Funktion	FPU-Befehl	Wert
1	Fld1	1.00000000...
π ,	fldpi	3.14159265...
$\log_2(e)$,	fld12e	1.44269504...
$\log_e(2)$,	fldln2	0.69314718...
$\log_{10}(2)$	fldlg2	0.30102999...

Abbildung 107 FPU Konstanten

UPN-Beispielrechnung:

Um z.B. die drei Zahlen a, b, c zu addieren, sind folgende Operationen notwendig:

- Lade ST0 Register mit c
- Push
- Lade ST0 Register mit b
- Push
- Lade ST0 Register mit a
- Addiere ST0 mit ST1 (Zwischenergebnis a+b landet in ST0)
- Addiere ST0 mit ST1 (Endergebnis (a+b)+c landet in ST0)
- Pop ST0 nach Ziel

3.5.2 Rechenweise

Um Fließkommazahlen miteinander zu verrechnen, sind unterschiedliche Arbeitsweisen notwendig.

Bei Additionen ist eine andere Vorbereitung der Operanden notwendig als bei Multiplikationen oder Divisionen oder bei trigonometrischen Operationen.

Beispiel Ablauf für eine Addition:

- Zahlen in Mantisse und Exponent und Vorzeichen zerlegen
- Exponenten vergleichen
- Der größere bestimmt den Ergebnisexponent
- Mantisse mit kleinerem Exponenten bis auf den Ergebnisexponenten verschieben
- Vorzeichen des Ergebnisses bestimmen durch Entscheidungsmatrix
- Mantissen addieren
- Mantisse auf normale Stellenzahl runden und kürzen
- Ergebnis aus Mantisse, Exponent und Vorzeichen wieder zusammensetzen

Beispiel Ablauf für eine Multiplikation:

- Zahlen in Mantisse und Exponent und Vorzeichen zerlegen
- Vorzeichen des Ergebnisses bestimmen durch EXOR
- Exponenten addieren
- Über/Unterlauf abfangen
- Mantissen multiplizieren
- Falls Überlauf verschieben und Exponenten anpassen
- Mantisse auf normale Stellenzahl runden und kürzen
- Ergebnis aus Mantisse, Exponent und Vorzeichen wieder zusammensetzen

Die FPU hatte bisher keinen Zugriff auf den Adressbus. Sämtliche Daten werden über die CPU transferiert. Diese grundsätzliche Arbeitsweise sorgte in der Vergangenheit dafür, dass bei FPU-Operationen die CPU quasi blockiert ist.

Bei späteren Weiterentwicklungen (SSE) der FPU wurde die Registerbreite auf 128-Bit erhöht und es wurde ermöglicht, dass die SSE-Einheit unabhängiger von der CPU arbeiten kann. Allerletzter Stand (SSE5) ist die Erweiterung der Registerbreite auf 256-Bit, und die Einführung sehr vieler Spezialbefehle wie FFT oder AES-Verschlüsselung.

3.6 Desktop CPUs von Intel

3.6.1 Intel CPU Evolution

Bezeichnung	Architektur	Besonderheiten
8086	16-Bit-CPU	<ul style="list-style-type: none"> - Data/Adress-Multiplexer - 4,77 MHz Takt - ca. 14 Takte / Assemblerbefehl
8088	16-Bit-CPU	<ul style="list-style-type: none"> - nur 8-Bit externer Datenbus - für low-cost System-Design
80186	16-Bit-CPU	<ul style="list-style-type: none"> - nur im Industriebereich eingesetzt - eher Mikrocontroller-Design
80286	16-Bit-CPU	<ul style="list-style-type: none"> - erste vollwertige 16-Bit-CPU von Intel - 2. ALU für Adressberechnungen - 24 Adressleitungen.
80386	32-Bit-CPU	<ul style="list-style-type: none"> - letzte wirklich große Änderung der Grundarchitektur - erste 1:1 Kopien von AMD im Streit mit Intel - später low cost SX-Variante mit 16-Bit Bus - fast pinkompatibel zum 80286
80486	32-Bit-CPU	<ul style="list-style-type: none"> - FPU und 1st Level Cache (8-16k) integriert - 1:1 Kopien von AMD - später viele Versionen von anderen Herstellern
Pentium	32-Bit-CPU	<ul style="list-style-type: none"> - doppelte Pipeline (U/V-Pipe) - getrennter jeweils 8k-Cache für Daten- und Programmcode - AMD kauft NexGen und bringt den pinkompatiblen K6
Pentium MMX	32-Bit-CPU	<ul style="list-style-type: none"> - SIMD Einführung - Cache auf 2x16k verdoppelt
Pentium Pro	32-Bit-CPU	<ul style="list-style-type: none"> - CISC-RISC Kombination - 2 Chips 2nd Level Cache mit vollem Takt - Einführung von Speculative Execution und Sprungvorhersage - Sehr schlechte Performance bei 16-Bit Betriebssystemen (Problem durch Segment Register)
Pentium II	32-Bit-CPU	<ul style="list-style-type: none"> - großes Plastikgehäuse - 2nd Level Cache mit ½-Speed - Bessere Performance bei 16-Bit Betriebssystemen (jetzt mit Segment Register Cache)
Pentium III	32-Bit-CPU	<ul style="list-style-type: none"> 1. Generation immer noch mit Plastikgehäuse und ½-Speed Cache 2. Generation mit Fullspeed-2nd-Level-Cache auf demselben Chip
Pentium M	32-Bit-CPU	<ul style="list-style-type: none"> - Stromsparversion des Pentium III für mobile Computer - Entwicklung Intel Israel
Pentium IV	32-Bit-CPU oder 64-Bit-CPU	<ul style="list-style-type: none"> - Intel-Netburst-Architektur mit zuletzt 31 Pipeline-Stufen - Heizleistung ca. 100W/cm² - Weiterentwicklung wurde aufgegeben!
Core	64-Bit-CPU	<ul style="list-style-type: none"> - Intel-Core-Architektur - Weiterentwicklung auf Basis des Pentium M - Nur 11 Pipeline-Stufen
Core iX	64-Bit-CPU	<ul style="list-style-type: none"> - Einführung 3rd Level Cache bis 10MB - neue 128-256-Bit FPU-Register - standardmäßig mit mindestens 2 Cores - teilweise mit integrierter GPU - aktuelle Technologiegröße (2024) 10nm

Abbildung 108 Überblick der CPU-Entwicklung von Intel:

3.6.2 Multiplikationsperformance

CPU - Typ	Multiplikation Taktcycles	Architektur
Z80	750	Software
8086	120	Mikrocode
80286	21	16-Bit-Hardware-Multiplier
80386	9-17	32 Bit Mixed-Design
80486	14	32-Bit-Hardware-Multiplier
Pentium	11	Doppel-Hardware-Multiplier
Pentium Pro	1-4	RISC-Interpreter
MMX	1-0,25	8/16 Bit mit SIMD

Abbildung 109 Rechenzeiten Integer-Multiplikation

3.6.3 Chipaufbau 80486

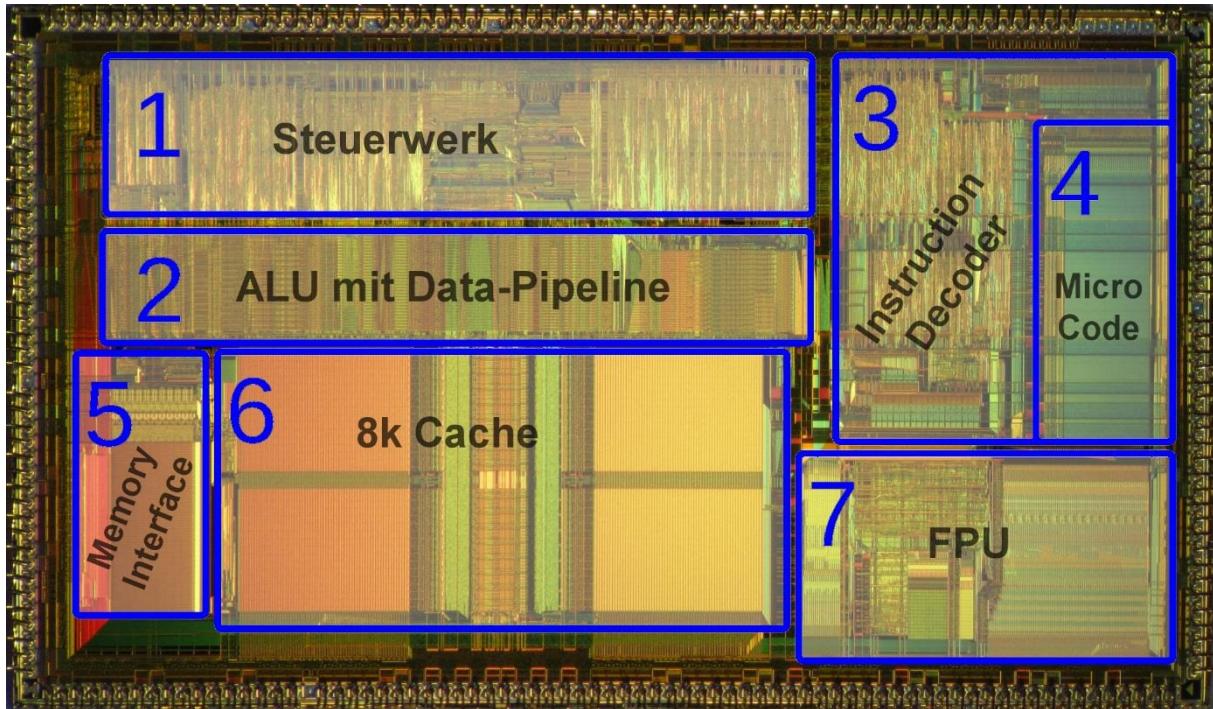


Abbildung 110 80486 Chipanalyse

4 Busse

4.1 Zusammenspiel aller Busse

Damit alle relevanten Daten über den Datenbus überragen werden können, gibt es in einem Rechnersystem ein Systemtiming, an das sich alle Komponenten halten müssen. Alle Daten- und Adressbussignale müssen eine gewisse Zeit anstehen, bevor sie von allen Komponenten korrekt verarbeitet werden können. Die CPU ist normalerweise die einzige Komponente, die bestimmt, was auf den Bussen wann abläuft.

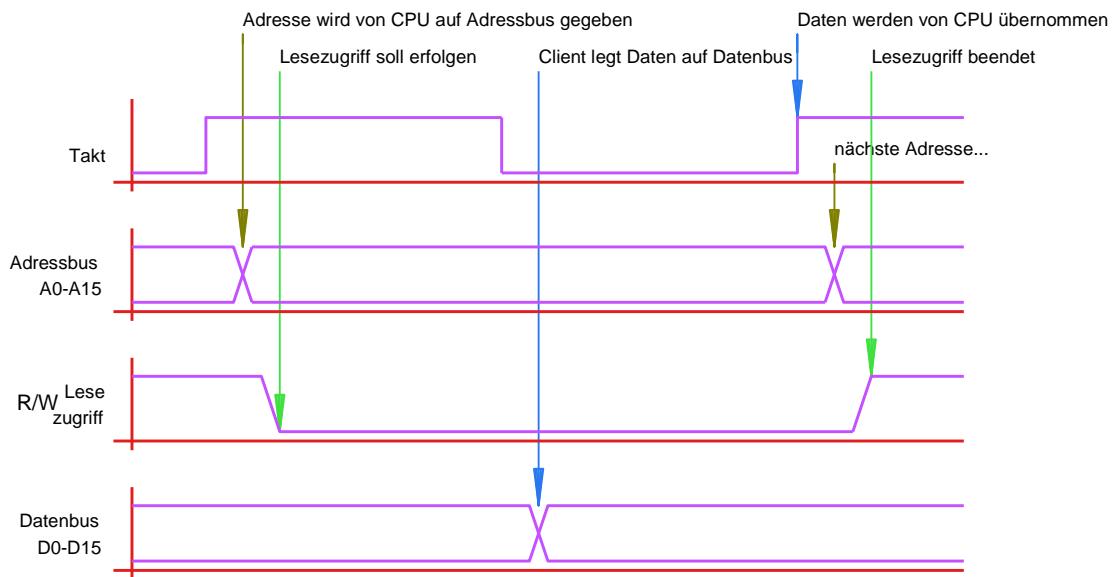


Abbildung 111 Bustiming Lesezugriff

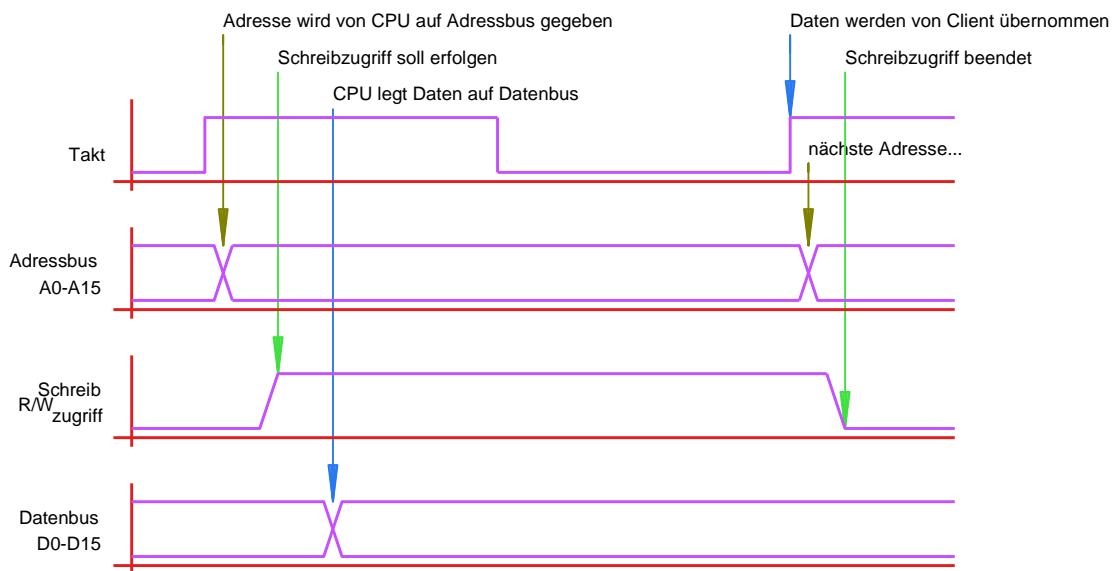


Abbildung 112 Bustiming Schreibzugriff

Vom Steuerbus ist hier als Beispiel nur eine einzige Leitung aufgezeigt: Die Leitung R/W, welche bestimmt ob die CPU Daten über den Datenbus lesen oder schreiben möchte. Das Anlegen der Adress/Daten- und der R/W Leitung kann auch gleichzeitig erfolgen. Hierbei wäre schon eine Übernahme auf der fallenden Flanke der Clock-Leitung möglich->DDR.

4.2 Ein/Ausgabe (I/O)

4.2.1 Memory Mapping

4.2.1.1 Hardwaresicht

In einem Computersystem ist normalerweise nicht an allen adressierbaren Speicheradressen Speicher angeschlossen. In bestimmten Adressbereichen wird das Datum auf dem Datenbus nicht in den Arbeitsspeicher geschrieben, sondern wird durch andere Geräte repräsentiert.

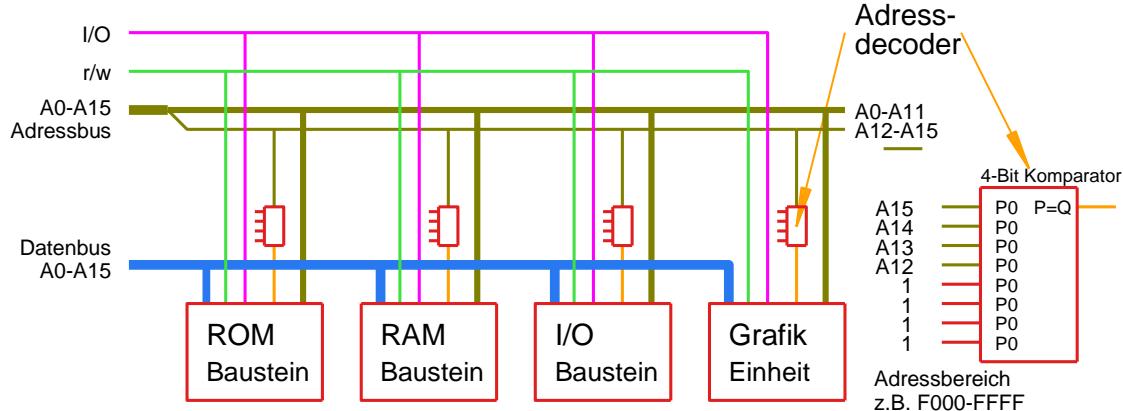


Abbildung 113 Adressdekodierung

Üblicherweise wird dafür der oberste Adressbereich reserviert um z.B. Grafikausgaben, Soundausgaben durchzuführen oder andere Geräte anzusprechen. Bei gewöhnlichen PCs sind deshalb bei 32-Bit-Systemen nicht 4 GByte (2^{32} Bit) Speicher maximal möglich, sondern nur ca. 3 GByte, da der obere Speicherbereich vom Betriebssystem für andere Hardware reserviert wird.

Für 64-Bit-Systeme ist dieser Effekt zu vernachlässigen, da der Adressvorrat bei 2^{64} Bit um ein Vielfaches größer ist.

Eine andere Methode Peripheriegeräte anzusprechen ist, über eine zusätzliche Leitung (I/O-Leitung) anzuzeigen, dass entweder Speicher oder andere Hardware angesprochen werden soll. Der Adressdecoder ist dann natürlich trotzdem notwendig.

Die mittlerweile bei aktuellen Hochleistungsprozessoren völlig andere Ansteuerung von Speicher über einzelne Speicherkanäle hat das Design entsprechend stark verändert. (siehe Kapitel 2.3.4.2)

4.2.1.2 Softwaresicht

Als Beispiel hier einen Datenblattauszug für das komplette Memory-Mapping eines typischen Cortex-M4-Mikrocontrollers:

2.2.2 Memory map and register boundary addresses				
	Bus	Boundary address	Size	Peripheral
AHB2		0xE000 0000 - 0xE010 0000	1MB	Cortex®-M4 with FPU internal peripherals
		0x4800 1800 - 0x5FFF FFFF	~384 MB	Reserved
		0x4800 1400 - 0x4800 17FF	1KB	GPIOF
		0x4800 1000 - 0x4800 13FF	1KB	GPIOE
		0x4800 0C00 - 0x4800 0FFF	1KB	GPIOD
		0x4800 0800 - 0x4800 0BFF	1KB	GPIOC
		0x4800 0400 - 0x4800 07FF	1KB	GPIOB
		0x4800 0000 - 0x4800 03FF	1KB	GPIOA
		0x4002 4400 - 0x47FF FFFF	~128 MB	Reserved
		0x4002 4000 - 0x4002 43FF	1 KB	TSC
AHB		0x4002 3400 - 0x4002 3FFF	3 KB	Reserved
		0x4002 3000 - 0x4002 33FF	1 KB	CRC
		0x4002 2400 - 0x4002 2FFF	3 KB	Reserved
		0x4002 2000 - 0x4002 23FF	1 KB	FLASH memory interface
		0x4002 1400 - 0x4002 1FFF	3 KB	Reserved
		0x4002 1000 - 0x4002 13FF	1 KB	RCC
		0x4002 0800 - 0x4002 0FFF	2 KB	Reserved
		0x4002 0400 - 0x4002 07FF	1 KB	DMA2
		0x4002 0000 - 0x4002 03FF	1 KB	DMA1
		0x4001 6C00 - 0x4001 FFFF	37 KB	Reserved
APB2		0x4001 6800 - 0x4001 6BFF	1 KB	SDADC3
		0x4001 6400 - 0x4001 67FF	1 KB	SDADC2
		0x4001 6000 - 0x4001 63FF	1 KB	SDADC1
		0x4001 5C00 - 0x4001 5FFF	1 KB	TIM19
		0x4001 5800 - 0x4001 5BFF	1 KB	DBGMCU
		0x4001 4C00 - 0x4001 57FF	4 KB	Reserved
		0x4001 4800 - 0x4001 4BFF	1 KB	TIM17
		0x4001 4400 - 0x4001 47FF	1 KB	TIM16
		0x4001 4000 - 0x4001 43FF	1 KB	TIM15
		0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved
		0x4001 3800 - 0x4001 3BFF	1 KB	USART1
		0x4001 3400 - 0x4001 37FF	1 KB	Reserved
		0x4001 3000 - 0x4001 33FF	1 KB	SPI1/I2S1
		0x4001 2800 - 0x4001 2FFF	1 KB	Reserved
		0x4001 2400 - 0x4001 27FF	1 KB	ADC
APB1		0x4001 0800 - 0x4001 23FF	7 KB	Reserved
		0x4001 0400 - 0x4001 07FF	1 KB	EXTI
		0x4001 0000 - 0x4001 03FF	1 KB	SYSCFG + COMP
		0x4000 A000 - 0x4000 FFFF	32 KB	Reserved

Abbildung 114 Memory-Map

4.2.2 Polling

Polling (auch programmed I/O) ist die einfachste Methode mit Computern Daten an andere Hardware zu übergeben. Beim Polling muss jedes Gerät zyklisch nach seinem Status befragt werden, es muss überprüft werden, ob das Gerät bereit ist, ob es Daten senden oder empfangen möchte usw.

Beim Polling ist die CPU ständig damit beschäftigt, alle Geräte mit maximaler Arbeitsfrequenz zu bedienen, d.h., es herrscht ständig 100 % Prozessorauslastung. Je mehr Prozesse dazukommen, umso langsamer wird der Zyklus. Polling ist nur sinnvoll, wenn in einem System wenige Geräte vorhanden sind und genügend (preiswerte) Prozessorleistung zur Verfügung steht. Polling ist relativ einfach in Software umzusetzen.

4.2.3 Interrupt

Wenn viele unterschiedlich schnelle Geräte mit einer CPU zu bedienen sind, ist es ratsam, einige dieser Aufgaben per Hardware-Interrupt zu erledigen.

Jeder Prozessor hat dazu eine extra Interrupt Leitung an Bord. Das Grundprinzip ist dabei Folgendes: Wenn ein Gerät kommunizieren möchte, verändert es den Pegel dieser Leitung. Die CPU unterbricht darauf das aktuelle Programm und springt mit seiner Programmausführung automatisch an die Speicher-Stelle, an der die Interrupt-Service-Routine (ISR) abgelegt ist. Generell haben Prozessoren nur eine Interrupt-Leitung. Wenn es mehrere gibt, ist immer vor die eigentliche Interruptleitung der CPU ein Interruptcontroller geschaltet.

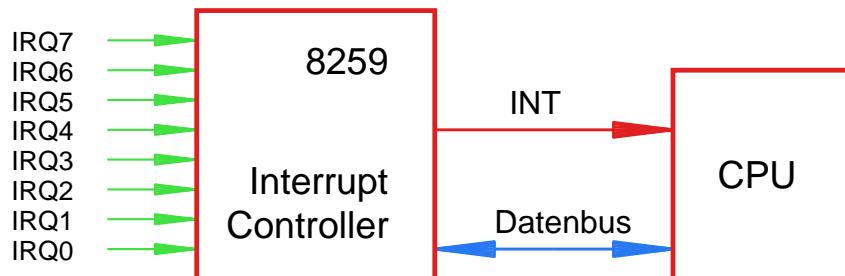


Abbildung 115 Interrupt-Controller

Oft ist der Interruptcontroller heutzutage mit in die CPU integriert. Dann ist auch die Verwaltung der unterschiedlichen Interrupt-Service-Routinen (Sprungvektortabelle) gleich auch noch mit dabei.

Bei den einzelnen Interrupts kann normalerweise auch noch die Priorität untereinander oder die Art des Interruptsignals (flankengesteuert oder pegelgesteuert) eingestellt werden.

IRQ-Nummer	Priorität	Typ	Sprung-Adresse
0	1	F	0x5456
1	2	P	0x3421
2	1	F	0x6542
3	0	F	0x1356
4	3	P	0x0201
5	4	F	0x4903

Abbildung 116 Beispiel ISR-Vektor-Tabelle

4.3 Multiplexing

Beim Multiplexing werden mehrere Signale über dasselbe physikalische Medium übertragen.
Es gibt verschiedene physikalische Multiplexmethoden:

- Raummultiplex (Richtfunk, MIMO (Multiple Input Multiple Output))
- Frequenzmultiplex (Funk, Lichtwellen)
- Zeitmultiplex (Busse)
- Codemultiplex (Computernetzwerke)

Innerhalb von Computern wird fast immer das Zeitmultiplexverfahren angewandt.

Insbesondere bei CPUs werden oft der Daten- und der Adressbus über gemeinsame Leitungen übertragen. Das wurde früher oft gemacht, um Pins an den Gehäusen von CPUs einzusparen.
Beispiel 8086 im DIL40 Gehäuse.

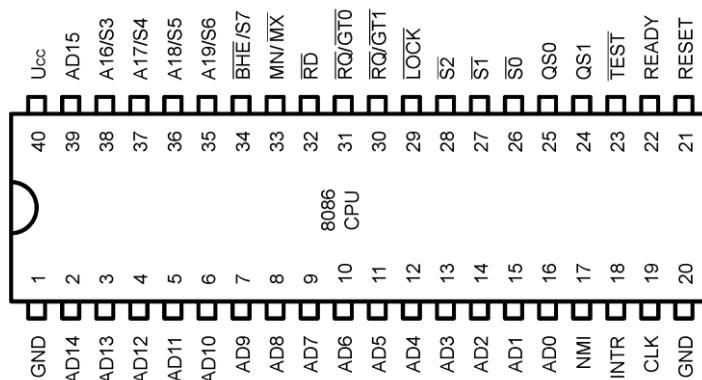


Abbildung 117 8086 Pinout

Später beim PCI-BUS wurde der Daten und Adressbus gemultiplexed um, die Größe des Bus-Steckers klein zu halten.

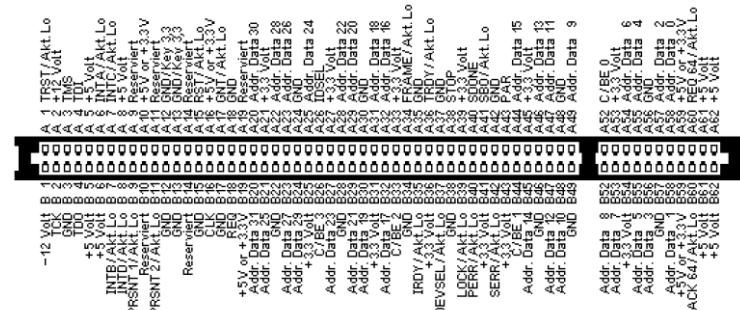


Abbildung 118 PCI-BUS-Belegung

Durch den Trend zu seriellen Bussen spielt das Multiplexing in größeren Designs heute kaum noch eine Rolle. Die Pinanzahl von IC-Gehäusen scheint auch fast keine Rolle mehr zu spielen. Aktuelle (2024) Intel XEON Prozessorgeneration 4679 Pins. Ein großer Teil der Anschlusspins wird allerdings durch die Versorgung belegt, bei 100 W Verlustleistung und Spannungen kleiner 1 V müssen über 100 A durch die Pins.

4.4 Burst Mode

Durch aufwendige statistische Erforschung von Speicherzugriffen wurde in den 1990er-Jahren bemerkt, dass eine Software oft auch die Daten der Nachfolgeadressen des gerade angeforderten Datums benötigt. Die Vorgehensweise darauf den Speicherzugriff auf den Hauptspeicher zu optimieren, hat sich zuerst im Zusammenhang mit der Einführung der Cache-Speicher durchgesetzt (siehe Kapitel 2.3.3).

Seit der Einführung der stark erhöhten Taktfrequenz innerhalb der CPU und langsamem Takt außerhalb werden Daten zwischen der CPU und der Umgebung nicht mehr einzeln übertragen sondern in Form von Bursts. Ein Burst ist eine Übertragung von gleich mehreren Datenwörtern direkt hintereinander, ohne jedes Mal die Adresse mit zu übertragen.

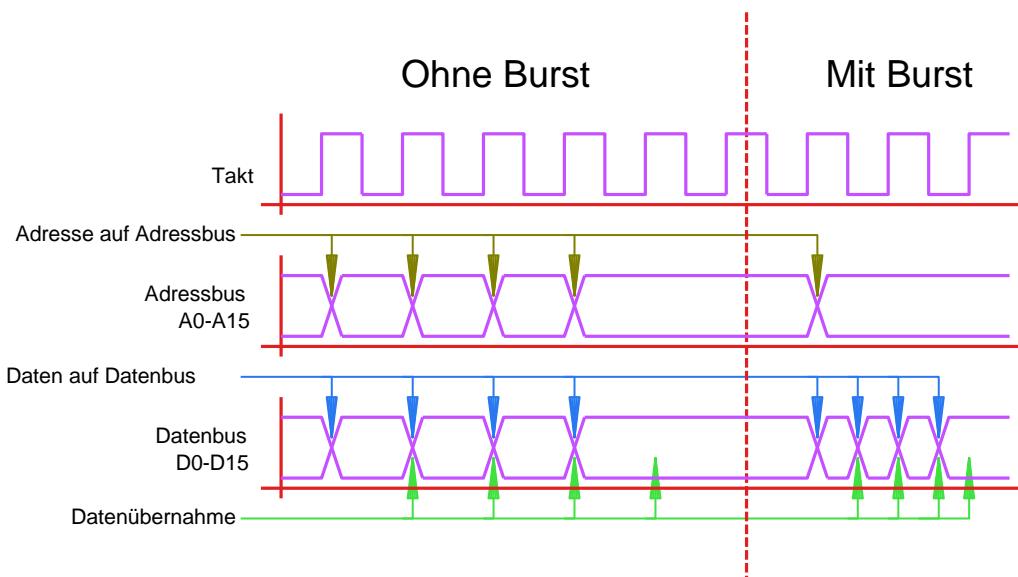


Abbildung 119 Burst

Die Datenbusbreiten der ersten CPUs mit Burstübertragung lagen bei 32 Bit (i486). Die Größe einer internen Cacheline lag damals bei 16 Bytes. Die ersten Burstlängen lagen deshalb bei 4 32-Bit-Zugriffen direkt hintereinander. Aktuelle Prozessoren mit 64-Bit-Bus haben interne Cachelines von 64 Bytes und übertragen Bursts jetzt mit 8 Zugriffen direkt hintereinander.

Beim Zugriff auf den DRAM-Arbeitsspeicher ist der Burstmodus ein großer Vorteil, da die verschachtelte RAS-CAS Adressierung eine viel kürzere Zugriffszeit nur auf direkt aufeinanderfolgende Adressen erlaubt (siehe Kapitel 5.2.2).

Der PCI-BUS kann Daten auch „burstmäßig“ übertragen, was hier ein starker Vorteil ist, da Daten- und Adressenleitungen wechselweise gemultiplexed werden. Die Burstlänge beim PCI-BUS ist nicht begrenzt, man kann mehrere Megabyte mit nur einer Adressangabe übertragen.

4.5 DMA

Falls von einem Gerät größere Datenmengen vom oder zum Arbeitsspeicher zu transferieren sind, müsste die CPU jedes einzelne Datum in einem Register kurz zwischenspeichern, um es dann gleich wieder auszugeben. Während dieser Zeit steht die CPU nicht für Rechenaufgaben zur Verfügung.

Dass diese Aufgabe, wenn an den Daten nichts zu verändern ist, nicht unbedingt von der CPU erledigt werden muss, wurde bei Computerarchitekturen schon sehr früh erkannt.

Dafür wurde kurz nach der 8-Bit-Ära ein zusätzlicher Baustein eingeführt, Der DMA-Controller (**Direct Memory Access**).

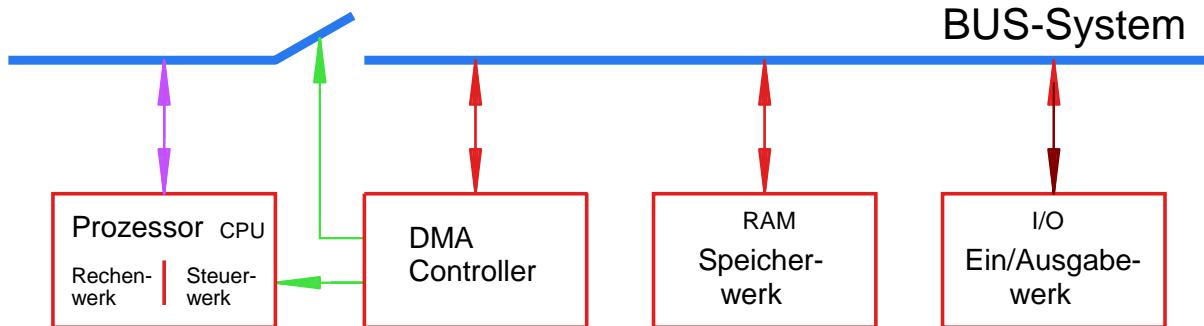


Abbildung 120 DMA-Zugriff

Die CPU überträgt dem DMA-Controller die Aufgabe, Daten vom Arbeitsspeicher in ein Peripheriegerät oder von Peripheriegerät zu Peripheriegerät oder von Arbeitsspeicher zu Arbeitsspeicher zu übertragen. Danach unterbricht der DMA-Controller die Verbindung der CPU zum Bussystem. Der DMA-Controller überträgt die notwendigen Daten und meldet über die DMA-Leitung, wann die Aufgabe erledigt ist. Danach übernimmt die CPU wieder den System-Bus.

Da aktuelle CPUs heutzutage über ein mehrstufiges Cachesystem On-Chip verfügen, kann das Programm auf der CPU ohne Zugriff auf dem Arbeitsspeicher eine Weile weiterlaufen. Das war bei Systemen ohne integrierten Cache früher nicht möglich.

In manchen Rechnersystemen kann es sein, dass mehrere Teilsysteme DMA-Fähigkeiten haben und selbst ihre Datenmengen ohne CPU transferieren können.

Manche Anwendungen sind ohne Nutzung eines DMA-Transfers nicht möglich:

- Beispiel: DVD-Brennen mit hohen Geschwindigkeiten.

4.6 MMU

Die **Memory Management Unit** ist aus Hardwaresicht eine nicht unbedingt notwendige Einheit. Sie ist nur vorhanden um dem Betriebssystem die Arbeit mit der Speicherverwaltung zu erleichtern. Moderne Betriebssysteme verwalten den verfügbaren Speicher in Pages. Die Page-Größen liegen üblicherweise zwischen 4Kbyte und mehreren Megabyte. Die MMU arbeitet mit den Cache-Controllern zusammen und markiert gerade nicht benötigten Speicher, der dann automatisch in den Hauptspeicher oder auf die Festplatte ausgelagert wird.

Die MMU gab es früher als Extra Baustein. Heutzutage ist die MMU in vielen CPUs integriert, allein schon deshalb, weil sie zwischen den Caches und dem Prozessorkern liegen muss.

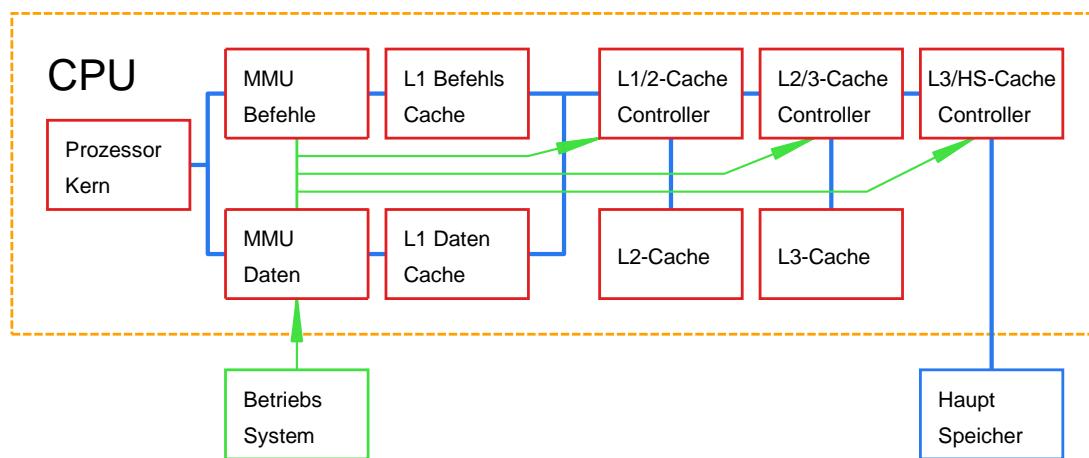


Abbildung 121 CPU mit Memory Management Unit

Bei modernen Betriebssystemen wird die MMU für mehrere Aufgaben genutzt:

- Auslagern von zur Zeit nicht benötigtem Speicher
- Bereitstellen von angefordertem, aber noch nicht genutztem Speicher.
- Isolation von Prozessen untereinander
- Specherschutz zwischen Programmen untereinander
- Specherschutz zwischen Programmen und Betriebssystem

Im Embedded-Bereich werden Mikrokontroller sehr oft ohne Betriebssystem betrieben. Die MMU wird dann nicht benötigt und kann abgeschaltet werden, bzw. sie wird erst gar nicht aktiviert.

5 Speicher

5.1 Speicherhierarchie

Der in Computern verbaute Speicher hat unterschiedliche Einsatzgebiete, je nach Arbeitsgeschwindigkeit und Größe.

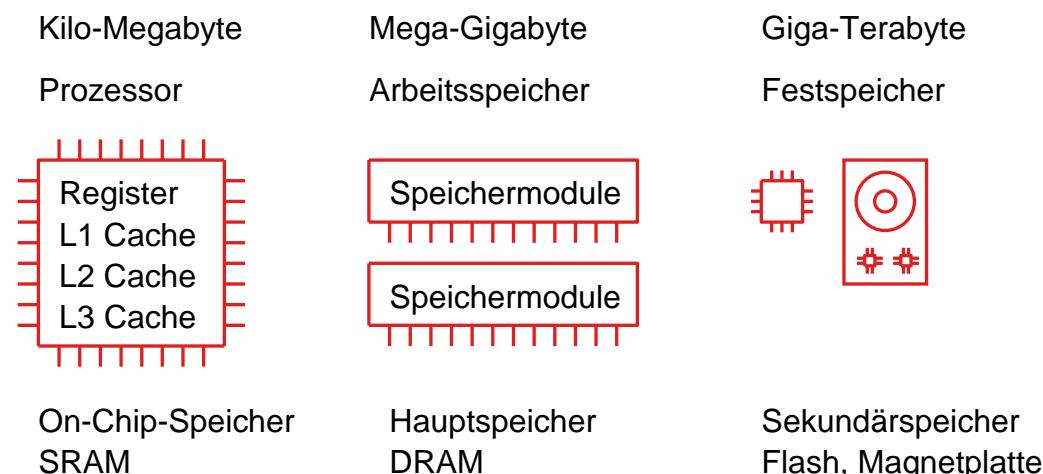


Abbildung 122 Speicherarten

Zugriffszeiten:

- | | | |
|-------------------|-----------------|-------------|
| • Register | ~0,25-5 ns | 2 – 4 GHz |
| • Cache | ~0,5-5 ns | 0,2 – 2 GHz |
| • Arbeitsspeicher | ~40-60 ns | 16-25 MHz |
| • Flash | ~30-250 μ s | 4 – 33 kHz |
| • Magnetplatten | ~3-20 ms | 50-300 Hz |

Diese Zugriffszeiten sind nur ungefähre Angaben und sie werden ständig geringer. Auch die Art des Zugriffs bestimmt zusätzlich noch die Zugriffszeit. Die meisten Speicherarten lassen sich wesentlich schneller lesen/schreiben, wenn mehrfache Zugriffe auf aufeinanderfolgende Speicheradressen stattfinden.

Speichertechnologien werden unterschieden in:

1. Flüchtigen Speicher z.B. SRAM, DRAM (Informationen gehen nach Ausschalten der Versorgungsspannung verloren)
2. Nichtflüchtigen Speicher z.B. PROM, EPROM, FLASH (Informationen bleiben ohne Versorgungsspannung erhalten)

5.2 Flüchtige Speicher (RAM)

5.2.1 Statisch (SRAM)

Das Grundprinzip von SRAM-Speicher (Static Random Access Memory) basiert auf einem Flip-Flop pro Bit Speicherzelle. Eine typische Standard-SRAM-Zelle wird in der Regel aus 6 Einzeltransistoren erzeugt.

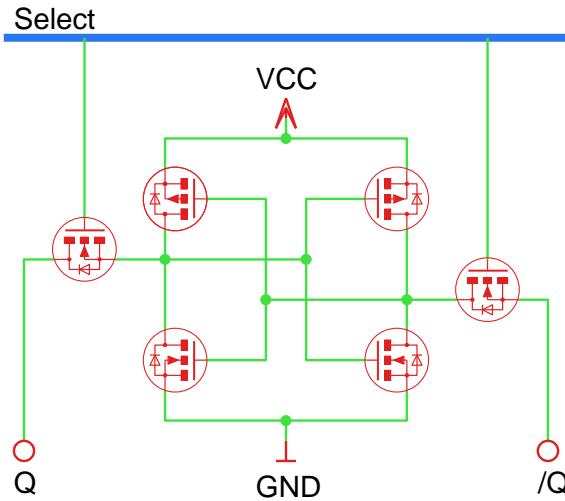


Abbildung 123 SRAM-MOSFET-Zelle

An den beiden Anschläßen Q und/oder Q-Nicht werden die Schreib-/Leseverstärker angeschlossen, mit denen die SRAM-Zelle auf einen der beiden stabilen Zustände gesetzt wird. Die Selectleitung dient zur Auswahl der Zelle. Die Selectleitungen der vielen SRAM-Zellen in einem Speicherarray werden mit einer Adressierungsmatrix über die Adressleitungen ausgewählt.

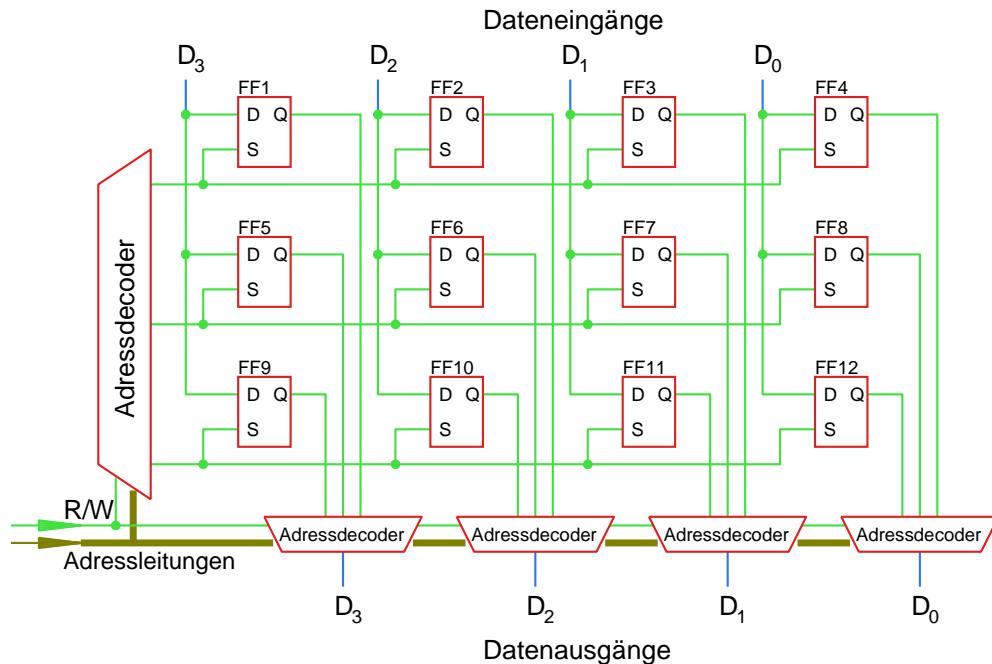


Abbildung 124 SRAM-Flip-Flop-Array

5.2.2 Dynamisch (DRAM)

5.2.2.1 Standard-DRAM

Das Grundprinzip von DRAM-Speicher (**Dynamic Random Access Memory**) basiert auf einem kleinen Kondensator, in dem der Wert eines Bit in Form einer elektrischen Ladung gespeichert wird. Der Vorteil gegenüber SRAM ist, dass nur ein einziger Transistor benötigt wird, um ein Bit zu speichern.

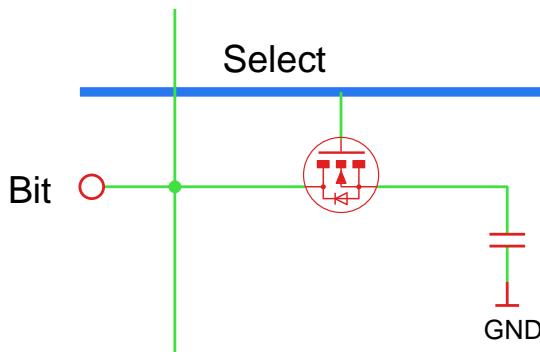
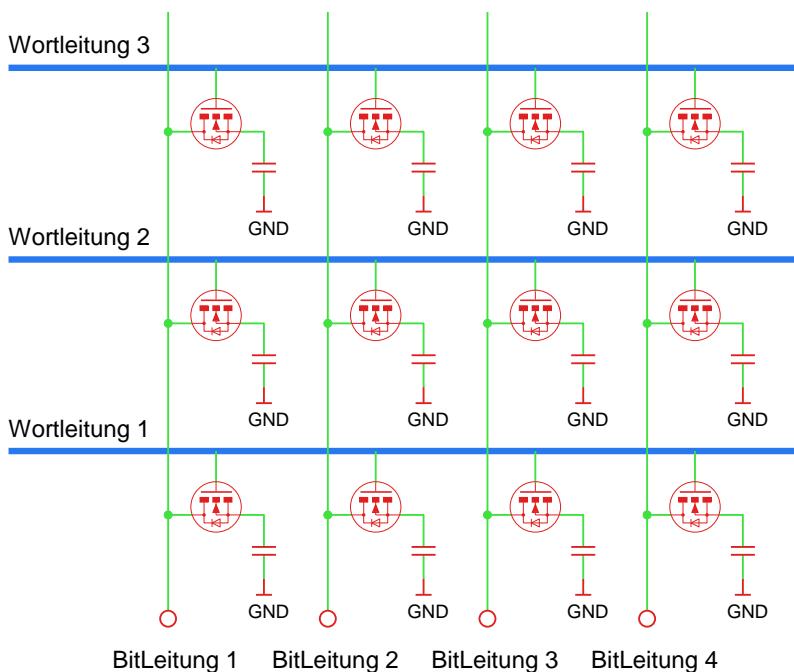


Abbildung 125 DRAM-MOSFET-Zelle

Die Größe des Kondensators einer DRAM-Zelle liegt bei heutigen Gigabit-DRAM-Speichern im aF-Bereich (Attofarad). Der Kondensator entlädt sich von selbst durch Leckströme innerhalb von einigen Millisekunden. Damit der Speicherinhalt nicht verloren geht, muss bei jedem Bit, bevor die Ladung verloren geht, der Inhalt ausgelesen, kurz zwischengespeichert und wieder zurückgeschrieben werden. Dieser Vorgang wird bei Computern Refresh genannt. Bei einem DRAM-Speicher wird dies immer zyklisch für den gesamten Speicher durchgeführt (Refresh-Zyklus). Bei frühen Computern wurde der Refresh durch die CPU durchgeführt, heutzutage wird die gesamte Refresh-Prozedur innerhalb der Speichermodule durchgeführt.



DRAM-Speicher werden gewöhnlich in einem zweidimensionalen Speicherfeld organisiert. Dabei werden die Wortleitungen und die Bitleitungen vom Adressdecoder in zwei Teile geteilt.

Abbildung 126 DRAM-Array

Die Adressierung des Speicherarrays wird dann in zwei Schritten durchgeführt. Die beiden Adresssteile werden in RAS (Row-Address-Select, Zeilenadresse) und CAS (Column-Address-Select, Spaltenadresse) unterteilt.

Die wahlfreie Zugriffszeit bei DRAM-Speichern liegt heute bei ca. 40-60 nS, das entspricht einer wahlfreien maximal möglichen Zugriffstaktfrequenz von 16-25 MHz. Wenn bei einem Adressierungsvorgang nur eine der beiden Adresssteile verändert wird, geht die Adressierung ungefähr doppelt so schnell -> 20-30 nS.

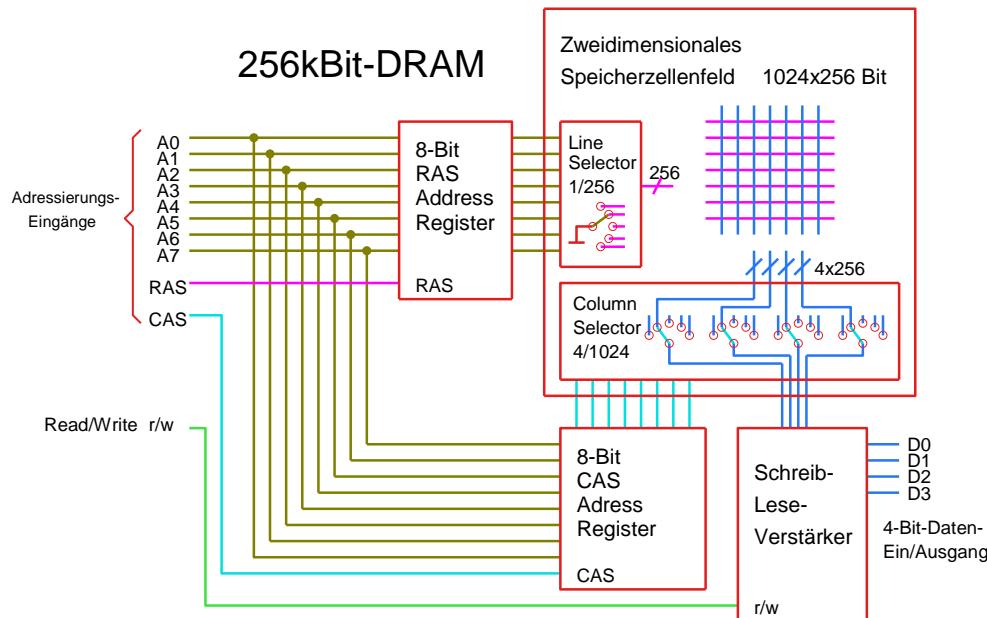


Abbildung 127 RAS-CAS-Adressierung

Je nachdem wie viele Datenleitungen bei dem DRAM-Baustein nach außen geführt sind, wird vom Schreib-Lese-Verstärker nur ein Teil des selektierten Datenworts vom CAS-Addressdecoder ausgewählt. Das war bei früheren Bausteinen oft nur ein einziges Bit. Bei den meisten RAM-Modulen werden dann bei mehreren DRAM-Bausteinen die Adressierungseingänge parallel geschaltet. Für jedes Bit des RAM-Moduls war dann ein eigener Baustein zuständig. Heutzutage sind bei aktuellen DRAM-Bausteinen meist 4-Bit herausgeführt.

Das Parallelschalten der Adressierungseingänge von vielen Bausteinen führt bei manchen Rechner-Designs zu einer Überlastung des Adressbusses. Das kann verhindert werden, indem man vor die Adressleitungen der Bausteine auf dem RAM-Modul einen zusätzlichen Buffer schaltet. Man spricht dann von Bufferd-RAM. Wenn zusätzlich noch ein Zwischenspeicher (Latch) für die Adresse enthalten ist, spricht man vom Registered-RAM.

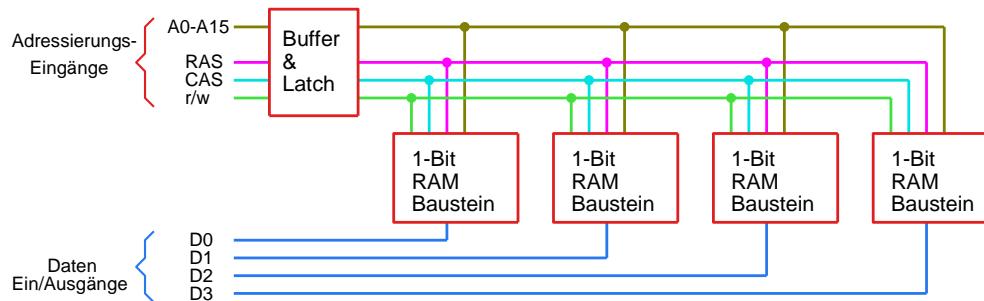


Abbildung 128 Buffered/Registered RAM

5.2.2.2 EDO

Da es während der Adressierungsphase eine gewisse Zeit dauert, bis die Daten am Ausgang anstehen, sind bis zum Ablauf dieser Adressierungszeit die Daten auf dem Datenbus ungültig. Erst danach stehen die Daten stabil an und können übernommen werden.

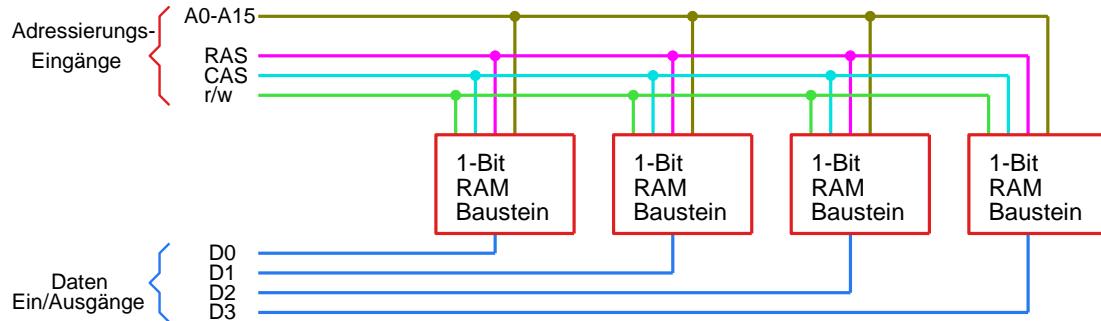


Abbildung 129 Fast Page Mode RAM

Um diese Wartezeit mit ungültigen Daten anderweitig nutzen zu können, wurde als nächster Entwicklungsschritt ein zusätzlicher Buffer mit Zwischenspeicher in die Datenausgänge der RAM-Bausteine eingefügt. Diese Technologie trägt den Namen EDO (Enhanced Dynamic Output).

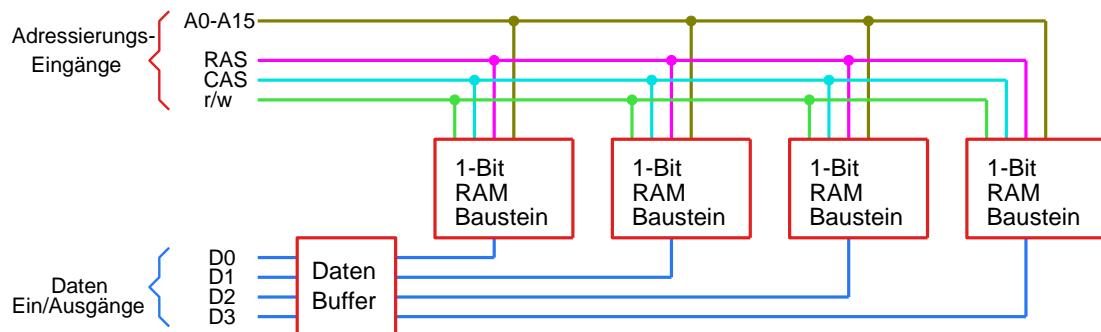


Abbildung 130 EDO-RAM

Während der Adressierungsphase ist im Datenbuffer noch das vorherige Datenwort gespeichert und liegt auf dem Datenbus. Damit wird eine verschachtelte Adressierung möglich. Während der Wartezeit der aktuellen Adressierung kann das vorherige Datenwort gelesen werden.

Der größte Nachteil beim EDO-RAM liegt in der immer noch asynchronen Arbeitsweise. Die Zeiten zwischen der RAS-CAS Adressierung und die Zeit, bis das Datum abrufbereit ist, folgen keinem festen Raster und sind bei jedem RAM-Baustein anders, d.h. der langsamste bestimmt die Schreib-/Leserate, und es muss beim Zugriff immer eine gewisse Sicherheitsreserve einbehalten werden, damit Schwankungen z.B. durch die Temperatur nicht zum Problem werden.

5.2.2.3 SDRAM

Die Einführung eines gemeinsamen Takts für den gesamten RAM-Baustein wurde kurz vor dem Jahr 2000 eingeführt. Ab jetzt beziehen sich alle Steuersignale auf das Taktsignal. Die Bezeichnung trägt jetzt den Namen SDRAM (Synchronous Dynamic Random Access Memory).

Fast alle modernen Prozessoren lesen und schreiben durch den zwischengeschalteten Cache den Speicher im Burst-Modus (siehe Kapitel 4.4), d.h., sie lesen immer mindestens 16 Bytes direkt hintereinander. Auf diese jetzt neue Arbeitsweise hin wurde der immer noch zu langsame Arbeitsspeicher optimiert.

Innerhalb der Speicherbausteine wurde alles daraufhin optimiert, Speicherzugriffe auf aufeinanderfolgende Adressen sehr schnell durchzuführen. Durch Pipelining, Vervielfachung der Speicherbänke und mehrere Buffer innerhalb des RAM-Bausteins, wurde ca. eine Verdopplung der Datenrate gegenüber EDO-RAM erreicht.

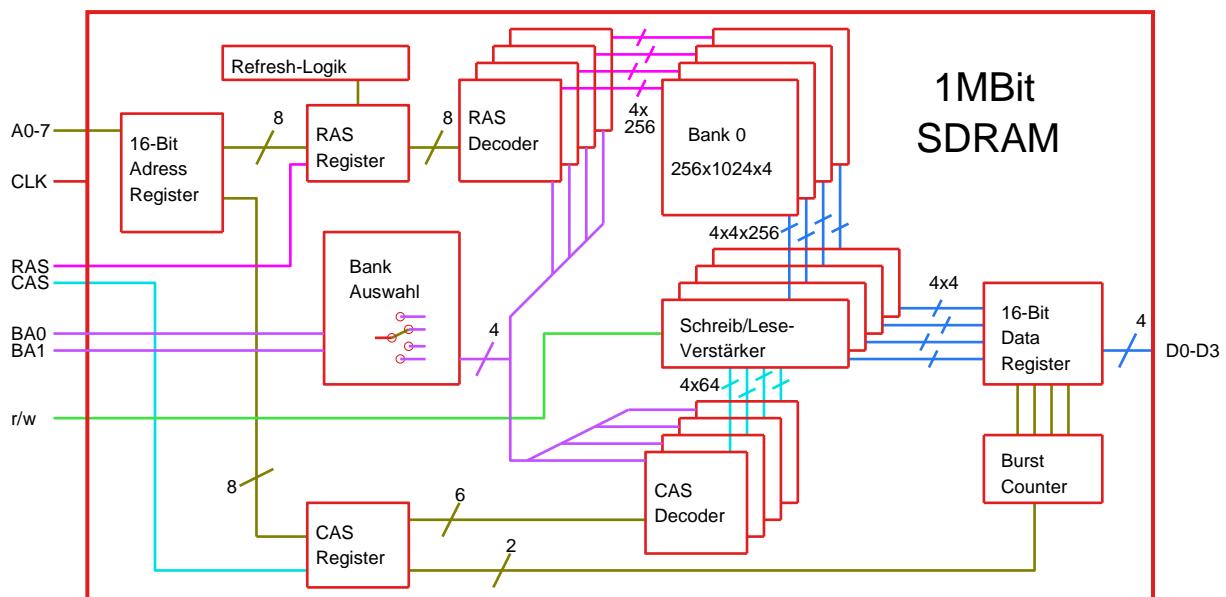


Abbildung 131 SDRAM

Eine zusätzliche Änderung gegenüber asynchronem RAM ist der jetzt direkte Zugriff auf die Bankauswahl des Moduls über 2 zusätzliche Bank-Select Leitungen (BA0, BA1). Damit ist eine freie Auswahl auf 4 getrennte Speicherbereiche möglich, auf die sehr schnell vom Speichercontroller der CPU aus zugegriffen werden kann ohne die RAS- oder CAS-Adresse zu ändern.

Die oft auftauchende Bezeichnung CL (CAS-Latency) gibt an, wieviele Takte man warten muss, bis das angeforderte Datum vom RAM bereitgestellt wird.

z.B. bei einer Übertragungsrate von 1333 MByte/s mit 64 Bit Busbreite liegt die Taktfrequenz des Busses bei $1333/8=166$ MHz, da 8 Byte gleichzeitig übertragen werden. Bei einer CL=10 muss der Speichercontroller 10 Takte warten bis das RAM-Modul bereit ist. Das entspricht dann einer maximalen wahlfreien Zugriffsrate von ca. 16,6 MHz.

5.2.2.4 DDR-SDRAM

Die synchrone Datenausgabe bei jeder 2. Taktflanke des SDRAMs wurde irgendwann wieder zu langsam. Der nächste Schritt erfolgte, indem man beim Taktsignal pro Taktperiode bei beiden Taktflanken ein Datum ausgibt. Die Bezeichnung heißt jetzt **DDR-SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory)**, siehe auch Kapitel 4.1.

5.2.2.5 DDR2/3/4

Bei den Nachfolgeversionen DDR 2/3/4 wurden die Anzahl der direkt ansteuerbaren Speicherbänke immer weiter erhöht. DDR2/3 -> 8 Bänke, DDR4 -> 16 Bänke. Dadurch kann die Zugriffszeit auf immer mehr Speicherbereiche sichergestellt werden, ohne dass die RAS- oder CAS Adressen geändert werden müssen. Auch die vorgehaltene Menge der Folgebytes (Prefetch) wurde immer weiter vergrößert, damit der Burstzugriff immer schneller erfolgen kann.

5.2.2.6 DDR5

Mit der Einführung von DDR5 ab ca. 2021 wurde das Speicherinterface nochmals stark verändert. Die Bankanzahl wurde auf 32 erhöht und es gibt jetzt pro Modul 2 Speicherkanäle, jedoch mit jeweils nur 32 Bit Datenbusbreite anstatt der vorherigen 64 Bit. Das Modul hat 2 getrennte Adressbusse mit zur Zeit (2024) jeweils 13 Bit. Es gibt jetzt auch die Möglichkeit, dass das Modul selbst intern schon eine ECC-Korrektur durchführen kann.

5.2.2.7 GDDR-SDRAM

Der Begriff **GDDR-SDRAM (Graphics Double Data Rate Synchronous Dynamic Random Access Memory)** taucht im Zusammenhang mit Grafikkarten auf.

Hierbei handelt es sich um die gleiche Speichertechnologie wie DDR, ihr Einsatz ist jedoch nicht dafür ausgelegt, diese auf Speichermodule zu setzen, sondern direkt auf die Leiterplatte am Einsatzort zu löten. Dadurch erreichen sie noch höhere Transferraten, da die Leiterbahnen kürzer und keine Steckkontakte dazwischen sind. GDDR-Speicher sind eher für höhere Transferraten optimiert und nicht wie CPU-Arbeitsspeicher auf geringe Zugriffszeit.

5.2.2.8 Fazit

Die Zugriffszeit des ersten wahlfreien Zugriffs bei DRAM-Speichern liegt heute (2024) immer noch bei ca. 40-60 nS, das entspricht einer wahlfreien maximal möglichen RAM-Zugriffstaktfrequenz von 16-25 MHz. Bei einer Adressänderung innerhalb derselben RAS/CAS-Page und weiteren Zugriffen auf direkte Folgeadressen sind zurzeit unter 4 nS möglich, das entspricht einer maximalen Zugriffstaktfrequenz von ca. 200-300 MHz. Das ist für Computer mit CPU-Taktfrequenzen im GHz-Bereich trotzdem immer noch viel zu langsam. Um dieses Problem zu umgehen, wird zwischen dem eigentlichen Arbeitsspeicher und der CPU schneller Cache-Speicher dazwischengeschaltet (siehe Kapitel 2.3.3).

5.3 Nichtflüchtige Speicher (ROM)

5.3.1 Allgemeines

5.3.1.1 Maskenprogrammiert (Fuse)

Die klassischen Speicher der ersten Computer enthielten viele ROM-Bausteine (Read Only Memory). Der Dateninhalt der ROM-Bausteine wird bereits bei der Herstellung des Halbleiterchips festgelegt. Diese Art der Speicherung ist nur bei sehr hohen Stückzahlen sinnvoll. Mittlerweile ist diese Art fast vollständig ausgestorben. Prinzipiell werden die Mikrocode-Programme innerhalb einer CPU auch heute noch maskenprogrammiert hergestellt. Der große Vorteil von maskenprogrammiertem Speicher ist seine relativ kurze Zugriffszeit.

5.3.1.2 Elektrisch Programmierbar

Bei elektrisch programmierbaren nichtflüchtigen Speichern wird die Information eines Bits in einem Floating-Gate eines Feldeffekttransistors gespeichert. Das Floating-Gate ist mittels Glas-Elektroden vom Rest isoliert und elektrisch nicht angeschlossen.

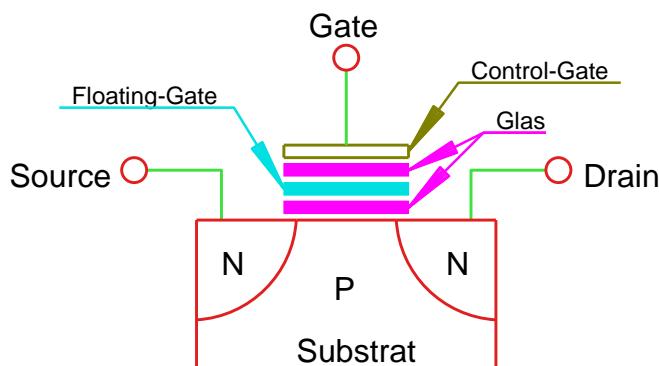


Abbildung 132 Floating Gate MOSFET

Schreiben

Um Elektronen auf das Floating-Gate aufzubringen, wird an das Control-Gate und den Drain Anschluss eine höhere Spannung (ca. 12-25 V) angelegt, Source liegt auf 0 V. Der Transistor wird leitend, und es fließt ein hoher Strom zwischen Source und Drain. Die Elektronen wandern dann mittels des quantenmechanischen Tunneleffekts durch die untere ca. 50 nm dicke Glas-Elektrode auf das Floating-Gate. Sie bleiben dort auch nach Abschalten der Spannung.

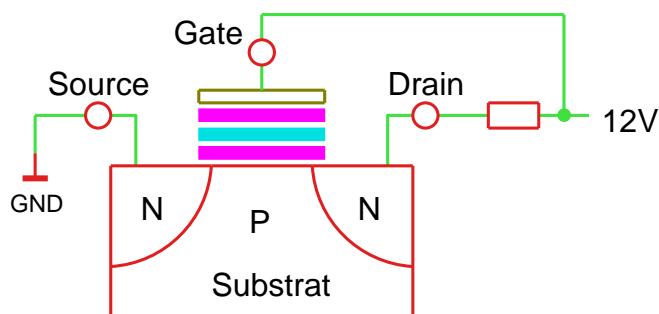


Abbildung 133 Floating Gate Schreiben

Lesen 0

Gelesen wird auch über das Control-Gate, jedoch mit einer wesentlich kleineren Spannung. Bei Lesen verhält sich der Transistor bei geladenem Floating-Gate anders als beim Ungeladenen.

Wenn das Floating-Gate vorher nicht mittels einer höheren Spannung geladen wurde, verhält sich der Transistor bei Ansteuerung über das Control-Gate wie ein normaler MOSFET. Der Transistor steuert durch und am Ausgang zum Leseverstärker liegen 0 V.

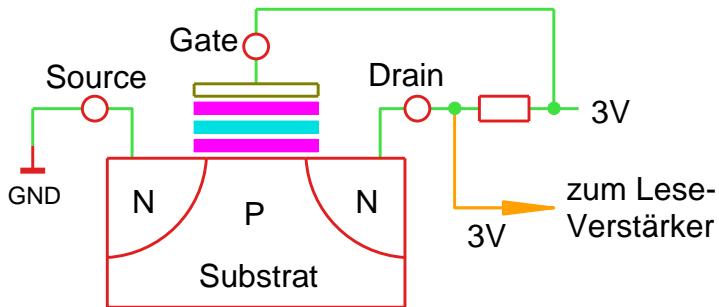


Abbildung 134 Floating Gate Lesen 0

Lesen 1

Das geladene Floating-Gate verhindert durch seine Ladung ein Durchsteuern der Drain-Source-Strecke über das Control-Gate. Der Transistor ist gesperrt und am Ausgang zum Leseverstärker liegen 3 V.

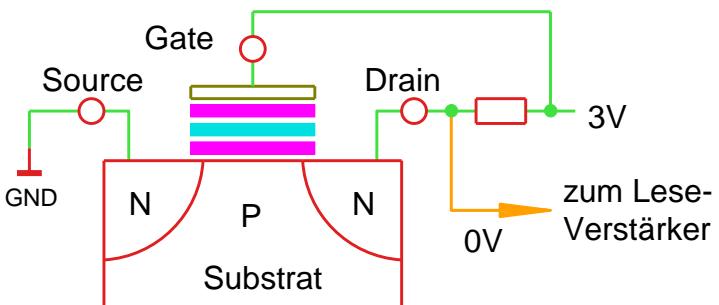


Abbildung 135 Floating Gate Lesen 1

5.3.2 EPROM

Der erste (ab ca. 1970) kommerzielle nichtflüchtige löschbare Speicher war das EPROM (**Erasable Programmable Read Only Memory**). EPROMs können nach dem elektrischen Beschreiben später mit UV-Licht wieder gelöscht werden (Löschezeit üblicherweise ca. 20 min). Nach einigen 100 Löschvorgängen ist die Kristallstruktur durch die harte UV-Strahlung zerstört, und der Speicherbaustein ist defekt.

Damit das UV-Licht an den Chip herankommt, ist in das IC ein Fenster aus Quarzglas (normales Glas ist nicht UV-durchlässig) eingebaut. Das Gehäuse ist deshalb immer aus relativ teurer Keramik. Die Wellenlänge der verwendeten UV-Strahlung liegt bei 254 nm und ist sehr gefährlich für die Augen (auch bei sehr kurzem Einwirken) und für die Haut.

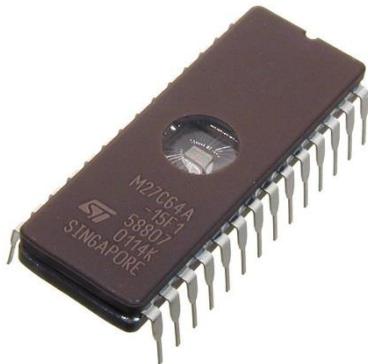


Abbildung 136 EPROM mit Fenster

Das eintreffende UV-Licht entfernt die auf das Floating-Gate aufgebrachte Ladung mittels Photoeffekt und löscht damit die gespeicherte Information auf dem gesamten Chip.

EPROMs haben üblicherweise eine 8-Bit-Datenbusbreite und übliche Größen liegen zwischen 64 kBit und 8 Mbit.

EPROMs gibt es mit derselben Technologie auch ohne Fenster, wodurch sie dann zum PROM (**Programmable Read Only Memory**) werden und nach einmaligem Programmieren nicht mehr löschar sind.

Die Typenbezeichnungen von EPROMs beginnen üblicherweise mit 27, die nächsten Ziffern geben normalerweise die Speichergröße in Kilobit an z.B 27256 für 256kByte= 32 kByte.

5.3.3 EEPROM

5.3.3.1 Allgemeines

Das EEPROM oder E²PROM (Electrically Erasable Programmable Read Only Memory) war der erste nichtflüchtige elektrisch löschbare Baustein auf dem Markt. Im Vergleich zum EPROM wird die höhere Schreib-Spannung (ca. 15 V) für das Floating-Gate intern erzeugt und muss nicht von außen angelegt werden.

Um die Ladung vom Floating-Gate wieder herunter zu bekommen, gibt es je nach Hersteller unterschiedliche Verfahren. Bei manchen geschieht das durch eine größere negative Spannung am Control-Gate oder durch eine positive Spannung am Drain Anschluss. Manche EEPROM-Technologien verwenden auch einen zweiten MOSFET für den Löschkvorgang FLOTOX-Zelle (FLoating Gate Tunnel Oxide).

5.3.3.2 Parallel

Parallele EEPROMs wurden meist als Ersatz für EPROMs verwendet. Die meisten Bausteine sind Pinkompatibel mit den Standard-EPROM-Typen und konnten diese direkt ersetzen. Die Typenbezeichnungen von parallelen EEPROMs beginnen meist mit 28 gefolgt von der Speichergröße in Kilobit z.B. 28512 für 512kBit=64 kByte. Die Speicherstellen von EEPROMs sind entweder einzeln oder in kleinen Blöcken lösbar. Parallele EEPROMs sind fast schon wieder ausgestorben und wurden durch Flashbausteine ersetzt.

5.3.3.3 Seriell

Serielle EEPROMs werden heute oft in Embedded-Systemen eingesetzt. Bei Ihnen kann fast jede Speicherstelle einzeln gelöscht werden, sie halten problemlos einige 100000 Lösch-Zyklen durch. Ein Speichervorgang (inklusive löschen) dauert in der Regel einige Millisekunden. Typische Bauformen für serielle EEPROMs sind 8-polige Gehäuse z.B. DIL8, SO8 oder MSO8.



Abbildung 137 Serielles 64kBit -EEPROM im SO8 Gehäuse

Auf DRAM-Speichermodulen wird oft ein serielles SPD-EEPROM (Serial Presence Detect) als Speicher für die Referenzdaten (Versorgungsspannung, Speicherkapazität, Timing, Refresh-Daten) eingesetzt.



Abbildung 138 SPD-EEPROM

Da manche Grenzdaten von DRAM-Modulen erst nach dem Zusammenbau aus mehreren Chips vorliegen, muss auf dem Modul ein beschreibbarer Festwertspeicher vorhanden sein.

5.3.4 Flash

5.3.4.1 Allgemeines

Auch beim Flash-Speicher wird die Information eines Bits in einem Floating-Gate eines Feldeffekttransistors gespeichert. Flash-Speicher verhält sich von der logischen Einteilung wie ein EEPROM, ist jedoch intern etwas anders aufgebaut und organisiert.

Angelehnt an die einfache Verschaltung von Transistoren als NAND und NOR Gatter,

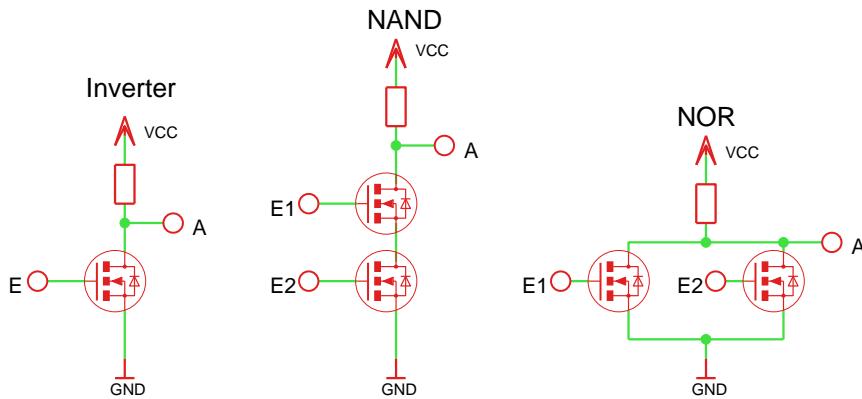


Abbildung 139 NOT NAND NOR

wird bei Flash Speichern grundsätzlich in **NOR-Flash** und **NAND-Flash** unterschieden:

5.3.4.2 NOR-Flash

Bei NOR-Flash sind die einzelnen Zellen als Matrix mit wahlfreiem Zugriff ausgeführt. Jede Zelle kann einzeln angesprochen werden.

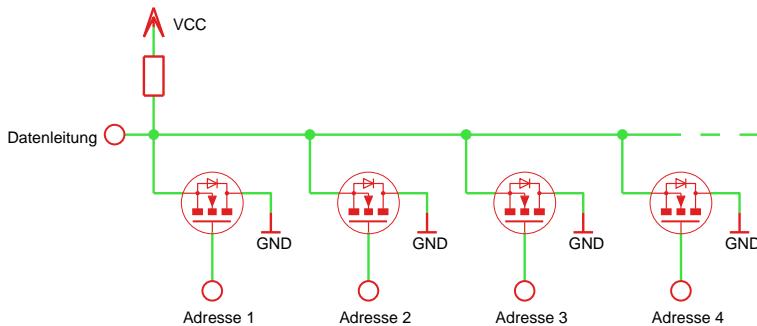


Abbildung 140 NOR-Flash

Mit der Ansteuerung der Zelle über richtige Adress-Wortleitung wird die Zelle direkt mit dem Daten-Ein/Ausgang verbunden. Die Zellen sind alle parallelgeschaltet, was jedoch auf einer 2-dimensionalen Chipfläche geometrisch leider relativ aufwendig ist. Eine Reihenschaltung wäre hier wesentlich einfacher zu realisieren.

5.3.4.3 NAND-Flash

Bei NAND-Flash werden immer Gruppen von Zellen hintereinander geschaltet. Wenn auf eine bestimmte Zelle zugegriffen werden soll, müssen alle Nachbarzellen in der Kette durchgesteuert werden, damit die Information der gewählten Zelle durchgereicht wird.

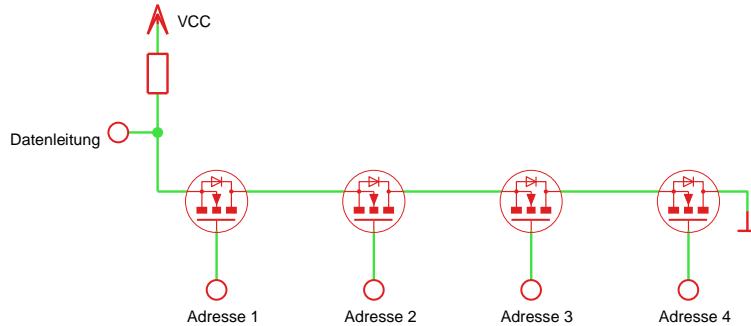


Abbildung 141 NAND-Flash

Das Signal der gewählten Zelle muss durch alle Nachbartransistoren hindurch, was die Störsicherheit/Zuverlässigkeit von NAND-Flash gegenüber NOR-Flash herabsetzt. Durch das Durchreichen der Information durch Zellen, die gerade nicht ausgewählt sind, gehen jedes Mal einige Elektronen auch der nicht selektierten Zellen verloren. Das führt dazu, dass die Haltbarkeit von NAND-Flash gegenüber NOR-Flash nur ca. 1/10 beträgt. Die belegte Chipfläche beträgt aber auch nur etwa 1/10.

5.3.4.4 Vergleich NAND-NOR-Flash

Eigenschaft	NOR	NAND
wahlfreier Zugriff	ja	nein
Löschgeschwindigkeit	langsam	schnell
Fläche/Zelle	groß	klein
Zuverlässigkeit	hoch	niedrig
Typische Verwendung	Programmspeicher innerhalb von Mikrocontrollern	USB-Sticks, Speicherkarten, Festplatten

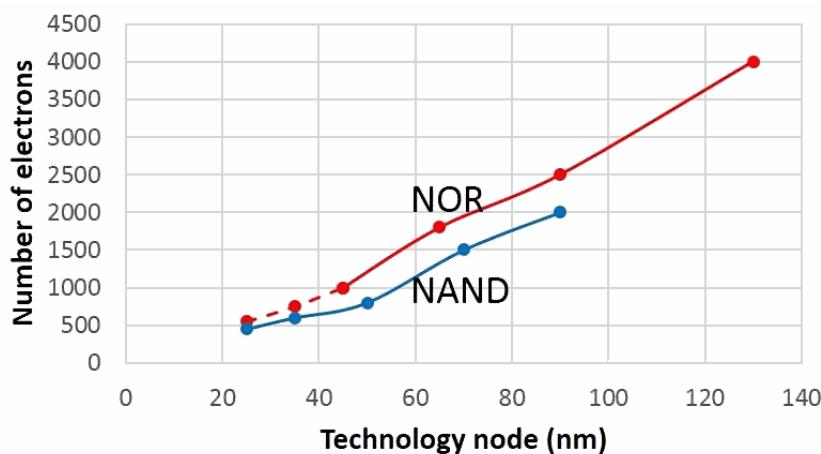


Abbildung 142 Elektronen pro Bit

5.3.4.5 SLC MLC TLC QLC

Um mehr Informationen pro Fläche speichern zu können, ist die Entwicklung ab ca. 2010 dazu übergegangen, mehr als eine 1 Bit Information in einer Flash-Zelle zu speichern.

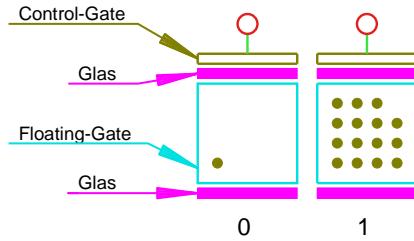


Abbildung 143 SLC-Zelle

Dazu wird die Menge, der in das Floating-Gate fließenden Elektronen, beim Befüllen entweder über die Zeit oder die angelegte Spannung gesteuert. Mit 3 Füllstufen kann man dann 2 Bit in einer Zelle speichern.

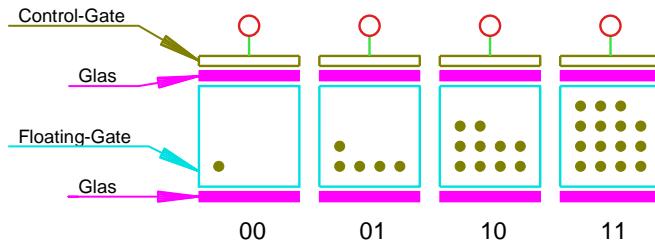


Abbildung 144 MLC-Zelle

Mit 7 Füllstufen kann man dann 3 Bit in einer Zelle speichern.

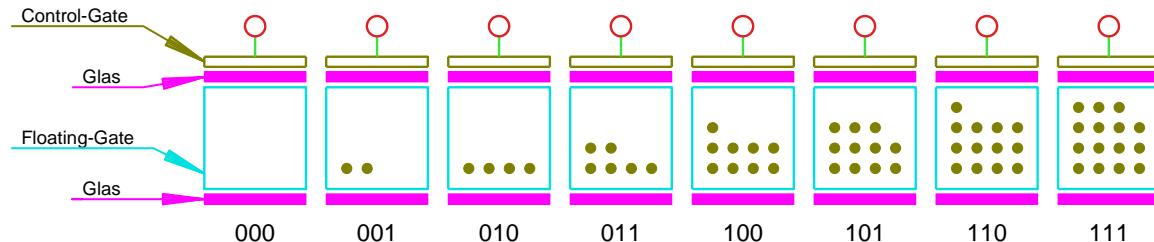


Abbildung 145 TLC-Zelle

Beim Auslesen muss eine saubere Trennung der Füllstufen unterschieden werden können. Da das nicht immer gelingt, ist ab TLC schon eine aufwendige Fehlerkorrektur notwendig. Die Anzahl der Löschzyklen sinkt mit mehr Stufen dadurch immer weiter.

	SLC	MLC	TLC	QLC
Bit/Zelle	1	2	3	4
Löschzyklen	100 k	3 k	1 k	100
Auslesezeit	ca. 25 µs	ca. 50 µs	Ca. 75 µs	>100 µs
Schreibzeit	200-300 µs	600-900 µs	900-1400 µs	>2 ms
Löschzeit	1-2 ms	3 ms	5 ms	>7 ms

Es ist auch möglich, in MLC- oder TLC Speicherstellen nur ein Bit zu speichern. Diese sogenannten pSLC-Zellen (pseudo-SLC) werden oft für industrielle Zwecke eingesetzt.

5.3.4.6 3D-NAND Flash

Die Art, wie man die Speicherstellen anordnet, hat sich in den letzten Jahren etwas verändert. Die Menge der unterscheidbaren Elektronen kann, ohne noch mehr Fehler zu verursachen, nicht mehr weiter verringert werden. Deshalb kann man die Integrationsdichte nicht mehr weiter steigern. Die einzige Möglichkeit noch mehr Bits in einem Baustein unterzubringen ist, sie in der 3. Dimension aufeinanderzustapeln.

Der erste Schritt ist hier das Floating Gate jetzt aus dem Material Siliziumnitrid herzustellen. Dieses Floating Gate wird dann als Charge Trap bezeichnet. Danach wird die Anordnung um 90° gedreht. Im Floating Gate können die Elektronen nicht von einem Bit zum anderen wandern, da das Material zwar Elektronen speichern kann aber keine Elektronen durchlässt (Nichtleiter).

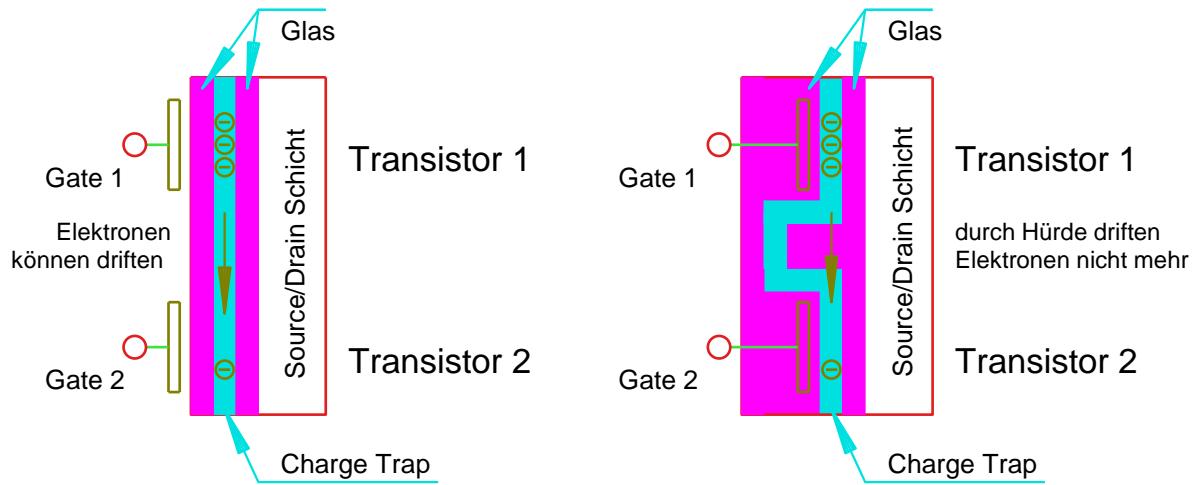


Abbildung 146 Charge Traps

Um das Driften der Elektronen von einem Transistor zum anderen zu verhindern, wird die Charge Trap geometrisch zusätzlich noch so verändert, dass die Elektronen den Umweg über die Hürde nicht mehr überwinden können.

Der Übergang vom Floating-Gate zur Charge Trap wurde auch schon bei 2D-Flash vorgenommen, weil dadurch die Lithographie bei der Fertigung stark vereinfacht wird. Die Floating Gates der einzelnen Transistoren in der Z-Achse werden hier als Charge Trap nicht mehr einzeln gefertigt, sondern sozusagen aus einem Stück hergestellt.

5.3.5 Modernere Speicherentwicklungen

5.3.5.1 Überblick

Neuere Entwicklungen zum Thema nichtflüchtige Speicher tragen paradoxe Weise fast alle die Bezeichnung RAM im Namen. Der Bezug auf die flüchtigen DRAM oder SRAM-Speicher wird hier nur über die Geschwindigkeit gemacht: Moderne nichtflüchtige Speicher sind deutlich schneller als FLASH oder EEPROM-Typen, und könnten teilweise wirklich als Ersatz für DRAM verwendet werden.

Die Information wird im Gegensatz zu FLASH oder EEPROMs nicht in einem elektrischen Feld, sondern mittels ferroelektrischer Materialien in einem Magnetfeld gespeichert.

Die Bausteine sind sehr robust und überleben über 10^{10} Schreib-Lese-Zyklen, womit sie praktisch unzerstörbar sind.

5.3.5.2 FRAM

FRAM-Speicherzellen (Ferroelectric Random Access Memory, auch FeRAM) sind physikalisch sehr ähnlich aufgebaut wie ein Feldeffekttransistor mit Floating-Gate. Der Speicher- und Löschkvorgang wird durch eine Polarisationsänderung in der ferroelektrischen Schicht realisiert. Bei FRAMs wird die Information beim Auslesen zerstört, und muss direkt danach wieder zurückgeschrieben werden. Das Verhalten ähnelt somit den früheren Ringkernspeichern aus den 1970er-Jahren (siehe Kapitel 2.3.6.3). Oberhalb der Curie-Temperatur des ferroelektrischen Materials sind FRAM nicht mehr funktionsfähig.

5.3.5.3 MRAM

Bei MRAM-Speicherbausteinen (Magnetoresistive Random Access Memory) gibt es einige unterschiedliche Verfahren um die Information eines Bits zu speichern. Prinzipiell ähneln sie alle dem GMR-Effekt, der bei Festplatten ausgenutzt wird, um den Lesekopf noch kleiner zu machen (siehe Kapitel 2.3.6.2). Der physikalische Aufbau einer MRAM-Zelle ist auch ähnlich wie beim Feldeffekttransistor mit Floating-Gate.

MRAM-Zellen sind sehr robust und unempfindlich gegenüber elektromagnetischer Strahlung und werden deshalb in der Luft- und Raumfahrt vermehrt eingesetzt.

2017 war der erste 1-GigaBit-Chip am Markt; Preisunterschied zu FLASH-Speicher ca. Faktor 50.

5.3.5.4 Fazit

In den letzten Jahrzehnten gab einige Weiterentwicklungen bei den FRAM- und MRAM-Speichern. Die Speicherdichte bleibt jedoch weit hinter der von Flash und DRAM-Speichern zurück. Die Hauptschwierigkeit liegt in der Integration der ferroelektrischen Materialien in die konventionelle Chipproduktion.

Richtig durchgesetzt hat sich ferroelektrischer Speicher noch nicht, da die Bausteine noch relativ teuer sind. Im Embedded-Bereich tauchen FRAM-Bausteine manchmal als schnellerer Ersatz für serielle EEPROMs auf, zu denen sie oft auch pinkompatibel sind.

5.4 Fehlerkorrektur

5.4.1 Softerror

In der Anfangszeit, als Speicherbausteine nur einige kByte RAM enthielten, ist es passiert, dass ein einzelnes Bit von alleine seinen Wert geändert hat. Als Ursache dafür wurde energiereiche Strahlung gefunden. Es handelt sich dabei um kosmische Strahlung oder ionisierende Strahlung radioaktiver Isotope, welche auch im Gehäusematerial enthalten sind. Die Menge an Gehäusematerial, die früher verbaut wurde, um z.B. 1 MByte Speicher „einzupacken“ war deutlich größer, da viele Bausteine auf einer Leiterplatte dafür benötigt wurden. In dem Gehäusematerial der Bausteine sind immer Isotope enthalten, die irgendwann zerfallen und energiereiche Teilchen können dann Speicherstellen beeinflussen.

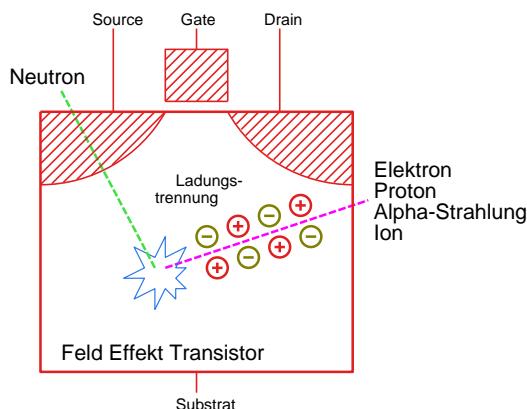


Abbildung 147 Softerror

Diese sogenannten „Softerrors“ sind im Laufe der Jahre immer weniger geworden (<0.3 Bit mal Stunde mal 10^{-15}), da die Vergussmaterialien immer besser wurden, und durch die gestiegene Integrationsdichte von Speicher und der aktuellen SMD-Technologie immer weniger Material verbaut wurde, welches radioaktive Isotope enthält.

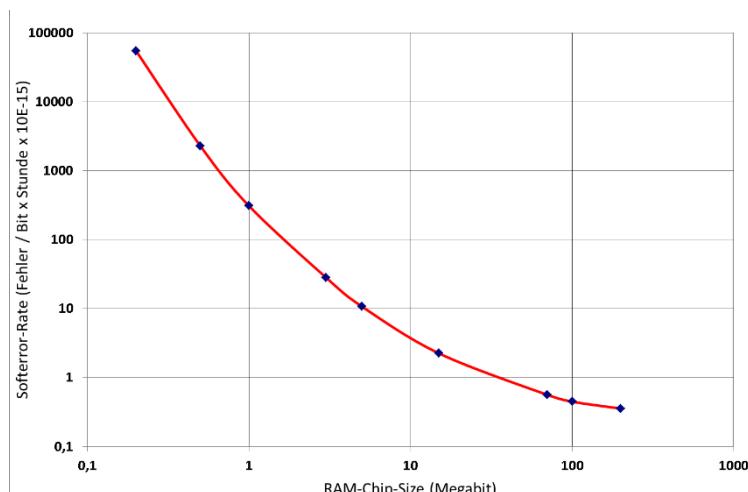


Abbildung 148 Soft Error Rate

Da in heutigen Computern viele Gigabyte an Speicher verbaut werden, ist das Softerror-Thema wieder aktuell, da durch die hohe Anzahl an Speicherstellen die Wahrscheinlichkeit für ein umgekipptes Bit wieder stören könnte.

5.4.2 Parity

Um die Beeinflussung der Softerrors in den Griff zu bekommen, wurde als erste Maßnahme die Parityprüfung zur Fehlererkennung eingeführt.

Für eine Gruppe von z.B. 8 Bit wurde ein 9. Bit als Parity-Bit hinzugefügt. Beim Abspeichern der 8-Bit-Gruppe wurde gezählt, ob die Summe der Bits in der Gruppe gerade oder ungerade ist. Das Ergebnis wurde dann in einem 9. Bit abgespeichert. Wenn das Byte dann wieder gelesen wird, wird überprüft, ob das Parity-Bit den korrekten Wert enthält. Wenn das nicht der Fall ist, wurde üblicherweise das gesamte System angehalten. Diese Prüfung wurde nicht von der CPU, sondern von einem zusätzlichen Parity-Check-Schaltkreis des Rechnersystems übernommen.

ASCII Buchstabe	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	Parity-Bit
1	0	0	1	1	0	0	0	1	0
2	0	0	1	1	0	0	1	0	0
3	0	0	1	1	0	0	1	1	1
4	0	0	1	1	0	1	0	0	0
5	0	0	1	1	0	1	0	1	1
6	0	0	1	1	0	1	1	0	1
7	0	0	1	1	0	1	1	1	0
8	0	0	1	1	1	0	0	0	0

Abbildung 149 Parity-Tabelle

Wenn zwei Bit in einem Datenwort umkippen, kann das die Parity-Prüfung nicht erkennen, und die Maschine würde mit falschen Daten weiterlaufen.

5.4.3 ECC

Die Parityprüfung kann nicht erkennen, welches Bit umgekippt ist. Als Weiterentwicklung wurde die ECC (Error Correction Code) - Prüfung eingeführt.

Als einfaches Beispiel wurde die Parity-Prüfung auf eine zweite Dimension ausgeweitet.

Dezimalwert	Bits								YP
170	1	0	1	0	1	0	1	0	1
85	0	1	0	1	0	1	0	1	1
20	0	0	0	1	0	1	0	0	1
250	1	1	1	1	1	0	1	0	1
127	0	1	1	1	1	1	1	1	0
128	1	0	0	0	0	0	0	0	0
33	0	0	1	0	0	0	0	1	1
175	1	0	1	0	1	1	1	1	1
<hr/>									
XP 1 0 0 1 1 1 1 1 1									

Abbildung 150 2D-Parity

Dezimalwert	Bits								YP
170	1	0	1	0	1	0	1	0	1
85	0	1	0	1	0	1	0	1	1
20	0	0	0	1	0	1	0	0	1
250	1	1	1	1	1	1	0	1	0
127	0	1	1	1	1	1	1	1	1
128	1	0	0	0	0	0	0	0	0
33	0	0	1	0	0	0	0	1	1
175	1	0	1	0	1	1	1	1	1
<hr/>									
XP 1 0 0 1 1 1 1 1 1									

Abbildung 151 ECC Fehlererkennung

Durch die zweidimensionale Verteilung der Parity-Bits wird es möglich, einen Bitfehler beim Lesen zu korrigieren. Falls in einem Array mehr als 1 Bit kippt, kann man jetzt auch dieses zumindest detektieren und entsprechend darauf reagieren.

6 Schnittstellen

6.1 Master - Slave

Die Begriffe **Master** und **Slave** sollen in allen Schnittstellenbezeichnungen nicht mehr verwendet werden und werden in Zukunft durch die Begriffe Server/Client ersetzt. Da hier die Hardwareschnittstellen betroffen sind, ist eine Änderung der Begriffe nicht so schnell und einfach möglich, weil alle Datenblätter aller betroffenen Bausteine nicht rückwirkend geändert werden können. Bei einer nur teilweisen Änderung entstehen Widersprüche und dadurch Fehler. Im Kapitel 6 sind die Begriffe Master und Slave deshalb noch nicht eliminiert worden.

6.2 Parallel

6.2.1 Centronics

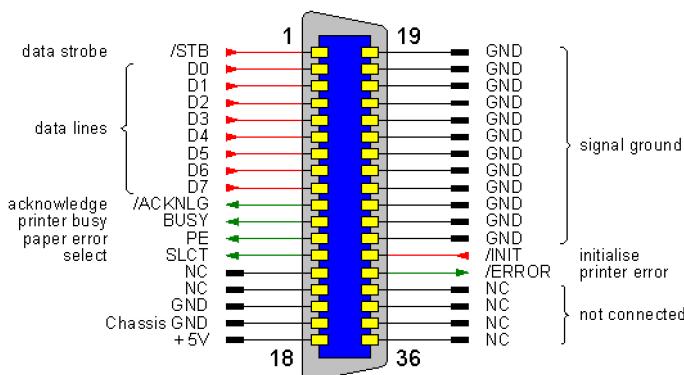
Die Centronics Schnittstelle war in den 1970er-Jahren eine der ersten externen Schnittstellen in einem Computer. Die ursprüngliche Anwendung lag in der Verbindung eines Computers mit einem Drucker. Die parallele Datenübergabe ist 8-Bit breit und war nur für die Richtung vom Computer zum Drucker vorgesehen. Die Datenübernahme erfolgt asynchron durch Handshakeleitungen.

Der anfangs benutzte große teure „Centronics-Stecker“ wurde später auf der Geräteseite durch DSUB ersetzt. Auf der Computerseite wurde der DSUB-Stecker von Anfang an eingesetzt.



Abbildung 152 Centronics-Kabel

Als in den 1980ern einige Peripheriegeräte (Scanner) aufkamen, für die die serielle Schnittstelle zu langsam war, wurde die Druckerschnittstelle zweckentfremdet. Da es mit der ursprünglichen Schnittstelle nicht möglich war, Daten zum Computer hin zu übertragen, wurden die 4 Leitungen **acknowledge**, **busy**, **paper error** und **select** zum 4 Bit breiten Empfangssignal umfunktioniert.



Die Nutzung der 4 Steuersignale als 4-Bit-Empfangsbus wird oft als Nibble-Modus bezeichnet.

Abbildung 153 Centronics Pinbelegung

Die Centronics Schnittstelle wurde einige Male weiterentwickelt. Der größte Schritt war 1994 mit grundsätzlichen elektrischen Definitionen und einigen Neuerungen. Die Schnittstelle trägt heute den offiziellen Namen IEEE1284.

In der Hauptsache wurde 1994 die Schnittstelle bidirektional gemacht, indem die 8-Bit-Datenleitungen für beide Richtungen genutzt wurden. Um noch schneller übertragen zu können, wurden die Verbindungskabel verbessert, indem die einzelnen Litzen verdrillt wurden (Twisted Pair), und es wurden definierte Busabschlüsse eingeführt.

Die Datenrichtung wird jetzt durch den Pegel der Strobe-Leitung definiert (Low: Schreibzugriff, High: Lesezugriff), welche vorher nur als Übernahmesignal für die Datensignale genutzt wurde.

Modus	Bedeutung	Features	Übertragungsrate
SPP	Standard Printer Port	herkömmliche Centronics Schnittstelle im Nibble-Modus	60 kByte/s
EPP	Enhanced Printer Port	bidirektionale Übertragung möglich	ca.180-500 kByte/s
ECP	Enhanced Capability Port	zusätzlicher DMA Betrieb möglich und FIFO-Buffer und RLE-Datenkompression	bis 2 MByte/s

Abbildung 154 Parallel-Port-Modi

Die Benutzung der Schnittstelle liegt heute fast nur noch im Bereich von Dongles für Kopierschutz und als einfache Programmierschnittstelle im Mikrocontroller-Bereich.

In der Übergangszeit zum USB wurden an der ECP-Schnittstelle oft Scanner und externe Festplatten betrieben.

6.2.2 IDE/ATA

Für die ersten Festplatten war als Einsteckkarte ein spezieller Festplattencontroller (ST506-Schnittstelle) notwendig, der in Prinzip eine Weiterentwicklung der damals üblichen Diskettenlaufwerke war (Shugart-Bus). Die Ansteuerung der Platte wurde vom Festplattencontroller übernommen. Auf der Festplatte waren nur eine einfache Motorsteuerung und die Elektronik für die Schreib-/Leseköpfe vorhanden. Verbunden wurden die beiden Einheiten über 2 Flachbandkabel.



Abbildung 155 Winchester Festplatte

In den 80er-Jahren wanderte fast die gesamte Controllerelektronik von der Einsteckkarte auf die Leiterplatte direkt auf der Festplatte -> Bezeichnung IDE (Integrated Drive Electronics). Die immer noch notwendige Erweiterungskarte enthielt nur noch einen Adressdecoder und zwei 8-Bit-Bustreiber für die 16 Datenleitungen. Die beiden Komponenten wurden dann mit einem 40-poligen Flachbandkabel verbunden. Western-Digital entwickelte in Kooperation mit einigen Festplattenherstellern diese Schnittstelle, und sie wurde 1986 als Quasi-Standard etabliert.

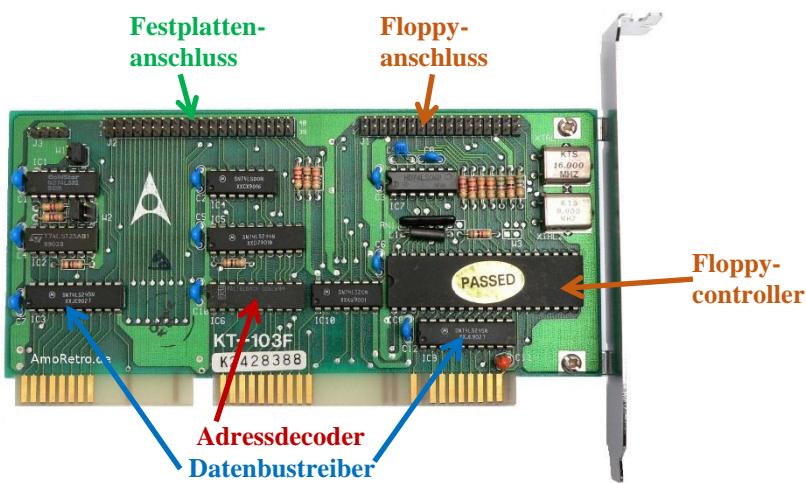


Abbildung 156 IDE-Controller

Später setzte sich die Bezeichnung ATA (Advanced Technology Attachment) für die Schnittstelle durch.

Die Schnittstelle wurde immer weiterentwickelt, und in der Version 4 ab ca. 1997 wurde die Bezeichnung ATAPI (Advanced Technology Attachment Packet Interface) eingeführt. Hier wurde im Prinzip nur das Softwareprotokoll erweitert, um SCSI-Kommandos über das ATA-Protokoll senden zu können. Das wurde notwendig, um die sich langsam verbreitenden CDROM-Laufwerke an dieselbe Schnittstelle anschließen zu können.

Ab Version 5 wurden diverse DMA-Transfers hinzugefügt, und durch die jetzt wesentlich schnellere Datenübertragung musste das Kabel verändert werden. Damit die Schnittstelle trotzdem rückwärtskompatibel bleibt, wurde der 40-polige Stecker beibehalten und ein 80-poliges Flachbandkabel mit halber Rasterbreite (0,635 mm) verwendet, bei dem jede 2. Ader auf Masse gelegt wurde. Dadurch wird ein Übersprechen zwischen benachbarten Datenleitungen verringert. Die Datenübertragungsrate von anfangs ca. 16 MByte/s steigerte sich auf bis zu 133 Mbyte/s.

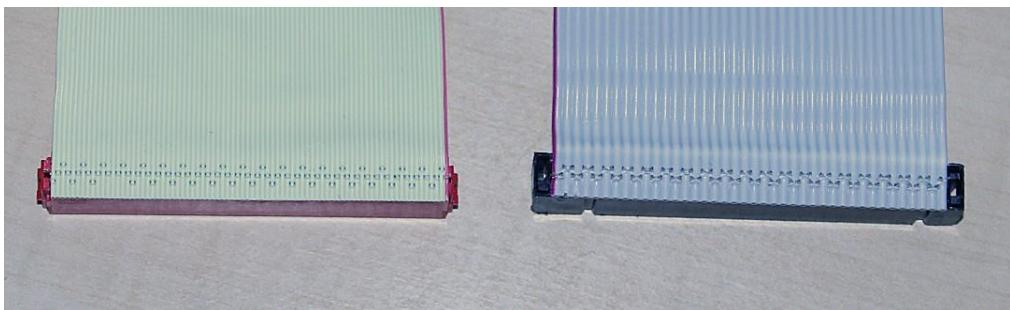


Abbildung 157 Vergleich 80/40 poliges Kabel

Bei dem 80-poligen Kabel wird ein spezieller Stecker mit einer 3. Pinreihe aufgecrimpt. Alle Anschlüsse dieser 3. mittleren Reihe sind im Stecker mit Masse verbunden.

Ab 2001 wurde die Bezeichnung auf PATA (**Parallel-ATA**) geändert zur besseren Unterscheidung von der SATA-Schnittstelle (**Seriell-ATA**).

Heutzutage gibt es die Parallel-ATA-Schnittstelle immer noch fast unverändert bei Compact-Flash-Karten für den industriellen Einsatz und bei professionellen digitalen Kameras.



Abbildung 158 Compact-Flash-Karte

Die Parallel-ATA-Schnittstelle wurde von 2000-2010 fast vollständig von der seriellen SATA-Schnittstelle abgelöst (siehe Kapitel 6.3.5).

6.2.3 SCSI

Im Gegensatz zur IDE-Schnittstelle wurde die parallele SCSI-Schnittstelle (**Small Computer System Interface**) von Anfang an elektrisch sehr robust und für eine sehr störsichere, zuverlässige Übertragung entwickelt (Shugart-Technologie ca. 1979). Die gesamte Verbindung ist im Prinzip ein komplettes Bussystem mit Adress- Daten- und Steuerleitungen. Anfangs hatte die Schnittstelle eine 8-Bit-Datenübertragung und eine 3-Bit-Adresse. Die Adressenleitungen sind nicht physikalisch vorhanden und werden hier über die Datenleitungen gesendet mit einem relativ aufwendigen Busprotokoll.

Ein auffälliges Element beim SCSI-Bus ist die Terminierung an beiden Enden des Busses.

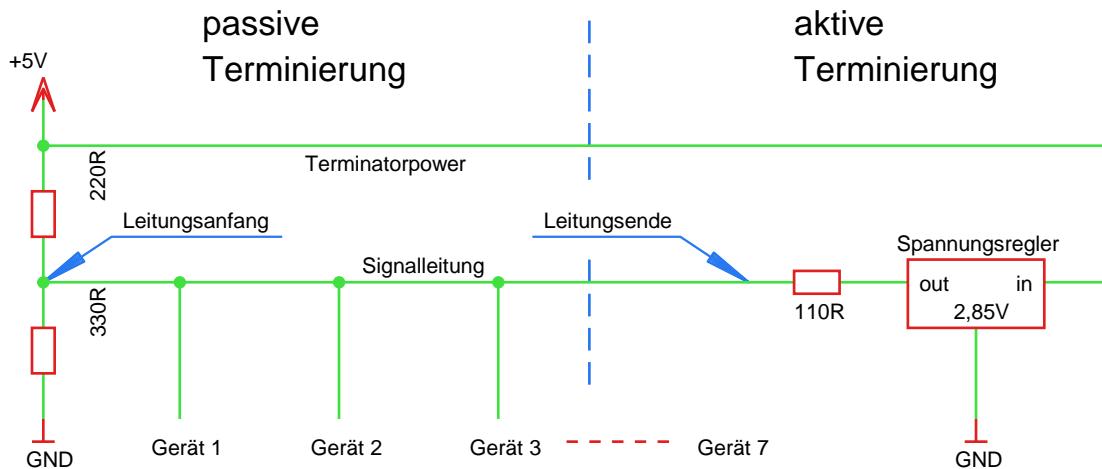


Abbildung 159 SCSI Terminierung

Ein typisches Flachbandkabel hat einen Wellenwiderstand von ca. $150\ \Omega$. Damit keine Reflexionen an den Leitungsenden stattfinden, ist eine Terminierung nahe $150\ \Omega$ notwendig. Die Terminierung muss auf beiden Seiten erfolgen und kann aktiv oder passiv sein. Passiv: $220\ \Omega$ parallel $330\ \Omega = 132\ \Omega$, aktiv=110 Ω .

Paralleles SCSI wurde ständig weiterentwickelt von anfangs 8 Bit auf 16 Bit und durch mehrere Taktfrequenzerhöhungen und Einführen von Burstmode und DDR (**Double Data Rate**) von anfangs 5 MByte/s auf bis zu 320 MByte/s.

Paralleles SCSI wurde früher oft im professionellen Server- und Workstationbereich eingesetzt. Heutzutage ist paralleles SCSI fast ausgestorben und wurde im professionellen Serverbereich durch SAS (**Serial Attached SCSI**) ersetzt.

In allen anderen Bereichen wurde SCSI durch SATA verdrängt, da dieses Bussystem auch von Grund auf neu entwickelt wurde und nicht wie bei IDE sich durch ständige Evolution in eine Sackgasse manövriert hat.

Als Softwareprotokoll wurden SCSI-Kommandos ca. 1997 in die ATA Schnittstelle übernommen und sind heutzutage in die Controller fast aller Speichermedien integriert (auch USB-Speichersticks).

6.3 Seriell

6.3.1 Allgemeines

6.3.1.1 Begriffe

Im Gegensatz zu parallelen Verbindungen mit vielen Datenleitungen wird bei seriellen prinzipiell nur eine einzige Leitung für die Datenübertragung verwendet.

Folgende Begriffe haben sich durchgesetzt:

Begriff	Bedeutung
Synchron	mit Taktleitung
Asynchron	ohne Taktleitung
Full Duplex	Eine Daten-Leitung pro Richtung
Halb Duplex	Eine Daten-Leitung für beide Richtungen
Asymmetrisch	Eine Leitung für die Übertragung
Symmetrisch	Zwei Leitungen für störsichere Übertragung

Ein weiterer Begriff taucht gelegentlich bei seriellen Schnittstellen auf. Manche Entwickler nutzen die Leitungen, die für eine serielle Übertragung gedacht waren als einfache selbstdefinierte proprietäre I/O-Signale. Für diese Zweckentfremdung hat sich der Begriff **Bit-Bang-Modus** durchgesetzt.

6.3.1.2 Synchron

Für eine synchrone Übertragung mit Taktleitung ist normalerweise eine Synchronisation über eine weitere Leitung nötig, um dem Empfänger zu signalisieren, wann die Übertragung eines Wortes beginnt bzw. endet.

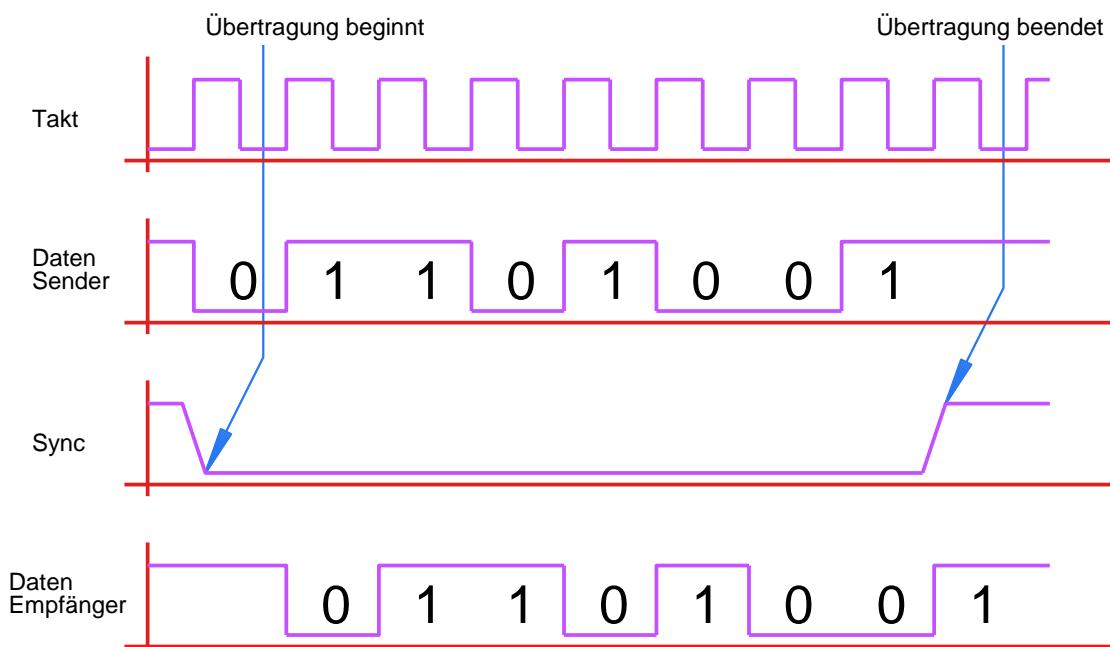


Abbildung 160 Synchrone serielle Übertragung

6.3.1.3 Asynchron

Für eine asynchrone Übertragung ohne Takteleitung müssen Sender und Empfänger vor der Übertragung wissen, wie lange ein Bit dauert. Um die Bits wortmäßig zu synchronisieren, werden zu den Daten zusätzlich noch Start- und Stoppbits hinzugefügt.

Für sichere Übertragungen ist es ratsam, nicht nur beim Wortanfang zu synchronisieren, sondern bei jeder Flanke. Damit kann auch bei einer leicht abweichenden Baudate zwischen Sender und Empfänger besser synchronisiert werden.

Nur die asynchrone, asymmetrische Schnittstelle kommt wirklich mit nur einer einzigen Leitung und Masse aus.

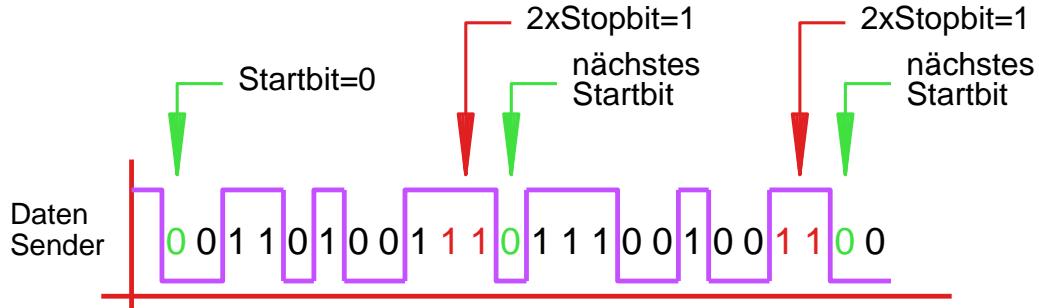


Abbildung 161 Asynchron

6.3.1.4 Fullduplex

Bei einer Fullduplex-Übertragung gibt es für jede Datenrichtung eine eigene Leitung. Theoretisch ist damit die doppelte Übertragungsrate möglich, wenn gleichzeitig gesendet und empfangen wird.

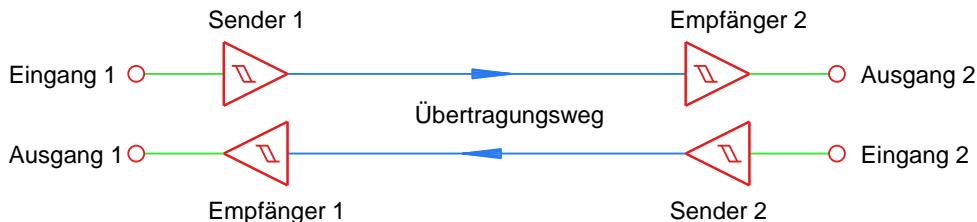


Abbildung 162 Fullduplex

6.3.1.5 Halbduplex

Bei einer Halbduplex-Übertragung wird eine Datenleitung für beide Richtungen verwendet. Vorher muss mittels eines Protokolls vereinbart werden, wer wann auf der Leitung sendet bzw. empfängt, damit nicht zwei Sender gleichzeitig senden.

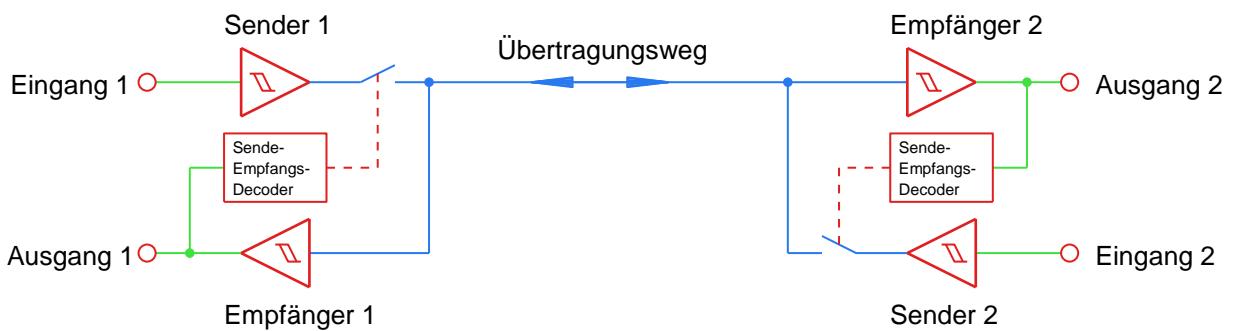


Abbildung 163 Halbduplex

6.3.1.6 Asymmetrisch

Bei asymmetrischen Übertragungen wird für jede Übertragungsfunktion genau eine elektrische Leitung benötigt.

6.3.1.7 Symmetrisch

Für eine symmetrische Übertragung wird jede Datenübertragungsleitung elektrisch doppelt ausgeführt. Jede der beiden Leitungen führt immer das invertierte Signal der anderen. Auf der Empfängerseite sitzt ein analoger Differenzverstärker, welcher die beiden Signale voneinander subtrahiert. Aus der Differenz der beiden Signale wird dann das Eingangssignal mit doppelter Amplitude zurückgewonnen.

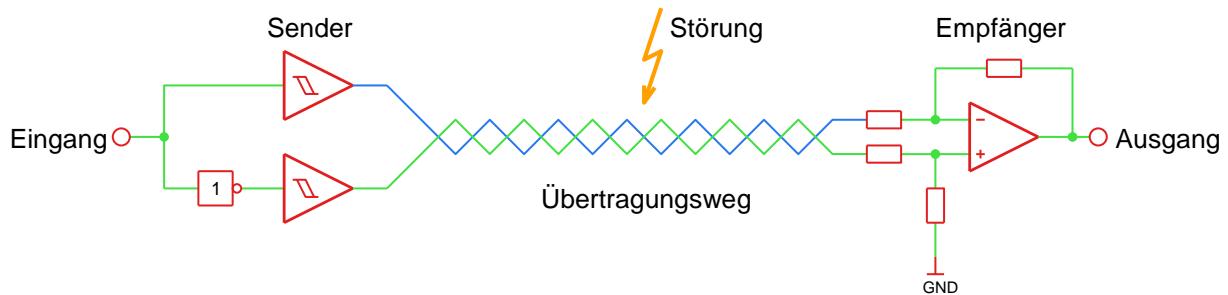


Abbildung 164 Differenzielles Paar

Als zusätzliche Maßnahme wird das differenzielle Paar miteinander verdrillt (Twisted-Pair). Dadurch werden störende elektromagnetische Einflüsse von außen möglichst ferngehalten.

Störungen treten normalerweise als Gleichtaktstörungen auf, d.h., die Störung wird auf beiden Übertragungsleitungen eingestrahlt und wird dann automatisch durch die Differenzbildung auf der Empfängerseite entfernt.

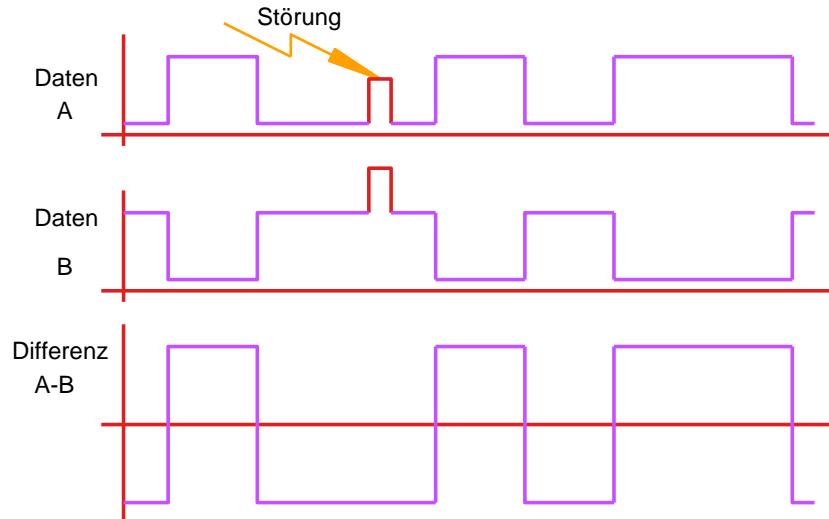


Abbildung 165 Differenzbildung

Durch die Differenzbildung erhält man auf der Empfängerseite die doppelte Amplitude. Dadurch entsteht nochmals ein zusätzlicher höherer Störabstand gegenüber der unsymmetrischen Übertragung.

6.3.1.8 Taktrückgewinnung

Bei modernen seriellen Übertragungen wird der Takt trotz synchroner Übertragung nicht mehr physikalisch auf einer eigenen Leitung übertragen. Damit der Empfänger trotzdem einen Takt erhält, wird dieser aus den Nutzdaten der Datenleitung zurückgewonnen.

Damit dies möglich ist, werden die Daten auf der Datenleitung nicht roh gesendet, sondern sie werden dafür in ein anderes Datenformat umcodiert, z.B. in den Manchester-Code. Hier wird z.B. eine logische 1 in einen 0-1 Übergang und eine logische 0 in einen 1-0 Übergang codiert.

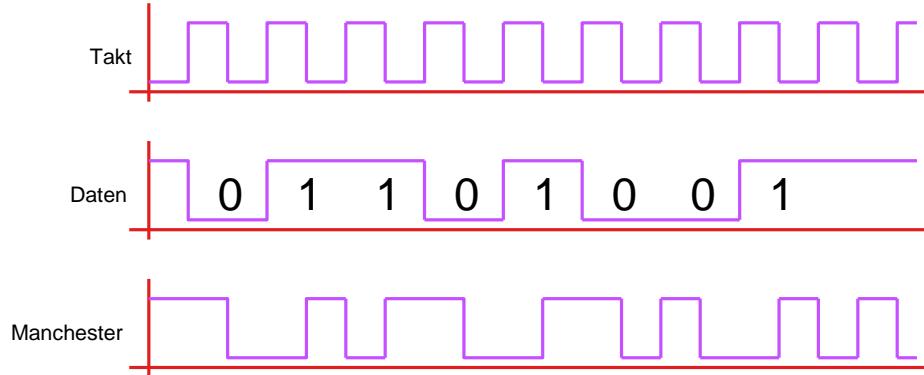


Abbildung 166 Manchester-Code

Mittels einer PLL-Schaltung kann wegen der jetzt permanenten Pegelwechsel direkt der Takt aus den Datenbits wieder zurückgewonnen werden.

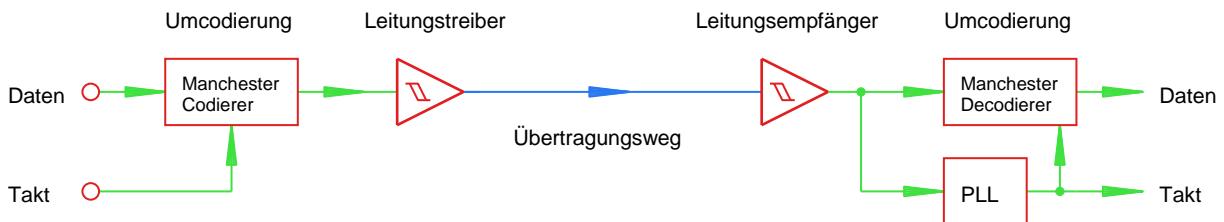


Abbildung 167 Taktrückgewinnung mit PLL

Die Methode der Taktrückgewinnung mit Manchestercode soll hier nur als einfaches Beispiel dienen. In der Praxis werden noch weitere Möglichkeiten zur Taktrückgewinnung eingesetzt. Wichtige Stichwörter hierzu:

- Scrambler
- RZ-Code (**R**eturn to **Z**ero)
- Biphase-Code
- 4B5B-Code

6.3.1.9 Kombination

Man kann alle Methoden serieller Verbindungen auch beliebig miteinander kombinieren. Für eine serielle, synchrone, symmetrische, fullduplex Übertragung sind dann insgesamt 10 Leitungen notwendig (2x2xData, 2x2xSync, 2xClk) + GND-Bezugspotential.

6.3.2 RS232/RS485/RS422

6.3.2.1 Grundsätzliches

RS232 (Recommended Standard 232) oder EIA232 (Electronic Industries Allianceist) ein Übertragungsstandard, der in den 1960er-Jahren festgelegt wurde. Ursprünglich war er für die DFÜ auf Telefonleitungen vorgesehen.

In den letzten Jahren wird sie im Consumerbereich kaum noch genutzt, sie ist aber bei der Mikrocontroller-Programmierung und im professionellen Industriebereich immer noch in Gebrauch. Insbesondere mit den Übertragungsarten RS422 und RS485.

6.3.2.2 RS232

Pegel

Die RS232-Schnittstelle ist eine asynchrone, serielle, full duplex Verbindung in Negativ-Logik. Die Übertragung erfolgt mittels eines Spannungspegels.



Abbildung 168 RS232 Pegel

Sehr oft wird die Schnittstelle mit modifiziertem Pegel betrieben. Dieser ist meistens für 5 V- und für 3 V-Logik passend. Man spricht dann auch von RS232-TTL.

- Logische 1 = 0 V
- Logische 0 = 3 V

Eine sehr verbreitete Methode, um die Pegel zwischen der echten RS232-Schnittstelle und RS232-TTL umzuwandeln, ist der Baustein MAX232 des Herstellers MAXIM.

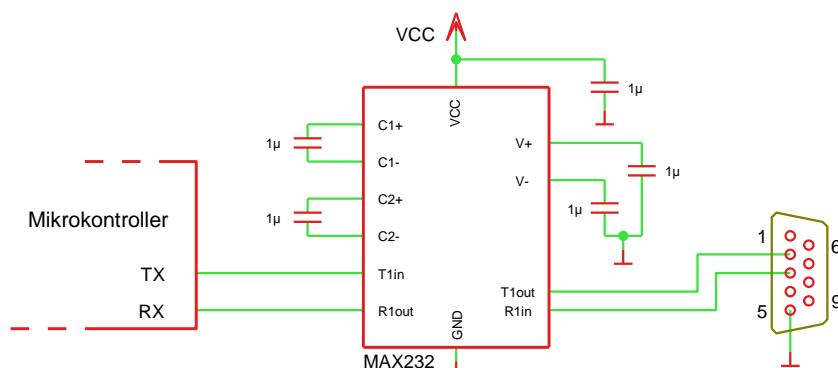


Abbildung 169 MAX232

Bezeichnungen

Die wichtigsten 4 Leitungen heutzutage sind folgende:

Abkürzung	Bedeutung
TX	Sendeleitung
RX	Empfangsleitung
RTS	Bereit zum Senden
CTS	Bereit zum Empfang

Früher wurde für die Schnittstelle ein 25-poliger DSUB-Stecker vorgesehen, der später durch einen 9-poligen DSUB-Stecker ersetzt wurde. Die komplette RS232 Schnittstelle hat heute insgesamt 9 Leitungen. Von den vielen Steuerleitungen, die früher auch notwendig waren, um ein MODEM an einer Telefonleitung komplett steuern zu können, sind im Laufe der Zeit einige weggefallen. Minimal reichen heute nur noch 2 Leitungen (TX/RX) aus.

Synchronisation und Timing

Die Synchronisation erfolgt über das immer vorhandene Startbit. Am Ende können ein bis 2 Stoppbits folgen.

Die Zeit, die für ein einzelnes Bit zur Verfügung steht, muss vor der Übertragung zwischen Sender und Empfänger festgelegt werden. Typische Übertragungsraten (Baud-Rates) sind folgende:

Bit/Sekunde	Maximale Leitungslänge in Metern
2400	900
4800	300
9600	150
19200	15
57600	5
115200	<2

UART

Damit die Kommunikation mit RS232 einfacher wird, ist sehr oft ein extra Baustein zwischen der eigentlichen Schnittstelle und der verarbeitenden CPU vorhanden: ein UART (Universal Asynchronous Receiver Transmitter). Dieser Baustein übernimmt die parallel-seriell-Wandlung und ist heutzutage oft in Mikrocontrollern gleich mitintegriert.

Handshake

Damit bei der Übertragung keine Daten verloren gehen, gibt es zwei Handshakeverfahren:

- Software-Handshake
- Hardware-Handshake

Beim Hardware-Handshake wird über die beiden zusätzlichen Leitungen RTS und CTS zwischen Sender und Empfänger die Übertragung angehalten und dann wieder fortgesetzt. Beim Software-Handshake sendet der Empfänger Xon und Xoff Zeichen um den Sender anzuhalten oder wieder fortzusetzen zu lassen (Xon = 11h und Xoff = 13h).

6.3.2.3 RS422

Die RS422 (auch EIA 422) Schnittstelle ist eine Weiterentwicklung der RS232 und benutzt dasselbe Bitmuster. Die Signale werden Vollduplex und symmetrisch mit 2 Leitungen pro Richtung übertragen, und zusätzlich ist es jetzt möglich, einen Sender mit bis zu 10 Empfängern zusammenzuschalten.

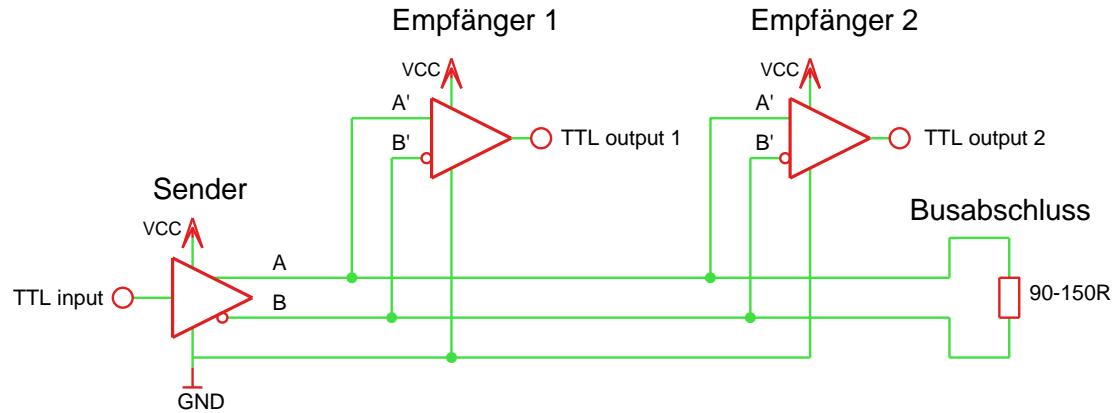


Abbildung 170 RS422 Verschaltung

Auf den beiden Signalleitungen A (auch Y) und B (auch Z) liegt immer ein Signal umgekehrter Polarität an, und der meiste Strom fließt durch den Abschlusswiderstand. Über die Masseleitung fließt so gut wie kein Strom.

Die Spannungspiegel liegen ähnlich wie bei RS232; typisch jedoch bei ± 5 V, und der Bereich zwischen +0,2 V und -0,2 V ist nicht definiert.

Die maximale Leitungslänge beträgt bis zu 1200 m, und die maximale Übertragungsrate liegt bei 10 Mbit/s, wobei nicht beides gleichzeitig möglich ist. Ab 200 kBit/s ist der Abschlusswiderstand unbedingt notwendig.

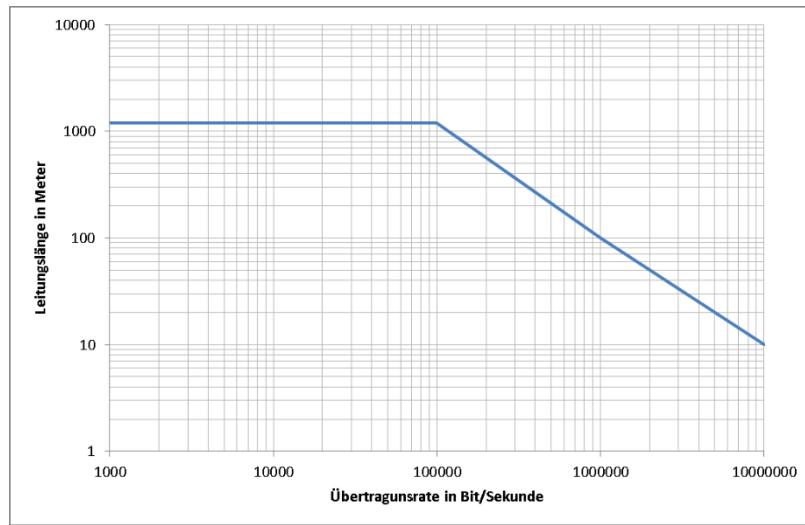


Abbildung 171 RS422 Leitungslänge

Die Qualität der Kabel hat auch noch großen Einfluss auf die maximale Kabellänge und Übertragungsrate. Ein großer Vorteil ist es, wenn die beiden symmetrischen Adern miteinander verdrillt sind (Twisted Pair). Grundsätzlich empfohlen werden CAT5-Kabel, die sonst für Ethernet Netzwerke verwendet werden.

6.3.2.4 RS485

Grundsätzliches

Bei der RS485-Schnittstelle (auch EIA 485) wurde die RS422-Schnittstelle in Halbduplex überführt. Es können jetzt bis zu 32 Geräte direkt zusammengeschaltet werden. Es existieren auch Treiberbausteine mit erhöhter Leistung, mit denen bis zu 256 Geräte möglich sind. Durch das Halbduplexverfahren muss jetzt der Sendeverstärker jedes Gerätes abgeschaltet werden können. Wann diese Abschaltung des Senders erfolgt, muss vorher über das Kommunikationsprotokoll vereinbart werden.

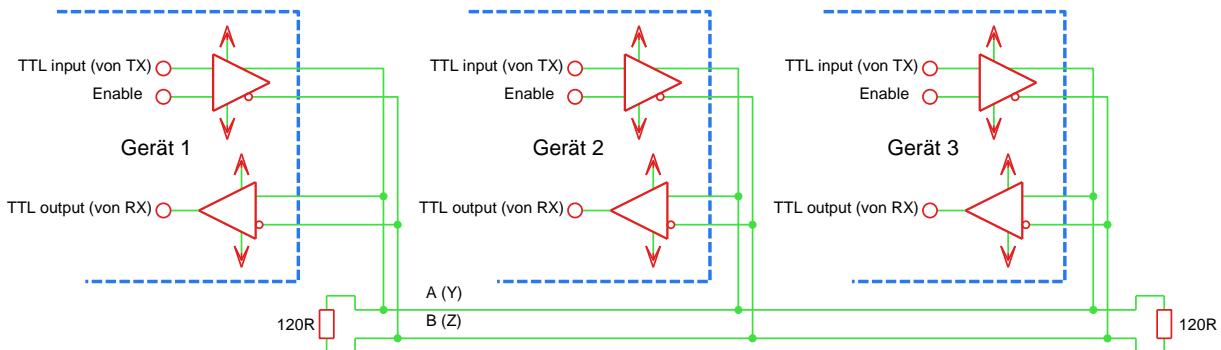


Abbildung 172 RS485 Verschaltung

Bezugsleitung

Durch die umgekehrte gegenphasige Polarität der beiden Signale fließt auf der GND-Bezugsleitung prinzipiell kein Strom. Theoretisch kann sie deshalb weggelassen werden. Die Geräte „holen“ sich ihr Bezugspotenzial dann über das Erdpotenzial. Es ist ein maximaler Gleichtaktfehler von -7 bis +12 V erlaubt. Bei großen Leitungslängen sollte jedoch eine Bezugsleitung mitgeführt werden, da es bei stark unterschiedlichem oder verrauschem Erdpotenzial zu Störungen kommen kann.

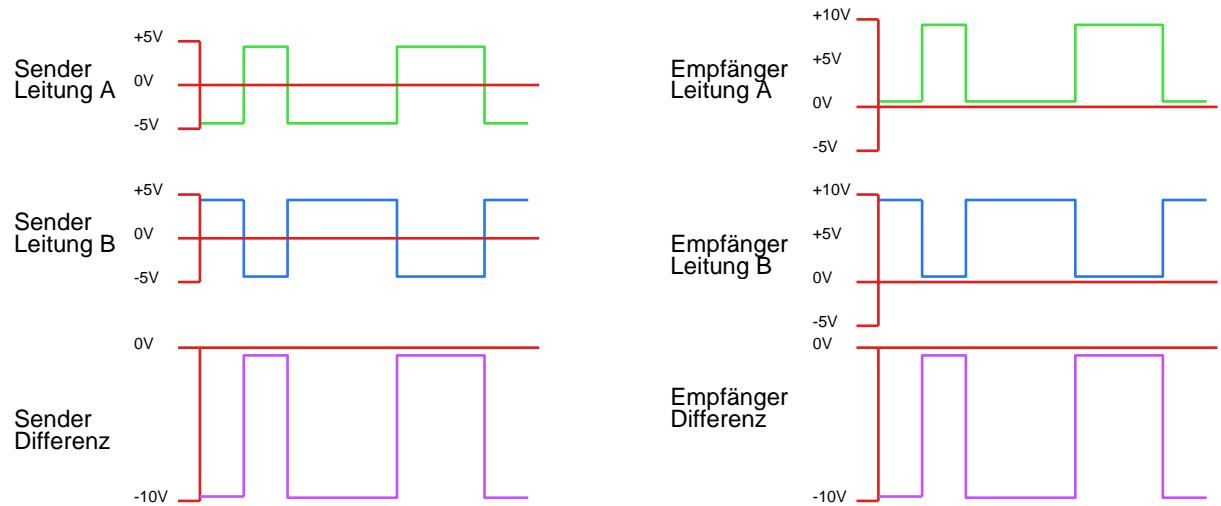


Abbildung 173 RS485 Gleichtaktfehler

Die über das „schlechte“ Bezugspotenzial eingebrachte Gleichtaktstörung fällt auf der Empfängerseite durch die Differenzbildung gar nicht auf.

Busabschluss

Da die Sender jetzt abschaltbar sind, ist es jetzt möglich, dass gar kein Sender am Bus aktiv ist. Bei dem Busabschluss mit nur einem $120\ \Omega$ -Widerstand wird es passieren, wenn gerade kein Sender aktiv am Bus ist, dass die Leitung auf den undefinierten Pegel von 0 V fällt. Manche RS485 Empfänger fangen bei ständigem 0 V Differenzsignal an zu schwingen und werden dadurch zum Zufallsgenerator.

Um dieses zu verhindern, gibt es als Busabschluss eine verbesserte Version, welche ohne Treiber auf einem korrekt definierten Pegel stehen bleibt. Man spricht auch vom Failsafe-Busabschluss.

Failsafe-Busabschluss

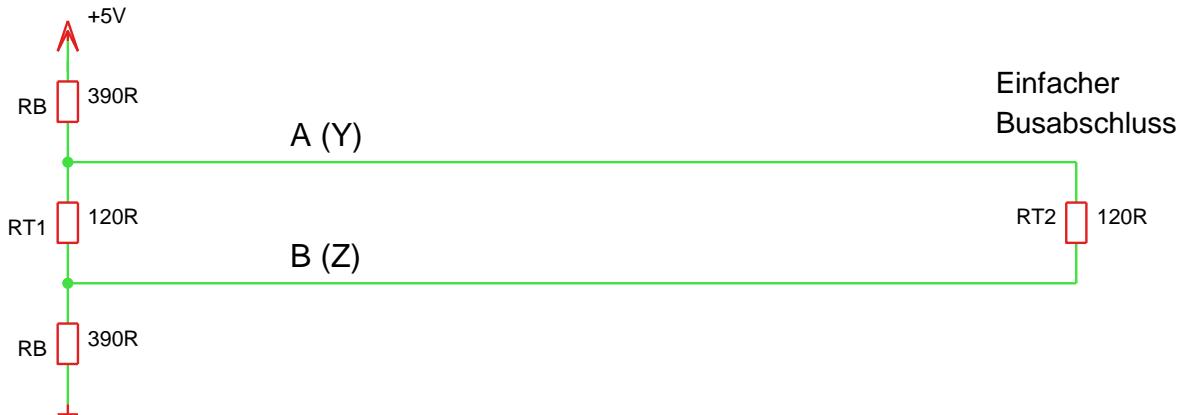


Abbildung 174 RS485 Beispiele Busabschluss

Neuere Versionen

Spätere Erkenntnisse haben ergeben, dass der Absolutpegel für den Störabstand fast keine Rolle spielt. Der Empfänger „sieht“ sowieso nur die Differenz der beiden Signale, egal ob diese jetzt positive oder negative Anteile enthalten.

Bei neueren RS485 Bausteinen wird deshalb auf die negative Komponente der Spannungspiegel verzichtet. Die Signale bewegen sich jetzt nur noch zwischen +5 V und 0 V.

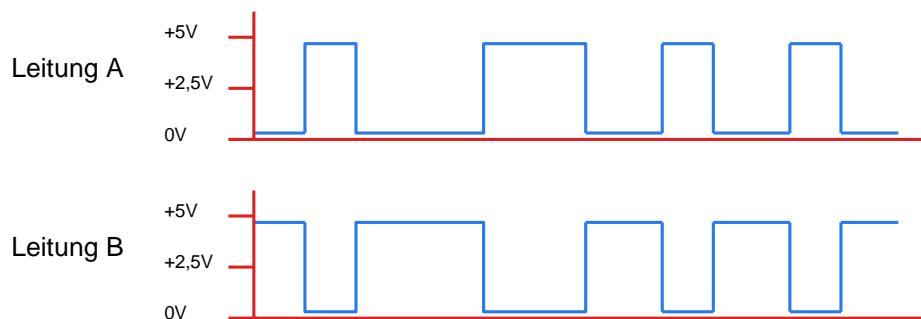


Abbildung 175 RS485 positive Pegel

Durch den Verzicht auf die negativen Spannungen wird das elektrische Design wesentlich einfacher, denn die Erzeugung der negativen Hilfsspannungen kann jetzt wegfallen. Der Störabstand des Signals wird dadurch nur unwesentlich schlechter.

6.3.3 SPI

Das SPI (Serial Peripheral Interface) ist eine einfache, synchrone, serielle, full-duplex-Schnittstelle nach dem Master-Slave-Prinzip, die 1987 vom Motorola eingeführt wurde. Die Schnittstelle ist hauptsächlich als digitale Kommunikation zwischen verschiedenen Bausteinen auf einer Leiterplatte oder innerhalb von Geräten gedacht. Einige Mikrocontroller-Hersteller benutzen die SPI-Schnittstelle dazu, um das Anwenderprogramm mittels ISP auf das System einzuspielen (siehe Kapitel 8.4.4). Oft wurde für diese Aufgabe der Parallel-Port eines PCs zweckentfremdet (siehe Kapitel 6.2.1). Die Datenrate kann bis zu mehrere MHz betragen, je nach Bausteintyp und Leitungslänge.

Die Schnittstelle besteht hauptsächlich aus 4 Leitungen:

- MOSI (Master Output, Slave Input)
- MISO (Master Input, Slave Output)
- SCLK (Serial Clock)
- SS, CS, EN (Slave Select, Chip Select, Enable)

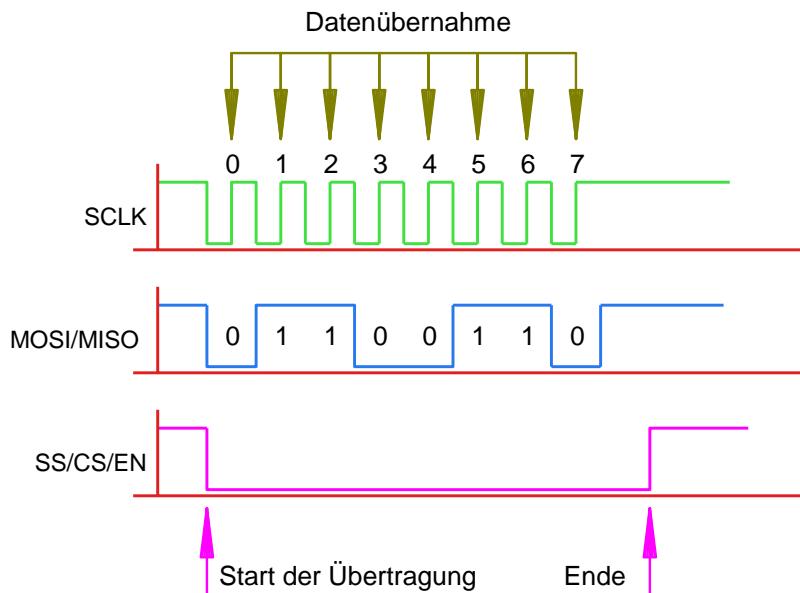


Abbildung 176 SPI Signale

Das anliegende Datensignal auf der MISO- oder MOSI-Leitung wird oft bei der steigenden Taktflanke der SCLK-Leitung vom Empfänger übernommen. Die gesamte Übertragung wird mit der SS/CS/EN-Leitung gestartet und synchronisiert. Die Datenpakete haben in der Regel 8 Bit, es gibt auch Bausteine, die mehr als 8 Bit direkt hintereinander ausgeben können.

Es gibt immer genau einen Masterbaustein, der die Kontrolle über den Bus übernimmt. Wenn mehrere Slave-Bausteine vorhanden sind, werden diese über eine oder mehrere SS/CS/EN-Leitungen aktiviert.

Manche Bausteine benötigen nur einen digitalen Dateneingang oder -Ausgang und nicht beides, sie haben deshalb nur 3 SPI-Anschlüsse.

Wenn mehrere Bausteine von einem Master bedient werden, gibt es zwei Möglichkeiten diese miteinander zu verbinden.

1. Man spendiert jedem einzelnen Baustein eine eigene Select-Leitung

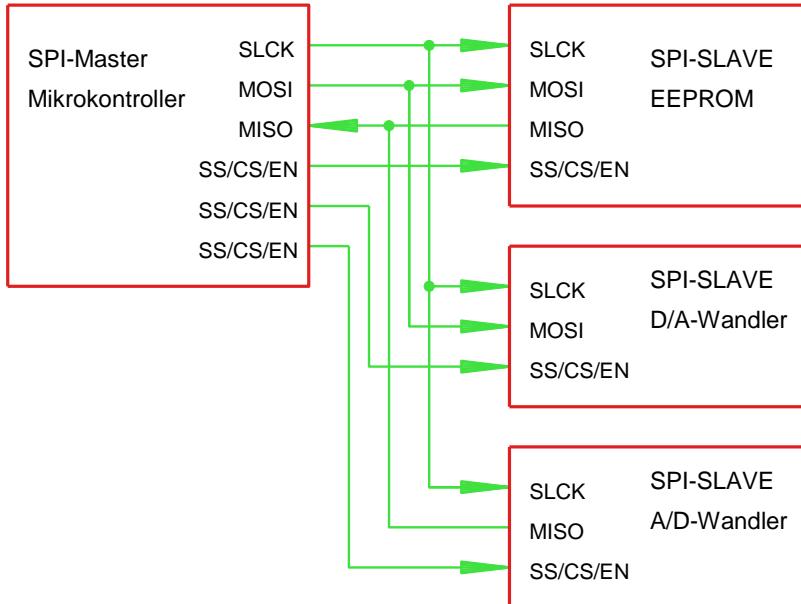


Abbildung 177 SPI Multi-CS

2. Die MISO/MOSI Leitungen der Bausteine werden hintereinander geschaltet

Die Verkettung von Bausteinen in dieser Art wird oft auch als Daisy-Chain bezeichnet.

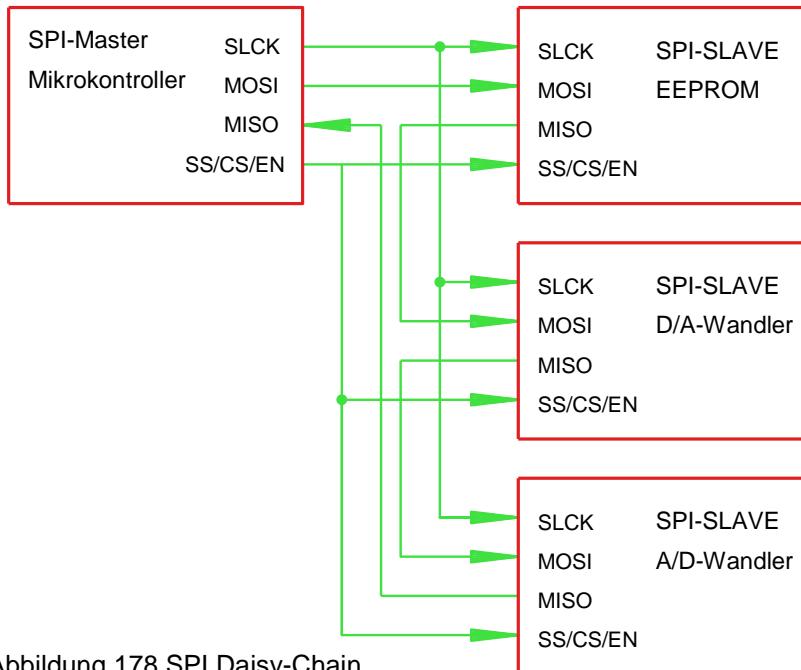


Abbildung 178 SPI Daisy-Chain

Wenn die gewählten Bausteine nicht die beiden Datenleitungen MISO/MOSI enthalten, ist eine Daisy-Chain-Verkettung nicht möglich.

6.3.4 I²C

Die I²C Schnittstelle (**I**nter-**I**ntegrated-**C**ircuit) wurde ähnlich wie die SPI-Schnittstelle dafür entwickelt, um die Kommunikation zwischen verschiedenen Bausteinen auf einer Leiterplatte, oder innerhalb von Geräten zu vereinfachen. Sie wurde in den 1980er-Jahren von Philips Semiconductors entwickelt, um die Kommunikation innerhalb von Fernsehgeräten zu standardisieren. In den 1990ern haben einige Hersteller die Schnittstelle übernommen und heutzutage gibt es weit über 1000 Bausteine, welche die I²C Schnittstelle unterstützen.



Abbildung 179 I²C Logo

Die Schnittstelle ist eine synchrone, serielle, halb-duplex Schnittstelle nach dem Master-Slave-Prinzip. Hier sind auch mehrere Master möglich. Sie kommt mit nur 2 Leitungen aus:

- SCL (Serial Clock)
- SDA (Serial Data)

Die Kommunikation entspricht einem kompletten Bustransfer mit getrennter Adress- und Datenübertragung. Alle Bausteine haben unterschiedliche vom Hersteller festgelegte Adressen. Es werden immer 8-Bit-Worte übertragen, die protokollbedingt Adresse, Kommandos oder Daten beinhalten können.

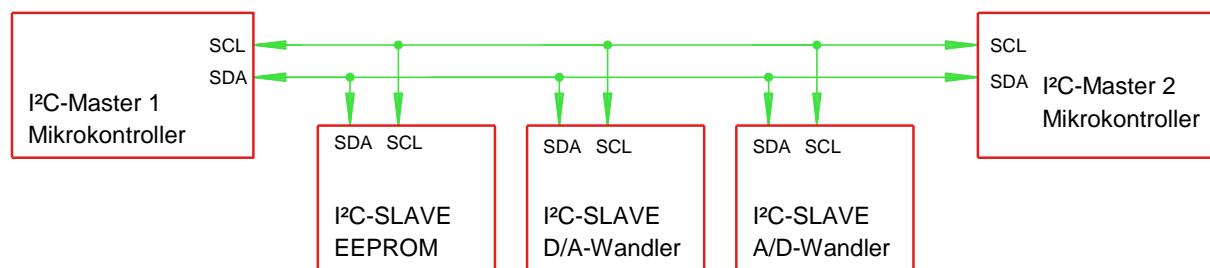


Abbildung 180 I²C Verschaltung

Die maximale Übertragungsrate für den I²C-Bus liegt zwischen 0,1 MBit/s und 5 MBit/s.

Bei manchen Bausteinen können, damit von einem Baustein mehrere Gleiche in einer Schaltung eingesetzt werden können, z.B., die unteren 1 bis 3 Bit der Adresse gewählt werden.

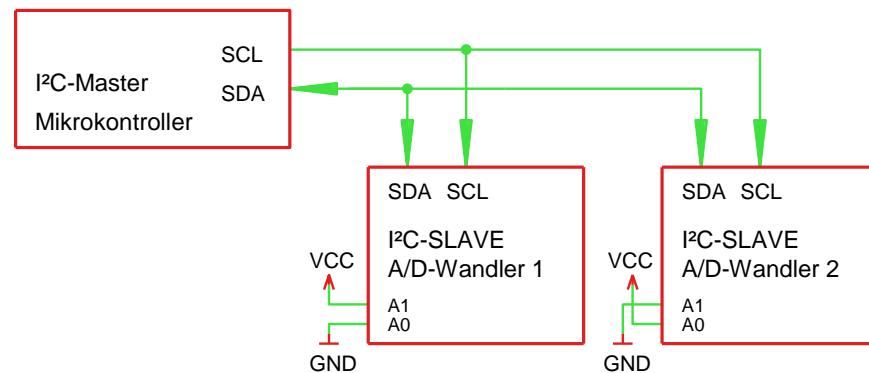


Abbildung 181 I²C Adresswahl

Die Übertragung auf dem I²C wird vom Master mit dem Start-Signal gestartet und mit dem Stop-Signal wieder beendet. Zum Start wird die SDA-Leitung vom Master auf Low gezogen, während die SCL-Leitung auf High-Signal liegt. Beim Stop-Signal wird die SDA-Leitung auf High-Pegel gebracht, während die SCL-Leitung auch auf High-Signal liegt.

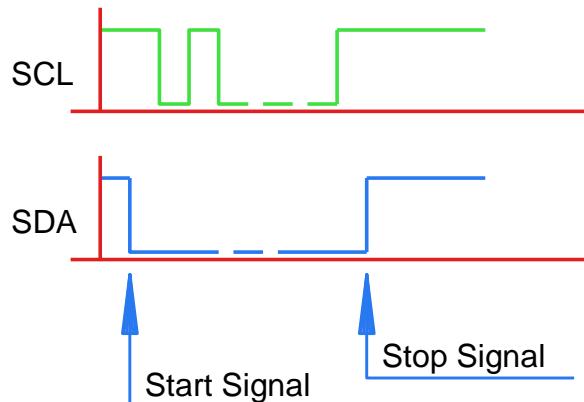


Abbildung 182 I²C Start-Stop

Nach dem Start-Signal beginnt der Master mit dem Anlegen der Slave-Adresse. Der Slave kann die SCL-Leitung nach Low ziehen um dem Master „Busy“ zu signalisieren. Wenn der Slave bereit ist, zieht er als Acknowledge-Signal die SDA-Leitung nach low. Danach sendet der Master die entsprechenden Daten zum Slave oder holt sie vom Slave.

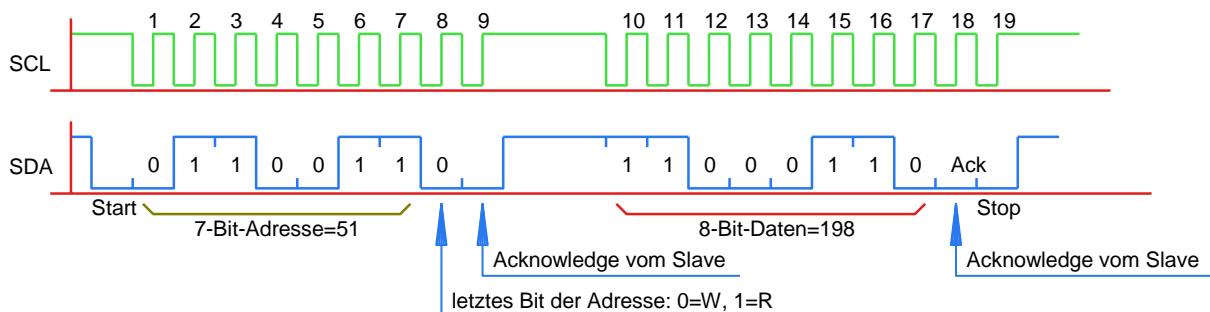


Abbildung 183 I²C Datenübertragung

Nach der Adresse können auch mehrere Datenwörter unterwegs sein bis zum Stop-Signal, je nachdem welche Art von Baustein angesprochen wird.

Der I²C-Bus ist nur für kurze Strecken bis ca. 30 cm geeignet, und ist wesentlich langsamer als eine SPI-Übertragung. Der größte Vorteil ist, dass man nur 2 Leitungen benötigt und damit relativ viele Bausteine zusammenschalten kann.

Es gibt Bausteine, die mehrere Peripherien enthalten und gleich mehrere I²C-Adressen belegen. Der Adressraum wurde später auf 10 Bit erweitert, dadurch sind jetzt anstatt 112 bis zu 1136 (112+1024) Adressen möglich.

6.3.5 SATA

In den 1990er-Jahren war es mit der parallelen PATA-Schnittstelle nicht mehr möglich, die Daten noch schneller zu übertragen. Die parallelen 16 Bit über ein bis zu 1 m langes Kabel zu übertragen setzt gewisse physikalische Grenzen (siehe Kapitel 2.5.3).

Die einzige Möglichkeit, die Übertragung schneller zu bekommen, war der Umstieg auf eine serielle Schnittstelle. Die SATA-Schnittstelle (**S**erial **A**dvanced **T**echnologie **A**ttachment) ist eine synchrone, symmetrische und serielle Punkt-zu-Punkt-Verbindung mit Taktrückgewinnung. Es wird eine 8b/10b-Kodierung verwendet, d.h., für 8-Bit Nutzdaten müssen 10 Bit übertragen werden, womit die Nettodatenrate um 25 % kleiner ist als die immer angegebene Bruttodatenrate.

Das Protokoll wurde beibehalten, und die seriell/parallel Umsetzung wurde vollkommen transparent von der Hardware übernommen.

Der neue kleine Stecker hat insgesamt 7 Leitungen, 3x Masse und 2x2 Daten-Leitungen für jede Richtung. Gleichzeitig wurde der Stromversorgungsanschluss überarbeitet. Alle Steckertypen sind für Hot-Plugging ausgelegt mit vorausseilenden Kontakten.



Abbildung 184 SATA-Kabel

Die Schnittstelle hat inzwischen schon eine Evolution hinter sich, und es gibt sie in verschiedenen Übertragungsgeschwindigkeiten. Die Schnittstelle ist aufwärts und abwärtskompatibel.

Bezeichnung	Netto-Übertragungsrate
SATA I 1,5 GBit/s	150 MByte/s
SATA II 3,0 GBit/s	300 MByte/s
SATA III 6,0 GBit/s	600 MByte/s
SATA Express 8 GBit/s	985 MByte/s
SATA Express 16 GBit/s	1969 MByte/s

Abbildung 185 SATA-Versionen

Später kam noch ein Stecker für externe Verbindungen mit zusätzlicher Verriegelung mit der Bezeichnung eSATA hinzu. Diese wurde jedoch von den heutzutage noch schnelleren USB-Varianten fast vollständig verdrängt.

6.3.6 SAS

Die serielle SCSI-Schnittstelle SAS (Serial Attached SCSI) ist der seriellen ATA-Schnittstelle technisch sehr ähnlich. Hauptunterschied ist der erhöhte Spannungspegel auf den Leitungen, die ein längeres Kabel bis zu 6 m ermöglichen. Im Gegensatz zu parallelem SCSI-Bus ist SAS eine Punkt-zu-Punkt-Verbindung. Der Einsatz ist hauptsächlich für den professionellen Serverbereich vorgesehen.

6.3.7 M.2

Die M.2 Schnittstelle ist eigentlich keine weitere neue Schnittstelle. Es wurde hier nur die PCI-Express-Schnittstelle (siehe Kapitel 2.5.3) als M-Key oder die SATA-Schnittstelle (siehe Kapitel 6.3.5) als B-Key auf einen veränderten Stecker geführt. Damit wurde eine sehr kompakte Möglichkeit geschaffen, um hauptsächlich SSD-Festplatten noch kleiner und flacher herstellen zu können. Es gibt auch B-Key Module, die kein SATA-Protokoll unterstützen.

Auf den Adapterkarten von M.2-M-Key auf PCI-Express ist, da es sich um dieselbe Schnittstelle handelt, bis auf einen Entkopplungskondensator keinerlei Elektronik drauf (Vor- und Rückseite):



Abbildung 186 M.2 PCIe Adapter

M.2-M-Key- und M.2-B-Key-Sockel sind nicht zueinander kompatibel!



Abbildung 187 M.2 Sockeltypen

Es gibt jedoch Module, die beide Key-Ausführungen bedienen können. Man kann sie an den **Zwei** Einkerbungen anstatt sonst nur **Einer** erkennen.

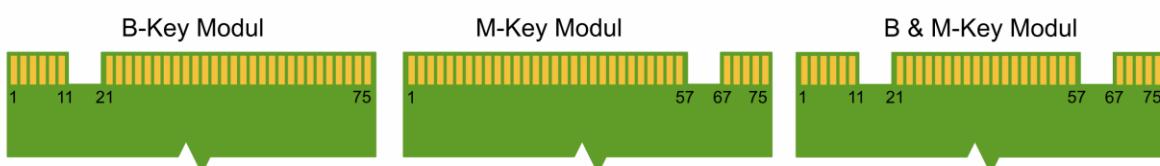


Abbildung 188 M.2 Modultypen

Anmerkung: Bei B-Key sind maximal 2 PCI-Express-Lanes möglich
Bei M-Key sind maximal 4 PCI-Express-Lanes möglich

6.3.8 USB

6.3.8.1 Allgemeines

Der **USB** (Universal Serial Bus) wurde 1996 als serielles Bussystem für alle Arten von Computern eingeführt. Er sollte die damals übliche serielle RS232, die parallele Centronics-Schnittstelle, Drucker, Scanner, Maus und Tastaturanschlüsse ersetzen. Anfangs wollte ihn keiner haben, und er wurde deshalb manchmal als **Useless Serial Bus** bezeichnet. Es hat einige Jahre gedauert, bis sich das Bussystem etabliert hat. Heutzutage ist der USB-Anschluss als Verbindung zwischen Computern der Standard für fast alle Computerverbindungen.

Die Schnittstelle besteht in der Grundversion aus 4 Leitungen +5 V, 0 V, D+, D-, und es gibt grundsätzlich 2 Steckertypen A und B. Der „Bus“ besteht immer aus einer Punkt-zu-Punkt-Verbindung mit einem Host-Master (Steckertyp A) und einem Slave-Client (Steckertyp B).



Abbildung 189 USB A/B-Stecker

Die 2 Kontakte für die Versorgung (+5 V und 0 V) sind immer etwas länger ausgeführt als die Datenleitungen, damit wird ein Hot-Plugging ermöglicht, da die Versorgung dann immer zuerst verbunden und zuletzt unterbrochen wird. Die maximale Stromstärke wurde auf 0,5 A festgelegt.

Die Übertragung der Daten erfolgt über die Leitungen **D+** und **D-**. Die Signale sind symmetrisch ausgeführt, **D+** hat immer den entgegengesetzten Pegel von **D-**. Damit ist ein sehr störsicherer Betrieb möglich (siehe Kapitel 6.3.1.7).

Die Schnittstelle ist inzwischen auch zur Standardverbindung als Stromversorgung und Ladeanschluss tragbarer Geräte geworden. Obwohl der USB-A-Stecker ursprünglich nur für Ströme bis 0,5 A entwickelt wurde, werden heute Ströme über 2 A damit übertragen.

Bei den Steckertypen gab es im Laufe der Jahre eine mechanische Evolution zu immer kleineren Versionen. Die elektrische Seite ist vom Prinzip her gleich geblieben. Ab USB-3 wurde die Anzahl der Datenleitungen auf 6 erhöht.

	USB 2.0		USB 3.0
Typ A			
Typ B			
Mini-B			-
Micro-B			

Abbildung 190 USB-Steckertypen

6.3.8.2 USB1

Der 1996 eingeführte USB 1.0 Standard wurde wegen einigen Fehlern quasi nie verwendet und der eigentlich erste USB-Standard ist USB1.1 ab ca. 1998.

Anfangs wurden 2 verschiedene Geschwindigkeitstypen spezifiziert. Low-Speed und Full-Speed. Die Clients teilen dem Host über ihre Datenleitungen mit einem Pull-Up-Widerstand nach +3,3 V statisch mit, zu welcher Gruppe sie gehören.

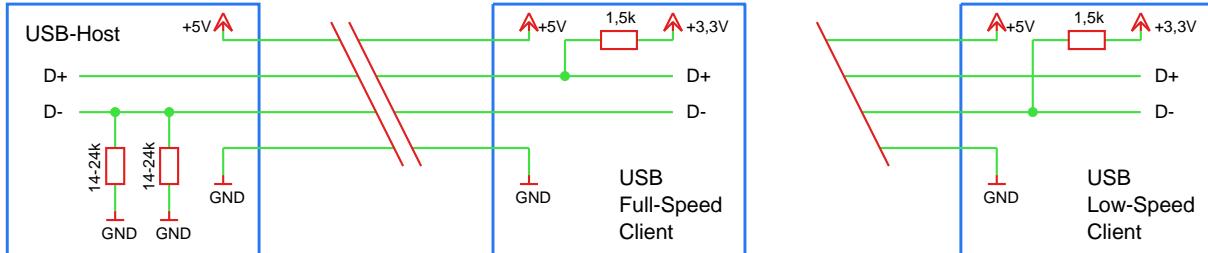


Abbildung 191 USB-low-full-Speed

Die maximal mögliche Übertragungsrate ist bei USB1.1 bei **Low-Speed** 1,5 MBit/s und bei **Full-Speed** 12 MBit/s. Die maximale Kabellänge beträgt beim USB1.1 **5 m**.

6.3.8.3 USB2

Als man USB immer mehr für externe Datenträger verwendete, wurde es notwendig, die Datenrate zu erhöhen. Mit dem Standard USB2 wurde sie, ohne den Steckertyp zu ändern, auf 480 Mbit/s erhöht. Das wurde erst möglich durch einen sehr niederohmigen Busabschluss von 45 Ohm auf beiden Leitungen. Das ergibt einen differenziellen Widerstand von 90 Ohm zwischen **D+** und **D-**. Dieser Wert entspricht etwa dem Wellenwiderstand der verwendeten Kabel.

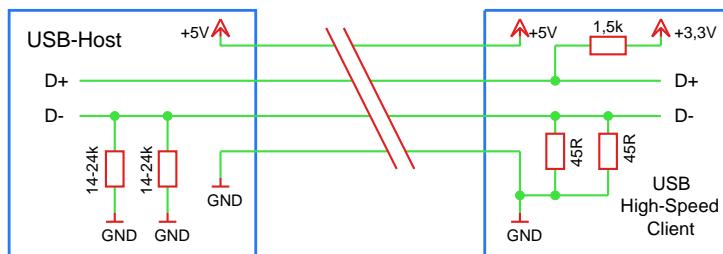


Abbildung 192 USB 2.0 High-Speed

Der USB2-Client gibt sich zuerst als Full-Speed-Gerät aus und schaltet dann kurz danach die beiden 45 Ohm Widerstände an die **D+** und **D-** Leitung. Der Host erkennt das und stuft das Gerät dann als **High-Speed** ein. Das Verfahren ist sehr aufwendig und wurde notwendig, um vollständige Rückwärts- und Vorwärtskompatibilität zu gewährleisten.

Zusätzlich wurde beim USB2.0 die Codierung der Bits auf den Leitungen für weniger Overhead geändert.

Die maximale Kabellänge beträgt beim USB2.0 immer noch **5 m**.

6.3.8.4 USB3

USB3.0

Um die Geschwindigkeit gegenüber USB2.0 noch weiter zu erhöhen, musste jetzt doch der Steckertyp auf mehr Daten-Leitungen hin verändert werden. Hierbei wurde trotzdem auf wieder auf Rückwärts- und Vorwärtskompatibilität geachtet.

Es wurden 5 zusätzliche Leitungen als 2 differenzielle-fullduplex-Paare mit extra Massepin im Jahr 2008 eingeführt.

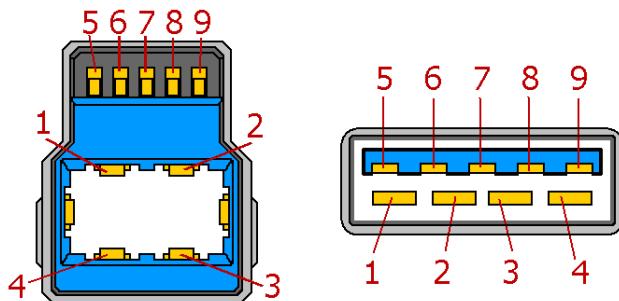


Abbildung 193 USB 3 Stecker

Pin Nr.	Name	Beschreibung
1	VBUS	+5V
2	D+	differenzielles Paar 1+
3	D-	differenzielles Paar 1-
4	Masse	0V
5	SSTX- / SSRX-	differenzielles Paar 2- TxRx
6	SSTX+ / SSRX+	differenzielles Paar 2+ TxRx
7	Masse	Masse für differenzielles Paar 2
8	SSRX- / SSTX-	differenzielles Paar 2- TxRx
9	SSRX+ / SSTX+	differenzielles Paar 2+ TxRx

Abbildung 194 Belegung USB3-Stecker

Die hohe Übertragungsrate nennt sich **Super-Speed-USB** und erreicht jetzt 5 GBit/s. Sie wurde erreicht durch die Fullduplex-Übertragung und auch wieder durch eine Änderung der Bit-Codierung auf der Leitung.

Um eine bessere Unterscheidung zu ermöglichen, wurde für die USB3-Stecker und Buchsen eine blaue Farbe festgelegt.

Zusätzlich wurde die Stromverfügbarkeit von 0,5 A ab dem USB3.0 auf 0,9 A bei immer noch 5 V erhöht. Maximale Leistung jetzt 4,5 Watt.

Die maximale Kabellänge beträgt beim USB3.0 jetzt **3 m**.

USB3.1/3.2

Die nächste große Änderung beim USB-Bus war ein neuer Steckertyp (Typ-C), der jetzt nicht mehr mechanisch rückwärtskompatibel ist. Das Wichtigste bei dem neuen Stecker ist die Tatsache, dass er in beiden Richtungen eingesteckt werden kann. Es sind jetzt 4 Halbduplex-SuperSpeed-Leitungspaare vorhanden.



Abbildung 195 USB-Steckertyp-C

Folgende Änderungen wurden für USB3.1 verwirklicht:

- Neuer symmetrischer Steckertyp-C
- Host und Client mit demselben Steckertyp
- Kabel mit 24 Leitungen
- Effizientere Codierung vorher 8b10b-Code jetzt 128b132b-Code
- Höhere Übertragungsrate
- Verschiedene Spannungen möglich
- Mehr elektrische Leistung möglich

Trotzdem ist der USB elektrisch immer noch rückwärtskompatibel mit allen vorherigen USB-Standards. Es sind alle möglichen Kabeladapter verfügbar.

Übertragungsrate USB 3.1 bis zu 10 GBit/s (**SuperSpeed+**) oder (**SuperSpeed USB 10Gbps**)

Beim USB3.2; definiert ab ca. 2017, sind durch Ausnutzung der zusätzlichen Leitungen im Kabel bis zu 20 GBit/s möglich (**SuperSpeed USB 20Gbps**).

Die maximale Kabellänge beträgt ab USB3.1 nur noch 1 m.

6.3.8.5 USB4

USB4 wurde 2020 eingeführt und ist für 40 Gbit/s ausgelegt. Es wird erstmalig das Thunderbolt-Protokoll von Intel verwendet. Die Weiterentwicklung ab 2022 nennt sich USB4 2.0 und überträgt bis zu 80 Gbit/s.

6.3.8.6 Power Delivery

Ungefähr mit der Einführung des USB-Typ-C-Steckers ab ca. 2013 wurde es auch möglich deutlich mehr Leistung zu übertragen als vorher mit den USB-Typ-A/B-Steckern.

Die oft bei reinen Ladegeräten verwendeten maximale Stromstärken von typisch 2.1A wurde für den Typ-A Stecker nie offiziell festgelegt.

Modus	Stecker	Spannung	max. Strom	max. Leistung
USB1&2	A/B	5 V	0,5 A	2,5 W
USB3.0	A/B	5 V	0,9 A	4,5 W
SPR	C	5 V	3 A	15 W
SPR	C	9 V	3 A	27 W
SPR	C	15 V	3 A	45 W
SPR	C	20 V	3 A	60 W
SPR	C	20 V	5 A	100 W
EPR	C	28 V	5 A	140 W
EPR	C	36 V	5 A	180 W
EPR	C	48 V	5 A	240 W

SPR -> Standard Power Range

EPR -> Extended Power Range

Die Spannungsversorgung aller USB-Schnittstellen beginnt immer mit 5 V. Es ist eine Stromstärke von maximal 100 mA erlaubt. Wenn höhere Ströme gewünscht werden, muss der versorgende Controller vorher gefragt werden. An diese Regel halten sich jedoch sehr wenige Geräte. Nahezu alle Controller beschweren sich deshalb normalerweise nicht, wenn die Schnittstelle sofort von Anfang an mit mehr als 100mA belastet wird.

Host und Client handeln über ein Protokoll aus, welche Spannungen möglich sind und/oder gewünscht werden. Erst danach wird auf der Hostseite auf eine Spannung größer als 5 V umgeschaltet.

Damit Hot-Plugging weiterhin gefahrlos möglich wird, sind in den USB-Typ-C Kabeln Kondensatoren verbaut. Nur so kann gewährleistet werden, dass beim Abziehen des Kabels unter voller Last kein Lichtbogen entsteht, der den Stecker zerstören würde.

Die „Extended Power Range“ wurde erst später ab ca. 2021 eingeführt und ermöglicht Spannungen bis 48 V. Die ersten USB-Typ-C-Kabel sind ausgelegt für die „Standard Power Range“ und für maximal 20 V. Das könnte zu Problemen führen, falls die im Kabel enthaltenen Kondensatoren nicht für 48 V geeignet sind.

6.3.8.7 USB-Converter

Als vor einigen Jahren die „älteren“ Standard-Schnittstellen bei einigen Computern langsam weggefallen sind, wurde eine Lösung benötigt, um ältere Geräte weiter zu betreiben. Dazu haben einige Hersteller Adapterchips entworfen, die für verschiedene Betriebssysteme die alten Schnittstellen simulieren.

Diese Methode ist jedoch nicht immer erfolgreich. Es sind immer noch einige Geräte auf dem Markt, die vom Entwickler über den Bit-Bang-Modus angesteuert werden (siehe Kapitel 6.3.1.1). Der USB-Adapter kann jedoch nur korrekt arbeiten, wenn die simulierte Schnittstelle korrekt über ihre Betriebssystem-APIs angesteuert wird. Besonders bei Timing-kritischen Anwendungen sind dann Probleme zu erwarten.

Es gibt viele Hersteller von USB-Schnittstellenadapters, hier einige Beispiele, die sich besonders bewährt haben, weil mit ihnen wenige Probleme aufgetaucht sind:

Hersteller FTDI:

Typ	Schnittstelle
FT232R	USB to RS232-UART-Bride
FT2232H	Double USB to RS232-UART-Bride
FT4222H	USB to SPI / I2C Bridge

Von FTDI gibt es fertige Kabel, bei denen die gesamte Elektronik komplett in einen USB-Typ-A-Stecker integriert ist.



Abbildung 196 USB-Adapter-Kabel

Hersteller Silabs:

Typ	Schnittstelle
CP2101	USB to RS232-UART-Bride
CP2105	Double USB to RS232-UART-Bride
CP2130	USB to SPI Bridge
CP2112	USB to I2C Bridge

Bei neueren Adaptersetzt sich langsam ein sehr preiswerter Konverter-Typ aus chinesischer Produktion durch: CH340/CH341.

6.3.8.8 USB-Umbenennung

Die Bezeichnungen der unterschiedlichen USB-Standards wurden in der Vergangenheit mehrmals umbenannt.

<i>Ursprünglich</i>	<i>1. Änderung</i>	<i>Aktuell</i>	<i>Speed</i>	<i>Einführung</i>	<i>Steckertyp</i>
USB 1.0	USB 1.0	USB 1.0	12 MBit/s	1996	A/B
USB 1.1	USB 1.1	USB 1.1	12 MBit/s	1998	A/B
USB 2.0	USB 2.0	USB 2.0	480 MBit/s	2000	A/B
USB 3.0	USB 3.1 Gen 1	USB 3.2 Gen 1	5 Gbit/s	2008	A/B/C
USB 3.1	USB 3.1 Gen 2	USB 3.2 Gen 2	10 Gbit/s	2013	C
USB 3.2	USB 3.2	USB 3.2 Gen 2x2	20 Gbit/s	2017	C
USB4		USB4 20	20 Gbit/s	2019	C
USB4		USB4 40	40 Gbit/s	2020	C
USB4 2.0			80 Gbit/s	2022	C

Die Vor- und Rückwärtskompatibilität ist immer gewährleistet, da im Typ-C Anschluss das Halbduplex-Leitungspaar des ersten USB-Standards 1.0 immer zusätzlich mit enthalten ist.

6.3.9 Netzwerke

6.3.9.1 Drahtgebunden

Allgemeines

Netzwerkverbindungen sind aus physikalischen Gründen immer serielle Verbindungen. Bei parallelen Verbindungen würden bei langen Leitungen die gleichen Laufzeit-Probleme auftauchen wie auch bei den parallelen Bussen innerhalb eines Computers (siehe Kapitel 2.5.3).

Früher gab es viele unterschiedliche Netzwerke, die im Laufe der Zeit fast alle durch das Ethernet ersetzt wurden. Im industriellen Bereich existieren immer noch einige kleinere Netzwerke, die oft zum Zusammenschalten von verschiedenen Messgeräten dienen, z.B. RS485 (siehe Kapitel 6.3.2.4).

Da die Netzwerke teilnehmer über sehr lange Leitungen miteinander verbunden werden, sollten die Teilnehmer galvanisch voneinander getrennt werden. Dazu wird oft bei jedem Teilnehmer ein Übertrager eingesetzt. Aus diesem Grund müssen die Bitmuster auf dem Kabel gleichanteilsfrei sein.

Koaxial

In den 1990er-Jahren wurden oft alle Teilnehmer eines Ethernet-Netzwerkes durch ein einziges Koaxial-Kabel miteinander verbunden. An den beiden Enden des Netzwerkes musste das Kabel durch einen 50Ω -Abschlusswiderstand terminiert werden. Das funktionierte nur zuverlässig, wenn nur wenige Teilnehmer (<10) beteiligt waren, da sich alle das Übertragungsmedium teilen und oft Kollisionen auftraten, wenn mehrere Sender gleichzeitig sendeten.

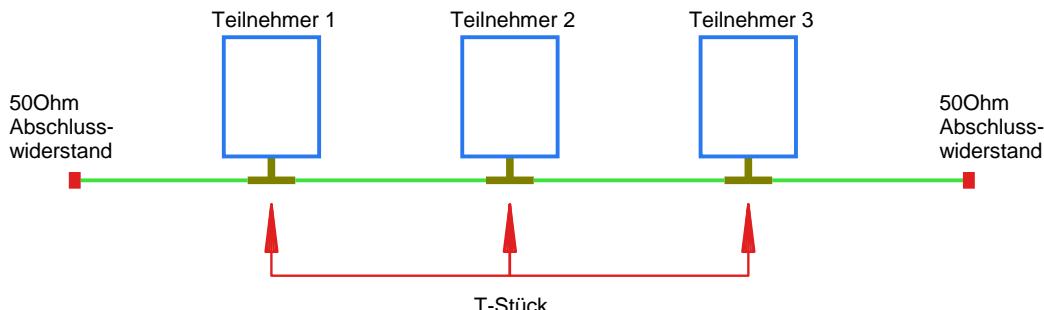


Abbildung 197 Koax Ethernet

Die Verbindung aller Teilnehmer über ein einziges Kabel war sehr fehleranfällig. Falls an einer einzigen Stelle ein Problem auftauchte, ist die Kommunikation für das gesamte Netzwerk nicht mehr funktionsfähig.

Die Übertragungsrate liegt bei dieser Art Netzwerkverbindung bei 10 MBit/s. Diese müssen sich jedoch alle Teilnehmer untereinander teilen, welche auch noch nur als Halbduplex zur Verfügung steht. Die maximale Länge zwischen 2 Teilnehmern beträgt 185 m.

Der Signalpegel auf dem Koaxkabel beträgt 0 und -2.2 V mit Manchester-Kodierung. Eine Kollision wird erkannt durch einen noch tieferen Pegel. Die belegte Frequenzbandbreite auf dem Kabel beträgt ca. 10 MHz.

Twisted Pair 2x2

Durch die Einführung der Twisted-Pair-Kabel (anfangs CAT3 Kabel, einfache Telefonleitungen) mit RJ45 Steckern und jetzt sternförmigen Verbindungen wurden viele Probleme gelöst, die früher durch die Koax-Verbindungen vorhanden waren. Die Signale zwischen den Teilnehmern wurden mittels eines aktiven Gerätes (Hub oder Switch) vermittelt.

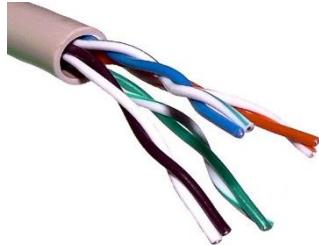


Abbildung 199 Twisted Pair Kabel



Abbildung 198 Ethernet Buchse mit Übertrager

Die Übertragung erfolgt jetzt mit ± 2.5 V, symmetrisch, full-duplex mit je einem Adernpaar pro Richtung. Die anderen beiden Aderpaare des Kabels/Steckers waren vorerst ungenutzt. An jedem Paar des Kabelendes ist ein Übertrager angeschlossen. Bei den meisten Netzwerkgeräten sind die Übertrager in die Netzwerkbuchse integriert.

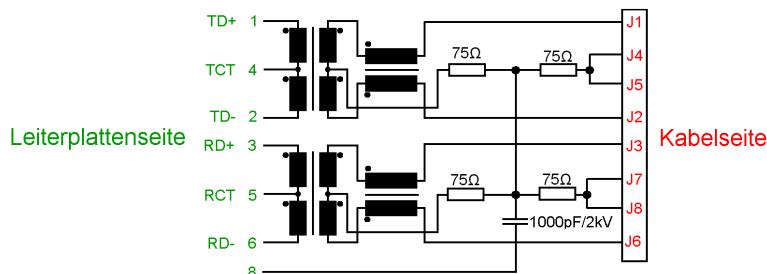


Abbildung 200 RJ45 Übertrager

Manchmal sind sie auch in einem IC auf der Leiterplatte „versteckt“.



Abbildung 201 Ethernet-Übertrager in IC

Der Umstieg auf 100 MBit/s wurde hauptsächlich durch ein verbessertes Kabel (CAT5) und eine verbesserte Codierung erreicht, es reichten dadurch immer noch 2 Aderpaare aus. Bei der Übertragung wird jetzt mit 3 Spannungspiegeln (PAM-3) gearbeitet +1 V/0 V/-1 V (differenziell am Empfänger +2 V/0 V/-2 V), wodurch die belegte Frequenzbandbreite (31,25 MHz) auf dem Kabel nicht so stark anwächst.

Twisted Pair 4x2

Erst beim Übergang auf 1 GBit/s werden alle 4 Adernpaare genutzt. 1 Gbit/s wurde durch mehrere Änderungen erreicht. Die Übertragung erfolgt jetzt Fullduplex auf allen Aderpaaren gleichzeitig in beiden Richtungen. Das wird erreicht durch eine Echokompensation an beiden Enden des Kabels (gleiches Prinzip wie ursprünglich bei analogen Telefonleitungen). Die zweite wichtige Änderung ist der Übergang zu 5 unterschiedlichen Signalpegeln (PAM-5) auf den Leitungen +1 V/+0,5 V/0 V/-0,5 V/-1 V (differenziell am Empfänger +2 V/+1 V/0 V/-1 V/-2 V). Dadurch wird der Durchsatz um ca. Faktor 2,5 gesteigert. Das Signal auf dem Kabel wird immer analoger, wodurch die benötigte Frequenzbandbreite (62,5 MHz) nicht so stark ansteigt. Bei 2,5/5/10 Gbit/s wird mit 16 Stufen (PAM-16) gearbeitet.

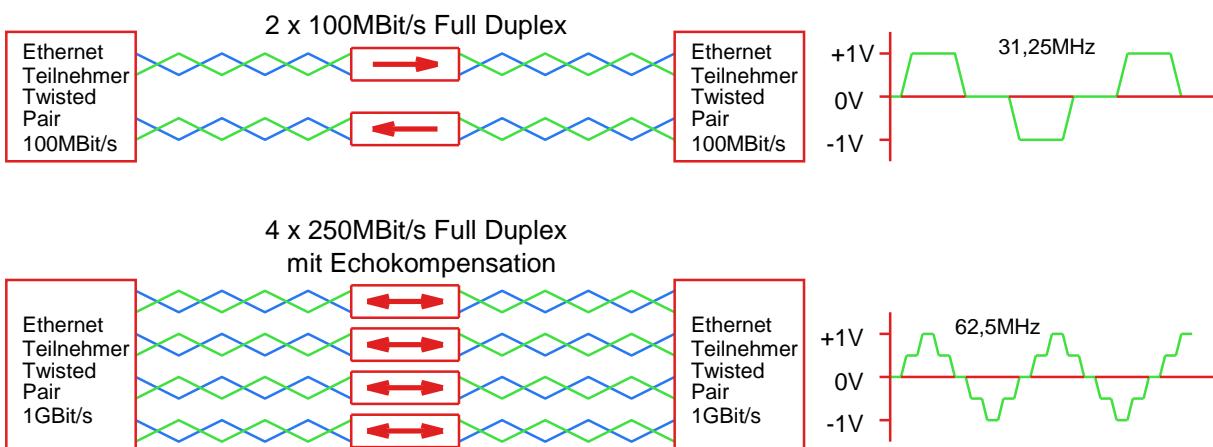


Abbildung 202 Vergleich Ethernet 100 Mbit/s und 1 Gbit/s

ix-Industrial

Ein wichtiger Schritt war auch vor allem im industriellen Bereich eine Überarbeitung der Steckverbinder, da die RJ45 Stecker nach über 50 Jahren mittlerweile eine Schwachstelle darstellen.

Für industrielle Geräte wurde 2018 ein neuer Steckverbinder eingeführt, welcher weniger als die Hälfte des vorherigen Platzes belegt.



Abbildung 203 Ethernet Industriestecker



Abbildung 204 Vergleich RJ45 <-> ix

Eine weitere Erweiterung bei Twisted-Pair ist die gleichzeitige Versorgung des Clients mit Energie mittels Phantomspeisung PoE (Power over Ethernet).

Twisted Pair 1x2

Neueste Entwicklungen (angefangen ab ca. 2015) verwenden im industriellen Bereich nur noch ein Adernpaar SPE (Single Pair Ethernet) und erreichen durch neuere Kabel CAT7 heute auch bis 1 Gbit/s. Die maximale Kabellänge sinkt jedoch dann auf 15-40 m.

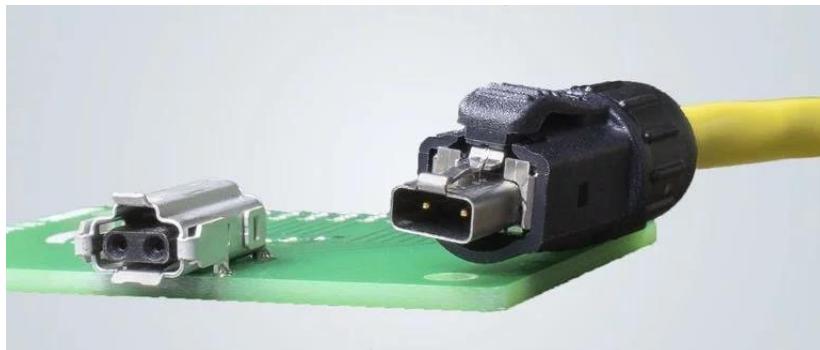


Abbildung 205 Single Pair Ethernet

Mit niedrigeren Datenraten (10 Mbit/s) sind auch bis zu 1000 m möglich. Die neuen endgültigen Standards wurden 2020 festgelegt.

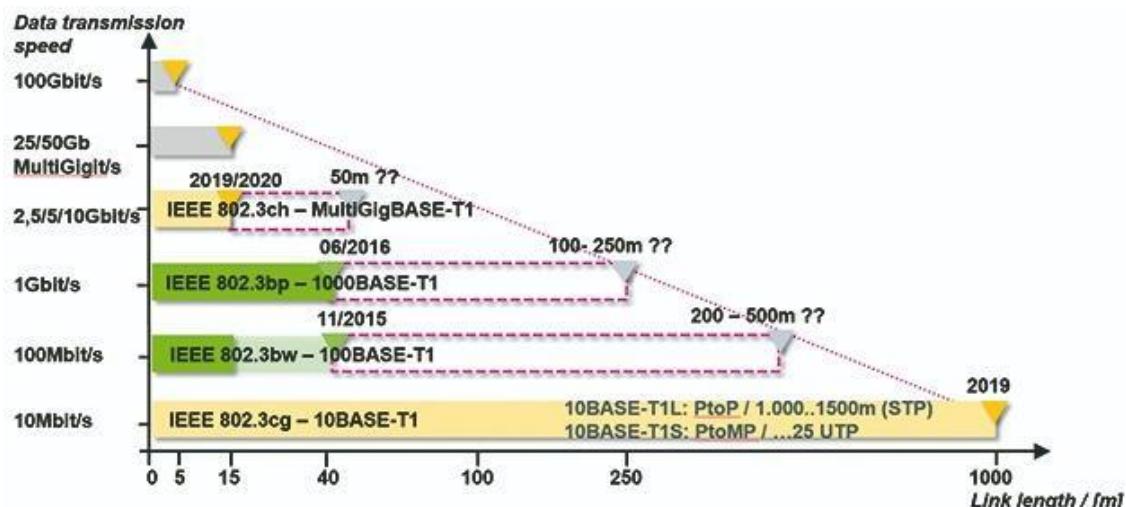


Abbildung 206 Reichweiten SPE

Wie auch bei mehrpaarigen Ethernet-Verbindungen (PoE) ist auch hier eine gleichzeitige Energieversorgung möglich. Diese Funktion nennt sich hier PoDL (Power over Data Line). Es sind hier geregelte oder ungeregelte Spannungen von 12-48 V möglich (Leistungsgrenze maximal von 0,5-50 W).

Diese Lösung hat das Potenzial alle bisherigen industriellen Bussysteme und Feldbusse für Sensorverbindungen zu ersetzen!

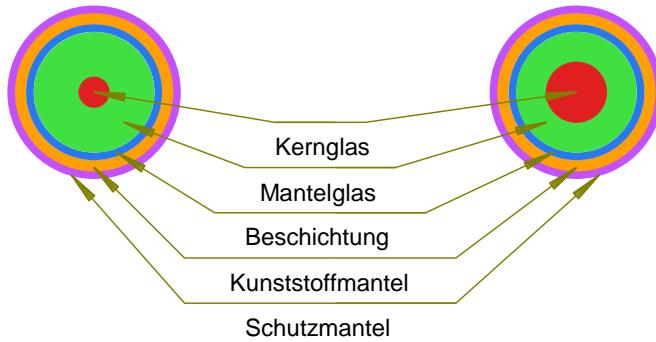
In den Standard wurde auch gleich die Möglichkeit für die Integration in den explosionsgefährdeten Bereich (ATEX) eingebaut.

Glasfaser

Parallel zu der elektrischen Übertragung auf Kupferkabeln wurde fast jedes Mal ein Standard geschaffen, um die Signale optisch über Glasfaser zu übertragen. Die Standards im Computernetzwerkbereich haben 1992 begonnen mit 10 MBit/s, sie sind jedoch heute nur bei Übertragungsraten über 1 GBit/s sinnvoll, weil die notwendigen Kabel und die Verbindungstechnik hier teilweise teurer werden als der Umstieg auf Glasfaserkabel.

Glasfasern gibt es in verschiedenen Ausführungen Monomode-, Multimode - und Gradientenindexglasfasern. Für Entferungen bis einige 100 m werden meist Multimodeglasfasern verwendet. Wenn Entferungen von einigen Kilometern überbrückt werden sollen, sind fast immer Monomodefasern im Einsatz. Die Dicke der Fasern liegt im μm -Bereich bei ca. 100-200 μm , je nach Fasertyp.

Monomode-Glasfaser

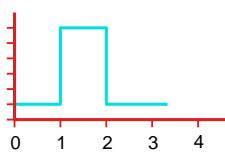
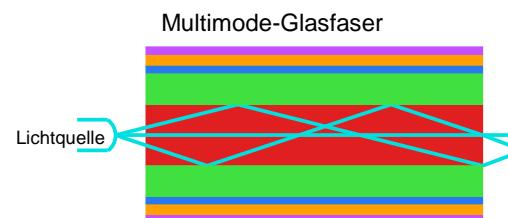
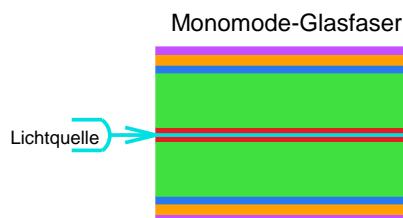


Multimode-Glasfaser

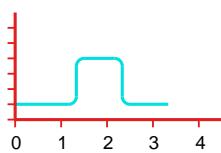
Der Kern einer Monomodefaser ist nur einige μm dick. Die Verbindungstechnik für Monomodeglasfasern ist entsprechend mechanisch sehr aufwendig und teuer. Im Netzwerkbereich werden deshalb fast ausschließlich Multimodefasern eingesetzt.

Abbildung 207 Glasfaser Typen

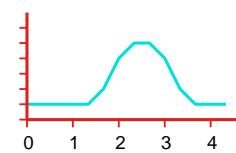
Das verwendete Licht liegt fast ausschließlich im Infrarotbereich mit einer Wellenlänge von 850 nm-1550 nm. Die Übertragung erfolgt meistens im Fullduplex mit je einer Glasfaser für jede Richtung. Es gibt auch Full duplex-Übertragungen auf nur einer Glasfaser, was oft mit unterschiedlichen Wellenlängen für jede Richtung gelöst wird.



Eingangssignal



Monomode Ausgangssignal



Multimode Ausgangssignal

Abbildung 208 Lichtausbreitung in Glasfasern

Der Vorteil einer Monomodefaser ist eine nahezu perfekte Signalübertragung. Die Multimodefaser verschleift und verbreitert (Dispersion) den Eingangsimpuls, was die maximale Übertragungsraten bei längeren Leitungen stark begrenzt.

Um von einer 1 GBit/s-Kupferleitung auf eine Glasfaserleitung zu wechseln, gibt es relativ einfache Medienkonverter von vielen Herstellern.



Abbildung 209 Medienkonverter für 1 Gbit-Full duplex-Glasfasern

Ab 10 GBit/s gibt es standardisierte Transceiver-Module, die direkt in den jeweiligen Netzwerkport gesteckt werden können.



Abbildung 210 10 GBit/s Tranceiver-Modul

TOSLINK

Die oft sehr bekannte, optische TOSLINK (**TOShiba-LINK**) S/PDIF Verbindung (**Sony/Philips Digital Interface**) aus der Consumer-Audientechnik ist keine Glasfaserübertragung. Hier wird immer rotes Licht mit ca. 650 nm Wellenlänge über einen Kunststofflichtwellenleiter übertragen. Die Übertragungsrate liegt hier üblicherweise deutlich unter 10 MBit/s. Der Notwendigkeit für eine optische Übertragung ist fast ausschließlich in der galvanischen Trennung und der geringeren elektromagnetischen Störbeeinflussung begründet.



Abbildung 211 TOSLINK

6.3.9.2 Drahtlos

Allgemeines

Bei Funkverbindungen ist zu beachten, dass sich unter allen Teilnehmer eines Frequenzkanals die gesamte Bandbreite aufteilt. Die Übertragung ähnelt somit der anfänglichen Ethernet-Übertragung auf einem gemeinsamen koaxialen Kabel (siehe Kapitel 6.3.9.1), wodurch Kollisionen entstehen können, wenn 2 Sender gleichzeitig senden.

WLAN

WLAN (Wireless Local Area Network) wurde in den 1990er-Jahren entworfen als drahtlose Erweiterung für ein Ethernet Netzwerk. Es wurde im Prinzip nur der entsprechende OSI-Layer 1 ausgetauscht. Der Standard hat einige Evolution hinter sich, und es sind für diverse Länder unterschiedliche Frequenzbereiche festgelegt.

Der erste und heute meistens genutzte Frequenzbereich ist 2,4-2,5 GHz. Der Bereich wurde in mehrere Frequenzbänder von 20 MHz Breite unterteilt. Direkt nebeneinanderliegende Bänder überlappen sich, wodurch eine Störbeeinflussung möglich wird.

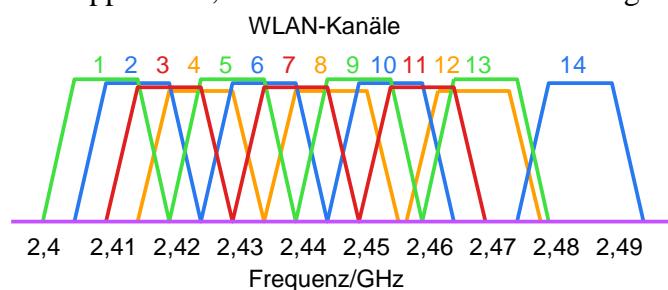


Abbildung 212 WLAN Überlappung

Die aktuelle, maximale Brutto-Übertragungsrate bei einem störungsfreien Kanal liegt bei 54 Mbit/s (netto ungefähr die Hälfte). Einige Geräte verwenden eine doppelte Kanalbreite von 40 MHz, wodurch sich die Nutzdatenrate etwa verdoppelt. Die maximale Leistung, mit der in Europa gesendet werden darf, liegt bei 100 mW (USA 1 W).

2009 wurde der Standard um ein 2. Frequenzband erweitert im Bereich von 5,1 bis 5,8 GHz. Gleichzeitig wurde das MIMO Verfahren (**M**ultiple **I**nput **M**ultiple **O**utput) definiert. Hier wird mit mehreren Antennen versucht, mehrere Übertragungswege zum Empfänger zu finden. Das ist vor allem innerhalb von Gebäuden sinnvoll, da hier Reflexionen durch Wände die Übertragung stark beeinflussen können.

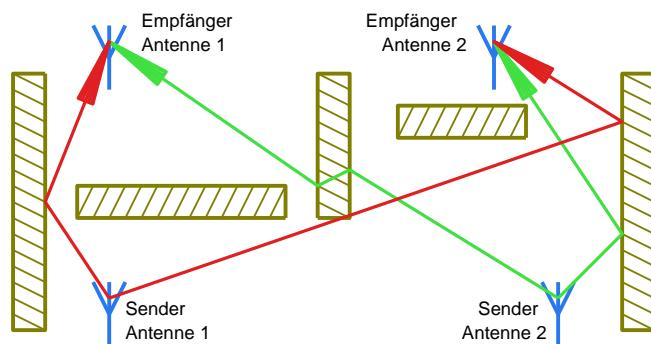


Abbildung 213 MIMO

Bluetooth

Bluetooth wurde 1999 gestartet und war in erster Version für eine Funkübertragung für Distanzen von einigen Metern gedacht. Mit Bluetooth sollten hauptsächlich kurze Kabelverbindungen ersetzt werden. Es ist aber auch möglich kleine Netzwerke damit aufzubauen. Bluetooth wird heute oft als drahtlose Verbindung für Audiogeräte genutzt.

Der Frequenzbereich liegt bei 2,45 GHz. Ab Version 1.2 wurde ein Adaptives-Frequenz-Hopping-Verfahren eingeführt, um die Empfindlichkeit gegenüber statischen Störern zu verbessern.

Bei der Sendeleistung wird in 3 Klassen unterteilt:

	Leistung	Reichweite
Klasse 1	100 mW	100 m
Klasse 2	2,5 mW	10 m
Klasse 3	1 mW	1 m

Bluetooth hat einiges an Evolution hinter sich, und es existieren verschiedene Versionen, die teilweise nicht miteinander vollständig kompatibel sind.

Die Datenrate begann bei 732 kBit/s und liegt heute üblicherweise bei maximal 2,1 MBit/s.

Ab Bluetooth 3.0 ist bei Nutzung eines High-Speed-Kanals (bis 24 MBit/s) eine gegenseitige Beeinflussung von WLAN möglich, da sich die genutzten Frequenzbänder teilweise überlappen.

Ab Version 4.0 ist durch die Einführung eines Low-Energy-Modes der Einsatz für einfache Funksensoren mit Batteriebetrieb möglich. Mit Bluetooth 4.0 kann eine Verbindung in weniger als 5 Millisekunden auf- und abgebaut werden.

Ab Version 5.0 ist eine dynamische Aushandlung der Sendeleistung oder der Datenrate möglich. Weiterhin wurde auch eine Entfernungsmessung für eine Indoor-Gebäude-Navigation integriert.

Sonstiges

Der freie ISM-Bereich (Industrial, Scientific and Medical) 868 MHz (Europa) oder 915 MHz (USA) wird von einigen Herstellern proprietär oder einheitlich genutzt. Viele Drahtlosthermometer nutzen diesen Bereich.

Des Weiteren existieren noch weitere standardisierte Funknetzwerke, die meist für Sensornetzwerke in der Industrie oder für die Gebäudeautomation genutzt werden.

ZigBee ist eine häufig genutzte Spezifikation für drahtlose Netzwerke mit geringem Datenaufkommen und nutzt die Frequenzbereiche 868 MHz (Europa) oder 915 MHz (USA) und zusätzlich 2,4 GHz zur Kommunikation. Die Datenrate liegt hier zwischen 20 kBit/s und 250 kBit/s. Die ZigBee-Teilnehmer können miteinander vermascht werden, womit eine Reichweite von mehreren Kilometern möglich ist.

7 Displays

7.1 Allgemeines

Um Computerergebnisse darzustellen, ist es üblich, diese optisch darzustellen.
Je nach Anwendungsfall gibt es verschiedene Ausbaustufen:

- Einzelne Leuchtdioden
- Gruppierte Leuchtdioden
- 7-Segment-Anzeigen
- Passive LED Punktmatrixanzeigen
- LC-Anzeigen ohne Controller
- LC-Anzeigen mit Controller
- Vollständige Bildschirme

7.2 Einfache Displays

7.2.1 Leuchtdioden

7.2.1.1 Einzelne Leuchtdioden

Um eine Leuchtdiode anzusteuernd, ist ein einfacher Port-Pin eines Mikrocontrollers ausreichend. Es gibt 2 Möglichkeiten zur Ansteuerung: High und Low-aktiv.

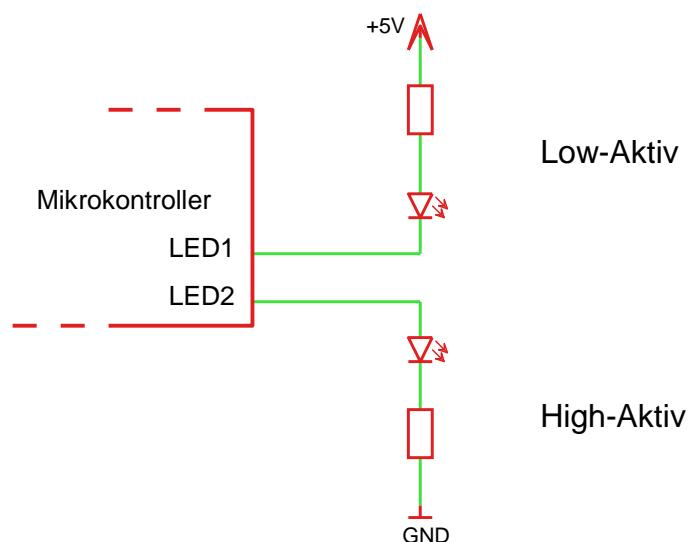


Abbildung 214 LED-Ansteuerung

Viele Digitalausgänge können bei Low-Pegel wesentlich mehr Strom aufnehmen, als sie bei High-Pegel liefern können, deshalb ist meist die Low-Aktiv-Version besser geeignet.

Bei LEDs mit unterschiedlichen Farben ist die Durchlassspannung der LEDs zu beachten:
Blaue LEDs haben oft eine Durchlassspannung über 3 V. Bei Controllern mit nur 3 V Versorgungsspannung, sind blaue LEDs nicht geeignet.

Die neuen, extrahellen, grünen LEDs mit der Farbe True-Green verhalten sich fast genauso wie blaue LED-Chips. Bei weißen LEDs sind immer blaue LED-Chips enthalten.

7.2.1.2 Gruppierte Leuchtdioden

Einfache Matrix

Falls eine größere Zahl LEDs anzusteuern ist, werden diese üblicherweise mittels einer gemultiplexten Matrix angesteuert.

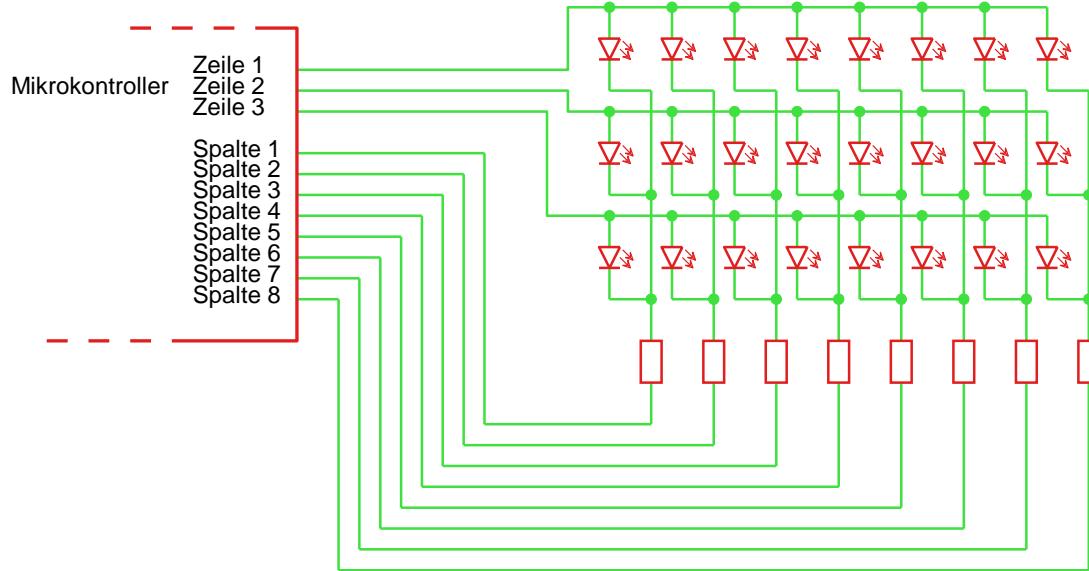


Abbildung 215 LED-Multiplexing

Dabei ist zu beachten, dass die LEDs immer nur eine kurze Zeit eingeschaltet sind. Um eine ausreichende Helligkeit zu erreichen, muss der Strom kurzzeitig sehr hoch sein. Manche Matrix-LEDs vertragen kurzzeitige Ströme von einigen Ampere.

7-Segment

Eine häufige Anordnung von gruppierten LEDs sind mehrstellige 7-Segment-Anzeigen. Diese gibt es in zwei Ausführungen: Gemeinsame-Kathode oder Gemeinsame-Anode.

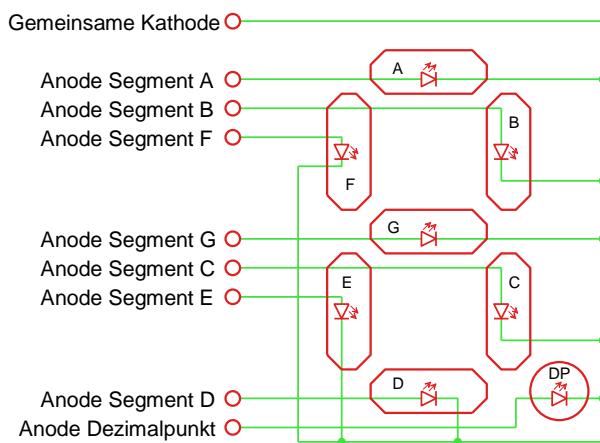


Abbildung 216 Gemeinsame Kathode

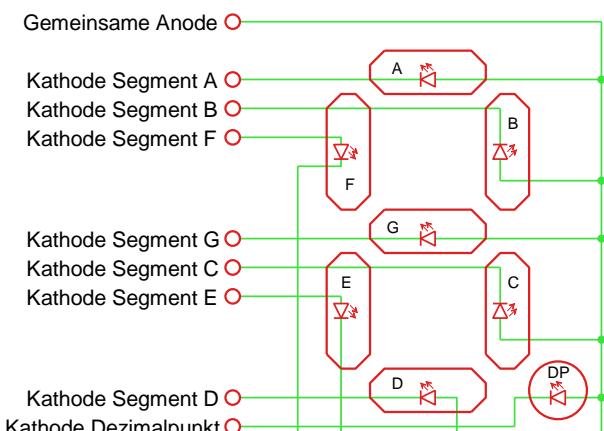


Abbildung 217 Gemeinsame Anode

Durch den gemeinsamen Anschluss kann der 8-fache Strom fließen, wenn alle Segmente leuchten sollen. Deshalb ist es meistens notwendig, die Ausgangspins des ansteuernden ICs mittels Transistoren zu verstärken. Dabei ist die Variante mit Gemeinsamer-Anode meistens besser geeignet. Die Ausgangspins der meisten Mikrocontroller sind in der Regel stark genug um die Kathoden der LEDs ohne Transistor direkt anzusteuern.

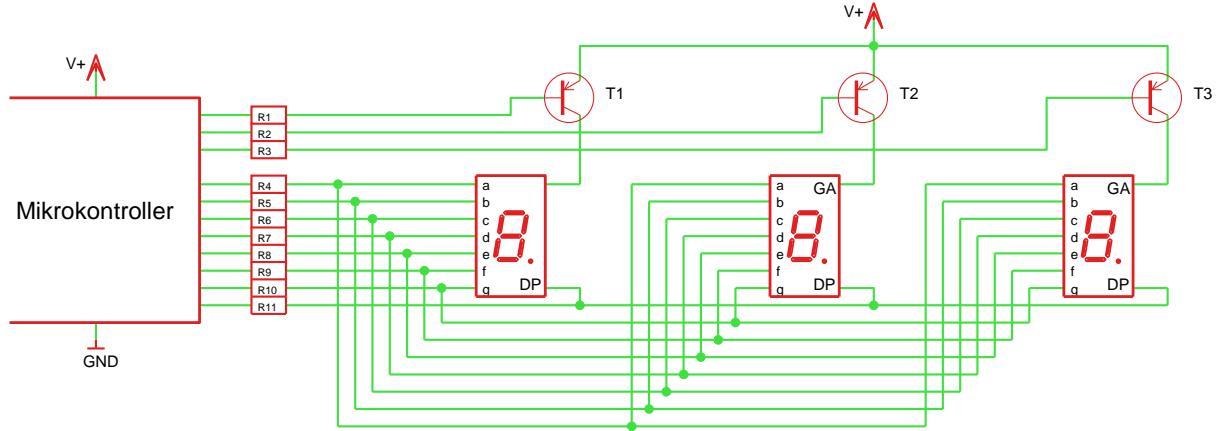


Abbildung 218 7-Segment mehrstellig

Die Versorgung des Mikrocontrollers muss hier mit der Versorgung der LEDs übereinstimmen, da sonst die Transistoren nicht korrekt gesperrt werden können.

7.2.2 LC-Anzeigen

7.2.2.1 Grundprinzip

Das Grundprinzip von (LCD= Liquid-Crystal-Display) Flüssigkristall-Anzeigen beruht auf der Eigenschaft, dass Flüssigkristalle die Polarisationsrichtung von Licht beeinflussen.

Als einfaches Beispiel ist hier der Aufbau einer TN-Zelle (Twisted Nematic) beschrieben.

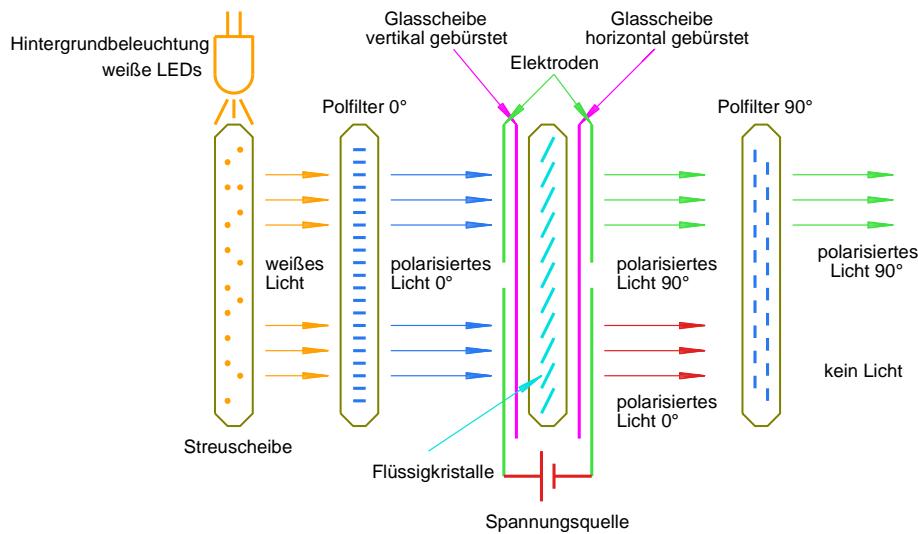


Abbildung 219 LCD-TN-Zelle

Das Licht gelangt über die Streuscheibe zum ersten Polfilter. Danach ist nur noch polarisiertes Licht vorhanden (50 % Verlust).

Die Flüssigkristalle sind zwischen zwei Glasscheiben eingebettet, auf die auf der Innenseite eine senkrechte oder waagerechte Oberflächenstruktur aufgebracht wurde. Dadurch legen sich die Flüssigkristallpartikel automatisch bei einer Scheibe senkrecht und auf der anderen waagerecht an. Dazwischen findet ein schraubenförmiger Übergang statt. Diese Anordnung dreht die Polarisationsebene des Lichts um 90°. Der nachfolgende um 90° gedrehte Polfilter lässt das gedrehte Licht dann passieren.

Wird jetzt eine Spannung an eine Zelle angelegt (Pixel=Aus), wird die schraubenförmige Anordnung zerstört und die Polarisationsebene des Lichts wird nicht mehr gedreht. Der nachfolgende um 90° gedrehte Polfilter lässt das gedrehte Licht dann nicht mehr passieren.

Als Lichtquelle für das LCD dienen heutzutage meistens mehrere weiße LEDs, früher (bis ca. 2010) wurden hier Kaltkathodenleuchtstofflampen mit Quecksilberdampf eingesetzt. Seit dem LED-Einsatz hat sich die Farbdarstellung etwas verbessert (Marketing Buzzword: LED-Monitor/TV).

Für eine farbige Darstellung wird bei aktueller Technologie bei jedem Pixel noch ein Farbfilter zwischen dem ersten Polfilter und der Flüssigkristallzelle eingefügt. Das führt leider dazu, dass der Wirkungsgrad der Anordnung nochmals um 2/3 fällt, da die beiden anderen Komponenten des RGB-Anteils immer weggefiltert werden müssen.

Neuere Entwürfe benutzen blaues Licht als Hintergrundbeleuchtung und konvertieren die Farbe der grünen und roten Pixel über Fluoreszenzstoffe, oder mithilfe von Quantenpunkten (Marketing Buzzword: QLED-Monitor/TV), was zu einem deutlich besseren Wirkungsgrad und einer noch besseren Farbdarstellung führt.

Als nächste Ausbaustufe ist angekündigt, die Farbkonvertierung auf der anderen Seite der Flüssigkristallzelle zu machen, wodurch die Blickwinkelabhängigkeit fast vollständig entfernt werden wird.

7.2.2.2 *Passive LCDs*

Passive LCDs erlauben normalerweise keine farbige Darstellung und einzelne Pixel können nur 2 Helligkeitszustände annehmen: Ein/Aus. Es gibt sie in vielen Standardausführungen und sie werden sehr häufig in allen möglichen Geräten eingesetzt, da sie sehr preisgünstig sind und die Ansteuerung relativ einfach ist.

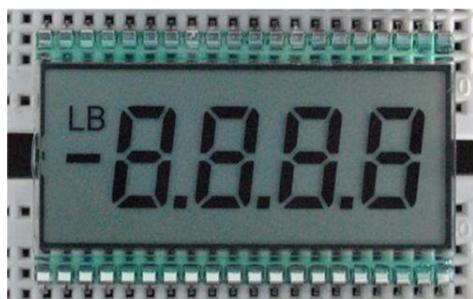


Abbildung 220 Passiv LCD

Bei einfachen z.B. 4-stelligen Standarddisplays ist z.B. jedes Segment auf einen extra Pin herausgeführt und kann direkt gegenüber einem Backplanesignal angesteuert werden. Wenn viele Segmente vorhanden sind, gibt es dann mehrere Backplanesignale für mehrere Segmentgruppen.

Sehr wichtig ist, dass an die Segmente keine dauerhafte Gleichspannung angelegt werden darf, da sonst bei den Flüssigkristallen irreversible elektrochemische Prozesse stattfinden und sie dadurch langsam zerstört werden.

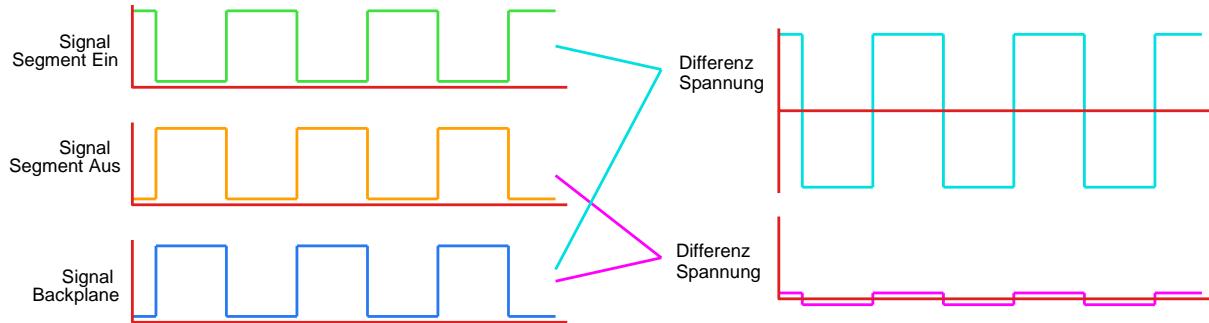


Abbildung 221 LCD-AC-Signal

Das wird dadurch erreicht, dass man eine pulsierende Gleichspannung mit max. 1 kHz an die Backplane anlegt und an die Ein-Segmente die invertierte Spannung. Die Flüssigkristallsegmente „sehen“ dadurch immer eine echte (Differenz) Wechselspannung.

7.2.2.3 LCDs mit integrierter Ansteuerung

Bei den aktiven LCDs gibt es viele verschiedene Typen. Viele Hersteller haben sich auf das von Hitachi entworfene HD44780-Interface geeinigt. Auch wenn heute keine Controller von Hitachi mehr bestückt sind, sind alle voll kompatibel zu Diesem.

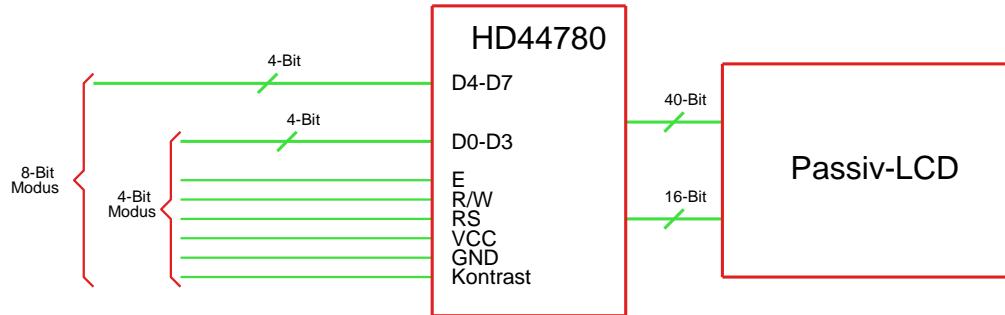


Abbildung 222 LCD-Controller

Der Controller hat einen 4-Bit-Modus, bei dem man für die Ansteuerung nur insgesamt 7 Leitungen benötigt.

An die Kontrastleitung muss bei vielen Typen eine echte einstellbare negative Spannung angelegt werden. Diese kann meistens über eine einfache Ladungspumpe erzeugt werden.

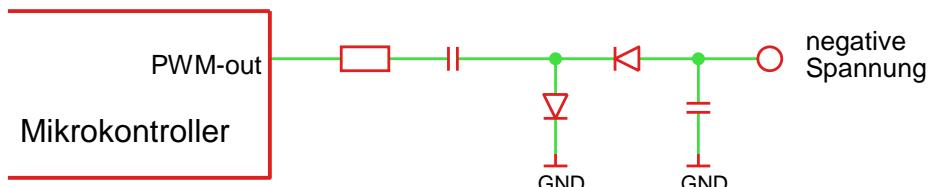


Abbildung 223 Ladungspumpe

Bei den meisten neueren, grafischen LCD-Modulen auf dem Markt hat sich die serielle SPI-Schnittstelle (siehe Kapitel 6.3.3) durchgesetzt.

7.3 Komplette Bildschirme

7.3.1 Grundlegendes

Die Bildausgabe auf einen Bildschirm wird aus historischen Gründen aus der Zeit der Bildröhren zeilenweise von links nach rechts und von oben nach unten durchgeführt. Zuerst mit einem Elektronenstrahl für eine einfarbige (anfangs nur grün) Darstellung. Später dann in Farbe mit drei Elektronenstrahlen und additiver Farbmischung der drei Grundfarben Rot, Grün und Blau. Der Elektronenstrahl ist normalerweise etwas kleiner als ein Pixel der Mattscheibe und damit war es möglich, auch halbe Pixel zum Leuchten zu bringen. Das hat in der Vergangenheit bei der Ansteuerung einer Bildröhre mit nicht nativer Auflösung zu einer relativ guten brauchbaren automatischen Interpolation geführt.

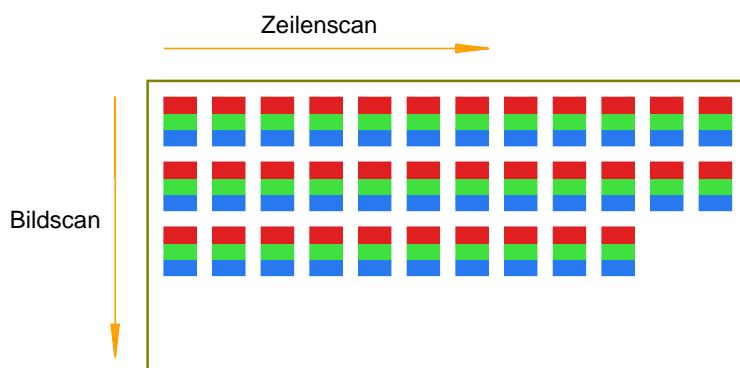


Abbildung 224 Bildaufbau

Früher wurde als Ausgabemedium immer eine Kathodenstrahl-Bildröhre benutzt, die eine relativ kurze Nachleuchtdauer (ca. 2 ms) hatte. Um Großflächenflimmern zu verhindern, musste die Bildwiederholfrequenz deshalb mindestens 70 Hz betragen, damit der Benutzer vor dem Bildschirm keine Kopfschmerzen bekommt. Bei heutigen Flachbildschirmen hat man sich auf eine Bildwiederholfrequenz von 60 Hz geeinigt, da diese durch ihre Konstruktion prinzipiell keine Flimmererscheinungen mehr erzeugen.

Am rechten und unteren Rand sind noch ca. 20 % mehr Pixel enthalten, als auf dem Bildschirm angezeigt werden. Das war früher notwendig, da Kathodenstrahl-Bildröhren etwas Zeit für einen Zeilenrücklauf oder Bildrücklauf benötigten.

Die Daten, welche den Bildschirminhalt enthalten, sind in einem Grafikspeicher-RAM gespeichert, welches entweder auf einer Grafikkarte sitzt, oder bei manchen Rechnerarchitekturen vom Hauptspeicher des Systems abgezweigt wird (Shared Memory siehe Kapitel 7.4.2.4).

Die Ausgabe der Daten an den Bildschirm erfolgt über einen Grafikcontroller, der den Inhalt des gesamten Grafikspeichers, der die Pixeldaten enthält, zyklisch an den Bildschirm sendet. Bei einer Bildschirmauflösung von 1920x1080 Pixel und einer Bildwiederholfrequenz von 60Hz bei 8-Bit pro Pixelfarbe ist eine Speicherbandbreite für den Bildtransfer von konstant $1920 \times 1080 \times 60\text{Hz} \times 3 \text{ Byte} = 373 \text{ Mbyte/s}$ notwendig.

Früher war auf Grafikkarten nur der Speicher für den Bildschirminhalt vorhanden. Dieser ist mit preisgünstigem DRAM-Speicher aufgebaut. Der aufwendige Refresh-Zyklus fällt hier komplett weg, weil der Speicher ständig zyklisch ausgelesen wird, was sozusagen einem Refresh entspricht.

Heutzutage ist auf modernen Grafikeinheiten ein komplettes Computersystem mit spezieller CPU (GPU) enthalten, welche darauf optimiert ist, viele Rechnungen parallel abzuarbeiten. (siehe Kapitel 7.4.2)

7.3.2 VGA

VGA (Video Graphics Array) ist der letzte noch gültige analoge Standard für PC-Bildschirmausgaben. Er wurde von IBM im Jahr 1987 geschaffen, und hat die damals üblichen binären Grafikausgaben ersetzt. Zuvor wurden meist monochrome Monitore mit zwei Helligkeitsstufen genutzt. Die analoge, farbige Ausgabe auf einen Monitor wurde mit dem VGA-Standard erstmals möglich. Das amerikanische NTSC-Fernsehsystem hat eine sichtbare Bildauflösung von 640x480 Pixel. Beim Entwurf des VGA-Systems wurde darauf geachtet, als kleinsten gemeinsamen Standard genau diese Auflösung zu wählen. Die Fernsehnorm arbeitet jedoch im Zeilensprungverfahren, welches für Computerbildschirme wegen des Kantenflackerns nicht gut geeignet ist. Die Bildwiederholfrequenz lag zwar bei 60 Hz, durch das Zeilensprungverfahren ergeben sich dann jedoch nur effektive 30 Hz. Höhere Auflösungen waren früher nur mit Zeilensprung möglich, da die erforderliche Bandbreite des Speichers, Kabels und der Monitore nicht ausgereicht hat.

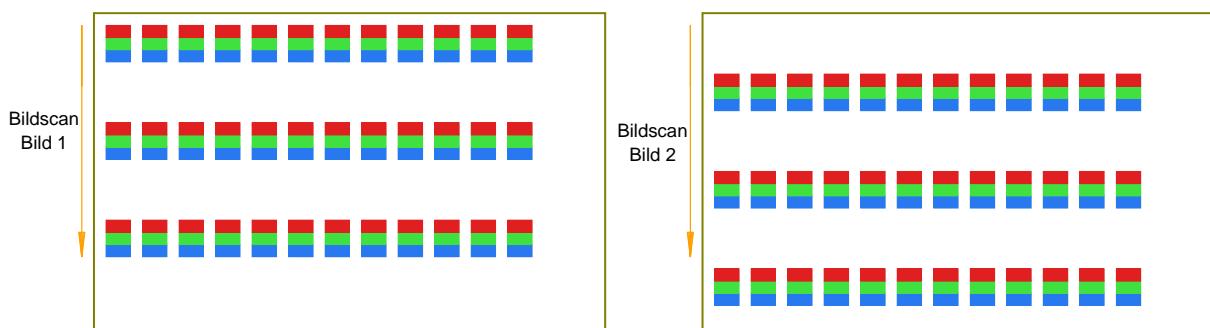


Abbildung 225 Zeilensprung

Die Daten des Grafikspeichers (oft als Dual-Port-RAM ausgeführt) werden für jede Farbe getrennt Pixel für Pixel vom Grafikcontroller ausgelesen und mittels eines D/A-Wandlers (RAMDAC) nach analog umgewandelt. Zusätzlich zu den drei Farben Rot, Grün, Blau, werden noch die zwei zusätzlichen Signale H-Sync und V-Sync erzeugt. Diese sind notwendig, um den Anfang jeder Zeile und den Anfang des gesamten Bildes zu kennzeichnen.

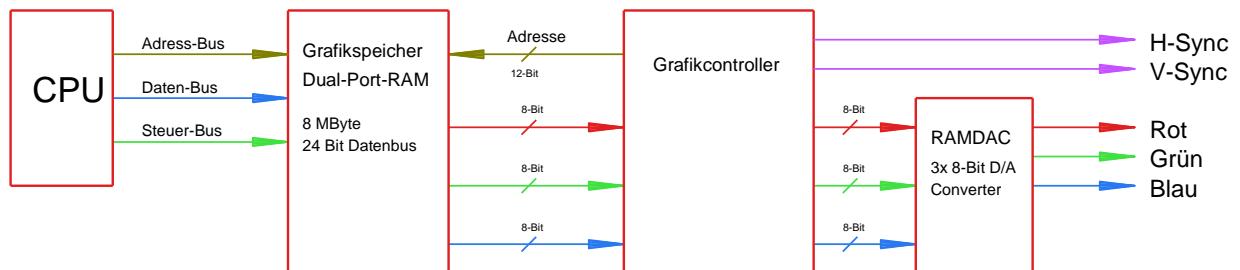


Abbildung 226 RAMDAC

Bei moderneren VGA-Schnittstellen werden zu den Bildsignalen noch zusätzliche Informationen zwischen den Geräten übertragen. Dazu werden einige Leitungen des 15-poligen DSUB-Steckers als einfache serielle Schnittstelle genutzt (DDC-Signal, Display Data Channel).

7.3.3 DVI

Als Ende der 1990er-Jahre die ersten Flachbildschirme auf den Markt kamen, war es sehr umständlich, diese mit dem damals üblichen analogen VGA-Signal anzusteuern. Die Elektronik in den Flachbildschirmen benötigte für die jetzt notwendige Umskalierung der Pixel die Bilddaten in 2-dimensionaler digitaler Form. Die vom VGA-Grafikkontroller gelieferten analogen Signale mussten deshalb wieder digitalisiert werden. Diese mehrfache Digital-Analog-Digital-Umwandlung gelingt nicht ohne Verluste. Um dieses Problem zu lösen, wurde einige Jahre später die DVI-Schnittstelle (**Digital Visual Interface**) entwickelt.

Die digitale Schnittstelle überträgt mittels synchroner, symmetrischer Übertragung 6 einzelne, differenzielle Signale (TDMS-Links, **Transition Differential Minimized Signaling**) über ein bis zu 10m langes Kabel. Das synchrone Clock-Signal wird hier auch symmetrisch mitübertragen. Für die Nutzdaten gibt es 2 Ausbaustufen. Es gibt Single-Link (3 TDMS-Signale) und Dual-Link-Verbindungen (6 TDMS-Signale), bei denen für höhere Auflösungen die doppelte Bandbreite zur Verfügung steht (7,44 GBit/s).

Zusätzlich ist es möglich, aus Gründen der Rückwärtskompatibilität 4 analoge Signale (RGB+H-Sync) über den Stecker zu übertragen. Von dem Stecker gibt es mehrere Ausführungen:

- DVI-D nur die digitalen Leitungen werden genutzt
- DVI-A nur die analogen Leitungen werden genutzt
- DVI-I analoge und digitale Leitungen werden genutzt

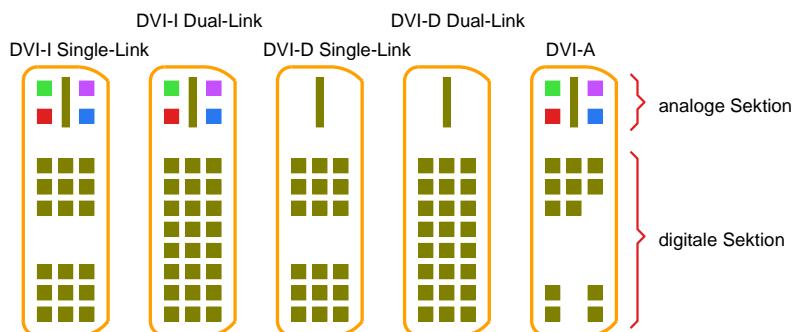


Abbildung 227 DVI-Ausführungen

Die digitale TDMS-Kodierung ist ein 8b10b-Code. Es werden aus dem 10-Bit-Vorrat nur Bitmuster verwendet, die gleich viele Anteile von Einsen und Nullen enthalten, damit die Übertragung immer gleichanteilsfrei bleibt.

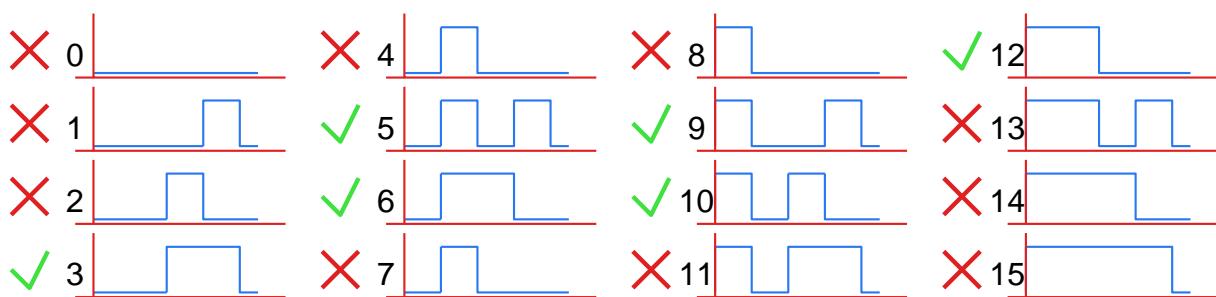


Abbildung 228 Beispiel gleichanteilsfreier 4-Bit-Code

7.3.4 HDMI

Die HDMI-Schnittstelle (**H**igh **D**efinition **M**ultimedia **I**nterface) wurde kurz nach der DVI-Schnittstelle entworfen und war hauptsächlich für die Unterhaltungsindustrie vorgesehen. Zusätzlich zur DVI-Schnittstelle sind ein Kopierschutz und ein digitales Audiosignal hinzugekommen.



Abbildung 229 HDMI Stecker



Abbildung 230 HDMI-DVI Adapter

Die Schnittstelle ist elektrisch kompatibel zur DVI-Schnittstelle, dadurch können passive Adapter verwendet werden. Sie unterstützt allerdings nur Single-Link-Verbindungen. Um trotzdem höhere Auflösungen anzeigen zu können, wurde die HDMI Schnittstelle mehrmals überarbeitet und dabei hauptsächlich die Datenrate erhöht. Die anfangs definierte, maximale Kabellänge ist bei den neueren Versionen nur noch mit sehr hochwertigen Kabeln möglich. Bei HDMI 2.1 werden bis zu 42 GBit/s übertragen. Ab HDMI 2.1 wurden alle Versionsnummern bei HDMI-Kabeln durch Text ersetzt (Standard, High-Speed, Premium-High-Speed, Ultra-High-Speed).

7.3.5 Display Port

Der Display-Port ist dem HDMI sehr ähnlich. Er ist elektrisch sehr nahe am HDMI und DVI. Auch hier sind deshalb einfache Adapter möglich. Er wurde etwas später als HDMI entwickelt und ist mechanisch stabiler. Es ist auch eine Version mit mechanischer Verriegelung verfügbar.



Abbildung 231 Display-Port-Stecker

Mit der aktuellen Version Diplay-Port-2.0 ist eine Übertragungsrate bis zu 80 Gbit/s möglich (128b/132b-Kodierung).

7.4 Grafiksysteme

7.4.1 Ohne Intelligenz

Erste Grafikausgaben bestanden nur aus einem DRAM-Speicher und einer Logik, die den Speicher zyklisch ausliest. Der Speicher war meistens von zwei Seiten gleichzeitig ansprechbar (siehe Kapitel 7.3.2). Bei manchen Rechnerdesigns wurde die gesamte Grafik-Elektronik transparent auch auf das Mainboard mit integriert.

Die Pixel-Daten wurden für eine monochrome Darstellung zuerst digital monochrom übertragen (Hercules Grafik), dann digital in Farbe mit je 2-Bit/Farbe (EGA), später wurden die Pixeldaten über einen Digital-Analog-Konverter als analoges Signal zum Monitor übertragen (VGA). Als letzten Schritt hat man dann wieder zu der digitalen Übertragung gewechselt (siehe Kapitel 7.3.3).

7.4.2 GPUs

7.4.2.1 2D-Beschleunigung

Die vorher in den 1980ern stark verbreitete reine Textdarstellung benötigte bei einer Bildschirmseite von 80x25 Zeichen einen Grafikspeicher von 2 kByte. Die Performance der gesamten Computerarchitektur war damals genau darauf ausgelegt. Als dann versucht wurde, mit derselben Hardware den Bildschirm von grafischen Betriebssystemen zu bedienen, kam es oft zu einem sehr langsamen Bildaufbau. Die CPU musste jetzt bei 640x480 Pixeln ca. die 150-fache Menge an Grafikspeicher bedienen (in Farbe Faktor 450).

Die ersten Funktionen waren einfache 2D-Funktionen, mit denen das Zeichnen von Geraden, Kreisen, Textausgaben und diverse Füllfunktionen erleichtert wurden. Diese werden von einer optimierten Hardware, die direkt mit dem Grafikspeicher verbunden ist, deutlich schneller durchgeführt als von der CPU.

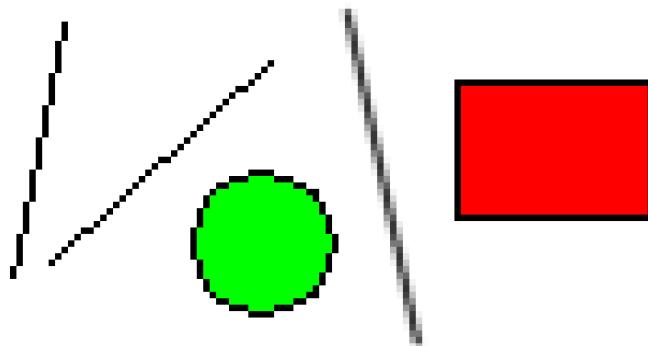


Abbildung 232 2D-Figuren auf Pixelevne

Die Entwicklung von Grafikkarten mit integrierten Rechenfunktionen wurde in den 1990ern sehr stark von Microsoft mit dem 16-Bit-Windows ca. ab Version 3.1 beeinflusst.

Bei den grafischen Betriebssystemen wurden zuerst API-Routinen ausgeführt, die direkt in den Grafikspeicher der Grafikkarte die notwendigen Pixel für z.B. für eine Gerade oder für eine Textausgabe geschrieben haben.

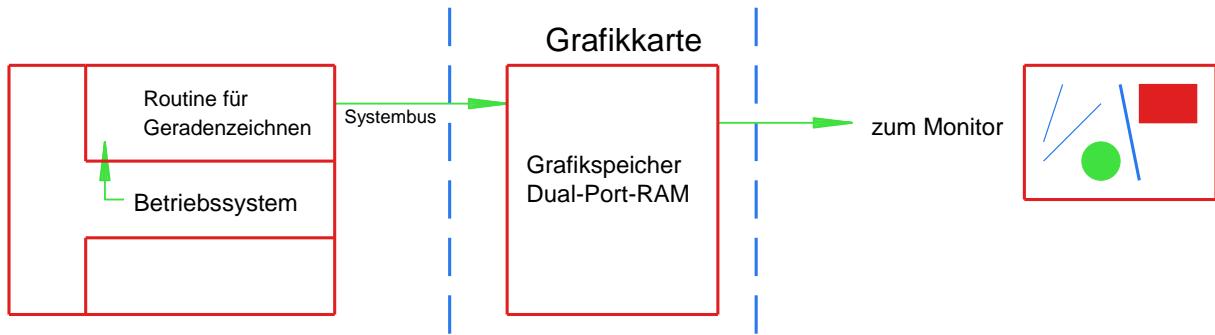


Abbildung 233 Grafik ohne Beschleunigung

Später wurden für verschiedene Betriebssysteme Grafik-Treiber eingeführt, die die alten API-Softwarefunktionen durch direkte Hardwarefunktionen in der Grafikkarte ersetzt haben und dann dort auch direkt ausgeführt wurden. Die CPU transferiert nur noch relativ wenige Vektordaten zur Grafikkarte.

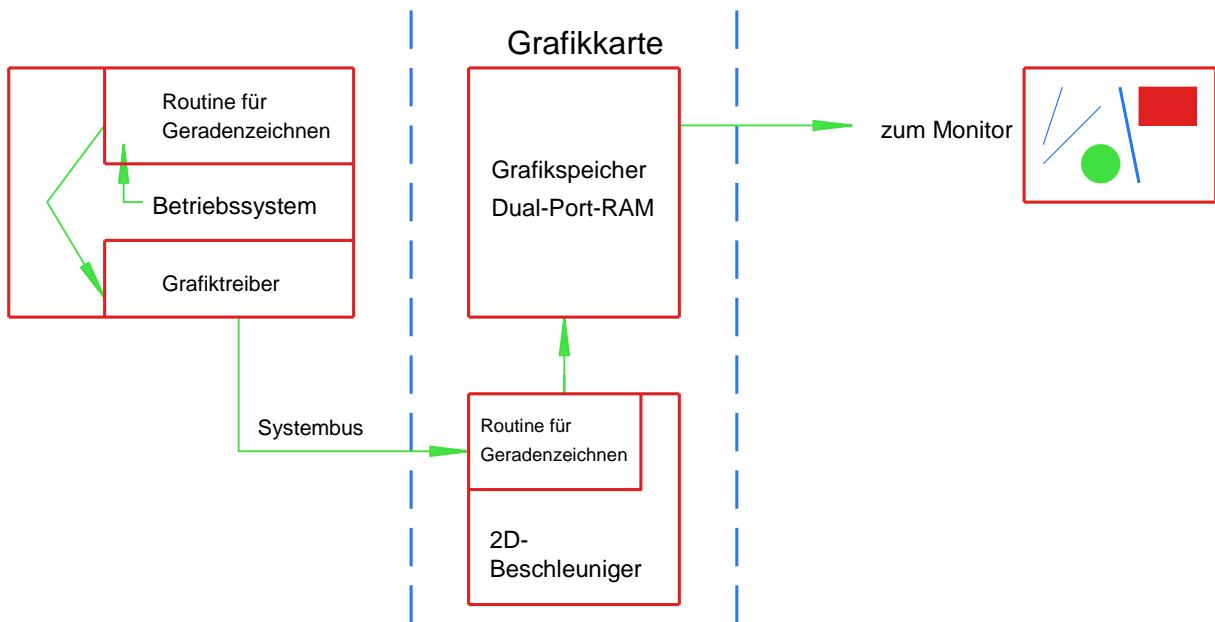


Abbildung 234 2D-Beschleunigung

Durch die direkte Verbindung der Grafikbeschleuniger mit dem Grafikspeicher wurde der Systembus während der Zeichenausgabe nicht belastet und die CPU konnte weiterarbeiten, während die Zeichenfunktion auf dem Grafikbeschleuniger lief.

Zum Vergleich der Ausgabegeschwindigkeit unterschiedlicher Grafikkartenherstellern wurden diverse Benchmarks geschaffen, welche die Hersteller in den 1990ern dazu veranlasst haben, ihre Zeichenroutinen auf die Benchmarktests hin zu optimieren. Im Prinzip genau dasselbe Vorgehen wie 2013 beim Abgasskandal diverser Autohersteller.

7.4.2.2 3D-Beschleunigung

Die ersten 3D-Beschleuniger, die sich in der breiten Masse durchgesetzt haben, waren die Voodoo-Karten mit den Chips von 3Dfx. Diese Erweiterung wurde als zusätzliche 2. 3D-Grafikkarte parallel zur 1. 2D-Grafikkarte eingeführt.

Da es Anfang der 1990er-Jahre keinerlei Unterstützung von Betriebssystemherstellern für 3D-Funktionen gab, wurde die Karte fast nur unter MS-DOS betrieben und direkt programmiert. Dazu mussten die „normale“ Grafikkarte abgeschaltet und der Monitor auf der Kabelseite mit der 3D-Karte verbunden werden.

Diese Umschaltung wurde dann softwaregesteuert oft mit 2 einfachen Relais ausgeführt.

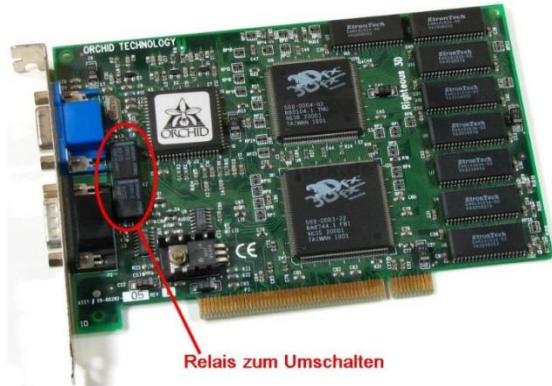


Abbildung 235 Voodoo-Karte Relais



Abbildung 236 Voodoo Kabelpeitsche

7.4.2.3 Vollwertige Kombi-2D/3D-Grafikkarten

Durch die langsame Ausbreitung von 32-Bit-Betriebssystemen Ende der 1990er wurden die 2D- und 3D-Funktionen jetzt endlich von Anfang an durch das Betriebssystem unterstützt.

Älterer professioneller Standard: **OpenGL**

Standard von Microsoft: **DirectX**

Einer der ersten Chipsätze auf der die 2D- und 3D-Funktionen beide enthalten waren, war der Banshee-Chipsatz vom Hersteller 3Dfx.

Später hat sich das Prinzip bei allen Herstellern durchgesetzt und die 3D-Funktionen wurden immer weiterentwickelt. Heutzutage sitzt auf der Grafikkarte meist ein sehr leistungsfähiger eigenständiger Rechner. Dieser ist jedoch sehr stark auf parallele Aufgaben optimiert und enthält oft mehrere 1000 ALUs (hier zu Shader-Einheiten zusammengefasst).

Die Firma 3Dfx wurde Anfang 2000 von Nvidia übernommen.

7.4.2.4 GPU in CPU integriert

Schon sehr früh wurde versucht, den für die Grafikausgabe notwendigen DRAM-Speicher einzusparen, indem man einen Teil des sowieso schon vorhandenen Hauptspeichers, dazu verwendet hat.

Zuerst wurde vom Hauptspeicher ein Teil abgezweigt, welcher als Bildspeicher verwendet wird. Das wurde bei manchen Rechnerentwürfen in den 1990ern noch mit einem externen Grafikkontroller realisiert. Die Pixeldaten werden hier zusätzlich zu den normalen Daten über den Systembus geleitet, was die Systeme damals stark ausgebremst hat.

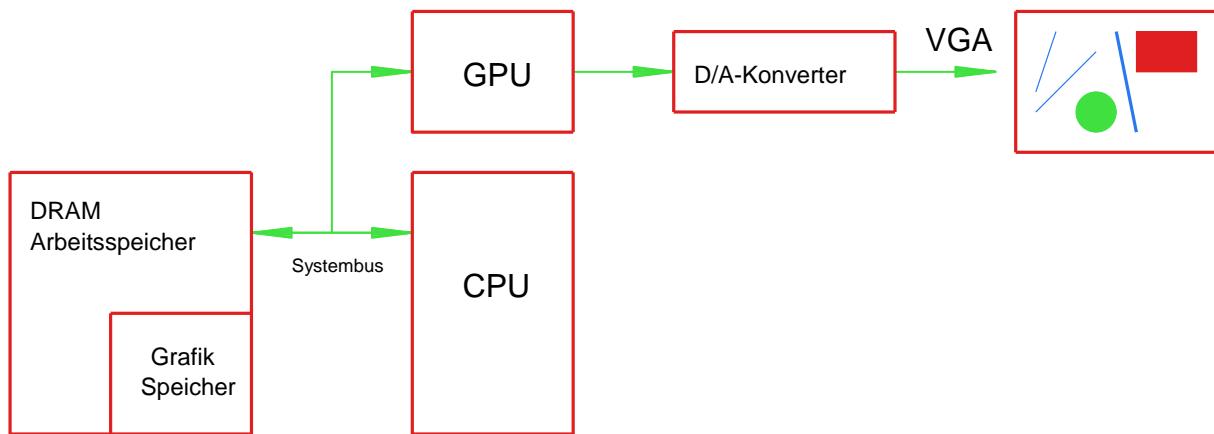


Abbildung 237 Pixel im Hauptspeicher

Da jedoch mittlerweile relative große Cache-Speicher in die CPUs mit integriert werden und das DRAM-Speicherinterface inzwischen mehrkanalig ausgelegt ist (siehe Kapitel 2.3.4.2), ist das bei ganz modernen Systemen nicht mehr der Fall.

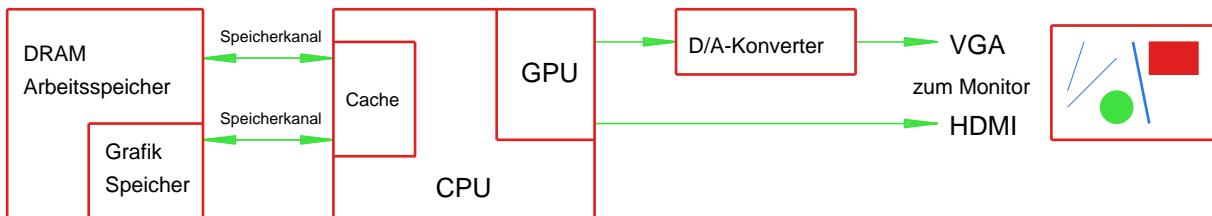


Abbildung 238 CPU GPU RAM

Als zweite Stufe wurden ab ca. 2010 alle benötigten Beschleunigerfunktionen in den Chip der Haupt-CPU mit integriert (Intel HD-Grafik, Nachfolger Intel-GMA, Intel **Graphics Media Accelerator**). Die Leistungsfähigkeit der integrierten 3D-Funktionen ist dabei nicht so stark wie bei externen Grafiksystemen mit eigenem Speicher.

8 Aktueller CPU-Markt

8.1 Hersteller von Desktop-CPUs

Bei den Desktop-CPUs sind von vielen Herstellern nur noch 2 übriggeblieben:

- Intel
 - Erfinder der x86-Prozessoren
 - Fertigung in Silicon Valley
- AMD
 - Anfangs Kopien der Intel-Chips, später eigenes Design
 - fertigt größtenteils in Deutschland (Dresden)

Für den Mobilen-Embedded-Smartphone-Markt, der in neuester Zeit alle anderen Computer stark verdrängt, sind einige Hersteller (Qualcomm, Samsung, seit 2021 auch Apple) auf dem Markt verbreitet. Die CPUs dieser Hersteller sind jedoch keine Eigenentwicklungen, sondern alles optimierte ARM-Cortex-Kerne. Auch diese werden stellen die Chips nicht selbst her, sondern die Herstellung wird von Auftragsfertigern übernommen (z.B. TSMC). Die Firma Apple hat mit dem M1/M2/M3 ab ca. 2020 einen eigenen ARM-Prozessor entwickelt. Dieser wird überwiegend in Taiwan (TSMC) hergestellt.

8.2 Hersteller ARM/Keil

ARM (Acorn Risc Machines, ehemals ACORN) ist kein Prozessorhersteller im herkömmlichen Sinn. Die Firma verkauft nur Lizenzen zur Herstellung von Prozessoren. Mittlerweile fertigt fast jeder Prozessorhersteller einen Prozessor mit ARM-Kern. Das hat leider dazu geführt, dass die meisten Prozessorhersteller beim Übergang auf 32-Bit-Prozessoren kein eigenes Design mehr entwickelt haben.

Der wichtigste Schritt von ARM war die Entwicklung der ARM-Cortex-Reihe (ab ca. 2004), die es für verschiedene Anwendungs-Optimierungen gibt, z.B. Cortex-A für rechenintensive Aufgaben, Cortex-M für Embedded-Aufgaben.

KEIL war bzw. ist ein deutscher Hersteller von verschiedenen Compilern für den Embedded-Bereich (Intel 8051, ARM, ...). Keil war längere Zeit der einzige Compiler-Hersteller für den ARM7 und ARM-Cortex Prozessor.

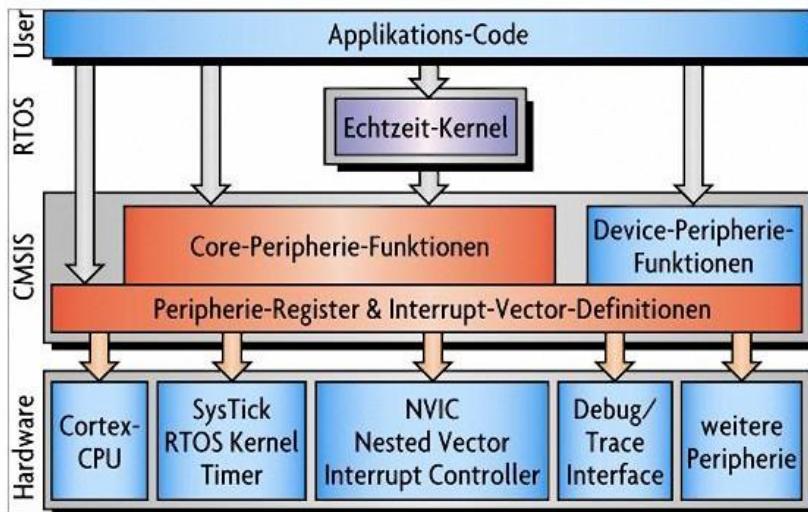
Keil wurde im Jahr 2005 von ARM übernommen, womit nun Hardware- und Compilerentwicklung im Prinzip von derselben Firma durchgeführt werden.

Mittlerweile gibt es einige Compilerhersteller für ARM-Prozessoren, einige basieren auf dem GNU-GCC-Compiler:

- ARM/KEIL-Development Studio 5/6
- Atmel-Studio
- Atollic True-Studio
- STM Cube IDE (entstanden durch Übernahme von True-Studio)
- Eclipse mit GNU Tools
- EM-Blocks
- IAR-Embedded Workbench
- Ride Rkit von Raisonance
- Segger Embeddes Studio
- Code Composer Studio von Texas Instruments

8.3 CMSIS

In der Vergangenheit haben viele Mikrocontroller-Hersteller durch eigene Architekturen unterschiedliche Ansteuerungen ihrer implementierten Spezialhardware definiert. ARM hat bei der Einführung der Cortex-Prozessoren den einheitlichen Standard CMSIS (Cortex Mikrocontroller Software Interface Standard) geschaffen, an den sich alle ARM-Cortex-CPU-Hersteller halten müssen.



CMSIS enthält Namensdefinitionen, Adressdefinitionen und Hilfsfunktionen für die Zugriffe auf die Register des Prozessorkerns und die Peripherie. Er wurde eine konsequente Zugriffsweise auf Peripherie, Interrupt-Vektoren eingeführt. Es wird auch noch eine Standard-Funktion (meist in Assembler) für den Systemstart zur Verfügung gestellt.

Viele CPU-Hersteller liefern zusätzlich zu CMSIS noch weitere Tools zur einfacheren Inbetriebnahme ihrer Hardware.

Die Übergabeschnittstellen sind meistens die von Softwareherstellern gelieferten IDE-Programme, welche die Softwareentwicklung stark vereinfachen.

Bei den meisten IDEs ist zusätzlich noch ein Debugging-Interface vorhanden, welches meistens schon durch die erweiterten Softwaretools vollständig parametrisiert wird.

Neuere Entwicklungen verlagern dem Softwareentwurf auch in den Internetbrowser.

8.4 SOCs (System On Chip)

8.4.1 Klassifizierung

System on Chip Prozessoren bilden ein komplettes Computersystem mit allen notwendigen Systemkomponenten auf einem Chip. Die ersten Bausteine dieser Art waren die Mikrocontroller. Man unterscheidet hier zwischen den folgenden 2 Grundtypen:

- Mikrocontroller (viel Funktionalität)
- Signalprozessoren (viel Rechenleistung)

Durch die ständig steigende Rechenleistung von Mikrocontrollern überlappen sich die beiden Gruppen je nach Hersteller immer mehr.

8.4.2 Hersteller von Signalprozessoren

Signalprozessoren bilden eine spezielle Familie von SOCs. Ihre Hardware ist spezialisiert auf hohe Rechenleistung und parallele Programmabarbeitung. Viele Hersteller haben hier schon sehr früh mit Multicore-Architekturen angefangen.

- Texas Instruments
- Analog Devices (stark optimiert auf Messsignalverarbeitung)
- Motorola (viele auf Audio-Signalverarbeitung spezialisierte Typen)

8.4.3 Hersteller von Embedded Mikrocontrollern

Hersteller von Mikrocontroller gibt es sehr viele am Markt. Hier nur eine Auswahl der am weitesten verbreiteten Typen.

- Analog Devices
- Cypress
- Microchip/Atmel
- PIC
- 8051
- AVR (Atmel)
- Freescale
- Renesas (ehemals Mitsubishi bzw. Hitachi)
- Infineon (ehemals Siemens)
- NXP Semiconductors (ehemals Philips Semiconductors)
- Silicon-Labs (Silabs)
- STM (Ehemals SGS-Thomson)
- Texas Instruments

8.4.4 Debugging & ISP

Bei System on Chip Prozessoren ist es normalerweise notwendig, auf das System das erforderliche Anwendungsprogramm zu übertragen und in dessen Flash-Speicher abzuspeichern. Dieser Vorgang wird meist „Flashing“ genannt.

Man unterscheidet dabei zwischen Programmierung/Flashing:

- bei der Entwicklung
- bei der Fertigung

Früher wurde das Programm mit einem speziellen Programmiergerät aufgespielt. Dazu musste die CPU in das Programmiergerät gesteckt werden und danach dann in die eigentliche Leiterplatte des Zielsystems. Diese bei der Entwicklung von Software sehr umständliche Methode wurde durch die Einführung des ISP (In System Programming) fast vollständig verdrängt.

Viele Prozessorhersteller integrieren direkt in die CPU eine Möglichkeit, den internen Programmspeicher der CPU direkt im Zielsystem zu programmieren. Dabei kommt bei fast allen Herstellern als letzte Fall-Back-Möglichkeit immer noch (Stand 2024) die RS232 Schnittstelle zum Einsatz. Zum Teil gibt es als 2. Option die Programmierung über die USB-Schnittstelle.

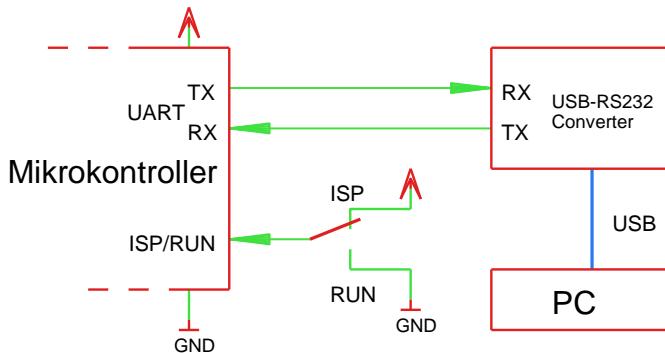


Abbildung 239 ISP Select

Die meisten Mikrocontroller haben einen speziellen Boot-Pin, über den man festlegen kann, welche Software nach einem Reset gestartet wird. Im Prozessor-Programmspeicher ist ein fest eingebautes Programm (Bootloader) vorhanden. Dieses wird automatisch gestartet, wenn der spezielle Boot-Pin vor dem Reset z.B. mit High-Pegel verbunden wird. Danach wartet der Bootloader auf die Kommunikation über z.B. RS232 auf den Programmdownload. Falls man vor dem nächsten Reset den Pin dann mit Low-Pegel verbindet, wird das Anwenderprogramm gestartet.

Die meisten modernen SOCs beinhalten zusätzlich zum ISP gleich eine Debugging-Schnittstelle, über die die Fehlersuche bei der Softwareentwicklung deutlich vereinfacht wird.

9 Beispielprojekt moderner Embedded Systementwurf

9.1 Projektidee

9.1.1 Beschreibung

Der Farocode auf den Gehäusen von elektrischen Widerständen ist schon seit langer Zeit nach DIN 41429 auf folgende Farbwerte festgelegt:

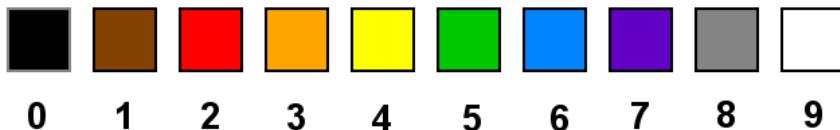


Abbildung 240 Widerstandsfarbcodes

Die Farbcodes erlauben eine eindeutige Zuordnung der Ziffern 0-9 zu ihrer entsprechenden Farbe. Damit ist es möglich, eine Dezimalziffer nur durch die Wiedergabe ihrer Farbe anzuzeigen. Wenn man für jede Digitalstelle der Anzeige eine einzelne RGB-LED verwendet, kann man damit eine mehrstellige Anzeige mit relativ wenig LEDs aufbauen. Mit einer sechsstelligen RGB-LED Anzeige ist bereits die Anzeige der Uhrzeit mit Stunde, Minute und Sekunde möglich.

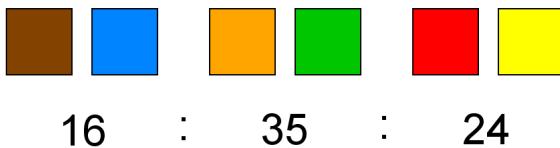


Abbildung 241 Farbcodeuhrzeit

Das hier vorgestellte Beispielprojekt stellt eine Digitaluhr dar, welche die Uhrzeit anhand von Widerstandsfarbcodes anzeigt. Die Uhr soll autark auf einer speziell dafür entworfenen Hardware realisiert werden.

Folgende Entwurfsschritte sind für das Projekt vorgesehen:

- Optische Software-Simulation auf einem PC
- Auswahl der Bauteile (Mikrocontroller, LEDs)
- Realisierung der Schaltung
- Entflechtung des Leiterplattenlayouts
- Konfiguration des Mikrocontrollers mithilfe des Herstellertools
- Automatische Erzeugung einer HAL-Umgebung mithilfe des Herstellertools
- Entwurf der Software für den Mikrocontroller

9.1.2 Simulation

Für die Simulation des Projekts wurde das Programm Paket Delphi des Herstellers Embarcadero mit der Programmiersprache Pascal gewählt. Bei diesem Softwarepaket ist durch die mitgelieferte Firemonkey-Bibliothek mit der Erstellung nur eines Quellcodes die Nutzung auf folgenden Plattformen möglich:

- 32-Bit Windows
- 64-Bit Windows
- iOS-32Bit
- iOS-64Bit
- OSX
- Android

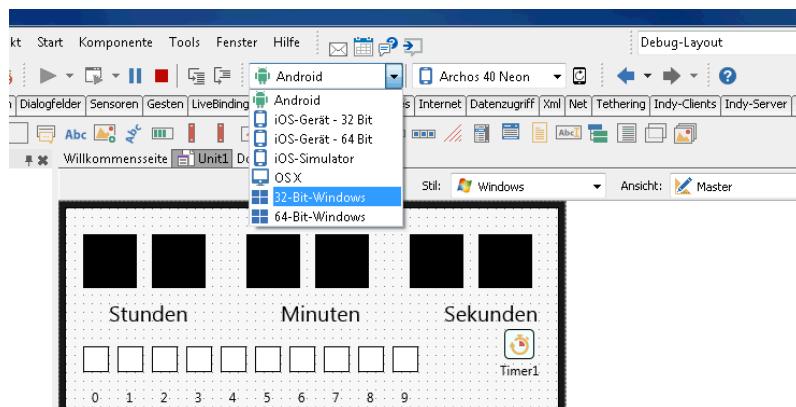


Abbildung 242 Delphi IDE

Die gesamte Aufgabe wird in einer Timerprocedure erledigt, welche jede Sekunde einmal vom System aufgerufen wird.

```
procedure TUah.Timer1Timer(Sender: TObject);

var
  AFDateTime : TDateTime;
  hund, katz, maus, fizl, jahr, monat, tag, tage : word;

begin
  AFdatetime:=time;
  decodetime(AFdatetime, hund, katz, maus, fizl);

  RectangleHZ.Fill.Color:=num2col[hund div 10];
  RectangleHE.Fill.Color:=num2col[hund-(hund div 10)*10];

  RectangleMZ.Fill.Color:=num2col[katz div 10];
  RectangleME.Fill.Color:=num2col[katz-(katz div 10)*10];

  RectanglesZ.Fill.Color:=num2col[maus div 10];
  RectangleSE.Fill.Color:=num2col[maus-(maus div 10)*10];

  Label1.Text:=inttostr(hund);
  Label2.Text:=inttostr(katz);
  Label3.Text:=inttostr(maus);

end;
```

Abbildung 243 Quellcode der Timerprocedure

9.2 Hardware

9.2.1 Schaltung

Um eine 6-stellige RGB-Anzeige zu ermöglichen, fiel die Wahl bei den LEDs auf fertige SMD-RGB-LEDs, bei der sich drei LED-Chips in einem Gehäuse befinden. Die Anode der 3 LEDs ist schon intern miteinander verbunden.

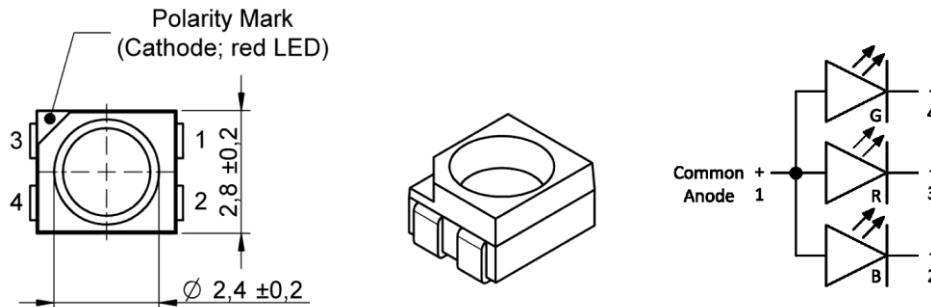


Abbildung 244 Datenblattauszug RGB-LED

Von den LEDs werden 6 Stück zu einer 3x6 Matrix zusammengeschaltet. Für jede LED-Farbe ist ein Vorwiderstand vorgesehen. Angesteuert werden die LEDs von einem ARM-Cortex M0+ Prozessor mit einem 32-poligen LQFP-Gehäuse, Pinabstand 0,8 mm. Zusätzlich ist ein Anschluss für die serielle Schnittstelle vorgesehen und ein 32-kHz-Quarz als Referenz für die Uhr. Die Ansteuerung der blauen LED-Chips reicht mit einer 3,3 V Versorgung gerade aus.

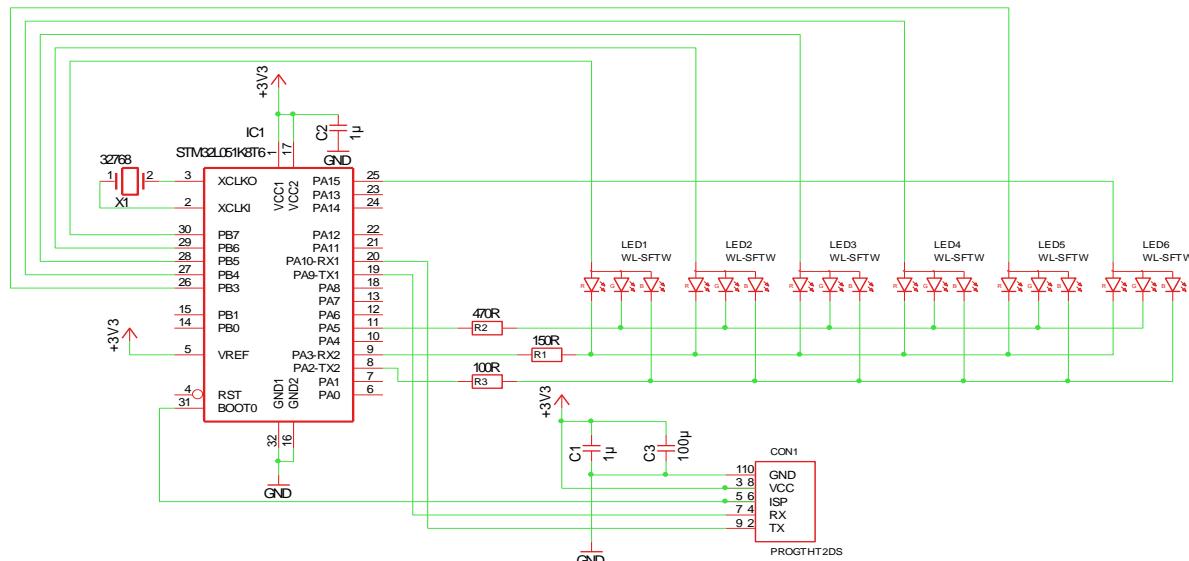


Abbildung 245 Stromlaufplan RGB-Uhr

9.2.2 Leiterplatte

Das Leiterplattenlayout für die Uhr wurde von Hand erstellt. Nach einer umfangreichen Umplatzierung der Bauteile wurde die Entflechtung auf einer Leiterplatte mit einer einseitigen Kupferauflage erreicht.

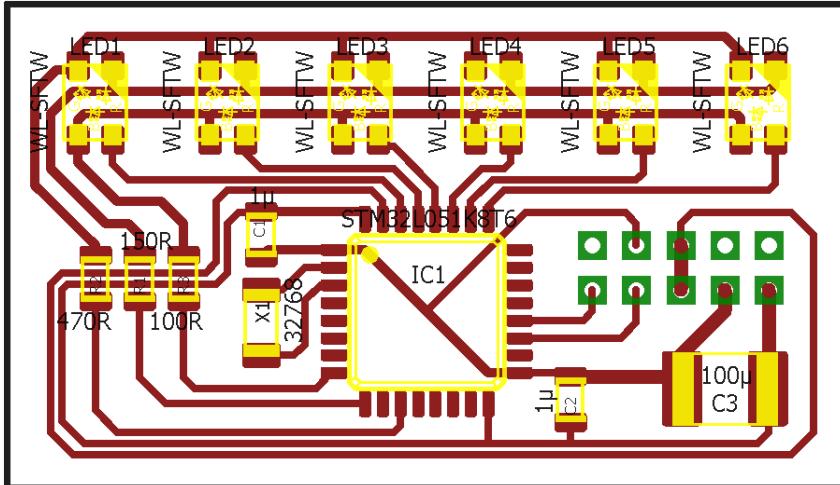


Abbildung 246 Leiterplatte Hand

Zum Vergleich hier ein Output des zur Verfügung stehenden Autorouters:
Der Autorouter ist im besten Fall bei 97 % (eine Verbindung fehlt) stehen geblieben.

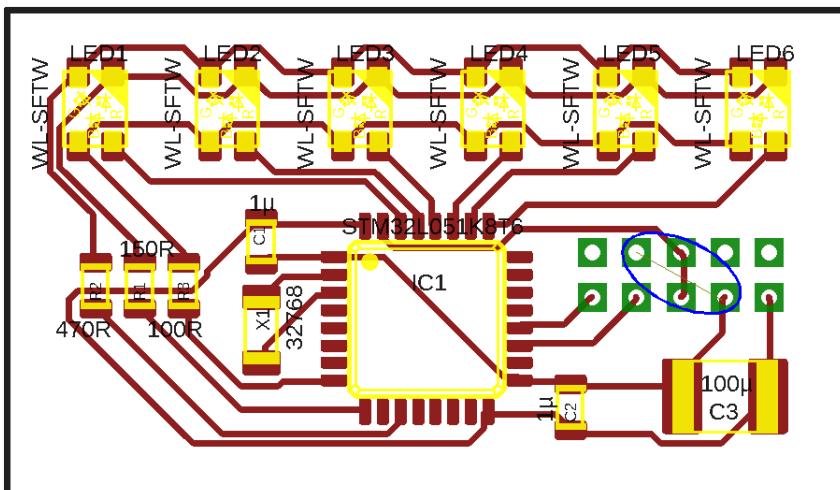


Abbildung 247 Layout Autorouter

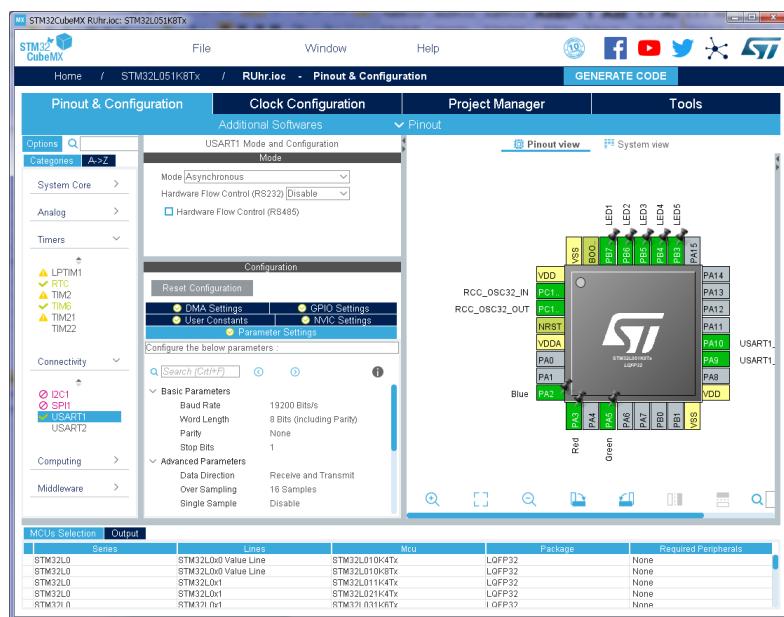
Der Autorouter wurde hier nur zu Test gestartet, nachdem das Layout von Hand zu 100% entflochten wurde. Es gab bereits eine 100 %-Lösung, die der Autorouter auch unter hier optimalen Bedingungen nicht gefunden hat!

9.3 Software

9.3.1 Beispiel der Firma STM mit STM32CubeMX

Die europäische Firma STM (SGS Thomson Microelectronics) stellt zum einfacheren Systementwurf das Softwaretool STM32-Cube-MX zur Verfügung. Mittels grafischer Eingabe können sämtliche Hard- und Softwareparameter eines ARM-Cortex-Mikrocontrollers festgelegt werden.

Das Softwaretool erzeugt automatisch einen HAL (Hardware Abstraction Layer), mit dessen Hilfe die Hardware der jeweiligen CPU mittels einfacher Softwareroutinen angesteuert werden kann. Der Output des Softwaretools erzeugt direkt ein komplettes Projekt mit allen Dateien für einige gängige Software IDEs in der Programmiersprache C.



9.3.2 Blick auf die automatisch erzeugte Software

9.3.2.1 Softwarestruktur

Die MX-Cube-Software kann ein komplettes Projekt für Entwicklungsumgebungen von verschiedenen Herstellern erzeugen. Als Beispiel soll hier die kommerzielle, sehr verbreitete Software der Firma ARM/Keil dienen, die als Lite/Demoversion auf eine Codegröße von 32 kByte beschränkt ist.

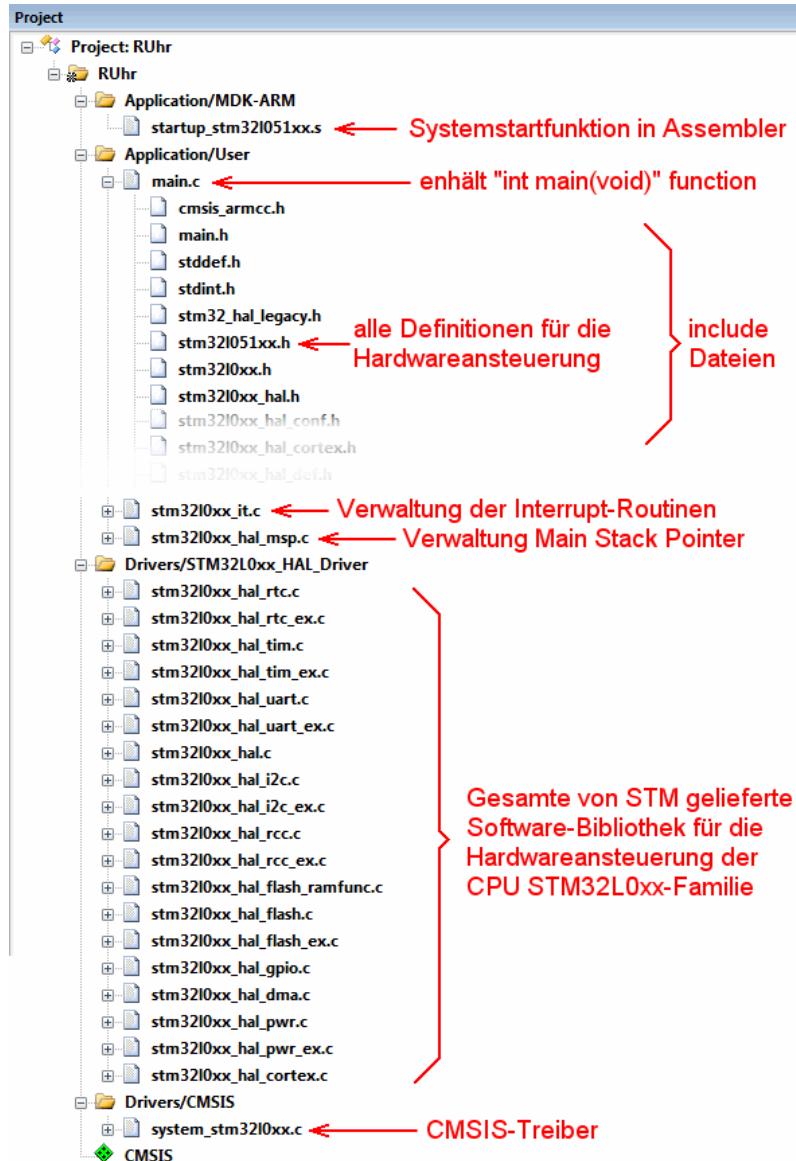


Abbildung 250 Softwarestruktur Beispiel Keil

9.3.2.2 Hardware-Ansteuerungs-Header-Datei

Die Ansteuerung der gesamten spezifischen Peripherie eines Mikrocontrollers ist fast nur über eine einzige Header-Datei gelöst. Diese Datei enthält im Prinzip keinen Programmcode, sondern nur **#defines** und **typedef struct** Instruktionen für den C-Präprozessor.

Die Datei ist immer relativ groß (0.5-1 MByte-ASCII-Code) und die darin enthaltenen Funktionen werden direkt vom Präprozessor in den aufrufenden Quellcode integriert. Dadurch geht bei einem Runtime-Funktionsaufruf keine Zeit verloren. Der Präprozessor ersetzt die Funktionsaufrufe direkt mit dem Zielcode. Diese Vorgehensweise führt zu sehr wenig Overhead und verschwendet keine Prozessorleistung für Funktionsaufrufe zur Hardwarekonfiguration.

```
typedef struct
{
    __IO uint32_t ISR;           /*!< ADC Interrupt and Status register,      Address offset:0x00 */
    __IO uint32_t IER;           /*!< ADC Interrupt Enable register,        Address offset:0x04 */
    __IO uint32_t CR;            /*!< ADC Control register,                  Address offset:0x08 */
    __IO uint32_t CFGR1;         /*!< ADC Configuration register 1,        Address offset:0x0C */
    __IO uint32_t CFGR2;         /*!< ADC Configuration register 2,        Address offset:0x10 */
    __IO uint32_t SMPR;          /*!< ADC Sampling time register,          Address offset:0x14 */
    uint32_t   RESERVED1;        /*!< Reserved,                           0x18 */
    uint32_t   RESERVED2;        /*!< Reserved,                           0x1C */
    __IO uint32_t TR;            /*!< ADC watchdog threshold register,     Address offset:0x20 */
    uint32_t   RESERVED3;        /*!< Reserved,                           0x24 */
    __IO uint32_t CHSELR;        /*!< ADC channel selection register,     Address offset:0x28 */
    uint32_t   RESERVED4[5];     /*!< Reserved,                           0x2C */
    __IO uint32_t DR;            /*!< ADC data register,                   Address offset:0x40 */
    uint32_t   RESERVED5[28];    /*!< Reserved,                           0x44 - 0xB0 */
    __IO uint32_t CALFACT;       /*!< ADC data register,                   Address offset:0xB4 */
} ADC_TypeDef;
```

Abbildung 251 Auszug 1 Hardware-Header-Datei

```
***** Bits definition for ADC_CR register *****
#define ADC_CR_ADCAL_Pos          (31U)
#define ADC_CR_ADCAL_Msk         (0x1UL << ADC_CR_ADCAL_Pos)  /*!< 0x80000000 */
#define ADC_CR_ADCAL_Msk        ADC_CR_ADCAL_Msk  /*!< ADC calibration */
#define ADC_CR_ADVREGEN_Pos       (28U)
#define ADC_CR_ADVREGEN_Msk      (0x1UL << ADC_CR_ADVREGEN_Pos) /*!< 0x10000000 */
#define ADC_CR_ADVREGEN_Msk     ADC_CR_ADVREGEN_Msk /*!< ADC Voltage Regulator Enable */
#define ADC_CR_ADSTP_Pos          (2U)
#define ADC_CR_ADSTP_Msk         (0x1UL << ADC_CR_ADSTP_Pos)  /*!< 0x00000010 */
#define ADC_CR_ADSTP_Msk        ADC_CR_ADSTP_Msk  /*!< ADC stop of conversion command */
#define ADC_CR_ADSTART_Pos        (2U)
#define ADC_CR_ADSTART_Msk       (0x1UL << ADC_CR_ADSTART_Pos) /*!< 0x00000004 */
#define ADC_CR_ADSTART_Msk      ADC_CR_ADSTART_Msk /*!< ADC start of conversion */
#define ADC_CR_ADDIS_Pos          (1U)
#define ADC_CR_ADDIS_Msk         (0x1UL << ADC_CR_ADDIS_Pos)  /*!< 0x00000002 */
#define ADC_CR_ADDIS_Msk        ADC_CR_ADDIS_Msk  /*!< ADC disable command */
#define ADC_CR_ADEN_Pos          (0U)
#define ADC_CR_ADEN_Msk         (0x1UL << ADC_CR_ADEN_Pos)  /*!< 0x00000001 */
#define ADC_CR_ADEN_Msk        ADC_CR_ADEN_Msk  /*!< ADC enable control */ /*#### TBV */
```

Abbildung 252 Auszug 2 Hardware-Header-Datei

Diese Auszüge aus der Hardware-Header-Datei zeigen einen kleinen Teil der Hardware-Ansteuerung des in den Mikrocontroller integrierten A/D-Wandlers.

9.3.2.3 *int main(void) Routine*

Die automatisch erzeugte `int main(void)`-Funktion befindet sich in der Datei `main.c`. Diese Datei enthält alle Routinen, um die gesamte für dieses Projekt mittels CubeMX definierte Hardware zu initialisieren. Sie endet in einer `while(1) { }` Endlosschleife. An dieser Stelle beginnt dann das eigentliche Anwenderprogramm.

Für den Hardwarezugriff auf manche Komponenten wird vom CubeMX eine Handle-Variable kreiert. In diesem Beispiel sind das die folgenden 3 Hardwarekomponenten:

```
RTC_HandleTypeDef hrtc;
TIM_HandleTypeDef htim6;
UART_HandleTypeDef huart1;
```

Abbildung 253 Software-Handles

Die Software erstellt auch automatisch die Funktionen, um die entsprechende Hardware zu initialisieren.

```
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_RTC_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_TIM6_Init(void);
```

Abbildung 254 Hardware-Init-Funktionen

In der Routine für die GPIO-Initialisierung werden alle Funktionsaufrufe getätig, anhand der vom User im CubeMX vorher festgelegten Ein- und/oder Ausgangspins.

```
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOA, Blue_Pin|Red_Pin|Green_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, LED5_Pin|LED4_Pin|LED3_Pin|LED2_Pin|LED1_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pins : Blue_Pin Red_Pin Green_Pin */
    GPIO_InitStruct.Pin = Blue_Pin|Red_Pin|Green_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /*Configure GPIO pins : LED5_Pin LED4_Pin LED3_Pin LED2_Pin LED1_Pin */
    GPIO_InitStruct.Pin = LED5_Pin|LED4_Pin|LED3_Pin|LED2_Pin|LED1_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

Abbildung 255 Funktion zur GPIO-Initialisierung

Die automatisch erzeugten Routinen sind alle sehr ausführlich dokumentiert.

9.3.2.4 Interrupt Steuerung

Alle Interrupt-Service-Routinen und ihre Verwaltung sind der Datei `stm3210xx_it.c` hinterlegt. Im Beispiel-Projekt wurden 2 Interrupts definiert. Der Timerinterrupt TIM6 und der Interrupt für die 1. serielle Schnittstelle USART1.

Die beiden entsprechenden Funktionen sind deshalb hier zu finden:

```
void TIM6_DAC_IRQHandler(void)
{
    /* USER CODE BEGIN TIM6_DAC_IRQn 0 */

    /* USER CODE END TIM6_DAC_IRQn 0 */
    HAL_TIM_IRQHandler(&htim6);
    /* USER CODE BEGIN TIM6_DAC_IRQn 1 */

    /* USER CODE END TIM6_DAC_IRQn 1 */
}

void USART1_IRQHandler(void)
{
    /* USER CODE BEGIN USART1_IRQn 0 */

    /* USER CODE END USART1_IRQn 0 */
    HAL_UART_IRQHandler(&huart1);
    /* USER CODE BEGIN USART1_IRQn 1 */

    /* USER CODE END USART1_IRQn 1 */
}
```

Abbildung 256 Auszug Interrupt-Datei

Der TIM6 teilt sich bei diesem Controllertyp die Interrupt-Service-Routine mit dem D/A-Wandler, der jedoch bei der hier verwendeten CPU-Ausführung gar nicht vorhanden ist.

Die Routinen werden automatisch bei einem Interrupt-Ereignis angesprungen und rufen automatisch ihre zugehörige HAL-Interrupt-Handler-Funktion auf, welche alle Hardwareflags wieder zurücksetzt, damit der Interrupt erneut wieder aufgerufen werden kann.

Der User kann hier vor und/oder nach dem HAL-Interrupt-Handler-Funktionsaufruf seinen eigenen Code einfügen.

9.3.3 Anwenderprogramm

9.3.3.1 Kommunikation mit der Softwareumgebung

Die `main()` Routine in der Datei `main.c` endet in einer `while(1) { }` Endlosschleife. Man könnte jetzt das gesamte Anwenderprogramm in diese while-Schleife einfügen. Um die Programmierung der Anwendung so modular wie möglich zu gestalten, sollte man jedoch sämtliche Anwender-Funktionen in ein externes Modul ausgliedern (hier die Datei `Ruhr.c`). Die Verbindung der `main.c` zum externen Modul erfolgt hier mit nur einem einzigen Funktionsaufruf.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    RClock(); // externe Funktionsaufruf

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Abbildung 257 Externer main()-Funktionsaufruf

Das gleiche Prinzip kann man auch beim Aufruf der Interrupt-Service-Routinen aus der Datei `stm3210xx_it.c` anwenden.

```
void TIM6_DAC_IRQHandler(void)
{
    /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
    TimerIRQ6AF(); // externe Funktionsaufruf
    /* USER CODE END TIM6_DAC_IRQn 0 */
    HAL_TIM_IRQHandler(&htim6);
    /* USER CODE BEGIN TIM6_DAC_IRQn 1 */

    /* USER CODE END TIM6_DAC_IRQn 1 */
}

void USART1_IRQHandler(void)
{
    /* USER CODE BEGIN USART1_IRQn 0 */
    USART1IRQAF(); // externe Funktionsaufruf
    /* USER CODE END USART1_IRQn 0 */
    HAL_UART_IRQHandler(&huart1);
    /* USER CODE BEGIN USART1_IRQn 1 */

    /* USER CODE END USART1_IRQn 1 */
}
```

Abbildung 258 Externe ISR-Aufrufe

Die Übergabe muss dann mit der Headerdatei `Ruhr.h` erfolgen, die in der `main.c` per `#include "Ruhr.h"` eingebunden werden muss.

Damit die Kommunikation des externen Moduls `Ruhr.c` mit den restlichen Dateien korrekt abläuft, ist in dieser Datei ein `#include` der Header-Datei der HAL-Bibliothek `stm3210xx_hal.h` notwendig.

Die Handle-Variablen, mit denen man auf die vordefinierte Hardwarestruktur zugreifen kann, müssen als extern deklariert werden, da sie bereits in der `main.c` definiert sind.

```
#include "stdio.h"
#include "Ruhr.h"
#include "stm3210xx_hal.h"

extern RTC_HandleTypeDef hrtc;
extern TIM_HandleTypeDef htim6;
extern UART_HandleTypeDef huart1;
```

Abbildung 259 Kopfteil Anwenderprogramm

9.3.3.2 Hauptroutine

Alle Anzeigeelemente werden in globalen Variablen gespeichert. Ebenso alle Uhrzeitelemente. Zusätzlich ist noch eine Definition der 10 Farben als Konstantenarray notwendig.

```
RTC_TimeTypeDef gTime; // Variablenrecord, in dem die Uhrzeit gespeichert ist

char LEDs[6][3], // aktuelle Farbe jeder einzelne RGB-LED mit Helligkeit in 100-x%
      mplex, // Multiplexer-Variable
      PWMcount; // Pulsweitencounter

const char num2col[10][3]= // Farbkonstanten, Helligkeit in 100-x%
{
    // R   G   B
    {100, 100, 100, // schwarz
     85, 98, 0, // braun
     50, 100, 100, // rot
     50, 75, 100, // orange
     75, 50, 100, // gelb
     100, 50, 100, // grün
     100, 75, 0, // blau
     75, 100, 0, // violett
     85, 85, 85, // grau
     50, 50, 50 // weiß
};
```

Abbildung 260 Globale Variablen

Die Hauptroutine holt die aktuelle Uhrzeit von der RTC (Real Time Clock) ab, und speichert die Zahlen konvertiert über die Farbkonstanten in den Variablen für die LEDs ab.

```
void RClock(void)
{
    HAL_RTC_GetTime(&hrtc, &gTime, RTC_FORMAT_BIN);

    LEDs[0][0]=num2col[gTime.Hours / 10][0]; // Stunden Zehner rot
    LEDs[0][1]=num2col[gTime.Hours / 10][1]; // Stunden Zehner grün
    LEDs[0][2]=num2col[gTime.Hours / 10][2]; // Stunden Zehner blau

    LEDs[1][0]=num2col[gTime.Hours-((gTime.Hours / 10)*10)][0]; // Stunden Einer rot
    LEDs[1][1]=num2col[gTime.Hours-((gTime.Hours / 10)*10)][1]; // Stunden Einer grün
    LEDs[1][2]=num2col[gTime.Hours-((gTime.Hours / 10)*10)][2]; // Stunden Einer blau

    LEDs[2][0]=num2col[gTime.Minutes / 10][0]; // Minuten Zehner rot
    LEDs[2][1]=num2col[gTime.Minutes / 10][1]; // Minuten Zehner grün
    LEDs[2][2]=num2col[gTime.Minutes / 10][2]; // Minuten Zehner blau

    LEDs[4][0]=num2col[gTime.Seconds / 10][0]; // Sekunden Zehner rot
    LEDs[4][1]=num2col[gTime.Seconds / 10][1]; // Sekunden Zehner grün
    LEDs[4][2]=num2col[gTime.Seconds / 10][2]; // Sekunden Zehner blau
    LEDs[3][0]=num2col[gTime.Minutes-((gTime.Minutes / 10)*10)][0]; // Minuten Einer rot
    LEDs[3][1]=num2col[gTime.Minutes-((gTime.Minutes / 10)*10)][1]; // Minuten Einer grün
    LEDs[3][2]=num2col[gTime.Minutes-((gTime.Minutes / 10)*10)][2]; // Minuten Einer blau

    LEDs[5][0]=num2col[gTime.Seconds-((gTime.Seconds / 10)*10)][0]; // Sekunden Einer rot
    LEDs[5][1]=num2col[gTime.Seconds-((gTime.Seconds / 10)*10)][1]; // Sekunden Einer grün
    LEDs[5][2]=num2col[gTime.Seconds-((gTime.Seconds / 10)*10)][2]; // Sekunden Einer blau
}
```

Abbildung 261 Hauptroutine

Die Inhalte der globalen Variablen werden dann vom Timerinterrupt übernommen und zur Anzeige auf die LEDs gebracht.

9.3.3.3 Timerinterrupt

Der Timerinterrupt wird mit einer Frequenz von 160kHz aufgerufen und schaltet die 6 LED-Gruppen mit je 3 LED nacheinander ein (Variable `mplex`). Bei jeder LED kann die Helligkeit über die Pulsweite eingestellt werden. Dazu wird die Variable `PWMcount` bei jedem Timerdurchlauf immer um 1 bis auf 100 erhöht. Bei 0 werden alle LEDs ausgeschaltet, und an der entsprechenden Stelle wird die zugehörige LED dann eingeschaltet.

```
void TimerIRQ6AF(void)
{
    PWMcount++;
    if (PWMcount>100) { PWMcount=0; mplex++;
                        if (mplex>5) mplex=0; }

    if (PWMcount==0) LEDsOff();

    SetAnode(mplex);
    SetKathode(mplex, PWMcount);
}
```

Abbildung 262 Timerinterrupt

Zur Steuerung der LED-Matrix werden Kathode und Anode über 2 getrennte Funktionen angesteuert, welche vom Timerinterrupt aufgerufen werden.

```
void SetAnode(char x)
{
    if (x==0) HAL_GPIO_WritePin(GPIOB, LED1_Pin, GPIO_PIN_SET);
    if (x==1) HAL_GPIO_WritePin(GPIOB, LED2_Pin, GPIO_PIN_SET);
    if (x==2) HAL_GPIO_WritePin(GPIOB, LED3_Pin, GPIO_PIN_SET);
    if (x==3) HAL_GPIO_WritePin(GPIOB, LED4_Pin, GPIO_PIN_SET);
    if (x==4) HAL_GPIO_WritePin(GPIOB, LED5_Pin, GPIO_PIN_SET);
    if (x==5) HAL_GPIO_WritePin(GPIOA, LED6_Pin, GPIO_PIN_SET);
}

void SetKathode(char x, char y)
{
    if (y>LEDs[x][0]) HAL_GPIO_WritePin(GPIOA, Red_Pin, GPIO_PIN_RESET);
    if (y>LEDs[x][1]) HAL_GPIO_WritePin(GPIOA, Green_Pin, GPIO_PIN_RESET);
    if (y>LEDs[x][2]) HAL_GPIO_WritePin(GPIOA, Blue_Pin, GPIO_PIN_RESET);
}
```

Abbildung 263 Timerinterrupt Unterfunktionen

Bei PWMcount=0 werden immer alle LEDs mit der Funktion LEDsOff über den Timerinterrupt komplett ausgeschaltet.

```
void LEDsOff()
{
    HAL_GPIO_WritePin(GPIOA, Red_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, Green_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, Blue_Pin, GPIO_PIN_SET);

    HAL_GPIO_WritePin(GPIOB, LED1_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, LED2_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, LED3_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, LED4_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, LED5_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, LED6_Pin, GPIO_PIN_RESET);
}
```

Abbildung 264 Alle LEDs aus

10 Abbildungsverzeichnis

Abbildung 1 Effektivwertgleichrichter als Analogrechner	6
Abbildung 2 Suanpan	6
Abbildung 3 Neperianische Rechenstäbchen	7
Abbildung 4 Rechenstäbchen mit Abakus	7
Abbildung 5 Rechenschieber.....	8
Abbildung 6 MyCPU.....	9
Abbildung 7 Monster 6502.....	9
Abbildung 8 Rechenmaschine von Professor Wilhelm Schickard.....	10
Abbildung 9 Rechenmaschine für sechsstellige Addition und Subtraktion Blaise Pascal	10
Abbildung 10 Rechenmaschine von Leibniz.....	10
Abbildung 11 Rechenmaschine von Gottfried Wilhelm Hahn.....	11
Abbildung 12 Arithmometer	11
Abbildung 13 Curta von Curt Herzstark	11
Abbildung 14 Webstuhl von Jacquard	12
Abbildung 15 Analytical Engine	12
Abbildung 16 Maschine von Hermann Hollerith	13
Abbildung 17 Z1 von Konrad Zuse.....	13
Abbildung 18 Turing Maschine von Alan Turing.....	13
Abbildung 19 Z3 von Konrad Zuse.....	14
Abbildung 20 ENIAC.....	14
Abbildung 21 EDSAC.....	14
Abbildung 22 Erstes IBM-PC Motherboard.....	15
Abbildung 23 Erster Software Bug	16
Abbildung 24 52+1-Bit Adresse.....	18
Abbildung 25 Systemaufbau	19
Abbildung 26 80386-CPU + 80387-FPU + 82385-Cachecontroller	20
Abbildung 27 Cache	20
Abbildung 28 CPU intern.....	21
Abbildung 29 CPU 32/64 Bit Multiplexing	22
Abbildung 30 CPU mit 2 Speicherkanälen	22
Abbildung 31 4 Bytes in einem 32-Bit-Wort	23
Abbildung 32 Auszug i386DX Datenblatt	23
Abbildung 33 Quarzoszillator	24
Abbildung 34 FSB SDR DDR QDR	24
Abbildung 35 Intel CPU 286 & FPU 287	25
Abbildung 36 Lochkarte Abbildung 37 EG-Führerschein	26
Abbildung 38 LTO-Band	26
Abbildung 39 Diskettengrößen.....	27
Abbildung 40 5,25 Zoll und 2,5 Zoll Diskette	27
Abbildung 41 Aufbau Standard-Festplatte	28
Abbildung 42 Festplattenentwicklung.....	29
Abbildung 43 Longitudinale Magnetisierung	29
Abbildung 44 Perpendikuläre Magnetisierung.....	29
Abbildung 45 Herkömmliche Aufzeichnung	30
Abbildung 46 Shingled Magnetic Recording	30
Abbildung 47 Ringkernspeicher.....	32
Abbildung 48 Preisentwicklung SSD	33
Abbildung 49 Aktuelle Daten (2018) von Flash-Speicherzellen.....	33
Abbildung 50 Hybrid-Festplatte.....	34
Abbildung 51 CD-ROM-Aufbau.....	35
Abbildung 52 DVD Aufbau	35

Abbildung 53 BluRay Aufbau.....	36
Abbildung 54 Vergleich CD DVD Blu-Ray	36
Abbildung 55 Software & Hardware RAID	37
Abbildung 56 RAID 1	38
Abbildung 57 RAID 5	38
Abbildung 58 ATX-Power	39
Abbildung 59 Crowbar	39
Abbildung 60 Reset	40
Abbildung 61 Watchdog	40
Abbildung 62 Von-Neumann	41
Abbildung 63 Full Harvard	42
Abbildung 64 XT-AT-ISA-BUS	43
Abbildung 65 PCI-Bus	43
Abbildung 66 Mäander	44
Abbildung 67 PCI-Express-Bus	44
Abbildung 68 PCI-Express PCI-Parallel Adapter	45
Abbildung 69 Darstellung von Dualzahlen in Registern	46
Abbildung 70 Gleitkommazahlen.....	47
Abbildung 71 ALU.....	48
Abbildung 72 Exklusiv ODER Abbildung 73 UND	49
Abbildung 74 Halbaddierer	49
Abbildung 75 Halbaddierer mit Standardelementen	50
Abbildung 76 Halbaddierer Schaltsymbole.....	50
Abbildung 77 Volladdierer aus Halbaddierern.....	51
Abbildung 78 Volladdierer Schaltsymbole	51
Abbildung 79 Paralleladdierwerk	52
Abbildung 80 Paralleladdierer mit Volladdierern	52
Abbildung 81 Übertragsbetrachtung	53
Abbildung 82 4-Bit Carry Look Ahead Addierer.....	53
Abbildung 83 Inkrementator	54
Abbildung 84 Parallele Subtraktion	55
Abbildung 85 Shift	56
Abbildung 86 Shift mit Flip-Flops	57
Abbildung 87 4-Bit-Barrel-Shifter	57
Abbildung 88 Multiplikation Schulmethode	58
Abbildung 89 Multiplikationsalgorithmus	58
Abbildung 90 4-Bit-Hardware-Multiplikation mit Addierwerken	59
Abbildung 91 32-Bit-Hardwaremultiplikation mit Addierwerken	59
Abbildung 92 Hardware-Multiplikation mit Look-Up-Table	60
Abbildung 93 Division Schulmethode	60
Abbildung 94 Shift um 2	61
Abbildung 95 Faktor 256	61
Abbildung 96 16-Bit MAC mit 48-Bit ACC	62
Abbildung 97 SIMD Beispiele bei 32-Bit-Registern	63
Abbildung 98 Opcode	64
Abbildung 99 OPcode mit 4 Adressen	66
Abbildung 100 OPcode mit 2 Adressen	66
Abbildung 101 OPcode mit einer Adresse	66
Abbildung 102 CPU Nonskalar zeitlicher Verlauf.....	68
Abbildung 103 CPU Skalar zeitlicher Verlauf.....	68
Abbildung 104 CPU Superskalar logischer Aufbau.....	69
Abbildung 105 Out Of Order Execution logischer Aufbau.....	69
Abbildung 106 FPU Register	71

Abbildung 107 FPU Konstanten	71
Abbildung 108 Überblick der CPU-Entwicklung von Intel:	73
Abbildung 109 Rechenzeiten Integer-Multiplikation.....	74
Abbildung 110 80486 Chipanalyse	74
Abbildung 111 Bustiming Lesezugriff	75
Abbildung 112 Bustiming Schreibzugriff	75
Abbildung 113 Adressdekodierung.....	76
Abbildung 114 Memory-Map	77
Abbildung 115 Interrupt-Controller	78
Abbildung 116 Beispiel ISR-Vektor-Tabelle.....	78
Abbildung 117 8086 Pinout	79
Abbildung 118 PCI-BUS-Belegung	79
Abbildung 119 Burst	80
Abbildung 120 DMA-Zugriff.....	81
Abbildung 121 CPU mit Memory Management Unit	82
Abbildung 122 Speicherarten	83
Abbildung 123 SRAM-MOSFET-Zelle	84
Abbildung 124 SRAM-Flip-Flop-Array.....	84
Abbildung 125 DRAM-MOSFET-Zelle	85
Abbildung 126 DRAM-Array	85
Abbildung 127 RAS-CAS-Adressierung	86
Abbildung 128 Buffered/Registered RAM	86
Abbildung 129 Fast Page Mode RAM	87
Abbildung 130 EDO-RAM	87
Abbildung 131 SDRAM.....	88
Abbildung 132 Floating Gate MOSFET	90
Abbildung 133 Floating Gate Schreiben	90
Abbildung 134 Floating Gate Lesen 0.....	91
Abbildung 135 Floating Gate Lesen 1.....	91
Abbildung 136 EPROM mit Fenster	92
Abbildung 137 Serielles 64kBit -EEPROM im SO8 Gehäuse.....	93
Abbildung 138 SPD-EEPROM	93
Abbildung 139 NOT NAND NOR.....	94
Abbildung 140 NOR-Flash	94
Abbildung 141 NAND-Flash	95
Abbildung 142 Elektronen pro Bit	95
Abbildung 143 SLC-Zelle	96
Abbildung 144 MLC-Zelle.....	96
Abbildung 145 TLC-Zelle	96
Abbildung 146 Charge Traps	97
Abbildung 147 Softerror	99
Abbildung 148 Soft Error Rate	99
Abbildung 149 Parity-Tabelle	100
Abbildung 150 2D-Parity	100
Abbildung 151 ECC Fehlererkennung	100
Abbildung 152 Centronics-Kabel.....	101
Abbildung 153 Centronics Pinbelegung.....	101
Abbildung 154 Parallel-Port-Modi.....	102
Abbildung 155 Winchester Festplatte	103
Abbildung 156 IDE-Controller.....	103
Abbildung 157 Vergleich 80/40 poliges Kabel	104
Abbildung 158 Compact-Flash-Karte	104
Abbildung 159 SCSI Terminierung.....	105

Abbildung 160 Synchrone serielle Übertragung	106
Abbildung 161 Asynchron	107
Abbildung 162 Fullduplex.....	107
Abbildung 163 Halbduplex	107
Abbildung 164 Differenzielles Paar	108
Abbildung 165 Differenzbildung	108
Abbildung 166 Manchester-Code.....	109
Abbildung 167 Taktrückgewinnung mit PLL	109
Abbildung 168 RS232 Pegel	110
Abbildung 169 MAX232.....	110
Abbildung 170 RS422 Verschaltung.....	112
Abbildung 171 RS422 Leitungslänge	112
Abbildung 172 RS485 Verschaltung.....	113
Abbildung 173 RS485 Gleichtaktfehler	113
Abbildung 174 RS485 Beispiele Busabschluss.....	114
Abbildung 175 RS485 positive Pegel.....	114
Abbildung 176 SPI Signale	115
Abbildung 177 SPI Multi-CS	116
Abbildung 178 SPI Daisy-Chain	116
Abbildung 179 I ² C Logo	117
Abbildung 180 I ² C Verschaltung	117
Abbildung 181 I ² C Adresswahl	117
Abbildung 182 I ² C Start-Stop	118
Abbildung 183 I ² C Datenübertragung	118
Abbildung 184 SATA-Kabel.....	119
Abbildung 185 SATA-Versionen.....	119
Abbildung 186 M.2 PCIe Adapter.....	120
Abbildung 187 M.2 Sockeltypen.....	120
Abbildung 188 M.2 Modultypen.....	120
Abbildung 189 USB A/B-Stecker	121
Abbildung 190 USB-Stekkertypen	121
Abbildung 191 USB-low-full-Speed.....	122
Abbildung 192 USB 2.0 High-Speed	122
Abbildung 193 USB 3 Stecker	123
Abbildung 194 Belegung USB3-Stecker.....	123
Abbildung 195 USB-Stekkertyp-C	124
Abbildung 196 USB-Adapter-Kabel	126
Abbildung 197 Koax Ethernet.....	128
Abbildung 198 Ethernet Buchse mit Übertrager	129
Abbildung 199 Twisted Pair Kabel	129
Abbildung 200 RJ45 Übertrager	129
Abbildung 201 Ethernet-Übertrager in IC	129
Abbildung 202 Vergleich Ethernet 100 Mbit/s und 1 Gbit/s	130
Abbildung 203 Ethernet Industriestecker	130
Abbildung 204 Vergleich RJ45 <-> ix	130
Abbildung 205 Single Pair Ethernet.....	131
Abbildung 206 Reichweiten SPE	131
Abbildung 207 Glasfaser Typen.....	132
Abbildung 208 Lichtausbreitung in Glasfasern.....	132
Abbildung 209 Medienkonverter für 1 Gbit-Full duplex-Glasfasern.....	133
Abbildung 210 10 GBit/s Tranceiver-Modul	133
Abbildung 211 TOSLINK	133
Abbildung 212 WLAN Überlappung	134

Abbildung 213 MIMO.....	134
Abbildung 214 LED-Ansteuerung	136
Abbildung 215 LED-Multiplexing	137
Abbildung 216 Gemeinsame Kathode.....	137
Abbildung 217 Gemeinsame Anode	137
Abbildung 218 7-Segment mehrstellig.....	138
Abbildung 219 LCD-TN-Zelle.....	138
Abbildung 220 Passiv LCD.....	139
Abbildung 221 LCD-AC-Signal.....	140
Abbildung 222 LCD-Controller	140
Abbildung 223 Ladungspumpe	140
Abbildung 224 Bildaufbau	141
Abbildung 225 Zeilensprung.....	142
Abbildung 226 RAMDAC	142
Abbildung 227 DVI-Ausführungen.....	143
Abbildung 228 Beispiel gleichanteilsfreier 4-Bit-Code	143
Abbildung 229 HDMI Stecker	144
Abbildung 230 HDMI-DVI Adapter.....	144
Abbildung 231 Display-Port-Stecker	144
Abbildung 232 2D-Figuren auf Pixelebene.....	145
Abbildung 233 Grafik ohne Beschleunigung	146
Abbildung 234 2D-Beschleunigung	146
Abbildung 235 Voodoo-Karte Relais	147
Abbildung 236 Voodoo Kabelpeitsche	147
Abbildung 237 Pixel im Hauptspeicher	148
Abbildung 238 CPU GPU RAM	148
Abbildung 239 ISP Select	152
Abbildung 240 Widerstandsfarbcodes	153
Abbildung 241 Farbcodesuhrzeit	153
Abbildung 242 Delphi IDE	154
Abbildung 243 Quellcode der Timerprocedure	154
Abbildung 244 Datenblattauszug RGB-LED	155
Abbildung 245 Stromlaufplan RGB-Uhr	155
Abbildung 246 Leiterplatte Hand	156
Abbildung 247 Layout Autorouter	156
Abbildung 248 MX-Cube-Software	157
Abbildung 249 STM-Manuals.....	157
Abbildung 250 Softwarestruktur Beispiel Keil	158
Abbildung 251 Auszug 1 Hardware-Header-Datei	159
Abbildung 252 Auszug 2 Hardware-Header-Datei	159
Abbildung 253 Software-Handles	160
Abbildung 254 Hardware-Init-Funktionen.....	160
Abbildung 255 Funktion zur GPIO-Initialisierung	160
Abbildung 256 Auszug Interrupt-Datei.....	161
Abbildung 257 Externer main()-Funktionsaufruf.....	162
Abbildung 258 Externe ISR-Aufrufe	162
Abbildung 259 Kopfteil Anwenderprogramm	163
Abbildung 260 Globale Variablen	163
Abbildung 261 Hauptroutine	164
Abbildung 262 Timerinterrupt	164
Abbildung 263 Timerinterrupt Unterfunktionen	165
Abbildung 264 Alle LEDs aus	165

11 Stichwortverzeichnis

- 1 GBit/s 130, 132, 133
10 GBit/s 124, 133
10 MBit/s 128, 132, 133
100 MBit/s 129
32-Bit-System 17
3Dfx 147
 50Ω 128
64-Bit-Adresse 18
7-Segment-Anzeigen 136, 137
Abakus 6, 7
Addition 7, 10, 48, 49, 52, 54, 55, 56, 65, 72
Adressbereich 76
Adressbus 18, 19, 21, 25, 72, 79
Adressdecoder 76, 85, 86, 103
Adresse 18, 22, 40, 43, 61, 66, 78, 80, 86, 105, 117, 118
Adressierung 18, 80, 86, 87
Adressierungsarten 67
Adressleitungen 18, 22, 23, 43, 73, 84, 86
Adressrechnung 64
Akkumulator 21, 66
Analogrechner 6
Arbeitsspeicher 20, 41, 64, 76, 80, 81, 83, 88, 89
Arithmetik-Prozessor 25
ARM 15, 17, 149, 150, 155, 157, 158
Array 37, 84, 85, 100, 142
Assemblerbefehl 64, 68, 73
Audio 26, 35, 151
Auflösung 62, 141, 142
ausfallsicher 37
Autorouter 156
AVX 63
Backup 26
Bankauswahl 88
Baudrate 107
Befehl 61, 64, 65, 66, 67, 68, 69, 71
Befehlswortlänge 67
Benchmarks 146
Bernoullieffekt 28
Beschleunigerfunktionen 148
Betriebssysteme 18, 126, 146
Biaswert 47
Bildschirmauflösung 141
Bildwiederholfrequenz 141, 142
Bit-Bang-Modus 106, 126
Bitbreite 17, 18, 52, 59
B-Key 120
Bohr-Bug 16
Bootloader 152
Bug 16
Burst 43, 80, 88
Burstlänge 80
Burstdmode 105
Burst-Modus 43, 88
Bussysteme 19, 43, 44, 131
Bustransfer 117
Byte Enable 23
Cache 20, 22, 24, 34, 41, 42, 73, 80, 81, 83, 88, 89, 148
CAS-Latency 88
CAT5 112, 129
Charge Trap 97
Chipebene 67
Compiler 61, 67, 149
Controller 20, 24, 34, 78, 81, 103, 105, 136, 140
Core 73
Cortex 15, 77, 149, 150, 155, 157
CubeMX 160
Curta 11
Daisy-Chain 116
Datenbus 18, 19, 21, 22, 23, 25, 73, 75, 76, 87
Datendurchsatz 44
Datenleitung 107, 109
Datenleitungen 86, 102, 103, 104, 105, 106, 116, 121, 122
DDR5 89
DDR-SDRAM 89
Dekrementieren 65
Delphi 154
Desktop-CPUs 149
DFÜ 110
Differenz 108, 114, 140
Differenzbildung 108, 113
Differenzverstärker 108
Digitalrechner 6, 8
DirectX 147
Disk 27, 35, 36
Division 48, 60, 61, 65
-

Dongles 102	Full-Speed 122
DRAM-Speicher 22, 85, 141, 145, 148	Füllstufen 96
Drucker 101, 121	Gasfüllung 28
DSUB 101, 111, 142	GDDR-SDRAM 89
duplex 110, 115, 117, 129	Gehäuse 79, 92, 93, 155
Durchlassspannung 136	Gleichtaktstörung 113
dynamisches RAM 20	Gleichtaktstörungen 108
EDO-RAM 87	Gleitkommaformate 47
Effektivwertgleichrichter 6	Gleitkommazahlen 47
EIA232 110	Grundrechenarten 63, 65
Einadressmaschine 66	HAL 153, 157, 160, 161, 162, 163, 164, 165
Elektromechanik 8	Halbaddierer 49, 50, 51, 53, 54
Elektronen 90, 95, 96, 97	Halbleiter 9, 32
Elektronenstrahl 141	Hardware 15, 37, 43, 45, 59, 60, 67, 70, 74, 76, 78, 111, 119, 145, 149, 150, 151, 153, 155, 157, 159, 160
Embedded 17, 19, 25, 42, 93, 98, 149, 151, 153	Hardwaremultiplizierer 67
ENIAC 8, 14	Harvard 41, 42
EPR 125	Hauptspeicher 20, 22, 80, 141, 148
erster IBM-PC 15	Heisen-Bug 16
Evolution 28, 73, 105, 119, 121, 134, 135	Hersteller 15, 20, 28, 34, 70, 93, 115, 117, 126, 140, 146, 147, 149, 150, 151
Execution 69, 70, 73	High-Speed 122, 135
Exponent 47, 72	Hitachi 140, 151
Exponenten 47, 72	Hochsprachen 67
Extended Power Range 125	Hot-Plugging 125
Failsafe 114	IBM 15, 42, 43, 142
Farbcode 153	IEEE1284 102
Feldbusse 131	Inkrementieren 54, 65
Festplatten 28, 29, 30, 31, 33, 34, 37, 38, 95, 98, 102, 103	Integrationsdichte 97, 99
Festplattencontroller 103	Interpreter 67, 74
FFT 62, 72	Interrupt 19, 65, 78, 150, 159, 161, 162
Firemonkey 154	Interruptcontroller 78
Flachbandkabel 103, 104, 105	KEIL 149
Flachbildschirme 143	Kodierung 67, 119, 128, 143
Flags 64, 65	Kollisionen 128, 134
flankengesteuert 78	Kondensator 85
Flash 27, 32, 33, 40, 83, 94, 95, 96, 97, 98, 104	Kondensatoren 125
Flashbausteine 93	Konstanten 71
Flashzellen 33, 34	Kupferkabel 132
Fließkommazahlen 25, 71, 72	Ladung 85, 91, 92, 93
Flip-Flop 84	Ladungspumpe 140
Flip-Flops 21, 48, 57, 67	Lanes 44, 45, 120
Floating-Gate 90, 91, 92, 93, 94, 96, 98	Latch 86
Flüssigkristalle 138, 139	LCD 138, 139, 140
Fourier-Transformation 62	Leckströme 85
FPU 20, 25, 62, 63, 71, 72, 73	LED-Chips 136, 155
Fullduplex 107, 123, 130, 132, 133	

Leiterbahnen	24, 44	NAND	94, 95, 97
Leiterplatte	24, 99, 103, 115, 117, 129, 152, 156	Nibble-Modus	101, 102
Leiterplatten	44	NOR	94, 95
Leiterplattenlayout	156	Nutzdaten	109, 119, 143
Leitungen	19, 23, 44, 57, 79, 101, 106, 108, 109, 111, 112, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 128, 130, 132, 140, 142, 143	Nvidia	147
Lesekopf	28, 29, 30, 98	NX-Bit	18
Leuchtdioden	136, 137	Opcode	66
Lichtbogen	125	OpenGL	147
Lithographie	97	Operanden	64, 66, 72
Lochkarten	26	Operationsverstärker	6
Löschzyklen	96	Parity	100
Low-Speed	122	Pascal	10, 154
MAC	62	pegelgesteuert	78
Magnetfeld	98	Pentium	22, 41, 67, 69, 73, 74
Magnetisierung	29, 30, 31	Performance	59, 70, 73, 145
Magnetplatten	28, 33, 83	Peripheriegeräte	76, 101
Manchester-Code	109	Pins	79
Mandel-Bug	16	Pipeline	67, 68, 69, 70, 73
Mantisse	47, 72	Pixel	139, 141, 142, 143, 145, 146, 148
Maschine	8, 13, 16, 64, 67, 70, 100	Polfilter	138, 139
Master	115, 116, 117, 118, 121	Polling	78
Master-Slave-Prinzip	115, 117	Power Delivery	125
Matrix	57, 94, 137, 155, 165	Programmcode	18, 41, 42, 70, 73, 159
MAX232	110	Prozessorhersteller	18, 67, 149, 152
Mechanik	7	Punktmatrixanzeigen	136
Messtechnik	6	radioaktive Isotope	99
MHz	20, 24, 43, 73, 83, 86, 89, 115, 128, 129, 130, 134, 135	RAID	37, 38
Mikrocontroller	15, 17, 19, 21, 73, 102, 110, 115, 138, 150, 151, 152, 153, 159	RASPI	17
Mikro-Programme	67	Rechenalgorithmen	71
MIMO	79, 134	Rechenregister	25, 48
MISO	115, 116	Rechenschieber	8
M-Key	120	Rechenwerk	8, 25, 49, 55, 67
MMX	25, 63, 73, 74	Redundanz	37, 38
mobile Massenspeicher	27	Refresh	85, 93, 141
MODEM	111	Register	17, 21, 48, 52, 62, 63, 65, 66, 69, 71, 73, 81, 83, 150
MOSI	115, 116	Registerbreite	21, 63, 72
MS-DOS	147	Registerinhalt	63
Multiplexing	22, 79, 137	Relais	8, 147
Multipliers	62	Röhren	8, 32
Multiplikation	7, 48, 58, 59, 60, 61, 62, 65, 72, 74	RS232 Schnittstelle	111, 152
Nachleuchtdauer	141	Scanner	101, 102, 121
		Schreibkopf	29, 30, 31
		Schreiblesekopf	27
		Schreib-Lese-Zyklen	98
		Schroedin-Bug	16
		SDRAM	88
		Segmente	138, 139, 140

- Shift 56, 57, 61, 65
Sicherheitslücke 70
Signalintegrität 24
Signalpegel 43, 128
Signalprozessoren 62, 151
Software 16, 25, 37, 41, 45, 62, 63, 67, 74,
 78, 80, 111, 150, 152, 153, 157, 158,
 160
Softwarealgorithmen 67
Softwareprotokoll 104, 105
Solid-State 32
Speicherbandbreite 141
Speicherbänke 88
Speicherbausteine 23, 88, 99
Speicherbereich 27, 34, 76
Speicherblöcke 34
Speicherinterface 22, 148
Speicherkanäle 22, 76
Speicherstellen 32, 34, 66, 93, 96, 97, 99
SPR 125
Sprungvorhersage 70, 73
SSD 33, 34
SSD-Festplatten 120
Standard Power Range 125
Startbit 111
Steckertyp-C 124
Steuerbus 19, 21, 75
Steuerwerk 8, 64, 67
STM 151, 157
Stoppbits 107, 111
Strahlung 98, 99
Suanpan 6
Subtraktion 10, 48, 52, 55, 56, 65
Super-Speed 123
symmetrisch 112, 121, 129, 143
Synchronisation 106, 111
Systemtakt 24
Systemtiming 75
Takt 20, 24, 44, 69, 73, 80, 109
Taktfrequenz 24, 43, 44, 80
Taktfrequenzen 20, 24, 89
Taktperiode 24, 89
Taktsignal 88, 89
Taktvervielfachung 24
Taktzyklus 56, 57, 61, 68
TDMS 143
Thunderbolt-Protokoll 124
Transistor 85, 90, 91, 97, 138
Turing 13
Twisted Pair 102, 112, 129
Twisted-Pair 108, 129, 130
UART 111, 126, 160, 161, 162, 163
Übertrag 48, 51, 52, 53
Übertragungsrate 28, 43, 44, 102, 107,
 112, 117, 122, 123, 124, 128, 132, 133,
 134, 144
UV-Licht 92
UV-Strahlung 92
verdrillt 102, 108, 112
Verschleiß 27, 34
Versorgung 39, 40, 79, 121, 130, 138, 155
Volladdierer 51, 52, 54, 59
Von-Neumann 18, 41
Von-Neumann-Zyklus 64
Voodoo 147
Vorwiderstand 155
Vorzeichen 46, 47, 56, 58, 72
Wahrheitstabelle 49, 51
Wellenlänge 36, 92, 132, 133
Wellenwiderstand 105, 122
Windows 145, 154
Zeilenrücklauf 141
Zeilensprungverfahren 142
Zelle 33, 84, 85, 93, 94, 95, 96, 98, 138,
 139
Zugriffszeit 26, 28, 33, 38, 80, 83, 86, 89,
 90
Zugriffszeiten 83
Zuse 14
Zweierkomplement 46, 52, 55, 56
-