

Folien zur Vorlesung
Informatik IV
Sommersemester 2004

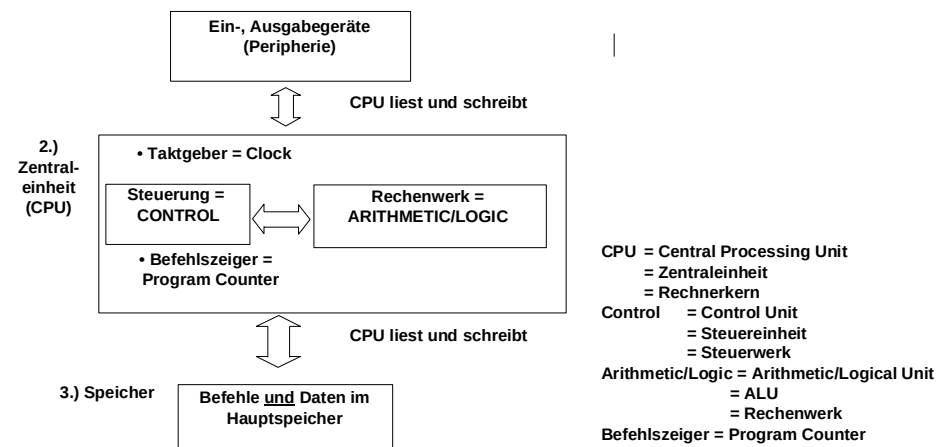
von
Prof. Dr. Harald Richter

Die Vervielfältigung oder Weitergabe
des Skripts an Dritte ist nur mit Genehmigung gestattet.
Alle Rechte liegen beim Verfasser.

1

**Die drei Hardware-Komponenten der
von Neumann-Maschine**

1.) Schnittstelle zum Bediener und der Umwelt

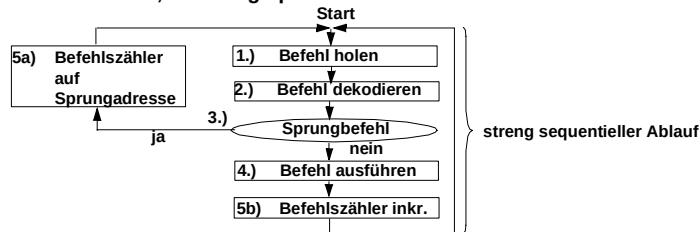


Teil A
Rechnerarchitektur

Wie arbeitet die von Neumann-Maschine?

- **Prinzip: Zentrale Befehlsschleife:** Die zentrale Befehlsschleife implementiert einen seriellen, schrittweisen Ablauf bei der Programmausführung

- Sie verhindert, daß Dinge parallel bearbeitet werden



- Dazu kommt ein „datenabhängiger Sprung“ (bedingter Sprung), er ist grundlegend für die Universalität der Maschine. Nur dadurch ist die von Neumann-Maschine so berechnungsstark wie die Turing-Maschine

- Turing-Maschine = theoretisches Modell eines Computers (Alan Turing 1936). Hat hohe Bedeutung für die Berechenbarkeits- und Komplexitätstheorie.

4

Vor- und Nachteile des von-Neumann-Prinzips

- Vorteil: Prinzip des minimalen Hardware-Aufwandes. Es kann nichts mehr weggelassen werden.

• Nachteile:

- „von Neumann-Flaschenhals“ (Flaschenhals = Engpaß): Zugriff ist langsam, da sequentiell (nur 1 Speicherport).
- „Semantische Lücke“ zwischen Anwendungsprogramm und Rechner, das heißt die Maschine kennt keine komplexen Datentypen und Operationen, so wie sie der Anwender braucht.

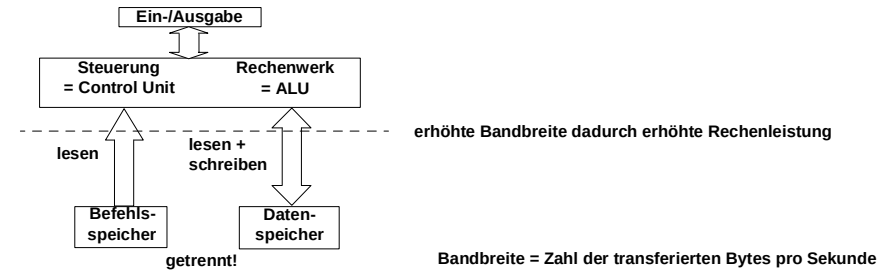
6

Alternative zum von-Neumann-Rechner z. B. Harvard-Architektur

- Befehle- und Daten sind getrennt in verschiedenen Speichern untergebracht
- Befehle und Daten können gleichzeitig geholt werden.

- Erhöhter Aufwand, nur in sog. Echtzeitanwendungen gerechtfertigt

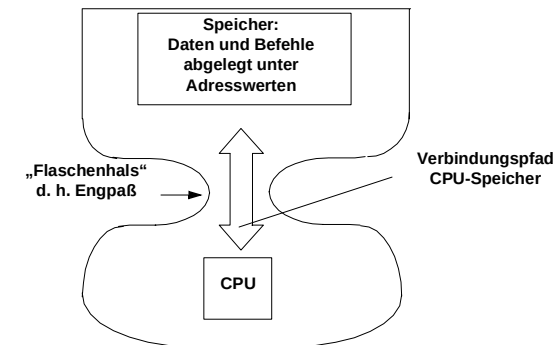
- Beispiel: „Signalprozessor“ TI 320C40 (TI = Texas Instruments = Prozessor-Hersteller)



- Nachteil: Ist der Datenspeicher nicht ganz voll, kann er nicht zur Speicherung von Befehlen verwendet werden und umgekehrt

5

Erläuterung des von Neumann-Flaschenhalses

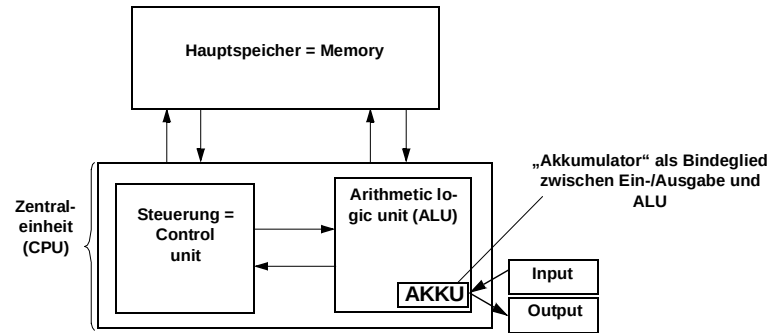


- Der v. Neumannsche Flaschenhals ist bis heute eines der nur näherungsweise gelösten Probleme der Rechnerarchitektur

- Eine der Lösungsmöglichkeiten sind sog. Caches

7

Erweiterung des von-Neumann-Rechner durch einen „Akkumulator“

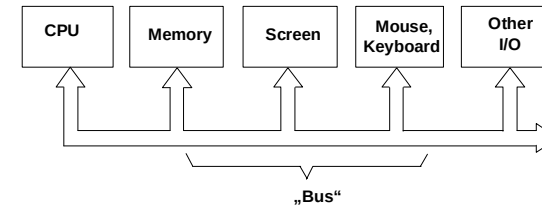


Kennzeichen:

- 1.) Programm und Daten werden in einem Speicher abgelegt (stored program computer)
- 2.) Der Speicher ist für beide gemeinsam
- 3.) Der Akku dient als CPU-interner, schneller Zwischenspeicher zum Rechnen und für die Ein-/Ausgabe

8

Erweiterung des von Neumann-Rechners durch einen „Bus“



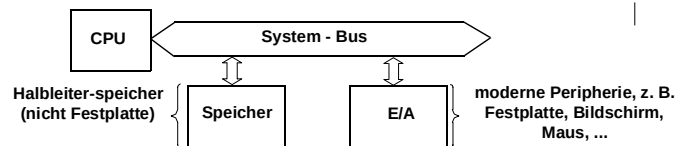
- Aus Kosten- und Effizienzgründen kann ein gemeinsames Transportmedium für Datenbits verwendet werden: der Bus.

- Omnibus lat. „für alle“; Bus ist die Abkürzung von Omnibus

9

Allgemeine Computer-Grundstruktur

- Ein typischer, konventioneller Computer besitzt folgende Grundstruktur:



- Kennzeichnend ist, daß es nur einen zentralen Verbindungspfad für die drei Hauptkomponenten CPU, Speicher und E/A — den sog. Systembus — gibt.

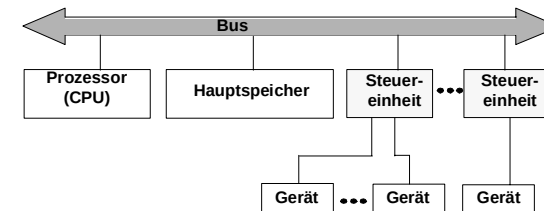
(Hinweis: Eine **Schnittstelle** verbindet *genau zwei* Seiten bzw. Teilnehmer miteinander; ein **Bus** verbindet *mehr als zwei* Teilnehmer.)

⇒ Problem des von Neumann-Flaschenhalses ist verstärkt, da auch die Ein-/Ausgaben über den gemeinsamen Bus gehen.

Bemerkung: Zur Leistungssteigerung werden häufig 2 Busse eingesetzt: 1 Systembus und 1 Ein-/Ausgabe-bus (z.B. PCI-Bus).

10

Grobarchitektur eines Universalrechners mit Peripherie, ohne E/A-Bus



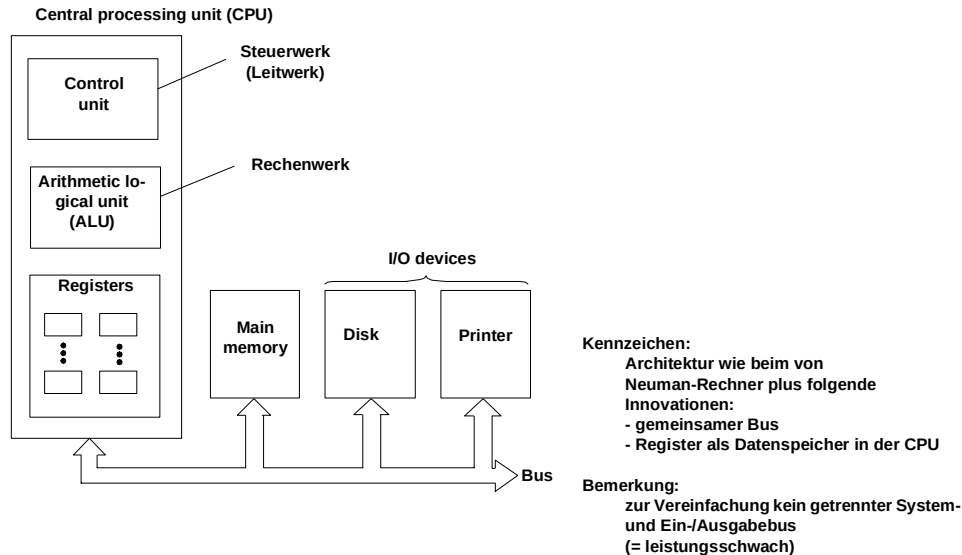
- Steuereinheiten (Controller) können intelligente Bausteine bzw. Spezialprozessoren für die Peripherie sein. Steuereinheit ist hier ein Oberbegriff für E/A-Module unterschiedlicher Leistungsfähigkeit vom einfachen seriellen Schnittstellen-Modul bis zum komplexen E/A-Prozessor (EA = Ein-/Ausgabe)

Beispiele:

Bezeichnung	Gerät	
serielle Schnittstelle (RS 232C, V.24)	Maus, Modem	einfach
parallele Schnittstelle (Centronics)	Drucker	
SCSI	Platte, Scanner	komplex
Ethernet-Adapter	Lokales Netz	

11

Zusammenfassung heutiger Rechner (nicht sehr leistungsstark)



12

Hauptspeicher

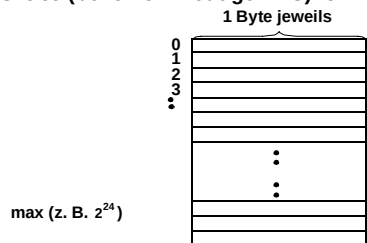
Der Hauptspeicher dient der temporären Speicherung der gerade aktiven Programme und ihrer Daten.

Er ist ein Halbleiterspeicher, der durch Siliziumchips realisiert ist.

Logisch gesehen besteht er aus einer Folge von Bytes die einzeln gelesen und geschrieben werden können (RAM = Random Access Memory).

Sein Inhalt geht beim Ausschalten des Rechners verloren. Nach dem Einschalten wird das Betriebssystem geladen.

Typische Größe (bei einem heutigen PC): 512 MByte

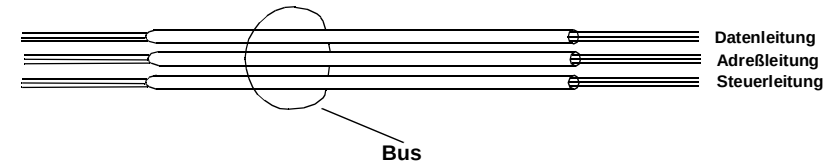


14

Überblick über die wichtigsten Komponenten eines Rechners

Der Bus

- Ist ein Übertragungsmedium, das von mehreren Einheiten gemeinsam benutzt wird
- Besteht aus einer Menge paralleler Leitungen in drei Gruppen



- Busse unterscheiden sich erheblich bezüglich der Daten und Adreßbreite, der Art der Zugangsregelung, der Anzahl der anschließbaren Geräte u. a.
- Ein typischer heutiger Systembus verfügt über 32 Daten- und 32 Adressleitungen und hat eine Bandbreite von ca. 1,5 GB/s. Ein Peripheribus (z. B. VME, PCI) hat ca. 150 MB/s

13

Plattenspeicher

- Der Plattenspeicher dient der *permanenten* Speicherung von Programmen und Daten. Typische Kapazität (in einem heutigen PC): 10 - 100 Gbyte.

- Ein Plattenspeicher besteht aus einer oder mehreren Platten oder Scheiben, deren Oberflächen magnetisierbar sind und die mit hoher Geschwindigkeit rotieren (> 4000 Upm).

- Für jede magnetisierbare Oberfläche existiert ein Schreib-/Lesekopf, der hin- und herbewegt werden kann, um die gesamte Oberfläche zu erreichen. Die Einrichtung, an der die Köpfe befestigt sind, heißt Plattenarm. Besteht der Plattenspeicher aus mehreren Einzelplatten, so hat der Plattenarm die Form eines Kamms. (Jeder Zacken den Kamm ist ein Schreib-(Lesekopf).

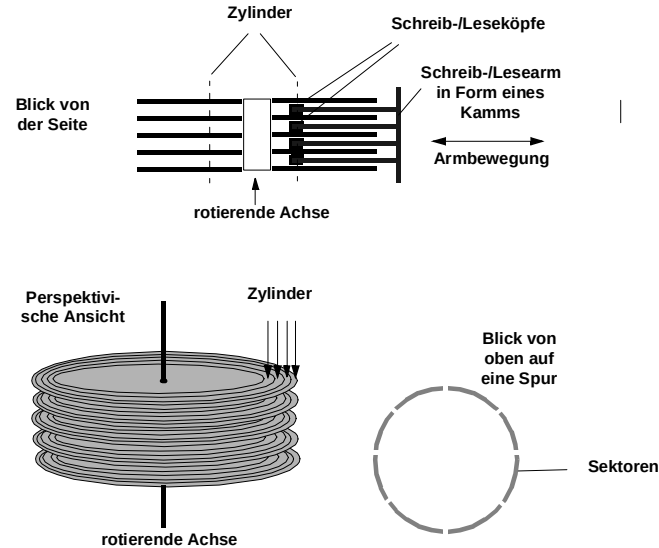
- Jede Oberfläche besteht aus konzentrischen Spuren (500 - 2000), auf denen die Information aufgezeichnet ist.

- Jede Spur wiederum ist in mehrere Sektoren (32 - 128) aufgeteilt. Diese Sektoren sind die kleinsten les- oder schreibbaren Einheiten (z. B. 512 Bytes), sog. Blöcke.

- Die Gesamtheit aller Spuren, die mit einer festen Armposition gelesen werden kann, wird auch Zylinder genannt.

15

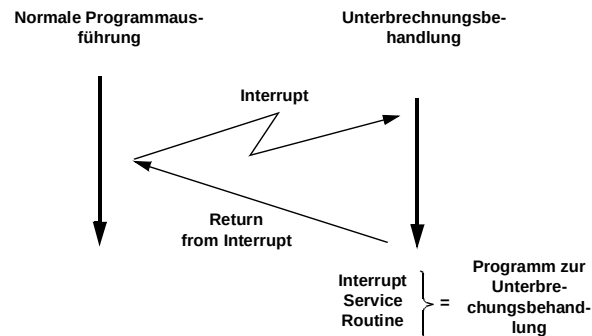
Organisation einer Festplatte



16

Unterbrechungen (Interrupts)

- Der Bus verfügt über (mindestens) eine Unterbrechungsleitung, die an den Prozessor angeschlossen ist.
- Nach jedem Befehl prüft der Prozessor, ob an dieser Leitung ein Signal (Spannung) anliegt.
- Falls ja, wird augenblicklich eine Prozedur angesprungen, die die Unterbrechung auswertet und dann die erforderlichen Aktionen durchführt oder veranlasst.

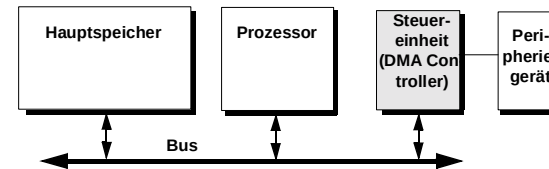


18

Datentransport von/zu Peripherieeinheiten

Der Datentransport zwischen Ein-/Ausgabegeräten und dem Hauptspeicher wird heute üblicherweise per DMA (direct memory access) durchgeführt.

DMA bedeutet, dass die Steuereinheit des Geräts in der Lage ist, selbstständig auf den Hauptspeicher zuzugreifen. (selbstständig = ohne CPU)



Der Prozessor muss zum Auslösen des DMA-Transfers entsprechende Register in der jeweiligen Steuereinheit laden, um z. B. Quelle, Ziel und Umfang des Datentransfers anzugeben.

Nach Beendigung der E/A muß der Prozessor informiert werden. Dazu gibt es zwei Möglichkeiten:

- **Polling:** Der Prozessor fragt periodisch das Statusregister der Steuereinheit ab (In den meisten Fällen zu ineffizient)
- **Unterbrechung (Interrupt):** Der Prozessor wird vom Ende der Übertragung durch ein spezielles Signal informiert (= übliches Verfahren)

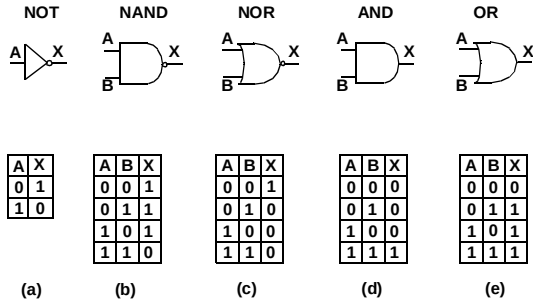
17

Gatter- und Registerebene in der Zentraleinheit (CPU) eines Rechners

= unterste Hardwareebene des Prozessors

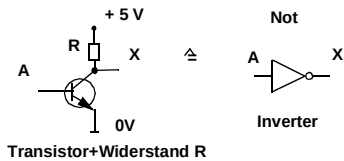
19

Elementare Gatterschaltungen (1)



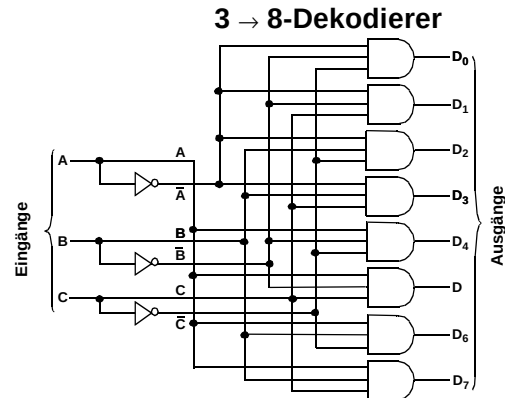
- a) Invertierer (NOT)
b) invertiertes UND-Gatter (NAND)
c) invertiertes ODER-Gatter (NOR)
d) UND-Gatter (AND)
e) ODER-Gatter (OR)

Beispielimplementierung eines Gatters:



20

Beispiel einer Gatterzusammenschaltung



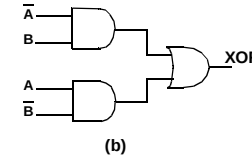
- setzt einen von 8 Ausgängen auf 1
- Auswahl des Eingangs erfolgt im Binärcode
- wird verwendet zur Dekodierung der Befehls Worte in der CPU

22

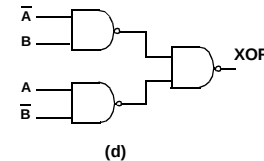
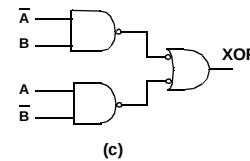
Elementare Gatterschaltungen (2) XOR

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

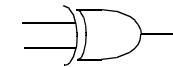
(a)



- a) Wertetafel des ausschließenden ODER (Exclusive OR, EXOR, XOR)
b)- d) drei äquivalente Implementierung von EXOR



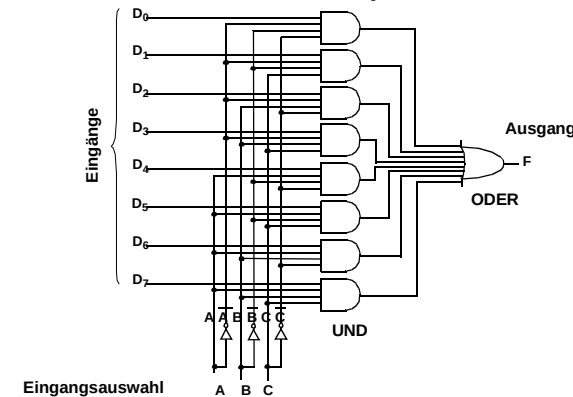
EXOR - Symbol



- Mit EXOR kann man einen der beiden Eingänge invertiert auf den Ausgang geben. Diese Funktionalität wird in der Praxis oft gebraucht.

21

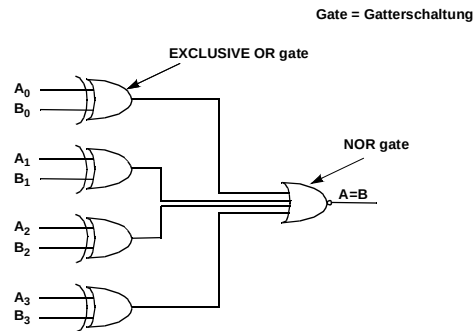
8 → 1-Multiplexer



- wählt einen von 8 Eingängen aus und gibt ihn auf den Ausgang
- Auswahl des Eingangs erfolgt im Binärcode
- nötig, um Register oder Rechenwerk aus verschiedenen Quellen mit Operanden zu versorgen

23

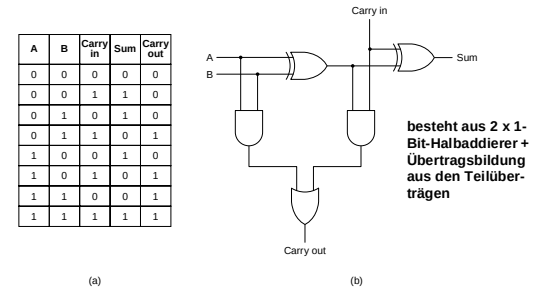
4-Bit-Komparator



- vergleicht zwei 4-Bit-Worte miteinander
- wird im Rechenwerk für die Ausführung des Assembler-Befehls COMPARE A,B verwendet

24

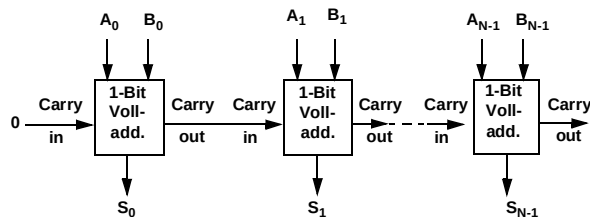
1-Bit-Volladdierer



- berücksichtigt zusätzlich den Übertrag einer vorangegangenen Addition
- mehrere 1-Bit Volladdierer können zu einem N-Bit-Voll-addierer kaskadiert werden! (= Syntheseschritt)
- wird so ähnlich im Rechenwerk für die Ausführung des Assembler-Befehls ADD A,B verwendet

25

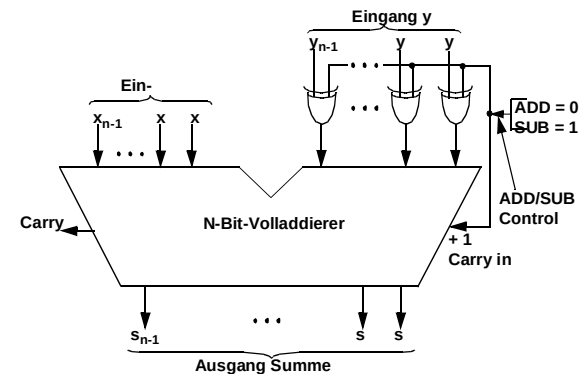
N-Bit Volladdierer (carry propagation adder)



- An die 2N Eingänge werden die je N Bits der Worte A und B angelegt
- Am Ausgang erscheint die Summe S incl. Übertrag (carry)
- Der Übertrag entsteht durch Ausbreitung des Teilübertrags von links nach rechts (= carry propagation)
- Nachteil: carry propagation hat Zeitkomplexität $O(\text{Zahl der Bits pro Addition}) = O(N)$

26

N-Bit-Addierer/Subtrahierer



- Die Steuerleitung ADD/SUB wählt aus, ob addiert oder subtrahiert werden soll. Die EXOR-Gatter dienen als steuerbare Inverter

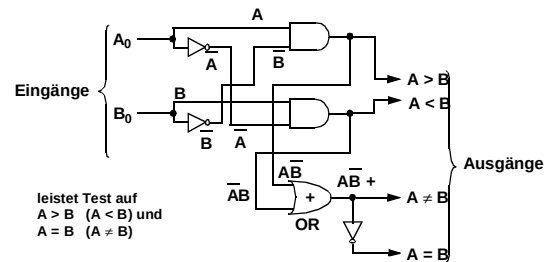
$$\text{ADD: } S = x + y$$

$$\text{SUB: } S = x + \overline{y} + 1 \triangleq x - y$$

2er-Komplement von y

27

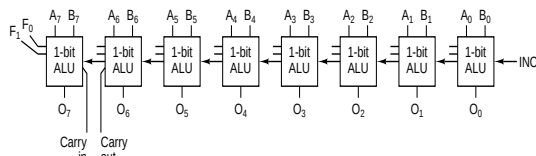
Einfache arithmetische Einheit (arithmetic unit)



- wird im Rechenwerk für die Ausführung des Assembler-Befehls CMPGT A,B (compare on greater than) und CMPEQ A,B (compare on equal) verwendet

28

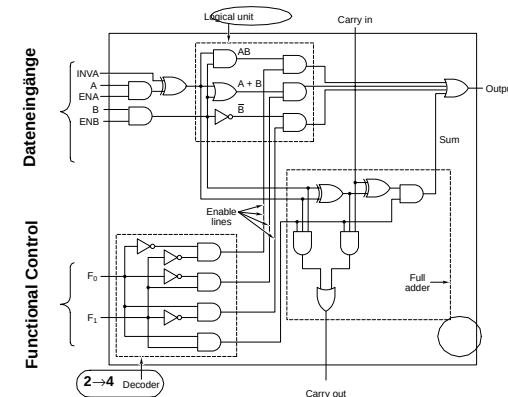
8-Bit-Rechenwerk (ALU)



- Erweiterung des 1-Bit-Rechenwerks auf 8 Bit (grössere Wortbreiten analog)
- Kaskadierung erfolgt über 8 1-Bit-Rechenwerke, die über Carry out → Carry in gekoppelt sind (=Syntheseschritt)
- ALU = Arithmetic/Logical Unit = Rechenwerk
- Nachteil von carry-out-carry-in-Kopplung: O(n)-Zeitkomplexität

30

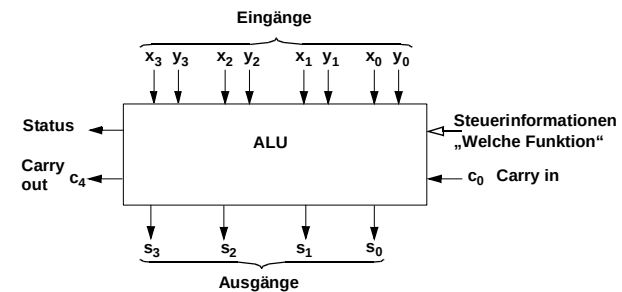
1-Bit-Rechenwerk (Arithmetisch-Logische-Einheit, ALU)



- kann auf Wunsch eine von vier Funktionen ausführen: A AND B, A OR B und A ADD B sowie $\bar{B}$
- Der Inhalt des A-Eingangs kann mit Hilfe des INVA-Signals zusätzlich invertiert werden
- ENA, ENB = Freigabe des Eingangs (Enable)

29

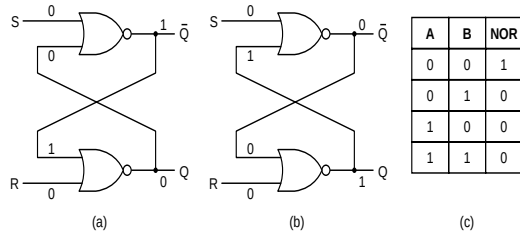
Verallgemeinertes Rechenwerk (ALU)



- Bei einer allgemeinen ALU gibt ein Bündel von Steuerleitungen an, welche Funktion von der ALU ausgeführt werden soll
- Neben dem Ergebnis am Ausgang gibt die ALU noch zusätzlich eine Statusmeldung aus, die Auskunft über Erfolg oder Mißerfolg der ausgeführten Operation gibt

31

1-Bit-Speicher (SR-Latch) mit Setzen/Rücksetzen



a) SR-Latch zurückgesetzt ($Q=0$)

b) SR-Latch gesetzt ($Q=1$)

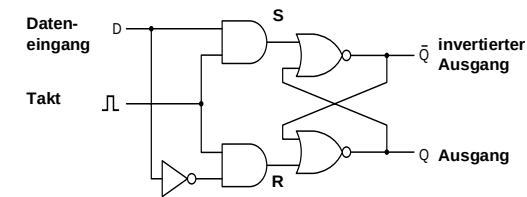
c) Wahrheitstafel eines NOR-Gatters

- der Latch-Ausgang Q hat zwei stabile Zustände: 0 oder 1
- der Latch-Ausgang Q wird gesetzt ($=1$) mit einer 1 am Setzeingang S
- der Latch-Ausgang Q wird zurückgesetzt ($=0$) mit einer 1 am Rücksetzeingang R

S, R = Set, Reset; Latch = (engl.), Schnappschloß

32

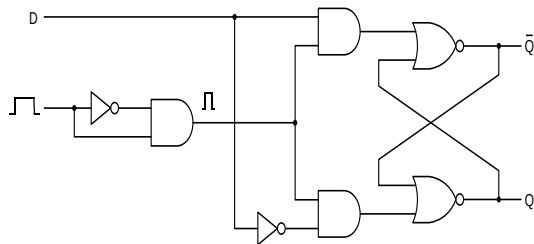
Taktgesteuerter 1-Bit-Datenspeicher (D-Latch)



- Erweiterung des SR-Latches um einen Takteingang
- Sobald der Takteingang aktiviert ist ($=1$) kann der Zustand des SR-Latches verändert werden
- der Zustand des SR-Latches wird von dem Wert am Dateneingang D bestimmt ($D=0 \Rightarrow Q=0$, $D=1 \Rightarrow Q=1$), d.h. der Ausgang Q speichert den Wert am Eingang D
- Der Vorteil ist, daß der Ausgang sich nur gleichzeitig mit dem Takt ($=$ synchron) ändern kann
- Taktsynchrone Schaltungen sind leichter beherrschbar als asynchrone, deswegen sind CPUs taktsynchron

33

Taktflankengesteuerter 1-Bit-Speicher (D-Flip-Flop)



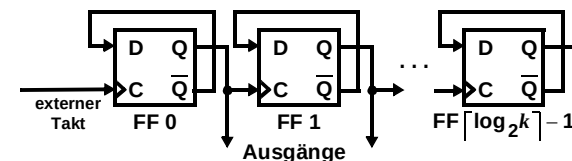
- Erweiterung des D-Latches so, daß nicht der Pegel am Takteingang, sondern ein Impuls beliebiger Länge genügt, um den Wert vom Eingang D zu speichern
- Der Taktimpuls wird durch die Gatterdurchlaufzeit des Inverters zu einem kurzen Impuls fester Länge umgewandelt
- D-Flip-Flops sind die wichtigsten statischen Speicherelemente

34

Modulo-k-Zähler

- Mehrere D-Flip Flops können zu einem sog. Zähler zusammengeschaltet werden
- Modulo-k-Zähler zählen anhand eines äußeren Taktimpulses von 0 bis $k-1$ und beginnen dann von vorne
- Die FF-Ausgänge bilden ein Binärwort ($=$ Zählerzustand)
- Für k verschiedene Zählerzustände, sind $\lceil \log_2 k \rceil$ FFs erforderlich
- Zähler können in den Varianten Synchron- und Asynchronzähler realisiert werden

Asynchroner Modulo-k-Zähler



35

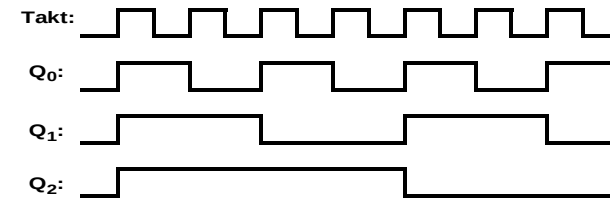
- Durch die Rückkopplung von $\bar{Q}$ -Ausgang zu D-Eingang ändert sich mit jedem Taktimpuls der Q-Ausgang und damit auch Q gemäß: $Q_{\text{neu}} := Q_{\text{alt}}$
- Da $Q, \bar{Q} \in \{0, 1\}$, geht Q mit jedem Taktimpuls in seinen komplementären Zustand $\bar{Q}$ über
 $\Rightarrow$ Q erzeugt die Folge 0,1,0,1,0,... (Anfangszustand von Q sei 0)
- Die 0/1-Folge von Q_i liefert den Taktimpuls für den Eingang C_{i+1} des nachfolgenden FFs ($0 \leq i \leq k-2$)
- Jede Periode von Q_i (= 1 Hin- und Her-Wechsel) liefert genau einen Taktimpuls für C_{i+1}
- Für jede Periode von Q_i sind seinerseits 2 Taktimpulse bei C_i erforderlich
 $\Rightarrow C_i = 2 C_{i+1}$, d.h. FF_{i+1} ändert seinen Ausgang Q_{i+1} nur halb so schnell wie FF_i $Q_i \Rightarrow$ FF-Kette hat die Funktion eines "Frequenzteilers"
- Beispiel: $k = 3$ und rückwärts zählen; zu Beginn: $Q_i = 0$

36

Synchroner Modulo-k-Zähler

- Zur Vermeidung der Summierung der Schaltverzögerungen werden synchrone Zähler eingesetzt
- Bei Synchronzählern ändern sich alle FF-Ausgänge gleichzeitig
- Dazu ist es erforderlich, daß alle FF-Eingänge D_i ebenfalls gleichzeitig mit dem Wert beaufschlagt werden, den der Ausgang Q_i mit dem nächsten Taktimpuls annehmen soll
- Definition: zum Zeitpunkt $T = n$ sei der Zählerzustand Z^n mit $Z^n = (Q_{k-1}, Q_{k-2}, \dots, Q_0)^n$, ($Z^n =$ „z oben n“)
- Die Werte D_i für den nachfolgenden Zählerzustand Z^{n+1} müssen einen Zustand vorher (d.h. in Z^n) berechnet und an die Dateneingänge D_i angelegt werden
 $\Rightarrow$ Mit dem jedem Taktimpuls können sich alle FFs gleichzeitig ändern, da die Verkettung $Q_{i-1}D_i$ entfällt
- Zur Vorausberechnung der D_i -Werte ist eine extra Hardware erforderlich

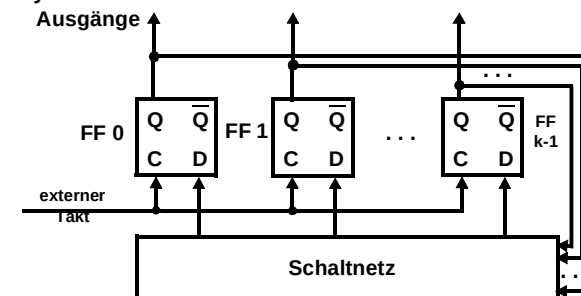
38



- Nachteil von Asynchron-Zählern:
- Jedes FF hat intern eine Schaltzeit T_s zwischen D-Eingang und Q-Ausgang
 $\Rightarrow$ Ausgang Q_{k-1} ändert sich erst kT_s Schaltzeiten später als Taktimpuls C_0 , da k FFs in Serie geschaltet sind
 $\Rightarrow$ FF-Ausgänge ändern sich asynchron zueinander und zum äußeren Takt, daher die Bezeichnung Asynchron-Zähler
- Anwendungen von Asynchron-Zählern:
 - als Frequenzteiler
 - als einfaches Leitwerk in einem Prozessor (nur solange wie kT_s klein gegenüber der Periode des Takts ist)

37

- Als Hardware wird ein sog. Schaltnetz (=kombinatorische Logik) verwendet
- Definition: Ein Schaltnetz besteht aus einer Menge von UND-, ODER und NICHT-Gattern ohne Speicherfunktionen. Rückkopplungen, so daß ein FF entsteht, sind verboten.
- Prinzipschaubild Synchronzähler:



- Das Schaltnetz bildet eine Rückkopplung von Q_i -Ausgängen zu D_i -Eingängen zur Verarbeitung des jeweiligen Zählerzustandes Z^n zur Zeit $T = n$ für alle n
- Definition: Ein Schaltwerk besteht aus einer Menge von UND-, ODER, NICHT-Gattern und FFs

39

inclusive von Rückkopplungen

⇒ Ein Schaltwerk hat u.a. eine Speicherfunktion

⇒ Ein Synchronzähler ist ein Beispiel für ein Schaltwerk

- Die Frage ist, wie das Schaltnetz intern aufgebaut sein muß, um die D_i -Eingänge zu jedem Zeitpunkt t richtig setzen zu können

- Diese Frage wird durch die sog. Schaltnetzsynthese beantwortet

Beispiel: Schaltnetzsynthese für Synchronzähler mit $k=3$

- Schritt 1: Aufstellen der Wahrheitstabellen für Zählerzustand Z^n und Z^{n+1}

Z^n			Z^{n+1}		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0

40

$$D_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0 + \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

Karnaugh-Diagramme:

Für D_0 :

	Q_0
$Q_2 Q_1$	
00	1 0
01	1 0
10	1 0
11	1 0

Für D_1 :

	$Q_1 Q_0$			
	00	01	10	11
Q_2	0 0	1 1	1 1	0 0
	1 0	1 1	1 1	0 0

Für D_2 :

	$Q_1 Q_0$			
	00	01	10	11
Q_2	0 0	0 0	0 0	1 0
	1 1	1 1	1 1	0 0

- Schritt 3: Optimieren der Gleichungen bzw. Zusammenfassen von Zuständen in den Karnaugh-Diagrammen zur Schaltnetzvereinfachung

Resultat der Vereinfachung im Beispiel:

$$D_0 = \overline{Q_0}$$

$$D_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot (\overline{Q_1} \cdot \overline{Q_0})$$

42

Z^n			Z^{n+1}		
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

- Dabei wird die Z^n -Tabelle als Eingangsinformation für das Schaltnetz und die Z^{n+1} -Tabelle als Ausgangsinformation verwendet: $Z^n \rightarrow Z^{n+1}$

- Mit jedem Takt gibt es einen Übergang $Z^n \rightarrow Z^{n+1}$

- Das Schaltnetz berechnet die sog. Übergangsfunktion so, daß gilt $(D_2, D_1, D_0)^n := (Q_2, Q_1, Q_0)^{n+1}$ (Gl. 1)

Schritt 2: Bestimmen der Übergangsfunktion entweder durch Aufstellen der Booleschen Gleichungen oder der Karnaugh-Diagramme gemäß (Gl. 1)

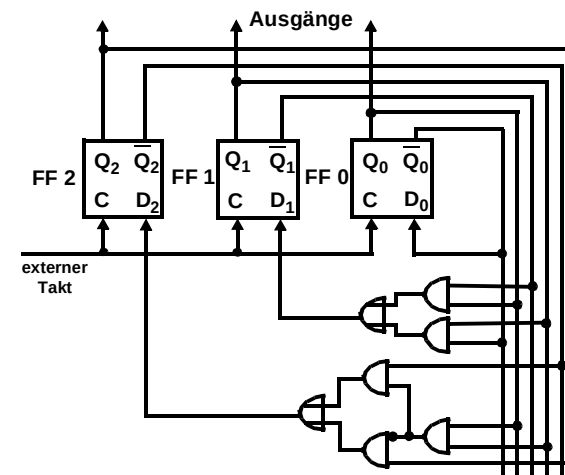
$$D_0 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

$$D_1 = \overline{Q_2} \cdot \overline{Q_1} \cdot Q_0 + \overline{Q_2} \cdot Q_1 \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot Q_0 + Q_2 \cdot Q_1 \cdot \overline{Q_0}$$

41

$$D_1 = \overline{Q_1} \cdot Q_0 + Q_1 \cdot \overline{Q_0}$$

- Ergebnis der Schaltnetzsynthese:



43

- Anwendungen von Synchron-Zählern:

- als Frequenzteiler
- als einfaches Leitwerk in einem Prozessor mit hohen Taktfrequenzen
- Nachteil: nicht immer ist die natürliche Zählreihenfolge 0,1,...,k-1 gewünscht sondern eine Permutation davon

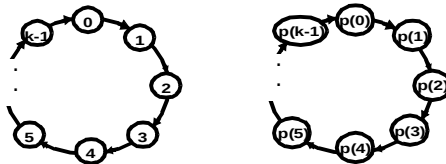
Synchron-Zähler mit permutierten Zählfolgen

- Fakt: Übergangsfunktion eines Synchronzählers wird mittels Tabellen für die Zählerzustände Z^n spezifiziert (für alle n)
- Jede Übergangsfunktion $Z^n \rightarrow Z^{n+1}$ mit endlicher Grundmenge ist mittels einer Tabelle spezifizierbar
- Diese kann mit Hilfe der gezeigten Schaltnetzsynthese in eine Hardware umgesetzt werden
- ⇒ Für jede beliebige endliche Übergangsfunktion gibt es ein dazugehöriges Schaltnetz

Beispiel: Ein Synchronzähler soll so konstruiert werden, daß er in einer permutierten Reihenfolge zählt

44

- Beispiel: EAs von Zählern für natürliche und permutierte Zählreihenfolge



- **Definition:** Die textuellen Bezeichnungen der Knoten des Graphen heißen Knotenmarkierungen
- Ebenso sind sog. Kantenmarkierungen zulässig
- Kantenmarkierungen im EA-Graphen haben die Bedeutung eines "endlichen Automaten mit Eingabe"

Endliche Automaten mit Eingabe

- Die EAs der gezeigten Zähler wechseln den Zustand mit jedem Taktimpuls
- Sie reagieren damit auf kein anderes äußeres Signal als den Takt
- EAs können dahingehend erweitert werden, daß nur dann ein Übergang zwischen zwei ben-

46

- Sei $p(0), p(1), \dots, p(k-1)$ eine permutierte Zählweise mit $p(i)$ als umkehrbar eindeutige Funktion (= Permutationsfunktion)

- Dann ersetze für alle i die Tabelleneinträge i von Z^n und Z^{n+1} durch $p(i)$, und erhalte so das Schaltnetz für die permutierte Zählweise

⇒ Die Schaltnetzsynthese für beliebige Übergangsfunktionen erfordert dieselben Schritte wie beim Zähler der Folge 0,1,...,k-1

Endliche Automaten

- Zum besseren Vergleich verschiedener Zähler dient das formale Hilfsmittel der endlichen Automaten
- **Definition:** Ein endlicher Automat (EA) ist ein Graph aus Knoten und gerichteten Kanten, wobei die Knoten die Zählerzustände und die Kanten die Übergänge zwischen den Zuständen repräsentieren

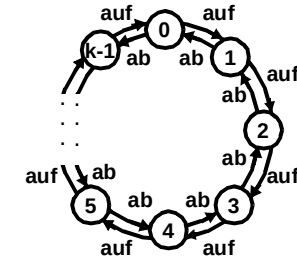
45

achbarten Knoten möglich ist, wenn zusätzlich eine Bedingung erfüllt ist

- Als Bedingung für das "Weiterschalten" können externe Signale verwendet werden

⇒ Die externen Signale dienen als Eingabe in den EA

- Beispiel: Zähler, der aufwärts und abwärts zählen kann



- Der EA des Auf-/Ab-Zählers wertet ähnlich wie ein Lift die beiden Signale "auf" und "ab" zur Steuerung aus
- Sind beide gesetzt oder keines davon, wird kein Zustandsübergang durchgeführt

47

Implementierung von endlichen Automaten mit Eingabe

- EAs mit Eingabe können ebenfalls durch einen Synchronzähler mit Schaltnetz implementiert werden
- Dazu muß das Schaltnetz nicht nur die Zählerzustände Z, sondern auch die Eingangssignale E auswerten
- ⇒ Ein Synchronzähler mit Schaltnetz, das über zusätzliche Steuereingänge verfügt, implementiert einen EA mit Eingabe

48

Schaltnetzsynthese beim Synchronzähler mit Steuereingängen

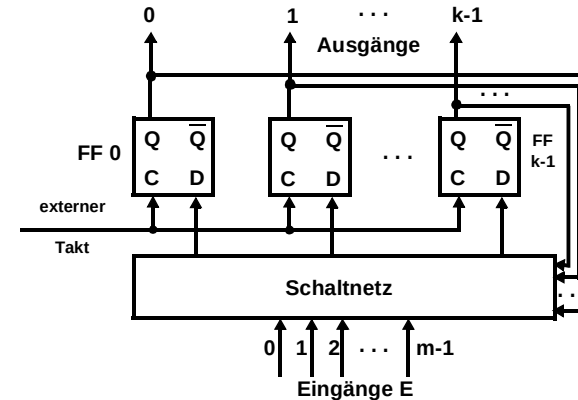
- Die Frage beim Synchronzähler mit Steuereingängen ist, wie die Schaltnetzsynthese abläuft
- Dazu müssen die Tabellen für die Zählerzustände Z^n und Z^{n+1} um neue Spalten für die Werte der Eingangssignale E erweitert werden
- ⇒ Die Übergangsfunktion wird durch E, Z^n und Z^{n+1} spezifiziert: $(E, Z^n) \rightarrow Z^{n+1}$
- Wie zuvor erfüllt dann die Übergangsfunktion die Gleichung $(D_{k-1}, \dots, D_1, D_0)^n = (Q_{k-1}, \dots, Q_1, Q_0)^{n+1}$ für alle n
- Ein Synchronzähler mit Steuereingängen kann durch tabellarische Spezifikation der Übergangsfunktion $(E, Z^n) \rightarrow Z^{n+1}$ jeden endlichen Automaten mit Eingabe implementieren

Endliche Automaten mit Ein- und Ausgabe

- Ein EA mit Ein- und Ausgabe kann ein Ausgangssignal erzeugen, das mehr ist als die Ausgabe seines momentanen Zustandswertes
- Es müssen zwei Varianten unterschieden werden:

50

- Prinzipschaubild Synchronzähler mit Steuereingängen:



49

Mealy-Automat und Moore-Automat

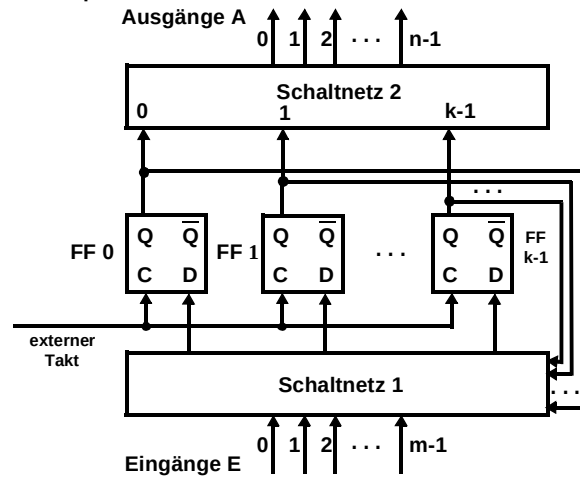
- Beim Moore-Automat ergibt sich der Ausgabewert als Funktion des jeweiligen Zustandswertes.
- Die Ausgabe ist synchron zum Takt
- Beim Mealy-Automat ergibt sich der Ausgabewert als Funktion des Zustandswertes und der Eingabewerte
- Die Ausgabe kann zu jedem Zeitpunkt auf Eingabeänderungen reagieren
- ⇒ Die Ausgabe ist asynchron zum Takt
- Formal wird der Graph eines EA mit Ein- und Ausgabe um eine Knotenmarkierung erweitert, die den Ausgabewert darstellt

Implementierung von endlichen Automaten mit Ein- und Ausgabe

- EAs mit Ein- und Ausgabe können durch einen Synchronzähler mit extra Schaltnetz implementiert werden

51

- Prinzipschaubild Moore-Automat:



- Die Synthese des Ausgabeschaltnetzes erfolgt ähnlich zu der des Eingabeschaltnetzes
- Es wird tabellarisch die Ausgabefunktion als Abbildung von Z auf die n Ausgänge spezifiziert

52

- Durch die direkte Verbindung von Eingängen E des Synchronzählers zum 2. Schaltnetz ergibt sich eine "Bypass"-Funktion
- Die Bypass-Funktion erlaubt, auf jede Änderung am Eingang E unmittelbar mit einer Änderung am Ausgang A zu reagieren

⇒ Mealy-Automat arbeitet mit asynchroner Ausgabe

- Vorteil von Mealy- gegenüber Moore-Automat:

- Mealy-A. enthält Moore-A. als Teilmenge
- ⇒ alles, was Moore-A. kann, kann auch Mealy-A.
- Mealy-A. reagiert schneller auf Eingaben als Moore-A.

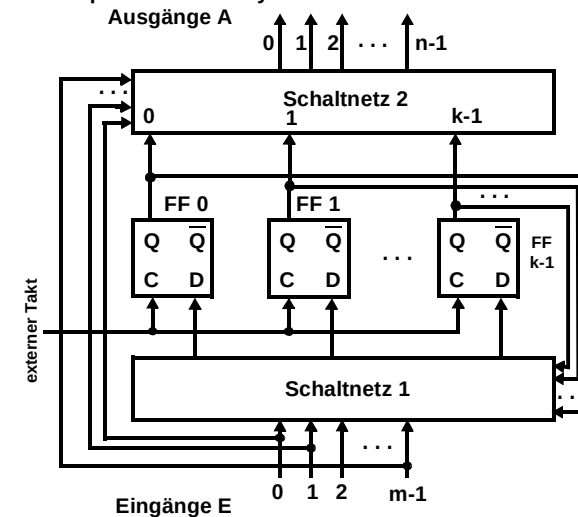
- Nachteil von Mealy-A.

- asynchrone Ausgaben sind beim Zusammenspiel mehrerer EAs mit Ein- und Ausgabe schlechter zu beherrschen als synchrone, d. h. die Kopplung mehrerer Mealy-A. ist i. A. schwierig

54

- Daraus werden die Booleschen Gleichungen oder die Karnaugh-Diagramme für jeden Ausgang aufgestellt und anschließend optimiert

- Prinzipschaubild Mealy-Automat:



53

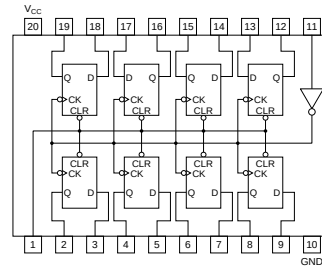
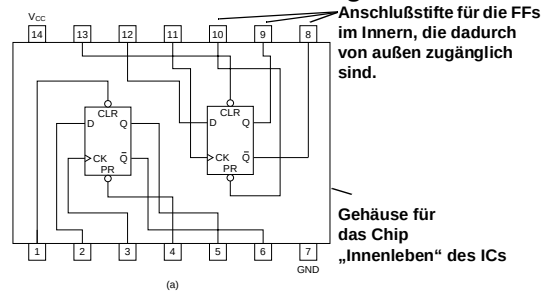
Registerspeicher

(= Vektor von 1-Bit-FFs zur Speicherung von Datenworten in der CPU)

55

Beispiele für 2-Bit- und 8-Bit-Register

Beispiele für Register, die als integrierte Schaltungen kommerziell erhältlich sind. Es gilt folgende Sprachregelung: Chip + Gehäuse für den Chip = integrierte Schaltung = integrated Circuit = IC



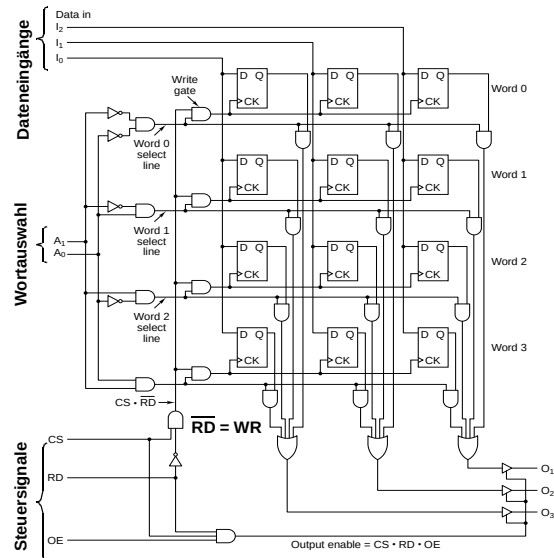
56

- Aus mehreren D-Flip-Flops wird ein Register aufgebaut. (Ein Register speichert mehrere Bits, sog. Bitvektor)
- Die gezeigten Register sind in einen Baustein (Chip) mit 14 bzw. 20 Anschlußstiften integriert und kommerziell erhältlich (Serie 74xx von Texas Instruments)
- Die Eingänge PR = Preset (setzt Q auf 1) und CLR = Clear (setzt Q auf 0) sind invertiert = „active low“

chip = Splitter, Span; hier: kleines Blättchen aus Silizium

57

Statisches RAM (SRAM)



58

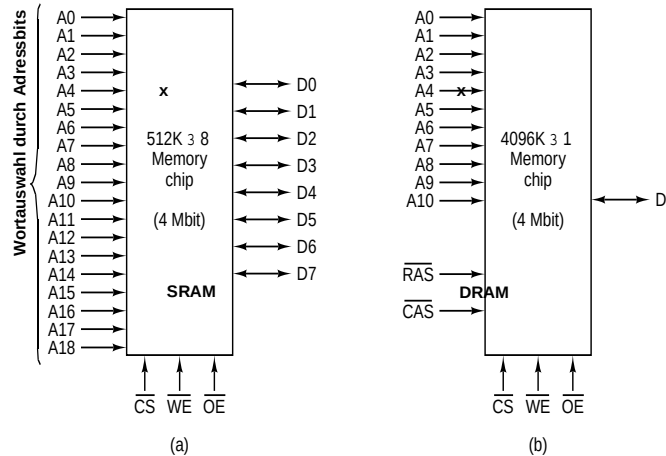
- mehrere Register speichern eine Matrix aus Bits (Jede Registerzeile heißt „Wort“)

Im Beispiel werden 4 Worte zu je 3 Bit gespeichert (I=Dateneingang, O=Datenausgang, A0A1=Wortauswahl, CS=Bausteinaktivierung, OE=Wort ausgeben)

I = Input, O = Output, CS = chip select; OE = output enable
RAM = Random Access Memory = Schreib-/Lesespeicher

59

SRAM und DRAM (Dynamisches RAM)



60

Unterschiede zwischen SRAM und DRAM

- SRAM speichert jedes Bit in einem eigenen Flip Flop
 - DRAM speichert jedes Bit in einem kleinen Kondensator: keine el. Ladung = „0“, Ladung = „1“
 - die Kondensatoren zur Bit-Speicherung verlieren ihre Ladung aus technischen Gründen rasch (nach ca. 10 ms).
 - zur dauerhaften Bitspeicherung muß bei jeder „1“ die Ladung nach ca. 1-2 ms immer wieder nachgefüllt werden = sog. „refresh“
 - wegen der notwendigen refresh-Zyklen spricht man von dynamischem RAM
 - Vorteil von DRAMs gegenüber SRAMs: schlechte Kondensatoren lassen sich viel leichter und viel dichter auf einem Chip integrieren als Flip Flops, die aus mehreren Transistoren bestehen.
 - um den Preis weiter zu reduzieren, spart man durch Multiplexen der Adreßbits Anschlußstifte
 - z.B. ergeben 10 Adressanschlußstifte 20 Adressbits, die zu je 10 Bits in 2 Schritten an die Stifte angelegt werden
- ⇒ Preis von DRAMs ist niedriger und Speicherkapazität höher (ca. 4-fache Kapazität)

62

- Speicher für 512 K x 8-Bit bzw. 4096 K x 1-Bit-Worte
- A_0 - A_{18} = 18 Adressbits zur Wortauswahl im SRAM. A_0 - A_{10} im DRAM stellen gemultiplexte Adressbits dar.
- D_0 - D_7 = Datenein- und ausgänge (bidirektional!) bzw. D alleine
- $\overline{CS}$ = invertierter Chip Select (Bausteinaktivierung)
- $\overline{OE}$ = invertierter Output Enable (Wortausgabe)
- $\overline{WE}$ = invertiertes Write Enable (Schreibaktivierung)

Der Strich bei einem Bezeichner heißt, dass das betreffende Signal invertiert wird, d.h. das Signal wird durch Anlegen einer „0“ aktiviert.

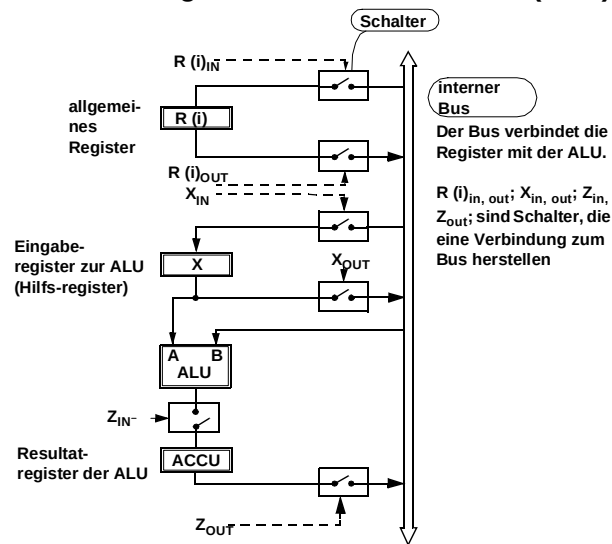
61

Die wichtigsten Register in einer CPU

- 1.) Der Befehlszeiger (PC, program counter)
Er enthält die Hauptspeicheradresse des jeweils auszuführenden Befehls.
- 2.) Das Instruktionsregister (IR, instruction register). Es enthält den jeweils auszuführenden Befehl, nachdem dieser aus dem Hauptspeicher gelesen wurde.
- 3.) Das Speicheradressenregister (MAR, memory address register)
Es enthält die Hauptspeicheradresse der Zelle, die gerade gelesen oder geschrieben wird.
- 4.) Das Speicherdatenregister (MDR, memory data register). Es enthält den Inhalt der Zelle, deren Adresse im MAR steht. (=MAR enthält die Adresse einer Zelle, MDR enthält den Inhalt)
- 5.) Der Akkumulator (ACCU).
Er dient als temporärer Zwischenspeicher der ALU und enthält Resultatwerte der ALU.
- 6.) Das ALU-Eingangsregister.
Es dient ebenfalls als temporärer Zwischenspeicher, um ein Operand für die ALU aufzunehmen. Der 2. ALU-Operand, der z. B. für die Operation $A + B$ nötig ist, kommt entweder von einem 2. ALU-Eingangsregister oder direkt vom ACCU.
- 7.) Allgemeine Rechenregister $R_0, R_1, \dots, R_{n-1}$
- 8.) Spezialregister wie z. B. Stack Pointer (Stapelzeiger)

63

Ein wichtiger Teil eines Prozessors (CPU): die ALU mit Registern



64

Mikroprogrammierung

- Das Leitwerk als EA mit Ein-/Ausgabe ist nur eine Möglichkeit der Realisierung.
- Bei der Generation der CISC-Rechner (1971 - ca. 1990) wurde das Leitwerk anders realisiert: als eigener kleiner Prozessor.
- Dieser Prozessor heißt Mikroprogramm-Sequenzler und stellt einen Prozessor innerhalb der CPU dar.
- Der Sequenzler führt ein Programm aus, das als Mikrogramm bezeichnet wird und das in einem Festwertspeicher innerhalb der CPU gespeichert ist (sog. Mikrogrammspeicher).
- Die einzelnen Befehle des Mikroprogramms werden von dem Mikrobefehlszähler des Sequenzers adressiert.
- Der Trick ist, daß der Sequenzler wesentlich weniger Befehle beherrscht, als der Prozessor, in dem er sitzt. Dadurch ist er einfacher zu realisieren.
- Da das Leitwerk im Falle des Sequenzers programmgesteuert ist, bewirkt jeder einzelne Befehl, den der Prozessor ausführt, die Ausführung eines kompletten Mikroprogramms durch den Sequenzler.

66

Das Leitwerk (Steuerwerk) eines Rechners

- Alle Signale zur Ansteuerung der Schalter $R(i)_{in}$, $R(i)_{out}$, X_{in} , X_{out} , Z_{in} , Z_{out} werden durch eine Steuereinheit (= Leitwerk, Control unit) erzeugt
- Die Steuereinheit gibt an, wann die verschiedenen Schalter geschlossen und wieder geöffnet werden sollen
- Die Steuerimpulse steuern die Schalter im Rechenwerk nacheinander an $\Rightarrow$ Sequenz von Impulsen
- Sie definieren ein Raster aus Zeitpunkten mittels Impulse unterschiedlicher Länge in unterschiedlichem Abstand
- Zusätzlich verarbeitet die Steuerlogik Statusbits aus der ALU und die Bits des Befehlswortes
- Durch gezieltes Öffnen und Schließen der Schalter kann ein Algorithmus ausgeführt werden, z. B. Berechnen des Skalarprodukts
- Ein Steuerwerk kann als EA mit Ein-/ Ausgabe (Moore Automat) implementiert werden
- Die Befehle, aus denen die Software besteht, werden bitweise den Eingängen des EA zugeführt

65

- Macht man den Mikroprogrammspeicher sehr groß (einige KB), kann der Sequenzer komplexe Folgen von Steuersignalen erzeugen.
 - Daraus resultieren komplexe Befehle des Prozessors. Sie geben dieser Prozessorgeneration den Namen: CISC = Complex Instruction Set Computer
 - Das Prinzip des Prozessors im Prozessor kann rekursiv fortgesetzt werden. Dann wird das Leitwerk des Sequenzers nicht hardwaremäßig ausgeführt, sondern als Sequenzer.
 - Man spricht dann von Nanoprogramm-Sequenzler bzw. Nanoprogramm
 - Nachteil der CISC-Prozessoren ist, daß sie langsam sind, weil für jeden Befehl mehrere Mikrobefehle ausgeführt werden müssen und für diese u. U. weitere Nanobefehle.
 - Vorteil ist, daß komplexe Befehle sich mit wenig Hardware-Aufwand in Software implementieren lassen.
- $\Rightarrow$ Prozessor kann hinsichtlich seiner Befehle relativ schnell an neue Erfordernisse angepasst werden (neuer Befehlssatz). Eine Änderung der Hardware ist nicht erforderlich.

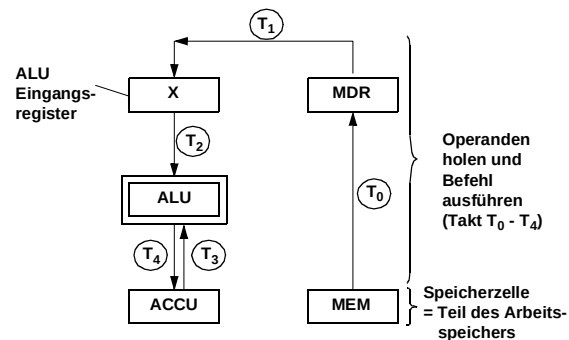
67

Interaktion zwischen den funktionellen Einheiten eines Prozessors

„Wie funktioniert ein Prozessor?“

68

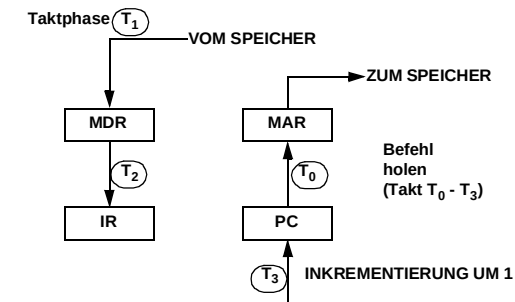
Die Phasen „Operand holen“ und „Befehl ausführen“ (Zusammenspiel ALU, ACCU, MDR)



- Operand holen und Befehl ausführen hat in diesem Beispiel 5 Unterphasen
- wiederum steuert das Leitwerk die Sequenz der Aktionen ($T_0 - T_4$), die für die Phasen „Operand holen“ und „Befehl ausführen“ nötig sind

70

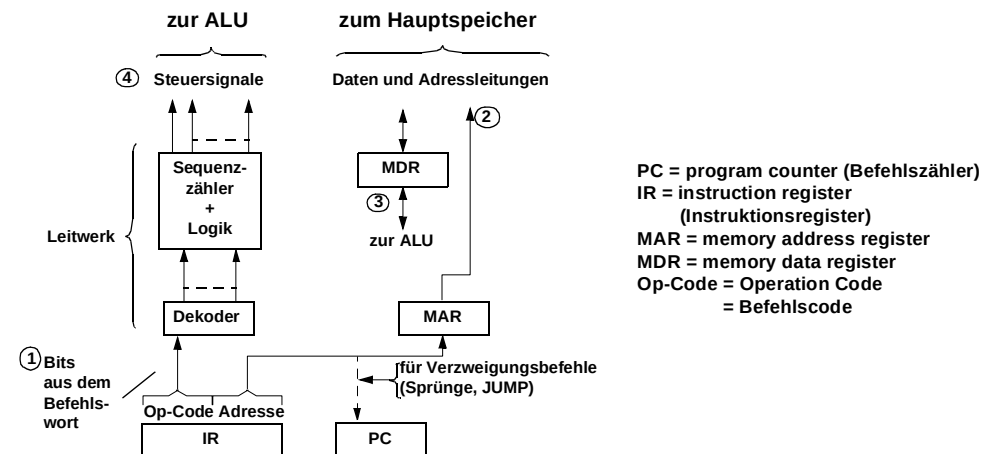
Die Befehlsholphase der CPU (Zusammenspiel PC, MAR, MDR und IR)



- Befehlsholphase hat 4 Unterphasen
- Jede Unterphase muss einzeln in Gang gesetzt werden
- Das Leitwerk erzeugt die Steuerimpulse $T_0 - T_3$
- Es gibt damit den Ablauf innerhalb der Befehlsholphase an
- Dieser Ablauf wird auch Sequenz von Aktionen bezeichnet

69

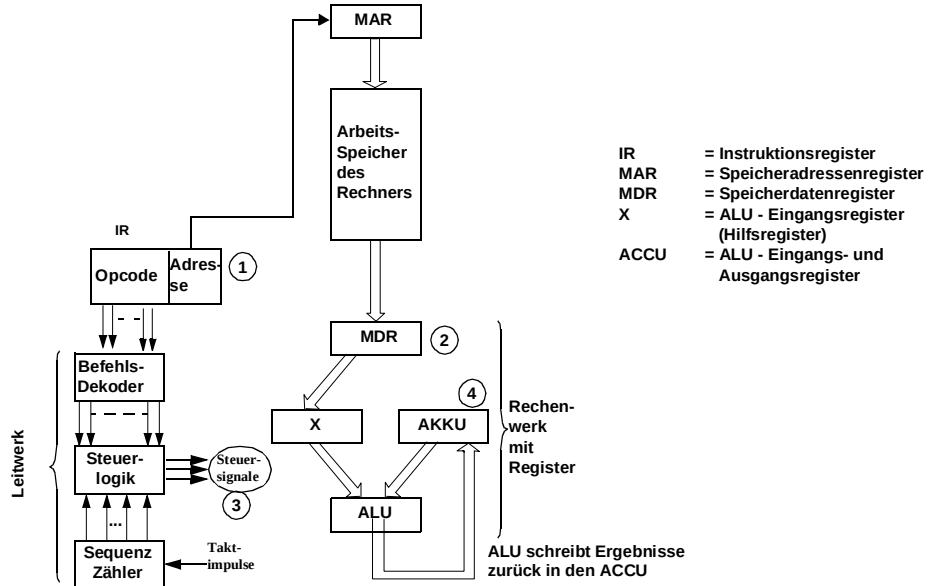
Zusammenspiel von Sequenzzähler mit Logik, PC, IR, MAR und MDR



- Der Inhalt vom IR wird in Befehlscode (op-Code) und in Operandadresse(n) aufgespalten

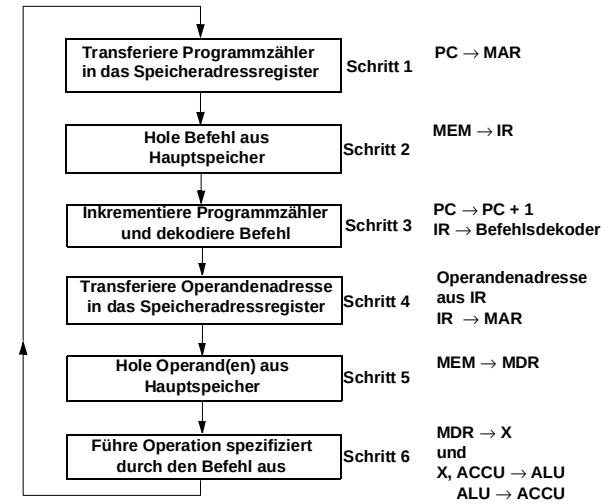
71

Zusammenspiel von Leitwerk, Rechenwerk und Register



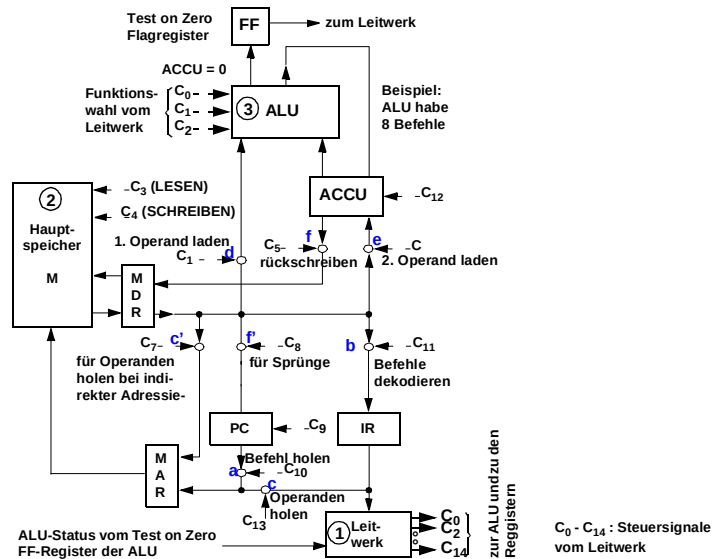
72

Die Hauptschleife in der CPU



73

Zusammenspiel zw. Rechenwerk, Leitwerk, Speicher, Register und Schaltern



Der Befehlssatz eines CISC-Prozessors

- ♦ Grundlegende Befehlsklassen
- ♦ Exemplarischer Befehlssatz
- ♦ Adressierungsarten

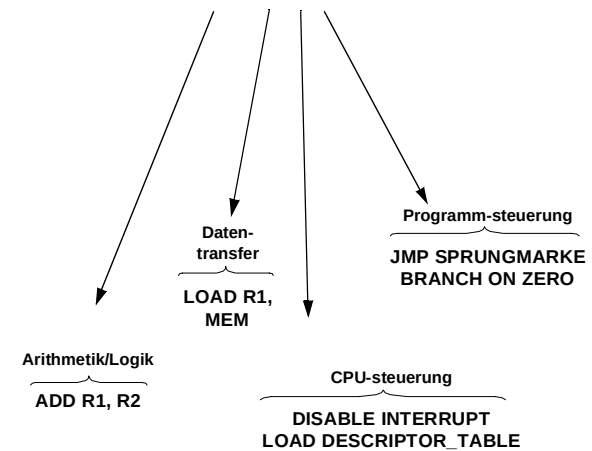
76

Exemplarischer Befehlssatz eines CISC-Prozessors

Typ	Operationsname(n)	Beschreibung
DATEN TRANSFER	MOVE (TRANSFER)	Transferiert Wort oder Block von Quelle nach Ziel
	STORE	Transferiert Wort vom <u>Prozes-</u> <u>sor</u> zum externen Speicher
	LOAD (FETCH)	Transferiert Wort vom exter- nen Speicher <u>zum</u> Prozessor
	EXCHANGE	Vertauscht Inhalt von Quelle und Ziel
	CLEAR (RESET)	Transferiert Null-Wort zum Ziel
	SET	Transferiert Wort aus 1en zum Ziel
	PUSH	Transferiert Wort von der Quelle zur Spitze des Stapel- speichers
	POP (PULL)	Transferiert Wort von der Spit- ze des Stapelspeichers zum Ziel

78

Grundlegende Befehlsklassen



77

ARITHMETIK	ADD	Berechnet die Summe zweier Operanden
	SUBSTRACT	Berechnet die Differenz zweier Operanden
	MULTIPLY	Berechnet das Produkt zweier Operanden
	DIVIDE	Berechnet den Quotient zweier Operanden
	ABSOLUTE	Ersetzt den Operanden durch seinen Absolutwert
	NEGATE	Ändert das Vorzeichen des Operanden
	INCREMENT	Addiert 1 zum Operanden
	DECREMENT	Subtrahiert 1 vom Operanden
LOGISCH	AND OR NOT (COMPLEMENT)	Führt die angegebenen logischen Operationen bitweise aus
	EXCLUSIVE-OR bzw. EQUIVALENCE	Test auf Bitgleichheit

79

	SHIFT	Links- (rechts-) verschieben des Operanden, Einführung von Konstanten am Ende
	ROTATE	Links- (rechts-) shift des Operanden entlang eines geschlossenen Weges
	CONVERT (EDIT)	Ändert Datenformat, z. B. von binär nach dezimal

80

	EXECUTE	Hole Operand von angegebener Stelle und führe ihn als Befehl aus. Beachte, dass der PC nicht modifiziert wird
	SKIP	Inkrementiere PC, um nächsten Befehl zu überspringen
	SKIP CONDITIONAL	Teste bestimmte Bedingung; abhängig von Bedingung, entweder PC inkrementieren oder nichts tun
	TEST	Teste bestimmte Bedingung; setze flag(s) entsprechend
	COMPARE	Mache logische oder arithmetische Vergleiche von zwei oder mehr Operanden; setze flags entsprechend
	SET CONTROL-VARIABLES	Umfangreiche Klasse von Befehlen für Sicherheitskontrollen verschiedene Zwecke, Unterbrechungsbehandlung, Uhrenkontrolle usw. (oft privilegierte Befehle)
	HALT	Stoppe Programmausführung

82

Grundbefehlssatz

PROGRAMM= STEUERUNG (flow control)	JUMP (BRANCH)	Unbedingter Sprung; lädt PC mit bestimmter Adresse
	JUMP CONDITIONAL	Teste bestimmte Bedingung; abhängig von Bedingung, entweder PC mit bestimmter Adresse laden oder nichts tun
	JUMP TO SUBROUTINE = (CALL, BRANCH-AND- LINK)	Speichere aktuelle Programmkontrollinformation (PC, Status Register, usw.) an bestimmte Stelle, z. B. auf den Stapelspeicher; springe zu angegebener Adresse
	RETURN	Ersetze Inhalte des PC, Status- Register, usw., mit Information von entsprechender Stelle, z. B. von der Spitze des Stapelspeichers

81

	WAIT (HOLD)	Stoppe Programmausführung; teste bestimmte Bedingung fortlaufend, wenn Bedingung zutrifft, Ausführung fortsetzen
	NO OPERATION	Keinen Befehl ausführen, Programm wird fortgesetzt
Eingabe Ausgabe	INPUT (READ)	Transferiert Daten von bestimmtem IO-Port oder Gerät an bestimmtes Ziel, z. B. Hauptspeicher oder Prozessor-Register
	OUTPUT (WRITE)	Transferiert Daten von bestimmter Quelle an IO-Port oder Gerät
	START IO	Transferiert Befehle an IOP um IO-Operation zu starten
	TEST IO	Transferiert Status Information von IO-System an bestimmtes Ziel
	HALT IO	Transferiert Befehle zur IOP um IO-Operation zu beenden

83

Adressierungsarten

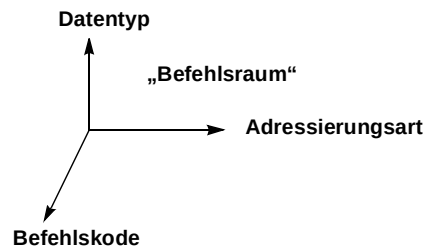
- Adressierungsarten sollen für den Compiler und für Assembler-Programmierer den Zugriff auf Daten und strukturierte Datentypen vereinfachen
- Beispiele eines strukturierten Datentyps: 2D-Array

Vorkommen	Art	Anwendung	Beispiel	Wirkung
RISC + CISC	Immediate	(einmalige) Konstante	ADD R1,#4	$R1 \leftarrow R1 + 4$
RISC + CISC	Direkt (<u>Absolut</u>)	statische Daten	ADD R1,(100)	$R1 \leftarrow R1 + \langle 100 \rangle$
RISC + CISC	Register	Wert ist im Register	ADD R1,R2	$R1 \leftarrow R1 + R2$
RISC + CISC	Register- <u>Indirekt</u>	Registerwert ist Pointer	ADD R1,(R2)	$R1 \leftarrow R1 + \langle R2 \rangle$
RISC + CISC	Displacement (<u>Basised</u>)	lokale Var. stat. Feldelement.	ADD R1,10(R2)	$R1 \leftarrow R1 + \langle R2 + 10 \rangle$
CISC	Basis- <u>Indiziert</u>	dyn. bestimmte Feldelemente	ADD R1,(R2)(R3)	$R1 \leftarrow R1 + \langle R2 + R3 \rangle$

84

Orthogonaler Befehlssatz

- Ein Befehlssatz heißt orthogonal, wenn Befehlscode, Adressierungsarten und Datentypen (und Anzahl der Operanden) unabhängig miteinander kombiniert werden können.



- Orthogonalität soll die Befehlssatznutzung vereinfachen
- Orthogonalität führt schnell zu riesigen Befehlssätzen, wenn es viele Adressierungsarten und Datentypen gibt. (>1000 Befehle)

86

CISC	Auto-Inkr./ <u>Dekrement</u>	Feldzugriff in Loops	ADD R1,(R2+/-)	$R1 \leftarrow R1 + \langle R2 \rangle$ $R2 \leftarrow R2 \pm \text{Elementlänge}$
CISC	Speicher- <u>Indirekt</u>	Speicherplatz ist Pointer	ADD R1,@(R2)	$R1 \leftarrow R1 + \langle \langle R2 \rangle \rangle$
CISC	Skaliert	Feldzugriff mit Datentyp der Länge d	ADD R1,100(R2)[R3]	$R1 \leftarrow R1 + \langle 100 + R2 + d \cdot R3 \rangle$

<z>: Inhalt des Speicherplatzes, der durch z gegeben ist.

85

Grundlegende Architekturen bei CISC-Prozessoren

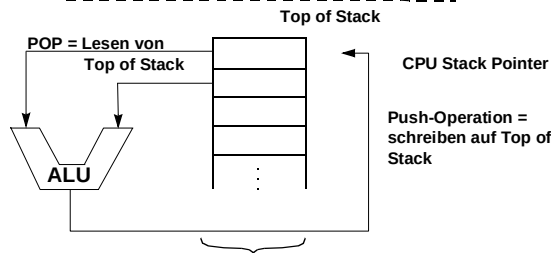
- Stack-Architektur
- Akkumulator-Architektur
- Register-Architektur
- Speicher-Architektur

87

Stack-Architektur

- Verknüpfungsoperationen benötigen keine Adressangaben (Nulladreibmaschine) für ihre Operanden.
- Quell- und Ergebnisoperanden befinden sich auf fest vereinbarten - impliziten - Plätzen an der Spitze von einem (Register-) Operanden-Stack.

normaler Arbeitsspeicher,
falls Stack CPU extern ist



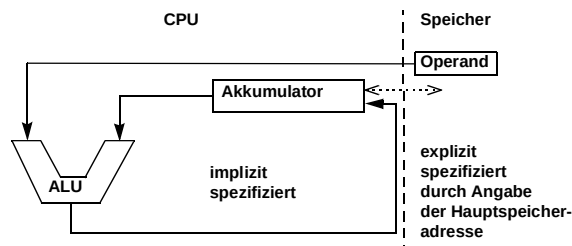
Der Operandenstack ist ein stack-artig organisiertes Registerfile, z. B. aus 3 Registern, oder ein im Speicherorganisierter Operandenstack. D. h. der Stack ist entweder CPU-intern oder extern im Speicher.

Registerfile: mehrere Register am Stück z. B. $R_0 - R_{15}$

88

Akkumulator-Architekturen

- Bei Verknüpfungsoperationen enthält das Akkumulator-Register implizit einen Quelloperanden und letztlich auch den Ergebnisoperanden.
- Außer dem Akkumulator-Register gibt es i. d. R. kein weiteres (für den Programmierer zugängliches) Register.
- Der zweite Operand liegt typisch im Hauptspeicher und muß explizit adressiert werden. (Ein-adress-Maschine)



90

Bewertung der Stackarchitektur

- Stackarchitekturen unterstützen die Ausführung gewisser Anweisungen in höheren Programmiersprachen, z.B.:
 - die Abarbeitung arithmetischer Ausdrücke (umgekehrte polnische Notation)
 - die Kontrolle des Gültigkeitsbereiches von Variablenamen in Verbindung mit einer Blockstruktur der Sprache = „Scope“
 - den Prozedurmechanismus (Unterprogramme) und
 - die Parameterübergabe. (Parameter kommen auf den Stack)
- Stackarchitekturen können speicherplatzsparenden Code ergeben, da bei Stackoperationen keine Adressangabe nötig ist (Nulladreibmaschine).
- Flexibilität und Eignung für allgemeine Datenverarbeitungsaufgaben werden kritisch beurteilt:
 - kein wahlfreier Zugriff auf beliebige Stackelemente
 - Stack(top) stellt Flaschenhals dar (Pipelining)
- Stackarchitekturen stellen keine Alternative zu Rechnern mit Registerstruktur (Universalregister-Maschinen) dar.
- Populäres Beispiel einer Stack-Architektur: Java Virtual Maschine (JVM)

89

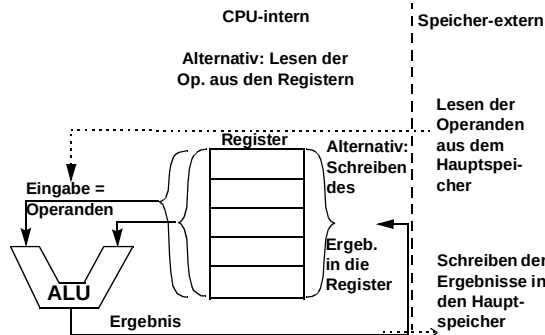
Bewertung der Akkumulator-Architektur

- Minimaler Aufwand, da nur ein internes Register, der Akkumulator, benötigt wird.
- Höherer Speicherverkehr, da der Akkumulator der einzige interne temporäre Speicher ist (vom und zum Speicher).
- Der interne Maschinenzustand (sog. Kontext) ist minimal.
- Die ersten 8-Bit-Mikroprozessoren hatten eine Akkumulator-Architektur.

91

(Universal-)Register-Architektur

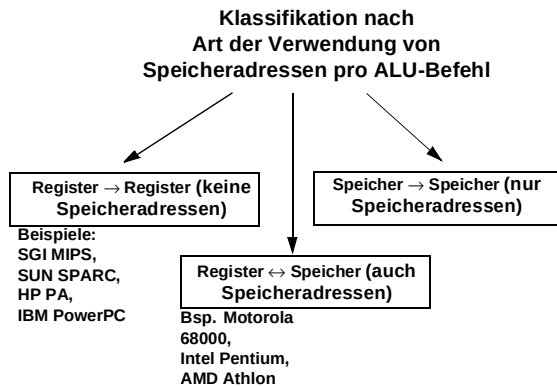
- Ein Satz gleichberechtigter Register dient für Variablen bzw. Operanden (Register-File).
- Im Op-Code sind typisch mehrere Operanden explizit anzugeben (Zweiadressmaschine, 3-Adressmaschine usw.).



- Operanden werden durch explizite Angabe ihrer Adressen aus dem Hauptspeicher in die ALU kopiert oder aber aus den Register geholt.
- Ergebnis wird entweder in ein Register zurückgeschrieben oder durch Angabe einer weiteren Adresse in den Arbeitsspeicher zurücktransportiert.

92

Verallgemeinerte Register-Architekturen



- Registerarchitekturen gibt es sowohl bei CISC - als auch bei RISC - Rechnern.
- RISC - Rechner = Rechner der heutigen Generation (RISC = Reduced Instruction Set Computer)
- CISC-Rechner haben ALU-Operationen sowohl auf Register als auch Speicheradressen.

94

Bewertung der Register-Architektur

Vorteile:

- Der Hauptvorteil liegt in der freien und effektiven Anwendbarkeit der Register durch den Compiler, vorausgesetzt, es gibt keine *impliziten Registerfestlegungen*.
- Ausdrucksberechnungen können in beliebiger Reihenfolge erfolgen, was Pipelining unterstützt.
- Das Halten von Variablen in Registern verringert den Speicherverkehr, beschleunigt die Ausführung, verbessert die Code-Dichte und ermöglicht den effektiven Einsatz von sog. Superskalartechniken und von Pipelining.

Nachteile:

- Bei Kontextwechsel z.B. aufgrund von Interrupt oder Time Sharing sind große Registersätze auszutauschen.
- Alle Operanden müssen adressiert werden, was zu längeren Befehlen führt.

93

Register-Speicher-Typ

- In Verknüpfungsbefehlen können auch Speicheroperanden adressiert werden.
- Vorteil:
 - Alle Daten können referenziert werden, ohne sie vorher in ein Register laden zu müssen.
- Nachteil:
 - Befehlslänge und die Taktzyklen pro Befehl variieren in Abhängigkeit von der Anzahl der übergebenen Speicheroperanden (schlecht für Pipelining).
 - ⇒ wir nur bei CISC-Rechnern verwendet

95

Speicher-Speicher-Typ

Vorteile:

- Im Extremfall gibt es nur Speicheroperanden.
- ⇒ Die Programmierung wird transparenter, da nicht zwischen Register- und Speicheroperanden unterschieden werden muss.
 - CISC-Intention

Nachteile:

- Hoher Speicherverkehr
- Variierende Befehlslänge und Befehlstaktzahl, wenn neben dem Nur-Speicheroperanden-Modell der Befehlssatz aus Flexibilitätsgründen auch Registeroperanden zulässt. (siehe: Orthogonaler Befehlssatz, CISC-Befehlssatz.)
 - ⇒ wird nur bei CISC verwendet

96

Heutiger Status der CISC-Prozessoren

- Heutzutage werden Prozessoren mit CISC-Architektur nicht mehr hergestellt.
- Trotzdem gibt es sehr viele Prozessoren mit typischem CISC-Befehlssatz (alle aus der Pentium-Familie, bzw. kompatibel dazu).
- Pentium-Prozessoren weisen intern eine RISC-ähnliche Architektur auf.
- Eine spez. Einheit im Pentium, der Microopsequenzer sorgt für die Abbildung des CISC-Befehlssatzes auf die RISC-Befehle des Prozessorkerns.
- Der Pentium-RISC-Kern beruht auf typischen RISC-Merkmalen wie „Superpipelining“, „Superakalarität“ und „out-of-order execution“.

98

Register-Register-Typ

- Verknüpfungsoperationen der ALU verwenden ausschliesslich Register (-adressen) für Quell- und Zieloperanden
- Nur in Lade- und Speicherbefehlen sind Speicheradressangaben zu finden
- Der Speicherverkehr erfolgt nur mittels Load- und Store-Befehlen vom Speicher in die Register und umgekehrt
 - Load / Store-Architektur

Vorteil:

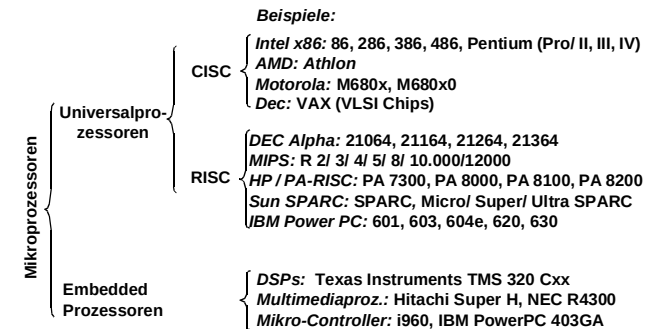
- Ein Verknüpfungsbefehl tritt nicht „mal mit, mal ohne“ Speicheroperand auf
 - einfache Befehlskodierungen fester Länge.
 daraus folgt:
 - einheitliche Taktzeiten je Befehl
 - Pipelining-Unterstützung

Nachteil:

- Alle Speichertransfers sind durch einen separaten Befehl auszuweisen
 - Code-Expansion (Programm wird größer)
- Status: wird exklusiv von RISC-Rechnern eingesetzt

97

Prozessorfamilien



DSP = digitaler Signalprozessor; DSPs sind in jedem Handy und in jedem Modem,
Multimedia-Prozessoren sind in jeder Spieleconsole,
Mikrocontroller sind in jeder Automobil-Motorsteuerung eingebaut (i.e. „eingebettet“)

99

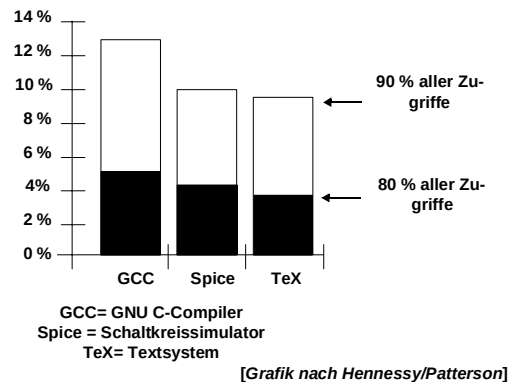
Caches

- Speicherhierarchie
- Lokalitätsprinzip
- Depot von Daten/Befehlen
- Aufbau und Arbeitsweise
- Ersetzungsstrategien
- Das Konsistenzproblem

100

Befehlslokalität

- Prozentsatz von Befehlen, der für 80 % bzw. 90 % der abgearbeiteten Befehle verantwortlich ist.



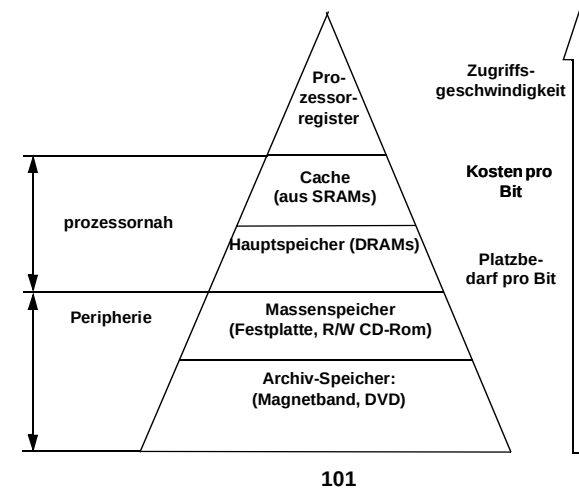
- Faustregel:
Ein Programm verbraucht 90 % seiner Zeit für 10 % des Codes!

102

Speicherhierarchie

- Ein hierarchischer Speicheraufbau ist ein guter Kompromiss bezüglich Zugriffszeit und Kosten pro Bit

⇒ Caches sind ein wichtiger Bestandteil der Hierarchie



101

Programmeigenschaft „Lokalität“

- Programme tendieren zur wiederholten Nutzung von Befehlen und Daten
- Es gibt Befehlslokalität und Datenlokalität
- Für die Lokalität von Befehlen und Daten gibt es zwei Unterfälle:
 - 1.) Zeitliche Lokalität: Auf ein gerade zugriffenes Speicherwort wird häufig auch in naher Zukunft zugriffen werden. (wegen Schleifen)
 - 2.) Räumliche Lokalität: Auf Speicherworte deren Adressen benachbart sind, wird häufig auch in naher Zukunft zugriffen. (wg. linearer Codeabarbeitung)
- Datenzugriffe zeigen eine geringere Lokalität als Befehlszugriffe
- ⇒ Aufgrund der Lokalitätseigenschaft macht es Sinn, ein kleines Depot für Befehle und/oder Daten einzurichten
- ⇒ Ein benötigter Befehl/Datum steht aufgrund des Lokalitätsprinzips oft im Depot und braucht nicht aus dem langsamen Hauptspeicher geholt werden
- Der Cache ist ein Depot für Befehle und Daten, auf das die CPU schnell zugreifen kann [Cache (engl.) = Depot] Da der Cache nur Kopien enthält, muss es ständig aktualisiert werden

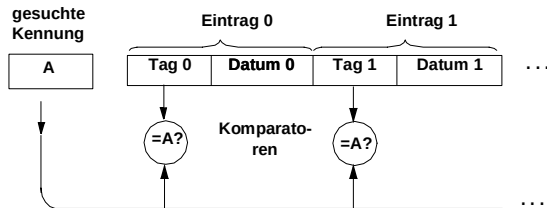
103

Aufbau und Arbeitsweise von Caches

Prinzip: Es werden beim Cache nicht nur Kopien von Hauptspeicherworten gespeichert, sondern Paare von Daten der Art (Kennung, Hauptspeicherwort)

Vollassoziativer Cache

- Ein vollassoziativer Cache enthält eine Folge von Einträgen
- Jeder der n Einträge besteht aus einem „Tag“-Feld (= Kennung) und einem Datenfeld (Tag = Etikett)
- Das Tagfeld enthält die Kennung eines gegebenen Eintrages, das Datenfeld die entsprechenden Daten

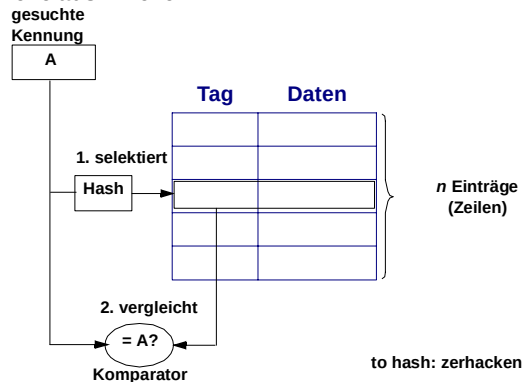


- Als Kennung wird bei Rechner-Caches immer direkt oder indirekt die Hauptspeicheradresse des Original-Speicherwortes verwendet, das als Kopie im Cache ist

104

Direct-Mapped Cache

- Der direct-mapped Cache ist eine Vereinfachung des vollassoziativen Cache
- Jeder der n Einträge besteht aus einem Tag-Feld und einem Datenfeld, genau wie beim vollassoziativen Cache
- Eine Hash-Funktion bildet einen Index i aus einem Eingabewert (= Adresse A). Der Index i selektiert eine Cache-Zeile aus n Zeilen



106

⇒ A = Hauptspeicheradresse des Datums aus dem Hauptspeicher

- Die gesuchte Kennung wird simultan von allen Komparatoren mit den Tags verglichen
- Ist ein Tag identisch mit der gesuchten Kennung, wird das dazugehörige Datum ausgegeben

Vollassoziativer Cache - Arbeitsweise

- Bei einem lesenden Cachezugriff wird die Adresse A mit allen n Tags gleichzeitig verglichen.
- Stimmt ein Tag überein (*Hit*), so wird das betreffende Datenfeld verwendet.
- Anderenfalls, bei einem *Miss*, muss geeignet reagiert werden, z. B. durch Löschung eines Eintrages und Einspeicherung des gewünschten Datums mit der Hauptspeicheradresse A in dem Cache.
- Vorteil:
Solange der Cache nicht voll ist, treten beim Schreiben von Einträgen in den Cache keine Probleme auf, da jeder Eintrag an beliebiger Stelle im Cache stehen kann ⇒ keine Konflikte
- Nachteil:
Dieser Cachetyp ist sehr aufwendig, da n Komparatoren nötig sind

105

Direct-Mapped Cache - Arbeitsweise

- Die Adresse A wird mit dem Tag verglichen, welches in der selektierten Zeile gespeichert ist
- Stimmen Tag und Adresse überein, haben wir einen Hit und das Datenfeld kann verwendet werden
- Anderenfalls, bei einem Miss, muss wiederum geeignet reagiert werden: durch Ersetzen der selektierten Zeile durch das Hauptspeicherdatum, das die Adresse A hat
- Es werden die hinteren Bits von A als Hash-Funktion verwendet
⇒ Die vorderen Bits von A werden im Tagfeld anstelle der vollständigen Adresse gespeichert, d. h. die Bits von A werden geteilt (hashing)
- Vorteil: Dieser Cache ist aufwandsarm. Bei gegebener Chipfläche gestattet er die größte Cache-Kapazität, da nur 1 Komparator nötig
- Nachteil: Beim Einspeichern in den Cache können Schreibkonflikte entstehen, da mehrere Einträge auf ein und denselben Index (Zeile) abgebildet werden
⇒ Nichteindeutigkeit der Hash-Funktion, da modulo-Funktion zum Hashing verwendet wird
- Das Abschneiden der vorderen Bits von A durch das Hashing ist eine Modulo-Funktion

107

Beispiel für hash-Funktion zur Abbildung von Hauptspeicheradressen auf Cache-Adressen bei Direct-mapped Caches

1.) Ein Cache habe Platz für 128 Einträge 0,1, ..., 127 und speichere pro Eintrag jeweils ein Hauptspeicherwort
⇒

- Das Datum aus Hauptspeicheradresse 0 oder 128 oder 256 oder 512 oder ... wird in die Cache-Adresse 0 gespeichert
- Das Datum aus Hauptspeicheradresse 1 oder 129 oder 257 oder 513 oder ... wird in die Cache-Adresse 1 gespeichert
- Das Datum aus HS-Adresse 127, 255, 511, ... wird bei 127 gespeichert

$$\Rightarrow \begin{cases} \text{Cache-Adresse} = \text{Hauptspeicheradresse} \text{ MOD Zahl der Cache-Einträge} \\ \text{Cache-Kennung} = \text{Hauptspeicheradresse} \text{ DIV Zahl der Cache-Einträge} \end{cases}$$

(DIV = Ganzzahldivision, MOD = Rest nach Division)

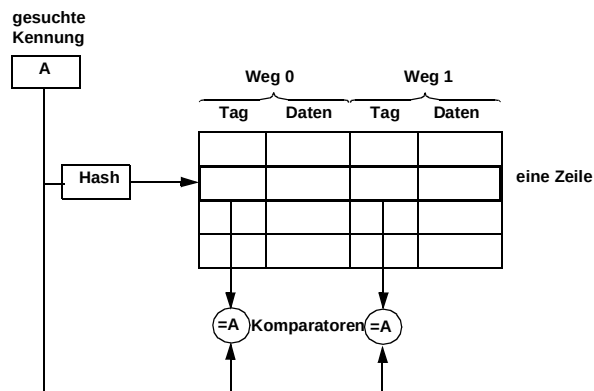
108

m-Wege Mengenassoziativer Cache

- Ein m -Wege mengenassoziativer Cache besteht im Prinzip aus m kleinen Direct-mapped Caches

- Er ist eine Kreuzung aus vollassoziativem und direktabbildendem Cache

Beispiel: 2-Wege Mengenassoziativer Cache



110

2.) Ein Cache habe n Einträge und speichere pro Eintrag jeweils k Hauptspeicherworte

⇒ Es werden jeweils ganze Blöcke aus k Worten zwischen Hauptspeicher und Cache transportiert.

⇒ der Hauptspeicher bekommt gedanklich eine blockweise Einteilung in Blockadressen

⇒ Blockadresse = Hauptspeicheradresse DIV k

$$\Rightarrow \begin{cases} \text{Cache-Adresse} = \text{Blockadresse} \text{ MOD } n \\ \text{Cache-Kennung} = \text{Blockadresse} \text{ DIV } n \end{cases}$$

= die hinteren Bits
= die vorderen Bits der Blockadresse

109

Mengenassoziativer Cache - Arbeitsweise

- Die Hash-Funktion selektiert eine Menge von m Einträgen gleichzeitig ⇒ m Möglichkeiten, das gesuchte Datum zu finden

- Alle m Tags werden gleichzeitig mit der Adresse A verglichen

- Bei n Zeilen und m Wegen enthält der Cache $n * m$ Einträge

- Es gibt m Wege, um ein Datum mit Hauptspeicheradresse A in eine Zeile einzuspeichern, weshalb man auch vom m -fach Mengenassoziativen Cache spricht

- Vorteil:

Die m -fache Einspeichermöglichkeit reduziert die Wahrscheinlichkeit für einen Speicherkonflikt auf $\frac{1}{m}$ im Vergleich zu einem Direct-Mapped Cache

- Nachteil:

Im Vergleich zum Direct-Mapped Cache benötigt er mehr Chipfläche, da m Komparatoren nötig sind

111

Wann und wie werden Daten zwischen Hauptspeicher und Cache transferiert?

- Mechanismen für den Transfer zwischen Cache und Hauptspeicher
 - Der Hauptspeicher wird logisch in Blöcke unterteilt
 - Der Cache wird in eine kleine Menge von Blöcken eingeteilt, die die gleiche Größe wie die Hauptspeicherblöcke haben
 - Ein Cache-Block ist entweder leer oder enthält eine Kopie eines Hauptspeicherblocks
 - Der Cache enthält ein Verzeichnis der aktuell im Cache enthaltenen Hauptspeicherblöcke in Form der Kennungen (Tags)
- Wenn ein Hauptspeicherwort von der CPU gelesen werden soll:
 - wird aus der Adresse die Hauptspeicher-Blocknummer ermittelt
 - nach dieser Blocknummer im Cache-Verzeichnis nachgesehen (= Vergleich mit dem Tag/den Tags)
 - wenn eine Kopie des Blockes gerade im Cache ist, werden die Daten aus dem entsprechenden Cache-Block ermittelt: wir haben einen Hit
 - sonst: wir haben ein Miss: eine Kopie des Blockes wird vom Hauptspeicher in einen Cache-Block transferiert (in einen Cache-Block passt genau ein Hauptspeicherblock)

112

Ersetzungsstrategien bei vollassoziativem und mengenassoziativem Cache

- Was tun, wenn der Cache voll ist? Welcher Cache-Eintrag soll überschrieben werden?
 - ⇒ „Ersetzungsstrategien“ für die Wahl des richtigen Cache-Blockes
 - Beim Direct-mapped Cache stellt sich diese Frage nicht, da die hash-Funktion keine Wahlmöglichkeit für die Einspeicherung von Hauptspeicherblöcken in Cache-Blöcke hat
 - Beim mengenassoziativen- und beim vollassoziativen Cache hingegen gibt es Freiheitsgrade für die „Blockersetzung“
- Die Ersetzungsstrategie entscheidet, wohin die Kopie eines Blockes innerhalb des Caches gehen soll
- Zufällige Ersetzung: irgendeiner der möglichen Cache-Blöcke wird durch den neuen Block ersetzt
 - LRU (least recently used)-Ersetzung:
 - Idee: wenn ein Cache-Block eine Weile nicht mehr benutzt worden ist, wird er wohl auch nicht mehr bald benutzt werden

114

Problemfall:

Es gibt einen Cache-Miss bei lesenden CPU-Zugriff, und der entsprechende Eintrag im Cache ist nicht frei zur Einspeicherung des gewünschten Hauptspeicherblocks.
Was ist zu tun?

- Wenn der Cache-Block, der den gesuchten Hauptspeicherblock aufnehmen soll, nicht leer ist, muss er überschrieben werden
- Dabei sind 2 Fälle zu unterscheiden:
 - 1.) wenn die Daten im zu verwendenden Cache-Block nicht modifiziert wurden, brauchen sie nicht weiter betrachtet zu werden und können überschrieben werden
 - 2.) sonst muss eine Kopie in den Hauptspeicher vorher zurückgeschrieben werden, bevor überschrieben wird, sonst sind die Daten verloren

113

- Für kleine Caches ist LRU besser, aber teuer: normalerweise wird eine Näherungslösung anstelle von LRU in den Cache eingebaut
- Für Caches, die groß genug sind, gibt es kaum einen Leistungsunterschied zwischen zufälliger Ersetzung und LRU, so dass die erstere gewählt wird, da einfacher zu implementieren

Das Konsistenzproblem zwischen Cache und Hauptspeicher

- Angenommen, die CPU möchte eine Speicherzelle beschreiben, die sich in einem Block befindet, der zuvor als Kopie in den Cache geladen wurde
- Wohin soll die Information geschrieben werden?
 - a) in den Cache b) in den Hauptspeicher c) in beides
- In den Fällen a) und b) gibt es zwei Blöcke, deren Inhalt nicht mehr deckungsgleich ist (= Konsistenzproblem)
- Man spricht in dem Fall, dass zunächst nur der Block im Cache modifiziert wird (= Fall a) von einem „dirty block“, da er inkonsistent mit dem Originalblock im Hauptspeicher ist

115

Schreibstrategien in Cache und Hauptspeicher

- Wenn der Hauptspeicherblock, in den geschrieben werden soll, als Kopie in Cache vorliegt, gibt es 2 Möglichkeiten:
 - 1.) **Write through:** der Hauptspeicher wird jedesmal wenn der Cache modifiziert wird, ebenfalls aktualisiert
 - Um die langsamen Hauptspeicherzugriffe zu verringern, genügt es, in einen schnellen „Schreibpuffer“ zu schreiben, der anschließend langsam in den Hauptspeicher übertragen wird
 - 2.) **Write back:** Daten werden nur in den Cache geschrieben; der Hauptspeicher wird nur aktualisiert, wenn der Inhalt eines „dirty“ Blocks aufgrund einer Ersetzungsstrategie überschrieben werden muss
- Wenn der Hauptspeicherblock, in den geschrieben werden soll, nicht als Kopie im Cache vorliegt, gibt es auch 2 Möglichkeiten:
 - 1.) **Fetch on write:**
Der Block wird zuerst in den Cache kopiert und dann dort modifiziert. Dies wird typischerweise eingesetzt, wenn auch die write-back-Strategie gewählt wurde
 - 2.) **Write around:**
Die Daten werden direkt in den Hauptspeicher geschrieben; der entsprechende Block wird nicht in den Cache kopiert

116

Hauptspeicher und MMU

MMU = Memory Management Unit
= Speicherverwaltungseinheit

118

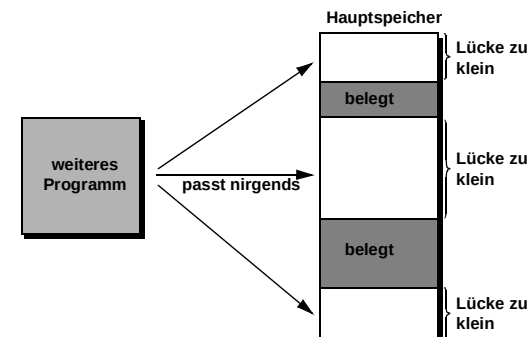
Split Caches (Teilung des Cache)

- Fast alle Prozessoren teilen den Cache auf in:
 - Daten-Cache
 - Befehls-Cache
- Vorteile der Aufteilung des Caches:
 - „verdoppelt“ die Bandbreite der CPU
 - Gelegenheit, jeden Cache unter Berücksichtigung der unterschiedlichen Charakteristiken des Daten- und Befehlsflusses zu optimieren
 - * Befehle müssen nur gelesen werden
 - * Befehle haben eine bessere örtliche und zeitliche Lokalität als Daten
 - ⇒ Befehlscache ist anders organisiert als Datencache bzgl. Ersetzungsstrategien, Schreibstrategien, etc.
- Nachteil der Cache-Aufteilung
 - Keine Lastverteilung der Caches
 - Keine Konsistenz zwischen den beiden Caches (wichtig bei selbst-modifizierendem Code)

117

Problem der nicht zusammenhängenden Adressräume

- Im Betrieb entstehen Lücken im Speicher, die zunächst nicht genutzt werden können



- Ein Verschieben der Speicherbereiche ist zu aufwendig
- Lösung des Problems: Speicheraufteilung in gleichlange Abschnitte (Seiten, pages)

119

- Der Programmadressraum wird in kleinere Stücke von jeweils gleicher Größe (Seiten) zerlegt, die unabhängig voneinander im Speicher abgelegt werden
- Für die Zerlegung müssen die Prozessoradressen von den Hauptspeicheradressen getrennt werden

⇒ Man spricht von dynamischer Adressumsetzung

- Eigenschaften der dynamischen Adressumsetzung:

- Geringerer Verschnitt (Positiv)
 - Höherer Aufwand beim Adressieren (Negativ)
 - Die Adressen im Programm („logische Adressen“) werden durch eine spezielle Einrichtung des Prozessors in „physikalische Adressen“ umgesetzt.
 - Die Einrichtung heisst Speicherabbildungseinheit, (*Memory Management Unit, MMU*)
 - Die Unterscheidung zwischen Prozessoradressen und Adressen, wie sie der Hauptspeicher versteht, hat sich bei praktisch allen Prozessoren durchgesetzt, auf denen ein Plattenbetriebssystem ausgeführt werden kann
- ⇒ Dynamische Adressumsetzung durch eine MMU ist Standard

120

Erweiterung der dynamischen Adressumsetzung zum virtuellen Speicher

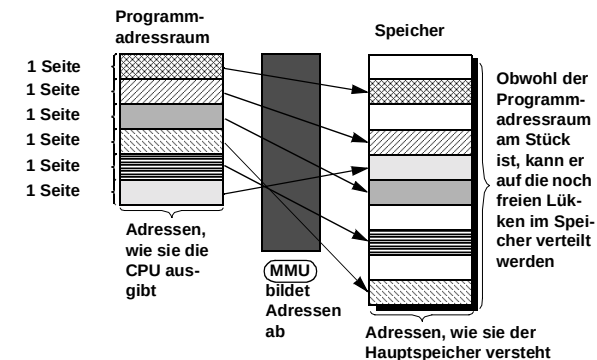
- Aufgrund der Lokaltätseigenschaft von Programmen reicht es aus, nur den Teil des Programmadressraums im Hauptspeicher zu halten, der gerade benötigt wird
- Der große Rest des Programms verbleibt auf der Festplatte in einer speziellen Datei, der Auslagerungsdatei (pagefile.sys) oder einer Partion (swap partition)

Virtueller Hauptspeicher

- Das Programm und der Hauptspeicher werden in gleichlange Stücke (Seiten bzw. pages) eingeteilt
- Wird auf eine Seite zugegriffen, die nicht im Hauptspeicher ist, sendet die MMU ein Signal zum Betriebssystem, die „page fault exception“ (Seitenfehlen)
- Anschliessend wird der Prozess, dem die Seite fehlt, „descheduled“
- Das Betriebssystem lädt die fehlende Seite von der Festplatte nach (= „paging“ oder „swapping“)
- Sobald die Seite im Hauptspeicher ist, wird der Prozess „rescheduled“

122

Adressabbildung durch die MMU



121

- Ist im Speicher kein Platz frei, wird eine andere Seite auf den Plattenspeicher ausgelagert
- Es existieren für die Verdrängung einer Seite dieselben Ersetzungsstrategien wie bei den Caches

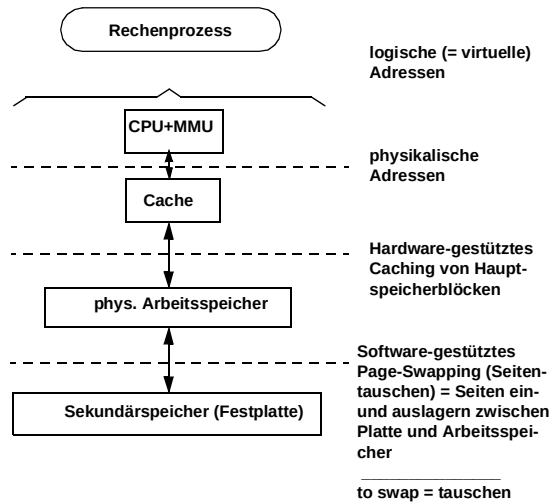
Vorteile des virtuellen Hauptspeichers

- 1.) Es können grosse Programme ausgeführt werden, die mehr Platz benötigen, als vorhanden ist
 - 2.) Es können im timesharing mehr Programme ausgeführt werden als in den Hauptspeicher passen
- Die MMU erweitert den realen Hauptspeicher um die Grösse der Auslagerungsdatei
 - Die Grösse der Auslagerungsdatei wird nur durch den zur Verfügung stehenden Plattenplatz begrenzt
 - Beides zusammen, pages im Hauptspeicher und auf der Festplatte, wird als virtueller Speicher bezeichnet, da der reale Hauptspeicher grösser erscheint als er ist
 - Praktischerweise entspricht die Seitengröße des virtuellen Speichers der Grösse, in der Daten zwischen Plattenspeicher und Hauptspeicher transportiert werden (z. B. 4 KByte)

⇒ Grösse einer Seite im Hauptspeicher = Grösse eines Records auf der Platte

123

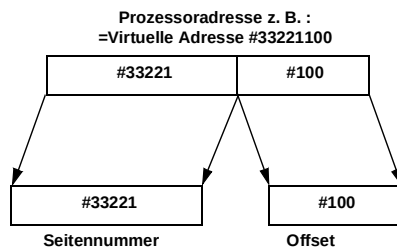
„Logische“ Prozeßadressen aus der Sicht der CPU



124

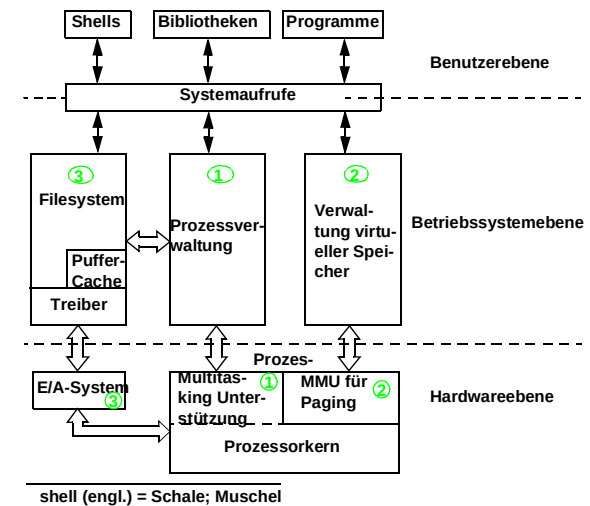
Adressumsetzung

- Eine virtuelle Prozessoradresse wird durch eine Abbildungsfunktion - einer Mapping-Funktion - auf eine physikalische Adresse abgebildet. (Mittels sog. MMU, Seitentabellen und dem Betriebssystem)
- Eine beliebige Adresse kann nicht beliebig abgebildet werden. Nur ganze Blöcke sind frei abbildbar. (= "Seite, page")
- Typische Seitengrößen sind 4 bzw. 8 KB.
- Eine virtuelle Adresse besteht aus 2 Komponenten: einer *virtuellen Seitennummer* und dem *Offset (Versatz)*



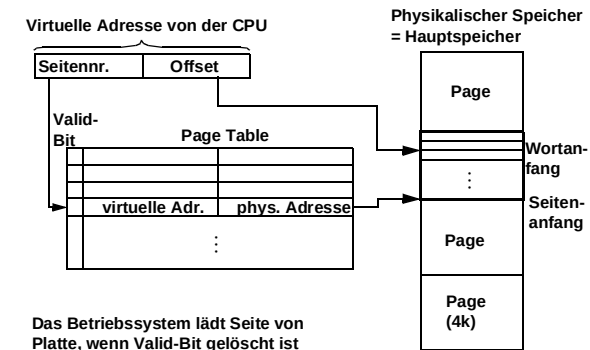
126

Einordnung der MMU in die Prozessor-Hardware und in die Betriebssystem-Software



125

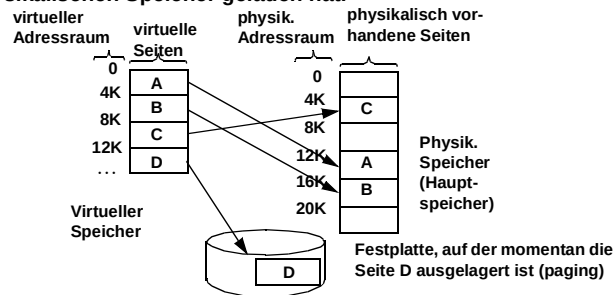
- In eine physikalische Adresse ist nur die virtuelle Seitennummer umzuwandeln. Der Offset bleibt unverändert.
- Der virtuellen Seitennummer muss zur Laufzeit eine physikalische Seitennummer zugeordnet werden.
- Die Zuordnung wird einer Seitentabelle (Map/Pagetabelle) entnommen. (= Listenstruktur im Hauptspeicher). Dazu liest ein Programm des Betriebssystems die Tabelle.



127

Virtueller Adressraum

- Die Adressumsetzung erlaubt es, dass der Speicher, den die CPU adressiert, größer als der physikalische sein kann (= virtueller Adressraum)
- Gibt es zu einer virtuellen Seite momentan keine zugehörige physikalische Seite, wird diese durch ein zusätzliches Flag - das *valid bit* - gekennzeichnet (valid = FALSE)
- Werden fehlende Seiten angesprochen, muss der „Zugriff“ warten bis das Betriebssystem die Seite in den physikalischen Speicher geladen hat.

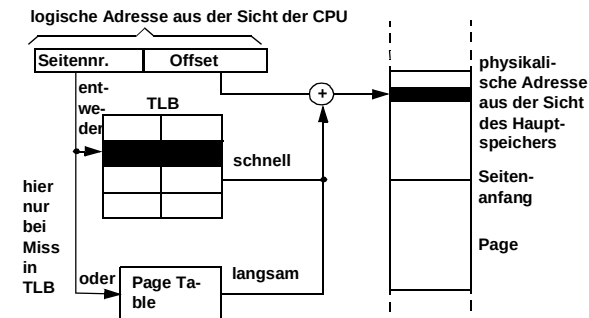


- Die MMU leistet Unterstützung bei der Adressabbildung

128

Translation Lookaside Buffer

- Hauptteil der MMU ist der Translation Lookaside Buffer (TLB), ein vollassoziativer Cache
- Die Seitentabelle wird vom Betriebssystem im Hauptspeicher und auf der Festplatte verwaltet
- Um schnelle Abbildungen zu ermöglichen, wird die Seitentabelle in Teilen in einem kleinen, schnelleren, auf der CPU befindlichen Zwischenspeicher - dem TLB - als Kopie gehalten
- Nur bei Cache Miss wird die Pagetable konsultiert



129

Mindestanforderungen für den Aufbau von Page Table und TLB

- mind. 2-spaltige Einträge in die Page Table für virtuelle/physikalische Seitenadressen
- Ein *Valid-Bit* für jeden Eintrag, d.h. jede Seite zur Verwaltung der Information „Page present in memory“
- Zusätzliche muss die CPU haben:
 - Ein Trap bzw. Exception-Mechanismus, ausgelöst im „Page not present“-Fall
 - Wiederaufsetzbare Befehle, im Fall eines Seiten Fehlens (d. h. Befehle können ohne störende Nebeneffekte mitten in der Ausführung abgebrochen und später wiederholt werden)

Erweiterte Anforderungen für den Aufbau der TLB

- Zusätzlich soll der TLB für Speicherschutz sorgen
- Speicherschutz ist unerlässlich für Multitasking und/oder Multiuser-Betrieb
- Ist Speicherschutz vorhanden, spricht man von einer Memory Management Unit (MMU), die als wichtigste Komponente die TLB hat
- Trap = Falle; exception = Ausnahme

130

Speicherschutz durch das Betriebssystem mit Unterstützung der MMU

- Bei Multitasking/Multiuser-Systemen ist notwendig:
 - a) Schutz durch das Betriebssystem vor unberechtigten Anwenderzugriffen, d. h.
 - unberechtigtem Datentransfer (Lesen, Schreiben) und
 - unberechtigtem Control-Transfer (Prozeduraufrufen, Eintrittspunkte)
 - b) Schutz vor Zugriffen zwischen Tasks (Task-Isolation)
- Dazu sind in der MMU für jeden Seiteneintrag zusätzlich Bits enthalten, die vom Betriebssystem geschrieben und von der MMU automatisch bei jedem Seitenzugriff überprüft werden. (=“Schutzbits“)
- Diese Bits enthalten Privilegien und Zugriffsinformationen für alle Seiten eines Prozesses
- Sie sind zusätzlich zum Valid-Bit im TLB in weiteren Spalten pro Eintrag gespeichert
- Privilegien sind Rechte, die einzelnen Benutzern oder Prozessen zugewiesen werden. Bsp.: Administratorrechte = Recht zur Ausführung eines Systemprozesses

131

Teil B Rechnernetze

132

Klassifikation von Verbindungen bei Netzen nach der Distanz

- Man unterscheidet zwischen lokalen Netzen (LAN = Local Area Network), Stadtnetzen (Metropolitan Area Network) und Weitverkehrsnetzen (WAN = Wide Area Network)

	Interprocessor distance	Processors located in same	Example
keine Rechnernetze	0,1 m	Circuit board	Multiprocessors and Multicomputers
	1 m	System	
	10 m	Room	
Rechnernetze	100 m	Building	Local area network (LAN)
	1 km	Campus	
	10 km	City	Metropolitan area network (MAN)
	100 km	Country	
	1.000 km	Continent	Wide Area Network (WAN)
	10.000 km	Planet	
			The Internet

134

Ziele von Rechnernetzen

- **Datenverbund**
 - Zugriff auf entfernte Daten, z. B. über NFS.
 - **Funktionsverbund**
 - Zugriff auf Spezialrechner, z. B. Compute-Server, Data Base-Server).
 - **Lastverbund**
 - gleichmäßige Lastverteilung z. B. über LSF.
 - **Verfügbarkeitsverbund**
 - Fehlertoleranz, schrittweises Wachstum.
- NFS = network file system
- LSF = load sharing facility
- Speziell für das Internet gilt zusätzlich:
- Informationsverbund (riesiges Lexikon weltweit verteilter Informationen)
 - Infrastruktur für Handel und Dienstleistungen (eCommerce)

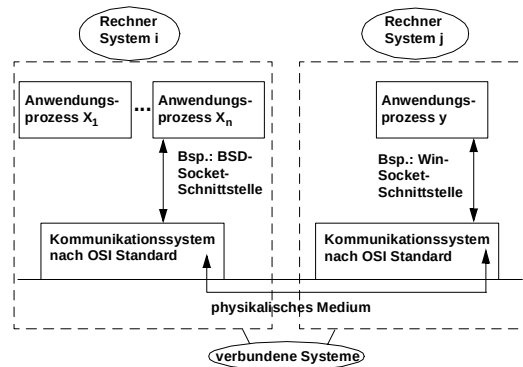
133

Standardmodell der Kommunikation: Das ISO/OSI-Referenzmodell für Offene Systeme

- OSI = OPEN SYSTEMS INTERCONNECTION = Untergruppe der ISO
- Einführung von neuen Grundbegriffe (Terminologie)
- Einführung eines neuen Modells für geschichtete Kommunikationssysteme
- Das ISO-7-Schichten-Modell schlägt sieben Schichten und ihre Funktionalität vor. Jede Schicht hat bei der Datenübertragung einen eigenen Satz von Aufgaben zu übernehmen.
- Datenaustausch ist nur zwischen benachbarten Schichten möglich, nicht kreuz und quer.
- Im Internet wurden vom ISO-7-Schichten-Modell nur Schicht 1-4 und Schicht 7 implementiert.

135

Standardmodell der Kommunikation bei ISO/OSI



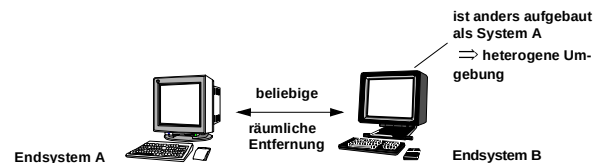
136

Grobstruktur im ISO-Referenzmodell

- Schicht 1 - 4**
- Transportorientierte Schichten = Schichten 1-4: Technische Erbringung von Bitstrom-Übertragungen
 - Pure Nachrichtenübertragung; nur Bedürfnis des Nachrichtenaustauschs unterstellt; keine höheren Funktionen
 - kein Bezug auf die Kooperationsbeziehung der Teilnehmer
 - Inhalt der Nachricht ist ohne Bedeutung für die transportorientierten Schichten
- Schicht 5 - 7**
- Anwendungsorientierte Schichten = Schichten 5-7: Anwendungsbezogene Kommunikationsdienstleistungen
 - Kooperation der Teilnehmer ist auf formale Weise berücksichtigt
 - Steuerung des Ablaufs (session)
 - Informationsdarstellung (Datenformate)
 - Kompensation von Fehlverhalten
 - Datenverschlüsselung (Kryptographie)
 - Datenkompression
- ⇒ Schicht 1-4 definiert Syntax der Daten
- ⇒ Schicht 5-7 definiert Semantik der Daten

137

Strukturierung der Kommunikation durch Schichten



Teil der Betriebssystem-Software bzw. Bibliotheken	Anwendungsschicht	Anwendungsspezifische Protokolle	7
	Darstellungsschicht	Datenformatierung und -darstellung	6
	Sitzungsschicht	Dialogsteuerung	5
	Transportschicht	Ende-zu-Ende-Datenaustausch	4
Teil der Rechner-Hardware	Vermittlungsschicht	Ende-zu-Ende-Vermittlung	3
	Sicherungsschicht	nur zwischen benachbarten Rechnern	2
	Physikalische Schicht	Physikalische Übertragung	1

138

Die sieben Schichten des ISO-Referenzmodells

- 1.) Die Bitübertragungsschicht ermöglicht die transparente Übertragung eines Stroms binärer Informationen über ein Kabel (ungesicherter Bitstrom, mech. u. el. Spezifikation)
- 2.) Die Sicherungsschicht soll insbesondere Übertragungsfehler entdecken und korrigieren (gesicherte Bits + Rahmung + Flusssteuerung)
- 3.) Die Vermittlungsschicht übermittelt Daten auf eine der Anwendung verborgenen Weise. Dazu wird von dieser Schicht selbstständig eine Netz-Route ausgewählt (von Rechner zu Rechner über Zwischenknoten)
- 4.) Die Transportschicht übermittelt die Daten von Endprozess zu Endprozess. Sie entlastet den Benutzer von den Details der Datenübertragung
- 5.) Die Kommunikationssteuerungsschicht koordiniert die Zusammenarbeit zwischen den verschiedenen miteinander kommunizierenden Anwendungsprozessen (Wiederanlauf = recovery, full-/halfduplex)
- 6.) Die Darstellungsschicht transformiert die Darstellung der übermittelten Daten in eine Form, die von den kommunizierenden Anwendungsprozessen verstanden wird (Datenformate, Kompression, Kryptologie = Verschlüsselung)
- 7.) Die Anwendungsschicht muss den Anforderungen der Benutzer genügen. Die Anwendungsschicht ist die einzige Zugriffsmöglichkeit der Anwendungsprozesse zur Datenübertragung (z. B. E-Mail, ftp, nfs,...)

139

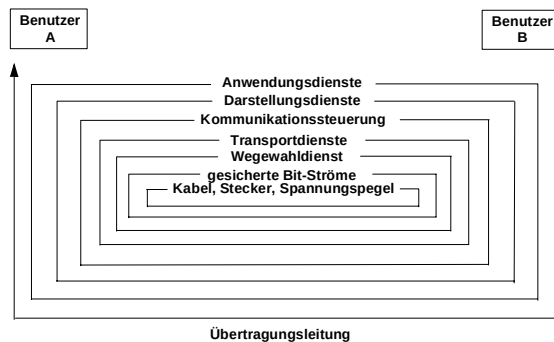
Sprachgebrauch und Prinzipien des ISO-Referenzmodells

- Offenes System für die Kommunikationsteilnehmer
 - jedes Rechnersystem (Hardware, Software, Peripherie, ...), das sich bei der Kommunikation an die OSI-Standards hält, kann kommunizieren.
- (N)-Schicht $1 \leq N \leq 7$
 - wird aus sämtlichen Einheiten einer (N)-Schicht in allen offenen Systemen gebildet. Analogie: die selben Stockwerke aller Gebäude der Welt bilden eine (N)-Schicht.
- (N)-Instanz, engl. (N)-Entity; Implementierung einer (N)-Schicht in einem System = Software/ Hardware
 - Es kann verschiedene Typen von (N)-Instanzen geben, die verschiedene Protokolle für die (N)-Schicht implementieren.
- Partnerinstanzen, Peer-Entities
 - Instanzen derselben Schicht an verschiedenen Orten. Partnerinstanzen erfüllen die Funktionen einer Schicht durch Datenaustausch von peer zu peer.

peer = Gleichgestellter, d. h. einer, der der gleichen Schicht angehört

140

Hierarchie von Diensten



- Jede Schicht benutzt die Funktionen (= Dienste) der darunterliegenden Schicht.
- Daten werden physikalisch nur auf der untersten Ebene zwischen den Rechnern übertragen, d. h. der Übertragungsleitung.
- Auf alle n-Schichten > 1 werden Daten innerhalb desselben Rechners nur „hin und her geschoben“.

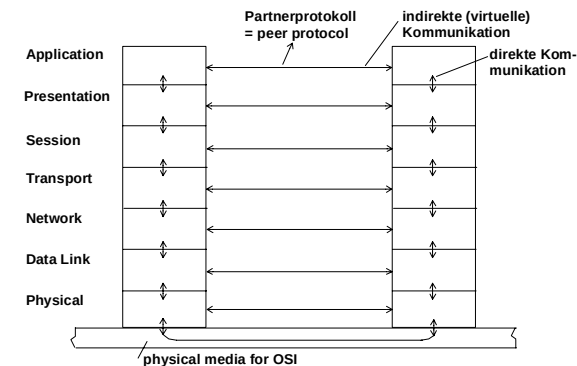
142

Was macht eine Schicht?

- Hauptaufgabe jeder Schicht ist es, der darüberliegenden Schicht Dienste anzubieten. Diese Dienste setzen sich zusammen aus:
 - Dienstleistungen, die innerhalb dieser Schicht erbracht werden und
 - der Summe der Dienstleistungen aller darunterliegenden Schichten.
- Schichten sind über sogenannte Dienstprimitive (= Dienstelemente) miteinander verknüpft.
- Da Schichten jeweils aufeinander aufbauen, ergibt sich eine Schichtenhierarchie bzw. eine Hierarchie von Diensten.

141

Indirekte Kommunikation im ISO-Referenzmodell/Partnerprotokolle

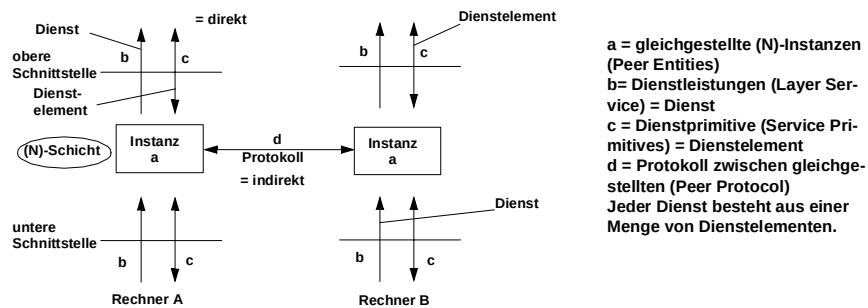


- Daten werden innerhalb eines Rechners von Schicht zu Schicht transportiert.
- Zwischen 2 Rechnern können Daten nur auf der Schicht 1 transportiert werden.
- Kommunikation zwischen Instanzen höher als Schicht 1 werden indirekt (virtuell) abgewickelt.
- Die indirekte Kommunikation wird mittels eines Protokolls zwischen Partnerinstanzen durchgeführt, so dass beide Partnerinstanzen „sich verstehen“ (Protokoll = gemeinsame Sprache).

143

Direkter und indirekter Datenaustausch

- Direkter Datenaustausch erfolgt mit Hilfe von sog. Dienstelementen eines Dienstes zwischen benachbarten Schichten des selben Rechners.
- Indirekter Datenaustausch erfolgt mit Hilfe eines Protokolls zwischen Partnerinstanzen auf verschiedenen Rechnern.
- **Direkte Kommunikation** mit Schichten (N+1) und (N-1) für Schicht N
- **Indirekte Kommunikation (Protokoll)** mit Partnerinstanzen (peer entities)



144

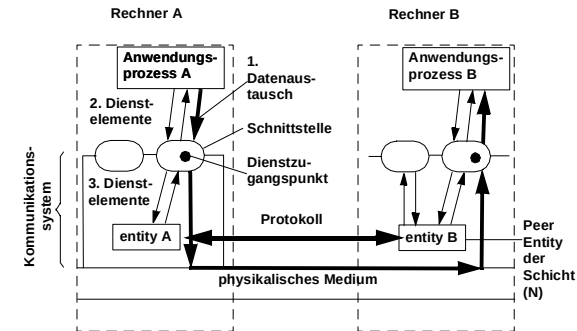
- Schnittstelle: Menge von Unterprogramm aufrufen einer direkt darunterliegenden Schicht N-1.
- Dienstzugangspunkt: Ein spezieller Unterprogrammaufruf.
- Jedes Dienstelement hat einen Dienstzugangspunkt, an dem das Resultat des Dienstelements vorliegt und an dem das Dienstelement zur Verfügung steht.

146

Was ist ein Protokoll?

Definition:

Menge der Regeln für den indirekten Datenaustausch zwischen Entities derselben Schicht (peer-to-peer communication).

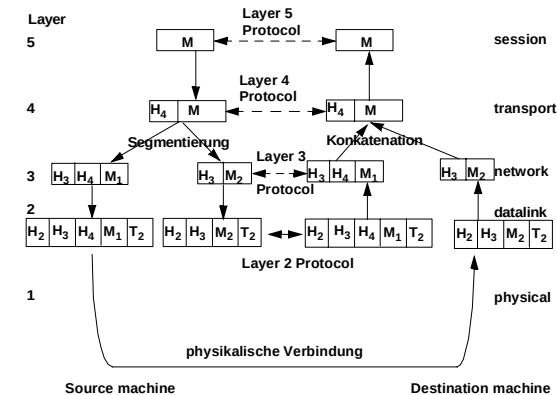


Datenaustausch → Dienstelement → Protokoll

145

Beispiel für virtuelle und reale Kommunikation nach ISO/OSI

Nachricht wird von „oben nach unten“ und dann von „unten nach oben“ durchgeschoben.

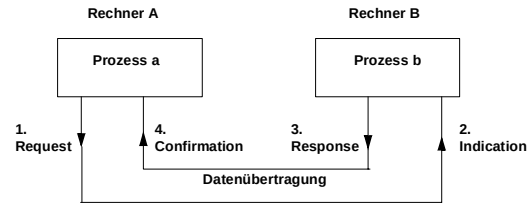


M = message = Nachricht H = header = Vorspann = Nachrichtenkopf T = trailer = Nachspann

- Nachricht wird bei jedem Übergang zwischen 2 Schichten um einen Vorspann (Header) verlängert.

147

Die 4 Arten von Dienstelementen

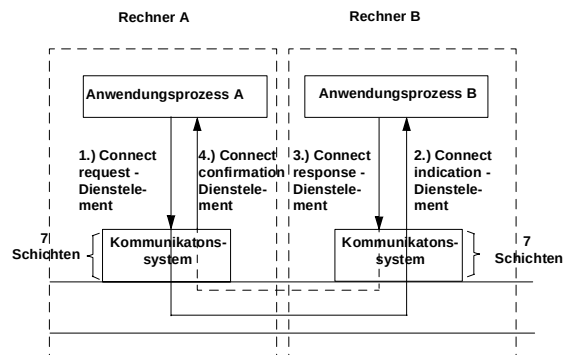


- Es gibt 4 Typen von Dienstelementen

- 1.) Anforderung (Request) = Aktivieren einer Dienstleistung durch den Benutzer
- 2.) Anzeige (Indication) = Dem Benutzer anzeigen, daß vom entfernten Benutzer ein Dienst angefordert wurde, ohne dass eine Dienstleistung vom Anbieter der Dienstleistung selbst ausgelöst wurde
- 3.) Antwort (Response) = Quittieren einer vorher. Anzeige durch den Benutzer
- 4.) Bestätigung (Confirmation) = Quittieren einer vorherigen Anforderung durch den Anbieter

148

Beispiel: CONNECT (Verbindungsaufbau)-Dienst



150

Die 3 Phasen der Kommunikation nach dem ISO/OSI-Modell

- Jede Verbindung, die in folgenden 3 Phasen abläuft, heißt verbindungsorientierte Kommunikation
 - 1. Phase: Verbindungsaufbau (Connect)
 - 2. Phase: Datentransfer (Data transfer)
 - 3. Phase: Verbindungsabbau (Disconnect)
- Frage 1:
 - Wie hängen die 3 Phasen mit den 4 Dienstelementen zusammen?
- Frage 2:
 - Wie hängen die 4 Dienstelemente mit den Diensten einer Schicht zusammen?

149

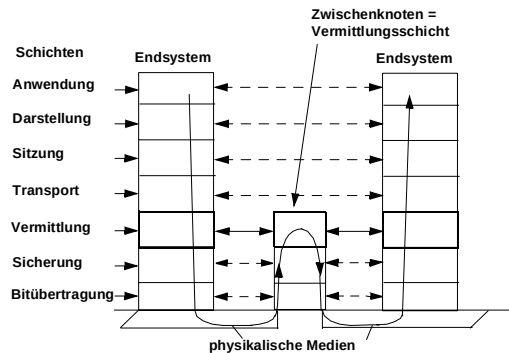
- request, indication, response und confirmation sind die 4 Typen von Dienstelementen
- In jeder der Schichten 2-7 gibt es diese 4 Typen von Dienstelementen (im Prinzip)
- Für jede der 3 Phasen einer Kommunikation
 - Verbindungsaufbau (connect)
 - Datentransfer (Data)
 - Verbindungsabbau (Disconnect)

gibt es im Prinzip diese 4 Typen von Dienstelementen

⇒ $4 \cdot 6 \cdot 3 = 72$ Kombinationen (im Prinzip)

151

Vermittlungssysteme als Zwischenknoten zwischen Endsystemen



⇒ Zwischenknoten hat nur die Schichten 1-3

- Mehrere Zwischenknoten können als ein großer virtueller Zwischenknoten angesehen werden, da jeder Zwischenknoten die Daten unverfälscht weiterreichen muss.

152

Definitionen in der Bitübertragungsschicht

mechanisch:

Abmessungen der Stecker, Anordnung der Pins, etc. z. B. „ISO 4903: Data Communication - 15 pin DTE/DCE interface connector and pin assignment“

elektrisch:

Spannungspegel auf Leitungen, etc. z. B. „CCITT X.27/V.11: Electrical characteristics for balanced double-current interchange for general use with integrated circuit equipment in the field of data communication“

funktional:

Klassifikation von Schnittstellenleitungsfunktionen (welcher Pin hat welche Funktion: data, control, timing, ground) z. B. „CCITT X.24: „List of definitions for interchange circuits between DTE and DCE on public data networks“

prozedural:

Regeln (Prozeduren) für die Benutzung der Schnittstellenleitungen, z. B. CCITT „X.21: Interface between DTE and DCE for synchronous operation on public data networks“

Und nicht zuletzt: Die Angabe der Bitrate, die zur Bitübertragung verwendet wird.

154

Bitübertragungsschicht

- ISO-Definition: Die Bitübertragungsschicht (physical layer) definiert die mechanischen, elektrischen, funktionalen und prozeduralen Eigenschaften, um physikalische Verbindungen zwischen Datenendeinrichtungen (DEE; englisch: DTE) und Datenübertragungseinrichtungen (DÜE, englisch: DCE, „Poststeckdose“) aufzubauen, aufrecht zu erhalten und abzubauen.

- Die Bitübertragungsschicht sorgt für die Übertragung eines transparenten Bitstroms zwischen Sicherungsschicht-Entities über physikalische Verbindungen. Eine physikalische Verbindung kann die Übertragung eines Bitstroms im Duplex-Mode oder im Halbduplex-Mode erlauben.

Duplex = bidirektionaler Datenaustausch

Simplex = unidirektionaler Datenaustausch

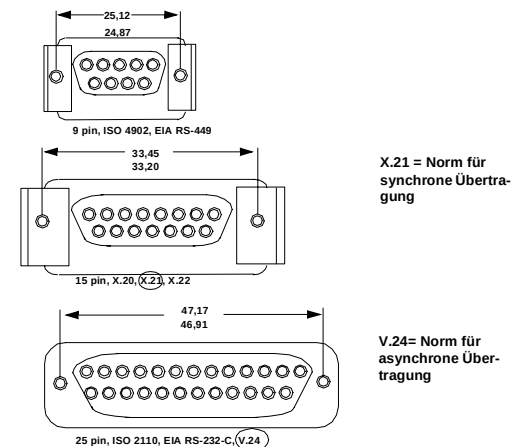
halbduplex = unidirektionaler Datenaustausch mit wechselnden Kommunikationsrichtungen

DEE = Datenendeinrichtungen = DTE = Data Terminal Equipment

DÜE = Datenübertragungseinrichtung = DCE = Data Communication Equipment

153

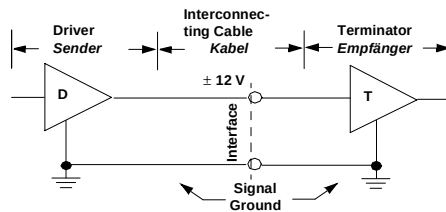
Beispiele für mechanische Spezifikationen



155

Beispiel für eine elektrische Spezifikation_

CCITT V.24 (EIA RS-232-C)
= asynchrone, serielle Übertragung



- V24 war für Sender/Empfänger gedacht, die aus diskreten elektronischen Bauelementen aufgebaut sind
- ⇒ einfache Struktur, geringe Datenraten
- Zwei Kabeladern pro Stromkreis, mit einer Erdung sowohl beim Sender als auch beim Empfänger
- Bitrate begrenzt auf 19,2 kbit/s, Entfernung begrenzt auf 15 m
- Erzeugt erhebliches Übersprechen aufgrund der fehlenden Abschirmung zwischen den Adern im Kabel

156

Asynchrone Übertragung (2) V.24

Vorteile:

- Es wird keine Synchronisierung der Taktgeber in den Endsystemen benötigt.
- Leicht zu implementieren.

Nachteile:

- Die Taktgeber können voneinander abweichen. Daher
 - ist die Rahmengröße sehr beschränkt (typischerweise auf ein Zeichen : entspricht 7-8 Bits)
 - nur anwendbar bei niedrigen Datenraten (<20 Mb/s)
- Die Start- und Stop-Bits stellen einen Mehraufwand (Overhead) dar.
 - Beispiel:
 - 7-Bit ASCII, 1 Paritätsbit, 1 Start-Bit, 1 Stop-Bit
 - d.h., insgesamt 10 Bits
- ⇒ Nur 70% der Leitungskapazität stehen in diesem Beispiel für echte Benutzerdaten zur Verfügung.

158

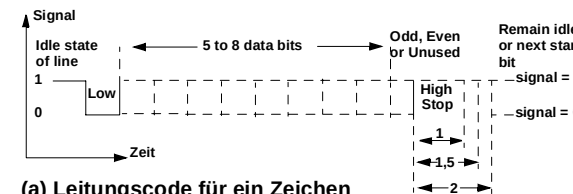
Asynchrone und synchrone Übertragung

- Asynchron (Bsp.: V.24): Kein explizites Taktsignal zwischen Sender und Empfänger vorhanden.
- Synchron (Bsp.: X.21): Taktinformation wird über die Leitung übertragen. Takt für die genaue Abstimmung der Bitimpulse (Synchronisation) auf beiden Seiten der Leitung benutzt.

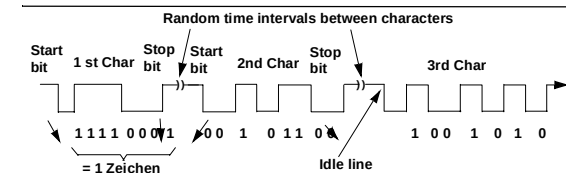
Beispiel für Asynchrone Übertragung (1) V.24

- Bei V.24 haben Sender und Empfänger besitzen voneinander unabhängige (lokale) Taktgeber.
- „Freie Leitung“ (= ohne Daten) entspricht einem kontinuierlich gesendeten 1-Bit ⇒ Problem der Unterscheidung zwischen freie Leitung und 1-Pegel.
- Das Start-Bit setzt die Leitung auf 0 und startet den Taktgeber des Empfängers.
- Ein Rahmen mit 5 bis 8 Bits (= ein Zeichen) wird übertragen.
- Das „Stop-Bit“ setzt die Leitung wieder auf 1. Dieses Signal muss 1, 1.5 oder 2 Bit-Intervalle andauern; entspricht 1, 1.5 oder 2 Stop-Bits.

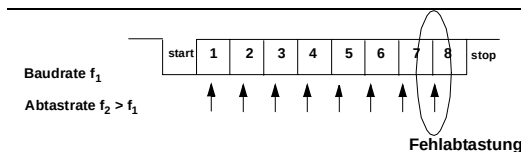
157



(a) Leitungscodierung für ein Zeichen



(b) asynchroner Bitstrom



(c) Effekt der auseinander laufenden Uhren

159

Synchrone Übertragung (1)

- Sende- und Empfangstakt laufen beliebig lange synchron.
- Eine Neusynchronisation nach jedem Zeichen (5-8 Bits) ist nicht erforderlich.
- Taktsignal wird entweder auf einer separaten Leitung übertragen (z. B. bei X.21 vom Dienstanbieter) oder aus dem Leitungssignal gewonnen (z. B. in den Modems, die an Zweidrahtleitungen angeschlossen sind).
- Letzteres erfordert spezielle Codierung der Daten (z. B. im Manchester-Code), um Daten und Takt gemeinsam im selben physikalischen Signal zu übermitteln.

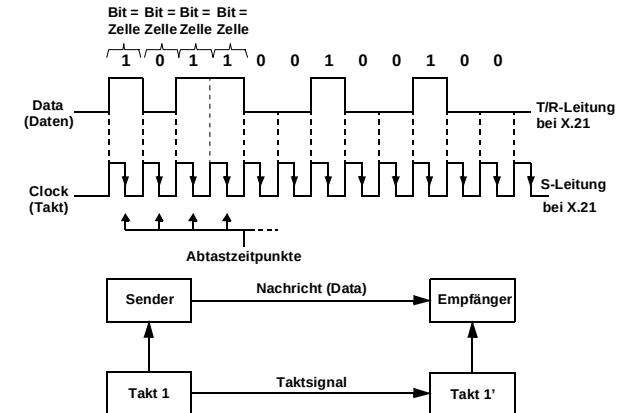
160

Physikalische Medien zur Datenübertragung

- Adernpaar, verdreht zur Verminderung von Störeinflüssen, deshalb „twisted pair“). Dies ist die klassische Telefonverkabelung. Braucht wenig Platz, enge Biegungsradien, sehr preiswert.
- Adernpaar, abgeschirmt („shielded twisted pair“ = Twinax-Kabel). Weniger anfällig bzgl. der Einkopplung von Störfrequenzen/Störströmen von außen. Ist teurer als „unshielded twisted pair“.
- Koaxialkabel: sehr störsicher, ermöglicht hohe Übertragungsraten. Teuer. Weit verbreitet zum Beispiel für die Ethernet-Verkabelung beim 10 Mb/s-Ethernet.
- Lichtwellenleiter (Glasfaser): sehr hohe Übertragungsraten, geringe Dämpfungen und Störeinkopplungen, aber aufwendige Anschlusstechnik, teuer.

162

Synchrone Übertragung (2)

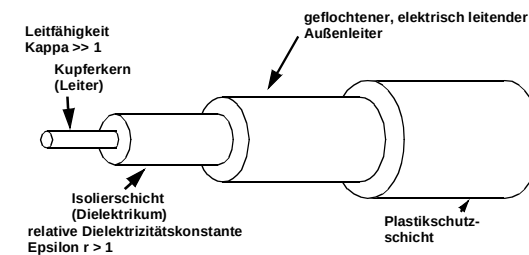


- Im Beispiel erfolgt das Auslesen der Datenbits bei abfallender Flanke des Taktsignals

161

Bsp. Koaxialkabel für 10 Mb/s-Ethernet

- sog. 50 Ohm-Koaxialkabel $\Rightarrow$ Wellenwiderstand = 50 Ohm



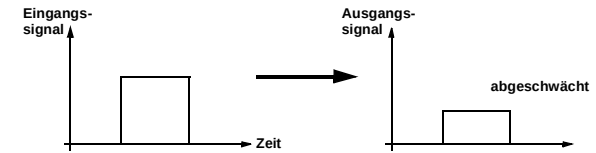
- Datenrate bei Thick-Ethernet ist auf 10 Mbit/s begrenzt.
- Maximale Kabellänge bei Thick-Ethernet ist 500 m pro Ethernet-Segment.
- Maximal 100 Transceiver (= Anschlüsse von Stationen) pro Segment.
- Maximal vier Repeater (= Verstärker) zwischen Sender und Empfänger.
- Abstand zwischen den Anschlüssen muss ein Vielfaches von 2,5 m sein, damit sich sog. stehende Wellen auf dem Kabel ausbilden können.

163

Glasfaserkabel

- Sehr hohe Datenraten:
 - Theoretisches Limit: 300 TeraBit/s = ungefähr die Frequenz des Lichts, Praktisches Limit: = 10-100 GigaBit/s (zur Zeit)
- als Transmitter und Receiver werden Halbleiterbauelemente verwendet:
 - Laserdiode oder Leuchtdiode als Sender, Photodiode als Empfänger
- Beschränkende Faktoren für die Datenrate: Absorption und Dispersion in der Faser

1. Absorption in der Faser:



2. Dispersion in der Faser:



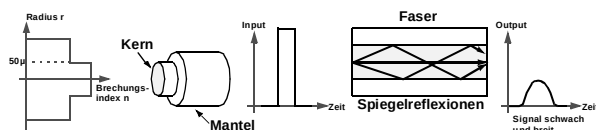
- Die Verbindung zwischen zwei Fasern ist schwierig, da nur 5 µm - 50 µm Durchmesser im Kern der Faser, wo das Licht geleitet wird.

164

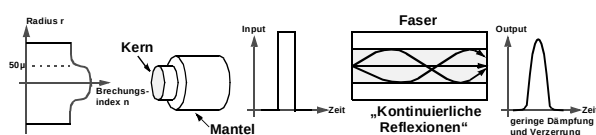
165

Technologie der Lichtwellenleiter

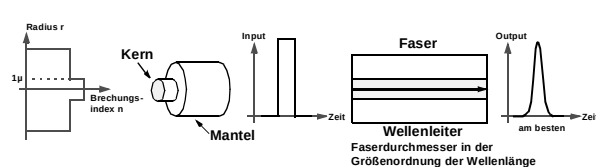
• Stufenindex-Faser



• Gradientenindex-Faser

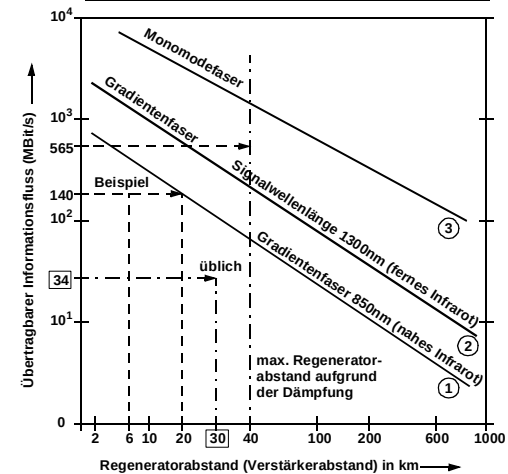


• Monomode-Faser



166

Vergleich der Glasfaserarten



Systemparameter:

- ① Gradientenfaser, LED als Sender, 850nm Wellenlänge
- ② Gradientenfaser, Laser als Sender, 1300nm Wellenlänge
- ③ Monomodefaser, Einmode-Laser als Sender, 1300nm Wellenlänge

167

Modulation

- Modulation = Aufprägung von digitalen Daten oder analogen Signalen auf hochfrequente analoge Signale (Beispiel, Rundfunk, Rechnernetz)

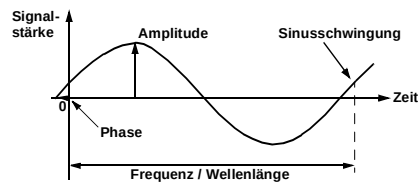
- Modem (= Modulator + Demodulator) = dient zur Modulation digitaler Daten (Modulator zum Senden; Demodulator dient zum Empfangen)

- Es gibt 3 Modulationsarten

1.) Amplitude Modulation (AM), Amplitude Shift Keying (ASK)

2.) Frequency Modulation (FM) Frequency Shift Keying (FSK)

3.) Phase Modulation (PM) Phase Shift Keying (PSK)



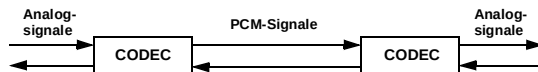
168

Codierung

Die quantisierten Werte werden durch die Zuordnung eines - frei wählbaren - (Binär-) Codes gekennzeichnet. Anstelle des ursprünglichen Analogsignals wird der digitale Codewert übertragen.

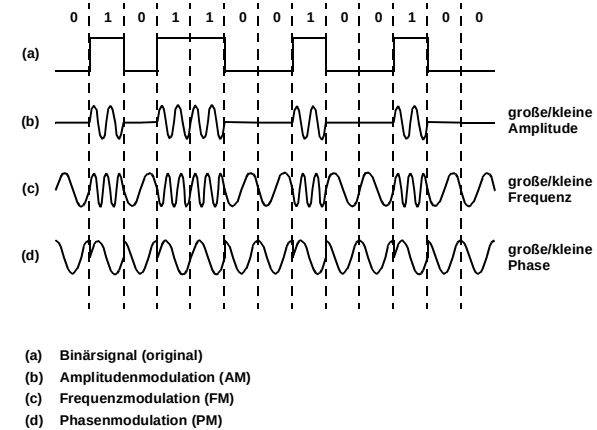
Im einfachsten Fall wird ein reiner Binärcode als Codierung des diskreten Digitalwertes gewählt (Darstellung als Binärzahl) $\hat{=}$ Pulse-Code-Modulation (PCM)

Die A/D-Umsetzung (Abtastung/Quantisierung) und Codierung sowie die Rückkonvertierung erfolgt im sogenannten CODEC (Codierer/Decodierer).



170

Modulationstechniken



169

Leitungscodierung digitaler Signale (1)

NRZ (Non Return to Zero) = ganze Gruppe von Codes

Das gemeinsame Kennzeichen der NRZ-Leitungscodes ist ein fester Pegel während eines Bitintervalls. Signal (Signalwechsel) erfolgen an den Intervallgrenzen. NRZ ist sehr einfach zu implementieren.

NRZ - L (Level) = 1 spezieller Code

Zuordnung eines festen Pegels zu einem binären Datenelement:

„1“ = hoher Pegel

„0“ = niedriger Pegel

NRZ-L ist der Standard innerhalb von Digitalgeräten (Rechnern, usw.)

• non return to zero = der Signalpegel für eine „1“ kehrt innerhalb der Bitzelle nicht auf 0 Volt zurück, sondern bleibt bei High-Pegel.

• Bitintervall = Bitzelle

171

Leitungscodierung digitaler Signale (2)

- NRZ-M (Mark), NRZ-S (Space)
Differenzielle Codierung: Es wird nicht der absolute Signalwert eines Signalelements in der Zuordnungsvorschrift verwendet, sondern der Signalwert in Abhängigkeit von der Polarität des vorhergehenden Signalelements.
- NRZ-M: Signalwechsel (=Pegelwechsel; Übergang in den entgegengesetzten Signalwert) zur Darstellung des Datenelements „1“.
- NRZ-S: Signalwechsel (= Pegelwechsel) bei Datenelement „0“.
- Vorteile gegenüber NRZ-L: Unter Einfluss von Störungen (Rauschen) sind Signalwechsel leichter zu detektieren als Signalpegel, die mit einem Schwellwert verglichen werden müssen.
- Nachteile aller NRZ-Codes: Die Gleichstromkomponente und Synchronisierungsprobleme zwischen Sender und Empfänger (z. B. bei langen „0“-Folgen bei NRZ-L und NRZ-M)

172

Leitungscodierung digitaler Signale (4)

- Biphase-L (Level)
- Wird auch als Manchester-Code bezeichnet. Signalwechsel in der Mitte des Intervalls. Je nach Bitfolge (Datenelemente) Hilfswechsel an den Intervallgrenzen erforderlich (verdoppelte Baudrate). 1. Hälfte der Bitzelle hat Absolutwert des Signals: High = hoher Pegel; low = niedriger Pegel.
- Biphase-M (Mark), Biphase-S (Space)
- Grundsätzlich immer ein Signalwechsel am Beginn des Intervalls.
- Biphase-M: bei „1“ zusätzlicher Signalwechsel in der Intervallmitte
- Biphase-S: bei „0“ zusätzlicher Signalwechsel in der Intervallmitte
- Differential Manchester
- Grundsätzlich ein Signalwechsel in der Intervallmitte. Bei „0“ zusätzlich ein Signalwechsel am Intervallbeginn

174

Leitungscodierung digitaler Signale (3)

- Return to Zero = ein spezieller Code
Gekennzeichnet durch einen Rechteckimpuls in der ersten Hälfte des Bitintervalls für das Datenelement „1“; Danach Rückkehr in einen Grundzustand (Zero). Die Baudrate (Rate der Rechteckimpulse) ist im Extremfall (Folge von Einsen) doppelt so hoch wie die Bitrate. Keine Vorteile gegenüber NRZ, aber höhere Bandbreite und alle anderen NRZ-Nachteile.
 - Biphase = ganze Gruppe von Codes = Zweiphasen-Code mit zwei Abschnitten pro Bitzelle
 - Alle Biphase-Leitungscodierungen haben mindestens einen Signalwechsel pro Bitintervall und höchstens zwei Signalwechsel pro Bitintervall. (Bitintervall = Bitzelle)
- Vorteile:
- Leichte Synchronisierung, da stets mindestens ein Signalwechsel pro Bitintervall (eine „Impulsflanke“ zum Triggern des Empfängers)
 - Keine Gleichstromkomponente, da Mittelwert = 0 möglich
 - Fehlererkennung auf Signalebene möglich: Fehlen eines erwarteten Übergangs leicht erkennbar

173

Leitungscodierung digitaler Signale (5)

Bipolar

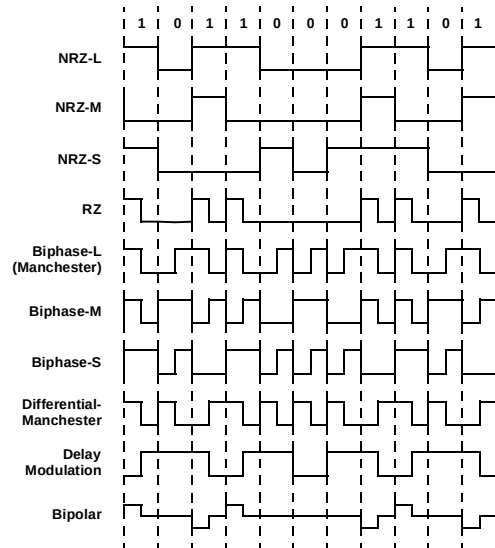
Beispiel einer Leitungscodierung mit mehr als zwei Signalwerten (hier tertiäres Signal).

„1“ wird abwechselnd durch positiven oder negativen Impuls in der 1. Hälfte des Bitintervalls dargestellt, dadurch keine Gleichstromkomponente.

Auch AMI (Alternate Mark Inversion) genannt.

175

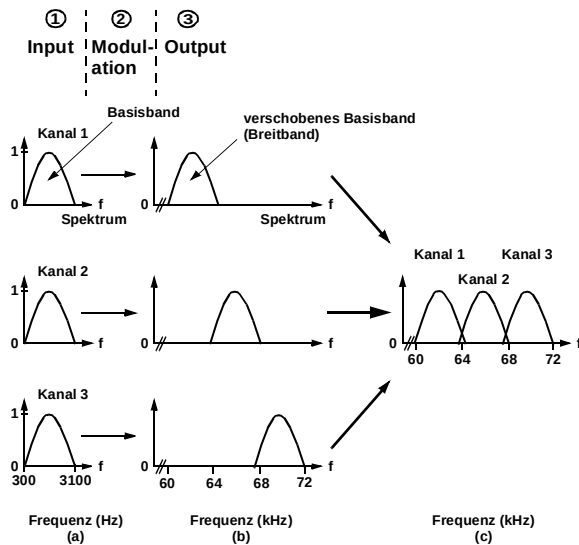
Leitungscode



NRZ-L = non return to zero-level
 NRZ-M = non return to zero-mark
 NRZ-S = non return to zero-space
 RZ = return to zero

176

Beispiel für Frequenzmultiplexing



- (a) Die ursprünglichen Bandbreiten (= Basisband)
- (b) Die Bandbreiten mit verschobener Frequenz (= Breitband)
- (c) Auf dem Übertragungsweg (Kabel)

178

Multiplexing: Mehrfachnutzung von Übertragungswegen

Definition: Übertragungsweg

- physikalisch-technisches Transportsystem für Signale (z. B. Kabel)

- Definition: Übertragungskanal

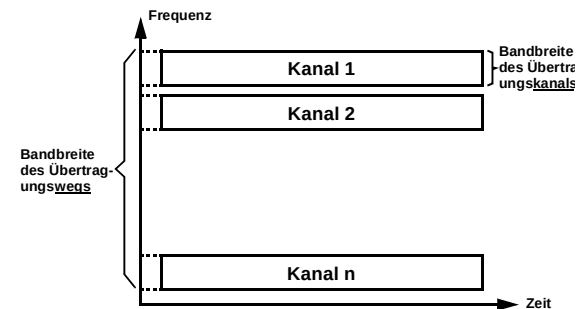
- Abstraktion eines Übertragungsweges für einen Signalstrom = logischer Übertragungsweg bzw. virtueller Übertragungsweg.

- Auf einem Übertragungsweg können oft mehrere Übertragungskanäle parallel betrieben werden. So ist beispielsweise eine Aufspaltung der gesamten Übertragungskapazität eines Übertragungsweges auf verschiedene Sender-Empfänger-Paare möglich.
- Die Zusammenfassung von mehreren Übertragungskanälen auf einem Übertragungsweg heißt Bündelung oder Multiplexing.

177

Frequenzmultiplexing (2) Frequency Division Multiplexing

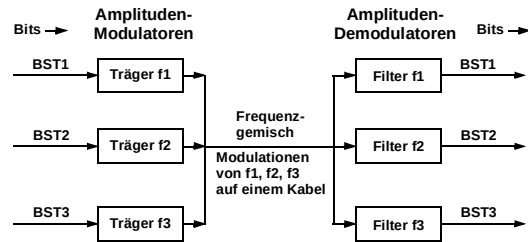
- Breitbandige Übertragungswege ermöglichen die Unterbringung vieler Übertragungskanäle in unterschiedlichen Frequenzbereichen (Frequenzbändern), d. h., man teilt die verfügbare Bandbreite in eine Reihe von - nicht notwendigerweise gleich breiten - Frequenzbändern auf und ordnet jedem Frequenzband einen Übertragungskanal zu.



$$\sum_{i=1}^n \text{Bandbreite (Kanal } i) \leq \text{Bandbreite des Übertragungswegs}$$

179

Schema der technischen Realisierung eines Frequenzmultiplex-Systems



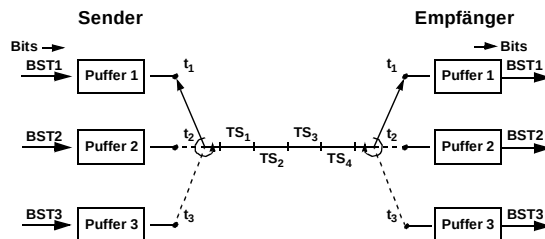
BST i = Bitstrom i, entspricht Kanal i

180

Synchrones Zeitmultiplexing (2)

Zeitmultiplexing ist nur für zeitdiskrete Signale einsetzbar (bevorzugt zeit- und wertdiskrete Signale = Digitalsignale), da das Signal nicht kontinuierlich übertragen wird.

Festes Zeitmultiplex mit starrer Zeitscheibenzuteilung (synchron)



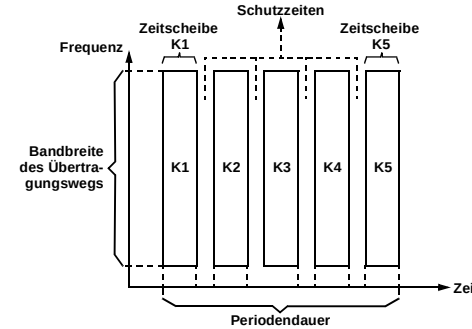
Jedem der n Sender wird periodisch eine Zeitscheibe (time slot, time slice) TS1, TS2,...TSn zugeeilt. Sender, Abtaster und Detektionsmechanismus beim Empfänger laufen im gleichen Takt. Deshalb auch als synchrones Zeitmultiplexing bezeichnet. Beispiel: Telefonvermittlungssystem (PCM-System).

Zeittakt-Stabilität ist wichtig!

182

Synchrones oder starres Zeitmultiplexing (Time Division Multiplexing)

- Die gesamte Übertragungskapazität (die ganze verfügbare Bandbreite) wird einer Sender-Empfänger-Kombination zur Verfügung gestellt. Nach einer Schutzzeit wird dann die gesamte Kapazität des Übertragungsweges dem nächsten Kanal zugeteilt. Pro Periode erhält also jeder Kanal einen Zeitschlitz (time slot) = time sharing des Übertragungsweges.



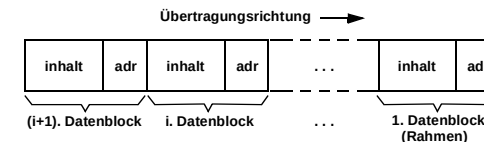
$$\sum_{i=1}^n \text{Zeitscheibe (Kanal)}_i \leq \text{Periodendauer des Übertragungsweges}$$

181

Asynchrones Zeitmultiplexing

Der Übertragungsweg wird dem Sender nicht fest, sondern nach Bedarf zugeteilt. Der Empfänger kann aus der Zeitlage der Zeitscheiben nicht mehr die Herkunft identifizieren! Es wird eine Kennung pro Datenblock (Paket, Zelle) erforderlich (Empfängeradresse, Kanalkennzahl, o. Ä.) Beispiel: Ethernet, ATM,...

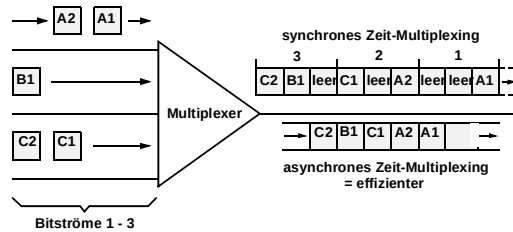
Schematischer Aufbau eines Übertragungsblocks mit Kennung



Das anforderungsgesteuerte Zeitmultiplexing (demand multiplexing) wird auch als statistisches Zeitmultiplexing (STDM = statistical time division multiplexing) bezeichnet.

183

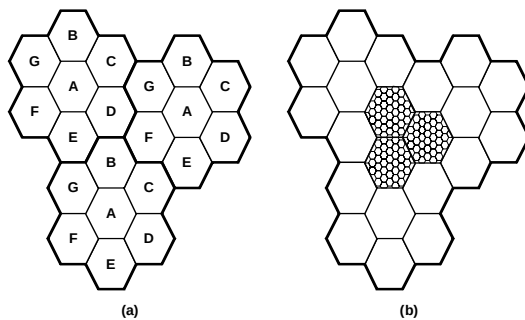
Zeit-Multiplexing-Techniken im Vergleich



- Bei LAN's wird asynchrones Zeitmultiplexing eingesetzt (LAN = local area network)
- Frequenzmultiplex dient zur Unterscheidung zwischen verschiedenen Gebieten und zwischen Senden/Empfangen
- Innerhalb eines Sendegebietes wird Zeitmultiplexing (TDM) benutzt.

184

Aufteilung der Frequenzen in Zellen



gleiche Buchstaben bezeichnen gleiche Zellen

a) homogene Zellgröße auf dem flachen Land

b) kleinere Zellen für Städte

186

Mobilfunknetze („Handy“, Cell phone Mobile phone)

- Internationale Standards für den digitalen Zelfunk sind verabschiedet (z. B. GSM in Europa) für Handy; UMTS in Zukunft.
- Starke Verbreitung für die Telefonie (D 1, D 2, E-plus,...)
- Bandbreite sehr niedrig (einige KHz); 8 Zeitscheiben pro Trägerkanal.
 - abhängig von der Breite eines Trägerkanals (200 KHz)
 - Dieser muss wegen des Frequenzmultiplexings innerhalb einer Zelle sehr schmalbandig sein: pro Zelle 124 KHz-Duplexkanäle + 8 Zeitscheiben pro 200 KHz Träger à 0,5 ms.
- Datenanwendungen in Mobilfunknetzen entwickeln sich rasant (Daten übers Handy, z. B. Fax mit 9600 Baud (Bit/s)).
- Zunehmend wird auch Mobilfunktechnik innerhalb von Gebäuden verwendet, z. B. für den Anschluss von autonomen Robotern an ein LAN oder für die Bürokommunikation von einem Meeting aus.

GSM = global system for mobile communication

UMTS = universal mobile telephone system

185

Sicherungsschicht (Data Link Layer)

- 1.) Übertragungsfehler: Ursachen
- 2.) Fehlererkennungs- und Fehlerkorrekturcodes
- 3.) Bit Stuffing und Rahmenbegrenzer
- 4.) Bestätigung und Sequenznummern
- 5.) Flusskontrolle
- 6.) Beispiel: HDLC

Aufgaben der Sicherungsschicht (Schicht 2)

- Verdeckung (= Unsichtbarmachung) von Übertragungsfehlern zwischen direkt benachbarten Partnern (keine Zwischenknoten) bestehend aus den Teilschritten Erkennung und Behebung.
- Flusststeuerung des Datenverkehrs zum Nachbarknoten.
- Bei LANs zusätzlich: Medienzugangskontrolle zum gemeinsam genutzten Übertragungsmedium

flow control = Flusststeuerung

187

Übertragungsfehler: Ursachen

- 1.) sog. weißes Rauschen = thermisches Rauschen von Leitungen und Bauteilen
- 2.) Dämpfung und Verzerrung
 - Signalverzerrung ist abhängig von der Frequenz
- 3.) Übersprechen auf Leitungen verursacht durch Magnetfelder
- 4.) Übersprechen auf Leitungen verursacht durch kapazitive Kopplung
- 5.) Impulsstörungen durch elektrische Felder (z. B. Erdschleifen)
 - Wird z. B. durch elektrische Geräte verursacht
 - Bei einer Störung von 10 ms Dauer entspricht dies 560 Bit bei einem 56kbit/s-Modem

Viele Fehler kommen kurz aufeinanderfolgend (sog. Bursts auch Fehlerbündel genannt)

Vorteil: Nur wenige Blöcke enthalten Fehler.

Nachteil: Schwer zu erkennen und zu korrigieren, da viele Fehler auf einmal gesammelt auftreten, wenn sie kommen. Dazwischen ist dann wieder eine fehlerfreie Zeit.

188

Fehlererkennung durch Paritätsbit

	Bsp. 1	Bsp. 2	Bsp. 3	
ein ASCII-Zeichen	Bit 1	0	1	1
	Bit 2	1	0	1
	Bit 3	0	0	0
	Bit 4	0	1	0
	Bit 5	0	0	0
	Bit 6	0	1	0
	Bit 7	1	1	1
Paritätsbit	0	0	1	gerade
	1	1	0	ungerade

- Das Paritätsbit ergänzt die Anzahl der Einsen im Zeichen so, dass sie gerade/ungerade wird (gerade Parität = gerade Zahl von Einsen).
- Berechnung der gesamten Quersumme (inkl. Parity) so, dass sie gerade oder ungerade wird.

190

Fehlererkennungs- und Fehlerkorrekturcodes

Fehler heißt: Die empfangene Information entspricht nicht der gesendeten.

- Zur Fehlerkorrektur wird die eigentliche Information durch Kontrollinformationen (Redundanz) ergänzt.
- Je nach Umfang der Redundanz kann eine gewisse Anzahl an Fehlern erkannt oder sogar korrigiert werden.
- Der Zusammenhang zwischen dem Umfang der notwendigen Redundanz, der Fehlerhäufigkeit der Übertragungsstrecke, der Entdeckung und der eventuellen Korrektur eines Fehlers wird in der Nachrichtentechnik behandelt.

189

Querparität (= 1 Bit pro Zeichen)

- Wird zum Beispiel bei der asynchronen Übertragung verwendet. Beispiel: V. 24 hat Paritätsbit gemäß Querparität.
- Die 5 bis 8 eigentlichen Informationsbits werden durch ein weiteres (redundantes) Bit innerhalb des Rahmens erweitert.
- Das Redundanzbit (= Paritätsbit) ergänzt die Anzahl der Einsen innerhalb der Information auf eine gerade oder ungerade Anzahl. (⇒ 9-Spur-Bänder)

```

1 0 0 1 1 1 0 0 0
1 0 1 1 1 1 0 0 1
  └──────────┬──────────┘
            Information Paritätsbit
                    (gerade Parität)
  
```

- Es kann nur ein Fehler erkannt und Keiner korrigiert werden.

191

Cycling Redundancy Check (CRC)

Idee: Betrachte Bitfolgen als Darstellung eines Polynoms von X nur mit den Koeffizienten 0 und 1. Beispiel: $11001 = x^4 + x^3 + x^0$ ist ein Polynom von Grad 4

- Wähle ein Generatorpolynom G(x) vom Grad g.
- Hänge an die Nachricht M(x) eine CRC-Prüfsumme derart an, dass das Polynom T(x), dargestellt durch die Nachricht mit angehängter Prüfsumme, durch G(x) teilbar ist.
- Übertrage T(x)
- Algorithmus für CRC:
 - 1.) Wir haben G(x) vom Grad g und hängen g 0-Bits an die Nachricht, also $x^g M(x)$. Das Anhängsel ist damit vom Grad g-1.
 - 2.) Teile $x^g M(x)$ durch G(x), Division modulo 2 (= ohne Borgen)
 - 3.) Subtrahiere den Rest von $x^g M(x)$, modulo 2 (= ohne Borgen)
 - 4.) Das Ergebnis ist T(x) = die Nachricht samt Prüfsumme für die Übertragung. T(x) ist garantiert durch G(x) teilbar.
 - 5.) Der Empfänger teilt T(x) durch G(x). Gibt es keinen Rest, ist T(x) mit hoher Wahrscheinlichkeit fehlerfrei übertragen worden.

192

Generatorpolynome in internationalen Standards

- CRC-16 = $x^{16} + x^{15} + x^2 + 1$ (ISO)
- CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$

beide werden für 8-Bit-Zeichencodes benutzt

Fehlererkennung mit CRC-16, CRC-CCITT (16 Bits)

erkannt werden:

- alle einfachen und zweifachen Bitfehler
- alle Fehler mit einer ungeraden Bitzahl
- alle stoßweisen Fehler (= Bursts) mit einer Länge < 16 Bit
- 99,998% aller längeren stoßweisen Fehler > 16 falsche Bit

Implementierung des CRC

In Hardware (z. B. im HDLC-Chip) mit Hilfe eines Schieberegisters und der bitweisen XOR-Funktion.

⇒ CRC lässt sich sehr einfach implementieren.

194

Beispiel für CRC-Berechnung

Rahmen: 1 1 0 1 0 1 1 0 1 Generator: 1 0 0 1 1 ⇒ Grad g = 4
 Nachricht nach dem Anhängen von 4 Nullbits: 1 1 0 1 0 1 1 0 1 1 0 0 0 0, dann teilen:
 1 1 0 1 0 1 1 0 1 1 0 0 0 0 : 1 0 0 1 1 = 1 1 0 0 0 0 1 0 1 0 mit Rest 1 1 1 0

```

- 1 0 0 1 1      ← kein Borgen!
  "1 0 0 1 1
- 1 0 0 1 1
  "0 0 0 0 1
- 0 0 0 0 0
  "0 0 0 1 0
- 0 0 0 0 0
  "0 0 1 0 1
- 0 0 0 0 0
  "0 1 0 1 1
- 0 0 0 0 0
  "1 0 1 1 0
- 1 0 0 1 1
  "0 1 0 1 0
- 0 0 0 0 0
  "1 0 1 0 0
- 1 0 0 1 1
  "0 1 1 1 0
- 0 0 0 0 0
  "1 1 1 1 0
  
```

Rest: 1 1 1 0
 (Division modulo 2: Kein Übertrag bei der Subtraktion)

193

Bit Stuffing und Rahmenbegrenzer

Zur Anwendung von fehlererkennenden und fehlerkorrigierenden Codes muss der Datenstrom in einzelne Rahmen unterteilt werden. ⇒ Pakete bzw. Rahmen.

Problem: Wie kann man solche Rahmen im Bitstrom begrenzen (= kennzeichnen), wenn jedes beliebige Bitmuster in den Nutzdaten vorkommen kann?

Lösung: Bit Stuffing (Bitstopfen)

Als Begrenzer (= Kennzeichner) wählt man 01111110 (= 6 Einsen hintereinander). Der sender fügt nach fünf Einsen im Nutzdatenstrom IMMER eine 0 ein.

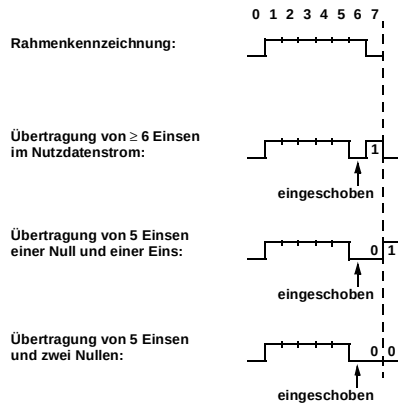
Beispiel:

Sender	0	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	1		
Leitung	0	1	1	1	1	1	0	1	0	1	0	1	1	1	1	1	0	0	1	0	1
Empfänger	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0	1	0	1

Wenn der Empfänger nach fünf Einsen eine Null sieht, entfernt er diese aus dem Datenstrom.

195

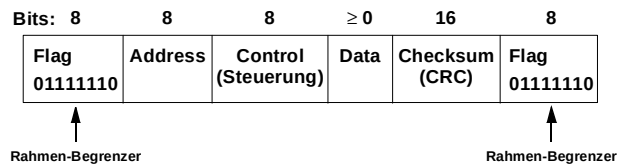
Bit Stuffing-Beispiele



⇒ Sechs Einsen hintereinander kann es auf der Leitung nur am Rahmenanfang geben.

196

Beispiel für ein Rahmenformat der Schicht 2



- Flag = „Flagge“ = Kennzeichnung
- Auf der Schicht 1 spricht man von Bitstrom
- Auf der Schicht 2 spricht man von Rahmen
- Auf allen Schichten ≥ 3 spricht man von Paket

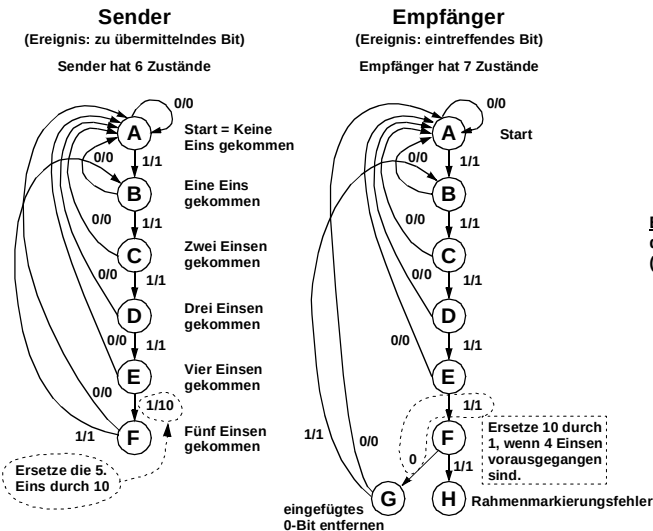
Was steht im „Control“-Feld eines Rahmens?

- Es stehen u. a. Bestätigungen (Acknowledgements) und Sequenznummern

Acknowledgement = Anerkennung, Bestätigung, Quittung

198

Realisierung des Bitstopfens



197

Bestätigung bzw. sog. Sequenznummern werden benutzt für

- Fehlererkennung fehlerhafter oder verlorener Blöcke
- Pufferverwaltung
- Flusssteuerung (= flow control)

Puffer (engl. buffer) ist ein kleines Stück Speicher zum Halten eines oder mehrerer Datenpakete

Puffer werden vom Sender wie vom Empfänger eingesetzt

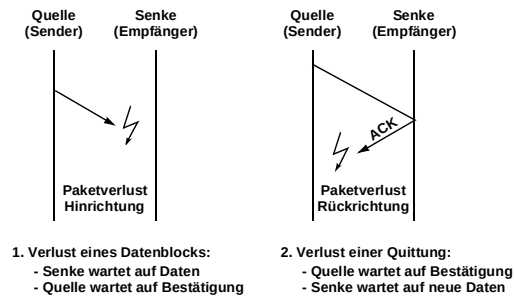
Beachte: to control: steuern, to check: kontrollieren, feed back control: regeln

Bestätigungen (Acknowledges)

- Für jedes Datenpaket vom Sender zum Empfänger wird eine Bestätigung in Gegenrichtung geschickt, sofern das Datenpaket unversehrt empfangen wurde.
- Wird das Paket nicht empfangen oder fehlerhaft empfangen, erhält der Sender keine Bestätigung.
- Probleme gibt es, wenn das Paket nicht empfangen wurde oder die Bestätigung verloren ging (⇒ zwei verschiedene Probleme).

199

Zwei Fälle von Datenverlust



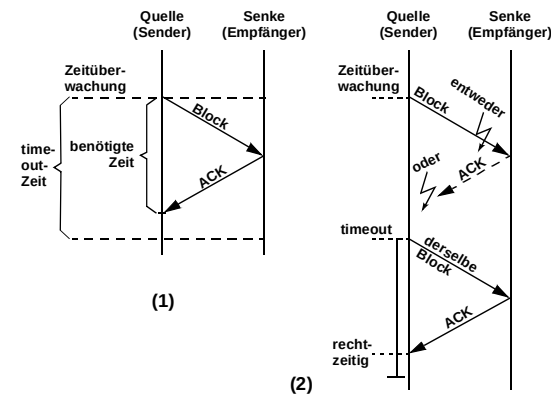
⇒ Problem von Acknowledges:

- Ohne Zeitschranke (Time-out): ewige Blockierung des Senders (ewig)
- Lösung: Einführung einer Zeitschranke (Time-out) beim Sender.

200

Bestätigung mit Zeitschranke (time out)

Zeitüberwachung auf der Senderseite



- 1.) Beispiel für eine fehlerfreie Übertragung
- 2.) Beispiel für das Erkennen eines Nachrichtenverlustes durch Ablauf der Zeitüberwachung

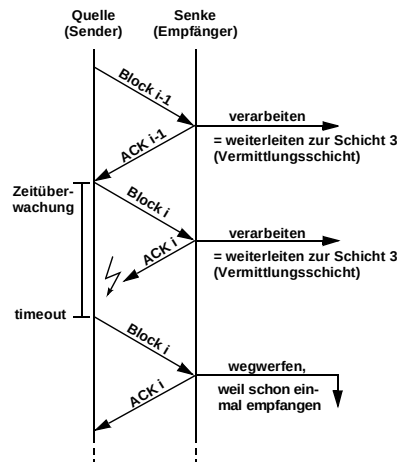
Neues Problem:

Der Block wird zweimal übertragen. Der Empfänger weiß nicht, ob er den Block zur Verarbeitung an die nächsthöhere Schicht weitergeben soll.

201

Bestätigung mit Zeitschranke und Sequenznummern

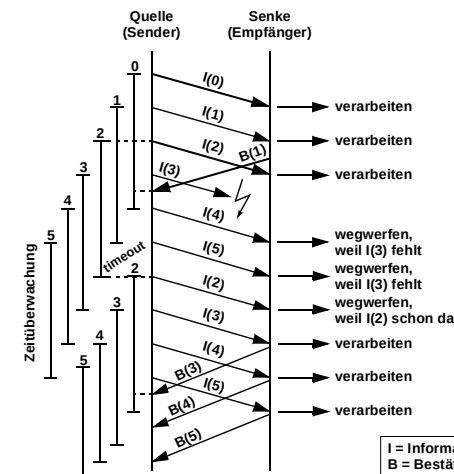
Lösung: Sequenznummern (= Blockadressen). Bei der Wiederholung des Rahmens wird die Sequenznummer beibehalten. Zurückgeschickte Sequenznummer dient zugleich als Bestätigung.



202

Sammelquittung über Sequenznummern

Mit einer Sequenznummer können auch mehrere Informationsblöcke quittiert werden.



Zurückgeschickte Sequenznummer B (3) dient als Bestätigung für die Blöcke 2 und 3

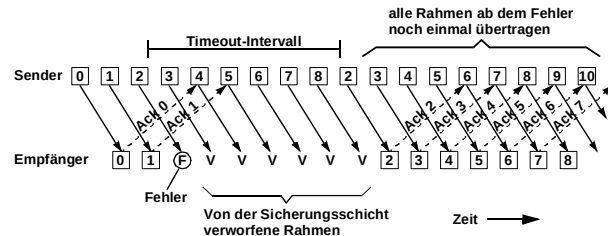
203

Fehlererkennung und -behebung

- Nachdem ein Übertragungsfehler durch Paritybit oder CRC-Prüfung und time out mit Sequenznummer und Acknowledge erkannt wurde, muss das betreffende Paket neu geschickt werden.
- Für die Paketwiederholung gibt es verschiedene Strategien.

Fehlerbehebung mit „go-back-n“ und ohne Pufferung beim Empfänger

Im Falle eines Fehlers bleibt das Ack aus. Nach Ablauf des Timers überträgt der Sender sämtliche Rahmen ab dem Unbestätigten neu.



- Dass ein Fehler aufgetreten ist, wird vom Empfänger am falschen CRC festgestellt und vom Sender am time out.

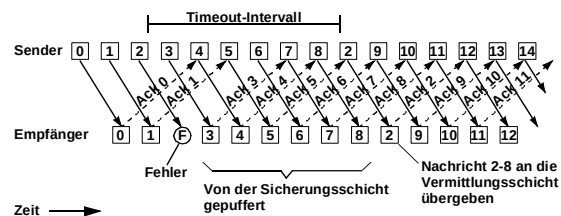
204

Dieses Verfahren ist allerdings vergleichsweise kompliziert und wird in der Praxis kaum eingesetzt.

- Sammelquittungen haben den Vorteil höherer Ausnutzung der Übertragungsbandbreite.

Fehlerbehebung durch „selective repeat“

Der Empfänger bestätigt jeden korrekt erhaltenen Rahmen sofort. Er puffert alle korrekt erhaltenen Rahmen auch nach einem fehlerhaften Rahmen. Bei Ablauf des Timers überträgt der Sender nur den nicht bestätigten Rahmen neu.



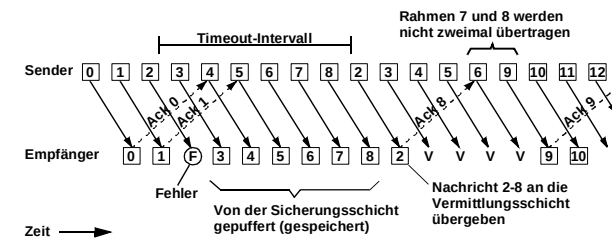
- jedes fehlerfreie Paket wird bestätigt. **Nachteil:** viele Bestätigungen
- **Vorteil:** keine unnötige Wiederholung korrekt empfangener Pakete.

206

- Jedes Paket muss quittiert werden; keine Sammelquittung möglich, da keine Pufferung
- **Vorteil:** Einsparen von Pufferplatz
- **Nachteil:** Ineffizient im Fehlerfall, da alle Rahmen während des time out-Intervalls neu übertragen werden müssen.

Fehlerbehebung durch „go-back-n“ mit Pufferung beim Empfänger

- Im Falle eines Fehlers bleibt das Ack aus. Der Empfänger puffert die danach noch korrekt erhaltenen Rahmen. Nach Ablauf des Timers beginnt der Sender, alle Rahmen ab dem Unbestätigten neu zu übertragen, bis er ein kumulatives Ack vom Empfänger erhält. Er macht dann weiter mit dem ersten noch nie übertragenen Rahmen.

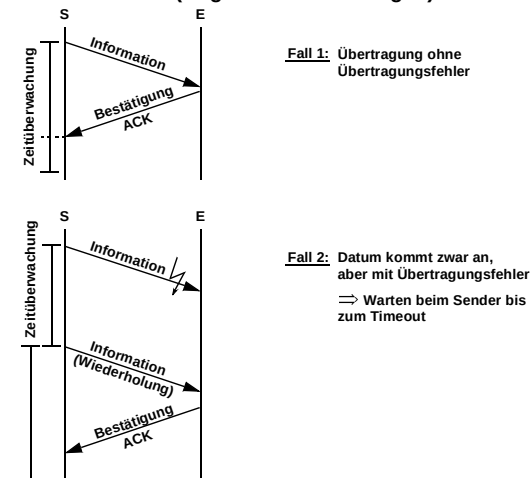


- Die erneute Übertragung von 7 und 8 kann durch Sammelacknowledge vermieden werden.

205

Bisher: Passive Fehlerkontrolle

bisher: Kein Versenden von NACKs (Negative Acknowledges)



207

Nachteil der passiven Fehlerkontrolle:

- Keine Unterscheidung zwischen fehlenden und fehlerhaften Blöcken
- Zeitverzögerung bis zur Wiederholung des Sendevorgangs = unnötige Wartezeit beim Sender im Fehlerfall (ineffizient)

208

NACK = Negative Acknowledgement

- In Rechnernetzen wird sowohl die aktive als auch die passive Fehlerkorrektur verwendet.
- Wenn ein Paket unterwegs verloren geht, enthält der Sender weder ACK noch NACK $\Rightarrow$ timeout des Senders wird aktiv.

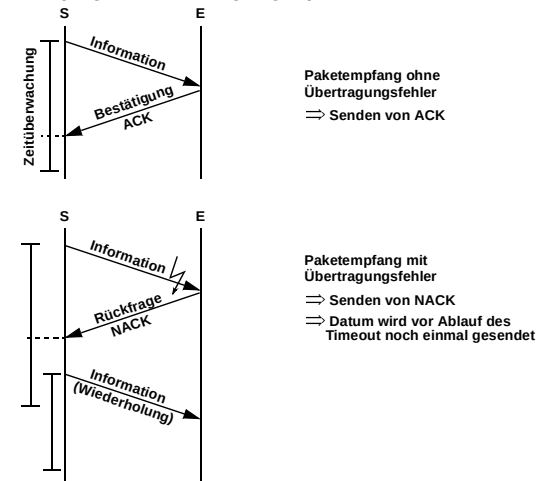
Flusssteuerung

- Die Flusssteuerung soll verhindern, dass der Sender den Empfänger mit Daten überschwemmt.
 - Für die Flusssteuerung wird oft kein eigener Mechanismus eingesetzt.
 - Vielmehr wird die Fehlererkennung über Acknowledges/Sequenznummer „zweckentfremdet“.
 - Acknowledges dienen dann zur Datensicherung und zur Flusssteuerung.
- $\Rightarrow$ Stop & Wait-Protokoll

210

Neu: Aktive Fehlerkontrolle

- Versenden von NACKs (negative Bestätigungen)



209

Stop-and-Wait-Protokoll zur Flusssteuerung benachbarter Rechner

Annahmen:

- Fehlerfreie Übertragung zwischen Sender und Empfänger.
 - Beschränkte Anzahl von Puffern (genau einer bei Sender und Empfänger).
 - Verarbeitungsgeschwindigkeiten können sich unterscheiden (z. B. schneller Großrechner mit langsameren PC).
 - Time out steht auf ausreichend hohem Wert.
- $\Rightarrow$ Benutze ACK für eine einfache Flusskontrolle ($\Rightarrow$ es ist nie mehr als ein Rahmen auf dem Weg).

ACK = Acknowledge = Bestätigung

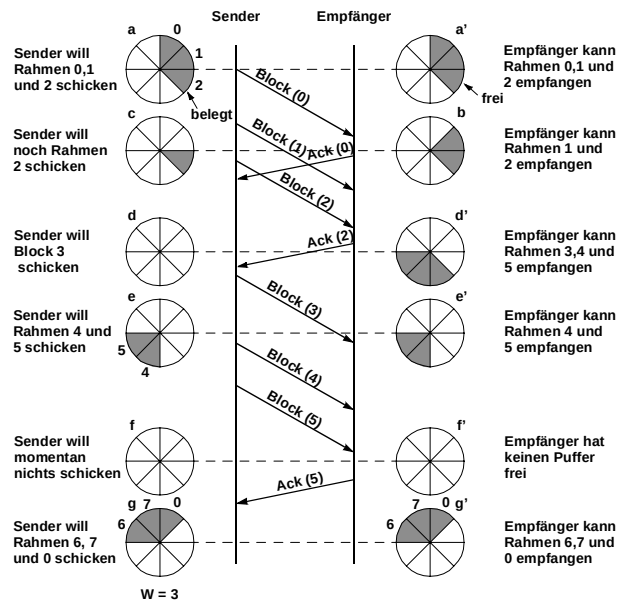
- Die Bestätigung wandert vom Empfänger zum Sender zurück.
 - Als Bestätigung dient entweder ein spezielles Paket, das Acknowledge-Paket, oder eine spezielle Steuerleitung, die Acknowledge-Leitung.
 - Immer, wenn der Sender vom Empfänger eine Empfangsbestätigung erhalten hat, weiss er, dass er ein neues Paket senden darf.
- $\Rightarrow$ Der Sender muss auf das Acknowledge warten = Stop & Wait-Protokoll.

211

Flusssteuerung mit Schiebefenster

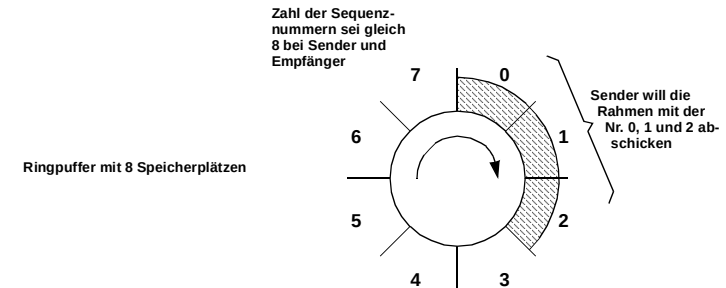
- Schiebefenster-Protokoll (sliding window) vermeidet Nachteile der vorangegangenen Protokolle und ist gleichzeitig effizient hinsichtlich der Bandbreiteausnutzung.
- TCP setzt das sliding window-Protokoll ein.
- Schiebefenster-Protokoll beruht auf einem Ringpuffer bei Sender und Empfänger.
- Fenstermechanismus:
 - Nach dem Verbindungsaufbau besitzt der Sender das Recht, so viele Informationsrahmen zu senden, wie durch die Fenstergröße vorgegeben ist ($\leq$ Zahl der Sequenznummern).
 - Spätestens dann muss vom Empfänger eine Bestätigung eintreffen, ansonsten unterbricht der Sender die Übertragung von Rahmen.
 - Der Empfänger kann schon vor dem Erreichen der Fenstergröße Bestätigungen an den Sender übermitteln (Öffnen des Fensters).

212



214

Beispiel: Fenstergröße = 3 $\Rightarrow$ Sender darf bis zu 3 Pakete ohne Bestätigung schicken.

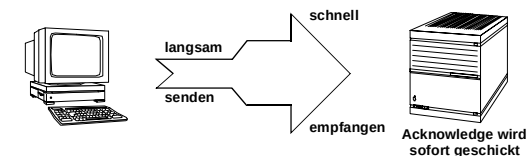


- Es gibt immer zwei Schiebefenster (eins beim Senden und Empfangen)
- Die Sequenznummern werden sowohl zur Fehlerkontrolle als auch zur Flusssteuerung verwendet.
- Fehlerkontrolle: „Sind alle Blöcke da?“
- Flusssteuerung: „Sind die letzten Blöcke zu schnell gekommen?“

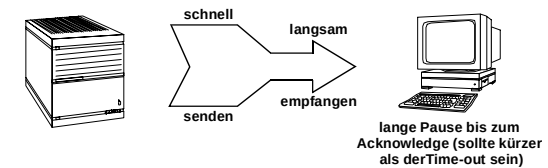
213

Wahl von Fenstergröße und Puffer beim Empfänger (1)

Beispiele für den Zusammenhang zwischen unterschiedlich leistungsfähigen Partnern, der Fenstergröße und der Anzahl zu reservierender Puffer beim Empfänger.



a) langsamer PC und schnelle Workstation: 1 Puffer ausreichend $\Rightarrow$ stop & wait-Protokoll mit window size = 1



215

b) schnelle Workstation und langsamer PC: 1 Puffer ausreichend $\Rightarrow$ stop & wait mit window size = 1

$\Rightarrow$ bei ungleichmäßig schnellen Kommunikationspartnern ist ein stop & wait-Protokoll ausreichend, sofern keine große Latenz zwischen beiden existiert.

• stop & wait-Protokoll heißt, dass ein Puffer der Größe 1 verwendet wird zusammen mit ACK.

Fenstergröße und Puffer beim Empfänger (2)

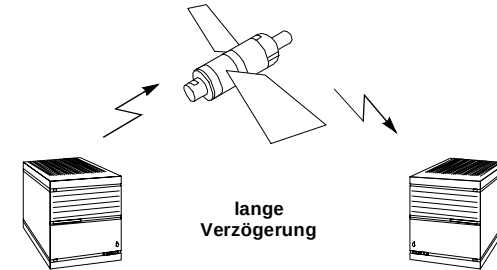
• Beide Rechner gleich schnell + kurzer Latenz, z.B. Kommunikation zwischen zwei Großrechnern: es sind w Puffer erforderlich mit $2 \leq w \leq 7$

• Beide Rechner gleich schnell + lange Latenz, z.B. Übertragung über einen Nachrichtensatelliten: es sind w Puffer erforderlich mit $16 \leq w \leq 127$

216

Lokale Netze (LANs)

218



- Je größer die Verzögerung bei der Übertragung, desto größer muss die Fenstergröße w gewählt werden, um die Latenzen des Protokolls auszugleichen, desto mehr Puffer werden auf der Empfängerseite benötigt.

- Bei Fenstergröße w benötigt der Empfänger mindestens w Pufferplätze. Idealerweise ist der Puffer bei Sender und Empfänger gleich groß, sonst wird Pufferplatz vergeudet.

217

Was ist ein LAN?

Ein LAN (Local Area Network) ist ein Netzwerk für (bitserielle) Übertragung von Informationen zwischen unabhängigen, untereinander verbundenen Stationen.

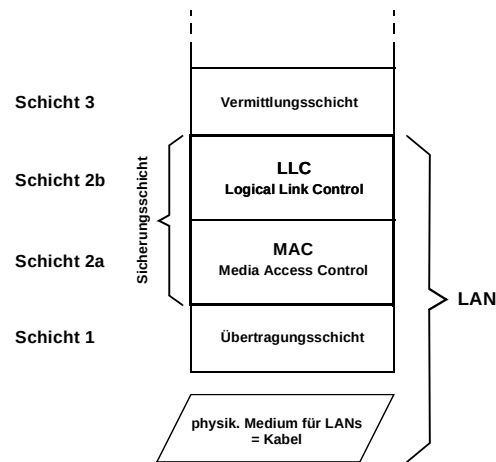
Es befindet sich rechtlich unter der Kontrolle des Benutzers und ist auf den Bereich innerhalb der Grundstücksgrenzen beschränkt.

Merkmale eines lokalen Netzes

- Hohe Geschwindigkeit (10-100 MBit/s - 1 GBit/s)
- Leichter, kostengünstiger Anschluss
- Keine Postregulierungen, da auf eigenem Grundstück
- Anschluss unterschiedlicher Geräte, wie
 - PCs
 - Workstations
 - Großrechner
 - Drucker und andere periphere Geräte
- Übergang auf Weitverkehrsnetze über Router und Gateways.

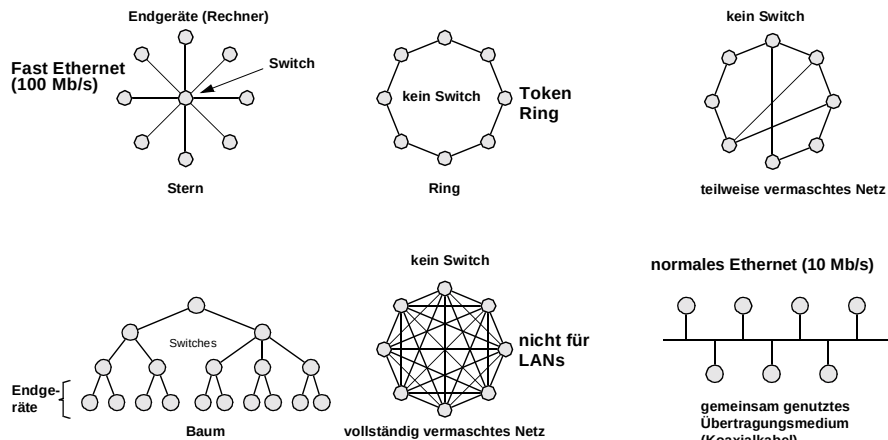
219

LANs im ISO-Referenzmodell



220

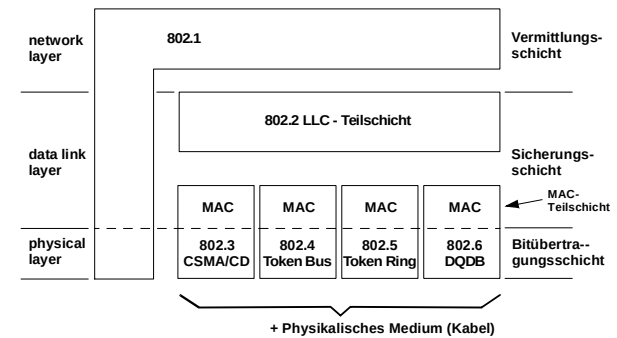
Topologien für lokale Netze



222

Die Sicherungsschicht im LAN nach IEEE 802

- Standardisierung bei IEEE und ISO



- 802.1 = Übersicht über die 802-Norm
- 802.2 = oberer Teil der Sicherungsschicht = Logical Link Control Protocol (LLC)
- 802.3 = Ethernet-Norm (CSMA/CD)
- MAC = unterer Teil der Sicherungsschicht = Medium Access Control

221

Schicht 2a bei LANs Zugang zum gemeinsamen Kabel

Medium Access Control (MAC) = Schicht 2a

Problem:

- Broadcast-Medium (gemeinsam genutzt) + Unabhängige Stationen fangen an gleichzeitig zu senden $\Rightarrow$ Sendekollisionen

Wie werden die Kollisionen geregelt?

„Medienzugangssteuerung“ (access control)

Zwei Medienbelegungsprinzipien:

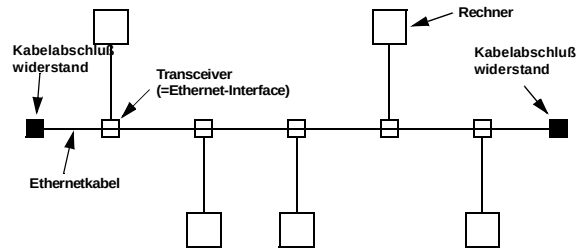
- 1.) **Kollisionsentdeckung** (so verfährt Ethernet)
Lasse Kollisionen stattfinden, entdecke sie, wiederhole Übertragung.
- 2.) **Kollisionsverhinderung** (so verfährt Token Ring)
Verwende ein zirkulierendes **Token (= Sendeberechtigungsrahmen)**, um den Zugriff auf das Medium zu steuern.

Token = „Zeichen“, ist bei Token Ring ein spezieller Rahmen, der den Zugriff auf das gemeinsame Kabel gewährt.

223

CSMA/CD: Topologie und Technologie

- CSMA/CD = Carrier Sense Multiple Access with Collision Detection = Träger abhören für mehrfachen Zugriff mit Kollisionserkennung
- Standard: ISO IS 8802/3: MAC- und Bitübertragungsschicht für CSMA/CD
- Topologie ist lineares, gemeinsames Medium (Koaxialkabel)



- Nachteil: Bei Kabelbruch ist das Netz als Ganzes nicht mehr funktionsfähig

224

CSMA/CD (Ethernet)

Verfahren: Die sendewillige Station hört das Medium ab („listen before talk“):

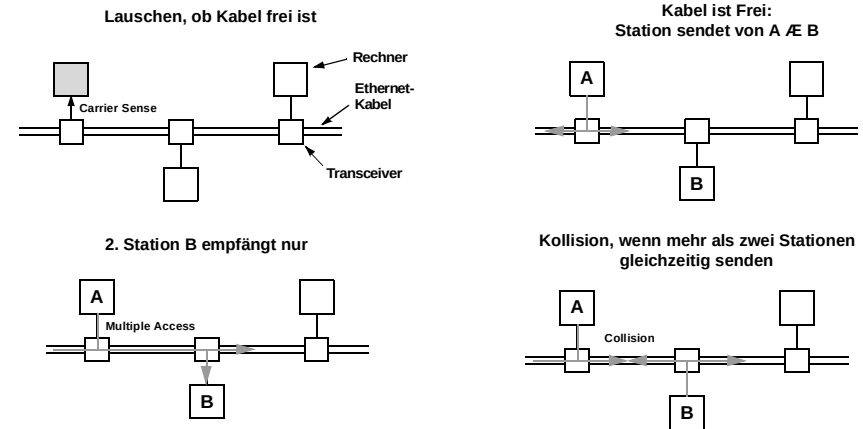
- Falls belegt, wird das Senden zurückgestellt
- Falls frei, wird mit dem Senden begonnen
- Zusätzlich gibt es Spezialregelung für sog. „Kollisionen“

Voraussetzung, dass CSMA funktioniert:

- 1.) Stationen können sich gegenseitig „hören“.
- 2.) Die Rahmenübertragungszeit muss größer sein als die doppelte maximale Laufzeitverzögerung im Kabel ($\geq 2T$, T-Laufzeit im Kabel von einem Ende des Kabels zum anderen)

226

CSMA/CD: Protokoll



225

Kollisionen

Beginnen zwei oder mehr Stationen gleichzeitig mit dem Senden, so tritt eine Kollision auf.

- Nachdem ein Sender eine Kollision erkannt hat, hört er auf zu senden und wartet eine Zeit ab, die von einer Zufallszahl bestimmt wird, bevor er wieder zu senden versucht.
 - Ein Sender kann mehrfach hintereinander das Schicksal haben, dass seine Sendung mit einer anderen Sendung kollidiert. (= mehrfache Kollision desselben Pakets)
 - Nach i Kollisionen ($1 \leq i \leq 10$) wird eine Zufallszahl aus dem Intervall $0, 1, 2, \dots, 2^i - 1$ gewählt.
 - Für $11 \leq i \leq 15$ wird die Zufallszahl aus dem konstanten Intervall $0, 1, 2, \dots, 2^{10} - 1$ gewählt.
 - Für $i = 16$ wird der Sendeversuch abgebrochen und eine Fehlermeldung erzeugt.
- ⇒ Das Intervall für die Zufallszahl wird nach jedem erfolglosen Senden verdoppelt, bis $i = 10$ erreicht ist.
- ⇒ Der Mittelwert der Wartezeit verdoppelt sich nach jeder Kollision des Senders. Dies wird als „exponential backoff“-Strategie bezeichnet.

227

Vermeiden von Kollisionen

Idee:

Erhöhte Kollisionsgefahr besteht nach Abschluss einer Übertragung: mehrere sendewillige wartende Stationen können bei der Feststellung „Medium frei“ gleichzeitig zu senden beginnen. Daher wird ein besonderer Algorithmus für das Sendebeginn-Verhalten nach Abschluss einer laufenden Sendung definiert (d. h., wenn das Medium vom Zustand „belegt“ in den Zustand „frei“ übergeht).

Bei Ethernet wird der relativ einfache „1-persistent-Algorithmus“ verwendet. Andere CSMA/CD-Netze können andere Strategien haben, z. B.

„p-persistent“ oder

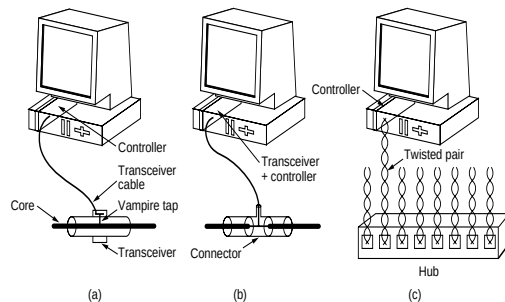
„non-persistent“

persistent = hartnäckig

- 1-persistent = hartnäckig mit Wahrscheinlichkeit = 1, d.h. Station hört das Medium ab und startet die eigene Übertragung sofort nach Abschluss der laufenden Übertragung (Sendewahrscheinlichkeit = 1)

228

Typen von 10Mb/s-Ethernet



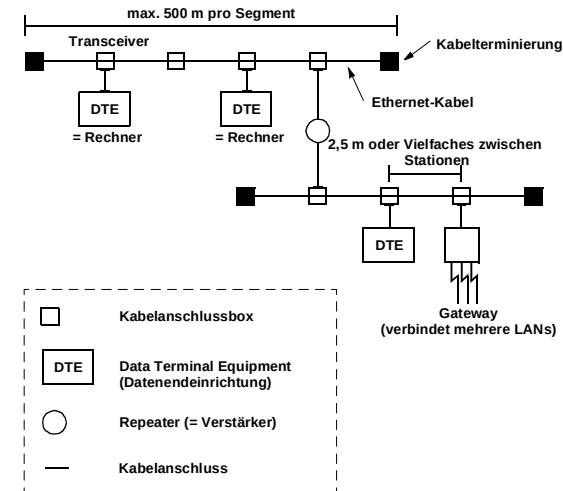
- a) Thick Ethernet (dickes gelbes Kabel, 10Base5)
- b) Thin Ethernet (Standard-Koaxialkabel, 10 Base2)
- c) Switched Ethernet (Twisted Pair, 10BaseT)

Hub = systemförmige Verbindung auf der Ebene 1 (keine Puffer, nur elektrische Trennung, keine Kollisionsvermeidung)

Switch = sternförmige Verbindung auf der Ebene 2 (Rahmen können gepuffert werden ⇒ keine Kollision an den Switch-Eingängen)

230

10 Mb/s-Ethernet-Beispiel



229

CSMA/CD: Rahmenformat (Ethernet)

7 Bytes	1	2/6	2/6	0-1500	4
Präambel	SD	DA	SA	INFO	FCS

Präambel = 7 Bytes (7 x 10101010)
SD = Starting Delimiter (1 Byte 10101011)
DA = Destination Address (2 oder 6 Bytes)
SA = Source Address (2 oder 6 Bytes)
Info = n Bytes = Datenteil $0 \leq n \leq 1500$
FCS = Frame Check Sequence = CRC (4 Bytes)

- Die Präambel dient zur Synchronisation des Empfängertaktes mit dem Sendetakt.
- Der Delimiter dient zur Kennzeichnung, wann ein Paket anfängt.
- Die Zieladresse dient zur Klärung, wohin ein Paket geschickt werden soll.
- Mindestlänge eines Rahmens muss 64 Bytes betragen.
- Falls Rahmen < 64 Bytes sein sollte, wird ein Füllfeld eingefügt (pad), so dass Länge = 64 wird.

231

CSMA/CD: Adressformat

Bei IEEE 802.x gibt es 2 Adressformate

1.) 48-Bit-Format = 6 Byte

I/G	U/L	46 Bits Adresse
-----	-----	-----------------

2.) 16-Bit-Format = 2 Byte

I/G	15 Bits Adresse
-----	-----------------

I/G= 0 individual address (Punkt-zu-Punkt-Verb.)
 I/G= 1 group address (Multicast-Verbindung)
 U/L= 0 globally administered address
 U/L= 1 locally administered address

- Bei 6 Byte-Adressen sind 46 Bit nutzbar = ca. $7 \cdot 10^{13}$ Adressen (Token Ring hat gleiches Adressformat!)
- Ethernet-Adressen sind weltweit eindeutig $\Rightarrow$ jede Ethernetkarte hat eine eindeutige Adresse.

232

FDDI (Fiber Distributed Data Interface)

- Nachfolger von Token Ring
- Motivation: Netzwerktechnik soll Leistungssteigerungen in der Rechnertechnologie folgen
- Sprung von der 10/16 Mbit/s-Geschwindigkeitsklasse (mittelschnelle LANs) auf 100 Mbit/s
- größere räumliche Ausdehnung (200 km)
- große Anzahl an Stationen (1.000)
- Topologie: Doppelring aus Fehlertoleranzgründen
- physikalisches Medium: Gradientenfaser oder Monomode-Faser, d. h. glasfaserbasierend
- Leuchtdioden mit Wellenlänge 1.300 nm als Sender
- Stecker mit je zwei Glasfasern (pro Punkt-zu-Punkt-Segment für Hin- und Rückrichtung)
- Medium Access Control über Token mit time-out-Überwachung
- Datenrate: 100 Mbit/s
- standardisiert durch ANSI und ISO

234

Stadtnetze (MANs)

MAN = Metropolitan Area Network

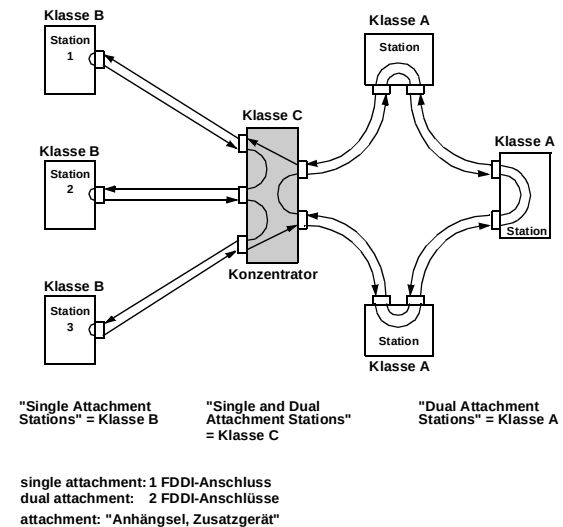
Beispiele für ein MAN: FDDI und DQDB

FDDI = Fibre distributed Data Interface
 = glasfaserbasiertes MAN mit 100 Mb/s

DQDB = Distributed Queue Dual Bus
 = MAN mit 34 Mb/s

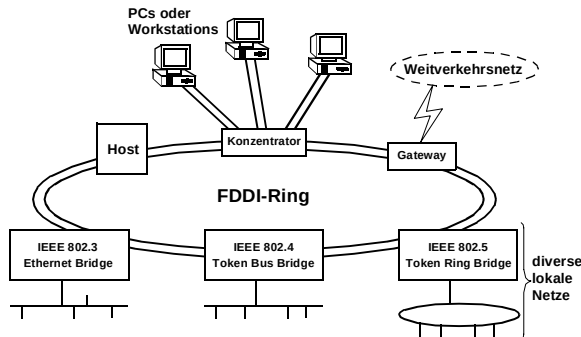
233

Beispiel für eine FDDI-Topologie



235

Typische FDDI-Konfiguration (FDDI als Backbone)



Bridge = Gerät, das zwei LANs auf der Ebene 2a der Sicherungsschicht koppelt. Gateway = Gerät, das LAN oder MAN mit einem WAN verbindet (Schicht 3-Kopplung)

Backbone: Rückgrat

236

Schicht 2 b der Sicherungsschicht

Macht die „Logical Link Control“ im LAN (= LLC)

- Die Sicherungsschicht in diesen LANs bzw. MANs ist identisch für CSMA/CD, Token Ring, Token Bus, FDDI
- Drei verschiedene Dienste werden von LLC unterstützt LLC Typ 1: Unbestätigter verbindungsloser Dienst
 - Unbestätigte Übertragung (kein Acknowledge) von Datenrahmen (= Datagramm)
 - Höhere Schichten sind für die Erhaltung der Reihenfolge, Fehlerbehebung und Flusssteuerung verantwortlich (passiert in Schicht 4, Transportschicht)

LLC Typ 2: Verbindungsorientierter Dienst (analog wie bei HDLC)

- Verbindungsauf- und abbau
- Datenübertragung mit Bestätigung (ACK)
- Garantierte Ablieferung beim Empfänger
- Garantierte Reihenfolge der Rahmen
- Flusssteuerung

LLC Typ 3: Bestätigter verbindungsloser Dienst

- Auf jedes Datagramm kann genau eine Bestätigung folgen (zwar Ack aber keine garantierte Reihenfolge)
- Anwendung in der Fertigungssteuerung zur Fabrikautomation (Steuerung von Bearbeitungsmaschinen an einem Fließband, z. B. in der Automobilproduktion.)

238

Vergleich Ethernet, Token Ring und FDDI

CSMA/CD (Ethernet): LAN

- Sehr gute Performance bei geringer bis mäßiger Belastung (viele Kollisionen bei hoher Belastung)
- Einfaches Protokoll, leicht implementierbar (z. B. kein Monitor notwendig)
- Topologische Einschränkungen bei der Bus-Topologie (2,5 m Raster)
- Keine Prioritätssteuerung
- Keine maximale Verzögerung garantierbar (für Echtzeit ungeeignet)

Token Ring: LAN

- Hohe Systemausnutzung (überträgt viel Verkehr effizient)
- Fehlerbehebung („Token Recovery“) aufwendig
- Flexible Verkabelung durch wire center
- Prioritäten möglich
- Maximale Verzögerung garantierbar (außer beim Auftreten von Fehlern) ⇒ deterministisch, für Echtzeit geeignet

FDDI: LAN/MAN

- Eine schnelle Variante des Token Rings
- Kombiniert synchronen und asynchronen Verkehr auf dem selben Medium

237

Weitverkehrsnetze (WANs)

Wan = Wide Area Network

239

Unterschied zwischen LANs/MANs und Weitverkehrsnetzen (WANs)

- LANs und MANs bestehen aus einem gemeinsamen Übertragungsmedium und den daran angeschlossenen Rechnern (sog. Hosts)
 - ⇒ LANs und MANs sind Broadcast-Netze auf der Schicht 1. Eine Wegewahl (Routing) ist nicht erforderlich, da jede Nachricht bei allen Hosts vorbeikommt.
- WANs bestehen aus mind. 1 Weitverkehrsstrecke sowie aus einem oder mehreren LANs und/oder MANs am Anfang und Ende der Weitverkehrsstrecke.
- Beispiel einer Weitverkehrsstrecke ist eine ATM-Verbindung, die über eine SONET-Glasfaserstrecke implementiert ist.
- ATM = Asynchronous Transfer Mode = Protokoll für den unteren Teil der Schichten 2-7 (umfasst ≈ Schicht 2-4)
- SONET = Synchronous Optical Network = eine Netztechnik der Schicht 1.
- Es gilt: alle Weitverkehrsstrecken sind sog. Punkt-zu-Punkt-Verbindungen.
- An den Endpunkten einer Weitverkehrsstrecke sitzen spezielle Hosts, sog. Router.
- Router dienen auch zur Kopplung von LANs und MANs.
- Es gilt: jeder Router ist mit einem Nachbar-Router über eine dedizierte Strecke verbunden. Diese Strecke kann ein LAN, MAN oder eine Weitverkehrsstrecke sein.

240

Zusammenfassung der Unterschiede zwischen Punkt-zu-Punkt-Netzen und Broadcast-Netzen

Broadcast-Netz = LAN, MAN

- Mehrere Stationen teilen sich das physikalische Medium.
- Alle Stationen hören alle Nachrichten.
- Falls zwei Stationen gleichzeitig senden, werden beide Nachrichten verfälscht bzw. zerstört.
- Der Sender kann seine eigene Nachricht hören. Falls er genau das hört, was er gesendet hat, kann er annehmen, dass auch der Empfänger die Nachricht korrekt empfangen hat.
- Innerhalb eines LAN-Segments ist eine Wegewahl nicht erforderlich.

Punkt-zu-Punkt-Netz = WAN

- Jeweils genau zwei Router sind physikalisch verbunden.
- Multicast und Broadcast erfordern die explizite Vervielfältigung der Nachrichten in den Zwischenknoten.
- Im Weitverkehrsnetz (= teilweise vermaschter Graph) muss eine explizite Wegewahl erfolgen, um einen bestimmten Empfänger zu erreichen.

242

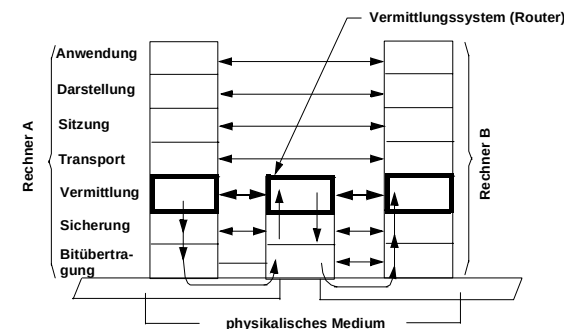
- bei WANs ist ein sendender Host i. A. nicht im selben LAN oder MAN wie der empfangende Host.
 - ⇒ Auf seinem Weg durch das WAN durchläuft ein Paket i. A. mehrere Netze (LANs, MANs und Weitverkehrsstrecken)
 - ⇒ Bei WANs ist eine Wegewahl (Routing) erforderlich. Die Wegewahl wählt einen Pfad durch das WAN. Sie wird von der OSI-Schicht 3 (Vermittlungsschicht) vorgenommen.
 - ⇒ Router sind Rechner oder spezielle Geräte, die auf der Schicht 3 arbeiten.

Die Vermittlungsschicht (Schicht 3)

- nötig, wenn mehrere LANs, MANs oder Weitverkehrsstrecken über Router gekoppelt werden.
- dient zur Wegewahl
- Die Vermittlungsschicht transportiert ein Datenpaket, das von einem Host erzeugt wird, über eine Folge von Routern zum empfangenen Host.
- Die Folge von Routern heißt Weg, Pfad oder Route (Leitweg) durch das Netz.
- Das Routing erfolgt mit Methoden der Graphentheorie. Dabei wird das Netz als Graph dargestellt. Knoten des Graphen sind die Router. Kanten des Graphen sind LANs, MANs und Weitverkehrsstrecken.

241

Die Vermittlungsschicht im Referenzmodell nach OSI



- In den Netzzwischenknoten (Routern) müssen Pakete nur bis Schicht 3 (Vermittlungsschicht) wandern, in den Endsystemen hingegen durch alle Schichten

243

Zusammenfassende Liste der Aufgaben der Vermittlungsschicht (nur ISO spezifiziert, nicht in Internet IP realisiert)

- Wegewahl (= „Vermittlung“ bzw. Routing)
- Verbindungsaufbau und -abbau von sog. „virtuellen Verbindungen“
- Multiplexen von virtuellen Verbindungen
- Segmentierung von langen Paketen in kurze und Zusammenstellung kurzer Pakete zu langen
- Fehlererkennung/Fehlerbehebung (Ende-zu-Ende)
- Sicherstellung der Paketreihenfolge (aufsteigende Sequenznummer)
- Flusssteuerung (Ende-zu-Ende) = Überlaststeuerung einzelner Hosts
- Überlaststeuerung der Teilnetze

Ziel: Eine möglichst große Vielfalt von Netzen und Technologien soll unterstützt werden.

Multiplexen = eine Leitung dient nacheinander der Übertragung mehrerer, gleichzeitig stattfindender Kommunikationen

244

Datagramme

Jedes Paket (Datagramm) wird als isolierte Einheit betrachtet (keine virtuelle Verbindung)

- Volle Zieladresse in jedem Paket notwendig
- Pakete können außerhalb der Reihenfolge eintreffen
- Keine Fehlerüberwachung, keine Flusssteuerung

Vorteile von Datagrammen:

- Primitiver als virtuelle Verbindungen, daher einfacher zu implementieren.
- Kein Verbindungsauf- und -abbau, deshalb niedriger Overhead für kurzlebige Verbindungen.
- flexibler und zuverlässiger bei Routerausfällen, da mehr als ein Weg zum Ziel führt.
- besser geeignet für heterogene Subnetze, da virtuelle Verbindung zwischen Netzen verschiedener Typen technisch aufwendig ist.

Bemerkung:

Im Internet werden sowohl Datagramme als auch virtuelle Verbindungen benutzt; von IP Datagramme, von TCP virtuelle Verbindungen. (TCP ist ein Protokoll der Transportschicht.)

246

Datentransfer über Virtuelle Verbindungen oder Datagramme

- Definition Virtuelle Verbindung:

Der Weg durch das Netz wird vor dem ersten Datentransfer vollständig festgelegt. Dieser Weg heißt virtuelle Verbindung. In jedem Zwischenknoten findet nur einmal eine Wegewahlentscheidung statt. Der ganze über diese virtuelle Verbindung fließende Verkehr nimmt denselben Weg durch das Netz.

- Definition Datagramm:

Die Paketadresse bestimmt in jedem Netzknoten auf dem Pfad die abgehende Leitung. Für jedes Datagramm wird in jedem Knoten erneut eine Wegewahlentscheidung getroffen.

- Bemerkung

Eine virtuelle Verbindung zwischen zwei Endsystemen (Hosts) tut so, als ob die beiden Rechner direkt miteinander verbunden wären. Da sie es aber i. a. nicht sind, spricht man von einer virtuellen Verbindung.

245

Virtuelle Verbindungen zwischen Rechnern

Eine virtuelle Verbindung hat die zusätzliche Eigenschaft, ein „perfekter“ Kanal durch das Netz zu sein:

- Ordnung der Nachrichten (Sicherstellung der Reihenfolge) ist gewährleistet.
- Fehlerüberwachung (verlorene und duplizierte Pakete) finden statt.
- Flusssteuerung wird durchgeführt (Ende-zu-Ende)

Drei Phasen in einer virtuellen Verbindung

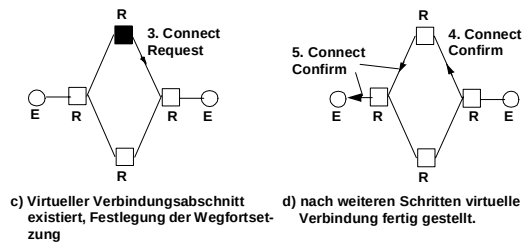
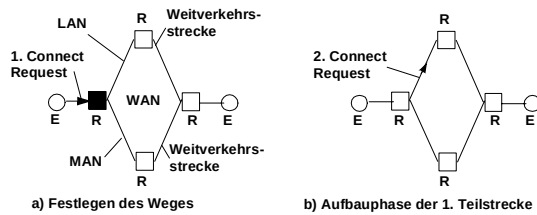
- Verbindungsaufbau
- Datenübertragung
- Verbindungsabbau

Zwei Vorteile von virtuellen Verbindungen

- Kein Aufwand für die Wegewahl während der Datenübertragung
- Keine Neusortierung der Pakete oder Fehlerüberwachung im Endsystem nötig (Transportschicht wird entlastet)

247

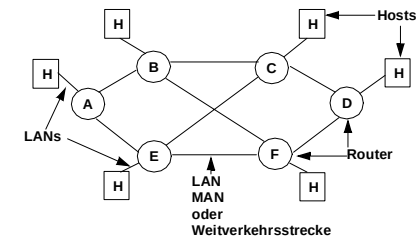
Aufbau einer virtuellen Verbindung in einem Beispiel WAN



E = Endsystem = Rechner (Host)
R = Router = spezieller Host

248

Virtuelle Verbindungen in einem Beispiel-WAN



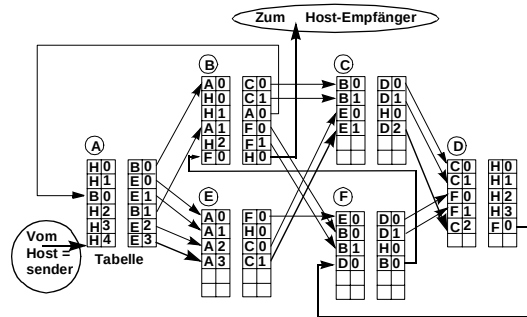
- Beispiel für acht virtuelle Verbindungen durch das WAN. In jedem Router werden Tabellen mit Zustandsinformationen über durchlaufende virtuelle Verbindungen verwaltet.

Ausgehend von A	Ausgehend von B
0 - ABCD	0 - BCD
1 - AEFD	1 - BAE
2 - ABFD	2 - BF
3 - AEC	
4 - AECDDB	

↑
Nummer der virtuellen Verbindung

249

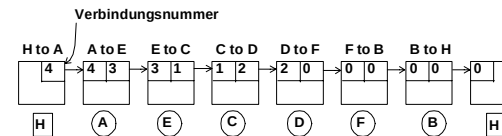
Beispiel für die Zustandsinformation in den Routern



- In jedem Router gibt es eine Tabelle zur Verwaltung der virtuellen Verbindungen
- Jede ankommende und abgehende virtuelle Verbindung erhält eine Nummer
- Beispiel H 4 E 3 heißt:
Vom Host kommt eine Verbindung mit der Nr. 4. Diese wird unter der Nr. 3 an den Router E weitergereicht.
- Dieselbe virtuelle Verbindung wird von Router zu Router unter einer anderen Nummer geführt.

250

Hier die Beispielverbindung HAECDDB H



- Der Vorteil ist, dass jeder Router seine Tabelle selbständig verwalten kann, ohne dass netzweit eindeutige Verbindungsnummern vergeben werden müssen.
- Nur zwischen benachbarten Routern muss für die selbe Verbindung eine Nummer verwendet werden.

251

Wegewahl für virtuelle Verbindungen und Datagramme

Routing-Algorithmen

Aufgabe von Routing-Algorithmen:

Leitwegbestimmung für Pakete durch das Netzwerk vom Quellsystem zum Zielsystem (Ende-zu-Ende)

Der Leitwegbestimmungsalgorithmus eines Vermittlungsrechners (Knotens) entscheidet, auf welcher Ausgangsleitung ein eingegangenes Paket weitergeleitet wird.

Wünschenswerte Eigenschaften eines Wegewahl-Algorithmus:

- Korrekt (Paket findet immer zum Ziel)
- Einfach (schnell auszuführen)
- Robust (bei Rechner- oder Leitungsausfällen)
- Stabil (Gleichmäßige Ergebnisse)
- Fair (kein Paket wird benachteiligt)
- Optimal (findet den schnellsten oder kürzesten Weg)

252

Verfahren zur Leitwegbestimmung

Klassifikation der Verfahren:

1. Statische (nicht adaptive) Verfahren

- keine Berücksichtigung des aktuellen Netzzustands
- gehen von Mittelwerten aus, nicht von lokalen Informationen
- Leitweg zwischen Knoten i und j wird für alle i, j vor der Inbetriebnahme des Netzwerks bestimmt
- keine Änderung während des Betriebs (=statisches Routing)

2. Adaptive Verfahren

- Entscheidungen basieren auf aktuellem Netzzustand
- Messungen/Schätzungen der Topologie und des Verkehrsaufkommens als Input für den Algorithmus
- Weitere Unterteilung der adaptiven Verfahren in:
 - zentralisierte Verfahren
 - lokale Verfahren
 - verteilte Verfahren

254

Algorithmen für die Leitwegbestimmung

Optimierungskriterien des Algorithmus (Beispiel):

- Durchschnittliche Paketverzögerung (Latenz)
- Gesamtdurchsatz des Netzes (Durchsatz)

Dies ist ein sich widersprechender Zielkonflikt, daher gebräuchlich:

- Minimierung der Teilstrecken (hops) pro Paket (= Kompromiss)

Vorteil:

- reduzierte Verzögerung (jedoch nicht in allen Fällen)
- vermindert benötigte Bandbreite
- steigert Durchsatz, allerdings i. Allg. nicht bis zum Maximum

Bemerkung:

In einem Netz können Latenz und Durchsatz nicht gleichzeitig optimiert werden, weil hoher Durchsatz nur durch eine hohe Latenz erreichbar ist (gemäß Warteschlangentheorie).

253

A) Statische Leitwegbestimmung

Beim statischen Routing ist die gesamte Topologie des Netzes einer zentralen Stelle bekannt. Sie berechnet die optimalen Pfade für jedes Paar (i, j) von Knoten, erstellt daraus die Routing-Tabelle für die einzelnen Knoten und versendet diese (= Routing über statische Tabellen).

aber: Nur sinnvoll, wenn das Netz relativ klein und relativ statisch ist.

Eigenschaften:

- optimale Pfade (i, j) für alle i, j sind
 - 1.) kürzeste Pfade *oder*
 - 2.) schnellste Pfade
- die kürzesten Pfade werden zentral über den Graphenalgorithmus „Spannbaum“ bestimmt bzw. über minimalen Spannbaum
- Das bekannteste Verfahren zur Berechnung eines minimalen Spannbaums ist das von Dijkstra
- Definition Spannbaum :
gegeben Graph $G = (V, E)$ dann ist $G' = (V, E')$ ein Spannbaum von G, wenn $G' = \text{Baum} \wedge E'$ erreicht alle Knoten von G gilt.

255

Zusammenfassung statisches Routing

- Eigenschaften:

Tabellen werden vom Netzwerkoperator zentral erstellt; Tabellen werden vor Inbetriebnahme des Netzes in die Knoten geladen und dann nicht mehr verändert.

- Vorteile:

- einfach zu implementieren
- gute Ergebnisse bei relativ konstanter Topologie und konstantem Verkehr

- Nachteile:

- schlecht bei stark variierendem Verkehrsaufkommen und bei Topologieänderungen
- schlecht bei großen Netzen (skaliert nicht)

256

Realisierung der alternativen Wegewahl

- Jeder Knoten enthält Routing-Tabelle mit je einer Zeile für jeden möglichen Zielknoten

eine Zeile

Ziel-knoten	A1	G1	A2	G2	...	An	Gn
-------------	----	----	----	----	-----	----	----

- A_i : i-beste Ausgangsleitung des jeweiligen Zwischenknotens

- G_i : Gewicht für A_i (drückt die Güte der Leitung aus)

- G_i bestimmt die Wahrscheinlichkeit, mit der A_i benutzt wird: $\sum_{i=1}^n G_i = 1$

Leitung hoher Güte wird häufig benutzt.

- Die Gewichte der Ausgangsleitungen werden gemäß der gewählten Metrik (schnellsten, kürzesten,...) bestimmt.

258

Verfeinerungen von Wegewahlalgorithmen

- Sowohl bei statischem als auch bei dynamischem (= adaptivem) Routing können 2 Verfeinerungen angebracht werden:

a) Mehrfachpfade ($\Rightarrow$ alternative Wege)

b) hierarchische Wegewahl ($\Rightarrow$ Vereinfachung der Verfahren)

Mehrfach-Pfade (Multipath routing)

Prinzip: Benutzung alternativer Leitwege zwischen jedem Knotenpaar (i, j)

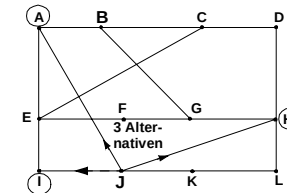
Vorteile

- alternative Wege können gemäß ihrer „Güte“ ausgewählt werden („gute“ Wege häufig, schlechte seltener)
- Höherer Durchsatz durch Verteilung des Datenverkehrs auf mehrere Pfade
- Höhere Zuverlässigkeit, da der Ausfall eines Links nicht so schnell zur Unerreichbarkeit von Knoten führt

Definition: guter Weg = schnellster/kürzester/geringste Warteschlange/höchste Bandbreite/höchste Zuverlässigkeit

257

- Beispiel: gegeben sei Ziel B und Quelle J mit $G_1 = 0,46$; $G_2 = 0,31$; $G_3 = 0,23$



- Gewichte werden auf dem Zahlenstrahl von 0 bis 1 aufgereiht (z. B. links das höchste, rechts das niedrigste; genaue Reihenfolge ist jedoch ohne Belang)



- Zufallsauswahl der alternativen Ausgangsleitung gemäß ihres Gewichts (= Güte)

- Generieren einer Zufallszahl z ($0 \leq z \leq 1$)

- Wähle A_1 , falls $0 \leq z < G_1$, Wähle A_2 , falls $G_1 \leq z < G_1 + G_2$, u.s.w.

- Jeder Weg wird entsprechend seines Gütemaßes mehr oder weniger häufig benutzt.

259

Zweite Verfeinerung bei allen Routing-Algorithmen

Hierarchische Leitwegbestimmung

Die Größe der Routing-Tabellen ist proportional zur Größe des Netzwerks:

- großer Speicherbedarf
- viel CPU-Zeit zum Durchsuchen der Tabellen
- viel Bandbreite zum Austausch von Routinginformation

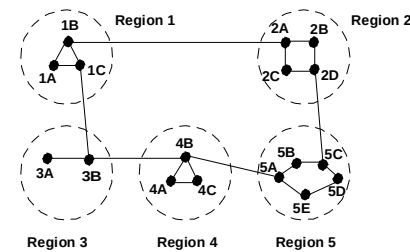
Hierarchische Leitwegbestimmung ab einer bestimmten Netzgröße notwendig:

- Knoten werden in Regionen gruppiert
- Jeder Knoten kennt
 - Details seiner Region
 - Leitweg zu allen anderen Regionen
 - aber nicht den Weg zu den Knoten außerhalb der eigenen Region, nur in welcher Region der Zielknoten liegt

Nachteil: nicht immer optimale Entscheidungen möglich

260

Hierarchische Wegewahl Beispiel



Hierarchische Tabelle für 1A

Ziel	Leitung	Teilstrecken
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

⇒ hierarchische Tabelle ist viel kürzer

Volle Tabelle für 1A

Ziel	Leitung	Teilstrecken
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	4
4B	1C	3
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

261

B) Adaptive Wegewahl

1.) Zentrale Verfahren

2.) Lokale Verfahren

3.) Verteilte Verfahren

zentral heißt: einer führt den Algorithmus aus (= zentrale Instanz)

lokal heißt: alle Router führen den Algorithmus aus, aber ohne Kommunikation (= isoliert)

verteilt heißt: alle Router führen den Algorithmus mit Kommunikation, d. h. Austausch von Information, aus.

B.3. Adaptive + Verteilte Wegewahl

Folgende drei Methoden werden bzw. wurden im Internet angewandt:

1.) „Distanz“vektor-Verfahren

2.) Link-Zustandsverfahren

3.) OSPF-Verfahren

262

B. 3. 1. Das „Distanz“vektor-Verfahren

Prinzip: Die Knoten tauschen mit ihren Nachbarn Leitweginformationen aus:

- Jeder Knoten kennt „Distanz“ (=Güte) zu jedem Nachbarn, z. B.:

- Anzahl der Teilstrecken
- Verzögerungszeit (wird über sog. Echopakete gemessen)
- Warteschlangenlänge etc.

- Jeder Knoten sendet periodisch seinen Nachbarn eine Liste mit seinen gemessenen/geschätzten Güten zu jedem Ziel

z. B.: X empfängt Liste vom Nachbarn Y
sei Entfernung (X, Y) = e und Z ein beliebiger Knoten
und Entfernung (Y, Z) = E(Z) (E = Listeninformation)

⇒ Entfernung (X, Z) über Y: E(Z) + e

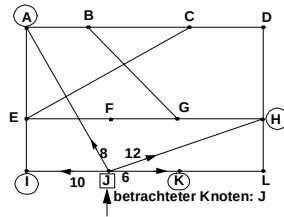
⇒ nach jeder Periode konvergiert die Tabelle etwas näher zur Wahrheit hin (iteratives Verfahren)

Problem:

Der Ausfall eines Routers wird nur sehr langsam erkannt (= count-to-infinity-Problem), da Ausbreitung dieser Information im Netz viele Iterationen erfordert.

263

Beispiel für das „Distanz“vektor-Verfahren



- J misst die Latenz zu seinen Nachbarn

JA-Verzögerung = 8

JI-Verzögerung = 10

JH-Verzögerung = 12

JK-Verzögerung = 6

- von seinen Nachbarn empfängt J die Latenzen zu allen anderen Knoten (vom jeweiligen Nachbarknoten aus gesehen)
- Aus beiden Informationen baut J seine lokale Latenztafel auf
- Zuletzt verteilt J seine lokale Tabelle an seine Nachbarn
- Vorgang wird periodisch wiederholt => es bildet sich ein eingeschwungener Zustand heraus
- Tabellen, die J empfängt:

264

- Ergebnis:

Ziel	Latenz	über
A	8	A
B	20	A
C	28	I
D	20	H
E	17	I
F	30	I

Ziel	Latenz	über
G	18	H
H	12	H
I	10	I
J	0	-
K	6	K
L	15	K

266

Ziel	von A	von I	von H	von K
A	0	24	20	21
B	12	36	31	28
C	25	18	19	36
D	40	27	8	24
E	14	7	30	22
F	23	20	19	40
G	18	31	6	31
H	17	20	0	19
I	21	0	14	22
J	9	11	7	10
K	24	22	22	0
L	29	33	9	9

265

- Beachte: die Latenzen $\overline{AJ}$, $\overline{IJ}$, $\overline{HJ}$ und $\overline{KJ}$, die die Nachbarknoten sehen, sind anders, als die, die J sieht: $\overline{AJ} \neq \overline{JA}$ usw.
- J wählt aus den eigenen Messungen und den empfangenen Latenzen den besten Weg aus ($\overline{JA} = 8$, $\overline{JI} = 10$, $\overline{JH} = 12$, $\overline{JK} = 6$)

B. 3.2 Das Link-Zustandsverfahren

- Prinzip:

- Jeder Router in einem Subnetz ist im Vollbesitz aller „Güte“-Werte des Subnetzes
- Dazu sendet jeder Router Datenpakete an alle anderen Router im Subnetz, nicht nur an seine Nachbarn
- Entsprechend empfängt jeder Router Pakete von allen anderen Routern
- Dadurch kann jeder Router lokal die Topologie des ganzen Subnetzes rekonstruieren
- Als Gütewerte werden die üblichen Metriken (Distanz, Latenz,...) verwendet
- Mit Hilfe der Topologie und der Güte jeder Verbindung kann jeder Router für sich einen minimalen Spannbaum aufbauen

267

B. 3. 3 Das OSPF-Verfahren

OSPF = open „shortest path first“

Prinzip:

- Wie das Link-Zustandsverfahren, aber mit einer Reihe von Erweiterungen
- 1.) unterstützt mehrere Metriken (kürzeste, schnellste Wege etc.)
- 2.) berücksichtigt alternative Wege
- 3.) unterstützt hierarchisches Routing (Link-Zustandsverfahren in jeder Region)
- 4.) ist sicherer bezügl. böswilliger Manipulation der Routing-Tabellen

Eigenschaften:

- Spannbaum einer Routing-Region wird gem. Dijkstra berechnet (von jedem Knoten in der Region)
- Topologie und Güte jeder Wegstrecke (= Kanten des Graphen) werden in einer Datenbank gehalten (nicht in einer Tabelle), und zwar in jedem Knoten für sich

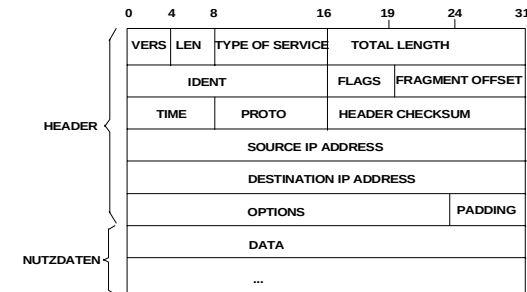
268

VERS	4b	Protokollversion von IP z. Z. IP V. 4
LEN	4b	Länge des Headers (Wörter)
TYPE OF SERVICE	8b	QoS Priorität und D/T/R (Delay, Throughput, Reliability) nur theoretisch
TOTAL LENGTH	16b	Länge incl. Nutzdaten in Bytes
IDENT	16b	Identität des Datagramms (zu welchem Paket nach Fragmentierung gehört)
FLAGS	3b	nicht fragmentieren/letztes Fragment
FRAGMENTOFFSET	13b	Offset dieses Teils innerhalb von IDENT Byte-Portionen
TIME	8b	Lebensdauer in Sekunden („time to live“)
PROTO	8b	Typ des höheren Protokolls (UDP, TCP)
HEADER CHECKSUM	16b	EXOR der Header-Wörter
SOURCE ADDRESS	32b	IP Adresse des Quell-Hosts
DEST ADDRESS	32b	IP-Adresse des Ziel-Hosts
OPTIONS		Kommandocode für Netzmanagement
PADDING		Auffüllen auf Wortgrenzen
DATA		Nutzdatenfeld

270

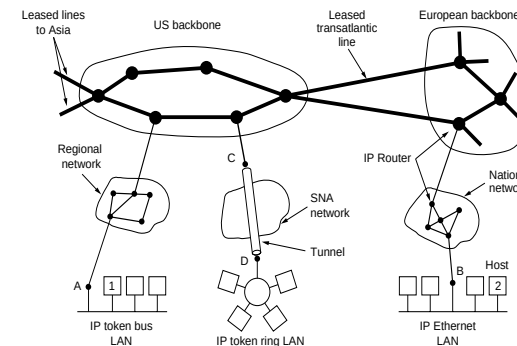
IP (Internet Protocol) = Protokoll der Vermittlungsschicht

- Ein Datagramm-Protokoll für Ende-zu-Ende-Kommunikation
- Macht die Wegewahl (Routing)
- Verbindungsloser Dienst, daher unzuverlässig (schützt nicht vor Paketverlusten)
- Fragmentiert große Pakete in kleinere.
- IP-Paketlänge: maximal 64 KB
- Format von IP:



269

Wie funktioniert die Adressierung im Internet?



- Das Internet besteht aus einer riesigen Zahl von Netzen, die durch IP adressiert werden
- Adressiert wird über die IP-Adressen (= 32 Bit-Feld „Destination address“)
- Tunnel: Verfahren, um IP-Pakete durch inkompatible Netze zu routen wie z.B. SNA. IP-Paket wird dabei als Nutzlast der Schicht 7 im Fremdnetz transportiert
- SNA (System Network Architecture): IBM-eigenes Netz, das inkompatibel zum Internet ist

271

IP-Adressen

- Die IP-Adresse ist eine hierarchische Adresse mit Netz- und Hostidentifikationsnummer (netid und hostid). Es gibt drei Formate für Subnetze unterschiedlicher Größe sowie ein Format für Multicast:

		Typ	Netzadr.	Hostadresse		
			0	123...78	16	24 31
3 Formate	CLASS A	0	netid	hostid		
	CLASS B	0 10	2	netid	15 16	31
	CLASS C	0 110	3	netid	23 24	31
nur für Multicast	CLASS D	0 1110	4	group address		

Gebräuchlich ist eine dezimale Schreibweise mit einer Zahl pro Byte. Beispiel:

10.0.0.0 für Arpanet (class A)
 128.10.0.0 für ein großes Ethernet-LAN (class B)
 192.5.48.0 für ein kleines Ring-LAN (class C)

272

o Bei den Adreßbereichen gibt es drei Adressen mit spezieller Bedeutung:

1) der Wert 0 bezeichnet das eigene Netz bzw. den eigenen Host

- o Wenn Netzadresse = 0, dann eigenes Netz
- o ((Wenn Netzadresse = eigenes Netz) oder (Wenn Netzadresse = 0)) und wenn Hostadresse = 0, dann eigener Host

2) der Wert -1 (=alle Bits gesetzt) sagt, daß ein Broadcast durchgeführt werden soll

- o ((Wenn Netzadresse = eigenes Netz) oder (Wenn Netzadresse = 0)) und wenn Hostadresse = -1, dann Broadcast an alle Hosts im eigenen Netz
- o Wenn Netzadresse = x und wenn Hostadresse = -1, dann Broadcast an alle Hosts im Netz x

3) der Wert 127 als Netzadresse sagt, daß ein Loopback-Test durchgeführt werden soll (Loopback = Host sendet an sich selber)

Wenn Netzadresse = 127, dann Loopback

274

IP-Adreßbereiche

- o Die Internet-Adreßbereiche werden zentral vom Network Information Center (NIC) vergeben (<http://www.nic.com>)

Es gibt die folgenden Adreßbereiche:

Klasse-A-Netze: 1.0.0.0 bis 127.255.255.255
 Klasse-B-Netze: 128.0.0.0 bis 191.255.255.255
 Klasse-C-Netze: 192.0.0.0 bis 223.255.255.255
 Multicast: 224.0.0.0 bis 239.255.255.255
 reserviert: 240.0.0.0 bis 247.255.255.255
 unbelegt: 248.0.0.0 bis 255.255.255.255

Es gibt:

- o 126 Klasse-A-Netze, mit jeweils bis zu 16M-2 Hosts
- o 16K-1 Klasse-B-Netze, mit jeweils bis zu 64K-2 Hosts
- o 2M-1 Klasse-C-Netze, mit jeweils bis zu 254 Hosts

273

Subnetze

- o Beim Internet gibt es die Möglichkeit, einen Teil der Hostadresse als Subnetzadresse lokal zu vergeben

Beispiel:

Die Uni Augsburg ist ein Klasse-B-Netz mit 137.250 als Netzadresse. Die verbleibenden 16 Bit Hostadresse können z.B. aufgeteilt werden in 8 Bit Subadresse und 8 Bit Host-Adresse. Jedem Fachbereich kann dann vom lokalen Netzwerkadministrator eine Subadresse zugewiesen werden. Der Fachbereich Informatik z.B. hat als Subnetzadresse 71, so daß dort alle IP-Adressen 137.250.71.x lauten (ist äquivalent mit 137.250.71.0)

o Achtung:

Ein IP-Subnetz hat zunächst nichts mit einem Subnetz der Schicht 1 und 2 zu tun! Beispielsweise können mehrere LANs ein gemeinsames IP-Subnetz bilden

- o Die Verwaltung der Subnetze geht über Subnetzmasken, die mit der IP-Adresse UND-verknüpft werden.

Beispiel:

137.250.71.x AND 255.255.255.0 liefert das Informatik-Subnetz 137.250.71.0 (255.255.255.0 ist die Subnetzmaske)

- o Klasse-B-Netz, organisiert in 62 Subnetzen mit max. 1022 Hosts pro Subnetz (16 Bit = 6 Bit + 10 Bit)

275

Vorteile von Subnetzen

- o Die Router müssen bzgl. ihrer Ausgangsleitungen nicht eine große Zahl von Hosts (bis zu 16M-2 bzw. 64K-2 Hosts) verwalten, sondern nur wissen, über welche Leitung man zum jeweiligen Subnetz kommt
 - ⇒ erhebliche Einsparung bei den Routing-Tabellen
- ⇒ Die Wegewahl erfolgt von Subnetz zu Subnetz (hierarchisch)
- ⇒ Einführung einer dreistufigen Hierarchie (Netzklasse, Netzadresse, Subnetzadresse)
- o Subnetze sind von außen nicht erkennbar, deshalb kann die Organisation in Subnetze ohne NIC gemacht werden!

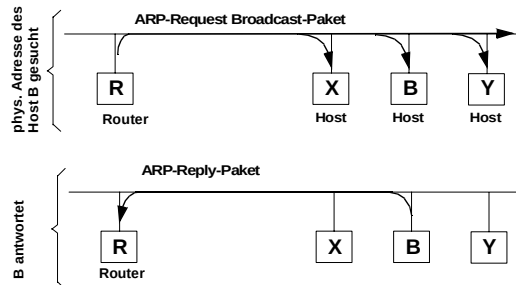
Wie funktioniert die Adressauflösung im LAN?

Problem:

Wie soll die Abbildung der Internet-Adresse (IP-Adresse) eines Rechners auf die physikalische Stationsadresse im LAN (IEEE 802-Adresse) erfolgen?

- 1.) Wenn die physikalische Adresse (Stationsadresse) durch den Benutzer frei gewählt werden kann, wähle LAN-Adresse = IP-Adresse (gilt für die ersten 4 Byte der LAN-Adresse)

276



278

- 2.) Wenn die physikalische Adresse vorkonfiguriert („fest verdrahtet“) ist, unterhalte eine Abbildungstabelle (z. B. im Router) und/oder benutze das Address Resolution Protocol ARP. Dieser Fall ist bei Ethernet/Token Ring immer gegeben

Address Resolution Protocol ARP

- Das Protokoll im Router läuft folgendermaßen ab:

- 1.) Sende mittels Broadcast auf dem LAN ein ARP-Request-Paket, welches die physikalische und die Internet-Adresse des Senders und die Internet-Adresse des gesuchten Empfängers enthält.
- 2.) Warte auf die Antwort des Empfängers durch ein ARP-Reply-Paket, welches seine physikalische Adresse enthält.
- 3.) Unterhalte einen Cache aus (I, P)-Adresspaaren für spätere Anfragen.
- 4.) Verbesserung: Der Empfänger des ARP-Requests speichert das (I, P)-Paar des Senders auch in seinem Netzwerk-Cache.

277

Die Transportschicht (Schicht 4)

ISO-Transportschicht

TCP-Transportschicht im Internet

279

Beispiel für eine Transportschicht: Der ISO-Transportdienst

Verbindungsorientierter Transportdienst
(ISO IS 8072, CCITT X.214)

Eigenschaften/Aufgaben:

- Aufbau einer Transportschichtverbindung
- Datenübertragung
- optional beschleunigte Datenübertragung (expedited data)
- Abbau einer Transportschichtverbindung
- Eingeschränkte QoS-Parameter (QoS = Quality of Service)
(QoS ist höher als bei TCP/IP, aber niedriger als bei ATM)

280

- Der Anwendungsprozeß spricht die Transportschicht über einen sog. Transportschicht-Dienstzugangspunkt (TSAP) an.
- Die Transportschicht wiederum spricht die Vermittlungsschicht über einen Vermittlungsschicht-Dienstzugangspunkt (NSAP) an (Network Layer Service Access Point)
- Die TSAPs werden bei TCP durch „Ports“ realisiert
- Operationen des ISO-Transportdienstes

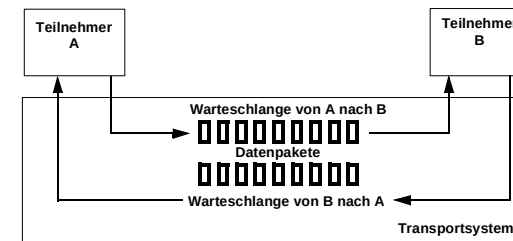
Primitive	TPDU sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA TPDU arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Berkeley-Sockets

- o Sind die Programmierschnittstelle zur Kommunikation mit TCP im Internet
- o Verwenden IP- und Port-Adresse zur netzweit eindeutigen Adressierung von Server-Diensten

282

ISO-Modell des Transportdienstes (gilt auch bei TCP)



- Vollduplex-Datenübertragung
- Übergebene Datenpakete werden zuverlässig in der richtigen Reihenfolge übertragen (FIFO-Reihenfolge)
- Ausnahme: beschleunigte Daten (expedited data). Diese halten untereinander die Reihenfolge ein, werden aber mit normalen Daten-PDU's verschränkt übertragen. Eine separate Warteschlange mit höherer Priorität kann, muss aber nicht existieren.
- durch den Dienst der Transportschicht erleiden Datenpakete nur eine gewisse Verzögerung. Alle anderen Fehler und Unzulänglichkeiten der Sicherungsschicht werden behoben.

281

Beispiel: TCP-Transportdienst

- TCP ist das wichtigste Transportprotokoll im Internet
- Neben TCP gibt es noch UDP als Transportprotokoll, das allerdings nur Datagramm-basiert ist

TCP- Eigenschaften

- verbindungsorientierter Dienst
- korrigiert Paketfehler, -verlust, -duplizierung und Fehler in der Paketreihenfolge
- ist sehr effizient hinsichtlich des Datenaustauschs
- ⇒ die TCP-Transportschicht verbessert die Übertragungsqualität (Quality of Service) der Vermittlungsschicht

Was ist der Unterschied zwischen der Sicherungs- und der Transportschicht?

- die Transportschicht arbeitet direkt von Benutzerprozeß zu Benutzerprozeß. Diese werden auf nicht-benachbarten Rechnern ausgeführt
- die Sicherungsschicht arbeitet nur zwischen benachbarten Rechnern. Prozesse gibt es für sie nicht

283

Optimierung der Datenübertragung bei TCP

- TCP ist sehr effizient hinsichtlich der Übertragung von Daten
- Die Gründe für die Effizienz liegen in:
 - Der Verwendung eines adaptiven Schiebefensters zur Überlaststeuerung
 - Dadurch wird die optimale Grösse eines Pakets ermittelt
 - Der Verwendung eines adaptiven Time-Outs
 - Dadurch werden unnötige Wartezeiten bei Paketverlust vermieden
 - Der Möglichkeit der hochpriorisierten Übertragung
 - Erlaubt falsche Übertragungen vorzeitig abzugeben

284

Zusammenfassende Liste der Dienste der TCP-Transportschicht

- Verbindung Prozess-zu-Prozess hinsichtlich
 - 1.) Verbindungsaufbau
 - 2.) Datenübertragung
 - 3.) Verbindungsabbau
- Zusammenfassen von mehreren kurzen TCP-Paketen derselben Verbindung zu einem langen Paket (=effizienter)
- Zerteilen langer Nachrichten der Sitzungsschicht in 64K-Pakete
- Fehlerbehebung u. a. durch
 - 1.) Verbindungsneuzuordnung (reset einer Verbindung)
 - 2.) Resynchronisation (Wiederaufbau einer unterbrochenen Verbindung)
- Flusskontrolle, Sicherstellung der Reihenfolge (Ende-zu-Ende)
- Bestätigte Datenübertragung (Ende-zu-Ende)

285