

Betriebssysteme

Kapitel 6: I/O und Storage

6.1: I/O, Disks, SCSI

Stand: WS 10/11 (15.3.2011)

Prof. Dr. Wolfgang Küchlin

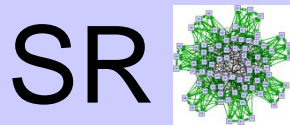
Dipl.-Inform., Dr. sc. techn. (ETH)

**Arbeitsbereich Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften**

Universität Tübingen

**Steinbeis Transferzentrum
Objekt- und Internet-Technologien (OIT)**

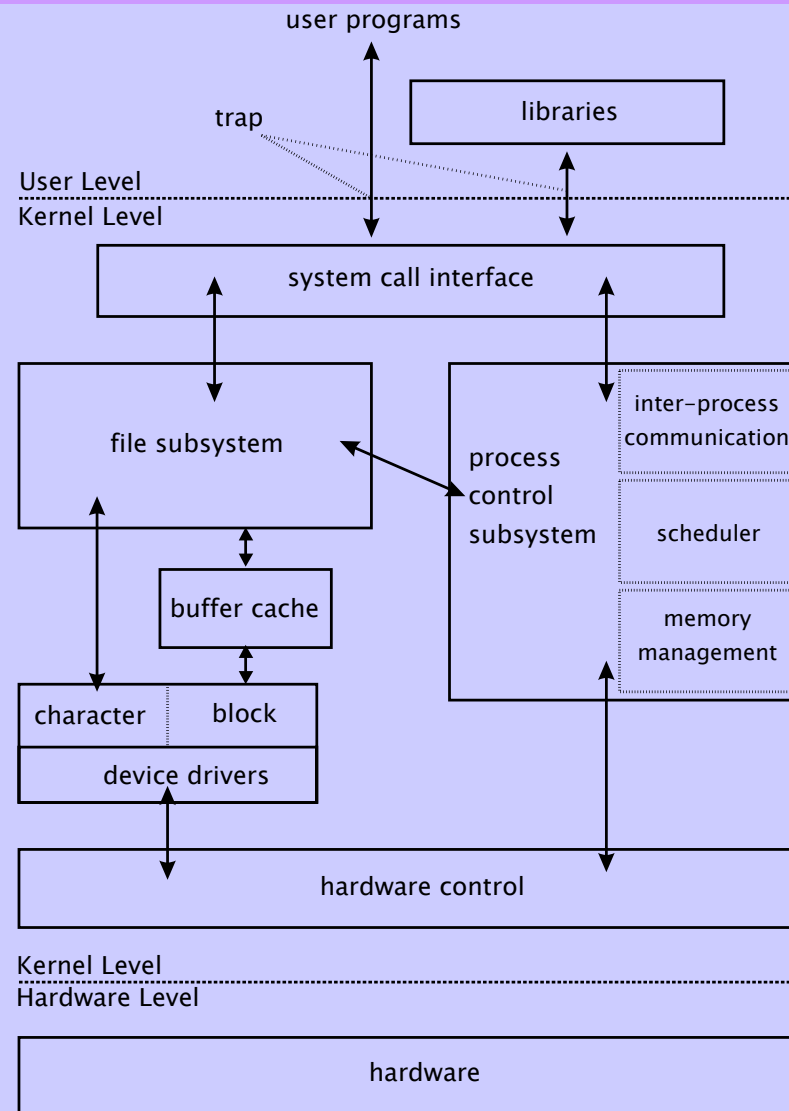
**Wolfgang.Kuechlin@uni-tuebingen.de
<http://www-sr.informatik.uni-tuebingen.de>**



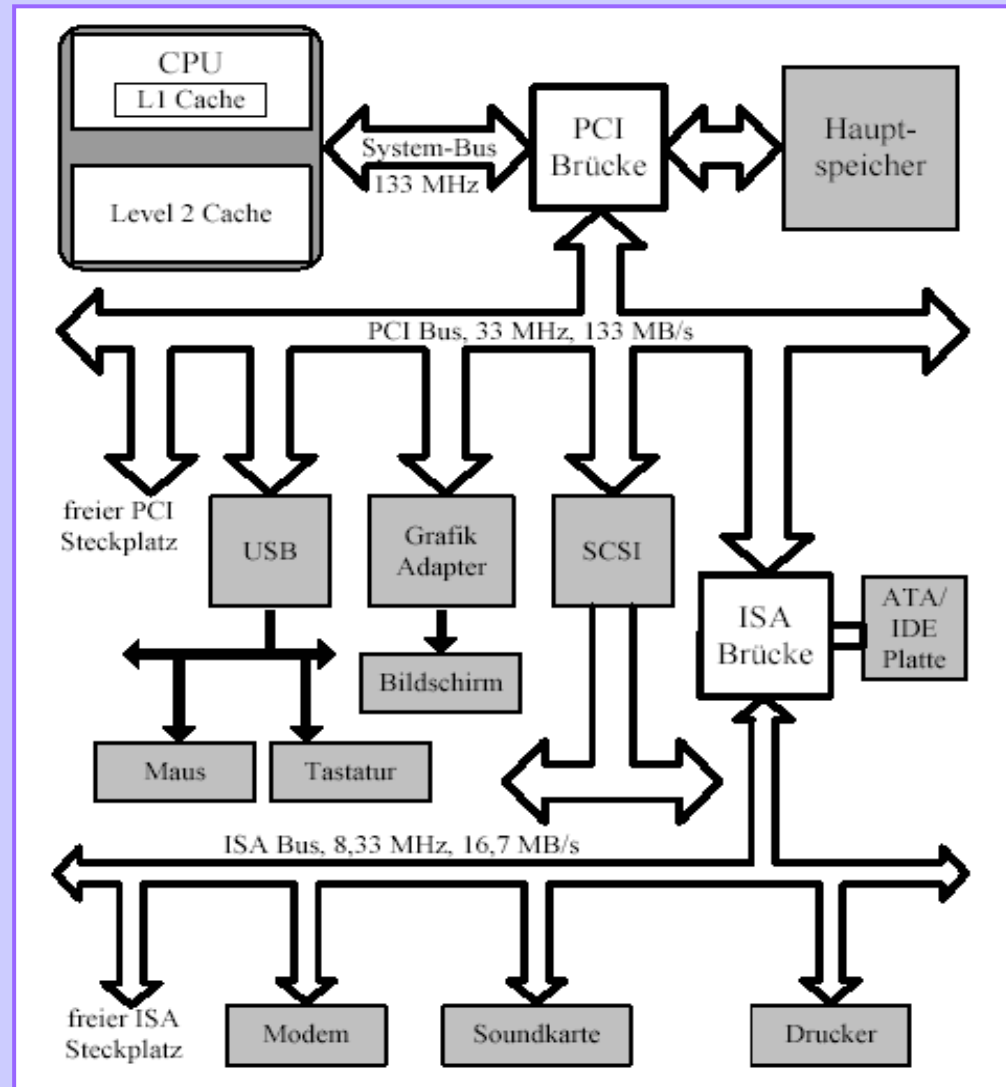
Teil 1: I/O und Disks

Teil 2: SCSI





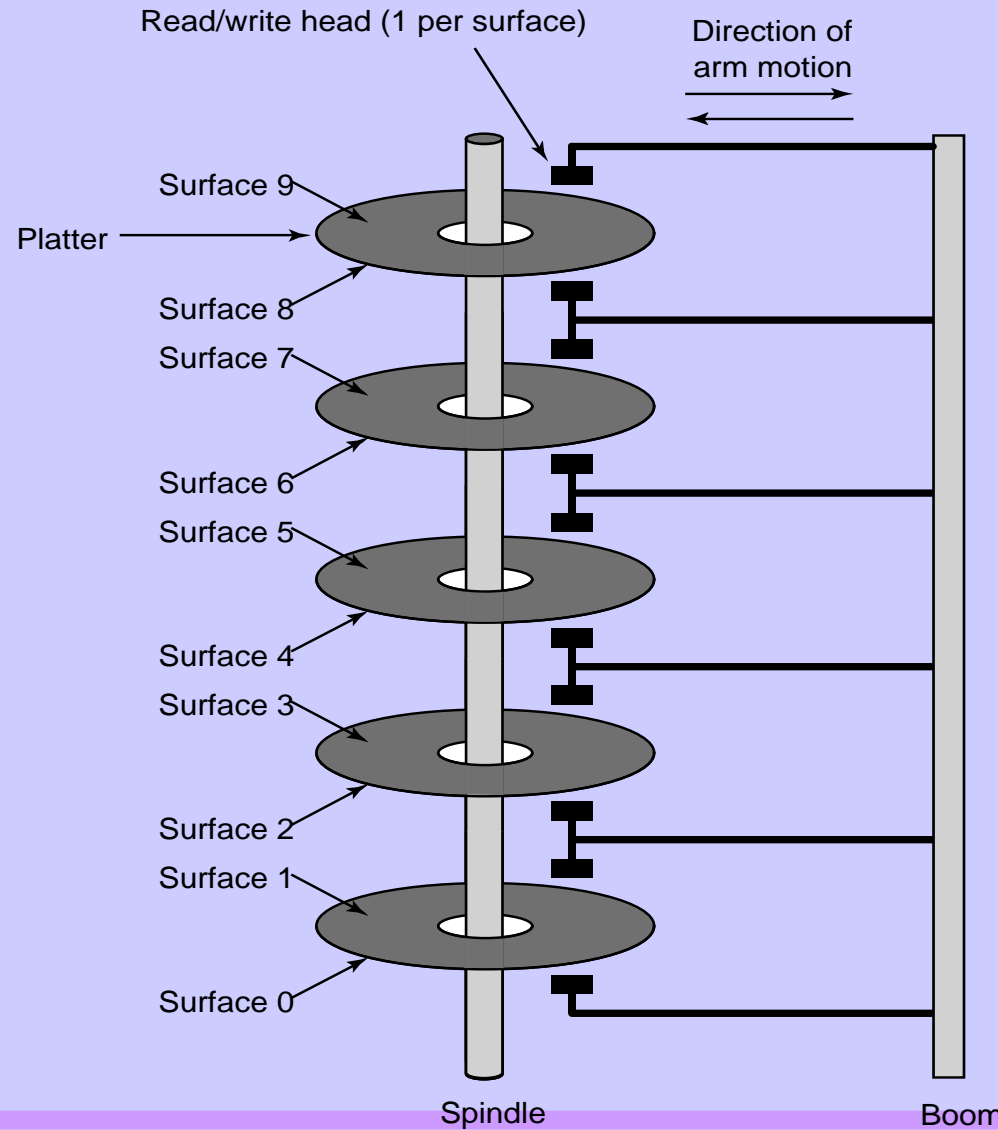
Erinnerung: System-Architektur der Hardware



Küchlin, Weber,
Abb.2.3:
„Architektur eines Intel
P2 PC Systems mit
mehreren Bussen an
Brücken“

Komponenten eines Plattenlaufwerks

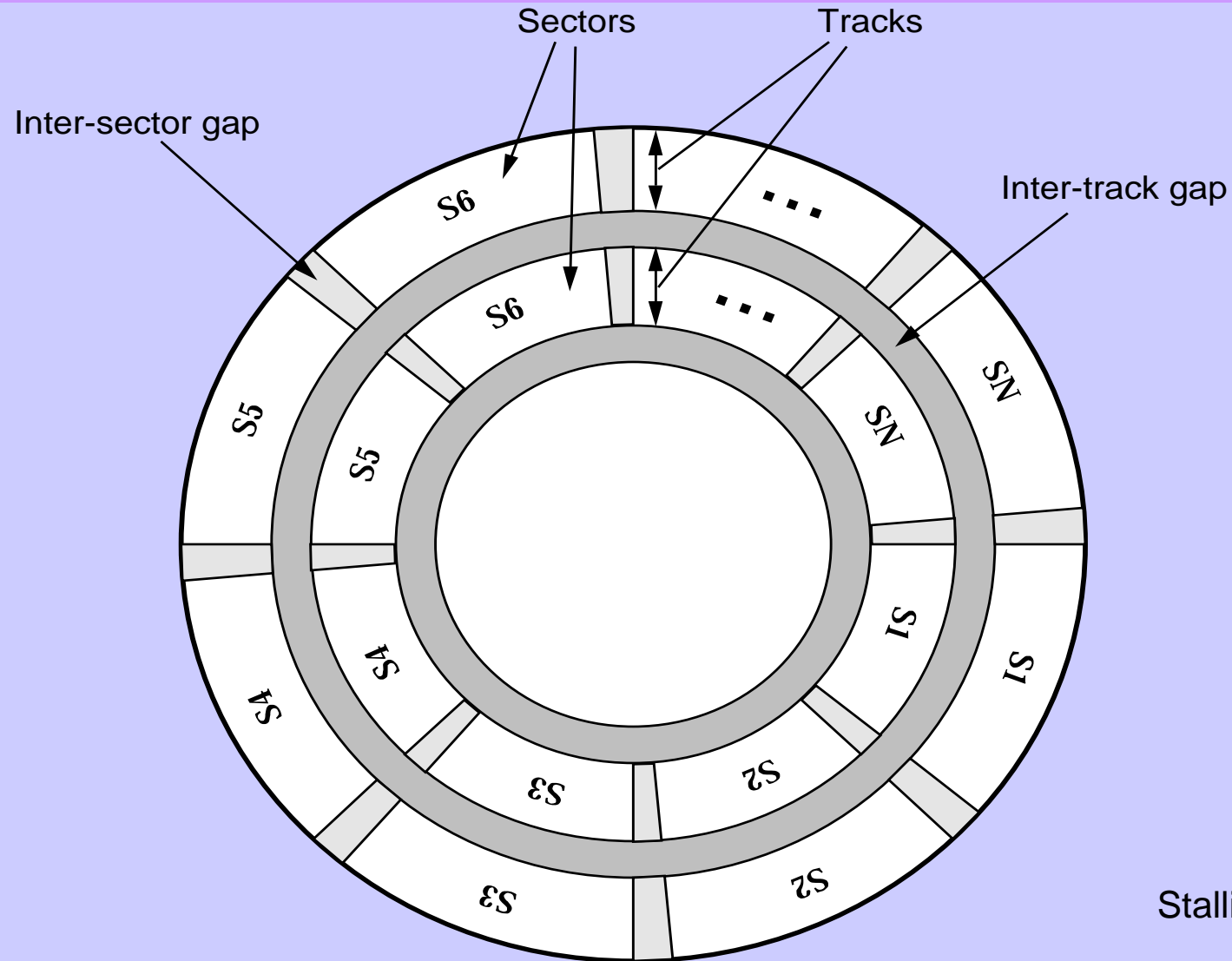
BS 6, Storage, WS10



Stallings, Abb.11.18



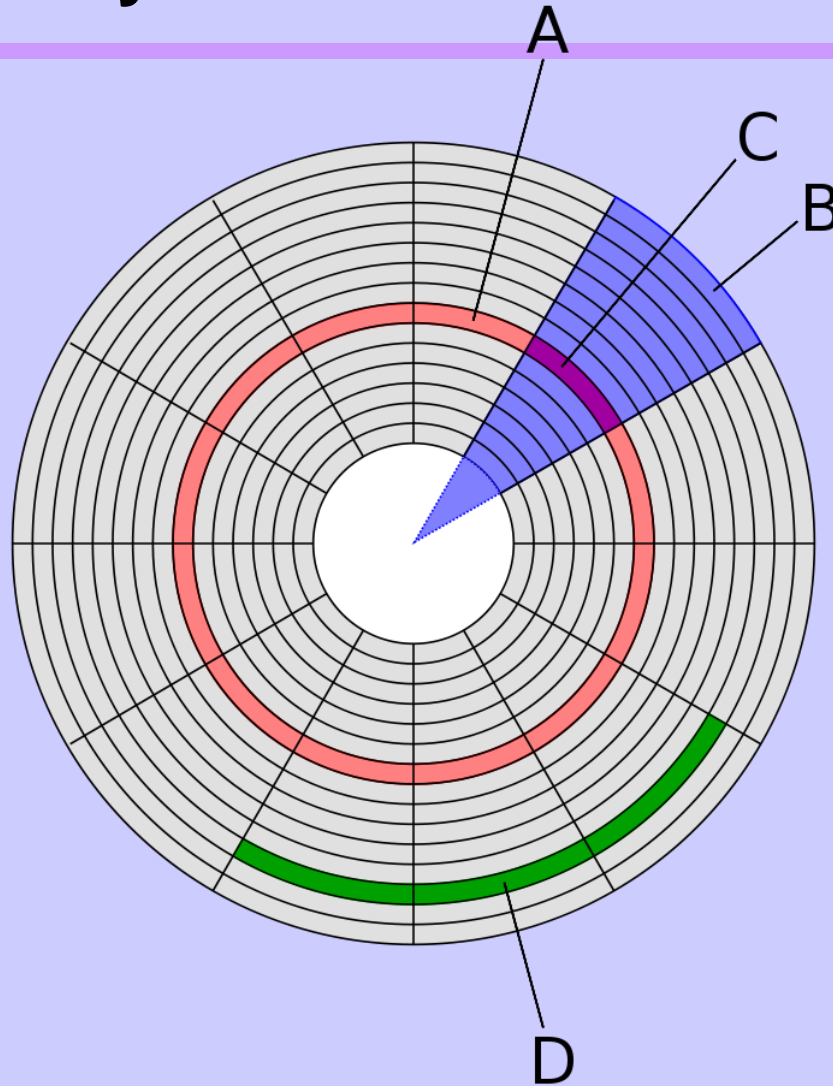
Disk Data Layout



Stallings, Abb.11.16

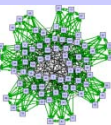


Disk Data Layout



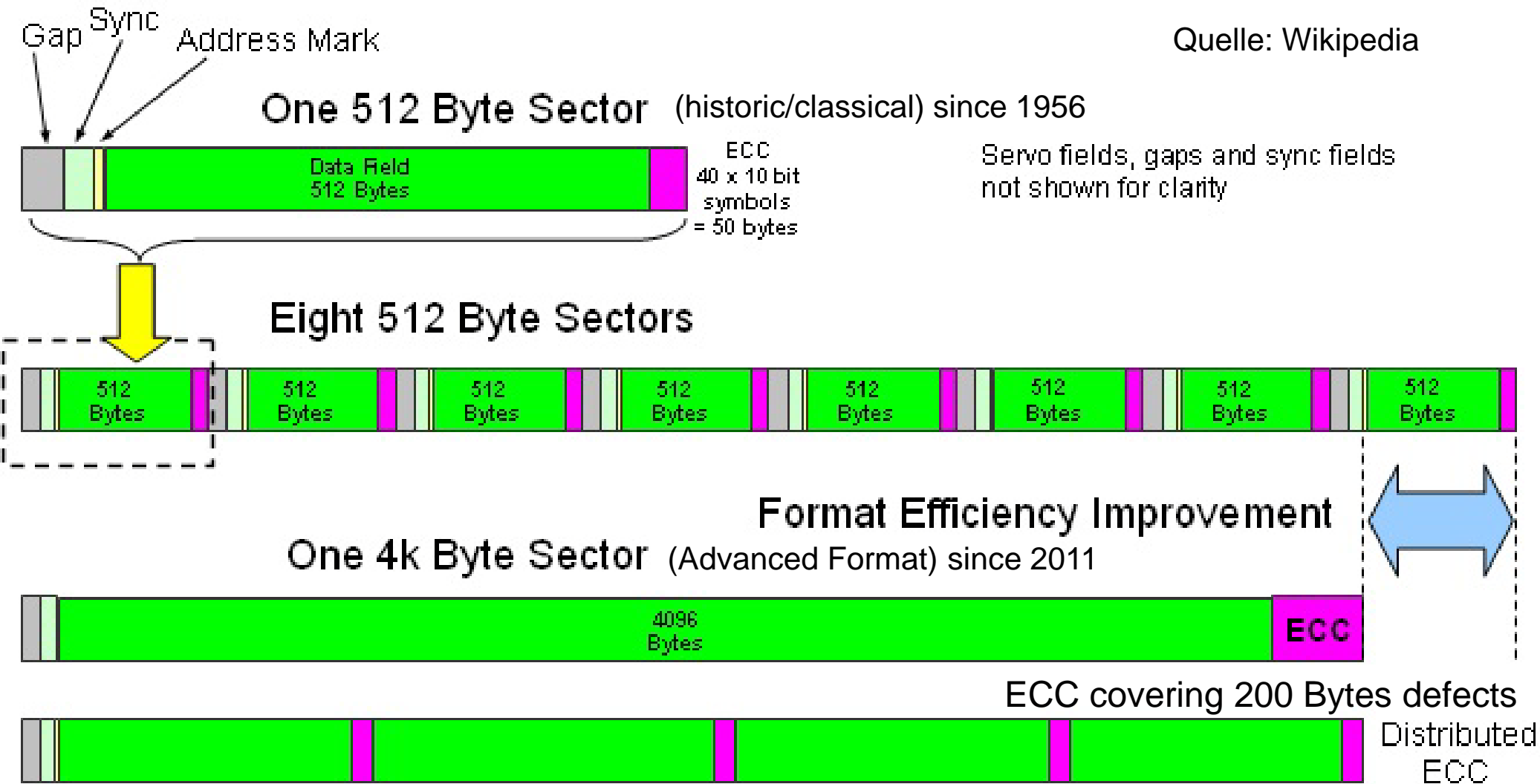
- A: Track (Spur)
- B: Geometrical Sector
- C: Track Sector
- D: Cluster (of sectors)

Quelle: Wikipedia



Disk Layout: Classical and Advanced Format Sectors

Quelle: Wikipedia



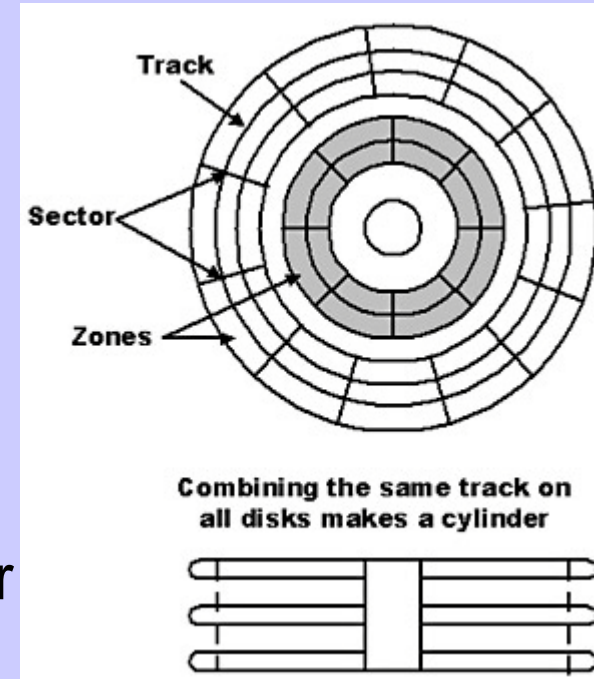
Disk Data Layout: Sectors

- Sektor: kleinste Speicher-Einheit auf der Festplatte
 - klassisch: 512 Bytes Nutzdaten (seit 1956)
- Block: kleinste Datenmenge, die das Filesystem überträgt
 - kann ein Vielfaches der Sektorgröße sein
- Advanced Format: 4096 Bytes
 - Neudefinition seit 2011
 - Durch IDEMA: International Disk Drive Equipment and Materials Association (Hersteller-Vereinigung)
 - 512e Format: Platte bildet externe 512byte Block-Requests ab auf 4KB Sektoren (write führt zu read-replace-write Sequenz)
- Nutzdaten sind gesichert durch Prüfsumme (ECC Block)
 - klassisch: 40x10bits sichern (korrigieren) bis zu 50 Bytes Nutzdaten
 - Advanced Format: neuer ECC Block sichert 200 Bytes Nutzdaten



Disk Data Layout: Tracks and Zones

- Defining a sector as the intersection between a radius and a track creates larger sectors towards the outside of the disk.
- The same number of bytes can be accomplished by reducing the bit density of the outer sectors relative to the inner ones.
- Modern hard discs use **zoned bit recording**, where the disk is divided into zones encompassing a small number of contiguous tracks. Each zone contains sectors of a similar physical size. Outer zones have more sectors per zone. This is known as **zoned bit rate**.



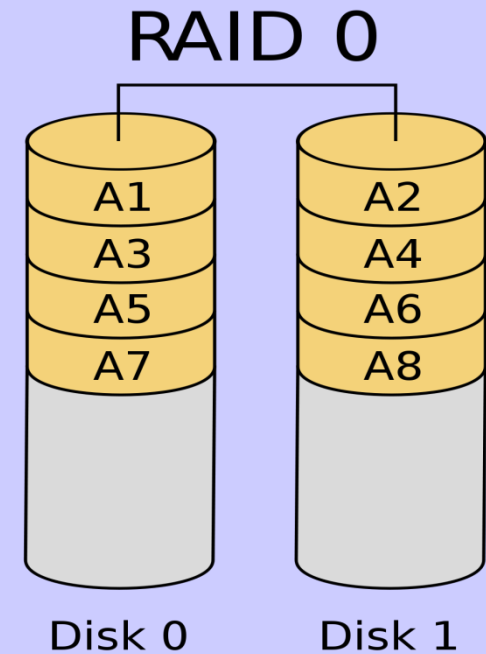
RAID

- RAID = Redundant Array of Inexpensive/Independent Disks
- Problemstellung:
 - Platten werden immer größer und immer billiger
 - Bei Ausfällen gehen große Datenmengen verloren
 - Die Wahrscheinlichkeit von Ausfällen steigt mit der Anzahl benutzter Platten
 - Platten werden relativ zu Prozessoren immer langsamer
- Durch Parallelschaltung werden Plattenzugriffe schneller
- gleichzeitig steigt das Ausfall-Risiko durch mehr Platten
 - 1 Platte: Ausfallrisiko p , kein Ausfall: $(1-p)$.
 - n Platten: Ausfallrisiko: $1 - (1-p)^n$
 - Bei Risiko 1/1000 und 1.000 Platten ist das Risiko = 1 !!
- Durch extra Platten bekommt man die nötige Redundanz
- Verschiedene RAID-Varianten im Einsatz
 - gebräuchlichste: RAID-0, RAID-1, RAID-5 und Kombinationen davon



RAID-0 (Striping)

- Jeder Datenblock wird in Streifen (*strip*) über parallele Disks verteilt
- Hier mit 2 Disks →
- Eigenschaften:
 - Beschleunigung durch Parallelzugriff
 - keine Redundanz

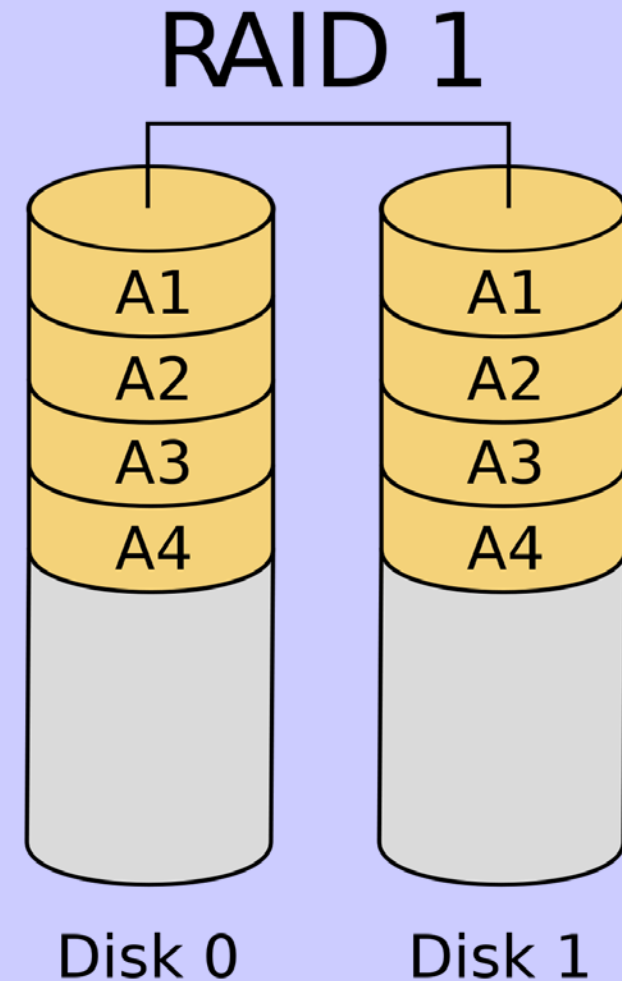


Grafik: Wikipedia



RAID-1 (Mirroring)

- Platten werden gespiegelt
- Hier mit 2 Disks →
 - jede Platte separat funktionsfähig
(alle I-nodes und Datenblöcke vollständig)
- Eigenschaften:
 - Redundanz: 1 Platte darf ausfallen
 - 2x-schnelles Lesen durch Parallelzugriff
 - Schreiben nur 1x-schnell



Grafik: Wikipedia

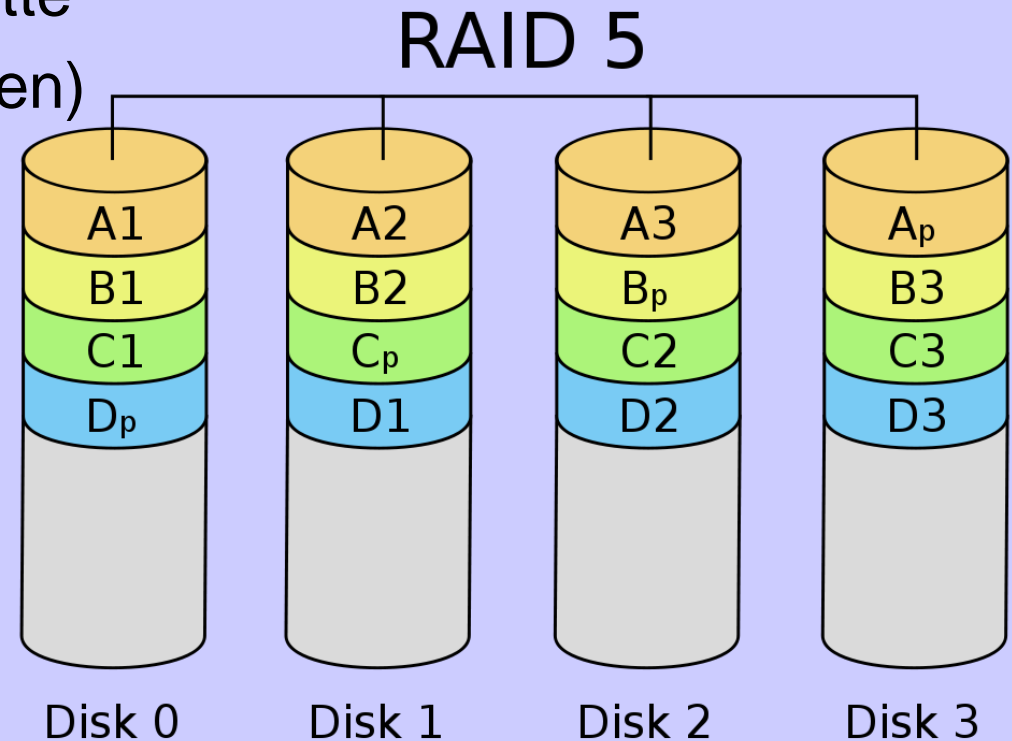
RAID-5 (Block-level striping with parity)

➤ Striping mit Paritätsbit

- z.B. 1 Wort = 4 Bytes + 1 Byte für die Paritätsbits
- Raid-5: Platte mit Paritätsbyte wechselt
- Raid-4: dedizierte Parity-Platte
(Flaschenhals beim Schreiben)

➤ Hier mit 3+1 Disks →

- oft auch 4+1 Disks



Grafik: Wikipedia

RAID-5 (Block-level striping with parity)

- Schreiben verlangt Parity-Berechnung
 - Entweder alle restlichen strips lesen und dann parity rechnen
 - Oder neuen strip mit altem vergleichen und parity rechnen
- Danach Daten strip und parity strip schreiben
 - Mehrere dieser Schreibvorgänge können bei Raid-5 im Durchschnitt parallel erfolgen, bei Raid-4 bildet Parity-Platte ein bottleneck.
- Lesen durch Parallelzugriff
 - z.B. wenn ein 2KB File-Block aus 4 Strips zu je 512KB besteht
 - z.B. wenn 4 Prozesse je 1 strip von versch. Platten lesen



RAID-Kombinationen

- Ein RAID bildet nach außen wieder eine logische Platte (die schneller und sicherer ist).
- Aus solchen „RAID“-Platten können wiederum RAIDs gebildet werden
 - Beispielsweise können mehrere Platten zu einem parallelen *RAID 0* zusammengefasst werden und aus mehreren dieser *RAID-0*-Arrays kann ein *RAID-5*-Array gebildet werden. Man bezeichnet diese Kombinationen dann etwa als *RAID 05* (0+5).
 - Umgekehrt würde ein Zusammenschluss von mehreren *RAID-5*-Arrays zu einem *RAID-0*-Array als *RAID 50* (oder *RAID 5+0*) bezeichnet werden.
 - Die beliebtesten Kombinationen sind *RAID 01* und *RAID 10*



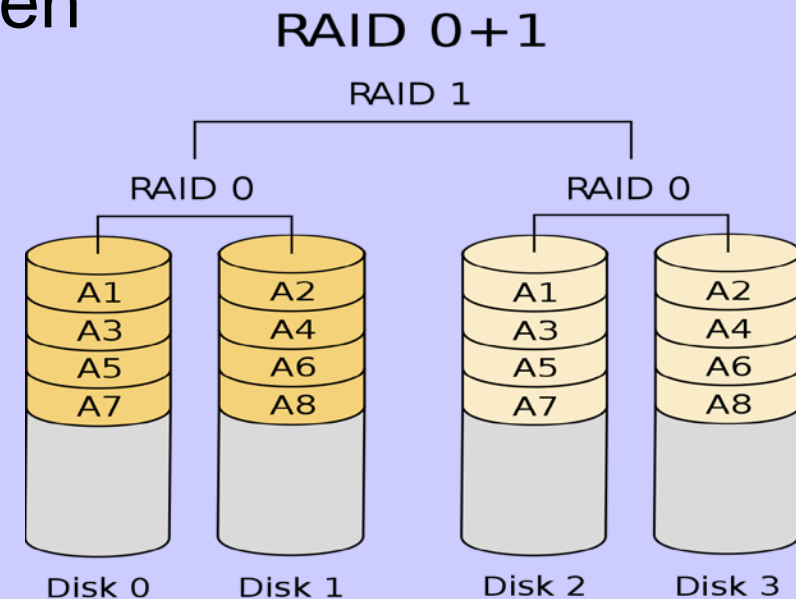
RAID-01

- Bei *RAID 01* („0+1“, bzw. „1 über 0“) arbeiten je zwei Platten parallel und werden dabei von zwei anderen Platten gespiegelt (insgesamt vier Platten).

- „sicheres Raid aus schnellen Platten“

- Eigenschaft

- Schnelligkeit durch RAID-0
- Sicherheit durch RAID-1
- Weniger Kapazität als RAID 5 mit 3+1 Platten (50% Nutzdaten i. Vgl. zu 75% Nutzdaten)
- Bei Ausfall einer RAID-1 „Platte“ muss ein ganzes RAID-0 erneuert werden



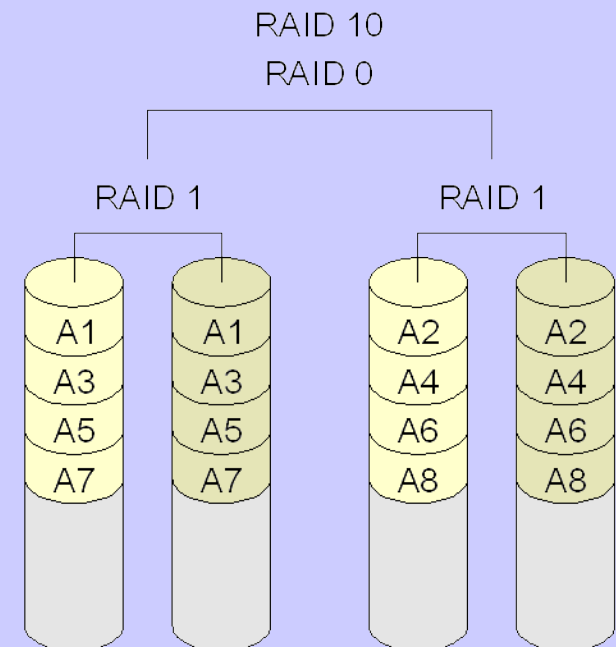
Grafik: Wikipedia

werden

Wolfgang Küchlin, WSI und STZ OIT, Uni Tübingen

RAID-10

- Bei *RAID 10* („1+0“, „0 über 1“) werden (mindestens) zwei mal zwei Platten gespiegelt und zu einem RAID 0 verbunden
- „schnelles Raid aus sicheren Platten“
- Eigenschaft
 - Sicherheit durch RAID-1 Platten
 - Schnelligkeit durch Parallelität
 - Weniger Kapazität als RAID 5 mit 3+1 Platten (50% Nutzdaten i. Vgl. zu 75% Nutzdaten)
 - Ausfall eines Laufwerks kann genau lokalisiert und lokal (schnell) behoben werden



Grafik: Wikipedia

werden

Wolfgang Küchlin, WSI und STZ OIT, Uni Tübingen

Teil 1: I/O und Disks

Teil 2: SCSI – Small Computer Systems Interface



SCSI – Funktionsprinzip

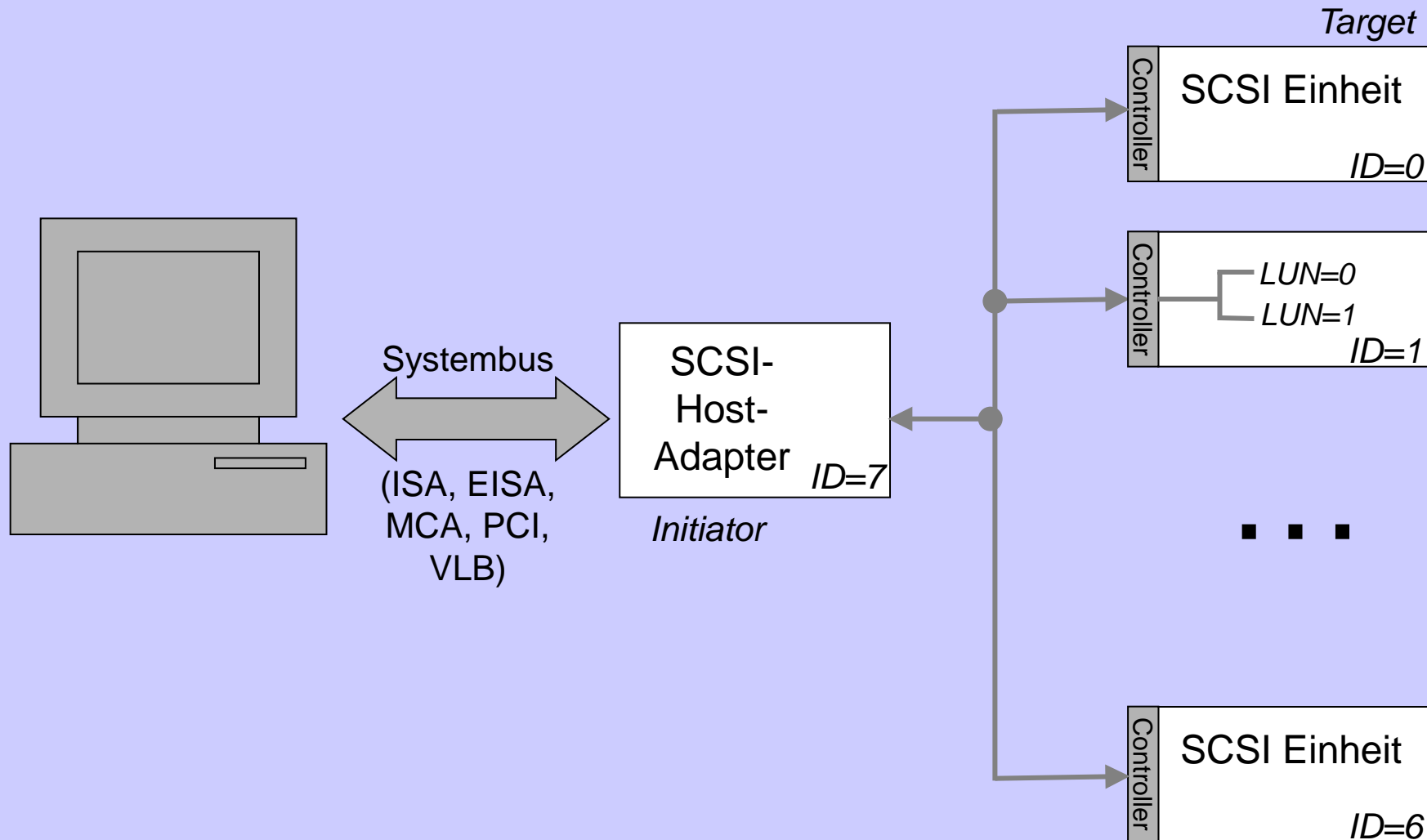
- SCSI definiert einen **Bus und ein Protokoll** zum Datenaustausch
- SCSI betrachtet gesamte Speicherkapazität einer Festplatte als zusammenhängende **Liste von Datenblöcken** fester Größe.
 - im Gegensatz zu anderen Festplattenschnittstellen (z.B. IDE) mit Spuren und Sektoren
- Verwaltung der Blöcke Aufgabe des intelligenten Controllers
- SCSI Festplatte setzt logische Blocknummer in Spuren und Sektoren um
 - autonomes bad block replacement möglich
- SCSI Bandlaufwerk übernimmt Blocknummern fast unverändert



SCSI – Funktionsprinzip

- SCSI-Protokoll besteht aus physikalischen Steuersignalen und logischen Messages.
- SCSI Bus:
 - max. 8 Teilnehmer
 - Beispiele:
 - Festplatten
 - Bandlaufwerke
 - optische Laufwerke
- Nur zwei Teilnehmer können gleichzeitig aktiv sein
- Datenaustausch ohne Eingreifen der CPU möglich





SCSI vs LUN

- SCSI-ID: SCSI-Adresse für jede SCSI-Einheit
 - Adressen 0 bis 7
 - Hostadapter verwendet meist Adresse Nr.7
 - Hostadapter spricht als Initiator die anderen SCSI-Einheiten (*Targets*) an
 - Genauer: Controller als Verbindung zum SCSI-Bus wird angesprochen
- LUN (= *Logical Unit Number*)
 - Jedes Target kann bis zu acht logische Einheiten enthalten
 - Identifizierung der logischen Einheiten innerhalb eines SCSI-Befehls über LUN.
 - Bsp.: SCSI-Controller, der mehrere Laufwerke steuert (→ RAID)



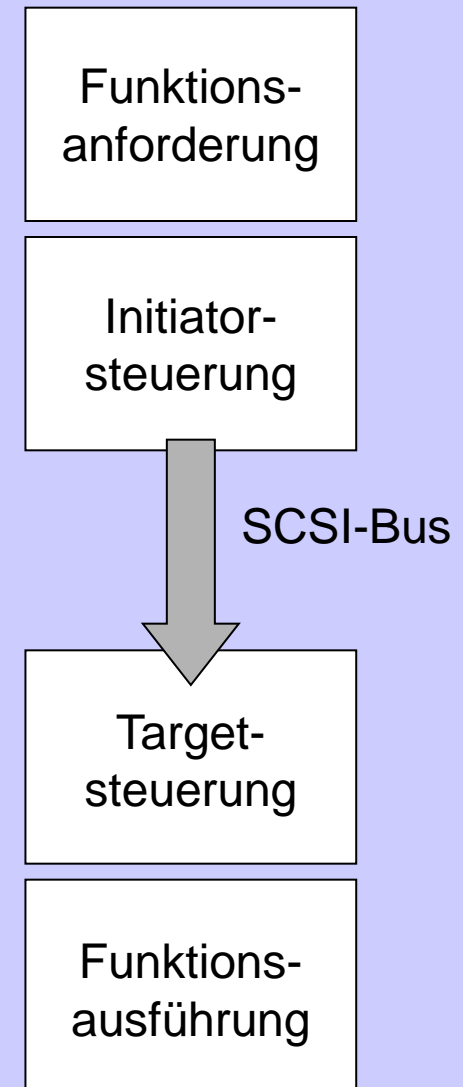
SCSI – Signale und Datenübertragung

- Datenübertragung
 - acht Datenbits DB(0) bis DB(7)
 - ein Paritätsbit DB(P)
 - neun Steuersignale
- Datenübertragungsrate bei verschiedenen Modi
 - asynchroner Modus: 3MB/s
 - synchroner Modus: 5MB/s
 - Fast-Modus: 10MB/s
 - Ultra-Fast-Modus: 20MB/s



SCSI – Datenübertragung

- Jede SCSI-Einheit kann als Initiator die Steuerung des SCSI-Bus übernehmen.
- Initiator aktiviert über SCSI-Adresse ein Target, das eine bestimmte Funktion ausführen soll.
 - Initiator belegt Bus nur zur Befehls- und Datenübertragung
 - Beispiel für auszuführende Funktion: Lesen eines Blocks
- Target stellt nach Befehlsausführung für Datenübertragung Verbindung zum Initiator wieder her.
- Ansonsten ist der Bus frei für andere SCSI-Einheiten.



SCSI – Busphasen und Messages

➤ Acht Busphasen bei SCSI

- Bus frei
 - Arbitration
 - Selection
 - Reselection
 - Command
 - Daten
 - Message
 - Status
- } Informationstransferphase



SCSI – Busphasen und Messages

➤ Bus-frei-Phase

- keine SCSI-Einheit nutzt oder steuert den Bus im Moment

➤ Arbitration-Phase

- SCSI-Einheit kann als Initiator oder Target die Steuerung des Buses übernehmen

➤ Selection-Phase

- Initiator kann eine Target-Einheit für eine auszuführende Funktion auswählen

➤ Reselection-Phase

- Target kann Verbindung mit dem Initiator *wieder* aufnehmen
- Bsp: Ausführung eines Lesebefehls
 - während interner Ausführung beim Target: Bus-frei-Phase



SCSI – Busphasen und Messages

- Command-Phase
 - adressiertes Target kann Befehlsdaten vom Initiator anfordern
- Daten-Phase
 - Target verlangt Daten vom Initiator (*Data-out*)
 - Target liefert Daten an den Initiator (*Data-In*)
- Message-Phase
 - Control Messages werden vom oder zum Target geschickt
 - Beispiele für Messages
 - Befehl abgeschlossen
 - Trennen (z.B. während der internen Verarbeitung)
 - Abbruch
 - ...
- Status-Phase
 - Target übergibt Statusinformationen an den Initiator



SCSI – Busphasen und Messages

➤ Jeder Initiator besitzt zwei Sätze à drei Zeiger

- aktive Zeiger
 - betreffen gegenwärtig aktive Verbindung
 - zeigen auf nächstes zu übertragendes Befehls-, Daten- bzw. Statusbyte
- gesicherte Zeiger
 - gibt es für jeden aktiven Befehl, auch wenn Verbindung zwischen Initiator und Target temporär nicht besteht
 - Befehlszeiger weist auf Beginn des Befehlsblocks
 - Statuszeiger weist auf Beginn des Statusbereichs



SCSI – Befehle und Programmierung

➤ Schema für SCSI-Befehle:

Befehlscode (1 Byte)

LUN (3 Bits)

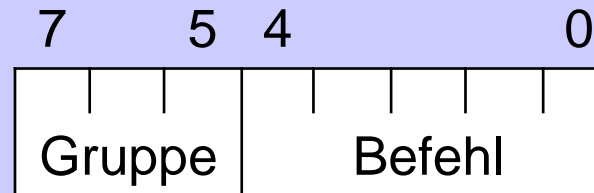
Blockadresse /-Länge/ Parameter
(variable Länge)

Steuerbyte (1 Byte)

➤ Aufbau eines Befehlsbytes

- Befehl (5 Bits)
 - Wert 0 bis 31: Befehlscode
- Gruppe (3 Bits)

0	6-Byte-Befehl
1,2	10-Byte-Befehl
3,4	Reserviert
5	12-Byte-Befehl
6,7	Hersteller



Aus dem Aufbau folgt:

- Acht mögliche Befehlsgruppen
- pro Gruppe 32 Befehle
- insgesamt: 256 verschiedene Befehle

SCSI – Aufbau eines 6-Byte-Befehls

Bit Byte	7	6	5	4	3	2	1	0
0	Befehlscode							
1	LUN			LBA (MSB)				
2	Logische Blockadresse							
3	Logische Blockadresse (LSB)							
4	Transfer-/Zuweisungslänge/ Parameter							
5	Steuerbyte							

Logische Blockadresse: umfasst 21Bits vom most significant bit (MSB) bis zum least significant bit (LSB).

Transferlänge : Anzahl der zu übertragene Daten in Blöcken
Bei 1Byte für Transferlänge können daher 256 Blöcke auf einmal übertragen werden.



SCSI – Aufbau eines 10-Byte-Befehls

Bit Byte	7	6	5	4	3	2	1	0
0	Befehlscode							
1	LUN			reserviert				REL
2	Logische Blockadresse (MSB)							
3	Logische Blockadresse							
4	Logische Blockadresse							
5	Logische Blockadresse (LSB)							
6	reserviert							
7	Transfer-/Zuweisungslänge/ Parameter							
8	Transfer-/Zuweisungslänge/ Parameter							
9	Steuerbyte							

REL : Bit für relative
Adressierung

2 Bytes für Transferlänge =
max. 65K Blöcke Übertragung



SCSI – Aufbau eines 12-Byte-Befehls

Bit Byte	7	6	5	4	3	2	1	0
0	Befehlscode							
1	LUN			reserviert				REL
2	Logische Blockadresse (MSB)							
3	Logische Blockadresse							
4	Logische Blockadresse							
5	Logische Blockadresse (LSB)							
6	Transfer-/Zuweisungslänge/ Parameter							
7	Transfer-/Zuweisungslänge/ Parameter							
8	Transfer-/Zuweisungslänge/ Parameter							
9	Transfer-/Zuweisungslänge/ Parameter							
10	reserviert							
11	Steuerbyte							

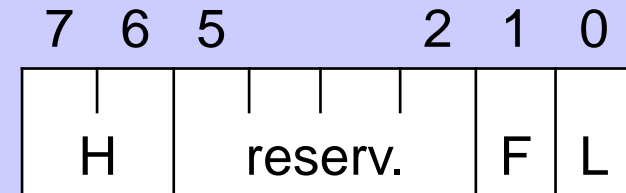
4 Bytes für Transferlänge =
max. 4G Blöcke Übertragung



SCSI – Befehle und Programmierung

➤ Aufbau eines Steuerbytes

- F (Flag) (1 Bit)
 - 0 : Message verketteter Befehl abgeschlossen übergeben
 - 1 : Message verketteter Befehl mit Flag abgeschlossen übergeben
- L (Link) (1 Bit)
 - 0 : keine Befehlsverkettung
 - 1 : Befehlsverkettung
- reserviert (4 Bits)
- H (Hersteller) (2 Bits)



➤ Ist L=1, verlangt Initiator Befehlsverkettung

- Target übergibt nach Abschluss eines Befehls nur Zwischenstatus und fordert nächsten Befehl
- Verbindung wird nicht unterbrochen und Initiator übergibt sofort nächsten Befehl



SCSI – beispielhafte Befehlsausführung (I)

- SCSI-Bus aktivieren
- Ende der **Arbitration-Phase** bestimmen und sicherstellen, dass Host-Adapter Kontrolle über Bus erlangt hat
- **Selection-Phase** zum Anwählen eines Targets
- Ermitteln ob Target auf Selection-Phase reagiert und Bussteuerung übernommen hat
- Beginn der **Command-Phase** ermitteln
- Befehlsausführung starten durch Übermittlung des Befehlsblocks an das Target über den SCSI-Datenport



SCSI – beispielhafte Befehlsausführung (II)

- Beginn der **Data-Out-Phase** ermitteln und wenn nötig
 - Parameterliste über den SCSI-Datenport ausgeben
 - Datenblock über den SCSI-Datenport ausgeben
- Wenn ein Datenblock an Initiator übergeben werden soll, Beginn der **Data-In-Phase** erkennen und Datenblock über den SCSI-Datenport einlesen.
- Beginn der **Status-Phase** ermitteln und über den SCSI-Datenport das Statusbyte empfangen.
- Beginn der **Message-In-Phase** bestimmen und die Message über den SCSI-Datenport einlesen.



SCSI – Die verschiedenen Standards

➤ Vorteile von SCSI gegenüber IDE

- bei Einsatz von diverser Peripherie
 - mehrere Festplatten
 - CD-Writer
 - Scanner
 - Tape-Streamer
- einfacher Datenaustausch bspw. mittels SCSI-Wechselplatten

➤ SCSI-II

- geräteunabhängige und daher sehr flexible Busschnittstelle

