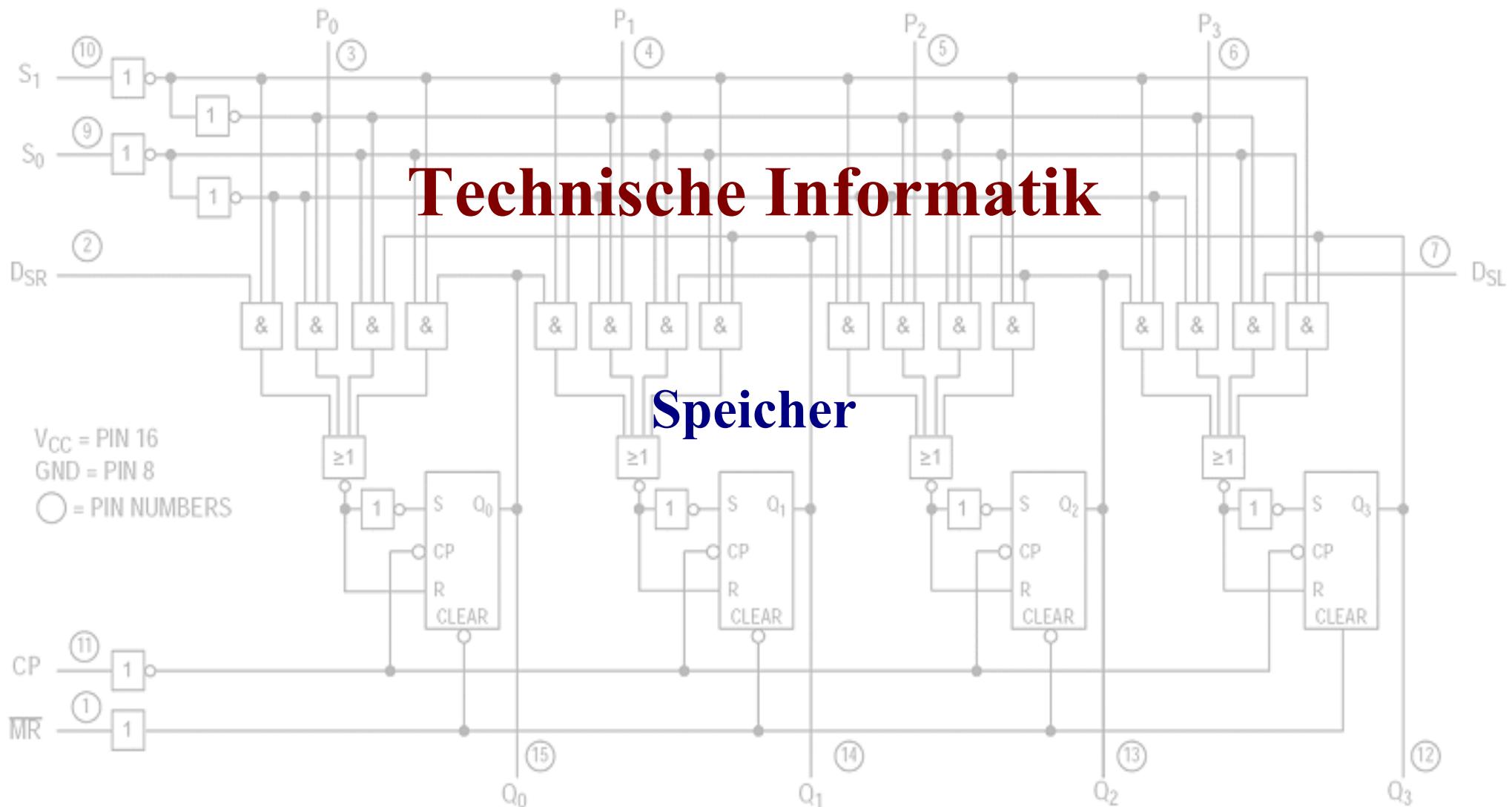


Technische Informatik

Speicher



Grundbegiffe

- Aufgabe eines Speichers:
 - Nach Vorgabe einer Adresse Daten abspeichern und unter derselben Adresse wieder ausgeben
- Einige Grundbegriffe:
 - *Random Access* (= Wahlfreier Zugriff)
 - Daten können in beliebiger Reihenfolge geschrieben und gelesen werden
 - *FIFO* (= First In First Out, Pipe)
 - Zuerst geschriebene Daten werden zuerst ausgegeben
 - *LIFO* (= Last In First Out, Stack)
 - Zuletzt geschriebene Daten werden zuerst ausgegeben
 - *Read Only*:
 - Daten können nur gelesen werden, Schreiben nur durch Hersteller oder spezielle Prozeduren
 - *Non-volatile* (= nicht flüchtig)
 - Daten bleiben ohne Strom erhalten

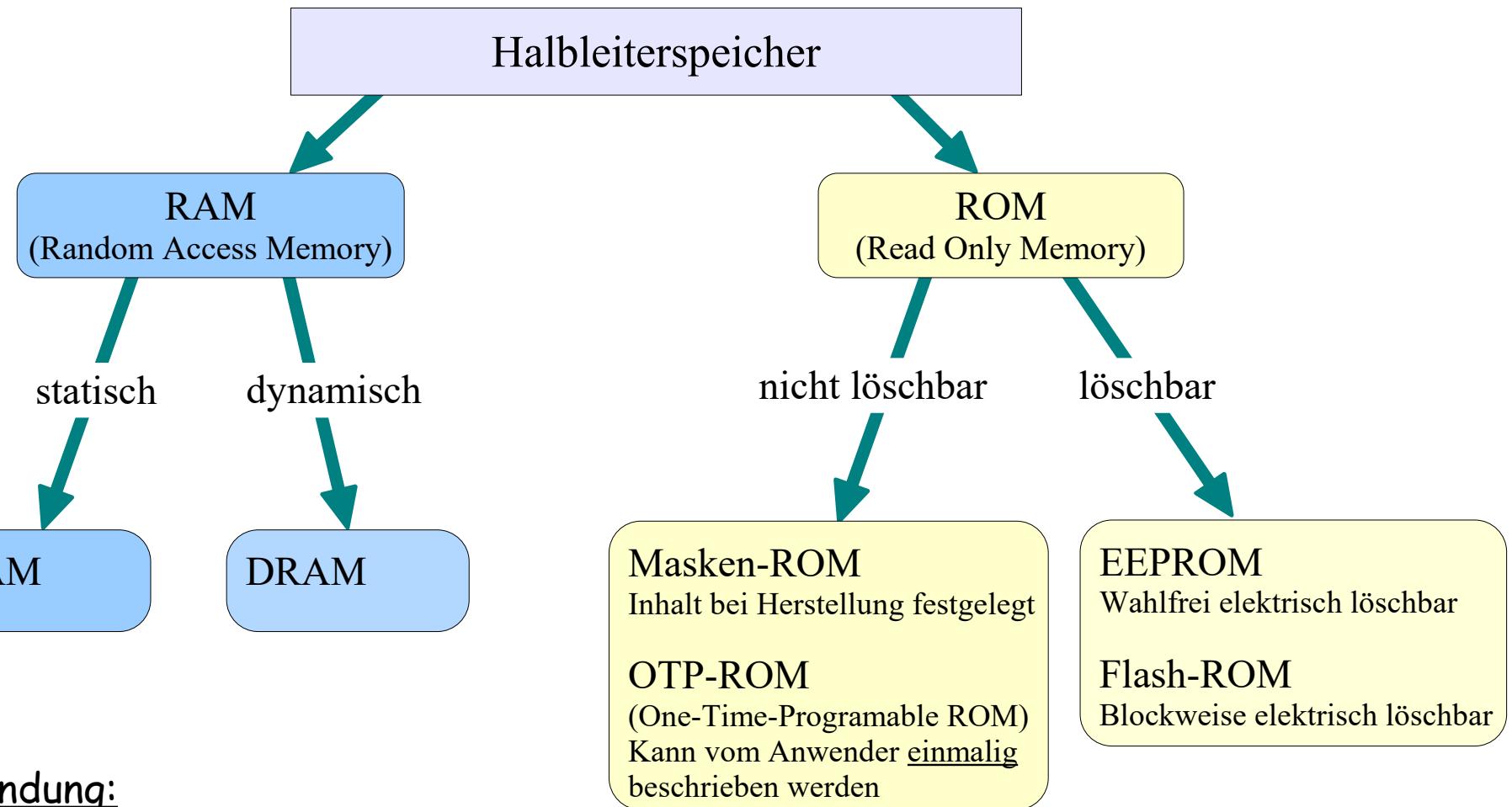
Speichertechnologien

Technologie	Anwendung als	Anwendung in	Geräte
Halbleiter-speicher	Register, RAM, ROM, PROM, EPROM, EEPROM, Flash-EPROM	Prozessor, Controller, Hauptspeicher, Cache, Puffer, Massenspeicher, Wechselspeicher, Chipkarten	Einsteckmodule, Embedded Systems, Solid-State-Drive
Magnetspeicher	Plattenspeicher, Bandspeicher	Massenspeicher, Wechselspeicher, Chipkarten	Festplatten, Bandlaufwerke, Embedded Systeme
Optische Speicher	Plattenspeicher, Bandspeicher	Massenspeicher, Wechselspeicher	Wechselplattenlaufwerke (Audio-CD, CDROM, CD-R, CD-RW, DVD, DVD-RAM), Embedded Systeme

Anforderungskriterien für Speicher

Kriterium	Einheit	Ziel
Datenrate	Bits/Sekunde	$\rightarrow \infty$
Zugriffszeit	Sekunden/Zugriff	$\rightarrow 0$
Lebensdauer (MTBF)	Jahre	$\rightarrow \infty$
Fehlerrate	Fehler/Sekunde	$\rightarrow 0$
max. Speicherkapazität	Bits/System	$\rightarrow \infty$
spezifische Speicherdichte	Bit/cm ³	$\rightarrow \infty$
spezifische Leistungsaufnahme	Watt/Bit, Watt·s/Zugriff	$\rightarrow 0$
spezifischer Preis	€/Bit	$\rightarrow 0$
Zugriffsart	wahlfrei, sequenziell, Bitstrom, Block,..	
Speichermodus	lesen, lesen & schreiben	
Speicherart	flüchtig, permanent	

Halbleiterspeicher - Übersicht



Anwendung:

Register
Cache
FIFO

Haupt-
speicher

Programm-Speicher
Daten-Speicher

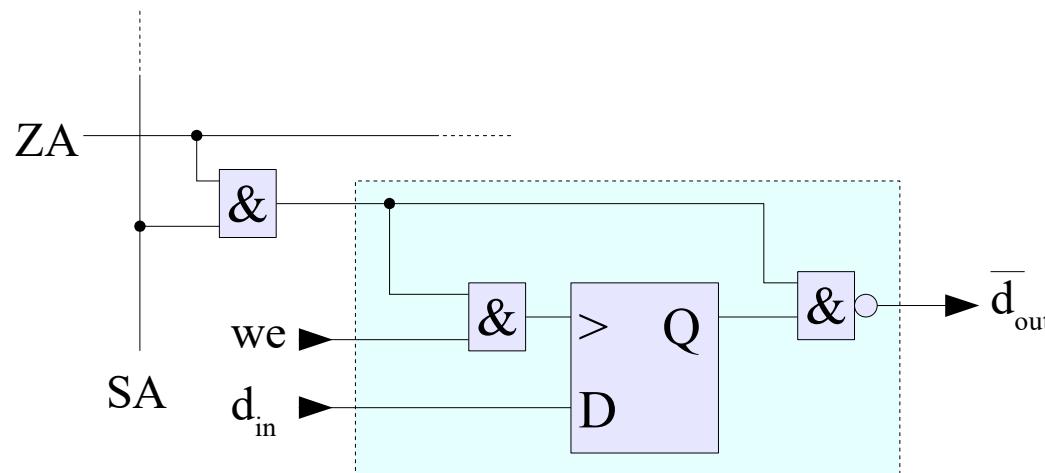
Struktur eines Halbleiterspeichers (SRAM)

- Logische Ersatzschaltung einer SRAM-Speicherzelle

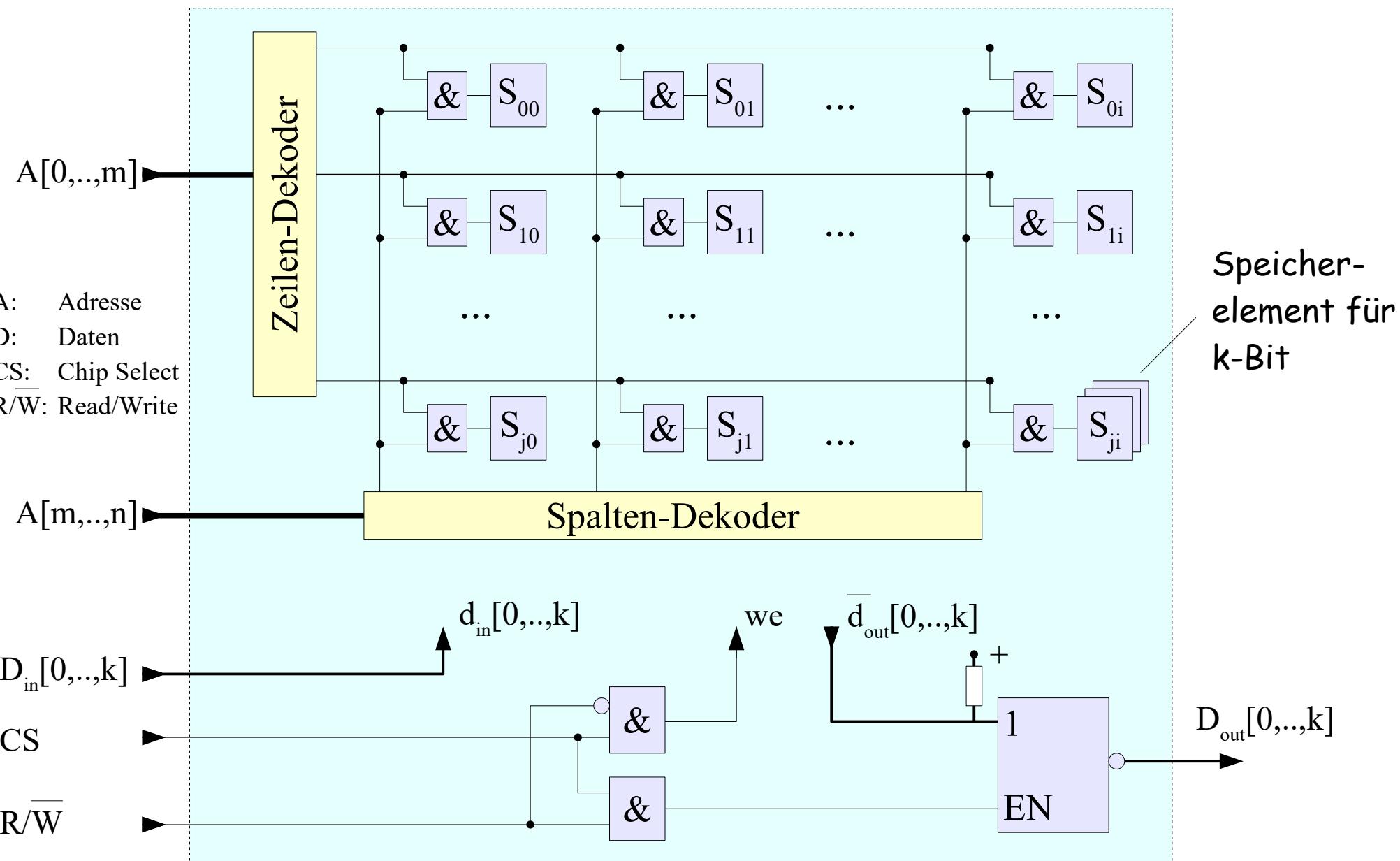
- Die Eingänge d_{in} und Ausgänge \bar{d}_{out} aller Speicherzellen sind miteinander verbunden.
Dies erfordert einen hochohmigen Daten-Ausgang (Open-Collector)
- Auswahl der Speicherzelle mit Zeilen- und Spaltenauswahl (ZA und SA)
- Schreiben: Mit $we = 1$ wird d_{in} in die durch ZA und SA selektierten D-Flip-Flops geschrieben
- Lesen: Die mit ZA und SA selektierte Speicherzelle setzt den Ausgang:

$$Q = 1: \bar{d}_{out} = 0$$

$$Q = 0: \bar{d}_{out} = \text{hochohmig}$$



Struktur eines Halbleiterspeichers (SRAM)



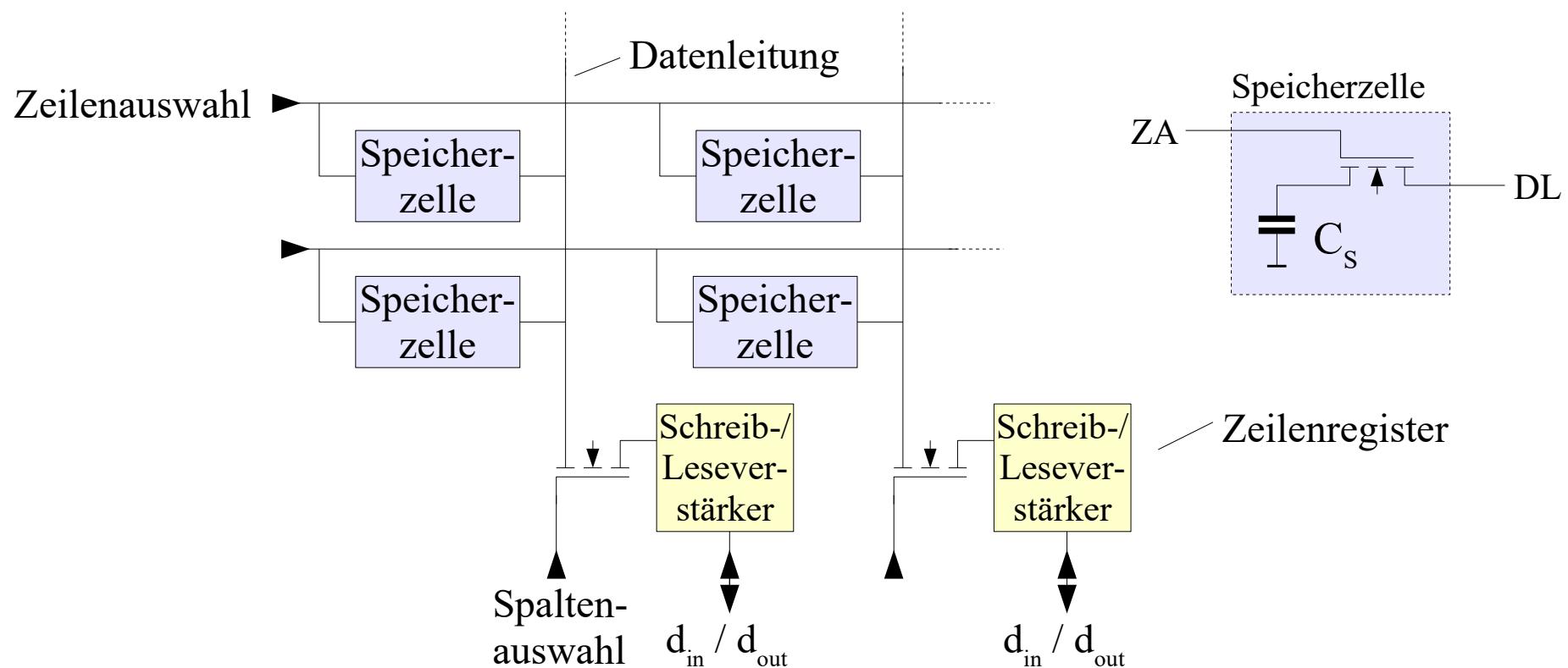
Eigenschaften (SRAM)

- Static RAM (SRAM)
 - Taktfrequenz: >250 MHz
 - Zugriffszeiten: < 8 ns
 - Kapazität: bis zu 32 MBit
 - Geringe Verlustleistung
 - Mit Batteriepuffer auch als nichtflüchtiger Speicher verwendbar
 - Geringe Speicherkapazität pro Chipfläche (im Vergleich zum DRAM)
 - Verwendung:
 - CPU-Register
 - Arbeitsspeicher in Mikrocontrollern
 - Cache-Speicher

Struktur eines Halbleiter-Speichers (DRAM)

- Dynamic RAM (DRAM)

- Information wird als Ladezustand des Kondensators C_s gespeichert
- Schaltungsaufwand: 1 Transistor / Kondensator pro Bit
- Speicherzellen müssen regelmäßig (~ 2 ms) refreshed werden
- Zusätzliche Steuerlogik erforderlich (DRAM-Controller)

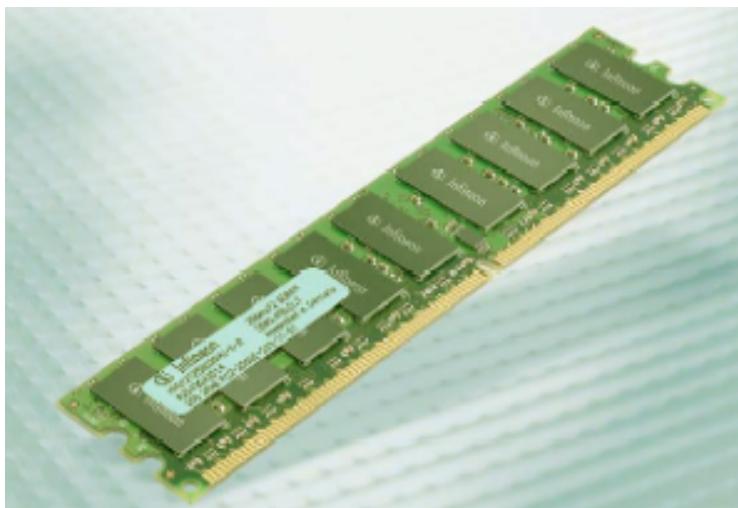


Eigenschaften (DRAM)

- Dynamic RAM (DRAM)
 - Flüchtiger Speicher
 - Inhalt aller Speicherzellen muss regelmäßig aufgefrischt werden (~2 ms). Während des Refresh ist Speicher nicht ansprechbar
 - Zugriffszeit ~ 10 ns
 - Potentiell langsamer als SRAM
 - Problem: Prozessor-Zykluszeit < Speicher-Zugriffszeit
 - Abhilfe durch interleaved memory (= verschränkte Speicheradressierung: Speicher wird auf mehrere Speicherbänke verteilt, die sich durch niederwertige Adressbits unterscheiden. Bei aufeinander folgenden Adresszugriffen ist kein Wartezyklus erforderlich)
 - Erzeugt relativ hohe Verlustleistung, insbesondere durch Refresh
 - Höhere Speicherkapazität im Vergleich zum SRAM
 - Typische Verwendung:
 - Hauptspeicher, Video-Speicher

Eigenschaften (DRAM)

- SDRAM = Synchronous DRAM
 - Datenübertragung mit steigender Taktflanke
 - Vorläufer des DDR-RAM
- DDR-RAM = Double Data Rate - DRAM
 - Datenübertragung mit steigender und fallender Taktflanke. Dadurch: doppelte Datenrate
 - Aktuelle Entwicklung: DDR5-RAM
 - Speichertakt: 300-500 MHz
 - Datentransfer : 4800-8000 MT/s
 - Kapazität: 8-64 GB



Für den Einsatz als Arbeitsspeicher werden DRAM-Speicherchips zu Modulen zusammengefasst:
SIMM = Single Inline Memory Module
DIMM = Dual Inline Memory Module

MT/s: Megatransfers pro Sekunde

Massenspeicher

- Ziele:
 - Hohe Kapazität, ggf. auch bei geringerer Zugriffsgeschwindigkeit
 - Nicht-flüchtig (non volatile) + hohe Lebensdauer (MTBF)
- Halbleiterspeicher
 - Speicherkarten, Solid-State-Harddisk, Flash-EEPROM
 - Verschiedene Standards: SD-Card, Multimedia-Card, USB-Stick, ...
- Magnetische Aufzeichnung
 - Bandlaufwerke
 - Festplatten
- Elektro-Optische Aufzeichnung
 - CD-ROMs, CD-R, DVD, ...

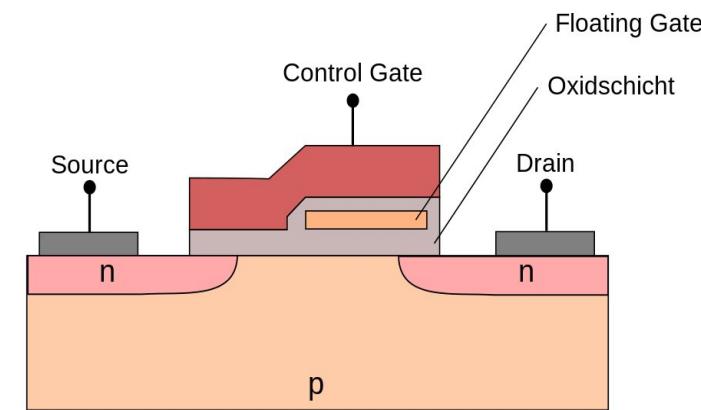
Flash-Speicher

- Nicht-flüchtiger Halbleiterspeicher

- Speicherzelle besteht ähnlich wie beim DRAM aus einem einzelnen Transistor
 - Information wird als elektrische Ladung auf einem (isolierten) Floating-Gate gespeichert
 - Lesen: Transistor leitet/sperrt je nach Ladungszustand
 - Schreiben: Ladungszustand kann durch Tunneleffekt (Elektronen hoher Energie überwinden Isolationsschicht) geändert werden
 - Aber: Anzahl der Löschzyklen begrenzt auf $< 10^4$

- Architekturen

- NAND-Flash
 - Speicherzellen werden in großen Gruppen ($\sim 1 \text{ kByte}$) zusammengefasst. Dadurch hohe Speicherdichte, aber Schreib- und Lesevorgang nur blockweise möglich
 - Anwendung: z.B. Massenspeicher
 - NOR-Flash
 - Separate Datenleitung je Speicherzelle, dadurch geringere Speicherdichte. Lesen: wahlfrei, Schreiben: „0“ wahlfrei / „1“ blockweise
 - Anwendung: z.B. Programmspeicher im Mikrocontroller



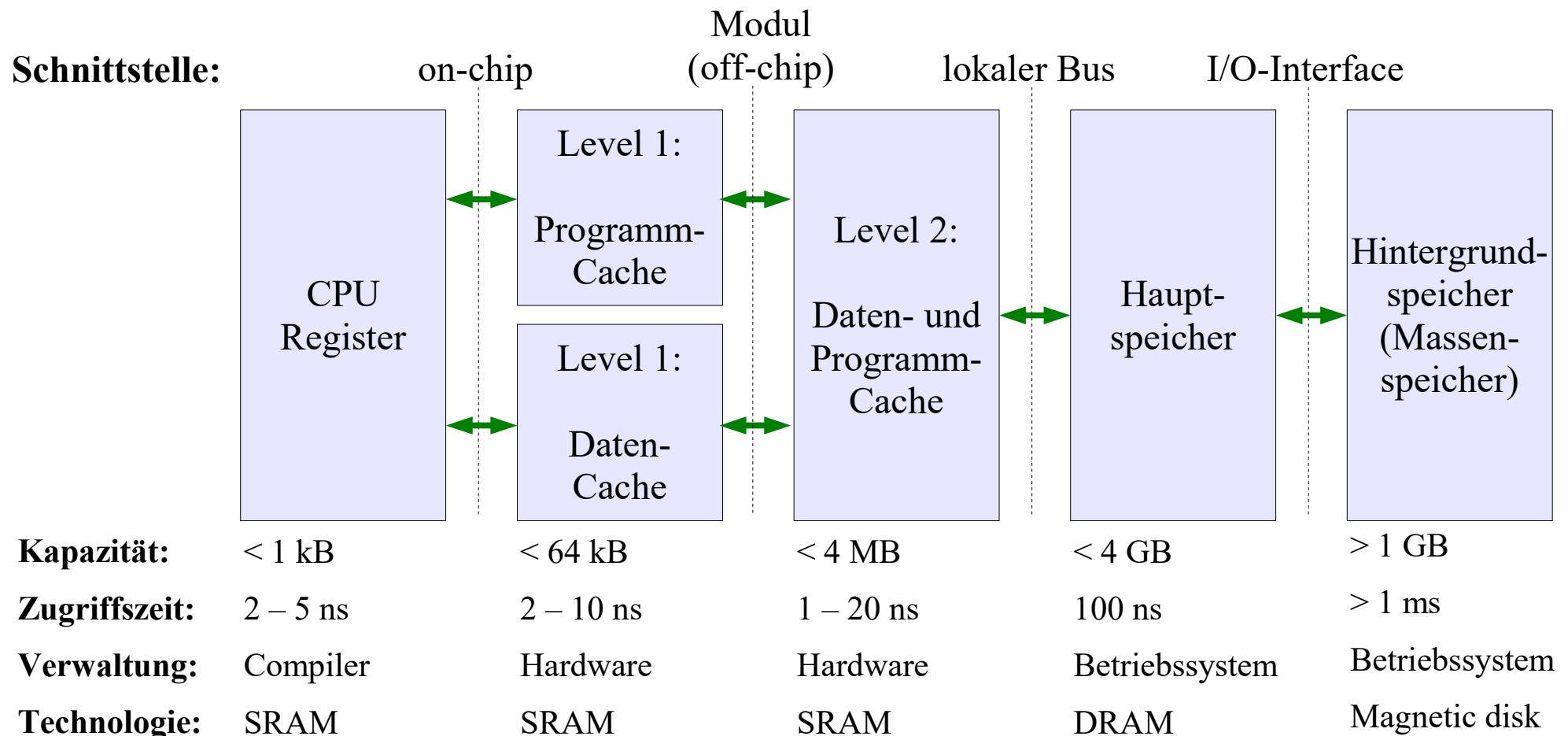
<https://www.notebookcheck.com>

Speicheroptimierung

- Speicherhierarchie (Cache):
 - Aufteilung in Hauptspeicher + Cache-Stufen
 - Probleme:
 - Aufwändige Logik zur zweckmäßigen Füllung der Caches und Sicherstellung der Cache-Kohärenz
- Datenbusverbreiterung:
 - Gleichzeitiger Zugriff auf mehrere Datenworte
 - Probleme:
 - Aufwändiger Datenbus, Optimierung nur für benachbarte Daten erreichbar
- Interleaved-Speicher:
 - Einteilung des Speichers in unabhängige Speicherbänke, die abwechselnd angesprochen werden
 - Probleme:
 - Zusätzliche Datenpfade und Zwischenspeicher erforderlich

Speicherhierarchie (Cache)

- Optimierung der Zugriffszeiten durch Speicherhierarchie
 - Langsamer, günstiger Hauptspeicher wird durch schnelle, teure Cache-Speicher ergänzt
 - Häufig verwendete Daten werden CPU-nah gepuffert



Cache-Verwaltung

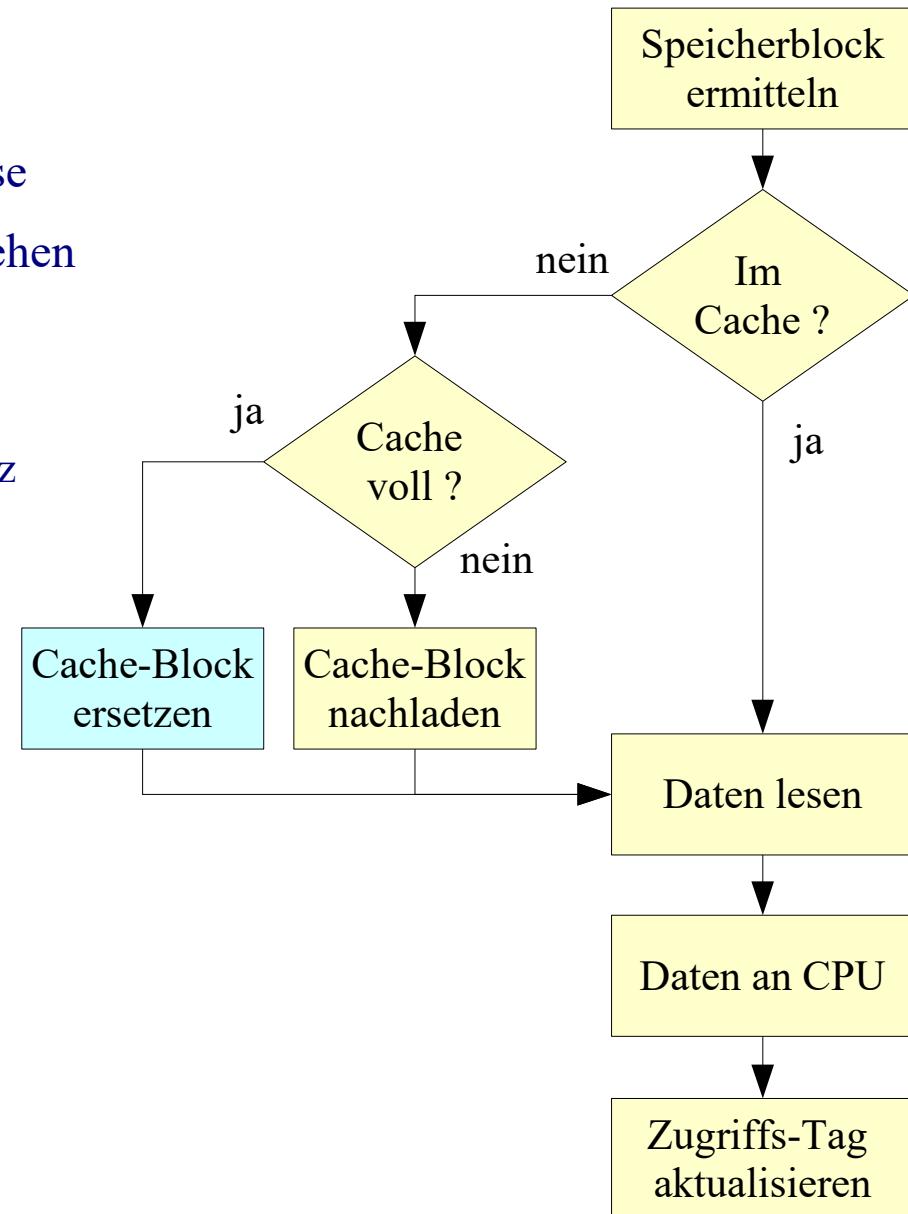
- Cache-Controller verwaltet Daten
 - beliefert CPU möglichst unmittelbar mit angeforderten Daten aus den Caches / nimmt CPU-Daten an
 - lädt neue Datenblöcke in die Caches, wenn angeforderte Daten nicht lokal verfügbar sind
 - schreibt geänderte Datenblöcke zurück
 - hat Strategien zur optimierten Verwaltung des Cache-Inhalts
 - Minimierung der Cache-Misses (min. Zugriffszeit)
 - Minimierung der Schreib- und Lese-Zugriffe (Buslast)
 - Optimierung der Speichernutzung
- Hauptspeicherzugriff erfolgt blockweise
 - höhere Zugriffsrate möglich bei entsprechender Speichertechnologie

Cache-Verwaltung: Lese-Zugriff

CPU fordert Lesezugriff auf Speicher an (1 Zyklus!)

- Vorgehensweise:

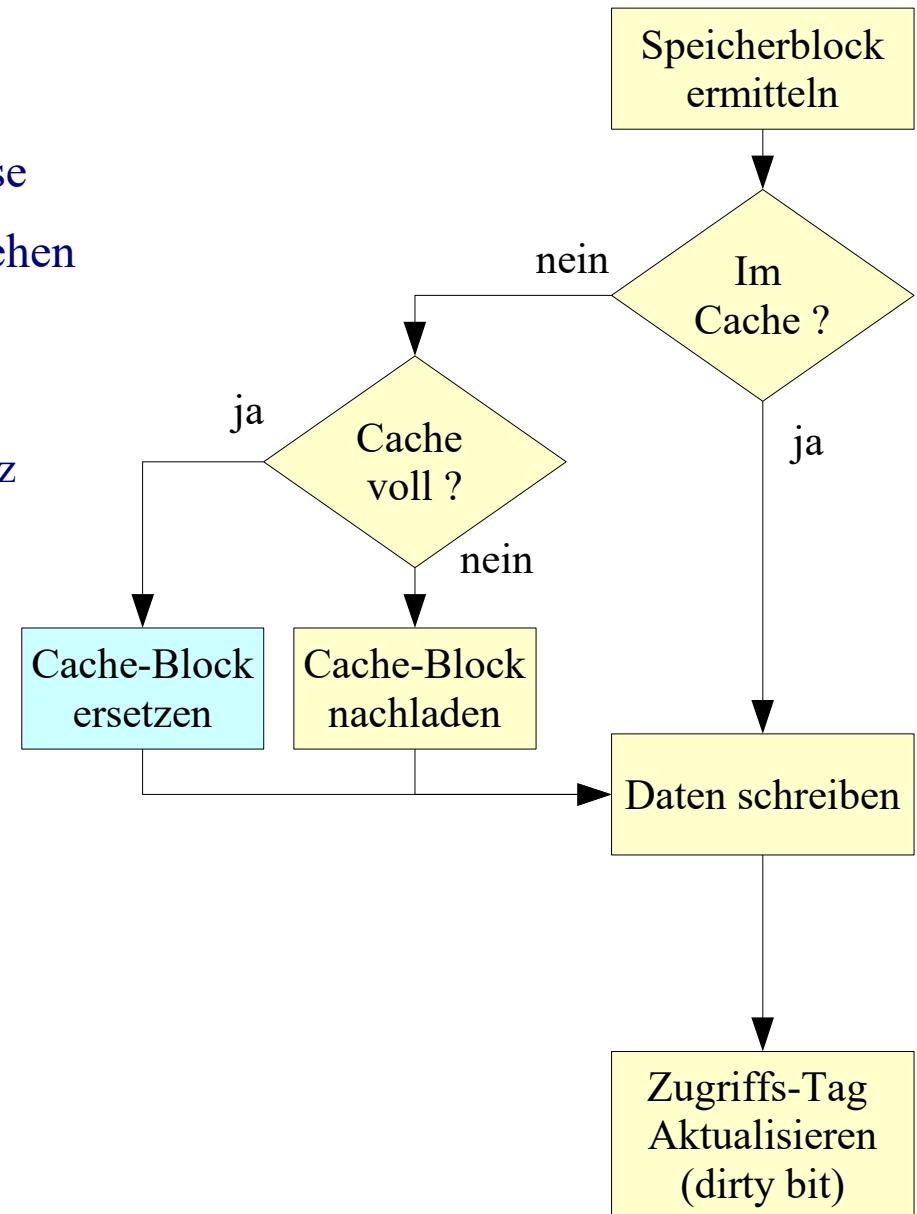
- Ermitteln des Speicherblocks anhand der Adresse
- Prüfen, ob die Daten dieses Blocks im Cache stehen
 - wenn ja (**cache hit**): Datum aus Cache lesen
 - wenn nein (**cache miss**): Laden eines neuen Cache-Blocks in den nächsten freien Cache-Platz
- kein freier Cache-Platz - Cache voll?
siehe: Übernächste Folie
- Daten an CPU schicken
- Zugriffs-Tags des Blocks aktualisieren



Cache-Verwaltung: Schreib-Zugriff

CPU fordert Schreibzugriff auf den Speicher an

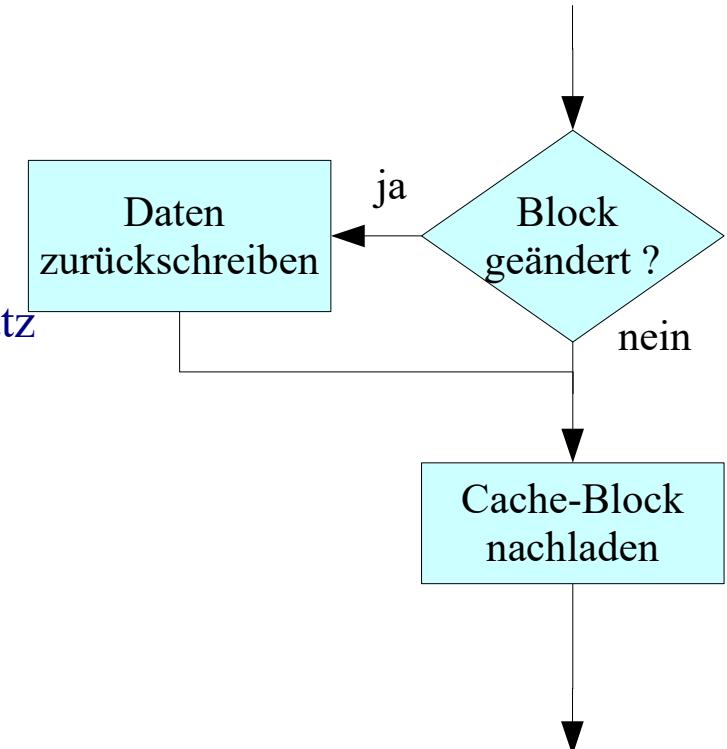
- Vorgehensweise:
 - Ermitteln des Speicherblocks anhand der Adresse
 - Prüfen, ob die Daten dieses Blocks im Cache stehen
 - wenn ja (**cache hit**): Datum in Cache schreiben
 - wenn nein (**cache miss**): Laden eines neuen Cache-Blocks in den nächsten freien Cache-Platz
 - kein freier Cache-Platz - Cache voll?
siehe: Nächste Folie
 - Zugriffs-Tags des Blocks aktualisieren,
"dirty bit"



Cache-Verwaltung: Ersatz von Cache-Lines

Cache-Verwaltung: Ersatz von Cache-Lines

- Vorgehensweise:
 - Prüfen, ob die Daten des Blocks durch die CPU verändert wurden ("dirty Bit" des Blocks gesetzt?)
 - wenn ja: zunächst Zurückschreiben des Cache-Blocks in den Hauptspeicher (**copy back**)
 - Laden des neuen Speicher-Blocks in den freien Cache-Platz



Cache-Verwaltung: Ersatz von Cache-Lines

- Problem:
 - Cache ist voll gefüllt, ein neuer Datenblock muss aufgenommen werden:
 - Welcher Block soll entfernt werden?
- Strategien:
 - LRU (Least Recently Used)
 - Der am längsten ungenutzte Block
 - LFU (Least Frequently Used)
 - Der am wenigsten genutzte Block wird ersetzt
 - FIFO (First In First Out)
 - Der am längsten im Speicher gehaltene Block
 - Random:
 - Ein beliebiger Block wird ersetzt