TECHNISCHE UNIVERSITÄT DRESDEN

Analyse aktueller Cache-Architekturen hinsichtlich Struktur und Effizienz

Markus Krause

Dresden, 11.02.2015

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

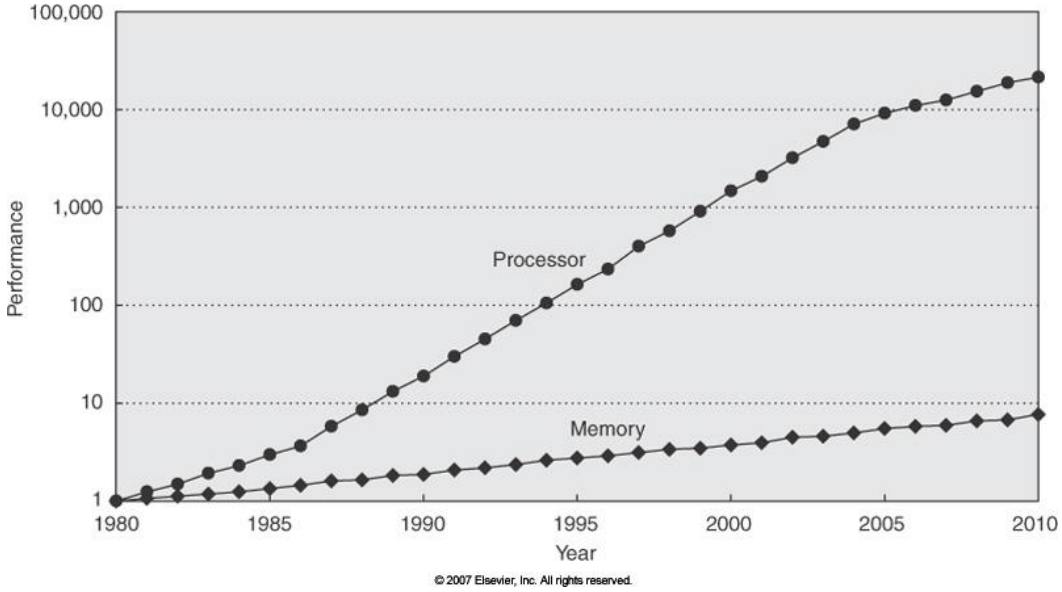
Gliederung

1. Einführung
2. Problemstellung
3. Lösungen
    a) Miss Rate
    b) Miss Penalty
    c) Hit Time
4. Zusammenfassung
5. Quellen

Einführung

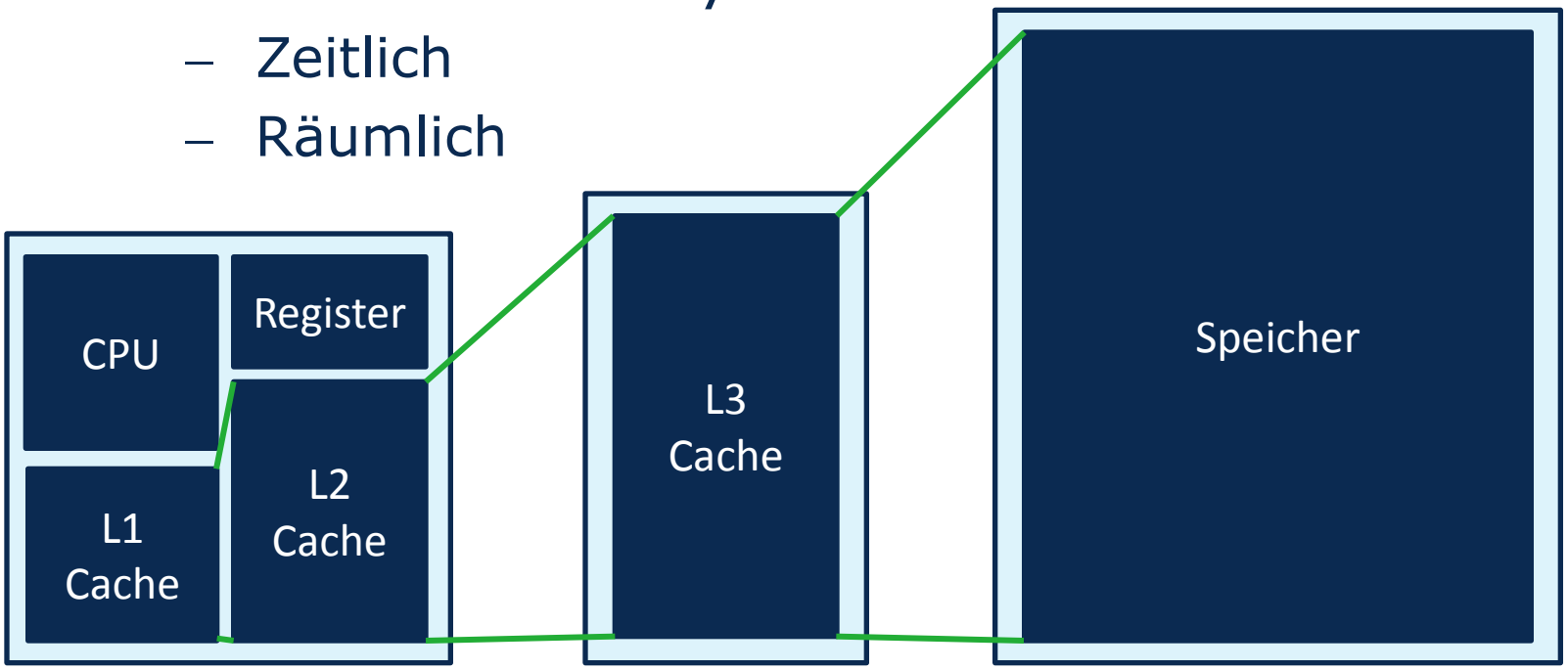
Wieso verwenden wir überhaupt Cache?

- CPU-Memory-Gap wächst immer weiter



Einführung

- Cache arbeitet mit Hilfe von Locality
- 2 Arten von Locality:
  - Zeitlich
  - Räumlich



Problemstellung

Die 3 entscheidenden Größen bei Cache-Architekturen:

1. Miss Rate
2. Miss Penalty
3. Hit Time

$$Miss\ Rate = \frac{Anzahl\ Misses}{Gesamtanzahl}$$

$$Miss\ Penalty = Gesamtzugriffszeit - Hit\text{-}Time\text{-}L1\text{-}Cache$$

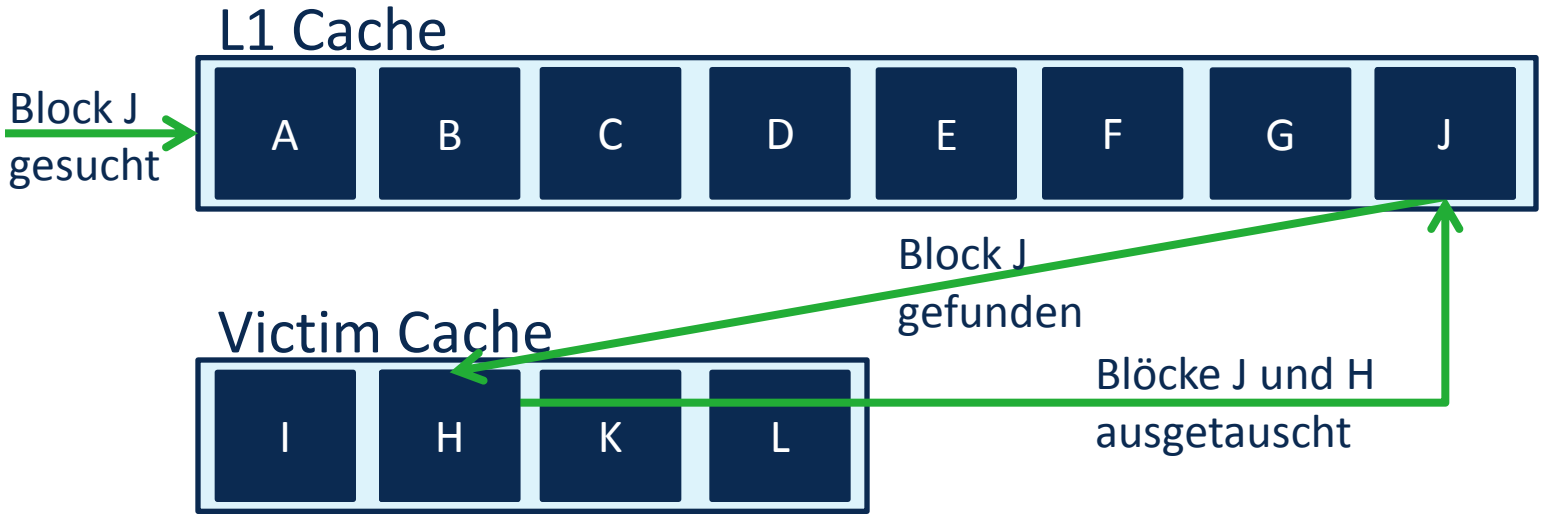
Problemstellung

Warum aber nicht einfach 1-stufiger Cache???

- riesige Miss Penalty
- deswegen großer L1
- aber langsame Hit Time
- deswegen mehrstufiger Cache

Lösungen – Miss Rate

Einsatz eines Victim Cache

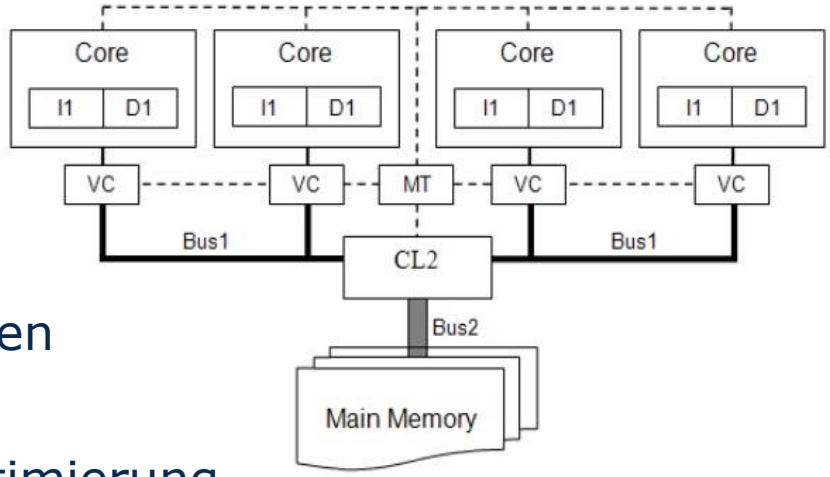


- bis zu 95% Konfliktreduzierung

Lösungen – Miss Rate

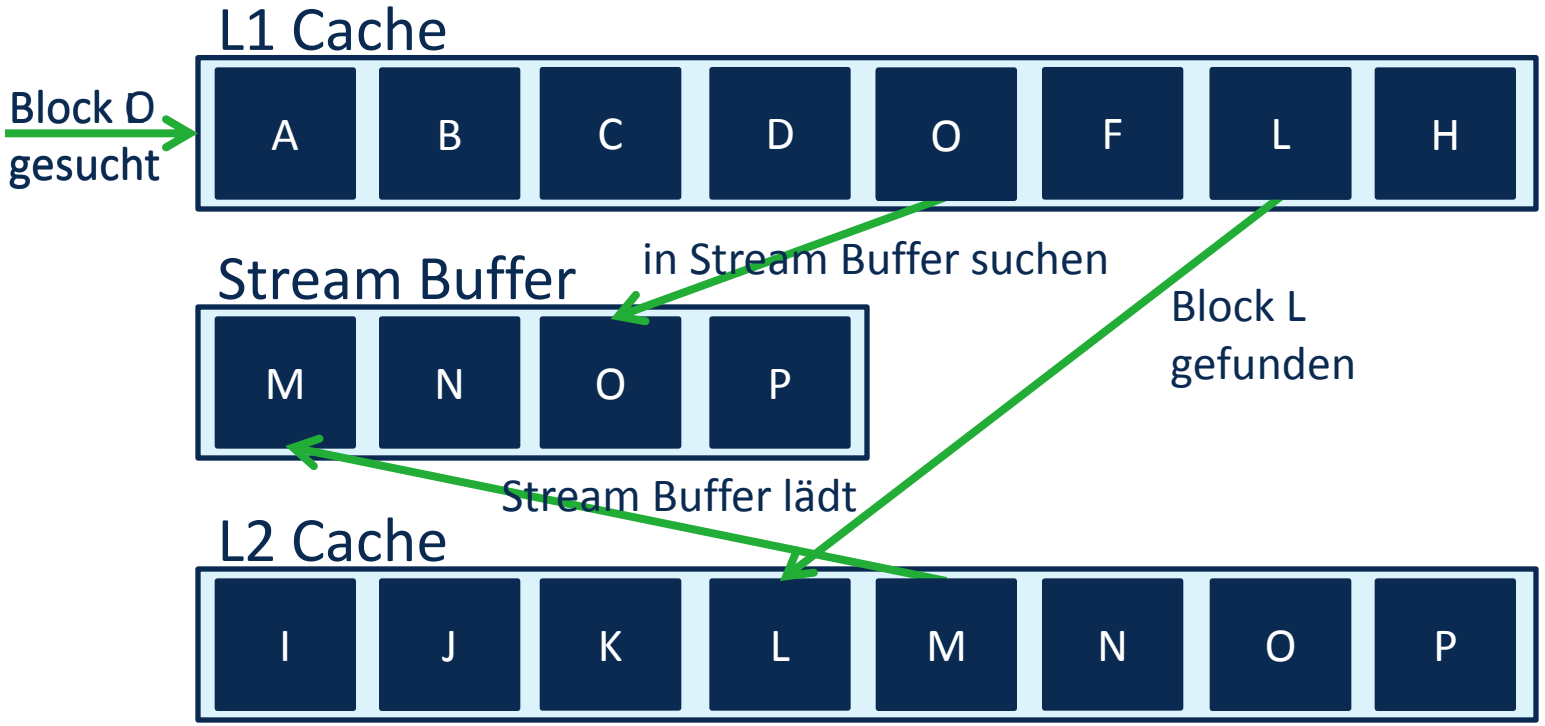
Verbesserungen des Victim Cache

- Miss Table:
  - locked einzelne Blöcke
  - bis zu 31% Miss Rate- und 38% Energieverbrauch-Reduzierung
  - nur für größere Datenmengen
- Selektiver VC:
  - Algorithmus basierte Optimierung
  - 21% Verbesserung der Miss Rate
  - 70% Reduktion des Blockaustauschs



Lösungen – Miss Rate

Stream Buffer



Lösungen – Miss Penalty

Read over Write – Write Buffer

- 2 Möglichkeiten für Schreibvorgang
  - Write-Through
  - Write-Back
- Einsatz eines Write Buffers führt zu RAW Konflikten – bis zu 50% größere Penalty
- Verhalten des Write Buffers anpassen
  - zuerst in Buffer schauen
  - Daten von L2 in Write Buffer kopieren

Lösungen – Miss Penalty

Early Restart und Critical Word First

- praktisch bei großen Blöcken
- räumliche Locality möglicherweise Problem

- **Early restart**

1 -> 2 -> 3 -> 4

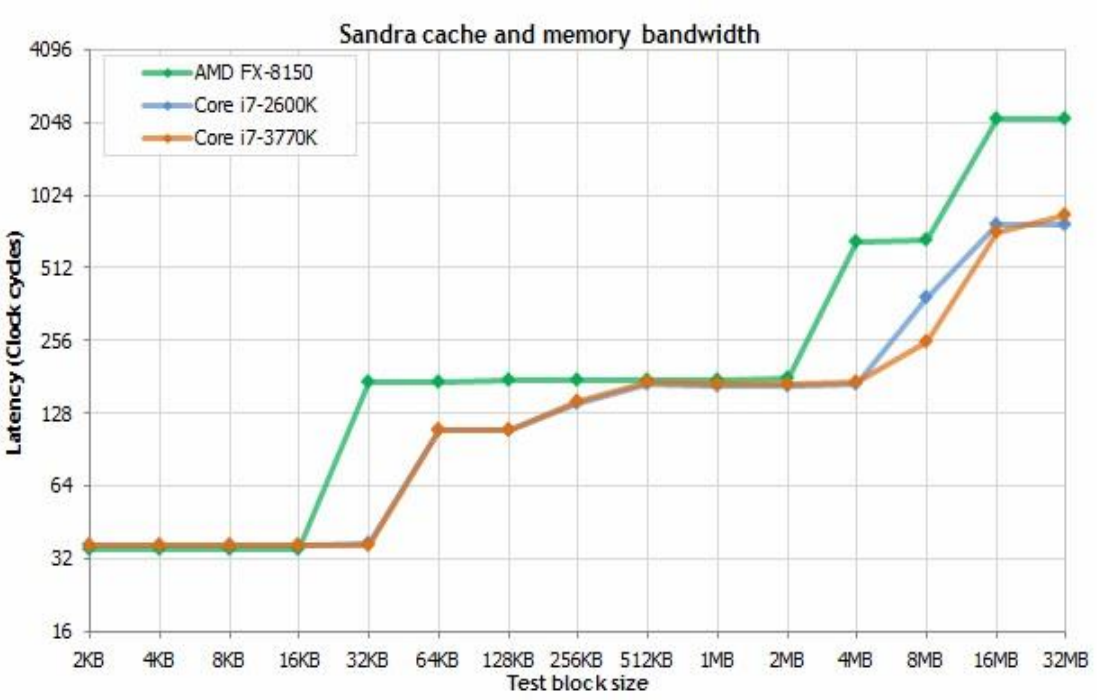
- **Critical word first**

3 -> 4 -> 1 -> 2

Lösungen – Miss Penalty

CAM Zellen

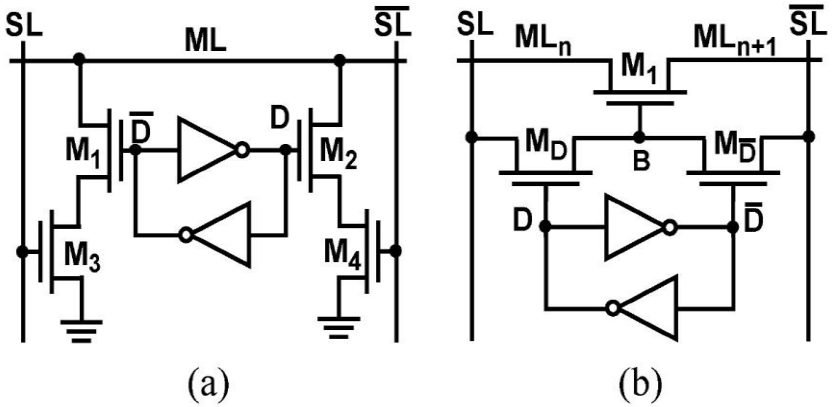
- bei AMD zwar größerer Cache
- jedoch besonders im L3 deutlich langsamer
- möglicher Grund ist der Einsatz von CAM Zellen



Lösungen – Miss Penalty

CAM Zellen

- prinzipiell 2 Aufgaben:
  - Speicher
  - Vergleicher
- besteht aus 9 – 10 T
- aber größere Leistungs-aufnahme und quadratisches Delay
- Trade-Off zwischen Chip-Fläche und Miss Penalty



Lösungen – Hit Time

Small and Simple Caches

- kleiner, direct mapped L1 Cache
- erhöht Miss Rate
  - bei L1 nicht so entscheidend
- wird in praktisch allen heutigen Prozessoren verwendet

Lösungen – Hit Time

Avoiding Address Translation

- viele Prozessoren setzen auf virtuelle Adressierung des Speichers
- phyischer TAG und virtueller Index
- Problem der Homonyme und Synonyme
  - aufgrund von Synonymen mittlerweile komplett physische Adressierung in größeren Caches
  - Trade Off zwischen Geschwindigkeit und Konsistenz

Lösungen – Hit Time

Pipelined Writes

- Hit Time bezieht sich auch auf Write
- im ersten Takt Adress-Tag Vergleich
- im zweiten Takt schreiben und nächster Vergleich

Zusammenfassung

- viele verschiedene Möglichkeiten zur Cache Optimierung vorhanden
  - Victim Cache und Optimierungen
  - Write Buffer und CAM Zellen
  - kleiner einfacher L1 und pipelined Writes

Quellen

- http://web.sfc.keio.ac.jp/~rdv/keio/sfc/teaching/architecture/architecture-2008/hennessy-patterson/Ch5-fig02.jpg
- http://cdn1.coresites.mpora.com/whitelines_new/wp-content/uploads/2014/01/homer-simpson.jpg
- alle Quellen aus schriftlicher Ausarbeitung

Zusatz

Non-Blocking Cache

- trotz Miss weiter arbeiten
- auch mehrere Misses möglich
- Out-Of-Order Completion nötig