

Cache Simulation und Analyse

Emina Sljivic | Dominik Weber | Matilda Briegel Technische Universität München

Garching, 27. August 2024





Einleitung

- Gründe für einen Cache
 - Von-Neumann-Flaschenhals (Limitierungen bzgl. Datenübertragung zwischen CPU und RAM)
 - Latenzzeiten RAM & Cache¹
 - L1 ~4 Zyklen
 - L2 ~10 Zyklen
 - L3 ~40 Zyklen
 - Lokaler RAM ~60 Zyklen
- Cache Mapping
 - Direkt abbildender Cache
 - Vollassoziativer Cache
 - (Mengenassoziativer Cache)

[1] https://www.intel.com/content/dam/develop/external/us/en/documents/performance-analysis-guide-181827.pdf



Problemstellung

- Gegenüberstellung von direkt abbildendem und vollassoziativem Cache
 - · Cache Replacement Policy (FIFO, LRU, ...)
- Vorgaben
 - Write Through
- In-Order
- Byte-adressierbarer Speicher

Direkt abbildender Cache

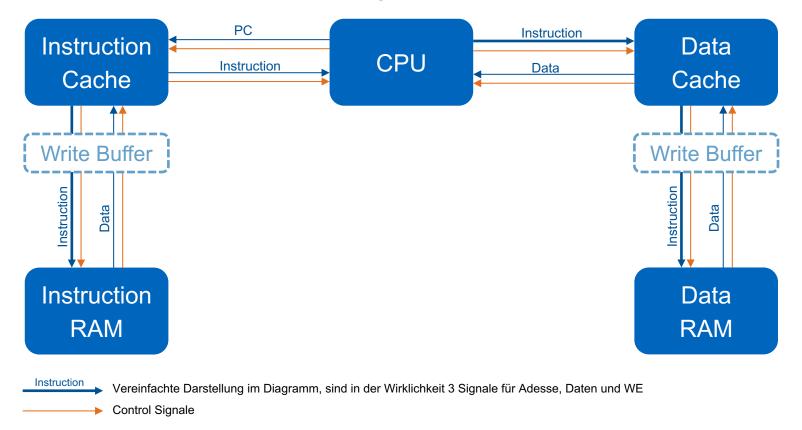
Tag	Daten	Adr.	Daten
		1	
		2	

Vollassoziativer Cache

Tag	Daten		Adr.	Daten
			1	
			2	
		£		

Struktur von Implementierung







Funktionsweise Cache - Ablauf

- 1. Aufteilung von Request in Sub-Requests mit auf die Cachezeilen Größe ausgerichteten Adressen
- 2. Ausführen von einzelnen Sub-Requests
 - 1. Kalkulierung von Cacheline abhängig vom Cache Mapping
 - 2. Überprüfen über Tag ob relevante Daten bereits im Cache sind
 - 3. HIT/MISS
 - HIT: Daten sind bereits im Cache
 - MISS: Daten aus der n\u00e4chsten Stufe lesen.
 - 4. Schreiben oder Daten in die CPU zurückgeben
 - 5. (Schreiben) Weitergabe des Sub-Requests an den Write Buffer zur Bearbeitung im RAM

Nach: David A. Patterson, John L. Hennessy - Computer Organization And Design: The Hardware/Software Interface, RISC-V Edition



Default Werte von Cache Attributen

Cachezeilen Größe von Direkt abbildenden Caches¹

L1: 16 KB bis 64 KB

L2: bis zu 1 MB

Vollassoziative Caches²

- Chachezeilen Größe
 - L1: 8 KB bis 64 KB
 - Sind kleiner als Direkt abbildenden Caches, da sie für TLBs benutzt werden
- Cachezeilen
 - L1: 32 bis 256 Einträge
 - L2: bis zu 2048

^[1] S. Kumar und P. K. Singh - An overview of modern cache memory and performance analysis of replacement policies [2] Intel, Intel® 64 and IA-32 Architectures Optimization Reference Manual: Volume 1

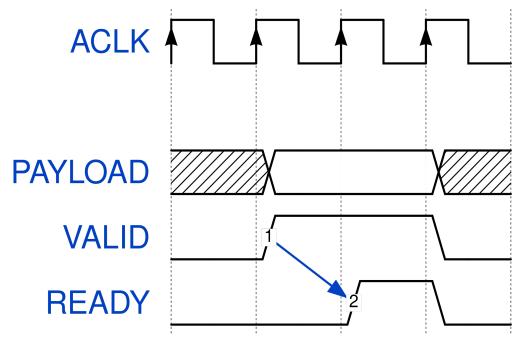


Lösungsansätze

- Warten im RAM
 - Warten am Anfang vom Bearbeiten der Abfrage
 - Dank Row-Buffer -> direkt subsequentes Lesen mehrerer Worte
- Byte adressierbarer Speicher
 - Aufteilung auf Sub-Requests mit auf die Cachezeilen Größe ausgerichteten Adressen
 - Je Adresse ein Byte
 - Beim Schreiben: Aufteilung von einem Integer in 4 Bytes
- Intermodulkommunikation
 - Variation von Ready/Valid Protokoll

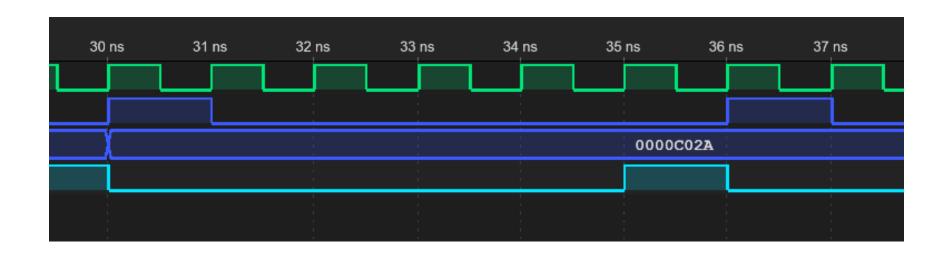


Ready / Valid Protokoll





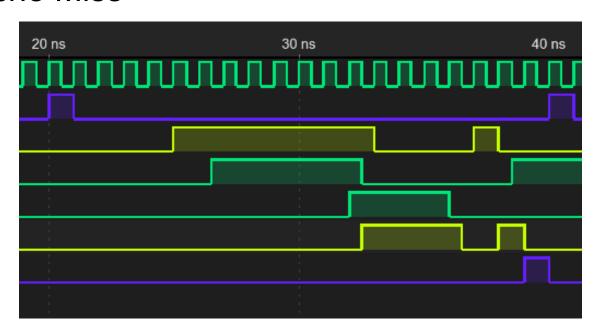
Kommunikation





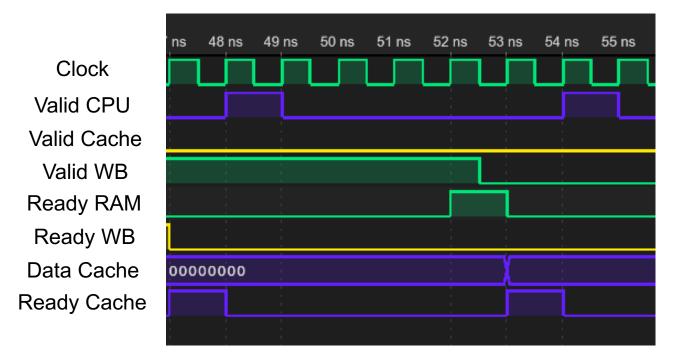
Write Cache Miss

Clock
Valid CPU
Valid Cache
Valid WB
Ready RAM
Ready WB
Ready Cache





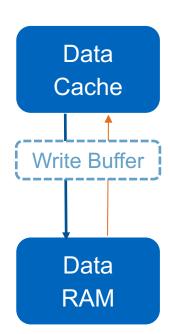
Cache Hit während Write





Optimierungen

- Write Buffer
 - 3 Modi: Optimiert, Lesen vor Schreiben, Disabled
 - State Machine
 - Starke Beschleunigung für lokale Algorithmen
 - Extra-Kommunikation
- Lookup-Table f
 ür vollassoziativen Cache
 - Realisierbar auf Intel Stratix 10 GX 2800 FPGA¹
 - 2 extra Zyklen
- Pipelining
 - Volle Ausnutzung der Caches durch Harvard-Architektur

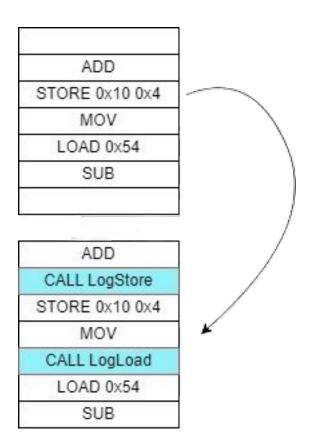


[1] Zhang et al. (2023). A High Throughput Parallel Hash Table on FPGA using Reversible XOR-based Memory



Simulations-Eingabengenerierung

- Merge Sort & Radix Sort
 - Einlesen
 - Sortieren
 - In volatile Variable Schreiben
- Analyse über LLVM-Pass
 - Während Kompilierung instrumentieren
 - Jeder Memory Access aufgezeichnet
- Normalisierung
 - 32 Bit Adressen
 - 32 Bit Daten





Korrektheitsüberprüfung

- Unit Tests
 - Prüfung der einzelnen Komponenten
 - Mocks
- Integration Tests
 - Prüfung des gesamten Systems
 - Edge-Cases (0 Latenz, 1GB Cache-Line Size, ...)
 - Kombinationen => über 14k Tests
- "Blackbox Fuzzing"
 - Emuliert durch zufällige Eingabegenerierung

```
100% tests passed, 0 tests failed out of 14400

Total Test time (real) = 2682.40 sec
```



Merge/Radix-Sort

```
mergeSort(List 1, int from, int to) {
  if (l.size() <= 1) return;
  int mid = (from + to)/2;

  mergeSort(l, from, mid);
  mergeSort(l, mid+1, to);

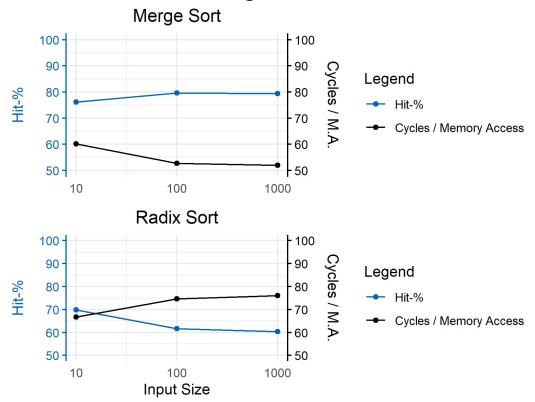
  merge(l, from, to);
}</pre>
```

```
radixSort(Sequence s) {
  for (int i = 0; i < d; i++)
    kSort(s,i);
}

Sequence kSort(Sequence s) {
  Sequence[] b = new Sequence[K];
  foreach (e ∈ s)
    b[key(e)].pushBack(e);
  return concatenate(b);
}</pre>
```

Algorithm



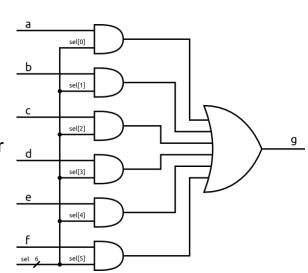


Direct Mapped, Cache Latency 20c, Memory Latency 100c, Cache-Line Size 16B, 20 Cache-Lines



Schaltkreisanalyse

- Gatteranzahl: ca. 6M für LRU Fully Associative
- Hash-Table, benutzt in Vollassoziativität + LRU
 - Auf Intel Stratix 10 GX1800 FPGA realisierbar¹, ca. 2753000 Gatter
- Cache-Tabelle
 - Pro Bit 4 Gatter -> (Cacheline-Size+Tag Bits)*#Cachelines*8*4
- Multiplexer
 - Ca. #Cachelines*Cacheline-Size*8 AND Gates
 - Cacheline-Size*8 OR Gates
- Subrequest-Splitting
 - Ca. 700
- ...



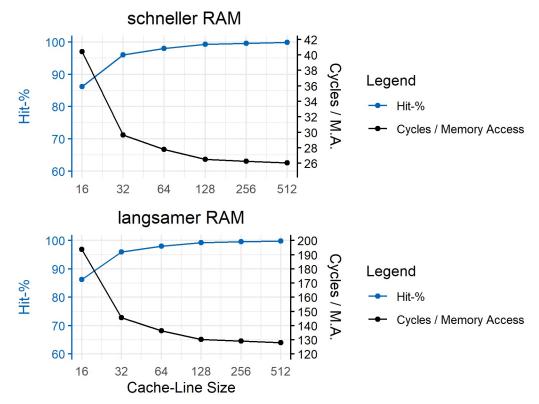


Vergleich mit "echtem Cache"

- Gil et al. "Reconfigurable Cache Implemented on an FPGA," 2010 International Conference on Reconfigurable Computing and FPGAs
 - xc3s200-ft256 FPGA => 200000 Gatter
 - Bloß 16 KB Cache
 - Unsere Hash-Tables: für Optimierung großer Caches

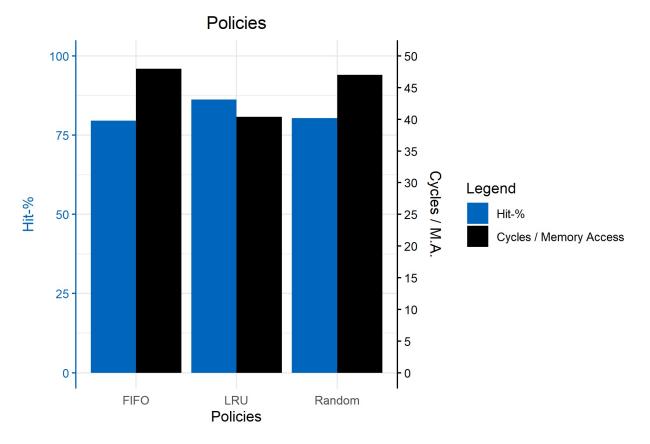
Cache-Line Size



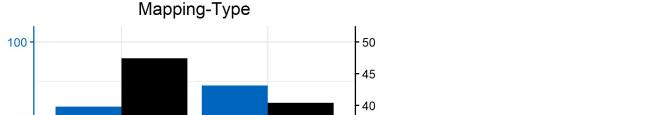


Fully associative, LRU, Merge Sort [100], Cache Latency 5c, Memory Latency 100c/500c, 8 Cache-Lines

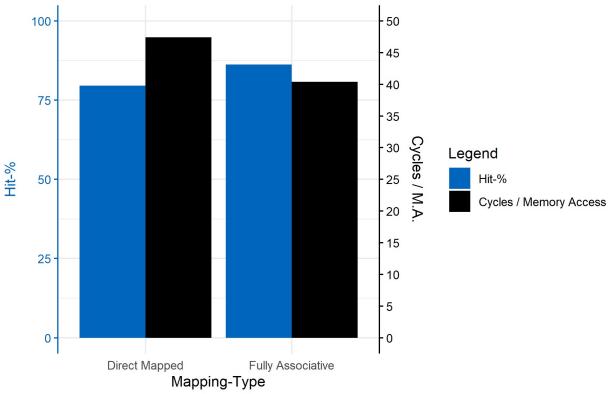




Fully Associative, Merge Sort [100], Cache Latency 5c, Memory Latency 100c, 16 Cache-Lines, Cache-Line Size 16B



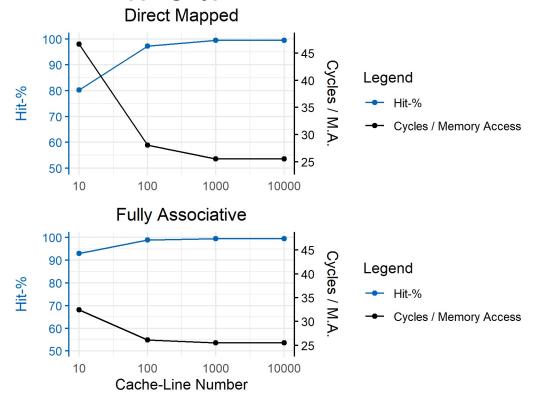




Merge Sort [100], LRU, Cache Latency 5c, Memory Latency 100c, 8 Cache-Lines, Cache-Line Size 16B

Mapping-Type / Cache-Line Number

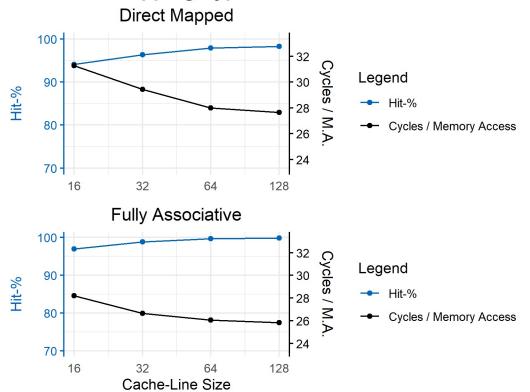




Merge Sort [100], LRU, Cache Latency 5c, Memory Latency 100c, Cache-Line Size 16B



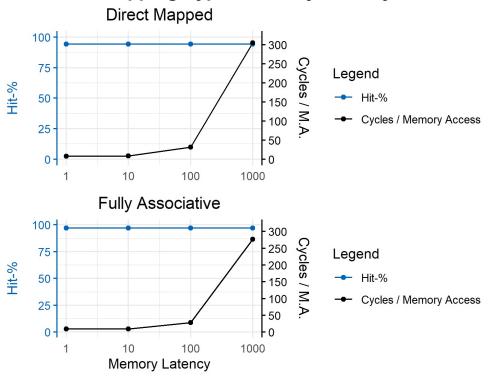
Mapping-Type / Cache-Line Size



Merge Sort [100], LRU, Cache Latency 5c, Memory Latency 100c, 32 Cache-Lines

Mapping-Type / Memory Latency

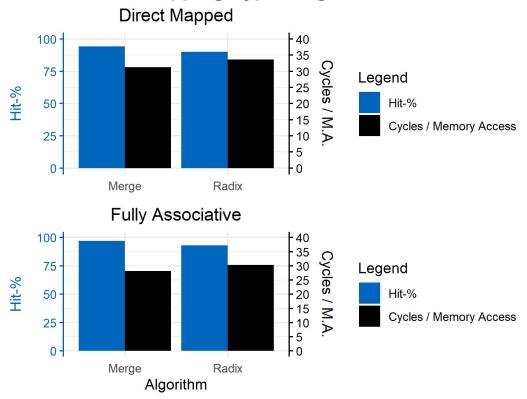




Merge Sort [100], Cache Latency 5c, 32 Cache-Lines, Cache-Line Size 16B

Mapping-Type / Algorithm

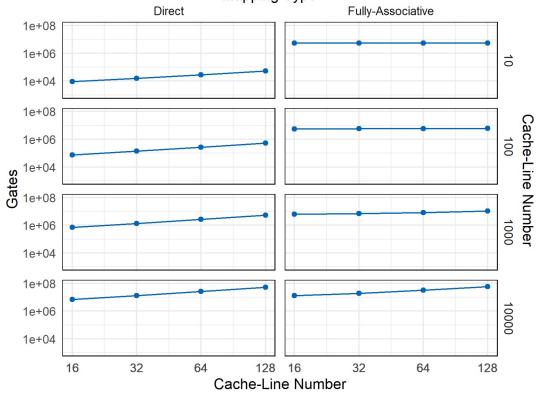




[100], LRU, Cache Latency 5c, Memory Latency 100c, 32 Cache-Lines, Cache-Line Size 16B

Auswirkung Cache-Line Size und Zahl auf Gates Mapping-Type

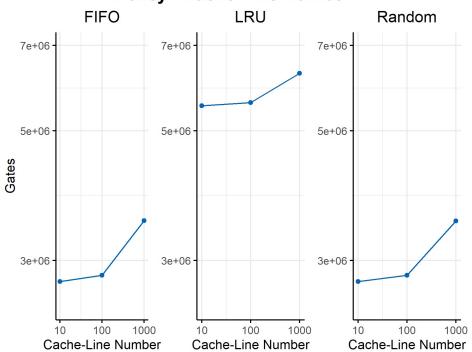




Merge Sort [100], Cache Latency 5c, Memory Latency 100c



Policy / Cache-Line Number



Fully Associative, Merge Sort [100], Cache Latency 5c, Memory Latency 100c, Cache-Line Size 16B



Ausblick

- Weitere Optimierungen
 - Multilevel-Caches
 - Gleichzeitiges Lesen und Schreiben in Cache
 - Burst mode
 - Instruktionen ohne Write Through
 - Victim Buffer
- Alternative Lösungsansätze
 - Von Intermodulkommunikation ausgehend designen



Zusammenfassung

- Caches sind f
 ür moderne Prozessoren essenziell
- Vollassoziative Caches mit ausgeklügelten Replacement Policies in Simulation besser als Direct
 - Aber: Viel mehr Gatter
 - => Hit-% steht immer in Konflikt mit Gatterzahl / Komplexität
- Vergrößern von Cache-Lines / Erhöhung Zahl unterliegt Gesetz des sinkenden Grenzertrags
- Speicherlokale Algorithmen profitieren von Caches stärker als nicht-lokale