# Grundlagenpraktikum: Rechnerarchitektur SS24

“Direkt abbildender vs 4-fach assoziativer Cache"

Bunzel Janick, Jurcenko Michael, Knoll Julian

# Gliederung

1. Problemstellung
2. Lösungsansätze und Umsetzung
3. Korrektheit bzw. Genauigkeit
4. Analyse
5. Fazit und Ausblick
6. Quellen

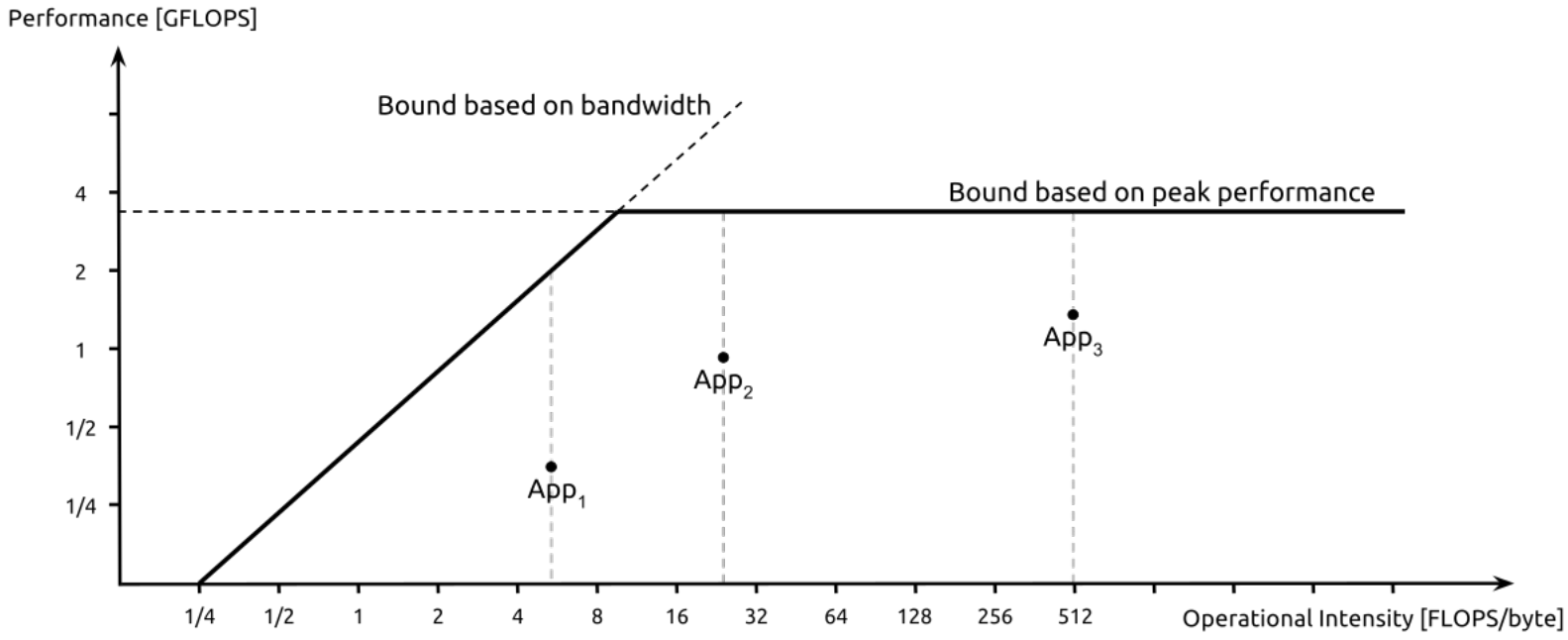
# 1. Problemstellung

## I. Aufgabenstellung

- Unterschiede zwischen direkt abbildendem und vierfach assoziativem Cache untersuchen.
- Recherche
- Implementierung
- Dokumentation

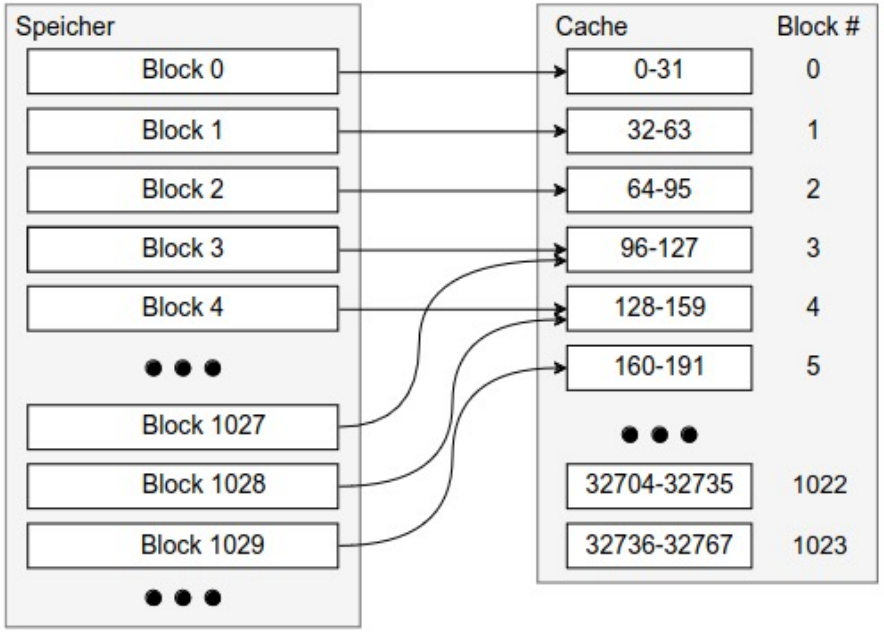
# 1. Problemstellung

## II. Von-Neumann-Flaschenhalses



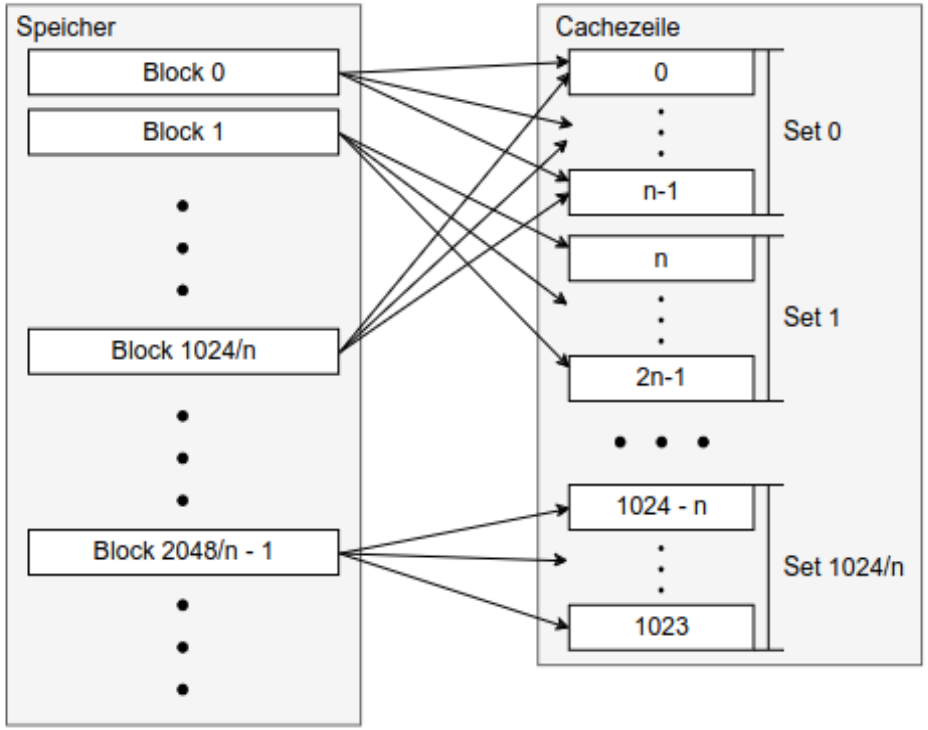
# 1. Problemstellung

## III. Primitiver Ansatz (direkt abbildend)



# 1. Problemstellung

## IV. 4-fach assoziativer Cache



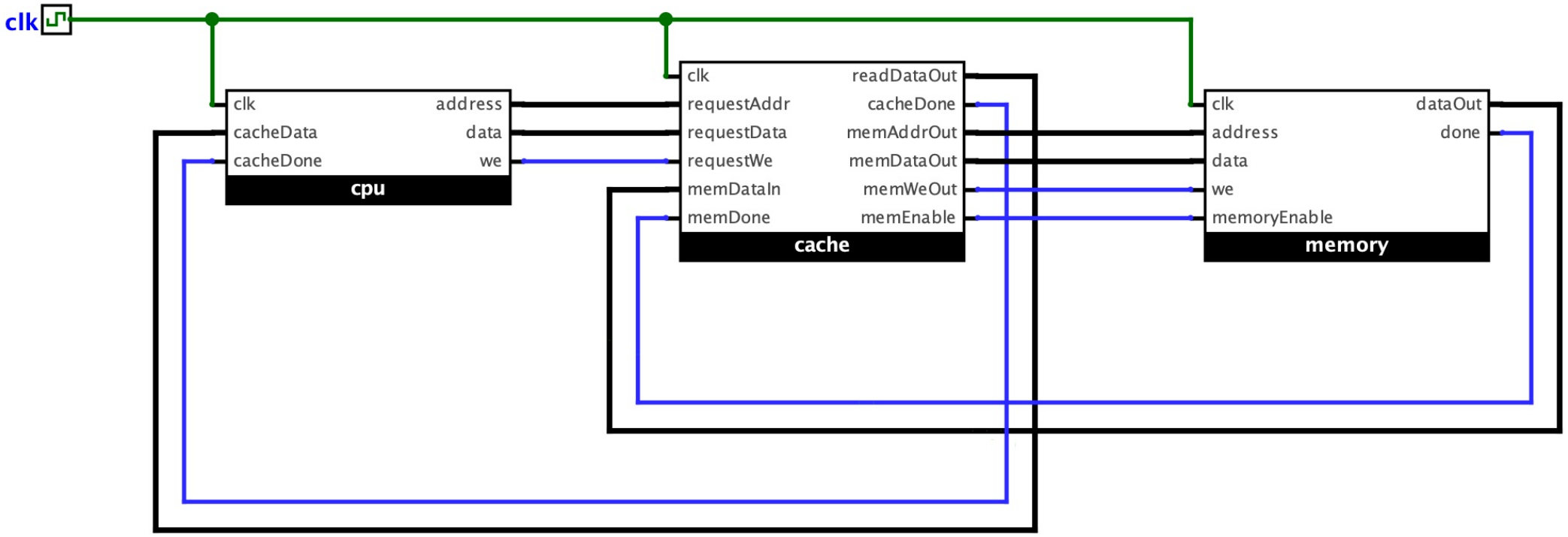
# 1. Problemstellung

## V. Resultate der Literaturrecherche

- **Gängige Cachegrößen:** L1-Cache (2KB-64KB), L2-Cache (256KB-512KB), L3-Cache (1MB-8MB)
- **Zugriffszeiten:** L1-Cache 1-4 Zyklen, L2-Cache 10-16 Zyklen, L3-Cache 30-40 Zyklen
- **Reale Beispiele:** Intel Prozessor, Eingebettete Systeme, Echtzeitsysteme

# 2. Lösungsansätze und Umsetzung

## I. Implementierung in Logisim Evolution



# 2. Lösungsansätze und Umsetzung

## II. Abstraktionslevel in SystemC

**Module:** CPU, Cache, Speicher

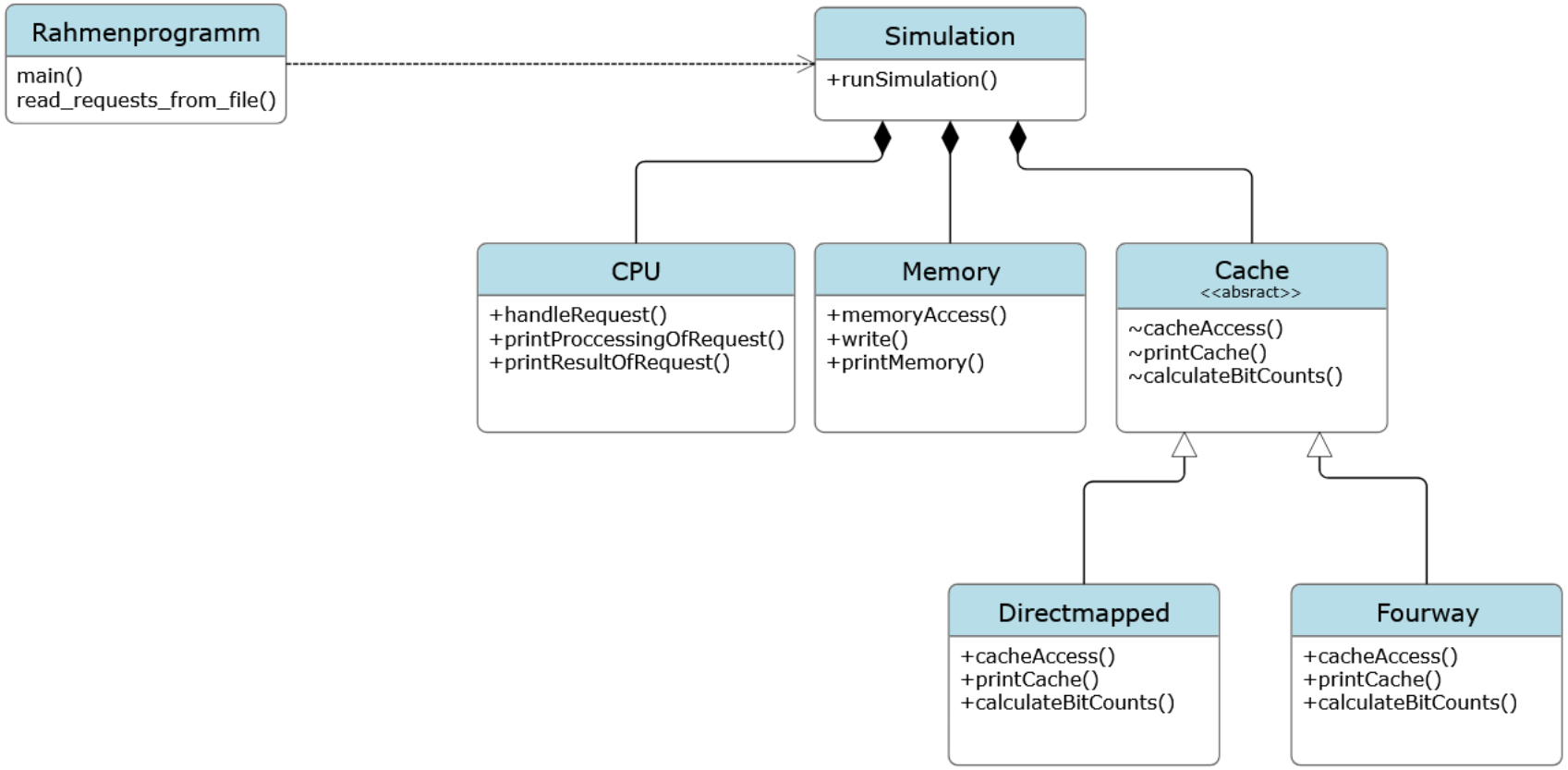
**Logik:** C++ Code für Funktionalität

**Signale:** Kommunikation zwischen den Modulen

**Synchronisation:** Delta-Zyklen für zeitlicher Abstimmung

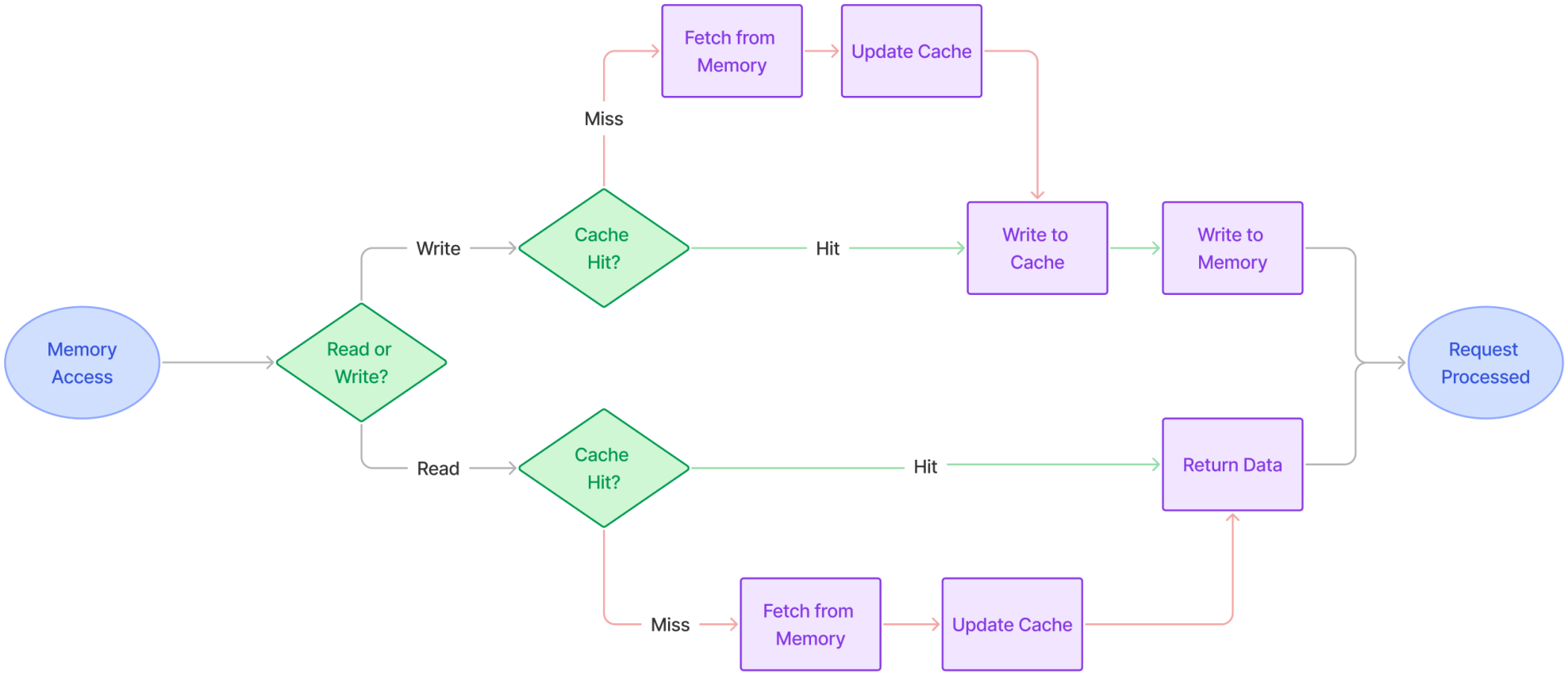
# 2. Lösungsansätze und Umsetzung

## III. Implementierung



# 2. Lösungsansätze und Umsetzung

## IV. Implementierung



# 3. Korrektheit bzw. Genauigkeit

## I. Testen und Überprüfen der Simulation

```
[Cpu]: request[0]: W,0xaffe,0xaaaaffff

Cache State:
-------------------------------
Index   Tag                 Valid   Data
-------------------------------
0       0x000000580         1       0xff 0xff 0x00 0x00
1       0x000000000         0       0x00 0x00 0x00 0x00
2       0x000000000         0       0x00 0x00 0x00 0x00
3       0x000000000         0       0x00 0x00 0x00 0x00
4       0x000000000         0       0x00 0x00 0x00 0x00
5       0x000000000         0       0x00 0x00 0x00 0x00
6       0x000000000         0       0x00 0x00 0x00 0x00
7       0x00000057f         1       0x00 0x00 0xaa 0xaa
-------------------------------
```

# 3. Korrektheit bzw. Genauigkeit

## I. Testen und Überprüfen der Simulation

```
[Cpu]: request[0]: W,0xaffe,0xaaaaffff

Cache State:
-----------------------------------------------
Set     Way     Tag             Valid   LRU     Data
-----------------------------------------------
0       0       0x00001600      1       3       0xff 0xff 0x00 0x00
0       1       0x00000000      0       0       0x00 0x00 0x00 0x00
0       2       0x00000000      0       0       0x00 0x00 0x00 0x00
0       3       0x00000000      0       0       0x00 0x00 0x00 0x00
-----------------------------------------------
1       0       0x000015ff      1       3       0x00 0x00 0xaa 0xaa
1       1       0x00000000      0       0       0x00 0x00 0x00 0x00
1       2       0x00000000      0       0       0x00 0x00 0x00 0x00
1       3       0x00000000      0       0       0x00 0x00 0x00 0x00
-----------------------------------------------
```

# 3. Korrektheit bzw. Genauigkeit

## II. Skalierte Tests

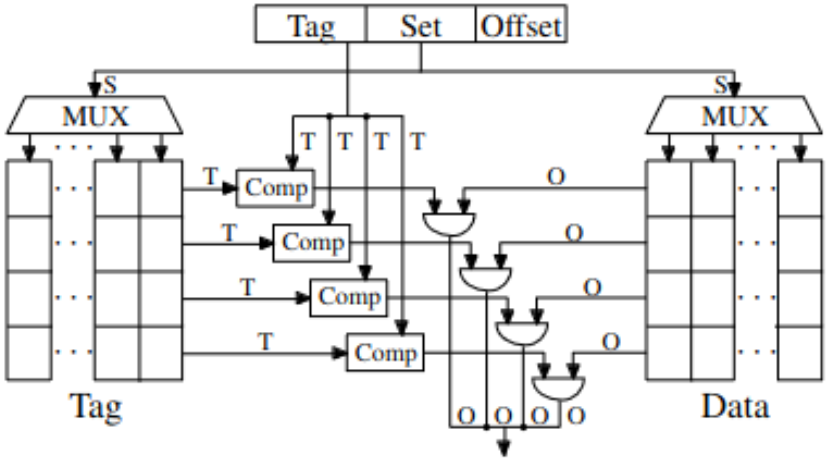
- Analyse diverser Zugriffsmuster
  - Hit/Miss vorhersehbar
  - Anzahl der Zyklen vorhersehbar
- Bsp: Iterieren durch ein Array mit 100 Elementen

```
Simulation Result:
 - Cycles: 4500
 - Cache misses: 100
 - Cache hits: 0
 - Primitive gate count: 90138
```

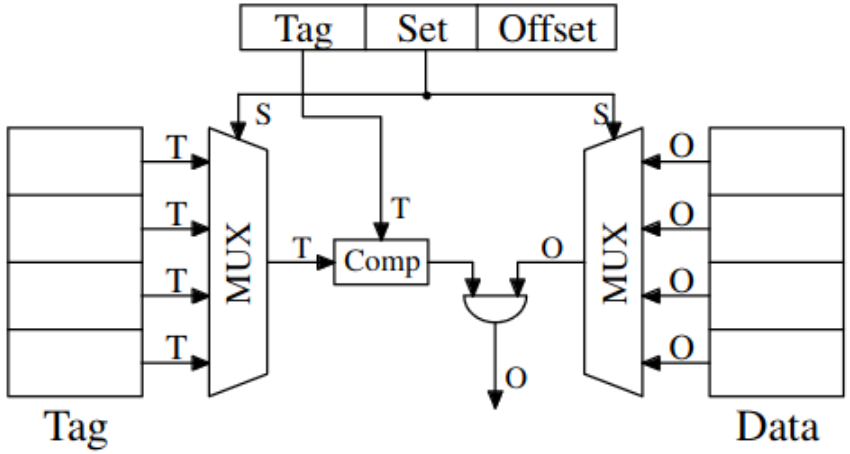
```
Simulation Result:
 - Cycles: 2500
 - Cache misses: 50
 - Cache hits: 50
 - Primitive gate count: 120346
```

# 4. Analyse

## I. Schaltkreis



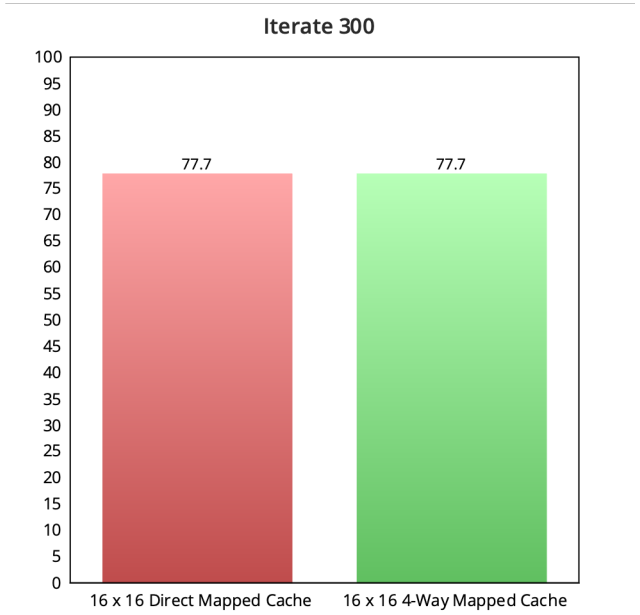

4-Fach assoziativ



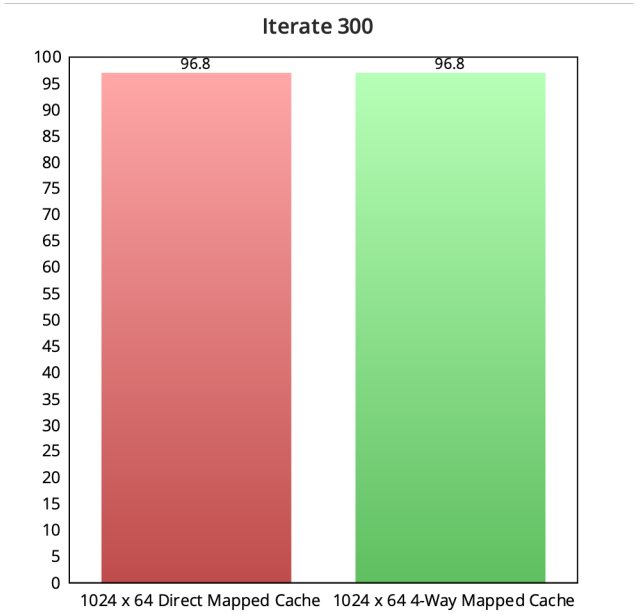

Direkt abbildend

# 4. Analyse

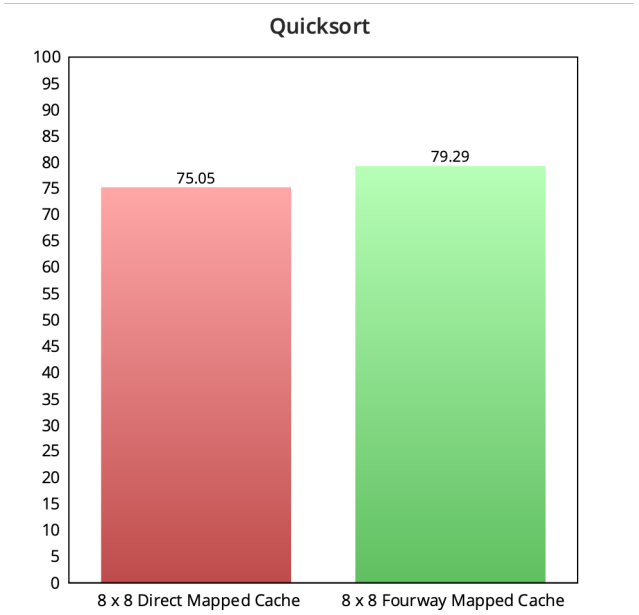
## II. Stärke direkt abbildender Cache




256byte Cache



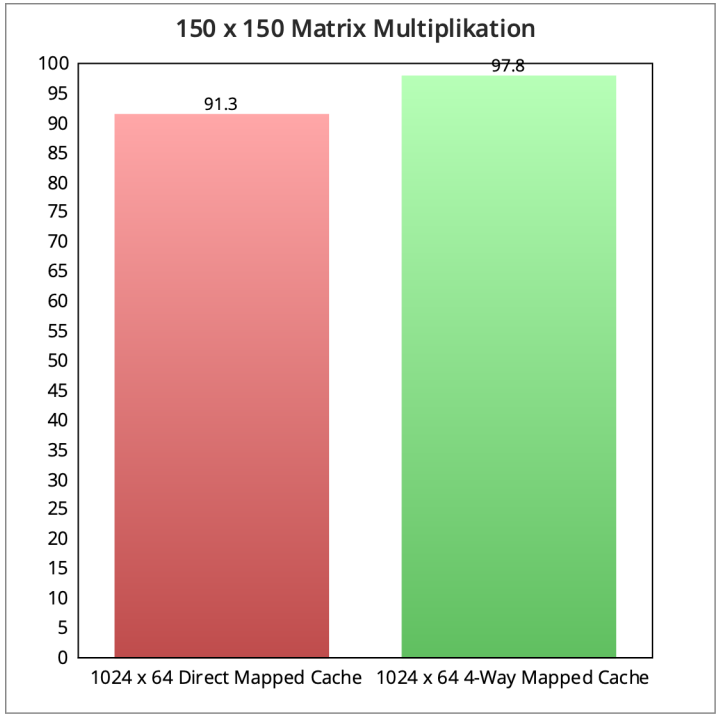

64kB Cache (L1 – Cache)



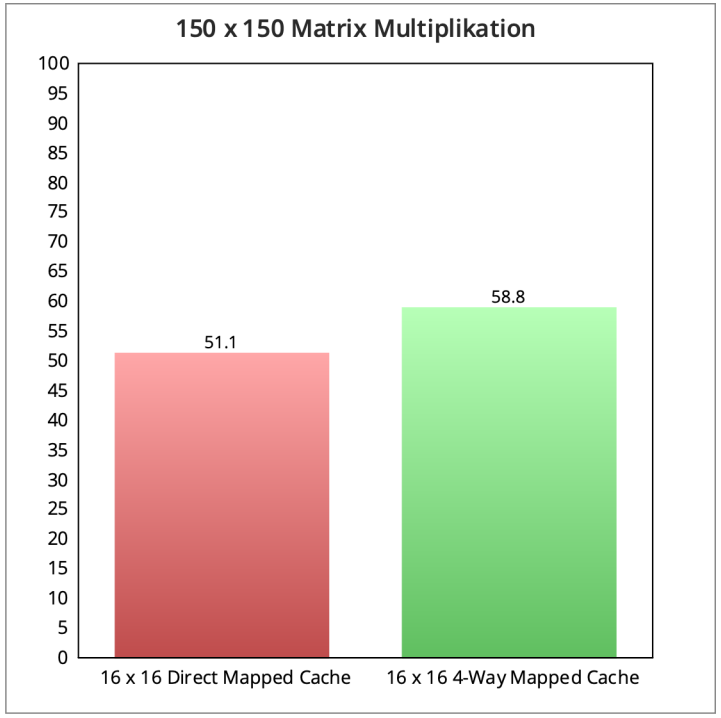

64byte Cache

# 4. Analyse

## II. Stärke 4-fach assoziativer Cache




64kB Cache (L1 – Cache)




256byte Cache

# 4. Analyse

## III. Speichereffizienz

# 5. Gegenüberstellung

- **Hitrate:** In den meisten realen Fällen hat ein 4-fach assoziativer Cache mehr Treffer als ein direkt abbildender und ist somit effizienter

- **Komplexität:** Direkt abbildender Cache ist einfacher zu implementieren und benötigt weniger Hardware

- **Zugriffszeit:** Direkt abbildender Cache ermöglicht schnellere Zugriffe aufgrund der einfacheren Struktur

- **Flexibilität:** 4-fach assoziativer Cache bietet mehr Flexibilität bei der Platzierung von Datenblöcken

# 6. Quellen


- https://en.wikipedia.org/wiki/Roofline_model
- https://people.freebsd.org/~lstewart/articles/cpumemory.pdf
- https://aws.amazon.com/de/caching/
- https://www.intel.com/content/www/us/en/products/details/processors/core/i9/docs.html?q=smart%20cache&s=Relevancy
- https://www.intel.com/content/www/us/en/products/details/processors/core/i7/products.html
- https://db.in.tum.de/teaching/ss20/dataprocessingonmodernhardware/MH_2.pdf?lang=de
- https://mecha-mind.medium.com/demystifying-cpu-caches-with-examples-810534628d71
- https://meribold.org/2017/10/20/survey-of-cpu-caches/
- https://www.tecchannel.de/a/so-funktioniert-dram,401175,8
- https://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2018_2019/ep-1819-wannags-cache-ausarbeitung.pdf