

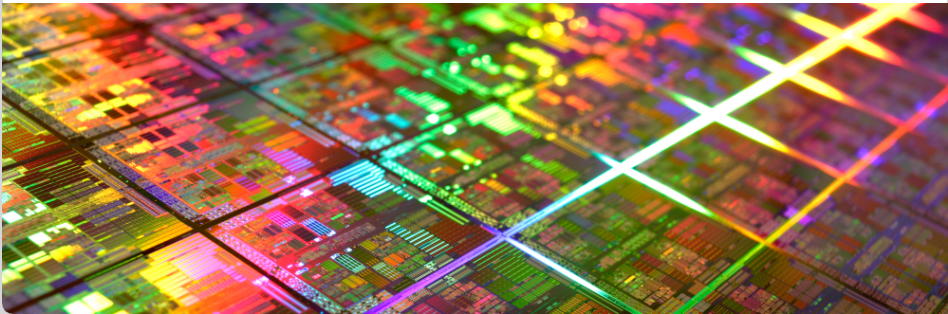
7. Zentralübung Rechnerstrukturen im SS 2013

Cache-Kohärenz

Mario Kicherer, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

15. Juli 2013



Inhalt der 7. Übung

- Motivation für Caches
- Implementierungsarten von Speicherzellen
- Cacheorganisation
- Cachekoheränzprotokolle
 - MESI
 - MOESI
- DSM-Systeme
- Organisatorisches zur Klausur
- Nachtrag Übung Superskalartechnik & Pipelining

Memory Bottleneck

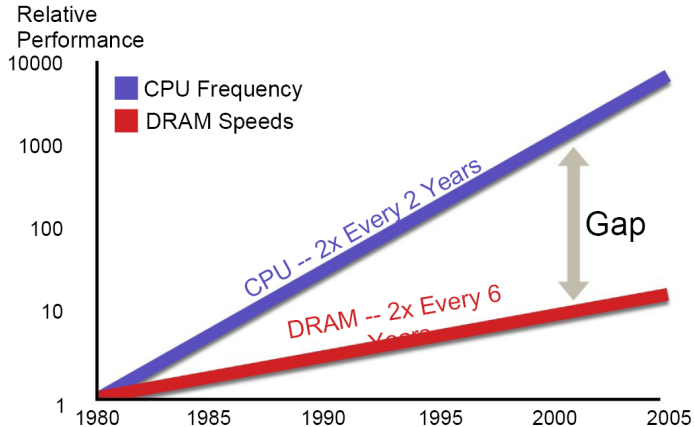


Abbildung : Quelle: <http://blogs.sun.com/toddjobson/resource/>

Cache - Motivation (forts.)

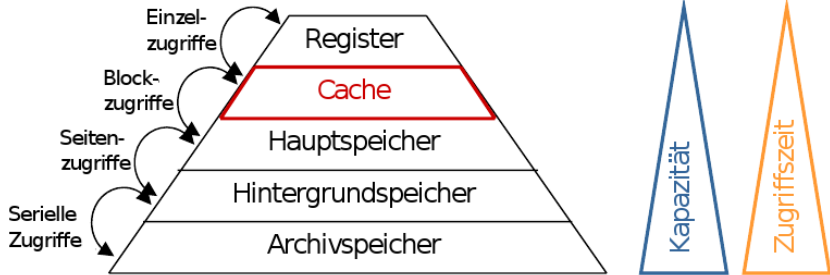


Abbildung : Speicherhierarchie

- Ausnutzung der räumlichen und zeitlichen Lokalität von Anwendungen
- 90/10 - Regel

Cache - Aufbau

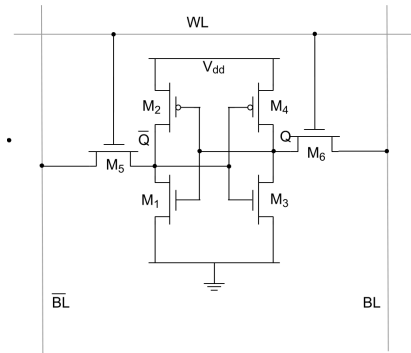


Abbildung : 6T-SRAM-Zelle

- Größe: über $100 F^2$
- Sehr schnell

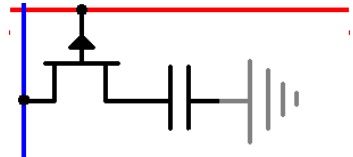


Abbildung : DRAM-Zelle

Quellen: Wikipedia

- Größe: ca. $6 - 10 F^2$
- Destructive Read
- Refreshzyklen notwendig

Organisation

- Direkt abgebildet (direct mapped)
- Satzassoziativ (set associative)
- Vollassoziativ (fully associative)

Hierarchie

- Mehrere Level (L1, L2, L3)
- im Multiprozessorfall: private vs. shared
- inclusive vs. exclusive

Größe

Charakterisiert durch:

- Cachelinesize
- # Cachelines bzw. Assoziativität * # Sätzen

Ersetzungstrategie

- least recently used (LRU)
- least frequently used (LFU)
- Round Robin
- Random

Schreibstrategie

- write-through vs. write-back
- write-allocation vs. no write-allocation

Cachekohärenzprotokoll (im Multiprozessorfall)

- Busbasiert
- Verzeichnisbasiert

Aufgabe 1.1: Cacheleistung

Cache-Performance

- Hit-Rate r_H
- Miss-Rate $r_M = 1 - r_H$
- Zugriffszeit bei Hit t_H
- Zugriffszeit bei Miss t_M

Mittlere Zugriffszeit

$$t_a = \underbrace{r_H * t_H}_{\text{Hit}} + \underbrace{r_M * t_{Mem}}_{\text{Miss}}$$

Mehrstufige Cache-Hierarchie: Konkretisierung des Miss-Zeig

$$t_a = \underbrace{r_{H1} * t_{L1}}_{\text{Hit L1}} + r_{M1} * \underbrace{\left(\underbrace{r_{H2} * t_{L2}}_{\text{Hit L2}} + \underbrace{r_{M2} * t_{Mem}}_{\text{Miss L2}} \right)}_{\text{Miss L1}}$$

Aufgabe 1.1: Cacheleistung

Bei dem Entwurf eines Systems stehen zwei Entwurfsalternativen zur Auswahl. In beiden Entwurfsalternativen kommt eine zwei-stufige Cache-Hierarchie zum Einsatz. Entwurfsalternative A hat einen kleinen L1-Cache mit einer Zugriffszeit von $t_{A-L1} = 10 \text{ ns}$, sowie einen L2-Cache mit einer Zugriffszeit von $t_{A-L2} = 30 \text{ ns}$. In Entwurfsalternative B kommt ein größerer L1-Cache mit einer Zugriffszeit von $t_{B-L1} = 12 \text{ ns}$ zum Einsatz, sowie ein L2-Cache mit einer Zugriffszeit von $t_{B-L2} = 25 \text{ ns}$. Die Zugriffszeit des Hauptspeichers sei in beiden Entwurfsalternativen gleich und betrage $t_{Mem} = 100 \text{ ns}$.

- a) Geben Sie eine Formel zur Berechnung t_a der mittleren Zugriffszeit in einer zwei-stufigen Cache-Hierarchie an.

Antwort:
$$t_a = \underbrace{r_{H1} * t_{L1}}_{\text{Hit L1}} + \underbrace{r_{M1} * \left(\underbrace{r_{H2} * t_{L2}}_{\text{Hit L2}} + \underbrace{r_{M2} * t_{Mem}}_{\text{Miss L2}} \right)}_{\text{Miss L1}}$$

Aufgabe 1.1: Cacheleistung

Bei dem Entwurf eines Systems stehen zwei Entwurfsalternativen zur Auswahl. In beiden Entwurfsalternativen kommt eine zwei-stufige Cache-Hierarchie zum Einsatz. Entwurfsalternative A hat einen kleinen L1-Cache mit einer Zugriffszeit von $t_{A-L1} = 10 \text{ ns}$, sowie einen L2-Cache mit einer Zugriffszeit von $t_{A-L2} = 30 \text{ ns}$. In Entwurfsalternative B kommt ein größerer L1-Cache mit einer Zugriffszeit von $t_{B-L1} = 12 \text{ ns}$ zum Einsatz, sowie ein L2-Cache mit einer Zugriffszeit von $t_{B-L2} = 25 \text{ ns}$. Die Zugriffszeit des Hauptspeichers sei in beiden Entwurfsalternativen gleich und betrage $t_{Mem} = 100 \text{ ns}$.

- a) Geben Sie eine Formel zur Berechnung t_a der mittleren Zugriffszeit in einer zwei-stufigen Cache-Hierarchie an.

Antwort:
$$t_a = \underbrace{r_{H1} * t_{L1}}_{\text{Hit L1}} + \underbrace{r_{M1} * \left(\underbrace{r_{H2} * t_{L2}}_{\text{Hit L2}} + \underbrace{r_{M2} * t_{Mem}}_{\text{Miss L2}} \right)}_{\text{Miss L1}}$$

Aufgabe 1.1: Cacheleistung

Bei dem Entwurf eines Systems stehen zwei Entwurfsalternativen zur Auswahl. In beiden Entwurfsalternativen kommt eine zwei-stufige Cache-Hierarchie zum Einsatz. Entwurfsalternative A hat einen kleinen L1-Cache mit einer Zugriffszeit von $t_{A-L1} = 10 \text{ ns}$, sowie einen L2-Cache mit einer Zugriffszeit von $t_{A-L2} = 30 \text{ ns}$. In Entwurfsalternative B kommt ein größerer L1-Cache mit einer Zugriffszeit von $t_{B-L1} = 12 \text{ ns}$ zum Einsatz, sowie ein L2-Cache mit einer Zugriffszeit von $t_{B-L2} = 25 \text{ ns}$. Die Zugriffszeit des Hauptspeichers sei in beiden Entwurfsalternativen gleich und betrage $t_{Mem} = 100 \text{ ns}$.

- a) Geben Sie eine Formel zur Berechnung t_a der mittleren Zugriffszeit in einer zwei-stufigen Cache-Hierarchie an.

Antwort:
$$t_a = \underbrace{r_{H1} * t_{L1}}_{\text{Hit L1}} + \underbrace{r_{M1} * \left(\underbrace{r_{H2} * t_{L2}}_{\text{Hit L2}} + \underbrace{r_{M2} * t_{Mem}}_{\text{Miss L2}} \right)}_{\text{Miss L1}}$$

Aufgabe 1.1: Cacheleistung

$t_{A-L1} = 10 \text{ ns}$, $t_{A-L2} = 30 \text{ ns}$, $t_{B-L1} = 12 \text{ ns}$, $t_{B-L2} = 25 \text{ ns}$, $t_{Mem} = 100 \text{ ns}$.

Bei der Evaluation wurden folgende Hit-Raten gemessen:

- Alternative A: $r_{A-L1} = 70 \%$, sowie $r_{A-L2} = 40 \%$
- Alternative B: $r_{B-L1} = 75 \%$, sowie $r_{B-L2} = 35 \%$

Alternative A

$$t_a = 70\% * 10 \text{ ns} + (1 - 70\%) * (40\% * 30 \text{ ns} + (1 - 40\%) * 100 \text{ ns})$$
$$t_a = 7 \text{ ns} + 0,3 * (12 \text{ ns} + 60 \text{ ns}) = 28,6 \text{ ns}$$

Alternative B

$$t_a = 75\% * 12 \text{ ns} + (1 - 75\%) * (35\% * 25 \text{ ns} + (1 - 35\%) * 100 \text{ ns})$$
$$t_a = 9 \text{ ns} + 0,25 * (8,75 \text{ ns} + 65 \text{ ns}) = 27,4375 \text{ ns}$$

Die durchschnittliche Zugriffszeit in **Entwurfsalternative B** ist geringer, somit ist diese Entwurfsalternative zu wählen.

Aufgabe 1.1: Cacheleistung

$t_{A-L1} = 10 \text{ ns}$, $t_{A-L2} = 30 \text{ ns}$, $t_{B-L1} = 12 \text{ ns}$, $t_{B-L2} = 25 \text{ ns}$, $t_{Mem} = 100 \text{ ns}$.

Bei der Evaluation wurden folgende Hit-Raten gemessen:

- Alternative A: $r_{A-L1} = 70 \%$, sowie $r_{A-L2} = 40 \%$
- Alternative B: $r_{B-L1} = 75 \%$, sowie $r_{B-L2} = 35 \%$

Alternative A

$$t_a = 70\% * 10 \text{ ns} + (1 - 70\%) * (40\% * 30 \text{ ns} + (1 - 40\%) * 100 \text{ ns})$$
$$t_a = 7 \text{ ns} + 0,3 * (12 \text{ ns} + 60 \text{ ns}) = 28,6 \text{ ns}$$

Alternative B

$$t_a = 75\% * 12 \text{ ns} + (1 - 75\%) * (35\% * 25 \text{ ns} + (1 - 35\%) * 100 \text{ ns})$$
$$t_a = 9 \text{ ns} + 0,25 * (8,75 \text{ ns} + 65 \text{ ns}) = 27,4375 \text{ ns}$$

Die durchschnittliche Zugriffszeit in **Entwurfsalternative B** ist geringer, somit ist diese Entwurfsalternative zu wählen.

Aufgabe 1.1: Cacheleistung

$t_{A-L1} = 10 \text{ ns}$, $t_{A-L2} = 30 \text{ ns}$, $t_{B-L1} = 12 \text{ ns}$, $t_{B-L2} = 25 \text{ ns}$, $t_{Mem} = 100 \text{ ns}$.

Bei der Evaluation wurden folgende Hit-Raten gemessen:

- Alternative A: $r_{A-L1} = 70 \%$, sowie $r_{A-L2} = 40 \%$
- Alternative B: $r_{B-L1} = 75 \%$, sowie $r_{B-L2} = 35 \%$

Alternative A

$$t_a = 70\% * 10 \text{ ns} + (1 - 70\%) * (40\% * 30 \text{ ns} + (1 - 40\%) * 100 \text{ ns})$$
$$t_a = 7 \text{ ns} + 0,3 * (12 \text{ ns} + 60 \text{ ns}) = 28,6 \text{ ns}$$

Alternative B

$$t_a = 75\% * 12 \text{ ns} + (1 - 75\%) * (35\% * 25 \text{ ns} + (1 - 35\%) * 100 \text{ ns})$$
$$t_a = 9 \text{ ns} + 0,25 * (8,75 \text{ ns} + 65 \text{ ns}) = 27,4375 \text{ ns}$$

Die durchschnittliche Zugriffszeit in **Entwurfsalternative B** ist geringer, somit ist diese Entwurfsalternative zu wählen.

Bei dem Entwurf eines Systems stehen zwei Entwurfsalternativen zur Auswahl. In beiden Entwurfsalternativen kommt eine zwei-stufige Cache-Hierarchie zum Einsatz.

Entwurfsalternative A hat einen größeren L1-Cache mit einer Zugriffszeit von $t_{A-L1} = 2,5 \text{ ns}$, Entwurfsalternative B besitzt einen kleineren L1-Cache mit einer Zugriffszeit von $t_{B-L1} = 2 \text{ ns}$. Beide Entwurfsalternativen besitzen einen L2-Cache mit $t_{L2} = 10 \text{ ns}$. Die Zugriffszeit des Hauptspeichers sei in beiden Entwurfsalternativen gleich und betrage $t_{Mem} = 100 \text{ ns}$.

Um die Leistung der Cache-Hierarchie zu steigern, werden bei Entwurfsalternative B alle Hierarchieebenen parallel angefragt.

Bei Entwurfsalternative A findet ein Zugriff auf die nächste Hierarchieebene erst statt, wenn die vorherige Ebene das angeforderte Datum nicht gespeichert hat.

a) Bei der Evaluation beider Entwurfsalternativen wurden folgende Hit-Raten gemessen:

- Alternative A: $r_{A-L1} = 80\%$, sowie $r_{A-L2} = 90\%$
- Alternative B: $r_{B-L1} = 70\%$, sowie $r_{B-L2} = 90\%$

Für welche Entwurfsalternative würden Sie sich aus Gründen der Leistung entscheiden? Begründen Sie ihre Antwort.

$$t_{A-L1} = 2,5 \text{ ns}, t_{B-L1} = 2 \text{ ns}, t_{L2} = 10 \text{ ns}, t_{Mem} = 100 \text{ ns}.$$
$$r_{A-L1} = 80 \%, r_{A-L2} = 90 \%, r_{B-L1} = 70 \%, r_{B-L2} = 90 \%$$

Alternative A - Sequentielle Anfrage

$$t_A = 0,8 * 2,5 \text{ ns} + 0,2 * (0,9 * 12,5 \text{ ns} + 0,1 * 112,5 \text{ ns})$$
$$t_A = 2 \text{ ns} + 0,2 * (11,25 \text{ ns} + 11,25 \text{ ns}) = 2 \text{ ns} + 0,2 * 22,5 \text{ ns}$$
$$t_A = 2 \text{ ns} + 4,5 \text{ ns} = 6,5 \text{ ns}$$

Alternative B - Parallele Anfrage

$$t_B = 0,7 * 2 \text{ ns} + 0,3 * (0,9 * 10 \text{ ns} + 0,1 * 100 \text{ ns})$$
$$t_B = 1,4 \text{ ns} + 0,3 * (9 \text{ ns} + 10 \text{ ns}) = 1,4 \text{ ns} + 0,3 * 19 \text{ ns}$$
$$t_B = 1,4 \text{ ns} + 5,7 \text{ ns} = 7,1 \text{ ns}$$

$t_A < t_B$, Entwurfsalternative A ist schneller und daher zu wählen.

$$t_{A-L1} = 2,5 \text{ ns}, t_{B-L1} = 2 \text{ ns}, t_{L2} = 10 \text{ ns}, t_{Mem} = 100 \text{ ns}.$$
$$r_{A-L1} = 80 \%, r_{A-L2} = 90 \%, r_{B-L1} = 70 \%, r_{B-L2} = 90 \%$$

Alternative A - Sequentielle Anfrage

$$t_A = 0,8 * 2,5 \text{ ns} + 0,2 * (0,9 * 12,5 \text{ ns} + 0,1 * 112,5 \text{ ns})$$
$$t_A = 2 \text{ ns} + 0,2 * (11,25 \text{ ns} + 11,25 \text{ ns}) = 2 \text{ ns} + 0,2 * 22,5 \text{ ns}$$
$$t_A = 2 \text{ ns} + 4,5 \text{ ns} = 6,5 \text{ ns}$$

Alternative B - Parallele Anfrage

$$t_B = 0,7 * 2 \text{ ns} + 0,3 * (0,9 * 10 \text{ ns} + 0,1 * 100 \text{ ns})$$
$$t_B = 1,4 \text{ ns} + 0,3 * (9 \text{ ns} + 10 \text{ ns}) = 1,4 \text{ ns} + 0,3 * 19 \text{ ns}$$
$$t_B = 1,4 \text{ ns} + 5,7 \text{ ns} = 7,1 \text{ ns}$$

$t_A < t_B$, Entwurfsalternative A ist schneller und daher zu wählen.

$$t_{A-L1} = 2,5 \text{ ns}, t_{B-L1} = 2 \text{ ns}, t_{L2} = 10 \text{ ns}, t_{Mem} = 100 \text{ ns}.$$
$$r_{A-L1} = 80 \%, r_{A-L2} = 90 \%, r_{B-L1} = 70 \%, r_{B-L2} = 90 \%$$

Alternative A - Sequentielle Anfrage

$$t_A = 0,8 * 2,5 \text{ ns} + 0,2 * (0,9 * 12,5 \text{ ns} + 0,1 * 112,5 \text{ ns})$$
$$t_A = 2 \text{ ns} + 0,2 * (11,25 \text{ ns} + 11,25 \text{ ns}) = 2 \text{ ns} + 0,2 * 22,5 \text{ ns}$$
$$t_A = 2 \text{ ns} + 4,5 \text{ ns} = 6,5 \text{ ns}$$

Alternative B - Parallele Anfrage

$$t_B = 0,7 * 2 \text{ ns} + 0,3 * (0,9 * 10 \text{ ns} + 0,1 * 100 \text{ ns})$$
$$t_B = 1,4 \text{ ns} + 0,3 * (9 \text{ ns} + 10 \text{ ns}) = 1,4 \text{ ns} + 0,3 * 19 \text{ ns}$$
$$t_B = 1,4 \text{ ns} + 5,7 \text{ ns} = 7,1 \text{ ns}$$

$t_A < t_B$, Entwurfsalternative A ist schneller und daher zu wählen.

$$t_{A-L1} = 2,5 \text{ ns}, t_{B-L1} = 2 \text{ ns}, t_{L2} = 10 \text{ ns}, t_{Mem} = 100 \text{ ns}.$$
$$r_{A-L1} = 80 \%, r_{A-L2} = 90 \%, r_{B-L1} = 70 \%, r_{B-L2} = 90 \%$$

Alternative A - Sequentielle Anfrage

$$t_A = 0,8 * 2,5 \text{ ns} + 0,2 * (0,9 * 12,5 \text{ ns} + 0,1 * 112,5 \text{ ns})$$

$$t_A = 2 \text{ ns} + 0,2 * (11,25 \text{ ns} + 11,25 \text{ ns}) = 2 \text{ ns} + 0,2 * 22,5 \text{ ns}$$

$$t_A = 2 \text{ ns} + 4,5 \text{ ns} = 6,5 \text{ ns}$$

Alternative B - Parallele Anfrage

$$t_B = 0,7 * 2 \text{ ns} + 0,3 * (0,9 * 10 \text{ ns} + 0,1 * 100 \text{ ns})$$

$$t_B = 1,4 \text{ ns} + 0,3 * (9 \text{ ns} + 10 \text{ ns}) = 1,4 \text{ ns} + 0,3 * 19 \text{ ns}$$

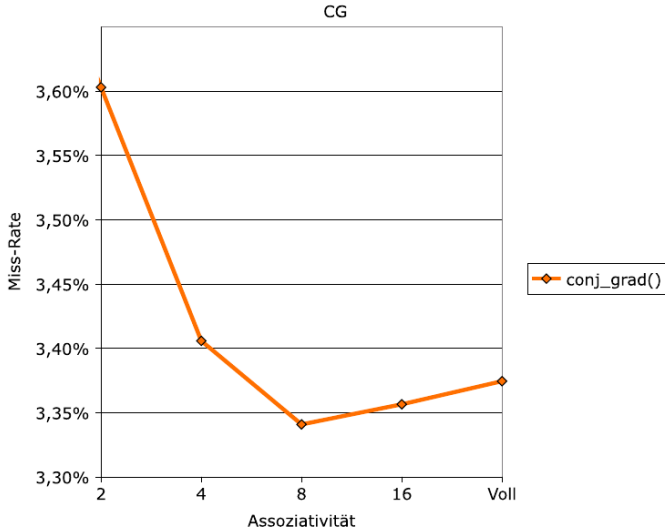
$$t_B = 1,4 \text{ ns} + 5,7 \text{ ns} = 7,1 \text{ ns}$$

$t_A < t_B$, Entwurfsalternative A ist schneller und daher zu wählen.

Beweisen oder widerlegen Sie folgende Behauptungen:
Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used (LRU)-Strategie verdrängen.

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.
- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Miss-Rate.

Aufgabe 1.2: Beweise - Motivation



Aufgabe 1.2: Beweise

Beweisen oder widerlegen Sie folgende Behauptungen:

Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used (LRU)-Strategie verdrängen.

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.
- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Miss-Rate.



Widerlegung:

Analog zu **Belady's Anomaly** [1]!

Aufgabe 1.2: Beweise (forts.)

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

4-fach assoziativ

Folge:

Voraussetzung: Speicherzugriffe A, B, E, F werden auf den ersten Satz abgebildet
C und D auf den zweiten Satz des 2-fach assoziativen Cache

Aufgabe 1.2: Beweise (forts.)

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	A
B	B

C
D

4-fach assoziativ

A	A
B	B
C	
D	

Folge: *ABCDAB*

Voraussetzung: Speicherzugriffe A, B, E, F werden auf den ersten Satz abgebildet

C und D auf den zweiten Satz des 2-fach assoziativen Cache

Aufgabe 1.2: Beweise (forts.)

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	A	E
B	B	F

C	
D	

4-fach assoziativ

A	A
B	B
C	E
D	F

Folge: *ABCDABEF*

Voraussetzung: Speicherzugriffe A, B, E, F werden auf den ersten Satz abgebildet

C und D auf den zweiten Satz des 2-fach assoziativen Cache

Aufgabe 1.2: Beweise (forts.)

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	A	E
B	B	F

C	C	
D	D	

4-fach assoziativ

A	A	C
B	B	D
C	E	
D	F	

Folge: *ABCDABEFCD*

Voraussetzung: Speicherzugriffe A, B, E, F werden auf den ersten Satz abgebildet

C und D auf den zweiten Satz des 2-fach assoziativen Cache

Aufgabe 1.2: Beweise (forts.)

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	A	E	A	E
B	B	F	B	F

C	C			
D	D			

4-fach assoziativ

A	A	C	E
B	B	D	F
C	E	A	
D	F	B	

Folge: *ABCDABEFCDABEF*

Voraussetzung: Speicherzugriffe A, B, E, F werden auf den ersten Satz abgebildet

C und D auf den zweiten Satz des 2-fach assoziativen Cache

Aufgabe 1.2: Beweise (forts.)

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

2-fach assoziativ

A	A	E	A	E
B	B	F	B	F

C	C	C		
D	D	D		

4-fach assoziativ

A	A	C	E
B	B	D	F
C	E	A	C
D	F	B	D

Folge: $ABCD(ABEFCD)^n$

Voraussetzung: Speicherzugriffe A, B, E, F werden auf den ersten Satz abgebildet

C und D auf den zweiten Satz des 2-fach assoziativen Cache

Aufgabe 1.2: Beweise (forts.)

Beweisen oder widerlegen Sie folgende Behauptungen:

Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used (LRU)-Strategie verdrängen.

- a) Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.
- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Miss-Rate.

Aufgabe 1.2: Beweise (forts.)

- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

Vollassoziativ

Folge:

Voraussetzung: A wird auf die erste, B auf die zweite, C auf die dritte und D und H auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 1.2: Beweise (forts.)

- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A
B
C
D

Vollassoziativ

A
B
C
D

Folge: *ABCD*

Voraussetzung: A wird auf die erste, B auf die zweite, C auf die dritte und D und H auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 1.2: Beweise (forts.)

- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A
B
C
D H

Vollassoziativ

A H
B
C
D

Folge: *ABCDH*

Voraussetzung: A wird auf die erste, B auf die zweite, C auf die dritte und D und H auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 1.2: Beweise (forts.)

- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A	A
B	B
C	C
D	H

Vollassoziativ

A	H
B	A
C	B
D	C

Folge: *ABCDHABC*

Voraussetzung: A wird auf die erste, B auf die zweite, C auf die dritte und D und H auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 1.2: Beweise (forts.)

- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A	A
B	B
C	C
D	H D H

Vollassoziativ

A	H	D
B	A	H
C	B	
D	C	

Folge: $(ABCDH)^n$

Voraussetzung: A wird auf die erste, B auf die zweite, C auf die dritte und D und H auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 1.2: Beweise (forts.)

- b) Vollassoziative Caches haben satzassoziativen Caches gegenüber immer eine niedrigere Missrate.

Direct Mapped

A	A	A	
B	B	B	
C	C	C	
D	H	D	H

Vollassoziativ

A	H	D	C
B	A	H	
C	B	A	
D	C	B	

Folge: $(ABCDH)^n$

Voraussetzung: A wird auf die erste, B auf die zweite, C auf die dritte und D und H auf die vierte Cachezeile des DM-Cache abgebildet.

Aufgabe 1.3 - Verständnisfragen

- a) Welche Eigenschaft von Anwendungen werden von Caches ausgenutzt?

Antworten:

- a) Räumliche und Zeitliche Lokalität

Aufgabe 1.3 - Verständnisfragen

- a) Welche Eigenschaft von Anwendungen werden von Caches ausgenutzt?

Antworten:

- a) Räumliche und Zeitliche Lokalität

- b) Welche Arten von Cache-Misses können unterschieden werden?

Antworten:

- b) Conflict-, Capacity-, Compulsory-Misses

Im Mehrprozessorfall:

- Coherency-Miss
- Unterscheidung in: False-Sharing-Miss bzw. True-Sharing-Miss

- b) Welche Arten von Cache-Misses können unterschieden werden?

Antworten:

- b) Conflict-, Capacity-, Compulsory-Misses

Im Mehrprozessorfall:

- Coherency-Miss
- Unterscheidung in: False-Sharing-Miss bzw. True-Sharing-Miss

- c) Warum ist der Aufbau des Hauptspeichers aus SRAM-Zellen nicht sinnvoll?

Antworten:

- c) SRAM-Zellen benötigen viel Chipfläche, deswegen ist der Aufbau des Hauptspeichers aus SRAM-Zellen entweder zu teuer oder man könnte nur geringe Kapazitäten anbieten.

- c) Warum ist der Aufbau des Hauptspeichers aus SRAM-Zellen nicht sinnvoll?

Antworten:

- c) SRAM-Zellen benötigen viel Chipfläche, deswegen ist der Aufbau des Hauptspeichers aus SRAM-Zellen entweder zu teuer oder man könnte nur geringe Kapazitäten anbieten.

Problem bei Verwendung von Caches in MP-Systemen

- Wahrung der Konsistenz
- CPUs operieren auf lokalen Kopien
- Daten in den Caches können sich von den Daten im Hauptspeicher unterscheiden

Bus-Snooping-basiert

- Write-Invalidate Protokoll
 - arbeitet auf Cacheline-Ebene
 - benötigt nur ein Invalidate
 - Vertreter
 - MSI [2]
 - MESI [2]
 - MOESI [3]
- Write-Update Protokoll
 - arbeitet auf Wort-Ebene
 - u. U. mehrere Update-Broadcasts nötig
 - z.B. Dragon [2]

Verzeichnisbasiert

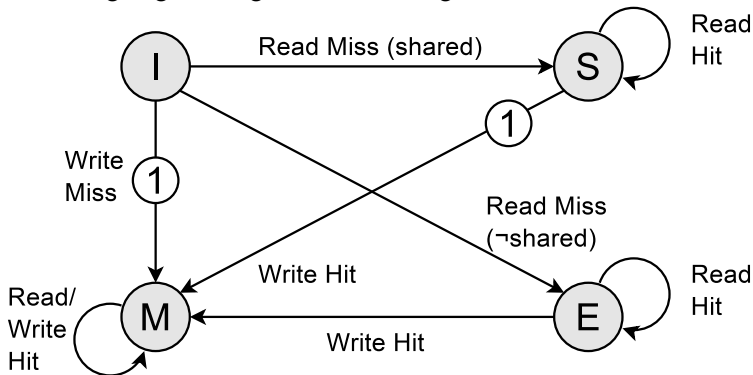
Wird in DSM-Systemen benötigt. (siehe [2], Kapitel 8)

- Beobachtung des Speicherbusses hinsichtlich Speicherzugriffe anderer Bus-Master
- Bus-Snooping erfordert einen **globalen Speicherbus**
- Jeder Cache muss um sog. **Snoop-Logik** und **Steuersignale** zu anderen Caches erweitert werden.
 - Invalidate-Signal
 - Shared-Signal
 - Retry-Signal

- Write-Invalidate Protokoll
- MESI: Zustandsautomat mit 4 Zuständen
 - M: exclusive Modified
 - E: Exclusive unmodified
 - S: Shared unmodified
 - I: Invalid
- Jede Cachezeile befindet sich in einem dieser Zustände
- d.h. jede Cachezeile wird um **2 Zustandsbits** erweitert
- Zustandsübergänge werden durch lokale und entfernte Speicherzugriffe ausgelöst

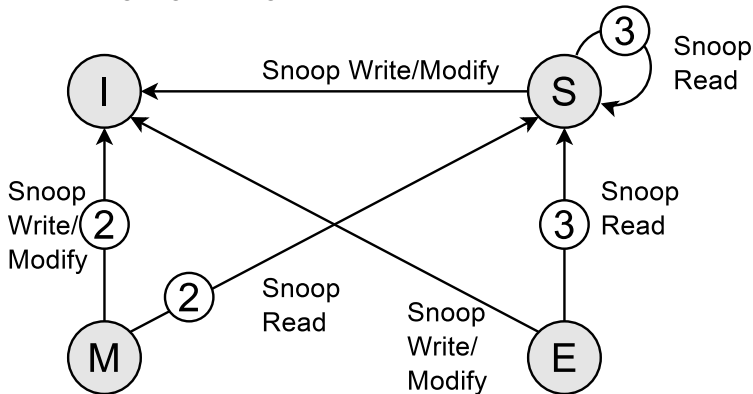
MESI-Zustandsautomat (1)

Zustandübergänge, ausgelöst durch eigene Aktionen:



MESI-Zustandsautomat (2)

Zustandsübergänge, ausgelöst durch andere Prozessoren:



- 2 Retry-Signal wird aktiviert, Cache-Zeile in den Hauptspeicher kopiert, dann Signal deaktiviert
- 3 Shared-Signal wird aktiviert

Aufgabe 2.1: MESI Cachekohärenz-Protokoll

Ein Dreiprozessorsystem sei speichergekoppelt. Die Caches haben je eine Größe von zwei Cachezeilen, welche je genau ein Speicherwort aufnehmen können. Die Füllung des Caches erfolgt von der niedrigsten Cachezeile aufwärts, sofern noch freie Zeilen zur Verfügung stehen, anderenfalls wird gemäß LRU-Strategie verdrängt. Als Cache-Kohärenzprotokoll komme das MESI-Protokoll zum Einsatz.

- Vervollständigen Sie die gegebene Tabelle: Geben Sie jeweils Inhalt der Cache-Zeile und MESI-Zustand an.

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6						
2	rd 2						
1	rd 4						
3	rd 4						
2	rd 3						
3	wr 7						
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4						
2	rd 3						
3	wr 7						
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3						
3	wr 7						
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4						
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7						
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5						
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5					5/M	
1	rd 3						
3	wr 3						
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5					5/M	
1	rd 3	3/S			3/S		
3	wr 3	3/I			3/I		3/M
2	wr 7						

Aufgabe 2.1: MESI-Protokoll (forts.)

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/S			7/S
3	wr 5					5/M	
1	rd 3	3/S			3/S		
3	wr 3	3/I			3/I		3/M
2	wr 7			7/M			

- Erweiterung des MESI-Protokolls um einen weiteren Zustand
- Neuer Zustand O: Owned (shared modified)

Vorteil

- Durch Cache-Cache-Transfers werden 2 Hauptspeicherzugriffe eingespart

Nachteile

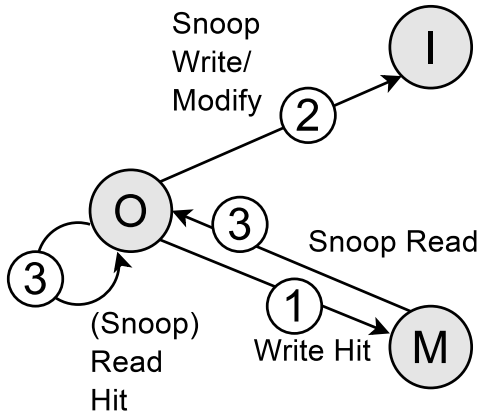
- Komplexere Logik notwendig
- 3 Zustands-Bits nötig

Wird ein in einem Cache bereits modifiziertes Datum von einem weiteren Cache gelesen, so wechselt der Cache mit dem modifizierten Datum von Zustand **M** in den Zustand **O**. Der lesende Cache übernimmt das Datum aus dem Cache des ersten Prozessors und lagert das Datum, gemäß MESI-Protokoll, als Shared **S** markiert ein. Werden die Daten in Caches mit dem Zustand **S** modifiziert, so wird das Datum in den anderen Caches invalidiert (Zustand **I**) und in den Zustand **M** gewechselt.

- a) Erweitern Sie den aus der Vorlesung bekannte MESI-Zustandsautomat um die oben beschriebene Erweiterungen zum MOESI-Zustandsautomat.

MOESI Zustandsautomat (nach [3])

Zusätzliche Kanten für O-Zustand:



- 1 Invalidate
- 2 Retry
- 3 Shared

Aufgabe 2.2: MOESI-Protokoll

Ein Dreiprozessorsystem sei speichergekoppelt. Die Caches haben je eine Größe von zwei Cachezeilen, welche je genau ein Speicherwort aufnehmen können. Die Füllung des Caches erfolgt von der niedrigsten Cachezeile aufwärts, sofern noch freie Zeilen zur Verfügung stehen, andernfalls wird gemäß LRU-Strategie verdrängt. Als Cache-Kohärenzprotokoll komme das MOESI-Protokoll zum Einsatz. Der Cache sei initial leer.

- b) Vervollständigen sie die gegebene Tabelle: Geben Sie jeweils Inhalt der Cache-Zeile und MOESI-Zustand an.

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4						
3	rd 4						
2	rd 3						
2	wr 5						
1	rd 2						
2	wr 4						
2	rd 1						
3	rd 4						
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3						
2	wr 5						
1	rd 2						
2	wr 4						
2	rd 1						
3	rd 4						
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2						
2	wr 4						
2	rd 1						
3	rd 4						
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4						
2	rd 1						
3	rd 4						
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4	4/I		4/M		4/I	
2	rd 1						
3	rd 4						
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4	4/I		4/M		4/I	
2	rd 1				1/E		
3	rd 4						
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4	4/I		4/M		4/I	
2	rd 1				1/E		
3	rd 4			4/O		4/S	
3	rd 3						
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4	4/I		4/M		4/I	
2	rd 1				1/E		
3	rd 4			4/O		4/S	
3	rd 3						3/E
1	wr 1						
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4	4/I		4/M		4/I	
2	rd 1				1/E		
3	rd 4			4/O		4/S	
3	rd 3						3/E
1	wr 1	1/M			1/I		
3	rd 1						

Aufgabe 2.2: MOESI-Protokoll

Proz.	Aktion	Prozessor 1		Prozessor 2		Prozessor 3	
		Line 1	Line 2	Line 1	Line 2	Line 1	Line 2
	init	-	-	-	-	-	-
1	rd 4	4/E					
3	rd 4	4/S				4/S	
2	rd 3			3/E			
2	wr 5				5/M		
1	rd 2		2/E				
2	wr 4	4/I		4/M		4/I	
2	rd 1				1/E		
3	rd 4			4/O		4/S	
3	rd 3						3/E
1	wr 1	1/M			1/I		
3	rd 1	1/O				1/S	

Aufgabe 2.2 - Teilaufgabe c

Wie viele Hauptspeicherzugriffe werden durch diese Speicherzugriffsfolge verursacht? Führt hier die Verwendung des MOESI gegenüber des MESI-Protokolls zu einer Leistungssteigerung und wenn ja, warum?

Antwort

MOESI: Lesend: 6 Zugriffe – Schreibend: 1 Zugriff

MESI: Lesend: 8 Zugriffe – Schreibend: 3 Zugriffe

Es würden zwei Lesezugriffe und zwei Schreibzugriffe eingespart → Leistungssteigerung vorhanden

Aufgabe 2.2 - Teilaufgabe c

Wie viele Hauptspeicherzugriffe werden durch diese Speicherzugriffsfolge verursacht? Führt hier die Verwendung des MOESI gegenüber des MESI-Protokolls zu einer Leistungssteigerung und wenn ja, warum?

Antwort

MOESI: Lesend: 6 Zugriffe – Schreibend: 1 Zugriff

MESI: Lesend: 8 Zugriffe – Schreibend: 3 Zugriffe

Es würden zwei Lesezugriffe und zwei Schreibzugriffe eingespart → Leistungssteigerung vorhanden

Aufgabe 2.3: Verständnisfragen

- a) Warum existiert im MOESI-Protokoll kein Zustandsübergang vom Zustand **S** nach Zustand **O**?

Antwort: Ein Zustandsübergang von **S** nach **O** kann es in einem Write-Invalidate-Protokoll nicht geben. Gemäß einem Write-Invalidate-Protokoll werden bei Veränderung eines geteilten Datums, die Daten in den entfernten Caches invalidiert und demzufolge dem Datum der Zustand **M** zugewiesen.

Der Wechsel von Zustand **S** in den Zustand **O**, müsste eine Aktualisierung des Datum in den entfernten Caches nach sich ziehen. Diese Vorgehensweise entspricht einem Write-Update-Protokoll

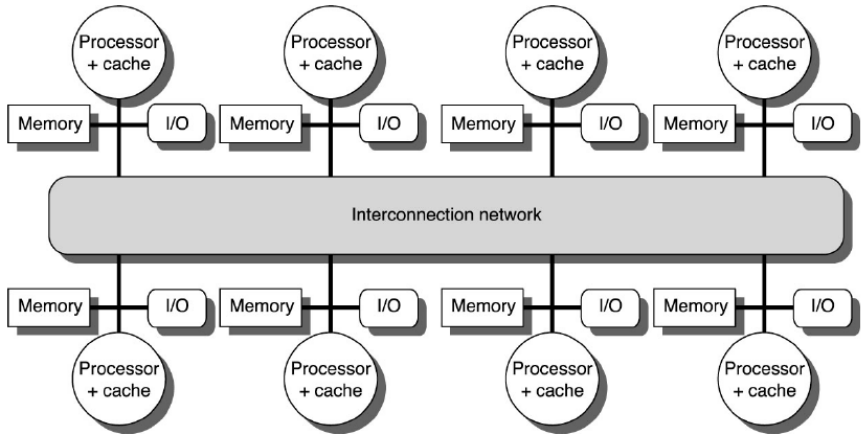
Aufgabe 2.3: Verständnisfragen

- a) Warum existiert im MOESI-Protokoll kein Zustandsübergang vom Zustand **S** nach Zustand **O**?

Antwort: Ein Zustandsübergang von **S** nach **O** kann es in einem Write-Invalidate-Protokoll nicht geben. Gemäß einem Write-Invalidate-Protokoll werden bei Veränderung eines geteilten Datums, die Daten in den entfernten Caches invalidiert und demzufolge dem Datum der Zustand **M** zugewiesen.

Der Wechsel von Zustand **S** in den Zustand **O**, müsste eine Aktualisierung des Datum in den entfernten Caches nach sich ziehen. Diese Vorgehensweise entspricht einem Write-Update-Protokoll

Distributed-Shared-Memory (DSM)-System

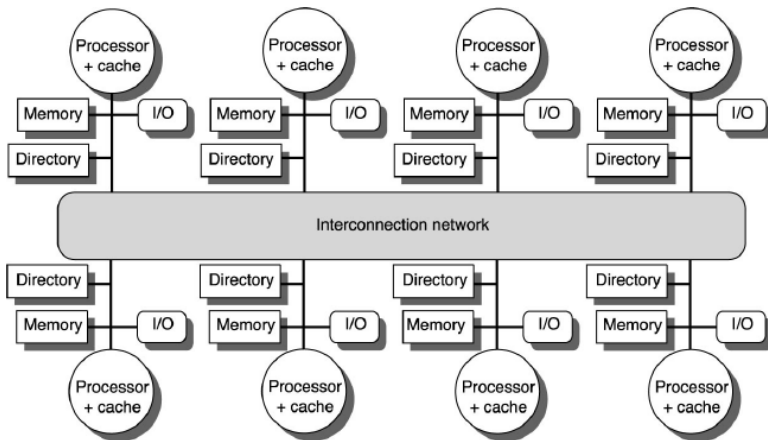


© 2003 Elsevier Science (USA). All rights reserved.

Einfachste Methode zur Wahrung der Kohärenz in DSM-Systemen:

- Nur Speicherzugriffe auf den lokalen Speicher werden in den Cache geladen

Verzeichnisbasierte Cache-Kohärenzprotokolle



© 2003 Elsevier Science (USA). All rights reserved.

- DSM-Systeme haben kein gemeinsamer Speicherbus, den eine Snooping-Logik überwachen könnte
- Herstellen der Cache-Kohärenz über Verzeichnistabellen
- Implementierung in Hard- oder Software möglich
- Die Tabelle protokolliert für jeden Blockrahmen, ob dieser in den lokalen oder einem entfernten Cache-Speicher als Cache-Block übertragen worden ist.
- Zustände werden analog zu den MESI-Zuständen definiert

Aufgabe 2.3: Verständnisfragen

- b) Warum läßt sich MESI nicht in Distributed Shared Memory (DSM) Systemen einsetzen?

Antwort:

- b) In DSM-Systemen existiert kein gemeinsamer Speicherbus, den eine Snooping-Logik überwachen könnte.

Aufgabe 2.3: Verständnisfragen

- b) Warum läßt sich MESI nicht in Distributed Shared Memory (DSM) Systemen einsetzen?

Antwort:

- b) In DSM-Systemen existiert kein gemeinsamer Speicherbus, den eine Snooping-Logik überwachen könnte.

Aufgabe 2.3: Verständnisfragen

- c) Welche Protokolle stellen in DSM-Systemen die Cache-Kohärenz sicher?

Antwort:

- c) In DSM-Systemen kommen verzeichnisbasierte Cache-Kohärenzprotokolle zum Einsatz.

Aufgabe 2.3: Verständnisfragen

- c) Welche Protokolle stellen in DSM-Systemen die Cache-Kohärenz sicher?

Antwort:

- c) In DSM-Systemen kommen verzeichnisbasierte Cache-Kohärenzprotokolle zum Einsatz.

Mul: 4 Takte Ausführungszeit

Div: 8 Takte Ausführungszeit

Übung: Takt "Execute" + Ausführungszeit:

#	Instruktion	Issue	Execute	Write Result
1	mul r2, r1, r1	1	2	7
2	div r4, r4, r2	2	8	17

Richtig: Ausführung beginnt mit erstem Takt "Execute":

#	Instruktion	Issue	Execute	Write Result
1	mul r2, r1, r1	1	2	6
2	div r4, r4, r2	2	7	15

In der Klausur eindeutig aus der Aufgabe heraus ersichtlich.

Klausur

- Anmeldung zur Klausur via Studierendenportal
- **Anmelde- und Abmeldeschluss** ist der 31.07.2013
- Danach besteht kein Anrecht mehr auf Klausurteilnahme!
- **Klausurtermin:** 07.08.2013 um 14 Uhr
- Hörsäle: HSaF, Benz-HS, Daimler-HS
- Sitzplatzverteilung wird online bekanntgegeben
- **Es sind keine Hilfsmittel zugelassen!**

- 1 Belady et al.: An anomaly in space-time characteristics of certain programs running in a paging machine, Communications of the ACM, Volume 12, Issue 6, Pages: 349 - 353, 1969
- 2 David E. Culler, Jaswinder Pal Singh: Parallel Computer Architecture - A Hardware/Software Approach Morgan Kaufmann, 1999, ISBN 1-55860-343-3
- 3 AMD64 Architecture Programmer's Manual Vol 2 'System Programming'
http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf
- 4 Brinkschulte, Ungerer: Mikrocontroller und Mikroprozessoren, Springer Verlag

7. Zentralübung Rechnerstrukturen im SS 2013

Cache-Kohärenz

Mario Kicherer, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

15. Juli 2013

