Zentralübung Rechnerstrukturen im SS2007

Cache-Kohärenz und -Konsistenz (korr. Version)

David Kramer

kramer@ira.uka.de

Universität Karlsruhe (TH) - Forschungsuniversität
Institute für Technische Informatik (ITEC)
Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

1. August 2007

Organisatorisches

Klausur

- Anmeldung zur Klausur ist nun möglich
- Anmeldeschluss ist der 20.7.2007 um 12 Uhr
- Anmeldeort: orangen Anmeldekasten vor dem Schwarzen Brett in ersten Stock, Gebäude 20.20
- Klausurtermin: 8.8.2007 um 14 Uhr

Cache - Überblick

- Programmierer wollen unendlich viel, sehr sehr schnellen Speicher
- Größe ist kein Problem $\rightarrow$ Virtueller Speicher
- Zugriffszeit ist ein Problem $\rightarrow$ Memory Wall
- $\rightarrow$ Einführung von schnellen, aber kleineren Zwischenspeicher
- 90/10 Regel
- Ausnutzung von
  - Zeitlicher Lokalität
  - Räumlicher Lokalität

Cache-Parameter (1)

- Cache Hierachie (L1, L2, L3)
  - im Multiprozessorfall: private vs. shared
- inclusive vs. exclusive
- Cache-Typ
  - Instruction-Cache
  - Data-Cache
  - Unified-Cache
  - Victim-Cache
  - Trace-Cache
- Cache-Organisation
  - Direkt abgebildet (direct mapped)
  - Satzassoziativ (set associative)
  - Vollassoziativ (fully associative)
- Cachelinesize
- # Cachelines

Cache-Parameter (2)

- Ersetzungstrategie
  - least recently used (LRU)
  - least frequently used (LFU)
  - Round Robin
  - Random
- Schreibstrategie
  - write-through vs. write-back
  - write-allocation vs. no write-allocation
- Prefetching
- Cachekohärenzprotokoll (im Multiprozessorfall)
  - MOESI-Familie
  - Verzeichnisbasiert

Aufgabe 6a: Cache-Funktionsweise

Ein Cache wird durch 16bit adressiert und hat eine Kapazität von 16 Cachezeilen. Pro Zeile können 4 Wörter mit je 4 Bytes gespeichert werden. Die Verdrängungsstrategie sei LRU. Der Cache sei zu Beginn der Programmabarbeitung leer, d.h. alle Blöcke sind unbesetzt.

Aufgabe 6a: Cache-Funktionsweise

a.1) Wie viele bits werden bei den Organisationsformen: Direct Mapped, 2-fach satzassoziativ und vollassoziativ für den Tag, den Index und die Wortauswahl benötigt? Welche Kapazität hat der Cache?

Antwort

- 16bit Adresse, 16 Zeilen mit je 4 Wörter zu je 4 Bytes
- Länge L_{WA} der Wortadresse = $log_2(\#Woerter) \rightarrow L_{WA} = log_2(4) = 2$
- Länge L_{BA} der Byteauswahl = $log_2(\#BytesProWort) \rightarrow L_{BA} = log_2(4) = 2$
- Länge $L_{Index} = log_2(\#Saetze)$
- $\#Saetze = \frac{Cachegroesse}{Blockkapazitaet * Assoziativitaet}$

Aufgabe 6a: Cache-Funktionsweise

- Länge $L_{Tag} = L_{Adresse} - L_{Index} - L_{WA} - L_{BA}$

Cache	# Sätze	# Blöcke / Satz	Indexlänge	Taglänge
Direct Mapped	16	1	4	8
2-fach	8	2	3	9
vollassoziativ	1	16	0	12

- Kapazität in Byte =
  # Cachezeilen * # Wörter * # ByteProWort = 16 * 4 * 4 = 256 Byte

Aufgabe 6a: Cache-Funktionsweise

a.2) Gegeben sei folgende Speicherzugriffsfolge:

0x6FFA, 0xF121, 0x6FFB, 0x1831,
0x1835, 0x6FFC, 0x7000, 0x18BA,
0x18BC, 0x7004, 0x183F, 0x7008,
0x4000, 0x4004, 0x700A, 0x8086

Bestimmen Sie für alle drei Organisationsformen die Hit-Rate.

Adresse	Tag	DM Index	Hit / Miss
0x6FFA	0x6F	1111	-
0xF121	0xF1	0010	-
0x6FFB	0x6F	1111	+
0x1831	0x18	0011	-
0x1835	0x18	0011	+
0x6FFC	0x6F	1111	+
0x7000	0x70	0000	-
0x18BA	0x18	1011	-
0x18BC	0x18	1011	+
0x7004	0x70	0000	+
0x183F	0x18	0011	+
0x7008	0x70	0000	+
0x4000	0x40	0000	-
0x4004	0x40	0000	+
0x700A	0x70	0000	-
0x8086	0x80	1000	-

Hitrate = 8 / 16

Adresse	Tag	2-fach Index	Hit / Miss
0x6FFA	0x6F,1	111	-
0xF121	0xF1,0	010	-
0x6FFB	0x6F,1	111	+
0x1831	0x18,0	011	-
0x1835	0x18,0	011	+
0x6FFC	0x6F,1	111	+
0x7000	0x70,0	000	-
0x18BA	0x18,1	011	-
0x18BC	0x18,1	011	+
0x7004	0x70,0	000	+
0x183F	0x18,0	011	+
0x7008	0x70,0	000	+
0x4000	0x40,0	000	-
0x4004	0x40,0	000	+
0x700A	0x70,0	000	+
0x8086	0x80,1	000	-

Hitrate = 9 / 16

Adresse	Tag	Voll Index	Hit / Miss
0x6FFA	0x6FF		-
0xF121	0xF12		-
0x6FFB	0x6FF		+
0x1831	0x183		-
0x1835	0x183		+
0x6FFC	0x6FF		+
0x7000	0x700		-
0x18BA	0x18B		-
0x18BC	0x18B		+
0x7004	0x700		+
0x183F	0x183		+
0x7008	0x700		+
0x4000	0x400		-
0x4004	0x400		+
0x700A	0x700		+
0x8086	0x808		-

Hitrate = 9 / 16

Leistung von Caches

- Cache-Performance charakterisiert durch
  - Hit-Rate r_H
  - Miss-Rate $r_M = 1 - r_H$
  - Zugriffszeit bei Hit t_H
  - Zugriffszeit bei Miss t_M
- Mittlere Zugriffszeit:
  $t_a = r_H * t_H + r_M * t_{Mem}$
- Die Cache-Hierachie fließt in den Miss-Zweig ein
  $t_a = r_{H1} * t_{L1} + r_{M1} * (r_{H2} * t_{L2} + r_{M2} * t_{Mem})$

Aufgabe 6b: Performance von Caches

Ein Prozessor verfügt über eine 2-stufige Speicherhierachie! Der L1-Cache hat eine Zugriffszeit von $t_{L1} = 1ns$, der L2-Cache hat eine Zugriffszeit von $t_{L2} = 10ns$ und beim Hauptspeicher liegt die Zugriffszeit bei $t_{HS} = 100ns$.
In dieser Aufgabe kann der Hauptspeicher sämtliche Daten speichern.

b.1) Berechnen Sie die mittlere Zugriffszeit t_a, wenn folgende Hitraten gegeben sind: $r_{H1} = 80\%$ und $r_{H2} = 60\%$

Antwort
$$t_a = t_{L1} * r_{H1} + (1 - r_{H1}) * (t_{L2} * r_{H2} + t_{HS} * (1 - r_{H2}))$$
$$t_a = 1ns*80\%+(1-80\%)*(10ns*70\%+100ns*(1-70\%))$$
$$t_a = 0.8ns + 0.2 * (7ns + 30ns) = 8.2ns$$

Aufgabe 6b: Performance von Caches

b.2) Bei einem Wechsel der Fertigungstechnologie des Prozessors wurde der L2-Cache verkleinert, aber zusätzlich ein L3-Cache hinzugefügt. Der L2-Cache hat nun eine Zugriffszeit von $t_{L2} = 7ns$, der L3-Cache besitzt eine Zugriffszeit von $t_{L3} = 16ns$. Die Zugriffszeit des L1-Caches bleibt unverändert. Welche mittlere Zugriffszeit ergibt sich bei $r_{H1} = 80\%$, $r_{H2} = 50\%$ und $r_{H3} = 70\%$

Antwort $t_a = t_{L1} * r_{H1} + (1 - r_{H1}) * (t_{L2} * r_{H2} + (1 - r_{H2}) * (t_{L3} * r_{H3} + t_{HS} * (1 - r_{H3})))$

$t_a = 1ns * 80\% + (1 - 80\%) * (7ns * 50\% + (1 - 50\%) * (16ns * 70\% + 100ns * (1 - 70\%)))$

$t_a = 5,62ns$

(Set-) Reuse Distance

- Die Reuse Distance ist eine Metrik für die Lokalität bzw. Cache Performance eines Programms

(Set-) Reuse Distance: Definitionen

Reuse Distance

Anzahl der unterschiedlichen Hauptspeicherblöcke zwischen zwei Zugriffen auf denselben Hauptspeicherblock

Set-Reuse Distance

Anzahl der in denselben Cache-Satz geladenen unterschiedlichen Hauptspeicherblöcke zwischen zwei Zugriffen auf denselben Hauptspeicherblock

(Set-) Reuse Distance

weitere Definitionen

- wird zu ersten mal auf einen Block zugegriffen ist die Reuse Distance unendlich
- bei der Berechnung der Reuse Distance muss vorher eine Blockgröße festgelegt werden. Üblich sind 4 Byte oder die Größe der Cacheline

Hit/Miss Berechnung

gilt nur für LRU
SRD $\geq$ Assoziativität $\rightarrow$ Miss
SRD $<$ Assoziativität $\rightarrow$ Hit

Aufgabe 6c: (Set) Reuse Distance

Gegeben sei folgende Speicherzugriffsfolge:
0x0115, 0x011A, 0x0120, 0x012A, 0x0116, 0x1AAA, Ox1AAB, Ox1AAC, Ox0120, 0x1AA3, 0xAB10, 0xBB00, 0x0122, 0x1AA8, 0xBB01, 0x1AA4

c.1) Ein Hauptspeicherblock sei 8 Byte groß, berechnen Sie die Reuse Distance für jeden Speicherzugriff.

Adresse	Block	Reuse Distance
0x0115	A	∞
0x011A	B	∞
0x0120	C	∞
0x012A	D	∞
0x0116	A	3
0x1AAA	E	∞
0x1AAB	E	0
0x1AAC	E	0
0x0120	C	3
0x1AA3	F	∞
0xAB10	G	∞
0xBB00	H	∞
0x0122	C	3
0x1AA8	E	4
0xBB01	H	2
0x1AA4	F	4

Aufgabe 6c: (Set) Reuse Distance

Gegeben sei folgende Speicherzugriffsfolge:
0x0115, 0x011A, 0x0120, 0x012A, 0x0116, 0x1AAA, Ox1AAB, Ox1AAC, Ox0120, 0x1AA3, 0xAB10, 0xBB00, 0x0122, 0x1AA8, 0xBB01, 0x1AA4

c.2) Ein 2-fach satzassoziativer Cache besitzt vier Cachezeilen, wobei jede Cachezeile kann genau einen Hauptspeicherblock aufnehmen kann. Berechnen sie die Set Reuse Distance. Bestimmen sie anhand der Set Reuse Distance die Hit-Rate des Caches.

Adresse	Block	Satz	Set-Reuse Distance
0x0115	A	0	∞
0x011A	B	1	∞
0x0120	C	0	∞
0x012A	D	1	∞
0x0116	A	0	1
0x1AAA	E	1	∞
0x1AAB	E	1	0
0x1AAC	E	1	0
0x0120	C	0	1
0x1AA3	F	0	∞
0xAB10	G	0	∞
0xBB00	H	0	∞
0x0122	C	0	3
0x1AA8	E	1	0
0xBB01	H	0	1
0x1AA4	F	0	3

Hitrate = 6 / 16

Caches in Multiprozessorsystemen

- CPUs operieren auf lokalen Kopien
- Problem hierbei: Wahrung der Konsistenz (z.B. bei Verwendung von Write-Back), d.h. Cache und Hauptspeicher haben nicht die gleichen Daten

Konsistenz

Ein System ist konsistent, wenn alle Kopien eines Datenwortes im Hauptspeicher und den verschiedenen Cache-Speichern identisch sind

Cache-Kohärenz

Ein System ist Cache-Kohärent, wenn bei einem Zugriff immer auf das aktuellste Datum zugegriffen wird.

Kohärenzprotokolle

- Bus-Snooping
  - MOESI-Familie
  - Firefly
  - Dragon
  - Berkeley
  - ...
- Verzeichnisbasiert

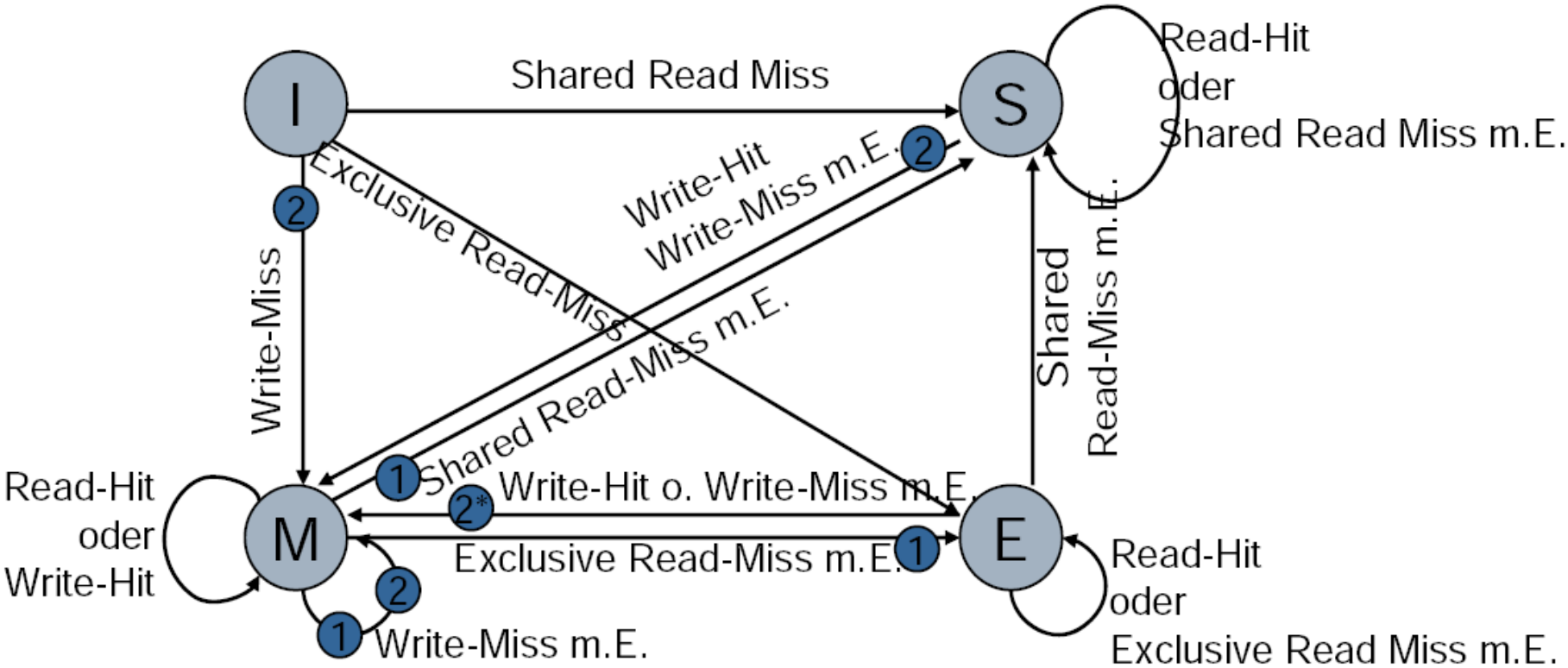
Bus-Snooping

- 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

Das MESI-Protokoll

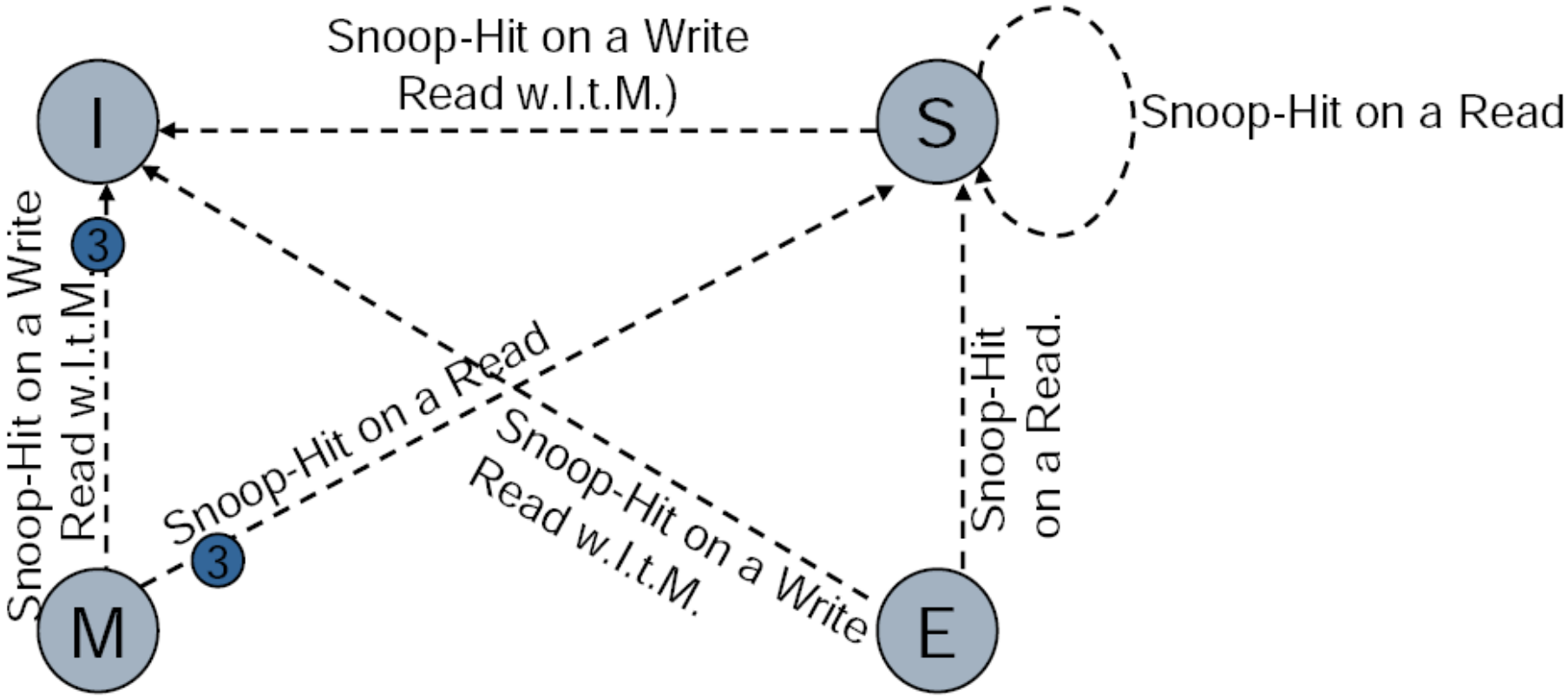
- MESI: Zustandsautomat mit 4 Zuständen
  - M: Modified
  - E: Exclusive unmodified
  - S: Shared unmodified
  - I: Invalid
- Jede Cachezeile befindet sich in einem dieser Zustände
- d.h. jede Cachezeile wird um 2 Zustandsbits erweitert
- Zustandsübergänge werden durch lokale und entfernte Speicherzugriffe ausgelöst
- Beim Write-Through-Verfahren sind nur die Zustände Shared und Invalid relevant

MESI-Zustandsautomat (1)



1. Cache-Zeile wird in den Hauptspeicher zurückkopiert
2. Cache-Zeilen in den anderen Caches mit gleicher Blockadresse werden invalidiert

MESI-Zustandsautomat (2)

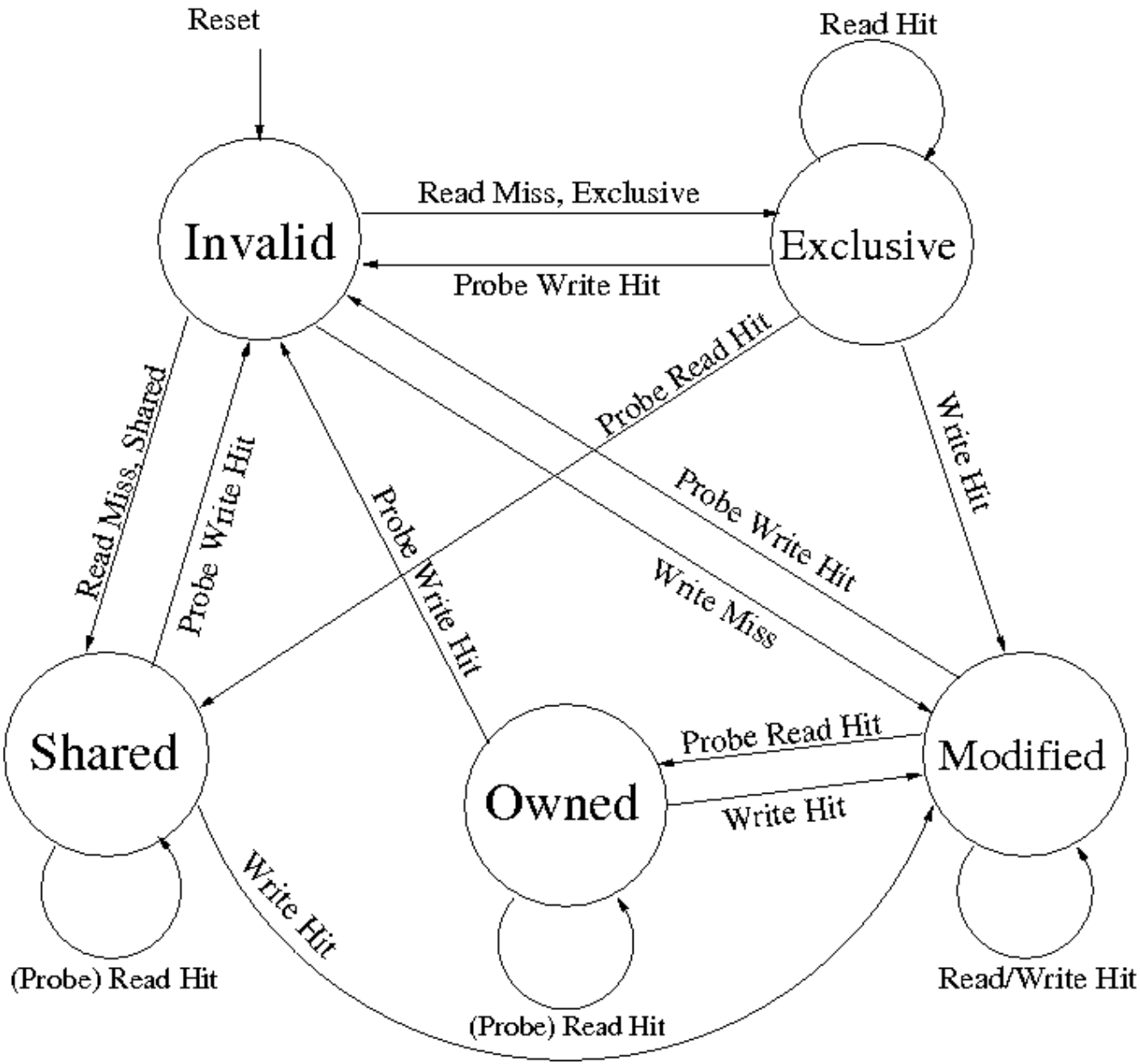


3 Retry-Signal wird aktiviert und danach wird die Cache-Zeile in den Hauptspeicher kopiert

MOESI-Protokoll

- Erweiterung des MESI-Protokolls um einen weiteren Zustand
- Neuer Zustand O: Owned (shared modified)
- 3 Zustandsbits nötig
- Vorteile wenn schnelle Cache-Cache Transfers möglich sind

MOESI Zustandsautomat



Aufgabe 6d: MOESI Cache-Kohärenz-Protokoll

Ein Zweiprozessor-System sei speichergekoppelt. Die Caches haben je eine Größe von zwei Cachelines, welche je genau ein Speicherwort aufnehmen können. Die Füllung des Caches erfolgt von der niedrigsten Cacheline aufwärts, sofern noch freie Line zur Verfügung stehen, anderenfalls wird die LRU-Strategie eingesetzt. Als Cache-Kohärenzprotokoll komme MOESI zu Einsatz.
Vervollständigen sie folgende Tabelle, geben sie jeweils Inhalt und MOESI-Zustand an.

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

Aufgabe 6d: MOESI Cache-Kohärenz-Protokoll

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

Aufgabe 6e: Verständnisfragen

e.1) Welche Gründe sprechen gegen den Aufbau des Hauptspeichers aus schnellen Cache-Speichern?

Antwort

- Kosten: Fläche: SRAM: ca. $100F^2$, DRAM ca. $10F^2$
- Technologische Gründe: Länge der Leiterbahnen begrenzen die max. erreichbare Taktfrequenz

Ausblick

- Nächste Übung: 19. Juli 2007
- Thema: Fehlertoleranz