timon.kelter [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerarchitektur
Sommersemester 2014

Übungsblatt 9

(10 Punkte)

Besprechung am Montag, 23. Juni 2014

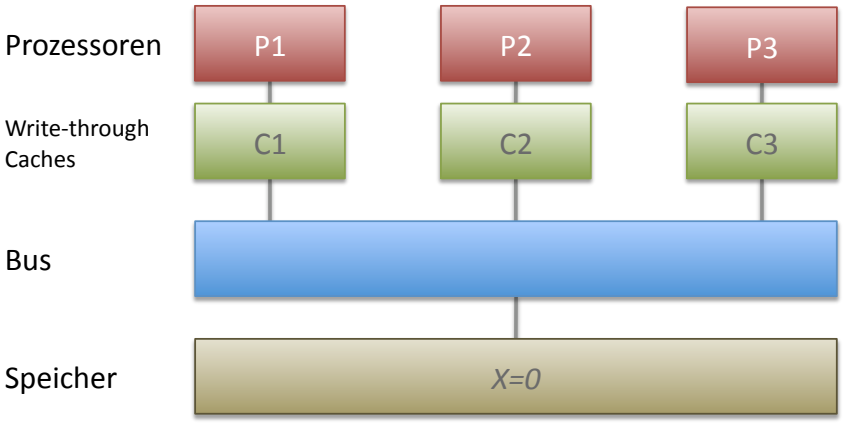

Abbildung 1: Ausgangssituation für Aufgabe 1.

9.1 Snoopy-Protokoll (3 Punkte)

Betrachten Sie das System aus Abb. 1. Gehen Sie davon aus, dass in diesem System ein Invalidation-basiertes Protokoll eingesetzt wird. Ergänzen Sie fehlende Informationen in der folgenden Sequenz.

1. Cache C1 lädt den Wert von *X* aus dem Speicher.
2. Cache C3 lädt den Wert von *X* aus dem Speicher.
3. Prozessor P3 schreibt den Wert 4 für *X* in seinen Cache (C3).
4. Der Cache Controller von C3 erzeugt eine Bus-Transaktion, um den Speicher zu aktualisieren.
5. _____ stellt fest, dass diese Schreibtransaktion für ihn relevant ist.
6. Der Cache Controller von C1 invalidiert seine Kopie des Blockes, der *X* enthält.
7. _____ aktualisiert den Wert der Speicherzelle *X* mit 4.
8. Die Leseoperation von P1 von *X* erzeugt einen *read-miss* im lokalen Cache C1.
9. _____ lädt den aktuellen Wert 4 aus dem Speicher.
10. Die Leseoperation von P2 von *X* erzeugt einen *read-miss* im lokalen Cache C2.
11. _____
12. P1 schreibt den Wert 8 für *X* in seinen Cache (C1).
13. Der Cache Controller von C1 erzeugt eine Bus-Transaktion, um den Speicher zu aktualisieren.
14. _____

15. Die Cache Controller von C2 und C3 invalidieren ihre Kopie des Blockes, der *X* enthält.
16. Der Speicher aktualisiert den Wert der Speicherzelle *X* mit 8.

9.2 Bus-Snooping mit Write-Through Caches (3 Punkte)

In dem System (s. Abb. 1) wird nun Bus-Snooping zum Erhalten der Cache-Kohärenz durchgeführt. Geben Sie für die folgenden Aussagen an, ob sie wahr oder falsch sind. Nehmen Sie dafür an, das jede Cache-Zelle nur die Zustände *valid* und *invalid* haben kann. Sie können *nicht* annehmen, dass es sich bei dem verwendeten Protokoll um ein MSI-Protokoll handelt!

a) Alle Transaktionen auf dem Bus sind für die Cache Controller in verschiedenen Reihenfolgen sichtbar.

☐ wahr ☐ falsch, Korrektur _____

_____

b) Wenn ein Prozessor auf Speicherzellen zugreift, überprüft der Cache Controller den Zustand des Caches und initiiert, falls nötig, Transaktionen auf dem Bus, um auf den Speicher (lesend/schreibend) zuzugreifen.

☐ wahr ☐ falsch, Korrektur _____

_____

c) Der Ausgangszustand einer Speicherzelle im lokalen Cache sei *invalid*. Der Prozessor liest den Wert der Speicherzelle direkt aus dem Cache.

☐ wahr ☐ falsch, Korrektur _____

_____

d) Der Ausgangszustand einer Speicherzelle im lokalen Cache sei *valid*. Sowohl Lese- als auch Schreibzugriffe des Prozessors führen zu keiner Änderung des Zustandes.

☐ wahr ☐ falsch, Korrektur _____

_____

e) Der Ausgangszustand einer Speicherzelle im lokalen Cache sei *invalid*. Schreibzugriffe des Prozessors auf diese Speicherzelle führen zu einer Änderung des Zustandes.

☐ wahr ☐ falsch, Korrektur _____

_____

f) Der Ausgangszustand einer Speicherzelle im lokalen Cache sei *invalid*. Lesezugriffe des Prozessors auf diese Speicherzelle führen zu keiner Änderung des Zustandes.

☐ wahr ☐ falsch, Korrektur _____

_____

g) Serialisierung bedeutet, dass Schreibzugriffe für andere Prozessoren sichtbar werden. Propagation bedeutet, dass alle Schreibzugriffe in derselben Reihenfolge von allen Prozessoren gesehen werden.

☐ wahr ☐ falsch, Korrektur _____

_____

h) Protokolle, die Cache-Kopien bei einem Schreibzugriff invalidieren, heißen invalidation-basiert. Protokolle, die Cache-Kopien bei einem Schreibzugriff aktualisieren, heißen update-basiert.

☐ wahr ☐ falsch, Korrektur _____

_____

9.3 Verzeichnisbasierte Kohärenzprotokolle (4 Punkte)

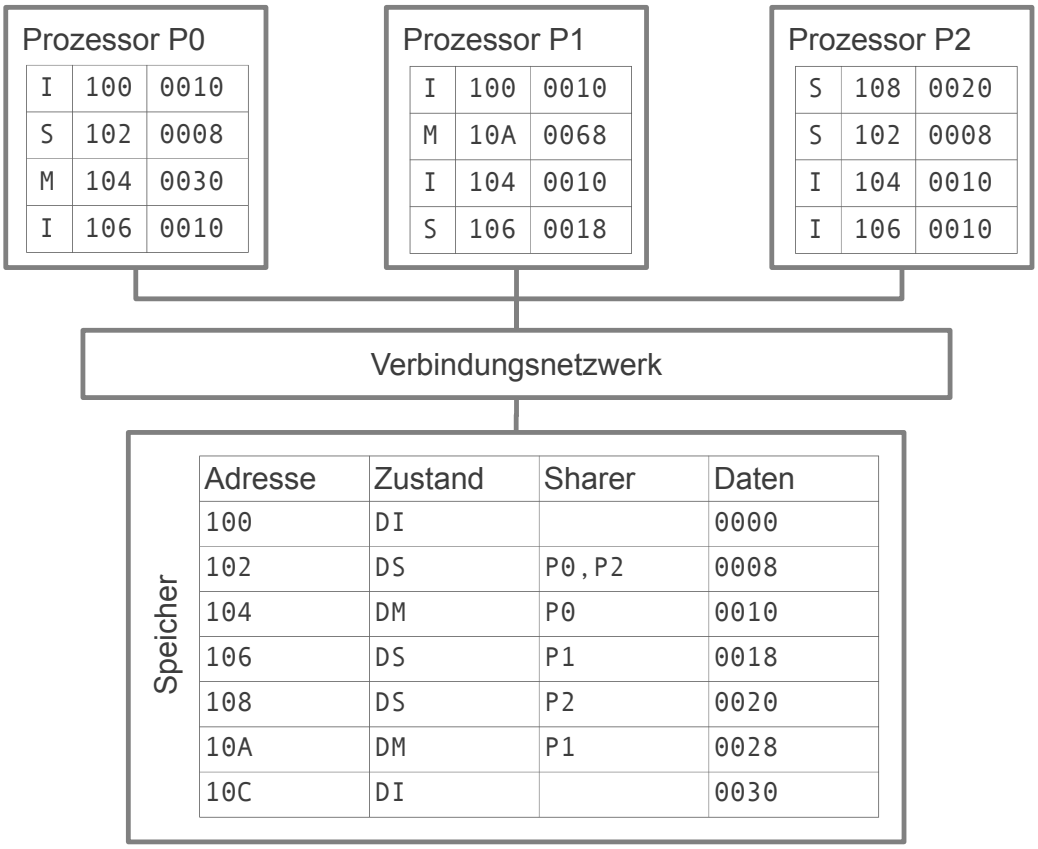

Abbildung 2: Ausgangssituation für Aufgabe 3.

Gegeben sei ein System wie in Abbildung 2 skizziert, mit 3 Prozessoren und einem verzeichnisbasierten Cachekohärenzmechanismus nach dem in der Vorlesung vorgestellten einfachen Verzeichnisprotokoll. Der Initialzustand der Caches und des Verzeichnisses ist in Abbildung 2 ebenfalls enthalten. Hierbei sind die Spalten der Cache-Zustandstabelle von links nach rechts: Kohärenzstatus, Address-Tag und lokal in dieser Cachezeile gespeicherte Daten. Die Caches sind direct-mapped und enthalten pro Zeile jeweils 2 Datenbytes (4 Hex-Zeichen). Zur Vereinfachung enthält die Address-Tag Spalte hier immer die komplette Adresse, d.h. Sie müssen aus den Zugriffsadressen keinen Index mehr extrahieren, sondern können die Caches direkt mit der vollen Adresse durchsuchen. Die Abkürzungen der Zustände sind:

- I Lokal “Invalid”
- S Lokal “Shared (read only)”
- M Lokal “Modified (read/write)”
- DI Verzeichnis “Uncached”

DS Verzeichnis “Shared (read only)”

DM Verzeichnis “Modified (read/write)”

a) Geben Sie für jede der folgenden Aktionssequenzen an, welche Nachrichten zwischen Caches und Directory verschickt werden, und wie der finale Zustand der betroffenen Cache- und Verzeichnis-Einträge ist. Die Aktionen sind in der Form: `<Prozessornummer>: <Operation> <Adresse> <Eingabe (bei Schreibvorgängen)>` vorgegeben. Lese- und Schreibzugriffe greifen jeweils nur auf eines der beiden Bytes in der Cachezeile zu. Nehmen Sie an, dass jede Aktualisierung der Cache- bzw. Verzeichniszustände erst vollständig abgearbeitet wird, bevor der nächste Zugriff stattfindet. Für jede der nachfolgenden 8 Sequenzen ist der Initialzustand des Systems so wie in Abbildung 2 vorgegeben, d.h. die 8 Sequenzen sind *nicht* als aufeinander folgend zu interpretieren.

- `P0: lese 100`
- `P0: lese 10A`
- `P0: schreibe 10A <-- 78`
- `P0: lese 108`
- `P0: lese 108`
  `P1: lese 108`
- `P0: lese 108`
  `P1: schreibe 108 <-- 80`
- `P0: schreibe 108 <-- 80`
  `P1: lese 108`
- `P0: schreibe 108 <-- 80`
  `P2: schreibe 10C <-- 90`

b) Erweitern Sie die Zustandsautomaten des einfachen Directory-Protokolls um einen *Owned*-Zustand, der folgendes beschreibt:

- Der Block wurde lokal modifiziert und sein Inhalt wurde noch nicht in den Hauptspeicher zurückgeschrieben (*Owned* kann also nur vom *Exclusive*-Zustand aus erreicht werden),
- nachfolgende Lese-Anfragen von anderen Caches werden direkt vom Owner-Cache beantwortet, ohne Umweg über den Hauptspeicher und
- Schreibzugriffe auf den Owned-Block und lokale Verdrängung des Blocks führen zum Zurückschreiben des Inhalts in den Hauptspeicher.

Benutzen Sie als Ein-/Ausgaben der Automaten nur die bereits in den ursprünglichen Automaten definierten Symbole und zusätzlich

- Eine Nachricht *O-Fetch*. Während *Fetch* bewirkt, dass der Empfänger-Cache einen Cacheblock an das Verzeichnis zurückschickt und dessen lokalen Status auf “Shared” setzt, bewirkt *O-Fetch*, dass der Empfänger-Cache einen Cacheblock an einen anderen Cache schickt und den lokalen Status auf “Owned” setzt.
- Eine Aktion *Send data* die ein Cache als Reaktion auf den Empfang der *O-Fetch*-Nachricht durchführt.

Allgemeine Hinweise: Die Übungstermine und weitere Informationen finden Sie unter http://ls12-www.cs.tu-dortmund.de/daes/de/lehre/lehrveranstaltungen/sommersemester-2014/rechnerarchitektur.html. Die Übungszettel werden zum Semesterbeginn online gestellt und sollen eigenständig bis zum jeweiligen Stichtag gelöst werden. Die Lösungen werden in den Gruppen besprochen. Auf Wunsch kann für diese Veranstaltung ein Übungsschein ausgestellt werden. Hierzu müssen die selbst erstellten Lösungen jeweils vor der Besprechung der Aufgaben beim Übungsgruppenleiter abgegeben werden. Dabei müssen 45% der Gesamtpunkte bei den Übungszetteln erreicht und eigene Lösungen in der Übungsgruppe präsentiert werden. Für die Teilnahme an der Klausur nach BPO 2013 / der Fachprüfung nach DPO 2001 ist der Übungsschein *nicht* erforderlich.