



(19)
Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 602 04 213 T2** 2006.06.29

(12)

Übersetzung der europäischen Patentschrift

(97) **EP 1 363 192 B1**

(51) Int Cl.⁸: **G06F 12/08** (2006.01)

(21) Deutsches Aktenzeichen: **602 04 213.5**

(96) Europäisches Aktenzeichen: **02 025 692.1**

(96) Europäischer Anmeldetag: **20.11.2002**

(97) Erstveröffentlichung durch das EPA: **19.11.2003**

(97) Veröffentlichungstag

der Patenterteilung beim EPA: **18.05.2005**

(47) Veröffentlichungstag im Patentblatt: **29.06.2006**

(30) Unionspriorität:

380740 P	15.05.2002	US
269828	11.10.2002	US

(84) Benannte Vertragsstaaten:

AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE, SK, TR

(73) Patentinhaber:

Broadcom Corp., Irvine, Calif., US

(72) Erfinder:

Rowlands, Joseph B., 95054 Santa Clara, US

(74) Vertreter:

**Bosch, Graf von Stosch, Jehle
Patentanwalts-gesellschaft mbH, 80639 München**

(54) Bezeichnung: **Level 2 Cache mit lokaler Beibehaltung von Kohärenzblöcken**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

[0001] Die Erfindung bezieht sich auf kohärente Speichersysteme, einschließlich kohärenter verteilter Speichersysteme, wie Cache-kohärente Speichersysteme mit uneinheitlichem Speicherzugriff (CC-NUMA = Cache-coherent nonuniform memory access).

[0002] Speichersysteme (einschließlich Hauptspeicher und beliebige Caches im System) sind oft kohärent ausgebildet. D.h., auch wenn mehrere Kopien von Daten aus einem bestimmten Speicherplatz in dem Speichersystem vorliegen können, gibt ein Lesen jenes Speicherplatzes die aktuellsten Daten zurück, die in jenen Speicherplatz geschrieben wurden. Typischerweise kann ein kohärentes System einen oder mehrere kohärente Agenten und eine Speichersteuerung einschließen, welche über eine Verdrahtung beliebiger Art verbunden ist.

[0003] Ein Mechanismus zum Abgleichen kohärenter Systeme an größere Zahlen von kohärenten Agenten ist die Verwendung von verteilten Speichersystemen. In einem solchen System ist der Speicher unter verschiedenen Knoten (welche auch kohärente Agenten einschließen können) aufgeteilt und die Knoten sind untereinander verbunden. Ein kohärenter Agent in einem Knoten kann auf Speicher in einem anderen Knoten zugreifen. Eine Klasse von Verfahren zur Aufrechterhaltung von Kohärenz in einem verteilten Speichersystem wird als Cache-kohärent, mit uneinheitlichem Speicherzugriff (CC-NUMA) bezeichnet. In einem CC-NUMA System kann der Speicherzugriff eine variierende Wartezeit aufweisen (z.B. kann auf Speicher in demselben Knoten wie ein Agent schneller zugegriffen werden, als auf Speicher in einem anderen Knoten, und Zugriffe auf verschiedene Knoten können auch variierende Wartezeiten aufweisen), aber Kohärenz wird aufrechterhalten.

[0004] Viele CC-NUMA Systeme implementieren ein Verzeichnis in jedem Knoten, welches den Zustand, in anderen Knoten, von jedem Cacheblock in dem zu jenem Knoten zugeordneten Speicher verfolgt. Zusätzlich muss jedoch der Zustand des Cacheblockes in den anderen Knoten auch in jenen anderen Knoten verfolgt werden. Z.B., wenn ein Cacheblock in einem der anderen Knoten modifiziert ist, muss jener andere Knoten den Cacheblock zurücksenden, wenn der Cacheblock aus jenem anderen Knoten ausgesondert wird.

[0005] US-B1-6 219 755 bezieht sich auf eine Schaltung zum Anpassen eines erweiterbaren L2 Cache Moduls in einem Computersystem, welches automatisch bestimmte Steuersignale sendet, um Cache Kohärenz aufrechtzuerhalten.

[0006] EP-A-0 936 55 bezieht sich auf ein Ca-

che-Kohärenzprotokoll, welches einen "markierten" Kohärenzstatus verwendet, um die Zuständigkeit für ein Speicher-Zurückschreiben zu verfolgen.

[0007] Es ist eine Aufgabe der vorliegenden Erfindung, eine Vorrichtung und ein Verfahren bereitzustellen, welche geeignet sind, mehrere Kopien von Daten aus bestimmten Speicherplätzen zu verwalten, wobei der Speicher unter verschiedenen untereinander verbundenen Knoten verteilt ist.

[0008] Diese Aufgabe wird durch die Vorrichtung und das Verfahren gemäß den unabhängigen Vorrichtungs- bzw. Verfahrensansprüchen gelöst. Vorteilhafte Ausführungsformen der Erfindung sind in den abhängigen Ansprüchen definiert.

[0009] Die folgende ausführliche Beschreibung nimmt Bezug auf die beiliegenden Zeichnungen, welche nun kurz beschrieben werden.

[0010] [Fig. 1](#) zeigt ein Blockdiagramm einer Ausführungsform eines Knotens.

[0011] [Fig. 2](#) zeigt ein Blockdiagramm einer Ausführungsform von mehreren, in einem System verbundenen Knoten.

[0012] [Fig. 3](#) zeigt eine Tabelle, welche einen beispielhaften Satz von Kohärenzbefehlen erläutert und eine Tabelle, welche einen beispielhaften Satz von Vorgängen, gemäß einer Ausführungsform des in [Fig. 1](#) gezeigten Knotens erläutert.

[0013] [Fig. 4](#) zeigt ein Blockdiagramm eines Adressplatzes, welcher von einer Ausführungsform der in [Fig. 1](#) und [Fig. 2](#) gezeigten Knoten unterstützt wird.

[0014] [Fig. 5](#) zeigt einen Entscheidungsbaum, welcher die Arbeitsweise einer Ausführungsform eines Knotens für einen Lesevorgang auf der Verbindung innerhalb des Knotens erläutert.

[0015] [Fig. 6](#) zeigt einen Entscheidungsbaum, welcher die Arbeitsweise einer Ausführungsform eines Knotens für einen Schreibvorgang auf der Verbindung innerhalb des Knotens erläutert.

[0016] [Fig. 7](#) zeigt ein Diagramm, welches die Arbeitsweise einer Ausführungsform der Speicherbrücke für entfernte Kohärenzbefehle erläutert, welche von der Speicherbrücke empfangen werden.

[0017] [Fig. 8](#) zeigt ein Blockdiagramm einer Ausführungsform einer in [Fig. 1](#) gezeigten Speicherbrücke.

[0018] [Fig. 9](#) zeigt ein Blockdiagramm einer Ausführungsform zweier Knoten, welches eine Übertra-

gung von Eigentumsrecht für einen Vorgang erläutert.

[0019] [Fig. 10](#) zeigt ein Blockdiagramm einer Ausführungsform eines entfernten Leistungsverzeichnisses und einer zugehörigen Schaltung.

[0020] [Fig. 11](#) zeigt ein Blockdiagramm einer Ausführungsform eines Eintrags in einem entfernten Leistungsverzeichnis.

[0021] [Fig. 12](#) zeigt ein Flussdiagramm, welches bestimmte Arbeitsvorgänge einer Ausführungsform der in [Fig. 10](#) gezeigten RLD Steuerschaltung erläutert.

[0022] [Fig. 13](#) zeigt ein Blockdiagramm, welches beispielhafte Cachezustände in einer Ausführungsform eines Knotens erläutert.

[0023] [Fig. 14](#) zeigt ein Flussdiagramm, welches bestimmte Arbeitsvorgänge einer Ausführungsform des in [Fig. 1](#) gezeigten L2 Caches in Erwiderung auf einen Lesevorgang erläutert.

[0024] [Fig. 15](#) zeigt ein Flussdiagramm, welches bestimmte Arbeitsvorgänge einer Ausführungsform des in [Fig. 1](#) gezeigten L2 Caches während eines Aussonderns aus dem L2 Cache erläutert.

[0025] [Fig. 16](#) zeigt ein Blockdiagramm eines Computer-zugänglichen Mediums.

[0026] [Fig. 17](#) zeigt eine Tabelle, welche beispielhafte Aktualisierungen einer Ausführungsform eines entfernten Leistungsverzeichnisses erläutert.

Knotenübersicht

[0027] Beginnend nun mit [Fig. 1](#), ist ein Blockdiagramm einer Ausführungsform eines Knotens **10** dargestellt. In dem Ausführungsbeispiel gemäß [Fig. 1](#) schließt der Knoten **10** einen oder mehrere Prozessoren **12A–12N**, eine Speichersteuerung **14**, einen Schalter **18**, einen Satz von Interface-Schaltungen **20A–20C**, eine Speicherbrücke **32** und ein L2 Cache **36** ein. Die Speicherbrücke **32** schließt ein entferntes Leistungsverzeichnis **34** ein. Der Knoten **10** schließt eine Verbindung **22** ein, mit welcher die Prozessoren **12A–12N**, die Speichersteuerung **14**, das L2 Cache **36**, die Speicherbrücke **32** und das entfernte Leistungsverzeichnis **34** verbunden sind. Der Knoten **10** ist durch die Speichersteuerung **14** mit einem Speicher **24** verbunden. Die Schnittstellen-Schaltungen **20A–20C** schließen jeweils eine Empfangsschaltung (Rx) **26A–26C** und eine Übertragungsschaltung (Tx) **28A–28C** ein. Der Knoten **10** ist mit einem Satz von Schnittstellen **30A–30C** durch entsprechende Schnittstellen-Schaltungen **20A–20C** verbunden. Die Schnittstellen-Schaltungen **20A–20C** sind mit dem

Schalter **18** verbunden, welcher weiter mit der Speicherbrücke **32** verbunden ist. Ein Konfigurationsregister **38**, welches eine Knotennummer (Knoten #) für den Knoten **10** speichert, ist auch in [Fig. 1](#) dargestellt. Das Konfigurationsregister **38** ist mit dem L2 Cache **36**, der Speichersteuerung **14**, der Speicherbrücke **32** und den Schnittstellen-Schaltungen **20A–20C** in der Ausführungsform gemäß [Fig. 1](#) verbunden. Zusätzlich können die Prozessoren **12A–12N** eingerichtet sein, die Knotennummer von dem Konfigurationsregister **38** zu empfangen.

[0028] Der Knoten **10** kann knoteninterne Kohärenz für Vorgänge auf der Verbindung **22** unterstützen. Zusätzlich kann der Knoten **10** knotenübergreifende Kohärenz mit anderen Knoten (z.B. eine CC-NUMA Kohärenz in einer Ausführungsform) unterstützen. Im Allgemeinen, wie hierin verwendet, schließt eine Speicherbrücke eine Schaltung ein, welche ausgebildet ist, knotenübergreifende Kohärenzfunktionen innerhalb eines Knotens zu verwalten. Insbesondere kann in einer Ausführungsform, wenn ein Vorgang auf der Verbindung **22** (z.B. ein von den Prozessoren **12A–12N** ausgegebener Vorgang) auf einen Cacheblock zugreift, welcher entfernt von dem Knoten **10** ist (d.h. der Cacheblock ist Teil des mit einem anderen Knoten verbundenen Speichers) und der Knoten **10** kein ausreichendes Eigentumsrecht besitzt, um den Vorgang durchzuführen, die Speicherbrücke **32** einen oder mehrere Kohärenzbefehle an die anderen Knoten ausgeben, um das Eigentumsrecht zu erhalten (und, in manchen Fällen, eine Kopie des Cacheblocks). In ähnlicher Weise kann, wenn der Vorgang auf einen lokalen Cacheblock zugreift, aber ein oder mehrere andere Knoten eine Kopie des Cacheblocks besitzen, die Speicherbrücke **32** Kohärenzbefehle an andere Knoten ausgeben. Des Weiteren kann die Speicherbrücke **32** Kohärenzbefehle von anderen Knoten empfangen und kann Vorgänge auf der Verbindung **22** durchführen, um die Kohärenzbefehle auszuführen.

[0029] In einer Ausführungsform kann ein Knoten, wie Knoten **10**, einen damit verbundenen Speicher aufweisen (z.B. Speicher **24**). Der Knoten kann zuständig sein für das Verfolgen des Zustandes, in anderen Knoten, von jedem Cacheblock aus dem Speicher in jenem Knoten. Ein Knoten wird als der "Heimatknoten" für Cacheblöcke aus dem zu jenem Knoten zugeordneten Speicher bezeichnet. Ein Knoten wird als ein "entfernter Knoten" für einen Cacheblock bezeichnet, wenn der Knoten nicht der Heimatknoten für jenen Cacheblock ist. In ähnlicher Weise wird ein Cacheblock als ein lokaler Cacheblock in dem Heimatknoten für jenen Cacheblock bezeichnet und als ein entfernter Cacheblock in anderen Knoten.

[0030] Im allgemeinen kann ein entfernter Knoten den Kohärenzprozess beginnen, indem er eine Kopie eines Cacheblocks aus dem Heimatknoten von je-

nem Cacheblock anfordert, wobei er einen Kohärenzbefehl verwendet. Die Speicherbrücke **32** in dem entfernten Knoten kann z.B. einen Vorgang auf der Verbindung **22** feststellen, welcher auf den Cacheblock zugreift, und kann feststellen, dass der entfernte Knoten kein ausreichendes Eigentumsrecht an dem Cacheblock besitzt, um den Vorgang zu vollenden (z.B. kann er überhaupt keine Kopie des Cacheblocks besitzen oder er kann eine mehrfach benutzte Kopie besitzen und ein ausschließliches Eigentumsrecht benötigen, um den Vorgang zu vollenden). Die Speicherbrücke **32** in den entfernten Knoten kann den Kohärenzbefehl erzeugen und an den Heimatknoten senden, um die Kopie zu erhalten oder um ein ausreichendes Eigentumsrecht zu erhalten. Die Speicherbrücke **32** in dem Heimatknoten kann ermitteln, ob irgendwelche Zustandsänderungen in anderen Knoten durchzuführen sind, damit das angeforderte Eigentumsrecht dem entfernten Knoten gewährt werden kann, und kann Kohärenzbefehle senden (z.B. Prüfbefehle), um die Zustandsänderungen zu bewirken. Die Speicherbrücke **32** kann in jedem Knoten, welcher die Prüfbefehle empfängt, die Zustandsänderungen bewirken und die Prüfbefehle erwidern. Sobald die Antworten empfangen worden sind, kann die Speicherbrücke **32** in dem Heimatknoten dem entfernten Knoten antworten (z.B. mit einem den Cacheblock einschließenden Fill Befehl).

[0031] Das entfernte Leitungsverzeichnis **34** kann in dem Heimatknoten verwendet werden, um den Zustand des lokalen Cacheblockes in den entfernten Knoten zu verfolgen. Das entfernte Leitungsverzeichnis **34** wird jedes Mal aktualisiert, wenn ein Cacheblock an einen entfernten Knoten übertragen wird, der entfernte Knoten den Cacheblock an den Heimatknoten zurücksendet oder der Cacheblock wird über Prüfbefehle für ungültig erklärt wird. Wie hier verwendet, bezeichnet der "Zustand" eines Cacheblockes in einem bestimmten Knoten eine Anzeige des Eigentumsrechts, welches der bestimmte Knoten für den Cacheblock gemäß dem durch die Knoten implementierten Kohärenzprotokoll hat. Bestimmte Stufen von Eigentumsrecht können keinen Zugriff, nur Lesezugriff, oder Schreib-Lese-Zugriff auf den Cacheblock erlauben. Z.B. werden in einem Ausführungsbeispiel die modifizierten, mehrfach benutzten und ungültigen Zustände in dem knotenübergreifenden Kohärenzprotokoll unterstützt. In dem modifizierten Zustand kann der Knoten den Cacheblock lesen und beschreiben und der Knoten ist verantwortlich für das Zurücksenden des Blockes an den Heimatknoten, wenn er aus dem Knoten ausgesondert wird. In dem mehrfach benutzten Zustand kann der Knoten den Cacheblock lesen, aber nicht den Cacheblock beschreiben, ohne einen Kohärenzbefehl an den Heimatknoten zu übertragen, um einen modifizierten Zustand für den Cacheblock zu erhalten. In dem ungültigen Zustand kann der Knoten den Cacheblock nicht lesen oder beschreiben (d.h., der Knoten besitzt kei-

ne gültige Kopie des Cacheblockes). Andere Ausführungsformen können andere Kohärenzprotokolle verwenden (z.B. das MESI Protokoll, welches die modifizierten, mehrfach benutzten und ungültigen Zustände und einen ausschließlichen Zustand einschließt, in welchen der Cacheblock noch nicht aktualisiert worden ist, aber der Knoten den Cacheblock lesen und beschreiben darf, oder das MOESI Protokoll, welches die modifizierten, ausschließlichen, mehrfach benutzten und ungültigen Zustände und einen zugeordneten Zustand einschließt, welcher anzeigt, dass mehrfach benutzte Kopien des Blockes existieren können, aber die Kopie im Hauptspeicher nicht aktuell ist). In einer Ausführungsform können die Agenten innerhalb des Knotens das MESI Protokoll für knoteninterne Kohärenz implementieren. So kann der Knoten als einen Zustand in der knotenübergreifenden Kohärenz aufweisend dargestellt werden und einzelne Agenten können einen Zustand in der knoteninternen Kohärenz aufweisen (konsistent mit dem knotenübergreifenden Kohärenzzustand für den Knoten, welcher den Agenten enthält).

[0032] Kohärenzbefehle werden auf einer der Schnittstellen **30A–30C** durch die zugehörige Schnittstellenschaltung **20A–20C** übertragen und empfangen. Die Schnittstellenschaltungen **20A–20C** empfangen Kohärenzbefehle zur Übertragung von der Speicherbrücke **32** und übertragen von den Schnittstellen **30A–30C** empfangene Kohärenzbefehle an die Speicherbrücke **32** zur Bearbeitung, falls die Kohärenzbefehle eine Bearbeitung in dem Knoten **10** erfordern. In einigen Ausführungsformen kann ein Kohärenzbefehl empfangen werden, welcher durch den Knoten **10** zu einem anderen Knoten weitergeleitet wird, und keine Verarbeitung in dem Knoten **10** erfordert. Die Schnittstellenschaltungen **20A–20C** können eingerichtet sein, derartige Befehle zu erkennen und sie zurückzusenden (durch eine andere Schnittstellenschaltung **20A–20C**), ohne die Speicherbrücke **32** einzubeziehen.

[0033] In dem erläuterten Ausführungsbeispiel, sind die Schnittstellenschaltungen **20A–20C** mit der Speicherbrücke **32** durch den Schalter **18** verbunden (obwohl in anderen Ausführungsformen die Schnittstellenschaltungen **20A–20C** direkte Pfade zu der Speicherbrücke **32** aufweisen können). Der Schalter **18** kann die Schnittstellenschaltungen **20A–20C** (und insbesondere die Rx-Schaltungen **26A–26C** in dem erläuterten Ausführungsbeispiel) selektiv mit anderen Schnittstellenschaltungen **20A–20C** (und insbesondere die Tx-Schaltungen **28A–28C** in dem erläuterten Ausführungsbeispiel) oder mit der Speicherbrücke **32** verbinden, um empfangene Kohärenzbefehle zu übertragen. Der Schalter **18** kann auch die Speicherbrücke **32** selektiv mit den Schnittstellenschaltungen **20A–20C** (und insbesondere an die Tx-Schaltungen **28A–28C** in dem erläuterten Ausführungsbeispiel) verbinden, um von der Speicherbrücke

cke erzeugte Kohärenzbefehle von der Speicherbrücke **32** zu den Schnittstellenschaltungen **20A–20C** zur Übertragung auf der zugehörigen Schnittstelle **30A–30C** übertragen. Der Schalter **18** kann auch Anforderungs/Gewährungsschnittstellen zu jeder der Schnittstellenschaltungen **20A–20C** und der Speicherbrücke **32** aufweisen, um Übertragungen anzufordern und diese Übertragungen zu gewähren. Der Schalter **18** kann auch einen Eingangspfad von jeder Quelle (die Rx-Schaltungen **26A–26C** und die Speicherbrücke **32**) und einen Ausgangspfad zu jedem Ziel (die Tx-Schaltungen **28A–28C** und die Speicherbrücke **32**) aufweisen und kann einen gewährten Eingangspfad mit einem gewährten Ausgangspfad zur Übertragung eines Kohärenzbefehles verbinden (oder eines Teiles davon, wenn Kohärenzbefehle größer sind als eine Übertragung durch den Schalter **18**). Die Verbindungen können dann auf dem nächsten gewährten Eingangspfad und gewährten Ausgangspfad abgeändert werden. Mehrere unabhängige Eingangspfad-/Ausgangspfad-Gewährungen können gleichzeitig erfolgen.

[0034] In einer Ausführungsform können die Schnittstellen **30A–30C** einen Satz von virtuellen Kanälen unterstützen, in welchen Befehle übertragen werden. Jeder virtuelle Kanal ist als unabhängig von den anderen virtuellen Kanälen fließend definiert, auch wenn die virtuellen Kanäle sich bestimmte physikalische Ressourcen teilen können (z.B. die Schnittstelle **30A–30C**, auf welcher die Befehle fließen). Diese virtuellen Kanäle können auf interne virtuelle Kanäle (hier bezeichnet als virtuelle Schaltkanäle) abgebildet werden. Der Schalter **18** kann virtuelle Kanäle beachten. D.h., der Schalter **18** kann eine Verbindung zwischen einer Quelle und einem Ziel gewähren, welche nicht nur auf der Fähigkeit der Quelle, Daten zu übertragen und des Zieles, Daten zu empfangen, basiert, sondern auch auf der Fähigkeit der Quelle, Daten in einem besonderen virtuellen Schaltkanal zu übertragen und der Fähigkeit des Zieles, Daten aus jenem virtuellen Schaltkanal zu empfangen. So können Anforderungen von Quellen das Ziel und den virtuellen Kanal anzeigen, auf welchem die Daten zu übertragen sind, und Anforderungen von Zielen können den virtuellen Kanal anzeigen, auf welchem Daten empfangen werden können.

[0035] Allgemein gesprochen kann ein Knoten einen oder mehrere kohärente Agenten (gepunktete Einfassung **16** in [Fig. 1](#)) einschließen. In dem Ausführungsbeispiel gemäß [Fig. 1](#) können die Prozessoren **12A–12N**, das L2 Cache **36** und die Speichersteuerung **14** Beispiele von kohärenten Agenten **16** sein. Zusätzlich kann die Speicherbrücke **32** ein kohärenter Agent sein (im Auftrag von anderen Knoten). Jedoch können andere Ausführungsformen auch andere kohärente Agenten einschließen, wie eine Brücke zu einer oder mehreren I/A Schnittstellenschaltungen, oder die I/A Schnittstellenschaltun-

gen selbst. Im allgemeinen schließt ein Agent jede Schaltung ein, welche an den Vorgängen auf einer Verbindung teilnimmt. Ein kohärenter Agent ist ein Agent, welcher geeignet ist, kohärente Vorgänge durchzuführen und, in Bezug auf Vorgänge, in kohärenter Weise arbeitet. Ein Vorgang ist eine Kommunikation auf einer Verbindung. Der Vorgang wird von einem Agent auf der Verbindung initiiert und kann einen oder mehrere Agenten als Ziel des Vorganges haben. Lese Vorgänge spezifizieren einen Datentransfer von einem Ziel zu der Quelle, während Schreibvorgänge einen Datentransfer von der Quelle zu dem Ziel spezifizieren. Andere Vorgänge können in einigen Ausführungsformen verwendet werden, um zwischen Agenten ohne Datentransfer zu kommunizieren.

[0036] Jede der Schnittstellenschaltungen **20A–20C** sind eingerichtet, auf den zugehörigen Schnittstellen **30A–30C**, mit welchen sie verbunden sind, zu empfangen und zu übertragen. Die Rx-Schaltungen **26A–26C** verwalten das Empfangen von Kommunikationen von den Schnittstellen **30A–30C** und die Tx-Schnittstellen **28A–28C** verwalten die Übertragung von Kommunikationen auf den Schnittstellen **30A–30C**.

[0037] Jede der für kohärente Kommunikationen benützten Schnittstellen **30A–30C** ist für ein Übertragen und Empfangen von Kohärenzbefehlen geeignet. Insbesondere können in dem Ausführungsbeispiel gemäß [Fig. 1](#) jene Schnittstellen **30A–30C** dazu bestimmt sein, Kohärenzbefehle zu und von dem Knoten **10** von anderen Knoten zu empfangen/zu übertragen. Zusätzlich können andere Befehlstypen übertragen werden. In einer Ausführungsform kann jede Schnittstelle **30A–30C** eine HyperTransport™ (HT)-Schnittstelle sein, einschließlich einer Erweiterung zu der HT Schnittstelle, um Kohärenzbefehle (HTcc) einzuschließen. Zusätzlich kann in einigen Ausführungsformen eine Erweiterung zu der HyperTransport-Schnittstelle unterstützt werden, um Paketdaten zu übertragen (Packet Over HyperTransport, oder PoHT). Wie hierin verwendet, schließen Kohärenzbefehle beliebige Kommunikationen zwischen Knoten ein, welche verwendet werden, um Kohärenz zwischen Knoten aufrechtzuerhalten. Die Befehle können Lese- oder Schreibanforderungen, welche von einem Knoten initiiert werden, um einen zu einem anderen Knoten gehörenden Cacheblock abzurufen oder zu aktualisieren, Prüfbefehle, um gespeicherte Kopien von Cacheblöcken in entfernten Knoten für ungültig zu erklären (und möglicherweise eine modifizierte Kopie des Cacheblockes an den Heimatknoten zurückzusenden), Antworten auf Prüfbefehle, Fill Befehle, welche Daten übertragen, etc. einschließen.

[0038] In einigen Ausführungsformen können eine oder mehrere der Schnittstellenschaltungen **20A–20C** nicht für Kohärenzverwaltung verwendet

werden und können als Paketschnittstellen definiert sein. Derartige Schnittstellen **30A–30C** können HT Schnittstellen sein. Alternativ können derartige Schnittstellen **30A–30C** System-Paket-Schnittstellen (SPI = System Packet Interface) gemäß jeden Levels der SPI Spezifikation, welche von dem Optical Inter-networking Forum festgelegt ist (z.B. Level 3, Level 4 oder Level 5). In einem besonderen Ausführungsbeispiel können die Schnittstellen SPI-4 Phase 2 Schnittstellen sein. In dem erläuterten Ausführungsbeispiel kann jede Schnittstellenschaltung **20A–20C** eingerichtet sein, entweder auf der SPI-4 Schnittstelle oder der HT Schnittstelle zu kommunizieren. Jede Schnittstellenschaltung **20A–20C** kann individuell programmierbar sein was verschiedene Kombinationen der HT und SPI-4 Schnittstellen als Schnittstellen **30A–30C** erlaubt. Das Programmieren kann in beliebiger Weise durchgeführt werden (z.B. Abtasten bestimmter Signale während eines Zurücksetzens, Verschieben von Werten in einem Konfigurationsregister (nicht gezeigt) während eines Zurücksetzens, Programmieren der Schnittstellen mit Konfigurationsraum-Befehlen nach einem Zurücksetzen, Anschlüsse, welche zusammengebunden oder extern angebunden sind, um die gewünschte Programmierung anzuzeigen, etc.). Andere Ausführungsformen können jede Schnittstelle einsetzen, welche geeignet ist, Paketdaten zu übertragen (z.B. die medienunabhängige Schnittstelle (MII = Media Independent Interface) oder die Gigabit MII (GMII) Schnittstellen, X.25, Frame Relay, Asynchronous Transfer Mode (ATM), etc.). Die Paketschnittstellen können Paketdaten direkt übertragen (z.B. Übertragen der Paketdaten mit verschiedenen Steuerinformationen, welche den Anfang des Pakets, das Ende des Pakets, etc. anzeigen) oder indirekt (z.B. Übertragen der Paketdaten als Nutzinformation eines Befehls, wie PollIT).

[0039] In Ausführungsformen, welche auch Paketdatenverkehr unterstützen, kann der Knoten **10** auch eine Paket-Direktspeicherzugriffs (DMA = Direct Memory Access)-Schaltung einschließen, welche eingerichtet ist, Pakete zu und von dem Speicher **24** im Auftrag der Schnittstellenschaltungen **20A–20C** zu übertragen. Der Schalter **18** kann verwendet werden, um Paketdaten von den Schnittstellenschaltungen **20A–20C** zu der Paket-DMA-Schaltung und von der Paket-DMA-Schaltung zu den Schnittstellenschaltungen **20A–20C** zu übertragen. Zusätzlich können in einigen Ausführungsformen Pakete von einer Rx-Schaltung **26A–26C** an eine Tx-Schaltung **28A–28C** durch den Schalter **18** geleitet werden.

[0040] Die Prozessoren **12A–12N** können auf jede Befehlssatzarchitektur ausgelegt sein und können Programme ausführen, welche an jene Befehlssatzarchitektur geschrieben sind. Beispielhafte Befehlssatzarchitekturen können die MIPS Befehlssatzarchitektur (einschließlich der MIPS-3D und MIPS MDNX anwendungsspezifischen Erweiterungen), der von

Intel Corp. entwickelten IA-32 oder IA-64 Befehlssatzarchitekturen, der Power PC Befehlssatzarchitektur, der Alpha Befehlssatzarchitektur, der ARM Befehlssatzarchitektur oder jeder anderen Befehlssatzarchitektur. Der Knoten **10** kann jede Zahl von Prozessoren einschließen (z.B. so wenige wie einen Prozessor, zwei Prozessoren, vier Prozessoren, etc.).

[0041] Das L2 Cache **36** kann von jedem Typus und Cachespeicherkapazität sein und jede Ordnung einsetzen (z.B. satzadressiert, direkt abgebildet, völlig inhaltsadressiert, etc.). In einem Ausführungsbeispiel kann das L2 Cache **36** ein 8 Wege, satzadressiertes, 1 MB Cache sein. Das L2-Cache **36** wird hierin als L2 bezeichnet, weil die Prozessoren **12A–12N** interne (L1) Caches einschließen können. In anderen Ausführungsformen kann das L2 Cache **36** ein L1 Cache, ein L3 Cache oder irgendein anderer gewünschter Level sein.

[0042] Die Speichersteuerung **14** ist eingerichtet, auf den Speicher **24** in Erwiderung auf Lese- und Schreibvorgänge zuzugreifen, welche auf der Verbindung **22** empfangen werden. Die Speichersteuerung **14** kann ein Treffersignal von dem L2 Cache empfangen und, wenn ein Treffer in dem L2 Cache für einen bestimmten Lese/Schreibvorgang entdeckt wird, kann die Speichersteuerung **14** nicht auf jenen Vorgang reagieren. Die Speichersteuerung **14** kann eingerichtet sein, auf jede der Vielzahl von Speichertypen zuzugreifen. Z.B. kann die Speichersteuerung **14** für einen synchronen, dynamischen Direktzugriffsspeicher (SDRAM = Synchronous dynamic random access memory) und insbesondere auf ein SDRAM mit doppelter Datenübertragungsrate (DDR = double data rate). Alternativ kann die Speichersteuerung **16** ausgelegt sein auf ein DRAM, DDR synchronous graphics RAM (SGRAM), DDR fast cycle RAM (FCRA), DDR-II SDRAM, Rambus DRAM (RDRAM), SRAM, oder jede andere geeignete Speichervorrichtung oder Kombinationen von den oben genannten Speichervorrichtungen.

[0043] Die Verbindung **22** kann jede Form von Kommunikationsmedium zwischen den an die Verbindung gekoppelten Vorrichtungen darstellen. Z.B. kann in verschiedenen Ausführungsformen die Verbindung **22** mehrfach benutzte Busse, Kreuzschienen(crossbar)-Verbindungen, Punkt-zu-Punkt-Verbindungen in einem Ring, Stern oder in irgendeiner anderen Topologie, Maschen, Würfel, etc. Die Verbindung **22** kann in einigen Ausführungsformen auch einen Speicher einschließen. In einer besonderen Ausführungsform kann die Verbindung **22** einen Bus aufweisen. In einem Ausführungsbeispiel kann der Bus einen geteilten Abwicklungsbuss darstellen (d.h., er besitzt getrennte Adress- und Datenphasen). Die Datenphasen verschiedener Vorgänge auf dem Bus können unabhängig von den Adressphasen ablaufen. Der

Bus kann auch Kohärenz unterstützen und kann folglich eine Antwortphase einschließen, um Kohärenz-Antwortinformation zu übertragen. Der Bus kann in einer Ausführungsform ein geteiltes Entscheidungssystem einsetzen. In einem Ausführungsbeispiel kann der Bus in einer Pipeline eingebunden sein. Der Bus kann jedes geeignete Signalübertragungsverfahren einsetzen. Z.B. kann in einer Ausführungsform eine Differentialsignalübertragung für eine Hochgeschwindigkeitssignalübertragung verwendet werden. Andere Ausführungsformen können jedes andere Signalübertragungsverfahren einsetzen (z.B. TTL, CMOS, GTL, HSTL, etc.). Andere Ausführungsformen können nicht-aufgeteilte Abwicklungsbusse, welche mit einer einzelnen Entscheidung für Adresse und Daten vermittelt werden, und/oder einen geteilten Abwicklungsbus einsetzen, in welchem der Datenbus nicht explizit vermittelt wird. Gemäß einer Ausbildungsform kann entweder ein zentrales Entscheidungsschema oder ein verteiltes Entscheidungsschema verwendet werden. Des weiteren kann, falls gewünscht, der Bus nicht in einer Pipeline eingebunden sein.

[0044] Verschiedene Ausführungsformen des Knotens **10** können eine zusätzliche Schaltungen einschließen, welche nicht in [Fig. 1](#) abgebildet ist. Z.B. kann der Knoten **10** verschiedene E/A-Vorrichtungen und/oder Schnittstellen aufweisen. Beispielhafte E/A-Vorrichtungen können ein oder mehrere PCI Schnittstellen, eine oder mehrere serielle Schnittstelle(n), Personal Computer Memory Card International Association (PCMCIA) Schnittstellen, etc. aufweisen. Derartige Schnittstellen können direkt mit der Verbindung **22** verbunden sein oder können über eine oder mehrere E/A Brückenschaltung(en) verbunden sein.

[0045] In einer Ausführungsform kann der Knoten **10** (und insbesondere die Prozessoren **12A–12N**, die Speichersteuerung **14**, das L2 Cache **36**, die Schnittstellenschaltungen **20A–20C**, die Speicherbrücke **32**, einschließlich des entfernten Leistungsverzeichnisses **34**, der Schalter **18**, das Konfigurationsregister **38** und die Verbindung **22**) auf einer einzelnen integrierten Schaltung als ein System auf einer Chipanordnung integriert sein. Die oben erwähnten zusätzlichen Schaltungen können auch integriert sein. Alternativ können andere Ausbildungsformen eine oder mehrere der Vorrichtungen als getrennte integrierte Schaltungen implementieren. In einer weiteren Anordnung kann der Speicher **24** auch integriert sein. Alternativ kann eine oder mehrere der Komponenten als getrennte integrierte Schaltungen implementiert sein, oder alle Komponenten können, wenn gewünscht, getrennte integrierte Schaltungen darstellen. Jeder Integrationsgrad kann verwendet werden.

[0046] Es wird erwähnt, dass, während in [Fig. 1](#) drei Schnittstellenschaltungen **20A–20C** abgebildet sind, in verschiedenen Ausführungsformen eine oder meh-

rere Schnittstellenschaltungen implementiert sein können. Wie hierin verwendet, schließt eine Schnittstellenschaltung jede Schaltung ein, welche eingerichtet ist, auf einer Schnittstelle gemäß dem für die Schnittstelle festgelegten Protokoll zu kommunizieren. Die Schnittstellenschaltung kann eine Empfangsschaltung einschließen, welche eingerichtet ist, Kommunikationen auf der Schnittstelle zu empfangen und die empfangenen Kommunikationen an andere Schaltungen zu übertragen, welche sich innerhalb des Systems befinden, welches die Schnittstellenschaltung einschließt. Die Schnittstellenschaltung kann auch eine Übertragungsschaltung einschließen, welche eingerichtet ist, Kommunikationen von den anderen Schaltungen zu empfangen, welche sich innerhalb des Systems befinden und eingerichtet sind, diese Kommunikationen auf der Schnittstelle zu übertragen.

[0047] Übergehend zu [Fig. 2](#) ist ein Blockdiagramm eines Ausführungsbeispiels eines kohärenten Systems **40** gezeigt. In dem Ausführungsbeispiel gemäß [Fig. 2](#) schließt das System **40** eine Vielzahl von Knoten **10A–10D** ein. Jeder der Knoten **10A–10D** kann eine Implantierung des Knotens **10** sein. In dem Ausführungsbeispiel gemäß [Fig. 2](#) ist der Knoten **10A** mit dem Knoten **10B** verbunden, welcher weiter mit dem Knoten **10C** verbunden ist, welcher wiederum weiter mit dem Knoten **10D** verbunden ist. Insbesondere ist die Schnittstellenschaltung des Knotens **10A** mit einer Schnittstelle verbunden, mit welcher die Schnittstellenschaltung **20A** des Knotens **10B** verbunden ist. Die Schnittstellenschaltung **20B** des Knotens **10B** ist mit einer Schnittstelle verbunden, welche mit der Schnittstellenschaltung **20B** des Knotens **10C** verbunden ist. Die Schnittstellenschaltung **20A** des Knotens **10C** ist mit einer Schnittstelle verbunden, welche mit dem Knoten **10D** verbunden ist.

[0048] Die Knoten **10A–10D** können die Schnittstellen verwenden, um kohärent zwischen den Knoten zu kommunizieren. D.h., von den Speicherbrücken **32** in den Knoten **10A–10D** erzeugte Kohärenzbefehle können über die Schnittstellen zwischen den Knoten übertragen werden. Zusätzlich können in einigen Ausführungsformen andere Befehlstypen übertragen werden. Z.B. können in einer Ausführungsform die Schnittstellen HT Schnittstellen sein, wie zuvor in Bezug auf [Fig. 1](#) erwähnt. Folglich können HT Befehle, HTcc Befehle und/oder POHT Befehle in verschiedenen Ausführungsformen übertragen werden. Die HT Schnittstelle kann ein Paar Punkt-zu-Punkt-Verbindungen zwischen jeder Schnittstelle aufweisen. Z.B. kann die Schnittstelle zwischen der Schnittstellenschaltung des Knotens **10A** und der Schnittstellenschaltung **20A** des Knotens **10B** ein Paar Punkt-zu-Punkt-Verbindungen aufweisen, wobei eine Verbindung von der Schnittstellenschaltung des Knotens **10A** zu der Schnittstellenschaltung **20A** des Knotens **10B** geführt ist und die andere Verbindung

durch die Schnittstellenschaltung **20A** des Knotens **10B** zu der Schnittstellenschaltung **20C** des Knotens **10A** geführt ist. Andere Schnittstellen können in ähnlicher Weise Punkt-zu-Punkt-Verbindungen sein. In anderen Ausführungsformen kann jeder Typ von Schnittstellen verwendet werden.

[0049] In dem Ausführungsbeispiel von [Fig. 2](#) sind die Knoten **10A–10D** in einer Prioritätsverkettung verbunden. Diese Verbindung kann für Standard HT Befehle verwendet werden. In anderen Ausführungsbeispielen kann jede Verbindung für HTcc Befehle verwendet werden. Z.B. können in einem Ausführungsbeispiel die Knoten **10A–10D** jeweils direkt mit jedem anderen Knoten verbunden sein (wobei eine der drei Schnittstellenschaltungen **20A–20C** verwendet wird, um an einer Schnittstellenschaltung jedes anderen Knotens anzukoppeln). Einige der Verbindungen können verwendet werden, um die Prioritätsverkettungssteuerung für Standard HT Befehle bereitzustellen.

[0050] Während vier Knoten **10A–10D** in dem kohärenten System **40** abgebildet sind, können andere Ausführungsformen, wie gewünscht, jede Zahl von Knoten einsetzen. Die Knoten können in jeder gewünschten Anordnung verbunden werden, wobei die Schnittstellenschaltungen **20A–20C** verwendet werden. Wenn die HT Schnittstelle verwendet wird und Standard HT Befehle in dem kohärenten System **40** verwendet werden (zusätzlich zu den HTcc Befehlen), kann die Verbindung einen Prioritätsverkettungspfad einschließen, welcher jeden Knoten erreicht.

[0051] Weitergehend zu [Fig. 3](#) ist eine Tabelle 42 gezeigt, welche einen beispielhaften Satz von Vorgängen erläutert, welche von einem Ausführungsbeispiel der Verbindung **22** unterstützt wird und eine Tabelle 44 gezeigt, welche einen beispielhaften Satz von Kohärenzbefehlen erläutert, welche von einer Ausführungsform der Schnittstellen **30** unterstützt werden. Andere Ausführungsformen, welche Untermengen, Obermengen oder alternative Befehlssätze einschließen, können verwendet werden.

[0052] Die in der Tabelle 42 erläuterten Vorgänge werden als nächstes beschrieben werden. Ein Agent in dem Knoten **10** kann einen Cacheblock (entweder entfernt oder lokal) lesen, wobei read shared (RdShd) oder read exclusive (RdExc) Vorgänge auf der Verbindung **22** verwendet werden. Der RdShd Vorgang wird verwendet, um eine mehrfach benutzte Kopie des Cacheblockes anzufordern und der RdExc Vorgang wird verwendet, um eine ausschließliche Kopie des Cacheblockes anzufordern. Wenn der RdShd Vorgang verwendet wird und kein anderer Agent während der Antwortphase des Vorgangs meldet, eine Kopie des Cacheblockes zu besitzen (ausgenommen das L2 Cache **36** und/oder die Speicher-

steuerung **14**), kann der Agent den Cacheblock in den ausschließlichen Zustand nehmen. In Erwiderung auf den RdExc Vorgang erklären andere Agenten in dem Knoten ihre Kopien des Cacheblockes (falls vorhanden) für ungültig. Zusätzlich kann ein ausschließlicher (oder modifizierter) Besitzer des Cacheblockes die Daten für den Vorgang in der Datenphase liefern. Andere Ausführungsformen können andere Mechanismen (z.B. eine Wiederholung auf der Verbindung **22**) einsetzen, um die Übertragung eines modifizierten Cacheblockes sicherzustellen.

[0053] Der Write Vorgang (Wr) und der Write Invalidate Vorgang (WrInv) kann von einem Agenten benutzt werden, um einen Cacheblock in einen Speicher zu schreiben. Der Wr Vorgang kann von einem Besitzer verwendet werden, welcher den modifizierten Zustand für den Block besitzt, da keine weiteren Kopien des Blockes für ungültig erklärt werden müssen. Der WrInv Vorgang kann von einem Agenten verwendet werden, welcher kein ausschließliches Eigentumsrecht an dem Block besitzt (der Agent kann sogar den ungültigen Zustand für den Block besitzen). Der WrInv Vorgang veranlasst andere Agenten, alle Kopien des Blockes für ungültig zu erklären, einschließlich der modifizierten Kopien. Der WrInv Vorgang kann von einem Agenten verwendet werden, welcher den gesamten Cacheblock beschreibt. Z.B. kann eine DMA, welche den gesamten Cacheblock mit neuen Daten beschreibt, den Vorgang verwenden, um einen Lesevorgang zu vermeiden, welchem ein Schreibvorgang nachfolgt.

[0054] Die RdKill und RdInv Vorgänge können von der Speicherbrücke **32** in Erwiderung auf die durch den Knoten **10** von anderen Knoten empfangenen Prüfbefehl verwendet werden. Die RdKill und RdInv Vorgänge veranlassen den Initiator (die Speicherbrücke **32**), exklusives Zugangsrecht an dem Cacheblock zu erwerben und veranlassen jeden Cacheagenten ihre Kopien für ungültig erklären (Übertragen von Daten an den Initiator in ähnlicher Weise wie die RdShd und RdExc Vorgänge). In einer Ausführungsform löscht der RdKill Vorgang eine Reservierung, welche durch die mit der Ladung verbundenen Anweisung in dem MIPS Anwendungssatz erstellt wurde, während der RdInv Vorgang dies nicht macht. In anderen Ausführungsformen kann ein einzelner Vorgang für die Prüfbefehle verwendet werden. In wiederum anderen Ausführungsformen kann ein von einem Prüfbefehl erzeugter Vorgang vorhanden sein, welcher die Agentenkopien des Cacheblockes für ungültig erklärt (in ähnlicher Weise wie die RdKill und RdInv Vorgänge) und ein weiterer, von einem Prüfbefehl erzeugter Vorgang, welcher den Agenten erlaubt, mehrfach benutzte Kopien des Cacheblockes zu behalten.

[0055] Der WrFlush Vorgang ist ein Schreibvorgang, welcher von einem Agenten veranlasst werden

kann, und ein weiterer Agent kann eine exklusive oder modifizierte Kopie des Blockes besitzen. Der andere Agent liefert die Daten für den WrFlush Vorgang oder der veranlassende Agent liefert die Daten, wenn kein anderer Agent eine exklusive oder modifizierte Kopie des Blockes besitzt. Der WrFlush Vorgang kann in einem unten beschriebenen Ausführungsbeispiel verwendet werden, in welchem das L2 Cache **36** den Knotenzustand für den Knoten **10** beibehält, aber andere Agenten können den Cacheblock auch in einem modifizierten Zustand besitzen. Das L2 Cache **36** kann den WrFlush Befehl verwenden, um einen entfernten Cacheblock auszusondern, welcher in einem Cache eines anderen Agenten in dem Knoten **10** modifiziert werden kann.

[0056] Der Nop Vorgang ist ein „keine-Operation“ Vorgang. Der Nop kann verwendet werden, wenn einem Agenten die Verwendung der Verbindung **22** erlaubt ist (z.B. der Adressbus, in Ausführungsformen, in welchen die Verbindung **22** ein geteilter Transaktionsbus ist) und der Agent ermittelt, dass er keinen Vorgang mehr auf der Verbindung **22** durchführen muss.

[0057] Die in der Tabelle 44 dargestellten Befehle sollen als nächstes beschrieben werden. In der Tabelle 44 ist sowohl der Befehl, als auch der virtuelle Kanal dargestellt, in welchem sich der Befehl sich auf den Schnittstellen **30** fortbewegt. In dem erläuterten Ausführungsbeispiel können die virtuellen Kanäle einschließen: Den virtuellen Kanal für kohärentes Lesen (CRd = coherent read); den virtuellen Kanal für Prüfbefehle (Probe); den virtuellen Kanal für Bestätigungen (Ack = acknowledge); den virtuellen Kanal für kohärentes Füllen (Cfill = Coherent Fill). Die virtuellen CRd, Probe, Ack, und Cfill Kanäle sind für die HTcc Befehle festgelegt. Es kann zusätzliche virtuelle Kanäle für die Standard HT Befehle geben (z.B. ein virtueller Kanal für nichtabgesendete Befehle (NPC = Non-Posted Command), ein virtueller Kanal für abgesendete Befehle (PC = Posted Command), und ein virtueller Kanal für Erwidierungen (RSP = Response)).

[0058] Die cRdShd oder cRdExc Befehle können von der Speicherbrücke **32** in Erwiderung auf einen RdShd oder RdExc Vorgang auf der Verbindung **22** jeweils ausgegeben werden, um einen entfernten, nicht in dem Knoten gespeicherten Cacheblock zu lesen (oder, für den Fall des RdExc, kann der Block in dem Knoten gespeichert sein, aber nicht in dem mehrfach benutzten Zustand). Wenn der Cacheblock in dem Knoten gespeichert ist (mit exklusivem Besitzrecht, für den Fall des RdExc Vorgangs), wird das Lesen auf der Verbindung **22** vollendet, ohne irgendeine Kohärenzbefehl-Übertragung durch die Speicherbrücke **32**.

[0059] Die Flush und Kill Befehle sind in diesem

Ausführungsbeispiel Prüfbefehle. Die Speicherbrücke **32** an dem Heimatknoten eines Cacheblockes kann in Erwiderung auf einen cRdSd oder cRdExc Befehl Prüfbefehle ausgeben. Die Speicherbrücke **32** an dem Heimatknoten des Cacheblockes kann auch in Erwiderung auf einen Vorgang für einen lokalen Cacheblock einen Prüfbefehl ausgeben, wenn ein oder mehrere entfernte Knoten eine Kopie des Cacheblockes besitzen. Der Flush-Befehl wird verwendet, um anzufordern, dass ein entfernter modifizierter Besitzer eines Cacheblockes den Cacheblock an den Heimatknoten zurückschickt (und den Cacheblock in dem entfernten modifizierten Besitzer für ungültig erklärt). Der Kill-Befehl wird verwendet, um anzufordern, dass ein entfernter Besitzer den Cacheblock für ungültig erklärt. In anderen Ausführungsformen können zusätzliche Prüfbefehle für andere Zustandsänderungs-Aufforderungen unterstützt werden (z.B. dass entfernten Besitzern erlaubt wird, eine mehrfach benutzte Kopie des Cacheblockes zu behalten).

[0060] Die Prüfbefehle werden erwidert (nachdem die von den Prüfbefehlen angeforderten Zustandsänderungen bewirkt worden sind), indem entweder die Kill_Ack oder WB Befehle verwendet werden. Der Kill_Ack Befehl ist eine Rückmeldung, dass ein Kill Befehl von einem empfangenden Knoten bearbeitet worden ist. Der WB Befehl ist ein Zurückschreiben des Cacheblockes und wird in Erwiderung auf den Flush-Befehl übertragen. Der WB Befehl kann auch von einem Knoten verwendet werden, um einen entfernten Cacheblock zurückzuschreiben, der gerade aus dem Knoten ausgesondert wird. Der Fill Befehl ist der Befehl, um Daten von einem entfernten Knoten, welcher einen Lesebefehl gesendet hat (cRdExc oder cRdShd), an den Heimatknoten, zu übertragen. Der Fillbefehl wird von der Speicherbrücke **32** in dem Heimatknoten ausgegeben, nachdem die Prüfbefehle (wenn überhaupt) für einen Cacheblock vollendet worden sind.

[0061] Übergehend auf [Fig. 4](#) ist ein Blockdiagramm gezeigt, welches ein Ausführungsbeispiel eines Adressplatzes erläutert, welcher von einem Ausführungsbeispiel des Knotens **10** implementiert worden ist. Die in [Fig. 4](#) gezeigten Adressen sind als Hexadezimalzahlen dargestellt, wobei ein Unterstrich ("_") Gruppen von vier Ziffern trennt. Folglich sind in dem in [Fig. 4](#) abgebildeten Ausführungsbeispiel 40 Adressbits unterstützt. In anderen Ausführungsformen können mehr oder weniger Adressbits unterstützt werden.

[0062] In dem Ausführungsbeispiel von [Fig. 4](#) wird der Adressplatz zwischen 00_0000_0000 und 0F_FFFF_FFFF als lokaler Adressplatz behandelt. Durch Agenten in dem lokalen Adressplatz erzeugte Vorgänge erzeugen nicht Kohärenzbefehle an andere Knoten, obwohl Kohärenz innerhalb des Knotens **10** für diese Adressen durchgesetzt werden kann.

D.h., der lokale Adressplatz wird nicht als kohärent mit anderen Knoten gehalten. Verschiedene Teile des lokalen Adressplatzes können nach Wunsch speicherkonform mit E/A Vorrichtungen, HT, etc. sein.

[0063] Der Adressplatz zwischen 40_0000_0000 und EF_FFFF_FFFF ist der entfernte kohärente Platz **48**. D.h., der Adressplatz zwischen 40_0000_0000 und EF_FFFF_FFFF wird als kohärent zwischen den Knoten gehalten. Jedem Knoten ist ein Teil des entfernten kohärenten Platzes zugeordnet und jener Knoten ist der Heimatknoten für den Teil. Wie in [Fig. 1](#) gezeigt, ist jeder Knoten mit einer Knotennummer programmierbar. Die Knotennummer ist in diesem Ausführungsbeispiel gleich groß wie das aussagekräftigste Halbbyte (4 Bits) der Adressen, für welche jener Knoten der Heimatknoten ist. Folglich können in dem gezeigten Ausführungsbeispiel die Knotennummern von 4 bis E reichen. Andere Ausführungsformen können nach Wunsch mehr oder weniger Knotennummern unterstützen. In dem erläuterten Ausführungsbeispiel ist jedem Knoten ein 64 Giga-byte (GB) Teil des Speicherplatzes zugeteilt, für welchen er der Heimatknoten ist. Die Größe des zu jedem Knoten zugeteilten Teiles kann in anderen Ausführungsformen variiert werden (z.B. basierend auf der Adressgröße oder anderen Faktoren).

[0064] Für einen bestimmten Knoten gibt es ein Aliasing zwischen dem entfernten kohärenten Platz, für welchen jener Knoten der Heimatknoten ist, und dem lokalen Adressplatz jenes Knotens. D.h., entsprechende Adressen in dem lokalen Adressplatz und der Teil des entfernten kohärenten Platzes, für welchen der Knoten der Heimatknoten ist, greifen auf die gleichen Speicherplätze in dem Speicher **24** des Knotens zu (oder sind speicherkonform zu den gleichen E/A Vorrichtungen oder Schnittstellen, etc.). Z.B. "aliases" der Knoten mit der Knotennummer **5** jeweils den Adressplatz 50_0000_0000 bis einschließlich 5F_FFFF_FFFF zu 00_0000_0000 bis einschließlich OF_FFFF_FFFF (Pfeil **46**). Knotenübergreifende, kohärente Zugriffe auf den Speicher **24** an dem Knoten **10** verwenden den Knoten-nummerierten Adressplatz (z.B. 50_0000_0000 bis 5F_FFFF_FFFF, wenn die in Knoten **10** programmierte Knotennummer **5** lautet), um auf Cacheblöcke in dem Speicher **24** zuzugreifen. D.h. Agenten in anderen Knoten und Agenten innerhalb des Knotens, welche kohärent auf Cacheblöcke in dem Speicher zugreifen, verwenden den entfernten kohärenten Platz, während ein Zugriff in dem lokalen Adressplatz nicht kohärent mit anderen Knoten gehalten wird (sogar wenn auf den gleichen Cacheblock zugegriffen werden kann). Folglich werden in diesem Ausführungsbeispiel Adressen "aliased", aber nicht kohärent gehalten. In anderen Ausführungsformen können die Adressen in dem entfernten kohärenten Platz und die entsprechenden Adressen in dem lokalen Adressplatz kohärent ge-

halten werden.

[0065] Ein Cacheblock wird in einem Knoten als lokal bezeichnet, wenn der Cacheblock Teil des zu dem Knoten zugeordneten Speichers ist (wie oben erwähnt). Demnach kann der Cacheblock lokal sein, wenn auf ihn von dem lokalen Adressplatz oder dem entfernten kohärenten Platz zugegriffen wird, solange die Adresse in dem Bereich ist, für welchen der Knoten der Heimatknoten ist. In ähnlicher Weise kann ein Vorgang auf der Verbindung **22**, welcher auf einen lokalen Cacheblock zugreift, als ein lokaler Vorgang oder lokaler Zugriff bezeichnet werden. Ein Vorgang auf der Verbindung **22**, welcher auf einen entfernten Cacheblock zugreift (über den entfernten kohärenten Adressplatz außerhalb des Teiles, für welchen der Knoten der Heimatknoten ist) kann als ein entfernter Vorgang oder als ein entfernter Zugriff bezeichnet werden.

[0066] Der Adressplatz zwischen 10_0000_0000 und 3F_FFFF_FFFF kann für zusätzliche HT Vorgänge (z.B. Standard HT Vorgänge) in dem erläuterten Ausführungsbeispiel verwendet werden. Zusätzlich kann der Adressplatz zwischen F0_0000_0000 und FF_FFFF_FFFF in dem erläuterten Ausführungsbeispiel reserviert sein.

[0067] Es wird darauf hingewiesen, dass, während das aussagekräftigste Halbbyte der Adresse bestimmt, auf welchen Knoten zugegriffen wird, andere Ausführungsformen jeden anderen Teil der Adresse verwenden können, um den Knoten zu identifizieren. Darüber hinaus kann in anderen Ausführungsformen andere Information in dem Vorgang verwendet werden, um entfernte gegenüber lokalen Vorgängen zu identifizieren (z.B. Befehlstyp, in dem Vorgang übermittelte Steuerinformationen, etc.).

[0068] Übergehend zu [Fig. 5](#), ist ein Entscheidungsbaum für einen Lesevorgang auf einer Speicherplatzadresse auf der Verbindung **22** des Knotens **10** für ein Ausführungsbeispiel gezeigt. Der Entscheidungsbaum kann den Arbeitsablauf des Knotens **10** für den Lesevorgang für unterschiedliche Vorgangsbedingungen, den Zustand des Cacheblocks, auf welchem durch den Vorgang zugegriffen wird, etc. erläutern. Der Lesevorgang kann in einem Ausführungsbeispiel die in Tabelle 42 aus [Fig. 3](#) gezeigten RdShd, RdExc, RdKill und RdInv Vorgänge einschließen. Jeder Punkt auf den Linien innerhalb des Entscheidungsbaums stellt einen Abzweigungspunkt eines oder mehrerer Glieder des Baumes dar, welche mit den zugehörigen Bedingungen beschriftet sind. Wo mehrere Glieder von einem Punkt ausgehen, impliziert das Wählen eines Gliedes, dass die Bedingungen für die anderen Glieder nicht erfüllt sind. In [Fig. 5](#) wird das Ausrufezeichen ("!") verwendet, um ein logisches NOT anzuzeigen. Nicht in [Fig. 5](#) gezeigt ist der Zustandsübergang, welcher von jedem

kohärenten Agenten unternommen wird, welcher eine Kopie des Cacheblocks für den Lesevorgang in einen Cachespeicher aufnimmt. Wenn der Lesevorgang ein RdShd ist, kann der kohärente Agent eine Kopie des Cacheblockes in dem mehrfach benutzten Zustand behalten. Sonst erklärt der kohärente Agent seine Kopie des Cacheblockes für ungültig.

[0069] Der Vorgang kann, wie oben erwähnt, entweder lokal oder entfernt sein. Für lokale Vorgänge, wenn der Vorgang nicht in einen Cachespeicher aufgenommen werden kann, wird ein Lesen aus dem Speicher **24** durchgeführt (Bezugszeichen **50**). In einem Ausführungsbeispiel kann der Vorgang ein Anzeigen einschließen, ob der Vorgang in einen Cachespeicher aufgenommen werden kann oder nicht. Wenn der Vorgang nicht in einen Cachespeicher aufgenommen werden kann, wird er in dem vorliegenden Ausführungsbeispiel als ein nicht-kohärenter Vorgang behandelt.

[0070] Falls der lokale Vorgang in einem Cachespeicher aufgenommen werden kann, ist der Arbeitsablauf des Knotens **10** abhängig von der während der Antwortphase des Vorgangs gelieferten Antwort. In einer Ausführungsform antwortet jeder kohärente Agent mit dem Zustand des Cacheblockes in jenem Agent. Z.B. kann jeder kohärente Agent ein zugeordnetes, mehrfach benutztes (SHD = Shared) und ein ausschließliches (EXC = Exclusive) Signal aufweisen. Der Agent kann einen ungültigen Zustand anzeigen, indem er weder das SHD noch das EXC Signal bestätigt. Der Agent kann einen mehrfach benutzten Zustand anzeigen, indem er das SHD Signal bestätigt und das EXC Signal nicht bestätigt. Der Agent kann einen ausschließlichen Zustand (oder modifizierten Zustand) anzeigen, indem er das EXC Signal bestätigt und das SHD Signal nicht bestätigt. In diesem Ausführungsbeispiel können die ausschließlichen und modifizierten Zustände in der Antwortphase als dieselben behandelt werden, und der ausschließliche/modifizierte Besitzer kann die Daten liefern. Der ausschließliche/modifizierte Besitzer kann, in Übereinstimmung mit den Daten, eine Anzeige liefern, ob der Zustand ein ausschließlicher oder ein modifizierter Zustand ist. Während jeder Agent in diesem Ausführungsbeispiel seine eigenen SHD und EXC Signale aufweisen kann (und der verursachende Agent kann die Signale von jedem anderen Agenten empfangen), können in anderen Ausführungsformen ein mehrfach benutztes SHD und EXC Signal von allen Agenten benutzt werden.

[0071] Wenn sowohl die SHD, als auch die EXC Antworten für den lokalen Vorgang empfangen werden, ist ein Fehler aufgetreten (Bezugszeichen **52**). In einem Ausführungsbeispiel kann die Speichersteuerung eine Anzeige eines schweren Fehlers für den Lesevorgang zurücksenden. Falls die Antwort ausschließlich lautet (SHD nicht bestätigt, EXC be-

stätigt) liefert der ausschließliche Besitzer die Daten für den Lesevorgang auf der Verbindung **20** (Bezugszeichen **54**). Falls der ausschließliche Benutzer die Speicherbrücke **32** ist (wie in dem entfernten Leistungsverzeichnis **34** aufgenommen), dann besitzt ein entfernter Knoten den Cacheblock in dem modifizierten Zustand. Die Speicherbrücke **32** gibt einen Prüfbefehl (Flush Befehl) aus, um den Cacheblock von jenem entfernten Knoten zurückzubekommen. Die Speicherbrücke **32** kann den von dem entfernten Knoten zurückgesendeten Cacheblock als die Daten für das Lesen auf der Verbindung **22** liefern.

[0072] Falls die Antwort mehrfach benutzt lautet (SHD bestätigt EXC nicht bestätigt), der lokale Vorgang ein RdExc ist und die Speicherbrücke **32** einer der Agenten ist, die mehrfach benutzt melden, dann kann mindestens ein entfernter Knoten eine mehrfach benutzte Kopie des Cacheblockes besitzen. Die Speicherbrücke **32** kann einen Prüfbefehl (KILL Befehl) veranlassen, um die mehrfach benutzten Kopien des Cacheblockes in dem/den entfernten Knoten (Bezugszeichen **56**) für ungültig zu erklären. In einem Ausführungsbeispiel können für diesen Fall die Daten aus einem Speicher (oder dem L2 Cache **36**) gelesen werden, aber die Datenübertragung kann verzögert werden, bis der/die entfernten Knoten den Prüfbefehl bestätigt haben. Die Speicherbrücke **32** kann der Speichersteuerung **14**/dem L2 Cache **36** anzeigen, wann die Bestätigungen empfangen worden sind. In einem Ausführungsbeispiel kann jeder Vorgang eine Vorgangskennung auf der Verbindung **22** aufweisen. Die Speicherbrücke **32** kann die Vorgangskennung des RdExc Vorgangs an die Speichersteuerung **14**/das L2 Cache **36** senden, um anzuzeigen, dass die Daten übertragen werden können.

[0073] Wenn die Antwort mehrfach benutzt lautet, der lokale Vorgang ein RdExc ist und die mehrfach benutzenden Agenten lokale Agenten sind (d.h., die Speicherbrücke **32** meldet nicht mehrfach benutzt), dann kann das L2 Cache **36** oder die Speichersteuerung **14** die Daten liefern, abhängig davon, ob ein L2 Treffer für den Cacheblock (Bezugszeichen **58**) vorliegt oder nicht. In ähnlicher Weise kann, wenn die Antwort mehrfach benutzt lautet und der Vorgang kein RdExc ist, das L2 Cache **36** oder die Speichersteuerung **14** die Daten liefern, abhängig davon, ob es einen L2 Treffer für den Cacheblock vorliegt oder nicht.

[0074] Wenn der Vorgang entfernt ist und nicht in einen Cachespeicher aufgenommen werden kann, dann kann die Speicherbrücke **32** einen nicht-kohärenten Lesebefehl auf den Schnittstellen **30** erzeugen, um die Daten zu lesen. Z.B. kann ein Standard HT Lesebefehl verwendet werden (Bezugszeichen **60**). Wenn der entfernte Vorgang in einem Cachespeicher abgelegt werden kann und die Antwort auf

der Verbindung 22 ausschließlich lautet, dann liefert der ausschließliche Besitzer die Daten für das Lesen (Bezugszeichen 62). Falls der entfernte Vorgang in einen Cachespeicher abgelegt werden kann, die Antwort nicht ausschließlich lautet, der Cacheblock ein L2 Cachetreffer ist und der Vorgang entweder ein RdShd ist oder der Vorgang ein RdExc ist und das L2 Cache den Block in dem modifizierten Zustand besitzt, dann liefert das L2 Cache 36 die Daten für das Lesen (Bezugszeichen 64). Sonst veranlasst die Speicherbrücke 32 einen entsprechenden Lesebefehl an den Heimatknoten des Cacheblockes (Bezugszeichen 66).

[0075] Übergehend zu [Fig. 6](#), ist ein Entscheidungsbaum für einen Schreibvorgang an eine Speicherplatzadresse auf der Verbindung 22 eines Knotens 10 für ein Ausführungsbeispiel gezeigt. Der Entscheidungsbaum kann den Arbeitsvorgang des Knotens für den Schreibvorgang für verschiedene Vorgangsbedingungen, den Zustand des Cacheblocks, auf welchen von dem Vorgang zugegriffen wird, etc., erläutern. Der Schreibvorgang kann in einem Ausführungsbeispiel die Wr, Wrlnv und WrFlush Vorgänge einschließen, welche in der Tabelle 42 aus [Fig. 3](#) gezeigt sind. Jeder Punkt auf den Linien innerhalb des Entscheidungsbaums stellt einen Abzweigungspunkt eines oder mehrerer Glieder des Baumes dar, welche mit den zugehörigen Bedingungen beschriftet sind. Wo mehrere Glieder von einem Punkt ausgehen, impliziert das Ergreifen eines Gliedes auch, dass die Bedingungen für die anderen Glieder nicht erfüllt werden. In [Fig. 6](#) wird das Ausrufezeichen (!) verwendet, um ein logisches NOT anzuzeigen. Nicht in [Fig. 6](#) gezeigt sind die Zustandsübergänge, welche von jedem kohärenten Agenten durchgeführt werden, welcher eine Kopie des Cacheblockes für den Schreibvorgang in einen Cachespeicher ablegt. Der kohärente Agent erklärt seine Kopie des Cacheblockes für ungültig.

[0076] Falls der Vorgang ein lokaler Vorgang ist, und der Vorgang ein Wrlnv Vorgang ist, welcher einen Treffer in dem entfernten Leitungsverzeichnis 34 liefert (d.h. ein entfernter Knoten legt eine Kopie des Cacheblockes in einen Cachespeicher ab), führt die Speichersteuerung 14 (und das L2 Cache 36, sofern ein L2 Treffer vorliegt) mit den Schreibdaten eine Aktualisierung durch (Bezugszeichen 70). Zusätzlich kann die Speicherbrücke 32 Prüfbefehle an die von dem entfernten Leitungsverzeichnis 34 angezeigten, entfernten Knoten erzeugen. Die Aktualisierung des Speichers/L2 Caches kann verzögert werden, bis die Prüfbefehle vollendet worden sind, zu welcher Zeit die Speicherbrücke 32 die Vorgangskennung des Wrlnv Vorgangs an das L2 Cache 36/die Speichersteuerung 14 übertragen kann, um die Aktualisierung zu ermöglichen.

[0077] Wenn der lokale Vorgang nicht in einen Ca-

chespeicher abgelegt werden kann, oder wenn das L2 Cache 36 Herr über diesen Vorgang ist (d.h., das L2 Cache 36 veranlasste diesen Vorgang), dann führt die Speichersteuerung 14 mit den Daten eine Aktualisierung durch (Bezugszeichen 72). Falls der lokale Vorgang in einem Cachespeicher abgelegt werden kann, führt die Speichersteuerung 14 und/oder das L2 Cache 36 mit den Daten eine Aktualisierung durch, darauf basierend, ob ein L2 Cachetreffer vorliegt oder nicht (und in einigen Ausführungsformen basierend auf einer L2 Cache-Zuordnungsanzeige in dem Vorgang, welche dem Urheber dieses Vorgangs ermöglicht, anzuzeigen, ob das L2 Cache eine Cacheleitung für ein L2 Cachekein-Treffer zuordnet oder nicht) (Bezugszeichen 74).

[0078] Falls der Vorgang ein entfernter Vorgang ist, der Vorgang ein WrFlush Vorgang ist und die Antwort auf den Vorgang ausschließlich lautet, liefert der ausschließliche Besitzer die Daten (Bezugszeichen 76). Falls der entfernte WrFlush Vorgang zu einer nicht-ausschließlichen Antwort (mehrfach benutzt oder ungültig) führt, liefert das L2 Cache 36 die Daten des WrFlush Vorgangs. In einem Ausführungsbeispiel behält das L2 Cache 36 den Zustand des Knotens, wie in dem Heimatknoten aufgezeichnet, und das L2 Cache 36 verwendet den WrFlush Vorgang, um einen entfernten Cacheblock auszusondern, welcher in dem modifizierten Zustand in dem Knoten vorliegt. Folglich, wenn ein anderer Agent den Cacheblock in dem ausschließlichen Zustand besitzt, kann jener Agent eine aktuellere Kopie des Cacheblockes besitzen, welche an den Heimatknoten zurückgesendet werden sollte. Andernfalls liefert das L2 Cache 36 den zurückzusendenden Block an den Heimatknoten. In beiden Fällen kann die Speicherbrücke 32 den WrFlush Vorgang und die Daten erlangen und kann einen WB Befehl durchführen, um den Cacheblock an den Heimatknoten zurückzusenden.

[0079] Falls der entfernte Vorgang kein WrFlush Vorgang ist und nicht Cache-kohärent ist, empfängt die Speicherbrücke 32 den Schreibvorgang und führt einen nicht-kohärenten Wr Befehl durch (z.B. ein Standard HT write), um den Cacheblock an den Heimatknoten zu übertragen (Bezugszeichen 80). Falls der entfernte Vorgang kein WrFlush Vorgang ist, Cache-kohärent ist und einen L2 Treffer liefert, kann das L2 Cache 36 mit den Daten eine Aktualisierung durchführen (Bezugszeichen 82).

[0080] Übergehend zu [Fig. 7](#) ist ein Blockdiagramm gezeigt, welches die Arbeitsweise eines Ausführungsbeispiels der Speicherbrücke 32 in Erwiderung auf verschiedene, von den Schnittstellenschaltungen 20A–20C empfangene Kohärenzbefehle erläutert. Der empfangene Befehl ist in einem Oval dargestellt. Befehle, die von der Speicherbrücke 32 in Erwiderung auf den empfangenen Befehl (und den Zustand des betroffenen Cacheblocks, wie in dem entfernten

Leitungsverzeichnis **34** angezeigt) veranlasst werden, sind in durchgezogenen Rechtecken dargestellt. Punktierter Rechtecke sind Befehle, welche von der Speicherbrücke **32** in Erwiderung auf die übertragene Befehle in den vorhergehenden durchgezogenen Rechtecken empfangen werden. Der von dem Befehl betroffene Cacheblock ist in Klammern nach dem Befehl dargestellt.

[0081] In einem Ausführungsbeispiel kann auf das entfernte Leitungsverzeichnis **34** in Erwiderung auf einen Vorgang auf der Verbindung **22** zugegriffen werden. In einem derartigen Ausführungsbeispiel kann die Speicherbrücke **32** einen Vorgang auf der Verbindung **22** in Erwiderung auf bestimmte kohärente Befehle veranlassen, um das entfernte Leitungsverzeichnis **34** abzurufen (sowie alle Zustandsänderungen in den mit der Verbindung **22** verbundenen kohärenten Agenten zu beeinflussen, sofern zutreffend). In anderen Ausführungsformen kann die Speicherbrücke **32** eingerichtet sein, das entfernte Leitungsverzeichnis **34** vor Erzeugen eines Vorgangs auf der Verbindung **22** zu lesen und kann, falls benötigt, bedingt einen Vorgang erzeugen, basierend auf dem Zustand des entfernten Leitungsverzeichnisses **34** für den angeforderten Cacheblock. Zusätzlich kann in einem Ausführungsbeispiel das entfernte Leitungsverzeichnis **34** den entfernten Zustand für eine Untermenge der lokalen Cacheblöcke behalten, welche entfernt mehrfach benutzbar sind (z.B. eine Untermenge des Teiles des entfernten kohärenten Platzes **48**, welcher dem lokalen Knoten zugeordnet ist). Wenn ein Cacheblock von einem entfernten Knoten angefordert wird, wobei ein Kohärenzbefehl verwendet wird und es keinen Eintrag in dem entfernten Leitungsverzeichnis **34** für den Cacheblock gibt, dann kann ein Opfer-Cacheblock in dem entfernten Leitungsverzeichnis **34** wieder ersetzt werden (und Prüfbefehle können erzeugt werden, um den Opfer-Cacheblock in den entfernten Knoten für ungültig zu erklären). In anderen Ausführungsformen kann das entfernte Leitungsverzeichnis **34** eingerichtet sein, den Zustand jedes Cacheblockes in dem Teil des entfernten kohärenten Platzes **48** zu verfolgen, welcher dem lokalen Knoten zugeordnet ist. In solchen Ausführungsformen können mit den Opfer-Cacheblöcken verbundene Arbeitsvorgänge aus [Fig. 7](#) weggelassen werden.

[0082] Für einen cRdShd Befehl für einen von der Speicherbrücke **32** empfangenen Cacheblock "A" (Bezugszeichen **90**), kann die Speicherbrücke **32** einen RdShd Vorgang auf der Verbindung **22** erzeugen. Basierend auf dem entfernten Leitungsverzeichnis (RLD = Remote Line Directory)-Zustand für den Cacheblock A kann eine Zahl von Arbeitsvorgängen auftreten. Falls der RLD Zustand mehrfach benutzt oder ungültig ist und es einen für die Zuordnung verfügbaren Eintrag gibt, ohne dass ein Aussondern eines Opfer-Cacheblockes nötig ist ("RLD leer" in

[Fig. 7](#)), dann kann die Speicherbrücke **32** einen Fill Befehl an den entfernten Knoten mit den Daten senden, welche an die Speicherbrücke **32** in Erwiderung auf den RdShd Vorgang auf der Verbindung **22** geliefert worden sind (Bezugszeichen **92**). Auf der anderen Seite, wenn der RLD Zustand ungültig ist und ein Aussondern eines Opferblockes verwendet wird, um einen RLD Eintrag für Cacheblock A freizumachen, dann kann die Speicherbrücke **32** Prüfbefehle an die entfernten Knoten übertragen, welche Kopien des Opfer-Cacheblockes besitzen. Falls der Opfer-Cacheblock mehrfach benutzt ist, kann die Speicherbrücke **32** einen Kill Befehl (oder Befehle, falls mehrere Knoten den Opfer-Cacheblock mehrfach benutzen) für den Opfer-Cacheblock übertragen (Bezugszeichen **94**). Die entfernten Knoten antworten mit Kill_Ack Befehlen für den Opferblock (Bezugszeichen **96**). Falls der Opferblock modifiziert ist, kann die Speicherbrücke **32** einen Flush-Befehl an den entfernten Knoten übertragen, welcher den modifizierten Zustand besitzt (Bezugszeichen **98**). Der entfernte Knoten kann den modifizierten Block mit einem WB Befehl zurücksenden (Bezugszeichen **100**). In beiden Fällen eines Aussonderns eines Opferblockes kann die Speicherbrücke **32** parallel einen Fill Befehl für den Cacheblock A erzeugen (Bezugszeichen **92**, über Pfeil **102**). Schließlich, wenn der RLD Zustand für den Cacheblock A modifiziert ist, kann die Speicherbrücke **32** einen Flush-Befehl für den Cacheblock A an den entfernten Knoten erzeugen (Bezugszeichen **104**), welcher mit einem WB Befehl und dem Cacheblock A antwortet (Bezugszeichen **106**). Die Speicherbrücke **32** kann dann den Fill Befehl mit dem über den Write Back Befehl gelieferten Cacheblock A übertragen (Bezugszeichen **92**).

[0083] In Erwiderung auf einen cRdExc Befehl für einen Cacheblock A (Referenzzeichen **110**) kann der Ablauf ähnlich wie der cRdShd Fall für einige RLD Zustände sein. Ähnlich dem cRdShd Fall kann die Speicherbrücke **32** einen RdExc Vorgang auf der Verbindung **22** in Erwiderung auf den cRdExc Befehl veranlassen. Ähnlich dem cRdShd Fall kann, wenn der RLD ungültig ist und kein Aussondern eines Opfer-Cacheblockes in dem RLD benötigt wird, um einen Eintrag für den Cacheblock A zuzuordnen, die Speicherbrücke **32** den auf der Verbindung **22** für den RdExc Vorgang gelieferten Cacheblock in einem Fill Befehl an den entfernten Knoten liefern (Bezugszeichen **112**). Zusätzlich kann, wenn der RLD Zustand ungültig für den Cacheblock A ist, und ein Opfer-Cacheblock aus dem RLD **34** ausgesondert wird, die Speicherbrücke **32** in zu dem cRdShd Fall ähnlicher Weise arbeiten (Bezugszeichen **114** und **116** und Pfeil **122** für den mehrfach benutzten Fall des Opferblocks und Bezugszeichen **118** und **120** und Pfeil **122** für den modifizierten Fall des Opferblocks). Wenn der RLD Zustand für den Cacheblock A modifiziert ist, kann die Speicherbrücke **32** in zu dem cRdShd Fall ähnlicher Weise arbeiten (Bezugszeichen **124** und

126). Wenn der RLD Zustand für den Cacheblock A mehrfach benutzt ist, kann die Speicherbrücke **32** Kill-Befehle für jeden entfernten, mehrfach benutzenden Knoten erzeugen (Bezugszeichen **128**). Die Speicherbrücke **32** kann auf die Kill_Ack Befehle von den entfernten, mehrfach benutzenden Knoten warten (Bezugszeichen **130**) und dann den Fill Befehl mit dem auf der Verbindung **22** in Erwiderung auf den RdExc Vorgang gelieferten Cacheblock A übertragen (Bezugszeichen **112**).

[0084] In Erwiderung auf einen Wr Befehl an den Cacheblock A kann die Speicherbrücke **32** einen Wr Vorgang auf der Verbindung **22** erzeugen. Wenn der RLD Zustand für den Cacheblock A ungültig ist, kann die Speicherbrücke **32** die Schreibdaten auf der Verbindung **22** übertragen und der Wr Befehl ist vollständig (Bezugszeichen **142**). Wenn der RLD Zustand für den Cacheblock A mehrfach benutzt ist, kann die Speicherbrücke **32** Kill-Befehle an jeden entfernten mehrfach benutzenden Knoten erzeugen (Bezugszeichen **144**) und die Kill_Ack Befehle von jenen entfernten Knoten zusätzlich zu dem Übertragen der Daten auf der Verbindung **22** empfangen (Bezugszeichen **146**). Wenn der RLD Zustand für einen entfernten Knoten modifiziert ist, kann die Speicherbrücke **32** einen Flush-Befehl an den entfernten Knoten erzeugen (Bezugszeichen **148**) und den WB Befehl von dem entfernten Knoten empfangen (Bezugszeichen **150**). In einer Ausführungsform kann die Speicherbrücke **32** das Übertragen der Schreibdaten auf der Verbindung **22** verzögern, bis der WB Befehl oder die Kill_Ack Befehle empfangen werden (obwohl die mit den WB Befehl zurückgesendeten Daten von der Speicherbrücke **32** fallengelassen werden können).

[0085] Die obigen Befehle werden von der Speicherbrücke **32** für Cacheblöcke empfangen, für welche der Knoten **10** mit der Speicherbrücke **32** der Heimatknoten ist. Die Speicherbrücke **32** kann auch Flush-Befehle oder Kill-Befehle für jeden Cacheblock empfangen, für welchen der Knoten **10** ein entfernter Knoten ist. In Erwiderung auf einen Flush-Befehl an den Cacheblock A (Bezugszeichen **160**) kann die Speicherbrücke **32** einen RdKill oder RdInv Vorgang auf der Verbindung **22** veranlassen. Wenn der lokale Zustand des Cacheblockes modifiziert ist, kann die Speicherbrücke **32** einen WB Befehl an den Heimatknoten übertragen, wobei der Cacheblock auf der Verbindung **22** in Erwiderung auf den RdKill oder RdInv Vorgang geliefert wird (Bezugszeichen **162**). Wenn der lokale Zustand des Cacheblocks nicht modifiziert ist, antwortet die Speicherbrücke **32** auf den Flush-Befehl vielleicht nicht. In diesem Fall hat der Knoten vielleicht bereits einen WB Befehl an den Heimatknoten übertragen (z.B. in Erwiderung auf das lokale Aussondern des Cacheblockes). In Erwiderung auf einen Kill-Befehl an den Cacheblock A (Bezugszeichen **170**) kann die Speicherbrücke **32** einen RdKill oder RdInv Vorgang auf der Verbindung **22** ver-

lassen. Die Speicherbrücke **32** kann auf den Kill-Befehl mit einem Kill_Ack Befehl antworten (Bezugszeichen **172**).

[0086] In einem Ausführungsbeispiel kann die Speicherbrücke **32** auch eingerichtet sein, einen Lesebefehl (RdNC) zu empfangen, welcher nicht in einem Cachespeicher aufgenommen werden kann (z.B. entsprechend einem Standard HT read) (Bezugszeichen **180**). Als Antwort kann die Speicherbrücke **32** einen RdShd Vorgang auf der Verbindung **22** veranlassen. Falls der RDL Zustand für den Cacheblock mit den zu lesenden Daten modifiziert ist, kann die Speicherbrücke **32** einen Flush-Befehl an den entfernten Knoten, welcher den modifizierten Cacheblock besitzt, senden (Bezugszeichen **182**) und kann den WB Befehl von dem entfernten Knoten empfangen (Bezugszeichen **184**). Zusätzlich kann die Speicherbrücke **32** in Erwiderung auf den RdShd Vorgang auf der Verbindung **22** empfangene Daten als eine Lese-Antwort (RSP = Read Response) an den anfordernden Knoten liefern (Bezugszeichen **186**).

[0087] Übergehend zu [Fig. 8](#), ist ein Blockdiagramm einer Ausführungsform der Speicherbrücke **32** ausführlicher gezeigt. In der Ausführungsform von [Fig. 8](#) schließt die Speicherbrücke **32** eine Eingangssteuerschaltung **190**, das entfernte Leitungsverzeichnis **34**, eine RLD Steuerschaltung **192**, eine Ausgangssteuerschaltung **194** und eine Schalter-Schnittstellenschaltung **196** ein. Die Eingangssteuerschaltung **190**, das entfernte Leitungsverzeichnis **34**, die RLD Steuerschaltung **192** und die Ausgangssteuerschaltung **194** sind jeweils angeschlossen, um die Knotennummer (Knoten #) von dem Konfigurationsregister **38** zu empfangen. Zusätzlich sind die Eingangssteuerschaltung **190**, das entfernte Leitungsverzeichnis **34**, die RLD Steuerschaltung **192** und die Ausgangssteuerschaltung **194** jeweils angeschlossen, um Information von der Verbindung **22** zu empfangen und/oder Information an die Verbindung **22** zu übertragen. Es kann eine Verbindungs-Schnittstellenschaltung (nicht gezeigt) vorhanden sein, eingerichtet, auf der Verbindung **22** im Auftrag der obigen Schaltungen zu kommunizieren, sofern gewünscht. Das entfernte Leitungsverzeichnis **34**, die Eingangssteuerschaltung **190** und die Ausgangssteuerschaltung **192** sind mit der RLD Steuerschaltung **192** verbunden. Die Eingangssteuerschaltung **190** und die Ausgangssteuerschaltung **194** sind mit der Schalter-Schnittstellenschaltung **196** verbunden, welche ferner angeschlossen ist, um Schnittstellen des Schalters **18** für die eingehenden und ausgehenden Befehle anzufordern/zu gestatten. Die Eingangssteuerschaltung **190** schließt eine Gruppe Puffer für ausgehende Befehle **200** ein, welche angeschlossen sind, um Daten an den Schalter **18** zu liefern. Die Ausgangssteuerschaltung **194** kann ferner eine Adresskarte **202** einschließen.

[0088] Die Eingangssteuerschaltung **190** kann Befehle von anderen Knoten über den Schalter **18** empfangen und kann entsprechende Vorgänge auf der Verbindung **22** initiieren lassen. Die Befehle können in den Puffern für eingehende Befehle **200** empfangen werden, welche Puffer für jeden virtuellen Kanal einschließen können, auf welchen Befehle übertragen werden können. So können in diesem Beispiel die Puffer CFill Puffer für den virtuellen CFill Kanal, ACK Puffer für den virtuellen ACK Kanal, Probe Puffer für den virtuellen Probe Kanal, RSP Puffer für den virtuellen RSP Kanal, PC Puffer für den virtuellen PC Kanal und NPC Puffer für den virtuellen NPC Kanal einschließen. Die Puffer **200** können sogar getrennte Adress- und Datenpuffer aufweisen, wobei die Adresspuffer die Adresse und andere Eigenschaften von jedem Befehl speichern und die Datenpuffer die entsprechenden Daten für jene Befehle, welche die Daten (z.B. ein Cacheblock) tragen, speichern. Im Allgemeinen kann, wenn die Eingangssteuerschaltung **190** einen für einen bestimmten virtuellen Kanal verfügbaren Puffer besitzt, sie dies der Schalter-Schnittstellenschaltung melden, wer von dem Schalter auf jenen virtuellen Kanal Daten anfordern kann. Falls eine Quelle (z.B. die Schnittstellenschaltungen **20A–20C**) einen Befehl zum Übertragen auf jenem virtuellen Kanal hat und jenen virtuellen Kanal anfordert, kann der Schalter **18** den virtuellen Kanal jener Quelle zusprechen und kann die Schalter-Schnittstellenschaltung **196** von dem Zusprechen in Kenntnis setzen. Das Zusprechen kann eine Anzeige der Quelle und des virtuellen Kanals, welcher zugesprochen wird, einschließen und kann optional eine Anzeige eines Paketendes (EOP = end of packet) einschließen, welche anzeigt, ob die aktuelle Übertragung die letzten in dem Befehl eingeschlossenen Daten einschließt oder nicht. In einem Ausführungsbeispiel kann der Schalter **18** zu einem Zeitpunkt eine Übertragung durch den Schalter gewähren, wobei eine Übertragung bis zu 16 Byte einschließt. Folglich können Befehle, welche eine Größe von mehr als 16 Byte aufweisen, mehr als eine Übertragung durch den Schalter **18** verwenden.

[0089] Die Eingangssteuerschaltung **190** kann im allgemeinen einen Befehl aus jedem virtuellen Kanal für das Veranlassen eines entsprechenden Vorgangs auf der Verbindung **22** auswählen. Die Eingangssteuerschaltung **190** kann im allgemeinen nicht eine Ordnung unter den virtuellen Kanälen durchsetzen. Eine Ausnahme für Ausführungsformen, welche auch die Standard HT Befehle beherrschen, sind die von HT eingesetzten Ordnungsregeln unter virtuellen Kanälen. Die Eingangssteuerschaltung **190** kann diesen Ordnungsregeln für Standard HT Befehle folgen. In Abwesenheit von Ordnungseinschränkungen kann jeder Auswahlmechanismus für die Auswahl eines auf der Verbindung **22** durchzuführenden Arbeitsvorgangs verwendet werden. Z.B. kann eine zyklische

Auswahl unter den virtuellen Kanälen verwendet werden und andere zyklische Mechanismen können innerhalb der Befehle in jedem virtuellen Kanal verwendet werden. Der zu einem bestimmten Vorgang zugeordnete Puffer **198** kann in einigen Ausführungsformen in Erwiderung auf den auf der Verbindung **22** veranlassten Vorgang freigegeben werden (um einen anderen Befehl zu speichern).

[0090] Es wird bemerkt, dass einige eingehende Befehle vielleicht nicht zu Vorgängen auf der Verbindung **22** führen. Z.B. werden Kill_Ack Befehle vielleicht nicht auf der Verbindung **22** übertragen. Stattdessen können Kill_Ack Befehle zu den entsprechenden Prüfbefehlen in dem Puffer für ausgehende Prüfbefehle (einer der Puffer **200**) gezählt werden. WB Befehle, die fallengelassen werden, führen vielleicht nicht zu Vorgängen auf der Verbindung **22**. CFill Befehle können mit der Datenphase eines vorhergehenden Vorgangs auf der Verbindung **22** enden.

[0091] Das entfernte Leitungsverzeichnis **34** und die RLD Steuerschaltung **192** können die Verbindung **22** für den Beginn von Vorgängen auf der Verbindung **22** überwachen. Wenn ein Vorgang einen lokalen Cacheblock in dem entfernten kohärenten Platz **48** adressiert (z.B. stimmt das wichtigste Halbbyte der Adresse mit der Knotennummer des Knotens überein), kann die RLD Steuerschaltung **192** eine Antwort während der Antwortphase des Vorganges liefern, basierend auf dem Zustand des Cacheblockes in anderen Knoten. Das entfernte Leitungsverzeichnis **34** kann eine Meldung an die RLD Steuerschaltung **192** ausgeben, ob der Vorgang ein Treffer oder kein Treffer in dem entfernten Leitungsverzeichnis **34** ist, und kann ferner den Zustand ausgeben (einschließlich einer Anzeige, ob der Block modifiziert ist oder nicht und einer Anzeige jedes mehrfach benutzenden Knotens). Die RLD Steuerschaltung **192** kann die Information verwenden, um die Antwort zu formulieren. Zusätzlich kann, wenn der Vorgang ein Treffer in dem entfernten Leitungsverzeichnis **34** ist und die Quelle des Vorgangs die Speicherbrücke **32** ist (z.B. die Eingangssteuerschaltung **190**), die RLD Steuerschaltung **192** eine Aktualisierung an das entfernte Leitungsverzeichnis **34** liefern. Das entfernte Leitungsverzeichnis **34** kann für einen Vorgang vor dem nächsten möglichen Vorgang zu der treffenden Leitung aktualisiert werden, sodass der nächste Vorgang Information von dem entfernten Leitungsverzeichnis **34** empfängt, welche den Vorgang widerspiegelt. In einem Ausführungsbeispiel folgt der nächste mögliche Vorgang zu der treffenden Leitung auf die Antwortphase des Vorgangs und somit kann die Aktualisierung von dem anfänglichen Lesen geleitet werden (welches in Erwiderung auf die Adressphase des Vorgangs erfolgt).

[0092] In einem Ausführungsbeispiel kann das ent-

fernte Leitungsverzeichnis **34** eine Untermenge der mehrfach benutzbaren Cacheblöcke in dem Knoten **10** überwachen. D.h., das entfernte Leitungsverzeichnis **34** ist vielleicht nicht groß genug, um das Mehrfachbenutzen aller Cacheblöcke innerhalb des zu dem Knoten **10** zugeordneten Teils des entfernten kohärenten Platzes **48** zu ermöglichen. In derartigen Ausführungsbeispielen kann das entfernte Leitungsverzeichnis **34** den Prüfpuffern innerhalb der Puffer für ausgehende Befehle **200** Prüfbefehle liefern, die zu einem Opfer-Cacheblock gehören, wenn der Vorgang kein Treffer in dem entfernten Leitungsverzeichnis **34** ist und das entfernte Leitungsverzeichnis **34** einen Eintrag auswählt, welcher aktuell dem Opfer-Cacheblock zugeordnet ist, um die fehlende Blockinformation zu speichern.

[0093] Die Ausgangssteuerschaltung **194** kann die Vorgänge auf der Verbindung **22** überwachen und kann Befehle erzeugen, die an andere Knoten in Erwiderung auf die Vorgänge zu übertragen sind. Die Ausgangssteuerschaltung **194** kann die Knotennummer empfangen und kann entfernte Vorgänge erfassen, welche die Knotennummer verwenden. Falls die Erwiderung auf den entfernten Vorgang anzeigt, dass der Knoten kein ausreichendes Benutzerrecht an dem entfernten Cacheblock besitzt, um den Vorgang zu vollenden, kann die Ausgangssteuerschaltung **194** eine entsprechende Anfrage an den Heimatknoten für den Cacheblock erzeugen (z.B. cRdShd, cRdExc oder ein nicht-kohärentes Schreiben).

[0094] Zusätzlich kann, falls der Vorgang von der Eingangssteuerschaltung **190** initiiert ist (in Erwiderung auf einen empfangenen Befehl), die Ausgangssteuerschaltung **194** einen Befehl erzeugen. Z.B. kann, falls der Vorgang in Erwiderung auf einen von der Eingangssteuerschaltung **190** empfangenen Prüfbefehl erfolgt, die Ausgangssteuerschaltung **194** eine Bestätigung (Kill_Ack oder WB) für den Befehl erzeugen.

[0095] Falls der Vorgang von der Eingangssteuerschaltung **190** oder irgendeinem kohärenten Agenten in dem Knoten initiiert ist und ein Vorgang ist, für welchen der Knoten der Heimatknoten ist, kann die Ausgangssteuerschaltung **194** Prüfbefehle, wie angezeigt, in Erwiderung auf den Vorgang und den RLD Status für den Cacheblock erzeugen. Zusätzlich kann die Ausgangssteuerschaltung **194** Fülldaten von der Datenphase eines Lesevorgangs erfassen, welcher von der Eingangssteuerschaltung **190** in Erwiderung auf einen Lese-Kohärenzbefehl von einem anderen Knoten erzeugt wurde und kann einen Fill-Befehl erzeugen, um die Daten an den anderen Knoten zu übertragen. Die Ausgangssteuerschaltung **194** kann Daten für einen RdNC Vorgang erfassen, welcher von der Eingangssteuerschaltung **190** in Erwiderung auf einen RdNC Befehl erzeugt wurde, und eine Erwiderung (in dem virtuellen RSP Kanal) für

derartige Lesevorgänge erzeugen.

[0096] Im allgemeinen kann die Ausgangssteuerschaltung **194** jeden Befehl für eine Übertragung in einen virtuellen Kanal auswählen. Jedoch kann es in einigen Ausführungsbeispielen einige Ordnungsregeln geben, welchen die Ausgangssteuerschaltung **194** folgen kann. Z.B. können Anfragen an denselben Cacheblock in der Ausgangssteuerschaltung **194** in eine serielle Reihenfolge gebracht werden (z.B. Lese-Anforderungen in den CRd Befehl-Puffern). Zusätzlich können Prüfbefehle an denselben Cacheblock in eine serielle Reihenfolge gebracht werden.

[0097] Die Ausgangssteuerschaltung **194** kann einen Ausgangs-NPC-Puffer in Erwiderung auf einen Vorgang auf der Verbindung **22** zuordnen, welcher an einen anderen Knoten in dem virtuellen NPC Kanal übertragen werden soll (z.B. ein nicht-kohärenter Lese-Befehl oder ein nicht-kohärenter Schreib-Befehl, welcher nicht gesendet wurde) und kann den Puffer in Erwiderung auf ein Übertragen des Befehls an den Schalter **18** freigeben. In ähnlicher Weise wird ein Ausgangs-PC-Puffer in Erwiderung auf einen Vorgang auf der Verbindung **22** zugeordnet, welcher an einen anderen Knoten in dem virtuellen PC Kanal übertragen werden soll. Die Ausgangssteuerschaltung **194** kann einen Ausgangs-CRd-Puffer in Erwiderung auf einen RdShd oder einen RdExc Vorgang reservieren, für welchen das lokale Eigentumsrecht unzureichend für ein Vollenden des Vorgangs ist, und kann den Puffer in Erwiderung auf ein Übertragen des Befehls an den Schalter **18** freigeben. Ein RSP-Puffer kann in Erwiderung auf einen von der Eingangssteuerschaltung **190** initiierten NPC Vorgang reserviert werden. Ein CFill Puffer kann in Erwiderung auf einen von der Eingangssteuerschaltung **190** initiierten CRd Vorgang reserviert werden, um die Fülldaten zu übertragen. Die RSP oder CFill Puffer werden freigegeben, sobald die Daten an den Schalter **18** übertragen worden sind. Prüfpuffer können in Erwiderung auf Vorgänge auf der Verbindung **22**, für welche der Knoten der Heimatknoten ist, oder in Erwiderung auf einen Opfer-Cacheblock von dem entfernten Leitungsverzeichnis **34** reserviert werden. In einem Ausführungsbeispiel bleiben Prüfpuffer reserviert, bis die Erwiderungen auf die Prüfbefehle empfangen worden sind. Ein ACK Puffer kann in Erwiderung auf ein Einsetzen eines Vorgangs in Erwiderung auf einen Prüfbefehl von der Eingangssteuerschaltung **190** reserviert werden und kann freigegeben werden, wenn die Bestätigung (Kill_Ack oder WB) an den Schalter **18** übertragen wird.

[0098] In dem vorliegenden Ausführungsbeispiel kann die Ausgangssteuerschaltung **194** die Adresskarte **202** zum Abbilden von Knotennummern (entweder von der Adresse eines Vorgangs oder Meldungen von dem entfernten Leitungsverzeichnis **34**, für Prüfbefehle) auf eine Schnittstellenschaltung

20A–20C, über welche der Befehl an den Knoten übertragen werden soll. Die Adresskarte **202** kann durch Software programmierbar sein (z.B. indem Konfigurationsraum-Befehle verwendet werden, oder als ein Satz von Konfigurationsregistern, welche für Software zugänglich sind). Eine Meldung der Schnittstellenschaltung **20A–20C** kann in den Befehlpuffern **200** gespeichert werden oder alternativ kann das Abbilden von der Knotennummer auf die Schnittstellenschaltung durchgeführt werden, während Anfragen an den Schalter **18** gemacht werden.

[0099] Die Ausgangssteuerschaltung **194** kann die Schalter-Schnittstellenschaltung **196** informieren, wenn ein Befehl bereit ist, auf eine bestimmte Schnittstellenschaltung **20A–20C** auf einem bestimmten virtuellen Kanal übertragen zu werden. Die Schalter-Schnittstellenschaltung **196** kann Anfragen für die bestimmte Schnittstellenschaltung **20A–20C** und den bestimmten virtuellen Kanal machen (Dest und VC in [Fig. 8](#)). Zusätzlich kann die Schalter-Schnittstellenschaltung **196** anzeigen, ob die Übertragung, wenn gewährt, die letzte Übertragung des aktuellen Befehls an jenes Ziel und an jenen virtuellen Kanal mit der EOP Anzeige ist. Wenn ein Ziel und ein virtueller Kanal gewährt werden, kann die Schalter-Schnittstellenschaltung **196** die Ausgangssteuerschaltung **194** informieren, welche Daten von dem ausgewählten VC Puffer übertragen kann (insbesondere von einem Befehl von dem ausgewählten VC Puffer, welches an das gewährte Ziel übertragen werden soll).

[0100] Es wird erwähnt, dass die Eingangs- und Ausgangssteuerschaltungen **190** und **194** eine Ablaufsteuerung einsetzen können, um sicherzustellen, dass die Puffer für verschiedene virtuelle Kanäle nicht überlaufen. Die Eingangssteuerschaltung **190** führt Ablaufsteuerung durch den Schalter **18** durch, indem Übertragungen von dem Schalter angefordert werden, wenn Puffer für einen bestimmten virtuellen Kanal verfügbar sind und indem keine Übertragungen für virtuelle Kanäle angefordert wird, für welche keine Puffer verfügbar sind. Die Ausgangssteuerschaltung **194** kann die auf der Verbindung **22** unterstützte Ablaufsteuerung verwenden. In einigen Ausführungsbeispielen kann die Verbindung **22** eine Wiederholung von Vorgängen unterstützen. In anderen Ausführungsbeispielen kann die Verbindung **22** eine Quell-Blockier-Maßnahme unterstützen, in welcher jeder Agent, der an verschiedenen Vorgängen teilnimmt, ein Blockier-Signal liefern kann, welches vorgebracht wird, um anzuzeigen, ob er oder ob er nicht geeignet ist, an einem Vorgang eines bestimmten Typs teilzunehmen, wenn er auf der Verbindung **22** übertragen werden würde. Jeder Agent, der Vorgänge initiiert, kann die Blockiersignale empfangen und kann das Initiieren eines Vorgangs unterdrücken, wenn ein Agent, der an jenem Vorgang teilnehmen soll, wenn auf der Verbindung **22** initiiert, sein Blo-

ckiersignal aktiviert hat. Ein bestimmter Agent kann mehr als ein Blockiersignal aufweisen, für unterschiedliche Arten von Vorgängen, unterschiedliche virtuelle Kanäle, etc. In einem Ausführungsbeispiel kann die Ausgangssteuerschaltung **194** ein separates Blockiersignal für jeden virtuellen Ausgangskanal einschließen und kann das entsprechende Blockiersignal, basierend auf der Pufferverfügbarkeit in den Puffern für ausgehende Befehle **200** für jenen virtuellen Kanal aktivieren.

Knoteninterne und knotenübergreifende Übertragung von Eigentumsrecht

[0101] Übergehend zu [Fig. 9](#), ist ein Blockdiagramm eines Ausführungsbeispiels von Knoten **10A** und **10B** gezeigt. Das Blockdiagramm von [Fig. 9](#) erläutert eine Übertragung von Eigentumsrecht für eine beispielhafte Anforderung (ein RdExc von einem anfragenden Agenten **210**) von dem Heimatknoten **10B** des zu der Anforderung gehörigen Cacheblockes an den entfernten Knoten **10A**. Jeder der Knoten **10A** und **10B** kann eine Instantiierung des Knotens **10** sein.

[0102] Wie oben erwähnt, können Vorgänge auf der Verbindung **22** in einem bestimmten Knoten **10** eine Adressphase, eine Antwortphase und eine Datenphase (für Lese- und Schreib-Vorgänge) einbinden. Die Adressphase wird von dem Urheber des Vorgangs übertragen und schließt die Adresse des betroffenen Cacheblockes, die Vorgangsart und andere Steuerinformation ein. Die Antwortphase erfolgt nach der Adressphase und zeigt das Eigentumsrecht von anderen Agenten innerhalb des Knotens an (und das entfernte Leitungsverzeichnis **34** in dem Heimatknoten, als ein Proxy für andere Knoten). In einem Ausführungsbeispiel erfolgt die Antwortphase mit einer vorbestimmten Verzögerung nach der zugehörigen Adressphase. Die vorbestimmte Verzögerung kann in verschiedenen Ausführungsbeispielen programmierbar oder festgelegt sein. In einer Ausführung, in welcher die Verbindung **22** als ein Bus ausgebildet ist, kann die vorbestimmte Verzögerung **2** Taktzyklen des Bus-Taktes nach der Adressphase betragen.

[0103] In dem erläuterten Ausführungsbeispiel wird knoteninternes Eigentumsrecht während der Antwortphase des Vorgangs übertragen. Folglich initiiert in diesem Beispiel der anfragende Agent **210** in dem Knoten **10A** einen RdExc Vorgang an einen Cacheblock, für welchen der Knoten **10B** der Heimatknoten ist. Der anfragende Agent **210** steuert die Adressphase auf der Verbindung **22** und die Adressphase wird von der Speicherbrücke **32** (und anderen kohärenten Agenten innerhalb des entfernten Knotens **10A**) erkannt (Pfeil **212**). Die Antwortphase erfolgt, indem ausschließliches Eigentumsrecht innerhalb des Knotens **10A** an den anfragenden Agenten **210** übertragen wird (Pfeil **214**). D.h., falls ein nachfolgender Vor-

gang auf der Verbindung **22** in dem Knoten **10A** übertragen wird, zeigt der anfragende Agent **210** ausschließliches Eigentumsrecht an. In einem Ausführungsbeispiel wäre der anfragende Agent **210** in diesem Fall für ein Übertragen von Daten für den nachfolgenden Vorgang verantwortlich.

[0104] Obwohl der anfragende Agent **210** ausschließliches Eigentumsrecht an dem betroffenen Cacheblock in dem Knoten **10A** besitzen kann, besitzt der Knoten **10A** vielleicht kein Eigentumsrecht an dem Cacheblock innerhalb des knotenübergreifenden Kohärenzsystems. In Erwiderung auf die Adressphase des Vorgangs (Pfeil **212**) überträgt die Speicherbrücke **32** in dem Knoten **10A** einen cRdExc Befehl an die Speicherbrücke **32** in dem Knoten **10B** (Pfeil **216**). In Erwiderung auf den cRdExc Befehl initiiert die Speicherbrücke **32** in dem Knoten **10B** einen RdExc Vorgang auf der Verbindung **22** in dem Knoten **10B** (Pfeil **218**). Während der Antwortphase des Vorgangs wird das knoteninterne Eigentumsrecht an die Speicherbrücke **32** übertragen (Pfeil **220**). In diesem Beispiel überträgt die Speichersteuerung **14** in dem Knoten **10B** die Daten für den RdExc Vorgang (d.h., in diesem Beispiel wird der Cacheblock nicht an anderer Stelle ausschließlich besessen) (Pfeil **222**) und die Speicherbrücke **32** erfasst die Daten (Pfeil **224**). In Erwiderung auf die Daten überträgt die Speicherbrücke **32** den Fill-Befehl mit den Daten an den Knoten **10A** (Pfeil **226**). Die Speicherbrücke **32** in dem Knoten **10A** kann dann die Daten für den Vorgang auf der Verbindung **22** übertragen (Pfeil **228**).

[0105] Wie [Fig. 9](#) erläutert, gibt es in dem System für einen bestimmten Vorgang zu einem entfernten Cacheblock 2 Ordnungspunkte: Die Verbindung **22** in einem entfernten Knoten und die Verbindung **22** in dem Heimatknoten (z.B. die Verbindungen **22** in jedem der Knoten **10A–10B** in dem Beispiel von [Fig. 9](#)). Während der anfragende Agent **210** ausschließliches Eigentumsrecht an dem betroffenen Cacheblock in dem Knoten **10A** besitzt (knoteninternes Eigentumsrecht), besitzt der Knoten **10A** vielleicht noch nicht ein allgemeines Eigentumsrecht an dem Cacheblock (knotenübergreifendes Eigentumsrecht). Knotenübergreifendes Eigentumsrecht ist abhängig von der Reihenfolge der Vorgänge an dem Heimatknoten (Knoten **10B** in diesem Beispiel). Die in dem Fill Befehl an den Knoten **10A** übertragenen Daten spiegeln die Vorgänge wider, welche dem RdExc Vorgang in dem Knoten **10B** vorausgegangen sind (auch wenn jene Vorgänge rechtzeitig, nachfolgend auf den RdExc Vorgang in dem Knoten **10A** erfolgten). In anderen Worten kann knoteninternes Eigentumsrecht in Erwiderung auf das Übertragen von Daten in diesem System auftreten. Das Knoteninterne Eigentumsrecht tritt in der Antwortphase auf, welche von der Adressphase ausgelöst wird. Demnach erfolgt das Übertragen von knoteninternem Eigentumsrecht in Erwiderung auf die Adressübertragung.

[0106] In einem Ausführungsbeispiel kann die Verbindung **22** keine Wiederholung unterstützen. Das bedeutet, sobald ein Vorgang auf der Verbindung **22** initiiert ist, erfolgt eine Antwortphase in einer vorbestimmten Zeitspanne und das Eigentumsrecht wird in jener Antwortphase übertragen. Auf Wiederholungs-geeigneten Verbindungen kann eine Adressübertragung wiederholt werden oder gelöscht werden (z.B., um zu ermöglichen, dass ein modifizierter Cacheblock in einen Speicher geschrieben wird oder dass eine andere Kohärenzaktivität erfolgt). Eine wiederholte Adressübertragung wird behandelt, bezüglich der Verbindung, als ob sie noch nicht erfolgt ist. Da die Adressübertragung gelöscht wurde, hat sie keine Auswirkung auf ein Eigentumsrecht innerhalb des Knotens. In der derartigen Systemen kann die Übertragung von knoteninternem Eigentumsrecht verzögert werden, bis die Übertragung von knotenübergreifendem Eigentumsrecht erfolgt. Jedoch erlaubt das vorliegende System des Übertragens von knotenübergreifendem Eigentumsrecht in Erwiderung auf den knotenübergreifenden Datentransfer in einigen Ausführungsbeispielen die Verwendung von nicht-Wiederholungs-geeigneten Verbindungen **22**. Andere Ausführungsbeispiele können Wiederholungs-geeignete Verbindungen einsetzen, falls gewünscht.

Entferntes Leitungsverzeichnis

[0107] Übergehend nun zu [Fig. 10](#), ist ein Blockdiagramm eines Ausführungsbeispiels eines Teils der Speicherbrücke **32** gezeigt. In [Fig. 10](#) sind das entfernte Leitungsverzeichnis **34**, die RLD Steuerschaltung **192**, der Puffer für ausgehende Prüfbefehle **200A** (Teil des in [Fig. 8](#) gezeigten Puffers für ausgehende Befehle **200**) und die Vergleichsschaltung **230** gezeigt. Das entfernte Leitungsverzeichnis **34**, die RLD Steuerschaltung **192**, der Puffer für ausgehende Prüfbefehle **200A** und die Vergleichsschaltung **230** empfangen die Adresse (oder zumindest einen Teil davon) eines Vorgangs von der Verbindung **22**. Der Puffer für ausgehende Prüfbefehle **200A** empfängt ferner Opfer-Cacheblock-Information von dem entfernten Leitungsverzeichnis **34**. Die RLD Steuerschaltung **192** empfängt die Ausgabe der Vergleichsschaltung **230**, Befehl-Information von der Verbindung **22** und optional andere Information (z.B. eine Markierung), welche den Ursprung des Vorgangs auf der Verbindung **22** und die Ursprungs-Knotennummer eines Befehls anzeigt, welcher einen von der Eingangssteuerschaltung **190** initiierten Vorgang veranlasst hat. Die RLD Steuerschaltung **192** empfängt ferner eine Treffer-/kein Treffer-Meldung von dem entfernten Leitungsverzeichnis **34** sowie eine Meldung des mehrfach benutzten/modifizierten Zustands des Cacheblocks in anderen Knoten, falls ein Treffer erkannt wird. Die RLD Steuerschaltung **192** kann eine Antwort auf der Verbindung **22** auf einen Vorgang liefern (z.B. mehrfach benutzt, ausschließ-

lich, etc.), welche die Antwort der Speicherbrücke **32** auf der Verbindung **22** auf den Vorgang darstellen kann. Zusätzlich kann die RLD Steuerschaltung **192** eine Aktualisierung zu dem entfernten Leitungsverzeichnis **34** liefern.

[0108] Das in [Fig. 10](#) erläuterte Ausführungsbeispiel kann ein Ausführungsbeispiel sein, in welchem das entfernte Leitungsverzeichnis **34** geeignet ist, den Zustand, in entfernten Knoten, von bis zu einer Maximalzahl von Cacheblöcken zu überwachen, wobei die Maximalzahl geringer ist als die Gesamtzahl an Cacheblöcken in dem Teil des zu dem Knoten zugeordneten entfernten kohärenten Platzes **48**. In einem weiteren Ausführungsbeispiel kann die Maximalzahl weniger als die Zahl von Einträgen in entfernten Caches sein. Da das entfernte Leitungsverzeichnis **34** weniger als die Gesamtzahl an Cacheblöcken überwacht, kann das entfernte Leitungsverzeichnis **34** kleiner sein als ein Verzeichnis, welches die Gesamtzahl an Cacheblöcken überwacht. Das entfernte Leitungsverzeichnis **34** kann in einigen Ausführungsbeispielen einfacher zu erstellen sein, als ein vollständiges Verzeichnis. Außerdem können manche Ausführungsbeispiele, dadurch, dass sie weniger Platz als ein vollständiges Verzeichnis einnehmen, offener für eine Eingliederung in einen Knoten sein, welcher auf einer einzelnen integrierten Schaltung integriert ist.

[0109] Das entfernte Leitungsverzeichnis **34** kann jede Gliederung aufweisen. Z.B. kann das entfernte Leitungsverzeichnis **34** jede der verschiedenen Cache-Gliederungen aufweisen (z.B. Satz-adressiert, vollständig inhaltsorientiert, oder direkt abgebildet). In einer Ausführung kann das entfernte Leitungsverzeichnis **34** z.B. 16 k Einträge aufweisen (wobei jeder Eintrag den entfernten Zustand eines Cacheblockes speichert), welche in einer 8-Weg-Satz-adressierten Struktur angeordnet sind. Jeder Ersetzungsalgorithmus kann verwendet werden. In einem Ausführungsbeispiel kann z.B. ein Zufalls-Ersetzungs-Algorithmus verwendet werden. In anderen Ausführungsbeispielen können verschiedene kürzlich-am-wenigsten-genutzt (LRU = leastrecently-used) basierte Algorithmen verwendet werden (z.B. LRU, Pseudo-LRU, nicht zuletzt genutzt, etc). In einigen Ausführungsbeispielen kann ein Versuch unternommen werden, einen Eintrag auszuwählen, welcher aktuell keinen gültigen Zustand für einen Cacheblock speichert. Falls alle derartigen Einträge einen gültigen Zustand überwachen, dann wird ein Eintrag ersetzt. Andere Ausführungsbeispiele können keinen Versuch unternehmen, einen Eintrag auszuwählen, welcher aktuell keinen gültigen Zustand speichert.

[0110] Auf das entfernte Leitungsverzeichnis **34** ist das Liefern einer kohärenten Antwort im Auftrag von entfernten Knoten auf einen Vorgang auf der Verbindung **22**, für welche der Knoten der Heimatknoten ist,

und das Ermitteln gestützt, ob Prüfbefehle oder ob keine Prüfbefehle in Erwiderung auf den Vorgang erzeugt wurden. Folglich, falls ein Eintrag in dem entfernten Leitungsverzeichnis **34** in Erwiderung auf einen Vorgang ausgewählt wurde und jener Eintrag einen gültigen Zustand des Cacheblockes speichert (bezeichnet als "Opfer-Cacheblock", da er ersetzt wird), erzeugt die Speicherbrücke **32** Prüfbefehle, um den Opfer-Cacheblock in den entfernten Knoten, welche den Opfer-Cacheblock speichern, für ungültig zu erklären. Zusätzlich können, falls ein entfernter Knoten eine modifizierte Kopie des Opfer-Cacheblockes besitzt, die Prüfbefehle ein WB des Opfer-Cacheblockes an den Heimatknoten veranlassen.

[0111] In dem Ausführungsbeispiel gemäß [Fig. 10](#) empfängt das entfernte Leitungsverzeichnis **34** die Adresse eines Vorgangs und ermittelt, ob die Adresse einen Treffer oder keinen Treffer darstellt. Falls der Vorgang keinen Treffer darstellt, liefert das entfernte Leitungsverzeichnis Opfer-Information an den Puffer für ausgehende Prüfbefehle **200A**. Die Opfer-Information kann die Adresse des Opfer-Blockes, eine Anzeige, ob der Block modifiziert ist oder nicht, und eine Anzeige davon aufweisen, welche Knoten eine Kopie des Cacheblockes besitzen. Die Opfer-Information kann in einem Ausführungsbeispiel eine Kopie des für eine Ersetzung ausgewählten Eintrags sein. Ein beispielhafter Eintrag ist in [Fig. 11](#) gezeigt. Der Puffer für ausgehende Prüfbefehle **200A** kann einen Eintrag für den Prüfbefehl reservieren und die Adresse und die Knoteninformation speichern und kann Prüfbefehle an jeden Knoten erzeugen, um die Kopie des Cacheblockes in jenen Knoten für ungültig zu erklären. Die Prüfbefehle können an den Schalter **18** in der gleichen Weise wie andere Prüfbefehle übertragen werden.

[0112] Falls die Adresse einen Treffer in dem entfernten Leitungsverzeichnis **34** darstellt, kann das entfernte Leitungsverzeichnis **34** den Treffer an die RLD Steuerschaltung **192** melden. Zusätzlich kann das entfernte Leitungsverzeichnis eine Anzeige, ob der Block in den entfernten Knoten mehrfach benutzt oder modifiziert ist, und eine Anzeige davon liefern, welcher oder welche Knoten Kopien besitzen. Die RLD Steuerschaltung **192** kann, basierend auf dem von dem entfernten Leitungsverzeichnis **34** gemeldete Zustand und der Vorgangsart, bestimmen, ob, oder ob nicht, Prüfbefehle an einen oder mehrere entfernte Knoten erzeugt werden, um das von dem Vorgang angeforderte Eigentumsrecht zu liefern. Die RLD Steuerschaltung **192** kann dem Puffer für ausgehende Prüfbefehle **200A** melden, dass ein Prüfbefehl erzeugt werden soll, die Art des Prüfbefehls (Flush oder Kill) und den Knoten oder die Knoten, an welche der Prüfbefehl übertragen werden soll. Der Puffer für ausgehende Prüfbefehle **200A** empfängt die Adresse des Vorgangs von der Verbindung **22** zur Verwendung für die Prüfbefehle.

[0113] Beispielhafte Prüfbefehl-Puffereinträge sind in [Fig. 10](#) erläutert, wobei zumindest etwas von der Information, welche in den Einträgen gespeichert werden kann, eingeschlossen ist. Jeder Eintrag kann in diesem Ausführungsbeispiel die Adresse des Blockes, für welchen die Prüfbefehle erzeugt worden sind (Addr), die Knoten, an welche der Prüfbefehl übertragen werden soll (N), ob die Prüfbefehle übertragen worden sind oder nicht (nicht gezeigt), ob die Bestätigungen von jedem Knoten empfangen worden sind oder nicht (A) und eine Anzeige, ob die Prüfbefehle als ein Ergebnis eines Aussonderns erzeugt wurden oder nicht (E) einschließen. Nicht in [Fig. 10](#) gezeigt ist die Kommunikation mit dem Eingangs-ACK-Puffer (Teil der Puffer **198**), um einen Empfang von Bestätigungen zu erfassen, um das A Feld zu aktualisieren.

[0114] Die Ausgesondert-Anzeige (E) in jedem Eintrag wird verwendet, da der Zustand des Opfer-Cacheblockes in den entfernten Knoten durch den Prüfbefehl-Puffereintrag für jenen Opfer-Cacheblock dargestellt ist. D.h., das entfernte Leitungsverzeichnis **34** ist aktualisiert worden, um die Opferblock-Information zu löschen. Folglich, falls ein Vorgang auf einen Prüfbefehl-Puffereintrag trifft, für welchen die Ausgesondert-Anzeige anzeigt, dass die Prüfbefehle das Ergebnis eines Aussonderns sind, schließt die Antwort auf den Vorgang den Zustand des entfernten Blockes, wie durch die Ausgesondert-Anzeige angezeigt, ein. Z.B., falls der Prüfbefehl einen modifizierten Block von einem entfernten Knoten abrufen kann, kann die Antwort von der Speicherbrücke **32** (geliefert von der RLD Steuerschaltung **192**) ausschließlich lauten (so dass der modifizierte Block als die Daten geliefert werden kann, wenn er von dem entfernten Knoten zurückgesendet wird). Durch Suchen nach ausgesonderten Einträgen in dem Prüfbefehl-Puffer kann die korrekte Antwort geliefert werden.

[0115] Die Vergleichsschaltung **230** kann die Adresse von der Verbindung **22** empfangen und kann die Adresse mit Adressen in den Puffer für ausgehende Prüfbefehle **200A** vergleichen. Die Vergleichsschaltung **230** kann zutreffende Einträge an die RLD Steuerschaltung **290** melden, welche die Ergebnisse (und die ausgesondert-Anzeigen) beim Ermitteln der auf der Verbindung **22** vorzulegenden Antwort verwenden kann. Zusätzlich kann, falls ein Vorgang auf der Verbindung **22** bewirkt, dass ein Eintrag in dem entfernten Leitungsverzeichnis **34** nach einem Aussondern neu zugeteilt wird, für welches Prüfbefehle in dem Puffer für ausgehende Prüfbefehle **200A** verbleiben, die RLD Steuerschaltung **192** die ausgesondert-Anzeige zu nicht mehr ausgesondert-anzeigen ändern. In dieser Weise kann der neu geschaffene Eintrag in dem entfernten Leitungsverzeichnis **34** die Antworten auf nachfolgende Vorgänge steuern.

[0116] Die RLD Steuerschaltung **192** kann eine Ak-

tualisierung an das entfernte Leitungsverzeichnis **34** für jeden Vorgang erzeugen, der einen Cacheblock betrifft, welcher von mindestens einem entfernten Knoten in einen Cachespeicher abgelegt wird. Die RLD Steuerschaltung **192** kann eine Meldung der Ursprungsknotennummer eines Kohärenzbefehles empfangen, welcher den Vorgang von der Eingangssteuerschaltung **190** aus verursacht hat. Die RLD Steuerschaltung **192** kann den Eintrag aktualisieren, um anzuzeigen, dass der Ursprungsknoten eine Kopie besitzt (mehrfach benutzt oder modifiziert, abhängig von dem Vorgang) und kann den Eintrag aktualisieren, um jeden Knoten zu löschen, an welchen Prüfbefehle in Erwiderung auf den Vorgang erzeugt wurden. Für den Fall kein Treffer von dem entfernten Leitungsverzeichnis **34** kann die RLD Steuerschaltung **192** den Teil der Adresse verwenden, welcher die Knotennummer anzeigt (z.B. in einem Ausführungsbeispiel das wichtigste Halbbyte), um die Vorgänge zu identifizieren, für welche der Knoten der Heimatknoten ist (und somit das entfernte Leitungsverzeichnis aktualisiert werden kann).

[0117] [Fig. 17](#) zeigt eine Tabelle, welche ein Ausführungsbeispiel von Aktualisierungen des entfernten Leitungsverzeichnisses **34** in Erwiderung auf Vorgänge auf der Verbindung **22** zeigt. Die „Ursprung“-Spalte in [Fig. 17](#) zeigt, ob der Ursprung ein lokaler kohärenter Agent ist (z.B. die Prozessoren **12A–12N**) oder ein entfernter Agent (über die Speicherbrücke **32**). Die „Vorgang“-Spalte in [Fig. 17](#) zeigt den Vorgang (einer der Vorgänge aus Tabelle 42). Die „RLD Zustand“-Spalte in [Fig. 17](#) zeigt den Zustand, welcher von dem entfernten Leitungsverzeichnis **34** in Erwiderung auf den Vorgang ausgegeben wird. Die möglichen Zustände in diesem Ausführungsbeispiel sind M (Modified = modifiziert), S (Shared = mehrfach benutzt) oder I (Invalid, or miss = ungültig oder kein Treffer). Die „neue RLD Zustand“-Spalte in [Fig. 17](#) zeigt den Zustand, welchen die RLD Steuerschaltung **192** zur Aktualisierung in das entfernte Leitungsverzeichnis **34** in Erwiderung auf den Vorgang erzeugt. Die „Eigentümer setzen?“-Spalte in [Fig. 17](#) zeigt an, ob der entfernte Knoten, welcher den Vorgang verursacht hat, als Eigentümer in dem entfernten Leitungsverzeichnis **34** angezeigt wird oder nicht (worin „–“, nicht relevant bedeutet). Z.B. zeigt in dem unten beschriebenen Ausführungsbeispiel gemäß [Fig. 11](#) die „Eigentümer setzen?“-Spalte an, ob der entfernte Knoten in dem Knotenvektor angezeigt ist oder nicht. Die „andere Eigentümer löschen?“-Spalte zeigt an, ob andere Eigentümer, die in dem entfernten Leitungsverzeichnis **34** angezeigt werden können, gelöscht werden oder nicht. Z.B. zeigt in dem Ausführungsbeispiel gemäß [Fig. 11](#) die „andere Eigentümer löschen?“-Spalte an, ob andere Eigentümer aus dem Knotenvektor gelöscht werden oder nicht.

[0118] Zusätzlich zu den gemäß den Erläuterungen in [Fig. 17](#) vorgenommenen Zustandsänderungen

kann die Erwiderung auf die RLD Steuerschaltung **194** mehrfach benutzt lauten (SHD bestätigt, EXC nicht bestätigt), falls der RLD Zustand mehrfach benutzt ist, und kann ausschließlich lauten (SHD nicht bestätigt, EXC bestätigt), falls der RLD Zustand modifiziert ist. Zusätzlich kann, wie oben erläutert, die Antwort basierend auf den Aussonderungen in dem Prüfbefehl-Puffer **200A** erzeugt werden.

[0119] Nun übergehend zu [Fig. 11](#), ist ein Blockdiagramm eines Ausführungsbeispiels eines Eintrags **240** für ein Ausführungsbeispiel des entfernten Leitungsverzeichnisses **34** gezeigt. In dem Ausführungsbeispiel gemäß [Fig. 11](#) schließt der Eintrag **240** ein Adressfeld **242**, eine gültig-Anzeige **244**, eine modifiziert-Anzeige **246** und einen Knotenvektor **248** ein.

[0120] Das Adressfeld **242** schließt mindestens einen Teil der Adresse des Cacheblockes ein, welche den Cacheblock in dem entfernten kohärenten Platz **48** identifiziert. Der in dem Adressfeld **242** gespeicherte Teil der Adresse kann die Adressbits, welche einen Offset innerhalb des Cacheblockes festlegen, die Adressbits, welche als ein Index in dem entfernten Leitungsverzeichnis **34** verwendet werden (für satzadressierte oder direkt-abgebildete Ausführungsbeispiele), und/oder die Adressbits einschließen, welche die Knotennummer festlegen. Die Index-Bits sind in der Reihe des entfernten Leitungsverzeichnisses **34** eingeschlossen, welche den Eintrag einschließt, und die Knotennummer-Bits sind eingeschlossen, da das entfernte Leitungsverzeichnis **34** den entfernten Zustand von Cacheblöcken überwacht, für welche der Knoten der Heimatknoten ist. Andere Ausführungsbeispiele können auch jedes der oben erwähnten Adress-Bits speichern, sofern gewünscht.

[0121] Die gültig-Anzeige **244** kann anzeigen, ob ein Eintrag gültig ist oder nicht. Die gültig-Anzeige **244** kann ein Bit aufweisen, welches anzeigt, wenn gesetzt, dass der Eintrag gültig ist und welches anzeigt, wenn nicht gesetzt, dass der Eintrag ungültig ist. Andere Ausführungsbeispiele können die gesetzt- und nicht gesetzt- Bedeutungen des Bits umkehren oder andere Anzeigen verwenden. Die modifiziert-Anzeige **246** kann anzeigen, ob der Cacheblock in einem entfernten Knoten modifiziert ist oder nicht. Die modifiziert-Anzeige **246** kann ein Bit aufweisen, welches anzeigt, wenn gesetzt, dass der Block modifiziert ist, und welches anzeigt, wenn nicht gesetzt, dass der Block nicht modifiziert ist (d.h. in einer Ausführung, welche ein knotenübergreifendes MSI-Protokoll verwendet, ist er mehrfach benutzt). Andere Ausführungsbeispiele können die gesetzt- und nicht gesetzt- Bedeutungen des Bits umkehren oder andere Anzeigen verwenden.

[0122] Der Knotenvektor **248** kann eine Anzeige für

jeden Knoten einschließen, welche anzeigt, ob jener Knoten eine Kopie des Cacheblockes besitzt oder nicht. Wenn die modifiziert-Anzeige **246** modifiziert anzeigt, kann der Knotenvektor **248** für das vorliegende, knotenübergreifende Protokoll anzeigen, dass der modifizierte Knoten eine Kopie besitzt und andere Knoten nicht. Falls die modifiziert-Anzeige **246** nicht modifiziert (mehrfach benutzt) anzeigt, kann der Knotenvektor **248** einen oder mehrere Knoten anzeigen, welche mehrfach benutzte Kopien besitzen. In einem Ausführungsbeispiel kann der Knotenvektor **248** ein Bit für jeden Knoten einschließen. Das Bit kann anzeigen, wenn gesetzt, dass der entsprechende Knoten eine Kopie des Cacheblockes besitzt, und kann anzeigen, wenn nicht gesetzt, dass der entsprechende Knoten keine Kopie des Cacheblockes besitzt. Andere Ausführungsbeispiele können die gesetzt- und nicht gesetzt- Bedeutungen des Bits umkehren oder andere Anzeigen verwenden.

[0123] Übergehend nun zu [Fig. 12](#) ist ein Flussdiagramm gezeigt, welches den Arbeitsablauf eines Ausführungsbeispiels der RLD-Steuerschaltung **192** und des entfernten Leitungsverzeichnisses **34** in Erwiderung auf einen Vorgang auf der Verbindung **22** erläutert. Während die in [Fig. 12](#) gezeigten Blöcke in einer bestimmten Reihenfolge dargestellt sind, können andere Reihenfolgen verwendet werden. Darüber hinaus können einige Blöcke von kombinatorischen Logikschaltungen in der RLD-Steuerschaltung **192** parallel ausgeführt werden. Z.B. können Entscheidungsblöcke **250**, **252**, **253**, **254** und **256** in einigen Ausführungsbeispielen parallel durchgeführt werden. Zusätzlich können Blöcke über einen oder mehrere Taktzyklen über eine Pipeline geleitet werden. Z.B. kann die Aktualisierung des Eintrags über eine Pipeline von den anderen Arbeitsvorgängen geleitet werden.

[0124] Die RLD-Steuerschaltung **192** kann ermitteln, ob der Vorgang einen Zugriff darstellt, für welchen der Knoten **10** der Heimatknoten ist (Entscheidungsblock **250**). Z.B. kann die RLD-Steuerschaltung **192** ermitteln, ob die Adresse des Vorgangs ein Teil des zu dem Knoten **10** zugeordneten Teils des entfernten kohärenten Platzes **48** ist. Die RLD-Steuerschaltung **192** kann z.B. die Knotennummer aus dem Steuerregister **38** mit der Knotennummer aus der Vorgangsadresse vergleichen. Falls der Vorgang nicht auf einen Cacheblock zugreift, für welchen der Knoten **10** der Heimatknoten ist, ist die RLD-Steuerschaltung **192**/das entfernte Leitungsverzeichnis **34** vielleicht nicht in den Vorgang involviert. Falls der Vorgang einen Zugriff auf einen Cacheblock darstellt, für welchen der Knoten der Heimatknoten ist (Entscheidungsblock **250**–ja-Zweig) und das entfernte Leitungsverzeichnis **34** einen Treffer anzeigt (Entscheidungsblock **252** – ja-Zweig), kann die RLD-Steuerschaltung **192** ermitteln, ob Prüfbefehle benötigt werden (Entscheidungsblock **256**). Prüfbe-

fehle können z.B. benötigt werden, wenn der Vorgang ausschließliches Eigentumsrecht an dem Cacheblock beansprucht und es einen oder mehrere entfernte Knoten mit einer Kopie des Cacheblockes gibt; oder falls der Vorgang mehrfach benutztes Eigentumsrecht an dem Cacheblock beansprucht und es eine modifizierte Kopie des Cacheblockes in einem entfernten Knoten gibt; oder wenn der Vorgang Kopien des Cacheblockes für ungültig erklärt und es mehrfach benutzte oder modifizierte Kopien des Cacheblockes in entfernten Knoten gibt. Falls Prüfbefehle benötigt werden, kann die RLD-Steuerschaltung **192** Information für das Erzeugen der Prüfbefehle für den Puffer für ausgehende Prüfbefehle **200A** liefern (Block **258**). In beiden Fällen kann die RLD-Steuerschaltung **192** den treffenden Eintrag aktualisieren, um wiederzuspiegeln, dass die Kopie des Cacheblockes an den entfernten Knoten übertragen wird, und um jeden entfernten Knoten zu löschen, für welchen Prüfbefehle erzeugt wurden (Block **260**).

[0125] Falls der Vorgang einen Zugriff auf einen Cacheblock darstellt, für welchen der Knoten der Heimatknoten ist, und das entfernte Leitungsverzeichnis **34** keinen Treffer anzeigt (Entscheidungsblock **250** – ja-Zweig, und Entscheidungsblock **252** – nein-Zweig), kann die RLD-Steuerschaltung **192** ermitteln, ob der Vorgang einen entfernten Zugriff darstellt (Entscheidungsblock **253**). D.h., die RLD-Steuerschaltung **192** kann ermitteln, ob der Vorgang aus einem Kohärenzbefehl von einem entfernten Knoten hervorgeht. Die RLD-Steuerschaltung **192** kann Information von der Eingangs-Steuerschaltung **190** (z.B. die Ursprungs-Knotennummer), die Adresse des Vorgangs und/oder andere Vorgangsinformation (z.B. die Befehlscodierung oder eine Markierung, welche den Ursprung des Vorgangs, wie die Speicherbrücke **32** oder einen anderen lokalen Agenten anzeigt) in verschiedenen Ausführungsbeispielen empfangen, um das Ermitteln durchzuführen. Falls der Vorgang keinen entfernten Zugriff darstellt (Entscheidungsblock **253** – nein-Zweig), dann wird das entfernte Leitungsverzeichnis **34** vielleicht nicht aktualisiert. Falls der Vorgang einen entfernten Zugriff darstellt (Entscheidungsblock **253** – ja-Zweig) und der ausgewählte Eintrag voll ist (d.h., aktuell den Zustand eines Cacheblocks in den entfernten Knoten überwacht) (Entscheidungsblock **254** – ja-Zweig), dann kann das entfernte Leitungsverzeichnis **34** den Opfer-Cacheblock aus dem ausgewählten Eintrag aussondern (Block **262**). Das entfernte Leitungsverzeichnis **34** kann die Zustandsinformation für den Opfer-Cacheblock an den Puffer für ausgehende Prüfbefehle **200A** liefern und die RLD-Steuerschaltung **192** kann den Puffer für ausgehende Prüfbefehle **200A** anzeigen, einen Eintrag für die Opfer-Prüfbefehle zu reservieren (Block **264**). Die RLD-Steuerschaltung **192** kann den Eintrag mit der Information für den Cacheblock (Adresse, modifiziert-Anzeige, den Ursprungsknoten anzeigenden Knotenvektor)

aktualisieren (Block **260**). Auf der anderen Seite kann, wenn der ausgewählte Eintrag nicht voll ist (Entscheidungsblock **254** – nein-Zweig), das entfernte Leitungsverzeichnis **34** den verfügbaren Eintrag reservieren (Block **266**). Die RLD-Steuerschaltung **192** kann den Eintrag mit der Information für den Cacheblock (Adresse, modifiziert-Anzeige, den Ursprungsknoten anzeigenden Knotenvektor) aktualisieren (Block **260**).

Knotenstatus speicherndes L2 Cache

[0126] Das entfernte Leitungsverzeichnis **34** verfolgt den Zustand der Heimatknoten-Cacheblöcke in entfernten Knoten. D.h., die knotenübergreifende Kohärenz hält einen Zustand für jeden Knoten aufrecht. Jedoch kann ein bestimmter Knoten **10** mehrere kohärente Agenten (z.B. Prozessoren **12A–12N**, das L2 Cache **36**, etc.) einschließen. Die kohärenten Agenten können Caches einschließen, welche entfernte Cacheblöcke in verschiedenen Zuständen speichern. In einem Ausführungsbeispiel kann der Knoten **10** bestimmen, dass der Zustand des Knotens für den entfernten Cacheblock (wie in dem Heimatknoten des entfernten Cacheblockes aufgezeichnet) in dem L2 Cache **36** gespeichert wird. D.h., der Zustand des L2 Caches **36** kann den Zustand des entfernten Cacheblockes in den entfernten Knoten, welcher das L2 Cache **36** einschließt, widerspiegeln. Andere Caches können andere Zustände für den Cacheblock aufweisen (konsistent mit dem Zustand des L2 Caches **36**).

[0127] Dadurch, dass der Knotenzustand (wie in dem Heimatknoten aufgezeichnet) des entfernten Cacheblocks in dem L2 Cache **36** gespeichert wird, kann die Verwaltung des Knotenzustands vereinfacht werden. Andere kohärente Agenten in dem Knoten **10** können in einigen Ausführungsbeispielen entfernte Cacheblöcke nicht anders behandeln als lokale Cacheblöcke. Die kohärenten Agenten können sowohl lokale, als auch entfernte Cacheblöcke innerhalb des Knotens kohärent reservieren und freigeben. Die kohärenten Agenten können mehrfach benutzte Kopien eines entfernten Cacheblocks innerhalb des Knotens **10** erhalten, auch wenn der Knoten **10** eine modifizierte Kopie des Cacheblockes in dem knotenübergreifenden Kohärenzsystem besitzt. Die kohärenten Agenten können den entfernten Cacheblock modifizieren und dann den entfernten Cacheblock in derselben Weise wie einen lokalen Cacheblock aussondern. Die lokalen Cacheblock-Modifikationen können in dem L2 Cache **36** und/oder der Speichersteuerung **14** erfasst werden. Die entfernten Cacheblock-Modifikationen können von dem L2 Cache **36** erfasst werden.

[0128] In einem Ausführungsbeispiel kann das knotenübergreifende Kohärenzsystem das stillschweigende Aussondern von mehrfach benutzten entfernten Cacheblöcken ermöglichen (z.B. das Aussondern

eines mehrfach benutzten entfernten Cacheblockes aus den entfernten Knoten, ohne den Heimatknoten zu informieren). Das L2 Cache **36** kann den Knotenzustand für entfernte Cacheblöcke speichern, welche ausschließlich besessen werden (z.B. in dem vorliegenden Ausführungsbeispiel der modifizierte Zustand), und mehrfach benutzte entfernte Cacheblöcke zuordnen, wenn sie in den Knoten übertragen werden, und es kann jedoch einen mehrfach benutzten entfernten Cacheblock aussondern, ohne sicherzustellen, dass sie von dem Knoten **10** ausgesondert werden und ohne den Heimatknoten des entfernten Cacheblockes zu informieren. In derartigen Ausführungsbeispielen kann das L2 Cache **36** als ausschließliche (modifizierte) entfernte Cacheblöcke einschließend betrachtet werden.

[0129] Es wird erwähnt, dass in einigen Ausführungsbeispielen das L2 Cache **36** programmierbar sein kann, um einen oder mehrere Wege für entfernte Cacheblöcke zu reservieren (und kann die verbleibenden Wege für lokale Cacheblöcke verwenden).

[0130] Übergehend nun zu [Fig. 13](#) ist ein Blockdiagramm gezeigt, welches den Zustand eines entfernten Cacheblockes in dem Knoten **10** für verschiedene Knotenzustände erläutert, welche für den Knoten **10** an dem Heimatknoten gemäß eines Ausführungsbeispiels aufgezeichnet werden.

[0131] Oben in [Fig. 13](#) ist der Zustand des entfernten Cacheblockes (Block A) in dem Knoten **10** für einen Knotenzustand gezeigt, welcher in dem Heimatknoten für den Knoten **10** als modifiziert aufgezeichnet ist. In diesem Fall wird der Block A in den L2 Cache **36** mit dem modifizierten Zustand (dem Zustand des entfernten Blockes, welcher für den Knoten **10** in dem Heimatknoten aufgezeichnet ist) gespeichert. Verschiedene andere Caches (z.B. die L1 Caches **270**, welche Teil der Prozessoren **12A-12N** sein können) können den Block A in jedem Zustand aufweisen. Z.B. können in Ausführungsformen, welche das MESI Kohärenzprotokoll für knoteninterne Kohärenz verwenden, der Zustand irgendeiner der Zustände modifiziert, ausschließlich, mehrfach benutzt, oder ungültig in jedem der verschiedenen Caches **270** sein.

[0132] In der Mitte von [Fig. 13](#) ist der Zustand des entfernten Cacheblockes (Block A) in dem Knoten **10** für einen Knotenzustand gezeigt, welcher in dem Heimatknoten für den Knoten **10** als mehrfach benutzt aufgezeichnet ist. In diesem Fall kann der Block A entweder mehrfach benutzt in dem L2 Cache **36** sein oder ungültig sein. Der Block A kann dem L2 Cache **36** zugeordnet werden, wenn der Block A von dem Heimatknoten zu dem Knoten **10** geholt wird, kann aber zu einem späteren Zeitpunkt stillschweigend ausgesondert werden. In ähnlicher Weise können verschiedene andere Caches (z.B. die L1 Caches

270) auch Block A in entweder dem mehrfach benutzten oder dem ungültigen Zustand aufweisen.

[0133] Unten in [Fig. 13](#) ist der Zustand des entfernten Cacheblockes (Block A) in dem Knoten **10** für einen Knotenzustand gezeigt, der in dem Heimatknoten für den Knoten **10** als ungültig aufgezeichnet ist. In diesem Fall haben sowohl das L2 Cache **36** und die verschiedenen anderen Caches **270** den Block A in dem ungültigen Zustand. Zusätzlich können sowohl das L2 Cache **36**, als auch die verschiedenen anderen Caches **270** den Block A in dem ungültigen Zustand aufweisen, falls der Block A nicht in dem entfernten Leitungsverzeichnis **34** in dem Heimatknoten aufgezeichnet ist (für Ausführungsbeispiele, in welchen das entfernte Leitungsverzeichnis **34** nicht jeden der Cacheblöcke für den Heimatknoten verfolgt).

[0134] Es wird bemerkt, dass ein Cache, welches den Block in dem ungültigen Zustand aufweist, sich auf den Cache-Ort welcher zuvor den als ungültig gekennzeichneten Block speichert, sowie auf keinen zu dem Block gehörigen Cache-Ort beziehen. Es wird ferner erwähnt, dass andere Ausführungsbeispiele andere Kohärenzprotokolle als das MESI Protokoll für die knoteninterne Kohärenz und das MSI Protokoll für die knotenübergreifende Kohärenz einsetzen können.

[0135] Im allgemeinen können die verschiedenen Caches in jedem der in [Fig. 13](#) gezeigten Fälle einen mit dem Zustand des L2 Caches **36** konsistenten Zustand für entfernte Cacheblöcke einnehmen. Ein erster Zustand kann mit einem zweiten Zustand konsistent sein, wenn der erste Zustand keine umfangreicheren Zugriffsrechte liefert als der zweite Zustand liefert (wobei Schreib-Zugriff umfangreicher ist als Lese-Zugriff und Lese-Zugriff umfangreicher ist als gar kein Zugriff).

[0136] Übergehend nun zu [Fig. 14](#), ist ein Flussdiagramm gezeigt, welches die Arbeitsweise eines Ausführungsbeispiels des L2 Caches **36** in Erwiderung auf einen Lesevorgang auf der Verbindung **22** erläutert. In dem Kontext von [Fig. 14](#) kann ein Lesevorgang sich entweder auf einem RdExc oder einem RdShd Vorgang beziehen. Die RdKill und RdInv Vorgänge können von der Speicherbrücke **32** in Erwiderung auf Prüfbefehle ausgegeben werden und das L2 Cache **36** kann sie als solche behandeln (z.B. indem sie den Cacheblock in dem L2 Cache **36** für ungültig erklärt und den Cacheblock in der Datenphase des RdKill/RdInv Vorgangs liefert, falls der Cacheblock in dem L2 Cache **36** modifiziert ist). Das L2 Cache **36** kann auf einen Schreibtrefter hin eine Aktualisierung durchführen, kann aber in diesem Ausführungsbeispiel nicht einen Ort für ein entferntes Schreiben reservieren. Stattdessen kann das Schreiben an den Heimatknoten übertragen werden, ohne in dem Knoten **10** in einen Cachespeicher abgelegt zu werden.

Die in [Fig. 14](#) dargestellten Blöcke sind für ein leichteres Verstehen in einer bestimmten Reihenfolge dargestellt, aber andere Reihenfolgen können auch verwendet werden. Darüber hinaus können Blöcke durch kombinatorische Logik in dem L2 Cache **36** parallel durchgeführt werden. Andere Blöcke können über zwei oder mehrere Taktzyklen über eine Pipeline geleitet werden, sofern gewünscht.

[0137] Falls der Lesevorgang nicht auf einen entfernten Cache-Block gerichtet ist (Entscheidungsblock **272** – nein-Zweig), kann das L2 Cache **36** regulär den Lesevorgang erwidern (Daten liefern, falls ein Treffer vorliegt, optional einen Cacheblock reservieren, falls kein Treffer vorliegt) (Block **274**). In einem Ausführungsbeispiel kann das L2 Cache **36** ermitteln, ob der Lesevorgang auf einen entfernten Cacheblock gerichtet ist, basierend auf der Knotennummer aus dem Konfigurationsregister **38** und dem wichtigsten Halbbyte der Adresse des Lesevorgangs. Andere Ausführungsbeispiele können entfernte Cacheblöcke auf andere Weise erkennen (z.B. unterschiedliche Vorgangskodierungen für entfernte gegenüber lokalen Vorgängen, andere Steuersignale, etc.). Falls der Lesevorgang auf einen entfernten Cacheblock gerichtet ist (Entscheidungsblock **272** – ja-Zweig) und einen Treffer in dem L2 Cache **36** darstellt (Entscheidungsblock **276** – ja-Zweig), kann das L2 Cache **36** auch regulär antworten (Block **274**). Ein RdExc Vorgang kann als ein Treffer angesehen werden, falls der Zustand in dem L2 Cache **36** zum Zweck von Entscheidungsblock **276** modifiziert ist, da, falls der Zustand mehrfach benutzt ist, der RdExc Vorgang einen cRdExc Befehl erzeugen kann, um eine ausschließliche Kopie des entfernten Cacheblockes zu erhalten.

[0138] Falls der Lesevorgang auf einen entfernten Block gerichtet ist (Entscheidungsblock **272** – ja-Zweig) und keinen Treffer in dem L2 Cache **36** darstellt (Entscheidungsblock **276** – nein-Zweig), kann das L2 Cache **36** einen Cacheeintrag für den Cacheblock reservieren (Block **278**) (obwohl für den Fall des RdExc, welcher mehrfach benutzt trifft, der existierende Cacheeintrag verwendet werden kann). Falls der Lesevorgang ein RdShd Vorgang ist (Entscheidungsblock **280** – ja-Zweig), kann das L2 Cache **36** den Zustand in dem Eintrag auf mehrfach benutzt setzen (Block **282**). Andernfalls ist der Lesevorgang ein RdExc Vorgang und das L2 Cache **36** kann den Zustand in dem Eintrag auf modifiziert setzen (Block **284**).

[0139] [Fig. 15](#) zeigt ein Flussdiagramm, welches die Arbeitsweise eines Ausführungsbeispiels des L2 Cache **36** während eines Aussonderns eines Cacheblockes erläutert (z.B. wegen einer Zuordnung des Eintrags, welcher den Cacheblock speichert, zu einem anderen Cacheblock). Die in [Fig. 15](#) gezeigten Blöcke sind für ein einfaches Verstehen in einer

bestimmten Reihenfolge erläutert, aber andere Reihenfolgen können verwendet werden. Darüber hinaus können Blöcke durch eine kombinatorische Logik in dem L2 Cache **36** parallel durchgeführt werden. Andere Blöcke können über zwei oder mehrere Taktzyklen über eine Pipeline geleitet werden, sofern gewünscht.

[0140] Falls der Opferblock nicht ein entfernter Cacheblock ist (Entscheidungsblock **290** – nein-Zweig), kann das L2 Cache **36** den Opferblock regulär aussondern (Block **292**). Z.B. kann das L2 Cache **36** einen Wr-Vorgang erzeugen, um den Opferblock in den Speicher **24** zu schreiben. Insbesondere kann das L2 Cache **36** nicht versuchen, den Speicherblock aus den verschiedenen anderen Caches **270** aussondern zu lassen. Wie oben erwähnt, kann es eine Zahl von Wegen geben, wie das L2 Cache **36** ermitteln kann, ob ein Block entfernt ist oder nicht. Z.B. kann das wichtigste Halbbyte der Adresse (wie in dem Konfigurationsregister **38** mit der Knotennummer verglichen) verwendet werden. In anderen Ausführungsbeispielen können andere Mechanismen verwendet werden (z.B. der entfernte/lokale Zustand des Cacheblockes kann in dem L2 Cache **36** gespeichert werden).

[0141] Falls der Opferblock ein entfernter Cacheblock ist (Entscheidungsblock **290** – ja-Zweig) und der entfernte Cacheblock in dem L2 Cache **36** nicht in dem modifizierten Zustand vorliegt (Entscheidungsblock **294** – nein-Zweig), kann das L2 Cache **36** den Block ohne weitere Maßnahmen aussondern (Block **296**). Falls der Opferblock ein entfernter Cacheblock ist (Entscheidungsblock **290** – ja-Zweig) und der entfernte Cacheblock in dem L2 Cache **36** in dem modifizierten Zustand vorliegt (Entscheidungsblock **294** – ja-Zweig), kann das L2 Cache **36** einen WrFlush Vorgang erzeugen, um den Opferblock zurück in den Heimatknoten zu schreiben (Block **298**). Der WrFlush Vorgang kann einen modifizierten Eigentümer in dem Knoten **10** veranlassen, die Daten für den WrFlush Vorgang anstelle des L2 Caches **36** zu liefern. Falls es keinen anderen modifizierten Eigentümer in dem Knoten **10** gibt, liefert das L2 Cache **36** die Daten. Folglich, falls ein kohärenter Agent eine aktuellere Kopie des Cacheblockes besitzt (z.B. in den Caches **270**), wird die aktuellere Kopie geliefert. Falls keine aktuellere Kopie existiert, liefert das L2 Cache **36** den Cacheblock.

[0142] Es wird erwähnt, dass, während die obige Erläuterung unter Bezugnahme auf [Fig. 1–Fig. 15](#) und [Fig. 17](#) sich auf einen Cacheblock als die Granularität bezogen haben, auf welcher Kohärenz aufrecht erhalten wird, können andere Ausführungsbeispiele verschiedene Granularitäten, welche einen größeren oder kleineren Umfang als ein Cacheblock aufweisen, verwendet werden. In derartigen Ausführungsbeispielen kann ein „Kohärenzblock“ in der gleichen Weise behandelt werden wie ein „Cacheblock“ oder

„Block“, wenn oben Kohärenz erläutert wird. Im allgemeinen kann ein „Kohärenzblock“ einen Satz von zusammenhängenden (im Speicher) Bytes einschließen, welche für Kohärenzzwecke als eine Einheit behandelt werden. In verschiedenen Ausführungsbeispielen kann ein Kohärenzblock einen Teil eines Cacheblockes, einen einzelnen Cacheblock oder mehrere Cacheblöcke aufweisen, sofern gewünscht.

Computer-zugängliches Medium

[0143] Übergehend zu [Fig. 16](#), ist ein Blockdiagramm eines Computer-zugänglichen Mediums **300** gezeigt, welches eine oder mehrere Datenstrukturen stellvertretend für die in dem Knoten **10** und/oder dem System **40** eingeschlossene Schaltung aufweist, dargestellt. Allgemein gesprochen kann ein Computer-zugängliches Medium Speichermedien, wie magnetische oder optische Medien, z.B. eine Diskette, eine CD-Rom oder eine DVD-Rom, flüchtige oder nicht-flüchtige Speichermedien, wie RAM (z.B. SDRAM, RDRAM, SRAM, etc.) ROM, etc., sowie Medien einschließen, welche über Übertragungsmedien oder Signale, wie elektrische, elektromagnetische oder digitale Signale, welche über ein Kommunikationsmedium, wie ein Netzwerk und/oder eine drahtlose Verbindung transportiert werden.

[0144] Im allgemeinen wird/werden die Datenstrukturen der Schaltung auf dem Computerzugänglichen Medium **300** von einem Programm gelesen und direkt oder indirekt verwendet, um die Hardware, welche die Schaltung aufweist, herzustellen. Z.B. kann/können die Datenstrukturen ein oder mehr Verhaltens-Level-Beschreibungen oder Register-Übertragungs-Level (RTL = Register Transfer Level)-Beschreibungen der Hardwarefunktionsweise in einer Hoch-Designsprache (HDL = High Level Design Language), wie Verilog oder VHDL, einschließen. Die Beschreibungen kann/können von Synthesewerkzeugen gelesen werden, welche die Beschreibung synthetisieren können, um eine oder mehrere Netzliste(n) zu erzeugen, welche Listen von Gattern von einer synthetischen Bibliothek aufweisen. Die Netzliste(n) weis/weisen einen Gattersatz auf, welcher auch die Funktionsweise der die Schaltung aufweisenden Hardware darstellt. Die Netzliste(n) kann/können dann angeordnet und weitergeleitet werden, um einen oder mehrere Datensätze zu erzeugen, welche die auf Masken aufzubringenden geometrischen Formen beschreiben. Die Masken können dann in verschiedenen Halbleiterfabrikationsschritten verwendet werden, um eine Halbleiterschaltung oder -schaltungen herzustellen, die der Schaltung entsprechen. Alternativ kann/können die Datenstrukturen auf einem Computerzugänglichen Medium **300** die Netzliste(n) sein (mit oder ohne der Synthesebibliothek) oder der Datensatz/die Datensätze, sofern gewünscht. In einer wiederum anderen Alternative können die Datenstrukturen eine Ausgabe ei-

nes schematischen Programmes, oder einer Netzliste(n) oder eines Datensatzes/Datensätze aufweisen, welche davon abgeleitet sind. Die Datenstrukturen kann/können eine Schaltung aufweisen, welche einen Mechanismus für den Knoten **10** bestimmen.

[0145] Während das Computer-zugängliche Medium **300** eine Darstellung des Knotens **10** und/oder des Systems **40** einschließt, können andere Ausführungsbeispiele eine Darstellung von jedem Teil des Knotens **10** und/oder des Systems **40** einschließen (z.B. Prozessoren **12A–12N**, Speichersteuerung **14**, L2 Cache **36**, Verbindung **22**, Speicherbrücke **32** (oder Teile davon), entferntes Leitungsverzeichnis **34**, Schalter **18**, Schnittstellenschaltungen **22A–22C** (oder Teile davon), Knoten **10A–10D** (oder Teile davon), etc.).

[0146] Zahlreiche Variationen und Abänderungen sind für den Fachmann offensichtlich, sobald die obige Offenbarung vollständig verstanden wird. Es ist beabsichtigt, dass die folgenden Ansprüche ausgelegt werden, um alle derartigen Variationen und Modifikationen zu umfassen.

Patentansprüche

1. System, umfassend einen ersten Knoten (**10**) und einen zweiten Knoten (**10**); der erste Knoten (**10**) umfassend:
eine Vielzahl von kohärenten Agenten (**16**);
eine Speicherbrücke (**32**); und
ein erstes Cache (**36**);
worin das erste Cache für die Vielzahl von kohärenten Agenten (**16**) zugänglich ist;
worin die Speicherbrücke (**32**) mit der Vielzahl von kohärenten Agenten (**16**) verbunden und dafür vorgesehen ist, einen ersten Kohärenzblock von einem mit dem zweiten Knoten (**10**) verbundenem Speicher (**24**) in Erwiderung auf eine Transaktion abzurufen, die von einem ersten kohärenten Agenten aus der Vielzahl von kohärenten Agenten (**16**) erzeugt wurde, **dadurch gekennzeichnet**, dass
der zweite Knoten (**10**) dafür vorgesehen ist, einen Zustand aufzuzeichnen, in welchem der erste Kohärenzblock an den ersten Knoten in Erwiderung auf den Abruf von der Speicherbrücke (**32**) geliefert wurde, und
das erste Cache (**36**) dafür angepasst ist, den Zustand festzuhalten, in welchem der erste Kohärenzblock an den ersten Knoten (**10**) geliefert wurde, wobei der Zustand einen Eigentumslevel umfasst, welcher eine Zugangserlaubnis anzeigt, die der erste Knoten (**10**) für den ersten Kohärenzblock besitzt.

2. System gemäß Anspruch 1, worin zumindest einige von der Vielzahl von kohärenten Agenten (**16**) Caches einschließen und worin die Caches in der Vielzahl von kohärenten Agenten (**16**) dazu geeignet sind, den ersten Kohärenzblock in anderen Zustän-

den als dem Zustand in dem ersten Cache zu speichern.

3. System gemäß Anspruch 1 oder 2, worin das erste Cache (36) Kohärenzblöcke einschließt, die von dem Knoten (10) entfernt liegen und zu welchen der Knoten (10) alleinigen Zugriff besitzt.

4. System gemäß Anspruch 3, worin das erste Cache (36) einen modifizierten Zustand für die Kohärenzblöcke besitzt, die von dem Knoten (10) entfernt liegen und zu welchen der Knoten (10) alleinigen Zugriff besitzt.

5. System gemäß einem der vorhergehenden Ansprüche, ferner eine Verbindung (22) umfassend, an welche die Vielzahl von kohärenten Agenten (16), die Speicherbrücke (32) und das erste Cache (36) gekoppelt sind.

6. System gemäß Anspruch 5, ferner umfassend ein oder mehrere Schnittstellenleitungen (20A, 20B, 20C), um die Schnittstellen mit anderen Knoten zu verbinden, worin die Speicherbrücke (32) mit den Schnittstellenleitungen (20A, 20B, 20C) getrennt von der Verbindung verbunden ist.

7. System gemäß einem der vorhergehenden Ansprüche, worin, falls das erste Cache den ersten Kohärenzblock aussondert und der Zustand in dem ersten Cache für den ersten Kohärenzblock modifiziert wird, das erste Cache dafür vorgesehen ist, zu veranlassen, dass der erste Kohärenzblock aus der Vielzahl von kohärenten Agenten (16) ausgesondert wird und dass der erste Kohärenzblock an den zweiten Knoten (10) zurückgesendet wird.

8. System gemäß Anspruch 7, ferner umfassend eine Speichersteuerung (14), die mit einem zweiten Speicher (24) verbunden ist, worin, falls das erste Cache (36) einen zweiten Kohärenzblock des zweiten Speichers (24) aus dem ersten Cache (36) aussondert, das erste Cache dafür vorgesehen ist, nicht zu veranlassen, dass der zweite Kohärenzblock aus der Vielzahl von kohärenten Agenten (16) ausgesondert wird.

9. System gemäß Anspruch 7, worin, falls der Zustand des ersten Kohärenzblockes in dem ersten Cache (36) nicht verändert wird und das erste Cache (36) den ersten Kohärenzblock aussondert, ist das erste Cache (36) dafür vorgesehen, nicht zu veranlassen, dass der erste Kohärenzblock aus der Vielzahl von kohärenten Agenten (16) ausgesondert wird und dass der erste Kohärenzblock an den zweiten Knoten (10) zurückgesendet wird.

10. System gemäß Anspruch 1, ferner umfassend:

– eine Vielzahl von Caches ersten Levels, dafür vor-

gesehen, Cacheblocks und zugehörige Zustände zu speichern; und

– ein Cache zweiten Levels, dafür vorgesehen, Cacheblöcke und zugehörige Zustände zu speichern, wobei ein entfernt liegender Cacheblock in dem Cache zweiten Levels gespeichert wird und der zugehörige Zustand in dem Cache zweiten Levels der Zustand ist, welcher von einem Heimatknoten des für den Knoten (10) entfernt liegenden Cacheblockes aufgezeichnet wurde.

11. Verfahren, umfassend:

Abrufen eines ersten Kohärenzblockes in einen ersten Knoten (10) von einem zweiten Knoten (10), wobei der erste Knoten eine Vielzahl von kohärenten Agenten (16), eine Speicherbrücke (32) und ein erstes Cache (36) umfasst, wobei die Speicherbrücke (32) mit der Vielzahl von kohärenten Agenten (16) verbunden ist, das erste Cache (36) für die Vielzahl von kohärenten Agenten (16) zugänglich ist, und das Abrufen durch die Speicherbrücke (32) von einem Speicher (24), der mit dem zweiten Knoten (10) verbunden ist, in Erwiderung auf eine Transaktion durchgeführt wird, die von dem ersten kohärenten Agenten aus der Vielzahl von kohärenten Agenten (16) erzeugt wurde, dadurch gekennzeichnet, dass das Verfahren ferner umfasst:

Aufzeichnen eines Zustands, in welchem der erste Kohärenzblock an den ersten Knoten (10) in Erwiderung auf den Abruf von der Speicherbrücke (32) geliefert wird, in dem zweiten Knoten (10), und Festhalten des Zustandes, in welchem der erste Kohärenzblock an den ersten Knoten (10) geliefert wird, in dem ersten Cache (36), wobei der Zustand einen Eigentumslevel umfasst, welcher eine Zugangserlaubnis anzeigt, die der erste Knoten (10) für den ersten Kohärenzblock besitzt.

12. Verfahren nach Anspruch 11, ferner umfassend:

Aussondern des ersten Kohärenzblockes aus dem ersten Cache (36); und Veranlassen des Aussonderns des ersten Kohärenzblockes aus anderen Caches innerhalb des zweiten Knotens (10) in Erwiderung auf das Verändern des Zustands in dem ersten Cache (36).

13. Verfahren nach Anspruch 12, ferner umfassend Zurücksenden des ersten Kohärenzblockes an den ersten Knoten in Erwiderung auf ein Verändern des Zustands in dem ersten Cache.

14. Verfahren nach Anspruch 12, ferner umfassend nicht Veranlassen des Aussonderns des ersten Kohärenzblockes aus anderen Caches innerhalb des zweiten Knotens (10) in Erwiderung auf kein Verändern des Zustands in dem ersten Cache (36).

15. Computer-zugängliches Medium (300), um-

fassend einen Programmcode zum Durchführen des Verfahrens gemäß einem der Ansprüche 11 bis 14, wenn er in ein Computersystem geladen wird.

Es folgen 17 Blatt Zeichnungen

Anhängende Zeichnungen

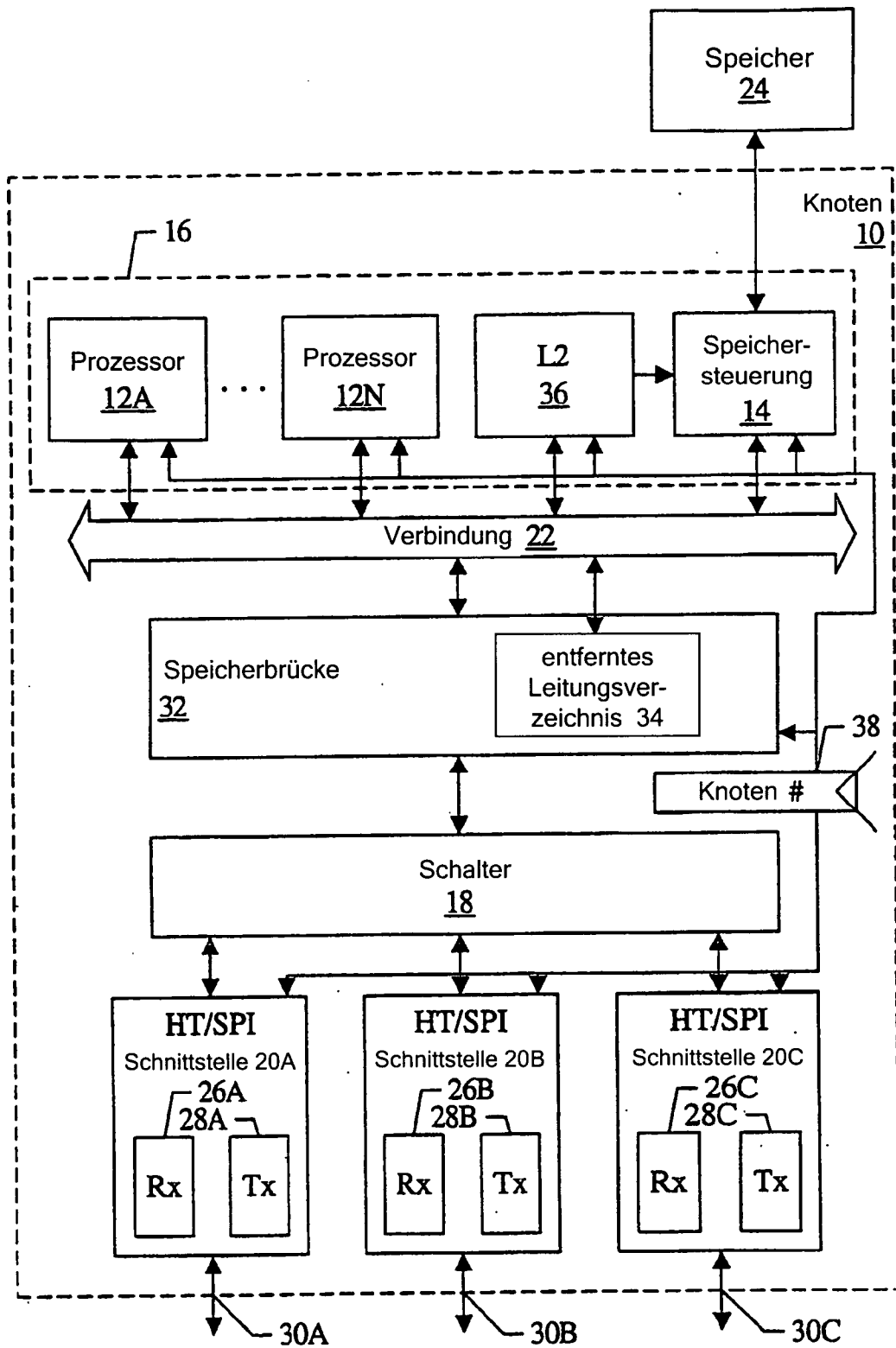


Fig. 1

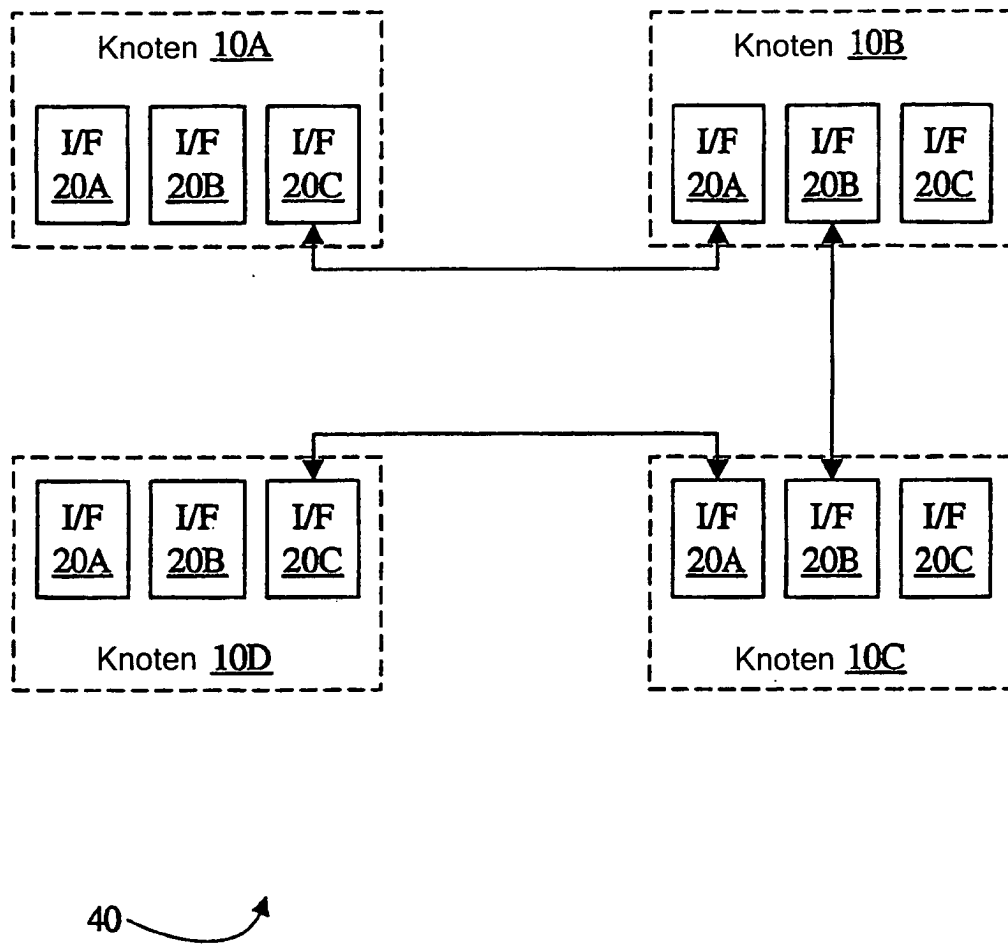


Fig. 2

Verbindungs-Vorgänge

<u>Vorgang</u>
RdShd
RdExc
Wr
WrInv
RdKill
RdInv
WrFlush
Nop

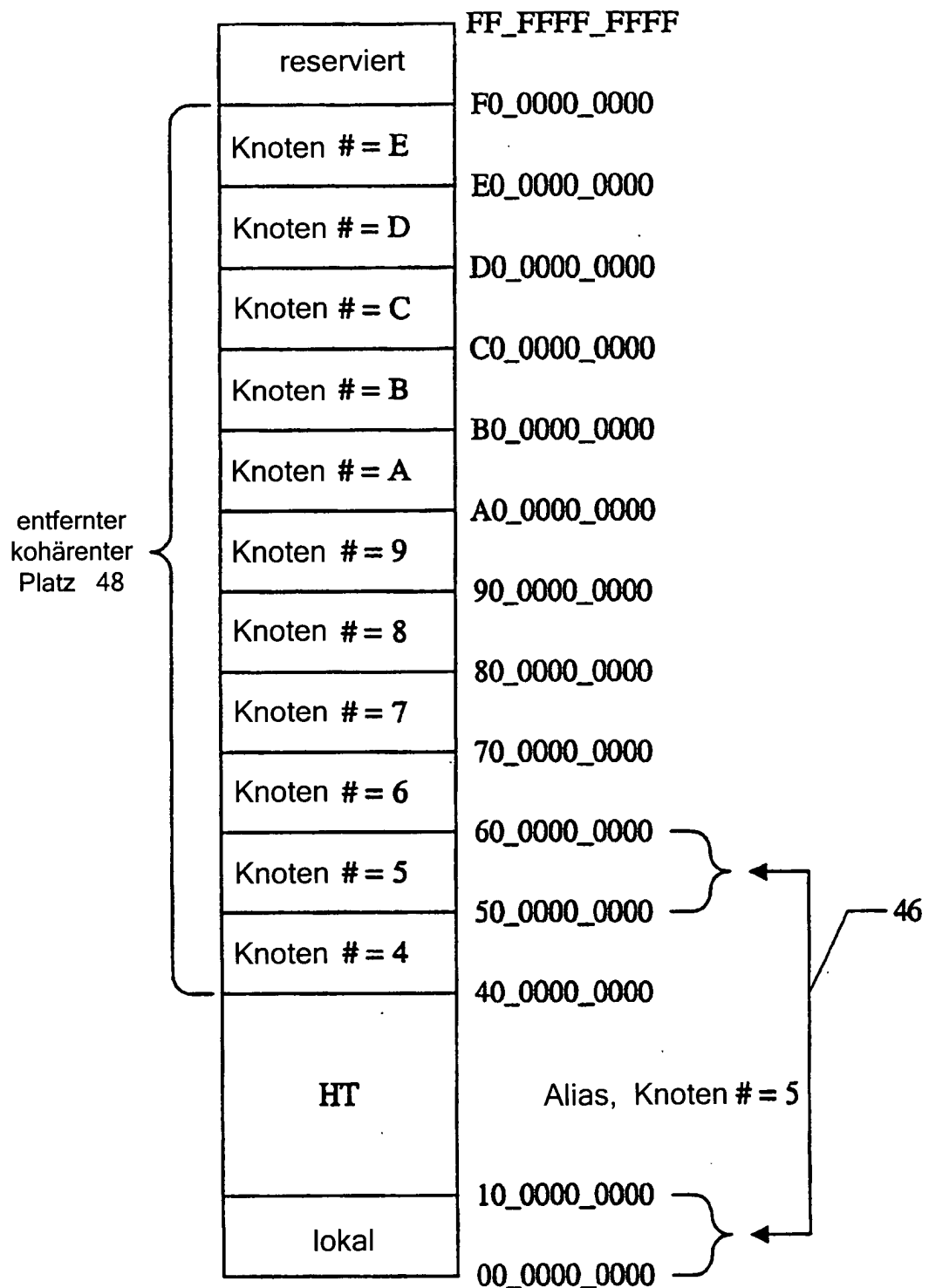
42

HTcc Befehle

<u>Befehl</u>	<u>virtueller Kanal</u>
cRdShd	CRd
cRdExc	CRd
Flush	Probe
Kill	Probe
Kill_Ack	Ack
WB	Ack
Fill	CFill

44

Fig. 3

Fig. 4

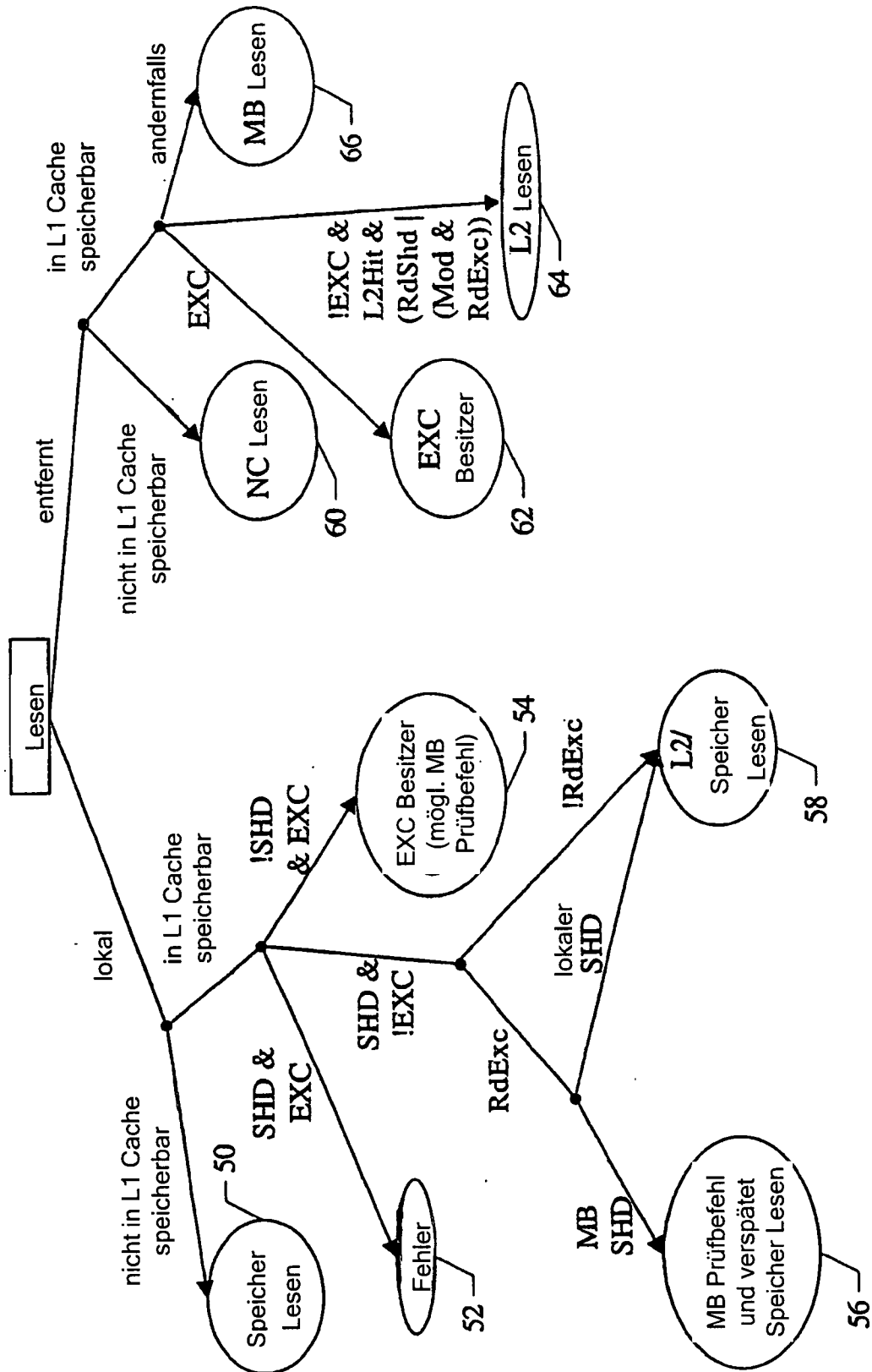


Fig. 5

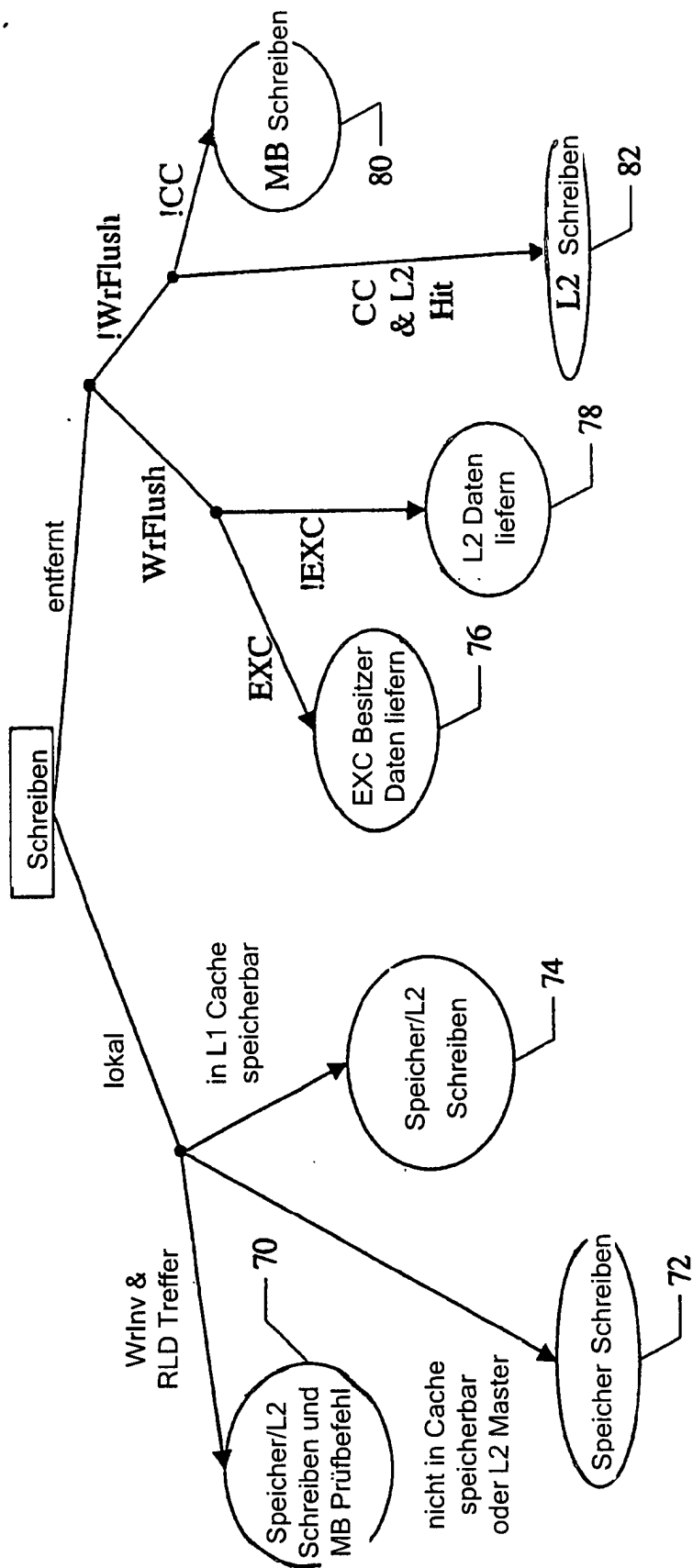


Fig. 6

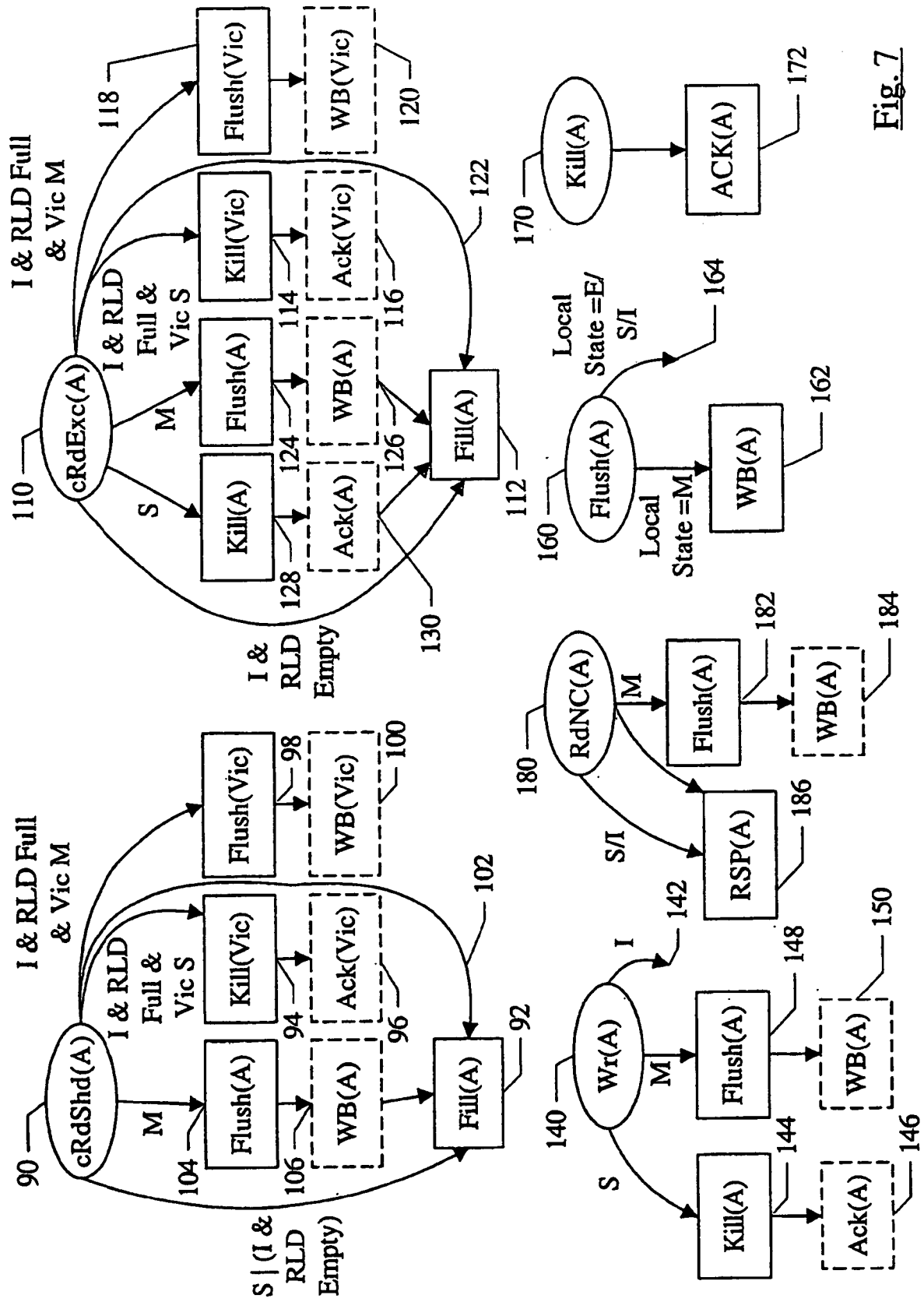


Fig. 7

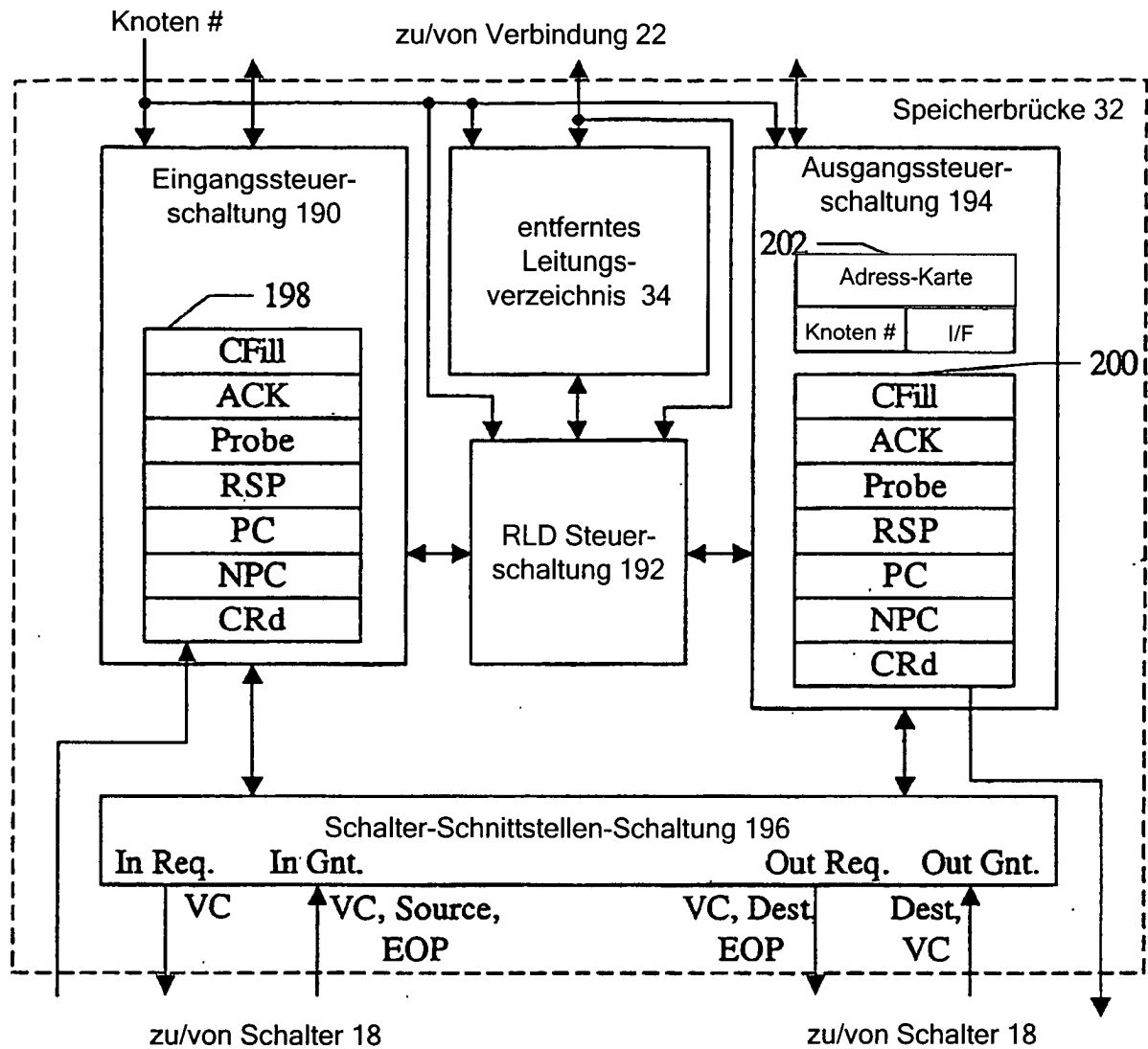


Fig. 8

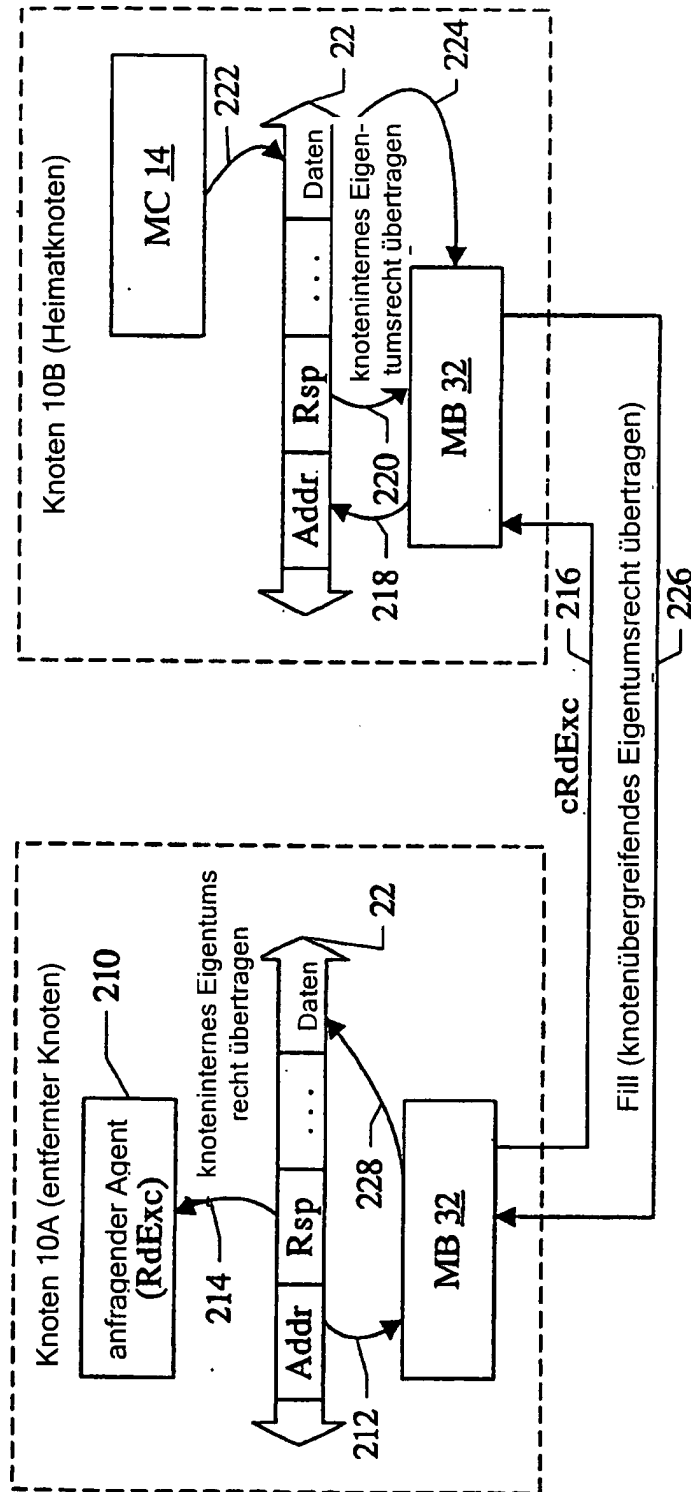
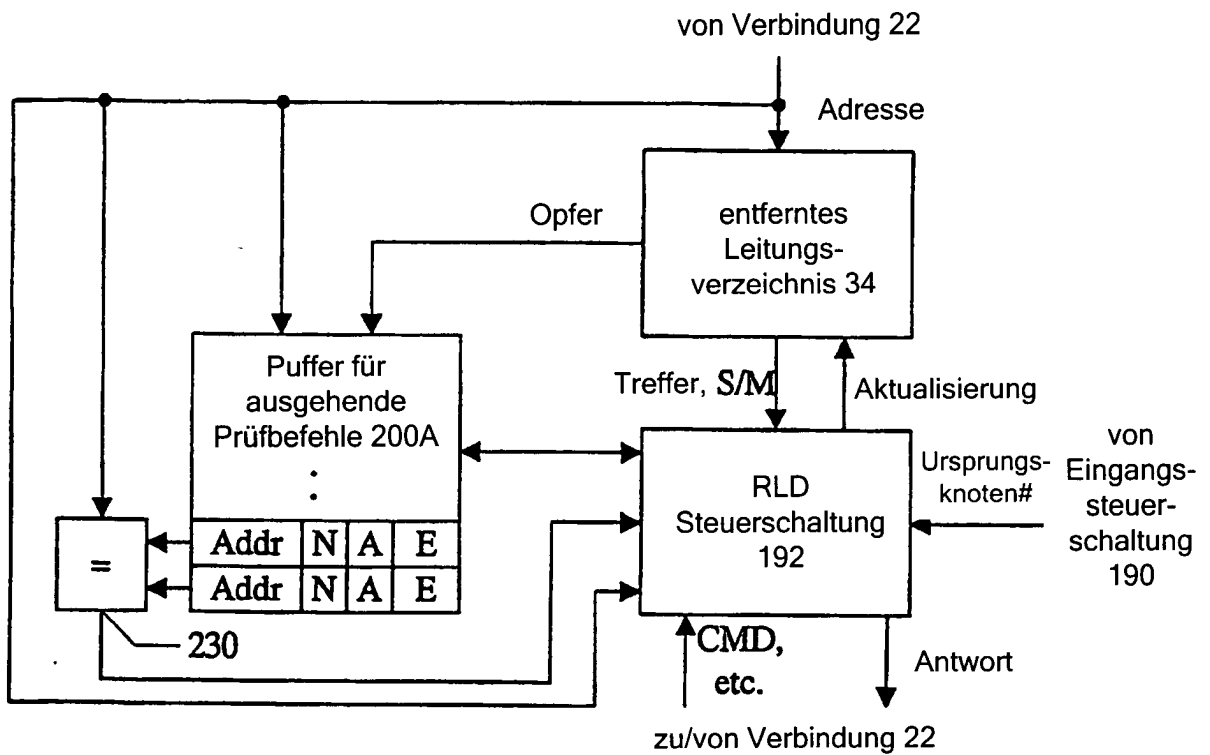


Fig. 9

**Fig. 10**

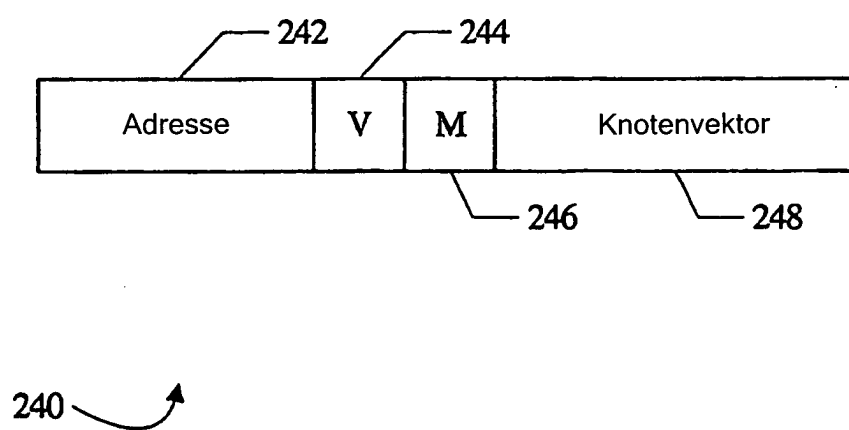
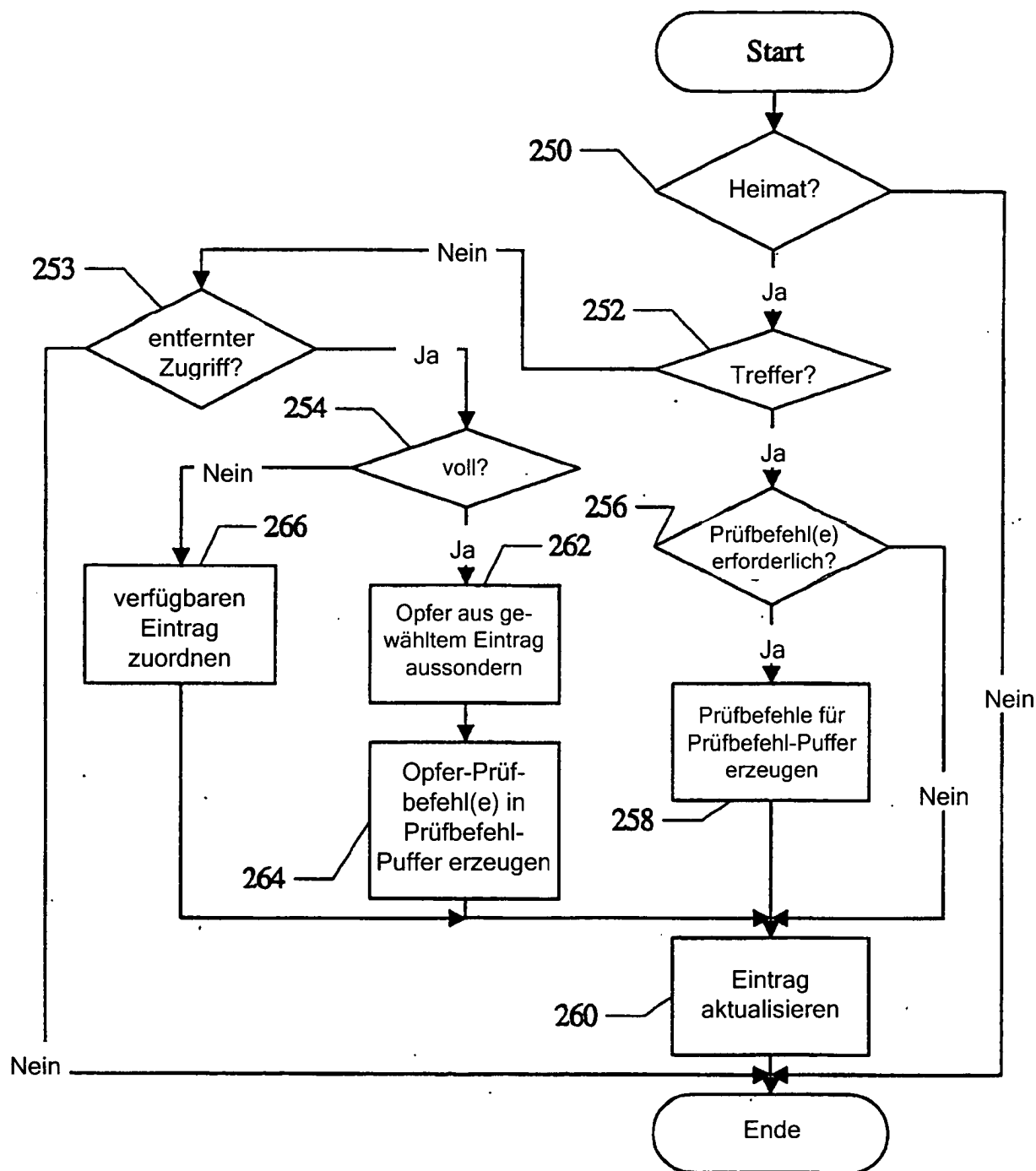


Fig. 11

Fig. 12

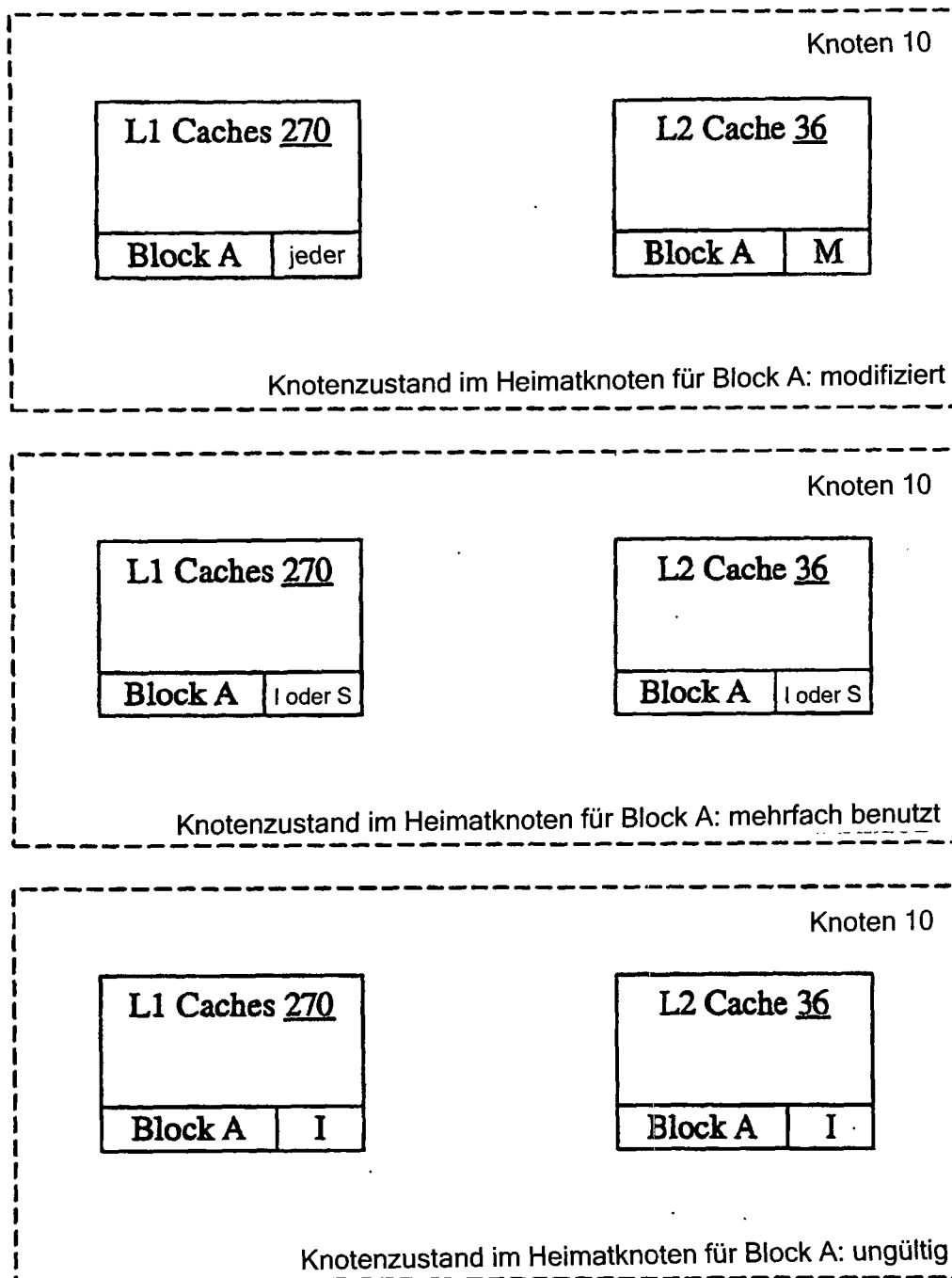
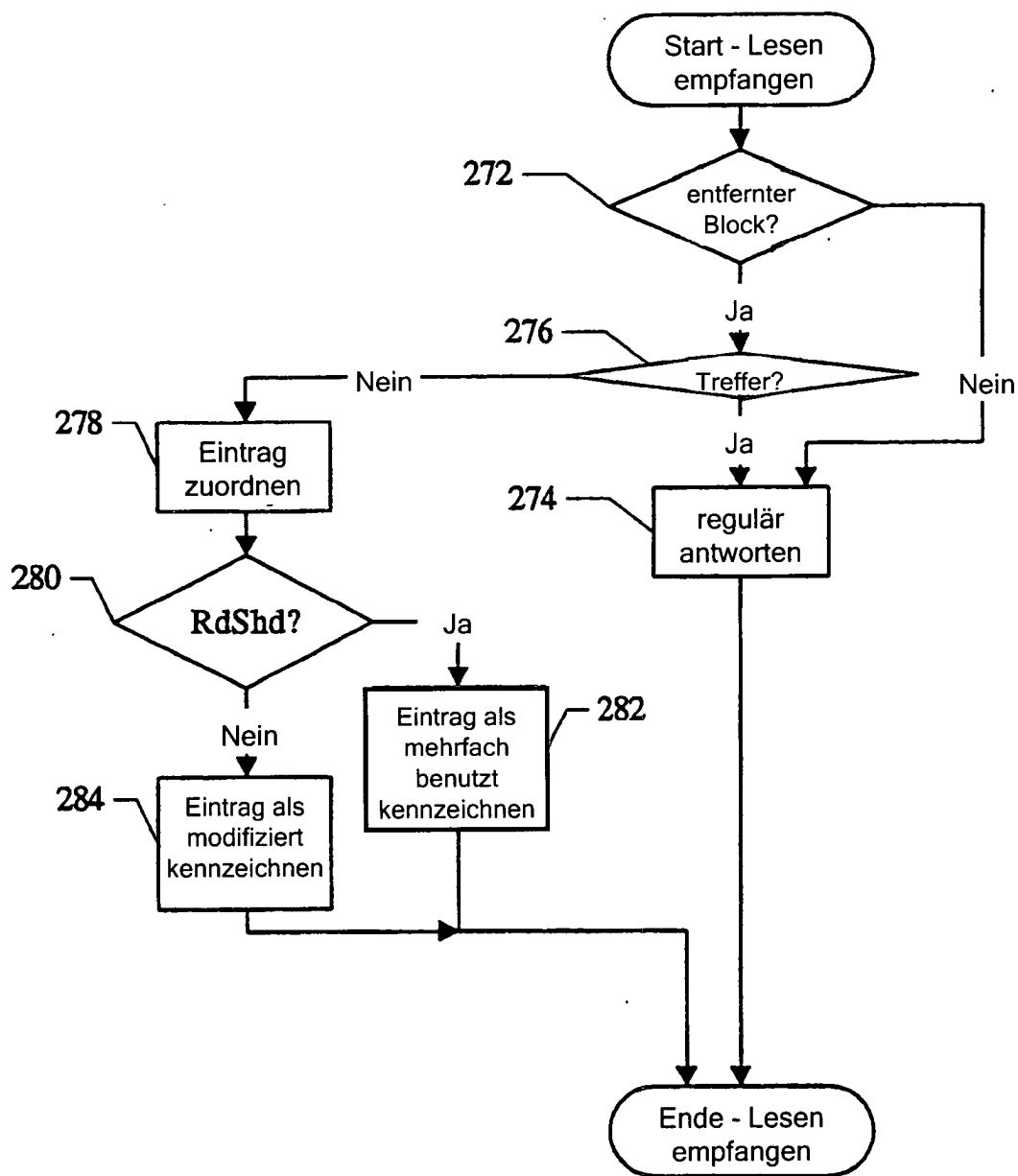
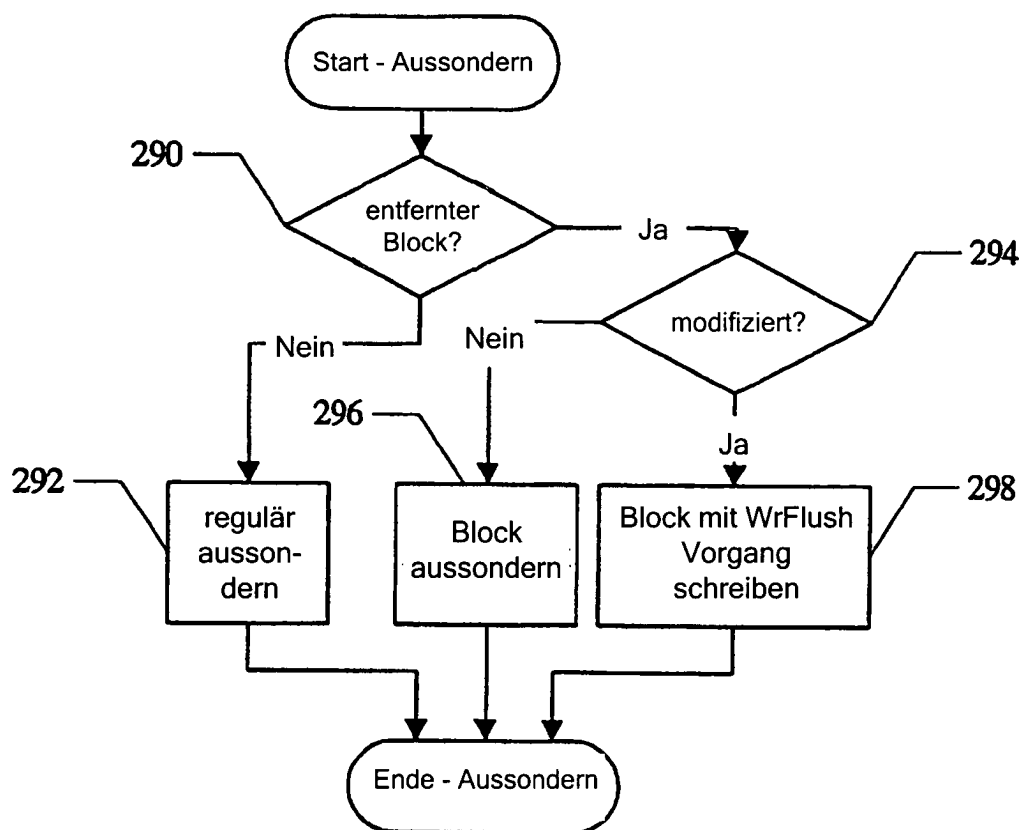


Fig. 13

**Fig. 14**

**Fig. 15**

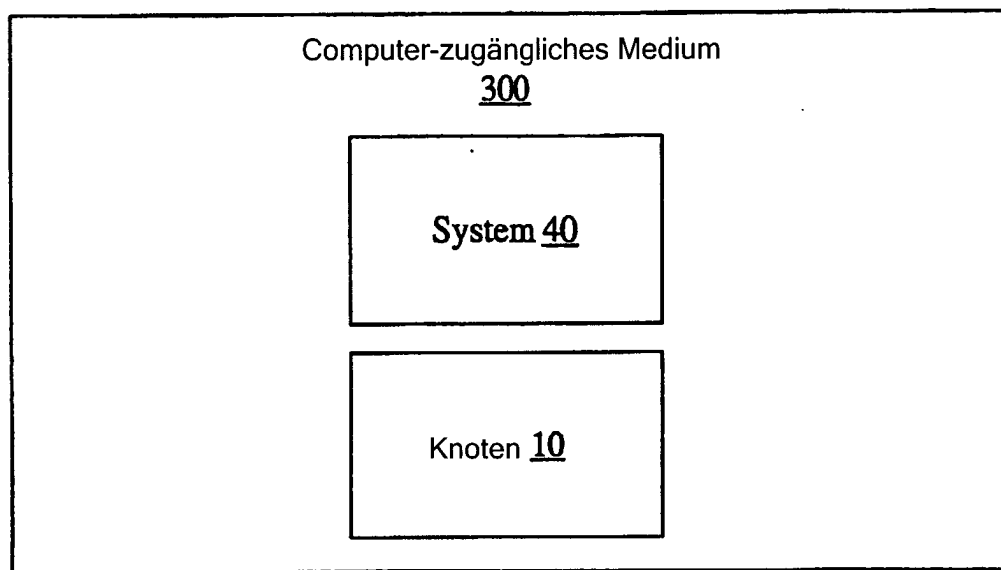


Fig. 16

<u>Quelle</u>	<u>Vorgang</u>	<u>RLD Zustand</u>	<u>Neuer RLD Zustand</u>	<u>Besitzer setzen ?</u>	<u>Andere Besitzer zurücksetzen ?</u>
lokal	RdExc	S	I	--	--
lokal	RdExc	M	I	--	--
lokal	RdShd	M	I	--	--
entfernt	RdExc	S	M	ja	ja
entfernt	RdExc	M	M	ja	ja
entfernt	RdExc	I	M	ja	n/a
entfernt	RdShd	S	S	ja	nein
entfernt	RdShd	M	S	ja	ja
entfernt	RdShd	I	S	ja	n/a
beides	WrInv	S oder M	I	--	--

Fig.17