(19) Deutsches Patent- und Markenamt

(10) **DE 11 2013 007 724 B4** 2024.01.11

(12) # Patentschrift

(21) Deutsches Aktenzeichen: **11 2013 007 724.8**
(86) PCT-Aktenzeichen: **PCT/US2013/077785**
(87) PCT-Veröffentlichungs-Nr.: **WO 2015/099730**
(86) PCT-Anmeldetag: **26.12.2013**
(87) PCT-Veröffentlichungstag: **02.07.2015**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **15.09.2016**
(45) Veröffentlichungstag der Patenterteilung: **11.01.2024**

(51) Int Cl.: ***G06F 13/14*** (2006.01)
G06F 13/20 (2006.01)
G06F 12/0806 (2016.01)

Innerhalb von neun Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten (§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 1 Patentkostengesetz).

(62) Teilung in:
11 2013 007 841.4; 11 2013 007 842.2

(73) Patentinhaber:
INTEL CORPORATION, Santa Clara, Calif., US

(74) Vertreter:
Samson & Partner Patentanwälte mbB, 80538 München, DE

(72) Erfinder:
Das Sharma, Debendra, Saratoga, CA, US; Blankenship, Robert G., Tacoma, Wash., US; Chittor, Suresh S., Portland, Oreg., US; Creta, Kenneth C., Gig Harbor, Wash., US; Fleischer, Balint, Groton, Mass., US; Jen, Michelle C., Sunnyvale, Calif., US; Kumar, Mohan J., Aloha, Oreg., US; Morris, Brian S., Santa Clara, Calif., US

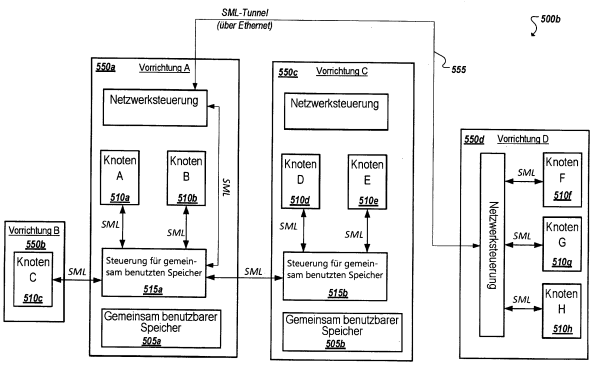
(56) Ermittelter Stand der Technik:

US	**2013 / 0 268 694**	**A1**
CN	**1 03 430 161**	**A**

CN 1 03 430 161 A (in maschineller Übersetzung)

(54) Bezeichnung: **SYSTEM, VORRICHTUNG UND VERFAHREN ZUR GEMEINSAMEN BENUTZUNG VON SPEICHER UND I/O-DIENSTEN ZWISCHEN KNOTEN**

(57) Hauptanspruch: Apparat aufweisend:
eine Steuerung (515; 515a, 515b) eines gemeinsam benutzten Speichers (505; 505a, 505b) zum:
Bedienen von Last- und Speicheroperationen, die über Datenverbindungen von mehreren unabhängigen Knoten (510a,....., 510n; 510a,....., 510e) empfangen werden, um einen Zugang zu einer gemeinsam benutzten Speicherressource vorzusehen, wobei jedem der mehreren unabhängigen Knoten (510a,.....,510n; 510a,.....510e) Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt wird; und
eine I/O-Logik zum:
Identifizieren von Übergängen zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten, die auf den Datenverbindungen gesendet werden,
dadurch gekennzeichnet dass Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch ein Datenstrom-Framing-Token, das zum Identifizieren der Übergänge codiert ist, identifziert werden.



Beschreibung

GEBIET

[0001] Diese Offenbarung betrifft ein System, eine Vorrichtung und ein Verfahren zum Identifizieren von Übergängen zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten. Die Druckschrift US 2013/0 268 694 A1 offenbart einen externen Host sowie ein System von Switches, an das über eine PCIe-Schnittstelle mehrere PCIe-Geräte angeschlossen werden können, wobei es die PCIe-Schnittstelle den daran angeschlossenen Geräten einen direkten Speicherzugriff erlaubt.

HINTERGRUND

[0002] Fortschritte in der Halbleiterbearbeitung und im logischen Design haben eine Erhöhung der Menge an Logik ermöglicht, die auf integrierten Schaltungsvorrichtungen vorhanden sein kann. Als unmittelbare Folge haben sich Computersystemkonfigurationen von einer einzelnen oder mehreren integrierten Schaltungen zu einem System mehrfacher Kerne, mehrfacher Hardware-Threads, und mehrfacher logischer Prozessoren entwickelt, die auf einzelnen integrierten Schaltungen vorhanden sind, wie auch anderen Schnittflächen, die in solchen Prozessoren integriert sind. Ein Prozessor oder eine integrierte Schaltung weist typischerweise ein einzelnes physisches Prozessor-Die auf, wobei das Prozessor-Die eine beliebige Anzahl von Kernen, Hardware-Threads, logischen Prozessoren, Schnittstellen, Speicher, Controller-Hubs, usw. enthalten kann.

[0003] Infolge der besseren Möglichkeit mehr Verarbeitungsleistung in kleinere Packages einzufügen, haben kleinere Rechnervorrichtungen an Beliebtheit gewonnen. Smartphones, Tablets, ultradünne Notebooks und andere Benutzergeräte haben exponentiell zugenommen. Diese kleineren Vorrichtungen basieren jedoch auf Servern sowohl zur Datenspeicherung wie auch zur komplexen Verarbeitung, was den Formfaktor überschreitet. Folglich hat auch der Bedarf am Hochleistungsrechnermarkt (d.h., Server-Raum) zugenommen. Zum Beispiel ist in modernen Servern typischerweise zur Erhöhung der Rechenleistung nicht nur ein einzelner Prozessor mit mehrfachen Kernen vorhanden, sondern es gibt auch mehrfache physische Prozessoren (auch als mehrfache Buchsen bezeichnet). Mit steigender Verarbeitungsleistung im Zusammenhang mit der Anzahl von Vorrichtungen in einem Rechnersystem wird jedoch die Kommunikation zwischen Buchsen und anderen Vorrichtungen kritischer.

[0004] Tatsächlich haben sich Zwischenverbindungen von eher traditionellen Multi-Drop-Bussen, die vorwiegend elektrische Kommunikationen bewältigten, zu vollständig ausgearbeiteten Zwischenverbindungsarchitekturen entwickelt, die eine schnelle Kommunikation erleichtern. Da der Verbrauch zukünftiger Prozessoren bei noch höheren Raten liegen wird, besteht leider ein entsprechender Bedarf bei den Fähigkeiten bestehender Zwischenverbindungsarchitekturen.

[0005] Die CN 103 430 161 A beschreibt ein Verfahren, eine Vorrichtung und ein System, die auf einer PCIE-(Peripheral Component Interconnect Express)-Switch-Kommunikation basieren, bezieht sich auf den Kommunikations- und den Rechnerbereich und wird für die Realisierung der gemeinsamen Nutzung von Daten zwischen Prozessoren durch einen PCIE-Switch verwendet. Bei dem Verfahren empfängt ein erstes Steuerungsprogramm eine von einem ersten Prozessor gesendete RAM-Anforderung. Ein zweites Steuerungsprogramm wird durch erste Markierungsinformationen in der RAM-Anforderung bestimmt und eine PCIE-Markierung des zweiten Steuerungsprogramms wird erhalten. Entsprechend der RAM-Anforderung, der PCIE-Markierung des ersten Steuerungsprogramms und der PCIE-Markierung des zweiten Steuerungsprogramms wird eine PCIE-Anforderungsnachricht erzeugt, die über den PCIE-Switch an das zweite Steuerungsprogramm gesendet wird, sodass das zweite Steuerungsprogramm die RAM-Anforderung an einen zweiten Prozessor senden kann, der von dem ersten Prozessor entsprechend der PCIE-Anforderungsnachricht besucht wird. Der erste Prozessor kann die Speicherdaten des zweiten Prozessors über den PCIE-Switch besuchen und die Daten des zweiten Prozessors abrufen. Dadurch wird der Zweck der gemeinsamen Nutzung von Daten zwischen den Prozessoren realisiert.

[0006] Der Erfindung liegt die Aufgabe zugrunde, einen verbesserten Zugang zu einem gemeinsam benutzten Speicher bereitzustellen.

[0007] Zur Lösung der Aufgabe schlägt die Erfindung einen Apparat gemäß Anspruch 1, einen Apparat gemäß Anspruch 21 und ein Verfahren gemäß Anspruch 25 vor.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

Fig. 1 veranschaulicht eine Ausführungsform eines Rechnersystems, das eine Zwischenverbindungsarchitektur enthält.

Fig. 2 veranschaulicht eine Ausführungsform einer Zwischenverbindungsarchitektur, die einen schichtenförmigen Stapel enthält.

Fig. 3 veranschaulicht eine Ausführungsform einer Anfrage oder eines Pakets, die bzw. das in einer Zwischenverbindungsarchitektur zu generieren ist.

Fig. 4 veranschaulicht eine Ausführungsform eines Sender- und Empfängerpaares für eine Zwischenverbindungsarchitektur.

Fig. 5A veranschaulicht ein vereinfachtes Blockdiagramm einer Ausführungsform eines beispielhaften Knotens.

Fig. 5B veranschaulicht ein vereinfachtes Blockdiagramm einer Ausführungsform eines beispielhaften Systems, das mehrere Knoten enthält.

Fig. 6 ist eine Darstellung von Daten, die gemäß einer beispielhaften, Verbindung zu einem gemeinsam benutzten Speicher übertragen werden.

Fig. 7A ist eine Darstellung von Daten, die gemäß einer anderen beispielhaften, Verbindung zu dem gemeinsam benutzten Speicher übertragen werden.

Fig. 7B ist eine Darstellung eines beispielhaften Beginns eines Daten-Framing-Tokens.

Fig. 8 ist eine Darstellung von Daten, die gemäß einer anderen beispielhaften, Verbindung zu dem gemeinsam benutzten Speichers übertragen werden.

Fig. 9A-9D sind Ablaufdiagramme, die beispielhafte Techniken zur Speicherzugangsnachrichtenübermittlung zeigen.

Fig. 10 veranschaulicht eine Ausführungsform eines Blockdiagramms für ein Rechnersystem, das einen Mehrfachkern-Prozessor enthält.

Fig. 11 veranschaulicht eine andere Ausführungsform eines Blockdiagramms für ein Rechnersystem, das einen Mehrfachkern-Prozessor enthält.

Fig. 12 veranschaulicht eine Ausführungsform eines Blockdiagramms für einen Prozessor.

Fig. 13 veranschaulicht eine andere Ausführungsform eines Blockdiagramms für ein Rechnersystem, das einen Prozessor enthält.

Fig. 14 veranschaulicht eine Ausführungsform eines Blocks für ein Rechnersystem, das mehrfache Prozessoren enthält.

Fig. 15 veranschaulicht ein beispielhaftes System, das als System-on-Chip (SoC) implementiert ist.

[0008] Gleiche Bezugszahlen und -zeichen in den verschiedenen Zeichnungen geben gleiche Elemente an.

AUSFÜHRLICHE BESCHREIBUNG

[0009] In der folgenden Beschreibung sind für ein umfassendes Verständnis der vorliegenden Erfindung zahlreiche spezielle Einzelheiten angegeben, wie beispielsweise spezielle Arten von Prozessoren und Systemkonfigurationen, spezielle Hardware-Strukturen, spezielle architektonische und mikroarchitektonische Einzelheiten, spezielle Registerkonfigurationen, spezielle Befehlsarten, spezielle Systemkomponenten, spezielle Maße/Höhen, spezielle Prozessor-Pipeline-Stufen und spezieller Betrieb usw. Für den Fachmann auf dem Gebiet ist jedoch klar, dass diese speziellen Einzelheiten in der Ausführung der vorliegenden Erfindung nicht verwendet werden müssen. In anderen Beispielen sind allgemein bekannte Komponenten oder Verfahren, wie spezielle und alternative Prozessorarchitekturen, spezielle Logikschaltungen/ein spezieller Code für beschriebene Algorithmen, ein spezieller Firmware-Code, ein spezieller Zwischenverbindungsbetrieb, spezielle Logikkonfigurationen, spezielle Herstellungstechniken und -materialien, spezielle Compiler-Implementierungen, ein spezieller Ausdruck von Algorithmen in einem Code, spezielle Abschalt- und Steuerungstechniken/Logik und andere spezielle betriebliche Einzelheiten eines Computersystems nicht ausführlich beschrieben, um die vorliegende Erfindung nicht unnötig zu verschleiern.

[0010] Obwohl die folgenden Ausführungsformen in Bezug auf Energieeinsparung und Energieeffizienz in speziellen integrierten Schaltungen, wie Rechnerplattformen oder Mikroprozessoren, beschrieben sein können, sind andere Ausführungsformen bei anderen Arten von integrierten Schaltungen und Logikvorrichtungen anwendbar. Ähnliche Techniken und Lehren von hier beschriebenen Ausführungsformen können bei anderen Arten von Schaltungen oder Halbleitervorrichtungen angewendet werden, die auch von einer besseren Energieeffizienz und Energieeinsparung profitieren können. Zum Beispiel sind die offenbarten Ausführungsformen nicht auf Desktop-Computersysteme oder Ultrabooks™ begrenzt. Und können auch in anderen Vorrichtungen, wie in der Hand gehaltenen Vorrichtungen, Tablets, anderen dünnen Notebooks, System-on-Chip-(SOC) Vorrichtungen und eingebetteten Anwendungen verwendet werden. Einige Beispiele für in der Hand gehaltene Vorrichtungen enthalten Mobiltelefone, Internet-Protokollvorrichtungen, Digitalkameras, persönliche digitale Assistenten (PDAs) und in der Hand gehaltene PCs. Eingebettete Anwendungen enthalten typischerweise eine Mikrosteuerung, einen Digitalsignalprozessor (DSP), ein System auf einem Chip, Netzwerkcomputer (NetPC), Set-Top-Boxes, Netzwerk-Hubs, Weitverkehrsnetz- (WAN) Schalter oder jedes andere System, das die in der Folge gelehrten Funktionen und Operationen ausführen kann. Ferner sind die hier beschriebenen

Apparate, Verfahren und Systeme nicht auf physische Rechnervorrichtungen begrenzt, sondern können sich auch auf Software-Optimierungen zur Energieeinsparung und -effizienz beziehen. Wie in der folgenden Beschreibung sofort offensichtlich wird, sind die Ausführungsformen von Verfahren, Apparaten und Systemen (egal ob unter Bezugnahme auf Hardware, Firmware, Software oder eine Kombination davon) für eine Zukunft `grüner Technologie', ausgewogen mit Leistungsüberlegungen, wesentlich.

[0011] Mit der Weiterentwicklung von Rechnersystemen werden die darin enthaltenen Komponenten komplexer. Infolgedessen nimmt auch die Komplexität der Zwischenverbindungsarchitektur zur Kopplung und Kommunikation zwischen den Komponenten zu um sicherzustellen, dass Bandbreitenanforderungen für einen optimalen Komponentenbetrieb erfüllt sind. Ferner verlangen unterschiedliche Marktsegmente unterschiedliche Aspekte von Zwischenverbindungsarchitekturen zur Erfüllung der Marktbedürfnisse. Zum Beispiel erfordern Server eine höhere Leistung, während das mobile Ökosystem manchmal aus Gründen der Leistungseinsparung bei der Gesamtleistung Abstriche hinnehmen kann. Dennoch ist es eine singuläre Aufgabe der meisten Matrizen, eine höchstmögliche Leistung mit einer maximalen Leistungseinsparung zu bieten. In der Folge werden zahlreiche Zwischenverbindungen besprochen, die möglicherweise von Aspekten der hier beschriebenen Erfindung profitieren würden.

[0012] Eine Zwischenverbindungsmatrixarchitektur enthält die Peripheral Component Interconnect (PCI) Express (PCIe) Architektur. Ein primäres Ziel von PCIe besteht darin, einen Betrieb zwischen Komponenten und Vorrichtungen von verschiedenen Verkäufern in einer offenen Architektur zu ermöglichen, wobei mehrfache Marktsegmente überspannt werden; Clients (Desktops und mobile), Server (Standard und firmeneigene) und eingebettete und Kommunikationsvorrichtungen. PCI Express ist eine Hochleistungs-, Allzweck-, I/O-Zwischenverbindung, die für eine breite Palette zukünftiger Rechner- und Kommunikationsplattformen definiert ist. Einige PCI-Attribute, wie sein Gebrauchsmodell, seine Last-Speicherarchitektur und seine Software-Schnittstellen, wurden durch seine Überarbeitungen beibehalten, während vorherige parallele Bus-Implementierungen durch eine hoch skalierbare, vollständig serielle Schnittstelle ersetzt wurden. Die jüngeren Versionen von PCI Express nutzen den Vorteil von Fortschritten in Punkt-zu-Punkt-Zwischenverbindungen, schalterbasierter Technologie und paketiertem Protokoll, um neue Leistungsebenen und Merkmale bereitstellen. Leistungsmanagement, Dienstgüte (Quality Of Service, QoS), Hot-Plug/Hot-Swap-Unterstützung, Datenintegrität und Fehlermanagement zählen zu den weiterentwickelten Merkmalen, die von PCI Express unterstützt werden.

[0013] Unter Bezugnahme auf **Fig.** 1 ist eine Ausführungsform einer Matrix dargestellt, die aus Punkt-zu-Punkt-Verbindungen besteht, die einen Satz von Komponenten verbinden. System 100 enthält einen Prozessor 105 und einen Systemspeicher 110, der an ein Controller-Hub 115 gekoppelt ist. Der Prozessor 105 enthält jedes Verarbeitungselement, wie einen Mikroprozessor, einen Host-Prozessor, einen eingebetteten Prozessor, einen Co-Prozessor oder anderen Prozessor. Der Prozessor 105 ist an den Controller-Hub 115 durch einen Front-Side Bus (FSB) 106. In einer Ausführungsform ist der FSB 106 eine serielle Punkt-zu-Punkt-Zwischenverbindung, wie unten beschrieben. In einer anderen Ausführungsform enthält die Verbindung 106 eine serielle, differentielle Zwischenverbindungsarchitektur, die mit verschiedenen Zwischenverbindungsstandards übereinstimmt.

[0014] Der Systemspeicher 110 enthält jede Speichervorrichtung, wie Direktzugriffsspeicher (RAM), nicht flüchtigen (NV) Speicher oder einen anderen Speicher, der für Vorrichtungen im System 100 zugänglich ist. Der Systemspeicher 110 ist durch eine Speicherschnittstelle 116 an den Controller-Hub 115 gekoppelt. Beispiele einer Speicherschnittstelle enthalten eine Doppeldatenrate- (DDR) Speicherschnittstelle, eine Dualkanal-DDR-Speicherschnittstelle und eine dynamische RAM (DRAM) Speicherschnittstelle.

[0015] In einer Ausführungsform ist der Controller-Hub 115 ein Root-Hub, Root-Komplex oder ein Root-Controller in einer Peripheral Component Interconnect Express (PCIe oder PCIE) Zwischenverbindungshierarchie. Beispiele für den Controller-Hub 115 enthalten einen Chipsatz, einen Memory Controller-Hub (MCH), eine Northbridge, einen Interconnect Controller-Hub (ICH), eine Southbridge und einen Root Controller/Hub. Oftmals bezieht sich der Begriff Chipsatz auf zwei physisch getrennte Controller-Hubs, d.h., einen Memory Controller-Hub (MCH), der an einen Interconnect Controller-Hub (ICH) gekoppelt ist. Es ist zu beachten, dass derzeitige Systeme häufig den MCH mit dem Prozessor 105 integriert enthalten, während der Controller 115 mit I/O Vorrichtungen auf gleiche Weise, wie unten beschrieben, kommunizieren kann. In einigen Ausführungsformen wird optional ein Peer-zu-Peer Routing durch den Root Komplex 115 unterstützt.

[0016] Hier ist der Controller-Hub 115 durch eine serielle Verbindung 119 an einen Schalter/eine Brücke 120 gekoppelt. Eingangs-/Ausgangsmodule 117 und 121, die auch als Schnittstellen/Ports 117 und 121 bezeichnet werden können, enthalten/implementieren einen schichtenförmigen Protokollstapel

zum Vorsehen einer Kommunikation zwischen dem Controller-Hub 115 und dem Schalter 120. In einer Ausführungsform können mehrfache Vorrichtungen an den Schalter 120 gekoppelt werden.

[0017] Der Schalter/die Brücke 120 leitet Pakete/-Nachrichten von der Vorrichtung 125 stromaufwärts, d.h., eine Hierarchie nach oben hin zu einem Root-Komplex, zum Controller-Hub 115 und stromabwärts, d.h., eine Hierarchie nach unten, weg vom Root-Komplex, vom Prozessor 105 oder Systemspeicher 110 zur Vorrichtung 125. Der Schalter 120 wird in einer Ausführungsform als eine logische Zusammenstellung mehrfacher virtueller PCI-zu-PCI Überbrückungsvorrichtungen bezeichnet. Die Vorrichtung 125 enthält jede interne oder externe Vorrichtung oder Komponente, die an ein elektrisches System gekoppelt werden soll, wie eine I/O-Vorrichtung, eine Network Interface Controller (NIC), eine Erweiterungskarte, einen Audioprozessor, einen Netzwerkprozessor, ein Festplattenlaufwerk, eine Speichervorrichtung, einen CD/DVD ROM, einen Monitor, einen Drucker, eine Maus, eine Tastatur, einen Router, eine tragbare Speichervorrichtung, eine Firewire-Vorrichtung, eine Universal Serial Bus (USB) Vorrichtung, einen Scanner und andere Eingangs-/Ausgangsvorrichtungen. Häufig wird im PCIe-Jargon eine solche Vorrichtung als Endpunkt bezeichnet. Obwohl nicht im Speziellen dargestellt, kann die Vorrichtung 125 eine Brücke von PCIe zu PCI/PCI-X enthalten, um althergebrachte oder eine andere Version von PCI Vorrichtungen zu unterstützen. Endpunktvorrichtungen in PCIe sind häufig als althergebrachte, PCIe, oder Root-Komplex integrierte Endpunkte klassifiziert.

[0018] Es ist auch ein Grafikbeschleuniger 130 durch die serielle Verbindung 132 an den Controller-Hub 115 gekoppelt. In einer Ausführungsform ist der Grafikbeschleuniger 130 an einen MCH gekoppelt, der an einen ICH gekoppelt ist. Der Schalter 120, und somit die I/O-Vorrichtung 125, wird dann an den ICH gekoppelt. I/O-Module 131 und 118 sollen auch einen schichtenförmigen Protokollstapel für eine Kommunikation zwischen Grafikbeschleuniger 130 und Controller-Hub 115 implementieren. Ähnlich wie bei der vorangehenden Besprechung des MCH können eine Grafiksteuerung oder der Grafikbeschleuniger 130 selbst im Prozessor 105 integriert sein.

[0019] In Hinblick nun auf **Fig.** 2 ist eine Ausführungsform eines schichtenförmigen Protokollstapels dargestellt. Der schichtenförmige Protokollstapel 200 enthält jede Form eines schichtenförmigen Kommunikationsstapels, wie einen Quick Path Interconnect (QPI) Stapel, einen PCie-Stapel, einen Hochleistungsrechner- Zwischenverbindungsstapel der nächsten Generation oder einen anderen schichtenförmigen Stapel. Obwohl sich die unmittelbare folgende Besprechung unter Bezugnahme auf **Fig.** 1-4 auf einen PCIe-Stapel bezieht, können dieselben Konzepte bei anderen Zwischenverbindungsstapeln angewendet werden. In einer Ausführungsform ist der Protokollstapel 200 ein PCIe-Protokollstapel, der eine Transaktionsschicht 205, eine Verbindungsschicht 210 und eine physische Schicht 220 enthält. Eine Schnittstelle, wie Schnittstellen 117, 118, 121, 122, 126 und 131 in **Fig.** 1, kann als Kommunikationsprotokollstapel 200 dargestellt werden. Die Darstellung als Kommunikationsprotokollstapel kann auch als Modul oder Schnittstelle bezeichnet werden, das bzw. die einen Protokollstapel implementiert/enthält.

[0020] PCI Express verwendet Pakete zur Kommunikation von Informationen zwischen Komponenten. In der Transaktionsschicht 205 und Datenverbindungsschicht 210 werden Pakete gebildet, um die Informationen von der sendenden Komponente zur empfangenden Komponente zu befördern. Während die gesendeten Pakete durch die anderen Schichten gehen, werden sie mit zusätzlichen Informationen erweitert, die zum Handhaben von Pakete in diesen Schichten erforderlich sind. An der Empfangsseite läuft der umgekehrte Prozess und die Pakete werden aus ihrer Darstellung in der physischen Schicht 220 in die Darstellung der Datenverbindungsschicht 210 und letztendlich (für Transaktionsschichtpakete) in die Form umgewandelt, die von der Transaktionsschicht 205 der Empfangsvorrichtung verarbeitet werden kann.

Transaktionsschicht

[0021] In einer Ausführungsform dient die Transaktionsschicht 205 zum Vorsehen einer Schnittstelle zwischen einem Verarbeitungskern einer Vorrichtung und der Zwischenverbindungsarchitektur, wie der Datenverbindungsschicht 210 und der physischen Schicht 220. In dieser Hinsicht ist eine primäre Zuständigkeit der Transaktionsschicht 205 die Zusammenstellung und Zerlegung von Paketen (d.h., Transaktionsschichtpaketen oder TLPs). Die Translationsschicht 205 verwaltet typischerweise die Credit-Based Flow Control für TLPs. PCIe implementiert geteilte Transaktionen, d.h., Transaktionen, bei welchen Anfrage und Antwort zeitlich getrennt sind, wodurch eine Verbindung anderen Verkehr befördern kann, während die Zielvorrichtung Daten für die Antwort sammelt.

[0022] Zusätzlich verwendet die PCIe die Credit-Based Flow Control. In diesem Schema kündigt eine Vorrichtung eine anfängliche Menge an Credits für jeden der Empfangspuffer in der Transaktionsschicht 205 an. Eine externe Vorrichtung am gegenüberliegenden Ende der Verbindung, wie der Controller-Hub 115 in **Fig.** 1, zählt die Anzahl von Credits, die von jedem TLP verbraucht wird. Eine

Transaktion kann gesendet werden, wenn die Transaktion einen Credit-Grenzwert nicht übersteigt. Bei Empfang einer Antwort wird die Credit-Menge wiederhergestellt. Ein Vorteil eines Credit-Schemas ist, dass die Latenz der Credit-Rückgabe die Leistung nicht beeinflusst, vorausgesetzt, der Credit-Grenzwert wird nicht erreicht.

[0023] In einer Ausführungsform enthalten vier Transaktionsadressenräume einen Konfigurationsadressenraum, einen Speicheradressenraum, einen Eingangs-/Ausgangsadressenraum und einen Nachrichtadressenraum. Speicherraumtransaktionen enthalten eine oder mehrere Leseanfragen und Schreibanfragen zum Übertragen von Daten zu/von einer speicherabgebildeten Stelle. In einer Ausführungsform sind Speicherraumtransaktionen imstande, zwei verschiedene Adressenformate zu verwenden, z.B. ein kurzes Adressenformat, wie eine 32-Bit-Adresse, oder ein langes Adressenformat, wie eine 64-Bit-Adresse. Konfigurationsraumtransaktionen werden für einen Zugriff auf den Konfigurationsraum der PCIe-Vorrichtungen verwendet. Transaktionen zum Konfigurationsraum enthalten Leseanfragen und Schreibanfragen. Nachrichtenraumtransaktionen (oder einfach Nachrichten) sind zum Unterstützen einer Kommunikation innerhalb des Bandes zwischen PCIe-Agenten definiert.

[0024] Daher stellt in einer Ausführungsform die Transaktionsschicht 205 die Paketkopfzeile/Nutzlast 206 zusammen. Das Format für aktuelle Paketkopfzeilen/Nutzlasten findet sich in der PCIe-Spezifikation auf der PCIe-Spezifikationen-Website.

[0025] Unter Bezugnahme auf **Fig.** 3 ist kurz eine Ausführungsform eines PCIe-Transaktionsdeskriptors dargestellt. In einer Ausführungsform ist der Transaktionsdeskriptor 300 ein Mechanismus zur Beförderung von Transaktionsinformationen. In dieser Hinsicht unterstützt der Transaktionsdeskriptor 300 die Identifizierung von Transaktionen in einem System. Andere mögliche Anwendungen enthalten eine Verfolgung von Modifizierungen einer vorgegebenen Transaktionsreihung und eine Zuordnung einer Transaktion mit Kanälen.

[0026] Der Transaktionsdeskriptor 300 enthält ein globales Kennungsfeld 302, ein Attributfeld 304 und ein Kanalkennungsfeld 306. In dem dargestellten Beispiel ist das globale Kennungsfeld 302 mit einem lokalen Transaktionskennungsfeld 308 und Quellenkennungsfeld 310 dargestellt. In einer Ausführungsform ist die globale Transaktionskennung 302 für alle ausstehenden Anfragen einzigartig.

[0027] Gemäß einer Implementierung ist das lokale Transaktionskennungsfeld 308 ein Feld, das von einem anfragenden Agenten generiert wird, und ist für alle ausstehenden Anfragen einzigartig, die für diesen anfragenden Agenten vollendet werden müssen. Ferner identifiziert in diesem Beispiel die Quellenkennung 310 den anfragenden Agenten einzigartig innerhalb einer PCIe-Hierarchie. Daher sieht das lokale Transaktionskennungs- 308 Feld gemeinsam mit der Quellen-ID 310 eine globale Identifizierung einer Transaktion innerhalb einer Hierarchiedomäne vor.

[0028] Das Attributfeld 304 spezifiziert Eigenschaften und Verhältnisse der Transaktion. In dieser Hinsicht wird das Attributfeld 304 möglicherweise zum Vorsehen zusätzlicher Informationen verwendet, die eine Modifizierung der vorgegebenen Handhabung von Transaktionen ermöglichen. In einer Ausführungsform enthält das Attributfeld 304 ein Prioritätsfeld 312, ein reserviertes Feld 314, ein Reihungsfeld 316, und ein No-Snoop-Feld 318. Hier kann ein Prioritätsteilfeld 312 von einem Initiator modifiziert werden, um der Transaktion eine Priorität zu verleihen. Das reservierte Attributfeld 314 bleibt für die Zukunft oder für eine vom Verkäufer definierte Nutzung reserviert. Mögliche Gebrauchsmodelle, die Prioritäts- oder Sicherheitsattribute verwenden, können mit Hilfe des reservierten Attributfeldes implementiert werden.

[0029] In diesem Beispiel wird ein Reihungsattributfeld 316 zum Zuleiten optionaler Informationen verwenden, die die Art der Reihung angeben, die vorgegebene Reihungsregeln modifizieren kann. Gemäß einer beispielhaften Implementierung gibt ein Reihungsattribut von „0“ an, dass vorgegebene Reihungsregeln gelten, wobei ein Reihungsattribut von „1“ eine entspannte Reihung angibt, wobei Schreibvorgänge Schreibvorgänge in derselben Richtung durchlaufen können und Lesevervollständigungen Schreibvorgänge in derselben Richtung durchlaufen können. Das Snoop-Attributfeld 318 wird zur Bestimmung verwendet, ob Transaktionen ausspioniert („snooped“) werden. Wie dargestellt, identifiziert das Kanal-ID-Feld 306 einen Kanal, dem eine Transaktion zugeordnet ist.

Verbindungsschicht

[0030] Die Verbindungsschicht 210, auch als Datenverbindungsschicht 210 bezeichnet, dient als Zwischenstufe zwischen der Transaktionsschicht 205 und der physischen Schicht 220. In einer Ausführungsform ist eine Zuständigkeit der Datenverbindungsschicht 210 ein Vorsehen eines zuverlässigen Mechanismus zum Austauschen von Transaktionsschichtpaketen (TLPs) zwischen zwei Komponenten einer Verbindung. Eine Seite der Datenverbindungsschicht 210 akzeptiert TLPs, die von der Transaktionsschicht 205 zusammengestellt wurden, verleiht eine Paketsequenzkennung 211, d.h., eine Identifizierungsnummer oder Paketnummer, berechnet einen Fehlererfassungscode, d.h. CRC 212, und

wendet ihn an, und unterbreitet die modifizierten TLPs der physischen Schicht 220 zur Übertragung über eine physische zu einer externen Vorrichtung.

Physische Schicht

[0031] In einer Ausführungsform enthält die physische Schicht 220 einen logischen Teilblock 221 und einen elektrischen Teilblock 222 zum physischen Senden eines Pakets zu einer externen Vorrichtung. Hier ist der logische Teilblock 221 für die „digitalen" Funktionen der physischen Schicht 221 zuständig. In dieser Hinsicht enthält der logische Teilblock einen Sendeabschnitt zur Vorbereitung ausgehender Informationen, die vom physischen Teilblock 222 gesendet werden, und einen Empfangsabschnitt zum Identifizieren und Vorbereiten empfangener Informationen, bevor diese zur Verbindungsschicht 210 geleitet werden.

[0032] Der physische Block 222 enthält einen Sender und einen Empfänger. Dem Sender werden vom logischen Teilblock 221 Symbole zugleitet, die der Sender serialisiert und zu einer externen Vorrichtung sendet. Dem Empfänger werden die serialisierten Symbole von einer externen Vorrichtung zugeleitet, und er wandelt die empfangenen Signale in einen Bit-Strom um. Der Bit-Strom wird entserialisiert und zum logischen Teilblock 221 geleitet. In einer Ausführungsform wird ein 8b/10b Sendungscode verwendet, wobei zehn-Bit Symbole gesendet/empfangen werden. Hier werden Spezialsymbole verwendet, um ein Paket mit Frames 223 zu versehen. Zusätzlich sieht in einem Beispiel der Empfänger auch einen Symboltakt vor, der aus dem eingehenden seriellen Strom gewonnen wird.

[0033] Wie oben angegeben, obwohl die Transaktionsschicht 205, die Verbindungsschicht 210 und die physische Schicht 220 in Bezug auf eine spezielle Ausführungsform eines PCIe-Protokollstapels besprochen werden, ist ein schichtenförmiger Protokollstapel nicht derart begrenzt. Tatsächlich kann jedes schichtenförmige Protokoll enthalten/implementiert sein. Als ein Beispiel enthält ein Port/eine Schnittstelle, dargestellt als ein schichtenförmiges Protokoll: (1) eine erste Schicht zum Zusammenstellen von Paketen, d.h., eine Transaktionsschicht; eine zweite Schicht zur Reihung von Paketen, d.h., eine Verbindungsschicht; und eine dritte Schicht zum Senden der Pakete, d.h., eine physische Schicht. Als ein spezielles Beispiel wird ein schichtenförmiges Protokoll einer allgemeinen Standardschnittstelle (Common Standard Interface, CSI) verwendet.

[0034] Unter Bezugnahme im Anschluss auf **Fig.** 4 ist eine Ausführungsform einer seriellen Punkt-zu-Punkt-PCIe-Matrix dargestellt. Obwohl eine Ausführungsform einer seriellen Punkt-zu-Punkt- PCIe-Verbindung dargestellt ist, ist eine serielle Punkt-zu-Punkt-Verbindung nicht derart begrenzt, da sie jeden Sendepfad zum Senden serieller Daten enthält. In der dargestellten Ausführungsform enthält eine grundlegende PCIe-Verbindung zwei unterschiedliche angesteuerte Niederspannungssignalpaare: ein Sendungspaar 406/411 und ein Empfangspaar 412/407. Daher enthält die Vorrichtung 405 eine Sendelogik 406 zum Senden von Daten zur Vorrichtung 410 und eine Empfangslogik 407 zum Empfangen von Daten von der Vorrichtung 410. Mit anderen Worten, in einer PCIe-Verbindung sind zwei Sendepfade, d.h., Pfade 416 und 417, und zwei Empfangspfade, d.h., Pfade 418 und 419, enthalten.

[0035] Ein Sendepfad bezieht sich auf jeden Pfad zum Senden von Daten, wie eine Sendeleitung, eine Kupferleitung, eine optische Leitung, einen drahtlosen Kommunikationskanal, eine Infrarot-Kommunikationsverbindung oder einen anderen Kommunikationspfad. Eine Anbindung zwischen zwei Vorrichtungen, wie Vorrichtung 405 und Vorrichtung 410, wird als Verbindung bezeichnet, wie Verbindung 415. Eine Verbindung kann eine Spur unterstützen - wobei jede Spur einen Satz unterschiedlicher Signalpaare darstellt (ein Paar zum Senden, ein Paar zum Empfangen). Zur Skalierung der Bandbreite kann eine Verbindung mehrfache Spuren zusammenfassen, die mit xN bezeichnet sind, wobei N jede unterstützte Verbindungsbreite, wie 1, 2, 4, 8, 12, 16, 32, 64 oder breiter, ist.

[0036] Ein unterschiedliches Paar bezieht sich auf zwei Sendepfade, wie Leitungen 416 und 417, zum Senden unterschiedlicher Signale. Als ein Beispiel, wenn die Leitung 416 von einem Niederspannungspegel zu einem Hochspannungspegel wechselt, d.h., eine ansteigende Flanke, steuert die Leitung 417 von einem hohen Logikpegel zu einem niederen Logikpegel, d.h., eine abfallende Flanke. Unterschiedliche Signale zeigen möglicherweise bessere elektrische Eigenschaften, wie eine bessere Signalintegrität, d.h., Kreuzkopplung, Spannungsüberschreitung/-unterschreitung, Ringing, usw. Dies erlaubt ein besseres Zeitsteuerungsfenster, das schnellere Sendefrequenzen ermöglicht.

[0037] Physische Schichten bestehender Zwischenverbindungs- und Kommunikationsarchitekturen, einschließlich PCIe, können aufgebaut werden, um gemeinsam benutzte Speicher- und I/O-Dienste in einem System vorzusehen. Üblicherweise können cachebare Speicher nicht zwischen unabhängigen Systemen unter Verwendung herkömmlicher Last-/Speicher- (LD/ST) Speichersemantik geteilt werden. Ein unabhängiges System oder ein „Knoten", kann in dem Sinn unabhängig sein, dass es bzw. er als eine einzige logische Einheit funktioniert, von einem einzigen Betriebssystem (und/oder einem einzigen BIOS oder Virtual Machine Monitor (VMM)) gesteuert

wird und/oder eine unabhängige Fehlerdomäne hat. Ein einzelner Knoten kann eine oder mehrere Prozessorvorrichtung(en) enthalten, kann auf einer einzigen Platine oder mehreren Platinen implementiert sein und einen lokalen Speicher, einschließlich eines cachebaren Speichers, enthalten, auf den mit Hilfe von LD/ST-Semantik durch die Vorrichtungen auf demselben Knoten zugegriffen werden kann. In einem Knoten kann ein gemeinsam benutzter Speicher einen Speicherblock oder mehrere Speicherblöcke enthalten, wie einen Direktzugriffsspeicher (RAM), auf den mehrere verschiedene Prozessoren (z.B. zentrale Verarbeitungseinheiten (CPUs)) in einem Knoten zugreifen können. Ein gemeinsam benutzter Speicher kann auch den lokalen Speicher der Prozessoren oder anderer Vorrichtungen im Knoten enthalten. Die mehrfachen Vorrichtungen in einem Knoten mit einem gemeinsam benutzten Speicher können sich eine einzelne Ansicht von Daten im gemeinsam benutzten Speicher teilen. Die I/O-Kommunikation, die den gemeinsam benutzten Speicher beinhaltet, kann eine sehr geringe Latenz aufweisen und einen raschen Zugang zum Speicher durch die mehrfachen Prozessoren ermöglichen.

[0038] Üblicherweise hat eine gemeinsame Speicherbenutzung zwischen verschiedenen Knoten nach einem Last-/Speicherparadigma keine gemeinsame Speicherbenutzung ermöglicht. Zum Beispiel wurde in einigen Systemen eine gemeinsame Speicherbenutzung zwischen verschiedenen Knoten durch verteilte Speicherarchitekturen erleichtert. In herkömmlichen Lösungen bearbeiten Rechenaufgaben lokale Daten und wenn Daten eines anderen Knotens erwünscht sind, kommuniziert die Rechenaufgabe (die z.B. von einem anderen CPU-Knoten ausgeführt wird) mit dem anderen Knoten zum Beispiel über einen Kommunikationskanal, der einen Kommunikationsprotokollstapel, wie Ethernet, InfiniBand oder ein anderes schichtenförmiges Protokoll verwendet. In herkömmlichen Mehrfachknotensystemen muss den Prozessoren verschiedener Knoten nicht bewusst sein, wo die Daten liegen. Eine gemeinsame Datenbenutzung unter Verwendung herkömmlicher Methoden, wie über einen Protokollstapel, kann eine signifikant höhere Latenz als eine gemeinsame Speicherbenutzung in einem Knoten haben, der ein Last-/Speicherparadigma verwendet. Anstatt eines direkten Adressierens und Bearbeitens von Daten in einem gemeinsam benutzten Speicher kann ein Knoten, neben anderen Beispielen, Daten von einem anderen mit Hilfe eines bestehenden Protokoll-Handshakes wie Ethernet (oder Infiniband) anfragen und der Quellenknoten kann die Daten bereitstellen, so dass die Daten durch den anfragenden Knoten gespeichert und bearbeitet werden können.

[0039] In einigen Implementierungen, kann eine gemeinsam benutzte Speicherarchitektur vorgesehen sein, die eine gemeinsame Speicherbenutzung zwischen unabhängigen Knoten für einen ausschließlichen oder gemeinsamen Zugang unter Verwendung einer Last-/Speicher- (LD/ST) Speichersemantik ermöglicht. In einem Beispiel können die Speichersemantik (und, falls zutreffend, Verzeichnisinformationen) und I/O-Semantik (für Protokolle wie PCIe) entweder auf einen gemeinsamen Satz von Pins oder einen separaten Satz von Pins exportiert werden. In einem solchen System kann jeder von mehreren Knoten in einem System die verbesserte, gemeinsam benutzte Speicherarchitektur, um seine eigene unabhängige Fehlerdomäne (und lokalen Speicher) beizubehalten, während ein gemeinsam benutzter Speicherpool für einen Zugang durch den Knoten und eine Nachrichtübermittlung niedriger Latenz, die zwischen Knoten läuft, unter Verwendung eines Speichers nach der LD/ST-Semantik möglich ist. In einigen Implementierungen kann ein solcher, gemeinsam benutzter Speicherpool dynamisch (oder statisch) zwischen verschiedenen Knoten zugeordnet werden. Daher können die verschiedenen Knoten eines Systems als sich dynamisch ändernde Gruppen von Knoten konfiguriert werden, die bei Bedarf gemeinsam und flexibel verschiedene Aufgaben bearbeiten, welche zum Beispiel die gemeinsam benutzte Speicherinfrastruktur verwenden.

[0040] In Hinblick nun auf **Fig.** 5A ist ein vereinfachtes Blockdiagramm 500a dargestellt, das ein beispielhaftes System zeigt, das einen gemeinsam benutzten Speicher 505 enthält, auf den mit Hilfe von Last-/Speichertechniken von jedem der mehreren unabhängigen Knoten 510a-510n zugegriffen werden kann. Zum Beispiel kann eine Steuerung 515 des gemeinsam benutzten Speichers 505 vorgesehen sein, die Last-/Speicherzugangsanfragen der verschiedenen Knoten 510a-510n auf dem System annehmen kann. Der gemeinsam benutzte Speicher 505 kann durch Verwendung eines synchronen dynamischen Direktzugriffsspeichers (SDRAM), dualer In-line-Speichermodule (DIMM) und eines anderen nicht flüchtigen Speichers (oder flüchtigen Speichers) implementiert sein.

[0041] Jeder Knoten kann selbst eine oder mehrere CPU-Buchse(n) haben und kann auch einen lokalen Speicher enthalten, der von einem LD/ST-Zugang durch andere Knoten im System isoliert bleibt. Der Knoten kann mit anderen Vorrichtungen auf dem System (z.B. der Steuerung 515 des gemeinsam benutzten Speichers 505, der Netzwerksteuerung 520, einem anderen Knoten, usw.) unter Verwendung eines Protokolls oder mehrerer Protokolle, einschließlich PCIe, QPI, Ethernet, neben anderen Beispielen, kommunizieren. In einigen Implementierungen kann ein Verbindungsprotokoll des gemeinsam benutzten Speichers (SML-Protokoll) vorgesehen sein, durch das eine LD/ST-Spei-

chersemantik niedriger Latenz unterstützt werden kann. Zum Beispiel kann SML in der Kommunikation von Lese- und Schreibvorgängen eines gemeinsam benutzten Speichers 505 (durch die Steuerung 515 des gemeinsam benutzten Speichers 505) von den verschiedenen Knoten 510a-510n eines Systems verwendet werden.

[0042] In einem Beispiel kann die SML auf einem Speicherzugangsprotokoll, wie Scalable Memory Interconnect (SMI) der 3. Generation (SMI3), basieren. Alternativ können andere Speicherzugangsprotokolle verwendet werden, wie transaktionale Speicherzugangsprotokolle, wie, neben anderen Beispielen, ein vollständig gepuffertes DIMM (FB-DIMM), DDR transaktional (DDR-T). In anderen Fällen kann die SML auf einer nativen PCIe-Speicherlese-/-schreibsemantik mit zusätzliche Verzeichniserweiterungen basieren. Eine auf einem Speicherprotokoll basierte Implementierung der SML kann Vorteile in der Bandbreiteneffizienz bieten, da sie auf Cachezeilen-Speicherzugänge zugeschnitten ist. Während Hochleistungskommunikationsprotokolle zwischen Vorrichtungen bestehen, wie PCIe, können obere Schichten (z.B. Transaktions- und Verbindungsschichten) solcher Protokolle eine Latenz einführen, die die Anwendung des vollständigen Protokolls zur Verwendung in LD/ST-Speichertransaktionen verschlechtert, einschließlich Transaktionen, die einen gemeinsam benutzten Speicher 505 beinhalten. Ein Speicherprotokoll, wie SMI3, kann einen möglichen zusätzlichen Vorteil bedeuten, indem Zugänge mit niedriger Latenz geboten werden, da es den Großteil eines anderen Protokollstapels, wie PCIe, umgehen kann. Daher können SMI,-Implementierungen SMI3 oder ein anderes Speicherprotokoll nutzen, das auf einer logischen und physischen PHY eines anderen Protokolls, wie SMI3 on PCIe, läuft.

[0043] Wie festgehalten wurde, kann in einigen Implementierungen eine Steuerung 515 des gemeinsam benutzten Speichers (SMC) vorgesehen sein, die eine Logik zur Bearbeitung von Last-/Speicheranfragen von Knoten 510a-510n im System enthält. Last-/Speicheranfragen können von der SMC 515 über Verbindungen empfangen werden, die die SML verwenden und die Knoten 510a-510n mit der SMC 515 verbinden. In einigen Implementierungen kann die SMC 515 als eine Vorrichtung implementiert sein, wie eine anwendungsspezifische integrierte Schaltung (ASIC), einschließlich einer Logik zum Bedienen der Zugangsanfragen der Knoten 510a-510n für gemeinsam benutzte Speicherressourcen. In anderen Fällen kann die SMC 515 (wie auch der gemeinsam benutzte Speicher 505) auf einer Vorrichtung, einem Chip oder einer Platine, getrennt von einem oder mehreren (oder sogar allen) der Knoten 510a-510n liegen. Die SMC 515 kann ferner eine Logik zum Koordinieren verschiedener Transaktionen der Knoten enthalten, die den gemeinsam benutzten Speicher 505 beinhalten. Zusätzlich kann die SMC einen Verzeichnisverfolgungszugang zu verschiedenen Datenressourcen führen, wie jede Cachezeile, die im gemeinsam benutzten Speicher 505 enthalten sind. Zum Beispiel kann eine Datenressource, unter anderen möglichen Beispielen, in einem gemeinsam benutzten Zugangszustand (z.B. imstande, dass mehrere Verarbeitungs- und/oder I/O-Vorrichtungen in einem Knoten gleichzeitig zugreifen (z.B. geladen oder gelesen)), einem ausschließlichen Zugangszustand (z.B. ausschließlich, wenn nicht temporär von einer einzigen Verarbeitungs- und/oder I/O-Vorrichtung in einem Knoten (z.B. für einen Speicher- oder Schreibvorgang) reserviert), einem Nicht-Cache-Zustand sein. Während ferner jeder Knoten einen direkten Zugang zu einem oder mehreren Abschnitt(en) eines gemeinsam benutzten Speichers 505 haben kann, können verschiedene Adressierungsschemata und -werte von den verschiedenen Knoten (z.B. 510a-510n) verwendet werden, was dazu führt, dass im selben gemeinsam benutzten Speicher von einem ersten Knoten gemäß einem ersten Adressenwert und einem zweiten Knoten, der auf dieselben Daten durch einen zweiten Adressenwert verweist, auf Daten (z.B. in einem Befehl) verwiesen wird. Die SMC 515 kann eine Logik enthalten, die Datenstrukturen beinhaltet, die Adressen der Knoten auf gemeinsam benutzte Speicherressourcen abbildet, so dass die SMC 515 die verschiedenen Zugangsanfragen der verschiedenen Knoten interpretieren kann.

[0044] Zusätzlich kann in einigen Fällen ein gewisser Teil des gemeinsam benutzten Speichers (z.B. gewisse Trennungen, Speicherblöcke, Aufzeichnungen, Dateien, usw.) gewissen Genehmigungen, Regeln und Zuordnungen unterliegen, so dass nur ein Teil der Knoten 510a-510n (z.B. durch die SMC 515) auf entsprechende Daten zugreifen kann. Tatsächlich kann jede gemeinsam benutzte Speicherressource einem entsprechenden (und in einigen Fällen anderen) Teilsatz der Knoten 510a-510n des Systems zugeordnet werden. Diese Zuordnungen können dynamisch sein und die SMC 515 kann solche Regeln und Genehmigungen (z.B. auf Anfrage, dynamisch, usw.) modifizieren, um neue oder geänderte Regeln, Genehmigungen, Knotenzuordnungen und Eigentumsrecht anzupassen, die für einen bestimmten Teil des gemeinsam benutzten Speichers 505 gelten.

[0045] Eine beispielhafte SMC 515 kann ferner verschiedene Transaktionen verfolgen, die Knoten (z.B. 510a-510n) im System beinhalten, die auf eine oder mehrere gemeinsam benutzte Speicherressourcen zugreifen. Zum Beispiel kann die SMC 515 Informationen für jede Transaktion eines gemeinsam benutzten Speichers 505 verfolgen, einschließlich einer

Identifizierung des (der) Knoten(s), der (die) an der Transaktion beteiligt ist (sind), des Fortlaufs der Transaktion (z.B. ob sie vollendet ist), neben anderen Transaktionsinformationen. Dies kann ermöglichen, dass einige der transaktionsorientierten Aspekte herkömmlicher verteilter Speicherarchitekturen bei der hier beschriebenen, verbesserten, gemeinsam benutzten Mehrfachknoten-Speicherarchitektur angewendet werden. Zusätzlich kann eine Transaktionsverfolgung (z.B. von der SMC) verwendet werden, eine Aufrechterhaltung oder Durchsetzung der separaten und unabhängigen Fehlerdomänen jedes entsprechenden Knoten zu unterstützen. Zum Beispiel kann die SMC die entsprechende Knoten-ID für jede laufende Transaktion in ihren internen Datenstrukturen, einschließlich im Speicher, verwalten und diese Informationen verwenden, um Zugangsrechte durchzusetzen und einzelne Fehlerdomänen für jeden Knoten zu verwalten. Wenn einer der Knoten abgeschaltet wird (z.B. aufgrund eines kritischen Fehlers, einer ausgelösten Wiederherstellungssequenz oder eines anderen Fehlers oder Ereignisses), werden daher nur dieser Knoten und seine Transaktionen, die den gemeinsam benutzten Speicher 505 beinhalten, unterbrochen (z.B. von der SMC fallengelassen) - Transaktionen der übrigen Knoten, die den gemeinsam benutzten Speicher 505 beinhalten, laufen unabhängig von dem Fehler in dem anderen Knoten weiter.

[0046] Ein System kann mehrfache Knoten enthalten. Zusätzlich können einige beispielhafte Systeme mehrfache SMCs enthalten. In einigen Fällen kann ein Knoten imstande sein, auf einen gemeinsam benutzten Speicher jenseits einer fernen SMC zuzugreifen, mit welcher er nicht direkt verbunden ist (d.h., die lokale SMC des Knotens ist mit der fernen SMC durch einen oder mehrere SML-Verbindungs-Hops verbunden). Die ferne SMC kann sich in derselben Platine befinden oder könnte sich in einer anderen Platine befinden. In einigen Fällen können einige der Knoten abseits des Systems sein (z.B. abseits der Platine oder abseits des Chips), aber dennoch auf einen gemeinsam benutzten Speicher 505 zugreifen. Zum Beispiel, können ein oder mehrere Knoten abseits des Systems unter Verwendung einer SML-konformen Verbindung, neben anderen Beispielen, direkt an die SMC angeschlossen sein. Zusätzlich können andere Systeme, die ihre eigene SMC und einen gemeinsam benutzten Speicher enthalten, auch mit der SMC 510 verbunden sein, um eine gemeinsame Benutzung des Speichers 505 auf Knoten zu erweitern, die zum Beispiel auf einer anderen Platine enthalten sind, die eine Schnittstelle mit der anderen SMC hat, die mit der SMC über eine SML-Verbindung verbunden ist. Ferner können Netzwerkverbindungen durchgetunnelt sein, um den Zugang auf die anderen Knoten abseits der Platine oder abseits des Chips zu erweitern. Zum Beispiel kann die SML über eine Ethernet-Verbindung (die z.B. durch die Netzwerksteuerung 520 bereitgestellt ist) tunneln, die das beispielhafte System von **Fig.** 5A kommunikativ mit einem anderen System koppelt, das auch einen oder mehrere andere Knoten enthält, und diesen Knoten ebenso einen Zugang zur SMC 515 und dadurch zum gemeinsam benutzten Speicher 505, neben anderen Beispielen, ermöglichen.

[0047] Als weiteres Beispiel, wie in dem vereinfachten Blockdiagramm 500b von **Fig.** 5B dargestellt, kann eine verbesserte, gemeinsam benutzte Speicherarchitektur, die einen gemeinsamen Zugang durch mehrfache unabhängige Knoten gemäß einer LD/ST-Speichersemantik ermöglicht, flexibel ein Vorsehen einer Reihe verschiedener Mehrfachknotensystemdesigns ermöglichen. Es können verschiedene Kombinationen der mehrfachen Knoten zugeordnet werden, um Teile eines gemeinsam benutzten Speicherblocks oder mehrerer gemeinsam benutzten Speicherblöcke gemeinsam zu benutzen, die in einem beispielhaften System vorgesehen sind. Zum Beispiel kann ein anderes beispielhaftes System, das in dem Beispiel von **Fig.** 5B dargestellt ist, mehrfache Vorrichtungen 550a-550d enthalten, die zum Beispiel als separate Dies, Platinen, Chips, usw. implementiert sind, wobei jede Vorrichtung einen oder mehrere unabhängige CPU-Knoten (z.B. 510a-510h) enthält. Jeder Knoten kann seinen eigenen lokalen Speicher enthalten. Eine oder mehrere der mehrfachen Vorrichtungen 550a-550d können ferner einen gemeinsam benutzten Speicher enthalten, auf den zwei oder mehr der Knoten 510a-510h des Systems zugreifen können.

[0048] Das in **Fig.** 5B dargestellte System ist ein Beispiel, das zur Veranschaulichung eines Teils der Variabilität vorgesehen ist, die durch eine verbesserte, gemeinsam benutzte Speicherarchitektur erreicht werden kann, wie hier dargestellt und beschrieben. Zum Beispiel kann jede von einer Vorrichtung A 550a und Vorrichtung C 550c ein entsprechendes gemeinsam benutztes Speicherelement (z.B. 505a, 505b) enthalten. Daher kann in einigen Implementierungen jedes gemeinsam benutzte Speicherelement auf einer eigenen Vorrichtung ferner eine entsprechende Steuerung 515a, 515b des gemeinsam benutzten Speichers (SMC) enthalten. Verschiedene Kombinationen von Knoten 510a-510h können kommunikativ an jede SMC (z.B. 515a, 515b) gekoppelt sein, wodurch die Knoten Zugang zu dem entsprechenden gemeinsam benutzten Speicher (z.B. 505a, 505b) erlangen. Als ein Beispiel kann die SMC 515a von Vorrichtung A 550a mit den Knoten 510a, 510b auf Vorrichtung A unter Verwendung einer direkten Datenverbindung, die SML unterstützt, verbunden sein. Zusätzlich können andere Knoten 510c auf einer anderen Vorrichtung (z.B. Vorrichtung C 550c) auch durch eine direkte, hartverdrahtete Verbindung (die SML unterstützt) vom Knoten 510c (und/oder seiner Vorrich-

tung 550c) zur SMC 515a Zugang zum gemeinsam benutzten Speicher 505a haben. Indirekte, auf dem Netzwerk basierende und andere derartige Verbindungen können auch verwendet werden, um Knoten (z.B. 510f-510h) einer fernen Vorrichtung oder abseits der Platine (z.B. Vorrichtung D 550d) die Verwendung eines herkömmlichen Protokollstapels zu ermöglichen, um eine Schnittstelle mit der SMC 515a zu haben, um ebenso auf den gemeinsam benutzten Speicher 505a Zugang zu haben. Zum Beispiel kann ein SML-Tunnel 555 über eine Ethernet-, InfiniBand- oder andere Verbindung errichtet werden, die Vorrichtung A und Vorrichtung D koppelt. Während der Errichtung und Aufrechterhaltung kann der Tunnel einen gewissen zusätzlichen Mehraufwand und eine Latenz im Vergleich zur SML einführen, die auf anderen, weniger durch Software verwalteten physischen Verbindungen läuft, wobei der SML-Tunnel 555, wenn er errichtet ist, wie andere SML-Kanäle arbeiten und dem Knoten 510f-510h ermöglichen kann, eine Schnittstelle mit der SMC 515a über SML zu haben und auf den gemeinsam benutzten Speicher 505a zuzugreifen, wie dies jeder andere Knoten kann, der mit der SMC über eine SML-Verbindung kommuniziert. Zum Beispiel können Zuverlässigkeit und Reihung der Pakete in den SML-Kanälen entweder durch die netzwerkenden Komponenten im System durchgesetzt werden oder können Ende zu Ende zwischen den SMCs durchgesetzt werden.

[0049] In einem weiteren anderen Beispiel können Knoten (z.B. 515d, 515e) auf einer Vorrichtung, die sich von jener unterscheidet, die einen bestimmten Teil des gemeinsam benutzten Speichers (z.B. 505a) beherbergt, indirekt mit der entsprechenden SMC (z.B. SMC 515a) durch eine direkte Verbindung mit einer anderen SMC (z.B. 515b) verbunden sein, die selbst (z.B. unter Verwendung einer SML Verbindung) an die entsprechende SMC (z.B. 515a) gekoppelt ist. Ein Verbinden von zwei oder mehr SMCs (z.B. 515a, 515b) kann effektiv die Menge an verfügbarem gemeinsam benutzten Speicher erhöhen, die den Knoten 510a-510h auf dem System zur Verfügung steht. Zum Beispiel kann aufgrund einer Verbindung zwischen SMCs 515a, 515b im Beispiel von **Fig.** 5B in einigen Implementierungen jeder der Knoten (z.B. 510a-510c, 510f-510h), der auf einen gemeinsam benutzten Speicher 505a durch die SMC 515a zugreifen kann, auch möglicherweise auf einen gemeinsam benutzbaren Speicher 505b aufgrund der Verbindung zwischen der SMC 515a und der SMC 515b zugreifen. Ebenso kann in einigen Implementierungen jeder der Knoten, die direkt auf die SMC 515b zugreifen kann, auch auf einen gemeinsam benutzbaren Speicher 505a aufgrund der Verbindung zwischen den SMCs 515a, 515b zugreifen, neben anderen möglichen Beispielen.

[0050] Wie oben festgehalten wurde, kann eine verbesserte, gemeinsam benutzte Speicherarchitektur ein Verbindungsprotokoll niedriger Latenz (d.h., SML) auf der Basis eines Speicherzugangsprotokolls, wie SMI3, enthalten und vorgesehen sein, um Last-/Speicheranfragen zu erleichtern, die den gemeinsam benutzten Speicher beinhalten. Während das traditionelle SMI3 und andere Speicherzugangsprotokolle zur Verwendung in einer gemeinsamen Speicherbenutzung innerhalb eines einzigen Knotens konfiguriert sein können, kann die SML die Speicherzugangssemantik auf mehrfache Knoten erweitern, um eine gemeinsame Speicherbenutzung zwischen den mehrfachen Knoten zu ermöglichen. Ferner kann die SML möglicherweise auf jeder physischen Kommunikationsverbindung verwendet werden. Die SML kann ein Speicherzugangsprotokoll verwenden, das eine LD/ST-Speichersemantik unterstützt, die auf eine physische Schicht (und entsprechende physische Schichtlogik) aufgelegt ist, die dazu angepasst ist, verschiedene Vorrichtungen (und Knoten) miteinander zu verbinden. Zusätzlich kann eine physische Schichtlogik der SML neben anderen Merkmalen keine Paketverlust- und Fehlerwiederholungsfunktionalität vorsehen.

[0051] In einigen Implementierungen kann die SML durch Auflegen des SMI3 auf ein PCIe PHY implementiert sein. Eine SML-Verbindungsschicht kann (z.B. anstelle einer traditionellen PCIe-Verbindungsschicht) vorgesehen sein, um auf eine Strömungssteuerung und andere Merkmale zu verzichten und einen Speicherzugang niedriger Latenz zu erleichtern, wie für traditionelle CPU-Speicherzugangsarchitekturen charakteristisch wäre. In einem Beispiel kann die SML-Verbindungsschichtlogik zwischen Transaktionen mit dem gemeinsam benutzten Speicher und anderen Transaktionen multiplexen. Zum Beispiel kann die SML-Verbindungsschichtlogik zwischen SMI3- und PCIe-Transaktionen multiplexen. Zum Beispiel kann das SMI3 (oder ein anderes Speicherprotokoll) auf einer PCIe (oder anderen Zwischenverbindungsprotokoll) liegen, so dass die Verbindung dynamisch zwischen SMI3- und PCIe-Transaktionen wechseln kann. Dies ermöglicht, dass in einigen Fällen traditioneller PCIe-Verkehr gleichzeitig auf derselben Verbindung wie SML-Verkehr vorhanden ist.

[0052] In Hinblick nun auf **Fig.** 6 ist eine Darstellung 600 gezeigt, die eine erste Implementierung von SML veranschaulicht. Zum Beispiel kann die SML durch Auflegen eines SMI3 auf eine PCIe PHY implementiert sein. Die physische Schicht kann eine Standard-PCIe 128b/130b verwenden, die für alle Aktivitäten der physischen Schicht codiert, einschließlich Verbindungstraining- wie auch PCIe-Datenblöcke. Die SML kann einen Verkehr auf den Spuren (z.B. Lane0 - Lane7) der zu multiplexenden Verbindung zwischen PCIe-Paketen und SMI3-Flits vorsehen.

Zum Beispiel kann in der in **Fig.** 6 dargestellten Implementierung die Sync-Kopfzeile der PCIe 128b/130b Codierung modifiziert sein und zur Anzeige verwendet werden, dass SMI3-Flits und nicht PCIe-Pakete auf den Spuren der Verbindung gesendet werden. In der traditionellen PCIe 128b/130b Codierung können gültige Sync-Kopfzeilen (z.B. 610) das Senden entweder eines 10b Musters auf allen Spuren der Verbindung (zur Anzeige, dass die Art von Nutzlast des Blocks der PCIe-Datenblock sein soll) oder eines 01b Musters auf allen Spuren der Verbindung (zur Anzeige, dass die Art von Nutzlast des Blocks der PCIe gereihte Blocksatz sein soll) enthalten. In einem Beispiel einer SML kann eine alternative Sync-Kopfzeile zur Unterscheidung eines SMI3-Flit Verkehrs von PCIe-Datenblöcken und gereihten Sätzen definiert sein. In einem Beispiel, das in **Fig.** 6 dargestellt ist, kann die PCIe 128b/130b Sync-Kopfzeile (z.B. 605a, 605b) mit abwechselnden 01b, 10b Mustern auf ungeraden/-geraden Spuren codiert sein um anzugeben, dass SMI3-Flits gesendet werden sollen. In einer anderen alternativen Implementierung, kann die 128b/130b Sync-Kopfzeile, die für SMI3-Verkehr codiert, durch abwechselnde 10b, 01b Muster auf ungeraden/geraden Spuren, neben anderen beispielhaften Codierungen, definiert sein. In einigen Fällen können SMI3-Flits unmittelbar nach der SMI3 Sync-Kopfzeile auf einer pro-Byte-Basis gesendet werden, wobei der Übergang zwischen PCIe- und SMI3-Protokollen an der Blockgrenze erfolgt.

[0053] In einigen Implementierungen, wie jenen, die in dem Beispiel von **Fig.** 6 gezeigt sind, kann der Übergang zwischen den Protokollen so definiert sein, dass er an der Blockgrenze erfolgt, unabhängig davon, ob er einer SMI3-Flit- oder PCIe-Paketgrenze entspricht. Zum Beispiel kann ein Block so definiert sein, dass er eine vordefinierte Datenmenge (z.B. 16 Symbole, 128 Bytes, usw.) enthält. Wenn in solchen Implementierungen die Blockgrenze nicht einer SMI3-Flit- oder PCIe-Paketgrenze entspricht, kann die Sendung eines gesamten SMI3-Flits unterbrochen werden. Ein unterbrochenes SMI3-Flit kann im nächsten SMI3-Block wiederaufgenommen werden, was durch das Senden einer anderen Sync-Kopfzeile angezeigt wird, die für SMI3 codiert ist.

[0054] In Hinblick nun auf **Fig.** 7A ist eine Darstellung 700 gezeigt, die eine andere beispielhafte Implementierung einer SML veranschaulicht. In dem Beispiel von **Fig.** 7A können anstelle einer Verwendung einer spezialisierten Sync-Kopfzeile, die Signalübergänge zwischen Speicherzugangs- und Zwischenverbindungsprotokollverkehr codiert, physische Schicht-Framing-Token verwendet werden. Ein Framing-Token (oder „Token“) kann eine Dateneinkapselung auf der physischen Schicht sein, die die Anzahl von Symbolen spezifiziert oder impliziert, die in einem Datenstrom enthalten sein soll, der mit dem Token verknüpft ist. Folglich kann das Framing-Token angeben, dass ein Strom beginnt, wie auch implizieren, wo er enden wird, und kann daher auch zum Angeben der Stelle des nächsten Framing-Tokens verwendet werden. Ein Framing-Token eines Datenstroms kann sich im ersten Symbol (Symbol 0) der ersten Spur (z.B. Spur 0) des ersten Datenblocks des Datenstroms befinden. In dem Beispiel von PCIs können fünf Framing-Token definiert sein, die das TLP-Verkehrsstart- (STP) Token, Datenstromende- (EDS) Token, Schlechtes-Ende- (EDB) Token, DLLP-Start- (SDP) Token und logischen Leerlauf- (IDL) Token enthalten.

[0055] In dem Beispiel von **Fig.** 7A kann die SML durch Auflegen eines SMI3- oder anderen Datenzugangsprotokolls auf eine PCIe implementiert sein und der Standard-PCIe STP-Token kann so modifiziert sein, dass er einen neuen STP-Token definiert, der angibt, dass SMI3 (anstelle von TLP-Verkehr) auf den Spuren der Verbindung beginnen wird. In einigen Beispielen können Werte von Reserve-Bits des Standard-PCIe STP-Tokens modifiziert werden, um das SMI3 STP-Token in SML zu definieren. Wie ferner in **Fig.** 7B dargestellt ist, kann ein STP-Token 705 mehrere Felder enthalten, einschließlich eines 710 Feldes, das die Länge der SMI3-Nutzlast (im Sinne der Anzahl von Flits) angibt, die folgen wird. In einigen Implementierungen können eine oder mehrere Standardnutzlastlängen für TLP-Daten definiert sein. SMI3-Daten können in einigen Implementierungen so definiert sein, dass sie eine festgesetzte Anzahl von Flits enthalten, oder in anderen Fällen eine variable Anzahl von Flits enthalten, wobei in diesem Fall das Längenfeld für die Anzahl von SMI3-Flits ein Feld wird, das vernachlässigt werden kann. Ferner kann das Längenfeld für ein SMI3 STP als eine Länge definiert sein, die sich von jener der definierten TLP-Nutzlastlängen unterscheidet. Daher kann ein SMI3 STP, als ein Beispiel, auf der Basis eines Nicht-TLP-Längenwerts angegeben sein, der im STP-Längenfeld vorhanden ist. Zum Beispiel können in einer Implementierung die oberen 3-Bits des 11-Bit STP-Längenfeldes auf 111b gesetzt sein, um das SMI3 Paket anzugeben (z.B. basierend auf der Annahme, dass kein spezifikationskonformes PCIe TLP lang genug sein kann, um eine Länge aufzuweisen, wo die oberen 3 Bits des Längenfelds jeweils zu „1“ führen würden). Andere Implementierungen können andere Felder des STP-Tokens verändern oder codieren, um ein PCIe STP-Token, das eine traditionelle PCIe TLP-Datennutzlast identifiziert, von einem SMI3 STP-Token zu unterscheiden, das angibt, dass SMI3-Daten in TLP-Daten eingekapselt sind.

[0056] Unter erneuter Bezugnahme auf das Beispiel von **Fig.** 7A können, Sync-Kopfzeilendaten der Codierung folgen, die für eine traditionelle PCIe 128b/130b Codierung spezifiziert ist. Zum Beispiel werden bei 715a-c Sync-Kopfzeilen mit einem Wert

10b empfangen, die anzeigen, dass Datenblöcke bevorstehen. Wenn ein PCIe STP (z.B. 720) empfangen wird, wird eine PCIe-TLP-Nutzlast erwartet, und der Datenstrom wird entsprechend verarbeitet. In Übereinstimmung mit der Nutzlastlänge, die im PCIe STP 720 identifiziert ist, kann die PCIe-TLP-Nutzlast die volle zugewiesene Nutzlastlänge nutzen. Ein anderes STP-Token kann im Wesentlichen zur selben Zeit innerhalb eines Datenblocks empfangen werden, der dem Ende der TLP-Nutzlast folgt. Zum Beispiel kann bei 725 ein SMI3 STP empfangen werden, das einen Übergang von PCIe-TLP-Daten zu SMI3-Flit Daten signalisiert. Das SMI3 STP kann zum Beispiel gesendet werden, sobald ein Ende der PCIe-Paketdaten identifiziert ist.

[0057] In Fortsetzung mit dem Beispiel von **Fig.** 7A kann, wie bei den PCIe-TLP-Daten, das SMI3 STP 725 eine Länge der SMI3-Flit-Nutzlast definieren, die folgen soll. Zum Beispiel kann die Nutzlastlänge der SMI3-Daten der Anzahl von SMI3-Flits im Sinne von DW entsprechen, die folgen sollen. Ein Fenster (das z.B. bei Symbol 15 von Spur 3 endet), das der Nutzlastlänge entspricht, kann dadurch auf den Spuren definiert sein, in welchen nur SMI3-Daten während des Fensters gesendet werden sollen. Wenn sich das Fenster schließt, können andere Daten gesendet werden, wie ein anderes PCIe STP, das wiederbeginnt, TLP-Daten oder andere Daten, wie einen gereihten Datensatz, zu senden. Wie zum Beispiel in dem Beispiel von **Fig.** 7A dargestellt, wird ein EDS-Token nach dem Ende des SMI3-Datenfensters gesendet, das durch das SMI3 STP-Token 725 definiert ist. Das EDS-Token kann das Ende des Datenstroms signalisieren und implizieren, dass ein gereihter Blocksatz folgen wird, wie im Falle des Beispiels von **Fig.** 7A. Eine Sync-Kopfzeile 740, die 01b codiert ist, wird gesendet um anzugeben, dass ein gereihter Blocksatz gesendet werden soll. In diesem Fall wird ein gereihter PCIe-SKP-Satz gesendet. Solche gereihten Sätze können periodisch oder in eingestellten Intervallen oder Fenstern gesendet werden, so dass verschiedene Aufgaben auf PHY-Ebene und eine Koordination durchgeführt werden können, einschließlich eines Initialisierens einer Bit-Ausreichung, eines Initialisierens einer Symbolausrichtung, eines Austausches von PHY-Parametern, eines Kompensierens verschiedener Bit-Raten für zwei kommunizierende Ports, neben anderen Beispielen. In einigen Fällen kann ein angeordneter gereihter Satz gesendet werden, um ein definiertes Fenster oder einen Datenblock, der für SMI3-Flit-Daten spezifiziert ist, durch ein entsprechendes SMI3 STP-Token zu unterbrechen.

[0058] Obwohl im Beispiel von **Fig.** 7A nicht ausdrücklich dargestellt, kann ein STP-Token auch für einen Übergang von SMI3-Flit Daten auf der Verbindung zu PCIe-TLP-Daten verwendet werden. Zum Beispiel kann nach dem Ende eines definierten SMI3 Fensters ein PCIe STP-Token (z.B. ähnlich dem Token 720) gesendet werden um anzuzeigen, dass das nächste Fenster zum Senden einer spezifizierten Menge an PCIe-TLP-Daten dient.

[0059] Speicherzugangs-Flits (z.B. SMI3-Flits) können in einigen Ausführungsformen in der Größe variieren, wodurch es schwierig wird vorherzusagen, wie viele Daten im entsprechenden STP-Token (z.B. SMI3 STP-Token) für die Speicherzugangsnutzlast zu reservieren sind. Als ein Beispiel, wie in **Fig.** 7 dargestellt, kann das SMI3 STP 725 ein Längenfeld haben, das anzeigt, dass erwartet wird, dass 244 Bytes SMI3-Daten dem SMI3 STP 725 folgen. In diesem Beispiel jedoch sind nur zehn Flits (z.B. SMI3-Flits 0-9) zur Sendung während des Fensters bereit und diese zehn SMI3-Flits verwenden nur 240 der 244 Bytes. Daher verbleiben vier (4) Bytes leere Bandbreite und diese werden mit IDL-Token gefüllt. Dies kann insbesondere suboptimal sein, wenn PCIe-TLP-Daten in einer Warteschlange sind und auf ein Schließen des SMI3-Fensters warten. In anderen Fällen kann das für die Sendung von SMI3-Flits vorgesehene Fenster unzureichend sein, um die Menge an SMI3-Daten zu senden, die für die Spur bereit ist. Es können Entscheidungstechniken zur Bestimmung verwendet werden, wie zwischen SMI3- und PCIe-TLP-Daten entschieden wird, die gleichzeitig auf der Verbindung vorhanden sind. Ferner kann in einigen Implementierungen die Länge der SMI3-Fenster dynamisch modifiziert werden, um eine effizientere Verwendung der Verbindung zu unterstützen. Zum Beispiel kann eine Entscheidungs- oder andere Logik überwachen, wie gut die definierten SMI3-Fenster zur Bestimmung genutzt werden, ob die definierte Fensterlänge besser angesichts der für die Spur erwarteten Menge an SMI3 (und konkurrierenden PCIe-TLP-Verkehr) optimiert werden kann. Daher können in solchen Implementierungen die Längenfeldwerte von SMI3 STP-Token abhängig von der Menge an Verbindungsbandbreite, der SMI3-Flit Daten zugewiesen werden sollten (z.B. relativ zu anderen PCIe-Daten, einschließlich TLP-, DLLP-Daten und eines gereihten Datensatzes), neben anderen Beispielen, dynamisch (z.B. zwischen verschiedenen Werte) eingestellt werden.

[0060] In Hinblick nun auf **Fig.** 8 ist eine Darstellung 800 einer anderen beispielhaften Implementierung einer SML dargestellt. In dieser alternativen Ausführungsform kann die SML verschachtelte SMI3- und PCIe-Protokolle durch ein modifiziertes PCIe-Framing-Token vorsehen. Wie oben festgehalten wurde, kann ein EDS-Token in der PCIe zum Anzeigen eines Endes eines Datenstroms und zum Anzeigen, dass der nächste Block ein gereihter Blocksatz sein wird, verwendet werden. In dem Beispiel von **Fig.** 8 kann die SML eine SMI3 EDS-Token (z.B. 805) definieren, der das Ende eines TLP-Datenstroms und den Übergang zu SMI3-Flit-Sendungen

anzeigt. Ein SMI3 EDS (z.B. 805) kann durch Codieren eines Teils der reservierten Bits des traditionellen EDS-Tokens definiert werden, um anzuzeigen, dass SMI3-Daten folgen werden und nicht gereihte PCIe-Sätze oder andere Daten, die einem PCIe EDS folgen. Anders als beim traditionellen EDS-Token kann der SMI3 EDS im Wesentlichen überall in einem PCIe-Datenblock gesendet werden. Dies kann eine zusätzliche Flexibilität beim Senden von SMI3-Daten und Aufnehmen entsprechender Transaktionen mit dem gemeinsam benutzten Speicher geringer Latenz ermöglichen. Zum Beispiel kann ein Übergang von PCIe zu SMI3 mit einem einzelnen Doppelwort (DW) an Mehraufwand erfolgen. Ferner kann, wie bei traditionellen EDS-Token, ein beispielhaftes SMI3 EDS keine Länge spezifizieren, die mit den SMI3-Daten verknüpft ist, die dem Token folgen sollen. Nach einem SMI3 EDS können PCIe-TLP-Daten enden und SMI3-Flits auf der Verbindung fortfahren. Der SMI3-Verkehr kann fortfahren, bis die SMI3-Logik die Steuerung zur PCIe-Logik zurückstellt. In einigen Implementierungen bewirkt ein Senden eines SMI3 EDS, dass die Steuerung von der PCIe-Logik zur SMI3-Logik geht, die zum Beispiel auf Vorrichtungen vorgesehen ist, die mit der Verbindung verbunden sind.

[0061] In einem Beispiel kann SMI3 (oder ein anderes Protokoll) seine eigene Verbindungssteuerungssignalisierung zur Verwendung in der Durchführung einer Verbindungsschichtsteuerung definieren. Zum Beispiel kann in einer Implementierung die SML eine spezialisierte Version eines SMI3-Verbindungsschichtsteuerungs- (LLCTRL) Flits (z.B. 810) definieren, der einen Übergang vom SMI3- zurück zum PCIe-Protokoll anzeigt. Wie beim SMI3 EDS kann der definierte LLCTRL-Flit (z.B. 810) bewirken, dass die Steuerung von der SMI3-Logik zur PCIe-Logik zurückgeht. In einigen Fällen, wie im Beispiel von **Fig.** 8 gezeigt, kann der LLCTRL-Flit (z.B. 810) mit einer vordefinierten Anzahl von LLCTRL-Leerlauf- (LLCTRL-IDLE) Flits (z.B. 815) ausgestattet werden, bevor der Übergang zur PCIe vollendet wird. Zum Beispiel kann die Anzahl von LLCTRL-IDLE Flits 815, die zum Ausstatten des SMI3 LLCTRL-Flits 810 gesendet wird, von der Latenz zum Decodieren des definierten SMI3 LLCTRL-Flits 810, das den Übergang signalisiert, abhängen. Nach Vollendung des Übergangs zurück zur PCIe kann ein STP-Paket gesendet werden und TLP-Paketdaten können wieder auf der Verbindung unter der Steuerung der PCIe beginnen.

[0062] Es sollte klar sein, dass die hier beschriebenen Implementierungen als Beispiele vorgesehen sind, um gewisse Prinzipien und Merkmale zu zeigen, die in der Patentschrift offenbart sind. Es sollte klar sein, dass alternative Konfigurationen, Protokolle, und Architekturen (neben jenen, die spezifisch in den Beispielen besprochen sind) solche Prinzipien und Merkmale verwenden und anwenden können. Als ein Beispiel für eine Alternative kann ein PCIe-Speicherlese-/-schreibvorgang (z.B. anstelle eines SMI3-Protokolls) verwendet werden, der mit Verzeichnungsinformationen verstärkt ist. Die Verzeichnisinformationen können durch Reserve-Bits des PCIe-Pakets implementiert werden. In einem anderen Beispiel kann der CPU-Knoten eine Cache-Steuerung (z.B. als eine Alternative zu einer Steuerung des gemeinsam benutzten Speichers) zum Senden von Speicherlese-/-schreibtransaktionen auf einer PCIe-Verbindung verwenden, zum Beispiel auf der Basis einer fernen Adressenbereichsprüfung, unter anderen möglichen Beispielen und Alternativen.

[0063] In Hinblick nun auf **Fig.** 9A-9D sind Ablaufdiagramme 900a-d dargestellt, die beispielhafte Techniken zur Kommunikation unter Verwendung eines MCPL zeigen. Zum Beispiel kann in **Fig.** 9A eine Last-/Speicher-Speicherzugangsnachricht von einem ersten Knoten empfangen werden 905, wobei die Nachricht bestimmte Daten eines gemeinsam benutzten Speichers anfragt. Ein Zugang zu den bestimmten Daten kann für den ersten Knoten vorgesehen werden 910. Eine zweite Last-/Speicher-Speicherzugangsnachricht kann von einem zweiten unabhängigen Knoten empfangen werden 915. Die zweite Nachricht kann eine Anfrage für einen Zugang zu denselben bestimmten Daten des gemeinsam benutzten Speichers sein und ein Zugang zu den bestimmten Daten kann für den zweiten Knoten vorgesehen werden 920. Daten im gemeinsam benutzten Speicher können somit von mehrfachen verschiedenen unabhängigen Knoten gemeinsam benutzt werden und für diese zugänglich sein.

[0064] In dem Beispiel von **Fig.** 9B kann eine erste Sync-Kopfzeile (wie eine PCIe-Sync-Kopfzeile) mit einer ersten Codierung empfangen werden 925. Die Codierung kann einen Übergang von einem Zwischenverbindungsprotokoll zu einem Speicherzugangsprotokoll anzeigen und der Übergang kann aus der ersten Sync-Kopfzeile identifiziert werden 930. Daten des Speicherzugangsprotokolls können nach der ersten Sync-Kopfzeile empfangen werden und die Daten können verarbeitet werden 935 (z.B. in Übereinstimmung mit dem Speicherzugangsprotokoll). In einigen Beispielen können die Speicherzugangsprotokolldaten Transaktionen enthalten, die einen gemeinsam benutzten Speicher beinhalten, der von mehreren mehrfacher unabhängigen Knoten gemeinsam benutzt wird. Eine zweite Sync-Kopfzeile kann empfangen werden 940, die eine zweite, andere Codierung enthält, die einen Übergang vom Zwischenverbindungsprotokoll anzeigt. Der Übergang vom Speicherzugangsprotokoll zurück zum Zwischenverbindungsprotokoll kann aus der zweiten Sync-Kopfzeile identifiziert werden 945.

[0065] In Hinblick nun auf **Fig.** 9C kann in einigen Fällen ein erster Datenstart-Token (z.B. ein PCIe STP-Token) empfangen werden 950, der einen oder mehrere Werte enthält, die zum Identifizieren eines Übergangs von einem Zwischenverbindungsprotokoll zu einem Speicherzugangsprotokoll codiert sind. Daten des Speicherzugangsprotokolls können nach dem ersten Datenstart-Token eintreffen und können identifiziert werden 955. Die Daten des Speicherzugangsprotokolls können verarbeitet werden 960. Ein Längenfeld kann im ersten Datenstart-Token enthalten sein, das anzeigt, wann Daten zu Zwischenverbindungsprotokolldaten zurückgehen. Tatsächlich kann in einigen Implementierungen das Längenfeld eines Datenstart-Tokens codiert sein, um eine Länge anzuzeigen, die den Daten des Speicherzugangsprotokolls entspricht. Ferner kann ein zweites, anderes Datenstart-Framing-Token definiert sein, das so zu interpretieren ist, dass er einem Eintreffen von Daten des Zwischenverbindungsprotokolls entspricht. Jeder von dem ersten und zweiten Datenstart-Framing-Token kann nach dem Zwischenverbindungsprotokoll (z.B. PCIe), neben anderen Beispielen, definiert sein.

[0066] In dem Beispiel von **Fig.** 9D kann ein Stromende-Token (z.B. ein spezialisierter PCIe EDS-Token) empfangen werden 965, der zum Anzeigen eines Übergangs zu Speicherzugangsprotokolldaten codiert ist. Das empfangene Stromende-Token kann einen Übergang 970 von der Verbindungsschichtlogik zur Verarbeitung von Zwischenverbindungsprotokolldaten zur Verbindungsschichtlogik zur Verarbeitung von Speicherzugangsprotokolldaten veranlassen. Daten des Speicherzugangsprotokolls können empfangen werden 975 und unter Verwendung der Verbindungsschichtlogik des Speicherzugangsprotokolls verarbeitet werden. Verbindungsschichtsteuerungsdaten des Speicherzugangsprotokolls können empfangen werden 980 (z.B. am Ende der Daten des Speicherzugangsprotokolls), um einen Übergang zu Daten des Zwischenverbindungsprotokolls anzuzeigen. Ein Empfang 980 der Verbindungsschichtsteuerungsdaten kann einen Übergang 985 von der Verbindungsschichtlogik des Speicherzugangsprotokolls zur Verbindungsschichtlogik des Zwischenverbindungsprotokolls veranlassen. Daten des Zwischenverbindungsprotokolls können nach den Verbindungsschichtsteuerungsdaten empfangen werden und können von der Verbindungsschichtlogik des Zwischenverbindungsprotokolls nach dem Übergang 985, neben anderen Beispielen, verarbeitet werden.

[0067] Es sollte festgehalten werden, dass, obwohl ein Großteil der Prinzipien und Beispiele im Zusammenhang mit PCIe und insbesondere Überarbeitungen der PCIe-Spezifikation beschrieben sind, die hier beschriebenen Prinzipien, Lösungen und Merkmale gleichermaßen bei anderen Protokollen und Systemen anwendbar sein können. Zum Beispiel können analoge Spurfehler in anderen Verbindungen, die andere Protokolle verwenden, auf der Basis analoger Symbole, Datenströme und Token, wie auch Regeln, die für die Verwendung, Anordnung und Formatierung solcher Strukturen innerhalb von Daten spezifiziert sind, die über dies anderen Verbindungen gesendet werden, erfasst werden. Ferner können alternative Mechanismen und Strukturen (z.B. neben einem PCIe LES Register oder SKP OS) zum Vorsehen einer Spurfehlererfassung und Meldefunktionalität in einem System verwendet werden. Ferner können Kombinationen der obengenannten Lösungen in Systemen angewendet werden, einschließlich, neben anderen Beispielen, Kombinationen logischer und physischer Verstärkungen an einer Verbindung und ihrer entsprechenden Logik wie hier beschrieben.

[0068] Es ist zu beachten, dass die oben beschriebenen Apparate, Verfahren und Systeme in jeder elektronischen Vorrichtung oder jedem System, wie oben erwähnt, implementiert sein können. Als spezielle Veranschaulichung sehen die folgenden Figuren beispielhafte Systeme zur Nutzung der Erfindung, wie hier beschrieben, vor. Da die folgenden Systeme ausführlicher beschrieben sind, ist eine Anzahl von verschiedenen Zwischenverbindungen offenbart, beschrieben und aus der vorangehenden Besprechung wieder erwähnt. Und es ist sofort offensichtlich, dass die oben beschriebenen Weiterentwicklungen bei jeder dieser Zwischenverbindungen, Matrizen oder Architekturen angewendet werden können.

[0069] Unter Bezugnahme auf **Fig.** 10 ist eine Ausführungsform eines Blockdiagramms für ein Rechnersystem, das einen Mehrfachkern-Prozessor enthält, dargestellt. Der Prozessor 1000 enthält einen beliebigen Prozessor oder eine beliebige Verarbeitungsvorrichtung, wie einen Mikroprozessor, einen eingebetteten Prozessor, einen Digitalsignalprozessor (DSP), einen Netzwerkprozessor, einen in der Hand gehaltenen Prozessor, einen Anwendungsprozessor, einen Co-Prozessor, ein System auf einem Chip (SOC) oder eine andere Vorrichtung zur Ausführung eines Codes. Der Prozessor 1000 enthält in einer Ausführungsform zumindest zwei Kerne - Kern 1001 und 1002, die asymmetrische Kerne oder symmetrische Kerne (die dargestellte Ausführungsform) enthalten können. Der Prozessor 1000 kann jedoch eine beliebige Anzahl von Verarbeitungselementen enthalten, die symmetrisch oder asymmetrisch sein können.

[0070] In einer Ausführungsform bezieht sich ein Verarbeitungselement auf eine Hardware oder Logik zur Unterstützung eines Software-Thread. Beispiele von Hardware-Verarbeitungselementen ent-

halten: eine Thread-Einheit, einen Thread-Slot, einen Thread, eine Prozesseinheit, einen Kontext, eine Kontexteinheit, einen logischen Prozessor, einen Hardware-Thread, einen Kern und/oder jedes andere Element, das imstande ist, einen Zustand für einen Prozessor zu halten, wie einen Ausführungszustand oder architektonischen Zustand. Mit anderen Worten, ein Verarbeitungselement bezieht sich in einer Ausführungsform auf jede Hardware, die imstande ist, unabhängig mit einem Code, wie einem Software-Thread, Betriebssystem, einer Anwendung oder einem anderen Code verknüpft zu werden. Ein physischer Prozessor (oder eine Prozessorbuchse) bezieht sich typischerweise auf eine integrierte Schaltung, die möglicherweise eine beliebige Anzahl anderer Verarbeitungselemente, wie Kerne oder Hardware-Threads, enthält.

[0071] Ein Kern bezieht sich häufig auf eine Logik, die sich auf einer integrierten Schaltung befindet, die imstande ist einen unabhängigen architektonischen Zustand aufrechtzuerhalten, wobei jeder unabhängig aufrechterhaltene architektonische Zustand mit zumindest einigen zweckbestimmten Ausführungsressourcen verknüpft ist. Im Gegensatz zu Kernen bezieht sich ein Hardware-Thread typischerweise auf jede Logik, die sich auf einer integrierten Schaltung befindet, die imstande ist einen unabhängigen architektonischen Zustand aufrechtzuerhalten, wobei die unabhängig aufrechterhaltenen architektonischen Zustände gemeinsam einen Zugang zu Ausführungsressourcen benutzen. Wie erkennbar ist, wenn gewisse Ressourcen gemeinsam benutzt werden und andere einem architektonischen Zustand gewidmet sind, überlappt die Linie zwischen der Nomenklatur eines Hardware-Thread und Kerns. Dennoch werden häufig ein Kern und ein Hardware-Thread von einem Betriebssystem als einzelne logische Prozessoren angesehen, wo das Betriebssystem imstande ist, einen Betrieb auf jedem logischen Prozessor individuell zu planen.

[0072] Der physische Prozessor 1000, wie in **Fig.** 10 dargestellt, enthält zwei Kerne - Kern 1001 und 1002. Hier werden Kern 1001 und 1002 als symmetrische Kerne angesehen, d.h., Kerne mit denselben Konfigurationen, Funktionseinheiten und/oder derselben Logik. In einer anderen Ausführungsform enthält der Kern 1001 einen Out-of-Order-Prozessorkern, während der Kern 1002 einen In-Order-Prozessorkern enthält. Die Kerne 1001 und 1002 können jedoch einzeln aus j eder Art von Kern gewählt werden, wie einem nativen Kern, einem softwareverwalteten Kern, einem Kern, der dazu angepasst ist, eine native Befehlssatzarchitektur (Instruction Set Architecture, ISA) auszuführen, einem Kern, der dazu ausgebildet ist, eine übersetzte Befehlssatzarchitektur (ISA) auszuführen, einem co-gestalteten Kern oder einem anderen bekannten Kern. In einer heterogenen Kernumgebung (d.h., asymmetrische Kerne) kann eine gewisse Form von Übersetzung, wie eine binäre Übersetzung, zum Planen oder Ausführen eines Codes auf einem Kern oder beiden Kernen verwendet werden. Für eine nähere Besprechung sind die in Kern 1001 dargestellten Funktionseinheiten in der Folge ausführlich beschrieben, da die Einheiten im Kern 1002 auf ähnliche Weise in der dargestellten Ausführungsform arbeiten.

[0073] Wie dargestellt, enthält der Kern 1001 zwei Hardware-Threads 1001a und 1001b, die auch als Hardware-Thread-Slots 1001a und 1001b bezeichnet werden können. Daher sehen Software-Einheiten, wie ein Betriebssystem, in einer Ausführungsform möglicherweise den Prozessor 1000 als vier separate Prozessoren, d.h., vier logische Prozessoren oder Verarbeitungselemente, die imstande sind, gleichzeitig vier Software-Threads auszuführen. Wie oben angedeutet, ist ein erster Thread mit Architekturzustandsregistern 1001a verknüpft, ein zweiter Thread ist mit Architekturzustandsregistern 1001b verknüpft, ein dritter Thread kann mit Architekturzustandsregistern 1002a verknüpft sein und ein vierter Thread kann mit Architekturzustandsregistern 1002b verknüpft sein. Hier kann jedes der Architekturzustandsregister (1001a, 1001b, 1002a, und 1002b) als Verarbeitungselemente, Thread-Slots oder Thread-Einheiten bezeichnet werden, wie oben beschrieben. Wie dargestellt, sind die Architekturzustandsregister 1001a in Architekturzustandsregistern 1001b repliziert, so dass einzelne Architekturzustände/Kontexte für den logischen Prozessor 1001a und logischen Prozessor 1001b gespeichert werden können. Im Kern 1001 können auch kleinere Ressourcen, wie Befehlspointer und Renaming-Logik im Allocator- und Renamer-Block 1030 ebene für die Threads 1001a und 1001b repliziert sein. Einige Ressourcen, wie Rückordnungspuffer in der Rückordnungs-/Retirement-Einheit 1035, ILTB 1020, Last-/Speicher- Puffer und Warteschlangen können durch Trennung gemeinsam benutzt werden. Andere Ressourcen, wie interne Allzweckregister, Seiten-Tabelle-Basisregister, Daten-Cache auf niedriger Ebene und Daten-TLB 1015, Ausführungseinheit(en) 1040 und Teile einer Out-of-Order-Einheit 1035 werden möglicherweise zur Gänze gemeinsam benutzt.

[0074] Der Prozessor 1000 enthält häufig andere Ressourcen, die zur Gänze gemeinsam benutzt werden können, durch Trennung gemeinsam benutzt werden können oder durch/für Verarbeitungselemente zweckbestimmt sind. In **Fig.** 10 ist eine Ausführungsform eines rein beispielhaften Prozessors mit veranschaulichenden logischen Einheiten/Ressourcen eines Prozessors dargestellt. Es ist zu beachten, dass ein Prozessor jede dieser Funktionseinheiten enthalten oder auf diese verzichten kann, wie auch sämtliche andere bekannte Funktionseinheiten, Logik oder Firmware, nicht dargestellt, ent-

halten kann. Wie dargestellt, enthält der Kern 1001 einen vereinfachten, repräsentativen Out-of-Order-Prozessorkern (OOO). Es kann aber auch in verschiedenen Ausführungsformen ein In-Order-Prozessor verwendet werden. Der OOO-Kern enthält einen Abzweigungszielpuffer 1020 zur Vorhersage von Abzweigungen, die ausgeführt/genommen werden sollen, und einen Befehlsübersetzungspuffer (I-TLB) 1020 zum Speichern von Adressenübersetzungseinträgen für Befehle.

[0075] Der Kern 1001 enthält ferner ein Decodiermodul 1025, das an eine Abrufeinheit 1020 zum Decodieren abgerufener Elemente gekoppelt ist. Eine Abruflogik enthält in einer Ausführungsform einzelne Sequenzierer, die mit dem Thread-Slot 1001a bzw. 1001b verknüpft sind. Üblicherweise ist der Kern 1001 mit einer ersten ISA verknüpft, die Befehle definiert/spezifiziert, die auf dem Prozessor 1000 ausführbar sind. Andere Maschinencode-Befehle, die Teil der ersten ISA sind, enthalten einen Teil des Befehls (als Opcode bezeichnet), der einen auszuführenden Befehl oder eine auszuführende Operation angibt/spezifiziert. Die Decodierlogik 1025 enthält einen Schaltkreis, der diese Befehle aus ihren Opcodes erkennt und die decodierten Befehle in der Pipeline zur Verarbeitung weiterleitet, wie von der ersten ISA definiert. Wie zum Beispiel in der Folge ausführlicher besprochen ist, enthalten die Decodierer 1025 in einer Ausführungsform eine Logik, die zum Erkennen spezieller Befehle gestaltet oder angepasst ist, wie eines Transaktionsbefehls. Infolge des Erkennens seitens der Decodierer 1025 ergreift die Architektur oder der Kern 1001 spezielle, vordefinierte Maßnahmen zur Durchführung von Aufgaben, die mit dem passenden Befehl verknüpft sind. Es muss festgehalten werden, dass sämtliche hier beschriebene Aufgaben, Blöcke, Operationen und Verfahren als Reaktion auf einen einfachen oder mehrfachen Befehl durchgeführt werden können; von welchen einige neue oder alte Befehle sein können. Es ist zu beachten, dass in einer Ausführungsform die Decodierer 1026 dieselbe ISA (oder einen Teilsatz davon) erkennen. Alternativ erkennen die Decodierer 1026 in einer heterogenen Kernumgebung eine zweite ISA (entweder einen Teilsatz der ersten ISA oder eine andere ISA).

[0076] In einem Beispiel enthält der Allocator- und Renamer-Block 1030 einen Allocator zum Reservieren von Ressourcen, wie Registerdateien zum Speichern von Befehlsverarbeitungsergebnissen. Die Threads 1001a und 1001b sind jedoch möglicherweise zur einer Out-of-Order-Ausführung imstande, wo der Allocator- und Renamer-Block 1030 auch andere Ressourcen reserviert, wie Rückordnungspuffer zum Verfolgen von Befehlsergebnissen. Die Einheit 1030 kann auch einen Register-Renamer zum Umbenennen von Programm-Befehlsreferenzregistern in andere Register im Prozessor 1000 enthalten. Die Rückordnungs-/Retirement-Einheit 1035 enthält Komponenten, wie die obengenannten Rückordnungspuffer, Lastpuffer und Speicherpuffer, zur Unterstützung einer Out-of-Order-Ausführung und einer späteren In-Order-Verzögerung von Befehlen, die in einer anderen Reihenfolge ausgeführt werden.

[0077] Der Planer- und Ausführungseinheit(en)-Block 1040 enthält in einer Ausführungsform eine Planungseinheit zum Planen von Befehlen/Operationen auf Ausführungseinheiten. Zum Beispiel wird ein Fließkommabefehl an einem Port einer Ausführungseinheit geplant, die eine verfügbare Fließpunktausführungseinheit hat. Registerdateien, die mit den Ausführungseinheiten verknüpft sind, sind ebenso zum Speichern von Informationen von Befehlsverarbeitungsergebnissen enthalten. Beispielhafte Ausführungseinheitenenthalten eine Fließkommaausführungseinheit, eine Ganzzahlausführungseinheit, eine Sprungausführungseinheit, eine Lastausführungseinheit, eine Speicherausführungseinheit und andere bekannte Ausführungseinheiten.

[0078] Ein Daten-Cache auf niederer Ebene und ein Datenübersetzungspuffer (D-TLB) 1050 sind an (eine) Ausführungseinheit(en) 1040 gekoppelt. Der Daten-Cache dient zum Speichern kürzlich verwendeter/bearbeiteter Elemente, wie Datenoperanden, die möglicherweise in Speicherkohärenzzuständen gehalten werden. Der D-TLB dient zum Speichern kürzlich durchgeführter virtueller/linearer zu physischen Adressenübersetzungen. Als ein spezielles Beispiel kann ein Prozessor eine Seitentabellenstruktur enthalten, um einen physischen Speicher in mehrere virtuelle Seiten aufzubrechen.

[0079] Hier benutzen die Kerne 1001 und 1002 gemeinsam einen Zugang zu einem Cache höherer Ebene oder einem weiter außenliegenden Cache, wie einen Cache zweiter Ebene, der mit einer On-chip-Schnittstelle 1010 verknüpft ist. Es ist zu beachten, dass „höhere Ebene“ oder „weiter außenliegend“ sich auf Cache-Ebenen bezieht, die steigen oder weiter weg von der (den) Ausführungseinheit(en) kommen. In einer Ausführungsform ist ein Cache höherer Ebene ein Daten-Cache letzter Ebene - der letzte Cache in der Speicherhierarchie auf dem Prozessor 1000 - wie ein Daten-Cache zweiter oder dritter Ebene. Ein Cache höherer Ebene ist jedoch nicht darauf beschränkt, da er mit einem Befehls-Cache verknüpft sein oder diesen enthalten kann. Ein Trace-Cache - eine Art von Befehls-Cache - kann stattdessen nach dem Decodierer 1025 gekoppelt sein, um kürzlich decodierte Verfolgungen zu speichern. Hier bezieht sich ein Befehl möglicherweise auf einen Makrobefehl (d.h., einen allgemeinen Befehl, der von den Decodierern erkannt wird), der in einer Anzahl von Mikrobefehlen (Mikrooperationen) decodiert werden kann.

[0080] In der dargestellten Konfiguration enthält der Prozessor 1000 auch ein On-Chip-Schnittstellenmodul 1010. Früher war eine Speichersteuerung, die in der Folge ausführlicher beschrieben ist, in einem Rechnersystem extern zum Prozessor 1000 enthalten. In diesem Szenario soll eine On-Chip-Schnittstelle 1010 mit Vorrichtungen extern zum Prozessor 1000 kommunizieren, wie dem Systemspeicher 1075, einem Chipsatz (der häufig einen Memory Controller-Hub zur Verbindung mit einem Speicher 1075 und einen I/O Controller-Hub zur Verbindung mit peripheren Vorrichtungen enthält), einem Memory Controller-Hub, einer Northbridge oder einer anderen integrierten Schaltung. Und in diesem Szenario kann der Bus 1005 jede bekannte Zwischenverbindung, wie einen Multi-Drop-Bus, eine Punkt-zu-Punkt-Zwischenverbindung, eine serielle Zwischenverbindung, einen parallelen Bus, einen kohärenten (z.B. Cache-kohärenten) Bus, eine schichtenförmige Protokollarchitektur, einen Differential-Bus und einen GTL-Bus enthalten.

[0081] Der Speicher 1075 kann dem Prozessor 1000 gewidmet sein oder von anderen Vorrichtungen in einem System gemeinsam benutzt werden. Allgemeine Beispiele für Arten von Speicher 1075 enthalten DRAM, SRAM, nicht flüchtigen Speicher (NV Speicher) und andere bekannte Speichervorrichtungen. Es ist zu beachten, dass die Vorrichtung 1080 einen Grafikbeschleuniger, einen Prozessor oder eine Karte, die an einen Memory Controller-Hub gekoppelt ist, einen Datenspeicher, der an einen I/O Controller-Hub gekoppelt ist, einen drahtlosen Sender/Empfänger, eine Flash-Vorrichtung, eine Audio-Steuerung, eine Netzwerksteuerung oder eine andere bekannte Vorrichtung enthalten kann.

[0082] Da jedoch kürzlich immer mehr Logik und Vorrichtungen auf einem einzigen Die, wie SOC, integriert werden, kann jede dieser Vorrichtungen auf dem Prozessor 1000 enthalten sein. Zum Beispiel ist in einer Ausführungsform ein Memory Controller-Hub auf demselben Package und/oder Die wie der Prozessor 1000. Hier enthält ein Teil des Kerns (ein Teil auf dem Kern) 1010 eine oder mehrere Steuerung(en) zur Schnittstellenbildung mit anderen Vorrichtungen wie dem Speicher 1075 oder einer Grafikvorrichtung 1080. Die Konfiguration, die eine Zwischenverbindung und Steuerungen zur Schnittstellenbildung mit solchen Vorrichtungen enthält, wird häufig als „auf dem Kern“ (oder UnCore-Konfiguration) bezeichnet. Als ein Beispiel enthält eine On-Chip-Schnittstelle 1010 eine Ringzwischenverbindung für eine On-Chip-Kommunikation und eine Hochgeschwindigkeits-, serielle, Punkt-zu-Punkt-Verbindung 1005 für eine Off-Chip-Kommunikation. Dennoch können in der SOC-Umgebung noch mehr Vorrichtungen, wie die Netzwerkschnittstelle, die Co-Prozessoren, der Speicher 1075, der Grafikprozessor 1080 und sämtliche anderen bekannten Computervorrichtungen/Schnittstellen auf einem einzelnen Die oder einer einzelnen integrierten Schaltung integriert sein, um einen kleinen Formfaktor mit hoher Funktionalität und geringem Stromverbrauch vorzusehen.

[0083] In einer Ausführungsform ist der Prozessor 1000 imstande, einen Compiler, eine Optimierung und/oder einen Übersetzercode 1077 auszuführen, um einen Anwendungscode 1076 zu kompilieren, zu übersetzen und/oder zu optimieren, um den hier beschriebenen Apparat und das hier beschriebene Verfahren oder die Schnittstelle mit diesem zu unterstützen. Ein Compiler enthält häufig ein Programm oder einen Satz von Programmen zum Übersetzen von Quelltext/-code in einen Zieltext/-code. Üblicherweise erfolgt ein Kompilieren eines Programm-/Anwendungscodes mit einem Compiler in mehrfachen Phasen und läuft zur Umwandlung eines Programmiersprachencodes hoher Ebene in einen Maschinen- Assembly-Sprachcode niederer Ebene. Einzeldurchgang-Compiler können jedoch noch immer für ein einfaches Kompilieren verwendet werden. Ein Compiler kann sämtliche bekannte Kompiliertechniken verwenden und sämtliche bekannte Compiler-Operationen ausführen, wie lexikalische Analyse, Vorverarbeitung, Syntaxanalyse, semantische Analyse, Codegenerierung, Codeumformung und Codeoptimierung.

[0084] Größere Compiler enthalten häufig mehrfache Phasen, aber besonders häufig sind diese Phasen in zwei allgemeinen Phasen enthalten: (1) einem Front-End, d.h., wo im Allgemeinen eine syntaktische Verarbeitung, semantische Verarbeitung und eine gewisse Umwandlung/Optimierung stattfinden kann, und (2) einem Back-End, d.h., wo im Allgemeinen eine Analyse, Umformungen, Optimierungen und Codegenerierung stattfinden. Einige Compiler beziehen sich auf eine Mitte, die das Verschwimmen einer Abgrenzung zwischen einem Front-End und Back-End eines Compilers veranschaulicht. Infolgedessen kann eine Bezugnahme auf ein Einsetzen, Verknüpfen, Generieren oder einen anderen Betrieb eines Compilers in jeder der obengenannten Phasen oder jedem der Durchläufen stattfinden, wie auch in sämtlichen anderen bekannten Phasen oder Durchläufen eines Compilers. Als ein veranschaulichendes Beispiel setzt ein Compiler möglicherweise Operationen, Anrufe, Funktionen usw. in einer oder mehreren Phasen des Kompilierens ein, wie ein Einsetzen von Anrufen/Operationen in einer Front-EndPhase des Kompilierens und ein anschließendes Umformen der Anrufe/Operationen in einen Code tieferer Ebene während einer Umformungsphase. Es ist zu beachten, dass während des dynamischen Kompilierens der Compilercode oder dynamische Optimierungscode solche Operationen/Anrufe einsetzen kann, wie auch den Code zur Ausführung während der Laufzeit optimieren kann. Als ein spezielles ver-

anschaulichendes Beispiel kann der binäre Code (bereits kompilierte Code) während der Laufzeit dynamisch optimiert werden. Hier kann der Programmcode den dynamischen Optimierungscode, den binären Code oder eine Kombination davon enthalten.

[0085] Ähnlich einem Compiler übersetzt ein Übersetzer, wie ein binärer Übersetzer, den Code entweder statisch oder dynamisch, um den Code zu optimieren und/oder zu übersetzen. Daher kann sich ein Verweis auf eine Ausführung eines Codes, eines Anwendungscodes, eines Programmcodes oder einer anderen Software-Umgebung auf folgendes beziehen: (1) Ausführung eines oder mehrerer Compilerprogramme, Optimierungscode-Optimierer oder Übersetzer, entweder dynamisch oder statisch, um den Programcode zu kompilieren, um die Software-Strukturen aufrechtzuerhalten, um andere Operationen auszuführen, um den Code zu optimieren oder den Code zu übersetzen; (2) Ausführung eines Hauptprogrammcodes, der Operationen/Anrufe enthält, wie eines Anwendungscodes der optimiert/kompiliert wurde; (3) Ausführung eines anderen Programmcodes, wie Bibliotheken, der mit dem Hauptprogrammcode verknüpft ist, um Software-Strukturen aufrechtzuerhalten, andere softwarebezogene Operationen auszuführen oder den Code zu optimieren; oder (4) eine Kombination davon.

[0086] Unter Bezugnahme nun auf **Fig.** 11 ist ein Blockdiagramm einer Ausführungsform eines Mehrfachkern-Prozessors dargestellt. Wie in der Ausführungsform von **Fig.** 11 gezeigt, enthält der Prozessor 1100 mehrfache Domäne. Im Speziellen enthält eine Kerndomäne 1130 mehrere Kerne 1130A-1130N, eine Grafikdomäne 1160 enthält eine oder mehrere Grafikmaschinen mit einer Medienmaschine 1165, und eine Systemagentendomäne 1110.

[0087] In verschiedenen Ausführungsformen behandelt die Systemagentendomäne 1110 Leistungssteuerungsereignisse und Leistungsmanagement, so dass einzelne Einheiten von Domänen 1130 und 1160 (z.B. Kerne und/oder Grafikmaschinen) unabhängig steuerbar sind, um angesichts der Aktivität (oder Inaktivität), die in einer bestimmten Einheit auftritt dynamisch in einem passenden Leistungsmodus/auf einer passenden Leistungsebene zu arbeiten (z.B. im Aktiv-, Turbo-, Schlaf-, Winterschlaf-, Tiefschlaf- oder einem anderen Advanced Configuration Power Interface-artigen Zustand). Jede der Domäne 1130 und 1160 kann bei einer anderen Spannung und/oder Leistung arbeiten und ferner arbeiten die einzelnen Einheiten innerhalb der Domäne möglicherweise jeweils bei einer unabhängigen Frequenz und Spannung. Es ist zu beachten, dass, während nur drei Domänen dargestellt sind, ein Verständnis des Umfangs der vorliegenden Erfindung in dieser Hinsicht nicht beschränkt ist und zusätzliche Domäne in anderen Ausführungsformen vorhanden sein können.

[0088] Wie dargestellt, enthält jeder Kern 1130 ferner Caches tiefer Ebene zusätzlich zu verschiedenen Ausführungseinheiten und zusätzlichen Verarbeitungselementen. Hier sind die verschiedenen Kerne aneinander und an einen gemeinsam benutzten Cache-Speicher gekoppelt, der aus mehreren Einheiten oder Slices eines Cache letzter Ebene (LLC) 1140A-1140N gebildet ist; diese LLCs enthalten häufig eine Speicher- und Cache-Steuerungsfunktionalität und werden von den Kernen, wie auch möglicherweise von der Grafikmaschine gemeinsam benutzt.

[0089] Wie erkennbar ist, koppelt eine Ringzwischenverbindung 1150 die Kerne aneinander und sieht eine Zwischenverbindung zwischen der Kerndomäne 1130, der Grafikdomäne 1160 und dem Systemagentenschaltkreis 1110 über mehrere Ringstopps 1152A-1152N vor, jeden an einer Kopplung zwischen einem Kern und LLC-Slice. Wie in **Fig.** 11 erkennbar ist, wird die Zwischenverbindung 1150 zum Befördern verschiedener Informationen verwendet, einschließlich Adressinformationen, Dateninformationen, Bestätigungsinformationen und Snoop-/ungültigen Informationen. Obwohl eine Ringzwischenverbindung dargestellt ist, kann jede bekannte On-Die-Zwischenverbindung oder Matrix verwendet werden. Als ein veranschaulichendes Beispiel können einige der oben besprochenen Matrizen (z.B. eine andere On-Die-Zwischenverbindung, eine On-Chip-Systemmatrix (OSF), eine Advanced Mirocontroller Bus Architecture- (AMBA) Zwischenverbindung, eine multidimensionale Netzmatrix oder eine andere Zwischenverbindungsarchitektur) auf gleiche Weise verwendet werden.

[0090] Wie ferner dargestellt ist, enthält die Systemagentendomäne 1110 eine Anzeigemaschine 1112, die eine Steuerung von einer und eine Schnittstelle für eine zugehörige(n) Anzeige bereitstellen soll. Die Systemagentendomäne 1110 kann andere Einheiten enthalten, wie: eine integrierte Speichersteuerung 1120, die eine Schnittstelle für eine Systemspeicher- (z.B. einen DRAM, implementiert mit mehrfachen DIMMs) Kohärenzlogik 1122 zur Durchführung von Speicherkohärenzoperationen vorsieht. Mehrere Schnittstellen können vorhanden sein, um eine Zwischenverbindung zwischen dem Prozessor und einem anderen Schaltkreis zu ermöglichen. Zum Beispiel ist in einer Ausführungsform zumindest eine direkte Medienschnittstelle- (DMI) 1116 Schnittstelle vorgesehen, wie auch eine oder mehrere PCIe™ Schnittstellen 1114. Die Anzeigemaschine und diese Schnittstellen sind typischerweise über eine PCIe™ Brücke 1118 an einen Speicher gekoppelt. Ferner können zum Vorsehen von Kommunikationen zwischen anderen Agenten, wie zusätzlichen

Prozessoren oder einem anderen Schaltkreis, eine oder mehrere andere Schnittstelle(n) vorgesehen sein.

[0091] Unter Bezugnahme nun auf **Fig.** 12 ist ein Blockdiagramm eines repräsentativen Kerns dargestellt; im Speziellen logische Blöcke eines Back-End eines Kerns, wie des Kerns 1130 von **Fig.** 11. Im Allgemeinen enthält die in **Fig.** 12 dargestellte Struktur einen Out-of-Order-Prozessor, der eine Front-End-Einheit 1270 hat, die zum Abrufen eintretender Befehle, Durchführen verschiedener Verarbeitungen (z.B. Caching, Decodieren, Abzweigungsvorhersage, usw.) und Weiterleiten von Befehlen/Operationen zu einer Out-of-Order-Maschine (OOO) 1280 verwendet wird. Die OOO-Maschine 1280 führt eine Weiterverarbeitung an decodierten Befehlen durch.

[0092] Im Speziellen in der Ausführungsform von **Fig.** 12 enthält eine Out-of-Order-Maschine 1280 eine Zuordnungseinheit 1282 für den Empfang decodierter Befehle, die die Form eines Mikrobefehls oder mehrerer Mikrobefehle oder uops aufweisen kann, von der Front-End-Einheit 1270 und zu deren Zuordnung zu geeigneten Ressourcen wie Register und so weiter. Anschließend werden die Befehle für eine Reservierungsstation 1284 vorgesehen, die Ressourcen reserviert und sie zur Ausführung auf einer oder mehreren Ausführungseinheit(en) 1286A-1286N plant. Verschiedene Arten von Ausführungseinheiten können vorhanden sein, einschließlich zum Beispiel arithmetischer Logikeinheiten (ALUs), Last- und Speichereinheiten, Vektorverarbeitungseinheiten (VPUs), Fließkommaausführungseinheiten, unter anderen. Ergebnisse von diesen verschiedenen Ausführungseinheiten werden einem Rückordnungspuffer (ROB) 1288 bereitgestellt, der ungereihte Ergebnisse nimmt und sie in die korrekte Programmreihenfolge bringt.

[0093] Unter weiterer Bezugnahme auf **Fig.** 12 ist zu beachten, dass sowohl die Front-End-Einheit 1270 wie auch die Out-of-Order-Maschine 1280 an verschiedene Ebenen einer Speicherhierarchie gekoppelt sind. Im Speziellen ist ein Befehlsebenen-Cache 1272 dargestellt, der seinerseits an einen Mittelebenen-Cache 1276 gekoppelt ist, der seinerseits an einen Cache letzter Ebene 1295 gekoppelt ist. In einer Ausführungsform ist der Cache letzter Ebene 1295 in einer On-Chip-Einheit 1290 (manchmal als UnCore bezeichnet) implementiert. Als ein Beispiel ist die Einheit 1290 dem Systemagenten 1110 von **Fig.** 11 ähnlich. Wie oben besprochen, kommuniziert der UnCore 1290 mit dem Systemspeicher 1299, der in der dargestellten Ausführungsform durch einen ED RAM implementiert ist. Es ist auch zu beachten, dass die verschiedenen Ausführungseinheiten 1286 in einer Out-of-Order-Maschine 1280 mit einem Cache der ersten Ebene 1274 in Kommunikation stehen, der auch mit dem Mittelebenen-Cache 1276 in Kommunikation steht. Es ist auch zu beachten, dass zusätzliche Kerne 1230N-2 - 1230N an den LLC 1295 gekoppelt sein können. Obwohl in der Ausführungsform von **Fig.** 12 bei dieser hohen Ebene dargestellt, ist klar, dass verschiedene Änderungen und zusätzliche Komponenten vorhanden sein können.

[0094] In Hinblick nun auf **Fig.** 13 ist ein Blockdiagramm eines beispielhaften Computersystems gemäß einer Ausführungsform der vorliegenden Erfindung dargestellt, das mit einem Prozessor gebildet ist, der Ausführungseinheiten zur Ausführung eines Befehls enthält, wo eine oder mehrere der Zwischenverbindungen ein oder mehrere Merkmal(e) implementieren. Das System 1300 enthält eine Komponente, wie einen Prozessor 1302, zur Verwendung der Ausführungseinheiten, einschließlich einer Logik zur Durchführung von Algorithmen zur Verarbeitung von Daten gemäß der vorliegenden Erfindung, wie in der hier beschriebenen Ausführungsform. Das System 1300 ist für Verarbeitungssysteme repräsentativ, die auf PENTIUM III™, PENTIUM 4™, Xeon™, Itanium, XScale™ und/oder StrongARM™ Mikroprozessoren basieren, obwohl andere Systeme (einschließlich PCs mit anderen Mikroprozessoren, Engineering-Workstations, Set-Top-Boxes und dergleichen) ebenso verwendet werden können. In einer Ausführungsform führt ein Sample-System 1300 eine Version des WINDOWS™ Betriebssystems durch, die von der Mikrosoft Corporation of Redmond, Washington, erhältlich ist, obwohl andere Betriebssysteme (UNIX und Linux zum Beispiel), eingebettete Software und/oder grafische Anwenderschnittstellen ebenso verwendet werden können. Somit sind Ausführungsformen der vorliegenden Erfindung nicht auf eine spezielle Kombination von Hardware-Schaltkreis und Software begrenzt.

[0095] Ausführungsformen sind nicht auf Computersysteme beschränkt. Alternative Ausführungsformen der vorliegenden Erfindung können in anderen Vorrichtungen, wie in in der Hand gehaltenen Vorrichtungen und eingebetteten Anwendungen, verwendet werden. Einige Beispiele für in der Hand gehaltene Vorrichtungen enthalten Mobiltelefone, Internetprotokollvorrichtungen, Digitalkameras, persönliche digitale Assistenten (PDAs) und in der Hand gehaltene PCs. Eingebettete Anwendungen können eine Mikrosteuerung, einen Digitalsignalprozessor (DSP), ein System auf einem Chip, einen Netzwerkcomputer (NetPC), Set-Top-Boxes, Netzwerk-Hubs, Weitverkehrsnetz- (WAN) Schalter oder jedes andere System enthalten, der einen oder mehrere Befehl(e) gemäß zumindest einer Ausführungsform ausführen kann.

[0096] In dieser dargestellten Ausführungsform enthält der Prozessor 1302 eine oder mehrere Ausführungseinheit(en) 1308 zum Implementieren eines Algorithmus, der zumindest einen Befehl aus-

führen soll. Eine Ausführungsform kann im Kontext eines einzigen Prozessor-Desktop- oder Serversystems beschrieben sein, aber alternative Ausführungsformen können in einem Multiprozessorsystem enthalten sein. Das System 1300 ist ein Beispiel für eine 'Hub'-Systemarchitektur. Das Computersystem 1300 enthält einen Prozessor 1302 zum Verarbeiten von Datensignalen. Der Prozessor 1302 enthält, als ein veranschaulichendes Beispiel, einen Mikroprozessor eines Rechners mit komplexem Befehlssatz (Complex Instruction Set Computer, CISC), einen Mikroprozessor eines Rechners mit reduziertem Befehlssatz (Reduced Instruction Set Computer, RISC), einen Mikroprozessor eines sehr langen Befehlsworts (Very Long Instruction Word, VLIW), einen Prozessor, der eine Kombination von Befehlssätzen, implementiert oder jede andere Prozessorvorrichtung, wie zum Beispiel einen Digitalsignalprozessor. Der Prozessor 1302 ist an einen Prozessorbus 1310 gekoppelt, der Datensignale zwischen dem Prozessor 1302 und anderen Komponenten im System 1300 sendet. Die Elemente des Systems 1300 (z.B. Grafikbeschleuniger 1312, Memory Controller-Hub 1316, Speicher 1320, I/O Controller-Hub 1324, drahtloser Sender/Empfänger 1326, Flash BIOS 1328, Netzwerksteuerung 1334, Audiosteuerung 1336, serieller Erweiterungsport 1338, I/O Steuerung 1340, usw.) führen ihre herkömmlichen Funktionen aus, die jenen, die mit dieser Technik vertraut sind, allgemein bekannt sind.

[0097] In einer Ausführungsform enthält der Prozessor 1302 einen internen Cache-Speicher 1304 der Ebene 1 (L1). Abhängig von der Architektur kann der Prozessor 1302 einen einzelnen internen Cache oder mehrfache Ebenen interner Caches enthalten. Andere Ausführungsformen enthalten eine Kombination aus sowohl internen wie auch externen Caches, abhängig von der besonderen Implementierung und den Bedürfnissen. Die Registerdatei 1306 dient zum Speichern verschiedener Arten von Daten in verschiedenen Registern einschließlich Ganzzahlregister, Fließkommaregister, Vektorregister, Banked-Register, Schattenregister, Prüfpunktregister, Statusregister, und Befehlspointerregister.

[0098] Die Ausführungseinheit 1308, die eine Logik zur Durchführung von Ganzzahl- und Fließkommaoperationen enthält, befindet sich auch im Prozessor 1302. Der Prozessor 1302 enthält in einer Ausführungsform einen Mikrocode (ucode) ROM zum Speichern eines Mikrocodes, der, wenn er ausgeführt wird, Algorithmen für gewisse Makrobefehle durchführt, oder zum Bewältigen komplexer Szenarien. Hier ist der Mikrocode möglicherweise aktualisierbar, um Logik-Bugs/Fixes für den Prozessor 1302 zu handhaben. Für eine Ausführungsform enthält die Ausführungseinheit 1308 eine Logik zur Behandlung eines gepackten Befehlssatzes 1309. Durch Aufnahme des gepackten Befehlssatzes 1309 in den Befehlssatz eines Allzweckprozessors 1302, gemeinsam mit dem zugehörigen Schaltkreis zur Ausführung der Befehle, können die Operationen, die von vielen Multimedienanwendungen verwendet werden, unter Verwendung gepackter Daten in einem Allzweckprozessor 1302 ausgeführt werden. Somit werden viele Multimedienanwendungen beschleunigt und effizienter unter Verwendung der vollen Breite eines Prozessordatenbusses zur Durchführung von Operationen an gepackten Daten ausgeführt. Dies behebt möglicherweise die Notwendigkeit, kleinere Einheiten von Daten über den Prozessordatenbus zu transferieren, um eine oder mehrere Operation(en), jeweils mit einem Datenelement, auszuführen.

[0099] Es können auch andere Ausführungsformen einer Ausführungseinheit 1308 in Mikrosteuerungen, eingebetteten Prozessoren, Grafikvorrichtungen, DSPs und anderen Arten von Logik Schaltungen verwendet werden. Das System 1300 enthält einen Speicher 1320. Der Speicher 1320 enthält eine dynamische Direktzugriffsspeicher- (DRAM) Vorrichtung, eine statische Direktzugriffsspeicher- (SRAM) Vorrichtung, eine Flash-Speichervorrichtung oder eine andere Speichervorrichtung. Der Speicher 1320 speichert Befehle und/oder Daten, die durch Datensignale repräsentiert werden, die vom Prozessor 1302 auszuführen sind.

[0100] Es ist zu beachten, dass jedes der obengenannten Merkmale oder jeder der obengenannten Aspekte der Erfindung auf einer oder mehreren Zwischenverbindung(en) benutzt werden kann, die in **Fig.** 13 dargestellt sind. Zum Beispiel implementiert eine On-Die-Zwischenverbindung (ODI), die nicht dargestellt ist, zum Koppeln interner Einheiten des Prozessors 1302, einen oder mehrere der oben beschriebenen Aspekte der Erfindung. Oder die Erfindung ist mit einem Prozessorbus 1310 (z.B. einer anderen bekannten, Hochleistungsrechnerzwischenverbindung), einem Speicherpfad hoher Bandbreite 1318 zum Speicher 1320, einer Punkt-zu-Punkt-Verbindung zum Grafikbeschleuniger 1312 (z.B. einer Peripheral Component Interconnect Express (PCIe) konformen Matrix), einer Controller-Hub-Zwischenverbindung 1322, einem I/O oder einer anderen Zwischenverbindung (z.B. USB, PCI, PCIe) zum Koppeln der anderen dargestellten Komponenten verbunden. Einige Beispiele für solche Komponenten enthalten die Audiosteuerung 1336, den Firmware-Hub (Flash BIOS) 1328, den drahtlosen Sender/Empfänger 1326, den Datenspeicher 1324, die althergebrachte I/O Steuerung 1310, die Benutzereingabe- und Tastaturschnittstellen 1342 enthält, einen seriellen Erweiterungsport 1338 wie ein serielles Bussystem (Universal Serial Bus, USB) und eine Netzwerksteuerung 1334. Die Datenspeichervorrichtung 1324 kann ein Festplattenlaufwerk, ein Diskettenlaufwerk, eine CD-ROM-Vorrichtung,

eine Flash-Speichervorrichtung oder eine andere Massenspeichervorrichtung enthalten.

[0101] Unter Bezugnahme nun auf **Fig.** 14, ist ein Blockdiagramm eines zweiten Systems 1400 gemäß einer Ausführungsform der vorliegenden Erfindung dargestellt. Wie in **Fig.** 14 dargestellt, ist ein Multiprozessorsystem 1400 ein Punkt-zu-Punkt-Zwischenverbindungssystem und enthält einen ersten Prozessor 1470 und einen zweiten Prozessor 1480, die über eine Punkt-zu-Punkt-Zwischenverbindung 1450 gekoppelt sind. Jeder der Prozessoren 1470 und 1480 kann eine Version eines Prozessors sein. In einer Ausführungsform sind 1452 und 1454 Teil einer seriellen, kohärenten Punkt-zu-Punkt-Zwischenverbindungsmatrix, wie einer Hochleistungsarchitektur. Infolgedessen kann die Erfindung innerhalb der QPI-Architektur implementiert werden.

[0102] Während nur zwei Prozessoren 1470, 1480 dargestellt sind, ist klar, dass der Umfang der vorliegenden Erfindung nicht darauf beschränkt ist. In anderen Ausführungsformen kann/können ein oder mehrere zusätzliche(r) Prozessor(en) in einem bestimmten Prozessor vorhanden sein.

[0103] Die Prozessoren 1470 und 1480 sind mit integrierten Speichersteuerungseinheiten 1472 bzw. 1482 dargestellt. Der Prozessor 1470 enthält auch als Teil seiner Bussteuerungseinheiten Punkt-zu-Punkt- (P-P) Schnittstellen 1476 und 1478; ebenso enthält der zweite Prozessor 1480 P-P-Schnittstellen 1486 und 1488. Die Prozessoren 1470, 1480 können Informationen über eine Punkt-zu-Punkt- (P-P) Schnittstelle 1450 unter Verwendung von P-P-Schnittstellenschaltungen 1478, 1488 austauschen. Wie in **Fig.** 14 dargestellt, koppeln IMCs 1472 und 1482 die Prozessoren an entsprechende Speicher, nämlich einen Speicher 1432 und einen Speicher 1434, die Teile des Hauptspeichers sein können, die lokal an den entsprechenden Prozessoren befestigt sind.

[0104] Die Prozessoren 1470, 1480 tauschen jeweils Informationen mit einem Chipsatz 1490 über einzelne P-P Schnittstellen 1452, 1454 unter Verwendung der Punkt-zu-Punkt-Schnittstellenschaltungen 1476, 1494, 1486, 1498 aus. Der Chipsatz 1490 tauscht auch Informationen mit einer Hochleistungsgrafikschaltung 1438 über eine Schnittstellenschaltung 1492 entlang einer Hochleistungsgrafikzwischenverbindung 1439 aus.

[0105] Ein gemeinsam benutzter Cache (nicht dargestellt) kann entweder im Prozessor oder außerhalb beider Prozessoren enthalten sein; aber dennoch mit den Prozessoren über eine P-P-Zwischenverbindung verbunden sein, so dass lokale Cache-Informationen entweder eines Prozessors oder beider Prozessoren im gemeinsam benutzten Cache gespeichert werden können, wenn ein Prozessor in eine leistungsarme Betriebsart gesetzt wird.

[0106] Der Chipsatz 1490 kann über eine Schnittstelle 1496 an einen ersten Bus 1416 gekoppelt sein. In einer Ausführungsform kann der erste Bus 1416 ein Peripheral Component Interconnect- (PCI) Bus oder ein Bus wie ein PCI Expressbus oder ein anderer I/O-Zwischenverbindungsbus der dritten Generation sein, obwohl der Umfang der vorliegenden Erfindung nicht darauf beschränkt ist.

[0107] Wie in **Fig.** 14 dargestellt, sind verschiedene I/O-Vorrichtungen 1414 an den ersten Bus 1416, gemeinsam mit einer Bus-Brücke 1418 gekoppelt, die den ersten Bus 1416 an einen zweiten Bus 1420 koppelt. In einer Ausführungsform enthält der zweite Bus 1420 einen Low Pin Count (LPC) Bus. Verschiedene Vorrichtungen sind in einer Ausführungsform an den zweiten Bus 1420 gekoppelt, einschließlich zum Beispiel einer Tastatur und/oder Maus 1422, Kommunikationsvorrichtungen 1427 und einer Speichereinheit 1428 wie ein Festplattenlaufwerks oder eine andere Massenspeichervorrichtung, die häufig Befehle/Code und Daten 1430 enthält. Ferner ist ein Audio-I/O 1424 an den zweiten Bus 1420 gekoppelt dargestellt. Es ist zu beachten, dass andere Architekturen möglich sind, wo die enthaltenen Komponenten und Zwischenverbindungsarchitekturen variieren. Zum Beispiel kann ein System anstelle der Punkt-zu-Punkt-Architektur von **Fig.** 14 einen Multi-Drop-Bus oder eine andere derartige Architektur implementieren.

[0108] Unter Bezugnahme nun auf **Fig.** 15 ist eine Ausführungsform eines System-on-Chip- (SOC) Designs gemäß der Erfindung dargestellt. Als ein spezielles veranschaulichendes Beispiel ist das SOC 1500 in Benutzergeräten (UE) enthalten. In einer Ausführungsform bezieht sich UE auf jede Vorrichtung, die von einem Endbenutzer zum Kommunizieren verwendet wird, wie ein in der Hand gehaltenes Telefon, ein Smartphone, ein Tablet, ein ultradünnes Notebook, ein Notebook mit Breitbandadapter oder jede andere ähnliche Kommunikationsvorrichtung. Häufig ist ein UE mit einer Basisstation oder einem Knoten verbunden, der möglicherweise in seiner Art einer Mobilstation (MS) in einem GSM-Netzwerk entspricht.

[0109] Hier enthält das SOC 1500 2 Kerne - 1506 und 1507. Ähnlich wie in der vorangehenden Besprechung können die Kerne 1506 und 1507 einer Befehlssatzarchitektur, wie einem auf einem Intel® Architecture Core™-basierenden Prozessor, einem Advanced Mikro Devices, Inc. (AMD) Prozessor, einem auf MIPS-basierenden Prozessor, einem auf ARMbasierenden Prozessordesign oder einem Kunden davon, wie auch deren Lizenznehmern oder Adoptierenden entsprechen. Die Kerne 1506 und

1507 sind an eine Cache-Steuerung 1508 gekoppelt, die mit der Bus Schnittstelleeinheit 1509 und einem L2 Cache 1511 verknüpft ist, um mit anderen Teilen des Systems 1500 zu kommunizieren. Die Zwischenverbindung 1510 enthält eine On-Chip-Zwischenverbindung, wie eine IOSF, AMBA oder andere Zwischenverbindung, wie oben besprochen, die möglicherweise einen oder mehrere der hier beschriebenen Aspekte implementiert.

[0110] Die Schnittstelle 1510 stellt Kommunikationskanäle zu den anderen Komponenten bereit, wie einem Subscriber Identity Module (SIM) 1530 für eine Schnittstelle mit einer SIM-Karte, einem Boot-ROM 1535, um einen Boot-Code für eine Ausführung durch die Kerne 1506 und 1507 zu halten, um das SOC 1500 zu initialisieren und zu booten, einer SDRAM Steuerung 1540 für eine Schnittstelle mit dem externen Speicher (z.B. DRAM 1560), einer Flash-Steuerung 1545 für eine Schnittstelle mit dem nicht flüchtigen Speicher (z.B. Flash 1565), einer peripheren Steuerung 1550 (z.B. seriellen peripheren Schnittstelle) für eine Schnittstelle mit peripheren Geräten, Video-Codecs 1520 und einer Video Schnittstelle 1525 zum Anzeigen und Empfangen eines Eingangs (z.B. durch Berührung ausgelösten Eingangs), einer GPU 1515 zur Durchführung grafikbezogener Berechnungen, usw. Jede dieser Schnittstellen kann Aspekte der hier beschriebenen Erfindung enthalten.

[0111] Zusätzlich veranschaulicht das System periphere Geräte zur Kommunikation, wie ein Bluetooth Modul 1570, 3G-Modem 1575, GPS 1585 und WiFi 1585. Es ist zu beachten, dass ein UE, wie oben angegeben, einen Funk zur Kommunikation enthält. Infolgedessen sind nicht alle diese peripheren Kommunikationsmodule erforderlich. In einem UE soll jedoch eine gewisse Form von Funkt zur externen Kommunikation enthalten sein.

[0112] Während die vorliegende Erfindung in Bezug auf eine begrenzte Anzahl von Ausführungsformen beschrieben wurde, sind zahlreiche Modifizierungen und Variationen daraus für einen Fachmann auf dem Gebiet offensichtlich. Es ist beabsichtigt, dass die beiliegenden Ansprüche alle derartigen Modifizierungen und Variationen abdecken, die in das wahre Wesen und den Umfang dieser vorliegenden Erfindung fallen.

[0113] Ein Design kann verschiedene Stufen durchlaufen, von der Erstellung bis zur Simulierung bis zur Herstellung. Daten, die ein Design präsentieren, können das Design auf zahlreiche Weisen darstellen. Zunächst kann die Hardware, wie in Simulationen sinnvoll ist, unter Verwendung einer Hardware-Beschreibungssprache oder einer anderen Funktionsbeschreibungssprache dargestellt werden. Zusätzlich kann ein Modell der Schaltungsebenen mit Logik und/oder Transistorgates in einigen Stufen des Designprozesses erzeugt werden. Ferner erreichen die meisten Designs, in einer gewissen Stufe, eine Datenebene, die die physische Anordnung verschiedener Vorrichtungen im Hardware-Modell darstellt. In dem Fall, wo herkömmliche Halbleiterherstellungstechniken verwendet werden, können die Daten, die das Hardware-Modell darstellen, die Daten sein, die das Vorhandensein oder Fehlen verschiedener Merkmale auf verschiedenen Maskenschichten für Masken präsentieren, die zur Herstellung der integrierten Schaltung verwendet werden. In jeder Darstellung des Designs können die Daten in jeder Form eines maschinenlesbaren Mediums gespeichert werden. Ein Speicher oder ein magnetischer oder optischer Speicher, wie eine Platte, kann ein maschinenlesbares Medium zum Speichern von Informationen sein, die über eine optische oder elektrische Welle gesendet werden, die moduliert oder auf andere Weise generiert ist, um solche Informationen zu senden. Wenn eine elektrische Trägerwelle, die den Code anzeigt oder trägt, gesendet wird, wird eine neue Kopie in dem Ausmaß, in dem ein Kopieren, Puffern oder erneutes Senden des elektrischen Signals durchgeführt wird, erstellt. Somit kann ein Kommunikationsanbieter oder ein Netzanbieter auf einem greifbaren, maschinenlesbaren Medium, zumindest vorübergehend, einen Artikel, wie Informationen, die in eine Trägerwelle codiert sind, speichern, was Techniken von Ausführungsformen der vorliegenden Erfindung verkörpert.

[0114] Ein Modul, wie hier verwendet, bezieht sich auf jede Kombination von Hardware, Software und/oder Firmware. Als ein Beispiel enthält ein Modul Hardware, wie eine Mikrosteuerung, die mit einem nicht flüchtigen Medium verknüpft ist, um einen Code zu speichern, der dazu ausgebildet ist, von der Mikrosteuerung ausgeführt zu werden. Daher bezieht sich ein Verweis auf ein Modul in einer Ausführungsform auf die Hardware, die spezifisch konfiguriert ist, um den Code zu erkennen und/oder auszuführen, der auf einem nicht flüchtigen Medium gehalten wird. Ferner bezieht sich in einer anderen Ausführungsform die Verwendung eines Moduls auf das nicht flüchtige Medium, das den Code enthält, der spezifisch dazu ausgebildet ist, von der Mikrosteuerung ausgeführt zu werden, um vorgegebene Operationen durchzuführen. Und, wie daraus abgeleitet werden kann, kann sich in einer weiteren Ausführungsform der Begriff Modul (in diesem Beispiel) auf die Kombination aus der Mikrosteuerung und dem nicht flüchtigen Medium beziehen. Häufig variieren allgemein Modulgrenzen, die als getrennt dargestellt sind, und überlappen möglicherweise. Zum Beispiel können ein erstes und ein zweites Modul gemeinsam Hardware, Software, Firmware oder eine Kombination davon benutzen, während sie möglicherweise eine gewisse unabhängige Hardware, Software oder Firmware behalten. In einer

Ausführungsform, enthält die Verwendung des Begriffs „Logik“ Hardware, wie Transistoren, Register oder andere Hardware, wie programmierbare Logikvorrichtungen.

[0115] Die Verwendung der Phrase `konfiguriert zum' bezieht sich in einer Ausführungsform auf die Anordnung, Zusammenstellung, Herstellung, Anbietung zum Verkauf, den Import und/oder den Entwurf eines Apparats, einer Hardware, einer Logik oder eines Elements, um eine angegebenen oder bestimmte Aufgabe auszuführen. In diesem Beispiel ist ein Apparat oder ein Element davon, der bzw. das nicht in Betrieb ist, weiterhin ‚konfiguriert‘, eine bestimmte Aufgabe auszuführen, wenn es zur Durchführung der bestimmten Aufgabe gestaltet, gekoppelt und/oder verbunden ist. Als ein rein veranschaulichendes Beispiel kann ein Logik-Gate eine 0 oder eine 1 während des Betriebs bereitstellen. Aber ein Logik-Gate, das zum Bereitstellen eines Freigabesignals für einen Takt ‚konfiguriert‘ ist, enthält nicht jedes mögliche Logik-Gate, das eine 1 oder 0 bereitstellen kann. Stattdessen ist das Logik-Gate eines, das auf gewisse Weise gekoppelt ist, so dass während des Betriebs die Ausgabe der 1 oder 0 zur Freigabe des Takts dient. Es ist erneut zu beachten, dass die Verwendung des Begriffs ‚konfiguriert zum‘ keinen Betrieb erfordert, sondern der Schwerpunkt vielmehr auf dem latenten Zustand eines Apparats, einer Hardware und/oder eines Elements liegt, wobei der Apparat, die Hardware und/oder das Element im latenten Zustand gestaltet ist, eine besondere Aufgabe auszuführen, wenn der Apparat, die Hardware, und/oder das Element arbeitet.

[0116] Ferner bezieht sich die Verwendung der Phrasen 'zum', 'imstande zu' und/oder 'betriebsbereit zu' in einer Ausführungsform auf einen gewissen Apparat, eine Logik, eine Hardware, und/oder ein Element, der/die/das so gestaltet ist, dass die Verwendung des Apparats, der Logik, der Hardware und/oder des Elements in einer spezifizierten Weise möglich ist. Es wird wie oben festgehalten, dass die Verwendung von 'imstande zu' oder 'betriebsbereit zu' sich in einer Ausführungsform auf den latenten Zustand eines Apparats, einer Logik, einer Hardware und/oder eines Elements bezieht, in dem der Apparat, die Logik, die Hardware und/oder das Element nicht in Betrieb sind, sondern derart gestaltet sind, dass sie die Verwendung eines Apparats in einer spezifizierten Weise ermöglichen.

[0117] Ein Wert, wie hier verwendet, enthält jede bekannte Darstellung einer Zahl, eines Zustands, eines logischen Zustands oder eines binären logischen Zustands. Häufig wird die Verwendung von Logikpegeln, Logikwerten oder logischen Werten auch als 1 und 0 bezeichnet, die einfach binäre Logikzustände darstellen. Zum Beispiel bezieht sich eine 1 auf einen hohen Logikpegel und 0 bezieht sich auf einen niederen Logikpegel. In einer Ausführungsform kann eine Speicherzelle, wie ein Transistor oder eine Flash-Zelle, imstande sein, einen einzelnen logischen Wert oder mehrfache logische Werte zu halten. Es wurden jedoch andere Darstellungen von Werten in Computersystemen verwendet. Zum Beispiel kann die Dezimalzahl zehn auch als ein binärer Wert 1010 und ein Hexadezimalbuchstabe A dargestellt werden. Daher enthält ein Wert jede Darstellung von Informationen, die imstande ist, in einem Computersystem gehalten zu werden.

[0118] Ferner können Zustände durch Werte oder Teile von Werten dargestellt werden. Als ein Beispiel kann ein erster Wert, wie eine logische Eins, einen vorgegebenen oder anfänglichen Zustand darstellen, während ein zweiter Wert, wie eine logische Null, einen nicht vorgegebenen Zustand darstellen kann. Zusätzlich beziehen sich die Begriffe Zurücksetzen und Setzen in einer Ausführungsform auf einen vorgegebenen bzw. aktualisierten Wert oder Zustand. Zum Beispiel enthält ein vorgegebener Wert möglicherweise einen hohen logischen Wert, d.h., Rücksetzen, während ein aktualisierter Wert möglicherweise einen niederen logischen Wert enthält, d.h., Setzen. Es ist zu beachten, dass jede Kombination von Werten zur Darstellung jeder Anzahl von Zuständen verwendet werden kann.

[0119] Die Ausführungsformen von Verfahren, Hardware, Software, Firmware oder Code, die oben angeführt sind, können durch Befehle oder einen Code implementiert sein, die bzw. der auf einem maschinenzugänglichen, maschinenlesbaren, computerzugänglichen oder computerlesbaren Medium gespeichert sind bzw. ist, die von einem Verarbeitungselement ausführbar sind. Ein nicht flüchtiges maschinenzugängliches/-lesbares Medium enthält jeden Mechanismus, der Informationen in einer Form vorsieht (d.h., speichert und/oder sendet), die von einer Maschine, wie einem Computer oder einem elektronischen System lesbar ist. Zum Beispiel enthält ein nicht flüchtiges maschinenzugängliches Medium einen Direktzugriffsspeicher (RAM), wie einen statischen RAM (SRAM) oder dynamischen RAM (DRAM); einen ROM; ein magnetisches oder optisches Speichermedium; Flash-Speichervorrichtungen; elektrische Speichervorrichtungen; optische Speichervorrichtungen; akustische Speichervorrichtungen; eine andere Form von Speichervorrichtungen zum Halten von Informationen, die von flüchtigen (verbreiteten) Signalen empfangen werden (z.B. Trägerwellen, Infrarotsignale, Digitalsignale); usw., die von den nicht flüchtigen Medien unterschieden werden, die Informationen daraus empfangen können.

[0120] Befehle, die zum Programmieren einer Logik verwendet werden, um Ausführungsformen der Erfindung auszuführen können in einem Speicher

im System gespeichert sein, wie in einem DRAM, Cache, Flash-Speicher oder einem anderen Speicher. Ferner können die Befehle über ein Netzwerk oder durch andere computerlesbare Medien verbreitet werden. Somit kann ein maschinenlesbares Medium jeden Mechanismus zum Speichern oder Senden von Informationen in einer Form, die von einer Maschine (z.B. einem Computer) lesbar ist, enthalten, wie, ohne aber darauf beschränkt zu sein, Disketten, optische Platten, Compact Disc Nur-Lese-Speicher (CD-ROMs) und magneto-optische Platten, Nur-Lese-Speicher (ROMs), Direktzugriffsspeicher (RAM), einen löschbaren, programmierbaren Nur-Lese-Speicher (EPROM), einen elektrisch löschbaren, programmierbaren Nur-Lese-Speicher (EEPROM), Magnet- oder optische Karten, einen Flash-Speicher oder einen greifbaren, maschinenlesbaren Speicher, der zum Senden von Informationen über das Internet in einer elektrischen, optischen, akustischen oder anderen Form von verbreiteten Signalen verwendet wird (z.B. Trägerwellen, Infrarotsignale, digitale Signale, usw.). Daher enthält das computerlesbare Medium jede Art von greifbarem, maschinenlesbaren Medium, das zum Speichern oder Senden elektronischer Befehle oder Informationen in einer Form geeignet ist, die von einer Maschine (z.B. einem Computer) gelesen werden kann.

[0121] Die folgenden Beispiele betreffen Ausführungsformen gemäß dieser Patentschrift. Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren vorsehen, um eine Steuerung eines gemeinsam benutzten Speichers bereitzustellen, um Last- und Speicheroperationen von mehreren unabhängigen Knoten zu bedienen, um einen Zugang zu einer gemeinsam benutzten Speicherressource bereitzustellen, wobei jedem der mehreren unabhängigen Knoten ein Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource zu gewähren ist.

[0122] In zumindest einem Beispiel werden die Last- und Speicheroperationen unter Verwendung eines Verbindungsprotokolls des gemeinsam benutzten Speichers kommuniziert.

[0123] In zumindest einem Beispiel enthält das Verbindungsprotokoll des gemeinsam benutzten Speichers ein Speicherzugangsprotokoll, das eine physische Schichtlogik eines anderen Zwischenverbindungsprotokolls verwendet.

[0124] In zumindest einem Beispiel stellt das Verbindungsprotokoll des gemeinsam benutzten Speichers ein Multiplexen zwischen einer Sendung von Daten der Speicherzugangsprotokolldaten und einer Sendung von Daten des Zwischenverbindungsprotokolls bereit.

[0125] In zumindest einem Beispiel weisen die Daten des Zwischenverbindungsprotokolls zumindest eines von Verbindungsschichtdaten und Transaktionsschichtdaten auf.

[0126] In zumindest einem Beispiel weist das Speicherzugangsprotokoll SMI3 auf und das Zwischenverbindungsprotokoll weist Peripheral Component Interconnect (PCI) Express (PCIe) auf.

[0127] In zumindest einem Beispiel werden Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch eine Sync-Kopfzeile identifiziert, die zum Identifizieren der Übergänge codiert ist.

[0128] In zumindest einem Beispiel werden Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch einen Datenstart-Framing-Token identifiziert, der zum Identifizieren der Übergänge codiert ist.

[0129] In zumindest einem Beispiel werden Übergänge von Zwischenverbindungsprotokolldaten zu Speicherzugangsprotokolldaten durch einen Datenstromende-Framing-Token des Zwischenverbindungsprotokolls identifiziert, der zum Identifizieren der Übergänge codiert ist, und Übergänge von Speicherzugangsprotokolldaten zu Zwischenverbindungsprotokolldaten werden durch Verbindungsschichtsteuerungs-Flits des Speicherzugangsprotokolls identifiziert.

[0130] In zumindest einem Beispiel ist das Verbindungsprotokoll des gemeinsam benutzten Speichers über einen Netzwerkprotokollstapel getunnelt.

[0131] In zumindest einem Beispiel weist der Netzwerkprotokollstapel Ethernet auf.

[0132] In zumindest einem Beispiel befindet sich ein erster der mehreren CPU-Knoten auf einer ersten Platine und ein zweiter der mehreren CPU-Knoten befindet sich auf einer zweiten Platine, die von der ersten Platine getrennt ist.

[0133] In zumindest einem Beispiel befinden sich zumindest zwei der mehreren CPU-Knoten auf derselben Vorrichtung.

[0134] In zumindest einem Beispiel dient die Steuerung des gemeinsam benutzten Speichers ferner zum Verfolgen von Speichertransaktionen, welche die Last- und Speicheroperationen beinhalten.

[0135] In zumindest einem Beispiel dient die Steuerung des gemeinsam benutzten Speichers ferner

zum Feststellen, dass ein bestimmter der mehreren CPU-Knoten versagt, Identifizieren eines Teils der Speichertransaktionen des bestimmten CPU-Knotens, und Fallenlassen des Teils der Speichertransaktionen des bestimmten CPU-Knotens, während alle anderen Speichertransaktionen aufrechterhalten bleiben.

[0136] In zumindest einem Beispiel dient die Steuerung des gemeinsam benutzten Speichers ferner zum Verwalten von Genehmigungen für einen Zugang der mehreren CPU-Knoten zu Daten in der gemeinsam benutzten Speicherressource.

[0137] In zumindest einem Beispiel ist zumindest ein bestimmter der mehreren CPU-Knoten von einem Zugang zu zumindest einem ersten Teil des gemeinsam benutzten Speichers blockiert und einem zweiten der mehreren CPU-Knoten ist ein Zugang zum ersten Teil möglich.

[0138] In zumindest einem Beispiel dient die Steuerung des gemeinsam benutzten Speichers ferner zum Verwalten von Verzeichnisinformationen für Daten in der gemeinsam benutzten Speicherressource.

[0139] In zumindest einem Beispiel stellt die Verzeichnisinformationen für jede von mehreren Datenressourcen, die in der gemeinsam benutzten Speicherressource gespeichert sind, fest, ob ein Zugang zur entsprechenden Datenressource ausschließlich für einen der mehreren CPU-Knoten ist oder von zwei oder mehr der mehreren CPU-Knoten gemeinsam benutzt wird.

[0140] In zumindest einem Beispiel dient die Steuerung des gemeinsam benutzten Speichers ferner zum Verhandeln einer Zugangsänderung für eine bestimmte der mehreren Datenressourcen, wobei die Änderung zumindest eine von einer Zugangsänderung von gemeinsam benutzt zu ausschließlich oder einer Zugangsänderung von ausschließlich zu gemeinsam benutzt aufweist.

[0141] In zumindest einem Beispiel ist die Steuerung des gemeinsam benutzten Speichers an zumindest eine andere Steuerung des gemeinsam benutzten Speichers gekoppelt, die zumindest eine andere gemeinsam benutzte Speicherressource verwaltet, und die teuerung des gemeinsam benutzten Speichers dient ferner zum Kommunizieren von Last-/SpeicherOperationen zur anderen Steuerung des gemeinsam benutzten Speichers, um den mehreren CPU-Knoten einen Zugang zu dem anderen gemeinsam benutzten Speicher zu ermöglichen.

[0142] In zumindest einem Beispiel dient die Steuerung des gemeinsam benutzten Speichers ferner zum Abbilden von Adressinformationen in den Last- und Speicheroperationen auf entsprechende Datenressourcen, die in der gemeinsam benutzten Speicherressource gespeichert sind.

[0143] Eine oder mehrere Ausführungsformen können einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, eine auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Senden einer Speicherzugangsanfrage zu einer Steuerung eines gemeinsam benutzten Speichers vorsehen, wobei die Speicherzugangsanfrage eine Last-/Speicheroperation aufweist und zum Identifizieren einer Adresse einer Datenressource dient, die in einer gemeinsam benutzten Speicherressource enthalten sein soll, die der Steuerung des gemeinsam benutzten Speichers entspricht, und jedem der mehreren unabhängigen Knoten wird ein Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt.

[0144] In zumindest einem Beispiel weist die Speicherzugangsanfrage eine Lastanfrage auf und die I/O-Logik dient ferner zum Empfangen von Daten entsprechend der Datenressource als Antwort auf die Lastanfrage.

[0145] In zumindest einem Beispiel weist die Speicherzugangsanfrage eine Speicheranfrage auf.

[0146] In zumindest einem Beispiel wird Speicherzugangsanfrage unter Verwendung eines Verbindungsprotokolls des gemeinsam benutzten Speichers gesendet und das des gemeinsam benutzten Speichers Verbindungsprotokoll des gemeinsam benutzten Speichers enthält ein Speicherzugangsprotokoll, das eine physische Schichtlogik eines anderen Zwischenverbindungsprotokolls benutzt.

[0147] In zumindest einem Beispiel sieht das Verbindungsprotokoll des gemeinsam benutzten Speichers ein Multiplexen zwischen einer Sendung von Daten der Speicherzugangsprotokolldaten und einer Sendung von Daten des Zwischenverbindungsprotokolls vor.

[0148] In zumindest einem Beispiel werden Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch zumindest eines der folgenden identifiziert: (a) eine Sync-Kopfzeile, die zum Identifizieren der Übergänge codiert ist; (b) einen Datenstart-Framing-Token, der zum Identifizieren der Übergänge codiert ist; und (c) einen Datenstromende-Framing-Token, der zum Identifizieren der Übergänge codiert ist.

[0149] In zumindest einem Beispiel weist das Speicherzugangsprotokoll ein SMI3 auf und das Zwischenverbindungsprotokoll weist ein auf PCIe basierendes Protokoll auf.

[0150] In zumindest einem Beispiel weist ein bestimmter der mehreren Knoten mehrfache CPU-Buchsen und einen lokalen Speicher auf. In zumindest einem Beispiel befindet sich die gemeinsam benutzte Speicherressource auf einer Vorrichtung getrennt vom bestimmten Knoten.

[0151] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, Auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Empfangen einer ersten Last-/Speicher- Nachricht von einem ersten unabhängigen CPU-Knoten, der bestimmte Daten in einem gemeinsam benutzten Speicher identifiziert, Vorsehen eines Zugangs zu den bestimmten Daten für den ersten CPU-Knoten als Antwort auf die erste Last-/Speicher-Nachricht, Empfangen einer zweiten Last-/Speicher-Nachricht von einem zweiten unabhängigen CPU-Knoten, der bestimmte Daten in einem gemeinsam benutzten Speicher identifiziert, und Vorsehen eines Zugangs zu den bestimmten Daten für den zweiten CPU-Speicher als Antwort auf die zweite Last-/Speicher-Nachricht vorsehen.

[0152] In zumindest einem Beispiel werden jede der ersten und zweiten ersten Last-/Speicher-Nachrichten über eine Datenverbindung unter Verwendung eines Verbindungsprotokolls des gemeinsam benutzten Speichers empfangen.

[0153] Zumindest einige Ausführungsformen können ein Identifizieren, dass der erste CPU-Knoten Zugang zu den bestimmten Daten hat, und Identifizieren, dass der zweite CPU-Knoten Zugang zu den bestimmten Daten hat, vorsehen.

[0154] Zumindest einige Ausführungsformen können ein Verfolgen von Transaktionen, die den gemeinsam benutzten Speicher beinhalten, für den ersten wie auch zweiten CPU-Knoten vorsehen.

[0155] Zumindest einige Ausführungsformen können ein Identifizieren von Verzeichnisinformationen der bestimmten Daten vorsehen, wobei die Verzeichnisinformationen identifizieren, ob die es bestimmten Daten in einem gemeinsam benutzten, ungecachten oder ausschließlichen Zustand sind.

[0156] In zumindest einem Beispiel identifiziert die erste Last-/Speicher-Nachricht die bestimmten Daten durch eine erste Adresse und die zweite erste Last-/Speicher-Nachricht identifiziert die bestimmten Daten durch eine zweite, andere Adresse.

[0157] Zumindest einige Ausführungsformen können ein Abbilden der ersten Adresse auf die bestimmten Daten und Abbilden der zweiten Adresse auf die bestimmten Daten vorsehen.

[0158] Zumindest einige Ausführungsformen können ein System vorsehen, das einen ersten Knoten aufweist, der eine oder mehrere Prozessorvorrichtung(en) aufweist, einen zweiten Knoten, unabhängig vom ersten Knoten und der eine oder mehrere Prozessorvorrichtung(en) aufweist, und einen gemeinsam benutzten Speicher, der für jeden von dem ersten und zweiten Knoten durch ein Last-/Speicher-Speicherzugangsprotokoll zugänglich ist.

[0159] In zumindest einem Beispiel hat der erste Knoten eine Fehlerdomäne unabhängig vom zweiten Knoten.

[0160] In zumindest einem Beispiel wird der erste Knoten von einem ersten Betriebssystem gesteuert und der zweite Knoten wird von einem zweiten Betriebssystem gesteuert.

[0161] In zumindest einem Beispiel ist das Last-/Speicher-Speicherzugangsprotokoll in einem Verbindungsprotokoll des gemeinsam benutzten Speichers enthalten und das Verbindungsprotokoll des gemeinsam benutzten Speichers wechselt zwischen dem Speicherzugangsprotokoll und einem anderen Zwischenverbindungsprotokoll.

[0162] In zumindest einem Beispiel kann eine Steuerung des gemeinsam benutzten Speichers Last- und Speicheroperationen vom ersten und zweiten Knoten bedienen und einen Zugang zum gemeinsam benutzten Speicher vorsehen.

[0163] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Senden einer ersten Sync-Kopfzeile auf Spuren einer Datenverbindung vorsehen, wobei die erste Sync-Kopfzeile zum Identifizieren eines Übergangs von Daten eines Zwischenverbindungsprotokolls zu Daten eines Speicherzugangsprotokolls und Senden einer zweite Sync-Kopfzeile auf den Spuren der Datenverbindung codiert ist, wobei die zweite Sync-Kopfzeile zum Identifizieren eines Übergangs von Daten des Speicherzugangsprotokolls zu Daten des Zwischenverbindungsprotokolls codiert ist.

[0164] In zumindest einem Beispiel identifiziert jede Sync-Kopfzeile eine Art eines Datenblocks, welcher der Sync-Kopfzeile folgt.

[0165] In zumindest einem Beispiel ist jeder Datenblock eine vordefinierte Länge.

[0166] In zumindest einem Beispiel weist das Speicherzugangsprotokoll ein Protokoll auf, das auf SMI3 basiert.

[0167] In zumindest einem Beispiel weist das Zwischenverbindungsprotokoll ein Protokoll auf, das auf einem auf PCIe basierenden Protokoll basiert.

[0168] In zumindest einem Beispiel ist jede Sync-Kopfzeile nach einer 128b/130b Codierung codiert.

[0169] In zumindest einem Beispiel zeigt die zweite Sync-Kopfzeile einen Datenblock des Zwischenverbindungsprotokolls an und eine dritte Sync-Kopfzeile wird auf den Spuren der Datenverbindung gesendet, um einen gereihten Blocksatz des Zwischenverbindungsprotokolls anzuzeigen.

[0170] In zumindest einem Beispiel ist die erste Sync-Kopfzeile mit abwechselnden Werten auf den Spuren codiert und die zweite Sync-Kopfzeile ist mit demselben Wert auf allen der Spuren codiert.

[0171] In zumindest einem Beispiel weisen die Daten des Speicherzugangsprotokolls Verbindungsschichtdaten auf und die Daten des Zwischenverbindungsprotokolls weisen eines von Transaktionsschicht- und Datenverbindungsschichtpaketen auf.

[0172] In zumindest einem Beispiel sind die Sync-Kopfzeilen gemäß dem Zwischenverbindungsprotokoll definiert.

[0173] In zumindest einem Beispiel unterstützt das Speicherzugangsprotokoll eine Last-/Speicher-Speicherzugangsnachrichtenübermittlung.

[0174] In zumindest einem Beispiel weisen die Speicherzugangsprotokolldaten eine Speicherzugangsnachrichtenübermittlung für einen Zugang zu einer gemeinsam benutzten Speicherressource auf, wobei jedem der mehreren unabhängigen Knoten Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt wird.

[0175] In zumindest einem Beispiel hat jeder der mehreren unabhängigen Knoten eine unabhängige Fehlerdomäne.

[0176] In zumindest einem Beispiel weist die Datenverbindung zumindest vier Spuren auf.

[0177] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Empfangen einer ersten Sync-Kopfzeile auf Spuren einer Datenverbindung, wobei die erste Sync-Kopfzeile mit einer ersten Codierung codiert ist, Identifizieren, aus der ersten Codierung der ersten Sync-Kopfzeile, eines Übergangs von Daten eines Zwischenverbindungsprotokolls zu Daten eines Speicherzugangsprotokolls, Empfangen einer zweiten Sync-Kopfzeile auf den Spuren der Datenverbindung, wobei die zweite Sync-Kopfzeile mit einer zweiten Codierung codiert ist, und Identifizieren, aus der zweiten Codierung der zweiten Sync-Kopfzeile, eines Übergangs von Daten des Speicherzugangsprotokolls zu Daten des Zwischenverbindungsprotokolls vorsehen.

[0178] In zumindest einem Beispiel identifiziert jede Sync-Kopfzeile eine Art eines Datenblocks, welcher der Sync-Kopfzeile folgt.

[0179] In zumindest einem Beispiel weist die Zwischenverbindungsprotokoll ein auf PCIe basierendes Protokoll auf.

[0180] In zumindest einem Beispiel basiert das Speicherzugangsprotokoll auf SMI3.

[0181] In zumindest einem Beispiel ist die Sync-Kopfzeile gemäß einer 128b/130b Codierung codiert.

[0182] In zumindest einem Beispiel weist die erste Codierung Werte von 01b und 10b auf, die auf den Spuren der Datenverbindung abwechseln.

[0183] In zumindest einem Beispiel weisen die Daten des Speicherzugangsprotokolls Last-/Speicher-Speicherzugangsnachrichten auf.

[0184] In zumindest einem Beispiel weisen die Speicherzugangsnachrichten Nachrichten für einen Zugang zu einer gemeinsam benutzten Speicherressource auf und jedem der mehreren unabhängigen Knoten in einem System wird Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt.

[0185] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum empfangen einer ersten Sync-Kopfzeile auf Spuren einer Datenverbindung, wobei die erste Sync-Kopfzeile mit einer ersten Codierung codiert ist, Identifizieren aus der ersten Codierung der ersten Sync-Kopfzeile eines Übergangs von Daten eines Zwischenverbindungsprotokolls zu Daten eines Speicherzugangsprotokolls, Verarbeiten der Daten des Speicherzugangsprotokolls, Empfangen einer zweiten Sync-Kopfzeile auf den Spuren der Datenverbindung, wobei die zweite Sync-Kopfzeile mit einer zweiten Codierung codiert ist, und Identifizieren, aus der zweiten Codierung der zweiten Sync-Kopfzeile, eines Übergangs von Daten des Speicherzugang-

sprotokolls zu Daten des Zwischenverbindungsprotokolls vorsehen.

[0186] In zumindest einem Beispiel weist das Zwischenverbindungsprotokoll ein auf PCIe basierendes Protokoll auf und das Speicherzugangsprotokoll basiert auf SMI3.

[0187] In zumindest einem Beispiel sind die Sync-Kopfzeilen gemäß PCIe.

[0188] In zumindest einem Beispiel werden die Daten des Speicherzugangsprotokolls verarbeitet, um eine Speicherzugangsanfrage zu bedienen, die in den Daten des Speicherzugangsprotokolls enthalten ist.

[0189] In zumindest einem Beispiel ist die Speicherzugangsanfrage eine Anfrage einer gemeinsam benutzten Speicherressource, die von mehreren unabhängigen CPU-Knoten gemeinsam benutzt wird.

[0190] In zumindest einem Beispiel weist die Speicherzugangsanfrage eine Last-/Speicher-Nachricht auf.

[0191] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Senden eines ersten Datenstart-Framing-Tokens auf Spuren einer Datenverbindung, wobei der erste Datenstart-Framing-Token zum Identifizieren eines Übergangs von Daten eines Zwischenverbindungsprotokolls zu Daten eines Speicherzugangsprotokolls codiert ist, und Senden eines zweiten Datenstart-Framing-Tokens auf den Spuren der Datenverbindung, wobei der zweite Datenstart-Framing-Token zum Identifizieren eines Übergangs von Daten des Speicherzugangsprotokolls zu Daten des Zwischenverbindungsprotokolls codiert ist, vorsehen.

[0192] In zumindest einem Beispiel weist der erste Datenstart-Framing-Token einen modifizierten PCIe STP Framing-Token auf und der zweite Datenstart-Framing-Token weist einen PCIe STP Framing-Token auf.

[0193] In zumindest einem Beispiel enthält jeder Datenstart-Framing-Token ein Längenfeld.

[0194] In zumindest einem Beispiel wird der Übergang von Daten des Zwischenverbindungsprotokolls zu Daten des Speicherzugangsprotokolls im ersten Datenstart-Framing-Token durch einen Wert im Längenfeld des ersten Datenstart-Framing-Tokens angezeigt.

[0195] In zumindest einem Beispiel sind die Daten des Speicherzugangsprotokolls in einem Fenster zu senden, das durch das Längenfeld des ersten Datenstart-Framing-Tokens definiert ist.

[0196] In zumindest einem Beispiel basiert das Speicherzugangsprotokoll auf SMI3.

[0197] In zumindest einem Beispiel weist das Zwischenverbindungsprotokoll ein auf PCIe basierendes Protokoll auf.

[0198] In zumindest einem Beispiel weisen die Daten des Speicherzugangsprotokolls Verbindungsschichtdaten auf und die Daten des Zwischenverbindungsprotokolls weisen eines von Transaktionsschicht- und Datenverbindungsschichtpaketen auf.

[0199] In zumindest einem Beispiel dient die physische Schichtlogik ferner zum Senden der Daten des Speicherzugangsprotokolls und die Daten des Speicherzugangsprotokolls weisen Last-/Speicher-Speicherzugangsnachrichten auf.

[0200] In zumindest einem Beispiel weisen die Speicherzugangsprotokolldaten Speicherzugangsnachrichten für einen Zugang zu einer gemeinsam benutzten Speicherressource auf, und jedem der mehreren unabhängigen Knoten wird Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt.

[0201] In zumindest einem Beispiel hat jeder der mehreren unabhängigen Knoten eine unabhängige Fehlerdomäne.

[0202] In zumindest einem Beispiel weist die Datenverbindung eine oder mehrere Spuren auf.

[0203] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Empfangen eines ersten Datenstart-Framing-Tokens auf Spuren einer Datenverbindung, Identifizieren, aus dem ersten Datenstart-Framing-Token, eines Eintreffens von Daten eines Speicherzugangsprotokolls, Empfangen eines zweiten Datenstart-Framing-Tokens auf Spuren der Datenverbindung, wobei sich das zweite Datenstart-Framing-Token vom ersten Datenstart-Framing-Token unterscheidet, und Identifizieren, aus dem zweiten Datenstart-Framing-Token, eines Eintreffens von Daten eines Zwischenverbindungsprotokolls vorsehen.

[0204] In zumindest einem Beispiel weist der erste Datenstart-Framing-Token ein modifiziertes PCIe STP Framing-Token auf und das zweite Datenstart-

Framing-Token weist einn PCIe STP Framing-Token auf.

[0205] In zumindest einem Beispiel enthält jeder Datenstart-Framing-Token ein Längenfeld.

[0206] In zumindest einem Beispiel wird der Übergang von Daten des Zwischenverbindungsprotokolls zu Daten des Speicherzugangsprotokolls im ersten Datenstart-Framing-Token durch einen Wert im Längenfeld des ersten Datenstart-Framing-Tokens angezeigt.

[0207] In zumindest einem Beispiel basiert das Speicherzugangsprotokoll auf SMI3 und das Zwischenverbindungsprotokoll weist ein auf PCIe basierendes Protokoll auf.

[0208] In zumindest einem Beispiel werden die Daten des Speicherzugangsprotokolls empfangen und

[0209] werden die Daten des Zwischenverbindungsprotokolls empfangen.

[0210] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Senden eines ersten Datenstromende-Framing-Tokens auf Spuren einer Datenverbindung, wobei das erste Datenstromende-Framing-Token zum Identifizieren eines Übergangs von einem Zwischenverbindungsprotokoll zu einem Speicherzugangsprotokoll codiert ist, Senden von Speicherzugangsprotokolldaten nach dem Übergang zum Speicherzugangsprotokoll, und Senden von Verbindungsschichtsteuerungsdaten des Speicherzugangsprotokolls, um einen Übergang vom Speicherzugangsprotokoll zum Zwischenverbindungsprotokoll zu identifizieren, vorsehen.

[0211] In zumindest einem Beispiel sind die Speicherzugangsprotokolldaten auf der Datenverbindung zu senden, bis die Verbindungsschichtsteuerungsdaten gesendet werden.

[0212] In zumindest einem Beispiel veranlasst der Übergang zum Speicherzugangsprotokoll einen Übergang von der Zwischenverbindungsprotokolllogik, die Daten auf der Datenverbindung behandelt, zur Speicherzugangsprotokolllogik, die Daten auf der Datenverbindung behandelt.

[0213] In zumindest einem Beispiel weist das Speicherzugangsprotokoll ein Protokoll auf, das auf SMI3 basiert.

[0214] In zumindest einem Beispiel weist das Zwischenverbindungsprotokoll ein auf PCIe basierendes Protokoll auf.

[0215] In zumindest einem Beispiel weist das erste Datenstromende-Framing-Token ein modifiziertes PCIe EDS Framing-Token auf.

[0216] In zumindest einem Beispiel wird ein PCIe EDS zum Anzeigen eines Endes eines Satzes von PCIe-Transaktionsschichtpaketen und eines Eintreffens eines PCIe gereihten Blocksatzes gesendet.

[0217] In zumindest einem Beispiel weisen die Daten des Speicherzugangsprotokolls Verbindungsschichtdaten auf und die Daten des Zwischenverbindungsprotokolls weisen eines von Transaktionsschicht- und Datenverbindungsschichtpaketen auf.

[0218] In zumindest einem Beispiel werden die Daten des Speicherzugangsprotokolls gesendet und weisen Last-/Speicher-Speicherzugangsnachrichten auf.

[0219] In zumindest einem Beispiel weist das Speicherzugangsprotokolldaten Speicherzugangsnachrichten für einen Zugang zu einer gemeinsam benutzten Speicherressource auf, und jedem der mehreren unabhängigen Knoten wird Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt.

[0220] In zumindest einem Beispiel hat jeder der mehreren unabhängigen Knoten eine unabhängige Fehlerdomäne.

[0221] Eine oder mehrere Ausführungsform(en) kann (können) einen Apparat, ein System, einen maschinenlesbaren Speicher, ein maschinenlesbares Medium, auf Hardware- und/oder Software basierende Logik und ein Verfahren zum Empfangen eines ersten Datenstromende-Framing-Tokens auf Spuren einer Datenverbindung, die zum Identifizieren eines Übergangs von einem Zwischenverbindungsprotokoll zu einem Speicherzugangsprotokoll codiert ist, Übergehen zu einer Verwendung einer Verbindungsschichtlogik des Speicherzugangsprotokolls auf der Basis des ersten Datenstromende-Framing-Tokens, Empfangen von Speicherzugangsprotokollverbindungsschichtdaten, Empfangen von Verbindungsschichtsteuerungsdaten des Speicherzugangsprotokolls zum Identifizieren eines Übergang vom Speicherzugangsprotokoll zum Zwischenverbindungsprotokoll, und Übergehen zu einer Verwendung einer Verbindungsschichtlogik des Zwischenverbindungsprotokolls auf der Basis der Verbindungsschichtsteuerungsdaten, vorsehen.

[0222] In zumindest einem Beispiel basiert das Speicherzugangsprotokoll auf SMI3.

[0223] In zumindest einem Beispiel weist das Zwischenverbindungsprotokoll ein auf PCIe basierendes Protokoll auf.

[0224] In zumindest einem Beispiel weist das erste Datenstromende-Framing-Token ein modifiziertes PCIe EDS Framing-Token auf.

[0225] In zumindest einem Beispiel weisen die Daten des Speicherzugangsprotokolls Verbindungsschichtdaten auf und die Daten des Zwischenverbindungsprotokolls weisen eines von Transaktionsschicht- und Datenverbindungsschichtpaketen auf.

[0226] In zumindest einem Beispiel weisen die Daten des Speicherzugangsprotokolls Last-/Speicher-Speicherzugangsnachrichten auf.

Patentansprüche

1. Apparat aufweisend:
eine Steuerung (515; 515a, 515b) eines gemeinsam benutzten Speichers (505; 505a, 505b) zum:
Bedienen von Last- und Speicheroperationen, die über Datenverbindungen von mehreren unabhängigen Knoten (510a,....., 510n; 510a,....., 510e) empfangen werden, um einen Zugang zu einer gemeinsam benutzten Speicherressource vorzusehen, wobei jedem der mehreren unabhängigen Knoten (510a,.....,510n; 510a,.....510e) Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt wird; und
eine I/O-Logik zum:
Identifizieren von Übergängen zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten, die auf den Datenverbindungen gesendet werden,
dadurch gekennzeichnet dass Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch ein Datenstrom-Framing-Token, das zum Identifizieren der Übergänge codiert ist, identifziert werden.

2. Apparat nach Anspruch 1, wobei die Last- und Speicheroperationen unter Verwendung eines Verbindungsprotokolls für einen gemeinsam benutzten Speicher (505; 505a, 505b) kommuniziert werden.

3. Apparat nach Anspruch 2, wobei das Verbindungsprotokoll des gemeinsam benutzten Speichers (505; 505a, 505b) über einen Netzwerkprotokollstapel getunnelt wird.

4. Apparat nach Anspruch 3, wobei der Netzwerkprotokollstapel Ethernet aufweist.

5. Apparat nach Anspruch 2, wobei das Verbindungsprotokoll des gemeinsam benutzten Speichers (505; 505a, 505b) ein Speicherzugangsprotokoll enthält, das eine physische Schichtlogik eines anderen Zwischenverbindungsprotokolls verwendet.

6. Apparat nach Anspruch 5, wobei das Verbindungsprotokoll des gemeinsam benutzten Speichers (505; 505a, 505b) ein Multiplexen zwischen einer Sendung von Daten der Speicherzugangsprotokolldaten und einer Sendung von Daten des Zwischenverbindungsprotokolls vorsieht.

7. Apparat nach Anspruch 1, wobei die Daten des Zwischenverbindungsprotokolls zumindest eines von Verbindungsschichtdaten und Transaktionsschichtdaten aufweisen.

8. Apparat nach Anspruch 1, wobei das Speicherzugangsprotokoll SMI3 aufweist und das Zwischenverbindungsprotokoll Peripheral Component Interconnect (PCI) Express (PCIe) aufweist.

9. Apparat nach Anspruch 1, wobei Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch eine Sync-Kopfzeile identifiziert sind, die zum Identifizieren der Übergänge codiert ist.

10. Apparat nach Anspruch 1, wobei Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch ein Datenstart-Framing-Token identifiziert sind, das zum Identifizieren der Übergänge codiert ist.

11. Apparat nach Anspruch 1, wobei Übergänge von Zwischenverbindungsprotokolldaten zu Speicherzugangsprotokolldaten durch einen Datenstromende-Framing-Token des Zwischenverbindungsprotokolls identifiziert sind, der zum Identifizieren der Übergänge codiert ist, und Übergänge von Speicherzugangsprotokolldaten zu Zwischenverbindungsprotokolldaten durch Verbindungsschichtsteuerungs-Flits des Speicherzugangsprotokolls identifiziert sind.

12. Apparat nach Anspruch 1, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner zum Verfolgen von Speichertransaktionen dient, welche die Last- und Speicheroperationen beinhalten.

13. Apparat nach Anspruch 12, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner dient zum:
Identifizieren, dass ein bestimmter der mehreren CPU-Knoten (510a,....., 510n; 510a, , 510e) versagt;
Identifizieren eines Teils der Speichertransaktionen des bestimmten CPU-Knotens (510a,....., 510n; 510a, , 510e); und
Fallenlassen des Teils der Speichertransaktionen des bestimmten CPU-Knotens (510a,....., 510n;

510a, , 510e), während alle anderen Speichertransaktionen aufrechterhalten bleiben.

14. Apparat nach Anspruch 1, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner zum Verwalten von Genehmigungen für einen Zugang der mehreren CPU-Knoten (510a, ..., 510n; 510a,....., 510e) zu Daten in der gemeinsam benutzten Speicherressource dient.

15. Apparat nach Anspruch 14, wobei zumindest ein bestimmter der mehreren CPU-Knoten (510a,....., 510n; 510a,....., 510e) von einem Zugang zu zumindest einem ersten Teil des gemeinsam benutzten Speichers (505; 505a, 505b) blockiert ist und einem zweiten der mehreren CPU-Knoten (S 10a, ..., 510n; 510a,....., 510e) ein Zugang zum ersten Teil gewährt ist.

16. Apparat nach Anspruch 1, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner zum Verwalten von Verzeichnisinformationen für Daten in der gemeinsam benutzten Speicherressource dient.

17. Apparat nach Anspruch 16, wobei die Verzeichnisinformationen für jede von mehreren Datenressourcen, die in der gemeinsam benutzten Speicherressource gespeichert sind, identifizieren, ob ein Zugang zur entsprechenden Datenressource ausschließlich für einen der mehreren CPU-Knoten (510a, , 510n; 510a,....., 510e) ist oder von zwei oder mehr der mehreren CPU-Knoten (510a, , 510n; 510a,....., 510e) gemeinsam benutzt wird.

18. Apparat nach Anspruch 17, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner zum Verhandeln einer Zugangsänderung für eine bestimmte der mehreren Datenressourcen dient, wobei die Änderung zumindest eine von einer Zugangsänderung von gemeinsam benutzt zu ausschließlich oder einer Zugangsänderung von ausschließlich zu gemeinsam benutzt aufweist.

19. Apparat nach Anspruch 1, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) an zumindest eine andere Steuerung des gemeinsam benutzten Speichers (505; 505a, 505b) gekoppelt ist, die zumindest eine andere gemeinsam benutzte Speicherressource verwaltet, und die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner zum Kommunizieren von Last-/Speicheroperationen zur anderen Steuerung des gemeinsam benutzten Speichers (505; 505a, 505b) dient, um den mehreren CPU-Knoten (510a,....., 510n; 510a,....., 510e) einen Zugang zu dem anderen gemeinsam benutzten Speicher zu ermöglichen.

20. Apparat nach Anspruch 1, wobei die Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) ferner zum Abbilden von Adressinformationen in den Last- und Speicheroperationen auf entsprechende Datenressourcen dient, die in der gemeinsam benutzten Speicherressource gespeichert sind.

21. Apparat aufweisend:
eine I/O-Logik zum:
Senden einer Speicherzugangsanfrage zu einer Steuerung (515; 515a, 515b) eines gemeinsam benutzten Speichers (505; 505a, 505b), wobei die Speicherzugangsanfrage eine Last-/Speicheroperation aufweist und zum Identifizieren einer Adresse einer Datenressource dient, die in einer gemeinsam benutzten Speicherressource enthalten ist, die der Steuerung (515; 515a, 515b) des gemeinsam benutzten Speichers (505; 505a, 505b) entspricht, und jedem von mehreren unabhängigen Knoten (510a,....., 510n; 510a,....., 510e) Zugang zu einem entsprechenden Teil der gemeinsam benutzten Speicherressource gewährt wird, wobei die Speicherzugangsanfrage unter Verwendung eines Verbindungsprotokolls des gemeinsam benutzten Speichers (505; 505a, 505b) gesendet wird, wobei das Verbindungsprotokoll des gemeinsam benutzten Speichers (505; 505a, 505b) ein Speicherzugangsprotokoll enthält, das eine physische Schichtlogik eines anderen Zwischenverbindungsprotokolls benutzt, und das Verbindungsprotokoll des gemeinsam benutzten Speichers (505; 505a, 505b) ein Multiplexen zwischen einer Sendung von Daten der Speicherzugangsprotokolldaten und einer Sendung von Daten des Zwischenverbindungsprotokolls vorsieht,
dadurch gekennzeichnet dass Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch ein Datenstrom-Framing-Token, das zum Identifizieren der Übergänge codiert ist, identifziert werden.

22. Apparat nach Anspruch 21, wobei die Speicherzugangsanfrage eine Lastanfrage aufweist und die I/O-Logik ferner zum Empfangen von Daten, die der Datenressource entsprechen, als Antwort auf die Lastanfrage dient.

23. Apparat nach Anspruch 21, wobei die Speicherzugangsanfrage eine Speicheranfrage aufweist.

24. Apparat nach Anspruch 23, wobei Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch zumindest eines der folgenden identifiziert werden:
(a) eine Sync-Kopfzeile, die zum Identifizieren der Übergänge codiert ist;

(b) einen Datenstart-Framing-Token, das zum Identifizieren der Übergänge codiert ist; und
(c) einen Datenstromende-Framing-Token, das zum Identifizieren der Übergänge codiert ist.

25. Verfahren aufweisend:
Empfangen einer ersten Last-/Speicher-Nachricht von einem ersten unabhängigen CPU-Knoten (510a,....., 510n; 510a,....., 510e), wobei die erste Last-/Speicher-Nachricht bestimmte Daten in einem gemeinsam benutzten Speicher (505; 505a, 505b) identifiziert,
Vorsehen eines Zugangs zu den bestimmten Daten für den ersten CPU-Knoten (510a,....., 510n; 510a,....., 510e) als Antwort auf die erste Last-/Speicher-Nachricht;
Empfangen einer zweiten Last-/Speicher-Nachricht von einem zweiten unabhängigen CPU-Knoten (510a,....., 510n; 510a,....., 510e), wobei die zweite Last-/Speicher-Nachricht bestimmte Daten in einem gemeinsam benutzten Speicher (505; 505a, 505b) identifiziert, und
Vorsehen eines Zugangs zu den bestimmten Daten für den zweiten CPU-Speicher als Antwort auf die zweite Last-/Speicher-Nachricht,
dadurch gekennzeichnet dass Übergänge zwischen Zwischenverbindungsprotokolldaten und Speicherzugangsprotokolldaten durch ein Datenstrom-Framing-Token, das zum Identifizieren der Übergänge codiert ist, identifziert werden.

Es folgen 17 Seiten Zeichnungen

Anhängende Zeichnungen

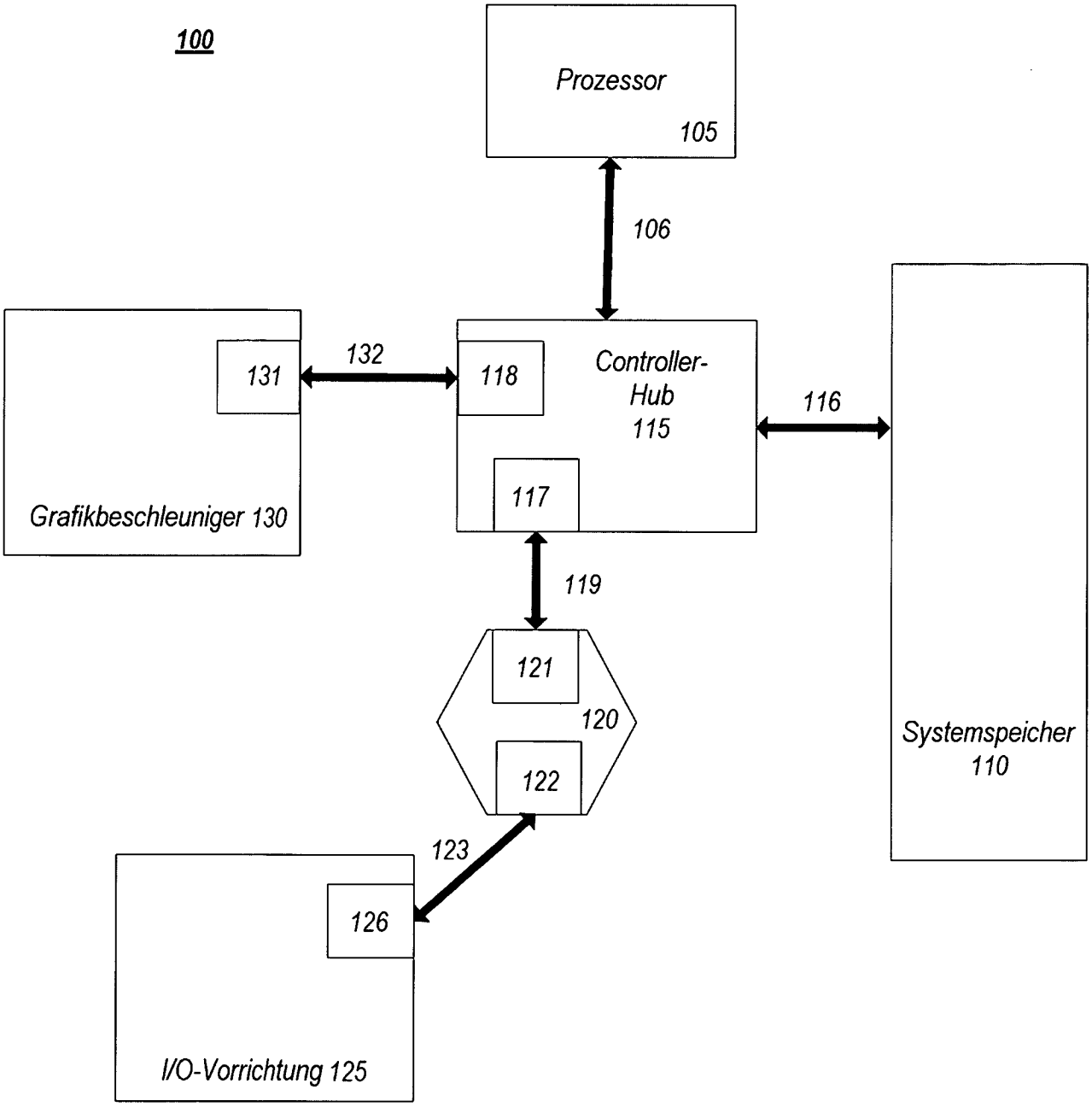


FIG. 1

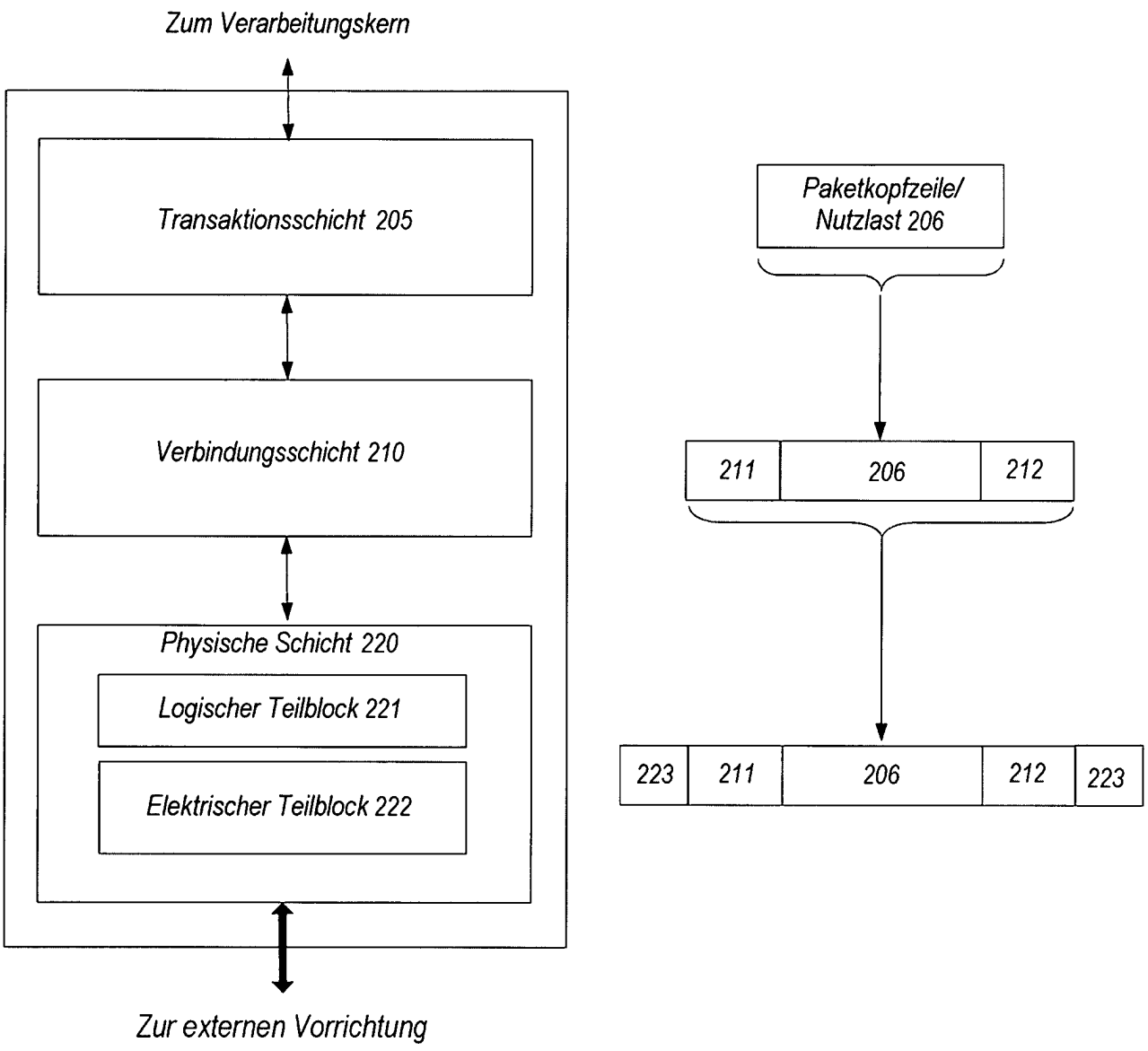

Schichtenförmiger Protokollstapel 200
Zum Verarbeitungskern
Transaktionsschicht 205
Verbindungsschicht 210
Physische Schicht 220
Logischer Teilblock 221
Elektrischer Teilblock 222
Zur externen Vorrichtung
Paketkopfzeile/ Nutzlast 206
211
206
212
223
211
206
212
223

FIG. 2

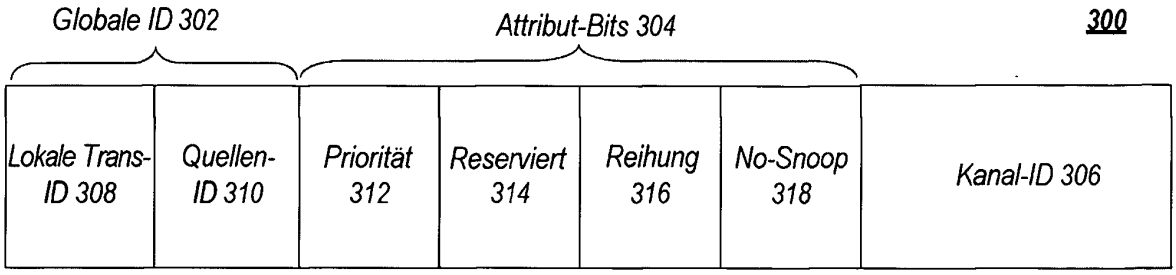

Globale ID 302
Attribut-Bits 304
300
Lokale Trans-ID 308
Quellen-ID 310
Priorität 312
Reserviert 314
Reihung 316
No-Snoop 318
Kanal-ID 306

FIG. 3

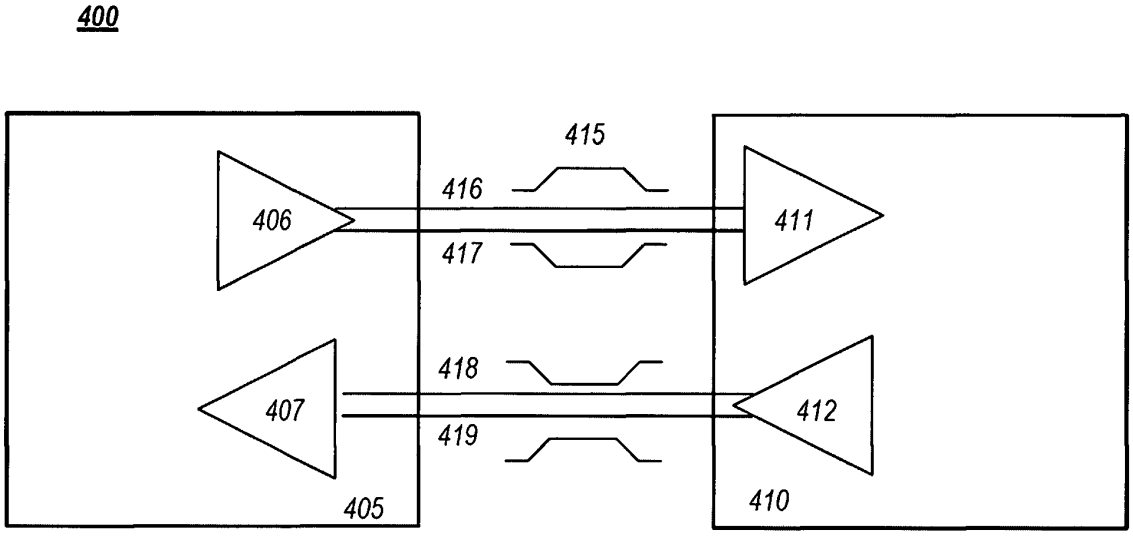

400
415
416
406
411
417
418
407
412
419
405
410

FIG. 4

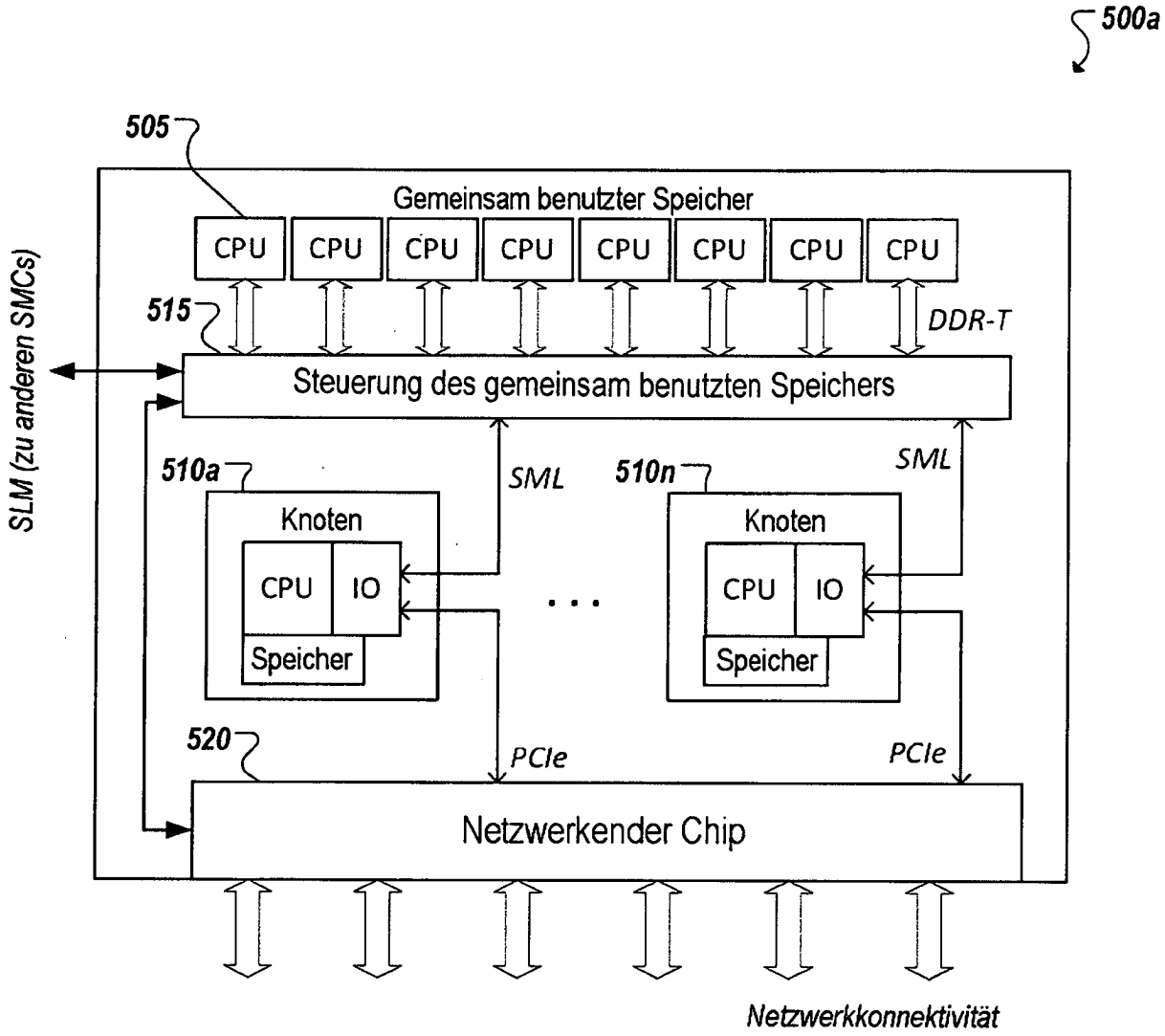

500a
505
Gemeinsam benutzter Speicher
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
515
DDR-T
SLM (zu anderen SMCs)
Steuerung des gemeinsam benutzten Speichers
510a
Knoten
CPU
IO
Speicher
SML
510n
Knoten
CPU
IO
Speicher
SML
520
PCIe
PCIe
Netzwerkender Chip
Netzwerkkonnektivität

FIG. 5A

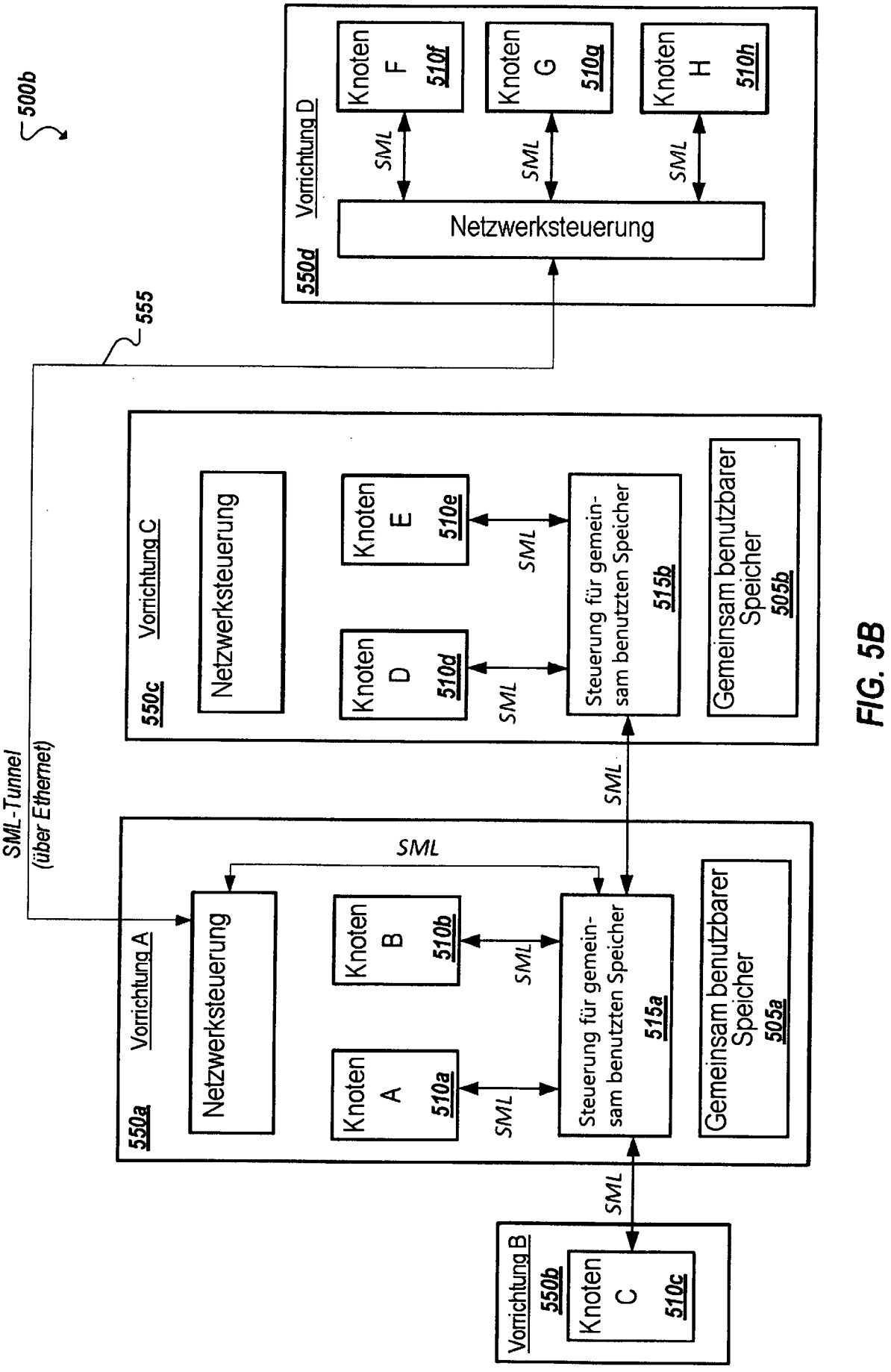

500b
555
SML-Tunnel
(über Ethernet)
Vorrichtung A
550a
Netzwerksteuerung
Knoten A
510a
Knoten B
510b
Steuerung für gemeinsam benutzten Speicher
515a
Gemeinsam benutzbarer Speicher
505a
SML
Vorrichtung B
550b
Knoten C
510c
Vorrichtung C
550c
Netzwerksteuerung
Knoten D
510d
Knoten E
510e
Steuerung für gemeinsam benutzten Speicher
515b
Gemeinsam benutzbarer Speicher
505b
Vorrichtung D
550d
Netzwerksteuerung
Knoten F
510f
Knoten G
510g
Knoten H
510h

FIG. 5B

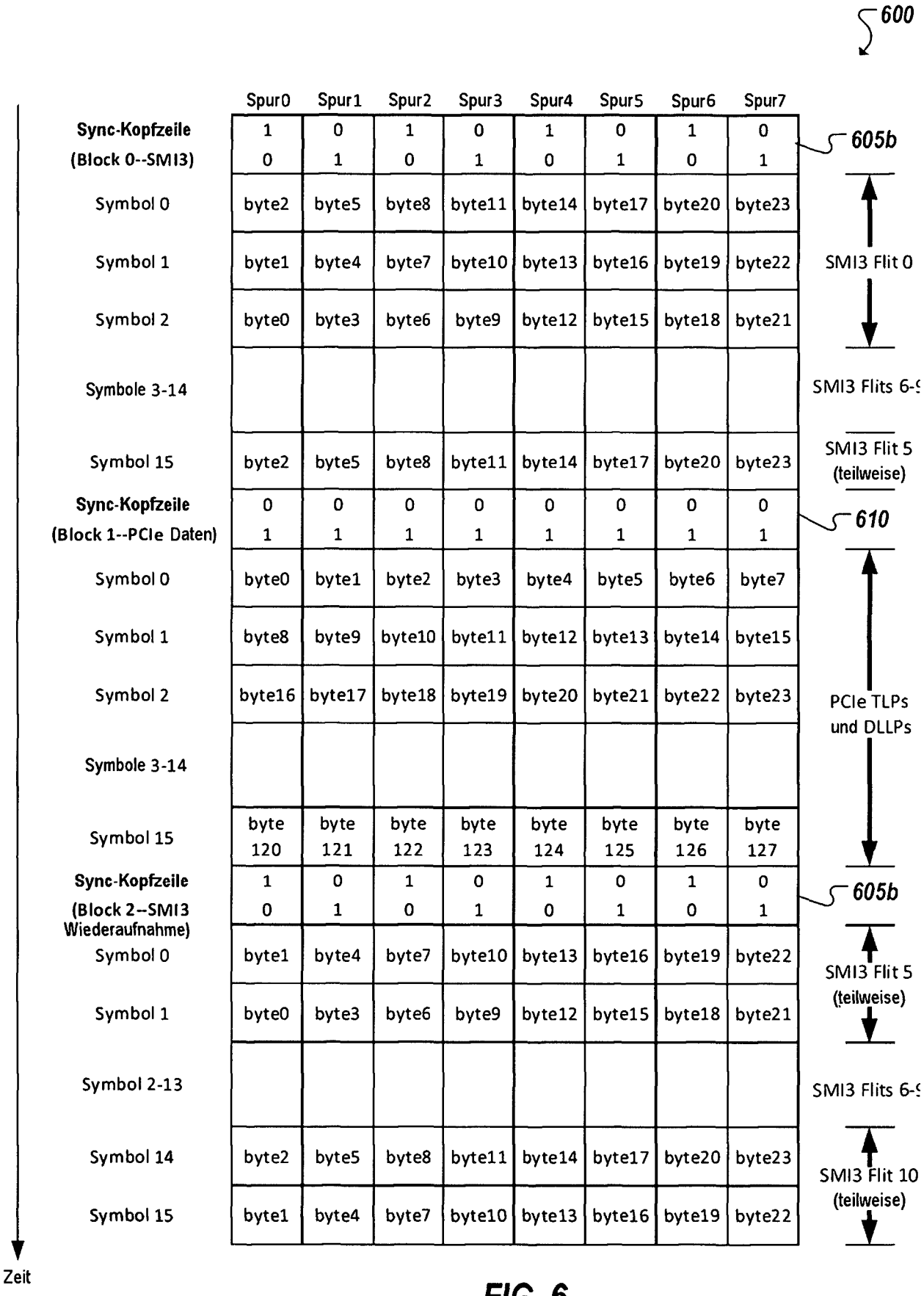

600
Spur0 Spur1 Spur2 Spur3 Spur4 Spur5 Spur6 Spur7
Sync-Kopfzeile (Block 0--SMI3)
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
605b
Symbol 0 byte2 byte5 byte8 byte11 byte14 byte17 byte20 byte23
Symbol 1 byte1 byte4 byte7 byte10 byte13 byte16 byte19 byte22
Symbol 2 byte0 byte3 byte6 byte9 byte12 byte15 byte18 byte21
SMI3 Flit 0
Symbole 3-14
SMI3 Flits 6-
Symbol 15 byte2 byte5 byte8 byte11 byte14 byte17 byte20 byte23
SMI3 Flit 5 (teilweise)
Sync-Kopfzeile (Block 1--PCIe Daten)
0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
610
Symbol 0 byte0 byte1 byte2 byte3 byte4 byte5 byte6 byte7
Symbol 1 byte8 byte9 byte10 byte11 byte12 byte13 byte14 byte15
Symbol 2 byte16 byte17 byte18 byte19 byte20 byte21 byte22 byte23
PCIe TLPs und DLLPs
Symbole 3-14
Symbol 15 byte 120 byte 121 byte 122 byte 123 byte 124 byte 125 byte 126 byte 127
Sync-Kopfzeile (Block 2--SMI3 Wiederaufnahme)
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
605b
Symbol 0 byte1 byte4 byte7 byte10 byte13 byte16 byte19 byte22
Symbol 1 byte0 byte3 byte6 byte9 byte12 byte15 byte18 byte21
SMI3 Flit 5 (teilweise)
Symbol 2-13
SMI3 Flits 6-
Symbol 14 byte2 byte5 byte8 byte11 byte14 byte17 byte20 byte23
Symbol 15 byte1 byte4 byte7 byte10 byte13 byte16 byte19 byte22
SMI3 Flit 10 (teilweise)
Zeit

FIG. 6

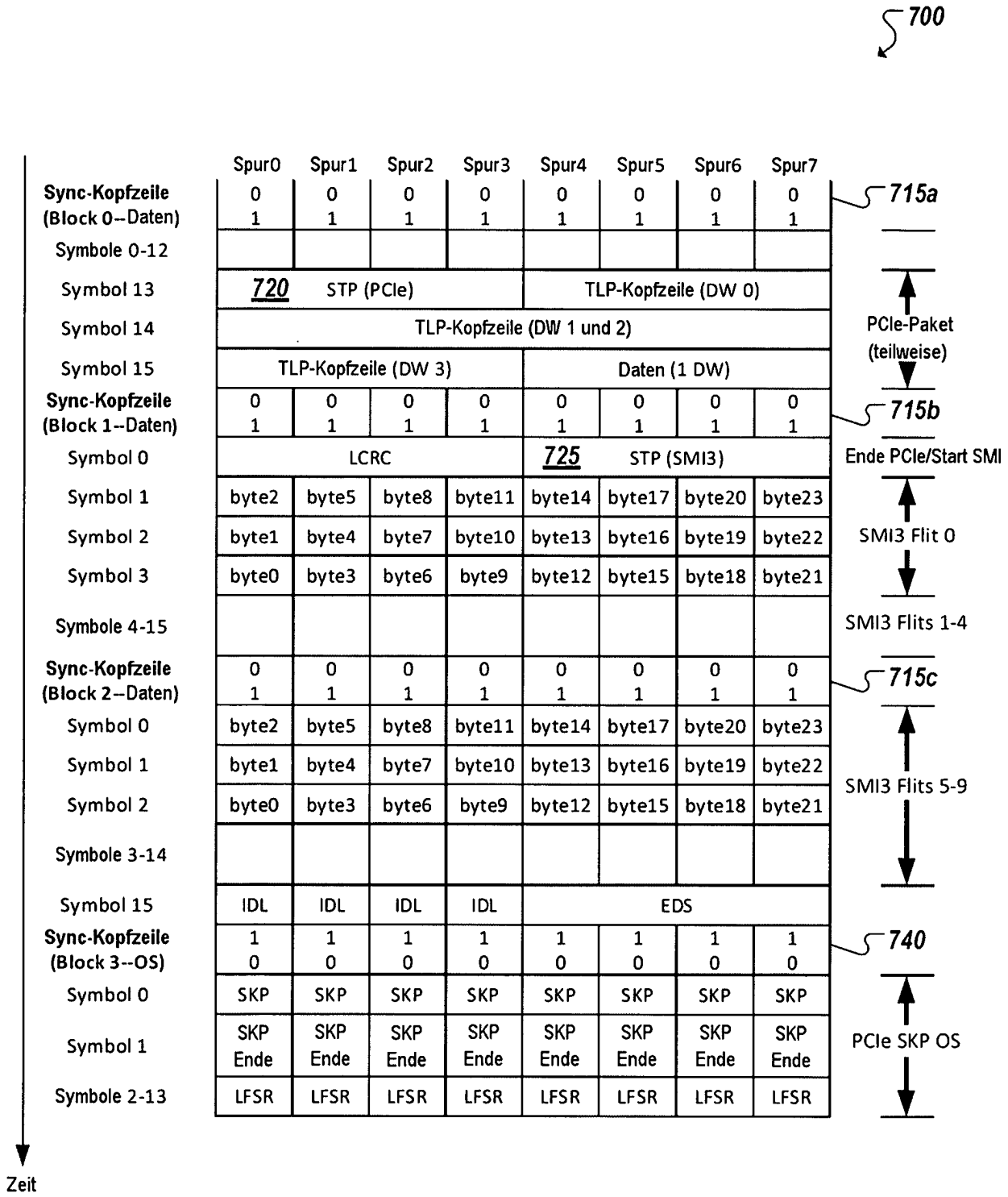

700
Spur0 Spur1 Spur2 Spur3 Spur4 Spur5 Spur6 Spur7
Sync-Kopfzeile (Block 0--Daten)
0 1
715a
Symbole 0-12
Symbol 13
720
STP (PCIe)
TLP-Kopfzeile (DW 0)
Symbol 14
TLP-Kopfzeile (DW 1 und 2)
Symbol 15
TLP-Kopfzeile (DW 3)
Daten (1 DW)
PCIe-Paket (teilweise)
Sync-Kopfzeile (Block 1--Daten)
715b
Symbol 0
LCRC
725
STP (SMI3)
Ende PCIe/Start SMI
Symbol 1
byte2 byte5 byte8 byte11 byte14 byte17 byte20 byte23
Symbol 2
byte1 byte4 byte7 byte10 byte13 byte16 byte19 byte22
SMI3 Flit 0
Symbol 3
byte0 byte3 byte6 byte9 byte12 byte15 byte18 byte21
Symbole 4-15
SMI3 Flits 1-4
Sync-Kopfzeile (Block 2--Daten)
715c
Symbol 0
byte2 byte5 byte8 byte11 byte14 byte17 byte20 byte23
Symbol 1
byte1 byte4 byte7 byte10 byte13 byte16 byte19 byte22
Symbol 2
byte0 byte3 byte6 byte9 byte12 byte15 byte18 byte21
SMI3 Flits 5-9
Symbole 3-14
Symbol 15
IDL IDL IDL IDL
EDS
Sync-Kopfzeile (Block 3--OS)
1 0
740
Symbol 0
SKP SKP SKP SKP SKP SKP SKP SKP
Symbol 1
SKP Ende
PCIe SKP OS
Symbole 2-13
LFSR LFSR LFSR LFSR LFSR LFSR LFSR LFSR
Zeit

FIG. 7A

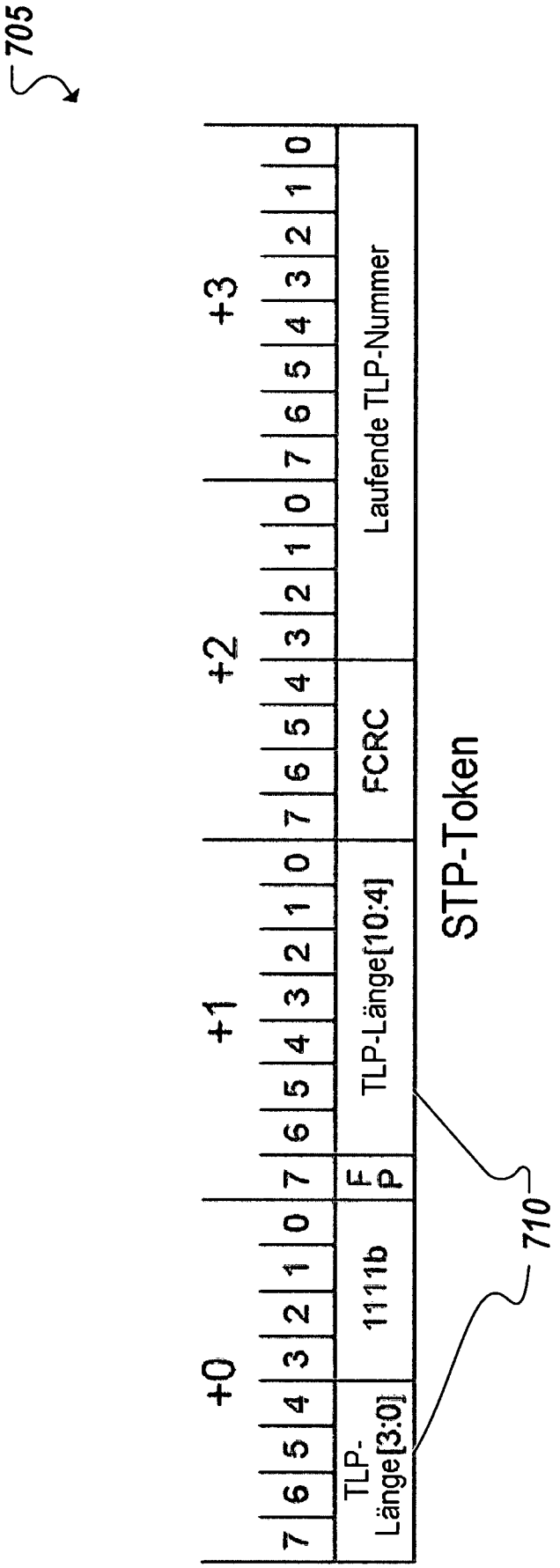

705
+0
+1
+2
+3
7 6 5 4 3 2 1 0
TLP-Länge[3:0]
1111b
F P
TLP-Länge[10:4]
FCRC
Laufende TLP-Nummer
STP-Token
710

FIG. 7B

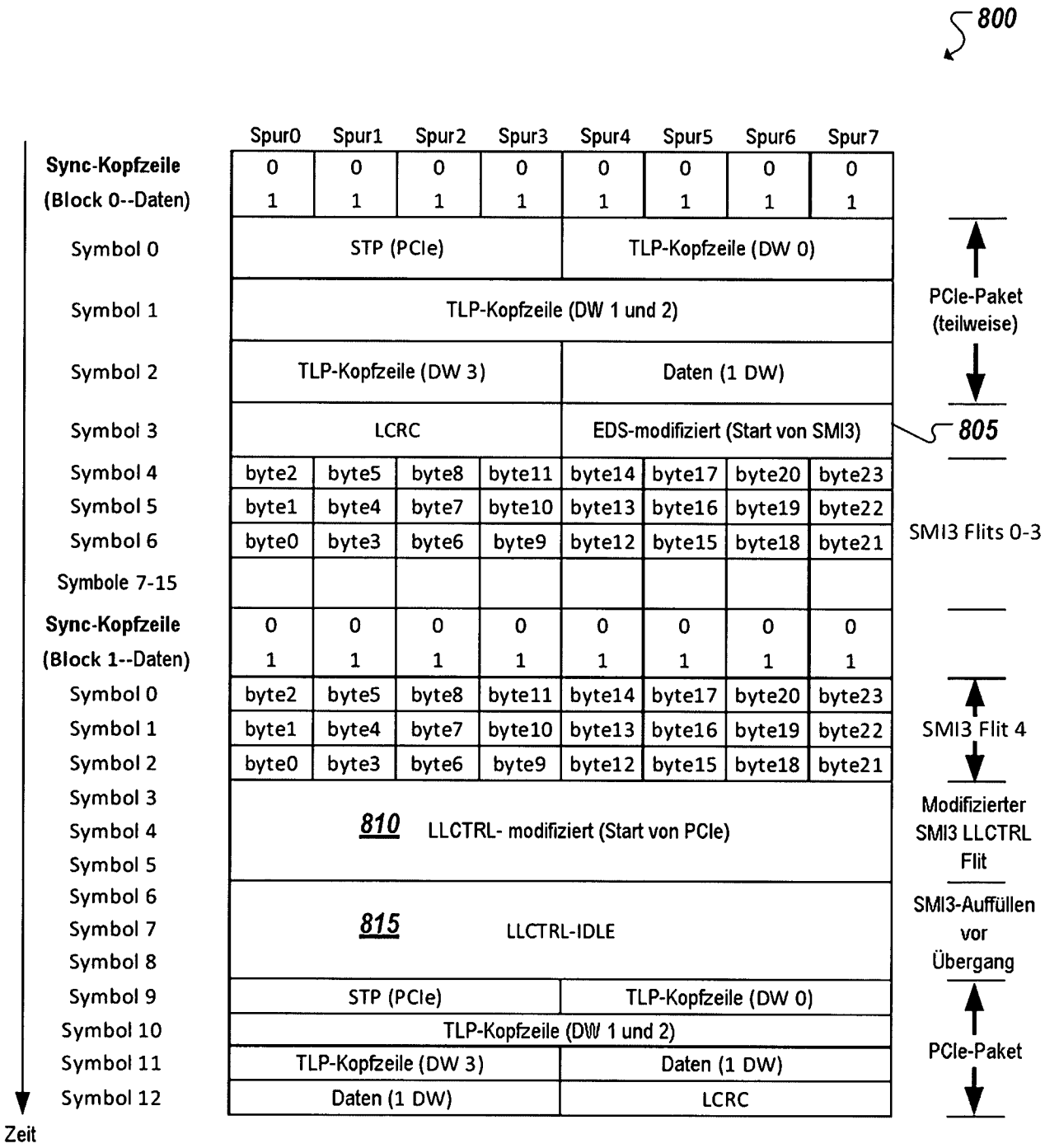

800
Spur0 Spur1 Spur2 Spur3 Spur4 Spur5 Spur6 Spur7
Sync-Kopfzeile
(Block 0--Daten)
Symbol 0
Symbol 1
Symbol 2
Symbol 3
Symbol 4
Symbol 5
Symbol 6
Symbole 7-15
Sync-Kopfzeile
(Block 1--Daten)
Symbol 0
Symbol 1
Symbol 2
Symbol 3
Symbol 4
Symbol 5
Symbol 6
Symbol 7
Symbol 8
Symbol 9
Symbol 10
Symbol 11
Symbol 12
Zeit
STP (PCIe)
TLP-Kopfzeile (DW 0)
TLP-Kopfzeile (DW 1 und 2)
TLP-Kopfzeile (DW 3)
Daten (1 DW)
LCRC
EDS-modifiziert (Start von SMI3)
805
byte2 byte5 byte8 byte11 byte14 byte17 byte20 byte23
byte1 byte4 byte7 byte10 byte13 byte16 byte19 byte22
byte0 byte3 byte6 byte9 byte12 byte15 byte18 byte21
810
LLCTRL- modifiziert (Start von PCIe)
815
LLCTRL-IDLE
PCIe-Paket (teilweise)
SMI3 Flits 0-3
SMI3 Flit 4
Modifizierter SMI3 LLCTRL Flit
SMI3-Auffüllen vor Übergang
PCIe-Paket

FIG. 8

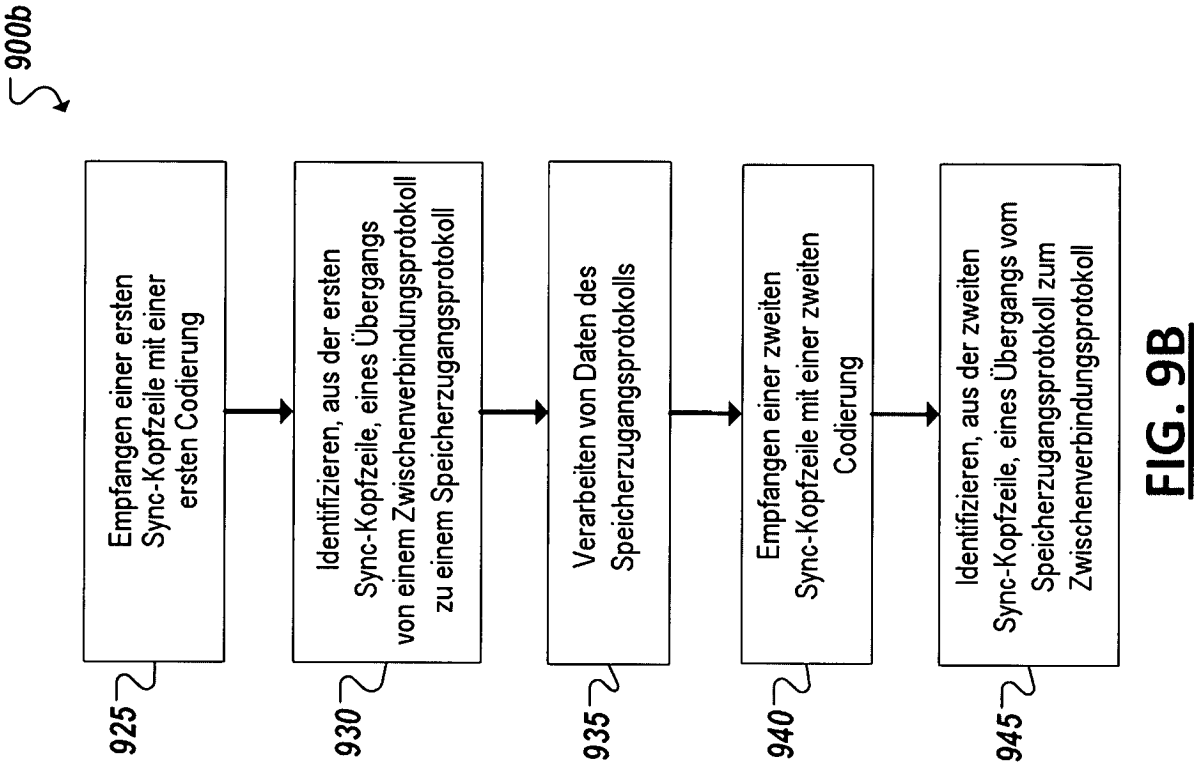

900b
Empfangen einer ersten Sync-Kopfzeile mit einer ersten Codierung
925
Identifizieren, aus der ersten Sync-Kopfzeile, eines Übergangs von einem Zwischenverbindungsprotokoll zu einem Speicherzugangsprotokoll
930
Verarbeiten von Daten des Speicherzugangsprotokolls
935
Empfangen einer zweiten Sync-Kopfzeile mit einer zweiten Codierung
940
Identifizieren, aus der zweiten Sync-Kopfzeile, eines Übergangs vom Speicherzugangsprotokoll zum Zwischenverbindungsprotokoll
945

FIG. 9B

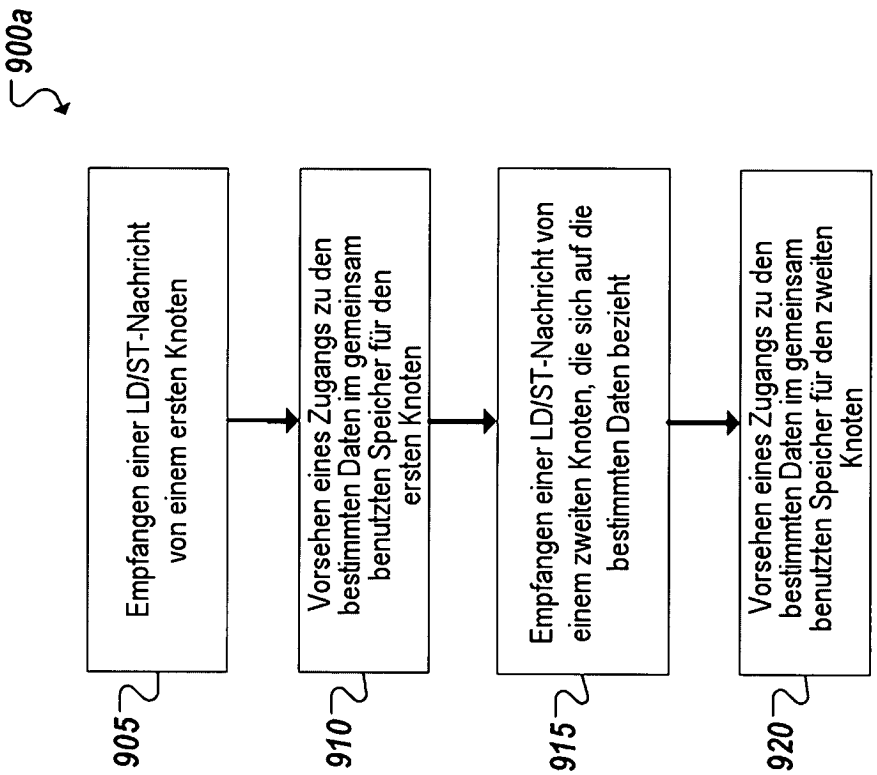

900a
Empfangen einer LD/ST-Nachricht von einem ersten Knoten
905
Vorsehen eines Zugangs zu den bestimmten Daten im gemeinsam benutzten Speicher für den ersten Knoten
910
Empfangen einer LD/ST-Nachricht von einem zweiten Knoten, die sich auf die bestimmten Daten bezieht
915
Vorsehen eines Zugangs zu den bestimmten Daten im gemeinsam benutzten Speicher für den zweiten Knoten
920

FIG. 9A

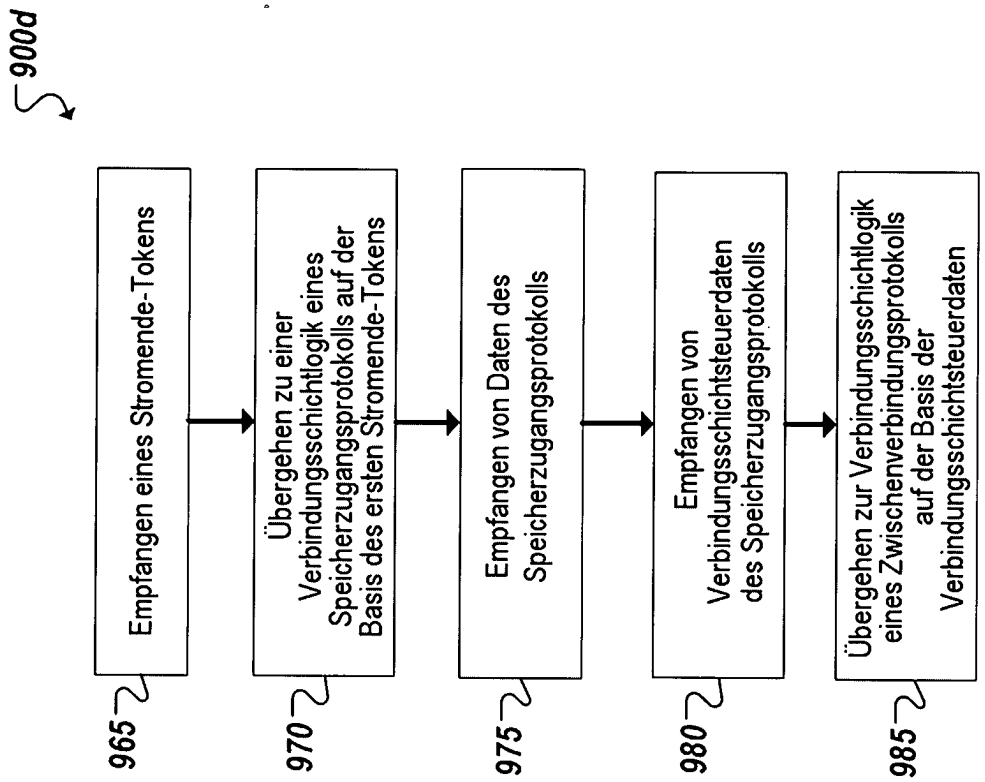

900d
Empfangen eines Stromende-Tokens
965
Übergehen zu einer Verbindungsschichtlogik eines Speicherzugangsprotokolls auf der Basis des ersten Stromende-Tokens
970
Empfangen von Daten des Speicherzugangsprotokolls
975
Empfangen von Verbindungsschichtsteuerdaten des Speicherzugangsprotokolls
980
Übergehen zur Verbindungsschichtlogik eines Zwischenverbindungsprotokolls auf der Basis der Verbindungsschichtsteuerdaten
985

FIG. 9D

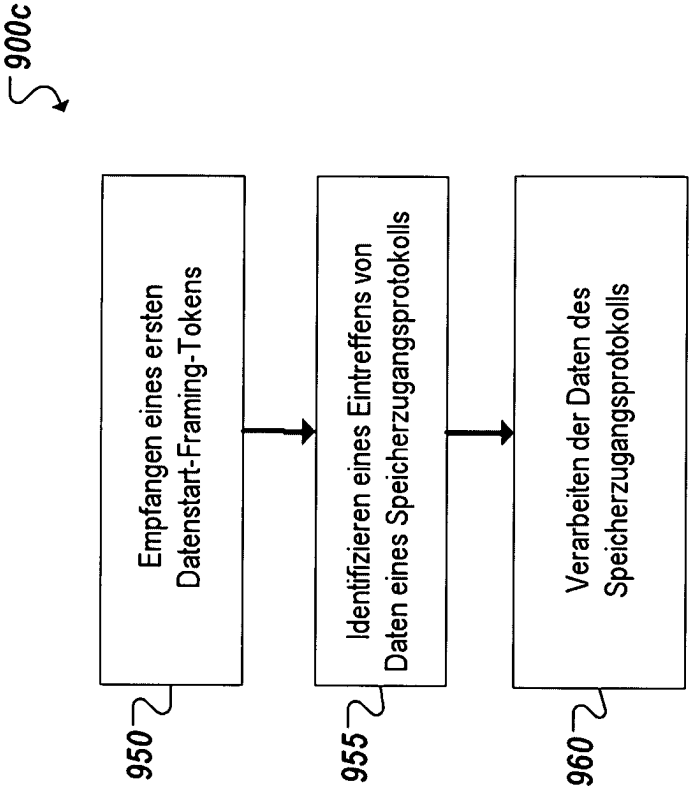

900c
Empfangen eines ersten Datenstart-Framing-Tokens
950
Identifizieren eines Eintreffens von Daten eines Speicherzugangsprotokolls
955
Verarbeiten der Daten des Speicherzugangsprotokolls
960

FIG. 9C

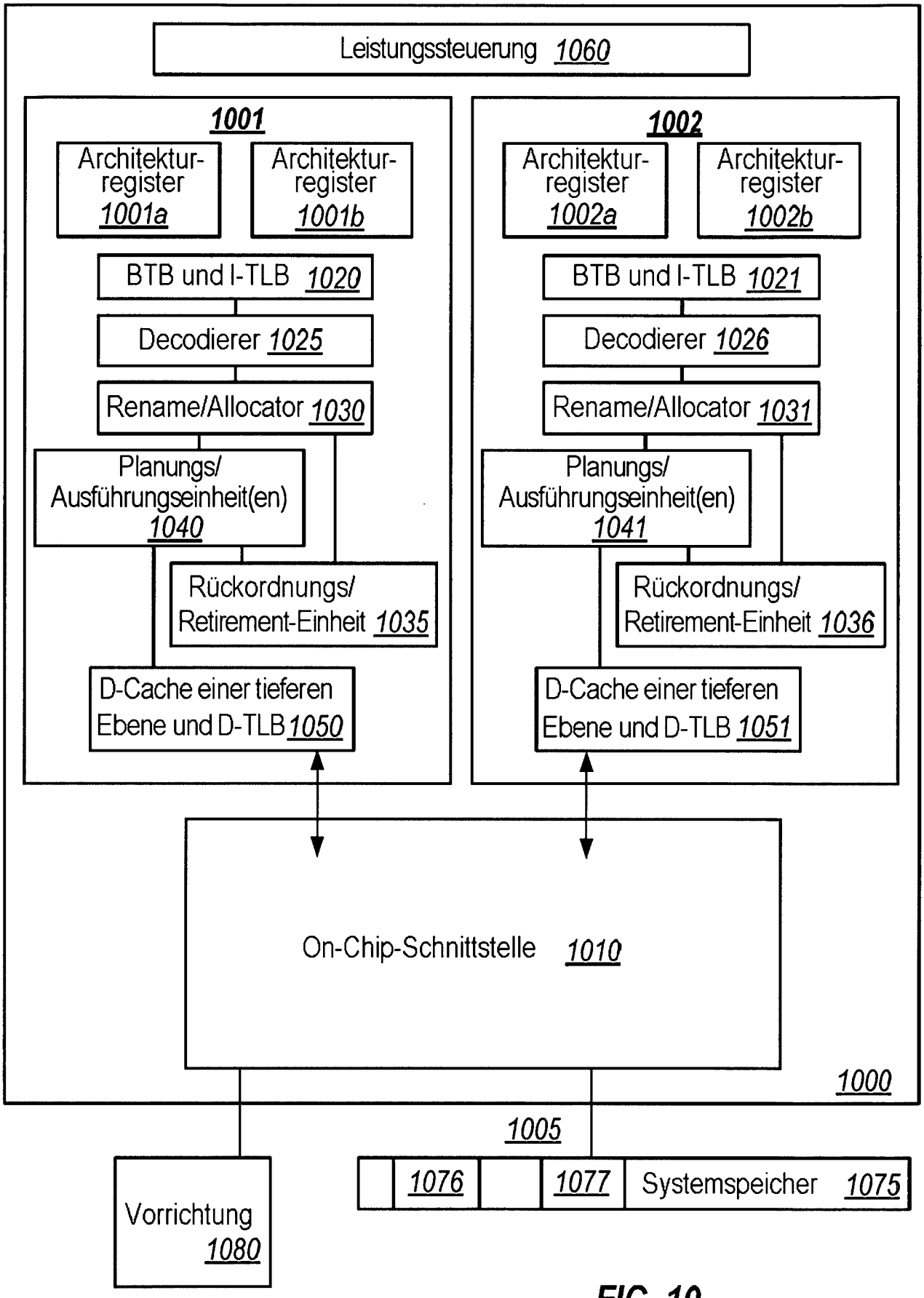

Leistungssteuerung 1060
1001
Architektur-register 1001a
Architektur-register 1001b
BTB und I-TLB 1020
Decodierer 1025
Rename/Allocator 1030
Planungs/ Ausführungseinheit(en) 1040
Rückordnungs/ Retirement-Einheit 1035
D-Cache einer tieferen Ebene und D-TLB 1050
1002
Architektur-register 1002a
Architektur-register 1002b
BTB und I-TLB 1021
Decodierer 1026
Rename/Allocator 1031
Planungs/ Ausführungseinheit(en) 1041
Rückordnungs/ Retirement-Einheit 1036
D-Cache einer tieferen Ebene und D-TLB 1051
On-Chip-Schnittstelle 1010
1000
1005
1076
1077
Systemspeicher 1075
Vorrichtung 1080

FIG. 10

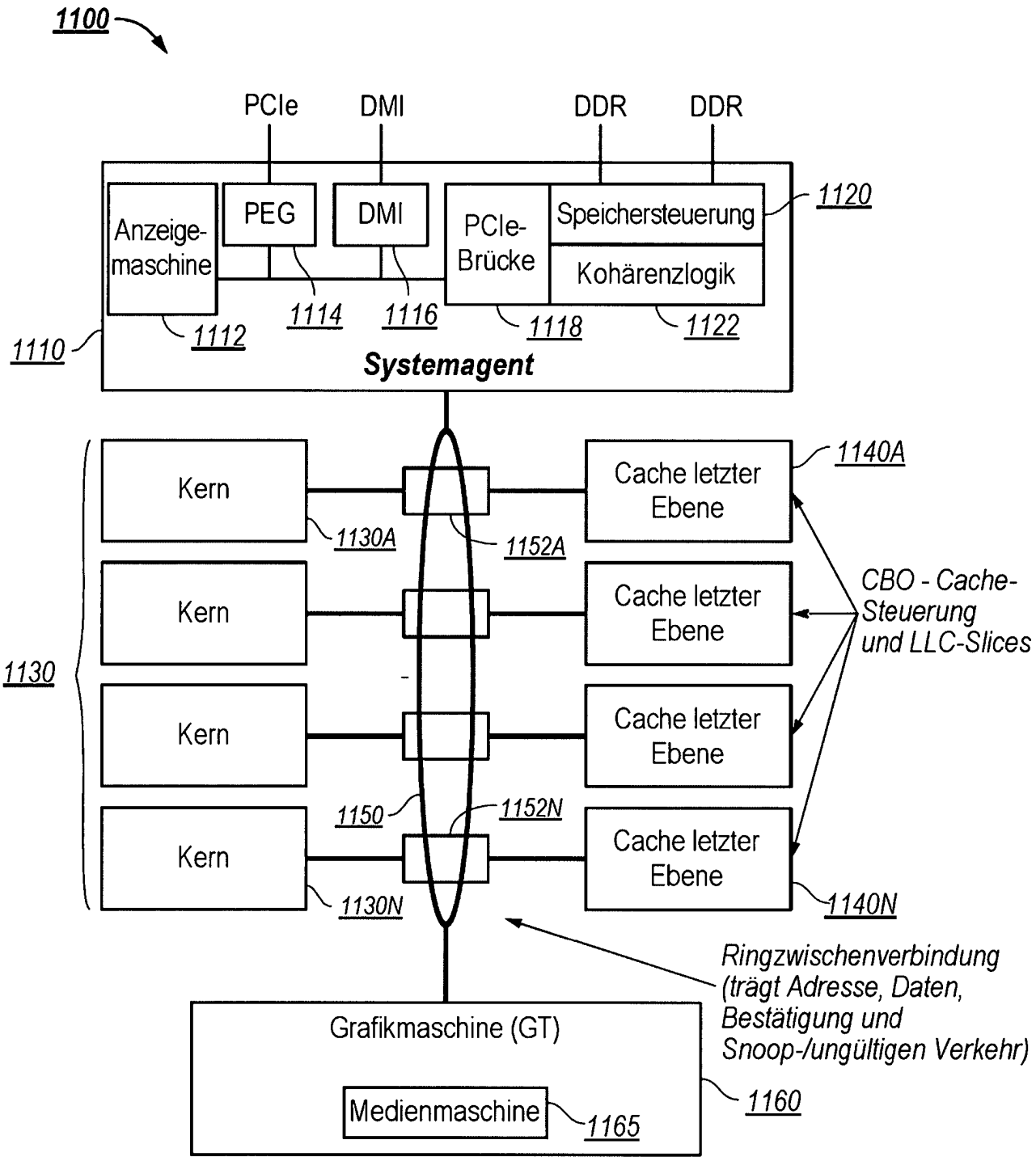

1100
PCIe
DMI
DDR
DDR
Anzeige-maschine
PEG
DMI
PCIe-Brücke
Speichersteuerung
Kohärenzlogik
1120
1112
1114
1116
1118
1122
1110
Systemagent
Kern
Kern
Kern
Kern
1130
1130A
1130N
1152A
1150
1152N
Cache letzter Ebene
Cache letzter Ebene
Cache letzter Ebene
Cache letzter Ebene
1140A
1140N
CBO - Cache-Steuerung und LLC-Slices
Ringzwischenverbindung (trägt Adresse, Daten, Bestätigung und Snoop-/ungültigen Verkehr)
Grafikmaschine (GT)
Medienmaschine
1165
1160

FIG. 11

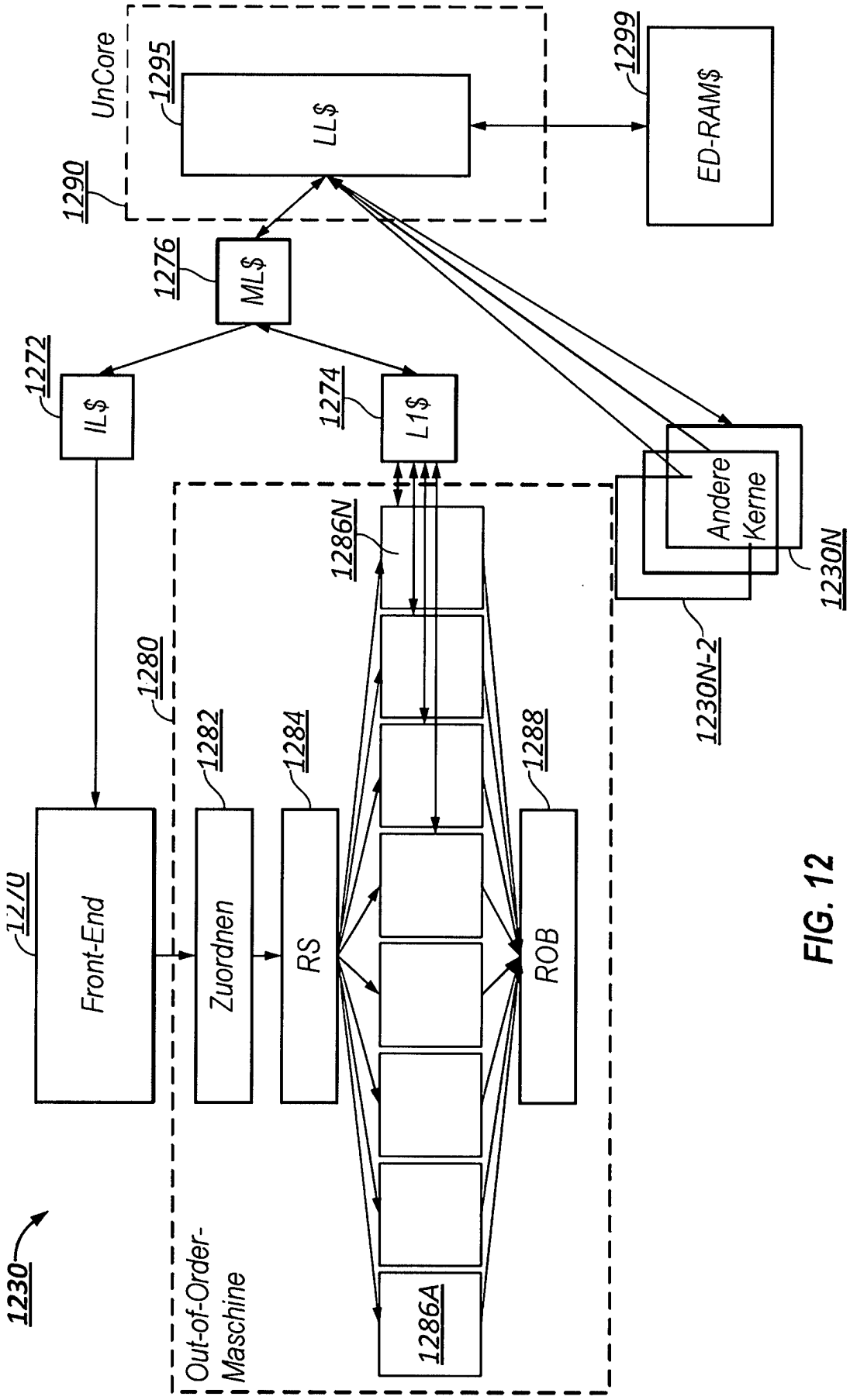

1230
Out-of-Order-Maschine
1210
Front-End
1280
1282
Zuordnen
1284
RS
1286N
1286A
1288
ROB
1230N
Andere Kerne
1230N-2
1272
IL$
1274
L1$
1276
ML$
1290
UnCore
1295
LL$
1299
ED-RAM$

FIG. 12

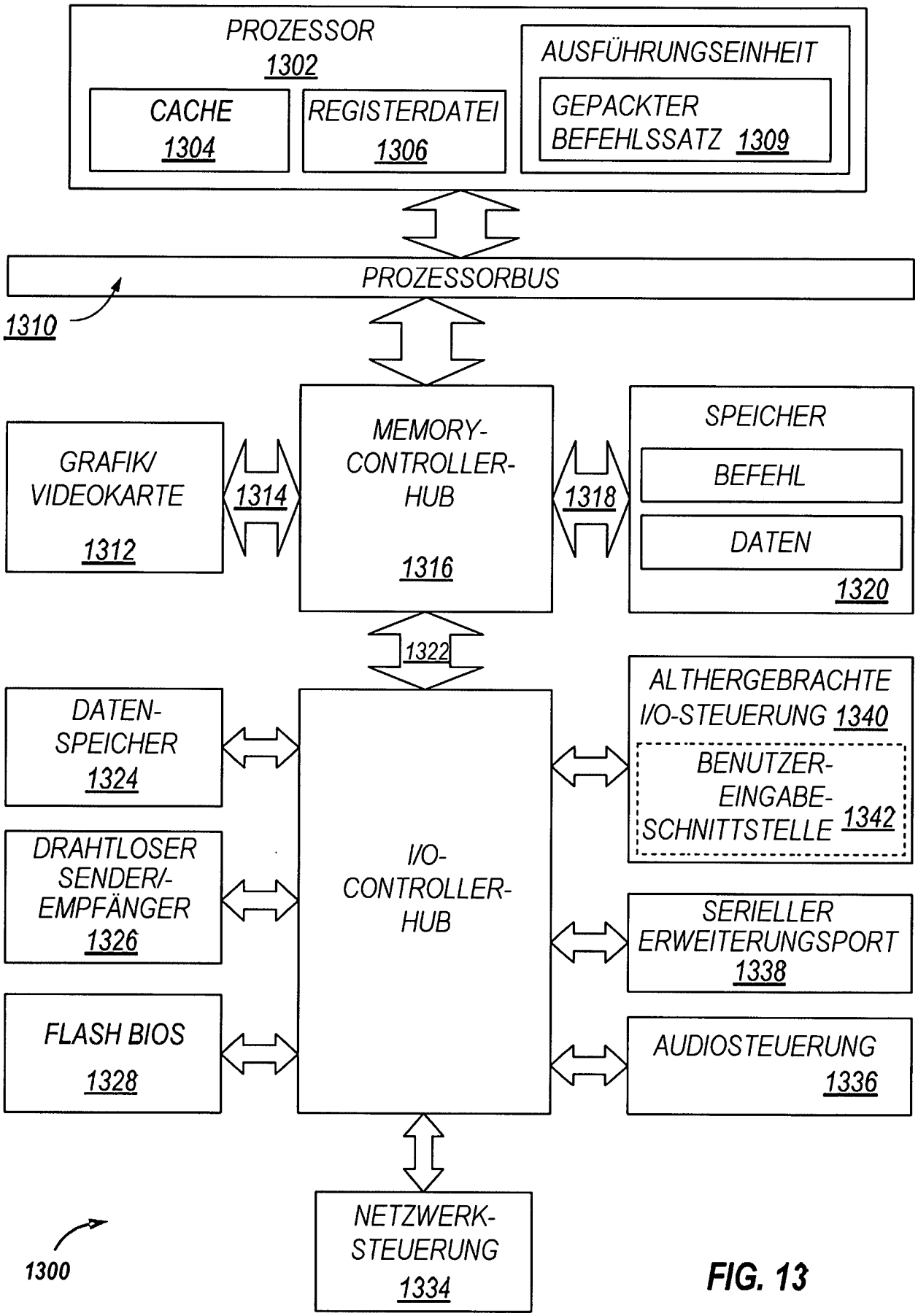

PROZESSOR
1302
CACHE
1304
REGISTERDATEI
1306
AUSFÜHRUNGSEINHEIT
GEPACKTER BEFEHLSSATZ 1309
PROZESSORBUS
1310
GRAFIK/ VIDEOKARTE
1312
1314
MEMORY-CONTROLLER-HUB
1316
1318
SPEICHER
BEFEHL
DATEN
1320
1322
DATEN-SPEICHER
1324
DRAHTLOSER SENDER/-EMPFÄNGER
1326
FLASH BIOS
1328
I/O-CONTROLLER-HUB
ALTHERGEBRACHTE I/O-STEUERUNG 1340
BENUTZER-EINGABE-SCHNITTSTELLE 1342
SERIELLER ERWEITERUNGSPORT
1338
AUDIOSTEUERUNG
1336
NETZWERK-STEUERUNG
1334
1300

FIG. 13

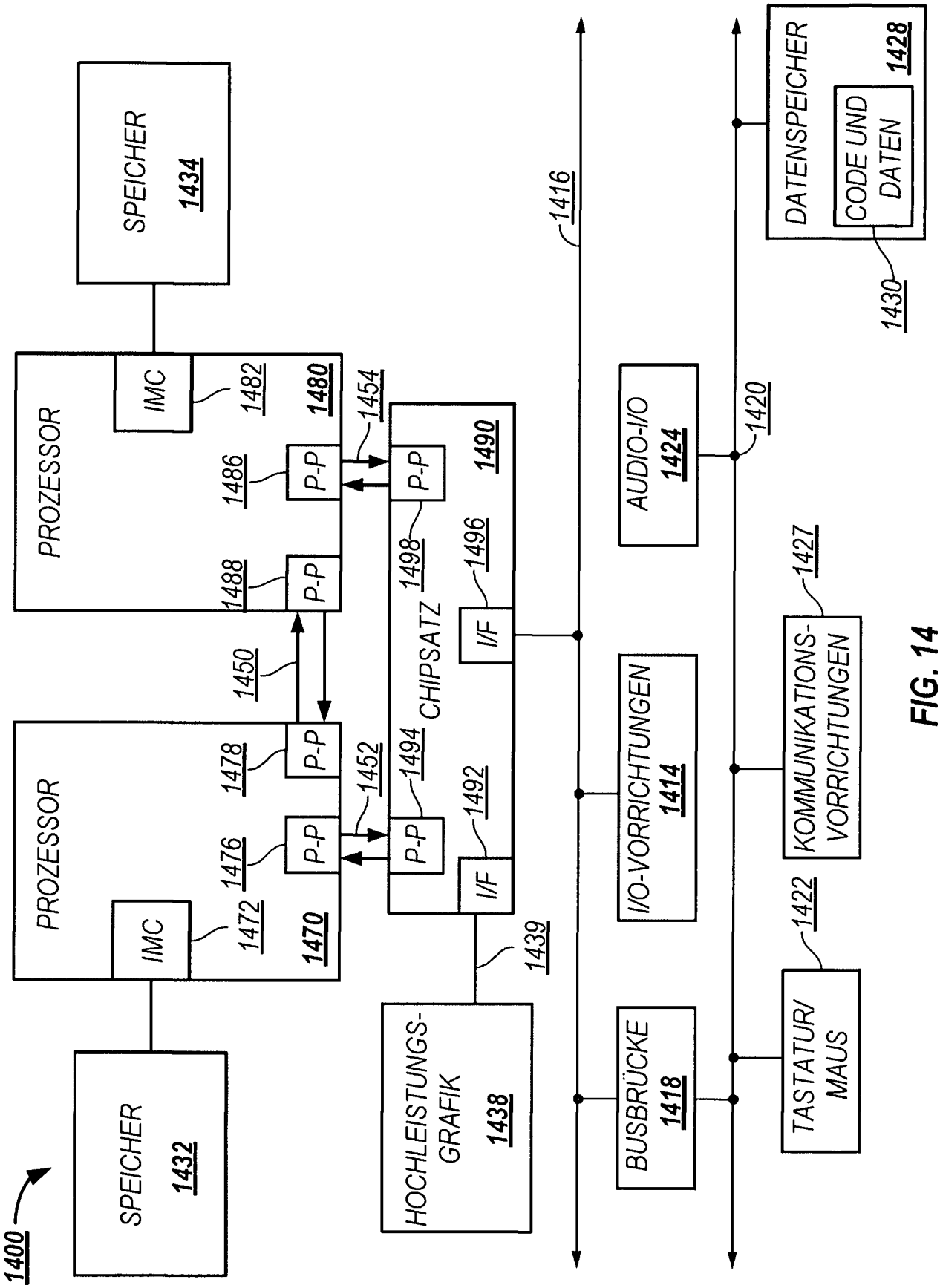

1400
SPEICHER 1432
SPEICHER 1434
PROZESSOR 1470
IMC 1472
P-P 1476
P-P 1478
PROZESSOR 1480
IMC 1482
P-P 1486
P-P 1488
1450
1452
1454
CHIPSATZ 1490
P-P 1494
P-P 1498
I/F 1492
I/F 1496
1439
HOCHLEISTUNGS-GRAFIK 1438
1416
BUSBRÜCKE 1418
I/O-VORRICHTUNGEN 1414
AUDIO-I/O 1424
1420
TASTATUR/ MAUS 1422
KOMMUNIKATIONS-VORRICHTUNGEN 1427
DATENSPEICHER 1428
CODE UND DATEN
1430

FIG. 14

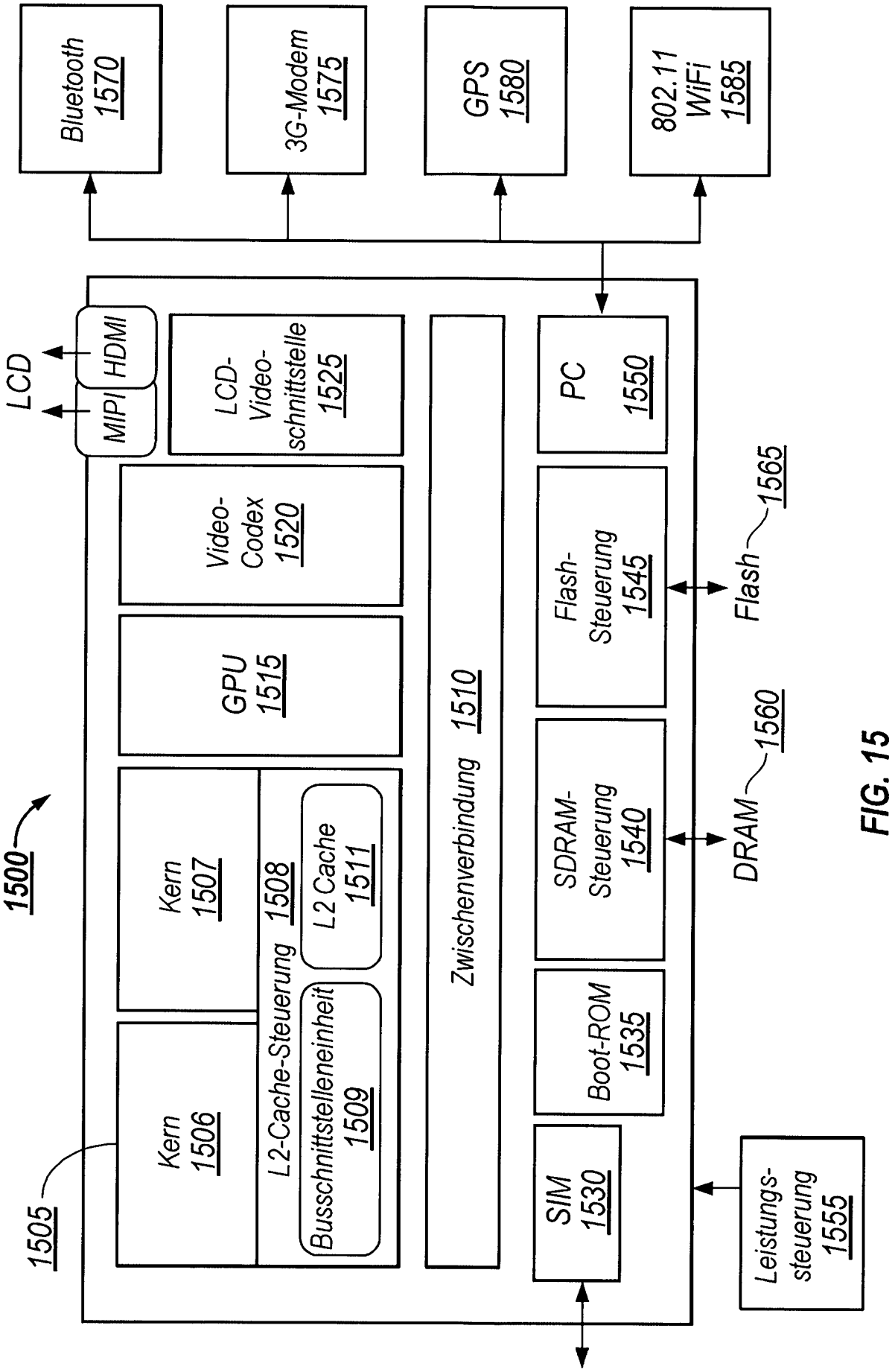

Bluetooth 1570
3G-Modem 1575
GPS 1580
802.11 WiFi 1585
LCD
MIPI
HDMI
LCD-Video-schnittstelle 1525
PC 1550
Video-Codex 1520
GPU 1515
Zwischenverbindung 1510
Flash-Steuerung 1545
Flash 1565
SDRAM-Steuerung 1540
DRAM 1560
Boot-ROM 1535
SIM 1530
Leistungs-steuerung 1555
1500
1505
Kern 1506
Kern 1507
L2-Cache-Steuerung 1508
Busschnittstelleneinheit 1509
L2 Cache 1511

FIG. 15