(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 102 19 359 B4** 2007.05.03

(12) **Patentschrift**

(21) Aktenzeichen: **102 19 359.2**
(22) Anmeldetag: **30.04.2002**
(43) Offenlegungstag: **27.11.2003**
(45) Veröffentlichungstag der Patenterteilung: **03.05.2007**

(51) Int Cl.[8]: ***G06F 13/00*** (2006.01)

(73) Patentinhaber:
Advanced Micro Devices, Inc., Sunnyvale, Calif., US

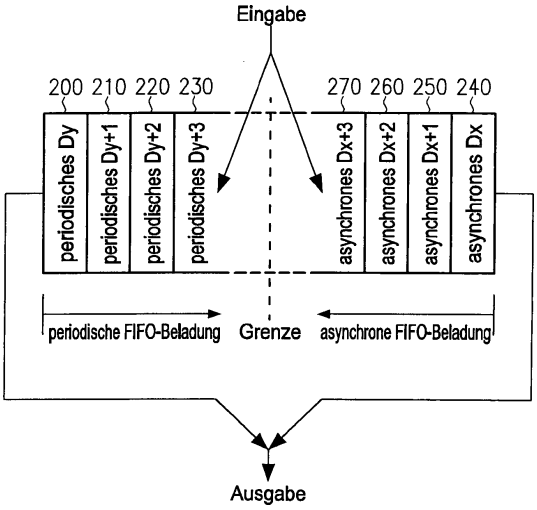
(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(72) Erfinder:
Hesse, Siegfried Kay, 01129 Dresden, DE; Gulick, Dale E., Astoria, Tex., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
DE 197 40 738 A1
US 50 43 981 A

(54) Bezeichnung: **Vorrichtung und Verfahren mit einem Queuemechanismus**

(57) Hauptanspruch: Vorrichtung mit einem Queuemechanismus, umfassend:
eine Datenspeichereinrichtung (200-270, 300, 1445, 1520) zum Speichern einer Vielzahl von Datenelementen, wobei die Datenspeichereinrichtung eine Vielzahl von Registerelementen (200-270) enthält, die jeweils wenigstens ein Datenelement speichern können, wobei die Vielzahl von Registerelementen eingerichtet ist, um eine Sequenz (300) von Registerelementen zu bilden; und
eine Steuereinrichtung (1500) zum Steuern der Datenspeichereinrichtung, um erste Daten in einem ersten Teil (200-230) der Sequenz von Registerelementen und zweite Daten in einem zweiten Teil (240-270) der Sequenz von Registerelementen zu speichern, wobei der erste Teil und der zweite Teil von variablen Längen sind und die Summe der variablen Längen gleich der Länge der Sequenz von Registerelementen ist,
wobei die Steuereinrichtung eingerichtet ist zum Steuern sowohl des ersten als auch des zweiten Teils der Sequenz von Registerelementen, um die jeweiligen Daten FIFO-mäßig zu speichern, und
wobei der erste und zweite Teil der Sequenz von Registerelementen jeweils...



Beschreibung

HINTERGRUND DER ERFINDUNG

1. Gebiet der Erfindung

[0001] Die Erfindung betrifft allgemein Vorrichtungen und Verfahren mit Queuemechanismen und insbesondere Vorrichtungen mit gesteuerten Datenspeichereinrichtungen sowie entsprechende Verfahren zum Speichern einer Vielzahl von Datenelementen.

2. Beschreibung des Standes der Technik

[0002] Übliche Motherboardlayouts umfassen eine Anzahl von Hardwarekomponenten einschließlich einer CPU (Central Processing Unit, zentrale Verarbeitungseinheit), einer Northbridge, einer Southbridge und einem Systemspeicher. Die Northbridge ist üblicherweise ein einzelner Chip in einem Core-Logic-Chipsatz, der den Prozessor mit dem Systemspeicher verbindet und z.B. mit AGP-Bussen (AGP: Accelerated Graphic Port) oder PCI-Bussen (PCI: Peripheral Component Interface). Die Southbridge ist gewöhnlich ein Chip in einem System-Core-Logic-Chipsatz zum Steuern des IDE-Busses (IDE: Integrated Drive Electronics) oder EIDE-Busses (EIDE: Enhanced IDE), des USB-Busses (USB: Universal Serial Bus), zum Bereitstellen einer Plug-and-Play-Unterstützung, zum Steuern einer PCI-ISA-Brücke (ISA: Industry Standard Architecture), zum Verwalten eines Tastatur/Maus-Controllers, zum Bereitstellen von Leistungsverwaltungsfeatures (Powermanagement) und zum Steuern anderer Peripheriegeräte. Für die Steuerung des USB-Busses enthalten Southbridges oft einen USB-Hostcontroller. Southbridges und Northbridges können auch in einem einzelnen Chip integriert sein.

[0003] In Computersystemen im Allgemeinen und in USB-Hostcontrollern im besonderen ist die Datenverwaltung eine wichtige Aufgabe, mit der umzugehen ist. Beispielsweise kann es Daten verschiedenen Typs geben, die behandelt werden müssen, und sogar bei Daten desselben Typs kann die Behandlung abhängig von einer aktuell durchzuführenden Funktion verschieden sein. Beispielsweise kann es Datenelemente geben, die nach einem periodischen Planungsschema aus dem Speicher gelesen oder in den Speicher geschrieben werden müssen, während andere Datenelemente in asynchroner Weise geholt oder geschrieben werden. Weiterhin kann es Daten geben, die sofort geholt werden müssen, wenn sie angefordert werden, während andere Daten im voraus geholt werden können (Prefetching).

[0004] Alle Daten, die geholt oder im voraus geholt werden oder die in den Speicher geschrieben werden müssen, werden üblicherweise in einer Art von Puffern gespeichert. Ein bekanntes Konzept solcher Puffer ist das der FIFO-Puffer (FIFO: First In First Out), die man sich als eine Sequenz von Registerelementen denken kann, die von einer Seite gefüllt werden und von der anderen Seite entleert werden. Solche Puffer können als eine Queue angesehen werden, da die Daten in Form einer Sequenz gespeichert werden.

[0005] Andere wohlbekannte Speicherobjekte sind doppelendige Queues, die durch mannigfaltige Funktionen manipuliert werden können, die es ermöglichen, Elemente entweder am Beginn der Queue, am Ende der Queue oder an irgendwelchen spezifischen Punkten innerhalb der Queue hinzuzufügen. Jedoch können solche Speicherobjekte noch den Nachteil aufweisen, dass sie gewöhnlicherweise nur für einen Datentyp und eine Art der Datenplanung geeignet sind. Das bedeutet, dass zwei separate Speichereinrichtungen bereitgestellt werden müssen, wenn es zu speichernde Datenelemente zweier Typen gibt.

[0006] Als ein Beispiel zeigt Fig. 1 den Fall, in dem Daten entweder periodisch oder asynchron zu speichern sind. Zu diesem Zweck gibt es zwei separate Speicherschaltungen **100** und **110**, die FIFO-Puffer oder eine andere Art von Queues sein können. Zwei separate Puffer, wie in Fig. 1 gezeigt, sind jedoch nachteilig, da jeder Puffer auf dem integrierten Schaltkreischip Hardwareschaltungen in signifikantem Umfang erfordert. Dies führt zu erhöhten Schaltkreisentwicklungskosten und Schaltkreisherstellungskosten und macht es weiterhin schwieriger, die Schaltkreise auf eine kleinere Größe herunterzuskalieren.

Stand der Technik

[0007] DE 197 40 738 A1 offenbart ein Verfahren zum Wiederanordnen einer Liste der in einer untergeordneten Managementebene eines Kommunikationsnetzes aktiven Alarme. Auftretende Alarme werden in entsprechenden Speichern abhängig von der Priorität der Alarme gespeichert. Die Speicher können als physikalisch getrennte Bereiche oder als fortlaufender Speicherbereich implementiert werden. Zeiger geben den ältesten Eintrag in jedem Speicher an. Andere Zeiger sind Fortsetzungsindices, die einen ältesten Alarm in einem Speicher für Alarme angeben, die andere Speicher betreffen.

[0008] US 5,043,981 A erwähnt FIFO-Puffer sowie ein Speicherkapazitätsmanagement, das einen „fast voll"-Schwellenwert verwendet.

Aufgabenstellung

ÜBERSICHT ÜBER DIE ERFINDUNG

[0009] Der Erfindung liegt die Aufgabe zugrunde, eine Vorrichtung und ein Verfahren mit einem Queue-

mechanismus bereitzustellen, die Daten verschiedenen Typs oder verschiedener Planungsschemata in effizienter Weise behandeln können.

[0010] Diese Aufgabe wird durch die Gegenstände der unabhängigen Patentansprüche gelöst.

[0011] Bevorzugte Ausgestaltungen sind in den abhängigen Ansprüchen definiert.

[0012] In einer Ausgestaltung wird eine Vorrichtung bereitgestellt, die eine Datenspeichereinrichtung zum Speichern einer Vielzahl von Datenelementen und eine Steuereinrichtung zum Steuern der Datenspeichereinrichtung umfasst. Die Datenspeichereinrichtung enthält eine Vielzahl von Registerelementen, wobei jedes Registerelement wenigstens ein Datenelement speichern kann. Die Vielzahl von Registerelementen ist eingerichtet, um eine Sequenz von Registerelementen zu bilden. Die Steuereinrichtung ist eingerichtet, um die Datenspeichereinrichtung zu steuern, um erste Daten in einem ersten Teil der Sequenz von Registerelementen und zweite Daten in einem zweiten Teil der Sequenz von Registerelementen zu speichern. Der erste Teil und der zweite Teil sind von variabler Länge, wobei die Summe der variablen Längen gleich der Länge der Sequenz von Registerelementen ist.

[0013] In einer anderen Ausgestaltung wird ein integrierter Southbridge-Schaltkreischip bereitgestellt. Der Chip umfasst eine Datenspeicherschaltung zum Speichern einer Vielzahl von Datenelementen. Die Datenspeicherschaltung enthält eine Vielzahl von Registerelementen, wobei jedes Registerelement wenigstens ein Datenelement speichern kann. Die Vielzahl von Registerelementen ist eingerichtet, um eine Sequenz von Registerelementen zu bilden. Der Chip umfasst weiterhin eine Steuerschaltung zum Steuern der Datenspeicherschaltung, um erste Daten in einem ersten Teil der Sequenz von Registerelementen zu speichern und zweite Daten in einem zweiten Teil der Sequenz von Registerelementen. Der erste Teil und der zweite Teil sind von variabler Länge, wobei die Summe der variablen Längen gleich der Länge der Sequenz von Registerelementen ist.

[0014] In einer weiteren Ausgestaltung kann ein Verfahren zum Betreiben einer Datenspeichereinrichtung bereitgestellt werden, um eine Vielzahl von Datenelementen zu speichern. Die Datenspeichereinrichtung enthält eine Vielzahl von Registerelementen. Jedes Registerelement kann wenigstens ein Datenelement speichern. Die Vielzahl von Registerelementen ist eingerichtet, um eine Sequenz von Registerelementen zu bilden. Das Verfahren umfasst das Steuern der Datenspeichereinrichtung, um erste Daten in einem ersten Teil der Sequenz von Registerelementen zu speichern, und das Steuern der Datenspeichereinrichtung, um zweite Daten in einem zweiten Teil der Sequenz von Registerelementen zu speichern. Der erste Teil und der zweite Teil sind von variabler Länge und die Summe der variablen Längen ist gleich der Länge der Sequenz von Registerelementen.

Ausführungsbeispiel

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0015] Die beigefügten Zeichnungen sind in die Beschreibung eingefügt und bilden einen Teil derselben zum Zwecke der Erläuterung der Prinzipien der Erfindung. Die Zeichnungen sind nicht als die Erfindung nur auf die verdeutlichten und beschriebenen Beispiele beschränkend zu verstehen, wie die Erfindung gemacht und verwendet werden kann. Weitere Merkmale und Vorteile werden aus der folgenden und genaueren Beschreibung der Erfindung ersichtlich werden, wie in den beigefügten Zeichnungen erläutert, in denen:

[0016] Fig. 1 ein herkömmliches Schema des Speicherns von Daten in periodischer sowie in asynchroner Weise verdeutlicht;

[0017] Fig. 2 eine Datenspeichereinrichtung gemäß einer ersten Ausgestaltung verdeutlicht;

[0018] Fig. 3 eine Datenspeichereinrichtung gemäß einer zweiten Ausgestaltung verdeutlicht;

[0019] Fig. 4 ein Flussdiagramm ist, das den Hauptprozess des Betriebs der Datenspeichereinrichtungen gemäß einer Ausgestaltung verdeutlicht, wie sie in den Fig. 2 oder Fig. 3 gezeigt sind;

[0020] Fig. 5 ein Flussdiagramm ist, das die Initialisierung verdeutlicht, die in dem Prozess von Fig. 4 durchgeführt wird;

[0021] Fig. 6 ein Flussdiagramm ist, das den Prozess des Ladens periodischer Deskriptoren in dem Prozess von Fig. 4 verdeutlicht;

[0022] Fig. 7 ein Flussdiagramm ist, das den Prozess des Herausladens (Unloading) periodischer Deskriptoren in dem Prozess von Fig. 4 verdeutlicht;

[0023] Fig. 8 ein Flussdiagramm ist, das den Prozess des Ladens asynchroner Deskriptoren in dem Prozess von Fig. 4 verdeutlicht;

[0024] Fig. 9 ein Flussdiagramm ist, das den Prozess des Herausladens asynchroner Deskriptoren in dem Prozess von Fig. 4 verdeutlicht;

[0025] Fig. 10 ein Flussdiagramm ist, das den Grenzenaktualisierungsprozess, der in dem Prozess

von Fig. 4 durchgeführt wird, verdeutlicht;

[0026] Fig. 11 ein Flussdiagramm ist, das einen Ladeprozess gemäß einer anderen Ausgestaltung verdeutlicht;

[0027] Fig. 12 ein Flussdiagramm ist, das den Herausladeprozess gemäß einer anderen Ausgestaltung verdeutlicht;

[0028] Fig. 13 die Hauptkomponenten eines USB-2.0-gemäßen Hostcontrollers gemäß einer Ausgestaltung verdeutlicht;

[0029] Fig. 14 ein Blockdiagramm ist, das die Komponenten des erweiterten Hostcontrollers verdeutlicht, der eine Komponente der Anordnung von Fig. 13 ist; und

[0030] Fig. 15 ein Blockdiagramm ist, das die Komponenten der DEQ-Einrichtung verdeutlicht, die in Fig. 14 gezeigt ist.

DETAILLIERTE BESCHREIBUNG DER ERFINDUNG

[0031] Die verdeutlichten Ausgestaltungen der vorliegenden Erfindung werden unter Bezugnahme auf die Zeichnungen beschrieben werden.

[0032] Wird nun auf die Zeichnungen und insbesondere auf Fig. 2 Bezug genommen, die eine Datenspeichereinrichtung gemäß einer Ausgestaltung verdeutlicht, so enthält die Datenspeichereinrichtung eine Vielzahl von Registerelementen **200**-**270**, die in Form einer Sequenz angeordnet sind. Das bedeutet, dass es ein Registerelement **200** gibt, das ein erstes Ende der Sequenz bildet, und ein anderes Registerelement **240**, das das andere Ende der Sequenz bildet. Jedes andere Registerelement **210**-**230**, **250**-**270** hat genau zwei Nachbarregisterelemente. Diese Organisation kann physikalisch realisiert sein, was bedeutet, dass die Registerelemente auf dem Chip Seite an Seite liegen, oder logisch realisiert sein, was bedeutet, dass die Registerelemente in sequenzieller Weise adressierbar sind.

[0033] Jedes Registerelement **200**-**270** dient der Speicherung eines Datenelements. In einer anderen Ausgestaltung kann ein Registerelement mehr als ein Datenelement speichern. Ein Datenelement kann ein Transaktionselement sein.

[0034] In der Ausgestaltung von Fig. 2 werden Eingabedaten aus einem Speicherholvorgang bereitgestellt und Ausgabedaten an die USB-Transaktionsaufbaueinrichtung geliefert. Es gibt periodische und asynchrone Transaktionen und die periodischen Transaktionen werden von rechts nach links eingefüllt, während die asynchronen Transaktionen von links nach rechts eingefüllt werden. Die mittlere Linie stellt eine Grenze dar, die die Datenspeichereinrichtung in einen periodischen Teil und einen asynchronen Teil trennt. Die Grenze kann geändert werden, so dass die Länge des periodischen Teils sowie die Länge des asynchronen Teils variabel sind. Die Summe beider variabler Längen ist gleich der Länge der gesamten Sequenz.

[0035] Wie aus Fig. 2 ersichtlich ist, kann die Datenspeichereinrichtung als ein Doppelenden-Queuemechanismus mit zwei miteinander verbundenen Queues betrachtet werden. Somit wird ein Mechanismus bereitgestellt, der verwendet werden kann, um Daten verschiedenen Typs zu speichern oder Daten, die entweder einer periodischen oder einer asynchronen Planung unterliegen. Dies ermöglicht die gemeinsame Verwendung von Pufferraum und dessen Optimierung auf Grundlage der aktuellen Arbeitsbelastung. Darüber hinaus macht die Grenzenjustierung einen geschichtsbasierten Adaptionsmechanismus möglich und kann Fläche auf dem Chip einsparen.

[0036] Ausgehend von dem logischen Erscheinungsbild, wie es in Fig. 2 gezeigt ist, geht die Funktionalität der doppelendigen Queue über die einfacher FIFO-Puffer darin hinaus, dass sie es gestattet, im voraus geholte asynchrone Deskriptoren zu halten und dennoch in effizienter Weise jeglichen freien Raum zu verwenden, um periodische Deskriptoren im voraus zu holen.

[0037] Wird nun zu Fig. 3 übergegangen, die eine andere Ausgestaltung verdeutlicht, so wird die Steuerung des doppelendigen Queuemechanismus durch ein spezielles Indexregister **300** bewerkstelligt. Es gibt einen Indexregisterplatz für jeden Queueeintrag. Der Inhalt eines Indexregisterplatzes (bezeichnet durch die Zahlen 0 ... N-1) ist eine Basisadresse für einen Deskriptor innerhalb eines RAM (Random Access Memory, wahlfreier Zugriffsspeicher), bezeichnet mit den Buchstaben A ... D und R ... T. Der Kopf der periodischen Queue ist am Platz 0 des Indexregisters **300** gespeichert und der Kopf der asynchronen Queue liegt im Platz N-1. Wie in Fig. 3 gezeigt, würde der nächste in den RAM geladene periodische Deskriptor beginnend mit Adresse C gespeichert werden und der nächste asynchrone Deskriptor an der Adresse S. Somit können die Datenelemente, die in den Registerplätzen des Indexregisters **300** gespeichert werden, Zeiger (Pointer) auf einen RAM sein.

[0038] Die doppelendige Queue von Fig. 3 kann in zwei verschiedenen Moden betrieben werden: im Push-Margin-Modus (Randschiebemodus) und im Load-Limit-Modus (Schrankenlademodus). Im Load-Limit-Modus kann jede Seite wachsen, bis sie ihren maximalen Ladewert **320**, **330** oder das Ende

der anderen Seite erreicht. Somit können Queueeinträge zwischen MaxALoad **320** und MaxPLoad **330** für beide Typen verwendet werden. Das bedeutet, dass es im Load-Limit-Modus einen Überlapp der zwei Teile geben kann. Im Push-Margin-Modus ist es jeder Seite gestattet, nur so lange zu wachsen, bis sie auf die Grenze **310** trifft. Diese Treffer (Hits) während eines Mikrorahmens werden erfasst und verwendet, um die Grenze **310** innerhalb der durch MaxALoad **320** und MaxPLoad **330** gegebenen Schranken zu aktualisieren. Somit kann die Grenze **310** in diesem Fall auf planungsgeschichtlicher Grundlage angepasst werden.

[0039] Im folgenden wird der Betrieb der in den Fig. 2 und Fig. 3 gezeigten Datenspeichereinrichtung in weiteren Einzelheiten diskutiert werden, wobei von den in Fig. 3 gezeigten Parametern Verwendung gemacht wird: der Grenze **310**, den maximalen Ladewerten MaxALoad **320** und MaxPLoad **330** und den aktuellen Ende-Zeigern P_Load **340** und A_Load **350**.

[0040] Wird zunächst zu dem Flussdiagramm von Fig. 4 übergegangen, so beginnt der Hauptprozess des Betriebs der Datenspeichereinrichtung mit einer Initialisierung in Schritt **400**. Ist die doppelendige Queue in Schritt **400** initialisiert worden, werden periodische und asynchrone Deskriptoren in den Schritten **410** bis **440** geladen und herausgeladen. Obwohl das Flussdiagramm von Fig. 4 eine bestimmte Abfolge der Schritte **410** bis **440** angibt, ist anzumerken, dass andere Abfolgen auch möglich sein können. Einige oder alle der Lade- und Herausladeschritte für jede Art von Deskriptoren können in einer anderen Ausgestaltung auch ineinandergreifend sein. Wie aus Fig. 4 ersichtlich ist, wird der Prozess in iterativer Weise durchgeführt, wobei jede Iteration einen Grenzenaktualisierungsschritt **450** enthält.

[0041] Wird zunächst der Initialisierungsprozess diskutiert, der in Schritt **400** des in Fig. 4 gezeigten Prozesses durchgeführt wird, so wird auf Fig. 5 Bezug genommen. Zunächst wird das Indexregister **300** mit Anfangswerten in Schritt **500** gefüllt. In einer Ausgestaltung können diese Werte einfach inkrementierte ganze Zahlen sein. Dann werden die Ende-Zeiger **340**, **350** gesetzt, um auf die Köpfe beider Teile zu zeigen (Schritt **510**). Das bedeutet, dass P_Load **340** anfänglich gesetzt wird, um auf Platz 0 des Indexregisters **300** zu zeigen, während A_Load **350** gesetzt wird, um auf Platz N-1 zu zeigen.

[0042] Es wird dann in Schritt **520** bestimmt, ob der Push-Margin-Modus oder der Load-Limit-Modus eingestellt ist. Wenn der Push-Margin-Modus eingestellt ist, werden die Grenzenwerte für periodische und asynchrone Deskriptoren in Schritt **550** auf denselben Anfangswert eingestellt. Wird die doppelendige Queue jedoch im Load-Limit-Modus betrieben, so wird der Grenzenwert für periodische Deskriptoren auf die minimale asynchrone Beladung in Schritt **530** eingestellt, während der Grenzenwert für asynchrone Deskriptoren auf das Maximum der asynchronen Beladung eingestellt wird (Schritt **540**). Dann werden in den Schritten **560** und **570** Hitflags für sowohl periodische als auch asynchrone Deskriptoren heruntergesetzt. Wie unten in weiteren Einzelheiten beschrieben werden wird, sind Hitflags Anzeigemittel, die bei der Aktualisierung der Grenzen verwendet werden.

[0043] Wird nun zu Fig. 6 übergegangen, so wird der Prozess des Ladens periodischer Deskriptoren in weiteren Einzelheiten erläutert. Zunächst wird in Schritt **600** bestimmt, ob der periodische Ende-Zeiger **340** auf einen Registerplatz hinter dem Grenzenwert für periodische Deskriptoren zeigt. Wie aus der obigen Diskussion des Initialisierungsprozesses ersehen werden kann, ist der Grenzenwert eine Variable, die sowohl im Push-Margin-Modus als im Load-Limit-Modus verwendet wird, und somit können die Grenzenwerte für periodische und asynchrone Deskriptoren verschieden oder auch gleich sein. Wenn in Schritt **600** festgestellt wird, dass das periodische Ende die Grenze übersteigt, so wird in Schritt **610** das entsprechende Hitflag gesetzt. Andernfalls wird in Schritt **620** überprüft, ob der periodische Ende-Zeiger auf einen Registerplatz hinter dem zeigt, worauf der asynchrone Ende-Zeiger zeigt. Ist dies der Fall, so wird in Schritt **630** ein Bounce-Flag gesetzt. Andernfalls fährt der Prozess mit Schritt **640** des Lesens eines Registereintrags am Ende der periodischen Seite fort, d.h. am Registerelement, auf das P_Load **340** zeigt. Schließlich wird in Schritt **650** der periodische Ende-Zeiger P_Load **340** inkrementiert.

[0044] Das Herausladen periodischer Deskriptoren wird in weiteren Einzelheiten in Fig. 7 verdeutlicht. Zunächst wird der Deskriptor, der aktuell am Kopf der periodischen Seite gespeichert wird, in einem temporären Puffer in Schritt **700** zur späteren Verwendung gepuffert. Dann wird für jeden Eintrag der periodischen Seite der Inhalt des nächsten Eintrags in Schritt **710** kopiert. Somit ist das, was in Schritt **710** tatsächlich getan wird, eine Rotation nach links. Der periodische Ende-Zeiger P_Load **340** wird dann in Schritt **720** dekrementiert und der Deskriptor, der in Schritt **700** gepuffert wurde, wird in Schritt **730** an das Ende geschrieben.

[0045] Während die Fig. 6 und Fig. 7 Flussdiagramme zur Verdeutlichung des Ladeprozesses und Herausladeprozesses periodischer Deskriptoren zeigen, sind entsprechende Flussdiagramme zur Verarbeitung der asynchronen Deskriptoren in den Fig. 8 und Fig. 9 gezeigt. Diese Flussdiagramme sind im wesentlichen dieselben wie die oben diskutierten, jedoch sollte erwähnt werden, dass einige Modifikationen infolge des Umstandes anfallen, dass die asynchrone Queue in entgegengesetzter Richtung, vergli-

chen mit der periodischen Queue, gefüllt wird. Somit müssen dort, wo die Zeiger der periodischen Queue dekrementiert worden waren, die entsprechenden Zeiger der asynchronen Seite inkrementiert werden. Wo in Schritt **710** der Inhalt des nächsten Eintrags kopiert wurde, um eine Rotation nach links durchzuführen, muss auf der asynchronen Seite der Inhalt des vorhergehenden Eintrags kopiert werden, wodurch die Daten nach rechts rotieren.

[0046] Wird nun zu Fig. 10 übergegangen, so wird eine Ausgestaltung des Grenzenaktualisierungsprozesses gezeigt, der im Schritt **450** des Flussdiagramms von Fig. 4 durchzuführen ist. Im Schritt **1000** wird der Betriebsmodus festgestellt. Wenn die doppelendige Queue im Load-Limit-Modus betrieben wird, können die Hitflags in Schritt **1090** direkt heruntergesetzt werden. Andernfalls, d.h. wenn die Datenspeichereinrichtung im Push-Margin-Modus betrieben wird, kann die weitere Verarbeitung davon abhängen, ob eines der periodischen und asynchronen Hitflags gesetzt ist.

[0047] Wenn in Schritt **1010** festgestellt wird, dass das periodische Hitflag gesetzt ist, wird eine Überprüfung durchgeführt, ob die aktuelle Grenze die minimale asynchrone Beladung erreicht (Schritt **1030**). Wenn das periodische Hitflag gesetzt ist und die aktuelle Grenze das asynchrone Beladungsminimum erreicht, werden die Grenzenwerte im Schritt **1040** und ebenfalls die aktuelle Grenze im Schritt **1050** inkrementiert. Somit wird die Grenze auf einen höheren Wert aktualisiert, da es noch Raum zum Erhöhen der Grenzenwerte gab, ohne die minimale asynchrone Beladung zu überschreiten.

[0048] In den Schritten **1020**, **1060**, **1070** und **1080** wird der entsprechende Prozess für den Fall durchgeführt, dass das asynchrone Hitflag gesetzt ist. Sind die Grenzenwerte einmal aktualisiert, können die Hitflags im Schritt **1090** heruntergesetzt werden.

[0049] Während die Fig. 4 bis Fig. 10 eine Ausgestaltung des Betriebs der doppelendigen Queue verdeutlichen, wird eine andere Ausgestaltung nun unter Bezugnahme auf die Fig. 11 und Fig. 12 diskutiert werden. In dieser Ausgestaltung kann die periodische FIFO-Beladung die Mitte der Queue überkreuzen und auf die asynchrone Queue treffen, während die asynchrone FIFO-Beladung die Mitte nicht überkreuzen darf (vgl. Fig. 2). Weiterhin wird für alle wartenden asynchronen Transaktionen, die in der Queue gespeichert sind, über eine geschätzte Routenzeit (ETE; "estimated time enroute") Buch geführt.

[0050] Wird nun auf das Flussdiagramm von Fig. 11 Bezug genommen, so ist der Ladeprozess zum Laden periodischer und asynchroner Deskriptoren gezeigt. In Schritt **1100** wird festgestellt, ob die periodische Liste leer ist. Ist dies der Fall, so fährt der Prozess mit Schritt **1140** fort. Andernfalls friert der Prozess ein, bis die periodische Beladung unter dem asynchronen Kopf liegt (Schritt **1110**). Dann wird im Schritt **1120** ein periodischer Deskriptor geholt und in Schritt **1130** das periodische Ende der doppelendigen Queue gefüllt. Der Prozess kehrt dann zu Schritt **1100** zurück.

[0051] Wenn in Schritt **1100** festgestellt wird, dass die periodische Liste leer ist, wird in Schritt **1140** überprüft, ob die gesamte geschätzte Zeit der asynchronen Beladung unterhalb der in dem Mikrorahmen übrigen Echtzeit liegt. Ist dies nicht der Fall, so kehrt der Prozess zu Schritt **1100** zurück. Andernfalls wird in Schritt **1150** überprüft, ob die asynchrone Beladung unter der Mitte der Queue liegt. Ist dies der Fall, so wird in Schritt **1160** ein asynchroner Deskriptor geholt und das asynchrone Ende der doppelendigen Queue in Schritt **1170** gefüllt, gefolgt von Schritt **1180** des Addierens der geschätzten Zeit (ETE) dieser Transaktion zu dem Zählwert der gesamten ETE. Der Prozess kehrt dann zu Schritt **1140** zurück.

[0052] Wird nun auf Fig. 12 Bezug genommen, so beginnt der Herausladeprozess der vorliegenden Ausgestaltung mit Schritt **1200** des Wartens auf den Mikrorahmenbeginn. Es wird dann in Schritt **1210** überprüft, ob die periodische Seite leer ist, und wenn dies nicht der Fall ist, wird in den Schritten **1220** und **1230** ein periodischer Deskriptor herausgeladen und die entsprechende Transaktion ausgeführt. Ist die periodische Seite leer, so wird überprüft, ob die verbleibende Echtzeit die geschätzte Zeit (ETE) des asynchronen Deskriptors übersteigt (Schritt **1240**). Ist dies der Fall, so wird in den Schritten **1250** und **1260** ein asynchroner Deskriptor herausgeladen und die entsprechende Transaktion ausgeführt. Dann wird die geschätzte Zeit von der gesamten ETE subtrahiert (Schritt **1270**). Wenn jedoch festgestellt wird, dass keine ausreichende Echtzeit übrig ist, so kehrt der Prozess zu Schritt **1200** zurück, um auf den nächsten Mikrorahmenbeginn zu warten.

[0053] In einer Ausgestaltung können die oben beschriebenen Prozesse von solcher Art sein, dass keine periodischen Transaktionen jenseits der 80%-Grenze eingeplant werden. Auch können die Prozesse eine adäquate Behandlung für Transmissionsfehler enthalten, die zu einer Situation führen können, in der nicht alle periodischen Transaktionen innerhalb der 80%-Grenze gesendet werden. Eine mögliche Lösung wäre die Detektion solcher Fälle und die Durchführung eines Flushs der periodischen Seite. Das bedeutet, dass die doppelendige Queue gesteuert werden kann, um den Füllzustand einer oder beider Teile der doppelendigen Queue unterhalb eines vorbestimmten Prozentwerts der variablen Länge des jeweiligen Teils zu halten. In einer anderen Ausgestaltung kann die doppelendige Queue gesteuert werden, um die variablen Längen beider Teile un-

terhalb eines vorbestimmten Prozentwerts der Länge der gesamten doppelendigen Queue zu halten.

[0054] Der oben in Verbindung mit den Fig. 11 und Fig. 12 diskutierte Prozess kann den Vorteil aufweisen, dass niemals bereits geholte asynchrone Transaktionen verworfen werden, wenn es noch genügend verbleibende Zeit in dem Mikrorahmen gibt. Stattdessen werden sie zu dem nächsten Mikrorahmen verschoben.

[0055] Wird nun zu Fig. 13 übergegangen, so ist ein USB-2.0-gemäßer Hostcontroller **1300** gemäß einer Ausgestaltung gezeigt. Wie aus der Figur ersichtlich ist, gibt es einen Hostcontroller **1320**, der ein erweiterter Hostcontroller (EHC: Enhanced Host Controller) für die USB-2.0-Hochgeschwindigkeitsfunktionalität ist. Dieser Hostcontroller arbeitet in Übereinstimmung mit der EHCI-Spezifikation (EHCI: Enhanced Host Controller Interface) für USB 2.0. Auf der Software-Seite ist dem Hostcontroller **1320** ein spezifischer Hostcontrollertreiber (EHCD) zugeordnet.

[0056] Ferner gibt es Hostcontroller **1310** für Vollgeschwindigkeits- und Niedergeschwindigkeitsvorgänge. UHCI (Universal Host Controller Interface) oder OHCI (Open Host Controller Interface) sind zwei Industriestandards, die in dem universellen oder offenen Hostcontroller (UHC/OHC) **1310** zum Bereitstellen von USB-1.1-Hostcontrollerschnittstellen angewendet werden. Den Hostcontrollern **1310** sind universelle/offene Hostcontrollergeräte (UHCD/OHCD) auf der untersten Softwareebene zugewiesen.

[0057] In der Begleithostcontrollereinheit **1310** der vorliegenden Ausgestaltung gibt es zwei OHCI-gemäße Hostcontroller, OHC0 **1330** und OHC1 **1340**. Diese Controller wickeln den gesamten USB-1.1-gemäßen Verkehr ab und können die Legacy-Tastaturemulation für Umgebungen enthalten, die nicht USB-gemäß sind.

[0058] Ein Port-Router **1350** kann bereitgestellt werden, um die physikalischen Portschnittstellen ihren jeweiligen Eignern zuzuweisen. Die Eignerschaft wird von EHC-Registern gesteuert und standardmäßig werden alle Ports auf die Begleithostcontroller geroutet, um zu ermöglichen, dass ein System mit nur USB-1.1-gemäßen Treibern funktioniert. Wenn ein USB-2.0-gemäßer Treiber in dem System vorhanden ist, wird er die Ports entweder einem Begleithostcontroller **1330**, **1340** für Niedergeschwindigkeits- und Vollgeschwindigkeitsgeräte und -hubs (USB-1.1-Verkehr) oder dem EHC **1320** für Hochgeschwindigkeitsgeräte und -hubs zuweisen.

[0059] Das bedeutet, dass der in Fig. 13 gezeigte USB-2.0-Hostcontroller die EHCI-Spezifikation erfüllt und die Verwendung bestehender OHCI-USB-1.1-Hostcontroller mit der minimal notwendigen Abänderung ermöglicht, um eine Schnittstelle zu dem Port-Router-Block **1350** auszubilden, anstelle von physikalischen USB-1.1-Geräten.

[0060] Der USB-2.0-gemäße Hostcontroller von Fig. 13 kann als Hardwarearchitektur definiert sein, um einen EHCI-gemäßen Hostcontroller zur Integration in eine Southbridge zu implementieren. Der Hostcontroller sitzt dann zwischen den analogen USB-2-Eingabe/Ausgabe-Pins und einem Verbindungsschnittstellenmodul zum Ausbilden einer Schnittstelle in Upstream-Richtung zum Systemspeicher hin, z.B. zu einer Northbridge, wenn eine in dem System vorhanden ist. Diese Schnittstelle kann eine interne HyperTransport™- Schnittstelle sein. Die HyperTransport-Technologie ist eine hochperformante Hochgeschwindigkeits-Punkt-zu-Punkt-Verbindung, um integrierte Schaltungen auf einem Motherboard miteinander zu verbinden. Sie kann signifikant schneller sein als ein PCI-Bus bei einer äquivalenten Anzahl von Pins. Die HyperTransport-Technologie ist entwickelt worden, um signifikant mehr Bandbreite bereitzustellen als gegenwärtige Technologien, um Low-Latency-Anworten zu verwenden, um eine niedrige Pinzahl bereitzustellen, um kompatibel mit Legacy-PC-Bussen zu sein, um auf neue Systemnetzwerkarchitekturbusse erweiterbar zu sein, um für Betriebssysteme transparent zu sein und um geringe Auswirkungen auf Peripherietreiber zu haben.

[0061] Somit wird in der Ausgestaltung von Fig. 13 ein HyperTransport-basierter USB-Hostcontroller bereitgestellt, in dem ein erweiterter Hostcontroller **1320** für die Bearbeitung des gesamten USB-Hochgeschwindigkeitsverkehrs verantwortlich ist sowie für die Steuerung der Eignerschaft für sich selbst und die Begleitcontroller **1310** über den Port-Router **1350**. Nach einem Rücksetzen beim Einschalten oder einem softwaregesteuerten Rücksetzen des EHC **1320** kann er defaultmäßig einen Zustand aufweisen, in dem alle Ports von den Begleithostcontrollern **1310** besessen und gesteuert werden, alle Betriebsregister ihre jeweiligen Defaultwerte aufweisen und der EHC **1320** angehalten ist, d.h. er weder aus dem Systemspeicher Deskriptoren holt noch irgendeine USB-Aktivität herausgibt. Im Normalbetrieb kann der EHC **1320** isochrone und Interrupt-Transfers von einer periodischen Liste verarbeiten sowie den Großteil ("bulk") und die Steuerung von einer asynchronen Liste.

[0062] Jede Liste kann leer oder ihre Verarbeitung softwaregesteuert außer Kraft gesetzt sein.

[0063] Wird nun zu Fig. 14 übergegangen, so sind die Komponenten des erweiterten Hostcontrollers EHC **1320** in weiteren Einzelheiten gezeigt. Die Abwicklung des Datenverkehrs zu und von dem Systemspeicher wird durch den Stub **1400** bewerkstelligt. Der Stub **1400** weist die internen Quellen und

Senken den jeweiligen HyperTransport-Strömen zu, d.h. Posted Requests, Non-Posted-Requests, Responses (Antworten). Der Stub **1400** arbitriert die interne HyperTransport-Schnittstelle zwischen allen internen Busmastern, d.h. der DMA-Empfangseinrichtung (DMA: Direct Memory Access) **1410**, dem Deskriptorcache **1425**, der Deskriptorverarbeitungseinrichtung **1430** und der DMA-Sendeeinrichtung **1450**. Somit arbitriert der Stub **1400** zwischen dem Holen eines Deskriptors, dem Zurückschreiben von Deskriptoren, dem Empfangen von Daten und dem Senden von Daten.

[0064] Der Stub **1400** ist mit einem Registerfile **1405** verbunden, der die EHCI-Register enthält. In der vorliegenden Ausgestaltung speichern die EHCI-Register Daten bezüglich der PCI-Konfiguration, den Hostcontroller-Fähigkeiten und den Hostcontroller-Betriebsmoden.

[0065] Die Deskriptorverarbeitungseinrichtung **1430** ist mit dem Stub **1400** verbunden und besteht aus drei Untereinheiten: der Deskriptorholeinrichtung (DescrFetch) **1435**, der doppelendigen Queue, die als Datenspeichereinrichtung agiert (DEQ) **1445**, und der Transaktionsvervollständigungsmaschine (TACM) **1440**. Die Deskriptorholeinrichtung **1435** bestimmt auf Grundlage von Timinginformationen und Registereinstellungen, welcher Deskriptor als nächster zu holen oder im voraus zu holen ist, und sendet die Anforderung an den Stub **1400** und/oder an den Deskriptorcache **1425**. Wenn sie den Deskriptor empfängt, sendet sie ihn an die DEQ-Einrichtung **1445**.

[0066] Die DEQ-Einrichtung **1445** hält die im voraus geholten Deskriptoren. Indem sie ein Speichermanagement durchführt, besteht ihre Hauptfunktion darin, eine Speicherkapazität bereitzustellen, um Speicherzugriffslegacies für Deskriptorholvorgänge zu mitteln.

[0067] Die Transaktionsvervollständigungsmaschine **1440** ist mit der Deskriptorholeinrichtung **1435** verbunden, um die Statusrückschreibung zu Deskriptoren zu verwalten. Zu diesem Zwecke ist die Transaktionsvervollständigungsmaschine **1440** mit dem Deskriptorcache **1425** verbunden.

[0068] Dieser Cache enthält Deskriptoren, die von der Deskriptorholeinrichtung **1435** für schnellen wiederholten Zugriff im voraus geholt worden sind. Die in dem Deskriptorcache **1425** gehaltenen Deskriptoren werden von der Transaktionsvervollständigungsmaschine **1440** aktualisiert und eventuell über den Stub **1400** in den Systemspeicher zurückgeschrieben. Der Deskriptorcache **1425** kann voll assoziativ mit Write-Through-Eigenschaften sein. Er kann ferner die Ersetzung der Inhalte jedes Mikrorahmens steuern.

[0069] Wie aus Fig. 14 ersichtlich ist, werden ferner eine DMA-Sendeeinrichtung **1450** und eine DMA-Empfangseinrichtung **1410** bereitgestellt. Die DMA-Sendeeinrichtung **1450** besteht aus einer Datenholeinrichtung (DataFetch) **1455** und einem Datensendepuffer (TxBuf) **1460**. Die Datenholeinrichtung **1455** ist ein DMA-Lesebusmaster und überprüft die Einträge in der DEQ-Einrichtung **1445** der Deskriptor-Verarbeitungseinrichtung **1430**. Die Datenholeinrichtung **1455** holt die entsprechenden Daten im voraus und leitet sie an den Datensendepuffer **1460** weiter.

[0070] Der Datensendepuffer **1460** kann ein FIFO-Puffer sein und seine Funktion entspricht der der DEQ-Einrichtung **1445** darin, dass er es ermöglicht, im voraus genügend Daten für herausgehende Transaktionen zu holen, um die Speichersystemlatency abzudecken.

[0071] Die DMA-Empfangseinrichtung **1410** besteht aus der Datenschreibeinrichtung (DataWrite) **1415**, die als DMA-Schreibbusmastereinrichtung dient, um die empfangenen Daten, die in dem Datenempfangspuffer (RxBuf) **1420** gespeichert sind, an ihre jeweilige Stelle im Systemspeicher zu schieben. Der Datenempfangspuffer **1420** kann ein einfacher FIFO-Puffer sein.

[0072] Weiterhin wird eine Rahmentimingeinrichtung (FrameTiming) **1465** bereitgestellt, die die USB-Masterzeitreferenz ist. Ein Taktpuls der Rahmentimingeinrichtung entspricht einem ganzzahligen (z.B. 8 oder 16) Vielfachen der Hochgeschwindigkeits-USB-Bitzeiten. Die Rahmentimingeinrichtung **1465** ist mit der DEQ-Einrichtung **1445** und dem Paketbearbeitungsblock **1470** verbunden.

[0073] Der Paketbearbeitungsblock **1470** besteht aus einer Paketaufbaueinrichtung (PktBuild) **1485**, die die notwendigen USB-Busvorgänge konstruiert, um Daten und Handshakes zu senden, und aus einem Paketdekodierer (PktDecode) **1475**, der empfangene USB-Pakete zerlegt. Weiterhin wird eine Transaktionssteuereinrichtung (TaCtrl) **1480** bereitgestellt, die die Paketaufbaueinrichtung **1485** und den Paketdekodierer **1475** überwacht. Weiterhin umfasst die Paketbearbeitungseinrichtung **1470** eine CRC-Einrichtung (CRC: Cyclic Redundancy Check) **1490** zum Erzeugen und Überprüfen von CRC-Daten für gesendete und empfangene Daten.

[0074] Die Paketaufbaueinrichtung **1485** und der Paketdekodierer **1475** der Paketbearbeitungseinrichtung **1470** sind mit dem Root-Hub **1495** verbunden, der portspezifische Steuerregister, eine Verbindungsdetektionslogik und eine Streu/Einsammelfunktionalität für Pakete zwischen der Paketbearbeitungseinrichtung **1470** und dem Port-Router enthält.

[0075] Wird nun zu Fig. 15 übergegangen, so ist die in Fig. 14 gezeigte DEQ-Einrichtung **1445** in weiteren Einzelheiten gezeigt. Die gesamte doppelendige Queue wird von der Steuereinrichtung QCTRL **1500** gesteuert, die auch mit der Deskriptorholeinrichtung **1435**, der Transaktionsvervollständigungsmaschine **1440** und der DMA-Sendeeinrichtung **1450** interagiert. Die Deskriptoren sind in dem Deskriptor-RAM (DESCRRAM) **1520** gespeichert. Die Datenspeichereinrichtung von Fig. 15 verfolgt auch die geschätzte Routenzeit (ETE) für die gespeicherten Deskriptoren. Die ETE-Werte werden in dem ETE-RAM **1510** parallel zu ihren Gegenparts in dem Deskriptor-RAM **1520** gespeichert. Der ETE-Akkumulator (ETE-ACCU)**1530** speichert einen Wert, der die geschätzte Zeit angibt, um alle gegenwärtig in der Queue befindlichen Deskriptoren zu vervollständigen. Der Akkumulator wird um die geschätzte Zeit für jeden neu gespeicherten Deskriptor inkrementiert und um den ETE-Wert des Deskriptors dekrementiert, der als letzter entfernt wurde. Es kann weitere Einheiten **1540** und **1550** geben, um eine Schnittstelle zu anderen Einheiten des erweiterten Hostcontrollers zu bilden.

[0076] Während die Erfindung in Bezug auf die physikalischen Ausgestaltungen, die in Übereinstimmung damit konstruiert worden sind, beschrieben worden ist, wird Fachleuten ersichtlich sein, dass verschiedene Modifikationen, Variationen und Verbesserungen der vorliegenden Erfindung im Lichte der obigen Lehren und innerhalb des Umfangs der beigefügten Ansprüche gemacht werden können, ohne von der Idee und dem beabsichtigten Umfang der Erfindung abzuweichen. Zusätzlich sind solche Bereiche, in denen davon ausgegangen wird, dass sich Fachleute auskennen, hier nicht beschrieben worden, um die beschriebene Erfindung nicht unnötig zu verschleiern. Demgemäß ist zu verstehen, dass die Erfindung nicht durch die spezifisch verdeutlichten Ausgestaltungen, sondern nur durch den Umfang der beigefügten Ansprüche beschränkt wird.

Patentansprüche

1. Vorrichtung mit einem Queuemechanismus, umfassend:
eine Datenspeichereinrichtung (**200**-**270**, **300**, **1445**, **1520**) zum Speichern einer Vielzahl von Datenelementen, wobei die Datenspeichereinrichtung eine Vielzahl von Registerelementen (**200**-**270**) enthält, die jeweils wenigstens ein Datenelement speichern können, wobei die Vielzahl von Registerelementen eingerichtet ist, um eine Sequenz (**300**) von Registerelementen zu bilden; und
eine Steuereinrichtung (**1500**) zum Steuern der Datenspeichereinrichtung, um erste Daten in einem ersten Teil (**200**-**230**) der Sequenz von Registerelementen und zweite Daten in einem zweiten Teil (**240**-**270**) der Sequenz von Registerelementen zu speichern, wobei der erste Teil und der zweite Teil von variablen Längen sind und die Summe der variablen Längen gleich der Länge der Sequenz von Registerelementen ist,
wobei die Steuereinrichtung eingerichtet ist zum Steuern sowohl des ersten als auch des zweiten Teils der Sequenz von Registerelementen, um die jeweiligen Daten FIFO-mäßig zu speichern, und
wobei der erste und zweite Teil der Sequenz von Registerelementen jeweils ein inneres und ein äußeres Ende aufweisen und die jeweiligen inneren Enden (**230**, **270**) innerhalb der Sequenz von Registerelementen und die jeweiligen äußeren Enden (**200**, **240**) am Anfang oder Ende der Sequenz von Registerelementen liegen, wobei die Steuereinrichtung eingerichtet ist zum Steuern sowohl des ersten als auch des zweiten Teils der Sequenz von Registerelementen, um hereinkommende Daten in den entsprechenden Teil am jeweiligen inneren Ende einzufüllen.

2. Vorrichtung nach Anspruch 1, wobei die ersten Daten periodische Daten und die zweiten Daten asynchrone Daten sind und die Steuereinrichtung eingerichtet ist zum Steuern des ersten Teils in periodischer Weise und des zweiten Teils in asynchroner Weise.

3. Vorrichtung nach Anspruch 2, wobei die ersten Daten und die zweiten Daten Datenelemente desselben Typs enthalten und die Datenelemente desselben Typs Transaktionselemente sind.

4. Vorrichtung nach Anspruch 3, wobei die Steuereinrichtung eingerichtet ist zum Steuern der Datenspeichereinrichtung abhängig von einer geschätzten Dauer des Transfers, der von den Transaktionselementen dargestellt wird.

5. Vorrichtung nach Anspruch 1, wobei die Datenelemente, die in den Registerelementen gespeichert sind, Zeiger auf entsprechende Stellen in einem wahlfreien Zugriffsspeicher sind.

6. Vorrichtung nach Anspruch 1, wobei die Steuereinrichtung eingerichtet ist zum Holen der ersten und zweiten Daten aus einem Speicher.

7. Vorrichtung nach Anspruch 1, wobei der erste und zweite Teil der Sequenz keinen Überlapp aufweisen.

8. Vorrichtung nach Anspruch 1, wobei der erste und zweite Teil der Sequenz einen Überlapp von wenigstens einem Registerelement aufweisen.

9. Vorrichtung nach Anspruch 1, weiterhin umfassend:
ein erstes Adressenregister (**310**, **330**), das ein erstes Registerelement identifiziert, das eine Grenze des ersten Teils der Sequenz von Registerelementen

bildet, wobei sich der erste Teil von einer Seite (200) der Sequenz zu dem identifizierten ersten Registerelement erstreckt; und
ein zweites Adressenregister (**310**, **320**), das ein zweites Registerelement identifiziert, das eine Grenze des zweiten Teils der Sequenz von Registerelementen bildet, wobei sich der zweite Teil von der anderen Seite (**240**) der Sequenz zu dem identifizierten zweiten Registerelement erstreckt.

10. Vorrichtung nach Anspruch 9, wobei die Steuereinrichtung in einem Modus betreibbar ist, in dem das erste Adressenregister (**310**) gleich dem zweiten Adressenregister (**310**) ist.

11. Vorrichtung nach Anspruch 9, wobei die Steuereinrichtung in einem Modus betreibbar ist, in dem das identifizierte erste Registerelement gleich dem identifizierten zweiten Registerelement ist.

12. Vorrichtung nach Anspruch 9, wobei die Steuereinrichtung in einem Modus betreibbar ist, in dem das erste Adressenregister (**330**) von dem zweiten Adressenregister (**320**) verschieden ist und sich das identifizierte erste Registerelement von dem identifizierten zweiten Registerelement unterscheidet.

13. Vorrichtung nach Anspruch 9, wobei die Steuereinrichtung imstande ist, den Betrieb zwischen einem ersten und einem zweiten Modus umzuschalten, wobei in dem ersten Modus das erste Adressenregister (**310**) gleich dem zweiten Adressenregister (**310**) ist und in dem zweiten Modus das erste Adressenregister (**330**) von dem zweiten Adressenregister (**320**) verschieden ist.

14. Vorrichtung nach Anspruch 9, weiterhin umfassend:
ein drittes Adressenregister (**340**), das ein drittes Registerelement identifiziert, das den Füllstand des ersten Teils der Sequenz von Registerelementen angibt; und
ein viertes Adressenregister (**350**), das ein viertes Registerelement identifiziert, das den Füllstand des zweiten Teils der Sequenz von Registerelementen angibt.

15. Vorrichtung nach Anspruch 14, wobei:
die Steuereinrichtung eingerichtet ist zum Setzen eines ersten oder zweiten Hitflags, wenn sie feststellt, dass der Füllstand des ersten oder zweiten Teils der Sequenz von Registerelementen jenseits der jeweiligen Grenze liegt; und
die Steuereinrichtung eingerichtet ist zum Aktualisieren des Inhalts des ersten bzw. zweiten Adressenregisters, wenn das entsprechende Hitflag gesetzt ist.

16. Vorrichtung nach Anspruch 14, wobei die Steuereinrichtung eingerichtet ist, um den Füllstand des ersten oder zweiten Teils der Sequenz von Registerelementen unterhalb eines vorbestimmten Prozentwerts der variablen Länge des jeweiligen Teils der Sequenz von Registerelementen zu halten.

17. Vorrichtung nach Anspruch 16, wobei der vorbestimmte Prozentwert etwa 80% beträgt.

18. Vorrichtung nach Anspruch 16, wobei die Steuereinrichtung eingerichtet ist zum Durchführen eines Flushs in einem der beiden Teile der Sequenz von Registerelementen, wenn sie feststellt, dass einer der Füllstände dabei ist, den vorbestimmten Prozentwert zu übersteigen.

19. Vorrichtung nach Anspruch 1, wobei die Steuereinrichtung eingerichtet ist zum Steuern der Datenspeichereinrichtung, um die variablen Längen des ersten und zweiten Teils der Sequenz von Registerelementen unterhalb eines vorbestimmten Prozentwerts der Länge der Sequenz von Registerelementen zu halten.

20. Vorrichtung nach Anspruch 1, wobei die ersten Daten und die zweiten Daten von verschiedenen Datentypen sind.

21. Vorrichtung nach Anspruch 20, wobei die ersten Daten Transaktionselemente enthalten und die zweiten Daten aus einem Speicher gelesene oder in einen Speicher zu schreibende Daten sind.

22. Vorrichtung nach Anspruch 1, nämlich ein USB-Hostcontroller (USB: Universal Serial Bus).

23. Vorrichtung nach einem der Ansprüche 1 bis 22, wobei die Vorrichtung ein integrierter Southbridge-Schaltkreischip ist.

24. Verfahren zum Betreiben einer Datenspeichereinrichtung (**200**-**270**, **300**, **1445**, **1520**) mit einem Queuemechanismus, um eine Vielzahl von Datenelementen zu speichern, wobei die Datenspeichereinrichtung eine Vielzahl von Registerelementen (**200**-**270**) enthält, von denen jedes Registerelement wenigstens ein Datenelement speichern kann, wobei die Vielzahl von Registerelementen eingerichtet ist, um eine Sequenz (**300**) von Registerelementen zu bilden, wobei das Verfahren umfasst:
Steuern der Datenspeichereinrichtung, um erste Daten in einem ersten Teil (**200**-**230**) der Sequenz von Registerelementen zu speichern; und
Steuern der Datenspeichereinrichtung, um zweite Daten in einem zweiten Teil (**240**-**270**) der Sequenz von Registerelementen zu speichern;
wobei der erste Teil und der zweite Teil von variablen Längen sind und die Summe der variablen Längen gleich der Länge der Sequenz von Registerelementen ist,
wobei das Verfahren eingerichtet ist zum Steuern sowohl des ersten als auch des zweiten Teils der Se-

quenz von Registerelementen, um die entsprechenden Daten FIFO-mäßig zu speichern, und
wobei sowohl der erste als auch der zweite Teil der Sequenz von Registerelementen ein inneres und ein äußeres Ende aufweisen und die jeweiligen inneren Enden (**230**, **270**) innerhalb der Sequenz von Registerelementen und die jeweiligen äußeren Enden (**200**, **240**) am Anfang oder Ende der Sequenz von Registerelementen liegen, wobei das Verfahren eingerichtet ist zum Steuern sowohl des ersten als auch des zweiten Teils der Sequenz von Registerelementen, um hereinkommenden Daten in den entsprechenden Teil am jeweiligen inneren Ende einzufüllen.

25. Verfahren nach Anspruch 24, wobei die ersten Daten periodische Daten und die zweiten Daten asynchrone Daten sind und das Verfahren eingerichtet ist zum Steuern des ersten Teils in periodischer Weise und des zweiten Teils in asynchroner Weise.

26. Verfahren nach Anspruch 25, wobei die ersten Daten und die zweiten Daten Datenelemente desselben Typs enthalten.

27. Verfahren nach Anspruch 26, wobei die Datenelemente desselben Typs Transaktionselemente sind.

28. Verfahren nach Anspruch 27, wobei das Verfahren eingerichtet ist zum Steuern der Datenspeichereinrichtung abhängig von einer geschätzten Dauer der Transfers, die von den Transaktionselementen dargestellt werden.

29. Verfahren nach Anspruch 24, weiterhin umfassend:
Holen der ersten und zweiten Daten aus einem Speicher.

30. Verfahren nach Anspruch 24, wobei der erste und zweite Teil der Sequenz keinen Überlapp aufweisen.

31. Verfahren nach Anspruch 24, wobei der erste und zweite Teil der Sequenz einen Überlapp von wenigstens einem Registerelement aufweisen.

32. Verfahren nach Anspruch 24, weiterhin umfassend:
Betreiben eines ersten Adressenregisters (**310**, **330**), das ein erstes Registerelement identifiziert, das eine Grenze des ersten Teils der Sequenz von Registerelementen bildet, wobei sich der erste Teil von einer Seite (**200**) der Sequenz zu dem identifizierten ersten Registerelement erstreckt; und
Betreiben eines zweiten Adressenregisters (**310**, **320**), das ein zweites Registerelement identifiziert, das eine Grenze des zweiten Teils der Sequenz von Registerelementen bildet, wobei sich der zweite Teil von der anderen Seite (**240**) der Sequenz zu dem identifizierten zweiten Registerelement erstreckt.

33. Verfahren nach Anspruch 32, wobei das erste Adressenregister (**310**) gleich dem zweiten Adressenregister (**310**) ist.

34. Verfahren nach Anspruch 32, wobei das identifizierte erste Registerelement gleich dem identifizierten zweiten Registerelement ist.

35. Verfahren nach Anspruch 32, wobei das erste Adressenregister (**330**) von dem zweiten Adressenregister (**320**) und das identifizierte erste Registerelement von dem identifizierten zweiten Registerelement verschieden ist.

36. Verfahren nach Anspruch 32, weiterhin umfassend:
Umschalten des Betriebs zwischen einem ersten und einem zweiten Modus, wobei in dem ersten Modus das erste Adressenregister (**310**) gleich dem zweiten Adressenregister (**310**) ist und in dem zweiten Modus das erste Adressenregister (**330**) von dem zweiten Adressenregister (**320**) verschieden ist.

37. Verfahren nach Anspruch 32, weiterhin umfassend:
Betreiben eines dritten Adressenregisters (**340**), das ein drittes Registerelement identifiziert, das den Füllstand des ersten Teils der Sequenz von Registerelementen angibt; und
Betreiben eines vierten Adressenregisters (**350**), das ein viertes Registerelement identifiziert, das den Füllstand des zweiten Teils der Sequenz von Registerelementen angibt.

38. Verfahren nach Anspruch 37, weiterhin umfassend:
Setzen eines ersten oder zweiten Hitflags, wenn der Füllstand des ersten oder zweiten Teils der Sequenz von Registerelementen jenseits der jeweiligen Grenze liegt; und
Aktualisieren des Inhalts des ersten bzw. zweiten Adressenregisters, wenn das entsprechende Hitflag gesetzt ist.

39. Verfahren nach Anspruch 37, wobei das Verfahren eingerichtet ist zum Halten des Füllstand des ersten oder zweiten Teils der Sequenz von Registerelementen unter einem vorbestimmten Prozentwert der variablen Länge des jeweiligen Teils der Sequenz von Registerelementen.

40. Verfahren nach Anspruch 39, wobei der vorbestimmte Prozentwert etwa 80% beträgt.

41. Verfahren nach Anspruch 39, wobei das Verfahren eingerichtet ist zum Durchführen eines Flushs in dem ersten oder zweiten Teil der Sequenz von Registerelementen, wenn einer der Füllstände dabei ist,

den vorbestimmten Prozentwert zu überschreiten.

42. Verfahren nach Anspruch 24, wobei das Verfahren eingerichtet ist zum Steuern der Datenspeichereinrichtung, um die variablen Längen des ersten und zweiten Teils der Sequenz von Registerelementen unterhalb eines vorbestimmten Prozentwerts der Länge der Sequenz von Registerelementen zu halten.

43. Verfahren nach Anspruch 24, wobei die ersten Daten und die zweiten Daten von verschiedenen Datentypen sind.

44. Verfahren nach Anspruch 43, wobei die ersten Daten Transaktionselemente und die zweiten Daten Daten enthalten, die aus einem Speicher gelesen oder in einen Speicher zu schreiben sind.

45. Verfahren nach Anspruch 24, nämlich zum Betreiben eines USB-Hostcontrollers (USB: Universal Serial Bus).

46. Verfahren nach Anspruch 24, nämlich zum Betreiben eines integrierten Southbridge-Schaltkreischips.

Es folgen 11 Blatt Zeichnungen

Anhängende Zeichnungen

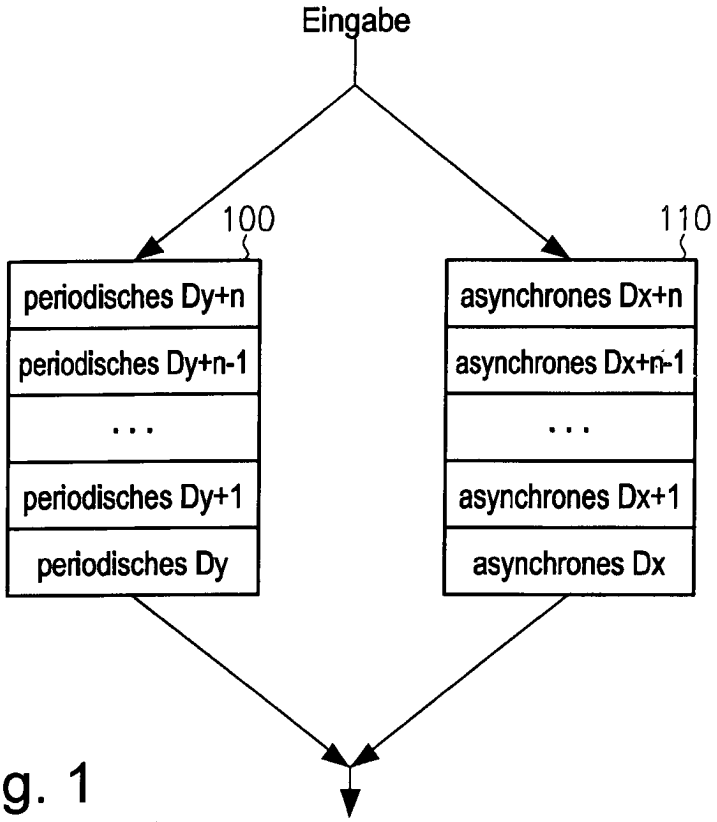


Fig. 1
(Stand der Technik)

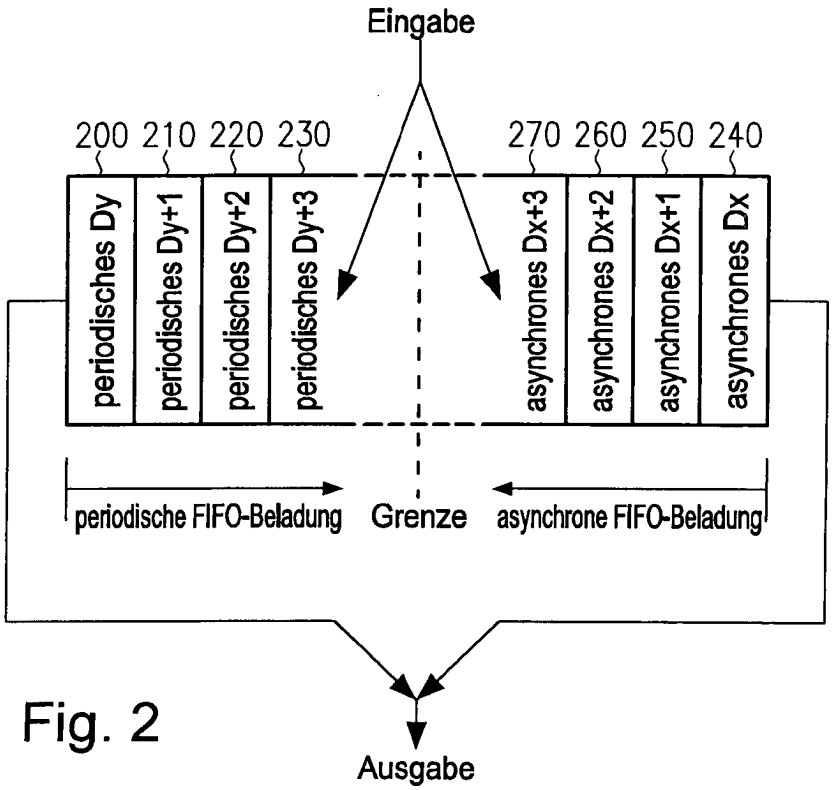


Fig. 2

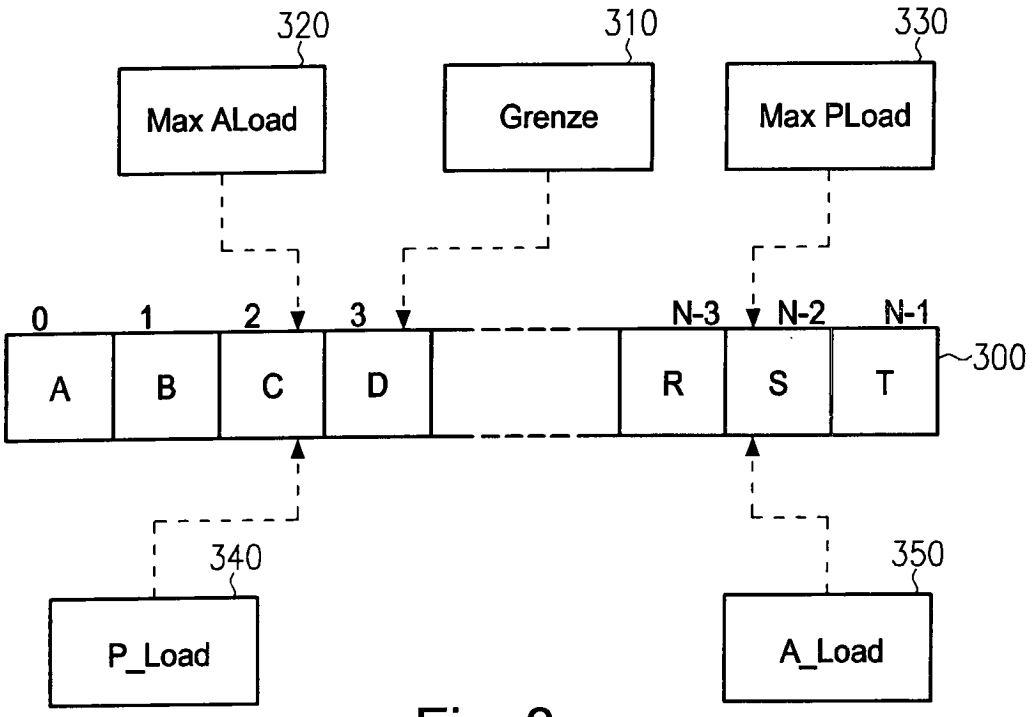

320
310
330
Max ALoad
Grenze
Max PLoad
0
1
2
3
N-3
N-2
N-1
A
B
C
D
R
S
T
300
340
350
P_Load
A_Load

Fig. 3

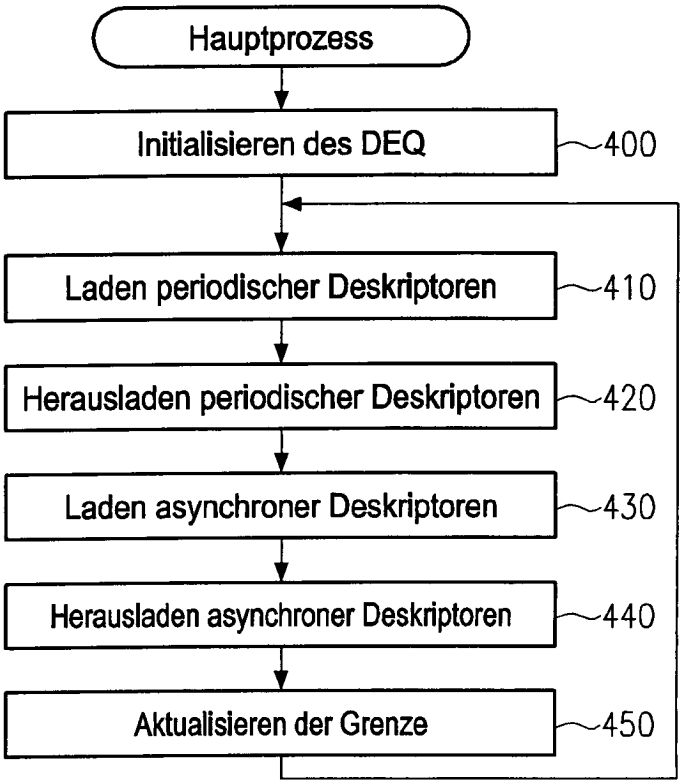

Hauptprozess
Initialisieren des DEQ
400
Laden periodischer Deskriptoren
410
Herausladen periodischer Deskriptoren
420
Laden asynchroner Deskriptoren
430
Herausladen asynchroner Deskriptoren
440
Aktualisieren der Grenze
450

Fig. 4

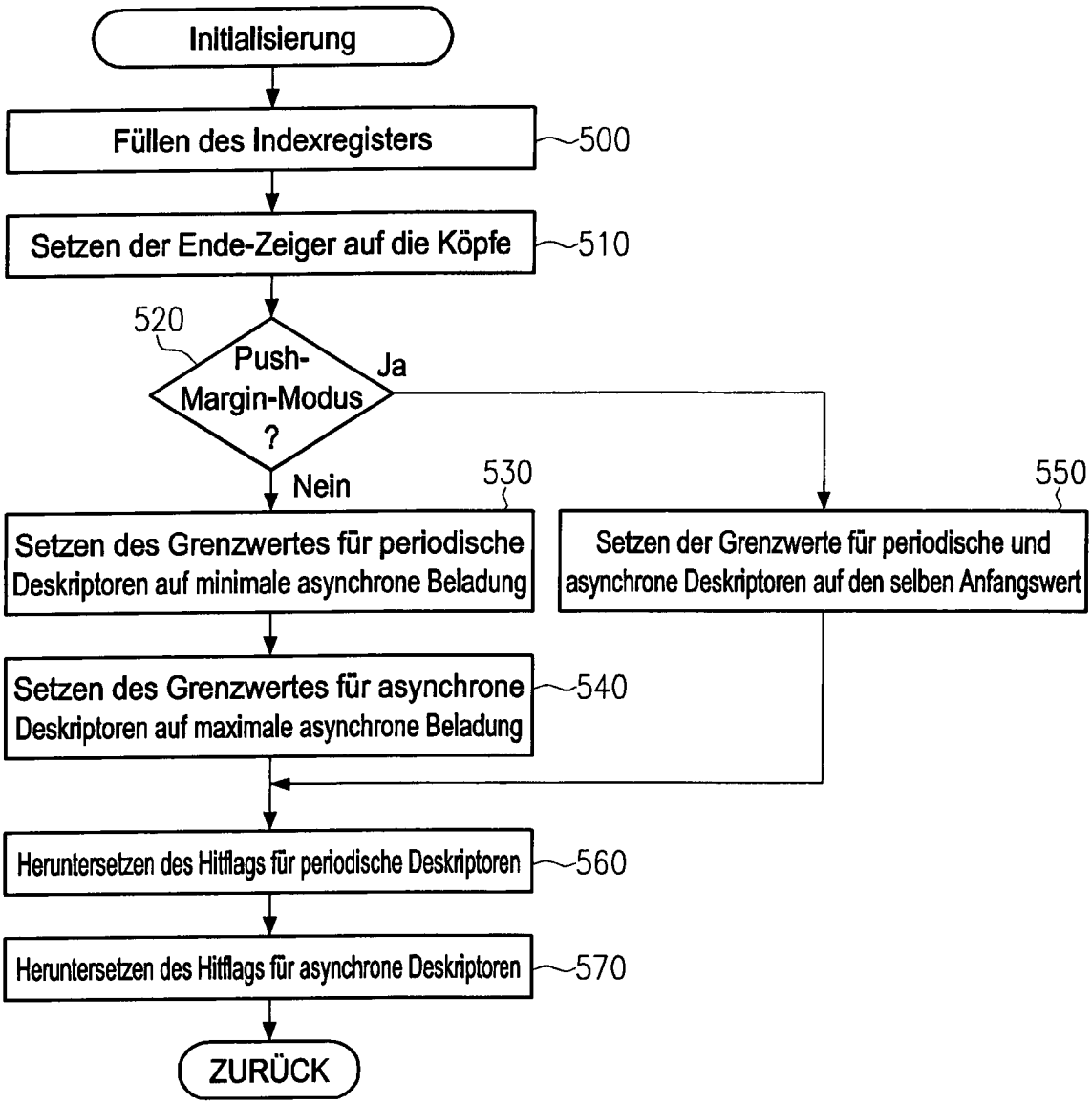

Initialisierung
Füllen des Indexregisters
500
Setzen der Ende-Zeiger auf die Köpfe
510
520
Push-Margin-Modus ?
Ja
Nein
530
550
Setzen des Grenzwertes für periodische Deskriptoren auf minimale asynchrone Beladung
Setzen der Grenzwerte für periodische und asynchrone Deskriptoren auf den selben Anfangswert
Setzen des Grenzwertes für asynchrone Deskriptoren auf maximale asynchrone Beladung
540
Heruntersetzen des Hitflags für periodische Deskriptoren
560
Heruntersetzen des Hitflags für asynchrone Deskriptoren
570
ZURÜCK

Fig. 5

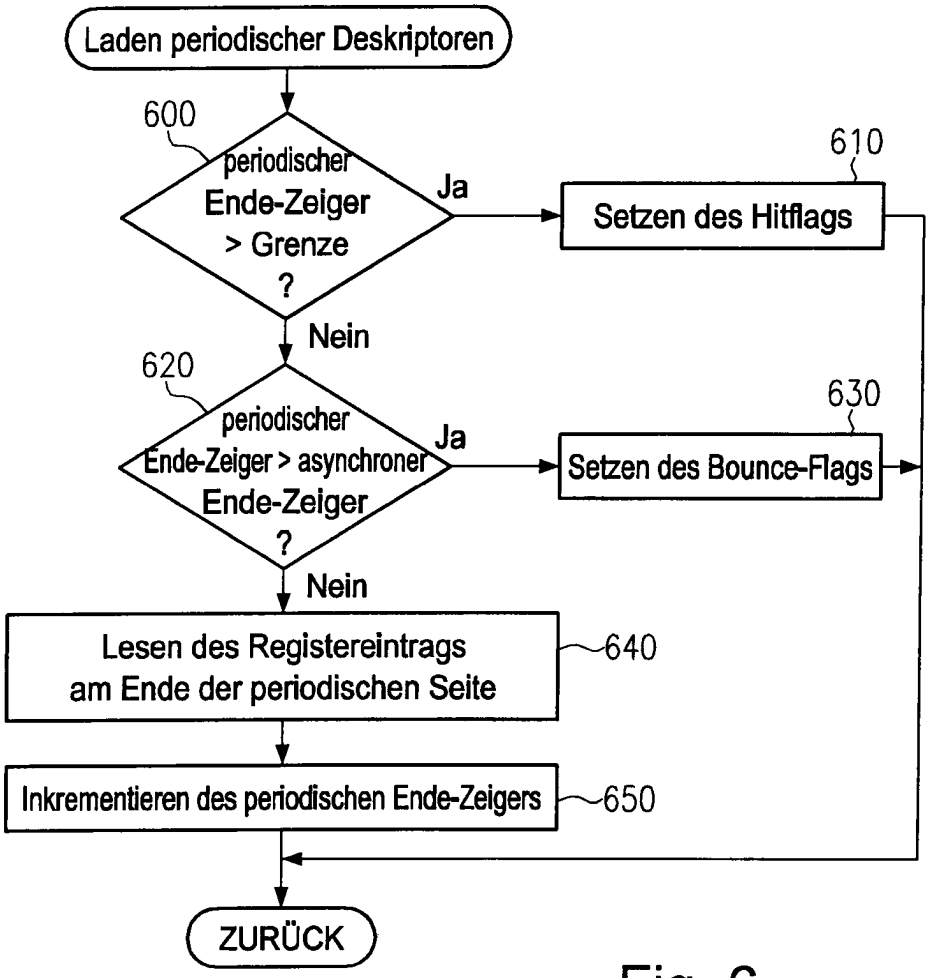

Laden periodischer Deskriptoren
600
periodischer Ende-Zeiger > Grenze ?
Ja
610
Setzen des Hitflags
Nein
620
periodischer Ende-Zeiger > asynchroner Ende-Zeiger ?
Ja
630
Setzen des Bounce-Flags
Nein
Lesen des Registereintrags am Ende der periodischen Seite
640
Inkrementieren des periodischen Ende-Zeigers
650
ZURÜCK

Fig. 6

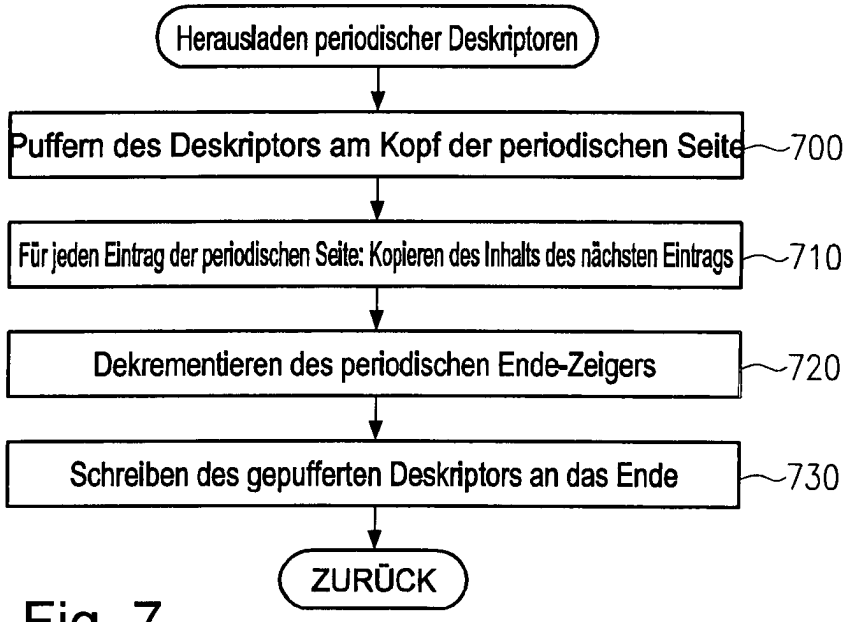

Herausladen periodischer Deskriptoren
Puffern des Deskriptors am Kopf der periodischen Seite
700
Für jeden Eintrag der periodischen Seite: Kopieren des Inhalts des nächsten Eintrags
710
Dekrementieren des periodischen Ende-Zeigers
720
Schreiben des gepufferten Deskriptors an das Ende
730
ZURÜCK

Fig. 7

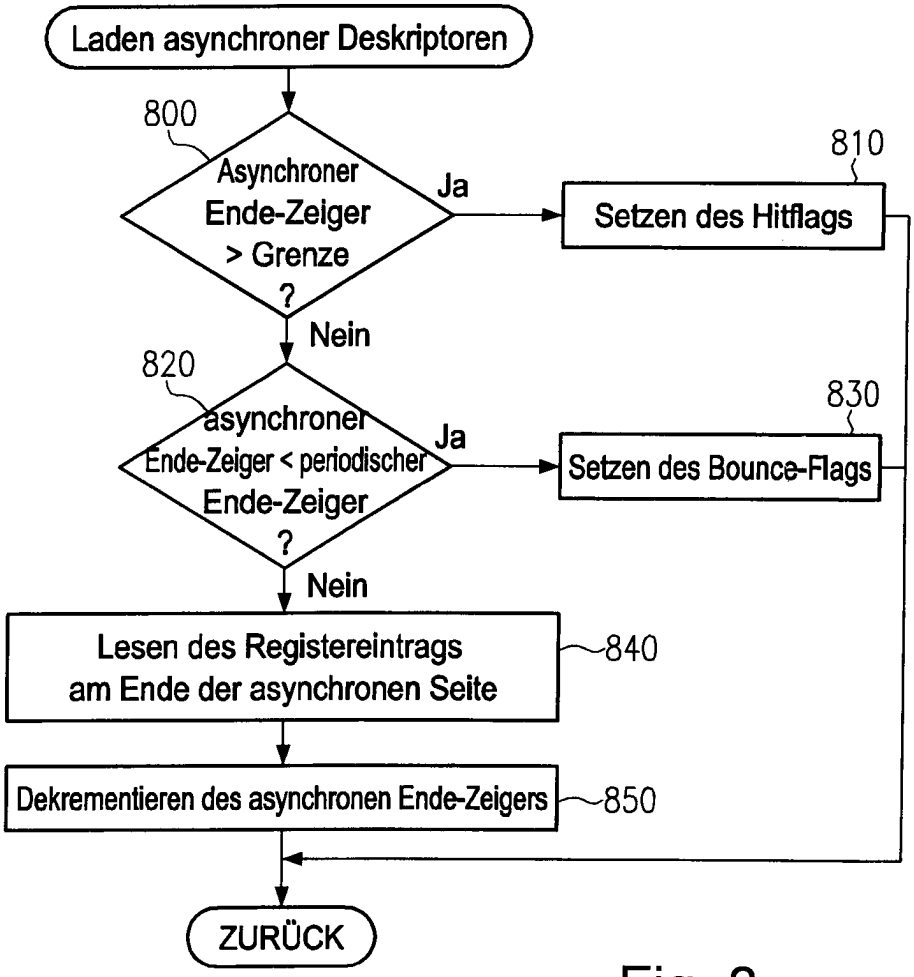

Laden asynchroner Deskriptoren
800
Asynchroner Ende-Zeiger > Grenze ?
Ja
810
Setzen des Hitflags
Nein
820
asynchroner Ende-Zeiger < periodischer Ende-Zeiger ?
Ja
830
Setzen des Bounce-Flags
Nein
Lesen des Registereintrags am Ende der asynchronen Seite
840
Dekrementieren des asynchronen Ende-Zeigers
850
ZURÜCK

Fig. 8

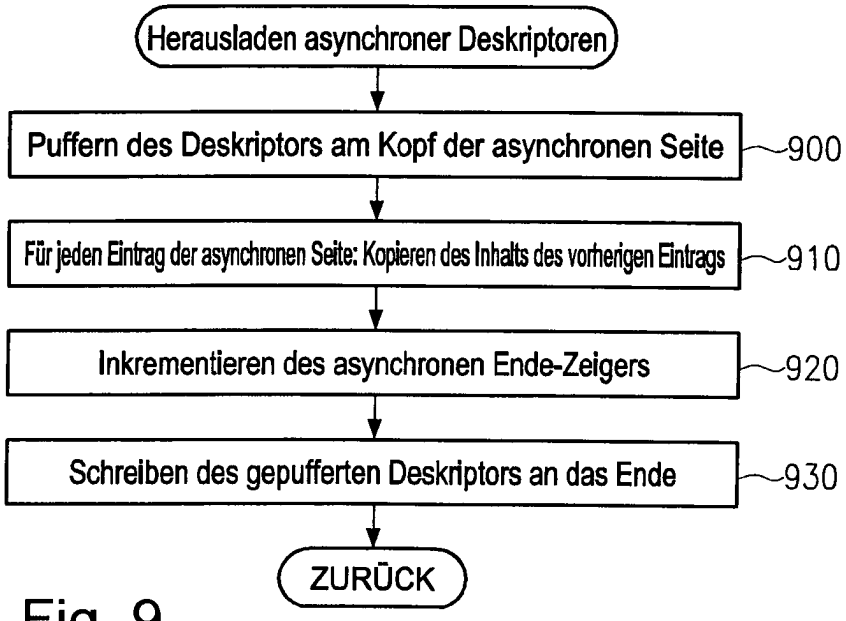

Herausladen asynchroner Deskriptoren
Puffern des Deskriptors am Kopf der asynchronen Seite
900
Für jeden Eintrag der asynchronen Seite: Kopieren des Inhalts des vorherigen Eintrags
910
Inkrementieren des asynchronen Ende-Zeigers
920
Schreiben des gepufferten Deskriptors an das Ende
930
ZURÜCK

Fig. 9

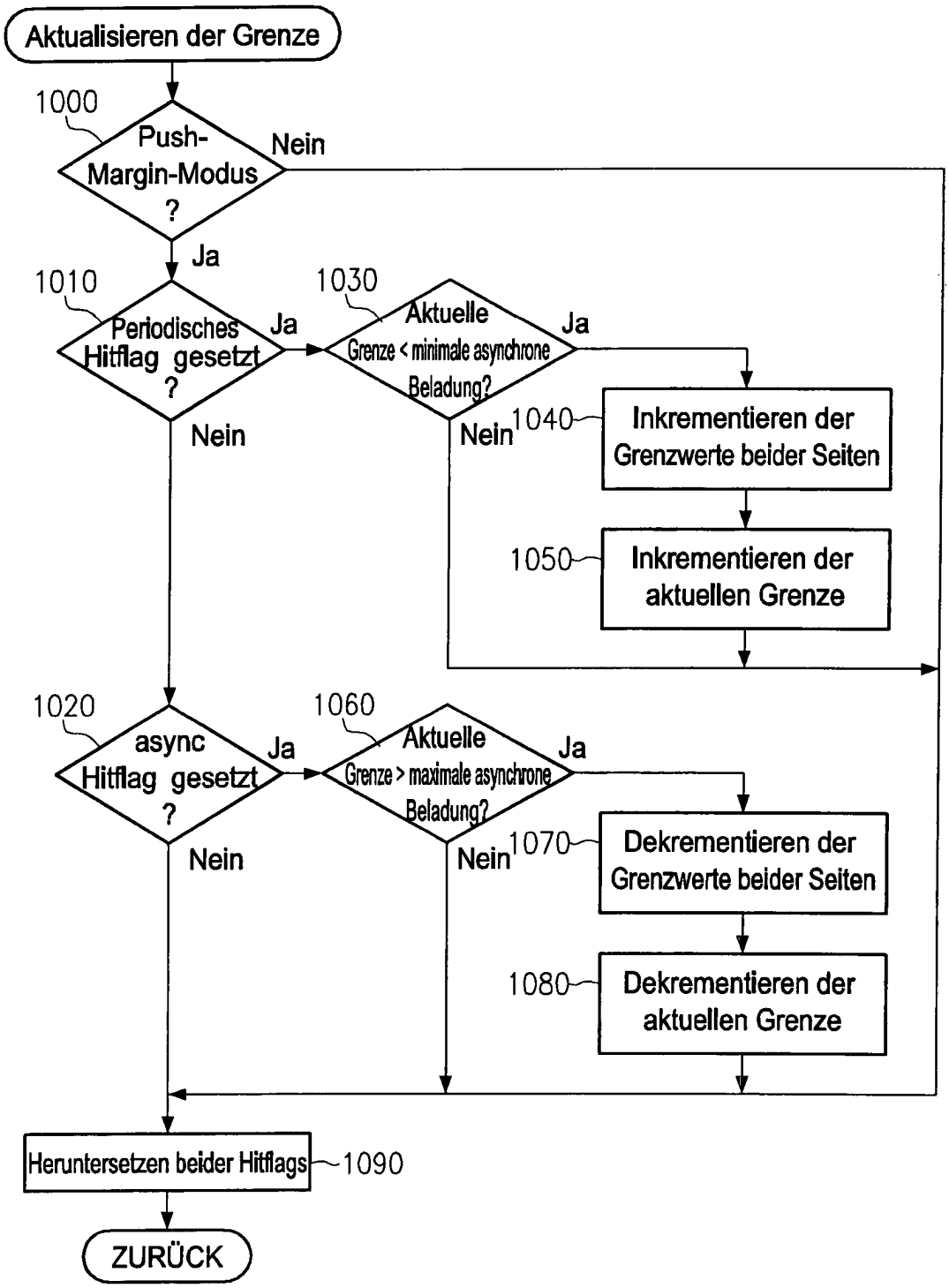

Aktualisieren der Grenze
1000
Push-Margin-Modus ?
Nein
Ja
1010
Periodisches Hitflag gesetzt ?
Ja
1030
Aktuelle Grenze < minimale asynchrone Beladung?
Ja
Nein
1040
Inkrementieren der Grenzwerte beider Seiten
1050
Inkrementieren der aktuellen Grenze
Nein
1020
async Hitflag gesetzt ?
Ja
1060
Aktuelle Grenze > maximale asynchrone Beladung?
Ja
Nein
1070
Dekrementieren der Grenzwerte beider Seiten
1080
Dekrementieren der aktuellen Grenze
Nein
Heruntersetzen beider Hitflags
1090
ZURÜCK

Fig.10

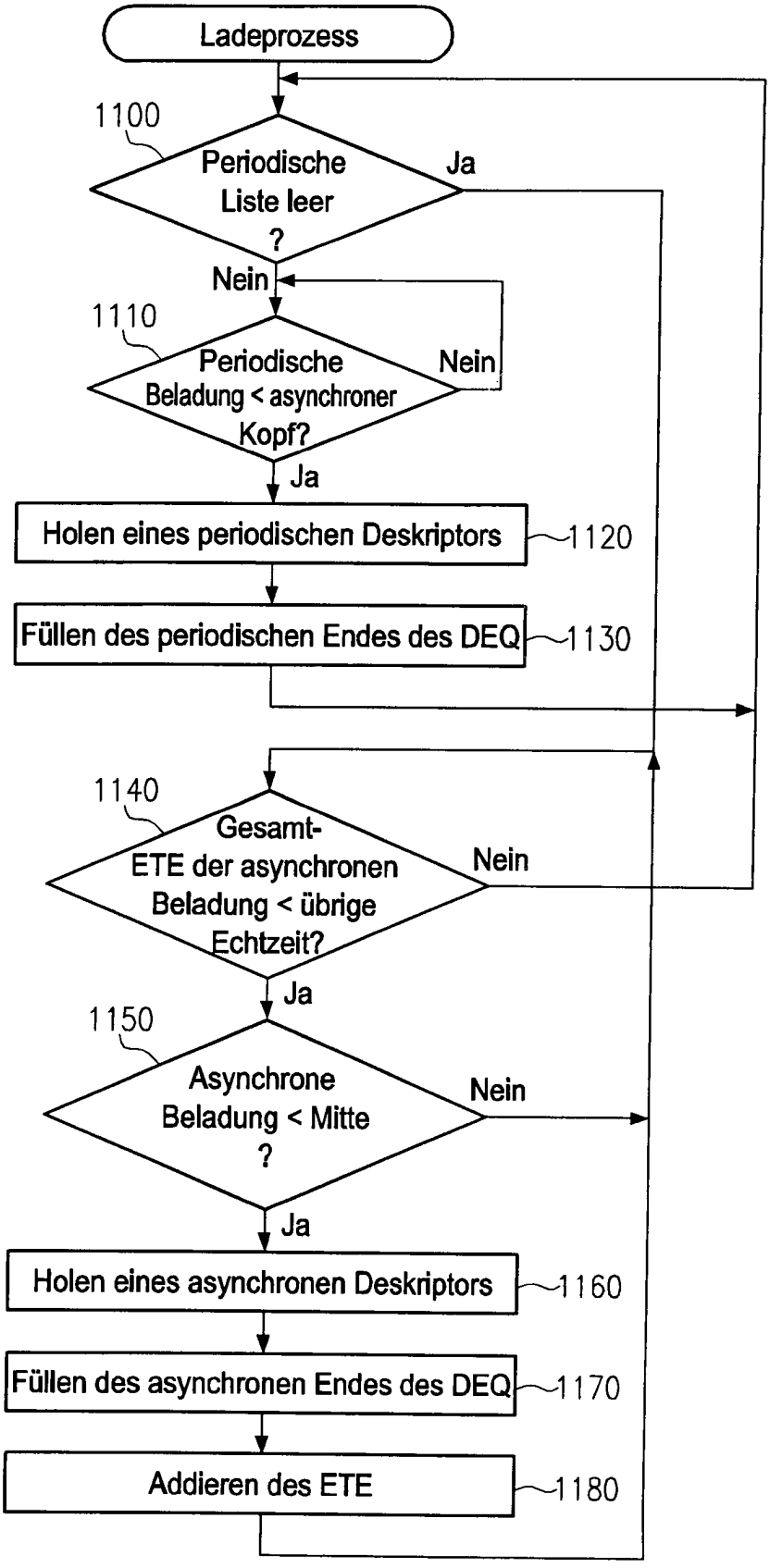

Ladeprozess
1100
Periodische Liste leer ?
Ja
Nein
1110
Periodische Beladung < asynchroner Kopf?
Nein
Ja
Holen eines periodischen Deskriptors
1120
Füllen des periodischen Endes des DEQ
1130
1140
Gesamt-ETE der asynchronen Beladung < übrige Echtzeit?
Nein
Ja
1150
Asynchrone Beladung < Mitte ?
Nein
Ja
Holen eines asynchronen Deskriptors
1160
Füllen des asynchronen Endes des DEQ
1170
Addieren des ETE
1180

Fig.11

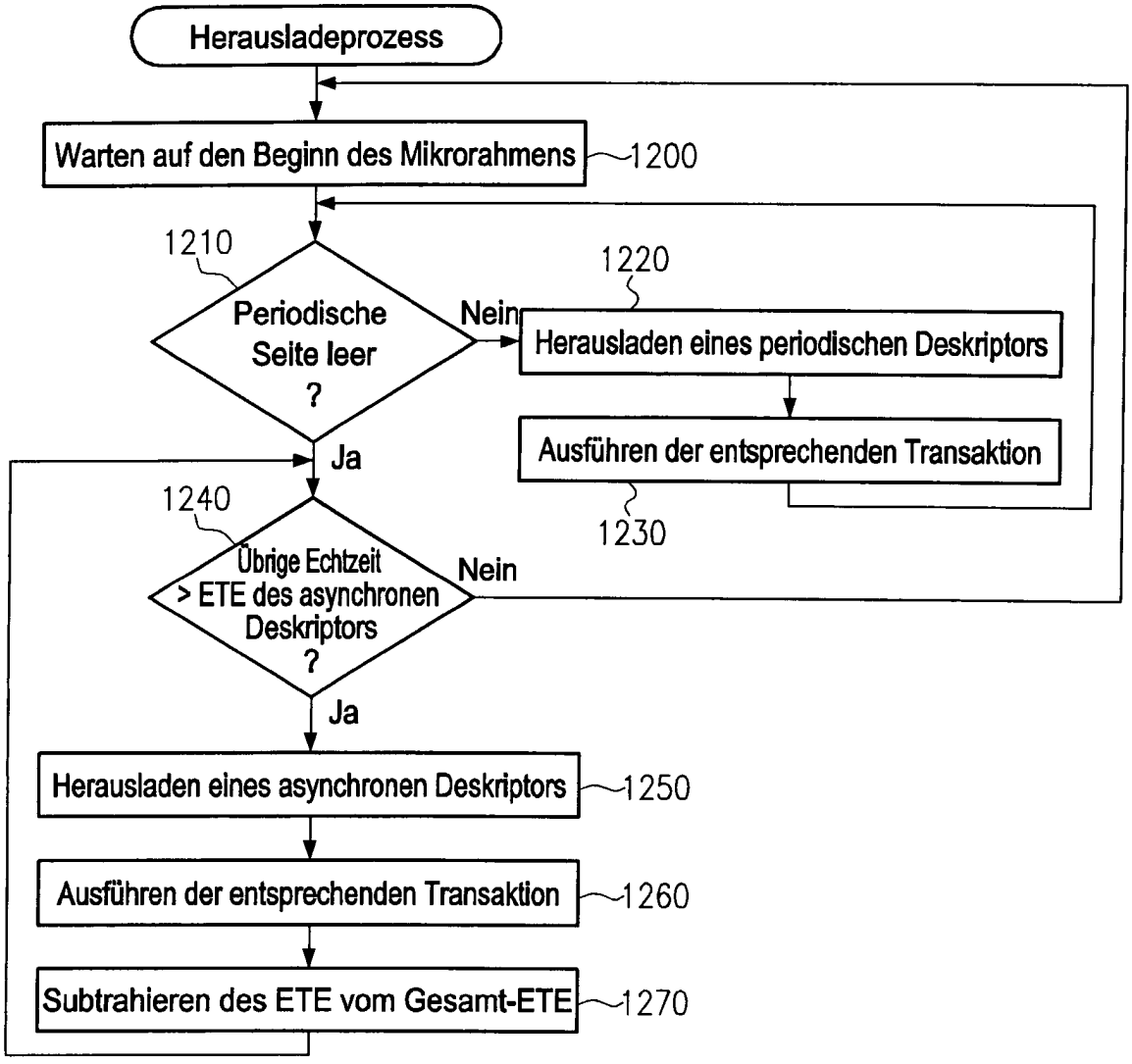

Herausladeprozess
Warten auf den Beginn des Mikrorahmens
1200
1210
Periodische Seite leer ?
Nein
1220
Herausladen eines periodischen Deskriptors
Ausführen der entsprechenden Transaktion
1230
Ja
1240
Übrige Echtzeit > ETE des asynchronen Deskriptors ?
Nein
Ja
Herausladen eines asynchronen Deskriptors
1250
Ausführen der entsprechenden Transaktion
1260
Subtrahieren des ETE vom Gesamt-ETE
1270

Fig.12

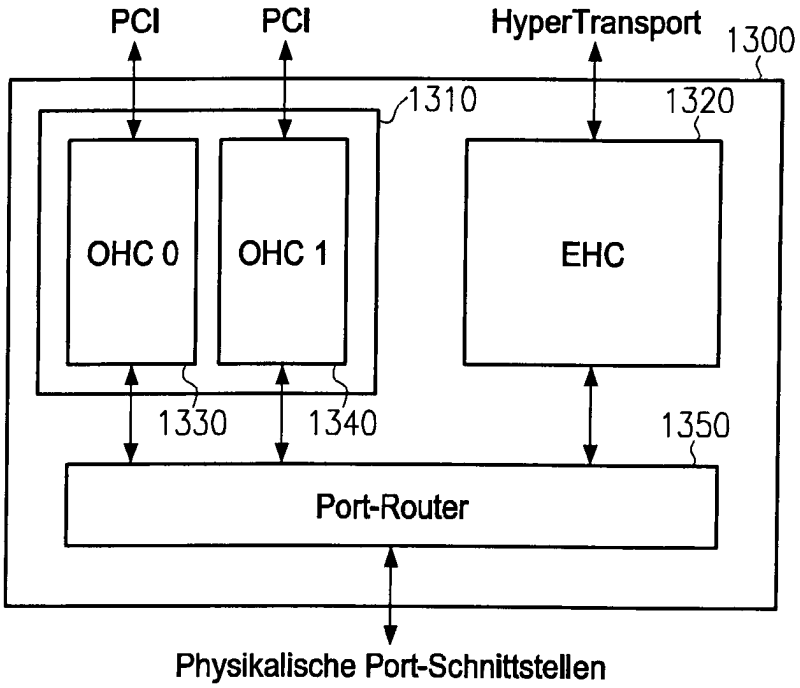

PCI
PCI
HyperTransport
1300
1310
1320
OHC 0
OHC 1
EHC
1330
1340
1350
Port-Router
Physikalische Port-Schnittstellen

Fig.13

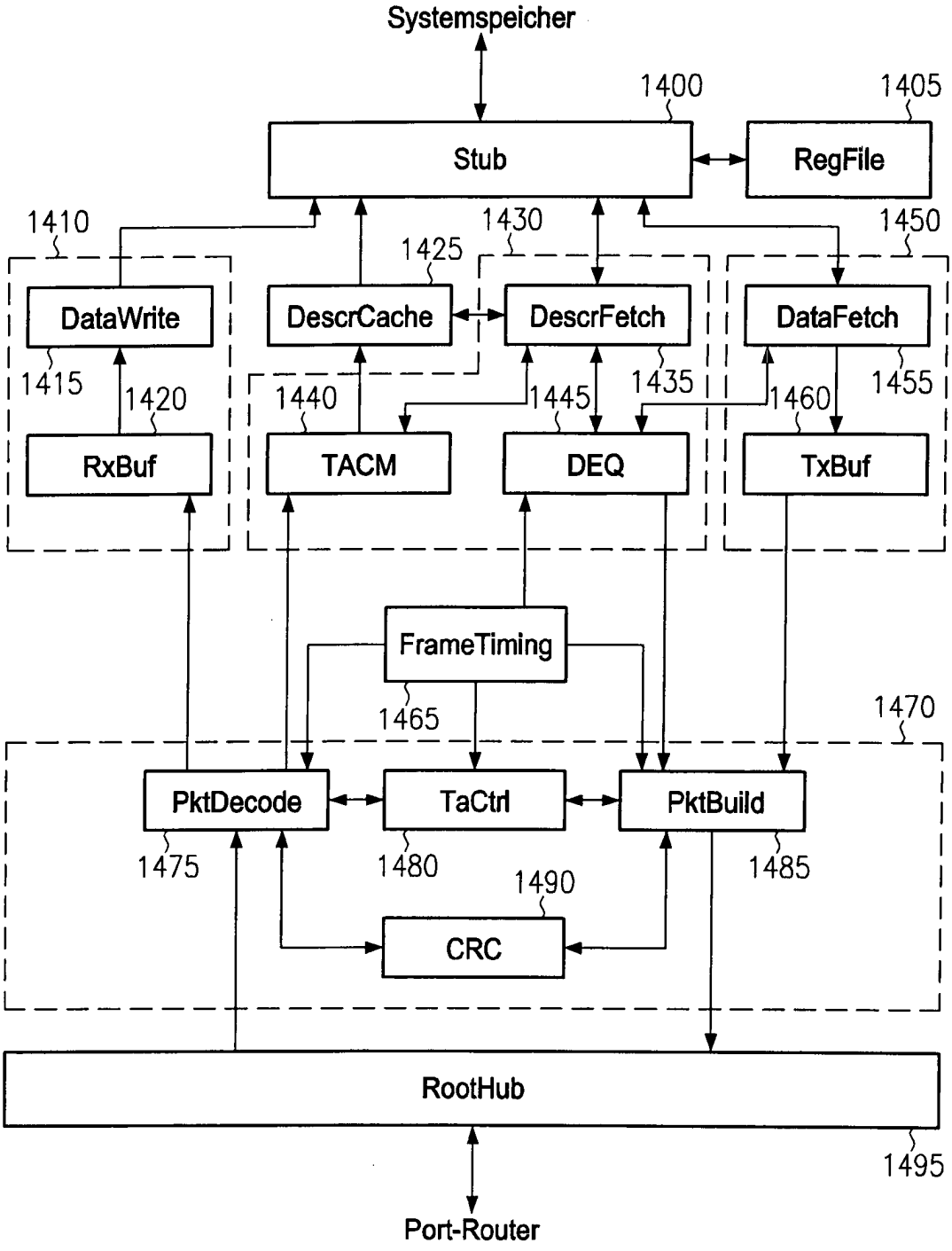

Systemspeicher
1400
1405
Stub
RegFile
1410
1430
1450
1425
DataWrite
DescrCache
DescrFetch
DataFetch
1415
1420
1440
1445
1435
1460
1455
RxBuf
TACM
DEQ
TxBuf
FrameTiming
1465
1470
PktDecode
TaCtrl
PktBuild
1475
1480
1485
1490
CRC
RootHub
1495
Port-Router

Fig.14

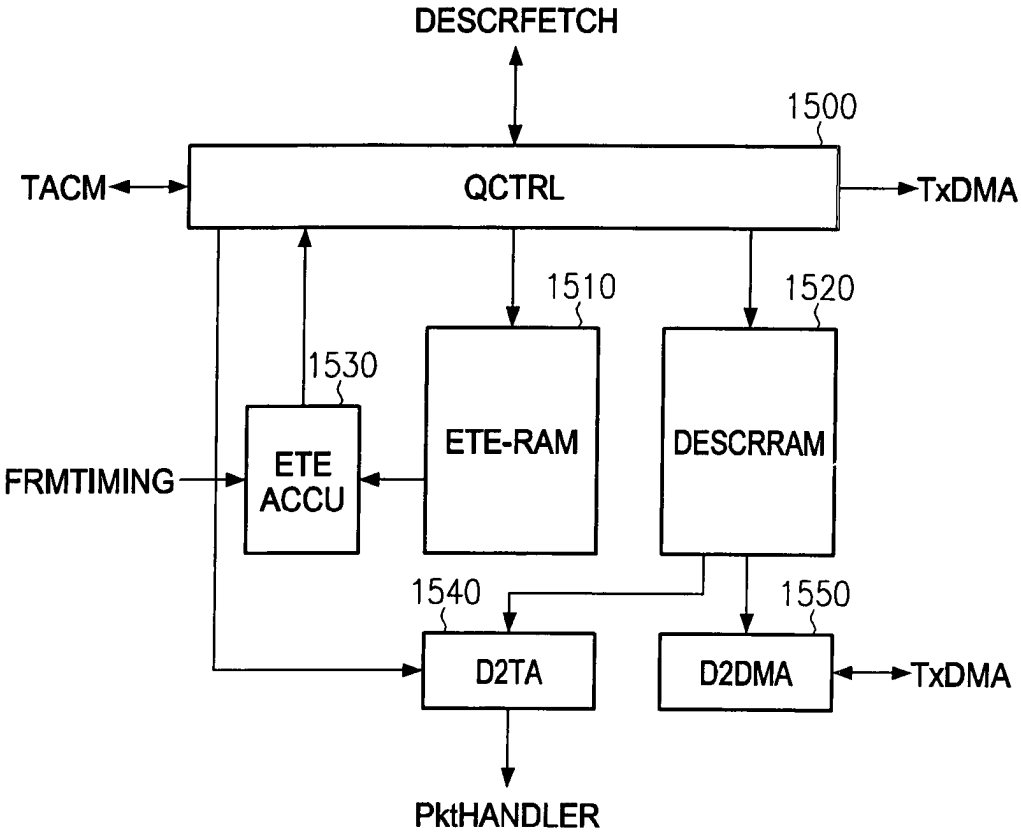

DESCRFETCH
1500
TACM
QCTRL
TxDMA
1510
1520
1530
ETE-RAM
DESCRRAM
FRMTIMING
ETE ACCU
1540
1550
D2TA
D2DMA
TxDMA
PktHANDLER

Fig.15