(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 199 83 709 B4** 2007.02.22

(12) **Patentschrift**

(21) Deutsches Aktenzeichen: **199 83 709.0**
(86) PCT-Aktenzeichen: **PCT/US99/19596**
(87) PCT-Veröffentlichungs-Nr.: **WO 2000/028418**
(86) PCT-Anmeldetag: **26.08.1999**
(87) PCT-Veröffentlichungstag: **18.05.2000**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **14.02.2002**
(45) Veröffentlichungstag der Patenterteilung: **22.02.2007**

(51) Int Cl.[8]: ***G06F 9/46*** (2006.01)

Innerhalb von drei 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. 2 Patentkostengesetz).

(30) Unionspriorität:
09/188,614 09.11.1998 US

(73) Patentinhaber:
Intel Corporation, Santa Clara, Calif., US

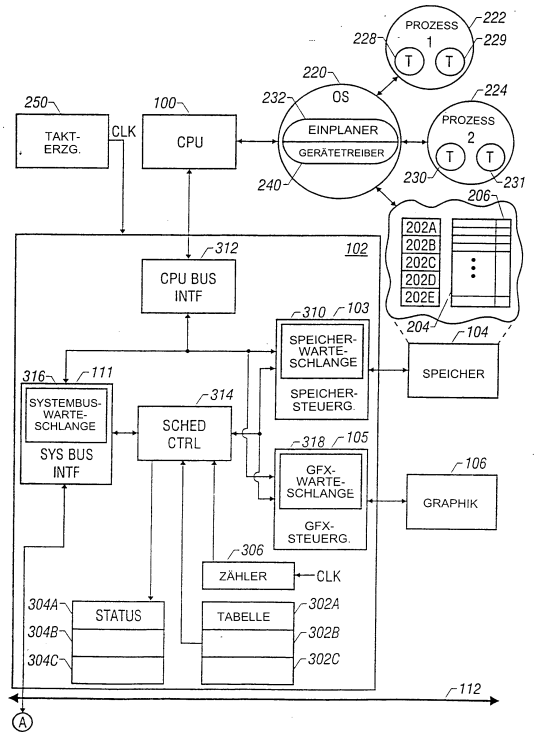
(74) Vertreter:
Zenz, Helber, Hosbach & Partner GbR, 45128 Essen

(72) Erfinder:
Haghighi, Siamack, Chandler, Ariz., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
EP 08 17 041 A2
JONES, M.B. et al.: Support for User-Centric Modular Real-Time Resource Management in the Rialto Operating System. In Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video, November 1995;

(54) Bezeichnung: **Einplanung von Ressourcenanforderungen in einem Computersystem**

(57) Hauptanspruch: Verfahren zum zeitlichen Einplanen von Ressourcen-Anforderungen aus aktiven Ausführungsentitäten, die Tasks, Prozesse und/oder Threads umfassen, wobei die Ressourcen-Anforderungen jeweils eine von mehreren Ressourcen eines Computersystems anfordern, wobei das Computersystem wenigstens einen mit einer CPU gekoppelten Controller (102, 110) aufweist, wobei jede Ressource (104, 106, 112, 118, 120) mit einem zugehörigen Controller (102, 110) gekoppelt ist, wobei:
a) jeder aktiven Ausführungsentität für jede der Ressourcen jeweils wenigstens ein Zeitfenster (402) innerhalb eines Einplanzyklus (400) zugewiesen wird, wobei die Zeitfenster (402) von einem auf der CPU laufenden Betriebssystem (220) zugewiesen werden, wobei die Zeitfenster (402) jeweils aus einer Anzahl von Zeit-Slots bestehen, wobei die Zuweisung der Zeitfenster in einer Mehrzahl von Tabellen (202) gespeichert wird, wobei jeweils eine Tabelle (202A–202E) einer Ressource zugeordnet ist, wobei die Tabellen (202A–202E) Zeit-Slot-Zuweisungen speichern, in denen jedem Zeit-Slot ein Identifizierer einer Ausführungsentität zugeordnet ist;
b) wobei ein Scheduler (232) des Betriebssystems (220) jeweils eine...



Beschreibung

[0001] Die Erfindung bezieht sich auf die deterministische Einplanung von Anforderungen in Systemen.

[0002] Softwareebenen (Layer) in einem System (wie beispielsweise einem Computer) umfassen typischerweise ein Betriebssystem und Anwendungsprogramme. Wenn Anwendungsprogramme ausgeführt werden, können ein oder mehrere Prozesse, Tasks oder andere Grundarbeitseinheiten oder Ausführungsentitäten erzeugt werden. Bei bestimmten Betriebssystemen, wie beispielsweise den Betriebssystemen Windows 95 oder Windows NT® der Microsoft Corporation, kann jeder Prozeß eine oder mehrere Arbeitseinheiten (die als "Threads" bezeichnet werden) enthalten, die einen in dem Adreßraum des Prozesses enthaltenen Code ausführen, um zugewiesene Funktionen auszuführen. Zu einem Elternprozeß gehörende Threads können so zugewiesen sein, daß sie verschiedene Funktionen ausführen. Bei einem Spreadsheet-Prozeß beispielsweise können Threads erzeugt werden, um zu rechnen, zu drucken, Benutzereingaben aufzunehmen, Hilfefunktionen zur Verfügung zu stellen usw. Bei anderen Betriebssystemen kann eine Task oder ein Prozeß die Grundarbeitseinheit oder Ausführungsentität bilden, die zur Ausführung durch eine zentrale Verarbeitungseinheit (CPU) eingeplant werden kann.

[0003] Betriebssysteme können einen Einplaner (Scheduler) enthalten, um mehrere aktive Threads oder Prozesse zu verwalten. Verschiedene Arten der Betriebssysteme können verschiedene Einplanungsschemata aufweisen. Bei einigen Windows-Betriebssystemen beispielsweise werden Zeitscheiben (Zeitfenster) aktiven Threads in einem umlaufenden Schema zugewiesen, während welcher es den zugehörigen Threads gestattet wird, ausgeführt zu werden. Darüber hinaus können bei einigen Windows-Betriebssystemen Prioritätsklassen den Threads zugewiesen werden. Threads in der höchsten Prioritätsklasse werden in den ihnen zugewiesenen Zeitscheiben zuerst ausgeführt, gefolgt von Threads in niedrigeren Prioritätsklassen. So kann in einer bestimmten Prioritätsklasse die Einplanung in einer umlaufenden Weise (round robin) durchgeführt werden. Ein Thread wird solange abgearbeitet, bis eines oder mehrere Ereignisse auftreten: Die Zeitscheibe endet oder der Thread durch einen anderen Thread einer höheren Prioritätsklasse, der zum Ablaufen bereit ist, verdrängt wird (preempted).

Stand der Technik

[0004] Aus der EP 0 817 041 A2 ist ein Verfahren zum Zuteilen von Ressourcen bekannt, bei dem ein Ressourcen-Manager verwendet wird. Dieser Manager empfängt Anforderungen nach Ressource, prüft deren Verfügbarkeit und reserviert die verfügbaren Ressourcen.

[0005] Aus M.B. Jones et al.: Support for User-Centric Modular Real-Time Resource Management in the Rialto Operating System sind weitere Möglichkeiten zur Ressourcenverteilung mit einem Ressourcen-Planer bekannt. Der lokale Resourcen-Planer verhandelt mit Einzelanwendungen und versucht die Verfügbarkeit der laufenden Anwendungen im Hinblick auf die Ressourcen Zuordnungen zu optimieren.

[0006] Verschiedene Prozesse, Threads oder andere Arbeitseinheiten können Systemressourcen in verschiedener Weise benutzen. Beispielsweise können bei einem Videowiedergabe- und- decodierprozeß Daten von einem Compact-Disc-(CD-) oder digitalen Videoplatten-(DVD-) Laufwerk zum Systemspeicher, zwischen der CPU und dem Systemspeicher und aus dem Systemspeicher zum Videospeicher zur Anzeige durch eine Graphikkarte auf einem Monitor übertragen werden. Grundsätzlich ist die Datenübertragung von dem CD- oder DVD-Laufwerk zum Systemspeicher langsamer als die Datenübertragung zwischen der Graphikkarte und dem Systemspeicher, welche wiederum langsamer sein kann als die Datenübertragung zwischen der CPU und dem Systemspeicher. Somit können Systemressourcen (einschließlich beispielsweise des Systemspeichers, der Busse und anderer Einrichtungen) in Abhängigkeit von den Anforderungen der verschiedenen Arbeitseinheiten oder Ausführungsentitäten in unterschiedlicher Weise benutzt werden.

[0007] Herkömmliche Betriebssysteme berücksichtigen üblicherweise die verschiedenen Ressourcenanforderungen der verschiedenen Prozesse, Threads oder anderen Arbeitseinheiten nicht effektiv. Solche herkömmlichen Betriebssysteme planen Prozesse, Threads oder andere Arbeitseinheiten auf der Anwendungsebene ein; beispielsweise werden die der jeweiligen Anwendung zugeordneten Prozesse, Threads oder anderen Arbeitseinheiten einer vorgegebenen Prioritätsklasse zugewiesen. Typischerweise werden Anforderungen aus den Arbeitseinheiten in Übereinstimmung mit der zuvor zugewiesenen Priorität und dem Einplanungsprotokoll ohne Rücksicht darauf eingeplant, ob die benötigten Systemressourcen verfügbar sind.

Aufgabenstellung

[0008] Es ist Aufgabe der Erfindung, die Ressourcen-Einplanung für ablaufende Tasks, Prozesse oder Threads derart zu verbessern, dass die Zuteilung von Systemressourcen effektiver erfolgt.

[0009] Diese Aufgabe wird erfindungsgemäß durch ein Verfahren mit den Merkmalen des Anspruchs 1

sowie ein System mit den Merkmalen des Anspruchs 9 gelöst.

[0010] Vorteilhafte Ausgestaltungen der Erfindung sind in den Unteransprüchen angegeben.

Ausführungsbeispiel

[0011] Fig. 1 ist ein Blockschaltbild eines Systems eines Ausführungsbeispiels der Erfindung.

[0012] Fig. 2A und Fig. 2B sind eine Blockdarstellung der Ebenen in dem System gemäß Fig. 1.

[0013] Fig. 3 veranschaulicht einen Einplanungszyklus in Übereinstimmung mit einem Ausführungsbeispiel.

[0014] Fig. 4 ist ein Ablaufdiagramm eines Einplanungsmoduls gemäß einem Ausführungsbeispiel in dem System gemäß Fig. 1.

[0015] Fig. 5 ist ein Ablaufdiagramm einer Basis-Eingabe/Ausgabe-System(BIOS)-Routine gemäß einem Ausführungsbeispiel in dem System gemäß Fig. 1.

[0016] Fig. 6 ist ein Ablaufdiagramm eines Betriebssystems gemäß einem Ausführungsbeispiel in dem System gemäß Fig. 1.

[0017] Ein System gemäß den Ausführungsbeispielen der Erfindung enthält verschiedene Ressourcen, die von Software- oder Firmware-Ebenen oder -Modulen, die in dem System ablaufen, benutzt werden oder auf die zugegriffen wird. Beispielsweise können die Systemressourcen einen Systemspeicher, einen oder mehrere Busse und weitere Einrichtungen umfassen. Ein Einplaner gemäß den Ausführungsbeispielen, der in dem System abläuft, plant Anforderungen aus Basisarbeitseinheiten oder Ausführungsentitäten (z.B. Prozessen, Tasks oder Threads) ein, die von den Software- und Firmware-Ebenen gemäß vorgegebenen Kriterien, welche bei einigen Ausführungsbeispielen umfassen können, ob bestimmte Systemressourcen verfügbar sind, sowie die Verzögerungs- und Bandbreitenerfordernisse der Anforderungen, erzeugt werden. Gemäß einigen Ausführungsbeispielen wird dem Einplaner eine Rückkopplung über die Systemressourcenverfügbarkeit und -benutzung zur Verfügung gestellt. Durch Verwendung einer deterministischen Lösung, bei welcher die Verfügbarkeit und Ausnutzung der angeforderten Ressourcen durch einen Einplaner bestimmt werden können, ist der Einplaner besser in der Lage, zu garantieren, daß eine Anforderung aus einer Ausführungsentität in Übereinstimmung mit den Bandbreiten- und Verzögerungserfordernissen der anfordernden Entität bedient wird.

[0018] Es wird auf Fig. 1 Bezug genommen, in der ein Blockschaltbild eines Systems **10** gezeigt ist, welches beispielsweise ein Mehrzweckcomputer oder ein Spezialcomputer oder ein anderes mikroprozessor- oder mikrocontroller-basiertes System, ein Hand-Held-Computer, eine Set-Top-Box, ein Gerät, ein Spielsystem oder irgendein anderes System sein kann, das eine Steuereinrichtung enthält, wie beispielsweise eine anwendungsspezifische integrierte Schaltung (ASIC) oder ein programmierbares Gatter-Array (PGA).

[0019] Obwohl diese Beschreibung auf spezielle Konfigurationen und Architekturen der verschiedenen Ebenen des Systems **10** Bezug nimmt, sollen zahlreiche Modifikationen und Variationen der beschriebenen und veranschaulichten Ausführungsbeispiele möglich sein.

[0020] Bei dem Ausführungsbeispiel gemäß Fig. 1 enthält das System **10** eine zentrale Verarbeitungseinheit (CPU) **100**, die über eine Host-Brückensteuereinrichtung **102** angekoppelt ist, die eine mit dem Hauptspeicher **104** gekoppelte Speichersteuereinrichtung **103** und eine mit einem Graphik-Controller **106** gekoppelte Graphikschnittstelle **105** enthalten kann. Die Graphikschnittstelle **105** kann beispielsweise eine Schnittstelle eines beschleunigten Graphik-Ports (AGP) gemäß der Accelerated Graphics Port Interface Specification, Revision **2.0**, veröffentlicht im Mai 1998, sein. Die Host-Brückensteuereinrichtung **102** kann außerdem eine Cache-Steuereinrichtung **107** zum Steuern eines sekundären (L2-)Cache-Speichers **109** enthalten. Die Host-Brückensteuereinrichtung **102** enthält eine Busschnittstelle **111**, die mit einem Systembus **112** gekoppelt ist, welcher bei einem Ausführungsbeispiel der Peripheriekomponentenverbindungs(PCI)-Bus sein kann, der gemäß der PCI Local Bus Specification, Production Version, Revision **2.1**, veröffentlicht im Juni 1995, arbeitet, oder bei alternativen Ausführungsbeispielen andere Arten von Schnittstellenprotokollen.

[0021] Beispielsweise kann bei einer anderen Konfiguration die Host-Brückensteuereinrichtung durch ein Speicher-Hub und ein Eingabe/Ausgabe-Hub, die durch eine Verbindung (Link) gekoppelt sind, ersetzt werden. Bei einer derartigen Konfiguration können Speicher und Graphikschnittstellenschaltung in dem Speicher-Hub und Brückensteuereinrichtungen in dem I/O-Hub sein.

[0022] Der Systembus **112** kann mit einer Speichereinrichtungssteuereinrichtung **114** gekoppelt sein, der den Zugriff auf eine oder mehrere Speichereinrichtungen, wie beispielsweise ein Festplattenlaufwerk **115** oder ein Compact-Disc-(CD) oder ein Digital-Video-Disc-(DVD)-Laufwerk **116**, steuert. Weitere Einrichtungen können ebenfalls mit dem Systembus **112** gekoppelt sein, wie beispielsweise eine Netz-

werkschnittstellenkarte und mit (nicht gezeigten) Peripheriegeräten gekoppelte Steckplätze. Das System gemäß den verschiedenen weiteren Integrationsebenen kann Steuereinrichtungen aufweisen, die in verschiedenen Blöcken implementiert sind. Beispielsweise können die Festplattenlaufwerks- und die CD- oder DVD-Laufwerkssteuereinrichtung in der Systembrücke **110** enthalten sein.

[0023] Das System **10** kann außerdem einen sekundären oder Erweiterungsbus **120** enthalten. Eine Systembrückensteuereinrichtung **110** ist zwischen dem Systembus **112** und dem Erweiterungsbus **120** eingekoppelt. Die Systembrückensteuereinrichtung **110** kann eine Systembusschnittstelle **113** enthalten, die mit dem Systembus **112** gekoppelt ist, und eine Erweiterungsbusschnittstelle **119**, die mit dem Erweiterungsbus **120** gekoppelt ist. Die Systembrückensteuereinrichtung kann außerdem eine Schnittstelle **117** eines universellen seriellen Busses (USB) enthalten, die mit einem USB-Port **118** gekoppelt ist, wie er in der Universal Bus Specification, Revision **1.0**, veröffentlicht im Januar 1996, beschrieben ist. Der Erweiterungsbus **120** kann mit verschiedenen Peripheriegeräten **122** und einem nichtflüchtigen Speicher **124** gekoppelt sein. Die mit den Bussen gekoppelten Komponenten und Einrichtungen, die Busse selbst und der Systemspeicher können Teil der Systemressourcen sein, die durch Anforderungen aus Arbeitseinheiten oder Ausführungsentitäten in dem System **10** benutzt werden bzw. auf die zugegriffen wird.

[0024] Es wird auf die Fig. 2A–Fig. 2B Bezug genommen, in denen verschiedene Software- und Hardwareebenen in dem System **10** detaillierter veranschaulicht sind. Beispielsweise kann das System **10** ein Betriebssystem (OS) **220** und Prozesse **222** und **224** enthalten. In der nachfolgenden Beschreibung sei angenommen, daß das Betriebssystem **220** ein thread-basiertes System ist, wie beispielsweise irgendein Windows-Betriebssystem, bei welchem jeder Prozeß einen oder mehrere Threads enthalten kann. Es ist jedoch klar, daß das Anforderungseinplanungsschema gemäß den beschriebenen Ausführungsbeispielen bei Betriebssystemen mit abweichend konfigurierten Ausführungsentitäten oder Arbeitseinheiten implementiert werden kann.

[0025] Wie es in Fig. 2A veranschaulicht ist, gehören die Threads **228** und **229** zu dem Prozeß **222** und die Threads **230** und **231** zu dem Prozeß **224**. Die Threads können mit dem OS **220** über eine zuvor definierte Schnittstelle kommunizieren, wie beispielsweise eine anwendungsprogrammierbare Schnittstelle (API), die unter dem Betriebssystem definiert sein kann. Alternativ kann eine "fremde" API verwendet werden. Das OS **220** enthält einen Einplaner (Scheduler) **232**, der Anforderungen aus den aktiven Threads über die vordefinierten Schnittstellen einplant. Bei einem Ausführungsbeispiel kann der Einplaner **232** einem Gerätetreiber **240** zugeordnet sein, der in der Lage ist, auf Speicher, I/O oder andere definierte Orte in dem System **10** zuzugreifen, um mit Hardwarekomponenten zu kommunizieren, um Einplanungsaktivitäten in Übereinstimmung mit Ausführungsbeispielen der Erfindung durchzuführen. Die Elemente **232** und **240** können gemeinsam ebenfalls als Einplaner bezeichnet werden. Bei weiteren Ausführungsbeispielen kann der Einplaner in mehrere Module oder Ebenen aufgeteilt sein.

[0026] Bei Empfang einer Anforderung aus einem Thread plant der Einplaner **232** die Anforderung auf der Grundlage einer rückkoppelnden Kommunikation aus den Hardwarekomponenten, der Anzahl der ausstehenden Anforderungen und der Verzögerungs-(Latenz-) und Bandbreitenerfordernisse des anfordernden Threads ein.

[0027] Gemäß einem Ausführungsbeispiel speichert der Einplaner **232** Anforderungen in eine Anforderungswarteschlange **204**, die eine vorgegebene Anzahl von Einträgen aufweist. Jedem Eintrag in der Anforderungswarteschlange **204** kann ein Status-Flag in einem Statusfeld **206** zugeordnet sein, das anzeigt, ob eine bestimmte Anforderung verarbeitet worden ist.

[0028] Ein Satz von Tabellen oder Tabellensegmenten **202**, der in dem Systemspeicher **104** (oder in irgendeinem anderen geeigneten Speicherort) gespeichert ist, auf den durch den Einplaner **232** über den Gerätetreiber **240** zugegriffen werden kann, identifiziert Kanalzuweisungen für zugehörige aktive Threads, die jeweils einen Thread-Identifizierer(-ID) aufweisen. Jeder Kanal ist so definiert, daß er eine vorgegebene Anzahl von Zyklen eines Basistaktes, wie beispielsweise eines von einem Taktgenerator **240** erzeugten Taktes, hat. Die Anzahl der Basistakte pro Kanal wird auf der Grundlage einer gewünschten Granularität ausgewählt. Jede der Tabellen oder Tabellensegmente **202** entspricht einer Ressource in dem System **10**, einschließlich beispielsweise des Systemspeichers **104**, der Graphikkarte **106**, des Systembusses **112**, des Erweiterungsbusses **120**, des USB-Ports **118** usw.

[0029] Gemäß einem Ausführungsbeispiel speichern die Brückensteuereinrichtungen **102**, **110** ebenfalls den verschiedenen Systemressourcen entsprechende Tabellen oder Tabellensegmente, die die Kanalzuweisungen für spezielle Threads durch Thread-IDs verfolgen. Die Tabellen oder Tabellensegmente in den Brückensteuereinrichtungen werden auf der Grundlage der entsprechenden Tabellen **202**, die von dem Betriebssystem **220** aufrechterhalten werden, geladen. Bei dem veranschaulichten Ausführungsbeispiel können die Tabellen oder Tabellensegmente **302A**, **302B** und **302C** durch die

Host-Brückensteuereinrichtung **102** gespeichert werden, um die Kanalzuweisungen für den Systemspeicher **104**, dem Systembus **112** und die Graphikkarte **106** zu verfolgen. Die Systembrückensteuereinrichtung **110** kann Tabellensegmente **302D** und **302E** speichern, um Kanalzuweisungen für den USB-Port **118** und den Erweiterungsbus **120** zu verfolgen. Weitere Tabellen können ebenfalls für weitere Systemressourcen gehalten werden.

[0030] Obwohl veranschaulicht ist, daß sie in den Brückensteuereinrichtungen **102**, **110** gespeichert sind, können die Tabellen **302A**–**302E** an anderen geeigneten Orten, wie beispielsweise in dem Systemspeicher **104** oder externen Speichereinrichtungen gespeichert werden. Alternativ können die verschiedenen Systemressourcen durch Steuereinrichtungen kontrolliert werden, die über das System verteilt sind, statt in den Brückensteuereinrichtungen **102**, **110** integriert zu sein, wie es veranschaulicht ist.

[0031] Die Tabellen **302A**–**302E** können periodisch durch das Betriebssystem **220** aktualisiert werden, wenn sich die Kanalzuweisungen ändern. In jeder der Tabellen **302A**–**302E** können die Threads der gleichen oder einer abweichenden Anzahl von Kanälen zugewiesen sein. Beispielsweise kann ein einem ersten Thread-ID zugeordneter Thread einer ersten Anzahl von Kanälen und ein einem zweiten Thread-ID zugeordneter Thread einer abweichenden Anzahl von Kanälen zugewiesen sein.

[0032] Die den Threads in den Tabellen **302A**–**302E** zugewiesenen Kanäle definieren Thread-Anforderungsausführungsfenster oder -Slots innerhalb eines Gesamteinplanungszyklus **400**, wie er in Fig. 3 veranschaulicht ist. Der Einplanungszyklus **400** enthält mehrere Thread-Anforderungsausführungsfenster oder -Slots $\mathbf{402_0}$, $\mathbf{402_1}$, ..., $\mathbf{402_{N-1}}$ und $\mathbf{402_N}$, die jeweils eine zugewiesene Anzahl von Kanälen umfassen. Jedes Anforderungsausführungsfenster **402** ist der Ausführung von Anforderungen aus einem Thread zugewiesen.

[0033] Bei einem Ausführungsbeispiel ist das erste Fenster **402** dem Einplaner **232** zum Aufrechterhalten der Kohärenz zwischen den OS-Tabellen **202** und entsprechenden Tabellen **302** in den Brückensteuereinrichtungen **102**, **110** zugewiesen. Die verbleibenden Fenster $\mathbf{402_1}$ bis $\mathbf{402_N}$ können den Anforderungen aus verschiedenen weiteren Threads zugewiesen sein. Bei dem veranschaulichten Ausführungsbeispiel werden die Tabellen **302A**–**302E** jeweils einmal in jedem Einplanungszyklus **402** in dem Kohärenzfenster $\mathbf{402_0}$ von dem Einplanergerätetreiber **240** aktualisiert. Während der Zeitdauer des Kohärenzfensters **402** oder alternativ in einem anderen Fenster $\mathbf{402_i}$ kann der Einplanergerätetreiber **240** außerdem Inhalte der Statusregister **304A**–**304E** in den Brückensteuereinrichtungen **102**, **110** lesen, um festzustellen, welche Anforderungen abgeschlossen sind. Dem Einplaner **232** wird somit eine Rückkopplung darüber zur Verfügung gestellt, welche Anforderungen abgeschlossen sind und welche noch anhängig sind, um ihm zu ermöglichen zu verfolgen, welche Systemressourcen verfügbar sind. Auf diese Weise kann dann, wenn eine Anforderung aus einem Thread von dem Einplaner **232** empfangen wird, der Einplaner **232** bestimmen, ob ausreichende Ressourcen verfügbar sind, um die Anforderung zu verarbeiten.

[0034] Es wird wieder auf die Fig. 2A–Fig. 2B Bezug genommen; gemäß einem Ausführungsbeispiel enthält jede der Brückensteuereinrichtungen **102**, **110** verschiedene Warteschlangen zum Speichern von Anforderungen für verschiedene Ressourcen im System **10**. Beispielsweise enthält die Host-Brückensteuereinrichtung **102** eine Speicherwarteschlange **310** in der Speichersteuereinrichtung **103**. von der Speichersteuereinrichtung **103** aus verschiedenen Quellen in dem System **10** empfangene Anforderungen, einschließlich solcher von der CPU **100** über eine CPU-Busschnittstelle **312** und von Einrichtungen an dem Systembus **112** über die Systembusschnittstelle **111** empfangene, werden in der Speicherwarteschlange **310** zur Ausführung gespeichert. Zusätzlich zu den Speicheradreß- und -dateninformationen ist die Speicherwarteschlange **310** darüber hinaus so ausgebildet, daß sie den zugeordneten Thread-ID einer Speicheranforderung speichert.

[0035] Bei einigen Ausführungsbeispielen der Erfindung ruft die CPU **100** einen Befehl zusammen mit den zugehörigen Thread-ID ab. Der Thread-ID wird weitergeleitet und in den Warteschlangen der Brückensteuereinrichtungen **102**, **110** gespeichert.

[0036] Eine Einplanersteuereinrichtung **314** in der Host-Brückensteuereinrichtung **102** empfängt Ausgabewerte aus einem Zähler **306** und den Tabellen oder Tabellensegmenten **302A**–**302C**. Der Zähler **306**, welcher von dem Grundtakt aus dem Taktgenerator **250** getaktet werden kann, kann so ausgebildet sein, daß er die Kanäle in dem Einplanungszyklus **400** durchzählt. Ausgehend von dem wert des Zählers **306** kann die Einplanungssteuereinrichtung **314** das aktuelle Thread-Anforderungsausführungsfenster $\mathbf{400_i}$ (i = 0 bis N) in dem Einplanungszyklus **400** feststellen. In Abhängigkeit davon, welches Fenster $\mathbf{400_i}$ aktiv ist, und in Abhängigkeit von den Kanalzuweisungen in der Tabelle **302A** kann eine dem zugehörigen Thread-ID in der Speicherwarteschlange **310** zugeordnete Anforderung für die Verarbeitung durch die Speichersteuereinrichtung **103** ausgewählt werden.

[0037] Die ausgewählte Anforderung wird innerhalb des aktuellen Fensters $\mathbf{402_i}$ ausgeführt. Bei Abschluß der Anforderung kann die Host-Brückensteuerein-

richtung **102** einen Anforderung-abgeschlossen-Status zurückgeben, indem sie geeignete Bits in dem Statusregister **304A** programmiert. In dem nächsten Kohärenzfenster $\mathbf{402}_0$ liest die CPU **100** unter der Steuerung des Einplanergerätetreibers **140** das Statusregister **304A** (sowie die anderen Statusregister **304B**–**304E**), um zu bestimmen, welche Anforderungen abgeschlossen sind. Der Einplanergerätetreiber **240** aktualisiert dann Flags **206** der abgeschlossenen Anforderungen in der Anforderungswarteschlange **204**.

[0038] Zusätzlich zu der Speicherwarteschlange **310** kann die Host-Brückensteuereinrichtung **102** bei einem Ausführungsbeispiel außerdem eine Systembuswarteschlange **316** in der Systembusschnittstelle **111** und eine Graphikkartenanforderungswarteschlange **318** in der Graphikschnittstelle **105** enthalten. Anforderungen an den Systembus **112** werden in die Systembuswarteschlange **316** eingegeben, während Anforderungen der Graphikkarte **106** in die Graphikkartenwarteschlange **318** eingegeben werden. Die Anforderungen sind den Thread-IDs zugeordnet, so daß die Einplanersteuereinrichtung **314** die richtige Anforderung für eine Verarbeitung durch die Busschnittstelle **111** oder die Graphikschnittstelle **105** auf der Grundlage des aktuellen Thread-Anforderungsausführungsfensters $\mathbf{402}_i$ und der in den Tabellen **302B** und **302C** gespeicherten Kanalzuweisungen auswählen kann. Der Abschluß von Anforderungen in den Warteschlangen **316** und **318** wird durch die Statusregister **304B** bzw. **304C** angezeigt.

[0039] In ähnlicher Weise enthält die Systembrückensteuereinrichtung **110** eine Erweiterungsbuswarteschlange **320** zum Speichern von an den Erweiterungsbus **120** gerichteten Anforderungen und eine USB-Buswarteschlange **322** zum Speichern von an den USB-Port **118** gerichteten Anforderungen. Ein Zähler **308**, welcher durch den Grundtakt aus dem Taktgenerator **250** getaktet werden kann, zählt die Kanäle des Einplanungszyklus **400** durch. Auf der Grundlage des aktuellen Thread-Anforderungsausführungsfensters $\mathbf{402}_i$, wie es vom Zähler **308** angezeigt wird, und der in den Tabellen **302D** und **302E** gespeicherten Kanalzuweisungen bestimmt eine Einplansteuereinrichtung **324** in der Systembrückensteuereinrichtung **110**, welche den Anforderungen in den Warteschlangen **320** und **322** verarbeitet werden sollen. Der Abschluß der Anforderungen wird durch die Statusregister **304D** und **304E** angezeigt.

[0040] Es wird auf Fig. 4 Bezug genommen; der Einplanergerätetreiber **240**, der in Kooperation mit dem Einplaner **232** arbeitet, wartet auf den Empfang bestimmter Ereignisse (bei **502**).

[0041] Wenn eine Anforderung aus einem Thread empfangen wird, welche in Form eines anwendungs-programmierbare-Schnittstelle(API)-Aufrufs vorliegen kann, greift (bei **504**) der Einplanergerätetreiber **240** auf die Anforderungswarteschlange **204** und die Kanalzuweisungstabellen **202** zu, so daß der Einplaner **232** (bei **506**) bestimmen kann, ob Ressourcen verfügbar sind, um die Thread-Anforderung verarbeiten zu können.

[0042] Wenn der Einplaner **232** feststellt, daß keine Ressourcen verfügbar sind, um die Anforderung ausreichend zu verarbeiten, so wird der anfordernde Thread benachrichtigt (bei **508**). In Erwiderung der Benachrichtigung kann der Thread für eine Zeitdauer warten, bevor er die Anforderung erneut ausgibt, oder der Thread kann auf eine andere elegante Weise die Situation behandeln. Wenn die angeforderten Ressourcen verfügbar werden, dann wird die Anforderung aus dem Thread (bei **510**) der Anforderungswarteschlange **204** hinzugefügt.

[0043] Wenn eine bestimmte Anforderung in der Anforderungswarteschlange **204** verarbeitet und abgeschlossen ist, kann ein Abgeschlossen-Flag in dem Flag-Feld **206** der Warteschlange **204** gesetzt werden, wie es oben erörtert wurde.

[0044] Der Einplanergerätetreiber **240** selbst ist ein Thread, der in der Lage ist, Anforderungen auszugeben, wie beispielsweise solche zum Zugreifen auf Speicherorte, die der Anforderungswarteschlange und den Thread-Kanalzuweisungstabellen entsprechen. Anforderungen aus dem Einplanergerätetreiber-Thread, die in die Anforderungswarteschlange **204** (bei **520**) eingegeben worden sind, können in dem ersten Fenster $\mathbf{402}_0$ des Einplanungszyklus **400** verarbeitet werden. Alternativ könnte ein anderes Fenster $\mathbf{402}_i$ dem Einplanergerätetreiber **240** zugewiesen sein.

[0045] Während des Kohärenzfensters $\mathbf{402}_0$ aktualisiert die CPU **100** unter der Steuerung des Einplanergerätetreibers **240** (bei **522**) die Brückensteuereinrichtungstabellen **302A**–**302E**, wie es erforderlich ist, um die Kanalzuweisungen zu ändern. Die CPU **100** kann darüber hinaus (bei **524**) auf die Statusregister **304A**–**304E** zugreifen, um zu bestimmen, welche Anforderungen abgeschlossen sind. Alternativ können die Vorgänge des Tabellenaktualisierens und des Statusregisterlesens getrennt werden. Die CPU **100** kann dann unter Steuerung des Gerätetreibers **240** (bei **526**) die Anforderungswarteschlange **204** aktualisieren.

[0046] Der Einplaner **232** kann feststellen, welche Systemressourcen durch eine Anforderung benötigt werden, indem er die Anforderung selbst und die Anforderungsparameter, wie beispielsweise die Parameter in einem API-Aufruf, betrachtet. Ein Parameter könnte beispielsweise einen Zugriff auf einen Ort im Speicheradreßraum in dem Systemspeicher **104** spezifizieren. Ein weiterer Parameter könnte einen

Ort im I/O-Adreßraum spezifizieren, welcher in einem der Busse **112**, **120**, in der Graphikkarte **106** auf dem USB-Bus, der mit dem USB-Port **118** gekoppelt ist, oder einem anderen Ort in dem System **10** lokalisiert sein. Auf der Grundlage der angeforderten Ressourcen, der bereits in der Warteschlange **204** befindlichen Anforderungen und den in den Tabellen oder Tabellensegmenten **202** spezifizierten Kanalzuweisungen, wie sie durch den Einplanergerätetreiber **240** gelesen worden sind, kann der Einplanergerätetreiber **232** bestimmen, ob eine Anforderung auf irgendeine vernünftige Weise verarbeitet werden kann. Dies kann in Übereinstimmung mit Kriterien definiert werden, die zuvor in das System **10** programmiert und durch eine Startroutine (z.B. eine Basis-Eingabe/Ausgabe-System- oder BIOS-Routine) geladen wurden. Der Einplaner **232** kennt die Verzögerungs-(Latenz-) und Bandbreitenerfordernisse der verschiedenen Threads in dem System. Die Latenz- und Bandbreitenanforderungen können von dem Einplaner **232** verwendet werden, um zu bestimmen, ob ausreichende Ressourcen verfügbar sind, um eine Thread-Anforderung zu befriedigen.

[0047] Beispielsweise könnte ein Thread eine Anforderung zum Übertragen eines Einzelbilds von Videodaten aus dem Videospeicher in der Graphikkarte **106** an den Systemspeicher **104** ausgeben. Bei einer gegebenen Einzelbildgröße (z.B. 420 × 480 Pixel) und einer für jedes Pixel vorgegebenen Bitanzahl können die Bandbreitenerfordernisse der Videoübertragung auf der Grundlage einer spezifizierten Zeitperiode, in welcher die Übertragung auftreten soll, bestimmt werden. Zusätzlich kann die Verzögerung der Übertragungsanforderung ebenfalls bekannt sein. Auf der Grundlage der Bandbreiten- und Verzögerungsinformationen und auf der Grundlage der ausstehenden Anforderungen nach Ressourcen kann der Einplaner bestimmen, ob die Videoübertragungsanforderung durch verfügbare Ressourcen verarbeitet werden kann. Wenn nicht, wird der Thread von dem Einplaner informiert, und der Thread kann auf verschiedene weise reagieren, beispielsweise warten, bevor eine weitere Anforderung ausgegeben wird, oder eine Anforderung in verschiedene Teile unterteilen.

[0048] Es wird auf Fig. 5 Bezug genommen; die Systeminitialisierung wird von einer System-BIOS-Routine in Übereinstimmung mit einem Ausführungsbeispiel durchgeführt. Nachdem ein Systemrücksetzen die System-Hardware in einen Anfangszustand gebracht hat, startet die CPU **100** mit der Ausführung von Befehlen in der Einschaltselbsttest(POST)-Prozedur des BIOS, welche für die Initialisierung von Komponenten in dem System **10** in bekannte Zustände und zum Aufbauen der Systemkonfigurationsinformationen für eine Verwendung durch das OS **220** verantwortlich ist. Nachdem irgendwelche Initialisierungsaufgaben in dem System **10** (bei **602**) durchgeführt sind, richtet die BIOS-Routine als nächstes (bei **604**) den Systemspeicher **104** oder einen anderen geeigneten Speicherort zum Speichern der Kanalzuweisungstabellen oder Tabellensegmente **202** ein. Die BIOS-Routine kann die Anzahl der Basistakte für jeden Kanal (z.B. einen oder mehrere Takte pro Kanal) und die Gesamtbreite des Einplanungszyklus **400** spezifizieren. Es können spezielle Speicheradressen reserviert werden, um die Tabellen oder Tabellensegmente **202** zu speichern. Zusätzlich können Kriterien spezifiziert werden für eine Verwendung durch den Einplaner **232** beim Bestimmen, ob eine Thread-Anforderung akzeptiert oder zurückgewiesen werden soll.

[0049] Als nächstes können Standardkanalzuweisungen (bei **606**) in die Tabellen geladen werden. Beispielsweise könnten bestimmte dem OS zugewiesene Threads (wie beispielsweise der Einplaner oder andere Systemmanagementebenen) den Fenstern in dem Zyklus **400** zugewiesen werden. Zusätzlich kann das BIOS Standardfenster einrichten, die eine vorgegebene Anzahl von Kanälen aufweisen, um Thread-Anforderungen zu behandeln, die nicht speziellen Fenstern zugewiesen worden sind. Das BIOS kann den Konfigurationsraum des Systems abfragen, um den Typ des verfügbaren Prozessors zu bestimmen, sofern der Prozessor ein Multi-Processing-System ist, und weitere Informationen. Aus den Informationen kann das BIOS die Fähigkeiten des Systems bestimmen. Auf der Grundlage der bestimmten Fähigkeiten kann das BIOS die Anzahl der Kanäle in den Standardfenstern entsprechend zuweisen. Als nächstes werden die Systemkomponenten durch die BIOS-Routine (bei **608**) initialisiert und konfiguriert. Das OS **220** wird dann gebootet (bei **610**).

[0050] Es wird auf Fig. 6 Bezug genommen; nachdem das OS **220** gebootet ist, identifiziert es zunächst (bei **650**) die Anzahl der aktiven Threads in dem System **10**. Das OS **220** fragt (bei **652**) jeden Thread nach seinen Bandbreiten- und Verzögerungserfordernissen für Ressourcen in dem System **10** ab. Einige Threads wissen, welches ihre Verzögerungs- und Bandbreitenerfordernisse für jede Systemressource sind. Beispielsweise kann ein einem Multimedia-Prozeß zugeordneter Thread "Echtzeit"-Erfordernisse aufweisen, die möglicherweise eine relativ geringe Verzögerung für Datenübertragungen tolerieren und die einen hohen Datenübertragungsdurchsatz erfordern. Andere Threads könnten in der Lage sein, höhere Verzögerungen und geringere Datenübertragungsbandbreiten zu tolerieren. Auf der Grundlage eines Vergleichs der verschiedenen Verzögerungs- und Bandbreitenerfordernisse aus den aktiven Threads kann die Anzahl der Kanäle zum Zuweisen der verschiedenen Anforderungsfenster $\mathbf{402}_i$, die den verschiedenen Threads entsprechen, von dem OS (bei **654**) so eingestellt werden, daß es die geringere Verzögerungen und höhere Bandbreiten erfordern-

den Threads favorisiert. Im Ergebnis könnten solchen Thread-Arten eine größere Anzahl von Kanälen und möglicherweise mehrere Fenster **402**$_i$ zugewiesen werden. Die mehreren zugewiesenen Fenster **402**$_i$ können aufeinanderfolgend oder in dem Einplanungszyklus **400** verteilt sein. Sofern ein Thread keine Verzögerungs- und Bandbreiteninformationen für irgendeine Systemressource zur Verfügung stellt, so kann dieser Thread den Standardfenstern **402**$_i$ in dem Einplanungszyklus **400** der Tabellen oder Tabellensegmente **202** zugewiesen werden.

[0051] Auf der Grundlage der berechneten Anzahl der Kanäle können die Tabellen oder Tabellensegmente **202** (bei **656**) mit den Kanalzuweisungen gemäß den Thread-IDs geladen werden.

[0052] Somit plant in Übereinstimmung mit einigen Ausführungsbeispielen ein Einplanschema Anforderungen aus Thread in dem System ein, indem es auf der Grundlage der Verfügbarkeit von Ressourcen und von Kanalzuweisungen bestimmt, ob Datenflußerfordernisse befriedigt werden können. Die Verfügbarkeit der Ressourcen wird durch Hardware-Komponenten einem Einplaner in dem System angezeigt. Darüber hinaus ist der Einplaner auf der Grundlage des Typs der Anforderung und von Anforderungsparametern in der Lage zu bestimmen, welche Ressourcen von der speziellen Anforderung benötigt werden.

[0053] Weitere Ausführungsbeispiele liegen innerhalb des Umfangs der nachfolgenden Patentansprüche. Beispielsweise können bei anderen Betriebssystemen die Basisarbeitseinheiten oder Ausführungsentitäten in dem System nicht Threads sondern Prozesse oder andere definierte Einheiten sein. Darüber hinaus können die Hardwarekomponenten in dem System abweichend konfiguriert sein. Die von den veranschaulichten Modulen und Ebenen der Software und Firmware ausgeführten Aktivitäten können variieren.

Patentansprüche

1. Verfahren zum zeitlichen Einplanen von Ressourcen-Anforderungen aus aktiven Ausführungsentitäten, die Tasks, Prozesse und/oder Threads umfassen, wobei die Ressourcen-Anforderungen jeweils eine von mehreren Ressourcen eines Computersystems anfordern, wobei das Computersystem wenigstens einen mit einer CPU gekoppelten Controller (**102**, **110**) aufweist, wobei jede Ressource (**104**, **106**, **112**, **118**, **120**) mit einem zugehörigen Controller (**102**, **110**) gekoppelt ist, wobei:
a) jeder aktiven Ausführungsentität für jede der Ressourcen jeweils wenigstens ein Zeitfenster (**402**) innerhalb eines Einplanzyklus (**400**) zugewiesen wird, wobei die Zeitfenster (**402**) von einem auf der CPU laufenden Betriebssystem (**220**) zugewiesen werden, wobei die Zeitfenster (**402**) jeweils aus einer Anzahl von Zeit-Slots bestehen, wobei die Zuweisung der Zeitfenster in einer Mehrzahl von Tabellen (**202**) gespeichert wird, wobei jeweils eine Tabelle (**202A**–**202E**) einer Ressource zugeordnet ist, wobei die Tabellen (**202A**–**202E**) Zeit-Slot-Zuweisungen speichern, in denen jedem Zeit-Slot ein Identifizierer einer Ausführungsentität zugeordnet ist;
b) wobei ein Scheduler (**232**) des Betriebssystems (**220**) jeweils eine Ressourcen-Anforderung empfängt, dann aufgrund der Zeit-Slot-Zuweisungen für die jeweilige Ausführungsentität und die jeweilige Ressource und aufgrund der angeforderten Ressource sowie aufgrund von vorangegangenen anhängigen Ressourcen-Anforderungen bestimmt, ob die Ressourcen-Anforderung verarbeitet werden kann, und dann, wenn dies der Fall ist, die Ressourcen-Anforderung in einer Anforderungswarteschlange (**204**) speichert;
c) wobei die Zeit-Slot-Zuweisungen enthaltenden Tabellen sowie die Ressourcen-Anforderungen für jede Ressource periodisch von dem Betriebssystem in Tabellen (**302A**–E) bzw. Anforderungs-Warteschlangen (**310**, **316**, **318**, **320**, **322**) in dem der Ressource zugeordneten Controller (**102**, **110**) übertragen werden; und
d) die in den Anforderungs-Warteschlangen (**310**, **316**, **318**, **320**, **322**) des wenigstens einen Controllers enthaltenen Ressourcen-Anforderungen in Abhängigkeit von den Zeit-Slot-Zuweisungen der zugehörigen Tabellen (**302A**–E) abgearbeitet werden.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß im Schritt b) dann, wenn der Scheduler bestimmt, daß die Ressourcen-Anforderung nicht verarbeitet werden kann, die anfordernde aktive Ausführungsentität benachrichtigt wird.

3. Verfahren nach Anspruch 2, dadurch gekennzeichnet, daß dann, wenn die aktive Ausführungsentität benachrichtigt wurde, daß die Ressourcen-Anforderung nicht verarbeitet werden kann, die Ressourcen-Anforderung, nachdem eine bestimmte Zeitdauer gewartet wurde, erneut ausgegeben wird.

4. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß im Schritt d) die Ressourcen-Anforderungen dann abgearbeitet werden, wenn das aktuelle Zeitfenster (**402**) derjenigen Ausführungsentität zugewiesen ist, aus der die Ressourcen-Anforderung herrührt.

5. Verfahren nach Anspruch 4, dadurch gekennzeichnet, daß das aktuelle Zeitfenster von einem mit dem Controller (**314**, **324**) der Ressource gekoppelten Zähler (**306**, **308**) angezeigt wird.

6. Verfahren nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß die Anzahl der Zeit-Slots, die als Zeitfenster den Ausführungsentitä-

ten im Schritt a) jeweils zugewiesen werden, in Abhängigkeit von den verschiedenen Verzögerungs- und/oder Bandbreitenerfordernissen der Ausführungsentität für die jeweilige Ressource bestimmt wird.

7. Verfahren nach Anspruch 6, dadurch gekennzeichnet, daß die Bandbreiten- und/oder Verzögerungserfordernisse für die Ressourcen von den Ausführungsentitäten an den Scheduler übermittelt werden.

8. Verfahren nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, daß der Scheduler, bevor er im Schritt b) bestimmt, ob eine Ressourcen-Anforderung verarbeitet werden kann, die Inhalte von Statusregistern (**304**) liest, um zu bestimmen, welche der vorangegangenen Ressourcen-Anforderungen anhängig und/oder abgeschlossen sind, um so die Verfügbarkeit der Ressourcen zu ermitteln.

9. Ein Computersystem zum Ausführen eines Verfahrens zum zeitlichen Einplanen von Ressourcen-Anforderungen aus aktiven Ausführungsentitäten gemäß einem der Ansprüche 1 bis 8, wobei das Computersystem aufweist:
eine CPU, wobei die CPU die Ausführungsentitäten und ein Betriebssystem mit einem Einplaner (scheduler) ausführt,
eine Mehrzahl von Ressourcen, und
wenigstens einen mit der CPU gekoppelten Controller, wobei jede Ressource mit einem zugehörigen Controller gekoppelt ist,
wobei der Controller Speicherplätze zum Speichern von Zeit-Slot-Zuordnungen aufweist, die Anforderungen von den mit dem Controller gekoppelten Ausführungsentitäten zugeordnet sind, wobei der Controller zum Zugreifen auf die Speicherplätze ausgebildet ist, und um die Anforderungen für die Ressource gemäß den Slot-Zuordnungen zu verarbeiten.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

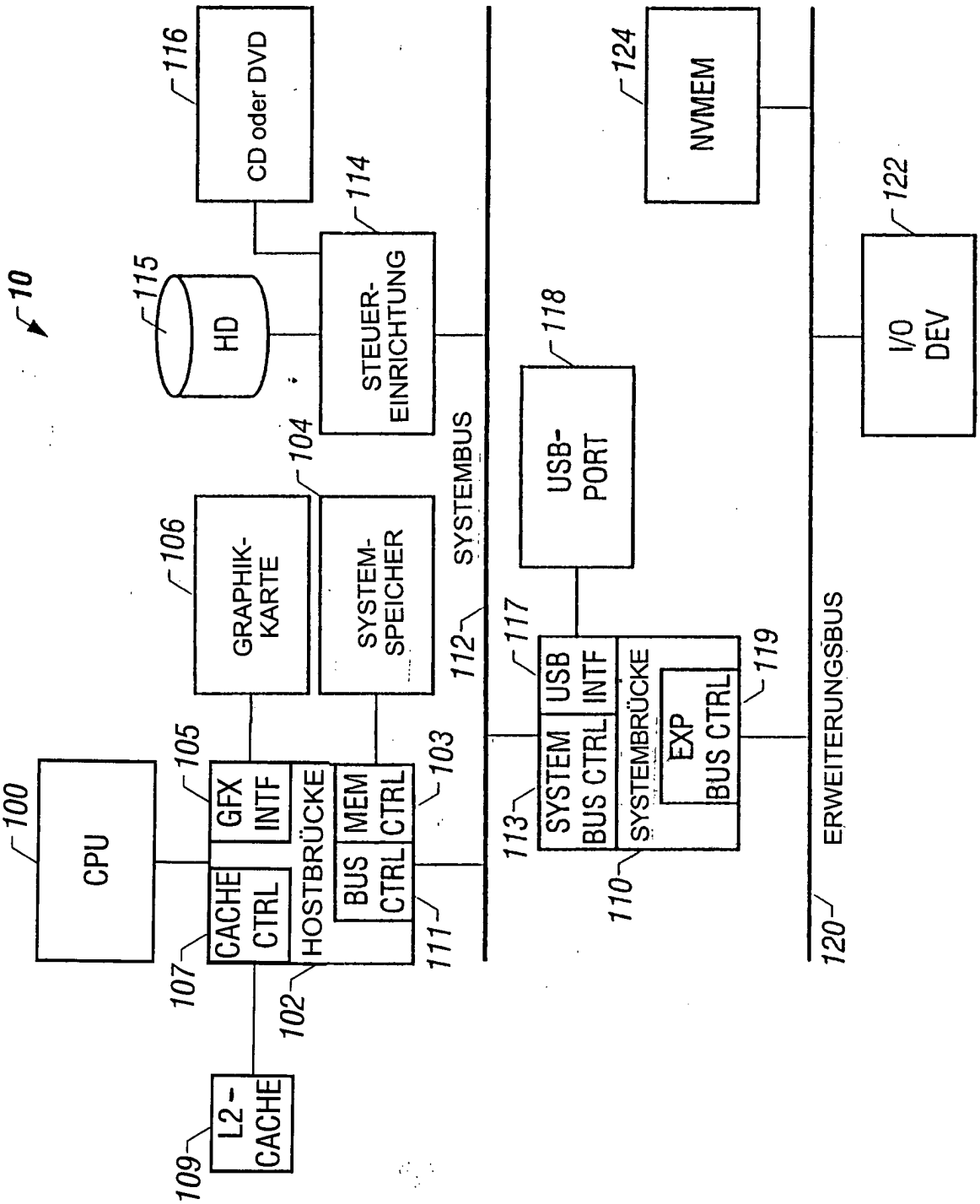


FIG. 1

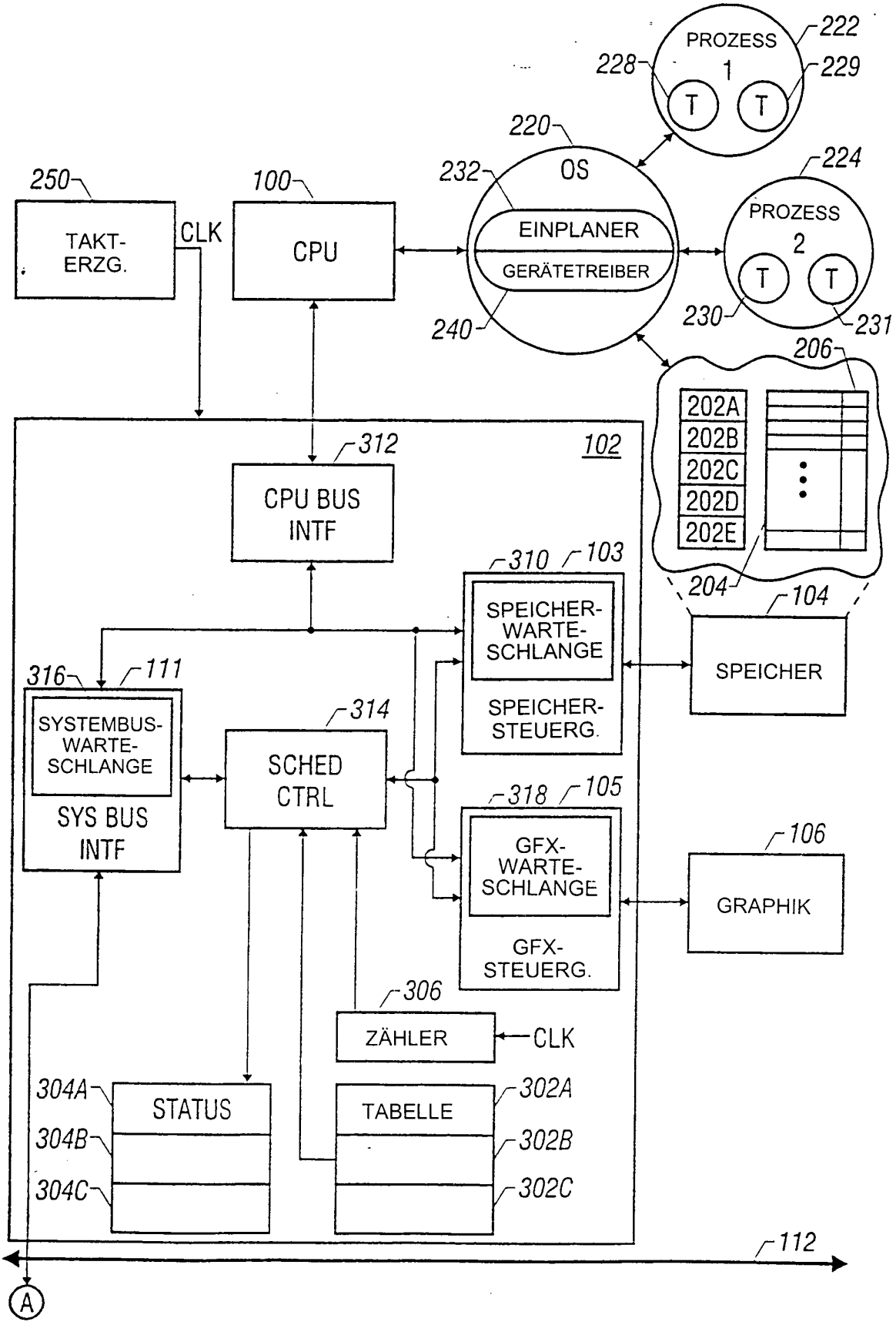

PROZESS 1
222
228
229
T
T
220
OS
232
EINPLANER
GERÄTETREIBER
240
224
PROZESS 2
T
T
230
231
206
202A
202B
202C
202D
202E
204
250
TAKT-ERZG.
CLK
100
CPU
102
312
CPU BUS INTF
310
103
SPEICHER-WARTE-SCHLANGE
SPEICHER-STEUERG.
104
SPEICHER
316
111
SYSTEMBUS-WARTE-SCHLANGE
SYS BUS INTF
314
SCHED CTRL
318
105
GFX-WARTE-SCHLANGE
GFX-STEUERG.
106
GRAPHIK
306
ZÄHLER
CLK
304A
304B
304C
STATUS
TABELLE
302A
302B
302C
112
A

FIG. 2A

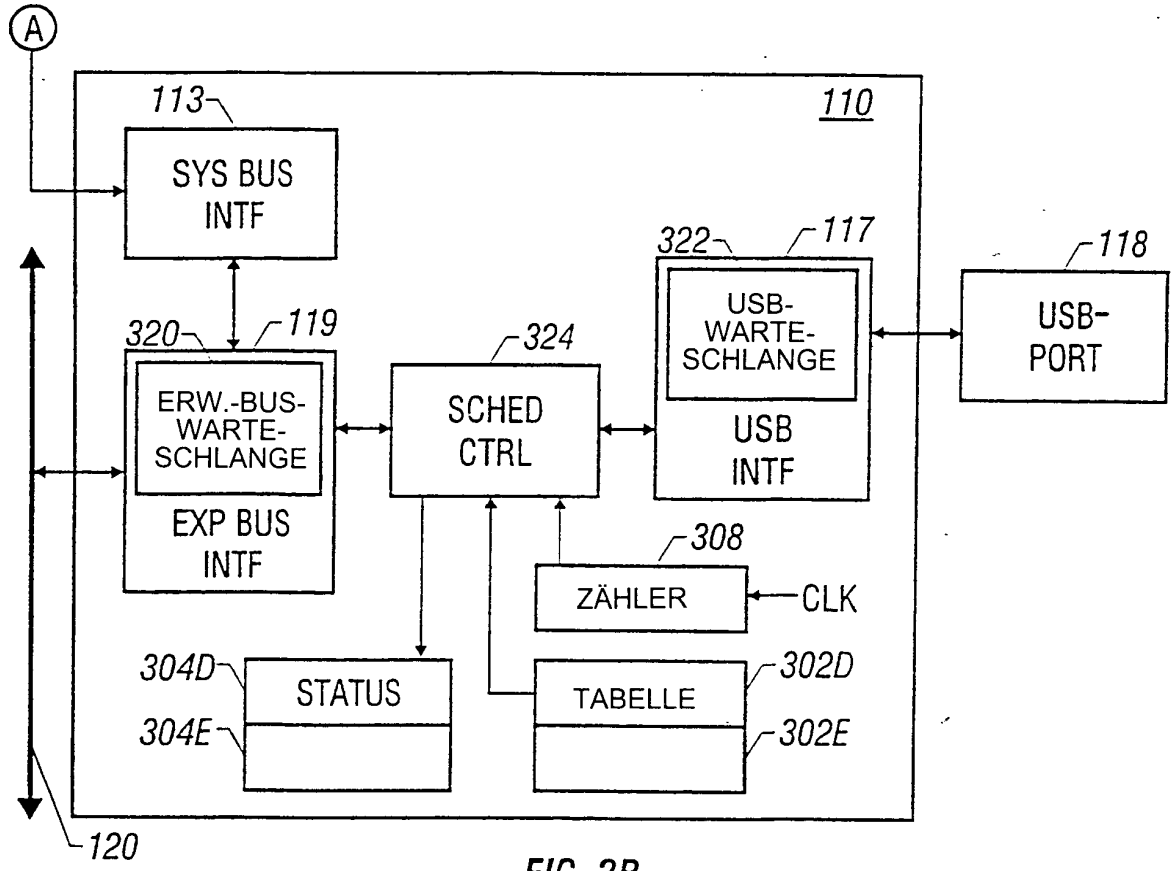

A
113
110
SYS BUS INTF
322
117
118
USB-WARTE-SCHLANGE
USB-PORT
320
119
324
ERW.-BUS-WARTE-SCHLANGE
SCHED CTRL
USB INTF
EXP BUS INTF
308
ZÄHLER
CLK
304D
STATUS
TABELLE
302D
304E
302E
120

FIG. 2B

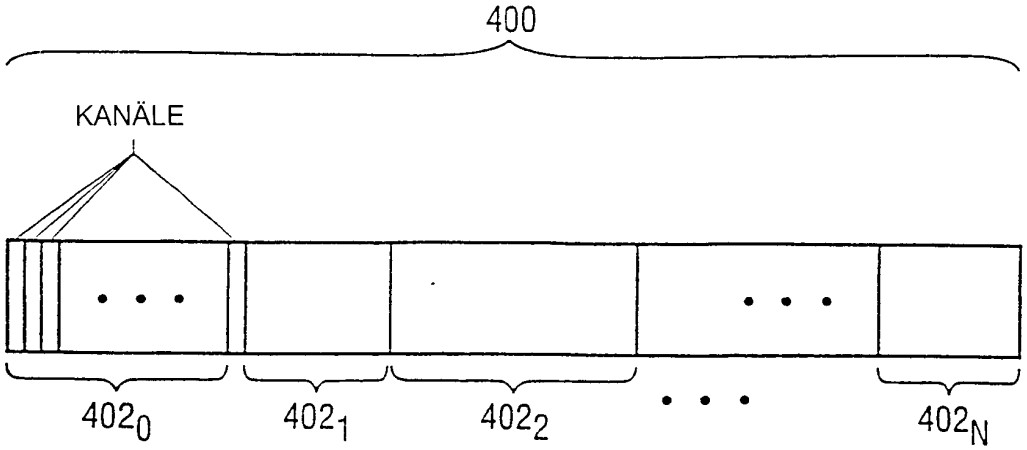

400
KANÄLE
402_0
402_1
402_2
402_N

FIG. 3

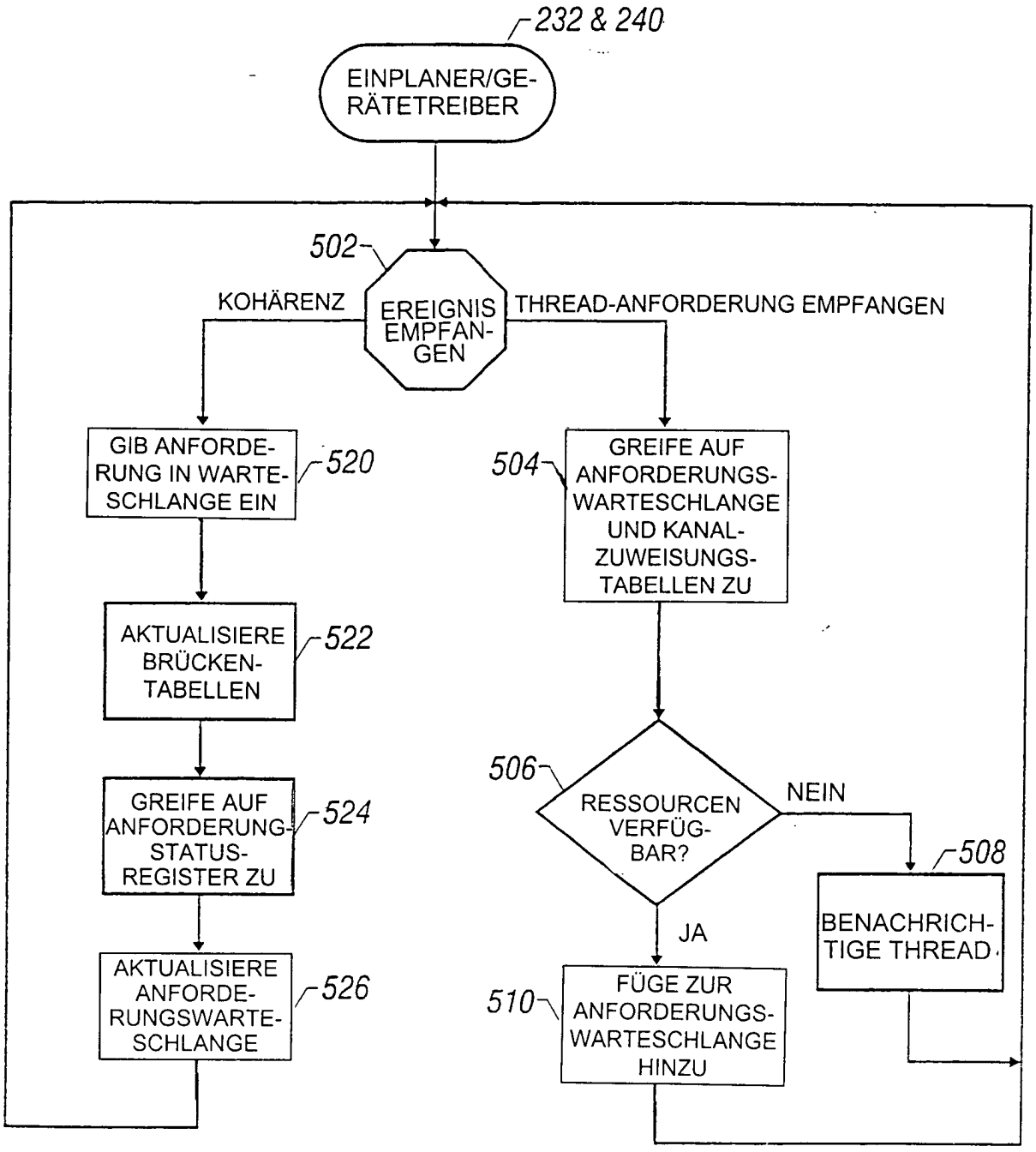

232 & 240
EINPLANER/GE-
RÄTETREIBER
502
KOHÄRENZ
EREIGNIS
EMPFAN-
GEN
THREAD-ANFORDERUNG EMPFANGEN
GIB ANFORDE-
RUNG IN WARTE-
SCHLANGE EIN
520
AKTUALISIERE
BRÜCKEN-
TABELLEN
522
GREIFE AUF
ANFORDERUNG-
STATUS-
REGISTER ZU
524
AKTUALISIERE
ANFORDE-
RUNGSWARTE-
SCHLANGE
526
504
GREIFE AUF
ANFORDERUNGS-
WARTESCHLANGE
UND KANAL-
ZUWEISUNGS-
TABELLEN ZU
506
RESSOURCEN
VERFÜG-
BAR?
NEIN
JA
508
BENACHRICH-
TIGE THREAD
510
FÜGE ZUR
ANFORDERUNGS-
WARTESCHLANGE
HINZU

FIG. 4

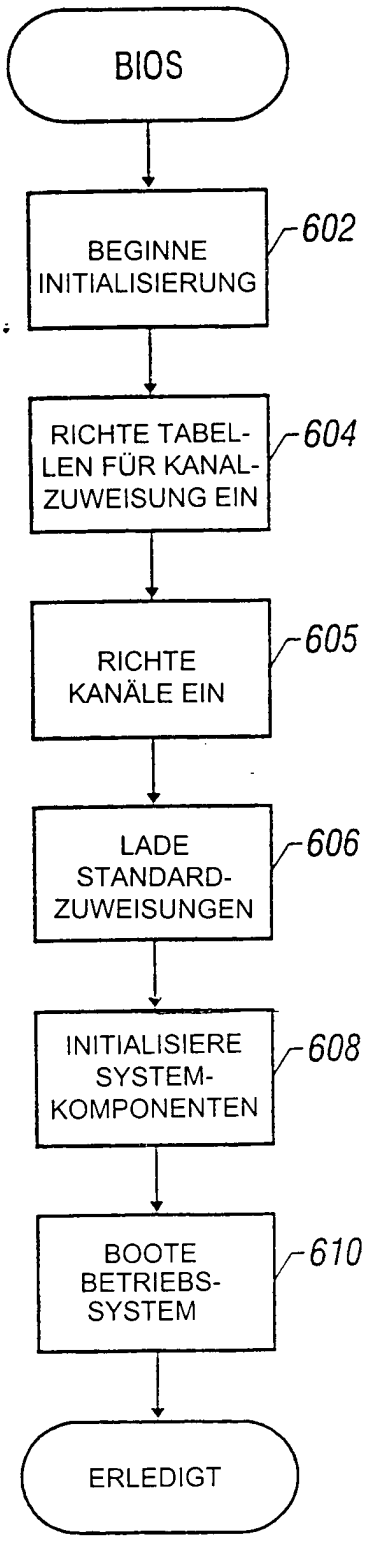

BIOS
BEGINNE INITIALISIERUNG
602
RICHTE TABELLEN FÜR KANALZUWEISUNG EIN
604
RICHTE KANÄLE EIN
605
LADE STANDARDZUWEISUNGEN
606
INITIALISIERE SYSTEMKOMPONENTEN
608
BOOTE BETRIEBSSYSTEM
610
ERLEDIGT

FIG. 5

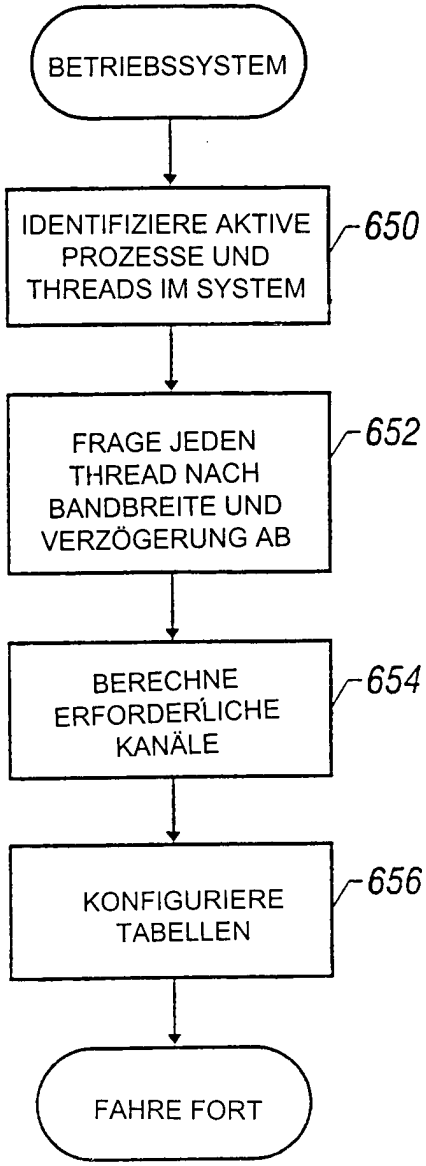

BETRIEBSSYSTEM
IDENTIFIZIERE AKTIVE PROZESSE UND THREADS IM SYSTEM
650
FRAGE JEDEN THREAD NACH BANDBREITE UND VERZÖGERUNG AB
652
BERECHNE ERFORDERLICHE KANÄLE
654
KONFIGURIERE TABELLEN
656
FAHRE FORT

FIG. 6