(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 60 2004 000 325 T2** 2006.08.10

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 528 477 B1**
(21) Deutsches Aktenzeichen: **60 2004 000 325.5**
(96) Europäisches Aktenzeichen: **04 255 941.9**
(96) Europäischer Anmeldetag: **29.09.2004**
(97) Erstveröffentlichung durch das EPA: **04.05.2005**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **11.01.2006**
(47) Veröffentlichungstag im Patentblatt: **10.08.2006**

(51) Int Cl.[8]: ***G06F 13/16*** (2006.01)

(30) Unionspriorität:
699315 **31.10.2003** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Lucent Technologies Inc., Murray Hill, N.J., US

(72) Erfinder:
Zievers, Peter J., Naperville, IL 60540, US

(74) Vertreter:
derzeit kein Vertreter bestellt

(54) Bezeichnung: **Speicherverwaltungssystem zur Verarbeitung von verketteten Listen**

Beschreibung

Gebiet der Erfindung

[0001] Diese Erfindung betrifft Speichermanagementeinrichtungen für ein Kommunikationsnetz und insbesondere Einrichtungen, die Verkehrsbedienungsfähigkeiten des Netzes durch den Abbau von Staus an den Netzknoten optimieren. Diese Erfindung betrifft des Weiteren Speichermanagementeinrichtungen, die den Verkehrsfluss verbessern, indem sie die Konkurrenzzeiten für den Zugriff auf Netzspeicherelemente verkürzen. Diese Erfindung betrifft des Weiteren eine Anordnung zum Verkürzen der Verarbeitungs- und Konkurrenzzeit durch Verteilen der Speicherung von Informationen zwischen Hochgeschwindigkeitsspeichern mit geringer Speicherkapazität und einem Massenspeicher mit langsamerer Geschwindigkeit und hoher Speicherkapazität.

Das Problem

[0002] Es ist bekannt, Mehrfachknoten-Kommunikationsnetze aktiv zu verwalten, um die Netzverkehrsbedienungsfähigkeiten zu verbessern. Netze werden mit genügend Einrichtungen an jedem Knoten konstruiert, um den erwarteten Verkehr ausreichend zu betreuen. Dies beinhaltet die Bereitstellung von Einrichtungen, die erforderlich sind, um das normale Verkehrsaufkommen zu bewältigen, sowie von zusätzlichen Einrichtungen, um nicht oft vorkommenden Spitzenverkehr in einem wirtschaftlich vertretbaren Umfang zu bewältigen. Kommunikationsnetze sind normalerweise nicht dafür konstruiert, den Umfang an Einrichtungen bereitzustellen, der erforderlich wäre, um Verkehrsspitzen zu bewältigen, die theoretisch möglich sind, aber allenfalls selten vorkommen.

[0003] In Mehrfachknoten-Kommunikationsnetzen kann es zu Verkehrsstaus kommen, auch wenn das Netz als Ganzes so konstruiert ist, ein adäquates Verkehrsaufkommen zu bewältigen. Dieser Stau ist die Folge einer ungleichen Verteilung des Verkehrs, wobei einige, aber nicht alle Netzknoten mit einem zu hohen Verkehrsaufkommen überlastet sind. Ein Netzknoten kann überlastet werden, wenn er der Zielknoten ist, an den die Netzverbindungsanforderungen gerichtet werden. Ein Netzknoten kann auch überlastet werden, wenn er über einen Link mit dem angeforderten Zielknoten verbunden wird und Anforderungen von vorgeschalteten Knoten empfängt, die an den Zielknoten gerichtet sind. Es ist bekannt, Netze mit Verkehrsgestaltungseinrichtungen auszustatten, um die Überlastung von Knoten infolge einer ungleichen Verkehrsverteilung zu minimieren. Diese Verkehrsgestaltungsknoten überwachen den Verkehr an jedem Knoten sowie die Verbindungsanforderungen, die durch jeden Knoten erzeugt werden. Ein Stau an einem entfernten Knoten wird verhindert, indem die Anzahl der Anforderungen gedrosselt wird, die ein Sendeknoten für den Zugriff auf einen entfernten Knoten, der bereits überlastet ist, erzeugen kann.

[0004] Mehrfachknoten-Netze und ihre Verkehrsgestaltungseinrichtungen versetzen Netze in die Lage, normale Verkehrsaufkommen mit einem zufriedenstellend niedrigen Staugeschehen zu bewältigen. Die Einrichtungen, die für die Verwaltung und Steuerung des Netzverkehrs benötigt werden, sind jedoch komplex, teuer und senken den Verkehrsdurchsatz des Netzes infolge der Komplexität der erforderlichen Verarbeitungsoperationen. Diese Einrichtungen beinhalten prozessorgesteuerte Linked-List-Maschinen am Eingang und Ausgang der Knoten, um ankommenden und abgehenden Verkehr zu puffern. Der Betrieb einer Linked-List-Maschine ist mit komplexen Datenverarbeitungsoperationen verbunden, die für die Minimierung von Konkurrenzproblemen innerhalb der Linked-List-Maschinen erforderlich sind. Die Komplexitäten dieser Konkurrenzprobleme mindern die Verkehrsbedienungsfähigkeiten des gesamten Netzes.

[0005] Eine Überlastung kann durch die Erschöpfung der Linked-List-Puffer an jedem Knoten verursacht werden, wenn Zeiträume mit hohem Verkehrsaufkommen zu bewältigen sind. Das kann dazu führen, dass Pakete abgeworfen werden, so dass die Leistung des Systems bedenklich gemindert wird. Eine Pufferüberlastung wird durch eine unzureichende Puffergröße oder durch die Verwendung von Puffern mit unzureichender Geschwindigkeit, um den ankommenden Verkehr zu bewältigen, verursacht. Die Systemkonstrukteure sahen sich bisher mit den Alternativen konfrontiert, Puffer zu verwenden, die langsam und groß waren oder die schnell sind, aber eine kleine Kapazität besitzen. Puffer, die langsam und groß sind, behindern den Fluss des Netzverkehrs, indem sie das Abwerfen von Paketen verursachen. Puffer, die schnell sind und eine kleine Kapazität besitzen, führen ebenfalls dazu, dass die Puffer überlastet werden und Pakete abgeworfen werden, weil während eines hohen Übertragungsblockaufkommens nicht genügend Puffer zur Verfügung stehen.

[0006] Ein zugrunde liegendes Problem, das mit beiden Typen von Puffern verbunden ist, hat seine Ursache in Konkurrenzproblemen, die auftreten, wenn mehrere Zugriffe zum Zweck der Verwendung derselben Einrichtung unternommen werden. Dazu kommt es beispielsweise, wenn mehrere Zugriffe empfangen werden, um

einen Zugriff zum Zweck des Beschreibens oder Lesens einer bestimmten Speicherbank zu unternehmen. Unter solchen Umständen ist ein einziger Zugriff erfolgreich, während der andere Zugriff auf die Verfügbarkeit der angeforderten Speicherbank wartet. Rufe, die mit dem erfolgreichen Zugriff verbunden sind, werden ausreichend bedient, während jene, die mit verzögerten Zugriffen verbunden sind, entweder fallengelassen werden oder unzureichend bedient werden.

[0007] Zur Konkurrenz um den Zugriff auf RAM-Speicherbänke kommt es entweder infolge der Verwendung einer unzureichenden Anzahl von RAM-Speicherbänken und/oder infolge der Konkurrenzeinrichtungen, die für die Bedienung der Speicherbänke bereitgestellt sind. Einige Konkurrenzeinrichtungen stützen sich auf Algorithmen und Prozesse, welche die Rate begrenzen, mit der Zugriffe bedient werden können. Eine solche Anordnung nach dem Stand der Technik verwendet einen Algorithmus, der eine Mindestzeitverzögerung von etwa 250 Nanosekunden zwischen Zugriffen erfordert. Dies ist eine merkliche Beschränkung, weil nichts vorgesehen ist, um zu bestimmen, ob eine zweite RAM-Speicherbank zur Verfügung steht oder nicht, um einen Zugriff zu empfangen, nachdem ein Zugriff auf eine erste Speicherbank zugewiesen wurde. Somit ist – bei einem erforderlichen Zeitintervall von 250 Nanosekunden zwischen der Bedienung von Zugriffen – der Systemdurchsatz auf die Bedienung von maximal 4.000.000 Zugriffen je Sekunde beschränkt, wobei der Verfügbarkeit von RAM-Speicherbänken keinerlei Beachtung geschenkt wird. Ein weiteres Problem, das mit den existierenden Konkurrenzanordnungen verbunden ist, besteht darin, dass viele von ihnen Logikeinrichtungen verwenden, die komplex, teuer und für die Bewältigung hoher Verkehrsaufkommen unzureichend sind.

Die Lösung

[0008] Die vorliegende Erfindung überwindet diese Konkurrenzprobleme gemäß einem ersten möglichen Ausführungsbeispiel, das eine höhere Anzahl von RAM-Speicherbänken bereitstellt. Dies allein verringert schon die Möglichkeit von Konkurrenzsituationen. Ein zweites Merkmal, das durch die vorliegende Erfindung bereitgestellt wird, besteht darin, jede RAM-Speicherbank mit einem zugehörigen Steuerelement auszustatten, das als "Zustandscontroller" bezeichnet wird. Der Zustandscontroller ist eine Schnittstelle zwischen seiner RAM-Speicherbank und dem Systembus, über den die Zugriffsanforderungen empfangen werden. Alle Zugriffsanforderungen werden dem Systembus durch einen Zugangsflussregler übermittelt, der alle Zugriffsanforderungen empfängt, die durch einen Knoten erzeugt wurden, der feststellt, ob eine RAM-Speicherbank verfügbar ist, um die Zugriffsanforderungen zu bedienen, der die Zugriffsanforderung puffert, wenn die spezifizierte RAM-Speicherbank momentan belegt ist, und der die Zugriffsanforderung an den zu der RAM-Speicherbank gehörenden Zustandscontroller übermittelt, wenn die RAM-Speicherbank gerade frei ist. Wenn die RAM-Speicherbank belegt ist, so übermittelt der Zustandscontroller ein Signal an den Zugangsflussregler, das anzeigt, dass seine RAM-Speicherbank momentan damit beschäftigt ist, einen anderen Zugriff zu bedienen, und nicht zur Verfügung steht, um im Augenblick weitere Zugriffsanforderungen zu bedienen.

[0009] Im Fall einer Schreib-Zugriffsanforderung tastet ein Zugangsflussregler alle Zustandscontroller ab, wenn er versucht, die Zugriffsanforderung an die RAM-Speicherbänke zu routen. Dabei umgeht er sofort Zustandscontroller, die momentan ein Belegt-Signal für ihre zugehörige RAM-Speicherbank oder ihre zugehörigen RAM-Speicherbänke, die keinen Speicherplatz frei haben, erzeugen. Der Zugangsflussregler umgeht die belegten oder die voll ausgelasteten RAM-Speicherbänke und ihre Zustandscontroller und richtet die Zugriffsanforderung an nicht-belegte RAM-Speicherbänke, die verfügbare Puffer für Speicherplatz haben.

[0010] Die Speicherbausteine der RAM-Speicherbank sind von dem Typ, der eine hohe Geschwindigkeit, aber eine relativ geringe Speicherkapazität besitzt. Jede RAM-Speicherbank kann rasch jede an sie gerichtete Zugriffsanforderung verarbeiten. Nach der Beendigung seines Zugriffszyklus' beendet sein Zustandscontroller das Belegt-Signal an einen Zugangsflussregler, der den Zustand seines RAM anzeigt. Sofort nach der Beendigung des Belegt-Signals weiß der Zugangsflussregler, dass die RRM-Speicherbank nun verfügbar ist, um neue Zugriffsanforderungen zu bedienen.

[0011] Ein weiteres mögliches Ausführungsbeispiel der Erfindung ist die Verwendung eines RAM-Speicherbank-Belegt-Signals, das nur für das kurze Zeitintervall fortdauert, in dem die RAM-Speicherbank belegt ist. Die Bereitstellung dieses Signals stellt eine Konkurrenzanordnung dar, die vorteilhaft und in der Lage ist, Zugriffsanforderungen mit einer Rate zu bedienen, die nur durch die Geschwindigkeit der RAM-Bausteine, aus denen die RAM-Speicherbank besteht, begrenzt ist. Diese Konkurrenzanordnung ist eine signifikante Verbesserung gegenüber den Anordnungen nach dem Stand die Technik, bei denen die Rate, mit der Zugriffe bedient werden können, durch die Auferlegung von obligatorischen Zeitintervallen begrenzt ist oder durch die Komplexität der bereitgestellten Konkurrenzlogik begrenzt ist.

[0012] Durch die Verwendung der Konkurrenzeinrichtungen, welche die vorliegende Erfindung verkörpern, ist die maximale Rate, mit der die dynamische RAM-Speicherbankeinrichtung arbeiten kann, nur durch die Geschwindigkeit der verwendeten RAM-Bausteine begrenzt, abgesehen von den jeweiligen Beschränkungen, die den Konkurrenzanordnungen eigen sind. Die dynamischen Hochgeschwindigkeits-RAM-Speicherbankeinrichtungen, welche die vorliegende Erfindung verkörpern, können in einem Pipeline-Verfahren betrieben werden, um Pakete zu bedienen, die mit der Bus-Rate von Lichtwellenleiterübertragungseinrichtungen ankommen. Die durch die vorliegende Erfindung bereitgestellte Konkurrenzanordnung erhöht die Rate, mit der die prozessorgesteuerten Linked-List-Maschinen ankommenden und abgehenden Verkehr mit minimalem Stau während der Bewältigung hoher Verkehrsaufkommen bedienen können.

Aspekte

[0013] Ein Aspekt der Erfindung ist ein Verfahren zum Betreiben eines Speichermanagementsystems, das dafür ausgelegt ist, Linked-List-Dateien zu verarbeiten; das System umfasst mehrere Hochgeschwindigkeitsspeicher mit niedriger Speicherkapazität und einen Massenspeicher mit niedrigerer Geschwindigkeit und hoher Speicherkapazität, die Hochgeschwindigkeitsspeicher weisen eine erste Datenrate auf, der Massenspeicher weist eine zweite Datenrate unter der ersten Datenrate auf, das System umfasst weiterhin einen Zugangsflussregler zum Erzeugen von Anforderungen für das Lesen und Schreiben von Linked-Lists durch die Speicher, das Verfahren umfasst die folgenden Schritte:
Einleiten des Schreibens einer Linked-List in den Hochgeschwindigkeitsspeichern durch Senden einer Schreibanforderung von dem Zugangsflussregler an die Hochgeschwindigkeitsspeicher;
Schreiben eines Kopfpuffers und eines Endpuffers und mindestens eines Zwischenpuffers der Linked-List in die Hochgeschwindigkeitsspeicher; und
Übertragen des mindestens einen Zwischenpuffers von den Hochgeschwindigkeitsspeichern zu dem Massenspeicher, während der Kopfpuffer und der Endpuffer in den Hochgeschwindigkeitsspeichern verbleiben.

[0014] Ein weiterer Aspekt der Erfindung ist ein Verfahren zum Betreiben eines Speichermanagementsystems, das dafür ausgelegt ist, Linked-List-Dateien zu verarbeiten; das System umfasst mehrere Hochgeschwindigkeitsspeicher mit niedriger Speicherkapazität und einen Massenspeicher mit niedrigerer Geschwindigkeit und hoher Speicherkapazität, die Hochgeschwindigkeitsspeicher weisen eine erste Datenrate auf, der Massenspeicher weist eine zweite Datenrate unter der ersten Datenrate auf, das System umfasst weiterhin einen Zugangsflussregler zum Erzeugen von Anforderungen für das Lesen und Schreiben von Linked-Lists durch die Speicher, das Verfahren umfasst die folgenden Schritte:
Senden einer Leseanforderung für eine spezifizierte Linked-List von dem Zugangsflussregler zu den Hochgeschwindigkeitsspeichern, die Puffer der spezifizierten Linked-List enthalten;
Lesen des Kopfpuffers der spezifizierten Linked-List;
Übertragen des mindestens einen Zwischenpuffers der Linked-List von dem Massenspeicher zu einem der Hochgeschwindigkeitsspeicher;
Bezeichnen des zu dem einen Hochgeschwindigkeitsspeicher übertragenen Zwischenpuffers als den Ersatzkopfpuffer der spezifizierten Linked-List;
Auslesen des Zwischenpuffers der spezifizierten Linked-List aus dem einen Hochgeschwindigkeitsspeicher; und
Senden der ausgelesenen Puffer der spezifizierten Linked-List zu dem Zugangsflussregler.

[0015] Das Verfahren beinhaltet des Weiteren vorzugsweise die folgenden Schritte:
Betreiben des Systems, um Linked-Lists für mehrere Anforderungen von dem Zugangsflussregler gleichzeitig zu verarbeiten;
Betreiben des Systems, um Puffer einer Linked-List zu verarbeiten, die in verschiedenen einzelnen der Hochgeschwindigkeitsspeicher gespeichert sind;
Betreiben des Systems, um einen Endpuffer als den zu einer neuen Linked-List geschriebenen ersten Puffer zu schreiben; und
Lesen des Kopfpuffers einer Linked-List zuerst.

[0016] Das System umfasst des Weiteren vorzugsweise mehrere Zustandscontroller, die jeweils einem entsprechenden der Hochgeschwindigkeitsspeicher zugeordnet sind, wobei das System weiterhin einen Anforderungsbus umfasst, der den Zugangsflussregler mit den Zustandscontrollern verbindet, wobei der Schritt des Sendens einer Leseanforderung die folgenden Schritte beinhaltet:
Betreiben des Zugangsflussreglers, um einen freien Hochgeschwindigkeitsspeicher auszuwählen, der die Leseanforderung empfangen soll;
Senden der Leseanforderung von dem Zugangsflussregler über den Anforderungsbus zu dem Zustandscont-

roller, der dem ausgewählten Hochgeschwindigkeitsspeicher zugeordnet ist;
Betreiben des Zustandscontrollers, um das gegenwärtige Belegungsniveau des ausgewählten Hochgeschwindigkeitsspeichers zu bestimmen;
Senden der Anforderung an den Hochgeschwindigkeitsspeicher, wenn das gegenwärtige Belegungsniveau ein vorbestimmtes Niveau nicht übersteigt; und
Anfordern einer Verbindung zu dem Massenspeicher, wenn das gegenwärtige Belegungsniveau des ausgewählten Hochgeschwindigkeitsspeichers das vorbestimmte Niveau übersteigt;
wobei das System weiterhin einen Hintergrundzugangsmultiplexer und einen die Zustandscontroller mit dem Multiplexer verbindenden Zugangsbus enthält, wobei das System weiterhin einen Bus enthält, der den Multiplexer mit dem Massenspeicher verbindet, wobei das Verfahren die Schritte des Betreibens des Multiplexers beinhaltet zum:
Empfangen einer Anforderung von den Zustandscontrollern für eine Verbindung zu dem Massenspeicher;
Bestimmen, welchem von mehreren anfordernden Zustandscontrollern Zugang zu dem Massenspeicher gewährt werden soll;
Verbinden des einen anfordernden Zustandscontrollers mit dem Massenspeicher;
Steuern des Betriebs des Massenspeichers bei der Übertragung von Daten von dem einen Hochgeschwindigkeitsspeicher zu dem Massenspeicher; und
Übertragen von Puffern der Linked-List von dem Zustandscontroller zu dem Multiplexer über den Zugangsbus.

[0017] Der Schritt des Übertragens der Puffer aus dem Massenspeicher beinhaltet vorzugsweise die folgenden Schritte:
Herausübertragen von Zwischenpuffern einer Linked-List aus dem Massenspeicher zu den Hochgeschwindigkeitsspeichern in einem Burstmodus mit einer Datenrate, die der Datenrate der Hochgeschwindigkeitsspeicher im Wesentlichen gleich ist;
Speichern der ausgelesenen Puffer in den Hochgeschwindigkeitsspeichern; und
danach Auslesen von Puffern der Linked-List von den Hochgeschwindigkeitsspeichern zur Übertragung zu dem Zugangsflussregler;
Schreiben von Puffern zu einer existierenden Linked-List durch Übertragen des existierenden Endes der existierenden Linked-List von den Hochgeschwindigkeitsspeichern zu dem Massenspeicher; und
Schreiben eines neuen Puffers als einen neuen Endpuffer der existierenden Linked-List in die Hochgeschwindigkeitsspeicher.

[0018] Der Schritt des Betreibens des Zustandscontrollers beinhaltet vorzugsweise die folgenden weiteren Schritte:
gleichzeitiges Empfangen von Puffern von mehreren Linked-Lists;
Trennen der Puffer von mehreren Linked-Lists, die zu dem Zugangsflussregler gerichtet sind; und
Ausweiten mehrerer von dem Zugangsflussregler empfangener Zugänge zu den Hochgeschwindigkeitsspeichern;
Reagieren auf jede empfangene Anforderung von dem Zugangsflussregler, um das gegenwärtige Belegungsniveau des Hochgeschwindigkeitsspeichers zu bestimmen, der dem Zustandscontroller zugeordnet ist;
Ausweiten des Zugangs auf den assoziierten Hochgeschwindigkeitsspeicher, wenn das gegenwärtige Belegungsniveau nicht überschritten ist;
Signalisieren dem Zugangsflussregler, die Anforderung zu puffern, wenn das gegenwärtige Belegungsniveau überschritten ist;
Steuern der Übertragung eines Puffers von dem Hochgeschwindigkeitsspeicher in einem Burstmodus zu dem Massenspeicher;
Steuern der Übertragung eines Puffers von dem Massenspeicher zu dem Hochgeschwindigkeitsspeicher;
Bestimmen, ob der Massenspeicher frei ist, wenn eine Übertragung angefordert wird;
Ausweiten der Puffer auf den Massenspeicher, falls frei; und
Puffern der Übertragung, wenn der Massenspeicher belegt ist.

[0019] Der Schritt des Betreibens des Multiplexers beinhaltet vorzugsweise die folgenden weiteren Schritte:
Bestimmen, welchem von mehreren bietenden Hochgeschwindigkeitsspeichern Zugang zu dem Massenspeicher gewährt werden soll;
Puffern der Anforderungen von anderen Puffern in dem einen Hochgeschwindigkeitsspeicher;
Bestimmen der Identität zu dem Hochgeschwindigkeitsspeicher, zu dem ein Puffer von dem Massenspeicher übertragen werden soll; und
Steuern der Übertragung des Puffers in einem Burstmodus von dem Massenspeicher zu dem identifizierten Hochgeschwindigkeitsspeicher.

[0020] Das Verfahren umfasst vorzugsweise die folgenden weiteren Schritte:
Erzeugen eines Signals, das für jeden Hochgeschwindigkeitsspeicher eindeutig ist und den belegten/freien Zustand jedes Hochgeschwindigkeitsspeichers anzeigt;
Ausweiten jedes erzeugten Signals auf den Zugangsflussregler;
Betreiben des Zugangsflussreglers, um Anforderungen für das Schreiben oder Lesen von Linked-Lists durch die Hochgeschwindigkeitsspeicher zu empfangen;
Betreiben des Zugangsflussreglers als Reaktion auf den Empfang einer Anforderung zum Lesen von belegten/freien Signalen, erzeugt von den Hochgeschwindigkeitsspeichern;
Betreiben des Zugangsflussreglers als Reaktion auf das Lesen, um einen freien der Hochgeschwindigkeitsspeicher zu identifizieren; und
Betreiben des Zugangsflussreglers zum Ausweiten einer Anforderung für das Lesen oder Schreiben einer Datei auf den freien Hochgeschwindigkeitsspeicher.

[0021] Ein weiterer Aspekt der Erfindung umfasst ein Speichermanagementsystem, das dafür ausgelegt ist, Linked-List-Dateien zu verarbeiten; wobei das System Folgendes umfasst:
mehrere Hochgeschwindigkeitsspeicher mit niedriger Speicherkapazität und einen Massenspeicher mit niedrigerer Geschwindigkeit und hoher Speicherkapazität, wobei die Hochgeschwindigkeitsspeicher eine erste Datenrate und der Massenspeicher eine zweite Datenrate unter der ersten Datenrate aufweisen;
einen Zugangsflussregler zum Erzeugen von Anforderungen für das Lesen und Schreiben von Linked-Lists durch die Speicher;
eine Vorrichtung zum Einleiten des Schreibens einer Linked-List in den Speichern durch Senden einer Schreib-Lese-Anforderung an einen freien der Hochgeschwindigkeitsspeicher;
eine Vorrichtung zum Schreiben eines Kopfpuffers und eines Endpuffers und mindestens eines Zwischenpuffers der Linked-List in die Hochgeschwindigkeitsspeicher;
eine Vorrichtung zum Übertragen des mindestens einen Zwischenpuffers der Linked-List von den Hochgeschwindigkeitsspeichern zu dem Massenspeicher, während der Kopfpuffer und der Endpuffer der Linked-List in den Hochgeschwindigkeitsspeichern verbleibt;
eine Vorrichtung für das nachfolgende Senden einer Anforderung für das Lesen der Linked-List aus dem Zugangsflussregler zu den Hochgeschwindigkeitsspeichern;
eine Vorrichtung zum Lesen des Kopfpuffers der Linked-List in einem der Hochgeschwindigkeitsspeicher;
eine Vorrichtung zum Übertragen des mindestens einen Zwischenpuffers der Linked-List von dem Massenspeicher zu den Hochgeschwindigkeitsspeichern;
eine Vorrichtung zum Bezeichnen des übertragenen Puffers in den Hochgeschwindigkeitsspeichern als einen neuen Kopfpuffer;
eine Vorrichtung zum nachfolgenden Auslesen des Kopfpuffers und des Endpuffers sowie der Zwischenpuffer aus den Hochgeschwindigkeitsspeichern; und
eine Vorrichtung zum Übertragen der ausgelesenen Puffer der Linked-List zu dem Zugangsflussregler.

[0022] Das Speichermanagementsystem umfasst vorzugsweise weiterhin Folgendes:
eine Vorrichtung, die die Hochgeschwindigkeitsspeicher enthält, zum Erzeugen eines Signals, das für jeden Hochgeschwindigkeitsspeicher eindeutig ist und den gegenwärtigen belegten/freien Zustand jedes Hochgeschwindigkeitsspeichers anzeigt;
eine Vorrichtung zum Ausweiten der Signale auf den Zugangsflussregler;
eine Vorrichtung, die den Zugangsflussregler enthält, zum Empfangen einer Anforderung für das Schreiben oder Lesen von Linked-Lists durch die Hochgeschwindigkeitsspeicher;
eine Vorrichtung, die den Zugangsflussregler enthält, zum Lesen der belegt/frei-Signale als Reaktion auf den Empfang der Anforderung;
eine Vorrichtung, die den Zugangsflussregler enthält, zum Bestimmen des aktuellen belegten/freien Zustands jedes der Hochgeschwindigkeitsspeicher als Reaktion auf das Lesen; und
eine Vorrichtung, die den Zugangsflussregler enthält, zum Gewähren einer Anforderung für das Lesen oder Schreiben einer Linked-List durch den Hochgeschwindigkeitsspeicher als Reaktion auf eine Bestimmung, dass einer der Speicher gegenwärtig frei ist.

Beschreibung der Zeichnungen

[0023] Diese und weitere Aspekte der Erfindung werden durch ein Studium der detaillierten Beschreibung in Verbindung mit den Zeichnungen besser verständlich.

[0024] Fig. 1 offenbart ein Mehrfachknoten-Netz.

[0025] Fig. 2 offenbart Hardware-Elemente, die einen Knoten umfassen.

[0026] Fig. 3 offenbart das Mehrfachknoten-Netz von Fig. 1 mit einem Stau.

[0027] Fig. 4–Fig. 6 offenbaren eine Anordnung von Linked-List-Puffern.

[0028] Fig. 7 offenbart einen hypothetischen Verkehrszustand an dem Knoten von Fig. 3.

[0029] Fig. 8 und Fig. 9 offenbaren die Verarbeitung von Zugriffsanforderungen durch ein Netz, das von mehreren RAM-Speicherbänken bedient wird.

[0030] Fig. 10 offenbart vier RAM-Speicherbänke, die mit einem Steuerbus und einem Datenbus verbunden sind.

[0031] Fig. 11 ist ein Zeitsteuerungsschaubild, das den Prozess beschreibt, mit dem die RAM-Speicherbank von Fig. 10 Zugriffsanforderungen, wie in den Fig. 8 und Fig. 9 gezeigt, bedient.

[0032] Fig. 12 offenbart ein prozessorgesteuertes Linked-List-Speichersystem mit vier RAM-Speicherbänken.

[0033] Fig. 13 ist einem Zeitsteuerungsschaubild, das den Betrieb des Systems von Fig. 12 veranschaulicht.

[0034] Fig. 14 ist ein weiteres Zeitsteuerungsschaubild, das einen alternativen Betrieb des Systems von Fig. 12 veranschaulicht.

[0035] Fig. 15 offenbart ein prozessorgesteuertes Linked-List-Verarbeitungssystem mit acht RAM-Speicherbänken.

[0036] Fig. 16 ist ein Zeitsteuerungsschaubild, das den Betrieb des Systems von Fig. 15 veranschaulicht.

[0037] Fig. 17 offenbart Elemente des Zustandscontrollers **1804** von Fig. 18.

[0038] Fig. 18 offenbart ein prozessorgesteuertes RAM-Speichersystem, das die vorliegende Erfindung verkörpert.

[0039] Fig. 19 offenbart eine Anordnung von Linked-List-Puffern gemäß der vorliegenden Erfindung.

[0040] Fig. 20 offenbart einen Lesebetrieb, der die Erzeugung eines Puffers als den neuen Kopf einer Linked-List veranschaulicht.

[0041] Fig. 21 ist ein Zeitsteuerungsschaubild, das einen Betrieb für das System von Fig. 18 veranschaulicht.

[0042] Fig. 22–Fig. 25 sind Flussdiagramme, die den Betrieb der vorliegenden Erfindung veranschaulichen.

Detaillierte Beschreibung

Beschreibung von Fig. 1

[0043] Die vorliegende Erfindung umfasst eine verbesserte Speicherschnittstelle zum Erhöhen des Verkehrsdurchsatzes eines Mehrfachknoten-Kommunikationsnetzes. Diese verbesserte Speicherschnittstelle verkörpert Verkehrsgestaltungselemente, welche die Freigabe von paketisierten Informationen in ein Kommunikationsnetz steuern, wie beispielsweise jenes, das in Fig. 1 gezeigt ist. Das Netz von Fig. 1 hat untereinander verbundene Schaltelemente – Knoten genannt –, die miteinander kommunizieren. Die Knoten sind mit A, B, C, D, E, F und G bezeichnet und sind durch separate Links verbunden, die als Link 1 bis Link 8 bezeichnet sind. Die Knoten von Fig. 1 definieren ein Netz, das Verkehr von Eingangsports zu Ausgangsports verteilt.

Beschreibung der Fig. 2 und Fig. 3

[0044] Fig. 2 offenbart die Ausrüstung, die einen Knoten von Fig. 1 verkörpert. Fig. 1 offenbart einen einzelnen Pfad (Link 1), der Knoten A und Knoten B untereinander verbindet. Jeder Knoten von Fig. 1 ist mit einem

weiteren Knoten durch ankommende Links und abgehende Links verbunden. Knoten A empfängt von Knoten B durch ankommende Links von Knoten A und sendet an Knoten B mittels der abgehenden Links von Knoten A.

[0045] Fig. 2 offenbart die Details jedes Knotens, einschließlich seiner ankommenden Links und abgehenden Links. Pfad **218** ist ein abgehender Link seines Knotens; Pfad **223** ist ein ankommender Link seines Knotens. Fig. 2 zeigt die Ausrüstung, die den Knoten definiert, und in der linken Hälfte von Fig. 2 umfasst der Knoten ankommende Links **223**, Splitter **222** und Pfade **221**, die jeden Splitter mit jedem der mehreren Ports **201** bis **205** verbinden. Port **201** ist auf seiner rechten Seite mit einem abgehenden Link **218** verbunden, der zu einem weiteren Knoten führt. Die auf der linken Seite von Fig. 2 gezeigte Ausrüstung enthält abgehende Ports **201** bis **205**, die es dem Knoten ermöglichen, mit Links **218** verbunden zu werden, die über die fünf abgehenden Ports **201** bis **205** zu fünf verschiedenen Knoten führen.

[0046] Die rechte Seite von Fig. 2 veranschaulicht die Ausrüstung, die den abgehenden Port **202** verkörpert, in größerer Detailliertheit. Port **202** enthält mehrere Linked-List-Schlangen **215** und eine Steuerlogik **1800**, die jeder einzelnen Schlange zugeordnet ist. Die Steuerlogik wird später in Fig. 18 detailliert veranschaulicht und umfasst die Ausrüstung, die erforderlich ist, um die Linked-List-Informationen zu verarbeiten und sie zum Multiplexer **213** zu übermitteln. Fünf Schlangen **215** werden durch die jeweiligen Steuerlogikelemente **1800** bedient, die durch Pfade **231** mit dem Multiplexer **213** verbunden sind. Multiplexer **213** verbindet die Pfade **231** mit dem abgehenden Link **218**, der einem der Links 1 bis 8 in Fig. 1 entsprechen würde. Der Knoten von Fig. 2 ist für eine maximale Leistung mit einer Ausgabewarteschlangeneinreihung ausgestattet. Für jeden abgehenden Port **201**–**205** wird zu jedem Zeitpunkt eine der Schlangen **215** durch den Multiplexer **213** ausgewählt, und ein Paket von einer Schlange **215** wird zu dem abgehenden Link **218** gesandt. Die Auswahl einer Schlange **1800** für einen bestimmten Port hängt von dem Verkehrsgestaltungsalgorithmus ab, der von dem Netz verwendet wird. Nehmen wir einmal an, dass verschiedene Knoten von Fig. 1 mit einer kleinen Community kommunizieren wollen, die durch einen der Knoten, wie beispielsweise Knoten A, bedient wird. Wenn der Verkehr um den Zugang zum Knoten A über einen Link zu Knoten A konkurriert, so kann es zu einer Überlastung kommen. Pakete werden gepuffert, was eine vollzogene Übertragung gestattet, während die beteiligten Links bedient werden.

[0047] Dieses Szenario mit hohem Verkehrsaufkommen ist detailliert in Fig. 3 gezeigt, wobei der Knoten A der anfordernde Knoten ist und die geschwärzten Links 1, 2, 3, 4, 5 und 6 Links darstellen, die mögliche Leitungspfade zum anfordernden Knoten A bedienen.

[0048] Knoten B trägt den Verkehr von vier Knoten seinen eigenen und den von drei Knoten (C, E, und F), die den Verkehr erzeugen, der durch Knoten B und/oder Knoten C hindurch fließt, um zu Knoten A zu gelangen. Knoten B hat nur einen einzigen Link (Link 1) zu Knoten A. Knoten G und D können Verkehr mit voller Linkkapazität transportieren. Der Link 1 ist deshalb überlastet. Knoten B muss Verkehr, der zum Knoten A fließt, während konkurrenzintensiver Zeiten puffern. Der Verkehr wird darum in dem Maße versandt, wie die Konkurrenzsituation sich allmählich auflöst.

[0049] Wenn das konkurrenzintensive Intervall lange genug andauert, so können die Puffer in Knoten B überlaufen. Um dies zu verhindern, regeln Verkehrsgestaltungsalgorithmen die Freigabe von Verkehr auf einen Link, damit die Puffer im gesamten Netz abfließen können und in die Lage versetzt werden, mehr Verkehr aufzunehmen, der zu einem anfordernden Knoten fließen soll. Beispielsweise kann es sein, dass die Knoten E und F keinen Verkehr an die Links 5 bzw. 6 freigeben, selbst wenn sie genügend Pakete haben, um diese Links zu füllen. Es würde dann länger dauern, diese Pakete zu versenden, aber der Knoten B würde dadurch eine Überlastung vermeiden, und das Netz würde weniger Pakete abwerfen. Verkehrsgestaltung kann als eine proaktive Flusssteuerung angesehen werden.

[0050] Eine Verkehrsgestaltung erfordert eine erfolgreiche Hochleistungspufferung. Diese Pufferung stützt sich auf Hardware-Linked-List-Prozessoren. Ein Hardware-Linked-List-Prozessor nutzt Speicher effizient, weist Puffer dynamisch ankommenden Paketen zu und sammelt Puffer, nachdem ihre gehaltenen Daten erfolgreich über einen abgehenden Link übertragen wurden.

Beschreibung der Fig. 4–Fig. 6

[0051] Ein Linked-List-Puffer wird gemäß der vorliegenden Erfindung zur Pufferung von Informationen benutzt. In einem initialisierten System ist der gesamte Speicher in generische Puffer unterteilt. Jeder Puffer hat Platz für den Inhalt **401**, der durch diesen Puffer gespeichert wird, und einen Zeiger **402** zum nächsten Puffer.

Dies ist in Fig. 4 gezeigt.

[0052] Bei Initialisierung werden alle Puffer miteinander verkettet, indem das Zeigerfeld eines vorherigen Puffer auf die Adresse des nächsten Puffers eingestellt wird. Dies nennt man eine Frei-Liste und ist in Fig. 5 gezeigt.

[0053] Ein Kommunikationssystem füllt diese generischen Puffer nacheinander mit Informationen und bindet die gefüllten Puffer in eine Schlange ein, die Informationen für eine spezifische Funktion speichert. Nach der Initialisierung des Systems sind alle Schlangen leer. Ihre Schlangenlänge ist null, und ihr Kopf- und Endpunkt ist auf NULL. Wenn Informationen für eine bestimmte Schlange von Fig. 6 ankommen, wird ein generischer Puffer aus der Frei-Liste entnommen, wird mit Informationen bestückt und wird der Liste der Schlange hinzugefügt. Der Endzeiger wird auf die Adresse des hinzugefügten Elements der Schlange geändert, und die Schlangenlängenzählung wird inkrementiert. Wenn Informationen aus einer Schlange gelesen werden sollen, so wird der Inhalt des Puffers am Kopf der Liste jener Schlange gelesen, und der Kopf der Liste wird zum nächsten Puffer in der Liste verschoben. Die Schlangenlänge wird ebenfalls dekrementiert.

[0054] Schlange A in Fig. 6 hat einen Kopfpuffer mit einem Inhalt Q und einen Endpuffer mit einem Inhalt Z; Schlange B hat einen Kopfpuffer mit einem Inhalt N und einen Endpuffer mit einem Inhalt G, während Schlange C einen Puffer mit sowohl einem Kopfpuffer als auch einem Ende mit einem Inhalt HH hat. Dieses System hat elf Puffer und drei Schlangen. Ein Hauptmerkmal von Linked-List-Pufferungssystemen ist, dass die Pufferzuweisung vollständig dynamisch erfolgt. Es ist eine beliebige Verteilung von Puffern auf Schlangen zulässig, so lange die Frei-Liste nicht leer geworden ist. Beispielsweise könnte Schlange A alle elf Puffer haben; dann – einige Zeit später – könnte Schlange A vier Puffer haben, Schlange B könnte vier Puffer haben, und Schlange C könnte drei haben. Einige Zeit später könnten alle Schlangen sein leer, und alle Puffer könnten wieder in der Frei-Liste sein.

[0055] Linked-List-Pufferung eignet sich besonders für Kommunikationsanwendungen, weil ankommende Informationen im Fall einer bestimmten Einrichtung, die mit voller Kapazität arbeitet, Puffer mit einer konstanten Rate in Anspruch nehmen. Aber die Einrichtung muss oft Informationen multiplexen, die zu mehr als einem einzigen Fluss oder Strom gehören. Linked-Lists sind eine effiziente Möglichkeit, die ankommenden Informationen zum Verarbeiten und anschließenden Multiplexen auf abgehende Übertragungsmedien zu demultiplexen und zu organisieren. Nachdem ein verarbeiteter Puffer versandt wurde, kann er durch die Frei-Liste wiederverwendet werden.

[0056] Nehmen wir einmal an, der Gesamtverkehr, der am Knoten B von Fig. 2 ankommt, nimmt zehn Puffer je Sekunde in Anspruch, aber Knoten B sendet ebenfalls zehn Puffer je Sekunde. Dieser Knoten ist ausgeglichen, weil die Puffer ebenso schnell geleert werden, wie sie befüllt werden. Vom Standpunkt der Pufferung aus betrachtet ist es gleichgültig, welche Flüsse kommen oder gehen. Beispielsweise ist es egal, ob der ankommende Verkehr von Knoten C zu Knoten B für Knoten G oder Knoten A bestimmt ist. In einem Moment könnten die Schlangen, die Link 1 zum Knoten A speisen, bestückt werden, und die Schlangen, die den Link zum Knoten G speisen, könnten leer sein; dann – einige Zeit später – könnte die erste Gruppe leer sein, und die zweite Gruppe könnte voll sein. Solange der Gesamtfluss auf die Pufferkapazität abgestimmt ist, bleibt die Systemintegrität gewahrt. Obgleich der Gesamtverkehr auf dem Medium mit einer festen maximalen Rate ankommt und abgeht, können dennoch viele Flüsse in dem unterstützten Kommunikationsmedium gemultiplext werden. Ein flexibles Management von Puffern mindert Übergangszustände in der Ausrüstung.

[0057] Ein Linked-List-Puffer ist für ein Verkehrsgestaltungsszenario nutzbringend. Jeder der Knoten unterstützt einen Teil des Gesamtdurchsatzes. Die Implementierung einer Verkehrsgestaltung bedeutet eine Pufferung in den Knoten hinein, da ankommender Verkehr, der für bestimmte abgehende Links gedacht ist, vor der Übertragung für ein Zeitintervall gepuffert werden muss, um dem Gestaltungsprofil gerecht zu werden. In verbindungslosen paketorientierten Netzen ist der Zielort eines ankommenden Paketes vor der Ankunft des Paketes unbekannt. Obgleich ein bestimmter Strom oder Fluss eine maximale Gesamtrate haben kann, kann ein bestimmter Fluss Informationen als Bursts in einen Knoten einspeisen. Weil während eines solchen Bursts das Fluss-Bursting die gesamte Zinkkapazität in Anspruch nehmen kann, sind Paketflüsse, die durch den Link unterstützt werden, der für andere Knoten vorgesehen ist, per Definition im Ruhezustand.

[0058] Stellen wir uns als Beispiel noch einmal das Netz von Fig. 3 vor. Knoten B ist mit drei Links 1, 4 und 7 verbunden. Der Verkehr auf Link 4 könnte entweder für Knoten A oder Knoten G vorgesehen sein. Die Reihe von Paketen, die für Knoten A bestimmt sind, bildet den Strom oder Fluss zum Knoten A von Knoten B. Die Reihe von Paketen, die für Knoten G bestimmt sind, bildet den Strom oder Fluss zum Knoten G von Knoten B.

Beschreibung von Fig. 7

[0059] Fig. 7 zeigt ein Zeitsteuerungsschaubild, das eine mögliche Abfolge von ankommenden Paketen auf Link 4 beschreibt.

[0060] Während des Bursts von vier Paketen, die für Knoten A bestimmt sind, werden die Puffer, die aus der Frei-Liste herausgenommen werden, alle der Schlange hinzugefügt, die den Link 1 unterstützt. Wenn dieser Burst versandt wird, so strömen die Puffer, die aus der Frei-Liste Strom genommen wurden, in diese Schlange. Aber in jedem Augenblick könnte das nächste ankommende Paket für Knoten A oder Knoten G bestimmt sein. Der Zielort des nächsten ankommenden Paketes ist erst dann bekannt, wenn es zu sehen ist. Somit ist eine flexible dynamische Zuweisung von Puffern für den Betrieb von effizienter Paketvermittelter Ausrüstung notwendig. Während dieses Bursts von vier Paketen empfängt die Schlange, die Link 7 zum Knoten G unterstützt, keine zusätzlichen Puffer, aber weil der Burst für Verkehr gedacht ist, der für Knoten A bestimmt ist, haben diese Schlangen keinen ankommenden Verkehr, weil sie keine Puffer mehr brauchen. Somit ist die flexible dynamische Zuweisung von Puffern für den Betrieb von effizienter paketvermittelter Ausrüstung ausreichend.

Betrieb einer Linked-List

[0061] Linked-List-Maschinen verwenden Halbleiter-Speicherbausteinen als RAM-Speicher. Die Mechanik der Handhabung einer Linked-List zum Hinzufügen oder Entfernen von Puffern beinhaltet eine Reihe von Lesevorgängen und Schreibvorgängen sowohl in den Pufferspeicher als auch in einen zugehörigen Linked-List-Tabellenspeicher. Der Linked-List-Tabellenspeicher ist eine statische Struktur, die eine Nachschlagetabelle für Köpfe und Enden zu jeder Linked-List enthält, die durch den Linked-List-Prozessor unterstützt wird. Wenn beispielsweise ein Fluss entweder Lese- oder Schreibverkehr hat, so verwendet der Linked-List-Prozessor zuerst die Flussnummer, um Adressen des Kopfes und des Endes der interessierenden Liste nachzuschlagen. Mit Kenntnis dieser Adressen für die interessierende Linked-List kann der Prozessor dann vorgeschriebene Operationen an dieser Liste ausführen. Wenn ein Puffer zu einer Linked-List hingefügt werden soll, so wird vom Kopf der Frei-Liste von Fig. 6 ein leerer Puffer genommen, und der Kopf der Frei-Liste wird so umgeschrieben, dass er der nächste leere Puffer in dieser Liste ist. Die Adresse des neuen Kopfes der Frei-Liste ist in dem Link des Puffers enthalten, der gerade zum Auffüllen genommen wurde. Der Inhalt dieses Puffers wird gefüllt, und das Linkfeld in dem Puffer am Ende der Linked-Lists wird mit der Adresse des gerade gefüllten Puffers geschrieben. Dann wird der Tabellenspeicher mit der neuen End-Adresse geschrieben. Im Prozess des Schreibens eines neuen Puffers in eine Linked-List hält der Tabellenspeicher einen Lesevorgang und einen Schreibvorgang aufrecht, und der Pufferspeicher hält zwei Schreibvorgänge aufrecht. Im Prozess des Lesens eines Puffers aus einer Linked-List hält der Tabellenspeicher einen Lesevorgang und einen Schreibvorgang aufrecht, und der Pufferspeicher hält einen Lesevorgang und einen Schreibvorgang aufrecht. Das Schreiben in den Pufferspeicher erfolgt, wenn der entleerte Puffer mit der Frei-Liste neu verknüpft werden muss.

Direktzugriff von Linked-List-Prozessoren auf Pufferspeicher

[0062] Ein wichtiger Aspekt dieses Prozesses ist das direkte Zugreifen (random access) auf Speicher. Eine Reihe von Faktoren beeinflussen die Randomisierung. Wenn eine Linked-List Verkehr von einer Kommunikationseinrichtung puffert, so richtet sich die Reihenfolge von Zugriffen ganz und gar nach dem Verkehr, der durch diese Einrichtung abgewickelt wird. In einem verbindungslosen Netz, wie beispielsweise dem Ethernet, kann es sein, dass ein Paket, das bei einer Einrichtung ankommt, für eine beliebige einer Anzahl von Schlangen bestimmt ist, während es entlang seines Weges geroutet wird. Die Zielschlange für ein künftiges Paket kann im Allgemeinen nicht vorhergesagt werden. Die Zufälligkeit des Eintreffens verwürfelt die Adressen in einer bestimmten Schlange. Obgleich die Übertragung von abgehenden Paketen gesteuert wird, beeinflussen auch hier die Netzbedingungen die Randomisierung. Nehmen wir beispielsweise an, dass eine Anzahl von Schlangen auf einer abgehenden Einrichtung gemultiplext werden sollen. Gelegentlich können alle diese Schlangen Einfluss ausüben, gelegentlich einige, gelegentlich eine einzige oder gar keine. Ein Stau könnte durch eine Flusssteuerung am entfernten Ende der abgehenden Einrichtung oder durch eine Anzahl von ankommenden Einrichtungen, die Verkehr übertragen, der für die abgehende Einrichtung bestimmt ist, verursacht werden.

[0063] Ein zweiter signifikanter Faktor, der die Randomisierung beeinflusst, ist die Frei-Liste. Einträge in der Frei-Liste hängen ganz und gar von der Reihenfolge von Pufferübertragungen auf abgehende Einrichtungen ab. Aber abgehende Übertragungen unterliegen unvorhersagbaren Bedingungen. Somit randomisieren Verkehrsbedingungen die Adressenreihenfolge von leeren Puffern auf der Frei-Liste.

[0064] Sobald ein typisches System, das einen Linked-List-Prozessor für Puffermanagementzwecke verwen-

det, ein oder zwei Sekunden lang unter hoher Last arbeitet, fehlt den Zugriffen auf Pufferspeicher jegliche Korrelation.

Parameter und Mechanik von Pufferspeicherzugriffen

[0065] Da Linked-List-Verarbeitung aus einer Reihe von vorskribierten Zugriffen auf Speicher besteht, hängt die Leistung des Linked-List-Prozessors in hohem Maße davon ab, wie diese Zugriffe verwaltet werden. Aber die letztendlichen Pufferungsanforderungen, wie beispielsweise Speicherkomponentenverfügbarkeit und Zugriffsanforderungen, setzen der Konstruktion von Speichersystemen für Linked-List-Prozessoren Grenzen.

[0066] Die Länge der anhängenden Einrichtung und ihre Kapazität müssen bei der Konstruktion von Speichersystemen berücksichtigt werden, wenn der Linked-List-Prozessor in einer leitungsvermittelten Anwendung verwendet wird. Beispielsweise beträgt das standardmäßige Meldeintervall für eine Kabeldurchtrennung in einem SONET-Netz 60 Millisekunden. Dieser Standard ist aber restriktiv für üblicherweise anzutreffende Netzkabelverlegungen, wo die Meldeintervalle eher bei 200 bis 300 Millisekunden liegen. Das entspricht einer Kabellauflänge von Tausenden Kilometern. Eine einzige OC-768-Faser sendet mehr als 23 Millionen Pakete in 300 Millisekunden beim Bursting von Paketen in der Mindestgröße. Ein Glasfaserkabel kann hundert oder mehr voneinander getrennte und individuelle Litzen mit verkehrstragender Faser aufweisen. Somit müsste ein System, in dem ein einziges derartiges Kabel endet, in der Größenordnung von Milliarden Paketen puffern, um eine nahtlose Wiederherstellung nach einer Kabeldurchtrennung zu bewerkstelligen.

Grundlegendes Problem mit bekannten Hardware-Linked-List-Maschinen

[0067] Ein Speicher-Teilsystem, das Hardware-Linked-List-Verarbeitung unterstützt, muss groß und schnell sein, muss viele Schlangen unterstützen und muss in der Lage sein, Speicher in jeder beliebigen Schlange an jeder beliebigen Taktimpulsflanke zu verarbeiten, um unmittelbare Verschiebungen bei dem augenblicklich anliegenden Verkehr berücksichtigen zu können. Direktzugriffsspeicherbausteine, die Speicher von Linked-List-Prozessoren in geeigneter Weise verarbeiten können, können entweder nicht schnell genug arbeiten oder sind nicht groß genug, um Einrichtungen mit größten Kapazitäten zu puffern.

[0068] Diese Speicher von zwei Typen, handelsübliche synchrone dynamische Speicher (SDRAM)-Bausteine, enthalten bis zu einem Megabit Speicherplatz, aber ihre Direkt-Lese-Schreib-Taktzeit beträgt etwa 60 Nanosekunden. Weil Speicherplatz in der Größenordnung von Megabits für eine OC-768-Fernleitung benötigt wird, ist die Größe des SDRAM geeignet, aber bei einer voll ausgelasteten OC-768-Fernleitung, die Pakete in Mindestgröße burstet, kommt ein Paket alle 40 Nanosekunden an. Somit sind handelsübliche SDRAMs zu langsam, um eine OC-768-Einrichtung zu bedienen.

[0069] Handelsübliche synchrone statische Speicher (SSRAM) arbeiten Lese-Schreib-Direktzugriffe in etwa 2 Nanosekunden, mit einer Latenz von 4 Nanosekunden, ab. Das ist schnell genug, um ein paar OC-768-Einrichtungen und die Steuerungsverwaltungsdaten zu verarbeiten. Aber SSRAMs gibt es nicht mit Kapazitäten von mehr als 16 Megabits. Man würde etwa 90 SSRAM-Bausteine brauchen, um eine OC-768-Einrichtung ausreichend zu puffern. Die Wärmemenge, die durch so viele SSRAM-Bausteine erzeugt werden würde, wäre ein Problem.

[0070] Schlussfolgernd ist festzustellen, dass ein grundlegendes Problem mit den verfügbaren Speichern, die mit Hardware-Linked-List-Prozessoren gebildet werden, darin besteht, dass sie entweder groß und langsam sind (SDRAMs) oder schnell, aber zu klein sind (SSRAMs). Es steht kein Kompromiss zur Verfügung, der zu einem Hardware-Linked-List-Prozessor führt, der sowohl groß als auch schnell ist.

Das verbesserte Linked-List-Prozessor-Design

[0071] Der verbesserte Linked-List-Prozessor der vorliegenden Erfindung verkörpert eine Lösung des Problems, wie man einen Puffer mit großer Kapazität und hoher Dichte sowie einen schnellen, leistungsstarken Puffer erhält. Dies geschieht auf eine neuartige Weise, indem man sich auf die Zufälligkeit des Stromes der Zugriffe auf die Speicher stützt.

[0072] Das Problem der Konkurrenz um Speicher-Teilsysteme, die SDRAM verwenden, hat bisher bewirkt, dass die Systeme darauf warten mussten, dass jeder nachfolgende Speicherzyklus beendet ist, anstatt Zugriffe zu überlappen, indem man das Pipeline-Merkmal von verfügbaren SDRAM nutzt. Dadurch wäre das System in der Lage, mit einer viel schnelleren Bustaktrate zu arbeiten. Die vorliegende Erfindung löst das Konkurrenz-

problem, indem sie es dem RAM-Speicher ermöglicht, mit seiner Port-Geschwindigkeit, nämlich Hunderten von Megahertz, anstatt mit seiner inhärenten Lese-Schreib-Direktzugriffsgeschwindigkeit, nämlich Dutzenden von Megahertz, zu arbeiten. Ein weiteres Merkmal der Erfindung ist, dass in der Gegenwart eines Direktzugriffsstromes zu RAM-Speicherbänken mehr Speicherbänke verwendet werden, weil mit steigender Anzahl von Bänken die Wahrscheinlichkeit abnimmt, dass der nächste Zugriff an eine bereits belegte Bank ergeht.

Beschreibung der Fig. 8 und Fig. 9

[0073] Dynamischer RAM wird verkapselt mit mehreren Bänken in einem einzelnen Gehäuse geliefert. Eine Bank ist eine separat adressierbare Speichereinheit. Weil Bänke Eingangs-/Ausgangsressourcen wie beispielsweise Pins an dem physischen Gehäuse gemeinsam nutzen, können mehrere Bänke gleichzeitig Zugriffsanforderungen verarbeiten. Die maximale Anzahl von unbearbeiteten Anforderungen, die bedient werden können, richtet sich nach der Taktgeschwindigkeit und dem Aufbau des synchronen SDRAM-Bausteins. Als ungefährer Wert kann gelten, dass, wenn ein Zugriff auf eine in dem SDRAM befindliche Bank bis zur Vollendung vier Taktzyklen dauert und sich vier oder mehr Bänke in einem Gehäuse befinden, vier Zugriffe gleichzeitig verarbeitet werden können. Ein neuer Zugriff wird an jeder aufsteigenden Flanke der vier Takte, die an der Vollendung des ersten Zugriff beteiligt sind, ermöglicht.

[0074] Fig. 8 zeigt die vier Bänke **810**, **811**, **812**, und **813** in dem Kreis **803**. Dies stellt vier Bänke in einem möglichen SDRAM dar. Der Trichter **802** und die Abflusstülle **804** stellen die gemeinsam genutzten Steuerungs- und Datenbusressourcen für den Zugriff auf das Innere des SDRAM dar. Vier Zugriffsanforderungen A, B, C, D (**801**) sind gezeigt, die in den Trichter **802** von SDRAM **803** eintreten.

[0075] Es kann immer nur eine Zugriffsanforderung A, B, C, oder D auf einmal in den Trichter **802** eintreten. Jede Bank **810**–**813** braucht für einen Zugriff ebenso viel Zeit wie die anderen. Wenn eine Bank eine Anforderung A beginnt und später eine weitere Bank eine Anforderung B beginnt, so kommen die Ergebnisse für die Anforderung A vor den Ergebnissen von Anforderung B aus der Abflusstülle **804** heraus. Aber für eine Zeit arbeitet eine Bank an der Anforderung A, und eine weitere Bank arbeitet gleichzeitig an der Anforderung B.

[0076] Die Zugriffe können durch eine beliebige Kombination von Bänken bedient werden. Sie könnten alle durch dieselbe Bank bedient werden, oder sie könnten durch verschiedene Bänke bedient werden. Einige Zugriffe haben eine Bank für sich, aber andere Zugriffe teilen sich eine Bank. Für bestimmte Gruppen von Zugriffen kann es sein, dass gar keine Bank benutzt wird. Beispielsweise könnten die Zugriffe A und B durch dieselbe Bank **810** bedient werden, Zugriff C durch Bank **812**, und Zugriff D durch Bank **813**, wie in Fig. 9. Der Prozess des Verteilens einer Gruppe von Zugriffen auf die verfügbaren Bänke nennt man Partitionierung. Aus Sicht der Bänke ist die Zählung von Zugriffen eine unverzichtbare Information, weil alle Zugriffe einen einheitlichen Charakter haben. Somit ist eine Partition ein Beleg für die Zugriffe auf die Bänke. Beispielsweise bedeutet {4, 0, 0, 0}, dass vier Zugriffe eine einzelne Bank belegen und dass die anderen drei Bänke unbelegt sind. In Fig. 9 ist die Partition {2, 1, 1, 0}.

Beschreibung der Fig. 10–Fig. 13

[0077] Der synchrone dynamische RAM (SDRAM) von Fig. 10 ist eine Speicherarchitektur mit vier unabhängigen Bänken. Für jede Bank beinhaltet der Betrieb eine Zugriffslatenz, den Transport von Informationen von den Pins zu der Speicheranordnung für einen Schreibvorgang oder den Transport von Informationen von der Speicheranordnung zu den Pins für den Lesebetrieb. Außerdem gibt es ein erforderliches Vorladungsintervall, damit sich Erfassungsverstärker im Inneren der Speicherbausteine auf den nächsten Lese- oder Schreibzyklus vorbereiten können. Jede der vier Bänke steht zur Verfügung. Jede der vier Bänke hat ihren eigenen Erfassungsverstärker, so dass Zugriffe nur und die Steuerungs- und Datenports des SDRAM konkurrieren.

[0078] Fig. 10 und Fig. 11 zeigen, dass die Steuerbus **801** Aktivitäten verarbeitet, die an die Bänke 1 und 2 eines SDRAM gerichtet sind. Es gibt eine Latenz zwischen den Zugriffsbefehlen "A" und den zugehörigen Lesebefehlen "R" von Fig. 11. Es gibt ebenso eine Latenz zwischen den Lesebefehlen und der Verfügbarkeit von Daten. Außerdem gibt es eine Latenz zwischen den Zugriffsbefehlen A1 und A2. In Fig. 11 wird der Einfachheit halber ein Taktzyklus von 10 Nanosekunden verwendet. Die Gesamtzykluszeit beträgt 80 Nanosekunden. Es kann auf alle vier verfügbaren Bänke des SDRAM während dieser Zeit zugegriffen werden, aber auf jede Bank kann innerhalb dieses Intervalls nur einmal zugegriffen werden. Bei diesen Latenzen spricht man davon, dass der SDRAM eine Pipelinetiefe von vier Stufen aufweist. Die erfolgreiche Abwicklung eines Kollisionsfalles erfordert einen 80 Nanosekunden dauernden zusätzlichen Zyklus des Speichers. Ein Kollisionsfall ist definiert als die Ankunft eines Zugangsflussreglers bei einer belegten SDRAM-Speicherbank. Wenn beispielsweise

eine Zugriffsanforderung für Bank 1 beim Taktzyklus 2 von Fig. 11 ankäme, so könnte sie dem Speicher erst bei Taktzyklus 9 zugeteilt werden, weil Anforderung A für Bank 1 beim Taktzyklus 1 einträfe.

[0079] Fig. 12 zeigt einen Hardware-Linked-List-Prozessor **1201**, der den SDRAM **1203** als seinen Pufferspeicher verwendet, und einen kleineren synchronen statischen RAM **1207** als Tabellenspeicher, der den Kopf, das Ende und die Zählung für jeden unterstützten Fluss hält. SDRAM **1203** hat vier SDRAMs **1210**, **1211**, **1213** und **1214**. Nehmen wir an, der Linked-List-Prozessor **1201** wird mit einem Strom von 5 Zugriffsanforderungen, wie in Fig. 13 gezeigt, für die Takte 1, 3, 5, 11 und 15 konfrontiert. Die ersten drei Zugriffe A1, A2, und A3 können in eine Pipeline eingefügt werden, weil sie zu verschiedenen Bänken gehen, aber der vierte Zugriff A2 bei Takt 11 muss verzögert werden, weil er zu der belegten Bank 2 geht, die den Zugriff A2 von Takt 3 bedient.

[0080] Im Allgemeinen kann das durchschnittliche Aufkommen an Konkurrenz, das man in Gegenwart eines willkürlichen Stromes von Zugriffen antrifft, als ein Maß für einen aus vier Bänken bestehenden SDRAM genommen werden. Es ist einfacher, die gewichtete durchschnittliche Zugriffszeit für einen Block von Zugriffen auf einen aus vier Bänken bestehenden SDRAM zu berechnen, weil die Bänke nur die Eingangs-/Ausgangseinrichtungen gemeinsam nutzen. Nehmen wir beispielsweise die Partition {3, 1, 0, 0}. Wenn die Zugriffe mit (A, B, C, D) bezeichnet werden, so sind die realisierbaren Gruppierungen {(A, B, C), (D)}, {(A, B, D), (C)}, {(A, C, D), (B)} und {(B, C, D), (A)}. Die zwei gewählten Bänke könnten {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4} oder {3, 4} sein.

[0081] Es gibt zwei Möglichkeiten, eine Partition von drei und eine Partition von einer in zwei Bänke hinein abzubilden. Wenn wir beispielsweise versuchen, die Partition {(A, B, C), (D)} in {1, 2} hinein abzubilden, so könnte es entweder (A, B, C) mit Bank 1 und (D) mit Bank 2 oder (D) mit Bank 1 und (A, B, C) mit Bank 1 sein. Weil 4 × 6 × 2 = 48 und es 256 Möglichkeiten gibt, vier Zugriffe zu platzieren, ist die Wahrscheinlichkeit, dass es zur Partition {3, 1, 0, 0} kommt, gleich 0,1875. Der Streuzugriff kann innerhalb zweier in Konflikt befindlicher Zugriffe in einer Pipeline angeordnet werden, so dass die Anzahl der Takte zur Vollendung gleich 8 × 3 = 24 ist.

Partitione n	Wahrschein-lichkeit	Takte bis zur Voll-endung	Gewichtete durchschnittliche Zeit bis zur Vollendung
4,0,0,0	0,0156	32	0,5
3,1,0,0	0,1875	24	4,5
2,2,0,0	0,1406	19	2,6714
2,1,1,0	0,5625	16	9,0
1,1,1,1	0,0938	14	1,3125

TABELLE 1: Anordnung von vier Transaktionen in vier Speicherbausteinen

[0082] Tabelle 1 veranschaulicht Daten, die zu den verschiedenen möglichen Verteilungen von Zugriffen auf eine Bank mit vier Speicherbausteinen gehören, zusammen mit relevanten Daten, die für jede mögliche Verteilung gezeigt sind. Spalte 1 von Tabelle 1 führt die verschiedenen möglichen Partitionen an, in die Zugriffe auf die verschiedenen Bänke erfolgen können. Die oberste Zeile von Tabelle 1 zeigt an, dass die erste Bank vier Zugriffe erhält, während die übrigen Bänke keine bekommen. Die Verteilung von Spalte 1, Zeile 2 ist 3, 1, 0, 0. Die Verteilung der untersten Zeile von Spalte 1 zeigt an, dass alle vier Zugriffe gleichmäßig auf eine jede Bank verteilt sind. Spalte 2 zeigt die Wahrscheinlichkeit jeder Verteilung für ihre Zeile an. Spalte 3 zeigt die Anzahl von Taktzyklen zur Vollendung der Funktion an. Spalte 4 zeigt die gewichtete durchschnittliche Zeit zur Vollendung an. Für Zeile 1 beträgt die Wahrscheinlichkeit der Verteilung aller vier Zugriffe auf die erste Bank 0,0156. Dies erfordert 32 Taktzyklen zur Vollendung mit einer gewichteten durchschnittlichen Zeit zur Vollendung von 0,5. Die unterste Zeile hat die Verteilung 1, 1, 1, 1 und eine Wahrscheinlichkeit von 0,0938, und es dauert 14 Taktzyklen zur Vollendung mit einer durchschnittlichen gewichteten Zeit von 1,3125. Aus dieser Tabelle ist zu erkennen, dass die wahrscheinlichste Verteilung 2, 1, 1, 0 mit einer Wahrscheinlichkeit von 0,5625 ist. Die bestmögliche Verteilung ist in der untersten Zeile gezeigt, wobei die Wahrscheinlichkeit 0,0938 beträgt.

[0083] Für die vier Bänke von Tabelle 1 beträgt die gesamte gewichtete durchschnittliche Zeit zur Vollendung von vier Zugriffen 17,98 Takte. Diese Zahl ist die Summe der gewichteten Zeiten zur Vollendung für alle möglichen Kombinationen von Zugriffen auf einen Speicher mit vier Bänken. Unter der Annahme eines Pipeline-Taktes von zwei 10 Nanosekunden dauernden Systemtakten und einem 16 Bit breiten Bus beträgt der durchschnittliche Dauerdurchsatz für einen SDRAM mit vier Bänken 1,24 GBits/Sekunde, weil jede Speichertransaktion 16 Bits für jeweils zwei Takte beinhaltet. Der Konkurrenzverwaltungsaufwand beträgt 28,5%. Dieser Wert wird berechnet, indem man die Differenz zwischen der durchschnittlichen Zeit zur Vollendung eines Zugriffs und der kürzestmöglichen Zeit zur Vollendung eines Zugriffs durch die kürzestmögliche Zeit zur Vollendung eines Zugriffs teilt. Nehmen wir beispielsweise an, die durchschnittliche Zeit zur Vollendung eines Zugriffs betrüge 50 Nanosekunden und die kürzestmöglichen Zeit zur Vollendung eines Zugriffs betrüge 40 Nanosekunden. Dann würde der Verwaltungsaufwand 25% betragen.

[0084] Fig. 13 und Fig. 14 veranschaulichen den Unterschied zwischen Zugriffen, die keine Konkurrenz beinhalten, und Zugriffen, die Konkurrenz beinhalten.

Beschreibung von Fig. 14

[0085] Häufige Konkurrenz beeinflusst Designentscheidungen. Betrachten wir das Zeitsteuerungsschaubild von Fig. 14. Hier ist der Zugriffsbus auf die Speicherbänke geschlitzt. Jeder Zugriff hat ein Intervall zum Lesen und ein Intervall zum Schreiben. Diese Zugriffe müssen so dicht wie möglich gepackt werden, um die nicht-belegte Zeit auf dem Steuerbus des Speichers zu minimieren.

[0086] Das Strukturieren der Zugriffe in dieser Weise ist sinnvoll, weil Konkurrenz die durchschnittliche Lese-Schreib-Zykluszeit des Speichers erheblich verlängert. Weil die durchschnittliche Lese-Schreib-Zykluszeit in der Größenordnung der maximalen Lese-Schreib-Zykluszeit liegt, wird es effizienter, das System auf die maximale Lese-Schreib-Zykluszeit des Speichers auszulegen. Die Erhöhung der Effizienz und Arbeitsgeschwindigkeit zur Verwendung eines kleineren Lese-Schreib-Zyklus' ist die Kosten der Hardware nicht wert, die erforderlich ist, um Konkurrenz zu verwalten.

[0087] Der Speicher kann Zugriffsanforderungen mit einer Rate entgegennehmen, die durch seine Busgeschwindigkeit definiert wird, weil sein internes Pipeline-Design ihm dies ermöglicht. Aber mit einer Konkurrenzrate von 28,5 erhöht sich das Einreihen in eine Schlange in diesem System – im Gegensatz zu einem konkurrenzfreierem System – drastisch, weil die Konkurrenzrate in einem exponentiellen Verhältnis zur Schlangentiefe steht. Die Einreihungs- und Zustandssteuerungsvorrichtung zum Unterstützen einer variierenden Zugriffszeit ist weit komplizierter als diejenige, die von einer konstanten maximale Zugriffszeit ausgeht. Beispielsweise erfordert ein Design mit einer konstanten maximalen Zugriffszeit keine Schlange, und die Zustandssteuerung ist insgesamt recht einfach. Ein Design mit einer variablen Zugriffszeit erfordert häufig mehr als eine Schlange innerhalb der Maschine, die den Speicher verwendet, sowie eine kompliziertere Zustandslogik, damit die Anwendung diese Maschine stoppen und starten kann, wenn es zu Konkurrenz kommt.

Beschreibung der Fig. 15 und Fig. 16

[0088] Nehmen wir an, es sind acht SDRAM-Bänke vorhanden, wie in Fig. 15 gezeigt. In diesem Fall ist der Speicher in Blöcke von der halben Größe des SDRAM von Fig. 13 partitioniert. Beispielsweise gehen Zugriffe, die für Bank 1 in Fig. 13 bestimmt sind, jetzt entweder zu Bank 1 oder Bank 2 in Fig. 16. Zugriffe, die für Bank 2 in Fig. 13 bestimmt sind, werden zwischen Bank 3 und Bank 4 in Fig. 16 aufgeteilt, und so weiter. In Fig. 13 steht der zweite Zugriff auf Bank 2 in Konkurrenz zum ersten Zugriff auf Bank 2 und verursacht eine Verzögerung. In Fig. 16 geht der erste dieser Zugriffe zu Bank 3, aber der zweite dieser Zugriffe geht zu Bank 4.

[0089] Ein Vergleich der Fig. 13 und Fig. 16 offenbart, dass eine Konkurrenzminderung die Nicht-Belegt-Lücke zwischen Takt 13 (A4) und Takt 16 (R4) in Fig. 13 schließt. In Fig. 16 ist der Datenbus ständig von den Takten 7 bis 16 belegt. Zu Konkurrenz kann es in verschiedenen Kombinationen kommen, wenn ein willkürlicher ankommender Strom von Zugriffen vorliegt. Eine Konkurrenz wird mit acht Bänken nicht ausgeschaltet, aber die Wahrscheinlichkeit, dass es zu Konkurrenz kommt, sinkt mit zunehmender Anzahl der Bänke.

Bänke	Gewichtete durchschnitt-liche Zeit zur Vollendung	Konkurrenzverwaltungs-aufwand	Maximaler Dauer-durchsatz (GBit/s)
4	17,98	28,5 %	1,24
5	17,06	21,9 %	1,31
6	16,47	17,6 %	1,36
7	16,06	14,7 %	1,39
8	15,77	12,6 %	1,42
9	15,55	11,1 %	1,44
10	15,38	9,9 %	1,45
11	15,24	8,9 %	1,47
12	15,12	8,0 %	1,48
13	15,03	7,4 %	1,49
14	14,95	6,8 %	1,5
15	14,88	6,3 %	1,5
16	14,82	5,9 %	1,51

TABELLE 2: Gewichtete durchschnittliche Zeit zur Vollendung von vier Zugriffen für Mengen von Bänken

[0090] Tabelle 2 zeigt, wie die gewichtete durchschnittliche Zeit zur Vollendung mit zunehmender Anzahl von Bänken abnimmt. Beim maximalen Dauerdurchsatz wird wieder von einem Systemtakt von 10 Nanosekunden und einem 16 Bit breiten Datenbus ausgegangen.

[0091] Die Pipelinetiefe beeinflusst ebenfalls die Leistung. Wenn beispielsweise acht Speicherbänke vorhanden sind, es aber nur zwei Stufen in der Pipeline gibt, so beträgt die gewichtete durchschnittliche Zeit zur Vollendung von zwei Zugriffen 6,25 Takte. Der Verwaltungsaufwand für diesen Fall beträgt 4,2%, und der maximale Dauerdurchsatz beträgt 1,51 GBit/s. Nehmen wir an, die Konkurrenzrate fällt auf 5%. Dann beträgt die durchschnittliche Zugriffszeit 3,5 Nanosekunden. Dies ist recht nahe bei der Bustaktzeit. Die Konsequenzen daraus sind in einem Vergleich der Zugriffssteuerungsanordnungen in Fig. 14 und Fig. 16 gezeigt. Es ist zu erkennen, dass die Dichte der Steuerbusaktivität in Fig. 14 höher ist, was bedeutet, dass es mehr Lese-Schreib-Direktzugriffe je Takt gibt als in Fig. 14.

Beschreibung der Fig. 17 und Fig. 18

[0092] Gemäß der vorliegenden Erfindung sind jeder Bank **1803** eines RAM-Gehäuses unabhängige zusammenwirkende Zustandscontroller **1804** zugewiesen. Es wird jeder Bank ermöglicht, unabhängig zu arbeiten und – über ihren Zustandscontroller **1804** – nahtlos Ergebnisse in koordinierter Weise mit den anderen RAM-Bänken **1803** beizusteuern. Außerdem übernehmen die Zustandscontroller **1804** die Flusssteuerung der Schlange des Zugangsflussreglers **1801**, welche die Zugriffsanforderungen hält, im Fall einer gelegentlichen Konkurrenz. Dadurch wird das Abwerfen von Zugriffsanforderungen verhindert. Die Zustandscontroller **1804** verwalten außerdem unabhängig Organisationsfunktionen, wie beispielsweise ein Auffrischen der Bank **1803**. Die Zustandscontroller **1804** sind unabhängig. Ein Zustandscontroller **1804** unterstützt Hintergrundbursttransfers zu und von seiner RAM-Bank **1803** gleichzeitig mit einer Vordergrundzugriffsaktivität in anderen RAM-Bänken **1803**. Dadurch kann der Mittelteil einer Linked-List in einem RAM **1806** gespeichert werden, der von den RAM-Bänken **1803** entfernt liegt. Dadurch bleiben nur die Köpfe und Enden einer Linked-List in den RAM-Bänken **1803** zurück. Beispielsweise würden – wenn wir uns einmal der Schlange **506** in Fig. 6 zuwenden – die Puffer Q und Z irgendwo in den RAM-Bänken **1803** angeordnet werden, aber die Puffer D und R werden in dem entfernten RAM **1806** gespeichert. Die Fähigkeit, die Mitte einer Linked-List räumlich abgesetzt zu speichern, ermöglicht es dem offenbarten System, Listen von beliebiger Größe unter Verwendung handelsüblicher

RAM-Bausteine zu unterstützen. Wenn der Großteil einer Linked-List räumlich abgesetzt in dem räumlich entfernten RAM **1806** gespeichert werden kann, so können die RAMs **1803**, die in FPGAs eingebettet sind, für die Köpfe und die Enden verwendet werden. Die Zustandscontroller **1804** sind mit dem RAM **1803** kombiniert, der die Köpfe und Enden hält. Dieses Design ist effizienter, als wenn die Zustandscontroller in einem anderen Gehäuse als die RAMs **1803** untergebracht sind. Die Anordnung von RAMs **1803** und Zustandscontrollern am selben Ort bietet technische Auswahlmöglichkeiten zum Speichern der Köpfe und Enden der Liste. Diese Auswahlmöglichkeiten sind Onboard-Register für wenige Linked-List-Schlangen, statische RAM **1803** für eine moderate Menge von Linked-List-Schlangen oder dynamische RAM **1806** für eine große Anzahl von Linked-List-Schlangen.

[0093] Ein Blockschaubild des Zustandscontrollers **1804** ist in Fig. 17 gezeigt.

[0094] Der Zustandscontroller **1804** wird durch die Arbitrierungs- und Sequenzierungslogik gesteuert, die den Fluss von Informationen vom Vordergrundport hereinführt und die RAM-Speicherbank **1803** vor neuer ankommender Aktivität schützt, wenn sie entweder mit einem Vordergrund- oder einem Hintergrundtransfer beschäftigt sind. Außerdem überwacht der Zustandscontroller die Zustände in der RAM-Speicherbank **1803** und bestimmt, wann Interaktionen mit dem räumlich abgesetzten RAM **1806** erfolgen sollen. Die Zustandscontroller **1804** passen zusammen in ein System mit einem Hintergrundzugriffsmultiplexer **1808**, einem räumlich abgesetzten RAM **1806** und einem Zugangsflussregler, wie in Fig. 18 gezeigt.

[0095] Der Zustandscontroller **1804** in Fig. 17 fungiert als eine Schnittstelle zwischen seiner zugehörigen RAM-Speicherbank **1803**, dem Zugangsflussregler **1801** und dem Hintergrundzugriffsmultiplexer **1808**. Die Verbindungen zu diesen Elementen sind detailliert in Fig. 17 gezeigt. Der Zustandscontroller **1804** enthält einen Multiplexer **1702** und ein Arbitrierungs- und Sequenzierungslogikelement **703**. An seiner unteren Seite ist der Multiplexer **1702** mit Pfaden **1710** und **1711** verbunden, die zu einem Bestandteil der Busse **1809-1** bis **1809-8** von Fig. 18 werden. Über diesen Pfad tauscht der Multiplexer Daten zu Lese- und Schreibvorgängen mit seiner zugehörigen RAM-Speicherbank über Pfad **1710** aus. Pfad **1711** ist ein bidirektionaler Steuerpfad, der es dem Zustandscontroller **1804** über den Multiplexer **1702** ermöglicht, den Betrieb seiner zugehörigen RAM-Speicherbank **1803** zu steuern. Der RAM-Datenpfad **1710** kann über den Multiplexer entweder mit dem Datenpfad **1704** oder dem Datenpfad, der zu dem Hintergrundzugriffsmultiplexer **1808** führt, verbunden werden, oder er kann über den Datenpfad **1705** und den Bus **1802** mit dem Zugangsflussregler **1801** verbunden werden. Diese Datenpfade können sowohl für Lese- als auch Schreibvorgänge verwendet werden.

[0096] Der RAM-Steuerpfad **1711** unten am Multiplexer **1702** ist über den Pfad **1712** und das Arbitrierungs- und Steuerlogikelement **1703** mit den Pfaden **1707** und **1706** verbunden. Der Pfad **1711** des Multiplexers kann nur mit einem der Pfade **1707** und **1706** auf einmal verbunden werden. Wenn er mit dem Pfad **1706** verbunden ist, so wird er über den Pfad **1810** weitergeführt, um den Betrieb des Hintergrundzugriffsmultiplexers **1808** und seines zugehörigen, räumlich abgesetzten RAM **1806** für sowohl Lese- als auch Schreibvorgänge zu steuern. Wenn der Pfad **1711** über das Element **1703** mit dem Pfad **1707** verbunden wird, so wird er über den Bus **1802** zum Zugangsflussregler **1801** weitergeführt. Das Arbitrierungs- und Sequenzierungslogikelement **1703** enthält die Informationen und die Logik, die erforderlich sind, um den Zugangsflussregler **1801** bei seinem Datenaustausch mit einem Zustandscontroller **1804** für sowohl Lese- als auch Schreibvorgänge zu steuern. Die Arbitrierungs- und Sequenzierungslogik **1703** kommuniziert des Weiteren über den Bus **1706** und **1810** mit dem Hintergrundzugriffsmultiplexer **1808**, um seine Operationen zu steuern, wenn der räumlich abgesetzte RAM **1806** Daten von einer RAM-Speicherbank **1803** empfängt, sowie Operationen zu steuern, bei denen der räumlich abgesetzte RAM **1806** Daten an einen Zustandscontroller **1804** zum Eintragen in die RAM-Speicherbank, die dem Zustandscontroller zugeordnet ist, sendet.

[0097] Der Zustandscontroller **1804** hat vier High-Level-Funktionen bei seinem Austausch von Steuerung und Daten mit seiner zugehörigen RAM-Speicherbank **1803** und mit dem Zugangsflussregler **1801** und mit dem räumlich abgesetzten RAM **1806** über den Hintergrundzugriffsmultiplexer **1808**. Diese vier High-Level-Funktionen werden als nächstes beschrieben.

[0098] Die erste Funktion, die durch den Zustandscontroller **1804** von Fig. 17 ausgeführt wird, ist die Initiierung und Steuerung der ankommenden Zugriffsanforderungssequenzen, die zur Übertragung von Informationen vom Zugangsflussregler **1801** zu Lese- oder Schreibanforderungen gehören, und dabei auch die Steuerung seiner zugehörigen RAM-Speicherbank **1803**, um im Fall von Schreibanforderungen vom Zugangsflussregler **1801** Daten in die RAM-Speicherbank **1803** zu schreiben und im Fall von Leseanforderungen vom Zugangsflussregler **1801** Daten aus der RAM-Speicherbank **1803** zu lesen.

[0099] Eine zweite Funktion, die durch den Zustandscontroller von Fig. 17 ausgeführt wird, ist das Reagieren auf Auslösesignale, die einen Pufferfüllstand innerhalb seiner zugehörigen RAM-Speicherbank **1803** erkennen. Dieser Auslöser zeigt an, dass seine Puffer innerhalb seiner zugehörigen RAM-Speicherbank entweder hinreichend aufgebraucht oder erschöpft sind. Wenn Puffer innerhalb seiner zugehörigen RAM-Speicherbank hinreichend aufgebraucht sind, so wird ein Schreibvorgang an den räumlich abgesetzten RAM **1806** ausgelöst. Wenn Puffer innerhalb seiner zugehörigen RAM-Speicherbank hinreichend erschöpft sind, so wird ein Lesevorgang von dem räumlich abgesetzten RAM **1806** ausgelöst.

[0100] Eine dritte Funktion, die durch den Zustandscontroller **1804** ausgeführt wird, ist die Initiierung und Verwaltung von Transfers von seiner zugehörigen RAM-Speicherbank **1803** zu dem räumlich abgesetzten RAM **1806** und ebenso die Verwaltung von Datentransfers in der umgekehrten Richtung zurück zur RAM-Speicherbank **1803** von dem räumlich abgesetzten RAM **1806**.

[0101] Eine vierte Funktion die durch den Zustandscontroller **1804** ausgeführt wird, ist das Warten auf ein Signal vom Multiplexer **1702** und die Initiierung eines Transfers von und zu dem räumlich abgesetzten RAM **1806** bei Erhalt dieses Signals vom Multiplexer **1702**.

[0102] Eine weitere Funktion, die durch den Multiplexer **1702** ausgeführt wird, ist es auszuwählen, welche bietende RAM-Speicherbank **1803** Zugriff zu dem räumlich abgesetzten RAM **1806** in dem Fall haben soll, dass mehrere RAM-Speicherbänke **1803** gleichzeitig Zugriff auf den räumlich abgesetzten RAM **1806** anfordern. Eine weitere Funktion, die durch den Multiplexer **1702** ausgeführt wird, ist die Initiierung der Transfer- und Zeitplanungsfunktionen, die den Operationen zwischen dem räumlich abgesetzten RAM **1806** und einer bietenden RAM-Speicherbank **1803** zugehörig sind, im Fall von Abhängigkeiten zwischen diesen Transfers. Solche Abhängigkeiten können aus Zugriffsströmen in das und aus dem Speichersystem entstehen.

[0103] Eine weitere Funktion, die durch den Multiplexer **1702** ausgeführt wird, ist es, die RAM-Speicherbank **1803** so zu steuern, dass sie einen Schreibeintrag vom räumlich abgesetzten RAM **1806** anweist. Der Multiplexer **1702** gewährt den Zugriff auf den räumlich abgesetzten RAM **1806** und routet Informationen zwischen dem räumlich abgesetzten RAM **1806** und der RAM-Speicherbank **1803**.

[0104] Tabelle 2 zeigt, dass eine Konkurrenz um die RAM-Bank **1803** die Leistung eines Systems, das auf herkömmlichen SDRAM basiert, begrenzen kann. Die Besprechung von Fig. 14 demonstriert, dass diese Beschränkung so schwerwiegend sein kann, dass Systeme um eine RAM-Speicherbank-Konkurrenz herum konstruiert wurden. Das Design von Fig. 14 vermeidet Konkurrenz bei jedem Zyklus durch Strukturieren der Zugriffe für das Zeitintervall einer vollen RAM-Speicherzykluszeit im Gegensatz zu einer Konkurrenz auf der Basis einer Prognostizierung der RAM-Bank-Verfügbarkeit. Das bedeutet, dass die RAM-Speicherbank konstruktionsbedingt langsamer läuft als mit optimaler Geschwindigkeit. Die Fig. 12 und Fig. 13 beschreiben ein System, das die Bank-Verfügbarkeit prognostiziert, aber im Fall von Konkurrenz sehr viel zusätzliche Logik haben muss. Das Problem bei dieser Implementierung ist, dass Konkurrenz die Leistung schmälert und dass die zusätzliche Hardware, die erforderlich ist, nicht genügend Leistungszuwachs bringt.

[0105] Die Fig. 15 und Fig. 16 zeigen einen verbesserten Betrieb durch Erhöhen der Anzahl von RAM-Speicherbänken, um die Konkurrenzintensität zu senken. Dies erfordert mehr Hardware-Logik. Die Beziehung zwischen der Erhöhung der Leistung und der Vermehrung der Hardware ist so, dass eine akzeptable Menge an zusätzlicher Hardware einen spürbaren Unterschied bei der Leistung bewirken kann. Um die Zugriffe auf die vielen RAM-Speicherbänke zu koordinieren, braucht man spezialisierte Zustandslogik, wie beispielsweise die, die in Fig. 17 gezeigt ist, die Hardware-Ressourcen aufbraucht. Diese Ressourcen finden sich innerhalb handelsüblicher FPGAs. Um die Leistung zu maximieren, ist ein breiter, maskierbarer Datenbus bevorzugt. Er sollte breit sein, um Burstdaten reibungslos aufnehmen zu können, und sollte überdies maskierbar sein, um die Speicherung der kleinsten Datenmenge zu erleichtern. Speicher, der auf diese Weise konfigurierbar ist, ist in handelsüblichen FPGAs verfügbar. Aber dieser Speicher steht nur in kleineren Mengen zur Verfügung, die für größere Pufferungsaufgaben nicht ausreichend sind, wie beispielsweise die Pufferung von 300 Millisekunden einer OC-768-Glasfaser, wie zuvor besprochen. Außerdem steht innerhalb eines integrierten Schaltkreises nur eine begrenzte Menge Platz zur Verfügung. Was für RAM-Speicher verwendet wird, kann nicht für die Zustandscontrollerlogik verwendet werden und umgekehrt.

[0106] Aber mehr RAM-Bänke bedeuten höhere Leistung, und jede Bank muss ihren eigenen Zustandscontroller haben. Es besteht also ein Konflikt zwischen der Erfüllung von Pufferungsanforderungen und der Leistung des Systems.

[0107] Ein Lösung dieses Konflikts besteht darin, die Menge des Speichers zu begrenzen, der auf der FPGA-RAM-Bank **1803** installiert ist. Es wird nur auf den Kopf und das Ende einer Linked-List zugegriffen, und die mittleren Elemente einer Linked-List sind immer inaktiv, bis sie an den Kopf der Liste rutschen. Indem die mittleren Elemente der Linked-List aus der FPGA-RAM-Bank **1803** zu dem räumlich abgesetzten RAM **1806** ausgelagert werden, wird eine höhere Leistung ermöglicht.

[0108] Ein System, das diesen Kompromiss implementiert, ist in den Fig. 17 und Fig. 18 gezeigt.

[0109] Diese Lösung ist auch aus wirtschaftlicher Sicht vernünftig. Konfigurierbare RAM-Speicherbänke haben einen viel höheren pro-Bit-Preis als handelsüblicher SDRAM. Wenn auf die räumlich abgesetzten RAMs **1806**, die die mittleren Elemente der Linked-List halten, nicht zugegriffen wird, so kommt es nicht zu der Zykluszeitverlängerung, die mit diesem Speicher verbunden ist. Letztendlich muss auf den räumlich abgesetzten RAM **1806** zugegriffen werden, um die Elemente in der Mitte der Liste zu speichern und abzurufen. Aber die oben angesprochene Konfigurierbarkeit der vielen Speicherbänke auf dem FPGA ermöglicht ein Design, das das mit dem Burstmodus der SDRAM-RAM-Bank **1803** kompatibel ist. Man könnte den Durchsatz des Systems aus vielen Speicherbänken an den SDRAM angleichen, wodurch man ein ausgeglichenes Design erhält. Dies minimiert die Kosten der SDRAM-Zykluszeit, indem es dem SDRAM **1803** möglich ist, im Burstmodus zu arbeiten, so dass die Zykluszeit mit den Daten, die auf dem Datenbus synchron getaktet sind, verknüpft werden kann. Dadurch werden die Speicher auch in den Arbeitsmodus zurückversetzt, für den sie ausgelegt wurden.

[0110] Linked-List-Puffer werden über den Hintergrundzugriffsmultiplexer **1808** in dem räumlich abgesetzten RAM **1806** abgelegt oder von dort abgerufen. Dadurch können die Vordergrundzugriffe von diesem zusätzlichen Verkehr befreit vonstatten gehen. Das ist wichtig, weil sich das durch Tabelle 2 veranschaulichte probabilistische Modell auf einen verfügbaren Vordergrundbus stützt, wie in Fig. 18 gezeigt. Ein Blockieren des im Verlauf befindlichen Hintergrundtransfers vom Vordergrundbus **1810** zu dem räumlich abgesetzten RAM **1806** verkompliziert das Modell, das zur Erstellung von Tabelle 2 verwendet wurde, erheblich. Das würde die Leistung schmälern. Es ist der Hintergrundzugriffsbus **1810** gezeigt.

[0111] Fig. 18 offenbart eine Linked-List-Maschine **1800**, die die vorliegende Erfindung verkörpert. Die Linked-List-Maschine **1800** ist als mit einen Kommunikationssystem **1811** verbunden dargestellt, das ankommende und abgehende Pfade **1812** und **1813** aufweist, die mit Ports **1817** und **1818** verbunden sind. Das System enthält einen Prozessor **1814** und enthält des Weiteren Pfade **1815** und **1816**, die sich über den Pfad **1819** zu Zugangsflussregler **1801** fortsetzen. Während seines Betriebes vollführt das System **1811** Lese- und Schreibvorgänge mit den Speichern der Linked-List-Maschine **1800**, um die Daten zu speichern, die von den Ports **1817** und **1818** für ihren Betrieb benötigt werden.

[0112] Der Zugangsflussregler **1801** und der Bus **1802** sind mit mehreren Zustandscontrollern **1804** verbunden, von denen jeder einer der RAM-Speicherbänke **1803** zugeordnet ist. Der Zugangsflussregler **1801** empfängt Schreibanforderungen vom System **1811**, mit denen die Speicherung von Informationen in einer der RAM-Speicherbänke **1803** angefordert wird. Der Zugangsflussregler **1801** empfängt und speichert diese Zugriffsanforderungen und verteilt sie selektiv auf die verschiedenen Zustandscontroller **1804**, damit die Daten in eine zugehörige RAM-Speicherbank **1803** eingetragen werden. Der Prozess läuft bei Speicherleseoperationen in der umgekehrten Richtung ab, wenn der Zugangsflussregler **1801** eine Leseanforderung von System **1811** empfängt und – über die Zustandscontroller **1804** – veranlasst, dass die RAM-Speicherbank **1803**, die die angeforderten Daten enthält, ausgelesen wird und die Daten über den Zustandscontroller **1804** und den Bus **1802** zurück zum Zugangsflussregler **1801** geführt werden, der sie zum System **1811** sendet.

[0113] Die RAM-Speicherbankbausteine **1803** sind Hochgeschwindigkeitselemente mit einer relativ kleinen Speicherkapazität, die durch dem räumlich abgesetzten RAM **1806** vergrößert wird, der Informationen von einer RAM-Speicherbank **1803** empfangen und speichern kann, wenn die Informationen nicht sofort von der RAM-Speicherbank benötigt werden. Der räumlich abgesetzte RAM **1806** wird in seinen Betrieb durch den Hintergrundzugriffsmultiplexer **1808** und die Buspfade **1810** unterstützt, von denen jeder zu einem eindeutig zugeordneten Zustandscontroller **1804** und seiner zugehörigen RAM-Speicherbank **1803** führt. Auf diese Weise kann eine RAM-Speicherbank **1803**, die voll oder leer wird, dies an ihren zugehörigen Zustandscontroller **1804** melden, der dann über die Buspfade **810** mit dem Hintergrundzugriffsmultiplexer **1808** kommuniziert. Der Multiplexer **1808** unterstützt den Zustandscontroller **1804** beim Lesen von Informationen aus seiner RAM-Speicherbank **1803** und beim Übertragen der Informationen zu dem räumlich abgesetzten RAM **1806** zum temporären Speichern, bis die Informationen wieder von der RAM-Speicherbank **1803**, von der sie kamen, benötigt werden. Zu diesem Zeitpunkt signalisiert die RAM-Speicherbank **1803** ihrem Zustandscontroller **1804**, dass der räumlich abgesetzte RAM **1806** Informationen enthält, die in Kürze von der RAM-Speicherbank be-

nötigt werden. Der Hintergrundzugriffsmultiplexer **1808** und der Zustandscontroller **1804** veranlassen dann zusammen, dass der entsprechende Teil des räumlich abgesetzten RAM **1806** ausgelesen wird und dass die Informationen zur RAM-Bank **1803**, von der sie stammen, zurück übertragen werden. Der räumlich abgesetzte RAM **1806** ist ein relativ langsames Massenspeicherelement, das effizient Informationen speichern kann, die von einer RAM-Speicherbank **1803** überlaufen, oder das Informationen zu einer unausgelasteten RAM-Speicherbank **1803** leiten kann.

[0114] Ein Aspekt der vorliegenden Erfindung beinhaltet das Verfahren und die Vorrichtung von Fig. 18, wobei ein Schreibvorgang in eine RAM-Speicherbank **1803** durch folgende Schritte ausgeführt wird: Schreiben von Linked-List-Informationen in eine RAM-Speicherbank **1803** über ihren Zustandscontroller **1804**; Fortsetzen des Schreibvorgangs zur RAM-Speicherbank **1803**, bis sie sich einem Voll-Zustand nähert; Fortsetzen des Schreibens zusätzlicher Informationen vom Zugangsflussregler **1801** in die RAM-Speicherbank, während gleichzeitig einige der neu empfangenen Informationen aus der RAM-Speicherbank **1803** über ihren Zustandscontroller **1804** und den Hintergrundzugriffsmultiplexer **1808** in den räumlich abgesetzten RAM **1806** hinein ausgelesen werden. Diese Informationen bleiben in dem räumlich abgesetzten RAM **1806**, bis sie später von der RAM-Speicherbank **1803**, von der sie kamen, benötigt werden.

[0115] Die System von Fig. 18 vollführt ein Auslesen von Daten aus einer RAM-Speicherbank **1803** mittels folgender Schritte: Meldung an den Zustandscontroller **1804**, der zu der RAM-Speicherbank **1803** gehört, welche die angeforderten Daten enthält; Initiieren eines Auslesens der gewählten Daten über den Zustandscontroller **1804** und den Bus **1802** zurück zum Zugangsflussregler **1801**; Fortsetzen des Auslesens der gewählten RAM-Speicherbank **1803** und gleichzeitiges Feststellen, dass durch diese Lesevorgänge alle interessierenden Daten aus der ausgewählten RAM-Speicherbank **1803** abgezogen wurden, und dass einige der interessierenden Daten, die bei anschließenden Lesevorgängen benötigt werden, momentan in dem räumlich abgesetzten RAM **1806** gespeichert werden; Initiieren einer Prefetch-Auslesung des räumlich abgesetzten RAM **1806**, um die Informationen zurück zur RAM-Speicherbank **1803** zu übertragen, von der sie stammten, bevor sie angefordert wurden; und Fortsetzen einen Auslesens der gewählten RAM-Speicherbank **1803** und erforderlichenfalls Fortsetzen des Auslesens von Daten aus dem räumlich abgesetzten RAM **1806** zurück zur RAM-Speicherbank **1803**, die gerade ausgelesen wird. Diese Operation setzt sich fort, bis die gesamten Informationen, die durch den Zugangsflussregler **1801** für den Lesebetrieb angefordert wurden, erhalten wurden.

[0116] Die Hochgeschwindigkeitsspeicherbausteine der RAM-Speicherbank **1803** und der mit langsamerer Geschwindigkeit arbeitende Massenspeicher des räumlich abgesetzten RAM **1806** wirken beim Speichern von Daten, welche die Kapazität der Hochgeschwindigkeits-RAM-Speicherbänke **1803** übersteigen, zusammen. Der räumlich abgesetzte RAM **1806** empfängt diese Daten zu Schreibvorgängen von einer RAM-Speicherbank, in die geschrieben wird, während es der RAM-Speicherbank **1803** ermöglicht wird, weiterhin weitere Daten mit hoher Geschwindigkeit vom Zugangsflussregler **1801** zu empfangen. Der Prozess arbeitet bei einem Lesevorgang in der umgekehrten Richtung, wenn eine RAM-Speicherbank **1803** zunächst mit hoher Geschwindigkeit ausgelesen wird. Die Daten, die von der RAM-Speicherbank benötigt und in dem räumlich abgesetzten RAM **1806** gespeichert werden, werden zur RAM-Speicherbank **1803** zurück übertragen, wenn die RAM-Speicherbank **1803** in einer Prefetch-Anordnung keine interessierenden Informationen mehr hat. Der Lesebetrieb kann fortdauern, wobei alle relevanten Informationen in dem räumlich abgesetzten RAM **1806** zurück zur Hochgeschwindigkeits-RAM-Speicherbank **1803** übertragen werden, die weiterhin mit einer hohen Datenrate ausgelesen wird, bis die gesamten Informationen, die durch den Zugangsflussregler **1801** angefordert wurden, durch die Hochgeschwindigkeits-RAM-Speicherbank **1803** über den Zustandscontroller **1804** und den Bus **1802** zurück zum Zugangsflussregler **1801** versandt wurden. Oder der Zustandscontroller **1804** und die RAM-Speicherbank **1803** können einen Schreibvorgang separat von dem vorherigen Lesevorgang ausführen, der den Abruf im Hintergrund auslöste. Weil sich der Prefetch-Vorgang beim Auslösen automatisch vollzieht, während ein bestimmter Zustandscontroller **1804** und eine bestimmte RAM-Speicherbank **1803** mit der Übertragung im Hintergrund beschäftigt sind, können die anderen Zustandscontroller **1804** und RAM-Speicherbänke **1803** ungehindert Operationen unabhängig von diesem Hintergrundbetrieb ausführen.

[0117] Ein weiterer Aspekt der Erfindung ist die Verwendung der Zustandscontroller **1804** zum Anlegen von Potenzialen über Pfade **1820** an den Zugangsflussregler **1801**, die anzeigen, ob die RAM-Speicherbänke **1803**, die jedem Zustandscontroller **1804** zugeordnet sind, momentan mit einem existierenden Lese- oder Schreibvorgang beschäftigt sind oder zur Verfügung stehen, um Anforderungen für einen neuen Lese- oder Schreibvorgang zu empfangen. Weil die Speicherelemente der RAM-Speicherbankbausteine **1803** mit der hohe Datenrate eines optischen Busses arbeiten, kann eine RAM-Speicherbank **1803** Lese- oder Schreibvorgänge mit den Geschwindigkeiten ausführen, die für Glasfaserbusse angemessen sind. Infolge dessen wird durch den Zustandscontroller über den Pfad **1820** ein Belegt-Signal an den Zugangsflussregler **1801** angelegt,

das seine Verfügbarkeit oder Nichtverfügbarkeit seiner zugehörigen RAM-Speicherbank anzeigt. Dieses Signal dauert nur die wenigen Nanosekunden, die von dem RAM-Speicherbankbaustein **1803** benötigt werden, um den Lese- oder Schreibvorgang auszuführen. Somit umfasst das Anlegen dieser belegt-/nicht-belegt-Potenziale an die Pfade **1820** durch den Zustandscontroller **1804** eine Konkurrenzeinrichtung, die es dem Zugangsflussregler **1801** und seinem Element **1821** ermöglicht, den belegt-/nicht-belegt-Zustand der RAM-Speicherbänke **1803** zu überwachen. Dies beseitigt jegliche Konkurrenzverzögerung infolge der zum Stand der Technik gehörenden komplexen Logikschaltungselemente oder eine Konkurrenzanordnung, die ein willkürliches vorgegebenes Mindestzeitintervall zwischen Lese- oder Schreibvorgängen zu einer RAM-Speicherbank **1803** auferlegt. Auf diese Weise stellen der Zugangsflussregler **1801**, die Pfade **1820** und der Zustandscontroller **1804** eine effiziente und mit hoher Geschwindigkeit arbeitende Konkurrenzeinrichtung bereit, die mit den Nanosekundenraten von optischen Bussen arbeitet. Diese verbesserte Hochgeschwindigkeits-Konkurrenzanordnung ermöglicht einen höheren Durchsatz von Daten, die durch das System **1811** bedient werden, weil ihre ankommenden und abgehenden Schlangen, die den Ports **1817** und **1818** zugeordnet sind, Daten mit einer höheren Rate austauschen und verarbeiten können, weil sie von den Hochgeschwindigkeitselementen der RAM-Speicherbank **1803** bedient werden. Somit kann die Linked-List-Maschine **1800** von Fig. 18 das Lesen und Schreiben der Datenschlangen, die von den Ports **1817** und **1818** benötigt werden, mit den Geschwindigkeiten ausführen, die denen von Glasfaser-Links angepasst sind.

Beschreibung von Fig. 19

[0118] Fig. 19 offenbart eine typische Linked-List, die fünf Puffer 1 bis 5 umfasst. Die Linked-List mit fünf Puffern wird nicht in derselben RAM-Bank **1803** von Fig. 18 gespeichert. Vielmehr werden die fünf Puffer nach dem Zufallsprinzip in fünf separaten Bänken **1803** gespeichert. Jeder Puffer hat einen ersten Teil, der die physikalischen Informationen oder Daten enthält, die von dem System gespeichert und verarbeitet werden sollen. Der untere Abschnitt jedes Puffers enthält eine Link-Feld-Adresse zu der RAM-Bank, die den nächsten Puffer der Linked-List speichert. Puffer 1 in Fig. 19 speichert physikalische Informationen in seinem oberen Abschnitt und speichert die Link-Feld-Adresse von 0100 in seinem unteren Abschnitt. Die Adresse der RAM-Bank, die den Puffer 1 speichert, ist 000/00, wie rechts neben dem Puffer 1 gezeigt.

[0119] Puffer 2 des Linked-List wird in der RAM-Adresse 010/00 gespeichert, wie in dem Link-Feld von Puffer 1 spezifiziert. Der obere Abschnitt von Puffer 2 enthält physikalische Informationen (Daten). Der untere Abschnitt enthält die Link-Feld-Adresse 01001, die die Identität und den Ort der RAM-Bank spezifiziert, die den Puffer 3 der Linked-List speichert.

[0120] Puffer 3 wird in der RAM-Bank-Adresse 010/01 gespeichert, wie durch das Link-Feld von Puffer 2 spezifiziert. Das Link-Feld von Puffer 3 enthält die Adresse 11010, die den Ort des vierten Puffer der Linked-List angibt.

[0121] Der vierte Puffer wird in der RAM-Adresse 110/10 gespeichert, wie durch das Link-Feld von Puffer 3 spezifiziert. Gleichermaßen wird der fünfte und letzte Puffer der Linked-List in der RAM-Adresse 101/11 gespeichert, wie durch das Link-Feld von Puffer 4 spezifiziert. Der obere Abschnitt von Puffer 5 zeigt an, dass es sich um den Kopfpuffer der Frei-Liste handelt, der für die Verwendung zur Verfügung steht.

[0122] Die Puffer in einer Linked-List werden nach dem Zufallsprinzip in separaten RAM-Bänken von Fig. 18 gespeichert. Diese Zufälligkeit ist für die Effizienz der Datenverarbeitung und der Steuerungsoperationen des Systems wünschenswert und erforderlich. Diese Zufälligkeit ist besonders erforderlich und nützlich, um den gewünschten Betrieb der Konkurrenzeinrichtungen der Erfindung zu bewerkstelligen.

Beschreibung der Fig. 20 und Fig. 21

[0123] Fig. 20 zeigt, wie der Kopf einer Linked-List-Schlange in einer RAM-Speicherbank **1803** erschöpft werden kann und einen Zugriff auf den RAM-Massenspeicher **1806** verlangt. Die gezeigte Linked-List-Schlange hat – in ihrer jeweiligen Reihenfolge – Puffer mit einem Inhalt Q, F, R und Z. In Fig. 21 liest der erste Zugriff des Taktzeitraums A1 aus dem RAM **1803** den die Adresse enthaltenden Inhalt Q in Taktzeiträumen 7 und 8 von Fig. 21. Aber der Zugriff A1 ruft den Kopf der Schlange zur Verwendung vom Hochgeschwindigkeitsspeicher **1803** ab und speichert das nächste Element F in dem räumlich abgesetzten RAM **1806**. Somit löst der Zugriff A1 eine Hintergrundanforderung Rq1 in Taktzeitraum 2 über den Zustandscontroller **1804** der RAM-Bank **1803**, die an dem Zugriff A1 beteiligt ist, aus. Diese Zugriffsanforderung des räumlich abgesetzten Speichers **1806** wird durch den Hintergrundzugriffsmultiplexer **1808** verarbeitet und dann durch den räumlich abgesetzten Speicherzugriff A1B im Taktzeitraum 3 von Fig. 21 zurückgesendet. Es ist zu beachten, dass Zugriffe des

räumlich abgesetzten RAM **1806** gleichzeitig mit dem Fluss von Zugriffen auf die RAM-Bank **1803** erfolgen. Die Daten D1B werden aus dem räumlich abgesetzten RAM **1806** während der Taktzeiträume 9 und 10 abgerufen. Diese Daten sind das Linked-List-Element F. Das Element F wird in ein leeres Element geschrieben, das der Frei-Liste entnommen wurde, deren Adresse sich in der RAM-Bank 5 befindet, wobei ein Zugriff A5 bei Taktzyklus 11 beginnt. Die Verbindungen werden vollständig beibehalten. Somit liegt der Kopf der Schlange, der das Element F enthält, wieder in der Hochleistungs-RAM-Bank **1803**–**1805**. Die RAM-Bank **1803**–**1805**, die das Element Q hält, und das räumlich abgesetzte Speicherelement, das zuvor den Inhalt F hielt, werden zu ihren jeweiligen Frei-Listen zurückgesandt.

[0124] Die Unkorreliertheit der Zugriffe auf den Hochleistungs-RAM **1803** wird beibehalten, wodurch das Modell, das zur Tabelle 2 führt, intakt bleibt und die Aussicht auf eine effizientere Nutzung erhalten bleibt, wie in Fig. 20 durch des Fehlen einer Lückenbildung in der Zugriffssteuerung demonstriert wird. Wir erinnern uns aus der vorangegangenen Besprechung, dass durch Konkurrenz Lücken entstehen. Korrelierter Verkehr, das heißt sequenzielle Zugriffe auf dieselbe Bank, verursacht Konkurrenz.

Beschreibung von Fig. 22

[0125] Fig. 22 offenbart den Prozess der vorliegenden Erfindung, der Leseanforderungen ausführt, die durch den Zugangsflussregler **1801** initiiert wurden und das Lesen einer Linked-List anfordern, deren Puffer nach dem Zufallsprinzip in den verschiedenen RAM-Bänken **1803-1** bis **1803-8** von Fig. 18 (im Weiteren die RAM-Bänke **1803**) gespeichert werden. Der Prozess beginnt in Schritt **2201**, wenn der Zugangsflussregler **1801** Anweisungen vom Prozessor **1814** empfängt, die einen Lesevorgang anfordern. Die einzelnen Puffer der Linked-List werden der Reihe nach einzeln gelesen und werden nach dem Zufallsprinzip in den verschiedenen RAM-Bänken **1803** gespeichert. Das Lesen jedes Puffers erfordert eine separate Leseanforderung durch den Zugangsflussregler **1801**.

[0126] Die erste Leseanforderung wird durch den Schritt **2202** empfangen und zum Schritt **2203** weitergetragen, wo festgestellt wird, ob bezüglich der Anzahl von Elementen, die aus der RAM-Bank **1803** zu lesen sind, eine Schwelle überschritten ist. Wie zuvor beschrieben, erfordert das Lesen einer Linked-List das Lesen des ersten Puffers (des Kopfes) der Liste von der RAM-Bank **1803**, in der der Kopfpuffer gespeichert ist. Der übrige Teil der Ausführung der Linked-List erfordert das Lesen von Zwischenspeichern der Linked-List, die in dem räumlich abgesetzten RAM **1806** gespeichert sind und die abgerufen und in die RAM-Bänke **1803** eingegeben werden müssen. Der effiziente Transfer von Puffern aus dem räumlich abgesetzten RAM **1806** zurück zu den RAM-Bänken **1803** erfordert, dass mehrere solcher Anforderungen an den Hintergrundzugriffsmultiplexer **1808** und im Gegenzug an den räumlich abgesetzten RAM **1806** gerichtet werden, um den Schritt effizient auszuführen. Aus diesem Grund gibt es das Schwellenerkennungselement **2203**, so dass mehrere solcher Anforderungen an den räumlich abgesetzten RAM **1806** gleichzeitig – anstatt einzeln eine nach der anderen – weitergeleitet werden.

[0127] Nehmen wir zunächst einmal an, dass das Element **2203** feststellt, dass die Schwelle nicht überschritten ist. In diesem Fall wird auf den räumlich abgesetzten RAM **1806** nicht sofort zugegriffen, und der Prozess schreitet zu Schritt **2204** voran, der die RAM-Bank liest, die durch die Leseanforderung für den ersten Puffer (Kopf) der Linked-List identifiziert wurde. Dieser Pufferort wird gelesen und temporär gespeichert, und der Prozess schreitet zu Schritt **2205** voran, der die Leseinformationen an den Zugangsflussregler **1801** zurücksendet. Der Prozess schreitet dann zu Schritt **2206** voran, der anzeigt, dass die RAM-Bänke **1803** bereit sind, um die nächste Zugriffsanforderung vom Zugangsflussregler **1801** zu empfangen. Ein Endpuffer wird in der gleichen Weise gelesen, wenn die Linked-List aus nur einem einzigen Puffer besteht.

[0128] Nehmen wir als nächstes an, dass das Element **2203** feststellt, dass die neu angekommene Leseanforderung von Schritt **2202** bewirkt, dass die Schwelle bezüglich der Anzahl der Puffer, die in der RAM-Bank **1803** für einen Lesevorgang verfügbar sind, überschritten wird.

[0129] In diesem Fall liest der Schritt **2211** den Kopfpuffer und schreitet zu Schritt **2220** voran, der die Leseinformationen von Schritt **2211** zum Zugangsflussregler **1801** übermittelt. Als nächstes fordert Schritt **2221** ein Lesen der Zwischenspeicher nahe dem Kopf der Liste in RAM **1806** an. Dies beinhaltet eine Anforderung für den Kopf der Liste. Dabei schickt Schritt **2221** eine Leseanforderung für einen neuen Kopfpuffer der Liste zum Hintergrundzugriffsbus **1810**, der die spezifizierte RAM-Bank **1806** bedient. Als nächstes ruft Schritt **2222** die mehreren Zwischenspeicher der Liste von dem räumlich abgesetzten RAM **1806** ab. Der Prozess schreitet dann zu Schritt **2223** voran, der anzeigt, dass der räumlich abgesetzte RAM **1806** für einen weiteren Zugriff bereit ist. Der Prozess schreitet ebenfalls zu Schritt **2212** voran, der in die RAM-Bank **1803** die Informationen

schreibt, die während des Schrittes **2222** aus dem räumlich abgesetzten RAM **1806** ausgelesen wurden und die in eine spezifizierte RAM-Bank **1803** geschrieben wurden. Die Informationen enthalten die Bildung eines neuen Kopfpuffers der Liste in der spezifizierten RAM-Bank **1803**. Der Prozess schreitet dann zu Schritt **2205** voran, der die Informationen zum Zugangsflussregler **1801** weiterleitet. Der Prozess schreitet dann zu Schritt **2206** voran, der die Vollendung der Leseanforderung darstellt und anzeigt, dass die RAM-Bänke **1803** für die nächste Zugriffsanforderung bereit sind.

Beschreibung von Fig. 23

[0130] Fig. 23 offenbart die Schritte, die erforderlich ist, um eine Schreibanforderung auszuführen, die vom Zugangsflussregler **1801** empfangen wurde. Der Prozess beginnt mit Schritt **2301**, in dem der Zugangsflussregler **1801** die Schreibanforderung zum Bus **1802** leitet. Schritt **2302** gibt die Schreibanforderung in die Zustandscontroller **1804** von Fig. 18 ein. Schritt **2203** stellt fest, ob die Schwelle für beschriebene Elemente nahe dem Ende der Liste überschritten ist. Wenn die Schwelle nicht überschritten ist (was für eine neue Liste der Fall wäre), so wird der letzte Puffer (das Ende) der Liste in eine RAM-Bank **1803** geschrieben. Der Prozess schreitet dann zu Schritt **2305** voran, der dem Zugangsflussregler **1801** anzeigt, dass eine Zugriffsanforderung an den räumlich abgesetzten RAM **1806** nicht erforderlich ist. Der Prozess schreitet dann zu Schritt **2306** voran, der anzeigt, dass die RAM-Bank **1803** für den nächsten Zugriff bereit ist.

[0131] Nehmen wir an, dass Schritt **2303** feststellt, dass die Schwelle für die Schreibanforderung überschritten ist. In diesem Fall schreitet der Prozess zu Schritt **2311** voran, der jedes Ende der Liste abruft, die in der RAM-Bank **1803** gehalten wird. Der Prozess schreitet dann zu Schritt **2321** voran, der den in Schritt **2311** abgerufenen Endpuffer zum Hintergrundzugriffsbus **1810** sendet, um ihn in dem räumlich abgesetzten RAM **1806** abzulegen. Als nächstes aktualisiert Schritt **2322** das Link-Feld vom nächsten bis letzten Puffer im räumlich abgesetzten RAM **1806** auf den Ort des Puffers, der in Schritt **2321** geschrieben wurde, weil die Puffer, der in dem Link-Feld des nächsten bis letzten Puffers angegeben ist, infolge der Schritte **2311** und **2321** seinen Ort von der RAM-Bank **1803** zu dem räumlich abgesetzten RAM **1806** geändert hat. Schritt **2323** zeigt an, dass der soeben beschriebene räumlich abgesetzte RAM **1803** für den Zugriff bereit ist.

[0132] Gleichzeitig mit Schritt **2321** wird in Schritt **2312** ein leerer Puffer in das Ende der Linked-List in der RAM-Bank **1803** geschrieben. Der Zeiger zu dem Puffer, der in Schritt **2321** geschrieben wurde, wird in das Link-Feld des leeren Puffers geschrieben, der in Schritt **2312** geschrieben wurde. Der Prozess schreitet dann zu Schritt **2306** voran, der die Vollendung der Schreibanforderung darstellt und anzeigt, dass die RAM-Bänke **1803** für die nächste Zugriffsanforderung bereit sind.

Beschreibung von Fig. 24

[0133] Es wurden Speichermanagementeinrichtungen für die Verarbeitung von Linked-List-Dateien beschrieben. Gemäß einer weiteren möglichen Ausführungsform der Erfindung können die offenbarten Speichermanagementeinrichtungen auch mit Hochgeschwindigkeits-RAM-Bankspeichern **1803** und einem räumlich abgesetzten Speicher **1806** arbeiten, um Dateien zu verarbeiten, die nicht vom Linked-List-Typ sind. Dies wird für die Prozessschritte der Fig. 24 und Fig. 25 beschrieben.

[0134] Das Folgende beschreibt die Prozessschritte von Fig. 24, in der ein Lesevorgang von Fig. 18 beschrieben ist, wobei die RAM-Bänke **1803** und der räumlich abgesetzte RAM **1806** für den Betrieb in einer solchen Weise konfiguriert sind, dass die RAM-Bänke **1803**, die eine hohe Geschwindigkeit, aber eine geringe Kapazität aufweisen, Informationen speichern, die vom Zugangsflussregler **1801** empfangen wurden. Der räumlich abgesetzte RAM **1806** wird als ein Überlauf zum Speichern von Informationen für große Dateien verwendet. Der Prozess beginnt in Schritt **2401** und schreitet zu Schritt **2402** voran, wo der Zugangsflussregler **1801** eine Leseanforderung an den Bus **1802** sendet, mit der Informationen angefordert werden, die in einer RAM-Bank **1803** gespeichert sind.

[0135] Schritt **2403** stellt fest, ob die Größe der abzurufenden Datei die momentane Speicherkapazität von RAM **1803** übersteigt. Wenn die Schwelle noch nicht überschritten wurde, so schreitet der Prozess zu Schritt **2404** voran, wo der RAM-Bank **1803** die angeforderte Datei liest, um sie zurück zum Zugangsflussregler **1801** zu übertragen. Der Prozess schreitet dann zu Schritt **2405** voran, der die angeforderten Informationen, die aus der RAM-Bank **1803** gelesen wurden, über den Zustandscontroller **1804** und über den Bus **1802** zum Zugangsflussregler **1801** zurücksendet. Die Zugangsflussregler **1801** empfängt die Informationen und leitet sie an den Prozessor **1814** weiter, um sie zum Steuern der Funktion zu verwenden, die mit den angeforderten Informationen verbunden sind. Der Prozess schreitet dann zu Schritt **2406** voran, der den Zugangsflussregler **1801** dar-

über informiert, dass die RAM-Bank **1803** für den Empfang einer weiteren Zugriffsanforderung bereit ist.

[0136] Wenn das Element **2403** feststellte, dass die Größe der zu lesenden angeforderten Datei die Schwelle überschreitet, so schreitet der Prozess zu Schritt **2410** voran, der den Abschnitt der Datei liest, der sich möglicherweise in der RAM-Bank **1803** befindet. Der Prozess schreitet zu Schritt **2411** voran, wo die gelesenen Informationen zum Zugangsflussregler **1801** zurückgesandt werden. Der Prozess schreitet zu Schritt **2412** voran, der eine Leseanforderung von dem räumlich abgesetzten RAM **1806** zum Hintergrundzugriffsbus **1810** sendet, der die Anforderung über den Hintergrundzugriffsmultiplexer **1808** zu dem angeforderten räumlich abgesetzten RAM **1806** weiterleitet.

[0137] Der Prozess schreitet dann zu Schritt **2413** voran, der die angeforderten Informationen von dem räumlich abgesetzten RAM **1806** abruft. Als nächstes sendet Schritt **2415** die in Schritt **2413** abgerufenen Informationen von dem räumlich abgesetzten RAM **1806** zur RAM-Bank **1803**, wo sie gespeichert werden. Der Prozess schreitet dann zu den Schritten **2414** und **2406** voran, die beide dem Zugangsflussregler **1801** anzeigen, dass die RAM-Bänke **1803** für den Empfang einer weiteren Zugriffsanforderung bereit sind.

Beschreibung von Fig. 25

[0138] Fig. 25 beschreibt, wie einer Datei, die in den RAM-Bänken **1803** vorhanden ist, eine Schreibanforderung hinzugefügt wird. Die RAM-Bank **1803** und der räumlich abgesetzte RAM **1806** wirken bei der Speicherung von Massendaten, die von dem Zugriffssteuerungsregler **1801** empfangen wurden, in einer ähnlichen Weise zusammen, wie es für den Lesebetrieb von Fig. 24 beschrieben wurde. Die RAM-Bank **1803** speichert eine existierende ausgewählte Menge an Daten für eine Datei, wobei der übrige Teil von großvolumigen Dateien überläuft und in den räumlich abgesetzten RAM **1806** geschrieben wird.

[0139] Der Prozess beginnt in Schritt **2501** und schreitet zu Element **2502** voran, das die Schreibanforderung, die vom Zugriffssteuerungsregler **1801** empfangen wurde, analysiert und feststellt, ob die Größe der zu schreibenden angeforderten Datei die Kapazität dessen überschreitet, was in der RAM-Bank **1803** gespeichert werden könnte. Wenn die Schwelle nicht überschritten wird, so veranlasst das Element **2502** den Schritt **2504**, zusätzliche Daten in die zugehörige Datei zu schreiben, die sich bereits in der ausgewählten RAM-Bank **1803** befindet. Der Prozess schreitet dann zu Element **2505** voran, der eine Bestätigung zum Zugriffssteuerungsregler **1801** zurücksendet, dass die angeforderte Datei in die RAM-Bank **1803** geschrieben wurde und dass ein Schreibvorgang zu dem räumlich abgesetzten RAM **1803** nicht erforderlich ist.

[0140] Wenn das Element **2502** feststellt, dass die Schwelle überschritten ist, so schreitet der Prozess zu Schritt **2511** voran, der veranlasst, dass der bereits gespeicherte Teil der Datei, die in der RAM-Bank **1803** gespeichert ist, gelesen wird. Der Prozess schreitet dann zu Schritt **2522** voran, der die Operationen initiiert, die erforderlich sind, damit dieser Teil der Datei, der in Schritt **2511** abgerufen wurde, in eine räumlich abgesetzte RAM-Bank **1806** geschrieben wird. Schritt **2523** zeigt an, dass der räumlich abgesetzte RAM **1806** wieder für einen Zugriff bereit ist. Schritt **2512** veranlasst, dass ein Schreibzeiger in die RAM-Bank **1803** geschrieben wird, damit sie in die Lage versetzt wird, eine Verbindung zu der Adresse des räumlich abgesetzten RAM **1806** herzustellen, die den übrigen Teil der Datei enthält, deren anderer Abschnitt in den RAM **1803** geschrieben wurde. Der Prozess schreitet dann zu Schritt **2506** voran, der anzeigt, dass die RAM-Bank **1803** für den Empfang einer weiteren Zugriffsanforderung bereit ist.

Schlusswort

[0141] Netzverkehrsgestaltung erfordert flexible Pufferung. Flüsse mit höherer Priorität haben Vorrang vor anderen Flüssen mit geringerer Priorität. Beispielsweise sind die Anforderungen eines Echtzeitverkehrs höher als die Anforderungen von gewöhnlichem Datenverkehr, wie beispielsweise Dateitransfer. Aber die unmittelbaren Charakteristika eines Verkehrs sind zufällig. Das nächste ankommende Paket könnte über jeden beliebigen Fluss geroutet werden. Wenden wir uns beispielsweise Fig. 3 zu. Ein Strom aus Paketen, der auf Link 1 beim Knoten A ankommt, könnte entweder über Link 2 oder Link 3 abgehenden. Der Verkehr, der auf Link 1 ankommt, könnte Tausende aufeinanderfolgender Pakete tragen, die auf Link 2 gerichtet sind, und dann ein einzelnes Paket tragen, das auf Link 3 gerichtet ist, gefolgt von Tausenden aufeinanderfolgender Pakete, die wieder auf Link 2 gerichtet sind. Zusätzlich zu der Anforderung, dass kein Paket abgeworfen wird und dass eine Wiederherstellung nach Totalabstürzen, wie beispielsweise einer Kabeldurchtrennung, unterstützt wird, muss es variabel verzögerte, geplante abgehende Verkehrsgestaltungsalgorithmen unterstützen. Somit wird sowohl der ankommende als auch der abgehende Verkehr in seinem Aufbau verkompliziert und erfordert flexible Pufferung.

[0142] Die effizienteste Anordnung zum Ausgleichen der Pufferhandhabung für ankommende Ströme und abgehende Ströme ist die Hardware-Linked-List-Maschine. Aber die gegenwärtigen Implementierungen der Hardware-Linked-List-Maschine sind entweder groß und langsam oder klein und schnell. Die verbesserte Linked-List-Maschine der vorliegenden Erfindung ist den zum Stand der Technik gehörenden List-Maschinen überlegen, weil sie preiswerte Gigabytes an Pufferspeicher bietet und mit dem maximalen Durchsatz der verfügbaren Halbleiterspeicher arbeitet. So ist die verbesserte Linked-List-Maschine der vorliegenden Erfindung so groß wie die derzeitigen Maschinen nach dem Stand der Technik und ist wesentlich schneller. Die zusätzliche Geschwindigkeit bietet sich für Verkehrsgestaltungsanwendungen an, weil die verbesserte Linked-List-Maschine bei gleichem Umfang an Hardware-Ressourcen, nämlich FPGAs und SDRAM, Faserleitungen mit höherer Kapazität (d. h. OC-768) unterstützt. Die verbesserte Linked-List-Maschine ist in der Lage, die Leitungen mit höherer Kapazität zu unterstützen, weil ihre Puffer eine Tiefe bis in den Gigabyte-Bereich hinein aufweisen.

[0143] Diese Erfindung unterstützt den Burstmodus von handelsüblichen dynamischen Speicherbausteinen. Diese Unterstützung zeigt sich in zwei Facetten dieser Erfindung. Erstens, wenn Zugriffe auf handelsübliche dynamische RAM kontinuierlich und aufeinanderfolgend sind, belegt diese Erfindung die Datenpins des handelsüblichen dynamischen Speichers entweder während des Lesens der abgerufenen Informationen oder während des Schreibens von zu speichernden Informationen, während der nächste folgende Zugriff initiiert wird. Zweitens führt diese Erfindung ein Caching der zu schreibenden Puffer in handelsüblichen dynamischen RAM durch und puffert Lesedaten von handelsüblichen dynamischen RAM kontextabhängig.

[0144] Durch gleichzeitige Verarbeitung der Datenpins für die momentanen Zugriffsanforderungen und Initiieren der nächsten Zugriffsanforderung verwendet diese Erfindung das Datenbus-Zwischenspeicherungsmerkmal des Datenbusses auf dem handelsüblichen dynamischen RAM. Diese Merkmal ermöglicht es, dass eine ganze Zeile des dynamischen Speichers mit der Geschwindigkeit des Datenbusses – in der Regel 166 Megahertz oder mehr – ausgelesen oder eingeschrieben werden kann. Diese Zeile kann 128 Bits lang sein, aber die Anzahl der verfügbaren Datenbuspins auf dem dynamischen Speicher beträgt vielleicht nur acht. So nehmen die Speicher intern eine Zwischenspeicherung der Zeile des zu lesenden Speichers vor und senden diese Informationen mit jeweils acht Bits auf einmal zu den Pins oder nehmen eine Zischenspeicherung der ankommenden Daten mit jeweils acht Bits auf einmal vor, bevor sie die gesamte Zeile in ihre interne Speicheranordnung schreiben. Durch ein derartiges Überlappen von Zugriffen hält diese Erfindung Daten in einer solchen Weise auf dem Datenbus, dass kontinuierliche Lesevorgänge von den handelsüblichen dynamischen Speicherbausteinen mit einer Rate stattfinden, die mit den RAM-Bänken der Erfindung kompatibel ist, wie beispielsweise RAM-Bank **1803** von Fig. 18. Dies ist von großem Nutzen, wenn auf eine Liste, die viele Puffer lang ist, nacheinander zugegriffen wird. In einem solchen Fall müssen Informationen aus dem handelsüblichen dynamischen Speicher abfließen. Um während einer solchen langen Reihe von Zugriffen die Leistung des handelsüblichen dynamischen Speichers aufrecht zu erhalten, muss die stationäre Leistung der Schnittstelle zu dem handelsüblichen dynamischen Speicher auf dem Niveau aller anderen Speicher gehalten werden.

[0145] Das Caching der zu schreibenden Puffer und der Puffer, die gelesen wurden, gestattet eine effiziente Nutzung der handelsüblichen dynamischen Speicherbausteine. Es ist überaus effizient, auf einen handelsüblichen dynamischen Speicher zeilenweise zuzugreifen. Aber Zeilen sind in der Regel groß, in der Größenordnung von 128 Bits oder mehr. Ein einzelner Puffer, der gespeichert oder abgerufenen werden soll, besteht aber vielleicht nur aus 16 Bits. Wenn die Puffer also einem Caching unterzogen werden, bis genug Informationen aufgelaufen sind, um eine Zeile voll zu bekommen, so kann ein Schreibzugriff auf den handelsüblichen dynamischen Speicher eine volle Zeile übertragen, was überaus effizient ist. Wenn die volle Zeile bei einem Lesevorgang einem Caching unterzogen wird, so braucht gleichermaßen nur ein einziger Zugriff per Zeile auf den handelsüblichen dynamischen Speicher zu erfolgen, was überaus effizient ist.

[0146] Somit führt ein Caching beim Schreiben und Lesen zu einer wesentlichen Leistungssteigerung, da weniger Zugriffe auf den handelsüblichen dynamischen Speicher erforderlich sind, und ein Verringern der Anzahl von Zugriffen mindert die Konkurrenz um den handelsüblichen dynamischen Speicher.

[0147] Diese Erfindung unterstützt viele gleichzeitige Linked-Lists. Jede Interaktion mit einer beliebigen Linked-List der vielen gleichzeitigen Linked-Lists ist vollkommen unabhängig von den übrigen der vielen gleichzeitigen Linked-Lists.

[0148] Die folgenden Ansprüche kennzeichnen RAM-Bänke **1803** als Hochgeschwindigkeitsspeicher und den räumlich abgesetzten Speicher **1806** als Massenspeicher.

[0149] Die obige Beschreibung offenbart mögliche Ausführungsbeispiele dieser Erfindung. Es wird davon ausgegangen, dass der Fachmann alternative Ausführungsformen konstruieren kann und wird, die die geistigen Schutzrechte dieser Erfindung, wie sie in den folgenden Ansprüchen wörtlich oder durch die Doktrin der Äquivalente dargelegt sind, verletzen.

Patentansprüche

1. Verfahren zum Betreiben eines Speichermanagementsystems, das dafür ausgelegt ist, Linked-List-Datendateien zu verarbeiten; das System umfaßt mehrere Hochgeschwindigkeitsspeicher mit niedriger Speicherkapazität und einen Massenspeicher mit niedrigerer Geschwindigkeit und hoher Speicherkapazität, die Hochgeschwindigkeitsspeicher weisen eine erste Datenrate auf, der Massenspeicher weist eine zweite Datenrate unter der ersten Datenrate auf, das System umfaßt weiterhin einen Zugangsflußregler zum Erzeugen von Anforderungen für das Lesen und Schreiben von Linked-Lists durch die Speicher, das Verfahren umfaßt die folgenden Schritte:
Einleiten des Schreibens einer Linked-List in den Hochgeschwindigkeitsspeichern durch Senden einer Schreibanforderung von dem Zugangsflußregler an die Hochgeschwindigkeitsspeicher;
Schreiben eines Kopfpuffers und eines Endpuffers und mindestens eines Zwischenpuffers der Linked-List in die Hochgeschwindigkeitsspeicher und Übertragen des mindestens einen Zwischenpuffers von den Hochgeschwindigkeitsspeichern zu dem Massenspeicher, während der Kopfpuffer und der Endpuffer in den Hochgeschwindigkeitsspeichern verbleiben.

2. Verfahren zum Betreiben eines Speichermanagementsystems, das dafür ausgelegt ist, Linked-List-Datendateien zu verarbeiten; das System umfaßt mehrere Hochgeschwindigkeitsspeicher mit niedriger Speicherkapazität und einen Massenspeicher mit niedrigerer Geschwindigkeit und hoher Speicherkapazität, die Hochgeschwindigkeitsspeicher weisen eine erste Datenrate auf, der Massenspeicher weist eine zweite Datenrate unter der ersten Datenrate auf, das System umfaßt weiterhin einen Zugangsflußregler zum Erzeugen von Anforderungen für das Lesen und Schreiben von Linked-Lists durch die Speicher, das Verfahren umfaßt die folgenden Schritte:
Senden einer Leseanforderung für eine spezifizierte Linked-List von dem Zugangsflußregler zu den Hochgeschwindigkeitsspeichern die Puffer der spezifizierten Linked-List enthalten;
Lesen des Kopfpuffers der spezifizierten Linked-List;
Übertragen des mindestens einen Zwischenpuffers der spezifizierten Linked-List von dem Massenspeicher zu einem der Hochgeschwindigkeitsspeicher;
Bezeichnen des zu dem einen Hochgeschwindigkeitsspeicher übertragenen Zwischenpuffers als den Ersatzkopfpuffer der spezifizierten Linked-List;
Auslesen des Zwischenpuffers der spezifizierten Linked-List aus dem einen Hochgeschwindigkeitsspeicher und
Senden der ausgelesenen Puffer der spezifizierten Linked-List zu dem Zugangsflußregler.

3. Verfahren nach den Ansprüchen 1 oder 2, weiterhin mit den folgenden Schritten:
Betreiben des Systems, um Linked-Lists für mehrere Anforderungen von dem Zugangsflußregler gleichzeitig zu verarbeiten;
Betreiben des Systems, um Puffer einer Linked-List zu verarbeiten, die in verschiedenen einzelnen der Hochgeschwindigkeitsspeicher gespeichert sind;
Betreiben des Systems, um einen Endpuffer als den zu einer neuen Linked-List geschriebenen ersten Puffer zu schreiben; und
Lesen des Kopfpuffers einer Linked-List zuerst.

4. Verfahren nach den Ansprüchen 1 oder 2, wobei das System weiterhin mehrere Zustandscontroller (**1804**) umfaßt, die jeweils einem entsprechenden der Hochgeschwindigkeitsspeicher zugeordnet sind, wobei das System weiterhin einen Anforderungsbus (**1802**) umfaßt, der den Zugangsflußregler mit den Zustandscontrollern verbindet, wobei der Schritt des Sendens einer Leseanforderung die folgenden Schritte beinhaltet:
Betreiben des Zugangsflußreglers, um einen freien Hochgeschwindigkeitsspeicher auszuwählen, der die Leseanforderung empfangen soll;
Senden der Leseanforderung von dem Zugangsflußregler über den Anforderungsbus zu dem Zustandscontroller, der dem ausgewählten Hochgeschwindigkeitsspeicher zugeordnet ist;
Betreiben des Zustandscontrollers, um das gegenwärtige Belegungsniveau des ausgewählten Hochgeschwindigkeitsspeichers zu bestimmen;
Senden der Anforderung an den Hochgeschwindigkeitsspeicher, wenn das gegenwärtige Belegungsniveau ein vorbestimmtes Niveau nicht übersteigt; und

Anfordern einer Verbindung zu dem Massenspeicher, wenn das gegenwärtige Belegungsniveau des ausgewählten Hochgeschwindigkeitsspeichers das vorbestimmte Niveau übersteigt;
wobei das System weiterhin einen Hintergrundzugangsmultiplexer und einen die Zustandscontroller mit dem Multiplexer verbindenden Zugangsbus enthält, wobei das System weiterhin einen Bus enthält, der den Multiplexer mit dem Massenspeicher verbindet, wobei das Verfahren die Schritte des Betreibens des Multiplexers beinhaltet zum:
Empfangen einer Anforderung von den Zustandscontrollern für eine Verbindung zu dem Massenspeicher;
Bestimmen, welchem der mehreren anfordernden Zustandscontrollern Zugang zu dem Massenspeicher gewährt werden soll;
Verbinden des einen anfordernden Zustandscontrollers mit dem Massenspeicher;
Steuern des Betriebs des Massenspeichers bei der Übertragung von Daten von dem einen Hochgeschwindigkeitsspeicher zu dem Massenspeicher und Übertragen von Puffern der Linked-List von dem Zustandscontroller zu dem Multiplexer über den Zugangsbus.

5. Verfahren nach den Ansprüchen 1 oder 2, wobei der Schritt des Übertragens der Puffer aus dem Massenspeicher die folgenden Schritte beinhaltet:
Herausübertragen von Zwischenpuffern einer Linked-List aus dem Massenspeicher zu den Hochgeschwindigkeitsspeichern in einem Burstmodus mit einer Datenrate, die der Datenrate der Hochgeschwindigkeitsspeicher im wesentlichen gleich ist;
Speichern der ausgelesenen Puffer in den Hochgeschwindigkeitsspeichern;
danach Auslesen von Puffern der Linked-List von den Hochgeschwindigkeitsspeichern zur Übertragung zu dem Zugangsflußregler;
Schreiben von Puffern zu einer existierenden Linked-List durch Übertragen des existierenden Endes der existierenden Linked-List von den Hochgeschwindigkeitsspeichern zu dem Massenspeicher und
Schreiben eines neuen Puffers als einen neuen Endpuffer der existierenden Linked-List in die Hochgeschwindigkeitsspeicher.

6. Verfahren nach Anspruch 4, wobei der Schritt des Betreibens des Zustandscontrollers die weiteren Schritte beinhaltet:
gleichzeitiges Empfangen von Puffern von mehreren Linked-Lists;
Trennen der Puffer von mehreren Linked-Lists, die zu dem Zugangsflußregler gerichtet sind;
Ausweiten mehrerer von dem Zugangsflußregler empfangener Zugänge zu den Hochgeschwindigkeitsspeichern;
Reagieren auf jede empfangene Anforderung von dem Zugangsflußregler, um das gegenwärtige Belegungsniveau des Hochgeschwindigkeitsspeichers zu bestimmen, der dem Zustandscontroller zugeordnet ist;
Ausweiten des Zugangs auf den assoziierten Hochgeschwindigkeitsspeicher, wenn das gegenwärtige Belegungsniveau nicht überschritten ist;
Signalisieren dem Zugangsflußregler, die Anforderung zu puffern, wenn das gegenwärtige Belegungsniveau überschritten ist;
Steuern der Übertragung eines Puffers von dem Hochgeschwindigkeitsspeicher in einem Burstmodus zu dem Massenspeicher;
Steuern der Übertragung eines Puffers von dem Massenspeicher zu dem Hochgeschwindigkeitsspeicher;
Bestimmen, ob der Massenspeicher frei ist, wenn eine Übertragung angefordert wird;
Ausweiten der Puffer auf den Massenspeicher, falls frei; und
Puffern der Übertragung, wenn der Massenspeicher belegt ist.

7. Verfahren nach Anspruch 4, wobei der Schritt des Betreibens des Multiplexers die weiteren Schritte beinhaltet:
Bestimmen, welchem von mehreren bietenden Hochgeschwindigkeitsspeichern Zugang zu dem Massenspeicher gewährt werden soll;
Puffern der Anforderungen von anderen Puffern in dem einen Hochgeschwindigkeitsspeicher;
Bestimmen der Identität zu dem Hochgeschwindigkeitsspeicher, zu dem ein Puffer von dem Massenspeicher übertragen werden soll; und
Steuern der Übertragung des Puffers in einem Burstmodus von dem Massenspeicher zu dem identifizierten Hochgeschwindigkeitsspeicher.

8. Verfahren nach den Ansprüchen 1 oder 2, das die weiteren Schritte umfaßt:
Erzeugen eines Signals, das für jeden Hochgeschwindigkeitsspeicher eindeutig ist und den belegten/freien Zustand jedes Hochgeschwindigkeitsspeichers anzeigt;
Ausweiten jedes erzeugten Signals auf den Zugangsflußregler;

Betreiben des Zugangsflußreglers, um Anforderungen für das Schreiben oder Lesen von Linked-Lists durch die Hochgeschwindigkeitsspeicher zu empfangen;
Betreiben des Zugangsflußreglers als Reaktion auf den Empfang einer Anforderung zum Lesen von belegten/freien Signalen, erzeugt von den Hochgeschwindigkeitsspeichern;
Betreiben des Zugangsflußreglers als Reaktion auf das Lesen, um einen freien der Hochgeschwindigkeitsspeicher zu identifizieren; und
Betreiben des Zugangsflußreglers zum Ausweiten einer Anforderung für das Lesen oder Schreiben einer Datendatei auf den freien Hochgeschwindigkeitsspeicher.

9. Speichermanagementsystem (**1800**), das dafür ausgelegt ist, Linked-List-Datendateien zu verarbeiten; wobei das System folgendes umfaßt:
mehrere Hochgeschwindigkeitsspeicher mit niedriger Speicherkapazität (**1803-1**, **1803-2**, **1803-3**, **1803-4**, **1803-5**, **1803-6**, **1803-7**, **1803-8**) und einen Massenspeicher mit niedrigerer Geschwindigkeit und hoher Speicherkapazität (**1806-1**, **1806-2**, **1806-3**), wobei die Hochgeschwindigkeitsspeicher eine erste Datenrate und der Massenspeicher eine zweite Datenrate unter der ersten Datenrate aufweisen;
einen Zugangsflußregler (**1801**) zum Erzeugen von Anforderungen für das Lesen und Schreiben von Linked-Lists durch die Speicher (**1803**, **1806**);
eine Vorrichtung (**1819**) zum Einleiten des Schreibens einer Linked-List in den Hochgeschwindigkeitsspeichern durch Senden einer Schreib-Lese-Anforderung an einen freien der Hochgeschwindigkeitsspeicher (**1803-1**, **1803-2**, **1803-3**, **1803-4**, **1803-5**, **1803-6**, **1803-7**, **1803-8**);
eine Vorrichtung (**1819**) zum Schreiben eines Kopfpuffers und eines Endpuffers und mindestens eines Zwischenpuffers der Linked-List in die Hochgeschwindigkeitsspeicher;
eine Vorrichtung (**1804**, **1808**, **1810**) zum Übertragen des mindestens einen Zwischenpuffers der Linked-List von den Hochgeschwindigkeitsspeichern zu dem Massenspeicher, während der Kopfpuffer und der Endpuffer der Linked-List in den Hochgeschwindigkeitsspeichern verbleibt;
eine Vorrichtung (**1802**) für das nachfolgende Senden einer Anforderung für das Lesen der Linked-List aus dem Zugangsflußregler zu den Hochgeschwindigkeitsspeichern;
eine Vorrichtung (**1804**) zum Lesen des Kopfpuffers der Linked-List in einem der Hochgeschwindigkeitsspeicher;
eine Vorrichtung (**1808**, **1810**, **1804**) zum Übertragen des mindestens einen Zwischenpuffers der Linked-List von dem Massenspeicher zu den Hochgeschwindigkeitsspeichern;
eine Vorrichtung (**1810**) zum Bezeichnen des übertragenen Puffers in den Hochgeschwindigkeitsspeichern als einen neuen Kopfpuffer;
eine Vorrichtung (**1804**) zum nachfolgenden Auslesen des Kopfpuffers und des Endpuffers sowie der Zwischenpuffer aus den Hochgeschwindigkeitsspeichern und
eine Vorrichtung (**1802**) zum Übertragen der ausgelesenen Puffer der Linked-List zu dem Zugangsflußregler.

10. Speichermanagementsystem nach Anspruch 9, das weiterhin folgendes umfaßt:
eine Vorrichtung (**1804**), die die Hochgeschwindigkeitsspeicher (**1803**) enthält, zum Erzeugen eines Signals, das für jeden Hochgeschwindigkeitsspeicher eindeutig ist und den gegenwärtigen belegten/freien Zustand jedes Hochgeschwindigkeitsspeichers anzeigt;
eine Vorrichtung (**1802**) zum Ausweiten der Signale auf den Zugangsflußregler;
eine Vorrichtung (**1814**), die den Zugangsflußregler (**1801**) enthält, zum Empfangen einer Anforderung für das Schreiben oder Lesen von Linked-Lists durch die Hochgeschwindigkeitsspeicher;
eine Vorrichtung (**1821**), die den Zugangsflußregler (**1801**) enthält, zum Lesen der belegt/frei-Signale als Reaktion auf den Empfang der Anforderung;
eine Vorrichtung (**1821**), die den Zugangsflußregler (**1801**) enthält, zum Bestimmen des aktuellen belegten/freien Zustands jedes der Hochgeschwindigkeitsspeicher als Reaktion auf das Lesen und eine Vorrichtung (**1821**, **1804**), die den Zugangsflußregler (**1801**) enthält, zum Gewähren einer Anforderung für das Lesen oder Schreiben einer Linked-List durch den Hochgeschwindigkeitsspeicher als Reaktion auf eine Bestimmung, daß einer der Speicher gegenwärtig frei ist.

Es folgen 14 Blatt Zeichnungen

Anhängende Zeichnungen

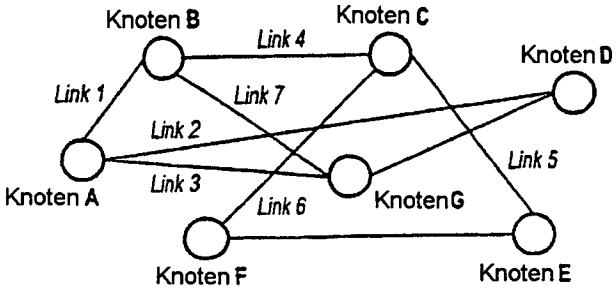


FIG. 1

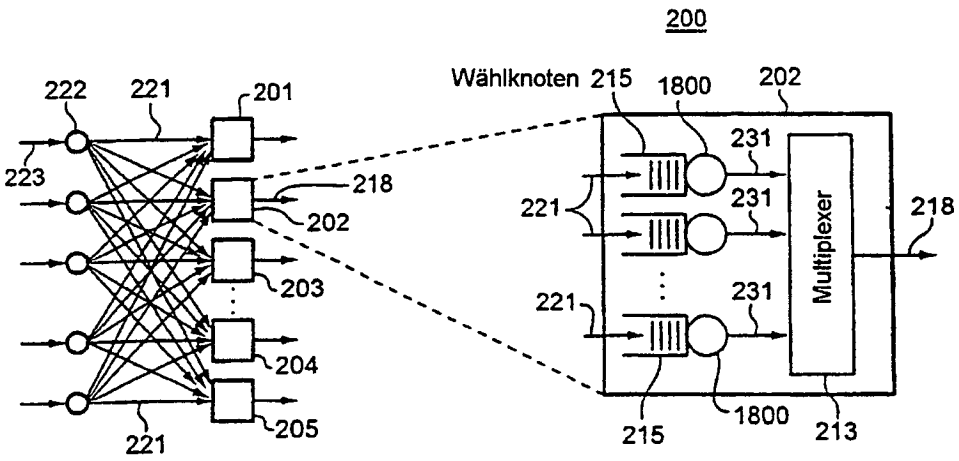


FIG. 2

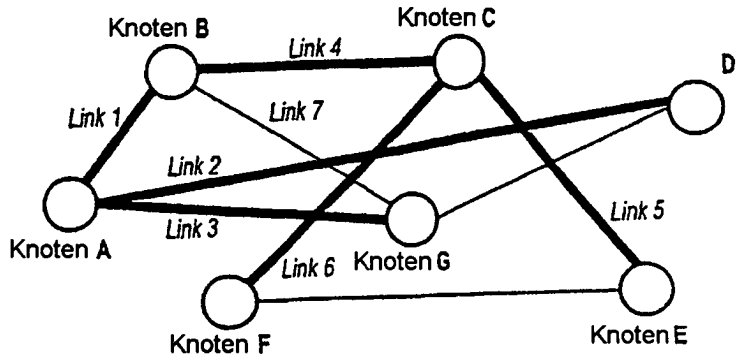


FIG. 3

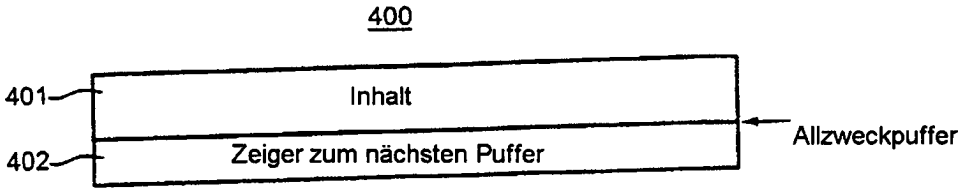


FIG. 4

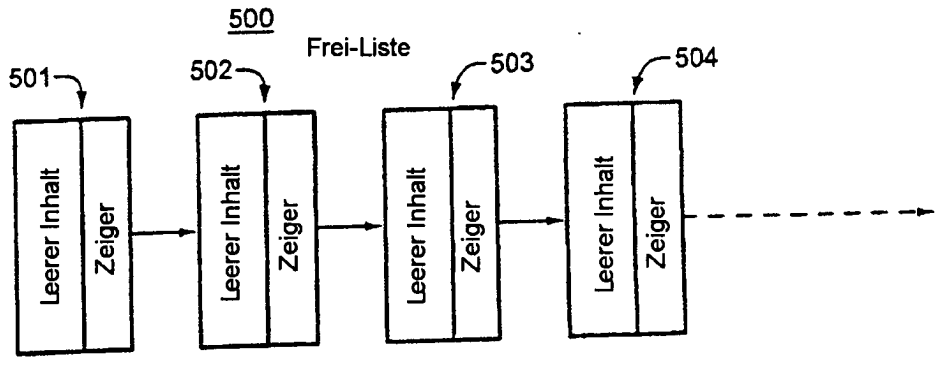


FIG. 5

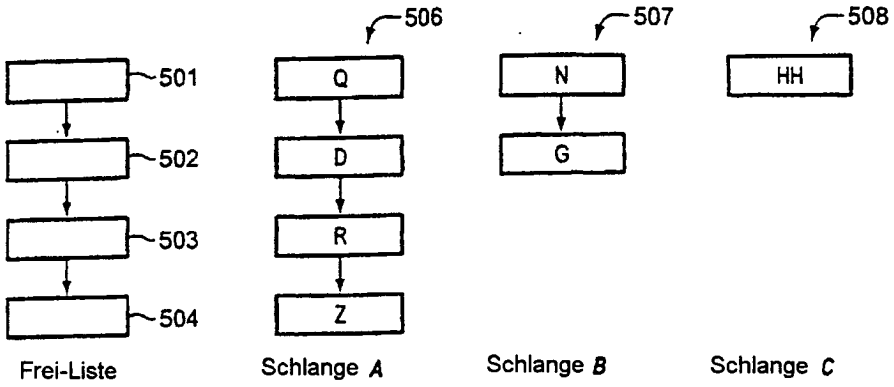


FIG. 6

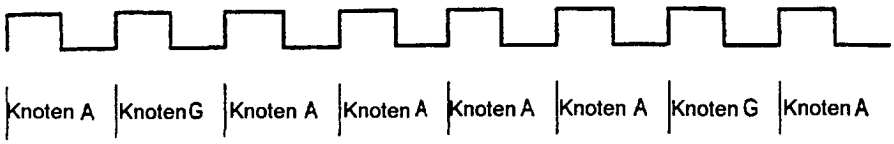


FIG. 7

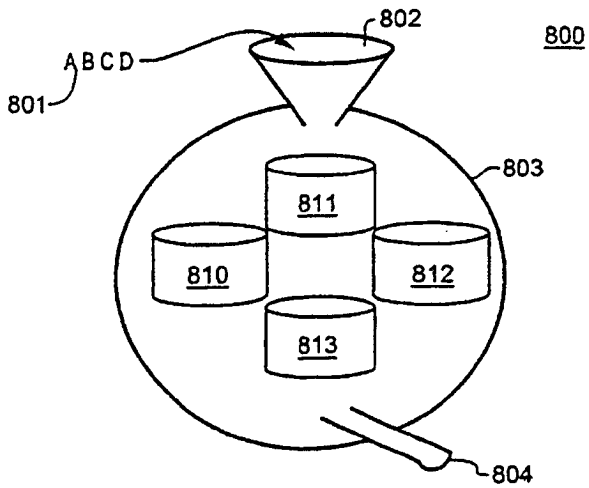


FIG. 8

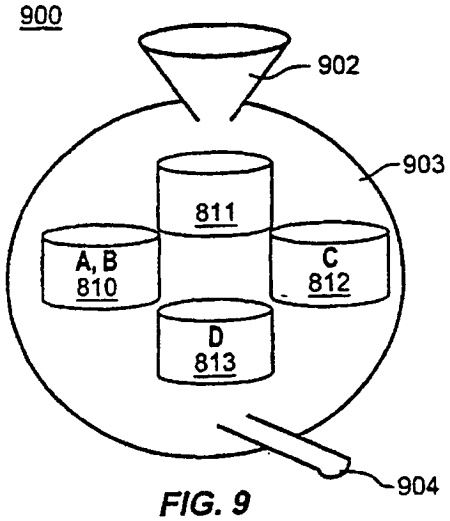


FIG. 9

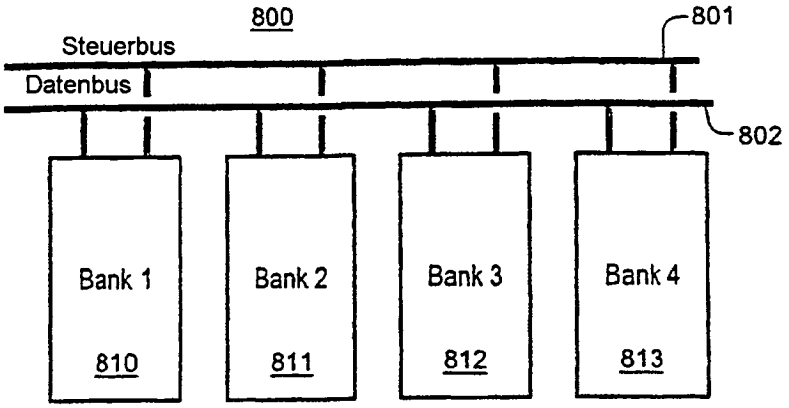


FIG. 10
Stand der Technik

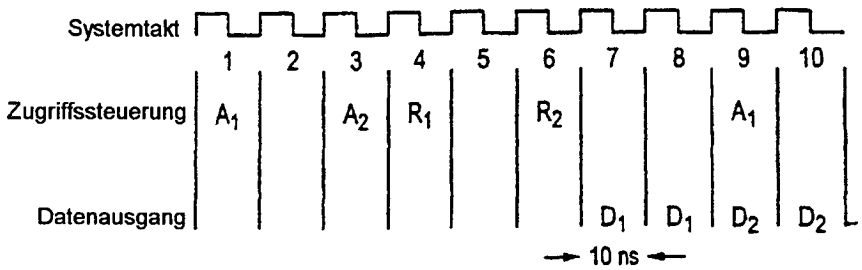


FIG. 11

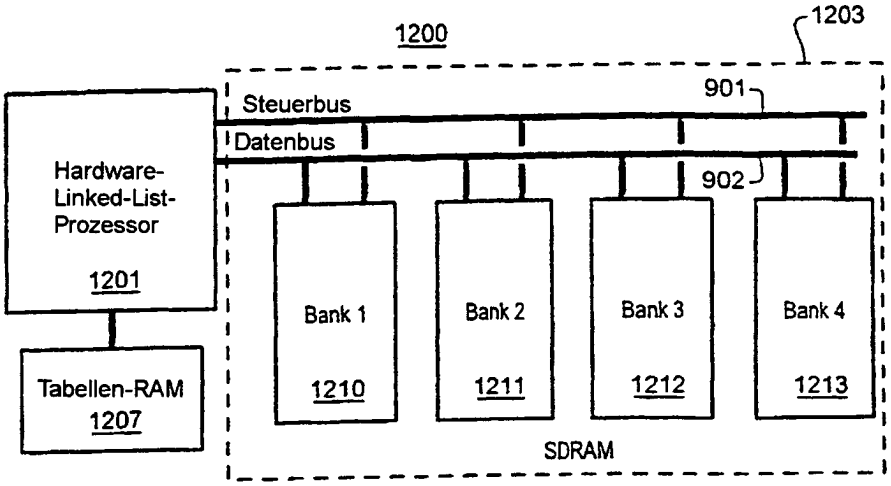


FIG. 12
STAND DER TECHNIK

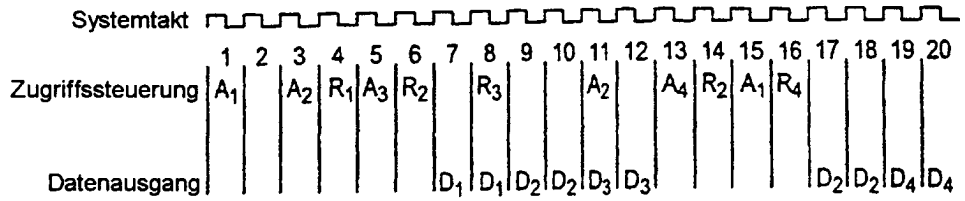


FIG. 13

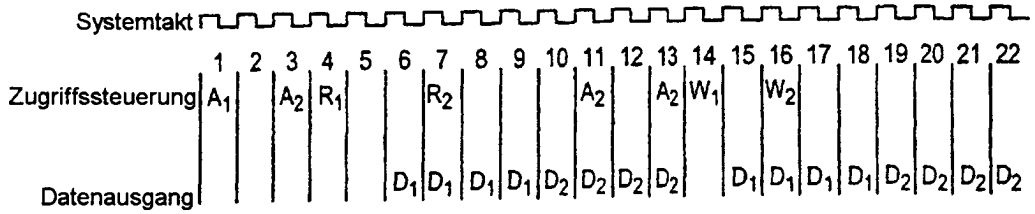


FIG. 14

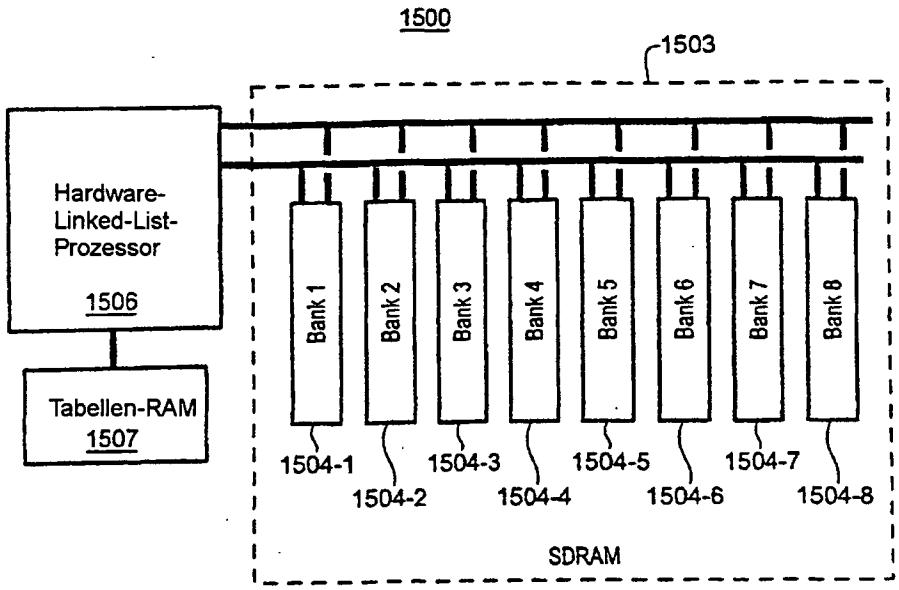

1500
1503
Hardware-Linked-List-Prozessor
1506
Tabellen-RAM
1507
Bank 1
Bank 2
Bank 3
Bank 4
Bank 5
Bank 6
Bank 7
Bank 8
1504-1
1504-2
1504-3
1504-4
1504-5
1504-6
1504-7
1504-8
SDRAM

FIG. 15

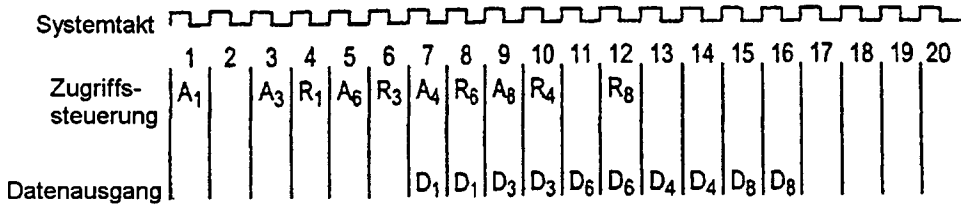

Systemtakt
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Zugriffs-steuerung
A1 A3 R1 A6 R3 A4 R6 A8 R4 R8
Datenausgang
D1 D1 D3 D3 D6 D6 D4 D4 D8 D8

FIG. 16

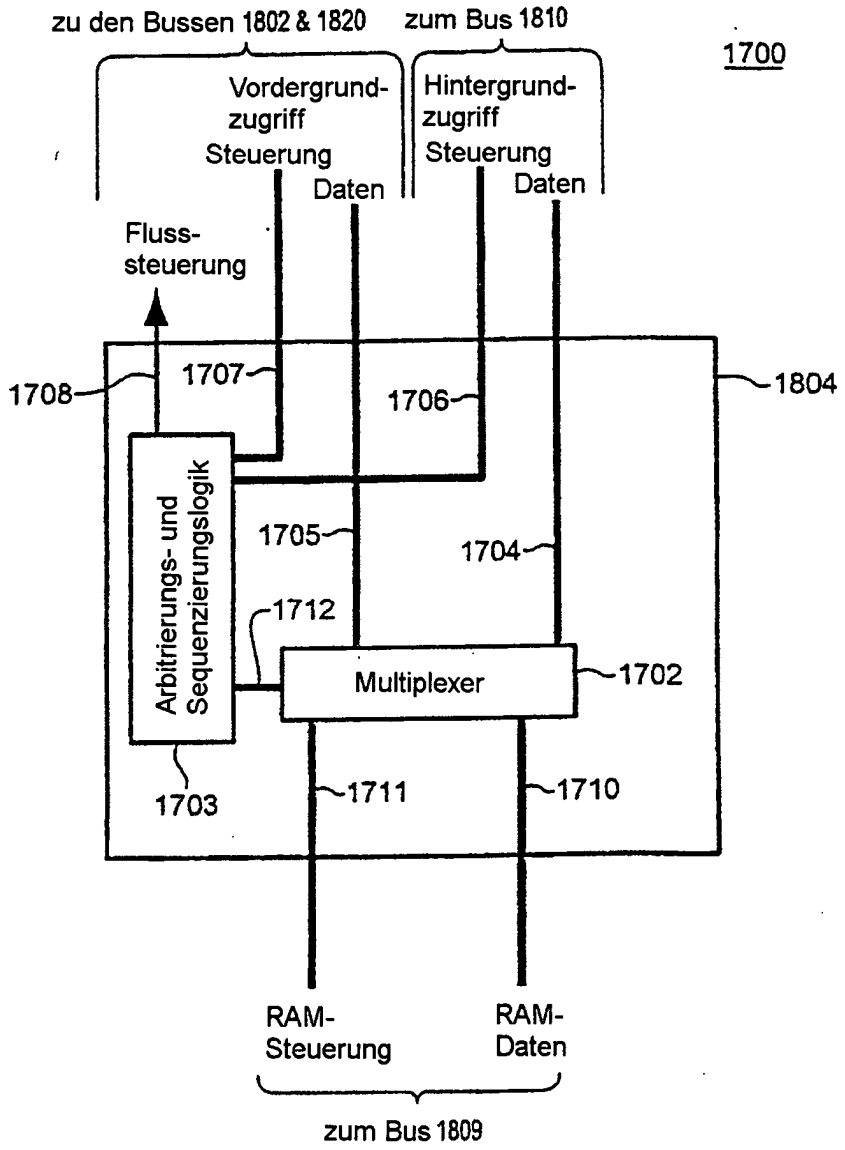

zu den Bussen 1802 & 1820
zum Bus 1810
1700
Vordergrund-zugriff Steuerung
Daten
Hintergrund-zugriff Steuerung
Daten
Fluss-steuerung
1708
1707
1706
1804
Arbitrierungs- und Sequenzierungslogik
1705
1704
1712
Multiplexer
1702
1703
1711
1710
RAM-Steuerung
RAM-Daten
zum Bus 1809

FIG. 17

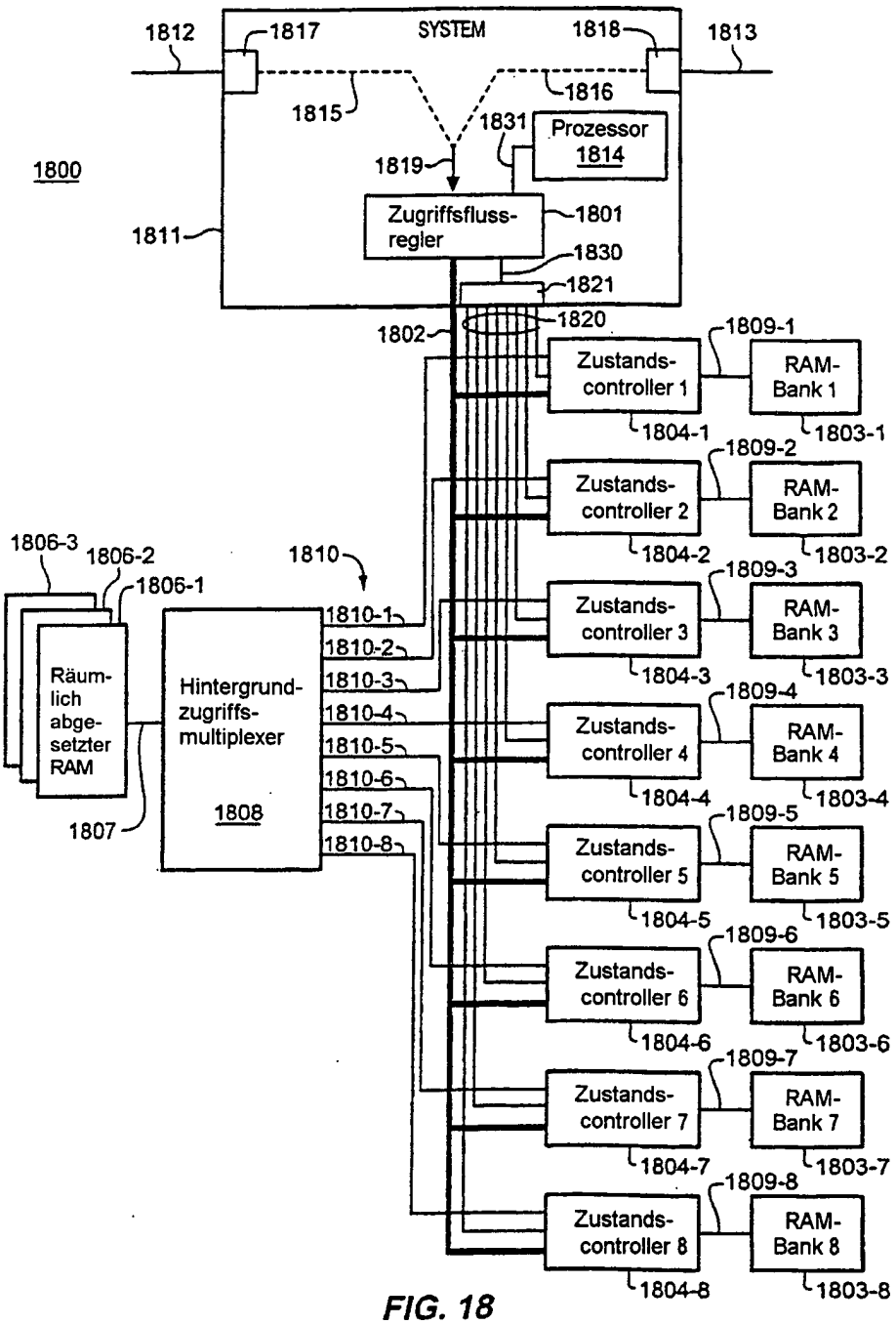

1800
1812
1817
SYSTEM
1818
1813
1815
1816
1831
Prozessor 1814
1819
Zugriffsfluss-regler
1801
1811
1830
1821
1820
1802
1809-1
Zustands-controller 1
RAM-Bank 1
1804-1
1803-1
1809-2
Zustands-controller 2
RAM-Bank 2
1804-2
1803-2
1806-3
1806-2
1806-1
1810
1809-3
Zustands-controller 3
RAM-Bank 3
1804-3
1803-3
Räumlich abgesetzter RAM
Hintergrund-zugriffs-multiplexer
1808
1807
1810-1
1810-2
1810-3
1810-4
1810-5
1810-6
1810-7
1810-8
1809-4
Zustands-controller 4
RAM-Bank 4
1804-4
1803-4
1809-5
Zustands-controller 5
RAM-Bank 5
1804-5
1803-5
1809-6
Zustands-controller 6
RAM-Bank 6
1804-6
1803-6
1809-7
Zustands-controller 7
RAM-Bank 7
1804-7
1803-7
1809-8
Zustands-controller 8
RAM-Bank 8
1804-8
1803-8

FIG. 18

FIG. 19

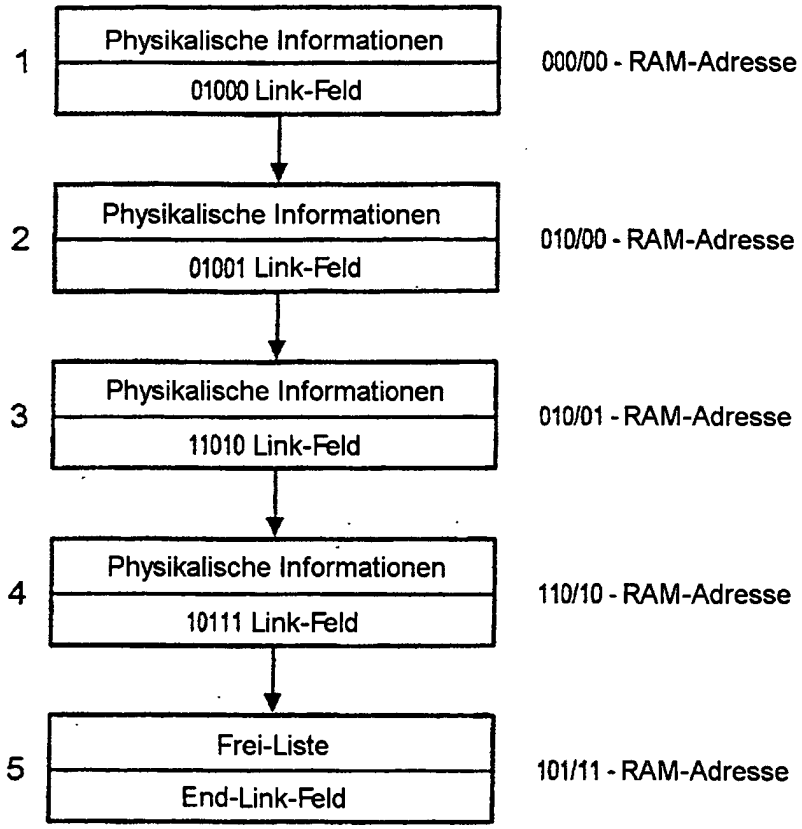

1
Physikalische Informationen
01000 Link-Feld
000/00 - RAM-Adresse
2
Physikalische Informationen
01001 Link-Feld
010/00 - RAM-Adresse
3
Physikalische Informationen
11010 Link-Feld
010/01 - RAM-Adresse
4
Physikalische Informationen
10111 Link-Feld
110/10 - RAM-Adresse
5
Frei-Liste
End-Link-Feld
101/11 - RAM-Adresse

FIG. 20

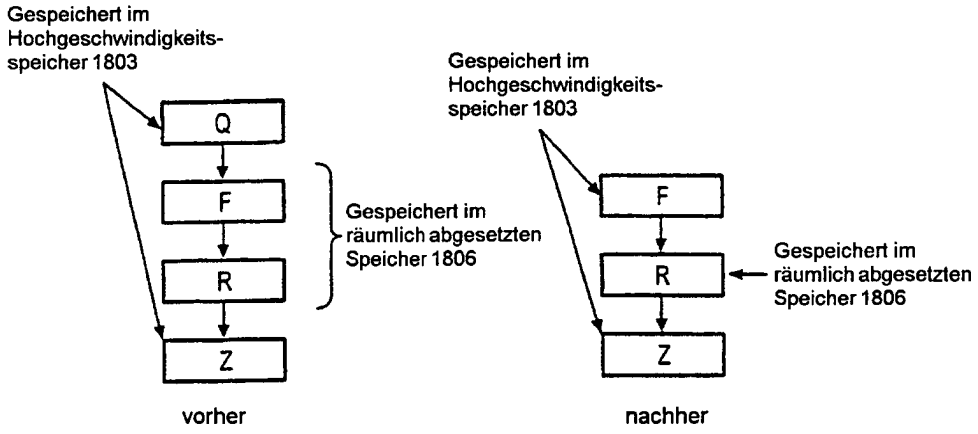

Gespeichert im Hochgeschwindigkeits-speicher 1803
Q
F
R
Z
Gespeichert im räumlich abgesetzten Speicher 1806
vorher
Gespeichert im Hochgeschwindigkeits-speicher 1803
F
R
Z
Gespeichert im räumlich abgesetzten Speicher 1806
nachher

FIG. 21

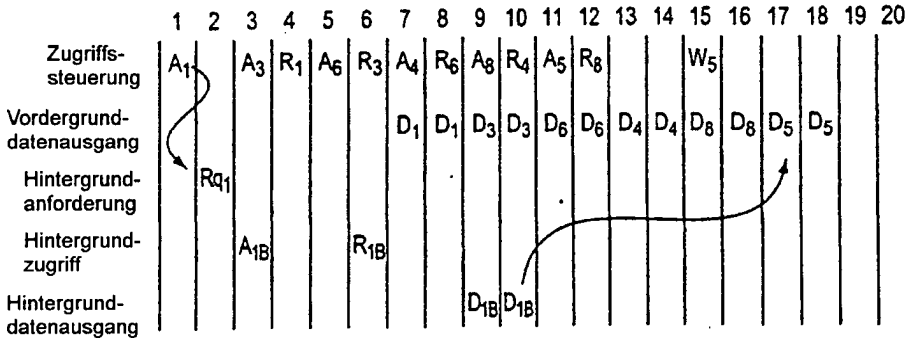

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Zugriffs-steuerung
Vordergrund-datenausgang
Hintergrund-anforderung
Hintergrund-zugriff
Hintergrund-datenausgang
A1 A3 R1 A6 R3 A4 R6 A8 R4 A5 R8 W5
D1 D1 D3 D3 D6 D6 D4 D4 D8 D8 D5 D5
Rq1
A1B R1B
D1B D1B

FIG. 22

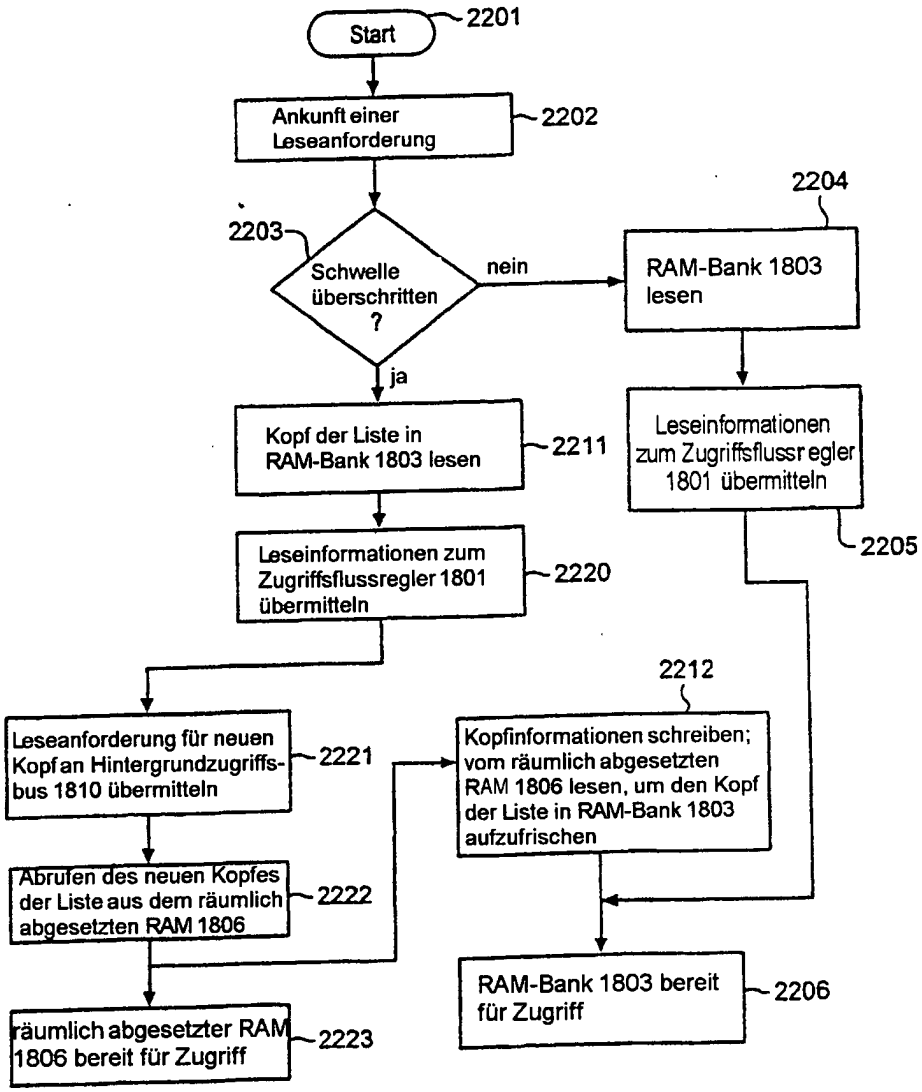

Start
2201
Ankunft einer Leseanforderung
2202
2203
Schwelle überschritten ?
nein
ja
2204
RAM-Bank 1803 lesen
Leseinformationen zum Zugriffsflussr egler 1801 übermitteln
2205
Kopf der Liste in RAM-Bank 1803 lesen
2211
Leseinformationen zum Zugriffsflussregler 1801 übermitteln
2220
2212
Leseanforderung für neuen Kopf an Hintergrundzugriffs-bus 1810 übermitteln
2221
Kopfinformationen schreiben; vom räumlich abgesetzten RAM 1806 lesen, um den Kopf der Liste in RAM-Bank 1803 aufzufrischen
Abrufen des neuen Kopfes der Liste aus dem räumlich abgesetzten RAM 1806
2222
RAM-Bank 1803 bereit für Zugriff
2206
räumlich abgesetzter RAM 1806 bereit für Zugriff
2223

FIG. 23

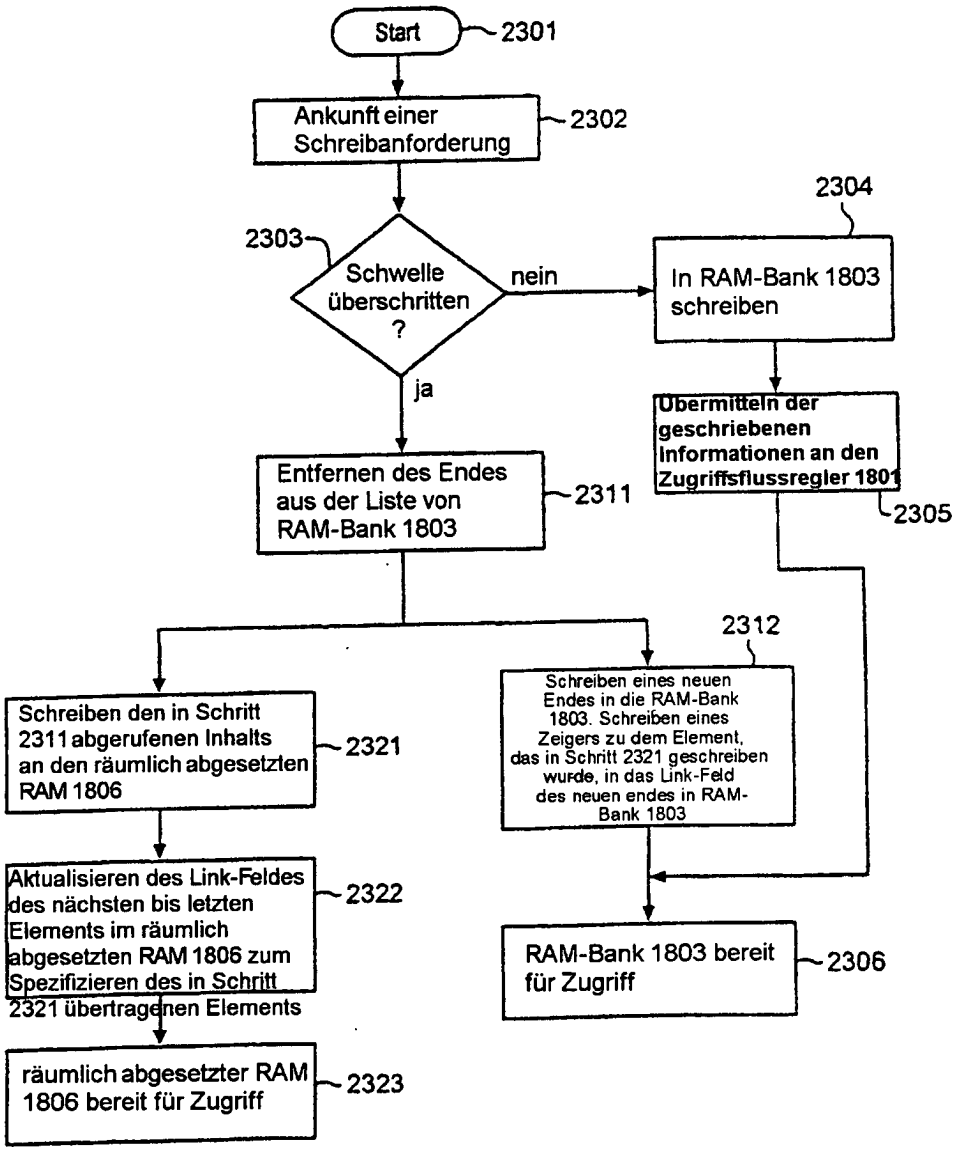

Start
2301
Ankunft einer Schreibanforderung
2302
2303
Schwelle überschritten ?
nein
ja
2304
In RAM-Bank 1803 schreiben
Übermitteln der geschriebenen Informationen an den Zugriffsflussregler 1801
2305
Entfernen des Endes aus der Liste von RAM-Bank 1803
2311
2312
Schreiben eines neuen Endes in die RAM-Bank 1803. Schreiben eines Zeigers zu dem Element, das in Schritt 2321 geschreiben wurde, in das Link-Feld des neuen endes in RAM-Bank 1803
Schreiben den in Schritt 2311 abgerufenen Inhalts an den räumlich abgesetzten RAM 1806
2321
Aktualisieren des Link-Feldes des nächsten bis letzten Elements im räumlich abgesetzten RAM 1806 zum Spezifizieren des in Schritt 2321 übertragenen Elements
2322
räumlich abgesetzter RAM 1806 bereit für Zugriff
2323
RAM-Bank 1803 bereit für Zugriff
2306

FIG. 24

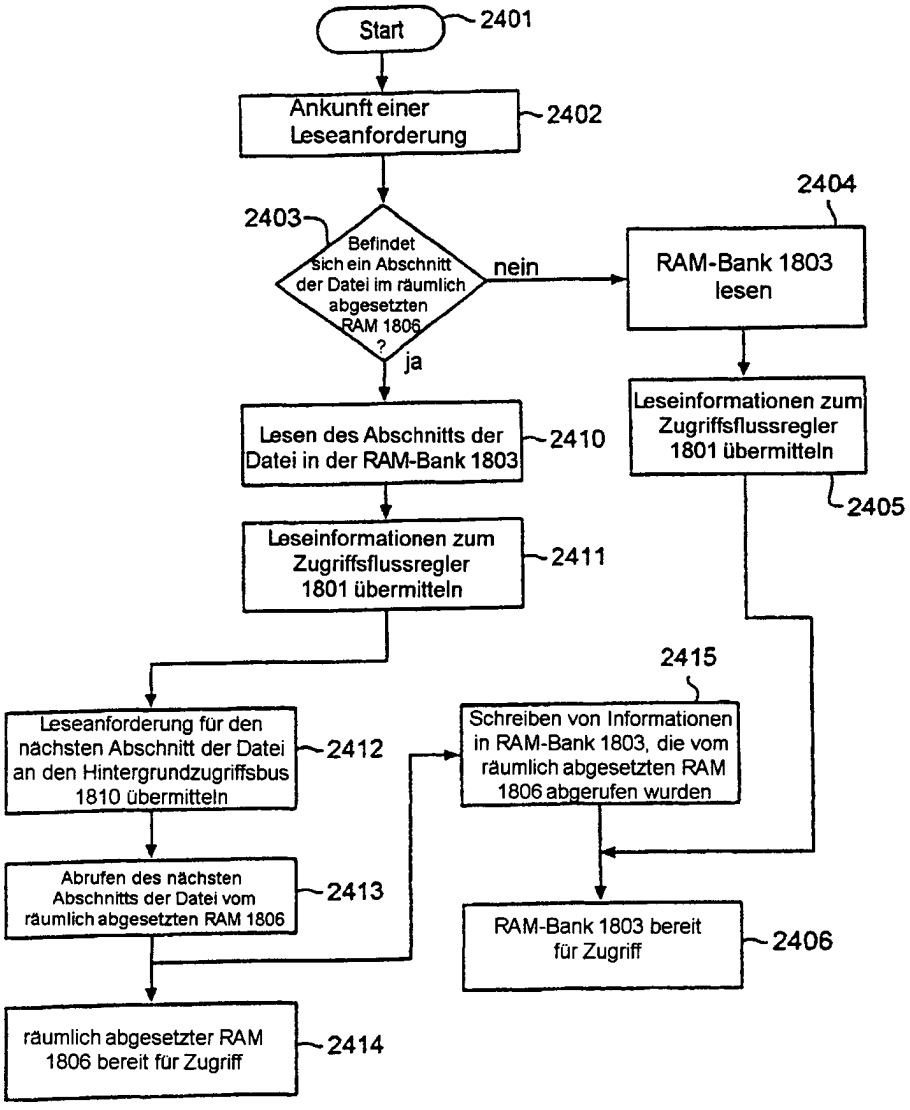
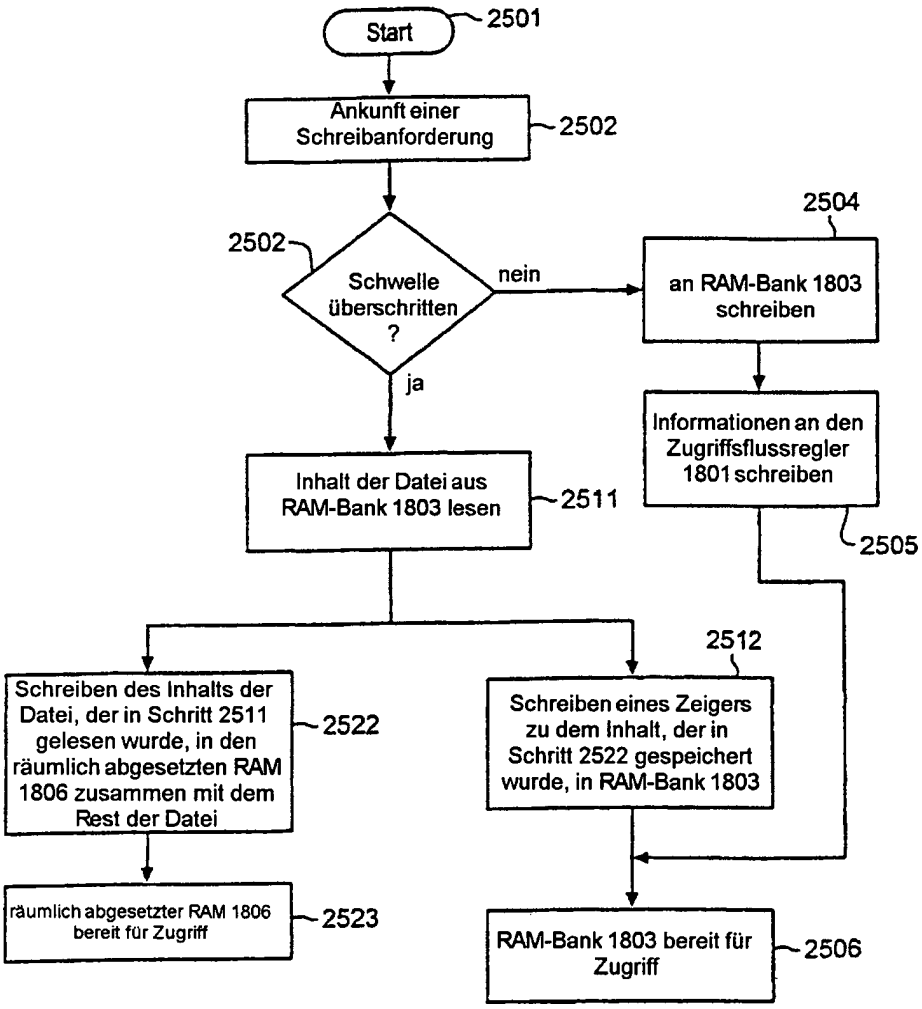

Start
2401
Ankunft einer Leseanforderung
2402
2403
Befindet sich ein Abschnitt der Datei im räumlich abgesetzten RAM 1806 ?
nein
ja
2404
RAM-Bank 1803 lesen
Leseinformationen zum Zugriffsflussregler 1801 übermitteln
2405
Lesen des Abschnitts der Datei in der RAM-Bank 1803
2410
Leseinformationen zum Zugriffsflussregler 1801 übermitteln
2411
2415
Leseanforderung für den nächsten Abschnitt der Datei an den Hintergrundzugriffsbus 1810 übermitteln
2412
Schreiben von Informationen in RAM-Bank 1803, die vom räumlich abgesetzten RAM 1806 abgerufen wurden
Abrufen des nächsten Abschnitts der Datei vom räumlich abgesetzten RAM 1806
2413
RAM-Bank 1803 bereit für Zugriff
2406
räumlich abgesetzter RAM 1806 bereit für Zugriff
2414

FIG. 25


Start
2501
Ankunft einer Schreibanforderung
2502
2502
Schwelle überschritten ?
nein
ja
2504
an RAM-Bank 1803 schreiben
Informationen an den Zugriffsflussregler 1801 schreiben
2505
Inhalt der Datei aus RAM-Bank 1803 lesen
2511
2512
Schreiben des Inhalts der Datei, der in Schritt 2511 gelesen wurde, in den räumlich abgesetzten RAM 1806 zusammen mit dem Rest der Datei
2522
Schreiben eines Zeigers zu dem Inhalt, der in Schritt 2522 gespeichert wurde, in RAM-Bank 1803
räumlich abgesetzter RAM 1806 bereit für Zugriff
2523
RAM-Bank 1803 bereit für Zugriff
2506