(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
12. Dezember 2002 (12.12.2002)

PCT

(10) Internationale Veröffentlichungsnummer
WO 02/099653 A1

(51) Internationale Patentklassifikation[7]: G06F 12/10

(21) Internationales Aktenzeichen: PCT/EP02/06146

(22) Internationales Anmeldedatum:
4. Juni 2002 (04.06.2002)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
101 27 186.7 5. Juni 2001 (05.06.2001) DE

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **INFINEON TECHNOLOGIES AG** [DE/DE]; St.-Martin-Str. 53, 81669 München (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **GAMMEL, Berndt** [DE/DE]; Dr. Brenner Str. 16, 85570 Markt-Schwaben (DE). **KNIFFLER, Oliver** [DE/DE]; Weddigenstr. 1, 81737 München (DE). **LEDWA, Ralph** [DE/DE]; Burk 15, 87616 Marktoberdorf (DE). **SEDLAK, Holger** [DE/DE]; Neumuenster 10a, 85658 Egmating (DE).

(74) Anwälte: **SCHOPPE, Fritz** usw.; Schoppe, Zimmermann, Stöckeler & Zinkler, Postfach 71 08 67, 81458 München (DE).

(81) Bestimmungsstaat *(national)*: US.

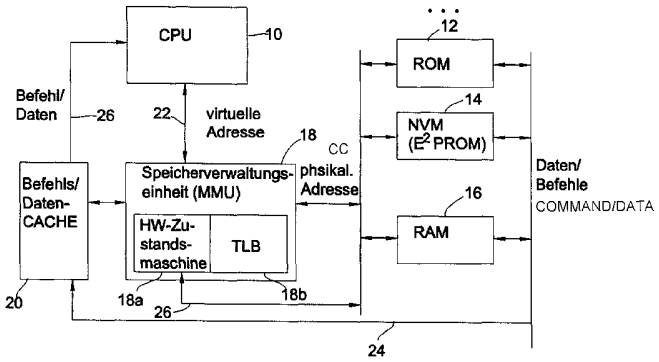
(84) Bestimmungsstaaten *(regional)*: europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Veröffentlicht:
— *mit internationalem Recherchenbericht*

[Fortsetzung auf der nächsten Seite]

(54) Title: COMPUTER SYSTEM HAVING VIRTUAL ADDRESSING AND METHOD FOR DETERMINING A PHYSICAL ADDRESS FROM A VIRTUAL ADDRESS

(54) Bezeichnung: RECHNERSYSTEM MIT VIRTUELLER ADRESSIERUNG UND VERFAHREN ZUM ERMITTELN EINER PHYSIKALISCHEN ADRESSE AUS EINER VIRTUELLEN ADRESSE



18a...MACHINE FOR DETERMINING THE STATE OF THE HARDWARE
18...MEMORY MANAGEMENT UNIT (MMU)
26...COMMAND/DATA
22...VIRTUAL ADRESS
CC...PHYSICAL ADDRESS

(57) Abstract: The invention relates to a computer system having virtual addressing, comprising a non-volatile memory (12, 14) for storing a representation specification, by which means a virtual address is clearly associated with a physical address. Said computer system also comprises a machine for determining the state of the hardware, which can access (26) the non-volatile memory (12, 14) and is embodied in such a way that it can determine the physical address corresponding to the virtual address, on the basis of the virtual address and the representation specification. Given that the representation specification is stored in the non-volatile memory (12, 14) and the machine for determining the state of the hardware (18a) can access said non-volatile memory (12, 14) by means of a bus (26), the computer system can be initialised as soon as it is in the virtual addressing mode in such a way that it does not operate in the physical addressing mode. The point of attack which usually represents the physical addressing mode is thus overcome.

[Fortsetzung auf der nächsten Seite]

WO 02/099653 A1

— *vor Ablauf der für Änderungen der Ansprüche geltenden Frist; Veröffentlichung wird wiederholt, falls Änderungen eintreffen*

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

(57) Zusammenfassung: Ein Rechnersystem mit virtueller Adressierung umfasst einen nicht-flüchtigen Speicher (12, 14) zum Speichern einer Abbildungsvorschrift, über die eine virtuelle Adresse einer physikalischen Adresse eindeutig zugeordnet ist. Das Rechnersystem umfasst ferner eine Hardware-Zustandsmaschine, die auf den nicht-flüchtigen Speicher (12, 14) zugreifen kann (26) und ausgebildet ist, um unter Verwendung der virtuellen Adresse und der Abbildungsvorschrift die der virtuellen Adresse zugeordnete physikalische Adresse zu ermitteln. Dadurch, dass die Abbildungsvorschrift im nicht-flüchtigen Speicher (12, 14) gespeichert ist und die Hardware-Zustandsmaschine (18a) über einen Bus (26) auf den nicht-flüchtigen Speicher (12, 14) zugreifen kann, kann das Rechnersystem bereits im virtuellen Adressierungsmodus initialisiert werden, so dass das Rechnersystem an keiner Stelle im physikalischen Adressierungsmodus betrieben wird. Der Angriffspunkt, den der physikalische Adressierungsmodus üblicherweise darstellt, ist somit überwunden.

Beschreibung

Rechnersystem mit virtueller Adressierung und Verfahren zum Ermitteln einer physikalischen Adresse aus einer virtuellen Adresse

Die vorliegenden Erfindung bezieht sich auf Rechnersysteme und insbesondere auf Rechnersysteme mit virtueller Adressierung.

Die virtuelle Adressierung ist seit längerem aus dem Bereich der Workstations bekannt. Unter Verwendung einer bestimmten Abbildungsvorschrift werden virtuelle Adressen auf physikalische Adressen abgebildet. Die physikalischen Adressen adressieren Speicherzellen eines physikalischen Speichers, wie z. B. des Arbeitsspeichers, einer Festplatte, eines Bandspeichers etc.. Virtuelle Adressen verweisen dagegen nicht direkt auf physikalische Speicherzellen, sondern lediglich indirekt, über die Abbildungsvorschrift. Der Vorteil dieser Adressierungsart besteht darin, daß sich der Programmierer einer Anwendung nicht um die in einem Rechnersystem vorhandenen diversen physikalischen Speicher kümmern muß. Der Programmierer hat einen virtuellen Adressraum zur Verfügung, den er für sein Programm je nach Bedarf nützen kann. Die Abbildung auf den physikalischen Adressraum, den ein spezielles Rechnersystem zur Verfügung stellt, wird getrennt vom Programmcode erstellt, so daß durch Bereitstellen verschiedener Abbildungsvorschriften ein mit virtuellen Adressen programmiertes Programm auf verschiedenen Rechnersystemen laufen kann.

In der Fachveröffentlichung „Protected Virtual Memory - 32-Bit-Power ohne angezogene Handbremse", Stephan Ondrusch, 10. GMD-SmartCard-Workshop, Darmstadt, 9. Feb. 2000, werden die Vorteile einer virtuellen Adressierung für Chipkarten beschrieben. In Chipkarten, auf denen verschiedene Anwendungen laufen sollen, die aus Sicherheitsgründen voneinander getrennt sein sollen, wird jedem Prozeß bzw. jeder Task ein ei-

gener virtueller Adressraum zur Verfügung gestellt, der beim Erstellen des Programmcodes für den Prozeß definiert und vom Betriebssystem zur Laufzeit eingerichtet wird. Unter der Kontrolle des Betriebssystems der Chipkarte werden Zugriffsrechte auf weitere Chipressourcen, z. B. gemeinsam genutzte Speicherbereiche, eingetragen und ständig per Hardware überwacht. Damit ist es möglich, die Programmcodes für zwei Prozesse auf einer Chipkarte unabhängig voneinander zu erstellen, indem zwei voneinander getrennte virtuelle Adressräume zur Verfügung gestellt werden. Durch die Abbildungsvorschriften für die beiden virtuellen Adressräume auf den einzigen vorhandenen physikalischen Adressraum kann sichergestellt werden, daß die beiden Prozesse nicht interferieren bzw. daß die beiden Prozesse kontrolliert einen physikalischen Speicherbereich gemeinsam nutzen, um beispielsweise bestimmte Routinen gemeinsam zu verwenden, so daß diese Routinen nur ein einziges Mal in dem Speicher der Chipkarte abgespeichert werden müssen, oder um einen sicheren Datenaustausch zwischen den Prozessen zu erreichen.

In einem Prozessor mit einem virtuellen Speichersystem läuft eine Applikation somit im sogenannten virtuellen Adressraum ab. Jede Adresse des virtuellen Speichers, an der sich von der Applikation les/schreibbare Daten oder ausführbarer Code befinden, ist auf eine Adresse im physikalischen Speicher abgebildet, an der diese Daten bzw. dieser Code tatsächlich gespeichert sind. Die virtuelle Adresse (VA) und die über die Abbildungsvorschrift zugeordnete physikalische Adresse (PA) müssen keinerlei Bezug zueinander haben. Der virtuelle Adressraum kann ferner wesentlich größer als der physikalische Adressraum sein.

Virtuelle Adressen, an denen sich keine les/schreibbaren Daten oder ausführbarer Code befinden, werden in der Regel nicht auf einen physikalischen Speicher abgebildet. Für die ausgeführte Applikation ist diese Abbildung völlig transparent.

Bei einer Organisation des Speichers in Pages bzw. Seiten ist der virtuelle Adressraum in gleich große, überlappungsfreie Speicherbereich unterteilt. Einer Seite im virtuellen Adressraum ist eine Seite im physikalischen Adressraum über die Abbildungsvorschrift zugeordnet, wobei die Seite im physikalischen Adressraum auch als Page Frame bezeichnet wird.

Der Nutzdatenspeicher eines Page Frames des physikalischen Adressraums ist genauso groß wie der einer Page des virtuellen Adressraums.

Die Zuordnung einer virtuellen Seite zu einer physikalischen Seite wird üblicherweise über die sogenannte Seitentabelle oder Page Table erreicht, welche Adresspaare von jeweiligen Startadressen der virtuellen Seiten und der zugeordneten physikalischen Seiten enthält.

Bei Workstations befindet sich ein Teil der Page Table in einem Cache, der auch als „Translation Look Aside Buffer (TLB) bezeichnet wird. Befindet sich das Startadresspaar für eine virtuelle Seite und die zugeordnete physikalische Seite in dem TLB, so erfolgt die Berechnung der Adressabbildung in den virtuellen Speicherbereich beschleunigt, da lediglich ein Zugriff auf eine Tabelle erforderlich ist, um die einer virtuellen Adresse zugeordnete physikalische Adresse zu erhalten.

Befindet sich das Startadresspaar, d. h. die virtuelle Adresse und die zugeordnete physikalische Adresse, nicht im TLB, so findet ein TLB-Miss (TLB-Fehlschlag) statt, was üblicherweise zu einem Trap an das Betriebssystem führt, welches das Adressen-Tupel in den TLB nachtragen muß.

Im Bereich der Workstations wird die Abbildungsvorschrift zwischen virtuellem Adressraum und physikalischem Adressraum, die beispielsweise als eine einzige Page Table implementiert

sein kann, im flüchtigen Arbeitsspeicher gehalten. Wenn eine Workstation hochgefahren wird, so startet sie zunächst im reellen Adressierungsmodus. Dies bedeutet, daß das Betriebssystem der Workstation die CPU der Workstation veranlasst, im reellen, d. h. physikalischen Adressierungsmodus, in dem flüchtigen Arbeitsspeicher der Workstation eine Page Table nach und nach aufzubauen. Erst wenn eine Page Table konstruiert ist, kann die Workstation in den virtuellen Adressierungsmodus umschalten. Frägt dann die CPU nach Daten an einer virtuellen Adresse, so wird im flüchtigen Arbeitsspeicher der CPU die dazu gehörige physikalische Adresse ermittelt, um die Daten von dem Speicher abrufen zu können. Übliche Workstations zeichnen sich also dadurch aus, daß sie in einem realen Adressierungsmodus hochfahren und dann, wenn die Abbildungsvorschrift von dem virtuellen Adressraum zu dem physikalischen Adressraum im flüchtigen Speicher aufgebaut ist, in den virtuellen Adressierungsmodus umschalten.

Nachteilig an diesem Konzept ist zum einen die Tatsache, daß ein relativ großer Arbeitsspeicherbereich benötigt wird, um eine Page Table zu speichern. Dieser Nachteil ist für Workstations nicht von größter Bedeutung, da sie sehr große Mengen an Arbeitsspeicher zur Verfügung haben. Für andere Anwendungen, wie z. B. für sicherheitsrelevante Rechnersysteme, wie sie beispielsweise in Chipkarten-ICs implementiert sind, sind die Speicherressourcen aufgrund des geringen zur Verfügung stehenden Platzes begrenzt. Eine Bereitstellung einer Menge an flüchtigem Arbeitsspeicher, um eine Page Table zu speichern, führt dazu, daß die auf der Chipkarte ausgeführten Applikationen unter Umständen zu wenig Arbeitsspeicher haben und damit Leistungseinbußen erfahren.

Ein weiterer Nachteil des bekannten Konzepts besteht darin, daß ein wesentlicher Verwaltungsaufwand nötig ist, um zunächst, beim Hochfahren des Rechnersystems, die Page Table aufzubauen, d. h. aus gespeicherten Informationen nach und nach die Adressenzuordnungen zu berechnen und abzuspeichern.

Neben der Tatsache, daß dafür Rechnerressourcen erforderlich sind, müssen auch entsprechende Programme auf einer Chipkarte bereitgestellt werden, um die für den virtuellen Adressierungsmodus erforderlichen Vorkehrungen zu treffen. Auch solche Programme benötigen Speicherplatz, der aus Platzgründen insbesondere bei Chipkarten oder anderen Sicherheits-ICs ein rares Gut ist.

Ein weiterer Nachteil des beschriebenen Konzepts des Initialisierens im physikalischen Adressierungsmodus und des darauf folgenden Umschaltens in den virtuellen Adressierungsmodus besteht darin, daß die Vorteile des virtuellen Adressierungsmodus, also des Trennens zweier voneinander zu trennenden Applikationen, nicht von Anfang an erhalten werden, sondern erst dann, wenn die CPU den realen Modus beendet hat und in den virtuellen Adressierungsmodus umgeschaltet hat. Ein physikalischer Adressierungsmodus bzw. ein „realer" Adressierungsmodus stellt insbesondere bei Sicherheits-Controllern eine Sicherheitsschwäche dar, die auch als „Single Point of Attack" bezeichnet wird. Gelingt es einem Angreifer, in das Betriebssystem des Chipkarten-IC einzudringen, während sich dasselbe im physikalischen Adressierungsmodus befindet, so kann der Angreifer ohne Barrieren auf die physikalischen Speicherzellen zugreifen und sensible Daten auslesen oder sogar manipulieren. Die Sicherheitsmerkmale des virtuellen Adressierungsmodus werden somit nicht von Anfang an erreicht, sondern erst dann, wenn das Rechnersystem in den virtuellen Adressierungsmodus umgeschaltet hat.

Die Aufgabe der vorliegenden Erfindung besteht darin, ein sicheres und einfaches Rechnersystem zu schaffen.

Diese Aufgabe wird durch ein Rechnersystem mit virtueller Adressierung nach Patentanspruch 1 und durch ein Verfahren zum Ermitteln einer physikalischen Adresse aus einer virtuellen Adresse nach Patentanspruch 17 gelöst.

Der vorliegenden Erfindung liegt die Erkenntnis zugrunde, dass die Sicherheit eines Rechnersystems mit virtueller Adressierung dadurch verbessert werden kann, daß das Rechnersystem keinen physikalischen Adressierungsmodus verwendet, sondern von vorneherein im virtuellen Adressierungsmodus arbeitet. Um dies zu erreichen, wird in einem nicht-flüchtigen Speicher des Rechnersystems eine Abbildungsvorschrift abgespeichert, mittels der aus einer virtuellen Adresse eine physikalische Adresse ermittelt werden kann. Die Abbildungsvorschrift wird im nicht-flüchtigen Speicher gespeichert, so daß dieselbe unmittelbar beim Hochfahren des Rechnersystems vorhanden ist und nicht erst, wie im Stand der Technik, in einem realen Modus erzeugt und im flüchtigen Speicher abgespeichert werden muß. Damit wird auch der komplette Verwaltungsaufwand und die dafür erforderlichen Programme, um eine Abbildungsvorschrift beispielsweise in Form einer Hash-Tabelle zu erzeugen, hinfällig, da die Abbildungsvorschrift bereits im nicht-flüchtigen Speicher fertig verfügbar ist.

Damit das Rechnersystem bereits im virtuellen Adressierungsmodus hochfahren kann, wird erfindungsgemäß neben dem nicht-flüchtigen Speicher, in dem die Abbildungsvorschrift gespeichert ist, eine Hardware-Zustandsmaschine vorgesehen, die auf den nicht-flüchtigen Speicher zugreifen kann und ausgebildet ist, um unter Verwendung der virtuellen Adresse und der Abbildungsvorschrift die der virtuellen Adresse zugeordnete physikalische Adresse zu ermitteln. Die Hardware-Zustandsmaschine führt, wie es bei Zustandsmaschinen üblich ist, einen fest vorgegebenen Algorithmus selbsttätig aus, wobei der Algorithmus, den die Hardware-Zustandsmaschine ausführt, einerseits Eingangsdaten von dem nicht-flüchtigen Speicher erhält und andererseits die virtuelle Adresse als Eingangsdaten, um als Ausgangsdaten die physikalische Adresse auszugeben.

Die einfachste Art und Weise, jedoch nicht die optimalste Art und Weise besteht darin, eine komplette Page Table im nicht-

flüchtigen Speicher zu speichern. In diesem Fall würde die Hardware-Zustandsmaschine aktiviert werden, wenn die CPU des Rechnersystems Daten bei einer virtuellen Adresse wünscht. Die Hardware-Zustandsmaschine wird in diesem Fall die virtuellen Adresse einlesen, auf den nicht-flüchtigen Speicher zugreifen, die physikalische Adresse, die der virtuellen Adresse zugeordnet ist, aus der Page Table extrahieren und dann ausgeben.

Aus Effizienz- und Speicherplatzgründen wird es jedoch bevorzugt, die Abbildungsvorschrift in Form eines hierarchischen Baums mit einem Wurzelknoten für eine Wurzelebene, zumindest einem Zwischenknoten für zumindest eine Zwischenebene und einem Endknoten für eine Endebene zu speichern, so daß die Hardware-Zustandsmaschine, gesteuert von der virtuellen Adresse und von in dem nicht-flüchtigen Speicher abgespeicherten Listen für die einzelnen Knoten einen sogenannten Page Table Walk („Seitentabellenspaziergang") durchführt, um nach Durchlaufen des Baumes die physikalische Adresse auszugeben, die der eingegebenen virtuellen Adresse entspricht. Bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung, bei dem der virtuellen Adressraum wesentlich größer als der physikalischen Adressraum ist, und somit Listen für Knoten relativ wenige benutzte Einträge und dagegen relativ viele Null-Einträge aufweisen, werden die Listen für Knoten des hierarchischen Baums komprimiert, um Speicherplatz im nicht-flüchtigen Speicher zu sparen.

Im Falle einer solchen hierarchisch organisierten Abbildungsvorschrift zwischen virtueller und physikalischer Adresse muß nicht die ganze Abbildungsvorschrift im nicht flüchtigen Speicher gespeichert werden, sondern wenigstens der Teil der Abbildungsvorschrift, durch den es möglich ist, einen Hochfahrvorgang des Systems im virtuellen Modus zu beginnen. Bei geeigneten Listeneinträgen im virtuellen Modus ist es dann möglich, bereits beim Wiedergewinnen der benötigten Daten aus dem physikalischen Speicher im flüchtigen Arbeitsspeicher den

restlichen Teil der Abbildungsvorschrift zu erzeugen und für weitere Adressübersetzungen von virtuell zu physikalisch zu verwenden. Die Hardware-Zustandsmachine kann daher nach dem Hochfahren des Systems im virtuellen Modus durchaus auch auf zur Laufzeit im flüchtigen Speicher programmierte Daten zugreifen.

Bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung wird ferner eine seitenweise Adressierung bevorzugt. Um eine Speicherfragmentierung zu vermeiden, werden in diesem Fall so viel als möglich kombinierte Listen in ein und derselben physikalischen Seite abgespeichert.

Ein Vorteil der vorliegenden Erfindung besteht darin, daß das Rechnersystem lediglich eine virtuellen Adressierung und keine physikalische Adressierung durchführen kann, und die Sicherheitsschwäche nicht mehr vorhanden ist.

Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß keine Programme bzw. kein Verwaltungsaufwand erforderlich ist, um in dem flüchtigen Speicher eine Page Table aufzubauen, da die Page Table bzw. der hierarchische Baum in dem nicht-flüchtigen Speicher persistent gespeichert ist.

Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß im Falle der Abbildungsvorschrift als hierarchische Baumstruktur durch Implementieren von Zugriffsrechten an dem Knoten des Baumes eine differenzierte Zugriffsrechtevergabe mit einer einstellbaren Granularität erreicht werden kann.

Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß durch Abspeichern der Abbildungsvorschrift im nichtflüchtigen Speicher, und zwar in einer Form, wie sie von der Hardware-Zustandsmaschine ohne Einschaltung der CPU verwendet werden kann, CPU-Ressourcen gespart werden. Selbstverständlich muß die Abbildungsvorschrift, beispielsweise die Page

Table oder die hierarchische Struktur von Knotenlisten erstellt werden. Dies kann jedoch außerhalb des Betriebs des Rechnersystems gewissermaßen „Off-Line" bzw. im Falle einer Chipkarte „Off-Card" durchgeführt werden. Das Erstellen der Abbildungsvorschrift und das Abspeichern der Abbildungsvorschrift im nicht-flüchtigen Speicher nimmt somit keine wertvolle On-Line-Ressource an Speicher oder CPU in Anspruch, sondern kann, wenn genügend Zeit ist, beispielsweise bei der Herstellung der Chipkarte oder im Falle einer dynamischen Modifikation der Abbildungsvorschrift dann vorgenommen werden, wenn das Rechnersystem gerade keine sicherheitsrelevante Anwendung durchführt. Erfindungsgemäß können daher Bearbeitungsschritte aus dem On-Line-Betrieb heraus verlagert werden, um wertvolle Online-Ressourcen, wie z. B. Rechnerleistung, Speicherplatz, Energieverbrauch etc., frei zu machen bzw. einzusparen.

Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend bezugnehmend auf die beiliegenden Zeichnungen detailliert erläutert. Es zeigen:

Fig. 1 ein erfindungsgemäßes Rechnersystem mit virtueller Adressierung;

Fig. 2 eine schematische Darstellung einer Abbildung eines virtuellen Adressraums in einen physikalischen Adressraum für ein Rechnersystem auf einer Chipkarte;

Fig. 3 eine Übersichtsdarstellung einer Adressübersetzung unter Verwendung einer Page Table;

Fig. 4a eine schematische Darstellung einer Adressübersetzung unter Verwendung einer Abbildungsvorschrift in Form einer hierarchischen Baumstruktur;

Fig. 4b eine Tabelle zur Darstellung der Knotenebenen und der durch einen Knoten adressierten Adressbereiche

gemäß einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung;

Fig. 5 ein Beispiel für eine Abbildungsvorschrift in Form einer hierarchischen Baumstruktur, bei der Zwischenknoten übersprungen werden können;

Fig. 6 eine Tabelle zur Darstellung von Knotengrößen auf verschiedenen Ebenen für das Beispiel von Fig. 4a;

Fig. 7 ein Beispiel für eine Abbildungsvorschrift in Form eines n-Baums mit gleichen Größen für zusätzliche Knoten einer Ebene;

Fig. 8 eine schematische Darstellung eines Kompressionsverfahrens für Knotenlisten, um das Verhältnis von benutzten Einträgen zu der Gesamtzahl der Einträge einer Liste zu verbessern;

Fig. 9 Kompressionsbeispiele gemäß dem Kompressionsverfahren von Fig. 8;

Fig. 10 eine komprimierte Darstellung des Baums von Fig. 7;

Fig. 11 eine Speicherplatz-optimierte Abspeicherung des Baums von Fig. 10; und

Fig. 12 eine Darstellung einer virtuellen Adresse, die modifiziert ist, um auf eine physikalische Adresse zu verweisen, an der eine Knotenliste abgespeichert ist.

Fig. 1 zeigt ein erfindungsgemäßes Rechnersystem mit einer CPU 10, einem nicht-flüchtigen Speicher in Form eines ROM 12 und eines E^2PROM 14, der in Fig. 1 auch als NVM (NVM = non volatile memory) bezeichnet wird. Das Rechnersystem kann ferner einen flüchtigen Speicher 16, der in Fig. 1 mit RAM be-

zeichnet ist, und gegebenenfalls weitere Peripheriekomponenten, wie z. B. einen Coprozessor, eine Eingabe/Ausgabe-Schnittstelle, einen Zufallszahlengenerator etc. umfassen.

Das Rechnersystem weist ferner eine Speicherverwaltungseinheit (MMU; MMU = Memory Management Unit) 18 sowie einen Cache 20 auf, wobei der Cache 20 in einen Befehls-Cache und einen Daten-Cache aufgeteilt sein kann, wenn das Rechnersystem gemäß der Harvard-Architektur aufgebaut ist. Die CPU ist über einen Bus 22 für virtuelle Adressen mit der Speicherverwaltungseinheit 18 verbunden. Die Speicherverwaltungseinheit 18 umfaßt eine Hardware-Zustandsmaschine 18a sowie vorzugsweise einen Translation-Look-Aside-Buffer (TLB) 18b, um aus einer von der CPU 10 über den Bus 22 gelieferten virtuellen Adresse eine physikalische Adresse zu ermitteln, um die einer virtuellen Adresse zugeordneten Daten von einem Speicher 12, 14, 16 zu adressieren und über einen Daten/Befehle-Bus 24 in den Cache 20 zu laden, aus dem die CPU 10 über einen weiteren Befehle/Daten-Bus 26 die einer virtuellen Adresse zugeordneten Daten erhält.

Aus Fig. 1 ist ferner zu sehen, daß die Hardware-Zustandsmaschine 18a der Speicherverwaltungseinheit 18 über einen Zugriffsbus 26 auf einen nicht-flüchtigen Speicher in Form des ROM 12 oder des NVM 14 zugreifen kann, um die Abbildungsvorschrift, die zur Berechnung der physikalischen Adresse aus der virtuellen Adresse erforderlich ist, aus dem nicht-flüchtigen Speicher zu laden. Die Hardware-Zustandsmaschine 18a kann somit über den Zugriffsbus 26 auf den nicht-flüchtigen Speicher zugreifen, um unter Verwendung einer virtuellen Adresse und der in dem nicht-flüchtigen Speicher gespeicherten Abbildungsvorschrift die der virtuellen Adresse zugeordnete physikalischen Adresse zu ermitteln.

Aus Fig. 1 ist ersichtlich, daß die CPU 10 vorzugsweise ausschließlich über den virtuellen Adressbus 22 adressiert, und zwar insbesondere die Hardware-Zustandsmaschine 18a adres-

siert, die erfindungsgemäß direkt auf den nicht-flüchtigen Speicher zugreifen kann, so daß das erfindungsgemäße Rechnersystem keinen physikalischen Adressierungsmodus benötigt, sondern im virtuellen Adressierungsmodus hochfahren kann, so daß es keinen Angriffspunkt in Form eines im physikalischen Adressierungsmodus laufenden Betriebssystems des erfindungsgemäßen Rechnersystems gibt.

Im nachfolgenden wird allgemein auf virtuelle Speichersysteme eingegangen. Ein Schutzmechanismus für Multitasking-Rechnersysteme besteht darin, daß separaten Prozessen oder Anwendungen voneinander getrennte virtuelle Adressräume zugeordnet werden, so daß ein Multitasking-Betrieb mit verschiedenen Anwendungen möglich ist, wobei diese Anwendungen jedoch aus Sicherheitsgründen vollständig voneinander getrennt sein müssen. Bei einem bevorzugten Ausführungsbeispiel des erfindungsgemäßen Rechnersystems, das für eine Chipkarte geeignet ist, wird ein virtueller Adressraum von 4 Gigabyte bereitgestellt, wie es in Fig. 2 zu sehen ist. Der Adressraum wird aus Gründen der besseren Effizienz in virtuellen Speicherseiten 30 aufgeteilt, wobei eine Speicherseite eine Größe von 64 Byte, also 512 Bits hat. Die Speicherseiten des virtuellen Adressraums können durch eine 32 Bit breite virtuelle Adresse adressiert werden.

Wesentlich am Konzept der virtuellen Adressierung ist, daß eine Anwendung, die auf der CPU läuft, nicht direkt auf den physikalischen Speicher zugreifen kann, sondern nur auf ihren eigenen virtuellen Adressraum. Unter Verwendung der Speicherverwaltungseinheit 18 von Fig. 1 wird der virtuelle Adressraum unter Verwendung einer Abbildungsvorschrift 32, die durch die in Fig. 2 gezeigten Pfeile symbolisiert ist, in physikalische Speicheradressen abgebildet. Bei dem in Fig. 2 gezeigten Ausführungsbeispiel ist der physikalische Adressraum auf 4 MB festgelegt, wobei jedoch darauf hingewiesen wird, daß eine übliche Chipkarte bei weitem nicht so viel Speicher hat. Typische Speichergrößen für das E^2PROM 14 von

Fig. 1 sind 64 kByte. Der RAM 16 von Fig. 1 hat typischerweise 8 kByte, während der ROM 12 von Fig. 1 typischerweise 168 Kbyte hat. Wenn ein virtueller Adressraum von 4 GB auf etwa 240 kByte Speicherzellen abgebildet wird, so ist zu sehen, daß zunächst eine große Menge virtueller Adressen nicht in physikalische Speicheradressen abgebildet werden muß, und daß zusätzlich auch eine große Anzahl von physikalischen Speicheradressen nicht auf tatsächlich vorhandene physikalische Speicherzellen zeigen. Andererseits erlaubt die Überdimensionierung des virtuellen Adressraums gegenüber dem physikalischen Adressraum und die Überdimensionierung des physikalischen Adressraums bezüglich der tatsächlich vorhandenen Speicherzellen einfache Modifikationen dahingehend, daß zusätzliche Speicher ohne weiteres nachträglich eingefügt werden können bzw. daß der physikalische Adressraum je nach Notwendigkeit erweitert werden kann.

Wie es später noch erläutert werden wird, wird die Abbildungsvorschrift 32 erfindungsgemäß nicht von der CPU im RAM 16 aufgebaut, sondern beispielsweise bei der Herstellung der Chipkarte entwickelt und in dem ROM 12 in Form einer entsprechend gestalteten ROM-Maske einprogrammiert, bevor das erfindungsgemäße Rechnersystem in Betrieb genommen wird. Im Betrieb benötigt das Rechnersystem daher abgesehen von der Zustandsmaschine keinerlei Ressourcen zum Erzeugen der Abbildungsvorschrift 32. Die dafür erforderlichen Schritte wurden bereits Off-Line ausgeführt, um nicht wertvolle On-Line-Kapazitäten des Rechnersystems hierzu in Anspruch nehmen zu müssen.

Eine virtuelle Adresse wird in eine entsprechende physikalischen Adresse z. B. unter Verwendung einer Übersetzungstabelle übersetzt, die üblicherweise als Page Table bezeichnet wird. Die Page Table kann in Form einer einzigen Tabelle organisiert sein, die Einträge hat, wobei jeder Eintrag eine virtuelle Adresse und die derselben zugeordnete physikalische Adresse umfaßt. Wie später ausgeführt wird, wird es gemäß der vorliegenden Erfindung jedoch bevorzugt, die Abbildungsvor-

schrift in Form eines hierarchischen Seitenzuordnungsbaums zu organisieren. Eine solchermaßen organisierte Abbildungsvorschrift hat den Vorteil größerer Flexibilität zum Verwalten von Zugriffsrechten. Dieselbe ist ferner besser geeignet, um kleine Seitengrößen handzuhaben, was dann von Bedeutung ist, wenn das Rechnersystem als Sicherheits-IC in einer Chipkarte verwendet wird. Kleine Seitengrößen, z. B. kleiner oder gleich 256 Byte, dienen ferner dazu, eine Seitentabellen-Fragmentierung zu vermeiden. Eine Speicherseite hat daher, wie es bezugnehmend auf Fig. 2 ausgeführt worden ist, z. B. eine Größe von 64 Byte, d. h. 512 Bit. Dies bedeutet, daß der Seiten-Offset, um die 64 Byte ausgehend von der Startadresse für die Seite adressieren zu können, eine Länge von 6 Bits haben muß.

Im nachfolgenden wird auf Fig. 3 Bezug genommen. Fig. 3 zeigt eine schematische Darstellung einer Adressübersetzung, bei der eine Page Table verwendet wird. Eine virtuelle Adresse 30, ein AMO-Feld 32 und ein TID-Feld 34 werden als Eingabe verwendet. Das AMO-Feld 32 bezeichnet den Zugriffs-Modus, der durch den gegenwärtigen Zustand der CPU und den beabsichtigten Zugriffstyp (Lesen, Schreiben, Ausführen, etc.) eingestellt wird. Das TID-Feld 34 wird beim Multitasking-Betrieb benötigt und liefert einen Aufgaben-Identifizierer (Task Identifier), der darauf hinweist, welcher Aufgabe die virtuelle Adresse 30 zugeordnet ist, um verschiedene virtuelle Adressräume verschiedener Anwendungen unterscheiden zu können. Aus der virtuellen Adresse 30 und dem TID-Feld 34 wird die sogenannten erweiterte virtuelle Adresse erhalten, die die Startadresse für die virtuelle Seite (VP 36) und einen Versatzwert (DP 38) aufweist.

Die Zuordnungsvorschrift in Form einer Seitentabelle 40 umfaßt verschiedene Einträge, wobei jeder Eintrag eine Spalte 42 für die Startadresse der virtuellen Seite und eine Spalte 44 für die Startadresse der physikalischen Seite, die der in der Spalte 42 stehenden virtuellen Seite zugeordnet ist, auf-

weist. Die Seitentabelle umfaßt gemäß einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung ferner eine Spalte 46 für Zugriffsrechte (EAR; EAR = Effective Access Rights), wobei anhand eines Rechtemoduls 48 überprüft wird, ob der durch das AMO-Feld 32 festgelegte CPU-Modus Zugriff auf eine virtuelle Adresse mit einem bestimmten EAR-Feld hat. Wird festgestellt, daß die CPU keinen Zugriff auf eine virtuelle Adresse hat, so wird die Adressübersetzung verweigert, und es wird eine Zugriffsverletzung ausgegeben. Die CPU kann daher, da die Zugriffsrechteüberprüfung vor der Adressübersetzung, also im virtuellen Adressraum, stattfindet, nicht einmal die physikalische Adresse geschweige denn den Inhalt der Speicherzelle, die durch die physikalische Adresse adressiert ist, erhalten. In der Seitentabelle wird eine assoziative Suche durchgeführt, um die Abbildung der virtuellen Adresse auf die physikalische Adresse zu liefern. Die virtuelle Adresse in der Seitentabelle muß mit dem Feld 36 der erweiterten virtuellen Adresse übereinstimmen. Wird kein derartiger Eintrag in der Seitentabelle gefunden, so wird durch ein Modul 50 ein Seitenfehler ausgegeben. Wird dagegen ein passender Eintrag gefunden, so wird aus der Spalte 44 die physikalische Seitenadresse ausgelesen. Bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung ist eine virtuelle Seite genau so groß wie eine physikalische Seite, und sind der Versatz der virtuellen Adresse (DP 38) und der Versatz 52 der physikalischen Adresse gleich groß, so daß keine Speicherung von Versatzwerten in der Page Table bzw. eine spezielle Verarbeitung von Versatzwerten nötig ist.

Wie es bereits bezugnehmend auf Fig. 1 ausgeführt worden ist, umfaßt die Speicherverarbeitungseinheit 18 vorzugsweise einen TLB, um eine schnellere Adressierung zu erreichen. Die anhand von Fig. 3 beschriebene Seitentabelle wird gemäß einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung in dem TLB gehalten, wobei der TLB optional zusätzlich zur Hardware-Zustandsmaschine 18a vorhanden ist. Der TLB ist in Form der in Fig.3 gezeigten Cache-Tabelle ausgestaltet und umfaßt eine

Liste von virtuellen Adressen mit entsprechenden physikalischen Adressen für einen schnellen Zugriff. Der TLB wird mit den zuletzt verwendeten Adressenpaaren gefüllt und kann entweder zufällig oder nach zeitlichen Gesichtspunkten aktualisiert werden. Eine Möglichkeit besteht beispielsweise darin, daß, sobald der TLB aufgefüllt ist, und eine neuer Eintrag hinzugefügt werden soll, der älteste Eintrag gelöscht wird, um Platz für einen neuen Eintrag zu machen. Die Größe des TLB ist ein Hardware-Faktor und kann daher je nach Ausführungsform spezifisch gewählt werden.

Sind die Hardware-Ressourcen beispielsweise aufgrund des benötigten Chipplatzes begrenzt, so wird der TLB nicht all zu groß ausgestaltet sein, und die Hardware-Zustandsmaschine wird öfters aktiviert werden, während in anderen Fällen, bei denen der TLB sehr groß gehalten werden kann, die Hardware-Zustandsmaschine lediglich zu Beginn aktiv ist, wenn der TLB, der ein flüchtiger Puffer ist, noch leer ist, um den TLB nach und nach aufzufüllen.

Insbesondere aufgrund von Größenbegrenzungen für den TLB können oftmals nicht alle virtuellen Seiten mittels des TLB auf ihre entsprechenden physikalischen Seiten abgebildet werden. Darüber hinaus müssen aufgrund der starken Überdimensionierung des virtuellen Adressraums gegenüber dem physikalischen Adressraum nicht alle virtuellen Seiten auf physikalische Seiten abgebildet werden, sondern es müssen lediglich virtuelle Adressen für die physikalischen Adressen vorhanden sein, an denen tatsächlich Code oder Daten für die tatsächlich laufenden Tasks einer Multitasking-Umgebung gespeichert sind. Wenn eine virtuelle Seite nicht auf eine physikalische Seite abgebildet wird, die entweder nicht vorhanden ist, oder die auf keine tatsächlich vorhandene physikalische Speicherzelle zeigt, so wird ein Seitenfehler ausgegeben, was darauf hinweist, daß die virtuelle Adresse beschädigt war, oder daß bei der Adressübersetzung ein Fehler aufgetreten ist.

Falls im TLB kein Eintrag mit einer entsprechenden virtuellen Adresse gefunden wird, so wird gemäß der vorliegenden Erfindung die Hardware-Zustandsmaschine 18a von Fig. 1 aktiviert, um eine virtuelle Adresse, die im TLB nicht gefunden worden ist, in eine physikalische Adresse umzurechnen, um die physikalische Adresse zusammen mit ihrer virtuellen Adresse in den TLB zu laden. Zu diesem Zweck wird, wie es ausgeführt worden ist, die Hardware-Zustandsmaschine auf den nicht-flüchtigen Speicher (z. B. 12 oder 14 von Fig. 1) zugreifen, um unter Verwendung der dort gespeicherten Abbildungsvorschrift die physikalische Adresse zu ermitteln.

Erfindungsgemäß wird als Abbildung eine hierarchische Baumstruktur mit physikalisch adressierten Knoten bevorzugt. Eine solche hierarchische Baumstruktur, die auch als Multilevel-Seitentabellen-Abbildungsvorschrift bezeichnet werden kann, hat den Vorteil, daß nicht eine große Seitentabelle im nicht-flüchtigen Speicher gehalten werden muß, sondern daß statt einer großen Tabelle mehrere Ebenen oder Levels mit kleineren Listen eingesetzt werden können. Dies erlaubt eine effizientere Verwaltung insbesondere bei kleinen physikalischen Speicherseiten.

Die Hardware-Zustandsmaschine 18a ist dann in der Lage, die hierarchische Baumstruktur von Knoten zu Knoten zu durchlaufen, um schließlich eine physikalische Adresse zu einer gegebenen virtuellen Adresse zu ermitteln. Dieser Prozeß wird als „Page Table Walking" bezeichnet.

Ein solcher Vorgang wird anhand von Fig. 4a beschrieben. In Fig. 4a ist eine virtuelle Adresse 400 gezeigt, die verschiedene Abschnitte 402 bis 310 aufweist. Der Abschnitt 410 ist einer Wurzelebene, d. h. der höchsten Ebene, zugeordnet. Der Abschnitt 408 ist der nächsthöheren Ebene, im Beispiel der Ebene 4, zugeordnet. Der Abschnitt 406 der virtuellen Adresse 400 ist entsprechend der Ebene 3 zugeordnet. Der Abschnitt 404 ist der Ebene 2 zugeordnet, während der Abschnitt 402 der

Ebene 1, d. h. dem Endknoten, zugeordnet ist. Der letzte Abschnitt der physikalischen Adresse, der auch als Abschnitt für die Ebene 0 bezeichnet werden kann, enthält den Seiten-Versatz, der in Fig. 3 mit dem Bezugszeichen 38 bezeichnet ist. Die virtuelle Adresse umfaßt ferner eine sogenannten Package-Adresse 412, die ein Speicher-Package adressiert. Bei dem bevorzugten Ausführungsbeispiel ist der virtuelle Adressraum in 256 gleich große Packages aufgeteilt, so daß jede Package einen Adressraum von 16 MB hat. Damit ist es beispielsweise möglich, für verschiedene Speicher-Packages des virtuellen Adressraum verschiedene Zugriffsrechte zu vergeben.

Der Abschnitt 410 der virtuellen Adresse 400, der bei einem bevorzugten Ausführungsbeispiel lediglich 1 Bit umfaßt, ist einem Wurzelknoten der hierarchischen Baumstruktur zugeordnet.

Die Liste für den Wurzelknoten kann im nicht-flüchtigen oder in Registern der Speicherverwaltungseinheit gespeichert sein. Alternativ kann die Liste für den Wurzelknoten auch an einem fest vereinbarten Platz in dem physikalischen Speicher abgelegt sein.

Die Liste für den Wurzelknoten wird als Package-Descriptor-Buffer 414 bezeichnet und umfaßt aufgrund der Tatsache, daß der Abschnitt 410 der virtuellen Adresse 400 lediglich ein Bit hat, lediglich zwei Listeneinträge. Ein Eintrag in der Wurzel-Liste 414 umfaßt einen Index, der durch das Bit des Abschnitts 410 der virtuellen Adresse indexiert ist. Hat das Bit im Abschnitt 410 einen Wert von Eins, wie es bei dem in Fig. 4a bezeichneten Beispiel der Fall ist, so wird der erste Eintrag der Liste 414 ausgewählt. Der Eintrag umfaßt ferner einen Zeiger 416 auf die physikalische Adresse der Seite im nicht-flüchtigen Speicher, an der eine Liste 418 für den ersten Zwischenknoten gespeichert ist, dem der Abschnitt 408 der virtuellen Adresse 400 zugeordnet ist. Ist das Bit in dem Abschnitt 410 der virtuellen Adresse dagegen eine Null, so wird

der zweite Eintrag der Liste 414 ausgewählt, der einen Zeiger 420 auf die physikalische Adresse einer Speicherseite im nicht-flüchtigen Speicher umfaßt, in der eine weitere Liste 422 für den ersten Zwischenknoten gespeichert ist, dem der Abschnitt 408 der virtuellen Adresse 400 zugeordnet ist. Nachdem der Abschnitt 408 der virtuellen Adresse vier Bits umfaßt, haben die Listen 418 und 422 für den ersten Zwischenknoten jeweils 16 Einträge. Jeder Eintrag hat eine Länge von 32 Bits, so daß bei dem in Fig. 4a gezeigten Ausführungsbeispiel jede Liste genau eine Speicherseite im nicht-flüchtigen Speicher in Anspruch nimmt.

Nachdem die Hardware-Zustandsmaschine den oberen Eintrag der Wurzelliste 414 aufgrund des Abschnitts 410 der virtuellen Adresse 400 ermittelt hat, kann die Hardware-Zustandsmaschine aufgrund des Zeigers 416 unmittelbar auf die physikalische Speicherseite zugreifen, in der die Liste 418 für den ersten Zwischenknoten gespeichert ist. Die Hardware-Zustandsmaschine liest dann den Abschnitt 408 der virtuellen Adresse 400 ein und wählt aufgrund der Tatsache, daß der Abschnitt den Wert „1100" hat, den dreizehnten Eintrag in der Liste 418 aus.

Der dreizehnte Eintrag umfaßt neben einem Eintragsindex wieder einen Zeiger 424 auf eine Liste 426 für einen weiteren Zwischenknoten, dem der Abschnitt 406 der virtuellen Adresse 400 zugeordnet ist.

Die Hardware-Zustandsmaschine liest dann den Abschnitt ein und wählt den ersten Eintrag der Liste 426 aus, da der Abschnitt 406 den Wert „0000" hat.

Der erste Eintrag in der Liste 426 enthält wiederum einen Zeiger 428 auf eine Liste 430, die einem weiteren Zwischenknoten mit niedrigerer Hierarchie zugeordnet ist. Die Hardware-Zustandsmaschine liest nun den Abschnitt 404 der virtuellen Adresse 400 ein und wählt den achten Eintrag dieser Liste aus, da der Wert des Abschnitts 404 „0111" beträgt.

Der ausgewählte Eintrag der Liste 430 enthält wiederum einen Zeiger 432 auf eine physikalische Adresse der physikalischen Seite im nicht-flüchtigen Speicher, in der eine Liste 434 für einen Endknoten, der dem Abschnitt 402 der virtuellen Adresse 400 zugeordnet ist, gespeichert ist. Die Hardware-Zustandsmaschine liest nun den Abschnitt 402 der virtuellen Adresse ein und wählt aus der Liste 434 aufgrund der Tatsache, daß der Wert des Abschnitts 402 „10011" beträgt, den zwanzigsten Eintrag der Liste 434 aus. Da der Abschnitt 402 der virtuellen Adresse 400 der End-Ebene zugeordnet ist, enthält der ausgewählte Eintrag der Liste 434 für den Endknoten einen Zeiger 436 auf die physikalische Adresse der physikalischen Seite, die der virtuellen Startadresse der virtuellen Seite entspricht. Zu der physikalischen Adresse, auf die der Zeiger 436 zeigt, muß nun lediglich noch der Seitenversatz hinzugefügt werden, wie es durch einen gestrichelten Pfeil 438 in Fig. 4a symbolisch dargestellt ist, um die physikalische Adresse 440 zu erhalten, die der virtuellen Adresse 400 zugeordnet ist.

Es sei darauf hingewiesen, daß das in Fig. 4a beschriebene Konzept ebenso verwendet werden kann, wenn keine seitenweise Speicherorganisation eingesetzt wird, sondern eine direkte Adressierung von Speicherzellen ohne Seitenadresse und Versatzwert. In diesem Fall würde einfach der letzte Schritt, der durch den Pfeil 438 in Fig. 4a dargestellt ist, entfallen.

Wie es bereits ausgeführt worden ist, ist der virtuelle Adressraum wesentlich größer als der physikalische Adressraum. Aus diesem Fall existieren in den in Fig. 4 gezeigten Listen für die verschiedenen Knoten viele Einträge, die auf keinen nächst-niedrigeren Knoten verweisen. In Fig. 4a sind dies sämtliche Einträge, die keinen Ausgangs-Zeiger enthalten. Diese Einträge werden auch als Null-Einträge bezeichnet. Aufgrund der Übersichtlichkeit der Darstellung wurde bei sämtli-

chen Zeigern, die in Fig. 4a ins Leere zeigen, die denselben zugeordneten Listen weggelassen. Eine vollständige Abbildungsvorschrift würde jedoch zu jedem Zeiger in Fig. 4a eine entsprechende Liste umfassen.

Abgesehen von dem PAD-Abschnitt 412 wird bei dem in Fig. 4a gezeigten Ausführungsbeispiel die virtuelle Adresse 400 in sechs Teile oder Abschnitte aufgeteilt. Jedem Abschnitt ist eine Ebene in dem Adressübersetzungsprozeß, der durch die Hardware-Zustandsmaschine ausgeführt wird, zugeordnet. Die höchste Ebene, d. h. Ebene 5, indexiert, wie es ausgeführt worden ist, einen von zwei Einträgen in der Liste 414. Selbstverständlich könnte auch die Liste 414 mehrere Einträge enthalten. In diesem Fall müßte der Abschnitt 410 der virtuellen Adresse 40 dementsprechend mehr Bit haben. Mit zwei Bits könnten bereits vier Einträge in der Liste 414 vorhanden sein.

Fig. 4b zeigt den Adressbereich, der durch jede Ebene gewissermaßen adressiert werden kann. Durch die Wurzel-Liste 414 (letzte Zeile der Tabelle in Fig. 4b) können die gesamten 16 Megabyte des virtuellen Adressraum adressiert werden, wobei der obere Zeiger 416 die oberen acht Megabyte auswählt, während der untere Zeiger 420 die unteren acht Megabyte auswählt. Daher kann durch die Liste 418 bzw. analog dazu durch die Liste 422 jeweils ein physikalischer Adressraum von acht Megabyte adressiert werden, wobei jeder Eintrag der Liste 418, d. h. jeder Pfeil, der aus der Liste 418 heraus zeigt, 512 Kilobyte adressieren kann. Jeder Zeiger aus der Liste 418 zeigt nämlich auf eine Liste 426 der dritten Ebene (dritte Zeile der Tabelle von Fig. 4b). Analog dazu kann wieder jeder Zeiger aus einer Liste der dritten Ebene, wie z. B. der Zeiger 428, wieder einen Adressbereich von 32 Kilobyte adressieren. Der Adressbereich von 32 Kilobyte ist der Adressbereich, der durch die Liste 430 der zweiten Ebene umspannt wird, wobei ein Eintrag in dieser Liste wiederum einen Adressbereich von zwei Kilobyte adressieren kann.

Fig. 5 zeigt einen Ausschnitt aus der hierarchischen Page Table Struktur von Fig. 4a, jedoch mit einer anderen virtuellen Adresse, die in den Abschnitten 408 und 406 für die vierte Ebene und für die dritte Ebene jeweils lauter Einsen hat. Die Abbildungsvorschrift in Form des hierarchischen Baums, der mit der Wurzelliste 414 beginnt und mit der physikalischen Adresse 440 endet, erlaubt im Gegensatz zu der in Fig. 4a gezeigten Abbildungsvorschrift ein Überspringen von mindestens einer Ebene. Das Überspringen einer Ebene wird in der virtuellen Adresse durch einen bestimmten Abschnitt signalisiert, beispielsweise dadurch, daß in einem Abschnitt lauter Einsen stehen, wie es in Fig. 5 der Fall ist. Für das Überspringen einer Ebene könnte jedoch auch jeder andere vorbestimmte Code reserviert sein. Der Zeiger, der von der Wurzelliste 414 ausgeht und in Fig. 5 mit 500 bezeichnet ist, zeigt nun nicht mehr auf eine Liste der vierten Ebene oder der dritten Ebene, sondern gleich auf die Liste 430 der zweiten Ebene. Ein Überspringen einer beliebigen Anzahl von Ebenen bzw. Knoten ist möglich. Die Bits in der virtuellen Adresse, die diesen Ebenen entsprechen, müssen den vorbestimmten Code haben, um der Zustandsmaschine das Ebenen-Überspringen zu signalisieren. Diese Optimierung erfordert eine zusätzliche Information für den Zeiger 500, wobei diese zusätzlichen Informationen in den Einträgen der Wurzelliste 414 abgespeichert sind. Selbstverständlich könnte auch nur die Ebene 3 übersprungen werden, während der Knoten der Ebene 4 nicht übersprungen werden darf. In diesem Fall müßten die zusätzlichen Informationen in einem Eintrag einer Liste für den Knoten der Ebene 4 vorhanden sein.

In nachfolgenden wird auf Fig. 6 Bezug genommen. Fig. 6 zeigt eine zu Fig. 4a und Fig. 5 korrespondierende Tabelle für die Bedeutung der Bits der virtuellen Adresse und den Zusammenhang zwischen der Knotengröße, d. h. der maximalen Anzahl von Einträgen in die Knotenliste. Anders ausgedrückt stellt jede Zeile der Tabelle in Fig. 6 einen Abschnitt der ganz links in

Fig. 6 bezeichneten Ebene der virtuellen Adresse dar. In der ersten Zeile der Tabelle von Fig. 6, die sich auf die Ebene 5, also auf die Wurzelebene, bezieht, ist ausgeführt, daß der Abschnitt 410 lediglich ein Bit hat, nämlich bei dem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung das Bit 23 der virtuellen Adresse. Durch ein Bit können zwei verschiedene Einträge in die Wurzelliste indexiert werden, so daß die Knotengröße des Wurzelknotens 2 beträgt. Die vierte Ebene umfaßt die Bits 19-22. Durch vier Bits können 16 verschiedene Einträge in einer Liste für die vierte Ebene indexiert werden, so daß die Knotengröße, also die Größe einer Liste der vierten Ebene 16 beträgt. Die Anzahl der Listen in der vierten Ebene beträgt 2, da die Wurzelliste zwei Einträge hat.

Sowohl der Abschnitt für die dritte Ebene als auch der Abschnitt für die zweite Ebene haben beide vier Bits, so daß eine Liste der zweiten Ebene oder der dritten Ebene ebenfalls maximal 16 Einträge haben kann. Der Abschnitt 402 für die erste Ebene umfaßt fünf Bits, so daß durch diesen Abschnitt 32 Einträge einer Liste indexiert werden können, wie es auch aus der Liste 434 für die erste Ebene von Fig. 4a deutlich wird, dieselbe hat 32 Zeilen, während eine Liste der zweiten Ebene lediglich 16 Zeilen hat. Die nullte Ebene, d. h. der Seitenversatz ausgehend von einer Seitenanfangsadresse, umfaßt sechs Bits, so daß hiermit 64 Listeneinträge indexiert werden können. Nachdem eine physikalische Seite 64 Byte hat, können minimale Versätze von einem Byte adressiert werden. Wäre die Seitengröße beispielsweise 128 Byte, so wäre die minimale Speichergröße, die adressiert werden kann, zwei Byte groß.

Im nachfolgenden wird auf Fig. 7 Bezug genommen. In Fig. 7 sind ebenso wie in den Fig. 5 und 4a benutzte Einträge in eine Liste fett gezeichnet, während Null-Einträge als nichtausgefülltes Rechteck dargestellt sind. Aus der in Fig. 7 gezeigten Abbildungsvorschrift wird klar, daß sehr viele Null-

Einträge vorhanden sind, und lediglich ein paar verwendete Einträge. Dies rührt daher, daß der virtuelle Adressraum wesentlich größer ist als der physikalische Adressraum. Bei dem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung ist der virtuelle Adressraum 4 Gigabyte groß, während der physikalische Adressraum lediglich 4 Megabyte groß ist. Wenn eine seitenweise Speicherorganisation verwendet wird, muß jedoch dennoch für jede in Fig. 7 gezeigte Liste eine Speicherseite vorgesehen werden. Bei dem in Fig. 7 gezeigten Beispiel für eine Abbildungsvorschrift müssen daher 10 Speicherseiten verwendet werden, um lediglich 4 physikalische Seiten, in denen Code bzw. Daten vorhanden sind, zu verwalten. In Anwendungen, bei denen eine ausreichende Menge an nicht-flüchtigem Speicher vorhanden ist, um die Listen für die Ebenen zu speichern, spielt dies keine Rolle.

Wenn jedoch der Speicherplatz begrenzt ist und eine wertvolle Ressource ist, so wird es bevorzugt, die n-Knoten-Listen zu komprimieren, um das Verhältnis der Anzahl der verwendeten Einträge einer Liste zur Gesamtanzahl der Einträge in der Liste zu vergrößern. Es wird daher von den sogenannten n-Knoten zu den q-Knoten übergegangen. Während der Ausdruck „n-Knoten" bedeutet, daß sämtliche Listen eines Knotens einer Ebene dieselbe Anzahl von Einträgen, nämlich n Einträge hat, bedeutet der Ausdruck „q-Knoten", daß die Knotenliste komprimiert ist, und daß die Anzahl der Einträge in eine Liste für ein und dieselbe Ebene von Liste zu Liste variieren kann.

Teilweise gefüllte Knotenlisten werden komprimiert und, wie es insbesondere bezugnehmend auf Fig. 11 ausgeführt wird, können mehrere komprimierte Knoten auf einer Seite des physikalischen Speicherbereichs abgespeichert werden. Die komprimierten Knoten werden q-Knoten genannt, wobei die hierarchische Baumstruktur mit komprimierten Knoten als q-Baum bezeichnet wird.

Die Theorie, die hinter einem q-Knoten steht, besteht darin, daß alle verwendeten Einträge einer Liste, d. h. alle Nicht-Null-Zeiger, in einem n-Knoten in einer Struktur plaziert werden können (dem q-Knoten), die kleiner als der ursprüngliche n-Knoten ist. Dies wird erreicht, indem der n-Knoten in kleinere Abschnitt aufgeteilt wird, wobei für eine maximale Kompression der kleinste Abschnitt genommen wird, der alle Nicht-Null-Zeiger enthält. Um die Position eines Zeigers in dem n-Knoten zu spezifizieren, wird ein Offset-Wert für den q-Knoten-Zeiger benötigt. Dieser Offset-Wert wird auch als virtueller Offset bezeichnet.

Im nachfolgenden wird auf Fig. 8 Bezug genommen, um eine mögliche Kompressionsart einer Liste 800 eines n-Knotens darzustellen. Die Liste 800 umfaßt zwei Nicht-Null-Einträge, die binär durch 1100 und 1110 indexiert werden können. Neben der Liste 800 sind mögliche q-Knoten dargestellt. Bei der beschriebenen abschnittsweisen Kompression können prinzipiell drei verschiedene „komprimierte" Listen, d. h. q-Knoten, 802, 804 und 806 erzeugt werden. Die Liste 802 entspricht der Liste 800. Hier handelt es sich um die triviale Form der Kompression, die q-Knoten-Liste 802 ist genau so groß wie die Liste 800, und es wird kein Offset-Bit benötigt. Dagegen ist die Liste 804 bereits auf die Hälfte des Speicherplatzes komprimiert. Die Liste enthält zwei Einträge, die mit 110 und 100 indexiert werden können. Um von den Einträgen der komprimierten Liste 804 auf die Einträge der nicht-komprimierten Liste 800 zu kommen, wird ein virtuelles Offset-Bit benötigt, das einen Wert von 1 hat. Das virtuelle Offset-Bit entspricht dem höchstwertigen Bit (msb) beider Einträge der Liste 800. Eine Kompression ist also möglich, wenn die msbs der Nicht-Null-Einträge der Liste 800 gleich sind. Da auch die höchstwertigen Bits der Einträge der komprimierten Liste 804 gleich sind, ist noch eine höhere Kompression zu erreichen, wie es die komprimierte Liste 806 zeigt. Die beiden Nicht-Null-Einträge der Liste 806 haben höchstwertige Bits, die nicht gleich sind, so daß keine weitere Kompression möglich

ist. Um von der komprimierten Liste 806 wieder auf die nicht-komprimierte Liste 800 zu kommen, werden zwei Offset-Bits benötigt, nämlich die zwei höchstwertigen Bits der Einträge in der Liste 800, die für beide Nicht-Null-Einträge in der Liste gleich sind.

Abhängig vom Ausmaß der Kompression des Knotens (keine Kompression, eine Kompression von 16 Einträgen auf 8 Einträge und schließlich eine Kompression von 16 Einträgen auf 4 Einträge) beträgt der virtuelle Offset-Wert 0, 1 Bit oder 2 Bits. Die Kompression auf 8 Einträge, d. h. eine halbe Speicherseite, bedeutet, daß die Zeiger im ursprünglichen n-Knoten nur in entweder der oberen oder der unteren Hälfte sind. Der virtuelle Offset muß daher 1 Bit betragen, um die Position zu spezifizieren, wobei das Offsetbit mit einem Wert von 0 bedeutet, daß sämtliche Nicht-Null-Einträge in der unteren Hälfte sind, während ein Offsetbit von 1 bedeutet, daß sämtliche Einträge der Liste in der oberen Hälfte der n-Knotenliste 800 sind.

Analog dazu ist, wie es anhand von Fig. 8 dargestellt worden ist, bei der Liste 800 eine weitere Kompression möglich, da sämtliche Nicht-Null-Einträge nicht nur in der oberen Hälfte sondern sogar im oberen Viertel der Liste sind. In diesem Fall sind die Offset-Bits bei einem Wert von „11", was anzeigt, daß die Zeiger in der n-Knoten-Liste im vierten Viertel zu finden sind. Wenn keine Kompression durchgeführt wird (Liste 802), weil das Kompressionsverfahren keine Kompression erlaubt, wie es anhand von Fig. 9 dargestellt wird, werden auch keine Offset-Bits benötigt. Die Größe der Liste im nächstniedrigeren Knoten wird in der Knotenliste spezifiziert, die den Zeiger umfaßt, der auf den nächstniedrigeren (komprimierten) Knoten zeigt. Wenn beispielsweise die Liste 430 von Fig. 4a komprimiert ist, beispielsweise um die Hälfte, so würde der Eintrag 426 der nächsthöheren Liste neben der physikalischen Adresse, an der die Liste 430 gespeichert ist, auch die Größe der Liste spezifizieren. Die Liste 430

wäre dann beispielsweise bei einer Kompression um die Hälfte, lediglich 8 Einträge groß, obgleich der Abschnitt 404 der virtuellen Adresse, der dem Level 2 entspricht, 4 Bits hat und eigentlich eine 16-Einträge-Liste indexiert. Der Abschnitt 404 der virtuellen Adresse wäre dann so gestaltet, daß ein Bit desselben, typischerweise das höchstwertige Bit, als virtuelles Offsetbit interpretiert wird.

Dadurch ergibt sich auch ein weiterer Kontrollmechanismus. Die Hardware-Zustandsmaschine wird das höchstwertige Bit des Abschnitts 404 mit dem Größenbit im Nicht-Null-Eintrag der Liste 426 vergleichen und bei Übereinstimmung eine Adressübersetzung fortsetzen, während, falls die Bits nicht übereinstimmen, ein Seitenfehler ausgegeben wird, da dann zumindest entweder die Abbildungsvorschrift oder die virtuelle Adresse fehlerhaft ist.

Im nachfolgenden wird auf Fig. 9 eingegangen, um weitere Beispiele darzustellen, wie n-Knoten auf ihre minimalen q-Knoten reduziert werden können, wenn das bezüglich Fig. 8 beschrieben Kompressionsverfahren verwendet wird. Die Liste 900 in Fig. 9 umfaßt lediglich einen Nicht-Null-Eintrag, der durch die Bitkombination „1100" indexiert ist. Der minimale q-Knoten umfaßt lediglich einen einzigen Eintrag und, dementsprechend, 4 virtuelle Offsetbits „1100". Eine Liste 900 mit lediglich einem einzigen Nicht-Null-Eintrag kann somit auf einfache Art und Weise hinsichtlich ihres Speicherplatzbedarfs um das 1/16-fache reduziert werden.

Die Liste 902 umfaßt zwei Nicht-Null-Einträge, die mit „0001" und „0011" indexiert werden können. Die beiden Einträge haben zwei gleiche höchstwertige Bits, so daß zwei virtuelle Offsetbits 00 entstehen, und die Liste um ein Viertel reduziert werden kann.

Die Liste 904 umfaßt zwei Nicht-Null-Einträge, deren höchstwertige Bits jedoch ungleich sind, so daß bei dem hier

gewählten beispielhaften Kompressionsalgorithmus keine Kompression erreichbar ist. Der q-Knoten ist daher genau so groß wie der n-Knoten.

Die Beispielliste 906 umfaßt zwei Einträge mit „0100" und „0101". Die drei höchstwertigen Bits sind gleich, so daß eine Kompression um das 1/8-fache erreicht werden kann, was zu den virtuellen Offset-Bits „010" führt.

Die Beispielliste 908 umfaßt vier Nicht-Null-Einträge zwischen „1010" und „1101". Alle vier Einträge haben dasselbe höchstwertige Bit, so daß eine Kompression um das ½-fache erreicht werden kann, was zu einem virtuellen Offset-Bit von „1" führt.

Es sei darauf hingewiesen, daß die q-Knoten der Ebene 1 (Fig. 4a) eine spezielle Rolle spielen. Da ihre Einträge nicht auf weitere q-Knoten zeigen, sondern direkt auf die physikalischen Speicherseiten, müssen keine zusätzlichen Informationen, wie z. B. ein Größenwert einer hierarchisch niedrigen q-Liste, zusammen mit den Zeigern gespeichert werden. Folglich wird ein Eintrag in einer Liste auf Ebene 1 dazu verwendet, zwei Zeiger zu speichern. Daher umfaßt der Abschnitt 402 der virtuellen Adresse, der der Ebene 1 zugeordnet ist, fünf virtuelle Adressbits. Das zusätzliche Bit spezifiziert, welcher der Zeiger in dem ausgewählten q-Knoten-Eintrag verwendet werden soll. Es sei darauf hingewiesen, daß auch einer der zwei Zeiger 0 sein kann. Nachdem ein Eintrag in einer Liste der Ebene 1 zwei Zeiger speichert, ist die Listenlänge einer Liste, wie z. B. der Liste 434 von Fig. 4a, doppelt so groß wie die Länge einer Liste einer höheren Ebene, wie z. B. der Ebene 430.

Im nachfolgenden wird auf Fig. 10 Bezug genommen, um darzustellen, wie die komprimierten q-Knoten dazu eingesetzt werden können, um die Speicherbelegung zu minimieren. Zunächst werden beispielsweise gemäß der Tabelle von Fig. 9 die mini-

malen q-Knoten einer hierarchischen Abbildungsvorschrift identifiziert. Daraus entsteht eine Abbildungsvorschrift, wie sie in Fig. 10 dargestellt ist. Die ausgefüllten Rechtecke bezeichnen benutzte Einträge, während die nicht-ausgefüllten Rechtecke Null-Einträge bezeichnen. Jeder q-Knoten ist durch eine dicke Linie umrandet. Es ist zu sehen, daß bei dem gewählten Beispiel der Abbildungsvorschrift sämtliche Listen außer einer Liste 1000 komprimiert werden konnten. Die Liste 1000 konnte mit dem gewählten Kompressionsverfahren nicht komprimiert werden, da die zwei Nicht-Null-Einträge, die dieselbe umfaßt, nicht in der derselben Hälfte der Liste aufgeführt sind.

Bei der in Fig. 10 gezeigten Abbildungsvorschrift sind noch alle, auch die komprimierten, Listen an jeweils eigenen physikalischen Speicherseiten oder „Page Frames" abgelegt, so daß die Page Frames sehr locker belegt sind, es wurde jedoch noch keine Reduktion der Anzahl von Page Frames erreicht. Die nicht-belegten Worte in den Page Frames können bei dem in Fig. 10 gezeigten Ausführungsbeispiel jedoch bereits durch andere Daten belegt werden, so daß in einer physikalischen Speicherseite eine Liste für die Abbildungsvorschrift und zusätzlich weitere Daten gespeichert werden können. Aus Speicherverwaltungsgründen wird es jedoch bevorzugt, zu einem Konzept gemäß Fig. 11 überzugehen, um die Abbildungsvorschrifts-Informationen, also die Listen für die einzelnen Ebenen, in möglichst wenigen physikalischen Speicherseiten zu konzentrieren, damit komplette Speicherseiten frei gemacht werden, um in denselben andere Daten abspeichern zu können.

Nachdem die meisten q-Knoten klein sind, können bei der in Fig. 11 dargestellten Abbildungsvorschrift alle Knotenlisten außer einer Knotenliste 1100 in die gleiche physikalische Speicherseite 1102 gepackt werden. Folglich wird die Abbildungsvorschrift bzw. die Seitenzuordnungsstruktur, von 10 auf 2 Speicherseiten reduziert. Auf diese Art und Weise liefert die q-Baum-Struktur eine beträchtliche Optimierung im Ver-

gleich zu der in Fig. 4a gezeigten Option, bei der keine Listenkompression durchgeführt worden ist.

Der einzige „verschwendete" Speicher ist aufgrund der Listen vorhanden, die mehrere Null-Zeiger enthalten und nicht weiter komprimiert werden können. Die Liste 1100 fällt in diese Kategorie. Erfahrungsgemäß existieren jedoch wenige solcher q-Knoten. Normale Programme verwenden üblicherweise zumindest einige zusammenhängende Speicherbereiche, wie z. B. Code, statische Daten, Stack und Heap. Speicherlayouts dieser Art haben für die Zuordnungsstruktur einen kleineren Zusatzaufwand. Dennoch sind manchmal fragmentierte Speicherlayouts notwendig.

Obgleich im vorhergehenden lediglich die in Fig. 9 näher dargestellte Listenkompressionsfunktion dargestellt worden ist, sei darauf hingewiesen, daß sämtliche Listenkompressionsverfahren eingesetzt werden können. Die Kompression muß also nicht ausschließlich darauf basieren, daß Einträge in derselben Hälfte bzw. in dem gleichen Viertel einer Liste sein müssen. So könnte die Liste 1100 beispielsweise dadurch komprimiert werden, daß ein alternatives Kompressionsverfahren verwendet wird, das darauf beruht, daß Einträge in unterschiedlichen Hälften der Liste sind. Je nach Layout der virtuellen Adresse und nach der Bedeutung der Offset-Bits in einem Abschnitt der virtuellen Adresse können auch mehrere verschiedene Kompressionsverfahren kombiniert werden, wenn die Speicheranforderungen so sind, daß auch eine geringe Anzahl von Listen, die nicht-komprimiert werden können, wie z. B. die Liste 1100 von Fig. 11, nicht akzeptierbar ist.

Bei einem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung wird ein Knotenadressierungsmodus (NAM; NAM = Node Addressing Mode) vorgesehen, um die Abbildungsvorschrift im nicht-flüchtigen Speicher modifizieren zu können. Hierzu wird das in Fig. 12 dargestellte Format für eine virtuelle Adresse 1200 verwendet. Die virtuelle Adresse ist wie in Fig. A in

mehrere Abschnitte aufgeteilt, wobei die Abschnitte 1210 bis 1202 den Abschnitten 410 bis 402 von Fig. 4a prinzipiell entsprechen. Der letzte Abschnitt 1212, der in Fig. 4a den Versatzwert bezeichnet hatte, wird bei der virtuellen Adresse 1200 jedoch dafür verwendet, die q-Knoten-Ebene zu signalisieren, deren Liste im NAM-Modus modifiziert werden soll.

Der NAM-Modus wird dazu verwendet, um virtuelle Adressen zu manipulieren, da aus Sicherheitsaspekten lediglich ein virtueller Adressierungsmodus gefahren werden soll, so daß die CPU keinen direkten Zugriff auf physikalische Seiten hat. Durch Einstellen des Abschnitts 1212 der virtuellen Adresse 1200 kann auf Listen für einzelne Ebenen und insbesondere auf die Einträge in diesen Listen zugegriffen werden.

Hierzu umfaßt jeder Paketbeschreiber, der in der Wurzelliste 414, d. h. dem Package Descriptor Buffer, abgespeichert ist, ein NAM-Bit, das, wenn es gesetzt ist, einen Zugriff auf die q-Knoten-Listen für diese spezielle Speicherpackage erlaubt. Es sei jedoch darauf hingewiesen, daß nur Packages in priviligierten Schichten, d. h. privilegierte Modi des Betriebssystems, das NAM-Bit manipulieren können. Wenn das NAM-Bit gesetzt ist, wird der letzte Abschnitt 1212 der virtuellen Adresse nicht mehr als Seitenversatzwert aufgefaßt, sondern wird dazu verwendet, um der Hardware-Zustandsmaschine zu signalisieren, ob der q-Knoten auf Ebene 4, auf Ebene 3, auf Ebene 2 oder auf Ebene 1 adressiert werden soll, um auf die Liste bzw. auf Einträge in der entsprechenden Liste zuzugreifen.

Wenn das NAM-Bit gesetzt ist, wird die virtuelle Adresse von der Hardware-Zustandsmaschine somit anders als bei dem in Fig. 4a beschriebenen Fall interpretiert.

Die Hardware-Zustandsmaschine führt nun, wenn das NAM-Bit gesetzt ist, eine Adressübersetzung lediglich durch, bis der q-Knoten der durch den Abschnitt 1212 definierten Stoppebene

wiedergewonnen ist. Dann wird ein endgültiger Buszyklus auf die physikalische Seite erzeugt, die durch den Zeiger eines Eintrags in der Liste für die definierte Stoppebene bezeichnet ist. Auf die angeforderten Listendaten wird dann zugegriffen und dieselben werden vorzugsweise in dem Datencache 20 (Fig. 1) gespeichert. Es sei darauf hingewiesen, daß diese Adressübersetzung nicht im TLB 18b (Fig. 1) gespeichert wird, da es sich um die Abbildungsvorschrift selbst handelt und nicht um eine Zuordnung einer virtuellen Adresse zu einer physikalischen Adresse.

Patentansprüche

1. Rechnersystem mit virtueller Adressierung, wobei eine virtuelle Adresse über eine Abbildungsvorschrift einer physikalischen Adresse zugeordnet ist, mit folgenden Merkmalen:

einem nicht-flüchtigen Speicher (12, 14) zum Speichern zumindest eines Teils der Abbildungsvorschrift; und

einer Hardware-Zustandsmaschine (18a), die auf den nicht-flüchtigen Speicher zugreifen kann (26) und ausgebildet ist, um unter Verwendung der virtuellen Adresse und zumindest des Teils der Abbildungsvorschrift die der virtuellen Adresse zugeordnete physikalische Adresse zu ermitteln.

2. Rechnersystem nach Anspruch 1, bei dem die virtuelle Adresse eine Adresse für eine virtuelle Speicherseite und einen virtuellen Versatzwert umfaßt, um ein Wort in der virtuellen Speicherseite zu adressieren, und bei dem die physikalische Adresse eine Adresse für eine physikalische Speicherseite und einen physikalischen Versatzwert umfaßt, um ein Wort in der physikalischen Speicherseite zu adressieren.

3. Rechnersystem nach Anspruch 2, bei dem die Abbildungsvorschrift derart ausgebildet ist, daß die virtuelle Speicherseite und die physikalische Speicherseite gleich groß sind, und der virtuelle Versatzwert und der physikalische Versatzwert gleich groß sind.

4. Rechnersystem nach einem der vorhergehenden Ansprüche, das angeordnet ist, um in einem virtuellen Modus initialisiert zu werden.

5. Rechnersystem nach einem der vorhergehenden Ansprüche, das ferner folgende Merkmale aufweist:

einen flüchtigen Speicher (18b), in dem Zuordnungen von physikalischen Adressen zu virtuellen Adressen speicherbar sind,

wobei das Rechnersystem angeordnet ist, um zunächst auf den flüchtigen Speicher (18b) zuzugreifen, um festzustellen, ob eine angeforderte virtuelle Adresse gespeichert ist, und

wobei das Rechnersystem angeordnet ist, um die Hardware-Zustandsmaschine (18a) zu einer Adressübersetzung zu aktivieren, falls die angeforderte virtuelle Adresse nicht in dem flüchtigen Speicher (18b) gespeichert ist.

6. Rechnersystem nach einem der vorhergehenden Ansprüche,

bei dem die Abbildungsvorschrift hierarchisch als Baum mit Knoten (414, 418, 422, 426, 430, 434) strukturiert ist, wobei der Baum einen Wurzelknoten (414), zumindest einen Zwischenknoten (418, 422, 426, 430) und einen Endknoten (434) aufweist,

bei dem für einen Knoten eine Liste in dem nicht-flüchtigen Speicher gespeichert ist, wobei die Liste eines Knotens Einträge aufweist, wobei ein Eintrag einen Eintragsindex und einen Zeiger (424, 428, 432, 436) auf eine physikalische Adresse aufweist, wobei der Zeiger für die physikalische Adresse auf eine physikalische Adresse des nicht-flüchtigen Speichers zeigt, mit der eine Liste für einen anderen Knoten des Baums wiedergewonnen werden kann,

bei dem die virtuelle Adresse (400) in Abschnitte (402, 404, 406, 408, 410) gegliedert ist, wobei für den Wurzelknoten (414), den zumindest einen Zwischenknoten (418, 422, 426, 430) und den Endknoten (434) je ein Abschnitt vorhanden ist, wobei der Abschnitt (402, 404, 406, 408, 410) einen Listeneintrag einer zugehörigen Liste identifiziert, und

bei dem die Hardware-Zustandsmaschine (18a) angeordnet ist, um aus der virtuellen Adresse einen Abschnitt zu extrahieren, um auf den nicht-flüchtigen Speicher (12, 14) zuzugreifen, um unter Verwendung des extrahierten Abschnitts einen Eintrag in einer Liste für den Knoten, der dem Abschnitt zugeordnet ist, zu ermitteln, und um das Extrahieren und das Zugreifen sequentiell so lange fortzusetzen, bis der Endknoten erreicht ist, um die physikalische Adresse (440), der die virtuelle Adresse (400) zugeordnet ist, zu erhalten.

7. Rechnersystem nach Anspruch 6,

bei dem der Paketbeschreibungspuffer (414) einen nicht-flüchtigen Speicheranteil aufweist, in dem die Liste für einen Wurzelknoten der Abbildungsvorschrift gespeichert ist,

wobei die Liste (414) für den Wurzelknoten zumindest zwei Einträge aufweist, wobei jeder Eintrag die physikalische Adresse (416) einer Liste für einen Zwischenknoten einer niedrigeren Knotenebene aufweist, und

wobei die virtuelle Adresse einen Wurzelabschnitt (410) aufweist, der den Eintrag in der Wurzelliste für die virtuelle Adresse identifiziert.

8. Rechnersystem nach Anspruch 6 oder 7,

bei dem die Hardware-Zustandsmaschine (18a) ausgebildet ist, um eine Zwischen-Knotenebene zu überspringen (500), wenn der Abschnitt (406, 408) der virtuellen Adresse (400), der der Zwischen-Knotenebene zugeordnet ist, einen vorbestimmten Wert aufweist, und

bei dem im Falle eines Überspringens eines Zwischen-Knotens der Eintrag in einer Liste des Ausgangsknotens die physikalische Adresse einer Liste eines Sprung-Ausgangsknotens auf-

weist, der in dem hierarchischen Baum auf einen übersprungenen Knoten folgt.

9. Rechnersystem nach einem der Ansprüche 6 bis 8,

bei dem die Anzahl der virtuellen Adressen größer als die Anzahl der physikalischen Adressen ist, so daß eine Liste Leereinträge aufweist, die nicht auf eine physikalische Adresse verweisen,

bei dem die in dem nicht-flüchtigen Speicher gespeicherte Liste eines Knotens komprimiert ist, so daß das Verhältnis der Anzahl von Einträgen, die einen Zeiger auf eine physikalische Adresse enthalten, zu der Gesamtanzahl der Einträge in der Liste im Vergleich zu einer nicht-komprimierten Liste erhöht ist, und

bei dem ein Eintrag in die Liste die Komprimierung des Knotens, zu dem die Liste gehört oder die Komprimierung eines in der Hierarchie niedrigeren Knotens identifiziert.

10. Rechnersystem nach Anspruch 9, bei dem der nicht-flüchtige Speicher (12, 14) seitenweise organisiert ist,

bei dem zumindest zwei komprimierte Listen in der selben Speicherseite (1102) gespeichert sind, und

bei dem ein Listeneintrag einer Liste eines höheren Knotens die physikalische Adresse der Speicherseite sowie einen Versatzwert umfaßt, an dem die Liste des niedrigeren Knotens in der Speicherseite abgespeichert ist.

11. Rechnersystem nach Anspruch 10, bei dem in einer Speicherseite des nicht-flüchtigen Speichers (12, 14) nur komprimierte Listen gespeichert sind, deren Anzahl von Einträgen nach einer Kompression gleich groß sind.

12. Rechnersystem nach einem der Ansprüche 6 bis 11,

bei dem die für eine virtuelle Adresse durchlaufenen Listeneinträge Zugriffsrechteinformationen (EAR) für die virtuelle Adresse aufweisen, die identifizieren, ob oder wie in einem Betriebsmodus des Rechnersystems auf Daten an der physikalischen Adresse, die der virtuellen Adresse durch die Abbildungsvorschrift zugeordnet ist, zugegriffen werden darf, und

bei dem der TLB (18b) angeordnet ist, um eine Zugriffsrechteüberprüfung (48) auf der Basis der Zugriffsrechteinformationen durchzuführen, bevor eine physikalische Adresse ermittelt wird.

13. Rechnersystem nach einem der Ansprüche 6 bis 12,

bei dem die Hardware-Zustandsmaschine (18a) angeordnet ist, um in einen Knotenadressierungsmodus versetzt zu werden, und

bei dem die virtuelle Adresse ausgestaltet ist, um über einen Abschnitt (1212) derselben den Knoten zu identifizieren, auf dessen Liste zugegriffen werden soll.

14. Rechnersystem nach einem der vorhergehenden Ansprüche, das für sicherheitsrelevante Anwendungen bestimmt ist.

15. Rechnersystem nach einem der vorhergehenden Ansprüche, das auf einer Chipkarte integriert ist.

16. Rechnersystem nach einem der vorhergehenden Ansprüche, bei dem zumindest der Teil der in dem nicht-flüchtigen Speicher gespeicherten Abbildungsvorschrift in Form einer ROM-Maske bei der Herstellung des Rechnersystems einprogrammiert wird.

17. Verfahren zum Ermitteln einer physikalischen Adresse aus einer virtuellen Adresse, mit folgenden Schritten:

Erhalten einer virtuellen Adresse;

Zugreifen (26) auf einen nicht-flüchtigen Speicher (12, 14), in dem zumindest ein Teil einer Abbildungsvorschrift gespeichert ist, über die die virtuelle Adresse einer physikalischen Adresse eindeutig zugeordnet ist;

Ermitteln (18a) der physikalischen Adresse unter Verwendung zumindest des Teils der in dem nicht-flüchtigen Speicher gespeicherten Abbildungsvorschrift; und

Ausgeben der ermittelten physikalischen Adresse.

18. Verfahren nach Anspruch 17, bei dem die Abbildungsvorschrift hierarchisch als Baum mit Knoten (414, 418, 422, 426, 430, 434) strukturiert ist, wobei der Baum einen Wurzelknoten (414), zumindest einen Zwischenknoten (418, 422, 426, 430) und einen Endknoten (434) aufweist,

bei dem für einen Knoten eine Liste in dem nicht-flüchtigen Speicher gespeichert ist, wobei die Liste eines Knotens Einträge aufweist, wobei ein Eintrag einen Eintragsindex und einen Zeiger (424, 428, 432, 436) auf eine physikalische Adresse aufweist, wobei der Zeiger für die physikalische Adresse auf eine physikalische Adresse des nicht-flüchtigen Speichers zeigt, mit der eine Liste für einen anderen Knoten des Baums wiedergewonnen werden kann,

bei dem die virtuelle Adresse (400) in Abschnitte (402, 404, 406, 408, 410) gegliedert ist, wobei für den Wurzelknoten (414), den zumindest einen Zwischenknoten (418, 422, 426, 430) und den Endknoten (434) je ein Abschnitt vorhanden ist, wobei der Abschnitt (402, 404, 406, 408, 410) einen Listeneintrag einer zugehörigen Liste identifiziert, und

bei dem der Schritt des Ermittelns folgende Schritte aufweist:

Extrahieren eines Abschnitts (410) für den Wurzelknoten aus der virtuellen Adresse (400);

Zugreifen auf eine Liste (414), die dem extrahierten Abschnitt (410) der virtuellen Adresse zugeordnet ist, um einen Zeiger (416) auf eine physikalische Adresse einer Liste für einen Zwischenknoten zu erhalten;

Extrahieren eines Abschnitts (408) aus der virtuellen Adresse (400) für den Zwischenknoten;

Zugreifen auf einen Eintrag in die Liste unter Verwendung des extrahierten Abschnitts (408), um einen Zeiger (424) auf eine physikalische Adresse wiederzugewinnen, an der eine Liste (426) für einen niedrigeren Abschnitt gespeichert ist; und

Wiederholen der Schritte des Extrahierens und Zugreifens, bis sämtliche Abschnitte der virtuellen Adresse abgearbeitet sind, um unter Verwendung der Liste für den Endknoten und eines Abschnitts (402) der virtuellen Adresse für den Endknoten die physikalische Adresse (440) zu ermitteln, die der virtuellen Adresse (400) zugeordnet ist.

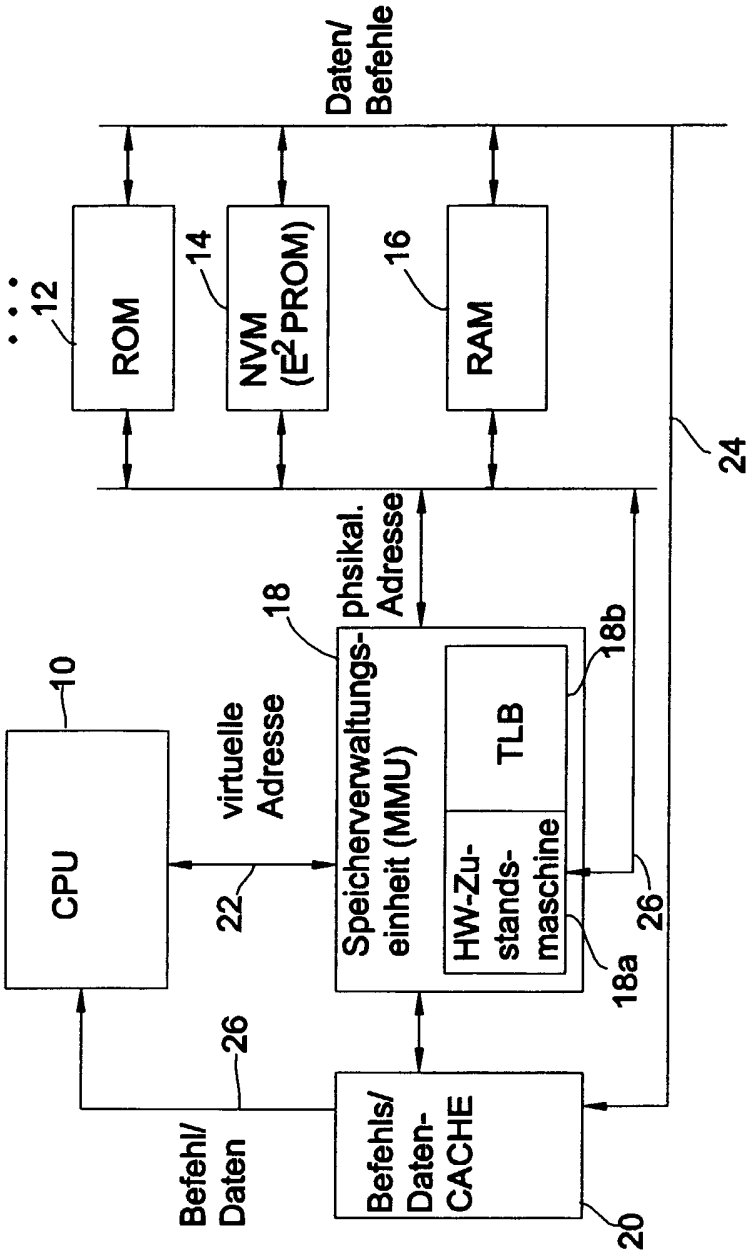

Daten/
Befehle
ROM
NVM
(E2 PROM)
RAM
12
14
16
24
phsikal.
Adresse
18
18b
10
virtuelle
Adresse
CPU
Speicherverwaltungs-
einheit (MMU)
TLB
HW-Zu-
stands-
maschine
22
26
18a
26
Befehl/
Daten
Befehls/
Daten-
CACHE
20

FIG 1

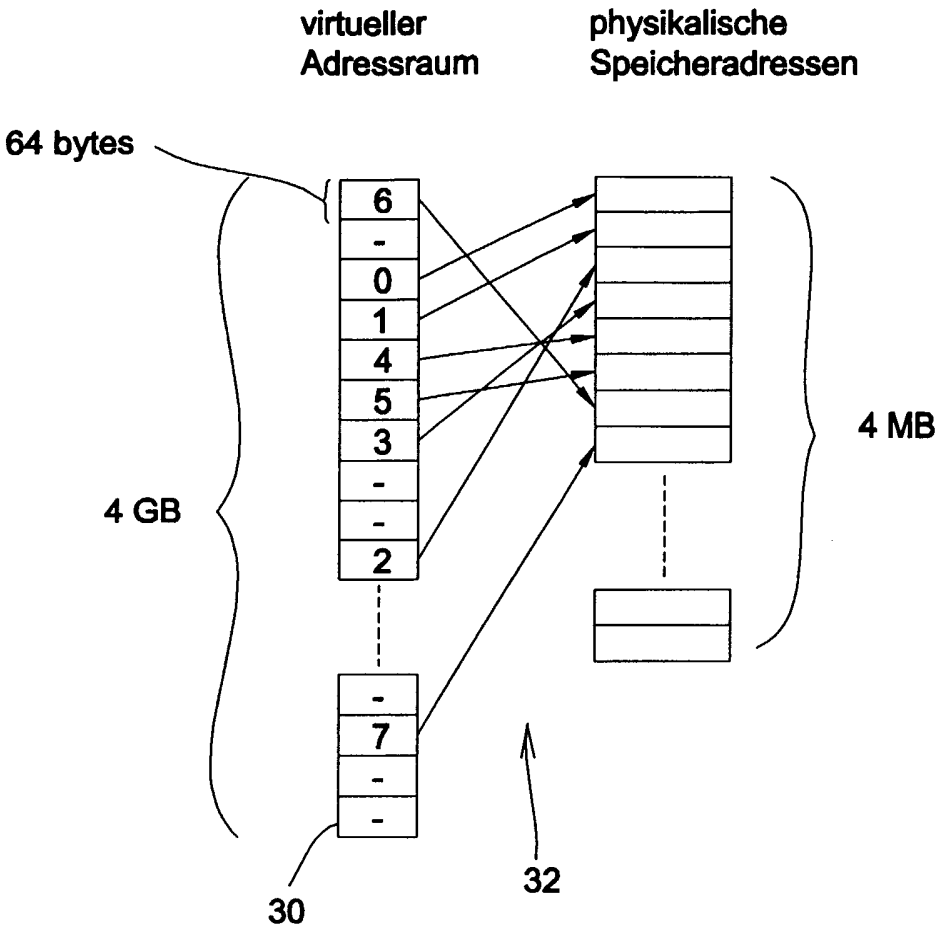

virtueller
Adressraum
physikalische
Speicheradressen
64 bytes
4 GB
4 MB
6
-
0
1
4
5
3
-
-
2
-
7
-
-
30
32

FIG 2

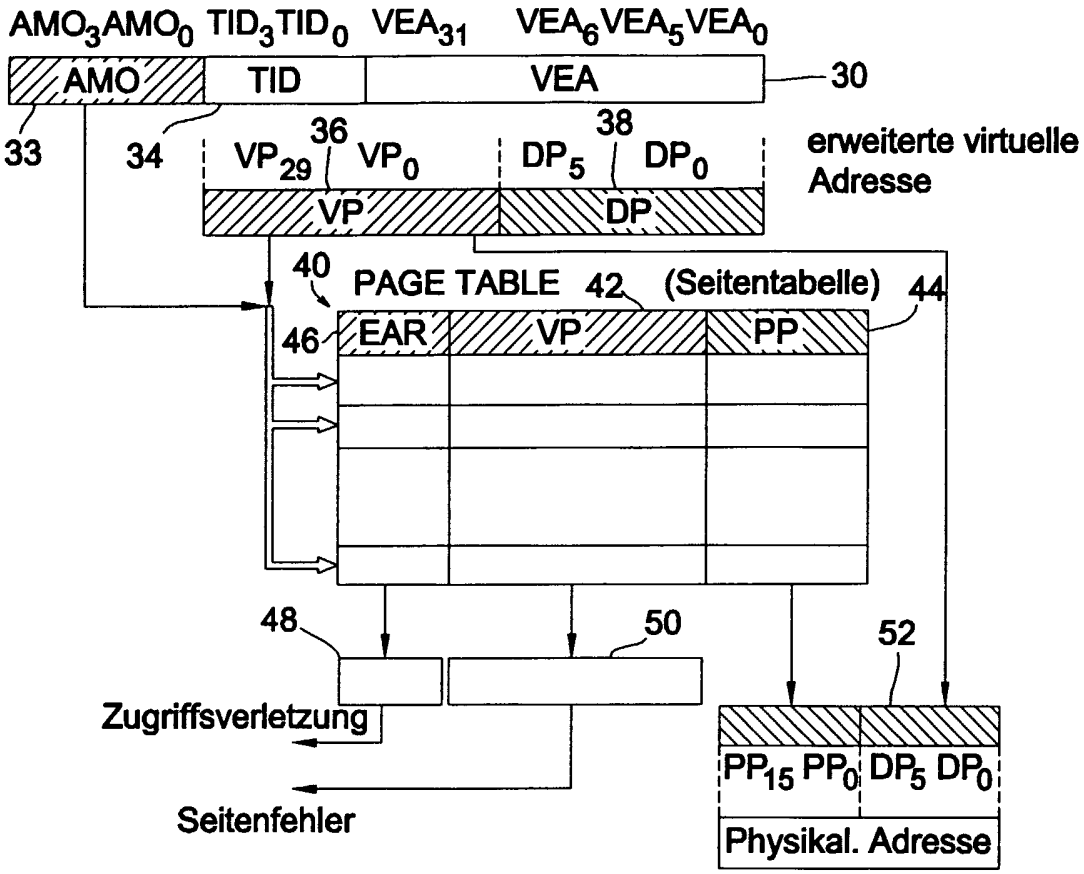

AMO3AMO0 TID3TID0 VEA31 VEA6VEA5VEA0
AMO
TID
VEA
30
33
34
36
38
VP29 VP0
DP5 DP0
erweiterte virtuelle Adresse
VP
DP
40
PAGE TABLE 42 (Seitentabelle)
44
46
EAR
VP
PP
48
50
52
Zugriffsverletzung
Seitenfehler
PP15 PP0 DP5 DP0
Physikal. Adresse

FIG 3

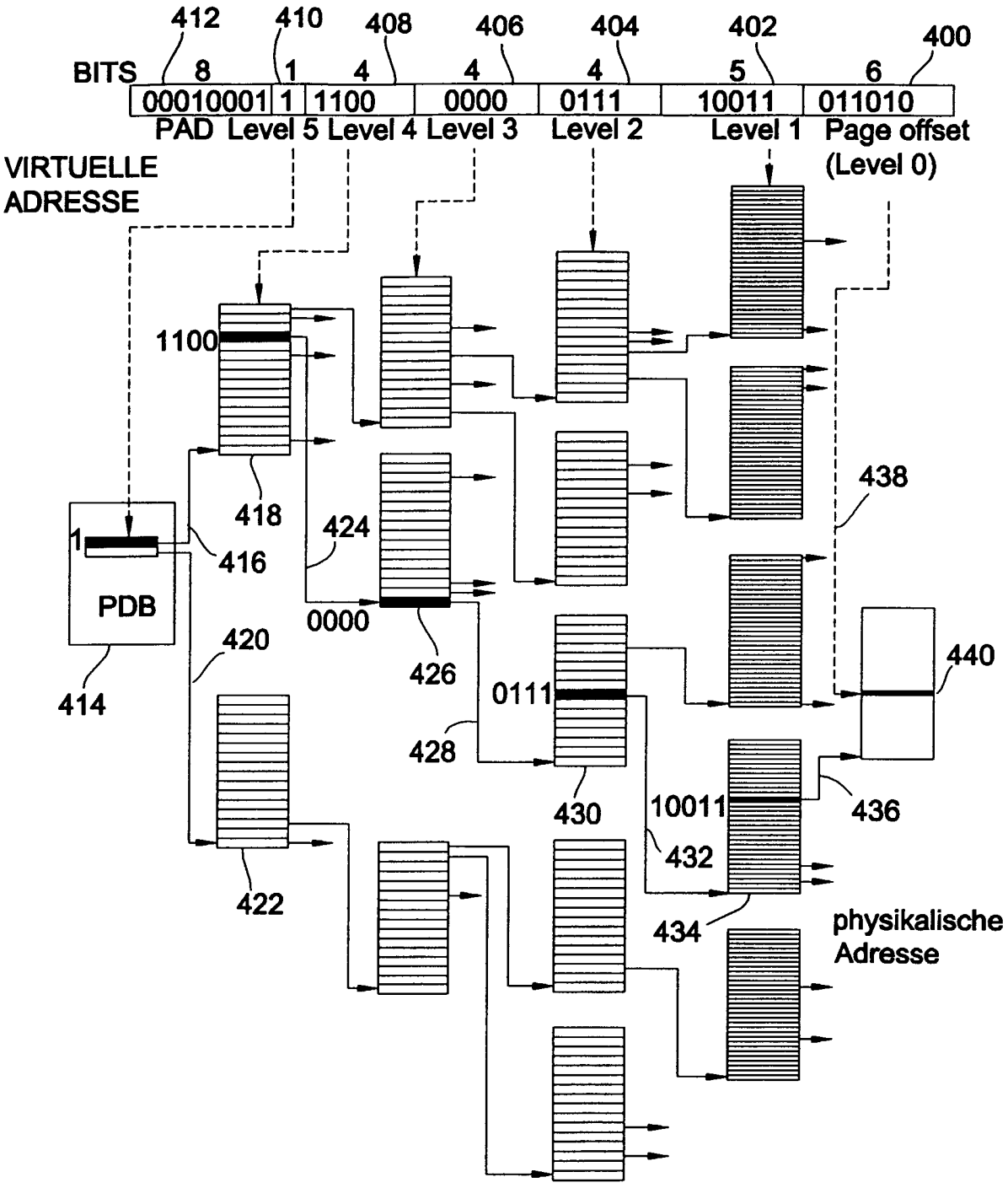

412
410
408
406
404
402
400
BITS
8
1
4
4
4
5
6
00010001
1
1100
0000
0111
10011
011010
PAD
Level 5
Level 4
Level 3
Level 2
Level 1
Page offset
(Level 0)
VIRTUELLE
ADRESSE
1100
418
1
416
PDB
414
424
0000
426
420
422
0111
428
430
438
440
10011
432
436
434
physikalische
Adresse

FIG 4A

Ebene	Ausschnitt	Adressbereich in Bytes
1	VEA_{10-6}	2K (2^{11})
2	VEA_{14-11}	32K (2^{15})
3	VEA_{18-15}	512K (2^{19})
4	VEA_{22-19}	8M (2^{23})
5 (root)	VEA_{23}	16M (2^{24})

FIG 4B

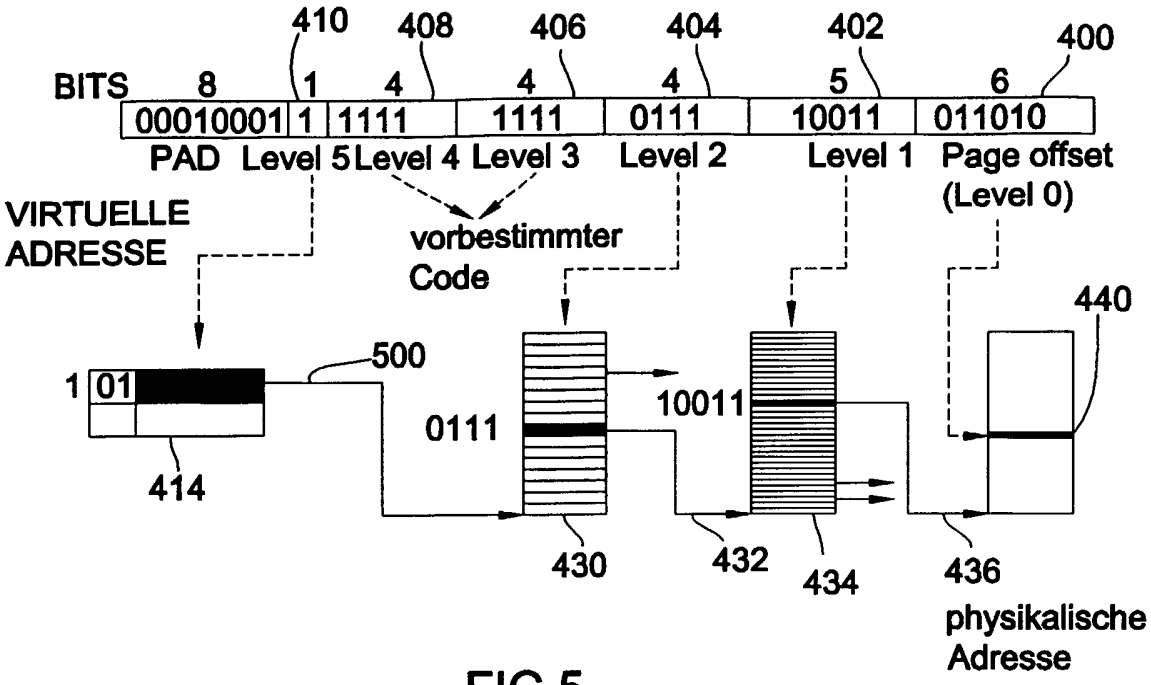


FIG 5

Ebene (Level)	Abschnitte der virtuellen Adresse in Bits	Knotengröße (max. Anzahl von Einträgen in die Knotenliste)
5	23	2
4	22 : 19	16
3	18 : 15	16
2	14 : 11	16
1	10 : 6	32
0	5 : 0	64

FIG 6

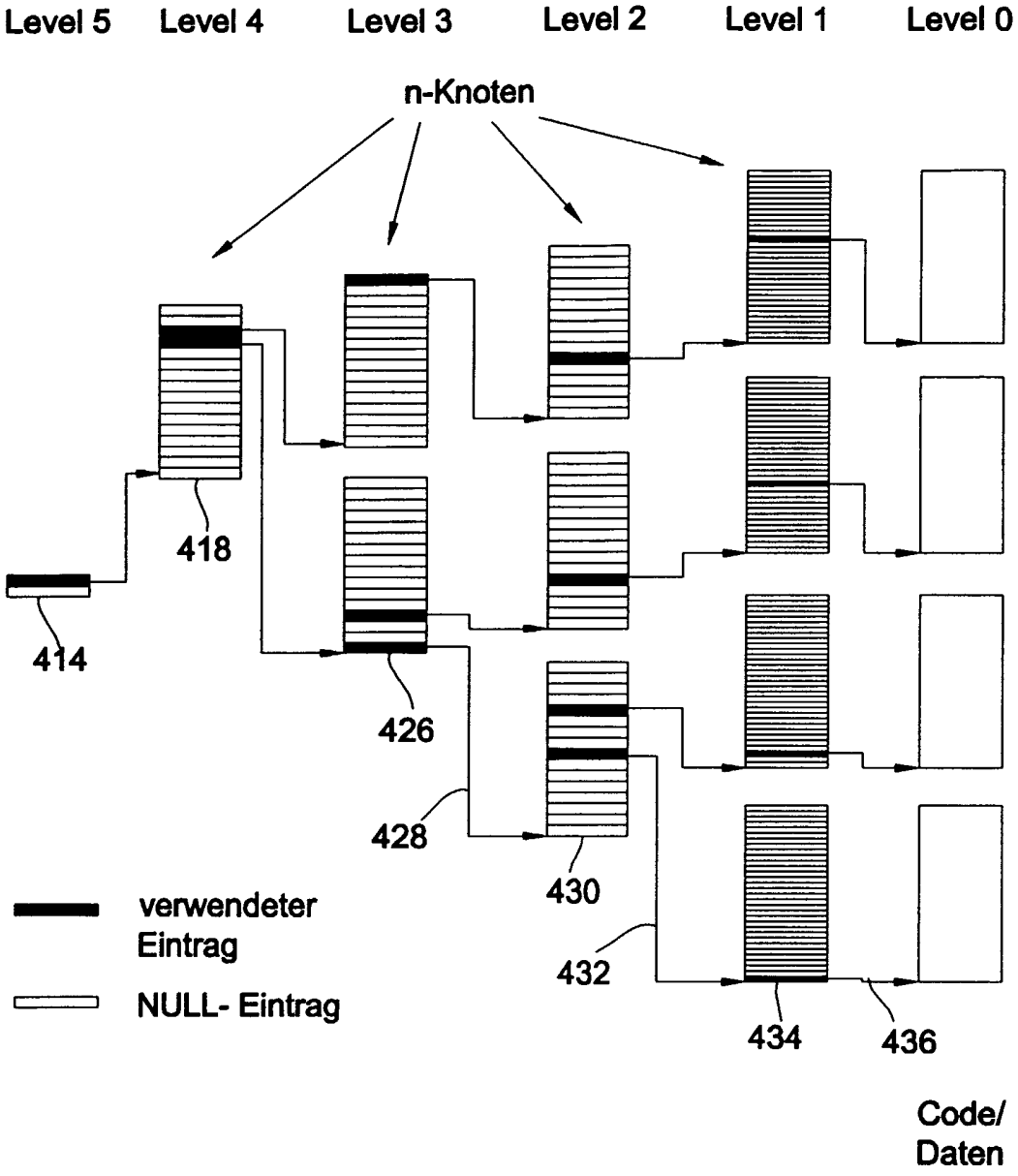

Level 5
Level 4
Level 3
Level 2
Level 1
Level 0
n-Knoten
418
414
426
428
430
432
434
436
verwendeter Eintrag
NULL- Eintrag
Code/ Daten

FIG 7

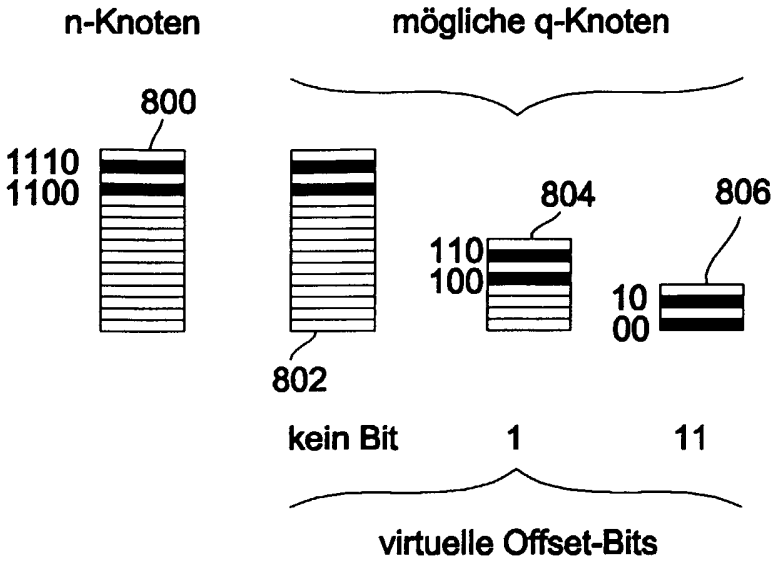

n-Knoten
mögliche q-Knoten
800
1110
1100
802
804
110
100
806
10
00
kein Bit
1
11
virtuelle Offset-Bits

FIG 8

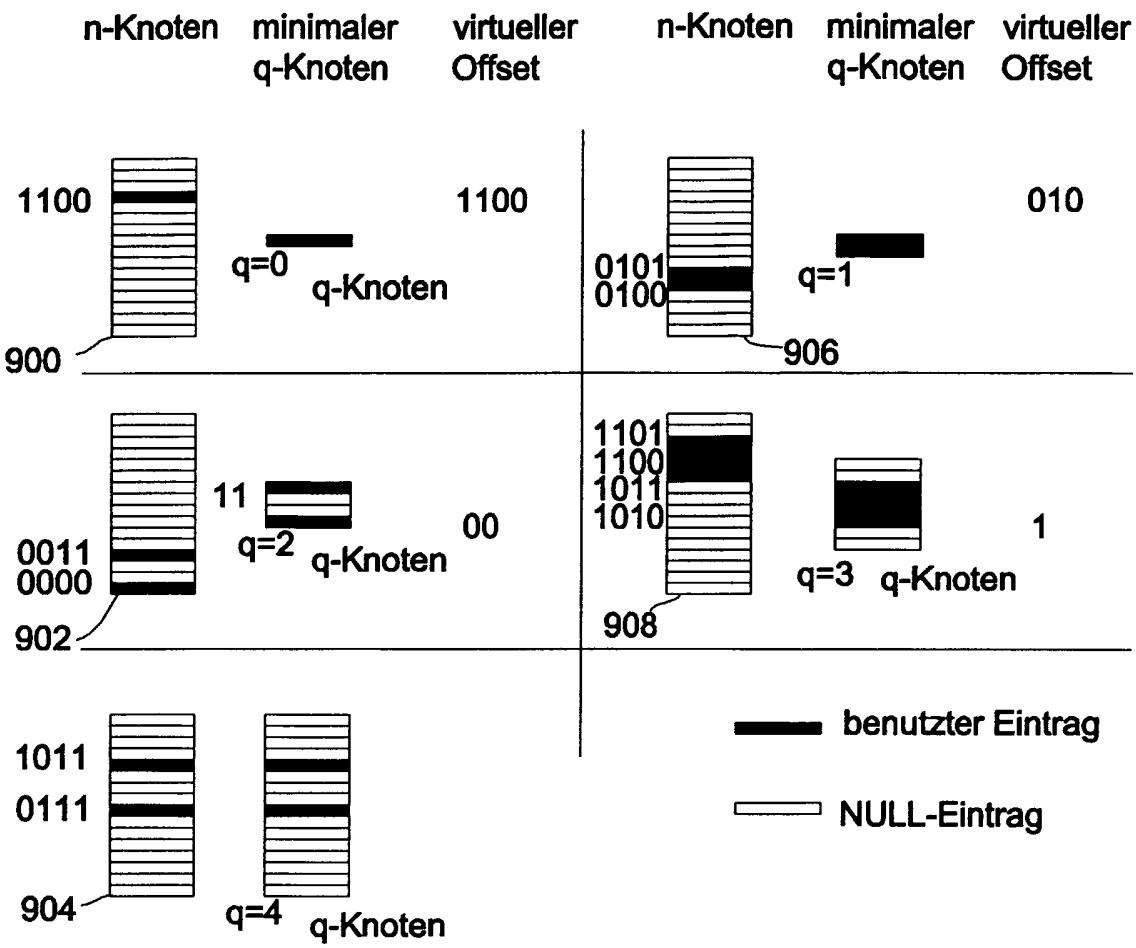

n-Knoten
minimaler q-Knoten
virtueller Offset
1100
1100
q=0
q-Knoten
900
0101
0100
010
q=1
906
0011
0000
11
q=2
q-Knoten
00
902
1101
1100
1011
1010
1
q=3
q-Knoten
908
1011
0111
904
q=4
q-Knoten
benutzter Eintrag
NULL-Eintrag

FIG 9

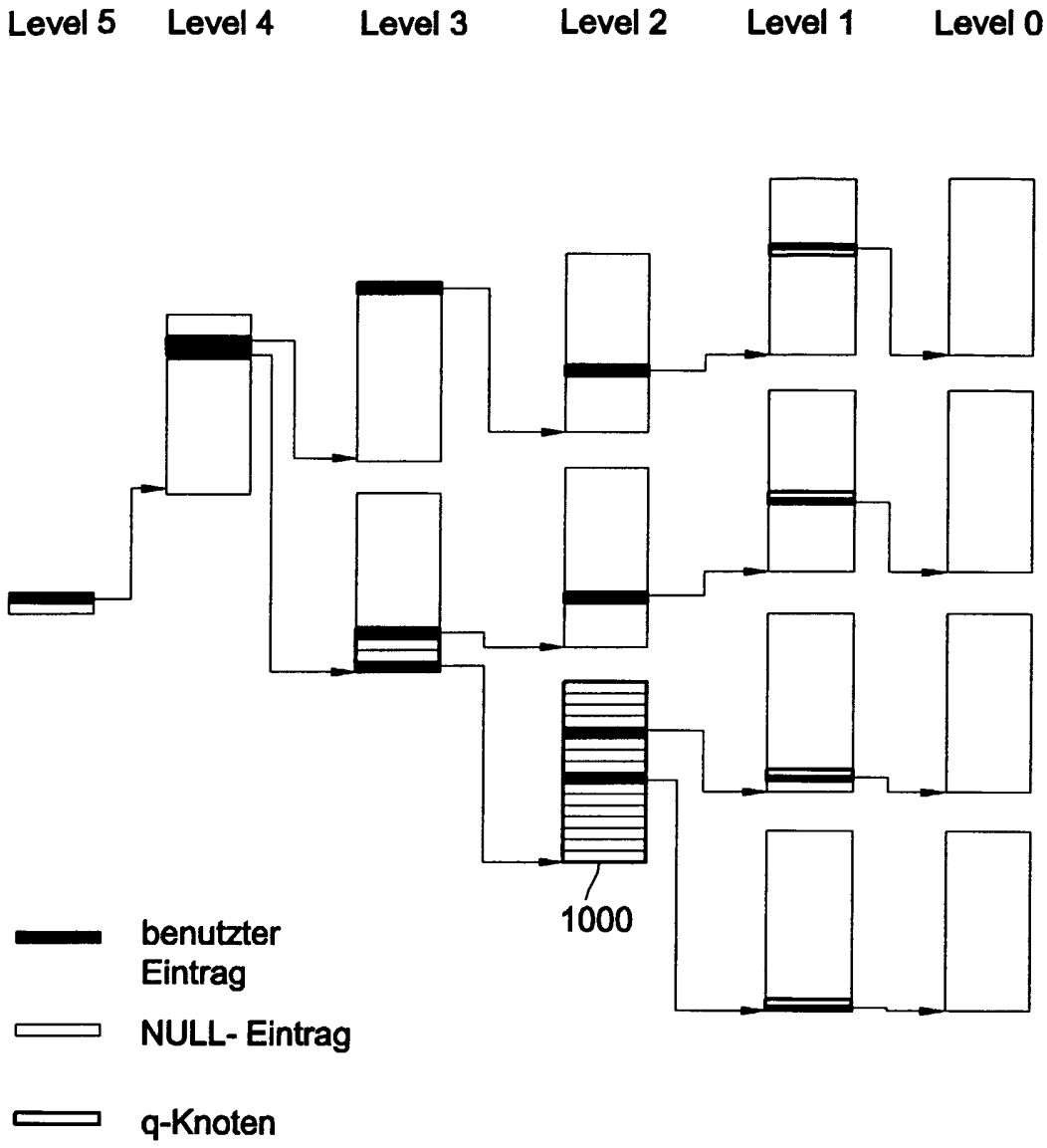

Level 5
Level 4
Level 3
Level 2
Level 1
Level 0
1000
benutzter Eintrag
NULL- Eintrag
q-Knoten

FIG 10

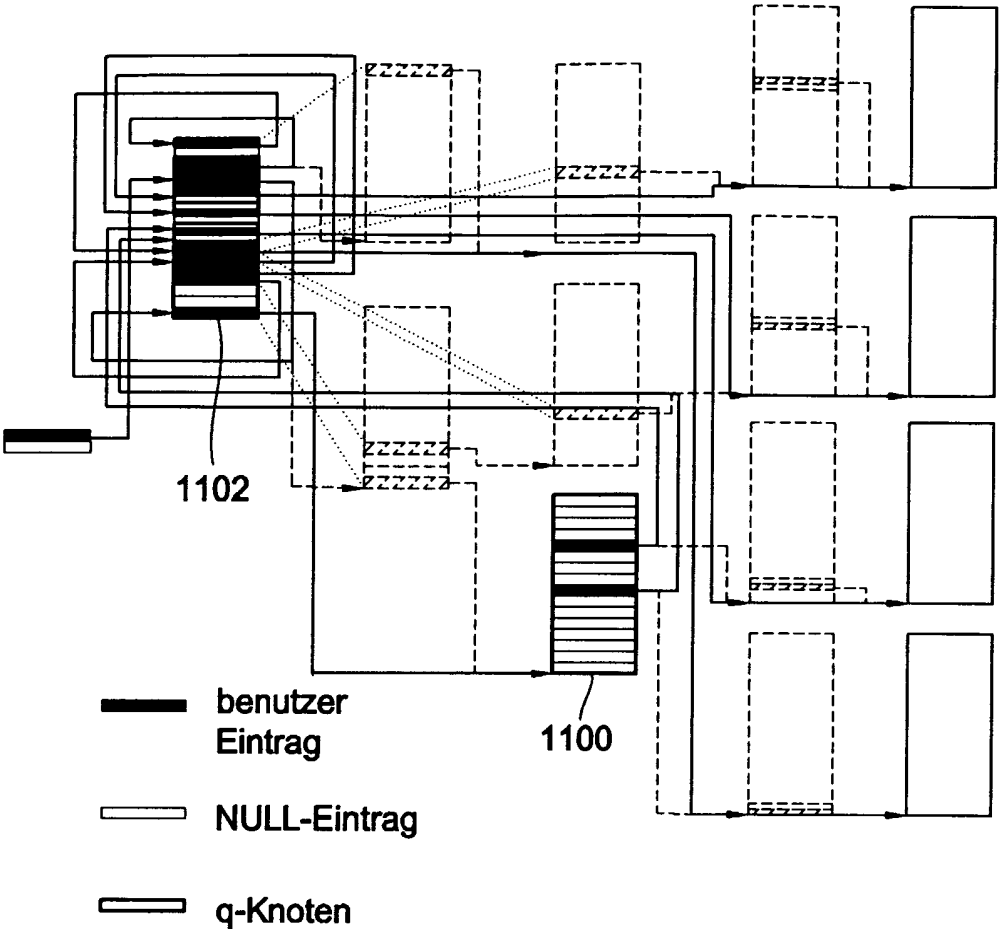


FIG 11

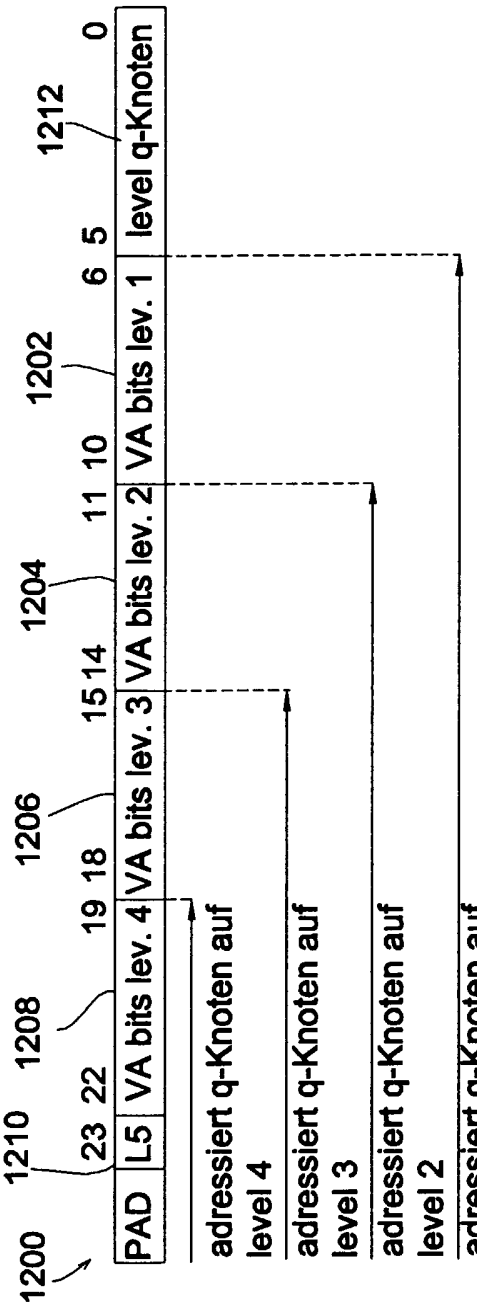

1200
1210
1208
1206
1204
1202
1212
23
22
19
18
15
14
11
10
6
5
0
PAD
L5
VA bits lev. 4
VA bits lev. 3
VA bits lev. 2
VA bits lev. 1
level q-Knoten
adressiert q-Knoten auf level 4
adressiert q-Knoten auf level 3
adressiert q-Knoten auf level 2
adressiert q-Knoten auf level 1

FIG 12

Bezugszeichenliste

10 CPU
12 ROM
14 NVM (E^2PROM)
16 RAM
18 Speicherverwaltungseinheit
18a Hardware-Zustandsmaschine
18b Translation Look Aside Buffer (TLB)
20 Befehls/Daten-Cache
24 Daten/Befehle-Bus
26 Hardware-Zustandsmaschine-Zugriffsbus
30 virtuelle Speicherseite
32 Abbildungsvorschrift
33 Zugriffsmodus-Bits
34 Aufgaben-Identifizierer-Bits
36 Adresse einer virtuellen Seite
38 Adressversatz einer virtuellen Seite
40 Zugriffsrechteinformationen
42 Spalte für eine virtuelle Adresse
44 Spalte für eine physikalische Adresse
46 Seitentabelle
48 Zugriffsüberprüfungsmodul
50 Seitenfehlermodul
52 physikalische Adresse
400 virtuelle Adresse
402 Abschnitt für Ebene 1
404 Abschnitt für Ebene 2
406 Abschnitt für Ebene 3
408 Abschnitt für Ebene 4
410 Abschnitt für Ebene 5
412 Paketadresse
414 Paketbeschreiberpuffer
416 Zeiger auf physikalische Adresse
418 Liste für Ebene 4
420 Zeiger auf physikalische Adresse
422 Liste für Ebene 4

424 Zeiger auf physikalische Adresse
426 Liste für Ebene 3
428 Zeiger auf physikalische Adresse
430 Liste für Ebene 2
432 Zeiger auf physikalische Adresse
434 Liste für Ebene 1
436 Zeiger auf physikalische Adresse
438 Seitenversatzwert
440 physikalische Adresse
500 Zeiger zum Überspringen einer oder mehrerer Ebenen
800 n-Knoten-Liste
802 nicht-komprimierte q-Knoten-Liste
804 einfach komprimierte q-Knoten-Liste
806 doppelt komprimierte q-Knoten-Liste
900 Beispiel für n-Knoten-Liste
902 Beispiel für n-Knoten-Liste
904 Beispiel für n-Knoten-Liste
906 Beispiel für n-Knoten-Liste
908 Beispiel für n-Knoten-Liste
1000 nicht-komprimierte q-Knoten-Liste
1100 Speicherseite mit einer einzigen nicht-komprimierten q-Knoten-Liste
1102 physikalische Speicherseite mit einer Mehrzahl von komprimierten q-Knoten-Listen
1200 virtuelle Adresse für den Knotenadressierungsmodus
1210 Abschnitt für Ebene 5
1208 Abschnitt für Ebene 4
1206 Abschnitt für Ebene 3
1204 Abschnitt für Ebene 2
1202 Abschnitt für Ebene 1
1212 Abschnitt zum Identifzieren der Liste, auf die zugegriffen werden soll

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP 02/06146

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F12/10

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 255 384 A (CHO JAMES Y ET AL) 19 October 1993 (1993-10-19) column 23, line 60 -column 24, line 15; figures 8,8A,8B,13 column 22, line 1-14 column 42, line 1-16	1-3,16, 17
X	column 24, line 3-15	4
X	column 26, line 6-12	4
X	column 1, line 65-68	4
X	column 3, line 48-57; figure 13	4
X	column 31, line 1 -column 33, line 2; figure 19	5-8,13, 18
X	column 23, line 16-30	12,14
X	column 31, line 29-31	12,14

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 September 2002	21/10/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Weber, R

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP 02/06146

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 123 101 A (SINDHU PRADEEP S) 16 June 1992 (1992-06-16) column 8, line 9-24; figure 2 column 11, line 28 -column 12, line 33 column 19, line 57 -column 20, line 13 column 27, line 54-64	1-5,16
Y	column 22, line 3-23	5-14
X	column 26, line 24-49	12,14
X	WU M ET AL: "ENVY: A NON-VOLATILE, MAIN MEMORY STORAGE SYSTEM" ACM SIGPLAN NOTICES, ASSOCIATION FOR COMPUTING MACHINERY, NEW YORK, US, vol. 29, no. 11, 1 November 1994 (1994-11-01), pages 86-97, XP000491727 ISSN: 0362-1340 page 94, column 1, paragraph 2; figure 11	1-3,5
X	US 6 069 638 A (PORTERFIELD A KENT) 30 May 2000 (2000-05-30) column 7, line 61 -column 8, line 11; figure 6A column 14, line 45 -column 15, line 6 column 8, line 50 -column 9, line 2	1-5
Y	WO 94 27222 A (LIEDTKE JOCHEN) 24 November 1994 (1994-11-24) page 13, last paragraph -page 15, paragraph 2; figures 1,2 page 17, last paragraph	5-14
X	BURNS D ET AL: "68020 coprocessors" ELECTRONICS & WIRELESS WORLD, JUNE 1987, UK, vol. 93, no. 1616, pages 614-616, XP002215219 ISSN: 0266-3244 page 614, column 1, last line -column 3, paragraph 1; figures 2,3	1-14,17, 18

INTERNATIONAL SEARCH REPORT

Information on patent family members

Internatio[redacted] Application No
PCT/EP 02/06146

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US 5255384	A	19-10-1993	CA	1272301 A1	31-07-1990
			CA	1283219 A2	16-04-1991
			CA	1283220 A2	16-04-1991
			DE	3650584 D1	23-01-1997
			DE	3650584 T2	26-06-1997
			EP	0196244 A2	01-10-1986
			EP	0732656 A2	18-09-1996
			JP	2083650 C	23-08-1996
			JP	7117913 B	18-12-1995
			JP	61275946 A	06-12-1986
			US	4899275 A	06-02-1990
			US	4860192 A	22-08-1989
US 5123101	A	16-06-1992	US	5230045 A	20-07-1993
US 6069638	A	30-05-2000	NONE		
WO 9427222	A	24-11-1994	DE	4405845 A1	17-11-1994
			WO	9427222 A1	24-11-1994
			US	5790979 A	04-08-1998

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP 02/06146

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
IPK 7 G06F12/10

Nach der Internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
IPK 7 G06F

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, PAJ

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	US 5 255 384 A (CHO JAMES Y ET AL) 19. Oktober 1993 (1993-10-19) Spalte 23, Zeile 60 -Spalte 24, Zeile 15; Abbildungen 8,8A,8B,13 Spalte 22, Zeile 1-14 Spalte 42, Zeile 1-16	1-3,16, 17
X	Spalte 24, Zeile 3-15	4
X	Spalte 26, Zeile 6-12	4
X	Spalte 1, Zeile 65-68	4
X	Spalte 3, Zeile 48-57; Abbildung 13	4
X	Spalte 31, Zeile 1 -Spalte 33, Zeile 2; Abbildung 19	5-8,13, 18
X	Spalte 23, Zeile 16-30	12,14
X	Spalte 31, Zeile 29-31	12,14

-/--

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen

[X] Siehe Anhang Patentfamilie

° Besondere Kategorien von angegebenen Veröffentlichungen :

"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden

"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
30. September 2002	21/10/2002

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Weber, R

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP 02/06146

C.(Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	US 5 123 101 A (SINDHU PRADEEP S) 16. Juni 1992 (1992-06-16) Spalte 8, Zeile 9-24; Abbildung 2 Spalte 11, Zeile 28 -Spalte 12, Zeile 33 Spalte 19, Zeile 57 -Spalte 20, Zeile 13 Spalte 27, Zeile 54-64	1-5,16
Y	Spalte 22, Zeile 3-23	5-14
X	Spalte 26, Zeile 24-49	12,14
X	WU M ET AL: "ENVY: A NON-VOLATILE, MAIN MEMORY STORAGE SYSTEM" ACM SIGPLAN NOTICES, ASSOCIATION FOR COMPUTING MACHINERY, NEW YORK, US, Bd. 29, Nr. 11, 1. November 1994 (1994-11-01), Seiten 86-97, XP000491727 ISSN: 0362-1340 Seite 94, Spalte 1, Absatz 2; Abbildung 11	1-3,5
X	US 6 069 638 A (PORTERFIELD A KENT) 30. Mai 2000 (2000-05-30) Spalte 7, Zeile 61 -Spalte 8, Zeile 11; Abbildung 6A Spalte 14, Zeile 45 -Spalte 15, Zeile 6 Spalte 8, Zeile 50 -Spalte 9, Zeile 2	1-5
Y	WO 94 27222 A (LIEDTKE JOCHEN) 24. November 1994 (1994-11-24) Seite 13, letzter Absatz -Seite 15, Absatz 2; Abbildungen 1,2 Seite 17, letzter Absatz	5-14
X	BURNS D ET AL: "68020 coprocessors" ELECTRONICS & WIRELESS WORLD, JUNE 1987, UK, Bd. 93, Nr. 1616, Seiten 614-616, XP002215219 ISSN: 0266-3244 Seite 614, Spalte 1, letzte Zeile -Spalte 3, Absatz 1; Abbildungen 2,3	1-14,17, 18

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/EP 02/06146

Im Recherchenbericht angeführtes Patentdokument		Datum der Veröffentlichung	Mitglied(er) der Patentfamilie		Datum der Veröffentlichung
US 5255384	A	19-10-1993	CA	1272301 A1	31-07-1990
			CA	1283219 A2	16-04-1991
			CA	1283220 A2	16-04-1991
			DE	3650584 D1	23-01-1997
			DE	3650584 T2	26-06-1997
			EP	0196244 A2	01-10-1986
			EP	0732656 A2	18-09-1996
			JP	2083650 C	23-08-1996
			JP	7117913 B	18-12-1995
			JP	61275946 A	06-12-1986
			US	4899275 A	06-02-1990
			US	4860192 A	22-08-1989
US 5123101	A	16-06-1992	US	5230045 A	20-07-1993
US 6069638	A	30-05-2000	KEINE		
WO 9427222	A	24-11-1994	DE	4405845 A1	17-11-1994
			WO	9427222 A1	24-11-1994
			US	5790979 A	04-08-1998