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

(10) **DE 602 21 030 T2** 2008.03.20

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 430 658 B1**
(21) Deutsches Aktenzeichen: **602 21 030.5**
(86) PCT-Aktenzeichen: **PCT/US02/29074**
(96) Europäisches Aktenzeichen: **02 780 304.8**
(87) PCT-Veröffentlichungs-Nr.: **WO 2003/030461**
(86) PCT-Anmeldetag: **12.09.2002**
(87) Veröffentlichungstag der PCT-Anmeldung: **10.04.2003**
(97) Erstveröffentlichung durch das EPA: **23.06.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **04.07.2007**
(47) Veröffentlichungstag im Patentblatt: **20.03.2008**

(51) Int Cl.[8]: ***H04L 12/46*** (2006.01)

(30) Unionspriorität:
966349 **28.09.2001** **US**

(84) Benannte Vertragsstaaten:
AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE, SK, TR

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

(74) Vertreter:
Rummler, F., Dipl.-Ing.Univ., Pat.-Anw., 80802 München

(72) Erfinder:
HOOPER, Donald, Shrewsbury, MA 01545, US;
HIRNAK, Stephanie, Bedford, NH 03110, US

(54) Bezeichnung: **VERFAHREN, EINRICHTUNG UND RECHNERPROGRAMM FÜR DIE ENTKAPSELUNG UND VERKAPSELUNG VON PAKETEN MIT MEHREREN KOPFFELDERN**

Beschreibung

ALLGEMEINER STAND DER TECHNIK

[0001] Diese Erfindung betrifft das Weiterleiten von Netzwerkpaketen zwischen Netzwerkdomänen.

[0002] Pakete werden durch eine Reihe von Routervorrichtungen geleitet, von denen jede Pakete auf ihrem Weg von einer Quelle zu einem Ziel speichert und weiterleitet. Zum Beispiel kann ein Paket am Anfang ein Internetpaket sein, über einen ATM (asynchronen Übertragungsmoduspfad) weitergeleitet und dann zurück zum Ethernet auf ein gemeinsames Netzwerk zu seinem zugedachten Endempfänger geleitet werden. Während das Netzwerk durch diese Netzwerkdomänen geht, können verschiedene Kopffeldverkapselungen zu dem Paket hinzugefügt oder davon entfernt werden. Manche Verbindungen benutzen Punkt-zu-Punkt-Protokolle (PPP), wohingegen andere das Multiprotocol Label Switching MPLS, das Layer-to-Tunneling-Protokoll LTTP, ATM und so fort benutzen.

[0003] WO 98/20647 beschreibt eine Anordnung, bei der Mehrparteienkommunikationen in einer digitalen Vorrichtung gemäß einer Datenstruktur gehandhabt werden, bei der ein formatiertes Paket von einem bestimmten Sender in ein gemeinsames Format M umgesetzt wird, das dann erneut in verschiedene Ziele umgesetzt wird. Mit dieser Vorrichtung wird die Handhabung von Gruppenadresspaketen mit kundenspezifischer Schaltung erreicht und Flexibilität erreicht, indem Parameter der Wegeermittlungsroutine im programmierbar veränderbaren Speicher spezifiziert werden können. In einer Variation der Vorrichtung ist die Darstellung einer Gruppe eine kreisförmig verbundene Liste. Jede Partei hat einen Eintrag in der verbundenen Liste, weshalb die gesamte Struktur N untergeordnete Strukturen für N Teilnehmer hat.

[0004] US-Patentschrift Nr. 5,251,205 betrifft ein Verfahren zum Verbinden eines Netzwerks, so dass TCP/IP- und ISO-8473-Pakete in der gleichen Domäne gesendet werden können. Die Unabhängigkeit der Adressen wird bewahrt: eine Vorrichtung in dem Netzwerk kann nur einer TCP/IP-Adresse zugeordnet sein und andere Vorrichtungen können nur einer ISO-8473-Adresse zugeordnet sein. Darüber hinaus teilen bei solchen Anordnungen alle Router Verbindungszustandsinformation durch Verwenden eines gemeinsamen Verbindungszustandspaketformats (wie dem ISO-10589-Format); auf diese Weise können Wege durch das Netzwerk ungeachtet der Protokolle, die von den Routern entlang des Wegs unterstützt werden, berechnet werden. Gegebenenfalls werden Pakete verkapselt und durch Router weitergeleitet, die in dem Protokoll des Pakets nicht kompetent sind.

[0005] WO 01/17179 beschreibt eine Netzwerkschaltvorrichtung, Komponenten für solch eine Vorrichtung und Verfahren zum Betreiben solch einer Vorrichtung, in denen die Datenflusshandhabung und Flexibilität durch die Kooperation mehrerer Schnittstellenprozessoren und einer Reihe peripherer Elemente, die auf einem Halbleitersubstrat ausgebildet sind, verbessert werden. Mit dieser Konfiguration bilden die Schnittstellenprozessoren und peripheren Elemente zusammen einen Netzwerkprozessor, der dazu in der Lage ist, mit anderen Elementen, einschließlich einer optionalen Schaltstrukturvorrichtung, bei der Ausführung von Befehlen, die den Datenfluss in dem Netzwerk lenken, zu kooperieren.

[0006] US-Patentschrift Nr. 5,651,002 betrifft eine Vernetzungsvorrichtung, die eine Paketkopffeldumsetzung zum Umsetzen des Formats eines Kopffeldes, das mit einem Quellennetzwerk verbunden ist, in ein Kopffeldformat, das mit einem Zielnetzwerk eines anderen Typs als dem Quellennetzwerk verbunden ist, bereitstellt. Solch eine Vorrichtung weist einen Speicher zum Speichern eines Adressabschnitts eines empfangenen Paketkopffeldes in einem ersten Block von Pufferstellen und des Restes des empfangenen Pakets in einem zweiten Block von Pufferstellen, die von dem ersten Block von Pufferstellen durch eine Leerstelle beabstandet sind, auf.

[0007] Die Leerstelle ermöglicht, dass zusätzliche Kopffeldinformation in den Puffer geschrieben werden kann, wenn dies zur Umsetzung notwenig ist. Die Vorrichtung weist ferner einen geteilten Speicher auf, der einen SRAM-Abschnitt und einen DRAM-Abschnitt aufweist, die jedem Puffer zugewiesen sind, wobei sich mindestens der Adressabschnitt eines empfangenen Pakets und die Leerstelle in dem SRAM-Abschnitt befinden.

[0008] Man wird zu schätzen wissen, dass ein Bedarf an einem generischen Verfahren besteht, das in einem Router umgesetzt werden kann, der eine Vielzahl von Protokolltypen handhaben kann, die keinen übermäßigen Coderaum verbrauchen.

KURZDARSTELLUNG DER ERFINDUNG

[0009] Gemäß der Erfindung wird ein Verfahren gemäß Anspruch 1, ein Computerprogrammprodukt gemäß Anspruch 19 und eine Vorrichtung gemäß Anspruch 28 bereitgestellt.

BESCHREIBUNG DER ZEICHNUNGEN

[0010] Fig. 1 ist ein Blockdiagramm eines beispielhaften Kommunikationssystems, das einen auf Hardware basierenden mehrfädigen Prozessor einsetzt, der zum Verständnis und zur Umsetzung der Erfindung nützlich ist.

[0011] Fig. 2-1 bis Fig. 2-4 sind ein detailliertes beispielhaftes Blockdiagramm eines auf Hardware basierenden mehrfädigen Prozessors aus Fig. 1, der zum Verständnis und zur Umsetzung der Erfindung nützlich ist.

[0012] Fig. 3 ist ein Blockdiagramm, das eine funktionelle Anordnung des mehrfädigen Prozessors aus **Fig.** 2 darstellt.

[0013] Fig. 4 ist ein Blockdiagramm, das Datenstrukturen im Speicher darstellt, die für den Prozessor aus Fig. 1 benutzt werden.

[0014] Fig. 5 ist ein Blockdiagramm, das Formate zum Weiterleiten von Tabellen darstellt, die in den Tabellen aus Fig. 4 benutzt werden.

[0015] Fig. 6 ist ein Flussdiagramm, das einen generischen Paketweiterleitungsprozess darstellt.

[0016] Fig. 7 ist ein Flussdiagramm, das einen alternativen Aspekt des Paketweiterleitungsprozesses aus Fig. 6 darstellt.

AUSFÜHRLICHE BESCHREIBUNG

[0017] Mit Bezug auf Fig. 1 weist ein Kommunikationssystem **10**, das für das Verständnis und zur Umsetzung der Erfindung nützlich ist, einen parallelen, auf Hardware basierenden mehrfädigen Prozessor **12** auf. Der auf Hardware basierende mehrfädige Prozessor **12** ist mit einem Bus wie einem PCI-Bus **14**, einem Speichersystem **16** und einem zweiten Bus **18** verbunden. Das System **10** ist für Aufgaben, die in parallele untergeordnete Aufgaben oder Funktionen aufgeteilt werden können, besonders nützlich. Der auf Hardware basierende mehrfädige Prozessor **12** ist insbesondere für Aufgaben geeignet, die bandbreitenorientiert und nicht latenzorientiert sind. Der auf Hardware basierende mehrfädige Prozessor **12** weist eine Vielzahl von Mikromaschinen **22** auf, die jeweils gleichzeitig aktiv und auf einer Vielzahl von hardwaregesteuerten Fäden arbeiten können, die unabhängig an einer Aufgabe arbeiten.

[0018] Der auf Hardware basierende mehrfädige Prozessor **12** weist auch eine zentrale Steuerung **20** auf, die das Laden der Mikrocodesteuerung für andere Ressourcen des auf Hardware basierenden mehrfädigen Prozessors **12** unterstützt und andere Funktionen des Universalrechnertyps wie die Handhabung von Protokollen, Ausnahmen, zusätzliche Unterstützung für die Paketverarbeitung ausführt, wobei die Mikromaschinen die Pakete für eine detailliertere Verarbeitung wie bei Grenzbedingungen weitergeben. In einer Ausführungsform ist der Prozessor **20** eine auf Strong Arm® basierende Architektur. Der Universalmikroprozessor **20** hat ein Betriebssystem. Durch das Betriebssystem kann der Prozessor **20** Funktionen aufrufen, um auf den Mikromaschinen **22a** bis **22f** zu laufen. Der Prozessor **20** kann jedes beliebige unterstützte Betriebssystem, vorzugsweise ein Echtzeitbetriebssystem wie das Echtzeitbetriebssystem MicrosoftNT, VXWorks benutzen.

[0019] Der auf Hardware basierende mehrfädige Prozessor **12** weist auch mehrere Funktionsmikromaschinen **22a** bis **22f** auf. Die funktionellen Mikromaschinen (Mikromaschinen) **22a** bis **22f** verwalten mehrere Programmzähler in Hardware und Zustände, die mit den Programmzählern verbunden sind. Tatsächlich können mehrere Sätze von Fäden auf jeder der Mikromaschinen **22a** bis **22f** gleichzeitig aktiv sein, während jeweils nur eine zu einem gegebenen Zeitpunkt tatsächlich in Betrieb ist.

[0020] In einer Ausführungsform gibt es sechs Mikromaschinen **22a** bis **22f**, wie dargestellt. Jede Mikromaschine **22a** bis **22f** hat Leistungsfähigkeiten zum Verarbeiten von vier Hardwarefäden. Die sechs Mikromaschinen **22a** bis **22f** arbeiten mit gemeinsam benutzten Ressourcen, einschließlich des Speichersystems **16** und der Busschnittstellen **24** und **28**. Das Speichersystem **16** weist eine synchrone dynamische wahlfreie Zugriffsspeicher (Synchronous Dynamic Random Access Memory = SDRAM)-Steuerung **26a** und eine statische wahlfreie Zugriffsspeicher (Static Random Access Memory = SRAM)-Steuerung **26b** auf. Der SDRAM-Speicher **16a** und die SDRAM-Steuerung **26a** werden in der Regel zum Verarbeiten großer Datenmengen, zum Beispiel zum Verarbeiten von Netzwerknutzdaten aus Netzwerkpaketen benutzt. Die SRAM-Steuerung **26b** und der SRAM-Speicher **16b** werden in einer Vernetzungsanwendung für Aufgaben mit geringer Latenz und schnellem Zugriff, zum Beispiel zum Zugreifen auf Nachschlagtabellen, Speicher für den Kernprozessor **20** und so fort benutzt.

[0021] Die sechs Mikromaschinen **22a** bis **22f** greifen basierend auf den Eigenschaften der Daten entweder auf den SDRAM **16a** oder SRAM **16b** zu. Folglich werden Daten mit geringer Latenz und geringer Bandbreite in dem SRAM gespeichert und daraus abgerufen, wohingegen Daten mit höherer Bandbreite, für welche die Latenz nicht so bedeutend ist, in dem SDRAM gespeichert und daraus abgerufen werden. Die Mikromaschinen **22a** bis **22f** können Speicherreferenzbefehle entweder für die SDRAM-Steuerung **26a** oder für die SRAM-Steuerung **16b** ausführen.

[0022] Vorteile der Hardwaremehrfädigkeit können durch SRAM- oder SDRAM-Speicherzugriffe erläutert werden. Als ein Beispiel bewirkt ein SRAM-Zugriff, der von einem Faden_0 angefordert wird, von einer Mikromaschine, dass die SRAM-Steuerung **26b** einen Zugriff auf den SRAM-Speicher **16b** initiiert. Die SRAM-Steuerung steuert die Arbitration für den SRAM-Bus, greift auf den SRAM **16b** zu, ruft die

Daten aus dem SRAM **16b** ab und sendet Daten zurück an eine anfordernde Mikromaschine **22a** bis **22b**. Wenn die Mikromaschine, zum Beispiel **22a**, während eines SRAM-Zugriffs nur einen einzigen Faden hätte, der arbeiten könnte, wäre diese Mikromaschine inaktiv, bis Daten aus dem SRAM zurückgesendet werden. Durch den Einsatz von Hardwarekontextauslagerung innerhalb jeder der Mikromaschinen **22a** bis **22f** bewirkt die Hardwarekontextauslagerung, dass andere Kontexte mit einzigartigen Programmzählern in der gleichen Mikromaschine ausgeführt werden. Folglich kann ein anderer Faden, zum Beispiel Faden_1, arbeiten, während der erste Faden, zum Beispiel Faden_0, auf die Rücksendung der Lesedaten wartet. Während der Ausführung kann Faden_1 auf den SDRAM-Speicher **16a** zugreifen. Während Faden_1 auf der SDRAM-Einheit arbeitet und Faden_0 auf der SRAM-Einheit arbeitet, kann nun ein neuer Faden, zum Beispiel Faden_2, in der Mikromaschine **22a** arbeiten. Faden_2 kann für eine bestimmte Zeit arbeiten, bis er auf den Speicher zugreifen oder einen anderen Vorgang mit langer Latenz wie den Zugriff auf eine Busschnittstelle ausführen muss. Folglich kann der Prozessor **12** gleichzeitig einen Busvorgang, SRAM-Vorgang und SDRAM-Vorgang, die alle von einer Mikromaschine **22a** ausgeführt oder bearbeitet werden, und einen weiteren Faden aufweisen, der zur Verarbeitung von mehr Arbeit in dem Datenweg verfügbar ist.

[0023] Die Hardwarekontextauslagerung synchronisiert auch die Vollendung von Aufgaben. Zum Beispiel könnten zwei Fäden auf die gleiche gemeinsame Ressource, zum Beispiel SRAM, treffen. Jede dieser getrennten funktionellen Einheiten, zum Beispiel die FBUS-Schnittstelle **28**, die SRAM-Steuerung **26a** und die SDRAM-Steuerung **26b**, melden ein Flag zurück, das die Vollendung eines Vorgangs signalisiert, wenn sie eine angeforderte Aufgabe von einer der Mikromaschinen-Fadenkontexte vollenden. Wenn das Flag von der Mikromaschine empfangen wird, kann die Mikromaschine bestimmen, welcher Faden aktiviert wird.

[0024] Ein Beispiel einer Anwendung für den auf Hardware basierenden mehrfädigen Prozessor **12** ist ein Netzwerkprozessor. Als ein Netzwerkprozessor ist der auf Hardware basierende mehrfädige Prozessor **12** mit Netzwerkvorrichtungen wie einer Medienzugriffssteuervorrichtung, zum Beispiel einer Vorrichtung 10/100BaseT Octal MAC **13a** oder einer Gigabit-Ethernet-Vorrichtung **13b** verbunden. Im Allgemeinen kann der Netzwerkprozess mit jeder beliebigen Art von Kommunikationsvorrichtung oder Schnittstelle verbunden sein, die große Datenmengen empfängt/sendet. Das Kommunikationssystem **10**, das in einer Vernetzungsanwendung funktioniert, könnte mehrere Netzwerkpakete von den Vorrichtungen **13a**, **13b** empfangen und diese Pakete parallel verarbeiten. Mit dem auf Hardware basierenden mehrfädigen Prozessor **12** kann jedes Netzwerkpaket unabhängig verarbeitet werden. Ein anderes Beispiel zur Verwendung des Prozessors **12** ist eine Druckmaschine für einen Postscript-Prozessor oder ein Prozessor für ein untergeordnetes Speichersystem, das heißt, einen RAID-Plattenspeicher. Eine weitere Verwendung ist eine Anpassungsmaschine. In der Sicherheitsbranche zum Beispiel erfordert das Aufkommen des elektronischen Handels die Verwendung elektronischer Anpassungsmaschinen, um die Bestellungen zwischen Käufern und Verkäufern anzupassen. Diese und andere parallele Aufgabentypen können auf dem System **10** erreicht werden.

[0025] Der Prozessor **12** weist eine Busschnittstelle **28** auf, die den Prozessor mit dem zweiten Bus **18** verbindet. Die Busschnittstelle **28** verbindet den Prozessor **12** in einer Ausführungsform mit dem so genannten FBUS **18** (FIFO-Bus). Die FBUS-Schnittstelle **28** ist für die Steuerung und Verbindung des Prozessors **12** mit dem FBUS **18** verantwortlich. Der FBUS **18** ist ein 64 Bit breiter FIFO-Bus, der zurzeit als der beste Bus für Medienzugriffssteuerungs-(MAC) Vorrichtungen an Akzeptanz gewinnt.

[0026] Der Prozessor **12** weist eine zweite Schnittstelle, zum Beispiel eine PCI-Busschnittstelle **24** auf, die andere Systemkomponenten, die sich auf dem PCI **14**-Bus befinden, mit dem Prozessor **12** verbindet. Die PCI-Busschnittstelle **24** stellt dem Speicher **16**, zum Beispiel dem SDRAM-Speicher **16a**, einen Hochgeschwindigkeitsdatenweg **24a** bereit. Durch diesen Weg können Daten durch den PCI-Bus **14** über Direktspeicherzugriffs (DMA)-Übertragungen schnell aus dem SDRAM **16a** bewegt werden. Außerdem unterstützt die PCI-Busschnittstelle **24** Ziel- und Master-Vorgänge. Zielvorgänge sind Vorgänge, bei denen Slave-Vorrichtungen auf dem Bus **14** durch Lese- und Schreibvorgänge, die als ein Slave-zu-Ziel-Vorgang bedient werden, auf SDRAMs zugreifen. Bei Master-Vorgängen sendet der Prozessorkern **20** Daten direkt an die PCI-Schnittstelle **24** oder empfängt sie direkt von dieser.

[0027] Jede der funktionellen Einheiten ist mit einem oder mehreren internen Bussen verbunden. Der Prozessor weist einen AMBA-Bus auf, der den Prozessorkern **20** mit der Speichersteuerung **26a**, **26c** und mit einem AMBA-Umsetzer **30** verbindet, der unten beschrieben wird. Der Prozessor weist auch einen privaten Bus **34** auf, der die Mikromaschineneinheiten mit der SRAM-Steuerung **26b**, dem AMBA-Umsetzer **30** und der FBUS-Schnittstelle **28** verbindet. Ein Speicherbus **38** verbindet die Speichersteuerung **26a**, **26b** mit den Busschnittstellen **24** und **28** und dem Speichersystem **16**, das den Flash-ROM **16c** aufweist, der für Hochfahrvorgänge und so fort benutzt wird.

[0028] Mit Bezug auf Fig. 2-1 bis Fig. 2-4 ist ein de-

tailliertes beispielhaftes Blockdiagramm eines auf Hardware basierenden mehrfädigen Prozessors dargestellt, der zum Verständnis und zur Umsetzung der Erfindung nützlich ist, wobei jede der Mikromaschinen **22a** bis **22f** einen Arbiter aufweist, der Flags untersucht, um die verfügbaren zu verarbeitenden Fäden zu bestimmen. Jeder beliebige Faden von jeder beliebigen der Mikromaschinen **22a** bis **22f** kann auf die SDRAM-Steuerung **26a**, SDRAM-Steuerung **26b** oder FBUS-Schnittstelle **28** zugreifen. Die Speichersteuerungen **26a** und **26b** weisen jeweils mehrere Warteschlangen auf, um ausstehende Speicherreferenzanforderungen zu speichern. Die Warteschlangen behalten entweder die Reihenfolge der Speicherreferenzen bei oder ordnen Speicherreferenzen an, um die Speicherbandbreite zu optimieren. Wenn zum Beispiel ein Faden_0 nicht von einem Faden_1 abhängt oder mit diesem in Beziehung steht, gibt es keinen Grund dafür, dass Faden_1 und 0 ihre Speicherreferenzen bezüglich der SRAM-Einheit nicht in Unordnung vollenden können. Die Mikromaschinen **22a** bis **22f** erteilen Speicherreferenzanforderungen an die Speichersteuerungen **26a** und **26b**. Die Mikromaschinen **22a** bis **22f** überschwemmen die untergeordneten Speichersysteme **26a** und **26b** mit genug Speicherreferenzvorgängen, so dass die untergeordneten Speichersysteme **26a** und **26b** für den Betrieb des Prozessors **12** zum Flaschenhals werden.

[0029] Wenn das untergeordnete Speichersystem **16** mit Speicheranforderungen überschwemmt wird, die von unabhängiger Natur sind, kann der Prozessor **12** eine Speicherreferenzsortierung ausführen. Die Speicherreferenzsortierung verbessert die erreichbare Speicherbandbreite. Die Speicherreferenzsortierung, wie unten beschrieben, verringert die Totzeit oder eine Blase, die bei Zugriffen auf den SRAM auftritt. Mit Speicherreferenzen auf den SRAM ruft das Umschalten der Stromrichtung auf Signalleitungen zwischen Lese- und Schreibvorgängen eine Blase oder eine Totzeit hervor, die darauf wartet, dass sich der Strom auf Leitern absetzt, die den SRAM **16b** mit der SRAM-Steuerung **26b** verbinden.

[0030] Das heißt, die Treiber, die den Strom auf dem Bus antreiben, müssen sich vor dem Wechsel der Zustände absetzen. Folglich können sich wiederholende Zyklen eines Lesevorgangs gefolgt von einem Schreibvorgang die Spitzenbandbreite verschlechtern. Eine Speicherreferenzsortierung ermöglicht, dass der Prozessor **12** Referenzen bezüglich des Speichers organisiert, so dass lange Reihungen von Lesevorgängen von langen Reihungen von Schreibvorgängen gefolgt werden können. Dies kann benutzt werden, um die Totzeit in der Pipeline zu minimieren und eine näher am Maximum liegende verfügbare Bandbreite effektiv zu erreichen. Eine Referenzsortierung trägt zur Verwaltung paralleler Hardwarekontextfäden bei. Auf dem SDRAM ermöglicht die Referenzsortierung das Verbergen von Vorladungen von einer Bank zu einer anderen Bank. Insbesondere wenn das Speichersystem **16b** in eine ungerade Bank und eine gerade Bank organisiert ist, kann die Speichersteuerung mit dem Vorladen der geraden Bank beginnen, während der Prozessor auf der ungeraden Bank arbeitet. Das Vorladen ist möglich, wenn sich die Speicherreferenzen zwischen ungeraden und geraden Banken abwechseln. Durch Ordnen von Speicherreferenzen, um zwischen Zugriffen auf entgegengesetzte Banken zu wechseln, verbessert der Prozessor **12** die SDRAM-Bandbreite.

[0031] Die FBUS-Schnittstelle **28** unterstützt Sende- und Empfangsflags für jeden Anschluss, den eine MAC-Vorrichtung unterstützt, zusammen mit einem Unterbrechungsflag, das anzeigt, wenn ein Dienst gewährleistet ist. Die FBUS-Schnittstelle **28** weist auch eine Steuerung **28a** auf, die eine Kopffeldverarbeitung hereinkommender Pakete aus dem FBUS **18** ausführt. Die Steuerung **28a** extrahiert die Paketkopffelder und führt einen mikroprogrammierbaren Quelle/Ziel/Protokoll-Streuspeichersuchlauf (der zum Adressglätten benutzt wird) in SRAM aus. Wenn die Streuspeicherung nicht erfolgreich ist, wird das Paketkopffeld zur weiteren Verarbeitung an den Prozessorkern **20** gesendet. Die FBUS-Schnittstelle **28** unterstützt die folgenden internen Datentransaktionen:
FBUS-Einheit (über AMBA-Bus) zu/von Prozessorkern.
FBUS-Einheit (über privaten Bus) zu/von SRAM-Einheit.
FBUS-Einheit (über Mbus) zu/von SDRAM.

[0032] Der FBUS **18** ist ein standardgemäßer Industriebus und weist einen Datenbus, der zum Beispiel 64 Bit breit ist, und eine Seitenbandsteuerung zur Adress- und Lese-/Schreibsteuerung auf. Die FBUS-Schnittstelle **28** stellt die Fähigkeit bereit, mit Hilfe einer Reihe von Eingabe- und Ausgabe-FIFOs **29a** bis **29b** große Datenmengen einzugeben. Aus den FIFOs **29a** bis **29b** rufen die Mikromaschinen **22a** bis **22f** Daten ab oder befehlen der SDRAM-Steuerung **26a**, Daten aus einem Empfangs-FIFO, in dem Daten aus einer Vorrichtung auf einem Bus **18** gekommen sind, in die FBUS-Schnittstelle **28** zu bewegen. Die Daten können durch die Speichersteuerung **26a** über einen Direktspeicherzugriff an den SDRAM-Speicher **16a** gesendet werden. In ähnlicher Weise können die Mikromaschinen Daten von dem SDRAM **26a** zu der Schnittstelle **28** zu dem FBUS **18** über die FBUS-Schnittstelle **28** bewegen.

[0033] Datenfunktionen sind unter den Mikromaschinen verteilt. Die Verbindungsfähigkeit zu dem SRAM **26a**, SDRAM **26b** und FBUS **28** wird über Befehlsanforderungen ausgeführt. Eine Befehlsanforderung kann eine Speicheranforderung oder eine FBUS-Anforderung sein. Zum Beispiel kann eine Befehlsanforderung Daten aus einem Register, das sich

in einer Mikromaschine **22a** befindet, zu einer gemeinsamen Ressource, zum Beispiel einer SDRAM-Stelle, SRAM-Stelle, einem Flashspeicher oder einer MAC-Adresse bewegen. Die Befehle werden an jede der funktionellen Einheiten und die gemeinsamen Ressourcen gesendet. Allerdings brauchen die gemeinsamen Ressourcen keine lokale Pufferung der Daten zu verwalten. Vielmehr greifen die gemeinsamen Ressourcen auf verteilte Daten zu, die sich in den Mikromaschinen befinden. Dies befähigt die Mikromaschinen **22a** bis **22f** dazu, einen lokalen Zugriff auf Daten zu haben anstatt für einen Zugang auf einem Bus zu vermitteln und eine Auseinandersetzung für den Bus zu riskieren. Mit Hilfe dieses Merkmals gibt es keinerlei Zyklusverzögerung, um auf Daten zu warten, die sich in den Mikromaschinen **22a** bis **22f** befinden.

[0034] Die Datenbusse, zum Beispiel der AMBA-Bus **30**, SRAM-Bus **34** und SDRAM-Bus **38**, die die gemeinsamen Ressourcen, zum Beispiel die Speichersteuerungen **26a** und **26b** verbinden, weisen eine ausreichende Bandbreite auf, so dass keine internen Flaschenhälse vorliegen. Um Flaschenhälse zu vermeiden, weist der Prozessor **12** folglich eine Bandbreitenanforderung auf, bei der jede der funktionellen Einheiten mit mindestens dem Doppelten der maximalen Bandbreite der internen Busse bereitgestellt wird. Als ein Beispiel kann der SDRAM einen 64 Bit breiten Bus bei 83 MHz betreiben. Der SRAM-Datenbus könnte getrennte Lese- und Schreibbusse aufweisen, zum Beispiel könnte er ein 32 Bit breiter Lesebus, der bei 166 MHz läuft, und ein 32 Bit breiter Schreibbus sein, der bei 166 MHz läuft. Im Wesentlichen bedeutet dies 64 Bit, die bei 166 MHz laufen, was effektiv dem Doppelten der Bandbreite des SDRAM entspricht.

[0035] Der Kernprozessor **20** kann auch auf die gemeinsamen Ressourcen zugreifen. Der Kernprozessor **20** weist eine direkte Verbindung zu der SDRAM-Steuerung **26a**, zu der Busschnittstelle **24** und über den Bus **32** zu der SRAM-Steuerung **26b** auf. Um jedoch auf die Mikromaschinen **22a** bis **22f** zuzugreifen und Register zu übertragen, die sich bei irgendeiner der Mikromaschinen **22a** bis **22f** befinden, greift der Kernprozessor **20** auf die Mikromaschinen **22a** bis **22f** über den AMBA-Umsetzer **30** über den Bus **34** zu. Der AMBA-Umsetzer **30** kann sich physikalisch in der FBUS-Schnittstelle **28** befinden, ist jedoch logisch eigenständig. Der AMBA-Umsetzer **30** führt eine Adressumsetzung zwischen FBUS-Mikromaschinen-Übertragungsregisterstellen und Kernprozessoradressen (das heißt, AMBA-Bus) aus, so dass der Kernprozessor **20** auf Register zugreifen kann, die zu den Mikromaschinen **22a** bis **22c** gehören.

[0036] Der Prozessorkern **20** weist einen RISC-Kern **50** auf, der in einer fünfstufigen Pipeline umgesetzt ist, die eine Einzyklusverschiebung eines Operanden oder zweier Operanden in einem einzigen Zyklus ausführt, stellt eine Multiplikationsunterstützung und 32-Bit-Barrel-Verschiebungsunterstützung bereit. Der RISC-Kern **50** ist eine standardgemäße Strong Arm®-Architektur, ist jedoch aus Gründen der Leistungsfähigkeit mit einer fünfstufigen Pipeline umgesetzt. Der Prozessorkern **20** weist auch einen 16-Kilobyte-Befehlscache **52**, einen 8-Kilobyte-Datencache **54** und einen Vorzugriffsstrompuffer **56** auf. Der Kernprozessor **20** führt parallel zu Speicherschreibvorgängen und Befehlsabrufen arithmetische Operationen aus. Der Kernprozessor **20** ist mit anderen funktionellen Einheiten über den ARM-definierten AMBA-Bus verbunden. Der AMBA-Bus ist ein bidirektionaler 32-Bit-Bus **32**.

[0037] Mit Bezug auf Fig. 3 ist der Mehrprozessor **12** bei der Ausführung von Netzwerkwegeermittlungsfunktionen dargestellt. In einem Beispiel treten ein asynchroner Transfermodus (ATM), Ethernet und andere Pakettypen durch die Netzwerkschnittstelle in MAC-Vorrichtungen ein und werden an den Netzwerkprozessor **12** gesendet. Diese Pakete werden in einer Anwendung auf einem Universalmikroprozessor **20** oder auf einem anderen Prozessor verarbeitet, der durch die PCI-Busschnittstelle (nicht dargestellt) verbunden ist. Zum Empfang und zur Übertragung solcher Pakete benutzt die Anwendung, die auf diesem Prozessor **20** oder dem Prozessor, der durch den PCI-Bus verbunden ist, einen Netzwerkstapel **72**, der Netzwerkverwaltungs-, Steuerungs- und Signalisierungsprozesse **74** aufweist, um Netzwerkkommunikationen zu verwalten.

[0038] Der Netzwerkstapel **72** und die Anwendung laufen in dem Prozessor **20**, der die Mikromaschinen steuert, oder einem anderen Prozessor, der mit dem PCI-Bus verbunden ist. Die Empfangs-, Sende- und Datenweiterleitungswege repräsentieren den Transport von Paketen durch den Prozessor **12**. Die Verwaltungssteuerung, Signalisierung und der Netzwerkstapel **72** sind gewöhnlich nicht an der Datenweiterleitung beteiligt. Im Wesentlichen empfängt und sendet der Prozessor **20**. Der Prozessor **20** erzeugt neue Pakete, die über das Netzwerk gesendet werden. Der Prozessor **20** kann im Ausnahmefall an der Datenweiterleitung beteiligt sein. Dies würde sehr ungewöhnliche Pakete betreffen, die möglicherweise einer speziellen Handhabung und komplexen Verarbeitung bedürfen.

[0039] Für Datenweiterleitungsprozesse werden die Mikromaschinen **22a** bis **22f** benutzt. In einigen Fällen kann die Datenweiterleitung auf der Ebene des Universalprozessors **20** stattfinden. Die Signale Init sind die Schnittstelle des Programmierers zur Initialisierung von Mikromaschinencode. Das Signal Fini wird zur Beendigung (um Steuerinformation in einen bekannten Zustand zu versetzen) benutzt. Die Mikro-

maschinen **22a** bis **22f** stellen schnelle Speicher- und Weiterleitungsfähigkeiten bereit. Die Maschinen benutzen einen mehrschichtigen generischen Suchlaufprozess, der mit Hilfe paralleler, hardwaregestützter Fäden des Prozesses eine Validierung, Klassifizierung, Kontrolle und Filterung ausführt. Ausnahmen und Steuerpakete werden zur Verarbeitung bei dem Netzwerkstapel **72** an den Prozessor **20** geleitet. Ein ternärer Netzwerkstapel (nicht dargestellt) kann chipextern bei einem Host über den PCI-Anschluss oder Vorrichtungsanschluss angeordnet sein. Dies kann benutzt werden, um den Prozessor **20** oder die zentralisierte Verwaltung und Steuerung für einen Ort zu entlasten. In einigen Ausführungsformen ist die Mikromaschine ein kompakter RISC-Prozessor und kann einen begrenzten Befehlsraum haben. Aus diesem Grund und aus anderen Gründen ist es wünschenswert, die Befehlscodegröße zu verringern, wenn eine Vielzahl von Protokollen ausgeführt wird. Der Netzwerkprozessor **12** setzt einen generischen Weiterleitungsprozess um, der benutzt werden kann, um verschiedene Protokolltypen (sowohl existierende als auch zukünftige Typen) zu handhaben, ohne die Befehlsspeichergrenzen zu überschreiten.

[0040] Mit Bezug auf Fig. 4 ist eine Verwaltungsanordnung **80** zum Weiterleiten von Tabellenstrukturen **90** dargestellt, die im Speicher gespeichert sind. Die Weiterleitungstabellenstrukturverwaltung **80** weist eine Steuer- und Verwaltungsstruktur **82** auf, die eine Netzwerkstapelschnittstelle **84** und Tabellenverwalter **86** aufweist. Die Tabellenverwalter **86** verwalten Wegeermittlungstabellen **90**, die im SRAM gespeichert sind, und können mehrere Tabellen aufweisen, wie die in Fig. 4 dargestellten, die eine Schicht-4-Verbindungstabelle **92**, eine Schicht-3-Zieltabelle **94**, eine Schicht-2-Brückentabelle **96** und eine Schicht-2-Verbindungstabelle **98** aufweisen. Außerdem können Datenstrukturen, die im Speicher gespeichert sind, einen Paketpuffer **100** aufweisen, der im DRAM gespeichert ist. Die Mikromaschinen, die als Prozessoren zur Paketdatenweiterleitung fungieren, rufen aus den Wegeermittlungstabellen **90** im SRAM Information ab und speichern und leiten die Paketinformation aus dem Paketpuffer im DRAM weiter. Die Vielzahl von Tabellen **90** wird von dem Steuerverwaltungsprozessor **20** eingerichtet. Zum Beispiel kann eine Schicht-2-Verbindungstabelle **96** für virtuelle ATM-Schaltungen, Rahmenrelaisverbindungs-MPLS-Kennsätze oder andere Verbindungen auf niedriger Stufe benutzt werden. Eine Schicht-2-Brückentabelle **96** könnte zur Ethernetüberbrückung benutzt werden. Eine Schicht-3-Zieltabelle **94** könnte basierend auf einer Ziel-IP-Adresse zur Internetprotokoll (IP)-Weiterleitung benutzt werden. Die Schicht-4-Verbindungstabelle **92** könnte basierend auf Quellen- und Zielanschlüssen, Adressen und Protokollen zur IP-Weiterleitung benutzt werden. All diese Tabellen können erfordern, dass das Paket entkapselt oder verkapselt wird.

[0041] Sobald die Tabellen **90** in einer im Allgemeinen herkömmlichen Weise mit Weiterleitungsinformation versehen sind, können Paketdatenweiterleitungsprozessoren Pakete empfangen, Tabellensuchläufe ausführen, um Information zu erhalten, und Pakete je nach Erfordernis des Tabelleneintrags umwandeln. Der Steuerverwaltungsprozess richtet die Tabellen **90** zum Zwecke der Verkapselung und Entkapselung mit einem gemeinsamen Format ein.

[0042] Mit Bezug auf Fig. 5 sind beispielhafte Tabelleneinträge, von denen ein untergeordneter Satz in jeder der Tabellen **90** enthalten ist, dargestellt. Die Tabelleneinträge weisen die folgenden Felder auf:

Weiterleitungstabellenformat

[0043] Entkapselungs-Flag. Zeigt an, ob die Bytes von dem Paket abgezogen werden sollten. Wenn das Flag bestätigt wird, befindet sich die Anzahl abzuziehender Bytes in dem Entkapselungs-Bytezählfeld.

[0044] Entkapselung-zu-Schicht. Dieses Feld spezifiziert die Entkapselung von Kopffeldschichten bis zu der spezifizierten Schicht. Die Länge der Schicht und somit die Entkapselung werden durch syntaktisches Analysieren des Paketkopffeldes bestimmt.

[0045] Entkapselungs-Bytezählung. Dieses Feld spezifiziert die Anzahl von Bytes, die aus der Vorderseite des Pakets zu entfernen sind. Eine Entkapselung wird durch Einstellen des Paketstartversatzes in dem Paketpuffer ausgeführt.

[0046] Derzeitige Verkapselung. Dieses Feld spezifiziert einen Identifikator des derzeitigen Paketverkapselungstyps.

[0047] Verkapselungs-Flag. Zeigt an, ob dem Paket Bytes vorangestellt werden sollen. Wenn dieses Flag bestätigt wird, dann befindet sich die Anzahl von Bytes in dem Verkapselungs-Bytezählfeld und die zu verkapselnden Bytes werden in dem Verkapselungs-Kopffeld verkapselt.

[0048] Verkapselungs-Bytezählung. Anzahl von Bytes, die dem Paket vorangestellt werden sollen.

[0049] Verkapselungskopffeld. Die tatsächlichen Bytes, die vorangestellt werden sollen.

[0050] Nächster Tabellentyp. Falls nicht null, zeigt dies an, dass ein weiterer Suchlauf erforderlich ist. Dies ergibt den Tabellentyp. Zum Beispiel den Schicht-3-Wegeermittlungs- oder Schicht-4-Verbindungstabellentyp. Ein Schicht-3-Wegeermittlungssuchlauf würde mit Hilfe der Ziel-IP-Adresse einen Suchlaufalgorithmus mit Übereinstimmung des längsten Präfixes benutzen. Ein Schicht-4-Verbindungssuchlauf würde unter Verwendung von Quel-

len- und Zieladressen, Quellen- und Zielanschlüssen und Protokoll einen 104-Bit-Hash-Algorithmus benutzen.

[0051] Nächste Tabellenadresse. Es kann eine Vielzahl nächster Tabellen und eine Vielzahl nächster Tabellen des gleichen Typs geben. Dieses Feld spezifiziert die Basisadresse der Tabelle.

[0052] Die Flags werden durch den Verwaltungsprozess gesetzt oder gelöscht. Signalisierungs- und Einrichtungsverbindungen sind Teil des Netzwerksystems, die bestimmen werden, dass ein bestimmter Weg durch das Netzwerk einen Wechsel des Kopffeldes erfordert. Es kann viele Gründe geben, warum ein Kopffeld gewechselt werden kann. Gewöhnlich wird ein Kopffeldwechsel benutzt, wenn das Protokoll von einer Netzwerkdomäne zu einer anderen wechselt.

[0053] Mit Bezug auf Fig. 6 ist ein Prozess **110** zum Verkapseln/Entkapseln generischer Protokolle dargestellt. Anfangs empfängt **112** eine der Mikromaschinen **22a** bis **22f** ein Paket von der Netzwerkschnittstelle. Das Paket besteht aus einem oder mehreren Kopffeldern, gefolgt von Nutzdaten. Die Mikromaschine, zum Beispiel **22a**, kopiert den Nutzdatenabschnitt des Pakets in einen Paketpuffer im DRAM und kann das Paket bei einem Versatz in dem Puffer anordnen, um Raum für neue Kopffelder zu schaffen, die dem Paket zur Paketweiterleitung vorangestellt werden könnten. Der Paketversatzparameter für dieses Paket wird auf einen Standardwert festgelegt, der bei dem Versatz in den Puffer bestimmt wird. Die Mikromaschine liest **114** das erste Kopffeld des Pakets und führt einen Schicht-2-Suchlauf aus. Der Schicht-2-Suchlauf liest die Tabelle Schicht-2-Brückentabelle und/oder Schicht-2-Verbindungstabelle. Die Tabellen senden verschiedene Parameter wie Entkapselungs- oder Verkapselungs-Flags zurück. Der Prozess **110** wird bestimmen **116**, ob die Entkapselungs- oder Verkapselungs-Flags gesetzt sind. Wenn die Entkapselungs- und Verkapselungs-Flags gesetzt sind, addiert **118** der Prozess die Entkapselungs-Bytezählung zu dem Paketstartversatz und subtrahiert **120** die Verkapselungs-Bytezählung von dem Paketstartversatz und stellt die Verkapselungsbytes dem Paket voran. Der Prozess **110** überprüft **122**, ob eine nächste zu untersuchende Tabelle vorhanden ist, indem ein leeres Feld in einer derzeit gelesenen Tabelle betrachtet wird. Wenn es eine nächste Tabelle gibt, analysiert der Prozess **110** das nächste Kopffeld **124** syntaktisch, ruft die nächste Tabelle ab und liest sie. Der Prozess **110** führt den Suchlauf fort, um die gesetzten Entkapselungs- oder Verkapselungs-Flags zu überprüfen.

[0054] Falls der Prozess jedoch nicht bestimmt hat, dass die Entkapselungs- und Verkapselungs-Flags gesetzt wurden (116, oben), würde er bestimmen **130**, ob das Verkapselungs-Flag oder das Entkapselungs-Flag gesetzt wurden **132**. Wenn das Verkapselungs-Flag gesetzt wurde, wird er die Verkapselungs-Flag-Bytezählung von dem Anfangsversatz subtrahieren **120** und dem Paket die Verkapselungsbytes voranstellen. Wenn andererseits das Entkapselungs-Flag nur gesetzt wurde **132**, addiert **134** der Prozess eine Entkapselungs-Bytezählung zu dem Pufferversatz und überprüft in jedem Fall die nächste Tabelle **112**. Wenn der Prozess bestimmt, dass er am Ende der Überprüfung der Tabellen ist, wird er dann das Paket in herkömmlicher Weise klassifizieren und weiterleiten **136**. Das heißt, die "Nein"-Bedingung zeigt an, dass der Prozess klassifizieren und weiterleiten kann. Das Weiterleiten des Kopffeldes kann bedeuten, dass die Mikromaschine das Kopffeld nimmt und es an den Prozessor **20** oder woandershin sendet, so dass es mit den Nutzdaten neu angeordnet werden kann. Das Weiterleiten des Kopffeldes könnte auch das Weiterleiten der Pakete usw. betreffen.

[0055] Mit Bezug auf Fig. 7 werden außerdem Byteverbreitungszählungen spezifiziert, die aus der Nachschlagtabelle erhalten werden. Das Entkapselungs-zu-Schicht-Feld kann in der Nachschlagtabelle gesetzt sein. Dieses Feld spezifiziert, dass der Vorderabschnitt eines Pakets bis zu einer bestimmten Schicht entkapselt werden soll. Bekanntermaßen sind Pakete in Protokollschichten definiert, die in dem siebenschichtigen OSI (Open Systems Interconnect)-Netzwerkprotokoll benutzt werden. Nach Passieren der physikalischen Schicht 1 ist die erste Softwareschicht, die von der Netzwerkprozessorschicht gesehen wird, Schicht 2, die auch als die Verbindungsschicht bezeichnet wird. Die Länge der zu entkapselnden Bytes wird durch syntaktisches Analysieren der Paketschichten vor der Schicht, die der Neustart des Pakets sein soll, bestimmt. Die Länge kann zu dem Paketstartversatz hinzugefügt werden.

[0056] Fig. 7 zeigt eine Variation, wobei die Entkapselungslänge nicht in der Tabelle spezifiziert ist, sondern durch Lesen des Pakets selbst bestimmt wird. Mit anderen Worten wäre dies ein Satz von Routinen, die in die Verarbeitung der verkapselten Ersatzbytezählung von Fig. 6 von dem Paket in den Versatz eingefügt würden.

[0057] Ein Prozess **140** zum Bestimmen dieses Versatzes ist in Fig. 7 dargestellt. Zu dem Prozess **140** gehört das Lesen der Tabelle **142**, Bestimmen, dass das Entkapselungs-zu-Paketschicht-Bit **144** gesetzt worden ist, und falls es gesetzt worden ist, Abrufen der Länge der zu entfernenden Schicht durch syntaktisches Analysieren des Kopffeldes **146** und Addieren der Länge zu dem Paketstartversatz **148**. Wenn die Entkapselungsschicht nicht gesetzt worden ist, dann übergeht der Prozess dies einfach. In jedem Fall kann dieser Prozess dem in Verbindung mit Fig. 6

beschriebenen Prozess vorangestellt werden.

[0058] Eine typische Verwendung eines Entkapselungs-zu-Schicht-Bits ist die Spezifizierung einer Entkapselung bis zu dem Schicht-3-IP-Kopffeld. Wenn die Paketverkapselung ein Multiprotokoll über ein ATM-Netzwerk wie der Standard RFC 1483 ist, wird die Schicht-2-Kopffeldlänge durch syntaktisches Analysieren der Schicht bezüglich des Kopffeldes selbst unter Anwendung der RFC 1483-Längenregeln bestimmt. Wenn die Paketverkapselung jedoch ein klassisches IP ist, wird die Schicht-2-Länge durch Befolgen der klassischen IP-Schichtlängenregeln bestimmt. Die Paketverkapselung kann durch den Anschlusstyp, durch den sie hereinkam, oder durch das vorangestellte benutzerdefinierte Kopffeld aus diesem Anschluss bekannt sein oder von der ersten Nachschlagtabelle in dem derzeitigen Verkapselungsfeld erhalten werden.

[0059] Anstatt dass jedes Netzwerkprotokoll eine getrennte Protokollumwandlung definiert, stellt diese Technik einen generischen Ansatz bereit. Der Ansatz spart Coderaum und Softwareentwicklungs-Produkteinführungszeit. In einer alternativen Ausführungsform kann diese Technik als eine Softwarebibliothekenroutine, zum Beispiel ein generischer Softwarebildungsblock zur Entkapselung/Verkapselung umgesetzt werden, wobei Kunden ihre firmeneigene Kopffeldverkapselung einfügen können und der Lieferant eines Kunden nicht an den firmeneigenen Protokollgestaltungen des Kunden beteiligt sein muss.

[0060] Verschiedene Ausführungsformen der Erfindung sind beschrieben worden. Dennoch wird man verstehen, dass verschiedene Modifikationen vorgenommen werden können, ohne den Schutzbereich der Erfindung zu verlassen. Demgemäß liegen andere Ausführungsformen innerhalb des Schutzbereichs der folgenden Ansprüche.

Patentansprüche

1. Verfahren (**110**) zum Weiterleiten von Paketen über mehrere Netzwerkprotokolle von einer Netzwerkwegeermittlungsvorrichtung, wobei das Verfahren (**110**) Folgendes umfasst:
Empfangen eines Pakets (**112**);
Lesen des Inhalts mindestens eines Kopffeldes des Pakets (**112**), wobei der Inhalt einem von mehreren Netzwerkprotokollen zugeordnet ist; gekennzeichnet durch
Lesen einer Tabelle (**114**), die dem Inhalt des mindestens einen Kopffeldes zugeordnet ist, wobei die Tabelle mehrere Flags enthält, einschließlich eines Verkapselungs-Flags, und die Tabelle ein Verkapselungskopffeld enthält, wobei das Verkapselungskopffeld Bytes zum Voranstellen vor ein Paket umfasst, um zu bestimmen, welches der mehreren Flags gesetzt oder gelöscht ist; und
Ausführen eines Vorgangs (**118**, **120**, **134**) bezüglich des Pakets, um das Paket durch Voranstellen (**120**) des Verkapselungskopfes vor das Paket zu verkapseln, wenn das Verkapselungs-Flag gesetzt ist (**116**, **130**).

2. Verfahren (**110**) nach Anspruch 1, wobei die Tabelle mit Weiterleitungsinformation gefüllt wird.

3. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabelle Weiterleitungstabellenstrukturen mit einer Steuerungs- und Verwaltungsstruktur aufweist, die eine Netzwerkstapelschnittstelle und Tabellenverwalter aufweist.

4. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabellenverwalter die Wegeermittlung von Tabellen verwalten und mehrere Tabellen aufweisen, die eine Schicht-4-Verbindungstabelle, eine Schicht-3-Zieltabelle, eine Schicht-2-Brückentabelle und eine Schicht-2-Verbindungstabelle aufweisen.

5. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabelle ein Flag aufweist, um anzuzeigen, ob Bytes von dem Paket abgezogen werden sollen, und ein Feld aufweist, das eine Anzahl von abzuziehenden Bytes anzeigt.

6. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabelle ein Feld aufweist, das die Entkapselung von Kopffeldschichten bis zu einer spezifizierten Schicht spezifiziert.

7. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabelle ein Feld aufweist, das einen Identifikator eines derzeitigen Paketverkapselungstyps spezifiziert.

8. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabelle ferner ein Feld aufweist, das eine Anzahl von zu verkapselnden Bytes spezifiziert.

9. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei die Tabelle ein Feld Nächster Tabellentyp aufweist, das anzeigt, dass ein weiterer Suchlauf erforderlich ist, und einen Tabellentyp identifiziert.

10. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, ferner umfassend, wenn das Verkapselungs-Flag gesetzt wurde (**116**, **130**), das Subtrahieren eines Verkapselungsbytes von einem Paketstartversatz (**118**).

11. Verfahren (**110**) nach Anspruch 1 oder Anspruch 10, wobei das Lesen eines Tabellenschritts (**114**) das Ausführen eines Schicht-2-Suchlauf-Lesevorgangs einer Verbindungstabelle aufweist, die Pa-

rameter zurücksendet.

12. Verfahren nach einem der vorhergehenden Ansprüche, wobei die Tabelle ein Entkapselungs-Flag enthält, ferner umfassend das Bestimmen (**116**, **130**), ob das Entkapselungs-Flag gesetzt ist.

13. Verfahren (**110**) nach Anspruch 12, ferner umfassend, wenn das Entkapselungs-Flag gesetzt wurde (**116**, **130**), das Addieren einer Entkapselungsbytezählung zu einem Paketstartversatz (**118**).

14. Verfahren (**110**) nach Anspruch 12 oder Anspruch 13, wobei, wenn das Entkapselungs-Flag gesetzt wurde, das Verfahren ferner das Addieren einer Entkapselungsbytezählung zu einem Pufferversatz (**134**) und Überprüfen einer nächsten Tabelle umfasst.

15. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, ferner umfassend: Bestimmen, ob eine nächste zu untersuchende (**122**) Tabelle vorhanden ist, indem ein leeres Feld in einer derzeit gelesenen Tabelle betrachtet wird.

16. Verfahren (**110**) nach Anspruch 15, wobei, falls eine nächste Tabelle vorhanden ist, das Verfahren ferner das syntaktische Analysieren eines nächsten Kopffeldes (**124**) und das Abrufen und Lesen einer nächsten Tabelle umfasst.

17. Verfahren (**110**) nach einem der vorhergehenden Ansprüche, wobei das Paket aus einem oder mehreren Kopffeldern gefolgt von Nutzdaten besteht, wobei das Verfahren (**110**) ferner Folgendes umfasst: Kopieren des Nutzdatenabschnitts des Pakets in einen Paketpuffer.

18. Verfahren (**110**) nach Anspruch 17, wobei das Kopieren das Paket bei einem Versatz in dem Puffer anordnet, um Raum für jedes beliebige neue Kopffeld zu schaffen, das dem Paket zur Paketweiterleitung vorangestellt werden könnte.

19. Computerprogrammprodukt zum Weiterleiten von Paketen über mehrere Netzwerkprotokolle durch eine Netzwerkwegeermittlungsvorrichtung, wobei das Computerprogrammprodukt Befehle umfasst, um einen Computer zu Folgendem zu veranlassen:
Empfangen eines Pakets (**112**);
Lesen des Inhalts mindestens eines Kopffeldes des Pakets (**112**), wobei der Inhalt einem der mehreren Netzwerkprotokolle zugeordnet ist; dadurch gekennzeichnet, dass das Computerprogrammprodukt Befehle umfasst, um einen Computer zu Folgendem zu veranlassen:
Lesen einer Tabelle, die dem Inhalt des mindestens einen Kopffeldes (**114**) zugeordnet ist, wobei die Tabelle mehrere Flags enthält, einschließlich eines Verkapselungs-Flags, und die Tabelle ein Verkapselungskopffeld enthält, wobei das Verkapselungskopffeld Bytes zum Voranstellen vor ein Paket umfasst, um zu bestimmen, welches der mehreren Flags gesetzt oder gelöscht ist; und
Ausführen eines Vorgangs bezüglich des Pakets, um das Paket durch Voranstellen des Verkapselungskopffeldes vor das Paket zu verkapseln, wenn bestimmt wurde, dass das Verkapselungs-Flag gesetzt ist.

20. Computerprogrammprodukt nach Anspruch 19, wobei die Tabelle mit Weiterleitungsinformation gefüllt wird.

21. Computerprogrammprodukt nach Anspruch 20, wobei die Weiterleitungsinformation eine Steuerungs- und Verwaltungsstruktur aufweist, die eine Netzwerkstapelschnittstelle und Tabellenverwalter aufweist.

22. Computerprogrammprodukt nach einem der Ansprüche 19 bis 21, wobei, wenn die Entkapselungs- und Verkapselungs-Flags gesetzt sind (**116**), das Computerprogramm Befehle ausführt, um Folgendes zu veranlassen: Addieren einer Entkapselungsbytezählung zu einem Paketstartversatz (**118**) und Subtrahieren einer Verkapselungsbytezählung von dem Paketstartversatz (**120**).

23. Computerprogrammprodukt nach einem der Ansprüche 19 bis 22, ferner umfassend Befehle, um Folgendes auszuführen: Bestimmen, ob eine nächste zu untersuchende (**122**) Tabelle vorhanden ist, indem ein leeres Feld in einer derzeit gelesenen Tabelle betrachtet wird.

24. Computerprogrammprodukt nach Anspruch 23, wobei, wenn eine nächste Tabelle vorhanden ist, das Computerprogramm Befehle ausführt, um Folgendes zu veranlassen: syntaktisches Analysieren des nächsten Kopffeldes (**124**) und Abrufen und Lesen der nächsten Tabelle.

25. Computerprogrammprodukt nach einem der Ansprüche 19 bis 24, wobei das Paket aus einem oder mehreren Kopffeldern gefolgt von Nutzdaten besteht, wobei das Computerprogramm ferner Befehle ausführt, um Folgendes zu veranlassen: Kopieren des Nutzdatenabschnitts des Pakets in einen Paketpuffer.

26. Computerprogrammprodukt nach Anspruch 25, wobei Befehle zum Kopieren das Paket bei einem Versatz in dem Puffer anordnen, um Raum für jedes beliebige neue Datenfeld zu schaffen, das dem Paket zur Paketweiterleitung vorangestellt werden könnte.

27. Computerprogrammprodukt nach einem der Ansprüche 19 bis 26, wobei sich das Computerprogrammprodukt auf von einem Computer lesbaren

Medien befindet.

28. Vorrichtung zum Weiterleiten von Paketen über mehrere Netzwerkprotokolle durch eine Netzwerkwegeermittlungsvorrichtung, wobei die Vorrichtung Folgendes umfasst:
eine Empfangseinheit zum Empfangen eines Pakets;
eine erste Leseeinheit zum Lesen des Inhalts mindestens eines Kopffeldes des Pakets, wobei der Inhalt einem von mehreren Netzwerkprotokollen zugeordnet ist; dadurch gekennzeichnet, dass die Vorrichtung Folgendes umfasst:
eine zweite Leseeinheit zum Lesen einer Tabelle, die dem Inhalt des mindestens einen Kopffeldes zugeordnet ist, wobei die Tabelle mehrere Flags enthält, einschließlich eines Verkapselungs-Flags, und die Tabelle ein Verkapselungskopffeld enthält, wobei das Verkapselungskopffeld Bytes zum Voranstellen vor ein Paket umfasst, um zu bestimmen, welches der mehreren Flags gesetzt oder gelöscht ist; und
eine Betriebseinheit zum Voranstellen des Verkapselungskopffeldes vor das Paket, wenn bestimmt wird, dass das Verkapselungskopffeld gesetzt ist.

29. Vorrichtung nach Anspruch 28, wobei die Tabelle Weiterleitungsinformation umfasst.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

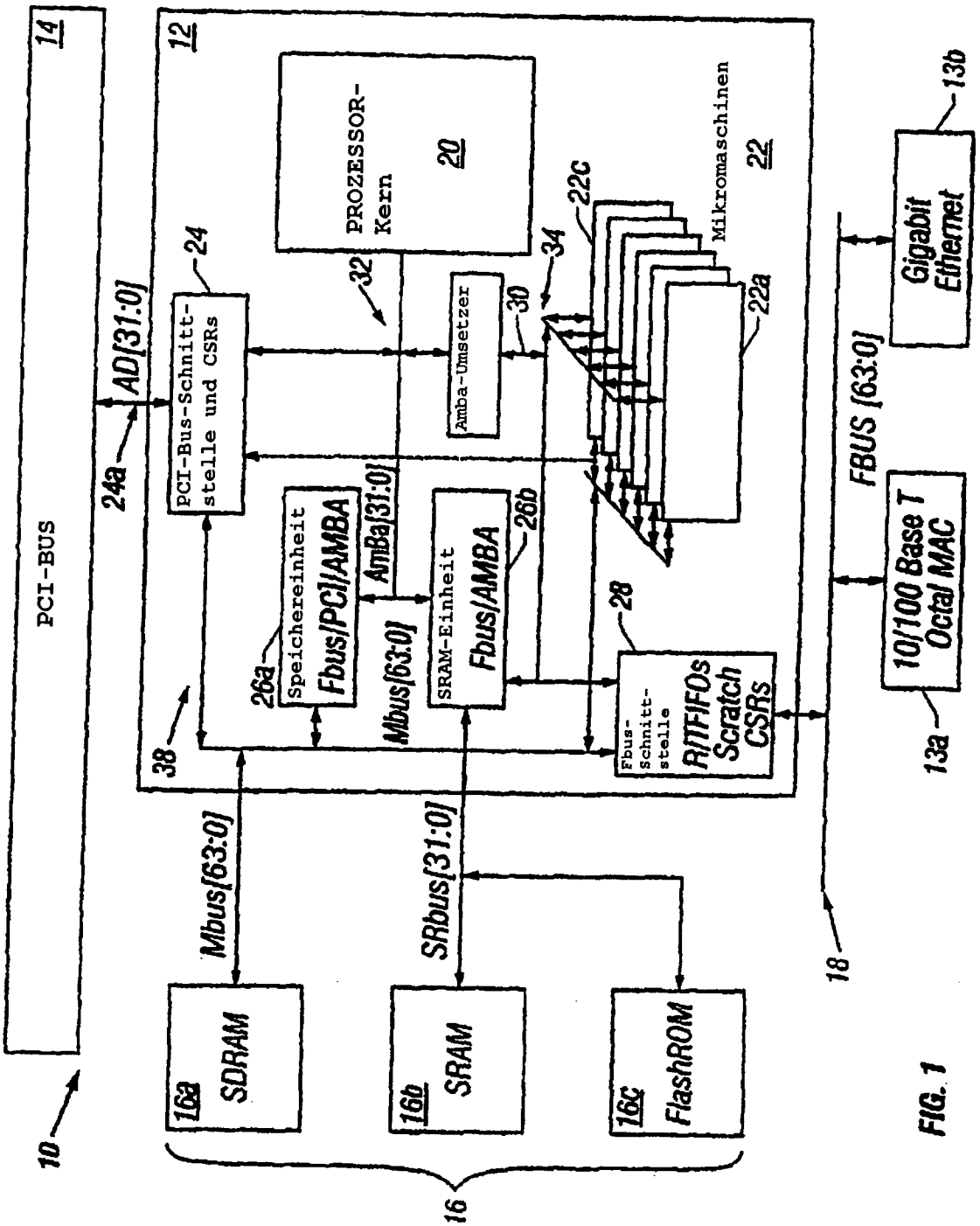


FIG. 1

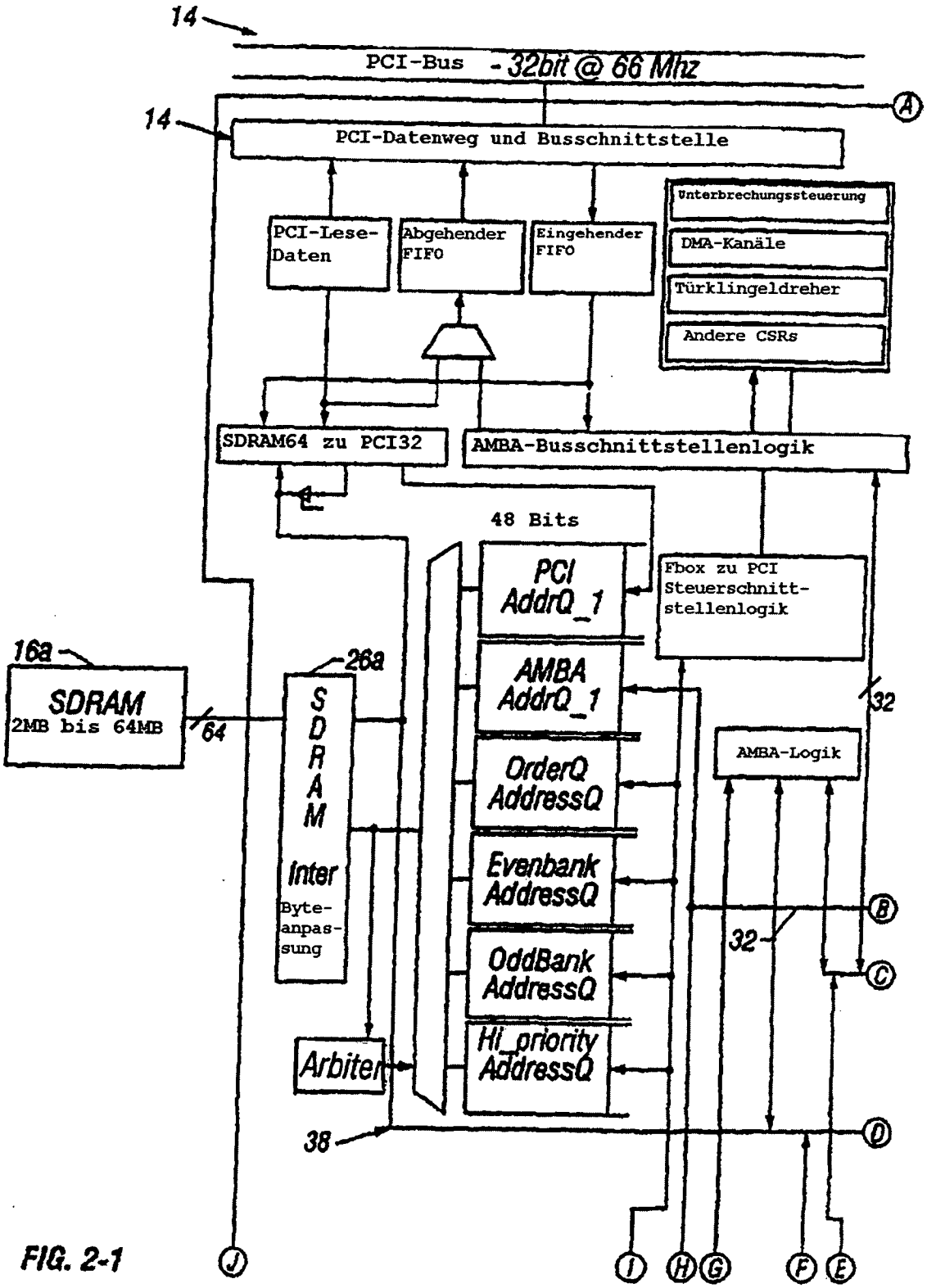

14
PCI-Bus - 32bit @ 66 Mhz
A
14
PCI-Datenweg und Busschnittstelle
Unterbrechungssteuerung
DMA-Kanäle
Türklingeldreher
Andere CSRs
PCI-Lese-Daten
Abgehender FIFO
Eingehender FIFO
SDRAM64 zu PCI32
AMBA-Busschnittstellenlogik
48 Bits
PCI AddrQ_1
Fbox zu PCI Steuerschnitt-stellenlogik
AMBA AddrQ_1
16a
SDRAM
2MB bis 64MB
64
26a
SDRAM Inter Byte-anpas-sung
OrderQ AddressQ
AMBA-Logik
32
Evenbank AddressQ
B
32
OddBank AddressQ
C
Arbiter
Hi_priority AddressQ
38
D
J
I
H
G
F
E

FIG. 2-1

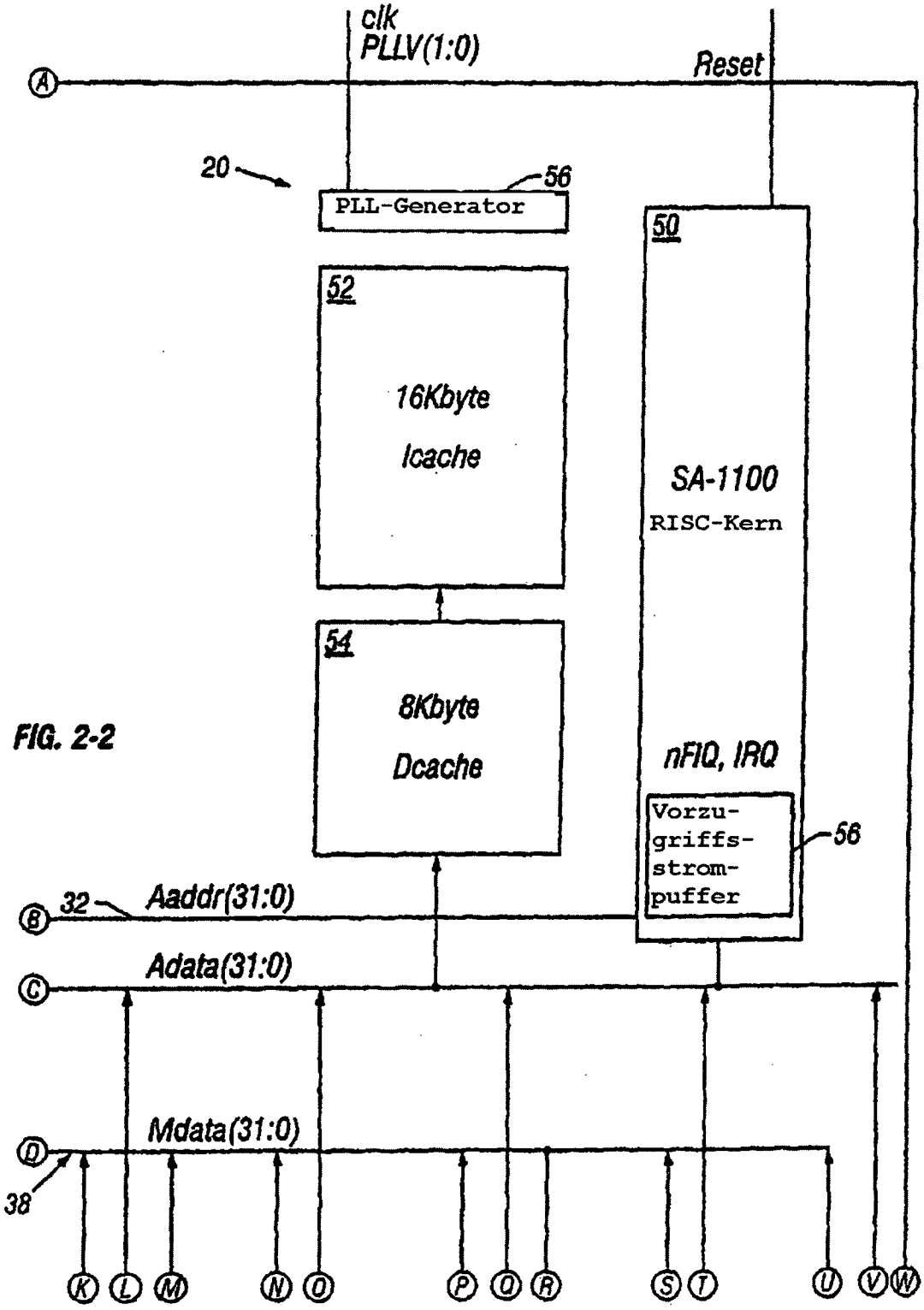

clk
PLLV(1:0)
Reset
A
20
56
PLL-Generator
50
52
16Kbyte
Icache
SA-1100
RISC-Kern
54
8Kbyte
Dcache
nFIQ, IRQ
Vorzu-
griffs-
strom-
puffer
56
B
32
Aaddr(31:0)
C
Adata(31:0)
D
Mdata(31:0)
38
K
L
M
N
O
P
Q
R
S
T
U
V
W

FIG. 2-2

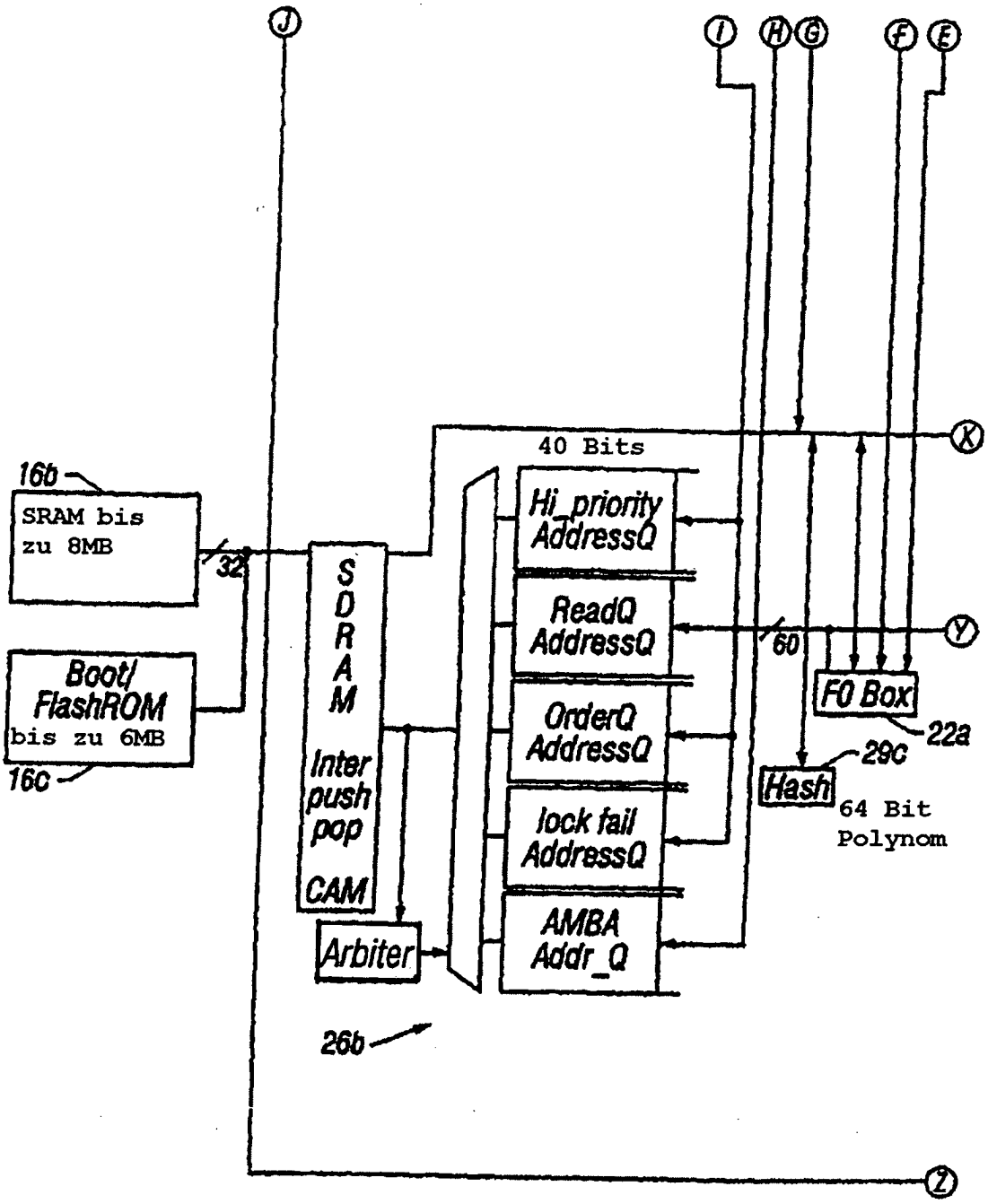

J
I
H
G
F
E
X
Y
Z
40 Bits
16b
SRAM bis
zu 8MB
32
Boot/
FlashROM
bis zu 6MB
16c
SDRAM
Inter
push
pop
CAM
Arbiter
26b
Hi_priority
AddressQ
ReadQ
AddressQ
OrderQ
AddressQ
lock fail
AddressQ
AMBA
Addr_Q
60
F0 Box
22a
29c
Hash
64 Bit
Polynom

FIG. 2-3

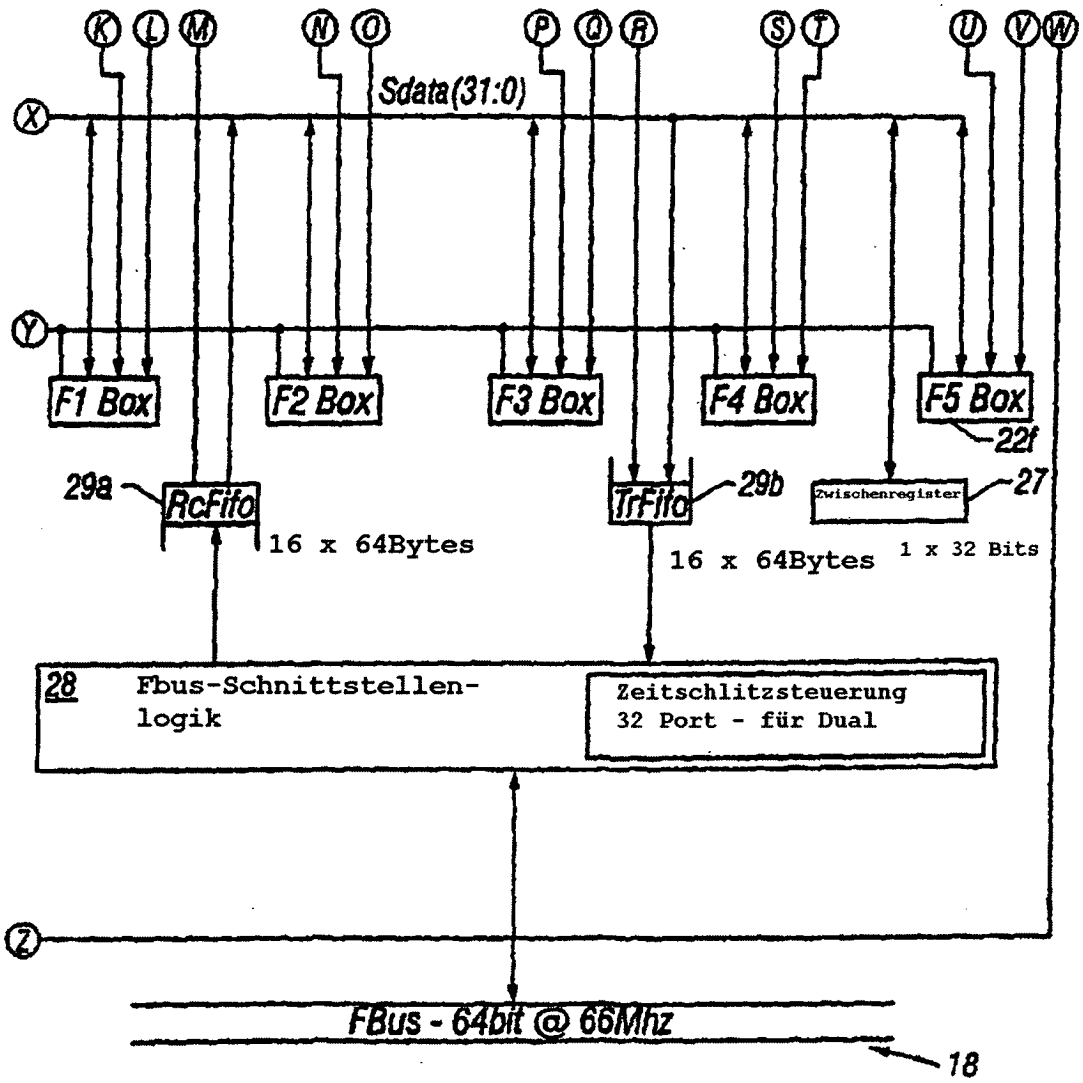
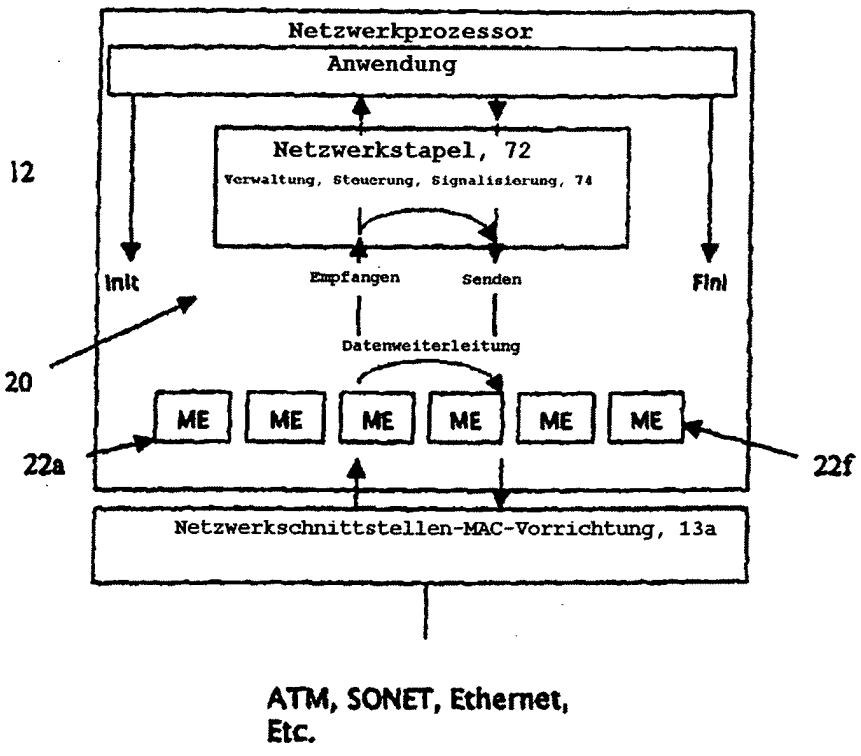

K
L
M
N
O
P
Q
R
S
T
U
V
W
Sdata(31:0)
X
Y
F1 Box
F2 Box
F3 Box
F4 Box
F5 Box
22f
29a
RcFifo
16 x 64Bytes
TrFifo
29b
Zwischenregister
27
16 x 64Bytes
1 x 32 Bits
28
Fbus-Schnittstellen-logik
Zeitschlitzsteuerung
32 Port - für Dual
Z
FBus - 64bit @ 66Mhz
18

FIG. 2-4

FIG. 3


Netzwerkprozessor
Anwendung
Netzwerkstapel, 72
Verwaltung, Steuerung, Signalisierung, 74
12
Init
Empfangen
Senden
Fini
Datenweiterleitung
20
ME
ME
ME
ME
ME
ME
22a
22f
Netzwerkschnittstellen-MAC-Vorrichtung, 13a
ATM, SONET, Ethernet,
Etc.

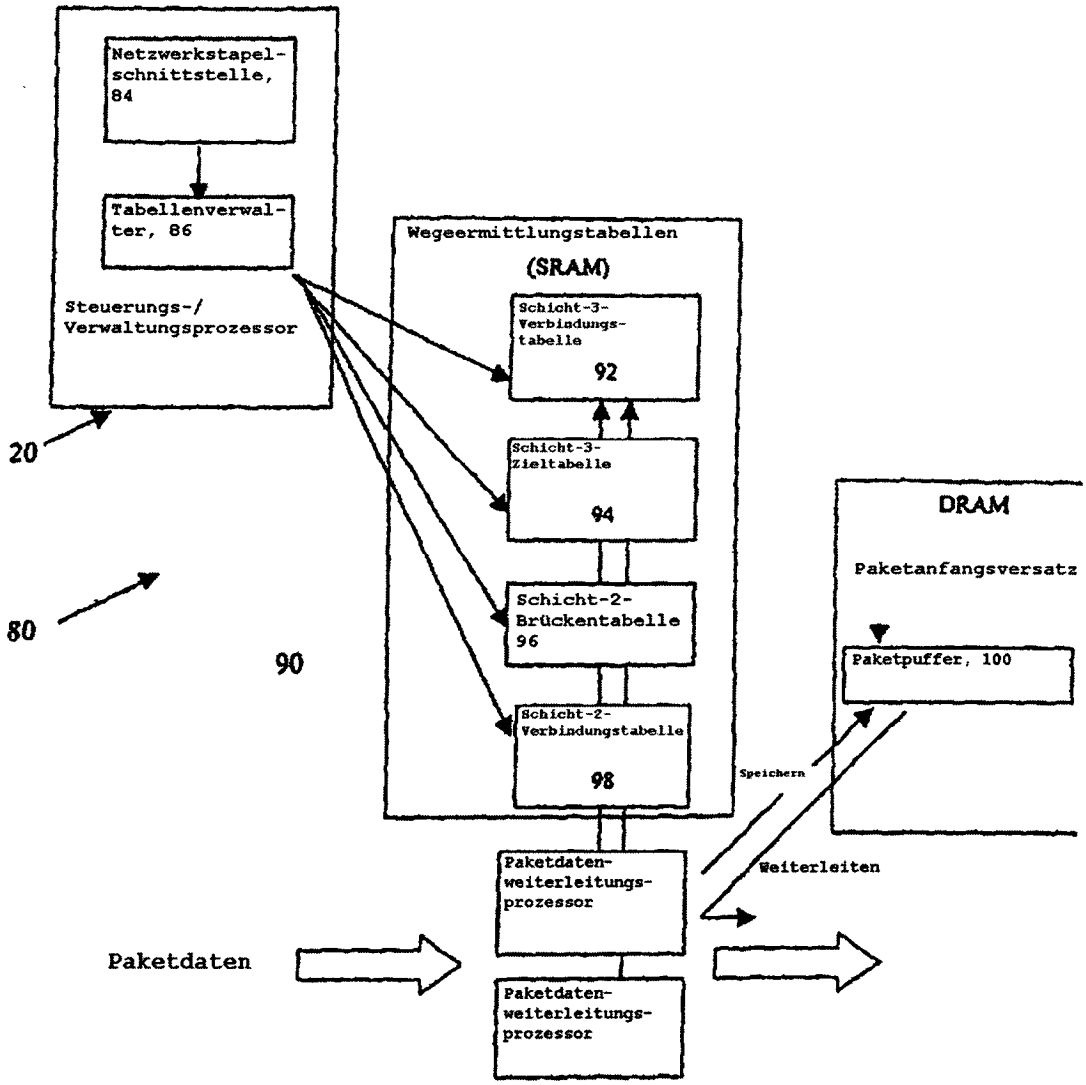


FIG. 4

Entkaps.-Flag	Entkaps.-zu-Schicht	Entkaps.-Bytezählung	Derzeitige Verkaps.	Verkaps.-Flag	Verkaps.-Bytezählung	Verkaps.-Bytes	Nächster Tabellentyp	Nächste Tabellenadresse

FIG. 5

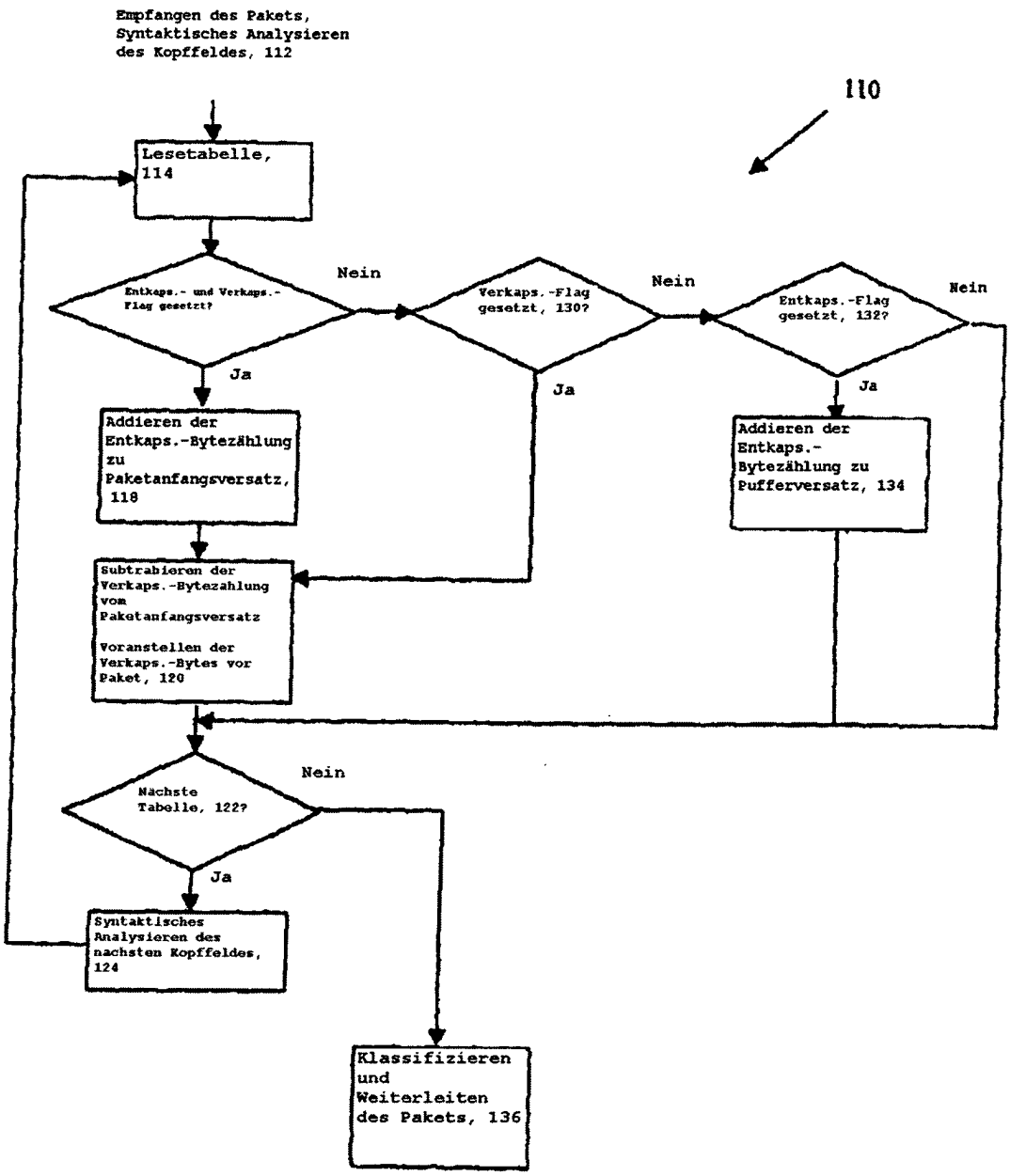

Empfangen des Pakets, Syntaktisches Analysieren des Kopffeldes, 112
110
Lesetabelle, 114
Entkaps.- und Verkaps.-Flag gesetzt?
Nein
Verkaps.-Flag gesetzt, 130?
Nein
Entkaps.-Flag gesetzt, 132?
Nein
Ja
Addieren der Entkaps.-Bytezählung zu Paketanfangsversatz, 118
Ja
Addieren der Entkaps.-Bytezählung zu Pufferversatz, 134
Subtrahieren der Verkaps.-Bytezahlung vom Paketanfangsversatz
Voranstellen der Verkaps.-Bytes vor Paket, 120
Nächste Tabelle, 122?
Nein
Ja
Syntaktisches Analysieren des nachsten Kopffeldes, 124
Klassifizieren und Weiterleiten des Pakets, 136

FIG. 6

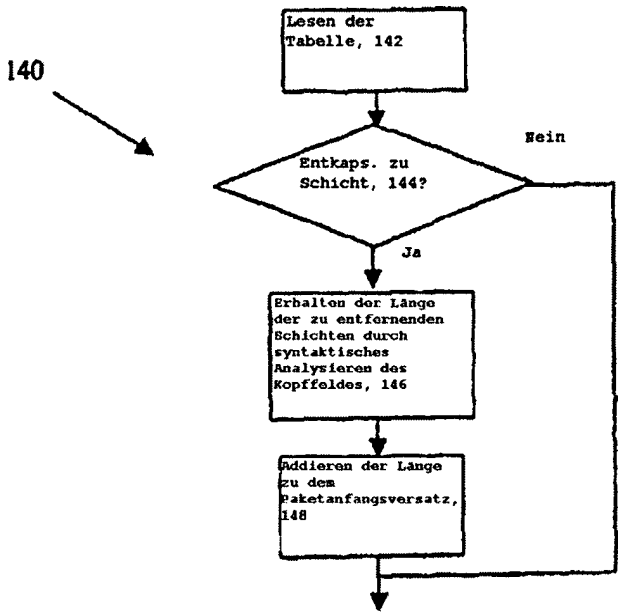

140
Lesen der Tabelle, 142
Entkaps. zu Schicht, 144?
Nein
Ja
Erhalten der Länge der zu entfernenden Schichten durch syntaktisches Analysieren des Kopffeldes, 146
Addieren der Länge zu dem Paketanfangsversatz, 148

FIG. 7