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

(10) **DE 696 27 750 T2** 2004.03.25

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 747 821 B1**
(21) Deutsches Aktenzeichen: **696 27 750.6**
(96) Europäisches Aktenzeichen: **96 304 202.3**
(96) Europäischer Anmeldetag: **06.06.1996**
(97) Erstveröffentlichung durch das EPA: **11.12.1996**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **02.05.2003**
(47) Veröffentlichungstag im Patentblatt: **25.03.2004**

(51) Int Cl.[7]: **G06F 11/16**

(30) Unionspriorität:
485053 07.06.1995 US

(73) Patentinhaber:
Compaq Computer Corp., Houston, Tex., US

(74) Vertreter:
Andrae Flach Haug, 83022 Rosenheim

(84) Benannte Vertragsstaaten:
DE, FR, GB, IT, SE

(72) Erfinder:
Sonnier, David Paul, Austin, Texas 78750, US; Baker, William Edward, Austin, Texas 78727, US; Bunton, William Patterson, Pflugerville, Texas 78660, US; Krause, John C., Georgetown, Texas 78626, US; Porter, Kenneth H., Austin, Texas 78756, US; Zalzala, Linda Ellen, Austin, Texas 78717, US; Watson, William J., Austin, Texas 78756, US

(54) Bezeichnung: **Verfahren zur Divergenzdetektion zwischen einem Paar von synchronisierten Duplexprozessorelementen**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

[0001] Die offengelegte Erfindung steht im Zusammenhang mit den gemeinsam abgetretenen, ebenfalls schwebenden Anmeldungen mit den Seriennummern 08/485,217, 08/482,618, 08/481,749, 08/474,772, 08/473,541, 08/474,770, 08/4,72,222, 08/477,807, 08/483,748, 08/489,281, 08/482,628, 08/479,473, 08/485,062, 08/485,446 and 08/485,055, welche gleichzeitig mit der vorliegenden Anmeldung eingereicht wurden.

[0002] Die vorliegende Erfindung betrifft im allgemeinen Datenverarbeitungssysteme und im spezielleren ein Mehrfach-Verarbeitungssystem und ein zuverlässiges, systemweites Netzwerk, welches eine Anschlussfähigkeit für die Kommunikation zwischen Rechnern sowie für die Eingangs-/Ausgangskommunikation bereitstellt. Weiterhin ist das System entsprechend aufgebaut, um Fehlertoleranzfähigkeit aufzuweisen.

[0003] Das heutige fehlertolerante Rechenwesen geht zurück auf spezielle, militärische und kommunikationstechnische Systeme und hat sich mittlerweile zu universell einsetzbaren, hochverfügbaren, kommerziellen Systemen weiterentwickelt. Die Entwicklungsgeschichte fehlertoleranter Computer 1982 ist gut dokumentiert (siehe D. P. Siewiorek, R. S. Swarz, "The Theory and Practice of Reliable System Design," (Theorie und Praxis des Entwurfs zuverlässiger Systeme) Digital Press, 1982, und A. Avizienis, H. Kopetz, J. C. Laprie, Hrsg., "The Evolution of Fault Tolerant Computing," (Die Entwicklungsgeschichte des fehlertoleranten Rechenwesens) Wien: Springer-Verlag, 1987). Die ersten hochverfügbaren Systeme wurden in den 50er Jahren des vergangenen Jahrhunderts von IBM, Univac und Remington Rand für militärische Anwendungen entwickelt. In den 1960er Jahren begannen die NASA, IBM, SRI, das C. S.

[0004] Draper Laboratory und das Jet Propulsion Laboratory damit, das Prinzip der Fehlertoleranz bei der Entwicklung von Computern für die Lenksysteme von Luftfahrtanwendungen einzusetzen. In den 1960er Jahren erfolgte auch die Entwicklung der ersten elektronischen Vermittlungsanlagen von AT & T.

[0005] Die ersten kommerziellen, fehlertoleranten Rechner wurden in den 1970er Jahren von Tandem Computers zum Einsatz in Anwendungen zur Verarbeitung von Online-Transaktionen eingeführt (J. Bartlett, "A Nonstop Kernal," in proc. Eighth Symposium on Operating System Principles (Protokoll zum Achten Symposion über Betriebssystem-Grundlagen), S. 22–29, Dez. 1981). Mehrere weitere kommerzielle fehlertolerante Systeme wurden in den 1980er Jahren eingeführt (0. Serlin, "Fault-Tolerant Systems in Commercial Applications" (Fehlertolerante Systeme in kommerziellen Anwendungen), Computer, S. 19–30, August 1984). Heutige kommerzielle, fehlertolerante Systeme umfassen Multiprozessoren mit verteiltem Speicher, transaktionsbasierte Systeme mit gemeinsam benutztem Speicher, fehlertolerante Systeme mit "pair and spare"-Hardware (siehe R. Freiburghouse, "Making Processing Fallsafe" [Störungssichere Datenverarbeitung], Mini-micro Systems, S. 255–264, Mai 1982; auch das U.S.-Patent Nr. 4.907.228 ist ein Beispiel für diese 'Pair and spare'-Technik und für das transaktionsbasierte System mit gemeinsam benutztem Speicher), und dreifach redundante Systeme wie zum Beispiel das "Integrity"-Rechnersystem, welches von der in Cupertino in Kalifornien beheimateten Gesellschaft Tandem Computers Incorporated, der Abtretungsempfängerin dieser Anmeldung und der darin offenbarten Erfindung, hergestellt wird.

[0006] Die meisten Anwendungen kommerzieller fehlertoleranter Computer fallen in die Kategorie der Online-Transaktionsverarbeitung. Finanzinstitute sind für elektronische Geldüberweisungen, für die Steuerung von Geldautomaten und für Börsenhandelssysteme auf eine hohe Verfügbarkeit angewiesen. Die Hersteller von Waren verwenden fehlertolerante Rechner zur automatisierten Fabriksteuerung, zur Lagerverwaltung und für Online-Dokumentenzugriffssysteme. Als weitere Anwendungen von fehlertoleranten Rechner sind unter anderem Reservierungssysteme, Behörden-Datenbanken, Wettsysteme und Telekommunikationssysteme zu nennen.

[0007] Die Anbieter fehlertoleranter Rechner sind bestrebt, sowohl eine erhöhte Systemverfügbarkeit und einen kontinuierlichen Rechenbetrieb, als auch die Korrektheit der Daten selbst bei Auftreten von Fehlern zu gewährleisten. Je nach verwendeter Systemarchitektur läuft die auf dem System ausgeführte Anwendungssoftware (d. h. die "Prozesse") entweder trotz der aufgetretenen Fehler weiter, oder aber die Prozesse werden bei Auftreten eines Fehlers automatisch ab einem rezenten Sicherungspunkt neu gestartet. Manche fehlertoleranten Systeme sind mit einer ausreichenden Anzahl redundanter Komponenten versehen, die es dem System ermöglichen, sich unter Umgehung einer fehlerhaften Komponente neu zu konfigurieren, wobei jedoch die in dem fehlerhaften Modul ausgeführten Prozesse verloren gehen. Die Anbieter kommerzieller, fehlertoleranter Systeme sind mittlerweile dazu übergegangen, den Begriff der Fehlertoleranz in einem prozessor- und plattenübergreifenden Zusammenhang zu sehen. Um weitreichende Verbesserungen im Bereich der Zuverlässigkeit zu erzielen, muss auf sämtliche Fehlerquellen eingegangen werden, einschliesslich Netzteile, Lüfter und Verbindungen zwischen den Modulen.

[0008] Die "Nonstop"- und "Integrity"-Architekturen, welche von Tandem Computers Incorporated hergestellt werden (beide jeweils ausführlich in dem U.S.-Patent Nr. 4.228.496 und in den U.S.-Patenten Nr. 5.146.589 und 4.965.717 dargestellt, welche alle an den Abtretungsempfänger dieser Anmeldung abgetreten worden sind; Nonstop und Integrity sind eingetragene Warenzeichen von Tandem Computers Incorporated) stellen

zwei aktuelle Ansätze im Bereich der kommerziellen, fehlertoleranten Datenverarbeitung dar. Das NonStop-System, das in seinen Grundzügen in dem oben bezeichneten U.S.-Patent Nr. 4.278.496 bedient sich einer Architektur, welche Mehrprozessorsysteme verwendet, die darauf ausgelegt sind, den Betrieb ungeachtet des Ausfalls einer einzelnen Hardware-Komponente fortzusetzen. Im Normalbetrieb verwendet jedes Prozessorsystem seine Hauptkomponenten eher unabhängig und gleichzeitig als in Form von "Hot Backups". Die Nonstop-Systemarchitektur kann aus bis zu **16** Prozessorsystemen bestehen, welche über einen Bus zum Datenaustausch zwischen Prozessoren miteinander verbunden sind. Jedes Prozessorsystem hat seinen eigenen Speicher, welcher eine Kopie des nachrichtenbasierten Betriebssystems enthält. Jedes Prozessorsystem steuert einen oder mehrere Eingabe/Ausgabebusse (E/A-Busse). Durch die Verwendung von Dual-Port-E/A-Controllern und -Geräten lassen sich mehrfache Übertragungswege zu jedem Gerät einrichten. Externe Speicherungen (in Bezug auf das Prozessorsystem), wie zum Beispiel Plattenspeicherungen, können gespiegelt werden, um stets über permanente, redundante Datenspeicherungen zu verfügen.

[0009] Bei dieser Architektur ist jedes Systemmodul mit fehlererkennender Hardware ausgestattet, wodurch ein "schnellfehlender" Betrieb gewährleistet wird: der Betrieb wird bei Auftreten eines Fehlers angehalten, um ein Übergreifen des Fehlers auf andere Module zu verhindern. Fehler werden zum Beispiel durch Paritätsprüfung, Duplikatvergleich und Fehlererkennungscodes erkannt. Die Fehlererkennung fällt primär in den Aufgabenbereich der Hardware, während es sich bei der Fehlerkorrektur um eine Softwareaufgabe handelt.

[0010] Weiterhin kann im Rahmen der NonStop-Multiprozessorarchitektur die Anwendungssoftware ("Prozess") auf dem System unter dem Betriebssystem als "Prozesspaare" laufen und einen Primärprozess und einen Sicherungsprozess beinhalten. Der Primärprozess läuft auf einem der Multiprozessoren, während der Datensicherungsprozess auf einem anderen Prozessor läuft. Der Datensicherungsprozess befindet sich gewöhnlich im Ruhezustand, aktualisiert seinen Zustand jedoch periodisch in Reaktion auf Sicherungspunkt-Nachrichten, welche vom Primärprozess übermittelt werden. Der Inhalt einer Sicherungspunkt-Nachricht kann die Form einer vollständigen Zustandsaktualisierung annehmen, oder auch nur die seit der vorangegangenen Sicherungspunkt-Nachricht vorgenommenen Änderungen umfassen. Ursprünglich wurden Sicherungspunkte in Anwendungsprogrammen manuell gesetzt, mittlerweile läuft der Anwendungscode jedoch meist unter einer Transaktionsverarbeitungssoftware, welche eine Korrektur mittels einer Kombination aus Sicherungspunkten und zweiphasigen Transaktions-Übergabeprotokollen gewährleistet.

[0011] Der Nachrichtenverkehr zwischen Prozessoren beinhaltet bei der Nonstop-Architektur von Tandem das durch einen jeden Prozessor periodisch erfolgende Aussenden einer für alle Prozessoren des Systems, einschliesslich des sendenden Prozessors, bestimmten "Unter Spannung"-Nachricht, wodurch die anderen Prozessoren informiert werden, dass der sendende Prozessor noch in Betrieb ist. Fällt ein Prozessor aus, so wird dieser Ausfall dadurch angekündigt und identifizierbar, dass die periodische "Unter Spannung"-Nachricht des ausgefallenen Prozessors ausbleibt. In Reaktion darauf weist das Betriebssystem die entsprechenden Datensicherungsprozesse an, ab dem letzten Sicherungspunkt mit der Abarbeitung als Primärprozess zu beginnen. Es können Datensicherungsprozesse in einem anderen Prozessor gestartet werden, oder der Prozess kann ohne Backup fortgesetzt werden, bis die Hardware wieder repariert ist. Ein Beispiel für diese Technik findet sich im U.S.-Patent Nr. 4.817.091.

[0012] Jeder E/A-Controller wird von einem der beiden Prozessoren verwaltet, mit denen er verbunden ist. Die Verwaltung des Controllers wird periodisch zwischen den beiden Prozessoren hin- und hergeschaltet. Fällt der verwaltende Prozessor aus, so geht die Eigentümerschaft des Controllers automatisch an den anderen Prozessor über. Fällt der Controller aus, so wird der Datenzugriff durch einen anderen Controller aufrechterhalten.

[0013] Über die erzielte Hardware-Fehlertoleranz hinaus bieten die Prozessorpaare der oben beschriebenen Architektur auch ein gewisses Ausmass an Software-Fehlertoleranz. Fällt ein Prozessor aufgrund eines Softwarefehlers aus, so ist der Datensicherungs-Prozessor häufig in der Lage, die Datenverarbeitung unbeeinträchtigt von dem betreffenden Fehler erfolgreich fortzusetzen. Die Softwareumgebung in dem Datensicherungs-Prozessor verfügt charakteristischerweise über andersgeartete Warteschlangenlängen, Tabellengrössen und Prozessmischungen. Da die meisten der von den Software-Qualitätssicherungstests nicht entdeckten Softwarefehler Grenzbedingungen betreffen, die von selten vorkommenden Daten abhängig sind, können die Datensicherungsprozesse in vielen Fällen erfolgreich abgeschlossen werden.

[0014] Im Gegensatz zu der oben beschriebenen Architektur stellt das Integrity-System ein Beispiel für einen anderen Ansatz der fehlertoleranten Datenverarbeitung dar. Das 1990 eingeführte 'Integrity' wurde für eine Standardversion des Unix-Betriebssystems entworfen ("Unix" ist ein eingetragenes Warenzeichen der in Delaware beheimateten Firma Unix Systems Laboratories, Inc.). In Systemen, in denen Kompatibilität ein Hauptziel darstellt, muss die Wahl logischerweise auf hardwaremässige Fehlerkorrektur fallen, da nur wenige Modifikationen an der Software erforderlich sind. Die Prozessoren und lokalen Speicher werden unter Verwendung eines TMR-Verfahrens (dreifache Redundanz) konfiguriert. Alle Prozessoren arbeiten dabei dieselbe Codeströme ab, die einzelnen Module sind jedoch, was ihre Taktgabe betrifft, unabhängig voneinander, um eine Fehlertoleranz bei den Taktgabeschaltungen zu ermöglichen. Die Ausführung der drei Ströme ist asynchron

wobei es zu wechselseitigen Verschiebungen über mehrere Taktintervalle hinweg kommen kann. Die Ströme werden periodisch und bei jedem Zugriff auf den globalen Speicher resynchronisiert. Auf dem TMR-Controller-Board befindliche Wähler erkennen und maskieren Fehler in einem Prozessormodul. Der Speicher ist zwischen dem lokalen Speicher auf den in dreifacher Ausführung vorhandenen Prozessorbaugruppen und dem globalen Speicher auf den doppelt vorhandenen TMR-Controller-Baugruppen partitioniert. Die verdoppelten Abschnitte des Systems verwenden zur Fehlererkennung selbstprüfende Techniken. Jeder globale Speicher verfügt über zwei Ports und ist über eine Schnittstelle mit den Prozessoren und mit den E/A-Prozessoren (EAPs) verbunden. Standard VME-Peripheriegeräte-Controller sind durch ein Busschnittstellenmodul (BSM) an ein Buspaar angeschlossen. Fällt ein EAP aus, so kann eine Software unter Verwendung der BSMs die Steuerung aller Controller auf den verbleibenden EAP umschalten. Gespiegelte Plattenspeichereinheiten können mit zwei verschiedenen VME-Controllern verbunden werden.

[0015] Bei dem Integrity-System werden sämtliche Hardwarefehler durch die redundante Hardware verdeckt. Die Komponenten werden nach Abschluss der Reparatur on-line wiedereingebaut.

[0016] Die US 4059736 beschreibt ein duales Testsystem zur Überwachung von Nachrichtengeräten, welches über ein Paar von Prozessoren verfügt, das den Betrieb eines Paars von Schaltnetzwerken überwacht. Die Prozessoren arbeiten im Gleichschritt miteinander und geben Befehle an Peripherieeinheiten aus, um diese Einheiten auf Ihre korrekte Funktion hin zu überprüfen. Die Prozessoren führen eine Gegenprüfung durch, um ihre jeweiligen Ausgabedaten miteinander zu vergleichen. Die Integritätsprüfung der Daten während der Übertragung zu den Peripherie-Einheiten erfolgt durch die Erkennung von Paritätsfehlern.

[0017] Die EP 0306209 beschreibt ein synchrones Dualprozessorsystem, bei dem Dualprozessoren unter Verwendung von Vernetzungseinheiten kreuzgekoppelt werden, welche Vergleiche von übertragenen Daten anstellen, um Nichtübereinstimmungen zwischen den von den Dualprozessoren ausgegebenen Daten zu erkennen.

[0018] Die vorangegangenen Beispiele veranschaulichen gegenwärtige Ansätze, die darauf abzielen, Fehlertoleranz in Datenverarbeitungssysteme zu integrieren. Ansätze, bei denen die Fehlerkorrektur über die Software erfolgen, benötigen weniger redundante Hardware und bieten bis zu einem gewissen Mass die Möglichkeit einer Software-Fehlertoleranz. Hardware-Ansätze verwenden eine zusätzliche Hardware-Redundanz, um eine vollständige Kompatibilität mit Standard-Betriebssystemen zu ermöglichen und um Anwendungen in transparenter Weise auszuführen, die auf anderen Systemen entwickelt worden sind.

[0019] Die weiter oben beschriebenen Systeme bieten somit fehlertolerante Datenverarbeitung entweder über Hardware(z. B. funktionellfehlend, unter Verwendung von Redundanz) oder über Software-Techniken (z. B. schnellfehlend, mit Fehlerkorrektur über die Software und unter Verwendung von Hardware mit hoher Datenintegrität). Von keinem der beschriebenen Systeme ist jedoch anzunehmen, dass es in der Lage ist, eine fehlertolerante Datenverarbeitung anzubieten, welche sowohl Hardwareansätze (funktionellfehlend) als auch Softwareansätze (schnellfehlend) in einem einzigen Datenverarbeitungssystem vereint.

[0020] Verwendet werden Computersysteme wie die weiter oben beschriebenen häufig für den elektronischen Geschäftsverkehr: elektronischer Datenaustausch (EDI) und globale Nachrichtenvermittlung. Die Anforderungen, welche heute an diese Art des elektronischen Geschäftsverkehrs gestellt werden, machen aufgrund der steigenden Benutzeranzahl und der zunehmend komplexer werdenden Nachrichten eine immer grössere Leistungsfähigkeit in puncto Datendurchsatz erforderlich. Zum Beispiel Nur-Text-Kommunikation per E-Mail, das meistgenutzte Leistungsmerkmal im Internet, wächst Jahr für Jahr in beträchtlichem Ausmass. Das Internet wird zunehmend dazu verwendet, um Bild-, Ton- und Videodateien zu übertragen. Sprachspeicher- und Weiterleitungsdienste sind dabei, in alle Anwendungsbereiche vorzudringen und Desktop-Videokonferenzsysteme und Video-Nachrichtenvermittlungssysteme gewinnen in bestimmten Organisationen an Akzeptanz. Mit jedem Nachrichtenvermittlungstyp wird ein fortschreitend höherer Datendurchsatz erforderlich.

[0021] In solchen Umgebungen werden parallele Architekturen verwendet, welche durch verschiedene Kommunikationsnetzwerke wie etwa lokale Netze (LANs) und dergleichen miteinander verbunden sind.

[0022] Eine Hauptanforderung an eine Server-Architektur ist die Fähigkeit, grosse Datenmengen zu übertragen. Der Server muss über eine grosse Bandbreite verfügen, welche skalierbar ist, so dass in dem Mass, in dem sich das Datenvolumen erhöht und die Übertragungen komplexer werden, zusätzliche Durchgangsleistungskapazität hinzugefügt werden kann.

[0023] Busarchitekturen begrenzen die für eine jede Systemkomponente verfügbare Bandbreite. Je grösser die Anzahl der mit dem Bus verbundenen Komponenten ist, umso geringer wird die für jede einzelne Komponente verfügbare Bandbreite.

[0024] Darüber hinaus ist eine prompte Systemantwort ein Vorteil für sämtliche Anwendungen und eine Notwendigkeit für interaktive Anwendungen, sie erfordert eine sehr geringe Latenz, worunter ein Mass für die Zeitdauer zu verstehen ist, die benötigt wird, um Daten von der Quelle zum Ziel zu übertragen. Die mit der Antwortzeit in enger Verbindung stehende Latenz wirkt sich auf das Serviceniveau und die Mitarbeiterproduktivität aus.

[0025] Die vorliegende Erfindung schafft ein Verfahren zum fehlertoleranten Betrieb eines Paares von Pro-

zessoreinheiten nach Anspruch 1, und ein Computersystem nach Anspruch 8.

[0026] Grob gesprochen beinhaltet die vorliegende Erfindung ein Verarbeitungssystem, welches sich aus mehreren Unterverarbeitungssystemen zusammensetzt. Jedes Unterverarbeitungssystem verfügt als Hauptverarbeitungselement über eine zentrale Verarbeitungseinheit (CPU), welche ihrerseits ein synchron im Gleichschritt laufendes Prozessorpaar umfasst, das jeden Befehl eines Befehlsstroms gleichzeitig ausführt. Jedes der Unterverarbeitungssysteme beinhaltet weiterhin ein Eingabe/Ausgabe (E/A)-System für das systemweite Netz zur Bereitstellung von redundanten Übertragungswegen zwischen verschiedenen Komponenten des grösseren Verarbeitungssystems, einschliesslich einer CPU und verschiedener Peripheriegeräte (z. B. Massespeichereinheiten, Drucker und dergleichen) eines Unterverarbeitungssystems, sowie zwischen den Subprozessoren, aus denen sich das grössere, Gesamtverarbeitungssystem zusammensetzen kann. Die Kommunikation zwischen jeder beliebigen Komponente des Verarbeitungssystems (z. B. einer CPU und einer anderen CPU, oder einer CPU und irgendeinem Peripheriegerät, ungeachtet dessen, zu welchem Unterverarbeitungssystem dieses gehören mag) wird durch die Bildung und Übertragung von paketierten Nachrichten durchgeführt, welche von der sendenden Komponente bzw. Ausgangskomponente (z. B. einer CPU) zu einem Zielelement (z. B. einem Peripheriegerät) geleitet werden, und zwar mittels einer systemweiten Netzstruktur bestehend aus einer Reihe von Routerelementen, die über eine aus einer Mehrzahl von Verbindungs-Links bestehende (im folgenden "TNet" genannte) Busstruktur untereinander verbunden sind. Die Routerelemente sind für die Auswahl der geeigneten bzw. verfügbaren Übertragungswege von einer sendenden Komponente des Verarbeitungssystems zu einer Zielkomponente verantwortlich, basierend auf in dem Datenpaket enthaltenen Informationen. Somit versorgt die Routing-Fähigkeit der Routerelemente das E/A-System der CPUs mit einem Signalübertragungsweg zu den Peripheriegeräten, ermöglicht jedoch gleichzeitig auch die Kommunikation zwischen Prozessoren.

[0027] Wie weiter oben erwähnt, weist das erfindungsgemässe Verarbeitungssystem einen entsprechenden Aufbau auf, um durch eine schnellfehlende und eine funktionell fehlende Arbeitsweise einen fehlertoleranten Betrieb zu schaffen. Eine schnellfehlende Arbeitsweise wird durch das Anbringen von Möglichkeiten zur Fehlerprüfung an strategischen Punkten des Systems erreicht. So hat zum Beispiel jede CPU Möglichkeiten zur Fehlerprüfung an zahlreichen Stellen in den verschiedenen Datenwegen zwischen den (im Gleichschritt betriebenen) Prozessorelementen der CPU und dem ihr zugeordneten Speicher. Insbesondere bei dem Verarbeitungssystem gemäss der vorliegenden Erfindung erfolgt die Fehlerprüfung an einer Schnittstelle und in einer Weise, welche nur eine geringe Auswirkung auf die Leistung haben. Systeme nach dem Stand der Technik implementieren Fehlerprüfverfahren typischerweise insofern, als Prozessorpaare betrieben werden und der Daten- und Befehlsstrom zwischen den Prozessoren und einem Cachespeicher geprüft (verglichen) werden. Diese Technik der Fehlerprüfung neigte dazu, zu Verzögerungen bei den Zugriffszeiten zu führen. Ausserdem konnten bei dieser Art der Fehlerprüfung keine gegebenenfalls verfügbaren Standardbauteile verwendet werden (z. B. Prozessor/Cachespeicher-Kombinationen auf einem einzelnen Halbleiterchip oder -Modul). Die erfindungsgemässe Fehlerprüfung der Prozessoren erfolgt an Stellen mit niedrigerer Betriebsgeschwindigkeit, wie etwa an den Hauptspeicher- und E/A-Schnittstellen, welche mit niedrigeren Geschwindigkeiten betrieben werden als die Prozessor-Cachespeicher-Schnittstelle. Darüber hinaus wird die Fehlerprüfung an Orten durchgeführt, die es ermöglichen, Fehler aufzufinden, welche in den Prozessoren, in ihrem Cachespeicher und in den E/A- und Speicherschnittstellen auftreten können. Dies ermöglicht Vereinfachungen beim Entwurf der Speicher- und E/A-Schnittstellen, da keine Paritätsprüfungen und keine sonstigen Datenintegritätsprüfungen erforderlich sind.

[0028] Die Fehlerprüfung des Kommunikationsflusses zwischen den Komponenten des Verarbeitungssystems erfolgt durch die Hinzufügung einer zyklischen Redundanzprüfung (CRC-Prüfung) zu den zwischen den Elementen des Systems übermittelten Nachrichtenpaketen. Der CRC eines jeden Nachrichtenpakets wird nicht nur am Bestimmungsort der Nachricht geprüft, sondern auch während die Nachricht zu ihrem Bestimmungsort unterwegs ist, und zwar von jedem Routerelement, welches dazu verwendet wird, das Datenpaket von seinem Ursprungsort zu seinem Bestimmungsort zu leiten. Entdeckt ein Routerelement, dass ein Nachrichtenpaket einen nicht korrekten CRC aufweist, so wird das Nachrichtenpaket als solches gekennzeichnet und einem Wartungs-Diagnose-System gemeldet. Dieses Merkmal stellt ein brauchbares Werkzeug zur Fehlerisolierung dar. Eine derartige Verwendung des CRC ist dazu geeignet, Nachrichtenpakete über den gesamten Datenweg hinweg zu schützen, da die Routerelemente den CRC weder modifizieren noch regenerieren, während das Datenpaket an durch sie hindurchgeleitet wird. Der CRC eines jeden Datenpakets wird an jeder Routerkreuzung einer Prüfung unterzogen. Ein Befehlssymbol-"This packet Good" (TPG) (Paket gut) bzw. "This Packet Bad" (TPB) (Paket schlecht)-wird zu jedem Paket hinzugefügt. Ein Wartungs-Diagnose-Prozessor kann diese Informationen verwenden, um ein Link oder Router-Element zu isolieren, welches einen Fehler eingeschleust hat, selbst wenn der Fehler nur vorübergehend auftrat.

[0029] Die Routerelemente sind mit einer Mehrzahl von ausgestattet, an denen Nachrichten empfangen und versandt werden können. Als solche eignen sie sich gut zum Einsatz in einer Vielzahl von Topologien, so dass bei Auftreten eines Fehlers alternative Übertragungswege für die Kommunikation zwischen zwei beliebigen

Elementen eines Verarbeitungssystems (z. B. zwischen einer CPU und einem E/A-Gerät) eingerichtet werden können, wodurch ein fehlertolerantes System geschaffen wird. Zusätzlich dazu beinhaltet die Router-Logik die Fähigkeit, basierend auf dem Router-Eingangsport, an dem ein Nachrichtenpaket empfangen wird, und auf dem Bestimmungsort des Nachrichtenpakets, gewisse Ports als Ausgangsports zu deaktivieren. Ein Router, der ein Nachrichtenpaket empfängt, welches eine Zieladresse enthält, die einen für dieses Nachrichtenpaket ungültigen Port als Ausgangsport des Routers angibt, scheidet das Nachrichtenpaket aus und benachrichtigt das Wartungs-Diagnose-System. Durch einen wohlüberlegten Einsatz dieses Merkmals kann ein Nachrichtenpaket davor bewahrt werden, in eine endlose Schleife zu geraten, und kann verzögert bzw. verhindert werden, dass andere Nachrichtenpakete in eine solche Schleife geraten (z. B. durch die Schaffung einer weiter unten erläuterten "Systemblockierungs"-Bedingung).

[0030] Die CPUs eines Verarbeitungssystems sind in der Lage, in einer von zwei Grundbetriebsarten zu arbeiten: einem "Simplexbetrieb", in welchem jede CPU (eines CPU-Paars) unabhängig von der anderen betrieben wird, oder einem "Duplexbetrieb", in welchem CPU-Paare jeweils synchron im Gleichschritt laufend betrieben werden. Der Simplexbetrieb schafft die Fähigkeit der Rekonfiguration und des Neustarts nach Eintreten von Fehlern, die durch fehlerprüfende Hardware entdeckt worden sind (vgl. das U.S.-Patent Nr. 4.228.496, in welchem ein Mehrfachverarbeitungssystem abgehandelt wird, bei welchem jeder Prozessor die Fähigkeit besitzt, die Betriebsfähigkeit jener Prozessoren zu überprüfen, die auf derselben Ebene arbeiten, und die Rechentätigkeit eines Prozessors zu übernehmen, von dem festgestellt bzw. vermutet wird, dass er ausgefallen ist). Beim Duplexbetrieb führen die gepaarten CPUs beide einen identischen Befehlsstrom aus, wobei jede CPU des Paars jeden Befehl des Befehlsstroms im wesentlichen zur gleichen Zeit ausführt.

[0031] Der Duplexbetrieb schafft eine fehlertolerante Plattform für weniger robuste Betriebssystems (z. B. das UNIX-Betriebssystem). Das erfindungsgemässe Verarbeitungssystem mit den gepaarten, im Gleichschritt betriebenen CPUs ist so aufgebaut, dass Fehler in vielen Fällen verborgen werden, und zwar primär durch die Hardware (z. B. Fortsetzung des Betriebs trotz Vorhandensein eines Fehlers).

[0032] Wird das Verarbeitungssystem im Duplexbetrieb ausgeführt, so verwendet jede CPU das E/A-System, um auf jedes beliebige Peripheriegerät des Verarbeitungssystems zuzugreifen, und zwar ungeachtet dessen, welchem (der beiden oder mehreren) Subprozessorsystemen das betreffende Peripheriegerät scheinbar zugeordnet ist. Ausserdem werden im Duplexbetrieb Nachrichtenpakete, die an ein CPU-Paar zu liefern sind, von dem E/A-System im wesentlichen zur selben Zeit an beide CPUs des CPU-Paars übermittelt, um den synchronen, im Gleichschritt laufenden Betrieb des CPU-Paars aufrechtzuerhalten. Somit wird durch den erfinderischen Hauptaspekt der Erfindung ein Duplexbetrieb geschaffen, der die Fähigkeit beinhaltet, zu gewährleisten, dass beide CPUs eines im Gleichschritt laufenden CPU-Paars E/A-Nachrichtenpakete zur selben Zeit in derselben Weise empfangen In diesem Sinn ist jedes mit einer CPU eines duplexbetriebenen CPU-Paars verbundene Routerelement mit beiden CPU-Elementen des CPU-Paars verbunden. Jeder so verbundene Router, welcher eine für eine CPU bestimmte Nachricht (welche entweder von einem Peripheriegerät wie z. B. einer Massenspeichereinheit oder einer Verarbeitungseinheit kommen kann) empfängt, kopiert die Nachricht und sendet sie an beide CPUs des CPU-Paars unter Verwendung von Synchronisationsverfahren, die eine Aufrechterhaltung des Synchronbetriebs der CPUs sicherstellen. In der Tat wird das im Duplexbetrieb ausgeführte CPU-Paar von dem E/A-System und von anderen im Duplexbetrieb laufenden CPU-Paaren als eine einzelne CPU angesehen. Somit wird das E/A-System, welches Elemente aus allen Unterverarbeitungssystemen beinhaltet, von dem im Duplexbetrieb laufenden CPU-Paar als ein homogenes System betrachtet, innerhalb welchem auf jedes Peripheriegerät zugegriffen werden kann.

[0033] Ein weiteres wichtiges Neuheitsmerkmal der Erfindung besteht darin, dass die Vielseitigkeit der Routerelemente es ermöglicht, dass Cluster von im Duplexbetrieb laufenden Subsystem-Paaren untereinander ein Multiprozessorsystem bilden, in welchem die CPU eines jeden Prozessors sich in Wirklichkeit aus einem Paar von synchron im Gleichschritt laufenden CPUs zusammensetzt.

[0034] Noch ein anderer wichtiger Aspekt der vorliegenden Erfindung besteht darin, dass Interrupts, welche von einem E/A-Element herrühren, in derselben Weise an die CPU (bzw. bei Duplexbetrieb an das CPU-Paar) übertragen werden, wie dies auch bei jeder anderen Informationsübertragung der Fall ist: nämlich als Datenpakete. Dies bringt zahlreiche Vorteile mit sich: Interrupts können-wie alle anderen E/A-Nachrichtenpakete auch-durch CRC geschützt werden.

[0035] Ausserdem wird dadurch das Erfordernis nach zusätzlichen Signalleitungen umgangen, die ansonsten speziell zur gleichzeitigen Interruptmeldung an beide CPUs benötigt würden; durch die Übermittlung von Interrups über das Nachrichtenpaketsystem kann sichergestellt werden, dass diese, ebenso wie E/A-Nachrichtenpakete, in synchronisierter Weise bei den im Duplexbetrieb laufenden CPUs eintreffen. Interrupt-Nachrichtenpakete enthalten Informationen bezüglich der Ursache für das Interrupt, wodurch das zeitaufwendige Lesen der das Interrupt ausgebenden Geräte durch die CPU(s) umgangen werden kann, wie es derzeit zur Ermittlung der Interruptursache gehandhabt wird. Ausserdem können die Routerelemente, wie bereits weiter oben angedeutet, mehrfache Pfade für die Übermittlung der Interrupt-Pakete bereitstellen, wodurch die Fehlertoleranzfähigkeit des Systems erhöht wird. Darüber hinaus kann durch die Verwendung desselben Nachrichtenübermitt-

lungssystems zur Übertragung von Daten zwischen E/A-Einheiten und den CPUs und zur Übertragung von Interrupts an die CPUs eine geordnete Reihenfolge von E/A-Informationen und Interrupts bewahrt werden; das heisst ein E/A-Gerät wartet bis eine E/A-Information vollständig übertragen ist, bevor eine Interrupt-Nachricht gesendet wird.

[0036] Ein weiterer Aspekt des hier beschriebenen Systems liegt in der Implementierung einer Technik zur Validierung des Zugriffs auf den Speicher einer jeden CPU. Das Verarbeitungssystem erlaubt einen Zugriff auf den Speicher einer jeden CPU seitens irgendeines anderen Elements des Systems (d. h. durch andere CPUs und Peripheriegeräte). Aufgrund dieses Umstands muss ein Verfahren zum Schutz gegen unbeabsichtigten und/oder unerlaubten Zugriff geschaffen werden. Gemäss diesem Aspekt der Erfindung wird von jeder CPU eine Zugriffsvalidierungs- und - translationstabelle (ZVT) aufrechterhalten, welche Einträge für jede CPU-externe Quelle enthält, der Zugriff auf den Speicher dieser CPU gewährt wird. Jeder dieser ZVT-Tabelleneinträge beinhaltet Informationen betreffend den erlaubten Zugriffstyp (z. B. eine Speicherschreiberlaubnis) und die Angabe des Ortes im Speicher, an dem dieser Zugriff erlaubt ist. Die Erstellung von Nachrichtenpaketen, die durch das E/A-System geleitet werden, erfolgt, wie weiter oben bereits angedeutet, unter Verwendung von Informationen zur Beschreibung des Ursprungselements des Nachrichtenpakets, des Bestimmungsortes des Nachrichtenpakets, des Inhalts des Nachrichtenpakets (z. B. am Bestimmungsort zu schreibende Daten, oder eine Anforderung von am Bestimmungsort auszulesenden Daten) und dergleichen. Darüber hinaus, dass es den Routerelementen ermöglicht wird, das Nachrichtenpaket rasch an seinen Endbestimmungsort zu leiten, verwendet die empfangende CPU die Informationen dazu, um auf die ZVT-Tabelle zuzugreifen und in dieser den sich auf die Quelle des Nachrichtenpakets beziehenden Eintrag auszuwählen und um zu prüfen, ob ein Zugriff erlaubt ist und, wenn ja, welcher Zugriffstyp und welchen Ort die empfangende CPU auswählt, um die Adresse abzubilden (d. h. zu übersetzen). Auf diese Weise wird der Speicher einer jeden CPU gegen fehlgeleitete Zugriffe geschützt. Die ZVT-Tabelle wird auch dazu herangezogen, Interrupts an die CPU durchzustellen.

[0037] Durch die ZVT-Tabelle wird gewährleistet, dass der Speicher einer CPU nicht durch ein fehlerhaftes E/A-Gerät zerstört wird. Zugriffsrechte können von einem Speicher gewährt werden, der sich grössenmässig über einen Bereich von einem Byte bis zu einer ganzen Reihe von Seiten erstreckt. Die Fehlerbegrenzung ist besonders wichtig im E/A-Bereich, weil die Systemanbieter von Systemen gewöhnlich viel weniger Einflussmöglichkeiten auf die Qualität der Hardware und Software von Drittanbietern haben. Probleme können auf ein einziges E/A-Gerät bzw. einen Controller anstatt auf das gesamte E/A-System eingegrenzt werden.

[0038] Ein weiterer Aspekt der Erfindung betrifft die Technik, die von einer CPU verwendet wird, um Daten an das E/A-System zu übertragen. Gemäss diesem Aspekt der Erfindung ist in jeder CPU eine Blockübertragungsmaschine vorgesehen, um Übertragungen von Eingabe/Ausgabeinformationen zwischen einer CPU und jeglichen anderen Komponenten des Verarbeitungssystems abzuwickeln. Dadurch werden die einzelnen Prozessoreinheiten der CPU von. den eher anspruchslosen Routineaufgaben enthoben, die darin bestehen, Informationen aus dem Speicher zu holen und in das TNet-Netzwerk zu stellen oder vom Netzwerk kommende Informationen in Empfang zu nehmen. Die Prozessoreinheit der CPU beschränkt sich darauf, Datenstrukturen im Speicher zusammenzusetzen, welche die zu sendenden Daten zusammen mit anderen Informationen enthalten, wie zum Beispiel dem gewünschten Bestimmungsort, der Datenmenge und, für den Fall, dass eine Antwort erforderlich ist, dem Ort im Speicher, an welchem die Antwort nach ihrer Inempfangnahme gespeichert werden soll. Sobald die Prozessoreinheit mit der Zusammensetzung der Datenstruktur fertig ist, wird die Blockübertragungsmaschine benachrichtigt, damit diese übernimmt und das Senden der Daten in Form von Nachrichtenpaketen einleitet. Ist eine Antwort zu erwarten, so erstellt die Blockübertragungsmaschine die für die Behandlung der Antwort erforderliche Struktur, einschliesslich des Ortes im Speicher, an dem die Antwort abzulegen ist. Sofern eine Antwort empfangen wird, wird diese an den identifizierten, erwarteten Speicherort geleitet und wird die Prozessoreinheit benachrichtigt, dass die Antwort empfangen wurde.

[0039] Weitere Aspekte und Merkmale der gegenständlichen Erfindung werden für einschlägig ausgebildete Fachleute bei der Lektüre der nachfolgenden, detaillierten Beschreibung der Erfindung offenkundig, welche zusammen mit den beiliegenden Zeichnungen betrachtet werden sollte.

[0040] **Fig.** 1A veranschaulicht ein gemäss den Ausführungen des vorliegenden Patents entworfenes Verarbeitungssystem, und die **Fig.** 1B und 1C veranschaulichen zwei Alternativkonfigurationen des Verarbeitungssystems aus **Fig.** 1A, und zwar unter Verwendung von Clustern bzw. Anordnungen des Verarbeitungssystems aus **Fig.** 1A;

[0041] **Fig.** 2 veranschaulicht in vereinfachter Form als Blockdiagramm die zentrale Verarbeitungseinheit (CPU), welche Teil eines jeden Subprozessorsystems aus den **Fig.** 1A–1C ist;

[0042] die **Fig.** 3A–3C und 4A–4C veranschaulichen den Aufbau der verschiedenen Nachrichtenpakete, die dazu verwendet werden, um Informationen wie zum Beispiel Eingabe/Ausgabe-Daten über das in **Fig.** 2 gezeigte Bereichsnetzwerk-E/A-System zu übermitteln;

[0043] **Fig.** 5 veranschaulicht die Schnittstelleneinheit, welche Teil der CPUs aus **Fig.** 2 ist, um den Prozessor und den Speicher an das E/A-System des Bereichsnetzwerks anzuschliessen;

[0044] **Fig.** 6 ist ein Blockdiagramm, welches einen Abschnitt des Paketempfängers der Schnittstelleneinheit aus **Fig.** 5 veranschaulicht;
[0045] **Fig.** 7A veranschaulicht in Diagrammform das Taktsynchronisierungs-FIFO (TS-FIFO), welches von dem Paketempfänger des Paketemfangssegments aus **Fig.** 6 verwendet wird;
[0046] **Fig.** 7B ist ein Blockdiagramm eines Entwurfs des in **Fig.** 7A gezeigten Taktsynchronisierungs-FIFO-Aufbaus;
[0047] **Fig.** 8 veranschaulicht die Querverbindungen für zum Zweck der Fehlerprüfung von den beiden Schnittstelleneinheiten einer CPU abgehende Übertragungen;
[0048] **Fig.** 9 ein codiertes (8B bis 9B) Daten/Befehls-Symbol;
[0049] **Fig.** 10 veranschaulicht das Verfahren und den Aufbau, welche von der Schnittstelleneinheit aus **Fig.** 5 dazu benutzt werden, um Daten, die zu den Datenfehlerprüfungs-Speichersteuerungen übertragen werden, einer Gegenkontrolle auf Fehler hin zu unterziehen;
[0050] **Fig.** 11 ist eine als Blockdiagramm ausgeführte Darstellung der Implementierung der Zugriffsvalidierungsund -übersetzungstabelle (ZVÜ-Tabelle), die dazu verwendet wird, um andere (CPU-externe) Komponenten des Verarbeitungssystems zu prüfen und diesen Lese- und/oder Schreibzugriff auf den Speicher einer CPU aus **Fig.** 2 zu gewähren;
[0051] **Fig.** 12 ist ein Blockdiagramm das in Diagrammform die Zusammensetzung einer Adresse veranschaulicht, welche dazu verwendet wird, um auf einen ZVÜ-Tabelleneintrag zuzugreifen;
[0052] die **Fig.** 13A, 13B und 13C veranschaulichen das jeweilige Aussehen von ZVÜ-Tabelleneinträgen für normale Anforderungen und für Interruptanforderungen;
[0053] **Fig.** 14A veranschaulicht die Logik zum Freigeben von Interruptanforderungen an Warteschlangen im Speicher und an die Prozessoreinheiten der CPU aus **Fig.** 2;
[0054] **Fig.** 14B veranschaulicht den Prozess, der verwendet wird, um eine Speicheradresse für einen Warteschlangeneintrag zu bilden;
[0055] **Fig.** 15 ist ein Blockdiagramm, das die Datenausgabe-Grundelemente veranschaulicht, welche von einer Prozessoreinheit in dem Speicher der CPU aus **Fig.** 2 gebildet werden und welche Daten enthalten, die über die in den **Fig.** 1A–1C gezeigten Bereichs-E/A-Netzwerke zu senden sind, wobei darüber hinaus auch die Blockübertragungsmaschineneinheit (BÜM) der Schnittstelleneinheit aus **Fig.** 5 veranschaulicht wird, welche betrieben wird, um auf die Datenausgabe-Grundelemente zuzugreifen, und zwar im Hinblick auf deren Übertragung an das Bereichs-E/A-Netzwerk durch das in **Fig.** 7 gezeigte Paketübertragungssegment;
[0056] **Fig.** 16 veranschaulicht den Aufbau des 72-Bit-Datenpfades, welcher zum Teil durch ein zwischen dem Speicher einer CPU aus **Fig.** 2 und seiner Schnittstelleneinheit befindliches Paar von Speichersteuerungen gebildet wird, um aus dem Speicher auf **72** Datenbits zuzugreifen, welche zwei an zwei aufeinanderfolgenden, geradzahligen Adressen befindliche 32-Bit-Wörter, auf die gleichzeitig zugegriffen wird, sowie **8** Prüfbits beinhalten;
[0057] **Fig.** 17 ist ein vereinfachtes Blockdiagramm einer der beiden in **Fig.** 2 dargestellten Speichersteuerungen, in welchem ein serieller Zugriff auf diese durch einen Online-Zugangspunkt veranschaulicht wird;
[0058] **Fig.** 18 veranschaulicht in vereinfachter Form die Ablaufsteuereinheit des Paars von Speichersteuerungen aus **Fig.** 2 sowie die Technik, die verwendet wird, um beide zum Zweck der Fehlerprüfung gegenzuprüfen;
[0059] **Fig.** 19A veranschaulicht in Form eines vereinfachten Blockdiagramms die Routereinheit, welche bei den Bereichs-Eingabe/Ausgabe-Netzwerken der in den **Fig.** 1A–1C gezeigten Verarbeitungssysteme zum Einsatz kommen;
[0060] **Fig.** 19B veranschaulicht einen Vergleich zwischen zwei Port-Eingangsdaten der Routereinheit aus **Fig.** 19A;
[0061] **Fig.** 20A ist ein Blockdiagramm des Aufbaus von einem der sechs Eingabeports der in **Fig.** 19A gezeigten Routereinheit;
[0062] **Fig.** 20B ist ein Blockdiagramm der Synchronisierungslogik, die verwendet wird, um Befehls-/Daten-Symbole zu validieren, welche an einem Eingabeport der Routereinheit aus **Fig.** 19A empfangen werden,
[0063] **Fig.** 21A ist ein Blockdiagramm, in welchem die an dem in **Fig.** 20A gezeigten Eingabeport vorhandene Zielport-Auswahllogik veranschaulicht wird;
[0064] **Fig.** 21B ist ein Entscheidungsdiagramm in welchem die Routing-Entscheidungen veranschaulicht werden, die von der in **Fig.** 21A dargestellten Zielport-Auswahllogik getroffen werden;-{}-
[0065] **Fig.** 21C ist ein Blockdiagramm der algorithmischen Routing-Logik, welche ein Teil der in **Fig.** 21A dargestellten Zielport-Auswahllogik ist;
[0066] **Fig.** 22 ist eine als Blockdiagramm ausgebildete Veranschaulichung des Aufbaus von einem der sechs Ausgabeports der in **Fig.** 19A gezeigten Routereinheit;
[0067] **Fig.** 23 ist eine Veranschaulichung des Verfahrens, das verwendet wird, um identische Informationen an ein synchron im Duplexbetrieb laufendes Paar von CPUs aus **Fig.** 2 zu übermitteln, wenn das Verarbeitungssystem sich in im Gleichschritt laufenden (Duplex-)Betrieb befindet, und zwar unter Verwendung eines

Paars von FIFOs aus **Fig.** 7A (eines für jede CPU);
[0068] **Fig.** 24 ist ein vereinfachtes Blockdiagramm zur Veranschaulichung des Taktgebersystems eines jeden der Unterverarbeitungssysteme aus **Fig.** 1A–1C zu Erzeugung der Mehrzahl von Taktsignalen, die zum Betreiben der verschiedenen Elemente dieses Unterverarbeitungssystems verwendet werden;
[0069] **Fig.** 25 veranschaulicht die Topologie, welche zum Zusammenzuschalten der Taktgebersysteme von gepaarten Unterverarbeitungssystemen verwendet wird, um die verschiedenen Taktsignale des Paars von Unterverarbeitungssystemen untereinander zu synchronisieren;
[0070] die **Fig.** 26A und 26B veranschaulichen eine Taktsteuerlogik zum Halten einer konstanten FIFO-Geschwindigkeit, die dazu verwendet wird, das Taktsynchronisierungs-FIFO aus **Fig.** 8 oder 20 entsprechend zu steuern, wenn eine Situation eintritt, in der die beiden Takte, die dazu dienen, Symbole in die Warteschlange des FIFO einzureihen und daraus zu entnehmen, sich beträchtlich voneinander unterscheiden;
[0071] **Fig.** 27 ist ein Taktdiagramm, das die Betriebsweise der Taktsteuerlogik zum Halten einer konstanten Geschwindigkeit aus den **Fig.** 26A und 26B;
[0072] **Fig.** 28 veranschaulicht den Aufbau des Online-Zugangspunkts (OLZP), der dazu verwendet wird, Zugriff zu dem Instandhaltungsprozessor (IP) für die verschiedenen Elemente des Systems aus **Fig.** 1A (bzw. jener aus den **Fig.** 1B oder 1C) zu erhalten, um diese Elemente zu konfigurieren;
[0073] **Fig.** 29 veranschaulicht einen Abschnitt des Systemspeichers und zeigt Cachespeicherblockgrenzen; und
[0074] die **Fig.** 30A und 30B veranschaulichen die Soft-Flag-Logik, die verwendet wird, um asymmetrische Variablen zwischen den CPUs von gepaarten, im Duplex-Betriebsmodus laufenden Unterverarbeitungssystemen zu befördern;
[0075] **Fig.** 31A zeigt ein Flussdiagramm, und **Fig.** 31B veranschaulicht einen Abschnitt des Synchronisierungstakts (SYNC TKT), die beide dazu verwendet werden, die Taktsynchronisierungs-FIFOs der CPUs und die Router des Verarbeitungssystems aus **Fig.** 1A, die Informationen voneinander empfangen, rückzusetzen und zu synchronisieren;
[0076] **Fig.** 32 ist ein Flussdiagramm, das in groben Umrissen das Verfahren veranschaulicht, das verwendet wird, um Divergenzen zwischen zwei im Duplexbetrieb arbeitenden CPUs zu erkennen und handzuhaben;
[0077] die **Fig.** 33A–33D veranschaulichen in allgemeiner Weise das Verfahren, das verwendet wird, um eine der CPUs des in **Fig.** 1A gezeigten Verarbeitungssystems in im Gleichschritt laufenden Duplexbetrieb mit der anderen der CPUs zu bringen, ohne den Betrieb des Verarbeitungssystems in messbarer Weise anzuhalten; und
[0078] **Fig.** 34 veranschaulicht eine kostenreduzierte Architektur, in welche Lehren der Erfindung einfliessen; und
[0079] **Fig.** 35 veranschaulicht die Betriebsweise einer Barrieren-Transaktion, um einen Übertragungsweg zwischen einer CPU aus **Fig.** 1A (oder **Fig.** 1B, 1C) und einem Eingabe-Ausgabe-Gerät zu prüfen und zu verifizieren.

Überblick:

[0080] Betrachtet man nun die Figuren genauer und widmet man sich fürs erste hauptsächlich der **Fig.** 1A, so findet man dort dargestellt ein Datenverarbeitungssystem, bezeichnet mit dem Bezugszeichen **10**, welches gemäss den verschiedenen Lehren der vorliegenden Erfindung aufgebaut ist. Wie **Fig.** 1A zeigt, umfasst das Datenverarbeitungssystem **10** zwei Subprozessorsysteme **10A** und **10B**, von denen beide im wesentlichen im Aufbau und in der Funktion identisch sind. Demgemäss sei an hier darauf verwiesen, dass sofern keine anderslautenden Angaben gemacht werden, eine Beschreibung eines beliebigen der Subprozessorsysteme **10** in der gleichen Weise auch auf jedes andere Subprozessorsystem **10** zutrifft.
[0081] Um nun bei **Fig.** 1A fortzusetzen, sei somit erläutert, dass jedes der Subprozessorsysteme **10A**, **10B** eine zentrale Verarbeitungseinheit (CPU) **12**, einen Router **14** und eine Mehrzahl von Eingabe-Ausgabe-(E/A)-Paketschnittstellen **16** beinhaltet, von denen eine jede ihrerseits über einen nativen Eingabe-Ausgabe-Bus (NEA) an eine Anzahl (n) von E/A-Geräten **17** angeschlossen ist. An mindestens eine der E/A-Paketschnittstellen **16** ist auch ein Instandhaltungsprozessor (IP) **18** angeschlossen.
[0082] Der IP **18A**, **18B** eines jeden. Subprozessorsystems **10A**, **10B** ist mit jedem der Elemente dieses Subprozessorsystems über einen IEEE 1149.1 Prüfbus **19A**, **19B** (in Durchsicht dargestellt in **Fig.** 1A; in den **Fig.** 1B und 1C aus Gründen der Klarheit nicht dargestellt) und über eine Online-Zugangspunkt-Schnittstelle (OLZP-Schnittstelle) verbunden, welche für jedes Element Register enthält, die von dem IP **18** verwendet werden, um Zustands- und Steuerinformationen zwischen dem Element und dem IP **18** auszutauschen. Der IP **18** kann auch, wie in **Fig.** 1A veranschaulicht, mit den CPUs **12** kommunizieren, indem er Nachrichtenpakete erstellt und versendet. (Eigentlich ist es die E/A-Paket-Schnittstelle **16**, die als Reaktion auf eine diesbezügliche Anfrage seitens des IP **18** ein Paket erstellt und sendet.)
[0083] Die CPU **12**, der Router **14** und die E/A-Paket-Schnittstellen **16** sind durch "TNet"-Links **L** miteinander

verbunden, welche bidirektionalen Datenaustausch ermöglichen. Jedes TNet-Link **L** umfasst zwei unidirektionale **10**-Bit-Sublink-Busse. Jedes TNet-Sublink überträgt **9** Datenbits und ein begleitendes Taktsignal. Wie **Fig.** 1A weiterhin veranschaulicht, verbinden die TNet-Links L auch die Subprozessorsysteme **10A** und **10B** untereinander, wodurch sowohl der Zugriff eines jeden Subprozessorsystems **10** auf die E/A-Geräte des jeweils anderen als auch als auch eine Kommunikation zwischen den CPUs gewährleistet wird. Wie noch zu erläutern sein wird, kann einer jeden CPU **12** des Verarbeitungssystems **10** Zugriff auf den Speicher jeder anderen CPU **12** gewährt werden, wenngleich auch dieser Zugriff einer Validierung bedarf, was einen wichtigen Aspekt der Erfindung darstellt. Auf gewissermassen ähnliche Art und Weise kann auch von den Peripheriegeräten aus auf den Speicher der CPU **12** zugegriffen werden, und zwar üblicherweise als Ergebnis einer von einer CPU eingeleiteten Operation. Diese Zugriffe bedürfen ebenfalls der Validierung, um die Korruption des Speichers einer CPU **12** durch ein nicht korrekt funktionierendes Peripheriegerät **17** zu verhindern.

[0084] Vorzugsweise sind die Subprozessorsysteme **10A**/**10B** gepaart, wie in **Fig.** 1A (sowie in den weiter unten besprochenen **Fig.** 1B und 1C) veranschaulicht, und jedes Subprozessorsystem-Paar **10A**/**10B** (d. h. umfassend eine CPU **12**, zumindest einen Router **14** und zumindest eine E/A-Paket-Schnittstelle **16** mit zugeordneten E/A-Geräten).

[0085] Jede CPU **12** verfügt über zwei E/A-Ports, einen X-Port und einen Y-Port, an denen Nachrichtenpakete übermittelt und/oder empfangen werden. Der X-Port einer CPU **12** (z. B. der CPU **12A**) ist über ein TNet-Link L an einen Router (**14A**) des entsprechenden Subprozessorsystems (z. B. **10A**) angeschlossen. Umgekehrt ist der Y-Port der CPU (**12A**) an den Router (**14B**) des dazugehörigen Subprozessorsystems (**10B**) angeschlossen. Diese letztere Verbindung stellt nicht nur einen Übertragungsweg für den Zugriff einer CPU (**12A**) auf die E/A-Geräte des anderen Subprozessorsystems (**10B**) bereit sondern auch auf die CPU (**12B**) dieses Systems zum Zweck der Kommunikation zwischen den CPUs.

[0086] Der Informationsaustausch zwischen einem beliebigen Element des Verarbeitungssystems **10** und einem beliebigen anderen Element (z. B. der CPU **12A** eines Subprozessorsystems **10A**) des Systems und einem beliebigen anderen Element des Systems (z. B. einem E/A-Gerät, das einer E/A-Paket-Schnittstelle **16B** des Subprozessorsystems **10B** zugeordnet ist), erfolgt über Nachrichten-"Pakete". Jedes Nachrichtenpaket besteht aus einer Anzahl von **9**-Bit-Symbolen, die Daten enthalten können oder ein Steuersymbol sein können. Nachrichtenpakete werden synchron auf den TNet-Links **L** in bitparalleler, symbolserieller Weise übertragen und von einem Sendeschritt-Takt begleitet, welcher von der das Nachrichtenpaket übertragenden Komponente bereitgestellt wird. Die Takte zwischen den kommunizierenden Elementen (d. h. zwischen Sender und Empfänger) können in einer von zwei Betriebsarten betrieben werden: einem "frequenznahen" Modus oder einem "frequenzsynchronisierten" Modus.

[0087] Beim Betrieb in frequenznahem Modus sind die von dem sendenden Element und dem empfangenden Element verwendeten Taktsignale voneinander getrennt und werden lokal generiert, wenngleich sie innerhalb eines vorbestimmten Toleranzbereichs im wesentlichen auf derselben Frequenz gehalten werden. Aus diesem Grund wurde ein einzigartiges Verfahren entwickelt, das es erlaubt, die Symbole beim Empfänger unter Verwendung einer (weiter unten eingehender beschriebenen) Taktsynchronisierungs-FIFO-Speicherstruktur (TS-FIFO) zu empfangen. Das TS-FIFO wird betrieben, um jedes Auseinanderdriften, zu dem es zwischen dem Taktsignal des Empfängers und des Überträgers eines Nachrichtenpakets infolge des frequenznahen Betriebs kommen kann, auszugleichen. Der frequenznahe Betriebsmodus kommt zum Einsatz bei der Übertragung von Symbolen von einem Router **14** zu einem anderen, oder zwischen einem Router **14** und einer I/O-Paket-Schnittstelle **16**, oder zwischen Routern **14** und CPUs **12**, welche im (weiter unten beschriebenen) Simplex-Betrieb laufen.

[0088] Frequenzsynchronisierter Betrieb bedeutet lediglich folgendes: die Frequenzen der Taktsignale der Sende- und Empfangseinheiten sind synchronisiert, jedoch nicht unbedingt phasensynchronisiert. Frequenzsynchronisierte Taktsignale finden Verwendung, um Symbole zwischen den Routern **14A**, **14B** und den CPUs **12** gepaarter Subprozessorsysteme (z. B. der Subprozessorsysteme **10A**, **10B** aus **Fig.** 1A) zu übertragen. Da die Takte des übertragenden und des empfangenden Elements nicht phasenbezogen sind, wird hier wieder ein Taktsynchronisierungs-FIFO verwendet, auch wenn dieses in einem leicht veränderten Modus betrieben wird, verglichen mit jenem, das für den frequenznahen Betrieb verwendet wird.

[0089] Jeder Router **14** ist mit **6** bidirektionalen TNet-Ports, 0–5, ausgestattet, von denen ein jeder im wesentlichen identisch aufgebaut ist, allerdings mit einer Ausnahme: die beiden Ports (**4**, **5**), die für den Anschluss an eine CPU **12** verwendet werden, weisen eine etwas veränderte Struktur auf. Dieser Unterschied, wie noch genauer zu sehen sein wird, ist durch die Tatsache bedingt, dass Paare der Subprozessorsysteme **10** in einem synchron im Gleichschritt laufenden Modus, auch Duplexbetrieb genannt, betrieben werden können, wobei eine jede CPU **12** betrieben wird, um denselben Befehl aus demselben Befehlsstrom zur selben Zeit auszuführen. Beim Duplexbetrieb ist es wichtig, dass von einem beliebigen E/A-Gerät kommende, eingehende E/A-Signale praktisch zur selben Zeit an beide CPUs **12** übermittelt werden. So wird zum Beispiel ein Nachrichtenpaket, das an Port 3 des Routers **14A** empfangen wird, von dem Router **14A** dupliziert und von den Router-Ports 4, 5 weg übertragen, so dass dasselbe Symbol im wesentlichen zur selben Zeit an die CPUs **12** über-

mittelt wird. In dieser Weise können die Ports **4**, **5** von den anderen Ports 0–3 des Routers **14** abweichen.

[0090] **Fig.** 1A veranschaulicht ein weiteres Merkmal der Erfindung: eine Quervernetzung zwischen den beiden Subprozessorsystemen **10A**, **10B** durch die Verwendung von zusätzlichen Routern **14** (in **Fig.** 1A durch die Bezugszeichen **RX_1**, **RX_2**, **RY_1** und **RY_2** gekennzeichnet) . Wie **Fig.** 1 veranschaulicht, bilden die hinzugefügten Router **RX_1**, **RX_2**, **RY_1** und **RY_2** eine Quervernetzung zwischen den Subprozessoren **10A**, **108** (bzw., wie gezeigt, zwischen den jeweiligen "Seiten" X und Y), um sie an die E/A-Paket-Schnittstellen **16X**, **16Y** anzuschliessen. Die Quervernetzungs-Links zwischen den Routern RX_1 – RY_2 und RY_1 – RX_2 stellen den Quervernetzungspfad von einer Seite (X bzw. Y) zu der anderen bereit, und zwar ziemlich genau in derselben Weise, wie dies auch durch die Quervernetzungen Ly zwischen den CPUs **12A**, **12B** und den Routern **14B**, **14A** erfolgt. Die durch die Router **RX_1**, **RX_2**, **RY_1** und **RY**, bereitgestellte Quervernetzung ermöglichen es jedoch den (nicht gezeigten) E/A-Geräten, die an die E/A-Paket-Schnittstellen **16X**, **16Y** angeschlossen werden können, um zur einen Seite (X bzw. Y) hin oder zur anderen gelenkt werden zu können.

[0091] Wie in **Fig.** 1A gezeigt, stellen die Router RX, und RY_2 den E/A-Paket-Schnittstelleneinheiten 16x und 16y eine Schnittstelle mit zwei Ports bereit. Nun ist es natürlich klar, dass die F/A-Paket-Schnittstellen 16X, 16Y ihrerseits so aufgebaut sein könnten, dass sie von sich aus zwei Ports aufweisen, als Alternative zu der Quervernetzung über die von den Routern **RX_2** and **RY_2** gebildeten Dual-Port-Verbindungen, und dass diese Dual Ports an die Router **RX_1**, **RY_1** angeschlossen werden können.

[0092] Wie später noch klar werden wird, nachdem der Aufbau und der Entwurf der Router **14** verstanden worden ist, eignen diese sich dazu, in einer Weise verwendet zu werden, die es ermöglicht, die Konfiguration des Verarbeitungssystems **10** auf zusätzliche Subprozessorsysteme auszuweiten, wie in den **Fig.** 1B und 1C veranschaulicht wird. In **Fig.** 1B wird zum Beispiel ein Port eines jeden der Router **14A** und **14B** dazu verwendet, die entsprechenden Subprozessorsysteme **10A** und **10B** mit zusätzlichen Subprozessorsystemer. **10A*** und **10B*** zu verbinden und somit ein grösseres Verarbeitungssystem zu bilden, welches Gruppen des elementaren Verarbeitungssystems aus **Fig.** 1 umfasst.

[0093] In ähnlicher Weise wird das obige Konzept in **Fig.** 1C auf einen Cluster bestehend aus acht Subprozessorsystemen ausgeweitet, welcher die Subprozessorsystem-Paare 10A/10B, 10A'/10B', 10A''/10B'' und 10A'''/10B''' umfasst. Jedes der Subprozessorsysteme (z. B. Subprozessorsystem **10A**) hat seinerseits im wesentlichen dieselbe Basis-Minimalkonfiguration bestehend aus einer CPU **12**, einem Router **14**, und durch eine F/A-Paket-Schnittstelle **16** mit dem TNet verbundenen E/A[-Elementen], mit der Ausnahme, dass, wie in **Fig.** 1C gezeigt, die Subprozessorsysteme **10A** und **10B** jeweils zusätzliche Router **14C** und **14D** beinhalten, um den Cluster über die Subprozessorsysteme 10A*/10B* hinausgehend auf die Subprozessorsysteme 10A''/10B'' und 10A'''/10B''' zu erweitern. Wie **Fig.** 1C weiterhin veranschaulicht, können die unbenutzten Ports **4** und **5** der Router **14C** und **14D** dazu verwendet werden, um den Cluster sogar noch weiter zu erweitern.

[0094] Bedingt durch den Entwurf der Router **14** und durch das zur Leitung von Nachrichtenpaketen benutzte Verfahren kann bei wohlüberlegter Verwendung der Router **14** die Konfiguration der Topologie des Systems **10** entsprechend erfolgen, so dass jede CPU **12** des Verarbeitungssystems **10** aus **Fig.** 1C auf jede beliebige andere "Endeinheit" (z. B. eine CPU oder ein E/A-Gerät) eines jeden der anderen Subprozessorsysteme zugreifen kann. Zwei Pfade sind von jeder CPU **12** zu dem letzten Router **14** verfügbar, welcher die Verbindung zu der E/A-Paket-Schnittstelle **16** herstellt. So kann zum Beispiel der Zugriff der CPU **12B** des Subprozessorsystems **10B'** auf die E/A **16'''** des Subprozessorsystems **10A'''** über den Router **14B** (des Subprozessorsystems **10B'**), den Router **14D**, und den Router **14B** (des Subsystems **10A'''**) und, auf dem Weg über das Link LA, über den Router **14A** (Subsystem **10A'''**), ODER über den Router **14A** (des Subsystems **10A'**), den Router **14C** und den Router **14A** (Subprozessorsystem **10A'''**) erfolgen. In ähnlicher Weise hat die CPU **12A** des Subprozessorsystems **10A''** (über zwei Pfade) Lese- oder Schreibzugriff auf Speicherorte, welche in der CPU **12B** des Subprozessors enthalten sind. (Der Speicherzugriff seitens einer CPU **12** einer anderen Komponente des Verarbeitungssystems erfordert, wie noch genauer zu erläutern sein wird, dass die Komponente, die den Zugriff anfordert, über eine Berechtigung dazu verfügt. Diesbezüglich wird von jeder CPU **12** eine Tabelle aufrechterhalten mit Einträgen für jede Komponente, die berechtigt ist, auf den Speicher der betreffenden CPU zuzugreifen, wobei dieser Zugriff üblicherweise auf ausgewählte Speicherbereiche und hinsichtlich der Zugriffsart beschränkt ist. Ein solches Berechtigungserfordernis verhindert die Korruption von Speicherdaten einer CPU durch einen Falschzugriff.)

[0095] Die Topologie des in **Fig.** 1B gezeigten Verarbeitungssystems wird dadurch erreicht, dass der Port **1** der Router **14A**, **14B** und die zusätzlichen TNet-Links LA zum Verbindungsaufbau mit den Routern **14A'**, **14B'** der Subprozessorsysteme **10A'**, **10B'** verwendet werden. Die dadurch erzielte Topologie gewährleistet redundante Übertragungswege zwischen jeder beliebigen CPU **12** (**12A**, **12B**, **12A'**, **12B'**) und jeder E/A-Paket-Schnittstelle 16 des in **Fig.** 1B gezeigten Verarbeitungssystems **10**. So kann zum Beispiel der Zugriff seitens der CPU **12A#** des Subprozessorsystems **10A'** auf die E/A **16A** eines Subprozessorsystems **10A** über einen ersten Pfad erfolgen, welcher durch den Router **14A'** (ein Port 4, aus Port 3), den Router **14A** (ein Port 3, aus Port 0) und durch diesen zugeordnete Verbindungs-TNet-Links L gebildet wird. Fällt jedoch der Router **14A'** aus, so kann der zugriff der CPU **12A'** auf die E/A **16A** über den Pfad erfolgen, welcher durch den Router

14B' (ein Port 5, aus Port 3), den Router **14B** (ein Port 3, aus Port 1), das Link LA und den Router **14A** (ein Port 1, aus Port 0) erfolgen.

[0096] Es sei darauf hingewiesen, dass die Topologie aus **Fig.** 1B ebenfalls redundante Übertragungswege zwischen jedem beliebigen Paar von CPUs **12** des Systems **10** gewährleistet und somit eine Einrichtung zur fehlertoleranten Kommunikation zwischen CPUs bereitstellt.

[0097] **Fig.** 1C veranschaulicht eine Erweiterung der in **Fig.** 1B gezeigten Topologie. Durch das Zusammenschalten eines Ports eines jeden Routers **14** eines jeden Subprozessorpaars und durch das Verwenden zusätzlicher TNet-Links LA (in **Fig.** 1C durch gestrichelte Verbindungslinien dargestellt) zwischen den Ports **1** der Routen **14** (**14A"** und **14B"**) der Subprozessorsysteme **10A"**, **10B"** und **10A'"**, **10B'"** ergeben sich zwei getrennte, unabhängige Datenpfade zwischen jeder CPU **12** und jeder E/A-Paket-Schnittstelle **16**. Auf diese Weise bestehen von jeder Endeinheit (z. B. einer CPU **12** oder einer E/A-Paket-Schnittstelle **16**) mindestens zwei Übertragungswege zu jeder anderen Endeinheit.

[0098] Die Bereitstellung alternativer Zugriffspfade zwischen zwei beliebigen Endeinheiten (z. B. zwischen einer CPU **12** und jeder beliebigen anderen CPU **12**, oder zwischen jeder beliebigen CPU **12** und jeder beliebigen E/A-Paket-Schnittstelle **16** innerhalb des Systems **10**-**Fig.** 1C) stellt ein wichtiges Konzept dar. Der Ausfall eines Fehlerbereichs führt nicht zur Unterbrechung der Übertragungswege zwischen zwei beliebigen anderen der verbleibenden Fehlerbereiche. Ein Fehlerbereich könnte hier ein Subprozessorsystem (z. B. **10A**) sein. Angenommen das Subprozessorsystem **10A** würde aufgrund eines Defekts in der elektrischen Stromversorgung ausfallen, so hätte ohne das zwischen den Routern **14A'"** und 14B'" angebrachte, zusätzliche TNet-Link LA die CPU **12B** des Subprozessorsystems **10B** den Zugriff auf die E/A-Paket-Schnittstelle **16** (über den Routen **14A**, den Routen **14C**, den Routen **14A'"** zu der E/A-Paket-Schnittstelle **16'"**) verloren. Dank der zusätzlichen Verbindung LA zwischen den Routern **14A'"** und **14B**'" ist selbst bei Verlust des Routers **14A** (und des Routers **14C**), der durch den Verlust des Subprozessorsystems **10A** bedingt ist, ein Datenverkehr zwischen der CPU **12B** weiterhin möglich, und zwar auf dem Weg über den Router **14B**, den Router **14D**, den Router **14B'"**, die zusätzliche Verbindung LA zu dem Router **19A'"**, und schliesslich zu der E/A-Paket-Schnittstelle **16'"**.

CPU-Architektur:

[0099] Unter Bezugnahme auf **Fig.** 2 wird im folgenden die CPU **12A** im Detail veranschaulicht. Da die beiden CPUs **12A** und **12B** in Aufbau und Funktion im wesentlichen identisch sind, werden hier nur die Details der CPU **12A** beschrieben. Es versteht sich jedoch, dass, sofern keine anderslautenden Angaben gemacht werden, die Abhandlung der CPU **12A** auch auf die CPU **12B** anwendbar ist. Wie **Fig.** 2 zeigt, beinhaltet die CPU **12A** ein Paar von Prozessoreinheiten **20a**, **20b**, welche insofern für einen synchronen, im Gleichschritt laufenden Betrieb konfiguriert sind, als beide Prozessoreinheiten **20a**, **20b** im wesentlichen zum selben Zeitpunkt identische Befehle empfangen und ausführen und identische Daten- und Befehlsausgaben vornehmen. Jede der Prozessoreinheiten **20a** und **20b** ist über einen Bus **21** (**21a**, **21b**) mit einem entsprechenden Cachespeicher **22** verbunden. Der verwendete spezielle Typ der Prozessoreinheit könnte auch über ausreichend internen Cachespeicher verfügen, so dass der Cachespeicher **22** nicht benötigt würde. Alternativ dazu könnte der Cachespeicher **22** verwendet werden, um den allfällig vorhandenen, internen Cachespeicher der Prozessoreinheiten **20** zu ergänzen. Wie dem auch sei, wenn der Cachespeicher **22** verwendet wird, so ist der Bus **21** so aufgebaut, dass er 128 Bits Daten, 16 Bits Fehlerkorrekturcode(FKC)-Prüfbits zum Schutz der Daten, **25** Identifizierungsbits (für die Daten und den entsprechenden FKC), **3** die Identifizierungsbits abdeckenden Prüfbits, **22** Adressbits, **3** die Adresse abdeckende Paritätsbits und **7** Steuerbits.

[0100] Die Prozessoren **20a**, **20b** sind auch jeweils über einen getrennten 64-Bit Adress-/Datenbus **23** an die X- und Y-Schnittstelleneinheiten **24a**, **24b** angeschlossen. Falls gewünscht, könnten die über jeden Bus **23a**, **23b** übertragenen Adressen/Daten ebenfalls durch Paritätsbits geschützt werden, dies erhöht jedoch die Busbreite. (Vorzugsweise beinhalten die Prozessoren **20** ihrem Aufbau nach Mikroprozessoren vom Typ RISC R4000 wie sie von der MIPS Abteilung der in Santa Clara in Kalifornien ansässigen Gesellschaft Silicon Graphics, Inc. vertrieben werden.)

[0101] Die X- und Y-Schnittstelleneinheiten **24a**, **24b** werden betrieben, um Daten- und Steuersignale zwischen den Prozessoreinheiten **20a**, **20b** und einem Speichersystem der CPU **12A**, welches eine Speichersteuerung (SpSt) **26** (zusammengesetzt aus zwei SpSt-Hälften **26a** und **26b**) und eine dynamische Schreib-Lese-Speicherplatzanordnung **28** umfasst, zu übertragen. Die Schnittstelleneinheiten **24** stehen untereinander und mit den Speichersteuerungen **26a**, **26b** über einen 72-Bit Adress-/Steuerbus **25** in Verbindung. Wie noch genauer zu erläutern sein wird, werden von den Schnittstelleneinheiten **24** zwar 64-Bit-Daten-Doppelwörter (begleitet von 8 Bit Fehlerkorrekturcode) in den Speicher **28** geschrieben, eine Schnittstelleneinheit **24** steuert jedoch nur ein Wort (z. B. den Abschnitt der **32** wichtigsten Bits) des zu schreibenden Doppelworts, während die andere Schnittstelleneinheit **24** das andere Wort des Doppelworts (z. B. den am wenigsten bedeutenden 32-Bit-Abschnitt des Doppelworts) in den Speicher schreibt. Darüber hinaus führen die Schnittstelleneinheiten **24a**, **24b** zum Zweck der Fehlerkontrolle bei jeder Speicherbeschreibung eine Gegenkontrolle jener Daten, die

nicht von der einen Schnittstelleneinheit **24** geschrieben worden sind mit jenen, die von der anderen geschrieben worden sind, durch; bei Leseoperationen werden die über den Bus **25** geleiteten Adressen in derselben Weise gegengeprüft. Der spezielle Fehlerkorrekturcode, welcher zum Schutz sowohl der in den Cachespeicher **22** geschriebenen Daten als auch des (Haupt)Speichers **28** verwendet wird, ist herkömmlicher Natur und ermöglicht Einzelbitfehlerkorrektur und Doppelbitfehlererkennung.

[0102] Konzeptuell enthält jedes Doppelwort ein "ungerades" und ein "gerades" Wort. Eine der Speichersteuerungen **26** schreibt die ungeraden Wörter in den Speicher, während die andere die geraden Wörter schreibt. Darüber hinaus schreiben die Speichersteuerungen **26** zwei Doppelwörter gleichzeitig, zusammen mit dem 8-Bit-Fehlerkorrekturcode (FKC) für dieses Doppelwort. Ausserdem sind die FKC-Prüfbits entsprechend ausgebildet so dass sie nicht nur das Doppelwort abdecken, sondern auch die Adresse des Speicherorts, an welchem das Doppelwort geschrieben wird. Anlässlich eines späteren Zugriffs wird der FCK dazu verwendet, um Einzelbitfehler zu korrigieren und Doppelbitfehler zu erkennen, welche in den Daten aufgetreten sein können, wobei gleichzeitig auch sichergestellt wird, dass das Datenwort, auf das zugegriffen wird, der Adresse des Ortes entspricht, von welchem das Doppelwort gespeichert wurde.

[0103] Die Schnittstelleneinheiten **24a**, **24b** der CPU **12A** bilden den Schaltungsaufbau um jeweils die X- und Y-(I/O)-Ports der CPU **12A** zu bearbeiten. So steht etwa die X-Schnittstelleneinheit **24a** über das bidirektionale TNet-Link Lx mit einem Port des Routers **14A** des Prozessorsystems **10A** (**Fig.** 1A) in Verbindung, während das Y-Schnittstellenelement **24b** in ähnlicher Weise über das TNet-Link Ly mit dem Router **14B** des Prozessorsystems **10B** in Verbindung steht. Die X-Schnittstelleneinheit **24a** handhabt den gesamten E/A-Datenverkehr zwischen dem Router **14A** und der CPU **12A** des Subprozessorsystems **10A**. In gleicher Weise ist die Y-Schnittstelleneinheit **24b** für den gesamten E/A-Datenverkehr zwischen der CPU **12A** und dem Router **14B** des dazugehörigen Subprozessorsystems **10B** verantwortlich.

[0104] Das TNet-Link **Lx**, welches die X-Schnittstelleneinheit **24a** mit dem Router **14A** (**Fig.** 1) verbindet, umfasst, wie weiter oben erwähnt, zwei 10-Bit-Busse $\mathbf{30}_x$, $\mathbf{32}_y$, von denen ein jeder ein Taktsignal und **9** Datenbits überträgt. Der Bus $\mathbf{30}_x$ befördert Übertragungsdaten an den Router **14A**; der Bus $\mathbf{32}_x$ befördert von dem Router **14A** eingehende Daten. In ähnlicher Weise ist die Y-Schnittstelleneinheit **24b** mit dem Router **14B** (des Subprozessorsystems **10B**) mittels zweier 10-Bit-Busse verbunden: **30y** (für abgehende Übertragungen) und **32y** (für eingehende Übertragungen), welche zusammen das TNet-Link **Ly** bilden.

[0105] Die X- und Y-Schnittstelleneinheiten **24a**, **24b** werden synchron im Gleichschritt betrieben und führen im wesentlichen zur selben Zeit im wesentlichen dieselben Operationen aus. Somit werden, obwohl nur die X-Schnittstelleneinheit **24a** tatsächlich Daten auf den Bus $\mathbf{30}_x$ überträgt, dieselben Ausgabedaten auch von der Y-Schnittstelleneinheit **24b** erzeugt und zur Fehlerprüfung verwendet. Die Ausgabedaten der Y-Schnittstelleneinheit **24b** werden über ein Quervernetzungs-Link **34y** an die X-Schnittstelleneinheit **24a** angelegt, wo sie von der X-Schnittstelleneinheit **24a** empfangen werden und im Zuge einer Gegenprüfung mit denselben, von der X-Schnittstelleneinheit erzeugten Ausgabedaten verglichen werden. Auf diese Weise werden die abgehenden Daten am X-Port der CPU **12A** bereitgestellt und auf Fehler geprüft.

[0106] In derselben Weise werden die von dem Port der CPU **12A** übertragenen Ausgangsdaten geprüft. Die Ausgabedaten von der Y-Schnittstelleneinheit **24b** werden über einen 10-Bit-Bus **30y** an den Y-Port und weiterhin über ein 9-Bit-Querverbindungs-Zink **34y** an die X-Schnittstelleneinheit **24a** angelegt, wo sie im Zuge einer Gegenprüfung mit den von der X-Schnittstelleneinheit erzeugten Ausgabedaten verglichen werden.

[0107] Wie erwähnt, befinden sich die beiden Schnittstelleneinheiten **24a**, **24b** in synchronem, im Gleichschritt laufenden Betrieb zueinander, wobei eine jede im wesentlichen zur selben Zeit im wesentlichen dieselben Operationen ausführen. Aus diesem Grund müssen Daten, welche an dem X- und/oder Y-Ports der CPU **12A** empfangen werden, von beiden Schnittstelleneinheiten **24a**, **24b** empfangen werden, damit die beiden Schnittstelleneinheiten ihren im Gleichschritt laufenden Betrieb beibehalten. Daten, welche von einer Schnittstelleneinheit **24a**, **24b** empfangen werden, werden somit an die andere weitergeleitet, wie durch die gestrichelten Linien und die 9-Bit-Quervernetzungen $\mathbf{36}_x$ angedeutet (die eingehende Daten, welche an dem X-Port durch die X-Schnittstelleneinheit **24a** empfangen werden, an die Y-Schnittstelleneinheit **24b** übertragen) und $\mathbf{36}_y$ (die Daten, welche an dem Y-Port durch die Schnittstelleneinheit **24b** empfangen werden, an die X-Schnittstelleneinheit **24a** übertragen.

[0108] Bestimmte robustere Betriebssysteme sind mit einer fehlertoleranten sind mit einer fehlertoleranten Fähigkeit im Zusammenhang mit einem Multiprozessorsystem ausgestattet. Multiprozessorsysteme dieser Art stellen insofern eine fehlertolerante Umgebung bereit, als sie der Software die Möglichkeit geben, von der Hardware oder der Software entdeckte Fehler zu korrigieren. In dem U.S.-Patent Nr. 4.817.091 wird zum Beispiel ein Multiprozessorsystem abgehandelt, bei welchem jeder Prozessor in regelmässigen Abständen jeden Prozessor des Systems (einschliesslich seiner selbst) softwaregesteuert eine Nachricht sendet, um anzuzeigen, dass er nach wie vor in Betrieb ist. Jeder der Prozessoren wird neben seinen normalen Aufgaben auch als Datensicherungs-Prozessor für einen anderen der Prozessoren eingesetzt. Gesetzt den Fall, dass ein Datensicherungs-Prozessor von dem ihm zugeordneten Prozessor keine Anzeigenachricht empfängt, übernimmt er die Rechentätigkeit dieses ihm zugeordneten Prozessors (von dem angenommen wird, dass er ausgefallen

ist), neben der Durchführung seiner eigenen Aufgaben mit. Andere fehlertolerante Techniken, welche weniger robuste Software bzw. Betriebssysteme verwenden (d. h. ohne eigene Fähigkeit, erkannte Fehler zu korrigieren) werden mit Hardware und mit einer Logik entworfen, die betreiben werden, um erkannte Fehler zu korrigieren.

[0109] Die vorliegende Erfindung zielt darauf ab, eine Hardwareplattform für beide Softwaretypen bereitzustellen. Ist also ein robustes Betriebssystem verfügbar, so kann das Verarbeitungssystem **10** entsprechend konfiguriert werden, um in einem "Simplex"-Betrieb zu arbeiten, in welchem jede der CPUs **12A** und **12B** in unabhängiger Weise voneinander betrieben wird. Die CPUs **12** sind mit Schaltungsanordnungen zur Fehlerprüfung an strategischen Punkten innerhalb verschiedener CPU-interner Übertragungswege ausgeführt. Die Router **14** stellen eine Kommunikation zwischen Prozessoren her und ermöglichen den Datenaustausch zwischen den verschiedenen CPUs **12**, die innerhalb des Systems **10** miteinander verbunden sein können, und schaffen weiterhin einen Kommunikationsweg von jeder CPU des Systems zu jedem Gerät, das über die E/A-Paket-Schnittstelle **16** angesteuert werden kann. Wird ein Fehler erkannt, so wird die Verantwortung für die Korrektur dieses Fehlers in den meisten Fällen der Software überlassen.

[0110] Alternativ dazu bietet das Verarbeitungssystem **10** für weniger robuste Betriebssysteme und Software eine hardwarebasierte Fehlertoleranz, indem es entsprechend konfiguriert wird, um in einem "Duplex"-Betriebsmodus zu arbeiten, in welchem ein Paar von CPUs (z. B. die CPUs **12A**, **12B**) wie in **Fig.** 1A gezeigt, zusammengeschaltet werden, um in synchron, im Gleichschritt laufender Weise betrieben zu werden, sodass sie dieselben Befehle im wesentlichen zum selben Zeitpunkt ausführen. Dadurch wird jede CPU als Kontrolleinrichtung für die andere. Gesetzt den Fall, dass eine der CPUs **12** einen Fehler entwickelt, so kann sie gemäss ihrer "Schnellfehl"-Eigenschaft schnell versagen und heruntergefahren werden, noch bevor sich der Fehler ausbreiten und den Rest des Systems korrumpieren kann. Die andere CPU **12** setzt ihren Betrieb fort und übernimmt dabei die Aufgabe(n) von beiden. Der Duplex-Betriebsmodus erlaubt es somit dem Hardwaresystem, die Auswirkung des Fehlers zu verdecken.

[0111] Daten- und Steuersymbole werden zwischen den verschiedenen CPUs **12** und den E/A-Paket-Schnittstellen **16** in Form von Nachrichtenpaketen ausgetauscht, welche **9** Bit Daten- und Steuersymbole umfassen. Um den Entwurfe der CPU **12** zu vereinfachen, werden die Prozessoren **20** daran gehindert, direkt mit einer externen Einheit (z. B. einer anderen CPU **12** oder einem E/A-Gerät über die E/A-Paket-Schnittstelle **16**) zu kommunizieren. Vielmehr erstellt der Prozessor, wie noch genauer zu erläuten sein wird, eine Datenstruktur im Speicher und übergibt die Steuerung hierauf an die Schnittstelleneinheiten **24**. Jede Schnittstelleneinheit **24** beinhaltet eine Blockübertragungsmaschine (BÜM; **Fig.** 5), die entsprechend konfiguriert ist, um eine Art von direkter Speicherzugriffsfähigkeit (DMA) bereitzustellen, welche es erlaubt, auf die Datenstruktur(en) aus dem Speicher zuzugreifen und sie über den geeigneten X- bzw. Y-Port zum Datenaustausch mit dem Ziel gemäss einer in dem Datenpaket enthaltenen Information zu übertragen.

[0112] Der Entwurf des Verarbeitungssystems **10** ermöglicht es, dass ein Speicher **28** einer CPU von externen Quellen aus (z. B. der CPU **12B** oder einem E/A-Gerät) gelesen bzw. beschrieben werden kann. Aus diesem Grund muss darauf geachtet werden, sicherzustellen, dass eine von aussen erfolgende Benutzung eines Speichers **28** einer CPU **12** nur in Verbindung mit einer entsprechenden Berechtigung erfolgt. Der Zugriff auf den Speicher **28** ist daher durch einen Zugriffsvalidierungsmechanismus geschützt, welcher einen Zugriff erlaubt oder verhindert, und zwar nach Prüfung verschiedener Faktoren, wie etwa woher die Zugriffsanforderung kam, welche Art von Zugriff angefordert wurde, der Ort des angeforderten Zugriffs und dergleichen. Die Zugriffsvalidierung wird über eine Zugriffsvalidierungstabellen(ZVÜ)-Logik realisiert, welche weiter unten im Zuge der Abhandlung der **Fig.** 11–13 beschrieben wird.

[0113] Verschiedene Aspekte der Erfindung benutzen die Konfiguration der Daten- und Steuerpakete, die über die Router **14** zwischen den E/A-Paket-Schnittstellen **16** und den CPUs **12** übertragen werden. Demgemäss ist es von Vorteil, bevor mit der Beschreibung des Aufbaus des Verarbeitungssystems **10** fortgefahren wird, als erstes die Konfiguration der Daten- und Steuersymbole und -pakete zu verstehen, die auf den TNet-Links **L** übertragen und durch die Router **14** gelenkt werden.

Paketkonfigurationen:

[0114] Vier Grundtypen von Nachrichtenpaketen werden verwendet, um Steuersymbole und Daten zwischen den CPUs **12** und Peripheriegeräten **17** eines Systems auszutauschen. Die **Fig.** 3A–3D veranschaulichen den Aufbau eines Nachrichtenpaket-Typs (**Fig.** 3A), zusammen mit einer Aufgliederung der Felder dieses Pakets (**Fig.** 3B–3D); die **Fig.** 4A–4C veranschaulichen den Aufbau der anderen drei Pakettypen. Der Nachrichtenpaket-Typ, der verwendet wird, um Schreibdaten auf das TNet-Bereichsnetzwerk zu übertragen wird als HADC-Paket bezeichnet und ist in **Fig.** 3A veranschaulicht. Wie dargestellt, hat das HADC-Paket vier Felder: ein 8-Byte-Headerfeld, ein 4-Byte-Datenadressfeld, ein N-Byte-Datenfeld (wobei N vorzugsweise auf ein Maximum von **64** beschränkt ist, obwohl klar ist, dass auch grössere Datenmengen in einem einzelnen Paket bewegt werden können), und ein 4-Byte-'Zyklische Redundanzprüfung'-Feld (CRC-Feld).

[0115] Das in **Fig.** 3B im Detail veranschaulichte Headerfeld beinhaltet eine 3-Byte-Bestimmungsort-ID, in welcher der Endbestimmungsort des Nachrichtenpakets angegeben wird; eine 3-Byte-Ursprungsort-ID, in welcher der Ursprungsort bzw. das Ursprungselement des Nachrichtenpakets angegeben wird; den Transaktionstyp (z. B. ein Lese- oder Schreiboperation), sowie den Nachrichtenpaket-Typ (z. B. ob es sich um eine Datenanforderung oder um eine Antwort auf eine Datenanforderung handelt). Die Bestimmungsort-ID enthält vier Unterfelder: ein 14-Bit-Unterfeld, das eine Regions-ID enthält, um eine "Region" anzugeben, in welcher sich der Bestimmungsort der Nachricht befindet; ein 6-Bit-Unterfeld, das eine Geräte-ID enthält, in welcher das im Bereich der spezifizierten Region gelegene Bestimmungs-Gerät angegeben wird (z. B. ein Gerät **17**, eine CPU **12**, oder vielleicht ein IP **18**); ein Pfadauswahl(P)-Bit, das dazu verwendet wird, um aus zwei Pfaden auszuwählen; und 3 Bit, die für zukünftige Erweiterungen reserviert sind. In ähnlicher Weise hat die Ursprungsort-ID drei Unterfelder; eine 14-Bit-Regions-ID, die die Region des Senders spezifiziert; eine 6-Bit-Geräte-ID, die das sendende Gerät innerhalb dieser Region spezifiziert; ein 4-Bit-Typen-Unterfeld, das, wie bereits erwähnt, den Transaktionstyp kennzeichnet. Ausserdem spezifiziert das Steuerfeld die Datenmenge, welche in dem das Nachrichtenpaket begleitenden Datenfeld enthalten ist, und zwar durch Angabe der Anzahl an 9-Bit-Steuer/Daten"Symbolen". (Jedes Symbol ist ein 8-Bit-Datenbyte, codiert als 9-Bit-Menge zum Schutz vor Einzelbitfehlern, die dazu führen könnten, dass ein Datenbyte wie ein Steuersymbol aussehen würde, oder umgekehrt, wie weiter unten noch genauer zu erläutern sein wird.)

[0116] Die Regions- und Geräte-Felder der Bestimmungsortbzw. Ursprungsort-ID identifizieren den Bestimmungsort bzw. den Ursprungsort des Nachrichtenpakets kumulativ und eindeutig. Das als Pfadauswahlbit reservierte Bit dient dazu, die eine oder die andere von zwei "Seiten" X oder Y (wie in **Fig.** 1A veranschaulicht) als den Bestimmungsort des Nachrichtenpakets enthaltend zu kennzeichnen. Das Pfadauswahlbit wird weiter unten im Zusammenhang mit der Speicherzugriffsvalidierung (**Fig.** 11 und 12) und der Portauswahloperation des Routers (**Fig.** 21A) genauer erläutert. Die verbleibenden 3 Bits werden für eine allfällige zukünftige Erweiterung reserviert.

[0117] Das 4-Byte-Datenadressfeld wird eingehender in **Fig.** 3C veranschaulicht. Das Adressfeld kennzeichnet im Fall eines HADC-Pakets den virtuellen Speicherort für den Bestimmungsort an welchem die begleitenden N Datenbytes geschrieben werden. Wenn zum Beispiel das Ursprungselement des Nachrichtenpakets ein E/A-Gerät **17** ist, das Daten enthält, welche in den Speicher **28** einer CPU **12** zu schreiben sind, so enthält das Datenadressfeld ein Adresse, die den Speicherort im Speicher **28** kennzeichnet, in welchen die Daten zu schreiben sind. (Wie noch genauer zu erläutern sein wird, wird im Fall von CPUs die Datenadresse durch die ZVÜ-Logik (**Fig.** 11) in eine physische Adresse überführt, die tatsächlich dazu benutzt wird, um auf den Speicher **28** zuzugreifen. Die E/A-Paket-Schnittstellen 16 verfügen über ähnliche Validierungs- und Übersetzungsmechanismen.) Wenn das Adressfeld einen Speicherort einer CPU **12** bezeichnet, umfasst das Feld zwei Unterfelder: die **20** bedeutendsten Bits des Adressfeldes bilden eine 20-Bit-Speicherseitennummer; die verbleibenden **12** Bits bilden einen Offset in die Speicherseite. Die Seitenzahl wird von der ZVÜ-Logik (**Fig.** 11) als ein Index zu einer Tabelle mit Einträgen verwendet, die Validierungsinformationen enthalten.

[0118] Wie bereits angedeutet, dient das HADC-Nachrichtenpaket dazu, Schreibdaten zwischen Endeinheiten (z. B. CPU **12**) des Verarbeitungssystems **10** zu übertragen. Andere Nachrichtenpakete können allerdings aufgrund ihrer Funktion und Verwendung anders aufgebaut sein. So veranschaulicht etwa **Fig.** 4A ein HAC-Nachrichtenpaket, welches nur Header-, Adress- und CRC-Felder umfasst. Das HAC-Paket wird verwendet, um Lesedatenanforderungen an eine Systemkomponente zu übertragen (z. B. ein E/A-Gerät 17).

[0119] **Fig.** 4B veranschaulicht ein Nachrichtenpaket vom Typ HDC mit einem 8-Byte Headerfeld, einem N-Byte-Datenfeld (wobei N wiederum maximal **64** beträgt, obwohl es theoretisch für eine beliebige ganze Zahl stehen könnte), und einem 4-Byte-CRC-Feld. Das HDC-Nachrichtenpaket dient dazu, Antworten auf Leseanforderungen zu übertragen, welche die Rücksendung der angeforderten Daten beinhalten.

[0120] **Fig.** 4C veranschaulicht ein HC-Nachrichtenpaket, welches lediglich einen 8-Byte-header und eine 4-Byte-CRC umfasst. Das HC-Nachrichtenpaket wird dazu verwendet, eine Anforderung zum Schreiben von Daten zu bestätigen.

Schnittstelleneinheit:

[0121] Die X- und Y- Schnittstelleneinheiten **24** (d. h. **24a** und **24b**-**Fig.** 2) werden dazu betrieben, um drei Hauptfunktionen innerhalb der CPU **12** auszuführen: die Prozessoren **20** an den Speicher **28** anzuschliessen; einen E/A-Dienst bereitzustellen, der transparent für die Prozessoren betrieben wird, aber dennoch von diesen gesteuert wird; und von externen Quellen kommende Zugriffsanforderungen auf den Speicher **28** zu validieren.

[0122] Betrachtet man zuerst die Schnittstellenfunktion, so werden die X- und Y-Schnittstelleneinheiten **24a**, **24b** betrieben, um jeweils die Prozessoren **20a**, **20b** mit den Speichersteuerungen (SpSt **26a**, **26b**) und dem Speicher **28** zum Zweck des Lesens und Schreibens von Daten in Verbindung zu setzen, und zwar in einer Art und Weise, die eine schnellfehlende Prüfung der gelesenen/geschriebenen Daten beinhaltet. So kooperieren zum Beispiel im Zuge von Schreiboperationen die beiden Schnittstelleneinheiten **24a**, **24b**, um die zu schrei-

benden Daten einer Gegenkontrolle zu unterziehen, um deren Integrität sicherzustellen (wobei die Schnittstelleneinheiten **24** gleichzeitig betrieben werden), um einen Fehlerkorrekturcode (FKC) zu entwickeln, der-wie noch genauer zu erläutern sein wird-nicht nur die in den Speicher **28** geschriebenen Daten abdeckt, sondern auch die Speicheradresse des Speicherorts, an welchem diese Daten geschrieben werden, so dass bei einem späteren Abrufen (Lesen) der Daten nicht nur die korrekten Daten abgerufen werden, sondern auch davon ausgegangen werden kann, dass diese von der richtigen Adresse abgerufen worden sind.

[0123] Was den E/A-Zugriff betrifft, so sind die Prozessoren nicht mit der Fähigkeit ausgestattet, direkt mit den Eingabe-/Ausgabesystemen zu kommunizieren; sie müssen vielmehr Datenstrukturen in den Speicher **28** schreiben und dann die Steuerfunktion an die Schnittstelleneinheiten **24** abtreten, die eine Direkt-Speicherzugriffsoperation (DMA) ausführen, um diese Datenstrukturen abzurufen und sie zum Zweck der Übermittlung an den gewünschten Bestimmungsort auf das TNet zu übertragen. (Die Adresse des Bestimmungsorts wird dabei in der Datenstruktur selbst angegeben.)

[0124] Die dritte Funktion der X- und Y-Schnittstelleneinheiten **24**, die Validierung des Zugriffs auf den Speicher **28**, verwendet eine Zugriffsvalidierungsund -übersetzungstabelle (ZVÜ), welche von den Schnittstelleneinheiten aufrechterhalten werden. Die ZVÜ-Tabelle enthält eine Adresse für jede zugriffsberechtigte Systemkomponente (z. B. ein E/A-Gerät **17**, oder eine CPU **12**), sowie die erlaubte Zugriffsart und die physische Speicheradresse, an welcher ein Zugriff gestattet ist. Die Tabelle spielt weiterhin auch eine Rolle bei der Durchführung der Adressübersetzung, da es sich bei den in den eingehenden Nachrichtenpaketen enthaltenen Adressen um virtuelle Adressen handelt. Diese virtuellen Adressen werden von der Schnittstelleneinheit in physikalische Adressen übersetzt, welche von den Speichersteuerungseinheiten **26** zum Zweck des Zugriffs auf den Speicher **28** gelesen werden können.

[0125] In **Fig.** 5, auf welche im folgenden Bezug genommen wird, ist ein vereinfachtes Blockdiagramm der X-Schnittstelleneinheit **24a** der CPU **12A** veranschaulicht. Die ihr entsprechende Y-Schnittstelleneinheit **24b** (sowie auch die Schnittstelleneinheiten **24** der CPU **12B**, oder jeder anderen CPU **12**) ist im wesentlichen identisch aufgebaut. Demgemäss versteht es sich, dass die Beschreibung der Schnittstelleneinheit **24a** gleichermassen auch auf die anderen Schnittstelleneinheiten **24** des Verarbeitungssystems **10** anwendbar ist.

[0126] Wie in **Fig.** 5 veranschaulicht, beinhaltet die X-Schnittstelleneinheit **24a** eine Prozessorschnittstelle **60**, eine Speicherschnittstelle **70**, eine Interrupt-Logik **86**, ein Blockübertragungsmaschine (BÜM) **88**, eine Zugriffsvalidierungs- und -übersetzungslogik **90**, einen Paketüberträger **94** und einen Paketempfänger **96**.

Prozessorschnittstelle:

[0127] Die Prozessorschnittstelle **60** wickelt den Informationsfluss (Daten und Steuersignale) zwischen dem Prozessor **20a** und der X-Schnittstelleneinheit **24a** ab. Ein Prozessorbus **23**, welcher einen 64-Bit-Adress- und Datenbus (SysAD) **23a** und einen 9-Bit-Befehlsbus **23b** beinhaltet, verbindet den Prozessor **20a** und die Prozessorschnittstelle **60** miteinander. Während der SysAD-Bus **23a** Speicheradresse und -daten überträgt, und zwar in herkömmlicher, zeitlich verzahnter Form, überträgt der Befehlsbus **23b** Befehls- und Datenbezeichnerinformationen (SysBef), welche die Befehle, die im wesentlichen gleichzeitig auf dem SysAD-Bus **23a** übertragen werden. Die Prozessorschnittstelle **60** wird betrieben, um von der Prozessoreinheit **20a** ausgegebene Befehle zu interpretieren, um Lese-/Schreiboperationen in den Speicher oder in das Steuerregister der Prozessorschnittstelle weiterzugeben. Ausserdem enthält die Prozessorschnittstelle **60** einen Zwischenspeicher (nicht dargestellt) zum Puffern von Adressen und Daten, welche für den Zugriff auf den Speicher **28** (über die Speichersteuerungen **26**) erforderlich sind. Daten und Steuerinformationen, welche vom Speicher gelesen werden, werden auf ähnliche Weise auf ihrem Weg zur Prozessoreinheit **20a** gepuffert und verfügbar gemacht, sobald die Prozessoreinheit bereit ist, sie in Empfang zu nehmen. Weiterhin wird die Prozessorschnittstelle **60** betrieben, um die benötigten Interrupt-Signale für die X-Schnittstelleneinheit **24a** zu generieren.

[0128] Die Prozessorschnittstelle **60** ist über einen bidirektionalen 64-Bit-Prozessoradress/-datenbus **76** mit einer Speicherschnittstelle **70** und mit den Konfigurationsregistern **74** verbunden. Die Konfigurationsregister **74** sind eine symbolische Repräsentation der verschiedenen Steuerregister, die in anderen Komponenten der X-Schnittstelleneinheit **24a** enthalten sind, und werden im Zusammenhang mit der Abhandlung dieser speziellen Komponenten genauer erläutern. Bedingt durch die Tatsache, dass verschiedene der Konfigurationsregister **74** über andere Elemente der zur Realisierung der X-Schnittstelle **24a** verwendeten Logik verteilt sind, ist der Prozessor-Adress/-Datenbus **76** auch entsprechend verkoppelt, um ein Lesen von diesen Registern bzw. ein Beschreiben derselben zu ermöglichen, wobei dies allerdings in **Fig.** 5 nicht spezifisch veranschaulicht wird.

[0129] Die Konfigurationsregister **74** ermöglichen einen Lese-/Schreibzugriff seitens des Prozessors **20a**; sie ermöglichen eine "Personalisierung" der X-Schnittstelleneinheit. So bezeichnet zum Beispiel ein Register die Knotenadresse der CPU **12A**, welche verwendet wird, um die Ursprungsort-Adresse von Nachrichtenpaketen zu bilden, die von der CPU **12A** ausgehen; ein anderes, das nur mit Lesezugriff ausgestattet ist, enthält eine feste Identifikationsnummer der Schnittstelleneinheit **24**, und wiederum andere Register definieren Speicher-

bereiche, welche beispielsweise von verschiedenen Elementen verwendet werden können wie etwa von der BÜM **88** (an denen sich Datenstrukturen und BÜM-Befehls-/Steuerwörter befinden), von der Interrupt-Logik **86** (welche auf Interrupt-Warteschlangen verweisen, die Informationen über extern generierte und über Nachrichtenpakete empfangene Interrupts enthalten) oder von der ZVÜ-Logik **90**. Wieder andere Register werden zum Freigeben von Interrupts durch die Interrupt-Logik **86** verwendet. Viele der Register werden weiter unten im Zusammenhang mit der Abhandlung der logischen Komponenten (z. B. Interrupt-Logik **86**, ZVÜ-Logik **90**, usw.), in denen sie Verwendung finden, genauer beschrieben.

[0130] Die Speicherschnittstelle **70** verbindet die X-Schnittstelleneinheit **24a** mit den Speichersteuerungen **26** (und mit der Y-Schnittstelleneinheit **24b**; siehe **Fig.** 2) über einen Bus **25**, welcher zwei bidirektionale 36-Bit-Busse **25a**, **25b** beinhaltet. Die Speicherschnittstelle wird betrieben, um über die Priorität von Speicher-Zugriffsanforderungen zwischen der Prozessoreinheit **20**, dem BÜM **88** und der ZVÜ-Logik **90** zu entscheiden. Zusätzlich zu den Speicherzugriffen von der Prozessoreinheit **20a** kann auf den Speicher **28** auch von Komponenten des Verarbeitungssystems **10** zugegriffen werden, um zum Beispiel Daten von einer E/A-Einheit **17** zu speichern, die von der Prozessoreinheit **20a** zum Lesen angefordert werden, oder es kann auch auf den Speicher **28** zugegriffen werden, um E/A-Datenstrukturen zu lesen, die zuvor von der Prozessoreinheit in den Speicher gestellt worden sind. Da diese Zugriffe alle asynchron erfolgen, muss über ihre Priorität entschieden werden, und diese Entscheidungen werden von der Speicherschnittstelle **70** getroffen.

[0131] Informationen in Form von Daten und Befehlen, auf die von dem Speicher **28** aus zugegriffen wird, werden von der Speicherschnittstelle über einen Speicherauslesebus **82** an die Prozessorschnittstelle **60**, sowie an eine Interrupt-Logik **86**, eine Blockübertragungsmaschine (BMÜ) **88** und an eine Zugriffsvalidierungs- und -übersetzungslogik (ZVÜ-Logik) **90** übertragen. Wie weiter unten genauer beschrieben, werden die Daten mengenmässig in Form von Doppelwörtern in den Speicher **28** geschrieben. Während jedoch die Speicherschnittstellen **70** der X- und der Y-Schnittstelleneinheit **24a** und **24b** das (64 Bit)-Doppelwort formulieren und an den Bus **25** anlegen, ist jede Speicherschnittstelle **70** für das Schreiben von nur **32** Bit dieser 64-Bit-Doppelwortmenge zuständig; die **32** Bits, die nicht von der Speicherschnittstelle **70** geschrieben werden, werden durch die ihr zugeordnete Schnittstelleneinheit **24** an die Speicherschnittstelle übertragen, wo sie mit denselben **32** Bits auf Fehler verglichen werden.

[0132] Es sei hier am Rande erwähnt, dass in dem in den **Fig.** 1A–1C gezeigten System die Übertragung von Interrupts in Form von Nachrichtenpaketen erfolgt, und nicht unter Verwendung der dem Stand der Technik entsprechenden Methode, die darin besteht, für die Übertragung von spezifischen Interrupt-Typen fest zugeordnete Signalleitungen zu verwenden. Werden Nachrichtenpakete empfangen, welche Interrupt-Informationen enthalten, so werden diese Informationen zusammen mit gegebenenfalls von der CPU **12A** intern generierten Interrupts der Interrupt-Logik **86** zugeführt, welche sie verarbeitet und freigibt, womit das Verhalten des Prozessors **20** neu definiert wird. Intern generierte Interrupts setzen ein Bit in einem Register **71** (in der Interrupt-Logik **86** integriert), wodurch der Grund für das Interrupt angezeigt wird. Der Prozessor **20** kann daraufhin das Interrupt lesen und verhält sich entsprechend. Die Interrupt-Logik wird im Detail weiter unten genauer behandelt.

[0133] Der BÜM **88** der X-Schnittstelleneinheit **24a** wird betrieben, um direkte Speicherzugriffe vorzunehmen, und stellt den Mechanismus bereit, der es den Prozessoren **20** ermöglicht, auf externe Ressourcen zuzugreifen. Der BÜM kann von den Prozessoren **20** entsprechend eingerichtet werden, um für die Prozessoren **20** transparente E/A-Anforderungen zu generieren und die Prozessoren zu benachrichtigen, wenn die Anforderungen vollständig sind. Die BÜM-Logik **88** wird weiter unten abgehandelt.

[0134] In eingehenden Nachrichtenpaketen enthaltene Speicherzugriffsanforderungen werden durch die ZVÜ-Logik **90** verifiziert. Die Verifizierung der Zugriffsanforderung erfolgt nach verschiedenen Erlaubniskriterien, unter anderem nach der Identität des Ursprungselements der Anforderung und nach dem angeforderten Zugriffstyp. Ausserdem übersetzt die ZVÜ-Logik die (in dem empfangenen Nachrichtenpaket als virtuelle Adresse enthaltene) Speicheradresse, an welcher ein Zugriff gewünscht wird, in eine physikalische Speicheradresse, die zur Durchführung des eigentlichen Zugriffs verwendet werden kann, nachdem die Anforderung ordnungsgemäss verifiziert worden ist. Die ZVÜ-Logik **90** wird ebenfalls im Detail weiter unten abgehandelt.

[0135] Die BÜM-Logik **88** arbeitet mit der ZVÜ-Logik **90** zusammen, um dem Paketüberträger **94** die zu sendenden Daten und/oder Befehlssymbole bereitzustellen. Der Paketüberträger **94** fügt seinerseits die von der BÜM-Logik **88** und der ZVÜ-Logik **90** empfangenen Informationen zu Nachrichtenpaketen zusammen und hält sie bis zu ihrer Übertragung in einem Zwischenspeicher. Ausserdem arbeitet die BÜM-Logik **88** und die ZVÜ-Logik **90** auch mit dem Paketempfänger **96** zusammen, um eingehende Nachrichtenpakete zu empfangen, auszuwerten und behandeln, wobei diese im Bedarfsfall zwischengespeichert werden und in das 8 Byte breite Format umgewandelt werden, in dem sie vorliegen müssen, um im Speicher **28** abgespeichert werden zu können.

[0136] Abgehende Nachrichtenpakete, welche prozessorgenerierte Transaktionsanforderungen enthalten (z. B. eine Leseanforderung nach einem Datenblock von einer E/A-Einheit) werden von der Anforderungstransaktionslogik (ATL) **100** überwacht. Die ATL **100** sieht einen Zeitüberwachungszähler für abgehende Anforde-

rungen vor, welcher prüft, ob die Anforderung innerhalb einer vorgegebenen Zeitspanne beantwortet wird; ist dies nicht der Fall, so generiert die ATL ein (von der Interrupt-Logik **86** zu behandelndes und zu meldendes) Interrupt, um den Prozessor **20** zu informieren, dass die Anforderung nicht beantwortet worden ist. Ausserdem validiert die ATL **100** eingehende Antworten. Die ATL **100** hält die Adresse für die Antwort bereit und leitet diese Adresse an die BÜM **88** weiter, sobald die Antwort empfangen worden ist, so dass die Antwort (im Zuge der von der BÜM **86** ausgeführten Direkt-Speicherzugriffsoperation) im Speicher **28** an einem dem Prozessor **20** bekannten Speicherort abgelegt werden kann, so dass dieser die Antwort auffinden kann.

[0137] Eine jede der CPUs **12** wird, wie im folgenden beschrieben, auf mehrfache Weise geprüft. Eine dieser Prüfungen besteht in einer fortlaufenden Monitoreinrichtung zur Überwachung des Betriebs der Schnittstelleneinheiten **24a**, **24b** einer jeden CPU. Da die Schnittstelleneinheiten **24a**, **24b** synchron im Gleichschritt zueinander laufend betrieben werden, kann die Prüfung insofern erfolgen, als die Betriebszustände der gepaarten Schnittstelleneinheiten **24a**, **24b** mittels eines kontinuierlichen Vergleichs von gewissen internen Zuständen derselben überwacht werden. Dieser Ansatz wird unter Verwendung einer Phase einer in der Einheit **24a** der CPU **12A** enthaltenen Ablaufsteuereinheit (nicht gezeigt) realisiert, wobei jeder von dieser Phase eingenommene Zustand mit seiner identischen Ablaufsteuerungsphase in der Schnittstelleneinheit **24b** verglichen wird. Sämtlichen Einheiten der Schnittstelleneinheit **24** verwenden Ablaufsteuereinheiten zu ihrer Betriebsüberwachung. Vorzugsweise wird daher eine Ablaufsteuereinheit der Speicherschnittstelle **70** verwendet, welche die Datenübertragungen zwischen der Schnittstelleneinheit **24** und der Speichersteuerung **26** überwacht. Auf diese Weise wird eine ausgewählte Phase der in der Speicherschnittstelle **70** der Schnittstelleneinheit **24a** verwendeten Ablaufsteuereinheit ausgewählt. Eine identische Phase einer Ablaufsteuereinheit einer der Schnittstelleneinheiten **24b** wird ebenfalls ausgewählt. Die beiden ausgewählten Phasen werden zwischen den Schnittstelleneinheiten **24a**, **24b** übertragen und von einer in beiden Schnittstelleneinheiten **24a**, **24b** enthaltenen Vergleichsschaltung empfangen. Da die Schnittstelleneinheiten im Gleichschritt zueinander betrieben werden, durchwandern die Ablaufsteuereinheiten ebenfalls dieselben identischen Zustände und nehmen dabei jeden Zustand im wesentlichen zeitgleich zueinander ein. Falls nun eine Schnittstelleneinheit auf einen Fehler stösst oder ausfällt, so führt dies dazu, dass die Schnittstelleneinheiten auseinanderdriften und die Ablaufsteuereinheiten unterschiedliche Zustände einnehmen. Mit der Zeit werden sich auch die von den jeweiligen Ablaufsteuereinheiten an die Vergleichsschaltungen übermittelten, ausgewählten Phasen voneinander unterscheiden. Dieser Unterschied veranlasst die Vergleichsschaltungen, ein "lost sync"-Fehlersignal auszugeben, wodurch den CPUs **12A** (oder **12B**) zur Kenntnis gebracht wird, dass die Schnittstelleneinheiten **24a**, **24b** dieser CPU nicht mehr im Gleichschritt laufen, und entsprechende Schritte zu setzen. Ein Beispiel für diese Technik findet sich in dem US-Patent Nr. 4.672.609, ausgegeben an Humphrey, et al. und abgetreten an den Abtretungsempfänger dieser Anmeldung.

[0138] Der Paketempfänger **96** der X-Schnittstelle der CPU **12A**-um wieder zu **Fig.** 5 zurückzukehren-funktioniert in einer Weise, dass nur der X-Port bedient wird, indem er nämlich nur jene Nachrichtenpakete empfängt, welche von dem Router **14A** des Subprozessorsystems **10A** (**Fig.** 1A) übertragen werden. Der Y-Port wird von der Y-Schnittstelleneinheit **24b** bedient und empfängt Nachrichtenpakete von dem Router **14B** des dazugehörigen Subprozessorsystems **10B**. Beide Schnittstellen (so wie auch die Speichersteuerungen **26** und der Prozessor **20**) sind jedoch, wie bereits gezeigt wurde, insofern im Grunde genommen Spiegelbilder voneinander, als beide in Bezug auf ihren Aufbau und ihre Funktion im wesentlichen identisch sind. Aus diesem Grund muss die von einer Schnittstelleneinheit (z. B. **24a**) empfangene Nachrichtenpaketinformation zum Zweck der Verarbeitung auch an die dazugehörige Schnittstelleneinheit (z. B. **24b**) weitergegeben werden. Da darüber hinaus beide Schnittstelleneinheiten **24a**, **24b** dieselben Nachrichtenpakete zur Übertragung von dem X- bzw. Y-Port zusammenstellen, wird das Nachrichtenpaket, das von der Schnittstelleneinheit (z. B. **24b**) übertragen wird, und tatsächlich von dem dazugehörigen Port (z. B. dem Y-Port) aus übermittelt wird, auch an die andere Schnittstelleneinheit (**24a**) gekoppelt, wo es auf Fehler hin gegengeprüft wird. Diese Merkmale werden in den **Fig.** 6 und 8 veranschaulicht.

Paketempfänger:

[0139] Im folgenden werden unter Bezugnahme auf die **Fig.** 6 die empfangenden Abschnitte des Paketempfängers **96** (**96x**, **96y**) der X- und Y-Schnittstelleneinheiten **24a**, **24b** in groben Zügen veranschaulicht. Wie gezeigt, verfügt jeder Paketempfänger **96x**, **96y** über ein Taktsynchronisierungs-FIFO (TS-FIFO) **102**, das entsprechend verkoppelt ist, um ein zu diesem gehöriges der TNet-Links **32** zu empfangen. Die TS-FIFOs **102** werden betrieben, um die eingehenden Befehls/Datensymbole insofern mit dem lokalen Takt des Paketempfängers **96** zu synchronisieren, als sie zwischengespeichert und anschliessend an einen Multiplexer (MUX) **104** angelegt werden. Hierbei ist allerdings zu beachten, dass Informationen, die am X-Port und am Paketempfänger **96x** der X-Schnittstelle **24a** empfangen werden, zusätzlich zu ihrer Weitergabe an den MUX **104x** über die Quervernetzung $\mathbf{36_x}$ an den MUX **104y** des Paketempfängers **96y** der Y-Schnittstelleneinheit **24b** gekoppelt werden. In ähnlicher Weise werden am Y-Port empfangene Informationen über die Quervernetzung **36y**

an die X-Schnittstelleneinheit **24a** gekoppelt. Auf diese Weise werden die Befehls/Datensymbole von Informationspaketen, welche an einem der X-, Y-Ports von der jeweiligen X-, Y-Schnittstelleneinheit **24a**, **24b** empfangen werden, an die andere weitergegeben, so dass beide dieselben Informationen verarbeiten und an weitere Komponenten der Schnittstelleneinheiten **24** und/oder des Speichers **28** übermitteln.

[0140] Die Multiplexer **104**, um nun mit **Fig.** 6 fortzufahren, wählen in Abhängigkeit davon, welcher X-, Y-Port ein Nachrichtenpaket empfängt, entweder die Ausgabesignale des einen oder des anderen der TS-FIFOs **102x**, **102y** zur Datenübertragung an den Speicher und die Verarbeitungslogik **110** der Schnittstelleneinheit **24** aus. Die in jedem 9-Bit-Symbol enthaltene Information besteht aus einem 8-Bit-Byte Steuer bzw. Dateninformation, dessen Codierung weiter unten im Zusammenhang mit **Fig.** 9 abgehandelt wird. Die Speicher- und Verarbeitungslogik **110** übersetzt zuerst die 9-Bit-Symbole in 8-Bit-Daten- bzw. – Steuerbytes, und die Bytes organisiert die Bytes als 64-Bit-Doppelwörter, wobei die dergestalt gebildeten Doppelwörter in einen Eingangspaket-Zwischenspeicher (hier nicht im besonderen gezeigt) gestellt werden. Der Eingangspaket-Zwischenspeicher hält die empfangenen Informationen zwischenzeitlich bis sie zu der Speicherschnittstelle **70**, sowie zu der ZVÜ-Logik **90** und/oder zu dem BÜM **88** weitergegeben werden können.

[0141] Die Paketempfänger **96** beinhalten jeweils eine CRC-Prüflogik **106** zum Prüfen des CRC des Nachrichtenpakets. Es ist im besonderen zu beachten, dass jede CRC-Prüflogik **106** an einer entsprechenden Stelle angeordnet ist, so dass ungeachtet dessen, an welchem Port (X oder Y) das Nachrichtenpaket empfangen wird, beide Empfänger **96x**, **96y** den CRC des empfangenen Nachrichtenpakets prüfen. Dieses Merkmal beinhaltet ein Fehlerisolationsmerkmal. [Obgleich in dieser Empfangsphase geprüft,] Eine CRC-Fehleranzeige seitens eines Empfängers nicht jedoch des anderen deutet auf ein Problem bei der Schnittstelle zwischen den beiden Empfängern oder bei der Logik des Empfängers, welcher die Fehlermeldung ausgibt, hin. Somit kann der Fehler zumindest zu Anfang auf diesen Abschnitt des Pfades vom Ausgang der empfangenden TS-FIFO eingegrenzt werden.

[0142] Nicht dargestellt ist die Tatsache, dass die Ausgänge der TS-FIFOs **102x**, **102y** zusätzlich zu dem MUX **104** auch an eine Steuerdecodiereinheit gekoppelt ist. Die Befehlsdecodiereinheit wird betrieben, um Befehlssymbole zu erkennen (indem sie in einer weiter unten genauer beschriebenen Weise von Datensymbolen unterschieden werden), sie zu decodieren, und daraus Befehlssignale zu generieren, welche an eine Empfängersteuereinheit angelegt werden, worunter ein auf einer Ablaufsteuereinheit basierendes Element zu verstehen ist, das dazu verwendet wird, die Paketempfänger-Operationen zu steuern.

[0143] Wie weiter oben gezeigt, sind die Pakete durch einen zyklischen Redundanzprüfungswert (CRC-Wert) gegen Fehler geschützt. Wenn die CRC-Information des empfangenen Pakets am Ausgang des MUX **104** erscheint, versetzt der Empfängersteuerabschnitt der Speichersteuereinheit die CRC-Prüflogik **106** somit in die Lage, ein CRC-Symbol zu berechnen, während die Datensymbole empfangen werden, um daran anschliessend die generierte Menge mit dem zusammen mit dem Nachrichtenpaket empfangenen CRC zu vergleichen. Im Fall eines Nicht-Übereinstimmens, welches darauf hindeutet, dass während der Übertragung zu dem Paketempfänger **96** möglicherweise ein Fehler aufgetreten ist, gibt die CRC-Prüflogik **106** ein Fehler-Interruptsignal (BADCRC) aus, das dazu verwendet wird, ein Interrupt-Register zu setzen (Interrupt-Register **280**; **Fig.** 14A), woraufhin das Paket ausgeschieden wird. Der Paket-Header wird jedoch zum Zweck der späteren Prüfung in einer Interrupt-Warteschlange gespeichert.

[0144] Wie weiter unten ausgeführt, finden sich TS-FIFOs nicht nur in den Paketempfängern **96** der Schnittstelleneinheiten **24**, sondern auch an jedem empfangenden Port der Router **14** und der E/A-Paket-Schnittstellen **16**. Allerdings sind die TS-FIFOs, welche zum Empfangen von Symbolen von den TNet-Links L, die die CPUs **12A**, **12B** und die Router **14A**, **14B** (d. h. die Ports **1** und **2**) verbinden, verwendet werden, in gewisser Weise unterschiedlich von jenen, die an den anderen Ports der Router **14** sowie an jeglichen anderen nicht direkt mit einer CPU **12** verbundenen Routern **14** verwendet werden. Um es anders auszudrücken, die TS-FIFOs, welche dazu verwendet werden, um Symbole zwischen Elementen mit frequenzsynchronisierter Taktgebung auszutauschen, unterscheiden sich von jenen, die dazu verwendet werden, um Symbole zwischen Elementen mit frequenznaher Taktgebung auzutauschen.

[0145] Die folgenden Ausführungen werden auch zeigen, dass die TS-FIFOs eine bedeutende Rolle bei der Übertragung von Informationen auf den TNet-Links L zwischen in frequenznahem Betrieb arbeitenden Elementen spielen (d. h. die Taktsignale der übertragenden und der empfangenden Elemente sind nicht notwendigerweise gleich, es ist jedoch zu erwarten, dass sie innerhalb eines vorgegebenen Toleranzbereichs gelegen sind). Eine noch wichtigere Rolle, um nicht zu sagen eine einzigartige Funktion, haben die TS-FIFOs jedoch dann, wenn ein Paar von Subprozessorsystemen im Duplexbetrieb arbeitet und die beiden CPUs **12A** und **128** der Subprozessorsysteme **10A**, **10B** in synchronisiertem Gleichschritt betrieben werden und dieselben Befehle zur selben Zeit ausführen. Bei diesem letzteren Betriebsmodus ist es unabdingbar, dass Informationen, welche von irgendeinem der Router **14A** oder **14B** an die CPUs **12A** und **12B** übertragen werden, von beiden CPUs im wesentlichen zur selben Zeit empfangen werden, damit ein synchroner, im Gleichschritt laufender Betrieb aufrechterhalten werden kann. Dies zu gewährleisten ist leider keine leicht Aufgabe, da es sehr schwierig ist, dafür zu sorgen, dass das Taktsystem der Router **14A** und **14B** genau mit jenem der CPUs **12A** und **12B** syn-

chron läuft-und das selbst bei Einsatz von frequenzsynchronisierter Taktgebung. Innerhalb der Paketempfänger **96** der CPUs **12** ist es die Aufgabe der TS-FIFOs **102**, die Differenz auszugleichen, zu der es zwischen dem Takt des Routers **14**, der dazu verwendet wird, Symbole an die CPU **12** zu übertragen, und dem Takt, mit welchem die Schnittstelleneinheit **24** diese Symbole empfängt, kommen kann.

[0146] Der Aufbau des TS-FIFOs **102** ist im Hinblick auf eine anschaulichere Beschreibung in Diagrammform in **Fig.** 7A dargestellt; ein bevorzugter Aufbau des TS-FIFOs wird in **Fig.** 7B gezeigt. Es versteht sich auch hier wieder, dass wenn in der Folge von einem TS-FIFO gesprochen wird, sofern nicht anders angegeben, auf eine Struktur Bezug genommen werden soll, welche von der Funktion und vom Betrieb her jener entspricht, die unter Bezugnahme auf die **Fig.** 7A beschrieben wird und der in **Fig.** 7B gezeigten Struktur entspricht. Die Abhandlung des TS-FIFOs aus **Fig.** 7A soll daher allgemein gehalten sein und auch so verstanden werden. Weiterhin soll, obwohl manche der für den frequenzsynchronisierten Betrieb verwendeten TS-FIFOs sich von jenen unterscheiden, die für den frequenznahen Betrieb verwendet werden, die nachfolgende Beschreibung auf beide zutreffen. Im Anschluss an diese Ausführung folgt eine Beschreibung der Abänderungen, die am allgemeinen Aufbau des TS-FIFOs vorgenommen werden müssen, damit dieses sich zum Einsatz für eine frequenznahe Betriebsweise eignet.

[0147] Gezeigt wird in **Fig.** 7A das TS-FIFO **102x** des Paketempfängers **96x**. Das TS-FIFO **102y** ist vom Aufbau und von der Betriebsweise her im wesentlichen identisch, so dass die folgende Beschreibung des TS-FIFOs **102x** auch als auf das TS-FIFO **102y** anwendbar zu verstehen ist. **Fig.** 7A zeigt das TS-FIFO **102x** entsprechend über das TNet-Link **32** gekoppelt, um 9-Bit-Befehls-/Datensymbole zu empfangen, welche von einem Senderegister (Snd) **120** des Routers **14A** übertragen werden (**Fig.** 1A) gemeinsam mit einem ebenfalls von dem Router kommenden Sendetakt (S_Tkt). (Die gestrichelte Linie B in **Fig.** 7A symbolisiert die Taktgrenze zwischen der sendenden Einheit (Router **14A**) an einem Ende des entsprechenden TNet-Links **32** und der empfangenden Einheit, dem Paketempfänger **96x** der CPU **12A**. Die TS-FIF0 **102x** empfängt somit die 9-Bit-Symbole an einem Empfangsregister (Empf) **124**, in welchem sie zwischenzeitlich gehalten werden (z. B. während eines S_Tkt-Intervalls) bevor sie an eine Speicherwarteschlange **126** weitergegeben werden. Die Speicherwarteschlange **126** ist zur besseren Illustration und einfacheren Beschreibung als vier Speicherorte enthaltend dargestellt. Für einschlägig gebildete Fachleute ist jedoch klar, dass weitere Speicherorte nicht nur bereitgestellt werden können sondern in der Tat notwendig bzw. wünschenswert sind.

[0148] Empfangene Symbole werden (von dem Empf-Register **124**) in die TS-FIFO **102x** eingegeben, und zwar an Speicherorten der Speicherwarteschlange **126**, die von einem Eingabe-Zeigerzähler **128** identifiziert worden sind. Der Eingabe-Zeigerzähler **128** hat vorzugsweise die Form eines S_Tkt-getakteten Binärzählers. Empfangene Symbole werden dann der Reihe nach aus Speicherorten der Speicherwarteschlange **126**, die von einem Entnahme-Zeigerzähler **130** identifiziert worden sind, entnommen und an ein FIFO-Ausgaberegister **132** weitergegeben. Ein lokales Taktsignal, "Empf-Tkt", das dazu verwendet wird, Symbole aus der Speicherwarteschlange **126** und dem FIFO-Ausgaberegister **130** zu entnehmen, wird von einem (CPU **12A**)intern generierten Signal erzeugt. Vom FIFO-Aüsgaberegister **132** werden die Symbole an den MUX **104x** weitergegeben.

[0149] Gemäss dem für TNet-Übertragungen verwendeten Protokoll wird von allen sendenden Ports (z. B. den X- und Y-Ports der CPU **12A**, jedem sendenden Port des Routers **14** bzw. der E/A-Schnittstelle **16**-**Fig.** 1A) fortlaufend ein konstanter Strom von Symbolen übertragen; dabei kann es sich entweder um tatsächliche Befehls-/Datensymbole (d. h.

[0150] ein Paket) oder um LEERLAUF-Symbole handeln-ausgenommen während bestimmter Situationen (z. B. Reset, Initialisierung, Synchronisierung, sowie andere, weiter unten behandelte Situationen). Wie bereits weiter oben erklärt, wird jedes Symbol, das in dem Senderegister **120** des Routers **14A** gehalten wird, an den Empf-Register **124** gekoppelt und in der Speicherwarteschlange **126** gespeichert, zusammen mit dem von dem Router **14A** bereitgestellten Taktsignal, S_Tkt. Umgekehrt erfolgt die Entnahme der Symbole aus der Speicherwarteschlange synchron mit dem lokal erzeugten Takt, Empf-Tkt. Dabei handelt es sich um zwei unterschiedliche Taktsignale, wenngleich auch mit im wesentlichen derselben Frequenz. Solange jedoch genügend Zeit verstreicht (z. B. einige Taktintervalle) zwischen dem Zeitpunkt da ein Symbol in die TS-FIFO **102x** eintritt und jenem Zeitpunkt, da dasselbe Symbol aus der TS-FIFO entnommen wird, sollte es zu keinen Metastabilitätsproblemen kommen. Wenn das eingehende Taktsignal (S_Tkt) und der Empf-Tkt in frequenzsynchronisiertem Modus betrieben werden, sollte es niemals zu einem Überlauf oder Unterlauf der TS-FIF0 **102x** kommen .

[0151] Die Initialisierung der TS-FIFO **102x** erfolgt wie folgt. Zu Beginn überträgt der Router **14A** LEERLAUF-Symbole für jeden Impuls des Sendetaktsignals, S_Tkt, wodurch schliesslich das Empf-Register **124**, die Speicherwarteschlange **126** und das FIFO-Ausgaberegister **132** mit LEERLAUF-Symbolen gefüllt wird und das TS-FIFO **102x** in einen Leerlaufzustand zurückgesetzt wird. Der Eingabe-Zeigerzähler **128** und der Entnahme-Zeigerzähler **130** werden nach Empfang (und Erkennung) eines SYNC-Befehlssymbols auf Null gesetzt. Durch den Empfang des SYNC-Signals wird der Eingabe-Zeigerzähler **128** so eingestellt, dass er auf einen bestimmten Speicherort in der Speicherwarteschlange **126** zeigt. Gleichzeitig wird auch der Entnahme-Zeigerzähler **130** in ähnlicher Weise eingestellt, so dass er auf einen Speicherort der Speichewarteschlan-

ge **126** zeigt, der vorzugsweise um zwei Speicherorte von jenem des Eingabe-Zeigerzählers beabstandet ist. Dadurch wird eine nominale Verzögerung von zwei Takten zwischen dem Eintritt eines Symbols in die Speicherwarteschlange **126** und dessen Verlassen der Speicherwarteschlange eingerichtet, wodurch es jedem in die Warteschlange **126** eintretenden Symbol ermöglicht wird, sich zu beruhigen, bevor es wieder ausgetaktet und von dem MUX **104x** (und **104y**) an die Speicher- und Verarbeitungseinheiten **110x** (und **110y**) weitergegeben wird. Da der Sendetakt und der Empfangstakt phasenunabhängig voneinander sind, beinhaltet eine nominale Verzögerung von zwei Takten einen Fehlerbereich von plus oder minus einem vorgegebenen Betrag, so dass angenommen wird, dass der erlaubte Reset-Zeitversatz geringer oder gleich einem Takt ist.

[0152] **Fig.** 7B veranschaulicht eine Realisierung des TS-FIFOs **192x**, wobei die Speicherwarteschlange **126** als aus Multiplexer/Auffangspeicher-Kombinationen **140**, **142** bestehend dargestellt ist und wobei jede dieser Kombinationen einen Speicherort der Speicherwarteschlange **126** bildet. Die Auffangspeicher **142** werden mit jedem Impuls des S_Tkt angesteuert. Der Eingabe-Zeigerzähler **128** wird von einem Decodierer **144** decodiert, um einen der Multiplexer **140** zu veranlassen, die Ausgabesignale des Empf-Registers **124** zur Koppelung an den ihm zugeordneten Auffangspeicher **142** auszuwählen. Der Auffangspeicher wird mit dem S_Tkt geladen und der Eingabe-Zeigerzähler wird inkrementiert, um einen weiteren der Multiplexer **140** zu veranlassen, das Empf-Register an einen zugeordneten Auffangspeicher **142** zu übertragen. Jene Auffangspeicher **142**, die nicht ausgewählt werden, um die Ausgabesignale des Empf-Registers **124** zu empfangen, empfangen und laden stattdessen den Auffangspeicherinhalt mit S_Tkt.

[0153] Im wesentlichen zeitgleich dazu wählt der Entnahmezähler **130** den Inhalt eines der Auffangspeicher über einen Multiplexer **146**, um ihn gemeinsam mit dem Emfp-Tkt an das FIFO-Ausgaberegister **132** zu übertragen und in dieses zu laden; gleichzeitig dazu wird der Entnahme-Zeigerzähler **132** aktualisiert (inkrementiert).

[0154] Das TS-FIFO **102x** ist entsprechend aufgebaut, um eine frequenzsynchronisierte Taktgabe durchzuführen (d. h. S_Tkt und Empf-Tkt sind im wesentlichen gleich was die Frequenz, nicht jedoch die Phase betrifft), die nur verwendet wird, wenn ein Paar von CPUs **12** im Duplex-Betriebsmodus läuft, und es sich um Übertragungen zwischen den Routern **14A**, **14B** und den gepaarten CPUs **12A**, **12B** (**Fig.** 1) handelt. Die anderen Ports der Router **14** (und der E/A-Schnittstellen **16**), die nicht mit (im Duplexbetrieb funktionierenden) CPUs **12** kommunizieren, werden betrieben, um Symbole unter frequenznaher Taktgabe zu übertragen. Dennoch werden auch an diesen anderen Ports Taktsynchronisierungs-FIFOs verwendet, um Symbole zu empfangen, welche unter frequenznaher Taktgabe übertragen wurden, und diese Taktsynchronisierungs-FIFOs sind von ihrem Aufbau her im wesentlichen identisch mit jenen, die in frequenzsynchronisierten Umgebungen eingesetzt werden, d. h. mit den TS-FIFOs **102**. Es bestehen jedoch Unterschiede. So sind zum Beispiel die Symbol-Speicherorte der Speicherwarteschlange **9** Bit breit; bei frequenznahen Umgebungen verwenden die Taktsynchronisierungs-FIFOs Symbol-Speicherorte der Warteschlange **126**, welche **10** Bit breit sind, wobei es sich bei dem zusätzlichen Bit um ein "gültig"-Flag handelt, an dessen Zustand zu erkennen ist, ob das dazugehörige Symbol gültig ist oder nicht. Auf dieses Merkmal wird weiter unten noch genauer eingegangen.

[0155] Ein Router **14** kommuniziert oft mit Geräten (z. B. anderen Routern oder E/A-Schnittstellen **16**) in anderen Computern, die unter dem Einfluss anderer Taktgeräte stehen, die zwar dieselbe Nennfrequenz aufweisen wie der Router **14**, welcher Symbole überträgt bzw. empfängt, deren reale Frequenzen sich jedoch oft geringfügig davon unterscheiden. Es ist also eine frequenznahe Situation gegeben, und mit dieser Form der Taktgebung beim Symboltransfer sind alle Ports eines Routers **14** konfrontiert, ausser jene, die beim Duplexbetrieb direkt mit einer CPU **12** verbunden sind. Beim frequenznahen Betrieb können die Taktsignale (z. B. der an einem Ende zur Symbolübertragung verwendete Takt und der am anderen Ende zum Signalempfang verwendete Takt) langsam auseinanderdriften, wobei der eine sich schlussendlich um einen Taktzyklus gegenüber dem anderen verschiebt. wenn dieser Fall eintritt, zeigen die beiden Zeiger (der Eingabe- und der Entnahme-Zeigerzähler **128** bzw. **130**) des TS-FIFOs **102** auf einen Symbol-Speicherort der Speicherwarteschlange **126**, der entweder um einen Symbol-Speicherort näher aneinander oder um einen Speicherort weiter voneinander entfernt liegt, je nachdem welche Einheit (Überträger oder Empfänger) über die schnellere Taktquelle verfügt. Um diese Taktverschiebung in den Griff zu bekommen, werden die beiden Zeiger in der Praxis in regelmässigen Abständen resynchronisiert.

[0156] Wenn die CPUs **12** gepaart sind und im Duplexbetrieb laufen, alle vier Schnittstelleneinheiten **24** im Gleichschritt betrieben werden, um unter anderem dieselben Daten zu übertragen und Daten im selben Takt zu empfangen (S_Tkt und Empf-Tkt), wird eine frequenzsynchronisierte Taktgebung benötigt und verwendet. Wenn die CPUs **12** im Simplexmodus betrieben werden, und jede unabhängig von der anderen arbeitet, genügt es, mit frequenznaher Taktgebung zu arbeiten.

[0157] Die Schnittstelleneinheit **24** empfängt ein SYNC TKT-Signal, welches in Kombination mit dem SYNC-Steuersymbol verwendet wird, um das Empf-Register **124** zu initialisieren und mit den übertragenden Router **14** zu synchronisieren. Sowohl für den frequenznahen als auch den frequenzsynchronisierten Taktbetrieb für Symbolübertragungen ist es vorzuziehen, wenn das TS-FIFO **102x** bei einem bekannten Zustand beginnt. Eingehende Symbole werden von den Speicher- und Verarbeitungseinheiten **110** des Paketempfängers

96 geprüft. Die Speicher- und Verarbeitungseinheiten suchen nach Steuersymbolen und verhalten sich diesen entsprechend. Das Festhaltenswerte dabei ist, dass wenn der Paketempfänger **96** ein SYNC-Befehlssymbol empfängt, dieses von der Speicher- und Verarbeitungseinheit **110** decodiert und erkannt wird. Die Erkennung des SYNC-Befehlssymbols durch die Speicher- und Verarbeitungseinheit **110** führt zur Aktivierung eines RESET-Signals. Das unter synchroner Steuerung durch das SYNC-Tkt-Signal generierte RESET-Signal wird dazu verwendet, die Eingangspufferspeicher (einschliesslich der Taktsynchronisierungs-Pufferspeicher) wieder auf vorgegebene Zustandswerte einzustellen und sie mit den Routern **14** zu synchronisieren.

[0158] Die Synchronisierung der TS-FIFOs **102** der Schnittstelleneinheiten **24**, sowie von jener eines oder beider Router **14A**, **14B** wird im einzelnen weiter unten im Abschnitt über Synchronisation behandelt.

Paketüberträger:

[0159] Jede Schnittstelleneinheit **24** hat die Aufgabe, jeweils nur von einem Port aus, dem X-Port oder dem Y-Port, der CPU **12** Übertragungen durchzuführen bzw. an diesem Port Daten zu empfangen. Wenn eine der Schnittstelleneinheiten **24** überträgt, besteht der Betrieb der anderen darin, die in Übertragung befindlichen Daten zu prüfen. Dies ist ein wichtiges Merkmal des Paketüberträgers, weil dadurch eine selbstprüfende Fehlererkennungs- und Fehlerbegrenzungsfähigkeit der CPU **12** geschaffen wird, und zwar selbst dann, wenn diese im Simplexbetrieb läuft.

[0160] Dieses Merkmal wird in **Fig.** 8 veranschaulicht, in welcher in abgekürzter Form die Paketüberträger **94x**, **94y** der jeweiligen X- und Y-Schnittstelleneinheiten **24a**, **24b** gezeigt werden. Beide Paketüberträger sind identisch aufgebaut, so dass die Besprechung des einen (Paketüberträger **94x**) auch auf den anderen (Paketüberträger **94y**) anwendbar ist, sofern nicht anders angegeben.

[0161] Wie **Fig.** 8 zeigt, beinhaltet der Paketüberträger **94x** eine Paketierlogik **152** welche von der BÜM **88** oder der ZVÜ **90** der zugeordneten Schnittstelleneinheit (hier die X-Schnittstelleneinheit **24a**) die zu übermittelnden Daten empfängt, und zwar in Doppelwortformat (64-Bit-Format). Die Paketierlogik **152** zwischenspeichert die Informationen bis diese bereit zur Übertragung aus dem X- oder Y-Port der CPU **12** sind, führt eine Byte-Steueroperation durch, um die Daten von dem Doppelwortformat in ein Byteformat zu übersetzen, setzt die Bytes in Paketform zusammen und gibt sie an einen der X- und Y-Codierer **150x**, **150y** weiter. Nur einer der Codierer **150** empfängt die Bytes, je nachdem, welcher Port (X oder Y) das entstehende Nachrichtenpaket überträgt.

[0162] Jener X- oder Y-Codierer **150**, der die 8-Bit-Bytes empfängt, wird betrieben, um sie in einem 9-Bit-Befehls/Datensymbolformat zu codieren, wie in **Fig.** 9 veranschaulicht. Die Codierung der drei linkerhand angeordneten Bits des sich ergebenden 9-Bit-Symbols wird in den drei linken Spalten der untenstehenden Tabelle 1 gezeigt.

8B–9B Symbolcodierung

TABELLE 1

CDC	CDB	CDA	Funktion
0	0	0	Befehl
0	0	1	Fehler
0	1	0	Fehler
1	0	0	Fehler
0	1	1	Daten <7:6> = 00
1	0	1	Daten <7:6> = 01
1	1	0	Daten <7:6> = 10
1	1	1	Daten <7:6> = 11

[0163] Wie Tabelle 1 bei gemeinsamer Betrachtung mit **Fig.** 9 veranschaulicht, sind die höherwertigen drei Bits (CDC, CDB, CDA) der **9** Bits entsprechend codiert, um anzuzeigen, ob die restlichen, niederwertigen sechs Bits des Symbols (CDS, CD4, CD3, CD2, CD1 und CDO) als (1) Befehlsinformation oder (2) Daten zu

interpretieren sind. Wenn somit die drei bedeutendsten Bits, CDC, CDB und CDA alle Null sind, so ist das 9-Bit-Symbol dadurch als Befehlssymbol gekennzeichnet, wobei die verbleibenden sechs Bits den Befehl bilden. So würde zum Beispiel ein Befehls/Datensymbol der Form "000cccccc" als Befehl interpretiert, wobei die "c"-Bits den Befehl darstellt.

[0164] Wenn die drei bedeutendsten Bits CDC, CDB und CDA des Befehls-/Datensymbols andererseits irgendeinen der vier charakteristischen Datenwerte annehmen, so werden sie als zwei Datenbits interpretiert, welche mit den verbleibenden sechs Datenbits zu einem Datenbyte zu kombinieren sind. Bei den verbleibenden sechs handelt es ich um die am wenigsten bedeutenden Bits des Datenbytes. Somit würde ein Befehls/Datensymbol, das sich wie folgt darstellt "110001101" als Datensymbol interpretiert und in ein Datenbyte der Form "10001101" übersetzt werden. Ein Fehler liegt vor, wenn die drei bedeutendsten Bits eine der Formen 001, 010 und 100 annehmen.

[0165] Die drei Fehlercodes, die die Datensymbole von den Befehlssymbolen trennen, bilden einen minimalen Hamming-Abstand von zwei zwischen Befehlen und Daten. So ist es nicht möglich, dass ein Einzelbitfehler Daten in ein Befehlssymbol verwandelt oder umgekehrt.

[0166] Ausserdem werden die sechs Bits niedrigeren Ranges eines Befehlssymbols (im Gegensatz zu einem Datensymbol) in dem bekannten "drei von sechs"-Code codiert, in welchem die sechs Bitpositionen, die den Befehl enthalten, immer genau drei "EINSEN" enthalten. Alle eindirektionalen Fehler, sowie jede ungerade Zahl von Fehlern in einem Befehlssymbol wird so erkannt. Fehler in den Daten werden durch Paket-CRC-Prüfungen erkannt, ebenso wie Fehler, welche Befehlssymbole in Daten umwandeln. Fehler, welche Daten in Befehlssymbole umwandeln, werden durch CRC und/oder Protokollverletzungsfehler erkannt, wie weiter unten noch genauer beschrieben wird.

[0167] Welcher der X- oder Y- Codierer **150** die Informationsbytes von der Paketierlogik **152** empfängt, hängt von der in der zu übertragenden Information enthaltenen Bstimmungsort-ID ab, in der auch das Pfad-Bit (P) beinhaltet ist, das den einzuschlagenden Pfad angibt. Angenommen zum Beispiel die in der Information enthaltene Bestimmungsort-ID legt nahe, dass diese über den X-Port der CPU **12** gesendet werden sollte. Die Paketierlogik **152** (der beiden Paketüberträger **94x**, **94y**) sendet diese Information an den X-Codierer **150x**; zur selben Zeit sendet sie LEERLAUF-Symbole an den Y-Codierer **150y**. Es werden kontinuierlich Symbole von dem X- und dem Y-Port gesendet: dabei handelt es sich entweder um Symbole, die ein gerade in Übertragung befindliches Nachrichtenpaket bilden, oder um LEERLAUF-Symbole, oder um andere Befehlssymbole, die zur Durchführung von Steuerfunktionen verwendet werden.

[0168] Die Ausgabesignale der X- und Y-Codierer **150** werden an eine Multiplexanordnung angelegt, welche die Multiplexer **154**, **156** beinhaltet. Der Ausgang des Multiplexers **154** ist mit dem X-Port verbunden. (Die Schnittstelleneinheit **24b** verbindet den Ausgang des Multiplexers **154** mit dem Y-Port.) Der Multiplexer **156** ist mit der Prüflogik **160** verbunden, welche über das Quervernetzungs-Link **34y** auch die Ausgangssignale des mit dem Y-Port verbundenen Multiplexers **154** empfängt. Es gilt zu beachten, dass die Ausgangssignale der Multiplexers **154**, der mit dem X-Port und dem TNet Link $\mathbf{30}_x$ verbunden ist, über das Quervernetzungs-Link $\mathbf{34}_x$ auch an die Prüflogik **160** des Paketüberträgers **94y** (der Schnittstelleneinheit **24b**) gekoppelt ist.

[0169] Ein Wahleingang (W) der Multiplexer empfängt ein 1 Bit Ausgabesignal von einer X/Y-Stufe des Konfigurationsregisters **162**. Das Konfigurationsregister **162** ist für den IP **18** über einen in der Schnittstelleneinheit **24** ausgebildeten OLZP (nicht dargestellt) zugänglich und wird mit Informationen beschrieben, durch welche die Schnittstelleneinheiten **24** unter anderem "personalisiert" werden. Hier konfiguriert die X/Y-Stufe des Konfigurationsregisters **162** den Paketüberträger **94x** der X-Schnittstelleneinheit **24a** entsprechend, so dass dieser die Ausgabesignale des X-Codierers **150x** an den X-Port übermittelt; die Ausgabesignale des Y-Codierers **150y** werden ebenfalls an die Prüflogik **160** gekoppelt. In ähnlicher Weise wird die X/Y-Stufe des Konfigurationsregisters **162** des Y-Paketüberträgers **94y** (der Y-Schnittstelle **24b**) in einen Zustand versetzt, der den Multiplexer **154** dazu veranlasst, die Ausgabesignale des Y-Codierers **150y** zur Übertragung an den Y-Port auszuwählen; und die Ausgabesignale des X-Codierers **150x** zur Koppelung an die Prüflogik **160** des Paketüberträgers **160** auszuwählen, wo sie mit den Übertragungen des X-Ports verglichen werden.

[0170] Kurz gesagt, läuft die Operation der Nachrichtenpaketübertragung von dem X-Port bzw, dem Y-Port wie folgt ab. Zu Beginn, wenn wie bereits angedeutet noch keine Nachrichtenpaketübertragungen im Gange sind, übertragen der X- und der Y-Codierer beide -BYTE-Symbole oder andere Symbole, die zur Durchführung von Steuerfunktionen verwendet werden. Wenn die X/Y-Stufen der Konfigurationsregister **162** beider Paketüberträger **94** wie oben angegeben eingestellt sind (d. h. wenn der X-Codierer **150x** des Paketüberträgers **94x** über den Multiplexer **154** mit dem Ausgangsport (X) verbunden ist; und der Y-Codierer **150y** des Paketüberträgers **94y** über den Multiplexer **154** mit dem Ausgangsport (Y) verbunden ist), werden von dem X-Codierer **150x** (des Paketüberträgers **94x**) LEERLAUF-Symbole von dem X-Port der CPU **12A** übertragen, und werden von dem Y-Codierer **150y** (des Paketüberträgers **94y**) erzeugte LEERLAUF-Symbole von dem Y-Port übertragen. Zur selben Zeit werden die X-Port-Übertragungen durch das Quervernetzungs-Zink **34x** an die Prüflogik **160** des Paketüberträgers **94y** gekoppelt und mit jenen gegengeprüft, die von dem X-Codierer **150x** dieses Paketüberträgers erzeugt werden. In derselben Weise werden die von dem Y-Port abgehenden LEER-

LAUF-Symbole von dem Paketüberträger **94y** aus an die Prüflogik **160** des Paketüberträgers **94x** gekoppelt, wo sie mit den von dem Y-Codierer **150y** des Paketüberträgers **94x** erzeugten Signalen gegengeprüft werden.

[0171] Aus diesen Ausführungen wird eine wichtige Tatsache klar: die korrekte Funktionsweise der Paketüberträger kann auch dann überwacht werden, wenn diese nicht gerade dabei sind, Nachrichtenpakete zu übertragen. Im Gegenteil, selbst wenn kein Nachrichtenpaketverkehr vorhanden ist, wird die Funktion der beiden Paketschnittstellen **94** (und damit auch der Schnittstelleneinheiten **24**, denen sie zugeordnet sind) kontinuierlich überwacht. Sollte eine der Prüflogiken ein Nichtübereinstimmen bei den an sie angelegten Signalströmen erkennen, so wird ein FEHLER-Signal ausgegeben, welches das Freigeben eines internen Interrupts zur Folge hat, mit dem der Prozessor **20** aufgefordert wird, entsprechende Schritte zu setzen.

[0172] Der Nachrichtenpaketverkehr funktioniert in derselben Weise. Fürs erste sei angenommen, dass die Paketierlogik **152** des Paketüberträgers **94** Informationen zur Übertragung empfängt und dass die Bestimmungsort-ID angibt, dass dafür der X-Port zu verwenden ist. Die Paketierlogik leitet die Information byteweise an den X-Codierer **150x** beider Schnittstelleneinheiten **96** weiter, welcher jedes Byte in die codierte 9-Bit-Form übersetzt. Die Ausgangsinformation des X-Codierers **150x** des Paketüberträgers **94x** wird von dem Multiplexer **154** an den X-Port und an das TNet-Zink **30x**, sowie an die Prüflogik **160** des Paketüberträgers **94y** übermittelt. In der Zwischenzeit wird die Ausgangsinformation des X-Codierers des Paketüberträgers **94y** von dem Multiplexer **156** lediglich an die Prüflogik **160** gekoppelt, wo sie mit jener, die von dem Paketüberträger **94x** kommt, gegengeprüft wird. Auch hier wieder wird die Funktion der Schnittstelleneinheiten **24a**, **24b** und die in ihnen enthaltenen Paketüberträger auf Fehler geprüft.

[0173] In derselben Art werden, wie nun ersichtlich, die vom Y-Port abgehenden Nachrichtenpaketübertragungen überwacht.

[0174] Unter erneuter Bezugnahme auf **Fig.** 5 sei hier festgehalten, dass wenn es sich bei dem abgehenden Nachrichtenpaket um eine prozessorinitiierte Transaktion handelt (z. B. um eine Leseanforderung), die Prozessoren **20** erwarten, dass ein Nachrichtenpaket als Antwort rückgesendet wird. Wenn die BÜM **88** somit die zu sendenden Daten von dem Speicher **28** zu dem Paketüberträger **94** überträgt, stellt sie einen in der Anforderungstransaktionslogik **100** enthaltenen Anforderungs-Zählerzeitgeber (nicht dargestellt) entsprechend ein, so dass dieser den Beginn einer Zeitüberwachungsdauer markiert, innerhalb derer eine Antwort empfangen werden sollte. Sofern eine Antwort auf die abgehende Anforderung empfangen wird, ermittelt eine Antwortabgleichschaltung des Paketempfängers **96**, dass es sich bei dem Nachrichtenpaket um eine Antwort handelt und setzt den Anforderungs-Zählerzeitgeber zurück. Es ist nur ein Anforderungs-Zählerzeitgeber (nicht dargestellt) für jede Anzahl ausstehender Anforderungen an einen Bestimmungsort vorhanden. Jedes Mal wenn die BÜM **88** die Übertragung einer Transaktion initiiert, wird der Zählerzeitgeber zurückgesetzt.

[0175] Wenn andererseits die Antwort nicht innerhalb der dafür eingeräumten Zeit empfangen wird, so gibt der Anforderungs-Zählerzeitgeber ein Zeitablauffehler-Signal an die Interrupt-Logik (**Fig.** 14A) aus, um damit die Prozessoren **20** über die nicht eingegangene Antwort auf eine bestimmte Transaktion (z. B. eine Leseanforderung) zu benachrichtigen. Sollen mehrere ausständige Anforderungen unterstützt werden, so bestünde die Möglichkeit, mehrere Anforderungs-Zählerzeitgeber – einen pro ausständiger Anforderung-zu verwenden.

[0176] Ein von aussen ergehender Zugriff auf den Speicher **28** der CPU **12A** ist zwar vorgesehen, dieser ist jedoch mit einer Schutzfunktion versehen. Extern generierte Anforderungen nach Zugriff auf den Speicher **28** werden nur dann gewährt und genehmigt, wenn sie gemäss gewisser Kriterien autorisiert sind, wobei hier die Identifizierung des Ursprungs der Anforderung, der angeforderte Zugriffstyp (z. B. Lese- oder Schreibzugriff), der Speicherbereich auf den zugegriffen werden soll, zu erwähnen sind, um nur einige zu nennen. Der Speicherbereich der Speichereinheit **28**, auf den zugegriffen werden soll, ist daher auch in dem Nachrichtenpaket mittels virtueller bzw. E/A-Speicheradresse angegeben, (wodurch die Anwendung virtueller Speicherverfahren ermöglicht wird). Es folgt alsdann die Ermittlung der Autorisierung, und falls erlaubt, der Zugriff, der es erforderlich macht, dass diese virtuellen Adressen in physikalische Adressen des Speichers **28** übersetzt werden. Schliesslich werden Interrupts, welche von ausserhalb der CPU **12A** gelegenen Einheiten bzw. Elementen generiert worden sind, auf dem Weg über die Nachrichtenpakete übertragen, um die Prozessoren **20** zu unterbrechen, wobei die Interrupts ebenfalls anlässlich ihrer Inempfangnahme in den Speicher **28** geschrieben werden. Alle diese Vorgänge werden durch die Interrupt-Logik und die ZVÜ-Logik **86**, **90** abgewickelt.

[0177] Die ZVÜ-Logikeinheit **90** verwendet eine Tabelle (aufrechterhalten in dem Speicher **28** durch den Prozessor **20**), welche für jede potentielle externe Quelle mit Zugriffsrecht auf den Speicher **28** entsprechende ZVÜ-Einträge enthält. Jeder ZVÜ-Eintrag bezeichnet ein spezifisches Ursprungselement bzw. eine Ursprungseinheit und die bestimmte Speicherseite oder den Speicherseitenabschnitt, auf die bzw. auf den zugegriffen werden darf. Soll von einem ausserhalb der CPU **12** gelegenen Element auf mehr als eine Seite zugegriffen werden dürfen, so muss für jede einzelne Seite, auf die von dem Element zugegriffen werden dürfen soll, ein eigener ZVÜ-Eintrag vorhanden sein. Ausserdem enthält jeder ZVÜ-Eintrag Informationen betreffend den Typ oder die Typen von Speicheroperationen (z. B. Schreiboperation, Leseoperation, oder beides), welche erlaubt sind. Für "erwartete" Speicherzugriffe wird die ZVÜ-Tabelle nicht benötigt, und somit auch nicht verwendet. Erwartete Speicherzugriffe sind solche, die von der CPU **12** (d. h. den Prozessoren **20**) eingeleitet werden, wie

zum Beispiel eine Leseanforderung für Informationen seitens eines E/A-Geräts. Diese letzteren Speicherzugriffe werden über eine Transaktionsfolgenummer (TFN) abgewickelt, welche jeder prozessorinitiierten Anforderung zugewiesen wird. Annähernd zeitgleich zur Generierung der Leseanforderung bestimmen die Prozessoren **20** einen Speicherbereich für die als Antwort auf die Leseanforderung zurückerwarteten Daten. Die Adresse für diesen Bereich wird in einer (nicht dargestellten) Registerdatei gespeichert, die zum Zeitpunkt des Sendens der Leseanforderung von der Anforderungstransaktionslogik **100** fortgeschrieben wird, wobei die TFN als Hinweismarke für die Adresse innerhalb der Registerdatei verwendet wird. Somit kommt die Antwort auf die Leseanforderung mit den Daten zurück, und verwendet die TFN, die sie trägt, um die Adresse des Pufferbereichs des Speichers zu erhalten, an der die Retourdaten zu speichern sind.

[0178] Eine eingehendere Abhandlung der Zugriffsvalidierung erfolgt im dem folgenden Abschnitt. Die Speichermatrix **28** ist nämlich in zwei Hälften geteilt, die jeweils von den Speichersteuerungen **26a**, **26b** verwaltet werden, von denen jede eine Hälfte von jedem 64-Bit-Doppelwort verwaltet, das in den Speicher **28** geschrieben oder aus diesem ausgelesen wird. Die Speichersteuerungen **26a** und **26b** sind ihrerseits jeweils an die Speicherschnittstellen **70** jeder Schnittstelleneinheit **24a**, **24b** gekoppelt. Die 64-Bit-Doppelwörter werden entsprechend in den Speicher **28** geschrieben, so dass die oberen **32** Bits (sowie der ihnen zugeordnete FKC) von der "oberen" SpSt **26a** und die unteren **32** Bits (sowie der ihnen zugeordnete FKC) von der "unteren" SpSt **26b** geschrieben werden. Die Speichersteuerungen **26a**, **26b** empfangen jeweils **32** Datenbits und **4** FKC-Prüfbits von den Speicherschnittstellen **70** (**70a**, **70b**) einer jeden der Schnittstelleneinheiten **24a**, **24b** (**Fig.** 5).

[0179] Unter Bezugnahme auf **Fig.** 10, empfängt jede Speicheschnittstelle **70** entweder über den von der Prozessoreinheit **60** kommenden Bus **82** oder über den von der ZVÜ-Logik **90** kommenden Bus **83** (siehe **Fig.** 5) der ihr zugeordneten Schnittstelleneinheit **24** einen Datenumfang von 64 Bits, die in den Speicher zu schreiben sind. Die Busse **82** und **83** werden an einen Multiplexer (MUX) **84** angelegt, der auswählt welcher davon an den SpSt-A/D-Bus **25** zu koppeln ist.

[0180] Obwohl jede Speicherschnittstelle **70a**, **70b** dieselben, in den Speicher zu schreibenden **64** Bit in ihrer Gesamtheit empfängt, überträgt jede nur eine Hälfte dieser **64** Datenbits (und vier der acht FKC-Prüfbits, die von jeder erzeugt werden) an die Speichersteuerungen **26a**, **26b**. Die **32** Bits (und **4** der **8** von der FKC-Logik **85** erzeugten FKC-Bits) die nicht zur Ansteuerung der Speichersteuerungen **26** verwendet werden, werden von jeder Speicherschnittstelle **70** zum Zweck der wechselseitigen Gegenprüfung an die jeweils andere gekoppelt. So steuert zum Beispiel die Speicherschnittstelle **70a** (der Schnittstelleneinheit **24a**) die SpSt **26a** nur mit den "oberen" 32 Bits der **64** Datenbits (und mit **4** Bits des 8-Bit-FKC-Prüfworts) an. Zur selben Zeit empfängt die Speicherschnittstelle **70a** die "unteren" 32 Datenbits von der ihr entsprechenden Speicherschnittstelle **70b** und vergleicht diese unter Verwendung der Vergleichslogik **81** mit ihren eigenen unteren **32** Bits. Bei Erkennung einer Nichtübereinstimmung wird ein FEHLER-Signal ausgegeben.

[0181] In ähnlicher Weise wird der Partner-Speicherschnittstelle **70b** ein in den Speicher **28** zu schreibendes 64-Bit-Doppelwort übermittelt, doch nur die unteren 32 Bits (und 4 Bits der generierten FKC-Prüfbits) werden verwendet. Die Speicherschnittstelle empfängt die oberen 32 Bits von der Speicherschnittstelle **70a** und vergleicht sie an der Vergleichslogik **81** mit ihren eigenen oberen 32 Bits, wobei bei Nichtübereinstimmen der Daten ein FEHLER-Signal ausgegeben wird.

[0182] Eine zusätzliche Fehlerprüfung wird bei Leseoperationen durch FKC-Prüfschaltungen **85** einer jeden Speicherschnittstelle **70** vorgenommen. Jedes von der SpSt **26** rückübermittelte 64-Bit-Doppelwort wird von beiden Speicherschnittstellen **70** gemeinsam mit 8 FKC-Prüfbits empfangen. Die Daten und FKC-Prüfbits werden an die FKC-Logik **85** einer jeden Speicherschnittstelle **70** angelegt, wodurch ein Syndrom zur herkömmlichen Integritätsprüfung der Daten aufgebaut wird. Wird ein Einzelbitfehler erkannt, führt die FKC-Logik **85** eine Korrektur durch; wird ein unkorrigierbarer Fehler erkannt, so gibt die FKC-Logik ein Fehlersignal aus (nicht dargestellt), das dazu führt, dass ein Status des Interrupt-Registers **280** gesetzt wird (**Fig.** 18), wodurch laufende Operationen eingefroren werden.

[0183] Die spezielle FKC-Prüfung, die von jeder FKC-Logik **85** einer jeden Speicherschnittstelle durchgeführt wird, verwendet **8** Prüfbits für Einzelbitfehlerkorrektur, Doppelbitfehlererkennung und Syndrombitfehlererkennung für ein Feld mit einer Grösse von bis zu 112 Bit. Der Code ist ein ungerader Spaltengewichtungscode, was bedeutet, dass jeder Fehler zu einer ungeraden Anzahl von Syndrombits führt. Von den 112 möglichen Bits sind 64 Datenbits und **8** Prüfbits, wobei 40 Bits ungenutzt bleiben.

Zugriffsvalidierung:

[0184] Wie bereits angesprochen, können ausserhalb der CPU **12A** gelegene Komponenten des Verarbeitungssystems **10** (z. B. Einrichtungen der E/A-Paketschnittstellen **16**, oder die CPU **12B**) direkten Zugriff auf den Speicher **28** haben, müssen dazu jedoch speziell qualifiziert sein. Die durch die ZVÜ-Logik **90** der Schnittstelleneinheiten **24** geschaffene Zugriffsvalidierung ist dazu da, um den Inhalt des Speichers **28** vor Zerstörung durch irrtümliches bzw. unbeabsichtigtes Überschreiben von guten Daten durch andere Daten zu schützen, die nicht in diese Speicherorte geschrieben werden sollten In ähnlicher Weise bietet die Zugriffsvalidierung auch

einen Schutz gegen einen Zugriff, der unbeabsichtigterweise die falschen Speicherorte ausliest und dadurch falsche Daten an die Einheit oder das Systemelement liefert, von der/dem die Lesedaten angefordert wurden. Aus diesen und ähnlichen Gründen wird ein Verfahren zur Zugriffsvalidierung geschaffen, um zu gewährleisten, dass ein Speicherzugriff ordnungsgemäss erfolgt, d. h. dass das richtige Gerät die richtigen Speicherorte beschreibt bzw, ausliest. Wenn eine eingehende Speicheranforderung (d. h. eine Lese- oder Schreiboperation) validiert wird, wird die Adresse des Speicherortes, welche auf dem Adressfeld des Nachrichtenpakets, mit dem die Anforderung übertragen wird, aufscheint, von der ZVÜ-Logik in eine Speicheradresse übersetzt.

[0185] Zugriffe auf den Speicher **28** werden von der ZVÜ-Logik **90** einer jeden Schnittstelleneinheit **24** (**Fig.** 5) nach der Durchführung aller sechs der folgenden Prüfungen validiert: es muss gewährleistet sein, (1) dass der CRC des Nachrichtenpakets, mit dem die Anforderung übertragen wird, fehlerfrei ist, (2) dass der in dem Nachrichtenpaket angegebene Bestimmungsort (z. B. die CPU **12A**) jener des Empfängers ist, (3) dass das in dem Nachrichtenpaket angegebene Ursprungselement der Anforderung ein korrektes Ursprungselement ist, (4) dass der angeforderte Zugriffstyp ein für das Ursprungselement der Anforderung erlaubter Zugriffstyp ist, (5) dass der Zugriff auf den Speicherort des Speichers **28** an welchem der Zugriff begehrt wird, für das jeweilige Ursprungselement erlaubt ist, und (6) dass die Übertragungsgrösse für den Zugriffs innerhalb von vorgegebenen Grenzen gelegen ist. Die erste Prüfung erfolgt beim Paketempfänger **96** durch die CRC-Prüflogik **106**, wie weiter oben beschrieben. Falls es sich herausstellt, dass das empfangene Nachrichtenpaket einen fehlerhaften CRC hat (oder wenn es mit einem Befehlssymbol "This Packet Bad" (TPB) (Paket schlecht) gekennzeichnet ist, siehe weiter unten) wird das Paket ausgeschieden und der Zugriff wird verweigert.

[0186] Die in dem Header des Nachrichtenpakets enthaltene Bestimmungsort-ID wird mit der dem empfangenden Element zugewiesenen Bestimmungsort-ID verglichen, um zu gewährleisten, dass der Bestimmungsort des Pakets korrekt ist (d. h. wenn das Paket von einer CPU empfangen wird, dass die richtige CPU **12** als Bestimmungsort angeführt ist). Ein Nichtübereinstimmen der Daten deutet darauf hin, dass das Paket in irgendeiner Weise fehlgeleitet worden ist, weshalb das Paket auch in diesem Fall ausgeschieden und der Speicherzugriff natürlich verweigert wird.

[0187] Die verbleibenden Prüfungen erfolgen insofern als im Speicher **28** ein Zugriffsvalidierungseintrag (ZVÜ) (**Fig.** 13A) für jedes Systemelement aufbewahrt wird, dem zumindest in gewisser Form Zugriff auf den Speicher des Elements gewährt wird auf dessen Speicher gerade zugegriffen wird. Das Adressfeld des eingehenden Pakets wird als Hinweismarke auf den Speicherort verwendet, welcher den ZVÜ-Eintrag für das in der Ursprungsort-ID angegebene Systemelement enthält. Die ZVÜ-Logik verwendet die Validierungsinformation des ZVÜ-Eintrags, um zu ermitteln, welcher Zugriffstyp dem Ursprungselement des Nachrichtenpakets gestattet ist.

[0188] Somit wird das Ursprungsort-ID-Feld des empfangenen Nachrichtenpakets dazu verwendet, um zu bestimmen, ob der Ersteller des Pakets ein Zugriffsrecht auf den Speicher **28** der CPU **12** hat. Diese Prüfung bedingt das Vergleichen des Ursprungsort-ID-Feldes auf dem Paket-Header mit einem Abschnitt eines ZVÜ-Eintrags (Ursprungsort-ID), um zu bestimmen, ob dem bestimmten Ursprungselement Zugriff auf die Einrichtungen des bestimmten Empfängers gewährt werden soll.

[0189] Das Typ-Feld des Pakets, welches den gewünschten Zugriffstyp angibt (z. B. eine Speicherlese- oder -schreiboperation), wird geprüft, um zu bestimmen, ob der begehrte Zugriffstyp für das von dem Nachrichtenpaket identifizierte Ursprungselement erlaubt ist, oder ob das Paket eine unaufgefordert eingehende Antwort ist (und als solche als Fehler gelöscht wird).

[0190] Schliesslich werden noch der angeforderte Speicherort und die mögliche Grösse einer Übertragung überprüft, um zu sehen, ob auch diese für das betreffende Ursprungselement des Nachrichtenpakets erlaubt sind.

[0191] Der Zugriffsvalidierungsmechanismus der Schnittstelleneinheit **24a** gemäss der ZVÜ-Logik **88** ist im Detail in **Fig.** 11 dargestellt. Eingehende Nachrichtenpakete, die auf den Speicherplatz der CPU **12** zugreifen wollen, lassen ausgewählte Abschnitte aus ihren Headern von dem Paketempfänger **96** (**Fig.** 5) an ein ZVÜ-Eingangsregister **170** der ZVÜ-Logik **90** übertragen. Das ZVÜ-Eingangsregister **170** empfängt daher von dem eingehenden Nachrichtenpaket die Ursprungsort-ID, das Länge-Feld (Len), welches den Datenumfang angibt, der in den Speicher **28** geschrieben werden soll bzw. aus diesem ausgelesen werden soll, eine Adresse (ZVÜ-Seitennummer), die auf den Eintrag im Speicher **28** verweist, welcher den entsprechenden ZVÜ-Eintrag enthält, den Offset in diese Speicherseite, auf welchen der ZVÜ-Eintrag verweist, und den gewünschten Zugriffstyp (Typ). Diese Werte sind jeweils in den Registersegmenten **170a**, **170b**, ..., **170e** des ZVÜ-Eingangsregisters enthalten.

[0192] Das in dem ZVÜ-Eingangsregister **170** enthaltene ZVÜ-Seitennummernfeld wird an die ZVÜ-Adresslogik **172** gekoppelt, wo es mittels der kombinatorischen Logik **176** mit dem Inhalt eines ZVÜ-Basisregisters **174** kombiniert wird, wodurch die Adresse des für die Validierung benötigten ZVÜ-Eintrags erzeugt wird. Das ZVÜ-Basisregister **174** enthält die im Speicher enthaltene Startadresse der gesamten ZVÜ-Tabelle. Unter Verwendung der so erzeugten Adresse greift die ZVÜ-Adresslogik **172** auf den Speicher **28** zu, um diesen ZVÜ-Eintrag zu erhalten, der daraufhin in das ZVÜ-Eintragsregister **180** geladen wird.

[0193] Die ZVÜ-Logik **172** enthält auch ein ZVÜ-Maskenregister **175** zum Erkennen von ZVÜ-Seitennummeradressen, die nicht in den Adressenbereich fallen, welche der ZVÜ-Tabelle zugewiesen worden ist. Die Regeln sind dabei entsprechend gestaltet, dass wenn irgendeine Bitposition des ZVÜ-Maskenregisters **175** eine 0 ist, das entsprechende Bit der ZVÜ-Seitennummernadresse ebenfalls 0 sein muss; andernfalls erkennt die Maskenprüflogik **177** einen Maskenfehler, was dazu führt, dass der Zugriff auf den Speicher **28** verweigert wird. Die Generierung der ZVÜ-Eintragsadresse und die Maskierungsoperation sind in **Fig.** 12 besser veranschaulicht.

[0194] Wie in **Fig.** 12 in Diagrammform veranschaulicht, wird der höherwertige 8-Bit-Abschnitt des 20-Bit-ZVÜ-Seitennummernwertes in dem Registersegment **170e** mit dem Inhalt des ZVÜ-Basisregisters **174** summiert, um den höherwertigen Abschnitt (Bits **16-31**) der ZVÜ-Tabelleneintragsadresse zu erzeugen. Gleichzeitig bilden die verbleibenden (niederwertigen) 12 Bits der ZVÜ-Seitennummernadresse aus dem Registersegment **170e** direkt einen Abschnitt der ZVÜ-Eintragsadresse. Da es sich bei den ZVÜ-Einträgen mengenmässig um 16-Byte-Wörter handelt, sind sie an Vierfachwortgrenzen angeordnet; daher sind die niederwertigen 4 Bits der ZVÜ-Eintragsadresse, wie aus der Darstellung hervorgeht, stets 0.

[0195] In **Fig.** 12 wird auch die Maskierungsoperation veranschaulicht. Die höherwertigen zwei Bytes der ZVÜ-Seitennummernadresse werden mit der in dem Maskenregister **175** enthaltenen Maske verglichen. Wenn an einer Bitposition des Maskenregisters, welche eine 0 enthält, in den höherwertigen zwei Bytes eine entsprechende Bitposition mit einer "I" erkannt wird, gibt das Maskenregister ein "Maskenfehler"-Signal aus, wodurch der Zugriff auf den Speicher **28** verweigert wird und ein Interrupt generiert und an die Interrupt-Logik **86** (**Fig.** 5) freigegeben wird, das den Prozessor **20** zu einer entsprechenden Aktion veranlasst.

[0196] Dank der Maskierungsoperation kann die Grösse der ZVÜ-Eintragstabelle verändert werden. Auf den Inhalt des ZVÜ-Maskenregisters **175** kann von dem Prozessor **20** aus zugegriffen werden, wodurch es den Prozessoren **20** möglich wird, wahlweise die Grösse der ZVÜ-Eintragstabelle auszuwählen. Eine maximale ZVÜ-Tabellengrösse erlaubt eine Verifizierung (und Übersetzung) jeder beliebigen 32-Bit-TNet-Adresse; das bedeutet, dass eine ZVÜ-Eintragstabelle maximaler Grösse in der Lage ist, 2^{20} verschiedene Seitenadressen zu verifizieren und zu übersetzen. Eine ZVÜ-Tabelle minimaler Grösse ermöglicht die Verifizierung und Übersetzung jeder beliebigen 24-Bit-TNet-Adresse (d. h. jener TNet-Adressen, deren höherwertige 8 Bits Null sind). Eine ZVÜ-Tabelle minimaler Grösse kann 2''' verschiedene Seitenadressen verifizieren und übersetzen.

[0197] Da ein ZVÜ-Tabelleneintrag 16 Byte gross ist, benötigt somit eine ZVÜ-Tabelle maximaler Grösse **16** Megabytes an zugeordnetem Speicherplatz. Die Maskierungsoperation, die von dem Inhalt des ZVÜ-Maskenregisters **175** und der ZVÜ-Adressenlogik **172** durchgeführt wird, macht es jedoch möglich, die ZvÜ-Grösse an die Bedürfnisse des Systems anzupassen. Ein Verarbeitungssystem **10**, welches eine grössere Anzahl von externen Elementen beinhaltet (wenn z. B. die Anzahl der E/A-Geräte in dem System gross ist), benötigt eine grosse Anzahl an TNet-Adressen, und von entsprechenden ZVÜ-Einträgen, und muss daher eine grössere Menge an Speicherplatz des Speichers **28** für die ZVÜ-Einträge bereithalten. Umgekehrt hat ein kleineres Verarbeitungssystem **10** mit einer kleineren Anzahl von externen Elementen keinen so grossen Bedarf an TNet-Adressen, so dass eine kleinere ZVÜ-Tabelle verwendet werden kann, wodurch Speicherplatz eingespart werden kann. Bei einem kleineren System werden daher die höherwertigen Bits nicht verwendet werden (bzw., genauer gesagt, sollten sie nicht verwendet werden).

[0198] Wenn eine kleinere ZVÜ-Tabelle zulässig ist, sollten die höherwertigen Bits der TNet-Adresse NULL sein; jeder Versuch, einen ZVÜ-Tabelleneintrag mit einer TNet-Adresse zu suchen, die ausserhalb des Bereichs für das jeweilige System gelegen ist, führt zu einem Fehler. Die Funktion der Maskierungslogik ist es, solche Fehler unter Verwendung des Maskenregisters **175** zu erkennen. Somit wird die zu einem bestimmten Zeitpunkt für eine CPU **12** (oder für jedes andere Systemelement, das diese Validierungstechnik verwendet) zulässige Grösse des ZVÜ-Tabellenumfangs durch den Inhalt des Maskenregisters **175** mit Hilfe von Bitpositionen, die auf eine logische "EINS" gesetzt sind, angegeben. Bitpositionen des Maskenregisters **175**, die auf eine logische "NULL" gesetzt sind, deuten auf eine nichtexistierende TNet-Adresse hin, die ausserhalb der Grenzen des Verarbeitungssystems **10** gelegen ist. Ein empfangenes Paket mit einer TNet-Adresse ausserhalb des zulässigen TNet-Bereichs hat seine Bitpositionen an solchen Stellen auf eine logische EINS gesetzt, an denen sie NULL sein sollten. Die ZVÜ-Adressenlogik **172** erkennt diese bereichsexterne TNet-Adresse und führt zur Ausgabe eines ZVÜ-Fehler-Interrupts.

[0199] Für einschlägig gebildete Fachleute wird dadurch klar, dass die in **Fig.** 12 veranschaulichte Technik, darüber hinaus, dass sie in der Lage ist, die Grösse der in dem Speicher **28** zu verwaltenden ZVÜ-Tabelle zu verändern, auch imstande ist, die ZVÜ-Tabelle innerhalb des Speichers **28** mit einer gewissen Flexibilität anzuordnen. **Fig.** 12 zeigt, dass die ZVÜ-Tabelle an einer Grenze mit einer Potenz von 2^{17}(128K) angeordnet werden kann.

[0200] Jeder ZVÜ-Eintrag ist ein Vierfachwort mit **128** Bit, das jene Felder enthält, die während des Validierungsprozesses in dem ZVÜ-Eintragsregister gehalten werden, wie in **Fig.** 11 veranschaulicht. ZVÜ-Einträge haben zwei Basisformate: Normal und Interrupt. Das Format eines normalen ZVÜ-Eintrags wird in **Fig.** 13A veranschaulicht (und bis zu einem gewissen Ausmass in **Fig.** 11, in welcher der Inhalt des ZVÜ-Eintragsregis-

ters **180** gezeigt wird); das Interrupt-Format wird in **Fig.** 13C veranschaulicht. Bevor mit der Abhandlung der ZVÜ-Logik **90** fortgesetzt wird, mag ein besseres Verständnis der Bedeutung und des Inhalts eines ZVÜ-Eintrags hilfreich sein.

[0201] Der normale ZVÜ-Eintrag beinhaltet, wie in **Fig.** 13A gezeigt, ein **52** Bit umfassendes 'Physikalische Seitenzahl'-Feld. Der Inhalt diese Feldes gibt die physikalische Adresse der Seite in dem Speicher **28** an, innerhalb derer dem anfordernden Ursprung des Nachrichtenpakets ein Zugriff gewährt wird. (Typischerweise umfasst jede Speicherseite **4K** (4096) Byte Speicherorte.) Der Inhalt des 'Physikalische Seitenzahl'-Feldes wird mit dem Inhalt des **12** Bit umfassenden Offset-Feldes **170d** verkettet, das in dem ZVÜ-Eingangsregister **170** gehalten wird (aus dem Adressfeld des zu validierenden Nachrichtenpakets entnommen). Das Ergebnis ist die gesamte physikalische Adresse des Ortes innerhalb des Speichers **28**, an welchem Daten geschrieben werden, bzw. von welchem Daten ausgelesen werden – sofern die Validierung erlaubt wird.

[0202] Der Zugriff kann zwar für alle Speicherorte einer bestimmten 4K-Seite gewährt werden, er kann aber auch auf bestimmte Abschnitte dieser Seite beschränkt werden. Zur Realisierung der zuletztgenannten Einschränkungen beinhaltet der ZVÜ-Eintrag zwei 12-Bit-Felder (Obergrenze, Untergrenze; **Fig.** 13A), die die obere und die untere Grenze innerhalb der angegebenen Seite des Speichers **28** definieren, innerhalb derer ein Zugriff gewährt wird. Insbesondere das 'Untergrenze'-Feld des ZVÜ-Eintrags spezifiziert den Offset zu der Speicherseite, den das Byte mit dem niedrigsten Wert aufweist, auf welchen dieser ZVÜ-Tabelleneintrag zutrifft. Das 'Obergrenze'-Feld spezifiziert den Offset in die Speicherseite, den das Byte mit der höchsten Adresse aufweist, auf das der ZVÜ-Eintrag zutrifft. Versuche auf Speicherorte zuzugreifen, die über diesen Wert hinausgehen (z. B. den Offset-Wert **170d** plus den Inhalt des 'Len'-Feldes **170b** des ZVÜ-Eingangsregisters **170**) führen zu einem Fehler, der über ein Interrupt an den Prozessor freigegeben wird.

[0203] Ein 12-Bit-"Erlaubnisse"-Feld ist in dem ZVÜ-Eintrag enthalten, um die Erlaubnisse zu spezifizieren, die dem anfordernden Ursprungselement gemäss dem ZVÜ-Eintrag gewährt werden. Das 'Erlaubnisse'-Feld wird in **Fig.** 13B veranschaulicht, in welcher gewisse 'Erlaubnisse'-Unterfelder (E, PEX, PEY, I, C, W, R und B) die folgenden Qualifikationen für den Speicherzugriff angeben:

[0204] E: (Error Enable) Durch diesen ZVÜ-Eintrag geleitete, fehlerhafte Zugriffe werden nur dann (an die Interrupt-Logik) gemeldet, wenn dieses Feld auf einen von zwei besonderen Zuständen gesetzt ist (z. B. auf eine "EINS").

[0205] PEX: (Path enable X) Der Zustand dieses 1-Bit-Feldes ist auf eine "EINS" gesetzt, um es zu ermöglichen, dass Nachrichtenpakete, bei deren Empfang das "Pfad"-Bit in dem Header gleich Null ist, diesen ZVÜ-Eintrag verwenden (sofern alle anderen anwendbaren Erlaubnisbedingengen erfüllt werden). Ist dieses Bit auf eine "NULL" gesetzt, so wird Nachrichtenpaketen, die über den "X-Pfad" (Pfad = 0) empfangen wurden und auf die der ZVÜ-Eintrag anwendbar ist, der Zugriff verweigert. Zugriffsverweigerungen werden auf der Ebene der Interrupt-Logik als Interrupts protokolliert und an den Prozessor **20** gemeldet-sofern das E-Feld auf einen Zustand gesetzt ist ("EINS"), der das Melden von Fehlern erlaubt.

[0206] PEY: (Path Enable Y) Dieses 1-Bit-Feld funktioniert in derselben Weise wie das PEX-Feld, ausser dass es auf Nachrichtenpakete angewendet wird, die mit einem auf Eins gesetzten Pfadbit empfangen werden.

[0207] I: (Interrupt) Wenn dieses Bit gesetzt ist (z. B. auf eine "EINS", erhalten die anderen Felder (Obergrenze, usw.) neue Definitionen zum Verarbeiten von Interrupt-Schreiboperationen und zum Verwalten von Interrupt-Warteschlangen. Dies wird weiter unten in Verbindung mit der Beschreibung der Interrupt-Logik **86** genauer erläutert.

[0208] C; (Cachespeicherkohärenz) Hierbei handelt es sich um ein 2-Bit-Feld, das entsprechend codiert ist, um zu spezifizieren, wie Anforderungen für Schreiboperationen in den Speicher **28** behandelt werden. Ist dieses Feld in einen Zustand gesetzt, so werden die angeforderten Schreiboperationen normal verarbeitet; ist das Feld in einen zweiten Zustand gesetzt, so werden Schreibanforderungen, die sich auf Adressen beziehen, bei denen ein Teil einer Cachezeile in der Ober- oder Untergrenze des in dem ZVÜ-Eintrag abgebildeten Speicherbereichs enthalten ist, in die Cachespeicherkohärenz-Warteschlange geschrieben, die von der weiter unten beschriebenen Interruptbehandlungsroutine **250** (**Fig.** 14A) aufrechterhalten wird. Dies ermöglicht es der CPU **12**, Schreibtransfers in eine Benutzerdatenstruktur oder einen Pufferspeicherbereich im Speicher **28** zu verwalten, bei dem keine vollständige Cachezeilenüberdeckung gegeben ist. Ist das Feld in einen dritten Zustand gesetzt, so werden alle Schreibanforderungen, die auf diesen ZVÜ-Eintrag zugreifen, in die Cachespeicherkohärenz-Warteschlange geschrieben. Ist das Feld in einen vierten Zustand gesetzt, so wird auf die durch diesen ZVÜ-Eintrag referenzierten, physikalischen Speicherorte unter Verwendung von Hardware-Kohärenzmechanismen zugegriffen.

[0209] W: (Write Access) Der Zustand dieses 1-Bit-Feldes gewährt bzw. verweigert einem anfordernden Ursprungselement den Speicherschreibzugriff-innerhalb eines Speicherbereichs, der von dem 'Obergrenze'-Feld und dem 'Untergrenze'-Feld angegeben wird.

[0210] R: (Read Access) Der Zustand dieses 1-Bit-Feldes bestimmt, ob das anfordernde Ursprungselement eine Speicherzugriffserlaubnis für Leseoperationen hat – innerhalb des angegebenen Speicherbereichs.

[0211] B: (Barrier Access) Der Zustand dieses 1-Bit-Feldes bestimmt, ob das anfordernde Ursprungselement

einen Speicherzugriff für Barrieren-Operationen hat – innerhalb des angegebenen Speicherbereichs.

[0212] Schliesslich gibt ein 20 Bit umfassendes "Ursprungsort-ID"-Feld ("Source-ID") des ZVÜ-Eintrags den konkreten Ursprungsort bzw. das Ursprungselement an, auf den/das die Erlaubnisinformation des ZVÜ-Eintrags sich bezieht.

[0213] Um nun wieder zu der in **Fig.** 11 veranschaulicrten ZVÜ-Logik zurückzukehren, sei festgehalten, dass nachdem die Adresse des ZVÜ-Eintrags gebildet worden ist, auf den Eintrag zugegriffen wird und dieser in dem ZVÜ-Tabelleneintragsregister **180** zwischengespeichert wird. Der Inhalt des in dem ZVÜ-Eintragsregister **180** enthaltenen "Erlaubnisse"-Feldes wird mit dem angeforderten Zugriffstyp verglichen, wie er in dem von der Zugriffslogik **184** in dem ZVÜ-Eintragsregister gehaltenen 'Typ'-Feld angegeben ist. Stimmt der angeforderte Zugriff nicht mit dem erlaubten übereinstimmt, so wird der Zugriff verweigert und die Zugriffslogik **184** gibt ein Fehlersignal ("No") aus, wodurch über die Fehlergenerierungslogik, welche das ODER-Glied **184** und das UND-Glied **186** umfasst, ein ZVÜ-Fehler-Interruptsignal generiert wird. Der Zugriff wird verweigert, wenn der gewünschte Zugriffstyp nicht in den 'Erlaubnissen' enthalten ist.

[0214] Das 'Ursprungsort-ID'-Feld des ZVÜ-Eintrags, auf den zugegriffen werden soll (in dem ZVÜ-Eintragsregister **180** als "src ID"-Wert angegeben) spezifiziert den Ursprungsort, welcher dem benutzen ZVÜ-Eintrag entspricht und wird mittels der Vergleichslogik **190** mit der Ursprungsort-ID verglichen, die in dem anfordernden Nachrichtenpaket enthalten ist. Auch hier wieder führt ein Nichtübereinstimmen der Daten dazu, dass die Vergleichslogik **190** die Generierung eines ZVÜ-Fehler-Interrupts bewirkt und dass der Zugriff verweigert wird.

[0215] Gleichzeitig wird das 'Untergrenze'-Feld des ZVÜ-Eintrags (in dem ZVÜ-Eintragsregister **180** als "1wr bnd" angegeben, wie in **Fig.** 11 dargestellt) an die Vergleichslogik **194** angelegt, wo es mit dem in dem ZVÜ-Eingangsregistersegment **170d** enthaltenen 'Offset'-Wert verglichen wird. Ist der 'Offset'-Wert geringer als der in dem 'Untergrenze'-Feld des ZVÜ-Eintrags enthaltene Wert, was darauf hindeutet, dass ein Zugriff ausserhalb des erlaubten Seitenabschnitts erfolgen würde, so initiiert die Vergleichseinrichtung **194** ein Signal, welches über das ODER-Glied **184** und das UND-Glied **186** ein ZVÜ-Fehler-Interrupt generiert, wodurch der Zugriff auf den Speicher **28** verweigert wird.

[0216] In ähnlicher Weise vergleicht die Vergleichslogik **196** das 'Obergrenze'-Feld ("upr bnd" im ZVÜ-Eintragsregister **180**) mit der-von der Addier-Logik **200** bezogenen-Summe des 'Len'-Feldes (d. h. der Zahl der zu schreibenden Datenbytes) und des 'Offset'-Feldes, um zu bestimmen, ob die Menge der zu schreibenden Daten (sofern es sich um eine Anforderung nach einer Schreiboperation handelt) die dem anfordernden Ursprungselement zugeteilte (und in dem 'Untergrenze'- und dem 'Obergrenze'-Feld des Eintrags definierte) Menge an Speicherplatz überschreitet.

[0217] Wird eine Zugriffsanforderung validiert, so wird der Inhalt des 'Physikalische Seitenzahl'-Feldes (phys pg #) des ZVÜ-Eintragsregisters **180** zusammen mit dem 'Offset' des ZVÜ-Eingangsregisters **170**, wie weiter oben im Zusammenhang mit **Fig.** 12 erklärt, dazu verwendet, um den Speicherort zu adressieren, an welchem der Zugriff stattfinden soll.

[0218] Die Übertragung von Nachrichtenpaketen zwischen Elementen des Systems **10** (z. B. einem Gerät **17** und einer CPU **12**; **Fig.** 1) wird weiterhin auch insofern in einer neuartigen Weise verwendet, als auf diesem Weg Interrupts verteilt werden, die dazu dienen, eine Aktivität anzufordern, eine Benachrichtigung über eine Aktivität zu senden, oder über das Eintreten von Fehlern zu informieren. Die Interrupt-Nachrichtenübermittlung verwendet somit das TNet-Netzwerksystem in derselben Weise wie jeder andere Datenaustausch zwischen Elementen und umfasst einen Ablauf in drei Schritten: (1) die Erstellung und den Versand des Interrupt-Nachrichtenpakets durch ein Ursprungselement; (2) die Übertragung des Interrupt-Nachrichtenpakets durch das TNet-Netzwerk zu seinem Bestimmungsort; und (3) Interpretation und Freigabe im Hinblick auf die Durchführung einer entsprechenden Aktion am Bestimmungsort. Jedes Systemelement kann Empfänger eines Interrupt-Nachrichtenpakets sein. Ist der Bestimmungsort eine CPU, so handelt es sich bei dem Interrupt-Nachrichtenpaket praktisch um eine normale Anforderung nach einer Schreiboperation, bei welcher das 'Bestimmungsort-ID'-Feld des Nachrichtenpaket-Headers (**Fig.** 3B) die CPU identifiziert und das 'Adresse'-Feld den ZVÜ-Eintrag (Interrupt-Beschreiben) auswählt, welcher die Anweisungen enthält, wie das Interrupt-Nachrichtenpaket zu behandeln ist.

[0219] Die Autorisierung, eine Interrupt-Aktivität zu initiieren, bedarf auch der Validierung durch die ZVÜ-Logik **90**. Ein empfangenes Nachrichtenpaket enthält somit Interrupt-Daten, die den Interrupt erklären. Diese Interrupt-Daten sind in eine im Speicher **28** gehaltene, spezielle Warteschlange (Interrupt-Warteschlange) zu schreiben, und zwar unter Benachrichtigung der Prozessoren **20**, um anzuzeigen, dass ein Interrupt empfangen und "freigegeben" wurde und bereit ist zur Abfertigung durch die Prozessoren **20**. Da sich die Interrupt-Warteschlangen an speziellen Speicherorten befinden, sind die Interrupt-Daten bei Bedarf stets verfügbar.

[0220] Ein ZVÜ-Interrupteintrag für einen Interrupt kann einen von zwei Typen annehmen: es kann sich um einen Warteschlangen-Mehrfacheintrag-Interrupt oder um einen Warteschlangen-Einzeleintrag-Interrupt handeln. Das Format für beide Typen von ZVÜ-Interrupteinträgen ist im Grunde genommen dasselbe und dieses Format wird in **Fig.** 13C veranschaulicht.

[0221] Ein ZVÜ-Interrupteintrag für einen Warteschlangen-Mehrfacheintrag-Interrupt wird für empfangene

Nachrichtenpakete verwendet, welche entweder ursprünglich dazu erstellt wurden, um ein Interrupt zu übermitteln, oder aber unterwegs durch irgendein aussergewöhnliches Ereignis, das von einem Router **14** oder von der empfangenden CPU erkannt wurde (z. B. schlechter CRC), zu einem Interrupt geworden sind. Verwendet werden diese Einträge von der ZVÜ-Logik **90**, um das Nachrichtenpaket mehr oder weniger in der oben beschriebenen Art und Weise zu validieren, und von der Interrupt-Logik **86** (**Fig.** 5 und 14A), um eine kreisförmige Warteschlange in dem Speicher **28** zu bezeichnen, in welcher der Header und begleitende Daten des Interrupt-Nachrichtenpakets gespeichert werden. Ausserdem setzt die Interrupt-Logik **86** ein Bit in einem Interrupt oder "Ursachen"-Register **280** (**Fig.** 14A; im Detail weiter unten abgehandelt) um den Empfang und/oder die Erstellung eines Mehrfacheintrag-Interrupts anzuzeigen.

[0222] Ein ZVÜ-Interrupteintrag für ein Warteschlangen-Einzeleintrag-Interrupt funktioniert im wesentlichen in derselben Art und Weise, abgesehen davon, dass es sich bei der Interrupt-Datenstruktur, an welche der ZVÜ-Interrupteintrag die Nachrichtenpaketinformation zum Zweck der Speicherung leitet, um einen fixen (wenngleich modifizierbaren) Ort im Speicher **28** handelt.

[0223] Beide ZVÜ-Interrupteintragstypen (Mehrfacheintragund Einzeleintrag-Interrupts) verwenden das in **Fig.** 13C veranschaulichte Vierfachwortformat (**128** Bits). Ein 64-Bit-Segment ("Queue Base Addr"-Warteschlangenbasisadresse) des ZVÜ-Interrupteintrags wird als Hinweismarke auf den Ort der Interrupt-Warteschlange in dem Speicher **28** verwendet, an dem Interrupt-Daten geschrieben werden. Die Interrupt-Warteschlange ist in der Form einer FIFO strukturiert, sodass Interrupt-Daten von der Interrupt-Logik **86** empfangen und am hinteren Ende der FIFO-Warteschlange eingefügt werden und von dem Prozessor **20** anlässlich der Abfertigung des Interrupts vom vorderen Ende der Warteschlange entnommen werden.

[0224] Der ZVÜ-Interrupteintrag beinhaltet auch ein 20-Bit-Segment ("Ursprungsort-ID"), das Ursprungs-ID-Informationen enthält, welche die zu bedienende, externe Einheit über den Interrupt-Prozess identifiziert. Stimmt die Ursprungsort-ID des ZVÜ-Interrupteintrags nicht mit jener überein, die in dem Header des eingehenden Nachrichtenpakets enthalten ist, überein (Ursprungsort-ID; **Fig.** 3B), wobei die Übereinstimmung durch einen von der Vergleichslogik **190** vorgenommenen Vergleich ermittelt wird (**Fig.** 11), so wird der Zugriff auf die Interrupt-Warteschlange verweigert und ein ZVÜ-Fehler-Interrupt generiert.

[0225] Ein **12** Bit umfassendes "Erlaubnisse"-Segment des ZVÜ-Interrupteintrags enthält dieselben erlaubnisbezogenen Informationen wie weiter oben in Verbindung mit normalen ZVÜ-Einträgen beschrieben. Typischerweise jedoch ist ein Nachrichtenpaket, welches ein Interrupt übermittelt, als Anforderung für eine Schreiboperation strukturiert und strebt danach, die Interrupt-Daten, welche es überträgt, wie weiter unten beschrieben, in den Speicher **28**, d. h. in die Interrupt-Warteschlange zu schreiben. Ein ordnungsgemässer ZVÜ-Interrupt-Eintrag hat somit das 'Write Access'-Bit entsprechend gesetzt, um es zu ermöglichen, dass die Interrupt-Daten in die in dem Speicher **28** eingerichtete Interrupt-Warteschlange geschrieben werden können. Das Interrupt-Bit (I) des 'Erlaubnisse'-Feldes identifiziert, sofern es gesetzt ist, den ZVÜ-Interrupt-Eintrag als einen zur Validierung und Behandlung eines Interrupt-Nachrichtenpakets geeigneten Eintrag.

[0226] Schliesslich identifizieren vier je ein Byte umfassende Segmente ("c", "q", "1", und "z") des ZVÜ-Interrupt-Eintrags jeweils (1) die "Klasse" des Interrupts, welche verwendet wird, um das in dem Prozessor **20** zu (2) eine setzende Interruptebene zu bestimmen; Warteschlangennummer, die, wie weiter unten noch genauer beschrieben wird, dazu verwendet wird, ein Register auszuwählen, dessen Inhalt angibt, an welcher Stelle der jeweiligen (von dem 'Warteschlangen-Basisadresse'-Feld angegebenen) Warteschlange die Interrupt-Daten zu schreiben sind; (3) die Grösse bzw. die Menge an Speicherplatz, der an jedem Warteschlangen-Speicherort verfügbar ist, ausgedrückt in Form der Anzahl von Doppelwörtern, die dort gespeichert werden können; und (4) die Anzahl von in einem Warteschlangenendzähler, die dazu verwendet werden, um zu festzustellen, an welcher Stelle der Warteschlange die Daten geschrieben werden. Die Warteschlangen-Basisadresse (Queue Base Addr) und die Segmente c, q, 1 und z werden von der Interrupt-Logik dazu benutzt, um auf den Speicherort im Speicher **28** zu verweisen. Die Interrupt-Logik **86** beinhaltet vier "End"-Zähler, von denen ein jeder auf eine von vier Warteschlangen verweist, in welche Interrupt-Daten eingefügt werden können. Welcher der vier Zähler ausgewählt wird, ist vom Inhalt des Segments q des ZVÜ-Interrupt-Eintrags abhängig. Das andere Ende der Warteschlange, d. h. der Punkt, an dem Interrupt-Einträge entnommen werden, wird durch einen der vier "Anfangs"-Zähler angegeben. Die Grösse der Anfangs- und Endzähler (ausgedrückt in Bitzahlen) wird durch das Unterfeld z angegeben (negativ vorgespannt bis 9), wie weiter unten in Tabelle 2 angegeben. Die Warteschlangenendzählergrösse wird verwendet, um zu bestimmen wann der Endzeiger auf den Wert Null zurückspringt. Die Anzahl der Wörter (Bytes) dividiert durch die Grösse eines jeden Eintrags ergibt die Anzahl der Warteschlangeneinträge. Die kürzeste Warteschlange könnte unter Umständen nur 32 Einträge enthalten (4 kB-Warteschlange dividiert durch 128 Byte Einträge) und erfordert 4 kB, während die längste Warteschlange nicht weniger als 32.768 Einträge aufweisen könnte (512 kB dividiert durch 16 Bytes pro Eintrag).

TABELLE 2

Z	Erklärung
0	512 Doppelwörter (4K Bytes)
1	1K Doppelwörter (8K Bytes)
2	2K Doppelwörter (16K Bytes)
3	4K Doppelwörter (32K Bytes)
4	8K Doppelwörter (64K Bytes)
5	16K Doppelwörter (128K Bytes)
6	32K Doppelwörter (256K Bytes)
7	64K Doppelwörter (512K Bytes)
8-15	Nicht in Verwendung

[0227] Die Grösse eines jeden Interrupt-Eintrags in der Interrupt-Warteschlange wird durch das Feld **1** angegeben, und zwar in Form von 16-Byte-Wörtern, wie in der folgenden Tabelle 3 veranschaulicht.

TABELLE 3

1	Erklärung
0	1 Vierfachwort (16 Bytes)
1	2 Vierfachwörter (32 Bytes)
2	4 Vierfachwörter (64 Bytes)
3	8 Vierfachwörter (128 Bytes)
4-15	Nicht in Verwendung (reserviert)

Interruptbehandlung:

[0228] Wie bereits weiter oben gezeigt, liegt ein Neuigkeitsmerkmal der vorliegenden Erfindung in der Fähigkeit, die TNet-Netzwerk-Nachrichtenübermittlungsfähigkeit dazu zu benutzen, um Interrupts an die CPU **12** zur Abfertigung zu nutzen. Zum Beispiel kann es sein, dass eine E/A-Einheit aus einer Reihe von Gründen nicht in der Lage ist, eine von einer CPU ausgegebene Lese- oder Schreibtransaktion fertigzustellen, sei es, aufgrund einer untauglichen Adresse in dem Nachrichtenpaket, mit dem die Transaktion übermittelt wurde, oder aufgrund des Empfangs des Nachrichtenpakets mit einem CRC-Fehler, oder weil festgestellt wird, dass das empfangene Nachrichtenpaket eine Bestimmungsort-Adresse aufwies, die nicht jener des Empfängers entsprach. Diese und andere Fehler, Ausnahmefälle und Unregelmässigkeiten, die von den E/A-Einheiten bzw. den E/A-Schnittstellenelementen festgestellt werden, können eine Situation herbeiführen, in der eine Intervention der CPU erforderlich wird. Bei Systemen nach dem Stand der Technik sind solche Situationen Gegenstand von Interrupts; nicht anders ist es auch im Fall der vorliegenden Erfindung, ausser dass solche Interrupts nicht in der bisher üblichen Weise übertragen werden-mit wenig bis überhaupt keiner Information über die Ursache der Unterbrechung und über ein Signalkabel, das ausschliesslich für solche Zwecke reserviert war-sondern direkt über das Nachrichtensystem, das auch für die E/A-Elemente des Systems verfügbar ist. Dieses Merkmal verringert nicht nur den Bedarf an zusätzlichen Signalleitungen (wodurch der für die Signalleitungen vorgesehene Platz anderweitig genutzt werden kann), sondern schafft auch eine Einrichtung, die bedeutend mehr Informationen über die Ursache für das Interrupt bereitstellen kann, so dass sich ein entsprechender Nachforschungsaufwand für die CPU erübrigt.

[0229] Gemäss diesem Merkmal wird ein eingehendes Nachrichtenpaket, welches einen in den Speicher **28** zu schreibenden Interrupt-Bericht enthält, vorerst zum Zweck der Validierung an die ZVÜ-Logik **90** geleitet (**Fig.** 11). Die ZVÜ-Logik **90** bestimmt auch, ob es sich bei dem Nachrichtenpaket um eine reguläre E/A-Schreibanforderung, ein Interrupt, oder möglicherweise um einen fehlgeleiteten Zugriff auf den Speicher

28 handelt, welcher verboten ist. Der Inhalt des ZVÜ-Basisregisters **174** der ZVÜ-Logik **90** wird gemeinsam mit dem in dem ZVÜ-Eingangsregister **170** enthaltenen Seitenzahlfeld **170c** (**Fig.** 11) dazu benutzt, um einen Zeiger für den ZVÜ-Interrupt-Eintrag im Hauptspeicher zu generieren, und zwar in derselben Weise wie weiter oben für normale Nachrichtenpakete beschrieben (d. h. die Zugriffsanforderung auf den Speicher **28** zum Zweck des Lesens oder Schreibens von Daten). Auf den dergestalt über die zusammengesetzte Adresse identifizierten ZVÜ-Eintrag wird vom Speicher **20** aus zugegriffen und dieser wird, wie in **Fig.** 14A im Detail veranschaulicht, zum weiteren Gebrauch durch die Interrupt-Logik **86** der Schnittstelleneinheit **24** (**Fig.** 5) in das ZVÜ-Eintragsregister **180** gestellt.

[0230] Es ist die Interrupt-Logik **86**, die für die Behandlung der Interrupt-Information verantwortlich ist, nachdem das Nachrichtenpaket, mit dem die Interrupt-Information übertragen wird, die ZVÜ-Logik **90** erfolgreich durchlaufen hat. Die Interrupt-Logik **86** beinhaltet, wie in **Fig.** 14A gezeigt, einen Multiplexer (MUX) **252**, welcher die Inhalte von vier Warteschlangenendregistern **256** empfängt und daraus eine Auswahl trifft. In ähnlicher Weise empfängt ein MUX **254** den Inhalt von vier Warteschlangenanfangsregistern **262** empfängt und daraus eine Auswahl trifft. Der Auswahleingang (s) eines jeden MUX **252**, **254** ist entsprechend verkoppelt, um den Inhalt des Segments "q" des abgerufenen (und dem Interrupt-Nachrichtenpaket entsprechenden) ZVÜ-Eintrags zu empfangen, der in dem ZVÜ-Eintragsregister **180** gehalten wird. Es ist der "q"-Wert, der dazu verwendet wird, um auszuwählen welcher aus jeder Gruppe von Warteschlangenregistern **256**, **262** zu verwenden ist.

[0231] Obwohl theoretisch jede beliebige Anzahl von Warteschlangen in dem Speicher eingerichtet werden kann, um die Speicherung von Interrupt-Daten abzuwickeln, ist deren Anzahl aus weiter unten erläuterten Gründen bei der vorliegenden Erfindung auf vier beschränkt. Der Speicherort innerhalb des Speichers **28** einer jeden solchen Warteschlange ist in dem Wert für die Warteschlangenbasisadresse des ZVÜ-Eintrags, auf den zugegriffen wird, angegeben und dieser wird in dem Eintragsregister **180** ("phys pg #" in **Fig.** 11; "base" in **Fig.** 14A). Die Inhalte von vier Warteschlangenendregistern **256** bilden jeweils einen Offset in die bestimmte Warteschlange, die dann durch den Warteschlangenbasisadresswert angegeben wird. Der Inhalt des ausgewählten Warteschlangenendregisters **256** wird durch den Addierer **258** mit der Warteschlangenbasisadresse kombiniert und bildet die Eingabestelle in die bezeichnete Warteschlange an welchem die Interrupt-Daten geschrieben werden. Die vier Warteschlangenanfangs- und -endregister **262**, **256** schränken die Interrupt-Logik **86** darauf ein, nur vier Warteschlangen zu beandeln, es sollte jedoch einschlägig gebildeten Fachleuten klar sein, dass auch mehr oder weniger Warteschlangen gehandhabt werden können.

[0232] Die Register **256** spezifizieren den Speicherort des "Endes" von bestimmten Warteschlangen und zeigen auf den Warteschlangeneingang, an welchem die als nächstes empfangenen Interrupt-Daten abgelegt werden. Die vier Warteschlangenanfangsregister **262** spezifizieren das jeweils andere Ende der bestimmten Warteschlange.

[0233] Die aus der Kombination der Warteschlangenbasisadresse und dem Inhalt des ausgewählten Warteschlangenendregisters **256** erstellte Adresse ist vorzugsweise so gebildet, dass sie sich an Vierfachwortgrenzen ausrichten. Dies wird dadurch erreicht, dass die niederwertigen vier Bits der Warteschlangeneingangsadresse zwingend auf 0 gesetzt werden.

[0234] Die Bildung der Warteschlangeneingangsadresse wird in **Fig.** 14B in Diagrammform veranschaulicht, worin gezeigt wird wie die höherwertigen sieben Bits des 15 Bit umfassenden Inhalts des ausgewählten Endregisters **256** zu den Bitpositionen 12–31 des in dem ZVÜ-Eintragsregister **180** enthaltenen Warteschlangenbasisadressfeldes hinzugefügt werden; das Ergebnis dieser Summierung bildet die höherwertigen **20** Bits (Bitpositionen 12–31) der Warteschlangeneingangsadresse. Der niederwertige 8-Bit-Inhalt des ausgewählten Endregisters **256** wird direkt als die niederwertigen Bitpositionen 4–11 der Warteschlangeneingangsadresse verwendet. Wie weiter oben abgehandelt, werden zum Zweck der gewünschten Ausrichtung die niederwertigen vier Bits (Bitpositionen 0–3) der Warteschlangeneingangsadresse alle zwingend auf Null gesetzt.

[0235] Zusammenfassend kann gesagt werden, dass Nachrichtenpakete, welche Interrupts enthalten, zu Beginn in derselben Weise gehandhabt werden wie andere Nachrichtenpakete auch, und zwar insofern als sie praktisch nichts anderes als Anforderungen für das Schreiben von Daten in den Speicher **28** der empfangenden CPU **12** sind und diese Anforderung durch die ZVÜ-Logik **90** validiert werden muss. Aus dem Nachrichtenpaket entnommene Informationen werden daher in das ZVÜ-Eingangsregister **170** gestellt und Abschnitte daraus (die Felder **170c** und **170d**) verwendet, um aus dem Speicher **28** heraus die Lage eines ZVÜ-Eintrags zu ermitteln und auf diesen zuzugreifen. Bei dem ZVÜ-Eintrag handelt es sich, sofern das Nachrichtenpaket die richtige Information für die Interruptbehandlung enthalten hat, um einen Interrupt-ZVÜ-Eintrag, der in das ZVÜ-Eintragsregister **180** gestellt wird und verwendet wird, um das Interrupt zu verifizieren (authentifizieren) und in der Folge unter Verwendung der Interrupt-Logik **86** die Interrupt-Daten in einer der vier kreisförmigen Warteschlangen zu speichern, die von der in dem ZVÜ-Eintrag enthaltenen Basisadressinformation spezifiziert wird. Der Prozessor **20** bzw. die Prozessoren **20** wird/werden daraufhin benachrichtigt, wobei es letzteren freigestellt ist, zu entscheiden ob und wie das Interrupt behandelt wird.

[0236] Nachdem die Interrupt-Nachrichtenpaketdaten in der Warteschlange gespeichert worden sind, muss

die Adressierung im Hinblick auf den Empfang der Interrupt-Daten des nächsten Nachrichtenpakets aktualisiert werden. Nachdem die Interrupt-Daten in die ausgewählte Warteschlange geschrieben worden sind, wird der Inhalt des in dem ZVÜ-Tabelleneintragsregister **180** enthaltenen "1"-Feldes durch die Kombinationsschaltung **270** mit dem ausgewählten Warteschlangenendregister **256** kombiniert und das Ergebnis wird dann von der "mod z"-Schaltung **273** verarbeitet, um einen neuen Offset in die Warteschlange zu setzen, an welchem die Interrupt-Daten des nächsten Interrupt-Nachrichtenpakets gespeichert werden. Dieser neue Offset-Wert wird an das ausgewählte Warteschlangenendregister **256** rückübermittelt. Gleichzeitig werden die Ausgabedaten der Kombinationsschaltung **270** an die Vergleichsschaltung **272** angelegt. Die Interrupt-Anfragen sind so strukturiert, dass sie kreisförmig ausgebildet sind mit einer Modulgrösse z. Die "mod z"-Schaltung erzeugt Ausgabedaten, die die Zirkularität aufrechterhalten. Da der Warteschlangenendzeiger den nächsten Eingabepunkt in der Warteschlange anzeigt, und der Anfangszeiger angezeugt hat, wie viel Platz in Bezug auf den entsprechenden Endzeiger in der Warteschlange übrigbleibt, bedeutet es, dass die Warteschlange voll ist, wenn diese beiden Werte gleich sind. Somit ist es die Vergleichsschaltung **272**, die diese Bestimmung vornimmt, und zwar durch einen Vergleich des Anfangszeigers (der von dem ausgewählten Warteschlangenanfangsregister **262** bereitgestellt wird) mit dem Endzeiger, der als Ergebnis des letzten Eintrags erzeugt wurde. Wenn nun der Anfangspunkt und der Endpunkt für die Warteschlange gleich sind, so gibt die Vergleichsschaltung **272** ein "Queue Full"-Warnsignal (Warteschlange Voll) aus, das selber wiederum ein Interruptsignal ist. Das "Warteschlange Voll"-Warnsignal wird zu einem eigengenerierten Interrupt, welches an die Prozessoreinheiten **20** übermittelt wird, um diesen als Warnung zu dienen, dass wenn die Angelegenheit nicht unverzüglich gelöst wird, später empfangene Interrupt-Daten verlorengehen können, denn sollte sich die Warteschlange füllen, so werden zusätzliche Interrupt-Nachrichten ausgeschieden.

[0237] Eingehende Nachrichtenpaket-Interrupts bewirken, dass Interrupts an den Prozessor **20** freigegeben werden, indem sie zuerst eine aus einer Reihe von Bitpositionen eines Interrupt-Registers **280** setzen. Warteschlangen-Mehrfacheintrag-Interrupts werden in den Interrupt-Registern **280a** zur Freigabe an den Prozessor gesetzt; Warteschlangen-Einzeleintrag-Interrupts verwenden das Interrupt-Register **280b**. Welches Bit gesetzt wird, hängt von dem Klasse-Feld (c) des in dem ZVÜ-Eingangsregister **180** gehaltenen ZVÜ-Eintrag ab.

[0238] Zuerst bezugnehmend auf Warteschlangen-Mehrfacheintrag-Interrupts, gibt die Schnittstelleneinheit bald nachdem ein Warteschlangen-Mehrfacheintrag-Interrupt erkannt worden ist, ein entsprechendes Interrupt-Signal (I1) aus, das an die Decodierschaltung **283** angelegt wird. Die Decodierschaltung **283** empfängt und decodiert den 'Klasse (c)'-Wert von dem ZVÜ-Eintragregister **180**, um zu bestimmen, welcher der Registerpositionen des Registers **280a** zu setzen ist, wobei eine Vorausinformation bezüglich des empfangenen Interrupts an den/die Prozessor en) **20** bereitgestellt wird, d. h. (1) der Typ des freigegebenen Interrupts und (2) die Klasse dieses Interrupts.

[0239] In ähnlicher Weise bewirken Warteschlangen-Einzeleintrag-Interrupts bei ihrem Empfang, dass ein entsprechendes Interrupt-Signal (I2) ausgegeben und an die Decodierlogik **287** angelegt wird, welche auch den 'Klasse (c)'-Wert empfängt und decodiert, um zu bestimmen, welcher Bitabschnitt des Registers **280b** zu setzen ist.

[0240] Die Warteschlangenend- und -anfangsregister **256**, 262 sind auch an ein weiteres Paar von Multiplexern (MUXs) **276**, **274** gekoppelt. Durch den Inhalt eines Aktualisierungsregisters **278** wird ausgewählt, welches entsprechende Paar von Registern **256**, **262** von der Vergleichsschaltung **279** miteinander verglichen wird. Das Aktualisierungsregister ist durch den Prozessor **20** beschreibbar, damit ein Registerpaar zum Vergleich ausgewählt werden kann. Stellt es sich heraus, dass die beiden ausgewählten Register **256**, **262** inhaltsgleich sind, was darauf hindeutet, dass die betreffende Warteschlange leer ist, so werden die jeweiligen Interrupt-Register gelöscht. Das Klassenregister **281** wählt das Interrupt-Bit des Interrupt-Registers 280aden muss, das gelöscht werden muss, (als Klasse) aus.

[0241] Der folgende kurze Exkurs behandelt die beiden Grundtypen von Interrupts, welche die Prozessoren **20** betreffen: solche Interrupts, die von einem Nachrichtenpaket an die CPU **12** übermittelt werden, und solche, die von der CPU **12** selbst generiert werden, und daher auch als "eigengenerierte" Interrupts bezeichnet werden. Eigengenerierte Interrupts sind das Ergebnis von intern erkannten Fehlern, wie zum Beispiel das von der Vergleichsschaltung **272** der Interrupt-Logik **86** generierte "Warteschlange Voll"-Warnsignal. Sie beinhalten aber auch Ausnahmesituationen, die beim Empfang von Nachrichtenpaketen bemerkt werden, welche ursprünglich nicht als Interrupt-Pakete gesendet wurden. Solche Nachrichtenpaket-Interrupts beinhalten Fehler, die darauf zurückzuführen sind, dass herausgefunden wurde, dass ein Nachrichtenpaket ein schlechtes Befehlssymbol aufweist, oder dass das empfangene Nachrichtenpaket einen schlechten CRC aufweist (oder mit einer (weiter unten erklärten) TPB-Zustandsanzeige ("Paket schlecht") gekennzeichnet ist). Diese eigengenerierten Interrupts bewirken, dass bestimmte Bitpositionen eines eigengenerierten Registers **280c**, an welchem eigengenerierte Interrupts freigegeben werden, in derselben Art und Weise gesetzt werden, wie dies bei der Freigabe von Mehrfacheintrag- und Einzeleintrag-Interrupts erfolgt: nämlich durch das Setzen von Bitpositionen des Interrupt-Registers **180a**, **180b**. Ausserdem ist bei der in dem Speicher **28** gehaltenen ZVÜ-Tabelle die erste Einträgenummer für eigengenerierte ZVÜ-Interrupts reserviert. Wenn ein eigengeneriertes Inter-

rupt-Signal erzeugt wird, um das eigengenerierte Interrupt-Register **180c** zu setzen, so bewirkt dieses auch, dass auf den ZVÜ-Eintrag, welcher der Ausnahmesituation entspricht, die das Interrupt verursacht hat, zugegriffen wird und dass dieser in das ZVÜ-Eintragregister **180** der ZVÜ-Logik **90** geladen wird. Von diesem Punkt an wird das Interrupt in derselben Weise behandelt wie Interrupts, die über Nachrichtenpakete übertragen weren.

[0242] Einem jeden der Interrupt-Register **280a**, **280b** und **280c** sind bitweise jeweils entsprechende Maskenregister **282a**, **282b** und **282c** zugeordnet. Zu jeder Bitposition des Interrupt-Registers **280** (z. B. **280a**) existiert eine entsprechende Bitposition in dem Maskenregister **282** (z. B. **282a**). Wenn ein bestimmtes Bit des Maskenregisters **282** gesetzt wird, so wird das Erkennen des zugeordneten Interrupts verhindert.

[0243] Der Inhalt der Interrupt-Register **280** wird, wenn er durch den Inhalt der Maskenregister **282** hindurchgelassen worden ist, an die kombinatorische Logik **286** gekoppelt, die eine Mehrzahl von ODER-Elementen umfasst, wo er zu sieben Interrupt-"Freigaben" (Signalen) kombiniert wird. Die kombinatorische Logik **286** koppelt die sieben Interrupt-Freigaben an einen Auffangspeicher **288**, von wo aus sie an den Prozessor **20** (**20a**, **20b**) gekoppelt werden, der über ein Interrupt-Register zum Empfangen und Halten der Freigaben verfügt .

[0244] Ausserdem wird der Inhalt des Registers **288** an eine Vergleichsschaltung **289** angelegt und (vor jedem Takt, der das Register **288** lädt) mit dem Eingangsdaten des Registers **288** verglichen. Wird dabei ein Unterschied festgestellt, was darauf hindeutet, dass eine Veränderung bei den Interrupts stattgefunden hat (entweder ein Interrupt wurde durch den Prozessor **20** abgefertigt und seine Freigaben wurden dabei gelöscht, oder ein neues Interrupt wurde freigegeben), wird ein "CHANGE"-Signal an die Prozessorschnittstelle **60** ausgegeben, um sie zu informieren, dass eine Veränderung bei den Interrupt-Freigaben stattgefunden hat und dass sie dem Prozessor **20** die Veränderung mitteilen sollte.

[0245] Vorzugsweise ist das ZVÜ-Eintragsregister entsprechend konfiguriert, um wie ein Einzeilen-Cachespeicher betrieben zu werden, komplett mit Identifizierungskennzeichen und gültigem Bit. Das Identifizierungskennzeichen würde aus dem Abschnitt der TNet-Adresse bestehen, der dazu verwendet wird, um den ZVÜ-Eintrag in dem Systemspeicher **28** aufzufinden. Stimmt das Identifizierungskennzeichen nicht mit der TNet-Adresse eines eingehenden Pakets überein, so wird der korrekte ZVÜ-Eintrag aus dem Systemspeicher **28** ausgelesen und in das ZVÜ-Eintragsregister **206** eingelesen, wo es den alten ZVÜ-Eintrag ersetzt. Einschlägig vorgebildete Fachleute können unschwer erkennen, dass auch andere Cachespeicherorganisationen möglich sind, wie zum Beispiel Cache mit teilassoziativer Abbildung, Assoziativcachespeicher, Cache mit direkter Speicherplatzzuordnung, um nur einige zu nennen.

Kohärenz:

[0246] Datenverarbeitungssysteme, welche Cachespeicher verwenden, sehen sich schon seit langem immer wieder mit dem Problem der Kohärenz konfrontiert: hierbei gilt es zu gewährleisten, dass ein Zugriff auf den Cachespeicher oder den Hauptspeicher nie veraltete Daten rückliefert bzw. gute (aktuelle) Daten überschreibt. Es existieren zahlreiche Lösungen für dieses Problem, von denen viele auf den extensiven Einsatz von komplexer Hardware zurückgreifen. Das Kohärenzproblem tritt auch auf, wenn Daten von einem (CPU-)externen E/A-Gerät oder von einer anderen CPU **12** in den Speicher geschrieben werden, wie zum Beispiel im Zusammenhang mit dem System **10** (z. B. **Fig.** 2), bei dem Daten von der CPU **12B** in den Speicher **28** der CPU **12A** geschrieben werden. Eine Lösung besteht darin, zu gewährleisten, dass eingehende Daten in Speicherpuffer geschrieben werden, und zwar so, dass die Grenzen des Puffers an Cacheblockgrenzen ausgerichtet werden. Diese Lösung findet jedoch nur dann Anwendung, wenn sie zusammen mit Softwarekonfigurationen benutzt wird, die es erlauben, Cachespeicherblöcke für eingehende Daten ungültig zu machen und ein Rückschreiben von Cachespeicherblöcken, welche für abgehende Daten verwendet werden, zu erzwingen.

[0247] So existieren etwa traditionelle Techniken für die softwaremässige Handhabung von Kohärenzproblemen, welche für eingehende Leseanforderungen (von einem E/A-Gerät bzw. einer anderen CPU **13**), und für abgehende Lese- und Schreibanforderungen. Diese traditionellen Techniken eignen sich jedoch nicht zur Behandlung von eingehenden Schreibanforderungen, welche sich an einen E/A-Puffer im Speicher **28** richten, der nicht an Cachespeicherblockgrenzen ausgerichtet ist.

[0248] Die Anforderung der Ausrichtung von E/A-Puffern im Speicher an Cachespeicherblockgrenzen führt jedoch zu einem weniger flexiblen System, und einem System, das mit bestehender (Betriebssystem-)Software inkompatibel sein kann. Daher wird der erfindungsgemässe Interrupt-Mechanismus dazu verwendet, um Kohärenz in einer Weise zu erreichen, die es erlaubt, Datenpuffer im Speicher abzulegen, ohne dabei darauf achten zu müssen, dass die Grenze des betreffenden Puffers an den Grenzen des Cachespeicherblocks ausgerichtet sind.

[0249] In diesem Zusammenhang wird das Feld in dem ZVÜ-Tabelleneintragsregister **180** (**Fig.** 11), das die Obergrenze und die Untergrenze (upr bnd, lwr bnd) des Speicherbereichs **28** definiert, auf welchen das Ursprungselement des eingehenden Pakets zugreifen darf, an eine Grenzüberschreitungs-Prüfeinheit **219** (Bdry Xing) angelegt. Die Grenzprüfeinheit **219** empfängt auch eine Meldung betreffend die Grösse des Cache-

speicherblocks, mit dem zu arbeiten die CPU **12** konfiguriert ist, die Kohärenzbits ("c[1 : 0]") aus dem 'Erlaubnisse'-Feld des in dem ZVÜ-Eintragsregister **180** gehaltenen ZVÜ-Eintrags, und das 'Len'-Feld aus der dem ZVÜ-Eingangsregister **170** entnommenen Header-Information. Die Grenzüberschreitungs-Prüfeinheit stellt fest, ob die Daten des eingehenden Pakets nicht an einer Cachespeichergrenze ausgerichtet sind, und erzwingt – sofern die Kohärenzbits ("c[1 : 0]") korrekt gesetzt sind – den Abruf einer Adresse eines Interrupt-Eintrags, der dazu verwendet wird, auf die spezielle Kohärenz-Warteschlange zur Speicherung der Daten und des Headers des Pakets zu verweisen, das diese Daten enthält.

[0250] Im folgenden wird vorübergehend auf **Fig.** 29 Bezug genommen, in welcher ein Abschnitt **28'** eines von dem Speicher **28** (**Fig.** 2) einer CPU **12** implementierten Speicherplatzes veranschaulicht wird. Wie in **Fig.** 29 weiterhin dargestellt, sind in dem Speicherabschnitt **28'** drei Cachespeichergrenzen $\mathbf{CB_a}$, $\mathbf{CB_b}$, und $\mathbf{CB_c}$ enthalten, durch die zwei Cachespeicherblöcke C_BLK_a und C_BLK_b definiert werden. Angenommen, es wird ein Nachrichtenpaket mit einer Schreibanforderung empfangen (z. B. von einer anderen CPU **12**, oder einem E/A-Gerät), und die in dem Nachrichtenpaket enthaltenen und hier schraffiert dargestellten Daten sind in einen Speicherbereich **28** zu schreiben, welcher den Speicherabschnitt **28'** beinhaltet. Genau genommen werden durch die zu schreibenden Daten der Cachespeicherblock C_BLK_a nur teilweise, der Cachespeicherblock C_BLK_b und andere Cachespeicherblöcke jedoch zur Gänze überschrieben. Enthält der Cachespeicher **22** der CPU **12**, welcher beschrieben wird, den Cachespeicherblock C_BLK_b oder irgendeinen anderen Cachespeicherblock ausser dem Cachespeicherblock C_BLK_a (oder denjenigen Cachespeicherblock, welcher das andere Ende der eingehenden Daten enthält, sofern diese nicht mit einer Cachespeichergrenze übereinstimmen), so kann der Block als "ungültig" markiert werden, wodurch verhindert wird, dass er in den Speicher zurückgeschrieben wird, wo er die zuletzt empfangenen Daten überschreiben würde.

[0251] Wenn jedoch der Cachespeicher **22** den Cachespeicherblock C_BLK_a enthält, so muss die Grenzüberschreitungslogik **219** (falls sie durch Setzen von "c" in dem 'Erlaubnisse'-Feld aktiviert wurde; siehe dazu die **Fig.** 11 und 13B) der ZVÜ-Logik **90** (**Fig.** 11) das E/A-Paket erkennen und dabei den Cachespeichereintrag teilweise ungültig setzen, und ein Kohärenz-Interrupt erzwingen. Dies bewirkt das Abrufen eines Interrupt-Beschreibers, welcher einen Zeiger auf eine spezielle Interrupt-Warteschlange enthält, wodurch das gesamte eingehende TNet-Anforderungspaket in diese Warteschlange geschrieben wird.

[0252] Gleichzeitig wird ein Interrupt in das Warteschlangeninterrupt-Register **280** geschrieben, um den Prozessoren **20** zu signalisieren, dass ein Abschnitt der eingehenden Daten sich in der speziellen Warteschlange befindet.

[0253] Kurz gesagt, wenn ein eingehendes Paket Daten beinhaltet, die in den Speicher **28** zu schreiben sind, so prüft die Grenzüberschreitungslogik **219**, ob die Grenzen des Puffers, in welchen die Daten geschrieben werden sollen, an den Grenzen des Cachespeichers ausgerichtet sind. Ist dem der Fall, so werden die Daten anweisungsgemäss geschrieben. Andernfalls wird das Paket (Header und Daten) in eine spezielle Warteschlange geschrieben und die Prozessoren durch den weiter oben beschriebenen, eigengenerierten Interruptprozess von diesem Umstand in Kenntnis gesetzt. Die Prozessoren können dann die Daten von der speziellen Warteschlange in den Cachespeicher **22** verschieben und den Cachespeicherinhalt später in den Speicher **22** schreiben, damit gewährleistet ist, dass keine guten Daten überschrieben werden oder anderweitig verloren gehen und dass die Kohärenz zwischen dem Cachespeicher **22** und dem Speicher **28** gewahrt bleibt.

BLOCKÜBERTRAGUNGSMASCHINE (BÜM):

[0254] Da es dem Prozessor **20** nicht möglich ist, an ausserhalb der CPU **12A** gelegene Elemente Informationen auf direktem Weg zu übermitteln (d. h. zu senden), steht die BÜM **88** der Schnittstelle **24a** (**Fig.** 5) für ein indirektes Verfahren zur Informationsübermittlung zur Verfügung.

[0255] Die BÜM **88** ist ein Mechanismus, der verwendet wird, um den gesamten prozessorinitiierten E/A-Datenverkehr zum Transport von Informationsblöcken zu implementieren. Die BÜM **88** ermöglicht die Erstellung von Lese- und Schreibpaketen mit einer Länge, die der maximalen Länge entspricht, welche gemäss der TNet-Paketdefinition erlaubt ist, also derzeit **64** Bytes. Die BÜM **88** stellt zwei "virtuelle" Kanäle bereit, von denen einem eine höhere Priorität eingeräumt wird als dem anderen. Wie in **Fig.** 15, auf die hier Bezug genommen wird, veranschaulicht, beinhaltet die BÜM **88** zwei BÜM-Register **300**, **302**, deren Inhalt an den MUX **306** (der Schnittstelleneinheit **24a**; **Fig.** 5) gekoppelt ist und dazu verwendet wird, um über die (in **Fig.** 15 nicht dargestellt) Speichersteuerungen **26** (in **Fig.** 15 nicht dargestellt) auf den Systemspeicher **28** zuzugreifen. Ein Abschnitt der Register **300**, **302** enthält Zeiger (d. h. BÜM-Adresse **0** und BÜM-Adresse **1**) auf den Anfang der BÜM-Datenstruktur **304** in dem Speicher **28** der CPU **12A** (**Fig.** 2). Jedes Mal wenn Informationen an das eine oder andere ausserhalb der CPU **12A** gelegene Element (z. B. die CPU **12B** oder irgendeines der E/A-Geräte **17**, **18** der E/A-Paket-Schnittstelle **16**) zu senden ist bzw. von einem solchen zu empfangen ist, schreiben die Prozessoren eine Datenstruktur **304** in den Speicher **28**. Jede Datenstruktur muss dabei an einer Vierfachwortgrenze beginnen und die BÜM-Register **300**, **302** sind nur von den Prozessoren **20** beschreibbar. Beschreibt ein Prozessor tatsächlich einen der BÜM-Register **300**, **302** beschreibt, so tut er dies mit einem Wort, welches

das Anforderungsbit (rc0, rc1) in einen klaren Zustand versetzt, der dazu dient, den BÜM-Prozess zu initiieren, welcher von der BÜM-Ablaufsteuereinheit gesteuert wird.

[0256] Die BÜM-Register **300**, **302** beinhalten auch Fehlerbits (e0, e1), die zur Übertragung einer Zeitablauffehlermeldung oder sonstigen negativen Rückmeldung dienen. Die Fehlerbits werden gelöscht, wenn das entsprechende BÜM-Register beschrieben wird. Durch das Fehlerursachenbit (ec) wird zwischen Zeitablauffehlern und sonstigen negativen Rückmeldungen unterschieden.

[0257] Wenn Informationen von den Prozessoren **20** an eine externe Einheit übertragen werden, werden die zu übertragenden Informationen in dem Datenpufferabschnitt **304b** gehalten. Wenn von einer externen Einheit kommende Informationen von den Prozessoren **20** empfangen werden, ist der Datenpufferabschnitt **304b** jener Ort, in welchem die Leseantwortinformation gehalten werden soll.

[0258] Der Anfang der Datenstruktur **304**, der von dem Prozessor **20** geschriebene Abschnitt **304**, beinhaltet ein Informationsfeld (Dest), in welchem das externe Element angegeben wird, welches das Paket empfängt, welches gesendet werden wird. Der Abschnitt **304a** beinhaltet auch ein Informationsfeld (TYPE), welches die gewünschte Operation beschreibt (z. B. Information über Lese- oder Schreiboperation), ein Längeninformationsfeld (Len), das die Anzahl der Datenbytes beschreibt, welche geschrieben oder angefordert werden, und ein Adressinformationsfeld (Address), welches angibt, wo an dem externen Element (Dest) sich die gewünschten Daten befinden bzw. wohin die übertragenen Daten zu schreiben sind. Diese Informationen werden von der Paketüberträgereinheit **94** (**Fig.** 5) verwendet, um das Paket in der in den **Fig.** 3–4 gezeigten Form zusammenzustellen.

[0259] In dem Datenstrukturabschnitt **304a** befindet sich unmittelbar auf die Adressinformationen folgend ein Wort (Local Buffer Ptr) welches die Adresse jenes Ortes im Speicher enthält, an dem sich der Datenpufferabschnitt **304b** befindet. Dieses wird seinerseits unmittelbar von einem Wort gefolgt, welches einen Verkettungszeiger, ein Listenende-Flag (e1) zur Anzeige, dass eine Anforderung vollständig ist, ein Zeitablauf-Interruptbit bzw. ein Negativrückmeldungs-Maskenbit (it), einen Fertigstellungsinterruptanzeiger (ic) sowie eine Prüfsummenanforderung (cs) umfasst.

[0260] Für jeden im Speicher befindlichen und zu einem externen Element (z. B. zu einer E/A-Speichereinrichtung) zu verschiebenden Datenabschnitt mit einer Länge von bis zu maximal 64 Byte wird jeweils eine Datenstruktur **304** verwendet. Die BÜM wird betrieben, um jeweils in Reaktion auf eine Anforderungsstruktur sequentiell auf den Speicher **28** zuzugreifen und aus diesem Daten in Form von 64-Byte-Segmenten zu entnehmen, wobei für jedes Segment ein Nachrichtenpaket erstellt wird und dieses Nachrichtenpaket abgeschickt wird. Der Verkettungszeiger führt die BÜM zu der nächsten Datenstruktur für die nächsten 64 Bytes an Daten, sofern nicht das Listenende-Bit (e1) gesetzt ist, welches bewirkt, dass die Operation beendet wird.

[0261] Sind Daten zu einer Anzahl verschiedener externer Elemente zu senden, so muss für jedes einzelne dieser verschiedenen Elemente eine eigene Datenstruktur (bzw. eigene Datenstrukturen, falls mehr als 64 Bytes zu übertragen sind) eingerichtet werden. Diese einzelnen Datenstrukturen können dann unter Verwendung der in den Verkettungszeigerfeldern der Anforderungsstrukturen enthaltenen Verkettungszeiger miteinander verkettet werden.

[0262] Das Verkettungszeigerfeld wird verwendet, um als Inhalt des BÜM-Registers für nachfolgende Datenstrukturen zu fungieren. Wenn zum Beispiel ein in dem Speicher **28** vorhandener, grosser Datenblock an N verschiedene, externe Geräte zu senden ist, so wird für ein jedes der N externen Elemente eine Datenstruktur in den Speicher geschrieben, wobei jede Datenstruktur angibt, an welchem Ort im Speicher **28** die BÜM-Logik **88** die zu übermittelnden Daten finden kann. Es wird sodann von der BÜM-Logik **88** auf die Daten für jedes Element zugegriffen, es werden Nachrichtenpakete erstellt, die diese Daten enthalten, und es werden diese Pakete an den Paketüberträger **120** übermittelt, von wo aus sie in angemessener Weise über das TNet übertragen werden. Daraufhin wird, sofern eine Verkettung mit einer anderen Datenstruktur erforderlich ist, auf den in der Datenstruktur enthaltenen Verkettungszeiger zugegriffen und dieser in das entsprechende BÜM-Register **300**, **302** geschrieben, welches die Aktion initiierte, wodurch die Adresse für die nächste Struktur für jenes Element bereitgestellt wird, das als nächstes ein Anforderungspaket empfangen soll.

[0263] Ein gesetztes Listenende-Bit (e1) zeigt an, dass das Ende der Verkettung erreicht ist, woraufhin die BÜM-Verarbeitung angehalten wird.

[0264] Das Fertigstellungsinterrupt-Bit (ic) bewirkt, sofern es gesetzt ist, dass die Schnittstelleneinheit **24a** ein Interrupt (BTECmp) ausgibt, welches ein Bit in dem Interrupt-Register **280** (**Fig.** 14A) setzt, um die Fertigstellung des vorangegangenen, von der BÜM übertragenen Pakets anzeigt (nicht zu verwechseln mit jenem, auf welches der Verkettungszeiger zeigt).

[0265] Das Zeitablauf-Interruptbit (it) bewirkt, sofern es gesetzt ist, dass die Schnittstelleneinheit **24a** ein Interruptsignal an den Prozessor **20** ausgibt, wenn die für die Bestätigung des Zugriffs vorgesehene Zeit überschritten wird (d. h. wenn der Anforderungs-Zählerzeitgeber (nicht dargestellt) ein Zeitablaufsignal ausgibt, das anzeigt, dass eine erwartete Antwort nicht innerhalb der dafür zugemessenen Zeit empfangen wurde), oder wenn als Antwort eine negative Rückmeldung eingeht (was darauf hindeutet, dass das Zielelement der Anforderung diese nicht verarbeiten konnte).

[0266] Wenn schliesslich das Prüfsummenbit (cs) gesetzt ist, so werden die in das externe Element zu schreibenden Daten durch den in der BÜM **88** (Schnittstelle **24a**; **Fig.** 5) enthaltenen Prüfsummengenerator (nicht dargestellt) geleitet, wo eine Prüfsummenmenge zu erstellt wird. Die generierte Prüfsumme kann in den Speicher geschrieben werden und anschliessend in ein eigenes Paket verpackt werden und an den Bestimmungsort des Nachrichtenpakets (bzw. der Nachrichtenpakete) gesendet werden, worin die Daten enthalten sind, aus denen die Prüfsumme gebildet worden ist.

[0267] Somit kann zusammenfassend gesagt werden, dass wenn die Prozessoren **20** der CPU **12A** Daten an eine externe Einheit senden wollen, sie eine Datenstruktur **304** in den Speicher **28** schreiben, welche in Abschnitt **304a** der Datenstruktur bezeichnende Informationen und im Pufferabschnitt **304b** die eigentlichen Daten umfassen. Die Prozessoren bestimmen daraufhin die Priorität der Daten und beschreiben das BÜM-Register **300**, **302** mit der Adresse in dem Speicher **28**, an welcher die Datenstruktur **304** (d. h. der Headerabschnitt **304a**) aufzufinden sind, wobei gleichzeitig das Anforderungsfertigstellungsbit (rcl) des BÜM-Registers **300**, **302** gelöscht wird, und die BÜM-Operation veranlasst wird, gesteuert von der BÜM-Ablaufsteuereinheit **306** ihre Funktion aufzunehmen. Auf die in dem Abschnitt **304a** enthaltenen Felder 'Dest', 'TYPE', 'Len' und 'Address" wird von dem Speicher **28** aus zugegriffen und die darin befindlichen Informationen werden an den Paketüberträger **120** übermittelt, wo sie ordnungsgemäss in Paketform gebracht werden. Ist in der Datenstruktur **304** spezifiziert, dass es sich bei der Übertragung um eine Schreiboperation handeln soll, so wird auf den lokalen Pufferzeiger zugegriffen, welcher verwendet wird, um den Datenpufferabschnitt **304b** ausfindig zu machen. Anschliessend erfolgt der Zugriff auf die Daten, deren Übermittlung an den Paketüberträger **120**, deren Paketierung gemeinsam mit den Header- und Adressinformationen, und deren Versand.

[0268] Wenn die Datenstruktur **304** eine Leseanforderung bezeichnet (d. h. wenn die Prozessoren **20** auf Daten von einer externen Einheitßentweder einem E/A-Gerät oder einer CPU **12**-zugreifen wollen), wird unter Verwendung der Felder 'Len' und 'Local Buffer Ptr' eine Schreibanforderung für den Speicher **28** generiert, wenn das Leseantwortpaket (von dem externen Element, an das die Anforderung adressiert wurde) retourniert wird. Die Daten werden in dem Eingangspaket-Zwischenspeicher **110** gestellt, bis die lokale Speicherschreiboperation durchgeführt ist.

[0269] Antworten auf eine prozessorgenerierte Leseanforderung an eine externe Einheit werden von der ZVÜ-Tabellenlogik **146** nicht verarbeitet. In diesem Fall wird vielmehr, wenn die Prozessoren **20** die BÜM-Datenstruktur einrichten, der Anforderung eine Transaktionsfolgenummer (TFN) zugewiesen und in das Headerfeld des Nachrichtenpakets integriert, das von der BÜM **88** zusammengestellt und gesendet wird und bei dem es sich um ein bereits weiter oben besprochenes Paket vom Typ HAC (**Fig.** 4) handelt. Die Prozessoren **20** nehmen auch eine Speicheradresse in die BÜM-Datenstruktur auf, an welcher die Daten, nachdem sie empfangen worden sind, abzulegen sind. Wenn die BÜM-Logik **88** das Paket abschickt, wird die Speicheradresse des Pufferspeicherorts in eine (nicht dargestellte) in der Anforderungstransaktionslogik **100** enthaltene Registerdatei geschrieben (**Fig.** 5), wobei die TFN als Zeiger in die Registerdatei verwendet wird.

[0270] Wenn die Antwort (welche in Form eines HDC-Nachrichtenpakets erfolgen wird-**Fig.** 4B) von der CPU **12** empfangen wird, verwendet die Anforderungstransaktionslogik **100** die Transaktionsfolgenummer (TFN) aus dem Header des Pakets als Zeiger in die (nicht dargestellte) Registerdatei, um die entsprechende Pufferspeicheradresse ausfindig zu machen, an welcher die in dem eingehenden Nachrichtenpaket enthaltenen Daten innerhalb des Speichers **28** abzulegen sind.

[0271] Im Hinblick auf ein besseres Verständnis der Priorisierung der BÜM-Register **300**, **302** sei hier angenommen, dass es sich bei der vorangegangenen Datenübertragung von der CPU **12A** zu einer externen Einheit um die Übertragung eines grossen Datenblocks handelt. Demgemäss würde von den Prozessoren **20** in dem Speicher **28** eine Reihe von Datenstrukturen eingerichtet werden, von denen eine jede (mit Ausnahme der letzten) jeweils einen Verkettungszeiger zu einer weiteren Datenstruktur enthält, und wobei die Gesamtsumme dieser Datenstrukturen (in den Datenpufferabschnitten **304b** der Datenstrukturen **304**) die zu sendenden Daten umfasst. Es sei nun weiter angenommen, dass von den Prozessoren **20** eine Anforderung mit höherer Priorität durchgeführt werden möchte. IN solch einem Fall würde die einer solchen Anforderung mit höherer Priorität zugeordnete Datenstruktur **304** in dergleichen Form wie weiter oben beschrieben in den Speicher **28** geschrieben werden. Dann würde das mit höherer Priorität zu behandelnde BÜM-Register **300** mit der BÜM-Adresse beschrieben werden, die zum Wiederauffinden der Datenstruktur nötig ist, woraufhin das Anforderungsfertigstellungsanzeigebit (rc0) gelöscht würde.

[0272] Die durch das Beschreiben des BÜM-Registers **300** angezeigte BÜM-Anforderung wird jedoch nicht sofort gestartet. Sie wartet bis die von dem Inhalt des BÜM-Registers **302** initiierte Operation zwischen Paketen anhält. Weitere von dem Inhalt des BÜM-Registers **302** angekündigte BÜM-Operationen werden vorübergehend zugunsten der von dem Inhalt des BÜM-Registers **300** angezeigten BÜM-Operation in Warteposition gehalten. Diese BÜM-Operation wird bis zu ihrer Fertigstellung fortgesetzt, woraufhin die von dem Inhalt des BÜM-Registers **302** gemeldete BÜM-Operation wiederaufgenommen wird und bis zu ihrer Fertigstellung abgearbeitet wird, sofern nicht das BÜM-Register **300** erneut mit einem anderen BÜM-Operationsbeschreiber beschrieben wird.

Speichersteuerung:

[0273] Im folgenden wird vorübergehend wieder auf **Fig.** 2 Bezug genommen, in welcher die Schnittstelleneinheiten **24a**, **24b** über ein Paar von Speichersteuerungen (SpSt) **26a**, **26b** auf den Speicher **28** zugreifen. Die Speichersteuerungen schaffen eine schnellfehlende Schnittstelle zwischen den Schnittstelleneinheiten **24** und dem Speicher **28**. Die Speichersteuerungen **26** stellen die für den Zugriff auf die Speichermatrix **28** nötige Steuerlogik bereit (die in der Form eines dynamischen Schreib-Lese-Speichers (DRAM) implementiert ist). Die Speichersteuerungen empfangen Speicheranforderungen von den Schnittstelleneinheiten **24** und führen Lese- und Schreiboperationen aus und stellen Aktualisierungssignale für die DRAMs, die die Speichermatrix an **28** implementieren. Die beiden Speichersteuerungen **26a**, **26b** werden parallel zueinander betrieben, um einen 72 Bit umfassenden Datenpfad zwischen der Speichermatrix **28** und den Schnittstelleneinheiten **24a**, **24b** bereitzustellen, die eine FKC-Konfiguration nach dem Schema SBC-DBD-SbD verwenden, wobei b = 4, bei einer Gesamtmenge von 100 Bits (**64** Datenbits plus 28 Adressbits plus 8 Prüfbits), von denen nur **72** Bits (**64** Datenbits und **8** Prüfbits) tatsächlich in den Speicher **28** geschrieben werden.

[0274] Im folgenden wird vorübergehend auf **Fig.** 16 Bezug genommen, in welcher die beiden Speichersteuerungen **26a**, **26b** gezeigt werden, die parallel betrieben werden, um **144** Bits an Daten von dem Speicher **28** abzurufen. Eine SpSt (z. B. **26a**) ist entsprechend angeschlossen, um auf zwei 32-Bit-Wörter an aufeinanderfolgenden, geraden Adressen, sowie auf 8 Prüfbits gleichzeitig zuzugreifen, um so einen 72-Bit-Pfad **330a** zwischen der SpSt und dem Speicher **28** zu bilden. Die andere SpSt (d. h. **26b**) ist entsprechend angeschlossen, um in ähnlicher Weise auf zwei ungerade 32-Bit-Wörter, sowie auf weitere 8 Prüfbits zuzugreifen, um so einen zweiten 72-Bit-Pfad **330b** zu bilden. Diese Anordnung ermöglicht es den beiden Speichersteuerungen **26a**, **26b**, zusammenzuarbeiten und gleichzeitig den Schnittstelleneinheiten **24** mit minimaler Latenzzeit ein 64-Bit-Wort zur Verfügung zu stellen, von dem eine Hälfte (D0) von der SpSt **26a** und die andere Hälfte (D1) von der SpSt **26b** stammt. Die Schnittstelleneinheiten **24** generieren und prüfen die FKC-Prüfbits. Die verwendete FKC-Konfiguration erkennt und korrigiert nicht nur (Einzelbit-)Datenfehler, sondern erkennt auch sämtliche Doppelbitfehler, sowie jeden Fehler bis hinauf zu Vierbitfehlern aus jedem einzelnen DRAM. Die schnellfehlende Ausführung prüft die Parität bei Adressübertragungen zwischen der Schnittstelle **24** und der SpSt **26**, dem Bus **25**, sowie auch in internen Registern.

[0275] Vom Standpunkt der Schnittstelleneinheiten **24** aus gesehen, wird mit zwei Befehlen auf den Speicher **28** zugegriffen: einem "Lese N-Doppelwort" und einem "Schreibe N-Doppelwort". Beide dieser Befehle ergehen an die Speichersteuerungen **26**, mit Adresse und Steuerinformation in der ersten 36-Bit-Übertragung und einer Byte-Zählung in der zweiten 32-Bit-Übertragung. Bei Schreiboperationen gliedern die Speichersteuerungen **26** den Befehl in entweder eine Doppelwort-Schreiboperation oder in einen Block von Doppelwort-Schreiboperationen auf. Im Fall von Leseoperationen werden die angeforderten Daten entweder in einer. einzelnen Doppelwort-Leseoperation oder in einem Blockleseformat retourniert. Das "data valid"-Signal teilt den Schnittstelleneinheiten **24** zwei Zyklen im voraus mit, dass Lesedaten retourniert bzw. nicht retourniert werden.

[0276] Wie weiter oben gezeigt, hat der Instandhaltungsprozessor (IP **18**; **Fig.** 1A) zwei Möglichkeiten, auf die CPUs **12** zuzugreifen. Die eine besteht darin, die TNet-Struktur, einschliesslich der Router **14**, zu verwenden, um paketierte Informationen zu senden (bzw. zu empfangen). Eine andere, wenn auch beschränktere Möglichkeit besteht über einen Online-Zugangspunkt (OLZP), der in die verschiedenen Elemente des Systems **10** (z. B. die Router **14**, die CPUs **12**, die E/A-Paket-Schnittstellen **16**) integriert ist. Diese letztere Zugriffsform wird in **Fig.** 17 veranschaulicht, worin ein serieller OLZP-Port **285** gezeigt wird, welcher für den IP **18** über jede der Speichersteuerungen **26** sowohl Lese- als auch Schreibzugriff bereitstellt. (In **Fig.** 17 wird der OLZP-Zugriff auf die Speichersteuerung **26a** dargestellt; die Speichersteuerung **26b** ist im wesentlichen von identischer Bauart.) Zum Zeitpunkt des Ladens beschreibt der IP **18** ein in dem OLZP enthaltenes Register mit Befehlen, die es den Prozessoren **20** erlauben, ein Abbild einer Befehlsfolge in dem Speicher zu erstellen, das es ihnen (den Prozessoren **20**) ermöglicht, ihren Betrieb aufzunehmen, indem sie zum Beispiel zu E/A-Geräten gehen, um Befehle und Daten von einem externen (Speicher-)Gerät zu übertragen, mit denen der Ladevorgang abgeschlossen wird.

[0277] Der OLZP **285** wird von den Prozessoren **20** auch dazu verwendet, um Fehlermeldungen an den IP **18** zu übertragen. Wenn zum Beispiel eine der Schnittstellen **24** in Daten, die sie von der Speichersteuerung **26** empfängt, einen Paritätsfehler entdeckt, gibt sie ein Fehlersignal aus, das den Betrieb anhält und gleichzeitig bewirkt, dass in dem OLZP **285** eine Bitposition gesetzt wird, durch die der IP **18** von dem Fehler benachrichtigt wird. Die von der Speichersteuerung **26** durchgeführte Fehlerprüfung (z. B. eine negativ verlaufende Paritätsprüfung bei Register-Leseoperationen) führt in ähnlicher Weise zu einer Betriebsanhaltung und zu einer über den OLZP **285** erfolgenden Benachrichtigung des IP **18**, dass ein Fehler aufgetreten ist.

[0278] Der IP **18** und die verschiedenen in dem System vorhandenen OLZPs (z. B. der in der SpSt **26a** integrierte OLZP **285**) kommunizieren untereinander über einen seriellen Bus **287**, welcher gemäss der IEEE-Spezifikation 1149.1 aufgebaut ist.

[0279] Die Architektur der Speichersteuerungen folgt im allgemeinen einem herkömmlichen Entwurf, abgese-

hen von einer speziellen Form der Fehlerprüfung, welche durch die Überwachung der verschiedenen Ablaufsteuerungen erfolgt, die zur Implementierung der Speichersteuerungen **26** herangezogen werden. Wie in **Fig.** 18 veranschaulicht, werden die Funktionsweisen der SpSt **26a** (dasselbe gilt im übrigen auch für die SpSt **26b**) von drei zentralen Ablaufsteuereinheiten gesteuert, von denen eine jede dupliziert und verglichen wird. Ein übergeordnetes Ablaufsteuereinheitenpaar **390** wird betrieben, um die Funktionen der SpSt **26a** selbst zu steuern, wie etwa das Verschieben von Daten und Befehlen von dem SpSt-A/D-Bus **25** auf den DRAM-Datenbus, um Daten an den Speicher **28** zu übermitteln. Das übergeordnete Ablaufsteuereinheitenpaar **390** übt seinerseits eine Steuerfunktion über die Speichersteuerungs-Adress/Daten-Ablaufsteuerungseinheit (SpSt-A/D-Ablaufsteuerungseinheit) **392** aus, welche die Daten- und Adressübertragungen auf dem Bus **25** zwischen der SpSt **26a** und der entsprechenden Schnittstelleneinheit **24a** abwickelt. Die Adressierungen und die Datenübertragungen auf dem DRAM-Datenbus, sowie die Generierung und Folgesteuerung der erforderlichen Auffrischsignale werden von der DRAM-Ablaufsteuereinheitenpaar **394** gesteuert. Die von den Ablaufsteuereinheitenpaaren **390**, **392** und **394** eingegebenen, digitalen Zustände werden mithilfe von Vergleichsschaltungen **395** miteinander verglichen. Jedes Nichtübereinstimmen von Daten führt zur Ausgabe eines Fehlersignals seitens der Vergleichsschaltung **395**, welche die Nichtübereinstimmung erkannt hat, wodurch der Betrieb der CPU **12** angehalten wird.

Paket-Wegeleitverfahren:

[0280] Die Nachrichtenpakete, welche zwischen den verschiedenen Elementen des Verarbeitungssystems **10** (z. B. den CPUs **12A**, **12B**, und Geräten, die an die E/A-Paket-Schnittstellen **16** angeschlossen sind) übermittelt werden, werden von den Routern **14** gemäss den in dem Paket enthaltenen Informationen (d. h. dem Bestimmungsort-Feld des Headers, **Fig.** 3B, obwohl auch andere Informationen, wie etwa das Ursprungsort-Feld, dazu verwendet werden könnten) einer "Wegeleitung" unterworfen. Vor einer Abhandlung des Entwurfs und der Bauart der Router **14** ist es jedoch von Vorteil, sich als erstes ein Verständnis des Protokolls zu verschaffen, das zum Nachrichtenaustausch auf den TNet-Links **L** zwischen den CPUs **12** und den Routern **14** bzw. zwischen den Routern **14** und den E/A-Paket-Schnittstellen 16 verwendet wird.

[0281] Als erstes wird jedes TNet-Link L über einen Port, welcher sowohl mit Empfangs- als auch mit Sendefähigkeit ausgestattet ist, an ein Element (z. B. einen Router **14A**) des Verarbeitungssystems **10** angeschlossen. Jeder Sendeport eines jeden Elements muss ein Sendetaktsignal (S_Tkt-Signal) bereitstellen, das für die synchrone, symbolweise Übertragung von Nachrichtenpaketen verwendet wird. Bei jedem einzelnen Taktzyklus (d. h. bei jedem Taktintervall) des Sendetaktes wird ein Symbol übertragen, so dass das Taktsynchronisierungs-FIFO am empfangenden Ende der Übertragung die Synchronisation beibehält.

[0282] Die Taktsynchronisierung ist von dem Modus abhängig, in welchem das Verarbeitungssystem **10** betrieben wird. Bei einem Simplexbetrieb, bei welchem die CPUs **12A** und **12B** zum Beispiel getrennt voneinander betrieben werden, ist die Taktgabe zwischen den Routern **14** und den CPUs **12"** frequenznah"; das bedeutet, dass die Takte, welche von den CPUs **12** und den Routern **14** verwendet werden, die direkt an die CPUs angeschlossen sind, auseinanderdriften können. Wenn das Verarbeitungssystem **10** umgekehrt im Duplexmodus betrieben wird (und z. B. die CPUs sich in synchron im Gleichschritt laufendem Betrieb befinden), sind die Takte zwischen den Routern **14** und den CPUs **12**, an die sie angeschlossen sind, frequenzsynchronisiert (jedoch nicht unbedingt phasensynchronisiert).

[0283] Der Fluss der Datenpakete zwischen den verschiedenen Elementen des Verarbeitungssystems **10** wird durch Befehlssymbole gesteuert, die zu jeder Zeit, auch innerhalb eines Pakets, in Erscheinung treten können. Wie bereits weiter oben (in Bezug auf die Tabelle 1) erwähnt, ist ein Befehlssymbol daran erkennbar, dass die bedeutendsten Bits alle auf 0 gesetzt sind. Im einzelnen handelt es sich dabei um folgende Befehlssymbole.

[0284] LEERLAUF (IDLE): Das LEERLAUF-Befehlssymbol wird mit jedem Takt übertragen, wenn es kein anderes Befehlssymbol oder Datenpaket zu senden gibt. Das LEERLAUF-Befehlssymbol hat die Funktion eine Füllelements zwischen Paketen bzw. Befehlssymbolen auf den TNet-Links.

[0285] BELEGT (BUSY): Das BELEGT-Befehlssymbol wird gesendet, wenn das empfangende Gerät erkennt, dass es bald nicht mehr in der Lage sein wird, Datensymbole anzunehmen.

[0286] FÜLLEN (FILL): Das FÜLLEN-Befehlssymbol wird von dem sendenden Element in ein Nachrichtenpaket eingegeben, wenn das sendende Element weiss, dass das empfangende Element, an das es gerade Symbole sendet, belegt ist (z. B. durch den Empfang eines BELEGT-Befehlssymbols).

[0287] STOPP (HALT): Dieses Befehlssymbol wird von einer CPU **12** oder einem IP 18 ausgelöst und von den Routern **14** an alle Elemente des Verarbeitungssystems **10** übermittelt, um diesen ein Ereignis mitzuteilen, das ein Aktivwerden der Software aller CPUs **12** und gewisser E/A-Geräte erfordert. Das STOPP-Befehlssymbol stellt einen Mechanismus bereit, der es erlaubt, rasch alle CPUs **12** in einem System **10** zu informieren, dass es nötig ist, die Einleitung jeglicher E/A-Aktivitäten zu stoppen.

[0288] ANDERES LINR SCHLECHT (OTHER LINR BAD-OLB): Wenn ein Router **14**, welcher an die CPU an-

geschlossen ist und im Duplexbetrieb arbeitet, einen Fehler in dem Befehlssymbol oder Paket entdeckt, das gerade von einer der CPUs **12** kommend empfangen wird, jedoch in dem Befehlssymbol bzw. Paket, das gerade von der anderen der CPUs **12** kommend empfangen wird, keine Fehler entdeckt, so sendet der Router **14** ein OLB-Befehlssymbol an jene CPU **12**, welche das gute Paket bzw. Befehlssymbol übermittelt hat. Dieses Befehlssymbol wird auch als Reaktion auf CRC-Fehler, Befehlssymbolfehler und Protokollverletzungsfehler gesendet, allerdings nur im Duplexbetrieb. OLB- und (weiter unten beschriebene) TLB-Befehlssymbole werden gleichzeitig an im Duplexbetrieb arbeitende CPUs gesendet; das heisst ein TLB-Befehlssymbol wird an jene CPU **12** gesendet, von welcher das fehlerhafte Paket bzw. Symbol empfangen wurde, oder der Fehler bemerkt wurde, und im wesentlichen zeitgleich dazu wird ein OLB-Symbol zu der anderen CPU **12** des im Duplexbetrieb arbeitenden CPU-Paars gesendet.

[0289] BEREIT (READY): Dieses Befehlssymbol wird gesendet, wenn das zuvor belegt gewesene Element nun wieder in der Lage ist, zusätzliche Daten anzunehmen.

[0290] ÜBERSPRINGEN (SKIP): Dieses Befehlssymbol zeigt einen Taktzyklus, der wahlweise übersprungen werden kann. Dieses Befehlssymbol wird in Verbindung mit dem frequenznahen Betrieb als Hilfe zur Aufrechterhaltung der Synchronisation zwischen den beiden Taktsignalen verwendet, die (1) jedes Symbol zu jedem empfangenden Taktsynchronisierungs-FIFO übertragen und in dieses laden, und (2) Symbole von dem FIFO abrufen.

[0291] SCHLAF (SLEEP): Dieses Befehlssymbol wird von jedem beliebigen Element des Verarbeitungssystems **10** gesendet, um anzuzeigen, dass kein weiteres Paket (nach dem momentan übermittelten, falls ein solches vorhanden ist) mehr über den speziellen Link **L** gesendet werden darf, bis ein BEREIT-Befehlssymbol (weiter unten beschrieben) empfangen wird.

[0292] SOFT RESET (SRST): Das SRST-Befehlssymbol wird als Auslöser während jener Prozesse ("Synchronisierung" und "Reintegration", weiter unten beschrieben) verwendet, die dazu benutzt werden, um die Symbolübertragungen zwischen den CPUs **12** und den Routern **14A**, **14B** zu synchronisieren und um daraufhin die CPUs **12** für den Duplexbetrieb jeweils in einen identischen Zustand zu versetzen.

[0293] SYNC Das SYNC-Befehlssymbol wird von einem Router **14** zu der CPU **12** des Verarbeitungssystems **10** (d. h. der Subprozessorsysteme **10A**/**10B**) gesendet, um vor dem Eintritt in den Duplexbetrieb eine Frequenzsynchronisierung zwischen der CPU **12** und den Routern **14A**, **14B** zu erreichen, bzw. wenn bereits im Duplexbetrieb, eine Synchronisierung anzufordern, wie weiter unten in Detail beschrieben. Das SYNC-Befehlssymbol wird zusammen mit dem SRST-Befehlssymbol dazu verwendet, um unter anderem zwischen Systembetriebsarten (d. h. von Simplexbetrieb auf Duplexbetrieb bzw. von Duplexbetrieb auf Simplexbetrieb) hin und herzuwechseln, wie weiter unten in dem Abschnitt über Synchronisierung und Reintegration beschrieben wird.

[0294] LINK SCHLECHT (THIS LINK BAD-TLB): Wenn irgendein Systemelement, das ein Symbol von einem TNet-Link **L** empfängt (z. B. ein Router, eine CPU, oder ein E/A-Gerät), beim Empfangen eines Befehlssymbols oder eines Pakets einen Fehler bemerkt, sendet es ein TLB-Befehlssymbol an das Systemelement zurück, welches das fehlerhafte Paket oder Symbol übermittelt hat. Es wird üblicherweise als Reaktion auf CRC-Fehler, Befehlssymbolfehler und Protokollverletzungsfehler gesendet.

[0295] ICH BIN DIR ÜBERGEORDNET (I OWN YOU-IOY): IOY-Befehlssymbole werden (ausschliesslich im Duplexbetrieb) nur von einer CPU **12** an einen Router **14** gesendet, um den Router **14** dazu zu zwingen, Daten von der sendenden CPU auszuwählen, wodurch der sendenden CPU **12** praktisch ein Besitzrecht zugesprochen wird; sämtliche anderen Datenübertragungen von der nicht-sendenden CPU werden nicht beachtet. Die tatsächliche Bitstruktur des IOY-Befehlssymbols ist dieselbe wie jene, die für das Anderes Link Schlecht-Befehlssymbol (OLB) verwendet wird-wobei das Ursprungselement entscheidet, um welches Symbol es sich konkret handelt. Wird das IOY/OLB-Symbol von einer CPU **12** gesendet, so wird es als IOY-Symbol interpretiert; wird das IOY/OLB-Symbol von einem Router gesendet, so wird es als OLB-Symbol interpretiert. Um es anders auszudrücken, in der Kommunikation zwischen den CPUs **12** und den Routern **14A**, **14b** werden IOY-Befehlssymbole ausschliesslich von CPUs und OLB-Befehlssymbole ausschliesslich von Routern gesendet.

[0296] ABWEICHUNG (DIVERGE-DVRG): Das DVRG-Befehlssymbol wird im Duplexbetrieb von einem Router gesendet, um die im Duplexbetrieb laufenden CPUs zu informieren, dass eine Divergenz in den Datenströmen, der gerade von den CPUs empfangen wird, erkannt wurde; der Router empfängt nämlich von der CPU **12** identische Symbolpaare, die miteinander verglichen werden, wenn sie aus den Synchronisierungs-FIFOs entnommen werden. Das DVRG-Befehlssymbol signalisiert der CPU **12**, dass ein Nichtübereinstimmen der Daten festgestellt worden ist. Wenn es von den CPUs empfangen wird, wird ein Divergenzauffindungsprozess eingeleitet, wodurch durch die CPUs eine Bestimmung erfolgt, welche CPU ausgefallen ist oder fehlerhaft arbeitet und der weitere Betrieb dieser CPU eingestellt wird.

[0297] PAKET GUT (THIS PACKET GOOD-TPG): Ein Befehlssymbol, welches auf ein Nachrichtenpaket folgt und angibt, dass der Sender des Pakets bestimmt hat, dass der CRC des Pakets in Ordnung ist. Weitere Details dazu sind dem Abschnitt "Paketzustand" weiter unten zu entnehmen.

[0298] PAKET SCHLECHT (THIS PACKET BAD-TPB): Das TPB-Befehlssymbol ersetzt das TPG-Befehls-

symbol, wenn das empfangende Element festgestellt hat, dass der CRC des empfangenen Nachrichtenpakets unkorrekt ist.

Flusssteuerung:

[0299] Die Router **14** verfügen über ein begrenztes Speichervermögen und verwenden daher beim Weiterleiten von Nachrichtenpaketen keinerlei Art von Store-and-Forward-Strategie; sie arbeiten vielmehr nach der sogenannten Wormhole-Strategie: das vordere Ende eines Nachrichtenpakets hat den Router bereits durchlaufen und wieder verlassen, noch bevor das hintere Ende empfangen worden ist. Dies ist einer der Gründe dafür, dass der Nachrichtenstrom zwischen den verschiedenen Elementen des Verarbeitungssystems **10** (z. B. den CPUs **12**, den Routern **14** und dergleichen) durch die oben beschriebenen Befehlssymbole gesteuert wird, wobei in erster Linie die dort beschriebenen Befehlssymbole BELEGT/FÜLLEN/BEREIT (BUSY/FILL/READY) verwendet werden. Diese Flusssteuerung wird als "Gegendruck"-Verfahren bezeichnet.
[0300] Wann immer ein bestimmtes Systemelement erkennt, dass seine Empfangswarteschlange (d. h. der elastische Puffer **506**-**Fig.** 19A) nahezu voll ist, bedient es sich der bidirektionalen Fähigkeit des TNet-Links L, auf dem es das gerade eingehende Nachrichtenpaket empfängt, und überträgt an das sendende Element am anderen Ende des TNet-Links **L** über den zugeordneten Sendeport ein BELEGT-Befehlssymbol, mit welchem das sendende Element aufgefordert wird, vorerst mit einer weiteren Sendetätigkeit zuzuwarten. Die Verwendung des BELEGT-Befehlssymbols wird als das Ausgeben von "Gegendruck" bezeichnet. Die CPUs **12** oder die E/A-Paket-Schnittstellen **16** können auf diese Weise Gegendruck ausgeben, während sie darauf warten, dass interne Ressourcen verfügbar werden, wenn und nur wenn solche internen Ressourcen verfügbar werden, unabhängig von Gegendruck, der an irgendeinem bestimmten TNet-Port ausgegeben wird, ein "Endknoten" (d. h. eine CPU **12** oder eine E/A-Einheit **17** -**Fig.** 1) kann jedoch keinen Gegendruck ausgeben, weil an einem ihrer Sendeports Gegendruck anliegt. Wird diese Anforderung nicht beachtet, so kann es zu Gegendruck-Systemblockaden kommen, wobei ein Empfangsport nicht empfangen kann, weil ein Sendeport nicht senden kann, weil wiederum der ihm zugeordnete Empfänger Gegendruck ausgibt. Daher können nur Router **14** Gegendruck weitergeben; Endknoten (CPUs **12**, E/A-Paket-Schnittstellen **16**) dürfen empfangenen Gegendruck nicht in Sende-Gegendruck übersetzen.
[0301] Ein Router **14** kann an jedem seiner Empfangsports Gegendruck ausgeben, wann immer weitere an diesem Port ankommende Datensymbole nicht mehr zwischengespeichert oder weitergeleitet werden können. Unkorrekt adressierte Pakete werden von dem Router **14** ausgeschieden.
[0302] Wenn ein Systemelement des Verarbeitungssystems **10** an einem TNet-Link **L**, auf welchem es gerade ein Nachrichtenpaket sendet, ein BELEGT-Befehlssymbol empfängt, setzt das Element das Senden des Pakets aus und beginnt damit, so lange FÜLLEN-Befehlssymbole zu senden, bis eine BEREIT-Befehlssymbol mit jeden Taktzyklus des Sendetakts S_Tkt empfangen wird. FÜLLEN-Befehlssymbole werden weiterhin gesendet. Ausserdem nimmt, wenn ein BELEGT-Befehlssymbol an einem TNet-Link **L** empfangen wird, während der zugeordnete Sendeport gerade kein Paket sendet, das Element, welches das BELEGT-Befehlssymbol empfängt, so lange davon Abstand, eine neue Paketübertragung einzuleiten, bis es in der Folge ein BEREIT-Symbol auf diesem Link empfängt. Der Sendeport behält ansonsten die Fähigkeit, andere Befehlssymbole (BEEIT, BELEGT, usw.) zu übertragen.
[0303] Wann immer ein TNet-Port eines Elements des Verarbeitungssystems **10** den Empfang eines BEREIT-Befehlssymbols erkennt, beendet er die Übertragung von FÜLLEN-Befehlssymbolen von dem zugeordneten Sendeport und fährt mit dem Senden des Pakets fort, das durch den zuvor empfangenen BELEGT-Befehlssymbol ausgesetzt worden war, bzw. er beendet das Eingeben von LEERLAUF-Befehlssymbolen und leitet das Senden eines anhängigen Pakets ein, oder setzt das Senden von LEERLAUF-Befehlssymbolen so lange fort bis ein Paket verfügbar ist.
[0304] Es sei jedoch darauf hingewiesen, dass die BELEGT/Bereit-Flusssteuerung sich nicht auf die Übertragung anderer Befehlssymbole bezieht. Es sei auch noch einmal daran erinnert, dass, wie weiter oben erwähnt, jeder Zyklus eines Sendetaktes, S_Tkt, von der Übertragung eines Befehls- bzw. Datensymbols begleitet wird. Somit müssen alle TNet-Schnittstellen bereit sein, mit jedem Taktzyklus des zugeordneten Sendetaktes, S_Tkt, den die TNet-Schnittstelle empfängt, ein neues Befehls- bzw. Datensymbol anzunehmen.
[0305] Wie noch zu zeigen sein wird, empfangen alle Elemente (z. B. die Router **14**, die CPUs **12**) eines Verarbeitungssystems **10**, die an ein TNet-Link **L** zum Empfang von übertragenen Symbolen angeschlossen sind, diese Elemente über ein Taktsynchronisierungs-FIFO (TS-FIFO). So beinhalten zum Beispiel, wie weiter oben besprochen, die Schnittstelleneinheiten **24** der CPUs **12** alle die TS-FIFOs **102x**, **102y** (dargestellt in **Fig.** 6). Jedes dieser TS-FIFOs **102** ist entsprechend geschaltet, um Befehls- bzw. Datensymbole von einem zugeordneten TNet-Link **L** zu empfangen. Das TS-FIFO muss eine ausreichende Tiefe aufweisen, um eins Geschwindigkeitsanpassung zu ermöglichen, und die elastischen FIFOs müssen eine ausreichende Tiefe aufweisen, um Verzögerungen verarbeiten zu können, zu denen es zwischen der Übertragung eines BELEGT-Befehlssymbols während des Empfangs eines Nachrichtenpakets und der Aussetzung des eingehenden Nachrichtenpa-

kets zugunsten von FÜLLEN bzw. LEERLAUF-Befehlssymbolen kommen kann. Die elastischen FIFOs **506** (**Fig.** 19A) in den Routern **14** sollten auch eine ausreichende Tiefe aufweisen, um die Eingabe von BELEGT- und BEREIT-Befehlssymbolen in den Übertragungspfad zu ermöglichen. So sei etwa unter Bezugnahme auf **Fig.** 1 angenommen, dass die CPU **12A** dabei ist, ein Nachrichtenpaket zu übertragen, das zum Empfang durch einen der E/A-Paket-Schnittstellen **16A** bestimmt ist-und zwar über den Port **2** des Routers **14A**.

[0306] Zur selben Zeit sendet dieselbe E/A-Schnittstelle **16A**, die gerade das von der CPU **12A** gesendete Nachrichtenpaket empfängt, auch ein Nachrichtenpaket auf demselben (bidirektionalen) TNet-Zink **L** an den Port **2** des Routers **14A**. Es sei weiterhin angenommen, dass der Router **14A** eine Verzögerung (Gegendruck) von Seiten des Bestimmungsortes des von der E/A-Paket-Schnittstelle **16A** gerade gesendeten Nachrichtenpakets erfährt. Nach einer gewissen Zeit füllt sich das elastische FIFO **518** (**Fig.** 19A) bis zu einem Punkt, der es erforderlich macht, dass der Router **14A** die E/A-Paket-Schnttstelle auffordert, die Übertragung des Nachrichtenpakets vorübergehend auszusetzen. Demgemäss überträgt der Router **14A** ein BELEGT-Symbol, welches von dem Port **2** abgeht, (demselben Port, von welchem er gerade den von der E/A-Paket-Schnittstelle **16A** kommenden Nachrichtenverkehr empfängt). Dieses BELEGT-Symbol wird in den Symbolstrom des Nachrichtenpakets eingegeben, das gerade von der CPU **12A** durch den Router gesendet wird. Die Eingabe eines BELEGT-Symbols in den Datenstrom eines eingehenden Nachrichtenpakets macht es erforderlich, dass der Router **14A** ein zusätzliches Symbol des eingehenden Pakets speichert. Nachdem das BELEGT-Symbol gesendet ist, kann der Router **14A** mit der Übertragung des eingehenden Nachrichtenpakets von der CPU **12A** fortfahren, und zwar bis zu dem Zeitpunkt, da er wieder mit dem Empfang der diskontinuierlichen Übertragung des von der E/A-Paket-Schnittstelle **16A** kommenden Nachrichtenpakets fortfahren kann. Um die E/A-Paket-Schnittstelle **16A** dazu zu veranlassen, mit der weiteren Übertragung des unterbrochenen Nachrichtenpakets zu beginnen, gibt der Router **14A** ein BEREIT-Signal in den vom Port **2** aus gesendeten Symbolstrom ein, was wiederum erfordert, dass der Router ein weiteres Symbol des von der CPU **12A** kommenden Nachrichtenpakets zu speichern.

[0307] Dieses Paar von BELEGT/BEREIT-Befehlssymbolen kann von jedem Router **14** und von jeder CPU **12** in den Pfad zwischen der E/A-Paket-Schnittstelle **16** und der CPU **12** eingegeben werden. Der direkt an die E/A-Paket-Schnittstelle **16** angeschlossene Router **14** könnte 2n Befehlssymbole (n = die Anzahl der in dem Pfad vorhandenen Router +1) in ein einzelnes Paket eingeben. Das setzt voraus, dass 2n Byte FIFO in dem Router **14** der untersten Ebene (d. h. der einer beliebigen E/A-Paket-Schnittstelle **16** am nächsten gelegenen Router) erforderlich wären, um zu gewährleisten, dass in einer Richtung ausgegebener Gegendruck es nicht erforderlich macht, Gegendruck auch in der anderen Richtung auszugeben. So sei zum Beispiel angenommen, dass eine E/A-Paket-Schnittstelle 16 gerade ein Paket A an einen Router **14** überträgt, während sie zur selben Zeit ein Paket B von demselben Router empfängt und dass der das Paket A empfangende Router dieses aufgrund eines vorhandenen Gegendrucks nicht weiterleiten kann. Dieser Router muss ein BELEGT-Signal in das Paket **8** eingeben, um der E/A-Paket-Schnittstelle **16** anzuweisen, mit der Übertragung des Pakets aufzuhören. Das in das Paket B eingegebene BELEGT-Befehlssymbol verschiebt ein Datensymbol, wodurch die FIFO-Tiefe um eins inkrementiert wird. Durch die darauffolgende Entfernung des Gegendrucks durch das Eingeben von BEREIT wird ein weiteres Datenbyte in Paket B verschoben. Mit der Weiterleitung des Pakets A zum nächsten Router würde sich dieser Vorgang wiederholen. Wenn der Router **14** mehr Datenbytes verschiebt, als das FIFO bewältigen kann, so muss er in Richtung des Ursprungsortes des Pakets B Gegendruck ausgeben.

Paketzustand:

[0308] Jedes übertragene Paket wird unmittelbar von einem TPG- bzw. einem TPB-Befehlssymbol gefolgt, das Auskunft über die Integrität des zugeordneten Pakets gibt. Das Systemelement, von welchem das Paket seien Ursprung nimmt, gibt das entsprechende TPG- bzw. TPB-Befehlssymbol ein. Der Router **14** validiert für sämtliche Pakete den beiliegenden CRC und führt verschiedene Protokollprüfungen durch, während die Pakete von dem Ursprungsort (d. h. der E/A-Paket-Schnittstelle **16** oder der CPU **12A**) kommend durch diesen hindurch an den Bestimmungsort (d. h. die CPU **12A** oder die E/A-Paket-Schnittstelle **16**) fliessen. Falls ein Router **14** in dem Pfad des besagten Datenflusses bei einem eingehenden Paket einen Fehler erkennt, und das Paket mit einem TPG-Befehlssymbol endet (das anzeigt, dass das Paket gut ist), so ersetzt der Router das TPG-Befehlssymbol durch ein TPB-Befehlssymbol.

[0309] Als Fehler, welche zu einer Änderung eines TPG-Befehlssymbols in ein TPB-Befehlssymbol führen können, ist neben dem misslungenen Versuch, die eingegangenen Daten durch die Prüfung ihrer CRC-Daten zu validieren, vor allem die Tatsache zu nennen, dass ein Paket eine grössere Länge aufweist als dies gemäss dem verwendeten Protokoll erlaubt ist. Obwohl jede Paketlänge verwendet werden kann, wird hier mit einer auf 1024 Symbole beschränkten Paketlänge gearbeitet, und zwar einschliesslich des Zustandssymbols (TPG/TPB). Wird also erkannt, dass ein empfangenes Paket diese Grenze umfangmässig überschreitet, so beendet der empfangende Router das Paket mit einem TPB-Befehlssymbol als 1024. Symbol und ignoriert den

Rest des Pakets. Bei dieser Beschränkung der Paketlänge handelt es sich um eine Technik zur Fehlerprüfung, die dazu dient, das Auftreten eines Fehlers in einem paketübertragenden Element zu vermeiden, der dieses ansonsten veranlassen würde, ununterbrochen irgendwelche Nachrichtenpakete auszusenden, was schliesslich zu einer Verstopfung des TNet-Netzwerks führen würde.

[0310] Ein Router **14**, der ein Paket mit einem TPB-Befehlssymbol empfängt, leitet das TPB-Befehlssymbol unverändert weiter, ungeachtet der Ergebnisse der von ihm selbst durchgeführten Prüfungen.

SCHLAFEN-Protokoll:

[0311] Das SCHLAFEN-Protokoll wird von einem Instandhaltungsprozessor über eine weiter unten genauer beschriebene Instandhaltungsschnittstelle (einen Online-Zugangspunkt-OLZP) initiiert. Das SCHLAFEN-Protokoll stellt einen Mechanismus bereit, der es erlaubt, ein oder mehrere TNet-Links **L** an den Paketgrenzen in einen Ruhezustand zu versetzen. Es ist nötig, die Betriebsart zu wechseln (z. B. vom Duplex- in den Simplexbetrieb), um ein Prozessorelement des Systems **10** zu reintegrieren. Die Router **14** müssen sich im Leerlauf befinden (keine Pakete in Übertragung befindlich), um zwischen Betriebsarten zu wechseln, ohne dabei Datenverlust bzw. Datenkorruption zu verursachen. Wird ein SCHLAFEN-Befehlssymbol empfangen, so verhindert das empfangende Element des Verarbeitungssystems **10** den Start der Übertragung irgendeines neuen Pakets auf dem zugeordneten Sendeport, der nur erlaubte Befehlssymbole auf diesem TNet-Link **L** übertragen darf. (Eine Ausnahme stellt das eigenadressierte AtomicWrite-Nachrichtenpaket dar, das weiter unten in dem Abschnitt über Reintegration beschrieben wird.) Jedes Paket, das übertragen wird während das SCHLAFEN-Befehlssymbol empfangen wird, wird auf normale Weise übertragen bis es vollständig ist. Der Sendeport, welcher dem Empfangsport entspricht, an dem das SCHLAFEN-Befehlssymbol empfangen wurde, fährt nichtsdestoweniger fort, erlaubte Befehlssymbole (z. B. BELEGT, BEREIT, LEERLAUF, FÜLLEN) zu übertragen, darf jedoch so lange kein neues Paket zur Übertragung initiieren bis ein BEREIT-Befehlssymbol an dem ihm zugeordneten Empfangsport empfangen wird.

STOPP-Protokoll:

[0312] Das STOPP-Befehlssymbol stellt einen Mechanismus bereit, durch den rasch sämtlichen in einem Verarbeitungssystem **10** vorhandenen CPUs **12** mitgeteilt werden kann, dass es erforderlich ist, sämtliche E/A-Aktivitäten (d. h. Nachrichtenübermittlungen zwischen den CPUs **12** und den E/A-Paket-Schnittstellen **16**, oder Nachrichtenübertragungen zwischen verschiedenen CPUs **12**) einzustellen. Jeder Router **14** verfügt über ein systemweites STOPP-Freigabekonfigurationsregister, welches durch den IP **18** über den OLZP **285'** (**Fig.** 19A) gesetzt werden kann, so dass, wenn von einer CPU **12** ein STOPP-Befehlssymbol empfangen wird, der empfangende Router **14** von jedem seiner Sendeports ein STOPP-Befehlssymbol aussendet und sein Systemstopp-Freigabebit löscht. Ein Router **14** ignoriert sämtliche STOPP-Befehlssymbole, welche empfangen werden während das Systemstopp-Freigabebit sich in gelöschtem Zustand befindet. Auf diese Weise hat das Systemstopp-Freigabebit sowohl die Funktion einer über die Software einstellbaren Freigabeeinrichtung für die Stoppfunktion als auch die der Verhinderung eines unendlichen Rneinanderreihens von STOPP-Befehlssymbolen, nachdem einmal das erste STOPP-Befehlssymbol ausgegeben ist.

[0313] CPUs, welche STOPP-Befehlssymbole an einem ihrer Empfangsports (der Schnittstelleneinheiten **24**) empfangen, geben ein Interrupt an das Interruptregister **280** frei, wenn das Systemstopp-Interrupt freigegeben ist (d. h. wenn die zugeordnete Anordnung des Maskenregisters **282** das Interrupt aktiviert; **Fig.** 14A).

[0314] Die CPUs **12** können mit der Fähigkeit ausgestattet sein, die STOPP-Verarbeitung zu deaktivieren. So können zum Beispiel die Konfigurationsregister **75** der Schnittstelleneinheiten **24** ein "Stoppfreigabe-Register" enthalten, das, sofern es auf einen vorgegebenen Zustand (z. B. NULL) gesetzt ist, die STOPP-Verarbeitung deaktiviert, dabei jedoch die Erkennung eines STOPP-Symbols als einen Fehler meldet.

Routerarchitektur:

[0315] Im folgenden wird auf **Fig.** 19A Bezug genommen, in welcher ein vereinfachtes Blockdiagramm des Routers **14A** veranschaulicht ist. Die anderen Router **14** des Verarbeitungssystems **10** (z. B. die Router **14B**, **14A'**, etc.) sind von im wesentlichen gleicher Bauart und die Beschreibung betreffend den Router **14A** ist daher auch auf die anderen Router **14** anwendbar.

[0316] Wie in **Fig.** 19A veranschaulicht, beinhaltet der Router **14A** sechs TNet-Ports **0**,..., 5, von denen ein jeder seinerseits einen Porteingang **502** (502_0, ... 502_5.) und einen Ausgang **504** (504_0, ... 504_5) beinhaltet.

[0317] Jeder Portausgang **504** verfügt über die **10** aus diesem hinausführenden, weiter oben beschriebenen Signalleitunge n neun Signalleitungen, welche parallele 9-Bit-Befehls-/Datensymbole übertragen und ein Signalbit, das den zugeordneten Sendetakt (S_Tkt) überträgt. In ähnlicher Weise ist ein jeder der Porteingänge **502** entsprechend geschaltet, um **10** parallele, Daten und den Empfangstakt (Empf-Tkt) umfassende Signale

zu empfangen. Wie weiterhin gezeigt, beinhaltet ein jeder Porteingang **502** eine Eingangslogik **505** und ein elastisches FIFO **506**, um eingehende Datenpakete zu empfangen und zwischenzuspeichern, bevor sie an einen Kreuzschienenverteiler **500** angelegt werden. Die Kreuzschienenlogik **500** wird betrieben, um Nachrichtenpakete, welche an den Porteingängen **502** empfangen werden, gemäss den in der Bestimmungsort-ID des Nachrichtenpakets enthaltenen Informationen zu einem Portausgang **504** zu leiten. Die Kreuzschienenlogik **500** wird als regelrechter Kreuzschienenverteiler betrieben, der es erlaubt, Nachrichtenpakete, die an einem beliebigen Porteingang **502** empfangen werden, an einen beliebigen Portausgang **504** zu leiten, auch an jenen Portausgang **504**, der dem Porteingang **502** zugeordnet ist, welcher das Paket empfängt (z. B. Porteingang $\mathbf{502_2}$ und Portausgang $\mathbf{502_2}$. Die Kreuzschienenlogik **500** ist auch in der Lage, zwei oder mehr Nachrichtenpakete von bestimmten Porteingängen **502** an diesen jeweils entsprechende Portausgänge **504** zu leiten. Die Kreuzschienenlogik ist von herkömmlicher Bauart, so dass sich weitere Ausführungen zu ihrem Entwurf erübrigen.

[0318] Zwei der Ports, 4 und 5, des Routers **14A**, die in der Figur durch Schattierung hervorgehoben sind, unterscheiden sich von ihrem Aufbau her gewissermassen von den anderen; diese zwei Ports sind dafür gedacht, als jene Ports benutzt zu werden, die (über die TNet-Links Lx und Ly) direkt an ein CPU-Paar **12** angeschlossen werden. Die Porteingänge $\mathbf{502_9}$, $\mathbf{502_5}$ für diese Ports **4**, **5** sind entsprechend aufgebaut, um in einer frequenzsynchronisierten Umgebung betrieben zu werden, wenn ein Verarbeitungssystem **10** im Duplexbetrieb läuft. Ausserdem wird im Duplexbetrieb ein Nachrichtenpaket, das an einem beliebigen der Eingangsports 0–5 empfangen wird und für eine der CPUs **12**, an welche der Router angeschlossen ist, bestimmt ist, von der Kreuzschienenlogik **500** dupliziert und an beide Portausgänge $\mathbf{405_9}$, $\mathbf{405_5}$ gesendet, welche im Gleichschritt betrieben werden, um an die CPUs, mit denen sie verbunden sind, Symbol für Symbol im wesentlichen zeitgleich dieselben Symbole übertragen. Wenn kein Duplexbetrieb vorliegt (d. h. im Simplex- Betriebsmodus) werden die Porteingänge $\mathbf{502_4}$, $\mathbf{502_5}$, sowie alle anderen Porteingänge im frequenznahen Modus betrieben.

[0319] Ausserdem ist die Eingangslogik **502** für die Router-Ports 4 und 5 auch mit einer Vergleichsschaltung versehen, welche betrieben wird, wenn die CPUs **12A**, **12B** im Duplexbetrieb arbeiten, um einen symbolweisen Vergleich zwischen den von den beiden CPUs empfangenen Befehls/Datensymbolen anzustellen. Demgemäss empfangen die Porteingänge $\mathbf{502_4}$, $\mathbf{502_5}$, wie in **Fig.** 19B veranschaulicht, die Befehls-/Datensymbole von den CPUs, leiten diese durch die (weiter unten besprochenen) Taktsynchronisierungs-FIFOs **518** und vergleichen jedes Symbol, welches die Taktsynchronisierungs-FIFOs verlässt mittels einer gattergesteuerten Vergleichsschaltung **517**. Beim Eintritt in den Duplexbetrieb wird ein in der Steuerlogik **509** enthaltenes (nicht dargestelltes) Konfigurationsregister in einen entsprechenden Zustand gesetzt, so dass es ein DPX-Signal ausgibt. Dieses DPX-Signal wird von der Steuerlogik **509** an die gattergesteuerte Vergleichsschaltung **517** übermittelt, um den symbolweisen Vergleich der aus den beiden Synchronisierungs-FIFOs **518** der Routereingangslogik **502** für die Ports 4 und 5 heraustretenden Symbole zu aktivieren. Wenn das DPX-Bit in der Steuerlogik **509** nicht gesetzt ist, ist die Vergleichsfunktion natürlich deaktiviert.

[0320] Die identischen Symbolströme, welche den Nachrichtenverkehr von den im Duplexmodus betriebenen CPUs **12** ausmachen, werden von den Porteingängen $\mathbf{502_4}$, $\mathbf{502_5}$; empfangen, wobei jedes Symbol des Symbolstroms, das von einem Porteingang empfangen wird, identisch mit jenem ist, das im wesentlichen gleichzeitig von dem anderen Porteingang empfangen wird.

[0321] Um die Synchronisierung im Duplexbetrieb aufrechtzuerhalten, müssen die beiden Portausgänge des Routers **14A**, welche an die CPUs **12** übertragen, im Gleichschritt betrieben werden; das heisst, die Portausgänge müssen entsprechend betrieben werden, so dass Zyklus für Zyklus dieselben Symbole an beide CPUs **12** gesendet werden. Somit muss, unter Bezugnahme auf **Fig.** 2, ein an einem der Ports 0–5 (**Fig.** 19A) des Routers **14A** empfangener und für die CPUs **12** bestimmter Symbolstrom an beide der im Duplexbetrieb laufenden CPUs **12** weitergeleitet werden, und zwar so, dass von den CPUs im wesentlichen zeitgleich zueinander identische Symbole empfangen werden.

[0322] (Die CPUs **12** können eigenadressierte Nachrichtenpakete senden, welche im Duplexbetrieb von den Routern **14** dupliziert und an beide CPUs retourniert werden.) Die Ausgangslogikeinheiten **504**, $\mathbf{504_5}$, die direkt an die CPUs **12** gekoppelt sind, empfangen beide Symbole von der Kreuzschienenlogik **500** (obwohl in dem Bestimmungsort-Feld des Nachrichtenpakets nur eine der beiden duplexbetriebenen CPUs **12**, z. B. die CPU **12A**, als Bestimmungsort angegeben ist) in synchronisierter Weise und legen diese Symbole im wesentlichen simultan an beide CPUs **12** an. Natürlich empfangen die CPUs **12** (genauer gesagt die ihnen zugeordneten Schnittstelleneinheiten **24**) die übertragenen Symbole mit Synchronisierungs-FIFOs mit im wesentlichen demselben Aufbau wie das in **Fig.** 7A veranschaulichte FIFO, so dass, selbst wenn eine geringfügige Echtzeit-Phasendifferenz vorhanden sein mag, mit der die Symbole von den CPUs **12** empfangen werden, die zwischen den beiden CPUs **12** aufrechterhaltene Taktung gewährleistet, dass mit demselben Befehlszyklus von beiden CPUs **12** dasselbe Symbol aus den FIFO-Strukturen entnommen wird, wodurch der für den Duplex-Betriebsmodus erforderliche, synchrone, im Gleichschritt laufende Betrieb der CPUs **12** aufrechterhalten wird.

[0323] Wie im Zusammenhang mit der Abhandlung eines detaillierteren Diagramms der Porteingänge **502** (**Fig.** 20A und 21A) gezeigt werden wird, erfolgt die Steuerung der Wegeleitung hauptsächlich durch die Logik

der Porteingänge **502** in Verbindung mit Konfigurationsdaten, die von dem Instandhaltungsprozessor **18** (über den Online-Zugangspunkt **285'** und den seriellen Bus **19A**; siehe **Fig.** 1A) in Register geschrieben werden, welche in der Steuerlogik **509** enthalten sind.

[0324] Der Router **14A** beinhaltet zusätzlich eine selbstprüfende Logik **511**, um zur Gewährleistung eines ordnungsgemässen Betriebs Prüfungen der verschiedenen Komponenten durchzuführen, aus denen sich der Router **14A** zusammensetzt. Im allgemeinen realisiert die selbstprüfende Logik **511** solche Operationen in Form von internen Paritätsprüfungen, durch Erkennung ungültiger Zustände von Ablaufsteuereinheiten und durch den Vergleich der Ausgabedaten von duplizierter Logik. Die durchgeführten Selbstprüfverfahren sind von konventioneller Natur.

[0325] Der Synchronbetrieb der Router **14A** erfolgt gemäss den von der Taktgabelogik **510** generierten, (lokalen) Taktsignalen.

[0326] Jeder Ausgangsport **504** des Routers **14** ist entsprechend aufgebaut, um die Anforderungen des weiter oben beschriebenen Flusssteuerungsprotokolls zur Übermittlung von Symbolen auf den TNet-Links L zu implementiern. Die Eingangslogik **505** jedes Porteingangs **502** hat ebenfalls eine unterstützende Funktion bei der Aufrechterhaltung der Synchronisierung-zumindest für jene Ports, welche Symbole in frequenznaher Umgebung aussenden- und zwar durch das Entfernen von empfangenen ÜBERSPRINGEN-Befehlssymbolen (SKIP). Das ÜBERSPRINGEN-Befehlssymbol wird in diesem Zusammenhang als Platzhaltersymbol verwendet, das bewirkt, dass Taktzyklen übersprungen werden, wodurch es einem langsameren Empfänger möglich wird, Daten von einem schnelleren Empfänger anzunehmen. Da Geräte an den Enden eines beliebigen TNet-Links L in einer frequenznahen Betriebsumgebung mit einem unterschiedlichen Takt arbeiten, kann mit relativ grosser Wahrscheinlichkeit angenommen werden, dass ein Takt um einen geringfügigen Betrag schneller als der andere ist. Ohne Taktkontrolle könnte es bei einem langsamer empfangenden Element, welches Symbole von einem schneller sendenden Element empfängt, zu einer Überlastung des Eingangs-Taktsynchronisierungs-FIFO des langsamer empfangenden Elements kommen. Das heisst, wenn ein langsamerer Takt verwendet wird, um Symbole aus dem Taktsynchronisierungs-FIFO zu entnehmen, welche mit einem schnelleren Takt in dieses eingegeben werden, so wird es früher oder später zu einem Überlaufen des Taktsynchronisierungs-FIFOs kommen.

[0327] Die in diesem Zusammenhang bevorzugte Technik besteht darin, in regelmässigen Abständen ÜBERSPRINGEN-Symbole in den Symbolstrom einzugeben, um die Wahrscheinlichkeit eines Überlaufens des Taktsynchronisierungs-FIFOs (d. h. des Taktsynchronisierungs-FIFOs **518**, **Fig.** 20A) eines Routers **14** (oder einer CPU **12**) zu vermeiden bzw. wenigstens zu minimieren, zu dem es infolge eines S Tkt-Signals kommen könnte, welches Symbole mit einer geringfügig höheren Frequenz in das FIFO eingibt als der lokale Takt, der verwendet wird, um Symbole aus dem Synchronisierungs-FIFO zu entnehmen. Das Verwenden des ÜBERSPRINGEN-Symbols, um eine Eingabeoperation (in das FIFO) zu umgehen, wirkt sich dahingehend aus, dass der Eingabezeiger des FIFOs jedes mal wenn ein ÜBERSPRINGEN-Befehlssymbol empfangen wird, blockiert wird, so dass, was das Taktsynchronisierungs-FIFO betrifft, der das ÜBERSPRINGEN-Symbol begleitende Sendetakt fehlte.

[0328] Dadurch kann eine in jedem Porteingang **502** enthaltene Logik die zur Synchronisierung in einer frequenznahen Taktgabeumgebung verwendeten ÜBERSPRINGEN-Befehlssymbole erkennen und über einen Schlüssel deaktivieren, so dass nichts in das FIFO eingegeben, andererseits jedoch ein Symbol daraus entnommen wird. Vorzugsweise werden ÜBERSPRINGEN-Symbole ungefähr alle 512 Sendetakte eingefügt. Da Symbole auf Links L (z. B. zwischen einer CPU **12** und einem Router **14**, oder zwischen Routern **14** oder zwischen einem Router **14** und einer E/A-Schnittstelleneinheit **16A**-**Fig.** 1) mit einer Rate von 50 MHz übertragen werden, kann dadurch schlimmstenfalls ein Frequenzunterschied von 2000 ppm bewältigt werden.

[0329] Die elastischen FIFOs **506** an jedem Porteingang **502** sind von herkömmlicher Bauart und werden dazu verwendet, um Signalunbeständigkeiten in dem Symbolstrom zu absorbieren und zu glätten, welche zum Beispiel durch das Eingeben von Flusssteuerungs- und Befehlssymbolen in das in Übertragung befindliche Nachrichtenpaket verursacht werden. Am wichtigsten jedoch ist vielleicht, dass die eastischen FIFOs **506** eine Zwischenspeicherung des eingehenden Nachrichtenverkehrs erlauben, wenn der Ausgangsport belegt ist.

[0330] Der Router **14A** kann wie jedes andere Element des Systems **10** mit "Gegendruck" konfrontiert werden, wenn er dabei ist, ein empfangenes Nachrichtenpaket zu dem Gerät, für das es bestimmt ist, weiterzuleiten, und das Bestimmungsgerät vorübergehend sein Unvermögen, weiter Symbole zu empfangen, signalisiert (z. B. mittels eines BELEGT-Steuersymbols). Für eine angemessene Implementierung des Gegendruck-Verfahrens ist es erforderlich, dass die elastischen FIFOs **506** eine Tiefe aufweisen, (d. h.. eine ausreichende Anzahl von Speicherorten beinhalten), die gross genug ist, um so lange eingehende Symbole zu empfangen und aufzubewahren, nachdem das Bestimmungsgerät aufgehört hat, Daten zu empfangen, bis das vorgeschaltete Gerät (d. h. das Gerät, welches das Nachrichtenpaket an den Router liefert) auf das BELEGT-Symbol reagieren kann und FÜLLEN- bzw. LEERLAUF-Symbole liefert (welche zwar empfangen und in die Taktsynchronisierungs-FIFOs eingegeben, jedoch nicht an die elastischen FIFOs weitergereicht werden). Kurzum, jedes elastische FIFO **506** muss über ausreichend Platz verfügen, um das Speichern von Symbolen fortsetzen zu

können, bis das sendende Gerät vorübergehend das Senden einstellen kann.

[0331] Um zu einer Verringerung der Signalunbeständigkeit beizutragen, arbeiten die elastischen FIFOs mit oberen und unteren "Wasserzeichen". Wenn das elastische FIFO **506** beginnt, sich zu füllen, und das obere Wasserzeichen erreicht, wird von jenem Sendeport, welcher dem Empfangsport entspricht, der den Symbolstrom empfängt, ein Gegendruck-Symbol (z. B. BELEGT) übertragen. Wird zum Beispiel gerade ein Symbolstrom von dem Router-Porteingang **502_3** empfangen und zeigt die zum Steuern des elastischen FIFOs **506_3** verwendete FIFO-Steuerlogik **546** an, dass das FIFO allmählich voll wird (d. h. das obere Wasserzeichen überschritten hat), so signalisiert der Eingangsport **502** dem entsprechenden Ausgangsport **504_3**, dafür zu sorgen, dass ein BELEGT-Symbol übertragen wird. Der BELEGT-Zustand wird durch den Router **14** (und durch das Gerät am anderen Ende des TNet-Links L, das das Paket senden wollte, aufrechterhalten, bis die Tiefe des elastischen FIFOs **506_3** sich unterhalb des von der FIFO-Steuerlogik **546** festgelegten, unteren Wasserzeichens (**Fig.** 20A) befindet, woraufhin der Portausgang **504_3** angewiesen wird, ein BEREIT-Symbol zu senden und dadurch die Wiederaufnahmen des Sendens des Symbolstroms anzufordern.

[0332] Am anderen Ende des TNet-Links L reagiert das Gerät, das dabei war, das Nachrichtenpaket zu senden, auf den Empfang des BELEGT-Befehlssymbols an dem Eingangslink mit FÜLLEN-Befehlssymbolen, die auf dem zugeordneten Ausgangs-Befehlslink übertragen werden. Das sendende Gerät fährt damit fort, FÜLLEN-Symbole zu senden und die weitere Übertragung des Nachrichtenpakets zurückzuhalten, bis das Gerät, das das BELEGT-Befehlssymbol sandte, ein BEREIT-Symbol sendet. Die Übertragung des Nachrichtenpakets wird fortgesetzt bis sie vollendet ist bzw. bis von dem Empfänger erneut Gegendruck ausgegeben wird.

[0333] Es ist zu beachten, dass das elastische FIFO nicht nur breit genug sein muss, um diese durch "Gegendruck" bedingten Signalunbeständigkeiten zu bewältigen, sondern dass es auch in der Lage sein muss, Datensymbole zu speichern, die sich in dem FIFO ansammeln, während Steuersymbole in den Symbolstrom eingefügt werden, durch welche die TNet-Links L in andere Richtungen gesteuert werden. Eine BELEGT/BEREIT-Kombination nimmt zwei Zyklen von dem Portausgang **504** weg, wodurch das elastische FIFO **506**, das diesen Portausgang **504** versorgt, dazu veranlasst wird, sich um zwei Zeichen zu füllen. Um Signalunbeständigkeiten möglichst gering zu halten, muss die Dimensionierung der elastischen FIFOs **506** (und die Anbringung des oberen und des unteren Wasserzeichens) dergestalt erfolgen, dass das Einfügen von mindestens zwei Zeichen, vorzugsweise jedoch von mehreren Zeichen in den Symbolstrom möglich ist, bevor Gegendruck ausgegeben wird. Innerhalb der Umgebung des hier beschriebenen Systems sind die elastischen FIFOs in der Lage, vorübergehend 96 Symbole zu speichern.

[0334] Der Router **14A** ermöglicht die Einfügung einer vorgegebenen Anzahl von Symbolen bevor Gegendruck benötigt wird (Gegendruck wird anlässlich des nächstfolgenden Symbols ausgegeben, das eingeht, nachdem die vorgegebene Zahl empfangen und vorübergehend gespeichert worden ist). Die auf **96** Symbole dimensionierte Tiefe des elastischen FIFOs **506** ermöglicht einen normalen Aufbau einer vorgegebenen Anzahl von Symbolen und **12** Zyklen von Gegendruckverzögerung, bevor ein Porteingang **502** (unter Ausgabe von Gegendruck) die weitere Annahme von Daten einstellen muss oder in die peinliche Lage gerät, einen überlaufbedingten Datenverlust hinnehmen zu müssen.

[0335] Jeder der Porteingänge **502** weist im wesentlichen einen identischen Aufbau auf, so dass die Beschreibung eines von ihnen auch auf alle anderen anwendbar ist. Demgemäss wird, wie in **Fig.** 20A veranschaulicht, das detaillierte Blockdiagramm des Porteingangs **502o** für Port **0** gezeigt. Der Porteingang **502o** empfängt jedes 9-Bit-Daten-/Befehlssymbol an einem Eingangsregister **516**, wo es durch den begleitenden Sendetakt (S_Tkt) zwischengespeichert wird. Die empfangenen Symbole werden dann von dem Eingangsregister **516** weitergereicht und, ebenfalls durch den S_Tkt, an ein Taktsynchronisierungs-FIFO **518** angelegt werden. Bei dem Taktsynchronisierungs-FIFO **518** handelt es sich von der Logik her gesehen um dasselbe wie jenes, das in den **Fig.** 8A und 8B veranschaulicht ist und das in den Schnittstelleneinheiten **24** der CPUs **12** verwendet wird. In diesem Fall umfasst das Taktsynchronisierungs-FIFO **518**, wie in **Fig.** 20A gezeigt, eine Mehrzahl von Registern **520**, welche parallel die Ausgabesignale des Eingangsregisters **516** empfangen. Einem jeden der Register **520** zugeordnet ist ein zweistufiger Gültigkeitsbit-Synchronisierer (G), der in **Fig.** 20B genauer im Detail dargestellt und weiter unten besprochen wird. Der Inhalt eines jeden der Register **520** wird zusammen mit dem ein Bit umfassenden Inhalt eines jeden zugeordneten zweistufigen Gültigkeitsbit-Synchronisierer **522** an einen Multiplexer **524** angelegt und das ausgewählte Register wird zusammen mit dem Synchronisierer durch ein Registerpaar **526** aus dem elastischen FIFO **506** entnommen. Die Auswahl des Registers **520**, das den Inhalt des Eingangsregisters **520** empfängt, wird durch den Zustand des Eingabewahl-Signals bestimmt, welches von einer Eingabezeiger-Logikeinheit **530** geliefert wird; und die Auswahl des Registers **520**, das seinen Inhalt über den MUX **524** an die Register **526** bereitstellt, wird durch den Zustand eines Entnahmewahl-Signals bestimmt, welches von einer Entnahmezeiger-Logik **532** geliefert wird. Die Eingabe- und Entnahmezeiger-Logik **530**, **532** werden von der Synchronisierungs-FIFO-Steuerlogik **534** gesteuert. Die Synchronisierungs-FIFO-Steuerlogik **534** empfängt den eingehenden S_Tkt, um die Eingabezeiger-Logik **530** (sowie das Eingangsregister **516**) zu betreiben und das Laden des von der Eingabezeiger-Logik **530** gewählten Registers **520** zu besorgen. In ähnlicher Weise empfängt die Synchronisierungs-FIFO-Steuerlogik **534** das

routereigene, lokale Taktsignal (Empf-Tkt), um damit die Entnahmezeiger-Logik **532** zu steuern.

[0336] Der folgende kurze Exkurs behandelt unter Bezugnahme auf **Fig.** 20B etwas mehr ins Detail gehend den Gültigkeitsbit-Synchronisierer **522**, welcher ein D-Flipflop **541** mit Aktivierfunktion, ein Verzögerungselement **541a**, ein ODER-Glied **541b**, ein D-Flipflop **542** (mit Setz/Rücksetz/Aktivierungs-Fähigkeit, um die in der weiter unten dargestellten Wahrheitstabelle gezeigten Funktionen bereitzustellen), und ein D-Flipflop **543** umfasst. Das D-Flipflop **541** ist entsprechend verkoppelt, um die Ausgangssignale der ÜBERSPRINGEN-Prüflogik **541** an seinem Dateneingang (D) zu empfangen. Der Aktivierungs-Eingang des Flipflop **541** empfängt die von dem Eingabezeiger **530** gelieferte, decodierte Information, Eingabewahl, und der Takt-Eingang (Tkt) des Flipflop **541** empfängt den eingehenden Sendetakt (S_Tkt), welcher die eingehenden Symbole begleitet. Der Ausgang (Q) des Flipflop **541** wird an einen Eingang des ODER-Glieds **541b** und, über das Verzögerungselement **541a**, auch an dessen anderen Eingang angelegt. Der Ausgang (Q) des Flipflop **541** wird (auf einen logischen Zustand "eins") gesetzt, wenn das von der Zeigerlogik **530** (**Fig.** 20A) kommende Eingabewahl-Signal das Register **520** des FIFOs auswählt, welchem der Gültigkeitsbit-Synchronisierer zum Empfang des nächsten Symbols-falls es sich dabei nicht um ein ÜBERSPRINGEN-Symbol handelt-zugeordnet ist.

[0337] Das Verzögerungselement **541a** und das ODER-Glied **541b** werden betrieben, um eine Impulsdehnungsschaltung herkömmlicher Bauart zu bilden, die gewährleistet, dass das Signal an dem Setz-Eingang des Flipflops **542** eine Dauer von zumindest einem Taktintervall hat. Da dies der Fall ist, und da bekannt ist, dass die Signale des lokalen (routereigenen) Empfangstaktes (Empf-Tkt) und des empfangenen Sendetaktes (S_Tkt) ähnliche, um nicht zu sagen identische Frequenzen aufweisen, wird klar dass zumindest ein aktiver Übergang des Empfangstaktes es dem Flipflop **542** erlaubt, durch Setzen des Ausgangs (Q) des Flipflops das gedehnte Signal aufzuzeichnen (siehe die Wahrheitstabelle weiter unten). Das D-Flipflop **543** fungiert als zusätzliche Synchronisierungsstufe, indem es ein stabiles Niveau am G-Ausgang in Bezug auf den lokalen Empf-Tkt gewährleistet. Das Entnahmewahl-Signal, eine decodierte Information des Entnahmezeigers **532**, wird an den Aktivierungs-Eingang des Flipflop **532** angelegt und erlaubt es dem Entnahme-Signal (einem von der Synchronisierungs-FIFO-Steuereinheit **534** kommenden, periodischen Impuls), das Gültigkeitsbit auf diesem Gültigkeits-Synchronisierer **522** zu löschen, nachdem das zugeordnete Register **520** gelesen worden ist.

Wahrheitstabelle

Setzen	Reset	Aktivieren	Q_n	Q_{n+1}
1	X	X	X	1
0	X	0	0	0
0	X	0	1	1
0	1	1	X	0
0	0	1	1	1
0	0	1	0	0

[0338] Zusammenfassend gesprochen wird der Gültigkeits-Synchronisierer **522** betrieben, um ein "Gültig"-Signal (G) auszugeben, wenn ein Symbol in ein Register **520** des FIFO **518** geladen wird, mit welchem dieses Symbol als ein gültiges Symbol identifiziert wird. Wenn es sich andererseits bei dem Symbol um ein ÜBERSPRINGEN-Symbol handelt, so nimmt der Ausgang der ÜBERSPRINGEN-Prüflogik **540** einen L-Zustand an, was bewirkt, dass das Flipflop **541** (d. h. der Datenausgang (Q)) auf Null bleibt, wodurch angezeigt wird, dass das zugeordnete Symbol nicht gültig ist und ignoriert werden sollte.

[0339] Der Inhalt des Eingangsregisters **516**, um nun mit **Fig.** 20A fortzufahren, wird auch an die ÜBERSPRINGEN-Prüflogik **540** angelegt. Der Empfang eines ÜBERSPRINGEN-Befehlssymbols bewirkt, wenn er von der ÜBERSPRINGEN-Steuerlogik **540** erkannt wird, dass die Funktion der Eingabezeigerlogik **530** gehemmt wird und das Laden des betreffenden Symbols in das Taktsynchronisierungs-FIFO **518** während des einen begleitenden Taktintervalls des Sendetakts verhindert wird Der Empfang eines ÜBERSPRINGEN-Befehlssymbols führt nicht dazu, dass der Eingabezeiger **530** vorgerückt wird oder dass das Gültigkeitsbit G gesetzt wird, da nämlich in der Praxis die Entnahme-Seite des FIFO in keiner Weise über den Empfang des ÜBERSPRINGEN-Symbols durch die Eingabe-Seite informiert wird.

[0340] Eingehende Daten-/Befehlssymbole, welche von der Registerpipeline **526** weitergereicht werden, werden ebenfalls an die Befehlsdecodierlogik **544** angelegt, wo jegliche Befehlssymbole des eingehenden Daten-

stroms decodiert werden und dazu verwendet werden, die FIFO-Steuerlogik **546** zu steuern. Ausser Ihrer Funktion beim Betreiben der elastischen FIFOs **506**, wird die FIFO-Steuerlogik **546** betrieben, um die erforderlichen Handshake-Signale an die Portausgänge **504** zu generieren, welche die Symbole über die Kreuzschienenlogik **500** von dem Porteingang $\mathbf{502_0}$ empfangen.

[0341] Die Befehls-/Datensymbole werden auch an die Protokoll- und Paketprüflogik **550** angelegt, welche betrieben wird, um das Zinkebenen- und Paketprotokoll zu überprüfen, und zwar einschliesslich des (weiter unten beschriebenen) Linkebenen-Halteprotokolls, der Nachrichtenpaket-Terminierungsprüfungen, usw.

[0342] Diese Symbole, die keine Befehlssymbole sind (welche, wenn sie aufgefunden werden, aus dem Symbolstrom entfernt werden), d. h. die Datensymbole, werden an das elastische FIFO $\mathbf{506_0}$ weitergereicht und dort gespeichert, um in der Folge, wenn auf sie zugegriffen wird, an die Kreuzschienenlogik **500** übermittelt zu werden. Die Bestimmungsort-ID des Nachrichtenpäkets wird ebenfalls an eine Zielport-Auswahllogik **560** übermittelt. Die Zielport-Auswahllogik **560** wird betrieben, um ausgehend von der empfangenen Bestimmungsort-ID und von Informationen aus bestimmten Konfigurationsregistern des Routers die "Zielport-Adresse" des Portausgangs **504** zu bestimmen, an welche die Nachricht zum Zweck der Übertragung zu leiten ist. Die Zielport-Auswahllogik **560** generiert einen drei Bit umfassenden Code, der an die Kreuzschienenlogik **500** angelegt wird und von dieser dazu verwendet wir, um die richtige Quervernetzung zu erstellen.

[0343] Der ausgewählte Portausgang **504** muss jedoch "aktiviert" sein, um Nachrichtenpakete von dem Porteingang $\mathbf{502_0}$ empfangen zu können. Zu diesem Zweck beinhaltet der Porteingang $\mathbf{502_0}$ ein 6-Bit-Portfreigaberegister **652**, welches Informationen darüber enthält welche Portausgänge **504** berechtigt sind, Nachrichtenpakete von dem Porteingang **502** zu empfangen. Jede Bitposition des Portfreigaberegisters **562** entspricht einem Portausgang **504** und abhängig von dem Zustand der jeweiligen Bitposition kann der entsprechende Portausgang "aktiviert" sein, so dass ein Nachrichtenverkehr von dem Porteingang zu ihm hin möglich ist, oder "deaktiviert", in welchem Fall kein Nachrichtenverkehr von dem Eingang $\mathbf{502_0}$ zu ihm geleitet werden kann. Angenommen zum Beispiel der Porteingang $\mathbf{502_0}$ beginnt, ein Nachrichtenpaket zu empfangen, das eine Bestimmungsort-Information enthält, welche die Zielport-Auswahllogik veranlassen würde, den Portausgang **504y** als Zielport anzugeben. Es sei jedoch weiterhin angenommen, dass der Zustand des Portfreigaberegisters **562** entsprechend beschaffen ist, so dass es dem Portausgang **504** nicht erlaubt ist, Nachrichtenverkehr von dem Porteingang $\mathbf{502_0}$ zu empfangen. Da dies der Fall ist, verhindert der Inhalt des Portfreigaberegisters **562**, dass irgendeine von der Zielport-Auswahllogik **506** erstellte Auswahlinformation an die Kreuzschienenlogik **500** angelegt wird. Das Paket wird vielmehr verworfen und ein Fehlersignal wird erstellt, um anzuzeigen, dass der Router **14A** ein Paket empfangen hatte, das für einen Port bestimmt war, welcher für den Port, an dem das Paket empfangen worden war, als Zielport nicht erlaubt war. Der Fehler wird über den OLZP **285'** (**Fig.** 19A) an den IP **18** gemeldet.

[0344] Das Portfreigabemerkmal dient daher dazu, gezielt bestimmte Routing-Pfade durch den Router **14** zu verhindern. Dieses Merkmal kann einen wichtigen Mechanismus bei der Verhinderung von Systemblockierungszuständen darstellen. Ein Systemblockierungszustand tritt dann ein, wenn ein zur Übermittlung von Nachrichten dienendes Netzwerk "Routingschleifen" enthält, welche aus Routinggeräten und Verbindungs-Links gebildet werden. Dies läuft so ab, dass eine Nachricht, die an einem Routinggerät empfangen wird, davon abgehalten wird, den Router an einem bestimmten Ausgangsport zu verlassen, da bereits eine andere Nachricht gerade über diesen Port geleitet wird. Diese andere Nachricht wird jedoch ihrerseits an einem anderen Routinggerät durch eine dritte Nachricht blockiert, und so weiter. Alle Nachrichten werden jeweils in einer kreisförmigen Schleife blockiert. Es bewegt sich nichts mehr, da jede Nachricht in der Schleife von einer anderen Nachricht in der Schleife blockiert wird und ihrerseits eine andere Nachricht in der Schleife blockiert; die Nachrichten blockieren einander gegenseitig. Ohne sachgemässem Entwurf kann es bei grossen Routernetzwerken unter Umständen zu einer ganzen Reihe von systemblockadeträchtigen Umgebungen kommen, welche zu Gruppen von Nachrichtenpaketen führen, die in dem Übertragungsnetzwerk nicht mehr weiter kommen, und zwar aufgrund solcher zirkulärer Abhängigkeiten, bei denen ein jedes Nachrichtenpaket aus der Gruppe von Nachrichtenpaketen warten muss bis ein anders Paket weiter vorangekommen ist bevor es selber Zugang einem Verbindungs-Link erhält. Dank der Möglichkeit, die Freigabe bestimmter Übermittlungspfade durch den Router zu deaktivieren, wird es möglich, sämtliche möglicherweise auftretenden Routingschleifen zu eliminieren und somit das Auftreten von Systemblockadezuständen zu vermeiden.

[0345] Natürlich besteht der beste Schutz gegen Routingschleifen und möglicherweise daraus entstehende Systemblockaden in erster Linie darin, sicherzustellen, dass zur Auswahl der Zielportadresse ordnungsgemässe Routinginformationen verwendet werden, so dass ein eingehendes Nachrichtenpaket nicht zu einem Ausgangsport des Routers **14** geleitet werden, der Teil einer Routingschleife sein könnte. Durch die Fähigkeit, die Freigabe bestimmter Routingpfade durch den Router **14** zu deaktivieren, wie dies durch die Portfreigaberegister möglich ist, kann gewährleistet werden, dass Routingfehler oder sonstige Fehler nicht zu Systemblockadezuständen führen. Die Realisierung dieses Konzepts wird weiter unten im Detail besprochen.

[0346] Auch hier wieder, um nun mit **Fig.** 20A fortzufahren, werden wenn die Header der eingehenden Nachrichtenpakete empfangen werden die Bestimmungsort-IDs der Reihe nach an die Zielport-Auswahllogik **560**

weitergereicht und dort in der Reihenfolge ihres Eintreffens einer Prüfung unterzogen. Die Zielport-Auswahllogik **560** stellt daraufhin eine Zielportadresse zusammen, in welcher der zugeteilte Portausgang angegeben ist. Diese Adresse wird an die Kreuzschienenlogik **500** angelegt, welche die Auswahl der entsprechenden Kreuzschiene vornimmt, die die Ausgangsdaten des das Nachrichtenpaket empfangenden, elastischen FIFOs **506** an den korrekten Portausgang **504** übermittelt- vorausgesetzt, wie weiter oben erwähnt, der ausgewählte Portausgang **504** ist für die Annahme der Porteingangsdaten freigegeben. (Falls es sich bei dem Router **14** um einen im Duplexbetrieb arbeitenden Router mit direkter TNet-Anbindung an die CPUs **12** handelt, wird ein eingehendes Nachrichtenpaket, das für die CPUs bestimmt ist, von der Kreuzschienen-Logikeinheit insofern dupliziert, als es gleichzeitig an beide Portausgänge **504** und $\mathbf{504_5}$ geleitet wird.) Die Zielport-Auswahllogik **560** wird im Detail in **Fig.** 21A veranschaulicht, und beinhaltet, wie in der Figur dargestellt, ein Bestimmungsortregister **570**, welches von den elastischen FIFOs **506** der Portausgänge **502** (**Fig.** 19 und 22A) die 3 Byte umfassende Bestimmungsort-ID des eingehenden Pakets empfängt. Die Bestimmungsort-ID beinhaltet die folgenden drei bereits weiter oben in Verbindung mit **Fig.** 3B besprochenen Felder: Regions-ID, Geräte-ID, und ein 1-Bit-Feld welches das Pfadauswahlbit (P) enthält. Die Regions-ID gibt, wie schon der Name sagt, einen Bestimmungsort nach Region an und die Geräte-ID gibt das konkrete Gerät innerhalb dieser Region an. Das Pfadauswahlbit (P) gibt an, welcher der beiden Pfade (X oder Y) verwendet werden soll, um auf zwei Subprozesseinheiten des Geräts zuzugreifen.

[0347] Die Router **14** schaffen die Fähigkeit, ein grossdimensioniertes, vielseitiges Routingnetzwerk zum Beispiel für massiv parallele Verarbeitungsarchitekturen aufzubauen. Die Router werden entsprechend ihrer Anordnung (d. h.. Ebene) in dem Netzwerk durch die Information konfiguriert, welche in gewissen in der Steuerlogik **509** eines Routers enthaltenen Konfigurationsregistern festgelegt ist. Diese Konfigurationsregister sind in **Fig.** 21A dargestellt, und zwar als obere Regions-ID-Register $\mathbf{509_a}$, untere Regions-ID-Register $\mathbf{509_b}$, HiLo-Register **509**, Vorgabe-Register $\mathbf{509_4}$, Quervernetzungsport-Register $\mathbf{509_a}$, Leitweg-zu-Standard-Register **509**, Geräte-ID-Vergleichs-Register $\mathbf{509_8}$ und Seitenregister $\mathbf{509_h}$. Zwei zusätzliche Konfigurationsregister werden in **Fig.** 21C als Gerätepositions- und Mit-Register $\mathbf{509_j}$ bzw. $\mathbf{509_k}$ gezeigt. Der Inhalt dieser verschiedenen Konfigurationsregister bestimmt gemeinsam mit der Bestimmungsort-ID und dem begleitenden Pfadauswahlbit (P) eines Nachrichtenpakets die Auswahl des Portausgangs **504**, zu welchem das Nachrichtenpaket durch die Querschienenlogik **500** geleitet wird.

[0348] Die Ebene eines Routers ist mitbestimmend dafür, welche Abschnitte der Bestimmungsort-ID bei der Auswahl eines Zielports verwendet werden und ob eine algorithmische Adressauswahl verwendet werden kann. Zu diesem Zweck ist die Regions-ID weiterhin in zwei überlappende 10-Bit-Ebenenbezeichnungen unterteilt. Die wichtigsten **10** Bit des Inhalts der Regions-ID werden als die obere Ebene definiert, während die weniger wichtige 10 Bits der Regions-ID eine untere Ebene darstellen. Die beiden Ebenenbezeichnungen werden jeweils an einem entsprechenden der beiden 10-Bit-Eingänge eines Multiplexers **572** angelegt. Der Multiplexer **572** wählt einen der beiden 10-Bit-Eingänge als Reaktion auf den Inhalt des die (obere oder untere) Router-Ebene angebenden HiLo-Registers **509** aus und führt die ausgewählten **10** Bits einer Routing-Tabelle **584** als Adresse zu.

[0349] Die **Fig.** 20A und 21A veranschaulichen die Porteingänge **502** und zeigen, dass jeder über eine eigene, individuelle Zielport-Auswahllogik **560** und Routing-Tabelle **584** verfügt. Für einschlägig gebildete Fachleute ist jedoch offensichtlich, dass aus Gründen der Platzersparnis eine einzige Routing-Tabelle zur gemeinsamen Verwendung durch die Zielport-Auswahllogiken aller sechs Porteingänge **502** verwendet werden kann. Die Ausgabedaten des Multiplexers **572** können ihrerseits an die Routing-Tabelle **584** (die in der Zustands- und Steuerlogik enthalten sein kann) multiplexiert werden, und zwar auf der Basis einer Zuordnungsentscheidung, unter Verwendung eines herkömmlichen Verfahrens der zyklischen Zuordnung Das Ergebnis des Zugriffs auf die Routing-Tabelle wird an den Eingang des Multiplexers **586** zurückgegeben und an diesen angelegt. Der Einfachheit halber werden diese Schritte der Zuordnungsentscheidung und des Multiplexens in **Fig.** 21 nicht dargestellt.

[0350] Die 4 wichtigsten Bits der Regions-ID werden ausserdem an eine 4-Bit-Vergleichsschaltung **574** angelegt, wo sie mit dem Inhalt eines 'obere Regions-ID'-Registers $\mathbf{509_a}$ verglichen werden. Die am wenigsten bedeutenden 10 Bits der Regions-ID werden an eine Vergleichsschaltung **587** gekoppelt, wo sie mit dem Inhalt des 'untere Regions-ID'-Registers $\mathbf{509_b}$ verglichen werden.

[0351] Die Routing-Tabelle **584**, die zum Beispiel die Form eines Schreib-Lese-Speichers haben kann, wird betrieben, um eine Mehrzahl von 3-Bit-Zielportangaben zu speichern. Wenn von dem einen oder anderen 10-Bit-Abschnitt der Regions-ID auf sie zugegriffen wird, werden die drei Bits an Zielport-Auswahlinformation an einen Eingang des Multiplexers **586** gekoppelt; der andere Eingang des Multiplexers **586** empfängt den 3 Bit umfassenden Inhalt des Standardportregisters $\mathbf{509_4}$. Die Auswahl der Ausgangsdaten der Routing-Tabelle **584** durch den Multiplexer **586** erfolgt entweder über den Inhalt des HiLo-Registers **509**, im Fall einer EINS (wodurch angezeigt wird, dass es sich bei dem Router um einen Router der oberen Ebene handelt), oder über einen erfolgreichen Vergleich zwischen dem Inhalt des 'obere Regions-ID'-Register $\mathbf{509_a}$ und den vier MSBs der Regions-ID durch die Vergleichsschaltung **574** (wodurch angezeigt wird, dass der Bestimmungsort in der-

selben "hohen Region" gelegen ist wie dieser "Router niedriger Ebene"). Ist keine dieser Bedingungen erfüllt, so wählt der Multiplexer **586** stattdessen den (3 Bit umfassenden) Inhalt des Standard-Port-Registers **509a** als Zielportangabe aus.

[0352] Die Routing-Tabelle **584** kann jede beliebige Grösse haben. Für einschlägig gebildete Fachleute ist jedoch klar, dass die Grösse der Routing-Tabelle **584** von Faktoren wie zum Beispiel der Anzahl der adressierbaren Elemente in dem System, in welchem der Router verwendet wird, und dem für die Tabelle verfügbaren Speicherplatz bestimmt wird. Die Zielport-Auswahllogik **560** führt einen neuartigen Kompromiss ein, welcher darin besteht, sofern erforderlich eine Tabellen-Nachschlagetechnik zu verwenden, und falls dies nicht erforderlich ist, ein algorithmisches Routingverfahren zu benutzen, um Platz in der Routing-Tabelle zu sparen. Diese Kombination ermöglicht es, eingehende Nachrichtenpakete an jeden beliebigen der sechs verfügbaren Ports des Routers **14** zu leiten und auch von diesem aus weiterzuübertragen und schafft so eine sehr vielseitige Routingfähigkeit.

[0353] Die von dem Multiplexer **586** ausgewählte, 3 Bit umfassende Zielportangabe wird an einen (3 Bit umfassenden) Eingang eines weiteren Multiplexers **590** übermittelt, welcher zwischen den Ausgangsdaten des Multiplexers **586** und dem 3 Bit umfassenden Inhalt des Quervernetzungsportregister **509** auswählt. Welcher der beiden Werte ausgewählt wird, wird von der Seite (z. B. X oder Y) bestimmt, auf welcher der Endbestimmungsort gelegen ist, was durch den Zustand des Pfadauswahlbits (P) der eingehenden Nachricht angezeigt wird. Das Pfadauswahlbit (P) des eingehenden Nachrichtenpakets wird durch die Vergleichslogik **592** mit dem Inhalt des Seitenregisters $\mathbf{509_h}$, verglichen, wobei die von dem Multiplexer **590** zu treffende Auswahl durch die Ausgabedaten der Vergleichslogik bestimmt wird. Befindet sich der Router nicht auf der Seite (X oder Y), für welche das Nachrichtenpaket bestimmt ist, so beeinflussen die Ausgabedaten der Vergleichslogik **592** die Auswahl des Inhalts des Quervernetzungsportregisters $\mathbf{509_P}$ entsprechend. Dadurch wird das Nachrichtenpaket zu jenem Portausgang **504** geleitet, welcher entweder auf direktem oder auf indirektem Weg (d. h. über einen anderen Router bzw. über andere Router) das Nachrichtenpaket von der Xbzw. der Y-Seite, auf welcher sich der Router befindet, auf die andere Seite leitet, auf der sich der Bestimmungsort des Nachrichtenpakets befindet.

[0354] Die von dem Multiplexer **590** getroffene Auswahl wird an den Eingang eines Multiplexers **594** angelegt, dessen Auswahleingang die Ausgangsdaten der UND-Gatter-Logik **596** empfängt. Der Multiplexer **594** wählt zwischen den von dem Multiplexer **590** und einem Multiplexer **598** bereitgestellten Portadressen aus. Der Multiplexer **598** wählt seinerseits zwischen den Ausgangsdaten der algorithmischen Routinglogik **600** und dem Inhalt des Standard-Port-Registers $\mathbf{509_d}$ aus. Diese Auswahl wird von der Auswahl- und Vergleichsschaltung **601** getroffen, welche den Inhalt des Geräte-ID(Konfigurations-)Registers **509** und einen ausgewählten Abschnitt der sechs Bits der Geräte-ID der eingehenden Nachricht empfängt. Es ist hier nicht im speziellen dargestellt, dass die jeweiligen Gerätebitpositions- und -expansionsregister $\mathbf{509_j}$ bzw. $\mathbf{509_k}$ der algorithmischen Routinglogik **600** (**Fig.** 21C) ebenfalls an die Auswahl- und Vergleichslogik **601** (**Fig.** 21C) angelegt werden. Die in den Gerätebitpositions- und -expansionsregistern $\mathbf{509_j}$ und $\mathbf{509_k}$ enthaltenen Werte dienen dazu, die Geräte-ID-Bits der Nachricht zu maskieren, so dass nur die nicht von der algorithmischen Routing-Technik verwendeten, höherwertigen Bits der Geräte-ID mit dem Inhalt des Geräte-ID-Registers $\mathbf{509_g}$ verglichen werden.

[0355] Eine Übereinstimmung zwischen den ausgewählten (maskierten) Bits der Regions-ID der Nachricht und dem Inhalt des Geräte-ID-Registers $\mathbf{509_g}$ führt zur Auswahl der Ergebnisse des algorithmischen Routers **600** mit dem Multiplexer **598** als möglicher Zieladresse. Wenn zum Beispiel die Regions-ID "abcdef" lautet (wobei a das höherwertige Bit darstellt) und wenn die in den Gerätebitpositions- und -expansionsregistern $\mathbf{509_j}$ und $\mathbf{509_k}$ enthaltenen Werte dergestalt sind, dass die Bits "def" in dem algorithmischen Prozess verwendet werden, dann werden die Bits "abc" der Regions-ID von der Auswahl- und Vergleichsschaltung **601** mit dem Inhalt des Geräte-ID-Registers $\mathbf{509_8}$ verglichen. Wenn umgekehrt die Bits "cdef" für das algorithmische Routing verwendet werden, dann werden nur die Bits "ab" mit dem Inhalt des Geräte-ID-Registers $\mathbf{509_g}$ verglichen.

[0356] Welche Bits der Geräte-ID der Nachricht in den algorithmischen Routingprozess involviert sind und welche nicht, wird auch, wie weiter unten unter Bezugnahme auf **Fig.** 21C besprochen, von den Gerätebitpositions- und -expansionsregistern $\mathbf{509_j}$, $\mathbf{509_k}$ bestimmt.

[0357] Die algorithmische Routing-Logik **600**, deren Funktionsweise weiter unten umfassender beschrieben wird, empfängt die **6** Bit umfassende Geräte-ID sowie Informationen, die von den Gerätebitpositions- und -expansionsregistern $\mathbf{509_j}$, $\mathbf{509_k}$. (in **Fig.** 21A aus Gründen der Klarheit nicht dargestellt, siehe **Fig.** 21C) bereitgestellt werden, um daraus eine 3 Bit umfassende Zielportidentität zu erstellen, die anstelle der Zielportidentitäten ausgewählt werden kann, welche von der Routing-Tabelle **584** bzw. dem Inhalt des Standardregisters $\mathbf{509_d}$ geliefert werden. Die algorithmische Routing-Logik **600** kommt nur in jenen Fällen zum Einsatz, in denen ein Router als Router unterer Ebene konfiguriert ist.

[0358] Die von dem Multiplexer **594** getroffene Auswahl wird an den letzten Multiplexer **599** angelegt, welcher diese Auswahl bzw. den 3 Bit umfassenden Inhalt des Standardportregisters (je nachdem Zustand des Inhalts des Leitweg-zu-Standard-Registers $\mathbf{509_f}$) an die letzte Station des Auswahlprozesses übergibt: die Prüflogik **602**. Die Prüflogik **602** wird betrieben, um den Zustand der Portausgangsdaten zu prüfen, die von dem Produkt

der Zielportauswahlentscheidung angezeigt werden: es sind dies die Ausgangsdaten des Multiplexers **599**. So muss es sich zum Beispiel um eine gültige Zielportangabe handeln (z. B. nicht 6 oder 7). Es werden auch andere Prüfungen vorgenommen, die unter anderem darin bestehen, zu prüfen, dass der angegebene Portausgang für den konkreten Porteingang, der- wie oben geschildert-Zugriff sucht, "freigegeben" sein muss. Diese letztere Prüfung wird dazu verwendet, um gegenüber Fehlern sicher zu sein, welche zu Routing-Schleifen führen können, die in weiterer Folge Systemblockierungszustände verursachen können.

[0359] Die Prüflogik **602** empfängt, wie in **Fig.** 21A gezeigt, den Inhalt des Portfreigaberegisters **562** eines jeden der sechs Portausgänge **502**. Wie bereits erwähnt, bestimmt der Inhalt eines jeden Portfreigaberegisters **562** für jeden Eingangsport **502** zu welchen Ausgangsports **504** eine eingehende Nachricht weitergereicht werden kann und natürlich auch zu welchen nicht. Wenn somit zum Beispiel Port **0** einen Nachrichtenverkehr empfängt, welcher eine Bestimmungsort-ID enthält, die angibt, dass die Nachricht an den Port 3 zu leiten ist, um von dort aus weiterübertragen zu werden, so erstellt die Auswahllogik **560** eine 3-Bit-Menge, durch welche der Port **3** als Zielport angegeben wird, und legt diese Menge an die Prüflogik **602** an. Falls es sich im weiteren herausstellt, dass eine von Port 3 abgehende Übertragung von Nachrichtenverkehr für eingehenden Nachrichtenverkehr, der an Port 0 empfangen wurde, nicht erlaubt ist, so blockiert der Inhalt des Portfreigaberegisters für Port 0 die Übermittlung der Zielportadresse an die Kreuzschienenlogik **500**. Die Nachricht wird statt dessen zu einem nichtexistenten Ausgang der Kreuzschienenlogik **500** geleitet und tatsächlich ausgeschieden, und es wird ein Fehlersignal generiert, durch welches das IP-System **18** benachrichtigt wird.

[0360] Ist der Port 3 jedoch andererseits für Nachrichtenverkehr, der vom Port 0 zu ihm geleitet wird, freigegeben, so übergibt die Prüflogik **602** die von der Auswahllogik **560** erstellte Zielportangabe an die Kreuzschienenlogik **500**, wodurch die Nachricht zu Port 3 geleitet wird.

[0361] Die Prüflogik **602** ist von herkömmlicher Bauart und umfasst zum Beispiel kombinatorische Logik, die entsprechend strukturiert ist, um die durchzuführenden Prüfungen und Entscheidungen in herkömmlicher Weise zu realisieren.

[0362] Ein Grund dafür, dass die konzeptuelle Hierarchie in obere und untere Ebene visualisiert worden ist, liegt unter anderem in dem Bestreben, die Komponentenzahl der Zielport-Auswahllogik und die Grösse der Routing-Tabelle **584** gering zu halten. Und nicht zuletzt aufgrund ebendieser Hierarchie kann ein Router **14** als Router der oberen Ebene bzw. als Router der unteren Ebene bezeichnet werden und kann er in dem einen oder dem anderen der Subprozessorsysteme **10A**, **10B** gelegen sein. Ob ein Router ein Router der oberen Ebene oder ein Router der unteren Ebene ist, hängt von der Konfiguration dieses Routers ab, so wie sie durch die Informationen bestimmt wird, die in seine verschiedenen Konfigurationsregister der Steuerlogik **509** geschrieben worden sind, die ausserdem auch darüber Aufschluss geben, welche Abschnitte der Regions-ID der eingehenden Nachricht dazu verwendet werden, um auf die Routing-Tabelle **584** zuzugreifen.

[0363] Nach Kenntnisnahme dieser Konzepte kann nun zur Betrachtung der **Fig.** 21B übergegangen werden, welche die Entscheidungstafel **604** zeigt, die dazu verwendet wird, die letzte Zielportadresse auszuwählen, die dazu dient, den Leitweg eines eingehenden Nachrichtenpakets durch die Kreuzschienenlogik **500** zu dem richtigen Portausgang auszuwählen. Die Entscheidungstafel **604** veranschaulicht die Entscheidungen, welche auf der Basis der Bestimmungsort-ID (und des Pfadauswahlbits P) des eingehenden Nachrichtenpakets und der Konfiguration des betreffenden Routers (d. h. der in **Fig.** 21A gezeigten Register $\mathbf{509_a}$, ..., $\mathbf{509_h}$) getroffen worden sind.

[0364] Wie **Fig.** 21B zeigt, hat der Inhalt des Leitweg-zum-Standardport-Registers $\mathbf{509_f}$ Vorrang vor allen andern Entscheidungen: ist er entsprechend eingestellt, so dass er den Inhalt des Standardportregisters $\mathbf{509_d}$ auswählt, so werden sämtliche anderen Informationen (Bestimmungsort-ID, Pfadauswahlbit P, der Inhalt der anderen Konfigurationsregister, usw.) überflüssig.

[0365] Wie weiter oben erklärt, ist jeder Router entweder als Router der oberen oder der unteren Ebene konfiguriert. Die Router-Ebene bestimmt, welche Bits der Bestimmungsort-ID verwendet werden, um auf die Routing-Tabelle **584** zuzugreifen und um zu entscheiden, ob das algorithmische Routingverfahren zu verwenden ist. Router der oberen Ebene (als solche angegeben durch den Inhalt des HiLo-Registers $\mathbf{509_0}$ verwenden entweder die Routing-Tabelle, eine Quervernetzungsadresse oder eine Standardadresse. Router der unteren Ebene (das HiLo-Register **509** enthält eine NULL) verwenden tabellenbasierte, Standard-, Quervernetzungs- und algorithmische Routingverfahren.

[0366] Generell werden Router, welche als Router der oberen Ebene konfiguriert sind, dazu verwendet, um Netzwerk-"Wolken" (arbiträre Netzwerke) bestehend aus einer Anzahl von Routern **14** und TNet-Verbindungs-Links L, die eine Anzahl von CPUs **12** und E/A-Geräte **16** untereinander verbinden, wodurch ein Massenparallelverarbeitungssystem (MPP) gebildet wird Es können andere solche MPP-Systeme existieren, und es sind diese als Router der oberen Ebene konfigurierten Router, die in erster Linie dazu verwendet werden, um die Netzwerk-Wolken eines MPP-Systems mit anderen MPP-Systemen zu verbinden.

[0367] Im folgenden wird vorübergehend wieder auf die **Fig.** 19 und 20A Bezug genommen, worin veranschaulicht wird, dass die Bestimmungsort-ID eingehender Nachrichtenpakete, wenn sie von der Eingangslogik **502** eines bestimmten Ports empfangen wird, an das elastische FIFO **506** übermittelt wird, und von dem elas-

tischen FIFO **506** weiter an das Register **570** der Zielport-Auswahllogik **560** (**Fig.** 21A) übermittelt wird, wo es erfasst wird. Sobald die Bestimmungsort-ID des Nachrichtenpakets auf diese Weise erfasst ist, beginnt der Auswahlprozess, der im Hinblick auf die Herausbildung einer Zielportadresse voranschreitet, die dazu verwendet wird, um das Nachrichtenpaket durch die Kreuzschienenlogik zu dem korrekten Ausgangsport zu leiten - vorausgesetzt dieser Ausgangsport ist freigegeben, und zwar sowohl allgemein als auch für den speziellen Eingangsport, der das Nachrichtenpaket empfängt.

[0368] Im folgenden sei unter Bezugnahme auf **Fig.** 21C die algorithmische Routing-Logik **600** im Detail veranschaulicht, welche drei 8-Bit-zu-eins-Multiplexer **620**, **622** und **624** umfasst. Die drei Auswahleingänge (A, B, C) eines jeden der Multiplexer **620**, **622**, **624** empfängt den Inhalt eines 3 Bit umfassenden Gerätepositionsregisters **509**, bei dem es sich um ein anderes der in der Steuerlogik **509** enthaltenen Konfigurationsregister handelt. Die Eingänge (**0**, **1**, ...) eines jeden Multiplexers **620**, **622**, **624** empfangen vorgegebene Bits aus den sechs Bits der Geräte-ID. Der Inhalt des Gerätepositionsregisters **509** steuert die Auswahl jener drei Bits, die als Zielportadresse gemäss der Codierung aus der weiter unten besprochenen Tabelle 4 verwendet werden.

[0369] Die von den Multiplexern **620**, **622**, **624** ausgewählten drei Bits der Geräte-ID werden nicht unmittelbar verwendet. Die ausgewählten Bits werden vielmehr an die kombinatorische Logik angelegt, welche ein ODER-Glied **626** mit zwei Eingängen sowie drei UND-Glieder **628** (**628a**, **628b** und **628c**) umfasst, abhängig von dem Inhalt eines 2 Bit umfassenden Breite-Registers $\mathbf{509}_k$, bei dem es sich um ein anderes der in der Steuer- und Zustandslogik **509** enthaltenen Konfigurationsregister (**Fig.** 21A) handelt. Das Produkt der kombinatorischen Logik ist eine 3 Bit umfassende Zielportangabe. Das Breite-Feld spezifiziert die für die Portauswahl zu verwendende Anzahl an Gerätefeldbits. Ein Wert 0 (Null) in dem Breite-Feld deutet darauf hin, dass alle algorithmisch adressierten Geräte über den Port 0 angeschlossen sind. Ein Wert 3 in dem Breite-Feld deutet darauf hin, dass algorithmisch adressierte Geräte an jedem beliebigen Port angeschlossen werden können.

[0370] Der Inhalt des Gerätefeldbreitenexpansionsregisters $\mathbf{509}_k$ spezifiziert die Bitauswahl, die zu verwenden ist, um die Zielportadresse zu bezeichnen, die an die Kreuzschienenlogik **500** angelegt wird. Die Werte und die Bedeutung der Positions- und Breite-Bits sind nachfolgend in den Tabellen 4 uns 5 dargelegt.

[0371]

TABELLE 4

Gerätebit= position	Verwendete Geräte-ID-Bits
000	2, 1, 0
001	3, 2, 1
010	4, 3, 2
011	5, 4, 3
100	5, 4
101	5
110	NA (Zielport = 0)
111	" " " "

[0372]

TABELLE 5

Gerätefeldbreiten= expansion	Nr. des verwendete ID-Bits
00	0
01	1
10	2
11	3

[0373] Tabelle 5 zeigt welche Bits der Geräte-ID des eingehenden Nachrichtenpakets von einem jeden der Multiplexer **620**, **622**, **624** ausgewählt werden. So werden zum Beispiel für einen (Binär-)Wert 000 in dem Gerätebitpositionsregister **509** von den Multiplexern **620**, **622** und **624** jeweils die Bits **2**, **1** und **0** aus der Geräte-ID des eingehenden Nachrichtenpakets ausgewählt. Umgekehrt werden, wenn der Inhalt- des Gerätebitpositionsregisters **509** den Binärwert **100** aufweist, von den Multiplexern **620** und **622** jeweils nur die Bits **5** und **4** aus der Geräte-ID ausgewählt; für den Ausgang des MUX **624** wird für die verbleibende Bitposition eine NULL erzwungen. Für die (Binär-)Werte **110** und **111** in dem Gerätebitpositionsregister **509** wird von den Multiplexern **620**, **622** und **624** jeweils eine NULL erzwungen und somit der Zielport **0** ausgewählt.

[0374] Die von den Multiplexern **620**, **622** und **624** auf diese Weise ausgewählten Bits werden gemäss dem Inhalt des Gerätefeldbreitenexpansionsregisters $\mathbf{509}_k$ verwendet. Ein Breite-Wert **00** wählt somit, wie in **Fig.** 6 gezeigt, keines der Bits von den Multiplexern **620**, **622** und **624** aus und erzwingt dadurch eine Zielportadresse **000**. Umgekehrt verwendet ein Breite-Wert **10** in dem Gerätefeldbreitenexpansionsregister $\mathbf{509}_k$ zwei der von

den Multiplexern **620**, **622** und **624** ausgewählten Bits.

[0375] Das Ergebnis ist eine 3 Bit umfassende Zielportnummer mit einer in vielen Fällen eingeschränkten Wertemenge. Bewirkt wird dadurch die Generierung eines 3 Bit umfassenden Feldes, beginnend mit der von dem Inhalt des Registers **509** spezifizierten Bitposition mit einer Breite, die von dem Inhalt des Registers $\mathbf{509}_k$ spezifiziert wird.

[0376] Im folgenden wird vorübergehend erneut auf die **Fig.** 19 und 21C Bezug genommen, worin gezeigt wird, dass die Ausgangsports **504** für die Reihung der Nachrichtenpakete verantwortlich sind. Generell erfolgt die Inempfangnahme und die Wegeleitung der Nachrichtenpakete durch einen Portausgang **504** in der Reihenfolge ihres Einlangens. Befindet sich allerdings ein Nachrichtenpaket gerade in Übertragung von einem bestimmten Portausgang, so können mehrere andere Pakete zurückgehalten werden, während sie auf ein Freiwerden dieses Portausgangs warten. Dadurch kann es erforderlich werden, auf Zuweisungsentscheidungsmethoden zurückzugreifen, um zwischen diesen Paketen auszuwählen. Dazu kann eine Reihe herkömmlicher Zuweisungsentscheidungstechniken, wie etwa das Entscheidungsverfahren der zyklischen Zuordnung. Die bevorzugte Zuweisungsentscheidungstechnik ist allerdings jene, die in der ebenfalls schwebenden Patentanmeldung 'Biased Routing Arbitration of Message Traffic in a Communications System' (Beeinflusste Routing-Zuweisungsentscheidung von Nachrichtenverkehr in einem Kommunikationssystem), lfd. Nr. 08/469.756, eingereicht am 6. Juni 1995 und an den Abtretungsempfänger der vorliegenden Anmeldung abgetreten.

[0377] Kurz gesagt beinhaltet jeder Portausgang **504** die autonome Zuweisungsentscheidungslogik **630** (**Fig.** 22). Diese Zuweisungsentscheidungseinheiten **630** nehmen Routing-Anforderungen von einem jeden der Porteingänge **502** entgegen und erteilen die entsprechenden Zuweisungen in einer Reihenfolge, welche auf einer Beeinflussungstechnik basiert, bei welcher jeder Porteingang **502** in Kraft ist und wobei ein Beeinflussungswert vorausgesetzt wird, der einen Anteil an der gesamten Bandbreite jedes beliebigen Portausgangs repräsentiert, den dieser zu benutzen berechtigt ist.

[0378] Gemäss dieser Zuweisungsentscheidungstechnik melden die Porteingänge **502**, welche Nachrichtenverkehr an einen der Portausgänge **504** zu übermitteln haben, ihre Zugriffsanforderungen an. Wollen zwei oder mehrere Porteingänge auf einen Portausgang zugreifen, so trifft der angeforderte Portausgang eine Prioritätsentscheidung zwischen diesen Porteingängen, indem er den Beeinflussungswert eines jeden von ihnen mit denen der anderen vergleicht und einen davon (z. B. den Porteingang mit dem höchsten Beeinflussungswert) auswählt, dem der Zugriff gewährt wird. Der Porteingang (bzw. die Porteingänge), welche bei dieser Zuweisungsentscheidung keine Berücksichtigung gefunden haben, erhalten neue, modifizierte Beeinflussungswerte, um ihre Erfolgschancen bei der folgenden Zuweisungsentscheidung zu erhöhen; der Porteingang **502**, welcher Zugriff erhalten hat, bekommt seinen Beeinflussungswert ebenfalls geändert, jedoch in einer Weise, die seine Chancen für die nächste Zuweisungsentscheidung verringert.

[0379] Im folgenden wird auf **Fig.** 22 Bezug genommen, in welcher ein Blockdiagramm des Portausgangs $\mathbf{504}_n$ veranschaulicht wird, von dem aus Nachrichtenpakete durch den Router **14A** (**Fig.** 19A) übertragen werden. Die Grundkomponenten des Portausgangs **504** und deren Funktionen sind wie folgt:

– die Zuweisungsentscheidungslogik **630**, welche betrieben wird, um Prioritätsentscheidungen zwischen Eingabeports zu treffen, wodurch die Reihenfolge bestimmt wird, in welcher die Pakete von dem Ausgangsport übertragen werden;

– der Befehlssymbolgenerator **632**, der betrieben wird, um (unter Verwendung des Multiplexers **634**) Befehlssymbole zu generieren und in den Symbolstrom einzugeben, die erforderlich sind, um die Regeln des Protokolls aufrechtzuerhalten und zu befolgen. Stellt zum Beispiel der Router **14A** fest, dass er nicht in der Lage ist zu senden, da das empfangende Element belegt ist, so muss der zugeordnete Portausgang **504** "Gegendruck" ausgeben, indem er als Reaktion auf den Empfang eines BELEGT-Befehlssymbols die Übertragung des Nachrichtenpakets stoppt und statt dessen so lange FÜLLEN- oder LEERLAUF-Symbole einfügt bis mit der Übertragung des Nachrichtenpakets fortgefahren werden kann, was durch den Empfang eines BEREIT-Befehlssymbols angezeigt wird. Füllsymbole werden gesendet, wenn der Router ein bereits in Übertragung befindliches Nachrichtenpaket stoppen muss. Befand sich der Router **14A** hingegen im Ruhezustand (kein Nachrichtenpaket in Übertragung), als das BELEGT-Symbol empfangen wurde, so sendet er LEERLAUF-Symbole und verzögert den Beginn der Übertragung eines Nachrichtenpakets so lange, bis von jenem Systemelement, das zuvor ein BELEGT-Befehlssymbol übertragen hatte, ein BEREIT-Symbol empfangen wird. Die FÜLLEN-Symbole werden von dem Befehlssymbolgenerator **632** an den Portausgang **504** geliefert. Das Protokoll erfordert auch, dass die Ausgangslogik einen "Halte"-Mechanismus implementieren: darunter ist die periodische Übertragung von Symbolen (d. h. BELEGT, LEERLAUF, falls kein Nachrichtenpaket zur Übertragung ansteht) zu verstehen, die dazu dienen, das empfangende Element darüber zu informieren, dass der Router **14A** sich nach wie vor im Betriebszustand befindet. Die Art des Symbols, das für diesen Haltemechanismus verwendet wird, hängt von dem gerade ausgeführten Betriebsmodus ab. Während Perioden, in denen kein Nachrichtenverkehr stattfindet, werden zum Beispiel BEREIT-Symbole verwendet und periodisch mit jedem Taktintervall bzw. Zyklus des Sendetaktes, S_Tkt, übertragen. Alternativ dazu werden, wenn ein Portausgang Gegendruck ausgeübt hat, BELEGT-Symbole gesendet. Wird innerhalb eines vorgegebenen Zeitraums (z. B. 10,24 Mikrosekun-

den) überhaupt kein Signal empfangen, so führt dies zu einer Fehlermeldung, die über den OLZP des Routers (oder der Schnittstelleneinheit) zur Bearbeitung an den IP 18 übermittelt wird.

[0380] Es sei hier nur nebenbei erwähnt, dass diese von den Routern **14** beachteten Protokollregeln auch von den CPUs **12** (d. h. den Schnittstelleneinheiten **24**) und den E/A-Paket-Schnittstellen **17** eingehalten werden.
[0381] Und schliesslich ist noch zu erwähnen, dass wenn der Router **14A** in dem System **10** (**Fig.** 1A) direkt mit den CPUs **12A**, **12B** kommunizieren soll, und der Duplexbetrieb verwendet wird, eine Duplexbetriebs-Logikeinheit **638** verwendet wird, um den an eine der CPUs **12A**, **12B** angeschlossenen Portausgang mit dem anderen Portausgang, der ebenfalls an eine der CPUs **12A**, **12B** angeschlossen ist, zu koordinieren.
[0382] Jeder der Portausgänge **504** des Routers **14A** ist für die Reihung der Pakete verantwortlich. Generell werden die Pakete in der Reihenfolge ihres Einlangens weitergeleitet. Wenn jedoch bereits ein Paket in Übertragung befindlich ist, müssen unter Umständen mehrere andere in einem Wartezustand belassen werden. Dabei besteht die Funktion der Zuweisungsentscheidungslogik **630** eines jeden Ausgangsportausgangs **504** darin, Routing-Anforderungen von jeder Routereingangslogik **502** entgegenzunehmen und jedem anfordernden Eingangsport in der richtigen Reihenfolge basierend auf dem Prioritätsschema, das in der oben erwähnten, ebenfalls schwebenden Anmeldung beschrieben ist, Zugriff auf den jeweiligen Ausgangsport zu gewähren. Jede Zuweisungsentscheidungseinheit **630** des Portausgangs **504**, die einen angeforderten Zugriff gewährt, meldet dies an alle anderen Zuweisungsentscheidungseinheiten **630**.
[0383] Von den Taktsynchronisierungs-FIFOs **518** der Eingabelogik werden (im Duplexbetrieb) Paare identischer Symbole empfangen, welche von den beiden CPUs **12** übermittelt werden. Jedes Taktsynchronisierungs-FIFO ist in der Lage, jedes verzögerungsbedingte Auseinanderdriften, das zwischen den von den beiden CPUs kommenden Symbolströmen auftreten kann auszugleichen.
[0384] Wie wohl noch aus der weiter oben erfolgten Abhandlung der CPUs **12** in Erinnerung ist, war in der SpSt ein Online-Zugangspunkt (OLZP) enthalten, welcher dem IC **18** einen Zugriff zum Datenaustausch mit der CPU **12** ermöglichte. Der IP **18** war in der Lage, Befehle in den OLZP **285** zu schreiben, die von den Prozessoren **20** ausgeführt würden, um ein kleines Speicherabbild und eine Routine zu erstellen, die es der CPU **12** erlaubten, eine Boot(Hochfahr-) Operation abzuschliessen. Ein ähnlicher Zugriff wird dem IP **18** durch die Router **14** ermöglicht. Im folgenden wird vorübergehend wieder auf **Fig.** 19A Bezug genommen, in welcher der Router **14A** veranschaulicht wird, der einen OLZP **285'** beinhaltet, der eine Reihe von Konfigurationsregistern enthält, wie zum Beispiel die 'obere' und 'untere Region'-Register **509a**, **509b** (**Fig.** 21A) der Zielport-Auswahllogik, und die Gerätebitpositions- und -expansionsregister **509**$_j$, 509k der algorithmischen Routing-Logik **600** (**Fig.** 21C). Während der Initialisierung des den Router **14A** enthaltenden Subprozessorsystems **10A** werden die in dem OLZP **285'** enthaltenen Konfigurationsregister mit Informationen beschrieben, die von dem IP **18** (über den OLZP-Bus **287'**) bereitgestellt werden, um den Router **14A** mit einer Konfiguration zu versehen, die es ihm erlaubt, in der einen oder anderen Weise betrieben zu werden.
[0385] Jedoch auch der Router **14A** kann Informationen (Fehlermeldungen, etc.) über den OLZP **285'** an den IP **18** übergebn. Zum Beispiel wird der CRC eines jeden von dem Router **14A** geleiteten Nachrichtenpakets, wie weiter oben beschrieben, einer Prüfung unterzogen. Wird dabei von dem Router **14** festgestellt, dass der CRC des Pakets schlecht ist, so wird das Nachrichtenpaket mit einem TPB-Symbol als schlecht gekennzeichnet und der Router meldet darüber hinaus den Fehler dem IP **18** durch Setzen eines in dem OLZP **285'** enthaltenen Fehlerregisters (nicht dargestellt), welches später von dem IP **18** ausgelesen werden kann. Somit ist das System durch dieses Merkmal mit einer Einrichtung zur Meldung eines Übertragungsfehlers ausgestattet.

Taktgabe:

[0386] Sollen die CPUs **12** ordnungsgemäss als synchron gleichlaufende Paare im Duplexbetrieb laufen, so müssen die von ihnen verwendeten Taktsignale klarerweise synchronisiert sein. **Fig.** 24 zeigt den Aufbau der Taktgabeschaltung. In jedem synchron zu haltenden Subprozessorsystem 10A/10B (**Fig.** 1) ist eine Taktgeberschaltung vorhanden. Die generell mit der Bezugszahl **650** versehene Taktgeberschaltung beinhaltet einen Schwingkreis **652**, welcher einen Quarzoszillatorkreis **652a** und einen mit der Bezugszahl **652b** versehenen Teilerzähler im Verhältnis 1 : 8 umfasst. Der Quarzoszillatorkreis **652a** produziert ein periodisches Signal mit einer Frequenz von 12,5 MHz, welches durch **8** dividiert wird, um ein Haupttaktsignal (H_TKT) mit einer Frequenz von 25/16 MHz zu erhalten. Das H_Tkt-Signal wird auch SYNC-TKT genannt. Das H_Tkt-Signal wird an den Taktgenerator **654** angelegt und dazu verwendet, ein Anzahl von 50-MHz-Taktsignalen zu erzeugen, die alle mit dem H_Tkt phasensynchronisiert sind. Diese 50-MHz-Signale werden an die verschiedenen Elemente (z. B. an die CPU **12**, die Router **14**, usw.) des Subprozessorsystems, das die Taktschaltung **650** enthält (z. B. **10A**), verteilt und von diesen verwendet.
[0387] Wie gezeigt, beinhaltet der Taktgenerator **654** einen Phasenvergleicher **660**, der entsprechend geschaltet ist, um das H_TKT-Signal zu empfangen und es mit einem Rückführungs-Taktsignal, einer phasensynchronisierten Kopie seiner selbst zu vergleichen. Die Ausgangsdaten der Phasenvergleichsschaltung **660**,

eine analoge Spannung (V), welche den Phasenunterschied zwischen dem H_TKT- und dem Rückführungs-Taktsignal anzeigt, wird an einen spannungsgesteuerten Quarzoszillator (VCXO) **662** angelegt, um die Synchronisierung der von dem Taktgenerator erzeugten 50-MHz-Signale mit dem H_TKT-Signal sowohl in Bezug auf Phasengang als auch in Bezug auf Frequenz beizubehalten. Falls der Phasenvergleicher **660** einen Phasenunterschied zwischen den H_TKT- und den Rückführungssignalen erkennt, die einen vorgegebenen Phasenbereich überschreitet, so deaktiviert er ein VERRIEGELUNG-Signal, um anzuzeigen, dass die Phasenverriegelung bzw. Phasensynchronisation nicht mehr gegeben ist.

[0388] Der VCXO **6b2** (**Fig.** 24) ist ein spannungsgesteuerter Quarzoszillator mit 100 MHz, der entsprechend konfiguriert ist, um innerhalb von engen Toleranzgrenzen betrieben zu werden. Das Produkt des VCXO **662** wird an einen Synchronzähler angelegt, der die Ausgangsdaten des VCXD **662** um die Hälfte nach unten zählt (durch **2** dividiert), um die 50-MHz-Signale zu erzeugen, und durch **64** dividiert, um eine Kopie des H_Tkt-Signals, das Rückführungssignal, zu erzeugen. Die von dem Zähler **663** erzeugten 50-MHz-Taktsignale werden über das gesamte Subprozessorsystem verteilt und überall bereitgestellt, wo sie benötigt werden.

[0389] Im folgenden wird auf **Fig.** 25 Bezug genommen, in welcher die Zwischenverbindung und die Verwendung zweier Taktsignale **650** veranschaulicht wird, die dazu benutzt werden, synchrone Taktsignale für ein Paar von Subprozessorsystemen **10A**, **10B** (**Fig.** 1) für einen frequenzsynchronisierten Betrieb zu erzeugen. Wie in **Fig.** 25 veranschaulicht, verfügen die CPUs **12A** und **12B** der Subprozessorsysteme **10A**, **10B** über je eine Taktschaltung **650**, die in **Fig.** 25 als Taktschaltungen **650A** und **650B** dargestellt sind und Schwingkreise **652A**, **652B** beinhalten. Es wird jedoch nur der Taktoszillator **652** einer der CPUs **12** verwendet, um daraus das H_TKT-Signal für beide CPUs **12** zu bilden. **Fig.** 25 zeigt den Schwingkreis **625A** der CPU **12A**, der zum Steuern des Taktgenerators **654A** und **654B** beider CPUs **12** verwendet wird. Eine Steuer- und Signalleitung **667** verbindet die beiden Subprozessorsysteme untereinander, um das von dem Schwingkreis **652A** gebildete H_TKT-Signal an den Taktgenerator **654B** des Subprozessorsystems **10B** zu liefern. Zur Fehlereingrenzung und zur Aufrechterhaltung der Signalqualität wird das H_TKT-Signal über ein getrenntes Ansteuerungsglied und über eine Rückübertragungsverbindung **668** an den Taktgenerator **654A** des Subprozessorsystems **10A** geliefert. Der Grund für die Rückübertragungsverbindung **668** besteht darin, ein Verzögerung zwischen dem Schwingkreis **652A** und dem Taktgenerator **654A** zu erzwingen, welche annähernd jener entspricht, die der Taktgenerator **654B** aufgrund der durch die Signalverbindung **667** erzwungenen Verzögerung erfährt.

[0390] Aus Gründen der Klarheit wird in **Fig.** 25 nicht spezifisch dargestellt, dass der Schwingkreis **652** über Treiber und Verbindungen verfügt, welche jenen des Oszillators **652A** spiegelbildlich entsprechen. Durch die Kabelverbindung zwischen den CPUs **12A**, **12B** wird bestimmt, welcher der Schwingkreise **652A**, **652B** der Oszillator ist, der die beiden Taktgeneratoren **654A**, **654B** steuert. Das heisst wenn das Kabel (nicht dargestellt) auf die eine Weise angeschlossen ist, so wird die in **Fig.** 25 gezeigte Verbindung zwischen den Subprozessorsystemen **10A**, **10B** realisiert; wenn es in einer anderen Weise angeschlossen ist, so sind die Verbindungen ähnlich aber es ist nun der Schwingkreis **652B**, der als Oszillator verwendet wird.

[0391] Im folgenden wird mit **Fig.** 25 fortgefahren, wobei das von dem Schwingkreis **652A** des Unterverarbeitungssystems **10A** erzeugte H_TKT-Signal von beiden Unterverarbeitungssystemen **10A**, **10B** als ihr jeweiliges SYNC-TKT-Signal verwendet wird und die verschiedenen anderen Taktsignale aus den 50-MHz-Signalen gebildet werden, die von den Taktgeneratoren **654A**, **654B** erzeugt werden. Dadurch werden die Taktsignale der gepaarten Unterverarbeitungssysteme **10A**, **10B** im Hinblick auf eine frequenzsynchronisierte Funktionsweise, wie sie für den Duplexbetrieb erforderlich ist, synchronisiert.

[0392] Die VCXOs **662** der Taktgeneratoren **654A**, **654B** sind von herkömmlicher Bauart und von einem Typ, welcher eine gewünschte Frequenz selbst dann beibehält, wenn die angelegte, analoge Spannung (V) von dem Phasenvergleicher **660** ausserhalb des Steuergrenzbereichs liegt (was darauf hindeutet, dass die von dem Phasenvergleicher **660** kommenden Taktsignale in beträchtlichem Ausmass phasenverschoben sind. Dadurch wird es möglich, dass beide Taktgeneratoren **654A**, **654B** auch bei nicht ordnungsgemässer Funktionsweise des Schwingkreises **652A** weiterhin Taktsignale an die beiden Unterverarbeitungssysteme **10A**, **10B** liefern, obwohl die Subprozessorsysteme nicht mehr phasensynchronisiert sind.

[0393] Die VERRIEGELUNG-Signale, welche von dem Phasenvergleicher **660** der Taktgeberschaltungen **654A**, **654B** ausgegeben werden (und darauf hindeuten, dass der H_TKT vorhanden ist und mit seiner Kopie, dem Rückführungssignal synchron ist), sind beide an eine Fehlerlogik **670A**, **670B** gekoppelt. Die Ausgabe des VERRIEGELUNG-Signals zeigt an, dass die von dem Taktgenerator **654** erzeugten 50-MHz-Signale, was den Phasengang und die Frequenz betrifft, mit dem H_TKT-Signal synchronisiert sind. Wenn somit eines der VERRIEGELUNG-Signale gleich NULL (d. h. deaktiviert) ist, so ermittelt die Fehlerlogik **670** welcher der Taktgeneratoren sein VERRIEGELUNG-Signal deaktiviert hat und melden dies über den OLZP **285** an den IP 18. Sind beide VERRIEGELUNG-Signale deaktiviert, so können die CPUs davon ausgehen, dass der Schwingkreis **652A**, der die Taktgeneratoren **654A**, **654B** steuert, nicht ordnungsgemäss funktioniert.

Konstantverhältnistaktgabe:

[0394] Wie weiter oben erwähnt, erfolgen Symbolübertragungen zwischen einem Paar von duplexbetriebenen CPUs **12** und Routern **14A**, **14B** (**Fig.** 1) in einem frequenzsynchronisierten Betriebsmodus; das bedeutet, dass das den Symbolstrom begleitende Taktsignal, das dazu verwendet wird, um Symbole in das Taktsynchronisierungs-FIFO des empfangenden Elements (des Routers **14**, oder der CPU **12**) einzugeben, zwar nicht dem Phasengang aber der Frequenz nach im wesentlichen mit jenem des empfangenden Elements identisch ist, das verwendet wird, um Symbole aus dem Taktsynchronisierungs-FIFO zu entnehmen. So ist zum.

[0395] Beispiel in **Fig.** 23, auf die im folgenden Bezug genommen wird und in welcher Symbole veranschaulicht werden, die von einem Router **14A** zu einem Paar von duplexbetrieben CPUs **12A**, **12B** gesendet werden, das Taktsignal, welches seinen Ursprung am Router **14A** nimmt (und den an den CPUs **12A**, **12B** zu empfangenden Symbolstrom als Empfangstakt (Empf-Tkt) begleitet, im wesentlichen, frequenzidentisch mit dem lokalen Takt (Lokaler Tkt). Der erstere (Empf-Tkt) wird verwenden, um Symbole in die Taktsynchronisierungs-FIFOs **126** einer jeden CPU einzugeben, wohingegen der letztere dazu dient, Symbole aus den FIFOs zu entnehmen.

[0396] Diese Technik eignet sich ausgezeichnet für Taktsignale (S Tkt, Empf-Tkt und lokaler Tkt), die dieselbe Frequenz aufweisen, welche darüber hinaus noch der Taktfrequenz entspricht, die für die Kommunikation auf den TNet-Links L verwendet wird. Angenommen jedoch, dass aus Kompatibilitätsgründen mit den elektrischen und sonstigen Eigenschaften des Übertragungsmediums, d. h. der TNet-Links L, die Frequenz des Taktsignals, das dazu verwendet wird, um Symbole über dieses Medium zu übertragen, begrenzt ist, dass aber die empfangende Einheit, in diesem Fall die CPUs **12**, in der Lage ist, mit einem Taktsignal von wesentlich höherer Frequenz betrieben zu werden. In solchen Situationen müssen entsprechende Vorkehrungen getroffen werden, um zu gewährleisten, dass die Synchronisation, mit der Symbole aus den Synchronisierungs-FIFOs **126** einer jeden CPU entnommen werden, zwischen den beiden CPUs aufrechterhalten bleibt.

[0397] In diesem Fall wird ein Konstantverhältnistaktgabemechanismus verwendet, um die Funktion der beiden Taktsynchronisierungs-FIFOs **126** zu steuern, wodurch gewährleistet wird, dass das Taktsignal, welches Symbole aus den beiden FIFOs entnimmt, mit derselben Geschwindigkeit arbeitet, mit der diese in die FIFOs eingegeben werden. In **Fig.** 26A, auf welche im folgenden Bezug genommen wird, wird ein solcher Konstantverhältnistaktgabemechanismus gezeigt und mit der Bezugszahl **700** bezeichnet. Wie in **Fig.** 26A veranschaulicht, beinhaltet der Taktsynchronisierungs-FIFO-Steuermechanismus **700** ein voreinstellbares, mehrstufiges, serielles Schieberegister **702**, dessen parallele Ausgänge an einen Nzu-l-Multiplexer (MUX) **704** angelegt werden. Das serielle Schieberegister **702** wird mit dem schnelleren (höherfrequenten) lokalen Taktsignal (Lokaler Tkt) betrieben, der an den Takteingang (TKT) des Schieberegisters angelegt wird. Ein 15-Bit-Bus **701** überträgt eine Vorgabe (Vgbl) an den Dateneingang (DE), um das serielle Schieberegister **702** voreinzustellen.

[0398] Für einschlägig gebildete Fachleute ist klar, dass die Anzahl der das serielle Schieberegister bildenden Stufen beliebig sein kann und, wie noch zu zeigen sein wird, von dem Verhältnis abhängig ist, welches zwischen dem Taktsignal, mit dem Symbole übermittelt und in die Taktsynchronisierungs-FIFOs **126** eingegeben werden, und der Frequenz des lokal verwendeten Taktsignals herrscht. Im vorliegenden Fall kann davon ausgegangen werden, dass 15 Stufen ausreichen.

[0399] Der MUX **704** wird betrieben, um einen der **15** parallelen Datenausgänge (DA) von dem Schieberegister **702** auszuwählen und an die Eingänge (E) des MUX anzulegen, und zwar als Ausgabedaten des Konstantverhältnistaktsteuermechanismus, der als Lokales Tkt-Signal verwendet wird, um Symbole aus den Taktsynchronisierungs-FIFOs **126** zu entnehmen und den Entnahmezeigerzähler **130** zu betätigen (d. h. zu aktualisieren). Der ausgewählte Ausgang wird auch von dem Ausgang (0) des MUX gekoppelt und an den Rückschalteingang (SI-Eingang) des seriellen Schieberegisters angelegt. Die Auswahl erfolgt durch die Zykluslängenlogik **706**, die durch einen (4-Bit-)Vorgabewert (Vgb,) voreinstellbar, der an den Dateneingang (DE) der Zykluslängenlogik angelegt wird - realisierbar beispielsweise als 4-Bit-Zähler. Der 4-Bit-Ausgangswert der Zykluslängenlogik bildet den Auswahlwert, der an den Auswahleingang (C) des MUX **704** angelegt wird.

[0400] Wesentlich ist, dass die Konstantverhältnistaktsteuerung betrieben wird, um ein Ausgangssignal zu erzeugen, welches über eine vorgegebene Zeitspanne hinweg dieselbe Anzahl an Taktausschlägen beim Empf-Tkt aufweist. Unter der Annahme eines Verhältnisses von N : M (wobei N > M) zwischen dem Taktsignal der CPU **12** und dem Taktsignal, das dazu verwendet wird, um Symbole in das Taktsynchronisierungs-FIFO **126** einzugeben (Empf-Tkt), wird das serielle Schieberegister entsprechend voreingestellt, so dass M Stufen des Schieberegisters einen ersten digitalen Zustand enthalten (z. B. eine EINS), und andere einen anderen digitalen Zustand enthalten (z. B. eine NULL). Die Zykluslängenlogik ist mit einem Wert voreingestellt, welcher Ausgangsdaten des seriellen Schieberegisters auswählt, wodurch praktisch ein abgeschnittenes, serielles Schieberegister mit M Stufen erzeugt wird (dies natürlich nur sofern M nicht gleich 15 ist, in welchem Fall die letzte oder 15. Stufe die Rückkopplung auf die erste Stufe bildet). Ein Beispiel soll dies verdeutlichen.

[0401] Es sei unter vorübergehender Bezugnahme auf **Fig.** 23 angenommen, dass Symbole von dem Router **14A** mit einem 50-MHz-Takt zu den beiden duplexbetriebenen CPUs **12** übertragen werden. Somit werden

Symbole mit einer Geschwindigkeit von 50 MHz in die Taktsynchronisierungs-FIFOs **126** der CPUs eingegeben. Weiterhin sei angenommen, dass die Taktsignale der CPUs bei 40 MHz liegen. Das Verhältnis zwischen lokalem Takt (80 MHz) und dem Empf-Tkt-Signal ist somit 8 : 5. Das serielle Schieberegister ist mit einem Bitmuster voreingestellt, gemäss welchem die am Anfang befindlichen bzw. ersten acht der fünfzehn Stufen **5** EINSEN und 3 NULLEN enthalten. Die Zykluslängenlogik ist mit einem Wert voreingestellt, der die Auswahl der achten Stufe des seriellen Schieberegisters durch den MUX **704** vornimmt. Somit werden dem Schieberegister und der Zykluslängenlogik Werte zugeführt, durch welche praktisch ein serielles Schieberegister mit acht Stufen gebildet wird, in denen drei "Warten"-Zustände und fünf "Aus"-Zustände mit einer Dauer von jeweils 100 ns beinhaltet sind. Demgemäss beinhalten die Ausgabesignale des MUX **704**, welche das Taktsignal erzeugen, das Symbole aus den Taktsynchronisierungs-FIFOs **126** entnimmt, Empf-Tkt, fünf Taktimpulse pro 100-ns-Periode. Somit werden während jeder 100-ns-Periode fünf Symbole in die Taktsynchronisierungs-FIFOs **126** eingegeben und ebenso viele wieder daraus entnommen .

[0402] Dieses Beispiel ist in **Fig.** 26B symbolisch dargestellt, während das in **Fig.** 27 gezeigte Ablaufdiagramm die Funktionsweise der Steuerlogik **700** veranschaulicht. Pro 100-ns-Periode geben fünf Taktimpulse (in **Fig.** 27 mit "EIN" bezeichnet) des Empf-Taktes Symbole in die Taktsynchronisierungs-FIFOs **126** ein. Während derselben 100-ns-Periode verschiebt das serielle Schieberegister **702** eine "01101011"-Sequenz zyklisch durch die von dem MUX **704** ausgewählte Stufe **710**, wodurch ein Lokales Tkt-Signal erzeugt wird, welches dieselbe Anzahl an aktiven Taktimpulsen aufweist wie das Empf-Tkt-Signal.

[0403] Für einschlägig gebildete Fachleute ist klar ersichtlich, dass die Anzahl der Stufen des Schieberegisters **702** entsprechend modifiziert werden kann, um die am häufigsten vorkommenden Taktgeschwindigkeitsunterschiede in Systemen wie dem hier veranschaulichten zu bewältigen. Vorzugsweise hat das Schieberegister **702**, wie erwähnt, 15 Stufen, wodurch es die Fähigkeit hat, einen relativ weiten Taktverhältnisbereich abzudecken. Wie nunmehr ersichtlich ist, kommt es bei dieser Technik der Konstantverhältnistaktgabe niemals zu einer Verschiebung um mehr als einen Takt. Weiterhin ist es eine bessere Implementation als zum Beispiel fünf Takte zu zählen und drei Takte innezuhalten, was zusätzlichen Speicherplatz (d. h. einen Zuwachs bei der Grösse des Synchronisierungs-FIFOs) beanspruchen würde und mehr Latenz verursachen würde.

[0404] Die hier in den **Fig.** 26 vorgestellte Konstantverhältnistaktschaltung wird dazu verwendet, um Datenelemente von einem Taktsystem mit einer bestimmten Frequenz in ein Taktsystem mit einer anderen, d. h. höheren Frequenz zu übertragen. Die Verwendung eines Taktsynchronisierungs-FIFOs ist hier erforderlich, um die Auswirkungen von Signalverzögerungen auszugleichen, die auftreten, während das System in synchronisiertem Duplexbetrieb läuft, damit Paare von identischen Befehls/Datensymbolen von zwei verschiedenen Quellen empfangen werden können. Für einschlägig gebildete Fachleute ist es jedoch klar ersichtlich, dass die hier offenbarte Konstantverhältnistaktschaltung zum Austausch von Daten zwischen zwei beliebigen, verschiedenen Taktsystemen verwendbar ist, solange zumindest zwei Register an der Stelle des Taktsynchronisierungs-FIFOs vorhanden sind. Für das Übertragen von Daten von einem höherfrequenten Taktsystem zu einem niedrigerfrequenten Taktsystem würde die Konstantverhältnistaktschaltung **702** verwendet, um Datenelemente zu der Eingangsstufe bzw. dem Register zu übertragen, welche s) sich unter der Kontrolle des von der Konstantverhältnistaktschaltung **702** gebildeten Taktsignals befinden; das Taktsignal des niedrigerfrequenten Taktsystems würde dazu verwendet, um die Datenelemente zwischen den beiden (oder, wie hier, mehreren) empfangenden Registerstufen zu übertragen und um Datenelemente daraus zu entfernen. Umgekehrt würden Datenelemente, die von einem niedrigerfrequenten Taktsystem in eines mit höherer Frequenz übertragen werden, im wesentlichen wie hier gezeigt ansprechen.

[0405] Dieses Konzept könnte überall dort eingesetzt werden, wo unterschiedliche Taktsignale zum Einsatz kommen. So sind zum Beispiel, wie in der Mikroprozessortechnik wohlbekannt, viele Mikroprozessoren so aufgebaut, dass sie "Warte"-Zustände einfügen, wenn ein Miroprozessor, welcher in Abhängigkeit von Taktsignalen einer bestimmten Frequenz betrieben wird, mit einem synchronen Gerät (z. B. einem Speicher oder einem externen Systembus) kommuniziert, das in Abhängigkeit von einem Taktsignal mit einer unterschiedlichen, üblicherweise niedrigeren Frequenz betrieben wird. Typischerweise ist es bei einer solchen Mikroprozessor/Geräte-Konfiguration erforderlich, dass das langsamere Taktsignal ein ganzes Vielfaches der Taktfrequenz des Mikroprozessors darstellt. Die Konstantverhältnistaktsteuerschaltung **702** könnte einen grossen Bereich möglicher Taktverhältnisse bereitstellen.

E/A-Paket-Schnittstelle:

[0406] Ein jedes der Subprozessorsysteme **10A**, **10B**, etc. verfügt über einen gewissen Grad an Eingabe-Ausgabe-Fähigkeit, welche mit verschiedenen Peripherieeinheiten realisiert ist, obwohl es auch denkbar ist, dass die E/A-Einrichtungen anderer Unterverarbeitungssysteme verfügbar sind, so dass ein Unterverarbeitungssystem nicht unbedingt über lokale E/A-Einrichtungen verfügen muss. Wie dem auch sein, sofern lokale E/A-Einrichtungen vorhanden sind, erfolgt die Kommunikation der Peripheriegeräte und/oder des IP 18 über die E/A-Paket-Schnittstelle 16.

[0407] Die E/A-Paket-Schnittstelle 16 wird betrieben, um die eingehenden Nachrichtenpakete, die sie von einem TNet-Link L empfängt, in eine mit dem angeschlossenen E/A-Gerät besser kompatible bzw. native Form zu übersetzen; andererseits übersetzt die E/A-Paket-Schnittstelle auch in die entgegengesetzte Richtung, wobei sie "native E/A-Daten" (NEA) von den angeschlossenen E/A-Geräten empfängt, Datenbytes in das weiter oben beschriebene '8B–9B'-Format (siehe weiter oben, Tabelle 1) codiert und die erforderlichen Pakete erstellt, um die Daten an den Bestimmungsort zu leiten. Darüber hinaus werden von den E/A-Geräten kommende Interrupts, welche in der für das betreffende E/A-Gerät üblichsten Form (z. B. über Signalleitung) ausgegeben werden, von der E/A-Paket-Schnittstelleneinheit **16** empfangen und dazu verwendet, um daraus ein Interrupt-Paket zu erstellen, das an die CPU **12** gesendet wird, wo es wie weiter oben beschrieben verarbeitet wird. Somit erfolgt die Durchführung von Leseund Schreiboperationen sowie die Ausgabe von Interrupts durch die an den NEA-Bus angeschlossenen Geräte routinemässig über Nachrichtenpakete, wobei Daten/Steuerinformationen in transparenter Weise durch die TNet-Links **L** und die Router **14** zu dem Speicher **28** einer CPU **12** befördert werden.
[0408] Schliesslich beinhaltet die E/A-Paket-Schnittstelle **16** -obwohl die E/A-Paket-Schnittstelle **16** als eines ihrer E/A-Geräte einen IP **18** angeschlossen haben kann – auch einen OLZP, welcher in vieler Hinsicht jenem (OLZP **285**) ähnelt, der in der SpSt **26** (**Fig.** 17) und in dem Router **14** (OLZP **285'**; **Fig.** 19A) enthalten ist, um über den OLZP-Bus Konfigurationsinformationen zu empfangen.

Online-Zugangspunkt:

[0409] Der IP **18** ist mit der Schnittstelleneinheit **24**, der Speichersteuerung (SpSt) **26**, den Routern **14** und den E/A-Paket-Schnittstellen verbunden, wobei die Schnittstellensignale der IEEE-Spezifikation 1149.1 (basierend auf IEEE 1149.1–1990 vom 21. Mai 1990, SH13144, Institute of Electrical and Electronic Engineers, 345 East 47th Street, New York, NY 10017) entsprechen. Der OLZP **259** implementiert diese IEEE-Spezifikation und der Aufbau und die Funktionsweise des OLZP **259** sind im wesentlichen dieselben, ungeachtet dessen mit welchem Element (z. B. einem Router **14**, einer Schnittstelleneinheit **24**, usw.) er verwendet wird. **Fig.** 28 veranschaulicht in Diagrammform den allgemeinen Aufbau des OLZP **259**, der eine Schnittstelle gemäss der IEEE-Spezifikation 1149.1 realisiert. Vorzugsweise ist der OLZP auf jedem IC-Chip vorhanden, der zur Realisierung gewisser hier besprochener Elemente verwendet wird. So ist zum Beispiel jede Schnittstelleneinheit **24**, jede Speichersteuerung **26** und jeder Router **14** mit einer anwendungsspezifischen integrierten Schaltung (ASIC) ausgestattet, die auch einen OLZP beinhaltet, der es dem IP **18** erlaubt, auf die Schaltungsanordnung der ASIC zuzugreifen. So beschreibt etwa die in **Fig.** 28 gezeigte Beschreibung des OLZP **259** den OLZP in Verbindung mit der Schnittstelleneinheit **24**, der SpSt **26** und dem Router **14** des Systems.
[0410] Wie in **Fig.** 28 gezeigt, umfasst der serielle Bus **19A** vier 1-Bit-Signalleitungen: eine Prüftakt-Signalleitung (P_TKT-Leitung), die ein periodisches Taktsignal an den OLAP **295** überträgt; eine Prüfbefehls-Signalleitung (P_BEF-Leitung), um ein Zweizustands-Befehlssignal zu übertragen; eine Testdaten Ein-Signalleitung (TDE-Leitung), auf welcher Daten zu dem OLZP übertragen werden; und eine Testdaten Aus-Signalleitung (TDA-Leitung), auf welcher von dem OLZP abgehende Daten übertragen werden. Diese Signale entsprechend den Anforderungen der Spezifikation IEEE 1149.1. Der OLZP **259** beinhaltet eine 4-Bit-Ablaufsteuereinheit, welche die Funktion des OLZP in Bezug auf die an den P_TKT- und P_BEF-Leitungen des seriellen Busses **19A** empfangenen Takt- und Befehlssignale steuert. Daten (und/oder Befehle), welche von dem OLZP **259** empfangen werden, werden in einem 16-Bit-Befehlsregister (BR), dem Speicher **802** und/oder einem 32-Bit-Datenregister (DR) **104** gespeichert; es können von beiden Registern, dem BR und dem DR Daten übermittelt werden, mit der einen Ausnahme, dass nur das DR **804** mit Daten von der zugeordneten Logik (z. B. einem Router **14**) geladen werden kann.
[0411] Dem OLZP **259** zugeordnet, aber nicht Teil davon, ist ein Kortfigurationsregister **806** in Form einer Registerdatei, welche bis zu 64 32-Bit-Register enthält, auf die sowohl von dem IP 18 (über den OLZP **258**) als auch von den Logikschaltungen aus, denen der OLZP **259** zugeordnet ist, zugegriffen werden kann. Zum Beispiel bilden bestimmte der Register des Konfigurationsregisters **806** die Steuer- und Zustandslogik **509** (**Fig.** 19A) eines Routers **14**. Das Konfigurationsregister **806** wird von dem DR **804** an einen (32-Bit-)Speicherort (d. h. an eine ausgewählte der **64** verfügbaren 32-Bit-Adressen) geschrieben, die von 10-Bit-Befehlen vorgegeben sind, welche zuvor ihrerseits von dem BR 802 bereitgestellt wurden. Die Befehle zum Laden des Konfigurationsregisters **806** beinhalten einen 4-Bit-Abschnitt, welcher von der Befehlsdecodierlogik **810** decodiert wird, und die sich daraus ergebenden, decodierten Daten werden an einen Befehlsgenerator **812** angelegt, der eine Lese- oder eine Schreiboperation identifiziert. Der Gegenstand der Operation, d. h. das eine der **64** Register, das das zu lesende bzw. zu beschreibende Konfigurationsregister **806** bildet, wird durch eine 6-Bit-Adresse identifiziert, die von der Adressendecodierlogik **814** decodiert wird. Der Befehlsgenerator **812** empfängt auch den Zustand der Ablaufsteuereinheit **800**. So wird in Abhängigkeit zu dem von der Ablaufsteuereinheit **800** jeweils eingenommenen, besonderen Zustand und zusammen mit dem von der Befehlsdecodierlogik **810** kommenden, decodierten Befehl von der Befehlsgeneratorlogik **812** ein Schreib- oder Lesebefehl ge-

neriert, um das Konfigurationsregister **806** dazu zu veranlassen, an dem einen der **64** Register, das durch die 6-Bit-Adresse des (von der Adressdecodierlogik **814** decodierten) Befehls bezeichnet wird, eine Lese- oder Schreiboperation vorzunehmen.

[0412] Die von dem IP **18** (**Fig.** 1A) gelieferten Daten werden durch den Multiplexer **816** in das DR **804** geschrieben. Die den OLZP **259** verwendende Logik kann das DR **804** von zwei getrennten Quellen aus beschreiben, wobei an diesen Quellen 32-Bit-Register bereitgestellt werden, die von dem IP **18** wahlweise mit dem DR **804** gekoppelt und unter Verwendung von bereits zuvor in das BR **802** geschriebenen Befehlsinformationen in dieses geschrieben werden, und zwar unter Zuhilfenahme der Ablaufsteuereinheit **800** durch Signalübertragung über die P_TKT- und P_BEF-Signalleitungen des seriellen Busses **19A**. Der Zugriff auf das 32-Bit-DR **804** erfolgt durch schrittweises Positionieren der Ablaufsteuereinheit **800** durch die in der Spezifikation 1149.1 als "CAPTURE-DR," "SHIFT-DR," und "UPDATE-DR" beschriebenen Zustände unter Verwendung eines geeigneten Befehls aus 1149.1. In dem Befehl enthaltene, zusätzliche Bits ermöglichen es dem DR **804**, anhand des CAPTURE-DR-Zustands ausgewählte Werte chip-interne Werte, welche die Chip-Statusinformation enthalten, zu lesen. Andere Befehle aus 1149.1 erlauben es dem UPDATE-DR-Zustand, den Registerinhalt zu Konfigurations- und Initialisierungszwecken an ausgewählte Register zu kopieren. Die Inhalte des DR **804** werden unter Verwendung des SHIFT-DR-Zustands aus 1149.1 (über den seriellen Bus **19A**) mit dem IP **18** ausgetauscht. Weitere Informationen über Aufbau und Funktion des OLZP können in der Spezifikation IEEE **1149.1** (IEEE 1149.1–1990 vom 21. Mai 1990, SH13144) nachgelesen werden.

Asymmetrische Variablen:

[0413] Unter "asymmetrischen Variablen" sind Werte zu verstehen, welche sich in einer CPU eines Paars von CPUs **12** von jenen der anderen CPU unterscheiden bzw. unterscheiden können. Beispiele für asymmetrische Variablen sind unter anderem eine Seriennummer, die zugewiesen wurde und an einem von der CPU lesbaren Speicherort, zum Beispiel einem Register ausserhalb des Speichers **28**, aufbewahrt wird und die sich von jener der anderen CPU unterscheidet, oder ein Inhalt eines Registers, der verwendet wird, um das Vorkommen von korrigierbaren Speicher- oder Cachespeicherfehlern zu verfolgen (unter der Voraussetzung, dass das Erkennen, das Korrigieren und das Melden des Fehlers nicht dazu führt, dass die duplexbetriebenen CPUs von ihrem synchron im Gleichschritt laufenden Betrieb abgebracht werden).

[0414] Im Duplex-Betriebsmodus ist die sorgfältige Behandlung von asymmetrischen Variablen von grosser Bedeutung, um zu gewährleisten, dass (in dem Speicher **28** einer jeden CPU **12**) mehrfach vorhandene Kopien des Systemspeichers, welche logisch äquivalent sein sollen, zu jeder Zeit identische Daten enthalten. Würde eine asymmetrische Variable einfach von jeder der beiden duplexbetriebenen CPUs gelesen und anschliessend in den Speicher geschrieben, so würden sich die Inhalte des Speichers **28** einer jeden CPU dadurch um mindestens diesen von jeder CPU gelesenen Wert unterscheiden.

[0415] Um einem Paar von CPUs **12**, welche im Duplexmodus betrieben werden, die Verarbeitung asymmetrischer Variablen zu ermöglichen, wird einer jeden Schnittstelleneinheit **24** einer jeden CPU **12** ein "Soft-Vote"-Logikelement **900** (SV-Logikelement) zur Seite gestellt. Wie in **Fig.** 30 veranschaulicht, sind die SV-Logikelemente **900** einer jeden Schnittstelleneinheit **24** untereinander über einen 2-Bit-SV-Bus **902** verbunden, welcher die Busleitungen **902a** und **902b** umfasst. Die Busleitungen **902a** übertragen Ein-Bit-Werte von den Schnittstelleneinheiten **24** der CPU **12A** zu jenen der CPU **12B**. Umgekehrt überträgt die Busleitung **902b** Ein-Bit-Werte von den SV-Logikelementen **900** der CPU **12B** zu jenen der CPU **12A**.

[0416] In **Fig.** 30B wird das SV-Logikelement **900a** der Schnittstelleneinheit **24a** der CPU **12A** veranschaulicht. Jedes SV-Logikelement **900** ist was Aufbau und Funktionsweise betrifft im wesentlichen identisch mit jedem anderen SV-Logikelement **900**, so dass die Beschreibung des Logikelements **900a** auch auf die anderen Logikelemente **900a** (der Schnittstelleneinheit **24b**, CPU **12A**) und **900b** (der Schnittstelleneinheiten **24a**, **24b** der CPU **12B**) anwendbar zu verstehen ist, sofern keine anderslautenden Angaben gemacht werden. Wie in **Fig.** 30B veranschaulicht, beinhaltet das SV-Logikelement **900a** vier Ein-Bit-Register: ein Ausgaberegister **904**, ein Lokaleingaberegister **906**, ein Ferneingaberegister **907** und ein Ausgabedatenfreigaberegister **912**. Das Ausgaberegister **904** ist über einen Multiplexer (MUX) **914** und einen Treiber **916** mit Dreizustandsverhalten an die gemeinsam genutzte Busleitung **902a** gekoppelt. Nur die Logikelemente **900a** der CPU **12A** steuern die Busleitung **902a**, und immer nur eines der beiden Logikelemente übernimmt die Steuerung. Welches das ist, hängt von dem Inhalt des Freigaberegisters **912** ab. Die Logikelemente **900b** der CPU **12B** können auf der Busleitung **902a** nur empfangen. Die Busleitung **902a** übermittelt daher die Ausgaberegister **904** der Logikelemente **900a** an die Ferneingaberegister **907** eines jeden der Logikelemente **900b** der CPU **12B**. Die Busleitung **902a** übermittelt ausserdem die Ausgaberegister **904** (über den Multiplexer **914** und den Treiber **916**) eines der Logikelemente **900a** an die Lokaleingaberegister des anderen der Logikelemente **900a** (sowie auch an ihr eigenes). Auf diese Weise sind die beiden Schnittstelleneinheiten **24a**, **24b** der CPU **12A** in der Lage, asymmetrische Variablen untereinander auszutauschen.

[0417] Auf ähnliche Weise werden die Ausgaberegister **904** der Logikelemente **900v** der CPU **12B** über die

Busleitung **902b** an das entfernte Register **907** des Logikelements **902a** (und an jenes der anderen Schnittstelleneinheit **24b**) übermittelt.

[0418] Die Logikeinheiten **902** bilden einen Teil der Konfigurationsregister **74** (**Fig.** 5). Somit können sie von der (den) Prozessoreinheit en) 20 beschrieben werden, indem diese die nötigen Daten-/Adressinformationen über zumindest einen Teil des Adress-/Datenbusses **74** (in **Fig.** 30B als Bus **74'** dargestellt) übermitteln, um das Ausgaberegister **904** und/oder das Freigaberegister **912** auszuwählen und zu beschreiben, oder um das lokale und das entfernte Register **906** und **907** zu lesen.

[0419] Der MUX **914** wird betrieben, um jeder Schnittstelleneinheit **24** der CPU **12A** die selektive Benutzung der Busleitung **902a** für die SV-Logikelemente **900a** zu ermöglichen, oder um ein BUSFEHLER-Signal zu übermitteln, falls während des (weiter unten beschriebenen) Reintegrationsprozesses, mithilfe dessen ein Paar von CPUs **12** in im Gleichschritt laufenden Duplexbetrieb gebracht wird, ein Busfehler auftritt. Das Ausgabedatenfreigaberegister wird mit einem Bit beschrieben, welches den Treiber mit Dreizustandsverhalten aktiviert (bzw. deaktiviert), so dass dieser die Busleitung **902a** gemäss dem Inhalt des SV-Ausgaberegisters **904** steuert.

[0420] Wie weiter oben erwähnt, erlauben es die SV-Logikelemente **900** den CPUs **12a**, **12B**, im Duplexbetrieb einen bitweisen Austausch von asymmetrischen Variablen durchzuführen. Es sei hier nochmals daran erinnert, dass beim Duplexbetrieb der CPUs **12A**, **12B** beide CPUs dieselben Befehle identischer Befehlsströme im wesentlichen zum selben virtuellen Zeitpunkt, wenn nicht zum selben Zeitpunkt ausführen. Der Austausch asymmetrischer Variablen zwischen ihnen erfolgt in der nachfolgend beschriebenen Weise. Beide CPUs beschreiben in Reaktion auf den Befehlsstrom und im wesentlichen zur selben Zeit die Freigaberegister **912** des Logikelements **900** beider Schnittstelleneinheiten **24** einer jeden CPU. Eines der beiden Logikelemente **900** einer jeden CPU wird mit einer Zustandsinformation beschrieben, durch welche der zugeordnete Treiber **916** freigegeben wird; das andere wird mit einem Zustand beschrieben, der den Ausgang des Treibers in einen hochohmigen Zustand versetzt.

[0421] Beschrieben werden annahmegemäss jene Ausgabedatenfreigaberegister **912**, welche den Logikelementen **900** der Schnittstelleneinheiten **24a** beider CPUs **12A**, **12B** zugeordnet sind, um die zugeordneten Treiber freizugeben. Somit werden die Ausgaberegister **904** der Schnittstelleneinheiten **24a** einer jeden CPU an die Busleitungen **902** übermittelt; das heisst das dem Logikelement **900a** der Schnittstelleneinheit **24a** (CPU **12A**) zugeordnete Ausgaberegister **904** wird an die Busleitung **902a** übermittelt, während das dem Logikelement **900b** der Schnittstelleneinheit **24a** der CPU **12B** zugeordnete Ausgaberegister an die Busleitung **902b** übermittelt wird. Die CPUs **12** schreiben beide ein Bit der asymmetrischen Variable in ihre jeweiligen Ausgaberegister **904**, gefolgt von einer Leseoperation der einem jeden von diesen zugeordneten Ferneingaberegister **907**, und zwar unter Einberechnung eines grösstmöglichen Taktimpulsabstandes. Die Ausgaberegister **904** werden hierauf erneut von jeder CPU beschrieben, worauf eine neuerliche Leseoperation der Ferneingaberegister **907** erfolgt. Dieser Prozess wird Bit für Bit fortgesetzt bis die Variable in ihrer Gesamtheit von dem Ausgaberegister **904** einer jeden CPU **12** zudem Ferneingaberegister der jeweils anderen übertragen worden ist. Es gilt zu beachten, dass beide Schnittstelleneinheiten **24** der CPU **12B** das asymmetrische Informationsbit empfangen.

[0422] Ein Anwendungsbeispiel des Soft-Vote-Mechanismus ist der Austausch von Seriennummern. Eines der Konfigurationsregister **74** ist ein Ein-Bit-Register (nicht dargestellt), welches bei Inbetriebnahme gesetzt wird, um eine jede der beiden CPUs, die miteinander in Duplexbetrieb gebracht werden können, zu identifizieren und um sie voneinander zu unterscheiden. Daher wird das Ein-Bit-Register der einen CPU in einen Zustand gesetzt, der sich von jenem der anderen CPU unterscheidet. Dasselbe geschieht, nach wie vor während der Inbetriebnahme, mit anderen Konfigurationsregistern, in welche die Seriennummer der betreffenden CPU zu laden ist. Welches der Konfigurationsregister für Seriennummern geladen wird, hängt vom Zustand des Ein-Bit-Identitätsregisters ab. Somit haben die beiden CPUs jeweils zwei identische Register, die Ihre Seriennummern enthalten, mit der einen Ausnahme, dass ein Register, nachfolgend "R1" genannt, (nicht dargestellt) in der einen CPU die eigene Seriennummer enthält, während die andere CPU ihre Seriennummer in einem Konfigurationsregister "R2" (nicht dargestellt) enthält. Bevor diese Werte von den duplexbetriebenen CPUs in den Speicher geschrieben werden können, müssen die Konfigurationsregister R1 unter Verwendung des Soft-Vote-Mechanismus miteinander "harmonisiert" werden.

[0423] Die SV-Logikelemente **900** werden auch dazu verwendet, um Busfehler zu übermitteln, zu denen es während des noch zu beschreibenden Reintegrationsprozesses kommen kann. Während der Durchführung der Reintegration wird ein REINT-Signal ausgegeben. Wie in **Fig.** 30B gezeigt, wird das REINT-Signal an den Steuereingang (St) des MUX **914** angelegt. Wenn daher REINT ausgegeben wird, wird von dem MUX **914** ein BUSFEHLER-Signal ausgewählt und an die Busleitung **902a** übermittelt.

Synchronisierung:

[0424] Um einen ordnungsgemässen Betrieb des Unterverarbeitungssystems **10A**, **10B** (**Fig.** 1A, 2), sei es unabhängig voneinander (Simplexbetrieb) oder gepaart und synchron im Gleichschritt miteinander laufend

(Duplexbetrieb) zu gewährleisten, ist es erforderlich sicherzustellen, dass zwischen den CPUs **12A**, **12B** und den Routern **14A**, **14B** ausgetauschte Daten ordnungsgemäss empfangen werden und dass ein ursprünglicher Inhalt der Taktsynchronisierungs-FIFOs **102** (der CPUs **12A**, **12B**; **Fig.** 5) und **519** (der Router **14A**, **14B**; **Fig.** 20A) nicht fälschlicherweise als Daten bzw. als Befehle interpretiert wird. Die Eingabe- und Entnahmezeiger der verschiedenen Taktsynchronisierungs-FIFOs **102** (in den CPUs **12**) und 518 (in den Routern **14**) müssen initialisiert werden, um zumindest einen frequenznahen Betrieb zu gewährleisten. Normalerweise erfolgt dies in der herkömmlichen Art durch ein Einschaltsignal (nicht dargestellt) wenn zu Beginn der Strom zugeschaltet wird, wodurch der Eingabe- und der Entnahme-Zeigerzähler um eine bestimmte Solldistanz voneinander beabstandet werden und die zugeordneten FIFO-Warteschlangen in einen bekannten Zustand voreingestellt werden. Nachdem dies erfolgt ist, sind alle Taktsynchronisierungs-FIFOs für einen frequenznahen Betrieb initialisiert. Wenn somit das System **10** zu Beginn in Online-Betrieb gesetzt (d. h. eingeschaltet) wird, befinden sich die Kommunikationsverbindungen zwischen den CPUs **12A**, **12B** und den Routern **14A**, **14B** in frequenznahem Betriebsmodus.

[0425] Die Anforderungen sind jedoch höher, wenn die CPUs **12A**, **12B** in den Duplex-Betriebsmodus umgeschaltet werden. Als erstes muss die zur Datenübermittlung zwischen den CPUs **12A**, **12B** und den Routern **14A**, **14B** verwendete Taktgabe auf jedem TNet-Zink auf frequenzsynchronisierten Betrieb umgeschaltet werden. Danach müssen im Hinblick auf eine ordnungsgemässe Implementierung des für den Duplex-Betriebsmodus erforderlichen Gleichschrittbetriebs die Taktsynchronisierungs-FIFOs entsprechend synchronisiert und auf das jeweilige Ursprungselement abgestimmt werden, von welchem sie Daten empfangen, um etwa eine Verzögerung berücksichtigen zu können, die in einem Pfad, nicht jedoch in einem anderen Pfad auftritt. Es sei zum Beispiel daran erinnert, dass es der Betrieb im Duplexmodus erfordert, dass gepaarte CPUs **12** jeden Befehl eines identischen Befehlsstroms zur selben virtuellen Zeit ausführen. (Mit "virtueller" Zeit ist gemeint, dass obwohl die tatsächliche Echtzeitausführung eines identischen Befehls durch gepaarte CPUs **12** um einen geringfügigen Betrag abweichen kann, deren Aktionen von aussen betrachtet exakt dieselben sind.) Eingehende, von den Routern **14A**, **14B** kommende Daten müssen in einer frequenzsynchronisierten Betriebsumgebung von den beiden CPUs nahezu gleichzeitig empfangen werden. Jede Verzögerung, zu der es in dem Kommunikationspfad zwischen dem einen oder anderen der Router **14A**, **14B** und den CPUs **12A**, **12B** kommen kann, muss berücksichtigt werden. Die Taktsynchronisierungs-FIFOs **102** der gepaarten CPUs **12** sind dafür zuständig, die Nachrichtenpaketsymbole zu empfangen, allfällige im Kommunikationspfad auftretende Verzögerungen auszugleichen und den beiden CPUs die Symbole unter Wahrung der Gleichzeitigkeit vorzulegen, um so den synchronisierten Gleichlauf aufrechtzuerhalten, welcher für den Duplex-Betriebsmodus erforderlich ist.

[0426] In ähnlicher Weise muss jedes Symbol, das von einer der CPUs **12** kommend von den Routern **14A**, **14B** empfangen wird, mit dem von der jeweils anderen CPU kommenden Symbol verglichen werden und (wie weiter unten im genauer abgehandelt) auf mögliche Abweichungen zwischen den CPUs hin geprüft werden. Auch hier sind es wieder die Taktsynchronisierungs-FIFOs **518** der Router **14A**, **14B**, welche Nachrichtenpakete von den CPUs **12** empfangen und in den Kommunikationspfaden auftretende Verzögerungen ausgleichen, so dass die von den beiden CPUs **12** kommend empfangenen Symbole gleichzeitig aus den Taktsynchronisierungs-FIFOs abgerufen werden

[0427] Bevor nun besprochen wird, wie die Taktsynchronisierungs-FIFOs der CPUs und der Router rückgesetzt, initialisiert und synchronisiert werden, wird es als hilfreich erachtet, sich ein besseres Verständnis davon zu verschaffen, wie sie betrieben werden, um den synchronisierten Gleichlauf beim Duplexbetrieb aufrechtzuerhalten. Im folgenden wird daher vorübergehend au **Fig.** 23 Bezug genommen, in welcher die Taktsynchronisierungs-FIFOs **102** der CPUs **12A**, **12B** gezeigt werden, welche Daten empfangen, die zum Beispiel von dem Router **14A** kommen können. **Fig.** 23 veranschaulicht die Portausgänge $\mathbf{504_4}$ und $\mathbf{504_5}$ des Routers **14A**, die an die 10-Bit-Busse $\mathbf{32_x}$; und $\mathbf{32_y}$ angeschlossen sind, welche ihrerseits jeweils Daten-/Befehlssymbole und Takt von dem Router **14A** an die beiden duplexbetriebenen CPUs **12A**, **12B** koppeln. Zur Erinnerung sei hier nochmals darauf hingewiesen, dass ein Nachrichtenpaket zwar eine einzige Bestimmungsadresse haben kann, welche eine CPU **12** bezeichnet, dass dieses Paket im Duplexbetrieb jedoch von dem Router **14A** symbolweise dupliziert und im wesentlichen zeitgleich an beide CPUs **12A** und **12B** übertragen wird.

[0428] Unter Umständen können die beiden CPUs **12A**, **12B** so angeordnet sein, dass ein Symbol, das vom Router **14A** kommend von einer der beiden CPUs (z. B. der CPU **12B**) empfangen wird, eine Verzögerung von unbekanntem (jedoch maximalbegrenztem) Ausmass in Bezug auf den Empfang des identischen (von dem Router duplizierten) Symbols durch die andere CPU (CPU **12A**) erfährt. Diese Verzögerung ist in dem Bus **32**, der das Symbol und den dazugehörigen Sendetakt, S_Tkt, von dem Router **14A** zu der CPU **12B** überträgt, mit der Bezugszahl **640** gekennzeichnet.

[0429] Nachfolgend sei der Betrieb der Taktsynchronisierungs-FIFOs $\mathbf{102_x}$, $\mathbf{102_y}$ im Zusammenhang mit dem Empfang identischer Daten im Zuge des Duplexbetriebs betrachtet. Die nachfolgende Tabelle 6 veranschaulicht diesen Betrieb. Der Einfachheit halber wird in Tabelle 6 davon ausgegangen, dass die Verzögerung **640** nicht mehr als ein Taktintervall des Sendetaktes (S_Tkt) beträgt. Falls jedoch die Verzögerung **640** grösser ist als eine Taktzeit des Sendetaktes, S_Tkt, so muss die Tiefe der Warteschlange **126** entsprechend vergrössert

werden, um eine grössere Distanz zwischen dem Inhalt des Eingabe- und des Entnahme-Zeigerzählers **128** und **130** unterstützen zu können. Ist zum Beispiel die Verzögerung **640** so gross, dass das Eintreffen eines Symbols bei der CPU **12B** um ganze drei S_Tkt-Intervalle später erfolgt als das Eintreffen desselben Symbols bei der CPU **12A**, so sollte der Abstand zwischen dem Eingabe- und dem Entnahme-Zeigerzähler zumindest vier betragen. Somit ist die Tiefe der Warteschlange **126** in einem solchen Fall sechs Symbolspeicherplätze oder grösser.

TABELLE 6

Punkt	RST	Tkt 1	Tkt 2	Tkt 3	Tkt 4	Tkt 5	Tkt 6	Tkt 7
CPU 12A-Werte								
Eingabe-Zgr	0	1	2	3	0	1	2	3
Entnahme-Zgr	2	3	0	1	2	3	0	1
Byte 0	LEERLAUF		A	A	A	E	E	E
Byte 1	LEERLAUF	LEERLAUF	B	B	B	B	F	F
Byte 2	LEERLAUF	LEERLAUF	LEERLAUF	C	C	C	C	G
Byte 3	LEER-LAUF	LEER-LAUF	LEER-LAUF	LEER-LAUF	D	D	D	D
AUS-Reg	LEER-LAUF	LEER-LAUF	LEER-LAUF	A	B	C	D	E
CPU 12B-Werte								
Eingabe-Zgr	0	0	1	2	3	0	1	2
Entnahme-Zgr	2	3	0	1	2	3	0	1
Byte 0	LEER-LAUF	LEER-LAUF	A	A	A	A	E	E

Byte 1	LEER-LAUF	LEER-LAUF	LEER-LAUF	B	B	B	B	F
Byte 2	LEER-LAUF	LEER-LAUF	LEER-LAUF	LEER-LAUF	C	C	C	C
Byte 3	LEER-LAUF	LEER-LAUF	LEER-LAUF	LEER-LAUF	LEER-LAUF	D	D	D
AUS-Reg	LEER-LAUF	LEER-LAUF	LEER-LAUF	A	B	C	D	E

[0430] Die Reihen der oberen Hälfte der Tabelle 6 zeigen die von den Entnahme-Zeigerzählern **128**, **130** für die CPU **12A** (Schnittstelleneinheit **24A**) angenommenen Werte, sowie den Inhalt eines jeden der vier Speicherorte (Byte 0. ..., Byte **3**) der Warteschlange **126**, und den Inhalt des Ausgaberegisters **132** während einer anfänglichen Reset-Periode (RST) und während der darauffolgenden Taktintervalle des Sendetaktes (S_Tkt). Die Reihen der unteren Hälfte der Tabelle 6 zeigen dasselbe für das FIFO **102**$_y$ der Schnittstelleneinheit **24a** der CPU **12B**, und zwar für jedes Symbol des duplizierten Symbolstroms.

[0431] Angenommen, die Verzögerung **640** ist nicht grösser als ein S_Tkt-Intervall, so zeigen die (in den Zählern **128**, **130** jeweils auf dem aktuellen Stand gehaltenen) Eingabeund Entnahmezeiger auf Speicherorte in der Warteschlange **126**, welche zwei Speicherorte voneinander entfernt sind. Die Eingabe-Zeigerzähler **128** zeigen jeweils auf den nächsten Speicherort in der Warteschlange **126** an welchem ein empfangenes Symbol gespeichert wird, und die Entnahme-Zeigerzähler **130** zeigen auf einen Ort, von welchem ein Symbol aus der Warteschlange entnommen wird.

[0432] Unter Bezugnahme auf Tabelle 6 und **Fig.** 23 sei nun angenommen, dass der Router **14A** gemäss dem Protokoll wonach er zuvor einen Strom von "LEERLAUF"-Symbolen übertrug, dazu übergeht, einen Symbolstrom (Nachrichtenpaket) zu senden und dabei mit dem Symbol A beginnt. Wie in Tabelle 6 veranschaulicht, trifft das Symbol A aufgrund der Verzögerung **640** bei der CPU **12B** um einen Zyklus später ein als bei der CPU **12A**. Hier ist jedoch zu beachten, dass der Inhalt des Eingabe-Zeigerzählers **128** für die CPU **128** ebenfalls um einen Zyklus verzögert auf jenen der CPU **12A** folgt. Obwohl also das Symbol A bei der CPU **12B** um einen Zyklus später eintrifft als bei der CPU **12A**, werden daher dennoch beide an "Byte **0"**-Speicherorten der Warteschlangen **126** gespeichert. Dies ist deshalb möglich, weil (1) die FIFOs **102** im Hinblick auf einen Synchronbetrieb synchronisiert worden sind (ein weiter unten genauer beschriebener Prozess), und (2) die Eingabe-Zeigerzähler **128** von dem Taktsignal getaktet werden, das am Herkunftsort der Symbole erzeugt wird, d. h. von dem vom Router **14A** kommenden S_Tkt, und dieses Taktsignal derselben Verzögerung **640** unterliegt wie die Symbole. Die Entnahme-Zeigerzähler **130** stimmen andererseits stets miteinander überein, weil sie durch die lokalen Empfangstakte (Empf-Tkt) getaktet werden, die von den Paketempfängern **94** der CPUs **12** erzeugt werden Diese lokalen Empfangstakte sind im Duplexbetrieb darüber hinaus frequenz- und phasensynchronisiert; sie unterliegen somit keinerlei Verzögerung.

[0433] Eine andere Art, die Verzögerung **640** zu betrachten besteht darin, sie sich als Teil einer Pipeline innerhalb des Kommunikationspfads (Bus **32y**) zwischen dem Router **14A** und der CPU **12B** vorzustellen. Die Verzögerung **640** kann jeden beliebigen Wert annehmen, solange auch durch die maximale Verzögerung gewährleistet bleibt, dass die Symbole noch rechtzeitig in die Speicherwarteschlange **126** eingegeben werden können, und zwar zumindest einen Taktzyklus bevor das betreffende Symbol wieder aus der Warteschlange entnommen wird. Die an die CPU **12A** übermittelten Symbole warten nämlich einen zusätzlichen Zyklus lang, bevor sie aus der Warteschlange **126** entnommen werden, und zwar zur selben Zeit, zu der ihre jeweilige Kopie aus der Warteschlange **126** der CPU **12B** entnommen wird. Auf diese Weise wird jedes Symbol des von dem Router **14A** übertragenen Symbolstroms jeweils gleichzeitig aus dem Taktsynchronisierungs-FIFO **102** der CPUs **12A**, **12B** entnommen, und zwar unter Beibehaltung der für den Duplex-Betriebsmodus erforderlichen Synchronisierung der empfangenen Daten. Die Tiefen der Warteschlangen der TS-FIFOs **102** passen sich nämlich an, um die beiden von dem Router **14A** zu den CPUs **12A**, **12B** führenden Pfade mit derselben Verzögerung auszustatten.

[0434] Um die soeben unter Bezugnahme auf Tabelle 6 beschriebene Operation zustandezubringen, wird der in **Fig.** 31A gezeigte Reset- und Synchronisierungsprozess verwendet. Der Prozess initialisiert nicht nur die Taktsynchronisierungs-FIFOs **102** der CPUs **12A**, **12B** für den Duplex-Betriebsmodus, sondern dient auch dazu, die Taktsynchronisierungs-FIFOs **518** (**Fig.** 19A) der CPU-Ports eines jeden der Router **14A**, **14B** für den Duplexbetrieb zu initialisieren. Der Reset- und Synchronisierungsprozess verwendet das SYNC-Befehlssym-

bol, um eine Zeitdauer zu initiieren, die von dem SYNC-Tkt-Signal **970** (**Fig.** 31B) begrenzt ist, um die entsprechenden Taktsynchronisierungs-FIFOs der CPUs **12A**, **12B** und der Router **14A**, **14B** rückzusetzen und zu initialisieren. (Das SYNC-TKT-Signal wird von dem Taktgenerator **654** (**Fig.** 24) zum Zweck der Verteilung an die Elemente des Systems **10**, insbesondere die Router **14A**, **14B** und die CPUs **12A**, **12B** erzeugt. Es weist i eine niedrigere Frequenz auf als jenes, das verwendet wird, um Symbole durch die Taktsynchronisierungs-FIFOs zu empfangen, S_Tkt. Wenn S_Tkt zum Beispiel ungefähr 50 MHz beträgt, liegt das SYNC-TKT-Signal bei etwa 3,125 MHz.) Der Reset- und Initialisierungsprozess beginnt, wie in **Fig.** 31A gezeigt, bei Schritt 950 damit, dass die Taktsignale, welche von den CPUs **12A**, **12B** und den Routern **14A**, **14B** als Sendetakt (S_Tkt) verwendet werden, und die lokalen Taktsignale (Lokaler Tkt) der Einheit entsprechend geschaltet werden, so dass sie von demselben Taktsignal abgeleitet werden. Die S_Tkt-Signale und die Lokalen Tkt-Signale haben im wesentlichen dieselbe Frequenz, wenn auch infolge von bei der Übermittlung der verschiedenen Taktsignale inhärenten Verzögerungen nicht notwendigerweise dieselbe Phase. Ausserdem werden die Konfigurationsregister in den CPUs **12A**, **12B** (die Konfigurationsregister **74** in den Schnittstelleneinheiten **24**) und in den Routern **14A**, **14B** (in der Steuerlogikeinheit **509** der Router **14A**, **14B** enthalten) in den frequenzsynchronisierten (FregSync) Zustand versetzt.

[0435] Die folgende Abhandlung betrifft den Schritt 952 und nimmt Bezug auf die Schnittstelleneinheit **24** (**Fig.** 5), den Router **14A** (**Fig.** 19A), sowie auf die **Fig.** 31A und **31B**. Bei frequenzsynchronisiertem Taktbetrieb sendet die CPU **12A** ein Nachrichtenpaket an die im Offline-Zustand befindliche CPU **12B**, um diese dazu zu veranlassen, mit dem Aussenden von SCHLAFEN-Befehlssymbole zu beginnen. Die CPU **12A** beginnt daraufhin ebenfalls mit dem Senden von SCHLAFEN-Befehlssymbolen an den Router **14A**, die an die Stelle der ansonsten gesendeten BEREIT-Befehlssymbole treten, und auf die ein eigenadressiertes Nachrichtenpaket folgt. Ein beliebiges Nachrichtenpaket, das sich gerade in Empfang oder in übertragung befinden mag, während die SCHLAFEN-Befehlssymbole von dem Router **14A** empfangen und erkannt werden, kann dabei fertig abgearbeitet werden. Alle weiteren Nachrichtenpakete werden jedoch zurückgehalten, mit einer Ausnahme: die von der CPU **12A** kommenden, eigenadressierten Nachrichtenpakete. Diese Nachrichtenpakete werden empfangen und von dem Router **14A** (über die Bestimmungsort-Adresse) zurück an die CPU **12A** gesendet. Das SCHLAFEN-Befehlssymbol hat die Funktion, den Router **14A** zum Zweck des Synchronisierungsprozesses in einen Ruhezustand zu versetzen. Das von der CPU **12A** gesendete, eigenadressierte Nachrichtenpaket dient dazu, wenn es von der CPU **12A** wieder rückempfangen worden ist, letztere davon in Kenntnis zu setzen, dass sich der Router **14A** nun in einem Ruhezustand befindet, da das nach den SCHLAFEN-Befehlssymbolen gesendete, eigenadressierte Nachrichtenpaket notwendigerweise das zuletzt von dem Router **14A** verarbeitete Paket sein muss.

[0436] Bei Schritt 954 prüft die CPU **12A**, ob sie das eigenadressierte Nachrichtenpaket, das sie im Anschluss an die Initiierung des Sendens von SCHLAFEN-Befehlssymbolen gesendet hat, zurückerhalten hat. Falls sie den Rückempfang des Nachrichtenpakets feststellt und somit sicher sein kann, dass der Router **14A** vorübergehend keine weiteren Nachrichtenpakete bearbeitet, geht die CPU **12A** zu Schritt 956 über und sendet ein SYNC-Befehlssymbol an den Router **14A**. Wenn das SYNC-Befehlssymbol von dem Router empfangen worden ist und von der Befehlsdecodierlogik **544** (**Fig.** 20A) als solches erkannt worden ist, wird die Steuerlogik **509** benachrichtigt. Die Steuerlogik **509** wartet auf die nächste Anstiegsflanke des SYNC-Taktes **970** (Zeit t_1-**Fig.** 31B), um den Befehlssymbolgenerator **632** (**Fig.** 22) der Portausgänge $\mathbf{504_9}$, $\mathbf{504_5}$ anzuweisen, SYNC-Befehlssymbole zu generieren, welche an die CPUs **12A**, **12B** zurückreflektiert werden (Schritt 958).

[0437] Als nächstes gibt in Schritt 960 (und Zeitabschnitt t_2 des SYNC-Taktes **970**) die Steuerlogik **509** des Routers ein RESET-Signal **972** aus, welches an die beiden Taktsynchronisierungs-FIFOs **518** angelegt werden, die in der Eingangslogik **5054**, **5055** jenes Routers enthalten sind, welcher Symbole unmittelbar von den CPUs **12A**, **12B** empfängt. Solange RESET aktiviert ist, werden dadurch die beiden Taktsynchronisierungs-FIFOs **518** in einem vorübergehend nicht-operativen Reset-Zustand gehalten, wobei der Eingabeund der Entnahme-Zeigerzähler **530**, **532** (**Fig.** 20A) jeweils in einen bekannten Zustand versetzt werden und auf Speicherorte innerhalb der Speicherwarteschlange **518** zeigen, die um eine vorgegebene Anzahl von Speicherorten (zwei in dem vorliegenden Beispiel) voneinander beabstandet sind, wie bereits weiter oben in Verbindung mit der Reset-Prozedur beim Einschalten des Stroms besprochen.

[0438] Ähnlich verhält es sich mit den SYNC-Symbolen, die von den Routern **14A**, **14B** an die CPUs **12** zurück reflektiert werden. Da jedes der von den CPUs **12** empfangenen SYNC-Symbole von den Speicher- und Verarbeitungseinheiten der Paketempfänger **96** (**Fig.** 5 und 6) dazu führt, dass das RESET-Signal von den Paketempfängern **96** (eigentlich von den Speicher- und Verarbeitungselementen **110**; Fig. 6) einer jeden CPU **12** ausgegeben wird, wird das RESET-Signal an die Taktsynchronisierungs-FIFOs **102** (**Fig.** 6) der CPUs **12** angelegt. Dieses CPU RESET-Signal hält ebenfalls die CPU-Taktsynchronisierungs-FIFOs **102** beider CPUs **12** in einem Reset-Zustand, wobei deren Speicherwarteschlangen **126** (**Fig.** 7A) und deren Eingabe- und Entnahmezähler **128**, **130** jeweils in einen bekannten Zustand versetzt werden.

[0439] Bei Schritt 962, und Zeitabschnitt t_3 des SYNC-TKT-Signals **970**, werden die Sendetaktsignale (S_Tkt), welche die Symbolübertragung zwischen den CPUs **12A**, **12B** und den Routern **14A**, **14B** begleiten,

vorübergehend gestoppt. Bei Schritt 963 (Zeitabschnitt t_4) deaktivieren die CPUs **12** und die Router **14A**, **14B** die RESET-Signale, wodurch die Taktsynchronisierungs-FIFOs der CPUs **12A**, **12B** und der Router **14A**, **14B** aus ihrem Reset-Zustand entlassen werden. Bei Schritt 964 (t_5) nehmen der Router **14A** und die CPUs **12** die S_TKT-gestützte Übertragung wieder auf, wobei mit einer kurzen, konfigurierbaren Verzögerung begonnen wird, die es erlaubt, Einstellungen vorzunehmen betreffend die während der Übertragung auf den Links maximal zu erwartende Verzögerungszeit. Nach Beendigung dieser Verzögerung fahren die Router **14A** und die CPUs **12** damit fort, Daten aus ihren jeweiligen Taktsynchronisierungs-FIFOs zu entnehmen und führen so ihren normalen Betrieb weiter. Die Taktsynchronisierungs-FIFOs des Routers **14A** beginnen damit, Symbole aus der Warteschlange zu entnehmen (zuvor durch RESET auf LEERLAUF-Symbole eingestellt), und der S_Tkt beginnt, Symbole in Warteschlange einzugeben. Zu beachten ist dabei, dass das erste Symbol, welches von der CPU **12A** kommend mit dem S_Tkt empfangen wird, beispielsweise an dem Warteschlangenspeicherort **0** (oder an irgendeinem anderen Speicherort, auf welchen der Wert, auf den der Eingabe-Zeigerzähler zurückgesetzt worden ist, zeigt) gemeinsam mit dem begleitenden S_Tkt-Signal in das Taktsynchronisierungs-FIFO eingegeben wird. In ähnlicher Weise wird auch das erste von der CPU **12B** kommende Symbol ebenfalls an dem Speicherort 0 (bzw. an jedem beliebigen anderen Speicherort, auf welchen der Eingabe-Zeiger während des RESET-Vorgangs eingestellt wurde) in die FIFO-Warteschlange eingegeben. Die Taktsynchronisierungs-FIFOs des Routers **14A** sind nun entsprechend synchronisiert, um sich auf jede denkbare Verzögerung **640** einstellen zu können, zu der es in dem einen Kommunikationspfad zwischen dem Router **14A** und den CPUs **12A**, **12B** in Bezug auf den anderen Pfad kommen kann.

[0440] In ähnlicher Weise wird zur selben virtuellen Zeit der Betrieb der Taktsynchronisierungs-FIFOs **102** beider CPUs **12A**, **12B** wiederaufgenommen und somit mit dem Router **14A** synchronisiert. Die CPUs **12A**, **12B** stellen daher auch das Senden von SCHLAFEN-Befehlssymbolen zugunsten von BEREIT-Symbolen ein und nehmen ordnungsgemäss die Übertragung von Nachrichtenpaketen wieder auf.

[0441] Damit ist der Synchronisierungsprozess für den Router **14A** abgeschlossen. Der Prozess muss jedoch auch für den Router **14B** durchlaufen werden. Die CPU springt also zum Schritt 952 zurück und führt die Schritte 952 – 966 erneut durch, diesmal allerdings mit dem Router **14B** anstelle des Routers **14A**, woraufhin alle CPUs **12A**, **12B** und Router **14A**, **14B** für einen frequenzsynchronisierten Betriebsmodus initialisiert sind. Was zur Einrichtung des Duplex-Betriebsmodus noch zu tun bleibt, ist die beiden CPUs **12A**, **12B** in denselben Betriebszustand zu versetzen und dafür zu sorgen, dass sie dieselben Befehle im wesentlichen zum selben Zeitpunkt ausführen. Die Vorgangsweise, die darin besteht, zwei CPUs **12** in denselben Zustand zu versetzen, auch Reintegration genannt, wird weiter unten beschrieben. Zuvor gilt es jedoch noch, unter Voraussetzung, dass die CPUs **12A**, **12B** nun im Duplexbetrieb arbeiten, das Verfahren und die Vorrichtung zu beschreiben, die verwendet werden, um einen möglichen Fehler zu erkennen und zu bearbeiten, welcher zu einer Divergenz der CPUs in Bezug auf deren Duplexbetrieb führt.

Divergenzerkennung und -handhabung:

[0442] Durch den Duplex-Betriebsmodus wird eine funktionellfehlende Fehlertoleranz auf CPU-Ebene implementiert. Eine jede des Paars von duplexbetriebenen CPUs (z. B. die CPUs **12A**, **12B** des Systems **10** – **Fig.** 1A) ist im wesentlichen eine identische Kopie der jeweils anderen, einschliesslich der Zustandsinformationen und des Speicherinhalts, und beide führen im wesentlichen zur selben Zeit identische Befehle eines identischen Befehlsstroms aus und bilden somit eine logische, fehlertolerante CPU. Durch einen Fehler der einen oder der anderen der CPUs **12A**, **12B** wird der Betrieb des Systems **10** nicht gestoppt bzw. nicht einmal verlangsamt-sofern dieser Fehler erkannt und ordnungsgemäss gehandhabt wird. Zur Erkennung einer fehlerhaften CPU ergibt sich naheliegenderweise aus dem Duplexbetrieb: die E/A-Ausgabedaten beider CPUs **12A**, **12B** sind bei einem ordnungsgemässen Duplexbetrieb Symbol für Symbol miteinander identisch. Alles was also zu tun ist, um die ordnungsgemässe Fortsetzung des Duplexbetriebs bestätigt zu bekommen, ist, die beiden E/A-Ausgabedaten der duplexbetriebenen CPUs symbolweise miteinander zu vergleichen. Eine fehlerhafte CPU weicht von dem Betriebszustand der anderen ab und diese Divergenz kommt schlussendlich in den E/A-Ausgabedaten der CPUs zum Ausdruck.

[0443] In **Fig.** 32 wird die Vorgehensweise veranschaulicht, die verwendet wird, um als erstes eine Divergenz bei den Routern **14A**, **14B** zu erkennen (Schritte 1000, 1002), und in weiterer Folge diese Divergenz auf elegante Weise handzuhaben, um den Betrieb der fehlerhaften CPU so bald wie möglich zu beenden und sie so daran zu hindern, schlechte Daten innerhalb des Systems **10** zu verbreiten. In Schritt 1000 der **Fig.** 32 arbeitet das duplexbetriebene Paar von CPUs **12A**, **12B** (**Fig.** I) somit synchron im Gleichschritt als eine logische CPU. In regelmässigen Abständen senden die CPUs **12** E/A-Daten über ein Nachrichtenpaket, das für ein Peripheriegerät des einen oder anderen Subprozessorsystems **10A**, **10B** bestimmt ist. Je nach Bestimmungsort des abgehenden Nachrichtenpakets empfängt bei Schritt 1002 einer der Router **14A** oder **14B** diese E/A-Daten und vergleicht beim Empfang des Pakets jedes Symbol des von der CPU **12A** kommenden Nachrichtenpakets mit jenem der CPU **12B**. Der Vergleich erfolgt an den Ausgängen der Eingangslogik **505** der Porteingängen

502_4 und **502_5**, die entsprechend geschaltet sind, um über eine (nicht dargestellte) Vergleichsschaltung herkömmlicher Bauart E/A-Daten von den CPUs **12A**, **12B** zu empfangen. Falls die empfangenen Symbole gleich sind, verbleibt das Verfahren auf den Schritten 1000 und 1002-was auf eine ordnungsgemässen Betrieb hindeutet.

[0444] Sollten während des Schritts des Vergleiches 1002 unterschiedliche Symbole erkannt werden, so gibt die (nicht dargestellte) Vergleichsschaltung des Routers **14** ein FEHLER-Signal an die Router-Steuerlogik **509** aus, wodurch der Prozess zu Schritt 1004 weiterspringt, bei welchem der die Divergenz erkennende Router **14** ein DVRG-Befehlssymbol an beide CPUs **12A**, **12B** überträgt. Vorzugsweise wartet der Routen so lange wie möglich zu, bevor er das DVRG-Signal sendet, um die Zeit zwischen dem Melden der Divergenz und dem Abklären, welche CPU den Betrieb fortsetzt, möglichst gering zu halten.

[0445] Es mag an diesem Punkt hilfreich sein, in einem kurzen Exkurs nachfolgend verschiedene gegensätzliche Ziele zu beschreiben, die durch diese Technik der Divergenzerkennung verfolgt werden sollen:

[0446] Als erstes müssen die Routen **14A**, **14B** unverzüglich Schritte setzen, um eine Ausbreitung des Fehlers auf den Rest des Systems zu verhindern. So setzen die Routen ungeachtet der Erkennung einer Divergenz mit dem Senden der Nachrichtenpakete über ihre vorbestimmten Leitwege fort, allerdings unter Zurückhaltung des abschliessenden Symbols des Nachrichtenpakets: des Zustandssymbols, welches ein "Paket schlecht"- bzw. ein "Paket gut"-Zustandssymbol (TPBbzw. TPG-Zustandssymbol) sein kann. Ohne dieses Symbol kann ein im Informationsstrom nachgeordnetes Bestimmungselement ein empfangenes Nachrichtenpaket nicht verwenden.

[0447] Als zweites Ziel sollen möglichst wenige Nachrichtenpakete zertrennt werden. Wie weiter unten beschrieben, wird eine der CPUs **12A**, **12B** als "bevorzugte" oder primäre CPU bestimmt, und wenn die CPUs im Duplexmodus betrieben werden, wird nur der von der bevorzugten CPU eines Routers kommende Nachrichtenverkehr weiterübertragen. Die Paketzertrennung wird minimiert, indem es einem Router ermöglicht wird, angesichts der Erkennung einer Divergenz die Fertigstellung der Übertragung eines Nachrichtenpakets so lange fortzusetzen, bis bestimmt werden kann, welche CPU fehlerhaft ist, sofern eine solche Bestimmung möglich ist.

[0448] Wenn es sich dabei nicht um die bevorzugte CPU handelt, so wird das Nachrichtenpaket durch die Übertragung des Abschlusssymbols, in diesem Fall eines TPG-Symbols, freigegeben.

[0449] Als drittes muss ein Router, der eine Divergenz erkennt, genau bestimmen, welche Fehler es sind, die passiert sind, damit es zu einer Divergenz kommen konnte. Zu diesem Zweck sucht er nach einfachen Zinkfehlern, nach verlorengegangenen Linkebenen-"Halte"-Symbolen und nach CRC-Fehlern. Die CPUs **12** räumen im Anschluss an dem Empfang des DVRG-Symbols hinreichend Zeit ein, um die Ausgabe allfälliger Linkebenen-Haltesymbol-Zeitablauf-Fehlermeldungen abwarten zu können. Ein Router, der eine Divergenz erkennt (ohne dabei einen entsprechenden, einfachen Zinkfehler erkennen zu können) nimmt sich die Zeit, um den CRC des empfangenen Nachrichtenpakets zu prüfen, indem er auf das Ende des Nachrichtenpakets wartet, bevor er die Divergenz mittels des DVRG-Symbols meldet.

[0450] Viertens und letztens muss das System **10** die Divergenzhandhabung innerhalb einer kurzen, begrenzten Zeitspanne vollenden, um TNet-Transaktions-Zeitablauffehler bzw. untragbare E/A-Verzögerungen zu vermeiden. Dieses Ziel steht gewissermassen im Widerspruch zu dem Zurückhalten der Freigabe des Nachrichtenpakets (durch das Zurückhalten des abschliessenden Zustandssymbols), da das Warten auf die Fertigstellung der Übertragung eines von einer CPU kommenden Nachrichtenpakets viel Zeit in Anspruch nehmen kann. Eine solche Verzögerung kann jedoch nicht zu einem TNet Zeitablauffehler führen, wenn die ungünstigste Zeitspanne, innerhalb derer eine CPU ein Nachrichtenpaket übertragen muss, garantiert ist.

[0451] Die CPUs **12** starten beim Empfang eines DVRG-Symbols einen Zeitgeber, der dazu verwendet wird, eine vorgegebene Zeitspanne festzusetzen, innerhalb derer die CPUs **12** zu bestimmen versuchen, welche von ihnen den Fehler aufweist und den Betrieb beenden muss und welche von ihnen weiterarbeiten muss (Schritt 1006). Ausserdem reflektieren beide CPUs **12A**, **12B** das DVRG-Befehlssymbol an die beiden Router **14A**, **14B** zurück. Wenn ein Router **14A**, **14B**, der dieses reflektierte DVRG-Symbol empfängt, keine Divergenz erkannt hat, oder zuvor noch auf kein DVRG-Symbol gestossen ist, so reflektiert er auch ein DVRG-Symbol zurück an die CPUs. Durch das in dieser Weise erfolgende Reflektieren der DVRG-Befehlssymbole wird gewährleistet, dass alle CPUs **12** und Router **14A**, **14B** auf ein DVRG-Symbol gestossen sind und so von dem möglichen Vorhandensein einer Divergenz unterrichtet sind.

[0452] Sind erst einmal alle Beteiligten (die CPUs **12A**, **12B**, und die Router **14A**, **14B**) informiert, dass eine Divergenz erkannt worden ist, was auf einen Fehler einer der CPUs (oder eines Routers) schliessen lässt, so gilt es, darauf zu achten, dass keine Folgeerscheinungen aus diesem Fehler in Form von schlechten Daten auf den Rest des Systems **10** übertragen werden. Gleichzeitig muss das System **10** den Fehler tolerieren und seinen Betrieb fortsetzen.

[0453] Somit müssen alle (von den CPUs) abgehenden Paketübertragungen zumindest teilweise fortgesetzt werden, bis entschieden werden kann, ob ein Nachrichtenpaket, das zur selben Zeit von den CPUs kommend eintrifft, zu welcher der Router eine Divergenz erkennt, gut oder schlecht ist.

[0454] Ausserdem muss die divergenzverursachende CPU bestimmt werden und in transparenter Weise (d. h. ohne äusseres Zutun) aus dem System entfernt werden. Diese letztere Aufgabe fällt in den Verantwortungsbereich der CPUs **12**, denn nachdem die Divergenz und allfällige Fehler an die CPUs **12** gemeldet worden sind, müssen diese untereinander entscheiden, welche von ihnen den Betrieb fortsetzt und welche ihren weiteren Betrieb beendet und sich somit praktisch selber aus dem System **10** zurückzieht.

[0455] Somit analysiert unter Schritt 1006 der Divergenzroutine eine jede der CPUs **12A**, **12B** die verschiedenen ihr bereitgestellten Fehlerangaben; diese Fehleranalyse wird weiter unten noch genauer besprochen.

[0456] Vorerst bedarf jedoch die Funktion des (bzw. der) divergenzerkennenden Router(s) **14**, durch welche die Ausbreitung schlechter Daten begrenzt wird, einer Erklärung. Nachdem das DVRG-Symbol von einem Router **14** ausgegeben, oder empfangen, worden ist, werden alle weiteren Nachrichtenpakete, die zu dem Zeitpunkt, da die Divergenz erkannt wurde, von den CPUs kommend empfangen worden sind bzw. im Begriff sind, weitergeleitet zu werden, bzw. das empfangene DVRG-Symbol, ausgenommen allerdings das paketabschliessende Zustandssymbol, d. h. das Zustandsanzeigesymbol 'TPG' (Paket gut) bzw. 'TPB' (Paket schlecht), durch den Router hindurchgeleitet. Während des Duplexbetriebs wird, wie weiter oben bereits kurz erklärt, ein jeder der Router **14A**, **14B** durch Setzen einer Bitposition in dem in der Steuerlogik **509** (**Fig.** 19A) enthaltenen Konfigurationsregister (nicht gezeigt) entsprechend konfiguriert, so dass er über eine "bevorzugte" CPU verfügt. Im Duplexbetrieb überträgt der Router das von der bevorzugten CPU empfangene Nachrichtenpaket weiter; das von der anderen oder "nicht bevorzugten" CPU kommende Paket wird nur zum Zweck der Divergenzerkennung verwendet. Die Router müssen abwarten, bis die CPUs eine Entscheidung getroffen haben, welche von ihnen den Betrieb fortsetzt, wovon die Router **14A**, **14B** benachrichtigt werden (Schritt 1012), bevor das Paket durch Anfügen des TPG/TPB-Zustandsanzeigesymbols "freigegeben" werden (Schritt 1014). Wenn der Router benachrichtigt wird, dass die bevorzugte CPU **12** zur Fortführung des Betriebs bestimmt wurde, so gibt der Router das Nachrichtenpaket durch Hinzufügen und Senden des Zustandsanzeigesymbols 'TPG' frei. Wenn der Router hingegen benachrichtigt wird, dass es sich umgekehrt verhält, d. h. dass es nicht die bevorzugte CPU ist, die den Betrieb fortführt, wird das Nachrichtenpaket mit dem Symbol 'TPB' versehen und somit ausgeschieden.

[0457] Um die Menge an verlorenen Daten zu beschränken (wie weiter oben als zweites Ziel erwähnt), werden die beiden Router mit unterschiedlichen bevorzugten CPUs konfiguriert (z. B. die bevorzugte CPU von Router **14A** ist CPU **12A**, die bevorzugte CPU von Router **14B** ist CPU **12B**).

[0458] Nachdem die Nachricht über die erkannte Divergenz per Rundruf an die CPUs **12A**, **12B** und an die Router **14A** und **14B** übermittelt worden ist, wird fortgesetzt, indem eine jede der CPUs **12A**, **12B** damit beginnt, die bei Schritt 1006 eingetretene Situation zu bewerten, in dem Bemühen, unabhängig voneinander zu bestimmen, wo der Fehler liegt, der für die Divergenz verantwortlich ist. Sobald feststeht, in welcher der CPUs **12A**, **12B** der Fehler aufgetreten ist (Schritt 1008), beendet diese CPU ihren Berieb selbsttätig (Schritt 1012) und lässt die andere alleine, wenn auch im Simplexmodus, weiterarbeiten. Für den Fall, dass die CPUs **12A**, **12B** nicht in der Lage sind, aus einem der erkannten bzw. gemeldeten Fehler zu bestimmen, bei welcher von ihnen der Fehler liegt, so kommt ein "Tie-Break"-Bit (Schritt 1010) zum Einsatz, welches in jeweils einem der Konfigurationsregister **74** (**Fig.** 5) der Schnittstelleneinheit **24** einer jeden CPU enthalten ist.

[0459] Im folgenden wird vorübergehend wieder auf Schritt 1006 Bezug genommen, wobei die Entscheidung der Frage, bei welcher der CPUs **12A**, **12B** der Fehler liegen mag, in erster Line darauf basiert, welche Fehler auf dem Kommunikationspfad zwischen den CPUs **12A**, **12B** und den Routern **14A**, **14B** erkannt werden. Nachdem die Router **14A**, **14B** das Vorhandensein einer Divergenz bemerkt haben, setzt jeder, wie oben erwähnt, seinen normalen Betrieb fort: jedes einzelne Nachrichtenpaket, das zu dem Zeitpunkt, da der eine Divergenz anzeigende Symbolunterschied erkannt wurde, oder im Anschluss daran, von den CPUs **12A**, **12B** kommend gerade von einem Router **14A**, **14B** empfangen wird, mit Ausnahme des abschliessenden Zustandsanzeigesymbols, wird durch den Router hindurchgeleitet. Beide Router **14A**, **14B** überwachen weiterhin die Kommunikationspfade (TNet-Links L), zwischen dem speziellen Router **14** und den CPUs **12** auf mögliche Linkprotokollfehler und/oder Verletzungen wie zum Beispiel auf einen erkannten CRC-Fehler, auf Befehlssymbolfehler oder auf andere Arten von Linkfehlern hin. Wenn ein solcher Fehler bzw. eine solche Verletzung auf einem Link **L** (z. B. dem Link Lx-**Fig.** 1A) erkannt wird, so sendet der erkennende Router **14A**, **14B** auf diesem Link (Lx) ein 'Link schlecht'-Symbol (TLB-Symbol) an die CPU zurück. Zur selben Zeit sendet der erkennende Router an die andere CPU **12** (über das andere Link Ly) ein 'Anderes Link schlecht'-Symbol (OLB-Symbol). Auf diese Weise werden beide CPUs **12** davon in Kenntnis gesetzt, dass ein Kommunikationspfad (die Linkverbindung Lx) zu einem der Router **14A**, **14B**, oder irgendein mit diesem Link verbundenes Element, fehlerhaft oder unzuverlässig sein kann. Diese Information wird von den CPUs bei dem in Schritt 1006 erfolgenden Versuch benutzt, zu bestimmen, welche CPU den Betrieb beenden sollte, während die andere CPU ihre Arbeit, wenn auch im Simplexbetriebsmodus, fortsetzt. Die CPUs akkumulieren diese Fehlerinformationen und erstellen eine Tabelle ähnlich der nachfolgenden Tabelle 7.

TABELLE 7

	Lokaler ROUTER	Entfernter ROUTER	AKTION der LOKALEN CPU
(1)	Kein Fehler	Kein Fehler	Primäre fortsetzten; Zweit-CPU bricht ab.
(2)	Kein Fehler	TLB oder Fehler	Abbrechen.
(3)	Kein Fehler	OLB	Fortsetzen; andere CPU bricht ab.
(4)	TLB oder Fehler	Beliebiger Zustand	Abbrechen.
(5)	OLB	Beliebiger Zustand	Fortsetzen; andere CPU bricht ab.

[0460] Die Tabelle 7 listet die fünf möglichen Zustände auf, die von einer CPU erkannt bzw. dieser gemeldet werden können und auf deren Basis eine CPU bestimmen kann, was bei Eintreten einer Divergenzmeldung zu tun ist. Der in Tabelle 7 verwendete Begriff "lokal" soll sich auf den Router **14A**, **14B** beziehen, welcher in demselben Subprozessorsystem **10A**, **10B** wie die CPU enthalten ist. So ist zum Beispiel, unter Bezugnahme auf **Fig.** 1A, der Router **14A** der "lokale" Router in Bezug auf die CPU **12A**, jedoch der "entfernte" Router in Bezug auf die CPU **128**. Das in Tabelle 7 verwendete "primär" bezieht sich auf das bereits weiter oben erwähnte Tie-Break-Bit: das in einem der Konfigurationsregister **74** der Schnittstelleneinheit **24** (**Fig.** 5) einer jeden CPU enthaltene Bit. Ist dieses Bit in einen ersten Zustand versetzt, so wird die betreffende CPU **12** dadurch als Primäre CPU gekennzeichnet, und umgekehrt muss bei der anderen CPU **12** dasselbe Konfigurationsbit in einen umgekehrten Zustand gesetzt sein, um anzuzeigen, dass es sich um die "Zweit-CPU" handelt. Diese Bits werden im Zuge der Divergenzhandhabung in jenen Fällen verwendet, um eine der CPUs auszuwählen, die den Betrieb fortsetzt, in denen die CPUs auf andere Weise nicht imstande sind, diese Bestimmung vorzunehmen. Es sei hier angemerkt, dass in allen Fällen, in denen der lokale Router keine Fehler meldet, die CPU das Fällen einer diesbezüglichen Entscheidung verzögert. Damit wird der Möglichkeit Rechnung getragen, dass die andere CPU Fehler erkennt und einer Selbstprüfung unterzieht, und dass der lokale Router in der Folge das Aussetzen der Haltesymbole erkennt und den Fehler in Form eines OLB-Symbols an die lokale CPU meldet.

[0461] Die Bemerkung "Beliebiger Zustand" bedeutet lediglich, dass ungeachtet des Meldezustands (Meldung eines Fehlers oder keine Fehlermeldung) seitens des entfernten Routers, die lokale CPU jene Aktion setzt, die unter dem Titel "Aktion der Lokalen CPU" angeführt ist.

[0462] Die Bemerkung "Aktion der Lokalen CPU" bezeichnet die von einer bestimmten der CPUs **12A**, **12B** zu setzenden Aktion, wenn die auf diese bestimmte CPU die in einer der Reihen der Tabelle angeführten Bedingungen zutreffen. Wenn zum Beispiel die in Reihe **4** angeführten Bedingungen auf die CPU **12A** zutreffen (der Router **14A** hat einen Fehler gemeldet, bzw. die CPU **12A** hat einen Fehler erkannt), so entscheidet die CPU **12A**, dass sie ihren Betrieb zu Gunsten der anderen CPU des duplexbetriebenen CPU-Paars abbricht und somit der CPU **12B** ermöglicht, ihren Betrieb fortzusetzen. Umgekehrt geben die Bedingungen der Reihe **4** an, dass die andere CPU **12B** von ihrem "entfernten" Router (dem Router **14A**) ein OLB-Symbol empfangen hat, mit welchem ihr die Tatsache gemeldet wurde, dass der Kommunikationspfad zwischen dem Router **14A** und der CPU **12A** fehlerverdächtig ist. Aus der Sicht der CPU **12B** handelt es sich hierbei um die Bedingung, die in der Reihe **3** oder **5** ausgedrückt wird. Hat nur einer der Router **14A**, **14B** einen Fehler entdeckt (in diesem Fall der Router **14A**), so wird der CPU **12B** eine in Reihe **3** angeführte Meldung vorgelegt. Für den Fall, dass beide Router **14A**, **14B** einen Fehler erkennen, meldet ein jeder ein TLB an die CPU **12A** und ein OLB an die CPU **12B**. Die CPU **12B** nimmt das von dem Router **14B** kommende OLB zur Kenntnis, setzt es mit der Bedingung aus Reihe **5** in Verbindung, gibt ein IOY-Symbol an den Router **14B** aus und setzt ihren Betrieb fort.

[0463] Zu beachten ist hier, dass die Reihen **4** und **5** der Tabelle 7 Anlass zu einigen Problemen geben könnten. Wenn zum Beispiel der Router **14A** ein TLB an die CPU **12A** meldete und der Router **14B** ein TLB an die CPU **12B** meldete, so würden beide CPUs sich selbst ausser Gefecht setzen (einfrieren). Wenn jedoch angenommen wird, dass zu einer gegebenen Zeit immer nur höchstens ein Fehler auftritt, so dass es nicht zu einem Zustand kommen kann, in welchem beide lokalen Router Fehler entwickeln, was keineswegs eine ungewöhn-

liche Annahme darstellt, so sind die Bedingungen der Tabelle annehmbar. Andererseits ist es nicht erforderlich, dass das System am Laufen gehalten wird, wenn Mehrfachfehler an mehr als einem Link zwischen den Routern **14** und den CPUs **12** auftreten. In ähnlicher Weise könnte es auch passieren, dass wenn beide Router OLBs an ihre lokalen CPUs melden, beide CPUs versuchen, den Betrieb im Alleingang fortzusetzen. Dies würde auf einen Taktgabefehler schliessen lassen. Die Taktschaltungen sollten solche Fehler erkennen und die fehlerbehaftete CPU einfrieren.

[0464] Die Tabelle 7 stellt daher jene Fehlerangaben dar, die von den CPUs **12** und den Routern **14A**, **14B** erkannt werden können. Generell verhält es sich so, dass wenn eine CPU **12** von ihrem lokalen Router eine Fehlermeldung empfängt, diese ihren Betrieb zu Gunsten der anderen abbricht und es so der anderen des CPU-Paars ermöglicht, den Betrieb fortzusetzen.

[0465] Auf die Bezeichnungen Primäre CPU und Zweit-CPU muss nur dann zurückgegriffen werden, wenn keine der CPUs bei Ablauf ihres jeweiligen Zeitgebers (der zum Zeitpunkt des Empfangs des DVRG-Befehlssymbols gestartet wurde) eine wie auch immer geartete Fehlermeldung empfangen hat. In diesem Fall fällt die Entscheidung durch den Rückgriff auf das primäre Konfigurationsbit einer jeden CPU. Die als Primäre CPU bezeichnete CPU setzt den Betrieb fort und nimmt an, dass die andere abgebrochen hat.

[0466] Auf diese Weise treffen die CPUs **12** eine Entscheidung (Schritt 1008) wer den Betrieb fortsetzt und wer nicht und gehen anschliessend zu Schritt 1012 über, in welchem eine CPU gemäss der in einem der Schritte 1006, 1010 getroffenen Entscheidung ihren Betrieb abbricht.

[0467] Die CPU **12**, welche den Betrieb abbricht, tut dies indem sie eine Selbstprüfung einleitet und anschliessend einfriert. Die CPU, welche den Betrieb fortsetzt, sendet ein IOY-Symbol ('Ich bin dir übergeordnet') an die Router **14A**, **14B**, um sie zu informieren, dass die Router sich nur an der CPU orientieren sollen, die den Betrieb fortsetzt, und sämtliche Übertragungen seitens der anderen CPU ignorieren sollen. Als Reaktion darauf ändern die in der Steuer- und Zustandseinheit **509** (**Fig.** 19A) enthaltenen Ablaufsteuereinheiten (nicht dargestellt) die weiter oben beschriebenen "primären" Bits entsprechend ab.

[0468] Einige Beispiele sollen in folgenden das Konzept der Divergenz verdeutlichen. Unter erneuter Bezugnahme auf **Fig.** 1A sei angenommen, dass die CPUs **12A**, **12B** im Duplex-Betriebsmodus arbeiten, und dass bei der CPU **12A** ein Fehler auftritt, so dass alle nachfolgenden E/A-Operationen sich von jenen der CPU **12B** unterscheiden. Demgemäss wird i anlässlich der nächsten E/A-Übertragung einer der Router **14A**, **14B** (und zwar der, an den Daten adressiert sind, bzw. beide, falls die CPU **12A** E/A-Daten fehlerbedingt an einen anderen Bestimmungsort adressiert als jenen der CPU **12B**) eine Divergenz erkennen. Sie warten jedoch ab, wie oben erwähnt, bis das gesamte Paket empfangen ist, um zu bestimmen ob die CRC-Prüfung des vorliegenden Nachrichtenpakets erfolgreich ist, oder bis irgendwelche einfache Fehler auftreten, woraufhin jeder Router ein DVRG-Symbol auf beiden Links L überträgt. Angenommen beide Router treffen auf Protokollfehler. Erkannte Protokollfehler führen umgehend dazu, dass die Router **14A**, **14B** ein DVRG-Symbol an beide CPUs **12** senden, und ein 'Link schlecht'-Symbol (TLB-Symbol) auf die Links L zurücksenden, auf denen die Fehler erkannt wurden, d. h. die Links Lx, Ly, welche jeweils die Router **14A**, **14B** mit der CPU **12A** verbinden. Zur selben Zeit, zu der sie das TLB-Symbol senden, senden beide Router **14A**, **14B'**Anderes Link schlecht'-Symbole (OLB-Symbole) an die CPU **12B**. Bei Empfang des DVRG-Symbols reflektiert die CPU **12A** dieses Symbol an die Router **14A**, **14B** zurück, startet ihren internen Divergenzprozess-Zeitgeber, und beginnt mit der Bestimmung ob ihr Betrieb fortgesetzt oder abgebrochen werden soll. Da sie ein TLB-Symbol von ihrem lokalen Router **14A** empfangen hat, entscheidet die CPU **12A** unverzüglich, dass sie ihren Betrieb zugunsten der CPU **12B** abbrechen muss, welche dadurch ihrerseits ihren Betrieb fortsetzen kann (Reihe **4**, Tabelle 7).

[0469] Gemäss diesem Szenario hat die CPU **12B** weiterhin von beiden Routern **14A**, **14B** OLB-Symbole empfangen und au^{f} Grund dieser Meldungen entschieden, dass sie jene CPU ist, die den Betrieb fortsetzen soll. Demgemäss sendet sie ein IOY-Symbol an beide Router **14A**, **14B**. In Reaktion darau^{f} konfigurieren sich die Router entsprechend, so da^{ss} Paketübertragungen nur mehr in Verbindung mit der CPU **1^{2B}** durchgeführt werden und jedwede von der CPU **12A** kommenden Übertragungen ignoriert werden.

[0470] Die Divergenzerkennung kann auch einen fehlerbehafteten Router erkennen. Angenommen zum Beispiel der Router **14A** ist mit einen Fehler behaftet, welcher bewirkt, dass der Router divergierende Aktionen ausführt, die wiederum zur Folge haben, dass die beiden CPUs **12A**, **12B** ein divergierendes Verhalten aufweisen. Der funktionstüchtige Router, d. h. der Router **14B**, erkennt diese Divergenz und meldet sie mit einem DVRG-Symbol an eine jede der CPUs. Jede CPU reflektiert das DVRG-Symbol an beide Router **14A**, **14B** zurück. Je nach Art des vorliegenden Fehlers reflektiert der Router **14A** das DVRG-Symbol an die CPUs zurück oder auch nicht. Die CPU **12A** bemerkt das Vorhandensein von Fehlern auf ihrem lokalen Link, welcher sie mit dem fehlerbehafteten Router **14A** verbindet, und entscheidet demgemäss, dass sie eine Selbstprüfung durchführen und den Betrieb abbrechen muss. Umgekehrt erkennt der Router **14B** diesen Betriebsabbruch und gibt ein TLB-Symbol an die CPU **12A** und ein OLB-Symbol an die CPU **12B** zurück. Die CPU **12B** gibt daraufhin ein IOY-Befehlssymbol an beide Router aus.

[0471] Bei der oben beschriebenen Divergenz handelte es sich um eine "saubere" Divergenz, bei welcher der eine oder andere Router **14A**, **14B** oder auch beide eine Divergenz erkennen und ein DVRG-Symbol an die

CPUs ausgeben, bei der jedoch weder die CPUs **12A**, **12B**, noch die Router **14A**, **14B** irgendeinen Fehler erkennen. Demgemäss entscheidet die "primäre" CPU, die zuvor während der Initialisierung in den Konfigurationsregistern als solche festgelegt worden ist, dass sie den Betrieb fortsetzen und ein IOY-Symbol an beide Router **14A**, **14B** ausgeben muss. Zur selben Zeit führt die "Zweit"-CPU eine Selbstprüfung durch und bricht anschliessend ihren Betrieb ab.

[0472] Neben den weiter oben beschriebenen sind unter anderem auch die folgenden Arten von Fehlern und Störungen als Ursachen von Divergenzen in Betracht zu ziehen:

– Unkorrigierbare Speicherfehler, die dazu führen, dass die CPU sofort den Betrieb einfriert, um eine möglichen Ausbreitung des Fehlers zu verhindern. Für die Router **14A**, **14B** erscheint die CPU stromlos, wodurch sie veranlasst werden, ein TLB-Symbol an die fehlerbehaftete CPU und ein OLB-Symbol an die andere (in Betrieb befindliche) CPU zu senden. Die in Betrieb befindliche CPU entscheidet, dass sie ihren Betrieb fortsetzen muss und sendet jeweils ein IOY-Symbol an beide Router **14A**, **14B**.

– Ein Softwarefehler, der zum Auftreten einer Divergenz zwischen den CPUs **12** führt, wobei allerdings keine Fehlermeldungen ausgegeben werden. Dazu kann es nur kommen, wenn Software (welche auf den Prozessoren **20** läuft) bekannte, divergierende Daten verwendet, um den Zustand zu wechseln. Angenommen zum Beispiel jede CPU **12** hat eine unterschiedliche Seriennummer (die z. B. in einem Nur-Lese-Bereich bzw. in einem programmierbaren Nur-Lese-Bereich des Adressraums aufbewahrt ist). Die Seriennummer der CPU **12A** unterscheidet sich von jener der CPU **12B**. Wenn nun die Prozessoren die Seriennummer dazu verwenden, um die Folge der auszuführenden Befehle zu verändern (etwa durch Verzweigen, wenn die Seriennummer nach einem bestimmten Wert kommt), oder um den in einem Prozessorregister enthaltenen Wert zu modifizieren, so unterscheidet sich der gesamte "Zustand" der einzelnen CPUs **12** voneinander. In solchen Fällen kommen die Mechanismen zur Handhabung asymmetrischer Variablen (z. B. der Soft-Vote-Mechanismus) zur Anwendung. Dadurch ist es den beiden CPUs **12** möglich, Informationen auszutauschen, so dass diese basierend auf identischen Daten identische Entscheidungen treffen und identische Operationen ausführen können. In diesem Beispiel würden die CPUs die Werte ihrer Seriennummern austauschen, so dass beide über eine Kopie verfügen. Sie könnten dann Code ausführen für "Wenn die Seriennummer der CPU **12A** nach einem bestimmten Wert kommt, dann führe diese Sequenz aus", usw. Durch die Vorkonfiguration und die schlussendlich erfolgende Auswahl einer primären CPU wird es für eine CPU, und somit für das System **10** möglich, ohne Softwareintervention den Verarbeitungsbetrieb fortzusetzen.

– Ein Fehler am Ausgang der Schnittstelleneinheit **24** einer CPU **12** wird von dem Router **14A**, **14B**, je nachdem an bzw. auf welchem Link L der Fehler auftritt, als Divergenz erkannt, und ein TLB/OLB-Zustand wird im Anschluss an das DVRG-Symbol an die CPUs zurückgegeben. CRC-Fehler und Befehlssymbolkorruptionen werden in ähnlicher Weise erkannt und gehandhabt.

– Ein Fehler am Ausgang eines Routers **14A**, **14B**, welcher an eine CPU **12** angeschlossen ist, wird von der Schnittstelleneinheit **24** der betreffenden CPU erkannt. Die CPU sendet ein TLB-Symbol an den fehlerbehafteten Router **14**, welches von diesem Router als Divergenz erkannt wird, der daraufhin die Divergenzbehandlungsroutine aus **Fig.** 31 einleitet. Die CPU, die den Fehler erkannt hat, beendet ihren Betrieb, und die CPU, die ihren Betrieb fortsetzt, sendet IOY-Symbole an beide Router.

[0473] Die Divergenz ermöglicht es dem System **10** (**Fig.** 1A), wenn es im Duplex-Betriebsmodus betrieben wird, ein mögliches Versagen zu erkennen, und ohne äusseres Zutun und in für den Systembenutzer transparenter Weise die fehlerbehaftete Einheit (die CPU **12A** bzw. **12B** oder den Router **14A** bzw. **14B**) aus dem System zu entfernen, um die Ausbreitung von Fehlern auf das System zu verhindern bzw. zu begrenzen, und zwar ohne dass dadurch ein Benutzer seine Operationen zu stoppen braucht. Ist es eine CPU **12**, die von dem Fehler betroffen war, so kann die CPU aus dem System genommen und durch eine geprüfte, funktionierende CPU ersetzt werden. Wie dieses Ersatzelement in das System integriert und in Duplexbetrieb versetzt wird, soll Gegenstand des folgenden Abschnitts: Reintegration.

Reintegration:

Überblick:

[0474] Dieser Abschnitt beschreibt die Verfahrensweise, die angewendet wird, um zwei CPUs **12** durch "Reintegration" in im Gleichschritt laufenden Duplex-Betriebsmodus zu versetzen. Die Abhandlung bezieht sich auf die CPUs **12A**, **12B**, die Router **14A**, **14B**, und die Instandhaltungsprozessoren **18A**, **18B**, welche- wie gezeigt- Teil des in **Fig.** 1A veranschaulichten Verarbeitungssystems **10** sind. Darüber hinaus wird in der Abhandlung auf die Prozessoren **20a**, **20b**, die Schnittstelleneinheiten **24a**, **24b** und die Speichersteuerungen **26a**, **26b** (**Fig.** 2) der CPUs **12A**, **12B** als einzelne Einheiten Bezug genommen, da dies ihrer Funktionsweise entspricht.

[0475] Die Reintegration wird dazu verwendet, um zwei CPUs anlässlich ihrer erstmaligen Integration in das Leitungssystem, oder nachdem sie für eine gewisse Zeit im Simplexmodus betrieben worden sind, oder nachdem ein vorgängiger Duplexbetrieb des Systems **10** zu einer Divergenz geführt hat und das fehlerbehaftete Element (z. B. eine der CPUs) entfernt und ersetzt worden ist, in den Duplexbetrieb zu versetzten.

[0476] Die Reintegration muss mit einer der CPUs **12** beginnen, die sich noch in Betrieb (d. h.. in einem Online-Zustand) befindet und mit grösster Wahrscheinlichkeit Benutzeranwendungen ausführt, da die Reintegration im Hintergrund und ohne äusseres Zutun erfolgt und daher im wesentlichen in einer für den Benutzer transparenten Weise **1** abläuft. Die andere CPU **12** befindet sich insofern im Offline-Zustand als sie keinen Benutzercode ausführt; sie führt eine ausreichende Codemenge aus, welche es ihr ermöglicht, die erforderlichen Mindestaufgaben für ihre Initialisierung und Reintegration auszuführen. Diese Initialisierung beinhaltet das Versetzen eines CPU-Paars **12** in einen virtuell identischen Zustand für den Duplex-Betriebsmodus, so dass diese in der Lage sind, dieselben Befehle identischer Befehlsströme virtuell zur selben Zeit auszuführen und folglich dieselben Aktionen zu setzen. Die Reintegration hat auch zur Folge, dass die Router **14A**, **14B** für den Duplex-Betriebsmodus konfiguriert werden, so dass eine Divergenzerkennung implementiert werden kann und der für die CPUs **12** bestimmte Nachrichtenverkehr an beide der gepaarten CPUs virtuell zur selben Zeit ausgeliefert wird.

[0477] Als wesentliche Prozessschritte für den Wechsel von einem Simplex-Betriebsmodus der einen Online-CPU in den Duplex-Betriebsmodus zweier CPUs, welche in etwas detaillierterer Form durch die Flussdiagramme der **Fig.** 33A–33D skizziert sind, sind die folgenden zu nennen:

1. Das Einrichten und Synchronisieren der beiden CPUs (die eine online, die andere offline) und der an sie angeschlossenen Router in Form eines verzögerten ("schatten-")frequenzsynchronisierten Duplex-Betriebsmodus, wobei unterschiedliche Befehlsströme abgearbeitet werden;
2. Das Kopieren des Speichers der Online-CPU in die Offline-CPU, und zwar unter Wahrung eines Nachlaufprozesses, welcher Änderungen in dem Speicher der Online-CPU überwacht, die noch nicht erfolgt sind und unter Umständen noch in die Offline-CPU hinüberkopiert werden müssen;
3. Das Einrichten und das Synchronisieren der CPUs, so dass diese in einem verzögerten (untergeordneten) Duplexmodus betrieben werden, und zwar basierend auf demselben Befehlsstrom (im Gleichschritt laufender Betrieb);
4. Das Kopieren sämtlicher verbleibender Speicherorte von der Online-CPU in die Offline-CPU (In diesem Schritt wird jeder Speicherort des Online-Speichers gelesen, es werden jedoch nur jene Speicherorte kopiert, von denen anzunehmen ist, dass sie sich von jenen der Offline-CPU unterscheiden, und zwar so lange bis der gesamte Speicher ausgelesen worden ist.); und
5. Das Einleiten eines vollständig im Gleichschritt laufenden Duplexbetriebs der beiden CPUs.

Setup

[0478] Im folgenden wird auf **Fig.** 33A Bezug genommen, wobei bevor das Reintegrationsverfahrert an sich in Angriff genommen wird, die CPUs **12A**, **12B** und ihre Primärleitungsrouter (d. h. jene, die unmittelbar an die CPUs angeschlossen sind) eingerichtet werden müssen. Dies macht die Verwendung des IP **18A** erforderlich. Bei Schritt 1050 schreibt der IP **18A** die vorherbestimmten Register (nicht dargestellt) der Steuerregister **74** in den Schnittstelleneinheiten **24** der CPUs **12A** und **12B** in einen nächsten Zustand (nach einer Soft-Operation), in welchem, obwohl sich die beiden CPUs in einem frequenzsynchronisierten Modus befinden, eine (die Offline-CPU) in verzögerter Weise, bzw. im "Schatten" der anderen betrieben wird und eine Anzahl von (z. B. **8**) Taktzyklen hinter der anderen nachgeht. Dieser Betriebsmodus der CPUs und der Routen wird im folgenden als "Schattenmodus" bezeichnet. Die Konfigurationsregister (nicht dargestellt) der Router werden in Schritt 1052 in ähnlicher Weise von dem IP 18A gesetzt. Ausserdem wird von dem IP **18A** ein Konfigurationsregister entsprechend beschrieben, um die Online-CPU **12A** dem Router **14A**, **14B** gegenüber als "bevorzugte" CPU zu kennzeichnen. Das hat zur Folge, dass die Router **14A**, **14B**, wenn sie sich im Schattenmodus befinden, sich ausschliesslich auf die CPU **12A** hin orientieren, was den Empfang von Übertragungen betrifft und alle Übertragungen, welche von der Offline-CPU **12B** kommen, ignoriert.

[0479] Als nächstes folgt eine Reihe von Schritten (Schritte 1060–1070), in denen die Taktsynchronisierungs-FIFOs der CPUs **12A**, **12B** und der Router **14A**, **14B** in im grossen und ganzen derselben Weise wie weiter oben beschrieben synchronisiert werden und daraufhin in einen Schattenbetriebszustand versetzt werden. Der Schattenbetriebszustand ist ein Zustand, in welchem die beiden CPUs **12A**, **12B** dieselben Nachrichtenpakete und anderen TNet-Symbole von den Routern **14A**, **14B** empfangen, in mehr oder weniger derselben Art, wie dies auch beim echten Duplex-Betriebsmodus der Fall ist, ausser dass Übertragungen, die an die Offline-CPU **12B** gesendet werden, Symbol für Symbol um eine Anzahl von (z. B. acht) S_Tkt-Takten hinter jenen liegen, die an die Online-CPU **12A** gesendet werden. Das bedeutet, dass ein Symbol, das von einem der Router **14A**, **14B** übertragen wird, von der Online-CPU **12A** um acht S_Tkt-Takte früher empfangen werden als dasselbe Symbol von der Offline-CPU **12B** empfangen wird.

[0480] Die Schritte 1060 und 1062 führen im wesentlichen dieselben Schritte des Synchronisierens der Taktsynchronisierungs-FIFOs aus, die weiter oben in Verbindung mit der Abhandlung der **Fig.** 31A, 31B beschrieben worden sind. Die Online-CPU **12A** sendet eine Sequenz bestehend aus einem SCHLAFEN-Symbol; einem eigenadressierten Nachrichtenpaket und einem SYNC-Symbol, welches zusammen mit dem SYNC-TKT-Signal dazu dient, die CPUs und die Router zu synchronisieren. Nachdem sie auf diese Weise synchronisiert ist, sendet die CPU **12A** dann in Schritt 1066 ein Soft-Reset-Befehlssymbol (SRST-Symbol), welches bewirkt, dass als erstes die Router in den nächsten Zustand gebracht werden, der in Schritt 1052 von dem IP 18A festgelegt wurde. Die Router **14A**, **14B** befinden sich nun in verzögertem Duplexmodus, so dass der gesamte Datenverkehr, der zu der Online-CPU **12A** geleitet wird, dupliziert und auch an die Offline-CPU **12B** geleitet wird, allerdings um 8 Takte nach hinten versetzt in Bezug auf die Online-CPU **12A**. Ausserdem werden die Router **14A**, **14B** entsprechend eingestellt, so dass sie sich ausschliesslich auf die CPU **12A** hin orientieren, was den Empfang von Nachrichtenpaketen betrifft und alle Übertragungen, welche von der Offline-CPU **12B** kommen, ignoriert Das SRST-Symbol wird an die CPUs **12A**, **12B** zurück reflektiert (wobei das SRST-Symbol um **8** Takte später bei der Offline-CPU **12B** eintrifft). Mit dessen Empfang durch die CPUs **12A**, **12B** werden die CPUs in den nächsten von dem IP **18A** festgelegten Betriebszustand gesetzt: den Schattenmodus.

[0481] Die Prozedur geht nun zu Schritt 1080 über (**Fig.** 33B), um die Überwachung von Speicher und Zustand einzurichten (z. B. Register, Cachespeicher, etc.), welche durchgeführt wird, während der Speicher von der Online-CPU **12A** in die Offline-CPU **12B** hinüberkopiert wird. Der Schritt des Kopierens des Zustands der Online-CPU in die Offline-CPU könnte einfach dadurch erfolgen, dass alle laufenden Operationen der Online-CPU gestoppt werden, dass der Zustand aller Konfigurationsregister und Steuerregister (z. B. die Konfigurationsregister **74** der Schnittstelleneinheiten **24**), der Cachespeicher und dergleichen in den Speicher **28** der Online-CPU geschrieben werden, dass der gesamte Inhalt des Speichers **28** der Online-CPU auf die Offline-CPU hinüberkopiert wird und dass beide CPUs einer Reset-Routine unterzogen werden, um sie beide wieder in Betrieb zu nehmen. Bei grossen Systemen könnte die Fertigstellung dieses Vorgangs jedoch eine Zeitspanne in Anspruch nehmen, die in einem Bereich von mehreren zehn Sekunden liegen würde, was einen unannehmbar langen Zeitraum darstellen würde, in dem das System zu Reintegrationszwecken offline wäre. Aus diesem Grund wird der Reintegrationsprozess in einer Art und Weise durchgeführt, die es der Online-CPU ermöglicht, mit der Ausführung von Benutzer-Anwendungscode fortzufahren, während der das Hinüberkopieren des Zustands auf die Offline-CPU grösstenteils im Hintergrund abläuft.

[0482] Da allerdings die Online-CPU mit der Ausführung von Benutzer-Anwendungscode fortfährt, ist die Wahrscheinlichkeit gross, dass während des Kopierens des Zustands in die Offline-CPU Bereiche des Speichers **28** verändert werden, nachdem sie kopiert worden sind, so dass nach Abschluss eines ersten Arbeitsgangs des Kopierens des Speichers der Speicher der Online-CPU nicht identisch ist mit jenem der Offline-CPU. Der Grund dafür liegt darin, dass die normale Verarbeitung durch den Prozessor **20** der Online-CPU den Speicherinhalt verändern kann, nachdem er bereits in die Offline-CPU hinüberkopiert worden ist. (Im Speicher **28** der Online-CPU **12A** vorgenommene E/A-Schreiboperationen haben keine dahingehenden Auswirkungen auf den Kopiervorgang dass dadurch der Inhalt der Speicher der CPUs **12A**, **12B** inkonsistent würde, da sie ja auch in der Offline-CPU vorgenommen werden.) Zwei Mechanismen werden verwendet, um diesem Problem während der Reintegration zu begegnen: Erstens, wenn ein Speicherort in der Online-CPU während des Reintegrationsprozesses beschrieben wird, so wird er als "unsauber" markiert; zweitens, jeder Vorgang des Kopierens von Speicher in die Offline-CPU wird durch einen "AtomicWrite"-Mechanismus durchgeführt, der die Speicherorte überwacht, die gerade kopiert werden, um sie vor einem Überschreiben durch Daten aus der Online-CPU zu schützen, bevor der Kopiervorgang bestätigt ist (wodurch, wie noch zu zeigen sein wird, der kopierte Speicherort überschrieben wird). Beide Mechanismen werden nur während der Reintegrationsprozedur verwendet, und ein Verständnis der beiden trägt zu einem besseren Verständnis des Speicher-Vorabkopiervorgangs und des darauf folgenden Kopierens des Zustands von der Online-CPU in die Offline-CPU bei.

Speichermarkierung

[0483] Das Markieren von Speicherbereichen als "unsauber" erfolgt über die Verwendung eines der FKC-Prüfbits, das spezifisch diesem Zweck dient. Es sei hier daran erinnert, dass 64-Bit-Datenwörter mit 8 FKC-Bits gespeichert werden, um eine Einzelbitfehlerkorrektur und eine Doppelbitfehlererkennung bereitzustellen. Die Verwendung eines der **8** FKC-Bits hat keinerlei Einfluss auf die Einzelbitkorrekturfähigkeit. Sie kann allerdings das Erkennungsvermögen von Doppelbitfehlern einschränken. Da jedoch Speicherkopiervorgang nur eine relativ kurze Zeitzpanne andauert, wird dieses Risiko als annehmbar betrachtet.

[0484] Während der Reintegration wird mit jeder Schreiboperation an einen Speicherort durch die Online-CPU **12A** ein Bit des FKC-Prüfcodes invertiert, wodurch bewirkt wird, dass eine später erfolgende Leseoperation desselben Speicherortes ein vorgegebenes FKC-Syndrom hervorruft, das als Markierungszeichen für

einen unsauberen Speicherort interpretiert wird. (Es gibt zwei Ausnahmen: die erste sind Schreiboperationen von eingehenden E/A-Daten, da ja der Offline-Speicher ebenfalls mit denselben E/A-Daten beschrieben wird. Die zweite sind Schreiboperationen im Zusammenhang mit dem AtomicWrite-Mechanismus ("Write Conditional".) Auf diese Weise werden Veränderungen des Inhalts des Speichers **28** der Online-CPU **12A**, die eventuell nicht in den Speicher **28** der Offline-CPU **12B** mit übernommen worden sind, gekennzeichnet und in der Folge durch Kopieren dieser Speicherorte in den Speicher **28** der Offline-CPU **12B"** sauber" gemacht.
[0485] Welches der FKC-Bits für diese Markierungszwecke verwendet wird, ist weniger von Bedeutung, solange gewährleistet ist, dass immer dasselbe Bit verwendet wird.

AtomicWrite-Mechanismus

[0486] Das Hinüberkopieren des Zustands der Online-CPU **12A** in die Offline-CPU **128** macht die Verwendung von Nachrichtenpaketübertragungen auf der TNet-Struktur über einen der Router **14A**, **14B** erforderlich. Da aber die Router **14A**, **14B** für den Duplex-Betriebsmodus eingerichtet sind, so dass E/A-Schreiboperationen in den Speicher **28** der Online-CPU **12A** zugleich auch in den Speicher der Offline-CPU **12B** geschrieben werden, wird die Übertragung von Nachrichtenpaketen, welche den Online-Zustand enthalten, in ähnlicher Weise an beide CPUs **12A**, **128** übertragen, jedoch auf sinnvolle Weise nutzbar gemacht: der Empfang des den Zustand enthaltenden Nachrichtenpakets durch die Online-CPU **12A** bestätigt dessen fehlerfreien Empfang und Weiterübermittlung durch den Router **14** (**14A** bzw. **14B**, je nachdem welcher benutzt wurde). Überdies wird durch die Schreiboperation des AtomicWrite-Mechanismus als sauber markiert. Somit werden die markierten Speicherorte erst dann als sauber markiert, wenn die Daten, welche sie enthielten, in dem Nachrichtenpaket, welches sie zur Offline-CPU beförderte, zurückerhalten werden und in den Speicher geschrieben werden, von dem Sie kamen. Auf diese Weise erfolgt eine Bestätigung, dass die Speicherorte erfolgreich in den Speicher der Offline-CPU kopiert worden sind.
[0487] Angenommen jedoch es erfolgt zwischen dem Auslesen eines Speicherorts der Online-CPU **12A** zum Zweck des Hinüberkopierens in die Offline-CPU **12B** und dem darauffolgenden Zurückschreiben der kopierten Daten in den Online-Speicher eine andere Schreiboperation in den betreffenden Speicherort in der CPU **12A** (entweder eine eingehende E/A-Schreiboperation oder eine Prozessor-Schreiboperation). Die zurückkehrenden Daten (die in die Offline-CPU hinüberkopiert worden waren) würden an den Speicherort geschrieben, von dem sie ursprünglich gekommen waren, und somit alle neuen Werte, die der Speicherort inzwischen enthielt, überschreiben, den Speicherort als sauber markieren, und Daten zerstören, die unter Umständen für den fortlaufenden Betrieb der Online-CPU **12A** benötigt werden. Um diesem Problem zu begegnen, wurde der AtomicWrite-Mechanismus geschaffen.
[0488] Der AtomicWrite-Mechanismus benutzt die Speichersteuerung **26** (**Fig.** 2) der Online-CPU, um Speicherorte zu überwachen, die gerade dabei sind, in die Offline-CPU **12B** hinüberkopiert zu werden. Die Speichersteuerung verwendet einen Satz von Registern und eine Kontroll-Ablaufsteuereinheit, die während der Reintegration in Betrieb genommen werden, um in diesen Registern die Adressen jener Speicherorte nachzuverfolgen, die zum Zweck des Hinüberkopierens in die Offline-CPU **12B** ausgelesen worden sind, die jedoch noch nicht mit den zurückgegebenen Daten beschrieben worden sind, um sie als sauer zu markieren. Sollte es dazwischen zu einer Schreiboperation von Daten (d. h.. anderen Daten als denen, die zurückgegeben werden) kommen an einem auf der Liste befindlichen Speicherort kommen, bevor die kopierten Daten wieder an die CPU zurückgekehrt sind, so wird die Liste entsprechend gekennzeichnet. Wenn die kopierten Daten an die Online-CPU zurückgegeben werden, wird diese Liste geprüft, bevor die Daten in den Speicher geschrieben werden. Falls Speicherorte in der Zwischenzeit als mit neuen Daten beschrieben gekennzeichnet sind, werden die zurückgegebenen Daten ausgeschieden und die ausgelassenen Speicherorte als unsauber markiert. Wenn andererseits die Speicherorte seit ihrem Auslesen zum Zweck des Hinüberkopierens in die Offline-CPU nicht beschrieben worden sind, werden die zurückgegebenen, kopierten Daten in die Speicherorte zurückgeschrieben und diese Speicherorte als sauber markiert.
[0489] Im wesentlichen verwendet der AtomicWrite-Mechanismus zwei Operationen: eine "ReadLinked" Speicheroperation und eine "WriteConditional" Operation. Die ReadLinked Speicheroperation dient dazu, um die Speicherorte des Speichers **28** der Online-CPU **12A**, die in die Offline-CPU **12B** hinüberkopiert werden sollen, die Adresse dieser Operation in einer Link-Tabelle zu speichern, die von der SpSt **26** aufrechterhalten wird, und die zu kopierenden Daten in einer Warteschlange der BÜM **88** zu installieren, wo Sie zusammengefügt und als Nachrichtenpaket an die Offline-CPU (sowie auch an die Online-CPU) gesendet werden. Durch das Speichern der Adresse der ReadLinked Operation wird diese mit einer zukünftigen WriteConditional Operation verbunden, welche die Daten an den Speicherort zurückgibt und den entsprechenden Eintrag in der Link-Tabelle löscht.
[0490] Typischerweise handelt es sich bei der Operation um eine herkömmliche Blockleseoperation, bei der ein Block von Daten aus einer Anzahl von Speicherorten gebildet wird. Die Adresse, die in die Link-Tabelle geschrieben wird, ist jene des am Beginn oder am Ende des Blocks von Speicherorten befindlichen Speicherorts.

Wenn das Datenpaket, welches Daten enthält, die aus einem Block von kopierten Speicherorten ausgelesen wurden, von der Online-CPU **12A** zurückerhalten wird, werden sie mittels einer WriteConditional-Operation in den Speicher **28** geschrieben. Bevor jedoch die Daten zurückgeschrieben werden, überprüft die SpSt **26** die Link-Tabelle. Wenn der Speicherort innerhalb des Blocks in der Zwischenzeit durch eine andere Operation (z. B. eine Schreiboperation durch den Prozessor **20**, eine E/A-Schreiboperation, etc.) beschrieben worden ist, so kennzeichnet diese frühere Schreiboperation den Speicherort in der Link-Tabelle mit einem Flag (und markiert darüber hinaus die beschriebenen Speicherorte als unsauber). Die SpSt **26** bemerkt das Flag und scheidet die WriteConditional Daten aus, ohne sie zu schreiben, wodurch die betreffenden Speicherorte als unsauber markiert belassen werden, was anzeigt, dass sie noch in die Offline-CPU **12B** hinüberkopiert werden müssen.

[0491] Im folgenden wird wieder auf den Reintegrationsprozess Bezug genommen, und zwar auf **Fig.** 33B, worin die Speicherrückverfolgung (der AtomicWrite-Mechanismus und die Verwendung des FKC zur Markierung von Speicherorten) in den Schritten 1080 und 1082 aktiviert werden. Dies beinhaltet das Beschreiben eines Reintegrationsregisters (nicht dargestellt; eines der Konfigurationsregister **74** der Schnittstelleneinheit **24**- **Fig.** 5), um zu bewirken, dass ein Reintegrationssignal (REINT-Signal) ausgegeben wird. Das REINT-Signal wird an die FKC-Logik **85** einer jeden Speicherschnittstelle **70** (**Fig.** 10) gekoppelt, um das eine der acht FKC-Bits, die von der FKC-Logik **85** für alle Schreiboperationen ausser WriteConditional-Operationen und für alle E/A-Schreiboperationen erzeugt werden, zu invertieren, so dass beim darauffolgenden Lesen die Daten mit diesem invertierten Bit ein Syndrom erzeugen, welches die Speicherorte als unsauber markiert identifiziert.

[0492] Nachdem die Speichernachverfolgung auf diese Weise aktiviert worden ist, springt die Reintegrationsprozedur zu der "Vorabkopier"-Sequenz (Schritte 1084–1088), in welcher Inhalte des Online-Speichers **28** in einem ersten Durchgang in den Speicher der Offline-CPU **12B** hinüberkopiert werden, und zwar von unten nach oben (oder von oben nach unten, falls gewünscht) (Schritt 1084). Speicherorte, welche später durch Schreiboperationen beschrieben werden, die nicht von eingehenden E/A-Daten und dem AtomicWrite-Mechanismus herrühren, verwenden das FKC-Bit, um den beschriebenen Speicherort (bzw. die Speicherorte, je nach Sachlage) als unsauber zu kennzeichnen. Schreiboperationen, die an einem Speicherort erfolgen, nachdem dieser durch eine ReadLinked-Operation kopiert worden ist, jedoch bevor eine darauffolgende WriteConditional Operation durchgeführt wird, ebenfalls markiert wird.

[0493] Nachdem der gesamte Inhalt des Speichers **28** einmal durchlaufen und in den Offline-Speicher hinüberkopiert worden ist, springt die Sequenz zu den Schritten 1086 und 1088, um nun im Zuge eines inkrementellen Kopiervorgangs all jene Online-Speicherorte zu kopieren, welche verglichen mit jenen des Offline-Speichers inkonsistent sind, d. h. jene Speicherorte, die bei Abschluss des Schrittes 1084 noch als unsauber markiert verbleiben. Das inkrementelle Kopieren umfasst mehrere Durchgänge, bei denen jeweils der gesamte Online-Speicher durchlaufen wird und jeder Speicherort gelesen wird, um das resultierende Syndrom dahingehend zu prüfen, ob der so markierte Speicherort unsauber oder sauber ist. Ist er als unsauber markiert, so wird der Speicherort in die Offline-CPU kopiert und als sauber markiert. Wenn der Speicherort als sauber markiert ist, wird er nicht angerührt. Während der Durchführung der inkrementellen Kopiervorgänge markieren die normalen Aktionen des Online-Prozessors einige Speicherorte als unsauber.

[0494] Mehrere Durchgänge des inkrementellen Kopierens müssen unter Schritt 1052 vollzogen werden bis ein Punkt erreicht ist, bei welchem die Rate mit der unsaubere Speicherorte kopiert und gesäubert werden, im wesentlichen jener Rate entspricht, mit der Speicherplatz als unsauber markiert wird. Zu diesem Zweck sind in der SpSt **26** jeweils spezielle Zähler für ReadLinked-, WriteConditional-, fehlgeschlagene ReadLinked-, und fehlgeschlagene WriteConditional-Operationen integriert. Durch Notieren der Anzahl der erfolgreichen WriteConditional-Operationen am Ende eines jeden Durchgangs durch den Speicher ist der Prozessor **20** in der Lage, den Wirkungsgrad eines gegebenen Durchgangs verglichen mit jenem des vorausgegangenen Durchgangs zu bestimmen. Wenn der Nutzen nachlässt, beenden die Prozessoren **20** die Vorabkopieroperationen. An diesem Punkt ist der Reintegrationsprozess bereit, die beiden CPUs **12A**, **12B** in im Gleichschritt laufenden Betrieb zu setzen.

[0495] Damit springt die Reintegrationsprozedur zu der in **Fig.** 33C veranschaulichten Sequenz von Schritten, wobei in Schritt 1100 die Online-CPU **12A** kurzzeitig die Vordergrundverarbeitung, d. h. die Ausführung einer Benutzeranwendung, stoppt Der verbleibende Zustand (z. B. Konfigurationsregister, Cachespeicher, usw.) des Online-Prozessors **20** und seiner Cachespeicher wird dann gelesen und in einen Puffer (eine Reihe von Speicherorten) in dem Speicher **28** geschrieben (Schritt 1102). Dieser Zustand wird dann in die Offline-CPU **12B** hinüberkopiert, gemeinsam mit einem "Reset-Vektor", der die Prozessoreinheiten **20** beider CPUs **12A**, **12B** zu einem Reset-Befehl leitet.

[0496] Als nächstes werden in Schritt 1106 die Router **14A**, **14B** durch ein SCHLAFEN-Symbol in einen Ruhezustand versetzt, gefolgt von einem eigenadressierten Nachrichtenpaket, um sicherzustellen, dass die FIFOs der Router gelöscht sind, dass die FIFOs der Prozessorschnittstellen **24** gelöscht sind und dass keine weiteren eingehenden E/A-Daten anstehen. In Schritt 1108 überträgt die Online-CPU **12A** ein SRST-Befehlssymbol an die Router **14A**, **14B**, welche das SRST-Symbol an beide CPUs **12A**, **12B** zurückreflektieren. Da der zurückreflektierende Router nach wie vor in dem weiter oben beschriebenen, untergeordneten Duplexmo-

dus betrieben wird, kommt das an die Offline-CPU **12B** zurückreflektiere SRST immer noch 8 Takte nach jenem an, das an die Online-CPU **12A** zurückreflektiert wird. Das zurückreflektierte SRST-Symbol wird von beiden CPUs **12A**, **12B** empfangen und beide CPUs reagieren darauf, indem sie die Prozessoreinheiten **20** einer jeden CPU dazu veranlassen, zu jenem Ort im Speicher **28** zu springen, welcher den Reset-Vektor enthält, und eine Subroutine einzuleiten, welche den gespeicherten Zustand beider CPUs **12A**, **12B** in den Prozessoreinheiten **20**, den Cachespeichern **22**, den Registern, etc. wiederherstellt. Die CPUs **12A**, **12B** beginnen daraufhin, denselben Befehlsstrom auszuführen.

[0497] Somit befinden sich die CPUs **12A**, **12B** bei Schritt 1112 immer noch im Schattenbetriebsmodus, d. h. die beiden arbeiten zwar denselben Befehlsstrom ab, die CPU **12B** liegt dabei jedoch um 8 Taktzyklen hinter der CPU **12A** und die Router sind nach wie vor so konfiguriert, dass sie von der CPU **12B** kommende Übertragungen ignorieren. Die CPU **12A** kehrt in den Online-Zustand zurück und fährt mit der Ausführung der Benutzeranwendung fort.

[0498] Die Reintegrationsprozedur tritt nun in die letzten Reintegrationsetappen ein, welche auch als "Träge Reintegration" bezeichnet und in **Fig.** 33D veranschaulicht werden. Die Freigabe des FKC-Bits zum Markieren unsauberer Speicherorte muss nun deaktiviert werden, da die Prozessoren mit demselben Speicher dasselbe machen. wenn die Online-CPU **12A** während dieser Etappe der Reintegrationsprozedur beim Auslesen des Speichers im Zuge der Ausführung von Befehlen (welche die Offline-CPU ebenfalls ausführt-wenn auch mit einer Verzögerung von 8 Takten) auf einen Speicherort im Speicher **28** trifft, der als unsauber gekennzeichnet ist, so leitet sie die Ausgabe eines "Busfehlers" ein (Schritt 1120). Eine Meldung dieses Busfehlers wird unter Verwendung der Auswahllogik **920** des "Soft-Flag"-Logikelements 900 (**Fig.** 30B) an die CPU **12B** übertragen (Schritt 1122), um in der Offline-CPU **12B** für denselben Befehl einen Busfehler zu erzwingen. Unter vorübergehender Bezugnahme auf **Fig.** 30B lässt sich zeigen, dass durch die Ausgabe von REINT über den MUX **914** das BUSFEHLER-Signal zu Übergabe an die CPU **12B** ausgewählt wird, wodurch die CPU **12B** benachrichtigt wird, dass die CPU **12A** auf einen Busfehler gestossen ist.

[0499] Unterdessen bewirkt der Busfehler in der CPU **12A**, dass die Prozessoreinheit **20** in eine Fehlerbehandlungsroutine versetzt wird, um zu bestimmen (1) was die Ursache des Fehlers ist, und (2) wie, sofern möglich, mit dem Fehler umzugehen ist. In diesem Augenblick wird ermittelt, dass der Fehler durch den Versuch ausgelöst wurde, einen als unsauber gekennzeichneten Speicherort auszulesen. Demgemäss leitet die Prozessoreinheit **20** (über die BÜM **88**-**Fig.** 5) den AtomicWrite-Mechanismus ein, um den Inhalt des Speicherorts in die CPU **12B** hinüberzukopieren. Die CPU **12A** führt daraufhin den Befehl, welcher den Busfehler verursachte, erneut aus, und fährt in der Verarbeitung fort.

[0500] Bei der CPU **12B**, welche mit einem Versatz von **8** Takten hinter der CPU **12A** betrieben wird, wird ebenfalls ein Busfehler erzwungen, und zwar durch die Übermittlung dieses Fehlers von der CPU **12A** über den Bus **902**, welche vor der Ausführung desselben Befehls erfolgt, welcher den Busfehler bei der CPU **12A** verursacht hatte. Zu dem Zeitpunkt, da die CPU **12B** diesen Befehl ausführt, ist jedoch die Busfehlermeldung bereits an die CPU **12B** übermittelt worden und wird mit demselben, **8** Takte später erfolgenden Befehl der CPU **12B** in Beziehung gesetzt. Dieses In-Beziehung-setzen wird dadurch erreicht, dass die Verzögerung bei der Übertragung des Busfehlersignals von der Online-CPU **12A** zu der Offline-CPU **128** mit der acht-Takte-Verzögerung zur Übereinstimmung gebracht wird, welche durch die Routerübertragungen an die CPUs bewirkt werden (d. h. mit der acht-Takte-Verzögerung des Schattenbetriebsmodus). Die CPU **12B** wird dabei dennoch gezwungen, dieselbe Busfehlerbehandlungsroutine zu durchlaufen, wie sie von der CPU **12A** eingeleitet wurde. Um in synchronem, im Gleichschritt laufendem Betrieb zu bleiben, führt die Offline-CPU **12B** exakt dieselbe Sequenz an Operationen aus wie die Online-CPU **12A**, und zwar einschliesslich der Busfehlerroutine und der Übertragung der Daten von "unsauberen" Speicherorten an den Router. Wohlgemerkt, die Router ignorieren zwar von der CPU **12B** kommende Übertragungen, es muss jedoch gewährleistet sein, dass die CPU **12B** zur Ausführung derselben Operation wie sie von der CPU **12A** ausgeführt wird, auch dieselbe Zeitspanne wie diese benötigt.

[0501] In der Zwischenzeit verwendet die Online-CPU **12A** eine gewisse Zeit darauf, um einen letzten Durchgang durch den gesamten Speicher der CPU **12A** zu vollziehen, wobei jene Speicherorte, die immer noch als unsauber gekennzeichnet sind, hinüberzukopieren, während gleichzeitig die Ausführung eines Benutzer-Anwendungsprogramms fortgesetzt wird. Während der letzten Etappe der Reintegration wird der gesamte Speicher ausgelesen und dabei jeder Speicherort überprüft. Jeder Speicherort, der geprüft und als unsauber markiert erkannt wird, wird in die Offline-CPU, d. h. die CPU **12B** kopiert. Schlussendlich sind die Zustände der CPUs **12A**, **12B** konsistent, so dass die beiden CPUs in einen echten, nicht-verzögerten, im Gleichschritt laufenden Betrieb gesetzt werden können.

[0502] Nachdem somit dank dem Schritt 1124 feststeht, dass tatsächlich der gesamte Speicher überprüft, und gegebenenfalls kopiert, worden ist, setzt der IP 18 in Schritt 1128 die Router **14A** in einen nächsten Duplex-Betriebszustand, indem er die in der Steuerlogik **509** enthaltenen Konfigurationsregister entsprechend beschreibt. Die CPU **12A** gibt daraufhin, wie bereits zuvor, die Sequenz bestehend aus SCHLAFEN-Symbolen und eigenadressiertem Nachrichtenpaket aus. Wenn die CPU **12A** sichergestellt hat, dass der Router sich in

einem Ruhezustand befindet, sendet die CPU **12A** an beider Router **14A**, **14B** (gleichzeitig) ein SRST-Symbol. Durch den Empfang dieses Symbols werden die Router **14A 14B** in den Duplexbetrieb versetzt, so dass wenn sie das SRST-Symbol an die CPUs **12A**, **12B** zurückreflektieren, beide Symbole gleichzeitig zurückreflektiert werden. Wenn die SRST-Symbole nun von den CPUs **12A**, **12B** empfangen werden, so veranlassen sie beide Prozessoreinheiten **20** der CPUs, zurückgesetzt zu werden und ihren Betrieb von demselben Speicherort in demselben Zustand und zur selben virtuellen Zeit aufzunehmen. Die CPUs **12A**, **12B** befinden sich nun in im Gleichschritt laufendem Betrieb.

Zusätzliche Merkmale

Kostenreduziertes Duplexsystem

[0503] Im folgenden wird vorübergehend auf **Fig.** 1 Bezug genommen, wobei die CPUs **12A**, **12B** wie erwähnt entweder getrennt oder als duplexbetriebenes Paar verwendet werden. Im ersteren Fall schafft die in dem Entwurf einer jeden der unabhängig voneinander betriebenen CPUs verwendete Redundanz eine schnellfehlende Architektur. CPUs können gepaart (nicht im Duplexmodus) betrieben werden, so dass eine CPU als "primäre" CPU und die andere CPU als "sekundäre" CPU bezeichnet wird, wodurch ein softwarebezogener Ansatz zur Fehlertoleranz implementiert wird. Somit hat die sekundäre CPU eine Benutzer-Anwendung zur Verfügung, welche auf der primären CPU läuft, und die primäre CPU aktualisiert in regelmässigen Abständen z. B. eine Datenbank oder eine Auditdatei, welche Aufschluss gibt über die bis zu dem Zeitpunkt der Aktualisierung erfolgten Verarbeitungsaktivitäten der primären CPU. Sollte die primäre CPU ausfallen, so aktiviert die sekundäre CPU die Datensicherungsanwendung und übernimmt die Aufgabe der ausgefallenen CPU, und zwar von dem Zeitpunkt an, da die Datenbank bzw. die Auditdatei zum letzten Mal aktualisiert wurde. Darauf basiert der softwarebezogene Ansatz zur Fehlertoleranz.

[0504] Softwarebezogene Ansätze zur Fehlertoleranz werden charakteristischerweise durch Betriebssysteme implementiert. Für jene Betriebssysteme, die nicht so robust ausgelegt sind, und daher nicht über eine entsprechende Fähigkeit verfügen, wird der weiter oben (siehe **Fig.** 1) beschriebene Duplex-Betriebsmodus mit zwei CPUs **12** vorgestellt, welche betrieben werden, um identische Befehle ein und desselben Befehlsstroms auszuführen. In **Fig.** 34 wird ein kostenreduziertes, duplexbetriebenes Paar von CPUs gezeigt, wobei eine der CPUs nicht über dieselbe Redundanz verfügt wie die andere.

[0505] Allerdings sei, unter Bezugnahme auf **Fig.** 1, darauf hingewiesen, dass die CPU **12A** in fehlerprüfender Redundanz zu der CPU **12B** betrieben werden kann, und zwar in derselben Weise, in der die einzelnen Prozessoreinheiten **20a**, **20b** der CPU **12A** eine schnellfehlende Fehlertoleranz für die CPU bereitstellen-sofern beide im Duplexmodus und in im Gleichschritt laufendem Betrieb zueinander betrieben werden. Somit kann für den Duplexbetrieb, wie in **Fig.** 34 veranschaulicht, ein kostenreduziertes System angewendet werden. Wie in **Fig.** 34 gezeigt, beinhaltet ein Verarbeitungssystem **10'** die CPU **12A** und die Router **14A**, **14B** in der weiter oben beschriebenen Struktur. Die CPU, mit welcher die CPU **12A** gepaart ist, hier als CPU **12B'** dargestellt, ist jedoch als einprozessorbasiertes CPU-System strukturiert. Die Verbindungen zwischen den Routern **14A**, **14B** und den CPUs sind ebenfalls dieselben.

[0506] Somit umfasst die CPU **12B'** nur eine Einprozessoreinheit **20'** und dieser zugeordnete, unterstützende Komponenten, unter anderem den Cachespeicher **22'**, die Schnittstelleneinheit (SE) **24'**, die Speichersteuerung **26'** und den Speicher **28'**. Während die CPU **12A** in der in **Fig.** 2 gezeigten Art strukturiert ist, d. h. mit Redundanzen in den Bereichen der Cachespeichereinheit, der Schnittstelleneinheit und der Speichersteuerung, wird annähernd eine Hälfte dieser Komponenten zur Implementierung der CPU **12B'** benötigt.

[0507] Was ihren Betrieb betrifft, würden die CPUs **12A**, **12B'** im Duplexmodus betrieben, wobei eine jede im wesentlichen zur selben Zeit dieselben Befehle eines identischen Befehlsstroms ausführen. Die CPU **12A** von ihrem Entwurf her entsprechend ausgelegt, um durch die Verdoppelung der Prozessoreinheit **20** und anderer die CPU konstituierender Elemente einen schnellfehlenden Betrieb bereitzustellen. Darüber hinaus stellt die CPU **12A** durch den Duplexbetrieb und durch die von den Routern **14A**, **14B** vorgenommenen Divergenzprüfungen auch ein Prüfverfahren für ihre Partner-CPU, die CPU **12B'**, bereit. Die von dem CPU-Paar gebildete logische CPU stellt insofern eine funktionellfehlende Betriebsweise bereit, als für den Fall, dass von einem der Router **14A**, **14B** eine Divergenz erkannt werden sollte, und auf diese Divergenzerkennung in der oben beschriebenen Art durch Stoppen der fehlerbehafteten CPU reagiert wird, die in Betrieb verbleibende CPU mit der Ausführung der laufenden Anwendung fortfahren kann.

[0508] Handelt es sich bei der in Betrieb verbleibenden CPU um die CPU **12A**, so existiert immer noch ein kleines Mass an Datenintegrität dank der duplizierten Komponenten, aus denen die CPU **12A** gebildet ist. Sollte es sich bei der in Betrieb verbleibenden CPU um die CPU **128'** handeln, so muss allerdings eingeräumt werden, dass ausgenommen für die in konventioneller Weise implementierte Fehlerprüfungen (z. B. Paritätsprüfungen an verschiedenen Schnittstellen) eine Datenintegrität nicht gewährleistet ist.

[0509] Wie in **Fig.** 34 gezeigt, beinhaltet das Verarbeitungssystem **10'** ein Paar von Routern **14A**, **14B**, um

den Vergleich der von den CPUs **12A** und **12B'** kommenden Ausgabedaten durchzuführen. Für einschlägig gebildete Fachleute ist klar ersichtlich, dass nur ein Router **14** (z. B. der Router **14A**) benötigt wird, wenn nur eine Divergenzprüfung durchgeführt werden soll. Die Verwendung eines Routers könnte in der Praxis auch durch nichts weiter als eine blosse Vergleichsschaltung ersetzt werden, welche die erforderliche Divergenzprüfung vornimmt, vorausgesetzt die beiden Eingänge, die entsprechend geschaltet sind, um die Ausgangsdaten von den CPUs **12A**, **12B'** zu empfangen, verfügen über Taktsynchronisierungs-FIFOs der weiter oben beschriebenen Art, um den in gewisser Weise asynchronen Ausgabedaten zu empfangen, wobei diese empfangenen Ausgabedaten in synchroner Weise aus den FIFOs entnommen werden.

Standby-Redundanzprinzip

[0510] Im folgenden wird vorübergehend auf die **Fig.** 1A –1C Bezug genommen, wobei ein wichtiges Merkmal der Architektur des in diesen Figuren veranschaulichten Verarbeitungssystems darin besteht, dass jede CPU **12** die Dienste einer jeden E/A-Paket-Schnittstelle **16** und eines jeden mit einer solchen verbundenen E/A-Geräts für sich zur Verfügung hat, und zwar ohne dass dazu eine andere CPU in dem System benötigt wird. Zahlreiche Parallelverarbeitungssysteme nach dem Stand der Technik ermöglichen einen Zugriff auf E/A-Geräte bzw. die Inanspruchnahme ihrer Dienste nur auf dem Weg über eine Zuhilfenahme eines bestimmten Prozessors oder einer bestimmten CPU. In einem solchen Fall steht, sollte der für die Dienste eines E/A-Geräts verantwortliche Prozessor ausfallen, das betreffende E/A-Gerät für den Rest des Systems nicht mehr zur Verfügung. Andere Systeme nach dem Stand der Technik bieten Zugriff auf E/A-Einrichtungen über Prozessorpaare, so dass bei Ausfall eines Prozessors der Zugriff auf die entsprechende E/A-Einrichtung immer noch über den in Betrieb verbleibenden Prozessor verfügbar ist. Wenn natürlich beide Prozessoren ausfallen, so geht der Zugriff auf die E/A-Einrichtung erneut verloren.

[0511] Ausserdem stellt das Anfordern der Ressourcen eines Prozessors mit dem Ziel, diese einem anderen Prozessor eines parallelen Systems bzw. eines Mehrprozessorsystems bereitzustellen, eine Leistungseinbusse für das System dar.

[0512] Die Fähigkeit, einer jeden CPU eines Multiprozessorsystems Zugriff auf jedes Peripheriegerät zu gewähren, wie dies hier der Fall ist, trägt dazu bei, den "Primär"/"Datensicherungs"-Prozess auszuweiten, der in dem oben angegebenen U.S.-Patentschrift Nr. 4.228.496 offenbart wird. Dort kann ein Mehr-CPU-System über einen Primärprozess verfügen, welcher auf einer CPU läuft, während ein Datensicherungsprozess auf einer anderen der CPUs im Hintergrund abläuft. In regelmässigen Abständen führt der Primärprozess eine Operation des "Sicherungspunkt-Setzens" durch, im Zuge derer Daten betreffend die Operation des Prozesses an einem Speicherort gespeichert werden, auf welchen von dem Datensicherungsprozess aus zugegriffen werden kann. Wenn die CPU, auf welcher der Primärprozess durchgeführt wird, einen Fehler aufweist, so wird dieser Fehler von den verbleibenden CPUs, einschliesslich jener, auf welcher sich die Datensicherung befindet, erkannt. Diese Erkennung des CPU-Fehlers führt dazu, dass der Datensicherungsprozess aktiviert wird, und dass dieser auf die Sicherungspunkt-Daten zugreift, wodurch es der Datensicherung ermöglicht wird, von dem Punkt der letzten Sicherungspunkterstellungsoperation an den Betrieb des früheren Primärprozesses fortzuführen. Der Datensicherungsprozess wird nun zum Primärprozess und es wird eine andere CPU aus dem Pool der verbleibenden CPUs ausgewählt, um den Datensicherungsprozess zu übernehmen. Demgemäss kann das System rasch wieder in einen Zustand versetzt werden, in welchem ein weiterer Fehler toleriert werden kann, der sogar bereits auftreten kann bevor die Reparatur des ursprünglichen Fehlers (d. h. der fehlerhaften CPU) abgeschlossen ist.

[0513] Somit wird ersichtlich, dass durch das Verfahren und die Vorrichtung zur wechselseitigen Verbindung der verschiedenen Elemente eines Verarbeitungssystems **10** jede CPU Zugriff auf jedes E/A-Element des Systems sowie auf jede andere CPU des Systems erhält. Jede CPU kann auf jede E/A-Einrichtung zugreifen, ohne dazu die Dienste einen anderen Prozessors in Anspruch nehmen zu müssen. Dadurch wird die Systemleistung erhöht und im Vergleich zu jenen Systemen verbessert, die sehr wohl einen spezifischen Prozessor für die Realisierung des Zugriffs auf E/A-Einrichtungen benötigen.

[0514] Ausserdem ist zu bemerken, dass wenn eine CPU **12** einen Fehler aufweisen sollte oder in Offline-Betrieb gesetzt wird, dies keinerlei Auswirkungen auf den Zugriff irgendeiner anderen CPU auf irgendeine E/A-Einrichtung des Systems hat.

Transaktionsfolgeprotokoll und Barrieren-Transaktionen:

[0515] Wie weiter oben erklärt, beinhaltet das Headerfeld eines Pakets eine **4** Bit umfassendes 'Transaktionsfolgenummer'-Feld (TFN-Feld); siehe die **Fig.** 3A und 3B. Elemente des Verarbeitungssystems **10** (**Fig.** 19; die in der Lage sind, mehr als eine ausstehende Anforderung zu verwalten, wie zum Beispiel die CPUs **12** oder bestimmte E/A-Geräte, stellen für jede ausstehende Anforderung in dem TFN-Feld eine nur einmal vorkommende Folgenummer bereit. Wenn das Bestimmungselement ein Antwortpaket zu der spezifischen Anforde-

rung generiert, so enthält das TFN-Feld in dem Antwortpaket denselben TFN-Wert wie in dem Anforderungspaket, das zur Generierung der Antwort geführt hat. Das Systemelement, welches die Antwort empfängt, kann daraufhin die in der Antwort enthaltene TFN mit jenen der Anforderungen vergleichen, um zu bestimmen, welcher Anforderung die Antwort zuzuordnen ist.

[0516] Die TFN erlaubt es einem Systemelement, zu bestimmen, ob die Antwort einer Anforderung zuzuordnen ist, welche nicht mehr existiert. Zum Beispiel ist es bei manchen Systemen, wie auch bei dem vorliegenden, erforderlich, dass die Antworten auf eine Anforderung innerhalb einer vordefinierten Zeitspanne empfangen werden. Wird innerhalb der dafür eingeräumten Zeit keine Antwort empfangen, gibt das Systemelement, das die Anforderung initiierte, einfach eine zweite (wiederholte) Anforderung aus. Wird im Anschluss daran die Antwort auf die frühere Anforderung empfangen, so ist das Systemelement in der Lage, aufgrund der TFN zu bestimmen, welche Anforderung (die frühere, mittlerweile gelöschte Anforderung, oder die spätere gültige Anforderung) durch die Antwort beantwortet wird. Handelt es sich um die frühere, so wird die Antwort ausgeschieden.

[0517] TFNs sind auch hilfreich bei der Handhabung des sogenannten Problems der "veralteten Pakete". Bei Auftreten eines Fehlers ist es möglich, dass gerade in Übertragung befindliche Nachrichtenpakete irgendwo im Netzwerk blockiert werden. Ist keine Möglichkeit vorgesehen, diese veralteten Nachrichtenpakete zu entfernen, so können sich diese möglicherweise später wieder bemerkbar machen, nachdem das System im Anschluss an einen Fehler wiederhergestellt ist, und dessen Betrieb unter Umständen stören. Eine TFN eines empfangenen Nachrichtenpakets ermöglicht es dem Empfänger, durch Vergleichen der in der Antwort enthaltenen TFN mit der TFN des Nachrichtenpakets, das zur Beantwortung aufgefordert hat, zu bestimmen, ob es sich um eine aktuelle Antwort handelt oder nicht.

[0518] Bei Verwendung eines kleinen TFN-Feldes besteht die Möglichkeit, dass eine veraltete, spät eintreffende Antwort eine TFN trägt, welche mit der einer aktuell ausstehenden Anforderung übereinstimmt. Ein grosses TFN-Feld hat hingegen zur Folge, dass entweder jedes der zu übertragenden Nachrichtenpakete um vieles grösser wird oder aber dass die verfügbaren Datenfelder entsprechend verringert werden.

[0519] Die vorliegende Erfindung löst dieses Problem durch einen Mechanismus, welcher als "Barrieren-Transaktion" bezeichnet wird. TFNs werden zwar auch hierbei verwendet, jedoch verringert der Barrieren-Transaktionsmechanismus die erforderliche Grösse einer TFN auf ein nur vier Bits umfassendes Feld.

[0520] Kurz gesagt, wird eine Barrieren-Transaktion dazu verwendet, um die Integrität eines Übertragungsleitwegs zwischen einem sendenden Knoten und einem empfangenden Knoten zu prüfen. Eine Barrieren-Transaktion wird primär durch eine CPU eingeleitet, obwohl sie auch von einer E/A-Schnittstelle **16** ausgegeben werden könnte. Sie wird in erster Linie dann eingesetzt, wenn eine erwartete Antwort auf ein zuvor an eine E/A-Einheit **17** oder eine CPU **12** ausgegebenes Anforderungs-Nachrichtenpaket nicht innerhalb einer vordefinierten und dafür zugemessenen Zeitspanne empfangen wird. Eine CPU **12** hat die Möglichkeit, den Leitweg zu überprüfen, indem sie ein Barrieren-Transaktions-Nachrichtenpaket in Form eines HADC-Pakets (**Fig.** 3A) erstellt und sendet, welches die üblichen Header-Adress-, Daten- und CRC-Felder enthält. Die von dem Barrieren-Transaktions-Nachrichtenpaket übertragenen Daten dienen ausschliesslich der Identifikation der Transaktion und eine Kopie dieser Daten wird von der CPU zum Zweck eines später von der CPU durchzuführenden Vergleichs abgespeichert.

[0521] Das Systemelement (z. B. eine der E/A-Schnittstellen 16, obwohl es sich dabei ebenso gut um eine andere CPU handeln könnte), welches ein Barrieren-Transaktions-Nachrichtenpaket empfängt, wird dadurch aufgefordert, seinerseits eine Barrieren-Transaktions-Antwort zu generieren und zu senden. Zuvor jedoch muss der Barrieren-Transaktions-Antwortadressat alle (von dem Systemelement, das die Anforderung ausgegeben hat erhaltenen) Anforderungen, welche vor dem Empfang des Barrieren-Transaktions-Nachrichtenpaket empfangen wurden, entweder vollenden oder ausscheiden, bevor er auf die Barrieren-Transaktion reagieren kann. Die Barrieren-Transaktions-Antwort erfolgt in Form eines HDC-Pakets (**Fig.** 4B), welches dieselben Daten enthält, die mit der Barrieren-Transaktions-Anforderung übertragen wurden. Wenn die Barrieren-Transaktions-Antwort von der CPU, welche die Transaktion eingeleitet hat, empfangen wird, so werden die in der Antwort enthaltenen Daten mit jenen Daten verglichen, die in dem zuvor gesendeten (und zuvor von der CPU gespeicherten) Barrieren-Transaktions-Nachrichtenpaket enthalten waren, um unter anderem festzustellen, welcher Barrieren-Transaktion die Antwort zuzuordnen ist (es können nämlich eine ganze Reihe von ausständigen Barriere-Transaktionen vorhanden sein).

[0522] Da es nur einen gültigen Leitweg zwischen jedem beliebigen Systemelement und jedem anderen Systemelement (z. B. der CPU **12A** und dem E/A-Einrichtung **17**$_n$; **Fig.** 1) geben kann, und da kein Nachrichtenpaket auf dem Weg zu seinem Bestimmungsort ein anderes Nachrichtenpaket überholen kann, entspricht die Reihenfolge, in der die Nachrichtenpakete empfangen werden auch der Reihenfolge, in der sie gesendet wurden. Somit wird die Barrieren-Transaktion dazu verwendet, um den Leitweg zwischen dem die Barrieren-Transaktion sendenden Systemelement und dem die Barrieren-Transaktion empfangenden und darauf reagierenden Systemelement freizumachen. Der Empfang der Barrieren-Transaktions-Antwort setzt das Systemelement, das die Barrieren-Transaktion ausgegeben hat, davon in Kenntnis, dass alle vor dem Versand der

Barrieren-Transaktion gesendeten Anforderungen beantwortet sind bzw. dass keine Antwort zu erwarten ist. Unbeantwortete Anforderungen können somit erneut ausgegeben werden, wobei davon ausgegangen werden kann, dass wenn schliesslich eine Antwort eintrifft, es sich dabei um das Ergebnis der erneut ausgegebenen Anforderung und nicht um eine verspätet enlangende Antwort auf die frühere (und zuvor unbeantwortet gelassene) Anfrage handelt. Es lässt sich unschwer erkennen, dass der Barrieren-Transaktions-Mechanismus mit einer sehr geringen Anzahl von TFN-Nummern ausommt (Hier wird ein nur vier Bits umfassendes TFN-Feld verwendet, im Gegensatz zu Systemen nach dem Stand der Technik, die etwas längere Felder verwenden müssen.) Die Funktionsweise der Barrieren-Transaktion ist in **Fig.** 35 veranschaulicht, in welcher eine CPU **12A** gezeigt wird, die an eine E/A-Paket-Schnittstelle 16A gekoppelt ist, und zwar über einen X-Pfad, welcher die in Reihe geschalteten Router **14A** und **14X** sowie die TNet-Links **L**() umfasst. Wie weiter oben beschrieben, verfügt jeder Router über einen Porteingang **502**, welcher elastische FIFOs **506** beinhaltet. Da für die folgende Abhandlung nur das elastische FIFO benötigt wird, wird auch nur dieses gezeigt.

[0523] Angenommen der abgehende Abschnitt des Links L(1) zwischen den Routern **14A** und **14X**, der Zinkabschnitt L(1)', wird unbenutzbar, wie in **Fig.** 35 durch die gestrichelte Linie angedeutet. Dazu kann es aus einer ganzen Reihe vonGründen kommen: fehlerhafte Anschlüsse, ein verrücktes Kabel, usw. Somit endet der abgehende Nachrichtenverkehr von dem Router **14A** zu dem Router **14X**. Alle von der CPU **12A** eingeleiteten Nachrichtenpaketübertragungen, die sich gerade auf dem Weg zu der E/A-Paket-Schnittstelle **16A**, jedoch noch vor dem fehlerbehafteten Linkabschnitt L(1)', befinden, erhalten keine Reaktion und enden daher mit der Anzeige einer Zeitablauffehlermeldung für den betreffenden Kommunikationspfad. Es wird intern ein Interrupt generiert und die Prozessoren **20** (**20a**, **20b**-**Fig.** 2) leiten die Ausführung einer Barrieren-Anforderungsroutine (BA-Routine) ein. Diese Barrieren-Anforderungsroutine (BA) deaktiviert zuerst den Pfad durch Löschen des PEX-Bits in dem 'Erlaubnisse'-Feld (**Fig.** 13B) eines jeden ZVÜ-Eintrags (**Fig.** 13A) für jede ausgegebene Transaktion (d. h. jedes Nachrichtenpaket), das in Ermangelung einer Antwort mit einer Zeitablauffehlermeldung endet. Dadurch wird gewährleistet, dass wenn das von der ausstehenden Transaktion geforderte Antwort-Nachrichtenpaket verspätet eintritt, es ausgeschieden wird, wenn auf die ZVÜ-Tabelle zugegriffen wird, um diese Antwort anhand der Tabelle zu prüfen; damit sind jene Nachrichtenpakete gemeint, welche nicht aufgrund eines Fehlers in einem Link blockiert werden, sondern nur vorübergehend ausser Evidenz geraten, bevor sie schliesslich einen Bestimmungsort erreichen.

[0524] Zu einem späteren Zeitpunkt wird das Link L(1) repariert und gibt nun veraltete Nachrichtenpakete frei, wie zum Beispiel jenes, das in 1152 in dem elastischen FIFO **506'** des Routers **14A** angezeigt wird. Die Wiederherstellung, des Links L(1) wird von dem IP 18 an die CPU **12A** gemeldet, so dass die CPU **12A** nun von der Möglichkeit unterrichtet ist, dass der X-Pfad zwischen ihr und der E/A-Schnittstelle 16A nunmehr wieder in Betrieb ist. Die CPU kann jedoch diesen Pfad (durch Zurücksetzen der PEX-Bits in den jeweiligen ZVÜ-Einträgen, welche der E/A-Paket-Schnittstelle **16A** entsprechen) noch nicht aktivieren. Der Grund dafür liegt in dem möglichen Vorhandensein veralteter Transaktions-Nachrichtenpakete, wie zum Beispiel jenes, das in 1152 in dem elastischen FIFO **506'** angezeigt wird, welches zu seinem ursprünglichen Bestimmungsort (der E/A-Paket-Schnittstelle) weiterfliesst, und die E/A-Paket-Schnittstelle **16A** dazu veranlasst, es als ein völlig anderes Nachrichtenpaket zu interpretieren und dementsprechend darauf zu reagieren.

[0525] Um diesem Problem zu begegnen, und bevor der X-Pfad von der CPU **12A** erneut für den normalen Datenverkehr genutzt wird, verwendet die in der CPU **12A** laufenden BA-Subroutine die BÜM-Logik **88** (siehe auch die **Fig.** 5 und 15) um eine Barrieren-Transaktion einzuleiten, indem sie ein "Barrieren-Anforderungs"-Nachrichtenpaket zu der E/A-Paket-Schnittstelle **16A** sendet. Das Barrieren-Anforderungs-Nachrichtenpaket wird von einem Unterfeld des in dem Header des Nachrichtenpakets enthaltenen Ursprungsort-Feldes als solches erkannt (siehe die **Fig.** 3A und 3B). Wie weiter oben angemerkt, enthält das Datenfeld des Barrieren-Anforderungs-Nachrichtenpakets einen von der bestimmten Transaktion einmalig verwendeten Datenwert.

[0526] Wenn das Barrieren-Anforderungs-Nachrichtenpaket (d. h. 1150) von der X-Schnittstelleneinheit **16a** der E/A-Paket-Schnittstelle **16A** empfangen wird, formuliert diese ein Antwort-Nachrichtenpaket, dessen Datenabschnitt denselben, identischen, einmalig verwendeten Datenwert enthält, der in dem empfangenen Barrieren-AnforderungsNachrichtenpaket 1150 enthalten war. Die E/A-Paket-Schnittstelle **16A** überträgt daraufhin die Antwort über die Router **14X**, **14A** zurück zu der CPU **12A**.

[0527] Wenn die Antwort auf das Barrieren-Anforderungs-Nachrichtenpaket von der CPU **12A** empfangen wird, wird es durch die ZVÜ-Logik **90'** verarbeitet (siehe auch die **Fig.** 5 und 11). Die Barrieren-Antwort verwendet einen ZVÜ-Eintrag mit aktiviertem "B"-Feld in dem entsprechenden 'Erlaubnisse'-Feld des Eintrags, um es Barrieren-Antworten, jedoch keinen anderen Transaktionsarten, zu ermöglichen, vollständig verarbeitet zu werden. (Zum Zeitpunkt des Sendens der Barrieren-Transaktion wurde von der CPU ein zur Verwendung bei der Überprüfung der Antwort bestimmter ZVÜ-Eintrag erstellt.)

[0528] Wie oben erklärt, enthält jede Barrieren-Transaktion einen Datenwert, welcher in der Antwort an den Absender zurückgegeben wird. Dieser einmalig verwendete Wert erlaubt es der CPU (d. h.. der BA-Routine), den gesendeten Datenwert mit dem in der Antwort empfangenen zu vergleichen, wodurch gewährleistet wer-

den kann, dass es sich bei der Antwort nicht um einen Teil einer anderen Barrieren-Transaktion handelt. Ist die Barrieren-Antwort an die CPU **12A** zurückgekehrt, so ist damit die Möglichkeit ausgeschlossen, dass sich noch irgendwo entlang diesem Pfad irgendwelche veralteten Pakete in den FIFO-Puffern befinden. Ausserdem hat die CPU **12A** damit auch abgeklärt, dass der zuvor deaktivierte Pfad nunmehr wieder für den normalen Datenverkehr genutzt werden kann. Demgemäss reaktiviert die CPU **12A** den Pfad durch Setzen des PEX-Erlaubnisse-Feldes in allen ZVÜ-Einträgen, die diesen Pfad verwenden.

[0529] Obgleich hier eine vollständige und komplette Offenlegung der Erfindung vorgenommen wurde, ist es für einschlägig gebildete Fachleute offensichtlich, dass verschiedene Abänderungen und Modifikationen an den verschiedenen Aspekten der Erfindung vorgenommen werden können, ohne dass dadurch von dem wahren Umfang der nachfolgenden Ansprüche abgegangen wird. So wurde zum Beispiel ein Schema zur Schaffung einer Erkennung von Fehlern, die bei der Übertragung von Befehls-/Datensymbolen auftreten können, in Form eines 8-Bit/9-Bit-Codes, welcher gewisse Fehler erkennen kann, offengelegt. Es sollte für einschlägig gebildete Fachleute klar sein, dass dieses Konzept auf andere, ähnliche Codes, wie zum Beispiel einen 9-Bit/10-Bit-Code oder einen sich über mehrere Bytes erstreckenden Code erweitert werden kann. Weiterhin könnten die Router **14** entsprechend konfiguriert werden, sodass sie eine beliebige Anzahl von Ports aufweisen; die Befehls/Datenpaketformate könnten anders gestaltet sein (mit mehr oder weniger Bits in dem Headerfeld und in anderen Feldern); die Leitwegtopologien können, unter Verwendung der Router **14**, als Ringe, als Bäume, als Hypercube und dergleichen ausgebildet sein.

Patentansprüche

1. Verfahren zum fehlertoleranten Betrieb eines Paars von Prozessoreinheiten (**12A**, **128**) in einem Rechensystem, wobei das Paar von Prozessoreinheiten entsprechend betrieben wird, um im wesentlichen identische Befehle in synchroner Weise abzuarbeiten, wobei jede Einheit des Paars von Prozessoreinheiten zum Zweck des Informationsaustausches mit einem ersten und einem zweiten Datenübertragungselement (**14A**, **14B**) verbunden ist, welche betrieben werden, um Daten zu und von dem Paar von Prozessoreinheiten zu übertragen, wobei das Verfahren die folgenden Schritte umfaßt:
das erste und das zweite Datenübertragungselement (**14A**, **14B**) empfangen und vergleichen (**1002**) die von dem Paar von Prozessoreinheiten (**12A**, **12B**) kommenden Daten, und zumindest eines der ersten und zweiten Datenübertragungselemente wird betrieben, um an das Paar von Prozessoreinheiten ein Fehlersignal (DVRG) zu übermitteln (**1009**), das auf eine Nichtübereinstimmung (miscompare) der Daten hinweist;
jede der Prozessoreinheiten empfängt das Fehlersignal und übermittelt als Reaktion darauf ein reflektiertes Fehlersignal an das erste und zweite Datenübertragungselement;
jedes der ersten und zweiten Datenübertragungselemente (**14A**, **14B**), das keine Nichtübereinstimmung (miscompare) der Daten erkennt, wird betrieben, um in Reaktion auf das Empfangen des von zumindest einer Einheit des Paars von Prozessoreinheiten kommenden, reflektieren Fehlersignals ein weiteres, reflektiertes Fehlersignal an das Paar von Prozessoreinheiten zu übermitteln;
jede Einheit des Paars von Prozessoreinheiten bestimmt (**1006**) durch Analyse der ihnen vorliegenden, verschiedenen Fehlerangaben, ob der Betrieb fortgesetzt wird oder nicht; und
eine Einheit des Paars von Prozessoreinheiten setzt den Betrieb fort, und die andere Einheit des Paars von Prozessoreinheiten beendet den Betrieb (1012).

2. Verfahren nach Anspruch 1, welches als weitere Schritte das durch die andere Einheit des Paars von Prozessoreinheiten (**12A**, **128**) nach Empfangen des Fehlersignals bzw. des weiteren, reflektierten Fehlersignals erfolgende Erkennen eines Datenfehlers beim Informationsaustausch mit dem ersten oder dem zweiten Datenübertragungselement und das in Reaktion darauf erfolgende Beenden des Betriebs beinhaltet.

3. Verfahren nach Anspruch 1, wobei jedes Element des ersten und zweiten Datenübertragungselementes (**14A**, **14B**) mit mindestens einem Peripheriegerät (**17**, **18**) zum Zweck der Datenübermittlung zu und von dem Peripheriegerät und dem Paar von Prozessoreinheiten verbunden ist, und wobei das Verfahren den Schritt des Fortsetzens der Übertragung von Daten enthält, die von dem Paar von Prozessoreinheiten an das Peripheriegerät übermittelt werden, und das Halten eines letzten, für das Peripheriegerät bestimmten Datenabschnitts, wenn eine Nichtübereinstimmung (miscompare) der Daten erkannt wird.

4. Verfahren nach Anspruch 1, welches den Schritt enthält, dass die eine Einheit des Paars von Prozessoreinheiten auf den Übertragungswegen ein Eigentümersignal (IOY) an das erste und zweite Datenübertragungselement (**14A**, **14B**) übermittelt.

5. Verfahren nach Anspruch 4, wobei die Eigentümerdaten das erste und das zweite Datenübertragungselement (**14A**, **14B**) anweisen, Datenübertragungen von der anderen Einheit des Paars von Prozessoreinhei-

ten zu ignorieren.

6. Verfahren nach Anspruch 4, wobei das Eigentümersignal (IOY) das erste und das zweite Datenübertragungselement anweist, nur mit der einen Einheit des Paars von Prozessoreinheiten Daten auszutauschen.

7. Verfahren nach Anspruch 4, wobei das erste und das zweite Datenübertragungselement in Reaktion auf das Empfangen der Eigentümerdaten die von der anderen Einheit des Paars von Prozessoreinheiten kommenden Datenübertragungen ignoriert, und wobei die von der anderen Einheit des Paars von Prozessoreinheiten kommenden Daten zusammen mit einer Datenkennzeichnung als schlechte Daten (TPB) zur Übertragung an eines der Peripheriegeräte bestimmt werden, wohingegen die von der einen Einheit des Paars von Prozessoreinheiten kommenden Daten mit einer anderen Datenkennzeichnung als gute Daten (TPG) ausgewiesen werden.

8. Rechensystem mit:
mindestens einem Paar von Prozessoreinheiten (**12A**, **12B**), von denen eine jede geeignet ist, dieselben Befehle eines im wesentlichen identischen Befehlsstroms im wesentlichen zeitgleich zueinander abzuarbeiten;
einem ersten und einem zweiten Datenübertragungselement (**14A**, **14B**), welche zum Zweck des wechselseitigen Datenaustauschs kommunikativ mit einer jeden Einheit des Paars von Prozessoreinheiten verbunden sind, wobei das erste und das zweite Datenübertragungselement jeweils folgendes beinhaltet:
(i) eine Einrichtung zum Vergleichen der von einer Einheit des Paars von Prozessoreinheiten empfangenen Daten mit den von der anderen Einheit des Paars von Prozessoreinheiten stammenden Daten, sowie zum Erkennen einer Nichtübereinstimmung (miscompare);
(ii) eine Einrichtung zum Übertragen eines Fehlersignals (DVRG) an das Paar von Prozessoreinheiten, welches auf eine Nichtübereinstimmung (miscompare) der Daten hinweist;
einer in jeder Einheit des Paars von Prozessoreinheiten vorhandenen Einrichtung zum Empfangen des Fehlersignals und zum in Reaktion darauf erfolgenden Übermitteln eines reflektierten Fehlersignals an das erste und das zweite Datenübertragungselement;
einer jeweils im ersten und im zweiten Datenübertragungselement (**14A**, **14B**) vorhandenen Einrichtung zum Empfangen des reflektierten Fehlersignals und, wenn keine Nichtübereinstimmung (miscompare) der Daten erkannt worden ist, zum Betreiben, um in Reaktion auf das Empfangen des von zumindest einer Einheit des Paars von Prozessoreinheiten kommenden, reflektierten Fehlersignals ein weiteres reflektiertes Fehlersignal an das Paar von Prozessoreinheiten zu übertragen;
einer in jeder Einheit des Paars von Prozessoreinheiten vorhandenen Einrichtung, mittels welcher durch Analyse der ihnen vorliegenden, verschiedenen Fehlerangaben bestimmt wird, ob der Betrieb fortgesetzt werden soll oder nicht; und
einer in jedem Prozessor des Prozessorpaars vorhandenen Einrichtung zum entsprechend der getroffenen Bestimmung erfolgenden Fortsetzen bzw. Beenden des Betriebs.

Es folgen 30 Blatt Zeichnungen

Anhängende Zeichnungen

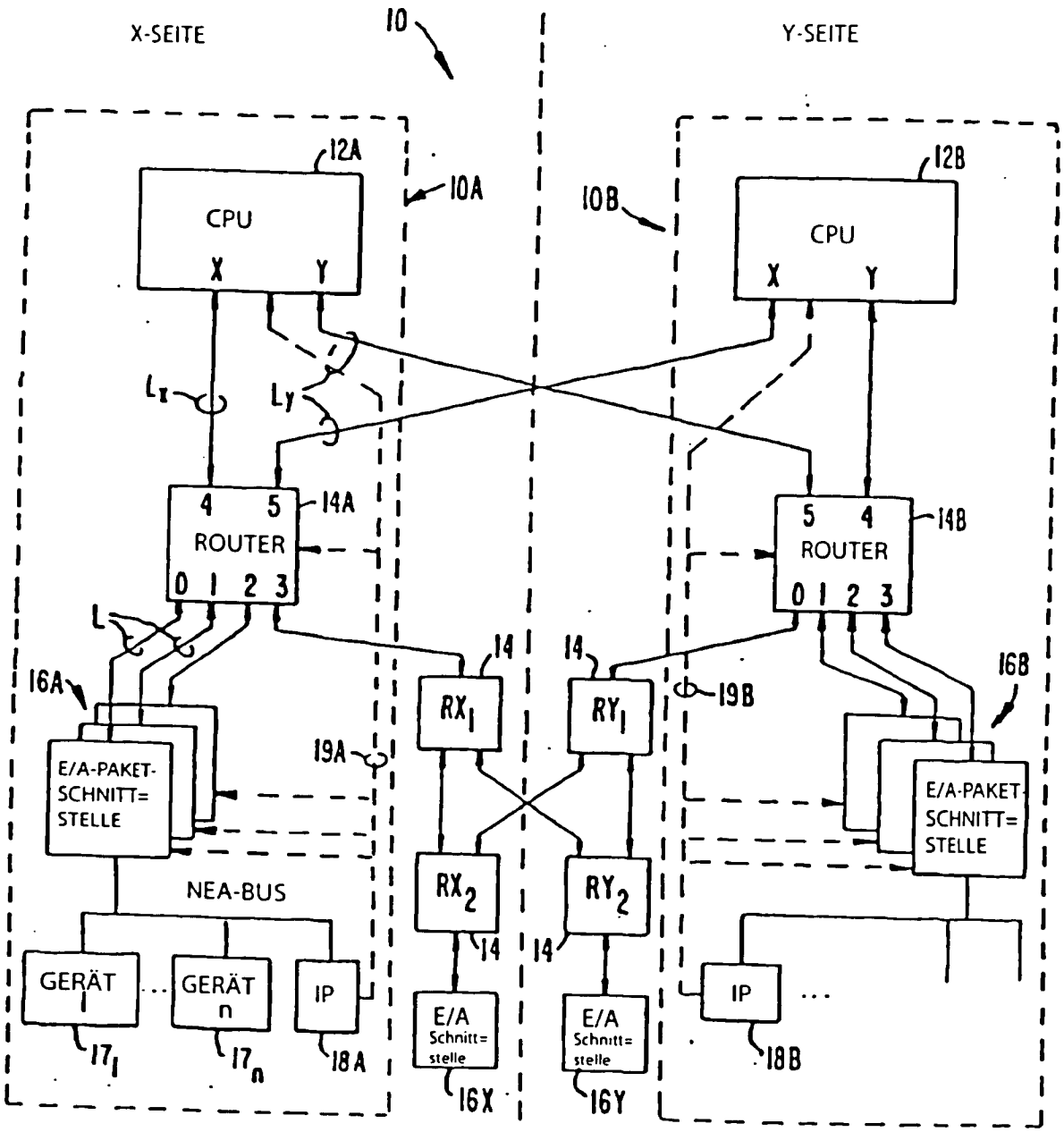


FIG. 1A.

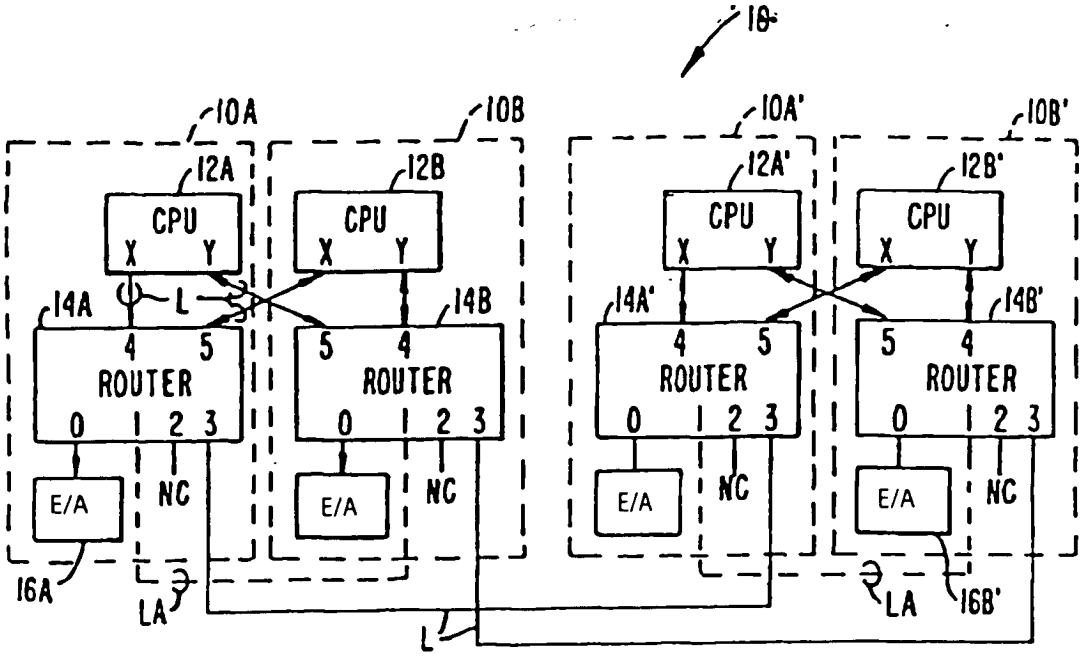

18
10A
10B
10A'
10B'
12A
12B
12A'
12B'
CPU
X
Y
14A
14B
14A'
14B'
L
4
5
ROUTER
0
2
3
E/A
NC
16A
LA
16B'

FIG. 1B.

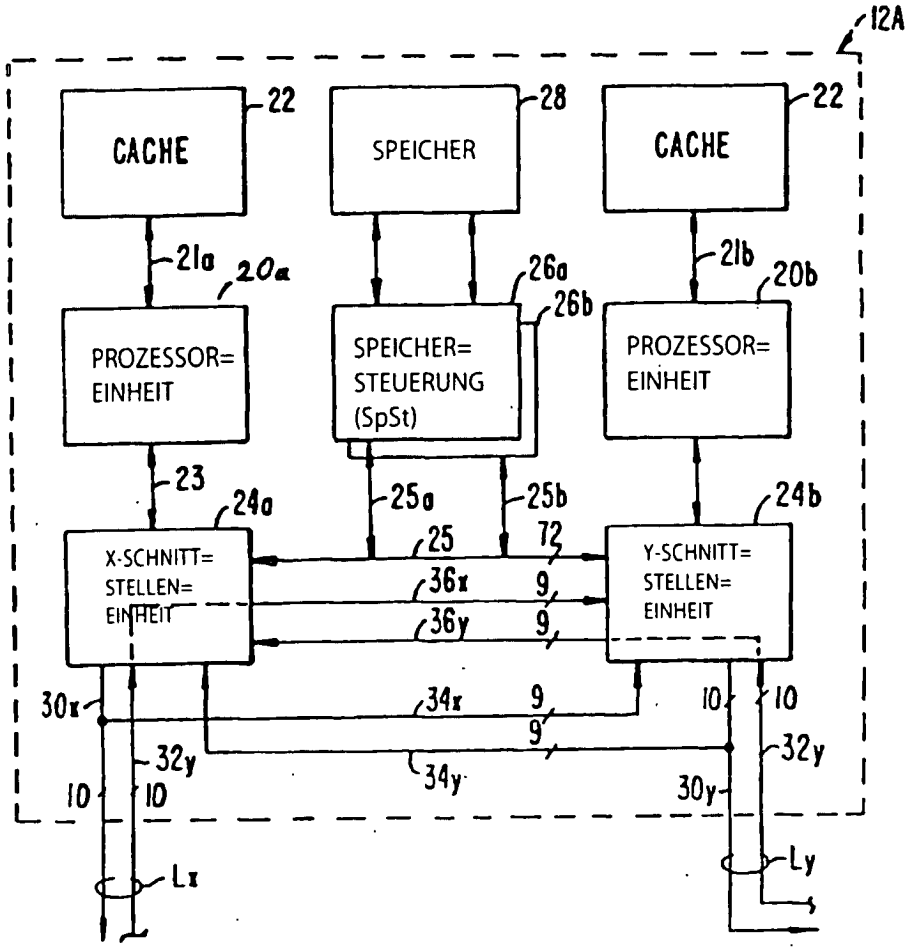

12A
CACHE
22
SPEICHER
28
21a
20a
26a
26b
21b
20b
PROZESSOR=
EINHEIT
SPEICHER=
STEUERUNG
(SpSt)
23
24a
25a
25b
24b
X-SCHNITT=
STELLEN=
EINHEIT
25
72
Y-SCHNITT=
STELLEN=
EINHEIT
36x
36y
9
30x
34x
34y
10
32y
30y
Lx
Ly

FIG. 2.

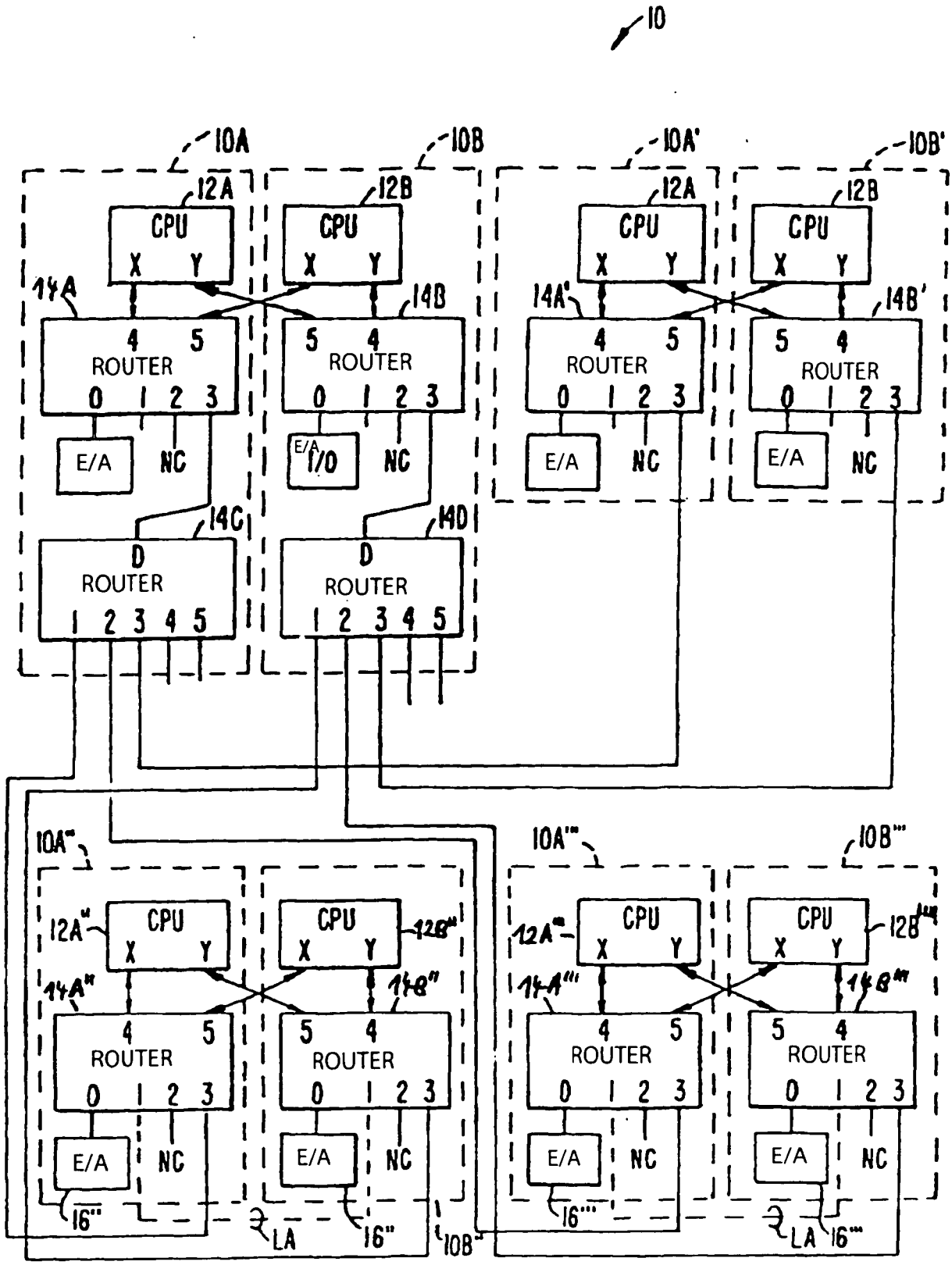

10
10A
10B
10A'
10B'
12A
12B
CPU
X Y
14A
14B
14A'
14B'
4 5
ROUTER
0 1 2 3
5 4
E/A
NC
E/A I/O
14C
14D
D
1 2 3 4 5
10A''
10A'''
10B'''
12A''
12B''
12A'''
12B'''
14A''
14B''
14A'''
14B'''
16''
16'''
LA
10B''

FIG. 1C.

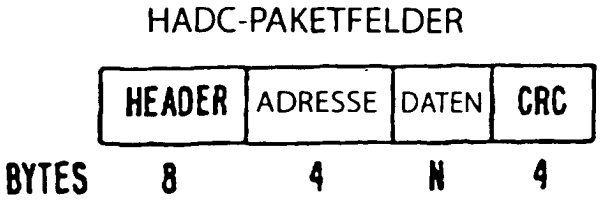


FIG. 3A.

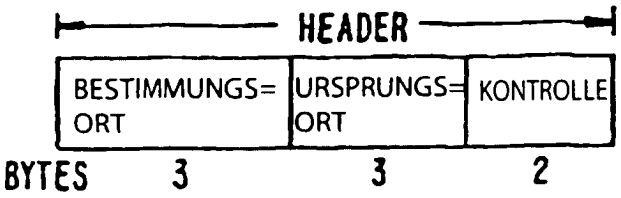


FIG. 3B.

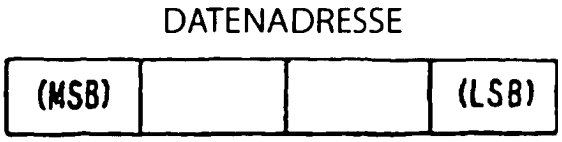


FIG. 3C.

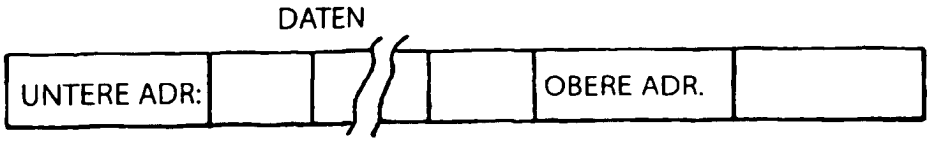


FIG. 3D.

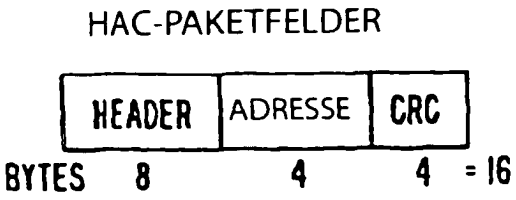


FIG. 4A.

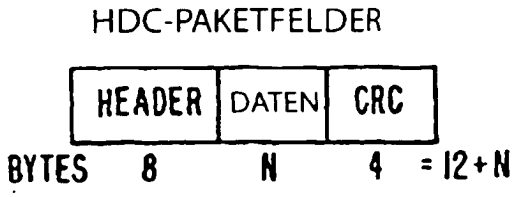


FIG. 4B.

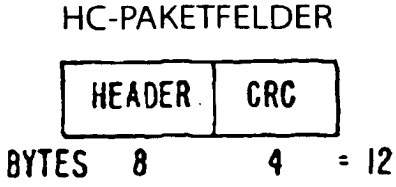


FIG. 4C.

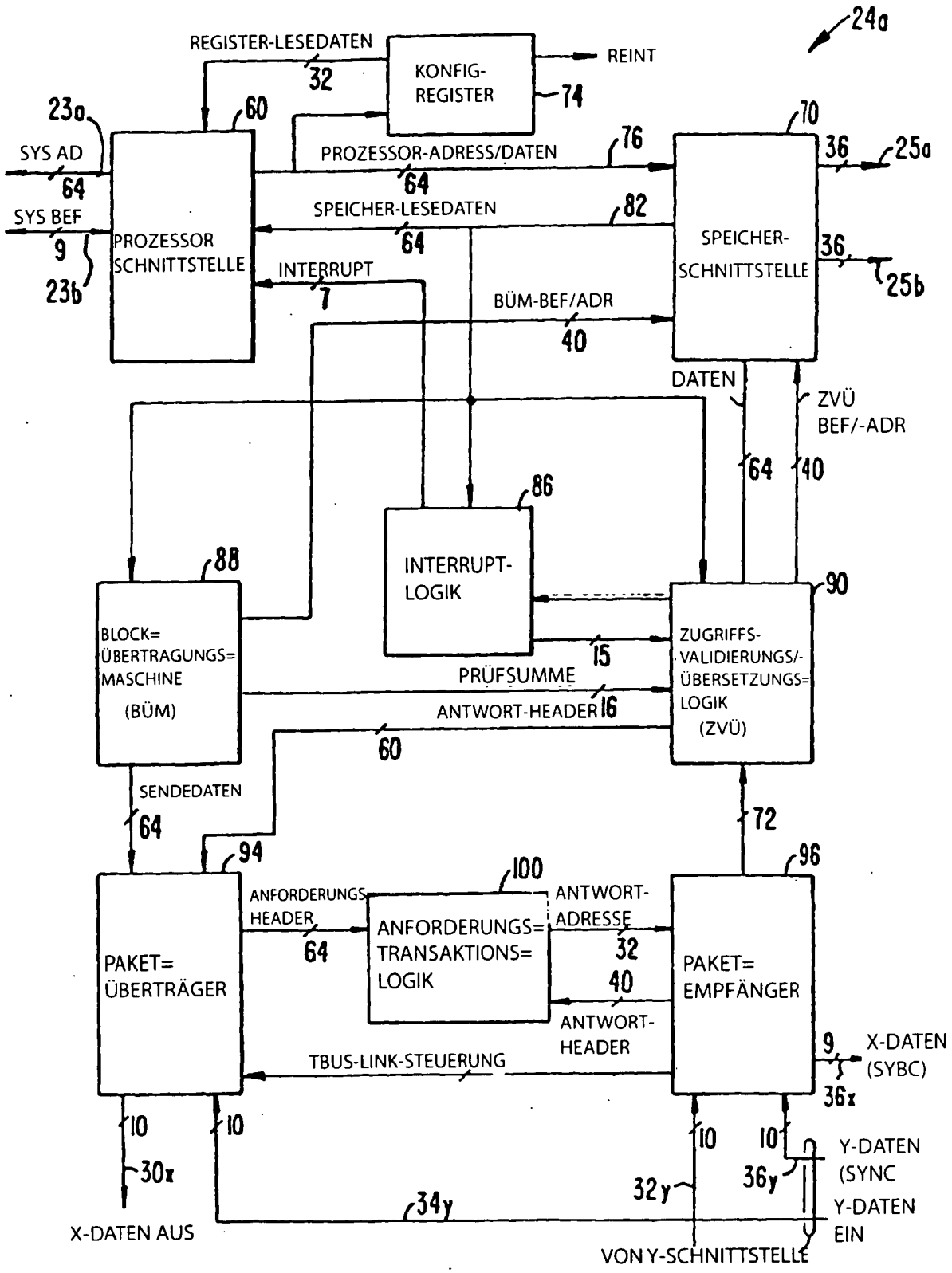

24a
REGISTER-LESEDATEN
32
KONFIG-REGISTER
REINT
74
23a
60
70
SYS AD
64
PROZESSOR-ADRESS/DATEN
76
64
36
25a
SYS BEF
9
23b
PROZESSOR SCHNITTSTELLE
SPEICHER-LESEDATEN
82
64
SPEICHER-SCHNITTSTELLE
36
25b
INTERRUPT
7
BÜM-BEF/ADR
40
DATEN
ZVÜ BEF/-ADR
64
40
86
INTERRUPT-LOGIK
88
90
BLOCK= ÜBERTRAGUNGS= MASCHINE (BÜM)
15
PRÜFSUMME
ANTWORT-HEADER
16
ZUGRIFFS-VALIDIERUNGS/-ÜBERSETZUNGS= LOGIK (ZVÜ)
60
SENDEDATEN
64
72
94
100
96
ANFORDERUNGS HEADER
64
ANFORDERUNGS= TRANSAKTIONS= LOGIK
ANTWORT-ADRESSE
32
40
ANTWORT-HEADER
PAKET= ÜBERTRÄGER
PAKET= EMPFÄNGER
9
X-DATEN (SYBC)
36x
TBUS-LINK-STEUERUNG
10
10
10
10
30x
32y
36y
Y-DATEN (SYNC
Y-DATEN EIN
34y
X-DATEN AUS
VON Y-SCHNITTSTELLE

FIG. 5.

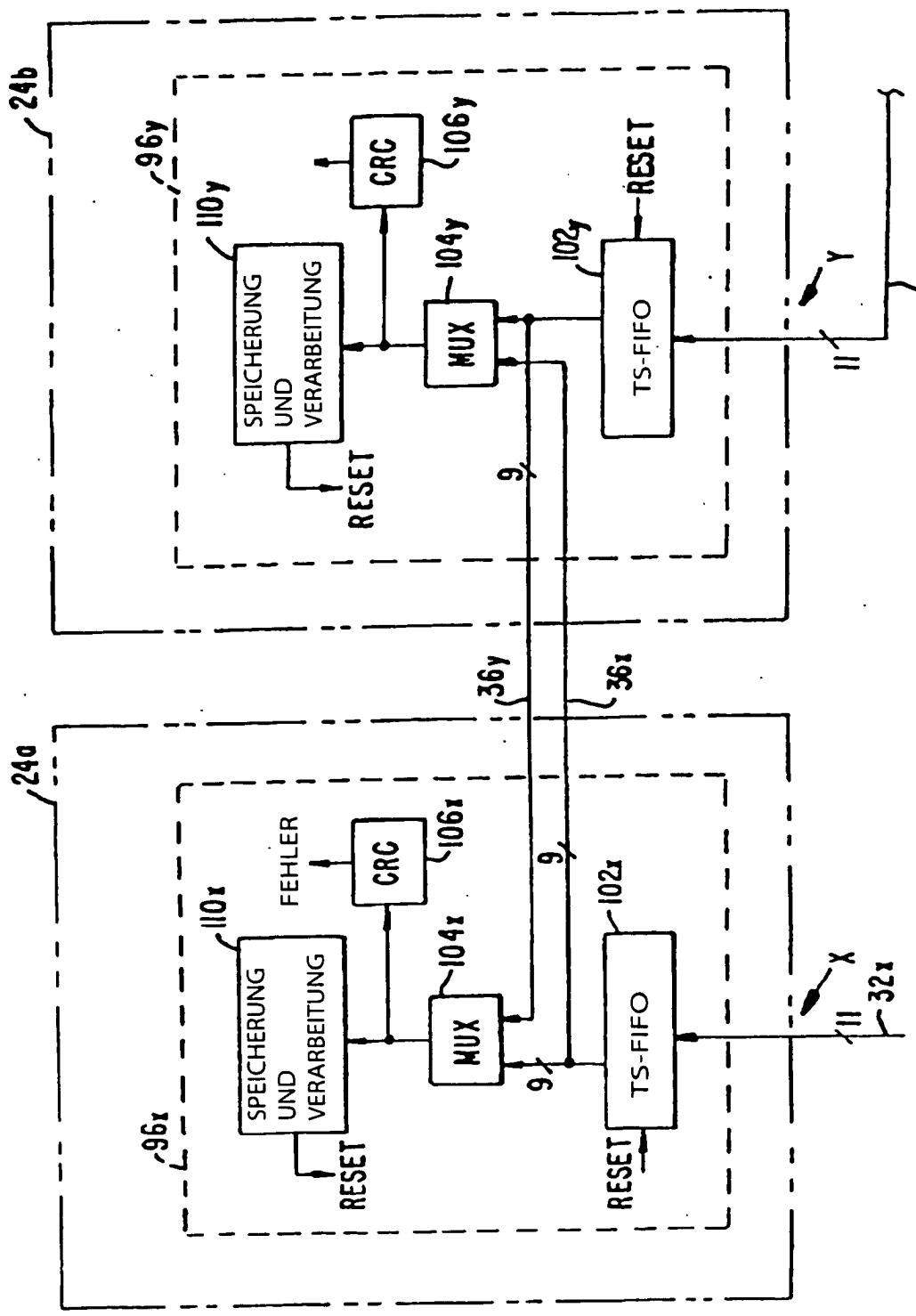

24b
96y
110y
SPEICHERUNG UND VERARBEITUNG
RESET
CRC
106y
104y
MUX
102y
TS-FIFO
RESET
Y
32y
11
9
36y
36x
24a
96x
110x
SPEICHERUNG UND VERARBEITUNG
RESET
FEHLER
CRC
106x
104x
MUX
102x
TS-FIFO
RESET
X
32x
11
9

FIG. 6.

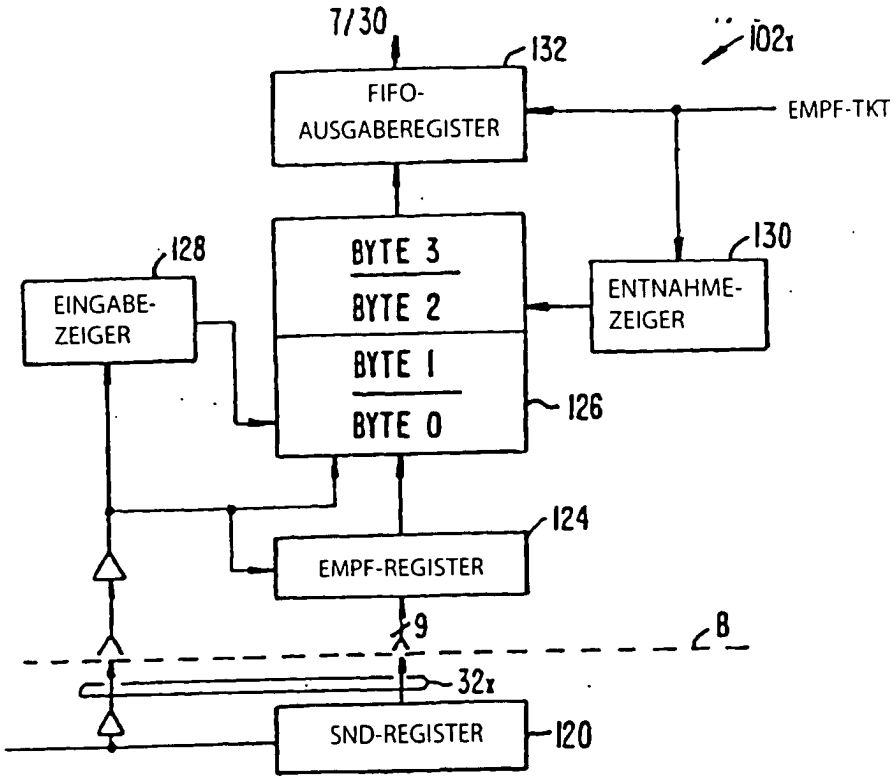

7/30
132
102x
FIFO-
AUSGABEREGISTER
EMPF-TKT
128
EINGABE-
ZEIGER
BYTE 3
BYTE 2
BYTE 1
BYTE 0
126
130
ENTNAHME-
ZEIGER
124
EMPF-REGISTER
9
8
32x
SND-REGISTER
120

FIG. 7A.

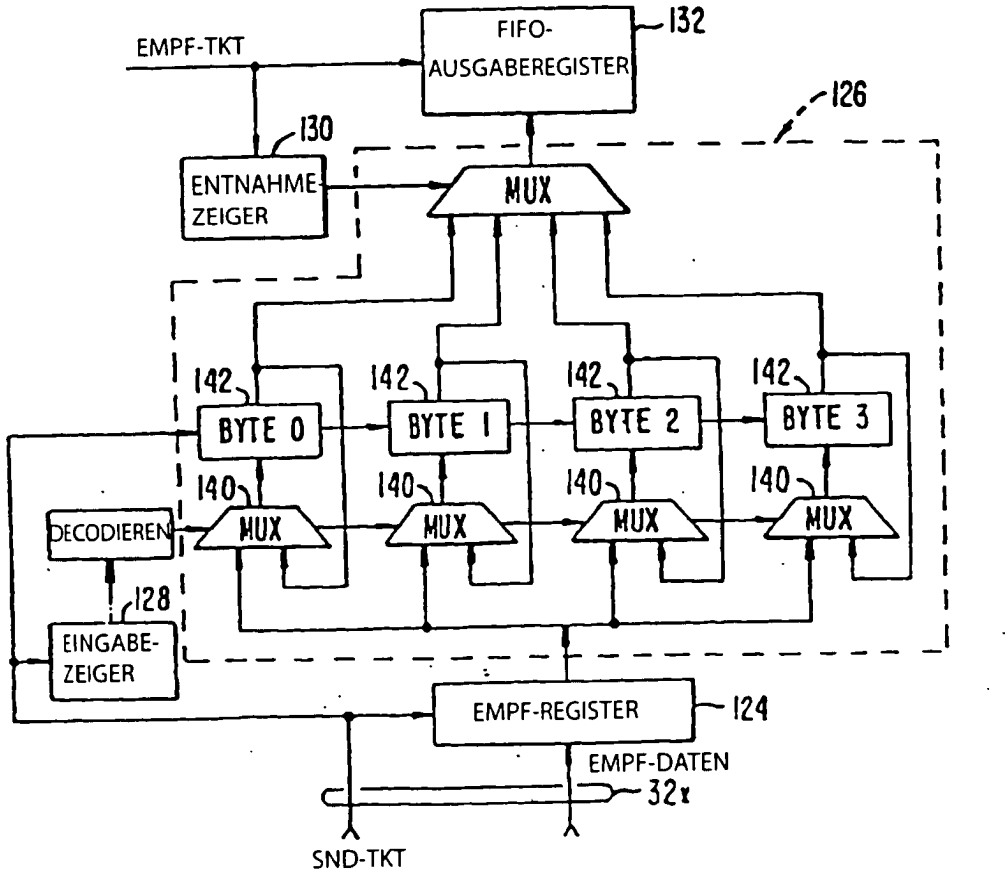

EMPF-TKT
FIFO-
AUSGABEREGISTER
132
126
130
ENTNAHME-
ZEIGER
MUX
142
142
142
142
BYTE 0
BYTE 1
BYTE 2
BYTE 3
140
140
140
140
DECODIEREN
MUX
MUX
MUX
MUX
128
EINGABE-
ZEIGER
EMPF-REGISTER
124
EMPF-DATEN
32x
SND-TKT

FIG. 7B.

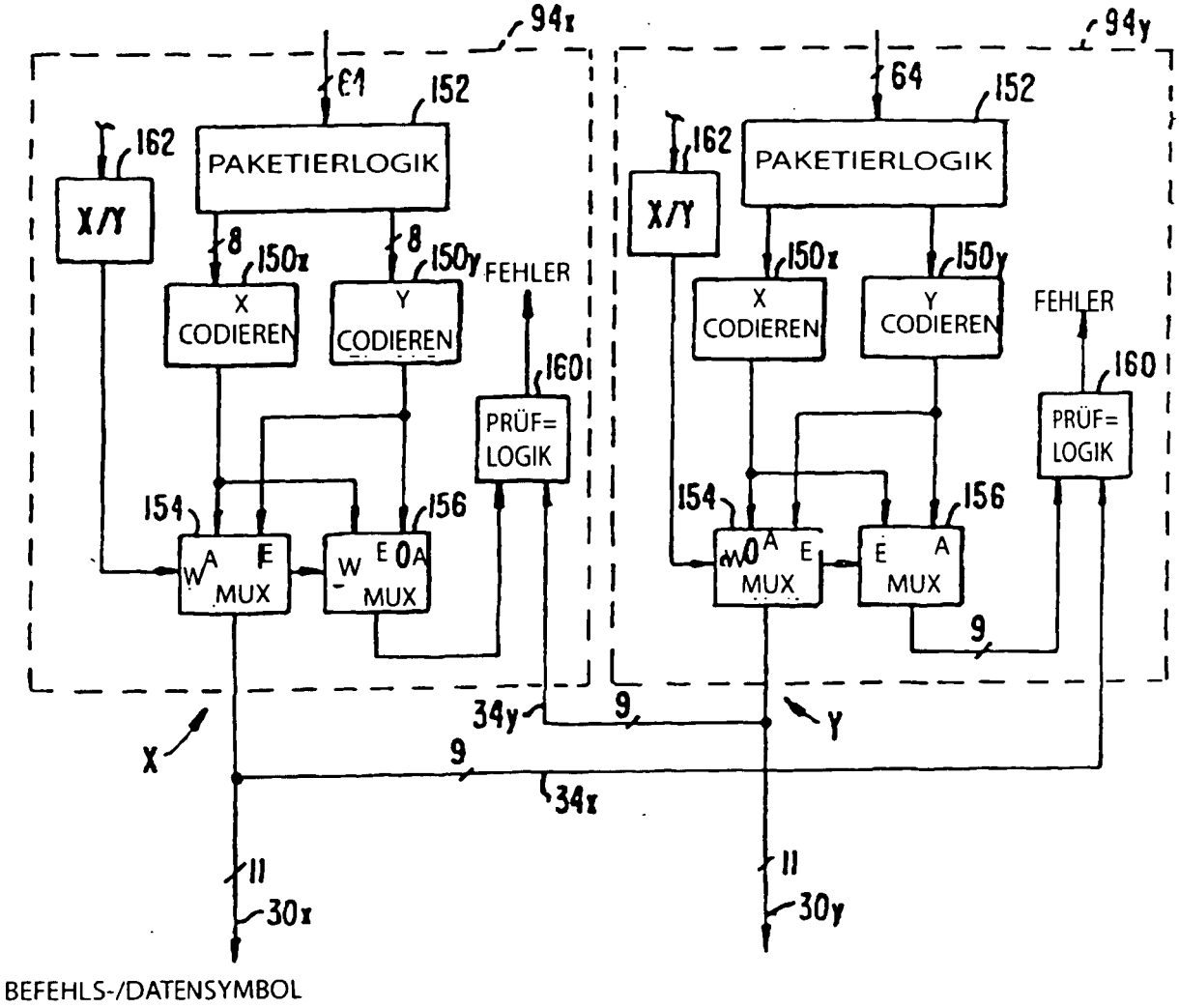


FIG. 8.

CDC	CDB	CDA	CD5	CD4	CD3	CD2	CD1	CD0

FIG. 9.

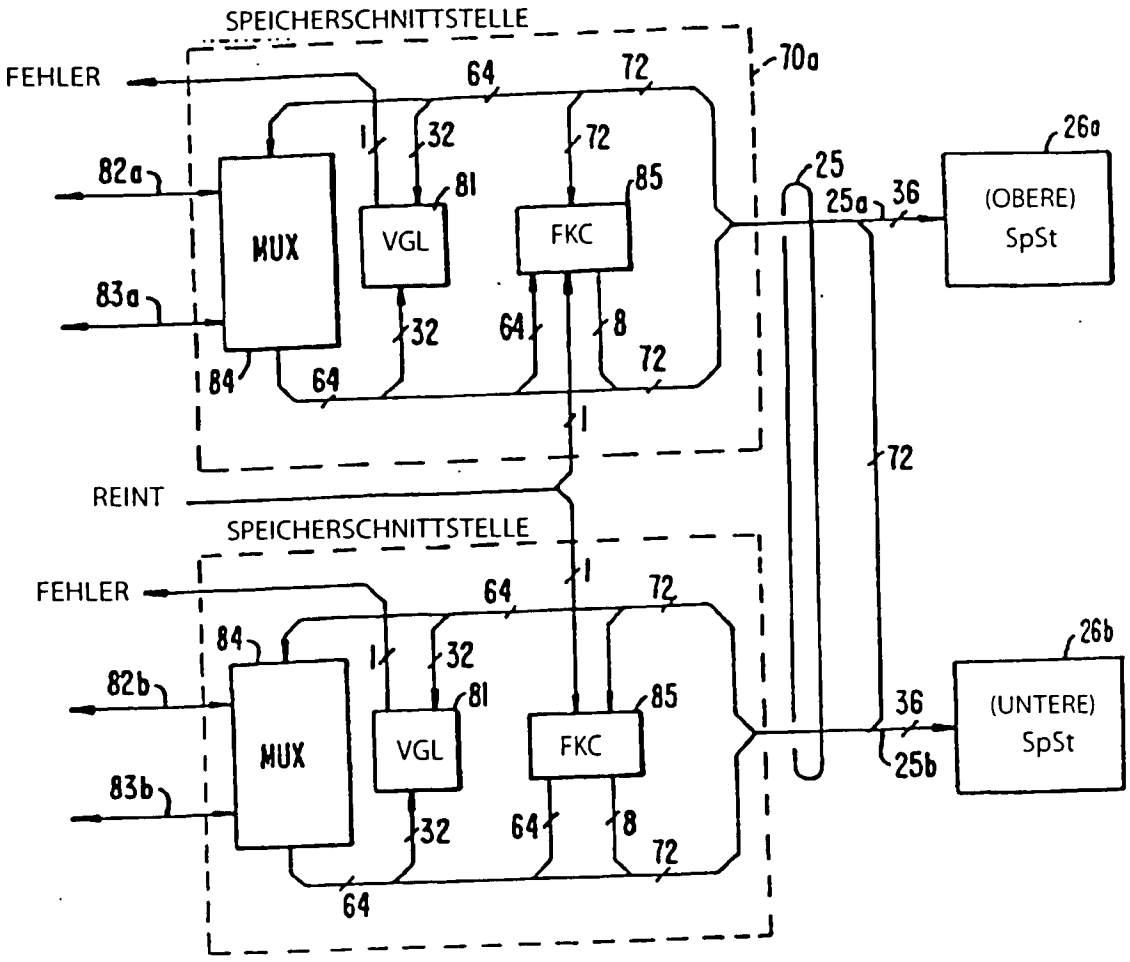

SPEICHERSCHNITTSTELLE
FEHLER
64
72
70a
1
32
72
26a
82a
MUX
VGL
81
FKC
85
25
25a
36
(OBERE)
SpSt
83a
32
64
8
72
84
64
1
72
REINT
SPEICHERSCHNITTSTELLE
FEHLER
64
72
1
84
26b
82b
32
81
85
MUX
VGL
FKC
36
(UNTERE)
SpSt
83b
25b
32
64
8
72
64

FIG. 10.

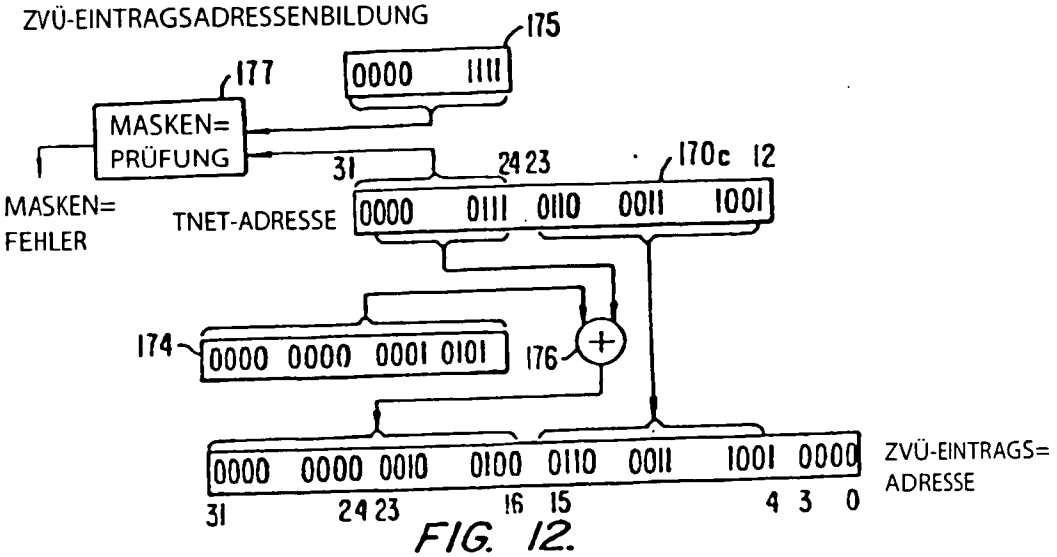

ZVÜ-EINTRAGSADRESSENBILDUNG
175
177
0000 1111
MASKEN=
PRÜFUNG
31
24 23
170c 12
MASKEN=
FEHLER
TNET-ADRESSE
0000 0111 0110 0011 1001
174
0000 0000 0001 0101
176
0000 0000 0010 0100 0110 0011 1001 0000
ZVÜ-EINTRAGS=
ADRESSE
31
24 23
16 15
4 3 0

FIG. 12.

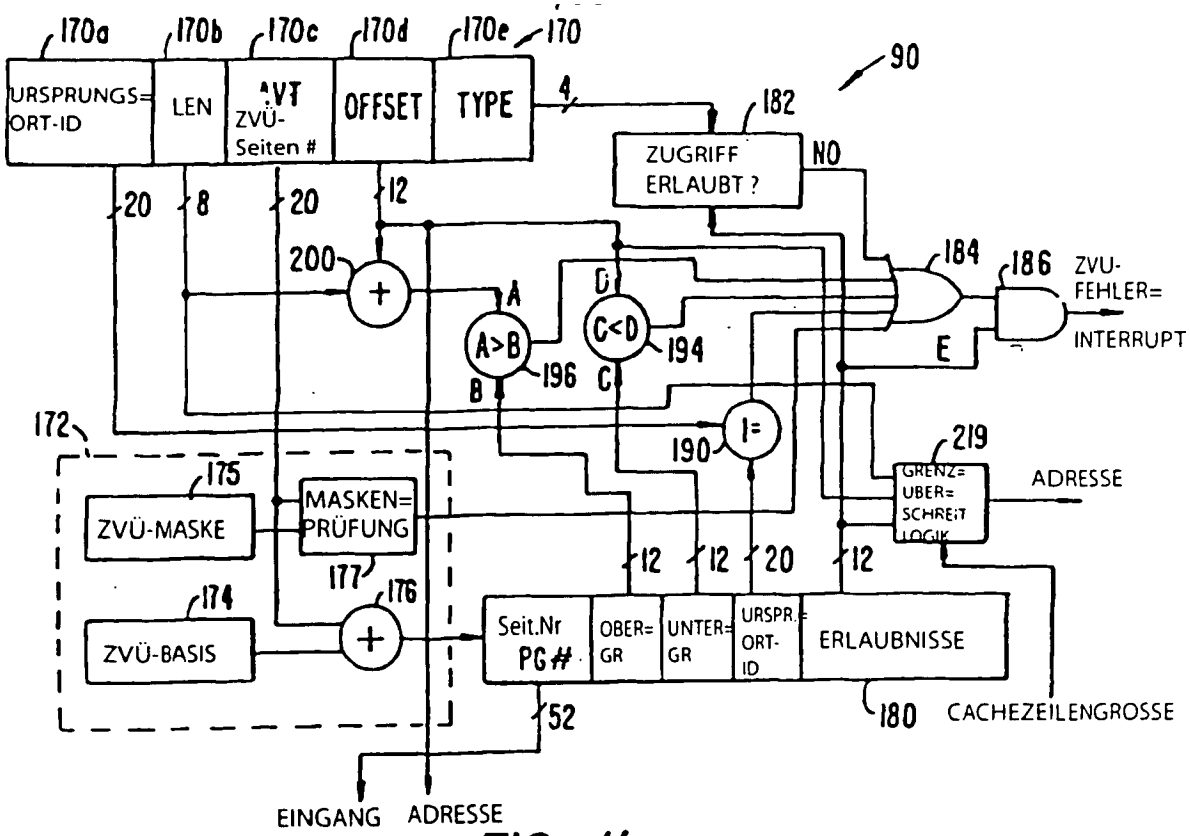


FIG. 11.

ZVÜ-EINGANG (NORMAL)

20 BITS	52 BITS	12 BITS	12 BITS	20 BITS	12 BITS
(RESERV.)	PHYSIKALISCHE SEITENZAHL	UNTERGRENZE	ERLAUBNISSE	URSPRUNGSORT-ID	OBERGRENZE

FIG. 13A.

E	PEX	PEY	I	C[E:A]	W	R	B	RES.

FIG. 13B.

ZVÜ-EINGANG (INTERRUPT)

64 BITS	12 BITS	20 BITS	16 BITS	16 BITS
WARTESCHLANGEN-BASISADRESSE	ERLAUBNISSE	URSPRUNGSORT-ID	(RESERVIERT)	c q I z

FIG. 13C.

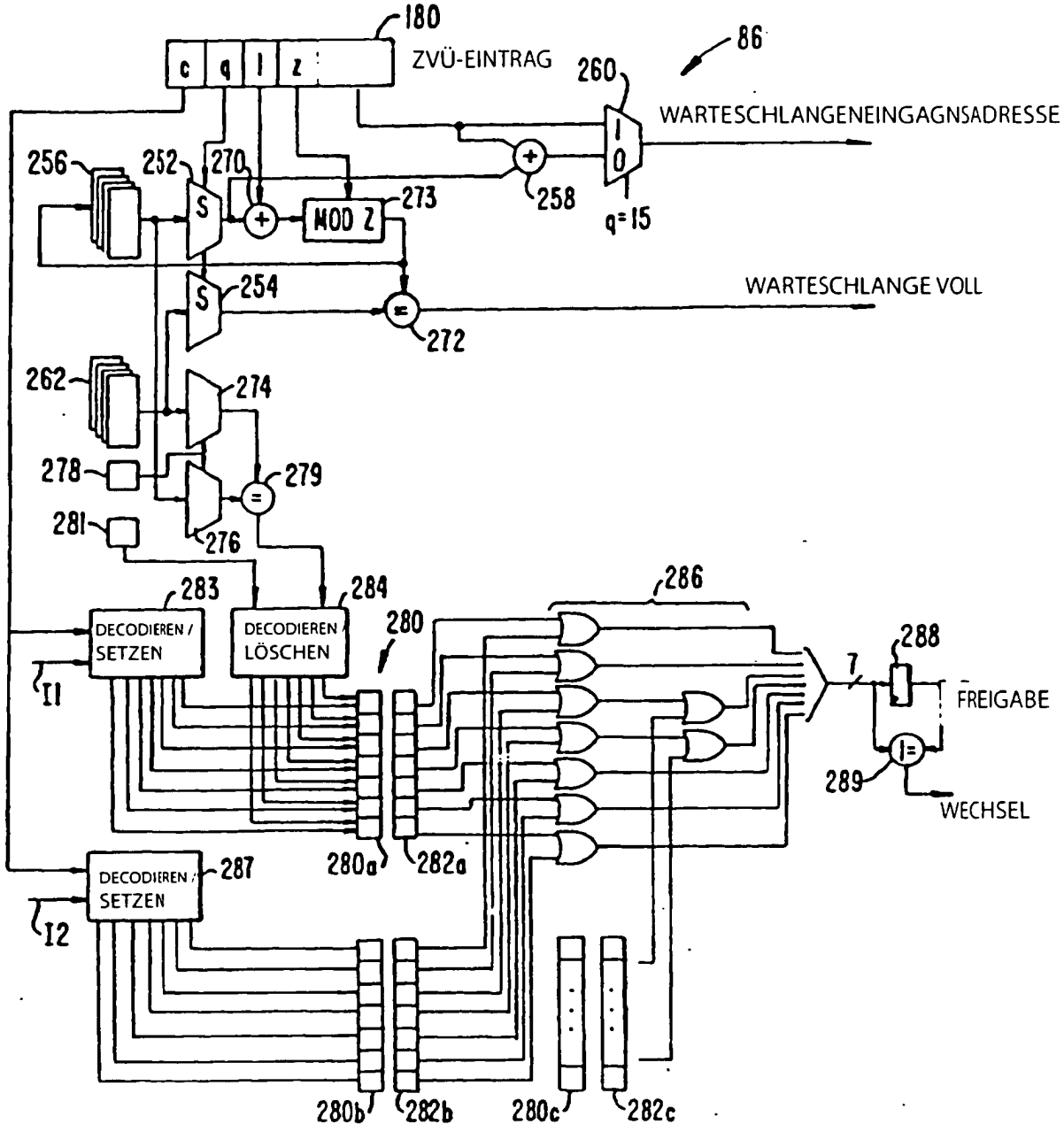

180
c
q
l
z
ZVÜ-EINTRAG
86
260
WARTESCHLANGENEINGAGNSADRESSE
I
O
256
252
270
S
MOD Z
273
258
q=15
S
254
WARTESCHLANGE VOLL
272
262
274
278
279
281
276
283
284
286
DECODIEREN / SETZEN
DECODIEREN LÖSCHEN
280
288
FREIGABE
289
WECHSEL
I1
I2
DECODIEREN / SETZEN
287
280a
282a
280b
282b
280c
282c

FIG. 14A.

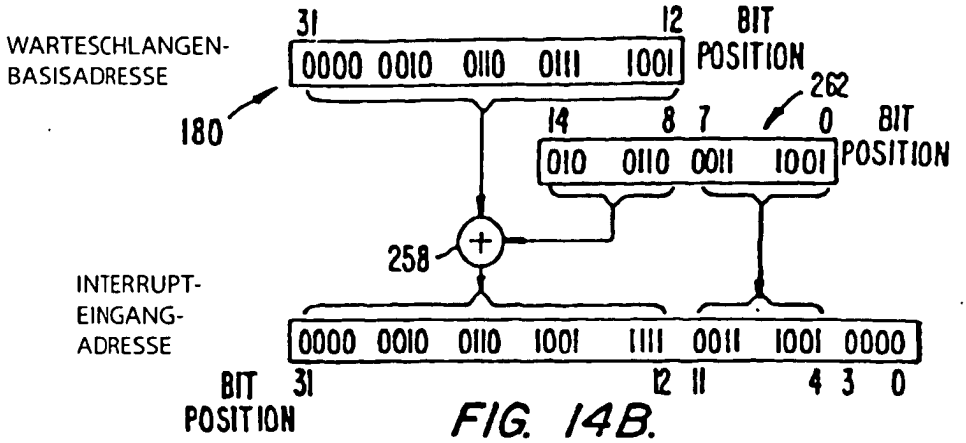
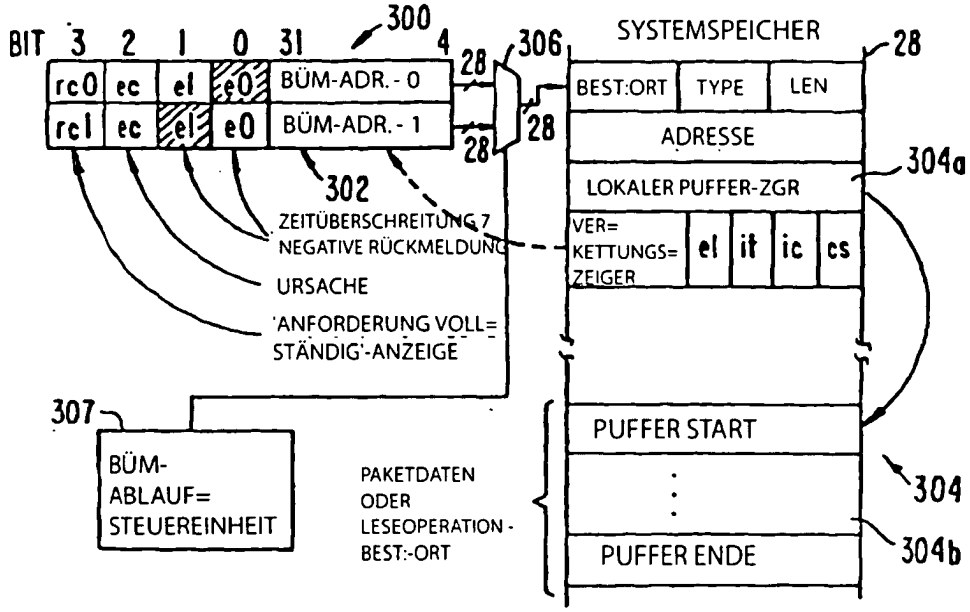


FIG. 14B.



rc 0/1 -
ec - FEHLERURSACHE: 1= ZEITÜBERSCHREITUNG
0 = NEGATIVE RÜCKMELDUNG
e 0/1 - FEHLER AUF KANAL 0/1

el - LINKENDE
it - INTERRUPT BEI ZEITÜBERSCHREITUNG ODER NEGATIVER RÜCKMELDUNG
ic - INTERRUPT BEI FERTIGSTELLUNG
cs - PRÜFSUMMEN-DATENPAKET

FIG. 15.

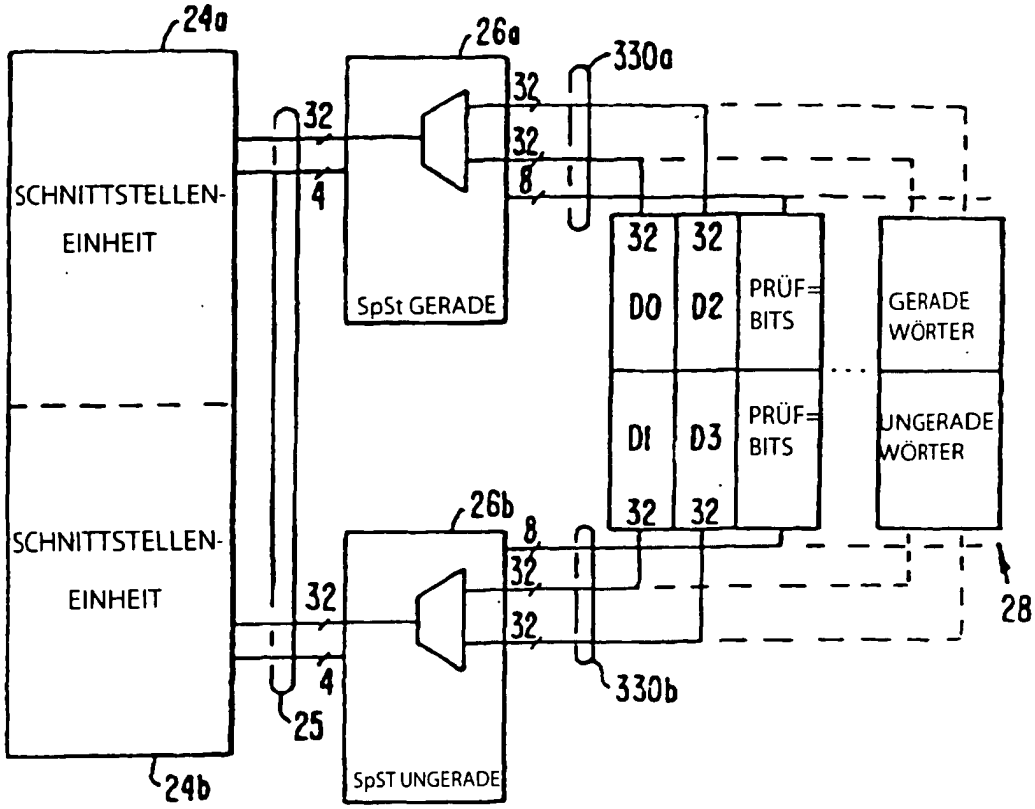

24a
26a
330a
32
32
32
8
4
SCHNITTSTELLEN-
EINHEIT
SpSt GERADE
32
32
D0
D2
PRÜF=
BITS
GERADE
WÖRTER
D1
D3
PRÜF=
BITS
UNGERADE
WÖRTER
32
32
26b
8
32
32
SCHNITTSTELLEN-
EINHEIT
32
4
25
330b
28
24b
SpST UNGERADE

FIG. 16.

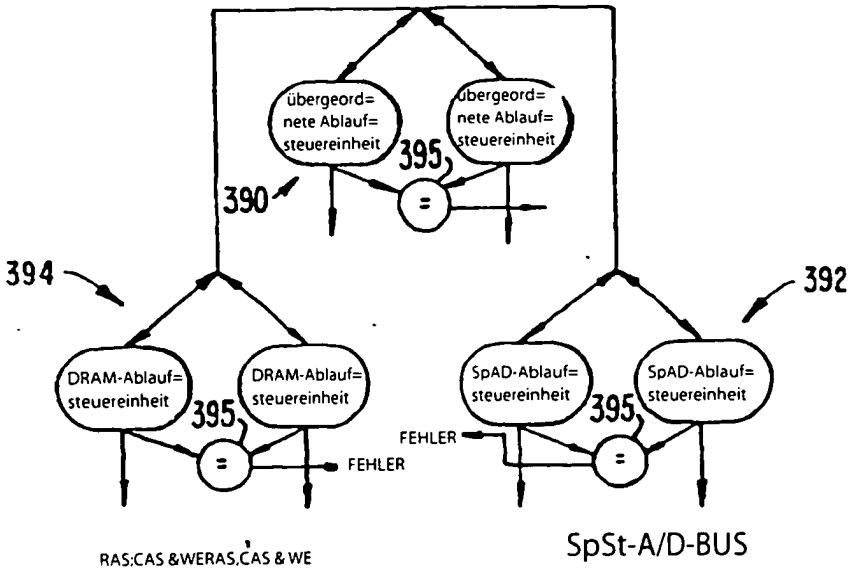

übergeord=
nete Ablauf=
steuereinheit
übergeord=
nete Ablauf=
steuereinheit
395
390
=
394
392
DRAM-Ablauf=
steuereinheit
DRAM-Ablauf=
steuereinheit
SpAD-Ablauf=
steuereinheit
SpAD-Ablauf=
steuereinheit
395
395
FEHLER
=
=
FEHLER
RAS;CAS &WERAS,CAS & WE
SpSt-A/D-BUS

FIG. 18.

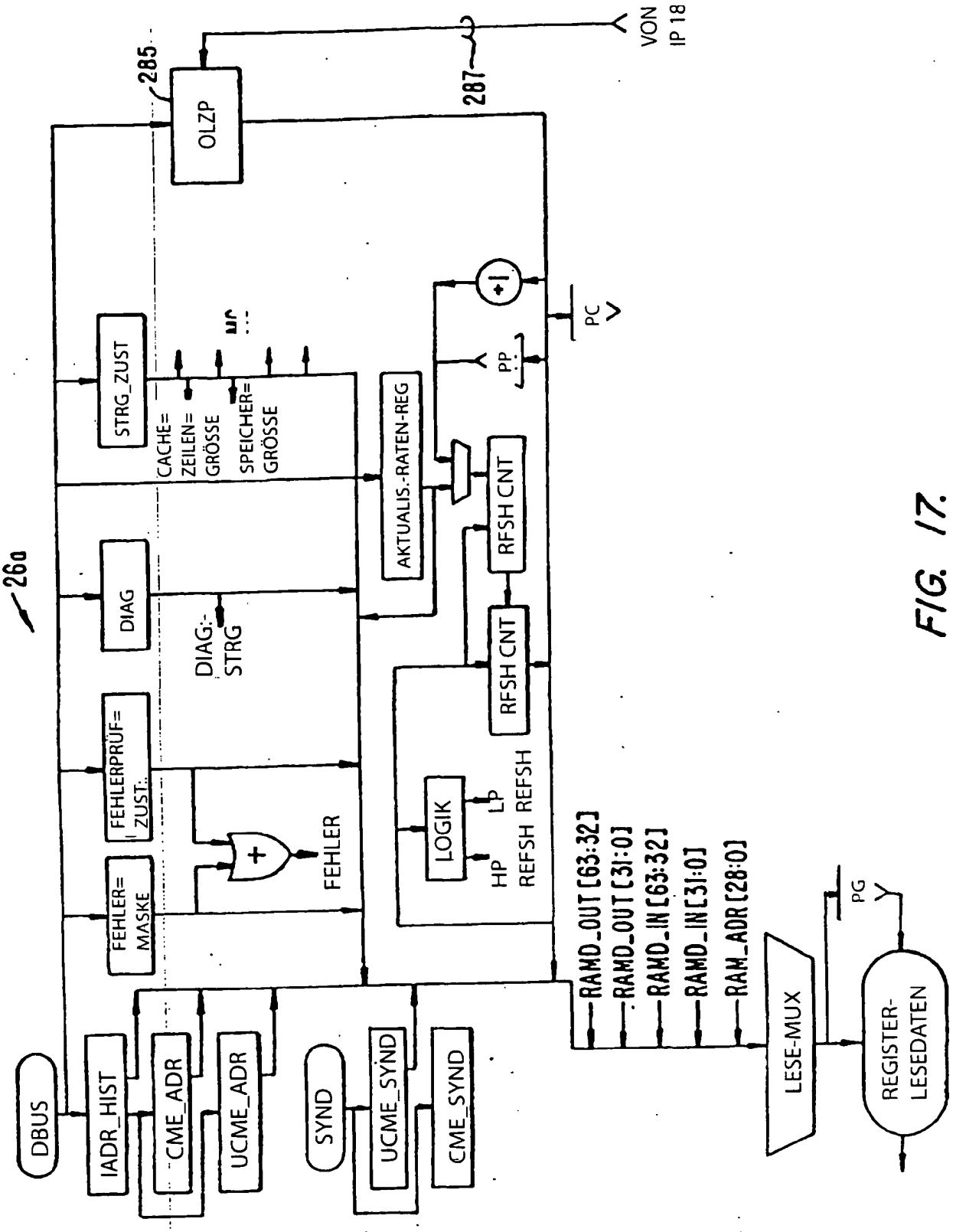

260
285
OLZP
287
VON IP 18
STRG_ZUST
CACHE= ZEILEN= GRÖSSE
SPEICHER= GRÖSSE
DIAG
DIAG:- STRG
FEHLERPRÜF= ZUST.
FEHLER= MASKE
FEHLER
AKTUALIS.-RATEN-REG
RFSH CNT
RFSH CNT
LOGIK
HP
LP
REFSH
REFSH
PC
PP
DBUS
IADR_HIST
CME_ADR
UCME_ADR
SYND
UCME_SYND
CME_SYND
RAMD_OUT[63:32]
RAMD_OUT[31:0]
RAMD_IN[63:32]
RAMD_IN[31:0]
RAM_ADR[28:0]
LESE-MUX
PG
REGISTER- LESEDATEN

FIG. 17.

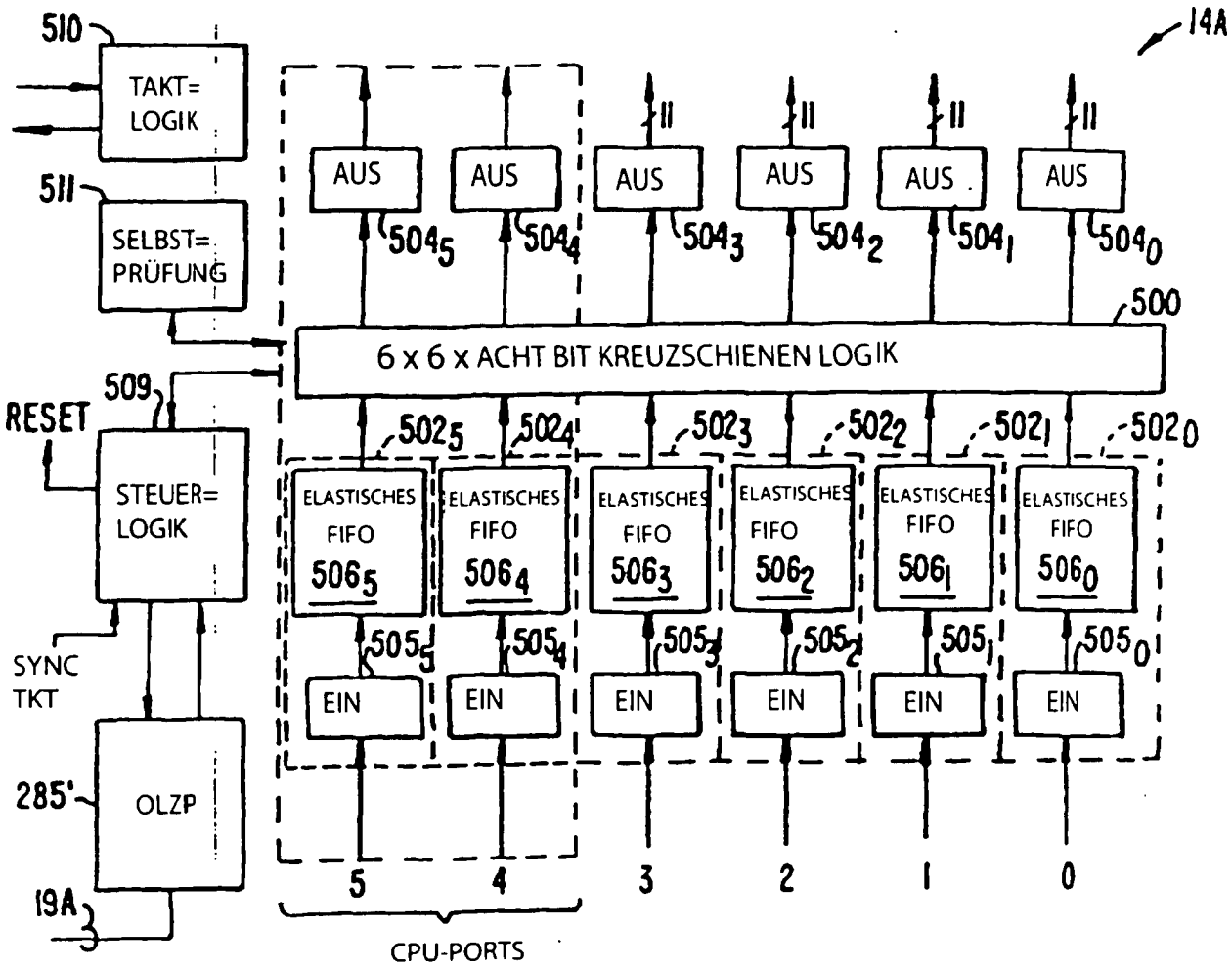


FIG. 19A.

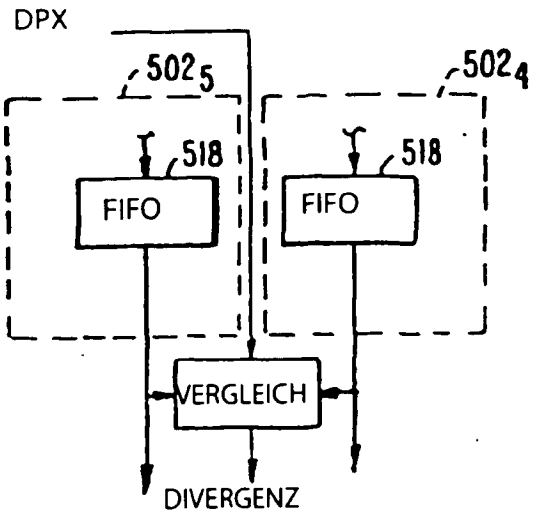


FIG. 19B.

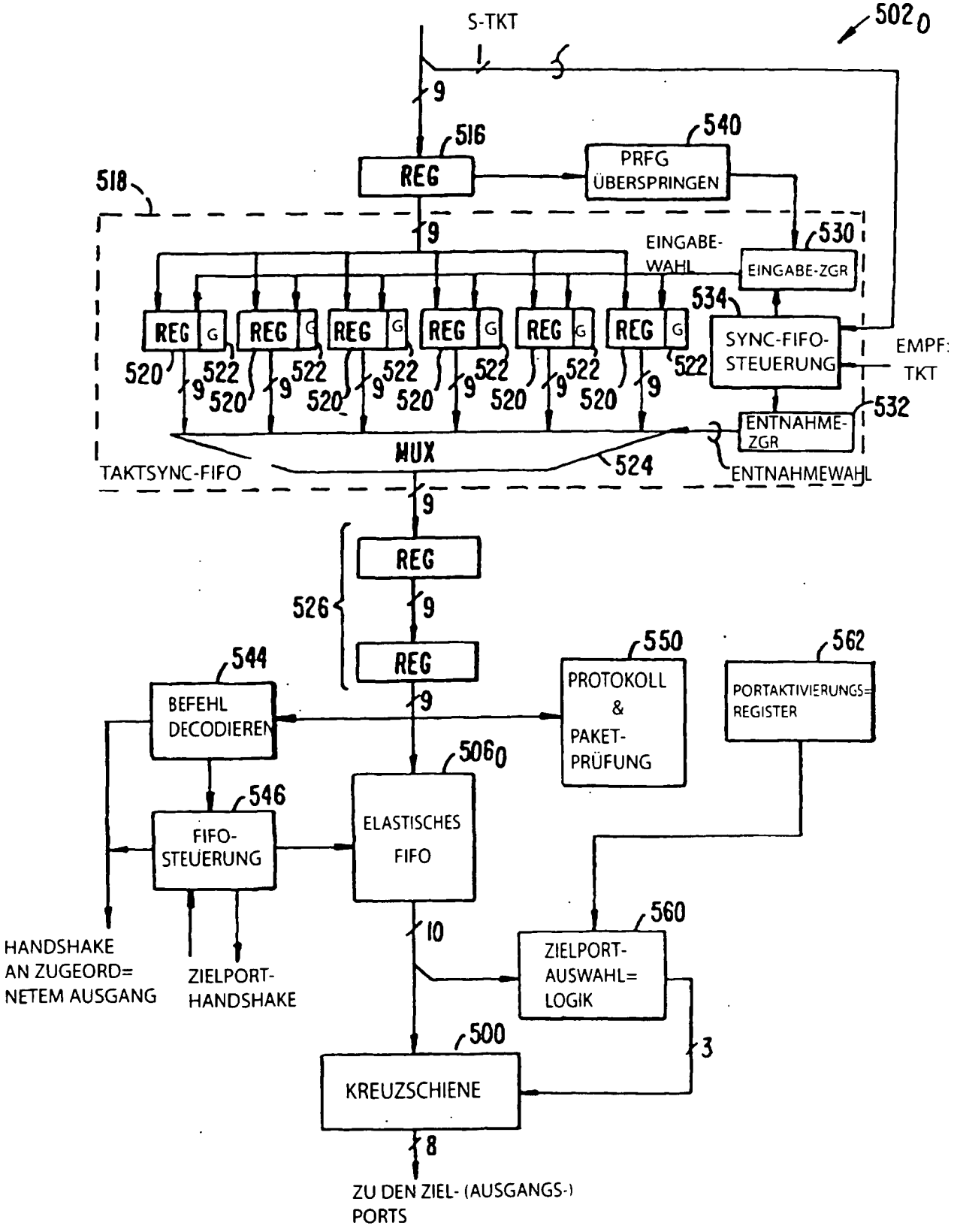

S-TKT
502 0
9
516
REG
540
PRFG ÜBERSPRINGEN
518
9
EINGABE-WAHL
530
EINGABE-ZGR
534
REG G
SYNC-FIFO-STEUERUNG
EMPF TKT
520
522
MUX
524
TAKTSYNC-FIFO
ENTNAHME-ZGR
532
ENTNAHMEWAHL
526
REG
544
BEFEHL DECODIEREN
550
PROTOKOLL & PAKET-PRÜFUNG
562
PORTAKTIVIERUNGS-REGISTER
506 0
546
FIFO-STEUERUNG
ELASTISCHES FIFO
10
560
ZIELPORT-AUSWAHL-LOGIK
3
500
KREUZSCHIENE
8
HANDSHAKE AN ZUGEORD-NETEM AUSGANG
ZIELPORT-HANDSHAKE
ZU DEN ZIEL- (AUSGANGS-) PORTS

FIG. 20A.

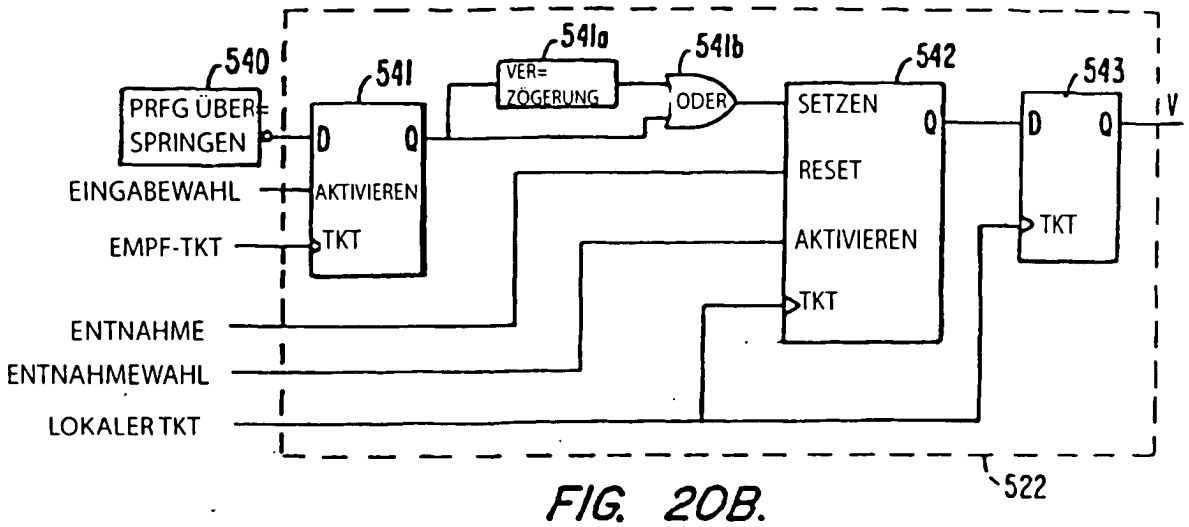


FIG. 20B.

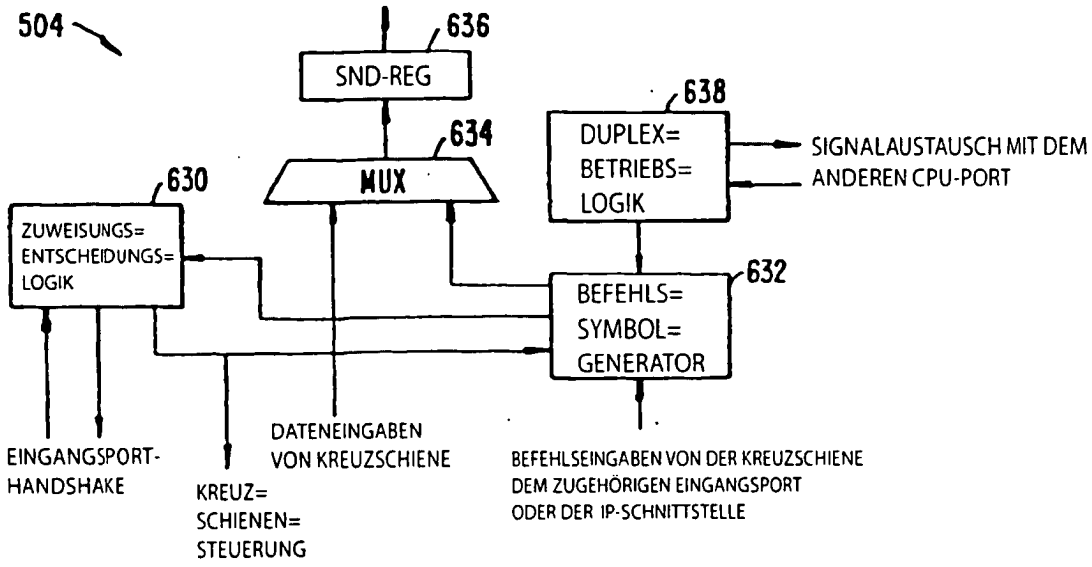


FIG. 22.

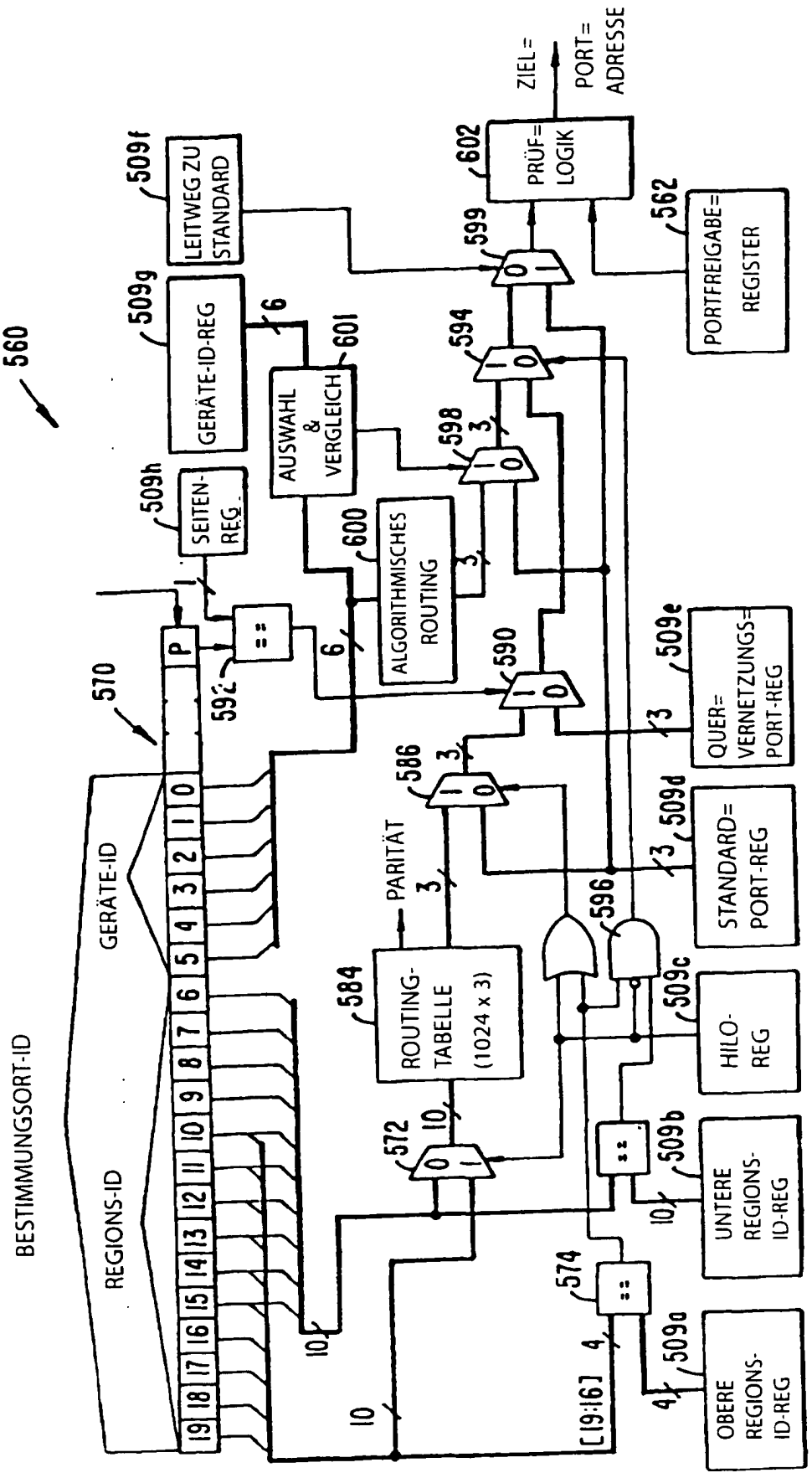

560
BESTIMMUNGSORT-ID
REGIONS-ID
GERÄTE-ID
570
P
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
592
==
509h
SEITEN-REG.
509g
GERÄTE-ID-REG
509f
LEITWEG ZU STANDARD
AUSWAHL & VERGLEICH
601
600
ALGORITHMISCHES ROUTING
584
ROUTING-TABELLE (1024 x 3)
PARITÄT
586
590
594
598
599
602
PRÜF= LOGIK
ZIEL=
PORT= ADRESSE
562
PORTFREIGABE= REGISTER
509e
QUER= VERNETZUNGS= PORT-REG
509d
STANDARD= PORT-REG
596
509c
HILO-REG
572
509b
UNTERE REGIONS-ID-REG
574
[19:16]
509a
OBERE REGIONS-ID-REG

FIG. 21A.

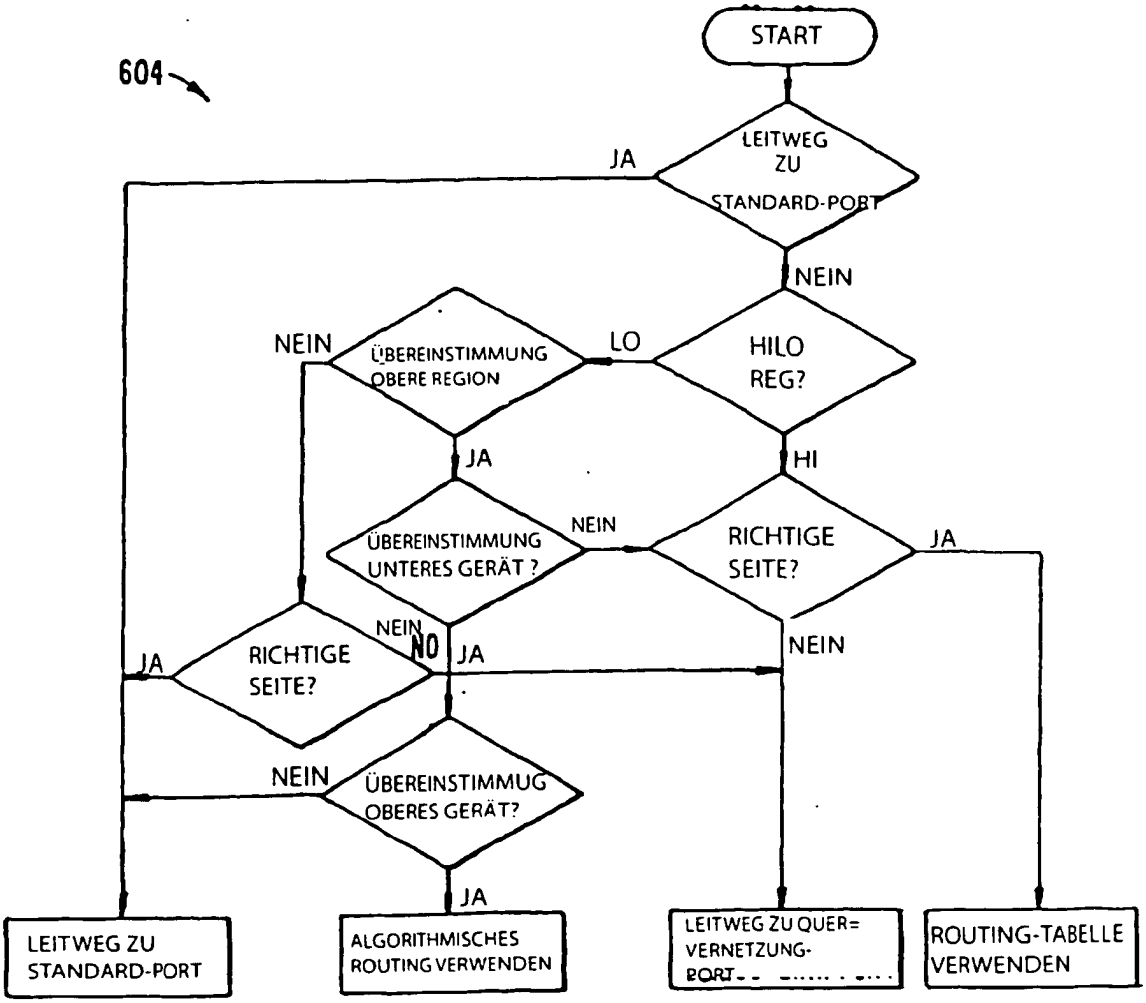


FIG. 21B.

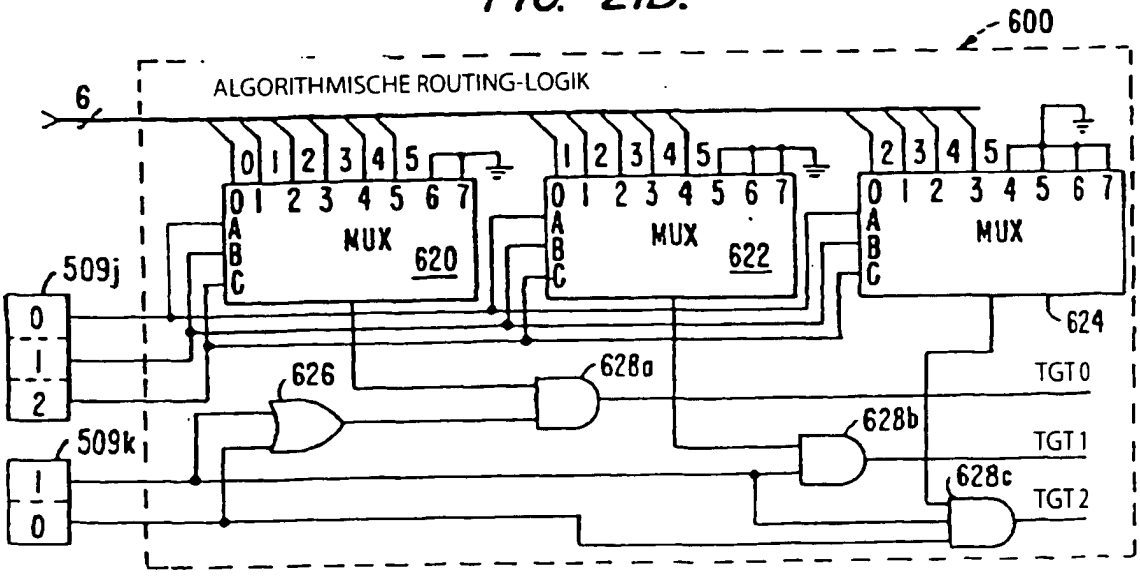


FIG. 21C.

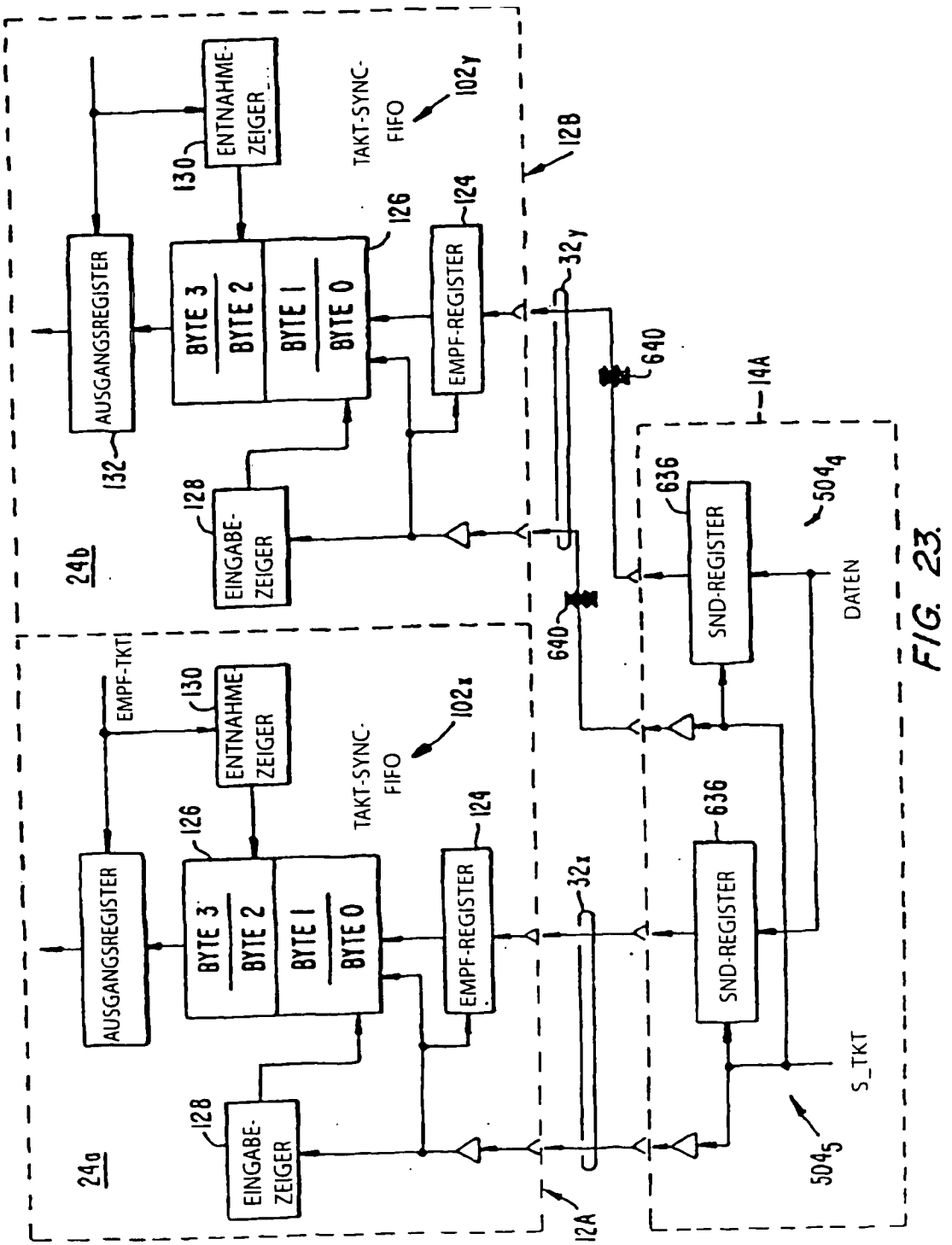

24a
24b
AUSGANGSREGISTER
BYTE 3
BYTE 2
BYTE 1
BYTE 0
EMPF-REGISTER
EINGABE-ZEIGER
ENTNAHME-ZEIGER
TAKT-SYNC-FIFO
EMPF-TKT
SND-REGISTER
DATEN
S_TKT
12A
12B
14A
126
124
128
130
132
102x
102y
32x
32y
636
640
504D
504S

FIG. 23

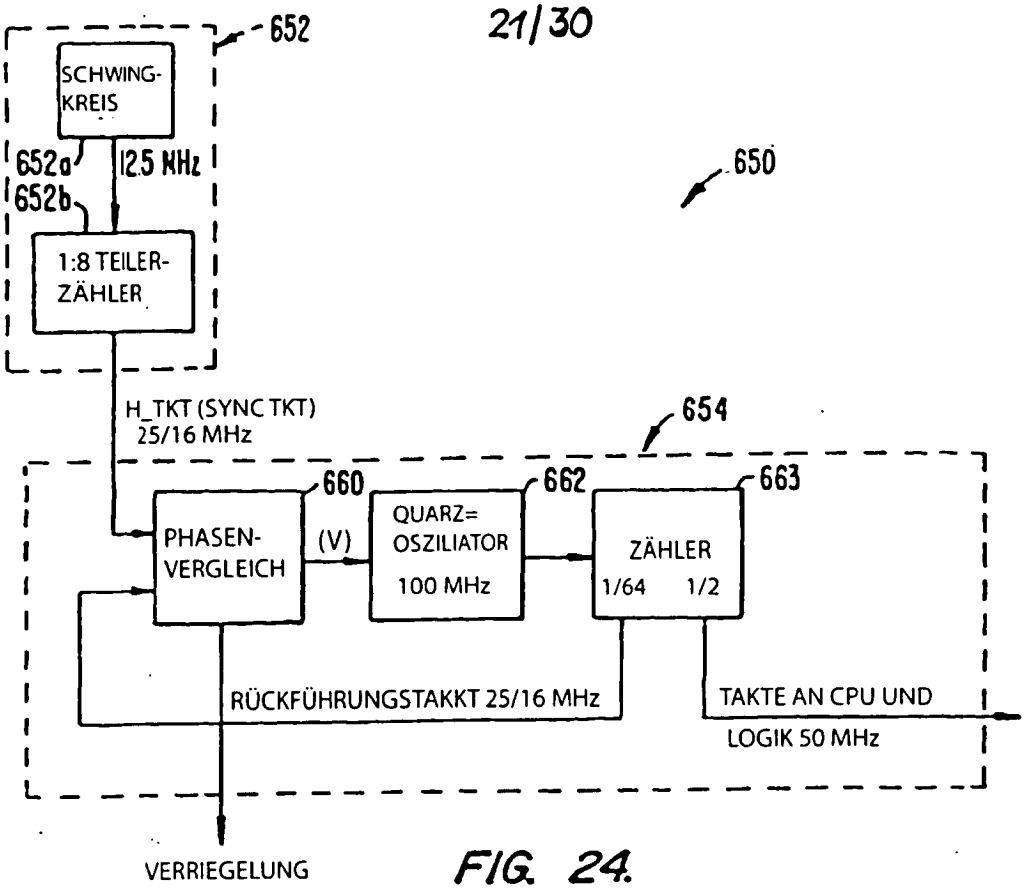

652
SCHWING-KREIS
652a
652b
12.5 MHz
1:8 TEILER-ZÄHLER
650
H_TKT (SYNC TKT)
25/16 MHz
654
660
662
663
PHASEN-VERGLEICH
(V)
QUARZ= OSZILIATOR
100 MHz
ZÄHLER
1/64 1/2
RÜCKFÜHRUNGSTAKKT 25/16 MHz
TAKTE AN CPU UND
LOGIK 50 MHz
VERRIEGELUNG

FIG. 24.

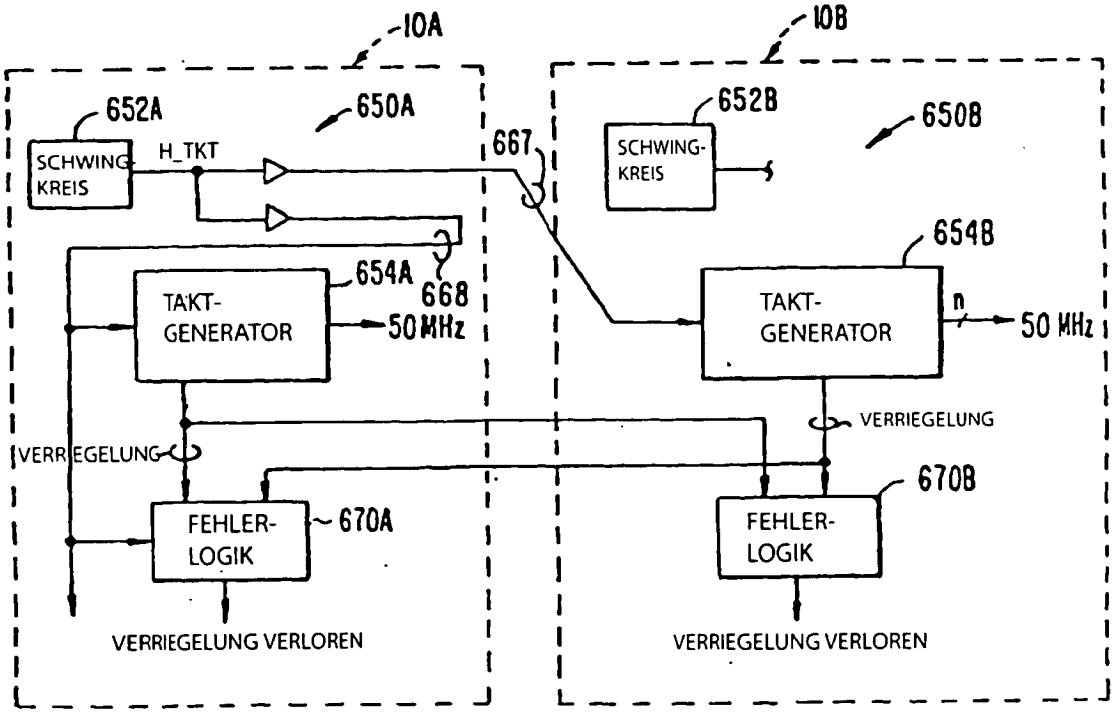

10A
10B
652A
650A
SCHWING-KREIS
H_TKT
667
652B
650B
SCHWING-KREIS
654B
654A
668
TAKT-GENERATOR
50 MHz
TAKT-GENERATOR
n
50 MHz
VERRIEGELUNG
VERRIEGELUNG
670B
FEHLER-LOGIK
670A
FEHLER-LOGIK
VERRIEGELUNG VERLOREN
VERRIEGELUNG VERLOREN

FIG. 25.

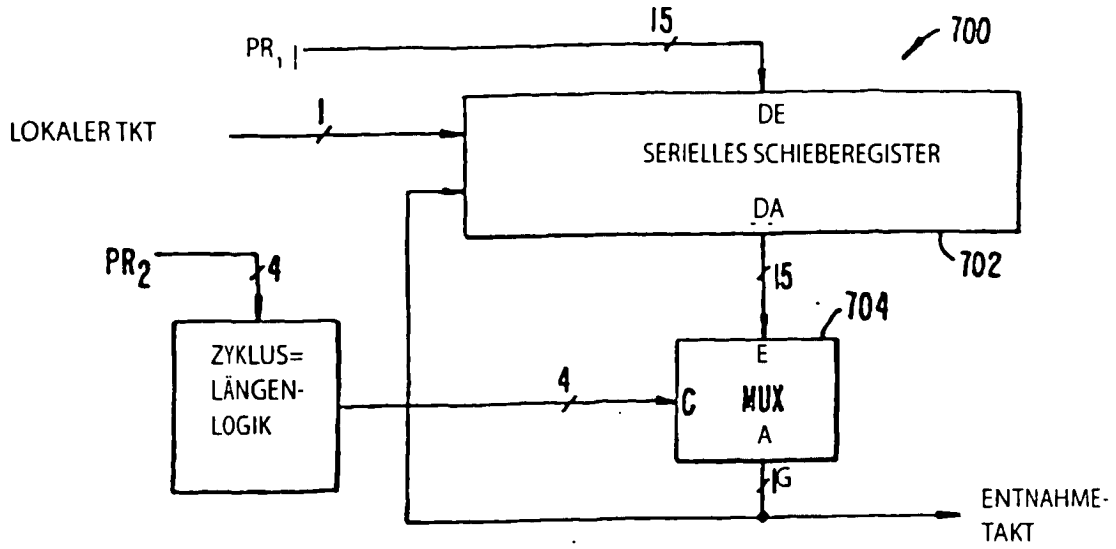


FIG. 26A.

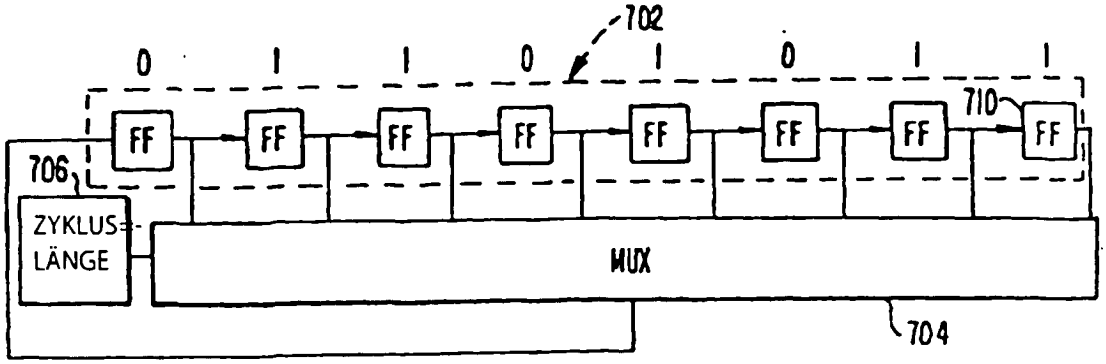


FIG. 26B.

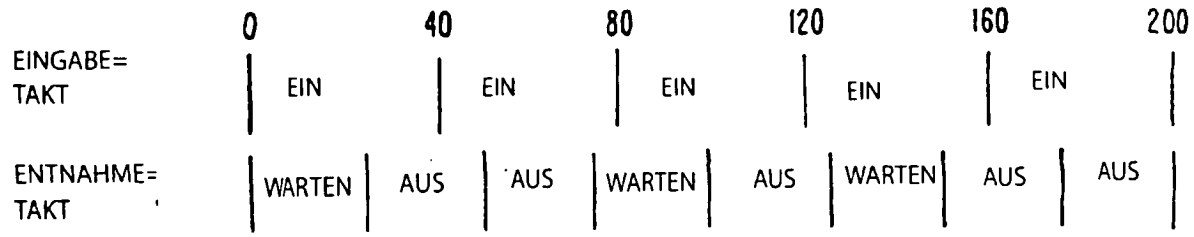


FIG. 27.

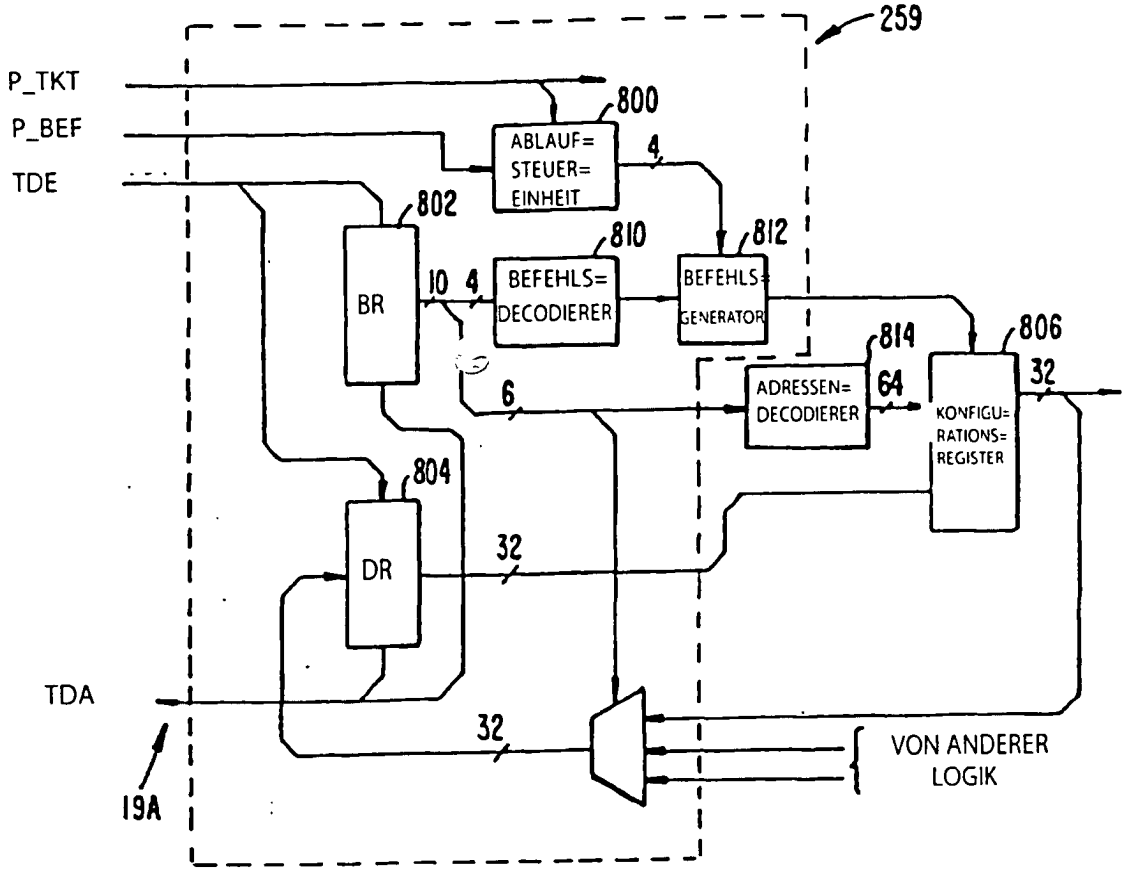

259
P_TKT
P_BEF
TDE
800
ABLAUF= STEUER= EINHEIT
4
802
BR
10
4
810
BEFEHLS= DECODIERER
812
BEFEHLS= GENERATOR
814
806
ADRESSEN= DECODIERER
64
32
KONFIGU= RATIONS= REGISTER
6
804
DR
32
TDA
32
VON ANDERER LOGIK
19A

FIG. 28.

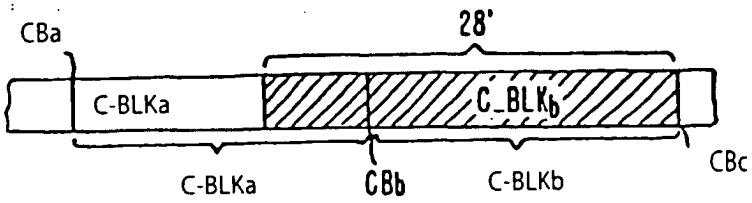

CBa
28'
C-BLKa
C_BLKb
CBc
C-BLKa
CBb
C-BLKb

FIG. 29.

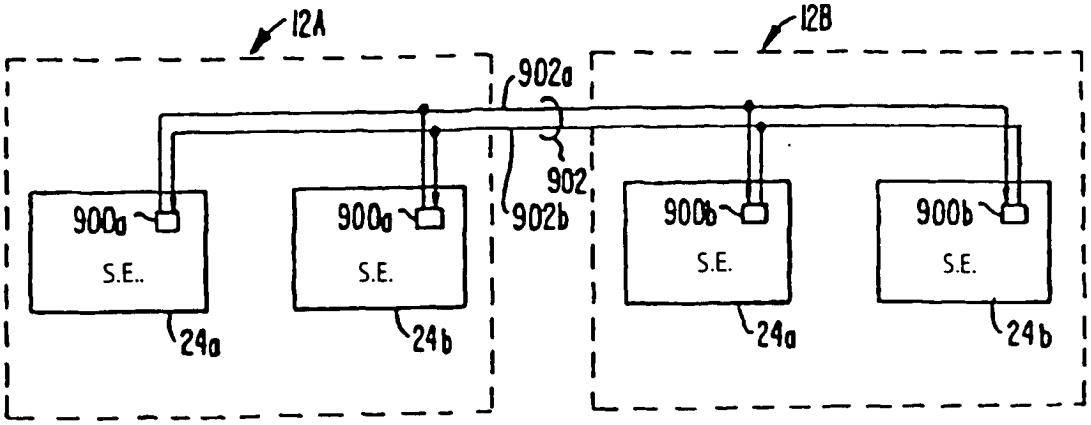


FIG. 30A.

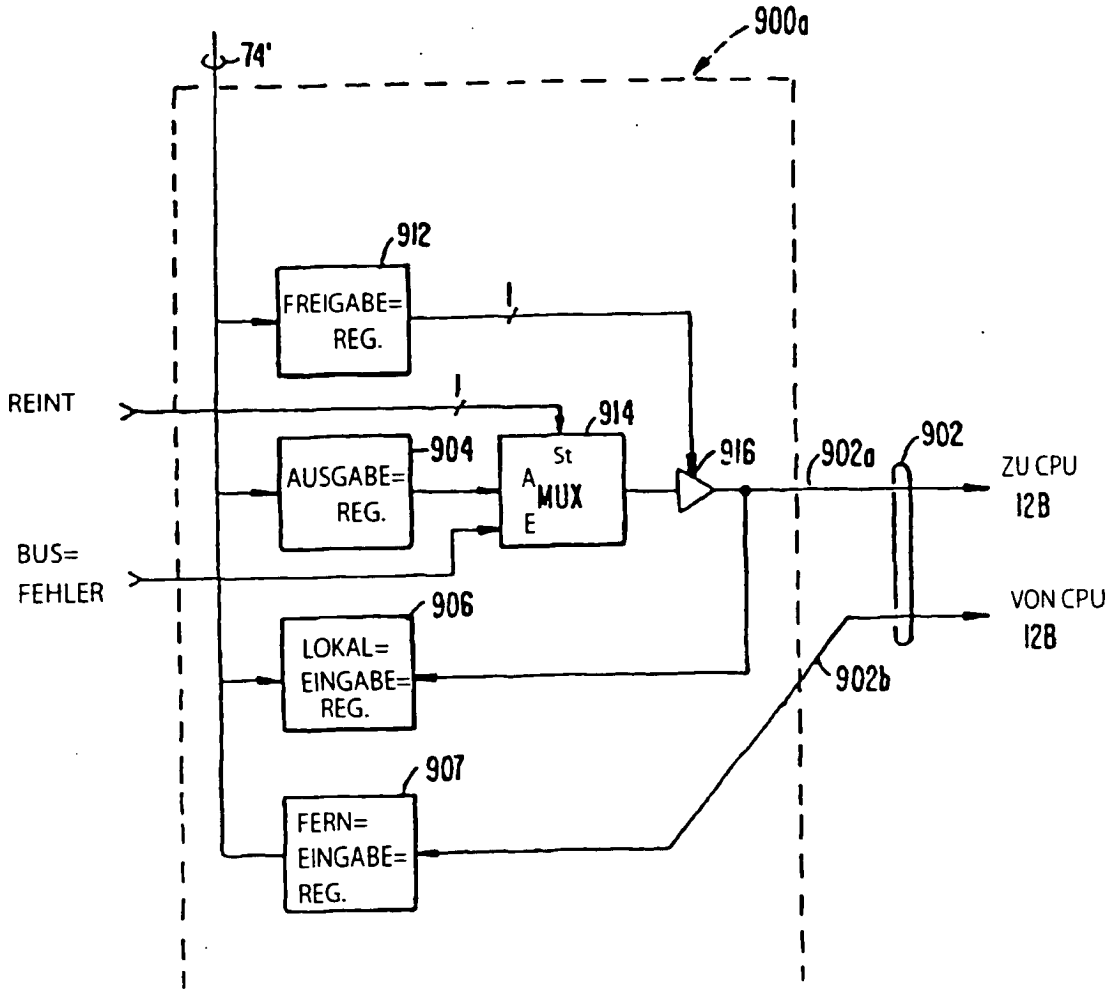


FIG. 30B.

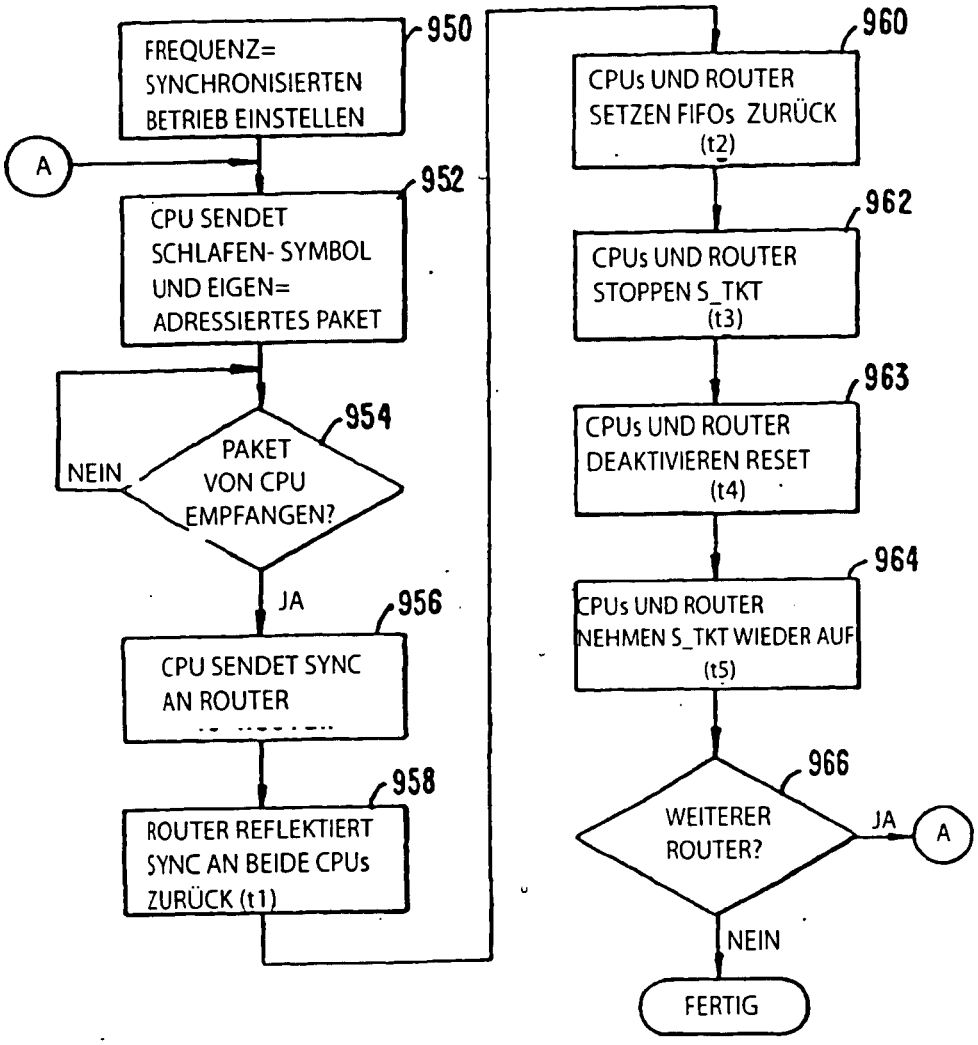


FIG. 31A.

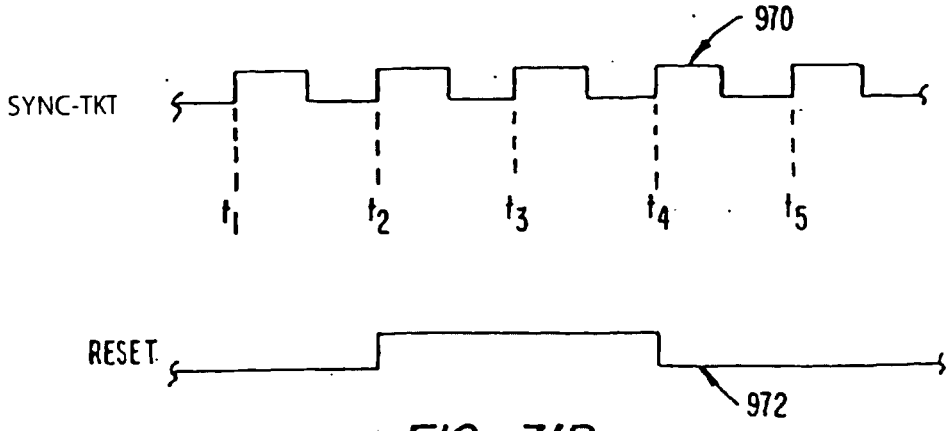


FIG. 31B.

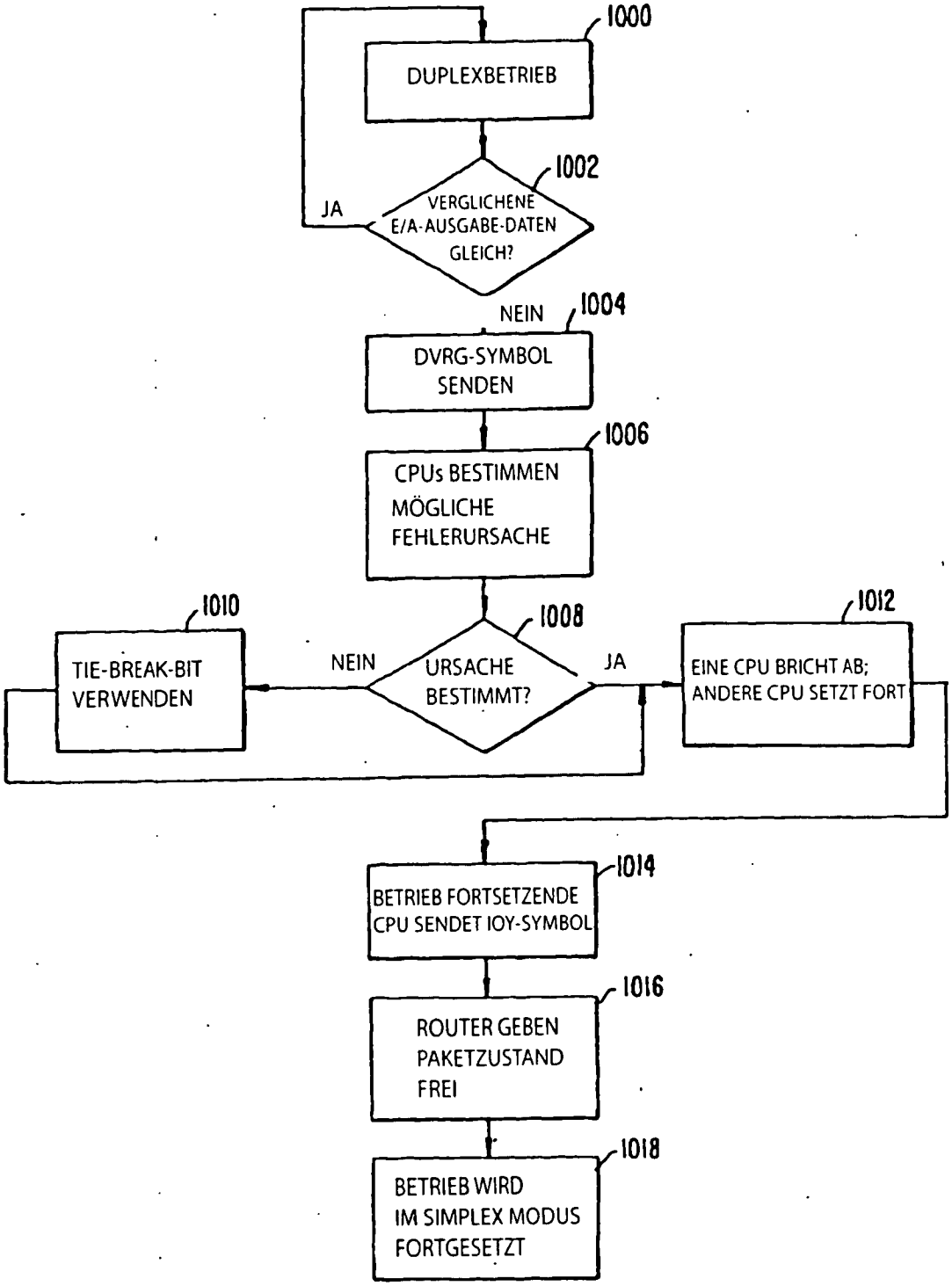

1000
DUPLEXBETRIEB
1002
VERGLICHENE E/A-AUSGABE-DATEN GLEICH?
JA
NEIN
1004
DVRG-SYMBOL SENDEN
1006
CPUs BESTIMMEN MÖGLICHE FEHLERURSACHE
1008
URSACHE BESTIMMT?
NEIN
JA
1010
TIE-BREAK-BIT VERWENDEN
1012
EINE CPU BRICHT AB; ANDERE CPU SETZT FORT
1014
BETRIEB FORTSETZENDE CPU SENDET IOY-SYMBOL
1016
ROUTER GEBEN PAKETZUSTAND FREI
1018
BETRIEB WIRD IM SIMPLEX MODUS FORTGESETZT

FIG. 32.

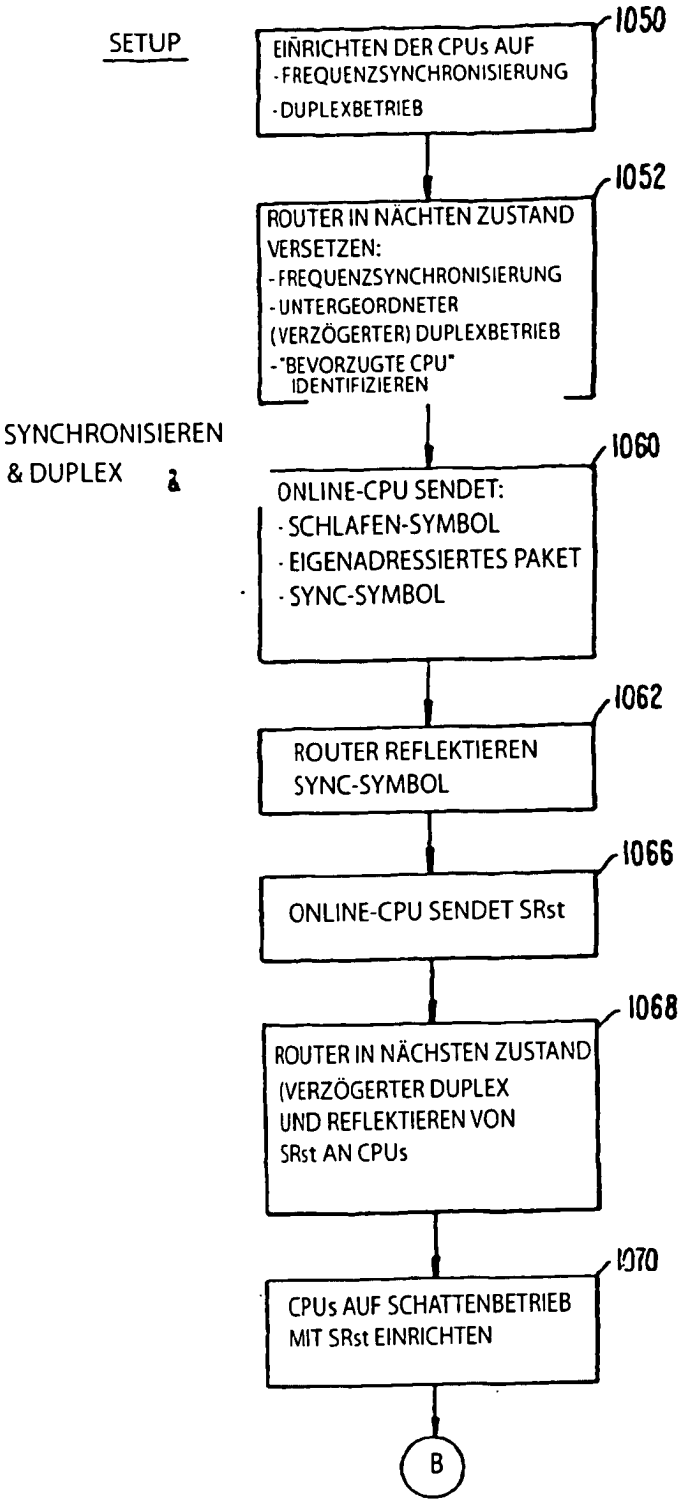


FIG. 33A.

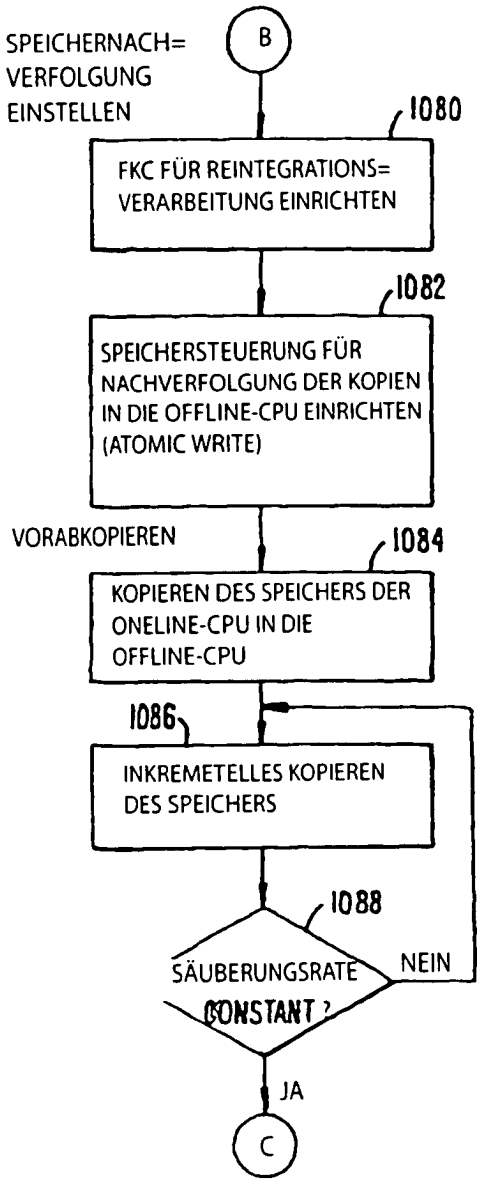


FIG. 33B.

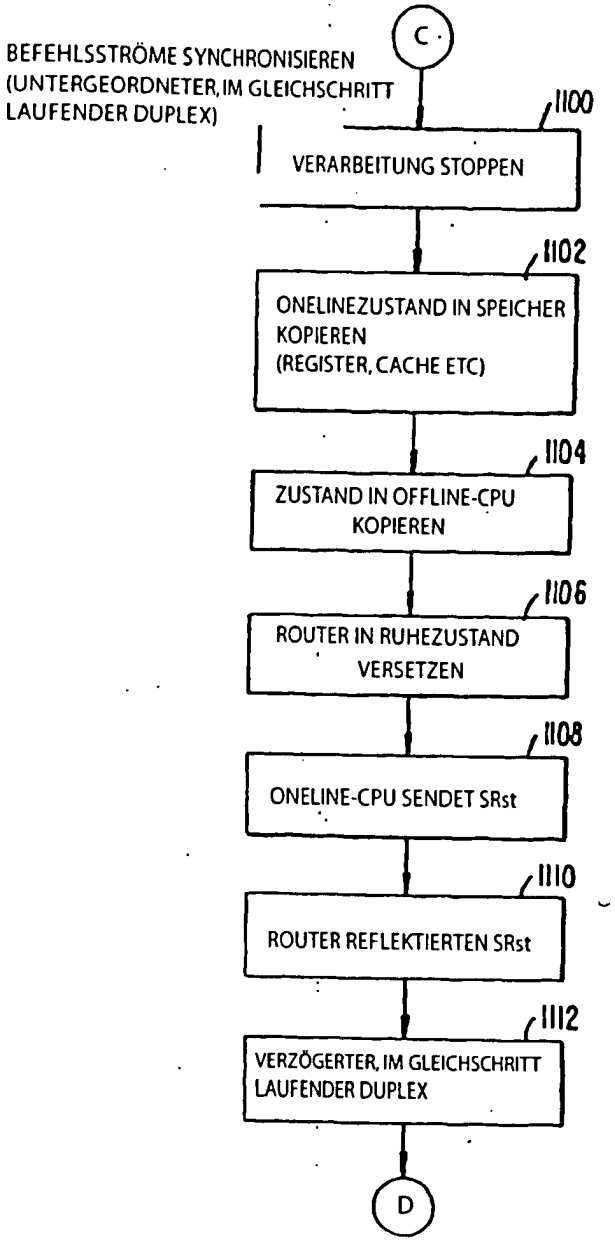


FIG. 33C.

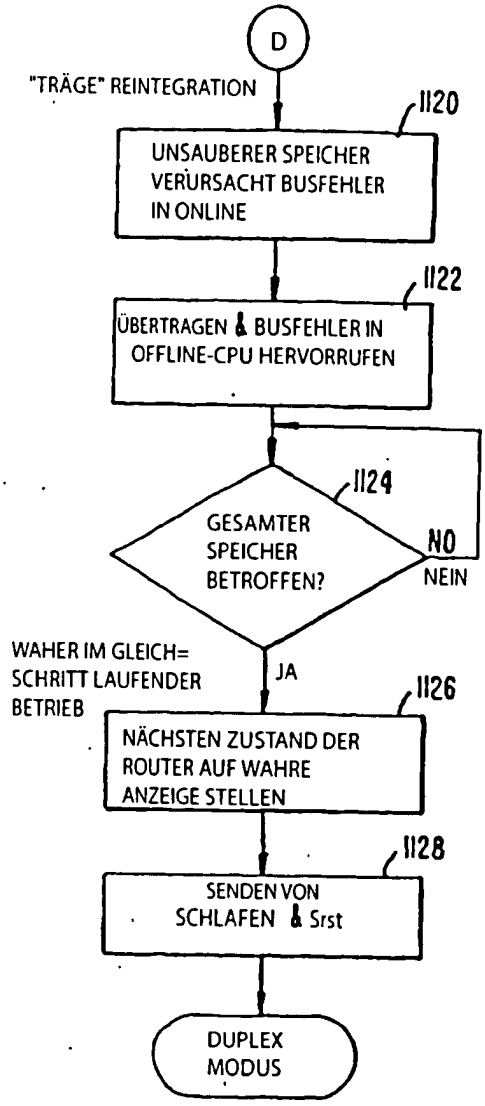


FIG. 33D.

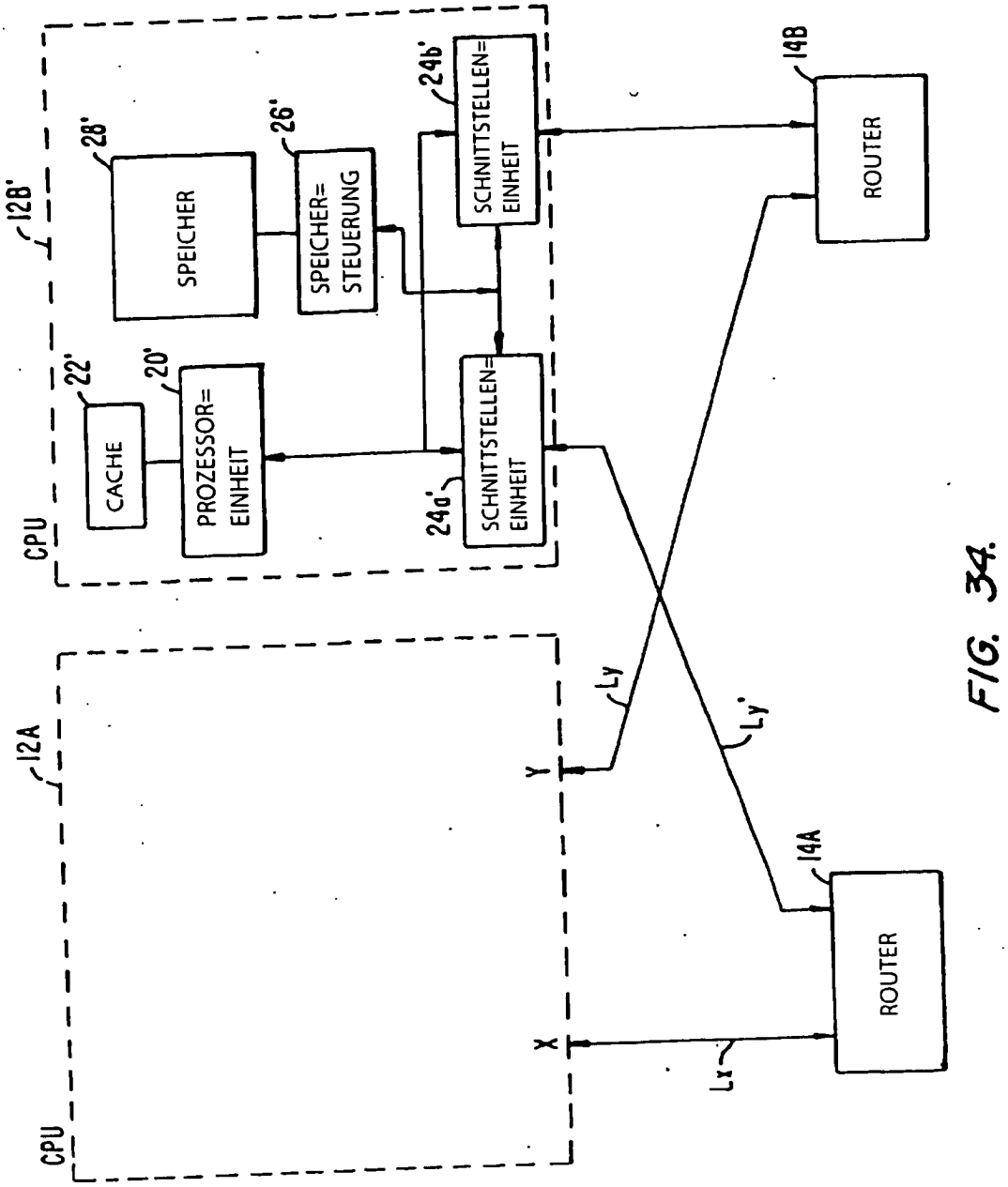

12B'
28'
SPEICHER
26'
SPEICHER= STEUERUNG
24b'
SCHNITTSTELLEN= EINHEIT
14B
ROUTER
22'
CACHE
20'
PROZESSOR= EINHEIT
24a'
SCHNITTSTELLEN= EINHEIT
CPU
12A
Ly
Ly'
Y
14A
ROUTER
X
Lx
CPU

FIG. 34.

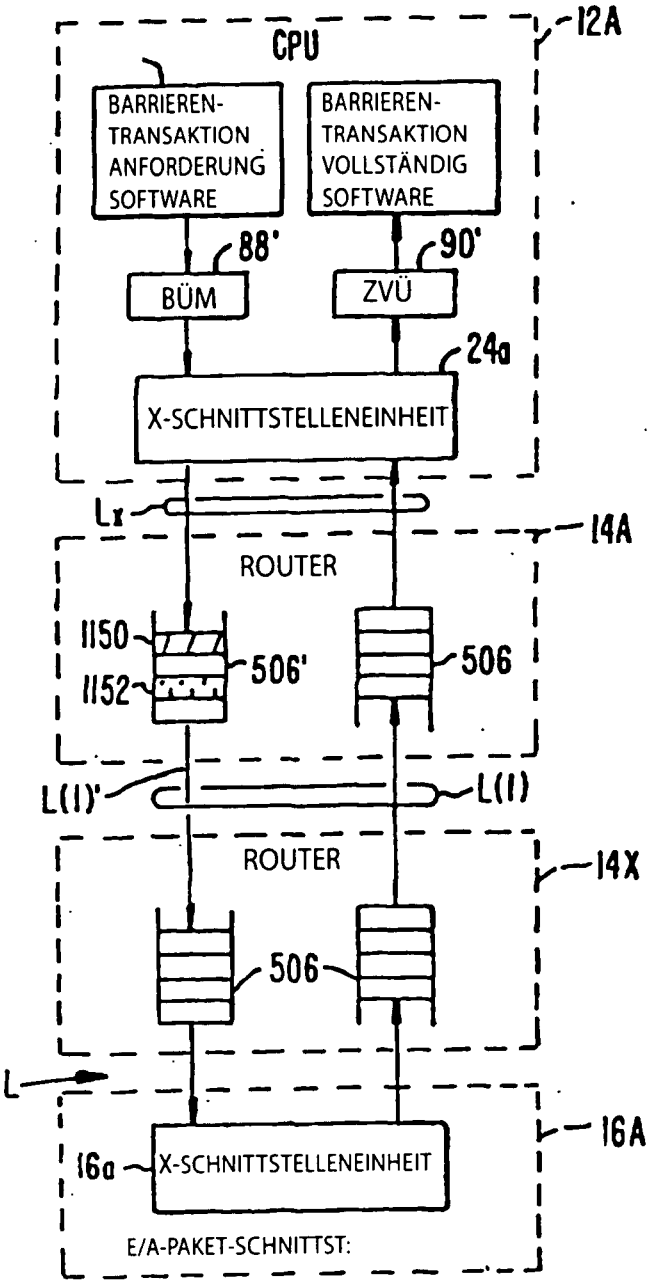

CPU
12A
BARRIEREN-TRANSAKTION ANFORDERUNG SOFTWARE
BARRIEREN-TRANSAKTION VOLLSTÄNDIG SOFTWARE
88'
90'
BÜM
ZVÜ
24a
X-SCHNITTSTELLENEINHEIT
Lx
14A
ROUTER
1150
1152
506'
506
L(1)'
L(1)
ROUTER
14X
506
L
16A
16a
X-SCHNITTSTELLENEINHEIT
E/A-PAKET-SCHNITTST:

FIG. 35.