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

(10) **DE 697 33 101 T2** 2005.09.29

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 811 934 B1**
(21) Deutsches Aktenzeichen: **697 33 101.6**
(96) Europäisches Aktenzeichen: **97 303 804.5**
(96) Europäischer Anmeldetag: **04.06.1997**
(97) Erstveröffentlichung durch das EPA: **10.12.1997**
(97) Veröffentlichungstag
der Patenterteilung beim EPA: **27.04.2005**
(47) Veröffentlichungstag im Patentblatt: **29.09.2005**

(51) Int Cl.[7]: **G06F 13/40**

(30) Unionspriorität:
658533 **05.06.1996** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

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

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(72) Erfinder:
Pettey, Christopher J., Houston, Texas 77070, US;
Maclaren, John M., Cypress, Texas 77429, US

(54) Bezeichnung: **Datenüberlaufverwaltung in einem Rechnersystem**

Beschreibung

[0001] Die Erfindung bezieht sich auf eine Datenströmung in einem Computersystem.

[0002] Computersysteme umfassen allgemein eine oder mehrere Peripheral Component Interface (PCI) Busse, die ein spezielles Kommunikationsprotokoll zwischen peripheren Komponenten, wie beispielsweise Video-Steuereinheiten und Netzwerk-Schnittstellen-Karten, und dem Hauptspeicher des Computersystems vorsehen. Wenn ein Systemspeicher und periphere Komponenten (PCI-Vorrichtungen) auf unterschiedlichen Bussen vorhanden sind, ist eine Brücke erforderlich, um den Fluss von Daten-Transaktionen zwischen den zwei Bussen zu verwalten. Eine PCI-Bus-Architektur ist durch die PCI Local Bus Specification, Revision 2.1 („PCI Spec 2.1"), veröffentlicht im Juni 1995, durch die PCI Special Interest Group, Portland, Oregon, die hier unter Bezugnahme darauf eingeschlossen wird, definiert. Eine PCI-zu-PCI-Brücken-Architektur ist durch die PCI-to-PCI-Bridge Architecture Specification, Revision 1.0 („PCI Bridge Spec 1.0"), veröffentlicht im April 1994, durch die PCI Special Interest Group, die hier unter Bezugnahme darauf eingeschlossen ist, definiert.

[0003] Unter den PCI Spec 2.1 und PCI Bridge Spec 1.0 Architekturen unterstützen PCI-Brücken zwei Typen von Transaktionen: gepostete Transaktionen (umfassend alle Speicher-Schreibzyklen), die auf dem initiierenden Bus abschließen, bevor sie auf dem Target-Bus abschließen, und verzögerte Transaktionen (umfassend Speicher-Lese-Anforderungen und I/O- und Konfigurations-Lese/Schreib-Anforderungen), die auf dem Target-Bus abschließen, bevor sie auf dem initiierenden Bus abschließen. Allgemein wird einer Vorrichtung, die eine gepostete Schreib-Transaktion initiiert, eine finite Menge an Pufferraum in der PCI-Brücke gegeben, um Daten, die geschrieben werden sollen, zu der Targetvorrichtung zu posten. Falls die Menge an Daten, die geschrieben werden soll, den zugeordneten Pufferraum übersteigt, wird die Transaktion beendet, wenn der zugeordnete Raum gefüllt ist, und die anfordernde Vorrichtung muss warten, bis sie wieder eine Kontrolle über den PCI-Bus erhält, um die gepostete Schreib-Transaktion abzuschließen.

[0004] Die EP 0710913 offenbart ein Computersystem, in dem eine CPU mit einem Systemspeicher durch eine Host-Brücke verbunden ist, die auch mit einer PCI-PCI-Brücke zwischen zwei PCI-Bussen, verbunden mit peripheren Vorrichtungen, verbunden ist. Die PCI-PCI-Brücke steuert Transaktionen zwischen peripheren Vorrichtungen, die mit einem PCI-Bus verbunden sind, und peripheren Vorrichtungen, die mit dem anderen PCI-Bus verbunden sind.

[0005] Die US 5363485 offenbart eine Bus-zu-Bus-(BTB)-Schnittstelle zwischen zwei Bussen, wobei mit jedem ein Mikrocomputer und eine Speichervorrichtung verbunden ist. Ein zentraler Puffer in der BTB-Schnittstelle umfasst Mehrfachkanal-FIFO-Vorrichtungen zum Speichern von Daten während einer Übertragungstransaktion zwischen den Bussen. Bevor Daten über eine Mehrkanalvorrichtung übertragen werden können, muss ein bestimmter Kanal spezifiziert werden, der die Daten enthalten wird. Eine zentrale Steuerung protokolliert und steuert einen Datenfluss durch die Mehrkanalvorrichtungen.

[0006] Gemäß der vorliegenden Erfindung wird ein Computersystem geschaffen, wie es in Anspruch 1 definiert ist.

[0007] Die vorliegende Erfindung schafft auch ein Verfahren zum Handhaben von Daten in einem Computersystem, wie es in Anspruch 10 definiert ist.

[0008] Weitere Ausführungsformen der Erfindung sind in den beigefügten, abhängigen Ansprüchen spezifiziert.

[0009] Die Puffer-Management-Logik kann den zweiten Transaktionspuffer nur dann zuweisen, wenn der zweite Transaktionspuffer keine Daten enthält, die einer anderen Datentransaktion, gespeichert in der Brücke, zugewiesen sind. Die Puffer-Management-Logik kann einen dritten Transaktionspuffer, falls notwendig, zu der Transaktion zuweisen, wenn Daten, die der Transaktion zugeordnet sind, den zweiten Transaktionspuffer zum Überlaufen bringen. Nachdem die Brückenvorrichtung damit beginnt, die Daten, gespeichert in dem ersten Transaktionspuffer, zuzuführen, kann die Puffer-Management-Logik den ersten Transaktionspuffer, falls notwendig, zu der Datentransaktion erneut zuweisen, wenn die Daten, die der Transaktion zugeordnet sind, den zweiten Transaktionspuffer zum Überlaufen bringen. Die Transaktion, die den ersten Transaktionspuffer zum Überlaufen bringt, kann eine gepostete Speicher-Schreibtransaktion sein. Mindestens einer der Busse kann ein PCI-Bus sein, und die Brückenvorrichtung kann eine PCI-zu-PCI-Brücke sein.

[0010] Vorteile der Erfindung können ein oder mehrere der nachfolgenden sein. Einer Vorrichtung, die eine

Datenübertragung initiiert, kann mehr als ein ursprünglich zugeordneter Pufferraum gegeben werden. Die gesamte Effektivität des Computersystems kann erhöht werden, da die gesamten Daten, die übertragen werden sollen, übertragen werden können, bevor die Transaktion beendet ist, sogar dann, wenn die Menge an Daten, die übertragen ist, größer als eine vorbestimmte, zugelassene Menge ist. Die gesamte Effektivität des Computersystems kann weiterhin verbessert werden, da Datenflüsse wahrscheinlicher dann auftreten, wenn Datenübertragungen ermöglicht wird, dass sie über die vorbestimmte, zugelassene Menge „überlaufen".

[0011] Andere Vorteile und Merkmale werden aus der nachfolgenden Beschreibung und aus den Zeichnungen ersichtlich werden, in denen:

[0012] Fig. 1 zeigt ein Blockdiagramm eines Computersystems.

[0013] Fig. 2 zeigt ein Blockdiagramm eines Erweiterungskastens des Computersystems der Fig. 1.

[0014] Fig. 3 zeigt ein Blockdiagramm der Brücken-Chips in dem Computersystem.

[0015] Fig. 4 zeigt ein Blockdiagramm eines Warteschlangen-Blocks in jedem der Brücken-Chips.

[0016] Fig. 5 zeigt ein Blockdiagramm des Takt-Routing-Schemas in den Brücken-Chips.

[0017] Fig. 6 zeigt ein Blockdiagramm eines Taktgenerators in jedem der Brücken-Chips.

[0018] Fig. 7 zeigt ein Blockdiagramm einer Master-Kabel-Schnittstelle in jedem der Brücken-Chips zum Übertragen von Daten über ein Kabel, das die Brücken-Chips verbindet.

[0019] Fig. 8 zeigt ein Zeitabstimmungsdiagramm von Signalen in der Master-Kabel-Schnittstelle.

[0020] Fig. 9 zeigt ein Blockdiagramm einer Slave-Kabel-Schnittstelle in jedem der Brücken-Chips zum Aufnehmen von Daten, übertragen über das Kabel.

[0021] Fig. 10 zeigt ein Blockdiagramm von Logik-Erzeugungs-Eingangs- und Ausgangs-Hinweiszeigern für die empfangene Logik in der Slave-Kabel-Schnittstelle.

[0022] Fig. 11 zeigt ein Zeitabstimmungsdiagramm von Signalen in der Slave-Kabel-Schnittstelle.

[0023] Fig. 12 zeigt ein Zeitabstimmungsdiagramm der Eingangs- und Ausgangs-Hinweiszeiger und deren Beziehung zu den aufgenommenen Kabeldaten.

[0024] Fig. 13 zeigt ein Blockdiagramm der Plazierung von Flip-Flop's und Eingangs- und Ausgangsanschlußflächen in jedem der Brücken-Chips.

[0025] Fig. 14 zeigt eine Tabelle der Informationen, geführt durch das Kabel.

[0026] Fig. 15A zeigt eine Tabelle, die den Typ von Informationen darstellt, geführt durch die Kabelsignale, die Einzel-Adressenzyklus-Transaktionen zugeordnet sind.

[0027] Fig. 15B zeigt eine Tabelle, die den Typ von Informationen darstellt, geführt durch die Kabelsignale, zugeordnet zu Dual-Adressen-Zyklus-Transaktionen.

[0028] Fig. 16 zeigt eine Tabelle von Parametern, zugeordnet zu dem Kabel.

[0029] Fig. 17 zeigt ein Logikdiagramm einer Fehlererfassungs- und Korrekturschaltung.

[0030] Fig. 18 zeigt eine Parität-Prüf-Matrix zum Erzeugen von Prüf-Bits in der Fehlererfassungs- und Korrekturschaltung.

[0031] Fig. 19 zeigt eine Syndrom-Tabelle zum Erzeugen von Fix-Bits in der Fehlererfassungs- und Korrekturschaltung.

[0032] Fig. 20A zeigt ein Zustand-Diagramm, das ein Round-Robin Arbitrierungs-Schema darstellt.

[0033] Fig. 20B zeigt ein Zustandsdiagramm, dass ein Zwei-Niveau-Entscheidungs-Schema darstellt.

[0034] Fig. 21 zeigt ein logisches Diagramm einer Entscheidungseinrichtung in jedem der Brücken-Chips.

[0035] Fig. 22 zeigt ein Zustandsdiagramm einer Erteilungs-Zustand-Maschine in einer Entscheidungseinrichtung.

[0036] Fig. 23 zeigt ein Zustandsdiagramm eines Level eins einer Entscheidungs-Zustand-Maschine in der Entscheidungseinrichtung.

[0037] Fig. 24 zeigt eine Tabelle, die eine Erzeugung von neuen Erteilungs-Signalen basierend auf dem momentanen Master darstellt.

[0038] Fig. 25 zeigt ein Blockdiagramm einer Logik zum Erzeugen von Masken-Bits und Master-Indikations-Bits mit Multi-Threading-Fähigkeit.

[0039] Fig. 26A zeigt ein logisches Diagramm von Schaltungen zum Erzeugen der maskierten Bits.

[0040] Fig. 26B zeigt ein Blockdiagramm eines Computersystems mit Mehrfach-Schichten von Bussen.

[0041] Fig. 27A zeigt eine Seitenansicht einer Erweiterungskarte, eingesetzt in einen Schlitz.

[0042] **Fig.** 27B–C zeigen schematische Diagramme einer Hebel-Schaltung.

[0043] Fig. 28–31 zeigen schematische Diagramme einer Schaltung des Erweiterungskastens.

[0044] Fig. 32A zeigt ein Zustandsdiagramm von der Schaltung des Erweiterungskastens.

[0045] Fig. 32B zeigt Wellenformen für die Schaltung des Erweiterungskastens.

[0046] Fig. 33A zeigt ein schematisches Diagramm einer Schaltung des Erweiterungskastens.

[0047] Fig. 33B zeigt Wellenformen für die Schaltung des Erweiterungskastens.

[0048] Fig. 33C–H zeigen ein Zustandsdiagramm von der Schaltung des Erweiterungskastens.

[0049] Fig. 34 zeigt ein schematisches Diagramm einer Schaltung des Erweiterungskastens.

[0050] Fig. 35A zeigt ein Zustandsdiagramm von der Schaltung des Erweiterungskastens.

[0051] Fig. 35B zeigt Wellenformen von der Schaltung des Erweiterungskastens.

[0052] Fig. 36 zeigt ein schematisches Diagramm einer Schaltung des Erweiterungskastens.

[0053] Fig. 37 zeigt ein Flußdiagramm eines nicht-maskierbaren Unterbrecher-Handler's, aufgerufen in Abhängigkeit einer Erfassung eines Bus-Hängend-Zustands in dem Computer-System.

[0054] Fig. 38 zeigt ein Fluß-Diagramm eine BIOS-Programms, das durch ein Computersystem-Durchsichts-Ereignis aufgerufen wird.

[0055] Fig. 39 zeigt ein Fluß-Diagramm eines BIOS-Isolier-Programms, aufgerufen auf einen Bus-Hängend-Zustand oder das Computer-Durchsichts-Ereignis hin.

[0056] Fig. 40 zeigt ein Blockdiagramm eines Bus-Watcher's in jedem der Brücken-Chips.

[0057] Fig. 41 zeigt ein Zustandsdiagramm einer Logik in dem Bus-Watcher zum Zurückführen des Busses zu einem Leerlaufzustand.

[0058] Fig. 42 zeigt ein logisches Diagramm von Status-Signalen in dem Bus-Watcher.

[0059] Fig. 43 zeigt ein logisches Diagramm von Bus-Historie-FIFOs und Bus-Zustand-Vektor-FIFOs in der Fehler-Isolations-Schaltung.

[0060] Fig. 44 zeigt ein logisches Diagramm einer Schaltung zum Erzeugen von Bereitschafts-Signalen zum Anzeigen, wenn die Bus-Historie- und Zustand-Vektor-Informationen verfügbar sind.

[0061] Fig. 45 zeigt ein Flußdiagramm eines Programms zum Zuordnen einer Bus-Zahl zu einem eingeschalteten oder leeren Schlitz.

[0062] Fig. 46 zeigt ein Flußdiagramm eines Programms zum Zuordnen von Speicherraum für das Computersystem.

[0063] Fig. 47 zeigt ein Flußdiagramm eines Programms zum Zuordnen eines I/O-Raums für das Computersystem.

[0064] Fig. 48 zeigt ein Flußdiagramm eines Programms zum Handhaben einer neu eingeschalteten Karte.

[0065] Fig. 49 zeigt ein Blockdiagramm eines Konfigurationsraums für eine PCI-Brückenschaltung.

[0066] Fig. 50A zeigt ein Blockdiagramm eines Computersystems.

[0067] Fig. 50B zeigt einen Bus-Zahl-Zuordnungs-Baum.

[0068] Fig. 51 zeigt ein Blockdiagramm, das Konfigurations-Transaktionen vom Typ 0 und Typ 1 darstellt.

[0069] Fig. 52 zeigt eine Tabelle, die eine Auflistung einer Adresse von einem primären Bus zu einem sekundären Bus aufweist.

[0070] Fig. 53A und Fig. 53B zeigen ein logisches Diagramm einer Schaltung zum Handhaben von Konfigurations-Zyklen vom Typ 0 und Typ 1.

[0071] Fig. 54A zeigt ein Blockdiagramm einer Schaltung zum Speichern von Informationen, um eine Berechnung von Bus-Funktions-Parametern zu ermöglichen.

[0072] Fig. 54B zeigt ein Blockdiagramm von Vorabruf-Zählern.

[0073] Fig. 55 zeigt ein Blockdiagramm eines Computersystems.

[0074] Fig. 56 zeigt ein Blockdiagramm eines PCI-Entscheidungs-Schemas.

[0075] Fig. 57 zeigt ein schematisches Diagramm eines Puffer-Entleerungs-Logik-Blocks.

[0076] Fig. 58 zeigt ein schematisches Diagramm eines Kabel-Decodierers.

[0077] Fig. 59–Fig. 62 zeigen schematische Diagramme einer geposteten Speicher-Schreib-Warteschlange, umfassend eine Steuerlogik.

[0078] Fig. 63–Fig. 65 zeigen schematische Diagramme einer verzögerten Anforderungs-Warteschlange, umfassend eine Steuerlogik.

[0079] Fig. 66–Fig. 69b zeigen schematische Diagramme einer verzögerten Abschluss-Warteschlange, umfassend eine Steuerlogik.

[0080] Fig. 70–Fig. 74 zeigen schematische Diagramme und eine Tabelle einer Master-Zyklus-Entscheidungseinrichtung.

[0081] Fig. 75–Fig. 87 zeigen schematische und Zustand-Übergangs-Diagramme einer Warteschlange-Block-zu-PCI-Bus-Schnittstelle.

[0082] Fig. 88 zeigt ein schematisches Blockdiagramm, das Bus-Vorrichtungen darstellt, verbunden mit ei-

nem Expansions-Bus.

[0083] Fig. 89 zeigt ein schematisches Blockdiagramm, dass eine Schaltung darstellt, um Unterbrechungs-Anforderungen weiterzuführen.

[0084] Fig. 90 zeigt ein schematisches Diagramm einer Vorrichtungs-Auswahl-Logik.

[0085] Fig. 91–Fig. 94 zeigen schematische Blockdiagramme von Registern.

[0086] Fig. 95 zeigt eine grafische Darstellung, die Wellenformen für das Computersystem darstellt.

[0087] Fig. 96 zeigt ein schematisches Diagramm der im Multiplex-Betrieb arbeitenden Schaltung.

[0088] Fig. 97A–D zeigen schematische Diagramme des Unterbrechungs-Aufnahme-Blocks.

[0089] Fig. 98 zeigt ein schematisches Diagramm des Unterbrechungs-Ausgabe-Blocks.

[0090] Fig. 99 zeigt ein Diagramm, das die Zeitmultiplexverarbeitung von Unterbrechungs-Anforderungs-Signalen darstellt.

[0091] Fig. 100 zeigt ein Diagramm, das eine Unterbrechungs-Anforderungs-Auflistung darstellt.

[0092] Fig. 101 zeigt ein schematisches Blockdiagramm, dass Bus-Vorrichtungen darstellt, verbunden mit einem Erweiterungsbus.

ÜBERSICHT

[0093] In der folgenden Beschreibung geben alle Signal-Mnemoniken, gefolgt durch ein „#", „_" oder „!", oder diesen vorausgehend, invertierte, logische Zustände an.

[0094] Wie in Fig. 1 dargestellt ist, umfasst ein Computersystem **10** einen primären PCI-Bus **24**, der mit einem Brücken-Chip **26a** und einem Brücken-Chip **26b** verbunden ist, wobei beide davon von einem gemeinsamen Design **26** sind. Der Brücken-Chip **26a** ist mit einem Brücken-Chip **48a** über ein Kabel **31** verbunden und der Brücken-Chip **26b** ist mit dem Brücken-Chip **48b** über ein Kabel **28** verbunden. Die Brücken-Chips **48a** und **48b** sind von einem gemeinsamen Design **48**, was gemeinsam zu dem Design **26** ist, mit der Ausnahme, das das Design **26** ein eingangsseitiger Mode ist und das Design **48** ein ausgangsseitiger Mode ist.

[0095] Der PCI-Bus **24** ist mit einem lokalen Bus **22** über eine System-Steuereinheit/Host-Brücken-Schaltung **28** schnittstellenmäßig verbunden. Die Systemsteuereinheit/Host-Brücken-Schaltung **18** steuert auch einen Zugriff zu einem Systemspeicher **20**, der auch mit dem lokalen Bus **22** zusammen mit der CPU **14** und einem Level-2-(L2)-Cachespeicher **16** verbunden ist.

[0096] Eine PCI Extended Industry Standard Architecture (EISA) Brücke **15** verbindet schnittstellenmäßig den PCI-Bus **24** mit einem ISA-Bus **17**. Sowohl eine Tastenfeld-Steuereinheit **21** als auch ein Read Only Memory (ROM) **23** sind mit dem ISA-Bus **17** verbunden. Ein nicht-flüchtiger Random Access Speicher (NVRAM) **70**, verbunden mit dem ISA-Bus **17**, speichert Informationen, die das Computersystem übernehmen sollte, wenn es abgeschaltet wird. Ein automatischer Server-Zurückgewinnungs-Zeitgeber **72** überwacht das Computersystem hinsichtlich einer Inaktivität. Falls sich das System verriegelt, wird der ASR-Zeitgeber **72** nach ungefähr 10 Minuten ablaufen. Ein Tastenfeld **19** wird durch die Tastenfeld-Steuereinheit **21** hinsichtlich einer Erfassung von niedergedrückten Tasten überwacht.

[0097] Wie Fig. 2 zeigt, bildet der Brücken-Chip **48a** eine Schnittstelle zu einem PCI-Bus **32a** und der Brücken-Chip **48b** bildet eine Schnittstelle zu einem PCI-Bus **32b**. Die PCI-Busse **32a** und **32b** sind an zwei Expansionskästen **30a** und **30b**, mit einem gemeinsamen Design **30**, angeordnet, und jeder Expansionskasten **30** besitzt sechs Hot-Plug-Schlitze bzw. Steckplätze **36** (**36a**–f), die dazu geeignet sind, herkömmliche Erweiterungskarten **807** aufzunehmen (Fig. 27A). Ein Schlitz **34** an dem Expansionskasten nimmt eine Karte **46** auf, die den Brücken-Chip **26** besitzt. Jeder Hot-Plug-Schlitz **36** besitzt zugeordnet eine Umschalt-Schaltung **41** zum Verbinden und Trennen des Schlitzes **36** mit und von dem PCI-Bus **32**. Sechs mechanische Hebel **802** werden dazu verwendet, selektiv die Karten **807** an entsprechenden Schlitzen zu sichern (wenn sie geschlossen oder verriegelt sind), wie dies weiterhin in dem US-Patent 5822196 mit dem Titel „Securing a Card in an

Electronic Device", angemeldet an demselben Datum wie diese Anmeldung, beschrieben ist. Jeder Erweiterungskasten **30** umfasst Register **52** und **82** zum Überwachen der Hebel **802** und von Status-Signalen des Erweiterungskastens **30** und ein Register **80** zum Kontrollieren einer Verbindung und einer Trennung von Schlitzen **36** und von dem PCI-Bus **32**.

[0098] Wie Fig. 3 zeigt, ist der Brücken-Chip so ausgelegt, um in Paaren **26** und **48** verwendet zu werden, um eine PCI-PCI-Brücke zwischen dem primären PCI-Bus **24** und dem sekundären PCI-Bus **32** zu bilden. Das Programmiermodel ist dasjenige von zwei hierarchischen Brücken. Zu der Systemsoftware hin erscheint das Kabel **28** als ein PCI-Bus, der exakt eine Vorrichtung enthält, und zwar den ausgangsseitigen Brücken-Chip **48**. Dies vereinfacht stark die Konfiguration der 2-Chip-PCI-PCI-Brücke, die den primären und sekundären Bus miteinander verbindet. Der Brücken-Chip **26**, der näher zu der CPU **14** hin liegt, verbindet den primären PCI-Bus **24** mit dem Kabel **28**. Die zweite PCI-PCI-Brücke **48** ist in dem Erweiterungskasten **30** vorhanden und verbindet das Kabel **28** mit dem sekundären Bus **32**. Ein Mode-Stift-Upstream-Chip bestimmt, ob der Brücken-Chip in dem ausgangsseitigen Mode oder dem eingangsseitigen Mode arbeitet. Einige Nicht-Brücken-Funktionen, wie beispielsweise ein Bus-Monitor **106** und eine Hot-Plug-Logik in einem SIO **50**, werden nur in dem Erweiterungskasten **30** verwendet und sind nicht in dem einlaufseitigen Mode-Chip **26** funktional.

[0099] Ein Taktgenerator **102** in dem Brücken-Chip **26** erzeugt Takte basierend auf dem Takt PCI CLK1 auf dem primären PCI-Bus **24**, wobei einer der erzeugten Takte über das Kabel **28** zu einem Taktgenerator **122** in dem ausgangsseitigen Brücken-Chip **48** vorgesehen wird. Der Taktgenerator **122** erzeugt die PCI-Takte in dem Erweiterungskasten **30** bei derselben Frequenz des primären PCI-Busses **24** und steuert ihn an, was dazu führt, dass beide Brücken-Chips **26** und **48** unter derselben Frequenz laufen. Der ausgangsseitige Brücken-Chip **48** läuft dem eingangsseitigen Brücken-Chip **24** in der Phase mit der Verzögerung des Kabels **28** hinterher. Eine asymetrische Grenze in dem eingangseitigen Brücken-Chip **26** an dem Punkt, wo Daten von dem Kabel **28** abgegriffen werden, ermöglicht, dass sich die Phasenverzögerung bei irgendeinem Wert befindet (und deshalb soll das Kabel von irgendeiner Länge sein), wobei das einzige Erfordernis dasjenige ist, dass die Frequenz der zwei Brücken-Chips dieselbe ist.

[0100] Die Kern-Logik jedes Brücken-Chips ist der Brücken-Logik-Block (**100** oder **120**), der einen PCI-Master (**101** oder **123**) zum Arbeiten als ein Master auf dem jeweiligen PCI-Bus, ein PCI-Target oder eine -Slave (**103** oder **121**) zum Arbeiten als eine Slave-Vorrichtung auf dem jeweiligen PCI-Bus, Konfigurations-Register (**105** oder **125**), die die Konfigurations-Informationen des entsprechenden Brücken-Chips enthalten, und einen Warteschlangen-Block (**107** oder **127**), der verschiedene Warteschlangen enthält, in denen Daten, zugeordnet zu Transaktionen zwischen dem primären PCI-Bus und dem sekundären PCI-Bus **32**, in die Warteschlange gestellt und gemanagt werden, aufweist. Die Daten, übertragen zwischen dem eingangsseitigen Brücken-Chip **26** und dem ausgangsseitigen Brücken-Chip **48**, werden durch Kabelschnittstellen **104** und **130** in den Brücken-Chips **26** und **48** jeweils gepuffert.

[0101] Eine Unterbrechungs-Programm-Logik ist auch in jedem Brücken-Chip enthalten. Dabei sind 8 Unterbrechungen, 6 von den Sekundär-Bus-Schlitzen, 1 von einer SIO-Schaltung **50** und 1 von dem ausgangsseitigen Brücken-Chip **48**, vorhanden. In dem ausgangsseitigen Chip **48** werden die Unterbrechungen durch einen Unterbrechungs-Aufnahme-Block **132** aufgenommen und entlang des Kabels **28** als eine serielle Datenfolge in sequenziellen Zeit-Stücken geschickt. In dem ausgangsseitigen Brücken-Chip **26** werden die Unterbrechungen durch einen Unterbrechungs-Ausgangs-Block **114** empfangen, der die Unterbrechungen zu einer Unterbrechungs-Steuereinheit weiterleitet.

[0102] Die SIO-Schaltung **50** liefert Steuersignale zum Beleuchten von LED's, zum Steuern eines Reset und für ein selektives Verbinden der Schlitze **36** mit dem Bus **32**. Sie umfasst auch eine Logik zum Lesen des Eingriffs-Status der Hebel **802** und des Status-Zustands der Karten **807** in jedem Schlitz **36**.

[0103] Die Brücken-Schaltung **26** umfasst einen Support für Unterbrechungen in dem Erweiterungskasten **30**, und, wenn sie in dem Host-System mit der anwendereigenen Schnittstelle zu einer Mehrfachkanal-Unterbrechungs-Steuereinheit installiert ist, schickt sie die Status-Zustände jeder Unterbrechung in einer seriellen Datenfolge. Die Brückenschaltung **26** kann auch so konfiguriert sein, um standardmäßige PCI-INTA, INTB, INTC und INTD Signale anzusteuern, falls sie in einem Standard-Schlitz in dem Host-System installiert ist.

[0104] Jeder Brücken-Chip umfasst auch einen PCI-Arbitrierer (**116** oder **124**) zum Steuern eines Zugriffs auf bis zu sieben Bus-Master. Da die eingangsseitige Brücke **26** in einem Schlitz installiert ist, wird der PCI-Arbitrierer **116** in dem eingangsseitigen Brücken-Chip **26** gesperrt. Jeder Brücken-Chip umfasst auch eine I^2C-Steuereinheit (**108** oder **126**) für eine Kommunikation mit Vorrichtungen, wie beispielsweise EEPROMs, Tempera-

tursensoren, usw., einen JTAG-Master (**110** oder **128**) zum Durchführen von Test-Zyklen, einen Bus-Monitor (**106** oder **127**) zum Messen einer Bus-Benutzung und einer Effektivität und der Effektivität des Vorabruf-Algorithmus des Brücken-Chips, und einen Bus-Beobachter (Bus-Watcher) (**119** oder **129**) zum Speichern einer Bus-Historie und von Status-Vektor-Informationen und zum Informieren der CPU **14** über einen Bus-Hängend-Zustand. Bestimmte Blöcke werden in jedem Brücken-Chip gesperrt, wenn sie nicht verwendet werden. In dem eingangsseitigen Brücken-Chip **26** werden Bus-Watcher **119**, der SIO **118**, der PCI-Arbitrierer **116** und der Bus-Monitor **106** gesperrt. Zusätzlich werden der die Unterbrechung aufnehmende Block **112** in dem eingangsseitigen Chip **126** und der Unterbrechungs-Ausgangs-Block **134** in dem ausgangsseitigen Chip **48** gesperrt.

ÜBERSICHT DES WARTESCHLANGEN-BLOCKS

[0105] Wie Fig. 4 zeigt, managen die Warteschlangen-Blöcke **107** und **127** Transaktionen die zwischen dem primären PCI-Bus **24** (in dem eingangsseitigen Chip) oder dem sekundären PCI-Bus **32** (in dem ausgangsseitigen Chip) und der Kabel-Schnittstelle **130** fließen. (Von hier an wird auf den ausgangsseitigen Brücken-Chip unter der Annahme Bezug genommen, dass der eingangsseitige Chip identisch arbeitet, ohne dass dies ansonsten angegeben ist). Der Warteschlangen-Block **127** umfasst einen Kabel-Decodierer **146**, der von der Kabel-Schnittstelle **130** Transaktionen aufnimmt, die auf dem sekundären PCI-Bus **32** abgeschlossen werden. Nach einem Decodieren einer Transaktion platziert der Decodierer **146** die Transaktionen, zusammen mit allen Informationen, umfasst in der Transaktion, in eine von drei Warteschlangen **140**, **142** und **144**. Jede Warteschlange enthält verschiedene Transaktions-Puffer, wobei jeder davon eine einzelne Transaktion speichert, und ist deshalb in der Lage, verschiedene Transaktionen simultan zu handhaben.

[0106] Die erste Warteschlange, eine gepostete Speicher-Schreib-Warteschlange (PMBQ) **140**, speichert gepostete Speicher-Schreib-Zyklen, ausgegeben durch die CPU, auf dem primären Bus zusammen mit allen Informationen, die erforderlich sind, um jeden Zyklus auf dem sekundären Bus **32** auszuführen. Die PMWQ **140** besitzt vier Transaktions-Puffer, wobei jeder davon eine gepostete Speicher-Schreib-Transaktion hält, die bis zu 8 Cache-Linien (256 Bytes) an Daten enthält. Unter bestimmten Umständen kann eine gepostete Speicher-Schreib-Transaktion, die mehr als acht Cache-Linien bzw. -Zeilen an Daten besitzt, in einen oder mehrere darauffolgende Puffer überlaufen, wie dies nachfolgend beschrieben ist.

[0107] Die zweite Warteschlange, eine verzögerte Anforderungs-Warteschlange (Delayed Request Queue – **142**), speichert verzögerte Anforderungs-Transaktionen (d. h. verzögerte Lese-Anforderungen (Delayed Read Requests – DRR), wie beispielsweise ein Speicher-Lesen (Memory Read – MR), eine Speicher-Lese-Linie (Memory Read Line – MRL), und Speicher-Lese-Mehrfach-(Memory Read Multiple – MRM)-Anforderungen; und in dem ausgangsseitigen Chip, Eingangs/Ausgangs-(I/O)-Lese-Schreib-Vorgänge und Konfigurationen (config – Lese/Schreib-Vorgänge), ausgegeben durch die CPU auf dem primären Bus, zusammen mit allen Informationen, erforderlich dazu, jede Transaktion auf dem sekundären Bus **32** auszuführen. Die DRQ **142** besitzt drei Transaktions-Puffer, wobei jeder davon in der Lage ist, ein Doppelwort, oder „dword", an Daten für verzögerte Schreibvorgänge zu halten.

[0108] Die dritte Warteschlange, eine verzögerte Abschluss-Warteschlange (Delayed Completion Queue (DCQ) **144**, speichert verzögerte Abschluss-Informationen, geliefert durch den eingangsseitigen Chip, in Abhängigkeit von verzögerten Anforderungs-Transaktionen, erzeugt auf dem sekundären Bus **32**. Für eine verzögerte Lese-Anforderung enthalten die entsprechenden Abschluss-Informationen die Lese-Daten, angefordert durch die initiierende Vorrichtung, und den Lese-Status (d. h. eine Indikation darüber, ob ein Paritäts-Fehler auf dem Target-Abort aufgetreten ist). Die verzögerten Abschluss-Informationen, zurückgeführt für eine verzögerte Schreib-Transaktion, sind dieselben wie diejenigen, zurückgeführt für eine verzögerte Lese-Anforderung, mit der Ausnahme, dass keine Daten für verzögerte Schreibvorgänge zurückgeführt sind. Da I/O- und config-Lese-Schreibvorgänge nur auf dem ausgangsseitigen Bus auftreten, wird nur die eingangsseitige DCQ verzögerte Abschluss-Informationen entsprechend zu einer von diesen Transaktionen enthalten. Die DCQ **144** besitzt acht Abschluss-Puffer, wobei jeder davon bis zu acht Cache-Linien bzw. Zeilen an Abschluss-Informationen für eine einzelne, verzögerte Anforderung halten kann. Zusätzlich zu den Abschluss-Informationen enthält jeder Abschluss-Puffer auch eine Kopie der verzögerten Anforderung, die die Informationen erzeugte. Für verzögerte Lese-Transaktionen kann eine Daten-„Folge" zwischen dem primären Bus **24** und dem sekundären Bus **32** eingerichtet werden, wenn die anfordernde Vorrichtung damit beginnt, die angeforderten Daten aufzusuchen, bevor die Target-Vorrichtung damit stoppt, sie zu der DCQ **144** zu liefern. Unter bestimmten Umständen wird die DCQ **144** automatisch zusätzliche Daten erneut aufsuchen, oder „vorababrufen" („prefetch"), wenn eine anfordernde Vorrichtung alle die angeforderten Daten von dem entsprechenden Puffer in der DCQ **144** aufsucht. Sowohl ein Streaming als auch ein automatisches Vorababrufen (prefetching) werden in weite-

rem Detail nachfolgend diskutiert.

[0109] Eine Warteschlangen-zu-PCI-Schnittstelle (Queue-to-PCI Interface – QPIF) **148** verwaltet Transaktionen, die von den Warteschlangen **140**, **142** und **144** zu dem PCI-Bus **32** und von dem PCI-Bus **32** zu der DCQ **144** und zu dem eingangsseitigen Chip über die Kabel-Schnittstelle **130** fließen. Die QPIF **148** tritt in einen „Master" Mode ein, um gepostete Speicher-Schreib- und verzögerte Anforderungs-Transaktionen von der PMWQ **140** und der DRQ **142** auf dem sekundären Bus laufen zu lassen. Für sowohl gepostete Speicher-Schreib- als auch verzögerte Lese-Transaktionen kann die QPIF **148** eine Transaktion „unterstützen", die weniger als eine Cache-Zeile an Daten einsetzt (d. h. eine Speicher-Schreib-(Memory Write – MW) oder eine Speicher-Lese-(Memory Read – MR) Transaktion), gegenüber einersolchen, die eine oder mehrere Cache-Zeilen (d. h. eine Speicher-Schreib- und Ungültigkeits-(MWI)-Transaktion oder eine Speicher-Lese-Zeile (MRL) oder eine Speicher-Lese-Mehrfach-(MRM) Transaktion erfordert, falls bestimmte Zustände erfüllt werden. Die QPIF **148** kann auch eine Lese-Transaktion, die eine Einzel-Cache-Zeile von Daten einsetzt (d. h. eine MRL Transaktion), in eine von mehreren Cache-Zeilen an Daten (d. h. eine MRM-Transaktion) umwandeln. Die QPIF **148** kann auch eine MRL- oder MRM-Transaktion „korrigieren", die in der Mitte einer Cache-Zeile beginnt, und zwar durch Lesen der gesamten Cache-Zeile und dann aussondern des nicht angeforderten Teils der Daten. Eine Transaktion-Unterstützung und eine Lese-Korrektur, wobei beide davon in weiterem Detail nachfolgend beschrieben sind, verbessern eine Systemeffektivität durch Verringern der Zeit, die dazu erforderlich ist, Daten von einer Speichervorrichtung aufzusuchen.

[0110] Die QPIF **148** gibt einen „Slave" Code ein, um Daten von der DCQ **144** zu einer anfordernden PCI-Vorrichtung zu liefern oder Transaktionen von dem PCI-Bus **32** zu der DCQ **144** und zu dem eingangsseitigen Chip über das Kabel zu schicken. Wenn die QPIF **148** eine gepostete Schreib-Transaktion von dem Bus **32** empfängt, führt sie die Transaktion weiter zu dem eingangsseitigen Chip, falls eine entsprechende Eine einer Gruppe von Transaktionszählern **159** anzeigt, dass die PMWQ in dem anderen Brücken-Chip nicht voll ist, wie dies nachfolgend diskutiert ist. Wenn die QPIF **148** eine verzögerte Anforderung aufnimmt, führt sie zuerst die Anforderung zu der DCQ **144** weiter, um zu bestimmen, ob die Transaktion bereits in der DCQ platziert worden ist, und, falls dies der Fall ist, ob die entsprechenden, verzögerten Abschluss-Informationen zu der DCQ **144** zurückgeführt worden sind. Falls die verzögerten Abschluss-Informationen in der DCQ vorhanden sind, werden die Informationen zu der anfordernden Vorrichtung geführt und die Transaktion wird beendet. Falls die Anforderung bereits in eine Warteschlange gestellt ist, allerdings die Verzögerungs-Abschirmungs-Informationen nicht zurückgeführt worden sind, wird die anfordernde Vorrichtung erneut versucht und die Transaktion wird auf dem PCI-Bus **32** beendet. Falls die Transaktion noch nicht in die Warteschlange gestellt ist, reserviert die DCQ **144** einen Abschluss-Puffer für die Transaktion und die QPIF **148** führt die Transaktion zu dem eingangsseitigen Chip über die Kabel-Schnittstelle **130** weiter, solange wie der entsprechende Transaktionszähler **149** anzeigt, dass der andere Brückenchip nicht voll ist.

[0111] Falls die DCQ **144** bestimmt, dass einer deren Pufferdaten enthält, vorgesehen für eine anfordernde Vorrichtung, allerdings unterschiedlich als die Daten, die in der momentanen Transaktion angefordert sind, kann der Puffer geleert werden, um zu verhindern, dass der anfordernde Master überholte Daten empfängt. Der Puffer wird dann geleert, wenn er Prefetch-Daten enthält, (d. h. Daten, die in dem Puffer verbleiben, nachdem die anfordernde Vorrichtung einige der Daten aufgesucht hat, oder Daten, die nicht spezifisch durch die Vorrichtung angefordert wurden), allerdings wird er nicht gelöscht, wenn er Abschluss-Daten enthält (d. h. spezifisch angefordert durch eine Vorrichtung, die bis jetzt noch nicht zurückgekehrt ist, um sie zu empfangen). Falls der Puffer-Abschluss-Daten enthält und die anfordernde Vorrichtung eine Anforderung ausgegeben hat, die nicht den Puffer „trifft" („hit"), versieht die DCQ **144** die Vorrichtung mit einem Zeichen als eine „Multi-Threaded" Vorrichtung (d. h. eine solche, die in der Lage ist, mehr als eine Transaktion zu einem Zeitpunkt beizubehalten), und ordnet einen anderen Abschluss-Puffer der neuen Anforderung zu. Die Puffer-Lösch- und Mehrfach-Puffer-Zuordnungs-Schemen werden in weiterem Detail nachfolgend beschrieben.

[0112] Ein Master-Zyklus-Arbitrierer (MCA) **150** in dem Warteschlangen-Block **127** hält Standard-Reihenfolgen-Einschränkungen zwischen geposteten Speicher-Schreib-, verzögerten Anforderungs- und verzögerten Abschluss-Transaktionen bei, wie dies in der PCI Bridge Architecture Specification, Version 2.1, angegeben ist. Diese Einschränkungen erfordern, dass Bus-Zyklen eine starke Schreib-Reihenfolge beibehalten und dass Entblockungen (Deadlocks) nicht auftreten. Deshalb bestimmt der MCA **150** die Reihenfolge, in der gepostete Speicher-Schreib-Transaktionen in der PMWQ **140** und verzögerte Anforderungs-Transaktionen in der DRQ **142** auf dem PCI-Bus **32** laufen. Der MCA **150** steuert auch die Verfügbarkeit von verzögerten Abschluss-Informationen, gespeichert in der DCQ **144**. Um eine Übereinstimmung mit diesen Regeln sicherzustellen, gibt der ausgangsseitige MCA **150** jedem geposteten Speicher-Schreib-Zyklus eine Gelegenheit, früher ausgegebene, verzögerte Anforderungs-Zyklen im Bypass zu passieren, während sowohl der eingangsseitige als auch

der ausgangsseitige MCA **150** nicht zulassen, dass verzögerte Anforderungs- und verzögerte Abschluss-Zyklen früher ausgegebene, gepostete Speicher Schreib-Zyklen im Bypass passieren. Eine Transaktions-Reihenfolge durch den MCA **150** wird in weiterem Detail nachfolgend beschrieben.

[0113] Die Transaktions-Zähler **159** in dem ausgangsseitigem Warteschlangen-Block **127** behalten eine Zählung der Zahl von Transaktionen, in dem eingangsseitigen Brücken-Chip in die Warteschlange gestellt, bei. Ein Zähler **160** für ein gepostetes Speicher-Schreiben (Posted Memory Write – PMW) zeigt die Zahl von PMW-Transaktionen, gehalten in der eingangsseitigen, geposteten Speicher-Schreib-Warteschlange, an. Der PMW-Zähler **160** wird zu jedem Zeitpunkt erhöht, zu dem eine PMW-Transaktion zu der Kabel-Schnittstelle **130** geschickt wird. Der Zähler **160** wird zu jedem Zeitpunkt erniedrigt, zu dem die QPIF **148** ein Signal von dem Kabel-Decodierer **146** aufnimmt, anzeigend, dass ein PMW-Zyklus auf dem eingangsseitigen PCI-Bus **24** abgeschlossen worden ist. Wenn die eingangseitige PMWQ die maximale Zahl (vier) der PMW Transaktionen in die Warteschlange gestellt hat, gibt der PMW-Zähler **160** ein PMW-Voll-Signal (tc_pmw_full) aus, und teilt der QPIF **148** mit, zusätzliche PMW-Zyklen von dem PCI-Bus **32** erneut zu versuchen. In ähnlicher Weise zählt ein Zähler **161** für eine verzögerte Anforderung (Delayed Request – DR) die Zahl von DR-Transaktionen, gehalten in der eingangsseitigen, verzögerten Anforderungs-Warteschlange. Wenn die DRQ die maximale Zahl (drei) von DR-Transaktionen hält, stellt der DR-Zähler **161** ein DR-Voll-Signal (tc_dr_full) auf, das anzeigt, dass die QPIF **148** alle darauffolgenden DR-Transaktionen von dem PCI-Bus **32** erneut versuchen muss. Ein Zähler **162** für einen verzögerten Abschluss (Delayed Completion – DC), zählt die Zahl von verzögerten Abschlüssen, die in dem eingangsseitigen Master-Zyklus-Arbitrierer in die Warteschlange gestellt sind. Wenn der MCA die maximale Zahl (vier) von verzögerten Abschlüssen hält, stellt der DC-Zähler **162** ein DC-Voll-Signal (tc_dc_full) auf, das verhindert, dass die ausgangsseitige QPIF **148** verzögerte Anforderungs-Transaktionen auf dem sekundären PCI-Bus **32** laufen lässt. Sobald der Voll-Zustand verschwindet, können Informationen über den verzögerten Abschluss zu der ausgangsseitigen DCQ geschickt werden.

[0114] Ein PCI-Schnittstellen-Block **152** ist zwischen dem PCI-Bus **32** und dem Warteschlangen-Block **127** vorhanden. Die PCI-Schnittstelle **152** umfasst einen Master-Block **123** und einen Slave-(Target)-Block **121**. Der Slave-Block **121** ermöglicht PCI-Vorrichtungen auf dem Bus **32**, auf interne Register von Brücken-Chips (z. B. Target-Speicher-Bereichs-Register **155** und Konfigurations-Register) zuzugreifen, um Abschluss-Informationen zu beanspruchen, die in der DCQ **144** gespeichert sind, und um auch Transaktionen zu initiieren, die durch die QPIF **148** und die Kabel-Schnittstelle **130** zu dem primären Bus geführt werden. Der Slave-Block **121** steuert die Verfügbarkeit des PCI-Busses **32** zu den PCI-Vorrichtungen auf dem Bus **32** durch Erkennen, wann jede Vorrichtung auf deren REQ#-Zeile zugreift und die REQ#-Signale zu den PCI-Arbitrierer **124** weiterführt. Wenn der PCI-Arbitrierer **124** eine anfordernde Vorrichtung auswählt, um eine Steuerung des Busses zu empfangen, erteilt der Slave-Block **121** den Bus zu der Vorrichtung durch Aufstellen der GNT# Leitung der Vorrichtung. Sobald der Bus **32** zu der anfordernden Vorrichtung erteilt ist und die Vorrichtung deren FRAME# Signal aufstellt, das den Beginn einer Transaktion anzeigt, verriegelt der Slave-Block **121** die Transaktions-Informationen (z. B. Adresse, Befehl, Daten, Byte-Freigaben, Parität, usw.) in ein Slave-Verriegelungsregister **156**. Der Warteschlangen-Block **127** ist dann in der Lage, die Transaktions-Informationen von dem verriegelnden Register **156** aufzusuchen und sie zu der DCQ **144** und/oder der Kabel-Schnittstelle **130** zu liefern.

[0115] Transaktionen, unterstützt durch den PCI-Slave-Block **121**, sind in der folgenden Tabelle dargestellt.

Transaktions-Typ	Primäre Schnittstelle	Sekundäre Schnittstelle
Unterbrechungs-Bestätigung	nicht unterstützt	nicht unterstützt
spezieller Zyklus	verzögert	verzögert
I/O-Lesen	verzögert	verzögert
I/O-Schreiben	verzögert	verzögert
Speicher-Lesen	verzögert	verzögert
Speicher-Schreiben	gepostet	gepostet
Konfigurations-Lesen (Typ 0)	sofort	nicht unterstützt
Konfigurations-Schreiben (Typ 0)	sofort	nicht unterstützt
Konfigurations-Lesen (Typ 1)	verzögert	nicht unterstüzt
Konfigurations-Schreiben (Typ 1)	verzögert	nicht unterstützt
Speicher-Lese-Multiple	verzögert (streaming)	verzögert (streaming)
Dual-Adressen-Zyklus	nicht unterstützt	sofort
Speicher-Lese-Zeile	verzögert	verzögert
Speicher-Schreiben und Ungültigkeit	gepostet	gepostet

PCI-Schnittstellen-Slave-Transaktionen

[0116] Der Master-Block **123** der PCI-Schnittstelle **152** lässt nur einen Zyklus laufen, initiiert durch den Warteschlangen-Block **127** (d. h. Transaktionen, gehalten in der PMWQ **140** und der DRQ **142**). Der Warteschlangen-Block **127** fordert von dem PCI-Bus durch Senden eines Anforderungs-Signals (q2p_req) zu dem PCI-Master **123** an, der dann bestimmt, ob ein entsprechendes Anforderungs-Signal (b1req_) zu dem PCI-Arbitrierer **124** aufgestellt ist. Der Master-Block **123** stellt b1req_ auf, falls der Warteschlangen-Block **127** nicht einen verriegelten Zyklus läuft und der PCI-Bus **32** nicht durch eine PCI-Vorrichtung verriegelt ist. Wenn der PCI-Arbitrierer **124** den Warteschlangen-Block **127** auswählt, schickt der Master-Block **123** ein Bestätigungs-Signal (p2q_ack), um den Warteschlangen-Block **127** wissen zu lassen, das er eine Kontrolle des Busses **32** inne hat. Falls der PCI-Arbitrierer **124** keine offenen Anforderungen von anderen Vorrichtungen auf dem Bus **32** vorliegen hat, schickt der Master-Block **123** automatisch das p2q_ack Erteilungs-Signal zu dem Warteschlangen-Block **127**, sogar dann, wenn der Warteschlangen-Block **127** nicht das q2p_req Signal aufgestellt hat. Sobald wie der Warteschlangen-Block **127** eine Arbitrierung erhält (d. h. der Arbitrierer **124** stellt das b1gnt_ Signal auf) und sein q2p_frame Signal aufstellt, um den Beginn einer Transaktion anzuzeigen, verriegelt der PCI-Master **123** Informationen über abgehende Transaktionen (d. h. Adresse, Befehl, Daten, Byte-Freigaben, Parität, usw.) in ein Master-Verriegelungs-Register **158** in der PCI-Schnittstelle **152**. Die Transaktions-Informationen werden dann verwendet, um die Transaktion auf dem PCI-Bus **32** abzuschließen.

[0117] Transaktionen, unterstützt durch den Master-Block **123**, sind in der folgenden Tabelle dargestellt.

Transaktions-Typ	Primäre Schnittstelle	Sekundäre Schnittstelle
Unterbrechungs-Bestätigung	nicht unterstützt	nicht unterstützt
spezieller Zyklus	unterstützt	unterstützt
I/O-Lesen	unterstützt	unterstützt
I/O-Schreiben	unterstützt	unterstützt
Speicher-Lesen	unterstützt	unterstützt
Speicher-Schreiben	unterstützt	unterstützt
Konfigurations-Lesen	nicht unterstützt	unterstützt
Konfigurations-Schreiben	nicht unterstützt	unterstützt
Speicher-Lese-Multiple	unterstützt	unterstützt
Dual-Adressen-Zyklus	unterstützt	nicht unterstützt
Speicher-Lese-Zeile	unterstützt	unterstützt
Speicher-Schreiben und Ungültigkeit	gepostet	gepostet

PCI-Schnittstellen-Master-Transaktionen

[0118] Allgemein arbeitet der Master-Block **123** als ein Standard-PCI-Master. Allerdings wird, im Gegensatz zu Standard-PCI-Brücken, der Master-Block nicht eine MRL, MRM oder MWI Transaktion beenden, bis eine Cache-Zeilen-Grenze erreicht ist, gerade nachdem der Master-Latenz-Zeitgeber (Master Latency Timer – MLT) abläuft. Auch stellt der Master-Block **123** nicht „Initiator Ready" (IRDY) Warte-Zustände auf. Der Master-Block **123** läuft einen verriegelten Zyklus auf dem PCI-Bus **32**, falls der Warteschlangen-Block **127** sein „Verriegelungs" Signal (q2p_lock) aufstellt und seine Verriegelung auf dem Bus **32** freigibt, wenn der Warteschlangen-Block **127** sein „Entriegelungs" Signal (q2p_unlock) aufstellt.

[0119] Wie auch Fig. 57 zeigt, enthält die PCI-Schnittstelle **152** eine Puffer-Entleerungs-Logik **154**, die bestimmt, wann einer oder alle der DCQ Abschluss-Puffer durch den Warteschlangen-Block **127** geleert werden sollten. Die PCI-Slave **121** erzeugt zwei Signale, die durch den Warteschlangen-Block **127** verwendet werden, um die Abschluss-Puffer zu leeren: ein Flush-Signal (p2q_flush), das anzeigt, wann ein Puffer geleert werden sollte, und ein Schlitz- bzw. Steckplatzauswahl-Signal (p2q_slot[2:0]), das anzeigt, welche PCI-Vorrichtung (d. h. welcher Einsteckplatz an dem PCI-Bus) hinsichtlich seiner Daten geleert werden sollte. Die folgende Tabelle stellt die Beziehung zwischen p2q_slot[2:0], und der PCI-Schlitz-Zahl dar.

p2q_slot[2:0]	Einsteckplatz-Zahl
000	alle
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Erzeugung von p2q_slot[2:0]

[0120] Wenn p2q_flush aufgestellt ist, wird der Warteschlangen-Block **127** entweder alle Abschluss-Puffer in der DCQ **144** entleeren, falls p2q_slot[2:0] gleich zu „000" ist, oder den entsprechenden Einen der acht Abschluss-Puffer, falls p2q_slot[2:0] irgendeinen anderen Wert hat. Der Warteschlangen-Block **127** behält ein Protokoll davon bei, welche Abschluss-Puffer, falls welche vorhanden sind, jedem PCI-Steckplatz zu irgendeinem gegebenen Zeitpunkt entsprechen.

[0121] Das p2q_flush Signal wird an der ansteigenden Flanke des ersten PCI-Takt-(CLK)-Zyklus aufgestellt, nachdem ein config Schreib-(WR-CFG)-Zyklus auftritt, oder nachdem ein I/O-Schreib-(iowr)-Zyklus auftritt, oder ein Speicher-Schreib-(memwr)-Zyklus ein ausgangsseitiges Target (hit_tmem) während eines Befehl-Prüf-Zustands (cmd_chk_st) trifft. Gates **2014**, **2016**, **2018** und **2020** und ein Flip-Flop **2022** sind so angeordnet, um p2q_flush auf diese Art und Weise zu erzeugen.

[0122] In dem eingangsseitigen Brücken-Chip (d. h. wenn das Upstream_chip_i Signal aufgestellt ist) besitzt p2q_slot[2:0] immer einen Wert von „001", da die CPU der einzige Master auf dem primären PCI-Bus ist. In dem ausgangsseitigen Chip hängt der Wert von p2q_slot davon ab, ob der Zyklus, der zu einem Flush-Zustand (Flush Condition) führt, ein Zyklus von dem sekundären Bus **32** zu dem Warteschlangen-Block **127** ist (d. h. falls p2q_qcyc aufgestellt ist). Falls das p2q_qcyc Signal aufgestellt ist, nimmt p2q_slot[2:0] den Wert des reg_slot[2:0] Signals an, erzeugt durch die PCI-Slave **121**. Das reg_slot[2:0] Signal zeigt an, welcher der sieben Vorrichtungen auf dem sekundären PCI-Bus **32** eine Kontrolle des Busses **32** erteilt worden ist. Die PCI-Slave **121** erzeugt das reg_slot[2:0] Signal durch Verriegeln des Werts der GNT# Leitung für jeden der sieben Schlitze bzw. Steckplätze auf dem Bus **32**, um ein verriegeltes Erteilungs-Signal mit sieben Bits (latched_gnt_[7:1]; die achte Erteilungs-Leitung, die zu dem Warteschlangen-Block gehört, wird ignoriert) zu bilden, und durch Codieren von latched_gnt[7:1] entsprechend einer Durchsichtstabelle **2006**, die nachfolgend angegeben ist.

latched_gnt[7:1]	reg_slot[2:0]
1111111	000
1111110	001
1111101	010
1111011	011
1110111	100
1101111	101
1011111	110
0111111	111

Erzeugung von req_slot[2:0]

[0123] Falls der Zyklus, der zu dem Entleeren führt, nicht ein Sekundär-PCI-zu-Warteschlangen-Block-Zyklus ist, muss er eine I/O-Lesung oder eine config-Lesung zu dem Sollspeicherbereich von einem der Steckschlitze an dem sekundären Bus **32** sein. Wenn der Zyklus eine I/O-Lesung oder eine config-Lesung ist (d. h. !iowr AND !wr_cfg), nimmt p2q_slot[2:0] den Wert des PCI-Schlitzes an, dessen Speicherbereich getroffen worden ist (mrange_slot[2:0]). Ansonsten ist der Zyklus ein I/O-Schreiben oder ein config-Schreiben und p2q_slot[2:0] wird gleich zu „000", so dass alle Abschluss-Puffer geleert werden. Gates **2008** und **2010** und Multiplexer **2002** und **2004** sind so angeordnet, um p2q_flush[2:0] auf diese Art und Weise zu erzeugen.

KABEL-DECODIERER

[0124] Wie Fig. 58 zeigt, empfängt der Kabel-Decodierer **146** Transaktionen von der Kabel-Schnittstelle und wählt die geeignete Warteschlange so aus, um jede Transaktion zu empfangen. Wenn sich der Kabeldecodierer in der Daten-Phase befindet (d. h. wenn data_phase oder next_data_phase vorliegt, wird ein asynchrones Signal, das den Wert von data_phase einstellt, an dem nächsten CLK-Zyklus aufgestellt ist), sieht der Kabel-Decodierer **146** an dem Befehl-Code (cd_cmd[3:0]), geschickt über das Kabel, nach, um zu bestimmen, welche Warteschlange die Transaktion empfangen sollte. Wie in der Tabelle nachfolgend dargestellt ist, ist, wenn cd_cmd[3:0] einen Wert von „1001" hat, die Transaktion ein verzögerter Abschluss, so dass der Kabel-Decodierer ein cd_dcq_select Signal aufstellt, das der dcq mitteilt, die Transaktion zu beanspruchen. Wenn die drei LSB des Befehl-Code-Signals (cd_cmd[2:0]) „111" sind, ist die Transaktion ein gepostetes Speicher-Schreiben, so dass der Kabel-Decodierer ein cd_pmwq_select Signal erzeugt, um die pmwq über die ankommende Transaktion zu warnen. Wenn die Transaktion weder ein gepostetes Speicher-Schreiben noch ein verzögerter Abschluss ist und der Befehl-Code nicht ein Streaming-Signal darstellt, stellt der Kabel-Decodierer ein cd_drq_select Signal auf, das der DRQ mitteilt, die Transaktion zu beanspruchen. Gates **2024**, **2026**, **2028** und **2030** sind so konfiguriert, um die cd_dcq_select, cd_pmwq_select, und cd_drg_select Signale auf diese Art und Weise zu erzeugen.

[0125] Die nachfolgende Tabelle stellt die Befehl-Code mit vier Bits, zugeordnet zu jedem Typ einer Transaktion, dar.

Transaktions-Typ	Befehl-Code
I/O-Lesen	0010
I/O-Schreiben	0011
Config-Lesen	1010
Config-Schreiben	1011
Speicher-Lesen	0110
MRL	1110
MRM	1100
Speicher-Schreiben	0111
MWI	1111
verzögerter Abschluss	1001
Datenfolge eingereichtet	1000

Transaktions-Befehl-Code

[0126] Wenn der ausgangsseitige Brücken-Chip eine Datenfolge zwischen dem primären Bus und einem sekundären Bus-Master eingerichtet hat, empfängt der eingangsseitige Kabel-Decodierer einen Befehl-Code von „1000". Dieser Code stellt ein Streaming-Signal dar, erzeugt durch den ausgangsseitigen Chip, um den eingangsseitigen Chip zu informieren, dass eine Datenfolge eingerichtet worden ist. Wenn der Kabeldecodierer diesen Befehl-Code empfängt, stellt er ein cd_stream Signal auf, das der QPIF in der eingangsseitigen Vorrichtung mitteilt, die Transaktion fortzuführen. Der Kabel-Decodierer erzeugt auch ein cd_stream_next data Signal, das den eingangsseitigen Chip instruiert, einen anderen Teil von Daten zu dem sekundären Bus zu liefern. Das cd_stream-next_data Signal wird dann aufgestellt, wenn ein cd_stream Signal aufgestellt ist, wobei sich die Transaktion in der Daten-Phase befindet (d. h. data_phase ist aufgestellt), und ein next_data Signal ist von dem eingangsseitigen Chip über die Kabel-Schnittstelle empfangen worden (das next_data Signal erscheint auf einer der Leitungen des c2q_buff[3:0] Signals, das, wenn keine Datenfolge auftritt, dem Warteschlangen-Block mitteilt, welcher ausgangsseitige DCQ-Puffer der momentanen Transaktion zugeordnet ist). Das cd_stream_next_data Signal wird zurückgenommen, wenn entweder das cd_stream Signal zurückgenommen ist oder wenn eine neue Anforderung von der Kabel-Schnittstelle empfangen ist (d. h. c2q_new_req ist aufgestellt). Gates **2032** und **2034** sind so konfiguriert, um die cd_stream und die cd_stream_next_data Signale auf diese Art und Weise zu erzeugen.

GESPEICHERTE-SCHREIB-WARTESCHLANGE

[0127] Wie Fig. 59 zeigt, ist die gepostete Speicher-Schreib-Warteschlange (PMWQ) **140** ein Speicherelement, das alle die Befehl-Informationen enthält, die benötigt werden, um gepostete Schreib-Transaktionen auf dem Target-Bus auszuführen. Die PMWQ umfasst einen Zeichen- bzw. Tag-Speicher-Bereich **2036**, der Informationen hält, die jede Transaktion identifizieren, einen Daten-RAM **2038**, der die Schreib-Daten hält, die jeder Trans aktion in der PMWQ zugeordnet sind, und verschiedene Steuerblöcke, um den Fluß von Transaktionen in die PMWQ hinein und von dieser heraus zu verwalten. Für jede Transaktion in der PMWQ behält der Zeichen-Speicher **2036** Informationen bei, wie beispielsweise über die Adresse, zu der geschrieben werden soll, den PCI-Befehl-Code (MW oder MWI), ein Adressen-Paritäts-Bit, und „einen verriegelten Zyklus" und einen „Dual-Adressen-Zyklus" Indikations-Bits, wie in der folgenden Tabelle dargestellt ist. Der Zeichenspeicher **2036** speichert auch einen Hinweiszeiger zu der Daten-RAM-Stelle der Daten entsprechend zu jeder der Transaktionen in der PMWQ.

Feld	Bits	Anmerkungen
Adresse	64	eingangsseitige Transaktionen unterstützende Dual-Adressen-Zyklen
PCI Befehl	1	Speicher-Schreiben 0111 Speicher-Schreiben und Ungültigkeit 1111 (nur notwendig,um cbe[3] zu speichern)
Byte-Freigaben	0	Speichere BEs an jedem gültigen Übertragungs-Takt in dem Daten-RAM
Parität	1/Adresse 0	muss PAR mit jeder Übertragung zusammen mit 32-Bit addr/Daten speichern. muss Daten-Paritäts-Bits bei jeder gültigen Daten-Übertragung in einem Daten-RAM speichern
Daten	0	gespeichert in Daten-RAM bis zu 8 Cache-Zeilen
Verriegelung	1	
DAC-Indikation	1	zeigt an, ob Adresse 32 oder 64 Bits ist

Inhalte von PMWQ

[0128] Da die PCI Spec. 2.1 fordert, dass gepostete Speicher-Schreib-Transaktionen in der Reihenfolge ausgeführt werden, in der sie empfangen werden, ist der Zeichen- bzw. Tag-Speicher **2036** eine zirkulare FIFO-Vorrichtung. Die PMWQ, und deshalb der Zeichen-Speicher **2036**, können bis zu vier gepostete Speicher-Schreib-Transaktionen simultan handhaben.

[0129] Der Daten-RAM **2038**, umfasst vier Daten-Puffer **2042**, **2044**, **2046** und **2048**, einen für jede Transaktion in der PMWQ. Jeder Puffer kann bis zu acht Cache-Zeilen, oder 256 Bytes, an Daten (acht Worte pro Cache-Zeile) speichern. Für jede Cache-Zeile in einem Puffer speichert der Puffer acht Daten-Paritäts-Bits **2040** (eins pro dword) und zweiunddreißig Freigabe-Bits **2050** (eines pro Byte).

[0130] Ein Kabel-Schnittstellen-Block **2060** empfängt jede Transaktion und die entsprechenden Daten von dem Kabel-Codierer und platziert die Transaktion in dem Zeichen-Speicher **2036**. Ein Warteschlangen-Schnittstellen-Block **2053** empfängt die Daten von dem Kabel-Schnittstellen-Block **2060** und platziert sie in der geeigneten Stelle in dem Daten-RAM **2038**. Die Warteschlangen-Schnittstelle **2053** empfängt auch Daten von dem Daten-RAM **2038** und liefert sie zu der QPIF, wenn die QPIF die entsprechende Transaktion auf dem PCI-Bus laufen lässt. Ein Eingangs-Hinweis-Zeiger Logik-Block **2054** erzeugt vier Eingangs-Hinweis-Zeiger, einen für jeden Puffer, die der Warteschlangen-Schnittstelle **2053** mitteilen, wo das nächste Wort von Daten zu platzieren ist. Ein Gültigkeits(Ausgangs)-Hinweiszeiger-Block **2056** erzeugt vier Ausgangs-Hinweiszeiger, einen für jeden Puffer, die die Position des nächsten Worts, das herangezogen werden soll, anzeigen.

[0131] Wie Fig. 60 zeigt, hält ein Gültigkeits-Flag-Logik-Block **2052** ein Acht-Bit-Gültigkeits-Zeilen-Register **2062** für jeden der vier Puffer in dem Daten-RAM **2038** aufrecht. Das Gültigkeits-Zeilen-Register **2062** zeigt an, welche der acht Cache-Zeilen in jedem Puffer gültige Daten enthält. Wenn das letzte Wort in einer Cache-Zeile mit Daten gefüllt worden ist (d. h. valid_pointer[2:0] entspricht „111" und cd_next-data wird aufgestellt, was anzeigt, dass das Wort gefüllt worden ist), wird das entsprechende Bit in einem Acht-Bit-Kabel-Gültigkeits-Signal (d. h. q0_cable_valid[7:0], q1_cable_valid[7:0], usw.) eingestellt. Das Bit, das eingestellt werden soll, wird durch die drei signifikantesten Bits des Gültigkeits-Hinweiszeigers (valid_pointer[5:3]) bestimmt, die anzeigen, dass die Cache-Zeile gefüllt ist. Das entsprechende Bit in dem Kabel-Gültigkeits-Signal wird auch eingestellt, wenn ein Einsteckplatz-Gültigkeitssignal (validate_slot) von dem Kabel-Decodierer an dem Ende einer Transaktion empfangen ist. Das Kabel-Gültigkeits-Signal wird in das Gültigkeits-Zeilen-Register **2062** entsprechend dem ausgewählten Daten-Puffer an der ansteigenden Flanke des ersten PCI-Takt-Zyklus (CLK) verriegelt, nachdem das letzte Wort gefüllt ist, oder das valid_slot Signal empfangen ist. Ansonsten hält das Gültigkeits-Zeilen-Register seinen momentanen Wert bei. Die Bits in dem Gültigkeits-Zeilen-Register **2062** werden dann gelöscht, wenn die entsprechenden Bits eines Acht-Bit-Ungültigkeits-Signals (d. h. q0_invalid[7:0], q1_invalid[7:0], usw.) aufgestellt sind.

[0132] Der Gültigkeits-Flag-Logik-Block **2052** erzeugt ein pmwq_valid[3:0] Signal, das anzeigt, welcher, falls

irgendeiner vorhanden ist, der vier Daten-Puffer mindestens eine gültige Zeile an Daten enthält. Der Gültigkeits-Block **2052** erzeugt auch ein pmwq_valid_lines[7:0] Signal, das anzeigt, welche der acht Cache-Zeilen eines ausgewählten Daten-Puffers gültig sind. Ein Warteschlangen-Auswahl-Signal von dem QPIF (p2pif_queue_select[1:0]) wird dazu verwendet, auszuwählen, welches gültige Zeilen-Register **2062** eines Daten-Puffers verwendet wird, um das pmwq_valid_lines[7:0] Signal zu erzeugen. Wenn der Warteschlangen-Block eine Steuerung des Busses erhält, um einen geposteten Speicher-Schreib-Zyklus laufen zu lassen, und zwar von einem ausgewählten Daten-Puffer, überträgt der Warteschlangen-Block alle Daten in jeder Zeile, deren entsprechendes Bit in dem pewg_valid_lines[7:0] Signal eingestellt ist. Gates **2064**, **2066**, **2068**, **2070** und **2072** und ein Flip-Flop **2074** sind so angeordnet, um die Werte in dem Gültigkeits-Zeilen-Register **2062** für den ersten Daten-Puffer (q0_valid[7:0]) einzustellen. Eine ähnliche Schaltung bestimmt die Inhalte der Gültigkeits-Register für die anderen drei Daten-Puffer. Ein Multiplexer **2076** wählt den Wert des pmwg_valid_lines[7:0] Signals aus.

[0133] Wie nun Fig. 61 zeigt, hält ein Voll-Zeilen-Logik-Block **2058** ein Acht-Bit-Voll-Zeilen-Register **2078** für jeden der vier Daten-Puffer aufrecht. Die Inhalte jedes Voll-Zeilen-Registers **2078** zeigen an, welche der acht Cache-Zeilen in dem entsprechenden Daten-Puffer voll sind. Die Bits in jedem Voll-Zeilen-Register **2078** werden durch ein asynchrones next_full_line_bit Signal, erzeugt durch die Voll-Zeilen-Zustand-Maschine **2080**, was nachfolgend beschrieben ist, eingestellt. Wenn ein Warteschlangen-Auswahl-Signal von der QPIF (select_nex_queue[3:1)) einen der Daten-Puffer auswählt und das next_full_line_bit Signal aufgestellt ist, wird das Bit in dem Voll-Zeilen-Register **2078** entsprechend zu der Cache-Zeile, angezeigt durch die drei signifikantesten Bits, des Gültigkeits-Hinweiszeigers (valid_pointer[5:3]) eingestellt. Ein 3 × 8 Decodierer **2082** wandelt den Drei-Bit-Gültigkeits-Hinweiszeiger in ein Acht-Bit-Signal um, das bestimmt, welches Bit einzustellen ist. Ein Acht-Bit-Voll-Zeilen-Signal (q0_full_line) wird für jeden Daten-Puffer von den Inhalten des entsprechenden Voll-Zeilen-Registers **2078** erzeugt. Das Voll-Zeilen-Signal zeigt an, welche Zeilen in dem entsprechenden Daten-Puffer voll sind. Der Voll-Zeilen-Logik-Block **2058** erzeugt auch ein pmwg_full_line[7:0] Signal, das anzeigt, welche Cache-Zeilen eines ausgewählten Daten-Puffers voll sind. Der Multiplexer **2084** und das q2pif_queue_select[1:0] Signal werden dazu verwendet, das pmwg_full_line[7:0] Signal zu erzeugen.

[0134] Wie auch Fig. 62 zeigt, wird die Voll-Zeilen-Zustand-Maschine **2080** in einen IDLE Zustand **2086** bei einem Reset platziert. In dem IDLE Zustand **2086** wird das next_full_line_bit auf Null gesetzt. Wenn eine Transaktion in die PMWQ platziert wird, tritt die Transaktion in zwei Phasen auf, eine Adressen-Phase und eine Daten-Phase. Wenn die Daten-Phase beginnt (d. h. ein clock_second_phase Signal wird aufgestellt) und der Gültigkeits-Hinweiszeiger zu dem ersten Wort in der Cache-Zeile hinweist (valid_pointer[2:0] = „000"), geht die Zustand-Maschine **2080** zu einem DATA Zustand **2088** über. In dem Datenzustand wird das next_full_line_bit Signal nur dann aufgestellt, wenn der Gültigkeits-Hinweiszeiger auf das letzte Wort in der Cache-Zeile hinweist (valid_pointer[2:0] = "111"), wird das cd_next_data Signal durch den Kabel-Decodierer aufgestellt (anzeigend, dass das letzte Wort mit Daten gefüllt wurde), und das Byte-Freigabe-Signal von dem Kabel-Decodierer (cd_byte_en[3:0]) gleicht „0000". Die Zustand-Maschine geht auch zurück zu dem IDLE Zustand **2086**, wenn diese Zustände auftreten. Falls diese Zustände nicht auftreten, bevor die Transaktion endet (d. h. cd_complete wird aufgestellt), bleibt das next_full_line_bit Signal nicht aufgestellt und die Zustand-Maschine **2080** geht zurück zu dem IDLE Zustand **2086**. Die Zustand-Maschine **2080** geht auch zu dem IDLE Zustand **2086** ohne Aufstellen des next_full_line_bit Signals über, wenn das cd_byte_en[3:0] Signal einen Wert, einen anderen als „0000", annimmt.

[0135] Wie wiederum Fig. 59 und auch Fig. 63 zeigen, muss die PMWQ normalerweise eine Transaktion von dem Kabel-Decodierer beenden, wenn der Daten-Puffer, der die entsprechenden Daten aufnimmt, voll ist. Allerdings lässt, wenn der Kabel-Decodierer fortfährt, Daten zu verschicken, nachdem der Puffer voll ist, ein Überlauf-Logik-Block **2090** zu, dass die Daten in den nächsten, leeren Puffer überlaufen. Der Überlauf-Logik-Block **2090** führt ein Überlauf-Register **2092**, das anzeigt, welche, falls irgendwelche vorhanden sind, der vier Daten-Puffer als Überlauf-Puffer verwendet werden. Die Inhalte des Überlauf-Registers **2092** werden dazu verwendet, ein Vier-Bit-Überlauf-Signal (pmwq_overflow[3:0]) zu erzeugen. Wenn sich die Transaktion in der Daten-Phase befindet (d. h. data_phase ist aufgestellt), erreicht der Gültigkeits-Hinweiszeiger das letzte Wort eines Daten-Puffers (d. h. valid_pointer[5:0] = „111111"), der Kabel-Decodierer zeigt an, dass mehr Daten ankommen (d. h. cd_next_data wird aufgestellt), und der Kabel-Decodierer hat nicht angezeigt, dass die Transaktion abgeschlossen ist (d. h. cd_complete ist nicht aufgestellt), das select_next_queue[3:0] Signal, das auf den am kürzesten vorher gefüllten Daten-Puffer hinweist, wird dazu verwendet, das Überlauf-Register-Bit entsprechend dem nächsten Daten-Puffer einzustellen. Falls die Bedingungen nicht erfüllt sind, wird das Überlauf-Bit gelöscht. Gates **2094** und **2095** werden in Verbindung mit dem select_next_queue[3:0] Signal verwendet, um die geeigneten Überlauf-Register-Bits einzustellen und zu löschen, wenn diese Zustände erfüllt sind.

[0136] Eine einzelne Transaktion kann fortführen, um in zusätzliche Puffer überzulaufen, bis der letzte, nicht benutzte Puffer voll ist. Falls mehr als ein Puffer als ein Überlauf-Puffer verwendet wird, werden Mehrfach-Überlauf-Register-Bits eingestellt. Aufeinanderfolgende, eingestellte Bits in dem Überlauf-Register zeigen an, dass eine einzelne Transaktion in mehr als einen Puffer übergelaufen ist. Die Überlauf-Bits werden entweder eingestellt oder gelöscht, wenn die gepostete Schreib-Transaktion in die PMWQ hinein platziert ist. Auch kann, falls die QPIF beginnt, die PMW-Transaktion auf dem Target-Bus laufen zu lassen und den originalen Puffer zu entleeren, während sich die Daten noch dabei befinden, in die PMWQ einzutreten, wobei der Original-Puffer wieder verwendet werden, um die Überlauf-Transaktion fortzuführen. Der Überlauf kann fortführen, bis alle verfügbaren Puffer voll sind.

WARTESCHLANGE FÜR VERZÖGERTE ANFORDERUNG

[0137] Wie Fig. 64 zeigt, speichert die DRQ **142** alle die Informationen, die dazu benötigt werden, eine verzögerte Lese-Anforderung- (Delayed Read Request – DRR) und eine verzögerte Schreib-Anforderung- (Delayed Write Request – DRW) Transaktionen auf dem Target-Bus abzuschließen. Die DRQ umfasst einen Warteschlangenspeicher **2100**, der Informationen hält, wie beispielsweise die Adresse, die davon gelesen oder dazu geschrieben werden soll, den PCI-Befehl-Code, Byte-Freigaben, Adressen- und Daten-Paritäts-Bits, Indikations-Bits über einen „verriegelten Zyklus" und einen „Dual-Adressen-Zyklus", und die Puffer-Zahl des Puffers für den verzögerten Abschluss, reserviert in dem initiierenden Brücken-Chip für die Abschluss-Informationen. Der Warteschlangen-Speicher **2100** hält auch bis zu zweiunddreißig Bits (ein Wort) an Daten, die zu dem Target-Bus in einem verzögerten Schreib-Zyklus geschrieben werden sollen. Da verzögerte Schreib-Zyklen niemals mehr als ein Wort an Daten einsetzen, wird kein Daten-RAM in der DRQ benötigt. Die DRQ, und deshalb der Warteschlangen-Speicher **2100**, ist in der Lage, bis zu drei verzögerte Anforderungs-Transaktionen auf einmal zu halten. Ein Kabel-Schnittstellen-Block **2102** beansprucht verzögerte Anforderungs-Transaktionen von dem Kabel-Decodierer und platziert sie in den Warteschlangen-Speicher **2100**. Die folgende Tabelle zeigt die Informationen, die in dem DRQ-Warteschlangen-Speicher beibehalten werden.

Feld	Bits	Anmerkungen
Adresse	64	eingangsseitige Transaktionen unterstützen Dual-Adressen-Zyklen
PCI-Befehl	4	I/O-Lesen I/O-Schreiben Config-Lesen Config-Schreiben Speicher-Lesen Speicher-Lese-Zeile Speicher-Lese-Multiple
Byte-Freigaben	4	Byte-Freigaben, nicht notwendig an MRL, MRM
Parität	1/Adresse 1/Daten-Übertragung	schicke Daten PAR mit verzögerten Schreib-Transaktionen
Daten	32	Daten, in die Warteschlange gestellt, bei verzögerten Schreib-Transaktionen
Verriegelung	1	
DAC-Indikation	1	zeigt an, ob Adresse 32 oder 64 Bits ist
Buff Num	3	zeigt DCQ-Puffer, zugeordnet für Abschluss-Daten, an

Inhalte von DRQ

[0138] Wie auch Fig. 65 zeigt, bestimmt ein Gültigkeits-Flag-Logik-Block **2104**, wann die DRQ alle die Informationen empfangen hat, die dazu notwendig sind, die Transaktionen in dem Warteschlangen-Speicher **2100** laufen zu lassen. Wenn einer der DRQ-Schlitze durch ein entsprechendes Schlitz-Auswahl-Signal ausgewählt ist (d. h. select_zero für den ersten Schlitz, select_one für den zweiten Schlitz und select_two für den dritten Schlitz) und der Schlitz durch ein validate_slot Signal für gültig erklärt ist, anzeigend, dass der Kabel-Decodierer ein Zuführen der Transaktion zu der DRQ abgeschlossen hat, wird ein Gültigkeits-Signal entsprechend zu dem Schlitz (d. h. q0_valid, q1_valid oder q2_valid) an der ansteigenden Flanke des nächsten

PCI-Takt-(CLK)-Zyklus aufgestellt. Falls ein Schlitz nicht ausgewählt ist und durch ein valid_slot Signal für gültig erklärt ist, wird das Gültigkeits-Signal für den Schlitz bzw. den Einsteckplatz wieder zurückgenommen, falls die QPIF den Schlitz ausgewählt hat, durch Aufstellen eines DRQ-Auswahl-Signals (q2pif_drq_select) und durch Identifizieren des Schlitzes bzw. des Einsteckplatzes (q2pif_queue_select = Schlitz-Zahl), allerdings die Transaktion durch Aufstellen eines Zyklus-Aussonderungs-Signals (q2pif_abort_cycle) ausgesondert hat. Das Gültigkeits-Signal wird auch zurückgenommen, falls die DRQ die Transaktion durch Aufstellen eines Zyklus-Abschluss-Signals (z. B. q0_cycle_complete) beendet, während die QPIF auf mehr Daten wartet (d. h. q2pif_next_data ist aufgestellt). Allerdings wird das Zyklus-Abschluss-Signal ignoriert, falls die QPIF momentan Daten zu dem anderen Brücken-Chip (d. h. q2pif_streaming ist aufgestellt) als Datenfolge überträgt. Ansonsten hält, falls das Gültigkeits-Signal des Schlitzes nicht spezifisch aufgestellt oder an einem Takt-Zyklus zurückgenommen ist, er seinen momentanen Wert bei. Der Gültigkeits-Flag-Logik-Block **2104** erzeugt auch ein DRQ Gültigkeits-Signal (drq_valid[3:0]), das anzeigt, welcher, falls irgendeiner vorhanden ist, der drei DRQ-Schlitze eine gültige Transaktion enthält, und zwar durch Kombinieren der Gültigkeits-Signale für jeden individuellen Schlitz (d. h. drq_valid = {0, q2_valid, q1_valid, q0_valid}). Gates **2106**, **2108**, **2110**, **2112** und **2114**, Multiplexer **2116** und **2118** und ein Flip-Flop **2120** sind so angeordnet, um die Schlitz-Gültigkeits-Signale und die DRQ-Gültigkeits-Signale auf diese Art und Weise zu erzeugen.

[0139] Die DRQ umfasst auch Hinweiszeiger-Logik-Blöcke, die Hinweiszeiger zu den Speicher-Stellen beibehalten, von denen Daten während verzögerter Lese-Anforderungs-Transaktionen gelesen werden sollen. Wenn die Adresse, an der die verzögerten Lese-Transaktionen beginnen wird, in den Warteschlangen-Speicher **2100** eingeladen ist, erzeugt ein Gültigkeits-Hinweiszeiger-Logik-Block **2122** einen Sechs-Bit-Gültigkeits-Hinweiszeiger, der anzeigt, wo die Transaktion enden wird. Falls die Transaktion ein einzelnes Wort umfasst (z. B. ein Speicher-Lesen), stellt die Gültigkeits-Hinweiszeiger-Logik **2122** den gültigen Hinweiszeiger gleich zu der Adresse, eingeladen in den Warteschlangen-Speicher **2100** hinein, auf. Für eine Speicher-Lese-Zeilen-Transaktion gibt die Gültigkeits-Hinweiszeiger-Logik **2122** dem gültigen Hinweiszeiger einen Wert von „000111", was anzeigt, dass der letzte, gültige Teil der Daten ein acht dwords (d. h. eine Cache-Zeile) über den Startpunkt hinaus ist. Für eine Speicher-Lese-Mehrfach-Transaktion wird der gültige Hinweiszeiger auf „111111" gesetzt, was anzeigt, dass der letzte, gültige Teil an Daten vierundsechzig dwords (d. h. acht Cache-Zeilen) über den Startpunkt ist. Die Gültigkeits-Hinweiszeiger-Logik **2122** behält einen gültigen Hinweiszeiger für jeden Schlitz in der DRQ (valid_pointer_0[5:0], valid_pointer_1[5:0] und valid_pointer_2[5:0]) bei. Die Stelle des gültigen Hinweiszeigers wird durch die DRQ dann ignoriert, wenn sie ein Streaming-Signal von der QPIF (q2pif_streaming) empfängt, wie dies in weiterem Detail nachfolgend beschrieben ist.

[0140] Ein Ausgangs-Hinweiszeiger-Logik-Block **2124** behält drei Ausgangs-Hinweiszeiger (output_pointer_0[5:0], output_pointer_1[5:0] und output_pointer_2(5:0]), einen für jeden Schlitz in der DRQ, bei, die das nächste Wort an Daten anzeigen, das von dem Speicher gelesen werden soll, und zugeführt zu dem anderen Brücken-Chip. Der Hinweiszeiger wird dann erhöht, wenn die QPIF anzeigt, dass sie bereit ist, den nächsten Teil an Daten zu lesen (d. h. sie stellt das q2pif_next_data Signal auf), einmal für jede Wort-Lesung. Mit Ausnahme in Streaming-Situationen, wird eine Transaktion beendet (abgeschlossen), wenn der Ausgangs-Hinweiszeiger den gültigen Hinweiszeiger erreicht. Wenn eine Transaktion endet, bevor alle Daten gelesen sind (d. h. bevor der Ausgangs-Hinweiszeiger den Eingangs-Hinweiszeiger erreicht), wird die QPIF an der Stelle, angezeigt durch den Ausgangs-Hinweiszeiger, aufnehmen, wenn die Transaktion wieder beginnt. Wenn der Ausgangs-Hinweiszeiger erhöht wird, allerdings die Ausgangs-Hinweiszeiger-Logik **2124** ein Stepback-Signal (q2pif_step_back) empfängt, was anzeigt, dass die Transaktion an dem PCI-Bus beendet wurde, bevor die QPIF in der Lage war, den letzten Teil von Daten zu lesen, erniedrigt die Ausgangs-Hinweiszeiger-Logik **2124** den Zähler einmal, so dass der letzte, nicht gelesene Teil der Daten gelesen werden kann, wenn die Transaktion wieder beginnt. Ein Warteschlangen-Schnittstellen-Block **2126** liefert Transaktions-Informationen und den gültigen und Ausgangs-Hinweiszeiger zu der QPIF.

WARTESCHLANGE FÜR VERZÖGERTEN ABSCHLUSS

[0141] Wie Fig. 66 zeigt, speichert die DCQ **144** verzögerte Abschluss-Nachrichten, die die Antwort des Target-Busses auf jede verzögerte Anforderung, ausgegeben zu dem initiierenden Bus hin, enthalten. Verzögerte Abschluss-Nachrichten entsprechend zu verzögerten Lese-Anforderungen umfassen die angeforderten Daten, während verzögerte Abschluss-Nachrichten, entsprechend zu verzögerten Schreib-Anforderungen, keine Daten umfassen. Ein Kabel-Schnittstellen-Block **2130** beansprucht verzögerte Abschluss-Nachrichten von dem Kabel-Decodierer und liefert die verzögerten Abschluss-Informationen zu einem Tag-Speicher **2132**. Die DCQ, und deshalb der Tag-Speicher **2132**, ist dazu geeignet, bis zu acht verzögerte Abschluss-Nachrichten auf einmal zu speichern. Der Tag-Speicher **2132** speichert Informationen, wie beispielsweise den PCI-Befehl und die Adresse, enthalten in der originalen Anforderung, zu der verzögerten Abschluss-Nachricht führend,

Byte-Freigabe-Bits, Adressen- und Daten-Paritäts-Bits und Bits für einen „verriegelten Zyklus" und einen „Dual-Adressen-Zyklus". Für verzögerte Schreib-Transaktionen, die immer nur ein einzelnes Wort an Daten einsetzen, speichert der Tag-Speicher **2132** eine Kopie der geschriebenen Daten. Jeder der acht Schlitze bzw. Steckplätze in dem Tag-Speicher **2132** umfasst einen implizierten Hinweiszeiger zu einem von acht entsprechenden Daten-Puffern in einem DCQ-Daten-RAM **2134**. Für verzögerte Lese-Transaktionen werden die zurückgeführten Daten in einem entsprechenden Daten-Puffer **2135a–h** in dem Daten-RAM **2134** gespeichert. Die folgende Tabelle stellt die Informationen dar, gespeichert in dem Tag-Speicher **2132** für jede Transaktion, gehalten in dem DCQ.

Feld	Bits	Anmerkungen
Adresse	64	eingangsseitige Transaktionen unterstützen Dual-Adressen-Zyklen
PCI-Befehl	4	I/O-Lesen I/O-Schreiben Config-Lesen Config-Schreiben Speicher-Lesen Speicher-Lese-Zeile Speicher-Lese-Multiple
Byte-Freigaben	4	Byte-Freigaben, nicht notwendig an MRL, MRM
Parität	1/Daten-Übertragung	schicke Daten PAR mit verzögerten Schreib-Transaktionen
Daten	32	Daten, in die Warteschlange gestellt, bei verzögerten Schreib-Transaktionen
Verriegelung	1	
DAC-Indikation	1	zeigt an, ob Adresse 32 oder 64 Bits ist

Inhalte von DCQ

[0142] Jeder der acht Daten-Puffer in dem DCQ-Daten-RAM **2134** kann bis zu acht Cache-Zeilen (256 Bytes) an verzögerten Abschlussdaten speichern. Deshalb sind die Puffer groß genug, um alle Abschlussdaten für sogar die größten, verzögerten Anforderungs-Transaktionen (Speicher-Lese-Mehrfach-Transaktion) zu speichern. Allerdings kann die Kapazität jedes Daten-Puffers auf vier Cache-Zeilen durch Einstellen eines Konfigurations-Bits (cfg2q_eight_line_) in den Konfigurations-Registern des Brücken-Chips reduziert werden. Jeder Daten-Puffer kann durch Daten gefüllt werden, vorgesehen in einer einzelnen, verzögerten Abschluss-Transaktion, oder falls nicht alle angeforderten Daten in einer einzelnen, verzögerten Abschluss-Transaktion zurückgeführt werden, durch mehrfache, verzögerte Abschluss-Transaktionen. Allerdings kann jeder Daten-Puffer Daten entsprechend zu nur einer originalen, verzögerten Anforderung enthalten, ungeachtet davon, wieviele verzögerte Abschluss-Transaktionen es benötigt, die angeforderten Daten zu liefern.

[0143] Ein Warteschlangen-Schnittstellen-Block **2136** steuert den Fluss von Abschluss-Daten von der DCQ-Kabel-Schnittstelle **2130** in den Daten-RAM **2134** hinein und aus dem Daten-RAM **2134** zu der QPIF heraus. Drei Logik-Blöcke erzeugen Hinweiszeiger, die die Eingabe und die Ausgabe von Daten, gespeichert in den acht Daten-Puffern, leiten. Der erste Block, ein Eingangs-Hinweiszeiger-Logik-Block **2138**, behält einen Sechs-Bit-Eingangs-Hinweiszeiger für jeden der acht Daten-Puffer bei (in_pointer_0[5:0], in_pointer_1[5:0], usw.). Jeder Eingangs-Hinweiszeiger weist auf die Stelle in dem entsprechenden Daten-Puffer hin, um das nächste Wort an Daten zu platzieren. Der zweite Block, ein Ausgangs-Hinweiszeiger-Logik-Block **2140**, behält einen Sechs-Bit-Ausgangs-Hinweiszeiger für jeden der acht Puffer bei (out_pointer_0[5:0], out_pointer_1[5:0], usw.). Jeder Ausgangs-Hinweiszeiger weist auf die Stelle des Worts von Daten unmittelbar der Wort, das als letztes von der QPIF entfernt ist, hin. Der Ausgangs-Hinweiszeiger für einen ausgewählten Daten-Puffer wird dann erhöht, wenn die QPIF anzeigt, dass sie für den nächsten Teil von Daten bereit ist (d. h. wenn q2pif_next_data aufgestellt ist). Falls der Ausgangs-Hinweiszeiger erhöht wird, allerdings der letzte Teil von Daten nicht die anfordernde Vorrichtung erreicht, da die Transaktion durch eine Vorrichtung, eine andere als die QPIF, beendet wurde, stellt die QPIF ein Rückschritt-Signal (q2pif_step_back) auf, das bewirkt, dass der Ausgangs-Hinweiszeiger-Logik-Block **2140** den Ausgangs-Hinweiszeiger um ein Wort erniedrigt.

[0144] Der dritte Hinweiszeiger-Block, ein gültiger Hinweiszeiger-Logik-Block **2142**, behält für jeden der Acht-Daten-Puffer einen Sechs-Bit-Gültigkeits-Hinweiszeiger bei (valid_pointer_0[0:5], valid_pointer_1[5:0],

usw.), der das nächste Wort an Daten in dem entsprechenden Daten-Puffer anzeigt, der zu der QPIF verfügbar ist. Da die PCI Spec. 2.1 erfordert, dass Lese-Abschluss-Daten nicht vor einer früher initiierten, geposteten Speicher-Schreib-Transaktion zurückgeführt werden, können verzögerte Abschluss-Daten, platziert in die DCQ hinein, während ein gepostetes Speicher-Schreiben in dem PMWQ anhängig ist, nicht zu der anfordernden Vorrichtung verfügbar gemacht werden, bis das gepostete Speicher-Schreiben auf dem PCI-Bus abgeschlossen ist und von der PMWQ entfernt ist. Deshalb muss, solange wie irgendwelche früher in die Warteschlange gestellten, geposteten Speicher-Schreib-Transaktionen in der PMWQ verbleiben, der gültige Hinweiszeiger bei seiner momentanen Position verbleiben. Dann kann, wenn alle früher in die Warteschlange gestellten, geposteten Speicher-Schreib-Vorgänge von der PMWQ entfernt worden sind, der gültige Hinweiszeiger zu derselben Position wie diejenige in dem Hinweiszeiger bewegt werden. Wenn die PMWQ leer ist, sind alle verzögerten Abschluss-Daten gültig (d. h. zu der anfordernden Vorrichtung hin verfügbar), sobald wie sie in der DCQ gespeichert sind.

[0145] Wie auch die Fig. 67A und Fig. 67B zeigen, muss der Gültigkeits-Hinweiszeiger-Logik-Block **2142** den Master-Zyklus-Arbitrierer (Master Cycle Arbiter – MCA) fragen, um alle verzögerten Abschluss-Transaktionen für gültig zu erklären, die in die verzögerte Abschluss-Warteschlange eintreten, während ein gepostetes Speicher-Schreiben in der PMWQ anhängig ist. Allerdings kann, da der MCA nicht mehr als vier verzögerte Abschluss-Transaktionen auf einmal in die Warteschlange stellen kann, wie dies nachfolgend diskutiert ist, der Gültigkeits-Hinweiszeiger-Logik-Block **2142** eine Gültigkeit von nicht mehr als vier verzögerten Abschluss-Daten-Puffern auf einmal anfordern. Der Gültigkeits-Hinweiszeiger-Logik-Block **2142** muss auch eine Protokollierung beibehalten, welche vier verzögerten Abschluss-Transaktionen in dem MCA zu irgendeinem gegebenen Zeitpunkt in die Warteschlange gestellt sind. Um dies so vorzunehmen, behält der Gültigkeits-Hinweiszeiger-Logik-Block **2142** zwei Vier-Schlitz-Register bei: ein DCQ-Puffer-Zahl-Register **2144** und ein Gültigkeits-Anforderungs-Register **2146**. Das Puffer-Zahl-Register **2144** behält die Drei-Bit-DCQ-Puffer-Zahl bei, wie dies durch das DCQ-Puffer-Zahl-Signal (cd_dcq_buff_num[2:0]) bestimmt ist, geliefert durch den Kabel-Decodierer, und zwar von jeder verzögerten Abschluss-Transaktion, in dem MCA in die Warteschlange gestellt. Das Gültigkeits-Anforderungs-Register **2146** behält ein Transaktions-Gültigkeits-Anforderungs-Bit für jeden der DCQ-Puffer bei, deren Zahlen in den vier Schlitzen bzw. Einsteckplätzen **2148a–d** des Puffer-Zahl-Registers **2144** gespeichert sind. Das Anforderungs-Bit in jedem Schlitz **2150a–d** des Gültigkeits-Anforderungs-Registers **2146** wird aufgestellt, falls eine entsprechende, verzögerte Abschluss-Transaktion in dem MCA in die Warteschlange gestellt ist. Die Werte der Bits in den vier Gültigkeits-Anforderungs-Schlitzen **2150a–d** werden zusammen mit dem MCA als ein Vier-Bit-Gültigkeits-Anforderungs-Signal (dcq_valid[3:0]) vorgesehen.

[0146] Wenn eine verzögerte Abschluss-Transaktion in dem MCA in die Warteschlange gestellt werden soll, wird seine entsprechende DCQ-Puffer-Zahl in einen der Puffer-Zahl-Schlitze bzw. -Einsteckplätze **2148a–d** durch das cd_dcq_buff_num[2:0] Signal eingeladen. Der Schlitz **2148a–d**, der beladen werden soll, wird durch ein Zwei-Bit-Auswahl-Signal (next_valid_select[1:0]) ausgewählt. Der Wert des Auswahl-Signals hängt von dem Wert des dcq_valid[3:0] Signals, erzeugt durch das Gültigkeits-Anforderungs-Register **2146** und die Durchsichts-Tabelle **2152**, ab, wobei die Inhalte davon in der Tabelle nachfolgend gezeigt sind. Der Schlitz wird dann beladen bzw. belegt, wenn er durch next_valid_select[1:0] ausgewählt ist, wenn der Kabel-Decodierer die DCQ ausgewählt hat und die Transaktion abgeschlossen hat (d. h. cd_dcq_select und cd_complete werden aufgestellt), und wenn mindestens eine gepostete Speicher-Schreib-Transaktion in der PMWQ anhängig ist (d. h. pmwq_no_pmw ist nicht aufgestellt). Gates **2154**, **2156**, **2158**, **2160** und **2162** und ein 2 × 4 Decodierer **2164** sind so angeordnet, um das Puffer-Zahl-Register **2144** auf diese Art und Weise zu laden. In ähnlicher Weise wird das entsprechende Bit in dem Gültigkeits-Anforderungs-Register **2146** durch den Ausgang von Gates **2154**, **2156**, **2158**, **2160** und **2162** und den 2 × 4 Decodierer **2164** eingestellt.

dcq_valid[3:0]	next_valid_select[1:0]	Schlitz #
xxx0	00	0
xx01	01	1
x011	10	2
0111	11	3

Puffer-Zahl-Register-Schlitz-Auswahl

[0147] Auf das dcq_valid[3:0] Signal hin gibt der MCA ein Vier-Bit-DCQ-Lauf-Signal (mca_run_dcq[3:0]) aus,

das anzeigt, welcher der DCQ-Puffer, auf den durch das Puffer-Zahl-Register hingewiesen ist, seinen Gültigkeits-Hinweiszeiger aktualisiert haben kann. Das mca_run_dcq[3:0] Signal wird zu einem Gültigkeits-Hinweiszeiger-Aktualisierungs-Logik-Block **2166** geliefert, und zwar zusammen mit dem pmwg_no_pmw Signal und den In-Hinweiszeigern für jeden der acht Daten-Puffer. Falls eine gepostete Speicher-Schreib-Transaktion in der PMWQ verbleibt, nachdem der MCA eines der mca_run_dcq[3:0] Bits aufstellt (was dann auftreten wird, wenn eine gepostete Speicher-Schreib-Transaktion in die Warteschlange gestellt wurde, nachdem die verzögerte Abschluss-Transaktion in die Warteschlange gestellt wurde, allerdings bevor der MCA das entsprechende mca_run_dcq Bit aufgestellt hat), wird der entsprechende Gültigkeits-Hinweiszeiger aktualisiert, solange wie keine anderen, verzögerten Abschluss-Transaktionen entsprechend zu demselben DCQ-Puffer noch in dem MCA in die Warteschlange gestellt sind. Falls eine verzögerte Abschluss-Transaktion für denselben DCQ-Puffer noch in der MCA in die Warteschlange gestellt ist, kann der Gültigkeits-Hinweiszeiger nur aktualisiert werden, wenn das mca_run_dcq Bit entsprechend dieser Transaktion aufgestellt ist. Andererseits werden, sobald das pmwq_no_pmw Signal weggenommen ist, alle Gültigkeits-Hinweiszeiger aktualisiert, um die entsprechenden In-Hinweiszeiger anzupassen, ungeachtet davon, ob verzögerte Abschlüsse noch in der Warteschlange in dem MCA gestellt sind. Wenn ein mca_run_dcq Bit aufgestellt ist, wird das entsprechende Bit in dem Gültigkeits-Anforderungs-Register **2146** gelöscht. Gates **2168**, **2170**, **2172**, **2174** und **2176** sind so angeordnet, um die Gültigkeits-Anforderungs-Register-Bits auf diese Art und Weise zu löschen. Wie wiederum **Fig. 66** zeigt, bestimmt ein Hit-Logik-Block **2180**, wenn eine verzögerte Anforderungs-Transaktion von einer anfordernden Vorrichtung auf dem PCI-Bus eine der verzögerten Abschluss-Nachrichten in der DCQ „getroffen" hat. Entsprechend zu der PCI Spec 2.1 müssen die folgenden Attribute identisch für einen verzögerten Abschluss sein, um zu einer Anforderung angepasst zu werden: Adresse, PCI Befehl, Byte-Freigaben, Adressen- und Daten-Parität (falls eine Schreib-Anforderung vorliegt), REQ64# (falls eine 64-Bit-Daten-Transaktion vorliegt), und LOCK# (falls unterstützt wird). Falls eine Anforderung durch die PCI-Slave verriegelt ist, sucht die QPIF die Anforderungs-Informationen auf, schickt sie zu der DCQ und stellt ein Prüf-Zyklus-Signal auf (q2pif_check_cyc), das die DCQ-Hit-Logik **2180** instruiert, die Anforderungs-Informationen zu den verzögerten Abschluss-Nachrichten, gespeichert in dem DCQ-Tag-Speicher **2132**, zu vergleichen. Die Hit-Logik **2180** empfängt das 64 Bit-Adressen-Signal (q2pif_addr[63:2]), das Vier-Bit-PCI-Befehls-Signal (q2pif_cmd[3:0]), die vier Freigabe-Bits (q2pif_byte_en[3:0]), das Dual-Adressen-Zyklus-Bit (q2pif_dac), (das dem PCI REQ64# Signal entspricht), das Verriegelungs-Bit (q2pif_lock) von der QPIF, und, falls die Anforderung eine Schreib-Anforderung ist, die Daten, die geschrieben werden sollen (q2pif_data[31:0]). Obwohl es nicht durch die PCI Spec 2.1 erforderlich ist, liefert die QPIF auch die Schlitz-Zahl (q2pif_slot[2:0]) der anfordernden Vorrichtung, um das Puffer-Entleerungs-Programm für den Warteschlangen-Block fortzuführen, wie dies nachfolgend beschrieben ist. Die Hit-Logik **2180** vergleicht dann jedes dieser Signale mit verzögerten Abschluss-Informationen, gespeichert in den acht DCQ-Puffern. Falls alle Signale die Informationen irgendwelcher der verzögerten Abschluss-Nachrichten anpassen, identifiziert die Hit-Logik **2180** den Puffer, der die Anpassungs-Abschluss-Nachricht enthält, durch Aufstellen eines entsprechenden Bits in einem Acht-Bit-Hit-Signal (dcq_hit[7:0]). Wenn ein Treffer bzw. Hit auftritt, sucht die QPIF die Abschluss-Nachricht auf und liefert sie zu der anfordernden Vorrichtung. und, falls die Anforderung eine Lese-Anforderung ist, beginnt sie ein Entfernen der zurückgeführten Daten von dem entsprechenden Daten-Puffer in dem Daten-RAM **2134**. Falls die angeforderten Informationen nicht die Abschluss-Informationen irgendeiner der verzögerten Abschluss-Nachrichten in der DCQ anpassen, ist die Anforderung in Bezug auf die DCQ „fehlgeschlagen" („missed"), und wird in dem nächsten, verfügbaren DCQ-Puffer gespeichert und über das Kabel zu dem anderen Brücken-Chip durch die QPIF weitergeführt. Eine PCI-Vorrichtung, die eine Anforderung initiiert, die die DCQ verfehlt hat, kann deren REQ# Leitung maskiert haben, bis deren Abschluss-Nachricht zurückgeführt ist, wie dies in weiterem Detail nachfolgend beschrieben ist.

[0148] Die Hit-Logik **2180** verbindet sich auch schnittstellenmäßig mit einem Multi-Threaded-Master-Erfassungs-Block **2182**, um zu erfassen, welche PCI-Schlitze bzw. – Steckplätze, falls welche vorhanden sind, Multi-Threaded Vorrichtungen enthalten. Multi-Threaded Vorrichtungen sind in der Lage, mehr als eine, verzögerte Transaktion zu einem Zeitpunkt beizubehalten, und müssen deshalb speziell behandelt werden. Wenn ein Multi-Threaded-Master erfasst ist, wird ein entsprechendes Bit in den Konfigurations-Registern eingestellt, um anzuzeigen, dass die Vorrichtung in der Lage ist, mehrere, offenstehende, verzögerte Transaktionen zu unterstützen, und deshalb sollte deren REQ# Zeile nicht maskiert werden. Eine Multi-Threaded-Master-Erfassung wird in weiterem Detail nachfolgend diskutiert.

[0149] Eine andere Funktion der DCQ ist diejenige, zu bestimmen, wenn eine Gelegenheit existiert, um eine Datenfolge von Lese-Daten zwischen dem primären und dem sekundären PCI-Bus erzeugen. Eine Streaming-Gelegenheit existiert dann, wenn verzögerte Abschluss-Daten in die DCQ durch den Kabel-Decodierer platziert werden, während sie noch auf dem Target-Bus durch die Target-Vorrichtung platziert sind. Falls die PCI-Vorrichtung, die die Transaktion initiierte, wieder deren Anforderung liefert, während die Target-Vorrich-

tung noch Daten auf dem PCI-Bus platziert, wird eine Lese-Datenfolge eingerichtet. Da ein Lese-Streaming eine effiziente Art und Weise ist, um Daten zwischen dem primären und dem sekundären PCI-Bus zu übertragen, gibt der PCI-Brücken-Chip nicht nur eine höhere Priorität in den Bus-Arbitrierungs-Prozess zu einer Vorrichtung, deren Abschluss-Daten ankommen, sondern wird auch versuchen, eine Nicht-Streaming-Transaktion zu beenden, um die Möglichkeit zu verbessern, dass eine Datenfolge eingerichtet werden wird. Allerdings ist es, während in der Theorie ein Streaming während irgendeines Lese-Zyklus auftreten kann, in der Praxis wahrscheinlich, dass dies nur während Transaktionen auftritt, die eine große Menge an Daten umfassen (d. h. Speicher-Lese-Mehrfach-Transaktionen). Deshalb wird der Warteschlangen-Block versuchen, Transaktionen zugunsten von potentiellen Streaming-Gelegenheiten nur dann zu beenden, wenn eine potentielle Streaming-Transaktion eine Speicher-Lese-Mehrfach-Transaktion ist.

[0150] Wie auch Fig. 68 zeigt, bestimmt ein Stream- bzw. Datenfolge-Logik-Block **2184** in der DCQ, ob eine Streaming-Gelegenheit existiert, und, falls dies der Fall ist, erzeugt er die Signale, erforderlich dazu, die Datenfolge zu unterstützen. Der Datenfolge-Logik-Block **2184** erzeugt die Signale, erforderlich dazu, eine momentane Transaktion zugunsten einer potentiellen Datenfolge zu unterbrechen. Wenn der Kabel-Decodierer eine verzögerte Abschluss-Transaktion in der DCQ platziert, verwendet die Datenfolge-Logik **2184** das DCQ-Puffer-Zahl-Signal, geliefert durch den Kabel-Decodierer (cd_dcq_buff_num), um den PCI-Befehl-Code, gespeichert in dem entsprechenden DCQ-Puffer (q0_cmd[3:0], q1_cmd[3:1] usw.), aufzusuchen. Falls der Befehl-Code eine Speicher-Lese-Mehrfach-Anforderung (d. h. „1100") darstellt, stellt die Datenfolge-Logik **2184** ein Unterbrechungs-Für-Datenfolge-Signal (dcq_disconnect_for_stream) auf, das die QPIF und die PCI-Schnittstelle instruiert, die momentane Transaktion aufgrund einer potentiellen Streaming-Gelegenheit zu beenden. Ein Multiplexer **2186** und ein Komparator **2188** sind so angeordnet, um das dcq_disconnect_for_stream Signal zu erzeugen. Dann liefert, solange wie der Kabel-Decodierer fortfährt, die Abschluss-Daten zu der DCQ zu liefern (d. h. das cd_dcq_select Signal verbleibt aufgestellt) und keine gepostеten Speicher-Schreibvorgänge in der PMWQ erscheinen (d. h. pmwq_no_pmw verbleibt aufgestellt), liefert die Datenfolge-Logik **2184** ein Streaming-Request-Signal (q2a_stream) direkt zu dem PCI-Arbitrierer. Die Datenfolge-Logik **2184** liefert auch die Schlitz- bzw. Einsteckplatz-Zahl der potentiellen Streaming-Vorrichtung (q2a_stream_master[2:0]) zu dem PCI-Arbitrierer unter Verwendung des cd_dcq_buff_num[2:0] Signals, um die PCI-Schlitz-Zahl, gespeichert in dem ausgewählten DCQ-Puffer (q0_master[2:0] für DCQ-Puffer-Null **2135a**, q1_master[2:0] für DCQ-Puffer-Eins **2135b**, usw.), auszuwählen. Der PCI-Arbitrierer hebt dann die Bus-Arbitrierungs-Priorität der potentiellen Streaming-Vorrichtung an, wie dies in weiterem Detail nachfolgend diskutiert ist. Falls dem potentiellen Streaming-Master nicht der Bus erteilt wird, bevor die Streaming-Gelegenheit verschwindet, wird deren Priorität zu Normal zurückgeführt. Da der eingangsseitige Bus nur eine Master-Vorrichtung (die CPU) besitzt, wird dieses Merkmal in dem eingangsseitigen Chip gesperrt. Das Gate **2190** und der Multiplexer **2192** sind so angeordnet, um die q2a_stream und q2a_stream_master Signale zu erzeugen.

[0151] Wenn eine anfordernde Vorrichtung eine verzögerte Abschluss-Nachricht, gespeichert in der DCQ, trifft, wird das entsprechende Bit eines Acht-Bit-Hit-Signals (hit[7:0]) aufgestellt. Das hit[7:0] Signal zeigt an, welcher der acht DCQ-Puffer durch die momentane Anforderung getroffen ist. Wenn dies auftritt, verriegelt, falls der entsprechende DCQ-Puffer Daten enthält (d. h. dcq_no_data wird nicht aufgestellt), die Datenfolge-Logik **2118** den Wert des Hit-Signals für die Dauer der Transaktion (d. h. solange, wie q2pif_cyc_complete aufgestellt ist). Die verriegelte Version des Hit-Signals bildet ein „verzögertes" Hit-Signal (dly_hit [7:0]). Wenn entweder das Hit-Signal oder das verzögerte Hit-Signal anzeigt, dass ein DCQ-Puffer getroffen worden ist, liefert ein Drei-Bit-DCQ-Datenfolge-Puffer-Signal (dcq_stream_buff[2:0]) die Puffer-Zahl des getroffenen bzw. Hit-DCQ-Puffers. Dann stellt, falls der Kabel-Decodierer verzögerte Abschluss-Daten in den Puffer platziert, während sich der Zyklus, der den Puffer traf, in Arbeit befindet (d. h. cd_dcq_select wird aufgestellt und cd_dcq_buff_num[2:0] entspricht dcq_stream_buff[2:0]), der Datenfolge-Logik-Block **2180** ein Datenfolge-Verbindungs-Signal auf (dcq_stream_connect), das der QPIF mitteilt, dass eine Datenfolge eingerichtet worden ist. Die QPIF informiert dann den Brücken-Chip auf dem Target-Bus, dass eine Datenfolge eingerichtet worden ist. Falls bestimmte Bedingungen erfüllt sind, wird die Target-QPIF fortfahren im Datenfluss zu arbeiten, bis sie mitteilt, zu stoppen, durch Initiieren von QPIF, wie dies in weiterem Detail nachfolgend diskutiert ist. Gates **2194** und **2196**, Multiplexer **2198** und **2200** und ein Flip-Flop **2202** sind so angeordnet, um das verzögerte Hit-Signal zu erzeugen. Gates **2204**, **2206** und **2208** und ein Codierer **2210** sind so angeordnet, wie dies dargestellt ist, um die dcq_stream_connect und dcq_stream_buff[2:0] Signale zu erzeugen.

[0152] Wie wiederum Fig. 66 zeigt, wird die DCQ, unter bestimmten Umständen, automatisch Daten von dem Target-Bus zu Lasten eines PCI-Masters bei der Antizipierung automatisch vorab Abrufen, dass der Master zurückkommen wird und die Daten anfordern wird. Ein Prefetch-Logik-Block **2212** in der DCQ stellt Daten vorab ein, wenn der lesende Master alle die Daten in seinem DCQ-Puffer verbraucht und die Prefetch-Logik **2212**

erwartet, dass die anfordernde Vorrichtung mit einer sequenziellen Lese-Anforderung zurückkehren wird (d. h. eine Anforderung, die Daten aufnimmt, angeordnet an der nächsten, sequenziellen Stelle in dem Speicher). Da einige Vorrichtungen, wie beispielsweise Multi-Threaded-Master, routinemäßig alle die Daten lesen, angefordert in einer Transaktion, und dann mit einer unterschiedlichen, nicht-sequenziellen Anforderung zurückkehren, umfasst die Prefetch-Logik **2212** eine Vorhersageschaltung, die die Prefetch-Fähigkeiten für jede Vorrichtung auf dem PCI-Bus sperrt, bis die Vorrichtung eine Tendenz dahingehend gezeigt hat, sequenzielle Lese-Anforderungen auszugeben. Sobald sie eine Vorrichtung, die vorab eingestellte Daten empfangen hat, mit einer nicht-sequenziellen Lese-Anforderung zurückkehrt, wird die Voraussage-Schaltung die Prefetching-Funktion für diesen Master sperren.

[0153] Unter Bezugnahme auch auf die Fig. 69a und Fig. 69b umfasst der Prefetch-Logik-Block **2212** ein Prefetch-Vorhersage-Register **2214**, wobei der Ausgang davon ein Acht-Bit-Prefetch-Freigabe-Signal (prefetch_set[7:0]) ist, das beurteilt, ob die Prefetch-Funktion für jede der Vorrichtungen auf dem PCI-Bus verfügbar ist. Alle Bits in dem Prefetch-Freigabe-Signal werden bei einem Reset (RST) gelöscht und wenn die QPIF ein allgemeines Löschen aller der DCQ-Register fordert (d. h. general_flush wird aufgestellt und q2pif_slot[2:0] entspricht „000"). Das general_flush Signal wird in weiterem Detail nachfolgend diskutiert. Gates **2216** und **2218** erzeugen das Signal, das die prefetch_set Bits wiedereinstellt.

[0154] Ein individuelles Bit in dem Prefetch-Freigabe-Signal wird dann eingestellt, wenn der entsprechende PCI-Schlitz bzw. PCI-Einsteckplatz durch das q2pif_slot Signal ausgewählt ist und die folgenden Zustände auftreten: die anfordernde Vorrichtung trifft einen verzögerten Abschluss-Puffer in der DCQ (d. h. eines der Bits in den cycle_hit[7:0] Signal wird aufgestellt), die momentane Transaktion ist eine Speicher-Lese-Zeile oder ein Speicher-Lese-Mehrfach-Zyklus (d. h. q2pif_cmd[3:0) entspricht „1100" oder „11110"), die QPIF hat angezeigt, dass der Zyklus vollständig ist, (d. h. q2pif_cyc_complete wird aufgestellt) und das letzte Wort von Daten wurde von dem DCQ-Puffer herangezogen (d. h. last_word ist aufgestellt). Gates **2220**, **2222**, **2224** und **2228a**–**h** und ein Decodierer **2226** sind so angeordnet, um die Vorhersage-Bits auf diese Art und Weise einzustellen. Das last_word Signal wird durch die Prefetch-Logik **2212** aufgestellt, wenn die anfordernde Vorrichtung versucht, hinter dem Ende des DCQ-Puffers zu lesen. Dies tritt dann auf, wenn der Out-Pointer (Out-Hinweiszeiger) und der In-Pointer (In-Hinweizeiger) gleich sind, was anzeigt, dass das Ende des DCQ-Puffers erreicht worden ist (d. h. für einen Vier-Cache-Zeilen-Puffer, out_pointer_x[4:0] entspricht valid_pointer_x[4:0] oder, für einen Acht-Cache-Zeilen-Puffer, out_ointer_x[5:0] entspricht valid_pointer_x[5:0]), und wenn die anfordernde Vorrichtung versucht, einen anderen Teil von Daten zu lesen (d. h. q2pif_next_data ist aufgestellt). Gates **2230**, **2232** und **2234** sind so angeordnet, um das last word Signal zu erzeugen.

[0155] Ein individuelles Bit in dem Prefetch-Freigabe-Signal wird dann gelöscht, wenn der entsprechende PCI-Schlitz ausgewählt ist und entweder ein PCI-Lösch-Zustand auftritt (p2q_flush wird aufgestellt), die QPIF mitteilt der DCQ, den Gültigkeits-Hinweiszeiger des Puffers zurückzusetzen (q2p_step_back ist aufgestellt), oder die anfordernde Vorrichtung eine Transaktion initiiert, die alle der DCQ-Puffer verfehlt (q2pif_check_cyc wird aufgestellt und dcq_hit wird nicht aufgestellt). Gates **2236**, **2238** und **2240a**–**h** und ein Decodierer **2226** sind so angeordnet, um die Vorhersage-Freigabe-Bits in dieser Art und Weise zu löschen.

[0156] Wenn die Prefetch-Funktion für eine Vorrichtung auf dem PCI-Bus freigegeben ist, kann die Prefetch-Logik **2212** zwei Typen von Prefetch-Signalen für die Vorrichtung erzeugen: ein Prefetch-Zeilen-Signal (dcq_prefetch_line) und ein Prefetch-Mehrfach-Signal (dcq_prefetch_mul). Das Prefetch-Leitungs-Signal wird dann erzeugt, wenn der momentane PCI-Befehl von der anfordernden Vorrichtung ein Speicher-Lese-Leitungs-Zeilen-Signal ist, und das Prefetch-Mehrfach-Signal wird dann erzeugt, wenn der momentane PCI-Befehl ein Speicher-Lese-Mehrfach-Signal ist. In jedem Fall wird das entsprechende Prefetch-Signal erzeugt, wenn die folgenden Zustände auftreten: das prefetch_set Bit für den anfordernden PCI-Schlitz wird eingestellt; ein entsprechendes Prefetch-Freigabe-Bit in den Konfigurations-Registern wird eingestellt (cfg2q_auto_refetch_enable); die DRQ in dem eingangsseitigen Chip ist nicht voll (!tc_dc_full); der DCQ-Puffer hat Raum für die entsprechende Menge an Prefetch-Daten (_!dcq_no_prefetch_room); der momentane Zyklus trifft den DCQ-Puffer; und der anfordernde Master hat versucht, hinter das Ende des DCQ-Puffers zu lesen (last_word und q2pif_cyc_complete). Gates **2242**, **2244**, **2246**, **2248**, **2250**, und **2252**, ein Decodierer **2254** und Multiplexer **2256** und **2258** sind so angeordnet, um die Prefetch-Signale auf diese Art und Weise zu erzeugen.

[0157] Wenn die Prefetch-Logik **2212** ein Prefetch-Signal erzeugt, erzeugt sie ein entsprechendes Prefetch-Adressen-Signal (dcq_prefetch_addr[63:2]) durch Verknüpfen der oberen siebenundfünfzig Bits der Adresse, gespeichert in dem entsprechenden DCQ-Puffer (q0_addr[63:7] für Puffer Null, q1_addr[63:7] für Puffer Eins, usw.) mit den unteren fünf Bits des Ausgangs-Hinweiszeigers des Puffers (out_pointer_0[4:0], usw.). Ein Dual-Adressen-Zyklus Signal (dcq_prefetch_dac) zeigt an, ob die Prefetch-Transaktion ein Dual- oder Ein-

zel-Adressen-Zyklus ist. Das dcq_prefetch_cycle Signal nimmt den Wert des Dual-Adressen-Bits, gespeichert in dem DCQ-Puffer (q0_dac, q1_dac, usw.), an. Für sowohl die Prefetch-Adressen- und Dual-Adressen-Zyklus-Signale wird der geeignete Wert von einem Multiplexer **2260** oder **2262** ausgegeben und durch das Drei-Bit-DCQ-Puffer-Zahl-Signal ausgewählt, anzeigend, welcher DCQ-Puffer durch die momentane Anforderung getroffen wurde.

[0158] Wiederum unter Bezugnahme auf Fig. 66, besitzt jeder DCQ-Daten-Puffer verschiedene mögliche Zustände, wobei jeder davon durch einen Puffer-Zustand-Logik-Block **2264** in der DCQ bestimmt wird. Das Folgende sind die möglichen Puffer-Zustände.

1. Leer (Empty). Verfügbar für eine Zuordnung. Ein Puffer ist leer (empty) nach einem Laden bzw. Hochfahren und nachdem er geleert ist.
2. Complete. Der Puffer enthält Abschluss-Informationen für einen verzögerten Abschluss von einer realen, verzögerten Anforderung von einer Vorrichtung auf dem PCI-Bus (d. h. keine Prefetch-Anforderung). Die PCI-Vorrichtung hat noch nicht wieder verbunden und Daten von dem Puffer genommen. Die verzögerte Abschluss-Transaktion ist abgeschlossen.
3. Prefetch. Der Puffer enthält Abschluss-Daten für eine Prefetch-Anforderung oder angeforderte Daten, die in dem Puffer belassen wurden, nachdem der anfordernde Master von dem Puffer getrennt ist. Alle Abschluss-Daten sind von dem Target angekommen.
4. PartComplete. Der Puffer ist für Abschluss-Informationen für eine reale, verzögerte Anforderung reserviert und kann diese enthalten (d. h. keine Prefetch-Anforderung). Der Master hat noch nicht wieder verbunden und Daten von dem Puffer genommen, und nicht alle Abschluss-Informationen sind von dem Target angekommen.
5. PartPrefetch. Der Puffer ist für Abschluss-Informationen für eine Prefetch-Anforderung reserviert oder enthält sie, oder der Puffer enthält angeforderte Daten, die in dem Puffer verblieben, nachdem der anfordernde Master von dem Puffer getrennt ist. Nicht alle Abschluss-Informationen sind von dem Target angekommen.
6. Discard. Der Puffer wurde gelöscht, während er in dem PartPrefetch-Zustand war, allerdings sind die letzten Abschluss-Daten bis jetzt noch nicht von dem Target angekommen. Der Puffer wird in den Discard-Zustand versetzt, um zu verhindern, dass er verwendet wird, bis die Transaktion auf dem Target-Bus abgeschlossen ist und die letzten Daten ankommen.

[0159] Wenn die QPIF einen DCQ-Puffer für eine verzögerte Anforderungs-Transaktion anfordert, ordnet die Puffer-Zustand-Logik **2264** die Puffer in der folgenden Reihenfolge zu. Falls kein Puffer in dem Empty-Zustand oder dem Prefetch-Zustand vorliegt, muss der anfordernde Master erneut versucht werden.

Puffer-Zahl	Puffer-Zustand
Q0	Empty
Q1	Empty
Q2	Empty
Q3	Empty
Q4	Empty
Q5	Empty
Q6	Empty
Q7	Empty
Q0	Prefetch
Q1	Prefetch
Q2	Prefetch
Q3	Prefetch
Q4	Prefetch
Q5	Prefetch
Q6	Prefetch
Q7	Prefetch

DCQ-Puffer-Zuordnung

[0160] Wenn eine Vorrichtung auf dem PCI-Bus eine verzögerte Lese-Anforderung initiiert und ein DCQ-Abschluss-Puffer daneben eingestellt ist, ändert die Puffer-Zustand-Logik **2264** den Zustand des Puffers zu Part-Complete. Falls die DCQ eine Prefetch-Lesung initiiert, wird der Puffer-Zustand zu PartPrefetch geändert. Wenn der letzte Teil von Abschluss-Daten ankommt, wird der Zustand des Puffers von PartComplete oder Part-Prefetch zu Complete oder Prefetch jeweils hin geändert. Wenn die anfordernde Vorrichtung wieder eine neu versuchte Lese-Anforderung zuführt und den Puffer trifft, werden irgendwelche gültigen Abschluss-Daten zu dem Master hin gegeben, falls sich der Puffer in dem Complete, Prefetch, PartComplete oder PartPrefetch-Zustand befindet.

[0161] Wenn der Master nicht alle die Daten vor einer Unterbrechung nimmt, wird der Zustand des Puffers zu Prefetch oder PartPrefetch hin geändert, um anzuzeigen, dass nicht beanspruchte Daten dahingehend angesehen werden, dass sie Prefetch-Daten sind. Falls der Master den letzten Teil von Daten heranzieht, wenn sich der Puffer in dem Complete oder Prefetch-Zustand befindet, wird der Zustand des Puffers zu Empty hin geändert.

[0162] Falls ein Lösch-Signal empfangen wird, während sich ein Puffer in dem Prefetch-Zustand befindet, werden die Prefetch-Daten in dem Puffer ausgesondert und der Puffer-Zustand wird zu Empty hin geändert. Falls ein Lösch-Ereignis auftritt, während sich der Puffer in dem PartPrefetch-Zustand befindet und Abschluss-Daten noch ankommen, wird der Puffer zu dem Discard-Zustand geändert, bis alle Prefetch-Daten ankommen. Wenn die Transaktion abgeschlossen ist, werden die Prefetch-Daten ausgesondert und der Puffer-Zustand wird zu Empty hin geändert. Falls sich der Puffer in dem Complete oder Part-Complete-Zustand befindet, wenn ein Lösch-Signal empfangen wird, werden die Abschluss-Daten in dem Puffer belassen und der Puffer-Zustand verbleibt ungeändert. Falls das Lösch-Signal auftritt, da die entsprechende PCI-Vorrichtung eine neue Anforderung ausgegeben hat, d. h. eine Anforderung, die nicht momentan in die Warteschlange gestellt ist, und die alle die Abschluss-Puffer „verfehlt"), wie dies nachfolgend diskutiert worden ist, ordnet die DCQ einen neuen Puffer für die Transaktion zu, wie dies vorstehend diskutiert ist. Deshalb kann eine PCI-Vorrichtung mehr als einen Abschluss-Puffer zugeordnet haben. Mehrfach-Puffer können zu einer PCI-Vorrich-

tung zugeordnet werden, wenn die Vorrichtung einen Puffer besitzt, der Abschluss-Daten enthält oder erwartet (d. h. der Puffer befindet sich in dem Complete oder PartComplete-Zustand) und die Vorrichtung eine neue Anforderung ausgibt. Da Multi-Threaded-Vorrichtungen die einzigen Vorrichtungen sind, die mehrere Transaktionen auf einmal beibehalten können, können nur Multi-Threaded-Vorrichtungen Mehrfach-Abschluss-Puffer gleichzeitig reserviert haben.

MASTER-ZYKLUS-ARBITRIERER

[0163] Der Master-Zyklus-Arbitrierer (Master Cycle Arbiter – MCA) bestimmt die Ausführungs-Reihenfolge von geposteten Speicher-Schreib- und verzögerten Anforderungs-Transaktionen, während die Reihenfolgen-Einschränkungen zwischen geposteten Speicher-Schreib-, verzögerten Anforderungs- und verzögerten Abschluss-Zyklen beibehalten werden, angegeben in PCI Spec. 2.1. Entsprechend der PCI Spec 2.1 muss der MCA garantieren, dass ausgeführte Zyklen eine starke Schreib-Reihenfolge beibehalten und dass keine Deadlocks auftreten. Um sicherzustellen, dass keine Deadlocks auftreten werden, muss geposteten Speicher-Schreib-Zyklen ermöglicht werden, früher in die Warteschlage gestellte, verzögerte Anforderungs-Zyklen zu passieren, und die geforderten Reihenfolgen-Einschränkungen beizubehalten, wobei verzögerten Anforderungs-Zyklen und verzögerten Abschluss-Zyklen niemals erlaubt werden muss, früher in die Warteschlange gestellte, gepostete Speicher-Schreib-Zyklen zu passieren.

[0164] Unter Bezugnahme wiederum auf Fig. 70 verwendet der MCA zwei Transaktions-Warteschlangen, eine Transaktions-Lauf-Warteschlange (Transaction Run Queue – TRQ) oder (Transaktions-Ausführungs-Warteschlangen) **2270** und eine Transaktions-Reihenfolge-Warteschlange (Transaction Order Queue – TOQ) **2272**, um Zyklen, in die PMWQ, die DRQ und die DCQ warteschlangenmäßig gestellt, zu verwalten. Ein MCA-Steuerblock **2274** nimmt Transaktionen von der PMWQ, DRQ und DCQ in der Form von Vier-Bit-Gültigkeits-Anforderungs-Signalen (pmwq_valid[3:0], drq_valid[3:0] und dcq_valid[3:0]) auf und gibt Lauf-Befehle in der Form von Vier-Bit-Lauf-Signalen (mca_run_pmwq[3:0], mca_run_drq[3:0] und mca_run_dcq[3:0]) aus. Die Transaktionen werden in die TRQ **2270** und die TOQ **2272** durch einen TRQ-Steuerblock **2276** und einen TOQ-Steuerblock **2278** jeweils hinein- und herausbewegt.

[0165] Unter Bezugnahme auch auf Fig. 71 ist die TRQ **2270** die Warteschlange, von der der MCA die Transaktions-Ausführungs-Reihenfolge bestimmt. Transaktionen in der TRQ **2270** können in irgendeiner Reihenfolge ausgeführt werden, ohne die Transaktions-Reihenfolge-Regeln zu verletzten, allerdings kann, wenn einmal ein geposteter Speicher Schreib-Zyklus in der TRQ **2270** platziert ist, kein anderer Zyklus in die TRQ **2270** platziert werden, bis das gepostete Speicher-Schreiben entfernt ist. Transaktionen in der TRQ **270** werden in zirkularer Reihenfolge versucht und sind allgemein in der Reihenfolge, in der sie empfangen wurden, abgeschlossen. Allerdings kann, falls eine Transaktion in der TRQ **2270** auf dem PCI-Bus versucht wird, der MCA die nächste Transaktion in der TRQ **2270** auswählen, die auf dem PCI-Bus versucht werden soll. Da verzögerte Abschluss-Transaktionen Slave-Zyklen, im Gegensatz zu Master-Zyklen, sind, werden sie niemals in der TRQ **2270** platziert. Weiterhin werden, da verzögerte Abschluss-Informationen zu der anfordernden Vorrichtung verfügbar gemacht werden können, sobald sie in die DCQ eintreten, falls keine geposteten Speicher-Schreib-Zyklen in der PMWQ anhängig sind, verzögerte Abschluss-Transaktionen in der TOQ **2272** nur dann platziert, wenn ein geposteter Speicher-Schreib-Zyklus in der TRQ **2270** anhängig ist, wie dies in weiterem Detail nachfolgend diskutiert ist.

[0166] Die TRQ **2270** ist eine zirkulare Warteschlange, die bis zu vier Transaktionen auf einmal hält. Da die MCA immer in der Lage sein muss, mindestens eine gepostete Speicher-Schreib-Transaktion laufen zu lassen, um die erforderlichen Reihenfolgen-Beschränkungen zu wahren, kann die TRQ **2270** niemals mehr als drei verzögerte Anforderungs-Transaktionen auf einmal halten. Weiterhin kann die TRQ nur eine gepostete Schreib-Transaktion zu einem Zeitpunkt halten, da gepostete Schreibvorgänge nicht durch irgendeine später initiierte Transaktion, umfassend andere gepostete Schreibvorgänge, hindurchgeführt werden können. Jeder Schlitz **2280a–d** in der TRQ **2270** enthält drei Bits an Informationen: ein Ein-Bit-Zyklus-Typ-Indikator **2282** („1" gleich für gepostete Speicher-Schreib-Transaktionen und „0" gleich für verzögerte Anforderungs-Transaktionen), und einen Zwei-Bit-Gültigkeits-Hinweiszeiger **2284**, wobei die vier möglichen Werte davon identifizieren, welche der Puffer in der PMWQ oder der DRQ die in die Warteschlangen gestellten Transaktionen belegen. Die TRQ **2270** umfasst auch einen Eingabe/Ausgabe-Freigabe-Block **2286**, der bestimmt, wann eine Transaktion in die TRQ **2270** hinein oder aus dieser heraus bewegt werden kann, einen Eingangs-Logik-Block **2288**, der die Platzierung einer Transaktion in die TRQ **2270** hinein steuert, und einen Ausgangs-Logik-Block **2290**, der eine Entfernung einer Transaktion von der TRQ **2270** steuert. Diese Logik-Blöcke enthalten eine standardmäßige Warteschlangen-Management-Schaltung.

[0167] Ein zirkularer Eingangs-Hinweiszeiger **2292** wählt den nächsten, verfügbaren Schlitz bzw. Einsteckplatz zur Platzierung einer ankommenden Transaktion aus. Der Eingangs-Hinweiszeiger ist zirkular, um so umfangreich wie möglich eine historische Reihenfolge der ankommenden Transaktionen beizubehalten.

[0168] Ein zirkularer Ausgangs-Hinweiszeiger **2294** arbitriert zwischen den Transaktionen in der TRQ **2270** und bestimmt deren Reihenfolge einer Ausführung. Der Ausgangs-Hinweiszeiger **2294** beginnt immer mit dem oberen Schlitz **2286a** in der TRQ **2270** beim Startup und schreitet zirkular durch die TRQ **2270** hindurch. Der Ausgangs-Hinweiszeiger **2294** kann konfiguriert sein, um in entweder einem infiniten Retry- oder einem Null-Retry-Mode zu arbeiten, durch Einstellen oder Löschen, jeweils, eines infiniten Retry-Bits in den Konfigurations-Registern (cfg2q_infretry). In einem infiniten Retry-Mode verbleibt der Ausgangs-Hinweiszeiger **2294** auf einer Transaktion, bis die Transaktion erfolgreich auf dem PCI-Bus läuft. In einem Null-Retry-Mode wird der Ausgangs-Hinweiszeiger **2294** zu jedem Zeitpunkt erhöht, zu dem eine Transaktion auf dem Bus versucht wird (d. h. q2pif_cyc_complete war dem vorherigen PCI-Takt-Zyklus aufgestellt), ungeachtet davon, ob die Transaktion erfolgreich abschließt oder erneut versucht wird. Da die PCI Spec 2.1 vorschreibt, dass gepostete Speicher-Schreib-Transaktionen dahingehend zugelassen werden, im Bypass an verzögerten Anforderungs-Transaktionen vorbeizuführen, muss der Ausgangs-Hinweiszeiger **2294** in mindestens einem der Brücken-Chips so konfiguriert werden, um in einem Null-Retry-Mode zu arbeiten. Hierbei ist der ausgangsseitige Chip immer so konfiguriert, um in einem Null-Retry-Mode zu arbeiten. Alternativ kann der Ausgangs-Hinweiszeiger so konfiguriert sein, um in einem finiten Retry-Mode zu arbeiten, indem jede Transaktion auf dem PCI-Bus eine vorbestimmte Anzahl (z. B. drei) von Malen versucht werden kann, bevor sich der Ausgangs-Hinweiszeiger erhöht. Sowohl der eingangsseitige als auch der ausgangsseitige Chip können so konfiguriert sein, um in einem finiten Retry-Mode zu arbeiten, mit einem Verletzen der Reihenfolgen-Beschränkungen der PCI Spec 2.1. In jedem Fall versucht der Ausgangs-Hinweiszeiger, die historische Reihenfolge von Transaktionen, gespeichert in der TRQ **2270**, beizubehalten, was sich nur dann erhöht, wenn eine Transaktion nicht erfolgreich auf dem Target-PCI-Bus abgeschlossen werden kann.

[0169] Wenn ein gepostеter Speicher-Schreib- oder verzögerter Anforderungs-Zyklus aus der TOQ **2272** ausgesondert ist (new_toq_cycle wird aufgestellt), wie dies nachfolgend diskutiert ist, oder wenn die TOQ **2272** nicht freigegeben ist (!toq_enabled) und ein neuer Zyklus durch die MCA empfangen wird (new_valid_set), werden das Zyklus-Typ-Bit und die Gültigkeits-Bits für den neuen Zyklus in den nächsten, leeren Schlitz in die TRQ eingeladen. Falls der Zyklus von der TOQ **2272** kommt, werden die gültigen Bits und das Zyklus-Typ-Bit durch die TOQ als gültig geliefert und Zyklus-Typ-Signale (toq_valid[1:0] und toq_cyctype[0]) jeweils. Ansonsten werden die neuen Zyklus-Informationen durch die MCA als gültige und Zyklus-Typ-Signale (d_valido[1:0] und d_cyctype[0]) geliefert. Gates **2296** und **2298** und Multiplexer **2300** und **2302** sind angeordnet, um die Auswahl von Zyklen zu steuern, die in die TRQ **2270** hineingeladen werden sollen. Wenn ein Zyklus erfolgreich auf dem PCI-Bus läuft, wird der Zyklus von der Transaktions-Reihenfolge-Warteschlange entfernt und sein Zyklus-Typ-Bit und Gültigkeits-Bits werden zu dem MCA-Steuer-Block **2274** als TRQ-Zyklus-Typ und Gültigkeits-Signale (trq_cyctype[0]) und trq_valido[1:0]) jeweils geliefert.

[0170] Der TRQ-Steuer-Block **2276** erzeugt ein trq_pmw Signal, das anzeigt, wenn eine gepostete Speicher-Schreib-Transaktion in die TRQ **2270** warteschlangenmäßig gestellt wird. Wenn dieses Signal aufgestellt ist, müssen darauffolgend ausgegebene, verzögerte Anforderungs- und verzögerte Abschluss-Transaktionen in die TOQ **2272** in die Warteschlange gestellt werden, wie dies nachfolgend diskutiert ist. Das trq_pmw Signal wird dann aufgestellt, wenn der MCA-Steuer-Block **2274** die TRQ **2270** instruiert hat, einen neuen, gepostеten Speicher-Schreib-Zyklus (trq_slot_valid_set entspricht nicht „0000" und d_trq_cyctype entspricht „1") in die Warteschlange zu stellen, oder, alternativ, wenn irgendeiner der TRQ-Schlitze **2280a–d** einen Zyklus (trq_slot_valid[3:0] entspricht nicht „0000"), ist mindestens einer der Zyklen ein gepostеter Speicher-Schreib-Zyklus (trq_cyctype entspricht „1") und der gepostete Speicher-Schreib-Zyklus ist nicht von dem entsprechenden Schlitz **2280a–d** gelöscht worden (!trq_slot_valid_rst[3:0]). Gates **2304**, **2306**, **2308**, **2310**, und **2312** sind so angeordnet, um das trq_pmw Signal in dieser Art und Weise zu erzeugen.

[0171] Wie nun die Fig. 72 zeigt, ist die TOQ **2272** eine First-In-First-Out (FIFO) Warteschlange, die die historische Reihenfolge von Transaktionen enthält, empfangen durch die Brücke, nachdem eine gepostete Speicher-Schreib-Transaktion in der TRQ **2270** platziert ist. Da alle Transaktionen auf früher ausgegebene, gepostete Speicher-Schreibvorgänge warten müssen, damit diese laufen, werden alle Transaktionen, umfassend gepostete Speicher-Schreib-, verzögerte Anforderungs- und verzögerte Abschluss-Transaktionen, in der TOQ **2270** platziert, wenn ein gepostetes Speicher-Schreiben in die TRQ **2270** warteschlangenmäßig gestellt ist. Transaktionen in der TOQ **2272** müssen in der TOQ **2272** verbleiben, bis die gepostete Speicher-Schreib-Transaktion von der TRQ **2270** entfernt ist.

[0172] Die TOQ **2270**, die acht Schlitze **2314a**–**h** hat, kann bis zu drei gepostete Speicher-Schreib-Transaktionen (die vierte wird in der TRQ **2270** gespeichert werden), drei verzögerte Anforderungs-Transaktionen und vier verzögerte Abschluss-Transaktionen halten. Jeder der Schlitze **2314a**–**h** in der TOQ **2272** enthält zwei Zyklus-Typ-Bits **2316**, die die entsprechende Transaktion („01" ist ein gepostetes Speicher-Schreiben, „00" ist eine verzögerte Anforderung, und „1x" ist ein verzögerter Abschluss) identifizieren, und zwei Gültigkeits-Bits **2318**, die identifizieren, welcher der Puffer in der PMWQ, DRQ und DCQ die entsprechende Transaktion belegt. Die TOQ **2272** umfasst auch Standard-Eingabe- und Ausgabe-Logik-Blöcke **2320** und **2322**, die die Bewegung von Transaktionen in die TOQ **2272** hinein und aus dieser heraus steuern.

[0173] Die Positionen, an denen Transaktionen in die TOQ **2272** hinein platziert und von dieser entfernt werden, werden durch einen Drei-Bit-Eingangs-Zähler **2326** (inputr[2:0]) und einen Drei-Bit-Ausgangszähler **2324** (outputr[2:0]) jeweils bestimmt. Beide Zähler beginnen an dem ersten Schlitz **2314a** in der TOQ **2272** und erhöhen sich durch die Warteschlange hindurch, wenn Transaktionen in die Warteschlange eingegeben und von dieser entfernt werden. Der Eingangs-Zähler **2326** erhöht sich an der ansteigenden Flanke jedes PCI-Takt-Zyklus, wo die TOQ **2272** freigegeben wird (toq_enabled wird aufgestellt), und der MCA-Steuer-Block **2274** liefert einen neuen Zyklus zu der TOQ **2272** (new_valid_set wird aufgestellt). Die Gültigkeits-Bits und die Zyklus-Typ-Bits für jeden neuen Zyklus werden durch den MCA als gültig und die Zyklus-Typ-Signale (d_valido[1:0] und d_cyctype[1:0]) geliefert. Der Ausgangs-Zähler **2324** erhöht sich an der ansteigenden Flanke jedes PCI-Takt-Zyklus, an dem der MCA-Steuerblock **2274** die TOQ **2272** instruiert, zu dem nächsten Zyklus (next_toq_cycle wird aufgestellt) zu gehen, und die TOQ **2272** ist nicht leer: d. h. inputr [2:0] entspricht nicht outputr[2:0]). Zyklen, die in der TOQ **2272** existieren, werden durch die TOQ-Gültigkeits- und cycletype Signale (toq_valido[1:0] und toq_cyctypeo[1:0]) dargestellt. Gates **2328** und **2330** und ein Komparator **2332** sind so angeordnet, um geeignet den Eingangs-Hinweiszeiger **2326** und den Ausgangs-Hinweiszeiger **2324** zu takten.

[0174] Wenn eine verzögerte Anforderungs-Transaktion oder eine gepostete Speicher-Schreib-Transaktion aus der TOQ **2272** ausgesondert ist, wird die Transaktion in die TRQ **2270** platziert, um auf eine Arbitrierung zu warten. Da allerdings verzögerte Abschluss-Transaktionen Target-Transaktionen sind und keine Master-Transaktionen, werden verzögerte Abschlüsse nicht in die TRQ **2270** platziert. Anstelle davon werden verzögerte Abschlüsse einfach aus der TRQ **2272** ausgesondert und dazu verwendet, die entsprechenden Daten in den DCQ-Daten-Puffern für gültig zu erklären. Allerdings müssen, solange wie eine gepostete Speicher-Schreib-Transaktion in der TRQ **2270** warteschlangenmäßig hineingestellt ist, alle verzögerten Abschlüsse in die TOQ **2272** platziert werden, sogar dann, wenn zwei oder mehr verzögerte Abschlüsse derselben, verzögerten Anforderung entsprechen, und deshalb denselben, verzögerten Abschluss-Puffer, wie dies vorstehend beschrieben ist.

[0175] Wie die Fig. 73A bis 73D zeigen, steuert der MCA-Steuerblock **2274** den Fluss von Transaktionen über den MCA. Wie vorstehend diskutiert ist, wird die PMWQ, DRQ und DCQ Anforderungs-Gültigkeit von Transaktionen in den Warteschlangen durch Vorsehen von Vier-Bit-Gültigkeits-Signalen pmwq_valid[3:0], drq_valid[3:0] und dcq_valid[3:0] jeweils, zu dem MCA, gehalten. Unter diesen Signalen kann sich nur ein Bit während jedes Takt-Impulses ändern, da nur eine einzelne, neue Transaktion in den Warteschlangen-Block bei jedem Taktimpuls platziert werden kann. Deshalb identifiziert der MCA-Steuer-Block neue Gültigkeits-Anforderungen durch Überwachen der sich ändernden Bits in den pmwq_valid, drq_valid und dcq_valid Signalen. Um dies zu vorzunehmen, verriegelt der MCA-Steuer-Block jedes Signal und invertiert es an der ansteigenden Flanke jedes PCI-Takts, um ein verzögertes, invertiertes Signal zu erzeugen, und vergleicht das verzögerte, invertierte Signal mit dem momentanen Signal (d. h. dem Signal an dem nächsten Taktimpuls). Da nur ein neu geändertes Bit denselben Wert wie sein verzögertes und invertiertes Gegenstück haben wird, ist der MCA-Steuer-Block in der Lage, zu erfassen, welches Bit geändert ist. Unter Verwendung von Flip-Flops **2340**, **2342** und **2344** und Gates **2346**, **2348** und **2350**, erzeugt die MCA-Steuereinheit new_pmwq_valid[3:0], new_drq_valid[3:0] und new_dcq_valid[3:0] Signale, die, bei jedem Taktimpuls, zusammen identifizieren, ob die DMWQ, DRQ oder DCQ, falls irgendeine vorhanden ist, irgendeine neue Transaktion für eine Validitierung lieferte und welcher Puffer in der entsprechenden Warteschlange die neue Transaktion enthält. Wie auch Fig. 74 zeigt, verwendet der MCA-Steuer-Block eine Durchsichts-Tabelle **2352**, um die zwölf Bits der new_pmwq_valid, new_drq_valid und new_dcq_valid Signale in die Zwei-Bit d_valid[1:0] und d_cyctype[1:0] Signale, geliefert zu der TRQ und der TOQ, wie dies vorstehend diskutiert ist, umzuwandeln.

[0176] Die MCA-Steuereinheit gibt die TOQ durch Verriegeln des toq_enabled Signals auf einen Wert von „1" frei, wenn entweder das trq_pmw aufgestellt ist, was anzeigt, dass ein gepostet Speicher-Schreib-Zyklus in der TRQ in die Warteschlange gestellt ist, oder wenn das toq_enable Signal bereits aufgestellt ist und die TOQ nicht leer ist (!toq_empty). Gates **2354** und **2356** und ein Flip-Flop **2358** sind so angeordnet, um toq_enabled

auf diese Art und Weise zu erzeugen.

[0177] Der MCA-Steuerblock stellt das new_toq_cycle Signal auf, das die TRQ instruiert, den Zyklus in die Warteschlange zu stellen, der von der TOP ausgesondert ist, wenn dort nicht ein gepostеter Speicher-Schreib-Zyklus in TRQ während des vorherigen Takt-Zyklus (!s1_trq_pmw) vorhanden war, wenn die TOQ nicht leer ist (!toq_empty), und wenn der Zyklus, der von der TOQ ausgesondert werden soll, nicht eine verzögerte Abschluss-Transaktion ist (!(toq_cyctypeo[1] = „DC")). Die MCA-Steuereinheit verwendet ein Gate **2360**, um das new_toq_cycle Signal zu erzeugen.

[0178] Das next_toq_cycle Signal, das verwendet wird, um den TOQ-Ausgangs-Zähler auf den nächsten Zyklus in der TOQ zu erhöhen, wird aufgestellt, wenn die TOQ nicht leer ist (!toq_empty) und entweder wenn keine gepostеten Speicher-Schreib-Zyklen momentan in der TRQ in die Warteschlange gestellt sind (!trq_pmw) und der nächste Zyklus in der TOQ ein verzögerter Abschluss ist (toq_cyctype[1] = „DC"), oder wenn der nächste TOQ-Zyklus ein gepostetes Speicher-Schreiben oder eine verzögerte Anforderungs-Transaktion ist (!(toq_cyctype[1] = „DC")) und dabei keine gepostеten Speicher-Schreib-Transaktionen während des vorherigen Taktzyklus vorhanden waren (!s1_trq_pmw). Der Steuerblock verwendet Gates **2362**, **2364**, **2366** und **2368**, um das next_toq_cycle Signal zu erzeugen.

[0179] Die MCA-Steuereinheit erzeugt das mca_run_dcq[3:0] Signal, um anzuzeigen, dass eine verzögerte Abschluss-Transaktion von der TOQ ausgesondert worden ist. Wenn die TRQ keine gepostеten Speicher-Schreib-Zyklen enthält (!trq_pmw), ist die TOQ nicht leer (!toq_empty), und der TOQ-Zyklus ist ein verzögerter Abschluss (toq_cyctype[1] = „DC"), das mca_run_dcq[3:0] Signal nimmt den Wert des decodierten toq_valido[1:0] Signals an, was vorstehend diskutiert ist. Ansonsten ist das mca_run_dcq[3:0] Signal gleich „0000". Das Gate **2370**, der Decodierer **2372** und der Multiplexer **2374** sind so angeordnet, um mca_run_dcq[3:0] auf diese Art und Weise zu erzeugen.

[0180] Der MCA-Steuer-Block erzeugt new_mca_run_dr[3:0] und new_mca_run_pmw[3:0] Signale, um anzuzeigen, dass er eine neue, verzögerte Anforderungs-Transaktion und eine gepostete Speicher-Transaktion jeweils hat, die in die Warteschlange gestellt werden sollen. Das new_mca_run_dr[3:0] Signal nimmt den Wert von dem 2 × 4 decodierten d_valido[1:0] Signal an, was vorstehend diskutiert ist, wenn der neue Zyklus ein verzögerter Anforderungs-Zyklus ist (d_cyctype[0] = „DR"). Ansonsten werden alle Bits des new_mca_run_dr[3:0] Signals auf Null gesetzt. Ähnlich nimmt das new_mca_run_pmw[3:0] Signal den Wert des 2 × 4 decodierten d_valido[1:0] Signals an, wenn der neue Zyklus eine gepostete Speicher-Schreib-Transaktion ist, und wird ansonsten auf „0000" eingestellt. Decodierer **2376** und **2380** und Multiplexer **2378** und **2382** sind so angeordnet, um die new_mca_run_dr und die new_mca_run_pmw Signale auf diese Art und Weise zu erzeugen.

[0181] Die MCA-Steuereinheit erzeugt toq_mca_run_dr[3:0] und toq_mca_run_pmw[3:0] Signale, um anzuzeigen, wenn eine neue, verzögerte Anforderungs-Transaktion oder eine gepostete Speicher-Schreib-Transaktion, jeweils, von der TOQ ausgesondert wurde. Das toq_mca_run_dr[3:0] Signal nimmt den Wert des 2 × 4 decodierten toq_valido[1:0] Signals an, wenn ein verzögerter Anforderungs-Zyklus von der TOQ ausgesondert ist, und einen Wert von „0000" ansonsten. In ähnlicher Weise nimmt das toq_mca_run_pmw[3:0] Signal den Wert des 2 × 4 decodierten toq_valido[1:0] Signals an, wenn ein gepostеter Speicher-Schreib-Zyklus von der TOQ ausgesondert ist, und einen Wert von „0000" ansonsten. Decodierer **2384** und **2388** und Multiplexer **2386** und **2390** werden verwendet, um die toq_mca_run_dr und toq_mca_run_pmw Signale auf diese Art und Weise zu erzeugen.

[0182] Die MCA-Steuereinheit erzeugt die trq_mca_run_dr[3:0] und trq_mca_run_pmw[3:0] Signale, um anzuzeigen, wenn eine neue, verzögerte Anforderungs-Transaktion oder eine gepostete Speicher-Schreib-Transaktion jeweils die Arbitrierung in der TRQ erlangt hat, und bereit ist, auf dem PCI-Bus zu laufen. Das trq_mca_run_dr[3:0] Signal nimmt den Wert des 2 × 4 decodierten trq_valido[1:0] Signals an, wenn ein verzögerter Anforderungs-Zyklus die Arbitrierung erlangt hat und die TRQ nicht leer ist. Das trq_mca_run_dr[3:0] nimmt einen Wert von „0000" ansonsten an. In ähnlicher Weise nimmt das trq_mca_run_pmw[3:0] Signal den Wert des 2 × 4 decodierten trq_valido[1:0] Signals an, wenn ein gepostеter Speicher-Schreib-Zyklus die Arbitrierung erlangt hat und die TRQ nicht leer ist. Das trq_mca_run_pmw[3:0) Signal wird auf einen Wert von „0000" ansonsten eingestellt. Die Gates **2392** und **2398**, die Decodierer **2394** und **2400** und die Multiplexer **2396** und **2402** werden dazu verwendet, die trq_mca_run_dr und trq_mca_run_pmw Signale in dieser Art und Weise zu erzeugen.

[0183] Wenn die TRQ leer ist, kann der MCA eine Anforderung ausgeben, um die nächste Transaktion in der

TOQ laufen zu lassen, während die Transaktion in die TRQ platziert wird. Wenn sowohl die TRQ als auch die TOQ leer sind, können Transaktionen damit beginnen, zu laufen, sogar bevor sie in die TRQ warteschlangenmäßig gestellt sind. Deshalb umfasst der MCA-Steuer-Block eine Logik, die bestimmt, wenn die new_mca_run oder die toq_mca_run Signale asynchron verwendet werden können, um anzuzeigen, dass eine Transaktion auf dem PCI-Bus versucht werden kann. Durch Umwandeln der new_mca_run oder der der toq_mca_run Signale in asynchrone Laufsignale, sichert die MCA-Steuereinheit einen PCI-Takt-Warte-Zustand. Wenn das new_valid_set Signal durch den MCA-Steuerblock aufgestellt ist und die TOQ nicht freigegeben ist (!toq_enabled), nehmen die async_mca_run_dr[3:0] und async_mca_run_pmw[3:0] Signale die Werte der new_mca_run_dr[3:0] und new_mca_run_pmw[3:0] Signale jeweils an. Ansonsten nehmen die asynchronen Laufsignale die Werte von toq_mca_run_dr[3:0] und toq_mca_run_pmw[3:0] Signale an. Die MCA-Steuereinheit verwendet das Gate **2404** und die Multiplexer **2406** und **2408** dazu, die asynchronen Laufsignale zu erzeugen.

[0184] Wenn ein PCI-Bus-Master eine Transaktion abgeschlossen hat (s1_q2pif_cyc_complete ist aufgestellt), ist die TRQ nicht leer (!trq_empty) und ist für einen Betrieb in dem Null-Retry-Mode konfiguriert (!cfg2q_infretry), und irgendeine neue Transaktion ist von der TOQ ausgesondert worden (new_toq_cycle) oder die TOQ ist nicht freigegeben (!toq_enabled) und die MCA hat einen neuen Zyklus empfangen, der für gültig erklärt wird (new_valid_set), die MCA kann nicht einen Zyklus auswählen, um auf dem PCI-Bus zu laufen, so dass sowohl das mca_run_dr[3:0] als auch das mca_run_pmw[3:0] Signal auf „0000" gesetzt werden. Ansonsten nehmen, falls die TRQ leer ist (trq_empty) und entweder eine neue Transaktion von der TOQ ausgesondert ist (new_toq_cycle) oder die TOQ nicht freigegeben ist (!toq_enabled) und die MCA einen neuen Zyklus empfangen hat, der für gültig erklärt wird (new_valid_set), dann die mca_run_dr[3:0] und mca_run_pmw[3:0] Signale den Wert der asynchronen Lauf-Signale async_mca_run_dr[3:0] und async_mca_run_pmw[3:0], jeweils, an. Ansonsten nimmt das mca_run_dr[3:0] Signal den Wert des trq_mca_run_dr[3:0] Signals an und das mca_run_pmw[3:0] Signal nimmt den Wert des trq_run_pmw[3:0] Signals an, mit AND mit dem Gültigkeits-Anforderungs-Signal von der PMWQ verknüpft (pmwq_valid[3:0]). Gates **2410**, **2412**, **2414**, **2416** und **2418** und Multiplexer **2420**, **2422**, **2424** und **2426** sind so angeordnet, um die MCA-Lauf-Signale auf diese Art und Weise zu erzeugen.

DIE WARTESCHLANGEN-BLOCK-ZU-PCI-SCHNITTSTELLE (QPIF)

[0185] Wie wiederum die Fig. 4 und Fig. 75 zeigen, leitet die QPIF **148** den Fluss von Transaktionen zwischen dem Warteschlangen-Block **127** und dem PCI-Bus **32**. Die QPIF **148** liefert auch Transaktionen, initiiert auf dem PCI-Bus **32**, zu der Kabel-Schnittstelle **130**. Die QPIF **148** arbeitet in zwei Moden: einem Master-Mode und einem Slave-Mode. In dem Master-Mode hat die QPIF **148** eine Kontrolle über den PCI-Bus und führt deshalb Transaktionen, vorgesehen für Target-Vorrichtungen auf dem Bus, aus. Eine Master-Zustand-Maschine **2500** in der QPIF **148** sucht Transaktionen von der PMWQ und DRQ auf und führt sie auf dem PCI-Bus aus, wenn sich die QPIF in dem Master-Mode befindet. In dem Slave-Mode empfängt die QPIF **148** Transaktionen, initiiert durch eine Vorrichtung auf dem PCI-Bus, und liefert entweder die angeforderten Informationen zu der initiierenden Vorrichtung (falls die Informationen verfügbar sind) oder sucht die initiierende Vorrichtung erneut auf (falls die Transaktion eine verzögerte Anforderung ist) und führt die Transaktion weiter zu dem eingangsseitigen Chip. Die Transaktion wird auch aufgesucht, wenn der entsprechende eine der Transaktions-Zähler **159** anzeigt, dass der andere Brücken-Chip voll ist, wie dies vorstehend diskutiert ist. Eine Slave-Zustand-Maschine **2502** empfängt eine ankommende Transaktion von dem PCI-Bus und prüft dann die DCQ nach einer entsprechenden Abschluss-Nachricht und/oder führt die Transaktion zu einem Kabel-Nachrichten-Generator **2504** weiter, der wiederum die Transaktion über das Kabel zu dem eingangsseitigen Brücken-Chip weiterführt.

[0186] Unter Bezugnahme auch auf die Fig. 76A und Fig. 76B umfasst die QPIF eine Adressen- und Daten-Verriegelungs-Logik **2506**, die die ankommenden Adressen-Phasen- und Daten-Phasen-Informationen, zugeordnet zu jeder Transaktion, initiiert durch eine Vorrichtung auf dem PCI-Bus, verriegelt. Die QPIF-Slave-Zustand-Maschine **2502** kontrolliert die Betriebsweise der Adressen- und Daten-Verriegelungs-Logik **2506**. Wenn eine neue Transaktion, initiiert auf dem PCI-Bus, für die QPIF vorgesehen ist, stellt die Slave-Zustand-Maschine **2502** ein Adressen-Phasen-Verriegelungs-Signal (reg_latch_first_request) auf, das anzeigt, dass die Adressen-Phasen-Informationen von dem PCI-Bus verriegelt werden sollten. An der nächsten, abfallenden Flanke des PCI-Takt-Signals bewirkt die Aufstellung des reg_latch_first_request Signals, dass ein verzögertes Adressen-Phasen-Verriegelungs-Signal (dly_reg_latch_first request) aufgestellt werden soll. Wenn sowohl das originale als auch das verzögerte Adressen-Phasen-Verriegelungs-Signal aufgestellt sind, erzeugt die Verriegelungs-Logik **2506** ein erstes Verriegelungs-Signal (latch1). Ein Flip-Flop **2508** und ein Gate **2510** sind so angeordnet, um das erste, verriegelnde Signal auf diese Art und Weise zu erzeugen.

[0187] Die Verriegelungs-Logik **2506** lädt die Adressen-Phasen-Informationen von dem PCI-Bus (über die PCI-Schnittstelle) in drei Adressen-Phasen-Register ein, wenn das erste Verriegelungs-Signal aufgestellt ist. Das erste Register ist ein dreißig-Bit-Adressen-Register **2512**, das die Start-Adresse der momentanen Transaktion anzeigt. Wenn das erste Verriegelungs-Signal aufgestellt ist, wird das Adressen-Signal von der PCI-Schnittstelle (p2q_ad[31:2]) in das Adressen-Register **2512** eingeladen. Das Adressen-Register **2512** gibt das Adressensignal aus, verwendet durch die QPIF (q2pif_addr[31:2]). Das zweite Register ist ein Vier-Bit-Befehls-Register **2514**, das den PCI-Befehl-Code von dem PCI-Bus empfängt (p2q_cmd[3:0]), und das QPIF-Befehls-Signal ausgibt (q2pif_cmd[3:0]). Das dritte Register ist ein Drei-Bit-Schlitz-Auswahl-Register **2516**, das das p2q_slot[2:0] Signal empfängt, das anzeigt, welche PCI-Vorrichtung der momentane Bus-Master ist, und gibt das QPIF-Schlitz-Auswahl-Signal aus (q2pif_slot[2:0]).

[0188] Wenn die Adressen-Phase der PCI-Transaktion endet, stellt die Slave-Zustand-Maschine **2502** ein Daten-Phasen-Verriegelungs-Signal auf (reg_latch_second_request), das anzeigt, dass die Daten-Phasen-Informationen von dem PCI-Bus verriegelt werden sollten. An der nächsten, abfallenden Flanke des PCI-Takt-Signals bewirkt das aufgestellte reg_latch_first_request Signal, dass ein verzögertes Daten-Phasen-Verriegelungs-Signal (dly_reg_latch_second_request) aufgestellt wird. Wenn sowohl das originale als auch das verzögerte Daten-Phasen-Verriegelungs-Signal aufgestellt sind, erzeugt die Verriegelungs-Logik **2506** ein zweites Verriegelungs-Signal (latch2). Ein Flip-Flop **2518** und ein Gate **2520** sind so angeordnet, um das zweite Verriegelungs-Signal auf diese Art und Weise zu erzeugen.

[0189] Die Verriegelungs-Logik **2506** lädt dann die Daten-Phasen-Informationen von dem PCI-Bus (über die PCI-Schnittstelle) in drei Daten-Phasen-Register, wenn das zweite Verriegelungs-Signal aufgestellt ist. Das erste Daten-Phasen-Register ist ein Zweiunddreißig-Bit-Daten-Register **2522**, das die Daten empfängt, die der momentanen Transaktion zugeordnet sind, und zwar auf den PCI-Adressen/Daten-Leitungen (p2q_ad[31:0]), und gibt das QPIF-Daten-Signal aus (q2pif_data[31:0]). Das zweite Daten-Phasen-Register ist ein Vier-Bit-Freigabe-Register **2524**, das Freigabe-Bits von dem PCI-Bus empfängt (p2q_cbe[3:0]), und das QPIF-Byte-Freigabe-Signal ausgibt (qq2pif_byte_en[3:0]). Das dritte Register ist ein Drei-Bit-Verriegelungs-Register **2526**, das das PCI-Verriegelungs-Signal empfängt (p2q_lock), das anzeigt, dass die momentane Transaktion als eine verriegelte Transaktion laufen sollte, und das QPIF-Verriegelungs-Signal ausgibt (q2pif_lock).

[0190] Wie wiederum die **Fig. 75** und auch die **Fig. 77** zeigen, umfasst das QPIF einen „Verriegelungs"-Logik-Block **2528**, der den „Verriegelungs"-Zustand des QPIF kontrolliert. Die QPIF besitzt drei Verriegelungs-Zustände: einen nicht verriegelten Zustand **2530** (lock_state[1:0] = „00"), der anzeigt, dass keine verriegelten Transaktionen in der DCQ anhängig sind; einen verriegelten Zustand **2532** (lock_state[1:] = „01"), der anzeigt, dass eine verriegelte Transaktion in der DCQ empfangen worden ist oder sich im Abschluss auf dem PCI-Bus befindet; und einen unlocked-but-retry Zustand **2534** (lock_state[1:0] = „10"), der anzeigt, dass die Verriegelung entfernt worden ist, allerdings eine gepostete Speicher-Schreib-Transaktion, die in dem anderen Brücken-Chip anhängig ist, laufen muss, bevor die nächste Transaktion angenommen werden kann.

[0191] Bei einem Power-up bzw. Einschalten und einem Reset tritt die Verriegelungs-Logik **2528** in den nicht verriegelten Zustand **2530** ein und wartet auf eine verriegelte Transaktion, um in die DCQ einzutreten (angezeigt durch die Aufstellung des dcq_locked Signals). Bei dem ersten Takt-Impuls, nachdem das dcq_locked Signal aufgestellt ist, tritt die Verriegelungs-Logik in den verriegelten Zustand **2532** ein, der die QPIF-Slave-Zustand-Maschine **2502** dazu bringt, alle Transaktions-Anforderungen von dem PCI-Bus erneut zu versuchen. Die PCI-Schnittstelle stellt auch ein Verriegelungs-Signal (p2q_lock) auf, das anzeigt, dass sie den PCI-Bus für die Transaktion verriegelt hat. Nachdem die Verriegelungs-Transaktion abgeschlossen ist und die anfordernde Vorrichtung die verriegelten Abschluss-Daten von der DCQ aufgesucht hat, wird das dcq_locked Signal weggenommen Bei dem ersten Taktimpuls, nachdem das dcq_locked weggenommen ist, während das p2q_lock Signal noch aufgestellt ist, falls keine gepostеten Speicher-Schreib-Vorgänge in dem anderen Brücken-Chip anhängig sind (d. h. das pmw_empty Signal wird durch den Kabel-Decodierer aufgestellt), kehrt die Verriegelungs-Logik **2528** zu dem nicht verriegelten Zustand **2530** zurück und die Slave-Zustand-Maschine **2502** ist wieder in der Lage, Transaktions-Anforderungen zu akzeptieren. Allerdings tritt, falls das pwm_empty Signal nicht bei dem ersten Taktimpuls aufgestellt ist, nachdem das dcq_lock Signal weggenommen ist, die Verriegelungs-Logik **2528** in den unlocked-but-retry Zustand **2534** ein, der die Slave-Zustand-Maschine **2502** dazu bringt, alle Transaktionen erneut zu versuchen, bis der gepostete Speicher-Schreib-Zyklus auf dem anderen PCI-Bus abgeschlossen ist.

[0192] Nachdem der gepostete Speicher-Schreib-Zyklus abgeschlossen ist, wird das pmw_empty Signal aufgestellt, und die Verriegelungs-Logik **2528** kehrt zu dem nicht verriegelten Zustand **2530** zurück.

[0193] Wie wiederum die Fig. 75 und auch die Fig. 78 zeigen, umfasst die QPIF eine Puffer-Flush-Logik **2536**, die bestimmt, wann die die DCQ Daten von einem oder allen deren Daten-Puffer entleeren sollte. Wie vorstehend diskutiert ist, erzeugt die PCI-Schnittstelle in dem ausgangsseitigen Chip ein p2q_flush Signal, wenn der eingangsseitige Chip ein I/O oder config Schreiben oder ein Speicher-Schreiben ausgibt, das das Soll-Speicher-Bereichs-Register (Target Memory Range Register – TMRR) einer ausgangsseitigen Vorrichtung trifft. Die QPIF-Puffer-Flush-Logik **2536** stellt ein QPIF-Flush-Signal auf (general_flush), das den entsprechenden Daten-Puffer oder alle Daten-Puffer entleert (in Abhängigkeit von dem Wert des p2q_slot Signals, wie dies vorstehend diskutiert ist), wenn das p2q_flush Signal empfangen ist. Ansonsten stellt die Puffer-Flush-Logik **2536** das allgemeine Flush-Signal nur dann auf, wenn eine Vorrichtung auf dem sekundären Bus eine verzögerte Anforderung ausgibt, die alle der DCQ-Puffer verfehlt, wenn durch die DCQ-Steuer-Logik geprüft ist (d. h. !dcq_hit und q2pif_check_cyc sind aufgestellt). In jedem Fall wird das general_flush Signal dazu verwendet, nur Puffer zu entleeren, die sich in dem „prefetch" Zustand befinden, wie dies vorstehend diskutiert ist. Deshalb werden Prefetch-Daten in der DCQ gehalten, bis die PCI-Schnittstelle ein Entleeren verlangt oder bis die entsprechende PCI-Vorrichtung eine nicht-sequenzielle Anforderung ausgibt (d. h. verfehlt die DCQ). Gates **2538** und **2540** sind so angeordnet, um das general_flush Signal auf diese Art und Weise zu erzeugen.

[0194] Wenn eine Multi-Threaded-Vorrichtung mehr als einen Abschluss-Puffer zugeordnet besitzt, wobei mindestens einer davon Prefetch-Daten enthält, verbleiben die Prefetch-Daten in dem entsprechenden Puffer so lange, wie die Vorrichtung nicht eine Anforderung ausgibt, die alle die DCQ-Puffer verfehlt. Sobald die Vorrichtung eine neue Anforderung ausgibt, werden alle deren Prefetch-Puffer entleert. Alternativ könnte ein Prefetch-Puffer, zugeordnet einer Multi-Threaded-Vorrichtung, entleert werden, sobald die Vorrichtung eine Anforderung ausgibt, die einen anderen DCQ-Puffer trifft.

[0195] Wie wiederum Fig. 75 zeigt, umfasst die QPIF einen Lese-Befehl-Logik-Block **2542**, der Lese-Befehle von der PCI-Schnittstelle und Prefetch-Befehle von der DCQ empfängt und ein abgehendes Nachrichten-Befehl-Signal (message_cmd) zu dem Kabel zuführt. In Nicht-Streaming-Situationen kann der abgehende Nachrichten-Befehl derselbe wie der Befehl sein, der von dem PCI-Bus oder der DCQ empfangen ist, oder die Lese-Befehl-Logik **2542** kann den Befehl in einen solchen umwandeln, der eine größere Menge an Daten einsetzt. Da Transaktionen, ausgeführt dword-by-dword, länger benötigen, um auf dem Host-Bus abzuschließen, als Transaktionen, die eine gesamte Cache-Zeile von Daten einsetzen, und da einzelne Cache-Zeilen-Transaktionen länger benötigen, um auf dem Host-Bus abzuschließen, als Mehrfach-Cache-Zeilen-Transaktionen, unterstützt die Lese-Befehl-Logik oft „kleinere" Befehle zu „größeren" solchen, um die Zahl von Takt-Zyklen, verbraucht durch die Transaktion, zu reduzieren („read promotion"). Zum Beispiel ist, wenn eine Vorrichtung auf dem sekundären PCI-Bus einen Speicher-Lese-Befehl ausgibt und dann nach jedem dword von Daten in einer Cache-Zeile fragt, die Lese-Befehl-Logik **2542** in der Lage, die Host-Latenzzeit durch Unterstützen des PCI-Befehls zu eine Speicher-Lese-Zeile zu reduzieren, was ermöglicht, dass der eingangsseitige Chip die gesamte Cache-Zeile an Daten auf einmal liest anstelle davon, jedes dword individuell zu lesen.

[0196] Wie auch Fig. 79 zeigt, erzeugt, wenn die DCQ anzeigt, dass eine Lese-Datenfolge eingerichtet worden ist (d. h. dcq_stream_connect ist aufgestellt), wie dies vorstehend diskutiert ist, die Lese-Befehl-Logik **2542** einen Nachrichten-Befehl von „1000", was den eingangsseitigen Chip darüber informiert, dass eine Datenfolge auftritt. Wenn keine Datenfolge eingerichtet worden ist, muss die Lese-Befehl-Logik **2542** entscheiden, ob ein Speicher-Lese-, ein Speicher-Lese-Zeilen- oder ein Speicher-Lese-Mehrfach-Befehl zu verschicken ist. Falls der Befehl, empfangen von dem PCI-Bus, ein Speicher-Lese-(MR)-Befehl ist (q2p_CMD[2:0] entspricht „0110") und das entsprechende Speicher-Lese-zu-Speicher-Lese-Zeilen-Unterstützungs-Bit (cfg2q_mr2mr1) in den Konfigurations-Registern eingestellt ist, erzeugt die Lese-Befehl-Logik **2542** einen Speicher-Lese-Zeilen-Befehl („1110"). Andererseits erzeugt, falls der PCI-Befehl ein Speicher-Lese-Befehl ist, und das entsprechende Speicher-Lese-zu-Speicher-Lese-Mehrfach-Bit (cfg2q_mr2mrm) eingestellt ist, oder falls der Befehl ein Speicher-Lese-Zeilen-Befehl (q2pif_cmd[3:0] gleich zu „1110") von dem PCI-Bus ist oder ein Prefetch-Zeilen-Befehl (dcq_prefetch_linie ist aufgestellt) von der DCQ ist und das entsprechende Speicher-Lese-Zeilen-zu-Speicher-Mehrfach-Bit (cfg2q_mrl2mrm) eingestellt ist, oder falls der Befehl ein Prefetch-Mehrfach-Befehl (dcq_prefetch_mu1) von der DCQ ist, die Lese-Befehl-Logik **2542** einen Speicher-Lese-Mehrfach-Befehl (d. h. message_cmd entspricht „1100"). Falls der Befehl ein Prefetch-Zeilen-Befehl ist und das entsprechende Speicher-Lese-Zeilen-zu-Speicher-Lese-Mehrfach-Bit nicht eingestellt ist, erzeugt die Lese-Befehl-Logik **2542** einen MRL-Befehl („1110"). Ansonsten gibt die Lese-Befehl-Logik **2542** den empfangenen PCI-Befehl (q2pif_cmd[2:0]) als das Nachrichten-Befehl-Signal aus. Gates **2544**, **2546**, **2548**, **2550**, **2552**, **2554**, **2556** und **2558** und Multiplexer **2560**, **2562** und **2564** sind so angeordnet, um das message_cmd Signal auf diese Art und Weise zu erzeugen.

[0197] Wie wiederum Fig. 75 zeigt, erzeugt, wenn die QPIF im Master-Mode arbeitet und eine Steuerung

über den Bus empfangen hat, um eine Transaktion, gespeichert in der PMWQ, laufen zu lassen, ein Schreib-Befehl-Logik-Block **2566** den Befehl-Code, der auf dem PCI-Bus ausgeführt wird. Um Transaktions-Zeit zu reduzieren, wie dies vorstehend diskutiert ist, kann eine Schreib-Befehl-Logik Speicher-Schreib-(MW)-Befehle umwandeln, was Daten-Übertragungen ein dword zu einem Zeitpunkt in Speicher-Schreib- und Ungültigkeits-Befehle (MWI) hinein einsetzt, was Übertragungen von mindestens einer gesamten Cache-Zeile an Daten einsetzt. Der Schreib-Befehl-Logik-Block **2566** kann einen Befehl-Midstream umwandeln, wenn z. B. die Transaktion als ein Speicher-Schreiben in der Mitte einer Cache-Zeile beginnt und Daten enthält, die die nächste Cache-Zeilen-Grenze kreuzt und alle acht dwords an Daten in der nächsten Cache-Zeile umfasst. In dieser Situation beendet die Schreib-Befehl-Logik **2566** die Speicher-Schreib-Transaktion, wenn sie die erste Cache-Zeilen-Grenze erreicht, und initiiert ein Speicher-Schreiben und erklärt eine Transaktion für ungültig, um nächste, volle Cache-Zeilen an Daten zu übertragen. Die Schreib-Befehl-Logik **2566** kann auch einen MWI-Transaktions-Midstream zugunsten einer MW-Transaktion beenden, falls weniger als eine Cache-Zeile an Daten zu dem Target-Bus hin geschrieben werden soll, nachdem eine Cache-Zeilen-Grenze gekreuzt wird.

[0198] Wie wiederum Fig. 75 und auch Fig. 80 zeigen, hält die Slave-Zustand-Maschine **2502** auch zwei Zähler aufrecht, die anzeigen, wenn eine gepostete Schreib-Transaktion, initiiert auf dem PCI-Bus, beendet werden sollte. Ein 4 K Seiten-Grenzen-Zähler (Page Boundary Counter) **2594** erzeugt ein Seiten-Zähl-Signal (page_count_reg[11:2]), das anzeigt, wenn Daten, übertragen von dem PCI-Bus, eine 4 K Seiten-Grenze erreichen. Da ein einzelner Speicher-Zugriff nicht erlaubt ist, um eine 4 K Seiten-Grenze zu überqueren, muss die gepostete Schreib-Transaktion auf dem initiierenden Bus beendet werden, wenn eine Grenze erreicht ist. Der 4 K Seiten-Grenzen-Zähler **2594** wird mit dem dritten bis zwölften Bit der Transaktions-Adresse (q2pif_addr[11:2]) geladen, wenn die Zustand-Maschine ein load_write_counter Signal aufstellt (die Umstände, die eine Aufstellung vor, diesem Signal mit sich bringen, werden in weiterem Detail nachfolgend diskutiert). Der Zähler **2594** erhöht sich dann um eins an der ansteigenden Flanke jedes Takt-Impulses, nachdem das load_write_counter Signal weggenommen ist. Der Zähler **2594** wird nicht bei Takt-Impulsen erhöht, während denen die intiierende Vorrichtung einen Initiator-Warte-Zustand eingesetzt hat (d. h. p2q_irdy aufgestellt). Der Ausgang von Gate **2592** bestimmt, wann dem Zähler erlaubt wird, sich zu erhöhen. Wenn alle Bits in dem page_count_reg[11:2] Signal hoch sind, ist eine 4 K Seiten-Grenze erreicht worden und die Slave-Zustand-Maschine muss die gepostete Schreib-Transaktion beenden und erneut die initiierende Vorrichtung versuchen.

[0199] Ein dword Zähler **2598** erzeugt ein pmw_counter[5:0] Signal, das die Zahl von dwords, geschrieben von dem initiierenden Bus, während einer geposteten Schreib-Transaktion anzeigt. Das pmw_counter[5:0] Signal wird dann verwendet, um anzuzeigen, wann ein Überlauf aufgetreten ist oder wann die letzte Zeile der Transaktion erreicht worden ist, wie dies nachfolgend diskutiert ist. Wenn die Slave-Zustand-Maschine **2503** das load_write_counter Signal aufstellt, werden die dritten bis fünften Bits des Adressen-Signals (q2pif_addr[4:2]) in die unteren drei Bits des Zählers **2598** eingeladen, während die oberen drei Bits auf Null gesetzt werden. Dieses Adressen-Offset zeigt an, an welchem dword in einer Cache-Zeile eine gepostete Schreib-Transaktion gestartet wurde. Der Zähler **2598** erhöht sich dann um eins an der ansteigenden Flanke jedes Takt-Impulses, nachdem das load_write_counter Signal weggenommen ist. Der Zähler **2598** wird nicht bei Takt-Impulsen erhöht, während denen die initiierende Vorrichtung einen Initiator-Warte-Zustand eingesetzt hat (d. h. p2q_irdy ist aufgestellt). Der Ausgang des Gates **2596** bestimmt, wann dem Zähler erlaubt wird, sich zu erhöhen. Wenn alle Bits in dem pmw_counter[5L0] Signal hoch sind, hat das gepostete Schreiben das Ende der achten Cache-Zeile erreicht.

[0200] Wie wiederum die Fig. 81A bis Fig. 81C zeigen, erzeugt der Schreib-Befehl-Logik-Block **2566** ein Vier-Bit-Schreib-Befehl-Signal (write_cmd[3:0]), das den Befehl-Code der geposteten Schreib-Transaktion anzeigt, die auf dem PCI-Bus ausgeführt werden soll. Falls der Befehl-Code, gespeichert in der PMWQ, einen Speicher-Schreib- und Ungültigkeits-Befehl darstellt (pmwq_cmd[3] = „1"), erzeugt die Schreib-Befehl-Logik **2566** einen Schreib-Befehl-Code von „1111". Falls der PMWQ-Befehl-Code einen Speicher-Schreib-Befehl darstellt, verriegelt sich die Schreib-Befehl-Logik **2566** an dem Speicher-Schreib-zu-Speicher-Schreib- und Ungültigkeits-Konfigurations-Bit (cfg2q_mw2mwi) entsprechend dem Target-PCI-Schlitz. Falls das cfg2q_mw2mwi Bit nicht eingestellt ist, erzeugt die Schreib-Befehl-Logik **2566** einen Speicher-Schreib-Befehl („0111"). Falls das Konfigurations-Bit eingestellt ist, erzeugt die Schreib-Befehl-Logik **2566** einen MWI-Befehl, falls die nächste Zeile in dem PMWQ-Daten-Puffer voll ist (pmwg_full_line ist aufgestellt), und erzeugt einen MW-Befehl ansonsten. Die Multiplexer **2568** und **2570** sind so angeordnet, um das write_cmd Signal auf diese Art und Weise zu erzeugen.

[0201] Wenn die QPIF eine Transaktion auf dem PCI-Bus ausführt und eine Cache-Zeilen-Grenze erreicht hat, kann die Schreib-Befehl-Logik **2566** ein new_write_cmd Signal aufstellen, das anzeigt, dass die momen-

tane Transaktion zugunsten eines neuen Schreib-Befehls beendet werden muss. Falls die Transaktion die letzte Cache-Zeile in dem PMWQ-Daten-Puffer erreicht hat (d. h. pmwq_pointer[5:3] entspricht „111"), wird das new write Befehl-Signal aufgestellt, um anzuzeigen, dass die Transaktion beendet werden sollte, falls der nächste PMWQ-Puffer nicht ein Überlauf-Puffer ist, gültige Daten enthaltend, falls das entsprechende cfg2q_mw2mwi Bit nicht eingestellt ist, oder falls die full_line Bits entsprechend zu der momentanen Cache-Zeile und der nächsten Cache-Zeile unterschiedlich sind (d. h. pmwq_full_line[7] entspricht nicht pmwq_next_full_line). Falls die Transaktion nicht das Ende des PMWQ-Puffers erreicht hat, wird das new_write_cmd Signal aufgestellt, entweder falls die nächste Zeile in dem PMWQ-Puffer nicht gültige Daten enthält (!pmwq_valid_lines[x + 1]), oder falls das cdfg2q_mw2mwi Bit eingestellt ist und die Voll-Zeilen-Bits für die momentane Zeile und die nächste Zeile unterschiedlich sind (d. h. pmwq_full_line[x] entspricht nicht pmwq_full_line[x + 1]). Gates **2572**, **2474**, **2576**, **2578** und **2580** und ein Multiplexer **2582** sind so angeordnet, um das new write Befehl-Signal auf diese Art und Weise zu erzeugen.

[0202] Nachdem das new_write_cmd Signal aufgestellt ist, wird die Transaktion nicht beendet, bis der Schreib-Befehl-Logik-Block **2566** ein synchrones, neues Schreib-Befehl-Signal aufstellt (held_new_write_cmd). Das held_new_write_cmd Signal wird an dem ersten Taktimpuls aufgestellt, nachdem das new_write_cmd Signal aufgestellt ist und das end_of_line Signal aufgestellt ist, was anzeigt, dass das Ende der Cache-Zeile erreicht worden ist, so lange wie die PCI-Schnittstelle nicht die Transaktion beendet hat (d. h. p2q_start_pulse ist aufgestellt). Der held_new_write Befehl wird bei einem Reset und bei dem ersten Takt-Impuls weggenommen, nachdem die new_write_cmd, end of-line und p2q_start_pulse Signale weggenommen sind und die QPIF die Transaktion beendet (d. h. das asynchrone early_cyc_complete Signal ist aufgestellt). Ansonsten behält das held_new_write_cmd Signal seinen momentanen Wert. Gates **2584** und **2586**, ein Invertierer **2588** und ein Flip-Flop **2590** sind so angeordnet, um das held_new_write_cmd Signal auf diese Art und Weise zu erzeugen.

[0203] Wie wiederum Fig. 75 und auch Fig. 82A zeigen, umfasst die QPIF einen Überlauf-Logik-Block **2600**, der der Master-Zustand-Maschine **2500** ermöglicht, Überlauf-Daten zu verwalten, falls irgendwelche vorhanden sind, wenn eine gepostete Schreib-Transaktion auf dem Target-Bus ausgeführt wird. Wenn die QPIF ein Transaktions-Lauf-Signal (mca_run_pmw oder mca_run_dr, wie vorstehend diskutiert ist) von dem MCA empfängt, erzeugt die Überlauf-Logik **2600** ein Zwei-Bit-Initial-Warteschlangen-Auswahl-Signal (start_queu_select[2:0]), was anzeigt, welcher der Puffer in der PMWQ oder der DRQ ausgewählt werden sollte, um die momentane Transaktion laufen zu lassen. Die folgende Tabelle stellt dar, wie das start_queue_select Signal erzeugt wird.

MCA Run Code	start_queue_select
00000001	00
00000010	01
00000100	10
00001000	11
00010000	00
00100000	01
01000000	10
10000000	11

Erzeugung eines start_queue_select Signals

[0204] Wenn die QPIF eine gepostete Schreib-Transaktion auf dem Target-Bus ausführt, wird ein Zwei-Bit-QPIF-Warteschlangen-Auswahl-Signal (q2pif_queue_select[1:0]) verwendet, um den geeigneten Puffer in der PMWQ auszuwählen. Wenn die Transaktion initiiert wird, stellt die Master-Zustand-Maschine **2500** ein Warteschlangen-Auswahl-Signal auf (initial_queue_select), das bewirkt, dass das q2pif_queue_select Signal den Wert des anfänglichen Warteschlangen-Auswahl-Signals annimmt (start_queue_select). Zu demselben Zeitpunkt wird ein Warteschlangen-Auswahl-Zähler **2602** mit dem Wert des start_queue_select Signals geladen. Nachdem das initial_queue_select Signal weggenommen ist, nimmt

das q2pif_queue_select Signal den Wert des count_queue_select Signals an, erzeugt durch den Zähler **2602**. Wenn die gepostete Speicher-Schreib-Transaktion in den nächsten PMWQ-Puffer überläuft, stellt die Master-Zustand-Maschine **2500** ein Erhöhungs-Warteschlangen-Auswahl-Signal (inc_queue_select) auf, das bewirkt, dass sich der Zähler **2602** um Eins erhöht. Als Folge wird das q2pif_select_signal erhöht und der nächste Puffer in der PMWQ wird ausgewählt, um die Transaktion fortzuführen. Ein Mulitplexer **2604** bestimmt den Wert des q2pif_queue_select Signals.

[0205] Wie auch Fig. 82B zeigt, stellt die Überlauf-Logik **2600** ein overflow_next_queue Signal auf, wenn die Master-Zustand-Maschine **2500** fortführen sollte, Informationen von dem nächsten PMWQ-Puffer während einer geposteten Speicher-Schreib-Transaktion zu sammeln. Unter Verwendung des q2pif_queue_select[1:0] Signals um zu bestimmen, welche PMWQ momentan ausgewählt ist, stellt die Überlauf-Logik **2600** das overflow_next_queue Signal auf, wenn das gültige Bit (pmwq_valid) und ein Überlauf-Bit (pwq_overflow) entsprechend zu dem nächsten PMWQ-Puffer eingestellt sind. Die pmwq_valid und die pmwq_overflow Zeichen werden nachfolgend diskutiert. Gates **2606**, **2608**, **2610** und **2612** und ein Multiplexer **2614** sind so angeordnet, um das overflow_next_queue Signal auf diese Art und Weise zu erzeugen.

[0206] Wie wiederum Fig. 75 zeigt, umfasst die QPIF einen Lese-Ausrichtungs-Logik-Block **2616**, der der QPIF ermöglicht, fehlausgerichtete Speicher-Lese-Zeilen- und Speicher-Lese-Mehrfach-Transaktionen zu korrigieren. Eine Lese-Zeilen Korrektur tritt dann auf, wenn die QPIF, während sie in dem Master-Mode arbeitet, eine MRL- oder MRM-Transaktion empfängt, die in der Mitte einer Cache-Zeile beginnt. Um Transaktions-Zeit zu reduzieren, beginnt die QPIF die Lese-Transaktion an der Cache-Zeilen-Grenze und ignoriert die nicht angeforderten dwords anstelle eines individuellen Lesens nur der angeforderten dwords von Daten.

[0207] Unter Bezugnahme auch auf Fig. 83 aktiviert die Lese-Ausrichtungs-Logik **2616** das Lese-Ausrichtungs-Merkmal durch Aufstellen eines align_read Signals. Dieses Signal wird darin aufgestellt, wenn der Befehl, gespeichert in dem entsprechenden DRQ-Puffer, ein Speicher-Lese-Zeilen oder ein Speicher-Lese-Mehrfach-Befehl ist (d. h. drq_cmd[3:0] entspricht „1110" oder „1100" jeweils), und wenn das Lese-Ausrichtungs-Konfigurations-Bit (cfg2q_read_align) entsprechend zu der Target-PCI-Vorrichtung eingestellt ist. Gates **2618** und **2620** sind so angeordnet, um das align_read Signal auf diese Art und Weise zu erzeugen.

[0208] Wie wiederum die Fig. 84A bis Fig. 84C zeigen, umfasst die Lese-Ausrichtungs-Logik **2616** einen Lese-Ausrichtungs-Abwärts-Zähler **2622**, der die dwords von jeder Cache-Zeilen-Grenze zählt und anzeigt, wenn die Master-Zustand-Maschine **2500** das erste, angeforderte dword erreicht. Der Zähler **2622** umfasst eine Zustand-Maschine **2624**, die die Betriebsweise des Zählers **2622** steuert.

[0209] Bei einem Reset tritt der Zähler **2622** in einen IDLE_CNT Zustand **2626** ein, in dem keine Zählung auftritt. Wenn der MCA die QPIF dahingehend instruiert, eine verzögerte Anforderungs-Transaktion auf dem PCI-Bus laufen zu lassen (d. h. wenn irgendwelche Bits in dem mca_run_dr[3:0] aufgestellt sind), stellt die QPIF ein verzögertes Anforderungs-Lauf-Signal auf (any_drq_run), was anzeigt, dass sie versucht, eine verzögerte Anforderungs-Transaktion laufen zu lassen. Während sich der Zähler in dem IDLE_CNT Zustand **2622** befindet, wird sein Drei-Bit-Ausgangs-Signal (throw_cnt[2:1]) mit dem dword offset der Transaktion-Adresse (drq_addr[4:2]) geladen, wenn das any_rundrq Signal aufgestellt ist und die QPIF die Steuerung des PCI-Busses erhält (d. h. p2q_ack wird aufgestellt). Das Gate **2623** erzeugt das Lade-Freigabe-Signal. Dann tritt, an der ansteigenden Flanke des nächsten PCI-Takt-Zyklus, der Zähler **2622** in den COUNT Zustand **2628** ein. Falls die Transaktion an einer Cache-Zeilen-Grenze beginnt, gleicht das dword offset „000" und keine Zählung wird benötigt. Wenn eine Lese-Ausrichtung aktiviert wird, beginnt die Master-Zustand-Maschine **2500** jede MRL- und MRM-Transaktion an der Cache-Zeilen-Grenze, ungeachtet der tatsächlichen Start-Adresse.

[0210] Während sich der Zähler **2622** in dem COUNT Zustand **2628** befindet, verringert er sich um Eins bei jedem Takt-Impuls, solange wie das p2q_ack Signal aufgestellt ist, throw_cnt nicht Null erreicht hat, sich die Transaktion in der Daten-Phase befindet (d. h. das asynchrone Signal eary_data_phase ist aufgestellt) und die Target-Vorrichtung nicht einen Target-Ready-Warte-Zustand (!p2q_trdy) ausgegeben hat. Das Gate **2625** bestimmt, wann der Zähler erniedrigt wird. Falls die PCI-Schnittstelle den Bus von der QPIF wegnimmt (p2q_ack wird weggenommen) oder falls die Daten-Phase endet (early_data_phase wird weggenommen), beendet der Zähler **2622** ein Erniedrigen und tritt erneut in den IDLE_CNT Normalzustand **2626** ein. Falls das throw_cnt Signal „000" erreicht, während das p2q_ack Signal noch aufgestellt ist, stoppt der Zähler **2622** ein Zählen und tritt in den DONE Zustand **2630** ein. Ansonsten verbleibt der Zähler in dem COUNT Zustand **2628**.

[0211] Wenn der Zähler „000" erreicht, stellt die Lese-Ausrichtungs-Logik **2616** ein read_data_start Signal auf, das die Master-Zustand-Maschine **2500** instruiert, ein Lesen von Daten von der Target-Vorrichtung zu be-

ginnen. Ein Komparator **2632** erzeugt das read_data_start Signal. Nachdem das read_data_start Signal aufgestellt ist, verbleibt der Zähler **2622** in dem DONE Zustand **2630** bis die Daten-Phase endet (early_data_phase wird weggenommen).

[0212] Wie Fig. 85 zeigt, steuert die Master-Zustand-Maschine die Betriebsweise der QPIF, wenn die QPIF in dem Master-Mode arbeitet. In dem Master-Mode führt die QPIF gepostete Schreib-Transaktionen und verzögerte Anforderungs-Transaktionen auf dem PCI-Bus aus. Die folgende Tabelle stellt die Ereignisse dar, die Zustand-Übergänge in der Master-Zustand-Maschine bewirken.

MASTER STATE MACHINE		
Current State	Event	Next State
IDLE	A=(any_run&&!cable_busy&&!p2q_master_dphase) | (any_run_drq && tc_dc_full)	IDLE
IDLE	B: p2q_ack && q2p_dac_flag	MASTER_DAC
IDLE	C: p2q_ack && any_drq_run	RDATA1
IDLE	D: p2q_ack && !(q2p_dac_flag | any_drq_run)	WDATA1
MASTER_DAC	E: p2q_ack && any_drq_run && p2q_start_pulse	RDATA1
	F: p2q_ack && p2q_start_pulse &&!any_drq_run	WDATA1
	G: !p2q_ack	IDLE
RDATA1	H: !p2q_ack	IDLE
	I: p2q-ack && p2q_start_pulse	RBURST
	J: p2q_ack && !p2q_start-pulse	RDATA1
RBURST	K: !p2q_ack | p2q_retry | p2q_target_abort | (queue_cyc_complete && !(!p2q_last_dphase && p2q_master_dphase && cd_stream && stream_match && !cfg2q_stream_disable) && !p2q_trdy) | (read_page_disconnect && !p2q_trdy)	IDLE
	L: p2q_ack && !p2q_retry && !p2q_target_abort && ((read_page_disconnect && p2q_trdy) | (queue_cyc_ complete && ((!p2q_last_dphase && p2q_master_dphase && cd_stream && stream_match && !cfa2q_stream_disable) | p2q_trdy)) | !p2q_trdy | otherwise)	RBURST
WDATA1	M: !p2q_ack | p2q_retry | p2q_target_abort | ((queue_cyc_complete | hold_new_write_cmd | end_of_line && new_write_cmd | p2q_last_dphase | sl_p2q_last_dphase) && !p2q_trdy)	IDLE
	N: p2q_ack && !p2q_retry && !p2q_target_abort && (queue_cyc_complete | hold_new_write_cmd | end_of_line && new_write_cmd | p2q_last_dphase | sl_p2q_last_dphase) && p2q_trdy	WDATA1
	O: otherwise	WDATA2
WDATA2	P: !p2q_ack | (p2q_retry&&!p2q_trdy) | p2q_target_abort	IDLE
	Q: p2q_ack && p2q_retry && p2q_trdy	WRETRY
	R: p2q_ack && !p2q_retry && !p2q_target_abort && (queue_cyc_complete | end_of_line && new_write_ cmd) && (!p2q_trdy | p2q_start_pulse)	WSHORT_BURST
	S: otherwise	WDATA2
WRETRY	T: Always	IDLE
WSHORT_BURST	U: !p2q_ack | p2q_retry | p2q_target_abort	IDLE
	V: p2q_ack && !p2q_retry && !p2q_target_abort &&((overflow_next_queue && !new_write_cmd && !p2q_trdy) | !p2q_trdy)	WCOMPLETE
	W: otherwise	WSHORT_BURST
WCOMPLETE	X: p2q_retry | p2q_target_abort | (!(overflow_next_ queue && !new_write_cmd && !p2q_last_dphase) && !p2q_trdy)	IDLE
	Y: !p2q_retry &&!p2q_target_abort && ((overflow_ next_queue &&!new_write_cmd && !p2q_last_dphase) &&!p2q_trdy)	WDATA1
	Z: otherwise	WCOMPLETE

Master-Zustand-Transaktionen

[0213] Bei einem Reset tritt die Master-Zustand-Maschine in einen IDLE Zustand **2700** ein, in dem die QPIF auf Instruktionen wartet, um eine Transaktion auf dem PCI-Bus laufen zu lassen. Wenn die QPIF ein Lauf-Signal von dem MCA empfängt (any_run wird aufgestellt, wenn irgendein Bit in dem mca_run_pmw Signal oder dem mca_run_dr Signal aufgestellt ist), ist das Kabel nicht belegt, eine Nachricht weiterleitend (!cable_busy), und die PCI-Schnittstelle versucht nicht, die vorherige Transaktion zu beenden (!p2q_master_dphase), die Master-Zustand-Maschine versucht, die Transaktion auf dem PCI-Bus laufen zu lassen. Falls die Transaktion eine verzögerte Anforderungs-Transaktion ist (any_run_drq ist aufgestellt) und der andere Chip keinen Raum für einen verzögerten Abschluss hat (tc_dc_full ist aufgestellt), ist die Master-Zustand-Maschine nicht in der Lage, die Anforderung laufen zu lassen, und führt den MCA zu der nächsten Transaktion weiter. Ansonsten beginnt, falls die PCI-Schnittstelle die QPIF Steuerung des Busses (p2q_ack ist aufgestellt) gegeben hat, die Master-Zustand-Maschine damit, die Transaktion auf dem PCI-Bus auszuführen. In dem IDLE Zustand **2700** liefert der Master die Adressen-Phasen-Informationen, die vorstehend diskutiert sind, zu dem PCI-Bus. Falls die Transaktion, die laufen soll, ein Dual-Adressen-Zyklus ist (q2pif_dac_flag ist aufgestellt), tritt die Master-Zustand-Maschine in einen MASTER_DAC Zustand **2702** ein, in dem die zweite Hälfte der Adressen-Informationen geliefert wird. Falls die Transaktion nicht ein Dual-Adressen-Zyklus ist und eine verzögerte Anforderungs-Transaktion ist (any_run_drq ist aufgestellt), dann tritt die Master-Zustand-Maschine in einen RDATA1 Lese-Zustand **2704** ein, indem die Master-Zustand-Maschine die Datenphase der verzögerten Anforderungs-Transaktion beginnt. Falls die Transaktion nicht ein Dual-Adressen-Zyklus ist und nicht eine verzögerte Anforderung ist, ist sie eine gepostete Speicher-Schreib-Transaktion, so dass die Master-Zustand-Maschine in einen WDATA1 Schreib-Zustand **2706** eintritt, indem die Master-Zustand-Maschine in die Datenphase der geposteten Speicher-Schreib-Transaktion eintritt.

[0214] In dem MASTER_DAC Zustand **2704** = 2 liefert die Master-Zustand-Maschine die zweite Hälfte der Adressen-Phasen-Informationen. Dann tritt, falls das p2q_ack Signal noch aufgestellt ist und die Transaktion eine verzögerte Anforderung ist, die Master-Zustand-Maschine in den RDATA1 Zustand **2704** ein, wenn sie das Start-Signal (p2q_start_pulse) von der PCI-Schnittstelle empfängt. Falls die Transaktion nicht eine verzögerte Anforderung ist, tritt die Master-Zustand-Maschine in den WDATA1 Zustand **2706** ein, wenn sie den PCI-Start-Impuls empfängt. Die Master-Zustand-Maschine initiiert auch eine verzögerte Abschluss-Nachricht auf dem Kabel, wenn der PCI-Start-Impuls empfangen ist, durch Aufstellen eines Asynchron-Abschluss-Nachrichten-Signals (early_master_send_message). Falls das p2q_ack Signal durch die PCI-Schnittstelle weggenommen worden ist, kehrt die Master-Zustand-Maschine zu dem IDLE Zustand **2700** zurück und wartet darauf, erneut die Transaktion zu versuchen.

[0215] Der RDATA1 Zustand **2704** ist der Anfangs-Zustand für verzögerte Lese- und verzögerte Schreib-Anforderungen. In diesem Zustand wartet die Master-Zustand-Maschine auf den PCI-Start-Impuls, und zwar vor Eintreten in eine RBURST Burst-Daten-Phase **2708**. Wenn die Zustand-Maschine zuerst in den RDATA1 Zustand **2704** eintritt, initiiert sie eine Abschluss-Nachricht auf dem Kabel, falls dies nicht schon bereits in dem MASTER_DAC Zustand **2702** vorgenommen ist. Dann beendet, falls das p2q_ack durch die PCI-Schnittstelle weggenommen ist, die Master-Zustand-Maschine die Transaktion, führt den MCA zu der nächsten Transaktion weiter, und tritt erneut in den IDLE Zustand **2700** ein. Ansonsten präpariert, wenn der PCI-Start-Puls erscheint, sich die Master-Zustand-Maschine so, um in den RBURST Zustand **2708** einzutreten. Falls die QPIF das Ende der Transaktion anzeigt (queue_cyc_complete) oder falls die Transaktion eine 4 K Seiten-Grenze erreicht hat (read_page_disconnect ist aufgestellt, da alle Bits in dem drq_addr [11:2] Signal hoch sind), entfernt die Master-Zustand-Maschine das frame_ signal von der QPIF und zeigt an, dass der nächste Teil an Daten der letzte Teil ist (asynchrones Signal early_last_master data ist aufgestellt), bevor in den RBURST Zustand **2708** eingetreten wird. Die Master-Zustand-Maschine stellt auch ein asynchrones early_master_lastline Signal auf, das anzeigt, dass die letzte Zeile von Daten erreicht worden ist, falls das read_page_disconnect_lastline Signal aufgestellt ist oder falls das DRQ-Last-Zeilen-Signal (drq_lastline) aufgestellt ist und die QPIF nicht ein Streaming-Signal von dem anderen Brücken-Chip empfangen hat (cd_stream oder stream_match sind nicht aufgestellt oder cfg2q_stream_disable ist nicht eingestellt). Falls der PCI-Start-Impuls nicht aufgestellt ist, verbleibt die Master-Zustand-Maschine in dem RDATA1 Zustand **2704**, bis die QPIF die Transaktion beendet oder eine 4 K Seiten-Grenze erreicht ist, was die Zustand-Maschine zu dem IDLE Zustand **2700** zurückführen wird, oder bis der PCI-Start-Impuls erscheint, was die Zustand-Maschine dazu bringt, in den RBURST Zustand **2708** einzutreten.

[0216] In dem RBURST Zustand **2708** führt die Master-Zustand-Maschine burstmäßig Daten zu dem PCI-Bus. Falls eine Abschluss-Nachricht bis jetzt noch nicht initiiert worden ist, initiiert die Master-Zustand-Maschine eine Abschluss-Nachricht unter Eintreten in den RBURST Zustand **2708**. Dann beendet, falls das p2q_ack Signal weggenommen ist, oder falls die QPIF Transaktion erneut durch die PCI-Schnittstelle versucht wird (p2q_retry ist aufgestellt), oder falls die PCI-Schnittstelle die Transaktion aussondert (p2q_target_abort ist

aufgestellt), beendet die Master-Zustand-Maschine die Transaktion auf dem PCI-Bus, sondert die Abschluss-Nachricht auf dem Kabel aus und kehrt zu dem IDLE Zustand zurück. Wenn das p2q_ack Signal weggenommen ist, fährt der Master-Zyklus-Arbitrierer fort, die momentane Transaktion auszuwählen. Wenn allerdings die Transaktion erneut aufgesucht oder ausgesondert ist, führt die Master-Zustand-Maschine den MCA zu der nächsten Transaktion weiter.

[0217] Während das p2q_ack Signal noch aufgestellt ist und die QPIF-Transaktion nicht erneut versucht oder ausgesondert wird, beendet die Master-Zustand-Maschine niemals die Transaktion und kehrt zu dem IDLE Zustand **2700** zurück, falls eine 4 K Seiten-Grenze erreicht ist und die PCI-Schnittstelle anzeigt, dass die Target-Vorrichtung aufgehört hat, Daten aufzunehmen (p2q_trdy ist nicht länger aufgestellt). Falls die Target-Vorrichtung den letzten Teil von Daten nimmt, verbleibt die Zustand-Maschine in dem RBURST Zustand **2708**.

[0218] Falls die QPIF das queue_cyc_complete Signal aufstellt, was anzeigt, dass die Transaktion abgeschlossen ist, wird der Master allgemein die Transaktion beenden und zu dem IDLE Zustand **2700** zurückkehren, falls das p2q_trdy Signal weggenommen ist oder in dem RBURST Zustand **2708** verbleibt, bis das letzte dword an Daten übertragen wird, falls das p2q_trdy Signal aufgestellt verbleibt. Falls sich allerdings die Transaktion in der Daten-Phase befindet und sich nicht in der letzten Daten-Phase befindet (p2q_master_dphase und !p2q_last_dphase) und eine Datenfolge mit dem anderen Brücken-Chip eingerichtet worden ist (cd_stream und stream_match und !cfg2q_stream_disable), wird die Master-Zustand-Maschine in der RBURST-Phase indefinit verbleiben. Wenn sich die QPIF in einem Streaming-Vorgang befindet, stellt die Master-Zustand-Maschine ein Streaming-Signal auf (q2pif_streaming), das die QPIF dazu bringt, fortzufahren, Daten zu der anfordernden Vorrichtung auf dem anderen PCI-Bus zu liefern, bis die Vorrichtung die Transaktion beendet.

[0219] Falls das p2q_ack Signal aufgestellt verbleibt und weder das p2q_retry, das p2q_target_abort oder das queue_cyc_complete Signal aufgestellt ist, sieht die Master-Zustand-Maschine bei dem p2q_trdy Signal nach. Falls das Signal nicht aufgestellt ist, was anzeigt, dass die Target-Vorrichtung den letzten Teil von Daten genommen oder geliefert hat, stellt die Master-Zustand-Maschine deren nächstes Daten-Signal auf (early_next_data), das anzeigt, dass die QPIF bereit ist, einen anderen Teil von Daten zu übertragen. Das nächste Daten-Signal wird nur dann aufgestellt, falls die Transaktion nicht eine korrekte Lesung ist (align_read ist nicht aufgestellt) oder falls die Transaktion eine korrekte Lesung ist und das read_data_start Signal aufgestellt worden ist. Falls das p2q_trdy Signal aufgestellt ist, was anzeigt, dass das Target nicht die letzte Daten-Übertragung durchgeführt hat, verbleibt die Zustand-Maschine in dem RBURST Zustand **2708**.

[0220] In dem WDATA1 Zustand **2706** beginnt die Master-Zustand-Maschine die Daten-Phase einer gepostеten Speicher-Schreib-Transaktion. Falls das p2q_ack Signal weggenommen ist oder die p2q_retry oder das p2q_target_abort Signal aufgestellt ist, während sich die Master-Zustand-Maschine in diesem Zustand befindet, wird die Transaktion auf dem PCI-Bus beendet und die Zustand-Maschine kehrt zu dem IDLE Zustand **2700** zurück. Wenn das p2q_ack Signal weggenommen ist, verbleibt der MCA bei dem momentanen Zyklus; ansonsten führt die Master-Zustand-Maschine die MCA schrittmäßig zu der nächsten Transaktion weiter.

[0221] Falls das p2q_ack Signal aufgestellt verbleibt und die Transaktion weder erneut aufgesucht noch ausgesondert wird, muss die Master-Zustand-Maschine bestimmen, ob der Schreibvorgang ein einzelnes dword oder mehr als ein dword einsetzt. Falls in dem WDATA1 Zustand das queue_cyc_complete Signal aufgestellt ist, wird das neue Halte-Schreib-Befehl-Signal aufgestellt, die end_of_line und new_write_cmd Signale werden aufgestellt, oder die Transaktion hat das letzte dword von Daten erreicht, die Transaktion setzt ein einzelnes dword ein. In dieser Situation endet die Transaktion und die Zustand-Maschine kehrt zu dem IDLE Zustand **2700** nur dann zurück, wenn das Target den letzten Teil von Daten nahm (!p2q_trdy). Ansonsten verbleibt die Zustand-Maschine in dem WDATA2 Zustand **2710**. Falls die Transaktion mehr als ein dword von Daten einsetzt, tritt die Master-Zustand-Maschine in einen WDATA2-Burst-Daten-Phasen-Zustand **2710** ein. Unmittelbar vor einem Eintreten in den WDATA2 Zustand setzt die Master-Zustand-Maschine einen q2p_irdy Warte-Zustand ein, falls das overflow_next_queue Signal aufgestellt worden ist.

[0222] In dem WDATA2 Zustand **2710** leitet die Master-Zustand-Maschine die Daten burstmäßig zu dem PCI-Bus weiter. Falls das p2q_ack Signal weggenommen ist oder die Transaktion durch die PCI-Schnittstelle ausgesondert ist, wird die Transaktion in der QPIF beendet und die Master-Zustand-Maschine tritt erneut in den IDLE Zustand **2710** ein. Falls die Transaktion erneut durch die PCI-Schnittstelle versucht wird, allerdings die PCI-Schnittstelle die Daten nahm, die geliefert sind (!p2q_trdy), tritt die Master-Zustand-Maschine erneut in den IDLE Zustand **2700** ein. Ansonsten tritt die Zustand-Maschine in einen WRETRY-Stepback-Zustand **2712** ein, der die PMWQ außerhalb des Hinweiszeigers zurück zu dem vorherigen Teil von Daten durch Erzeu-

gen des Rückschreit-Signals, das vorstehend diskutiert ist, führt. Von dem WRETRY Zustand **2712** tritt die Zustand-Maschine immer wieder in den IDLE Zustand **2700** ein.

[0223] Falls das p2q_ack Signal aufgestellt verbleibt und die Transaktion weder erneut versucht noch ausgesondert ist, bestimmt die Master-Zustand-Maschine, ob die Transaktion abgeschlossen ist. Falls die QPIF das Ende der Transaktion anzeigt (queue_cyc_complete ist aufgestellt) oder das Ende einer Cache-Zeile erreicht ist und ein neuer Schreib-Befehl benötigt wird (end_of_line und new_write_command sind aufgestellt), tritt die Zustand-Maschine in einen WSHORT_BURST Zustand **2714** ein, wenn entweder der letzte Teil der Daten genommen wurde (!p2q_trdy) oder der PCI-Start-Impuls empfangen ist. In jedem Fall müssen nur zwei dwords an Daten zu dem PCI-Bus hingeschrieben werden. Ansonsten verbleibt die Zustand-Maschine in dem WDATA2 Zustand **2710**. Wenn die Zustand-Maschine in den WSHORT_BURST Zustand **2714** eintritt, verbleibt das QPIF frame_signal aufgestellt, falls die Transaktion in die nächste Warteschlange überlaufen kann, und ein neuer Schreib-Befehl wird nicht benötigt.

[0224] In dem WSHORT_BURST Zustand **2714** präpariert sich die Master-Zustand-Maschine, um die abschließenden zwei dwords Daten zu dem PCI-Bus zu schreiben. Falls das p2q_ack Signal weggenommen ist oder der Zyklus erneut versucht wird oder durch die PCI-Schnittstelle ausgesondert wird, beendet die Zustand-Maschine die Transaktion und kehrt zu dem IDLE Zustand **2700** zurück. Wenn das PCI-Kenntnis-Signal verschwindet oder der Zyklus ausgesondert wird, stellt die Master-Zustand-Maschine das Stepback-Signal auf, um anzuzeigen, dass der PMWQ-Out-Pointer zurück zu dem vorherigen dword schrittmäßig geführt werden sollte. Wenn die Transaktion erneut durch die PCI-Schnittstelle versucht wird, wird der Out-Pointer zurück nur dann schrittmäßig geführt, falls die Target-Vorrichtung nicht den letzten Teil an Daten nahm (p2q_trdy ist aufgestellt). Wenn die Transaktion nicht beendet ist und sie in den nächsten PMWQ-Puffer überlaufen kann (overflow_next_cueue ist aufgestellt) und ein neuer Schreib-Befehl nicht benötigt wird, behält die Master-Zustand-Maschine das QPIF-Frame-Signal, das aufgestellt ist, bei, und tritt dann in einen WCOMPLETE Zustand **2716** ein, falls die Target-Vorrichtung den letzten Teil an Daten genommen hat oder in dem WSHORT_BURST Zustand **2714** hat, oder verbleibt ansonsten in dem WSHORT-BURST Zustand **2714**. Falls die Transaktion nicht in die nächste Warteschlange überlaufen kann oder ein neuer Schreib-Befehl benötigt wird, nimmt die Zustand-Maschine das Frame-Signal weg, um das Ende der QPIF-Transaktion anzuzeigen, und tritt dann in den WCOMPLETE Zustand **2716** ein, falls der letzte Teil von Daten durch die Target-Vorrichtung genommen wurde, oder verbleibt in dem WSHORT_BURST Zustand **2714** ansonsten.

[0225] In dem WCOMPLETE Zustand **2716** beendet die Master-Zustand-Maschine die gepostete Speicher-Schreib-Transaktion. Die Zustand-Maschine tritt in den IDLE Zustand **2700** ein, wenn die Transaktion erneut versucht oder durch die PCI-Schnittstelle ausgesondert ist. Falls die Transaktion erneut versucht wird, wird der PMWQ-Out-Pointer nur dann erhöht, falls die Target-Vorrichtung den letzten Teil an Daten nahm. Falls die Transaktion in die nächste Warteschlange überlaufen kann, wird ein neuer Schreib-Befehl nicht benötigt, und die Transaktion befindet sich nicht in der letzten Daten-Phase, die Master-Zustand-Maschine erhöht den Warteschlangen-Auswahl-Zähler und kehrt zu dem WDATA1 Zustand **2706** zurück, um die Transaktion von der Überlauf-Warteschlange fortzuführen, solange wie die Target-Vorrichtung den letzen Teil an Daten nahm. Falls die Target-Vorrichtung nicht den letzten Teil an Daten nahm, verbleibt die Master-Zustand-Maschine in dem WCOMPLETE Zustand **2716**.

[0226] Falls die Transaktion nicht in den nächsten PMWQ-Puffer überlaufen wird, beendet die Master-Zustand-Maschine die Transaktion und kehrt zu dem IDLE Zustand **2700** zurück, falls das Target den letzten Teil an Daten nahm. Ansonsten verbleibt die Zustand-Maschine in dem WCOMPLETE Zustand **2716**, bis eines der beendenden Ereignisse, diskutiert vorstehend, auftritt.

[0227] Wie die Fig. 86 zeigt, steuert die Slave-Zustand-Maschine die Operation der QPIF, wenn die QPIF in dem Slave-Mode arbeitet. In dem Slave-Mode empfängt die QPIF gepostete Schreib-Transaktionen und verzögerte Anforderungs-Transaktionen von Vorrichtungen auf dem PCI-Bus und führt die Transaktionen weiter zu dem Target-Bus über das Kabel. Die folgende Tabelle stellt die Ereignisse dar, die Zustand-Übergänge in der Slave-Zustand-Maschine verursachen.

SLAVE STATE MACHINE		
CURRENT STATE	EVENT	NEXT STATE
SLAVE_IDLE	A: p2q_qcyc && p2q_dac_flag && !p2q_perr B: p2q_qcyc && !p2q_dac_flag && pmw_request && !p2q_perr &&(!tc_pmw_full && !dcq_locked && !lock_state[1]) C: p2q_qcyc && !p2q_dac_flag &&!pmw_request && !p2q_perr &&(mem_read_line | mem_read_mul) && (dcq_hit &&!dcq_no_data &&!lock_state[1]) D: p2q_qcyc &&!p2q_dac_flag &&!pmw_request && !p2q_perr &&!(mem_read_line | mem_read_mul) E: [p2q_qcyc &&!p2q_dac_flag && pmw_request && !p2q_perr &&!(!tc_pmw_full&&!dcq_locked&&!lock_state[1])] | [p2q_qcyc&&p2q_dac_flag&&p2q_perr] | [p2q_qcyc &&!p2q_dac_flag&&!pmw_request&& (p2q_perr) | ((mem_read_line | mem_read_mul)&& !(dcq_hit&&!dcq_no_data &&!lock_state[1])] | otherwise	SLAVE_DAC PMW1 STEP_AHEAD SECOND_CHECK SLAVE_IDLE
SLAVE_DAC	F: p2q_qcyc&&pmw_request&& !p2q_perr && (!tc_pmw_full &&!dcq_locked &&!lock_state[1]) G: p2q_qcyc &&!pmw_request && !p2q_perr && (mem_read_line | mem_read_mul) && (dcq_hit && !dcq_no_data && ! lock_state[1]) H: p2q_qcyc&&!pmw_request&& !p2q_perr && !(mem_read_line | mem_read_mul) I: otherwise	PMW1 STEP_AHEAD SECOND_CHECK SLAVE_IDLE
SECOND_CHECK	J: !io_write && !config_write && !p2q_perr && (dcq_hit && !dcq_no_data && !lock_state[1] && dwr_check_ok)_ K: otherwise	STEP_AHEAD SLAVE_IDLE
STEP_AHEAD	L: dcq_no_data M: otherwise	HIT_DCQ_FINAL HIT_DCQ
HIT_DCQ	N: !p2q_qcyc O: p2q_qcyc&&(dcq_no_data&& !p2q_irdy | (pmw_counter[2] &&pmw_counter[1] && pmw_counter[0] && read_disconnect_for_stream) P: otherwise	SLAVE_IDLE HIT_DCQ_FINAL HIT_DCQ
HIT_DCQ_FINAL	Q: !p2q_qcyc | !p2q_irdy R: otherwise	SLAVE_IDLE HIT_DCQ_FINAL
PMW1	S: !p2q_qcyc , T: otherwise	SLAVE_IDLE , PMW1

Slave-Zustand-Übergänge

[0228] Bei einem Reset tritt die Slave-Zustand-Maschine in einen IDLE Zustand **2720** ein, in dem die QPIF auf eine Transaktion wartet, die initiiert werden sollte, und zwar durch eine Vorrichtung auf dem PCI-Bus. Falls eine Transaktion, initiiert auf dem Bus, nicht die QPIF als Ziel trifft (q2p_qcyc ist nicht aufgestellt), fährt die Slave-Zustand-Maschine in dem IDLE Zustand **2720** fort. Wenn eine Transaktion auf dem PCI-Bus nicht die QPIF als Ziel trifft, tritt die Slave-Zustand-Maschine in einen SLAVE_DAC-Dual-Adressen-Zyklus-Zustand **2722** ein, falls das p2q_dac_flag aufgestellt ist und ein Adressen-Paritäts-Fehler nicht aufgetreten ist (p2q_perr ist niedrig). Falls die Transaktion nicht ein Dual-Adressen-Zyklus ist und eine gepostete Speicher-Schreib-Anforderung ist, und falls ein Paritäts-Fehler nicht in der Adressen-Phase aufgetreten ist, lädt die Slave-Zustand-Maschine die Schreib-Zähler (d. h. stellt load_write_counter auf) und bestimmt, ob sie die Transaktion akzeptieren kann. Falls die PMWQ in dem anderen Brücken-Chip voll ist (tc_dc_full ist durch den DC-Transaktions-Zähler aufgestellt) oder die DCQ verriegelt ist (dcq_locked ist aufgestellt) oder sich die QPIF-Verriegelungs-Logik in dem unlocked-but-retry Zustand befindet (lock_state[1] entspricht „1"), dann beendet die Slave-Zustand-Maschine die Transaktion durch Aufstellen eines asynchronen Retry-Signals (early_retry), das zu der PCI-Schnittstelle als q2pif_retry zugeführt wird, und verbleibt in dem IDLE Zustand **2720**. Falls die QPIF die Transaktion annehmen kann, initiiert die Slave-Zustand-Maschine die gepostete Speicher-Schreib-Nachricht auf dem Kabel und tritt in einen PMW1 Zustand **2724** ein, in dem die Transaktion weiter zu dem Kabel geführt wird.

[0229] Falls die Transaktion nicht ein Dual-Adressen-Zyklus oder eine gepostete Speicher-Schreib-Anforderung ist, lädt die Slave-Zustand-Maschine den dword Zähler (stellt load_write_counter auf), und, falls nicht ein Paritäts-Fehler aufgetreten ist, analysiert sie die verzögerte Anforderungs-Transaktion. Falls die Transaktion eine MRL- oder eine MRM-Transaktion ist und die QPIF-Verriegelungs-Logik sich nicht in dem unlocked-but-retry Zustand befindet, stellt die Slave-Zustand-Maschine das QPIF-Prüf-Zyklus-Signal (q2pif_check_cyc) auf, was die DCQ instruiert, die verriegelte Anforderung zu den verzögerten Abschluss-Nachrichten in den DCQ-Puffern zu vergleichen. Falls die Anforderung einen DCQ-Puffer trifft, der nicht leer ist (dcq_hit und !dcq_no_data), tritt die Slave-Zustand-Maschine in einen STEP_AHEAD Zustand **2726** ein, in dem die QPIF beginnt, die angeforderten Informationen zu dem PCI-Bus zuzuführen. Falls die MRL- oder MRM-Anforderung alle die DCQ-Daten-Puffer verfehlt (!dcq_hit), ist die DCQ nicht voll (!dcq_full), die verzögerte Anforderungs-Warteschlange in dem anderen Brücken-Chip ist nicht voll (!tc_dr_full) und die DCQ und QPIF sind nicht verriegelt (!dcq_locked und !lock_state[1]) stellt die Slave-Zustand-Maschine das q2pif_retry Signal auf, führt die Anforderung weiter zu dem Kabel, und verbleibt in dem IDLE Zustand **2720**. Falls die Anforderung die DCQ verfehlt und die Anforderung nicht entlang des Kabels geschickt werden kann, versucht die QPIF einfach erneut die anfordernde Vorrichtung und verbleibt in dem IDLE Zustand **2720**.

[0230] Falls die verzögerte Anforderung nicht eine MRL- oder MRM-Transaktion ist, wird ein zweiter Takt-Zyklus benötigt, um die Anforderung zu prüfen, da die Daten- oder Byte-Freigaben mit den Inhalten der DCQ-Puffer verglichen werden müssen, so dass die Slave-Zustand-Maschine in einen SECOND_CHECK Zustand **2728** eintritt. Falls ein Paritäts-Fehler auftritt oder falls sich die Verriegelungs-Logik in dem unlocked-but-retry Zustand befindet, versucht die Zustand-Maschine erneut die anfordernde Vorrichtung und verbleibt in dem IDLE Zustand **2720**.

[0231] In dem SLAVE_DAC Zustand **2722** empfängt die Slave-Zustand-Maschine die zweite Hälfte der Adressen-Phasen-Informationen. Falls die anfordernde Vorrichtung nicht target-mäßig die QPIF getroffen hat, ignoriert die Slave-Zustand-Maschine die Transaktion und verbleibt in dem IDLE Zustand **2720**. Wenn die QPIF die Target-Vorrichtung ist, sind die Zustand-Übergangs-Ereignisse dieselben wie solche in dem IDLE Zustand **2720**. Genauergesagt lädt, falls die Transaktion eine gepostete Speicher-Schreib-Anforderung ist und ein Paritäts-Fehler nicht aufgetreten ist, die Slave-Zustand-Maschine die Schreib-Zähler und bestimmt, ob sie die Transaktion annehmen kann. Falls die PMWQ in dem anderen Brücken-Chip voll ist (tc_pmw_full ist aufgestellt), wird die DCQ verriegelt, oder die QPIF-Verriegelungs-Logik befindet sich in dem unlocked-but-retry Zustand, die Slave-Zustand-Maschine versucht erneut die anfordernde Vorrichtung und kehrt zu dem IDLE Zustand **2720** zurück. Falls die QPIF die Transaktion annehmen kann, initiiert die Slave-Zustand-Maschine die gepostete Speicher-Schreib-Nachricht auf dem Kabel und tritt in den PMW1 Zustand **2724** ein.

[0232] Falls die Transaktion nicht eine gepostete Speicher-Schreib-Anforderung ist, lädt die Slave-Zustand-Maschine den dword Zähler, und, falls kein Paritäts-Fehler aufgetreten ist, analysiert sie die verzögerte Anforderungs-Transaktion. Falls die Transaktion eine MRL- oder MRM-Transaktion ist und sich die QPIF-Verriegelungs-Logik nicht in dem unlocked-but-retry Zustand befindet, stellt die Slave-Zustand-Maschine das QPIF-Prüf-Zyklus-Signal auf. Falls die Anforderung einen DCQ-Puffer trifft, der nicht leer ist, tritt die Slave-Zustand-Maschine in den STEP_ADHEAD Zustand **2726** ein. Falls die MRL- oder MRM-Anforderung alle die DCQ-Daten-Puffer verfehlt, ist die DCQ nicht voll, die verzögerte Anforderungs-Warteschlange in dem anderen Brücken-Chip ist nicht voll (tc_dr_full ist nicht aufgestellt) und die DCQ und die QPIF sind nicht verriegelt, stellt die Slave-Zustand-Maschine stellt das q2pif_retry Signal auf, führt die Anforderung weiter entlang des Kabels und kehrt zu dem IDLE Zustand **2720** zurück. Falls die Anforderung die DCQ verfehlt und die Anforderung nicht entlang des Kabels geschickt werden kann, versucht die QPIF einfach erneut die anfordernde Vorrichtung und kehrt zu dem IDLE Zustand **2720** zurück.

[0233] Falls die verzögerte Anforderung nicht eine MRL- oder MRM-Transaktion ist, wird ein zweiter Takt-Zyklus benötigt, um die Anforderung zu prüfen, da die Daten- oder Byte-Freigaben mit den Inhalten der DCQ-Puffer verglichen werden müssen, so dass die Slave-Zustand-Maschine in den SECOND_CHECK Zustand **2728** eintritt. Falls ein Parität-Fehler auftritt oder falls sich die Verriegelungs-Logik in dem unlocked-but-retry Zustand befindet, versucht die Zustand-Maschine erneut die anfordernde Vorrichtung und kehrt zu dem IDLE Zustand **2720** zurück.

[0234] In dem PMW1 Zustand **2724** führt die Slave-Zustand-Maschine eine gepostete Speicher-Schreib-Transaktion über das Kabel zu der Target-Vorrichtung weiter. Wenn die Zustand-Maschine zuerst in den PMW1 Zustand **2724** eintritt, nimmt sie das load_write_counter Signal weg. Falls der dword Zähler anzeigt, dass die gepostete Speicher-Schreib-Transaktion die letzte Cache-Zeile ist (pmw_counter[5:3] entspricht „111") und die PMWQ in der anderen Brücke voll ist (tc_pmw_full) und das Schreib-Überlauf-Merkmal

gesperrt ist (!cfg2q_write_overflow), oder falls das write_page_disconnect Signal aufgestellt ist, da die Transaktion eine 4 K Seiten-Grenze erreicht hat, oder falls die DCQ das dcq_disconnect_for_stream Signal aufgestellt hat und das Schreib-Unterbrechungs-Merkmal nicht gesperrt ist (!cfg2q_wr_discnt_disable), stellt die Slave-Zustand-Maschine das slave_lastline Signal auf, das anzeigt, dass die momentane Cache-Zeile die letzte sein wird, die übertragen werden soll. Die Slave-Zustand-Maschine verbleibt dann in dem PMW1 Zustand **2724**, bis das p2q_qcyc Signal weggenommen ist, was anzeigt, dass die Transaktion auf dem PCI-Bus abgeschlossen wurde. Nach Verlassen des PMW1 Zustands **2724**, tritt die Slave-Zustand-Maschine wieder in den IDLE Zustand **2720** ein.

[0235] In dem SECOND_CHECK Zustand **2728** hat die Slave-Zustand-Maschine die DCQ die zweite Phase der Anforderungs-Informationen zu den verzögerten Abschluss-Informationen in den DCQ-Puffern vergleichen lassen. Falls die Transaktion nicht eine verzögerte Schreib-Anforderung ist (!io_write und !config_write) oder dabei ein Paritäts-Fehler vorhanden ist (!p2q_perr) und falls die DCQ nicht verriegelt ist und das dwr_check_ok Signal aufgestellt ist, stellt die Slave-Zustand-Maschine das q2pif_check_cyc auf. Das dwr_check_ok Signal wird entweder dann aufgestellt, wenn die Transaktion nicht eine verzögerte Schreib-Anforderung ist oder wenn sie eine verzögerte Schreib-Anforderung ist und ein p2q_irdy Warte Zustand nicht eingesetzt worden ist. Falls die Anforderung einen der DCQ-Puffer trifft und der Puffer nicht leer ist, tritt die Slave-Zustand-Maschine in den STEP_AHEAD Zustand **2726** ein. Falls die Anforderung alle der DCQ-Puffer verfehlt, allerdings die QPIF die Nachricht entlang des Kabels schicken kann, versucht die Slave-Zustand-Maschine die anfordernde Vorrichtung erneut, führt die Transaktion entlang des Kabels weiter und tritt erneut in den IDLE Zustand **2720** ein. Ansonsten wird, falls die Anforderung alle der DCQ-Puffer verfehlte, und die QPIF nicht die Transaktion entlang des Kabels schicken konnte, oder falls ein Paritäts-Fehler an einer verzögerten Schreib-Anforderung auftritt, die Zustand-Maschine erneut die anfordernde Vorrichtung versuchen und wieder in den IDLE Zustand **2720** eintreten.

[0236] In dem STEP_AHEAD Zustand **2726** erhöht die Slave-Zustand-Maschine den DCQ-Ausgangs-Hinweiszeiger zu dem nächsten dword. Dieser Zustand ist notwendig, unmittelbar nachdem ein DCQ-Puffer getroffen wird, da die PCI-Schnittstelle das erste dword von Daten ohne Zugreifen auf das !p2q_trdy Signal verriegelt. Von dem STEP_AHEAD Zustand **2726** tritt die Zustands_Maschine in einen HIT_DCQ Zustand **2730** ein, in dem Daten von dem geeigneten DCQ-Puffer zu der anfordernden Vorrichtung geliefert werden, falls das letzte dword von Daten nicht genommen worden ist. Ansonsten tritt die Zustand-Maschine in einen HIT_DCQ_FINAL Zustand **2732** ein, in dem die anfordernde Vorrichtung erneut versucht wird, da der DCQ-Puffer keine weiteren Daten enthält.

[0237] Von dem HIT_DCQ Zustand **2730** beendet, wenn die verzögerte Anforderungs-Transaktion auf dem PCI-Bus endet, bevor sie in der QPIF endet (d. h. p2q_qcyc wird weggenommen), die Zustand-Maschine die Transaktion in der QPIF und stellt das Stepback-Signal auf, das anzeigt, dass der DCQ Out-Pointer verringert werden sollte, da der letzte Teil von Daten nicht durch die anfordernde Vorrichtung genommen wurde. Die Zustand-Maschine tritt dann erneut in den IDLE Zustand **2720** ein. Falls der DCQ-Puffer „Out-Of-Data" läuft, während die anfordernde Vorrichtung fortfährt, ihn anzufordern (dcq_no_data und !p2q_irdy), oder falls der pmw_counter anzeigt, dass das letzte dwora erreicht worden ist und das read_disconnect_for_stream Signal aufgestellt worden ist, versucht die Slave-Zustand-Maschine erneut die anfordernde Vorrichtung und tritt in den HIT_DCQ_FINAL Zustand **2732** ein. Falls die Transaktion endet, um eine Datenfolge einzurichten, wird das Stepback-Signal aufgestellt und der Ausgangs-Hinweiszeiger des geeigneten DCQ-Puffers wird erniedrigt. In irgendeiner anderen Situation fährt die Slave-Zustand-Maschine fort, Daten in dem HIT_DCQ_FINAL Zustand **2732** vorzusehen.

[0238] In dem HIT_DCQ_FINAL Zustand **2732** besitzt die Slave-Zustand-Maschine ein dword an Daten übrig, um sie zu übertragen. Falls der PCI-Bus die Transaktion beendet, bevor die anfordernde Vorrichtung den letzten Teil von Daten nimmt (d. h. p2q_qcyc wird weggenommen), stellt die Slave-Zustand-Maschine das Stepback-Signal auf und kehrt zu dem IDLE Zustand **2720** zurück. Falls das p2q_qcyc Signal aufgestellt verbleibt und die anfordernde Vorrichtung nicht einen Initiator-Warte-Zustand aufgestellt hat (!p2q_irdy), wird die anfordernde Vorrichtung erneut versucht, da der letzte Teil an Daten genommen worden ist. Die Zustand-Maschine tritt dann erneut in den IDLE Zustand **2720** ein. Ansonsten verbleibt die Slave-Zustand-Maschine in dem HIT_DCQ_FINAL Zustand **2732**.

[0239] Wie die Fig. 87 zeigt, ist der Kabel-Nachrichten-Generator eine Zustand-Maschine, die Kabel-Nachrichten von Transaktions-Informationen erzeugt, erhalten von der Master- und der Slave-Zustand-Maschine. Zusätzlich zu einem IDLE Zustand **2740** umfasst der Nachrichten-Generator auch einen Dual-Adressen-Zyklus-(CABLE_DAC) Zustand **2742**, einen Master-Daten-Phasen-(MASTER_DPHASE) Zustand **2744** und ei-

nen Slave-Daten-Phasen-(SLAVE_DPHASE) Zustand **2746**. Die folgende Tabelle stellt die Ereignisse dar, die Zustand-Übergänge in dem Kabel-Nachrichten-Generator zu erzeugen.

CABLE MESSAGE GENERATOR		
CURRENT STATE	**EVENT**	**NEXT STATE**
CABLE_IDLE	A: (send_message && q2pif_dac) | ((dcq_prefetch_mul | dcq_prefetch_line) && dcq_prefetch_dac) B: (send_message && !q2pif_dac) | ((dcq_prefetch_mul | dcq_prefetch_line) && !dcq_prefetch_dac) | (dcq_stream_connect && !(|drq_valid[3:0])) && (dcq_stream_connect | !p2q_ack | dcq_prefetch_line | dcq_prefetch_mul) C: (send_message && !q2pif_dac) | ((dcq_prefetch_mul | dcq_prefetch_line) && !dcq_prefetch_dac) | (dcq_stream_connect && !(|drq_valid[3:0])) && !dcq_stream_connect && !(!p2q_ack | dcq_prefetch_mul | dcq_prefetch_line) D: otherwise	CABLE_DAC SLAVE_DPHASE MASTER_DPHASE CABLE_IDLE
CABLE_DAC	E: !p2q_ack | dcq_prefetch_mul | dcq_prefetch_line F: otherwise	SLAVE_DPHASE MASTER_DPHASE
MASTER_DPHASE	G: send_message &&q2pif_dac H: send_message && !q2pif_dac I: !send_message &&(early_last_master_data && !p2q_trdy | master_abort_cable) J: otherwise	CABLE_DAC SLAVE_DPHASE CABLE_IDLE MASTER_DPHASE
SLAVE_DPHASE	K: [!(drq_stream_connect &&!(|drq_valid[3:0]&&p2q_qcyc)] &&[(dly_read_request | dly_single_write_request | dcq_prefetch_mul | dcq_prefetch_line)] | L: early_last_slave_data | dcq_stream_connect &&!(|drq_valid[3:0]&&p2q_qcyc and otherwise	CABLE_IDLE SLAVE_DPHASE

Kabel-Nachrichten-Generator-Zustand-Übergänge

[0240] Bei einem Reset tritt der Kabel-Nachrichten-Generator in den IDLE Zustand **2740** ein, in dem er auf Transaktions-Informationen wartet, damit sie von der Master- oder Slave-Zustand-Maschine ankommen. Von dem IDLE Zustand **2740** entspricht, falls der Kabel-Nachrichten-Generator ein Prefetch-Mehrfach-Signal (dcq_prefetch_mul) oder ein Prefetch-Leitungs-Signal (dcq_prefetch_line) empfängt, das Kabel-Adressen-Signal (early_cad[31:2]) dem Prefetch-Adressen-Signal (dcq_prefetch_addr[31:2]). Ansonsten nimmt das early_cad[31:2] Signal den Wert des QPIF-Adressen-Signals (q2pif_addr[31:2]) an. Wenn die Kabel-Nachricht durch die Master-Zustand-Maschine initiiert wird, ist die Nachricht eine verzögerte Abschluss-Nachricht, so dass der Befehl-Code (early_ccbe[3:0]) „1001" entspricht. Wenn die Kabel-Nachricht durch die Slave-Zustand-Maschine initiiert wird, nimmt der Befehl-Code den Wert des message_cmd[3:0] Signals an, wie dies vorstehend diskutiert ist.

[0241] Falls das send_message Signal aufgestellt ist, anzeigend, dass entweder die Master-Zustand-Maschine oder die Slave-Zustand-Maschine eine Nachricht initiiert hat, und die entsprechende Transaktion nicht ein Dual-Adressen-Zyklus ist, oder falls der Kabel-Nachrichten-Generator eine Prefetch-Anforderung empfängt, die nicht ein Dual-Adressen-Zyklus ist, oder falls der Kabel-Nachrichten-Generator ein Datenfolgen-Verbindungs-Signal empfängt und keine verzögerten Anforderungen von der CPU in der ausgangsseitigen DRQ anhängig sind, stellt der Kabel-Nachrichten-Generator ein sent_pmw Signal auf, das anzeigt, dass eine gepostete Speicher-Schreib-Anforderung von dem PCI-Bus entlang des Kabels geschickt werden wird. Das sent_pmw Signal wird nicht aufgestellt, falls eine Datenfolge durch die DCQ eingerichtet worden ist. Der Kabel-Nachrichten-Generator stellt ein sent_dr Signal auf, wenn eine Lese-Anforderung oder eine verzögerte Schreib-Anforderung von der Slave-Zustand-Maschine empfangen ist oder ein Prefetch-Signal empfangen ist, und wenn eine Datenfolge nicht durch die DCQ eingerichtet worden ist.

[0242] Falls die DCQ eine Datenfolge eingerichtet hat (dcq_stream_connect ist aufgestellt), nimmt die Pufferzahl für das Kabel-Signal (early_cbuff[2:0]) den Wert des DCQ-Datenfolge-Puffers an (dcq_stream_buff[2:0]), der Kabel-Befehl-Code (early_ccbe[3:0]) wird gleich zu „1000" eingestellt, und der Kabel-Nachrichten-Generator tritt in den SLAVE_DPHASE Zustand **2746** ein. Ansonsten nimmt, falls sich die QPIF in dem Slave-Mode

befindet und der Kabel-Nachrichten-Generator entweder ein Prefetch-Mehrfach- oder ein Prefetch-Leitungs-Signal empfängt, das Kabel-Puffer-Signal den Wert der DCQ-Puffer-Zahl an (dcq_buff[2:0]) und der Kabel-Nachrichten-Generator tritt in den SLAVE_DPHASE Zustand **2746** ein. Ansonsten arbeitet die QPIF in dem Master-Mode und der Kabel-Nachrichten-Generator tritt in den MASTER_DPHASE Zustand **2744** ein.

[0243] Wenn der Kabel-Nachrichten-Generator das send_message Signal und eine Transaktion, die ein Dual-Adressen-Zyklus ist, empfängt, oder wenn er eine Prefetch-Anforderung empfängt, die ein Dual-Adressen-Zyklus ist, tritt der Nachrichten-Generator in den CABLE_DAC Zustand **2742** ein. Für ein Prefetch-Signal wird das Kabel-Adressen-Signal gleich zu den oberen zweiunddreißig Bits des dcq_prefetch_addr[63:0] Signals eingestellt; ansonsten entspricht die Kabel-Adresse den oberen zweiunddreißig Bits des q2pif_addr[63:0] Signals. Auch entspricht, falls der Kabel-Nachrichten-Generator die Transaktion von der Slave-Zustand-Maschine empfängt, die Kabel-Puffer-Zahl der DCQ-Puffer-Zahl; ansonsten entspricht die Kabel-Puffer-Zahl der DRQ-Puffer-Zahl (keine Abschluss-Nachrichten werden für gepostete Speicher-Schreib-Transaktionen erzeugt).

[0244] In dem CABLE_DAC Zustand **2742** erzeugt der Kabel-Nachrichten-Decodierer die zweite Hälfte der Adressen-Phasen-Informationen. Wie in dem IDLE Zustand **2740** nimmt das Kabel-Adressen-Signal den Wert der Prefetch-Adresse an, wenn die empfangene Transaktion eine Prefetch-Leitungs- oder Prefetch-Mehrfach-Anforderung ist, und nimmt den Wert von q2pif_addr[31:2] ansonsten an. Das sent_pmw Signal wird dann aufgestellt, wenn der Nachrichten-Generator eine gepostete Speicher-Schreib-Transaktion von der Slave-Zustand-Maschine empfängt, und das sent_dr Signal wird dann aufgestellt, wenn sie eine Prefetch-Anforderung oder eine verzögerte Anforderung von der Slave-Zustand-Maschine empfängt. Falls eine Prefetch-Anforderung oder eine Anforderung von der Slave-Zustand-Maschine empfangen wird, tritt der Kabel-Nachrichten-Generator in den SLAVE_DPHASE Zustand **2746** ein. Ansonsten tritt der Nachrichten-Generator in den MASTER_DPHASE Zustand **2744** ein.

[0245] In dem MASTER_DPHASE Zustand **2744** versucht der Kabel-Nachrichten-Generator, eine verzögerte Abschluss-Nachricht entlang des Kabels zu schicken. Allerdings muss, falls die PCI-Schnittstelle den Bus zu einer Vorrichtung auf dem PCI-Bus erteilt, bevor die QPIF eine Kontrolle des Busses erhält, der Kabel-Nachrichten-Generator den MASTER_DPHASE Zustand **2744** verlassen, um die neu empfangene Nachricht zu schicken. Deshalb wird, falls das send_message Signal aufgestellt ist, während sich der Nachrichten-Generator in dem MASTER_DPHASE Zustand **2744** befindet, das q2c_new_req Signal aufgestellt, um den Start einer neuen Nachricht anzuzeigen. Falls das q2pif_dac_flag aufgestellt ist, ist die neue Transaktion ein Dual-Adressen-Zyklus und der Kabel-Nachrichten-Generator tritt in den CABLE_DAC Zustand **2742** ein. Ansonsten tritt der Nachrichten-Generator in den SLAVE_DPHASE Zustand **2746** ein.

[0246] Falls das send_message Signal nicht aufgestellt ist, schickt der Kabel-Nachrichten-Generator eine verzögerte Abschluss-Nachricht von der Master-Zustand-Maschine aus. Wenn die Master-Zustand-Maschine die letzte Daten-Übertragung mit dem PCI-Bus abgeschlossen hat und die Target-Vorrichtung die Übertragung bestätigt hat (!p2q_trdy), oder wenn der Master die Transaktion auf dem Kabel ausgesondert hat, stellt der Kabel-Nachrichten-Generator ein sent_dc Signal auf, das anzeigt, dass die verzögerte Abschluss-Nachricht entlang des Kabels geschickt worden ist, und tritt erneut in den IDLE Zustand **2740** ein. Ansonsten verbleibt der Nachrichten-Generator in dem MASTER_DPHASE Zustand **2744** und fährt fort, die verzögerte Abschluss-Nachricht zu erzeugen.

[0247] Von dem SLAVE_DPHASE Zustand **2746** sind, solange wie eine Datenfolge mit dem eingangsseitigen Chip eingerichtet wird, keine verzögerten Anforderungen von der CPU in der ausgangsseitigen DRQ anhängig, und die anfordernde Vorrichtung fährt fort, Daten zu der QPIF zu schicken (q2p_qcyc ist aufgestellt), der Kabel-Nachrichten-Generator verbleibt in dem SLAVE_DPHASE Zustand **2746** und fährt fort, die Transaktion von der anfordernden Vorrichtung weiterzuführen. Ansonsten führt, falls der Kabel-Nachrichten-Generator eine verzögerte Anforderung oder eine Prefetch-Anforderung empfängt, der Kabel-Nachrichten-Generator die Anforderung weiter, und, in dem Fall einer verzögerten Schreib-Anforderung, das eine dword an Daten zu der eingangsseitigen Vorrichtung weiter, und tritt dann in den IDLE Zustand **2740** ein. Ansonsten hat der Kabel-Nachrichten-Generator eine gepostete Speicher-Schreib-Anforderung empfangen. In dieser Situation verbleibt der Kabel-Nachrichten-Generator in dem SLAVE_DPHASE Zustand **2746** und fährt fort, die geposteten Speicher-Schreib-Informationen entlang des Kabels weiterzuführen, bis das early_last_slave_data Signal aufgestellt ist, was anzeigt, dass der letzte Teil an Daten durch die Slave-Zustand-Maschine geschickt worden ist. Der Nachrichten-Generator beendet dann die Kabel-Transaktion und tritt erneut in den IDLE Zustand **2740** ein.

[0248] Um die gültige Übertragung von Daten zwischen den zwei Brücken-Chips sicherzustellen, müssen Daten, geschickt über das Kabel **28**, geeignet zu den Takten von den Takt-Generatoren **102** und **122** synchronisiert werden. Der ausgangsseitige Takt-Generator **122** legt seine Takte basierend auf einem eingangsseitigen Takt (der wiederum auf dem PCI-Bus-Takt PCICLK1 basiert) fest, übertragen entlang des Kabels **28** mit den Daten. Als Folge werden eingangsseitige Daten, übertragen zur Ausgangsseite hin, zu den Takten synchronisiert, erzeugt in dem ausgangsseitigen Brücken-Chip **48**. Allerdings ist die Phasen-Verzögerung, zugeordnet zu dem Kabel **28**, zwischen den Haupt-Takten, erzeugt in dem eingangsseitigen Chip **26**, und die Daten, übertragen zurück eingangsseitig von dem ausgangsseitigen Chip **48**, unbekannt. Die Länge des Kabels **28** reicht von 10 bis zu 100 Fuß (falls eine geeignet Schnittstellen-Technologie verwendet wird). Die empfangende Logik in der eingangsseitigen Kabel-Schnittstelle **104** ist effektiv eine asynchrone Grenze in Bezug auf den eingangsseitigen Takt. Demzufolge muss die empfangende Logik die eingangsseitigen-zu-ausgangsseitigen Daten zu dem Takt von dem eingangsseitigen Takt-Generator **102** erneut synchronisieren.

[0249] In Fig. 5 nun ist das Takt-Verteilungs-Schema in dem 2-Chip in der 2-Chip-PCI-PCI-Brücke dargestellt. Transaktionen, die nach vorne zwischen den Brücken-Chips **46** und **48** geführt werden, werden zu mehrfachen, zeit-multiplexierten Nachrichten codiert. Das Format der Nachrichten ist ähnlich zu dem PCI-Transaktions-Format (mit der Ausnahme eines Zeit-Multiplexing) und umfasst eine Adresse und eine oder mehrere Daten-Phasen und modifizierte Handshake-Signale zusätzlich zu den Signalen, die hinzugefügt sind, um eine Puffer-Zahl und spezielle Brücken-Funktions-Befehle anzuzeigen. Jede Kabel-Schnittstelle **104** oder **130** umfasst eine Master-Kabel-Schnittstelle (**192** oder **194**) und eine Slave-Kabel-Schnittstelle (**196** oder **198**). Die Master-Kabel-Schnittstelle **192** oder **194** überträgt Nachrichten weiter zu dem Kabel **28** und die Slave-Kabel-Schnittstelle **196** oder **198** empfängt Nachrichten von dem Kabel **28**.

[0250] Der Takt-Generator **102** oder **122** in jedem Brücken-Chip umfasst zwei sich auf dem Chip befindliche PLLs für eine Takt-Erzeugung. Eine PLL **184** in dem eingangsseitigen Brücken-Chip **26** verriegelt sich auf dem Primär-PCI-Bus-Eingangs-Takt PCICLK1. In dem ausgangsseitigen Brücken-Chip **48** verriegelt sich die PLL **180** auf einem ankommenden Takt PCICLK2 von einem Takt-Puffer **181**.

[0251] In der nachfolgenden Beschreibung bezieht sich ein „1 × Takt" auf einen Takt, der dieselbe Frequenz wie der Takt PCICLK1 besitzt, während sich ein „3 × Takt" auf einen Takt bezieht, der dreimal die Frequenz des Takts PCICLK1 besitzt. Ein 1 × Takt PCLK, erzeugt durch die PLL **184** oder **180** (in dem Brücken-Chip **26** oder **48** jeweils), wird für die PCI-Bus-Schnittstellen-Logik **188** oder **190** für den Brücken-Chip verwendet, und der 3 × Takt PCLK3 wird dazu verwendet, die Kabel-Nachrichten-Erzeugungs-Logik in der Master-Kabel-Schnittstelle **192** oder **194** laufen zu lassen. Die andere PLL **186** oder **182** wird dazu verwendet, sich auf einen Kabel-Eingangs-Takt CABLE_CLK1 (von Eingangsseite) oder einen CABLE_CLK2 (von Ausgangsseite) zu verriegeln, und um einen 1 × Takt CCLK und einen 3 × Takt CCLK3 zu erzeugen, um ankommende Kabel-Daten zu erfassen. Die Taktausgänge der PLL **186** und **182** werden zu der Slave-Kabel-Schnittstelle **196** und **198** jeweils weitergeführt.

[0252] Die PLLs sind in dem Layout so angeordnet, um die 1 × und 3 × Takte so nahe wie möglich auszubalancieren, um die Verschiebung dazwischen zu minimieren.

[0253] Die PLL **184** oder **180** erzeugt ein Phasen-Indikator-Signal PCLKPHI1, das der Master-Kabel-Schnittstelle **192** oder **194** anzeigt, wenn die erste Phase von Daten zu dem Kabel **28** vorhanden sein sollte. Auf der Eingangsseite ist das Signal PCLKPHI1 auf dem PCI-Takt PCICLK1 basierend; auf der Ausgangsseite ist das Signal PCLKPHI1 auf dem PCI-Takt PCICLK2 basierend. Die PLL **186** oder **182** erzeugt ein Phasen-Indikator-Signal CCLKPHI1, basierend auf dem Kabel-Takt CABLE_CLK1 oder CABLE_CLK2, um zu der Slave-Kabel-Schnittstelle **196** oder **198** anzuzeigen, wenn die erste Phase von Daten entlang des Kabels **28** angekommen ist. Der PCI-Takt PCICLK2 für den sekundären PCI-Bus **32** wird außerhalb eines 1 × Takts BUFCLK der PLL **182** in dem ausgangsseitigen Brücken-Chip **48** erzeugt. Der Takt BUFCLK steuert den Takt-Puffer **181** über einen Treiber **179** an. Der Puffer **181** gibt ein separates Taktsignal für jeden der sechs Schlitze auf dem sekundären PCI-Bus **32** ebenso wie den Takt PCICLK2 aus, was zurück zu dem Bus-Eingangs-Takt zu dem ausgangsseitigen Brücken-Chip **48** geführt wird. Indem der Takt PCLK auf dem Takt PCICLK2 von dem Takt-Puffer **181** basierend ist, werden die Takt-Schernen der eingangsseitigen und ausgangsseitigen Chips so gestaltet, um ähnlicher zu erscheinen, da beide auf dem externen Bus-Takt basierend sind.

[0254] Der Kabel-Takt CABLE_CLK1 ist ein 33% Taktzyklus. Die PLL **182** wandelt erst den 33% Taktzyklus zu einem 50% Taktzyklus zur Ausgabe als BUFCLK um.

[0255] Die PCI-Spezifikation, Version 2.1, fordert, dass der PCI-Bus-Takt die folgenden Erfordernisse erfüllen muss: Takt-Zyklus-Zeit größer als oder gleich zu 30 ns; Takt-Hoch-Zeit größer als 11 ns; Takt-Niedrig-Zeit größer als oder gleich zu 11 ns und Taktanstiegsrate zwischen 1 und 4 ns.

[0256] Wenn das Computersystem hochgefahren wird, wird der eingangsseitige Chip zuletzt hochgefahren, die eingangsseitige PLL **184** schickt den Takt CABLE_CLK1 (über die Master-Schnittstelle **122**) nach unten entlang des Kabels **28**, das dann durch die ausgangsseitige PLL **182** und PLL **180** verriegelt wird. Die ausgangsseitige PLL **180** schickt dann den Takt CABLE_CLK2 zurück eingangsseitig, um durch die PLL **186** verriegelt zu werden. Das System ist nicht vollständig betriebsfähig, bis alle vier PLLs eine Verriegelung erhalten haben.

[0257] Falls der eingangsseitige Brücken-Chip **26** hochfährt und der ausgangsseitige Brücken-Chip **48** noch nicht eingeschaltet ist, verhält sich der eingangsseitige Brücken-Chip **26** als eine PCI-PCI-Brücke, wobei nichts mit dem ausgangsseitigen Bus (das Kabel **28**) verbunden ist. Als Folge nimmt der eingangsseitige Brücken-Chip **26** nicht irgendwelche Zyklen an, bis der ausgangsseitige Brücken-Chip **48** hochgefahren bzw. eingeschaltet ist und die ausgangsseitige PLL **186** eine „Verriegelung" von dem Kabel-Takt CABLE_CLK2 erhalten hat.

[0258] Der eingangsseitige Brücken-Chip **26** floatiert alle seiner PCI-Ausgangs-Puffer und Zustand-Maschinen asynchron mit einem Aufstellen des PCI-Reset-Signals PCIRST1_ auf dem primären Bus **24**. Während eines Resets kann die PLL **184** versuchen, eine Verriegelung auf dem PCI-Bus-Takt PCICLK1 zu erhalten. Da die PCI Spezifikation garantiert, dass das Signal PCIRST1_ aktiv für mindestens 100 µs verbleiben wird, nachdem der PCI-Bus-Takt stabil wird, hat die PLL **184** ungefähr 100 µs, um eine Verriegelung zu erhalten.

[0259] Der ausgangsseitige Brücken-Chip **48** setzt alle internen Zustand-Maschinen beim Erfassen des Signals PCIRST1_ für den primären Bus zurück. Daraufhin stellt der ausgangsseitige Brücken-Chip ein Schlitz-spezifisches Reset zu jedem Schlitz auf dem sekundären PCI-Bus **32** ebenso wie ein Reset-Signal PCIRST2_ für einen sekundären PCI-Bus auf.

[0260] Wie Fig. 6 zeigt, umfasst jede PLL einen spannungs-gesteuerten Oszillator (VCO) **200**, der einen Ausgang **201** (den 3 × Takt) zwischen 75 Mhz (für einen 25-Mhz PCI-Bus) und 100 Mhz (für einen 33-Mhz PCI-Bus) erzeugt. Der VCO **200** empfängt einen Referenz-Takt **197**, der der PCI-Bus-Takt ist. Jede PLL besitzt eine Verriegelungs-Erfassungs-Schaltung **205**, die durch ein Verriegelungs-Indikations-Bit anzeigt, dass die PLL-Phase auf deren Referenz genau genug verriegelt ist, um deren vorgesehene Funktion durchzuführen.

[0261] Die Verriegelungs-Anzeige-Bits werden zu einem Status-Register in dem Konfigurations-Raum **105** oder **125** jedes Brücken-Chips geschrieben. Auf der Ausgangsseite wird ein power-good/lock-Status-Bit zu dem eingangsseitigen Brücken-Chip **26** übertragen, um anzuzeigen, dass die Hauptelemente des ausgangsseitigen Brücken-Chips **48** stabil sind (Energie ist stabil) und die ausgangsseitigen PLLs verriegelt sind (Verriegelungs-Anzeige-Bits der zwei PLLs sind aktiv). Das Verriegelungs-Anzeige-Bit wird auch tormäßig mit den EDC-Status-Bits gesteuert, so dass EDC-Fehler nicht als solche berichtet werden, bis die PLLs verriegelt sind. Demzufolge kann das Brücken-Chip-Paar zu einem fehlerfreien-Kommunikations-Zustand ohne eine Software-Intervention gelangen. Das Verriegelungs-Anzeige-Bit liefert auch bestimmte, diagnostische Informationen, die zwischen einem PLL-Verriegelungs-Fehler und anderen Daten-Fehlern unterscheiden können. Die Takt-Erzeugungs-Schaltung umfasst eine Vier-Zustand-Maschine **202**, um einen durch 3 geteilten Takt (1 × Takt) des VCO Ausgangs **201** zu erzeugen. Der 1 × Takt wird zurück zu der PLL an dem Eingang **203** geführt.

[0262] Daten werden entlang des Kabels **28** unter einer 3 × Takt (PCLK3) Rate in drei Zeit-multiplexierten Phasen geführt, um eine 1 × Takt Nachrichten-Übertragungs-Rate zu erzeugen. Wie Fig. 7 zeigt, umfasst die Schaltung in der Master-Kabel-Schnittstelle **192** oder **194** zum Zerlegen und Übertragen der Kabel-Nachricht ein Register **204**, das die abgehende Nachricht unter einer lokalen PCLK-Grenze abtastet. Das Flip-Flop **208** liefert eine zusätzliche Zone für eine Halte-Zeit in der dritten Phase der übertragenden Nachricht durch Halten dieser Phase für eine zusätzliche Hälfte eines PCLK. Da das Ausgangs-Register **212** mit dem 3 × Takt PCLK3 getaktet ist, verringert dies das Erfordernis für eine enge Kontrolle in Bezug auf den Versatz zwischen den 1× und 3 × Takten. Von dem Phasen-Indikations-Signal PCLKPHI1 erzeugt ein Satz von drei Flips-Flops **210** aufeinanderfolgende PHI1, PHI2 und PHI3 Signale, die Phasen 1, 2 und 3 jeweils darstellen, was wiederum einen 60 : 20 Multiplexer **206** steuert. Die drei Phasen von Daten (LMUXMSG[19:0], LMUXMSG[39:20], {LMUXMSG[51:40], EDC[7:0]}) werden aufeinanderfolgend in das Register **212** multiplexiert und über das Kabel **28** angesteuert. Die dritte Phase an Daten umfasst Fehler-Korrektur-Bits EDC[7:0], erzeugt durch einen ECC-Generator **206** (Fig. 17), von den Ausgangs-Bits LMUXMSG[51:0] des Registers **204**. Das Flip-Flop **214**,

getaktet durch PCLK3, empfängt das PHI1 Signal und taktet es als den Kabel-Takt CABLE_CLK1 oder CABLE_CLK2 heraus.

[0263] Da die Master-Kabel-Schnittstelle **192** oder **194** eine 1 ×-zu-3 × Kommunikations-Schnittstelle ist, wird eine EIN-3×-Takt-Latenz hervorgerufen, die zu einer einzelnen 3 × Takt-Phasen-Verschiebung der übertragenden Kabel-Nachricht von dem PCI-Bus-Takt führt, wie dies in Fig. 8 dargestellt ist. In der Periode T0 wird eine Nachricht A dem Eingang des Registers **204** präsentiert und der erste Phasen-Takt-Indikator PCLKPHI1 wird auf hoch gesetzt. Das Signal PHI1 wird auf hoch von einem vorherigen Zyklus gesetzt. In der Periode P1 wird der Kabel-Takt CABLE_LK1 oder CABLE_CLK2 auf hoch in Abhängigkeit des Signals PHI1, das zu hoch übergeht, angesteuert. Der PCLKPHI1 Impuls bewirkt, dass das Signal PHI2 auf hoch in der Periode T1 gepulst wird. Als nächstes wird, in der Periode T2, das Signal PHI3 in Abhängigkeit des Signals PHI2 gepulst. In der Periode T3 wird das Signal PHI1 auf hoch in Abhängigkeit des Signals PHI3, das hoch ist, gepulst. Eine Nachricht A wird auch in das Register **204** an der ansteigenden Flanke des Takts PCLK in der Periode T3 eingeladen. Als nächstes bewirkt, in der Periode T4, das Signal PHI1, dass der Multiplexer **206** die ersten Phasen-Daten A1 zum Einladen in das Register **212** auswählt. Als nächstes werden, in der Periode T5, die zweiten Phasen-Daten A2 ausgewählt und in das Register **212** eingeladen. Dann werden, in der Periode T6, die dritten Phasen-Daten A3 in das Register **212** eingeladen. Dieser Prozess wird für die Nachrichten B, C, D und E in den darauffolgenden Takt-Perioden wiederholt.

[0264] Wie in Fig. 8 dargestellt ist, besitzt der Kabel-Takt CABLE_CLK einen 33% Taktzyklus. Alternativ kann der Kabel-Takt CABLE_CLK so ausgelegt werden, um einen durchschnittlichen Taktzyklus von 50% zu haben, was, zum Beispiel, durch Abschicken des Kabel-Taktes als 33% hoch – 66% niedrig – 66% hoch – 33% niedrig vorgenommen werden kann. In dem man einen durchschnittlichen 50% Taktzyklus hat, könnte dies zu besseren Durchgangscharakteristika in dem Kabel **28** führen.

[0265] Wie Fig. 9 zeigt, stellt ein Slave-Kabel-Schnittstellen-First-in-First-out-Puffer (FIFO) **216** ankommenden Daten von dem Kabel **28** zusammen und überträgt die zusammengestellten Daten zu den Warteschlangen und den PCI Zustand-Maschinen in dem empfangenden Brücken-Chip. Der FIFO **216** ist 4 Eintritte tief, wobei jeder Eintritt in der Lage ist, eine vollständige Kabel-Nachricht zu halten. Die Tiefe des FIFO **216** ermöglicht, dass Kabel-Daten zu dem Takt des lokalen Brücken-Chips ohne Verlieren irgendeiner effektiven Bandbreite in der Kabel-Schnittstelle synchronisiert werden können. Zusätzlich ist, auf der Eingangsseite, der FIFO **216** eine asynchrone Grenze für die Kabel-Daten, die von dem ausgangsseitigen Brücken-Chip **48** ankommen. Der FIFO **216** stellt sicher, dass die Kabel-Daten geeignet in Bezug auf PCLK synchronisiert sind, bevor sie zu dem Rest des Chips ausgegeben werden.

[0266] Die Eintritte des FIFO **216** werden durch einen Eingangs-Hinweiszeiger INPTR[1:0] von einem Eingangs-Hinweiszeiger-Zähler **226** ausgewählt, der durch das Signal CCLK3 getaktet wird, gelöscht wird, wenn ein Signal EN_INCENT niedrig ist, und durch den Phasen-Indikator CCLKPHI1 freigegeben wird. Die negative Flanke des 3 × Takts CCLK3 von der PLL **186** oder **182** wird dazu verwendet, ankommende Daten von dem Kabel **28** zu verriegeln, zuerst in ein 20-Bit Register **218** hinein und dann in ein Register **220** hinein, falls ein Phasen-Ein-Indikations-Signal PHI1_DLY aufgestellt ist, oder in ein Register **222** hinein, falls ein Phasen-2-Indikationssignal PHI2_DLY aufgestellt ist. Die Phase-1-Daten, die Phase-2-Daten und die Phase-3-Daten von den Registern **220**, **222** und **218** jeweils werden in den ausgewählten Eingang des FIFO **216** an der negativen Flanke von CCLK3 ausgewählt, wenn das Phasen-3-Indikations-Signal PHI3_DLY aufgestellt ist. Die vier Sätze von Ausgängen von dem FIFO **216** werden durch einen 240 : 60 Multiplexer **228** empfangen, der durch einen Ausgangs-Hinweiszeiger OUTPTR[1:0] von einem Ausgangs-Hinweiszeiger-Zähler **224**, getaktet durch PCLK und gelöscht dann, wenn ein Signal EN_OUTCNT niedrig ist, ausgewählt wird.

[0267] Wie die Fig. 10 zeigt, laufen die Eingangs-Hinweiszeiger- und Ausgangs-Hinweiszeiger-Zähler **226** und **224** kontinuierlich durch den FIFO **216**, was Daten füllt und leert. Die Zähler **226** und **224** sind in einer solchen Art und Weise versetzt, um gültige Daten in einer Stelle zu garantieren, bevor sie ausgelesen werden. Die Initialisierung der Hinweiszeiger ist unterschiedlich für einen eingangsseitigen Brücken-Chip **26** gegenüber einem ausgangsseitigen Brücken-Chip **48**, aufgrund von Synchronisierungsunsicherheiten.

[0268] Flip-Flops **236** und **238** synchronisieren das Reset-Signal C_CRESET, das synchron zu den Takten in dem Brücken-Chip ist, zu der CLK-Takt-Grenze. Das Signal EN_INCNT wird durch das Flip-Flop **238** erzeugt. Der Eingangs-Hinweiszeiger wird an der ansteigenden Flanke des Takts CCLK3 erhöht, wenn das erste Phasen-Indikations-Signal CCLKPHI1 und das Signal EN_INCNT vorliegen. Der Ausgangs-Hinweiszeiger wird dann an einer späteren, lokalen PCLK-Takt-Grenze PCLK gestartet, wenn garantiert werden kann, dass die Daten in dem FIFO **216** gültig sein werden. Der eingangsseitige und der ausgangsseitige Brücken-Chip müs-

sen das Starten des Ausgangs-Hinweiszeigers unterschiedlich handhaben, da die Phasen-Beziehung des Kabel-Taktes zu dem lokalen Takt nicht für den eingangsseitigen Brücken-Chip **26** bekannt ist, sondern für den ausgangsseitigen Brücken-Chip **48** bekannt ist.

[0269] In dem ausgangsseitigen Brücken-Chip **48** ist die Phasen-Beziehung zwischen dem ankommenden Kabel-Takt CABLE_CLK1 und dem sekundären PCI-Bus-Takt PCICLK2 bekannt, da der PCI-Takt PCICLK2 von dem Kabeltakt erzeugt wird. Als Folge existiert keine Synchronisations-Strafe für den Ausgangs-Hinweiszeiger OUTPTR[1:0] in dem ausgangsseitigen Brücken-Chip **48**, und der Ausgangs-Hinweiszeiger kann den Eingangs-Hinweiszeiger INPTR[1:0] so nahe wie möglich nachführen. Ein Flip-Flop **230**, das an der negativen Flanke des Takts PCLK getaktet wird, wird dazu verwendet, irgendwelche Taktverschiebungsprobleme zwischen dem Takt CCLK, erzeugt durch die PLL **182**, und dem Takt PCLK, erzeugt durch die PLL **180**, zu vermeiden. Obwohl diese zwei Takte identische Frequenzen haben und in Phase miteinander sein sollten, ist dabei eine unbekannte Verschiebung zwischen den zwei Takten vorhanden, da sie von zwei unterschiedlichen PLLs erzeugt werden. Auf der Ausgangsseite ist das Signal EN_OUTCNT das Signal EN_INCNT, verriegelt auf der negativen Flanke des Signals PCLK durch das Flip-Flop **230**. Ein Multiplexer **234** wählt den Ausgang des Flip-Flops **230** aus, da das Signal UPSTREAM_CHIP niedrig ist.

[0270] In dem eingangsseitigen Brücken-Chip **26** wird die Kabel-Schnittstelle bzw. das Kabel-Interface als vollständig asynchron behandelt. Die Phasen-Unsicherheit erfolgt aufgrund der unbekannten Phasenverschiebung des Kabels **28** selbst. Das Auslegen in Bezug auf diese Unsicherheit führt zu einer vollständigen Freiheit in Bezug auf die Länge des Kabels **28**. Dasjenige, was bekannt ist, ist das, dass die Takte in den eingangsseitigen- und ausgangsseitigen Brücken-Chips dieselbe Frequenz haben, da sie beide deren Ursprung in dem eingangsseitigen PCI-Bus-Takt PCICLK1 haben. In dem eingangsseitigen Brücken-Chip **26** ist das Signal EN_OUTCNT das Signal EN_INCNT verriegelt auf der postiven Flanke des Takts PCLK durch ein Flip-Flop **232**. Der Multiplexer **234** wählt den Ausgang des Flip-Flops **232** aus, da das Signal UPSTREAM_CHIP hoch ist. Das Flip-Flop **232** garantiert, dass gerade für das „Lineup" im schlechtesten Fall des Kabel-Taktes CABLE_CLK2 und des lokalen PCI-Taktes PCLK (eine vollständige PCLK Periode-Phasen-Verschiebung) gültige Daten in dem FIFO **216** vorhanden sind, bevor die Daten zu dem Rest des Chips übertragen werden.

[0271] Wie Fig. 11 zeigt, werden die Kabel-Daten durch die Slave-Kabel-Schnittstelle **196** oder **198** als Drei-Phasen-, in der Zeit multiplexierte, Signale A1, A2 und A3; B1, B2 und B3; C1, C2 und C3; usw., empfangen. Eine vorherige Transaktion wird in den Perioden T0, T1 und T2 abgeschlossen. Beginnend in der Periode T3 werden die ersten Phasen-Daten A1 dem Register **218** präsentiert und der erste Phasen-Indikator CCLKPHI1 wird auf hoch gepulst. An der abfallenden Flanke von CCLK3 in der Periode T3 werden die Daten A1 in das Register **218** eingeladen und das Indikations-Signal PHI_DLY der lokalen Phase 1 wird auf hoch gepulst. In der Periode T4 werden, an der abfallenden Flanke des Takts, die Daten A1 der Phase 1 in das Register **220** eingeladen, die Daten A2 der Phase 2 werden in das Register **218** eingeladen und das Indikations-Signals PHI2_DLY der Phase 2 wird auf hoch gepulst. In der Periode T5 werden, an der abfallenden Flanke von CCLK3, die Daten der Phase 2 in das Register **222** eingeladen, die Daten A3 der Phase 3 werden in das Register **218** eingeladen und das Indikations-Signal PHI3_DLY der Phase 3 wird auf hoch gepulst. In der Periode T6 werden die Inhalte der Register **220**, **222** und **218** in den ausgewählten Eingang des FIFO **216** an der folgenden Flanke CCLK3 eingeladen. Auch werden in der Periode T6 die Daten B1 dem Register **218** zusammen mit dem Indikations-Signal CCLKPHI1 präsentiert. Nachrichten B und C werden in das FIFO **216** in derselben Art und Weise wie eine Nachricht A in darauffolgenden Perioden eingeladen.

[0272] Wie Fig. 12 zeigt, beginnt der Eingangs-Hinweiszeiger INPTR[1:0] bei dem Wert 0 in der Periode T0 an der ansteigenden Flanke des Takts CCLK3. Auch wird, in einer Periode T0, eine Nachricht A in das FIFO0 an der abfallenden Flanke des Takts CCLK3 eingeladen. In dem ausgangsseitigen Brücken-Chip **48** wird der Ausgangs-Hinweiszeiger OUTPTR[1:0] auf den Wert 0 an der nächsten, ansteigenden Flanke des Takts PCLK in der Periode T3 erhöht. Auch wird, in der Periode T3, der Eingangs-Hinweiszeiger INPTR[1:0] auf den Wert 1 an der ansteigenden Flanke des Takts CCLK3 erhöht, und die Nachricht B wird in den FIFO 1 an der abfallenden Flanke von CCLK3 eingeladen. Kabel-Daten werden demzufolge in FIFO0, FIFO1, FIFO2 und FIFO3 in einer zirkularen Weise eingeladen.

[0273] Auf der Ausgangsseite wird, falls der Eingangs-Hinweiszeiger INPTR[1:0] in der Zeitperiode T0 den Wert 0 hat, der Ausgangs-Hinweiszeiger OUTPTR[1:0] auf den Wert 0 in der Periode T6 erhöht, zwei PCLK Perioden nach dem Eingangs-Hinweiszeiger INPTR[1:0]. Die zwei PCLK Perioden-Verzögerungen in dem ausgangsseitigen Brücken-Chip **26** ermöglicht, dass die Phasenverschiebung in dem Kabel **28** irgendein Wert ist, was den Vorteil hat, dass die Kabellänge nicht von einem spezifischen, festgelegten Wert sein muss.

[0274] Wie Fig. 13 zeigt, werden die Eingangs- und Ausgangs-Flips-Flops an der Kabel-Schnittstelle kundenseitig durch den Hersteller der Chips platziert, um die Verschiebung zwischen den Kabeldaten und dem Takt, der dadurch hindurchgeführt wird, zu minimieren. Die Menge an Draht zwischen jedem Flip-Flop und der I/O wird als so konsistent wie möglich zwischen allen Kabel-Schnittstellen-Signalen beibehalten.

KABEL-NACHRICHT

[0275] Sechzig Bits an Kabel-Daten bilden eine Nachricht. Die 60 Bits werden auf 20 Kabel-Zeilen multiplexiert und werden alle 10 ns über das Kabel **28** übertragen. Die Tabelle in Fig. 14 stellt die Bits dar und die Phase jedes Bits ist zugeordnet. Die ersten drei Spalten stellen das eingangsseitige-zu-ausgangsseitige Daten-Format dar, und die letzten drei Spalten stellen das ausgangsseitig-zu-eingangsseitig Daten-Übertragungs-Format dar. Das Folgende ist eine Beschreibung der Signale.

[0276] EDC[7:0]: Die Signale sind die acht Syndrom-Bits, verwendet dazu, Fehler zu erfassen und zu korrigieren, die beim Übertragen von Daten über das Kabel **28** vorgefunden werden.

[0277] CAD[31:0]: Die Signale sind die 32 Adressen- oder Daten-Bits.

[0278] CFRAME_: Das Signal wird dazu verwendet, den Beginn und das Ende einer Kabel-Transaktion zu signalisieren, ähnlich zu dem PCI FRAME_ Signal.

[0279] CCBE[3:0]_: Die vier Bits bilden Byte-Freigaben in einigen PCI-Takt-Phasen und entweder einen PCI-Befehl oder einen Nachrichten-Code in anderen PCI-Takt-Phasen.

[0280] CBUFF[3:0]: In der Adressen-Phase zeigen die Signale eine Puffer-Zahl zum Initialisieren der verzögerten Abschluss-Warteschlange (DCQ) des Brücken-Chips, **148**, an, um einen eingangsseitigen und ausgangsseitigen, verzögerten Lese-Abschluss (Delayed Read Completion – DRC) und eine verzögerte Lese-Anforderung (Delayed Read Request – DRR) Transaktionen festzulegen. Nach der Adressen-Phase enthalten die Signale das Paritäts-Bit, eine Paritäts-Fehler-Indikation und das Daten-Bereitschafts-Signal.

[0281] COMPLETION REMOVED: Das Bit wird dazu verwendet, zu signalisieren, dass ein verzögerter Abschluss von der Transaktions-Reihenfolge-Warteschlange (Transaction Ordereing Queue – TOC) auf der anderen Seite des Kabels **28** entfernt worden ist.

[0282] PMW ACKNOWLEDGE: Das Bit wird dazu verwendet, zu signalisieren, dass ein gepostetes Speicher-Schreiben (PMW) auf der anderen Seite abgeschlossen worden ist und von der Transaktions-Lauf-Warteschlange (Transaction Run Queue – TRQ) entfernt worden ist.

[0283] LOCK_: Das Bit wird zum Ausgang (allerdings nicht zum Eingang) hin übertragen, um verriegelte Zyklen zu identifizieren.

[0284] SERR_: Das Bit wird dazu verwendet, eine SERR_ Indikation zur Eingangsseite hin zu übertragen, wird allerdings nicht zur Ausgangsseite hin übertragen.

[0285] INTSYNC und INTDATA: Die Bits führen die acht Unterbrechungen von der Ausgangsseite zu der Eingangsseite in einem seriell multiplexierten Format. Das Signal INTSYNC ist das Synchronisations-Signal, das den Start f0 der Unterbrechungs-Sequenz anzeigt, und das Signal INTDATA ist das serielle Daten-Bit. Die Signale INTSYNC und INTDATA werden auf separaten Leitungen (Lines) über das Kabel **28** geführt.

[0286] RESET SECONDARY BUS: Das Bit wird dann aufgestellt, wenn die CPU **14** zu dem sekundären Reset-Bit in einem Brücken-Steuer-Register in dem eingangsseitigen Brücken-Chip **26** schreibt. Es bewirkt, dass sich der ausgangsseitige Brücken-Chip **48** auf einen Power-Up Zustand zurücksetzt. Die Reset Signale für die Schlitze werden auch aufgestellt. Das Signal RESET für den sekundären Bus wird auf einer separaten Leitung über das Kabel **28** weitergeführt.

[0287] Da die Adresse und die Daten in jeder PCI Transaktion über dieselben Leitungen multiplexiert werden, umfasst jede PCI-Transaktion eine Adressen-Phase und mindestens eine Daten-Phase (mehr als eine für Burst-Transaktion). Die PCI-Spezifikation unterstützt auch Einzel-Adressen-Transaktionen (ein 32-Bit Adressieren) und Dual-Adressen-Transaktionen (ein 64-Bit Adressieren).

[0288] In Fig. 15A stellt eine Tabelle dar, welche Informationen an jedem Bereich des Busses während Adressen und Daten-Phasen der Einzel-Adressen-Transaktionen erscheinen. Für eine Einzel-Adressen-Transaktion ist die erste Phase die Adressen-Phase und die zweite und die darauffolgenden Phasen sind Daten-Phasen. In der Adressen-Phase einer verzögerten Lese/Schreib-Anforderungs-Transaktion zeigen die Signale CBUFF[3:0] die DCQ-Puffer-Zahl zum Initialisieren des Brücken-Chips DCQ **148** an, um eingangsseitige und ausgangsseitige DRC- und DRR-Transaktionen festzulegen. Nach der Adressen-Phase enthält das Signal CBUFF[3:0] das Paritäts-Bit. Die Signale CCBE[3:0]_ enthalten den PCI-Befehl in der Adressen-Phase und die Byte-Freigabe-Bits in den Daten-Phasen.

[0289] Für gepostete Speicher-Schreib-Transaktionen sind die Signale CBUFF[3:0] „nicht sicher" in der Adressen-Phase und enthalten die Data-Ready-Indikation, die Parität-Fehler-Indikation und ein Parität-Bit in den Daten-Phasen.

[0290] In einer verzögerten Lese/Schreib-Abschluss-Transaktion enthalten die Signale CBUFF[3:0] die DCQ-Puffer-Zahlen in der Adressen-Phase und die End-Completion-Indikation, eine Daten-Ready-Indikation, eine Parität-Fehler-Indikation und ein Parität-Bit in den Daten-Phasen. Die Signale CCBE[3:0]_ enthalten einen Code, der eine DRC-Transaktion in der Adressenphase und die Status-Bits der DRC-Transaktion in den Daten-Phasen darstellen. Verzögerte Abschluss-Transaktionen führen den Status des Bestimmungs-Busses für jede Daten-Phase zurück. Das Daten-Parität-Bit wird auf CCBE[3]_ übertragen. Andere Status-Zustände werden auf den CCBE[2:0]_BUS codiert, wobei ein binärer Wert 000 einen normalen Abschluss anzeigt und ein binärer Wert 001 einen Target-Aussonderungs-Zustand anzeigt. Die Adressen/Daten-Bits CAD[31:0] sind „nicht sicher" in der Adressen-Phase und enthalten Daten während der Daten-Phasen.

[0291] In der Datenfolge-Verbindungs-Transaktion enthalten die Signale CBUFF[3:0] eine Puffer-Zahl in der Adressen-Phase und das Signal CBUFF[2] enthält die Daten-Ready-Indikation in den Daten-Phasen. Die Signale CCBE[3:0] enthalten einen Code, der eine Datenfolge-Verbindungs-Transaktion in der Adressen-Phase darstellt, und sind nicht in den Daten-Phasen „sicher". Die Adressen-Daten-Bits CAD[31:0] werden nicht während einer Datenfolge-Verbindungs-Transaktion verwendet.

[0292] Die Tabelle in Fig. 15B stellt das Codieren der Signale für Dual-Adressen-Transaktionen dar. In verzögerten Lese/Schreib-Anforderungs-Transaktionen enthalten die Signale CBUFF[3:0] eine Puffer-Zahl in der ersten und der zweiten Adressen-Phase und das Signal CBUFF[0] enthält das Paritäts-Bit in der Daten-Phase. Die Signale CCBE[3:0]_ enthalten einen Code, der einen Dual-Adressen-Zyklus in der ersten Adressen-Phase darstellt, den PCI-Befehl in der zweiten Phase und die Byte-Freigabe-Bits in der Daten-Phase. Die Signale CAD[31:0] enthalten die signifikantesten Adressen-Bits in der ersten Adressen-Phase, die am wenigsten signifikanten Adressen-Bits in der zweiten Adressen-Phase und die Daten-Bits in der Daten-Phase. In einer geposteten Dual-Adressen-Speicher-Schreib-Transaktion sind die Signale CBUFF[3:0] „nicht sicher" in den ersten zwei Adressen-Phasen, allerdings enthalten die Signale CBUFF[1:0] das Paritäts-Fehler-Indikations-Bit und das Paritäts-Bit in den Daten-Phasen. Die Signale CCBE[3:0]_ enthalten einen Code, der einen Dual-Adressen-Zyklus in der ersten Adressen-Phase darstellt, die PCI-Befehl-Bits in der zweiten Adressen-Phase und die Byte-Freigabe-Bits in den Daten-Phasen. Die Signale CAD[31:0] enthalten die signifikantesten Adressen-Bits in der ersten Adressen-Phase, die verbleibenden Adressen-Bits in der zweiten Adressen-Phase und die Daten-Bits in den Daten-Phasen.

[0293] Dabei sind drei mögliche Zustände für die Daten-Übertragung vorhanden: nicht-letzte (not-last), letzte einer Kabel-Übertragung (last-of-cable-transfer) und letzte von (last-of) Anforderung. Der nicht-letzte Zustand wird durch Aufstellen des Bits CBUFF[2] angezeigt, während FRAME_ aktiv ist, was anzeigt, dass ein anderes Wort von Daten vorhanden ist. Der last-of-cable Übertragungs-Zustand wird durch Aufstellen des Bits CBUFF[2; angezeigt, während das Signal CFRAME_ inaktiv ist. Der last-of-request Zustand wird durch Aufstellen der Bits CBUFF[3] und CBUFF[2] angezeigt, während das Signal CFRAME_ inaktiv ist.

[0294] Die folgenden vier IEEE 1149.1 Boundary-Scan (JTAG) Signale sind in dem Kabel **48** umfasst, um eine JTAG-Test-Kette zu bewirken: TCK (der Test-Takt), TDI (Test-Daten-Eingang), TDO (Test-Daten-Ausgang) und TMS (Test-Mode-Auswahl). Das optimale TRST_ wird nicht entlang des Kabels übertragen, allerdings kann TRST_ aus „power-good" erzeugt werden.

[0295] Die JTAG Signale werden von dem System-PCI-Verbinder über den eingangsseitigen Brücken-Chip **26**, umfassend JTAG Master **110**, entlang des Kabels **28** zu dem ausgangsseitigen Brücken-Chip **28** zu dem JTAG Master **128** übertragen, was die JTAG Signale zu jedem der sechs PCI-Schlitze auf dem sekundären PCI-Bus **32** verteilt. Der Rückführpfad reicht von dem JTAG Master **128** bis zum Kabel **28** zurück zu dem ein-

gangsseitigen Brücken-Chip **26** und dann zu dem PCI-Schlitz auf dem primären PCI-Bus **24**. Die Signale TDO, TCK und TMS sind ausgangsseitige Bound-Signale. Das Signal TDI ist ein eingangsseitiges Bound-Signal.

[0296] Ein Typ eines Kabels **28**, das verwendet werden kann, ist ein zylindrisches 50-Paar abgeschirmtes Kabel, ausgelegt dazu, einen High Performance Parallel Interface (HIPPI) Standard zu unterstützen. Ein zweiter Typ eines Kabels ist ein abgeschirmtes fünf-Paar Band-Kabel. Die Vorteile des ersten sind Standardisierung, Robustheit und zuverlässige, gleichförmige Herstellung. Die Vorteile des zweiten sind größere, mechanische Flexibilität, automatisches Verbinden mit dem Verbinder bei der Montage und möglicherweise geringere Kosten.

[0297] Die Tabelle der Fig. 16 stellt einige der HIPPI Kabel-Spezifikationen dar. Die Erdungs-Abschirmung besteht aus einem umwickelten Aluminium-Band und führt nur minimal DC-Ströme aufgrund der unterschiedlichen Art der Puffer, die verwendet werden sollen. Das Verfahren eines Signalisierens ist tatsächlich differenziell, was verschiedene Nachteile liefert, wobei die differenziellen Puffer verwendet werden, um Signale über das Kabel **28** zu verschicken und zu empfangen. Zunächst ist das einzige, differenzielle Verfahren weniger kostspielig als Faseroptiken für diese kurze Distanz und weniger komplex, um sich schnittstellenmäßig zu verbinden, als andere, serielle Verfahren. Ein differenzielles Signalisieren liefert eine wesentliche Rausch-Immunität für einen üblichen Mode und einen Betriebsbereich für einen üblichen Mode, ist in ASICs verfügbar und schneller als TTL. Wenn Twisted Pair und eine Abschirmung verwendet werden, minimiert dies die elektromagnetische Strahlung. Wenn niedrige Spannungsschwingungen verwendet werden, minimiert es eine Energieabnahme.

[0298] Die signalisierenden Pegel, die als ein Target ausgewählt werden, sind in dem IEEE Draft Standard für Low-Voltage Differential Signals (LVDS) für Scaleable Coherent Interface (SCI), Draft 1.10 (5. Mai 1995) beschrieben.

[0299] Der Kabel-Verbinder ist ein AMP-Metall-Mantel-Verbinder mit 100 Stiften, mit zwei Reihen von Stiften. Die Reihen sind 100 mils voneinander beabstandet und die Stifte sind bei 50 mil zentriert. Die Metallhülle liefert eine EMI-Abschirmung und gibt Verbindungen mit dem Massepfad von der Kabelabschirmung zu dem Leiterplattenverbinder. Der passende, rechtwinklige Leiterplattenverbinder passt nur zu einem PCI-Träger. Der Verbinder ist so, dass er einen Stab besitzt, der zwischen den zwei Reihen und Stiften verläuft, um elektrostatische Entladungen von Signalstiften abzuleiten, wenn der Verbinder getrennt wird. Ein Paar Flügelschrauben, befestigt an dem Kabelverbinder, wird die passenden Verbinder sichern.

FEHLER-ERFASSUNG UND -KORREKTUR

[0300] Ein Fehlererfassungs- und Korrektur-(EDC)-Verfahren wird an jedem Brücken-Chip ausgeführt, um eine Kommunikation über das Kabel **28** zu schützen. Da die Daten in drei 20-Bit-Gruppen zeit-multiplexiert werden, um über 20 Paare von Drähten verschickt zu werden, ist jedes Triplet von „angrenzenden" Bits (d. h. Bits, die demselben Draht in dem Kabel **28** zugeordnet sind) so angeordnet, um auf einem einzelnen Draht-Paar übertragen zu werden. Das EDC-Verfahren kann Einzel-Bit-Fehler und Mehrfach-Bit-Fehler korrigieren, die in derselben Bit-Position in jeder der drei zeit-multiplexierten Phasen auftreten. Die Multi-Bit-Fehler sind typischerweise einem Hardware-Fehler zugeordnet, z. B. einem gebrochenen oder defekten Draht oder einem fehlerhaften Stift an den Brücken-Chips **26**, **48**.

[0301] Zwanzig Draht-Paare des Kabels **28** werden für eine ausgangsseitige Kommunikation verwendet und 20 weitere für eine eingangsseitige Kommunikation. Für die verbleibenden 10 Paare in dem 50-Paar-HIPPI-Kabel **28** (das solche Informationen, wie die Takt-Signale CABLE_CLK1 und CABLE_CLK2, Reset-Signale und das Power Good/PLL-Lock Signal durchlässt), wird eine Fehler-Erfassung und -Korrektur nicht durchgeführt.

[0302] Das Folgende sind die Hintergrundannahmen für den EDC-Algorithmus. Die meisten Fehler sind Einzel-Bit-Fehler. Die Wahrscheinlichkeit, zufällige Mehrfach-Bit-Fehler in derselben Transaktion zu haben, ist extrem weit weg, da das Kabel **28** nicht für eine Interferenz von internen oder externen Quellen anfällig ist. Fehler, verursacht durch einen defekten Draht, können bewirken, dass ein einzelnes Bit oder eine Gruppe von Bits auf diesem Draht übertragen wird. Wenn ein Hardware-Fehler auftritt, ist der logische Zustand des entsprechenden, differenziellen Puffers ein Einzel-Gültigkeits-Logik Zustand.

[0303] Wie Fig. 17 zeigt, werden die Ausgangs-Signale FIFOOUT[59:0] von dem Multiplexer **228** in der Slave-Kabel-Schnittstelle **196** oder **198** zu dem Eingang eines Prüf-Bit-Generators **350** zugeführt, der Prüf-Bits

CHKBIT[7:0] erzeugt. Die Prüf-Bits werden entsprechend der Parität-Prüf-Matrix, dargestellt in Fig. 18, erzeugt, in der die erste Reihe zu CHKBIT[0] entspricht, die zweite Reihe zu CHKBIT[1] entspricht, usw.. Die Bits über eine Reihe entsprechen Daten-Bits FIFOOUT[0:59].

[0304] Die Prüf-Bits werden durch ein Exklusiv-ODER aller der Daten-Bits FIFOOUT[X] (X ist gleich zu 0–59) erzeugt, die einen Wert von „1" in der Parität-Prüf-Matrix haben. Demzufolge ist das Prüf-Bit CHKBIT[0] ein Exklusiv-ODER von Daten-Bits FIFOOUT[7], FIFOOUT[8], FIFOOUT[9], FIFOOUT[12], FIFOOUT[13], FIFOOUT[16], FIFOOUT[22], FIFOOUT[23], FIFOOUT[24], FIFOOUT[26], FIFOOUT[32], FIFOOUT[33], FIFOOUT[34], FIFOOUT[35], FIFOOUT[38], FIFOOUT[39], FIFOOUT[45], FIFOOUT[46], FIFOOUT[48], FIFOOUT[49], FIFOOUT[51] und FIFOOUT[52]. Ähnlich ist das Prüf-Bit CHKBIT[1] ein Exklusiv-ODER von Bits 0, 1, 4, 5, 9, 10, 12, 14, 15, 16, 23, 27, 35, 37, 38, 40, 43, 46, 47, 48, 50 und 53. Prüf-Bits CHKBIT[2:7] werden in einer ähnlichen Art und Weise entsprechend der Parität-Prüf-Matrix von Fig. 18 erzeugt. Die Parität-Prüf-Matrix ist auf den 20 Unterkanälen oder Drähten pro zeit-multiplexierter Phase, und einer Wahrscheinlichkeit, dass mehrere Fehler in den akkumulierten Daten einem fehlerhaften Unterkanal oder Draht zuzuschreiben sind, der dieselbe Daten-Position in jeder zeit-multiplexierten Phase beeinflusst, basierend.

[0305] In der Master-Kabel-Schnittstelle **192** oder **194** werden die Prüf-Bits CHKBIT[7:0] als Fehlererfassungs- und Korrektur-Bits EDC[7:0] zusammen mit anderen Kabel-Daten geliefert, um einer Fehler-Korrektur-Logik in der Slave-Kabel-Schnittstelle **196** oder **198** zu ermöglichen, Daten-Fehler zu erfassen und zu korrigieren.

[0306] Die Prüf-Bits CHKBIT[7:0] werden zu einem Fix-Bit-Generator **352** zugeführt, der Fix-Bits FIXBIT[59:0] entsprechend der Syndrom-Tabelle, dargestellt in Fig. 19, erzeugt. Die Prüf-Bits CHKBIT[7:0] besitzen 256 (2^8) mögliche Werte. Die Syndrom-Tabelle in Fig. 19 enthält 256 mögliche Positionen. Jede der 256 Positionen in der Syndrom-Tabelle enthält zwei Eintritte, wobei der erste Eintritt der hexadezimale Wert der Prüf-Bits CHKBIT[7:0] ist und der zweite Eintritt den Kabel-Daten-Status anzeigt, der dieser Position zugeordnet ist. Demzufolge zeigt, zum Beispiel, ein hexadezimaler Wert von 00 einen Nicht-Fehler-Zustand an, ein hexadezimaler Wert von 01 zeigt einen Fehler in einem Daten-Bit **52** an, ein hexadezimaler Wert von 02 zeigt einen Fehler in dem Daten-Bit **53** an, ein hexadezimaler Wert von 03 zeigt einen nicht-korrigierbaren Fehler (UNCER) an, usw..

[0307] Die EDC-Logik ist dazu geeignet, bis zu drei fehlerhafte Bits zu erfassen, so lange wie diese Daten-Bits benachbart zueinander sind, d. h. demselben Draht zugeordnet sind. Demzufolge sind, zum Beispiel, falls die Prüf-Bits CHKBIT[7:0] einen hexadezimalen Wert 3D enthalten, dann die Daten-Bits 3, 23 und 43 fehlerhaft. Das Kabel **28** führt Kabel-Daten CABLE_DATA[19:0]. Demzufolge sind die Daten-Bits FIFOOUT[3], FIFOOUT[23] und FIFOOUT[43] der vierten Position der Kabel-Daten zugeordnet, d. h. CABLE_DATA[3]. Das EDC-Verfahren kann auch Zwei-Bit-Fehler korrigieren, die demselben Kabel-Draht zugeordnet sind. Demzufolge zeigt, zum Beispiel, ein hexadezimaler Prüf-Bit-Wert von 0F Fehler in Daten-Bits FIFOOUT[4] und FIFOOUT[24] an, beide dem CABLE_DATA[4] zugeordnet.

[0308] Der Fix-Bit-Generator **352** erzeugt auch Signale NCERR (nicht-korrigierbarer Fehler) und CRERR (korrigierbarer Fehler). Falls kein Fehler durch die Prüf-Bits angezeigt wird, dann sind die Signale CRERR (korrigierbarer Fehler) und NCERR (nicht-korrigierbarer Fehler) beide auf niedrig zurückgenommen. In diesen Positionen in der Syndrom-Tabelle, die den nicht-korrigierbaren Zustand UNCER enthält, wird das Signal NCERR auf hoch gesetzt und das Signal CRERR wird auf niedrig zurückgenommen. Ansonsten wird dort, wo ein korrigierbarer Daten-Fehler angezeigt wird, das Signal NCERR auf niedrig zurückgenommen und die Signale CRERR werden auf hoch zurückgesetzt.

[0309] Die unteren 52 Bits der Fix-Bits FIXBIT[51:0] werden zu einem Eingang von 52 Exklusiv-ODER-Gates **354** zugeführt, deren anderer Eingang eines von jedem der unteren 52 Bits der FIFO-Daten FIFOOUT[51:0] empfängt. Die oberen 8 FIFO-Bits FIFOOUT[59:52], zugeordnet zu der Fehler-Erfassung und den Korrektur-Bits EDC[7:0], werden dazu verwendet, die Prüf-Bits und die Syndrom-Bits zu erzeugen, werden allerdings nicht einer Fehlerkorrektur unterworfen. Die Exklusiv-ODER-Gates **354** führen eine Bit-weise Exklusiv-ODER-Operation der festgelegten Bits FIXBIT[51:0] und der Daten-Bits FIFOOUT[51:0] aus. Falls die Daten-Signale FIFOOUT[51:0] korrigierbare, fehlerhafte Daten-Bits enthalten, werden diese Daten-Bits durch die Exklusiv-ODER-Operation „geflipped". Die Exklusiv-ODER-Gates **354** liefern die korrigierten Daten CORRMSG[51:0] zu dem 1-Eingang eines Multiplexers **360**. Der 0-Eingang des Multiplexers **360** nimmt die Daten-Bits FIFOOUT[51:0] auf und der Multiplexer **360** wird durch ein Konfigurations-Signal CFG2C_ENABLE_ECC ausgewählt. Der Ausgang des Multiplexers **360** erzeugt Signale MUXMSGI[51:0]. Falls die System-Software eine Fehler-Erfassung und eine Korrektur durch Einstellen des Signals

CFG2C_ENABLE_ECC auf hoch freigibt, dann wählt der Multiplexer **360** die korrigierten Daten CORRMSG[51:0] für eine Ausgabe aus. Ansonsten werden, falls die Fehler-Erfassung und -Korrektur gesperrt ist, die Daten-Bits FIFOOUT[51:0] verwendet.

[0310] Die nicht-korrigierbaren und die korrigierbaren Fehler-Indikatoren NCERR und CRERR werden zu Eingängen von UND-Gates **356** und **358** jeweils geliefert. Die UND-Gates **356** und **358** werden durch das Signal CFG2C_ENABLE_ECC freigegeben. Die Ausgänge der UND-Gates **356** und **358** erzeugen Signale C_NLERR und C_CRERR jeweils. Die Signale C_NLERR und C_CRERR können nur dann aufgestellt werden, wenn eine Fehler-Erfassung und eine Korrektur freigegeben ist. Wenn ein Fehler erfasst ist, werden die festgelegten Bits (Fix-Bits) verriegelt und für diagnostische Zwecke verwendet.

[0311] Falls ein korrigierbarer Fehler angezeigt wird (das Signal C_CRERR ist hoch), dann wird eine Unterbrechung zu dem Unterbrechungs-Empfangs-Block **132** hin erzeugt, weitergeführt zu dem Unterbrechungs-Ausgangs-Block **114**, und dann zu der System-Unterbrechungs-Steuereinheit übertragen, und dann zu der CPU **14**, um einen Unterbrechungs-Händler aufzurufen. Die nicht-korrigierbaren Fehler, angezeigt durch das Signal C_NCERR, werden bewirken, dass der System-Fehler SERR_ aufgestellt wird, was wiederum bewirkt, dass die System-Unterbrechungs-Steuereinheit (nicht dargestellt) die nicht-maskierbare Unterbrechung (Non-Maskable Interrupt – NMI) zu der CPU **14** hin aufstellt.

[0312] In dem ausgangsseitigen Brücken-Chip **48** werden nicht-korrigierbare Fehler bewirken, dass das Power-Good/PLL-Lock-Indikations-Bit weiter zu dem Kabel **28** geschickt wird, um vernachlässigt zu werden, so dass der eingangsseitige Brücken-Chip **26** keine Zyklen zur Ausgangsseite hin schickt.

[0313] Um zufällige Unterbrechungen während und nach einem Power-up-Vorgang zu verhindern, wird eine Fehlererfassung und -korrektur an sowohl dem eingangsseitigen als auch dem ausgangsseitigen Brücken-Chip während eines Power-up-Vorgangs gesperrt, bis sich die eingangsseitge PLL **186** und die ausgangsseitige PLL **182** auf den Takt CABLE_CLK1 oder CABLE_CLK2 verriegelt haben.

[0314] Eine System-Mangament-Software, die auf die Unterbrechung für korrigierbare Fehler anspricht, bestimmt die Ursache durch Lesen der verriegelten Fix-Bits. Falls ein Hardware-Fehler bestimmt wird (z. B. Mehrfach-Daten-Fehler-Bits, zugeordnet demselben Kabel-Draht), dann kann die System-Management-Software den Benutzer auf den Zustand hinweisen, um den Hardware-Fehler zu beseitigen. Die System-Management-Software spricht auf SERR_ an, verursacht durch einen nicht-korrigierbaren Fehler, unter Abschalten des Systems oder unter Durchführen von anderen Funktionen, die durch den Benutzer programmiert sind.

SEKUNDÄR-BUS-ARBITRIERER

[0315] Wie die Fig. 3 zeigt, umfasst jeder Brücken-Chip einen PCI-Arbitrierer **116** oder **124**. Da der eingangsseitige Brücken-Chip **26** normalerweise in einem Schlitz installiert ist, wird der PCI-Arbitrierer **116** gesperrt. Der PCI-Arbitrierer **124** unterstützt 8 Master: 7 allgemeine PCI-Master (REQ[7:1]_, GNT[7:1]_, umfassend die sechs PCI-Schlitze und die Hot-Plug-Steuereinheit in der SIO **50**, und den Brücken-Chip selbst (BLREQ_, BLGNT_). Die Signale BLREQ_ und BLGNT_ werden von und zu dem PCI-Master-Block **123** geführt. Der Brücken-Chip stellt das Signal BLREQ_ auf, falls eine Transaktion von der CPU **14**, zielmäßig vorgesehen für den sekundären PCI-Bus **32**, durch den eingangsseitigen und ausgangsseitigen Brücken-Chip **26** und **48** empfangen ist. Die Anforderungs- und Erteilungs-Leitungen REQ[1]_ und GNT[1]_ für die SIO **50** werden intern in den ausgangsseitigen Brücken-Chip **48** weitergeleitet. Der PCI-Arbitrierer **124** setzt eine PCICLK2 Verzögerung zwischen einer Negation eines GNT_ Signals für einen Master und das Aufstellen eines GNT_ Signals für einen anderen Master ein.

[0316] In dem ausgangsseitigen Brücken-Chip **48** wird der PCI-Arbitrierer **124** freigegeben oder gesperrt, und zwar basierend auf dem abgetasteten Wert von REQ[7]_ an der ansteigenden Flanke des Signals PCIRST2_. Falls der Brücken-Chip **48** REQ[7]_ niedrig auf PCIRST2_ abtastet, wird er den PCI-Arbitrierer **124** sperren. Falls der PCI-Arbitirierer **124** gesperrt ist, dann wird ein externer Arbitrierer (nicht dargestellt) verwendet, und die Hot-Plug-Anforderung wird auf dem REQ[1]_ Stift angesteuert und die Hot-Plug-Erteilung wird auf dem GNT[1]_ Stift eingegeben. Die Brücken-PCI-Bus-Anforderung ist auf dem REQ[2]_ Stift angesteuert und deren Erteilung wird auf dem GNT[2]_ Stift eingegeben. Falls der Brücken-Chip **48** REQ[7]_ auf hoch auf PCIRST2_ abtastet, wird er den PCI-Arbitrierer **124** freigeben.

[0317] Der PCI-Arbitrierer **124** negiert ein GNT_ Signal des Masters, entweder um einen Initiator mit höherer Priorität zu bedienen, oder auf das REQ_ Signal des Masters hin, das negiert werden soll. Wenn einmal sein

GNT_ Signal negiert ist, hält der momentane Bus-Master den Besitz über den Bus bei, bis der Bus zu seinem Leerlauf zurückkehrt.

[0318] Falls keine PCI-Agenten momentan den Bus verwenden oder anfordern, nimmt der PCI-Arbitrierer **124** eines von zwei Dingen in Abhängigkeit von dem Wert eines PARKMSTRSEL Konfigurations-Registers in dem Konfigurations-Raum **125** vor. Falls das Register den Wert von 0 enthält, verwendet der PCI-Arbitrierer **124** den letzten, aktiven Master, um auf dem Bus **32** zu parken; falls er den Wert 1 enthält, dann wird der Bus an dem Brücken-Chip **48** geparkt.

[0319] Der PCI-Arbitrierer **124** umfasst einen PCI-Minimal-Erteilungs-Zeitgeber **304** (Fig. 21), der die minimale, aktive Zeit aller der GNT_ Signale steuert. Der Fehler-Wert für den Zeitgeber **304** ist der hexadezimale Wert 0000, der anzeigt, dass dort kein minimales Erteilungszeit-Erfordernis vorhanden ist. Der Zeitgeber **304** kann mit einem Wert von 1 bis 255 programmiert sein, um anzuzeigen, dass die Zahl von PCICLK2 Taktperioden der GNT_ Leitung aktiv ist. Alternativ kann der individuelle, Minimum-Erteilungs-Zeitgeber jedem PCI-Master auf dem sekundären Bus **32** zugeordnet sein, um eine größere Flexibilität zu erzielen. Die Minimum-Erteilungs-Zeit ist nur dann anwendbar, wenn der momentane Master sein REQ_ Signal aufstellt. Wenn einmal das REQ_ Signal weggenommen ist, kann das GNT_ Signal ungeachtet des minimalen Erteilungs-Zeitwerts erteilt werden.

[0320] Wie Fig. 20A zeigt, führt, in einem normalen Betrieb, der PCI-Arbitrierer **124** ein Round-Robin-Prioritäts-Schema (Arbitrierungs-Schema auf dem zweiten Niveau) aus. Die acht Master in dem Round-Robin-Schema umfassen Vorrichtungen, die mit den sechs Schlitzen bzw. Einsteckplätzen des Erweiterungskastens **30**, dem SIO **50** und einer geposteten Speicher-Schreib-(PMW)-Anforderung von dem eingangsseitigen Brücken-Chip **26** verbunden sind. Alle Master auf dem PCI-Bus **32** in diesem Schema besitzen dieselbe Priorität wie der Brücken-Chip **48**. Nachdem der Master den sekundären PCI-Bus **32** erteilt hat und der Master das FRAME_ Signal aufgestellt hat, wird der Bus erneut arbitriert und der momentane Master wird an die Unterseite des Round-Robin-Stapels gesetzt. Falls der Master diese Anforderung negiert oder der Minimum-Erteilungs-Zeitgeber **304** abläuft, wird der PCI-Bus **32** dem Master mit der nächsten, höchsten Priorität erteilt. Verriegelte Zyklen werden nicht in irgendeiner Weise unterschiedlich durch den PCI-Arbitrierer **124** behandelt.

[0321] Auf bestimmte Ereignisse hin wird das Arbitrierungs-Schema so modifiziert, um eine System-Funktion zu optimieren. Die Ereignisse umfassen: 1) eine eingangsseitig-zu-ausgangsseitig verzögerte Lese- oder verzögerte Schreib-Anforderung ist anhängig, 2) eine ausgangsseitig-zu-eingangsseitig verzögerte Lese-Anforderung ist anhängig, ohne dass eine Lese-Abschluss-Indikation geliefert wird, und 3) eine Streaming-Möglichkeit existiert, während der Brücken-Chip **26** der momentane Master auf dem eingangsseitigen Bus **24** ist.

[0322] Wenn eine verzögerte Anforderung erfasst ist, wird der Brücken-Chip **48** der nächste Master, der dem sekundären PCI-Bus **32** erteilt werden soll. Wenn einmal dem Brücken-Chip **48** der Bus **32** erteilt ist, behält er einen Besitz über den Bus **32** bei, bis er alle offenstehenden, verzögerten Anforderungen abschließt oder einer seiner Zyklen erneut versucht wird. Falls der Brücken-Chip **48** erneut versucht wird, wird ein Zwei-Level-Arbitrierungs-Schema durch den Arbitrierer **124** ausgeführt. Eine primäre Ursache dafür, dass der Brücken-Chip-Lese-Zyklus erneut versucht wird, ist diejenige, dass die Target-Vorrichtung eine Brücke mit einem geposteten Schreib-Puffer ist, der gelöscht werden muß. In diesem Fall ist die optimale Operation diejenige, den Bus **32** zu dem erneut versuchenden Target zu erteilen, um ihm zu ermöglichen, seinen geposteten Schreib-Puffer zu entleeren, so dass er die Brücken-Chip-Lese-Anforderung annehmen kann.

[0323] Wie Fig. 20B zeigt, umfasst ein Zwei-Level-Arbitrierungs-Protokoll ein Arbitrierungs-Schema auf dem ersten Level, das ein Round-Robin-Schema ist, und zwar unter drei möglichen Mastern: die verzögerte Anforderung von der CPU **14**, eine Anforderung von dem erneut versuchenden Master und ein Master, der durch das Arbitrierungs-Schema auf dem zweiten Level ausgewählt ist. Jeder dieser drei Master in dem Arbitrierungs-Schema unter dem ersten Level wird jedem dritten Arbitrierungs-Schlitz erteilt. Für Speicher-Zyklen kann der Schlitz, der dem erneut versuchenden Target zugeordnet ist, von den Target-Speicher-Bereich-Konfigurations-Registern in dem Konfigurations-Raum **125** des Brücken-Chips **48** bestimmt werden, der den Speicherbereich speichert, der jeder PCI-Vorrichtung zugeordnet ist. Falls der erneut versuchende Master nicht bestimmt werden kann (wie in dem Fall einer I/O-Lesung), oder falls der erneut versuchende Master nicht den sekundären Bus **32** anfordert, dann würde das Arbitrierungs-Schema auf dem ersten Level zwischen dem Brücken-Chip **48** und einem Level-Two-Master vorliegen.

[0324] Der erneut versuchende Master wird nicht von der Level-Zwei-Arbitrierung maskiert. Demzufolge ist es für ihn möglich, zwei Back-to-Back-Arbitrierungs-Gewinne zu haben, falls er der nächste Master in dem Le-

vel-Two-Arbitrierungs-Schema ist.

[0325] Zum Beispiel würde, falls eine Eingangs-zu-Ausgangs-Lesung erneut versucht wird und der Master C (der erneut versuchende Master) den Bus **32** ebenso wie einen Master B und einen Master E anfordert, die Reihenfolge der Bus-Erteilungen wie folgt in einer absteigenden Reihenfolge sein: der Brücken-Chip **48**, der erneut versuchende Master (Master C), der Master C, der Brücken-Chip **48**, der erneut versuchende Master C, der Master E, der Brücken-Chip **48**, usw., bis der Brücken-Chip **48** in der Lage ist, seine Transaktion abzuschließen, und der PCI-Arbitrierer **152** zurück zu seinem Level-Two-Arbitrierungs-Schema für einen normalen Betrieb kehrt.

[0326] Falls, als ein anderes Beispiel, die Brücken-Chip-Lesung erneut versucht wird und die einzigen anderen die anfordernden Master Master A und Master D sind (d. h. der erneut versuchende Master fordert nicht den Bus an, oder er könnte nicht identifiziert werden, da er auf einen I/O-Raum gerade zugreift), ist die Reihenfolge der Bus-Erteilungen wie folgt: der Brücken-Chip **48**, der Master A, der Brücken-Chip **48**, der Master D, usw..

[0327] Das Zwei-Level-Arbitrierungs-Schema gibt verzögerten Anforderungen von der CPU **14** die höchste Priorität. Obwohl dieses Arbitrierungs-Verfahren stark die CPU **14** favorisiert, wird jeder anfordernden Vorrichtung auf dem Bus **32** schließlich der PCI-Bus **32** erteilt. Indem dies so vorgenommen wird, ist dabei eine geringere Chance vorhanden, dass die anderen, sekundären Bus-Master vernachlässigt werden würden, wenn eine PCI-Brücken-Chip-Anforderung erneut versucht wird.

[0328] Unter Bezugnahme auf Fig. 21 umfasst der PCI-Arbitrierer **124** eine L2 Zustand-Maschine **302**, um das Level-Two-Round-Robin-Arbitrierungs-Schema auszuführen. Die L2 Zustand-Maschine **302** empfängt Signale RR_MAST[2:0], die den momentanen Round-Robin-Master anzeigen. Die L2 Zustand-Maschine **302** empfängt auch Anforderungs-Signale RR_REQ[7:0], entsprechend zu den 8 möglichen Mastern des sekundären PCI-Busses **32**. Basierend auf dem momentanen Master und dem Zustand der Anforderungs-Signale, erzeugt die L2 Zustand-Maschine **302** einen Wert, der den nächsten Round-Robin-Master darstellt. Der Ausgang der L2 Zustand-Maschine **302** wird zu dem 0-Eingang eines 6 : 3 Multiplexers **306** geliefert, dessen 1-Eingang Signale Q2A_STRMAST[2:0] empfängt. Der Auswahl- bzw. Select-Eingang des Multiplexers **306** empfängt ein Signal STREAM_REQ, das auf hoch durch ein UND-Gate **308** gesetzt wird, wenn eine Streaming-Gelegenheit existiert (Q2A_STREAM ist hoch), der Streaming-Master auf dem sekundären PCI-Bus **32** seine Anforderungs-Zeile (MY_REQ[Q2A_STRMAST[2:0]] ist hoch) und eine verzögerte Anforderung nicht anhängig ist (BAL_DEL_REQ ist niedrig).

[0329] Der Ausgang des Multiplexers **306** steuert Signale N_RR_MAST[2:0] an, die den nächsten Round-Robin-Master in dem Level-Two-Arbitrierungs-Schema darstellen. Die Signale N_RR_MAST[2:0] werden durch eine L1 Zustand-Maschine **300** empfangen, die auch die folgenden Signale empfängt: ein Signal RTRYMAST_REQ (das die Anforderung des erneut versuchenden Bus-Masters darstellt); ein Signal MIN_GRANT (das dann aufgestellt wird, wenn der Minimal-Erteilungs-Zeitgeber **304** zeitmäßig abläuft); das verzögerte Anforderungs-Signal BAL_DEL_REQ; das Datenfolge-Anforderungs-Signal STREAM_REQ; ein Signal CURMAST_REQ (das anzeigt, dass der momentane Master ein Aufstellen seines Anforderungs-Signals beibehält); ein Signal_ANY_SLOT_REQ (das auf hoch gestellt wird, falls irgendeines der Anforderungs-Signale REQ[7:0]_ auf hoch, allerdings nicht die Brücken-Chip-Anforderung BLREQ_ umfassend, aufgestellt ist); und Signale L1STATE[1:0] (die den momentanen Zustand der L1 Zustand-Maschine **300** darstellt). Die L1 Zustand-Maschine **300** wählt eine von drei möglichen L1-Mastern aus, umfassend den erneut versuchenden Master (RTRYMAST_REQ), die verzögerte Anforderung von dem Brücken-Chip **48** (BAL_DEL_REQ) und den Level-Two-Master (ANY_SLOT_REQ).

[0330] Das Anforderungs-Signal des erneut versuchenden Master RTRYMAST_REQ wird durch ein UND-Gate **312** erzeugt, das das Signal BAL_DEL_REQ, das Signal MY_REQ[RTRY_MAT[2:0]] (das anzeigt, ob der erneut versuchende Master seine Anforderung aufstellt) und den Ausgang eines ODER-Gates **310** empfängt. Die Eingänge des ODER-Gates **310** nehmen die Signale RTRY_MAST[2:0] auf. Demzufolge ist, falls ein erneut versuchender Master identifiziert worden ist (RTRY_MAST[2:0] ist Nicht-Null), eine verzögerte Anforderung vorhanden ist (BAL_DEL_REQ ist hoch) und ein erneut versuchender Master seine Anforderung aufgestellt hat, dann das Signal RTRYMAST_REQ aufgestellt.

[0331] Die L1 Zustand-Maschine **300** erzeugt Signale N_L1STATE[2:0] (die den nächsten Zustand der L1 Zustand-Maschine **300** darstellen), ebenso wie Signale N_CURMAST[2:0] (die den nächsten Master gemäß dem Level-Two-Arbitrierungs-Schema darstellen). Die L1 Zustand-Maschine **300** erzeugt auch ein Signal

OPEN_WINDOW, das anzeigt, wenn ein erneut arbitrierendes Fenster für eine Erteilungs-Zustand-Maschine **306** existiert, um Master auf dem sekundären PCI-Bus **32** zu ändern. Ein Signal ADV_RR_MAST, geliefert durch die L1 Zustand-Maschine **300**, zeigt zu der Erteilungs-Zustand-Maschine **306** hin an, wenn der Wert der Signale N_RR_MAST[2:0] in die Signale RR_MAST[2:0] zu laden sind, um den nächsten Level-Two-Round-Robin-Master weiterzuführen.

[0332] Die Erteilungs-Zustand-Maschine **306** gibt Erteilungs-Signale GNT[7:0] ebenso wie ein Signal CHANGING_GNT aus, um anzuzeigen, dass die Inhaberschaft des Busses **32** geändert wird. Die Erteilungs-Signale GNT[7:1]_ werden von den GNT[7:1] Signalen invertiert, und das Erteilungs-Signal BLGNT_ wird von dem GNT[0] Signal invertiert. Die Erteilungs-Zustand-Maschine **306** erzeugt auch Signale L1STATE[1:0] und Signale RR_MAST[2:0].

[0333] Der Minimum-Erteilungs-Zeitgeber **304** wird durch das Signal PCLK getaktet und erzeugt das Signal MIN_GRANT. Der Minimum-Erteilungs-Zeitgeber **304** empfängt auch das Signal CHANGING_GNT und NEW_FRAME (anzeigend, dass ein neues FRAME_ Signal aufgestellt worden ist). Der Anfangs-Wert des Minimum-Erteilungs-Zeitgebers **304** wird als ein Wert geladen {CFG2A_MINGNT[3:0], 0000}, wobei die Signale CFG2A_MINGNT[3:0] gespeicherte Konfigurations-Bits in dem Konfigurations-Raum **125** sind, die den Anfangs-Wert des Minimum-Erteilungs-Zeitgebers **304** definieren. Der Minimum-Erteilungs-Zeitgeber **304** wird erneut geladen, nachdem er herunter auf Null gezählt hat, und das Signal CHANGING_GNT wird auf hoch gesetzt. Nachdem der Minimum-Erteilungs-Zeitgeber **304** mit einem neuen Wert geladen ist, beginnt er sich zu erniedrigen, wenn das Signal NEW_FRAME auf hoch gesetzt ist und das Signal CHANGING_GNT auf niedrig durch die Erteilungs-Zustand-Maschine **306** zurückgenommen ist, was anzeigt, dass eine neue Transaktion auf dem PCI-Bus **32** begonnen hat.

[0334] Signale MY_REQ[7:1] werden durch ein NOR-Gate **314** erzeugt, dessen Eingänge die Anforderungs-Signale REQ[7:1]_ und Maskierungs-Signale Q2AMASKREQ[7:1] empfangen. Ein Aufstellen des Maskierungs-Bits Q2AMASKREQ[X], X = 1–7, maskiert die Anforderung REQ[X]_ des entsprechenden Masters, was verhindert, dass der PCI-Arbitrierer **124** auf das Anforderungs-Signal anspricht. Ein Signal MY_REQ[0] wird durch einen Invertierer **316** angesteuert, der die Brücken-Anforderung BLREQ empfängt.

[0335] Wie Fig. 22 zeigt, umfasst die Erteilungs-Zustand-Maschine **306** vier Zustände: PARK, GNT, IDLE4GNT und IDLE4PARK. Beim Aufstellen eines RESET-Signals (erzeugt von dem PCI-Reset-Signal PCIRST2_), tritt die Erteilungs-Zustand-Maschine **306** in einen Zustand PARK ein, wo sie verbleibt, während ein Signal ANY_REQ weggenommen wird. Das Signal ANY_REQ wird auf hoch gesetzt, falls irgendeine der Anforderungs-Zeilen zu dem PCI-Arbitrierer **124** aufgestellt ist. In dem PARK Zustand wird die PCI-PCI-Brücke **48** als der Inhaber des PCI-Busses **32** geparkt, wenn eine andere Anforderung nicht vorhanden ist.

[0336] Falls das Signal ANY_REQ aufgestellt ist, geht die Erteilungs-Zustand-Maschine **306** von dem Zustand PARK zu dem Zustand IDLE4GNT über, und das Signal CHANGING_GNT wird auf hoch gesetzt, um anzuzeigen, dass der PCI-Arbitrierer **124** Master ändert. Die Erteilungs-Signale GNT[7:0] werden alle auf Null'en gelöscht, und die Signale CURMAST[2:0] werden mit dem Wert des nächsten Master N_CURMAST[2:0] aktualisiert. Zusätzlich werden die Round-Robin-Master-Signale RR_MAST[2:0] mit dem nächsten Round-Robin-Master-Wert N_RR_MAST[2:0] aktualisiert, wenn das Signal ADV_RR_MAST durch die L1 300 aufgestellt ist. Das Signal ADV_RR_MAST zeigt, wenn es hoch ist, an, dass der nächste L1-Master einer der L2 Master ist.

[0337] Von einem Zustand IDLE4GNT geht die Erteilungs-Zustand-Maschine **306** als nächstes zu dem GNT Zustand über, und die Signale GNT[7:0] werden auf den Zustand von neuen Erteilungs-Signalen NEWGNT[7:0] eingestellt und das Signal CHANGING GNT werden auf niedrig gesetzt. Die Signale NEWGNT[7:0] sind auf dem Zustand der momentanen Master-Signale CURMAST[2:0] basierend, wie dies in Fig. 24 dargestellt ist.

[0338] Von dem Zustand GNT sind drei Übergänge möglich. Die Erteilungs-Zustand-Maschine **306** kehrt zu dem PARK Zustand zurück, falls ein Arbitrierungs-Fenster offen ist (OPEN_WINDOW ist hoch), keine Anforderung anhängig ist (ANY_REQ ist niedrig), der PCI-Bus **32** leer ist (BUS_IDLE ist hoch) und der nächste Master der momentane Master ist (d. h. der momentane Master ist der parkende Master). Bei dem Übergang zurück von dem GNT Zustand zu dem PARK Zustand werden die Signale L1STATE[1:0] mit den Signalen N_L1STATE[1:0] aktualisiert. Allerdings wird, falls keine Anforderungen anhängig sind und der Bus leer ist, aber der momentane Master nicht der parkende Master ist (d. h. die Signale N_CURMAST[2:0] sind nicht gleich zu dem Wert der Signale CURMAST[2:0]), ein Leerlauf-Zustand benötigt und die Erteilungs-Zu-

stand-Maschine **306** geht von dem GNT Zustand zu dem IDLE4PARK Zustand über. Die L1 Zustand-Werte L1STATE[1:0] werden aktualisiert. Von dem IDLE4PARK Zustand geht die Erteilungs-Zustand-Maschine **306** zu dem PARK Zustand über, die Erteilungs-Signale GNT[7:0] gleich zu den neuen Erteilungs-Signalen NEWGNT[7:0] einstellend, um den PCI-Bus **32** dem neuen Master zu erteilen. Das Signal CHANGING_GNT wird auch auf niedrig gesetzt.

[0339] Falls sich das Arbitrierungs-Fenster öffnet (OPEN_WINDOW ist hoch) und der nächste Master nicht der momentane Master ist (die Signale N_CURMAST[2:0] sind nicht gleich zu den Signalen CURMAST[2:0]), dann geht die Erteilungs-Zustand-Maschine **306** zu dem Leerlauf Zustand IDLE4GNT über, um Bus-Master-Erteilungen zu ändern. Bei dem Übergang wird das Signal CHANGING_GNT hoch gesetzt, die Signale GNT[7:0] werden alle auf Null'en gelöscht, die Signale CURMAST[2:0] werden mit dem nächsten Master-Wert N_CURMAST[2:0] aktualisiert, und die L1 Zustand-Signale L1STATE[1:0] werden mit dem nächsten Zustand-Wert N_L1STATE[1:0] aktualisiert. Zusätzlich werden die Round-Robin-Master-Signale RR_MAST[2:0] mit dem nächsten Round-Robin-Master RR_MAST[2:0] aktualisiert, falls das Signal ADV_RR_MAST auf hoch gesetzt ist. Die Erteilungs-Signale GNT[7:0] werden dann auf den Wert NEWGNT[7:0] bei dem Übergang von dem IDLE4GNT Zustand zu dem GNT Zustand gesetzt.

[0340] Wie Fig. 23 zeigt, startet die L1 Zustand-Maschine **300** (Fig. 21) in eine, Zustand RR unter Aufstellen des RESET-Signals, wo die Zustand-Maschine **300** verbleibt, während ein verzögertes Anforderungs-Signal BAL_DEL_REQ auf niedrig gesetzt wird (anzeigend, dass dort keine verzögerte Anforderung anhängig ist). In dem RR Zustand wird das Signal ADV_RR_MAST auf hoch gesetzt, um der Erteilungs-Zustand-Maschine **300** zu ermöglichen, den Round-Robin-Master zu aktualisieren (d. h. Einstellen von Signalen RR_MAST[2:0] gleich zu dem Wert N_RR_MAST[2:0]). Der RR Zustand ist der Round-Robin Zustand, in dem das Level-Two-Arbitrierungs-Schema verwendet wird. In dem RR Zustand werden die nächsten Master-Signale N_CURMAST[2:0] gleich zu dem nächster Round-Robin-Master N_RR_MAST[2:0] gesetzt, und das Signal OPEN_WINDOW wird auf hoch gesetzt, falls die Datenfolge-Anforderungs-Gelegenheit existiert (STREAM_REQ ist hoch), oder der Minimum-Erteilungs-Zeitgeber **304** abgelaufen ist (MIN_GRANT ist hoch), oder der momentane Master seine Anforderung aufgestellt hat (CURMAST_REQ geht auf niedrig). Wenn hoch aufgestellt ist, ermöglicht das Signal OPEN_WINDOW, dass eine neue Arbitrierung stattfindet.

[0341] Falls eine verzögerte Anforderung erfasst wird (BAL_DEL_REQ geht zu hoch über), geht die L1 Zustand-Maschine **300** von dem RR Zustand zu dem BAL Zustand über, den nächsten Master Zustand N_CURMAST[2:0] als den Brücken-Chip **48** einstellend und das Signal ADV_RR_MAST wegnehmend, um die Level-Two-Round-Robin-Arbitrierung zu sperren. In dem BAL Zustand wird das Signal OPEN _WINDOW auf hoch gesetzt, falls die verzögerte Anforderung weggenommen ist (BAL_DEL_REQ geht zu niedrig über), oder die verzögerte Anforderung erneut versucht worden ist (BAL_RETRIED geht zu hoch über). Falls das Signal BAL_DEL_REQ auf niedrig gesetzt ist oder falls die verzögerte Anforderung BAL_DEL_REQ auf hoch gesetzt ist, allerdings die Anforderung des erneut versuchenden Masters auf niedrig gesetzt wird (RTRYMAST_REQ ist niedrig) und die Schlitz-Anforderung ANY_SLOT_REQ auf hoch gesetzt wird, dann geht die L1 Zustand-Maschine **300** zurück zu dem RR Zustand. Bei dem Übergang wird das Signal ADV_RR_MAST auf hoch gesetzt und die nächsten Master-Signale N_CURMAST[2:0] werden gleich zu dem nächsten Round-Robin-Master N_RR_MAST[2:0] gesetzt. Falls das Signal BAL_DEL_REQ weggenommen ist, zeigt dies an, dass der Arbitrierer **124** zurück zu dem Level-Two-Round-Robin-Schema kehren sollte. Falls das Signal der verzögerten Anforderung aufgestellt ist, allerdings die Anforderung des erneut versuchenden Masters weggenommen wird, dann ist das Level-One-Arbitrierungs-Schema zwischen den Schlitzen auf dem PCI-Bus **32** und dem Brücken-Chip **48**.

[0342] Falls sowohl die verzögerte Anforderung BAL_DEL_REQ als auch die Anforderung des erneut versuchenden Masters RTRYMAST_REQ aufgestellt sind, dann geht die L1 Zustand-Maschine **300** von dem Zustand BAL zu dem Zustand RETRY_MAST über, und der erneut versuchende Master wird als der nächste Master eingestellt (N_CURMAST[2:0] wird gleich zu RTRY_MAST[2:0]) eingestellt. Das Signal ADV_RR_MAST wird auf niedrig beibehalten. In dem RETRY_MAST Zustand ist, falls keiner der PCI-Schlitz-Master eine Anforderung aufstellt (ANY_SLOT_REQ ist niedrig), dann das Level-One-Arbitrierungs-Schema zwischen dem erneut versuchenden Master und dem Brücken-Chip **48**, und die L1 Zustand-Maschine **300** geht zurück zu dem BAL Zustand. Der Brücken-Chip **48** wird als der nächste Master eingestellt (N_CURMAST[2:0] ist gleich zu dem Zustand BAL-BOA), und das Signal ADV_RR_MAST wird auf niedrig beibehalten. Allerdings geht die L1 Zustand-Maschine **300** von dem RETRY_MAST Zustand zu dem RR Zustand über, falls irgendeiner der Schlitz-Master eine Anforderung aufstellt (ANY_SLOT_REQ ist hoch). Bei dem Übergang wird das Signal ADV_RR_MAST auf hoch gesetzt, und der nächste Round-Robin-Master wird als der nächste Master eingestellt (N_CURMAST[2:0] wird gleich zu N_RR_MAST[2:0] eingestellt).

[0343] Um von dem Vorteil der Streaming-Fähigkeiten des Brücken-Chips Gebrauch zu machen, wenn Daten für eine DRC damit beginnen, von dem Kabel **28** anzukommen, wird der Master, zugeordnet zu dieser DRC, die Vorrichtung mit der höchsten Priorität (unter der Annahme, dass deren REQ_ aufgestellt ist). Dies ermöglicht dem Master, die Daten-Folge, von dem Kabel **28** ankommend, aufzunehmen, während das Auswahl-Fenster dort für ein Streaming vorhanden ist. Falls der Brücken-Chip **48** nicht den Master verbinden kann, bevor die DRC-Warteschlange aufgefüllt ist, dann wird sich der eingangsseitige Brücken-Chip **24** trennen, und nur ein Teil der Daten würde zu dem anfordernden Master hindurchgeführt werden, was erfordert, dass der Master eine andere Lese-Anforderung auf den eingangsseitigen Bus **24** abgibt. Der Streaming-Master behält die höchste Priorität bei, so lange wie DRC-Daten fortfahren, von dem Kabel **28** anzukommen. Falls der Master eine unterschiedliche Zyklus-Adresse wiederholt, wird dies erneut versucht werden, allerdings wird er den Besitz über den sekundären PCI-Bus **32** beibehalten, bis seine Anforderung weggeht oder die Gelegenheit für ein Streaming vorüber ist.

ERNEUT VERSUCHENDE ANFORDERUNGEN UND MULTI-THREADED MASTER

[0344] Da jeder Brücken-Chip eine Vorrichtung mit verzögerter Transaktion ist, falls eine Vorrichtung auf dem ausgangsseitigen Bus **32** eine Lese-Anforderung abgibt, bestimmt für ein eingangsseitiges Target, wird der ausgangsseitige Brücken-Chip **48** eine Wiederversuch-Transaktion (beschrieben in der PCI Spezifikation) auf dem sekundären Bus **32** ausgeben und die Anforderung weiter zu dem Kabel **28** führen. Die Wiederversuch-Transaktion bewirkt, dass der anfordernde Master die Steuerung von dem PCI-Bus **32** aufgibt und seine REQ_ Zeile wegnimmt. Nach Wegnehmen seiner REQ_ Zeile wird der erneut versuchende Master wieder eine Anforderung für denselben Zyklus zu einer späteren Zeit aufstellen, was dazu führt, dass seine GNT_ aufgestellt wird (falls seine REQ_ Zeile nicht maskiert ist) und der Bus-Master erneut versucht, bis die Lese-Abschluss-Indikation in dem ausgangsseitigen Brücken-Chip **48** aufgestellt ist.

[0345] Wie die Fig. 25 zeigt, wird, um die unnötige Bearbeitung von Wiederversuch-Anforderungen zu vermeiden, die REQ_ Zeile einen sekundären Bus-Master, er eine erneut versuchte, verzögerte Lese- oder Schreib-Anforderung ausgibt, durch Aufstellen des geeigneten einen der Signale Q2A_MASK_REQ[7:1] (Anforderung von dem Brücken-Chip **48**, die erneut versucht sind, werden nicht maskiert), maskiert, bis der verzögerte Abschluss zurückführt. Auf diese Art und Weise wird anderen, anfordernden Mastern eine Priorität gegeben, um deren Anforderungen hineinzubringen. Sobald die ersten Informationen, zugeordnet dem verzögerten Abschluss, zurückgeführt werden, wird die REQ_ Zeile des entsprechenden Masters von der Maskierung befreit und der erneut versuchende Master ist in der Lage, in eine Arbitrierung erneut einzutreten.

[0346] Allerdings existiert ein spezieller Fall für Multi-Threaded- (oder Multi-Headed) Master auf dem ausgangsseitigen Bus **32** (Fig. 26B), die in der Lage sind, eine erste Anforderung aufzustellen, erneut versucht zu werden und zurückzukommen mit einer unterschiedlichen Anforderung. Eine solche Multi-Threaded-Bus-Vorrichtung ist eine PCI-PCI-Brücke **323**, die den sekundären PCI-Bus **32** und einen untergeordneten PCI-Bus **325** verbindet. Der Bus **325** ist mit Netzwerk-Schnittstellen-Karten (Network Interface Cards – NICs) **327A** und **327B** verbunden, die mit zwei unterschiedlichen Netzwerken verbunden sind. Demzufolge kann, falls die Anforderung von der NIC **327A** für den primären PCI-Bus **32** erneut durch den Brücken-Chip **48** versucht wird, die NIC **327B** eine unterschiedliche Anforderung erzeugen. In diesem Fall werden die REQ_ Zeilen der Multi-Threaded-Master nicht maskiert, wie dies durch das Signal CFG2Q_MULTI_MASTER[X], das auf hoch gesetzt wird, angezeigt ist.

[0347] Ein Status-Register **326** bestimmt, ob ein Schlitz ein einzelner oder ein Multi-Threaded ist. Bei einem Reset wird das Register **326** gelöscht, um anzunehmen, dass jede sekundäre Bus-Vorrichtung Einzel-Threaded ist. Der Schlitz wird dann überwacht, um zu bestimmen, ob er einen unterschiedlichen Zyklus anfordert, während ein anderer Zyklus von demselben Master anhängig ist. Falls ein Multi-Threaded-Verhalten in einem Master beobachtet wird, dann wird ein solcher durch Einstellen des entsprechenden Bits CFG2Q_MULTI_MASTER[X] auf hoch markiert.

[0348] Der Eingang des Status-Registers **326** wird mit dem Ausgang eines 14 : 7 Multiplexers **328** verbunden, dessen 0-Eingang mit dem Ausgang eines 14 : 7 Multiplexers **330** verbunden ist, dessen 1-Eingang mit Adressen-Bits P2Q_AD[22:16] verbunden ist. Ein Auswahl-Signal CFGWR_MM wählt die 0- und 1-Eingänge des Multiplexers **328** aus. Wenn hoch aufgestellt ist, bewirkt das Signal CFGWR_MM ein Konfigurations-Schreiben des Status-Registers **326** von den Daten-Bits P2Q_AD[22:16], was eine Software-Steuerung der Bits in dem Register **326** ermöglicht. Der 1-Eingang des Multiplexers **330** nimmt Multi-Master-Signale MULTI_MASTER[7:1] auf, der 0-Eingang empfängt den Ausgang des Registers **326** und der Multiplexer **330** wird durch ein Signal MULTI_SEL ausgewählt. Das Signal MULTI_SEL wird durch ein UND-Gate **338** erzeugt,

dessen erster Eingang ein Signal Q2PIF_CHECK_CYC aufnimmt (gesetzt auf hoch, um anzuzeigen, dass die momentanen Transaktions-Informationen gegenüber von Informationen geprüft werden sollten, die in dem Warteschlangen-Block **127** gespeichert sind, und zwar hinsichtlich einer Anpassung, wie beispielsweise während einer verzögerten Speicher-Lese- oder Schreib-Anforderung von einer Bus-Vorrichtung auf dem sekundären PCI-Bus **32**), und der andere Eingang nimmt den invertierten Zustand des Signals DCQ_HIT auf (was anzeigt, dass die momentenen Adressen-Informationen nicht die Adressen-Informationen anpassen, die einer anhängigen Anforderung des anfordernden Master in der DCQ **148** zugeordnet sind). Demzufolge wird, falls ein fehlerhafter Vergleich auftrat, der Wert der Signale CFG2Q_MULTI_MASTER[7:1] aktualisiert.

[0349] Ein Bit MULTI_MASTER[X] wird auf hoch gesetzt, falls der Master X eine anhängige Anforderung hat, die erneut versucht worden ist, und der Master X darauffolgend zurück zu einer unterschiedlichen Anforderung kommt. Dies wird durch Vergleichen der Transaktions-Informationen (z. B. Adresse, Byte-Freigaben, Daten für ein Schreiben) der anhängigen Anforderung mit der Adresse der neuen Anforderung geprüft. Ein fehlgeschlagener Vergleich zeigt an, dass der Master im Multi-Threaded-Betrieb ist. Wenn einmal ein Multi-Master-Konfigurations-Bit CFG2Q_MULTI_MASTER[X] (X = 1–7) auf hoch gesetzt ist, wird das Bit auf hoch beibehalten.

[0350] Die Signale MULTI_MASTER[7:1] werden durch einen Decodierer **336** erzeugt. Der Decodieren **336** nimmt Signale Q2PIF_SLOT[2:0] (Schlitz-Zahl für die momentane, verzögerte Anforderung von einem Master), Q[7:0]_MASTER[2:0] (der Master, zugeordnet zu jedem der acht Puffer in der DCQ **148**), Q[7:0]_COMPLETE (der Abschluss-Status jeder der acht Warteschlangen), und Q[7:0]_PART_COMPLETE (der Teil-Abschluss-Status jedes der Puffer in der verzögerten Abschluss-Warteschlange) auf. Zum Beispiel zeigt dann, falls das Signal Q0_MASTER[2:0] den Wert 4 enthält, dies an, dass der DCQ-Puffer 0 die Transations-Informationen einer verzögerten Anforderung von der Bus-Vorrichtung in dem Schlitz **4** speichert. Das Signal QY_COMPLETE, Y = 0–7, zeigt an, falls „hoch" aufgestellt ist, ob der DCQ-Puffer Y alle die Daten empfangen hat, die der verzögerten Anforderungs-Transaktion zugeordnet sind. Das Signal QY_PART_COMPLETE, Y = 0–7, zeigt an, falls hoch aufgestellt ist, dass der DCQ-Puffer Y als der DCQ-Puffer für eine verzögerte Transaktion von einem der Master zugeordnet worden ist, allerdings alle die Daten, die der verzögerten Transaktion zugeordnet sind, noch nicht empfangen worden sind.

[0351] Falls die momentane Schlitz-Zahl Q2PIF_SLOT[2:0] gleich zu dem Wert irgendeines der acht Warteschlangen-Master-Indikations-Signale Q[7:0]_MASTER[2:0] ist, und sich der entsprechende DCQ-Puffer in dem Abschluss- oder Teil-Abschluss-Zustand befindet, dann wird das entsprechende eine der Bits MULTI_MASTER[7:1] auf hoch gesetzt, falls das Signal DCQ_HIT niedrig ist, und das Signal Q2PIF_CHECK_CYC[2:0] hoch ist. Demzufolge wird, zum Beispiel, falls das Signal Q2PIF_SLOT[2:0] den Wert 2 enthält, was anzeigt, dass die Vorrichtung im Schlitz **2** der momante Master der verzögerten Anforderung ist, und der DCQ-Puffer **5** eine anhängige Anforderung für den Schlitz-2-Master speichert (Q5_MASTER[2:0] = 5), und irgendeines der Signale Q5_COMPLETE oder Q5_PART_COMPLETE hoch ist, und falls das Signal Q2PIF_CHECK_CYC hoch ist und das Signal DCQ_HIT niedrig ist, dann das Bit MULTI_MASTER[2] auf hoch gesetzt, um so anzuzeigen, dass die Schlitz-2-Vorrichtung ein Multi-Threaded-Master ist.

[0352] Ein Maskierungs-Anforderungs-Erzeugungs-Block **332** erzeugt Signale Q2A_MASK_REQ[X] (X = 1–7) in Abhängigkeit von Signalen Q[7:0]_MASTER[2:0], Q[7:0]_STATE[3:0], (was den Zustand von verzögerten Abschluss-Warteschlangen 0–7 anzeigt), SLOT_WITH_DATA[7:0] (was anzeigt, ob ein verzögerter Abschluss Qs 0–7 gültige Daten enthält), CFG2Q_MULTIMASTER[X] (X = 1–7), CFG2Q_ALWAYS_MASK und CFG2Q_NEVER_MASK.

[0353] Wie Fig. 26A zeigt, umfasst der Masken-Anforderungs-Erzeugungs-Block **332** einen 2 : 1 Multiplexer **320** zum Erzeugen des Signals Q2A_MASK_REQ[X] (X = 1–7). Der 1-Eingang des Multiplexers **320** ist mit dem Ausgang eines ODER-Gates **322** verbunden und der 0-Eingang ist auf niedrig gesetzt. Der Auswahl-Eingang des Multiplexers **320** wird durch ein Signal MASK_MUXSEL angesteuert. Ein Eingang des ODER-Gates **322** ist mit dem Ausgang eines NOR-Gates **324** verbunden, der ein Signal CFG2Q_MULTI_MASTER[X] aufnimmt (anzeigend einen Multi-Threaded-Master), und der andere Eingang empfängt ein Signal CFG2Q_NEVER_MASK (ein Konfigurations-Bit, das anzeigt, dass die Anforderungs-Zeile nicht maskiert werden sollte, falls ein Multi-Threaded-Master erfasst ist). Der andere Eingang des ODER-Gates **322** empfängt ein Signal CFG2Q_ALWAYS_MASK, das ein Konfigurations-Bit ist, das anzeigt, dass das entsprechende Masken-Bit Q2A_MASK_REQ[X] immer maskiert werden sollte, falls das Signal MUXSEL auf hoch gesetzt ist. Das Signal MASK_MUXSEL wird auf hoch gesetzt, falls die Anforderung von dem sekundären Bus-Master nicht zu Daten vorliegt, die bereits in dem Warteschlangen-Block **127** existieren, d. h. die Anforderung muss zu dem primären PCI-Bus **24** übertragen werden. Demzufolge wird, zu jedem Zeitpunkt, zu dem eine Anforderung von

einer Vorrichtung auf dem sekundären PCI-Bus **32** eingangsseitig zu dem primären PCI-Bus **24** übertragen wird, eine Prüfung in Bezug auf Bits CFG2Q_MULTI_MASTER[7:1] durchgeführt werden, um zu bestimmen, ob ein Multi-Threaded-Master erfasst worden ist.

[0354] Die Maskierung von Anforderungen kann durch Einstellen der geeigneten Bits in den Konfigurations-Registern **125** übergangen werden. Die verfügbaren Moden umfassen 1) normaler Mode, in dem die Anforderungs-Maskierung freigegeben ist, mit der Ausnahme, falls ein Multi-Threaded-Master (CFG2Q_NEVER_MASK = 0, CFG2Q_ALWAYS_MASK = 0), 2) immer ein Maskierungs-Mode, in dem Anforderungen von erneut versuchenden Mastern maskiert werden, gerade wenn Multi-Threaded (CFG2Q_ALWAYS_MASK = 1) vorliegt, und 3) niemals Maskierungs-Mode, in dem die Anforderungen niemals maskiert sind (CFG2Q_NEVER_MASK = 1, CFG2Q_ALWAYS_MASKED = 0).

ERWEITERUNGS-KARTEN-EINSETZEN UND ENTFERNEN VON VERBINDUNGS-EXPANSIONS-KARTEN

[0355] Wie in den Fig. 1 und Fig. 27A dargestellt ist, besitzen die zwei Expansionskästen **30a** und **30b**, von einem gemeinsamen Design **30**, jeweils die sechs Hot-Plug-Schlitze **36** (**36a–f**), in denen die herkömmlichen Erweiterungskarten **807** eingesetzt und entfernt werden können (Hot-Plugged), während das Computersystem **10** hochgefahren bzw. eingeschaltet verbleibt. Die sechs mechanischen Hebel **802** werden dazu verwendet, selektiv die Expansionskarten **807** zu sichern (wenn geschlossen, oder verriegelt, ist), die in die entsprechenden Hot-Plug-Schlitze **36** eingesetzt werden. Zu Zwecken eines Entfernens oder eines Einsetzens der Expansionskarten **807** in einen der Schlitze **36** muss der entsprechende Hebel **802** geöffnet werden, oder entriegelt werden, und so lange wie der Hebel **802** geöffnet ist, verbleibt der entsprechende Schlitz **36** abgeschaltet.

[0356] Wenn der Hebel **802**, der die Expansionskarte **807** an seinem Schlitz **36** sichert, geöffnet ist, erfasst das Computersystem **10** dieses Auftreten und fährt die Karte **802** herunter (und den entsprechenden Schlitz **36**), bevor die Karte **807** von deren Schlitz bzw. Einsteckplatz **36** entfernt werden kann. Schlitze bzw. Einsteckplätze **36**, die heruntergefahren sind, ähnlich anderen Schlitzen **36**, die keine Karten **807** halten, verbleiben heruntergefahren bzw. abgeschaltet, bis eine Software des Computersystems **10** selektiv die Schlitze bzw. Einsteckplätze **36** hochfährt.

[0357] Die Karte **46**, eingesetzt in den Kartenschlitz **34**, besitzt den Brücken-Chip **48**, der den Sicherungs-Status (offen oder geschlossen) der Hebel **802** überwacht und irgendeine Karte **807** (und einen entsprechenden Schlitz **36**) herunterfährt, der nicht durch seinen Hebel **802** gesichert ist. Eine Software des Computersystems **10** kann auch selektiv irgendeinen der Schlitze **36** herunterfahren.

[0358] Die Karten **807** werden durch eine Hochfahrsequenz hochgefahren und durch eine Herunterfahrsequenz heruntergefahren. In der Hochfahrsequenz (the power up sequence) wird Energie zuerst zu der Karte **807** zugeführt, die hochgefahren werden soll, und danach wird ein PCI-Takt-Signal (von dem PCI-Bus **32**) zu der Karte **807** geliefert, die hochgefahren wird. Verbleibende PCI-Bus-Signal-Leitungen der Karte **807** werden dann mit entsprechenden Leitungen des PCI-Busses **32** verbunden. Zuletzt wird das Reset-Signal für die Karte **807**, die hochgefahren wird, weggenommen, was die Karte **807** außerhalb eines Reset-Zustands bringt.

[0359] Die Hochfahrsequenz ermöglicht der Schaltung der Karte **807**, dass sie hochgefahren wird, um vollständig funktional mit dem PCI-Takt-Signal zu werden, bevor die verbleibenden PCI-Bus-Signale geliefert werden. Wenn das Taktsignal und die verbleibenden PCI-Bus-Signale mit der Karte **807** verbunden werden und bevor die Karte **807** zurückgesetzt wird, besitzt der Brücken-Chip **48** eine Kontrolle des PCI-Busses **32**. Da der Brücken-Chip **48** eine Kontrolle über den PCI-Bus **32** während dieser Zeiten besitzt, stören potentielle Defekte auf dem PCI-Bus **32** von der Power-up-Sequenz nicht die Operationen der Karten **807**, die hochgefahren sind.

[0360] In der Herunterfahr- bzw. Power-Down-Sequenz wird die Karte **807**, die heruntergefahren werden soll, zuerst zurückgesetzt. Als nächstes werden die PCI-Bus-Signale, ohne das PCI-Takt-Signal, von der Karte **807** entfernt. Der Brücken-Chip **48** unterbricht darauffolgend das PCI-Takt-Signal von der Karte **807**, bevor Energie von der Karte **807** entfernt wird. Die Herunterfahr- bzw. Power-Down-Sequenz minimiert die Propagation von falschen Signalen von der Karte **807**, die heruntergefahren werden soll, zu dem Bus **32**, da die Schaltung auf der Karte **807** deren vollständige Funktion beibehält, bis die PCI-Bus-Signal-Leitungen entfernt sind.

[0361] Wenn das PCI-Taktsignal und die verbleibenden PCI-Bus-Signale unterbrochen werden, und wenn die Karte **807** zurückgesetzt bzw. in einen Reset-Zustand versetzt wird, besitzt der Brücken-Chip **48** eine Kontrolle des PCI-Busses **32**. Da der Brücken-Chip **48** eine Kontrolle über den PCI-Bus **32** während dieser Zeitpunkte besitzt, stören potentielle Defekte auf dem PCI-Bus **32** von der Power-Down- bzw. Herunterfahr-Sequenz nicht

die Operationen der Karten **807**, die sie hochgefahren haben.

[0362] Der Brücken-Chip **48** umfasst die Seriell-Eingangs/Ausgangs-(SIO)-Schaltung **50**, die die Hochfahr- und Herunterfahr-Sequenzen der Schlitze bzw. Einsteckplätze **36** über vierundzwanzig Steuersignale POUT[39:16] steuert. Die Steuersignale POUT[39:16] sind ein Untersatz von vierzig Ausgangs-Steuersignalen POUT[39:0], erzeugt durch die SIO-Schaltung **50**. Die Steuersignale POUT[39:16] sind verriegelte Versionen von Schlitz-Bus-Freigabe-Signalen BUSEN#[5:0], Schlitz-Energie-Freigabe-Signalen PWREN[5:0], Schlitz-Takt-Freigabe-Signalen CLKEN#[5:0] und Schlitz-Reset-Signalen RST#[5:0], alle internen Signale der SIO-Schaltung **50**, wie weiter nachfolgend beschrieben ist. Die Steuersignale POUT[39:0] und deren Beziehung zu den Signalen BUSEN#[5:0], PWREN[5:0], CLKEN#[5:0] und RST#[5:0] sind in der nachfolgenden Tabelle beschrieben:

PARALLEL-AUSGANGS-STEUER-SIGNALE (POUT[39:01])

SIGNAL-POSITION	BESCHREIBUNG	ZUGEORDNETE STEUER-SIGNALE	WENN EIN SIGNAL AKTIV IST
0-11	Steuersignale für LEDs 54		
12-15	Ausgangssignale für allgemeine Zwecke	GPOA[3:0]	
16	Reset-Signal für Schlitz 36a	(RST#[0]	niedrig
17	Reset-Signal für Schlitz 36b	RST#[1]	niedrig
18	Reset-Signal für Schlitz 36c	(RST#[2]	niedrig
19	Reset-Signal für Schlitz 36d	RST#[3]	niedrig
20	Reset-Signal für Schlitz 36e	(RST#[4]	niedrig
21	Reset-Signal für Schlitz 36f	RST#[5]	niedrig
22	Takt-Freigabe-Signal für Schlitz 36a	(CLKEN#[0]	niedrig
23	Takt-Freigabe-Signal für Schlitz 36b	(CLKEN#[1]	niedrig
24	Takt-Freigabe-Signal für Schlitz 36c	(CLKEN#[2]	niedrig
25	Takt-Freigabe-Signal für Schlitz 36d	(CLKEN#[3]	niedrig
26	Takt-Freigabe-Signal für Schlitz 36e	(CLKEN#[4]	niedrig
27	Takt-Freigabe-Signal für Schlitz 36f	(CLKEN#[5]	niedrig
28	Bus-Freigabe-Signal für Schlitz 36a	(BUSEN#[0]	niedrig
29	Bus-Freigabe-Signal für Schlitz 36b	(BUSEN#[1]	niedrig
30	Bus-Freigabe-Signal für Schlitz 36c	(BUSEN#[2]	niedrig
31	Bus-Freigabe-Signal für Schlitz 36d	(BUSEN#[3]	niedrig
32	Bus-Freigabe-Signal für Schlitz 36e	(BUSEN#[4]	niedrig
33	Bus-Freigabe-Signal für Schlitz 36f	(BUSEN#[5]	niedrig
34	Power-Freigabe-Signal für Schlitz 36a	(PWREN[0]	hoch
35	Power-Freigabe-Signal für Schlitz 36b	(PWREN[1]	hoch
36	Power-Freigabe-Signal für Schlitz 36c	(PWREN[2]	hoch
37	Power-Freigabe-Signal für Schlitz 36d	(PWREN[3]	hoch
38	Power-Freigabe-Signal für Schlitz 36e	(PWREN[4]	hoch
39	Power-Freigabe-Signal für Schlitz 36f	(PWREN[5]	hoch

[0363] Wie in den Fig. 2 und Fig. 28 dargestellt ist, besitzt jeder Hot-Plug-Schlitz **36** die zugeordnete Um-

schalt-Schaltung **41** zum Verbinden und Trennen des Schlitzes **36** mit und von dem PCI-Bus **32**. Die Umschalt-Schaltung **41** für jeden Schlitz **36** empfängt vier der Steuersignale POUT[39:16]. Als ein Beispiel wird, für den Schlitz **36a**, wenn das Steuersignal POUT[28] aufgestellt ist, oder niedrig ist, der Schlitz **36a** mit den Bus-Signal-Leitungen des PCI-Busses **32** durch eine Umschalt-Schaltung **47** verbunden. Wenn das Steuersignal POUT[28] weggenommen ist, oder hoch ist, wird der Schlitz **36a** von den Bus-Signal-Leitungen des PCI-Busses **32** getrennt.

[0364] Wenn das Steuersignal POUT[22] aufgestellt ist, oder niedrig ist, wird der Schlitz **36a** mit einem PCI-Takt-Signal CLK über eine Umschalt-Schaltung **43** verbunden. Wenn das Steuersignal POUT[22] weggenommen ist, oder hoch ist, wird der Schlitz **36a** von dem Taktsignal CLK getrennt.

[0365] Wenn das Steuersignal POUT[34] aufgestellt ist, oder hoch ist, wird der Schlitz **36a** mit einem Karten-Spannungs-Versorgungs-Pegel V_{SS} über eine Umschalt-Schaltung **45** verbunden. Wenn das Steuersignal POUT[34] weggenommen ist, oder niedrig ist, wird der Schlitz **36a** von dem Karten-Spannungs-Versorgungs-Pegel V_{SS} getrennt.

[0366] Wenn das Steuersignal POUT[16] aufgestellt ist oder niedrig ist, wird der Schlitz **36a** zurückgesetzt, und wenn das Steuersignal POUT[16] weggenommen ist, oder hoch ist, gelangt der Schlitz **36a** aus seinem Reset-Zustand.

[0367] Wie in Fig. 2 zu sehen ist, kann die SIO-Schaltung **50** selektiv bis zu einhundertachtundzwanzig (sechzehn Bytes) von verriegelten Status-Signalen STATUS[127:0), geliefert durch den Erweiterungskasten **30**, überwachen. Die Status-Signale STATUS[127:0] bilden einen „Snapshot" von ausgewählten Zuständen des Erweiterungskastens **30**. Die Status-Signale STATUS[127:0] umfassen sechs Status-Signale STATUS[5:0], die den Sicherungs-Status (geöffnet oder geschlossen) jedes der Hebel **802** anzeigen. Die SIO-Schaltung **50** überwacht die Status-Signale STATUS[31:0] hinsichtlich Änderungen in deren logischen Spannungspegeln. Die SIO-Schaltung **50** verschiebt seriell die Status-Signale STATUS[127:32] in die SIO-Schaltung **50** hinein, wenn durch die CPU **14** angewiesen ist, dies so vorzunehmen.

[0368] Die SIO-Schaltung **50** empfängt seriell die Status-Signale STATUS[127:0], das am wenigsten signifikanteste Signal zuerst, und zwar über ein serielles Daten-Signal NEW_CSID. Das Daten-Signal NEW_CSID wird durch den seriellen Ausgang des parallelen Eingangs-Verschieberegisters **82** mit zweiunddreißig Bits, angeordnet auf einer Leiterplatte des Erweiterungskastens **30**, zusammen mit den Schlitzen bzw. Einsteckplätzen **36**, geliefert.

[0369] Das Register **82** empfängt, über dessen parallele Eingänge, vierundzwanzig Parallel-Status-Signale PIN[23:0], vier zugeordnet zu jedem der Hot-Plug-Schlitze **36**, die in den zweiunddreißig am wenigsten signifikanten Status-Signalen STATUS[31:0] umfasst sind. Wenn sich der Status, angezeigt durch eines oder mehrere der Status-Signale STATUS[31:0], ändert (der logische Spannungs-Pegel ändert sich), erzeugt der Brücken-Chip **48** eine Unterbrechungs-Anforderung zu der CPU **14** durch Aufstellen, oder Ansteuern auf niedrig, eines seriellen Unterbrechungs-Anforderungs-Signals SI_INTR#, das über einen Unterbrechungs-Empfangs-Block **132** empfangen wird. Die Status-Signale PIN[23:0] umfassen zwei PCI-Karten-Präsenz-Signale (PRSNT1# und PRSNT2#), zugeordnet zu jedem Schlitz **36**.

[0370] Sechs Status-Signale PIN[5:0], entsprechend zu deren verriegelten Versionen, Status-Signale STATUS[5:0], zeigen den Sicherungs- oder Eingriffs-Status (offen oder geschlossen) jedes der Hebel **802** an. Sechs Gleit-Schalter **805** (Fig. 27A–Fig. 27C) werden durch die Bewegung deren entsprechender Hebel **802** betätigt und werden dazu verwendet, elektrisch den Sicherungs-Status des entsprechenden Hebels **802** anzuzeigen. Jeder Schalter **805** besitzt ein erstes Terminal, verbunden mit Masse, und ein zweites Terminal, das das entsprechende eine der Status-Signale PIN[5:0] zuführt. Das zweite Terminal ist mit einem Versorgungs-Spannungs-Pegel V_{DD} über einen von sechs Widerständen **801** verbunden.

[0371] Falls sich einer der Hebel **802** öffnet und die Karte **807**, gesichert durch den Hebel **802**, entsichert wird, wird das entsprechende eine der Status-Signale PIN[5:0] aufgestellt, oder auf hoch angesteuert. Als ein Beispiel wird, für den Schlitz **36a**, das Status-Signal PIN[0] weggenommen oder auf niedrig angesteuert, wenn der entsprechende Hebel **802** geschlossen ist. Wenn der Hebel **802** für den Schlitz **36a** geöffnet ist, wird das Status-Signal PIN[0] aufgestellt, oder auf hoch angesteuert.

[0372] Das Register **82** empfängt auch eine serielle Datenfolge von verriegelten Status-Signalen STATUS[127:32], die keine Unterbrechungen verursachen, wenn sich der logische Spannungs-Pegel eines der Si-

gnale STATUS[127:32] ändert. Die Status-Signale STATUS[127:32] werden durch das Verschiebe-Register **52** mit sechzehn Bits, angeordnet auf der Leiterplatte des Expansionskastens **30**, mit den Schlitzen **36**, gebildet. Das Verschiebe-Register **52** empfängt Status-Signale an seinen parallelen Eingängen und verriegelt die Status-Signale STATUS[127:32], wenn durch die SIO-Schaltung **50** instruiert ist, dies so vorzunehmen. Das Verschiebe-Register **52** serialisiert die Status-Signale STATUS[127:32] und liefert die Signale STATUS[127:32] zu dem seriellen Eingang des Registers **82** über ein serielles Daten-Signal CSID_I.

[0373] Wenn durch die SIO-Schaltung **50** instruiert ist, verriegelt das Register **82** Status-Signale PIN[23:0], bildet die Status-Signale STATUS[31:0], liefert die Status-Signale STATUS[31:0] und liefert ein Byte oder mehr der Status-Signale STATUS[127:32] (wenn dies durch die CPU **14** angefordert ist), in einem am wenigsten signifikanten Signal einer ersten Art, zu der SIO-Schaltung **50**, und zwar über das serielle Daten-Signal NEW_CSID. Die Status-Signale STATUS[127:0] werden durch die nachfolgende Tabelle beschrieben:

STATUS[127:0]

BIT	BESCHREIBUNG
0	Status-Signal von Hebel 802 für Schlitz 36a (PIN[0])
1	Status-Signal von Hebel 802 für Schlitz 36b (PIN[1])
2	Status-Signal von Hebel 802 für Schlitz 36c (PIN[2])
3	Status-Signal von Hebel 802 für Schlitz 36d (PIN[3])
4	Status-Signal von Hebel 802 für Schlitz 36e (PIN[4])
5	Status-Signal von Hebel 802 für Schlitz 36f (PIN[5])
6	reserviert für Status-Signal von Hebel 802 für einen zusätzlichen Hot-Plug-Schlitz
7	reserviert für Status-Signal von Hebel 802 für einen zusätzlichen Hot-Plug-Schlitz
8	PRSNT2# Signal für Schlitz 36a (PIN[6])
9	PRSNT2# Signal für Schlitz 36b (PIN[7])
10	PRSNT2# Signal für Schlitz 36c (PIN[8])
11	PRSNT2# Signal für Schlitz 36d (PIN[9])
12	PRSNT2# Signal für Schlitz 36e (PIN[10])
13	PRSNT2# Signal für Schlitz 36f (PIN[11])
14	reserviert für PRSNT#2 Signal für einen zusätzlichen Hot-Plug-Schlitz 36

15	reserviert für PRSNT#2 Signal für einen zusätzlichen Hot-Plug-Schlitz 36
16	PRSNT1# Signal für Schlitz 36a (PIN[12])
17	PRSNT1# Signal für Schlitz 36b (PIN[13])
18	PRSNT1# Signal für Schlitz 36c (PIN[14])
19	PRSNT1# Signal für Schlitz 36d (PIN[15])
20	PRSNT1# Signal für Schlitz 36e (PIN[16])
21	PRSNT1# Signal für Schlitz 36f (PIN[17])
22	reserviert für PRSNT#1 Signal für einen zusätzlichen Hot-Plug-Schlitz 36
23	reserviert für PRSNT#1 Signal für einen zusätzlichen Hot-Plug-Schlitz 36
24	Power-Fehler-Status für Schlitz 36a (PIN[18])
25	Power-Fehler-Status für Schlitz 36b (PIN[19])
26	Power-Fehler-Status für Schlitz 36c (PIN[20])
27	Power-Fehler-Status für Schlitz 36d (PIN[21])
28	Power-Fehler-Status für Schlitz 36e (PIN[22])
29	Power-Fehler-Status für Schlitz 36f (PIN[23])
30	reserviert für Power-Fehler-Status für zusätzlichen Hot-Plug-Schlitz 36
31	reserviert für Power-Fehler-Status für zusätzlichen Hot-Plug-Schlitz 36
32-127	Status-Sginale, die keine Unterbrechungs-Anforderungen verursachen, wenn sich deren Status ändert

[0374] Wie in den Fig. 2 und Fig. 30 dargestellt ist, verriegelt, wenn eine SIO-Schaltung **50** ein Register-Lade-Signal CSIL_O_ aufstellt, oder auf niedrig ansteuert, das Schiebe-Register **52** die Status-Signale STATUS[127:32] und das Schiebe-Register **82** verriegelt die Status-Signale STATUS[31:0]. Wenn die SIO-Schaltung **50** das Signal CSIL_O_ wegnimmt, oder auf hoch ansteuert, verschieben beide Register **52** und **82** seriell deren Daten zu der SIO-Schaltung **50** an der positiven Flanke des Taktsignals CSIC_O, geliefert durch die SIO-Schaltung **50**. Das Taktsignal CSIC_O wird zu der und auf einem Viertel der Frequenz des PCI-Taktsignals CLK synchronisiert.

[0375] Wie in Fig. 29 dargestellt ist, verwendet, zu Zwecken einer Überwachung, oder für ein Abtasten, der Status-Signale STATUS[31:0], die SIO-Schaltung **50** ein 32-Bit-Unterbrechungs-Register **800**, dessen Bit-Positionen den Signalen STATUS[31:0] entsprechen. Die SIO-Schaltung **50** aktualisiert die Bits des Unterbrechungs-Registers **800**, um die entsprechenden Status-Signale STATUS[31:0] anzugleichen, die entprellt (debounced) worden sind, wie weiter nachfolgend beschrieben ist. Zwei Status-Signale STATUS[7:6] werden für zusätzliche Hot-Plug-Schlitze **36** reserviert, und das siebte und achte, signifikanteste Bit des Unterbrechungs-Registers **800** werden auch für die zusätzlichen Schlitze **36** reserviert. Das Unterbrechungs-Register **800** ist ein Teil eines Register-Logik-Blocks **808** der SIO-Schaltung **50**, die mit dem PCI-Bus **32** gekoppelt ist.

[0376] Eine serielle Abtast-Eingangs-Logik **804** der SIO-Schaltung **50** tastet sequenziell, oder überwacht, die Status-Signale STATUS[31:0], das am wenigsten signifikante Signal zuerst, hinsichtlich Änderungen, wie dies durch Übergänge in deren logischen Spannungs-Pegeln angezeigt ist. Falls sich der Status von einem oder mehr der Status-Signale STATUS[5:0], zugeordnet den Hebeln **802**, ändert, tritt die Seriell-Abtast-Eingangs-Logik **804** in einen langsamen Abtast-Mode ein, so dass die Status-Signale STATUS[5:0] zweiunddreißigmal innerhalb eines vorbestimmten Entprell- bzw. Debounce-Zeit-Intervalls abgetastet werden. Falls sich eines oder mehrere der Status-Signale STATUS[5:0] ändert, aktualisiert die serielle Abtast-Eingangs-Logik

804 das Unterbrechungsregister **800** (und stellt das serielle Unterbrechungs-Signal SI_INTR# auf), falls das geänderte Status-Signal STATUS[5:0] auf demselben, logischen Spannungsniveau für zumindest ein vorbestimmtes Entprell-Zeit-Intervall verbleibt. Die Seriell-Abtast-Eingangs-Logik **804** ist mit programmierbaren Zeitgebern **806** gekoppelt, die das Ende des Entprell-Verzögerungs-Intervalls erzeugen und anzeigen, initiiert durch die Seriell-Abtast-Logik **804**. Unter Fordern des Status, stabil für die Entprell-Zeit zu bleiben, minimiert das Intervall das unbeabsichtigte Power-Down von einem der Hot-Plug-Schlitze **36** aufgrund eines falschen Werts (d. h. eines „Defekts"), angezeigt durch eines der Status-Signale STATUS[5:0]. Wenn alle der Status-Signale STATUS[5:0] auf demselben, logischen Spannungspegel für mindestens das Entprell-Zeit-Intervall verbleiben, dann schreitet die Seriell-Abtast-Eingangs-Logik **804** fort, um noch einmal erneut alle zweiunddreißig Status-Signale STATUS[31:0] in dem schnelleren Abtast-Mode abzutasten.

[0377] Falls die Seriell-Abtast-Eingangs-Logik **804** eine Änderung in einem der Status-Signale STATUS[31:6] erfasst, instruiert die Seriell-Abtast-Eingangs-Logik **804** die Zeitgeber **806**, ein anderes Debounce- bzw. Entprell-Verzögerungs-Intervall zu messen, stellt darauf das Seriell-Unterbrechungs-Signal SI_INTR# auf, aktualisiert das Unterbrechungs-Register **800** mit den Signalen STATUS[31:6], die sich geändert haben, und ignoriert weitere Änderungen in den Status-Signalen STATUS[31:6], bis das Entprell-Zeit-Intervall abläuft. Nach Ablaufen des Entprell-Zeit-Intervalls schreitet die Seriell-Abtast-Eingangs-Logik **804** fort, um Änderungen in den zweiunddreißig Status-Signalen STATUS[31:0] zu erkennen.

[0378] Wenn das Seriell-Unterbrechungs-Signal SI_INTR# aufgestellt ist, liest die CPU **14** darauffolgend das Unterbrechungs-Register **800**, bestimmt, welche (es können mehr als eins sein) Status-Signale STATUS[3:0] die Unterbrechung verursachten, und nimmt das Seriell-Unterbrechungs-Signal SI_INTR# durch Schreiben einer „1" zu dem Bit oder den Bits des Unterbrechungs-Registers **800**, die sich geändert haben, weg.

[0379] Die CPU **14** kann selektiv Unterbrechungs-Anforderungen, verursacht durch die Status-Signale STATUS[31:0], durch Schreiben einer „1" zu einem entsprechenden Bit eines Unterbrechungs-Maskierungs-Registers **810** mit zweiunddreißig Bits maskieren. Die CPU **14** kann auch selektiv irgendein Byte der Status-Signale STATUS[47:0] durch Schreiben einer Byte-Zahl des ausgewählten Bytes zu einem Seriell-Eingangs-Byte-Register **812** lesen. Die SIO-Schaltung **50** überträgt dann das erwünschte Byte in ein Seriell-Daten-Register **815** hinein.

[0380] Zum Beispiel schreibt, um das dreißigste Byte (Byte-Zahl zwei) der Status-Signale STATUS[23:16] zu lesen, die CPU **14** eine „2" in das Seriell-Eingangs-Byte-Register **812**. Die Seriell-Abtast-Eingangs-Logik **804** verschiebt dann seriell Byte zwei der Status-Signale STATUS[23:16] in das Seriell-Daten-Register **815** hinein. Ein Busy-Status-Bit BS des Seriell-Eingangs-Byte-Registers **812** ist gleich zu „1", wenn die CPU **14** zu Anfang die erwünschte Byte-Zahl zu dem Seriell-Eingangs-Byte-Register **812** schreibt. Das Bit BS wird durch die SIO-Schaltung **50** gelöscht, nachdem das angeforderte Byte in das Seriell-Daten-Register **815** hinein verschoben worden ist.

[0381] Die CPU **14** kann einen der Schlitze **36** durch Schreiben einer „1" zu einem entsprechenden Bit eines Schlitz-Freigabe-Registers **817** hochfahren und den Schlitz **36** durch Schreiben einer „0" zu diesem Bit sperren. Weiterhin kann die CPU **14** einen der Schlitze **36** durch Schreiben einer „1" zu einem entsprechenden Bit eines Schlitz-Reset-Registers **819** zurücksetzen. Die Inhalte der Schlitz-Freigabe- **817** und der Schlitz-Reset- **819** Register sind durch Signale SLOT_EN[5:0] und SLOT_RST_[5:0] jeweils dargestellt.

[0382] Um die Anforderung zu initiieren, angezeigt durch das Schlitz-Freigabe- **817** und Reset- **819** Register, zu initiieren, schreibt die CPU **14** eine „1" zu einem SO-Bit eines Steuer-Registers **814**. Nachdem das SO-Bit aufgestellt ist (was ein GO_UPDATE Signal aufstellt, oder auf hoch ansteuert), initiiert die SIO-Schaltung **50** die erforderlichen Power-down- und/oder Power-up-Sequenzen und steuert sie.

[0383] Die Seriell-Abtast-Eingangs-Logik **804** ist mit einer EIN/AUS-Steuer-Logik **820** verbunden, die die Power-up- und Power-down-Sequenzen steuert. Die EIN/AUS-Steuer-Logik **820** liefert die Signale BUSEN#[5:0], CLKEN#[5:0], RST#[5:0] und PWREN[5:0] zu einer Seriell-Ausgangs-Logik **824**.

[0384] Jede Power-up- und Power-down-Sequenz umfasst vier Verschiebe-Phasen, während denen ein anderer Schritt der Power-down- oder Power-up-Sequenz durchgeführt wird. Während jeder Verschiebe-Phase instruiert die EIN/AUS-Steuer-Logik 820 die Seriell-Ausgangs-Logik **824**, die Steuer-Signale BUSEN#[5:0], CLKEN#[5:0], RST#[5:0] und PWREN[5:0] zu kombinieren; diese Signale zu verriegeln; und liefert seriell diese Signale (über ein serielles Daten-Signal CSOD_O) zu dem seriellen Eingang eines Ausgangs-Verschiebe-Registers **80**. An dem Ende jeder Verschiebe-Phase instruiert die EIN/AUS-Steuer-Logik **820** das Verschiebe-Re-

gister **80**, die Steuersignale POUT[35:12] zu aktualisieren.

[0385] Die EIN/AUS-Steuer-Logik **820** wird auch schnittstellenmäßig mit der Register-Logik **808** und einer Steuer-Logik **822** für eine Licht emittierende Diode (Light Emitting Diode – LED) verbunden. Die LED-Steuer-Logik **122** steuert den Ein/Aus-Status der sechs LEDs **54**, die visuell anzeigen, ob die entsprechenden Hebel **802** verriegelt sind oder entriegelt sind. Die LEDs **54** können so programmiert werden, um dann zu blinken, wenn sie eingeschaltet sind, und zwar über LED-Steuerregister (nicht dargestellt) der Register-Logik **808**.

[0386] Wie in **Fig. 31A** dargestellt ist, umfasst die Seriell-Abtast-Eingangs-Logik **804** eine Abtast-Zustand-Maschine **840**, die das Abtasten der Status-Signale STATUS[31:0] hinsichtlich Änderungen steuert und das Verschieben eines ausgewählten Bytes der Status-Signale STATUS[47:0] in das Seriell-Eingangs-Byte-Register **815** hinein steuert.

[0387] Die Abtast-Zustand-Maschine **840** wird an der negativen Flanke eines Takt-Signals DIV2CLK getaktet, das zu einem PCI-Taktsignal CLK synchronisiert ist, und von einer Hälfte der Frequenz des PCI-Taktsignals CLK ist. Die Last- und Taktsignale, CSIL_O_ und CSIC_O, jeweils, werden durch die Abtast-Zustand-Maschine **840** geliefert. Das Taktsignal wird, wenn es freigegeben wird, zu dem Taktsignal CSIC_O synchronisiert.

[0388] Ein Bit/Byte-Zähler **841** zeigt, über ein Zweiunddreißig-Bit-Signal BIT ACTIVE[31:0] an, welches Bit der Status-Signale STATUS[3:0] momentan durch das Seriell-Daten-Signal NEW_CSID repräsentiert wird. Das aufgestellte Bit des Signals BIT_ACTIVE[31:0] besitzt dieselbe Bit-Position wie das Status-Signal STATUS[31:0], dargestellt durch das Daten-Signal NEW_CSID.

[0389] Der Zähler **841** liefert auch ein Drei-Bit-Signal BIT[2:0], das darstellt, welches Bit des momentanen Bytes der Status-Signale STATUS[31:0] momentan durch die Abtast-Zustand-Maschine **840** abgetastet wird. Der Zähler **841** wird an der negativen Flanke eines Signals SHIFT_ENABLE getaktet. Die Ausgänge des Zählers **841** werden zurückgesetzt, oder gelöscht, wenn der Ausgang eines UND-Gates **842**, verbunden mit dem Lösch-Eingang des Zählers **840**, negiert wird.

[0390] Die Abtast-Zustand-Maschine **840** liefert ein Signal SCAN_IN_IDLE, das, wenn es aufgestellt wird, oder auf hoch gesetzt wird, anzeigt, dass sich die Abtast-Zustand-Maschine **840** in einem IDLE Zustand befindet, und momentan nicht irgendeines der Status-Signale STATUS[127:0] abtastet. Das Signal SCAN_IN_IDLE wird ansonsten weggenommen.

[0391] Das Signal SCAN_IN_IDLE wird zu einem Eingang des UND Gates **842** geliefert. Der andere Eingang des UND-Gates **842** ist mit dem Ausgang eines ODER-Gates **843** verbunden. Ein Eingang des ODER-Gates **843** empfängt ein invertiertes HOLD_OFF Signal, und der andere Eingang des ODER-Gates **843** empfängt ein Signal GETTING_BYTE.

[0392] Das Signal HOLD_OFF zeigt, wenn es aufgestellt ist, oder auf hoch angesteuert ist, an, dass eine Änderung in dem einen der Status-Signale STATUS[5:0] erfasst worden ist, und die Seriell-Abtast-Logik **804** in den Langsam-Abtast-Mode eingetreten ist. In diesem Langsam-Abtast-Mode bzw. Slow-Scan-Mode wartet die Seriell-Abtast-Eingangs-Logik **804** auf ein vorbestimmtes Langsam-Abtast-Intervall, bevor die Status-Signale STATUS[31:0] erneut weitergeführt werden. Die Seriell-Abtast-Eingangs-Logik **804** zählt die Zahl von Malen, für die die Seriell-Abtast-Signale STATUS[5:0] während des Langsam-Abtast-Modes abgetastet werden, und verwendet diese Zählung, um zu bestimmen, wenn eines von dem Status-Signal STATUS[5:0] unverändert für das Entprell-Verzögerungs-Intervall verblieb, wie weiterhin nachfolgend beschrieben werden wird.

[0393] Deshalb werden, wenn sich die Abtast-Zustand-Maschine **840** in dem IDLE Zustand befindet und entweder das HOLD_OFF Signal weggenommen ist oder sich die Abtast-Zustand-Maschine **840** beim Lesen eines ausgewählten Bytes (ausgewählt durch die CPU **14**) der Status-Signale STATUS[47:0] befindet, alle Ausgänge des Zählers **841** gelöscht oder gleich zu Null gesetzt.

[0394] Das Signal SHIFT_ENABLE wird durch den Ausgang eines UND-Gates **844** geliefert. Ein Eingang des UND-Gates **844** nimmt das Takt-Signal CSIC_O auf. Ein anderer Eingang des UND-Gates **844** nimmt ein Signal DIV2CLK# auf. Das Signal DIV2CLK# wird aufgestellt, oder auf niedrig angesteuert, an der negativen Flanke des Signals CLKDIV4. Der dritte Eingang des UND-Gates **844** empfängt ein Signal SCAN_IN_PROGRESS, das, wenn es aufgestellt wird, oder auf hoch angesteuert ist, anzeigt, dass die Abtast-Zustand-Maschine **840** momentan die Status-Signale STATUS[127:0] abtastet, und das Signal SCAN_IN_PROGRESS wird ansonsten weggenommen.

[0395] Deshalb wird, wenn die Abtast-Zustand-Maschine **840** nicht die Status-Signale STATUS[127:0] verschiebt, der Zähler **841** gesperrt. Weiterhin wird, wenn freigegeben ist, der Zähler **841** an der negativen Flanke des Taktsignals DIV2CLK getaktet.

[0396] Das Unterbrechungs-Register **800** empfängt Eingangs-Signale D_INTR_REG[31:0] an deren entsprechenden zweiunddreißig Eingängen. Die Last-Freigabe-Eingänge des Unterbrechungs-Registers **800** nehmen entsprechende Last-Freigabe-Signale UPDATE_IRQ[31:0] auf. Das Unterbrechungs-Register **800** wird an der positiven Flanke des PCI-Taktsignals CLK getaktet.

[0397] Für die Zwecke, ein Protokoll über die Status-Signale STATUS[5:0] nach jeder Abtastung beizubehalten, liefert ein Mehrfach-Bit, D-Typ-Flip-Flop **836** Status-Signale SCAN_SW[5:0]. Der Lösch-Eingang des Flip-Flops **836** nimmt das Reset-Signal RST auf, und das Flip-Flop **836** wird an der positiven Flanke des Taktsignals CLK getaktet. Der Eingang des Flip-Flops **836** ist mit dem Ausgang eines Mehrfach-Bit-ODER-Gates **850** verbunden, das einen Eingang mit dem Ausgang eines Multi-Bit-UND-Gates **846** verbunden und einen Eingang mit dem Ausgang eines Multi-Bit-UND-Gates **847** verbunden besitzt. Ein Eingang des UND-Gates **846** nimmt sechs Bit-Freigabe-Signale BIT_ENABLE[5:0] auf (nachfolgend beschrieben) und der andere Eingang des UND-Gates **846** nimmt das Seriell-Daten-Signal NEW_CSID auf. Ein Eingang des UND-Gates **847** nimmt invertierte Bit-Freigabe-Signale BIT_ENABLE[5:0] auf, und der andere Eingang des UND-Gates **847** nimmt die Signale SCAN_SW[5:0] auf.

[0398] Nur eines der Bit-Freigabe-Signale BIT_ENABLE[5:0] wird zu einem Zeitpunkt aufgestellt (wenn die Abtast-Zustand-Maschine **840** abtastet), und das aufgestellte Bit zeigt an, welches eine der entsprechenden Status-Signale STATUS[31:0] durch das Signal NEW_CSID dargestellt wird. Demzufolge werden, wenn die Abtast-Zustand-Maschine **840** abtastet, und zwar an jeder positiven Flanke des Taktsignals CLK, die Signale SCAN_SW[5:0] aktualisiert.

[0399] Die Bit-Freigabe-Signale BIT_ENABLE[31:0] werden durch den Ausgang eines Mehrfach-Bit-Multiplexers **832** geliefert, der die Bits BIT_ACTIVE[31:0] an seinem einen Eingang empfängt. Der Null-Eingang des Multiplexers **832** empfängt ein Zweiunddreißig-Bit-Signal, das für eine logische Null Indikativ ist. Der Auswahl-Eingang des Multiplexers **832** empfängt das Signal SHIFT_ENABLE.

[0400] Zu Zwecken einer Erfassung einer Änderung in den Status-Signalen STATUS[5:0], liefert ein Multi-Bit-, Exklusiv-Oder-(XOR)-Gate **848** Umschalt-Änderungs-Signale SW_CHG[5:0]. Wenn eines der Signale SW_CHG[5:0] aufgestellt ist oder hoch ist, änderte sich die logische Spannung des entsprechenden Status-Signals STATUS[5:0] während aufeinanderfolgender Abtastungen. Ein Eingang des XOR-Gates **848** ist mit dem Eingang des Flip-Flops **836** verbunden, und der andere Eingang des XOR-Gates **848** empfängt die Signale SCAN_SW[5:0].

[0401] Wie in **Fig. 31D** dargestellt ist, besitzt, zu Zwecken eines Anzeigens, wenn der logische Spannungs-Pegel eines ausgewählten Status-Signals STATUS[5:0] bei einem logischen Spannungs-Pegel für mindestens die Dauer des Debounce-Verzögerungs-Intervalls verblieben war, die Abtast-Eingangs-Logik **804** sechs Signale LSWITCH[5:0]. Der nicht-invertierende Eingang eines Flip-Flops **900** vom D-Typ liefert das Signal LSWITCH[5] an seinem nicht-invertierenden Ausgang. Das Signal LSWITCH[5] wird aufgestellt oder auf hoch angesteuert, um den vorstehend beschriebenen Zustand anzuzeigen, und wird ansonsten weggenommen. Das Flip-Flop **900** wird an der positiven Flanke des Taktsignals CLK getaktet, und der Lösch-Eingang des Flip-Flops **900** nimmt das RST-Signal auf.

[0402] Der Eingang des Flip-Flops **900** ist mit dem Ausgang eines Multiplexers **902** verbunden, der ein D_LSWITCH[5] Signal liefert. Der Auswahl-Eingang des Multiplexers **902** ist mit dem Ausgang eines UND-Gates **903** verbunden, der ein MAX5 Signal und ein SCAN_END Signal aufnimmt. Das SCAN_END Signal zeigt, wenn es aufgestellt ist, an, dass die Abtast-Zustand-Maschine **840** deren momentane Abtastung abgeschlossen hat. Fünf Signale (MAX5, MAX4, MAX3, MAX2, MAX1 und MAX0) zeigen an, ob das entsprechende Status-Signal STATUS[5], STATUS[4], STATUS[3], STATUS[2], STATUS[1] oder STATUS[0], jeweils, auf demselben, logischen Spannungs-Niveau für mindestens die Dauer des Debounce-Zeit-Intervalls verblieben ist. Der Null-Eingang des Multiplexers **902** empfängt das Signal LSWITCH[5], und der eine Eingang des Multiplexers **902** empfängt das Signal SCAN_SW[5]. Das Signal SCAN_END wird durch den Ausgang eines UND-Gates **851** geliefert (**Fig. 31B**). Das UND-Gate **851** empfängt ein Signal STOP_SCAN und ein Signal SCAN_DONE. Das Signal STOP_SCAN wird aufgestellt, oder auf hoch angesteuert, wenn Zustände zum Beenden des Abtastens durch die Abtast-Zustand-Maschine **840** vorhanden sind, wie weiter nachfolgend beschrieben ist. Das Signal SCAN_END ist eine gepulste Version (für einen Zyklus des CLK-Signals) des Signals

STOP_SCAN. Die Signale LSWITCH[4]–LSWITCH[0] und D_LSWITCH[4]–D_LSWITCH[0] werden in einer ähnlichen Weise aus den jeweiligen SCAN_SW[4]–SCAN_SW[0] Signalen und den jeweiligen Signalen MAX4–MAX0 erzeugt.

[0403] Zu Zwecken einer Aktualisierung wird der logische Spannungspegel der Status-Signale STATUS[31:6] als diese Signale eingetastet, ein Multi-Bit-D-Typ-Flip-Flop **905** (Fig. 31D) liefert sechsundzwanzig Signale SCAN_NSW[31:6]. Eines der Signale SCAN_NSW[31:6] wird aufgestellt, oder auf hoch angesteuert, um diesen Zustand anzuzeigen, und wird ansonsten weggenommen. Das Flip-Flop **905** wird an der positiven Flanke des Taktsignals CLK getaktet und der Löscheingang des Flip-Flops **905** nimmt das RST-Signal auf.

[0404] Der Eingang des Flip-Flops **905** ist mit dem Ausgang eines Multi-Bit-Multiplexers **906** verbunden. Der Auswahl-Eingang des Multiplexers **906** nimmt ein invertiertes CHECK_SWITCH_ONLY Signal auf. Das CHECK_SWITCH_ONLY Signal wird aufgestellt, oder auf hoch angesteuert, wenn die Abtast-Zustand-Maschine **850** nur die Status-Signale STATUS[5:0] oder die Status-Signale STATUS[127:32] abtastet (d. h. Änderungen in den Signalen STATUS[31:6] ignorieren), und sie ansonsten wegnimmt. Der Null-Eingang des Multiplexers **906** empfängt die Signale SCAN_NSW[31:6], und der eine Eingang des Multiplexers **906** ist mit dem Ausgang eines Multi-Bit-ODER-Gates **907** verbunden. Ein Eingang des ODER-Gates **907** ist mit dem Ausgang eines Multi-Bit-UND-Gates **908** verbunden, und der andere Eingang des ODER-Gates **907** ist mit dem Ausgang eines Multi-Bit-UND-Gates **872** verbunden.

[0405] Ein Eingang des UND-Gates **908** empfängt die Signale BIT_ENABLE[31:6]. Der andere Eingang des UND-Gates **908** ist mit dem Ausgang eines Multi-Bit-Multiplexers **909** verbunden. Falls das NEW_CSID Signal aufgestellt ist, oder hoch ist, liefert der Multiplexer **909** ein Signal mit sechsundzwanzig Bits gleich zu „h3FFFFFF". Ansonsten liefert der Multiplexer ein Signal mit sechsundzwanzig Bits gleich zu „0". Ein Eingang des UND-Gates **872** ist mit dem invertierten Ausgang des UND-Gates **908** verbunden und der andere Eingang des UND-Gates **872** nimmt die Signale SCAN_NSW[31:6] auf.

[0406] Zu Zwecken eines Speicherns des logischen Spannungs-Pegels der Status-Signale STATUS[31:6] nach jeder Abtastung liefert ein Multi-Bit-D-Typ-Flip-Flop **871** sechsundzwanzig Signale LNON_SW[31:6]. Eines der Signale LNON_SW[31:6] wird aufgestellt, oder auf hoch gesetzt, um diesen Zustand anzuzeigen, und wird ansonsten weggenommen. Das Flip-Flop **871** wird auf der positiven Flanke des Taktsignals CLK getaktet, und der Lösch-Eingang des Flip-Flops **871** empfängt das RST-Signal.

[0407] Der Eingang des Flip-Flops **871** ist mit dem Ausgang eines Multi-Bit-Multiplexers **870** verbunden, der die Signale D_LNON_SW[31:6] liefert. Der Auswahl-Eingang des Multiplexers **870** empfängt das Signal SCAN_END. Der Null-Eingang des Multiplexers **870** empfängt die Signale LNON_SW[31:6), und der eine Eingang des Multiplexers **807** empfängt die Signale SCAN_NSW[31:6].

[0408] Wie in Fig. 31B dargestellt ist, umfasst, zu Zwecken eines Erzeugens der MAX0, MAX1, MAX2, MAX3, MAX4 und MAX5 Signale, die Seriell-Eingangs-Logik **804** sechs Zähler **831a–f**, jeweils, von einem gemeinsamen Design **831**. Jeder Zähler **831** wird initialisiert (auf einen vorbestimmten Zähl-Wert), wenn ein UND-Gate **892** seinen Ausgang aufstellt, oder auf hoch ansteuert. Für den Zähler **831a** empfängt das UND-Gate **892** das Signal BIT_ENCABLE[0], das Signal SW_CHG[0] und ein invertiertes Signal QUICK_FILTER. Das Signal QUICK_FILTER kann, wenn es aufgestellt ist, oder hoch ist, dazu verwendet werden, das Debounce-Zeit-Intervall zu umgehen. Das QUICK_FILTER Signal wird normalerweise weggenommen oder auf niedrig gesetzt. Der Takt-Eingang des Zählers **831** ist mit dem Ausgang eines UND-Gates **893** verbunden. Für den Zähler **831a** empfängt das UND-Gate **893** das BIT_ENABLE[0] Signal, das invertierte SW_CHG[0] Signal, das invertierte GETTING_BYTE Signal und das invertierte MAX0 Signal. Deshalb wird, für den Zähler **831a**, wenn sich einmal die logische Spannung des Status-Signals STATUS[0] ändert, zu jedem Zeitpunkt, zu dem die Seriell-Abtast-Logik **804** das Status-Signal STATUS[0] abtastet, der Zähler **831a** erhöht. Wenn der Zähler **831a** seinen maximalen Wert erreicht, wird das Signal MAX0 aufgestellt, was anzeigt, dass das Debounce-Zeit-Interwall abgelaufen ist. Falls sich die logische Spannung des Status-Signals STATUS[0] während der Zählung ändert, wird der Zähler **831a** reinitialisiert, und die Zählung beginnt erneut. Die anderen Zähler **831b–f** arbeiten in einer ähnlich Weise in Bezug auf deren entsprechende Status-Signale STATUS[5:1].

[0409] Das HOLD_OFF Signal instruiert, wenn es aufgestellt ist, einen der Zeitgeber **806**, ein vorbestimmtes Langsam-Abtast-Intervall zu messen, das die serielle Abtast-Zustand-Maschine **840** in den Langsam-Abtast-Mode versetzt. Wenn der Zeitgeber **806** eine Messung dieses Verzögerungs-Intervalls abschließt, stellt der Zeitgeber **806** ein FTR_TIMEOUT Signal auf, oder steuert es auf hoch an, das ansonsten weggenommen wird, oder negiert wird. Das Produkt dieses Langsam-Abtast-Intervalls und der Zahl von Zählungen für den

Zähler **831**, um seinen maximalen Wert zu erreichen, ist gleich zu dem Debounce-Zeit-Intervall (8 ms).

[0410] Das HOLD_OFF Signal wird durch den Ausgang eines JK-Flip-Flops **885** geliefert. Das Flip-Flop **885** wird an der positiven Flanke des CLK Signals getaktet, und der Lösch-Eingang des Flip-Flops **885** empfängt das RST-Signal. Der J-Eingang ist mit dem Ausgang eines UND-Gates **883** verbunden und der K-Eingang ist mit dem Ausgang eines UND-Gates **884** verbunden. Ein Eingang des UND-Gates **884** ist mit dem Ausgang eines Flip-Flops **896** vom JK-Typ verbunden, und der andere Eingang des UND-Gates **893** empfängt das SCAN_END Signal. Ein Eingang des UND-Gates **884** ist mit dem invertierten Ausgang des UND-Gates **883** verbunden, ein Eingang des UND-Gates **884** empfängt das FTR_TIMEOUT Signal, und ein anderer Eingang des UND-Gates **884** empfängt ein SCAN_IN_IDLE Signal, das aufgestellt wird, wenn sich die Abtast-Zustand-Maschine **840** in deren IDLE Zustand befindet, wie dies weiter nachfolgend beschrieben ist.

[0411] Das Flip-Flop **895** wird an der positiven Flanke des CLK Signals getaktet und der Lösch-Eingang des Flip-Flops **895** empfängt das RST-Signal. Der J-Eingang ist mit dem Ausgang eines NAND-Gates **894** verbunden, der die MAX0, MAX1, MAX2, MAX3, MAX4 und MAX5 Signale aufnimmt. Der K-Eingang ist mit dem Ausgang eines UND-Gates **826** verbunden, der mit dem invertierten J-Eingang des Flip-Flops **895** verbunden ist, und empfängt ein invertiertes SCAN_IN_PROGRESS Signal, das dann aufgestellt wird, wenn die Abtast-Zustand-Maschine **840** die Status-Signale STATUS[31:0] abtastet.

[0412] Zu Zwecken einer Erzeugung des CHECK_SWITCH_ONLY Signals umfasst die Seriell-Abtast-Eingangs-Logik **804** ein Flip-Flop **864** vom JK-Typ, das das CHECK_SWITCH_ONLY Signal an dem nicht-invertierendem Ausgang liefert, und wird an der positiven Flanke des CLK-Signals getaktet. Der Lösch-Eingang des Flip-Flops **864** empfängt das RST-Signal, und der J-Eingang des Flip-Flops **864** empfängt ein DEBOUNCE Signal, das, wenn es aufgestellt, oder auf hoch angesteuert, ist, anzeigt, dass sich einer des logischen Spannungs-Pegels eines oder mehrere der Status-Signale STATUS[31:6] geändert hat. Der K-Eingang des Flip-Flops **864** ist mit dem Ausgang eines UND-Gates **865** verbunden. Ein Eingang des UND-Gates **865** nimmt das invertierte DEBOUNCE Signal auf und ein Eingang des UND-Gates **865** nimmt das SCAN_IN_IDLE Signal auf.

[0413] Wie in **Fig. 31C** dargestellt ist, wird das Debounce-Signal DEBOUNCE durch den nicht-invertierenden Ausgang eines Flip-Flops **860** vom JK-Typ geliefert. Das Flip-Flop **860** wird durch die positive Flanke des Taktsignals CLK getaktet, und der Lösch-Eingang des Flip-Flops **860** empfängt das Reset-Signal RST. Der J-Eingang des Flip-Flops **860** empfängt ein Signal CHANGE_ON_INPUT. Das Signal CHANGE_ON_INPUT wird aufgestellt, oder auf hoch angesteuert, wenn eine Änderung in einem der Status-Signale STATUS[31:6] an dem Ende einer Abtastung durch die Seriell-Eingangs-Logik **804** erfasst wird, und wird ansonsten weggenommen. Der K-Eingang ist mit dem Ausgang eines UND-Gates **861** verbunden, das ein DB_TIMEOUT Signal an einem seiner Eingänge aufnimmt. Der andere Eingang des UND-Gates **861** nimmt das invertierte CHANGE_ON_INPUT Signal auf. Das DB_TIMEOUT Signal wird durch die Zeitgeber **106** für einen Zyklus des CLK-Signals aufgestellt, wenn die Debounce-Zeit-Verzögerung (initiiert durch das Aufstellen des DEBOUNCE Signals) abgelaufen ist. Das Aufstellen des DB_TIMEOUT Signals negiert das DEBOUNCE Signal an der nächsten, positiven Flanke des CLK-Signals.

[0414] Das CHANGE_ON_INPUT Signal wird durch den nicht-invertierenden Ausgang eines Flip-Flops **866** vom JK-Typ geliefert, das an der positiven Flanke des CLK-Signals getaktet wird. Der Lösch-Eingang des Flip-Flops empfängt das RST-Signal. Der J-Eingang des Flip-Flops **866** ist mit dem Ausgang eines UND-Gates **869** verbunden, das das SCAN_END Signal aufnimmt, und der andere Eingang des UND-Gates **869** ist mit dem Ausgang eines ODER-Gates **867** verbunden. Das ODER-Gate **867** verknüpft logisch ODER-mäßig alle eines Satzes von NSW_CHG[31:6] Signalen. Die Bit-Positionen der Signale NSW_CHG[31:6] entsprechen den Bit-Positionen der Status-Signale STATUS[31:6] und zeigen an, durch deren Aufstellen, ob sich das entsprechende Status-Signal STATUS[31:6] nach der letzten Abtastung geändert hat. Das UND-Gate **869** nimmt weiterhin das SCAN_END Signal auf. Der K-Eingang des Flip-Flops **866** ist mit dem Ausgang eines UND-Gates **868** verbunden, der das invertierte SCAN_IN_PROGRESS Signal und den invertierten Ausgang des UND-Gates **869** aufnimmt. Die Signale NSW_CHG[31:6] werden durch den Ausgang eines Multi-Bit-XOR-Gates **862** geliefert, das die Signale D_LNON_SW[31:6] und LNON_SW[31:6] aufnimmt.

[0415] Der nicht-invertierende Ausgang eines Multi-Bit-D-Typ-Flip-Flops **912** liefert Bits SI_DATA[7:0] für das Seriell-Daten-Register **815**. Der Lösch-Eingang des Flip-Flops **912** empfängt das Signal RST und das Flip-Flop **912** wird an der positiven Flanke des CLK-Signals getaktet. Der Signal-Eingang des Flip-Flops **912** ist mit dem Ausgang eines Multi-Bit-Multiplexers **916** verbunden. Der Auswahl-Eingang des Multiplexers **916** ist mit dem Ausgang eines UND-Gates **914** verbunden, und der Null-Eingang des Multiplexers **916** nimmt die Bits

SI_DATA[7:0] auf. Das UND-Gate **914** nimmt die Signale GETTING_BYTE und SHIFT_ENABLE auf. Demzufolge werden, wenn die Seriell-Abtast-Logik **804** nicht ein angefordertes Byte der Status-Signale STATUS[47:0] verschiebt, die Werte der Bits SI_DATA[7:0] bewahrt.

[0416] Der eine Eingang des Multiplexers **916** ist mit dem Ausgang eines Multi-Bit-Multiplexers **910** verbunden. Der eine Eingang des Multiplexers **910** ist mit dem Ausgang eines Multi-Bit-ODER-Gates **911** verbunden, und der Null-Eingang des Multiplexers ist mit dem Ausgang eines Multi-Bit-UND-Gates **915** verbunden. Der Auswahl-Eingang des Multiplexers **910** empfängt das Signal NEW_CSID.

[0417] Ein Eingang des UND-Gates **915** empfängt die Bits SI_DATA[7:0], und ein invertierender Eingang des UND-Gates **915** ist mit dem Ausgang eines 3 × 8 Decodierers **913** verbunden. Der Decodierer **913** empfängt das Signal BIT[2:0]. Ein Eingang des ODER-Gates **911** empfängt die Bits SI_DATA[7:0], und der andere Eingang des ODER-Gates **911** empfängt den Ausgang des Decodierers **913**.

[0418] Die Seriell-Eingangs-Logik **804** liefert fünf Signale RST_SWITCH[5:0] (entsprechend den Bit-Positionenen der Status-Signale STATUS[5:0]) zu der EIN/AUS-Steuer-Logik **820**, was, durch deren Aufstellen, anzeigt, ob der entsprechende Schlitz **36a–f** heruntergefahren werden sollte. Die EIN/AUS-Steuer-Logik **820** zeigt an, wenn der Schlitz **36** (angezeigt durch die RST_SWITCH[5:0] Signale) durch das darauffolgende Einstellen eines von fünf Signalen CLR_SWITCH[5:0] heruntergefahren worden ist, deren Bit-Positionen den Signalen RST_SWITCH[5:0] entsprechen. Nach Empfangen der Anzeige, dass der Schlitz bzw. Einsteckplatz **36** heruntergefahren worden ist, nimmt die serielle Logik **804** dann das entsprechende RST_SWITCH[5:0] Signal zurück.

[0419] Die Signale RST_SWITCH[5:0] werden durch den nicht-invertierenden Ausgang eines Multi-Bit-Flip-Flops **891** vom D-Typ (Fig. 31B) geliefert. Der Lösch-Eingang des Flip-Flops **891** empfängt das Reset-Signal RST und das Flip-Flop **891** wird an der positiven Flanke des Taktsignals CLK getaktet. Der Eingang des Flip-Flops **891** ist mit dem Ausgang eines Multi-Bit-ODER-Gates **857** verbunden, der einen Eingang mit dem Ausgang eines Multi-Bit-UND-Gates **859** verbunden besitzt und einen Eingang mit dem Ausgang eines Multi-Bit-UND-Gates **855** verbunden besitzt. Ein Eingang des UND-Gates **859** ist mit dem Ausgang eines Multiplexers **853** verbunden, und der andere Eingang des UND-Gates **859** empfängt verriegelte Schlitz-Freigabe-Signale LSLOT_EN[5:0], die anzeigen, durch deren Aufstellen, ob der entsprechende Schlitz bzw. Einsteckplatz **36a–f** hochgefahren ist. Ein Eingang des UND-Gates **855** nimmt die Signale CLR_SWITCH_[5:0] auf. Ein anderer Eingang des UND-Gates **855** nimmt die Signale RST_SWITCH[5:0] auf. Ein anderer Eingang des UND-Gates **855** ist mit dem invertierten Ausgang des Multiplexers **853** verbunden.

[0420] Der Null-Eingang des Multiplexers **853** empfängt ein Sechs-Bit-Signal, das für Null Indikativ ist. Der eine Eingang des Multiplexers **853** ist mit dem Ausgang eines Multi-Bit-UND-Gates **849** verbunden. Ein Eingang des UND-Gates **849** empfängt die Signale D_LSWITCH[5:0], und der andere Eingang des UND-Gates **849** empfängt die invertierten Signale L_SWITCH[5:0]. Der Auswahl-Eingang des Multiplexers **853** empfängt das SCAN_END Signal.

[0421] Zu Zwecken einer Erzeugung des SI_INTR# Signals umfasst die Seriell-Abtast-Logik **804** ein Flip-Flop **882** vom D-Typ, das das Seriell-Unterbrechungs-Signal SI_INTR# an seinem invertierenden Ausgang liefert. Das Flip-Flop **882** wird an der positiven Flanke des CLK-Signals getaktet, und der Löscheingang des Flip-Flops **882** empfängt das RST-Signal. Der Eingang des Flip-Flops **882** ist mit dem Ausgang eines ODER-Gates **881** verbunden, der zweiunddreißig anhängige Unterbrechungs-Signale PENDING_IRQ[31:0] aufnimmt, die, durch deren Aufstellen, oder Ansteuern auf hoch, anzeigen, ob eine Unterbrechung für das entsprechende eine der Status-Signale STATUS[31:0] anhängig ist. Die Signale PENDING_IRQ[31:0] werden ansonsten weggenommen.

[0422] Wie in Fig. 31E dargestellt ist, liefert ein Multi-Bit-Flip-Flop **979** vom D-Typ die Signale PENDING_IRQ[31:0] an seinem nicht-invertierenden Ausgang. Das Flip-Flop **979** wird an der positiven Flanke des Signals CLK getaktet und empfängt das Signal RST an seinem Lösch-Eingang. Der Eingang des Flip-Flops **979** ist mit dem Ausgang eines Multi-Bit-UND-Gates **981** verbunden, das invertierte Unterbrechungs-Maskierungs-Signale INTR_MASK[31:0] an einem Eingang aufnimmt. Die Signale INTR_MASK[31:0] sind für ein entsprechendes Bit des Unterbrechungs-Masken-Registers **810** Indikativ. Der andere Eingang des UND-Gates **981** ist mit dem Ausgang eines Multi-Bit-ODER-Gates **835** verbunden. Ein Eingang des ODER-Gates **835** ist mit dem Ausgang eines Multi-Bit-UND-Gates **862** verbunden und der andere Eingang des ODER-Gates **835** ist mit dem Ausgang eines Multi-Bit-UND-Gates **834** verbunden.

[0423] Das UND-Gate **862** nimmt invertierte PENDING_IRQ[31:0] Signale auf und signalisiert SET_IRQ[31:0]. Die Signale SET_PIRQ[31:0] werden aufgestellt, um anzuzeigen, dass eine Unterbrechungs-Anforderung für das entsprechende eine der Status-Signale STATUS[31:0] erzeugt werden sollte. Deshalb werden die Signale PENDING_IRQ[31:0] mit den Signalen SET_PIRQ[31:0] aktualisiert, falls sie nicht durch die Signale INTR_MASK[31:0] maskiert sind.

[0424] Das UND-Gate **834** empfängt die Signale PENDING_IRQ[31:0], invertierte Signale SET_PIRQ[31:0] und invertierte WR_INTR_REG[31:0] Signale. Die Signale WR_INTR_REG[31:0] zeigen die Schreib-Daten an, geliefert durch die CPU **14**, und zwar zu dem Unterbrechungs-Register **800** hin. Die CPU löscht eine Unterbrechung durch Schreiben einer „1" zu dem entsprechenden Bit des Unterbrechungs-Registers **800**. Deshalb wird, falls dies auftritt, und keine neuen Unterbrechungs-Anforderungen für das entsprechende eine der Status-Signale STATUS[31:0] angezeigt werden, das entsprechende eine der Signale PENDING_IRQ[31:0] gelöscht.

[0425] Die Signale SET_PIRQ[31:0] werden durch den Ausgang eines Multi-Bit-UND-Gates **839** geliefert. Ein Eingang des UND-Gates **839** empfängt die Signale UPDATE_IRQ[31:0]. Der andere Eingang des UND-Gates **839** ist mit dem Ausgang eines Multi-Bit-XOR-Gates **837** verbunden. Ein Eingang des XOR-Gates **837** empfängt die Signale D_INTR_REG[31:0], der andere Eingang des XOR-Gates **837** empfängt die Signale INTR_REG[31:0]. Deshalb wird, wenn die Bits des Unterbrechungs-Registers **800** von einem logischen Zustand zu einem anderen übergehen, eine Unterbrechungs-Anforderung erzeugt.

[0426] Zu Zwecken einer Aktualisierung der Bits des Unterbrechungs-Registers **800** werden die Signale UPDATE_IRQ[31:0] zu den entsprechenden Last-Eingängen des Registers **800** geliefert. Wenn eines der Signale UPDATE_IRQ[31:0] aufgestellt ist, oder auf hoch angesteuert ist, wird das entsprechende Bit mit dem entsprechenden einen der Signale D_INTR_REG[31:0] geladen.

[0427] Die Signale UPDATE_IRQ[31:0] werden durch den Ausgang eines Multi-Bit-ODER-Gates **971** geliefert. Ein Eingang des ODER-Gates **971** ist mit dem Ausgang eines Multi-Bit-UND-Gates **973** verbunden. Ein Eingang des UND-Gates **973** ist mit dem Ausgang eines Multi-Bit-Multiplexers **977** verbunden, und der andere Eingang des UND-Gates **973** nimmt invertierte PENDING_IRQ[31:0] Signale auf. Der Auswahl-Eingang des Multiplexers **977** empfängt das Signal SCAN END, der eine Eingang des Multiplexers **977** empfängt ein Zweiunddreißig-Bit-Signal, indikativ für „hFFFFFFFF", und der Null-Eingang des Multiplexers **977** empfängt ein Zweiunddreißig-Bit-Signal, indikativ für „0". Deshalb ermöglichen, an dem Ende einer Abtastung, die Signale UPDATE_IRQ[31:0], dass die Bits des Unterbrechungs-Registers **800** aktualisiert werden, die den aufgestellten PENDING_IRQ[31:0] Signalen entsprechen.

[0428] Ein anderer Eingang des ODER-Gates **971** ist mit dem Ausgang eines Multi-Bit-UND-Gates **975** verbunden. Ein Eingang des UND-Gates **975** empfängt die invertierten INTR_MASK[31:0] Signale, ein anderer Eingang des UND-Gates **975** empfängt die Signale PENDING_IRQ[31:0], und der andere Eingang des UND-Gates **975** empfängt die Signale WR_INTR_REG[31:0]. Deshalb kann die CPU **14** selektive Bits der Signale PENDING_IRQ[31:0] löschen.

[0429] Die Signale D_INTR_REG[5:0] werden durch den Ausgang eines Multi-Bit-Multiplexers **830** geliefert. Wenn das SCAN_END Signal aufgestellt ist, sind die Signale D_INTR_REG[5:0] gleich zu den Signalen D_LSWITCH[5:0]. Wenn das SCAN_END Signal weggenommen ist, sind die Signale D_INTR_REG[5:0] gleich zu den Signalen LSWITCH[5:0].

[0430] Die Signale D_INTR_REG[31:6] werden durch den Ausgang eines Multi-Bit-Multiplexers **845** geliefert. Wenn das SCAN_END Signal aufgestellt ist, sind die Signale D_INTR_REG[31:6] gleich zu den Signalen D_LNON_SW[31:6]. Wenn das SCAN_END Signal weggenommen ist, sind die Signale D_INTR_REG[5:0] gleich zu den Signalen LNON_SW[31:6]. Das Unterbrechungs-Register **800** nimmt neue Werte nur dann auf, wenn das Signal SCAN_END aufgestellt ist.

[0431] Wie in den Fig. 32A–B dargestellt ist, tritt die Abtast-Zustand-Maschine **840** in einen IDLE Zustand nach dem Aufstellen des RESET-Signals RST ein. Wenn sie sich nicht in dem IDLE Zustand befindet, toggelt die Abtast-Zustand-Maschine **840** die Zustände des Seriell-Eingangs-Taktsignals CSIC_O, um das Schiebe-Register **82** zu takten. Weiterhin stellt, wenn sie sich nicht in einem ersten Lade-Zustand LD1 befindet, die Abtast-Zustand-Maschine **840** das Lade-Signal CSIL_O_ auf oder steuert es auf hoch an, um die Register **82** und **52** freizugeben, um seriell die Status-Signale STATUS[127:0] zu der SIO-Schaltung **50** zu verschieben. In dem IDLE Zustand setzt die Abtast-Zustand-Maschine **840** das Signal SCAN_DONE gleich zu Null.

[0432] Die Abtast-Zustand-Maschine **840** geht von dem IDLE Zustand zu dem Zustand LD1 über, wenn entweder das Signal GETTING_BYTE aufgestellt ist oder das Signal HOLD_OFF weggenommen ist. Ansonsten verbleibt die Abtast-Zustand-Maschine **840** in dem IDLE Zustand. In dem LD1 Zustand stellt die Abtast-Zustand-Maschine **840** das Lade-Signal CSIL_O_ auf oder steuert es auf niedrig an, das die Register **82** und **52** freigibt, um zu verriegeln und damit zu beginnen, die Status-Signale STATUS[127:0] aufzunehmen.

[0433] Die Abtast-Zustand-Maschine **840** geht von dem LD1 Zustand zu einem Lade-Zwei-Zustand LD2 über. In dem LD2 Zustand wird das Lade-Signal CSIL_O_ aufgestellt beibehalten, was den Registern **82** und **52** ermöglicht, seriell die Status-Signale STATUS[127:0] zu verschieben.

[0434] Die Abtast-Zustand-Maschine **840** geht darauffolgend zu einem Abtast-Zustand SCAN über. In dem SCAN Zustand tastet die Seriell-Abtast-Eingangs-Logik **804** die Status-Signale STATUS[127:0] an jeder negativen Flanke des Takt-Signals DIV2CLK ab.

[0435] Wenn das Signal STOP_SCAN aufgestellt ist, geht die Abtast-Zustand-Maschine **840** zurück zu dem IDLE Zustand. Das STOP_SCAN Signal wird aufgestellt, wenn entweder das erwünschte Byte der Status-Signale STATUS[127:0] in das Seriell-Daten-Register **815** hinein verschoben worden ist; die Hebel-Status-Signale STATUS[5:0] eingetastet worden sind und das Seriell-Unterbrechungs-Signal SI_INTR# aufgestellt worden ist; oder alle Status-Signale STATUS[31:0] hinein verschoben worden sind. In dem SCAN Zustand wird das SCAN_DONE Signal gleich zu dem STOP SCAN Signal gesetzt.

[0436] Wie in Fig. 33A dargestellt ist, umfasst die EIN/AUS-Steuer-Logik **820** eine EIN/AUS-Zustand-Maschine **998**, die die RST_SWITCH[5:0] Signale, SLOT_EN[5:0] und SLOT_RST_[5:0] empfängt. Basierend auf den Zuständen, angezeigt durch diese Signale, zeigt die EIN/AUS-Zustand-Maschine **998** die geeigneten Hochfahr- und Herunterfahr-Sequenzen an und steuert sie. Die EIN/AUS-Zustand-Maschine **998** liefert Steuersignale zu der Steuer-Logik **999**.

[0437] Die EIN/AUS-Zustand-Maschine **998** liefert ein Seriell-Ausgangs-Aktualisierungs-Signal SO_UPDATE zu der Seriell-Ausgangs-Logik **824**. Wenn das Signal SO_UPDATE aufgestellt ist, oder auf hoch angesteuert ist, beginnt die Seriell-Ausgangs-Logik **824** die Verschiebe-Phase und verschiebt Seriell-Steuer-Daten, über das Signal CSOD_O, zu dem Register **80**. Die Seriell-Ausgangs-Logik **824** zeigt einen Abschluss der Verschiebe-Phase durch Aufstellen eines Signals SO_UPDATE_DONE an, das durch die EIN/AUS-Zustand-Maschine **998** empfangen wird. Die EIN/AUS-Zustand-Maschine **998** aktualisiert darauffolgend die Steuersignale POUT[39:0] durch Negieren, oder Takten, des Verriegelungs-Signals CSOLC_O, das durch das Register **80** empfangen wird.

[0438] Die Steuer-Logik **999** liefert die Signale PWREN[5:0], CLKEN#[5:0], BUSEN#[5:0] und RST#[5:0] zu der Seriell-Ausgangs-Logik **824**. Die Steuer-Logik **999** liefert auch ein PCI-Bus-Anforderungs-Signal CAYREQ# zu dem und empfängt ein PCI-Bus-Erteilungs-Signal CAYGNT# von dem Arbitrierer **124**. Die EIN/AUS-Steuer-Logik **820** stellt das Signal CAYREQ# auf, oder steuert es auf niedrig an, um den PCI-Bus **32** anzufordern, wenn der Arbitrierer **124** das Signal CAYGNT# aufstellt, oder auf niedrig ansteuert, hat der Arbitrierer **124** eine Steuerung über den PCI-Bus **32** zu der EIN/AUS-Steuer-Logik **820** erteilt.

[0439] Wie in den Fig. 33B–G dargestellt ist, tritt die EIN/AUS-Zustand-Maschine **998** in einen Idle-Zustand IDLE unter Aufstellen des Reset-Signals RST ein. Falls kein Leerlauf vorliegt, steuert die EIN/AUS-Zustand-Maschine **998** eine von drei Sequenzen: die Power-down-Sequenz, die Power-on-Sequenz oder die eine Durchgangs-Sequenz, verwendet dazu, die Steuer-Signale POUT[39:0] zu aktualisieren, wie dies durch das Schlitz-Freigabe- **817** und das LED-Steuer- (nicht dargestellt) Register angezeigt ist. Die EIN/AUS-Zustand-Maschine **998** stellt das Lade-Signal CSOLC_O für einen Zyklus des Taktsignals CLK des Registers **80** auf, oder steuert es auf hoch, bis die EIN/AUS-Zustand-Maschine **998** bestimmt, dass die Steuer-Signale POUT[39:0] aktualisiert werden müssen. Wenn die Steuer-Signale POUT[39:0] aktualisiert sind, negiert die EIN/AUS-Zustand-Maschine **998** das Signal CSOLC_O, was die Steuer-Signale POUT[39:0] aktualisiert.

[0440] Die EIN/AUS-Zustand-Maschine **998** beginnt die Power-down-Sequenz, wenn entweder die Software ein energiemäßiges Herunterfahren bzw. Power-down mindestens eines der Schlitze bzw. Einsteckplätze **36** anfordert, wie dies durch das Wegnehmen der Signale SLOT_EN[5:0] angezeigt ist; oder die Seriell-Abtast-Eingangs-Logik **804** bestimmt, dass mindestens einer der Schlitze bzw. Einsteckplätze **36a–f** der Power-down-Sequenz unterworfen werden sollte, wie dies durch das Aufstellen der Signale RST_SWITCH[5:0] angezeigt ist. Um die Power-down-Sequenz zu beginnen, stellt die EIN/AUS-Zustand-Maschine **998** das SO_UPDATE Signal auf, um eine Verschiebe-Phase und Übergänge von dem IDLE Zustand zu einem RSTON

Zustand zu beginnen.

[0441] Während des RSTON Zustands negiert die Steuer-Logik **999** die Reset-Signale RST#[5:0] für die Schlitze **36**, die energiemäßig heruntergefahren werden sollen, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Reset-Signale RST#[5:0] zu dem Ausgangs-Register **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal alle vierzig Steuer-Signale durch die Seriell-Ausgangs-Logik **824** zu dem Register **80** verschoben sind, wie dies durch das Aufstellen des Signals SO_UPDATE_DONE angezeigt ist, geht die EIN/AUS-Zustand-Maschine **998** von dem RSTON Zustand zu einem OFF_ARB1 Zustand über.

[0442] In dem OFF_ARB1 Zustand fordert die EIN/AUS-Zustand-Maschine **998** eine Steuerung über den sekundären PCI-Bus **32** durch Aufstellen des Anforderungs-Signals CAYREQ# an. Die EIN/AUS-Zustand-Maschine **998** geht dann zu einem OFF_WGNT1 Zustand über, wo sie auf die Erteilung des sekundären PCI-Busses **32** wartet. Wenn der Arbitrierer **124** eine Steuerung über den Bus **32** erteilt, wie dies durch das Aufstellen des CAYREQ# Signals angezeigt ist, negiert die EIN/AUS-Zustand-Maschine **998** das Signal CSOLC_O für einen Zyklus des Signals CLK, um die Steuer-Signale POUT[39:0] zu aktualisieren, und geht zu einem OFF_LCLK1 Zustand über.

[0443] In dem OFF_LCLK1 Zustand stellt die EIN/AUS-Zustand-Maschine **998** das Signal SO_UPDATE auf, um so eine andere Verschiebe-Phase zu beginnen. Die EIN/AUS-Zustand-Maschine **998** geht von dem OFF_LCLK1 Zustand zu einem Bus-off-Zustand BUSOFF über. Während des BUSOFF Zustands nimmt die Steuer-Logik **999** die BUS-Freigabe-Signale BUSEN#[5:0] für die Schlitze **36** weg oder steuert sie auf hoch an, die energießmäßig heruntergefahren werden sollen, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Bus-Freigabe-Signale BUSEN#[5:0] zu dem Ausgangs-Register **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal alle vierzig Steuer-Signale durch die Seriell-Ausgangs-Logik **824** verschoben sind, wie dies durch das Aufstellen des Signals SO_UPDATE_DONE angezeigt ist, geht die EIN/AUS-Zustand-Maschine **998** von dem BUSOFF Zustand zu einem OFF_ARB2 Zustand über, wo die Zustand-Maschine **998** wieder erneut eine Kontrolle des sekundären PCI-Busses **32** anfordert. Die Zustand-Maschine **998** geht dann zu einem OFF_WGNT2 Zustand über, wo sie auf die Erteilung des PCI-Busses **32** wartet. Wenn einmal die Erteilung empfangen ist, geht die Zustand-Maschine **998** zu einem OFF_LCLK2 Zustand über, wo die Steuer-Signale POUT[39:0] durch Negieren des Signals CSOLC_O für einen Zyklus des Signals CLK aktualisiert werden. Die Zustand-Maschine **998** geht dann zu einem Takt-Off-Zustand CLKOFF über.

[0444] Während des CLKOFF Zustands nimmt die Steuer-Logik **999** die Takt-Freigabe-Signale CLKEN#[5:0] für die Schlitze **36** weg, oder steuert sie auf hoch an, die energiemäßig heruntergefahren werden sollen. Die Bus-Freigabe-Signale BUSEN#[5:0] ändern sich nicht, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Takt-Freigabe-Signale CLKEN#[5:0] zu dem Ausgangs-Register **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal alle vierzig Steuer-Signale durch die Seriell-Ausgangs-Logik **824** verschoben sind, wie dies durch das Aufstellen des Signals SO_UPDATE_DONE angezeigt ist, geht die EIN/AUS-Zustand-Maschine **998** von dem CLKOFF-Zustand zu einem OFF_ARB3 Zustand über, wo die Zustand-Maschine **998** erneut eine Steuerung bzw. Kontrolle über den PCI-Bus **32** anfordert. Die Zustand-Maschine **998** geht dann zu einem OFF_WGNT3 Zustand über, wo sie auf die Erteilung des PCI-Busses **32** wartet. Wenn einmal die Erteilung empfangen ist, geht die Zustand-Maschine **998** zu einem OFF_LCLK3 Zustand über, wo die Steuersignale POUT[39:0] durch Negieren des Signals CSOLC_O für einen Zyklus des Signals CLK aktualisiert werden. Die Zustand-Maschine **998** geht dann zu einem Power-Off-Zustand PWROFF über.

[0445] Während des PWROFF Zustands nimmt die Steuer-Logik **999** die Energie-Freigabe-Signale PWREN[5:0] für die Schlitze **36** weg oder setzt sie auf niedrig, die energiemäßig heruntergefahren werden sollen. Die Signale REST#[5:0], BUSEN#[5:0] und CLKEN#[5:0] ändern sich nicht, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Energie-Freigabe-Signale PWREN[5:0] zu dem Ausgangs-Register **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal alle vierzig Steuersignale durch die Seriell-Ausgangs-Logik **824** verschoben sind, wie dies durch das Aufstellen des Signals SO_UPDATE_DONE angezeigt ist, geht die EIN/AUS-Zustand-Maschine **998** von dem PWROFF Zustand zu einem OFF_LCLK4 Zustand über, wo die Signale POUT[39:0] durch Negieren des Signals CSOLC_O für einen Zyklus des Signals CLK aktuallisiert werden. Die Zustand-Maschine **998** geht dann zu dem IDLE Zustand über, der die Power-Down-Sequenz abschließt.

[0446] Falls eine Power-Down-Sequenz nicht erforderlich ist, dann bestimmt die EIN/AUS-Zustand-Maschine **998**, ob die Power-Up-Sequenz erforderlich ist. Falls entweder die Software mindestens angefordert hat, dass mindestens einer der Schlitze **36** energiemäßig hochgefahren werden soll, oder ein Hochfahren des Erweite-

rungskastens **30** anhängig ist, dann geht die EIN/AUS-Zustand-Maschine **998** von dem IDLE Zustand zu einem Power-On-PWRON Zustand über, um die Power-On-Sequenz zu beginnen. Um die Power-On-Sequenz zu beginnen, stellt die EIN/AUS-Zustand-Maschine **998** das SO_UPDATE Signal auf, um eine Verschiebe-Phase zu beginnen, und geht von dem IDLE Zustand zu einem Power-On-Zustand PWRON über.

[0447] Während des PWREN Zustands stellt die Steuer-Logik **999** die Power-Freigabe-Signale PWREN[5:0] für die Schlitze **36** auf, die energiemäßig hochgefahren werden sollen, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Power-Freigabe-Signale PWREN[5:0] zu dem Ausgangs-Register **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal alle vierzig Steuersignale durch die Seriell-Ausgangs-Logik **824** verschoben sind, wie dies durch das Aufstellen des Signals SO_UPDATE_DONE angezeigt ist, geht die EIN/AUS-Zustand-Maschine **998** von dem PWRON-Zustand zu einem Initialisierungs-Zustand LDCNT1 eines Zeitgebers **806** über und negiert das Lade-Signal CSOLC_O, um die Steuersignale POUT[39:0] zu aktualisieren.

[0448] In dem LDCNT1 Zustand initialisiert die EIN/AUS-Zustand-Maschine **998** die Zeitgeber **806** so, dass die Zeitgeber **806** eine Indikation liefern, wenn ein vorbestimmtes Stabilisierungs-Verzögerungs-Intervall abgelaufen ist. Das Stabilisierungs-Verzögerungs-Intervall ermöglicht eine ausreichende Zeit für die Karte **807**, die energiemäßig hochgefahren werden soll, um sich zu stabilisieren, wenn einmal der Spannungspegel V_{SS} zu der Karte **807** zugeführt ist. In dem LDCNT1 Zustand stellt die EIN/AUS-Zustand-Maschine **998** auch das Signal CSOLC_O auf. Die EIN/AUS-Zustand-Maschine **820** geht von dem LDCNT1 Zustand zu einem CLKON Zustand über.

[0449] Während des CLKON Zustands stellt die Steuerlogik **999** die Taktfreigabesignale CLKEN#[5:0] für die Schlitze bzw. Einsteckplätze **36** auf oder steuert sie auf niedrig an, die energiemäßig hochgefahren werden sollen. Die PWREN[5:0] Signale verbleiben unverändert, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Taktfreigabesignale CLKEN#[5:0] zu dem Ausgangs-Register **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal das Stabilisierungs-Verzögerungs-Intervall abgelaufen ist, geht die EIN/AUS-Zustand-Maschine **998** von dem CLKOFF Zustand zu einem ON_ARB1 Zustand über.

[0450] In dem ON_ARB1 Zustand fordert die EIN/AUS-Zustand-Maschine **998** eine Steuerung bzw. Kontrolle über den sekundären PCI-Bus **82** durch Aufstellen des Anforderungs-Signals CAYREQ# an. Die EIN/AUS-Zustand-Maschine **998** geht dann zu einem ON_WGNT1 Zustand über, wo sie auf die Erteilung des sekundären PCI-Busses **32** wartet. Wenn einmal die Kontrolle bzw. Steuerung des Busses **32** erteilt ist, wie dies durch das Aufstellen des CAYGNT Signals angezeigt ist, negiert die EIN/AUS-Zustand-Maschine **998** das Signal CSOLC_O, um die Steuer-Signale POUT[39:0] zu aktualisieren, und geht zu einem ON_LCLK1 Zustand über, wo die Signale POUT[39:0] aktualisiert werden.

[0451] Die EIN/AUS-Zustand-Maschine **998** geht von dem ON_LCLK1 Zustand zu einem LDCNT2 Zustand über, wo die Zeitgeber **806** so initialisiert werden, dass die Zeitgeber **806** eine Indikation liefern, wenn ein anderes, vorbestimmtes Stabilisierungs-Verzögerungs-Intervall abgelaufen ist. Dieses Verzögerungsintervall wird dazu verwendet, dem Takt-Signal an der Karte **807** zu ermöglichen, energiemäßig hochgefahren zu werden, um sich zu stabilisieren, bevor die Power-Up-Sequenz fortfährt. Die EIN/AUS-Zustand-Maschine **998** geht von dem LDCNT2 Zustand zu einem Bus-Ein-Zustand BUSON über.

[0452] Während des BUSON Zustands stellt die Steuerlogik **999** die Bus-Freigabe-Signale BUSEN#[5:0] für die Schlitze **36** auf, oder steuert sie auf niedrig an, die energiemäßig heruntergefahren werden sollen. Die Signale CLKEN#[5:0] und PWREN[5:0] verbleiben unverändert, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Bus-Freigabe-Signale BUSEN#[5:0] zu dem Ausgangsregister **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal das Stabilisierungs-Verzögerungs-Intervall abgelaufen ist, geht die EIN/AUS-Zustand-Maschine **998** von dem BUSON Zustand zu einem ON_ARB2 Zustand über, wo die Zustand-Maschine **998** erneut wieder eine Kontrolle des PCI-Busses **32** anfordert. Die Zustand-Maschine **998** geht dann zu einem ON_WGTN2 Zustand über, wo sie auf die Erteilung des Busses **32** wartet. Wenn einmal die Erteilung empfangen ist, geht die Zustand-Maschine **998** zu einem ON_CLK2 Zustand über, wo die Signale POUT[39:0] durch Negieren des Signals CSOLC_O für einen Zyklus des Signals CLK aktualisiert werden. Die Zustand-Maschine **998** geht dann zu einem Reset-Off-Zustand RSTOFF über.

[0453] Während des RESTOFF Zustands stellt die Steuerlogik **999** die Reset-Signale RST#[5:0] für die Schlitze **36** auf, oder negiert sie, die energiemäßig hochgefahren werden sollen, und zwar in Abhängigkeit von deren jeweiligen SLOT_RST_[5:0] Signalen. Die Signale CLKEN#[5:0], PWREN[5:0] und BUSEN#[5:0] verbleiben unverändert, und die Seriell-Ausgangs-Logik **824** verschiebt seriell die Reset-Signale RST#[5:0] zu dem Aus-

gangsregister **80**. Die EIN/AUS-Zustand-Maschine **998** negiert auch das Signal SO_UPDATE. Wenn einmal alle vierzig Steuersignale durch die Seriell-Ausgangs-Logik **824** verschoben sind, wie dies durch das Aufstellen des Signals SO_UPDATE_DONE angezeigt ist, geht die EIN/AUS-Zustand-Maschine **998** von dem RSTON Zustand zu einem ON_ARB3 Zustand über, wo die Zustand-Maschine **998** erneut wieder eine Kontrolle des Busses **32** anfordert. Die Zustand-Maschine **998** geht dann zu einem ON_WGTN3 Zustand über, wo sie auf die Erteilung des Busses **32** wartet. Wenn einmal die Erteilung empfangen ist, geht die Zustand-Maschine **998** zu einem ON_CLK3 Zustand über, wo die Signale POUT[39:0] durch Negieren des Signals CSOLC_O für einen Zyklus des Signals CLK aktualisiert werden. Die Zustand-Maschine **998** geht dann zurück zu dem IDLE Zustand.

[0454] Falls weder die Power-Up-Sequenz noch die Power-Down-Sequenz erforderlich ist, dann bestimmt die EIN/AUS-Zustand-Maschine **998**, ob eine Ein-Durchgang-Sequenz benötigt wird, um ausgewählte solche der Signale POUT[39:0] zu aktualisieren. Falls die GO_UPDATE Signal aufgestellt ist und falls sich irgendwelche Bits des Schlitz-Freigabe-Registers **817** oder des Schlitz-Reset-Registers **918** ändern, dann geht die EIN/AUS Zustand-Maschine **998** zu einem ONEPASS Zustand über und stellt das SO UPDATE Signal auf.

[0455] Die EIN/AUS Zustand-Maschine **998** verbleibt in dem ONEPASS Zustand, bis die vierzig Steuer-Signale zu dem Register **80** verschoben worden sind. Die EIN/AUS-Zustand-Maschine **998** geht dann zu einem OP_ARB Zustand über, wo die Zustand-Maschine **998** eine Kontrolle des PCI-Busses **32** durch Aufstellen des Signals CAYREQ# anfordert. Die Zustand-Maschine **998** geht dann zu einem OP_WGNT Zustand über, wo sie auf die Erteilung des Busses **32** wartet. Wenn einmal die Erteilung empfangen ist, geht die Zustand-Maschine **998** zu einem OP_LCLK Zustand über, wo die Signale POUT[39:0] durch Negieren des Signals CSOLC_O für einen Zyklus des Signals CLK aktualisiert werden. Die Zustand-Maschine **998** geht dann zurück zu dem IDLE Zustand.

[0456] Wie in Fig. 34 dargestellt ist, umfasst die Seriell-Ausgangs-Logik **824** einen Verschiebe-Ausgangs-Bit-Zähler **921**, der ein Sechs-Bit-Zähler-Ausgangs-Signal BIT_CNTR[5:0] liefert, das das Steuer-Signal protokolliert, aus der Seriell-Ausgangs-Logik **824** über das Signal CSOD_O heraus verschoben. Wenn das Signal BIT_CNTR[5:0] gleich zu einer Zahl mit sechs Ziffern gleich zu „39" ist, dann wird ein Signal MAX_CNT aufgestellt. Das Signal MAX_CNT wird zu dem Eingang eines UND-Gates **922** geliefert. Das UND-Gate **922** empfängt weiterhin ein Signal SHIFT4, das aufgestellt wird, wenn die Ausgangs-Verschiebe-Zustand-Maschine **920** in der SHIFT4 Zustand eintritt, der weiter nachfolgend beschrieben wird. Der Ausgang des UND-Gates **922** liefert das Signal SO_UPDATE_DONE.

[0457] Die Ausgangs-Verschiebe-Zustand-Maschine **920** liefert ein Erhöhungs-Zähler-Signal INC_CNTR zu dem Bit-Zähler **921**. Wenn das INC_CNTR-Signal aufgestellt ist, erhöht der Bit-Zähler **921** den Wert, dargestellt durch das Signal BIT_CNTR[5:0]. Wenn ein Lade-Zähler-Signal LOAD_CNTR aufgestellt ist oder wenn das RST-Signal aufgestellt ist, dann löscht der Ausgang eines ODER-Gates **925**, verbunden mit einem Lösch-Eingang des Bit-Zählers **921**, das Signal BIT_CNTR[5:0].

[0458] Das Signal BIT_CNTR[5:0] wird zu dem Auswahl-Eingang eines Multi-Bit-Multiplexers **924** geliefert, der das Signal CSOD_O liefert. Der nullte bis elfte Eingang des Multiplexers **924** empfangen die LED-Steuer-Signale LEDS[11:0]. Der zwölfte bis fünfzehnte Eingang des Multiplexers **924** empfangen Ausgangs-Signale GPOA[3:0] für allgemeine Zwecke. Der sechszehnte bis einundzwanzigste Eingang empfangen die Reset-Signale RST#[5:0]. Der zweiundzwanzigste bis siebenundzwanzigste Eingang empfangen die Takt-Freigabe-Signale CLKEN#[5:0]. Der achtundzwanzigste bis dreiunddreißigste Eingang empfangen die Bus-Freigabe-Signale BUSEN[5:0]. Der vierunddreißigste bis neununddreißigste Eingang empfangen die Energie-Freigabe-Signale PWREN[5:0].

[0459] Wie in den Fig. 35A–B dargestellt ist, tritt die Ausgangs-Verschiebe-Zustand-Maschine **920** in einen IDLE Zustand ein, wenn das Signal RST aufgestellt ist. Falls das Signal SO_UPDATE aufgestellt ist, dann geht die Ausgangs-Verschiebe-Zustand-Maschine **920** von dem IDLE Zustand zu einem SHIFT1 Zustand über.

[0460] Da die Ausgangs-Verschiebe-Zustand-Maschine **920** an der positiven Flanke des PCI-Takt-Signals CLK getaktet wird, geht die Ausgangs-Verschiebe-Zustand-Maschine **920** über einen SHIFT1 Zustand, einen SHIFT2 Zustand, einen SHIFT3 Zustand und einen SHIFT4 Zustand hindurch, um das Takt-Signal CSOSC_O zu erzeugen, das ein viertel der Frequenz des Takt-Signals CLK ist. Während des SHIFT1- und SHIFT2-Zustands wird das Takt-Signal CSOSC_O negiert und auf niedrig gesetzt, und während des SHIFT3 und SHIFT4 Zustands wird das Taktsignal CSOSC_O aufgestellt oder auf hoch gesetzt. Wenn die momentane Verschiebe-Phase abgeschlossen ist, wie dies durch das Aufstellen des Signals MAXCNT angezeigt ist, kehrt die Ver-

schiebe-Zustand-Maschine **920** zu dem IDLE Zustand zurück und das Takt-Signal CSOSC_O wird, bis zu dem Beginn der nächsten Verschiebe-Phase, aufgestellt.

[0461] Wie in Fig. 36 dargestellt ist, wird ein Signal HANG_PEND durch den Lösch-Eingang des Registers **80** empfangen. Das Aufstellen, oder Ansteuern auf hoch, des HANG_PEND Signals asynchron löscht die geeigneten Ausgangs-Steuer-Signale POUT[39:0], um alle Schlitze **36** energiemäßig herunterzufahren, wenn sich der PCI-Bus **32** in einem verriegelten Zustand befindet, wie dies weiter nachfolgend beschrieben ist.

FEHLER-ISOLATION

[0462] Der Bus-Beobachter bzw. Watcher **129** kann einen fehlerhaften bzw. hängenden Zustand auf dem sekundären PCI-Bus **32** erfassen. Falls ein hängender Zustand erfasst wird, stellt der Bus-Beobachter **129** ein Bus-Hängend-Anhängigkeits-Bit ein, das bewirkt, dass die SIO **50** die Schlitze bzw. Einsteckplätze auf dem sekundären PCI-Bus **32** energiemäßig herunterfährt, und eine nicht-maskierbare Unterbrechung (Non-Mascable Interrupt – NMI) wird zu der CPU **14** übertragen. Die CPU **14** spricht auf die NMI durch aufrufen eines NMI-Programms an, um den Schlitz (die Schlitze), die den hängenden Zustand verursachen, zu isolieren. Wenn einmal der defekte Schlitz bzw. die defekten Schlitzen identifiziert sind, werden sie gesperrt oder energiemäßig abgeschaltet.

[0463] Für Software-Diagnostik-Zwecke umfasst der Bus-Beobachter **129** in dem ausgangsseitigen Brücken-Chip **48** einen Bus-Historie-FIFO und einen Bus-Vektor-FIFO. Wenn der sekundäre PCI-Bus **32** geeignet funktioniert, werden die Bus-Historie-Informationen, die eine Adressen-Gruppe (umfassend die PCI-Adresse, die PCI-Befehls-Signale, die PCI-Master-Zahl und das Adressen-Paritäts-Bit) und eine Daten-Gruppe (umfassend die PCI-Daten, die Byte-Freigabe-Signale C/BE[3:0]_, ein Paritäts-Fehler-Signal PERR_, das Daten-Paritäts-Bit, ein Burst-Zyklus-Indikations-Bit und ein Daten-Gültigkeits-Zeichen) durch den Bus-Beobachter **129** bei jeder Transaktion aufgezeichnet. Wenn das PCI-Signal FRAME_ auf dem sekundären PCI-Bus **32** aufgestellt ist, um eine Bus-Transaktion zu starten, werden die Adressen-Gruppe und jede darauffolgende Daten-Gruppe in dem Bus-Hinstorie-FIFO gespeichert. Wenn die Transaktion eine Burst-Transaktion ist, dann wird das Burst-Zyklus-Indikations-Bit auf aktiv in der zweiten Daten-Phase gesetzt. Nach der ersten Datenphase wird das Adressen-Feld in der Adressen-Gruppe, zugeordnet der darauffolgenden Datengruppe in der Burst-Transaktion, um vier erhöht, und die neue Adressen-Gruppe und die Datengruppe werden in der nächsten Stelle des Bus-Hinstorie-FIFO gespeichert. Falls Daten nicht übertragen werden, da ein Zustand eines erneuten Versuchs oder ein Zustand einer Unterbrechung ohne Daten vorliegt, wird das Gültigkeits-Daten-Indikations-Bit auf niedrig gesetzt.

[0464] Sowohl die Adressen-Gruppe als auch die Daten-Gruppe fließen durch eine 2-Stufen-Pipeline, um Zeit für die Daten-Gruppe zuzulassen, um das Daten-Paritäts-Bit und das Daten-Paritäts-Fehler-Bit zu sammeln, und den Aufzeichnungsvorgang zu stoppen, wenn ein Daten-Paritäts-Fehler auftritt, bevor die nächste Adressen-Gruppe gespeichert wird. Falls der Bus in der Mitte einer Schreib-Daten-Phasen hängt, werden die Daten gespeichert, und ein Bus-Hängend-Status-Bit wird in einem Bus-Hängen-Indikations-Register **482** (Fig. 42), zugänglich über einen Kofigurations-Raum, eingestellt. Falls der Bus in der Mitte einer Lese-Daten-Phase hängt, werden die Daten als nicht gültig markiert, und das Bus-Hängend-Bit wird eingestellt.

[0465] Bus-Zustands-Vektoren werden in dem Bus-Vektor-FIFO zusammengestellt und gespeichert, umfassend die folgenden PCI-Steuer-Signale: Schlitz-Anforderungs-Signale REQ[]7:0]_; Schlitz-Erteilungs-Signale GNT[7:0]_; das FRAME_ Signal; das PCI-Vorrichtungs-Auswahl-Signal DEVSEL_; das PCI-Initiator-Bereitschafts-Signal IRDY_; das PCI-Target-Bereitschafts-Signal TRDY_; das Stopp_ Signal; das PCI-Paritäts-Fehler-Signal PERR_; das PCI-System-Fehler-Signal SERR_; und das LOCK_ Signal. An jedem PCI-Takt, in dem sich der Bus-Zustands-Vector ändert, d. h. irgendeines der aufgelisteten Signale ändert einen Zustand, wird der neue Vector in den Bus-Vektor-FIFO hineingespeichert.

[0466] Der Bus-Beobachter **129** umfasst einen Watch-Dog-Zeitgeber **454** (Fig. 40), um zu bestimmen, ob sich der sekundäre Bus **32** verriegelt hat. Falls der Watch-Dog-Zeitgeber **554** abläuft, dann hat der Bus **32** gehangen. Das folgende sind Beispiele von Bus-Hängend-Zuständen, die durch den Watch-Dog-Zeitgeber **454** erfasst werden können: Das FRAME_ Signal wird auf hoch oder niedrig gesetzt; das Signal TRDY_ wird nicht in Abhängigkeit von IRDY_ aufgestellt; der PCI-Arbitrierer **124** erteilt nicht den Bus zu irgendeinem Master; und ein Master, der den Bus **32** anfordert, behält bei, es zu versuchen.

[0467] Wenn der Watch-Dog-Zeitgeber **454** abläuft, wird das Bus-Hängend-Anhängigkeits-Bit auf aktiv in dem Bus-Hängend-Indikations-Register **482** gesetzt. Wenn das Bus-Hängend-Anhängigkeits-Bit auf aktiv ge-

setzt ist, gibt es den Bus-Beobachter **129** frei. Als nächstes werden die Schlitz-Freigabe-Bits in der SIO **50** gelöscht, was bewirkt, dass die Schlitze energiemäßig heruntergefahren werden. Die SIO **50** stellt dann das System-Fehler-Signal SERR_ auf.

[0468] Um die Ursache eines Bus-Hängend-Zustands zu isolieren, bewirkt das System-Fehler-Signal SERR_, dass die Unterbrechungs-Logik in dem System die NMI zu der CPU **14** ausgibt. Wie Fig. 37 zeigt, bestimmt der NMI-Handler zuerst, **400**, ob das Bus-Hängend-Anhängigkeits-Bit eingestellt ist, durch Lesen des Bus-Hängend-Indikations-Registers **482**. Falls dies der Fall ist, ruft der NMI-Handler, **401**, einen BIOS-Isolations-Handler zum Isolieren des defekten Schlitzes bzw. Einsteckplatzes oder der defekten Schlitze auf. Ansonsten werden andere NMI-Prozeduren aufgerufen, **402**.

[0469] Als ein Fehler-Sicherheits-Mechanismus umfasst das Computersystem auch den Automatik-Server-Wiederherstellungs- (Automatic Server Recovery – ASR) Zeitgeber **72**, der gelöscht wird, wenn bestimmte Software-Programme durch das Betriebssystem ausgeführt werden. Falls der ASR-Zeitgeber abläuft (z. B. nach 10 Minuten), zeigt dies an, dass sich das Betriebssystem verriegelt hat. Der sekundäre PCI-Bus **32**, der hängt, kann die Ursache der System-Verriegelung sein, wobei in einem solchen Fall der NMI niemals zu der CPU **14** gehen kann. Falls der ASR-Zeitgeber abläuft, dann tritt ein ASR-erzeugtes Reboot auf. Der ASR-Zeitgeber stellt auch sicher, dass, falls sich der BIOS-Isolations-Handler in der Mitte einer Isolation eines Fehler-Schlitzes auf dem PCI-Bus **32** befindet, und das Computer-System hängt, um ein ASR-Reboot zu verursachen, das Isolations-Programm dort weiterfahren kann, wo es unmittelbar vor dem ASR-Time-Out-Ereignis gelassen wurde.

[0470] Wie Fig. 38 zeigt, wird ein BIOS ASR Handler in Abhängkeit eines ASR-Reboot Zustands aufgerufen. Der ASR-Handler prüft zuerst, **444**, um zu bestimmen, ob eine Isolations-In-Progress-Ereignis-Variable (EV) aktive Informationen enthält, die anzeigen, dass der Isolations-Prozess vor dem ASR-Time-Out-Ereignis im Fortschreiten war. Das Isolations-In-Progress-EV wird in einem nicht-flüchtigen Speicher (NVRAM) **70** gespeichert und umfasst Header-Informationen, die auf aktiv gesetzt werden, um anzuzeigen, dass der Isolations-Prozess gestartet wurde. Das Isolations-In-Progress-EV wird auch mit dem momentanen Zustand des Isolations-Prozesses aktualisiert, umfassend die Schlitze bzw. Einsteckplätze, die geprüft worden sind, die Schlitze, die defekt sind, und die Schlitze, die freigegeben worden sind.

[0471] Falls der Isolations-Prozess im Fortschreiten war, gibt der BIOS-ASR-Handler wieder alle Schlitze frei, **448**, mit Ausnahme solcher, die unmittelbar vor dem ASR-Ereignis freigegeben wurden, was aus dem Isolations-In-Progress-EV bestimmt wird. Die freigegebenen Schlitze vor dem ASR-Reboot waren wahrscheinlich die Ursache der ASR-Durchsicht. Als Folge werden diese Schlitze gesperrt (d. h. energiemäßig heruntergefahren). Als nächstes werden die Zahlen der gesperrten Schlitze als Fehler-Status-Informationen protokolliert, **450**, gespeichert in dem NVRAM, und das Isolations-In-Progress-EV wird gelöscht. Der BIOS-ASR-Handler prüft dann, **452**, um zu bestimmen, ob das Bus-Hängend-Anhängigkeits-Bit eingestellt ist. Falls dies der Fall ist, wird das Bus-Hängend-Anhängigkeits-Bit gelöscht (unter Durchführen eines I/O-Zyklus auf dem sekundären PCI-Bus **32**), um den Bus-Beobachter **129** wieder freizugeben.

[0472] Falls das Isolations-In-Progress-EV nicht in den aktiven Zustand eingestellt ist, **444**, anzeigend, dass der Isolations-Prozess nicht lief, als das ASR-Ereignis auftrat, prüft das Programm, **446**, um zu bestimmen, ob das Bus-Hängend-Anhängigkeits-Bit eingestellt ist. Falls nicht, dann wird der BIOS-ASR-Handler vorgenommen. Falls das Bus-Hängend-Anhängigkeits-Bit eingestellt ist, **446**, was anzeigt, dass ein Bus-Hängend-Zustand aufgetreten ist, und zwar vor dem ASR-Ereignis, ruft der BIOS-ASR-Handler den BIOS-Isolations-Handler auf, um den Fehler-Schlitz oder Schlitze zu isolieren.

[0473] Wie Fig. 39 zeigt, protokolliert der Isoaltions-Handler zuerst, **408**, zu dem Fehler-Status-Informations-Bereich des NVRAM die Bus-Historie- und Bus-Zustand-Vektor-Informationen, gespeichert in den Historie- und Vektor-FIFOs, in dem Bus-Monitor **127**. Die Bus-Historie- und Bus-Zustand-Vektor-FIFOs werden gelesen und deren Inhalte werden zu dem NVRAM übertragen. Als nächstes werden die Header-Informationen der Isolation-In-Progress-Ereignis-Variable eingestellt, **410**, um anzeigen, dass sich der Isolations-Prozess beim Fortschreiten befindet. Das Bus-Hängend-Anhängigkeits-Bit wird gelöscht (durch Schreiben zu einer vorbestimmten Konfigurations-Adresse), um den Bus-Watcher bzw. -Beobachter **129** wieder freizugeben. Als nächstes gibt das Isolations-Programm den zuerst belegten Schlitz bzw. Einsteckplatz (d. h. ein Schlitz, mit dem eine PCI-Vorrichtung verbunden ist) wieder frei (d. h. fährt es energiemäßig hoch), **412**, und liest und schreibt von dem PCI-Konfigurations-Raum der Vorrichtung. Ein Schlitz wird durch Schreiben zu dem Schlitz-Freigabe-Register **817** (Fig. 29) wieder freigegeben. Als nächstes bestimmt das Programm **414**, ob das Bus-Hängend-Anhängigkeits-Bit auf aktiv gesetzt ist, was anzeigt, dass die Vorrichtung, verbunden mit dem

Schlitz, verursachte, dass der Bus hängt, während davon gelesen wurde. Falls dies nicht der Fall ist, bestimmt das Programm, **416**, ob alle belegten Schlitze geprüft worden sind. Falls dies nicht der Fall ist, wird der erste, belegte Schlitz, gesperrt, **418**, und das Isolation-In-Progress-EV wird aktualisiert, **420**, um anzuzeigen, dass der erste, belegte Schlitz erneut durch den BIOS-Isolations-Handler versucht worden ist. Falls das Programm bestimmt, **414**, dass das Bus-Hängend-Anhängigkeits-Bit auf aktiv gesetzt ist, wird der Schlitz als ausgefallen angezeigt (z. B. durch Einstellen auf aktiv eines Ausfall-Zeichens für diesen Schlitz), und zwar in dem Ausfall-Status-Informations-Bereich des NVRAM. Als nächstes wird die Schleife, die aus den Schritten **412**, **414**, **416**, **418** und **420** besteht, durchgeführt, bis alle belegten Schlitze geprüft worden sind.

[0474] Falls alle belegten Schlitze geprüft worden sind, **416**, nimmt das Programm eine Prüfung vor, **424**, um zu bestimmen, ob irgendein Schlitz als ausgefallen in dem Ausfall-Status-Informations-Bereich des NVRAM angezeigt ist. Falls dies der Fall ist, gibt das Programm **398** erneut nur die nicht ausgefallenen Schlitze frei, **426**. Dann wird das Isolation-In-Progress-EV gelöscht, **428**, und das BIOS-Isolations-Programm wird abgeschlossen.

[0475] Falls keiner der Schlitze als ausgefallen angezeigt ist, **424**, dann zeigt dies an, dass der Bus-Hängend-Zustand nicht durch einen einzelnen Schlitz verursacht worden ist, sondern kann durch mehr als eine Vorrichtung, die zu diesem selben Zeitpunkt aktiv sind, verursacht sein. Um dies zu bestätigen, sperrt der BIOS-Isolations-Handler zuerst alle Schlitze (d. h. fährt sie energiemäßig herunter), **430**, und aktualisiert das Isolation-In-Progress-EV mit diesen Informationen. Als nächstes löscht der BIOS-Isolations-Handler **431** eine Zähl-Variable N auf 0 und stellt eine Zähl-Variable I auf den Wert von N ein. Die Zähl-Variable N stellt die Zählung der belegten Schlitze dar.

[0476] Der BIOS-Isolations-Handler gibt wieder den belegten Schlitz I (der zu Anfang Schlitz N ist) frei (d. h. fährt ihn energiemäßig hoch), **432**, und liest und schreibt zu diesem PCI-Konfigurations-Raum. Der Handler prüft dann, **438**, um zu bestimmen, ob das Bus-Hängend-Anhängigkeits-Bit eingestellt ist. Falls dies nicht der Fall ist, verringert der Handler **433** die variable I und prüft, **434**, ob die Variable I größer als oder gleich zu null ist. Falls dies der Fall ist, aktualisiert der Handler **435** das Isolations-In-Progress-EV und gibt wieder den nächsten, belegten Schlitz I frei, **432**, und liest und schreibt mit diesem. Der Handler prüft dann, **438**, ob das Bus-Hängend-Anhängigkeits-Bit für diesen nächsten Schlitz eingestellt ist. Auf diese Art und Weise werden, für jeden Schlitz N, der freigegeben werden soll, die vorher freigegebenen Schlitze auch energiemäßig, einer zu einem Zeitpunkt, hochgefahren, um zu bestimmen, ob eine Kombination von Schlitzen den Fehler verursacht.

[0477] Falls die Variable I dahingehend bestimmt ist, **434**, dass sie geringer als Null ist, dann prüft der Handler **436**, um zu bestimmen, ob alle belegten Schlitze freigegeben worden sind. Falls dies nicht der Fall ist, wird die Variable N erhöht, **437**, das Isolation-In-Progress-EV wird aktualisiert, **439**, und die Variable I wird wieder eingestellt, **441**, und zwar gleich zu dem Wert von N.

[0478] Falls das Bus-Hängend-Anhängigkeits-Bit auf aktiv gesetzt ist, **438**, dann werden potentiell zwei Schlitze gesperrt, **440**: Schlitz N (der der Schlitz ist, der momentan freigegeben ist), und Schlitz I (der der Schlitz ist, von dem momentan gelesen und zu dem momentan geschrieben wird). Falls die Werte von I und N dieselben sind, dann wird nur Schlitz N gesperrt.

[0479] Falls der Handler bestimmt, **436**, dass alle belegten Schlitze freigegeben worden sind (und ein Fehler nicht identifiziert werden konnte), dann lockt der Handler den NVRAM seine Unfähigkeit ein, **442**, um den Fehler zu isolieren. Als nächstes löscht der Handler **428** das Isolation-In-Progress-EV.

[0480] Wie Fig. 40 zeigt, liefert der Watch-Dog-Zeitgeber **454** Ausgangs-Signale WD_TMR_OUT[17:0] (Zeitgeber-Zähl-Wert), HANG_PEND (Bus-Hängend-Zustand ist vorhanden), EN_CAP (die Software einer Erfassung der Bus-Vektor-Historie-Information ist freigegeben), TIME_OUT (der Watch-Dog-Zeitgeber **454** ist abgelaufen), ein Signal HANG_RCOVR_EN (eingestellt auf Hoch durch die Software, um die Hang-Recovery-Logik in dem Bus-Watcher **129** und in dem SIO **50** freizugeben), und ein Signal CAP_ILLEG_PROT (um einen illegalen Zyklus auf dem PCI-Bus **32** anzuzeigen).

[0481] Das Signal HANG_PEND wird zu der SIO **50** geliefert, um die Sekundär-Bus-Schlitze zu schließen. Die Eingangs-Signale zu dem Watch-Dog-Zeitgeber **454** umfassen. einige der PCI-Bus-Signale, ein Signal WRT_EN_CAP_1 (gepulst auf Hoch durch die Software, um erneut die Erfassung der Bus-Historie- und Bus-Vektor-Informationen durch den Fehler-Isolations-Block **129** freizugeben), und ein Energie-Good-Indikator-Signal SYNC_POWEROK (was anzeigt, dass Energie zu dem Computer-System stabil ist).

[0482] Eine Bus-Hängend-Zurückgewinnungs-Zustand-Maschine **456** empfängt die Signale HANG_PEND, TIME_OUT und HANG_RCOVR_EN von dem Watch-Dog-Zeitgeber **454**. Die Zurückgewinnungs-Zustand-Maschine **456** empfängt auch einige der PCI-Signale. Die Ausgangs-Signale von der Bus-Hängend-Zurückgewinnungs-Zustand-Maschine **456** umfassen ein Vorrichtungs-Auswahl-Signal DEVSEL_O zum Ansteuern des PCI-Signal DEVSEL_, ein Signal STOP_O zum Ansteuern des PCI-Signal STOP_, ein Signal SERR_EN, das ein Aufstellen des System-Fehler-Signals SERR_ freigibt, ein Signal BR_M_ABORT (was anzeigt, dass sich der Bus-Watcher **129** mit einem Master-Abort wieder hergestellt hat) ein Signal BR_T_ABORT, das anzeigt dass sich der Bus-Watcher **129** mit einem Target-Abort wieder hergestellt hat), und ein Signal RCOVR_ACTIVE (zum Anzeigen, wenn die Bus-Hängend-Zurückgewinnungs-Zustand-Maschine **456** aktiv ist). Die Bus-Hängend-Zurückgewinnungs-Zustand-Maschine **456** stellt sicher, dass der sekundäre PCI-Bus **32** zurück zu dem Leerlauf- bzw. IDLE Zustand gebracht wird, um der Software zu ermöglichen, den Fehler-Schlitz bzw. -Einsteckplatz zu isolieren. Wenn der Hang-Zustand erfasst ist, fährt die ISO 50 die sekundären Bus-Schlitze herunter, was automatisch den Bus **32** in den Leerlauf-Zustand versetzten würde, falls einer der Schlitz-Vorrichtungen der Bus-Master war, wenn der Hang-Zustand auftrat. Allerdings würde dann, wenn eine der Schlitz-Vorrichtungen das Target war (und der Brücken-Chip **48** der Master war), wenn das Bus-Hangen auftrat, dann der Brücken-Chip **48** an dem Bus verbleiben. Um den Brücken-Chip aus dem Bus herauszunehmen, erzwingt die Zurückgewinnungs-Zustand-Maschine **456** einen Zyklus für einen erneuten Versuch auf dem PCI-Bus **32** durch Aufstellen des Signals STOP_.

[0483] Ein Bus-Historie-Erfassungs-Block **458** überwacht den PCI-Bus **32** hinsichtlich Transaktionen und liefert die Bus-Historie-Informationen (umfassend die Adresse und die Daten) weiter zu Ausgangs-Signalen BUS_HIST_DATA3[31:0] (die Bus-Historie-Adresse), BUS_HIST_DATA2[31:0] (die Bus-Historie-Daten) und BUS_HIST_DATA1[15:0] (Paritäts-Fehler-Signal !PERR_, Paritäts-Bit-PAR, Gültigkeits-Daten-Bit VALID_DAT, Adressen-Paritäts-Bit ADDRPAR, Burst-Indikator BURST, Master-Zahl MASTER[2:0], Byte-Freigabe-Bits CBE[3:0]_, und Befehl-Bits CMD[3:0]). Der Bus-Historie-Erfassungs-Block **458** stellt ein Signal HIS_RDY auf, wenn Daten auf den BUS_HIST_DATA Signalen verfügbar sind, was der Fall an dem Ende jeder Daten-Phase in einer normalen Transaktion ist, oder falls die Transaktion mit einer Master-Aussonderung, einem erneuten Versuch, beendet wird, und zwar während des Aufstellens des Time-Out- Signals TIME_OUT.

[0484] Ein Bus-Vektor-Erfassungs-Block **460** erfasst die Zustände von bestimmten PCI-Steuersignalen, wenn irgendeines dieser Steuersignale seinen Zustand ändert. Der Vektor wird erfasst und als Signal BUS_VECT_DATA[20:0] ausgegeben, die die Anforderungs-Signale !REQ[7:0]_, Erteilungs-Signale !GNT[7:0]_, ein Time-Out-Signal TIME_OUT, ein Verriegelungs-Signal LOCK_, ein System-Fehler-Signal SERR_, ein Paritäts-Fehler-Signal PERR_, ein Stop-Signal STOP_, ein Target-Ready-Signal TRDY_, ein Initiator-Ready-Signal IRDY_, ein Vorrichtungs-Auswahl-Signal DEVSEL_ und ein Frame-Signal FRAME_ enthalten. Der Bus-Vektor-Erfassungs-Block **460** stellt ein Signal VECT_RDY auf, falls sich irgendein Bus-Vektor BUS_VECT_DATA[24:0] geändert hat oder der Watch-Dog-Zeitgeber **454** abgelaufen ist (TIME_OUT ist hoch).

[0485] Die Bus-Historie- und Bus-Vektor-Signale werden den Eingängen eines Bus- Watcher FIFOs präsentiert, das einen 2-Deep-Bus-Historie-FIFO und einen 4-Deep-Vektor-Historie-FIFO umfasst. Die Ausgänge der Bus-Historie-FIFOs werden als Signale BUS_HIST_REG1[31:0], BUS_HIST_REG2[31:0] und BUS_HIST_REG3[31:0] präsentiert. Die Ausgänge des Vektor-Historie-FIFOs werden als Signale BUS_VECT_REG[3:0] präsentiert. Die System-Software liest die Ausgänge des Bus-Historie-FIFO durch Erzeugen eines I/O-Lese-Zyklus, der bewirkt, dass ein Signal BUS_HIST_RD1 aufgestellt wird, und liest die Ausgänge des Vektors FIFO durch Erzeugen eines I/O-Lese-Zyklus, der bewirkt, dass ein Signal BUS_VECT_RD aufgestellt wird.

[0486] Wie Fig. 41 zeigt, beginnt die Zurückgewinnungs-Zustand-Maschine **456** in einem Zustand IDLE, wenn das Signal SYNC_POWEROK auf niedrig gesetzt wird, was anzeigt, dass Energie bis jetzt noch stabil ist. Die Zustand-Maschine verbleibt in einem Zustand IDLE, während das Signal HANG_PEND niedrig ist. In dem Zustand IDLE werden Signale BR_M_ABORT, BR_T_ABORT und RCOVR_ACTIVE auf niedrig gesetzt. Das Signal RCOVR_ACTIVE ist aktiv hoch in den anderen Zuständen WATT, ABORT und PEND_OFF. Falls das Signal SET_HANG_PEND auf hoch gestellt ist, geht die Zustand-Maschine zu einem Zustand WAIT über. Bei dem Übergang wird das Signal DEVSEL_O gleich zu dem invertierten Zustand des Vorrichtungs-Auswahl-Signals DEVSEL_ gesetzt. Dies stellt sicher, dass dann, wenn das Vorrichtungs-Auswahl-Signal DEVSEL_ durch ein Target vor dem Bus-Hängend-Zustand aufgestellt wird, die Zurückgewinnungs-Zustand-Maschine **456** das Signal DEVSEL_ aufgestellt beibehält. In dem Zustand WATT wird das Signal DEVSEL_O gleich zu dem Zustand des Signals DEV_SEL_WAS gesetzt, was auf hoch gesetzt wird, falls das Signal DEVSEL_ durch ein Target aufgestellt ist, bevor die Zustand-Maschine zu dem WAIT Zustand übergeht.

[0487] Von dem Zustand WAIT geht die Bus-Hängend-Zurückgewinnungs-Zustand-Maschine **456** zu dem PEND_OFF Zustand über, falls ein Signal PCI_IDLE aufgestellt ist, was anzeigt, dass der PCI-Bus **32** zu einem Leerlauf übergegangen ist (d. h. Signale FRAME_ und IRDY_ sind beide auf hoch gesetzt). Bei dem Übergang wird das Signal BR_M_ABORT auf hoch gesetzt, um anzuzeigen, dass eine der Schlitz-Vorrichtungen der Master vor dem Hang-Zustand war, und die Schlitz-Vorrichtung wird energiemäßig heruntergefahren, verursacht dadurch, dass der PCI-Bus zu einem Leerlauf übergeht. Ein Signal SERR_EN wird auch auf hoch gesetzt, um ein Aufstellen des System-Fehler-Signals SERR_ freizugeben oder falls INTA_ freigegeben ist.

[0488] Falls eine Schlitz-Vorrichtung ein Target war, und zwar vor dem Bus-Hängend-Zustand, dann wird der Bus-Master auf dem PCI-Bus **32** verbleiben. Um den Bus-Master aus dem PCI-Bus **32** herauszubringen, gibt die Bus-Hängend-Zurückgewinnungs-Zustand-Maschine **456** einen erneuten Versuch auf dem PCI-Bus **32** heraus. Ein Zähler **457** zählt eine vorbestimmte Zahl von PCLK-Perioden (z. B. 15 PCLK-Perioden), nachdem das Signal HANG_PEND auf hoch gesetzt ist. Die 15 PCLK-Perioden stellen eine ausreichende Anstiegszeit auf FRAME_ und IRDY_ sicher, um den Signalen Zeit zu geben, zurück zu deren Leerlauf-Zuständen zu gehen. Wenn **15** PCLK-Perioden abgelaufen sind, stellt der Zähler **457** das Signal TIME_OUT15 auf. Falls das Signal TIME_OUT15 auf hoch gesetzt ist, und das Signal PCI_IDLE niedrig verbleibt, dann geht die Zustand-Maschine von einem Zustand WATT zu einem Zustand ABORT über. Bei dem Übergang wird das Signal STOP_O auf hoch gesetzt, um das PCI STOP_ Signal auf aktiv anzusteuern, um den Bus-Master erneut zu versuchen. Die Zustand-Maschine verbleibt in einem Zustand ABORT, während der Bus-Master das Signal FRAME_ auf niedrig aufgestellt beibehält. In dem Zustand ABORT wird das Signal STOP_O auf hoch beibehalten. Wenn einmal der Bus-Master das FRAME_ Signal in Abhängigkeit des Wiederversuch-Zustands zurücknimmt, geht die Zustand-Maschine von einem Zustand ABORT zu einem Zustand PEND_OFF über. Bei dem Übergang wird das Signal BR_T_ABORT auf hoch gesetzt, um anzuzeigen, dass das Target-Abort bzw. die Ziel-Aussonderung notwendig war, und zwar nach dem Bus-Hängend-Zustand, um den Bus **32** in den Leerlauf-Zustand zu platzieren. Das Signal SERR_EN wird auch auf hoch gesetzt, um ein Aufstellen des Signals SERR_ zu ermöglichen, oder falls INTA_ freigegeben ist. Die Zustand-Maschine verbleibt in dem Zustand PEND_OFF, bis das Signal WRT_EN_CAP_1 auf hoch gesetzt worden ist, wobei es zu diesem Zeitpunkt zurück zu dem Zustand IDLE übergeht.

[0489] Die System-Software kann den Wert von BR_M_ABORT und BR_T_ABORT Signalen lesen, um zu bestimmen, ob die Schlitz-Vorrichtung, eingeschlossen in dem Bus-Hängend-Zustand, eine Master-Vorrichtung oder eine Slave-Vorrichtung war.

[0490] Wie Fig. 42 zeigt, umfasst der Watch-Dog-Zeitgeber **454** einen 18-Bit-LSFR-Zähler **464**, der durch das Signal PCLK getaktet wird. Der Zähler **464** wird dann freigegeben, wenn der Ausgang eines UND-Gates **467** auf hoch gesetzt ist, was dann auftritt, wenn ein neuer Master eine Anforderung (ANY_REQ ist hoch) ausgibt, der Bus-Zyklus gestartet ist (Signale FRAME_ und IRDY_ sind beide aufgestellt), das Freigabe-Erfassungs-Signal EN_CAP aufgestellt ist, und das Signal TIME_OUT niedrig ist. Ein ODER-Gate **466** empfängt das Signal ANY_REQ und die invertierten Zustände von Signalen FRAME_ und IRDY_. Das UND-Gate **467** empfängt den Ausgang fo des ODER-Gates **466**, das Signal EN_CAP und den invertierten Zustand des Signals TIME_OUT. Der Ausgang des Zählers steuert Signale WD_TMR_OUT[17:0] und wird gelöscht, wenn ein Time-Out-Zustand erfasst ist (TIME_OUT ist hoch), eine Datenübertragung stattgefunden hat (beide Signale IRDY_ und TRDY_ sind auf niedrig gesetzt), oder alle Ausgangs-Bits des Zählers **464** hoch sind (was ein illegaler Zustand ist). Der Lösch-Zustand wird durch ein ODER-Gate **470** angezeigt, das das Signal TIME_OUT empfängt, das bit-weise AND (UND) der Signale WD_TMR_OUT[17:0] und den Ausgang eines UND-Gates **472**. Die Eingänge des UND-Gates **472** nehmen den invertierten Zustand des Signals IRDY_ und den invertierten Zustand des Signals TRDY_ auf.

[0491] Das Signal TIME_OUT wird auf hoch durch einen Time-Out-Detektor **474** gesetzt, wenn die Zeitgeber-Signale WD_TMR_OUT[17:0] zu dem binären Wert 1000000000000000. Das Signal TIME_OUT wird zu einem Eingang eines ODER-Gates **476** geliefert, dessen Ausgang mit dem Eingang eines UND-Gates **478** verbunden ist. Der andere Eingang des UND-Gates **478** nimmt den invertierten Zustand eines Signals WRT_EN_CAP_1 auf (gesteuert durch eine Software, um erneut die Bus-Historie und Bus-Vektor-Erfassung zu ermöglichen), und sein Ausgang wird mit dem D_Eingang eines Flip-Flops **488** vom D-Typ verbunden. Das Flip-Flop **488** wird durch das Signal PCLK getaktet und steuert ein Ausgangs-Signal WD_TIME_OUT an, das zurück zu dem anderen Eingang des ODER-Gates **476** zugeführt wird. Das Flip-Flop **488** wird dann gelöscht, wenn das Power-Good-Signal SYNC_POWEROK negiert wird. Demzufolge löscht ein ASR-Reset nicht das Signal WD_TIME_OUT.

[0492] Das HANG_PEND Signal wird durch ein Flip-Flop **482** vom D-Typ auf hoch gesetzt, dessen D-Eingang

mit dem Ausgang eines UND-Gates **484** verbunden wird und der durch das Signal PCLK getaktet wird. Ein Eingang des UND-Gates **484** ist mit dem Ausgang eines ODER-Gates **486** verbunden, und sein anderer Eingang nimmt den invertierten Zustand des Signals WRT_EN_CAP_1 auf. Ein Eingang des ODER-Gates **486** ist mit dem Signal HANG_PEND verbunden, und der andere Eingang ist mit dem Ausgang eines UND-Gates **488** verbunden. Die Eingänge des UND-Gates **488** nehmen das Signal TIME_OUT und das Freigabe-Signal HANG_RCOVR_EN auf. Demzufolge wird, falls die System-Software eine Bus-Hängend-Wiederherstellung freigibt (HANG_RCOVREN ist hoch), dann wird ein Time-out-Zustand bewirken, dass das Signal HANG_PEND auf hoch gesetzt wird. Das Signal HANG_PEND wird dann gelöscht, wenn die System-Software bewirkt, dass das Signal WRT_EN_CAP_1 aufgestellt wird (unter Durchführen eines I/O-Zyklus auf dem Bus **32**), oder wenn das Signal SYNC_POWEROK negiert wird. Das Bit HANG_PEND wird nicht negiert durch einen ASR-Reboot.

[0493] Das Freigabe-Erfassungs-Signal EN_CAP wird durch ein Flip-Flop **490** vom D-Typ erzeugt, dessen D-Eingang den Ausgang eines UND-Gates **492** aufnimmt. Ein Eingang des UND-Gates **492** ist mit dem Ausgang eines ODER-Gates **494** verbunden, und sein anderer Eingang ist mit dem invertierten Zustand eines Signals CLR_EN_CAP verbunden. Ein Eingang des ODER-Gates **494** wird zurück zu dem Signal EN_CAP geführt und der andere Eingang nimmt das Signal WRT_EN_CAP_1 auf. Das Flip-Flop **490** wird durch das Signal PCLK getaktet und auf hoch gesetzt, wenn das Signal SYNC_POWEROK auf niedrig gesetzt wird. Wenn einmal das Signal EN_CAP auf hoch durch die Software über das Signal WRT_EN_CAP_1 gesetzt wird, wird es auf hoch beibehalten. Das Signal CLR_EN_CAP wird aufgestellt, um das Signal EN_CAP zu löschen (Disable Capture of Information – Sperren einer Erfassung von Information), was dann auftritt, wenn ein Zeitablauf bzw. Time-Out aufgetreten ist (TIME_OUT ist hoch), ein System-Fehler aufgetreten ist (SERR_ ist niedrig), ein Paritäts-Fehler aufgetreten ist (PERR_ ist niedrig), oder ein illegales Bus-Protokoll erfasst worden ist (CAP_ILLEG_PROT ist hoch).

[0494] Das Signal CAP_ILLEG_PROT wird durch ein Flip-Flop **483** vom D-Typ erzeugt, dessen D-Eingang den Ausgang eines UND-Gates **485** aufnimmt. Ein Eingang des UND-Gates nimmt den invertierten Zustand des Signals WRT_EN_CAP_1 auf, und der andere Eingang empfängt den Ausgang eines ODER-Gates **487**. Das ODER-Gate **487** empfängt die Signale CAP_ILLEG_PROT und SET_ILLEG_PROT. Das Signal SET_ILLEG_PROT wird dann aufgestellt, wenn ein Protokollieren (capture) freigegeben ist (EN_CAP ist hoch), die Zustand-Maschine **456** nicht aktiv ist (RCOVR_ACTIVE ist niedrig), der Bus leerläuft und irgendwelche Signale DEVSEL_, TRDY_ oder IRDY_ auf niedrig gesetzt sind. Dieser Zustand ist ein illegaler Zustand, der eine Protokollierung der Bus-Historie und von Bus-Vektor-Informationen triggert.

[0495] Wie Fig. 43 zeigt, wird das Bus-Historie-Bereitschaft-Signal HIST_RDY durch ein Flip-Flop **502** vom D-Typ erzeugt, das durch das Signal PCLK getaktet und durch das Signal RESET gelöscht wird. Der D-Eingang des Flip-Flops **502** ist mit dem Ausgang eines ODER-Gates **504** verbunden, dessen Eingänge das Signal TIME_OUT, ein Signal M_ABORT (Master-Abort-Signal, verzögert durch eine PCLK), den Ausgang eines UND-Gates **506** und den Ausgang eines UND-Gates **508** aufnehmen. Das UND-Gate **506** stellt seinen Ausgang auf, falls ein erneuter Versuch, C zu trennen, oder ein Target-Abort-Zyklus auf dem sekundären Bus **32** vorhanden ist (das Signal FRAME_, der invertierte Zustand des Signals IRDY_, der invertierte Zustand des Signals STOP_, und der invertierte Zustand des Signals DSC_A_B sind alle wahr). Das UND-Gate **508** stellt seinen Ausgang auf, wenn eine abgeschlossene Daten-Übertragung aufgetreten ist (die Signale IRDY_ und TRDY_ sind beide niedrig). Demzufolge werden die Bus-Historie-Informationen in die Bus-Historie-FIFOs eingeladen, wenn der Watch-Dog-Zeitgeber **554** zeitmäßig abläuft, ein erneuter Versuch, C zu unterbrechen, oder ein Target-Abort-Zustand vorhanden ist, der Master den Zyklus ausgesondert hat oder ein Zyklus erfolgreich abgeschlossen wurde.

[0496] Das Gültigkeits-Daten-Indikations-Signal VALID_DATA wird durch einen Flip-Flop **510** vom D-Typ erzeugt, das durch das Signal PCLK getaktet wird und durch das Signal RESET gelöscht wird. Der D-Eingang des Flip-Flops **510** wird mit dem Ausgang eines NOR-Gates **512** verbunden, das das Signal TIME_OUT, das Master-Abort-Signal M_ABORT und den Ausgang des UND-Gates **506** aufnimmt. Demzufolge sind Daten gültig, ohne dass ein Zeitablauf erfasst ist, ein Master-Abort-Zyklus ausgegeben ist, oder ein erneuter Versuch, C zu trennen, oder ein Target-Abort-Zyklus, vorhanden ist.

[0497] Das Signal VECT_RDY wird durch ein Flip-Flop **514** vom D-Typ erzeugt, das durch das Signal PCLK getaktet wird und durch das Signal RESET gelöscht wird. Der D-Eingang des Flip-Flops **514** wird mit dem Ausgang eines ODER-Gates **516** verbunden, der das Zeitablaufsignal TIME_OUT und ein Signal CHANGE_STATE aufnimmt, was anzeigt, dass eines der PCI-Steuer-Signale in dem Bus-Vector seinen Zustand geändert hat. Demzufolge werden die Zustands-Vektor-Informationen in die Vektor-FIFOs immer dann

eingeladen, wenn Steuersignale auf dem PCI-Bus **32** einen Zustand ändern oder wenn ein Zeitablauf aufgetreten ist.

[0498] Wie Fig. 44 zeigt, werden die Bus-Historie-Daten {BUS_HIST_DATA3[31:0], BUS_HIST_DATA2[31:0], BUS_HIST_DATA1[15:0]} zu dem Eingang des Bus-Historie-Registers **540** geliefert, das die erste Stufe des Bus-Historie-FIFO's ist. Die Bus-Historie **501** liefert Ausgangs-Signale BUS_HIST_FIFO1[79:0] zu dem Register **542** (der zweite Zustand der Pipeline), was Ausgangs-Signale BUS_HIST_FIFO0[79:0) liefert. Beide Bus-Historie-Register **540** und **542** werden durch das Signal PCLK getaktet und gelöscht, wenn das Power-Good-Signal SYNC_POWEROK niedrig ist.

[0499] Die Bus-Historie-Register **540** und **542** werden dann geladen, wenn der Ausgang des UND-Gates **518** auf hoch angesteuert wird. Das UND-Gate **518** empfängt das Freigabe-Erfassungs-Bit EN_CAP und das ODER des Bus-Historie-Bereitschafts-Signals HIST_RDY und des CAP_ILLEG_PROT Signals (ODER-Gate **519**). Die Ausgangs-Signale BUS_HIST_FIFO0[79:0] und BUS_HIST_FIFO1[79:0] werden zu den 0- und 1-Eingängen jeweils von Multiplexern **520**, **522** und **524** geliefert. Jeder der Multiplexer **520**, **522** und **524** wird durch ein Lese-Adressen-Signal HIST_FIFO_RD_ADDR ausgewählt (das mit niedrig startet, um den Ausgang des Bus-Registers **502** auszuwählen, und bei jeder darauffolgenden Lesung getoggelt wird). Die Multiplexer **520**, **522** und **524** steuern Ausgangs-Signale BUS_HIST_REG3[31:0], BUS_HIST_REG2[31:0], BUS_HIST_REG1[15:0] jeweils an.

[0500] Die Bus-Vektor-Daten-Signale BUS_VECT_DATA[24:0] werden zu den Eingängen eines Bus-Vektor-Registers **544** geliefert, dessen Ausgang zu dem Eingang eines Bus-Vektor-Registers **546** weitergeführt wird. Der Ausgang des Bus-Vektor-Registers **546** wird zu dem Eingang eines Bus-Vektor-Registers **548** weitergeführt, dessen Ausgang wiederum zu dem Eingang eines Bus-Vektor-Registers **550** zurückgeführt wird. Jedes der Bus-Vektor-Register 0–3 wird durch das Signal PCLK getaktet und dann gelöscht, wenn die Signale SYNC_POWEROK niedrig sind. Die Bus-Vektor-Register werden dann geladen, wenn der Ausgang des UND-Gates **521** auf hoch gesetzt wird. Das UND-Gate **521** empfängt das Signal EN_CAP und das ODER von Signalen VECT_RDY und CAP_ILLEG_PROT (ODER-GATE **523**). Die Bus-Vektor-Register **550**, **548**, **546** und **544** erzeugen Ausgangs-Signale BUS_VECT_FIFO0[24:0], BUS_VECT_FIFO1[24:0], BUS_VECT_FIFO0[24:0] und BUS_VECT_FIFO3[24:0] jeweils, die wiederum zu den 0-, 1-, 2- und 3-Eingängen eines Multiplexers **526** jeweils eingegeben werden. Der Ausgang des Multiplexers **526** liefert Signale BUS_VECT_REG[31:0], wobei der Multiplexer **526** einen seiner Eingänge basierend auf dem Zustand von Adressen-Signalen VECT_FIFO_RD_ADDR[1:0] auswählt (was mit einem binären Wert 00 beginnt und bei jeder darauffolgenden Lesung erhöht wird).

[0501] Demzufolge werden die Bus-Historie- und Bus-Zustand-Vektor-Informationen in Abhängigkeit eines Aufstellens von Signalen HIST_RDY oder VECT_RDY jeweils protokolliert, oder in Abhängigkeit eines Aufstellens des Signals CAP_ILEG_PROT, falls ein illegaler Bus-Protokoll-Zustand erfasst ist.

ERWEITERUNGS-KARTEN-RAUM-RESERVIERUNG

[0502] Im Gegensatz zu herkömmlichen Computersystemen reserviert, in der Anfangs-Konfiguration des Computersystems **10**, bei einem Einschalten, die CPU **14** einen Speicherraum und PCI-Bus-Zahlen für die Schlitze bzw. Eisteckplätze **36**, die leer sind (keine Karte **807** ist eingesetzt) oder abgeschaltet sind bzw. heruntergefahren sind.

[0503] Die CPU **14** ordnet, wie dies typischerweise vorgenommen wird, Bus-Zahlen für PCI-Busse (z. B. PCI-Busse **24**, **32a**–**b** und PCI-Buse) der Karten **807**, die in Schlitze **36** eingesetzt sind und eingeschaltet sind) zu, die dann vorhanden sind, wenn das Computersystem **10** zuerst eingeschaltet bzw. hochgefahren wird.

[0504] Jede PCI-PCI-Brücken-Schaltung (z. B. PCI-PCI-Brücke **26**, **48**) in diesem Konfigurations-Register-Raum **1252** (Fig. 49) besitzt ein Neben-Bus-Zahl-Register **1218** und ein Sekundär-Bus-Zahl-Register **1220**. Das Neben-Bus-Zahl-Register **1218** enthält eine Neben-Bus-Zahl, die die höchste PCI-Bus-Zahl ausgangsseitig der PCI-PCI-Brücken-Schaltung ist, und das Sekundär-Bus-Zahl-Register **1220** enthält eine Sekundär-Bus-Zahl, die die PCI-Bus-Zahl des PCI-Busses unmittelbar ausgangsseitig der PCI-PCI-Brücken-Schaltung ist. Demzufolge definieren die Werte, gespeichert in dem Neben- **1218** und dem Sekundär-**1220** Bus-Zahl-Register, den Bereich von PCI-Bus-Zahlen, die ausgangsseitig der PCI-PCI-Brücken-Schaltung vorhanden sind.

[0505] Der Konfigurations-Register-Raum **1252** besitzt auch ein Primär-Bus-Zahlen-Register **1222**. Das Pri-

mär-Bus-Zahl-Register **1222** enthält die Zahl des PCI-Busses, angeordnet unmittelbar eingangsseitig der PCI-PCI-Brücken-Schaltung.

[0506] Die System-Steuereinheit/Host-Brücken-Schaltung **18** besitzt auch das Neben- **1218** und Sekundär-**1220** Bus-Zahl-Register. Nach einer Konfiguration enthält das Neben-Bus-Zahl-Register **1218** der Schaltung **18** die maximale PCI-Bus-Zahl, die in dem Computersystem vorhanden ist. Das Sekundär-Bus-Zahl-Register **1220** der Schaltung **18** enthält die Bus-Zahl Null, da dem PCI-Bus unmittelbar ausgangsseitig der Schaltung **18** (PCI-Bus **24**) immer die Bus-Zahl Null zugeordnet wird.

[0507] Im Gegensatz zu dem bekannten System erkennt die CPU **14**, dass einer der Schlitze **36**, der zu Anfang eingeschaltet bzw. hochgefahren ist oder leer ist, einen oder mehrere zusätzliche PCI-Buse) (vorhanden auf der Karte **802**, eingesetzt in dem Schlitz **36**, zu Anfang heruntergefahren) in das Computersystem **10** hinein einführen kann, nachdem das Computersystem **10** bereits eingeschaltet bzw. hochgefahren und konfiguriert ist. Dementsprechend reserviert, während einer anfänglichen Konfiguration, die CPU **14** Speicherraum, I/O-Raum und eine vorbestimmte Zahl (z. B. eins oder drei) von PCI-Bus-Zahlen für irgendeinen Schlitz **36**, der heruntergefahren oder leer ist.

[0508] Demzufolge müssen die PCI-PCI-Brücken-Schaltungen des Computersystems **10** nicht rekonfiguriert werden, um die Karte **807** aufzunehmen, die vor kurzem eingeschaltet worden ist. Nur die PCI-PCI-Brücken-Schaltungen der Karte **807**, die vor kurzem eingeschaltet wurde, muss konfiguriert werden. Der Rest des Computersystems **10** verbleibt unverändert.

[0509] Als ein Teil des Resource-Reservierungs-Prozesses baut ein Basic-Input/Output-System (BIOS), gespeichert in dem ROM **23** und verdeckt in den Speicher **20** eingegeben (und schreibgeschützt), eine Tabelle auf, die Resource-Bereiche spezifiziert, die für die Schlitze **36** reserviert werden. Diese Tabelle umfasst eine Bus-Zahl, Speicher und I/O-Resource-Bereiche zur Verwendung beim Konfigurieren einer PCI-Vorrichtung, die neu zu dem System **10** hinzugefügt worden ist. Das Betriebssystem verwendet diese Tabelle, um zu bestimmen, welche Resourcen reserviert worden sind und welche Resourcen für eine Konfigurierung der neu hinzugefügten PCI-Vorrichtung verfügbar sind.

[0510] Wie in Fig. 45 dargestellt ist, ordnet, in einem rekursiven PCI-Konfigurations-Programm, bezeichnet als BUS_ASSIGN, die CPU **14** Konfigurations-Register **1252** für PCI-Bus-Zahlen und -Programme der PCI-PCI-Brücken-Schaltungen entsprechend zu. Die CPU **14** nimmt dies durch Abtasten eines PCI-Busses zu einem Zeitpunkt für PCI-Vorrichtungen vor. Das BUS_ASSIGN Programm ist Teil des BIOS, gespeichert in dem ROM **23**, und wird dazu verwendet, zu Anfang das Computersystem **10**, nach einem Einschalten, zu konfigurieren.

[0511] Die CPU **14** setzt zuerst **1000**, der Wert eines Such-Parameters PCI-Bus, gleich zu dem Wert eines anderen Such-Parameters CURRENT_PCI_BUS, und initialisiert **1000** Such-Parameter FCN und DEV. Der Parameter PCI-Bus zeigt die Bus-Zahl des PCI-Busses an, der momentan durch die CPU **14** abgetastet wird, und wenn das BUS_ASSIGN Programm zuerst durch die CPU **14** ausgeführt wird, zeigt der Parameter PCI-Bus die Bus-Zahl Null an.

[0512] Der Parameter CURRENT_PCI_BUS zeigt die nächste PCI-Bus-Zahl an, die zum Zuordnen durch die CPU verfügbar ist, und wenn das Programm BUS_ASSIGN zuerst durch die CPU **14** ausgeführt wird, zeigt der Parameter CURRENT_PCI_BUS eine Bus-Zahl Null an. Die Parameter FCN und DEV zeigen die momentane PCI-Funktion und die PCI-Vorrichtung, jeweils, an, die momentan durch die CPU **14** abgetastet werden.

[0513] Die CPU **14** bestimmt, **1001**, ob der Parameter PCI_BUS eine Bus-Zahl Null anzeigt, und falls dies der Fall ist, stellt die CPU **14** das Sekundär-Bus-Zahl-Register **1220** der System-Steuereinheit/Host-Brücken-Schaltung **18** gleich zu Null ein. Die CPU **14** findet dann, **1004**, die nächste PCI-PCI-Brücken-Schaltung oder den Schlitz **36**, der heruntergefahren bzw. ausgeschaltet ist oder leer ist, auf dem PCI-Bus, angezeigt durch den Parameter PCI_BUS. Zu Zwecken einer Bestimmung, ob die nächste, gefundene PCI-Vorrichtung eine PCI-PCI-Brücken-Schaltung ist oder nicht existiert (ein abgeschalteter oder leerer Schlitz), versucht die CPU **14**, von einem Wert eines EIN-Wort-Vendor-ID-Register, angeordnet in dem Konfigurations-Raum jeder PCI-Vorrichtung, zu lesen. Ein Wert von „hFFFF" (wobei die Vorsilbe „h" eine hexadezimale Darstellung bezeichnet) wird reserviert und nicht durch irgendeinen Vendor bzw. Lieferanten verwendet. Falls die versuchte Lesung von dem Vendor-ID-Register zu einem Wert „HFFFF" zurückkehrt, dann zeigt dies an, dass keine PCI-Vorrichtung vorhanden ist.

[0514] Falls die CPU **14** bestimmt, **1006**, dass dort keine weiteren, nicht vorgefundenen PCI-PCI-Brücken-Schaltungen oder Schlitze **36** vorhanden sind, die heruntergefahren sind oder leer sind, und zwar auf dem PCI-Bus, angezeigt durch den Parameter PCI BUS, wird eine Zurückführung von dem letzten Aufruf, vorgenommen zu dem BUS_ASSIGN Programm, vorgenommen. Ansonsten bestimmt die CPU **14**, **1008**, ob eine andere PCI-PCI-Brücken-Schaltung vorgefunden wurde, und falls nicht, erhöht die CPU **14**, **1010**, den Parameter CURRENT_PCI_BUS, da ein Schlitz bzw. Einsteckplatz **36**, der eingeschaltet oder leer ist, gefunden wurde, und findet, **1004**, die nächste PCI-PCI-Brücken-Schaltung oder einen Schlitz **36**, der abgeschaltet oder leer ist. Demzufolge reserviert, durch Erhöhen, **1010**, des Parameters CURRENT_PCI_BUS, die CPU **14** effektiv eine Bus-Zahl für den Schlitz **36**, der abgeschaltet oder leer ist. Alternativ kann die CPU **14** mehr als eine Bus-Zahl für den Schlitz **36** reservieren, der abgeschaltet oder leer ist.

[0515] Falls die CPU **14** eine PCI-PCI-Brücken-Schaltung fand, dann stellt die CPU **14**, **1012**, die Primär-Bus-Zahl der PCI-PCI-Brücken-Schaltung gleich zu dem Parameter CURRENT_PCI_BUS ein. Die CPU **14** erhöht dann, **1014**, den Parameter CURRENT_PCI_BUS und stellt, **1016**, die sekundäre Bus-Zahl der PCI-PCI-Brücke gleich zu der neuen Bus-Zahl, angezeigt durch den Parameter CURRENT_PCI_BUS, ein.

[0516] Die CPU **14** stellt dann, **1018**, die Neben-Bus-Zahl der gefundenen PCI-PCI-Brücken-Schaltung gleich zu der maximalen, möglichen Zahl von PCI-Bussen ein, und zwar durch Schreiben zu dem Neben-Bus-Zahl-Register **1218**. Dieser Wert für das Neben-Bus-Zahl-Register **1218** ist temporär und ermöglicht der CPU **14**, zusätzliche, ausgangsseitige PCI-PCI-Brücken-Schaltungen oder Schlitze **36** zu finden und zu programmieren, die abgeschaltet oder leer sind.

[0517] Die CPU **14** findet zusätzliche, ausgangsseitige PCI-PCI-Brücken-Schaltungen oder Schlitze **36**, die abgeschaltet oder leer sind, durch Aufbewahren, **1022**, der Parameter PCI_BUS, DEV und FCN und jeweils aufrufen, **1022**, des BUS_ASSIGN Programms. Die CPU **14** speichert dann wieder, **1024**, die Werte für die Parameter PCI_BUS, DEV und FCN, und kehrt zu dem letzten Aufruf des BUS_ASSIGN Programms zurück, um den Parameter CURRENT_PCI_BUS mit der nächsten PCI-Bus-Zahl zu aktualisieren, die durch die CPU **14** zugeordnet werden sollen.

[0518] Die CPU **14** aktualisiert dann, **1026**, die Neben-Bus-Zahl der aufgefundenen PCI-PCI-Brücke durch Einstellen, **1026**, der Neben-Bus-Zahl gleich zu dem Parameter CURRENT_PCI_BUS. Demzufolge schließt dies die Zuordnung der PCI-Bus-Zahl zu der gefundenen PCI-PCI-Brückenschaltung und zusätzlichen, ausgangsseitigen PCI-PCI-Brücken-Schaltungen und Einsteckplätzen bzw. Schlitzen **36** ab, die abgeschaltet oder leer sind. Die CPU **14** findet dann, **1004**, die nächste PCI-PCI-Brücken-Schaltung oder den Schlitz **36**, die abgeschaltet oder leer sind, und zwar auf dem PCI-Bus, angezeigt durch den Parameter PCI BUS.

[0519] Wie in Fig. 46 dargestellt ist, führt, nachdem die PCI-Bus-Zahlen zugeordnet sind, die CPU **14** ein Speicher-Raum-Zuordnungs-Programm, bezeichnet als MEM_ALLOC, aus, um Speicher-Raum für die PCI-Funktionen und Schlitze **36** zuzuordnen, die abgeschaltet oder leer sind. Die CPU **14** initialisiert zuerst, **1028**, Such-Parameter, verwendet beim Unterstützen der CPU **14** beim Auffinden der angeordneten PCI-Funktionen und Schlitze **36**, die abgeschaltet oder leer sind.

[0520] Die CPU **14**, findet dann, **1030**, die nächste PCI-Funktion oder die Schlitze **36**, die abgeschaltet oder leer ist. Falls die CPU **14** bestimmt, **1032**, dass alle PCI-Funktionen und alle Schlitze bzw. Einsteckplätze **36**, die abgeschaltet oder leer sind, einem Speicherraum zugeordnet worden sind, kehrt die CPU **14** von dem Programm MEM_ALLOC zurück. Ansonsten bestimmt die CPU **14**, **1032**, ob eine PCI-Funktion gefunden wurde.

[0521] Falls dies der Fall ist, ordnet die CPU **14** Speicher-Resourcen zu, **1038**, wie dies durch die PCI-Funktion spezifiziert ist. Ansonsten wird einer der Schlitze bzw. Einsteckplätze **36**, der abgeschaltet ist oder leer ist, vorgefunden, und die CPU **14** ordnet eine Fehler-Speicher-Größe und eine Speicher-Ausrichtung für den Schlitz **36** zu, **1036**. Die Fehler-Speicher-Größe kann entweder eine vorbestimmte Größe sein, bestimmt vor einem Einschalten des Computersystems **10**, oder eine Größe, die nach einer Bestimmung der Speicher-Resourcen, erforderlich durch das Computersystem **10**, bestimmt ist.

[0522] Wenn Speicherraum zugeordnet wird, programmiert die CPU **14** Speicher-Basis- **1212** und Speicher-Grenzen- **1214** Register der PCI-PCI-Brücken-Schaltungen, die eingangsseitig der gefundenen PCI-Funktion vorhanden sind. Die CPU **14** programmiert geeignet auch Basis-Adressen-Register der entsprechenden PCI-Vorrichtungen. Die CPU **14** findet dann, **1030**, die nächste PCI-Funktion oder den Schlitz **36**, der abgeschaltet oder leer ist.

[0523] Wie in Fig. 47 dargestellt ist, führt, nachdem die PCI-Bus-Zahlen zugeordnet sind, die CPU **14** ein I/O-Raum-Zuordnungs-Programm, bezeichnet als I/O_ALLOC, aus, um I/O/Raum für PCI-Funktionen und Schlitze **36**, die leer sind, zuzuordnen. Die CPU **14** initialisiert zuerst, **1040**, Such-Parameter, verwendet beim Unterstützen der CPU **14**, die zugeordneten PCI-Funktionen und Schlitze **36**, die abgeschaltet oder leer sind, zu finden.

[0524] Die CPU **14** findet, **1042**, die nächste PCI-Funktion oder den Schlitz **36**, der abgeschaltet oder leer ist. Falls die CPU **14** bestimmt, **1044**, dass alle PCI-Funktionen und Schlitze **36**, die abgeschaltet oder leer sind, einem I/O/Raum zugeordnet worden sind, kehrt die CPU **14** von dem I/O_ALLOC Programm zurück. Ansonsten bestimmt die CPU **14**, **1044**, ob eine PCI-Funktion gefunden wurde. Falls dies der Fall ist, ordnet die CPU **14**, **1050**, I/O-Resourcen zu, wie dies durch die PCI-Funktion spezifiziert ist. Ansonsten ordnen ein Schlitz **36**, der heruntergefahren ist oder leer vorgefunden wurde, und die CPU **14** eine Fehler-I/O-Größe und eine I/O-Ausrichtung für den Schlitz **36** zu, **1048**. Die Fehler-I/O-Größe kann entweder eine vorbestimmte Größe, bestimmt vor einem Abschalten des Computersystems **10**, oder eine Größe, bestimmt nach einer Bestimmung der I/O-Ressourcen, erforderlich durch das Computersystem **10**, sein.

[0525] Wenn ein I/O/Raum zugeordnet wird, dann programmiert die CPU **14** die I/O-Basis 1208 und begrenzt, **1012**, Register der PCI-PCI-Brücken-Schaltungen, eingangsseitig der PCI-Funktion oder des Schlitzes **36**. Die CPU **14** programmiert auch Basis-Adressen-Register der entsprechenden PCI-Vorrichtungen geeignet. Die CPU **14** findet dann, **1042**, die nächste PCI-Funktion oder den Schlitz **36**, der abgeschaltet oder leer ist.

[0526] Wie in Fig. 48 dargestellt ist, führt, nach einer anfänglichen Konfiguration, wenn eine Unterbrechung erzeugt ist, die anzeigt, dass einer der Hebel **802** geöffnet oder geschlossen ist, die CPU **14** ein Unterbrechungs-Service-Programm, bezeichnet als CARD_INT, aus. Die CPU **14** liest, **1052**, die Inhalte des Unterbrechungs-Registers **800**, um zu bestimmen, **1053**, ob der Hebel **802** geöffnet oder geschlossen worden ist. Falls die CPU **14** bestimmt, **1053**, dass der Hebel **802**, der die Unterbrechung verursacht, geöffnet wurde, kehrt die CPU **14** von dem Programm CARD_INT zurück.

[0527] Ansonsten schreibt die CPU **14**, **1054**, zu dem Schlitz-Freigabe-Register **817** und stellt, **1054**, das SO-Bit ein, um das Einschalten des Schlitzes **36** und der Karte **807**, eingesetzt in dem Schlitz **36**, zu initiieren. Die CPU **14** wartet dann (nicht dargestellt) auf die Karte **807**, um einzuschalten. Die CPU **14** greift dann, **1055**, auf den PCI-Bus auf der Karte zu, falls vorhanden. Die CPU **14** bestimmt dann, **1056**, ob die Karte **807**, die gerade eingeschaltet wurde, einen PCI-Bus besaß (und eine PCI-PCI-Brücken-Schaltung). Falls dies der Fall ist, bestimmt, **1057**, die CPU **14** die primären, sekundären und Unterprogramm-Bus-Zahlen, reserviert für den Schlitz **36**, in dem die Karte **807** eingeschaltet wurde. Die CPU **14** konfiguriert darauffolgend, **1058**, die PCI-PCI-Brücken-Schaltung auf der Karte **807**, die eingeschaltet wurde.

[0528] Die CPU **14** bestimmt dann, **1060**, die Stelle und die Größe von I/O-und Speicher-Räumen, reserviert für den Schlitz **36**. Die CPU **14** schreibt darauffolgend, **1062**, zu Basis-Adressen-Registern in dem PCI-Konfigurations-Header-Raum der Karte **807**, die eingeschaltet wurde. Die CPU **14** liest dann, **1064**, ein Unterbrechungs-Stift-Register in dem Konfigurations-Raum der Karte **807**, um zu bestimmen, **1066**, ob die Karte **807** Unterbrechungs-Anforderungen verwendet. Falls dies der Fall ist, schreibt die CPU **14**, **1068**, ein Unterbrechungs-Zeilen-Register in den Konfigurations-Raum der Karte **807** mit einer zugeordneten IRQ-Zahl.

[0529] Die CPU gibt dann, **1070**, Befehls-Register der Karte **870** frei, die in dem Konfigurations-Raum der Karte **807** angeordnet sind, und ermöglicht der Karte **807**, auf Speicher- und I/O-Zugriffe auf den PCI-Bus **32** anzusprechen. Die CPU **14** schreibt darauffolgend, **1072**, zu dem Unterbrechungs-Register **800**, um die Unterbrechungs-Anforderung zu löschen, und lädt, **1074**, einen Software-Vorrichtungs-Treiber für die Karte **807**. Die CPU **14** kehrt dann von dem Programm CARD_INT zurück.

BRÜCKEN-KONFIGURATION

[0530] Funktional bilden Brücken-Chips **26** und **48** eine PCI-PCI-Brücke zwischen PCI-Bussen **24** und **32**. Allerdings umfasst jeder Brücken-Chip einen Konfigurations-Raum, der unabhängig konfiguriert werden muss. Eine Lösung ist diejenige, zwei Brücken als unabhängige Vorrichtungen, eine Brücke bildend, zu behandeln, allerdings würde dies eine Modifikation des BIOS-Konfigurations-Programms erfordern. Die andere Lösung ist diejenige, das Kabel **28** als einen Bus zu definieren, so dass das Konfigurations-Programm den eingangsseitigen Brücken-Chip **26** als eine PCI-PCI-Brücke zwischen dem PCI-Bus **24** und dem Kabel **28** und dem ausgangsseitigen Brücken-Chip **48** als eine PCI-PCI-Brücke zwischen dem Kabel **28** und dem PCI-Bus **32** konfigurieren kann. Ein Vorteil dieser zweiten Lösung ist derjenige, dass Standard-PCI-Konfigurations-Zyklen lau-

fen können, um die Brücken-Chips **26** und **48** zu konfigurieren, falls sie zwei PCI-PCI-Brücken waren, wenn tatsächlich die zwei Brücken-Chips eine PCI-PCI-Brücke bilden.

[0531] Dabei sind zwei Typen von Konfigurations-Transaktionen auf dem PCI-Bus vorhanden: Typ 0 und Typ 1. Ein Konfigurations-Zyklus vom Typ 0 ist für Vorrichtungen auf dem PCI-Bus vorgesehen, auf dem der Konfigurations-Zyklus erzeugt ist, während ein Konfigurations-Zyklus vom Typ 1 für Vorrichtungen auf einem sekundären PCI-Bus, auf den über eine Brücke zugegriffen wird, vorgesehen ist. Fig. 51 stellt das Adressen-Format von Konfigurations-Zyklen vom Typ 0 und Typ 1 dar. Ein Konfigurations-Befehl vom Typ 0 wird durch Einstellen von PCI-Adressen-Bits AD[1:0] auf 00 während eines Konfigurations-Zyklus spezifiziert. Ein Konfigurations-Zyklus vom Typ 0 wird nicht über eine PCI-PCI-Brücke weitergeführt, sondern verbleibt lokal auf dem Bus, auf dem die Konfigurations-Transaktion vom Typ 0 erzeugt wurde.

[0532] Ein Kofigurations-Befehl vom Typ 1 wird durch Einstellen von Adressen-Bits AD[1:0] auf einen binären Wert 01 spezifiziert. Konfigurations-Befehle vom Typ 1 können durch eine PCI-PCI-Brücke zu irgendeinem Level in der PCI-Bus-Hierarchie weitergeführt werden. Schließlich wandelt eine PCI-PCI-Brücke einen Befehl vom Typ 1 zu einem Befehl vom Typ 0 um, um Vorrichtungen zu konfigurieren, die mit der sekundären Schnittstelle der PCI-PCI-Brücke verbunden sind.

[0533] Konfigurations-Parameter, gespeichert in den Konfigurations-Registern **105** oder **125** der Brücke, identifizieren die Bus-Zahlen für deren primäre PCI-Schnittstelle (Primär-Bus-Zahl) und sekundären PCI-Schnittstelle (Sekundär-Bus-Zahl) und eine nebengeordnete Bus-Zahl, die die höchste, nummerierte PCI-Bus-Unterordnung der Brücke angibt. Die Bus-Zahlen werden durch ein PCI-Konfigurations-Programm BUS_ASSIGN (Fig. 45) eingestellt. Zum Beispiel ist, in dem eingangsseitigen Brücken-Chip **26**, die Primär-Bus-Zahl des Busses **24**, die Sekundär-Bus-Zahl ist die Zahl des Kabels **28** und die Neben-Bus-Zahl ist die Zahl des Sekundär-PCI-Busses **32** oder die Zahl eines tieferen PCI-Busses, falls ein solcher existiert. In dem ausgangsseitigen Brücken-Chip **48** ist die Primär-Bus-Zahl die Zahl des Kabel-Busses **28**, die Sekundär-Bus-Zahl ist die Zahl des PCI-Busses **32** und die nebengeordnete Bus-Zahl ist die Zahl eines PCI-Busses, angeordnet tiefer in der PCI-Bus-Hierarchie, falls eine solche existiert.

[0534] Wie Fig. 53A zeigt, wird eine Erfassung von Konfigurations-Zyklen durch eine Logik in dem PCI-Target-Block **103** oder **121** in dem eingangsseitigen Brücken-Chip **26** oder dem ausgangsseitigen Brücken-Chip **48** jeweils behandelt. Ein Konfigurations-Zyklus vom Typ 0, erfasst auf dem eingangsseitigen Bus **24**, wird durch Aufstellen eines Signals TYP0_CFG_CYC_US, erzeugt durch ein UND-Gate **276**, angezeigt. Das UND-Gate **276** empfängt Signale UPSTREAM_CHIP, IDSEL (Chip-Select während einer Konfigurations-Transaktion), CFGCMD (Konfigurations-Befehl-Zyklus, der erfasst ist) und ADDR00 (Bits 1 und 0 sind beide 0'en). Ein Konfigurations-Zyklus vom Typ 0, erfasst durch den ausgangsseitigen Brücken-Chip **48**, wird durch ein Signal TYP0_CFG_CYC_DS, erzeugt durch ein UND-Gate **278**, angezeigt, das ein Signal S1_BL_IDSEL (IDSEL Signal für den ausgangsseitigen Brücken-Chip **48**), das Signal CFGCMD, das Signal ADDR00, ein Signal MSTR_ACTIVE (anzeigend, dass der Brücken-Chip **48** der Master auf einem sekundären PCI-Bus **32** ist), und den invertierten Zustand eines Signal UPSTREAM_CHIP empfängt.

[0535] Eine Erfassung eines Konfigurations-Zyklus vom Typ 1 durch das PCI-Target **103** in dem eingangsseitigen Brücken-Chip **26** wird durch Aufstellen eines Signals TYP1_CFG_CYC_US von einem UND-Gate **280** angezeigt, das Signale CFGCMD, ADDR01 (Bits 1 und 0 sind niedrig und hoch jeweils) und UPSTREAM_CHIP empfängt. Eine Erfassung eines Konfigurations-Zyklus vom Typ 1 auf dem ausgangsseitigen Bus **32** wird durch Aufstellen eines Signals TYP_CFG_CYC_DS von einem UND-Gate **282**, das die Signale CFGCMD, ADDR01 empfängt, und dem invertierten Zustand des Signals UPSTREAM_CHIP angezeigt.

[0536] Der Brücken-Chip, der eine Transaktion vom Typ 0 aufnimmt, verwendet das Register-Zahl-Feld **250** in der Konfigurations-Transaktions-Adresse, um auf das geeignete Konfigurations-Register zuzugreifen. Das Funktions-Zahl-Feld **252** spezifiziert eine von acht Funktionen, die in einer multi-funktionalen Vorrichtung während der Konfigurations-Transaktion durchgeführt werden soll. Eine PCI-Vorrichtung kann multi-funktional sein und kann solche Funktionen haben, wie eine Festplatten-Laufwerksteuereinheit, eine Speicher-Steuereinheit, eine Brücke, usw..

[0537] Wenn der Brücken-Chip **26** eine Konfigurations-Transaktion vom Typ 1 auf seinem eingangsseitigen Bus **26** sieht, kann er die Transaktion entweder ausgangsseitig weiterführen, die Transaktion zu einer Transaktion vom Typ 0 translatieren, die Transaktion zu einem speziellen Zyklus konvertieren oder die Transaktion ignorieren (basierend auf den Bus-Zahl-Parametern, gespeichert in den Konfigurations-Registern **105** oder **125**). Falls eine Transaktion weitergeführt wird, gelangt sie bis zu dem PCI-Master des Bestimmungs-Brü-

cken-Chips, um die Transaktion vom Typ 1 zu der entsprechenden, geeigneten Transaktion zu konvertieren. Falls ein Brücken-Chip die Transaktion selbst handhabt, dann spricht er durch Aufstellen des Signals DEVSEL_ auf den PCI-Bus an, und handhabt die Transaktion als eine normale, verzögerte Transaktion.

[0538] In einer Konfiguration-Transaktion vom Typ 1 wählt das Bus-Zahl-Feld **260** einen eindeutigen PCI-Bus in der PCI-Hierarchie aus. Ein PCI-Target-Block **103** führt einen Konfigurations-Zyklus vom Typ 1 von dem eingangsseitigen Chip **26** zu dem ausgangsseitiger Brücken-Chip **48** hindurch, falls ein Signal PASS_TYP1_DS durch ein UND-Gate **284** aufgestellt ist. Das UND-Gate **284** empfängt das Signal TYP1_CFG_CYC_US und ein Signal IN_RANGE (das Bus-Zahl-Feld **260** ist größer als oder gleich zu der gespeicherten, sekundären Bus-Zahl und geringer als oder gleich zu der gespeicherten Neben-Bus-Zahl). Der andere Eingang des UND-Gates **284** ist mit dem Ausgang eines ODER-Gates **286** verbunden, der einen Eingang mit dem Ausgang eines UND-Gates **288** verbunden besitzt, und wobei der andere Eingang den invertierten Zustand eines Signals SEC_BUS_MATCH aufnimmt. Demzufolge wird, falls ein Zyklus vom Typ 1 erfasst wird, das Signal IN_RANGE aufgestellt ist, und wenn das Bus-Zahl-Feld **260** nicht die gespeicherte, sekundäre Bus-Zahl anpasst, das Signal PASS_TYP1_DS aufgestellt. Falls das Bus-Feld **260** nicht die gespeicherte Sekundär-Bus-Zahl anpasst, dann werden die Bus-Vorrichtungen auf oder nach dem ausgangsseitigen Bus **32** adressiert. Das UND-Gate **288** wird auf hoch gesetzt und das Vorrichtungs-Zahl-Feld **258** zeigt an, dass das Target des Konfigurations-Zyklus vom Typ 1 der Konfigurations-Raum des ausgangsseitigen Brücken-Chips **48** ist. Falls dies wahr ist, wird die Konfigurations-Transaktion vom Typ 1 entlang des Kabels **28** zu dem ausgangsseitigen Brücken-Chip **48** für eine Translation einer Konfigurations-Transaktion vom Typ 0 weitergeführt. Das PCI-Target **121** in dem ausgangsseitigen Brücken-Chip **48** spricht auf die Transaktion an und liest von den und schreibt in die ausgangsseitigen Brücken-Konfigurations-Register **125** entsprechend zu der Transaktion vom Typ 0. Die Steuer-Stifte des ausgangsseitigen Chips werden angesteuert und Lese- und Schreib-Daten erscheinen auf dem ausgangsseitigen PCI-Bus **32**, falls eine Transaktion vom Typ 0 auf dem ausgangsseitigen Bus läuft (für Debug-Zwecke), obwohl jedes IDSEL auf dem ausgangsseitigen Bus **32** so blockiert wird, das keine Vorrichtung tatsächlich auf eine Transaktion vom Typ 0 anspricht.

[0539] Falls der PCI-Target-Block **103** in dem eingangsseitigen Brücken-Chip **26** eine Konfigurations-Transaktion vom Typ 1 auf seinen eingangsseitigen Bus **24** erfasst, mit einem Bus-Zahl-Feld gleich zu der gespeicherten Sekundär-Bus-Zahl (der Kabel-Bus **28**), allerdings nicht eine Vorrichtung 0 adressierend (suchen nach anderen Vorrichtungen auf dem Kabel-Bus **28**), dann ignoriert der Target-Block **103** die Transaktion auf dem primären Bus **26**.

[0540] Falls das PCI-Target **121** eine Konfigurations-Schreib-Transaktion vom Typ 1 (WR_hoch) auf dem sekundären PCI-Bus **32** erfasst, der ein Bus-Zahl-Feld außerhalb des Bereichs der Sekundär-Bus-Zahl und der Neben-Bus-Zahl besitzt (IN_RANGE niedrig), und falls die Vorrichtungs-Zahl **258**, die Funktions-Zahl **256** und die Register-Zahl **254** einen speziellen Zyklus anzeigen (SP_MATCH hoch), dann wird ein Signal PASS_TYP1_US durch ein UND-Gate **290** aufgestellt. Das UND-Gate **290** empfängt das Signal TYP1_CFG_CYC_DS, das Signal SP_MATCH, das Schreib/Lese-Strobe WR und den invertierten Zustand des Signal IN_RANGE. Wenn der PCI-Master **101** in dem eingangsseitigen Brücken-Chip **26** einen solchen Zyklus empfängt, lässt er einen speziellen Zyklus auf dem primären PCI-Bus **24** laufen.

[0541] Konfigurations-Transaktionen werden durch einen Brücken-Chip unter bestimmten Bedingungen ignoriert. Falls der Target-Block **103** in dem eingangsseitigen Brücken-Chip **26** eine Konfigurations-Transaktion vom Typ 1 auf dem PCI-Bus **24** erfasst (sein eingangsseitiger Bus) und das Bus-Zahl-Feld **260** geringer als die Sekundär-Bus-Zahl oder größer als die Neben-Bus-Zahl, gespeichert in dem Konfigurations-Raum des Brücken-Chips, ist, dann ignoriert der Target-Block **103** die Transaktion.

[0542] Falls der Target-Block **121** in dem ausgangsseitigen Brücken-Chip **48** eine Konfigurations-Transaktion vom Typ 1 auf dem sekundären PCI-Bus **32** erfasst (sein ausgangsseitiger Bus), und das Bus-Zahl-Feld **260** größer als oder gleich zu der Sekundär-Bus-Zahl oder geringer als oder gleich zu der Neben-Bus-Zahl, gespeichert in dem Konfigurations-Raum des Brücken-Chips, ist, dann ignoriert der Target-Block **121** die Transaktion. Zusätzlich werden Konfigurations-Befehle vom Typ 1, die zu der Eingangsseite hin gehen, ignoriert, falls ein Befehl vom Typ 1 nicht eine Konversion zu einer speziellen Zyklus-Transaktion spezifiziert, ungeachtet der Bus-Zahl, spezifiziert in dem Befehl vom Typ 1.

[0543] Wie Fig. 53B zeigt, überwacht der PCI-Master **101** oder **123** einen Konfigurations-Zyklus, übertragen über das Kabel **28**. Falls der PCI-Master **123** in dem ausgangsseitigen Brücken-Chip **48** eine Konfigurations-Transaktion vom Typ 1 von dem eingangsseitigen Brücken-Chip **26** erfasst, vergleicht das Bus-Zahl-Feld **260** mit der Primär-Bus-Zahl und der Sekundär-Bus-Zahl, gespeichert in dem Konfigurations-Raum des Brü-

cken-Chips **48**. Falls das Bus-Zahl-Feld **260** entweder die gespeicherte Primär-Bus-Zahl (d. h. Kabel **28**) oder die gespeicherte Sekundär-Bus-Zahl (Adressieren einer Vorrichtung direkt, verbunden mit dem ausgangsseitigen Bus **23**) anpasst, dann translatiert der ausgangsseitige Brücken-Chip **48** die Transaktion zu einer Transaktion vom Typ 0 (durch Einstellen von AD[1:0] = 00), wenn er die Konfigurations-Transaktion auf dem Bus weiterführt. Die Transaktion vom Typ 0 wird auf dem PCI-Bus **32** durch den PCI-Master-Block **123** durchgeführt.

[0544] Das nachfolgende sind Translationen, durchgeführt von Feldern in der Konfigurations-Transaktion vom Typ 1. Das Vorrichtungs-Zahl-Feld **258** in der Konfigurations-Transaktion vom Typ 1 wird durch den PCI-Master **123** decodiert, um eine eindeutige Adresse in der translatierten Transaktion vom Typ 0 auf dem sekundären Bus **32** zu erzeugen, wie dies in der Tabelle der Fig. 52 definiert ist. Die Sekundär-Adressen-Bits AD[31:16], decodiert von dem Vorrichtungs-Zahl-Feld **258**, werden durch den PCI-Master **123** verwendet, um die geeigneten Chip-Auswahl-Signale IDSEL für die Vorrichtungen auf dem sekundären PCI-Bus **32** zu erzeugen. Wenn das Adressen-Bit AD[15] gleich zu 1 ist, dann behält der Brücken-Chip **48** alle Adressen-Bits AD[31:16], gesetzt auf niedrig (kein IDSEL ist aufgestellt), bei. Das Register-Zahl-Feld **254** und das Funktions-Zahl-Feld **256** des Konfigurations-Befehls vom Typ 1 werden nicht modifiziert zu dem Konfigurations-Befehl vom Typ 0 hindurchgeführt. Das Funktions-Zahl-Feld **256** wählt acht Funktionen aus, und das Register-Zahl-Feld **254** wählt ein Doppel-Wort in dem Konfigurations-Register-Raum der ausgewählten Funktion aus.

[0545] Für eine Konfigurations-Transaktion vom Typ 1, zielmäßig vorgesehen zu dem ausgangsseitigen Brücken-Chip **48**, wandelt der Brücken-Chip **48** die Transaktion vom Typ 1 zu einer Transaktion vom Typ 0 um, als würde sie eine Vorrichtung auf dem ausgangsseitigen Bus **32** adressieren, allerdings werden die AD[31:16] Stifte auf 0'en gesetzt, so dass keine Sekundär-PCI-Bus-Vorrichtung ein IDSEL aufnimmt. Die PCI-Master-Logik **123** erfasst dies durch Aufstellen eines Signals TYP1_TO_INT0, angesteuert durch ein UND-Gate **262**. Das UND-Gate **262** empfängt ein Signal CFG_CMD (einen Konfigurations-Befehl-Zyklus anzeigend), den Ausgang eines ODER-Gates **264** und den invertierten Zustand des Signals UPSTREAM_CHIP (Translation Type-1-zu-Type-0 wird in dem eingangsseitigen Brücken-Chip **26** gesperrt). Das ODER-Gate **264** stellt seinen Ausgang auf hoch, falls ein Signal PRIM_BUS_MATCH aufgestellt ist (das Bus-Zahl-Feld **260** passt die gespeicherte, primäre Bus-Zahl an), oder falls die gespeicherte, primäre Bus-Zahl CFG2P_PRIM_BUS_NUM[7:0] gleich zu null ist (anzeigend, dass die Primär-Bus-Zahl in dem Konfigurations-Raum des Brücken-Chips **48** nicht durch das System BIOS bis jetzt konfiguriert worden ist und der momentane Konfigurations-Zyklus vom Typ 1 zu dem internen Konfigurations-Raum geht, um die Primär-Bus-Zahl des Brücken-Chips **48** zu programmieren).

[0546] Ein Signal TYP1_ TO_EXT0 wird durch ein UND-Gate **266** aufgestellt und spricht auf eine Anpassung zu einer gespeicherten Sekundär-Bus-Zahl an. Die Eingänge des UND-Gates **266** empfangen das Signal CFG_CMD, das Signal SEC_BUS_MATCH, den invertierten Zustand des Signals UPSTREAM_CHIP und den invertierten Zustand eines Signals SP_MATCH (nicht ein spezieller Zyklus). Das Signal TYP1_TO_EXT0 zeigt an, dass die konvertierte Konfigurations-Transaktion vom Typ 0 zu einer Vorrichtung zielmäßig auf dem sekundären PCI-Bus **32** geführt wird.

[0547] Das Signal TYP1_TO_INT0 wird zu dem 1-Eingang eines 4 : 1 Multiplexers **274** geliefert. Der 2-Eingang wird auf niedrig gelegt und der 0- und 3-Eingang des Multiplexers **274** nehmen ein Signal LTYP1_TO_INT0 von einem Flip-Flop **270** vom D-Typ auf. Der Auswahl-Eingang S1 des Multiplexers **274** empfängt ein Signal CMD_LATCH (FRAME_, aufgestellt für einen neuen Zyklus auf dem PCI-Bus **32**), und der Auswahl-Eingang S0 empfängt ein Signal P2Q_START_PULSE (das anzeigt, wenn es hoch ist, dass eine Adresse zu dem PCI-Bus **32** geschickt worden ist). Der Ausgang des Multiplexers **274** wird mit dem D-Eingang eines Flip-Flops **270** verbunden, das mit dem Signal PCLK getaktet wird und durch das Signal RESET gelöscht wird. Die IDSEL-Signale zu den Sekundär-Bus-Vorrichtungen werden durch Aufstellen eines Signals BLOCK_IDSEL von einem ODER-Gate **272** blockiert, das an seinen Eingängen Signale Q2P_AD[15] (keine Konversion wird entsprechend Tabelle 1 der Fig. 6 benötigt), TYP1_TO_INT0 und LTYP1_TO_INT0 empfängt. Das Signal LTYP1_TO_INT0 erweitert das Aufstellen des Signals BLOCK_IDSEL.

[0548] Wenn der PCI-Master **123** in dem ausgangsseitigen Brücken-Chip **48** eine Konfigurations-Transaktion vom Typ 1 von dem eingangsseitigen Brücken-Chip **26** empfängt, in dem das Bus-Zahl-Feld **260** größer als die gespeicherte, Sekundär-Bus-Zahl und geringer als oder gleich zu der gespeicherten Neben-Bus-Zahl ist, dann führt der PCI-Master-Block **123** die Transaktion vom Typ 1 zu dem Sekundär-PCI-Bus **32** unverändert weiter. Eine bestimmte andere Vorrichtung auf dem Sekundär-PCI-Bus **32**, z. B. eine andere Brücken-Vorrichtung **323** (Fig. 26B), wird die Konfigurations-Transaktion vom Typ 1 aufnehmen und sie zu dem sekundären Bus

(PCI-Bus **325**) weiterführen.

[0549] Eine Konfigurations-Transaktion vom Typ 1 zu einer speziellen Zyklus-Translation wird dann durchgeführt, wenn der PCI-Master **123** eine Konfigurations-Schreib-Transaktion vom Typ 1 von dem eingangsseitigen Brücken-Chip **26** empfängt und das Bus-Zahl-Feld **260** die gespeicherte Sekundär-Bus-Zahl anpasst und falls das Vorrichtungs-Zahl-Feld **258**, das Funktions-Zahl-Feld **256** und das Register-Zahl-Feld **254** einen speziellen Zyklus anzeigen (SP_MATCH ist hoch). Dies wird durch ein UND-Gate **268** angezeigt, das ein Signal TYP1_TO_SPCYC auf hoch setzt. Das UND-Gate **268** empfängt SP_MATCH, und Q2P_CBE_[0] (Befehl-Bit für einen speziellen Zyklus). Die Daten von der Konfigurations-Transaktion vom Typ 1 werden die Daten für den speziellen Zyklus an dem Bestimmungs-Bus. Die Adresse während eines speziellen Zyklus wird ignoriert.

BUS-FUNKTIONS-MONITOR

[0550] Der Bus-Monitor **127** (Fig. 3) umfasst eine Schaltung zum Speichern von Informationen, um bestimmte Bus-Funktions-Parameter zu berechnen. Die Parameter umfassen eine Bus-Nutzung, eine Bus-Effektivität und eine Lese-Daten-Effektivität. Eine Bus-Nutzung ist das Verhältnis der Zeit, die der Bus belegt ist, unter Durchführen einer Transaktion, zu einer vorgegebenen, globalen Zeitperiode. Eine Bus-Effektivität ist das Verhältnis der Zahl von PCI-Takt-Perioden, die tatsächlich für eine Daten-Übertragung verwendet werden, zu der gesamten Zahl von Taktperioden während der Bus-Beleg-Periode. Eine Lese-Daten-Effektivität ist das Verhältnis der Zahl der Lese-Daten-Bytes, auf die durch eine Vorrichtung auf dem Sekundär-PCI-Bus **32** zugegriffen ist, und zwar von der verzögerten Abschluss-Warteschlange (DCQ) **144** (Fig. 4), zu der gesamten Zahl von Daten-Bytes, abgerufen für diesen Master durch den Brücken-Chip **48**. Die Informationen, gespeichert in dem Bus-Monitor **127**, werden durch die System-Software wieder aufgesucht, um die erwünschten Parameter zu berechnen.

[0551] Wie Fig. 54A zeigt, zählt ein Global-Periode-Zeitgeber **1300** (der 32 Bits breit sein kann) eine gesamte Zeitperiode, während der die verschiedenen Parameter berechnet werden sollen. Der Zeitgeber **1300** wird auf den hexadezimalen Wert FFFFFFFF programmiert. Falls der PCI-Takt PCICLK2 bei 33 MHz läuft, dann beträgt die Zeitgeberperiode ungefähr 2 Minuten. Wenn sich der Zeitgeber **1300** auf 0 verringert, stellt er ein Signal GL_TIME_EXPIRE auf.

[0552] Der Bus-Monitor **127** umfasst 7 schlitz-spezifische Bus-Busy-Zähler **1302A**–**G**, wobei sechs der Zähler jeweils den 6 Schlitzen auf dem sekundären PCI-Bus **32** und einer der SIO **50** entsprechen. Die Bus-Busy-Zähler **1302A**–**G** werden dann gelöscht, wenn das Signal GL_TIME_EXPIRE aufgestellt ist. In Abhängigkeit davon, welche Bus-Vorrichtung eine Steuerung auf dem sekundären Bus **32** besitzt, erhöht sich der Bus-Busy-Zähler **1302** bei jedem PCI-Takt, in dem das Sekundär-PCI-Bus FRAME_ oder IRDY_ Signal aufgestellt ist. Der geeignete Eine der sieben Zähler wird durch eines der Erteilungs-Signale GNT[7:0]_ ausgewählt. Demzufolge wird, zum Beispiel, der Bus-Busy-Zähler **1302A** dann ausgewählt, wenn das Signal GNT[1]_ auf niedrig gesetzt ist, was anzeigt, dass der SIO der momentane Master auf dem sekundären PCI-Bus **32** ist.

[0553] Sieben Daten-Zyklus-Zähler **1306A**–**G** entsprechend, jeweils, zu den 6 Schlitzen auf dem sekundären PCI-Bus **32** und dem SIO **50**, führen die Zeit nach, während der eine Datenübertragung tatsächlich zwischen einem Master und einem Target während einer Transaktion auf dem PCI-Bus **32** auftritt. Der ausgewählte Daten-Zyklus-Zähler **1306** wird bei jedem PCI-Takt erhöht, bei dem die Sekundär-Bus- IRDY_ und TRDY_ Signale beide auf niedrig gesetzt sind. Die Daten-Zyklus-Zähler **1306A**–**G** werden dann gelöscht, wenn das Signal GL_TIME_EXPIRE aufgestellt ist.

[0554] Sechs DCQ-Daten-Zähler **1310A**–**F** sind in dem Bus-Monitor **127** zum Protokollieren der Menge an Daten, eingeladen in die DCQ-Puffer, umfasst. Die sechs DCQ-Daten-Zähler **1310A**–**F** entsprechen den 6 Schlitzen auf dem sekundären PCI-Bus **32**. Der ausgewählte DCQ-Daten-Zähler **1310** erhöht sich bei jedem PCI-Takt, indem verzögerte Lese-Abschluss- (Delayed Read Completion – DRC) Daten von dem Kabel **28** empfangen und in die Prefetch-Puffer hineingeladen werden.

[0555] Ein anderer Satz von Zählern, DCQ-Daten-Benutzungs-Zähler **1314A**–**F**, werden dazu verwendet, die Menge an Daten zu protokollieren, die in die DCQ **144** eingeladen sind, tatsächlich verwendet durch die 6 Schlitze auf dem sekundären PCI-Bus **32**. Der ausgewählte DCQ-Daten-Benutzungs-Zähler **1314** erhöht sich bei jedem PCI-Takt, in dem der sekundäre Bus-Masterdaten von dem entsprechenden DCQ-Puffer liest. Beide DCQ-Daten-Zähler **1310A**–F und DCQ-Daten-Benutzungs-Zähler **1314A**–**F** erhöhen sich bei jedem Daten-Zyklus ungeachtet der Zahl von Bytes, die tatsächlich übertragen werden. In den meisten Fällen beträgt die Zahl von Bytes, übertragen in jedem Daten-Zyklus, 4.

[0556] Wenn der Global-Perioden-Zeitgeber **1300** abläuft und das Signal GL_TIME_EXPIRE aufstellt, treten verschiedene Ereignisse auf. Zuerst lädt der Global-Perioden-Zeitgeber **1300** seinen Originalen Zähl-Wert wieder ein, der der hexadezimale Wert FFFFFFFF ist. Die Inhalte aller anderen Zähler, umfassend die Bus-Busy-Zähler **1302A**–**G** die Daten-Zyklus-Zähler **1306A**–**G**, die DCQ-Daten-Zähler **1310A**–**F** und die DCQ-Daten-Benutzungs-Zähler **1314A**–**F**, werden in Register **1304**, **1308**, **1312** und **1316** jeweils eingeladen. Die Zähler **1302**, **1306**, **1310** und **1314** werden dann auf 0 gelöscht. Der Global-Periode-Zeitgeber **1300** beginnt dann, erneut zu zählen, nachdem erneut mit seinem Original-Wert wieder geladen ist.

[0557] Das Signal GL_TIME_EXPIRE wird zu dem Unterbrechungs-Aufnahme-Block **132** zugeführt, der die Unterbrechung über das Kabel **28** zu dem Unterbrechungs-Ausgangs-Block **114** weiterführt, der wiederum eine Unterbrechung zu der CPU **14** erzeugt. Die CPU **14** antwortet auf die Unterbrechung durch Aufrufen eines Unterbrechungs-Händlers, um die Bus-Funktion-Analyse durchzuführen. Der Unterbrechungs-Händler greift auf die Inhalte der Register **1304**, **1308**, **1312** und **1316** zu, und berechnet die verschiedenen Parameter, umfassend die Bus-Benutzung, die Bus-Effektivität, die Prefetch-Effektivität-Parameter, zugeordnet den 6 Sekundär-Bus-Schlitzen bzw. -Einsteckplätzen und der SIO **50**.

[0558] Der Bus-Benutzungs-Parameter ist der Wert des Bus-Busy-Zählers **1302**, geteilt durch den Anfangswert des Global-Perioden-Zeitgebers **1300**, der der hexadezimale Wert FFFFFFFF ist. Demzufolge ist die Bus-Nutzung der Prozentsatz der gesamten, globalen Zeit, während der ein Bus-Master eine Bus-Transaktion durchführt.

[0559] Eine PCI-Transaktion umfasst eine Adressen-Phase und mindestens eine Daten-Übertragungs-Phase. Ein Bus-Master stellt das Signal FRAME_ auf, um den Beginn und die Dauer einer aktiven Bus-Transaktion anzuzeigen. Wenn das Signal FRAME_ weggenommen ist, zeigt dies an, dass die Transkaktion die End-Daten-Phase ist oder die Transaktion abgeschlossen worden ist. Das Signal IRDY_ zeigt an, dass der Bus-Master in der Lage ist, die momentane Daten-Phase der Bus-Transaktion abzuschließen. Während eines Schreibens zeigt das Signal IRDY_ an, dass gültige Daten auf dem Bus vorhanden sind. Während eines Lesens zeigt das Signal IRDY_ an, das der Master präpariert ist, um Lese-Daten anzunehmen. Das adressierte PCI-Target spricht auf die Bus-Transaktion durch Aufstellen des Signals TRDY_ an, um anzuzeigen, dass das Target in der Lage ist, die momentane Datenphase der Transaktion abzuschließen. Während eines Lesens zeigt das Signal TRDY_ an, dass gültige Daten auf dem Bus vorhanden sind; während eines Schreibens zeigt das Signal TRDY_ an, dass das Target präpariert wird, um Daten anzunehmen. Warte-Zustände können zwischen den Adressen- und Daten-Phasen und zwischen aufeinanderfolgenden Daten-Phasen der Bus-Transaktionen eingesetzt werden.

[0560] Während der Adressen-Phase oder den Warte-Zuständen tritt keine Daten-Übertragung tatsächlich auf.

[0561] Eine tatsächliche Daten-Übertragung tritt nur dann auf, wenn beide Signale IRDY_ und TRDY_ auf niedrig gesetzt sind. Um die Daten-, Übertragungs-Bus-Effektivität zu bestimmen, teilt der Unterbrechungs-Händler den Wert des Daten-Zyklus-Zählers **1306** durch den Wert des Bus-Busy-Zählers **1302**. Die Bus-Effektivität stellt die Menge an Zeit dar, während der eine Datenübertragung tatsächlich während einer Bus-Transaktion auftritt. Durch Berechnen dieses Werts kann sich das Computersystem über Target-Vorrichtungen bewusst werden, die viele Warte-Zustände erfordern und deshalb ineffektiv sind.

[0562] Der Brücken-Chip **48** kann Daten von dem primären PCI-Bus **26** abrufen und die Daten in der DCQ **144** speichern. Die DCQ **144** besitzt acht Puffer, wobei jeder einem sekundären Bus-Master zuordenbar ist. Zum Beispiel wird eine Speicher-Lese-Mehrfach-Transaktion, erzeugt durch einen sekundären Bus-Master, als Ziel an dem primären Bus vorgesehen, bewirken, dass die Brücke **26**, **48** insgesamt 8 Cache-Zeilen von dem Speicher **20** abruft und sie in die DCQ **144** einlädt. Eine Speicher-Lese-Zeilen-Transaktion wird bewirken, dass die PCI-PCI-Brücke **26**, **48** eine Zeile von Daten von dem Speicher **20** abruft. Zusätzlich kann, wie in Verbindung mit den Fig. 75 und Fig. 79 beschrieben ist, die PCI-PCI-Brücke **26**, **48** eine Lesepromotion durchführen, die eine Lese-Anforderung von einem Sekundär-Bus-Master zu einer Lese-Anforderung für einen größeren Block an Daten umwandelt. In diesen Fällen existiert eine Möglichkeit, dass nicht alle der abgerufener. Daten durch den Bus-Master verwendet werden. In diesem Fall werden nicht-gelesene Daten ausgesondert, was die Lese-Daten-Effektivität reduziert. Ein Messen der Lese-Daten-Effektivität ermöglicht System-Designern zu verstehen, wie ein Bus-Master Lese-Daten, abgerufen durch den Brücken-Chip **26**, **48**, von dem primären Bus **24**, verwendet.

[0563] Wie Fig. 54B zeigt, erhöht sich der Zähler **1310** an der ansteigenden Flanke des Takts PCLK, falls das

Signal DCQ_DATA_RECEIVED[X], X = 2–7, aufgestellt ist, was anzeigt, dass vier Bytes an Daten durch einen DCQ-Puffer, zugeordnet einem Master X von dem Kabel **28**, empfangen werden. Der Zähler **1310** gibt einen Zähl-Wert DCQ_Data[X][20:0], X = 2–7, aus, was auf Null gelöscht wird, wenn das Signal GL_TIME_EXPIRE aufgestellt ist.

[0564] Der Zähler **1314** erhöht sich an der ansteigenden Flanke des Takts PCLK, falls ein Signal DCQ_DATA_TAKEN[X], X = 2–7, aufgestellt ist, was anzeigt, dass vier Bytes an Daten von einem DCQ-Puffer zugeordnet zu Master X, gelesen sind. Der Zähler **1314** wird dann gelöscht, wenn das Signal GL_TIME_EXPIRE hoch ist.

[0565] Um die Menge der DCQ-Daten zu bestimmen, die tatsächlich durch die Vorrichtungen auf dem sekundären PCI-Bus **32** verwendet werden, wird die Prefetch-Effektivität durch den Unterbrechungs-Handler berechnet. Dies wird dadurch bestimmt, dass das Verhältnis des Werts in dem Zähler **1314**, der die DCQ-Daten verwendet, zu dem Wert des DCQ-Daten-Zählers **1310** gesetzt wird. Obwohl sogar nicht alle Daten, übertragen in die Prefetch-Puffer oder übertragen aus diesen heraus, 4 Bytes breit sind, ist das Verhältnis eng durch eine Annahme angenähert, dass alle Daten-Phasen dieselbe Zahl von Bytes übertragen.

[0566] In Abhängigkeit der berechneten Parameter kann ein Benutzer oder der Computerhersteller besser die Funktion des Computersystems verstehen. Zum Beispiel könnte dann, wenn eine Bus-Effektivität niedrig ist, die PCI-Vorrichtung, die eingesetzt ist, durch ein unterschiedliches Teil des Computerherstellers ersetzt werden. Eine Kenntnis der DCQ-Lese-Daten-Effektivität ermöglicht dem Computerhersteller, seinen DCQ-Abruf-Algorithmus zu ändern, um besser die Effektivität zu verbessern.

VERWENDUNG VON NEBEN-BUS-VORRICHTUNGEN

[0567] Wie in Fig. 88 dargestellt ist, führen sechs Erweiterungskarten, eingesetzt in die sechs Erweiterungskartenschlitze bzw. -Steckplätze **36a**–**f**, Bus-Vorrichtungen **1704**–**1708** ein, die zu der CPU **14** untergeordnet sind, und Bus-Vorrichtungen **1701**–**1702**, die zu einem I_2O Prozessor **1700** untergeordnet sind. Obwohl alle Neben-Bus-Vorrichtungen **1701**–**1708** mit dem gemeinsamen PCI-Bus **32** verbunden sind, erscheinen die I_2O-Neben-Vorrichtungen **1701**–**1702** zu der CPU **14** nur so, dass sie über den I_2O-Prozessor **1700** adressierbar sind und nicht direkt über den PCI-Bus **32** adressierbar sind. Deshalb dient der PCI-Bus **32** sowohl als ein I_2O-Neben-Vorrichtungs-Bus als auch als ein Neben-Vorrichtungs-Bus der CPU **14**.

[0568] Zu Zwecken eines Verhinderns, dass die CPU **14** die I_2O-Neben-Vorrichtungen **1701**–**1702** als Vorrichtung des PCI-Busses **32** erkennt, umfasst der Brücken-Chip **48** eine Logik **1710** (Fig. 90) zum Verhindern, dass die I_2O-Neben-Vorrichtungen **1701**–**1702** auf Konfigurations-Zyklen ansprechen, die durch die CPU **14** laufen. Der Erweiterungskasten **30** umfasst auch eine Multiplexing-Schaltung **1712**, die mit dem Unterbrechungs-Aufnahme-Block **132** des Brücken-Chips **48** zusammenarbeitet, um Unterbrechungs-Anforderungen zu maskieren, die von den I_2O-Neben-Vorrichtungen **1701**–**1702** ausgehen, dass diese zu der CPU **14** propagieren. Unterbrechungs-Anforderungen, die von den I_2O-Neben-Bus-Vorrichtungen **1701**–**1702** ausgehen, werden durch den Unterbrechungs-Aufnahme-Block **132** zu dem I_2O-Prozessor **1700** umgeleitet. Der I_2O-Prozessor **1700** konfiguriert die I_2O-Neben-Vorrichtungen **1701**–**1702**; empfängt und verarbeitet Unterbrechungs-Anforderungen, die von den I_2O-Neben-Vorrichtungen **1701**–**1702** ausgehen; und steuert einen Betrieb der I_2O-Neben-Vorrichtungen, wie dies durch die CPU **14** geleitet wird.

[0569] Nach einem Einschalten des Computersystems **10** und wenn eine Karte **807** eingeschaltet bzw. hochgefahren ist (d. h. eine neue Bus-Vorrichtung ist an dem PCI-Bus **32** eingeführt), tastet der I_2O-Prozessor **1700** den PCI-Bus **32** ab, um I_2O-Neben-Bus-Vorrichtungen zu identifizieren. Zu Zwecken eines Identifizierens des Typs einer Bus-Vorrichtung (I_2O-Neben-Bus-Vorrichtung oder Neben-Vorrichtung der CPU **14**) lässt der I_2O-Prozessor **1700** Konfigurations-Zyklen auf dem PCI-Bus **32** laufen, um das Vorrichtungs-Identifikations-Wort (Vorrichtungs-ID) jeder Bus-Vorrichtung zu lesen. Die Vorrichtungs-ID ist in dem Konfigurations-Header-Raum aller PCI-Vorrichtungen angeordnet. Der I_2O-Prozessor **1700** speichert die Ergebnisse dieser Abtastung in einem I_2O-Neben-Register **1729** mit sechs Bits (Fig. 93) innerhalb des I_2O-Prozessors **1700**, der durch die CPU **14** zugänglich ist. Bits null bis fünf des Registers **1729** sind Schlitzen bzw. Steckplätzen **36a**–f jeweils zugeordnet. Ein Wert von „1" für ein Bit zeigt an, dass der zugeordnete Schlitz **36** eine Bus-Neben-Vorrichtung zu der CPU **14** besitzt, und ein Wert von „0" für ein Bit zeigt an, dass der zugeordnete Schlitz **36** eine Bus-Neben-Vorrichtung zu dem I_2O-Prozessor **1700** besitzt.

[0570] Der I_2O-Prozessor **1700** kann in irgendeinen der Schlitze bzw. Steckplätze **36a**–**f** eingesetzt werden. Zu Zwecken eines Identifizierens, welcher Schlitz **36**, falls irgendeiner vorhanden ist, einen I_2O-Prozessor ent-

hält, tastet die CPU **14** den PCI-Bus **32** ab und liest die Vorrichtungs-ID der Bus-Vorrichtungen, verbunden mit dem Bus **32**. Die CPU **14** versucht nicht, irgendwelche Vorrichtungen **1704**–**1708** auf dem Bus **32** zu konfigurieren bis ein Host-Konfigurations-Freigabe-Bit **1726** (Fig. 94) innerhalb des I_2O-Prozessors **1700** der CPU **14** anzeigt, dass der I_2O-Prozessor **1700** seine Identifikation von I_2O-Neben-Vorrichtungen **1701**–**1702** auf dem Bus **32** abgeschlossen hat. Das Host-Konfigurations-Freigabe-Bit **1726** besitzt einen Wert von „0" (Wert bei einem Einschalten), um eine Konfiguration der Vorrichtungen auf dem Bus **32** durch die CPU **14** zu sperren, und einen Wert „1", um eine Konfiguration der CPU **14** der Nebenvorrichtungen **1704**–**1708** der CPU **14** auf dem Bus **32** freizugeben. Wenn die CPU **14** Bus-Vorrichtungen auf dem Bus **32** konfiguriert, „sieht" die CPU **14** nicht die I_2O-Neben-Vorrichtungen **1701**–**1702**, und zwar aufgrund der Maskierung durch die Logik **1710**, wie dies nachfolgend beschrieben ist.

[0571] Nachdem das Host-Freigabe-Konfigurations-Bit **1726** eingestellt ist, liest die CPU **14** die Inhalte des I_2O-Neben-Registers **1729** und überträgt die gelesenen Inhalte zu einem Sechs-Bits-I_2O-Neben-Register **1428** (Fig. 91) des Brücken-Chips **48**. Das Register **1728** zeigt den Neben-Status (Neben- I_2O-Prozessor **1700** oder Neben-CPU **14**) der Bus-Vorrichtungen in derselben Art und Weise wie das Register **1729** an. Bevor die CPU **14** zu dem Register **1728** schreibt, enthält das Register **1728** alle „Eins'en" (Werte bei einem Einschalten), was der CPU **14** ermöglicht, den Bus **32** nach dem I_2O-Prozessor **1700** abzutasten. Der Unterbrechungs-Aufnahme-Block **132** verwendet das Register **1728**, um zu identifizieren, welche Unterbrechungs-Anforderungen, empfangen durch den Block **132**, zu der CPU **14** geführt werden sollten, und welche Unterbrechungs-Anforderungen, empfangen durch den Block **132**, zu dem I_2O-Prozessor **1700** für eine Verarbeitung geführt werden sollten. Weiterhin verwendet die Logik **1710** die Inhalte des Registers **1728**, um eine Erkennung durch die CPU **14** der I_2O-Neben-Vorrichtungen **1701**–**1702** von der CPU **14** zu blockieren.

[0572] Zu Zwecken eines Anzeigens zu dem Unterbrechungs-Empfangs-Block **132**, welche Bus-Vorrichtung, falls irgendeine vorhanden ist, ein I_2O-Prozessor ist, stellt die CPU **14** ein Bit eines I_2O-Schlitz-Registers **1730** (**Fig.** 32) ein, dessen Bits 0–5 den Schlitzen **36a**–f jeweils entsprechen. Für dieses Register **1730**, angeordnet innerhalb des Brücken-Chips **48**, zeigt ein Wert von „0" für ein Bit an, dass der zugeordnete Schlitz **36** keinen I_2O-Prozessor besitzt, und ein Wert von „1" für das Bit zeigt an, dass der zugeordnete Schlitz **36** einen I_2O-Prozessor besitzt.

[0573] Wie in Fig. 90 dargestellt ist, umfasst die Logik **1710** ein Multi-Bit-UND-Gate **1711**, das Signale AD_IDSEL[5:0] zu Adressen/Daten-Leitungen des Busses **32** liefert, um Vorrichtungen auf dem Bus **32** während Konfigurations-Zyklen auszuwählen. Das UND-Gate **1711** empfängt ein Sechs-Bit-Signal ENABLE[5:0], das Bits besitzt, die für Bits des I_2O-Neben-Registers **1728** Indikativ sind und zu diesen entsprechen. Das UND-Gate **1711** empfängt auch die typischen Identifikations-Auswahl-Signale SLOT_IDSEL[5:0], geliefert durch den Brücken-Chip **48**, zum Auswählen von Vorrichtungen auf den Bus **32**, während Konfigurations-Zyklen. Deshalb werden die Signale ENABLE[5:0] dazu verwendet, selektiv die Signale SLOT_IDSEL[5:0] von dem PCI-Bus **32** zu maskieren, wenn Konfigurations-Zyklen durch die CPU **14** laufen.

[0574] Zu Zwecken eines Kontrollierens der Bestimmung von Unterbrechungs-Anforderungen von den Schlitzen **36a**–**d**, werden die Vier-Standard-PCI-Unterbrechungs-Anforderungs-Signale (INTA#, INTB#, INTC# und INTD#), geliefert durch jeden Schlitz **36**, zu einer Multiplexing-Schaltung **1712** geführt (Fig. 88). Die Multiplexing-Schaltung **1712** serialisiert die PCI-Unterbrechungs-Anforderungs-Signale, empfangen von den Schlitzen **36**, und liefert die Signale zu dem Unterbrechungs-Emfpangs-Block **132** über vier zeitmultiplexierte-, serielle Unterbrechungs-Anforderungs-Signale: INTSDA#, INTDSB#, INTSDC# und INTSDD#.

[0575] Wie in Fig. 89 dargestellt ist, liefert der Unterbrechungs-Empfangs-Block **132** Unterbrechungs-Anforderungs-Signale für die CPU **14** zu dem Unterbrechungs-Ausgabe-Block **114** über ein zeit-mulitplexiertes, serielles Unterbrechungs-Anforderungs-Signal INTSDCABLE#. Der Unterbrechungs-Empfangs-Block **132** liefert Unterbrechungs-Anforderungs-Signale für den I_2O-Prozessor **1700** über ein zeit-multiplexiertes, serielles Unterbrechungs-Anforderungs-Signal INTSDIIO#, geliefert über eine PCI INTC# Leitung **1709** des Busses **32** zu dem I_2O-Prozessor **1700**.

[0576] Der Unterbrechungs-Ausgabe-Block **114** liefert die Unterbrechungs-Anforderungen, bestimmt über die CPU **14**, zu einer oder mehreren der Standard-PCI-Unterbrechungs-Anforderungs-Leitungen (INTA#, INTB#, INTC# und INTD#) des PCI-Busses **24**. Eine Unterbrechungs-Steuereinheit **1900**, extern zu dem Brücken-Chip **26**, empfängt die Unterbrechungs-Anforderungen von den PCI-Unterbrechungs-Anforderungs-Leitungen des PCI-Busses **24**. Die Unterbrechungs-Steuereinheit **1900** priorisiert die Unterbrechungs-Anforderungen (die Unterbrechungs-Anforderungen von anderen Vorrichtungen auf dem PCI-Bus **24** umfassen können) und liefert sie zu der CPU **14**. Der Unterbrechungs-Ausgabe-Block **114** kann entweder asynchron (wenn

er sich in einem asynchronen Mode befindet) die Unterbrechungs-Anforderungs-Signale zu den Unterbrechungs-Anforderungs-Leitungen des PCI-Busses **24** liefern oder kann sie seriell (wenn er sich in einem seriellen Mode befindet) die Unterbrechungs-Anforderungs-Signale zu der INTA# Leitung des PCI-Busses **24** führen, wie dies weiter nachfolgend beschrieben ist.

[0577] Wie in Fig. 95 dargestellt ist, repräsentieren alle der zeit-multiplexierten, seriellen Daten-Signale deren Daten über einen Unterbrechungs-Zyklus **1850**, der acht aufeinanderfolgende Zeit-Teile (T0–T7) aufweist. Die Dauer jedes Zeit-Teils ist ein Zyklus des PCI-Takt-Signals CLK. Jeder Zeit-Teil stellt einen „Snapshot" über den Status von einem oder mehreren Unterbrechungs-Anforderungs-Signalen dar. Wie in Fig. 99 dargestellt ist, stellt das Signal INTSDA# die abgetasteten INTA# Unterbrechungs-Anforderungs-Signale von den Schlitzen **36a**–**f** dar. Das Signal INTSDB# stellt die abgetasteten INTB# Unterbrechungs-Anforderungs-Signale von den Schlitzen **36a**–**f** dar. Das Signal INTSDC# stellt die abgetasteten INTC# Unterbrechungs-Anforderungs-Signale von den Schlitzen **36a**–**f** dar. Das Signal INTSDD# stellt die abgetasteten INTD# Unterbrechungs-Anforderungs-Signale von den Schlitzen **36a**–**f** dar. Zu Zwecken eines Kombinierens der Unterbrechungs-Signale INTSDA#–D# in das Signal INTSDIIO# verknüpft der Unterbrechungs-Empfangs-Block **132** logisch die Signale INTSDA#–D# zusammen mit UND, während gleichzeitig Unterbrechungs-Anforderungs-Signale, bestimmt für die CPU **14**, maskiert werden. Ähnlich verknüpft, zu Zwecken eines Kombinierens der Unterbrechungs-Signale INTSDA#–D# in das Signal INTSDCABLE# der Unterbrechungs-Aufnahme-Block **132** logisch mit UND die Signale INTSDA#–D# zusammen, während simultan Unterbrechungs-Anforderungs-Signale, bestimmt für die CPU **14**, maskiert werden.

[0578] Für den Zweck eines Instruierens des Unterbrechungs-Ausgangs-Blocks **114**, wenn ein anderer Unterbrechungs-Zyklus **1850** beginnt, liefert der Unterbrechungs-Aufnahme-Block **132** ein Synchronisations-Signal INTSYNCCABLE# zu dem Unterbrechungs-Ausgangs-Block **114**. Die abfallende, oder negative, Flanke des Signals INTSYNCCABLE# zeigt an, dass der Zeit-Teil T0 des Unterbrechungs-Zyklus **1850**, übertragen über das Signal INTSDCABLE# an der nächsten, positiven Flanke des CLK Signals beginnt. Ein Signal INTSYNCIIO# wird in einer analogen Weise verwendet, um einen ankommenden Zeit-Schlitz T0 des Unterbrechungs-Zyklus **1850**, übertragen über das Signal INTSDIIO#, anzuzeigen. Das Signal INTSYNCIIO# wird durch den Unterbrechungs-Empfangs-Block **132** zu dem I_2O-Prozessor **1700** über eine PCI INTD# Leitung **1710** des Busses **32** geliefert. Für den Zweck eines Instruierens der Multiplexing-Schaltung **1712**, wenn ein anderer Unterbrechungs-Zyklus **1850** über die Unterbrechungs-Signale INTSDA#–D# zu übertragen ist, liefert der Unterbrechungs-Empfangs-Block **132** ein Synchronisations-Signal INTSYNC# zu der Multiplexing-Schaltung **1712**. Die abfallende, oder negative Flanke des Signals des INTSYNC# zeigt an, dass die Multiplexing-Schaltung **1712** den Zeit-Teil T0 der Signale INTSDA#–D# an der nächsten, positiven Flanke des CLK-Signals übertragen sollte.

[0579] Wie in Fig. 96 dargestellt ist, umfasst die Multiplexing-Schaltung **1712** vier Multiplexer **1741**–**1744**, die die Signale INTSDA#, INTSDB#, INTSDC# und INTSDD# jeweils liefern. Die Auswahl-Eingänge der Multiplexer **1741**–**1744** empfangen ein Zeit-Teil-Signal SLICEIN[2:0], das dazu verwendet wird, die Zeit-Teile T0–T7 der Signale INTSDA#–D# anzuzeigen. Die INTA-D# Unterbrechungs-Anforderungs-Signale von den Schlitzen **36** werden zu Eingängen der Multiplexer **1741**–**1744** jeweils geliefert.

[0580] Das Signal SLICEIN[2:0] wird durch den Ausgang eines Drei-Bit-Zählers **1745** geliefert, der an der positiven Flanke des PCI-Takt-Signals CLK getaktet wird. Das Unterbrechungs-Synchronisations-Signal INTSYNC# wird durch einen getakteten Freigabe-Eingang des Zählers **1745** empfangen. An der negativen Flanke des Signals INTSYNC# setzt sich der Zähler **1745** auf Null zurück (SLICEIN[2:0] entspricht Null). Der Zähler **1745** erhöht den Wert, angezeigt durch das SLICEIN[2.0] Signal, bis das SLICEIN[2:0] Signal gleich zu „7" ist, wo es verbleibt, bis der Zähler **1745** erneut durch das INTSYNC# Signal zurückgesetzt wird.

[0581] Wie in Fig. 97A dargestellt ist, umfasst, zu Zwecken eines Protokollierens der Zeit-Teile T0–T7, der Unterbrechungs-Aufnahme-Block **132** einen Drei-Bit-Zähler **1750**, der an der positiven Flanke des CLK-Signals getaktet wird. Der Zähler **1750** liefert ein Ausgangs-Signal SL1[2:0], das durch den Auswahl-Eingang eines 3 × 8 Decodierers **1752** empfangen wird. Der Decodierer **1752** liefert ein Acht-Bit-Signal G_CNTR[7:0], wobei das aufgestellte Bit des Signals G_CNTR[7:0] den Zeit-Teil der Signale INTSDIIO# und INTSDCABLE# anzeigt.

[0582] Das INTSYNC# Signal wird durch den Ausgang eines Invertierers **1774** geliefert, der das signifikanteste Bit des G_CNTR[7:0] Signals, G_CNTR[7], empfängt. Obwohl das INTSYNC# Signal auf niedrig während des Zeit-Teils T7 gepulst wird, könnte der Unterbrechungs-Aufnahme-Block **132** alternativ mehrere Zyklen des CLK-Signals nach Beenden eines Unterbrechungs-Zyklus **1850**, vor einem Pulsen des INTSYNC# Signals auf

niedrig, warten. Die Signale INTSYNCCABLE# und INTSYNCIIO# werden beide durch den Ausgang eines Invertierers **1755** geliefert, der das Bit G_CNTR[**0**] empfängt.

[0583] Ein zusätzliches Unterbrechungs-Anforderungs-Singal CAY_INT# für die CPU **14** wird durch die SIO-Schaltung **50** geliefert. Das CAY_INT# Signal wird logisch mit UND mit den INTSDA#–D# Signalen während des Zeit-Teils T0 verknüpft. Das CRY_INT# Signal wird durch den Ausgang eines UND-Gates **1756** geliefert, das ein SIO_CMPL# Signal, das SI_INTR# Signal und ein I^2C_INT# Signal aufnimmt. Das SIO_CMPL# Signal wird aufgestellt, oder auf niedrig angesteuert, wenn die SIO-Schaltung **50** einen Seriell-Ausgangs-Prozess abgeschlossen hat. Das I^2C_INT# Signal wird aufgestellt, oder auf niedrig angesteuert, um einen Abschluss einer Transaktion über einen I^2C-Bus (nicht dargestellt) anzuzeigen, verbunden mit dem Brücken-Chip **48**. Das I_2C_INT# Signal wird ansonsten weggenommen, oder auf niedrig angesteuert.

[0584] Zu Zwecken eines Maskierens von Unterbrechungs-Anforderungen erzeugt der Unterbrechungs-Empfangs-Block **132** vier Maskierungs-Signale MASKA, MASKB, MASKC und MASKD. Wenn das MASKA Signal aufgestellt ist, oder auf hoch angesteuert ist, während eines bestimmten Zeit-Teils (T0–T7) des Signals INTSDA#, wird eine Unterbrechungs-Anforderung, angezeigt durch das Seriell-Unterbrechungs-Signal INTSDA# während dieses bestimmten Zeit-Schlitzes, von der CPU **14** maskiert. Falls das MASKA Signal weggenommen ist, oder auf niedrig angesteuert ist, während des bestimmten Zeit-Teils, wird die Unterbrechungs-Anforderung, angezeigt durch das Seriell-Unterbrechungs-Signal INTSDA# von dem I_2O-Prozessor **1700** maskiert. Die MASKB-D Signale funktionieren ähnlich zu den Maskierungs-Unterbrechungs-Anforderungen, geliefert durch die Signale INTSDB#–D#.

[0585] Wie in Fig. 97B dargestellt ist, liefert ein Multiplexer **1758** das MASKA Signal. Der Auswahl-Eingang des Multiplexers **1758** empfängt das SL1[2:0] Signal. Die acht Eingänge des Multiplexers **1758** empfangen invertierte IIO_SUB[5:0] Signale, die für entsprechende Bits des I_2O-Neben-Registers **1728** Indikativ sind. Die Signale IIO_SUB[5:0] werden mit den geeigneten Eingängen des Multiplexers **1758** so verbunden, dass dann, wenn das INTSDA# Signal den Unterbrechungs-Status für einen bestimmten Schlitz bzw. Einsteckplatz **36** anzeigt, das MASKA Signal gleichzeitig das zugeordnete Bit des Registers **1728** für diesen Schlitz **36** anzeigt. Drei andere Multiplexer **1760**, **1762** und **1764** liefern die Signale MASKB, MASKC und MASKD jeweils. Ähnlich zur Erzeugung des MASKA Signals, sind die Signale IIO_SUB[5:0] mit den geeigneten Eingängen von Multiplexern **1760**, **1762** und **1764** so verbunden, dass die MASKB, MASKC und MASKD Signale das Bit des Registers **1728**, zugeordnet zu dem Schlitz, dargestellt durch die Signale INTSDB#, INTSDC# und INTSDD#, anzeigen. Die Multiplexer **1760**–**1764** empfangen das Signal SL1[2:0] an deren Auswahl-Eingängen.

[0586] Wie in Fig. 97C dargestellt ist, umfasst der Unterbrechungs-Aufnahme-Block **132** auch zwei Multiplexer **1768** und **1770**, die zwei Maskierungssignale IIOTS_D und IIOTS_C, verwendet dazu, die INTD# und INTC# Signale zu maskieren, geliefert durch die Schlitz-Unterbrechungs-Leitungen des I_2O-Prozessors **1700**, liefern, da die Leitungen **1709** und **1713** dazu verwendet werden, die Signale INTSDIIO# und INTSYNCIIO# jeweils zu dem I_2O-Prozessor **1700** zu liefern. Die Auswahl-Eingänge beider Multiplexer **1768** und **1770** empfangen das Signal SL1[2:0], und die Signal-Eingänge der Multiplexer **1768** und **1770** empfangen Signale IIOSLOT[5:0], die für die entsprechenden Bits des I_2O-Schlitz-Registers **1730** Indikativ sind. Die Signale IIOSLOT[5:0] werden mit den geeigneten Eingängen von Multiplexern **1768** und **1770** so verbunden, dass dann, wenn die INTSDC#–D# Signale den Unterbrechungs-Status für einen bestimmten Schlitz **36** anzeigen, das IIOSLOT[5:0] Signal, ausgewählt durch die Multiplexer **1768** und **1770**, gleichzeitig das zugeordnete Bit des Registers **1730** für diesen Schritt **36** anzeigt.

[0587] Wie in Fig. 97D dargestellt ist, werden sechs UND-Gates **1772**–**1782** dazu verwendet, um die Signale INTSDA#–INTSDD# zu kombinieren und ausgewählte Unterbrechungs-Anforderungs-Signale von der CPU **14** zu maskieren. Das UND-Gate **1772** empfängt ein invertiertes ECC_ERR_DOWN# Signal (aufgestellt dazu, um einen Fehler, erfasst durch den Chip **48b** bei Kabel-Übertragungen, anzuzeigen) und das Bit G_CNTRL[0]. Das UND-Gate **1774** empfängt ein invertiertes INTSDA# Signal und das MASKA Signal. Das UND-Gate **1776** empfängt ein invertiertes INTSDB# Signal und das MASKB Signal. Das UND-Gate **1778** empfängt ein invertiertes INTSDC# Signal, das MASKC Signal und das IIOTS_C Signal. Das UND-Gate **1780** empfängt ein invertiertes INTSDC# Signal, das MASKD Signal und das IIOTS_D Signal. Das UND-Gate **1782** empfängt ein invertiertes CAY_INT Signal und das G_CNTRL Signal.

[0588] Die Ausgänge der UND-Gates **1772**–**1782** sind als Eingänge zu einem ODER-Gate **1784** verbunden, das seinen Ausgang mit dem Signal-Eingang eines Flips-Flops **1786** vom D-Typ verbunden besitzt. Das Flip-Flop **1786** wird an der positiven Flanke des CLK-Signals getaktet, und der Einstell-Eingang des Flip-Flops **1786** empfängt das RST-Signal. Der invertierende Ausgang des Flip-Flops **1786** liefert das INTSDCABLE# Si-

gnal.

[0589] Vier UND-Gates **1790**–**1796** werden dazu verwendet, die INTSDA#–D# Signale zu kombinieren und ausgewählte Unterbrechungs-Anforderungs-Signale von dem I_2O-Prozessor **1700** zu maskieren. Das UND-Gate **1790** empfängt ein invertiertes INTSDA# Signal und ein invertiertes MASKA Signal. Ein anderer Eingang des UND-Gates **1790** ist mit dem Ausgang eines NOR-Gates **1802** verbunden, das das INTSDA# Signal während der Zeit-Teile T0 und T7 maskiert, da keine Karten- und Unterbrechungsanforderungen in diesen Zeit-Teilen umfasst sind. Das NOR-Gate **1802** empfängt die Bits G_CNTRL[0] und G_CNTRL[7]. Das UND-Gate **1732** empfängt ein invertiertes INTSDB# Signal und ein invertiertes MASKB Signal. Ein anderer Eingang des UND-Gates **1792** ist mit dem Ausgang eines NOR-Gates **1804** verbunden, der oder das das INTSDB# Signal während der Zeit-Teile T1 und T4 maskiert, da keine Karten-Unterbrechungs-Anforderungen in diesen Zeit-Teilen umfasst sind. Das NOR-Gate **1802** empfängt die Bits G_CNTRL[1] und G_CNTRL[4].

[0590] Das UND-Gate **1794** empfängt ein invertiertes INTSDC# Signal und ein invertiertes MASKC Signal. Ein anderer Eingang des UND-Gates **1794** ist mit dem Ausgang eines NOR-Gates **1806** verbunden, der das INTSDC# Signal während der Zeit-Teile T2 und T5 maskiert, da keine Karten-Unterbrechungs-Anforderungen in diesen Zeit-Teilen umfasst sind. Das NOR-Gate **1806** empfängt die Bits G_CNTRL[2] und G_CNTRL[5]. Das UND-Gate **1796** empfängt ein invertiertes INTSDD# Signal und ein invertiertes MASKD Signal. Ein anderer Eingang des UND-Gates **1796** ist mit dem Ausgang eines NOR-Gates **1808** verbunden, die das INTSDD# Signal während der Zeit-Teile T3 und T6 maskiert, da keins Karten-Unterbrechungs-Anforderungen in diesen Zeit-Teilen umfasst sind. Das NOR-Gate **1808** empfängt die Bits G_CNTRL[3] und G_CNTRL[6].

[0591] Die Ausgänge der UND-Gates **1790**–**1796** sind als Eingänge mit einem ODER-Gate **1798** verbunden, das einen Ausgang mit dem Signal-Eingang eines Flip-Flops **1800** vom D-Typ verbunden besitzt. Das Flip-Flop **1800** wird auf der positiven Flanke des CLK-Signals getaktet und der Einstell-Eingang des Flip-Flops **1800** empfängt das RST-Signal. Der invertierende Ausgang des Flip-Flops **1800** liefert das INTSDIIO# Signal.

[0592] Wie in Fig. 98 dargestellt ist, umfasst der Unterbrechungs-Ausgangs-Block **114** einen Drei-Bit-Zähler **1820** von einem gemeinsamen Design zu dem Zähler **1745**. Der Zähler **1820** wird an der positiven Flanke des Signals CLK getaktet, liefert ein Ausgangs-Signal G_CNTR2[2:0] und beginnt ein Zählen von 0 an bis 7, nachdem er durch das INTSYNC# Signal zurückgesetzt ist.

[0593] Zu Zwecken, das INTSYNCCPU# Signal zu liefern, umfasst der Unterbrechungs-Ausgangs-Block **114** ein Flip-Flop **1822** vom D-Typ, das an der positiven Flanke des CLK-Signals getaktet wird. Der Einstell-Eingang des Flip-Flops **1822** empfängt das RST-Signal und der Signal-Eingang des Flip-Flops **1822** empfängt das INTSYNCCABLE# Signal. Der nicht-invertierende Ausgang des Flip-Flops **1822** liefert das INTSYNCCPU# Signal.

[0594] Zu Zwecken, das INTSDCPU# Signal zu liefern, umfasst der Unterbrechungs-Ausgangs-Block **114** ein Flip-Flop **1824** vom D-Typ, das an der positiven Flanke des CLK-Signals getaktet wird. Der Einstell-Eingang des Flip-Flops **1824** empfängt das RST-Signal und der Signal-Eingang des Flip-Flops **1824** empfängt das INTSDCABLE# Signal. Der nicht-invertierende Ausgang des Flip-Flops **1824** liefert das INTSDCPU# Signal.

[0595] Die Unterbrechungs-Anforderungen, empfangen durch den Unterbrechungs-Empfangs-Block **114**, werden zu der Unterbrechungs-Steuereinheit **1900** entweder asynchron oder seriell zugeführt. In dem asynchronen Mode werden die Unterbrechungs-Anforderungen zu den vier PCI-Unterbrechungs-Leitungen (herkömmlich bezeichnet auch als „Barber Poling") auf dem PCI-Bus **24** aufgelistet, wie dies in Fig. 100 dargestellt ist.

[0596] Zu Zwecken eines Haltens der Unterbrechungs-Informationen, geliefert durch das INTSDCABLE# Signal, umfasst der Unterbrechungs-Ausgangs-Block **114** ein Acht-Bit-Register **1826**. Alle Signal-Eingänge empfangen das INTSDCABLE# Signal. Die Lade-Freigabe-Eingänge von Bits 0–7 empfangen die Bits G_CNTR[0]–G_CNTR[7] jeweils. Deshalb wird, zum Beispiel, während des Zeit-Teils T4, ein Bit 3 mit dem Wert, dargestellt durch das INTSDCABLE# Signal, geladen. Bits 0 (dargestellt durch ein INT_A1 Signal) und 4 (dargestellt durch ein INT_A2 Signal) werden in ein CPUINTA# Signal hinein aufgelistet. Bits 1 (dargestellt durch ein INT_B1 Signal) und 5 (dargestellt durch ein INT_B2 Signal) werden in ein CPUINTB# Signal hinein aufgelistet. Bits 2 (dargestellt durch ein INT_C1 Signal) und 6 (dargestellt durch ein INT_C2 Signal) werden in ein CPUINTC# Signal hinein aufgelistet. Bits 3 (dargestellt durch ein INT_D1 Signal) und 7 (dargestellt durch ein INT_D2 Signal) werden in ein CPUINTD# Signal hinein aufgelistet.

[0597] Vier ODER-Gates **1828**–**1834** liefern die Signale CPUINTA#, CPUINTB#, CPUINTC# und CPUINTD#,

die zu den PCI-Unterbrechungs-Leitungen INTA#, INTB#, INTC# und INTD# jeweils des PCI-Busses **24** geliefert werden. Das ODER-Gate **1828** besitzt einen Eingang mit dem Ausgang eines UND-Gates **1836** verbunden. Das UND-Gate nimmt ein invertiertes CM-Signal auf. Das Signal CM wird durch ein Bit eines Konfigurations-Registers des Brücken-Chips **26** geliefert und wird aufgestellt, oder auf hoch angesteuert, um den asynchronen Mode anzuzeigen, und wird weggenommen, oder auf niedrig angesteuert, um den synchronen Mode anzuzeigen. Das UND-Gate **1836** empfängt auch das Signal INT_A1, das Signal INT_A2, und ein Signal ECC_ERR_UP (verwendet dazu, einen Fehler in Kabel-Übertragungen anzuzeigen).

[0598] Das ODER-Gate **1828** besitzt einen Eingang mit dem Ausgang eines UND-Gates **1838** verbunden. Das UND-Gate **1838** empfängt das CM-Signal und das INTSDCPU# Signal. Ein anderer Eingang des UND-Gates **1838** ist mit dem Ausgang eines ODER-Gates **1848** verbunden. Das ODER-Gate **1848** empfängt das ECC_ERR_UP Signal und das Bit G_CNTR2[0].

[0599] Das ODER-Gate **1830** besitzt einen Eingang mit dem Ausgang eines UND-Gates **1840** verbunden und einen Eingang mit dem Ausgang eines UND-Gates **1842** verbunden. Das UND-Gate **1840** empfängt ein invertiertes CM-Signal, das Signal INT_B1, und das Signal INT_B2. Das UND-Gate **1842** empfängt das Signal CM und ein invertiertes Bit G_CNTR2[0] (verwendet dazu, das „sync" Signal zu der Unterbrechungs-Steuereinheit **1900** während des seriellen Modes zu liefern).

[0600] Das ODER-Gate **1832** besitzt einen Eingang mit dem Ausgang eines UND-Gates **1844** und mit einem Eingang, der das CM-Signal empfängt, verbunden. Das UND-Gate **1844** empfängt ein invertiertes CM-Signal, das INT_C1 Signal und das INT_C2 Signal. Das ODER-Gate **1834** besitzt einen Eingang mit dem Ausgang eines UND-Gates **1846** und mit einem Eingang, der das CM-Signal aufnimmt, verbunden. Das UND-Gate **1846** empfängt ein invertiertes CM-Signal, das INT_D1 Signal und das INT_D2 Signal.

[0601] Andere Ausführungsformen liegen innerhalb des Schutzumfangs der nachfolgenden Ansprüche. Zum Beispiel können unterschiedliche Typen von Bussen verwendet werden, einschließlich irgendeines Host-Busses oder eines peripheren Busses. Auch kann der Brücken-Chip z. B. eine Host-zu-Peripher-Bus-Brücke, eine Host-zu-Host-Bus-Brücke oder eine Peripher-zu-Peripher-Bus-Brücke sein.

Patentansprüche

1. Computersystem, das aufweist:
eine Brückenvorrichtung, verbunden zwischen Datenbussen (**24**, **32**), wobei die Brückenvorrichtung (**26**, **48**) Transaktionspuffer (**2042**–**2048**), geeignet zum Speichern individueller Daten-Transaktionen, zum Zuführen von einem Datenbus zu dem anderen Datenbus, umfasst, gekennzeichnet dadurch,
dass die Brückenvorrichtung (**26**, **48**) eine Puffer-Management-Logik (**140**) umfasst, die so betreibbar ist, um eine Daten-Transaktion, gehalten in der Brückenvorrichtung (**26**, **48**), zu einem ersten Transaktionspuffer zuzuweisen, und um einen zweiten Transaktionspuffer zu der Daten-Transaktion zuzuweisen, um zu ermöglichen, dass Daten in den zweiten, zugewiesenen Transaktionspuffer dann überlaufen, wenn die Daten-Transaktion nicht als abgeschlossen angezeigt wird, nachdem der erste Transaktionspuffer voll ist.

2. System nach Anspruch 1, wobei die Puffer-Management-Logik (**140**) so betreibbar ist, um den zweiten Transaktionspuffer nur dann zuzuweisen, wenn der zweite Transaktionspuffer nicht Daten, die einer anderen Daten-Transaktion, gespeichert in der Brückenvorrichtung (**26**, **48**), zugeordnet sind, enthält.

3. System nach Anspruch 1 oder Anspruch 2, wobei die Puffer-Management-Logik (**140**) so betreibbar ist, um einen dritten Transaktionspuffer zu der Transaktion zuzuweisen, wenn die Daten, die der Transaktion zugewiesen sind, den zweiten Transaktionspuffer zum Überlaufen bringen.

4. System nach einem der Ansprüche 1 bis 3, wobei, nachdem die Brückenvorrichtung (**26**, **48**) damit beginnt, die Daten, gespeichert in dem ersten Transaktionspuffer, zuzuführen, die Puffer-Management-Logik (**140**) so betreibbar ist, um den ersten Transaktionspuffer zu der Daten-Transaktion wieder zuzuweisen, falls die Daten, die der Transaktion zugeordnet sind, den zweiten Transaktionspuffer zum Überlaufen bringen.

5. System nach einem der Ansprüche 1 bis 4, wobei die Puffer-Management-Logik (**140**) ein Überlauf-Register (**2092**) umfasst, das mindestens ein Bit, zugeordnet zu dem zweiten Transaktionspuffer, besitzt, wobei die Puffer-Management-Logik so betreibbar ist, um das Bit in dem Überlauf-Register, zugeordnet zu dem zweiten Transaktionspuffer, in Abhängigkeit einer Speicherung von ersten Transaktionsdaten in dem zweiten Transaktionspuffer, einzustellen.

6. System nach einem der Ansprüche 1 bis 5, wobei die Transaktion, die den entsprechenden Transaktionspuffer zum Überlaufen bringt, eine Speicherschreibtransaktion aufweist.

7. System nach einem der Ansprüche 1 bis 5, wobei die Transaktion, die den ersten Transaktionspuffer zum Überlaufen bringt, eine gepostete Speicherschreibtransaktion aufweist.

8. System nach einem der Ansprüche 1 bis 7, wobei mindestens einer der Busse einen PCI-Bus aufweist.

9. System nach einem der Ansprüche 1 bis 8, wobei die Brückenvorrichtung eine PCI-zu-PCI-Brücke aufweist.

10. Verfahren zum Handhaben von Daten in einem Computersystem, das eine Datenspeichervorrichtung auf einem ersten Datenbus (**24**) und eine anfordernde Vorrichtung, die eine Daten-Transaktion auf einen zweiten Datenbus (**32**) initiiert, besitzt, wobei der erste und der zweite Datenbus (**24**, **32**), verbunden mit einer Brückenvorrichtung (**26**, **48**), Transaktionspuffer (**2042**–**2048**) umfassen, die dazu geeignet sind, individuelle Daten-Transaktionen zum Zuführen von einem Datenbus zu dem anderen zu speichern, wobei das Verfahren durch die Schritte gekennzeichnet ist:
Zuweisen von Daten, die einer Daten-Transaktion, gehalten in der Brückenvorrichtung (**26**, **48**), zugewiesen sind, zu einem ersten Transaktionspuffer, und wenn die Daten-Transaktion nicht als abgeschlossen angezeigt wird, nachdem der erste Transaktionspuffer voll ist, Zuweisen eines zweiten Transaktionspuffers, um die Überlaufdaten aufzunehmen.

11. Verfahren nach Anspruch 10, das weiterhin ein Zuweisen des zweiten Transaktionspuffers nur dann, wenn der zweite Transaktionspuffer keine Daten enthält, die einer anderen Daten-Transaktion zugeordnet sind, aufweist.

12. Verfahren nach Anspruch 10 oder Anspruch 11, das weiterhin ein sequenzielles Übertragen von Daten von dem ersten und dem zweiten Transaktionspuffer zu dem zweiten Datenbus aufweist.

13. Verfahren nach Anspruch 10 oder Anspruch 11, das weiterhin ein Zuweisen eines dritten Transaktionspuffers, um Transaktionsdaten zu empfangen, wenn die Transaktionsdaten den zweiten Speicherpuffer zum Überlaufen bringen, aufweist.

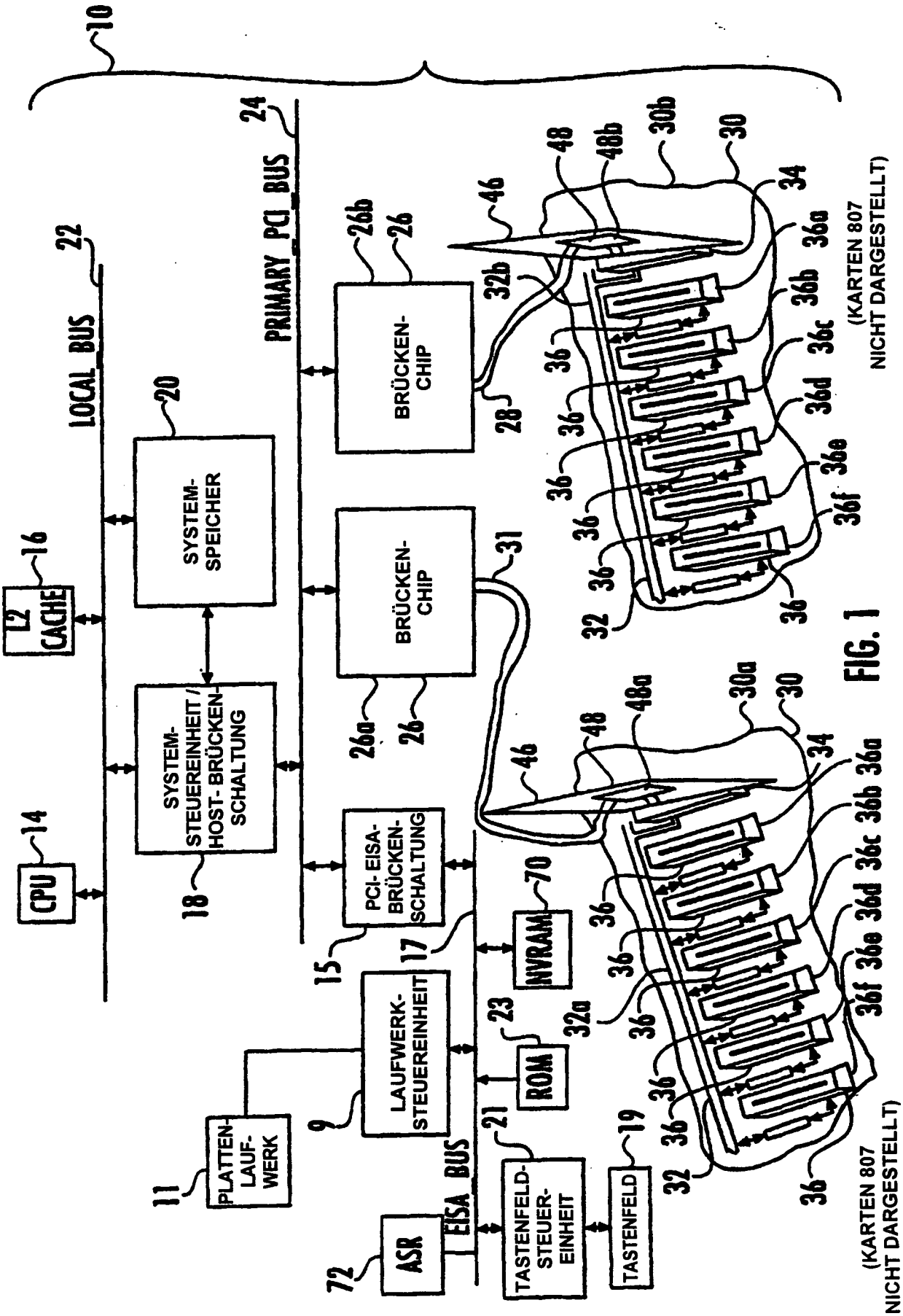
14. Verfahren nach Anspruch 10 oder Anspruch 11, das weiterhin ein Übertragen der Daten von dem ersten Transaktionspuffer zu dem zweiten Bus und danach wieder Zuweisen des ersten Transaktionspuffers, um Daten, die der Transaktion zugeordnet sind, zu empfangen, wenn die Daten den zweiten Puffer zum Überlaufen bringen, aufweist.

15. Verfahren nach einem der Ansprüche 10 bis 14, wobei die Transaktion eine Schreibtransaktion aufweist.

16. Verfahren nach einem der Ansprüche 10 bis 14, wobei die Transaktion eine gepostete Speicherschreibtransaktion aufweist.

Es folgen 127 Blatt Zeichnungen

Anhängende Zeichnungen



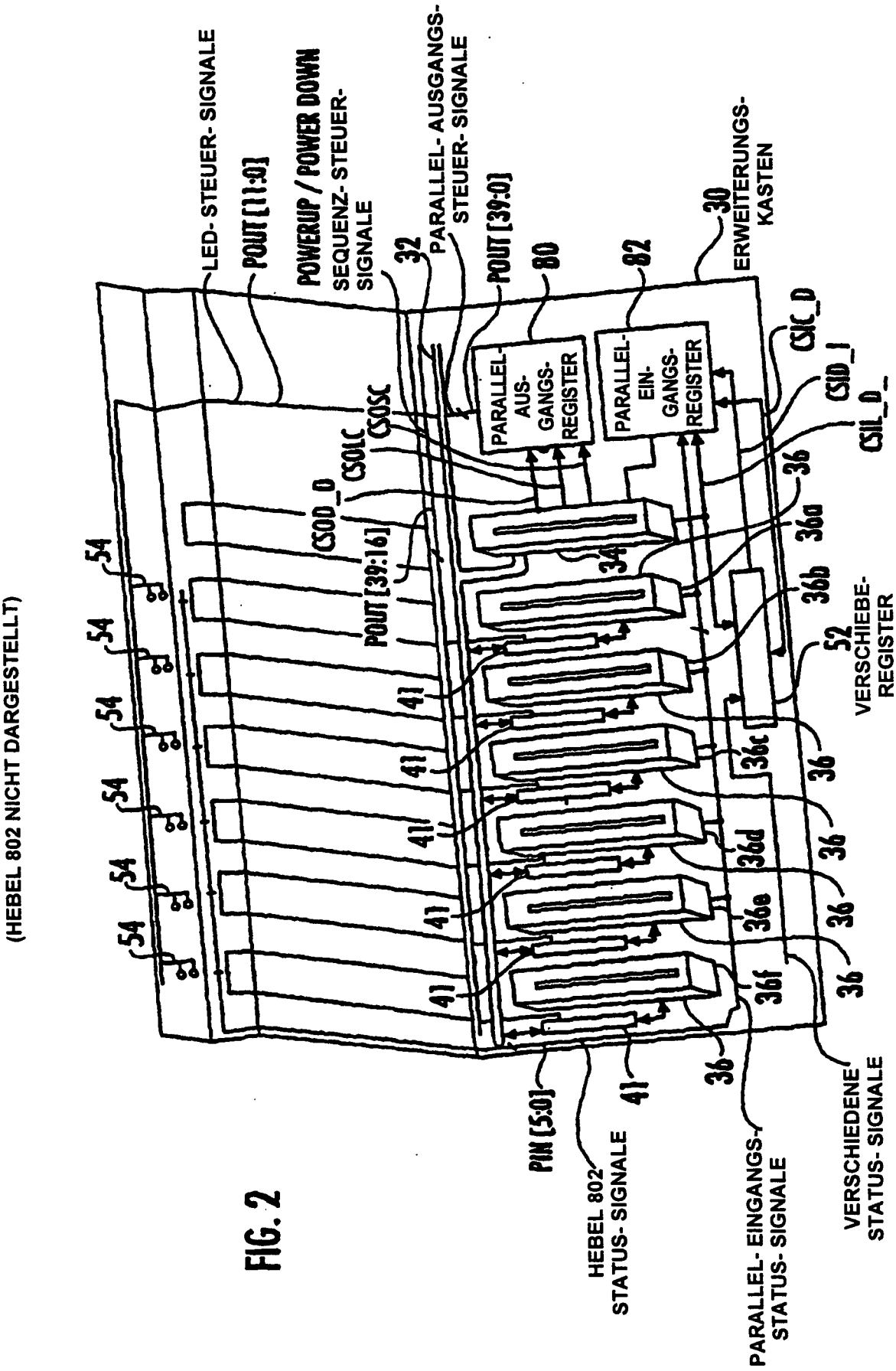

(HEBEL 802 NICHT DARGESTELLT)
LED- STEUER- SIGNALE
POUT [1:0]
POWERUP / POWER DOWN
SEQUENZ- STEUER- SIGNALE
32
PARALLEL- AUSGANGS- STEUER- SIGNALE
POUT [39:0]
80
82
30
ERWEITERUNGS- KASTEN
PARALLEL- AUS- GANGS- REGISTER
PARALLEL- EIN- GANGS- REGISTER
CSOD_0
CSOLC
CSOSC
CSIC_0
CSID_1
CSIL_0_
POUT [39:16]
54
41
34
36
36a
36b
36c
36d
36e
36f
52
VERSCHIEBE- REGISTER
PIN [5:0]
HEBEL 802 STATUS- SIGNALE
PARALLEL- EINGANGS- STATUS- SIGNALE
VERSCHIEDENE STATUS- SIGNALE

FIG. 2

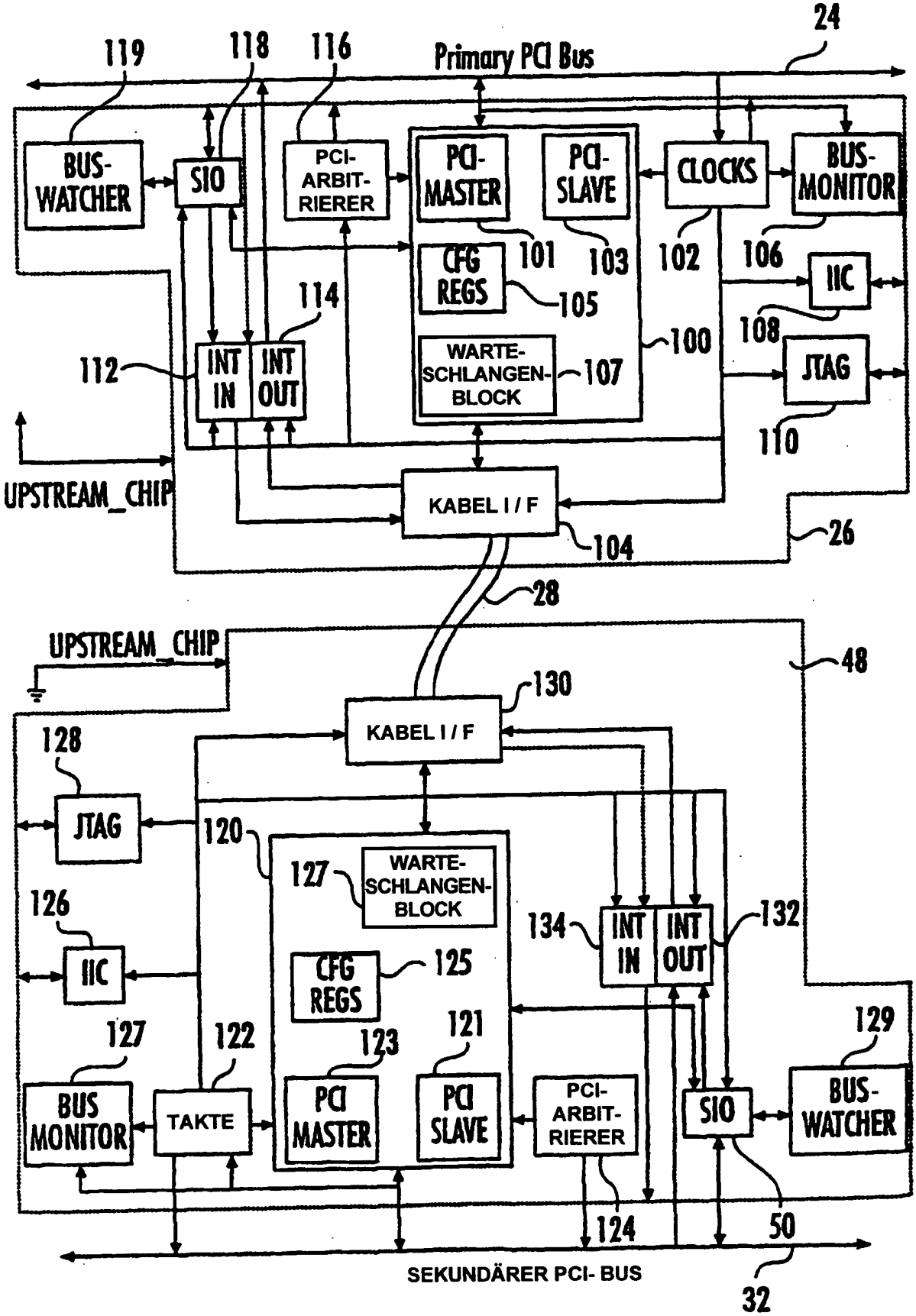

24
119
118
116
Primary PCI Bus
BUS-WATCHER
SIO
PCI-ARBIT-RIERER
PCI-MASTER
PCI-SLAVE
CLOCKS
BUS-MONITOR
CFG REGS
101
103
102
106
IIC
105
114
108
JTAG
112
INT IN
INT OUT
WARTE-SCHLANGEN-BLOCK
107
100
110
UPSTREAM_CHIP
KABEL I / F
104
26
28
UPSTREAM_CHIP
48
130
KABEL I / F
128
JTAG
120
127
WARTE-SCHLANGEN-BLOCK
126
134
INT IN
INT OUT
132
IIC
CFG REGS
125
127
122
123
121
129
BUS MONITOR
TAKTE
PCI MASTER
PCI SLAVE
PCI-ARBIT-RIERER
SIO
BUS-WATCHER
124
50
SEKUNDÄRER PCI- BUS
32

FIG. 3

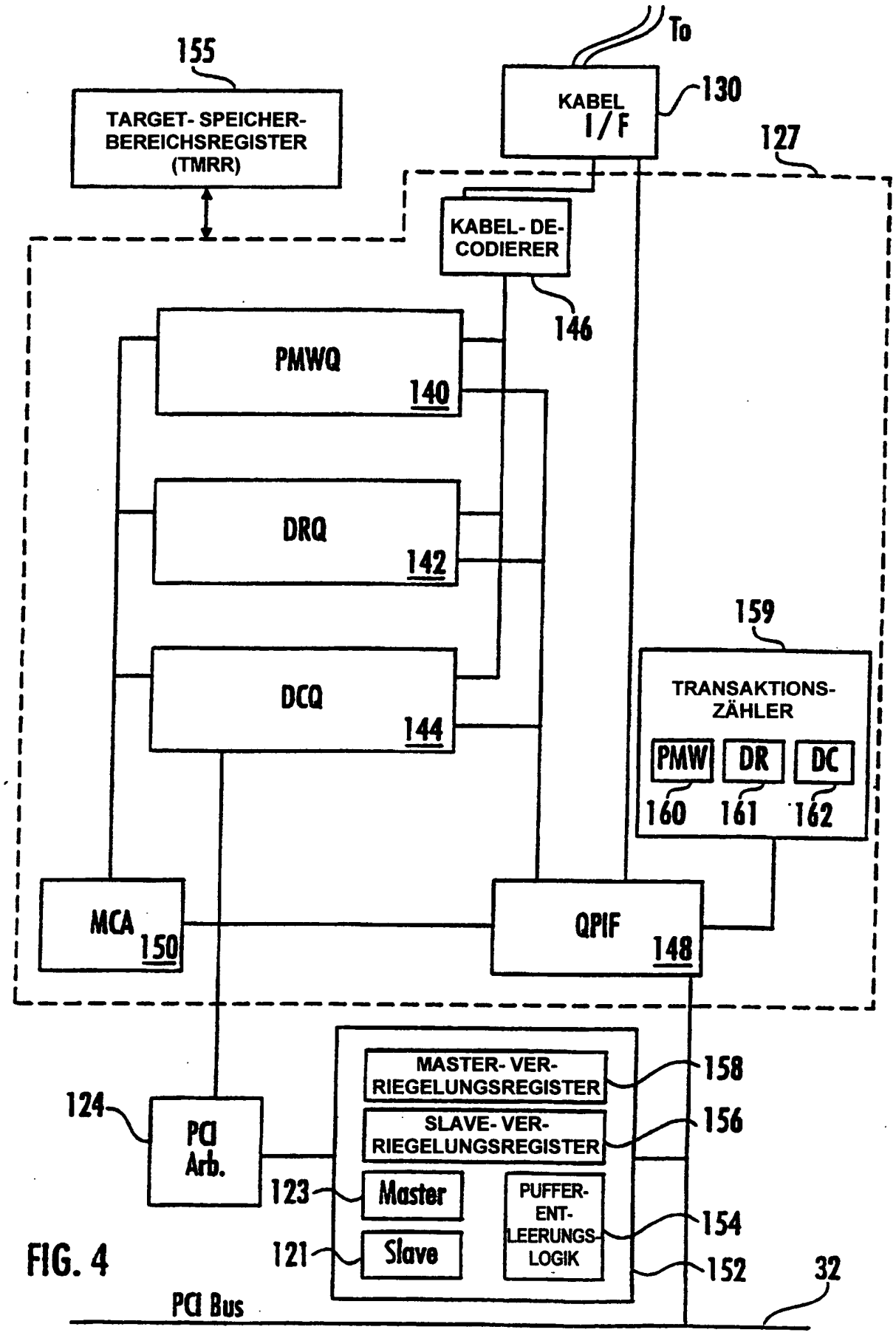

To
155
TARGET- SPEICHER-BEREICHSREGISTER (TMRR)
KABEL I/F
130
127
KABEL- DE-CODIERER
146
PMWQ
140
DRQ
142
DCQ
144
159
TRANSAKTIONS-ZÄHLER
PMW
DR
DC
160
161
162
MCA
150
QPIF
148
MASTER- VER-RIEGELUNGSREGISTER
158
SLAVE- VER-RIEGELUNGSREGISTER
156
124
PCI Arb.
123
Master
121
Slave
PUFFER-ENT-LEERUNGS-LOGIK
154
152
32
PCI Bus

FIG. 4

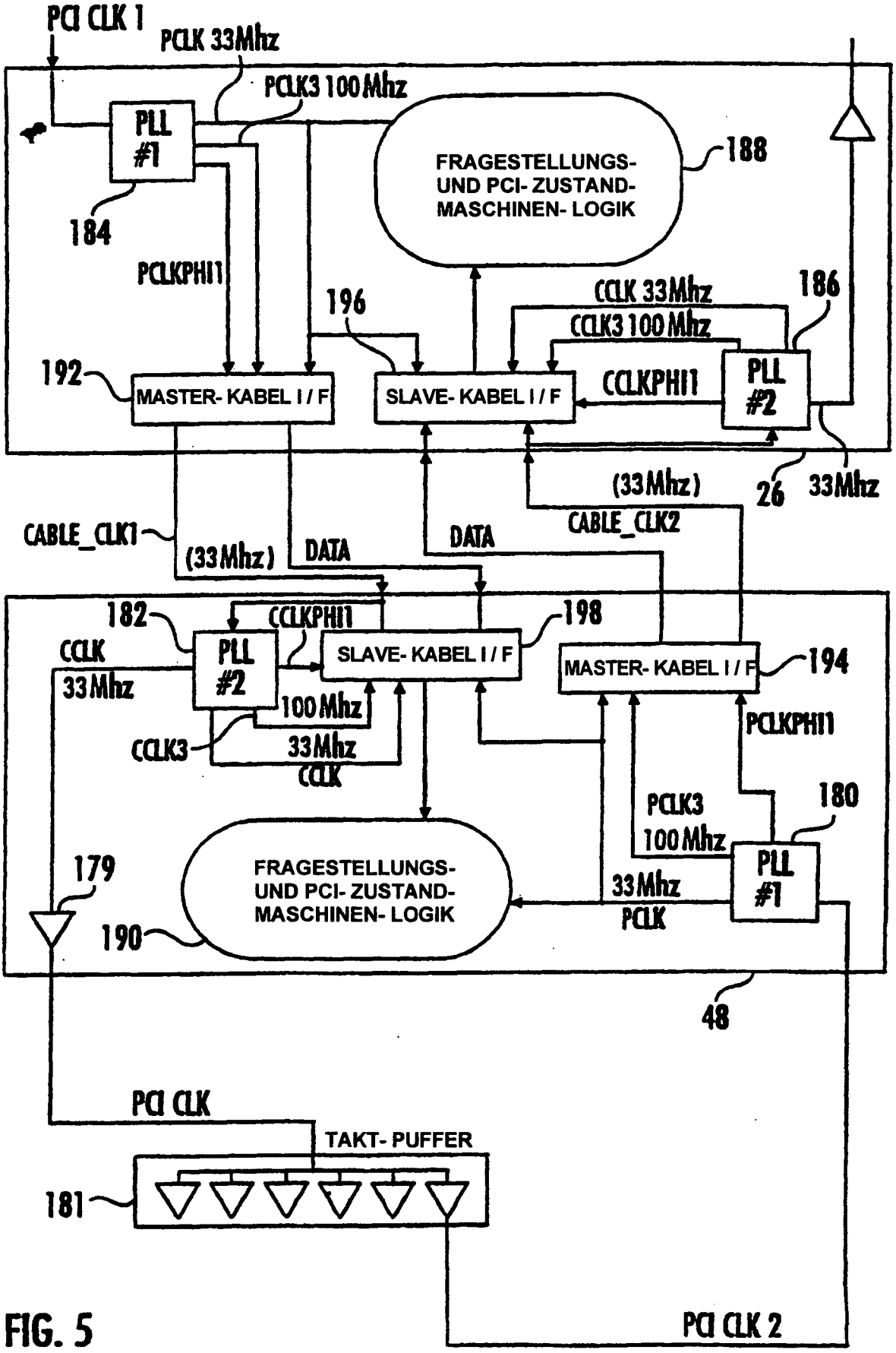

PCI CLK 1
PCLK 33Mhz
PCLK3 100Mhz
PLL #1
184
FRAGESTELLUNGS- UND PCI- ZUSTAND- MASCHINEN- LOGIK
188
PCLKPHI1
196
CCLK 33Mhz
CCLK3 100Mhz
186
192
MASTER- KABEL I / F
SLAVE- KABEL I / F
CCLKPHI1
PLL #2
(33Mhz)
26
33Mhz
CABLE_CLK1
(33Mhz)
DATA
DATA
CABLE_CLK2
182
CCLKPHI1
198
CCLK
33Mhz
PLL #2
SLAVE- KABEL I / F
MASTER- KABEL I / F
194
100Mhz
CCLK3
33Mhz
CCLK
PCLKPHI1
PCLK3
100Mhz
180
179
FRAGESTELLUNGS- UND PCI- ZUSTAND- MASCHINEN- LOGIK
33Mhz
PCLK
PLL #1
190
48
PCI CLK
TAKT- PUFFER
181
PCI CLK 2

FIG. 5

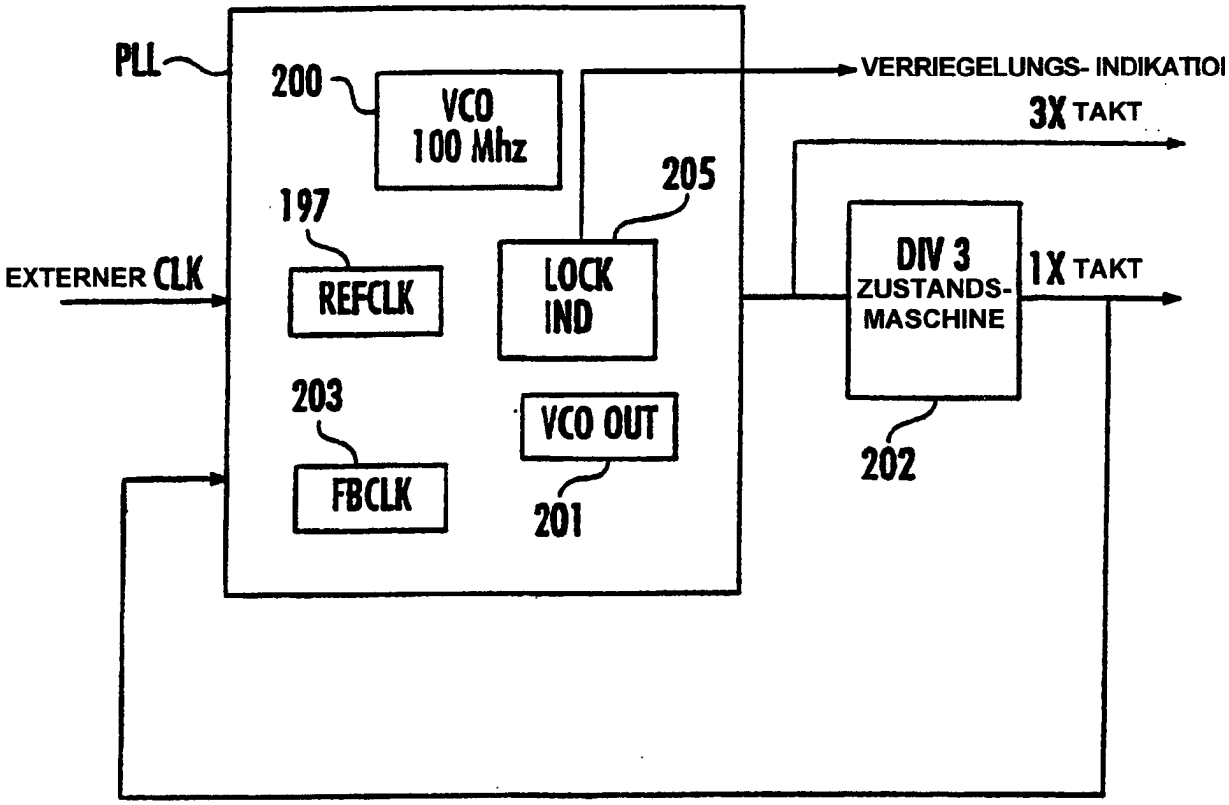

PLL
200
VCO
100 Mhz
VERRIEGELUNGS- INDIKATIOI
3X TAKT
205
197
EXTERNER CLK
REFCLK
LOCK
IND
DIV 3
ZUSTANDS-
MASCHINE
1X TAKT
203
VCO OUT
FBCLK
202
201

FIG. 6

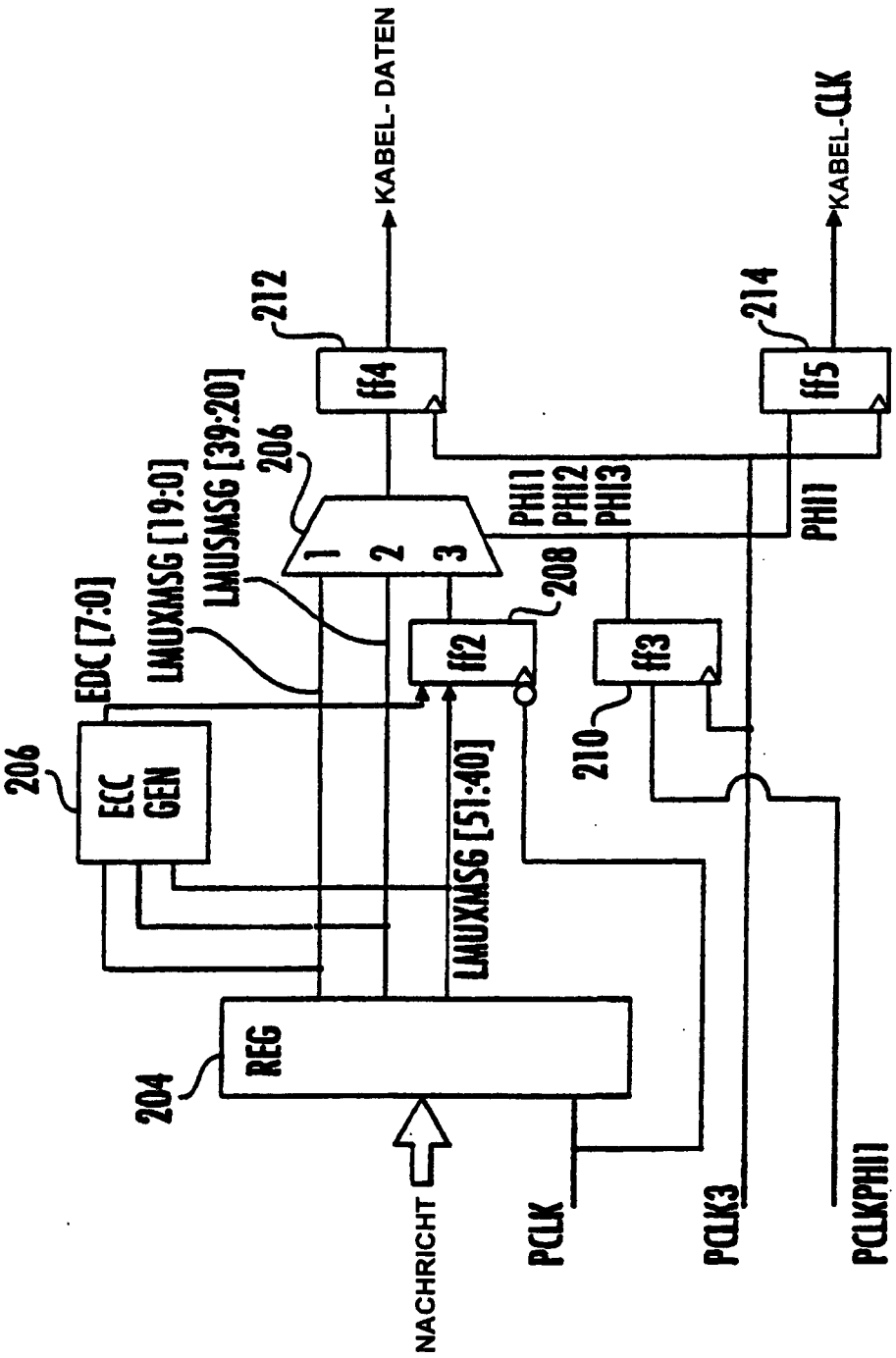

KABEL- DATEN
KABEL-CLK
212
214
ff4
ff5
206
PHI1
PHI2
PHI3
PHI1
1
2
3
LMUSMSG [39:20]
LMUXMSG [19:0]
208
EDC [7:0]
ff2
ff3
206
ECC GEN
210
LMUXMSG [51:40]
204
REG
NACHRICHT
PCLK
PCLK3
PCLKPHI1

FIG. 7

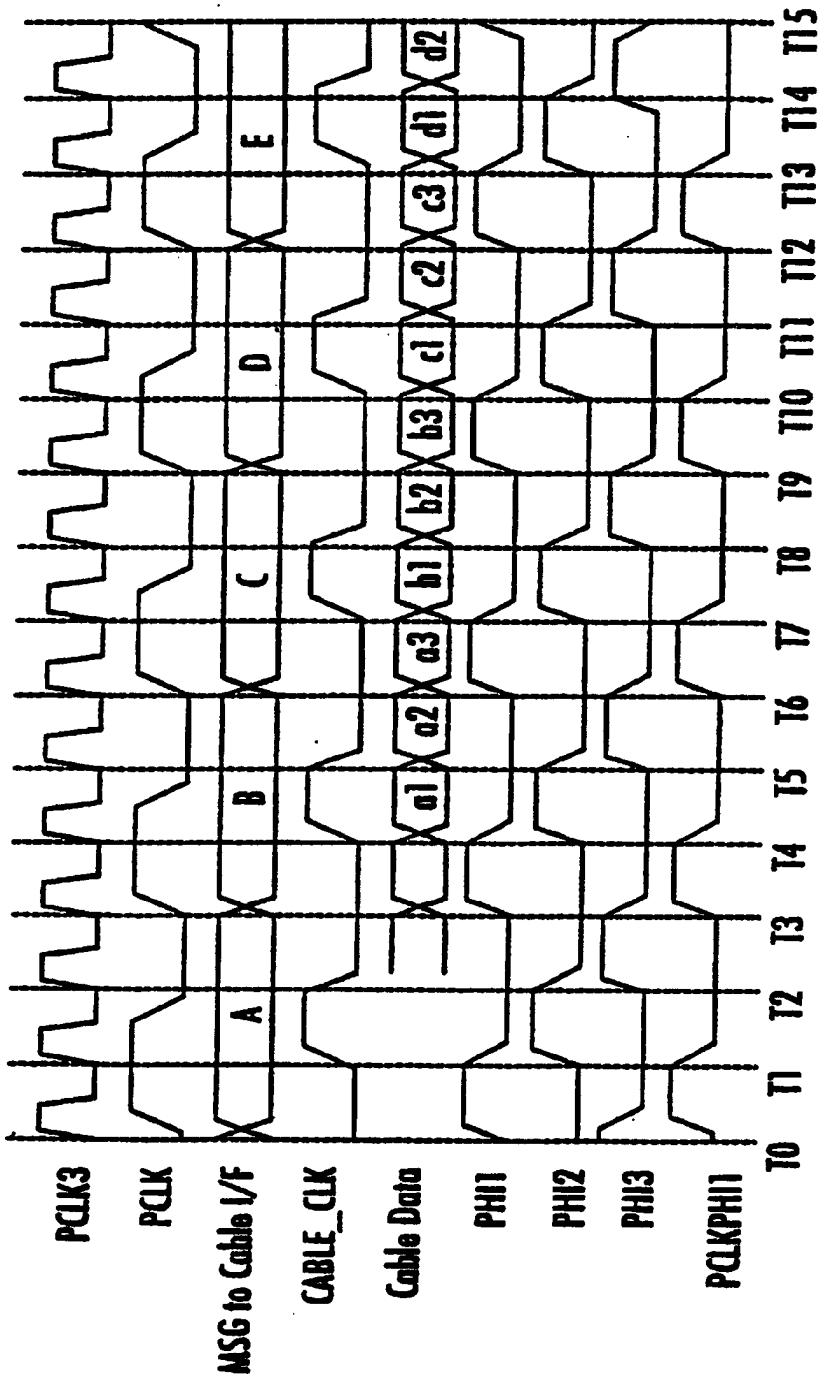

PCLK3
PCLK
MSG to Cable I/F
CABLE_CLK
Cable Data
PHI1
PHI2
PHI3
PCLKPHI1
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
A
B
C
D
E
a1
a2
a3
b1
b2
b3
c1
c2
c3
d1
d2

FIG. 8

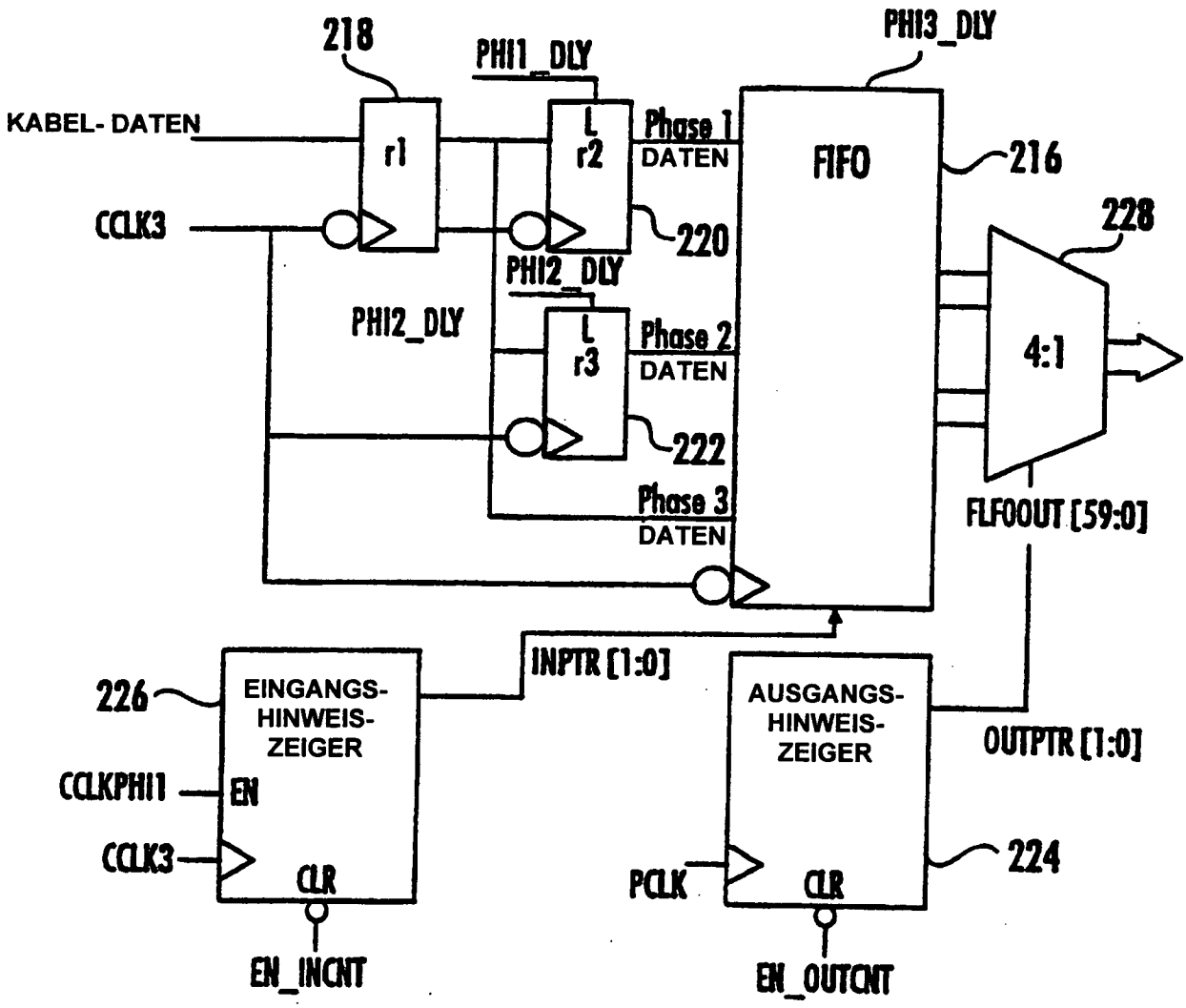

218
PHI1_DLY
PHI3_DLY
KABEL- DATEN
r1
L
r2
Phase 1
DATEN
FIFO
216
CCLK3
220
228
PHI2_DLY
PHI2_DLY
L
r3
Phase 2
DATEN
4:1
222
Phase 3
DATEN
FLF0OUT [59:0]
INPTR [1:0]
226
EINGANGS-
HINWEIS-
ZEIGER
AUSGANGS-
HINWEIS-
ZEIGER
OUTPTR [1:0]
CCLKPHI1
EN
CCLK3
CLR
PCLK
CLR
224
EN_INCNT
EN_OUTCNT

FIG. 9

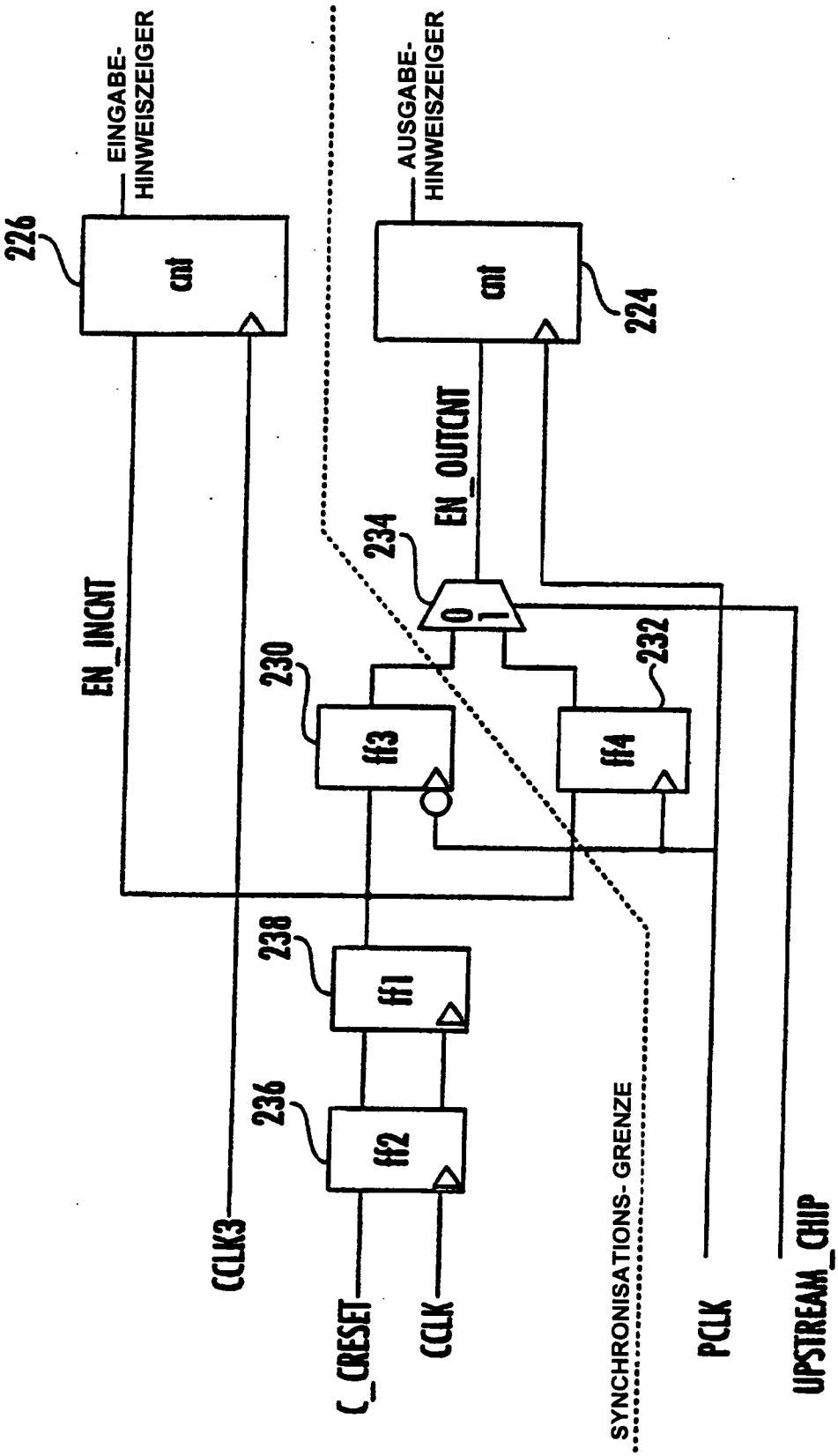

EINGABE-
HINWEISZEIGER
AUSGABE-
HINWEISZEIGER
226
cnt
cnt
224
EN_OUTCNT
234
0
1
EN_INCNT
230
232
ff3
ff4
238
ff1
236
ff2
CCLK3
C_CRESET
CCLK
SYNCHRONISATIONS- GRENZE
PCLK
UPSTREAM_CHIP

FIG. 10

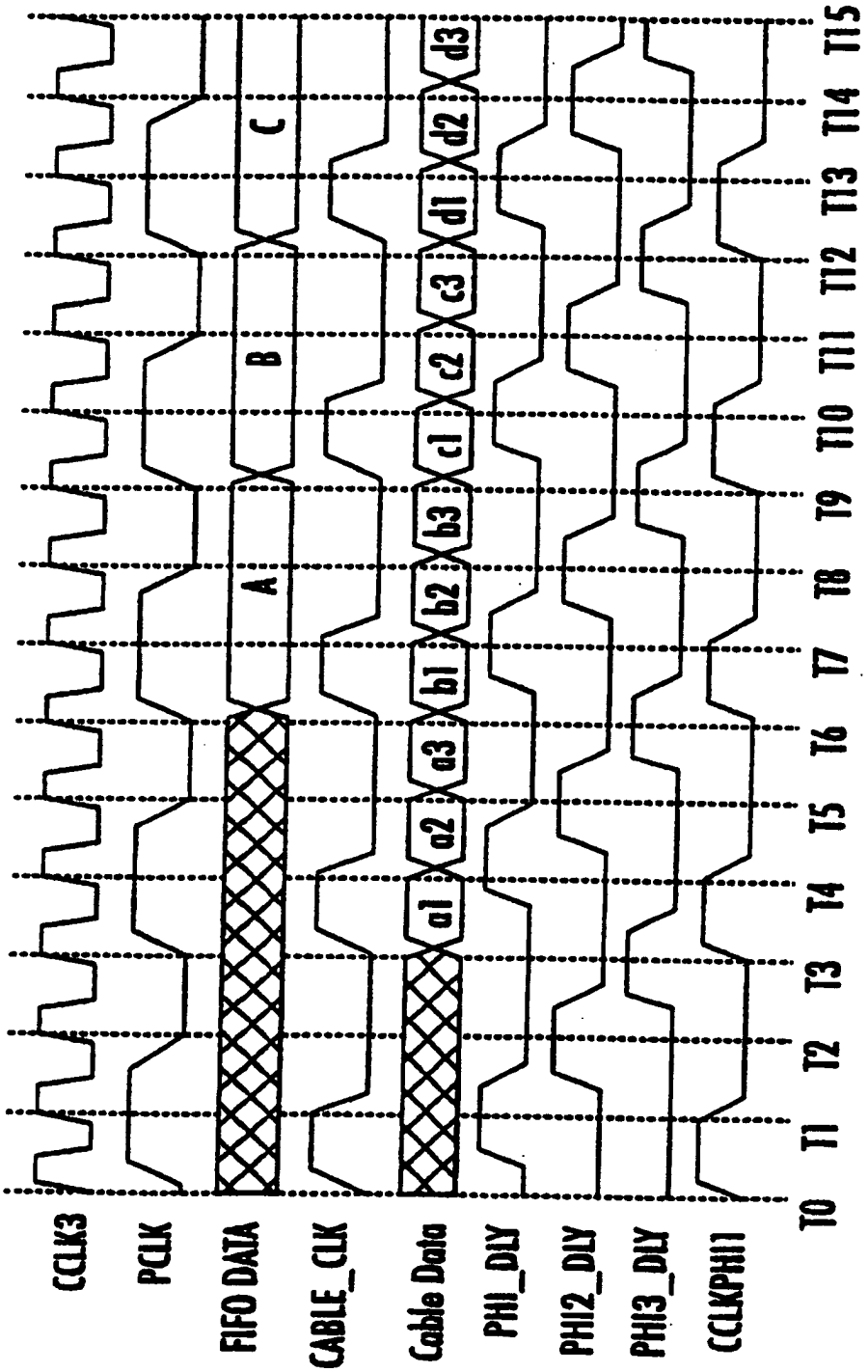

CCLK3
PCLK
FIFO DATA
CABLE_CLK
Cable Data
PHI_DLY
PHI2_DLY
PHI3_DLY
CCLKPHI1
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15
A
B
C
a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3

FIG. 11

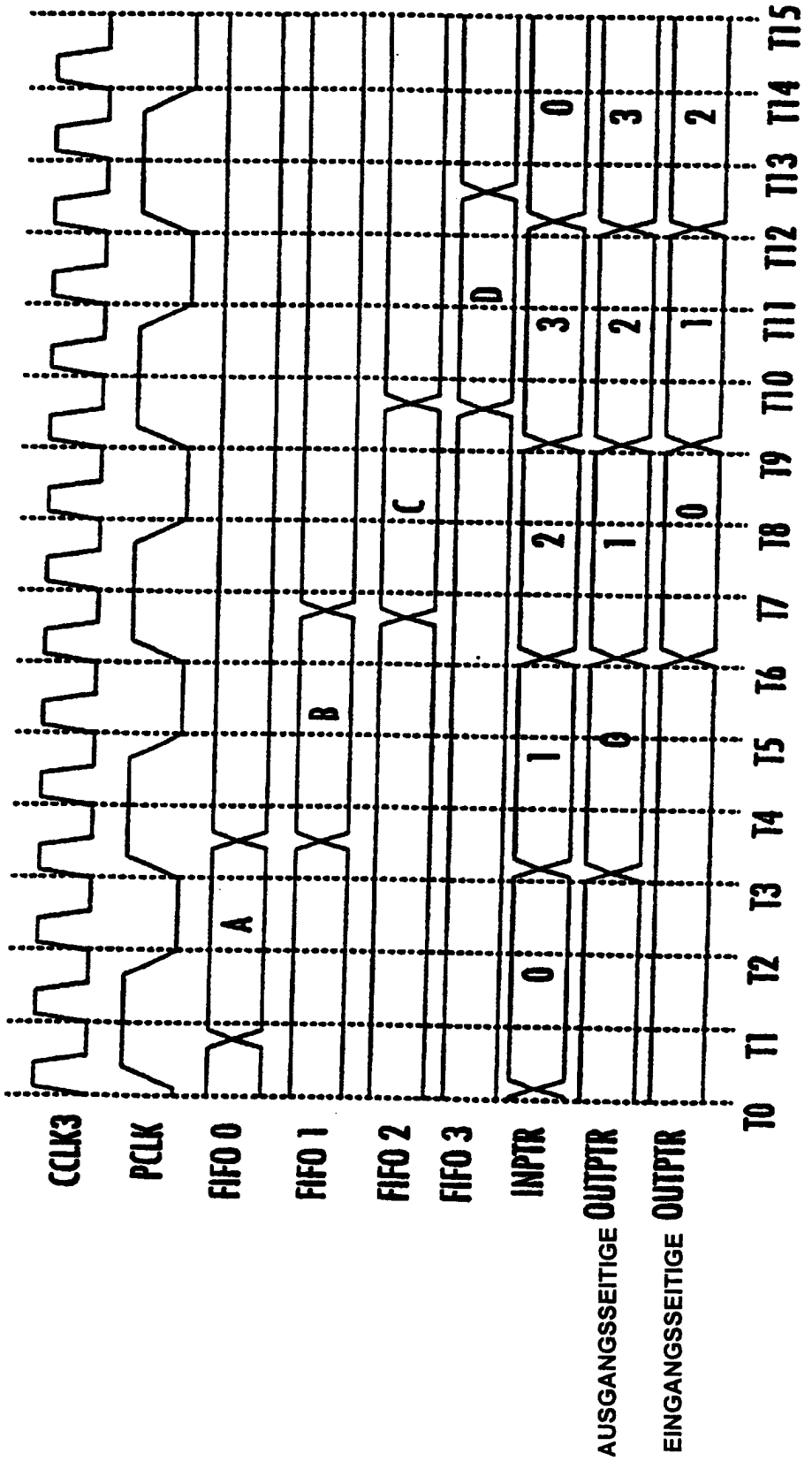

CCLK3
PCLK
FIFO 0
FIFO 1
FIFO 2
FIFO 3
INPTR
AUSGANGSSEITIGE OUTPTR
EINGANGSSEITIGE OUTPTR
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
A
B
C
D
0
1
2
3
0
1
2
3
0
1
2
3

FIG. 12

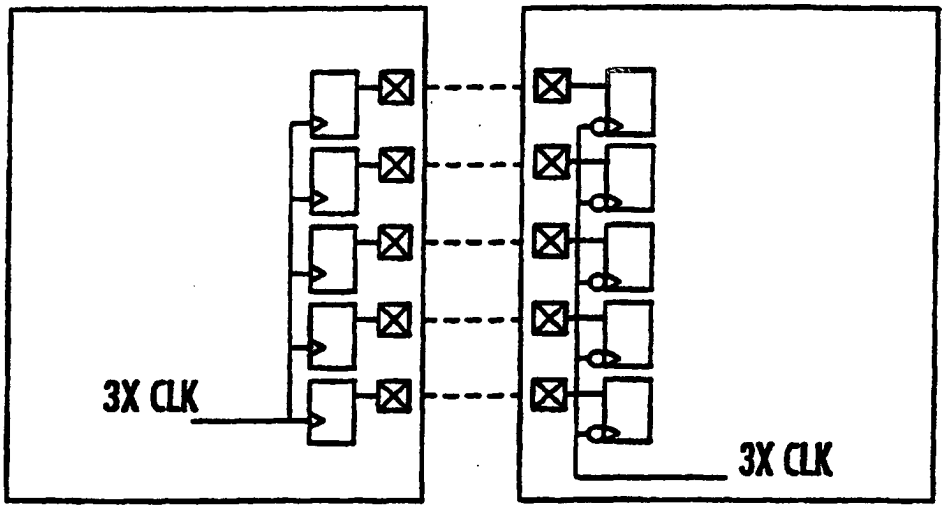


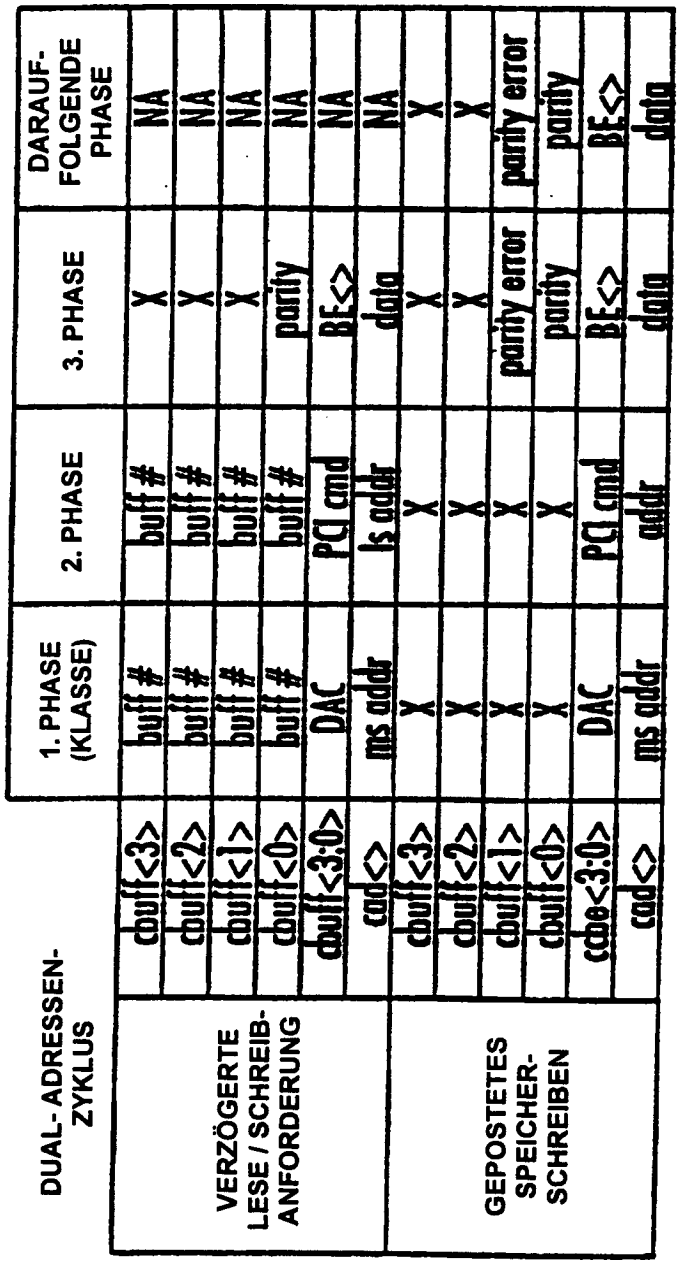
FIG. 13

	EINGANGSSEITIG- ZU- AUSGANGSSEITIG			AUSGANGSSEITIG- ZU- EINGANGSSEITIG		
Bit	Phase 1	Phase 2	Phase 3	Phase 1	Phase 2	Phase 3
20	CAD00	CAD20	CFRAME	CAD00	CAD20	CFRAME
19	CCBE3	CAD19	CAD31	CCBE3	CAD19	CAD31
18	CCBE2	CAD18	CAD30	CCBE2	CAD18	CAD30
17	CCBE1	CAD17	CAD29	CCBE1	CAD17	CAD29
16	CCBE0	CAD16	CAD28	CCBE0	CAD16	CAD28
15	CBUFF3	CAD15	CAD27	CBUFF3	CAD15	CAD27
14	CBUFF2	CAD14	CAD26	CBUFF2	CAD14	CAD26
13	CBUFF1	CAD13	CAD25	CBUFF1	CAD13	CAD25
12	CBUFF0	CAD12	CAD24	CBUFF0	CAD12	CAD24
11	PMW Ack	CAD11	CAD23	PMW Ack	CAD11	CAD23
10	Completion Removed	CAD10	CAD22	Completion Removed	CAD10	CAD22
9	LOCK	CAD09	CAD21	SERR	CAD09	CAD21
8	New Req	CAD08	EDC7	New Req	CAD08	EDC7
7	SPARE	CAD07	EDC6	SPARE	CAD07	EDC6
6	SPARE	CAD06	EDC5	SPARE	CAD06	EDC5
5	SPARE	CAD05	EDC4	SPARE	CAD05	EDC4
4	SPARE	CAD04	EDC3	SPARE	CAD04	EDC3
3	SPARE	CAD03	EDC2	SPARE	CAD03	EDC2
2	SPARE	CAD02	EDC1	SPARE	CAD02	EDC1
1	SPARE	CAD01	EDC0	SPARE	CAD01	EDC0

FIG. 14

EINZEL- ADRESSEN-ZYKLUS		1. PHASE	2. PHASE	DARAUFFOLGENDE PHASEN
VERZÖGERTE-LESE / SCHREIB-ANFORDERUNG	cbuff <3>	buff#	X	NA
	cbuff <2>	buff#	X	NA
	cbuff<1>	buff#	X	NA
	cbuff<0>	buff#	parity[3]	NA
	ccbe<3:0>	PCI cmd	BE<>[1]	NA
	cad<>	addr	data<>[3]	NA
GEPOSTETES-SPEICHER-SCHREIBEN	cbuff <3>	X	X	X
	cbuff <2>	X	data ready	data ready
	cbuff<1>	X	parity error	parity error
	cbuff<0>	X	parity	parity
	ccbe<3:0>	PCI cmd	BE<>	BE<>
	cad<>	addr	data	data
VERZÖGERTE-LESE / SCHREIB-ABSCHLUSS	cbuff <3>	buff#	end of completion	end of completion
	cbuff <2>	buff#	data ready	data ready
	cbuff<1>	buff#	parity error	parity error
	cbuff<0>	buff#	parity	parity
	ccbe<3:0>	DRC	status	status
	cad<>	X	data	data
Stream- Connect	cbuff <3>	buff#	X	X
	cbuff <2>	buff#	data ready	data ready
	cbuff<1>	buff#	X	X
	cbuff<0>	buff#	X	X
	ccbe<3:0>	strm conn	X	X
	cad<>	X	X	X

FIG. 15A



DUAL-ADRESSEN-ZYKLUS		1. PHASE (KLASSE)	2. PHASE	3. PHASE	DARAUF-FOLGENDE PHASE
VERZÖGERTE LESE / SCHREIB-ANFORDERUNG	cbuff<3>	buff #	buff #	X	NA
	cbuff<2>	buff #	buff #	X	NA
	cbuff<1>	buff #	buff #	X	NA
	cbuff<0>	buff #	buff #	parity	NA
	cbuff<3:0>	DAC	PCI cmd	BE<>	NA
	cad<>	ms addr	ls addr	data	NA
GEPOSTETES SPEICHER-SCHREIBEN	cbuff<3>	X	X	X	X
	cbuff<2>	X	X	X	X
	cbuff<1>	X	X	parity error	parity error
	cbuff<0>	X	X	parity	parity
	cbe<3:0>	DAC	PCI cmd	BE<>	BE<>
	cad<>	ms addr	addr	data	data

FIG. 15B

Parameter	WERT
IMPEDANZ (Differential)	108 +/- 5 Ohm
IMPEDANZ (Single-ended)	67 +/- 5 Ohm
PROPAGATIONS- VERZÖGERUNG	1.54 ns/ft min, 1.58 ns/ft max
Delay Skew	0.025 ns/ft max
DÄMPFUNG (Differential)	0.08 db/ft max @ 50 MHz
LÄNGE	12"
DC WIDERSTAND	0.070 Ohm /ft max

FIG. 16

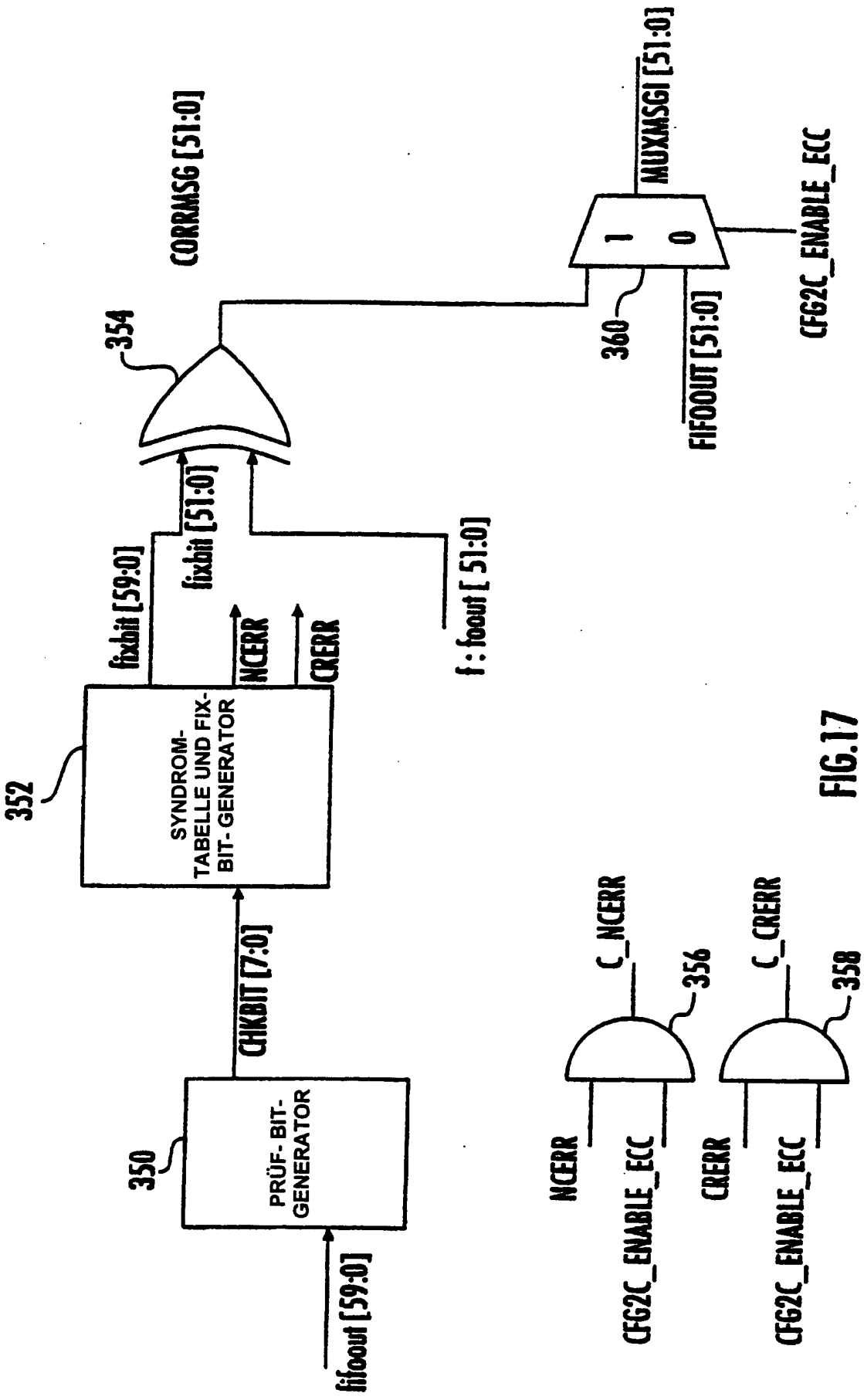

fifoout [59:0]
350
PRÜF- BIT- GENERATOR
CHKBIT [7:0]
352
SYNDROM- TABELLE UND FIX- BIT- GENERATOR
fixbit [59:0]
fixbit [51:0]
NCERR
CRERR
f : fout [51:0]
354
CORRMSG [51:0]
360
1
0
MUXMSGI [51:0]
FIFOOUT [51:0]
CFG2C_ENABLE_ECC
NCERR
CFG2C_ENABLE_ECC
C_NCERR
356
CRERR
CFG2C_ENABLE_ECC
C_CRERR
358

FIG.17

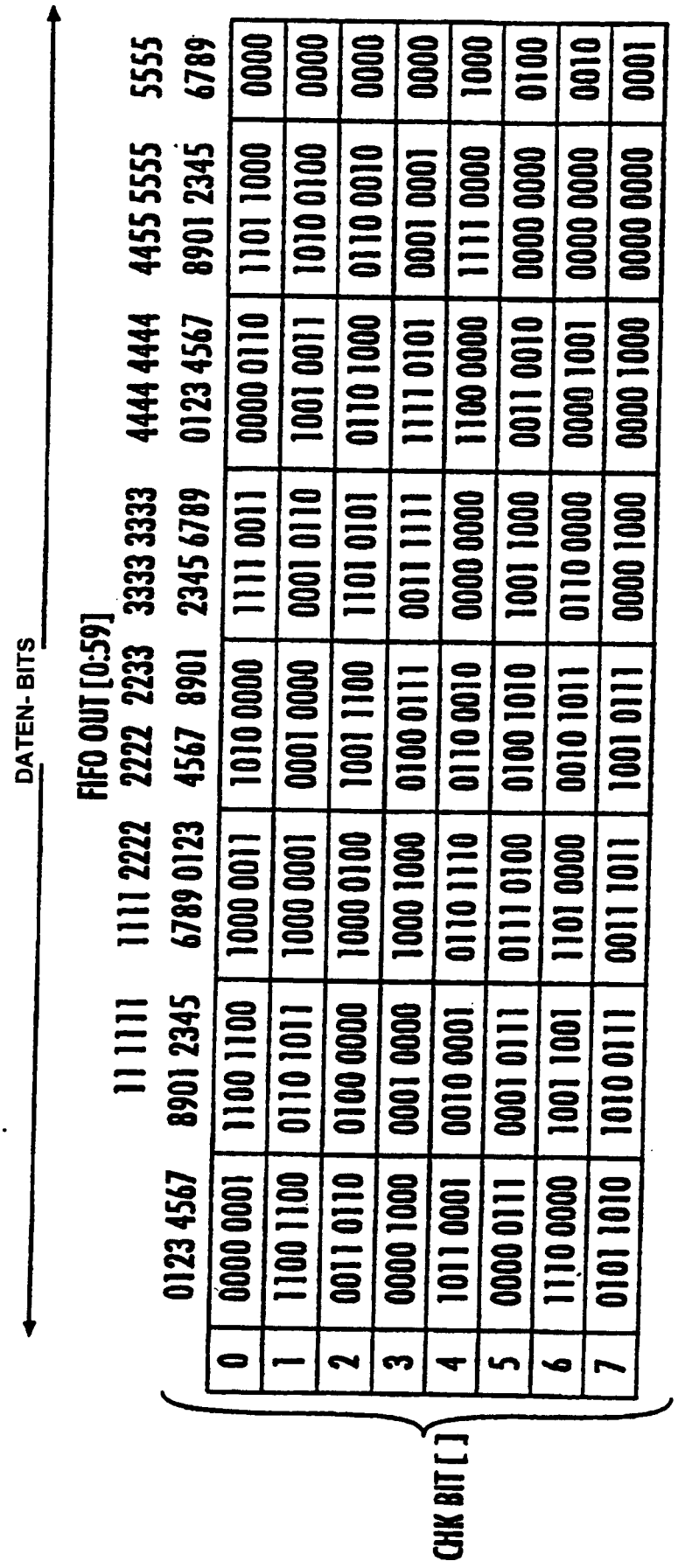
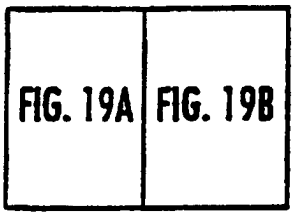

DATEN-BITS
FIFO OUT [0:59]
CHK BIT [ ]
0123 4567 8901 2345 6789 0123 4567 8901 2345 6789 0123 4567 8901 2345 6789 0123 4567 8901 2345 6789
0 0000 0001 1100 1100 1000 0011 1010 0000 1111 0011 0000 0110 1101 1000 0000
1 1100 1100 0110 1011 1000 0001 0001 0000 0001 0110 1001 0011 1010 0100 0000
2 0011 0110 0100 0000 1000 0100 1001 1100 1101 0101 0110 1000 0110 0010 0000
3 0000 1000 0001 0000 1000 1000 0100 0111 0011 1111 1111 0101 0001 0001 0000
4 1011 0001 0010 0001 0110 1110 0110 0010 0000 0000 1100 0000 1111 0000 1000
5 0000 0111 0001 0111 0111 0100 0100 1010 1001 1000 0011 0010 0000 0000 0100
6 1110 0000 1001 1001 1101 0000 0010 1011 0110 0000 0000 1001 0000 0000 0010
7 0101 1010 1010 0111 0011 1011 1001 0111 0000 1000 0000 1000 0000 0000 0001

FIG. 18

FIG. 19



00 No Error	20 DB57	40 DB58
01 DB 52	21 UNCER	41 DB44,24
02 DB53	22 UNCER	42 DB52,12
03 UNCER	23 DB 46	43 DB12
04 DB54	24 DB 52,32	44 UNCER
05 UNCER	25 DB32	45 DB33
06 UNCER	26 DB05	46 UNCER
07 DB09	27 DB55,35	47 DB53,33
08 DB55	28 DB41,21	48 DB40,00
09 UNCER	29 UNCER	49 DB34
0A UNCER	2A DB43	4A DB47
0B DB38	2B UNCER	4B DB58,38
0C UNCER	2C DB42	4C UNCER
0D DB39	2D UNCER	4D DB54,34
0E DB37	2E DB57,37	4E DB44,04
0F DB24,04	2F DB35	4F DB16
10 DB56	30 UNCER	50 DB57,17
11 UNCER	31 DB07	51 DB26
12 DB49,09	32 UNCER	52 DB00
13 DB48	33 UNCER	53 UNCER
14 UNCER	34 DB21	54 DB02
15 DB49	35 UNCER	55 UNCER
16 DB50	36 UNCER	56 UNCER
17 DB23,03	37 UNCER	57 UNCER
18 UNCER	38 DB25	58 UNCER
19 DB51	39UNCER	59 UNCER
1A DB40	3A UNCER	5A UNCER
1B UNCER	3B UNCER	5B UNCER
1C DB41	3C UNCER	5C UNCER
1D UNCER	3D DB43,23,03	5D UNCER
1E DB25,05	3E UNCER	5E DB57,37,17
1F UNCER	3F UNCER	5F DB56,16

FIG. 19A

60 DB59,19	80 DB59	A0 DB 31,11	C0 UNCER	E0 DB19
61 UNCER	81 UNCER	A1 DB13	C1 DB08	E1 UNCER
62 UNCER	82 DB40,20	A2 DB14	C2 DB01	E2 UNCER
63 UNCER	83 DB23	A3 DB53,13	C3 UNCER	E3 UNCER
64 DB28	84 DB50,10	A4 DB06	C4 DB44	E4 DB33,13
65 UNCER	85 DB24	A5 DB28,08	C5 DB22,02	E5 UNCER
66 DB32,12	86 DB27	A6 DB54,14	C6 UNCER	E6 DB53,33,13
67 DB52,32,12	87 DB46,06	A7 UNCER	C7 UNCER	E7 DB36,16
68 DB11	88 UNCER	A8 DB36	C8 DB31	E8 UNCER
69 DB30,10	89 DB45	A9 DB43,23	C9 UNCER	E9 DB42,22,02
6A DB30,10	8A DB04	AA UNCER	CA DB20,00	EA DB41,21,0
6B UNCER	8B DB29,09	AB UNCER	CB DB44,24,04	EB DB34,14
6C UNCER	8C DB29	AC UNCER	CC DB47,27	EC UNCER
6D DB59,39,19	8D DB59,39	AD UNCER	CD UNCER	ED DB39,19
6E UNCER	8E UNCER	AE UNCER	CE UNCER	EE DB50,30
6F UNCER	8F UNCER	AF DB45,05	CF UNCER	EF DB54,34,14
70 DB17	90 UNCER	B0 DB18	D0 DB40,20,00	F0 DB 58,18
71 DB51,11	91 DB22	B1 DB45,25	D1 DB51,31	F1 UNCER
72 DB46,26	92 DB10	B2 UNCER	D2 48,08	F2 DB15
73 UNCER	93 UNCER	B3 UNCER	D3 UNCER	F3 UNCER
74 UNCER	94 DB03	B4 UNCER	D4 UNCER	F4 UNCER
75 UNCER	95 UNCER	B5 UNCER	D5 DB55,35,15	F5 DB26,06
76 UNCER	96 UNCER	B6 DB48,28,08	D6 DB46,26,06	F6 DB21,01
77 DB48,28	97 DB45,25,05	B7 DB27,07	D7 UNCER	F7 DB56,36,16
78 DB42,02	98 DB20	B8 DB56,36	D8 UNCER	F8 DB30
79 UNCER	99 DB49,29	B9 DB51,31,11	D9 UNCER	F9 UNCER
7A UNCER	9A UNCER	BA UNCER	DA UNCER	FA DB55,15
7B DB47,07	9B UNCER	BB DB38,18	DB UNCER	FB DB58,38,18
7C DB50,30,10	9C UNCER	BC UNCER	DC UNCER	FC UNCER
7D UNCER	9D UNCER	BD DB42,22	DD DB35,15	FD DB47,27,0
7E DB37,17	9E DB49,29,09	BE DB43,03	DE DB41,01	FE UNCER
7F UNCER	9F UNCER	BF UNCER	DF UNCER	FF UNCER

FIG. 19B

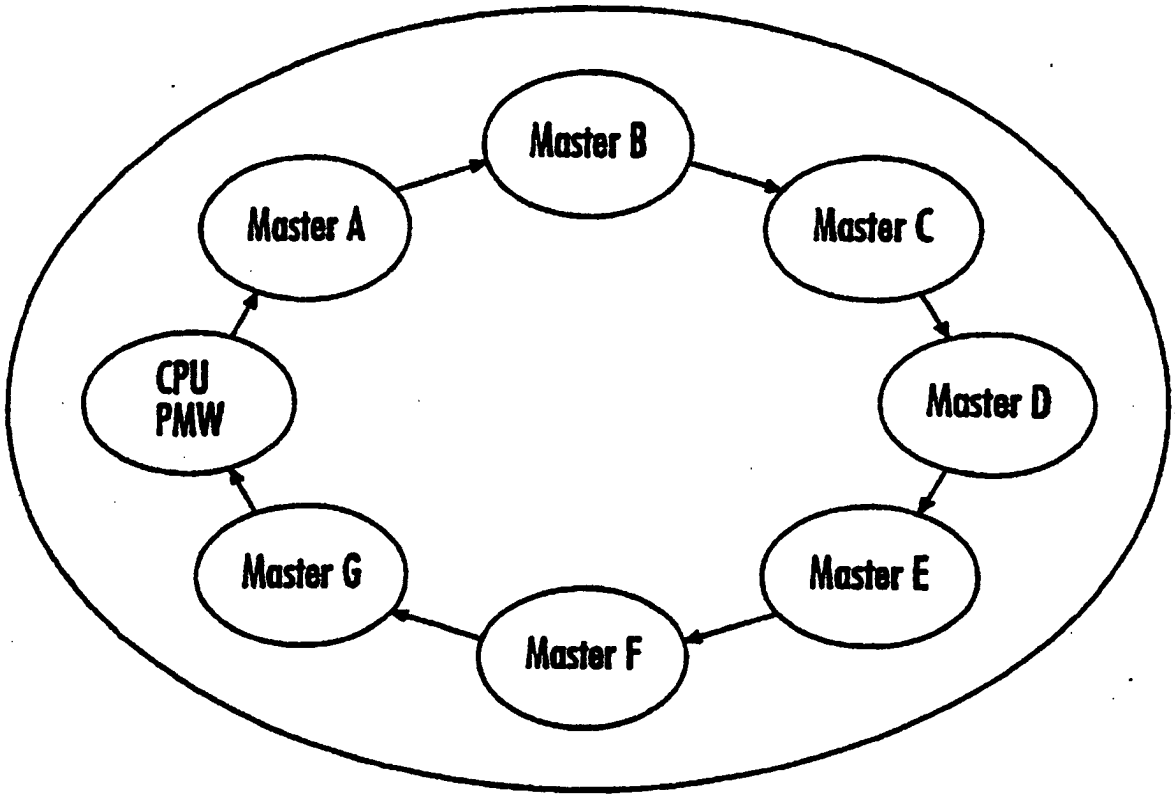

Master B
Master A
Master C
CPU
PMW
Master D
Master G
Master E
Master F

FIG. 20A

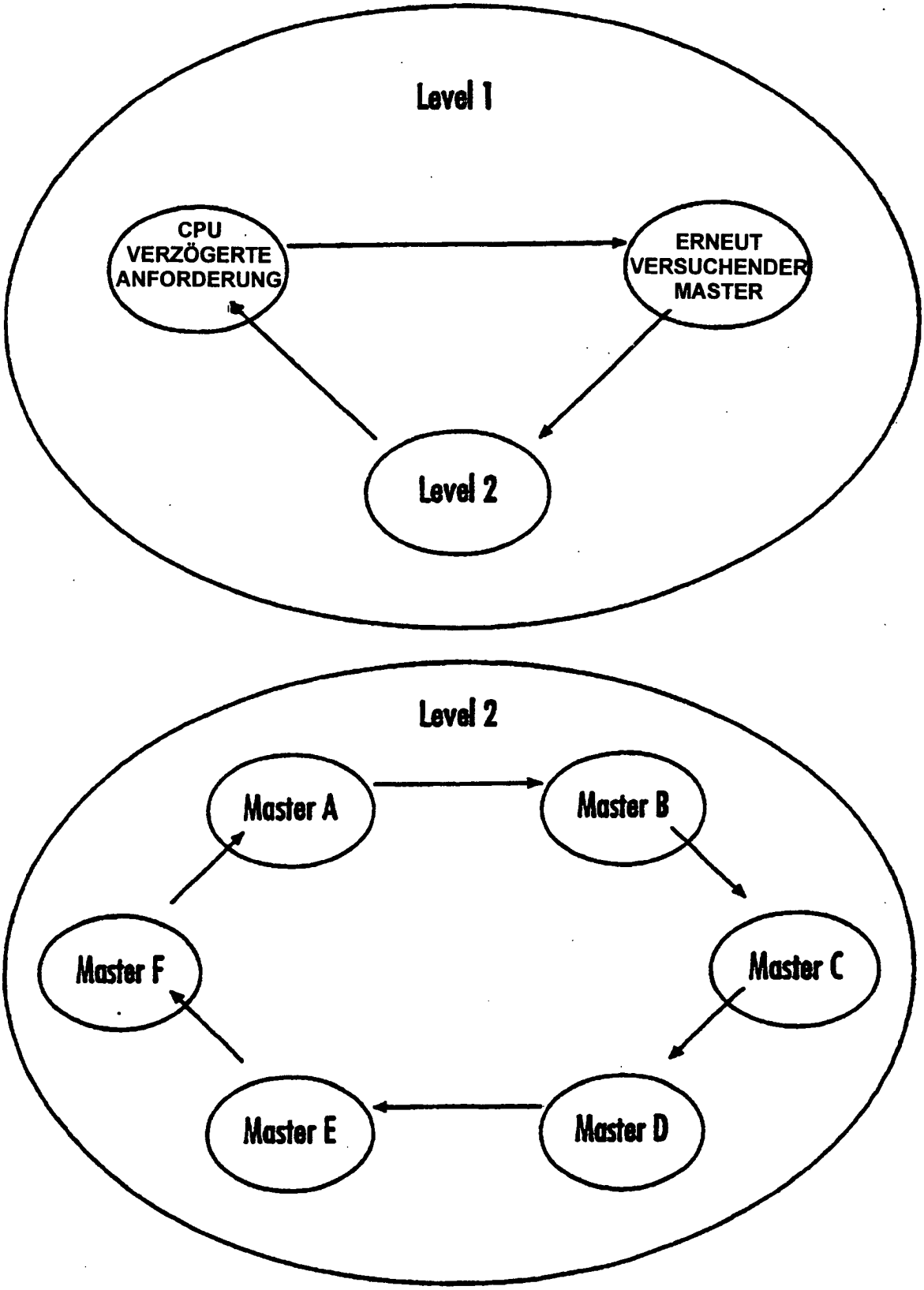

Level 1
CPU VERZÖGERTE ANFORDERUNG
ERNEUT VERSUCHENDER MASTER
Level 2
Level 2
Master A
Master B
Master C
Master D
Master E
Master F

FIG 20B

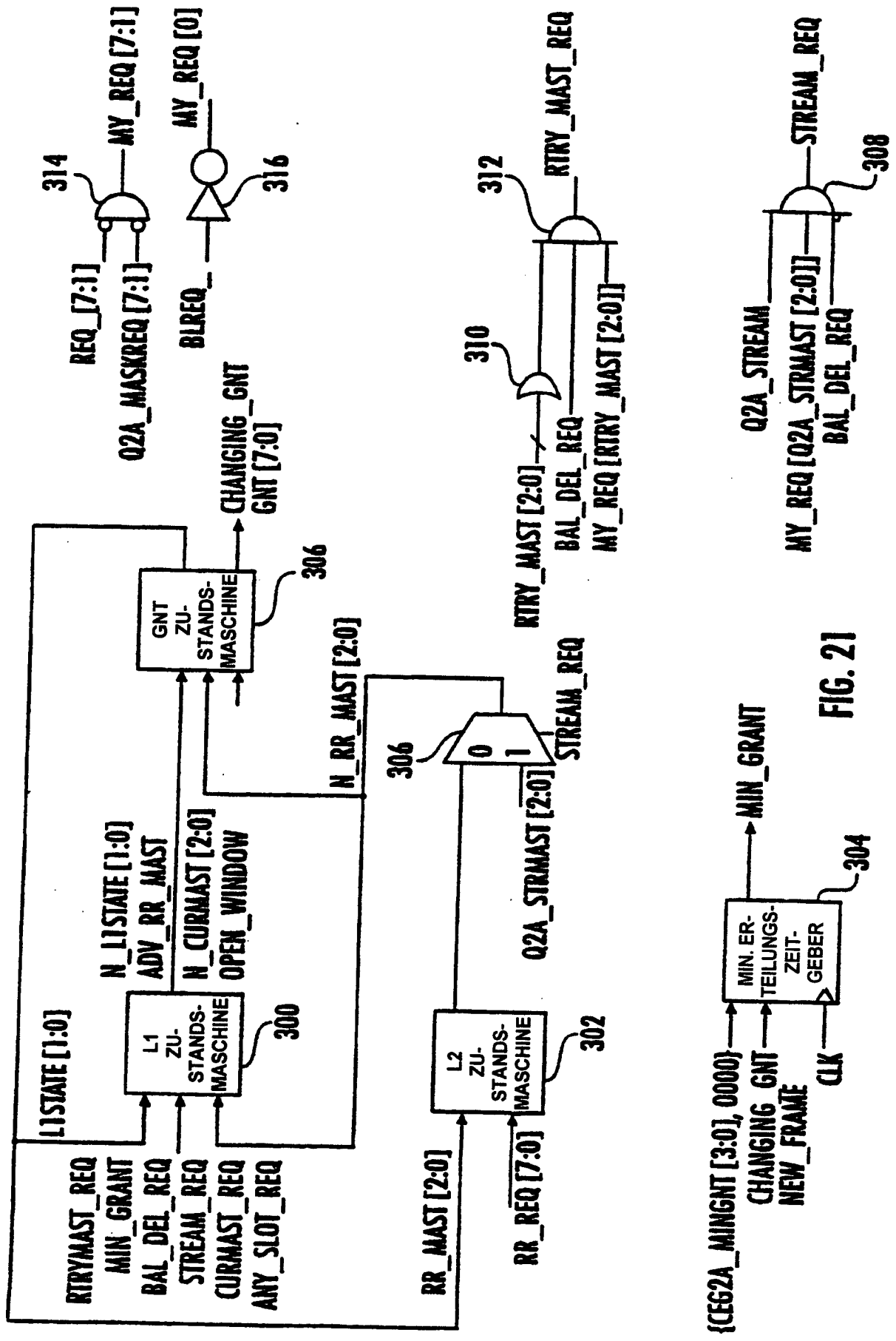

L1STATE [1:0]
RTRYMAST_REQ
MIN_GRANT
BAL_DEL_REQ
STREAM_REQ
CURMAST_REQ
ANY_SLOT_REQ
L1 ZU-STANDS-MASCHINE
300
N_L1STATE [1:0]
ADV_RR_MAST
N_CURMAST [2:0]
OPEN_WINDOW
GNT ZU-STANDS-MASCHINE
306
CHANGING_GNT
GNT [7:0]
REQ_[7:1]
Q2A_MASKREQ [7:1]
314
MY_REQ [7:1]
BLREQ_
316
MY_REQ [0]
N_RR_MAST [2:0]
306
0
1
STREAM_REQ
RR_MAST [2:0]
L2 ZU-STANDS-MASCHINE
302
RR_REQ [7:0]
Q2A_STRMAST [2:0]
RTRY_MAST [2:0]
310
BAL_DEL_REQ
MY_REQ [RTRY_MAST [2:0]]
312
RTRY_MAST_REQ
{CEG2A_MINGNT [3:0], 0000}
CHANGING_GNT
NEW_FRAME
MIN. ER-TEILUNGS-ZEIT-GEBER
CLK
304
MIN_GRANT
Q2A_STREAM
MY_REQ [Q2A_STRMAST [2:0]]
BAL_DEL_REQ
308
STREAM_REQ

FIG. 21

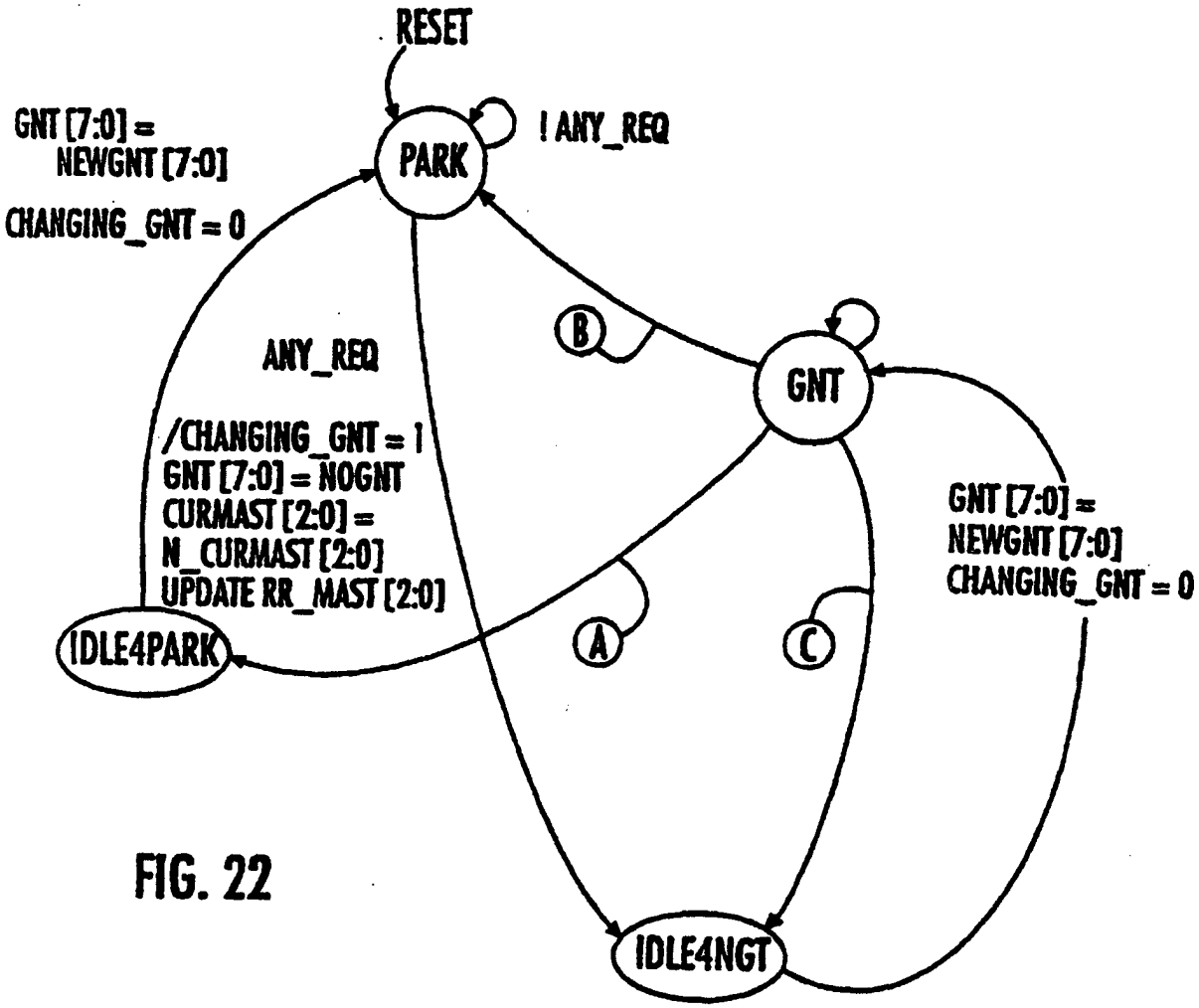
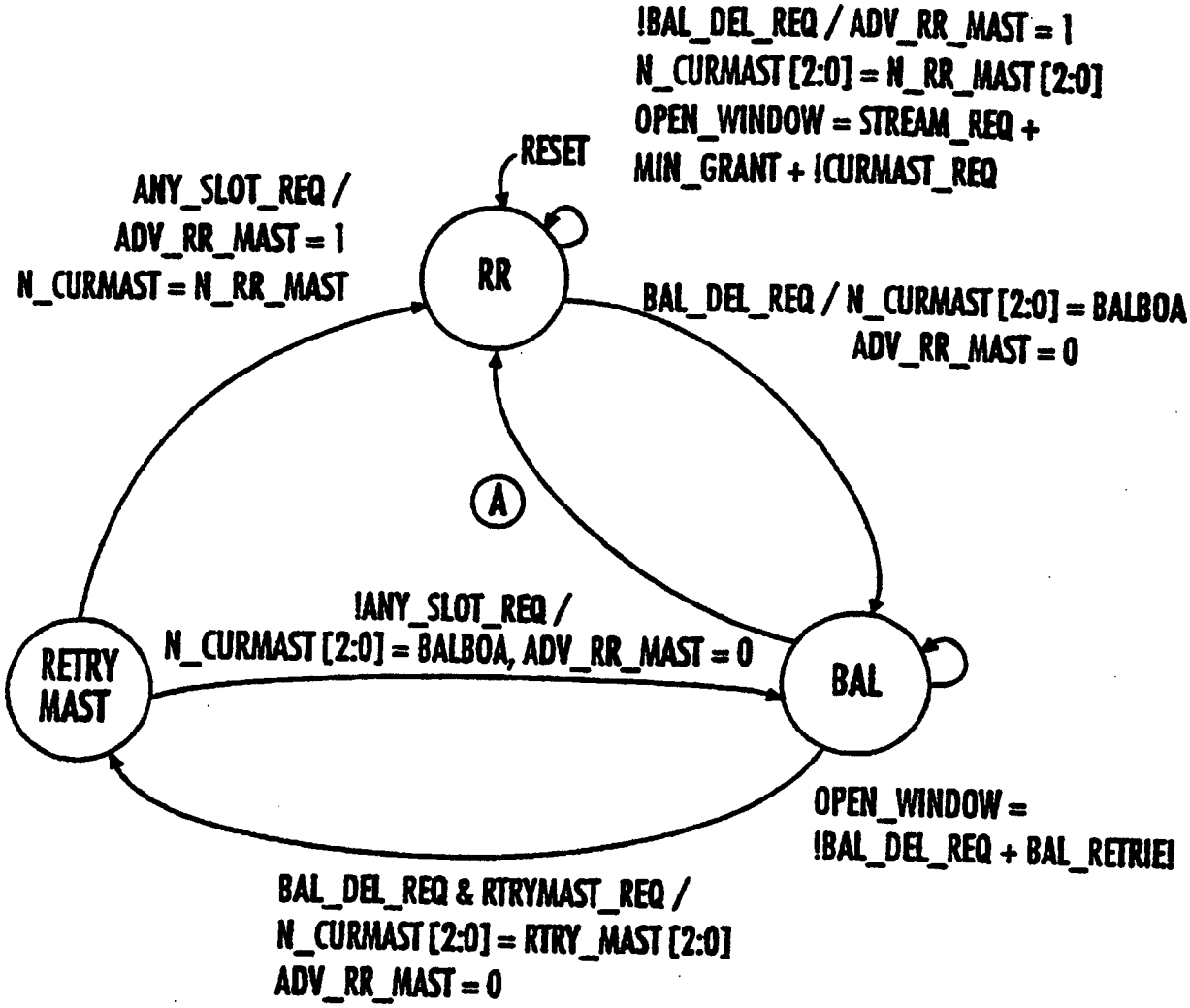


FIG. 22

(A) OPEN_WINDOW & !ANY_REQ & BUS_IDLE & (N_CURMAST! = CURMAST) / CHANGING_GNT = 1
GNT [7:0] = NOGNT
CURMAST [2:0] = N_CURMAST [2:0]
RR_MAST [2:0]
=N_RR_MAST [2:0]
LISTATE [1:0] =
N_LISTATE [1:0]

(B) OPEN_WINDOW & !ANY_REQ & BUS_IDLE & (N_CURMAST = CURMAST)
/LISTATE [1:0] = N_LISTATE [1:0]

(C) OPEN_WINDOW & (N_CURMAST! = CURMAST) / CHANGING_GNT = 1
GNT [7:0] = NOGNT
CURMAST [2:0] = N_CURMAST [2:0]
UPDATE RR_MAST [2:0]
LISTATE [1:0] = N_LISTATE [1:0]



(A) !BAL_DEL_REQ +
BAL_DEL_REQ & !RTRYMAST_REQ
ANY_SLOT_REQ / ADV_RR_MAST = 1
N_CURMAST [2:0] = N_RR_MAST [2:0]

FIG. 23

CURMAST [2:0]	NEWGNT [7:0]
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

FIG. 24

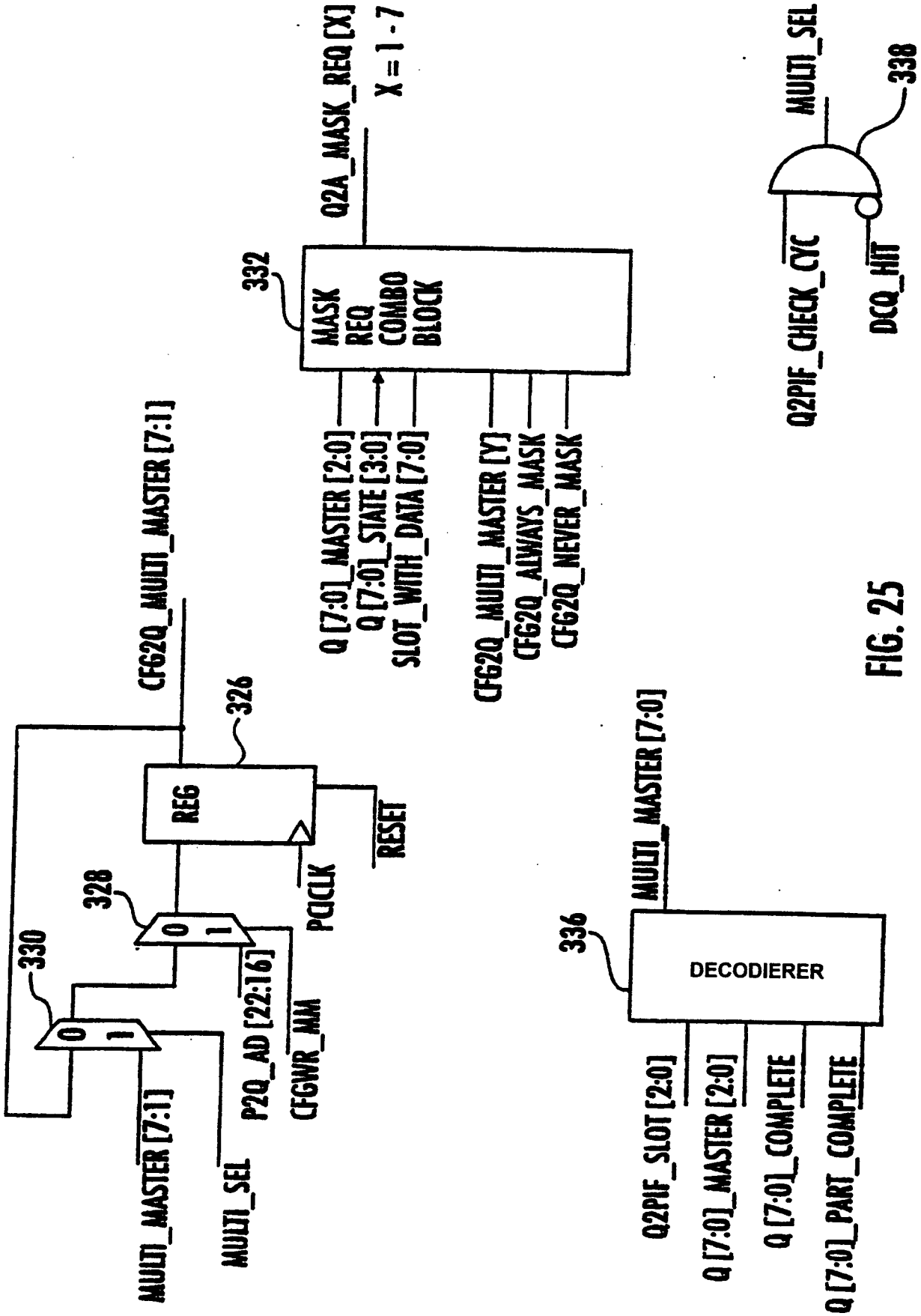

Q2A_MASK_REQ [X]
X = 1 - 7
332
MASK REQ COMBO BLOCK
Q [7:0]_MASTER [2:0]
Q [7:0]_STATE [3:0]
SLOT_WITH_DATA [7:0]
CFG2Q_MULTI_MASTER [Y]
CFG2Q_ALWAYS_MASK
CFG2Q_NEVER_MASK
Q2PIF_CHECK_CYC
MULTI_SEL
338
DCQ_HIT
CFG2Q_MULTI_MASTER [7:1]
326
REG
RESET
328
330
PCCLK
MULTI_MASTER [7:1]
MULTI_SEL
P2Q_AD [22:16]
CFGWR_MM
MULTI_MASTER [7:0]
336
DECODIERER
Q2PIF_SLOT [2:0]
Q [7:0]_MASTER [2:0]
Q [7:0]_COMPLETE
Q [7:0]_PART_COMPLETE

FIG. 25

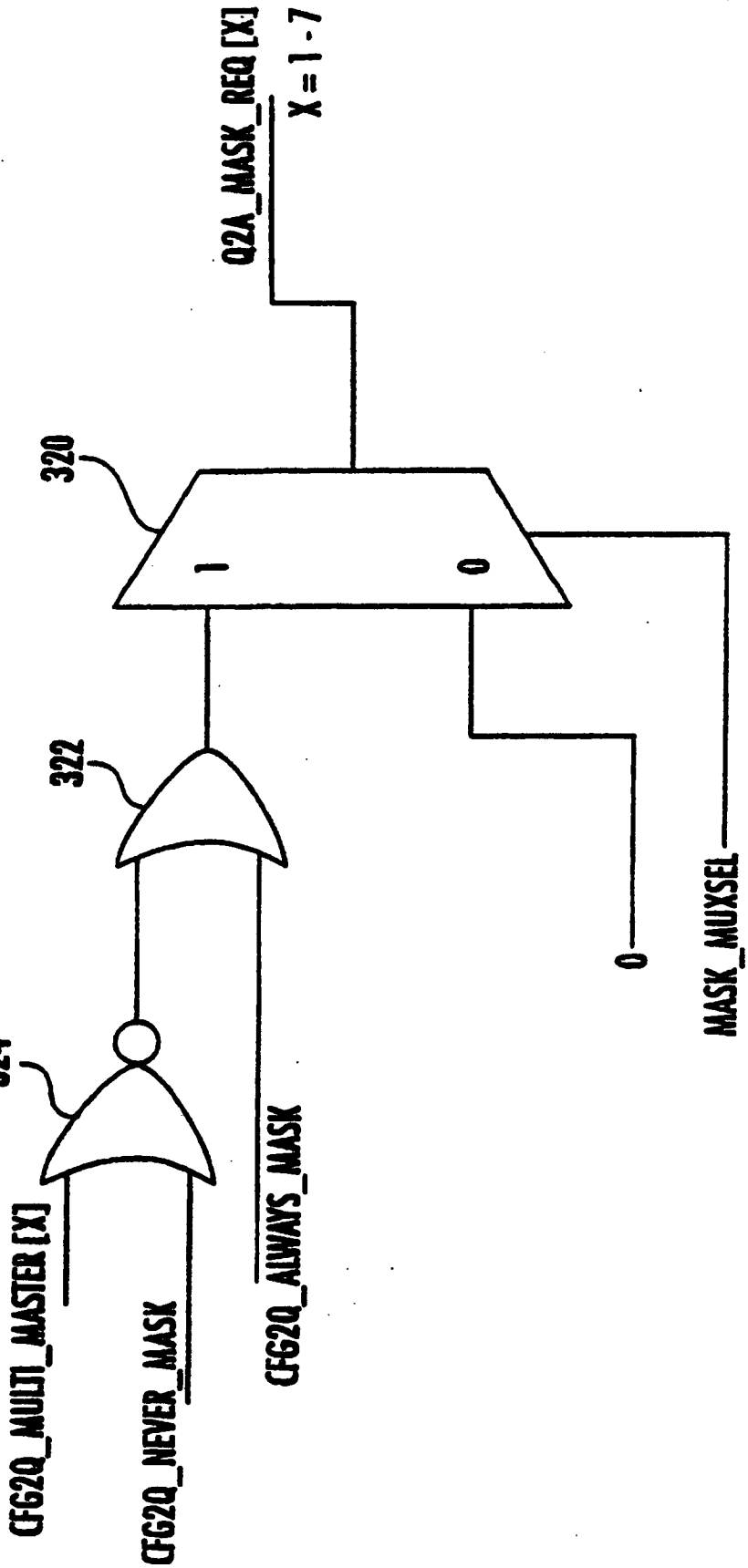

Q2A_MASK_REQ [X]
X = 1 - 7
320
1
0
322
324
0
MASK_MUXSEL
CFG2Q_MULTI_MASTER [X]
CFG2Q_NEVER_MASK
CFG2Q_ALWAYS_MASK

FIG. 26A

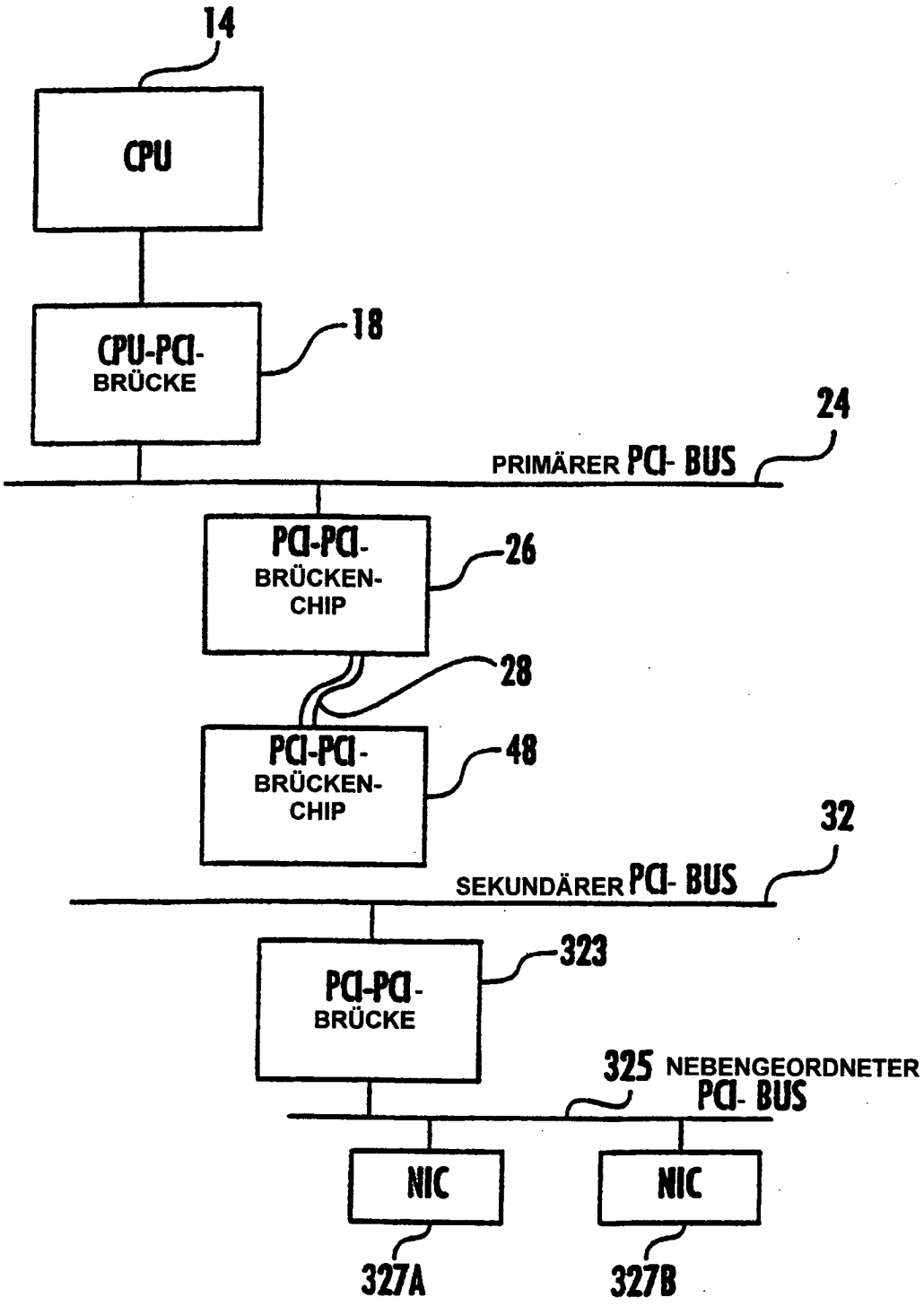

14
CPU
CPU-PCI-
BRÜCKE
18
24
PRIMÄRER PCI- BUS
PCI-PCI-
BRÜCKEN-
CHIP
26
28
PCI-PCI-
BRÜCKEN-
CHIP
48
32
SEKUNDÄRER PCI- BUS
PCI-PCI-
BRÜCKE
323
325
NEBENGEORDNETER
PCI- BUS
NIC
NIC
327A
327B

FIG. 26B

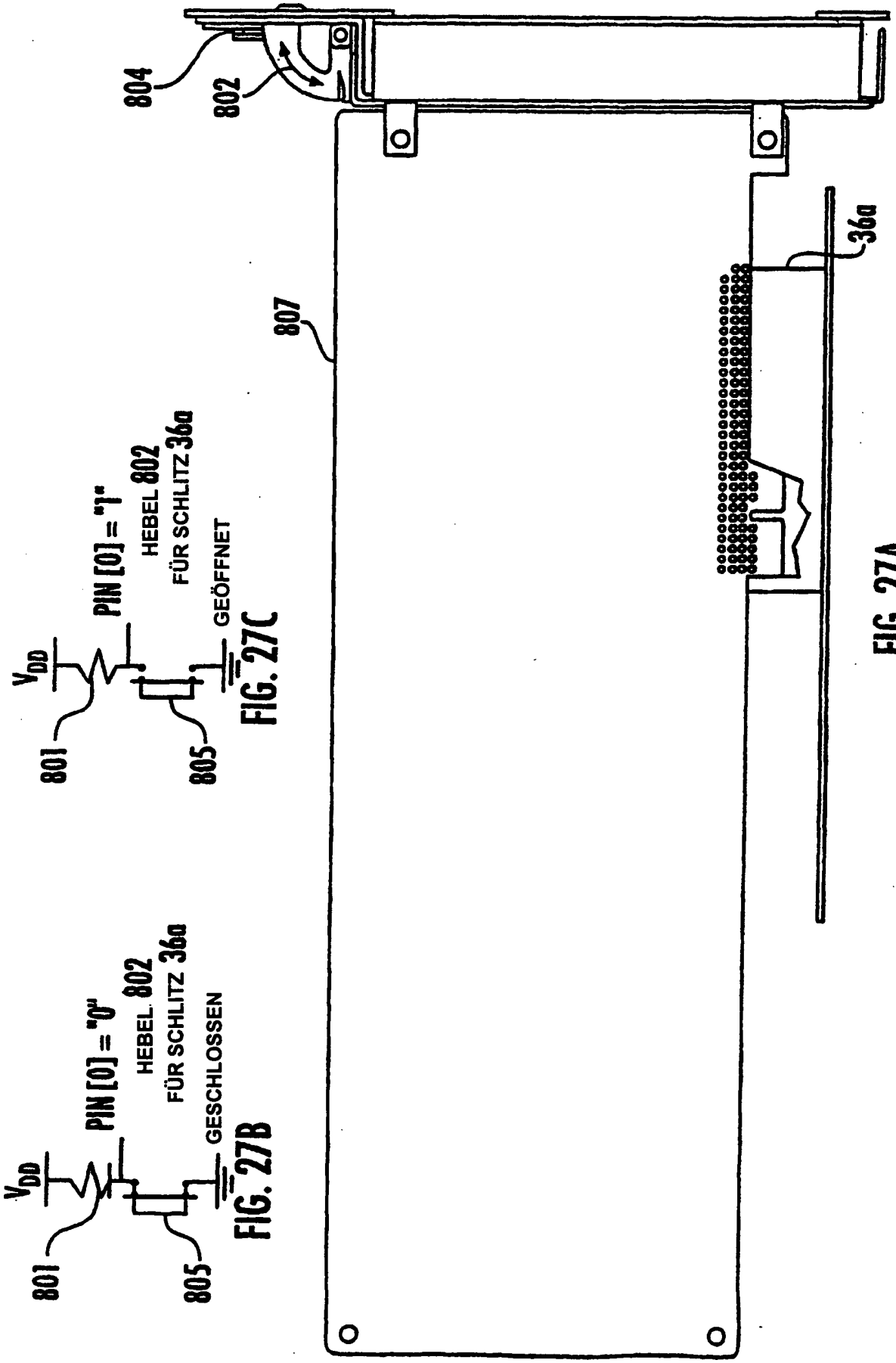


FIG. 27A

FIG. 27C

FIG. 27B

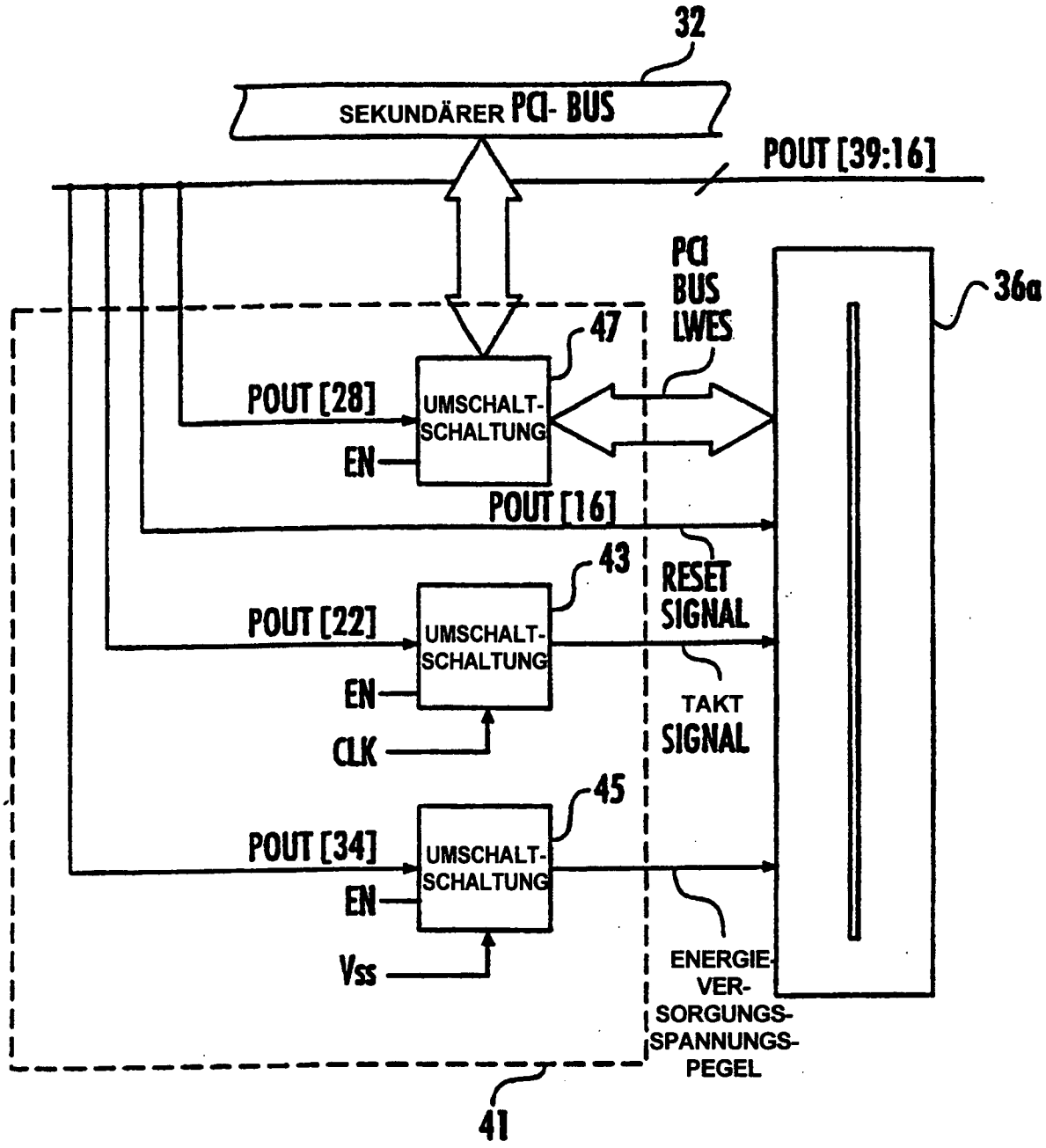

32
SEKUNDÄRER PCI- BUS
POUT [39:16]
PCI BUS LWES
36a
47
POUT [28]
UMSCHALT-SCHALTUNG
EN
POUT [16]
43
RESET SIGNAL
POUT [22]
UMSCHALT-SCHALTUNG
EN
CLK
TAKT SIGNAL
45
POUT [34]
UMSCHALT-SCHALTUNG
EN
Vss
ENERGIE-VER-SORGUNGS-SPANNUNGS-PEGEL
41

FIG. 28

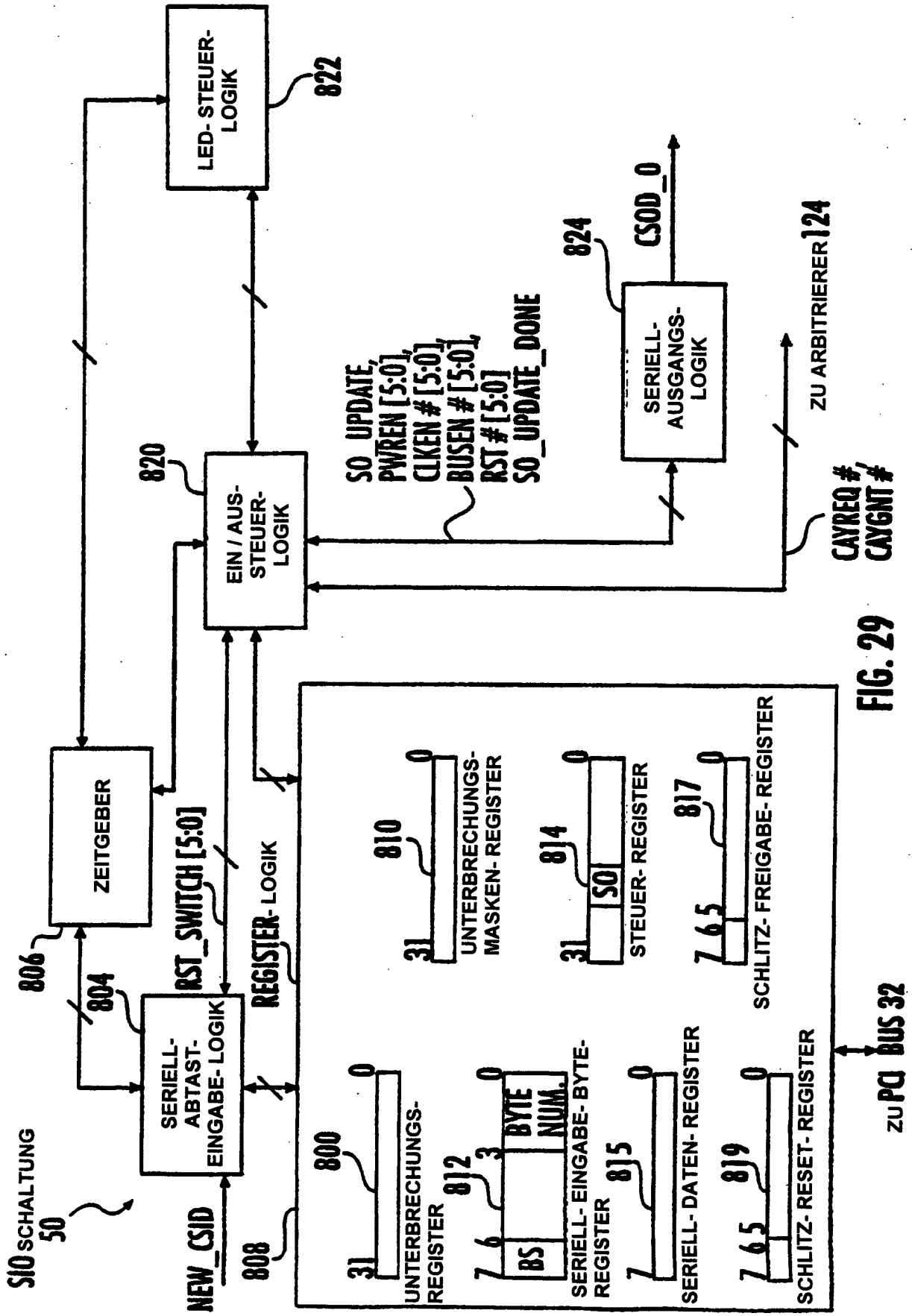

SIO SCHALTUNG 50
NEW_CSID
808
SERIELL-ABTAST-EINGABE-LOGIK
804
806
ZEITGEBER
RST_SWITCH [5:0]
REGISTER-LOGIK
820
EIN / AUS-STEUER-LOGIK
LED-STEUER-LOGIK
822
SO_UPDATE,
PWREN [5:0],
CLKEN # [5:0],
BUSEN # [5:0],
RST # [5:0]
SO_UPDATE_DONE
824
SERIELL-AUSGANGS-LOGIK
CSOD_0
ZU ARBITRIERER 124
CAYREQ #,
CAYGNT #
800
UNTERBRECHUNGS-REGISTER
810
UNTERBRECHUNGS-MASKEN-REGISTER
812
BS
BYTE NUM.
SERIELL-EINGABE-BYTE-REGISTER
814
SO
STEUER-REGISTER
815
SERIELL-DATEN-REGISTER
817
SCHLITZ-FREIGABE-REGISTER
819
SCHLITZ-RESET-REGISTER
ZU PCI BUS 32

FIG. 29

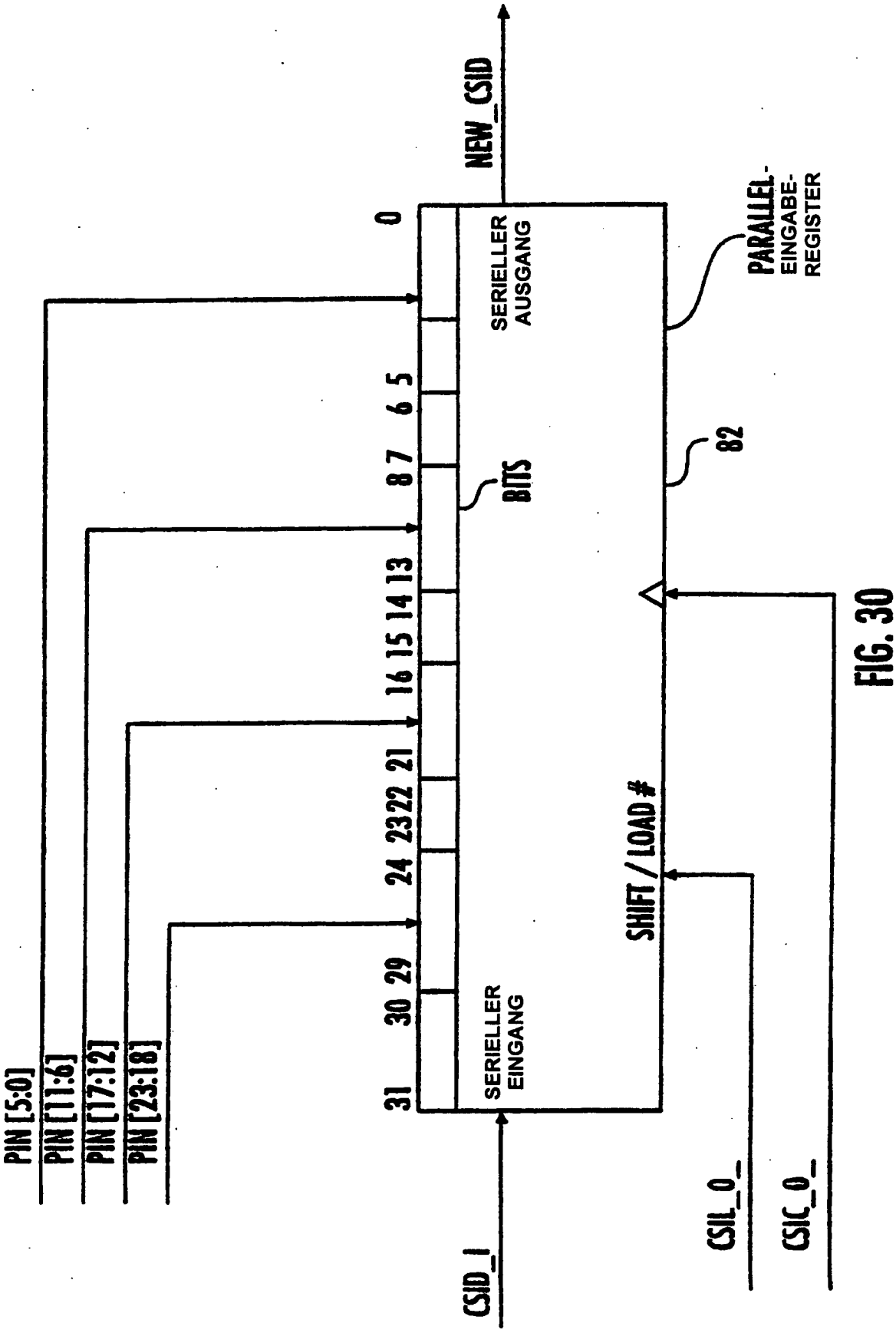

NEW_CSID
0
SERIELLER AUSGANG
PARALLEL-EINGABE-REGISTER
5
6
7
8
BITS
82
13
14
15
16
21
22
23
24
29
30
31
SERIELLER EINGANG
SHIFT/LOAD#
PIN [5:0]
PIN [11:6]
PIN [17:12]
PIN [23:18]
CSID_I
CSIL_O_
CSIC_O_

FIG. 30

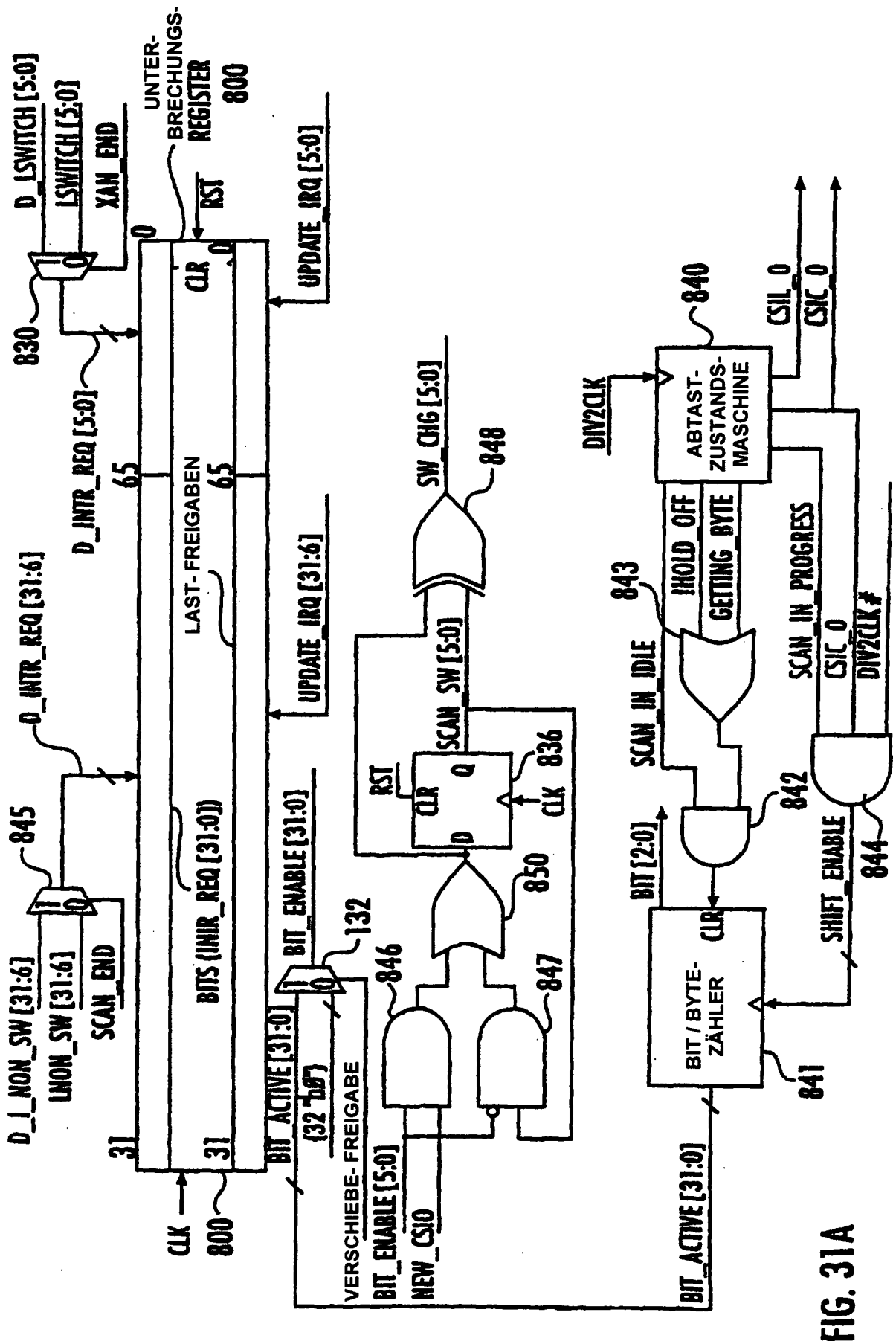

D_LSWITCH [5:0]
LSWITCH [5:0]
XAM_END
UNTER-BRECHUNGS-REGISTER
800
RST
CLR
UPDATE_IRQ [5:0]
830
D_INTR_REQ [5:0]
65
LAST-FREIGABEN
65
D_INTR_REQ [31:6]
D_LNON_SW [31:6]
LNON_SW [31:6]
SCAN_END
845
BITS (INTR_REQ [31:0])
BIT_ACTIVE [31:0]
(32 'b0)
BIT_ENABLE [31:0]
UPDATE_IRQ [31:6]
132
VERSCHIEBE-FREIGABE
BIT_ENABLE [5:0]
NEW_CSIO
CLK
800
SW_CHG [5:0]
848
SCAN_SW [5:0]
RST
CLR
Q
D
CLK
836
850
846
847
DIV2CLK
ABTAST-ZUSTANDS-MASCHINE
840
CSIL_0
CSIC_0
IHOLD_OFF
GETTING_BYTE
843
SCAN_IN_IDLE
SCAN_IN_PROGRESS
CSIC_0
DIV2CLK
BIT [2:0]
842
SHIFT_ENABLE
844
BIT / BYTE-ZÄHLER
CLR
841
BIT_ACTIVE [31:0]

FIG. 31A

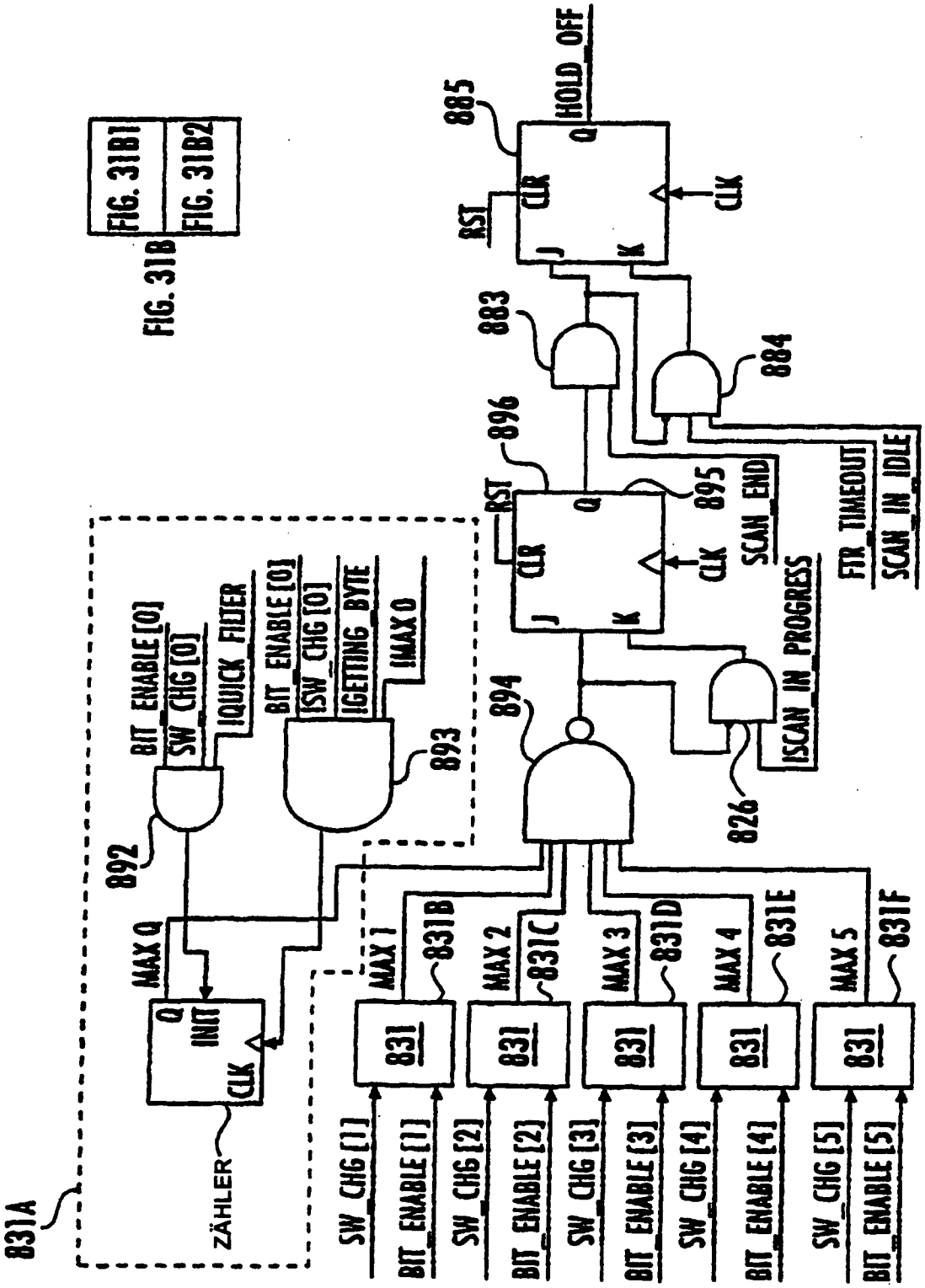


FIG. 31B1
FIG. 31B2
FIG. 31B
831A
ZÄHLER
Q
INIT
CLK
MAX Q
BIT_ENABLE[0]
SW_CHG[0]
!QUICK_FILTER
892
BIT_ENABLE[0]
!SW_CHG[0]
!GETTING_BYTE
!MAX 0
893
894
RST
CLR
J
K
Q
CLK
896
895
SCAN_END
!SCAN_IN_PROGRESS
826
FTR_TIMEOUT
SCAN_IN_IDLE
883
884
885
HOLD_OFF
RST
CLR
J
K
Q
CLK
MAX 1
831B
MAX 2
831C
MAX 3
831D
MAX 4
831E
MAX 5
831F
831
SW_CHG[1]
BIT_ENABLE[1]
SW_CHG[2]
BIT_ENABLE[2]
SW_CHG[3]
BIT_ENABLE[3]
SW_CHG[4]
BIT_ENABLE[4]
SW_CHG[5]
BIT_ENABLE[5]

FIG. 31B1

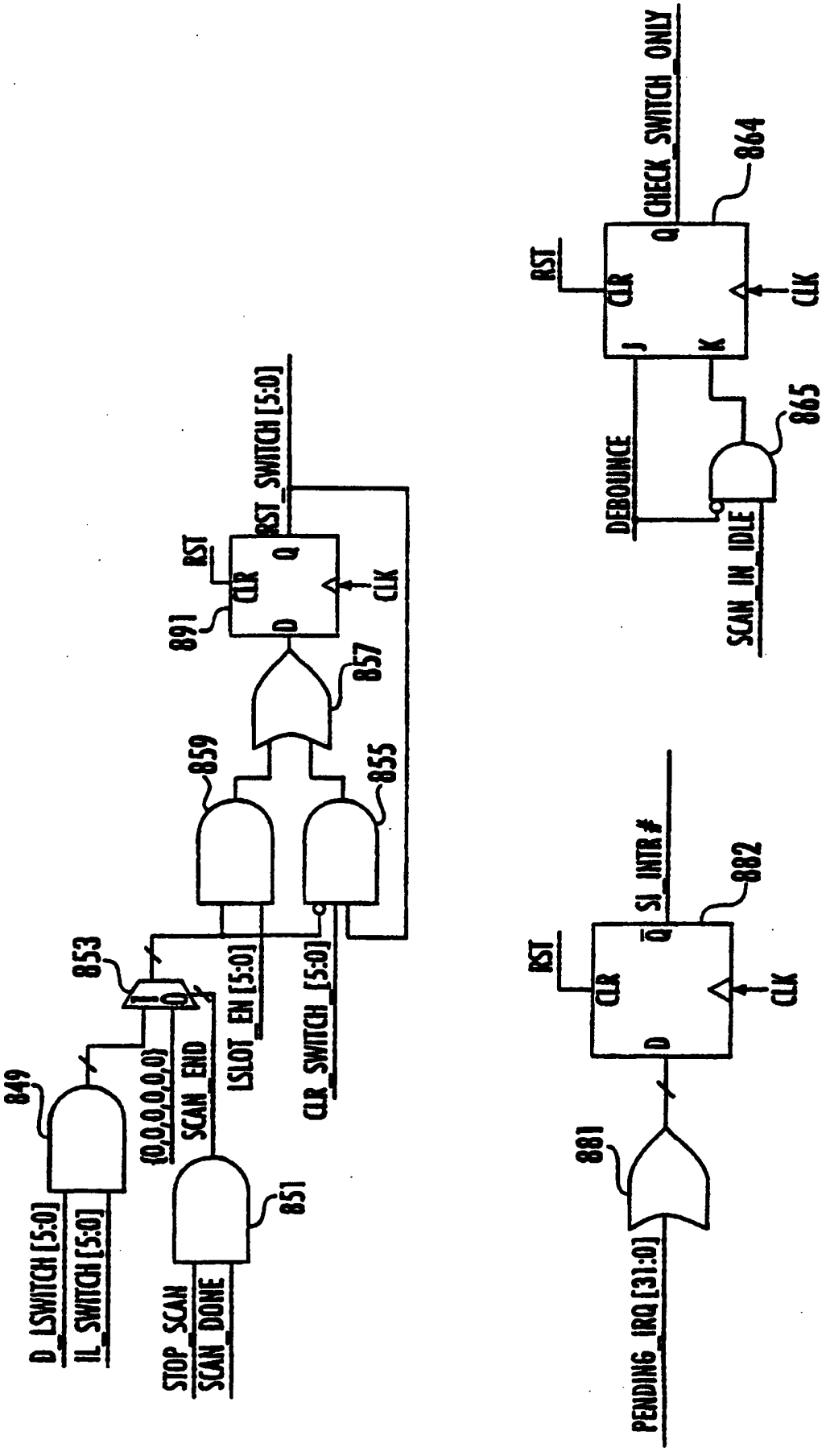

D_LSWITCH [5:0]
IL_SWITCH [5:0]
849
{0,0,0,0,0,0}
SCAN_END
STOP_SCAN
SCAN_DONE
851
853
LSLOT_EN [5:0]
CLR_SWITCH [5:0]
855
857
859
RST_SWITCH [5:0]
RST
CLR
CLK
D
Q
891
DEBOUNCE
SCAN_IN_IDLE
865
CHECK_SWITCH_ONLY
J
K
864
PENDING_IRQ [31:0]
881
SI_INTR#
882

FIG. 31B2

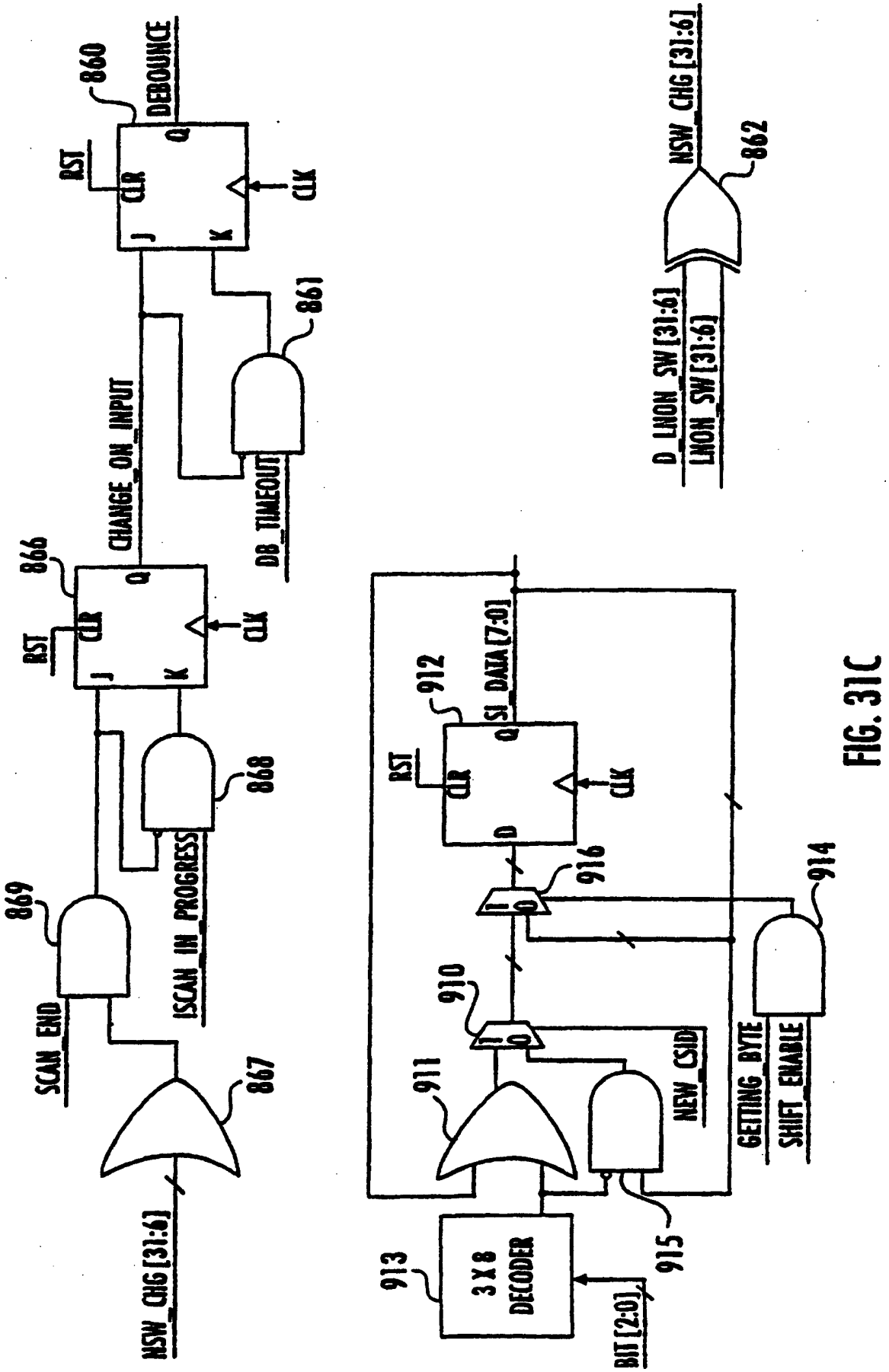

860
DEBOUNCE
RST
CLR
J
K
Q
CLK
861
CHANGE_ON_INPUT
DB_TIMEOUT
866
RST
CLR
J
K
Q
CLK
868
869
SCAN_END
ISCAN_IN_PROGRESS
867
NSW_CHG[31:6]
D_LNON_SW[31:6]
LNON_SW[31:6]
862
NSW_CHG[31:6]
912
SI_DATA[7:0]
RST
CLR
D
Q
CLK
916
914
910
911
NEW_CSID
GETTING_BYTE
SHIFT_ENABLE
913
3X8
DECODER
915
BIT[2:0]

FIG. 31C

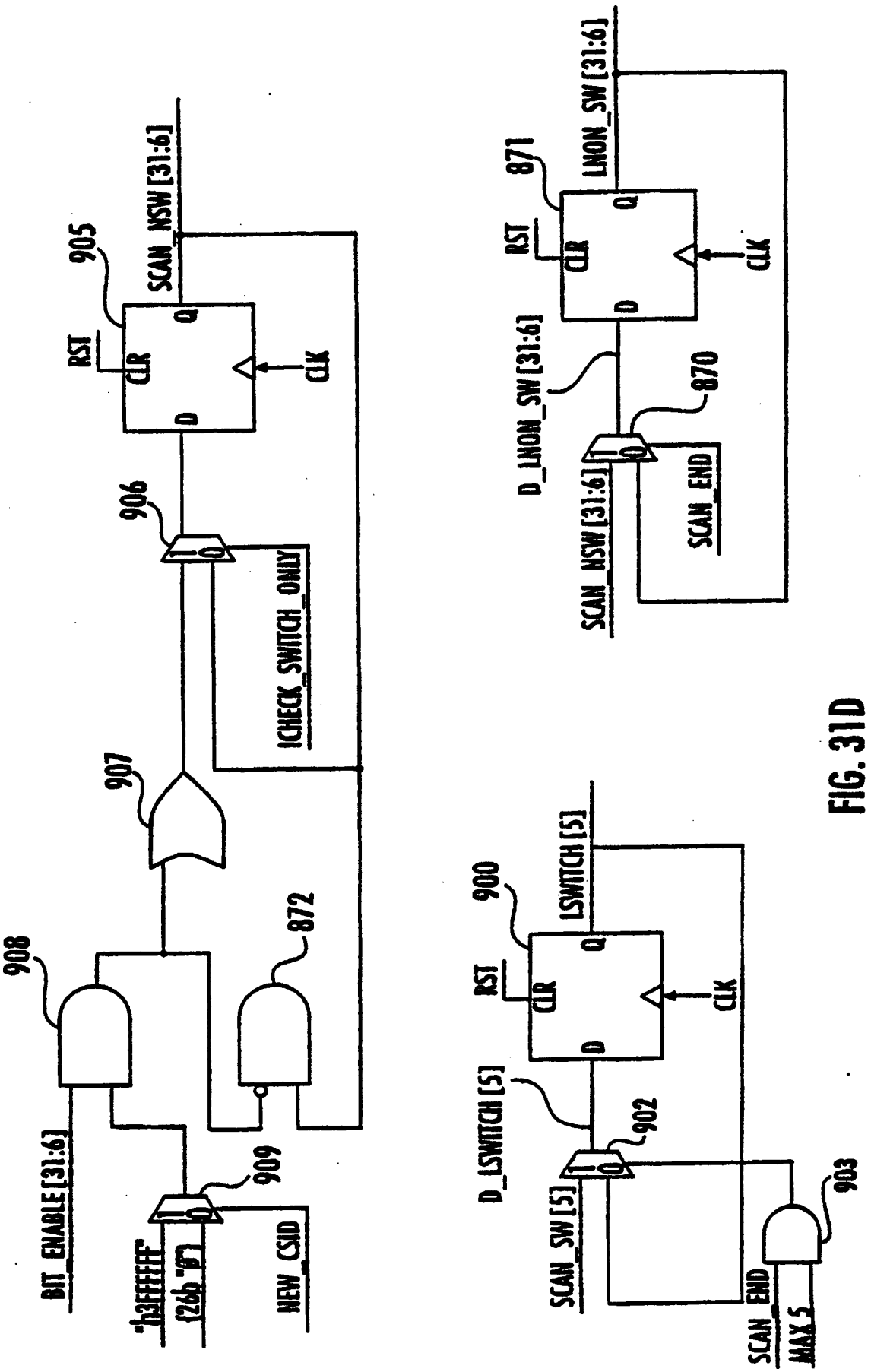

LNON_SW [31:6]
871
RST
CLR
Q
D
CLK
D_LNON_SW [31:6]
870
SCAN_NSW [31:6]
SCAN_END
SCAN_NSW [31:6]
905
RST
CLR
Q
D
CLK
906
ICHECK_SWITCH_ONLY
907
908
872
BIT_ENABLE [31:6]
"h3FFFFFFF"
[26 "0"]
909
NEW_CSID
LSWITCH [5]
900
RST
CLR
Q
D
CLK
D_LSWITCH [5]
902
SCAN_SW [5]
903
SCAN_END
MAX 5

FIG. 31D

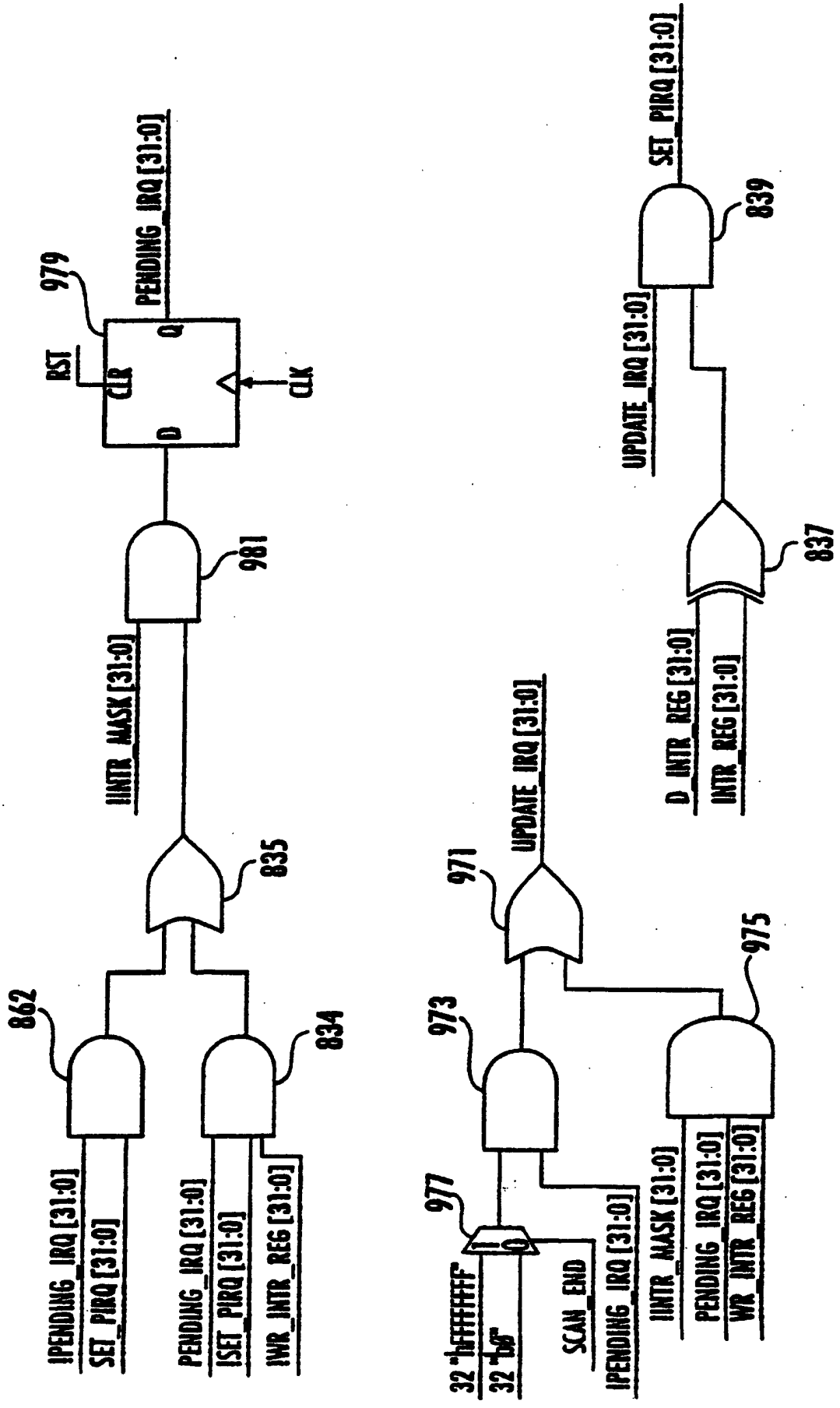
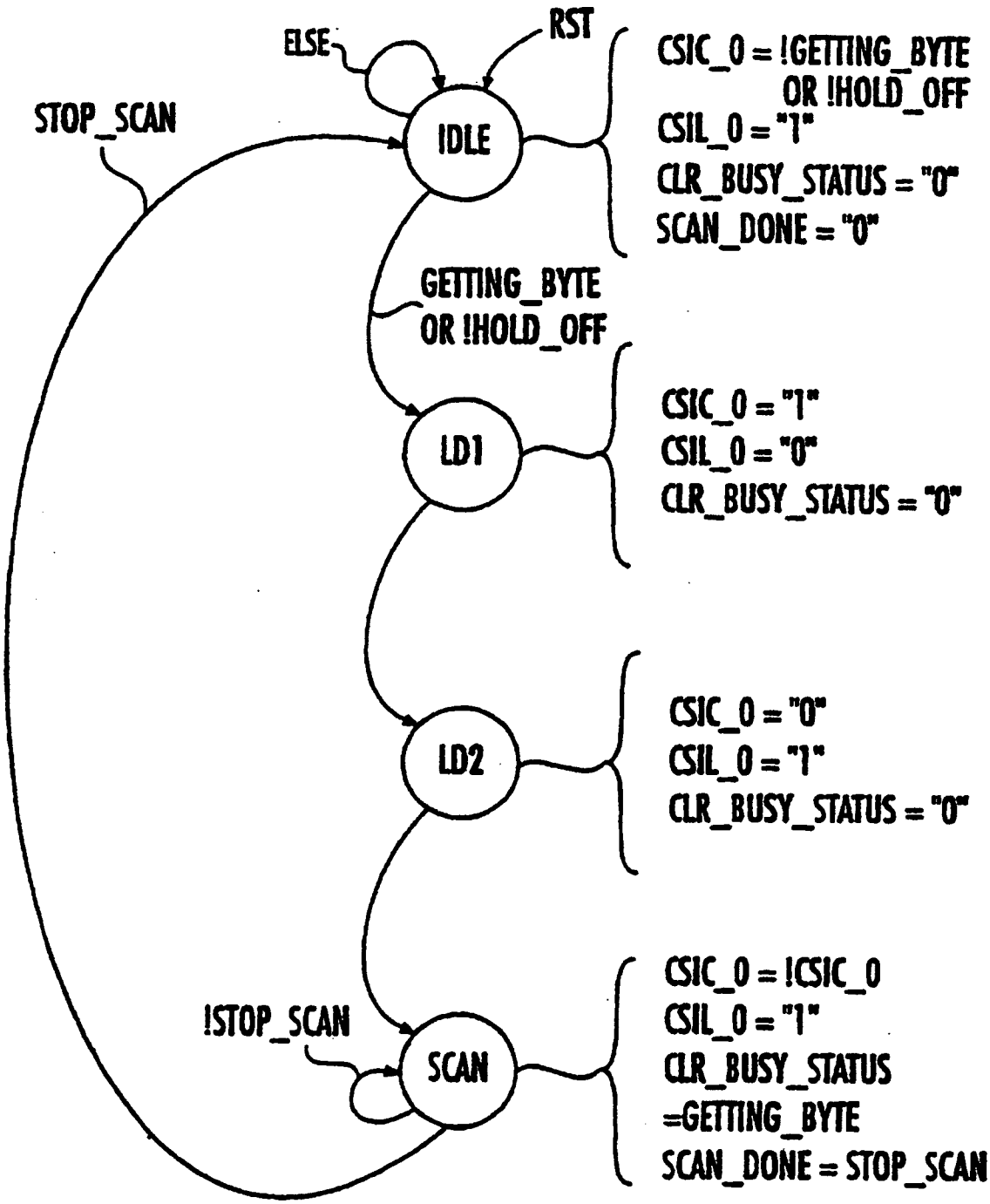

IPENDING_IRQ [31:0]
SET_PIRQ [31:0]
PENDING_IRQ [31:0]
ISET_PIRQ [31:0]
IWR_INTR_REG [31:0]
862
834
835
IINTR_MASK [31:0]
981
979
PENDING_IRQ [31:0]
RST
CLR
D
Q
CLK
32 'hFFFFFFFF
32 'h0
977
973
971
UPDATE_IRQ [31:0]
SCAN_END
IPENDING_IRQ [31:0]
IINTR_MASK [31:0]
PENDING_IRQ [31:0]
WR_INTR_REG [31:0]
975
D_INTR_REG [31:0]
INTR_REG [31:0]
837
UPDATE_IRQ [31:0]
839
SET_PIRQ [31:0]

FIG. 31E



STOP_SCAN = (BYTE_PTR_EQUAL_CNT & GETTING_BYTE) OR ((BYTE [1] & BIT [0] & CHECK_SWITCH_ONLY) OR (BYTE [4] & BIT [0] & !CHECK_SWITCH_ONLY)) & !GETTING_BYTE

FIG. 32A

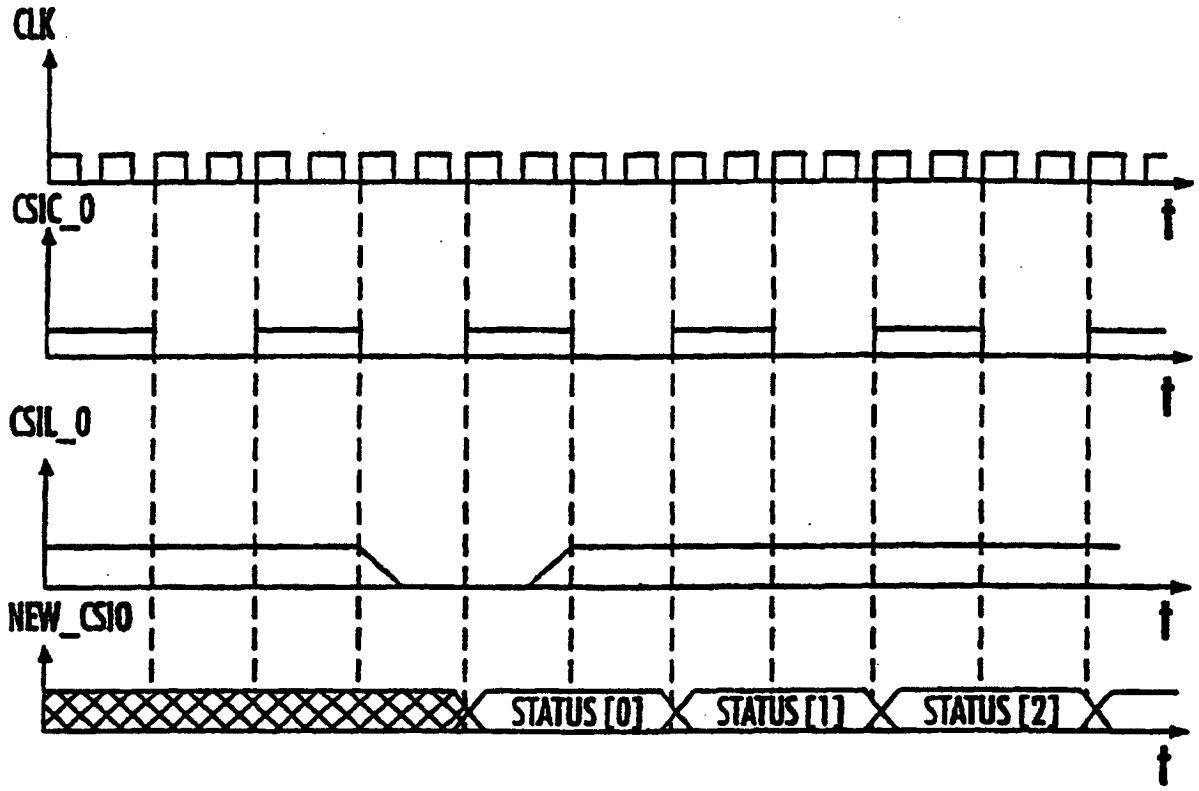


FIG. 32B

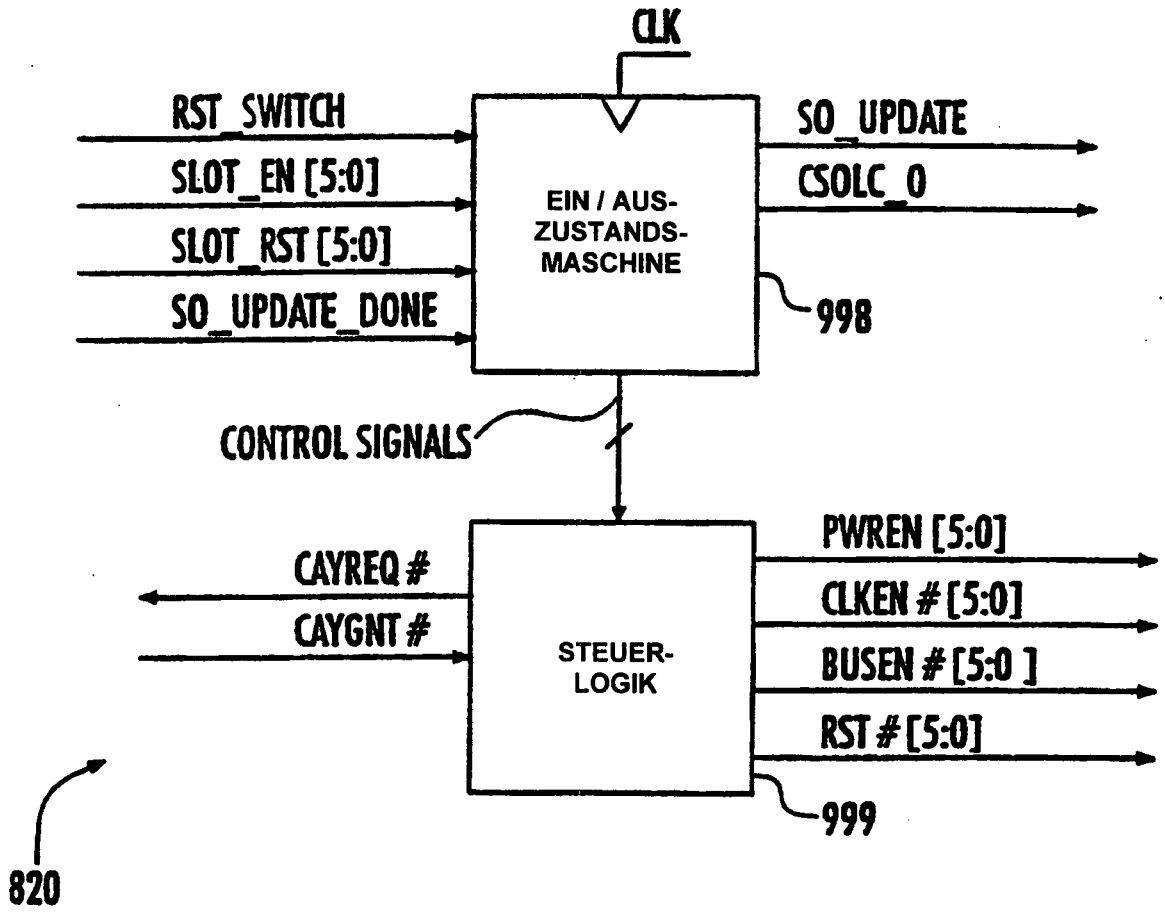

CLK
RST_SWITCH
SLOT_EN [5:0]
SLOT_RST [5:0]
SO_UPDATE_DONE
EIN / AUS-
ZUSTANDS-
MASCHINE
SO_UPDATE
CSOLC_0
998
CONTROL SIGNALS
CAYREQ #
CAYGNT #
STEUER-
LOGIK
PWREN [5:0]
CLKEN # [5:0]
BUSEN # [5:0]
RST # [5:0]
999
820

FIG. 33A

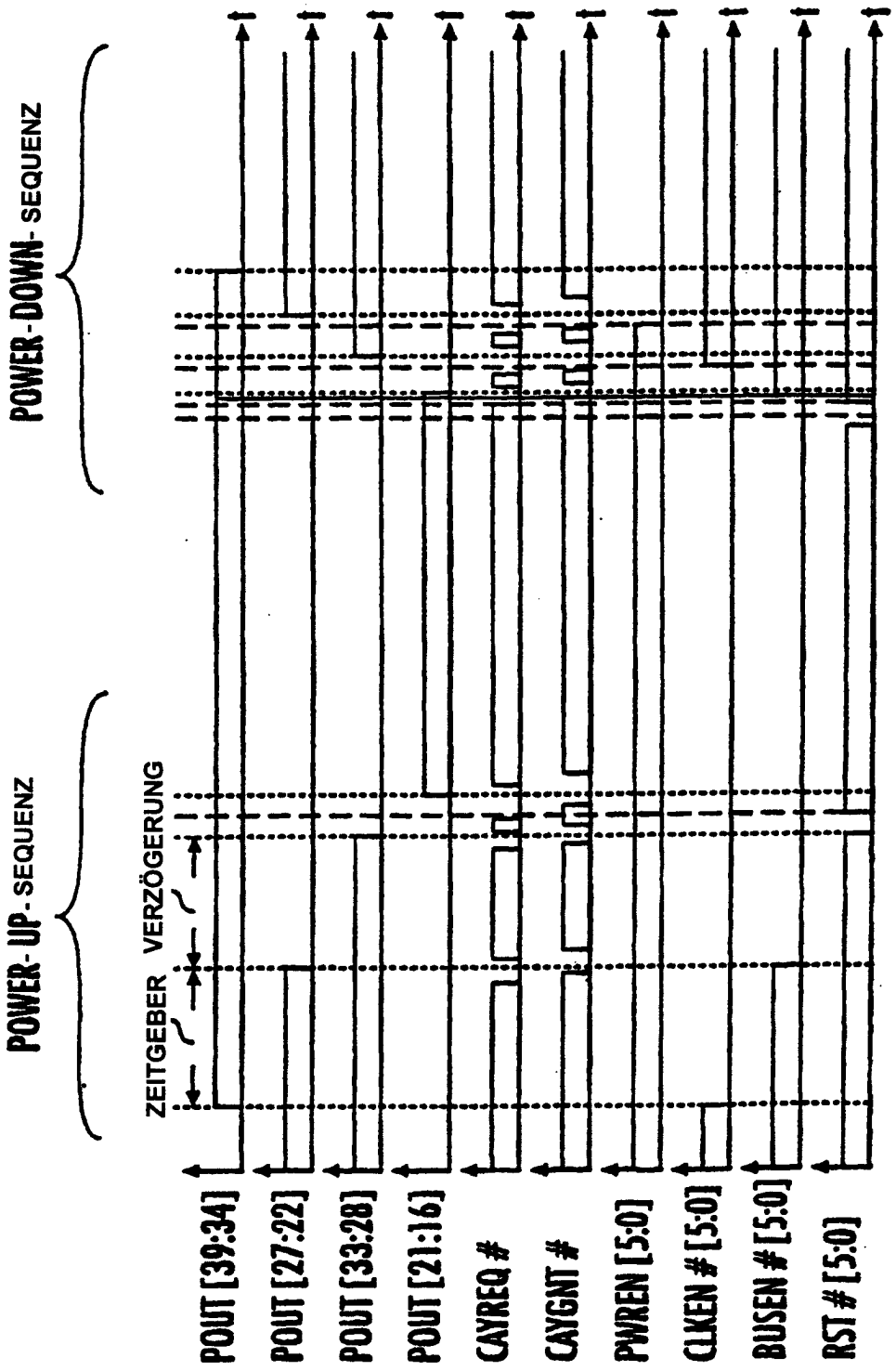

POWER-UP-SEQUENZ
POWER-DOWN-SEQUENZ
ZEITGEBER VERZÖGERUNG
POUT [39:34]
POUT [27:22]
POUT [33:28]
POUT [21:16]
CAYREQ #
CAYGNT #
PWREN [5:0]
CLKEN # [5:0]
BUSEN # [5:0]
RST # [5:0]

FIG. 33B

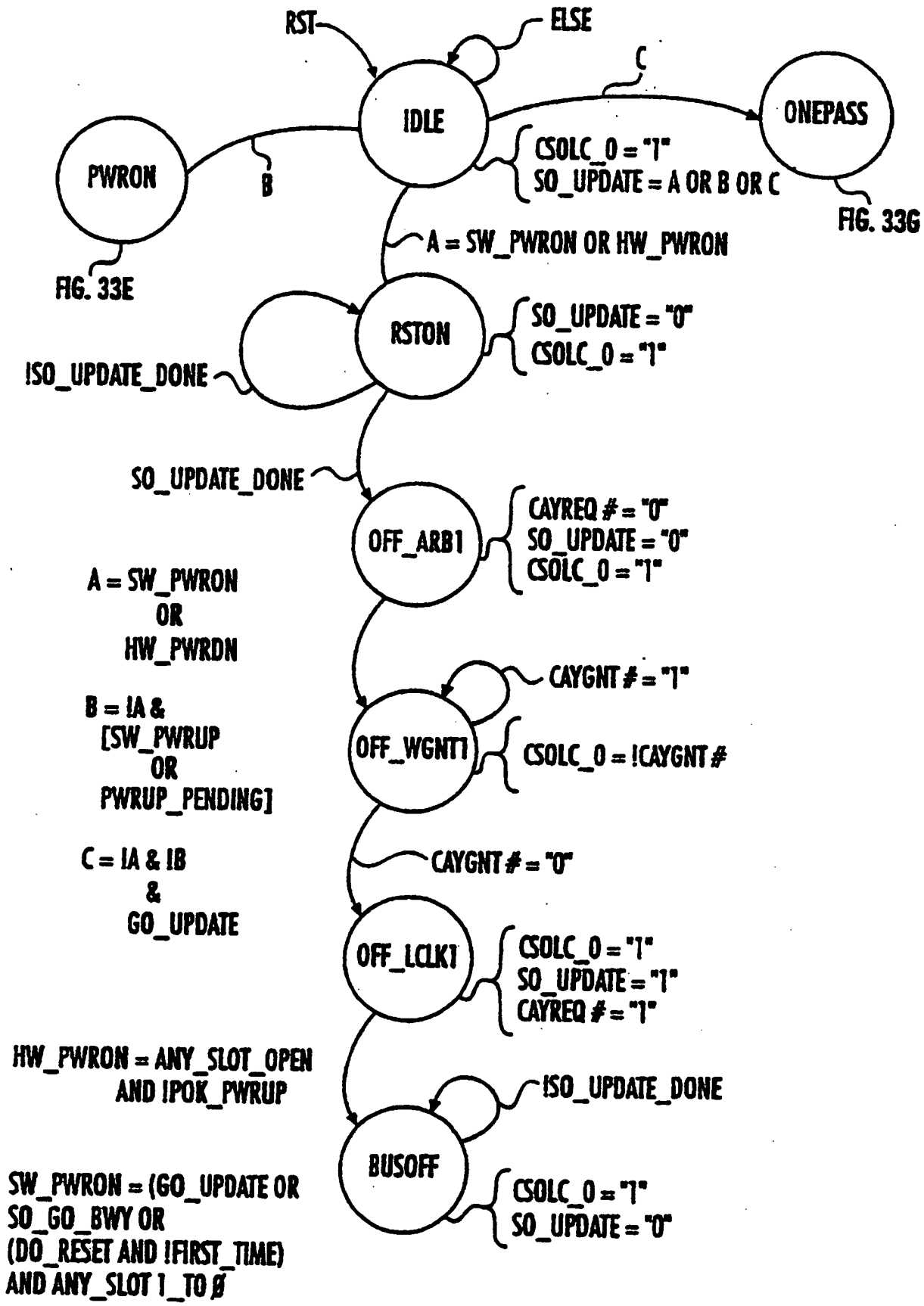

RST
ELSE
C
IDLE
ONEPASS
PWRON
B
CSOLC_0 = "1"
SO_UPDATE = A OR B OR C
FIG. 33G
FIG. 33E
A = SW_PWRON OR HW_PWRON
RSTON
SO_UPDATE = "0"
CSOLC_0 = "1"
!SO_UPDATE_DONE
SO_UPDATE_DONE
OFF_ARB1
CAYREQ # = "0"
SO_UPDATE = "0"
CSOLC_0 = "1"
A = SW_PWRON
OR
HW_PWRDN
CAYGNT # = "1"
B = !A &
[SW_PWRUP
OR
PWRUP_PENDING]
OFF_WGNT1
CSOLC_0 = !CAYGNT #
C = !A & !B
&
GO_UPDATE
CAYGNT # = "0"
OFF_LCLK1
CSOLC_0 = "1"
SO_UPDATE = "1"
CAYREQ # = "1"
HW_PWRON = ANY_SLOT_OPEN
AND !POK_PWRUP
!SO_UPDATE_DONE
BUSOFF
CSOLC_0 = "1"
SO_UPDATE = "0"
SW_PWRON = (GO_UPDATE OR
SO_GO_BWY OR
(DO_RESET AND !FIRST_TIME)
AND ANY_SLOT 1_TO ø

FIG. 33C

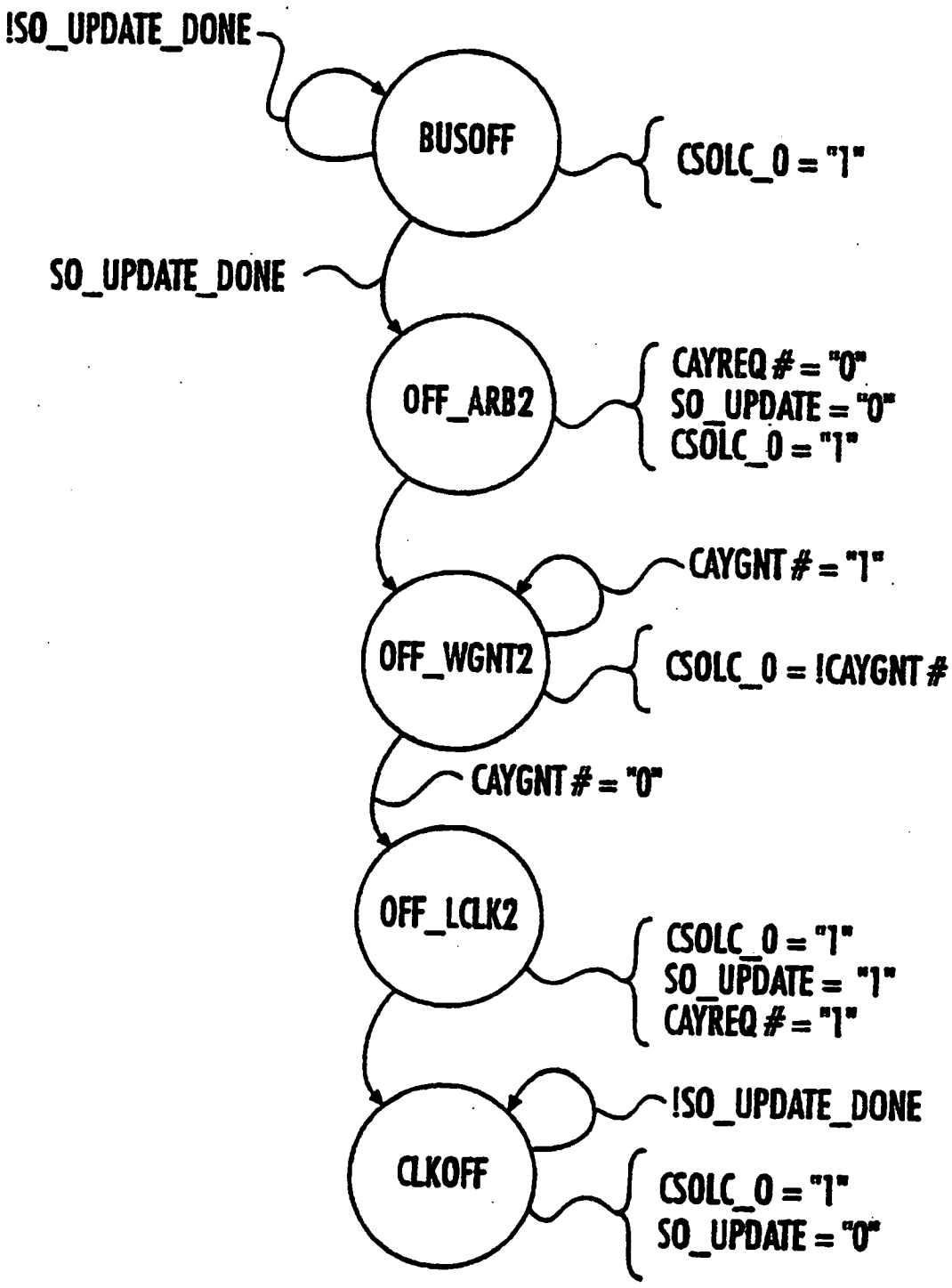

!SO_UPDATE_DONE
BUSOFF
CSOLC_0 = "1"
SO_UPDATE_DONE
OFF_ARB2
CAYREQ# = "0"
SO_UPDATE = "0"
CSOLC_0 = "1"
CAYGNT# = "1"
OFF_WGNT2
CSOLC_0 = !CAYGNT#
CAYGNT# = "0"
OFF_LCLK2
CSOLC_0 = "1"
SO_UPDATE = "1"
CAYREQ# = "1"
!SO_UPDATE_DONE
CLKOFF
CSOLC_0 = "1"
SO_UPDATE = "0"

FIG. 33D

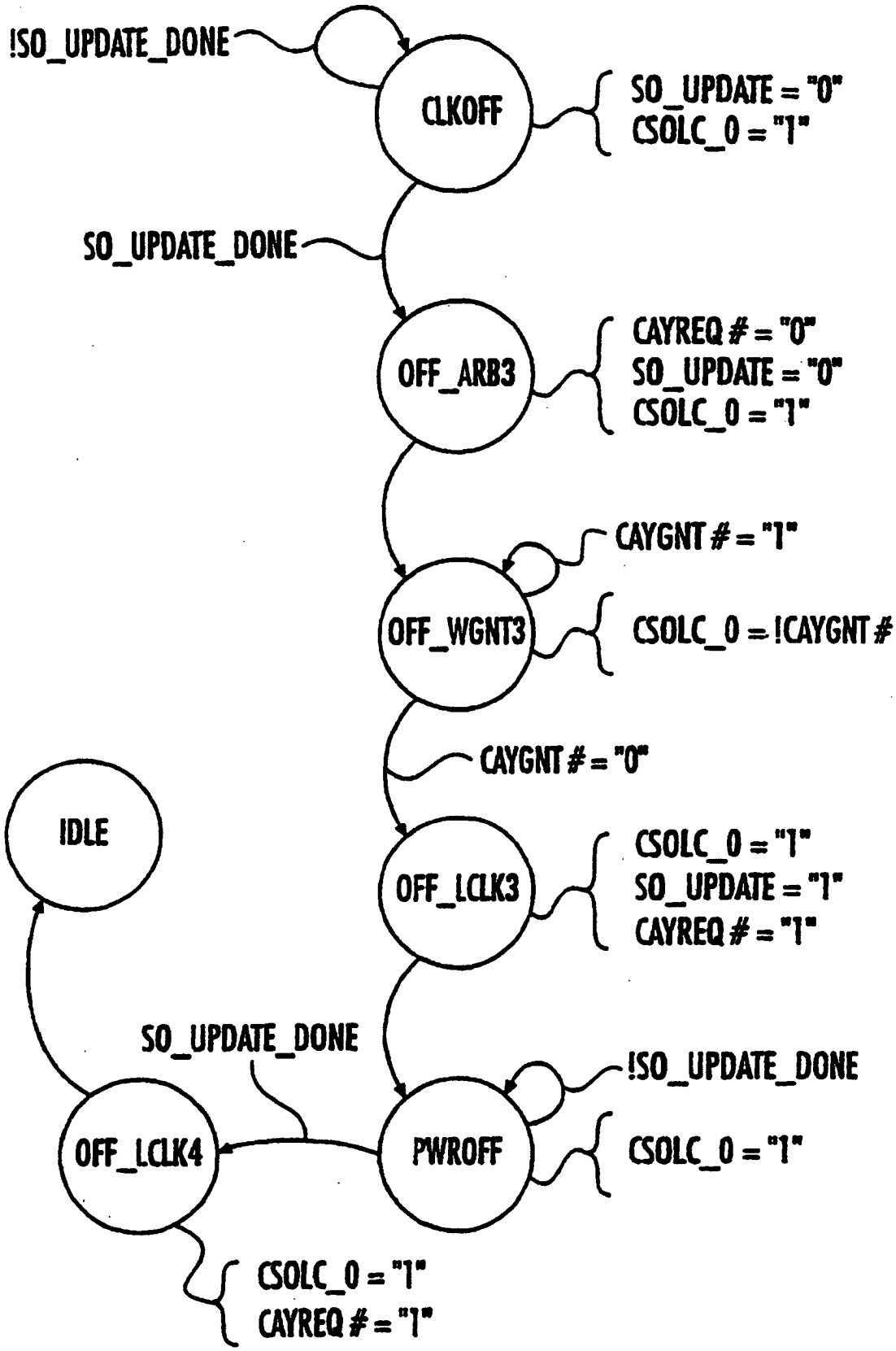

!SO_UPDATE_DONE
CLKOFF
SO_UPDATE = "0"
CSOLC_0 = "1"
SO_UPDATE_DONE
OFF_ARB3
CAYREQ # = "0"
SO_UPDATE = "0"
CSOLC_0 = "1"
CAYGNT # = "1"
OFF_WGNT3
CSOLC_0 = !CAYGNT #
CAYGNT # = "0"
IDLE
OFF_LCLK3
CSOLC_0 = "1"
SO_UPDATE = "1"
CAYREQ # = "1"
SO_UPDATE_DONE
!SO_UPDATE_DONE
OFF_LCLK4
PWROFF
CSOLC_0 = "1"
CSOLC_0 = "1"
CAYREQ # = "1"

FIG. 33E

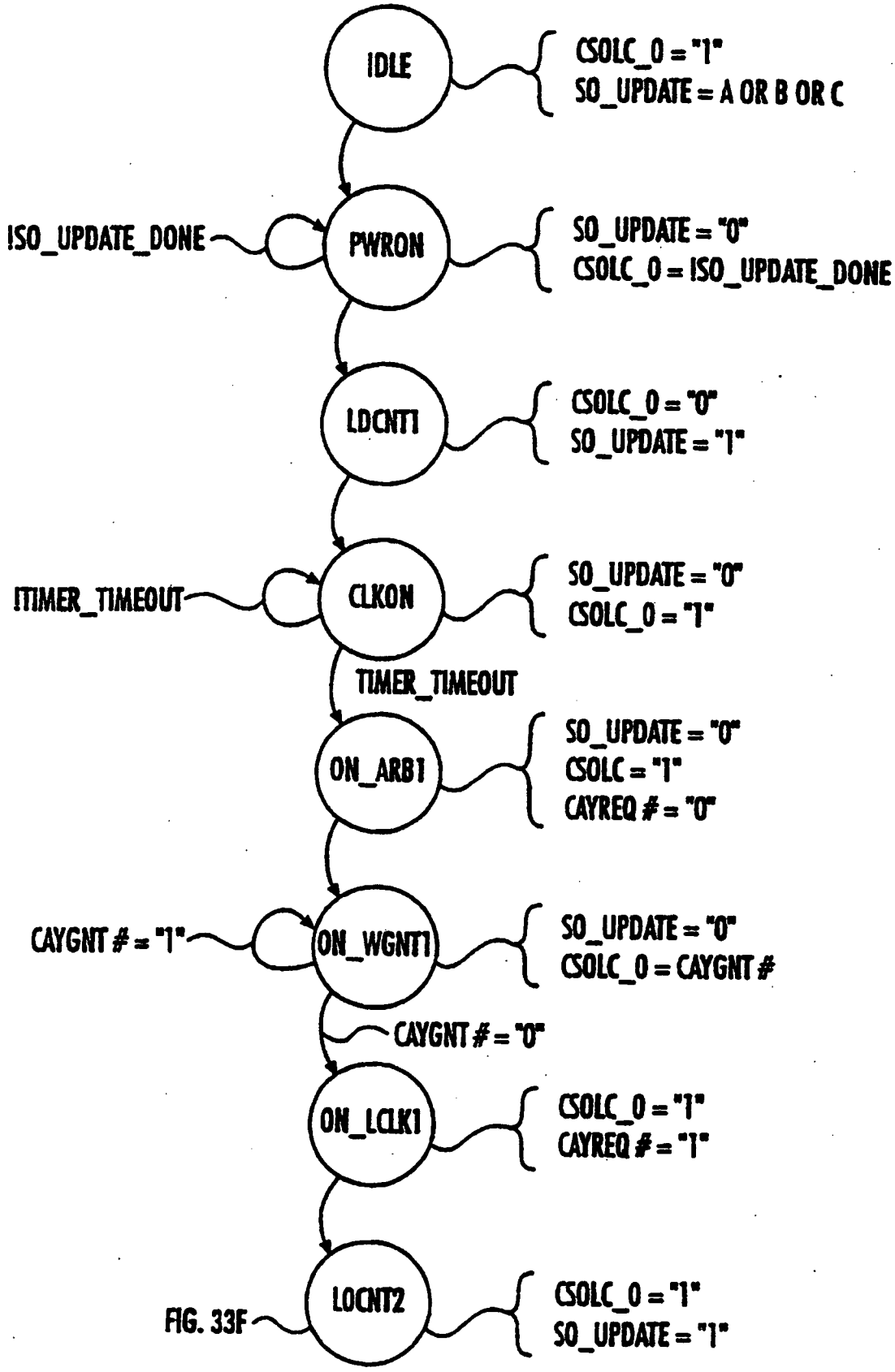

IDLE
CSOLC_0 = "1"
SO_UPDATE = A OR B OR C
ISO_UPDATE_DONE
PWRON
SO_UPDATE = "0"
CSOLC_0 = ISO_UPDATE_DONE
LDCNT1
CSOLC_0 = "0"
SO_UPDATE = "1"
ITIMER_TIMEOUT
CLKON
SO_UPDATE = "0"
CSOLC_0 = "1"
TIMER_TIMEOUT
ON_ARB1
SO_UPDATE = "0"
CSOLC = "1"
CAYREQ # = "0"
CAYGNT # = "1"
ON_WGNT1
SO_UPDATE = "0"
CSOLC_0 = CAYGNT #
CAYGNT # = "0"
ON_LCLK1
CSOLC_0 = "1"
CAYREQ # = "1"
FIG. 33F
LOCNT2
CSOLC_0 = "1"
SO_UPDATE = "1"

FIG. 33F

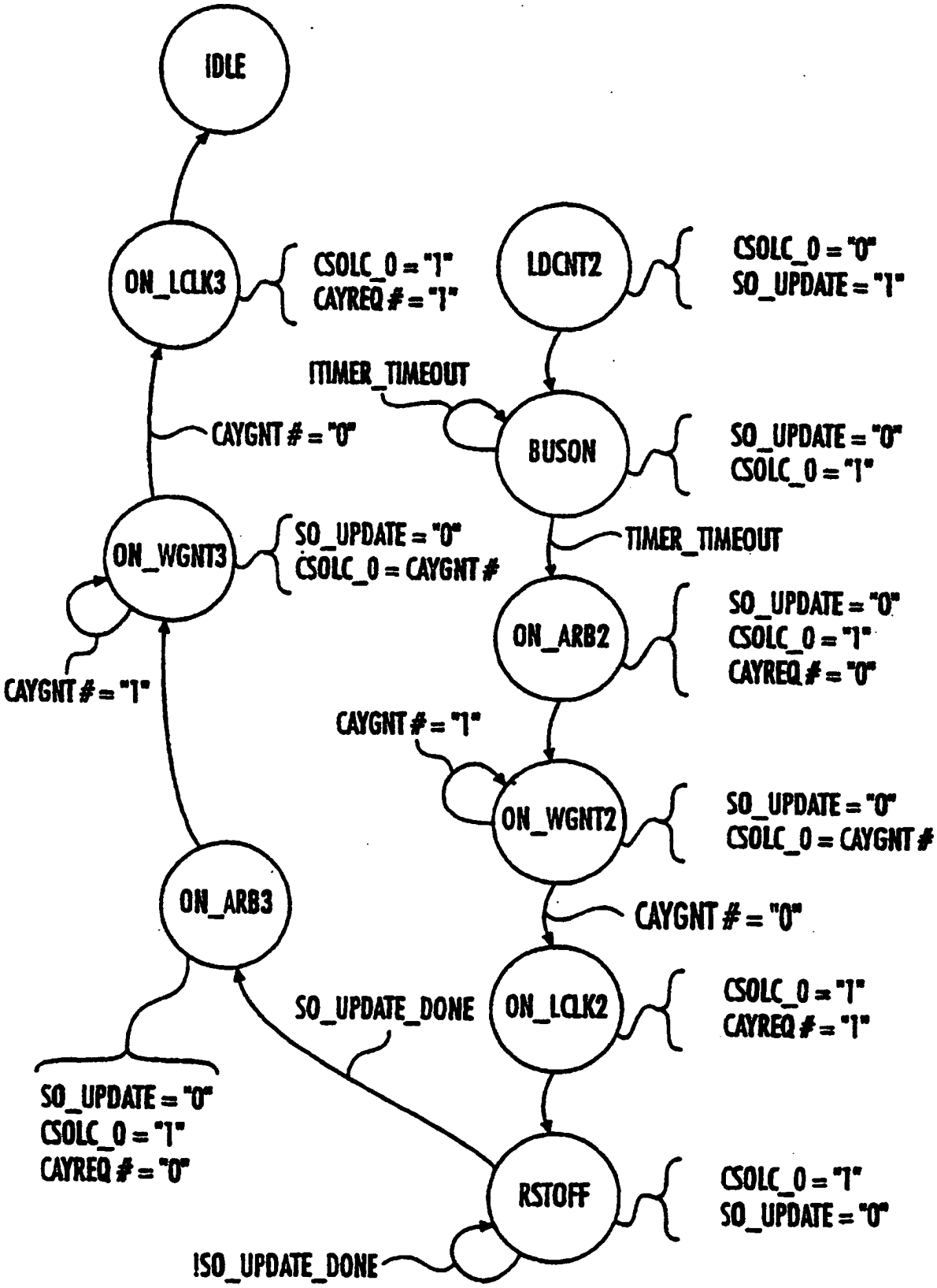

IDLE
ON_LCLK3
CSOLC_0 = "1"
CAYREQ # = "1"
LDCNT2
CSOLC_0 = "0"
SO_UPDATE = "1"
!TIMER_TIMEOUT
CAYGNT # = "0"
BUSON
SO_UPDATE = "0"
CSOLC_0 = "1"
ON_WGNT3
SO_UPDATE = "0"
CSOLC_0 = CAYGNT #
TIMER_TIMEOUT
ON_ARB2
SO_UPDATE = "0"
CSOLC_0 = "1"
CAYREQ # = "0"
CAYGNT # = "1"
CAYGNT # = "1"
ON_WGNT2
SO_UPDATE = "0"
CSOLC_0 = CAYGNT #
ON_ARB3
CAYGNT # = "0"
SO_UPDATE_DONE
ON_LCLK2
CSOLC_0 = "1"
CAYREQ # = "1"
SO_UPDATE = "0"
CSOLC_0 = "1"
CAYREQ # = "0"
RSTOFF
CSOLC_0 = "1"
SO_UPDATE = "0"
!SO_UPDATE_DONE

FIG. 33G

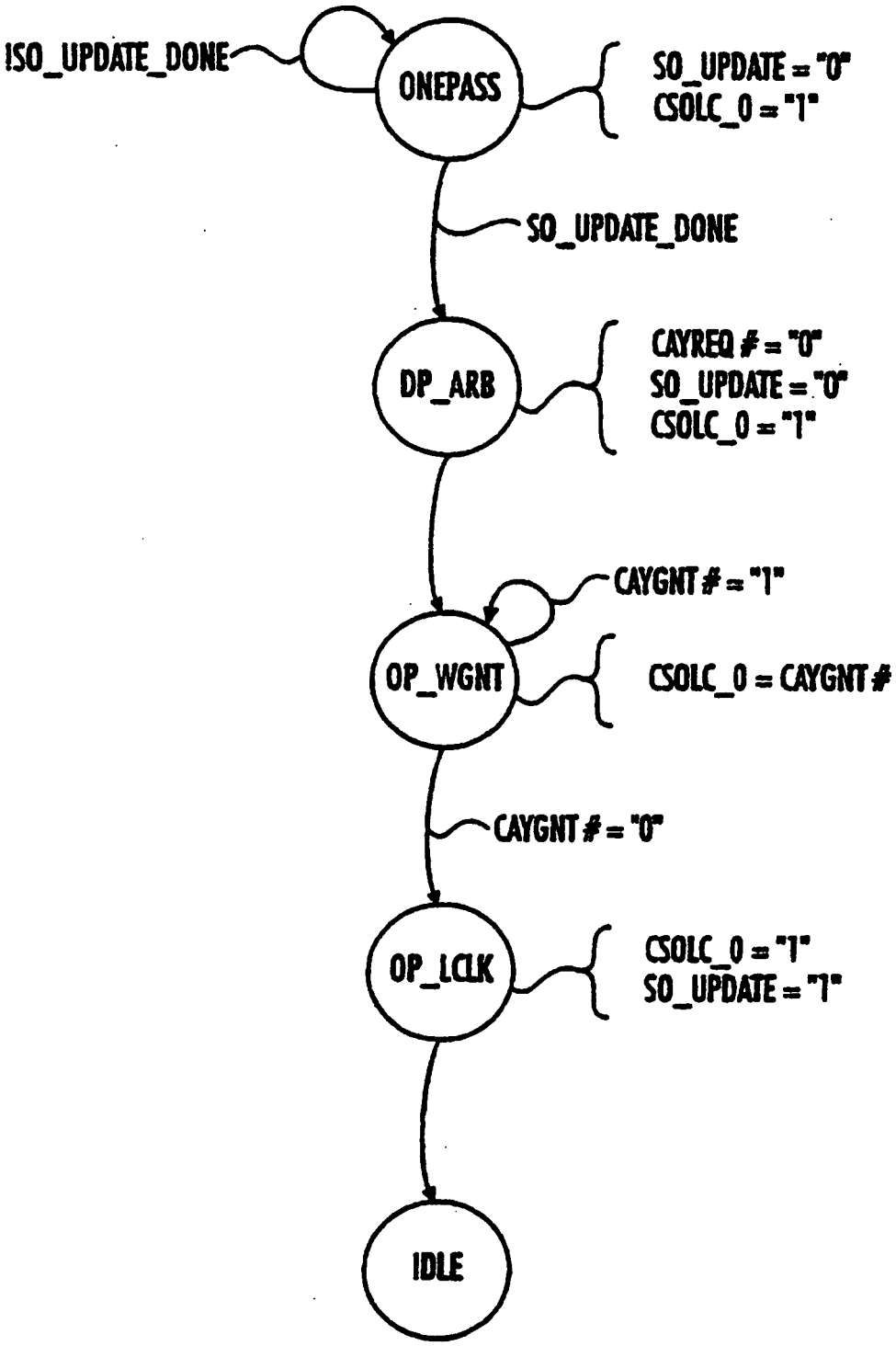

!SO_UPDATE_DONE
ONEPASS
SO_UPDATE = "0"
CSOLC_0 = "1"
SO_UPDATE_DONE
DP_ARB
CAYREQ # = "0"
SO_UPDATE = "0"
CSOLC_0 = "1"
CAYGNT # = "1"
OP_WGNT
CSOLC_0 = CAYGNT #
CAYGNT # = "0"
OP_LCLK
CSOLC_0 = "1"
SO_UPDATE = "1"
IDLE

FIG. 33H

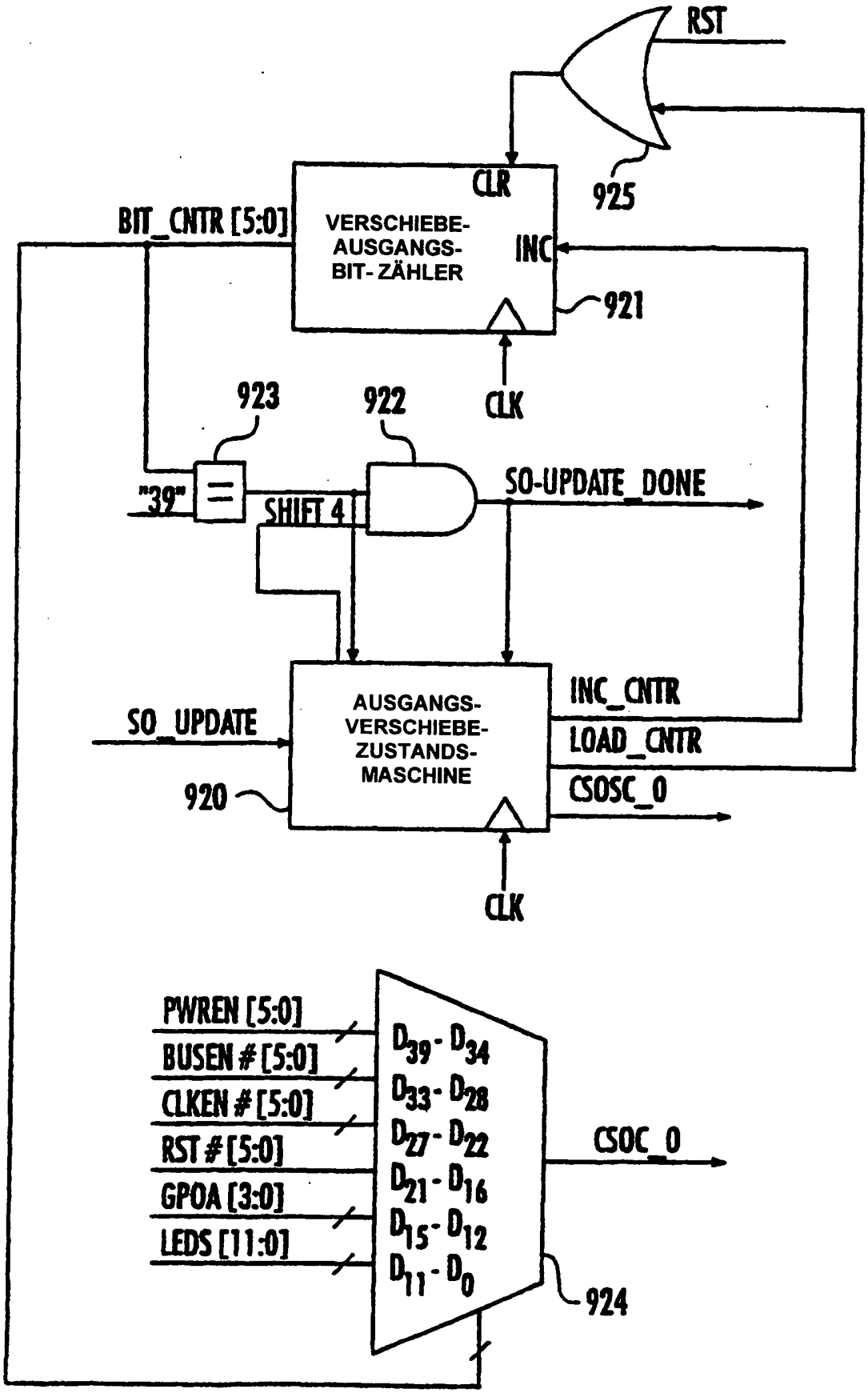

RST
925
CLR
BIT_CNTR [5:0]
VERSCHIEBE-
AUSGANGS-
BIT- ZÄHLER
INC
921
CLK
923
922
"39"
SHIFT 4
SO-UPDATE_DONE
AUSGANGS-
VERSCHIEBE-
ZUSTANDS-
MASCHINE
SO_UPDATE
920
INC_CNTR
LOAD_CNTR
CSOSC_0
CLK
PWREN [5:0]
BUSEN # [5:0]
CLKEN # [5:0]
RST # [5:0]
GPOA [3:0]
LEDS [11:0]
D39 - D34
D33 - D28
D27 - D22
D21 - D16
D15 - D12
D11 - D0
CSOC_0
924

FIG. 34

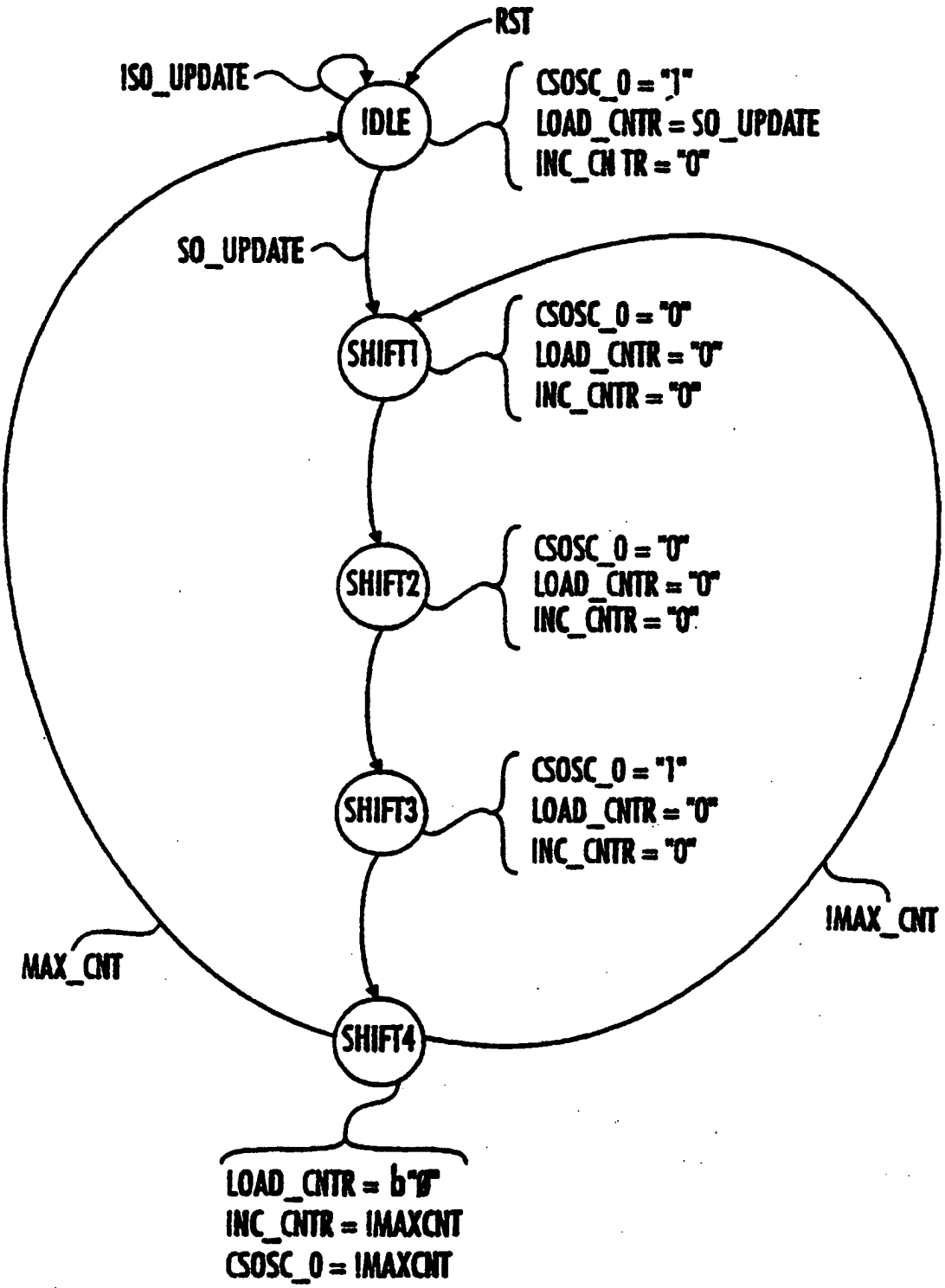

RST
!SO_UPDATE
IDLE
CSOSC_0 = "1"
LOAD_CNTR = SO_UPDATE
INC_CNTR = "0"
SO_UPDATE
SHIFT1
CSOSC_0 = "0"
LOAD_CNTR = "0"
INC_CNTR = "0"
SHIFT2
CSOSC_0 = "0"
LOAD_CNTR = "0"
INC_CNTR = "0"
SHIFT3
CSOSC_0 = "1"
LOAD_CNTR = "0"
INC_CNTR = "0"
!MAX_CNT
MAX_CNT
SHIFT4
LOAD_CNTR = b"0"
INC_CNTR = !MAXCNT
CSOSC_0 = !MAXCNT

FIG. 35A

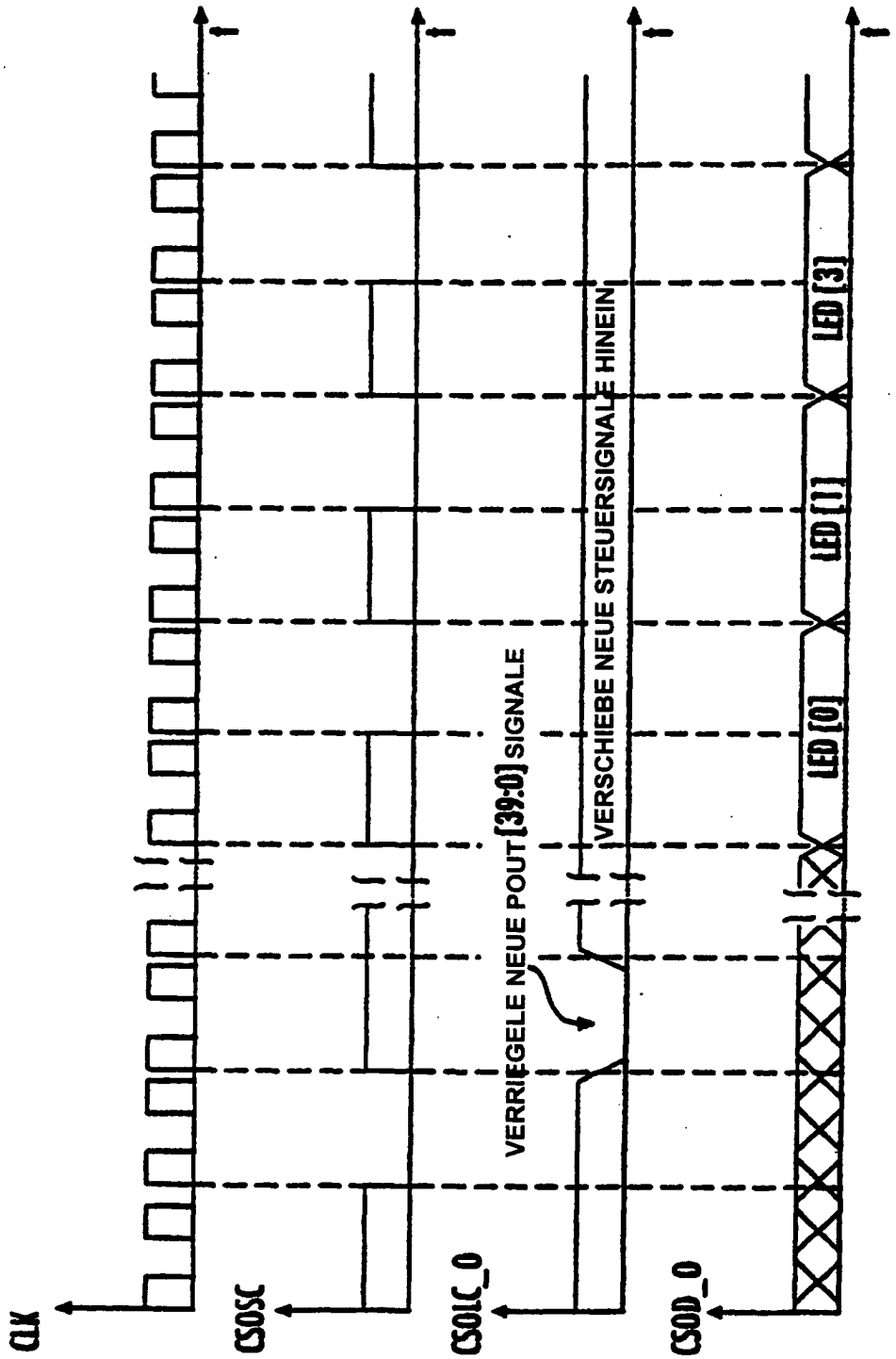

CLK
CSOSC
CSOLC_0
CSOD_0
VERRIEGELE NEUE POUT [39:0] SIGNALE
VERSCHIEBE NEUE STEUERSIGNALE HINEIN
LED [0]
LED [1]
LED [3]

FIG. 35B

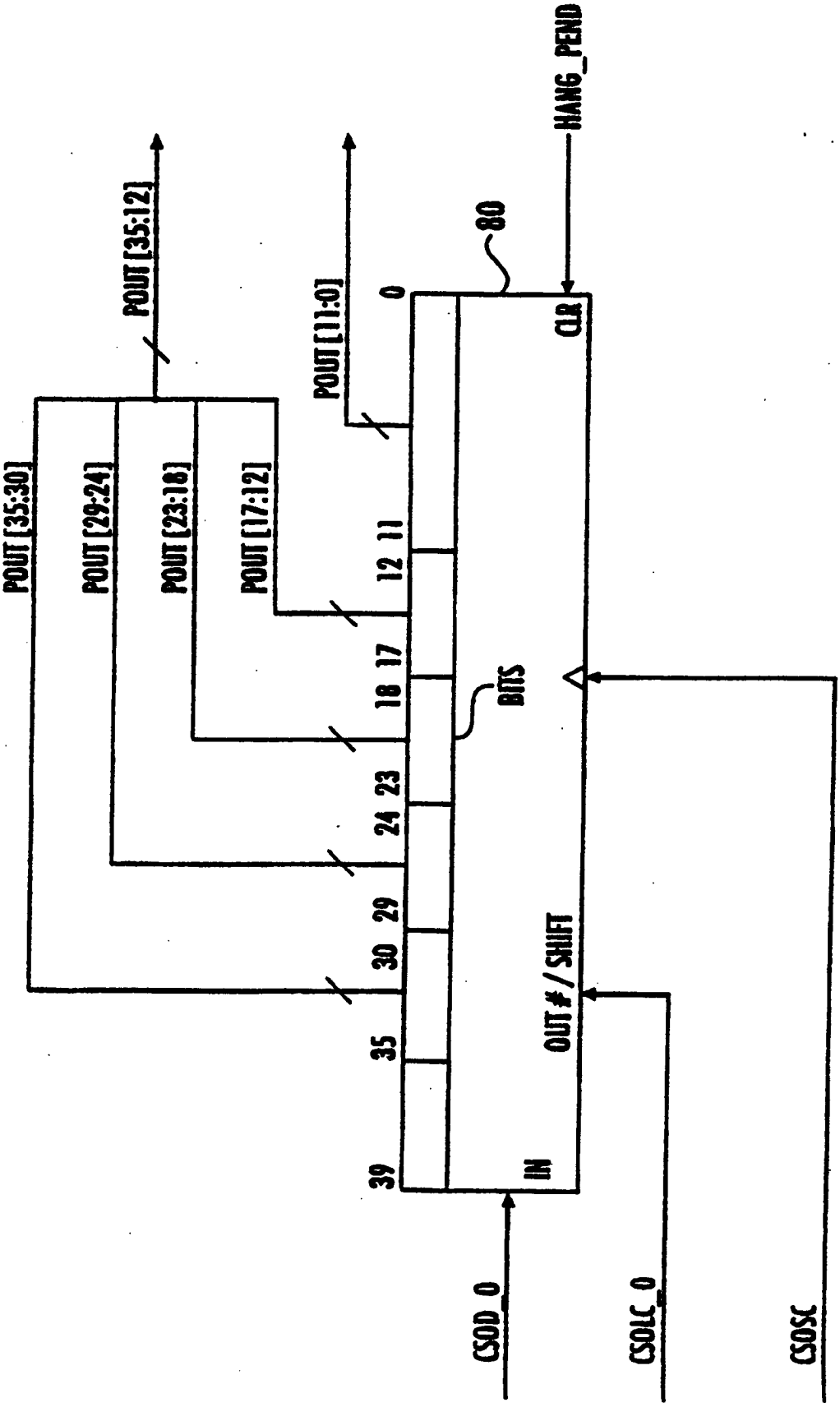

HANG_PEND
80
POUT [35:12]
POUT [11:0]
0
CLR
POUT [35:30]
POUT [29:24]
POUT [23:18]
POUT [17:12]
12 11
18 17
BITS
24 23
30 29
35
39
OUT # / SHIFT
IN
CSOD_0
CSOLC_0
CSOSC

FIG. 36

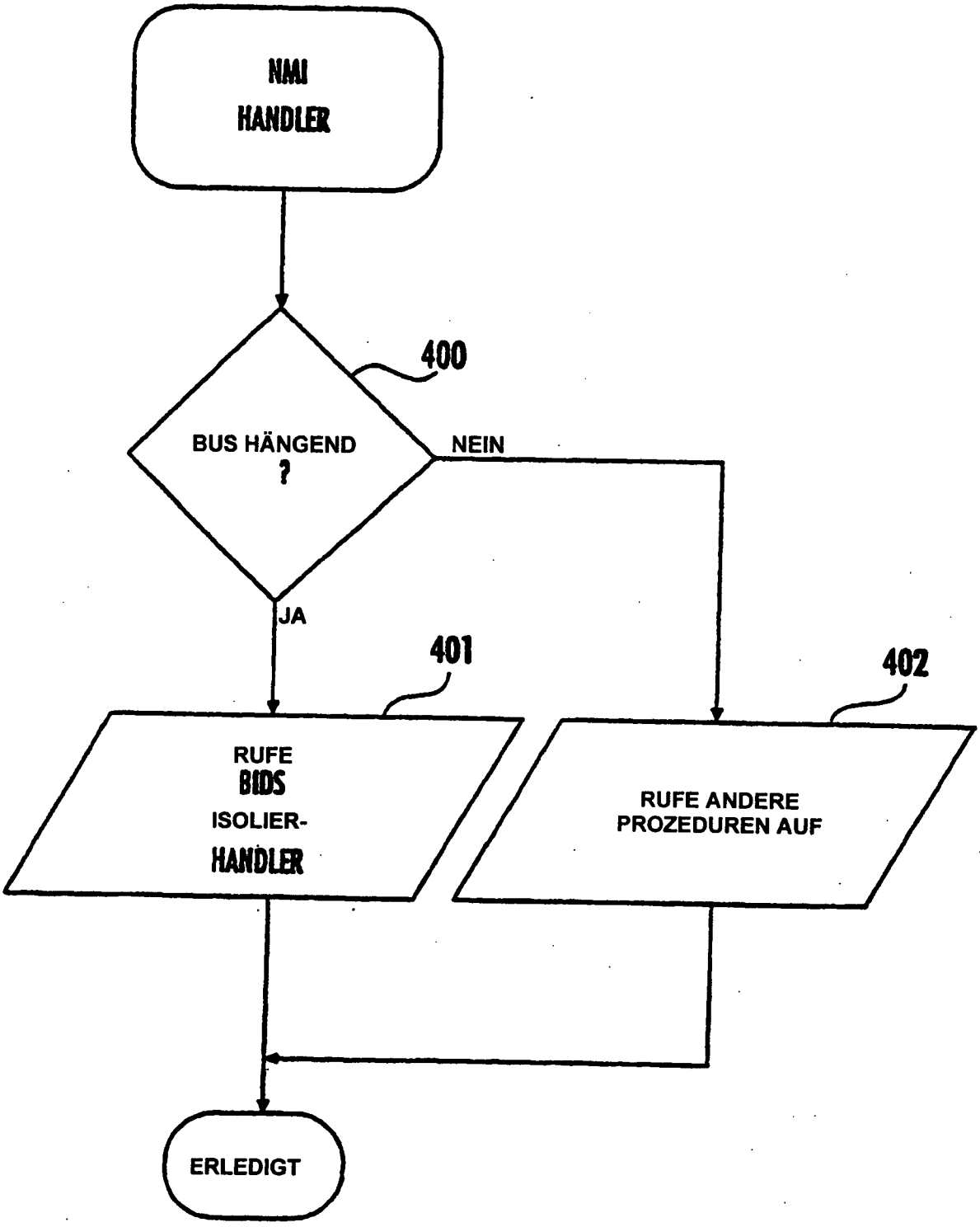

NMI
HANDLER
400
BUS HÄNGEND
?
NEIN
JA
401
402
RUFE
BIDS
ISOLIER-
HANDLER
RUFE ANDERE
PROZEDUREN AUF
ERLEDIGT

FIG. 37

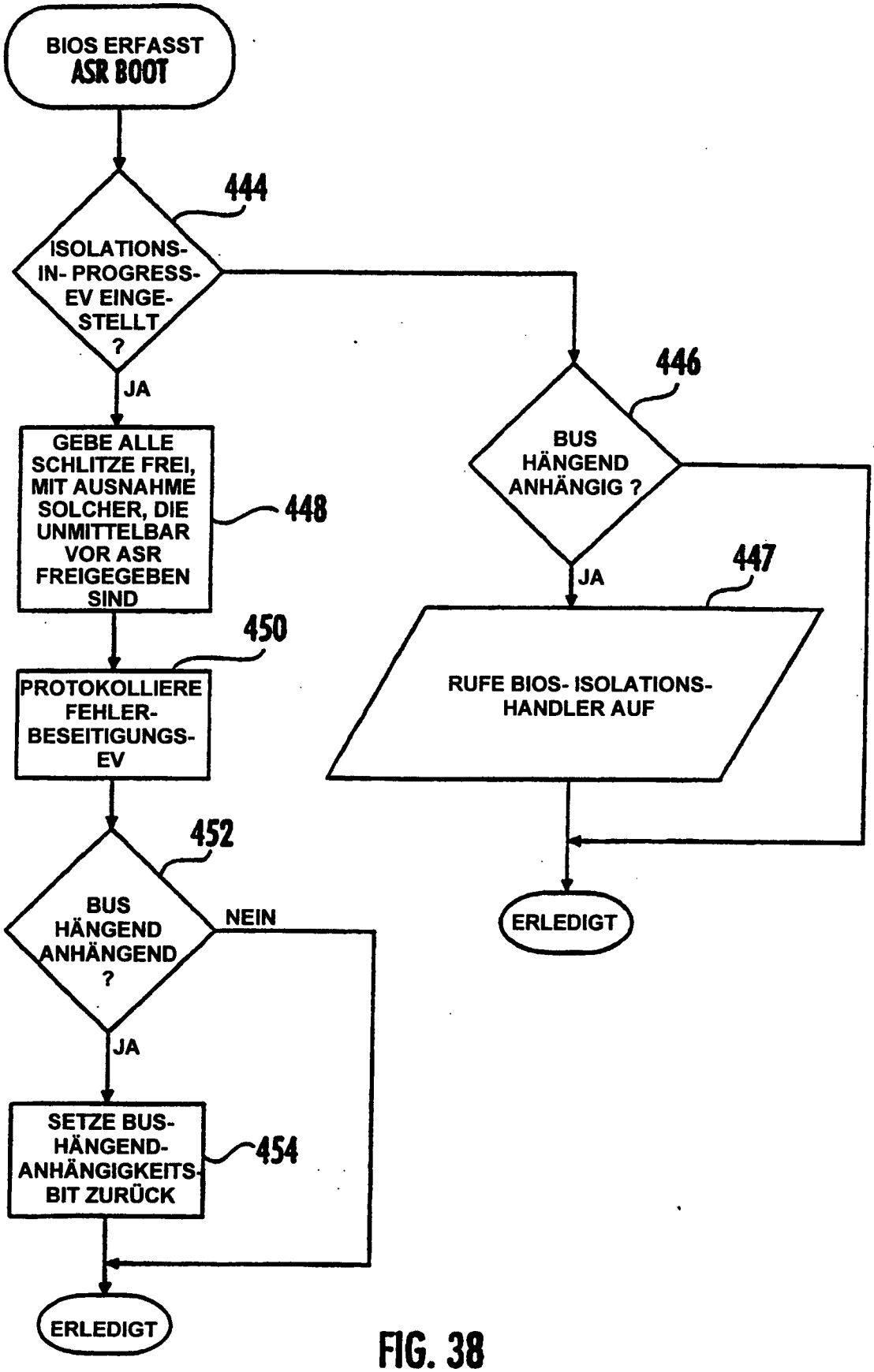

BIOS ERFASST ASR BOOT
444
ISOLATIONS-IN-PROGRESS-EV EINGE-STELLT ?
JA
GEBE ALLE SCHLITZE FREI, MIT AUSNAHME SOLCHER, DIE UNMITTELBAR VOR ASR FREIGEGEBEN SIND
448
450
PROTOKOLLIERE FEHLER-BESEITIGUNGS-EV
452
BUS HÄNGEND ANHÄNGEND ?
NEIN
JA
SETZE BUS-HÄNGEND-ANHÄNGIGKEITS-BIT ZURÜCK
454
ERLEDIGT
446
BUS HÄNGEND ANHÄNGIG ?
JA
447
RUFE BIOS-ISOLATIONS-HANDLER AUF
ERLEDIGT

FIG. 38

FIG. 39

FIG. 39A | FIG. 39B

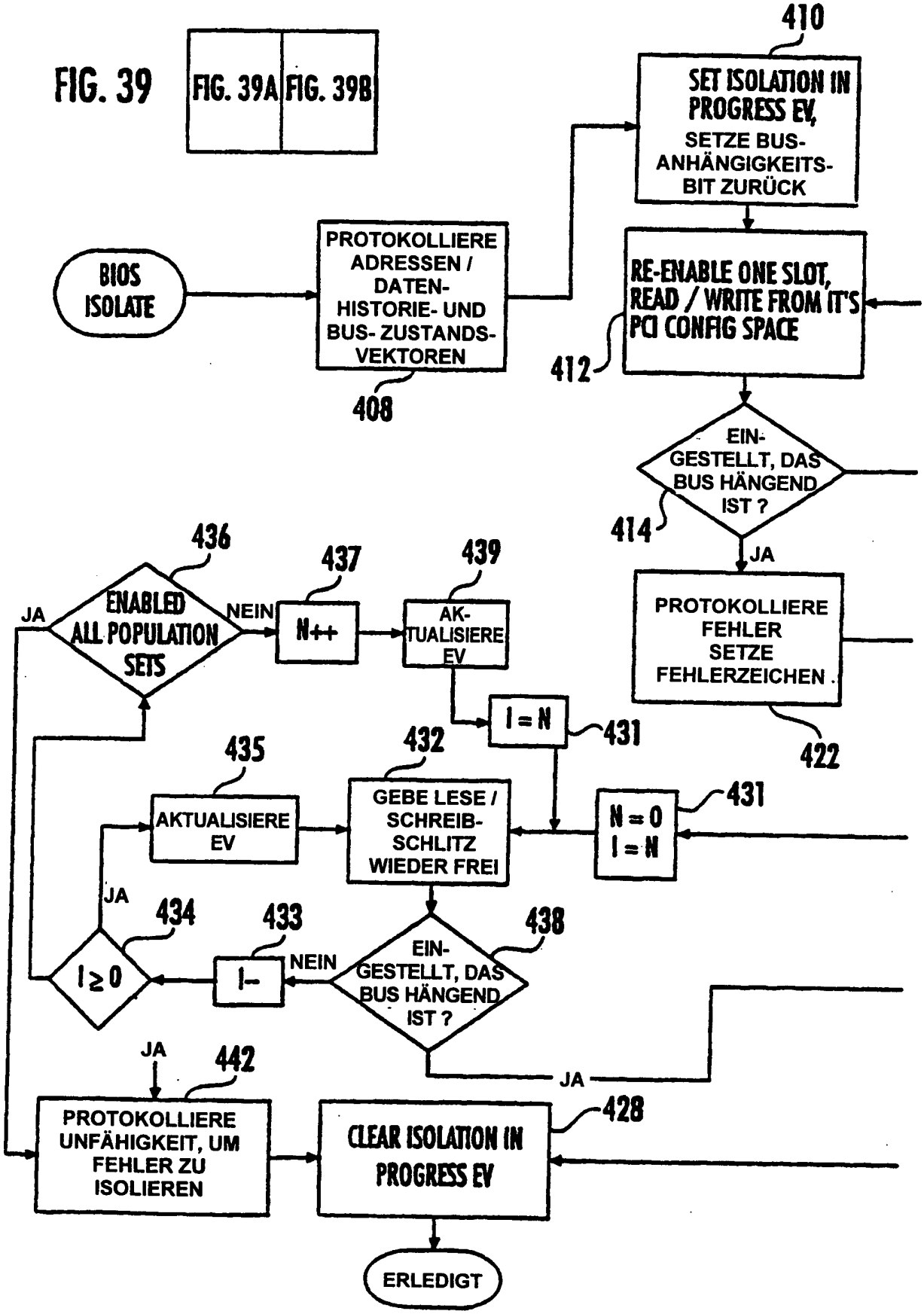


FIG. 39A

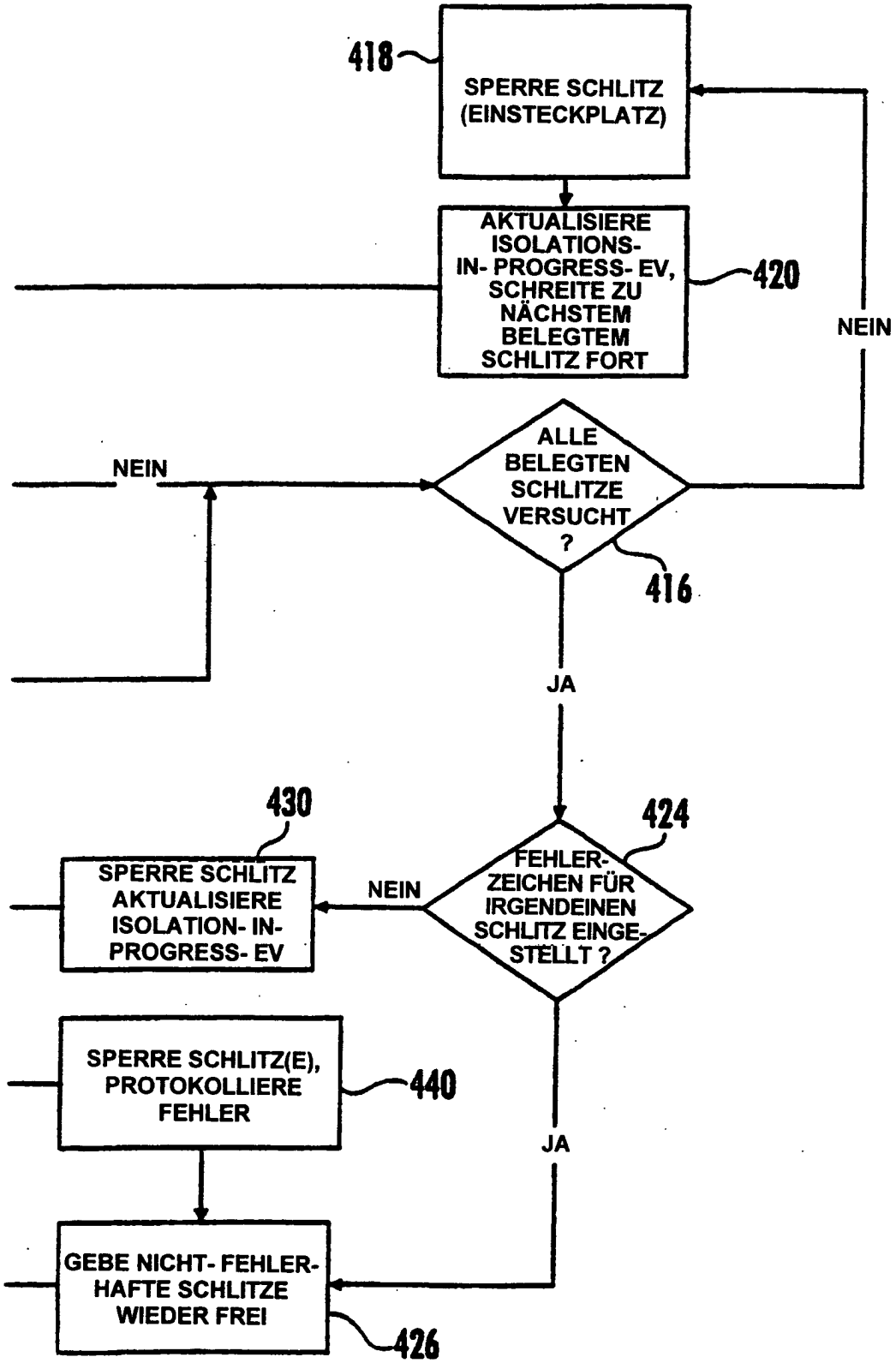

418
SPERRE SCHLITZ (EINSTECKPLATZ)
AKTUALISIERE ISOLATIONS- IN- PROGRESS- EV, SCHREITE ZU NÄCHSTEM BELEGTEM SCHLITZ FORT
420
NEIN
NEIN
ALLE BELEGTEN SCHLITZE VERSUCHT ?
416
JA
430
424
SPERRE SCHLITZ AKTUALISIERE ISOLATION- IN- PROGRESS- EV
NEIN
FEHLER- ZEICHEN FÜR IRGENDEINEN SCHLITZ EINGE- STELLT ?
SPERRE SCHLITZ(E), PROTOKOLLIERE FEHLER
440
JA
GEBE NICHT- FEHLER- HAFTE SCHLITZE WIEDER FREI
426

FIG. 39B

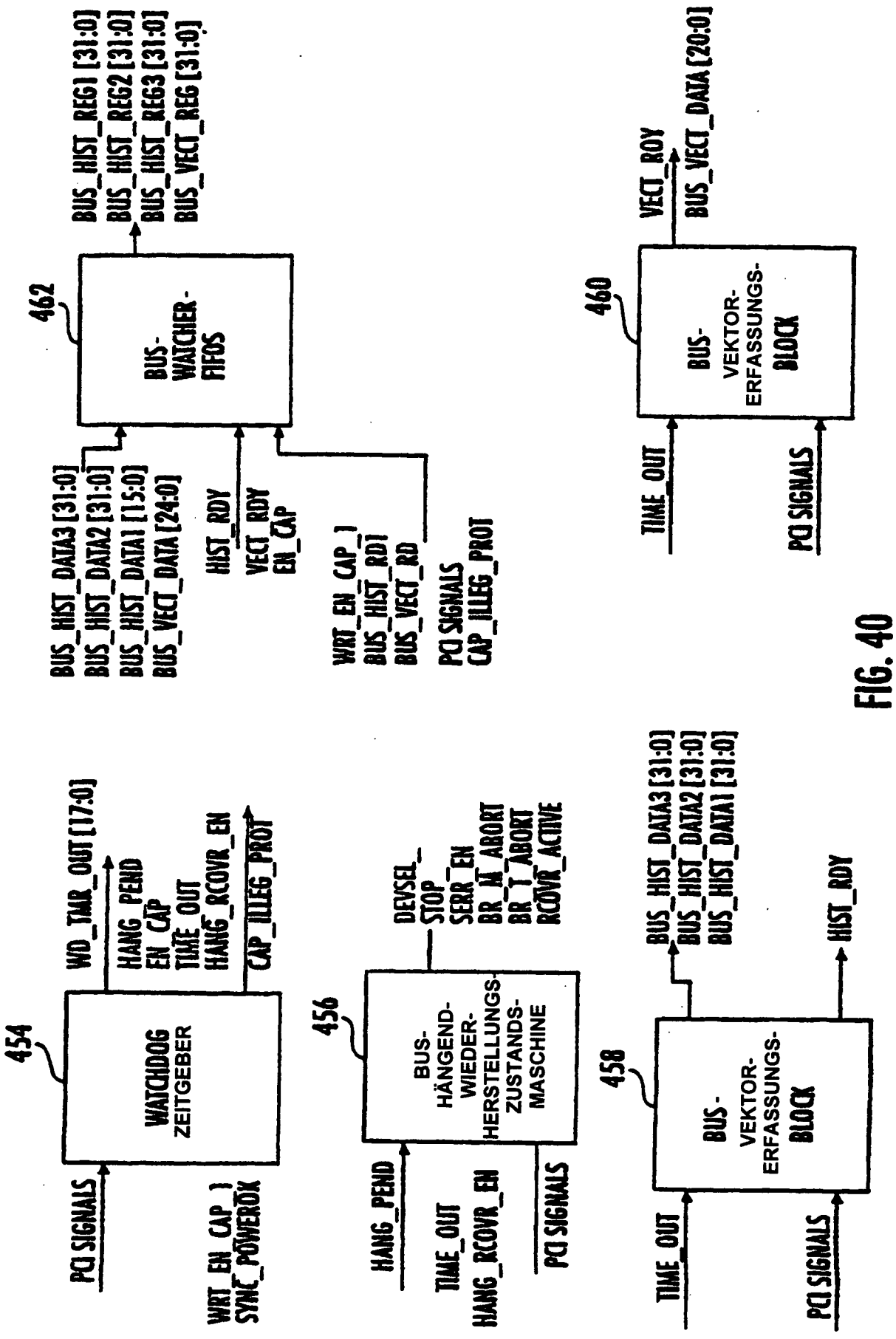
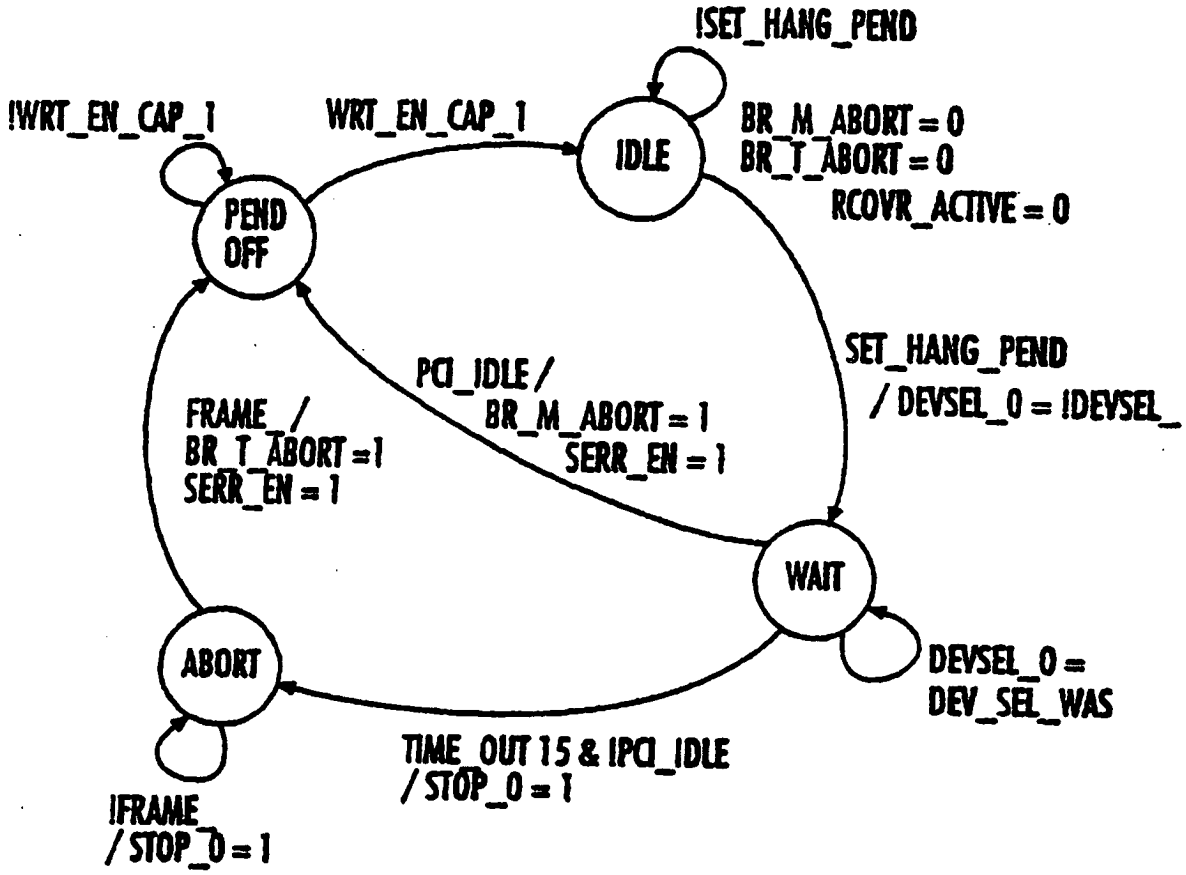

454
WATCHDOG ZEITGEBER
PCI SIGNALS
WRT_EN_CAP_) SYNC_POWERDX
WD_TMR_OUT [17:0]
HANG_PEND
EN_CAP
TIME_OUT
HANG_RCOVR_EN
CAP_ILLEG_PROT
456
BUS-HÄNGEND-WIEDER-HERSTELLUNGS-ZUSTANDS-MASCHINE
HANG_PEND
TIME_OUT
HANG_RCOVR_EN
PCI SIGNALS
DEVSEL_STOP
SERR_EN
BR_M_ABORT
BR_T_ABORT
RCOVR_ACTIVE
458
BUS-VEKTOR-ERFASSUNGS-BLOCK
TIME_OUT
PCI SIGNALS
BUS_HIST_DATA3 [31:0]
BUS_HIST_DATA2 [31:0]
BUS_HIST_DATA1 [31:0]
HIST_RDY
460
BUS-VEKTOR-ERFASSUNGS-BLOCK
TIME_OUT
PCI SIGNALS
VECT_RDY
BUS_VECT_DATA [20:0]
462
BUS-WATCHER-FIFOS
BUS_HIST_DATA3 [31:0]
BUS_HIST_DATA2 [31:0]
BUS_HIST_DATA1 [15:0]
BUS_VECT_DATA [24:0]
HIST_RDY
VECT_RDY
EN_CAP
WRT_EN_CAP_)
BUS_HIST_RDI
BUS_VECT_RD
PCI SIGNALS
CAP_ILLEG_PROT
BUS_HIST_REG1 [31:0]
BUS_HIST_REG2 [31:0]
BUS_HIST_REG3 [31:0]
BUS_VECT_REG [31:0]

FIG. 40


!SET_HANG_PEND
!WRT_EN_CAP_1
WRT_EN_CAP_1
IDLE
BR_M_ABORT = 0
BR_T_ABORT = 0
RCOVR_ACTIVE = 0
PEND
OFF
SET_HANG_PEND
/ DEVSEL_0 = !DEVSEL_
PCI_IDLE /
BR_M_ABORT = 1
SERR_EN = 1
FRAME_ /
BR_T_ABORT = 1
SERR_EN = 1
WAIT
ABORT
DEVSEL_0 =
DEV_SEL_WAS
TIME_OUT 15 & !PCI_IDLE
/ STOP_0 = 1
!FRAME_
/ STOP_0 = 1

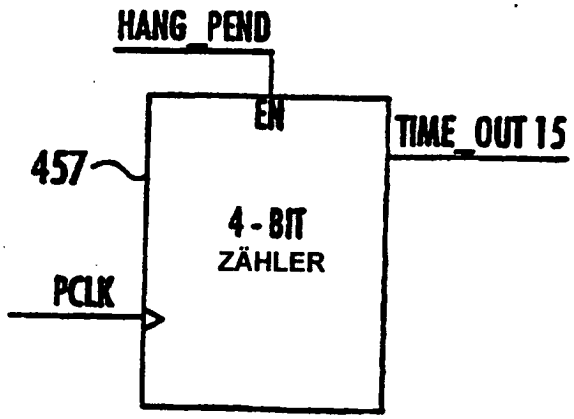

HANG_PEND
EN
TIME_OUT 15
457
4 - BIT
ZÄHLER
PCLK

FIG. 41

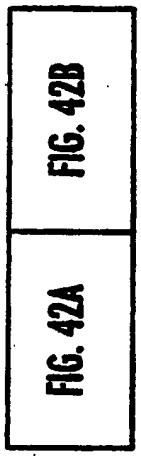


FIG. 42

FIG. 42A

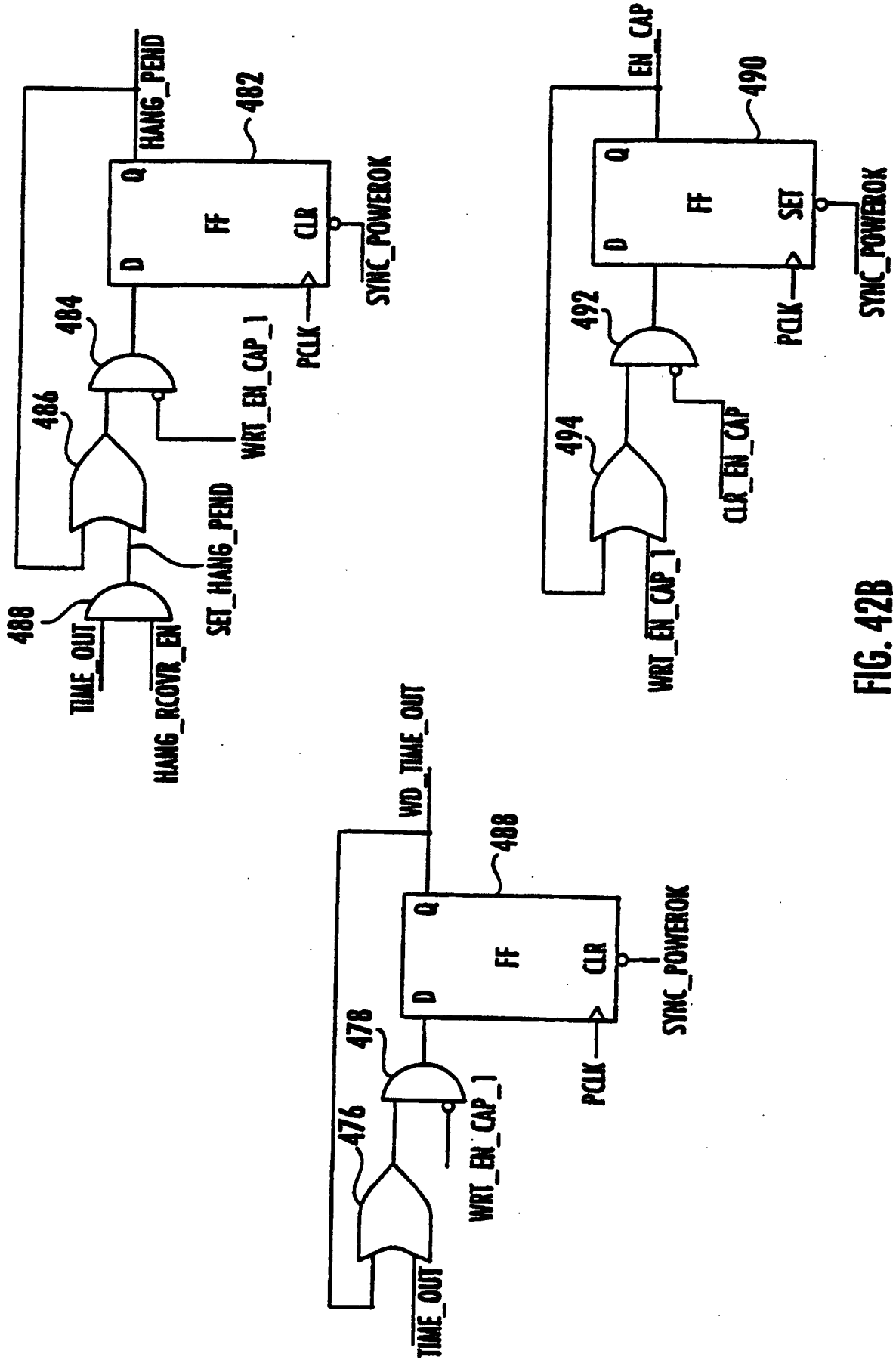

HANG_PEND
482
Q
D
FF
CLR
SYNC_POWEROK
PCLK
484
WRT_EN_CAP_1
486
SET_HANG_PEND
488
TIME_OUT
HANG_RCOVR_EN
EN_CAP
490
Q
D
FF
SET
SYNC_POWEROK
PCLK
492
CLR_EN_CAP
494
WRT_EN_CAP_1
WD_TIME_OUT
488
Q
D
FF
CLR
SYNC_POWEROK
PCLK
478
WRT_EN_CAP_1
476
TIME_OUT

FIG. 42B

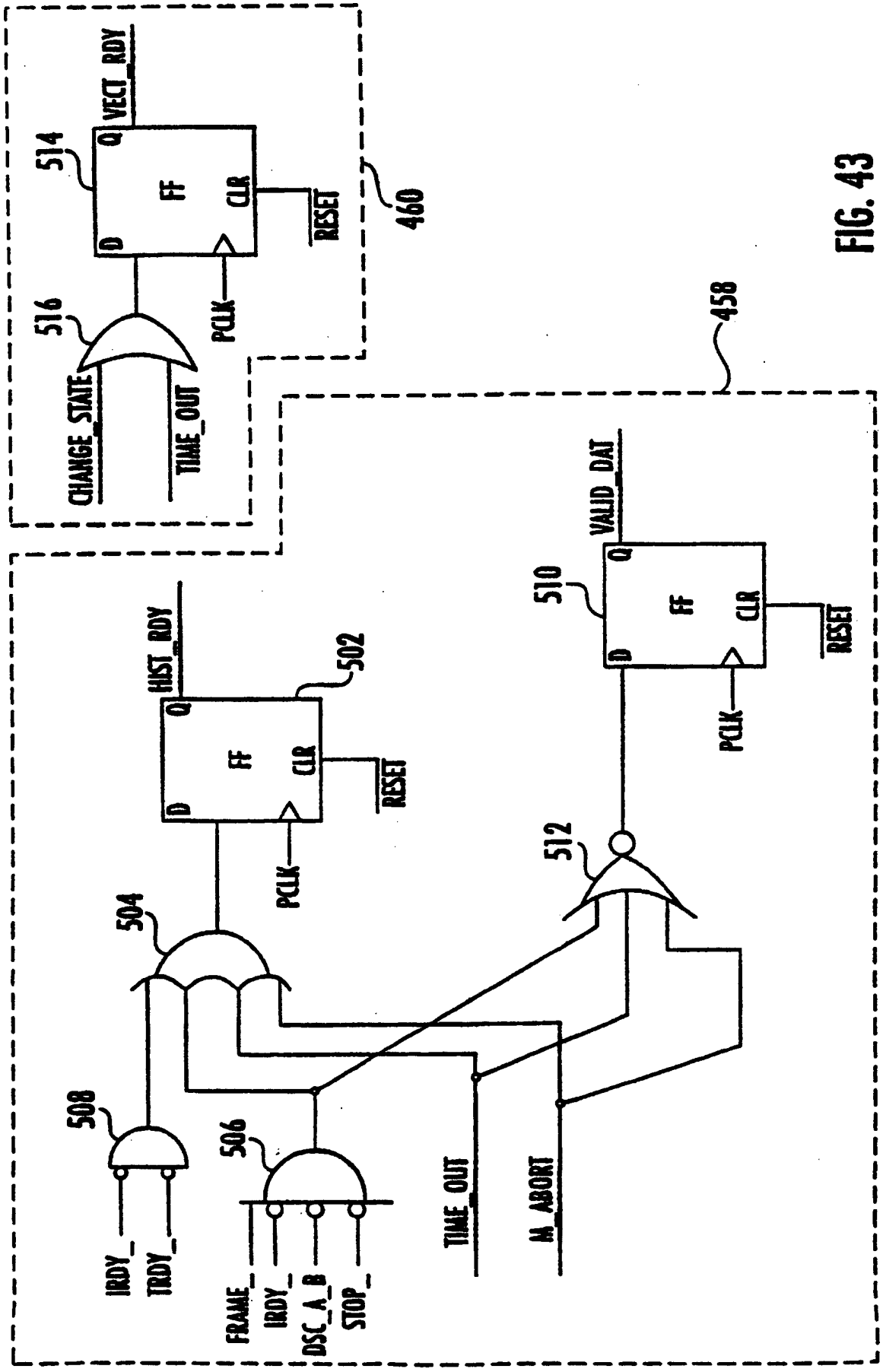

VECT_RDY
Q
514
FF
D
CLR
RESET
460
PCLK
516
CHANGE_STATE
TIME_OUT
458
VALID_DAT
510
HUST_RDY
502
504
512
508
506
IRDY_
TRDY_
FRAME_
IRDY_
DSC_A_B
STOP_
TIME_OUT
M_ABORT

FIG. 43

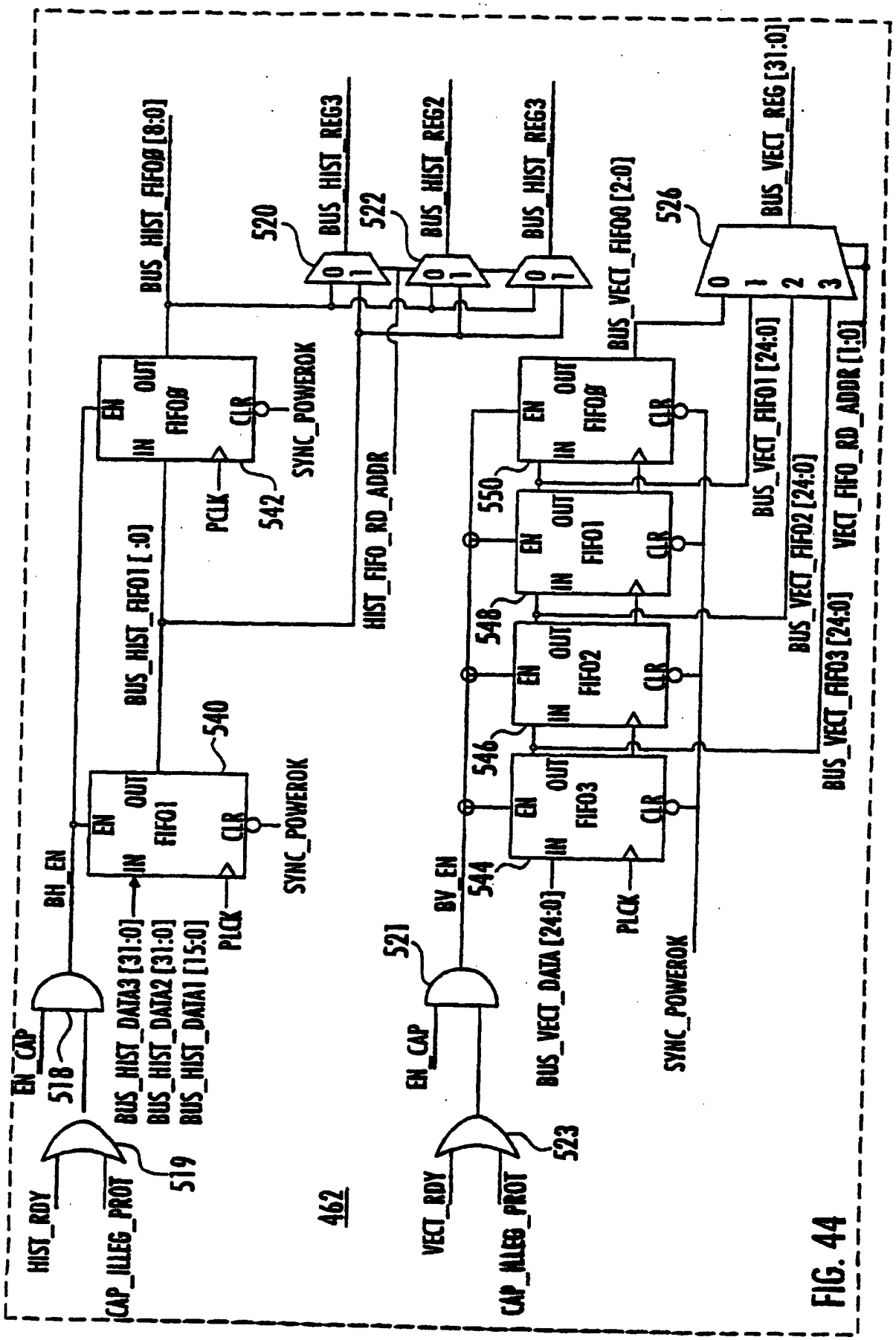

462
HIST_RDY
CAP_ILLEG_PROT
519
EN_CAP
518
BH_EN
BUS_HIST_DATA3 [31:0]
BUS_HIST_DATA2 [31:0]
BUS_HIST_DATA1 [15:0]
PCLK
SYNC_POWEROK
540
EN
OUT
IN
FIFO1
CLR
BUS_HIST_FIFO1 [:0]
PCLK
542
SYNC_POWEROK
EN
OUT
IN
FIFO0
CLR
BUS_HIST_FIFO0 [8:0]
HIST_FIFO_RD_ADDR
520
522
BUS_HIST_REG3
BUS_HIST_REG2
BUS_HIST_REG3
0
1
VECT_RDY
CAP_ILLEG_PROT
523
521
EN_CAP
BV_EN
544
546
548
550
EN
OUT
IN
FIFO3
FIFO2
FIFO1
FIFO0
CLR
BUS_VECT_DATA [24:0]
PCLK
SYNC_POWEROK
BUS_VECT_FIFO0 [2:0]
BUS_VECT_FIFO1 [24:0]
BUS_VECT_FIFO2 [24:0]
BUS_VECT_FIFO3 [24:0]
526
0
1
2
3
BUS_VECT_REG [31:0]
VECT_FIFO_RD_ADDR [1:0]

FIG. 44

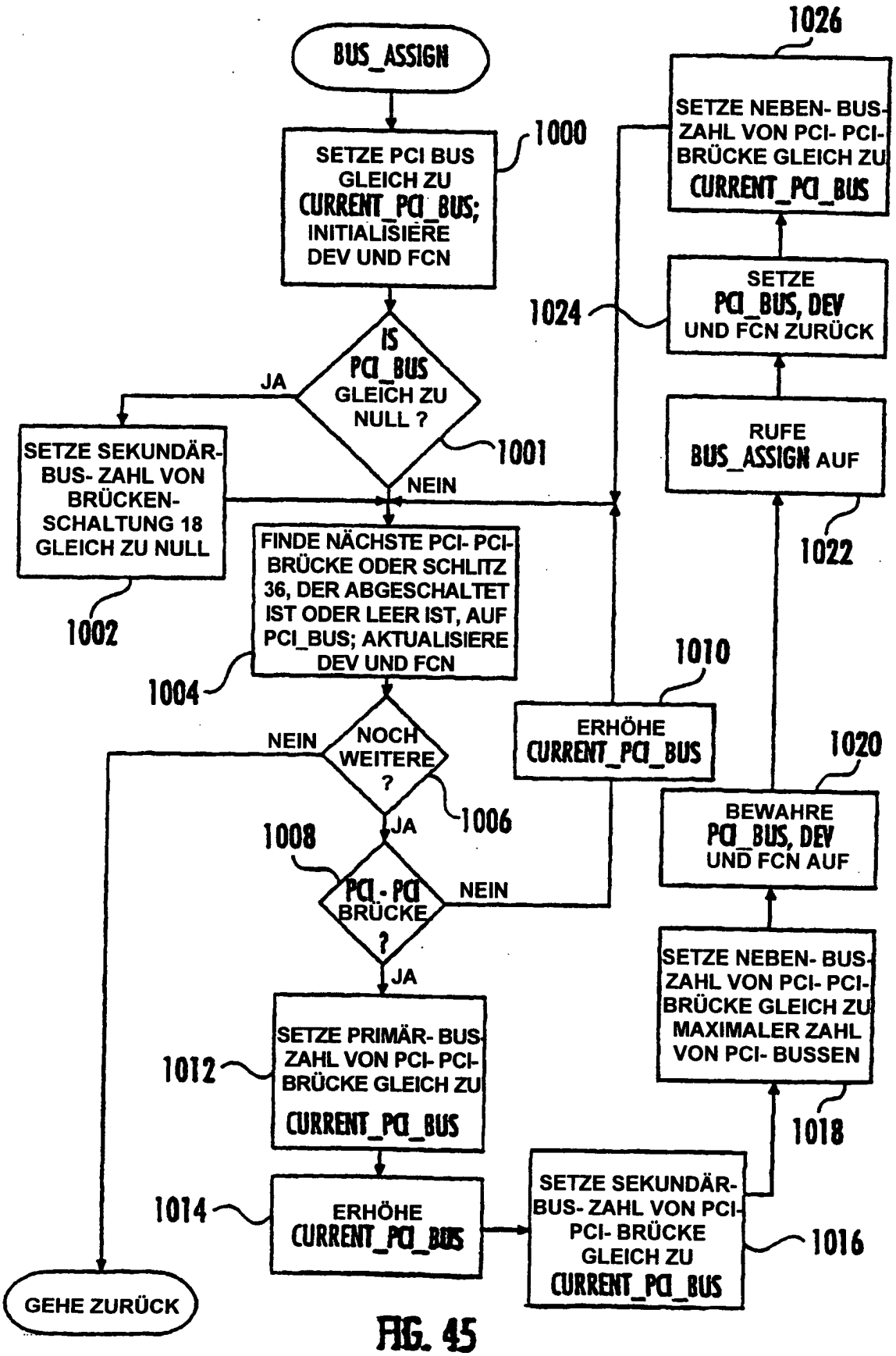

BUS_ASSIGN
SETZE PCI BUS GLEICH ZU CURRENT_PCI_BUS; INITIALISIERE DEV UND FCN
1000
IS PCI_BUS GLEICH ZU NULL ?
1001
JA
NEIN
SETZE SEKUNDÄR-BUS- ZAHL VON BRÜCKEN-SCHALTUNG 18 GLEICH ZU NULL
1002
FINDE NÄCHSTE PCI- PCI-BRÜCKE ODER SCHLITZ 36, DER ABGESCHALTET IST ODER LEER IST, AUF PCI_BUS; AKTUALISIERE DEV UND FCN
1004
NOCH WEITERE ?
1006
NEIN
JA
1008
PCI - PCI BRÜCKE ?
NEIN
JA
SETZE PRIMÄR- BUS-ZAHL VON PCI- PCI-BRÜCKE GLEICH ZU CURRENT_PCI_BUS
1012
ERHÖHE CURRENT_PCI_BUS
1014
SETZE SEKUNDÄR-BUS- ZAHL VON PCI-PCI- BRÜCKE GLEICH ZU CURRENT_PCI_BUS
1016
GEHE ZURÜCK
ERHÖHE CURRENT_PCI_BUS
1010
SETZE NEBEN- BUS-ZAHL VON PCI- PCI-BRÜCKE GLEICH ZU MAXIMALER ZAHL VON PCI- BUSSEN
1018
BEWAHRE PCI_BUS, DEV UND FCN AUF
1020
RUFE BUS_ASSIGN AUF
1022
SETZE PCI_BUS, DEV UND FCN ZURÜCK
1024
SETZE NEBEN- BUS-ZAHL VON PCI- PCI-BRÜCKE GLEICH ZU CURRENT_PCI_BUS
1026

FIG. 45

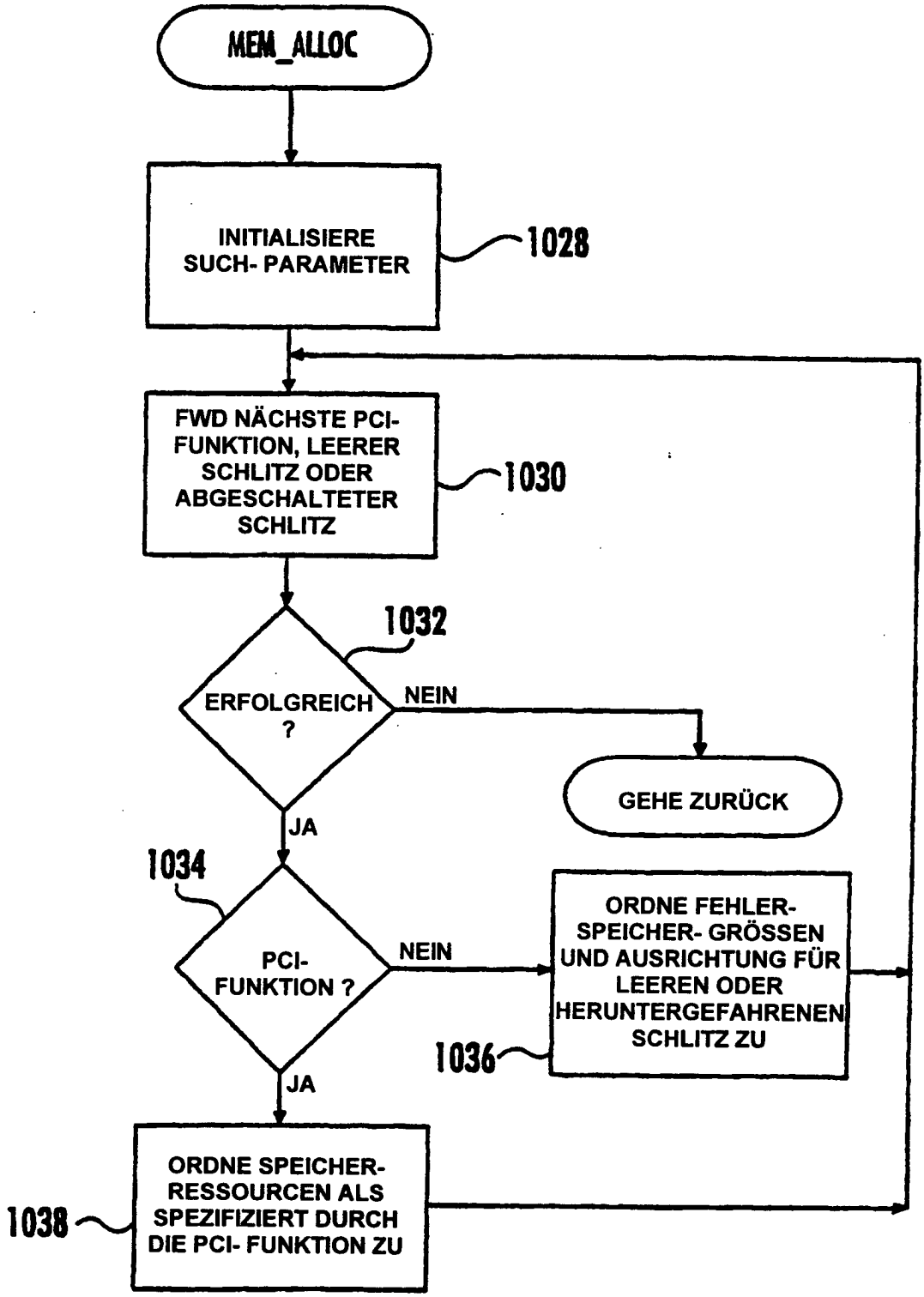

MEM_ALLOC
INITIALISIERE SUCH- PARAMETER
1028
FWD NÄCHSTE PCI-FUNKTION, LEERER SCHLITZ ODER ABGESCHALTETER SCHLITZ
1030
1032
ERFOLGREICH ?
NEIN
GEHE ZURÜCK
JA
1034
PCI-FUNKTION ?
NEIN
ORDNE FEHLER-SPEICHER- GRÖSSEN UND AUSRICHTUNG FÜR LEEREN ODER HERUNTERGEFAHRENEN SCHLITZ ZU
1036
JA
ORDNE SPEICHER-RESSOURCEN ALS SPEZIFIZIERT DURCH DIE PCI- FUNKTION ZU
1038

FIG. 46

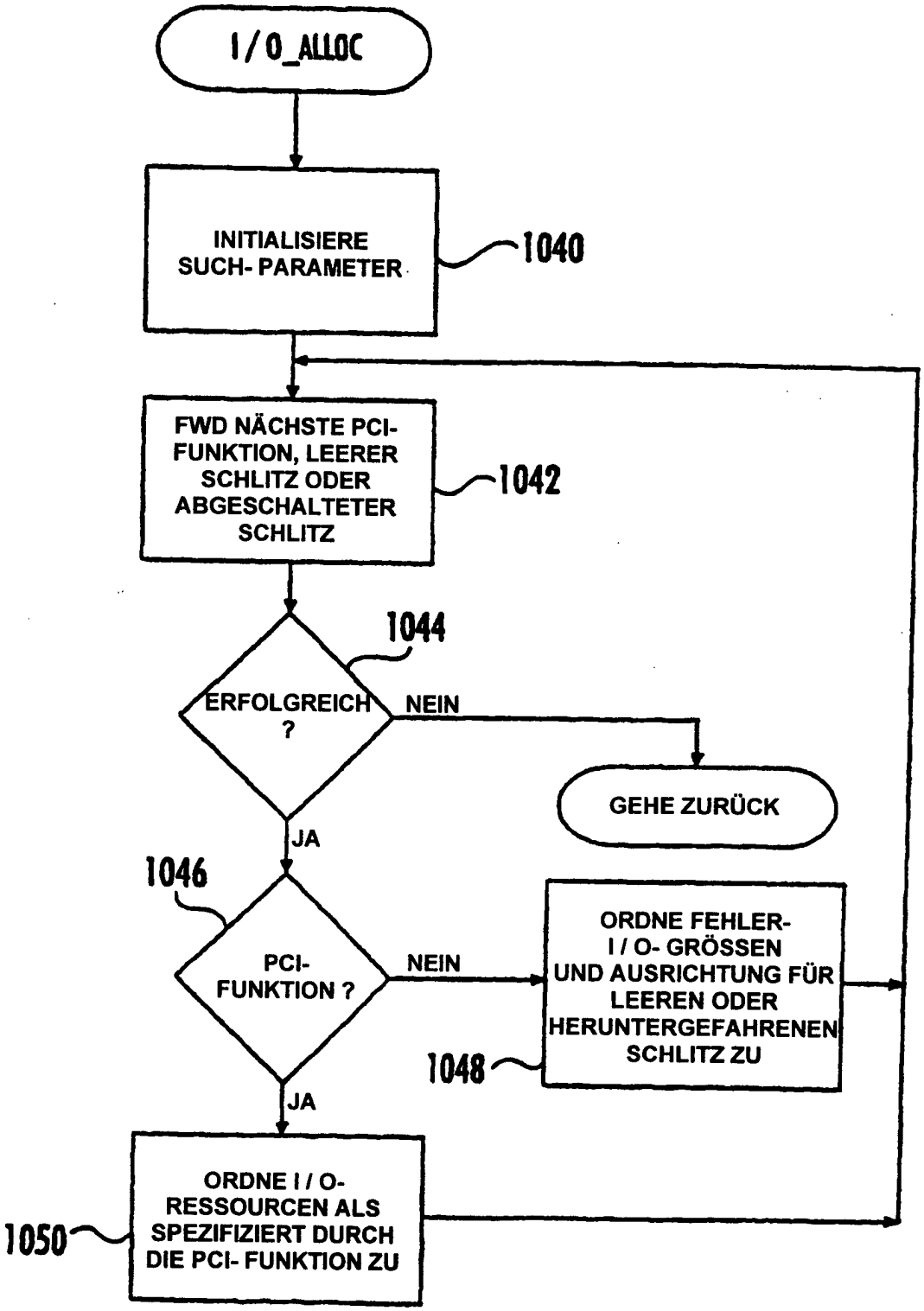

I/O_ALLOC
INITIALISIERE SUCH- PARAMETER
1040
FWD NÄCHSTE PCI-FUNKTION, LEERER SCHLITZ ODER ABGESCHALTETER SCHLITZ
1042
1044
ERFOLGREICH ?
NEIN
GEHE ZURÜCK
JA
1046
PCI-FUNKTION ?
NEIN
ORDNE FEHLER-I/O- GRÖSSEN UND AUSRICHTUNG FÜR LEEREN ODER HERUNTERGEFAHRENEN SCHLITZ ZU
1048
JA
ORDNE I/O-RESSOURCEN ALS SPEZIFIZIERT DURCH DIE PCI- FUNKTION ZU
1050

FIG. 47

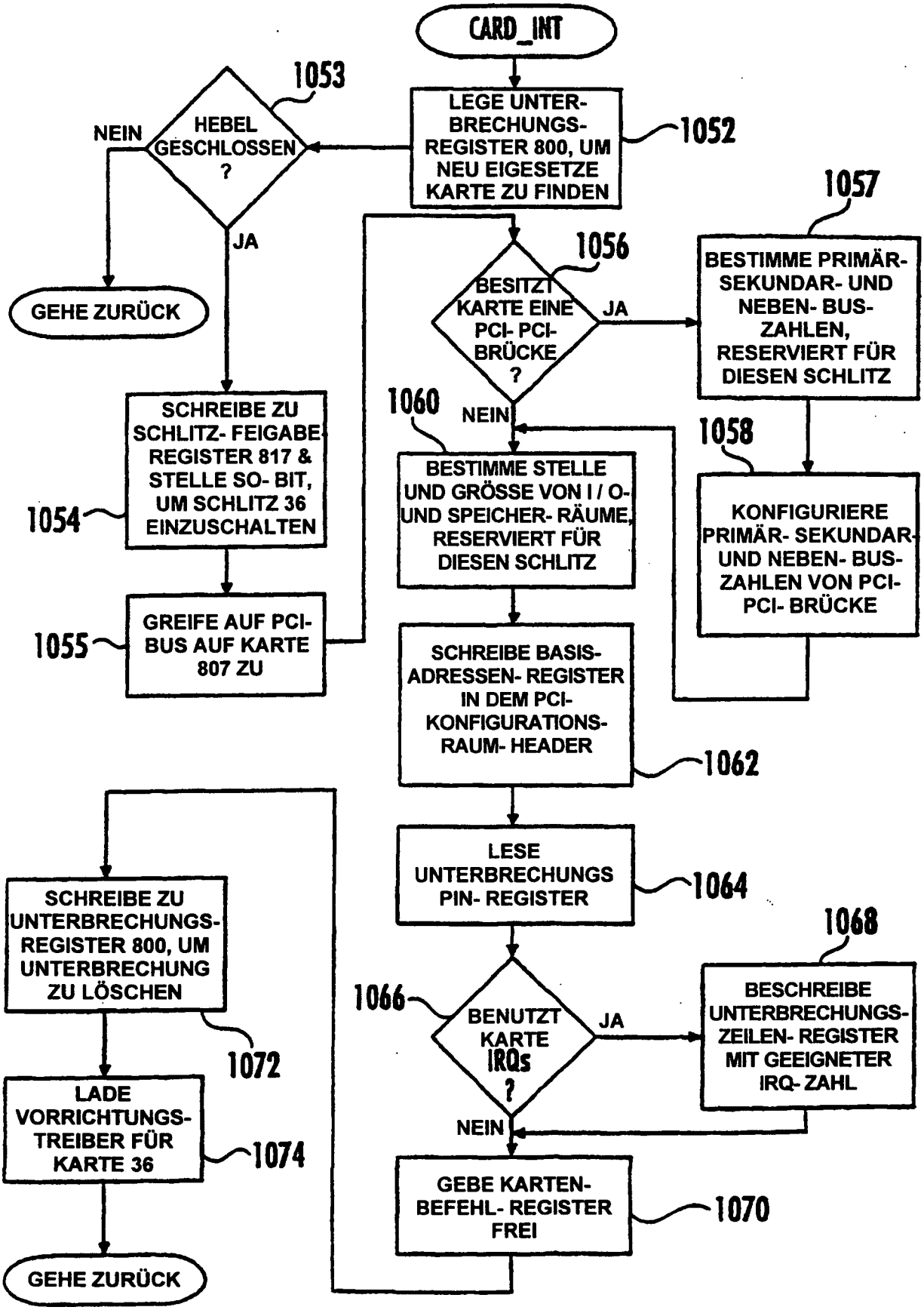

CARD_INT
LEGE UNTER-BRECHUNGS-REGISTER 800, UM NEU EIGESETZE KARTE ZU FINDEN
1052
HEBEL GESCHLOSSEN ?
1053
NEIN
JA
GEHE ZURÜCK
BESITZT KARTE EINE PCI- PCI-BRÜCKE ?
1056
JA
BESTIMME PRIMÄR-SEKUNDAR- UND NEBEN- BUS-ZAHLEN, RESERVIERT FÜR DIESEN SCHLITZ
1057
NEIN
1060
BESTIMME STELLE UND GRÖSSE VON I / O- UND SPEICHER- RÄUME, RESERVIERT FÜR DIESEN SCHLITZ
KONFIGURIERE PRIMÄR- SEKUNDAR- UND NEBEN- BUS-ZAHLEN VON PCI-PCI- BRÜCKE
1058
SCHREIBE ZU SCHLITZ- FEIGABE-REGISTER 817 & STELLE SO- BIT, UM SCHLITZ 36 EINZUSCHALTEN
1054
GREIFE AUF PCI-BUS AUF KARTE 807 ZU
1055
SCHREIBE BASIS-ADRESSEN- REGISTER IN DEM PCI-KONFIGURATIONS-RAUM- HEADER
1062
LESE UNTERBRECHUNGS PIN- REGISTER
1064
BENUTZT KARTE IRQs ?
1066
JA
BESCHREIBE UNTERBRECHUNGS-ZEILEN- REGISTER MIT GEEIGNETER IRQ- ZAHL
1068
NEIN
GEBE KARTEN-BEFEHL- REGISTER FREI
1070
SCHREIBE ZU UNTERBRECHUNGS-REGISTER 800, UM UNTERBRECHUNG ZU LÖSCHEN
1072
LADE VORRICHTUNGS-TREIBER FÜR KARTE 36
1074
GEHE ZURÜCK

FIG. 48

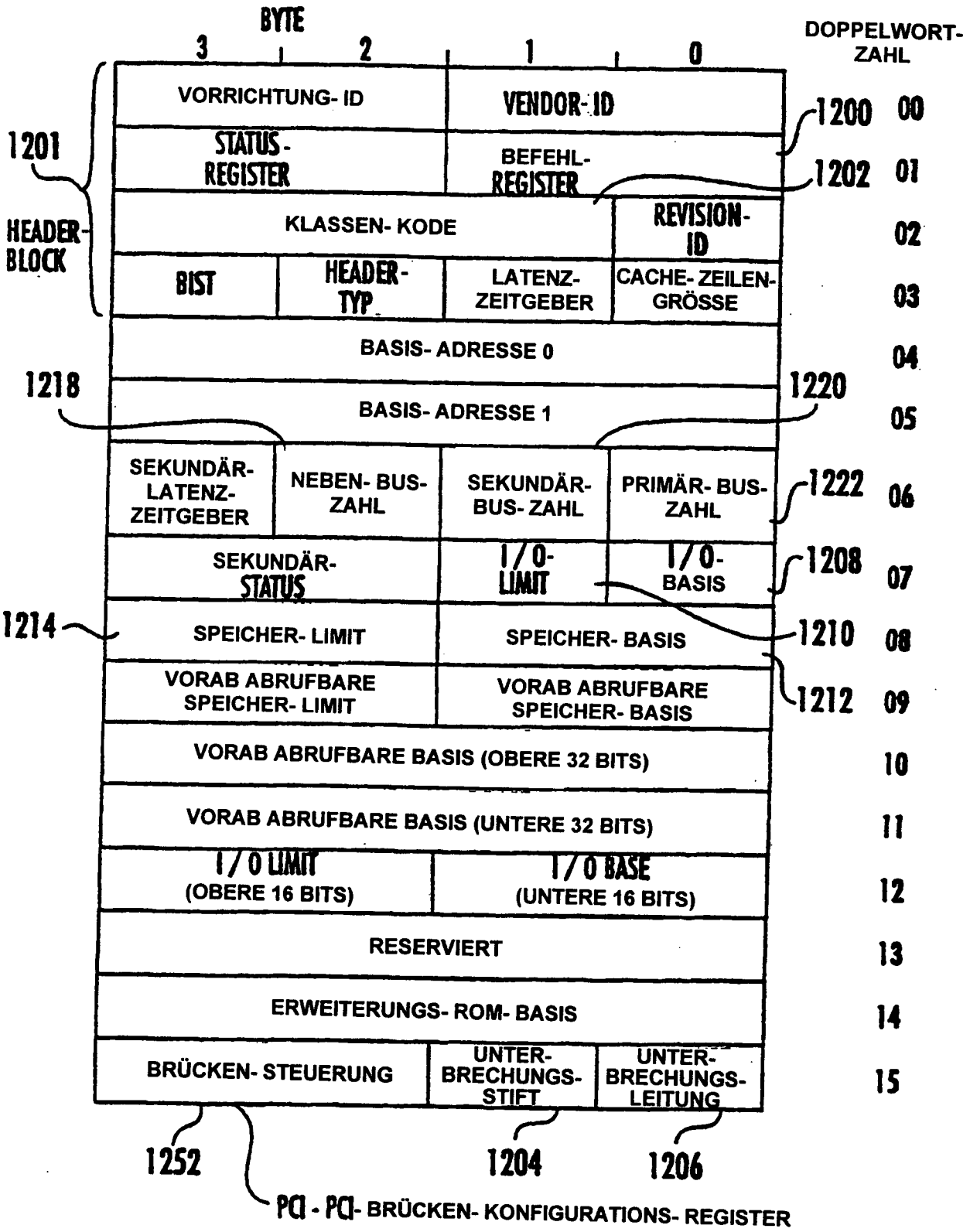

BYTE
3
2
1
0
DOPPELWORT-ZAHL
VORRICHTUNG- ID
VENDOR- ID
1200
00
1201
STATUS- REGISTER
BEFEHL- REGISTER
1202
01
HEADER- BLOCK
KLASSEN- KODE
REVISION- ID
02
BIST
HEADER- TYP
LATENZ- ZEITGEBER
CACHE- ZEILEN- GRÖSSE
03
BASIS- ADRESSE 0
04
1218
1220
BASIS- ADRESSE 1
05
SEKUNDÄR- LATENZ- ZEITGEBER
NEBEN- BUS- ZAHL
SEKUNDÄR- BUS- ZAHL
PRIMÄR- BUS- ZAHL
1222
06
SEKUNDÄR- STATUS
I/O- LIMIT
I/O- BASIS
1208
07
1214
SPEICHER- LIMIT
SPEICHER- BASIS
1210
08
VORAB ABRUFBARE SPEICHER- LIMIT
VORAB ABRUFBARE SPEICHER- BASIS
1212
09
VORAB ABRUFBARE BASIS (OBERE 32 BITS)
10
VORAB ABRUFBARE BASIS (UNTERE 32 BITS)
11
I/O LIMIT (OBERE 16 BITS)
I/O BASE (UNTERE 16 BITS)
12
RESERVIERT
13
ERWEITERUNGS- ROM- BASIS
14
BRÜCKEN- STEUERUNG
UNTER- BRECHUNGS- STIFT
UNTER- BRECHUNGS- LEITUNG
15
1252
1204
1206
PCI - PCI- BRÜCKEN- KONFIGURATIONS- REGISTER

FIG. 49

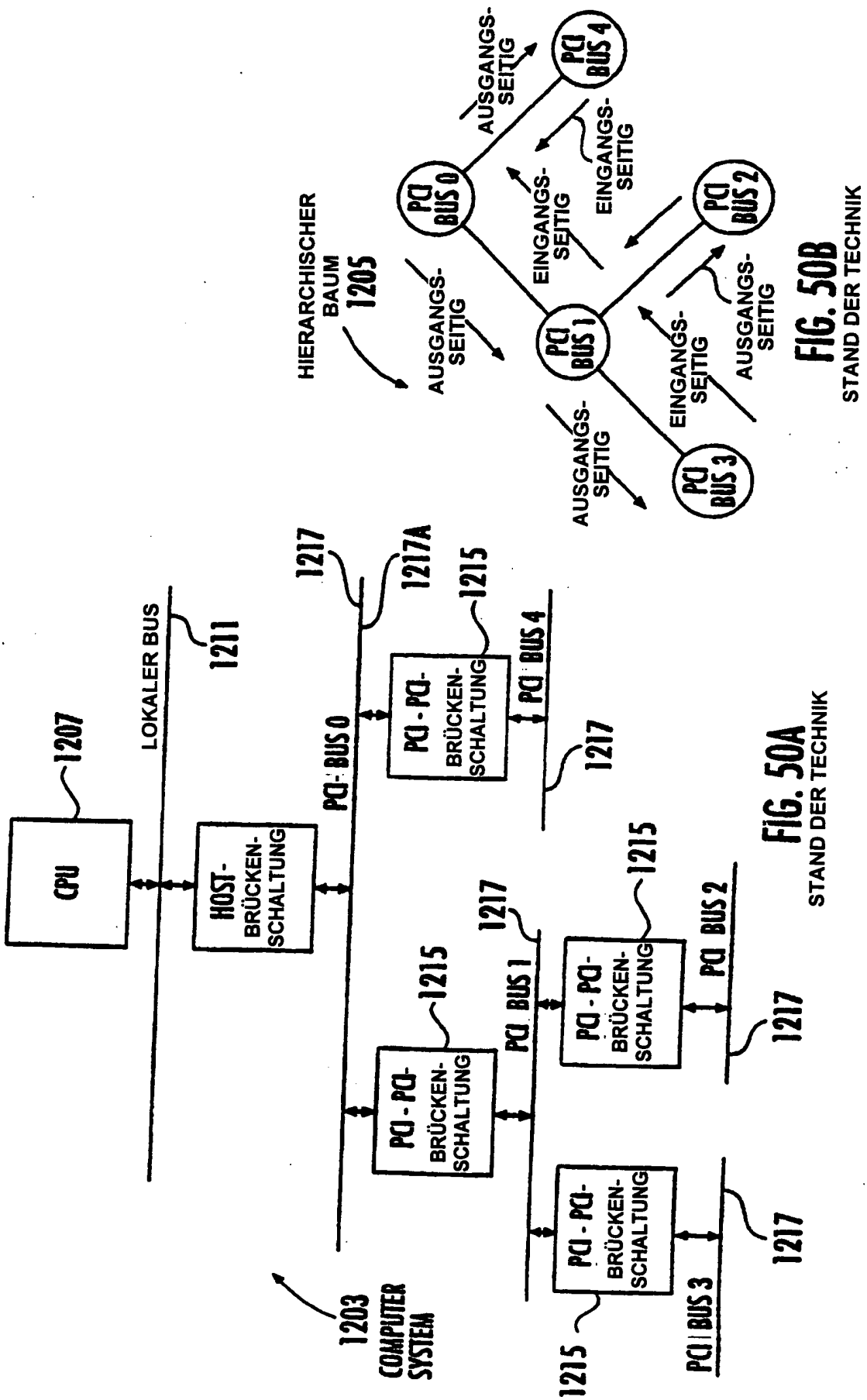


FIG. 50A
STAND DER TECHNIK

FIG. 50B
STAND DER TECHNIK

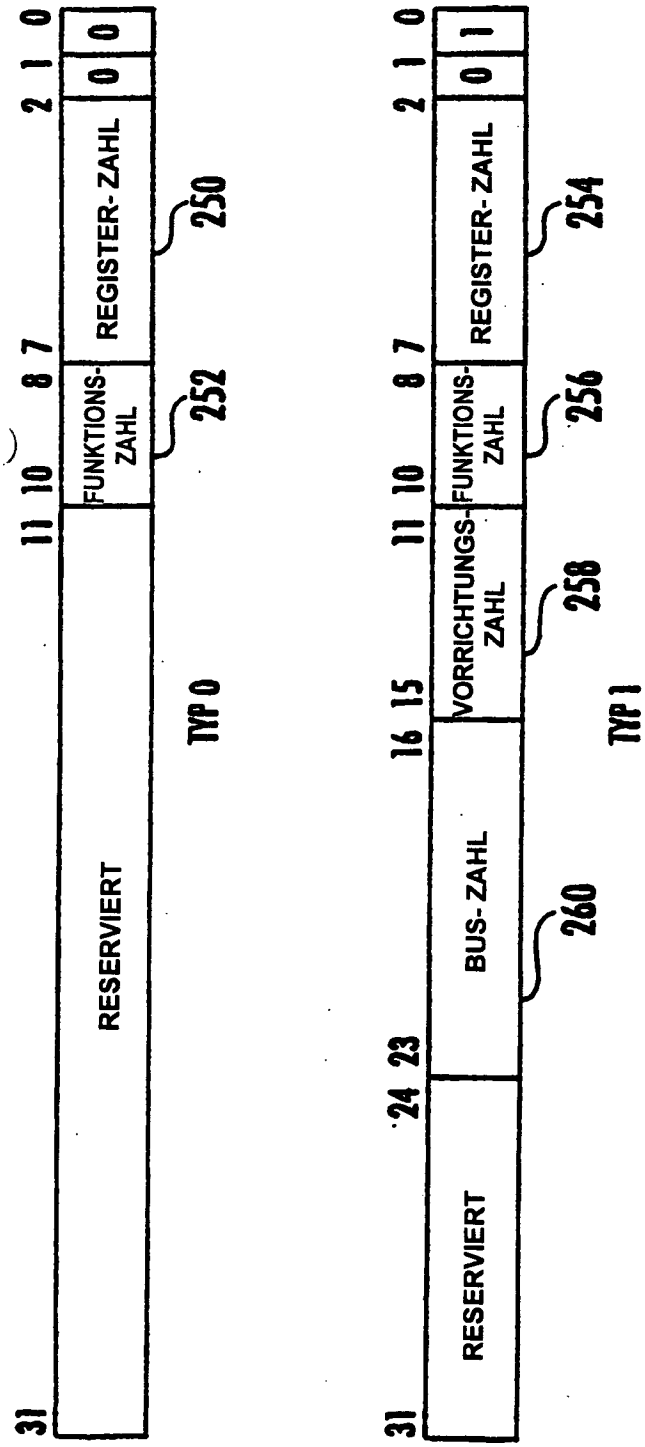

31
24 23
16 15
11 10
8 7
2 1 0
RESERVIERT
BUS- ZAHL
VORRICHTUNGS-ZAHL
FUNKTIONS-ZAHL
REGISTER- ZAHL
0
1
TYP 0
TYP 1
250
252
254
256
258
260

FIG. 51

PRIMÄR- ADRESSE AD [15::11]	SEKUNDÄR- ADRESSEN- BITS AD [31::16]
00000	0000 0000 0000 0001
00001	0000 0000 0000 0010
00010	0000 0000 0000 0100
00011	0000 0000 0000 1000
00100	0000 0000 0001 0000
00101	0000 0000 0010 0000
00110	0000 0000 0100 0000
00111	0000 0000 1000 0000
01000	0000 0001 0000 0000
01001	0000 0010 0000 0000
01010	0000 0100 0000 0000
01011	0000 1000 0000 0000
01100	0001 0000 0000 0000
01101	0010 0000 0000 0000
01110	0100 0000 0000 0000
01111	1000 0000 0000 0000
1XXXX	0000 0000 0000 0000

FIG. 52

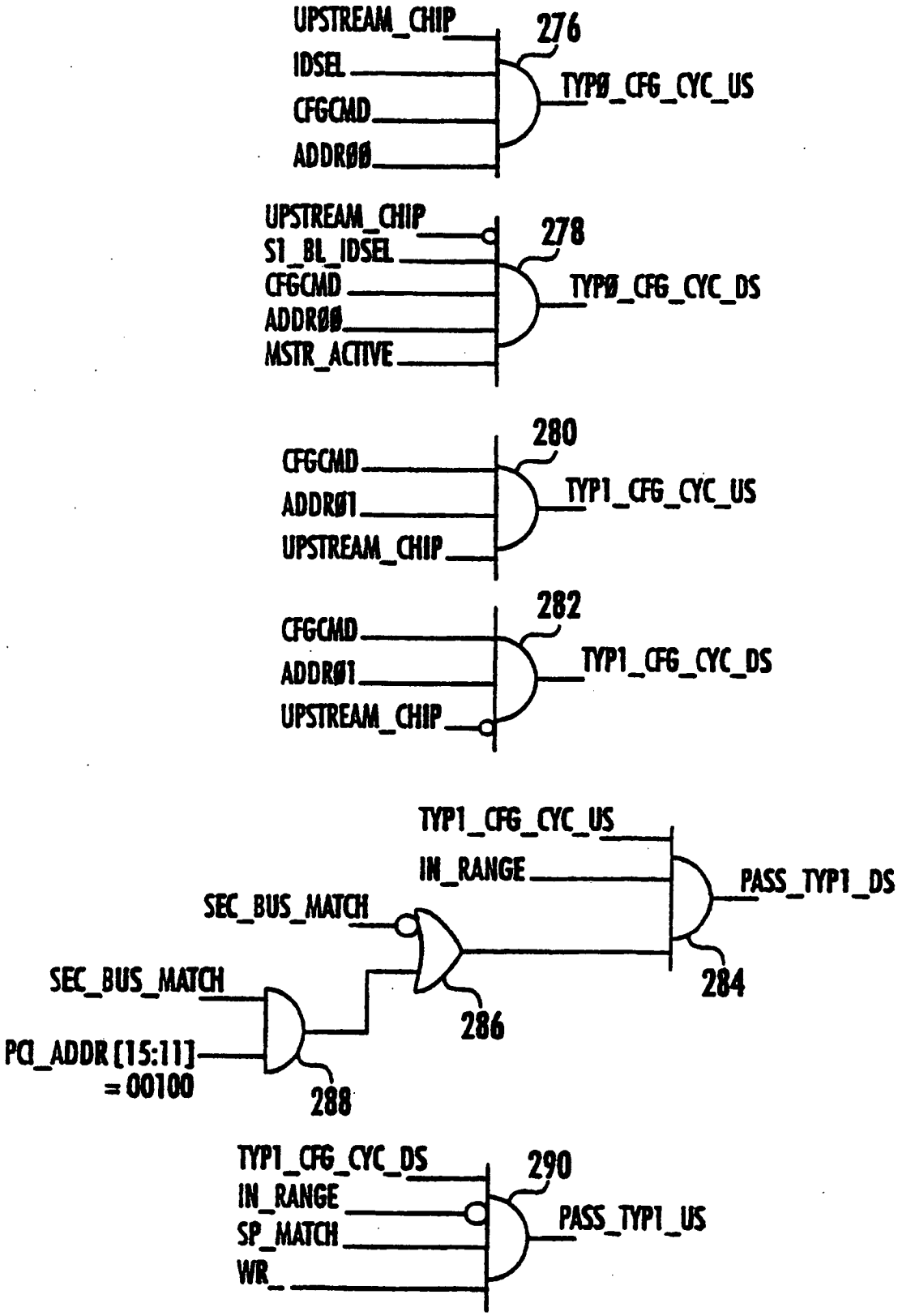

UPSTREAM_CHIP
IDSEL
CFGCMD
ADDR00
276
TYP0_CFG_CYC_US
UPSTREAM_CHIP
S1_BL_IDSEL
CFGCMD
ADDR00
MSTR_ACTIVE
278
TYP0_CFG_CYC_DS
CFGCMD
ADDR01
UPSTREAM_CHIP
280
TYP1_CFG_CYC_US
CFGCMD
ADDR01
UPSTREAM_CHIP
282
TYP1_CFG_CYC_DS
TYP1_CFG_CYC_US
IN_RANGE
SEC_BUS_MATCH
PASS_TYP1_DS
284
SEC_BUS_MATCH
286
PCI_ADDR[15:11]
=00100
288
TYP1_CFG_CYC_DS
IN_RANGE
SP_MATCH
WR_
290
PASS_TYP1_US

FIG. 53A

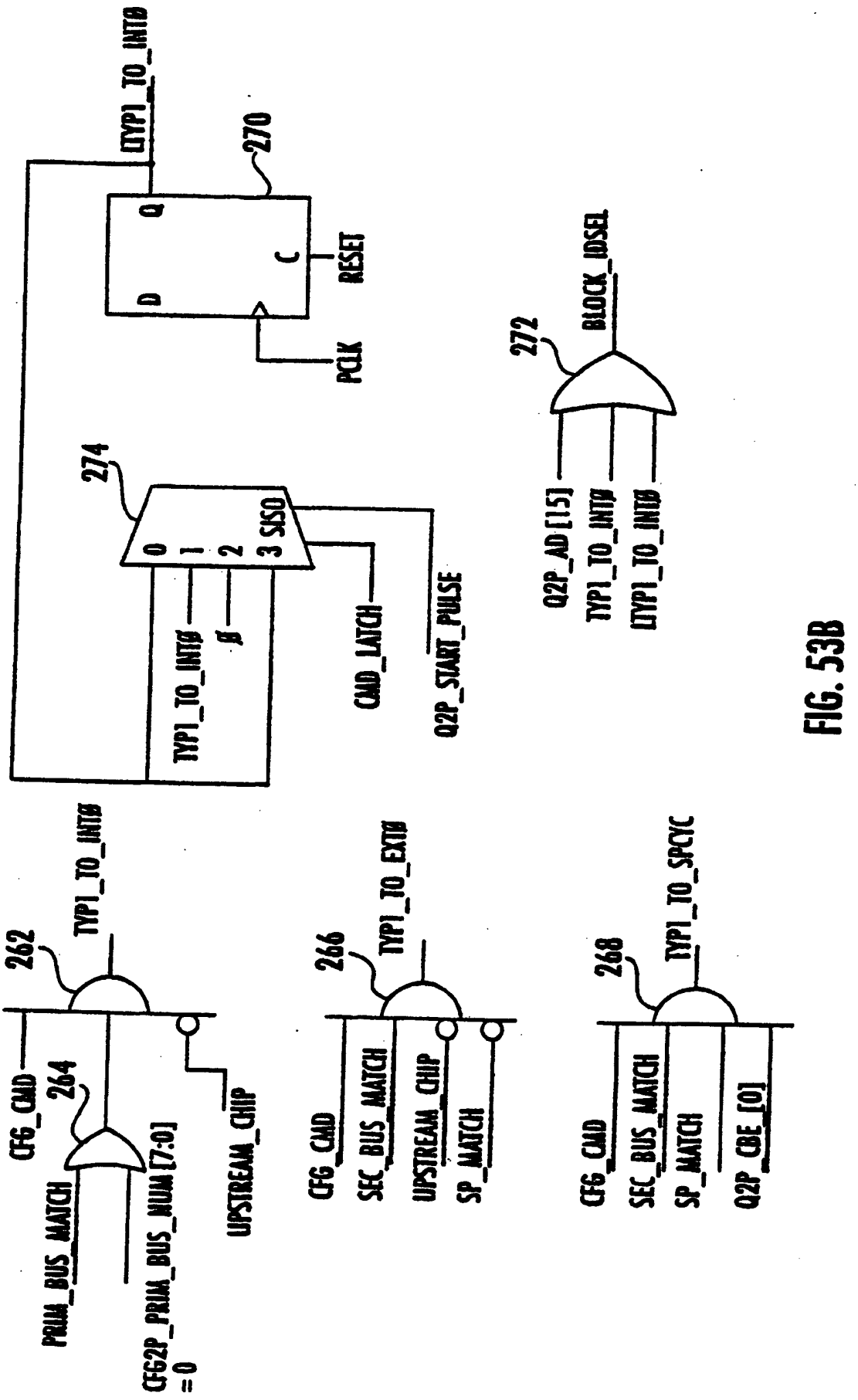

LTYP1_TO_INTB
270
Q
D
C
RESET
PCLK
274
0
1
2
3 SKSO
TYP1_TO_INTB
CMD_LATCH
Q2P_START_PULSE
272
BLOCK_IDSEL
Q2P_AD [15]
TYP1_TO_INTB
LTYP1_TO_INTB
262
TYP1_TO_INTB
CFG_CMD
264
PRIM_BUS_MATCH
CFG2P_PRIM_BUS_NUM [7:0]
= 0
UPSTREAM_CHIP
266
TYP1_TO_EXTB
CFG_CMD
SEC_BUS_MATCH
UPSTREAM_CHIP
SP_MATCH
268
TYP1_TO_SPCYC
CFG_CMD
SEC_BUS_MATCH
SP_MATCH
Q2P_CBE_[0]

FIG. 53B

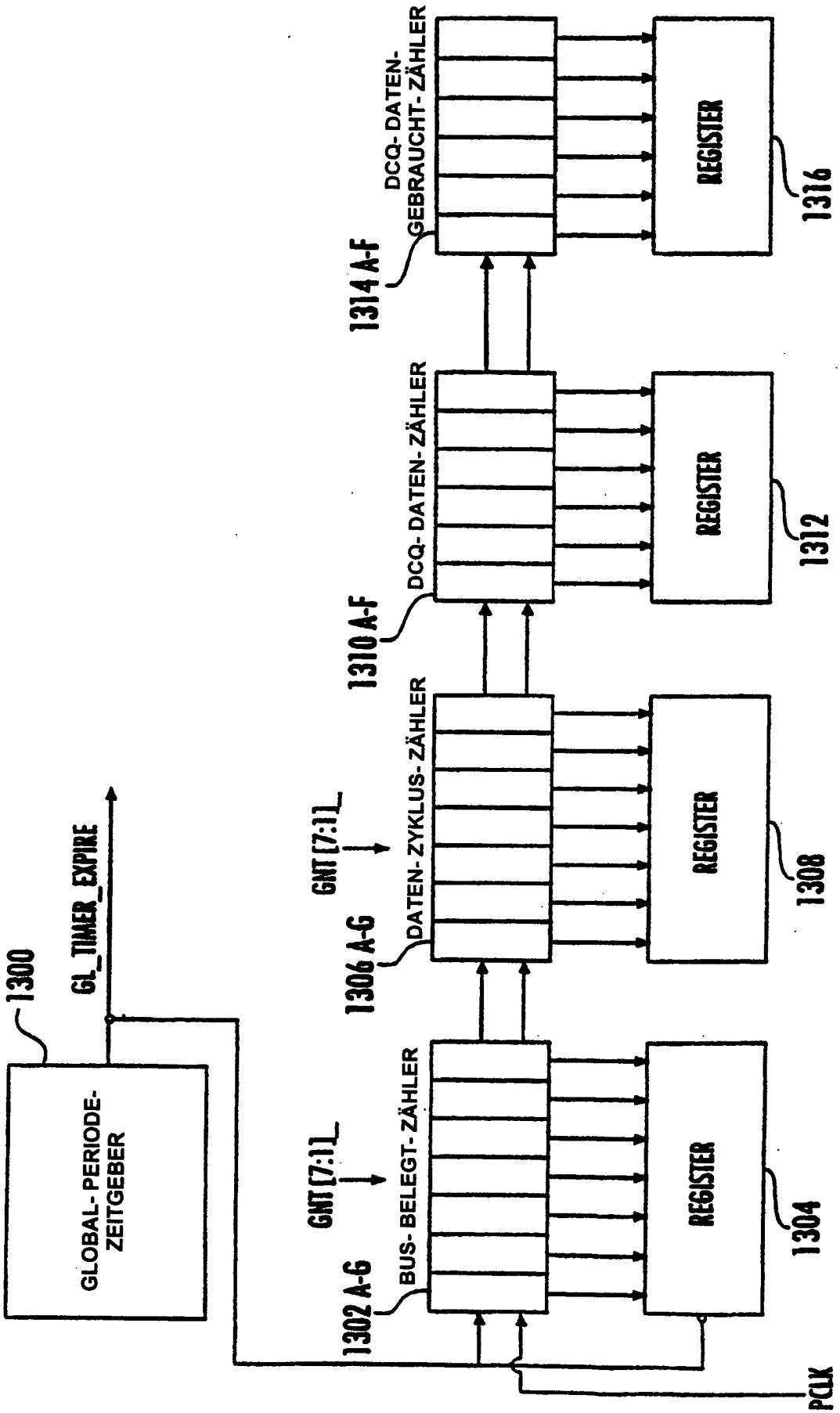

1300
GL_TIMER_EXPIRE
GLOBAL- PERIODE- ZEITGEBER
GNT [7:1]_
1302 A-G
BUS- BELEGT- ZÄHLER
REGISTER
1304
GNT [7:1]_
1306 A-G
DATEN- ZYKLUS- ZÄHLER
REGISTER
1308
1310 A-F
DCQ- DATEN- ZÄHLER
REGISTER
1312
1314 A-F
DCQ- DATEN- GEBRAUCHT- ZÄHLER
REGISTER
1316
PCLK

FIG. 54A

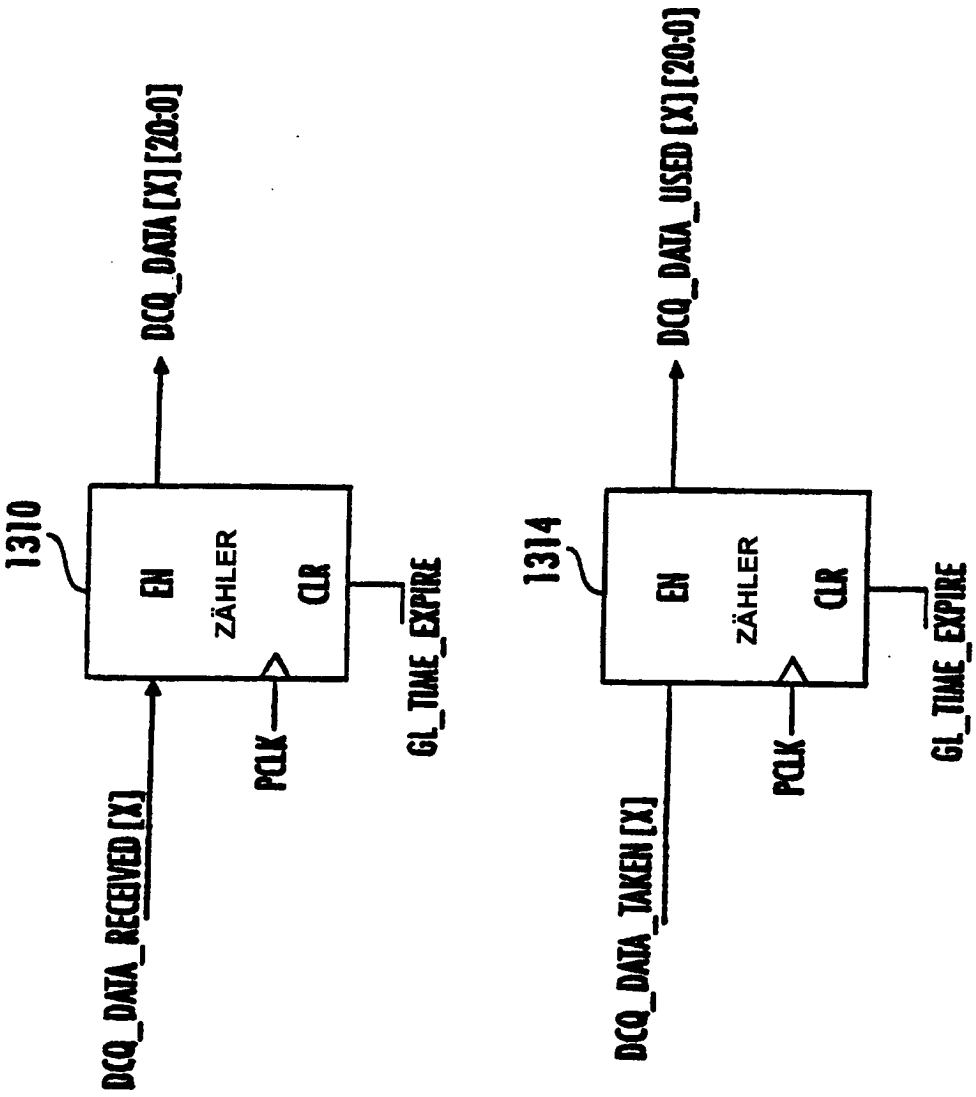

1310
EN
ZÄHLER
CLR
DCQ_DATA [X] [20:0]
DCQ_DATA_RECEIVED [X]
PCLK
GL_TIME_EXPIRE
1314
EN
ZÄHLER
CLR
DCQ_DATA_USED [X] [20:0]
DCQ_DATA_TAKEN [X]
PCLK
GL_TIME_EXPIRE

FIG. 54B

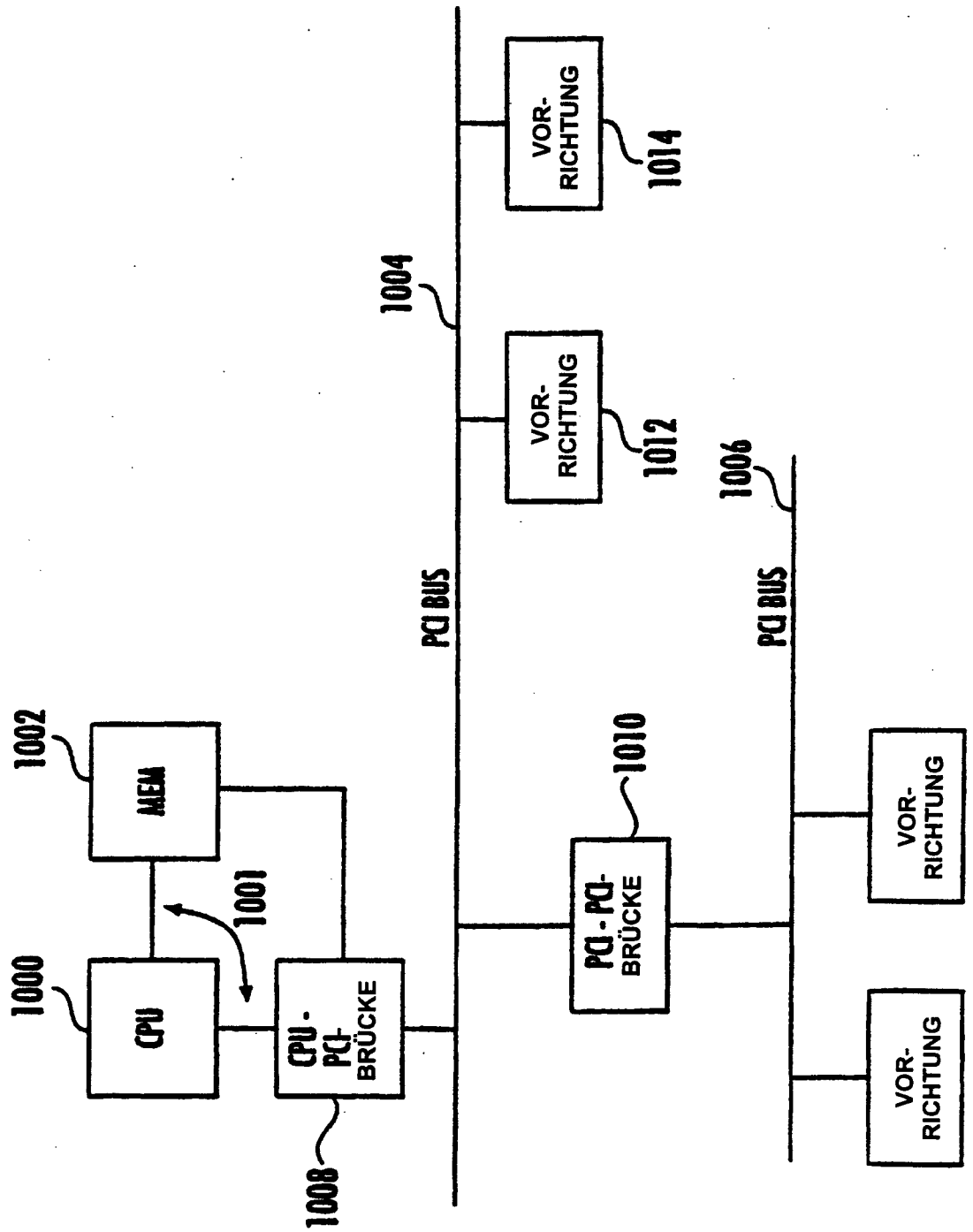


FIG. 55
STAND DER TECHNIK

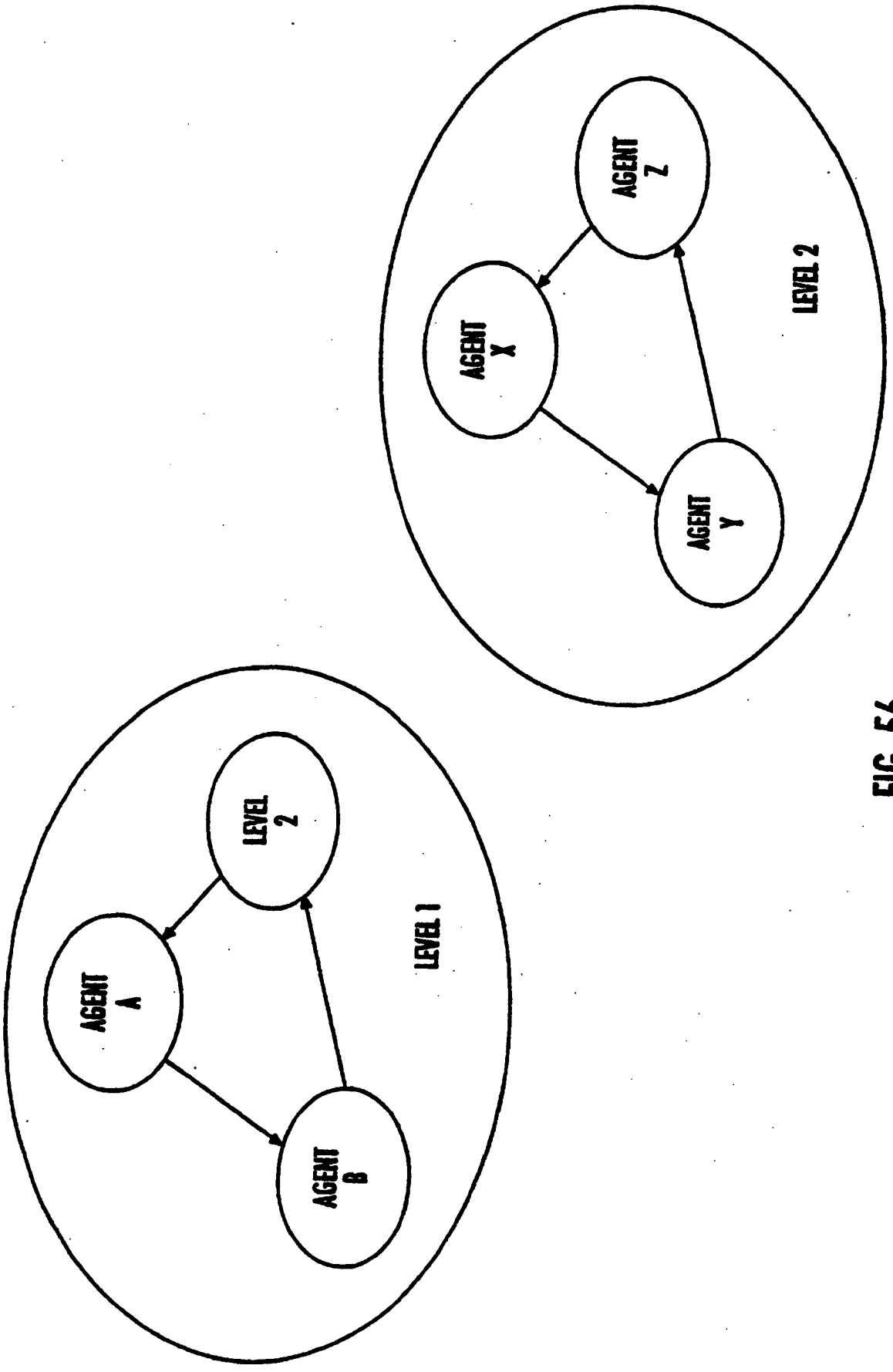


FIG. 56
STAND DER TECHNIK

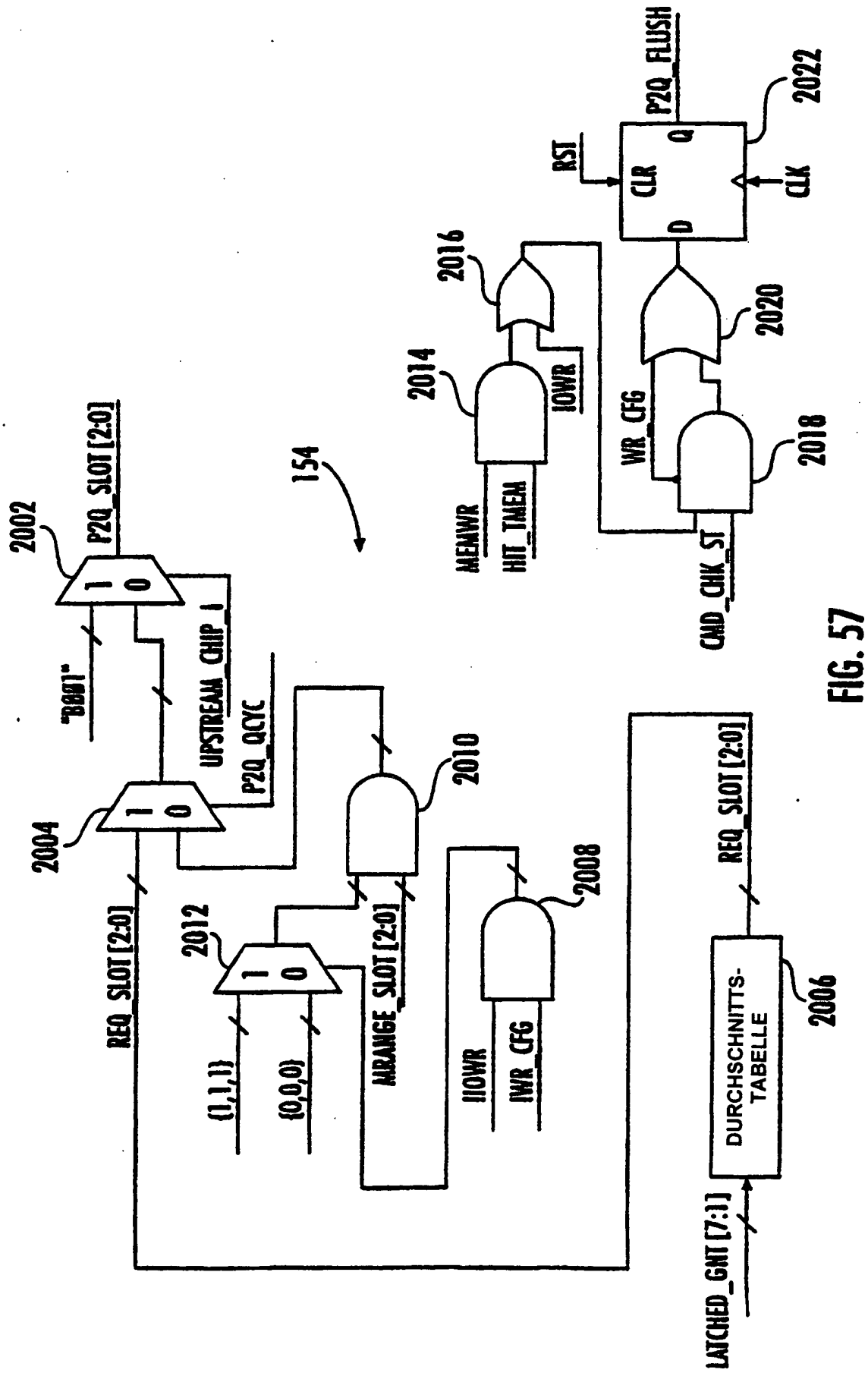

154
2002
P2Q_SLOT [2:0]
"BOOT"
UPSTREAM_CHIP_1
2004
REQ_SLOT [2:0]
P2Q_QCYC
2010
2012
{1,1,1}
{0,0,0}
MRANGE_SLOT [2:0]
2008
IIOWR
IWR_CFG
2014
2016
MEMWR
HIT_TMEM
IOWR
WR_CFG
2018
2020
CMD_CHK_ST
RST
CLR
D
Q
CLK
P2Q_FLUSH
2022
REQ_SLOT [2:0]
DURCHSCHNITTS-
TABELLE
2006
LATCHED_GNT [7:1]

FIG. 57

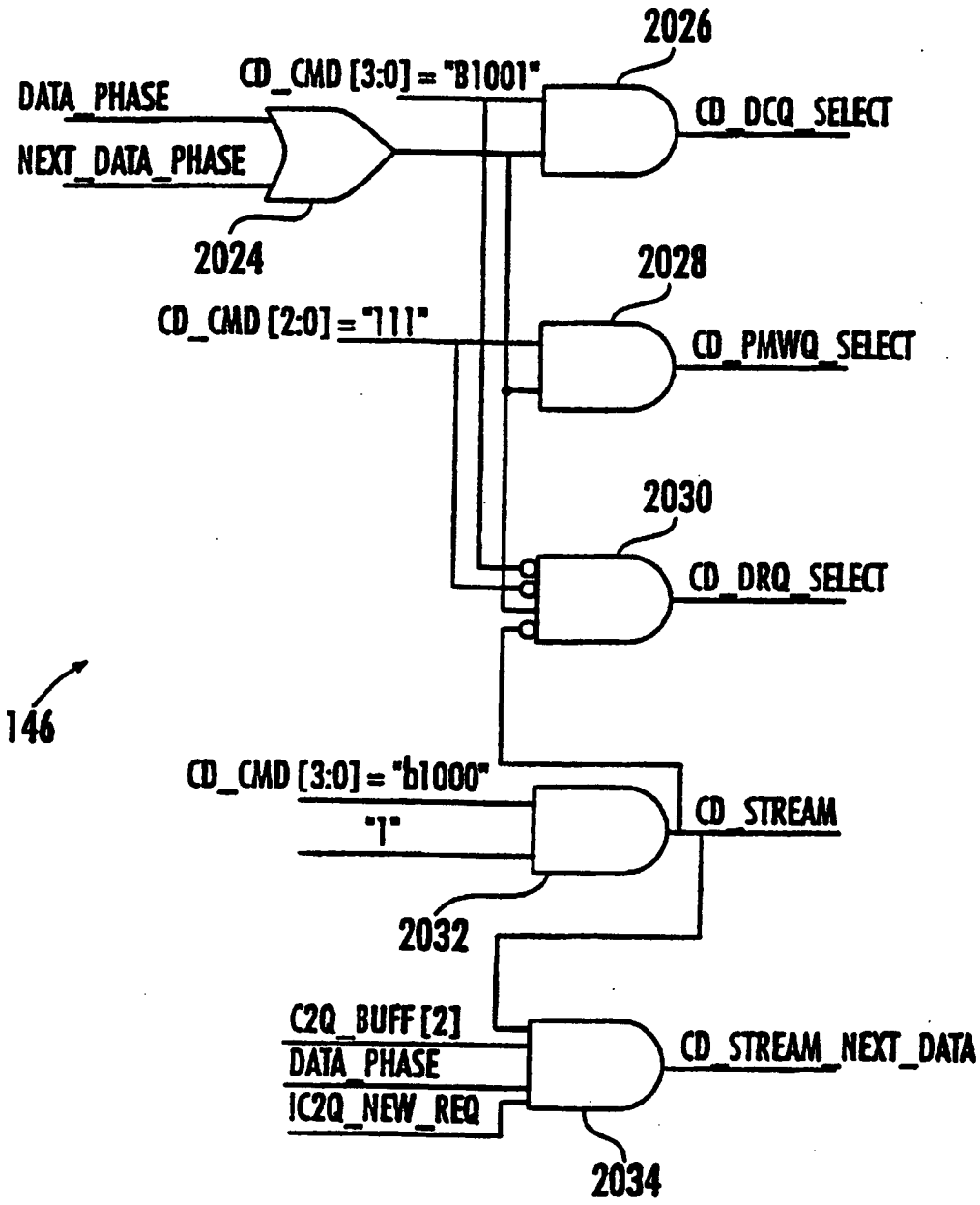

2026
CD_CMD [3:0] = "B1001"
DATA_PHASE
NEXT_DATA_PHASE
CD_DCQ_SELECT
2024
2028
CD_CMD [2:0] = "111"
CD_PMWQ_SELECT
2030
CD_DRQ_SELECT
146
CD_CMD [3:0] = "b1000"
"1"
CD_STREAM
2032
C2Q_BUFF [2]
DATA_PHASE
!C2Q_NEW_REQ
CD_STREAM_NEXT_DATA
2034

FIG. 58

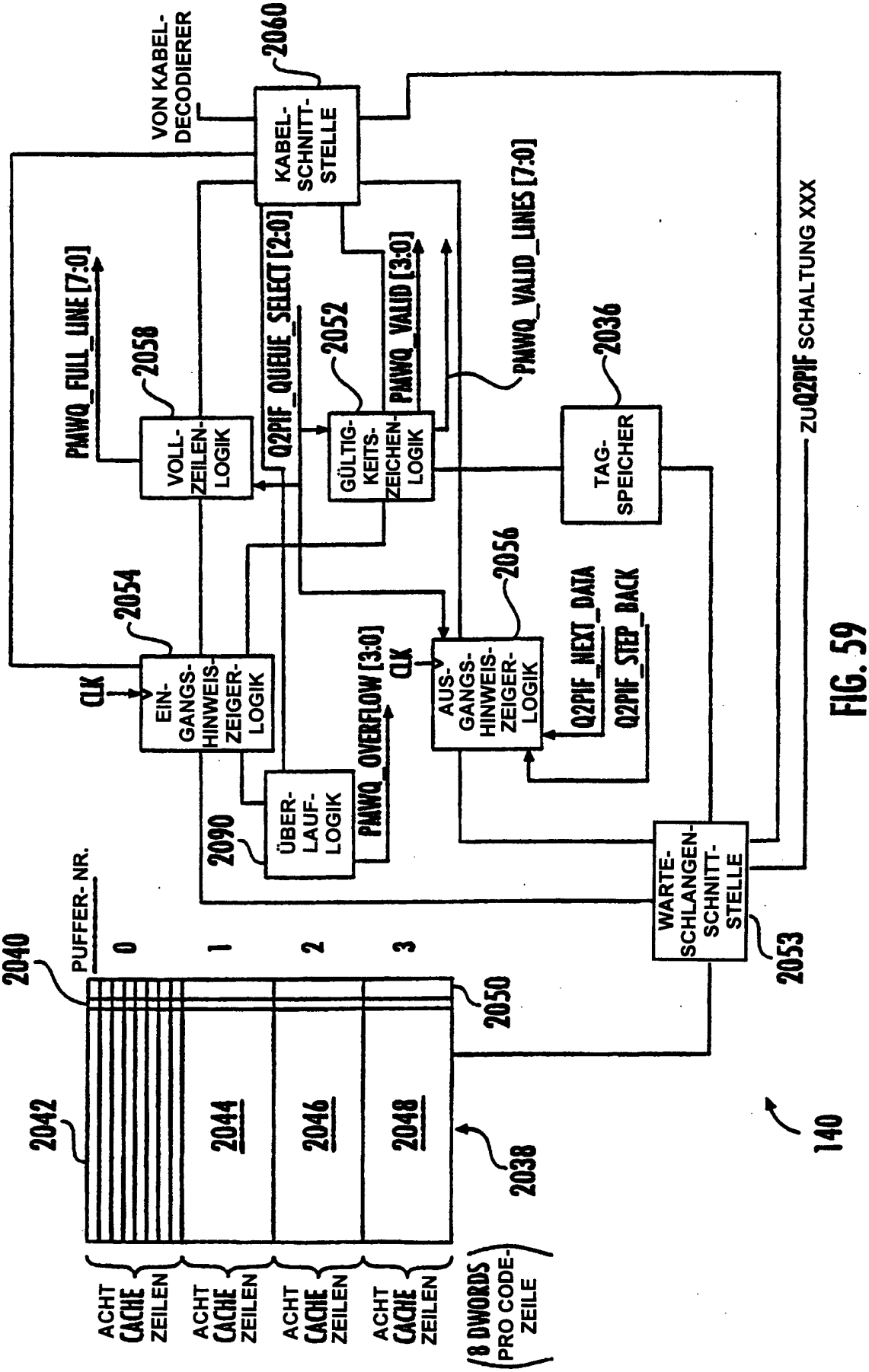

VON KABEL-DECODIERER
KABEL-SCHNITT-STELLE
2060
PMWQ_FULL_LINE [7:0]
2058
VOLL-ZEILEN-LOGIK
Q2PIF_QUEUE_SELECT [2:0]
2052
GÜLTIG-KEITS-ZEICHEN-LOGIK
PMWQ_VALID [3:0]
PMWQ_VALID_LINES [7:0]
2036
TAG-SPEICHER
ZU Q2PIF SCHALTUNG XXX
2054
CLK
EIN-GANGS-HINWEIS-ZEIGER-LOGIK
2056
AUS-GANGS-HINWEIS-ZEIGER-LOGIK
Q2PIF_NEXT_DATA
Q2PIF_STEP_BACK
PMWQ_OVERFLOW [3:0]
CLK
2090
ÜBER-LAUF-LOGIK
WARTE-SCHLANGEN-SCHNITT-STELLE
2053
2040
PUFFER-NR.
0
1
2
3
2050
2042
2044
2046
2048
2038
ACHT CACHE ZEILEN
ACHT CACHE ZEILEN
ACHT CACHE ZEILEN
ACHT CACHE ZEILEN
8 DWORDS PRO CODE-ZEILE
140

FIG. 59

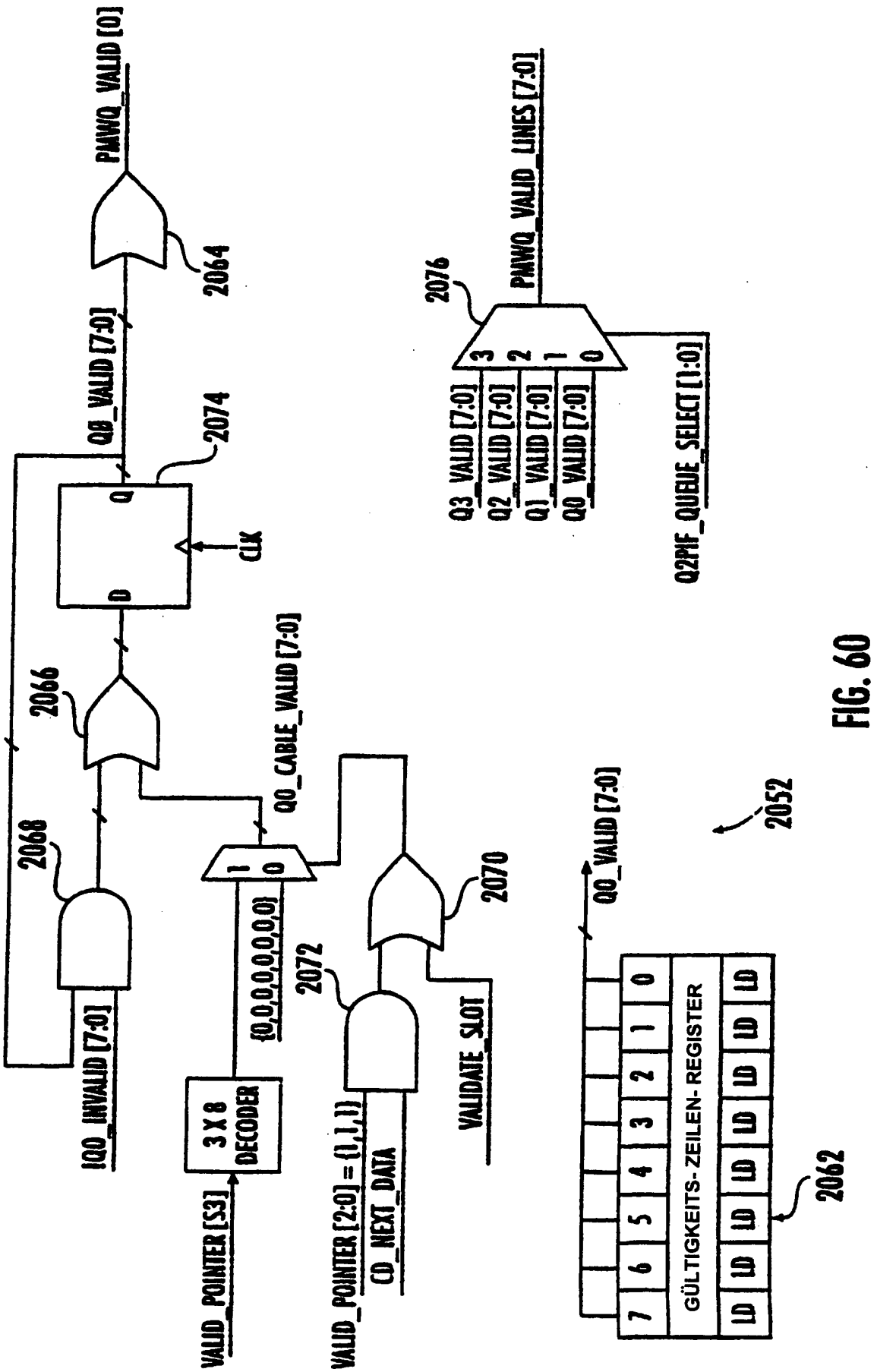

PMWQ_VALID [0]
2064
Q8_VALID [7:0]
2074
Q
D
CLK
2066
2068
!Q0_INVALID [7:0]
3 X 8 DECODER
VALID_POINTER [3:]
{0,0,0,0,0,0,0,0}
Q0_CABLE_VALID [7:0]
1
0
2072
VALID_POINTER [2:0] = {1,1,1}
CO_NEXT_DATA
2070
VALIDATE_SLOT
2076
Q3_VALID [7:0]
Q2_VALID [7:0]
Q1_VALID [7:0]
Q0_VALID [7:0]
3
2
1
0
PMWQ_VALID_LINES [7:0]
Q2PMF_QUEUE_SELECT [1:0]
Q0_VALID [7:0]
2052
7
6
5
4
3
2
1
0
GÜLTIGKEITS- ZEILEN- REGISTER
LD
2062

FIG. 60

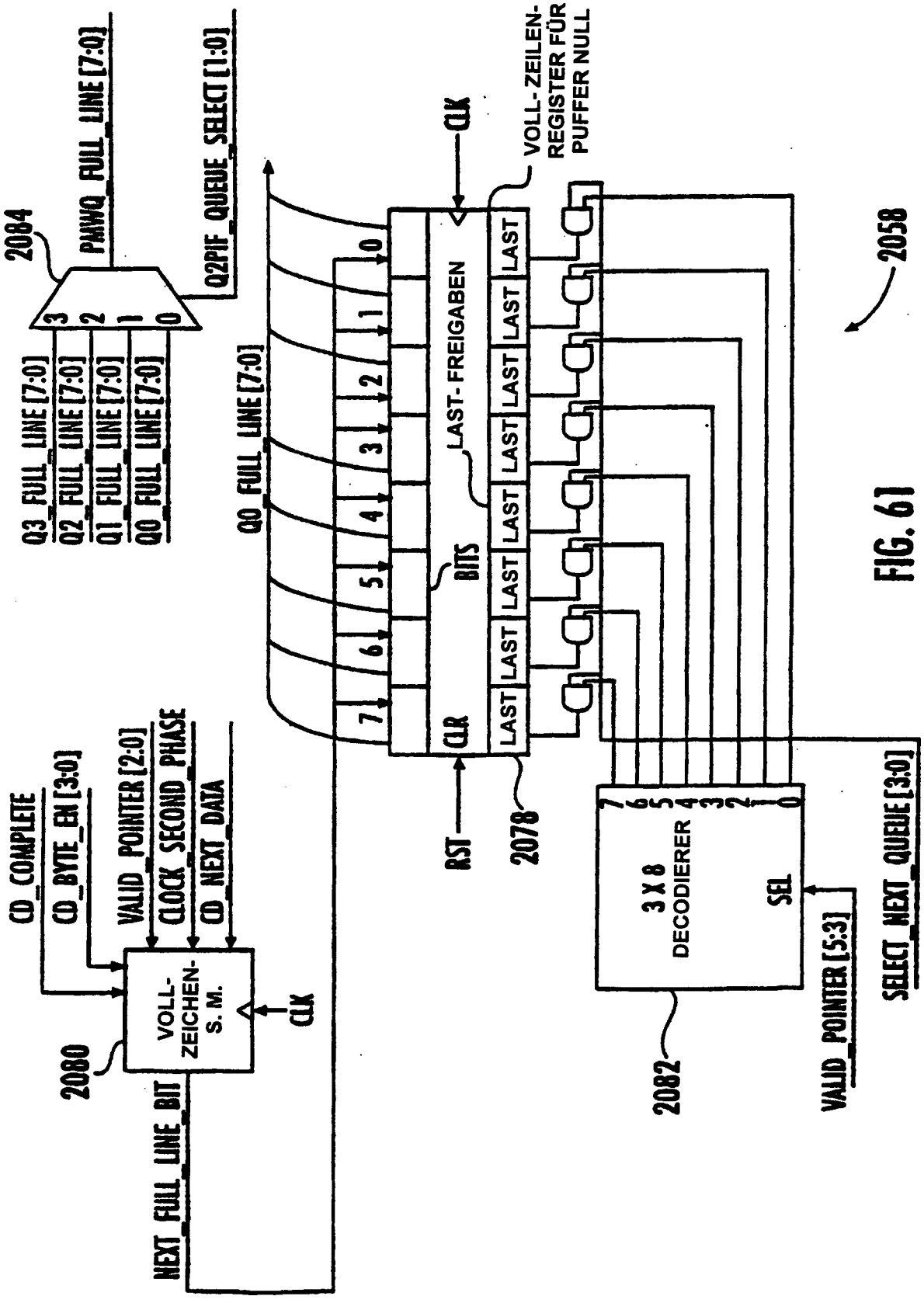

2084
PAWQ_FULL_LINE [7:0]
Q2PIF_QUEUE_SELECT [1:0]
Q3_FULL_LINE [7:0]
Q2_FULL_LINE [7:0]
Q1_FULL_LINE [7:0]
Q0_FULL_LINE [7:0]
CLK
VOLL- ZEILEN- REGISTER FÜR PUFFER NULL
LAST- FREIGABEN
BITS
CLR
RST
2078
2058
CD_COMPLETE
CD_BYTE_EN [3:0]
VALID_POINTER [2:0]
CLOCK_SECOND_PHASE
CD_NEXT_DATA
VOLL- ZEICHEN- S. M.
2080
NEXT_FULL_LINE_BIT
3 X 8 DECODIERER
SEL
2082
VALID_POINTER [5:3]
SELECT_NEXT_QUEUE [3:0]

FIG. 61

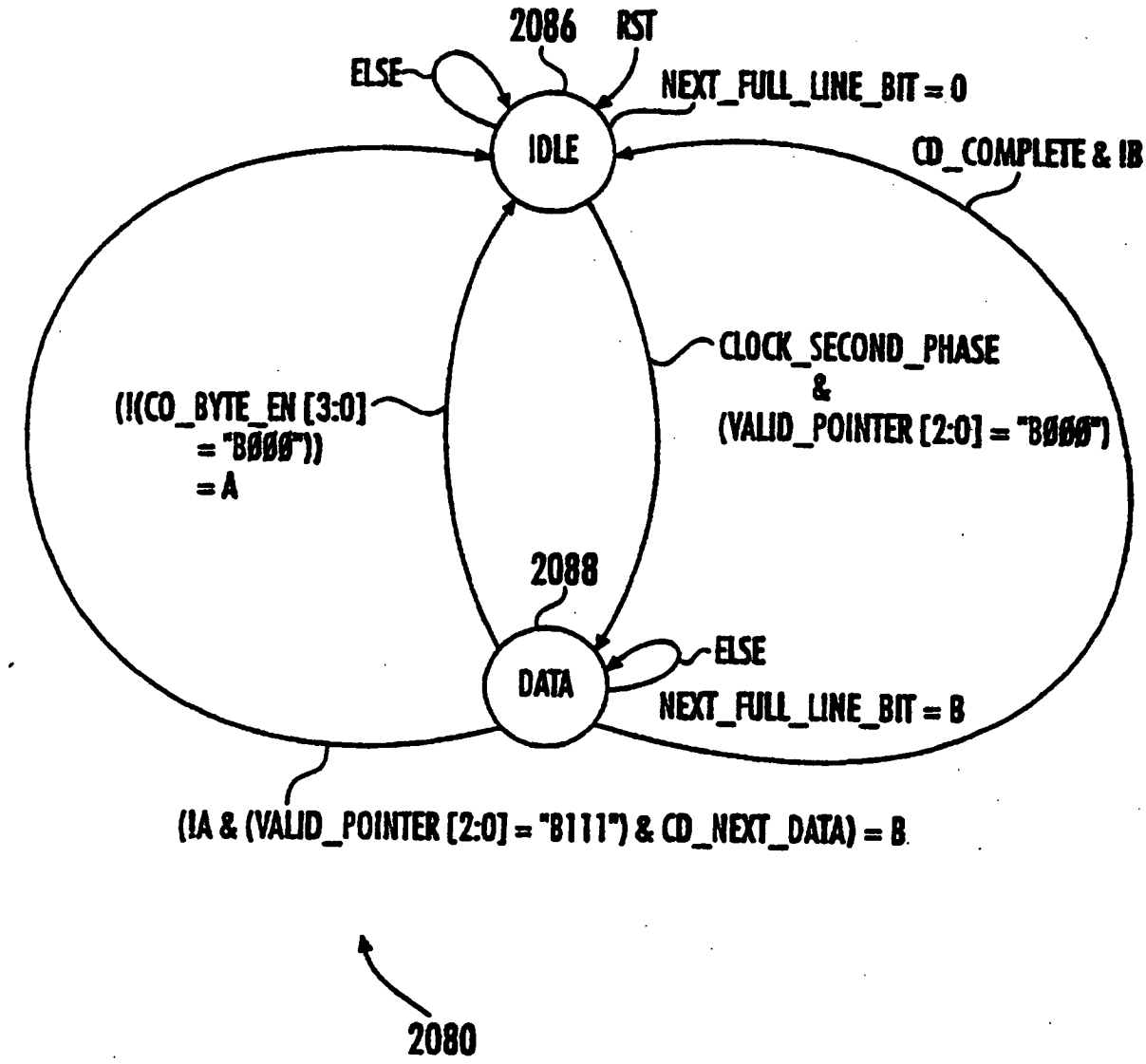

2086
RST
ELSE
NEXT_FULL_LINE_BIT = 0
IDLE
CD_COMPLETE & IB
CLOCK_SECOND_PHASE
&
(VALID_POINTER [2:0] = "BØØØ")
(!(CO_BYTE_EN [3:0]
= "BØØØ"))
= A
2088
ELSE
DATA
NEXT_FULL_LINE_BIT = B
(IA & (VALID_POINTER [2:0] = "B111") & CD_NEXT_DATA) = B
2080

FIG. 62

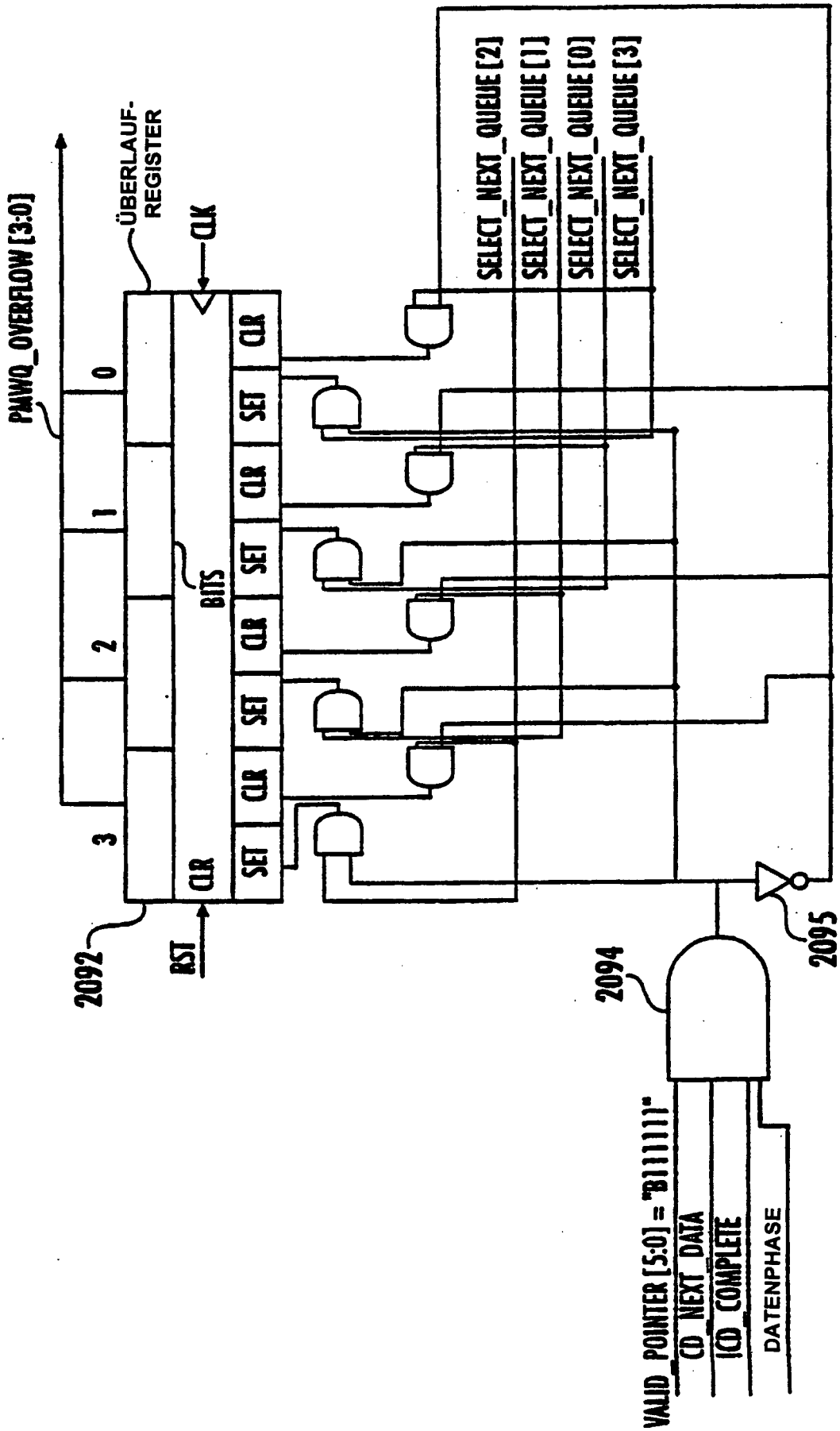

PMWQ_OVERFLOW [3:0]
ÜBERLAUF-REGISTER
CLK
BITS
RST
CLR
SET
0
1
2
3
2092
SELECT_NEXT_QUEUE [2]
SELECT_NEXT_QUEUE [1]
SELECT_NEXT_QUEUE [0]
SELECT_NEXT_QUEUE [3]
2090
2095
2094
VALID_POINTER [5:0] = "B111111"
CO_NEXT_DATA
ICO_COMPLETE
DATENPHASE

FIG. 63

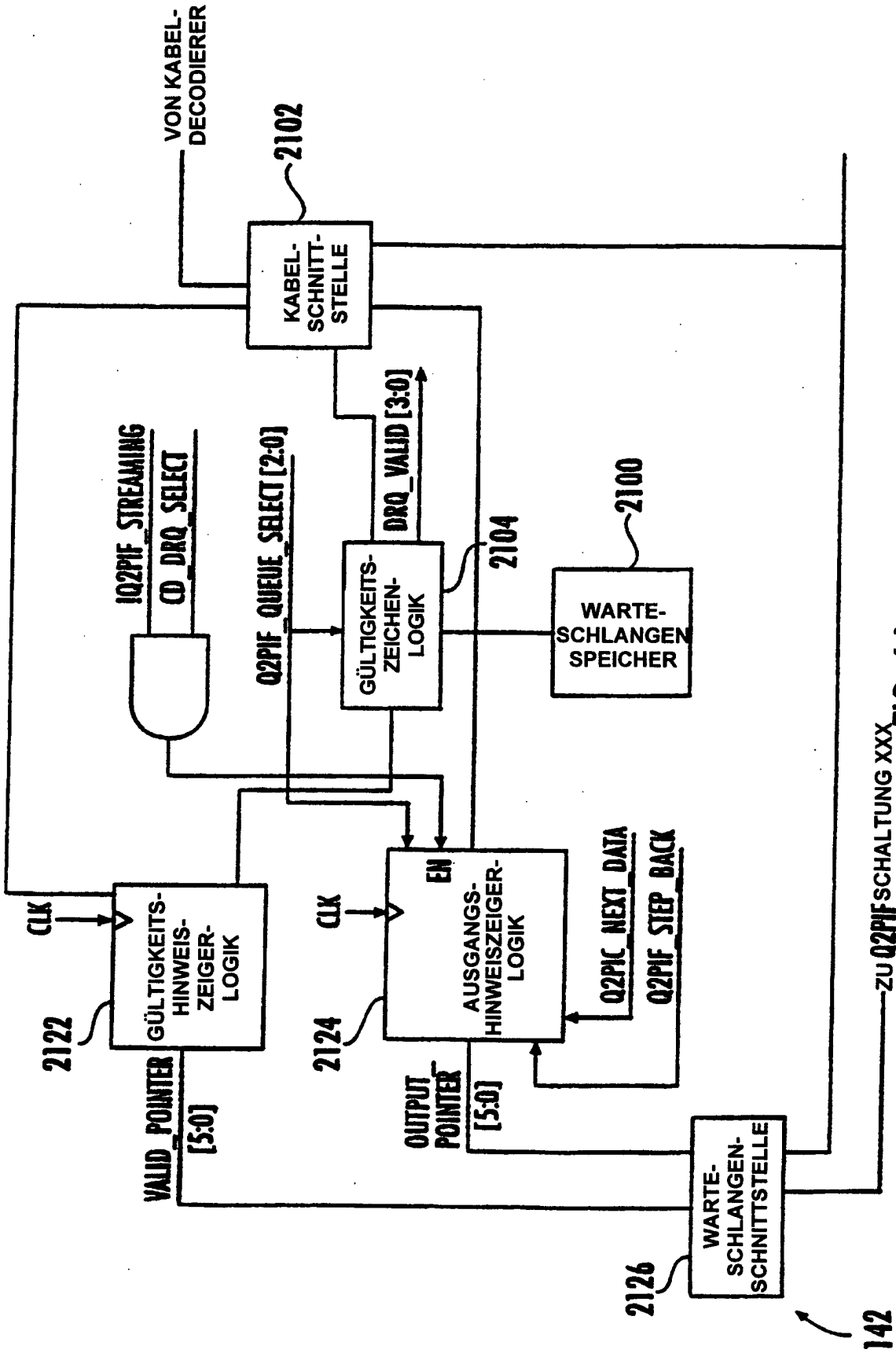

VON KABEL-DECODIERER
2102
KABEL-SCHNITT-STELLE
IQ2PIF_STREAMING
CD_DRQ_SELECT
Q2PIF_QUEUE_SELECT [2:0]
DRQ_VALID [3:0]
GÜLTIGKEITS-ZEICHEN-LOGIK
2104
2100
WARTE-SCHLANGEN SPEICHER
CLK
GÜLTIGKEITS-HINWEIS-ZEIGER-LOGIK
2122
VALID_POINTER [5:0]
CLK
EN
AUSGANGS-HINWEISZEIGER-LOGIK
2124
OUTPUT_POINTER [5:0]
Q2PIC_NEXT_DATA
Q2PIF_STEP_BACK
ZU Q2PIF SCHALTUNG XXX
WARTE-SCHLANGEN-SCHNITTSTELLE
2126
142

FIG. 64

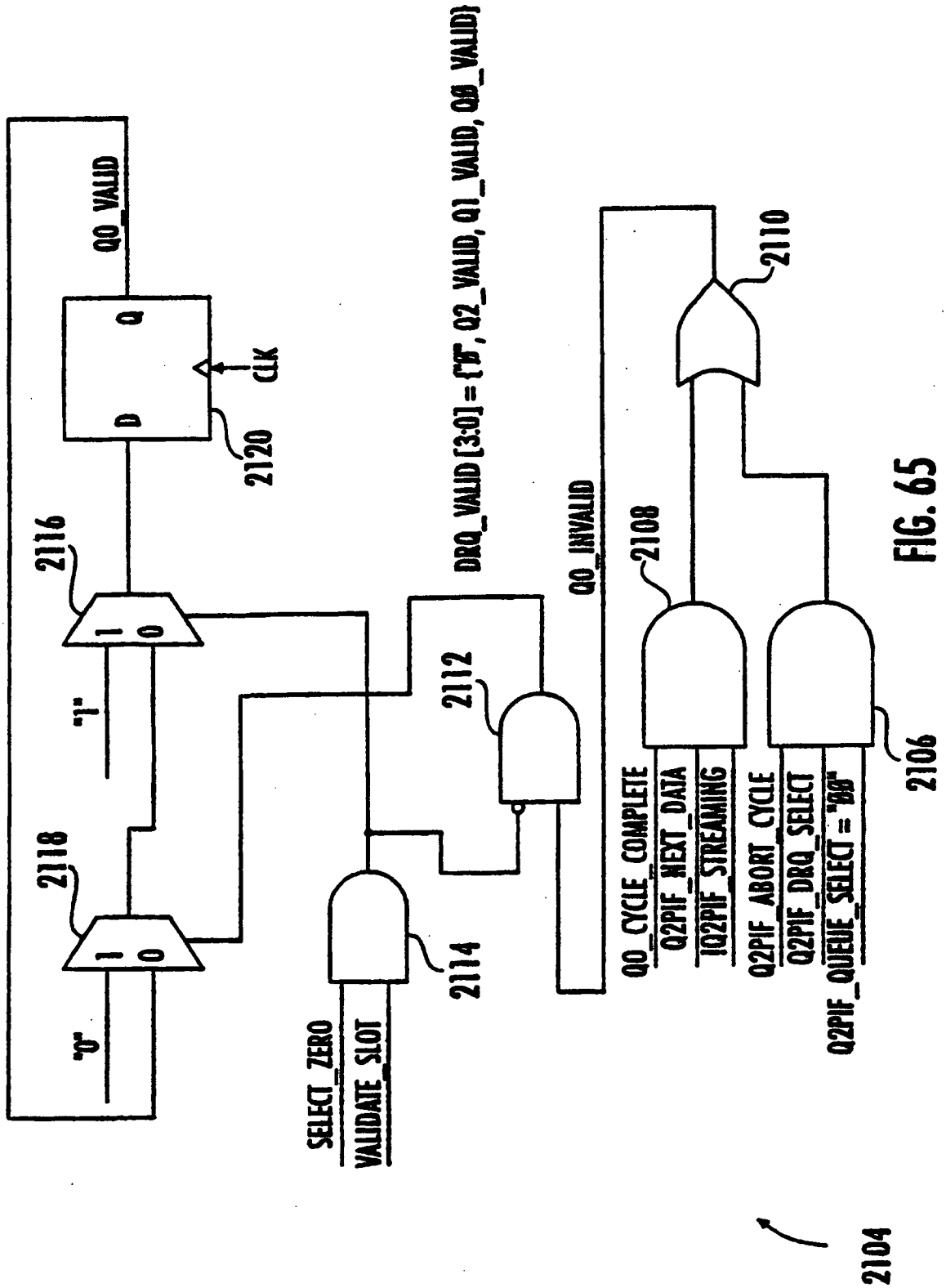

Q0_VALID
Q
D
CLK
2120
2116
1
0
"1"
2118
1
0
"0"
SELECT_ZERO
VALIDATE_SLOT
2114
DRQ_VALID[3:0] = {"0", Q2_VALID, Q1_VALID, Q0_VALID}
2112
Q0_INVALID
2108
Q0_CYCLE_COMPLETE
Q2PIF_NEXT_DATA
!Q2PIF_STREAMING
Q2PIF_ABORT_CYCLE
Q2PIF_DRQ_SELECT
Q2PIF_QUEUE_SELECT = "00"
2110
2106
2104

FIG. 65

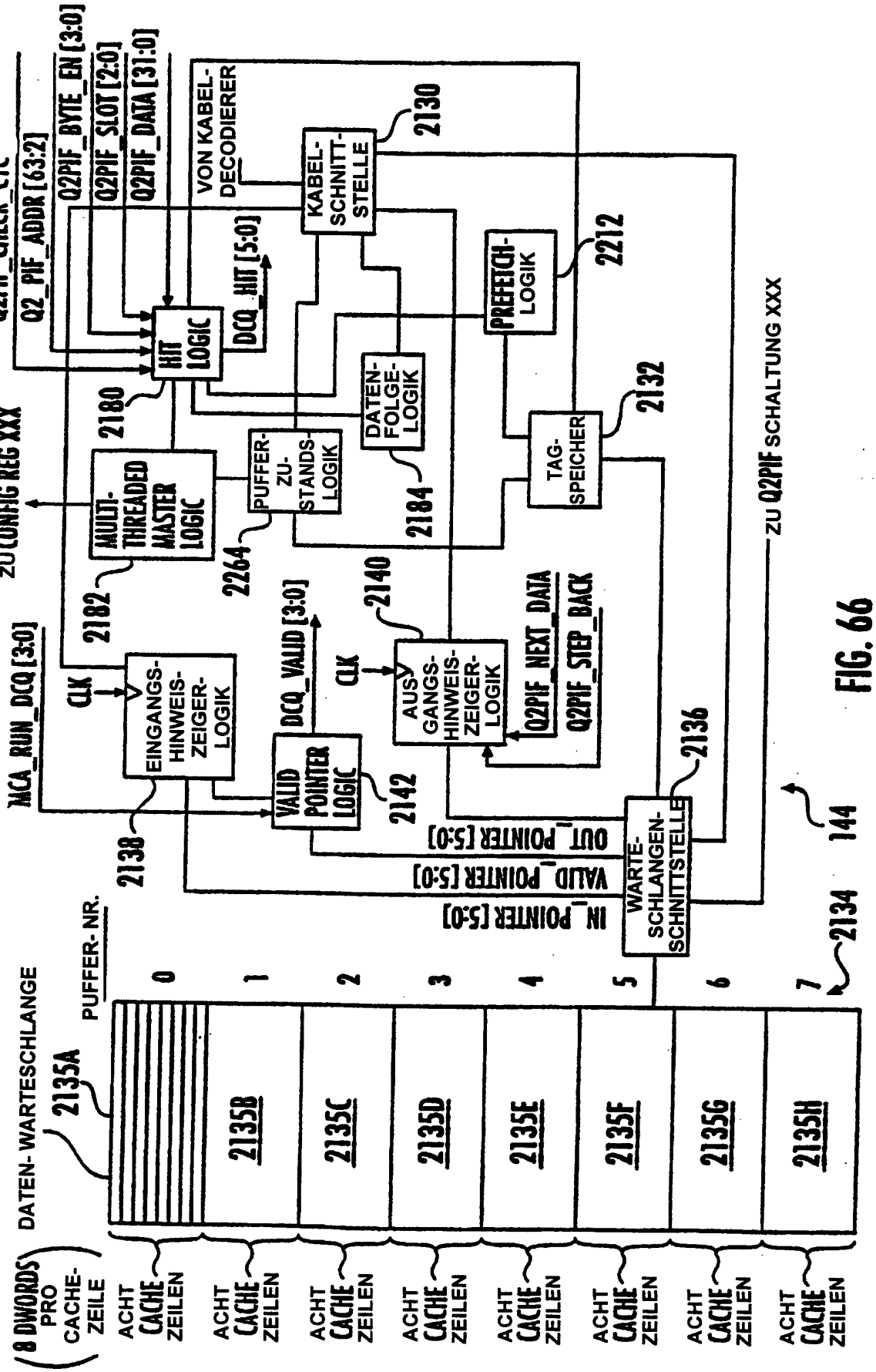

Q2PIF_CHECK_CYC
Q2_PIF_ADDR [63:2]
Q2PIF_BYTE_EN [3:0]
Q2PIF_SLOT [2:0]
Q2PIF_DATA [31:0]
VON KABEL-DECODIERER
KABEL-SCHNITT-STELLE
2130
HIT LOGIC
DCQ_HIT [5:0]
PREFETCH-LOGIK
2212
2180
DATEN-FOLGE-LOGIK
PUFFER-ZU-STANDS-LOGIK
TAG-SPEICHER
2132
ZU CONFIG REG XXX
MULTI-THREADED MASTER LOGIC
2182
2264
2184
ZU Q2PIF SCHALTUNG XXX
MCA_RUN_DCQ [3:0]
CLK
EINGANGS-HINWEIS-ZEIGER-LOGIK
DCQ_VALID [3:0]
CLK
AUS-GANGS-HINWEIS-ZEIGER-LOGIK
2140
Q2PIF_NEXT_DATA
Q2PIF_STEP_BACK
VALID POINTER LOGIC
2142
2138
OUT_POINTER [5:0]
VALID_POINTER [5:0]
IN_POINTER [5:0]
WARTE-SCHLANGEN-SCHNITTSTELLE
2136
144
2134
PUFFER-NR.
0
1
2
3
4
5
6
7
DATEN-WARTESCHLANGE
2135A
2135B
2135C
2135D
2135E
2135F
2135G
2135H
(8 DWORDS PRO CACHE-ZEILE)
ACHT CACHE ZEILEN

FIG. 66

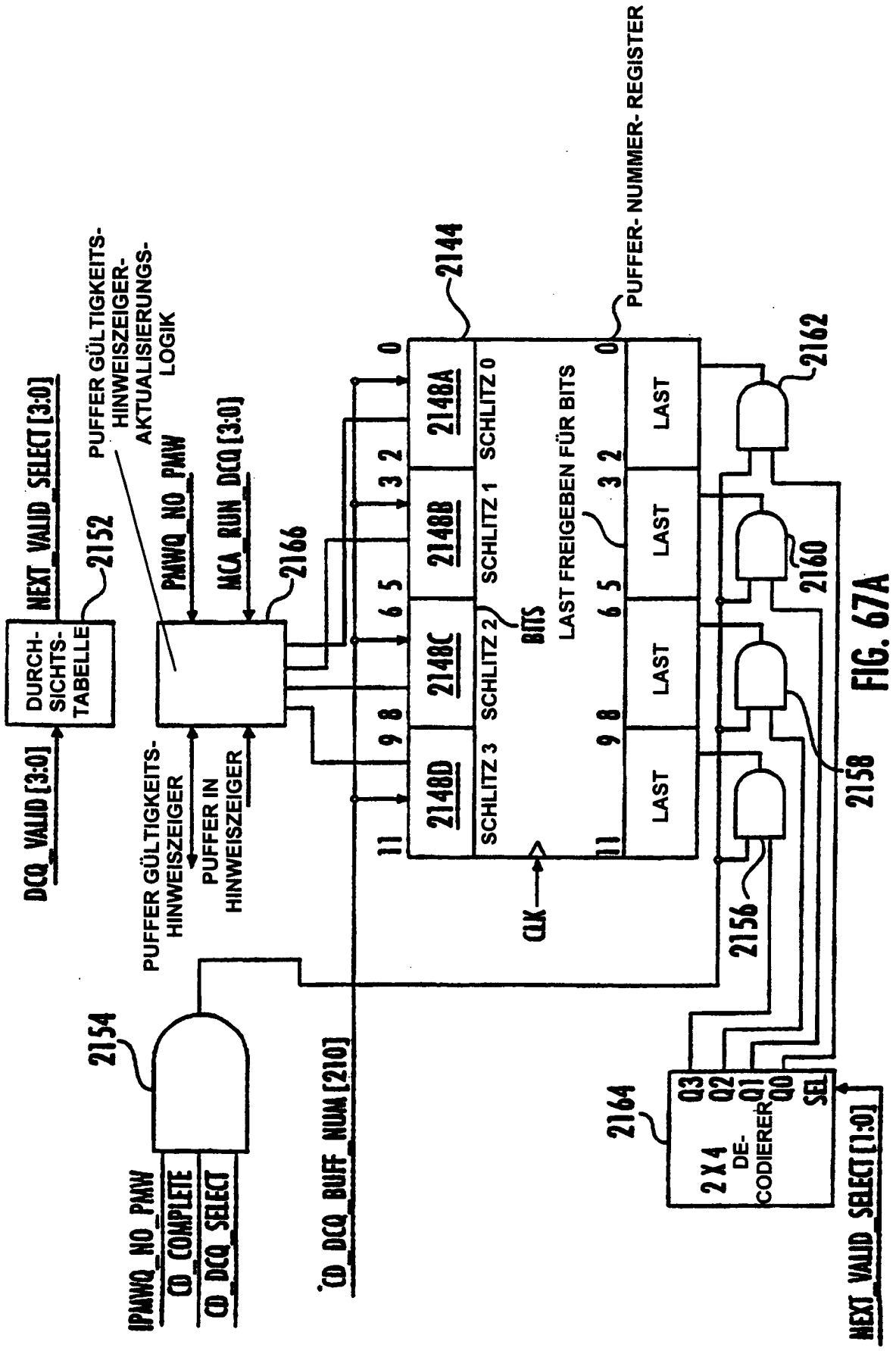

DCQ_VALID [3:0]
DURCH-SICHTS-TABELLE
NEXT_VALID_SELECT [3:0]
2152
PUFFER GÜLTIGKEITS-HINWEISZEIGER-AKTUALISIERUNGS-LOGIK
PMWQ_NO_PMW
NCA_RUN_DCQ [3:0]
2166
PUFFER GÜLTIGKEITS-HINWEISZEIGER
PUFFER IN HINWEISZEIGER
2144
2148A
2148B
2148C
2148D
SCHLITZ 0
SCHLITZ 1
SCHLITZ 2
SCHLITZ 3
BITS
CLK
PUFFER- NUMMER- REGISTER
LAST FREIGEBEN FÜR BITS
LAST
2162
2160
2158
2156
2154
IPMWQ_NO_PMW
CD_COMPLETE
CD_DCQ_SELECT
CD_DCQ_BUFF_NUM [2:0]
2164
2 X 4 DE-CODIERER
Q3
Q2
Q1
Q0
SEL
NEXT_VALID_SELECT [1:0]

FIG. 67A

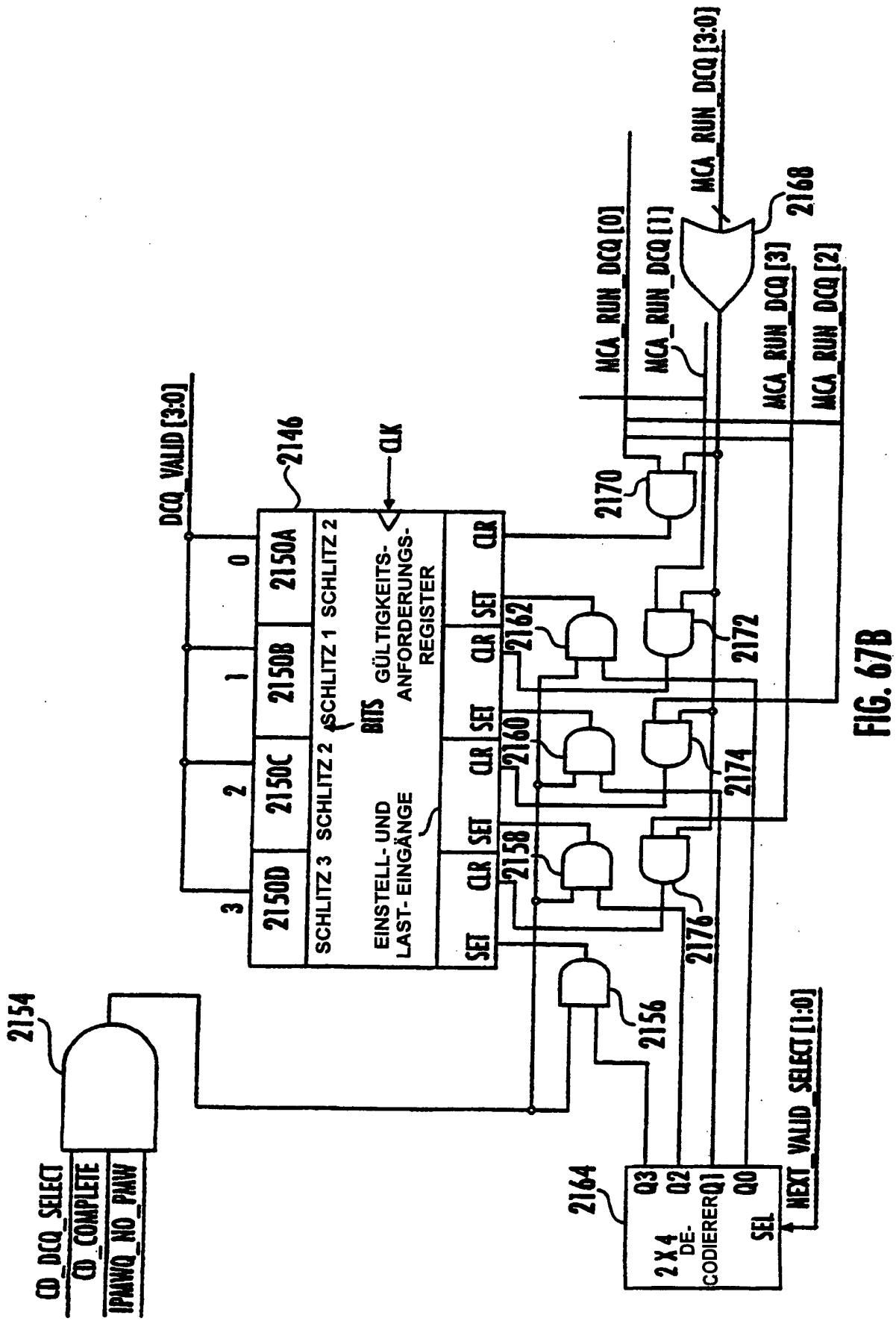

CD_DCQ_SELECT
CD_COMPLETE
!PMWQ_NO_PMW
2154
DCQ_VALID [3:0]
2146
CLK
0
1
2
3
2150A
2150B
2150C
2150D
SCHLITZ 1 SCHLITZ 2
SCHLITZ 3 SCHLITZ 2
BITS
GÜLTIGKEITS-ANFORDERUNGS-REGISTER
EINSTELL- UND LAST-EINGÄNGE
SET
CLR
2158
2160
2162
2156
2164
2 X 4 DE-CODIERER
Q3
Q2
Q1
Q0
SEL
NEXT_VALID_SELECT [1:0]
2170
2172
2174
2176
2168
MCA_RUN_DCQ [0]
MCA_RUN_DCQ [1]
MCA_RUN_DCQ [3]
MCA_RUN_DCQ [2]
MCA_RUN_DCQ [3:0]

FIG. 67B

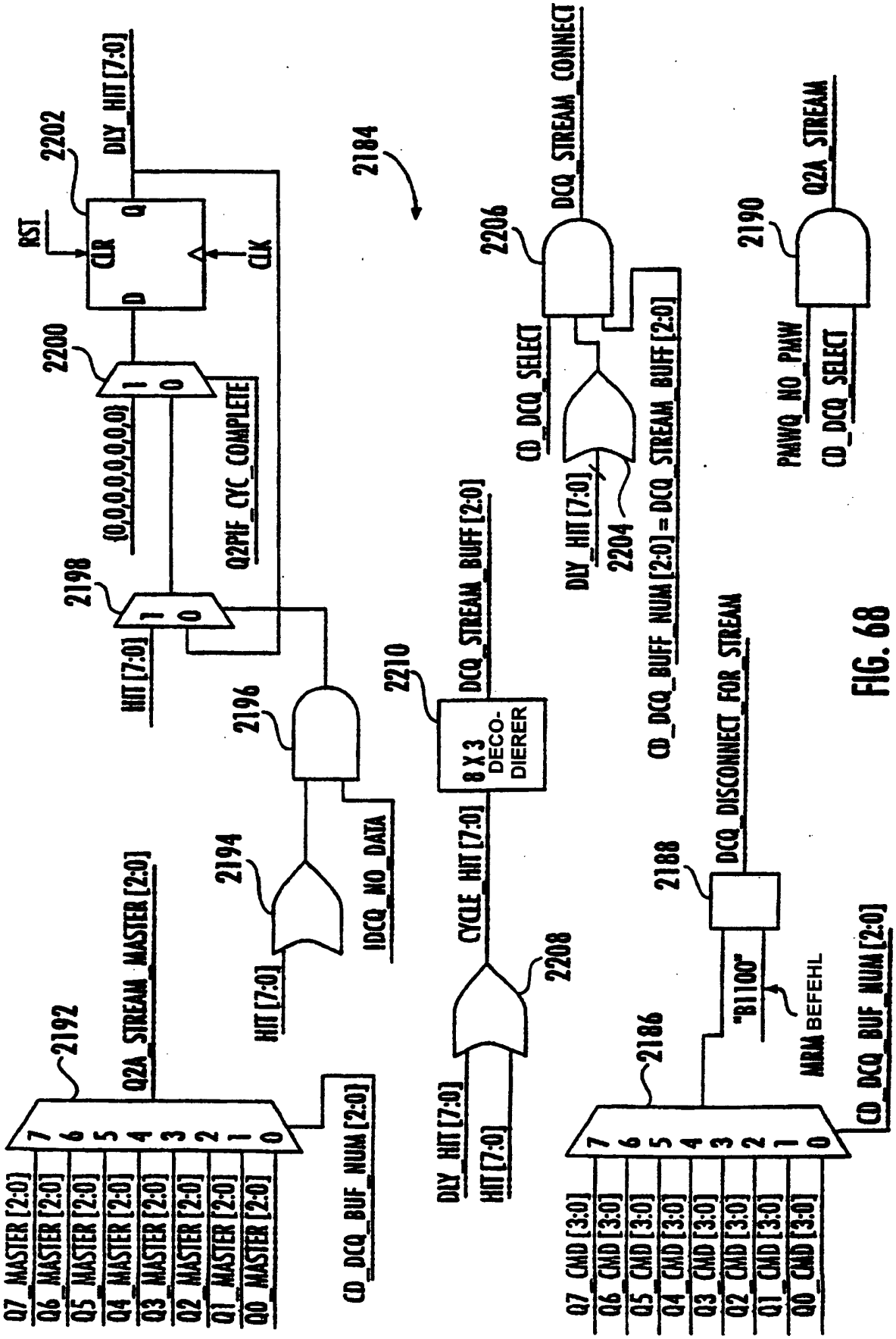

2184
2202
DLY_HIT [7:0]
RST
CLR
CLK
2200
{0,0,0,0,0,0,0,0}
Q2PIF_CYC_COMPLETE
2198
HIT [7:0]
2196
2194
HIT [7:0]
IDCQ_NO_DATA
Q2A_STREAM_MASTER [2:0]
2192
Q7_MASTER [2:0]
Q6_MASTER [2:0]
Q5_MASTER [2:0]
Q4_MASTER [2:0]
Q3_MASTER [2:0]
Q2_MASTER [2:0]
Q1_MASTER [2:0]
Q0_MASTER [2:0]
CD_DCQ_BUF_NUM [2:0]
2210
DCQ_STREAM_BUFF [2:0]
8 X 3 DECO-DIERER
CYCLE_HIT [7:0]
2208
DLY_HIT [7:0]
HIT [7:0]
2206
DCQ_STREAM_CONNECT
CD_DCQ_SELECT
DLY_HIT [7:0]
2204
CD_DCQ_BUFF_NUM [2:0] = DCQ_STREAM_BUFF [2:0]
2190
Q2A_STREAM
PNWQ_NO_PNW
CD_DCQ_SELECT
2188
DCQ_DISCONNECT_FOR_STREAM
"B1100"
MRM BEFEHL
2186
Q7_CMD [3:0]
Q6_CMD [3:0]
Q5_CMD [3:0]
Q4_CMD [3:0]
Q3_CMD [3:0]
Q2_CMD [3:0]
Q1_CMD [3:0]
Q0_CMD [3:0]
CD_DCQ_BUF_NUM [2:0]

FIG. 68

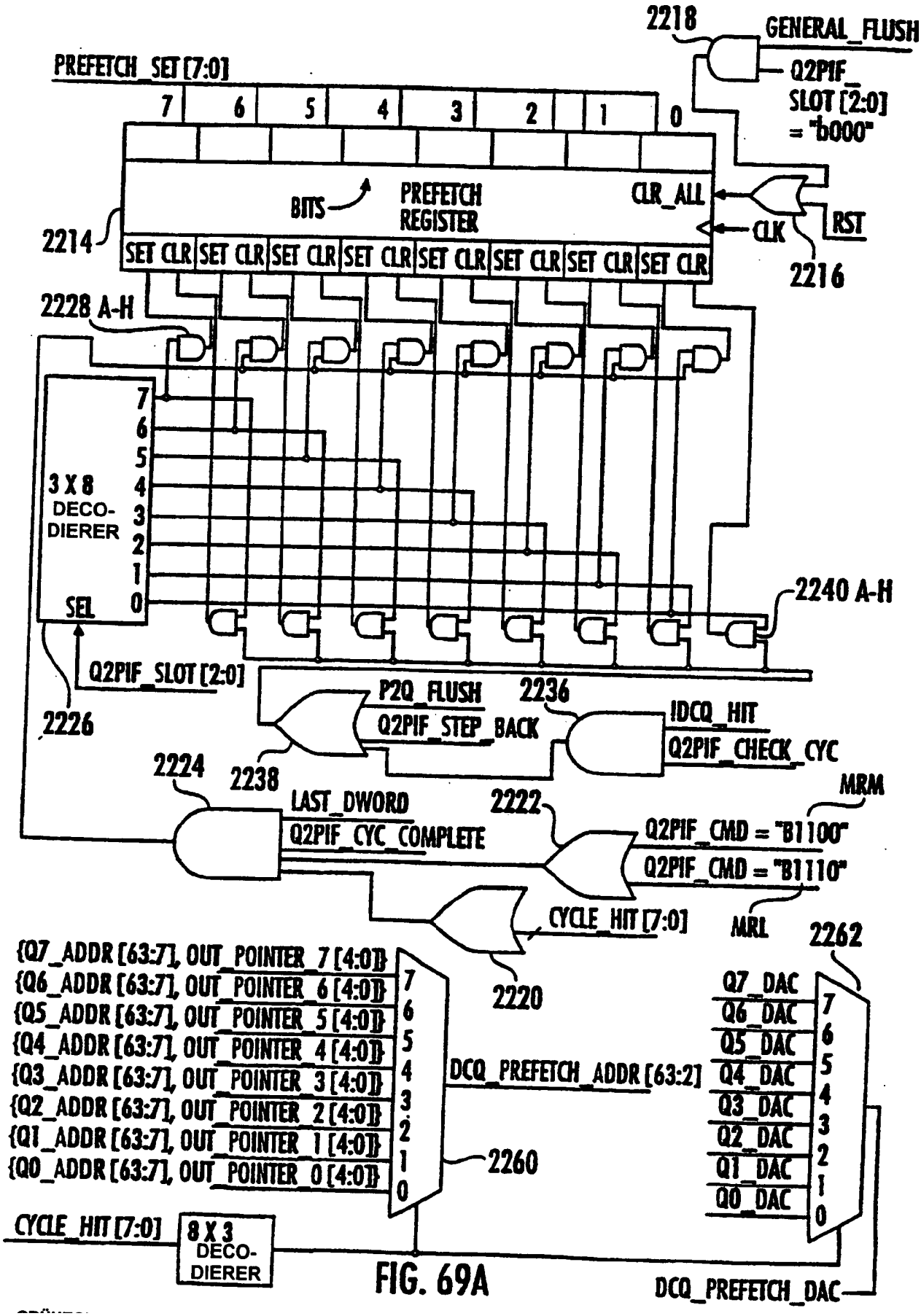

2218
GENERAL_FLUSH
PREFETCH_SET[7:0]
Q2PIF_SLOT[2:0] = "b000"
7 6 5 4 3 2 1 0
BITS
PREFETCH REGISTER
CLR_ALL
CLK
RST
2214
SET CLR SET CLR SET CLR SET CLR SET CLR SET CLR SET CLR SET CLR
2216
2228 A-H
3 X 8 DECO-DIERER
SEL
2240 A-H
Q2PIF_SLOT[2:0]
2226
P2Q_FLUSH
Q2PIF_STEP_BACK
2236
!DCQ_HIT
Q2PIF_CHECK_CYC
2224
2238
LAST_DWORD
Q2PIF_CYC_COMPLETE
2222
MRM
Q2PIF_CMD = "B1100"
Q2PIF_CMD = "B1110"
CYCLE_HIT[7:0]
MRL
2262
2220
{Q7_ADDR[63:7], OUT_POINTER_7[4:0]}
{Q6_ADDR[63:7], OUT_POINTER_6[4:0]}
{Q5_ADDR[63:7], OUT_POINTER_5[4:0]}
{Q4_ADDR[63:7], OUT_POINTER_4[4:0]}
{Q3_ADDR[63:7], OUT_POINTER_3[4:0]}
{Q2_ADDR[63:7], OUT_POINTER_2[4:0]}
{Q1_ADDR[63:7], OUT_POINTER_1[4:0]}
{Q0_ADDR[63:7], OUT_POINTER_0[4:0]}
DCQ_PREFETCH_ADDR[63:2]
2260
Q7_DAC
Q6_DAC
Q5_DAC
Q4_DAC
Q3_DAC
Q2_DAC
Q1_DAC
Q0_DAC
CYCLE_HIT[7:0]
8 X 3 DECO-DIERER
DCQ_PREFETCH_DAC

FIG. 69A

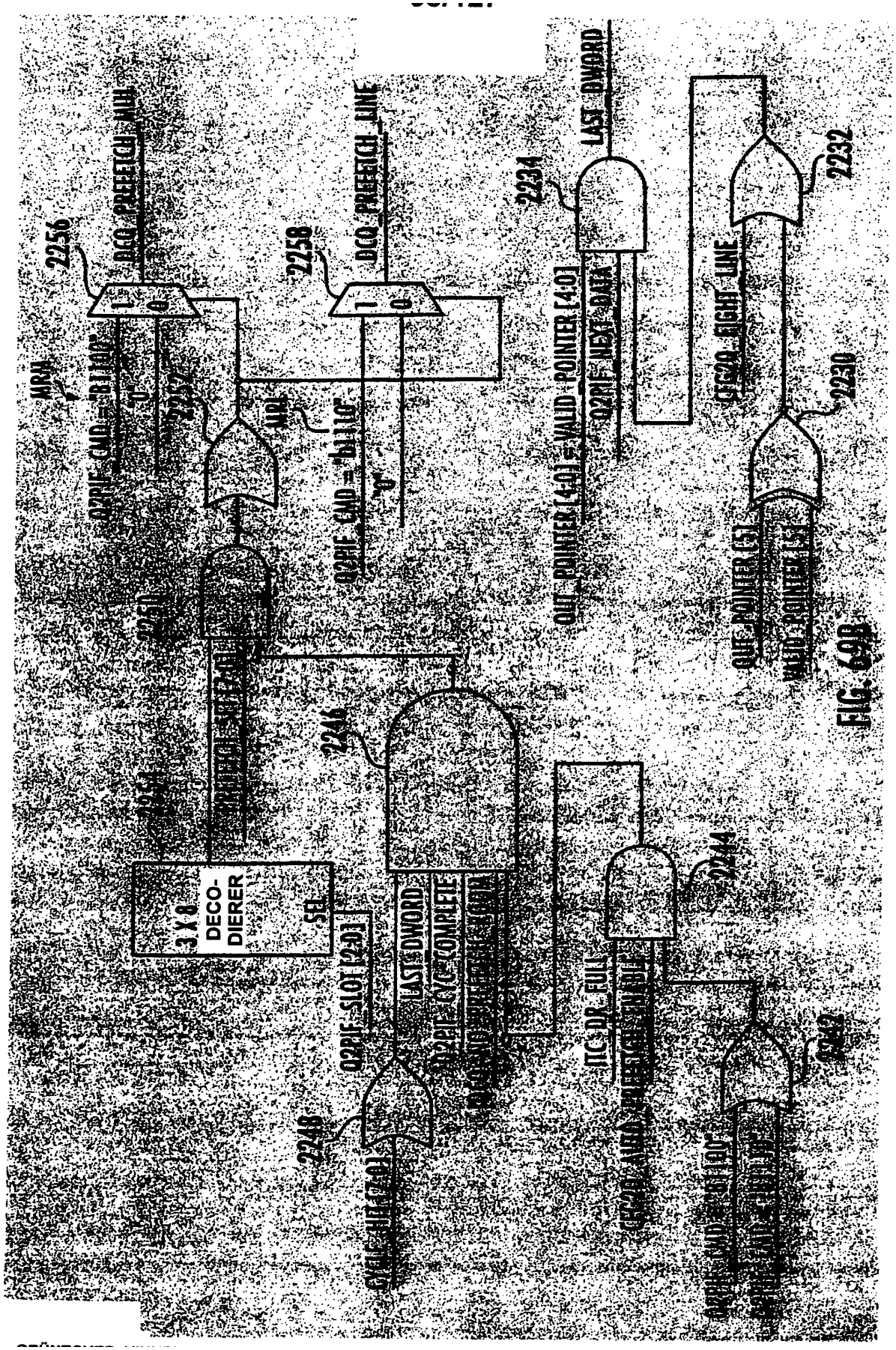

2256
BCQ PREFETCH MUX
Q2RIF_CMD = 5'b11001
2258
BCQ PREFETCH_LINE
Q2RIF_CMD = 5'b11101
2234
LAST_DWORD
OUT_POINTER [4:0] = VALID_POINTER [4:0]
Q2RIF_NEXT_DATA
2232
QF_POINTER [5]
2230
3 X 8 DECO-DIERER
SEL
Q2RIF_SLOT [2:0]
LAST_DWORD
Q2RIF_EXT_COMPLETE
2248
2246
2244
ITC_DP_FULL

FIG. 69B

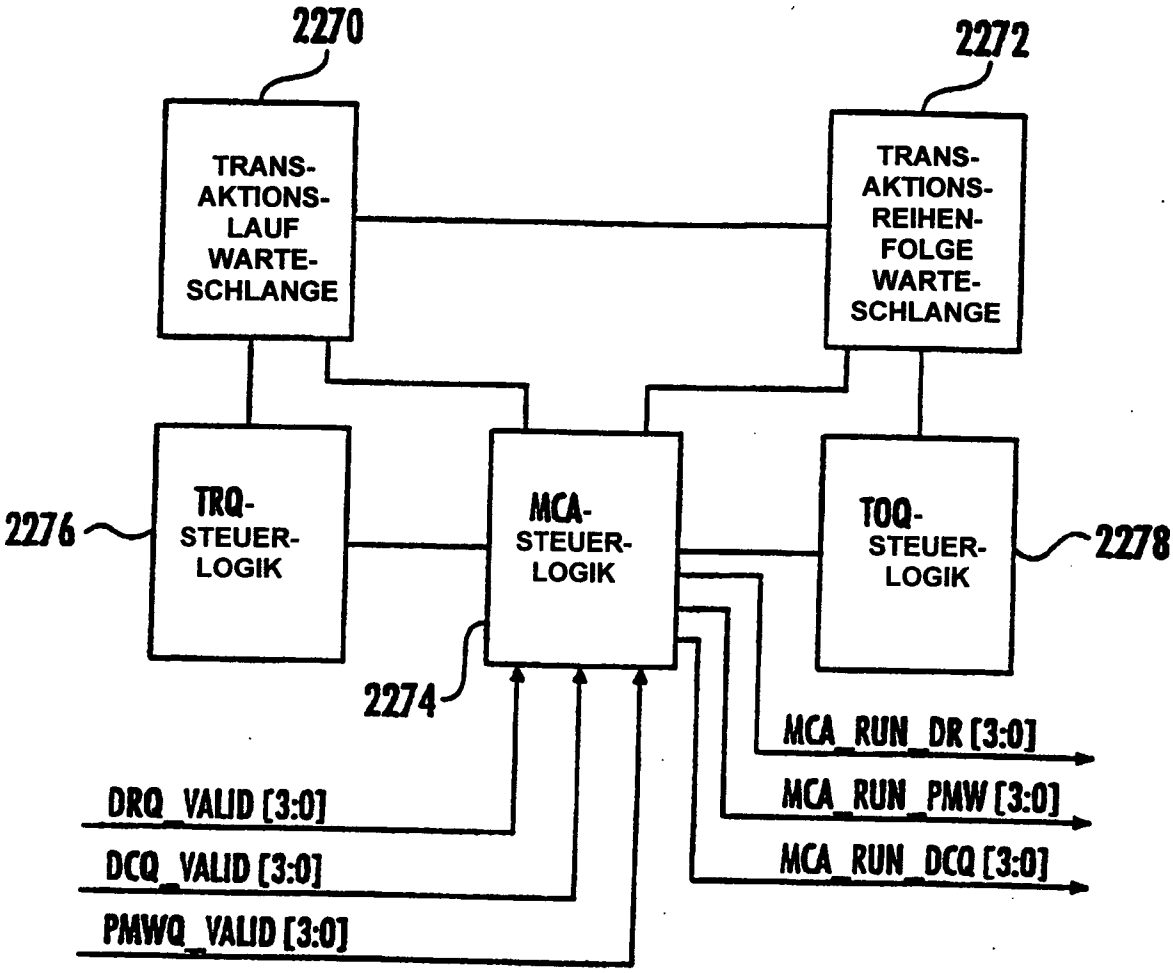

2270
2272
TRANS-
AKTIONS-
LAUF
WARTE-
SCHLANGE
TRANS-
AKTIONS-
REIHEN-
FOLGE
WARTE-
SCHLANGE
2276
TRQ-
STEUER-
LOGIK
MCA-
STEUER-
LOGIK
TOQ-
STEUER-
LOGIK
2278
2274
MCA_RUN_DR [3:0]
MCA_RUN_PMW [3:0]
MCA_RUN_DCQ [3:0]
DRQ_VALID [3:0]
DCQ_VALID [3:0]
PMWQ_VALID [3:0]

FIG. 70

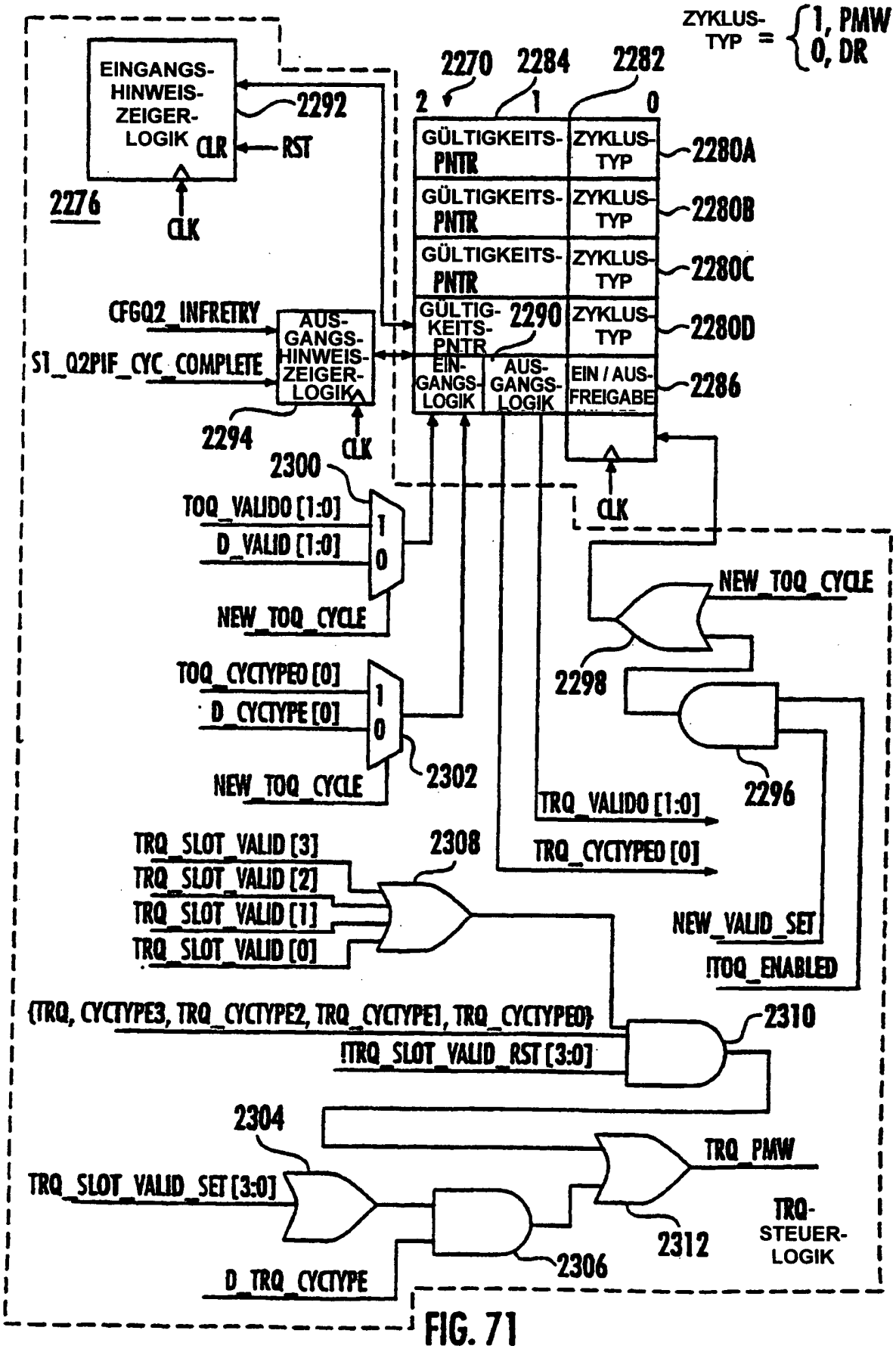

ZYKLUS-TYP = {1, PMW; 0, DR
EINGANGS-HINWEIS-ZEIGER-LOGIK
2292
CLR
RST
2276
CLK
2270
2284
2282
2
1
0
GÜLTIGKEITS-PNTR
ZYKLUS-TYP
2280A
2280B
2280C
2280D
2290
EIN-GANGS-LOGIK
AUS-GANGS-LOGIK
EIN / AUS-FREIGABE
2286
CFGQ2_INFRETRY
S1_Q2PIF_CYC_COMPLETE
AUS-GANGS-HINWEIS-ZEIGER-LOGIK
2294
2300
TOQ_VALID0 [1:0]
D_VALID [1:0]
NEW_TOQ_CYCLE
TOQ_CYCTYPE0 [0]
D_CYCTYPE [0]
2302
2298
2296
TRQ_VALID0 [1:0]
TRQ_CYCTYPE0 [0]
2308
TRQ_SLOT_VALID [3]
TRQ_SLOT_VALID [2]
TRQ_SLOT_VALID [1]
TRQ_SLOT_VALID [0]
NEW_VALID_SET
!TOQ_ENABLED
{TRQ_CYCTYPE3, TRQ_CYCTYPE2, TRQ_CYCTYPE1, TRQ_CYCTYPE0}
!TRQ_SLOT_VALID_RST [3:0]
2310
2304
TRQ_SLOT_VALID_SET [3:0]
TRQ_PMW
2312
TRQ-STEUER-LOGIK
2306
D_TRQ_CYCTYPE

FIG. 71

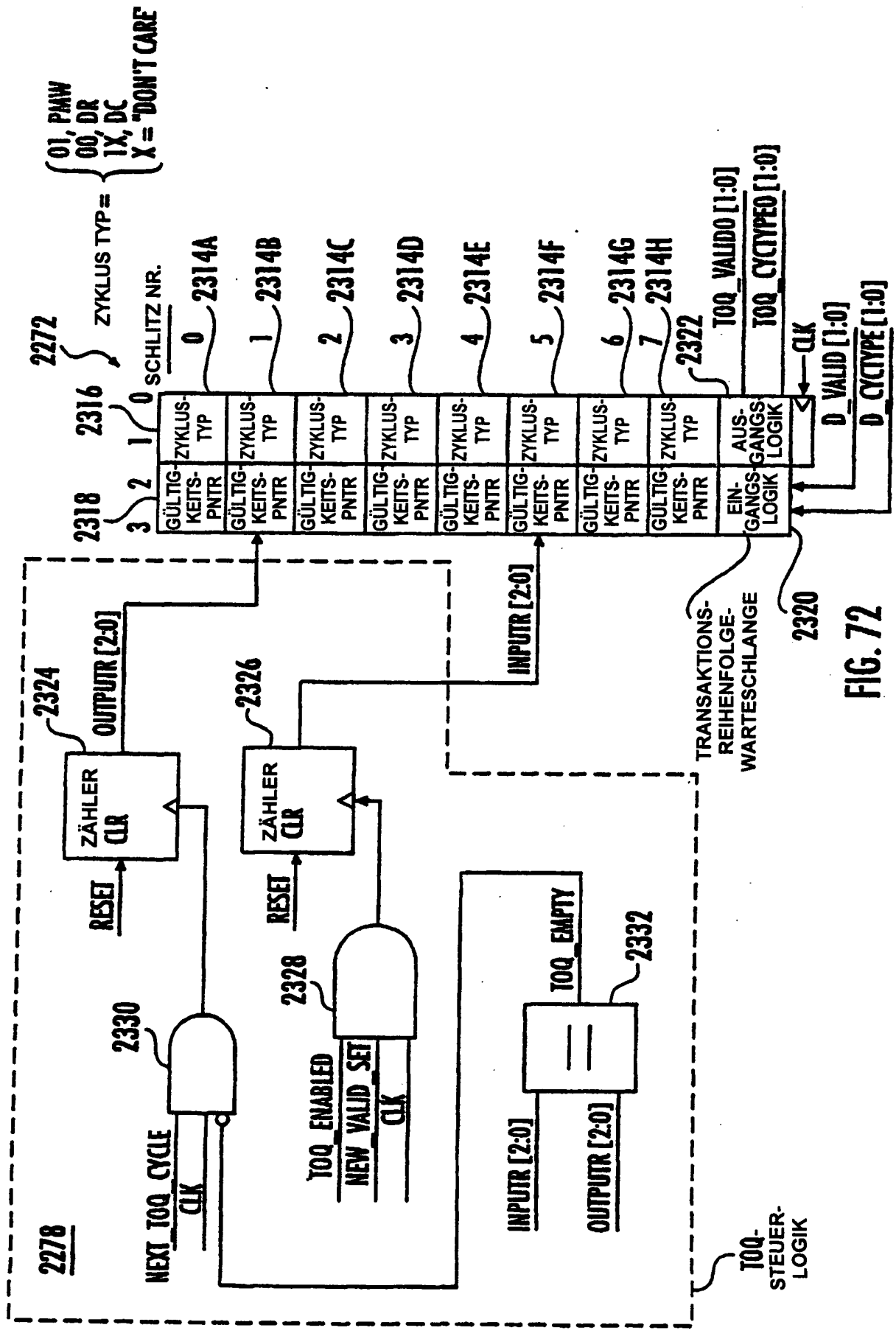

2272
ZYKLUS TYP=
01, PMW
00, DR
1X, DC
X = "DON'T CARE"
SCHLITZ NR.
2316
2318
0
1
2
3
2314A
2314B
2314C
2314D
2314E
2314F
2314G
2314H
2322
GÜLTIG-KEITS-PNTR
ZYKLUS-TYP
EIN-GANGS-LOGIK
AUS-GANGS-LOGIK
TOQ_VALIDO [1:0]
TOQ_CYCTYPEO [1:0]
CLK
D_VALID [1:0]
D_CYCTYPE [1:0]
2320
TRANSAKTIONS-REIHENFOLGE-WARTESCHLANGE
INPUTR [2:0]
2324
OUTPUTR [2:0]
2326
ZÄHLER
CLR
RESET
2328
2330
TOQ_ENABLED
NEW_VALID_SET
CLK
NEXT_TOQ_CYCLE
2332
TOQ_EMPTY
INPUTR [2:0]
OUTPUTR [2:0]
2278
TOQ-STEUER-LOGIK

FIG. 72

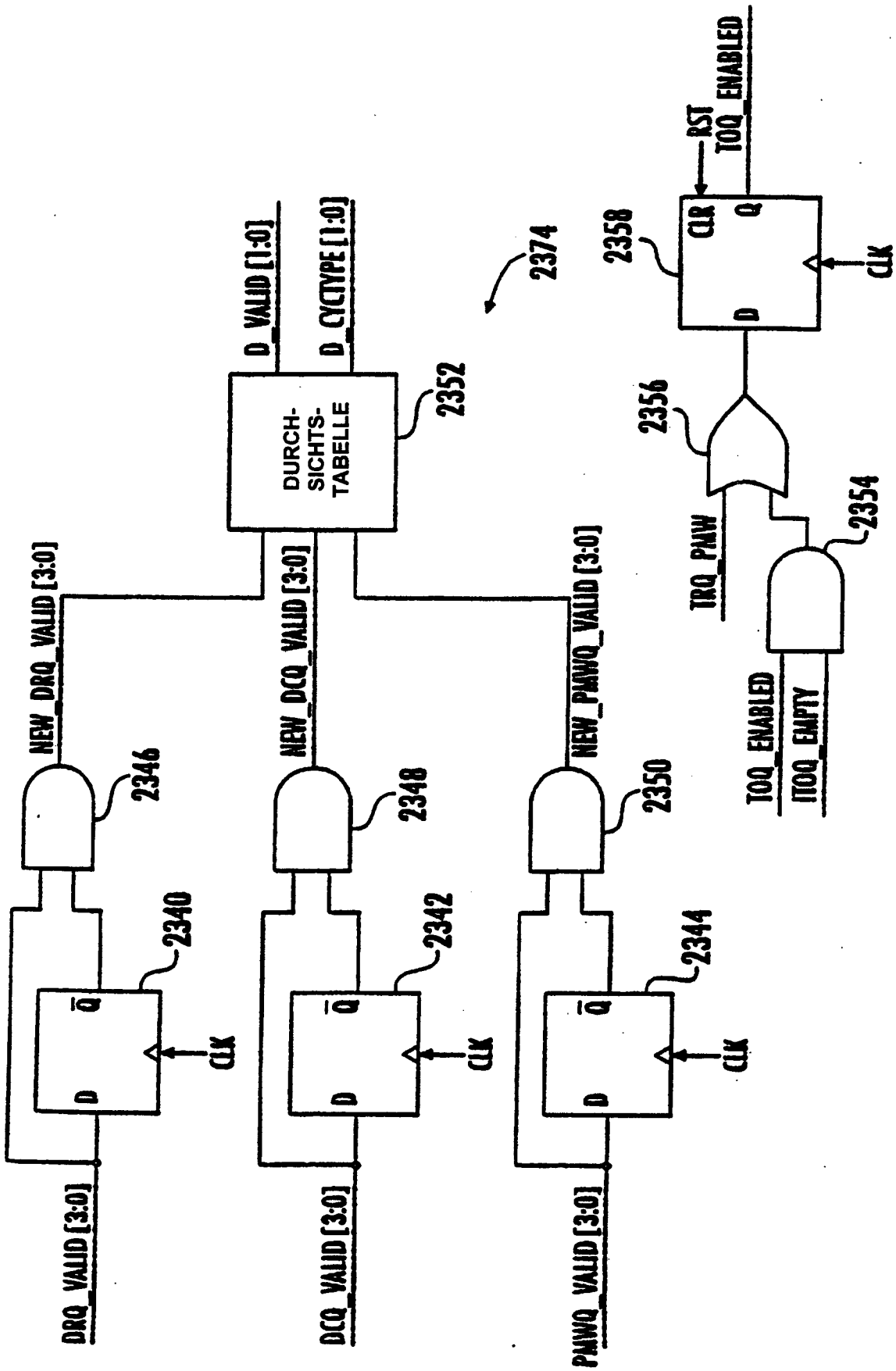

D_VALID [1:0]
D_CYCTYPE [1:0]
2374
2358
RST
TOQ_ENABLED
CLR
Q
D
CLK
2352
DURCH-
SICHTS-
TABELLE
2356
2354
TRQ_PNW
TOQ_ENABLED
ITOQ_EMPTY
NEW_DRQ_VALID [3:0]
NEW_DCQ_VALID [3:0]
NEW_PNWQ_VALID [3:0]
2346
2348
2350
2340
2342
2344
Q̄
DRQ_VALID [3:0]
DCQ_VALID [3:0]
PNWQ_VALID [3:0]

FIG. 73A

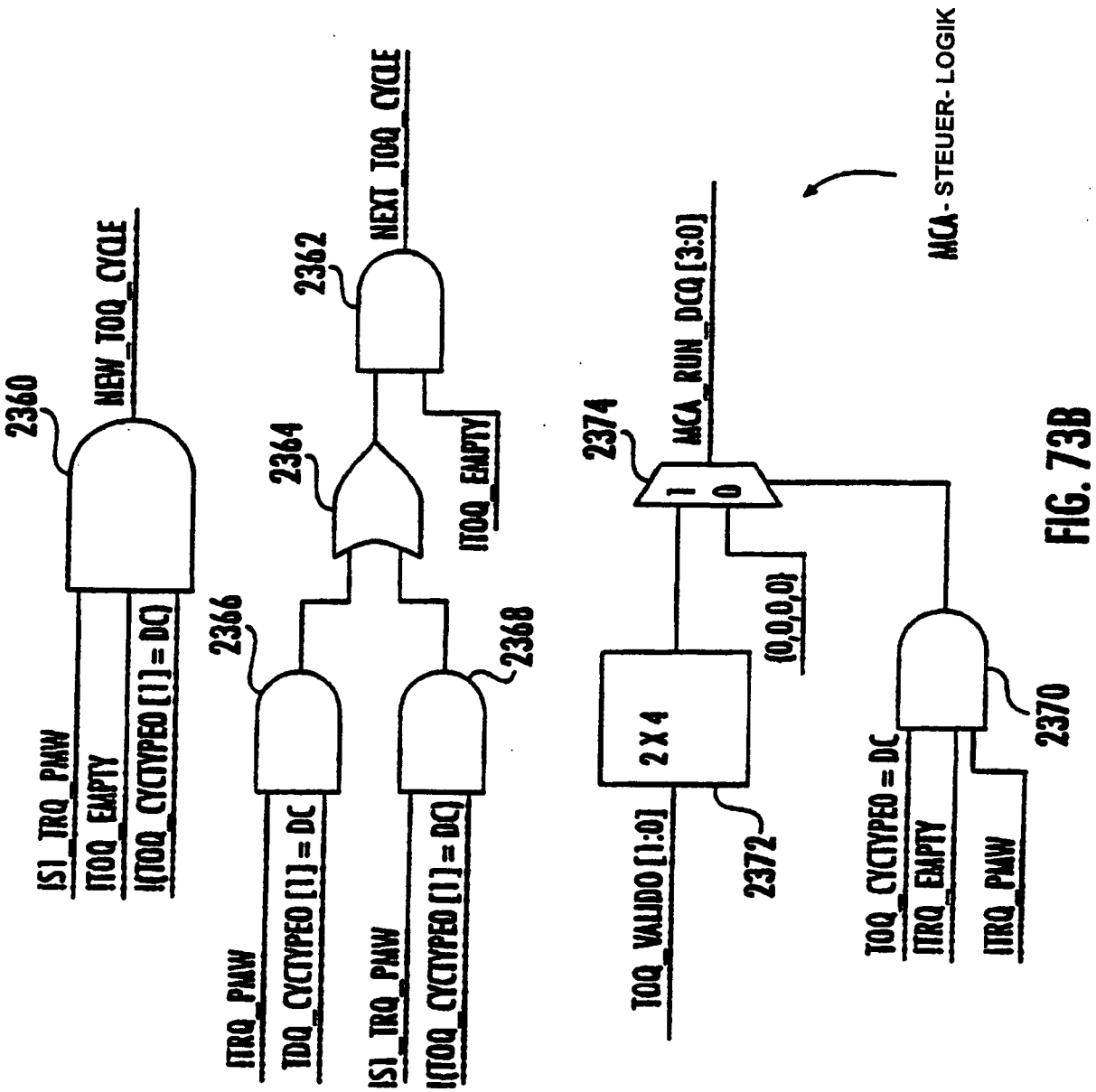

MCA-STEUER-LOGIK
2360
NEW_TOQ_CYCLE
!S1_TRQ_PNW
!TOQ_EMPTY
!(TOQ_CYCTYPE0[1]=DC)
2362
NEXT_TOQ_CYCLE
2364
!TOQ_EMPTY
2366
!TRQ_PNW
TDQ_CYCTYPE0[1]=DC
2368
S1_TRQ_PNW
!(TOQ_CYCTYPE0[1]=DC)
2374
MCA_RUN_DCQ[3:0]
1
0
(0,0,0,0)
2X4
TOQ_VALUDO[1:0]
2372
TOQ_CYCTYPE0=DC
!TRQ_EMPTY
!TRQ_PNW
2370

FIG. 73B

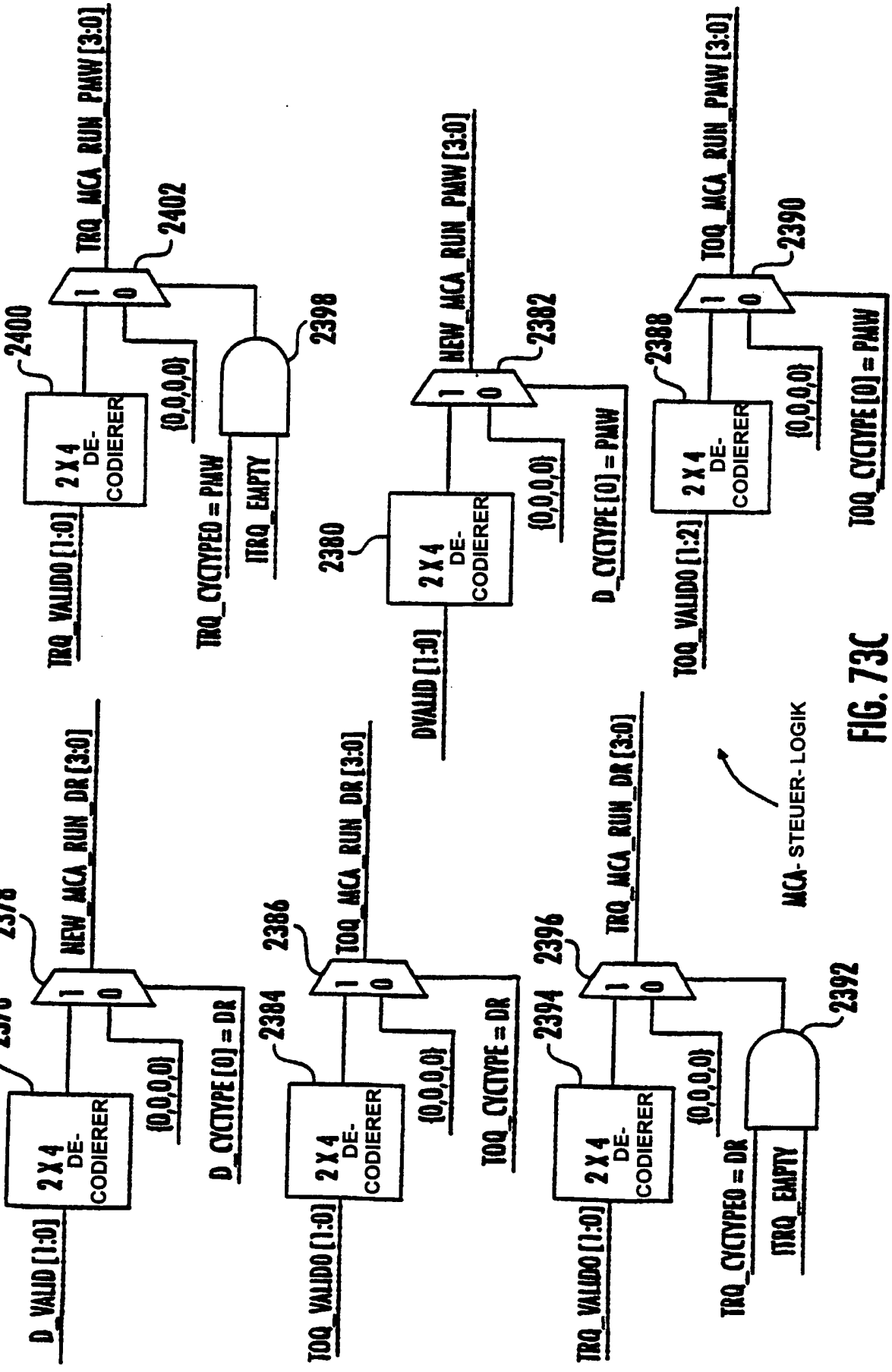

2400
2 X 4 DE-CODIERER
TRQ_VALIDO [1:0]
{0,0,0,0}
TRQ_CYCTYPE0 = PNW
ITRQ_EMPTY
2398
2402
TRQ_MCA_RUN_PNW [3:0]
2380
2 X 4 DE-CODIERER
DVALID [1:0]
{0,0,0,0}
D_CYCTYPE [0] = PNW
2382
NEW_MCA_RUN_PNW [3:0]
2388
2 X 4 DE-CODIERER
TOQ_VALIDO [1:2]
{0,0,0,0}
TOQ_CYCTYPE [0] = PNW
2390
TOQ_MCA_RUN_PNW [3:0]
2376
2 X 4 DE-CODIERER
D_VALID [1:0]
{0,0,0,0}
D_CYCTYPE [0] = DR
2378
NEW_MCA_RUN_DR [3:0]
2384
2 X 4 DE-CODIERER
TOQ_VALIDO [1:0]
{0,0,0,0}
TOQ_CYCTYPE = DR
2386
TOQ_MCA_RUN_DR [3:0]
2394
2 X 4 DE-CODIERER
TRQ_VALIDO [1:0]
{0,0,0,0}
TRQ_CYCTYPE0 = DR
ITRQ_EMPTY
2392
2396
TRQ_MCA_RUN_DR [3:0]
MCA- STEUER- LOGIK

FIG. 73C

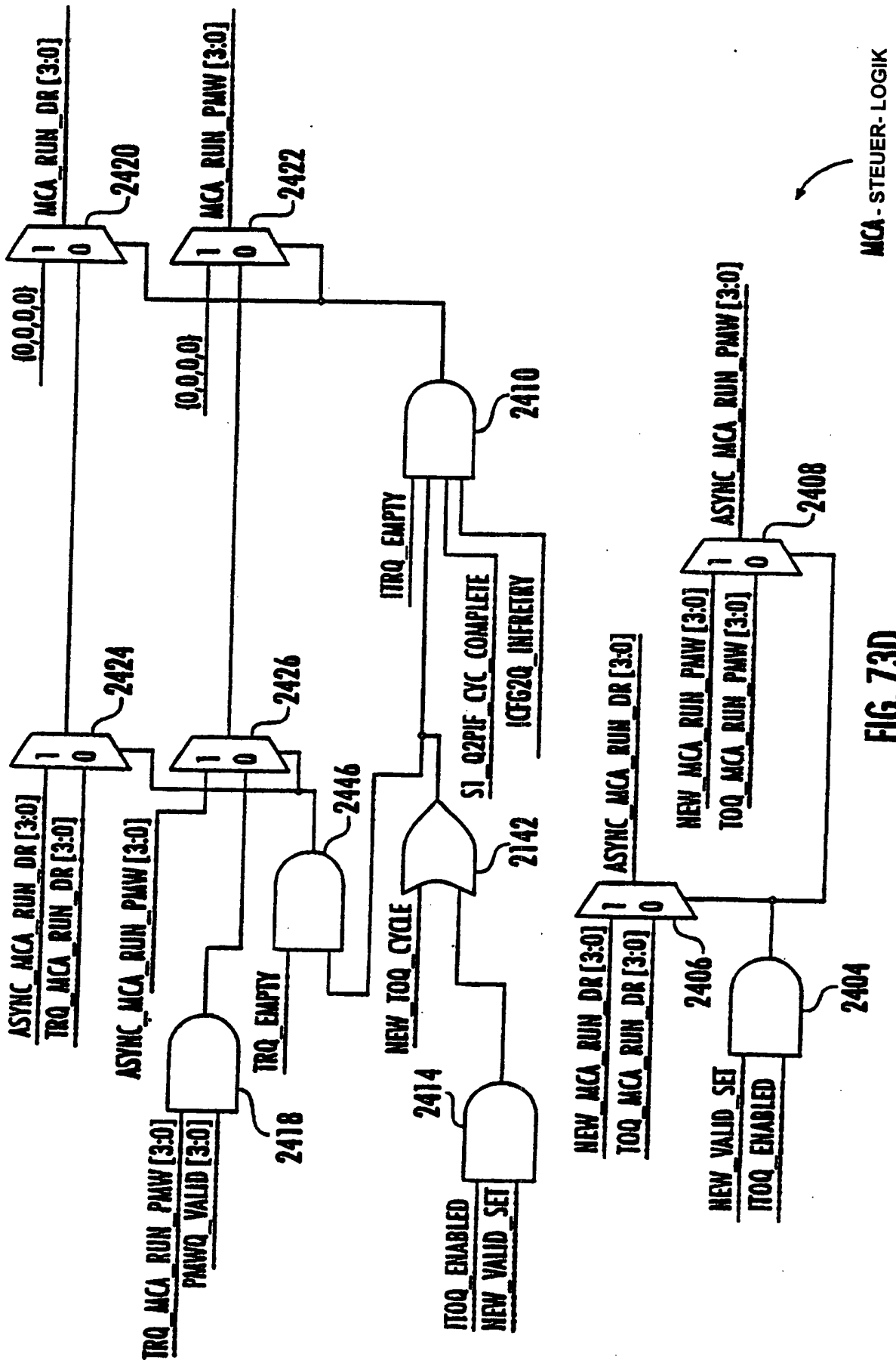

MCA - STEUER- LOGIK
MCA_RUN_DR [3:0]
2420
MCA_RUN_PNW [3:0]
2422
{0,0,0,0}
{0,0,0,0}
2410
!TRQ_EMPTY
S1_Q2PIF_CYC_COMPLETE
!CFG2Q_INFRETRY
2424
2426
2446
2142
ASYNC_MCA_RUN_DR [3:0]
TRQ_MCA_RUN_DR [3:0]
ASYNC_MCA_RUN_PNW [3:0]
TRQ_EMPTY
NEW_TOQ_CYCLE
2414
2418
TRQ_MCA_RUN_PNW [3:0]
PNWTQ_VALID [3:0]
!TOQ_ENABLED
NEW_VALID_SET
ASYNC_MCA_RUN_PNW [3:0]
2408
NEW_MCA_RUN_PNW [3:0]
TOQ_MCA_RUN_PNW [3:0]
ASYNC_MCA_RUN_DR [3:0]
NEW_MCA_RUN_DR [3:0]
TOQ_MCA_RUN_DR [3:0]
2406
2404
NEW_VALID_SET
!TOQ_ENABLED

FIG. 73D

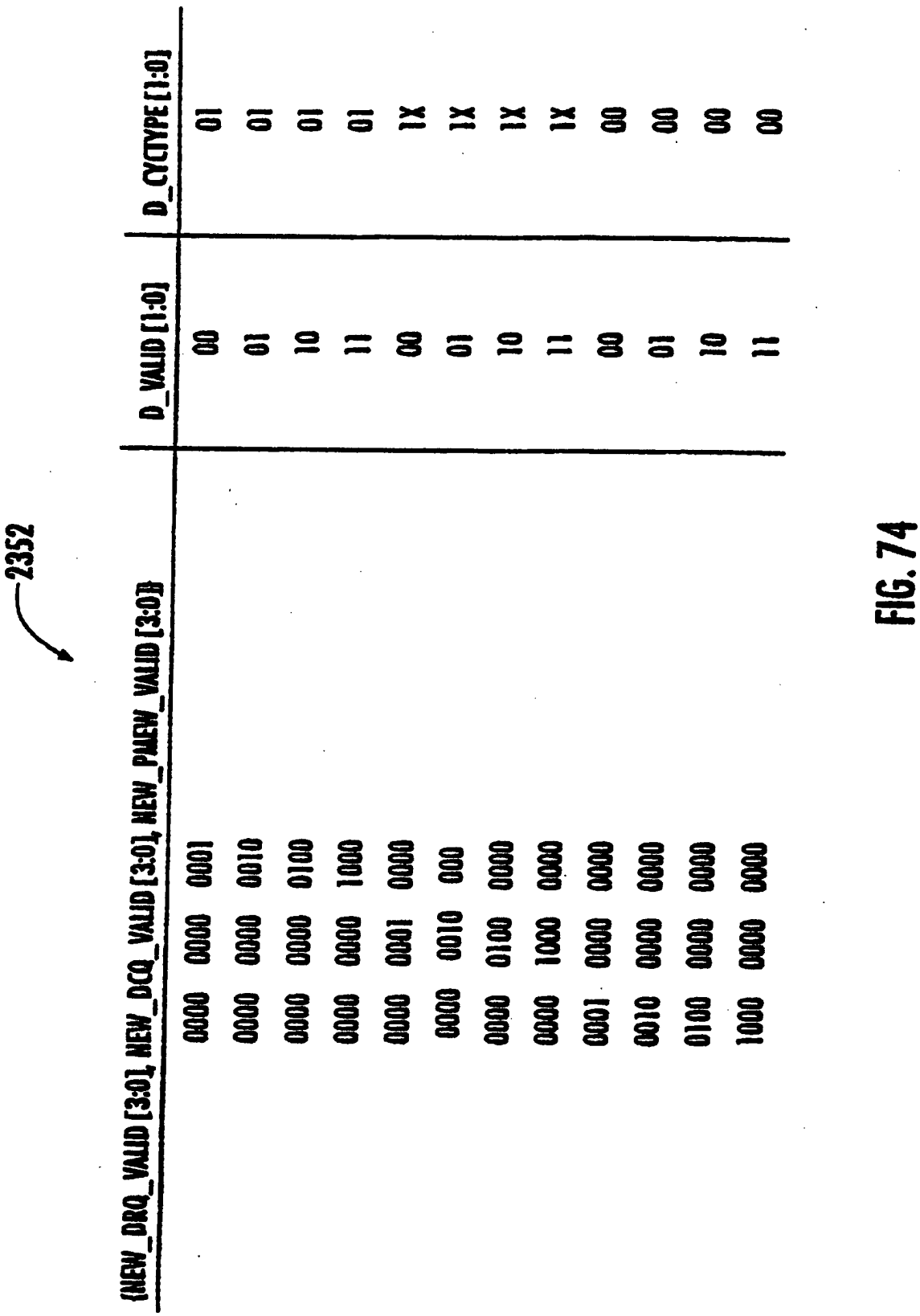

2352
{NEW_DRQ_VALID [3:0], NEW_DCQ_VALID [3:0], NEW_PNEW_VALID [3:0]}
D_VALID [1:0]
D_CYCTYPE [1:0]
0000 0000 0001 00 01
0000 0000 0010 01 01
0000 0000 0100 10 01
0000 0000 1000 11 01
0000 0001 0000 00 1X
0000 0010 000 01 1X
0000 0100 0000 10 1X
0000 1000 0000 11 1X
0001 0000 0000 00 00
0010 0000 0000 01 00
0100 0000 0000 10 00
1000 0000 0000 11 00

FIG. 74

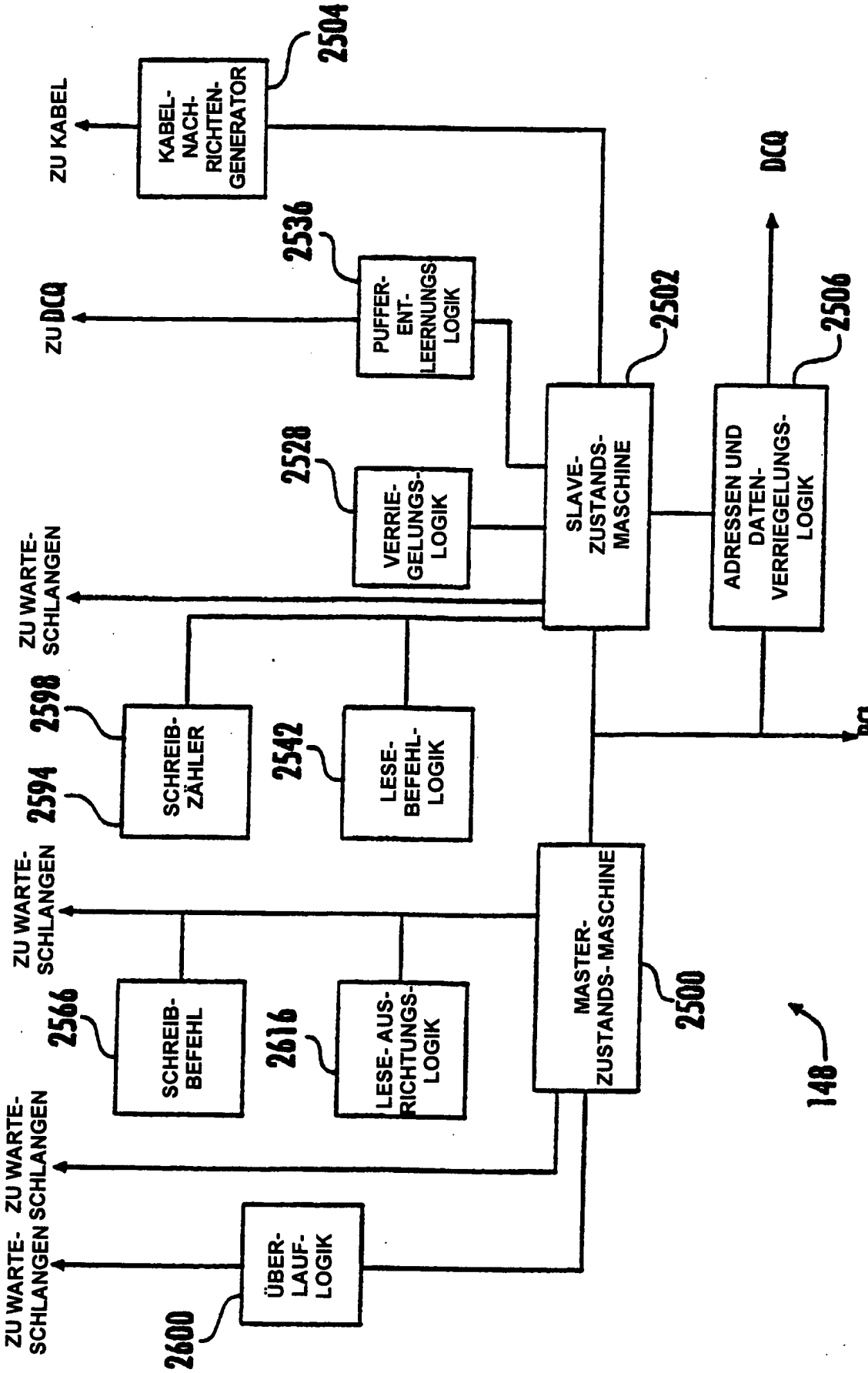

ZU KABEL
KABEL-NACH-RICHTEN-GENERATOR
2504
ZU DCQ
2536
PUFFER-ENT-LEERUNGS-LOGIK
DCQ
2502
2506
SLAVE-ZUSTANDS-MASCHINE
ADRESSEN UND DATEN-VERRIEGELUNGS-LOGIK
2528
VERRIE-GELUNGS-LOGIK
ZU WARTE-SCHLANGEN
2598
2594
SCHREIB-ZÄHLER
2542
LESE-BEFEHL-LOGIK
ZU PCI
ZU WARTE-SCHLANGEN
2566
SCHREIB-BEFEHL
2616
LESE-AUS-RICHTUNGS-LOGIK
MASTER-ZUSTANDS-MASCHINE
2500
148
ZU WARTE-SCHLANGEN
ZU WARTE-SCHLANGEN
2600
ÜBER-LAUF-LOGIK

FIG. 75

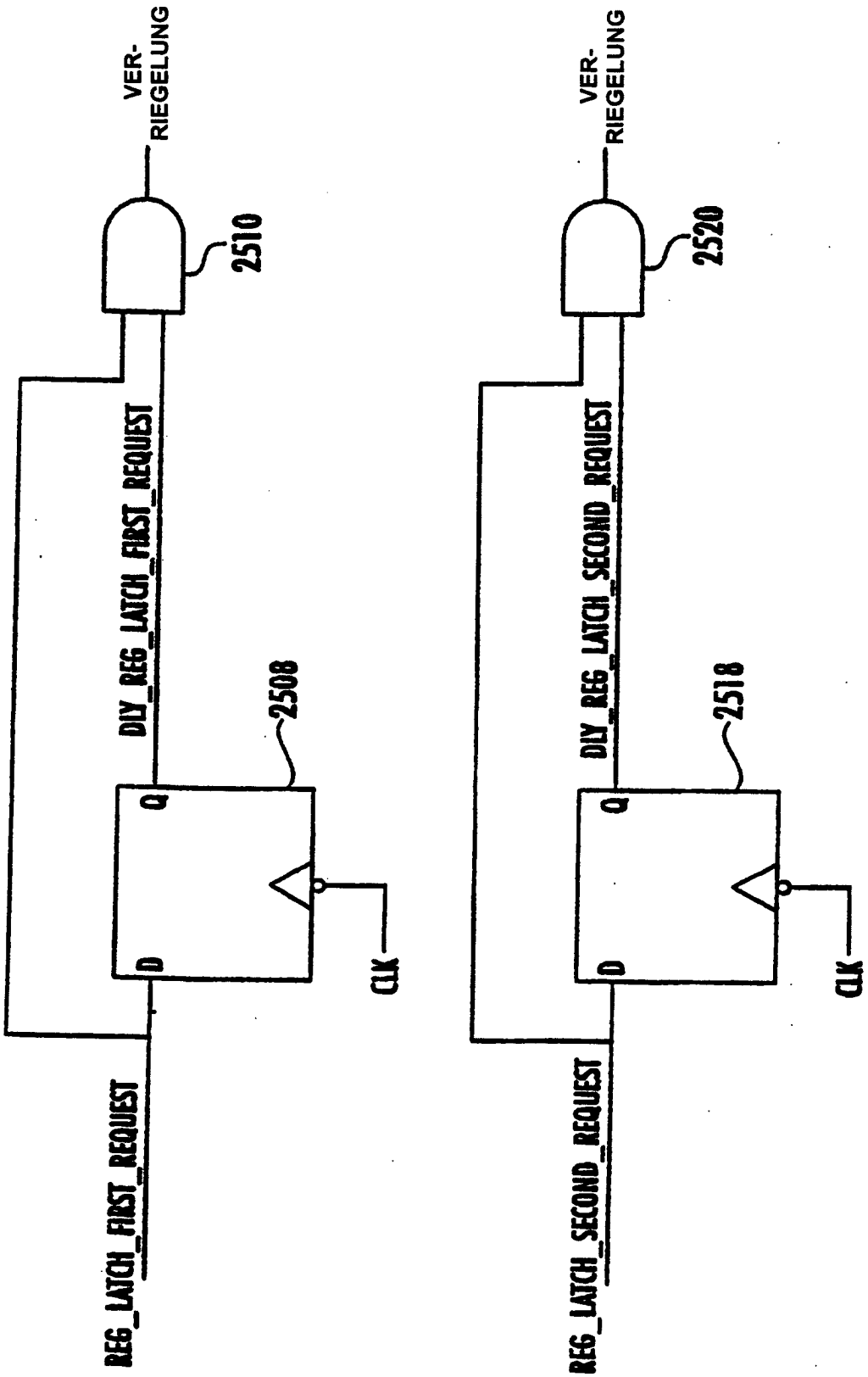

VER-
RIEGELUNG
2510
DLY_REG_LATCH_FIRST_REQUEST
2508
Q
D
CLK
REG_LATCH_FIRST_REQUEST
VER-
RIEGELUNG
2520
DLY_REG_LATCH_SECOND_REQUEST
2518
Q
D
CLK
REG_LATCH_SECOND_REQUEST

FIG. 76A

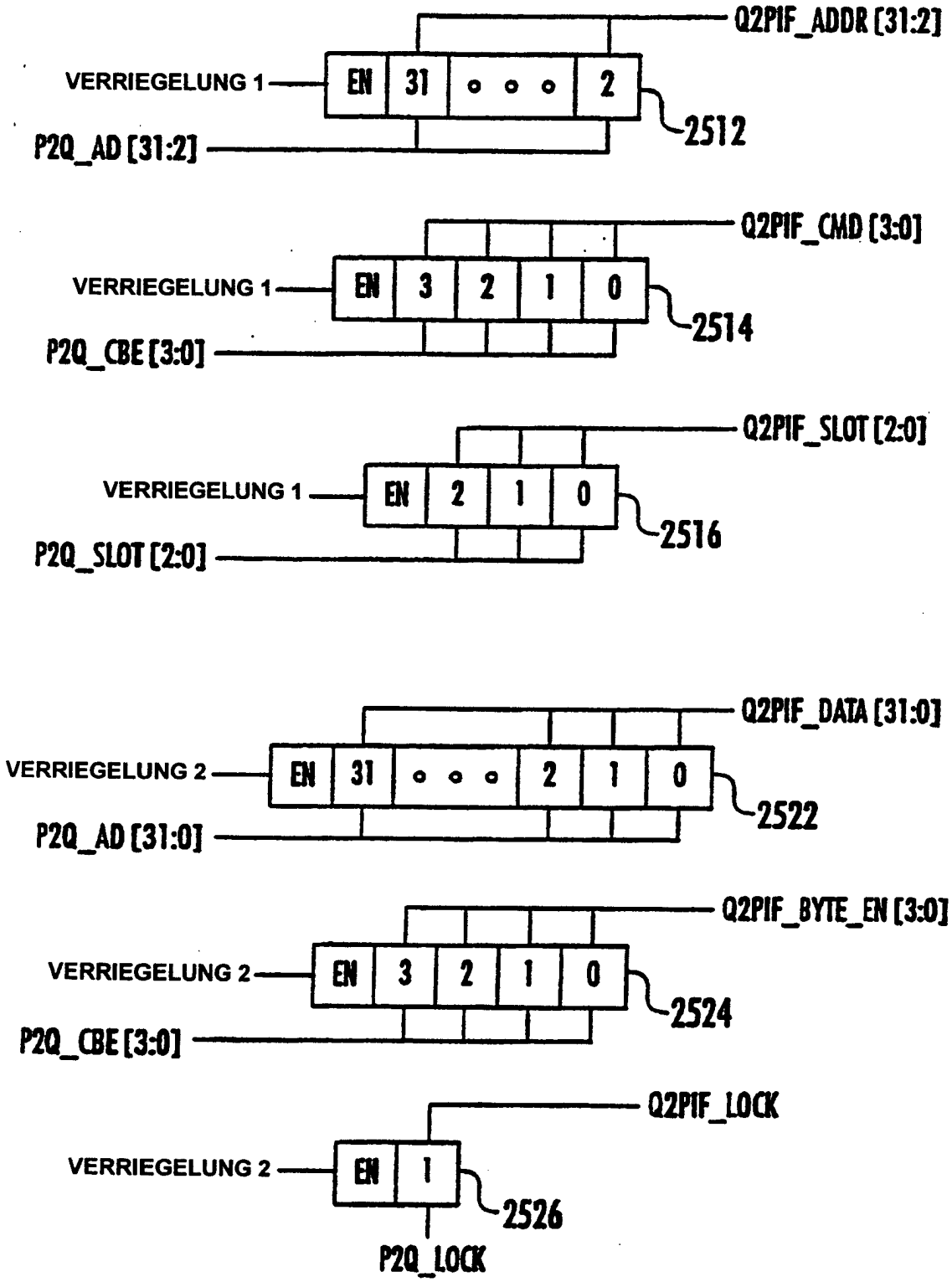

Q2PIF_ADDR [31:2]
VERRIEGELUNG 1
EN
31
2
2512
P2Q_AD [31:2]
Q2PIF_CMD [3:0]
VERRIEGELUNG 1
EN
3
2
1
0
2514
P2Q_CBE [3:0]
Q2PIF_SLOT [2:0]
VERRIEGELUNG 1
EN
2
1
0
2516
P2Q_SLOT [2:0]
Q2PIF_DATA [31:0]
VERRIEGELUNG 2
EN
31
2
1
0
2522
P2Q_AD [31:0]
Q2PIF_BYTE_EN [3:0]
VERRIEGELUNG 2
EN
3
2
1
0
2524
P2Q_CBE [3:0]
Q2PIF_LOCK
VERRIEGELUNG 2
EN
1
2526
P2Q_LOCK

FIG. 76B

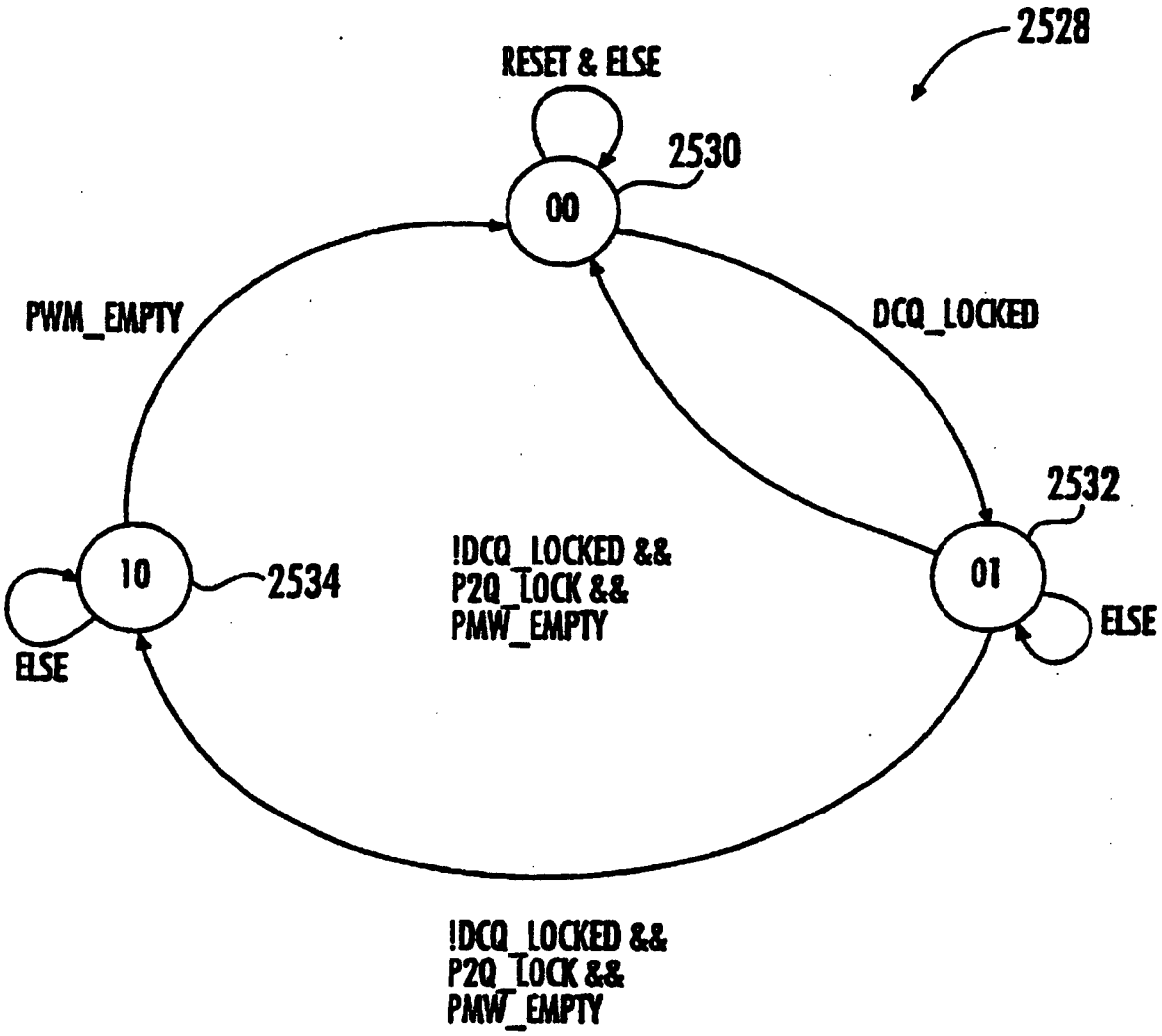

2528
RESET & ELSE
2530
00
PWM_EMPTY
DCQ_LOCKED
2532
10
2534
!DCQ_LOCKED &&
P2Q_LOCK &&
PMW_EMPTY
01
ELSE
ELSE
!DCQ_LOCKED &&
P2Q_LOCK &&
PMW_EMPTY

FIG. 77

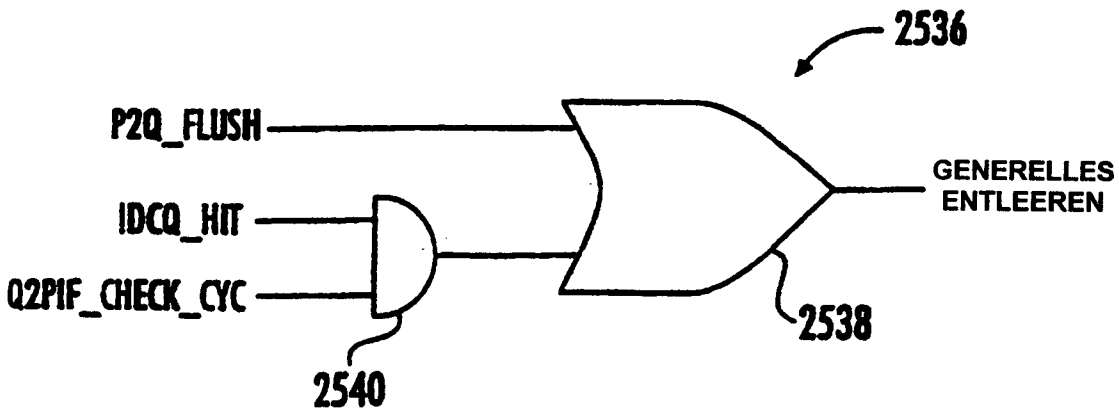

2536
P2Q_FLUSH
!DCQ_HIT
Q2P1F_CHECK_CYC
GENERELLES
ENTLEEREN
2538
2540

FIG. 78

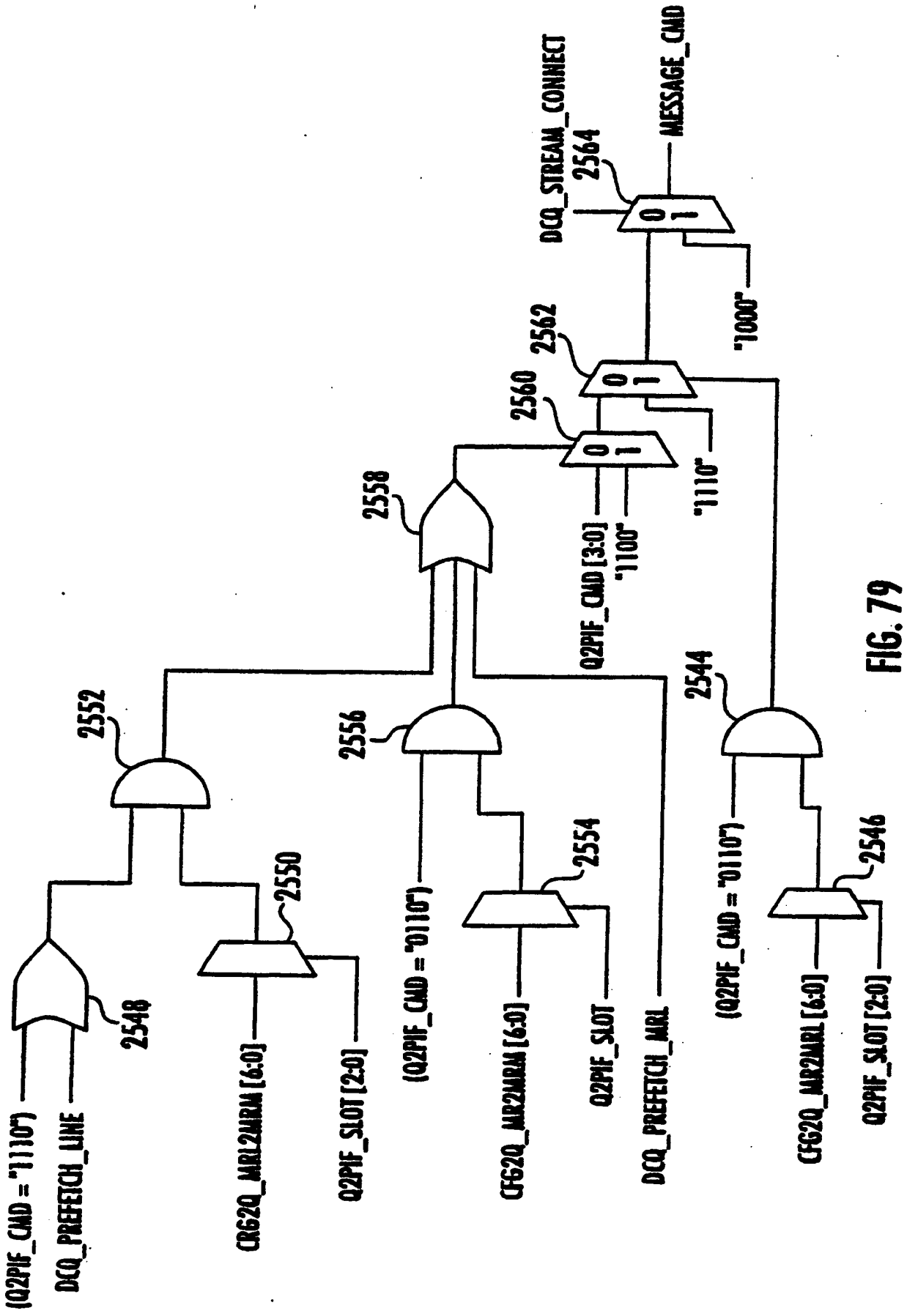

(Q2PIF_CMD = "1110")
DCQ_PREFETCH_LINE
2548
CR620_MRL2MRUM [6:0]
Q2PIF_SLOT [2:0]
2550
2552
(Q2PIF_CMD = "0110")
CR620_MR2MRUM [6:0]
Q2PIF_SLOT
2554
2556
2558
Q2PIF_CMD [3:0]
"1100"
2560
"1110"
2562
DCQ_STREAM_CONNECT
2564
MESSAGE_CMD
"1000"
DCQ_PREFETCH_MRL
(Q2PIF_CMD = "0110")
CR620_MR2MRL [6:0]
Q2PIF_SLOT [2:0]
2544
2546

FIG. 79

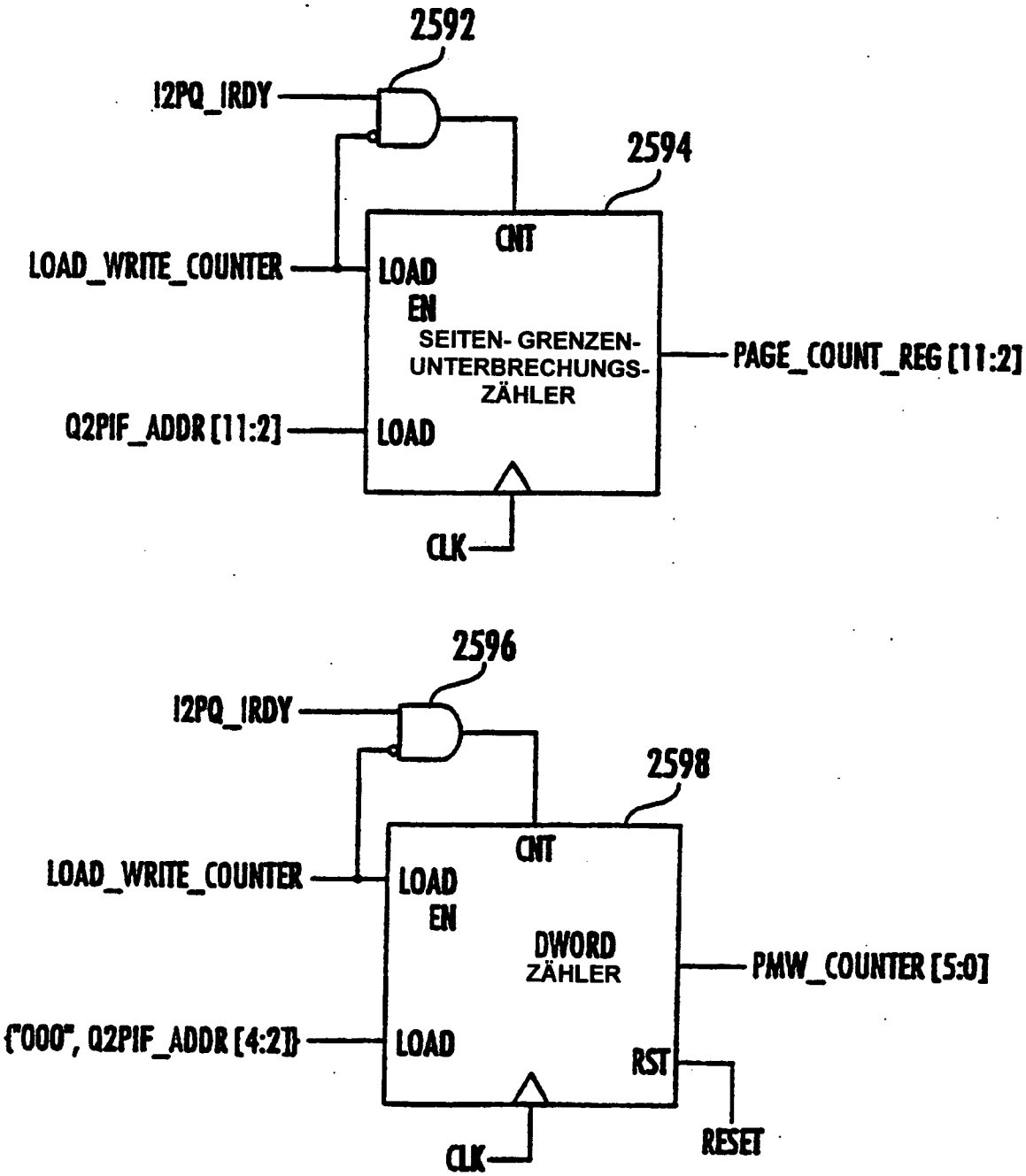

2592
I2PQ_IRDY
2594
CNT
LOAD_WRITE_COUNTER
LOAD
EN
SEITEN- GRENZEN-
UNTERBRECHUNGS-
ZÄHLER
PAGE_COUNT_REG [11:2]
Q2PIF_ADDR [11:2]
LOAD
CLK
2596
I2PQ_IRDY
2598
CNT
LOAD_WRITE_COUNTER
LOAD
EN
DWORD
ZÄHLER
PMW_COUNTER [5:0]
{"000", Q2PIF_ADDR [4:2]}
LOAD
RST
CLK
RESET

FIG. 80

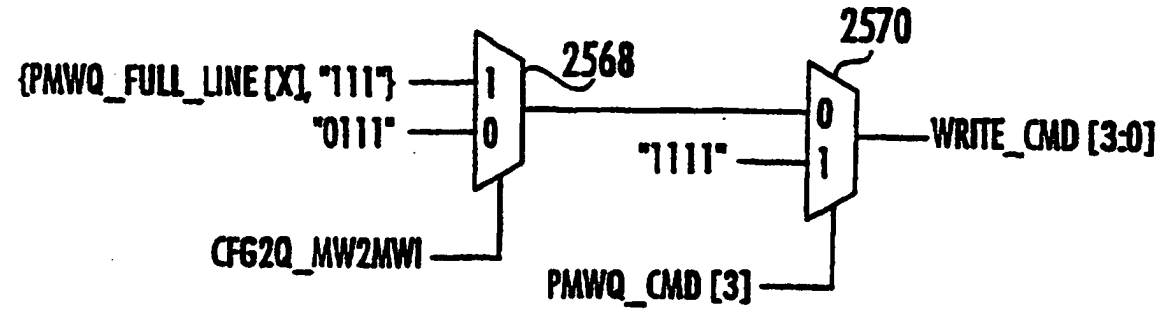


FIG. 81A

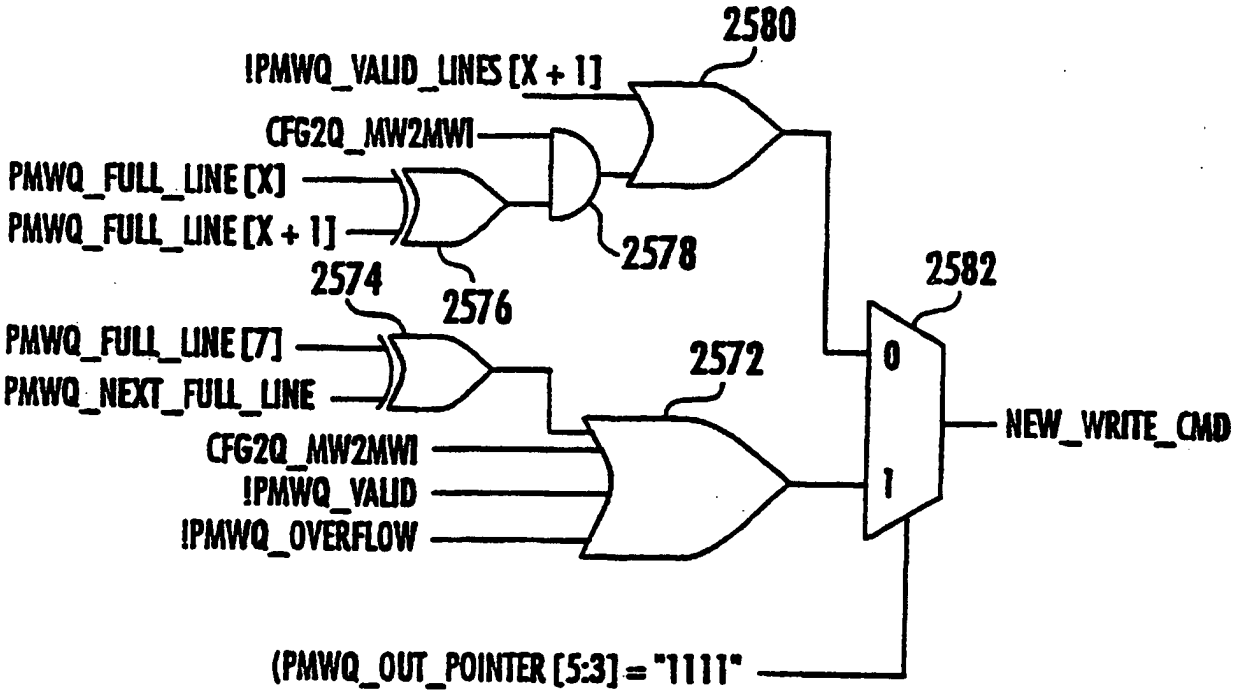


FIG. 81B

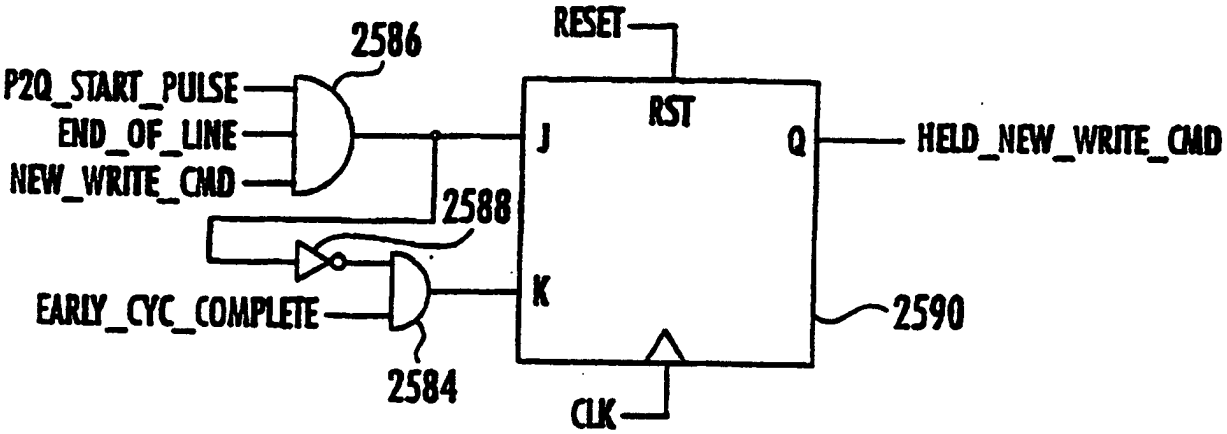


FIG. 81C

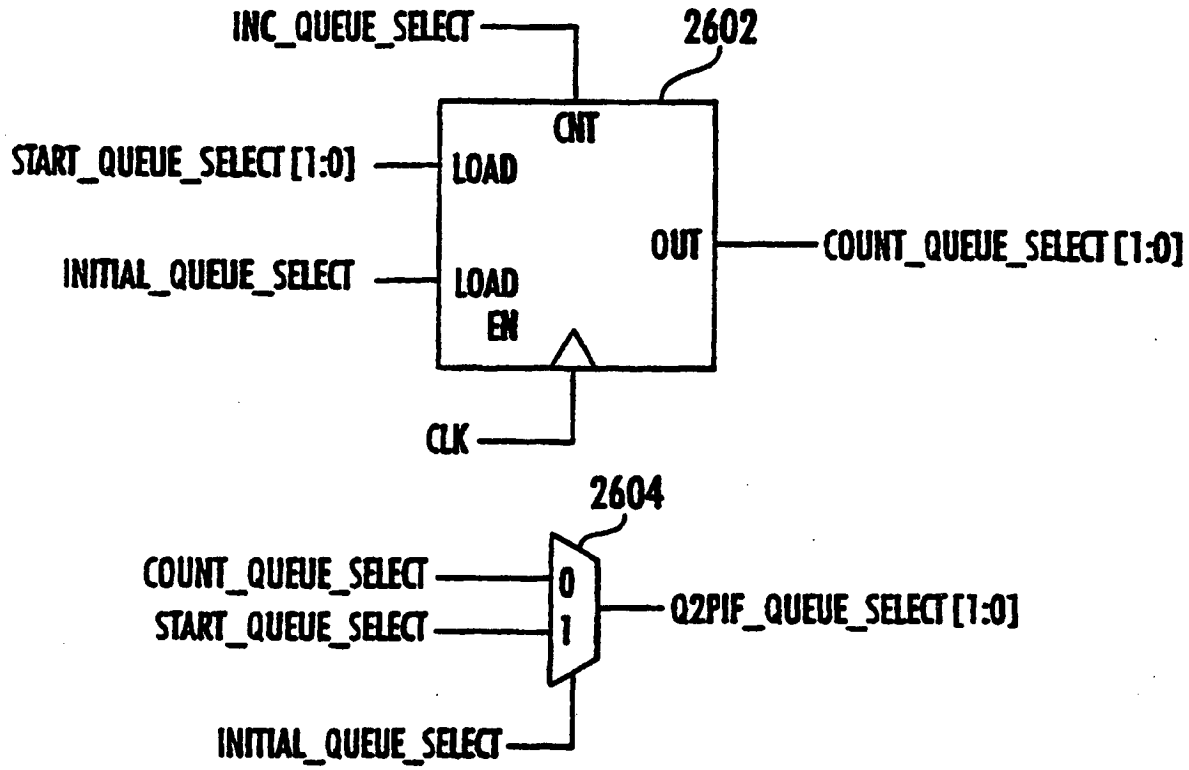


FIG. 82A

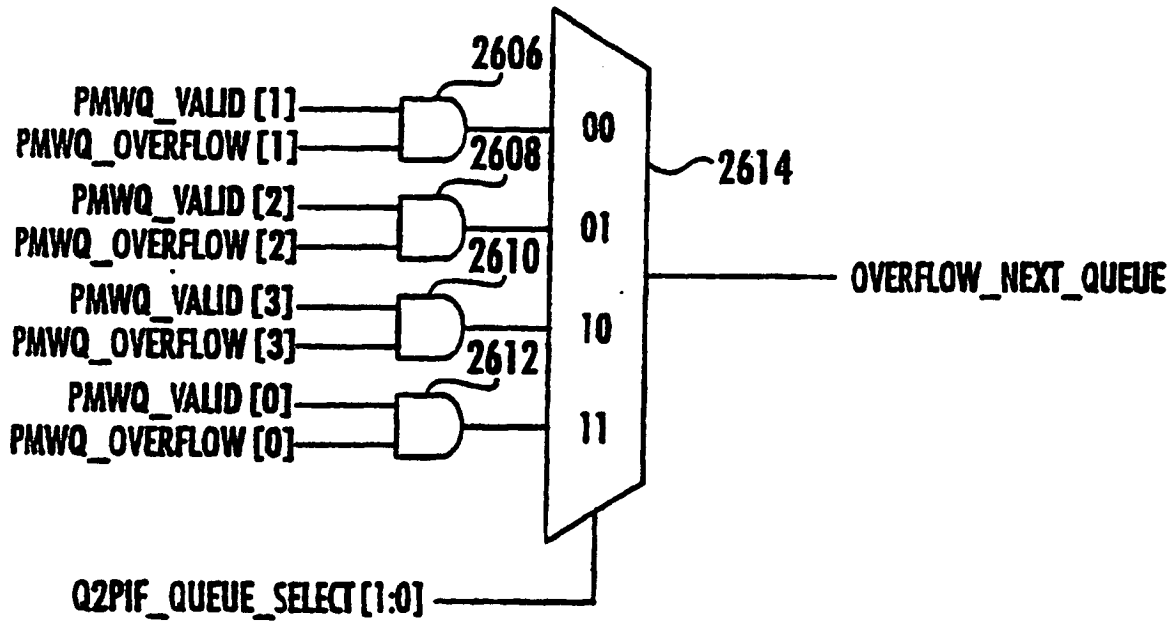


FIG. 82B

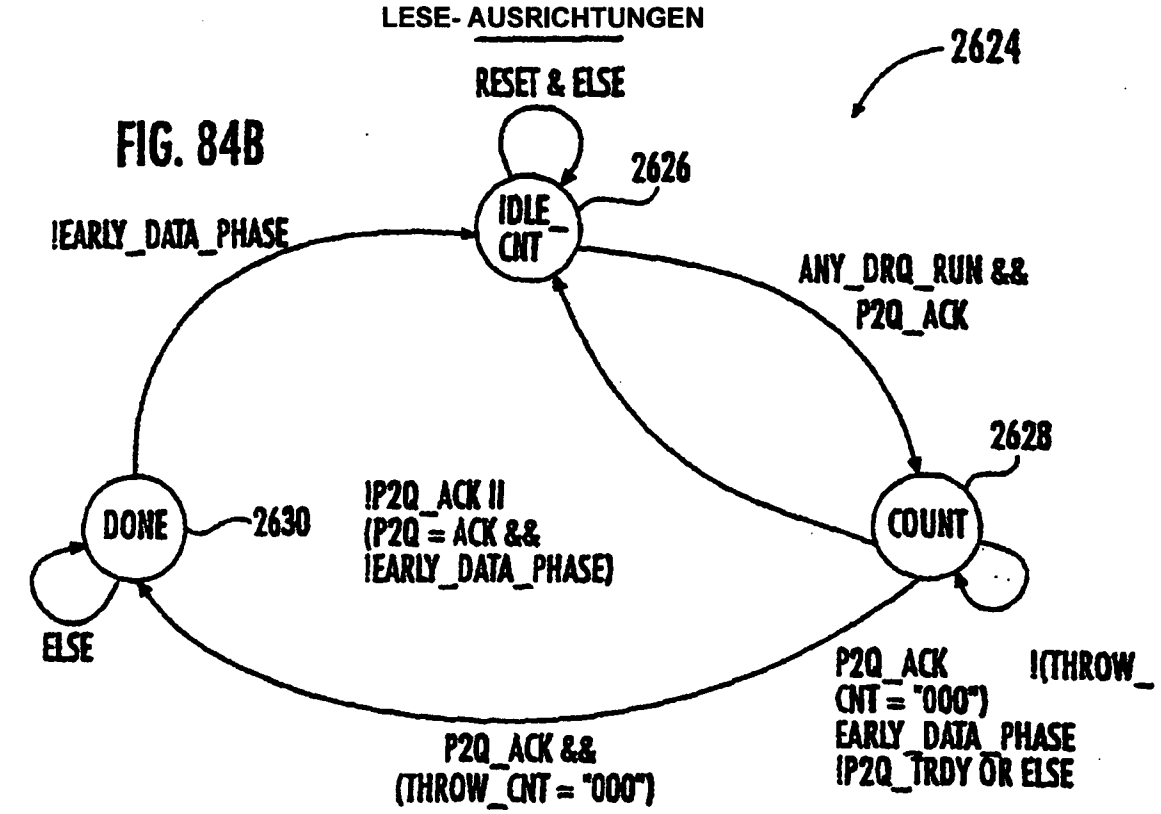


FIG. 84B

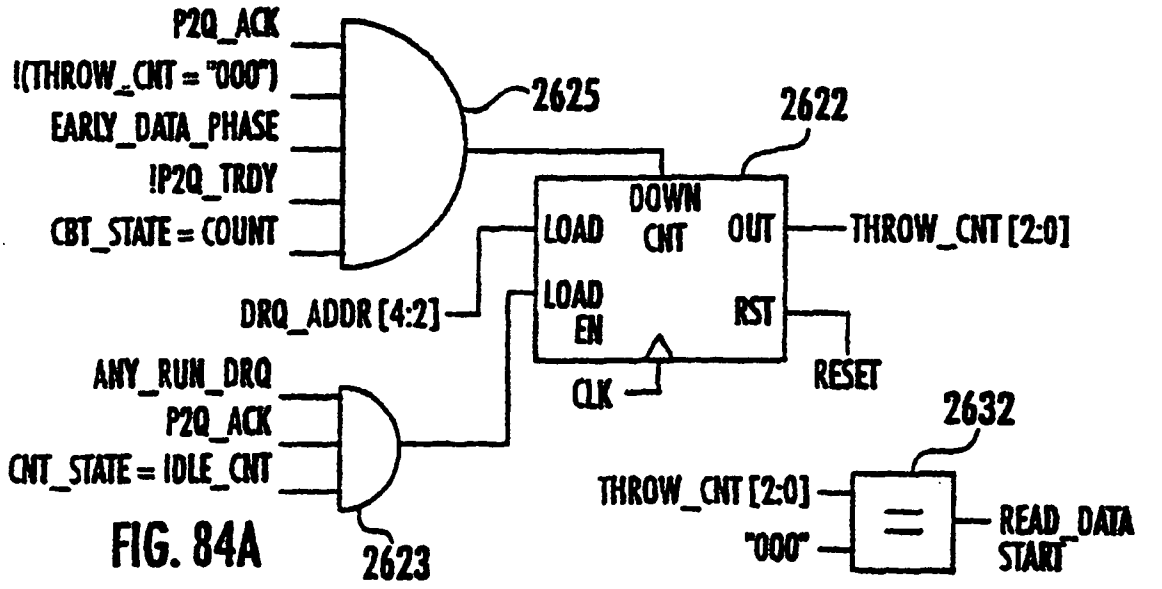


FIG. 84A

FIG. 84C

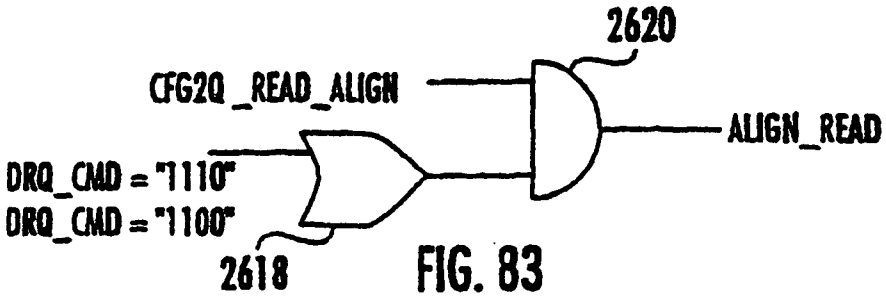


FIG. 83

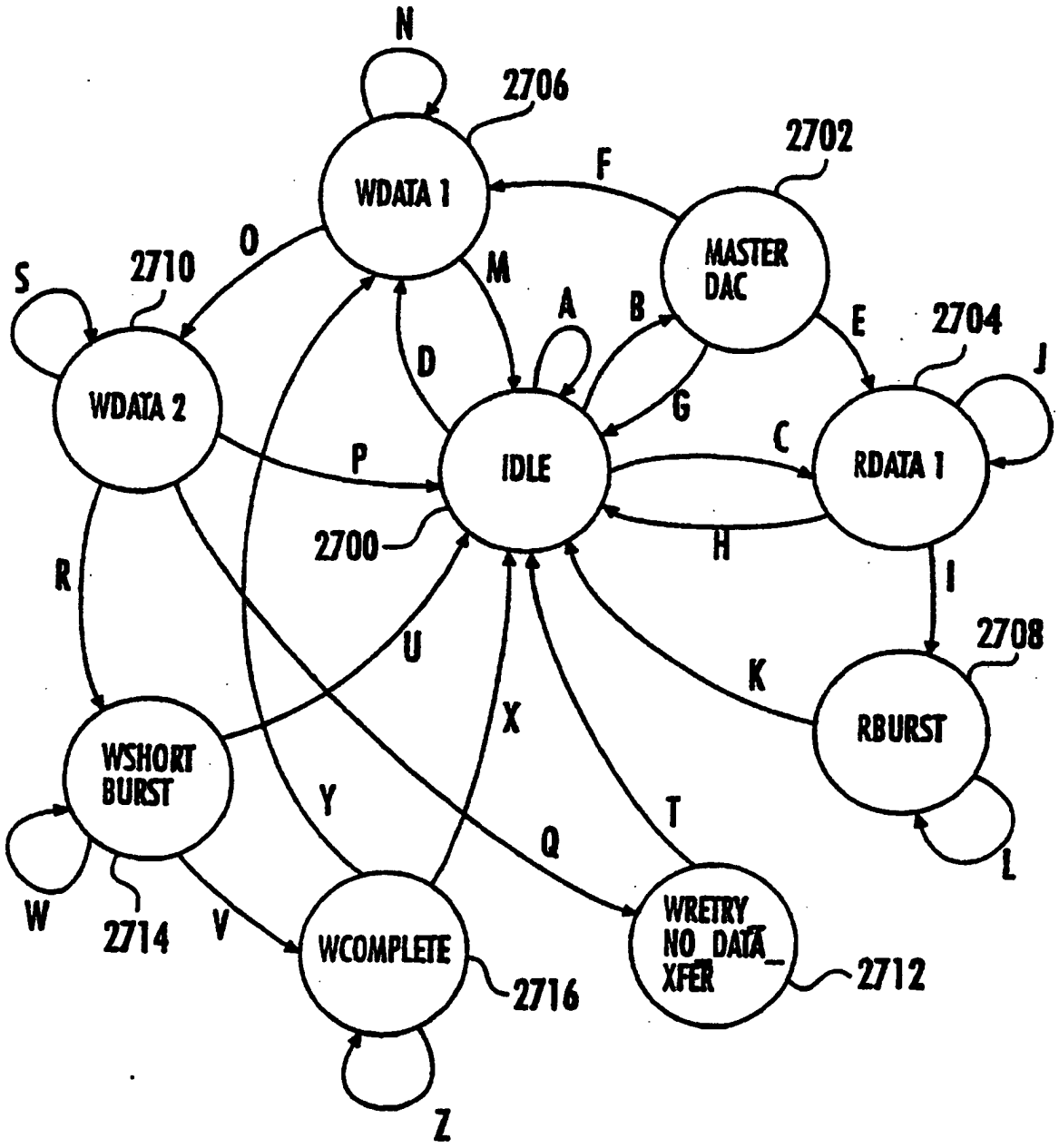

N
2706
WDATA 1
2702
F
MASTER DAC
O
2710
S
M
A
B
E
2704
J
WDATA 2
D
G
C
P
IDLE
RDATA 1
2700
H
R
I
2708
U
K
X
RBURST
WSHORT BURST
Y
T
Q
W
2714
V
WCOMPLETE
2716
WRETRY NO_DATA_ XFER
2712
L
Z

FIG. 85

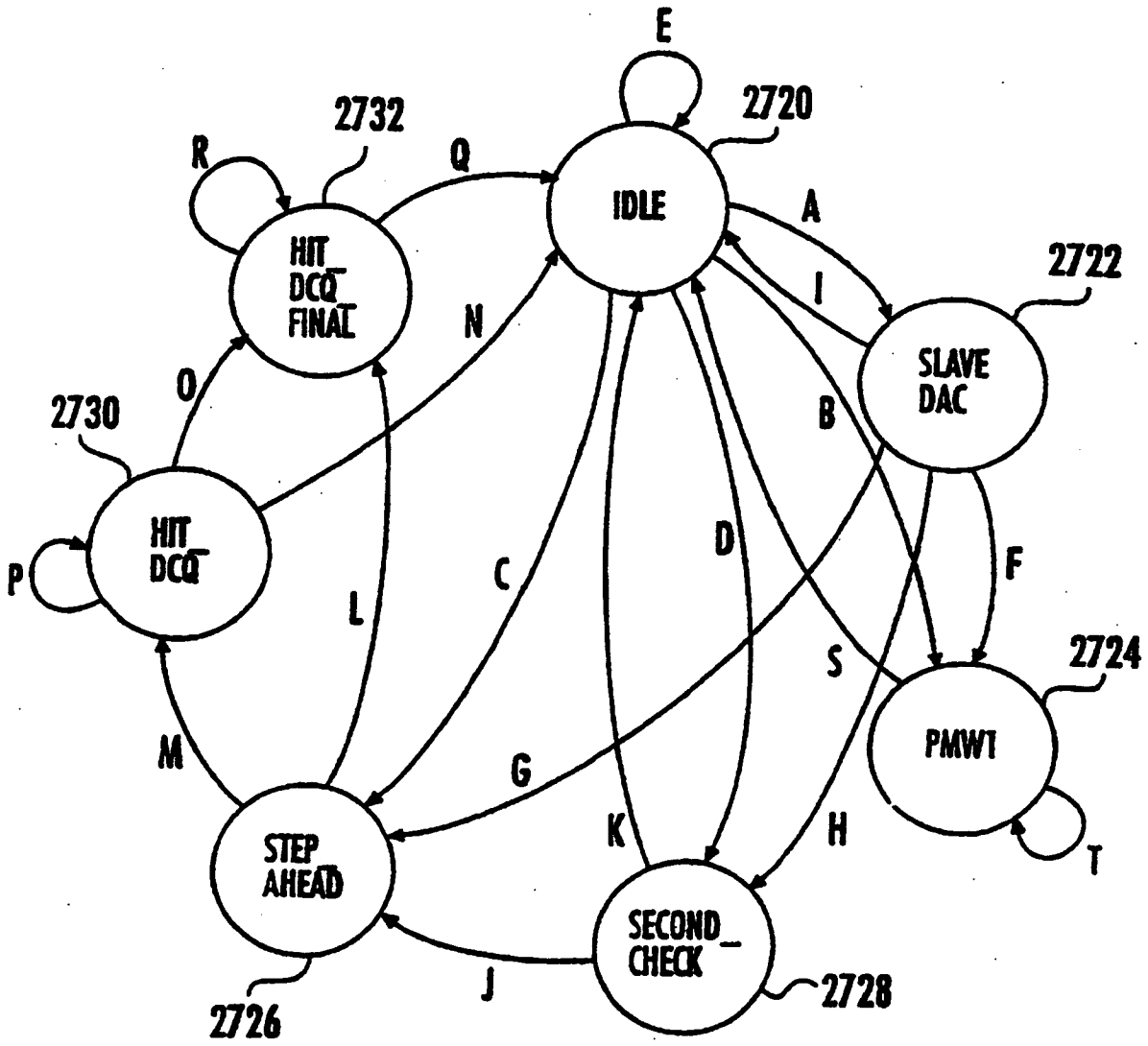

E
2720
2732
R
Q
IDLE
A
HIT_
DCQ_
FINAL
2722
I
SLAVE
DAC
N
O
2730
B
HIT_
DCQ_
D
F
P
C
L
2724
S
PMWT
M
G
K
H
STEP
AHEAD
SECOND_
CHECK
T
J
2726
2728

FIG. 86

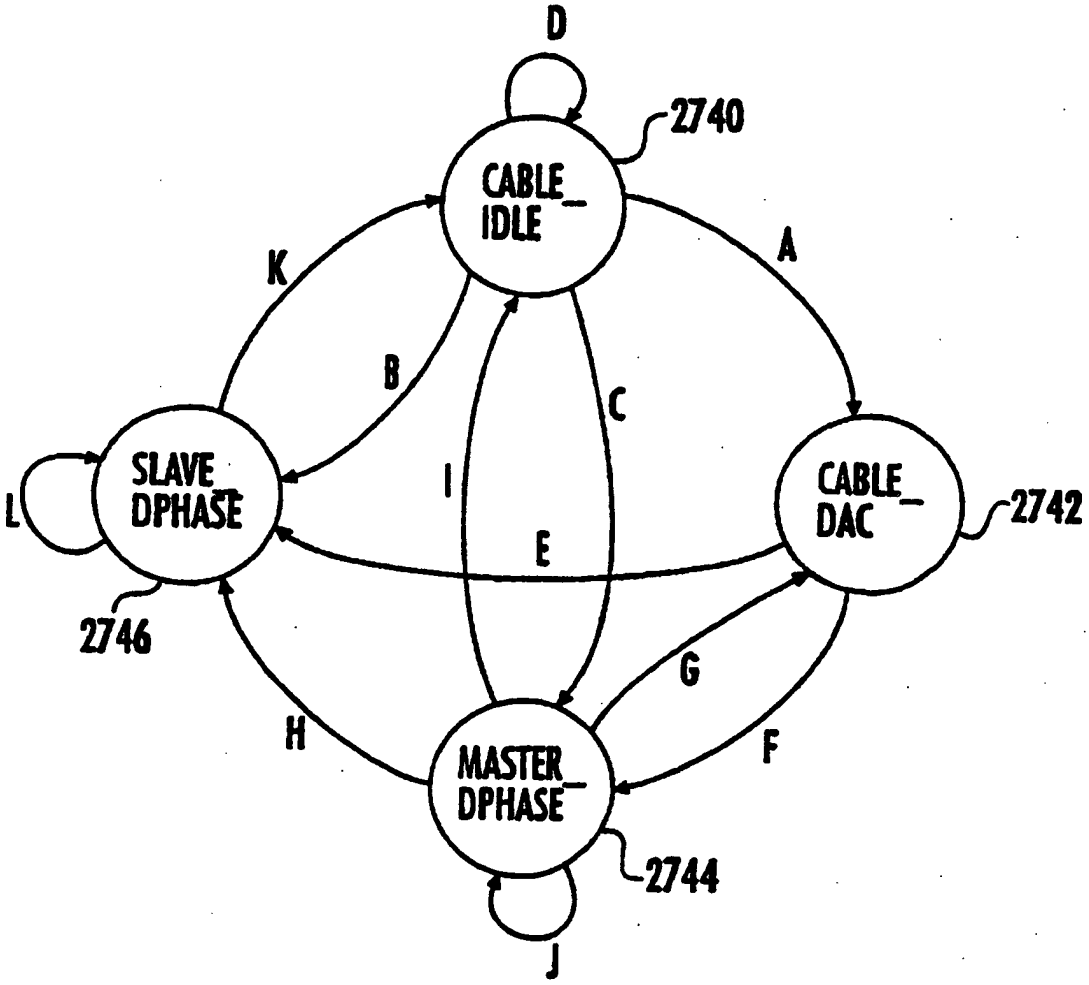

D
2740
CABLE_
IDLE
K
A
B
C
SLAVE
DPHASE
I
CABLE_
DAC
2742
L
E
2746
G
H
F
MASTER_
DPHASE
2744
J

FIG. 87

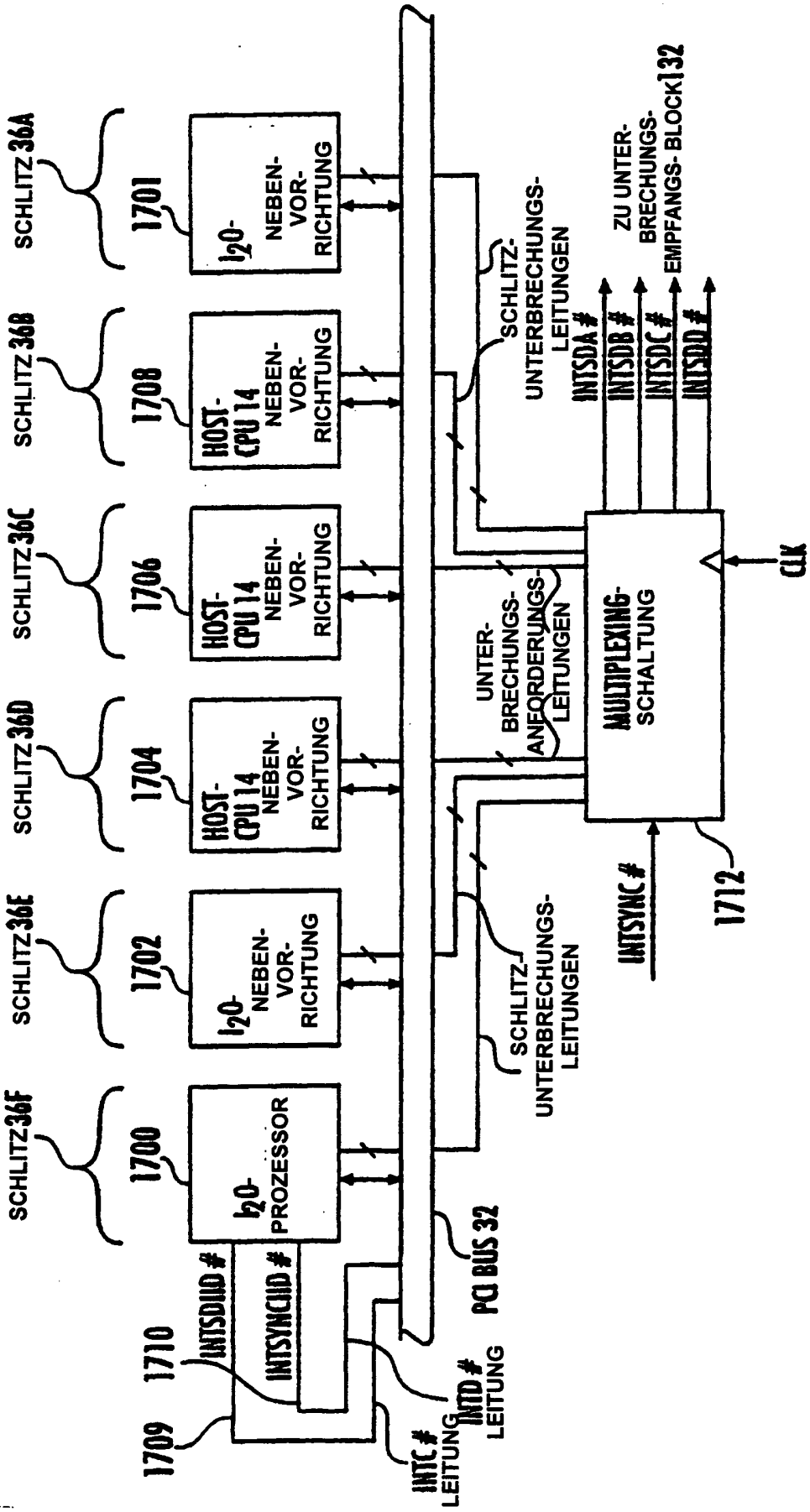

SCHLITZ 36A
SCHLITZ 36B
SCHLITZ 36C
SCHLITZ 36D
SCHLITZ 36E
SCHLITZ 36F
1701
1708
1706
1704
1702
1700
1710
1709
I2O-NEBEN-VOR-RICHTUNG
I2O-PROZESSOR
HOST-CPU 14 NEBEN-VOR-RICHTUNG
INTSDIID #
INTSYNCIID #
INTC # LEITUNG
INTD # LEITUNG
PCI BUS 32
SCHLITZ-UNTERBRECHUNGS-LEITUNGEN
UNTER-BRECHUNGS-ANFORDERUNGS-LEITUNGEN
MULTIPLEXING-SCHALTUNG
INTSYNC #
1712
CLK
INTSDA #
INTSDB #
INTSDC #
INTSDD #
ZU UNTER-BRECHUNGS-EMPFANGS-BLOCK 132

FIG. 88

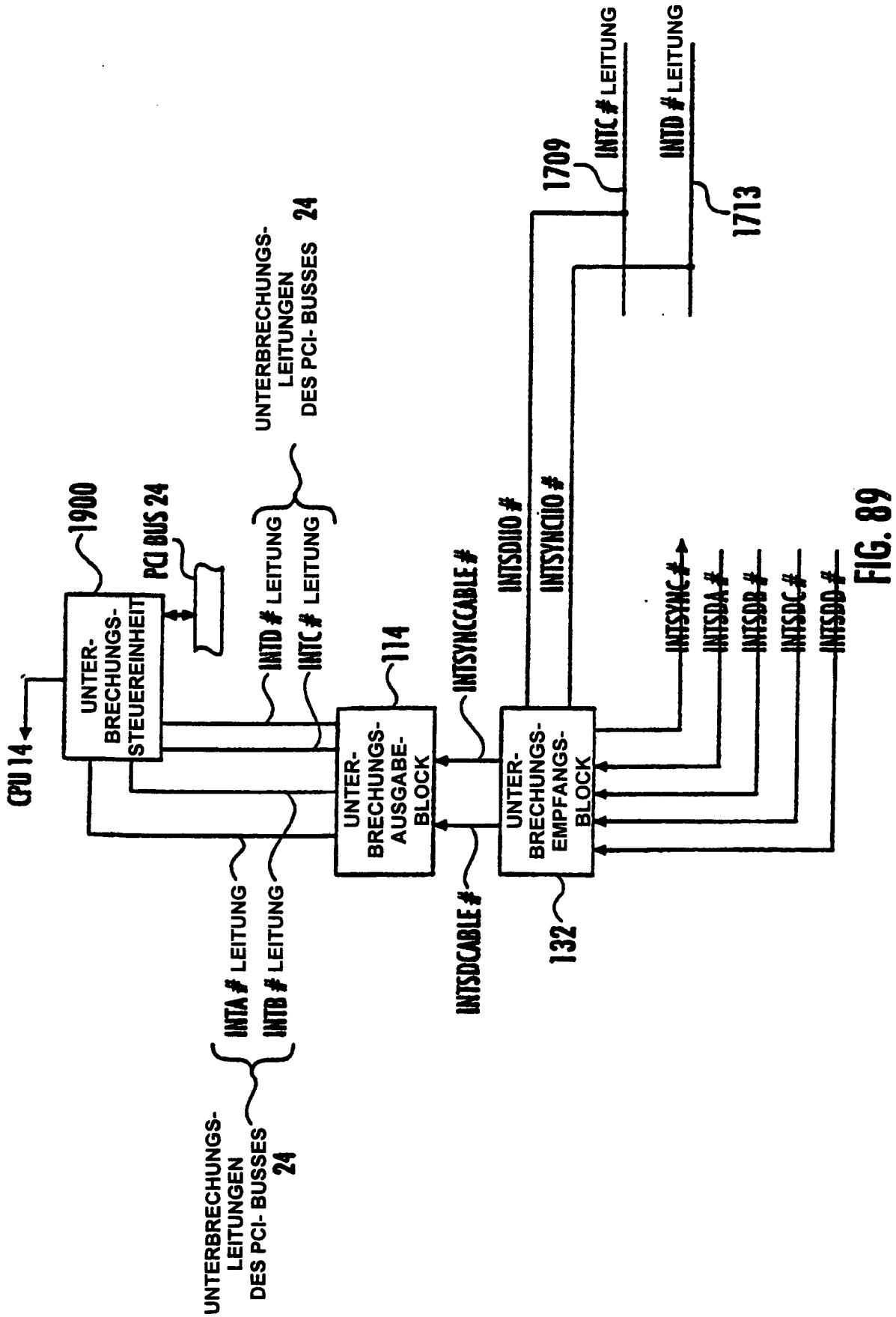

CPU 14
1900
UNTER-BRECHUNGS-STEUEREINHEIT
PCI BUS 24
INTA # LEITUNG
INTB # LEITUNG
UNTERBRECHUNGS-LEITUNGEN DES PCI- BUSSES 24
INTD # LEITUNG
INTC # LEITUNG
UNTERBRECHUNGS-LEITUNGEN DES PCI- BUSSES 24
114
UNTER-BRECHUNGS-AUSGABE-BLOCK
INTSYNCCABLE #
INTSDCABLE #
INTSDIIO #
INTSYNCIIO #
1709
INTC # LEITUNG
INTD # LEITUNG
1713
UNTER-BRECHUNGS-EMPFANGS-BLOCK
132
INTSYNC #
INTSDA #
INTSDB #
INTSDC #
INTSDD #

FIG. 89

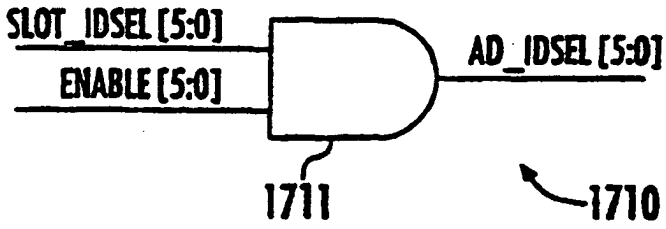


FIG. 90

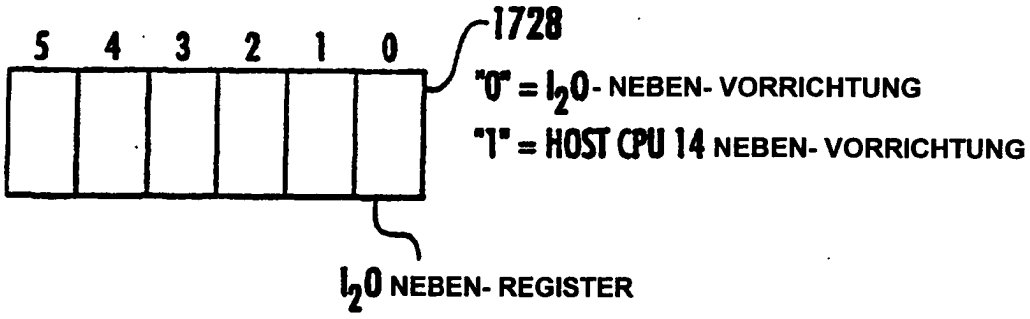


FIG. 91

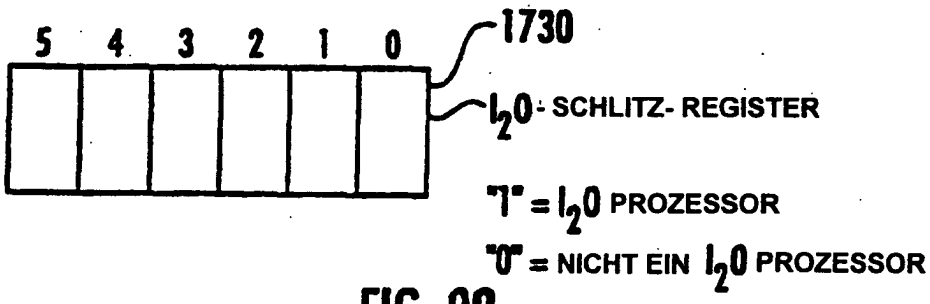
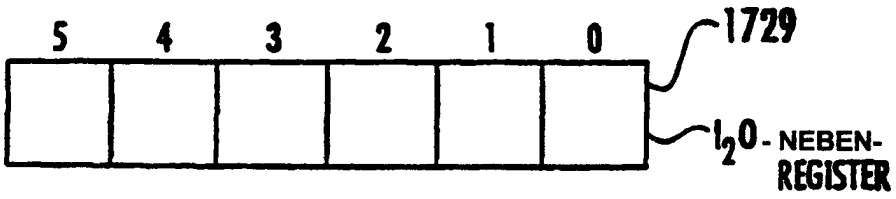


FIG. 92



"0" = I_2O NEBEN- VORRICHTUNG
"1" = HOST CPU 14 NEBEN- VORRICHTUNG

FIG. 93

HOST KONFIGURATIONS- FREIGABE- BIT

"1" = CPU 14 KANN AUF PCI- BUS 32 KONFIGURIEREN

"0" = I_2O PROZESSOR 1700 KONFIGURIERT
I_2O NEBEN- VORRICHTUNGEN AUF BUS 32

FIG. 94

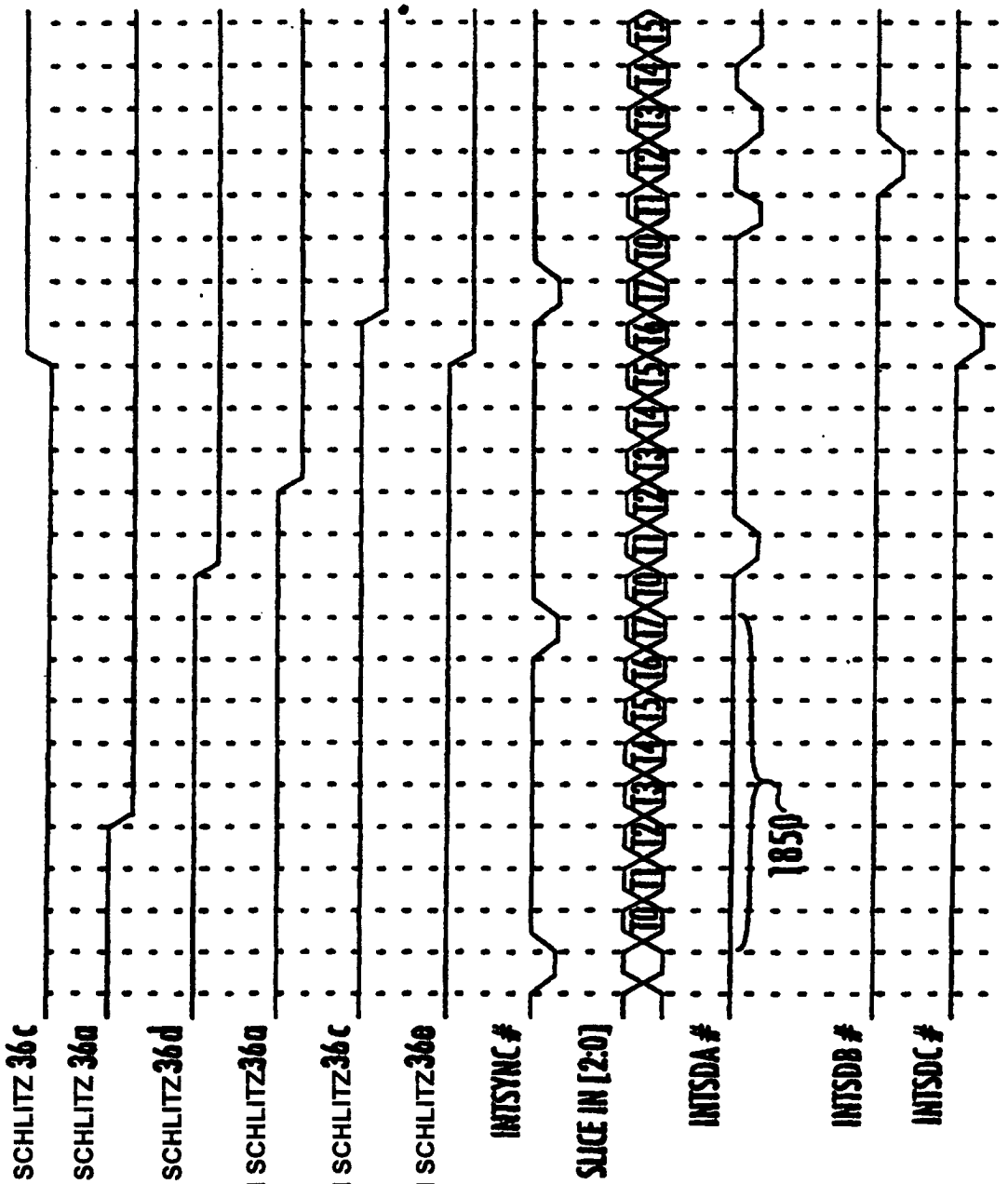


FIG. 95A

FIG. 95

FIG. 95A
FIG. 95B

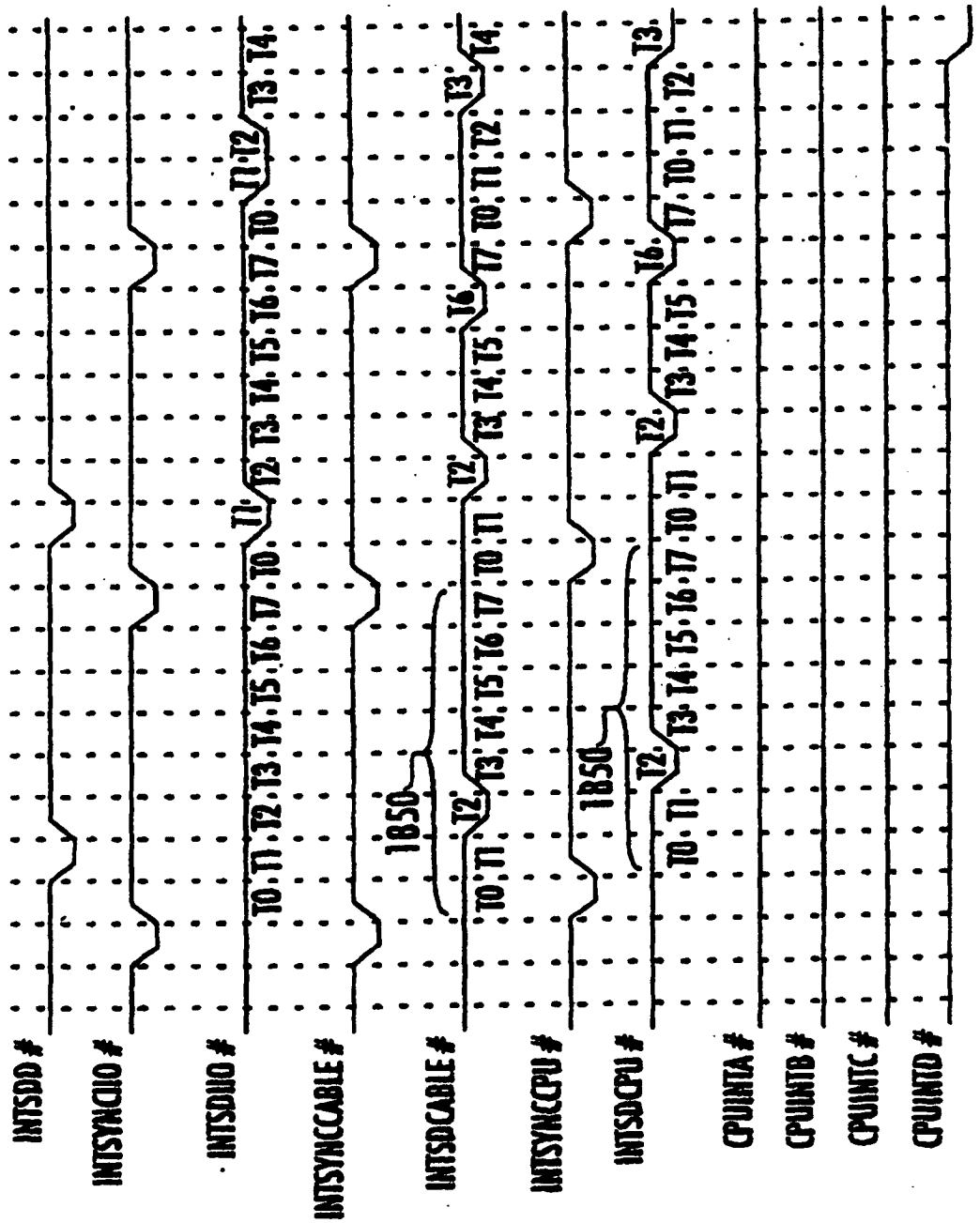

INTSDD #
INTSYNCIIO #
INTSDIIO #
INTSYNCCABLE #
INTSDCABLE #
INTSYNCCPU #
INTSDCPU #
CPUINTA #
CPUINTB #
CPUINTC #
CPUINTD #

FIG. 95B

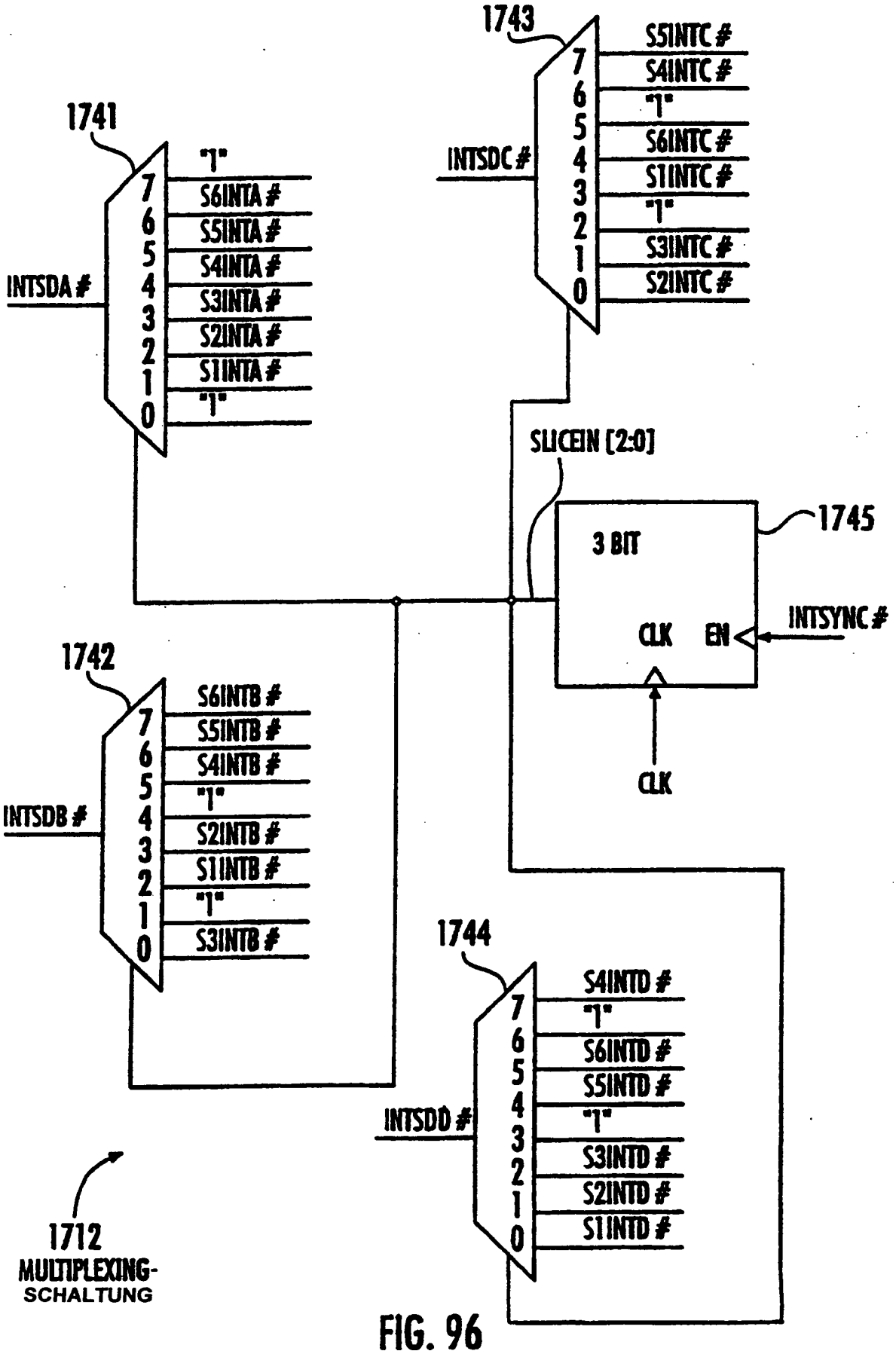

1741
"1"
S6INTA #
S5INTA #
S4INTA #
S3INTA #
S2INTA #
S1INTA #
"1"
INTSDA #
1743
S5INTC #
S4INTC #
"1"
S6INTC #
S1INTC #
"1"
S3INTC #
S2INTC #
INTSDC #
SLICEIN [2:0]
1745
3 BIT
CLK
EN
INTSYNC #
CLK
1742
S6INTB #
S5INTB #
S4INTB #
"1"
S2INTB #
S1INTB #
"1"
S3INTB #
INTSDB #
1744
S4INTD #
"1"
S6INTD #
S5INTD #
"1"
S3INTD #
S2INTD #
S1INTD #
INTSDD #
1712
MULTIPLEXING-
SCHALTUNG

FIG. 96

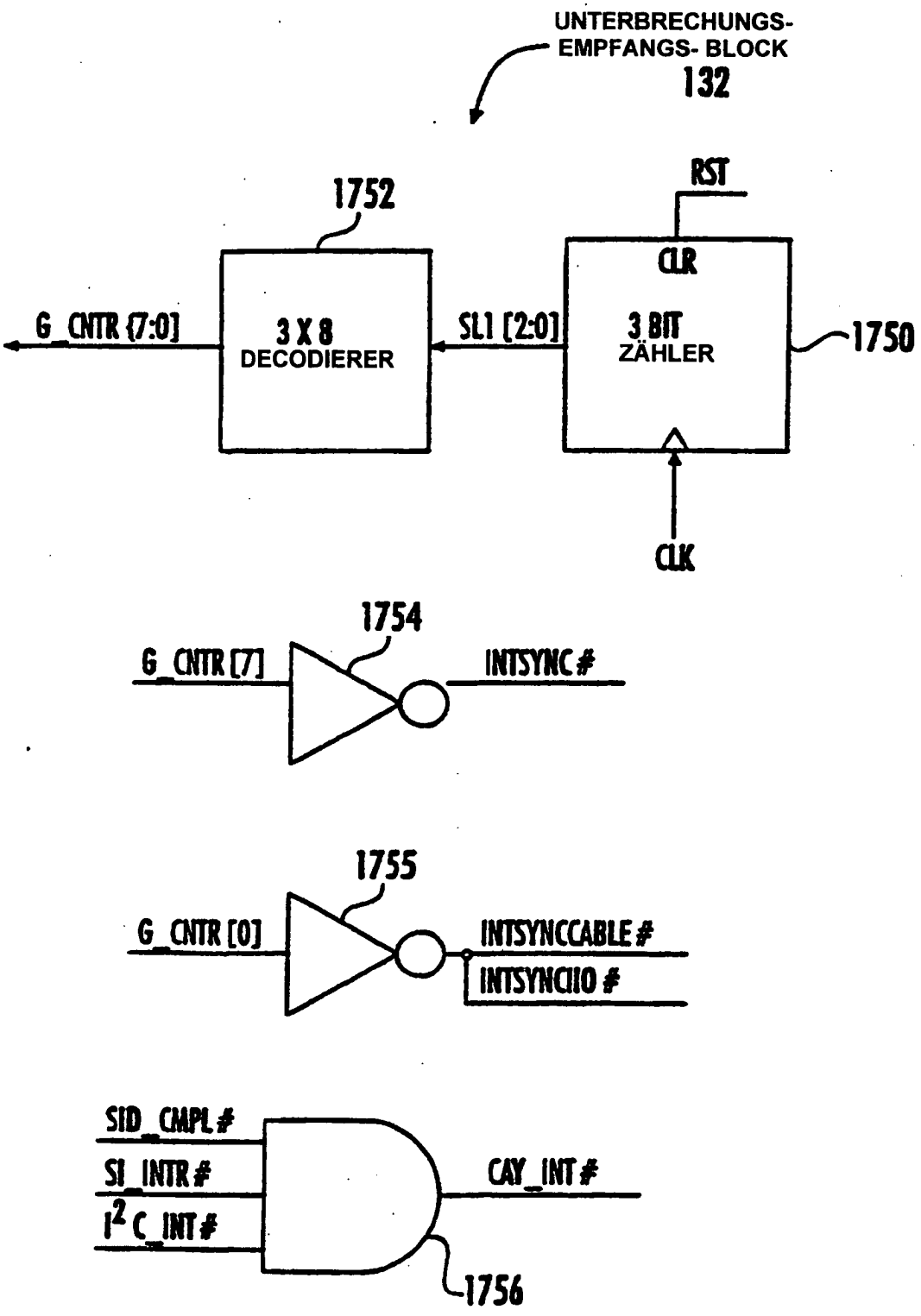

UNTERBRECHUNGS-
EMPFANGS- BLOCK
132
1752
RST
CLR
G_CNTR [7:0]
3 X 8
DECODIERER
SL1 [2:0]
3 BIT
ZÄHLER
1750
CLK
1754
G_CNTR [7]
INTSYNC #
1755
G_CNTR [0]
INTSYNCCABLE #
INTSYNCIIO #
SID_CMPL #
SI_INTR #
I2C_INT #
CAY_INT #
1756

FIG. 97A

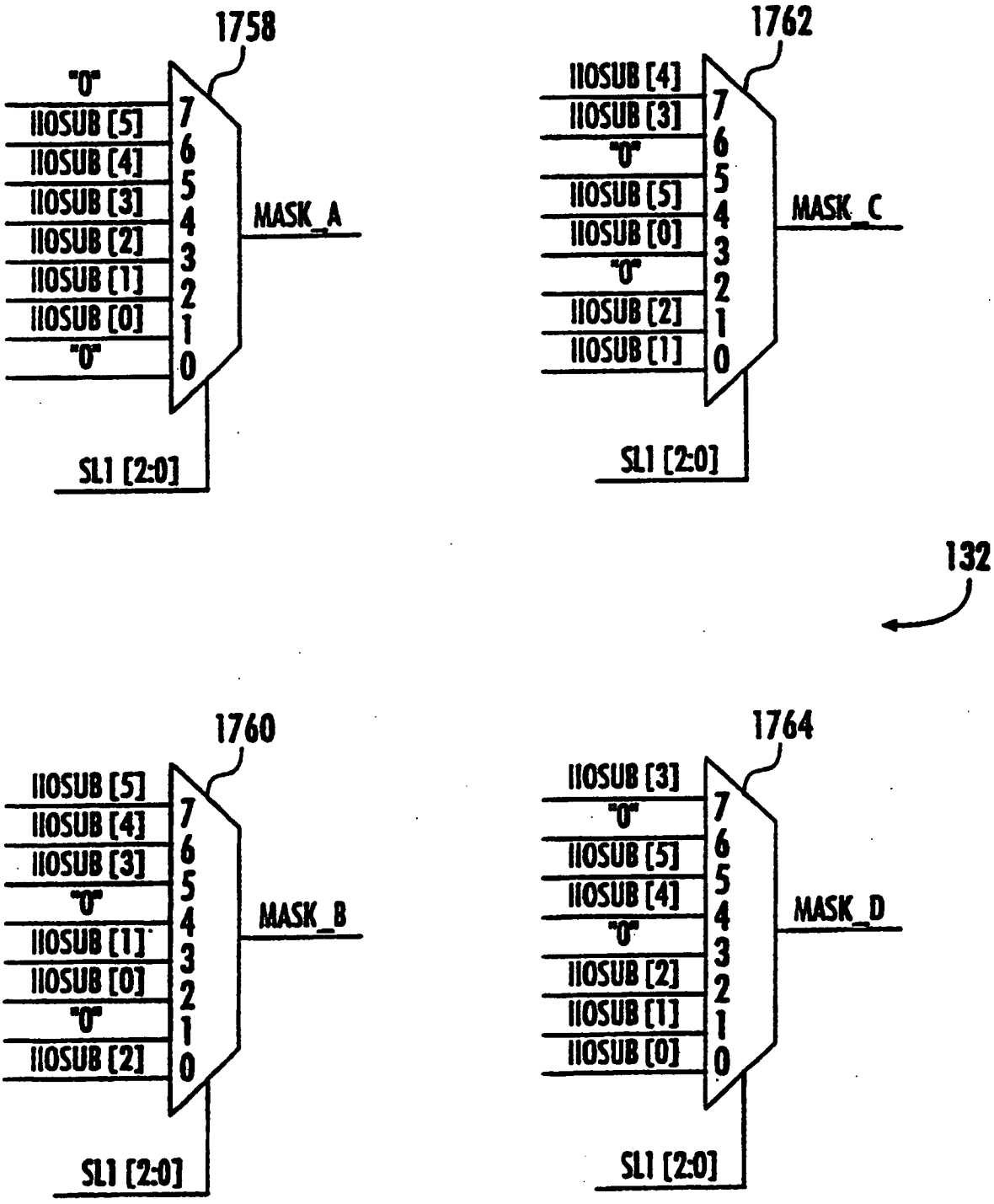

1758
"0"
IIOSUB [5]
IIOSUB [4]
IIOSUB [3]
IIOSUB [2]
IIOSUB [1]
IIOSUB [0]
"0"
7 6 5 4 3 2 1 0
MASK_A
SL1 [2:0]
1762
IIOSUB [4]
IIOSUB [3]
"0"
IIOSUB [5]
IIOSUB [0]
"0"
IIOSUB [2]
IIOSUB [1]
7 6 5 4 3 2 1 0
MASK_C
SL1 [2:0]
132
1760
IIOSUB [5]
IIOSUB [4]
IIOSUB [3]
"0"
IIOSUB [1]
IIOSUB [0]
"0"
IIOSUB [2]
7 6 5 4 3 2 1 0
MASK_B
SL1 [2:0]
1764
IIOSUB [3]
"0"
IIOSUB [5]
IIOSUB [4]
"0"
IIOSUB [2]
IIOSUB [1]
IIOSUB [0]
7 6 5 4 3 2 1 0
MASK_D
SL1 [2:0]

FIG. 97B

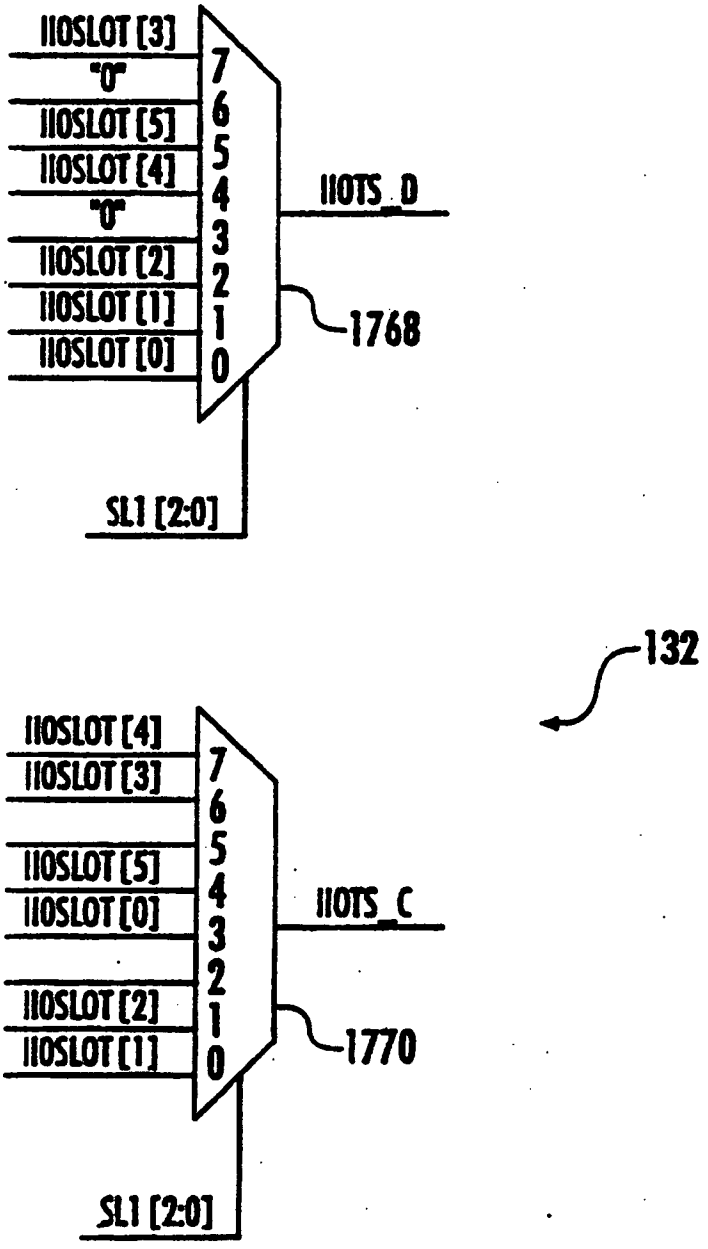

IIOSLOT [3]
"0"
IIOSLOT [5]
IIOSLOT [4]
"0"
IIOSLOT [2]
IIOSLOT [1]
IIOSLOT [0]
7
6
5
4
3
2
1
0
IIOTS_D
1768
SL1 [2:0]
132
IIOSLOT [4]
IIOSLOT [3]
IIOSLOT [5]
IIOSLOT [0]
IIOSLOT [2]
IIOSLOT [1]
IIOTS_C
1770
SL1 [2:0]

FIG. 97C

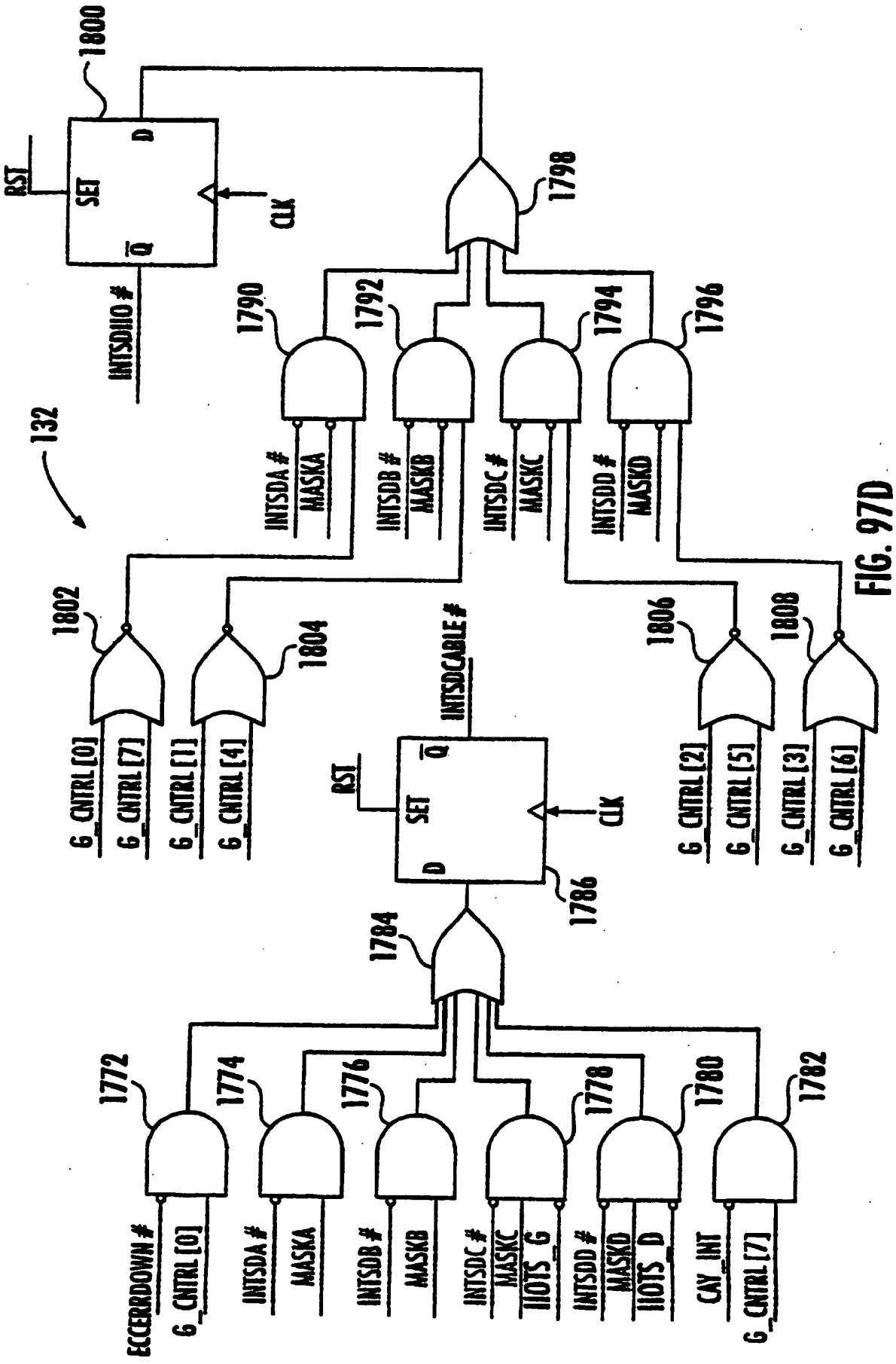

1800
RST
SET
D
Q
CLK
INTSDIIO #
1798
1790
1792
1794
1796
INTSDA #
MASKA
INTSDB #
MASKB
INTSDC #
MASKC
INTSDD #
MASKD
132
1802
1804
1806
1808
6_CNTRL [0]
6_CNTRL [7]
6_CNTRL [1]
6_CNTRL [4]
6_CNTRL [2]
6_CNTRL [5]
6_CNTRL [3]
6_CNTRL [6]
INTSDCABLE #
RST
SET
D
Q
CLK
1786
1784
1772
1774
1776
1778
1780
1782
ECCERRDOWN #
6_CNTRL [0]
INTSDA #
MASKA
INTSDB #
MASKB
INTSDC #
MASKC
IIOTS_G
INTSDD #
MASKD
IIOTS_D
CAV_INT
6_CNTRL [7]

FIG. 97D

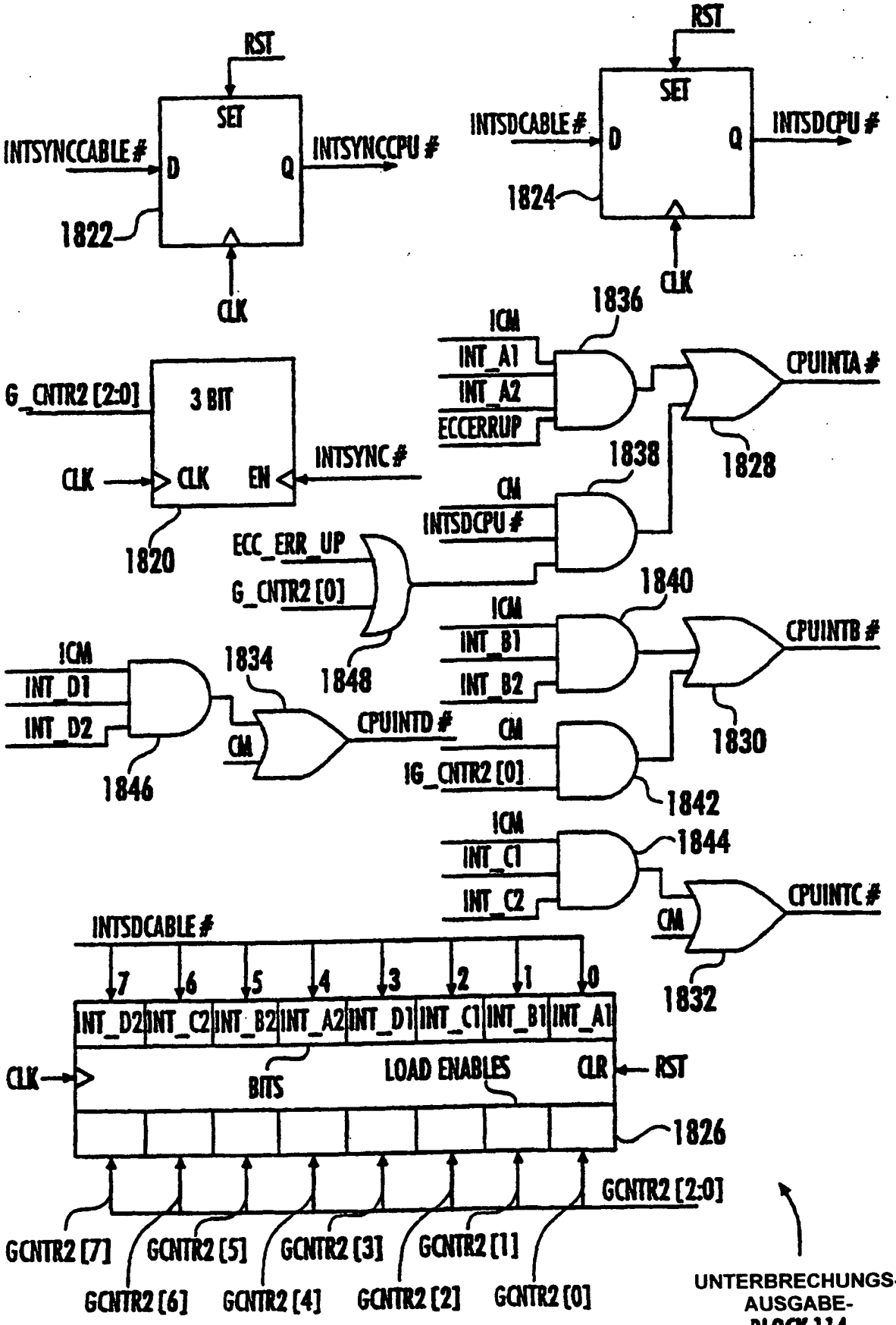

RST
SET
INTSYNCCABLE#
D
Q
INTSYNCCPU#
1822
CLK
RST
SET
INTSDCABLE#
D
Q
INTSDCPU#
1824
CLK
1836
ICM
INT_A1
INT_A2
ECCERRUP
CPUINTA#
1828
G_CNTR2 [2:0]
3 BIT
CLK
CLK
EN
INTSYNC#
1820
1838
CM
INTSDCPU#
ECC_ERR_UP
G_CNTR2 [0]
1848
1840
ICM
INT_B1
INT_B2
CPUINTB#
1830
ICM
INT_D1
INT_D2
1834
CM
CPUINTD#
1846
CM
IG_CNTR2 [0]
1842
ICM
INT_C1
INT_C2
1844
CPUINTC#
CM
1832
INTSDCABLE#
7
6
5
4
3
2
1
0
INT_D2
INT_C2
INT_B2
INT_A2
INT_D1
INT_C1
INT_B1
INT_A1
CLK
BITS
LOAD ENABLES
CLR
RST
1826
GCNTR2 [2:0]
GCNTR2 [7]
GCNTR2 [6]
GCNTR2 [5]
GCNTR2 [4]
GCNTR2 [3]
GCNTR2 [2]
GCNTR2 [1]
GCNTR2 [0]
UNTERBRECHUNGS-
AUSGABE-
BLOCK 114

FIG. 98

ZEIT-AB-SCHNITT	INTSDA #	INTSDD #	INTSDC #	INTSDB #
T0		SCHLITZ 1. INTD #	SCHLITZ 2. INTC #	SCHLITZ 3. INTB #
T1	SCHLITZ 1. INTA #	SCHLITZ 2. INTD #	SCHLITZ 3. INTC #	
T2	SCHLITZ 2. INTA #	SCHLITZ 3. INTD #		SCHLITZ 1. INTB #
T3	SCHLITZ 3. INTA #		SCHLITZ 1. INTC #	SCHLITZ 2. INTB #
T4	SCHLITZ 4. INTA #	SCHLITZ 5. INTD #	SCHLITZ 6. INTC #	
T5	SCHLITZ 5. INTA #	SCHLITZ 6. INTD #		SCHLITZ 4. INTB #
T6	SCHLITZ 6. INTA #		SCHLITZ 4. INTC #	SCHLITZ 5. INTB #
T7		SCHLITZ 4. INTD #	SCHLITZ 5. INTC #	SCHLITZ 6. INTB #

FIG. 99

SCHLITZ 1 = SCHLITZ 36A

SCHLITZ 2 = SCHLITZ 36B

SCHLITZ 3 = SCHLITZ 36C

SCHLITZ 4 = SCHLITZ 36D

SCHLITZ 5 = SCHLITZ 36E

SCHLITZ 6 = SCHLITZ 36F

UNTER-BRECHUNGS-LEITUNGEN AUF PCI BUS 24	UNTERBRECHUNGS- QUELLEN FÜR ERWEITERUNGS- QUELLEN							
INTA #	BR_INTR #	SCHLITZ 1 INTD #	SCHLITZ 2 INTC #	SCHLITZ 3 INTB #	SCHLITZ 4 INTA #	SCHLITZ 5 INTD #	SCHLITZ 6 INTC #	"1"
INTB #	SCHLITZ 1 INTA #	SCHLITZ 2 INTD #	SCHLITZ 3 INTC #	"1"	SCHLITZ 5 INTA #	SCHLITZ 6 INTD #	"1"	SCHLITZ 4 INTB #
INTC #	SCHLITZ 2 INTA #	SCHLITZ 3 INTD #	"1"	SCHLITZ 1 INTB #	SCHLITZ 6 INTA #	"1"	SCHLITZ 4 INTC #	SCHLITZ 5 INTB #
INTD #	SCHLITZ 3 INTA #	"1"	SCHLITZ 1 INTC #	SCHLITZ 2 INTB #	SI_INTR #	SCHLITZ 4 INTD #	SCHLITZ 5 INTC #	SCHLITZ 6 INTB #

FIG. 100

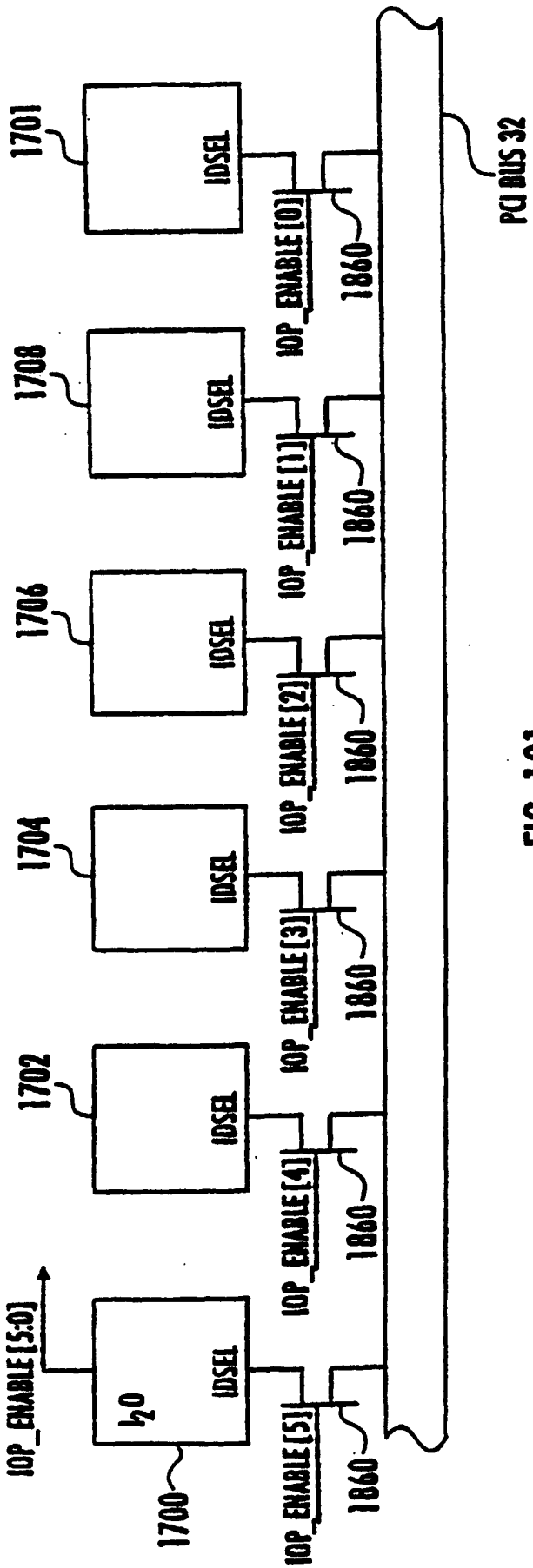

1701
IDSEL
1708
IDSEL
1706
IDSEL
1704
IDSEL
1702
IDSEL
1700
IDSEL
μO
IOP_ENABLE[5:0]
IOP_ENABLE[0]
IOP_ENABLE[1]
IOP_ENABLE[2]
IOP_ENABLE[3]
IOP_ENABLE[4]
IOP_ENABLE[5]
1860
1860
1860
1860
1860
1860
PCI BUS 32

FIG. 101