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

(10) **DE 601 08 124 T2** 2005.12.29

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 297 421 B1**
(21) Deutsches Aktenzeichen: **601 08 124.2**
(86) PCT-Aktenzeichen: **PCT/US01/14409**
(96) Europäisches Aktenzeichen: **01 932 994.5**
(87) PCT-Veröffentlichungs-Nr.: **WO 01/084311**
(86) PCT-Anmeldetag: **04.05.2001**
(87) Veröffentlichungstag der PCT-Anmeldung: **08.11.2001**
(97) Erstveröffentlichung durch das EPA: **02.04.2003**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **29.12.2004**
(47) Veröffentlichungstag im Patentblatt: **29.12.2005**

(51) Int Cl.[7]: **G06F 9/46**

(30) Unionspriorität:
565580 04.05.2000 US

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

(73) Patentinhaber:
Broadcom Corp., Irvine, Calif., US

(72) Erfinder:
RUSTAGI, Viresh, Irvine, US; FRENCH, S., Robert, Irvine, US; BANTA, H., Gareld, Irvine, US

(74) Vertreter:
Bosch, Graf von Stosch, Jehle Patentanwaltsgesellschaft mbH, 80639 München

(54) Bezeichnung: **ENTWICKLUNGSUMGEBUNG MIT MEHREREN KANÄLEN UND DIENSTEN**

Beschreibung

GEBIET DER ERFINDUNG

[0001] Die vorliegende Erfindung bezieht sich auf das Verarbeiten von Daten, insbesondere auf das Verarbeiten eines Datenflusses in einer Entwicklungsumgebung mit mehreren Kanälen und mehreren Diensten.

STAND DER TECHNIK

[0002] Die EP 690 376 beschreibt eine Digitalsignalverarbeitungsvorrichtung, die zur Berechnung von Bildcodierungs-Signalen oder ähnlichem verwendet wird, sowie ein bewegungskompensierendes Bearbeitungsverfahren, das eine Digitalsignalverarbeitungsvorrichtung verwendet. Die darin beschriebene Vorrichtung weist eine Vielzahl von parallel angeordneten Signalverarbeitungseinrichtungen und Steuerungseinrichtungen auf, die den Signalverarbeitungsvorrichtungen Ladungen zuordnen, so dass die Signalverarbeitungseinrichtungen gleichmäßige Berechnungsvolumen aufweisen. Alternativ ist ein Adressengenerator für jeden der unabhängig eingegebenen Datensätze vorgesehen. Eine Zwischenüberprüfung wird während der Berechnung für einen Block durchgeführt, was eine bewegungskompensierende Bearbeitung involviert. Während der Verarbeitung der Signale durch die Vorrichtung bleibt das Datenvolumen unverändert, was je nach verarbeiteten Signalen eine hohe Verarbeitungskapazität erfordert. Die Art der zu verarbeitenden Daten wird während der Verarbeitung nicht ausgewertet.

HINTERGRUND DER ERFINDUNG

[0003] Traditionell werden Digitalsignalprozessoren (Digital Signal Processors (DPSs)) verwendet, um einzelne Kanäle, beispielsweise einen einzelnen DS0 (Digital Signal 0)- oder TDM (Time Division Mutiplexing)-Slot, zu betreiben, die einzelne Dienste durchführen, beispielsweise Modem-, Vocoder- oder Paketverarbeitung. Mehrfachdienste erfordern mehrere Kanäle und mehrere Digitalsignalprozessoren, die jeweils ihre eigenen kleinen Ausführungsprogramme (kleiner Kern) und Anwendungen betreiben. Die Ausführungsprogramme reservieren einen Bereich im Speicher für den Anwendungscode. Wenn Anwendungen geschaltet werden müssen, überlagern diese Ausführungsprogramme diesen Speicher mit der neuen Anwendung.

[0004] Kanäle können eine der folgenden Formen aufweisen: ein Kanal, der auf einem Medium mit physikalischen Drähten oder auf einem drahtlosen Medium zwischen Systemen getragen wird (auch als Schaltung bezeichnet); TDM (Time Division Multiplexing)-Kanäle, in denen Signale von mehreren Quellen, beispielsweise Telefonen oder Computern, zu einem einzigen Datenstrom zusammengeführt werden und durch ein Zeitintervall getrennt sind; und FDM (Frequency Division multiplexing)-Kanäle, in denen Signale von vielen Quellen über ein einziges Kabel übertragen werden, indem jedes Signal auf einem Träger mit unterschiedlichen Frequenzen moduliert wird.

[0005] Jüngste Fortschritte bei der Verarbeitungskapazität ermöglichen es nun, dass ein einziger Chip mehrere Kanäle betreibt. Mit dieser Kapazitätserhöhung geht der Wunsch einher, verschiedene Dienste gleichzeitig zu betreiben und zwischen den Diensten zu schalten.

[0006] Ein aktuelles Verfahren zur Implementierung mehrerer Dienste oder mehrerer Kanäle involviert das Schreiben aller Steuerungs-, Overlay- und Task-Switching-Codes für jeden Dienst oder Kanal. Dieses Erfordernis führt zu einem zusätzlichen technischen Aufwand für die Entwicklung und das Austesten der Anwendungen. Außerdem ist es möglich, dass nicht alle Dienste in den für den Datensignalprozessor zur Verfügung stehenden Speicher passen und die Dienste vom Hostsystem ausgelagert werden müssen. Dieses Auslagern (Swapping) – Overlaying – führt dazu, dass die Implementierung der Digitalsignalprozessor-Dienste sehr komplex wird. Die zusätzliche Entwicklungsaktivität geht zu Lasten der Entwicklungszeit für die Digitalprozessor-Anwendung.

ZUSAMMENFASSUNG DER ERFINDUNG

[0007] Es wird ein Verfahren und ein System zum Verarbeiten eines Datenflusses in einer Umgebung mit mehreren Kanälen und mehreren Diensten beschrieben. Bei einer Ausführungsform wird ein Sockel dynamisch zugeordnet, wobei der Sockel einen dynamisch zugeordneten Dienst aufweist. Weiterhin verarbeitet der Server den Datenfluss auf der Grundlage der Datenart, die verarbeitet wird.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0008] Die vorliegende Erfindung wird beispielhaft und nicht einschränkend mittels der Figuren der beigefügten Zeichnungen beschrieben, in denen gleiche Bezugszeichen ähnliche Elemente bezeichnen. Es zeigt:

[0009] Fig. 1 eine Systemarchitektur einer Ausführungsform für ein System mit mehreren Kanälen und mehreren Diensten;

[0010] Fig. 2 ein Blockdiagramm einer Ausführungsform für einen Verarbeitungschip der Fig. 1;

[0011] Fig. 3 ein Blockdiagramm einer anderen Ausführungsform für ein System mit mehreren Kanä-

len und mehreren Diensten;

[0012] Fig. 4 ein beispielhaftes Diagramm von Kanälen in einem System mit mehreren Kanälen und mehreren Diensten;

[0013] Fig. 5 ein Blockdiagramm einer Ausführungsform für eine Dienststeuerungssockel (DSS) (Service Control Socket (SCS))-Konfiguration;

[0014] Fig. 6 ein beispielhaftes Blockdiagramm für eine Ausführungsform einer Dienststeuerungssockel-Konfiguration;

[0015] Fig. 7 ein Blockdiagramm einer Ausführungsform für eine Datenaggregationssockel (DAS) (Data Aggregation Socket (DAS))-Konfiguration;

[0016] Fig. 8 ein Blockdiagramm einer Ausführungsform für Sockeldaten;

[0017] Fig. 9a ein Blockdiagramm einer Ausführungsform für eine Steuerungsaggregationssockel (SAS) (Control Aggregation Socket (CAS))-Konfiguration;

[0018] Fig. 9b eine weitere Ausführungsform für eine Steuerungsaggregationssockel-Konfiguration;

[0019] Fig. 10 ein Flussdiagramm einer Ausführungsform zur Verarbeitung von Daten und Informationen durch Kanäle;

[0020] Fig. 11 ein Flussdiagramm einer Ausführungsform zum Erstellen von Kanalsockeln (KS);

[0021] Fig. 12 ein Flussdiagramm einer Ausführungsform zum Erzeugen eines Datenaggregationssockels; und

[0022] Fig. 13 ein Flussdiagramm einer Ausführungsform zum Schalten von Sockeln zwischen Dienststeuerungssockeln.

DETAILLIERTE BESCHREIBUNG

[0023] Es wird ein Verfahren und ein System zum Verarbeiten eines Datenflusses in einer Umgebung mit mehreren Kanälen und mehreren Diensten beschrieben. Bei einer Ausführungsform wird ein Sockel dynamisch zugeordnet, wobei der Sockel einen dynamisch zugeordneten Dienst aufweist. Weiterhin verarbeitet der Server den Datenfluss auf der Grundlage der Datenart, die verarbeitet wird.

[0024] In der folgenden detaillierten Beschreibung der vorliegenden Erfindung werden zahlreiche spezifische Einzelheiten erwähnt, um ein profundes Verständnis der vorliegenden Erfindung zu ermöglichen. Es ist jedoch für den Fachmann selbstverständlich, dass die vorliegende Erfindung ohne diese spezifischen Einzelheiten in die Praxis umgesetzt werden kann. In einigen Fällen werden bekannte Strukturen und Vorrichtungen in Form eines Blockdiagramms und nicht im einzelnen gezeigt, um die vorliegende Erfindung nicht undeutlich erscheinen zu lassen.

[0025] Einige Abschnitte der folgenden detaillierten Beschreibung sind mittels Algorithmen und symbolischen Darstellungen von Operationen mit Datenbits in einem Computerspeicher dargestellt. Diese algorithmischen Beschreibungen und Darstellungen sind diejenigen Mittel, die von Fachleuten bei der Datenverarbeitung verwendet werden, damit sie die Substanz ihrer Arbeit am effektivsten an andere Fachleute weitergeben können. Ein Algorithmus wird hier, und im allgemeinen, so verstanden, dass er eine in sich stimmige Folge von Schritten ist, die zu einem gewünschten Ergebnis führen. Die Schritte sind diejenigen, die physikalische Manipulationen physikalischer Größen erfordern. Normalerweise, aber nicht notwendigerweise, haben diese Größen die Form elektrischer oder magnetischer Signale, die gespeichert, übertragen, kombiniert, verglichen und auf andere Weise manipuliert werden können. Es hat sich schon immer als passend herausgestellt, hauptsächlich für den allgemeinen Gebrauch, diese Signale als Bits, Werte, Elemente, Symbole, Zeichen, Begriffe, Zahlen, etc. zu bezeichnen.

[0026] Es sollte jedoch beachtet werden, dass alle diese und ähnliche Begriffe mit den geeigneten physikalischen Größen zu assoziieren sind und nur geeignete Bezeichnungen sind, die auf diese Größen angewendet werden. Wenn es im folgenden nicht offensichtlich anders angegeben ist, ist es selbstverständlich, dass in der Beschreibung Bezeichnungen wie „Verarbeiten" oder „Rechnen" oder „Berechnen" oder „Bestimmen" oder „Zeigen" oder ähnliches sich auf die Aktionen und Prozesse eines Computersystems oder einer ähnlichen elektronischen Rechenvorrichtung beziehen, mit denen Daten, die als physikalische (elektronische) Größen in den Registern und Speichern des Computersystems vorliegen, zu anderen Daten, die ähnlich als physikalische Größen in den Speichern oder Registern des Computersystems oder in anderen Informationsspeicher-, -übertragungs- oder -anzeigevorrichtungen vorliegen, manipuliert und transformiert werden.

[0027] Die vorliegende Erfindung bezieht sich auch auf eine Vorrichtung zur Durchführung der Operationen. Diese Vorrichtung kann speziell für die geforderten Zwecke konstruiert sein, oder sie kann einen Computer für allgemeine Zwecke aufweisen, der von einem in dem Computer gespeicherten Computerprogramm selektiv aktiviert oder rekonfiguriert wird. Ein derartiges Computerprogramm kann in einem computerlesbaren Speichermedium gespeichert sein, beispielsweise – jedoch nicht beschränkt auf –

jede Art von Diskette einschließlich Floppy Disks, optischen Disketten, CD-ROMs, magnetooptischen Disketten, Read-Only Memories (ROMs), Random Access Memories (RAMs), EPROMs, EEPROMs, magnetischen oder optischen Karten, oder jeder Art von Medium, die zum Speichern elektronischer Befehle geeignet ist, jeweils an einen Computersystembus gekoppelt.

[0028] Die hier dargestellten Algorithmen und Anzeigen beziehen sich nicht inhärent auf einen bestimmten Computer oder eine andere Vorrichtung. Verschiedene Systeme für allgemeine Zwecke können mit Programmen gemäß der hier enthaltenen Lehre verwendet werden, oder es kann sich als zweckmäßig herausstellen, eine speziellere Vorrichtung zu konstruieren, um die geforderten Verfahrensschritte durchzuführen. Die erforderliche Struktur für eine Vielzahl dieser Systeme geht aus der nachfolgenden Beschreibung hervor. Außerdem wird die vorliegende Erfindung nicht unter Bezug auf eine bestimmte Programmiersprache beschrieben. Selbstverständlich können viele Programmiersprachen verwendet werden, um die hier beschriebene Lehre der Erfindung zu implementieren.

[0029] Fig. 1 stellt eine Systemarchitektur einer Ausführungsform für ein System **100** mit mehreren Kanälen und mehreren Diensten dar. Unter Bezug auf Fig. 1 ist das Systemelement **102** über einen Systembus **104** und eine Brücke **106** mit einer Vielzahl von Verarbeitungschips **108**, **110**, **112**, **114** verbunden. Außerdem ist die Brücke **106** mit dem Buffer-Speicher **116** verbunden. Das Systemelement kann eine andere Brückenkonfiguration **106** oder ein anderes geeignetes Bauteil sein. Die Brücke **106** ist über den Bus **118** mit den Verarbeitungschips **108**–**114** verbunden. Bei einer Ausführungsform sind die Verarbeitungschips **108**–**114** über den Bus **120** mit der TDM-Schnittstelle **122** verbunden. Bei anderen Ausführungsformen können die Chips **108**–**114** mit einer DS0-Schnittstelle oder einer anderen geeigneten Schnittstelle verbunden sein. Bei einer Ausführungsform ist die TDM-Schnittstelle **122** mit einer Reihe von Modulen und Ports verbunden, die auf dem TDM-Bus **124** installiert sind. Außerdem kann die TDM-Schnittstelle **122** wahlweise mit der TDM-Signaling-Schnittstelle **126** verbunden sein.

[0030] TDM ist eine Basisband-Technologie, bei der einzelne Kanäle von Daten oder Sprache in einem einzigen Strom von Bits (oder eingerahmten Bits) in einem Kommunikationskanal überlappt sind. Jeder Eingangskanal empfängt ein Überlappungs-Zeitsegment, damit alle Kanäle das Medium, das für die Übertragung verwendet wird, gemeinsam teilen. Wenn ein Kanal nichts zu senden hat, ist der Slot trotzdem für den Kanal vorgesehen und bleibt leer.

[0031] Bei einer Ausführungsform unterstützt ein Betriebssystem, das in dem System **100** mit mehreren Kanälen und mehreren Diensten läuft, Telekommunikations- und Datenkommunikationsanwendungen. Diese Anwendungen involvieren das Betreiben mehrerer Kanäle von Protokollstapeln, die aus mehreren Diensten aufgebaut sind. Das System **100** mit mehreren Kanälen und mehreren Diensten ermöglicht die dynamische Konfiguration von Diensten innerhalb der eingebetteten Telekommunikations- und Datenkommunikationsumgebung. Außerdem definiert das Betriebssystem automatisch die Zuordnung von Ressourcen für die Kanäle innerhalb des Systems **100**.

[0032] Bei einer Ausführungsform unterstützt das Betriebssystem, das in dem System **100** mit mehreren Kanälen und mehreren Diensten läuft, die Überwachung von Kanälen in Echtzeit. Wenn ein bestimmter Dienst in einem Kanal nicht wie erwartet antwortet, kann das Host-System von dem Betriebssystem fordern, dass es den Zustand eines oder aller seiner Dienste bei zuvor bestimmten Ereignissen an eine Off-Chip-Anwendung sendet. Beispielsweise nach der Verarbeitung jedes Framewerts von Daten. Die Daten werden gesammelt, ohne die Echtzeit-Leistung des Sockels zu beeinträchtigen. Die Off-Chip-Anwendung kann dann die Ursache des Problems durch Überprüfen der Daten in Nichtechtzeit analysieren.

[0033] Fig. 2 ist ein Blockdiagramm einer Ausführungsform für einen Verarbeitungschip **108**. Jeder Verarbeitungschip **108** enthält Cluster **202** und einen Hauptprozessor **204**. Jeder Cluster **202** enthält einen Clusterprozessor **208** und eine Reihe von Verarbeitungseinheiten (Processing Engines (PEs)) **210**. Der Hauptprozessor **204** ist so ausgestaltet, dass er alle Steuerungscodes und -operationen einschließlich des Empfangens von Steuerungsnachrichten von dem Host **102** und des Zuordnens von Kanälen zu den verschiedenen Clustern **202** durchführt.

[0034] Der Verarbeitungschip **108** weist auch ein gemeinsames Static Random Access Memory (gemeinsames SRAM) **206** auf. Auf das gemeinsame SRAM **206** kann direkt von allen Clusterprozessoren **202** und dem Hauptprozessor **204** zugegriffen werden. Ein in den Verarbeitungseinheiten (PEs) enthaltener Befehlsspeicher kann ebenfalls auf das gemeinsame SRAM **206** zugreifen. Das gemeinsame SRAM **206** wird verwendet, um Betriebssystem- und Anwendungscode zu speichern, und um die Daten für das Laufen des Codes auf dem Hauptprozessor **204** zu hosten.

[0035] Jeder Cluster **202** enthält ein Cluster-SRAM **212**. Das Cluster-SRAM **212** ist für das Halten der Kanaldaten verantwortlich, die auf jedem einzelnen Cluster **202** laufen. Das Cluster-SRAM **212** weist Eingangs-/Ausgangs-Buffer und Programmierstapel auf.

Das Betriebssystem des Systems **100** führt Speicherschutz durch, um zu verhindern, dass ein Kanal aus Versehen die Daten oder den Code eines anderen Kanals beschädigt.

[0036] Das externe Dynamic Random Access Memory (DRAM) **214** kann für Anwendungsdaten verwendet werden, die zu groß sind, um auf das On-Chip Cluster-SRAM **212** oder das gemeinsame SRAM **206** passen, und kann als Swap-Bereich für den Anwendungscode verwendet werden. Bei einer Ausführungsform können Anwendungen mehr Daten benötigen, als der On-Chip-Speicher unterstützen kann. In diesem Fall können die Daten und das Programm für einige der Dienste in einem Off-Chip-Speicher (beispielsweise dem DRAM **214**) gespeichert werden. Das Programm und die Daten werden in den On-Chip-Speicher geladen, wenn die Datenverarbeitung des Kanals beginnt. Auf diese Weise weiß der Dienst nicht, wo sich die Daten und das Programm in dem externen Speicher befinden. Dies wird durchgeführt, ohne die Echtzeit-Leistung der Anwendungen zu beeinträchtigen.

[0037] Jeder Verarbeitungschip **108** weist zwei leitungsseitige Ports **216** und zwei Systembus-Ports **218** auf. Diese Ports werden für paketseitigen Daten- und Steuerungstransport verwendet. Außerdem wird der Host-Port **220** zur Kommunikation mit dem Host **102** verwendet, und auf ihn kann nur von dem Hauptprozessor **204** und dem seriellen Boot-Port zugegriffen werden, der verwendet wird, um den Start-Datenstrom an den Chip zu senden.

[0038] Fig. 3 ist ein Blockdiagramm einer weiteren Ausführungsform für einen Teil eines Systems **100** mit mehreren Kanälen und mehreren Diensten. Unter Bezug auf Fig. 3 ist der Dienst **302** ein eigenständiger Satz von Befehlen mit einem Dateneingang/-ausgang, einer Steuerung und einer definierten Schnittstelle. Der Dienst **302** führt definiertes Verarbeiten einer bestimmten Menge und eines bestimmten Formats von Daten durch. Außerdem gibt der Dienst **302** eine bestimmte Menge und ein bestimmtes Format von Daten aus. Bei einer anderen Ausführungsform kann der Dienst **302** Daten auf bidirektionale Art und Weise verarbeiten. Der Dienststapel **304** ist ein verlinkter Satz von Diensten **302**, der eine größere Verarbeitungseinheit zur Verfügung stellt. Der Dienststapel **304** ist eine einzige, geordnete Sammlung von Diensten **302**, beispielsweise Echounterdrückungsdiensten, Tonermittlungsdiensten und Video- oder Sprachkonferenzdiensten. Die Dienste **302** innerhalb des Dienststapels **304** werden der Reihenfolge nach verarbeitet.

[0039] Der Sockel **306** ist eine virtuelle Konstruktion, die einen Satz von Diensten **302** in Form eines Dienststapels **304** bereitstellt. Das Betriebssystem verarbeitet die Dienste **302**, die in den Sockel **306** eingekapselt sind, einschließlich der Verbindung des Verkehrsflusses. Die Anzahl der Dienste **302** kann dynamisch angepasst und so definiert werden, dass die Notwendigkeit für Multitasking entfällt. Die Verarbeitung innerhalb des Sockels **306** ist datengetrieben. Das heißt, die Dienste **302** werden von den Sockeln **306** erst aufgerufen, nachdem die benötigten Daten bei dem Sockel **306** eingetroffen sind. Bei einer Ausführungsform können Anwendungen Protokollstapel durch Installieren eines Dienststapels **304** in einen Sockel **306** aufbauen. Die Dienste **302**, die Dienststapel **304** und die Sockel **306** werden nach Bedarf des Systems **100** zugeordnet und weggenommen.

[0040] Fig. 4 ist ein beispielhaftes Diagramm von Kanalsockeln **430** (**422**, **424**, **426**) in dem System **100**. Die Kanalsockel **430** sind Spezialsockel **306** die den Informationsfluss durch das System **100** zwischen zwei oder mehreren Vorrichtungen oder Endpunkten **402**, **404**, **406**, **408** leiten. Die Endpunkte können z.B. physikalische Vorrichtungen sein. Der Kanalsockel **430** ist ein Sockel **306**, der einen Dienststapel **304** aufnimmt und Kanaldaten verarbeitet. Der Kanalsockel **430** verbindet jeden leitungsseitigen Slot oder Buskanal an einem Ende des Kanalsockels **430** mit jedem anderen leitungsseitigen Slot oder Buskanal an dem entgegengesetzten Ende des Kanalsockels **430**. Der Kanalsockel **430** weist zwei Hauptattribute auf (1) einen definierten Eingang/Ausgang, der durch seine Funktion und Stelle impliziert ist, und (2) eine Programmierungs- und Anwendungsschnittstelle (Application Programming Interface (API)), wie sie durch eine mit dem Kanalsockel **430** verbundene Vorrichtung zu sehen ist. Der Kanalsockel **430** ist durch externe, physikalische Schnittstellenpunkte definiert und stellt die Fähigkeit bereit, den Dienststapel **304** zu verarbeiten. Informationen können von einem physikalischen Endpunkt **402** über eine Verbindung **418** zu dem Kanalsockel **424** fließen. Die Informationen werden von den Diensten **302** in dem Kanalsockel **424** verarbeitet und über die Verbindung **420** an den Endpunkt **406** weitergeleitet. Das Betriebssystem kann den Informationsfluss durch verschiedene Kanalsockel **430** in Abhängigkeit von den Bedürfnissen der Endpunkte **402**–**408** dynamisch verändern. Beispielsweise können Daten zunächst so gesetzt werden, dass sie von dem Endpunkt **404** über die Verbindung **410** durch den Kanalsockel **422** und über die Verbindung **412** zu dem Endpunkt **408** fließen. Wenn jedoch der Dienststapel **304** in dem Kanalsockel **422** mit den Daten nicht kompatibel ist, benachrichtigt der Kanalsockel **422** das Betriebssystem, den Fluss zu unterbrechen und die Information zurückzuleiten. Das Betriebssystem leitet dann den Fluss an einen bestehenden Kanalsockel **430** mit dem richtigen Betriebsstapel zurück oder generiert einen neuen Kanalsockel **430**. Unter Bezug auf Fig. 4 kann das Betriebssystem den Fluss über die Verbindung **414**, den Kanalsockel **426** und die

Verbindung **416** von dem Endpunkt **404** zu dem Endpunkt **408** zurückleiten.

[0041] Ein Kanalsockel **430** wird durch die externen, physikalischen Schnittstellen-Endpunkte **402**, **404**, **406** und **408** und die Daten, die durch den Kanalsockel **430** fließen, definiert. Die Endpunkte **402**–**408** können unterschiedliche physikalische Vorrichtungen oder dieselbe physikalische Schnittstelle oder Vorrichtung sein. Der Informationsfluss wird durch die Art und Weise geleitet, in der die Paketformate erzeugt werden. Die Header-Informationen in den Paketen geben die eindeutigen Endpunkte **402**–**408** an, an die die Informationen gesendet werden, und ob die Informationen in das System **100** hinein oder daraus hinaus gehen. Beispielsweise können die Dienste des Kanalsockels **422** eine Konvertierung von Daten durchführen. Der Mechanismus des Kanalsockels **430** ermöglicht es, dass ein Dienststapel **304** in den Informationsfluss eingebaut wird, in dem die Dienste **302** die Daten bei ihrem Fluss durch das System leiten oder verarbeiten können. Beispielsweise wartet, wenn ein erster Dienst einen 40-Byte Datenframe ausgibt und ein zweiter Dienst einen 80-Byte Frame verwendet, bei einer Ausführungsform der zweite Dienst, bis der erste Dienst genügend Daten ausgegeben hat, damit der zweite Dienst die Daten verarbeiten kann. Bei einer anderen Ausführungsform verzögert der erste Dienst das Senden von Daten an den zweiten Dienst, bis er genügend Daten angesammelt hat. Die Dienste **302** sind unabhängige Module und eigenständige Plugins. Daher können bei einer Ausführungsform die Dienste **302** dynamisch in das gemeinsame SRAM **206** in Echtzeit heruntergeladen werden, um nach Bedarf der Daten Kanalsockel **430** zu erzeugen.

[0042] Da die Sockel **306** durch das Betriebssystem dynamisch zugeordnet und weggenommen werden können, können Anwendungen geschrieben werden, um auf festgeschaltete Einzelkanal-Prozessoren zuzugreifen; die festgeschalteten Kanäle werden jedoch auf mehreren physikalischen Kanälen laufen. Somit stellt der Mechanismus des Kanalsockels **430** eine Einzelkanal-Programmierung mit mehrfacher Kanalausführung bereit. Außerdem kann einen Anwendung geschrieben werden, um einen Informationsfluss zwischen den Endpunkten **402**–**408** unabhängig von dem Betriebssystem und unabhängig von der Art der Daten, die verarbeitet werden, bereitzustellen. Die Funktionen des Kanalsockels **430** sind, ob sie Signalverarbeitungsfunktionen oder Paketverarbeitungsfunktionen sind, sowohl von dem Betriebssystem als auch von der Hardware-Konfiguration unabhängig. Der Mechanismus befreit die Anwendungen auch von dem Management der Kanäle und legt das Management in das Betriebssystem, wodurch kanalunabhängige Anwendungen erzeugt werden. Außerdem ermöglicht es der Mechanismus des Kanalsockels **430**, dass die Anwendungen und Dienste **302** plattformunabhängig sind.

[0043] Fig. 5 ist ein Blockdiagramm einer anderen Ausführungsform für einen Teil eines Systems **100** mit mehreren Kanälen und mehreren Diensten. Unter Bezug auf Fig. 5 weist das System **100** einen Dienststeuerungssockel **502** auf, der mit einem Host und einer Vielzahl von Kanalsockeln **510** verbunden ist. Der Dienststeuerungssockel **502** ist ein Sockel **306**, der den Steuerungsabschnitt der Dienste **302** für einen Dienststapel **304** enthält. Jeder einzelne Dienststapel **504** hat einen eigenen Dienststeuerungssockel **502**. Jeder Dienststeuerungssockel **502** steuert mehrere Instanzen desselben Kanalsockels **510**. Jeder Dienst **302** in dem Dienststeuerungssockel **502** ist der Steuerungsabschnitt für den jeweiligen Dienst **302** in dem Kanalsockel **510**. Die Dienste **302** in einem Dienststapel des Kanalsockels **510** können Steuerungsnachrichten von dem Dienststeuerungssockel **502** dieses Stapels empfangen. Jeder Dienst **302** weist eine Datendomain und eine Steuerungsdomain auf. Die Datendomain ist in dem Sockel **306** enthalten, und die Steuerungsdomain ist in dem Dienststeuerungssockel **502** enthalten.

[0044] Fig. 6 ist ein Blockdiagramm für eine weitere Ausführungsform eines Systems **100** mit mehreren Kanälen und mehreren Diensten. Unter Bezug auf Fig. 6 weist das System **100** einen Dienststeuerungssockel **601** auf, der Steuerungsinformationen von einem Host bei **622** empfängt. Jeder Dienst **302** hat einen Daten- und Steuerungs-Bestandteil. Die Steuerungsinformationen bestimmen, dass die Steuerungsinformationen an einen bestimmten Dienst **302** in einem bestimmten Sockel **306** gesendet werden sollen. In dem Beispiel der Fig. 6 steuert eine Dienststeuerungseinrichtung **624** alle Informationen für diesen bestimmten Dienst **626**, **628**, **630**, in denen der Dienst **626**, **628**, **630** eine Instantiation desselben Dienstes **302** ist. Der Dienststeuerungssockel **601** enthält alle Steuerungsbefehle für den bestimmten Dienst **626**, **628**, **630**. Daher steuert der Dienststeuerungssockel **601** alle Dienste, die in jedem Sockel **602**, **604** und **606** enthalten sind. Die Sockel **602**, **604** und **606** haben jeweils denselben Dienststapel **304**. Die Dienststeuerungseinrichtung **624** steuert die Dienste **626**, **628** und **630**. Wenn ein Befehl von dem Host empfangen wird, gibt der Befehl an, dass sowohl an den Dienst **302** als auch an den Sockel **306** die Steuerungsinformation zu senden ist. Somit gibt bei der Ausführungsform der Fig. 6 die von dem Host empfangene Information an, dass auf einen bestimmten Dienst (**626**, **628** oder **630**) zugegriffen werden soll, und dass auf den Sockel **602** zugegriffen werden soll. Somit sendet der Dienststeuerungssockel **502** die Informationen über die Verbindung **608** an den Dienst **626** in dem Sockel **602**.

[0045] Fig. 7 ist ein Blockdiagramm einer Ausführungsform für die Konfiguration des Datenaggregati-

onssockels **730**. Unter Bezug auf Fig. 7 ist der Datenaggregationssockel **730** mit einer Reihe von Sockeln **702**, **704**, **706** verbunden. Jeder Sockel **702**, **704**, **706** empfängt Datenframes jeweils durch die Eingänge **708**, **710**, **712**. Jeder Sockel **702**, **704**, **706** sendet jeweils durch die Verbindungen **720**, **722**, **724** Daten an den Datenaggregationssockel **730**. Die Dienste **302** in dem Datenaggregationssockel **730** sammeln und kombinieren die Daten und übertragen die Daten über die Verbindung **726** von dem Datenaggregationssockel **730** zu den Sockeln **702**, **704**, **706**. Jeder Sockel **702**, **704**, **706** gibt jeweils Daten über die Verbindungen **714**, **716**, **718** aus. Der Datenaggregationssockel **730** sammelt Daten von mehreren Sockeln und verarbeitet die angesammelten Daten. Beispielsweise kann der Datenaggregationssockel **730** verwendet werden, um Telekonferenzen und andere Anwendungen, die Datenaggregation erfordern, zu verarbeiten.

[0046] Bei einer Ausführungsform sendet ein Host eine Anfrage zur Zuordnung eines neuen Datenaggregationssockel **730** unter Angabe der maximalen Zahl von Eingängen, der bestimmten zu leistenden Dienste, und der Framegröße. Beispielsweise kann der Host eine Anfrage zur Zuordnung eines Datenaggregationssockels **730** für eine Telekonferenz senden. Vor der Initialisierung des Datenaggregationssockels **730** ordnet eine Host-Anwendung auch geeignete Sockel **510**, wie oben beschrieben, zu. Die Sockel **702**, **704**, **706** verbinden spezifische Daten, die in die Telekonferenz-Vorrichtung **730** eingegeben werden. Wenn sowohl der Datenaggregationssockel **730** also auch die Sockel **702**, **704**, **706** zugeordnet sind, verbindet der Host die Sockel **702**, **704**, **706** mit dem Datenaggregationssockel **730**. Bei einer Ausführungsform schaltet die Host-Anwendung Zeiger (pointer) in der Software zur Verbindung mit dem Datenaggregationssockel **730**. Wenn ein Frame von Daten vorhanden ist, empfängt der Datenaggregationssockel **730** die Informationen und verarbeitet die Informationen über die Dienste in dem Datenaggregationssockel **730**. Der Datenaggregationssockel **730** gibt die angesammelten Daten an jeden Sockel **702**, **704**, **706** aus. Der Datenaggregationssockel **730** kann dynamisch zugeordnet und weggenommen werden. Die Verbindung zwischen dem Datenaggregationssockel **730** und den einzelnen Sockeln **702**, **704**, **706** kann dynamisch hergestellt oder unterbrochen werden.

[0047] Fig. 8 ist ein Blockdiagramm einer Ausführungsform für Sockeldaten **800**, die von dem System **100** mit mehreren Kanälen und mehreren Diensten verwendet werden. Die Sockeldaten **800** weisen von 1 (**804**) bis n (**808**) Sockel **820** auf. Außerdem weisen die Sockeldaten **800** einen gemeinsamen Speicher **810** auf. Die Sockeldaten **800** werden verwendet, um Sockel **306** zu speichern, wenn sie dynamisch erzeugt, initialisiert und verwendet werden. Bei einer Ausführungsform befinden sich die Sockeldaten **800** in dem Clusterspeicher **212**. Bei anderen Ausführungsformen können sich die Sockeldaten **800** in dem gemeinsamen SRAM **206** oder dem externen DRAM **214** befinden. Das DRAM **214** wird verwendet, wenn die Sockeldaten **800** nicht in den Clusterspeicher **212** passen. Bei einer Ausführungsform können sich die Sockeldaten **800** in dem DRAM **214** befinden und bei Bedarf in den Clusterspeicher **212** übertragen werden.

[0048] Bei einer Ausführungsform wird jedem Dienst **302** eine Art zugeordnet. Wenn Daten durch den Sockel **306** geleitet werden, können die Steuerungsinformationen für die Daten so konfiguriert werden, dass sie nur durch eine vorgegebenen Art von Dienst **302**, z.B. einem Dienst **302** der Art 2, verarbeitet werden. Außerdem kann der Dienst **302** einen Sockel **306** instruieren, bestimmte Dienste **302**, die sich weiter entlang des Sockels **306** befinden, nur für diesen Frame von Daten nicht durchzuführen oder „auszustecken". Nach der Verarbeitung des Frames von Daten werden die Sockel **306** für die folgenden Frames von Daten wieder „eingesteckt".

[0049] Außerdem ist es während des Betriebs eines Sockels **306** manchmal erforderlich, dass ein Dienst **302** mit einem anderen Dienst **302** in dem Dienststapel **304** kommuniziert. Die Dienste **302** können unter Verwendung eines sockelweiten gemeinsamen Speichers **810** miteinander kommunizieren. Die Dienste **302** können Steuerungsinformationen und -daten an andere Dienste **302** in einem Sockel **306** über einen gemeinsamen Speicher **810** weitergeben. Wenn z.B. in einem anfänglichen Sockeldienst eine Ruftonermittlung gefunden wird, sollte der Sockel **306** den Sprachdecodierungsdienst nicht betreiben. Somit kann der anfängliche Tonermittlungsdienst die Information, dass der Ton ermittelt worden ist, in dem gemeinsamen Speicher **810** platzieren. Wenn ein folgender Dienst **302** initialisiert wird oder läuft, liest der Dienst **302** den gemeinsamen Speicher **810** und stellt fest, dass eine Tonermittlung gefunden wurde und die Dienste vorbeigeleitet wurden. Der gemeinsame Speicher **810** wird zu dem Zeitpunkt zugeordnet, zu dem ein Sockel **306** zugeordnet wird. Ein Zeiger auf den gemeinsamen Speicher **810** wird in die Initialisierungsroutine jedes Dienstes **302** eingeführt. Der Dienst **302** verwendet diesen Zeiger als notwendig, um mit anderen Diensten **302** zu kommunizieren. Die Größe des gemeinsamen Speichers **810** ist für alle Sockel **306** gleich und ist innerhalb des Konfigurations-Setups spezifiziert.

[0050] Fig. 9a ist ein Blockdiagramm einer Ausführungsform für eine Konfiguration eines Steuerungsaggregationssockels **908**. Unter Bezug auf Fig. 9a sammelt der Steuerungsaggregationssockel **908** alle Steuerungsinformationen, die aus den Steuerungssockeln kommen, und sendet die gesammelten Infor-

mationen an einen Host. Außerdem sammelt der Steuerungsaggregationssockel **908** die Steuerungsinformationen, die in das System hineinkommen, und verteilt die gesammelten Steuerungsinformationen an den geeigneten Steuerungssockel, entweder den Dienststeuerungssockel **902**, **904**, oder den Plattformsteuerungssockel (PSS) **906**. Bei einer Ausführungsform können Benutzer die Dienste **302** in dem Steuerungsaggregationssockel **908** installieren oder verändern. Der Steuerungsaggregationssockel **908** empfängt Nachrichten von der Host-Steuerung und leitet die Nachrichten an den entsprechenden Dienst in dem entsprechenden Sockel **306** weiter. Der Steuerungsaggregationssockel **908** teilt die Host-Steuerung in Abhängigkeit von der Adresse der Steuerungsinformationen auf den entsprechenden Sockel **306** auf. Bei einer Ausführungsform ist die Adresshierarchie Untersystem, Board, Chip, Sockel und Dienst. Der Steuerungsaggregationssockel **908** sendet die Steuerungsinformationen an den von dem Host angegebenen entsprechenden Dienst **302** und Sockel **306**.

[0051] Der Plattformsteuerungssockel **906** ist ein spezialisierter Sockel, der auf dem Hauptprozessor läuft, wenn das System startet. Er ist der einzige Sockel **306**, der Kenntnis der systemweiten Ressourcen hat. Der Plattformsteuerungssockel **906** managt alle Ressourcen, einschließlich der Zuordnung der Dienststeuerungssockel **902**, **904** zu den Clustern **202**, der Zuordnung der TDM-Zeitslots, und der Zuordnung der Bus-Kanäle. Anwendungen können innerhalb des Plattformsteuerungssockels **906** keine Dienste zuordnen oder entfernen. Insbesondere startet der Plattformsteuerungssockel **906** die Cluster **202** und die Chips **108**, lädt und entlastet die Dienste **302**, erzeugt und zerstört die Dienststeuerungssockel **902**, **904**, sendet dem Host **102** ein Heartbeat, und ermittelt, ob ein Cluster **202** außer Betrieb ist.

[0052] Der Plattformsteuerungssockel **906** überwacht die Ressourcen auf dem Chip einschließlich des verfügbaren Befehlsspeichers. Da sich das Verkehrsmuster erhaltener Dienste ändert, kann das Betriebssystem Dienste nach Bedarf laden und entlasten. Dies wird durchgeführt, ohne die Kanäle, die in dem System laufen, zu beeinträchtigen.

[0053] Fig. 9b ist eine weitere Ausführungsform für eine Konfiguration eines Steuerungsaggregationssockels **908**. Der Steuerungsaggregationssockel **908** ist sowohl mit dem Plattformsteuerungssockel **906** als auch mit einer Reihe von Dienststeuerungssockeln **902**, **904** verbunden. Jeder Dienststeuerungssockel **902**, **904** ist mit einer Reihe von Kanalsockeln **910** verbunden.

[0054] Fig. 10 ist ein Flussdiagramm einer Ausführungsform für die Verarbeitung von Daten und Informationen durch Kanalsockel **510**. Zuerst ordnet der Plattformsteuerungssockel **906** beim Verarbeitungsblock **1005** den Kanalsockel **510** auf Anfrage des Hosts **102** dynamisch zu. Die Erzeugung eines Kanalsockels **510** wird unten unter Bezug auf Fig. 11 beschrieben.

[0055] Nachdem der Kanalsockel **510** zugeordnet worden ist, werden Daten von dem Kanalsockel **510** empfangen. Bei einer Ausführungsform kann der Kanalsockel **510** Steuerungsinformationen für die Verarbeitung der Daten von dem Dienststeuerungssockel **902** empfangen. Daten können von jeder physikalischen Vorrichtungsschnittstelle, die mit dem System **100** verbunden ist, empfangen werden. Die Daten werden durch die Dienste **602** in einem Sockel **306** verarbeitet. Das Betriebssystem innerhalb des Systems **100** kann den Informationsfluss durch verschiedene Kanalsockel **510** in Abhängigkeit von dem Bedarf der physikalischen Vorrichtungen, die mit den Kanalsockeln **510** verbunden sind, verändern. Wenn ein Dienststapel **304** innerhalb des Kanalsockels **510** mit den ankommenden Daten nicht kompatibel ist, benachrichtigt der Kanalsockel **510** das Betriebssystem, den Informationsfluss zu ändern. Das Betriebssystem leitet den Fluss dann an einen anderen bestehenden Kanalsockel **510** mit den geeigneten Dienststapeln **304** zurück oder erzeugt einen neuen Kanalsockel **510**.

[0056] Bei dem Verarbeitungsblock **1051** wird der Datenframe durch die Dienste **302** in dem Kanalsockel **510** verarbeitet. Bei einer Ausführungsform werden die Dienste **302** dynamisch zugeordnet, wenn der erste Frame von Daten von dem Kanalsockel **510** empfangen worden ist. Bei einer anderen Ausführungsform können die Dienste **302** zu der Zeit zugeordnet werden, zu der der Kanalsockel **510** zugeordnet wird. Die Dienste **302** verarbeiten die Daten in Abhängigkeit von den Erfordernissen der Daten. Beispielsweise kann der Dienst **302** dynamisch zugeordnet werden, um Telefonsprachdaten zu verarbeiten. Bei einer Ausführungsform können Daten unter Verwendung eines Datenaggregationssockels **730** gesammelt werden, um die Daten zu kombinieren. Der Datenaggregationssockel **730** kann bei einer Telekommunikations-Telekonferenz-Anwendung verwendet werden.

[0057] Beim Verarbeitungsblock **1020** werden die verarbeiteten Datenframes an die entsprechenden Vorrichtungsschnittstellen ausgegeben. Die Verarbeitungsblöcke **1015** und **1020** werden so lange ausgeführt, wie Datenframes geliefert werden. Nachdem alle Datenframes verarbeitet worden sind, kann das System **100** den Kanalsockel **510** dynamisch wegnehmen. Das System **100** ordnet die weggenommenen Dienste **302** und Sockel **306** nach Bedarf des Systems **100** dynamisch zu, um den Vorteil der begrenzten physikalischen Kanäle in dem System **100** voll auszunutzen. Daher arbeitet das System **100** als

Plattform mit mehreren Kanälen und mehreren Diensten in der Entwicklungsumgebung eines einzigen Kanals.

[0058] Fig. 11 ist ein Flussdiagramm einer Ausführungsform zum Erstellen von Kanalsockeln **510**. Zuerst sendet der Host **102** beim Verarbeitungsblock **1105** ein Steuerungspaket an den Plattformsteuerungssockel **906**. Das Steuerungspaket gibt die Art des Kanalsockels **510** an, den der Host **102** zuordnen muss, um einen gegebenen Datenfluss zu verarbeiten. Beispielsweise kann es erforderlich sein, dass der Host **102** ein Leitung-zu-Paket-, ein Paket-zu-Paket-, oder ein Leitung-zu-Leitung-Steuerungspaket zuordnen muss. Das Paket enthält eine von oben nach unten geordnete Namensliste der Dienste **302**, die dem Sockel **306** zugeordnet werden sollen, um einen Dienststapel **304** zu erzeugen. Der Plattformsteuerungssockel **906** bestimmt, ob die erforderlichen Dienste **302** geladen worden sind, und ob die erforderlichen Dienste **302** in dem Betriebssystem registriert sind. Wenn ein Dienst **302** nicht verfügbar ist, informiert der Plattformsteuerungssockel **606** den Host **102**. Außerdem bestimmt der Plattformsteuerungssockel **906**, ob Ressourcen verfügbar sind, um die erforderlichen Sockel **306** zuzuordnen. Wenn keine Ressourcen verfügbar sind, informiert der Plattformsteuerungssockel **906** den Host **102**, dass das Betriebssystem nicht über genügend Ressourcen verfügt, um einen Sockel zuzuordnen.

[0059] Beim Verarbeitungsblock **1120** ermittelt der Plattformsteuerungssockel **906**, ob bereits ein Dienststeuerungssockel **902**, **906** mit demselben Dienststapel **304** existiert. Wenn bereits ein Dienststeuerungssockel **902**, **904** existiert, geht die Verarbeitung beim Verarbeitungsblock **1130** weiter. Wenn noch kein Dienststeuerungssockel existiert, geht die Verarbeitung beim Verarbeitungsblock **1125** weiter.

[0060] Beim Verarbeitungsblock **1125** ordnet der Plattformsteuerungssockel **906** die entsprechenden Dienststeuerungssockel für die Dienstekonfiguration zu. Bei einer Ausführungsform lädt der Plattformsteuerungssockel **906** die entsprechenden Steuerungsdienste in der von dem Dienststapel geforderten Reihenfolge in das gemeinsame SRAM.

[0061] Beim Verarbeitungsblock **1130** benachrichtigt der Plattformsteuerungssockel **906**, nachdem der Plattformsteuerungssockel **906** den neuen Dienststeuerungssockel **902** erstellt hat, die Host-Anwendung, dass der Dienststeuerungssockel erstellt ist. Der Plattformsteuerungssockel erstellt dann den Kanalsockel. Der Dienststeuerungssockel vervollständigt sodann die Kanaldienstsockel-Zuordnung, indem er Sockelparameter an den Sockel sendet. Diese Parameter können z.B. die Tail-Länge der Echounterdrückung umfassen.

[0062] Beim Verarbeitungsblock **1135** initialisiert der Kanalsockel die Dienste **302** durch Aufrufen ihrer Initialisierungsfunktionen. Bei einer anderen Ausführungsform können die Dienste **302** zu dem Zeitpunkt zugeordnet werden, zu dem der erste Datenframe von dem Kanalsockel **510** empfangen wird. Wenn der Host **102** bereit ist, die Datenverarbeitung durch einen gegebenen Sockel **306** zu beginnen, instruiert der Host **102** den Dienststeuerungssockel **502**, den Kanalsockel **510** zu starten. Dies initialisiert die Datenverarbeitung des Sockels **510**. Der Dienststeuerungssockel **502** startet den Kanalsockel **510** und informiert den Host **102**, dass der Kanalsockel **510** konfiguriert ist und läuft.

[0063] Fig. 12 ist ein Flussdiagramm einer Ausführungsform zum Erzeugen eines Datenaggregationssockels **730**. Vor der Erzeugung des Datenaggregationssockels **730** ordnet der Host **102** die entsprechenden Kanalsockel **510** zum Verarbeiten der Daten beim Verarbeitungsblock **1220** zu. Die Zuordnung der Kanalsockel **510** ist vorstehend unter Bezug auf Fig. 11 beschrieben.

[0064] Unabhängig von der Zuordnung des Kanalsockels **510** fordert der Host **102** auch, dass der Datenaggregationssockel **730** zugeordnet wird. Beim Verarbeitungsblock **1215** sendet der Host eine Anfrage zur Zuordnung eines neuen Datenaggregationssockels **730** mit der maximalen Anzahl von Eingängen, den bestimmten, zu laufenden Diensten, und der Framegöße. (Die Framegöße ist die Anzahl von Abtastungen an jedem der verbundenen Eingänge). Wenn sowohl der Datenaggregationssockel **730** als auch einige Kanalsockel **510** zugeordnet sind, geht die Verarbeitung beim Verarbeitungsblock **1225** weiter. Beim Verarbeitungsblock **1225** verbindet der Host die Kanalsockel **510** mit dem Datenaggregationssockel.

[0065] Beim Verarbeitungsblock **1230** sammelt der Datenaggregationssockel **730** die Daten durch Aufrufen von Dienststeuerungsprogammen in dem Datenaggregationssockel **730**, sobald ein Framewert von Daten an jedem der verbundenen Eingänge von den Kanalsockeln **510** verfügbar ist. Der Datenaggregationssockel **730** sammelt die Daten und gibt die spezifischen Daten aus. Die Daten werden von dem Server **302** in einem Sockel **510** gesammelt. Bei einer Ausführungsform wird die Aggregation vor der Herstellung durch einen Kunden oder Benutzer definiert.

[0066] Beim Verarbeitungsblock **1235** überträgt der Ausgang des Dienstes **302** des letzten Datenaggregationssockels **730** die Daten an die Eingänge aller verbundenen Kanalsockel **510**. Die gesammelten Daten werden durch die einzelnen Kanalsockel **510** verarbeitet. Der Datenaggregationssockel **730** kann dynamisch erzeugt oder zerstört werden, und die Verbindung zwischen dem Datenaggregationssockel

730 und den Kanalsockeln **510** kann dynamisch hergestellt werden.

[0067] Fig. 13 ist ein Flussdiagramm einer Ausführungsform zum Schalten der Sockel **306** zwischen den Dienststeuerungssockeln **502**. Wenn ein Sockel **306** oder ein Dienst **302** in einem Sockel ermittelt, dass der gegenwärtige Kanalsockel **510** von einem anderen Dienststapel **304** bearbeitet werden muss, muss das Betriebssystem zu einem anderen Dienststapel **304** schalten. Wenn z.B. ein Sprach-Dienststapel ermittelt, dass ein Anruf auf einen Fax-Dienststapel geschaltet werden muss, muss ein neuer Dienststapel **304** verwendet werden. Zuerst sendet der Dienst **302** beim Verarbeitungsblock **1305** eine Steuerungsnachricht an seinen Dienststeuerungssockel **502**, die an den Plattformsteuerungssockel **906** weitergeleitet wird. Die Nachricht kann Konfigurationsinformationen enthalten, die der neue Dienststapel **304** ggf. empfangen muss, und die Nachricht enthält die Information, dass ein neuer Stapel **304** verwendet werden muss.

[0068] Beim Verarbeitungsblock **1310** entfernt der Kanalsockel **510** alle seine vorexistierenden Dienste. Beim Verarbeitungsblock **1315** kommuniziert der Plattformsteuerungssockel **906** mit dem Host **102**, um einem anderen Dienststeuerungssockel **502** einen Sockel **306** zuzuordnen. Der Plattformsteuerungssockel **906** ordnet einem anderen Dienststeuerungssockel **502** auf der Grundlage des neuen Dienststapels **304** einen Sockel **306** zu. Wenn der Dienststapel **304** nicht existiert, kann der Plattformsteuerungssockel **906** auf der Grundlage der Erfordernisse des neuen Dienststapels **304** einen neuen Dienststeuerungsstapel **502** erzeugen.

[0069] Beim Verarbeitungsblock **1320** sendet der Dienststeuerungssockel **502** die Dienststapelinformationen an den Sockel **306**. Beim Verarbeitungsblock **1325** ordnet der Sockel **306** den neuen Dienststapel **304** zu und informiert den Dienststeuerungssockel **502**, dass der Stapel **304** zur Verarbeitung zur Verfügung steht.

[0070] Beim Verarbeitungsblock **1330** überträgt der Dienststeuerungssockel **502** Konfigurationsparameter und ein Startsignal an den neuen Sockel **306**.

[0071] Somit können Kanalsockel **510** nach Bedarf dynamisch erzeugt werden. Der Dienststeuerungssockel **502** steuert den physikalischen oder Hardware-Kanal, während die Kanalsockel **510** die Daten verarbeiten. Das Ressourcen-Management zur Ermittlung der Kosten von n Kanalsockeln **510** und zum Ermitteln der Verfügbarkeitsressourcen um festzustellen, ob n Kanalsockel **510** betrieben werden können, kann von dem Plattformsteuerungssockel **906** oder von entfernteren Ressourcen mit Kenntnis der Ressourcen innerhalb des Systems **100** vorgenommen werden. Das Betriebssystem kann somit zur Verarbeitung von Datenarten mit mehreren Kanälen einen einzigen Hardware-Kanal verwenden.

[0072] Bei einer Ausführungsform werden drei Arten von Kanalsockeln **510** eingesetzt: Leitung-zu-Paket, Paket-zu-Paket, und Leitung-zu-Leitung. Leitung-zu-Paket-Sockel sind immer mit der Leitungsseite an einem Ende des Pakets und der Paketseite am anderen Ende des Kanals verbunden. Einige Sprach- und Modem-Stapel verwenden Leitung-zu-Paket-Sockel. Diese Sockel verarbeiten einen Framewert von Daten. Die Framegröße wird als Anzahl von Bytes in der Leitungsseite spezifiziert. Einer der Dienste ist für das Einstellen der Framegröße verantwortlich. Beispielsweise sollen im Falle einer Sprachverarbeitung die Codes die Framegröße einstellen. Die Framegröße kann jederzeit während der Lebensdauer des Sockels verändert werden. Der Sockel liest einen Framewert von Daten. Die leitungsseitig empfangenen Daten können ein Format aufweisen, das zu dem von den Diensten erwarteten Format unterschiedlich ist. In diesem Fall führt das Betriebssystem die entsprechende Datenkonvertierung durch. Nach der Datenkonvertierung ruft der Sockel die Datenverarbeitungsfunktionen jedes Dienstes der Reihe nach auf. Die Argumente für diese Funktionen sind die Zeiger für die Daten in dem gemeinsamen Speicher und die Datenlänge in Bytes.

[0073] Nachdem die Datenverarbeitungsfunktionen für alle Dienste aufgerufen worden sind, erwartet der Sockel ein Paket von Eingangsdaten von dem Paket-Netz. Wenn ein Paket eingetroffen ist, konvertiert der Sockel die Basiszelldaten in flache Daten und ruft die Datenverarbeitungsfunktion aller Dienste auf. Die Argumente für diese Funktionen sind die Zeiger für die Daten und die Datenlänge in Bytes. Sind jedoch keine Pakete eingetroffen, ruft der Sockel diese Funktionen mit der auf Null eingestellten Datenlänge auf. Dies ermöglicht es einem Sockel, sinnvolle Daten über die Leitung zu schicken, auch wenn zu diesem Zeitpunkt kein Paket verfügbar ist. Sowohl der Sprachstapel als auch der Modemstapel erfordern diesen Mechanismus.

[0074] Paket-zu-Paket-Sockel sind immer mit der Paketseite an beiden Enden des Sockels verbunden. Die Paketlänge kann jedoch in jeder Richtung des Datenflusses unterschiedlich sein. Beide Enden der Sockel verarbeiten Paketgrößendaten, und die Verarbeitung findet immer dann statt, wenn Daten an einem Ende auftauchen.

[0075] Leitung-zu-Leitung-Sockel sind mit der Leitungsseite an beiden Enden des Sockels verbunden. Der Sockel verarbeitet einen Framewert von Daten in beiden Richtungen des Datenflusses. Die Framegröße ist in jeder Richtung gleich. Leitung-zu-Leitung-Sockel können leitungsseitige Standard-Codie-

rungs- und Decodierungstransformationen der Daten an beiden Enden durchführen. Die Konvertierungseinstellung braucht an den beiden Enden nicht gleich zu sein.

[0076] In der vorstehenden Beschreibung ist die Erfindung unter Bezug auf spezifische beispielhafte Ausführungsformen dargelegt worden. Es ist jedoch offensichtlich, dass verschiedene Modifizierungen und Veränderungen durchgeführt werden können, ohne vom breiteren Umfang der Erfindung, wie er in den angefügten Ansprüchen dargelegt ist, abzuweichen. Die Beschreibung und die Zeichnungen sind demgemäss in illustrativem und nicht in einschränkendem Sinn zu sehen.

Patentansprüche

1. Verfahren zum Verarbeiten eines Datenflusses in einer Umgebung mit mehreren Kanälen und mehreren Diensten, das umfasst:
– dynamisches Zuordnen wenigstens eines Sockels (**304**, **306**, **308**; **602**, **604**, **608**), wobei der wenigstens eine Sockel (**304**, **306**, **308**; **602**, **604**, **608**) wenigstens einen dynamisch zugeordneten Dienst (**302**) aufweist; und
– Verarbeiten der Daten unter Verwendung des wenigstens einen Dienstes (**302**) auf der Grundlage einer Datenart, die verarbeitet wird;
– Zuordnen wenigstens zweier Kanalsockel (**504**, **506**, **508**) für eine Datenart;
– Zuordnen eines Datenaggregationssockels (**702**, **704**, **706**; **902**, **904**, **906**) für die Datenart;
– Ansammeln von Daten im Ansprechen auf das Empfangen der Daten von den wenigstens zwei Kanalsockeln (**504**, **506**, **508**); und
– Ausgeben der angesammelten Daten an die wenigstens zwei Kanalsockel (**504**, **506**, **508**).

2. Verfahren nach Anspruch 1, wobei das Zuordnen des Datenaggregationssockels weiterhin umfasst:
Übermitteln einer Anfrage zum Zuordnen des Datenaggregationssockels (**702**, **704**, **706**; **902**, **904**, **906**).

3. Verfahren nach Anspruch 1, wobei das Zuordnen eines Kanalsockels weiterhin umfasst:
Feststellen, ob bereits ein Kanalsockel für eine Datenart existiert;
wenn kein Kanalsockel existiert,
Zuordnen eines Dienststeuerungssockels (**604**, **606**, **608**) für die Datenart; und
Benachrichtigen eines Hosts, dass der Dienststeuerungsstapel generiert worden ist.

4. Verfahren nach Anspruch 3, das weiterhin das Empfangen eines Steuerungspakets von einem Host aufweist.

5. Verfahren nach Anspruch 3, wobei das Benachrichtigen weiterhin das Senden von Sockelparametern zum Kanalsockel (**504**, **506**, **508**) aufweist.

Es folgen 14 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

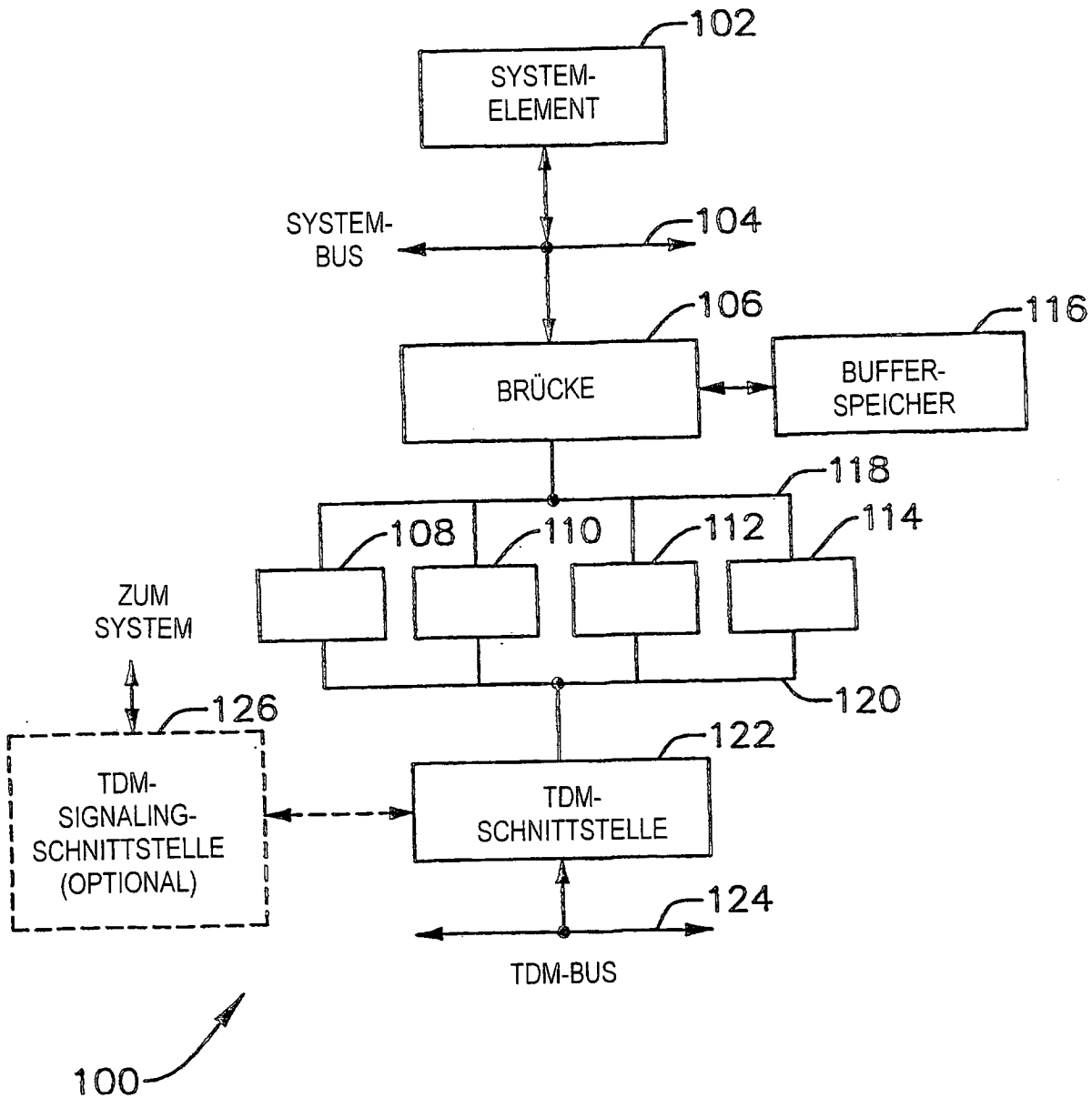


FIG. 2

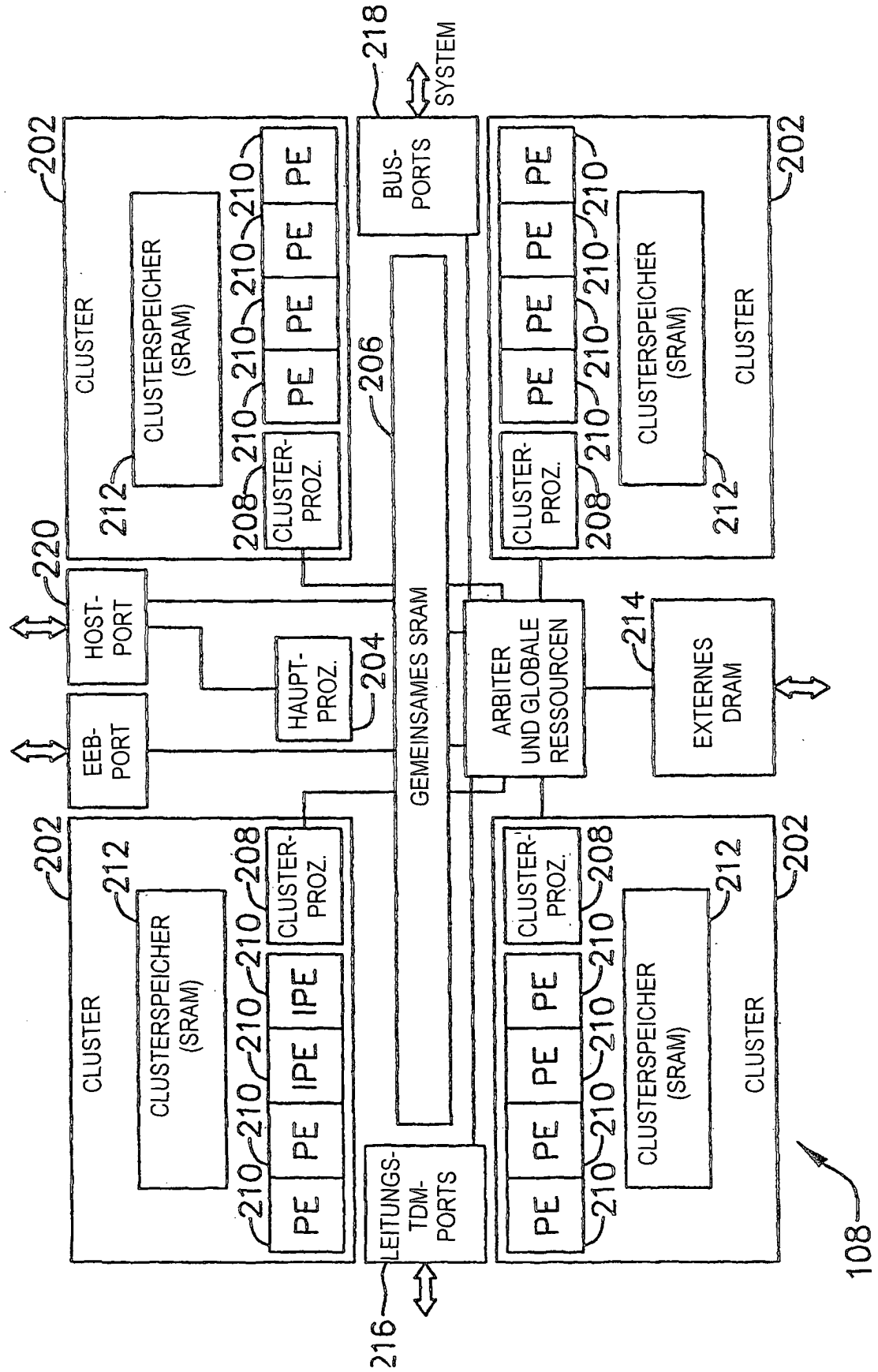

218
SYSTEM
202
CLUSTER
CLUSTERSPEICHER (SRAM)
212
PE
210
208
CLUSTER-PROZ.
BUS-PORTS
206
220
HOST-PORT
EEB-PORT
HAUPT-PROZ.
204
GEMEINSAMES SRAM
ARBITER UND GLOBALE RESSOURCEN
214
EXTERNES DRAM
IPE
LEITUNGS-TDM-PORTS
216
108

FIG. 3

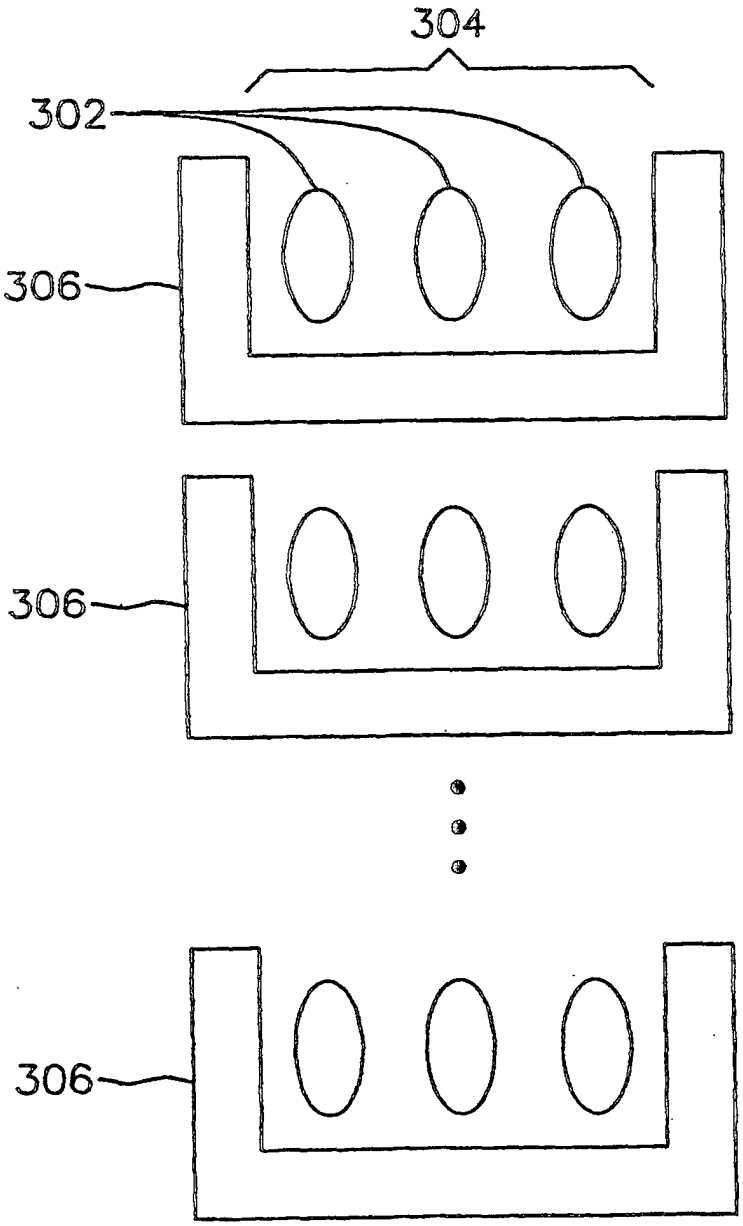


FIG. 4

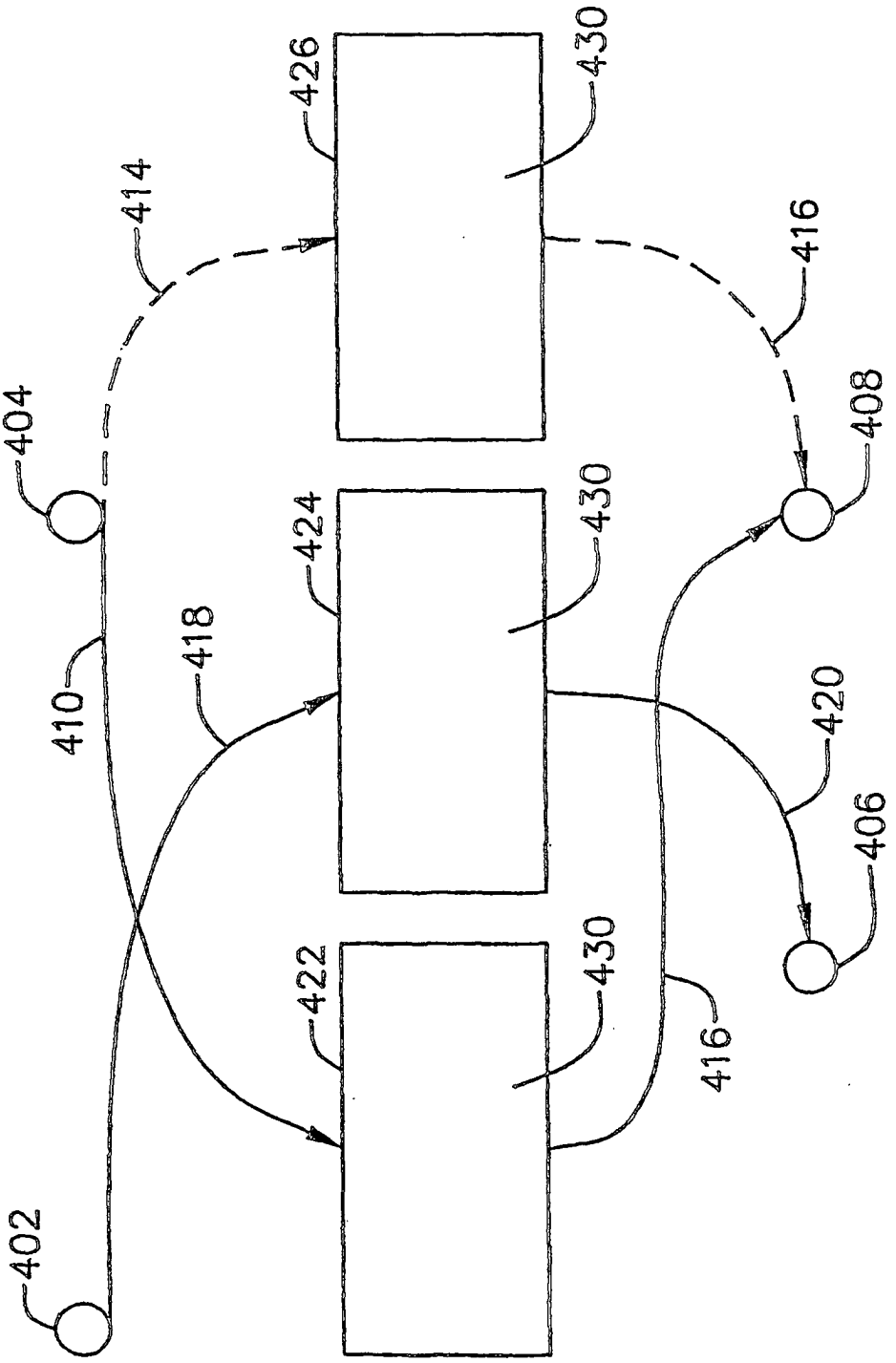


FIG.5

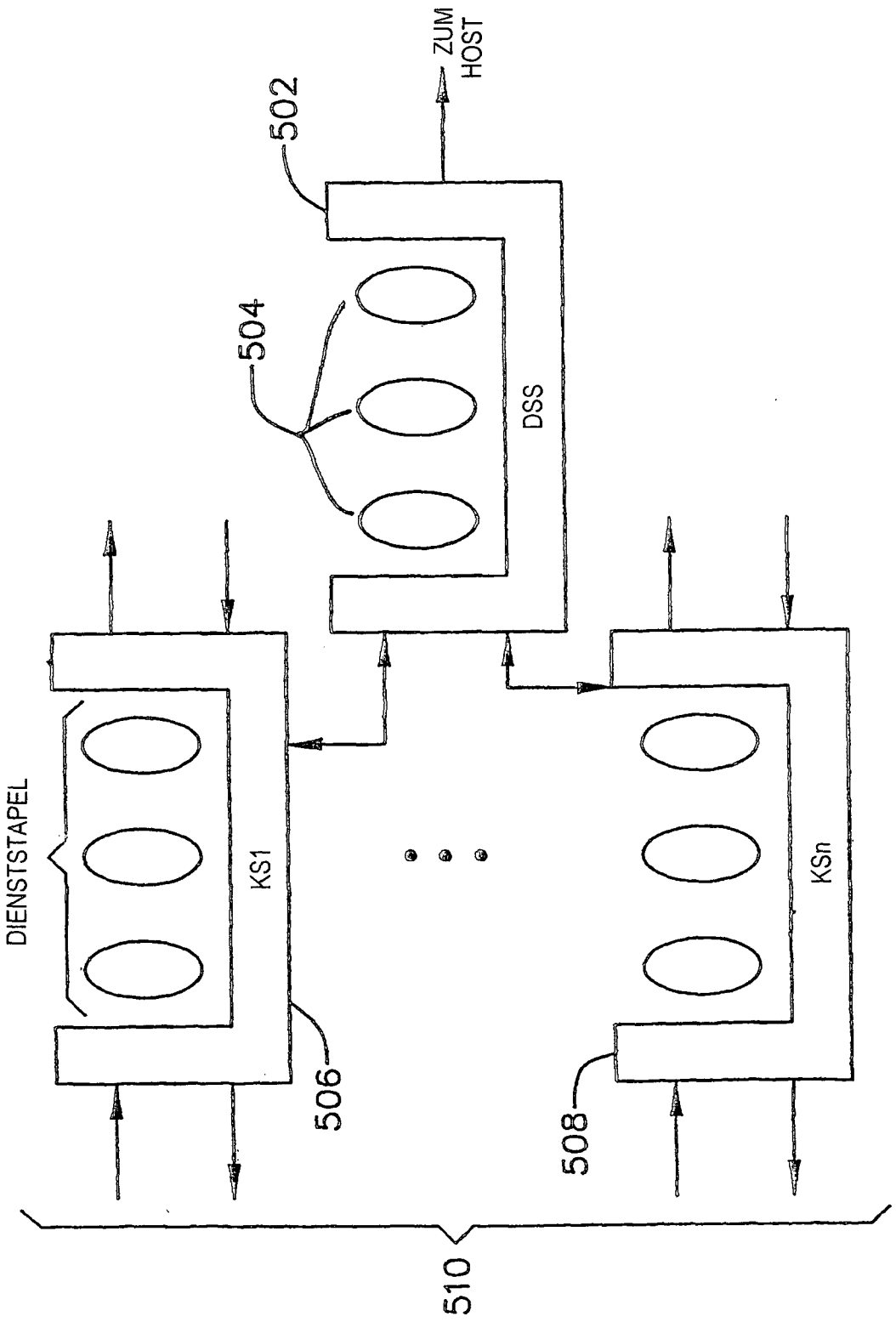
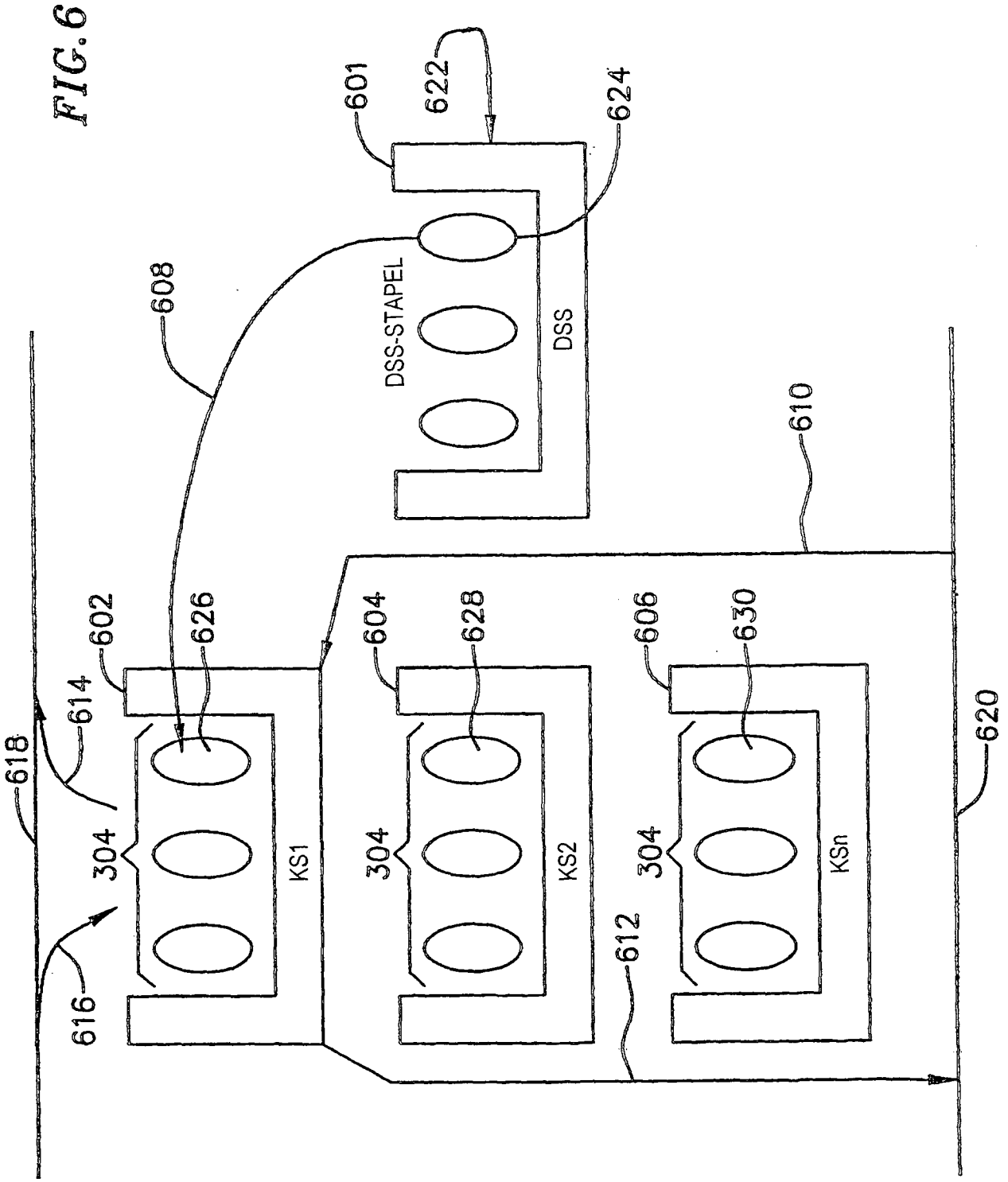


FIG. 6


601
622
624
608
DSS-STAPEL
DSS
610
602
626
604
628
606
630
614
618
304
KS1
304
KS2
304
KSn
620
616
612

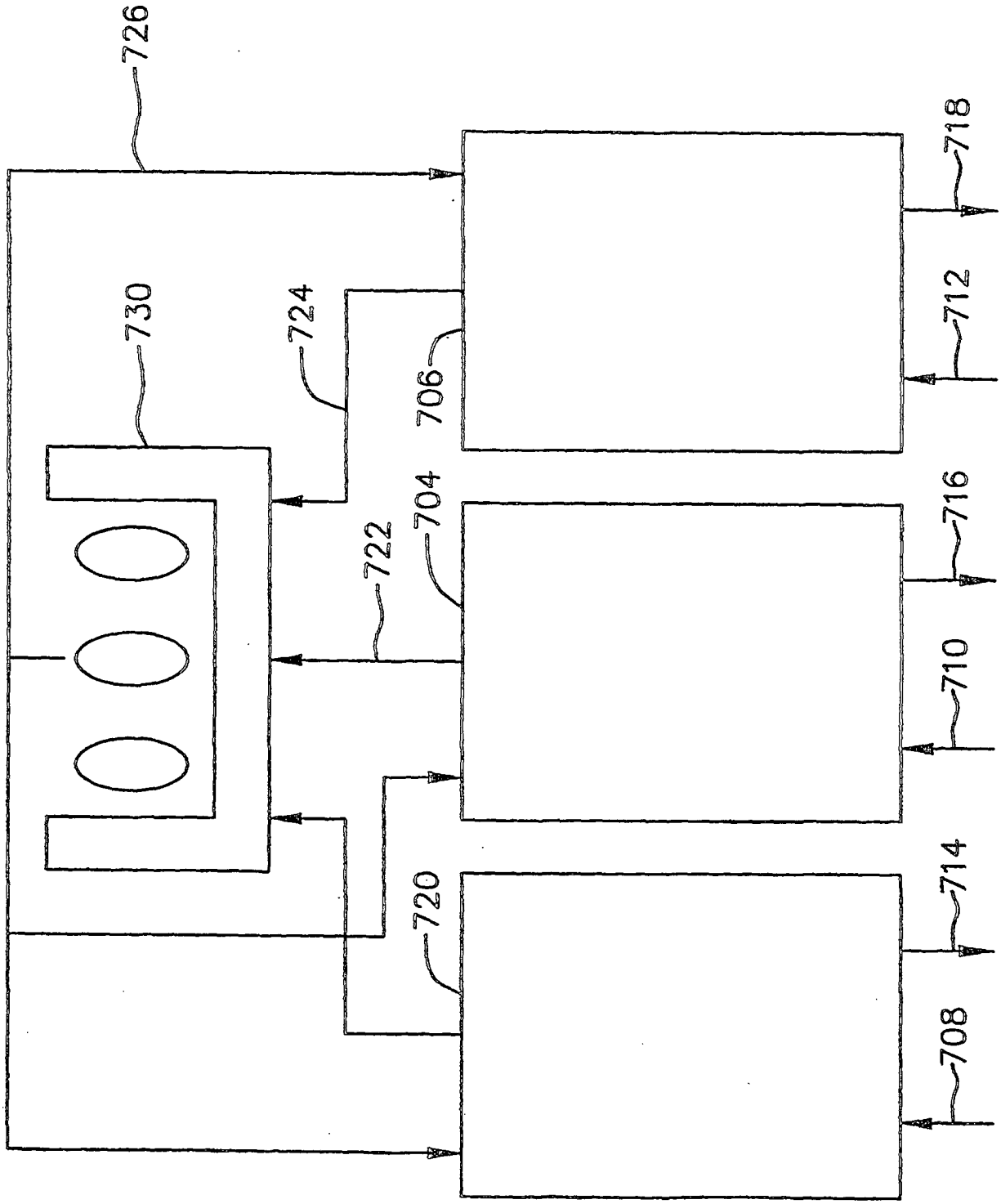
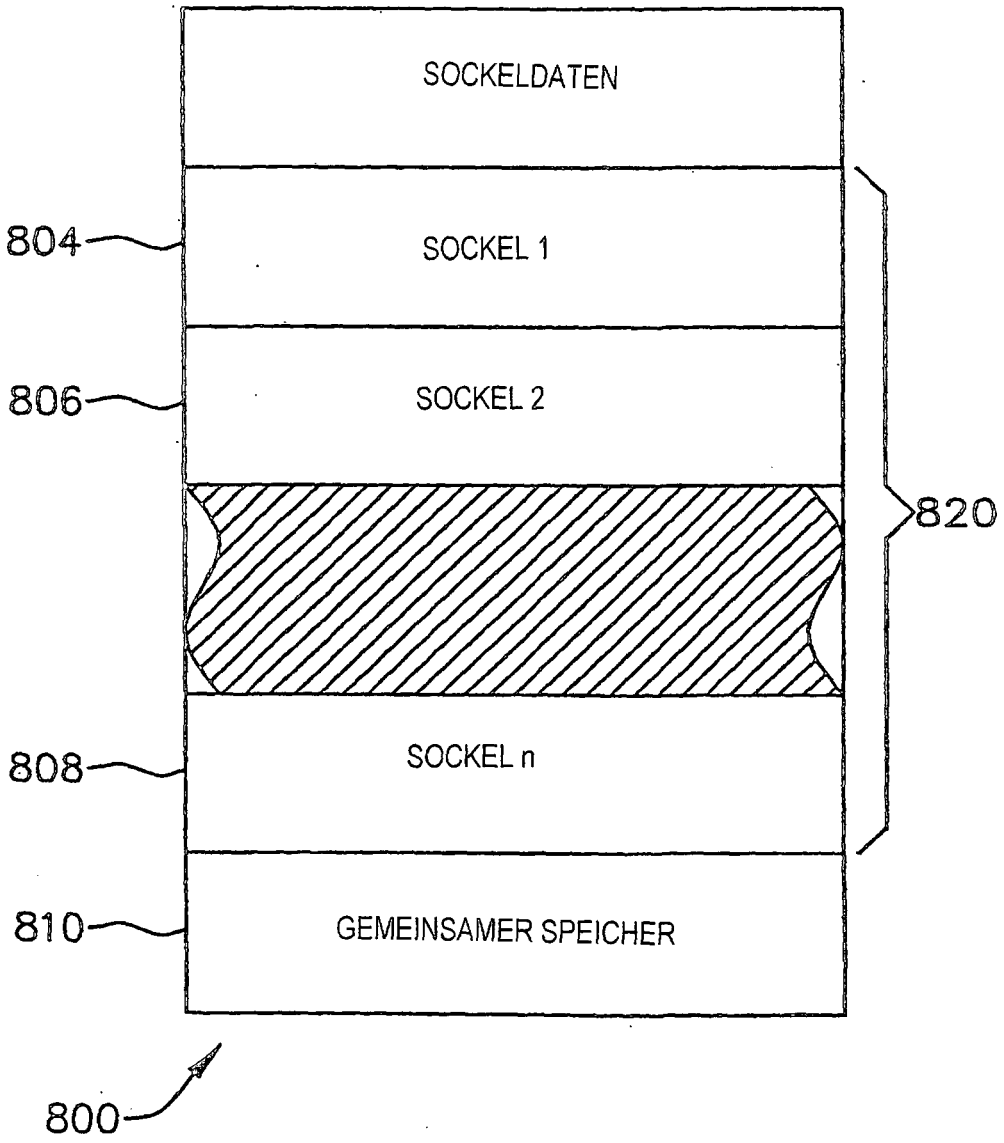


FIG. 7

FIG.8


SOCKELDATEN
804
SOCKEL 1
806
SOCKEL 2
820
808
SOCKEL n
810
GEMEINSAMER SPEICHER
800

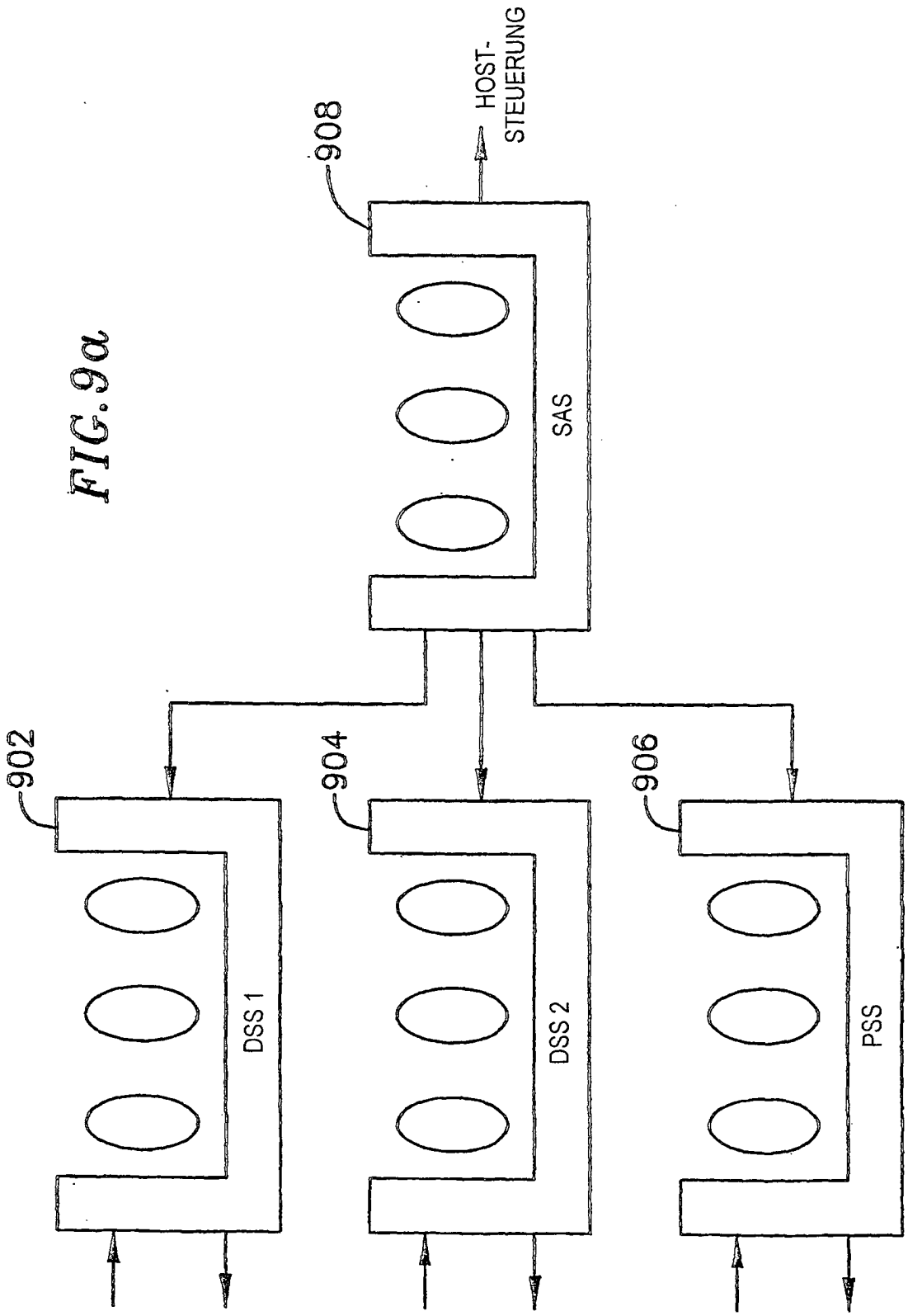


FIG. 9a

FIG. 9b

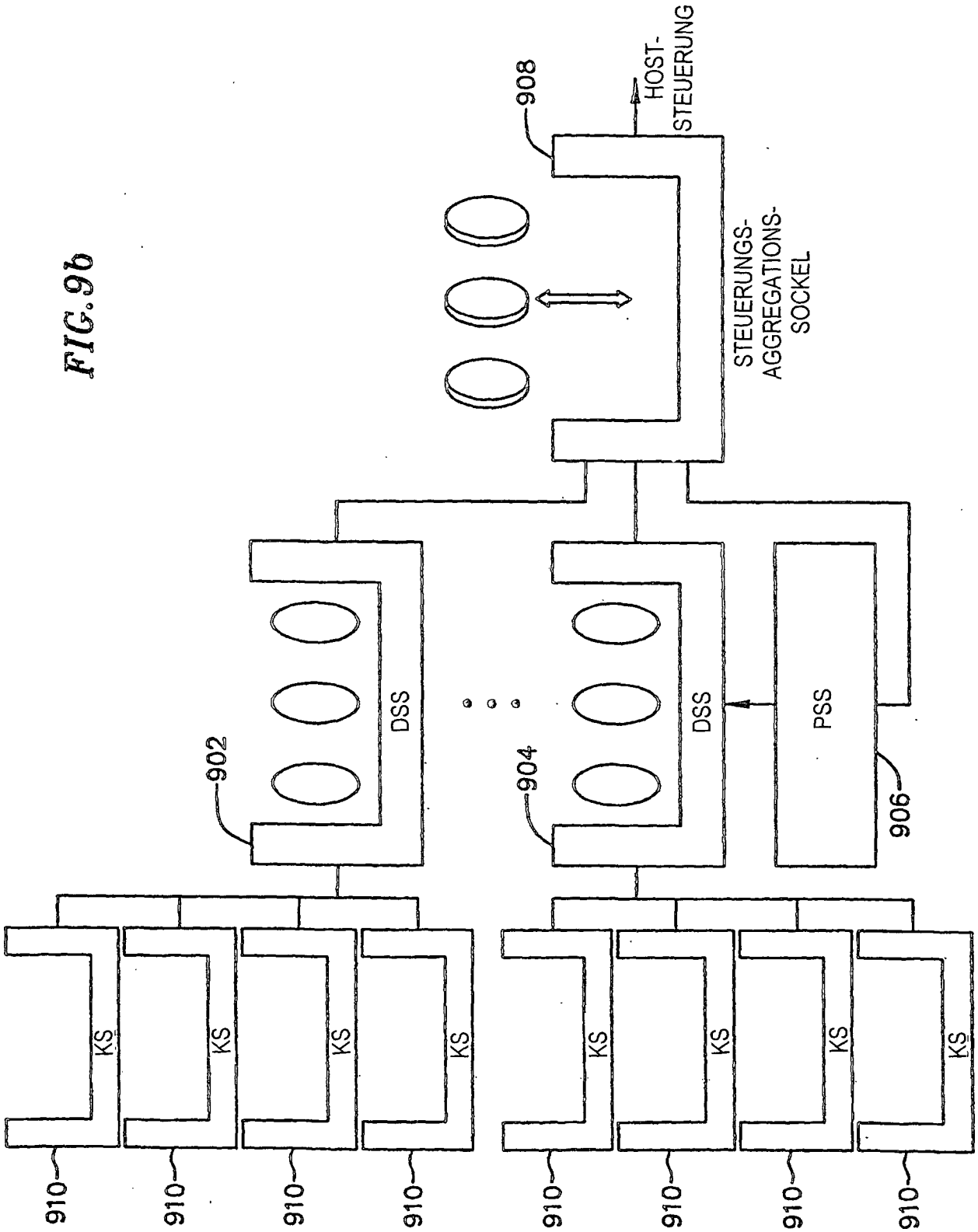

908
HOST-STEUERUNG
STEUERUNGS-AGGREGATIONS-SOCKEL
902
904
906
DSS
DSS
PSS
KS
910

FIG. 10

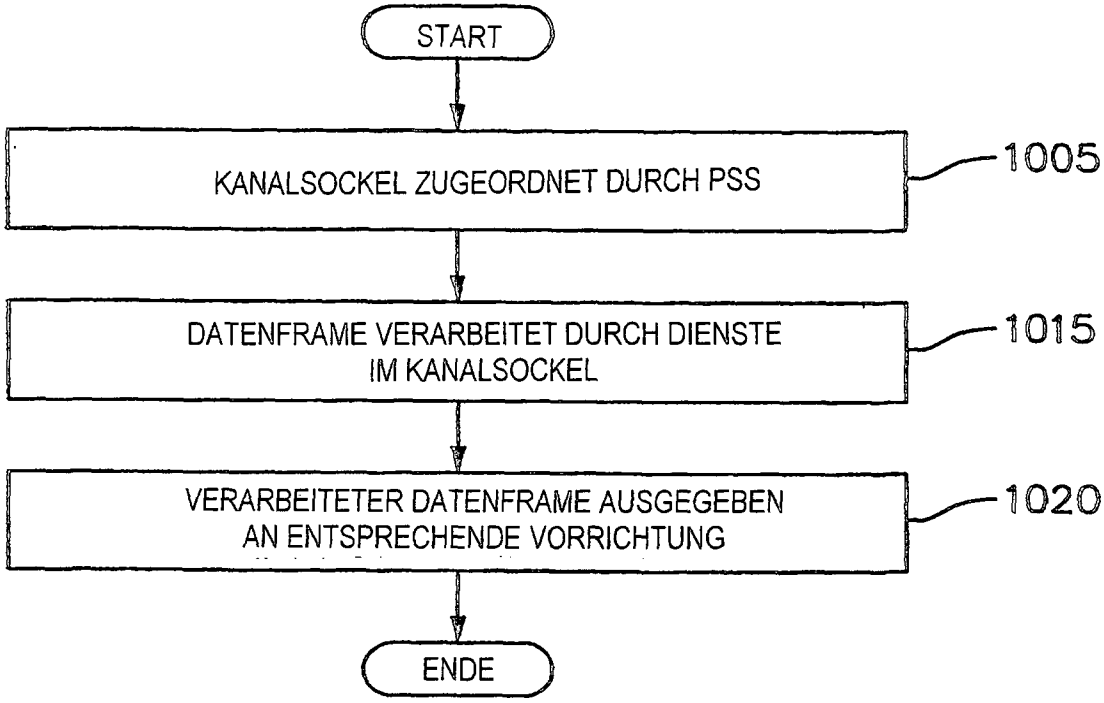

START
KANALSOCKEL ZUGEORDNET DURCH PSS
1005
DATENFRAME VERARBEITET DURCH DIENSTE IM KANALSOCKEL
1015
VERARBEITETER DATENFRAME AUSGEGEBEN AN ENTSPRECHENDE VORRICHTUNG
1020
ENDE

FIG. 11

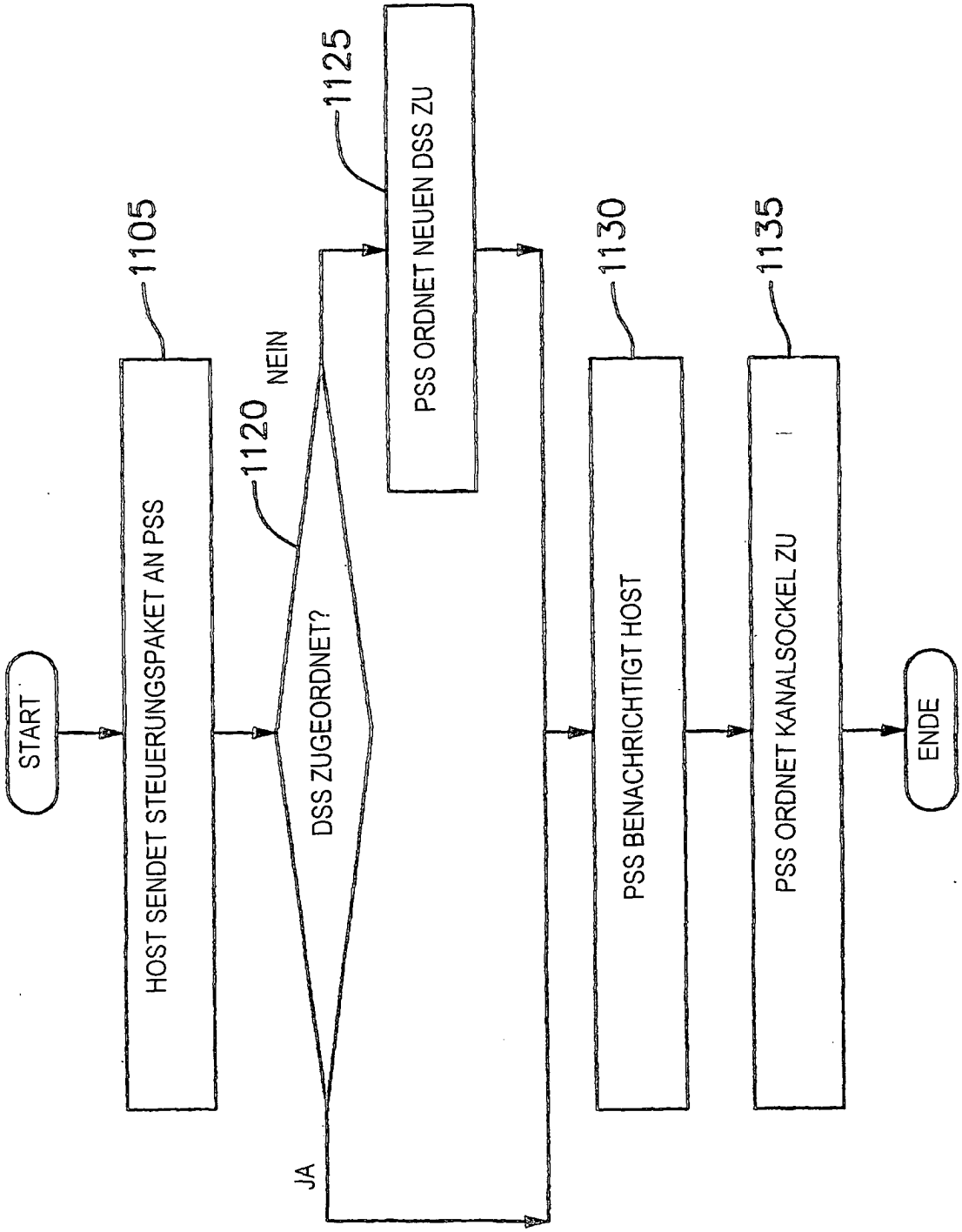

START
HOST SENDET STEUERUNGSPAKET AN PSS
1105
DSS ZUGEORDNET?
1120
NEIN
JA
PSS ORDNET NEUEN DSS ZU
1125
PSS BENACHRICHTIGT HOST
1130
PSS ORDNET KANALSOCKEL ZU
1135
ENDE

FIG. 12

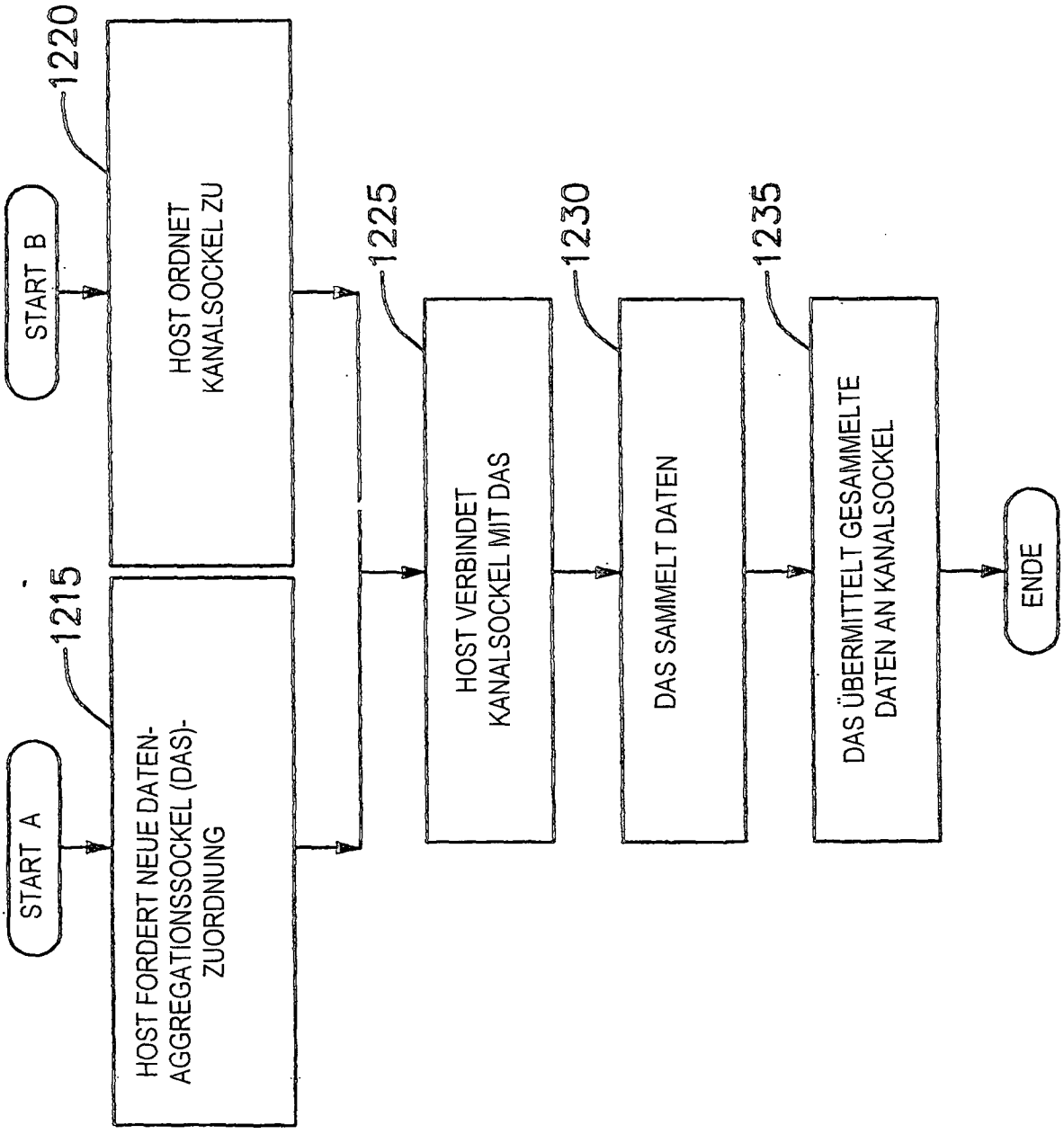
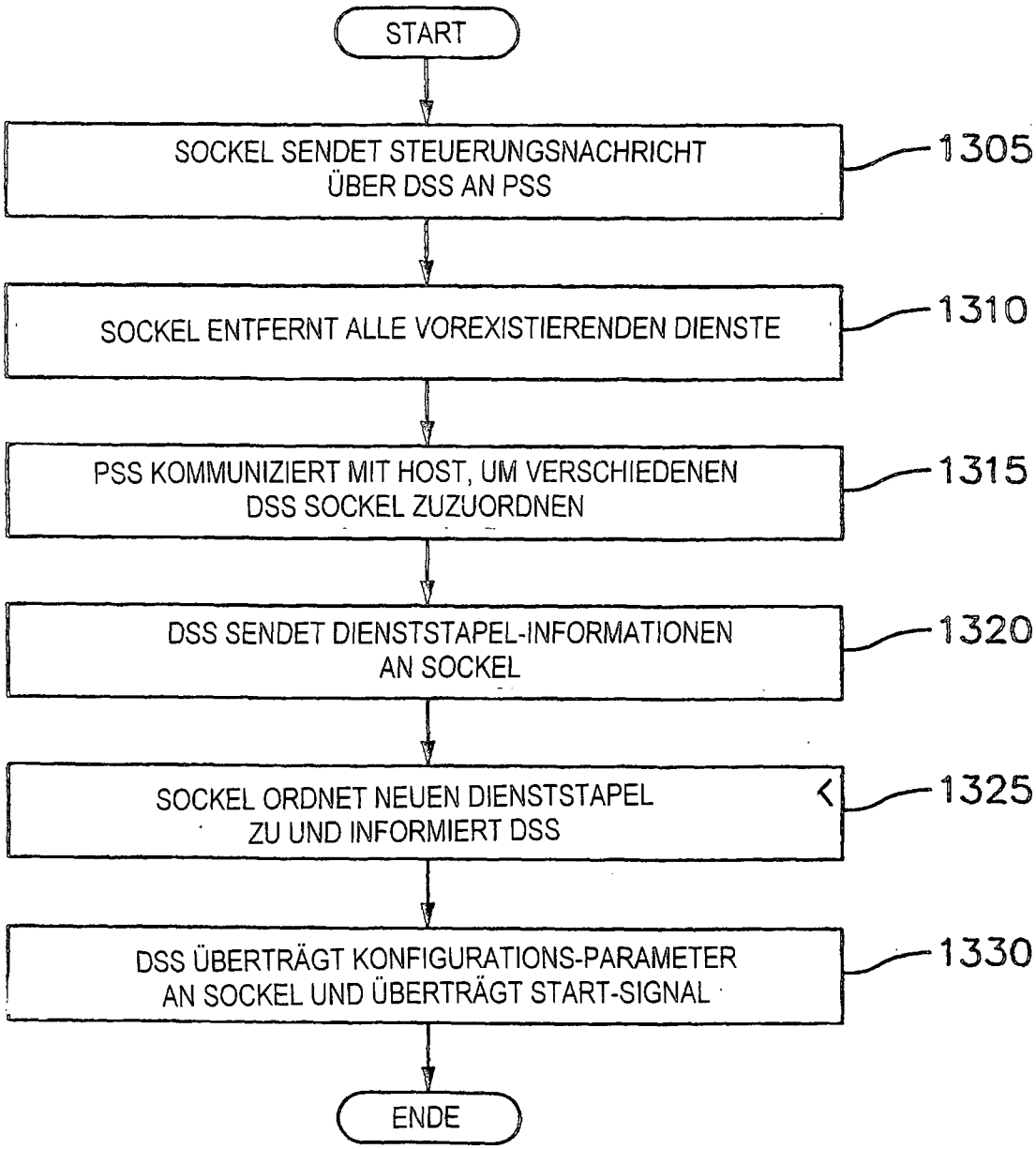

START A
START B
1215
HOST FORDERT NEUE DATEN-AGGREGATIONSSOCKEL (DAS)-ZUORDNUNG
1220
HOST ORDNET KANALSOCKEL ZU
1225
HOST VERBINDET KANALSOCKEL MIT DAS
1230
DAS SAMMELT DATEN
1235
DAS ÜBERMITTELT GESAMMELTE DATEN AN KANALSOCKEL
ENDE

FIG. 13


START
SOCKEL SENDET STEUERUNGSNACHRICHT ÜBER DSS AN PSS
1305
SOCKEL ENTFERNT ALLE VOREXISTIERENDEN DIENSTE
1310
PSS KOMMUNIZIERT MIT HOST, UM VERSCHIEDENEN DSS SOCKEL ZUZUORDNEN
1315
DSS SENDET DIENSTSTAPEL-INFORMATIONEN AN SOCKEL
1320
SOCKEL ORDNET NEUEN DIENSTSTAPEL ZU UND INFORMIERT DSS
1325
DSS ÜBERTRÄGT KONFIGURATIONS-PARAMETER AN SOCKEL UND ÜBERTRÄGT START-SIGNAL
1330
ENDE