(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 346 045**

(51) Int. Cl.:
G06F 12/14 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **02705345 .3**
(96) Fecha de presentación : **19.03.2002**
(97) Número de publicación de la solicitud: **1370948**
(97) Fecha de publicación de la solicitud: **17.12.2003**

(54) Título: **Sistema y método de control de acceso a memoria compartida para una arquitectura de ordenador para redes de banda ancha.**

(30) Prioridad: **22.03.2001 US 816020**

(45) Fecha de publicación de la mención BOPI:
08.10.2010

(45) Fecha de la publicación del folleto de la patente:
08.10.2010

(73) Titular/es: **Sony Computer Entertainment Inc.**
2-6-21, Minami-Aoyama
Minato-ku, Tokyo 107-0062, JP

(72) Inventor/es: **Suzuoki, Masakazu y**
Yamazaki, Takeshi

(74) Agente: **Elzaburu Márquez, Alberto**

DESCRIPCIÓN

Sistema y método de control de acceso a memoria compartida para una arquitectura de ordenador para redes de banda ancha.

Campo técnico

El presente invento se refiere a una arquitectura para procesadores de ordenador y redes de ordenadores y, en particular, a una arquitectura para procesadores de ordenador y redes de ordenadores en un entorno de banda ancha, y más particularmente a un sistema de tratamiento por ordenador con un sistema de control de acceso a memoria compartida de multiprocesador. El presente invento se refiere además a un modelo de programación para tal arquitectura y, en particular, a un método de tratamiento por ordenador.

Técnica antecedente

Los ordenadores y dispositivos informáticos de redes actuales de ordenadores, por ejemplo redes de área local (LAN) usados en redes de oficinas y redes globales tales como Internet, fueron diseñados principalmente para informática autónoma. La acción de compartir datos y programas de aplicación ("aplicaciones") sobre una red de ordenadores no era un objetivo del diseño principal de estos ordenadores y dispositivos informáticos. Estos ordenadores y dispositivos informáticos también fueron diseñados típicamente usando una amplia variedad de diferentes procesadores hechos por una variedad de diferentes fabricantes, por ejemplo, Motorola, Intel, Texas Instruments, Sony y otros. Cada uno de estos procesadores tiene su propio conjunto de instrucciones particulares y arquitectura de conjunto de instrucciones (ISA), es decir, su propio conjunto particular de instrucciones de estructura de lenguaje de ensamblaje para las principales unidades informáticas y unidades de memoria principales para que realicen estas instrucciones. Se requiere un programador para comprender, por ello, cada conjunto de instrucciones del procesador y la ISA para escribir aplicaciones para estos procesadores. Esta combinación heterogénea de ordenadores y dispositivos informáticos en las redes de ordenadores de hoy en día complica el tratamiento y la acción de compartir datos y aplicaciones. A menudo se requieren múltiples versiones de la misma aplicación, además, para acomodar este entorno heterogéneo.

Los tipos de ordenadores y dispositivos informáticos conectados a redes globales, particularmente a Internet, son extensivos. Además de los ordenadores personales (PC) y servidores, estos dispositivos informáticos incluyen teléfonos móviles, ordenadores móviles, asistentes digitales personales (PDA), codificadores, televisiones digitales y muchos otros. La acción de compartir datos y aplicaciones entre esta variedad de ordenadores y dispositivos informáticos presenta problemas sustanciales.

Se han empleado varias técnicas en un intento de resolver estos problemas. Estas técnicas incluyen, entre otras, interfaces sofisticadas y técnicas de programación complicadas. Estas soluciones requieren a menudo aumentos sustanciales en la potencia de tratamiento para llevarlas a la práctica. También a menudo dan como resultado un aumento sustancial del tiempo requerido para procesar aplicaciones y para transmitir datos sobre redes.

Los datos son transmitidos típicamente sobre Internet por separado desde las aplicaciones correspondientes. Esta aproximación evita la necesidad de enviar la aplicación con cada conjunto de datos transmitidos correspondientes a la aplicación. Aunque esta aproximación minimiza la cantidad de ancho de banda necesario, también causa a menudo frustración entre los usuarios. La aplicación correcta, o la aplicación más corriente, para los datos transmitidos puede no estar disponible en el ordenador del cliente. Esta aproximación también requiere la escritura de una multiplicidad de versiones de cada aplicación para la multiplicidad de ISA y conjuntos de instrucciones diferentes empleados por los procesadores de la red.

El modelo Java intenta resolver este problema. Este modelo emplea una pequeña aplicación ("miniprograma" o en inglés "applet") que cumple con un protocolo de seguridad estricto. Los miniprogramas son enviados desde un ordenador servidor sobre la red para ser ejecutados por un ordenador de cliente ("cliente"). Para evitar tener que enviar diferentes versiones del mismo miniprograma a clientes que emplean diferentes ISA, todos los miniprogramas de Java son ejecutados en una máquina virtual Java del cliente. Esta máquina virtual Java es un software que emula un ordenador que tiene una ISA de Java y un conjunto de instrucciones Java. Este software sin embargo, se ejecuta sobre la ISA del cliente y el conjunto de instrucciones del cliente. Una versión de la máquina virtual Java es proporcionada para cada ISA y conjunto de instrucciones diferentes de los clientes. Por ello, no se requiere una multiplicidad de diferentes versiones de cada miniprograma. Cada cliente descarga solo la máquina virtual Java correcta para su ISA y conjunto de instrucciones particulares para ejecutar todos los miniprogramas de Java.

Aunque proporciona una solución al problema de tener que escribir versiones diferentes de una aplicación para cada ISA y conjunto de instrucciones diferentes, el modelo de tratamiento Java requiere una capa adicional de software sobre el ordenador del cliente. Esta capa adicional de software degrada significativamente una velocidad de tratamiento del ordenador. Esta disminución de la velocidad es particularmente significativa para aplicaciones multimedia en tiempo real. Un miniprograma o "applet" de Java descargado también puede contener virus, fallos de tratamiento, etc. Estos virus y fallos pueden corromper una base de datos de un cliente y causar otros daños. Aunque un protocolo de seguridad empleado en el modelo Java intenta resolver este problema implantando un entorno restringido ("sandbox") de software, es decir, un espacio en la memoria del cliente más allá del cual el miniprograma de Java no puede escribir

datos, este modelo de seguridad activado mediante software es a menudo inseguro en su implantación y requiere incluso más tratamiento.

Las aplicaciones de red, multimedia, en tiempo real están resultando cada vez más importantes. Estas aplicaciones de red requieren velocidades de tratamiento extremadamente rápidas. Muchos miles de megabits de datos por segundo pueden ser necesarios en el futuro para tales aplicaciones. La arquitectura común de redes, y particularmente la de Internet, y el modelo de programación actualmente puesto en práctica, por ejemplo, en el modelo Java, hace que alcanzar tales velocidades de tratamiento sea extremadamente difícil.

Por ello, se requieren una nueva arquitectura de ordenador, una nueva arquitectura para redes de ordenadores y un nuevo modelo de programación. Esta arquitectura y modelo de programación nuevos deberían resolver los problemas de compartir datos y aplicaciones entre los distintos miembros de una red sin imponer cargas informáticas añadidas. Esta arquitectura de ordenador y modelo de programación nuevos deberían también resolver los problemas de seguridad inherentes al compartir aplicaciones y datos entre los miembros de una red.

El documento US-A-5.978.839 describe un sistema de tratamiento por ordenador en el que, cuando se asigna memoria para una CPU, la información de la CPU (por ejemplo el número de CPU, ID de CPU, y/o el tipo de CPU) deberían ser previamente clarificados o proporcionados al sistema. Si tal información no ha sido proporcionada previamente, el sistema no puede asignar memoria para la CPU ya que el sistema no conoce la información de la CPU.

Descripción del invento

El presente invento está definido en las reivindicaciones. En un aspecto, el presente invento proporciona una nueva arquitectura para ordenadores, dispositivos informáticos y redes de ordenadores. En otro aspecto, el presente invento proporciona un nuevo modelo de programación para estos ordenadores, dispositivos informáticos y redes de ordenadores.

De acuerdo con el presente invento, todos los miembros de una red de ordenadores, es decir, todos los ordenadores y dispositivos informáticos de la red, están construidos a partir de un módulo informático común. Este módulo informático común tiene una estructura consistente y preferiblemente emplea la misma ISA. Los miembros de la red pueden ser, por ejemplo, clientes, servidores, PC, ordenadores móviles, maquinas de juegos, PDA, codificadores, aparatos, televisiones digitales y otros dispositivos que usan procesadores de ordenador. La estructura modular consistente permite un tratamiento de aplicaciones y datos eficiente, de elevada velocidad por los miembros de la red y la rápida transmisión de aplicaciones y datos sobre la red. Esta estructura también simplifica la construcción de miembros de la red de distintos tamaños y potencia de tratamiento y la preparación de aplicaciones para su tratamiento por estos miembros. Además, de acuerdo con una realización del presente invento, se ha proporcionado una red de ordenadores que comprende una pluralidad de procesadores conectados a dicha red, comprendiendo cada uno de dichos procesadores una pluralidad de primeras unidades de tratamiento que tienen la misma arquitectura de conjunto de instrucciones y una segunda unidad de tratamiento para controlar dichas primeras unidades de tratamiento, siendo dichas primeras unidades de tratamiento operables para procesar celdas de software transmitidas sobre dicha red, comprendiendo cada una de dichas celdas de software un programa compatible con dicha arquitectura de conjunto de instrucciones, datos asociados con dicho programa y un identificador que identifica únicamente dicha celda de software entre la totalidad de dichas celdas de software transmitidas sobre dicha red. Preferiblemente, el identificador tiene un número de identificación que identifica únicamente dicha celda de software entre la totalidad de dichas celdas de software transmitidas sobre dicha red.

En otro aspecto, el presente invento proporciona un nuevo modelo de programación para transmitir datos y aplicaciones sobre una red y para tratar datos y aplicaciones entre los miembros de la red. Este modelo de programación emplea una celda de software transmitida sobre la red para su tratamiento por cualquiera de los miembros de la red. Cada celda de software tiene la misma estructura y puede contener tanto aplicaciones como datos. Como resultado del tratamiento de alta velocidad y de la velocidad de transmisión proporcionada por la arquitectura de ordenador modular, las celdas pueden ser procesadas rápidamente. El código para las aplicaciones está basado preferiblemente en el mismo conjunto de instrucción e ISA comunes. Cada celda de software contiene preferiblemente una identificación global (ID global) e información que describe la cantidad de recursos informáticos requeridos para el tratamiento de las celdas. Como todos los recursos informáticos tienen la misma estructura básica y emplean la misma ISA, el recurso particular que realiza este tratamiento puede estar situado en cualquier lugar de la red y asignado dinámicamente.

El módulo de tratamiento básico es un elemento de procesador (PE). Un PE comprende preferiblemente una unidad de tratamiento (PU), un controlador de acceso directo a la memoria (DMAC) y una pluralidad de unidades de tratamiento unidas (APU). En una realización preferida, un PE comprende ocho APU. La PU y las APU actúan entre ellas con una memoria dinámica compartida de acceso aleatorio (DRAM) que tiene preferiblemente una arquitectura de cruz. La PU programa y orquesta el tratamiento de datos y aplicaciones por las APU. Las APU realizan este tratamiento de una forma independiente y paralela. El DMAC controla los accesos por la PU y las APU a los datos y aplicaciones almacenados en la DRAM compartida.

De acuerdo con esta estructura modular, el número de PE empleado por un miembro de la red está basado en la potencia de tratamiento requerida por dicho miembro. Por ejemplo, un servidor puede emplear cuatro PE, una estación de trabajo puede emplear dos PE y una PDA puede emplear un PE. El número de APU de un PE asignado para procesar

una celda de software particular depende de la complejidad y magnitud de los programas y datos contenidos dentro de la celda.

En una realización preferida, una pluralidad de PE está asociada con una DRAM compartida. La DRAM esta preferiblemente segregada en una pluralidad de secciones, y cada una de estas secciones está segregada en una pluralidad de bancos de memoria. En una realización preferida particularmente, la DRAM comprende sesenta y cuatro bancos de memoria, y cada banco tiene un megabyte de capacidad de almacenamiento. Cada sección de la DRAM es controlada preferiblemente por un controlador de banco, y cada DMAC de un PE accede preferiblemente a cada controlador de banco. El DMAC de cada PE en esta realización, por ello, puede acceder a cualquier parte de la DRAM compartida.

En otro aspecto que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, puede proporcionarse un sistema y un método sincronizados para una lectura de datos de una APU desde la DRAM compartida, y la escritura de datos a la misma. Este sistema evita conflictos entre las múltiples APU y los múltiples PE que comparten la DRAM. De acuerdo con este sistema y método, un área de la DRAM es designada para almacenar una pluralidad de bits llenos-vacíos. Cada uno de estos bits llenos-vacíos corresponde a un área designada de la DRAM. El sistema sincronizado está integrado en el hardware de la DRAM y, por ello, evita la sobrecarga informática de un esquema de sincronización de datos puesto en práctica en software.

El presente invento también implanta entornos restringidos dentro de la DRAM para proporcionar seguridad contra la corrupción de datos para un programa que está siendo tratado por una APU a partir de los datos para un programa que está siendo tratado por otra APU. Cada entorno restringido define un área de la DRAM compartida más allá de la cual una APU particular, o conjunto de APU, no puede leer o escribir datos.

En otro aspecto que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, puede proporcionarse un sistema y un método para la emisión de órdenes de la PU a las APU para iniciar el tratamiento de aplicaciones y datos de las APU. Estas órdenes, denominadas llamadas de procedimiento remoto de la APU (ARPC), permiten que las PU orquesten y coordinen el tratamiento paralelo de aplicaciones y datos de las APU sin que las APU realicen la función de coprocesadores.

En otro aspecto que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, puede proporcionarse un sistema y un método para establecer una estructura de tubería dedicada para el tratamiento de la corriente de datos. De acuerdo con este sistema y método, un grupo coordinado de APU, y un grupo coordinado de entornos restringidos de memoria asociadas con estas APU, son establecidos por una PU para el tratamiento de estos datos. Las APU de tubería dedicada y los entornos restringidos de memoria permanecen dedicados a la tubería durante períodos en los que el tratamiento de datos no tiene lugar. En otras palabras, las APU dedicadas y sus entornos restringidos asociados son colocados en un estado reservado durante estos períodos.

En otro aspecto que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, puede proporcionarse un temporizador absoluto para el tratamiento de tareas. Este temporizador absoluto es independiente de la frecuencia de los relojes empleados por las APU para el tratamiento de aplicaciones y datos. Las aplicaciones son escritas basándose en el período de tiempo para tareas definido por el temporizador absoluto. Si la frecuencia de los relojes empleados por las APU aumenta debido, por ejemplo, a mejoras en las APU, el período de tiempo para una tarea dada como ha sido definido por el temporizador absoluto permanece el mismo. Este esquema permite la utilización de tiempos de tratamiento mejorados por versiones más nuevas de las APU sin incapacitar a estas APU más nuevas en aplicaciones de tratamiento más viejas escritas para los tiempos de tratamiento más lentos de las APU más viejas.

También puede proporcionarse, como un aspecto que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, un esquema alternativo para permitir que las APU más nuevas que tienen velocidades de tratamiento más rápidas traten aplicaciones más viejas escritas para las velocidades de tratamiento más lentas de las APU más viejas. En este esquema alternativo, las instrucciones o el microcódigo particulares empleados por las APU en el tratamiento de estas aplicaciones más viejas son analizados durante el tratamiento para problemas en la coordinación del tratamiento paralelo de las APU creado por las velocidades mejoradas. Las instrucciones "sin funcionamiento" ("NOOP") son insertadas en las instrucciones ejecutadas por alguna de estas APU para mantener la terminación secuencial del tratamiento por las APU esperado por el programa. Insertando estas NOOP en estas instrucciones, es mantenida la temporización correcta para la ejecución de todas las instrucciones de las APU.

En otro aspecto que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, puede proporcionarse un paquete de chip que contiene un circuito integrado en el que está integrada una guía de onda óptica.

Breve descripción de los dibujos

La fig. 1 ilustra la arquitectura total de una red de ordenadores que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 2 es un diagrama que ilustra la estructura de un elemento procesador (PE) de acuerdo con el presente invento.

La fig. 3 es un diagrama que ilustra la estructura de un motor de banda ancha (BE) que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 4 es un diagrama que ilustra la estructura de una unidad de tratamiento unida (APU) de acuerdo con el presente invento.

La fig. 5 es un diagrama que ilustra la estructura de un elemento de procesador, visualizador (VS) y una interfaz óptica que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 6 es un diagrama que ilustra una combinación de elementos de procesador de acuerdo con el presente invento.

La fig. 7 ilustra otra combinación de elementos de procesador que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 8 ilustra aún otra combinación de elementos de procesador que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 9 ilustra aún otra combinación de elementos de procesador que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 10 ilustra aún otra combinación de elementos de procesador que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 11A ilustra la integración de interfaces ópticas dentro de un paquete de chip que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 11B es un diagrama de una configuración de procesadores que usan las interfaces ópticas de la fig. 11A.

La fig. 11C es un diagrama de otra configuración de procesadores que usan las interfaces ópticas de la fig. 11A.

La fig. 12A ilustra la estructura de un sistema de memoria que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 12B ilustra la escritura de datos desde un primer motor de banda ancha a un segundo motor de banda ancha que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 13 es un diagrama de la estructura de una memoria compartida para un elemento de procesador de acuerdo con el presente invento.

La fig. 14A ilustra una estructura para un banco de la memoria mostrado en la fig. 13.

La fig. 14B ilustra otra estructura para un banco de la memoria mostrado en la fig. 13.

La fig. 15 ilustra una estructura para un controlador de acceso directo a la memoria de acuerdo con el presente invento.

La fig. 16 ilustra una estructura alternativa para un controlador de acceso directo a la memoria de acuerdo con el presente invento.

Las figs. 17A a 17O ilustran la operación de sincronización de datos que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 18 es un diagrama de memoria de tres estados que ilustra los distintos estados de una posición de memoria de acuerdo con el esquema de sincronización de datos que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 19 ilustra la estructura de una tabla de control de claves para un entorno restringido de hardware de acuerdo con el presente invento.

La fig. 20 ilustra un esquema para almacenar claves de acceso a la memoria para un entorno restringido de hardware de acuerdo con el presente invento.

La fig. 21 ilustra la estructura de una tabla de control de acceso a la memoria para un entorno restringido de hardware de acuerdo con el presente invento.

La fig. 22 es un diagrama de flujo de las operaciones para acceder a un entorno restringido de memoria que usa la tabla de control de claves de la fig. 19 y la tabla de control de acceso a la memoria de la fig. 21.

La fig. 23 ilustra la estructura de una celda de software que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 24 es un diagrama de flujo de las operaciones para emitir llamadas de procedimiento remoto a las APU que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 25 ilustra la estructura de una tubería dedicada para tratar la corriente de datos que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

Las figs. 26A-26B son diagramas de flujos de las operaciones realizadas por la tubería dedicada de la fig. 25 en el tratamiento de la corriente de datos que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 27 ilustra una estructura alternativa para una tubería dedicada para el tratamiento de la corriente de datos que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

La fig. 28 ilustra un esquema para un temporizador absoluto para coordinar el tratamiento paralelo de aplicaciones y datos por las APU que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado.

Mejor modo para poner en práctica el invento

La arquitectura total para un sistema informático 101 que no es parte del presente invento pero que sirve para un mejor entendimiento del tema reivindicado, es mostrada en la fig. 1.

Como se ha ilustrado en esta figura, el sistema 101 incluye una red 104 a la que están conectada una pluralidad de ordenadores y dispositivos informáticos. La red 104 puede ser una LAN, una red global, tal como Internet, o cualquier otra red informática.

Los ordenadores y dispositivos informáticos conectados a la red 104 (los "miembros" de la red) incluyen, por ejemplo, ordenadores de cliente 106, ordenadores servidores 108, asistentes digitales personales (PDA) 110, televisión digital (DTV) 112 y otros ordenadores y dispositivos informáticos conectados por cable o inalámbricos. Los procesadores empleados por los miembros de red 104 son construidos a partir del mismo módulo informático común. Estos procesadores preferiblemente también tienen todos la misma ISA y realizan el tratamiento de acuerdo con el mismo conjunto de instrucciones. El número de módulos incluido dentro de cualquier procesador particular depende de la potencia de tratamiento requerida por ese procesador.

Por ejemplo, como los servidores 108 del sistema 101 realizan más tratamiento de datos y aplicaciones que los clientes 106, los servidores 108 contienen más módulos informáticos que los clientes 106. Las PDA 110, por otro lado, realizan la menor cantidad de tratamiento. Las PDA 110, por ello, contienen el menor número de módulos informáticos. La DTV 112, realiza un nivel de tratamiento entre las de los clientes 106 y los servidores 108. La DTV 112, por ello, contiene un número de módulos informáticos entre el de los clientes 106 y los servidores 108. Como se ha descrito más adelante, cada módulo informático contiene un controlador de tratamiento y una pluralidad de unidades de tratamiento idénticas para realizar el tratamiento paralelo de los datos y aplicaciones transmitidos sobre la red 104.

Esta configuración homogénea para el sistema 101 facilita su adaptabilidad, velocidad de tratamiento y eficiencia de tratamiento. Debido a que cada miembro del sistema 101 realiza el tratamiento usando uno o más (o alguna fracción) del mismo módulo informático, el ordenador o dispositivo informático particular que realiza el tratamiento real de datos y aplicaciones no tiene importancia. El tratamiento de una aplicación y de datos particulares, además, puede ser compartido entre los miembros de red. Identificando únicamente las celdas que comprenden los datos y aplicaciones tratados por el sistema 101 a través del sistema, los resultados del tratamiento pueden ser transmitidos al ordenador o sistema informático solicitando el tratamiento independientemente de donde ha ocurrido este tratamiento. Debido a que los módulos que realizan este tratamiento tienen una estructura común y emplean una ISA común, las cargas informáticas de una capa añadida de software son evitadas para conseguir la compatibilidad entre los procesadores. Esta arquitectura y modelo de programación facilitan la velocidad de tratamiento necesaria para ejecutar, por ejemplo, aplicaciones multimedia en tiempo real.

Para aprovecharse más de las velocidades y eficiencias de tratamiento facilitadas por el sistema 101, los datos y aplicaciones tratados por este sistema son empaquetados en celdas de software 102 uniformemente formateadas identificadas únicamente. Cada celda de software 102 contiene, o puede contener tanto aplicaciones como datos. Cada celda de software también contiene una ID para identificar globalmente la celda a través de toda la red 104 y el sistema 101. Esta uniformidad de estructura para las celdas de software, y la identificación única de las celdas de software en toda la red, facilita el tratamiento de aplicaciones y datos sobre cualquier ordenador o dispositivo informático de la red. Por ejemplo, un cliente 106 puede formular una celda de software 102 pero, debido a la capacidades de tratamiento limitadas del cliente 106, transmitir esta celda de software a un servidor 108 para su tratamiento. Las celdas de software pueden migrar, por ello, a través de toda la red 104 para su tratamiento sobre la base de disponibilidad de recursos de tratamiento en la red.

La estructura homogénea de procesadores y celdas de software del sistema 101 también evita muchos problemas de las redes heterogéneas actuales. Por ejemplo, modelos de programación ineficientes que buscan permitir el tratamiento de aplicaciones sobre cualquier ISA usando cualquier conjunto de instrucciones, por ejemplo, máquinas virtuales tales como la máquina virtual Java, son evitados. El sistema 101, por ello, puede poner en práctica el tratamiento de banda ancha mucho más efectiva y eficientemente que las redes de hoy en día.

El módulo de tratamiento básico para todos los miembros de red 104 es el elemento de procesador (PE). La fig. 2 ilustra la estructura de un PE. Como se ha mostrado en esta figura, PE 201 comprende una unidad de tratamiento (PU) 203, un controlador de acceso directo de memoria (DMAC) 205 y una pluralidad de unidades de tratamiento unidas (APU), particularmente, APU 207, APU 209, APU 211, APU 213, APU 215, APU 217, APU 219 y APU 221. Un bus o línea de transmisión local 223 de PE transmite datos y aplicaciones entre los APU, DMAC 205 y PU 203. El bus local 223 de PE puede tener, por ejemplo, una arquitectura tradicional o ser implantado como una red conmutada por paquetes. La implantación como una red conmutada por paquetes, al tiempo que requiere más hardware, aumenta el ancho de banda disponible.

El PE 201 puede ser construido usando distintos métodos para poner en práctica la lógica digital. El PE 201 es construido preferiblemente, sin embargo, como un único circuito integrado que emplea un semiconductor de óxido metálico complementario (CMOS) sobre un substrato de silicio. Materiales alternativos para substratos incluyen arseniuro de galio, arseniuro de galio aluminio y otros compuestos denominados III-B que emplean una amplia variedad de dopantes. El PE 201 también podría ser puesto en práctica usando un material superconductor, por ejemplo, lógica rápida de quantum de flujo único (RSFQ).

El PE 201 está estrechamente asociado con una memoria dinámica de acceso aleatorio (DRAM) 225 a través de una conexión 227 de memoria de ancho de banda elevado. La DRAM 225 funciona como la memoria principal para el PE 201. Aunque una DRAM 225 es preferiblemente una memoria dinámica de acceso aleatorio, la DRAM 225 podría ser puesta en práctica usando otros medios, por ejemplo, como una memoria estática de acceso aleatorio (SRAM), una memoria magnética de acceso aleatorio (MRAM), una memoria óptica o una memoria holográfica. El DMAC 205 facilita la transferencia de datos entre la DRAM 225 y las APU y PU de PE 201. Como se ha descrito adicionalmente más adelante, el DMAC 205 designa para cada APU un área exclusiva en la DRAM 225 en la que sólo la APU puede escribir datos y desde la que sólo la APU puede leer datos. Esta área exclusiva es designada un entorno restringido.

La PU 203 puede ser, por ejemplo, un procesador estándar capaz de tratamiento autónomo de datos y aplicaciones. En funcionamiento, la PU 203 programa y orquesta el tratamiento de datos y aplicaciones por las APU. Las APU son preferiblemente procesadores de múltiples datos de una única instrucción (SIMD). Bajo el control de la PU 203, las APU realizan el tratamiento de estos datos y aplicaciones de una manera paralela e independiente. El DMAC 205 controla los accesos por la PU 203 y las APU a los datos y aplicaciones almacenados en la DRAM 225 compartida. Aunque el PE 201 incluye preferiblemente ocho APU, un mayor o menor número de APU puede ser empleado en un PE dependiendo de la potencia de tratamiento requerida. También, un número de PE, tal como el PE 201, puede estar unido o empaquetado junto para proporcionar la potencia de tratamiento mejorada.

Por ejemplo, como se ha mostrado en la fig. 3, cuatro PE pueden ser empaquetados o unidos juntos, por ejemplo, dentro de uno o más paquetes de chip, para formar un único procesador para un miembro de red 104. Esta configuración es designada un motor de banda ancha (BE). Como se ha mostrado en la fig. 3, el BE 301 contiene cuatro PE, particularmente, PE 303, PE 305, PE 307 y PE 309. Las comunicaciones entre estos PE están sobre el bus 311 del BE. La conexión 313 de memoria amplia de banda ancha proporciona comunicación entre la DRAM compartida 315 y estos PE. En lugar del bus 311 del BE, las comunicaciones entre los PE del BE 301 pueden ocurrir a través de la DRAM 315 y de esta conexión de memoria.

La interfaz de entrada/salida (I/O) 317 y el bus externo 319 proporcionan comunicaciones entre el motor de banda ancha 301 y los otros miembros de red 104. Cada PE del BE 301 realiza el tratamiento de datos y aplicaciones de una manera paralela e independiente análoga al tratamiento paralelo e independiente de aplicaciones y datos realizado por las APU de un PE.

La fig. 4 ilustra la estructura de una APU. La APU 402 incluye una memoria local 406, registros 410, cuatro unidades 412 de coma flotante y cuatro unidades 414 de número entero. De nuevo, sin embargo, dependiendo de la potencia de tratamiento requerida, puede emplearse un número mayor o menor de unidades 412 de coma flotante y unidades 414 de número entero. En una realización preferida, la memoria local 406 contiene 128 kilobytes de almacenamiento, y la capacidad de registros 410 es 128 x 128 bits. Las unidades 412 de coma flotante funcionan preferiblemente a una velocidad de 32 billones de operaciones de coma flotante por segundo (32 GFLOPS), y las unidades 414 de número entero funcionan preferiblemente a una velocidad de 32 billones de operaciones por segundo (32 GOPS).

La memoria local 406 no es una memoria caché. La memoria local 406 es construida preferiblemente como una SRAM. El soporte de coherencia caché para una APU es innecesario. Una APU puede requerir el soporte de coherencia caché para accesos de memoria directos iniciados por la PU. El soporte de coherencia caché no es requerido, sin embargo, para accesos de memoria directos iniciados por una APU o para accesos desde y hacia dispositivos externos.

La APU 402 incluye además el bus 404 para transmitir aplicaciones y datos hacia y desde la APU. En una realización preferida, este bus es de 1.024 bits de ancho. La APU 402 incluye además buses internos 408, 420 y 418. En una realización preferida, el bus 408 tiene una anchura de 256 bits y proporciona comunicaciones entre la memoria local 406 y los registros 410. Los buses 420 y 418 proporcionan comunicaciones entre, respectivamente, registros 410 y unidades 412 de coma flotante, y registros 410 y unidades 414 de número entero. En una realización preferida, el ancho de los buses 418 y 420 desde las unidades de cómo flotante o de número entero es de 384 bits, y el ancho de los buses 418 y 420 desde las unidades de número entero o de coma flotante hasta los registros 410 es de 128 bits. La anchura mayor de estos buses desde los registros 410 hasta las unidades de número entero o de coma flotante que desde estas unidades a registros 410 acomoda el mayor flujo de datos desde los registros 410 durante el tratamiento. Un máximo de tres palabras son necesarias para cada cálculo. El resultado de cada cálculo, sin embargo, normalmente es sólo una palabra.

Las figs. 5 a 10 ilustran además la estructura modular de los procesadores de los miembros de red 104. Por ejemplo, como se ha mostrado en la fig. 5, un procesador puede comprender un único PE 502. Como se ha descrito antes, este PE comprende típicamente una PU, un DMAC y ocho APU. Cada APU incluye un almacenamiento local (LS). Por otro lado, un procesador puede comprender la estructura del visualizador (VS) 505. Como se ha mostrado en la fig. 5, el VS 505 comprende la PU 512, el DMAC 514 y cuatro APU, particularmente, APU 516, APU 518, APU 520 y APU 522. El espacio entre el paquete de chip normalmente ocupado por las otras cuatro APU de un PE está ocupado en este caso por el motor de píxel 508, la imagen caché 510 y el controlador de tubo de rayos catódicos (CRTC) 504. Dependiendo de la velocidad de comunicaciones requerida para el PE 502 o el VS 505, el enlace óptico 506 también puede ser incluido en el paquete de chip.

Usando esta estructura modular, estandarizada, otras numerosas variaciones de procesadores pueden ser construidas fácil y eficientemente. Por ejemplo, el procesador mostrado en la fig. 6 comprende dos paquetes de chip, particularmente, un paquete de chip 602 que comprende un BE y un paquete de chip 604 que comprende cuatro VS. La entrada/salida (I/O) 606 proporciona una interfaz entre el BE del paquete de chip 602 y la red 104. El bus 608 proporciona comunicaciones entre el paquete de chip 602 y el paquete de chip 604. El procesador 610 de entrada salida (IOP) controla el flujo de datos dentro y fuera de la I/O 606. La I/O 606 puede ser fabricada como un circuito integrado específico de aplicación (ASIC). La salida desde los VS es una señal de video 612.

La fig. 7 ilustra un paquete de chip para un BE 702 con dos interfaces ópticas 704 y 706 para proporcionar comunicaciones de velocidad ultra elevada a los otros miembros de red 104 (u otros paquetes de chip localmente conectados). El BE 702 puede funcionar como, por ejemplo, un servidor en red 104.

El paquete de chip de la fig. 8 comprende dos PE 802 y 804 y dos VS 806 y 808. Una I/O 810 proporciona una interfaz entre el paquete de chip y la red 104. La salida del paquete de chip es una señal de video. Esta configuración puede funcionar como, por ejemplo, una estación de trabajo gráfica.

La fig. 9 ilustra aun otra configuración. Esta configuración contiene una mitad de la potencia de tratamiento de la configuración ilustrada en la fig. 8. En lugar de dos PE, hay previsto un PE 902, y en lugar de dos VS, hay previsto un VS 904. La I/O 906 tiene una mitad del ancho de banda de la I/O ilustrada en la fig. 8. Tal procesador también puede funcionar, sin embargo, como una estación de trabajo gráfica.

Se ha mostrado una configuración final en la fig. 10. Este procesador consiste de sólo un único VS 1002 y una I/O 1004. Esta configuración puede funcionar como, por ejemplo, una PDA.

La fig. 11A ilustra la integración de interfaces ópticas en un paquete de chip de un procesador de red 104. Estas interfaces ópticas convierten señales ópticas a señales eléctricas y señales eléctricas a señales ópticas y pueden ser construidos a partir de una variedad de materiales que incluyen, por ejemplo, arseniuro de galio, arseniuro de aluminio galio, germanio y otros elementos o compuestos. Como se ha mostrado en esta figura, las interfaces ópticas 1104 y 1106 son fabricadas sobre el paquete de chip de BE 1102. El bus 1108 del BE proporciona comunicación entre los PE del BE 1102, particularmente, PE 1110, PE 1112, PE 1114, PE 1116, y estas interfaces ópticas. La interfaz óptica 1104 incluye dos puertos, particularmente el puerto 1118 y el puerto 1120, y la interfaz óptica 1106 también incluye dos puertos, principalmente, el puerto 1122 y el puerto 1124. Los puertos 1118, 1120, 1122 y 1124 están conectados, respectivamente, a guías de onda óptica 1126, 1128, 1130 y 1132. Las señales ópticas son transmitidas hacia y desde el BE 1102 a través de estas guías de onda óptica a través de los puertos de interfaces ópticas 1104 y 1106.

Una pluralidad de BE pueden estar conectados juntos en distintas configuraciones usando tales guías de onda óptica y los cuatro puertos ópticos de cada BE. Por ejemplo, como se ha mostrado en la fig. 11B, dos o más BE, por ejemplo, BE 1152, BE 1154 y BE 1156, pueden estar conectados en serie a través de tales puertos ópticos. En este ejemplo, la interfaz óptica 1166 del BE 1152 está conectada a través de sus puertos ópticos a los puertos ópticos de la interfaz óptica 1160 del BE 1154. De modo similar, los puertos ópticos de la interfaz óptica 1162 en el BE 1154 están conectados a los puertos ópticos de la interfaz óptica 1164 del BE 1156.

Se ha ilustrado una configuración de matriz en la fig. 11C. En esta configuración, la interfaz óptica de cada BE está conectada a otros dos BE. Como se ha mostrado en esta figura, uno de los puertos ópticos de interfaz óptica 1188 de BE 1172 está conectado a un puerto óptico de la interfaz óptica 1182 de BE 1176. El otro puerto óptico de la interfaz óptica 1188 está conectado a un puerto óptico de la interfaz óptica 1184 de BE 1178. De manera similar, un puerto

óptico de la interfaz óptica 1190 de BE 1174 está conectado a otro puerto óptico de la interfaz óptica 1184 de BE 1178. El otro puerto óptico de la interfaz óptica 1190 está conectado a un puerto óptico de la interfaz óptica 1186 de BE 1180. Esta configuración de matriz puede ser extendida de forma similar a otros BE.

Usando o bien una configuración en serie o bien una configuración de matriz, puede construirse un procesador para red 104 de cualquier tamaño y potencia deseados. Desde luego, pueden añadirse puertos adicionales a las interfaces ópticas de los BE, o a procesadores que tienen un mayor o menor número de PE que un BE, para formar otras configuraciones.

La fig. 12A ilustra el sistema de control y la estructura para la DRAM de un BE. Un sistema de control y estructura similares son empleados en procesadores que tienen otros tamaños y que contienen más o menos PE. Como se ha mostrado en esta figura, un interruptor de travesaño conecta cada DMAC 1210 de los cuatro PE que comprende el BE 1201 a ocho controles de banco 1206. Cada control de banco 1206 controla ocho bancos 1208 (sólo se han mostrado cuatro en la figura) de DRAM 1204. La DRAM 1204, por ello, comprende un total de sesenta y cuatro bancos. En una realización preferida, la DRAM 1204 tiene una capacidad de 64 megabytes, y cada banco tiene una capacidad de 1 megabyte. La menor unidad accesible dentro de cada banco, en esta realización preferida, es un bloque de 1024 bits.

El BE 1201 también incluye la unidad de conmutador 1212. La unidad de conmutador 1212 permite a otras APU en los BE estar estrechamente acopladas al BE 1201 para acceder a la DRAM 1204. Un segundo BE, por ello, puede estar estrechamente acoplado a un primer BE, y cada APU de cada BE puede acceder dos veces el número de posiciones de memoria normalmente accesibles a una APU. La lectura o escritura directa de datos desde o a la DRAM de un primer BE desde o a la DRAM de un segundo BE puede ocurrir a través de una unidad de conmutador tal como la unidad de conmutador 1212.

Por ejemplo, como se ha mostrado en la fig. 12B, para realizar tal escritura, la APU de un primer BE, por ejemplo, la APU 1220 del BE 1222, emite una orden de escritura a una posición de memoria de una DRAM de un segundo BE, por ejemplo, la DRAM 1228 del BE 1226 (en vez de, como en el caso normal, a la DRAM 1224 del BE 1222). El DMAC 1230 del BE 1222 envía la orden de escritura a través de un conmutador de cruz 1221 al control de banco 1234, y el control de banco 1234 transmite la orden a un puerto externo 1232 conectado al control de banco 1234. El DMAC 1238 del BE 1226 recibe la orden de escritura y transfiere esta orden a la unidad de conmutador 1240 del BE 1226. La unidad de conmutador 1240 identifica la dirección de la DRAM contenida en la orden de escritura y envía los datos para almacenarlos en esta dirección a través del control de banco 1242 del BE 1226 al banco 1244 de la DRAM 1228. La unidad de conmutador 1240, por ello, permite tanto a la DRAM 1224 como a la DRAM 1228 funcionar como un único espacio de memoria para las APU del BE 1222.

La fig. 13 muestra la configuración de sesenta y cuatro bancos de una DRAM. Estos bancos están dispuestos en ocho filas, particularmente, las filas 1302, 1304, 1306, 1308, 1310, 1312, 1314 y 1316 y ocho columnas, particularmente, las columnas 1320, 1322, 1324, 1326, 1328, 1330, 1332 y 1334. Cada fila es controlada por un controlador de banco. Cada controlador de banco, por ello, controla ocho megabytes de memoria.

Las figs. 14A y 14B ilustran diferentes configuraciones para almacenar y acceder a la menor unidad de memoria accesible de una DRAM, por ejemplo, un bloque de 1024 bits. En la fig. 14A, el DMAC 1402 almacena en un único banco 1404 ocho bloques 1406 de 1024 bits. En la fig. 14B, por otro lado, mientras el DMAC 1412 lee y escribe bloques de datos que contienen 1024 bits, estos bloques son entrelazados entre dos bancos, particularmente, el banco 1414 y el banco 1416. Cada uno de estos bancos, por ello, contiene dieciséis bloques de datos, y cada bloque de datos contiene 512 bits. Este entrelazado puede facilitar un acceso más rápido de la DRAM y es útil en el tratamiento de algunas aplicaciones.

La fig. 15 ilustra la arquitectura para un DMAC 1506 dentro de un PE. Como se ha ilustrado en esta figura, el hardware estructural que comprende el DMAC 1506 es distribuido totalmente en el PE de tal modo que cada APU 1502 tiene un acceso directo a un nudo estructural 1504 del DMAC 1506. Cada nudo ejecuta la lógica apropiada para accesos de memoria por la APU a la que el nodo tiene acceso directo.

La fig. 16 muestra una realización alternativa del DMAC, particularmente, una arquitectura no distribuida. En este caso, el hardware estructural del DMAC 1606 está centralizado. La APU 1602 y la PU 1604 comunican con el DMAC 1606 a través del bus local 1607 del PE. El DMAC 1606 está conectado a través de un conmutador de cruz a un bus 1608. El bus 1608 está conectado a la DRAM 1610.

Como se ha descrito antes, todas las múltiples APU de un PE pueden acceder independientemente a datos en la DRAM compartida. Como resultado, una primera APU podría estar operando sobre datos particulares en su almacenamiento local en un instante el cual una segunda APU solicita estos datos. Si los datos han sido proporcionados a la segunda APU en ese instante desde la DRAM compartida, los datos podrían ser inválidos debido al tratamiento en curso de la primera APU que podría cambiar el valor de los datos. Si el segundo procesador ha recibido los datos desde la DRAM compartida en ese instante, por ello, el segundo procesador podría generar un resultado erróneo. Por ejemplo, los datos podrían ser un valor específico para una variable global. Si el primer procesador ha cambiado ese valor durante su tratamiento, el segundo procesador recibiría un valor no actualizado. Un esquema es necesario, por ello, para sincronizar la lectura y escritura de datos de las APU desde posiciones de memoria dentro de la DRAM compartida y a las mismas. Este esquema debe impedir la lectura de datos desde una posición de memoria después

de que otra APU al mismo tiempo está operando en su almacenamiento local y, por ello, que no son actuales, y la escritura de datos a una posición de memoria que almacena datos actuales.

Para resolver estos problemas, para cada posición de memoria accesible de la DRAM, un segmento adicional de memoria está asignado en la DRAM para almacenar la información de estado relativa a los datos almacenados en la posición de memoria. Esta información de estado incluye un bit de lleno/vacío (F/E), la identificación de una APU (ID de la APU) que solicita datos desde la posición de memoria y la dirección del almacenamiento local de las APU (dirección del LS) al que los datos solicitados deberían ser leídos. Una posición de memoria accesible de la DRAM puede ser de cualquier tamaño. En una realización preferida, el tamaño es de 1024 bits.

El ajuste del bit de F/E a 1 indica que los datos almacenados en la posición de memoria asociada son actuales. El ajuste del bit de F/E a 0, por otro lado, indica que los datos almacenados en la posición de memoria asociada no son actuales. Si una APU solicita los datos cuando este bit está ajustado a 0, se le impide a la APU leer inmediatamente los datos. En este caso, una ID de la APU que identifica la APU que solicita los datos, y una dirección del LS que identifica la posición de memoria dentro del almacenamiento local de esta APU a la que los datos han de ser leídos cuando los datos resulten actuales, son introducidos en el segmento de memoria adicional.

Un segmento de memoria adicional también está asignado para cada posición de memoria dentro del almacenamiento local de las APU. Este segmento de memoria adicional almacena un bit, designado el "bit ocupado". El bit ocupado es usado para reservar la posición de memoria del LS asociada para el almacenamiento de datos específicos que han de ser recuperados desde la DRAM. Si el bit ocupado es ajustado a 1 para una posición de memoria particular en el almacenamiento local, la APU puede usar esta posición de memoria solo para la escritura de estos datos específicos. Por otro lado, si el bit ocupado es ajustado a 0 para una posición de memoria particular en el almacenamiento local, la APU puede usar esta memoria para la escritura de cualquier dato.

Ejemplos de la manera en la que el bit de F/E, la ID de la APU, la dirección del LS y el bit ocupado son usados para sincronizar la lectura y escritura de datos desde y a la DRAM compartida de un PE están ilustrados en las figs. 17A-17O.

Como se ha mostrado en la fig. 17A, uno o más PE, por ejemplo el PE 1720 interactúa con la DRAM 1702. El PE 1720 incluye la APU 1722 y la APU 1740. La APU 1722 incluye la lógica de control 1724, y la APU 1740 incluye la lógica de control 1742. La APU 1722 incluye también el almacenamiento local 1726. Este almacenamiento local incluye una pluralidad de posiciones de memoria accesibles 1728. La APU 1740 incluye el almacenamiento local 1744, y este almacenamiento local incluye también una pluralidad de posiciones de memoria accesibles 1746. Todas estas posiciones de memoria accesibles son preferiblemente de 1024 bits de tamaño.

Un segmento adicional de memoria está asociado con cada posición de memoria accesible de LS. Por ejemplo, los segmentos de memoria 1729 y 1734 están asociados respectivamente con posiciones de memoria local 1731 y 1732, y el segmento de memoria 1752 está asociado con la posición de memorial local 1750. Un "bit de ocupación" como se ha descrito antes, está almacenado en cada uno de estos segmentos de memoria adicionales. La posición de memoria local 1732 está mostrada con varias X para indicar que esta posición contiene datos.

La DRAM 1702 contiene una pluralidad de posiciones de memoria accesibles 1704, incluyendo posiciones de memoria 1706 y 1708. Estas posiciones de memoria son preferiblemente también de 1024 bits de tamaño. Un segmento adicional de memoria está también asociado con cada una de estas posiciones de memoria. Por ejemplo, el segmento de memoria adicional 1760 está asociado con la posición de memoria 1706, y el segmento de memoria adicional 1762 está asociado con la posición de memoria 1708. La información de estado relacionada con los datos almacenados en cada posición de memoria es almacenada en el segmento de memoria asociado con la posición de memoria. Esta información de estado incluye, como se ha descrito antes, el bit de F/E, la ID de la APU y la dirección de LS. Por ejemplo, para la posición de memoria 1708, esa información de estado incluye el bit de F/E 1712, la ID de la APU 1714 y la dirección de LS 216.

Usando la información de estado y el bit de ocupación, la lectura y escritura sincronizadas de datos desde la DRAM compartida y a la misma entre las APU de un PE, o un grupo de PE, puede ser conseguida.

La fig. 17B ilustra la iniciación de la escritura sincronizada de datos desde la posición de memoria de LS 1732 de APU 1722 a la posición de memoria 1708 de la DRAM 1702. El control 1724 de la APU 1722 inicia la escritura sincronizada de estos datos. Como la posición de memoria 1708 está vacía, el bit de F/E 1712 es ajustado a 0. Como resultado, los datos en la posición de LS 1732 pueden ser escritos en la posición de memoria 1708. Si este bit fuera ajustado a 0 para indicar que la posición de memoria 1708 está llena y contiene datos actuales, válidos, por otro lado, el control 1722 recibiría un mensaje de error y se prohibiría escribir datos en esta posición de memoria.

El resultado de la escritura sincronizada satisfactoria de los datos en la posición de memoria 1708 está mostrado en la fig. 17C. Los datos escritos son almacenados en la posición de memoria 1708 y el bit de F/E es ajustado a 1. Este ajuste indica que la posición de memoria 1708 está llena y que los datos en esta posición de memoria son actuales y válidos.

La fig. 17D ilustra la iniciación de la lectura sincronizada de datos desde la posición de memoria 1708 de la DRAM 1702 a la posición de memoria de LS 1750 del almacenamiento local 1744. Para iniciar esta lectura, el bit de ocupación en el segmento de memoria 1752 de la posición de memoria de LS 1750 es ajustado a 1 para reservar esta posición de memoria para estos datos. El ajuste de este bit de ocupación a 1 impide que la APU 1740 almacene otros datos en esta posición de memoria.

Como se ha mostrado en la fig. 17E, la lógica de control 1742 siguiente emite una orden de lectura sincronizada para la posición de memoria 1708 de la DRAM 1702. Como el bit de F/E 1712 asociado con esta posición de memoria es ajustado a 1, los datos almacenados en la posición de memoria 1708 son considerados actuales y válidos. Como resultado, en preparación para transferir los datos desde la posición de memoria 1708 a la posición de memoria de LS 1750, el bit de F/E 1712 es ajustado a 0. Este ajuste está mostrado en la fig. 17F. El ajuste de este bit a 0 indica, que después de la lectura de estos datos, los datos en la posición de memoria 1708 serán inválidos.

Como se ha mostrado en la fig. 17G, los datos dentro de la posición de memoria 1708 siguiente son leídos desde la posición de memoria 1708 a la posición de memoria de LS 1750. La fig. 17H muestra el estado final. Una copia de los datos en la posición de memoria 1708 es almacenada en la posición de memoria de LS 1750. El bit de F/E 1712 es ajustado a 0 para indicar que los datos en la posición de memoria 1708 son inválidos. Esta invalidez es el resultado de alteraciones de estos datos que han de ser hechas por la APU 1740. El bit de ocupación en el segmento de memoria 1752 es también ajustado a 0. Este ajuste indica que la posición de memoria de LS 1750 está ahora disponible a la APU 1740 para cualquier propósito, es decir, esta posición de la memoria de LS ya no está en un estado reservado esperando la recepción de datos específicos. La posición de la memoria de LS 1750, por ello, puede ahora ser accedida por la APU 1740 para cualquier propósito.

Las figs. 17I-17O ilustran la lectura sincronizada de datos desde una posición de memoria de la DRAM 1702, por ejemplo posición de memoria 1708, a una posición de memoria de LS de un almacenamiento local de la APU, por ejemplo, posición de memoria de LS 1752 de almacenamiento local 1744, cuando el bit F/E para la posición de memoria de DRAM 1702 es ajustado a 0 para indicar que los datos en esta posición de memorias no son actuales o válidos. Como se ha mostrado en la fig. 17I, para iniciar esta transferencia, el día de ocupación en el segmento de memoria 1752 de la posición de memoria de LS 1750 es ajustado a 1 para reservar esta posición de memoria de LS para esta transferencia de datos. Como se ha mostrado en la fig. 17J, la lógica de control 1742 siguiente emite una orden de lectura sincronizada para posición de memoria 1708 de la DRAM 1702. Como el bit F/E asociado con esta posición de memoria, el bit de F/E 1712, es ajustado a 0, los datos almacenados en la posición de memoria 1708 son inválidos. Como resultado, una señal es transmitida a la lógica de control 1742 para bloquear la lectura inmediata de datos desde esta posición de memoria.

Como se ha mostrado en la fig. 17K, la ID de la APU 1714 y la dirección de LS 1716 para esta orden de lectura siguiente están escritas en el segmento de memoria 1782. En este caso, la ID de la APU para la APU 1740 y la posición de memoria de LS para la posición de memoria de LS 1750 están escritas en el segmento de memoria 1762. Cuando los datos situados dentro de la posición de memoria 1708 resultan actuales, por ello, esta ID de la APU y posición de memoria de LS son usadas para determinar la posición a la que los datos actuales han de ser transmitidos.

Los datos en la posición de memoria 1708 resultan válidos y actuales cuando una APU escribe datos en esta posición de memoria. La escritura sincronizada de datos en la posición en memoria 1708 desde, por ejemplo, la posición de memoria 1732 de la APU 1722 está ilustrada en la fig. 17L. Esta escritura sincronizada de estos datos es permitida debido a que el bit de F/E 1712 para esta posición de memoria es ajustado a 0.

Como se ha mostrado en la fig. 17M, después de esta escritura, los datos en la posición de memoria 1708 resultan actuales y válidos. La ID de la APU 1714 y la dirección de LS 1716 procedente del segmento de memoria 1762, por ello, son inmediatamente leídas desde el segmento de memoria 1782, y esta información es a continuación borrada de este segmento. El bit de F/E 1712 es también ajustado a 0 en anticipación de la lectura inmediata de los datos en la posición de memoria 1708. Como se ha mostrado en la fig. 17N, después de leer la ID de la APU 1714 y la dirección de LS 1716, esta información es usada inmediatamente para leer los datos válidos en la posición de memoria 1708 a la posición de memoria de LS 1750 de la APU 1740. El estado final está mostrado en la fig. 17O. Esta figura muestra los datos válidos procedentes de la posición de memoria 1708 copiados a la posición de memoria 1750, el bit de ocupación en el segmento de memoria 1752 ajustado a 0 y el bit de F/E 1712 en el segmento de memoria 1762 ajustado a 0. El ajuste de este bit de ocupación a 0 permite que la posición de memoria de LS 1750 sea ahora accedida por la APU 1740 para cualquier propósito. El ajuste de este bit de F/E a 0 indica que los datos en la posición de memoria 1708 ya no son actuales ni válidos.

La fig. 18 resume las operaciones descritas anteriormente y los distintos estados de una posición de memoria de la DRAM basado en los estados del bit de F/E, la ID de la APU y la dirección de LS almacenados en el segmento de memoria correspondiente a la posición de memoria. La posición de memoria puede tener tres estados. Estos tres estados son un estado vacío 1880 en el que el bit de F/E es ajustado a 0 y no se proporciona información para la ID de la APU o la dirección de LS, un estado lleno 1882 en el que el bit de F/E es ajustado a 1 y no se proporciona información para la ID de la APU o la dirección de LS y un estado de bloqueo 1884 en el que el bit de F/E es ajustado a 0 y se proporciona información para la ID de la APU y la dirección de LS.

Como se ha mostrado en esta figura, en el estado vacío 1880, se permite una operación de escritura sincronizada y da como resultado una transición al estado lleno 1882. Una operación de lectura sincronizada, sin embargo, da como resultado una transición al estado de bloqueo 1884 debido a que los datos en la posición de memoria, cuando la posición de memoria está en el estado vacío, no son actuales.

En el estado lleno 1882, se permite una operación de lectura sincronizada y da como resultado una transición al estado vacío 1880. Por otro lado, una operación de escritura sincronizada en el estado lleno 1882 es prohibida para impedir la sobrescritura de datos válidos. Si tal operación de escritura es intentada en este estado, no ocurren cambios de estado y se transmite un mensaje de error a la lógica de control correspondiente de las APU.

En el estado de bloqueo 1884, la escritura sincronizada de datos en la posición de memoria es permitida y da como resultado una transición al estado vacío 1880. Por otro lado, un operación de lectura sincronizada en el estado de bloqueo 1884 es prohibida para impedir un conflicto con la operación lecturas sincronizada anterior que da como resultado este estado. Si se intenta una operación de lectura sincronizada en estado de bloqueo 1884, no ocurren cambios de estado y se tramite un mensaje de error a la lógica de control correspondiente de la APU.

El esquema descrito anteriormente para la lectura y escritura sincronizadas de datos desde y a la DRAM compartida también puede ser usado también para eliminar los recursos informáticos normalmente dedicados por un procesador para leer datos desde dispositivos externos y escribir datos en ellos. Esta función de entrada/salida (I/O) podría ser realizada por una PU. Sin embargo, usando una modificación de este esquema de sincronización, una APU que ejecuta un programa apropiado puede realizar esta función. Por ejemplo, usando este esquema, una PU que recibe una solicitud de interrupción para la transmisión de datos desde una interfaz de I/O iniciada por un dispositivo externo puede delegar la manipulación de esta solicitud a esta APU. La APU emite entonces una orden de escritura sincronizada a la interfaz de I/O. Esta interfaz a su vez señala al dispositivo externo que los datos pueden ser escritos ahora en la DRAM. La APU siguiente emite una orden de lectura sincronizada a la DRAM para ajustar el espacio de memoria importante de la DRAM en un estado de bloqueo. La APU también ajusta a 1 los bits de ocupación para las posiciones de memoria del almacenamiento local de la APU necesarias para recibir los datos. En el estado de bloqueo, los segmentos de memoria adicional asociados con el espacio de memoria importante de la DRAM contienen la ID de la APU y la dirección de las posiciones de memoria importantes del almacenamiento local de la APU. El dispositivo externo siguiente emite una orden de escritura sincronizada para escribir los datos directamente al espacio de memoria importante de la DRAM. Como este espacio de memoria está en el estado de bloqueo, los datos son inmediatamente leídos de este espacio en las posiciones de memoria del almacenamiento local de la APU identificados en los segmentos de memoria adicionales. Los bits de ocupación para estas posiciones de memorias son entonces ajustados a 0. Cuando el dispositivo externo completa la escritura de los datos, la APU emite una señal a la PU de que la transmisión está completada.

Usando este esquema, por ello, las transferencias de datos desde dispositivos externos pueden ser tratadas con una carga informática mínima sobre la PU. La APU que ha delegado esta función, sin embargo, debería ser capaz de emitir una solicitud de interrupción a la PU, y el dispositivo externo debería tener acceso directo a la DRAM.

La DRAM de cada PE incluye una pluralidad de entornos restringidos. Un entorno restringido define un área de la DRAM compartida más allá de la cual una APU particular, o un conjunto de APU, no puede leer o describir datos. Estos entornos restringidos proporcionan seguridad contra la corrupción de datos que son tratados por una APU por datos que son tratados por otra APU. Estos entornos restringidos también permiten la descarga de celdas de software desde la red 104 a un entorno restringido particular sin la posibilidad de que la celda de software corrompa datos en todas la DRAM. En el presente invento, los entornos restringidos son implantados en el hardware de las DRAM y los DMAC. Implantando las entornos restringidos en este hardware en vez de en el software, se obtienen ventajas en la velocidad y en la seguridad.

La PU de un PE controla los entornos restringidos asignados a las APU. Como la PU ejecuta solo programas de confianza, tales como un sistema operativo, este esquema pone en peligro la seguridad. De acuerdo con este esquema, la PU construye y mantiene una tabla de control de claves. Esta tabla de control de claves está ilustrada en la fig. 19. Como se ha mostrado en esta figura, cada entrada en la tabla 1902 de control de claves contiene una identificación (ID) 1904 para una APU, una clave 1906 de APU para esa APU y una máscara de claves 1908. El uso de esta máscara de claves es explicado a continuación. La tabla 1902 de control de claves es almacenada preferiblemente en una memoria relativamente rápida, tal como una memoria estática de acceso aleatorio (SRAM), y es asociada con el DMAC. Las entradas en la tabla 1902 de control de claves son controladas por la PU. Cuando una APU solicita la escritura de datos, o la lectura de datos desde, una posición de almacenamiento particular de la DRAM, el DMAC evalúa la clave 1906 de APU asignada a esa APU en la tabla 1902 de control de claves contra una clave de acceso de memoria asociada con esta posición de almacenamiento.

Como se ha mostrado en la fig. 20, un segmento 2010 de memoria dedicado es asignado a cada posición 2006 de almacenamiento accesible de una DRAM 2002. Una clave 2012 de acceso a la memoria para la posición de almacenamiento es almacenada en este segmento de memoria dedicado. Como se ha descrito antes, otro segmento 2008 de memoria dedicado adicional, también asociado con cada posición 2006 de almacenamiento accesible, almacena la información de sincronización para escribir datos en la posición de memoria, y leer datos desde ella.

En funcionamiento, una APU emite una orden de DMA al DMAC. Esta orden incluye la dirección de una posición 2006 de almacenamiento de la DRAM 2002. Antes de ejecutar esta orden, el DMAC busca la clave 1906 de la APU

solicitante en la tabla 1902 de control de claves usando la ID 1904 de la APU. El DMAC compara entonces la clave 1906 de la APU de la APU solicitante a la clave 2012 de acceso a la memoria almacenada en el segmento 2010 de memoria dedicado asociado con la posición de almacenamiento de la DRAM a la que busca acceso la APU. Si las dos claves no coinciden, la orden de DMA no es ejecutada. Por otro lado, si las dos claves coinciden, la orden de DMA continúa y el acceso a la memoria solicitado es ejecutado.

Una realización alternativa está ilustrada en la fig. 21. En esta realización, la PU también mantiene una tabla 2102 de control de acceso a la memoria. La tabla 2102 de control de acceso a la memoria contiene una entrada para cada entorno restringido dentro de la DRAM. En el ejemplo particular de la fig. 21, la DRAM contiene 64 entornos restringidos. Cada entrada en la tabla 2102 de control de acceso a la memoria contiene una identificación (ID) 2104 para un entorno restringido, una dirección 2106 de memoria de base, un tamaño 2108 de entorno restringido, una clave 2110 de acceso a la memoria y una máscara 2110 de claves de acceso. La dirección 2106 de memoria de base proporciona la dirección a la DRAM que inicia un entorno restringido de memoria particular. El tamaño 2108 del entorno restringido proporciona el tamaño del entorno restringido y, por ello, el punto final del entorno restringido particular.

La fig. 22 es un diagrama de flujo de las operaciones para ejecutar una orden de DMA usando la tabla 1902 de control de claves y la tabla 2102 de control de acceso a la memoria. En la operación 2202, una APU emite una orden de DMA al DMAC para el acceso a una posición o posiciones de memoria particular dentro de un entorno restringido. La orden incluye una ID 2104 de entorno restringido que identifica el entorno restringido particular para el que es solicitado el acceso. En la operación 2204, el DMAC busca la clave 1906 de las APU solicitantes en la tabla 1902 de control de claves usando la ID 1904 de las APU. En la operación 2206, el DMAC usa la ID 2104 de entorno restringido en la orden para buscar en la tabla 2102 de control de acceso a la memoria la clave 2110 de acceso a la memoria asociada con el entorno restringido. En la operación 2208, el DMAC compara la clave 1906 de la APU asignada a la APU solicitante con la clave 2110 de acceso asociada con el entorno restringido. En la operación 2210, se hace una determinación de si coinciden las dos claves. Si las dos claves no coinciden, el proceso se mueve a la operación 2212 donde la orden de DMA no prosigue y un mensaje de error es enviado bien a la APU solicitante, o bien a la PU o a ambas. Por otro lado, si en la operación 2210 se ha encontrado que las dos claves coinciden, el proceso continúa a la operación 2214 donde el DMAC ejecuta la orden de DMA.

Las máscaras de claves para las claves de la APU y las claves de la memoria de acceso proporcionan una mayor flexibilidad a este sistema. Una máscara de claves para una clave convierte un bit enmascarado en un comodín o carácter de reemplazamiento. Por ejemplo, si la máscara de claves 1908 asociada con la clave de la APU 1906 tiene sus dos últimos bits ajustados a la "máscara" designada por ejemplo, ajustando estos bits en la máscara de claves 1908 a 1, la clave de la APU puede ser o bien un 1 o bien un 0 y aún coincidir con la clave de acceso a la memoria. Por ejemplo, la clave de la APU podría ser 1010. Esta clave de APU normalmente permite acceso a un entorno restringido que tiene una clave de acceso de 1010. Si la máscara de claves de la APU para esta clave de APU es ajustada a 0001, sin embargo, entonces esta clave de APU puede ser usada para ganar acceso a entornos restringidos que tienen una clave de acceso o bien de 1010 o bien de 1011. Similarmente, una clave de acceso 1010 con un ajuste de máscara a 0001 puede ser accedida por una APU con una clave de APU o bien de 1010 o bien de 1011. Como tanto la máscara de claves de la APU como la máscara de claves de la memoria pueden ser usadas simultáneamente, pueden ser establecidas numerosas variaciones de accesibilidad por las APU a los entornos restringidos.

El presente invento proporciona también un modelo de programación para los procesadores del sistema 101. Este modelo de programación emplea celdas 102 de software. Estas celdas pueden ser transmitidas a cualquier procesador en la red 104 para su tratamiento. Este nuevo modelo de programación también utiliza la arquitectura modular única del sistema 101 y los procesadores del sistema 101.

Las celdas de software son tratadas directamente por las APU a partir del almacenamiento local de las APU. Las APU no operan directamente sobre ningún dato ni programa en la DRAM. Datos y programas en la DRAM son leídos en el almacenamiento local de la APU antes de que la APU procese estos datos y programas. El almacenamiento local de las APU, por ello, incluye un contador, apilamiento de programas y otros elementos de software para ejecutar estos programas. La PU controlaba las APU emitiendo órdenes de acceso directo a la memoria (DMA) al DMAC.

La estructura de celda 102 de software está ilustrada en la fig. 23. Como se ha mostrado en esta figura, una celda de software, por ejemplo la celda 2302 de software contiene la sección 2304 y el cuerpo 2306 de información de encaminamiento. La información contenida en la sección 2304 de información de encaminamiento depende del protocolo de red 104. La sección 2304 de información de encaminamiento contiene el encabezamiento 2308, la ID de destino 2310, la ID de fuente 2312 y la ID de réplica 2314. La ID de destino incluye una dirección de red. Bajo el protocolo TCP/IP, por ejemplo, la dirección de red es una dirección de protocolo de Internet (IP). La ID de destino 2310 incluye además la identidad de la PE y la APU a la que la celda debía ser transmitida para su tratamiento. La ID de la fuente 2314 contiene una dirección de red e identifica la PE y la APU a partir de las cuales la celda se ha originado para permitir el destino de PE y APU para obtener información adicional relativa a la celda si fuera necesario. La ID de la réplica 2314 contiene una dirección de red e identifica el PE y la APU a las que deberían ser dirigidas solicitudes relativas a la celda, y el resultado del tratamiento de la celda.

El cuerpo 2306 de la celda contiene información independiente del protocolo de la red. La parte despiezada ordenadamente de la fig. 23 muestra los detalles del cuerpo 2306 de la celda. El encabezamiento 2320 del cuerpo 2306 de

la celda identifica el comienzo del cuerpo de la celda. La interfaz 2322 de la celda contiene información necesaria para la utilización de la celda. Esta información incluye una ID global única 2324, las APU 2326 requeridas, el tamaño 2328 del entorno restringido y la ID previa de la celda 2330.

La ID única global 2324 identifica únicamente la celda 2302 de software a través de la red 104. La ID única global 2324 es generada sobre la base de la ID de la fuente 2312, por ejemplo la identificación única de un PE o una APU dentro de la ID de fuente 2312, y el tiempo y fecha de generación o transmisión de la celda 2302 de software. Las APU requeridas 2326 proporcionan el número mínimo de APU requeridas para ejecutar la celda. El tamaño 2328 del entorno restringido proporciona la cantidad de memoria protegida en la DRAM asociada a las APU requeridas necesaria para ejecutar la celda. La ID 2330 de la celda previa proporciona la identidad de una celda previa en un grupo de celdas que requieren ejecución secuencial, por ejemplo una corriente de datos.

La sección 2332 de puesta en práctica contiene la información del núcleo de la celda. Esta información incluye la lista de órdenes 2334 de DMA, los programas 2336 y los datos 2338. Los programas 2336 contienen los programas que han de ser ejecutados por unas APU (llamados "apulets"), por ejemplo los programas de las APU 2360 y 2362, y los datos 2338 contienen los datos que han de ser tratados con estos programas. La lista de órdenes 2334 de DMA contiene una serie de órdenes de DMA necesarias para iniciar los programas. Estas órdenes de DMA incluyen órdenes de DMA 2340, 2350, 2355 y 2358. La PU emite estas órdenes de DMA al DMAC.

Las órdenes de DMA 2340 incluyen la VID 2342. La VID 2342 es la ID virtual de una APU que se hace corresponder a una ID física cuando las órdenes de DMA son emitidas. La orden de DMA 2340 incluye también una orden de carga 2344 y dirección 2346. La orden de carga 2344 dirige la APU para leer información particular desde la DRAM al almacenamiento local. La dirección 2346 proporciona la dirección virtual en la DRAM que contiene esta información. La información puede ser, por ejemplo, programas a partir de la sección de programas 2336, datos a partir de la sección de datos 2338 u otros datos. Finalmente, la orden de DMA 2340 incluye la dirección 2348 de almacenamiento local. Esta dirección identifica la dirección en el almacenamiento local donde debería ser cargada la información. Las órdenes de DMA 2350 contienen información similar. Son también posibles otras órdenes de DMA.

La lista 2334 de órdenes de DMA también incluye una serie de órdenes de expulsión ("kick"), por ejemplo órdenes de expulsión 2355 y 2358. Las órdenes de expulsión son órdenes emitidas por una PU a una APU para iniciar el tratamiento de una celda. La orden de DMA 2355 incluye la ID virtual de APU 2352, la orden de expulsión 2354 y el contador de programa 2356. La ID virtual de APU 2352 identifica la APU que ha de ser expulsada, la orden de expulsión 2354 proporciona la orden de expulsión importante y el contador de programa 2356 proporciona la dirección para el contador de programa para ejecutar el programa. La orden de expulsión de DMA 2358 proporciona información similar para la misma APU u otra APU.

Como se ha observado, las PU tratan a las APU como procesadores independientes, no coprocesadores. Para controlar el tratamiento por las APU, por ello, la PU usa ordenes análogas a las llamadas de procedimiento remotas. Estas órdenes son designadas "Llamadas de Procedimiento Remotas de la APU" (ARPC). Una PU pone en práctica una ARPC emitiendo una sede de órdenes de DMA al DMAC. El DMAC carga el programa de APU y su imagen de apilamiento asociado al almacenamiento local de una APU. La PU emite entonces una expulsión inicial a la APU para ejecutar el Programa de APU.

La fig. 24 ilustra las operaciones de un ARPC para ejecutar un "apulet". Las operaciones realizadas por la PU al iniciar el tratamiento del "apulet" por una APU designada están mostradas en la primera parte 2402 de la fig. 24, y las operaciones realizadas por la APU designada en el tratamiento del "apulet" están mostradas en la segunda parte 2404 de la fig. 24.

En la operación 2410, la PU evalúa el "apulet" y a continuación designa una APU para tratar el "apulet". En la operación 2412, la PU asigna espacio en la DRAM para ejecutar el "apulet" emitiendo una orden de DMA al DMAC para ajustar las claves de acceso a la memoria para el entorno restringido o los entornos restringidos necesarios. En la operación 2414, la PU permite una solicitud de interrupción para la APU designada para señalar la terminación del "apulet". En la operación 2418, la PU emite una orden de DMA al DMAC para cargar el "apulet" desde la DRAM al almacenamiento local de la APU. En la operación 2420, la orden de DMA es ejecutada, y el "apulet" es leído desde la DRAM al almacenamiento local de las APU. En la operación 2422, la PU emite una orden de DMA a la DMAC para cargar el imagen de apilamiento asociado con el "apulet" desde la DRAM al almacenamiento local de las APU. En la operación 2423, la orden de DMA es ejecutada y el imagen de apilamiento es leído desde la DRAM al almacenamiento local de las APU. En la operación 2424, la PU emite una orden de DMA para el DMAC para asignar una clave a la APU para permitir que la APU lea y escriba datos desde y al entorno restringido o entornos restringidos designados en la operación 2412. En la operación 2426, el DMAC actualiza la tabla de control de claves (KTAB) con la clave asignada a la APU. En la operación 2428, la PU emite una orden de "expulsión" de la DAM a la APU para iniciar el tratamiento del programa. Otras órdenes de DMA pueden ser emitidas por la unidad en la ejecución de un ARPC particular dependiendo del "apulet" particular.

Como se ha indicado anteriormente, la segunda parte 2404 de la fig. 24 ilustra las operaciones realizadas por la APU al ejecutar el "apulet". En la operación 2430, la APU comienza a ejecutar el "apulet" en respuesta a la orden de expulsión emitida en la operación 2428. En la operación 2432, la APU, en la dirección del "apulet", evalúa el imagen de apilamiento asociado al "apulet". En la operación 2434, la APU emite múltiples órdenes DMA al DMAC

para cargar datos asignados cuando sea necesario por el imagen de apilamiento desde la DRAM al almacenamiento local de las APU. En la operación 2436, estas órdenes de DMA son ejecutadas, y los datos son leídos desde la DRAM al almacenamiento local de las APU. En la operación 2438, la APU ejecuta el "apulet" y genera un resultado. En la operación 2440, la APU emite una orden de DMA al DMAC para almacenar el resultado en la DRAM. En la operación 2442, la orden de DMA es ejecutada y el resultado del "apulet" es escrito desde el almacenamiento local de las APU a la DRAM. En la operación 2444, la APU emite una solicitud de interrupción a la PU para señalar que el ARPC ha sido completado.

La capacidad de las APU para realizar tareas independientemente bajo la dirección de una PU permite que una PU dedique un grupo de APU, y los recursos de memoria asociados con un grupo de APU, para realizar tareas extendidas. Por ejemplo, una PU puede dedicar una o más APU, y un grupo de entornos restringidos de memoria asociados con estas una o más APU, para recibir datos transmitidos sobre la red 104 durante un período prolongado y para dirigir los datos recibidos durante este período a una o más APU distintas y sus entornos restringidos de memoria asociados para un tratamiento adicional. Esta capacidad es particularmente ventajosa para tratar una corriente de datos transmitidos sobre la red 104, por ejemplo una corriente de MPEG o una corriente de datos de audio o de video ATRAC. Una PU puede dedicar una o más APU y sus entornos restringidos de memoria asociados para recibir estos datos y una o más APU distintas y sus entornos restringidos de memoria asociados para descomprimir y tratar adicionalmente estos datos. En otras palabras, la PU puede establecer una relación de tubería dedicada entre un grupo de APU y sus entornos restringidos de memoria asociados para tratar tales datos.

A fin de que tal tratamiento sea realizado eficientemente, sin embargo, las APU de tubería dedicada y las entornos restringidos deberían permanecer dedicados a la tubería durante períodos en los que el tratamiento de "apulets" que comprenden la corriente de datos no ocurre. En otras palabras, las APU dedicadas y sus entornos restringidos asociados deberían ser colocadas en un estado reservado durante estos períodos. La reserva de una APU y su entorno restringido o entornos restringidos de memoria asociados al terminar el tratamiento de un "apulet" es denominada una "terminación residente". Una terminación residente ocurre en respuesta a una instrucción procedente de una PU.

Las figs. 25, 26A y 26B ilustran el establecimiento de una estructura de tubería dedicada que comprende un grupo de APU y sus entornos restringidos asociados para el tratamiento de la corriente de datos, por ejemplo corriente de datos MPEG. Como se ha mostrado en la fig. 25, los componentes de esta estructura de tubería incluyen el PE 2502 y la DRAM 2518. El PE 2502 incluye la PU 2504, el DMAC 2506 y una pluralidad de APU, incluyendo la APU 2508, la APU 2510 y la APU 2512. Las comunicaciones entre la PU 2504, el DMAC 2506 y estas APU ocurren a través del bus 2514 del PE. El bus 2516 de ancho de banda amplio conecta el DMAC 2506 a la DRAM 2518. La DRAM 2518 incluye una pluralidad de entornos restringidos por ejemplo el entorno restringido 2520, el entorno restringido 2522, el entorno restringido 2524 y el entorno restringido 2526.

La fig. 26A ilustra las operaciones para establecer la tubería dedicada. En la operación 2610, la PU 2504 asigna a la APU 2508 para tratar un "apulet" de red. Un "apulet" de red comprende un programa para tratar el protocolo de red de la red 104. En este caso, este protocolo es el Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP). Los paquetes de datos de TCP/IP que son conformes a este protocolo son trasmitidos sobre la red 104. A la recepción, la APU 2508 procesa estos paquetes y ensambla los datos en los paquetes en las celdas 102 de software. En la operación 2612, la PU 2504 instruye a la APU 2508 para realizar terminaciones residentes a la terminación del tratamiento del "apulet" de red. En la operación 2614, la PU 2504 asigna las APU 2510 y 2512 para tratar "apulets" de MPEG. En la operación 2616, la PU 2504 designa el entorno restringido 2520 como un entorno restringido fuente para acceso por la APU 2508 y la APU 2510. En la operación 2618, la PU 2504 designa al entorno restringido 2522 como un entorno restringido de destino para acceso por la APU 2510. En la operación 2620, la PU 2504 designa al entorno restringido 2524 como un entorno restringido fuente para acceso por la APU 2508 y la APU 2512. En la operación 2622, la PU 2504 designa al entorno restringido 2526 como un entorno restringido de destino para acceso por la APU 2512. En la operación 2624, la APU 2510 y la APU 2512 envían órdenes de lectura sincronizadas a los bloques de memoria dentro, respectivamente, del entorno restringido fuente 2520 y del entorno restringido fuente 2524 para ajustar estos bloques de memoria al estado de bloqueo. El proceso finalmente se mueve a la operación 2628 donde el establecimiento del tubería dedicado es completo y los recursos dedicados al tubería son reservados. Las APU 2508, 2510 y 2512 y sus entornos restringidos asociados 2520, 2522, 2524 y 2526, por ello, entran en el estado reservado.

La fig. 26B ilustra las operaciones para tratar la corriente de datos de MPEG por este tubería dedicado. En la operación 2630, la APU 2508, que trata el "apulet" de red, recibe en su almacenamiento local paquetes de datos de TCP/IP desde la red 104. En la operación 2632, la APU 2508 trata estos paquetes de datos de TCP/IP y ensambla los datos dentro de estos paquetes en celdas de software 102. En la operación 2634, la APU 2508 examina el encabezamiento 2320 (fig. 23) de las celdas de software para determinar si las celdas contienen datos de MPEG. Si una celda no contiene datos de MPEG, entonces, en la operación 2636, la PU 2508 transmite la celda a un entorno restringido de propósito general designado dentro de la DRAM 2518 para tratar otros datos por otras APU incluidas dentro del tubería dedicado. Las APU 2508 también notifican a la PU 2504 de esta transmisión.

Por otro lado, si una celda de software contiene una corriente de datos de MPEG, entonces, en la operación 2638, la APU 2508 examina la ID de la celda previa 2330 (fig. 23) de la celda para identificar la corriente de datos de MPEG a los que pertenece la celda. En la operación 2640, la APU 2508 elige una APU del tubería dedicado para el tratamiento de la celda. En este caso, la APU 2508 elige la APU 2510 para tratar estos datos. Esta elección está basada

en la ID de la celda previa 2330 y los factores de equilibrado de carga. Por ejemplo, si la ID de la celda previa 2330 indica que la celda de software previa de la corriente de datos de MPEG a la que pertenece la celda de software fue enviada a la APU 2510 para tratamiento, entonces la celda de software presente normalmente también la enviará a la APU 2510 para su tratamiento. En la operación 2642, la APU 2508 emite una orden de escritura sincronizada para escribir los datos de MPEG en el entorno restringido 2520. Como este entorno restringido fue ajustado previamente al estado de bloqueo, los datos de MPEG, en la operación 2644 automáticamente son leídos desde el entorno restringido 2520 al almacenamiento local de la APU 2510. En la operación 2646, la APU 2510 trata los datos de MPEG en su almacenamiento local para generar datos de video. En la operación 2648, la APU 2510 escribe los datos de video en el entorno restringido 2522. En la operación 2650, la APU 2510 emite órdenes de lectura sincronizada al entorno restringido 2520 para preparar este entorno restringido para recibir datos de MPEG adicionales. En la operación 2652, la APU 2510 trata una terminación residente. Este tratamiento hace que esta APU entre en el estado reservado durante el cual la APU espera para tratar datos de MPEG adicionales en la corriente de datos de MPEG.

Otras estructuras dedicadas pueden ser establecidas entre un grupo de APU y sus entornos restringidos asociados para tratar otros tipos de datos. Por ejemplo, como se ha mostrado en la fig. 27, un grupo dedicado de APU, por ejemplo, las APU 2702, 2708 y 2714, puede ser establecido para realizar transformaciones geométricas sobre objetos tridimensionales para generar dos listas de presentación dimensionales. Estas dos listas de presentación dimensionales pueden ser además tratadas (creadas sus imágenes virtuales) por otras APU para generar datos de píxel. Para realizar este tratamiento, los entornos restringidos son dedicados a las APU 2702, 2708 y 2414 para almacenar los objetos tridimensionales y las listas de presentación resultantes del tratamiento de estos objetos. Por ejemplo, los entornos restringidos fuente 2704, 2720 y 2716 son dedicados para almacenar los objetos tridimensionales tratados por, respectivamente, la APU 2702, la APU 2708 y la APU 2714. De una manera similar, las entornos restringidos de destino 2706, 2712 y 2718 están dedicados a almacenar las listas de presentación resultantes del tratamiento de estos objetos tridimensionales por la APU 2702, la APU 2708 y la APU 2712, respectivamente.

La coordinación de la APU 2720 está dedicada a recibir en su almacenamiento local las listas de presentación procedentes de entornos restringidos de destino 2706, 2712 y 2718. La APU 2720 arbitra entre estas listas de presentación y las envía a otras APU para la creación de imagen virtual de los datos de píxel.

Los procesadores del sistema 101 también emplean un temporizador absoluto. El temporizador absoluto proporciona una señal de reloj a las APU y otros elementos de un PE que es tanto independiente como más rápida que la señal de reloj que acciona estos elementos. El uso de este temporizador absoluto está ilustrado en la fig. 28.

Como se ha mostrado en esta figura, el temporizador absoluto establece un presupuesto de tiempo para la realización de tareas por las APU. Este presupuesto de tiempo proporciona un tiempo para completar estas tareas que es mayor que el necesario para el tratamiento de las tareas por las APU. Como resultado, para cada tarea, hay dentro del presupuesto de tiempo, un período de ocupación y un período de espera. Todos los "apulets" son escritos para el tratamiento sobre la base de este presupuesto de tiempo independientemente del tiempo o velocidad de tratamiento real de las APU.

Por ejemplo, para una APU particular de un PE, una tarea particular puede ser realizada durante el período de ocupación 2802, del presupuesto de tiempo 2804. Como el período de ocupación 2802 es menor que el presupuesto de tiempo 2804, ocurre un periodo de espera 2806 durante el presupuesto de tiempo. Durante este período de espera, la APU va a un modo durmiente durante el cual se consume menos potencia por la APU.

Los resultados del tratamiento de una tarea no son esperados por otras APU, u otros elementos de un PE hasta que expira un presupuesto de tiempo 2804. Usando el presupuesto de tiempo establecido por el temporizador absoluto, por ello, los resultados del tratamiento de las APU siempre están coordinados independientemente de las velocidades de tratamiento reales de las APU.

En el futuro, la velocidad de tratamiento por las APU resultará más rápida. El presupuesto de tiempo establecido por el temporizador absoluto, sin embargo, permanecerá el mismo. Por ejemplo, como se ha mostrado en la fig. 28, una APU en el futuro ejecutará una tarea en un período más corto y, por ello, tendrá un período de de espera más largo. El período de ocupación 2808, por ello, es más corto que el período de ocupación 2802, y el período de espera 2810 es más largo que el período de espera 2806. Sin embargo, como los programas están escritos para el tratamiento sobre la base del mismo presupuesto de tiempo establecido con el temporizador absoluto, se mantiene la coordinación de los resultados del tratamiento entre las APU. Como resultado, APU más rápidas pueden procesar programas escritos para APU más lentas sin causar conflictos en los tiempos en los que se esperan los resultados de este tratamiento.

En lugar de un temporizador absoluto para establecer la coordinación entre las APU, la PU, o una o más APU designadas, se pueden analizar las instrucciones particulares o microcódigo que son ejecutados por una APU en el tratamiento de un "apulet" para estudiar los problemas en la coordinación del tratamiento paralelo de las APU creado por velocidades de funcionamiento mejoradas o diferentes. Instrucciones de "no funcionamiento" ("NOOP") pueden ser insertadas en las instrucciones y ejecutadas por alguna de las APU para mantener la terminación secuencial propia del tratamiento por las APU esperado por el amuleto. Insertando estas NOOP en las instrucciones, la temporización correcta para la ejecución de las APU de todas las instrucciones puede ser mantenida.

Aunque el invento ha sido descrito aquí con referencia a realizaciones particulares, ha de comprenderse que estas realizaciones son simplemente ilustrativas de los principios y aplicaciones del presente invento. Ha de entenderse por ello que pueden hacerse numerosas modificaciones en las realizaciones ilustrativas y que pueden ser consideradas otras disposiciones sin salir del marco del presente invento según ha sido definido por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un sistema (101) de tratamiento por ordenador, comprendiendo dicho sistema (101) de tratamiento: una primera memoria (225) para almacenar programas y datos asociados con dichos programas; una pluralidad de primeras unidades (402) de tratamiento para tratar dichos programas y dichos datos asociados; un controlador de memoria (205) para controlar los accesos a dicha primera memoria por dichas primeras unidades (402) de tratamiento; una segunda memoria para almacenar una tabla de acceso (2102) y una tabla de claves (1902), comprendiendo dicha tabla de acceso (2102) una pluralidad de entradas de acceso, incluyendo cada una de dichas entradas de acceso una clave de acceso y una identificación de un espacio de memoria dentro de dicha primera memoria asociado con dicha clave de acceso, comprendiendo dicha tabla de claves (1902) una pluralidad de entradas de claves, incluyendo cada una de dichas entradas de claves una identificación de una de dichas primeras unidades (402) de tratamiento y una clave de solicitud asociada con dicha primera unidad (402) de tratamiento; y una segunda unidad (203) de tratamiento para controlar dicho tratamiento de dichos programas y dichos datos asociados por dichas primeras unidades (402) de tratamiento, siendo accionable dicha segunda unidad (203) de tratamiento para construir y mantener dicha tabla de acceso (2102) y dicha tabla de claves (1902), siendo además accionable dicha segunda unidad (203) de tratamiento para dirigir cualquiera de dichas primeras unidades (402) de tratamiento para procesar uno de dichos programas, siendo accionable dicha primera unidad (402) de tratamiento para tratar dicho programa para emitir una solicitud a dicho controlador de memoria (205) para acceder a una posición de almacenamiento dentro de dicha primera memoria (225), siendo accionable dicho controlador de memoria, en respuesta a dicha solicitud, para comparar la clave de solicitud asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) con las claves de acceso en dicha tabla de acceso (2102), y si una de dichas claves de acceso corresponde con la clave de solicitud asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) y dicha posición de almacenamiento corresponde al espacio de memoria asociado con dicha clave de acceso e identificado en dicha tabla de acceso (2102), ejecutar dicha solicitud.

2. Un sistema de tratamiento por ordenador según la reivindicación 1, que comprende además una pluralidad de memorias locales, estando asociada cada una de dichas memorias locales con una de dichas primeras unidades (402) de tratamiento, y en el que dicha segunda unidad de tratamiento (203) es accionable para dirigir dicha primera unidad (402) de tratamiento para procesar dicho programa dirigiendo dicho controlador de memoria para transferir dicho programa desde dicha primera memoria a la memoria local asociada con dicha primera unidad de tratamiento, tratando después de ello dicha primera unidad (402) de tratamiento dicho programa desde dicha memoria local.

3. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que dicha clave de acceso comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y dicho controlador de memoria es accionable para ejecutar dicha solicitud solo sí la totalidad de dicha primera pluralidad de bits coincide con la totalidad de dicha segunda pluralidad de bits.

4. Un sistemas de tratamiento por ordenador según la reivindicación 1, en el que dicha clave de acceso comprende una primera pluralidad de bits y una máscara de claves y dicha clave de solicitud asociada con dicha primera unidad de tratamiento comprende una segunda pluralidad de bits, y dicho controlador de memoria es accionable para ejecutar dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con dicha segunda pluralidad de bits o la totalidad de dicha de dicha primera pluralidad de bits que no coincide con dicha segunda pluralidad de bits son enmascarados por dicha máscara de claves.

5. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que dicha clave de acceso comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits y una máscara de claves, y dicho controlador de memoria es accionable para ejecutar dicha solicitud sólo si la totalidad de dicha segunda pluralidad de bits coincide con dicha primera pluralidad de bits o los bits de dicha segunda pluralidad de bits que no coinciden con dicha primera pluralidad de bits son enmascarados por dicha máscara de claves.

6. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que cada una de dichas entradas de claves incluye además una máscara de claves.

7. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que cada una de dichas entradas de acceso que incluye además una máscara de claves.

8. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que cada una de dichas entradas de claves incluye además una máscara de claves y cada una de dichas entradas de accesos incluye además una máscara de claves.

9. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que cada una de dichas entradas de acceso incluye además una entrada de posición base y una entrada de tamaño, proporcionando dicha entrada de posición base la dirección de comienzo dentro de dicha primera memoria de dicho espacio de memoria asociado con dicha entrada de acceso y proporcionando dicha entrada de tamaño el tamaño dentro de dicha primera memoria de dicho espacio de memoria asociado con dicha entrada de acceso.

10. Un sistema de tratamiento por ordenador según la reivindicación 1, en el que cada una de dichas primeras memoria es una memoria dinámica de acceso aleatorio y es una memoria principal para dicho sistema de tratamiento por ordenador.

11. Un método de tratamiento por ordenador, comprendiendo dicho método: almacenar en una primera memoria (225) programas y datos asociados con dichos programas; tratar con una pluralidad de primeras unidades (402) de tratamiento dichos programas y dichos datos asociados; controlar con un controlador de memoria (205) accesos a dicha primera memoria por dichas primeras unidades (402) de tratamiento; construir con una segunda unidad (203) de tratamiento en una segunda memoria una tabla de acceso (2102) y una tabla de claves (1902), comprendiendo dicha tabla de acceso (2102) una pluralidad de entradas de acceso, incluyendo cada una de dichas entradas de acceso una clave de acceso y una identificación de un espacio de memoria dentro de dicha primera memoria asociado con dicha clave de acceso, comprendiendo dicha tabla de claves (19002) una pluralidad de entradas de claves, incluyendo cada una de dichas entradas de claves una identificación de una de dichas primeras unidades (402) de tratamiento y una clave de solicitud asociada con dicha primera unidad (402) de tratamiento; controlar con dicha segunda unidad (203) de tratamiento dicho tratamiento de dichos programas y dichos datos asociados por dichas primeras unidades (402) de tratamiento; dirigir con dicha segunda unidad (203) de tratamiento de dichas primeras unidades (402) de tratamiento para procesar uno de dichos programas; emitir desde dicha primera unidad (402) de tratamiento, al tratar dicho programa, una solicitud a dicho controlador de memoria (205) para acceder a una posición de almacenamiento dentro de dicha primera memoria; comparar, en respuesta a dicha solicitud, la clave de solicitud asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) con las claves de accesos en dicha tabla de accesos (2102); si una de dichas claves de acceso corresponde con la clave de solicitud asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) y dicha posición de almacenamiento corresponde al espacio de memoria asociado con dicha clave de accesos identificada en dicha tabla de acceso (2102), ejecutar dicha solicitud.

12. Un método de tratamiento por ordenador según la reivindicación 11, que comprende además dirigir con dicha segunda unidad (203) de tratamiento dicha primera unidad (402) de tratamiento para tratar dicho programa emitiendo desde dicha segunda unidad (203) de tratamiento una orden a dicho controlador de memoria para transferir dicho programa desde dicha primera memoria a una memoria local asociada con dicha primera unidad (402) de tratamiento, y después de ello tratar con dicha primera unidad (402) de tratamiento dicho programa desde dicha memoria local.

13. Un método de tratamiento por ordenador según la reivindicación 11, en el que dicha clave de acceso comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y comprendiendo además ejecutar con dicho controlador de memoria dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con la totalidad de dicha segunda pluralidad de bits.

14. Un método de tratamiento por ordenador según la reivindicación 11, en el que dicha clave de acceso comprende una primera pluralidad de bits y una máscara de claves y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y comprendiendo además ejecutar con dicho controlador de memoria dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con dicha segunda pluralidad de bits o si todos los bits de dicha primera pluralidad de bits que no coinciden con dicha segunda pluralidad de bits son enmascarados por dicha máscara de claves.

15. Un método de tratamiento por ordenador según la reivindicación 11, en el que dicha clave de acceso comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits y una máscara de claves, y comprendiendo además ejecutar con dicho controlador de memoria dicha solicitud sólo si la totalidad de dicha segunda pluralidad de bits coincide con dicha primera pluralidad de bits o todos los bits de dicha segunda pluralidad de bits que no coinciden con dicha primera pluralidad de bits son enmascarados con dicha máscara de claves.

16. Un método de tratamiento por ordenador según la reivindicación 11, en el que cada una de dichas entradas de claves incluye además una máscara de claves.

17. Un método de tratamiento por ordenador según la reivindicación 11, en el que cada una de dichas entradas de acceso incluye además una máscara de claves.

18. Un método de tratamiento por ordenador según la reivindicación 10, en el que cada una de dichas entradas de claves que incluye además una máscara de claves y cada una de dichas entradas de acceso incluye además una máscara de claves.

19. Un método de tratamiento por ordenador según la reivindicación 11, en el que cada una de dichas entradas de acceso incluye además una entrada de posición base y una entrada de tamaño, proporcionando dicha entrada de posición base la dirección de comienzo dentro de dicha primera memoria de dicho espacio de memoria asociado con dicha entrada de acceso y proporcionando dicha entrada de tamaño el tamaño dentro de dicha primera memoria de dicho espacio de memoria asociado con dicha entrada de acceso.

20. Un método de tratamiento por ordenador según la reivindicación 11, en el que dicha primera memoria es una memoria dinámica de acceso aleatorio y es una memoria principal para dicho sistema (101) de tratamiento por ordenador.

21. Un sistema (101) de tratamiento por ordenador, comprendiendo dicho sistema (101) de tratamiento: una primera memoria (225) para almacenar programas y datos asociados con dichos programas, comprendiendo dicha primera memoria (225) una pluralidad de posiciones de almacenamiento accesibles, comprendiendo cada una de dichas posiciones de almacenamiento accesibles un segmento (2010) de memoria adicional asociado con dicha posición de almacenamiento accesible y conteniendo una clave de acceso (2012) para dicha posición de almacenamiento accesible; una pluralidad de primeras unidades (402) de tratamiento para tratar dichos programas y dichos datos asociados; un controlador de memoria (205) para controlar los accesos a dicha primera memoria por dichas primeras unidades (402) de tratamiento; una segunda memoria para almacenar una tabla de claves (1902) que comprende una pluralidad de entradas de claves, incluyendo cada una de dichas entradas de claves una identificación (1904) de una de dichas primeras unidades (402) de tratamiento y una clave de solicitud (1906) asociada con dicha primera unidad (402) de tratamiento; una segunda unidad (203) de tratamiento para controlar dicho tratamiento de dichos programas y dichos datos asociados por dichas primeras unidades (402) de tratamiento, siendo accionable dicha segunda unidad (203) de tratamiento para asignar y mantener dichas claves de acceso (2012) y construir y mantener dicha tabla de claves (1902), siendo además accionable dicha segunda unidad (203) de tratamiento para dirigir cualquiera de dichas primeras unidades (402) de tratamiento para tratar uno de dichos programas, siendo accionable dicha primera unidad (402) de tratamiento en el tratamiento de dicho programa para emitir una solicitud a dicho controlador de memoria (205) para acceder a una de dichas posiciones de almacenamiento accesibles, siendo accionable dicho controlador de memoria, en respuesta a dicha solicitud, para comparar la clave de solicitud (1906) asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) con la clave de acceso (2012) contenida en el segmento de memoria adicional asociado con dicha posición de almacenamiento accesible, y si la clave de solicitud (1906) asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) corresponde con dicha clave de acceso (2012) contenida en dicho segmento de memoria adicional asociado con dicha posición de almacenamiento accesible, ejecutar dicha solicitud.

22. Un sistema de tratamiento por ordenador según la reivindicación 21, que comprende además una pluralidad de memorias locales, estando asociada cada una de dichas memorias locales con una de dichas primeras unidades (402) de tratamiento, y en el que dicha segunda unidad (203) de tratamiento es accionable para dirigir dicha primera unidad (402) de tratamiento para tratar dicho programa dirigiendo dicho controlador de memorias para transferir dicho programa desde dicha primera memoria a la memoria local asociada con dicha primera unidad (402) de tratamiento, procesando después de ello dicha primera unidad (402) de tratamiento dicho programa desde dicha memoria local.

23. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que dicha clave de acceso contenida en dicho segmento de memoria asociado con dicha posición de almacenamiento accesible comprende una primera pluralidad de bits, y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y dicho controlador de memoria es accionable para ejecutar dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con la totalidad de dicha segunda pluralidad bits.

24. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que dicha clave de acceso contenida en dicho segmento de memoria asociado con dicha posición de almacenamiento accesible comprende una primera pluralidad de bits y una máscara de claves y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y dicho controlador de memoria es accionable para ejecutar dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con dicha segunda pluralidad de bits o los bits de dicha primera pluralidad de bits que no coinciden con dicha segunda pluralidad de bits son enmascarados por dicha máscara de claves.

25. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que dicha clave de acceso contenida en dicho segmento de memoria asociado con dicha posición de almacenamiento accesible comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits y una máscara de claves, y dicho controlador de memoria es accionable para ejecutar dicha solicitud sólo si la totalidad de dicha segunda pluralidad de bits coincide con dicha primera pluralidad de bits o todos los bits de dicha segunda pluralidad de bits que no coinciden con dicha primera pluralidad de bits son enmascarados por dicha máscara de claves.

26. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que cada una de dichas entradas de claves incluye además una máscara de claves.

27. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que cada uno de dichos segmentos de memoria adicional contiene también una máscara de claves.

28. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que cada una de dichas entradas de claves incluye además una máscara de claves y cada uno de dichos segmentos de memoria adicional contiene también una máscara de claves.

29. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que cada uno de dichos segmentos de memoria adicionales contiene también información del estado relativo al estado de los datos almacenados en la posición de almacenamiento accesible asociada con dicho segmento de memoria adicional.

30. Un sistema de tratamiento por ordenador según la reivindicación 21, en el que dicha primera memoria es una memoria dinámica de acceso aleatorio y es una memoria principal para dicho sistema (101) de tratamiento por ordenador.

31. Un método de tratamiento por ordenador, comprendiendo dicho método: almacenar en una primera memoria (225) programas y datos asociados con dichos programas, comprendiendo dicha primera memoria una pluralidad de posiciones de almacenamiento accesibles, incluyendo cada una de dichas posiciones de almacenamiento accesibles un segmento (2040) de la memoria adicional asociado con dicha posición de almacenamiento accesible; almacenar en dicho segmento de memoria adicional para cada posición de almacenamiento accesible una clave de acceso (2012) para dicha posición de almacenamiento accesible; tratar con una pluralidad de primeras unidades (402) de tratamiento dichos programas y dichos datos asociados; controlar con un controlador de memoria (205) accesos a dicha primera memoria por dichas primeras unidades (402) de tratamiento; almacenar en una segunda memoria una tabla de claves (1902) que comprende una pluralidad de entradas de claves, incluyendo cada una de dichas entradas de claves una identificación (1904) de una de dichas primeras unidades (402) de tratamiento y una clave de solicitud (1906) asociada con dicha primera unidad (402) de tratamiento; controlar con una segunda unidad (203) de tratamiento dicho tratamiento de dichos programas y dichos datos asociados por dichas primeras unidades (402) de tratamiento; construir con dicha segunda unidad (203) de tratamiento dicha tabla de claves (1902); dirigir con dicha segunda unidad (203) de tratamiento cualquiera de dichas primeras unidades (402) de tratamiento para tratar uno de dichos programas; emitir desde dicha primera unidad (402) de tratamiento, en el tratamiento de dicho programa, una solicitud a dicho controlador de memoria para acceder a una de dichas posiciones de almacenamiento accesibles; comparar, en respuesta a dicha solicitud, la clave de solicitud (1906) asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) con la clave de accesos (2012) contenida en el segmento de memoria adicional asociado con dicha posición de almacenamiento accesible; si la clave de solicitud (1906) asociada con dicha primera unidad (402) de tratamiento en dicha tabla de claves (1902) corresponde con dicha clave de accesos (2012) contenida en dicho segmento de memoria adicional asociado con dicha posición de almacenamiento accesible, ejecutar dicha solicitud.

32. Un método de tratamiento por ordenador según la reivindicación 31, que comprende además dirigir con dicha segunda unidad (203) de tratamiento dicha primera unidad (402) de tratamiento para tratar dicho programa dirigiendo dicho controlador de memoria para transferir dicho programa desde dicha primera memoria a la memoria local asociada con dicha primera unidad (402) de tratamiento, y después de ello tratar con dicha primera unidad (402) de tratamiento dicho programa desde dicha memoria local.

33. Un método de tratamiento por ordenador según la reivindicación 31, en el que dicha clave de acceso contenida en dicho segmento de memoria adicional asociado con dicha posición de almacenamiento accesible comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y comprendiendo además ejecutar con dicho controlador de memoria dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con la totalidad de dicha segunda pluralidad de bits.

34. Un método de tratamiento por ordenador según la reivindicación 31, en el que dicha clave de acceso contenida en dicho segmento de memoria asociado con dicha posición de almacenamiento accesible comprende una primera pluralidad de bits y una máscara de claves y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits, y comprendiendo además ejecutar con dicho controlador de memoria dicha solicitud sólo si la totalidad de dicha primera pluralidad de bits coincide con dicha segunda pluralidad de bits o si todos los bits de dicha primera pluralidad de bits que no coinciden con dicha segunda pluralidad de bits son enmascarados por dicha máscara de claves.

35. Un método de tratamiento por ordenador según la reivindicación 31, en el que dicha clave de acceso contenida en dicho segmento de memoria adicional asociado con dicha posición de almacenamiento accesible comprende una primera pluralidad de bits y dicha clave de solicitud asociada con dicha primera unidad (402) de tratamiento comprende una segunda pluralidad de bits y una máscara de claves, y comprendiendo además ejecutar con dicho controlador de memoria dicha solicitud sólo si la totalidad de dicha segunda pluralidad de bits coincide con dicha primera pluralidad de bits o todos los bits de dicha segunda pluralidad de bits que no coinciden con dicha primera pluralidad de bits son enmascarados con dicha máscara de claves.

36. Un método de tratamiento por ordenador según la reivindicación 31, en el que cada una de dichas entradas de claves incluye además una máscara de claves.

37. Un método de tratamiento por ordenador según la reivindicación 31, en el que cada uno de dichos elementos de memoria adicional contiene también una máscara de claves.

38. Un método de tratamiento por ordenador según la reivindicación 31, en el que cada una de dichas entradas de claves incluye además una máscara de claves y cada uno de dichos segmentos de memoria adicional contiene también una máscara de claves.

39. Un método de tratamiento por ordenador según la reivindicación 31, en el que cada uno de dichos segmentos de memoria adicional contiene también información de estado relativa al estado de datos almacenados en la posición de memoria accesible asociada con dicho segmento de memoria adicional.

40. Un método de tratamiento ordenador según la reivindicación 31, en el que dicha primera memoria es una memoria dinámica de acceso aleatorio y es una memoria principal para dicho sistema (101) de tratamiento por ordenador.

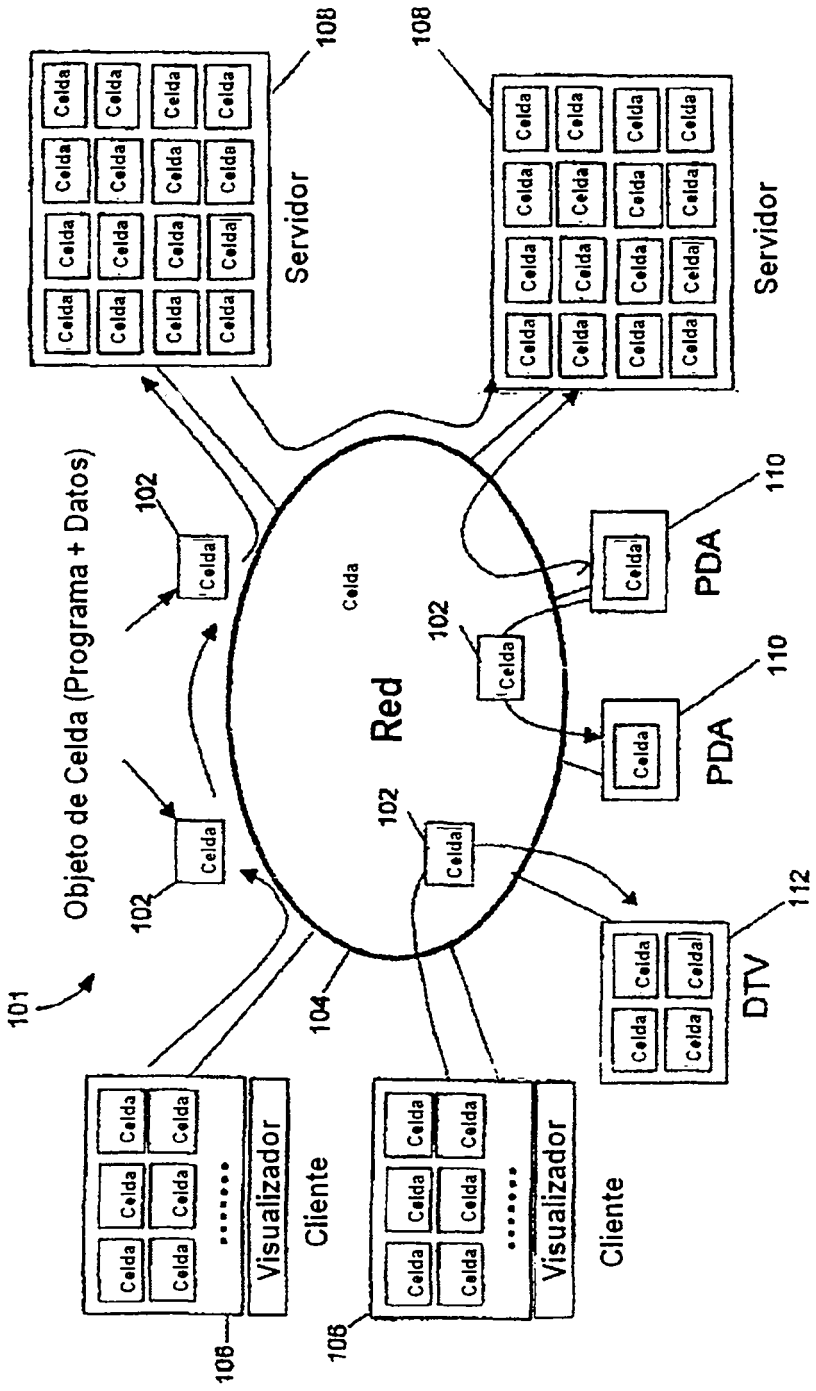
Objeto de Celda (Programa + Datos)
Celda
Red
Servidor
Servidor
PDA
PDA
DTV
Cliente
Cliente
Visualizador
Visualizador
101
102
104
106
108
110
112

FIG. 1

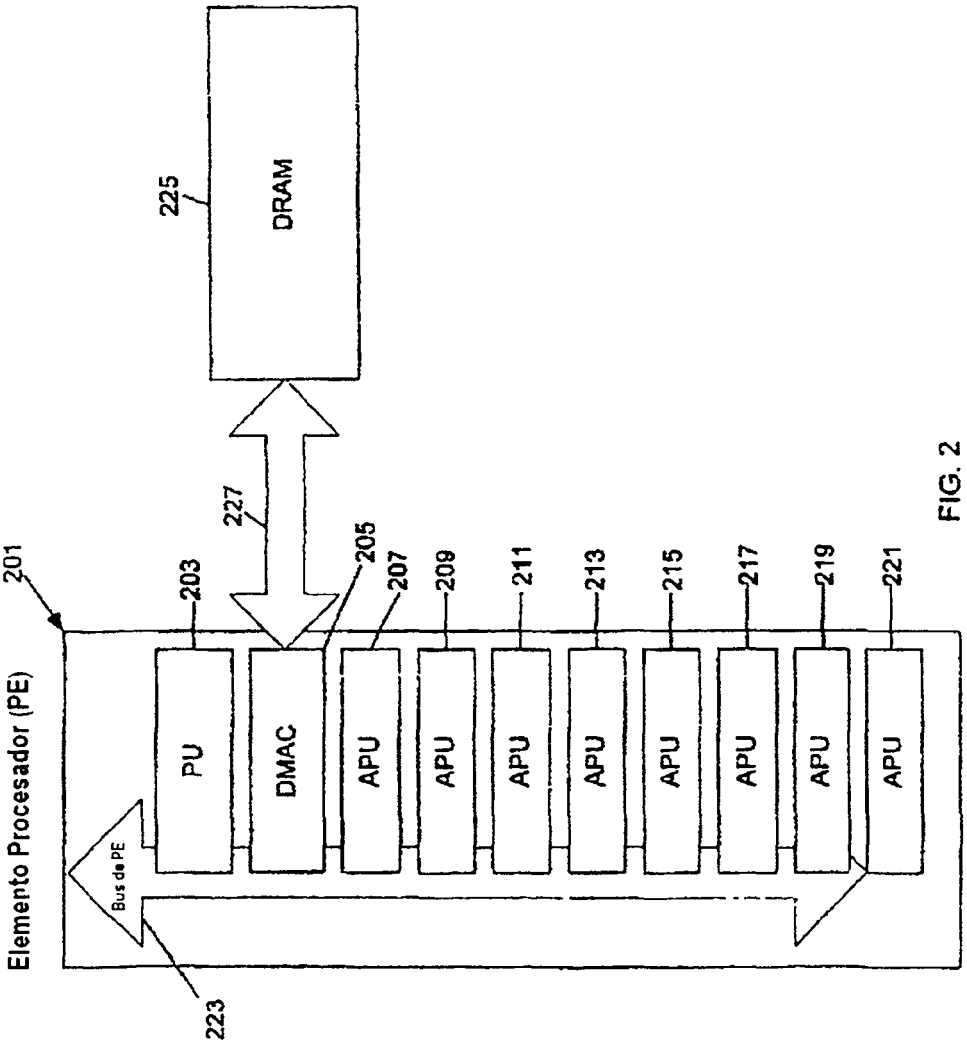
225
DRAM
227
201
203
205
207
209
211
213
215
217
219
221
Elemento Procesador (PE)
PU
DMAC
APU
APU
APU
APU
APU
APU
APU
APU
Bus de PE
223

FIG. 2

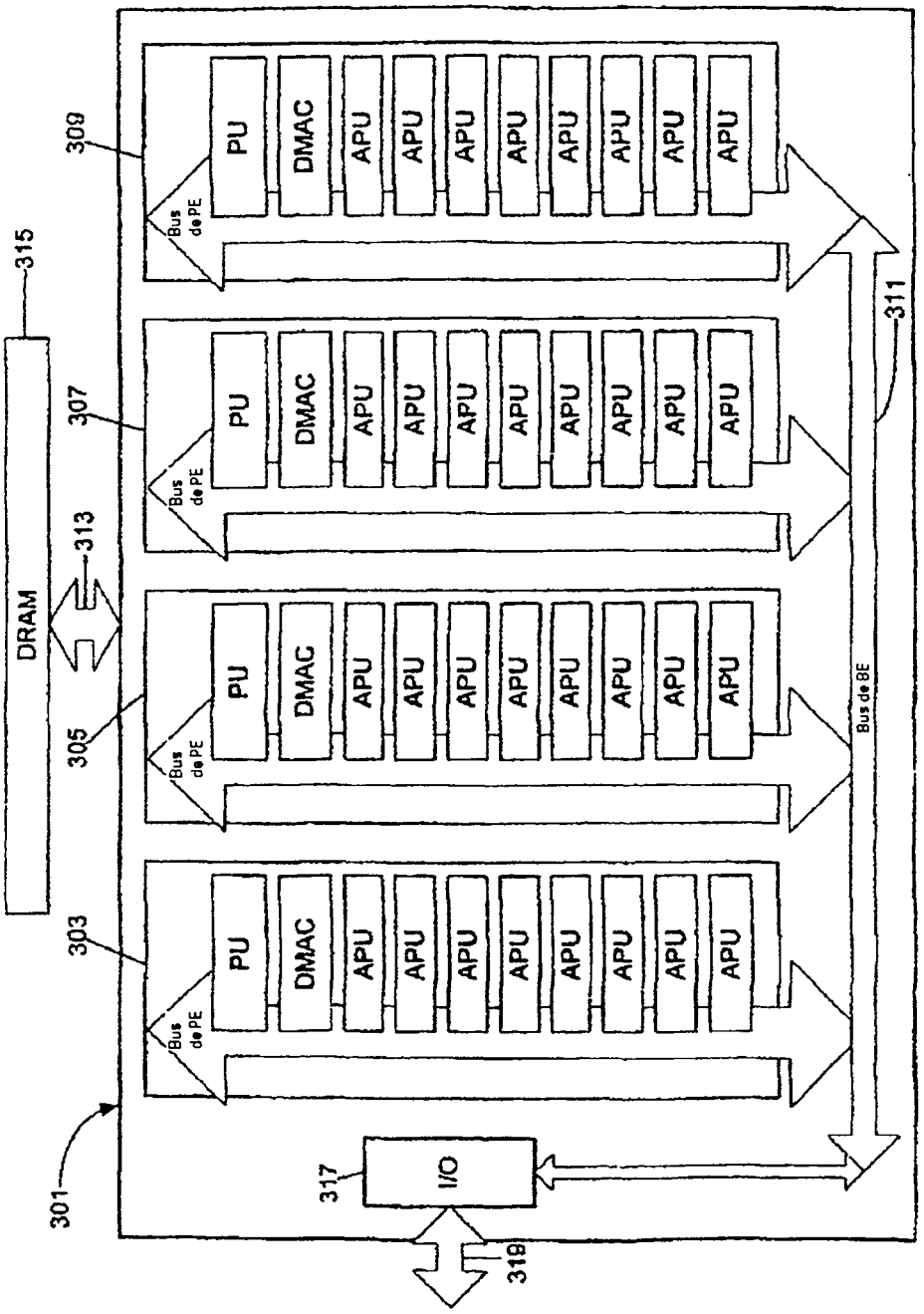
301
303
305
307
309
311
313
315
317
318
DRAM
I/O
PU
DMAC
APU
Bus de PE
Bus de BE

FIG. 3

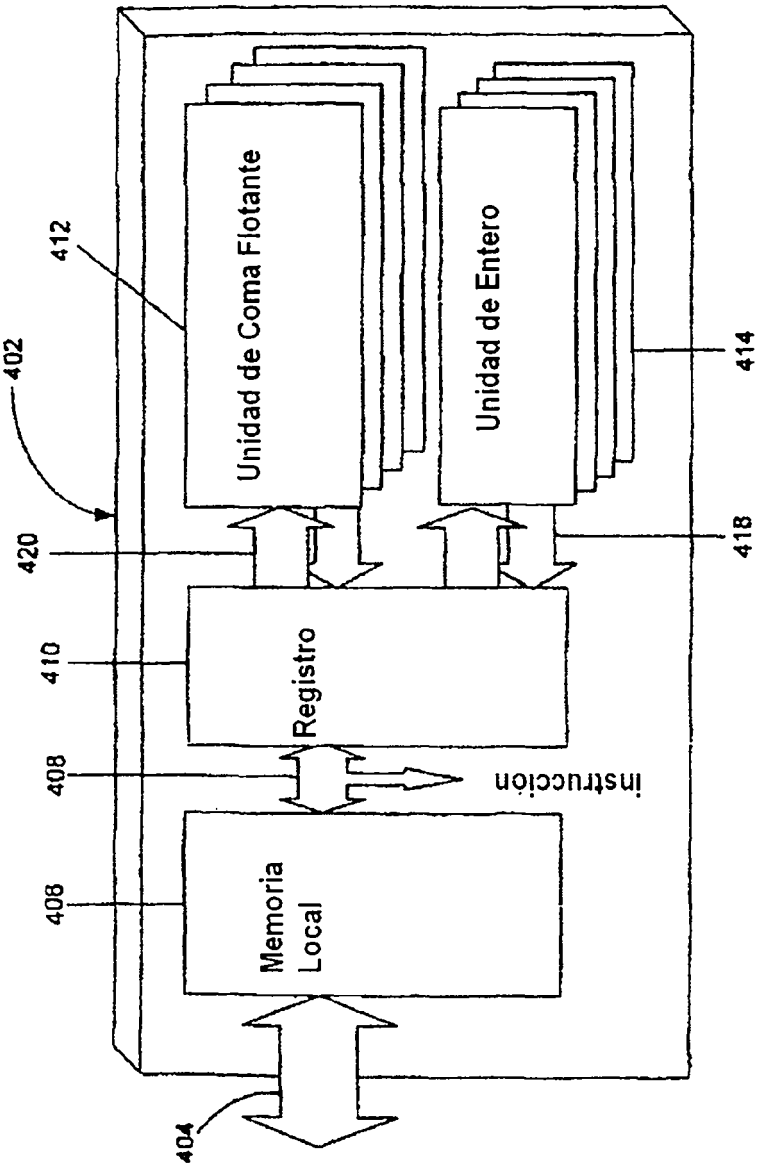
Unidad de Coma Flotante
Unidad de Entero
Registro
Memoria Local
instrucción
402
404
408
408
410
412
414
418
420

FIG. 4

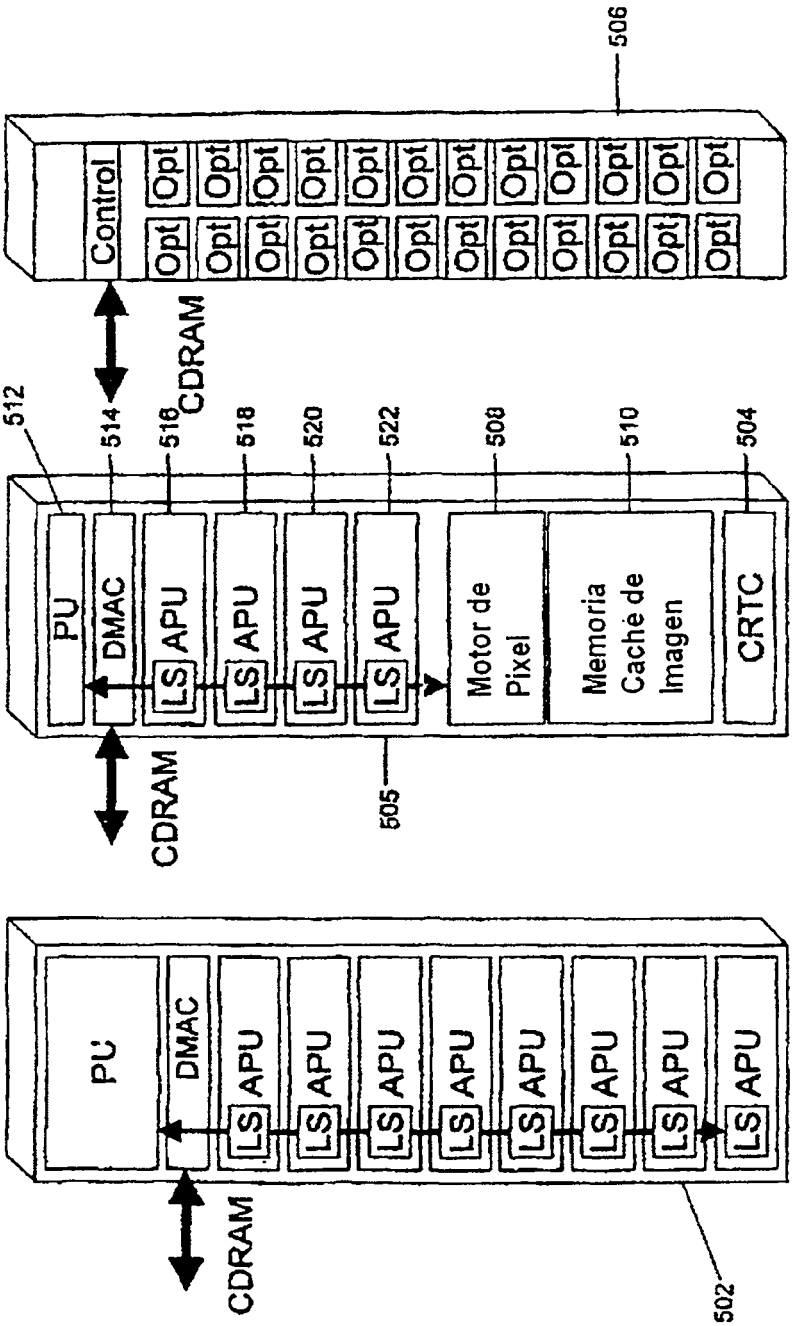
506
Control
Opt
CDRAM
512
514
516
518
520
522
508
510
504
PU
DMAC
LS
APU
Motor de Pixel
Memoria Caché de Imagen
CRTC
CDRAM
505
PU
DMAC
LS
APU
CDRAM
502

FIG. 5

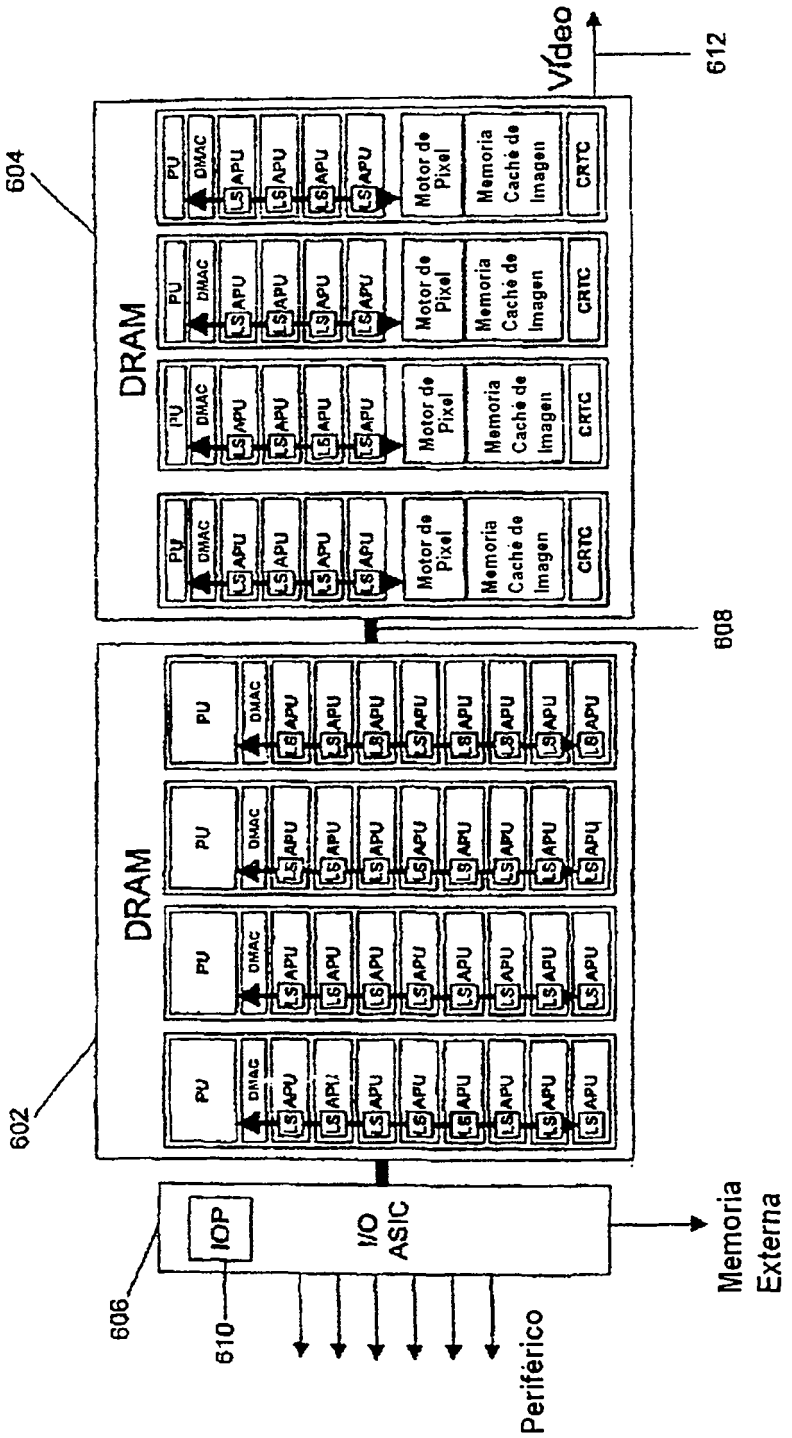
Video
612
604
DRAM
PU
DMAC
APU
Motor de Pixel
Memoria Caché de Imagen
CRTC
608
602
606
610
IOP
I/O ASIC
Periférico
Memoria Externa

FIG. 6

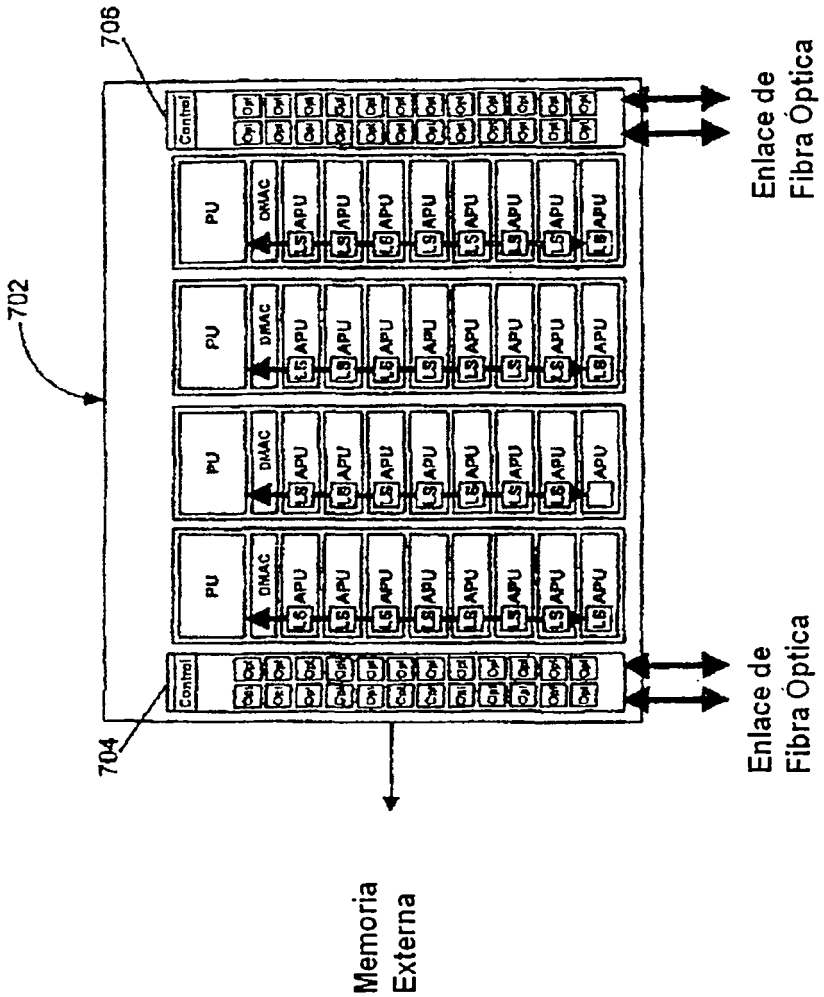
706
702
704
Control
PU
DMAC
APU
LS
Enlace de
Fibra Óptica
Enlace de
Fibra Óptica
Memoria
Externa

FIG. 7

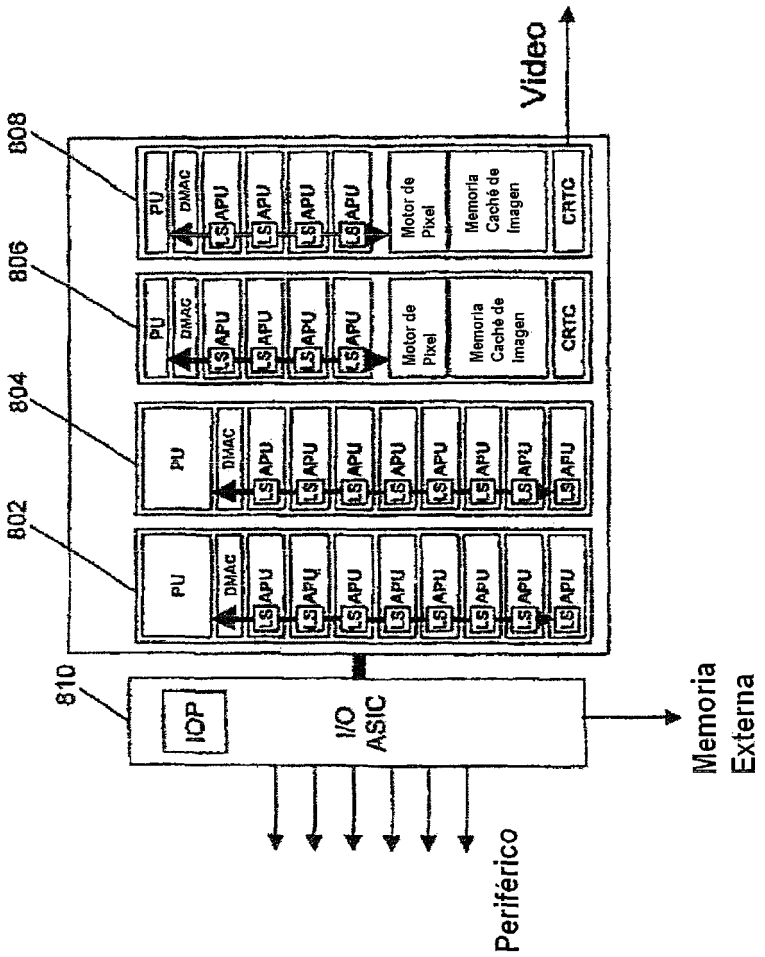
Video
808
806
804
802
810
PU
DMAC
LS
APU
Motor de Pixel
Memoria Caché de Imagen
CRTC
IOP
I/O ASIC
Memoria Externa
Periférico

FIG. 8

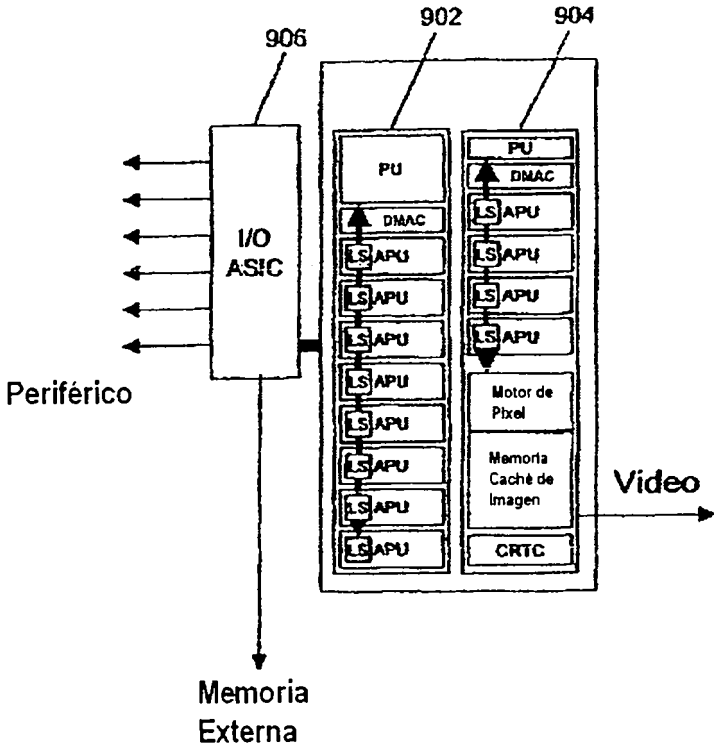
906
902
904
I/O
ASIC
Periférico
PU
DMAC
LS
APU
Motor de
Pixel
Memoria
Caché de
Imagen
CRTC
Video
Memoria
Externa

FIG. 9

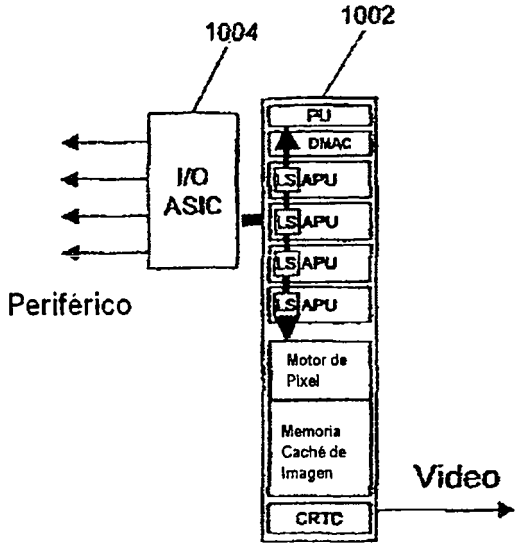
1004
1002
I/O
ASIC
Periférico
PU
DMAC
LS
APU
LS
APU
LS
APU
LS
APU
Motor de
Pixel
Memoria
Caché de
Imagen
CRTC
Video

FIG. 10

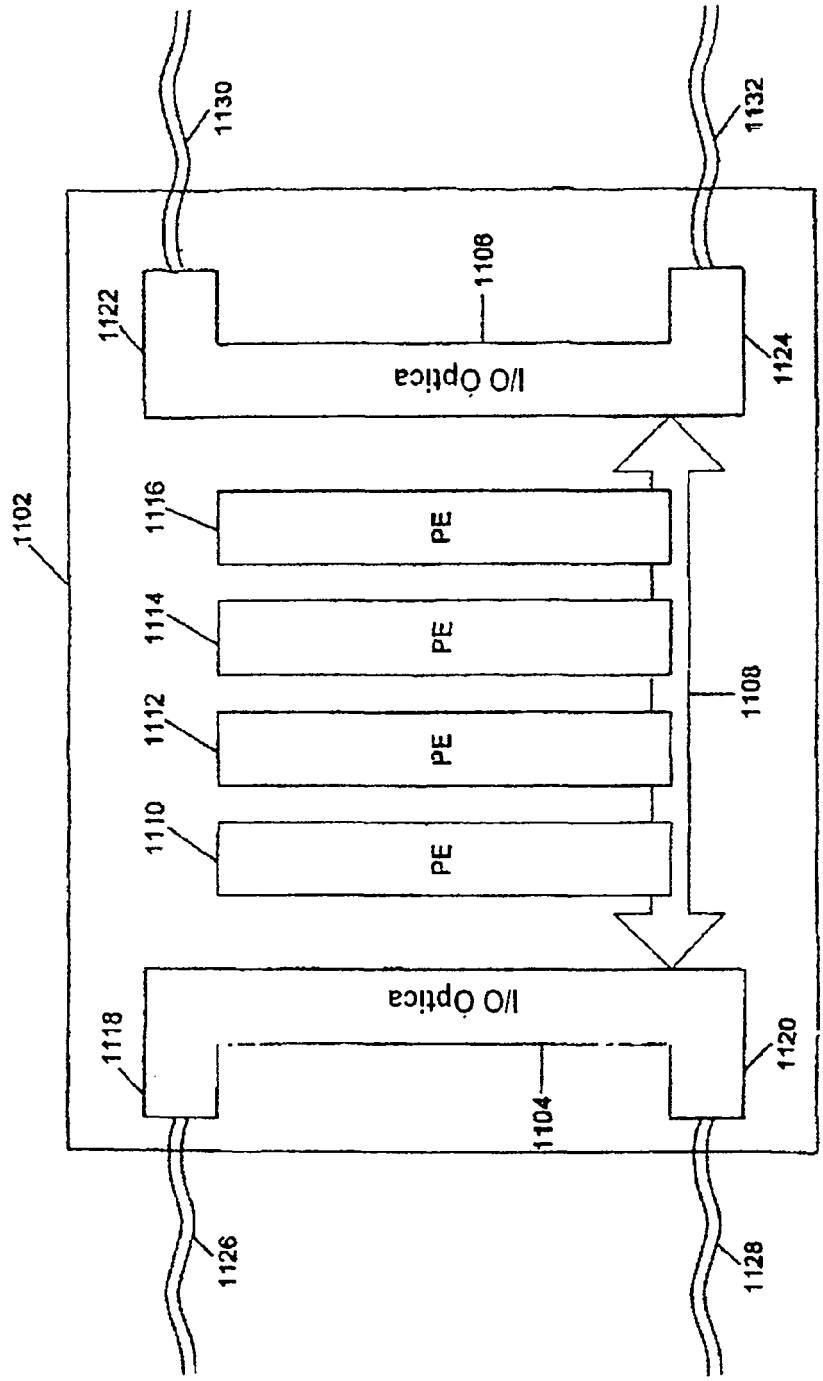
1130
1132
1108
1122
I/O Óptica
1124
1102
1116
PE
1114
PE
1112
PE
1110
PE
1108
I/O Óptica
1118
1120
1104
1126
1128

FIG. 11A

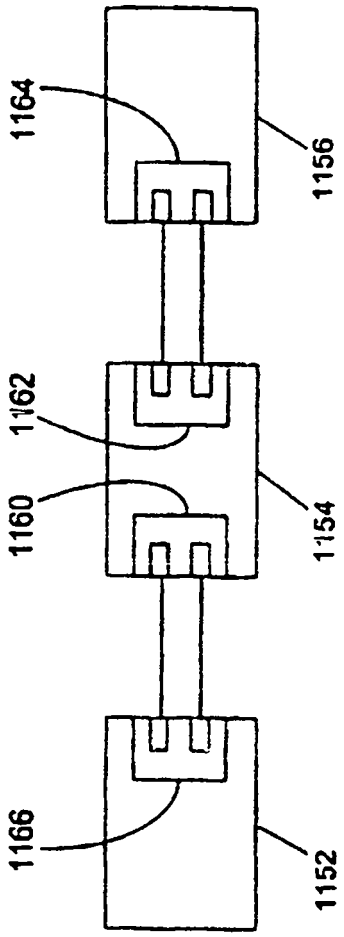

FIG. 11B

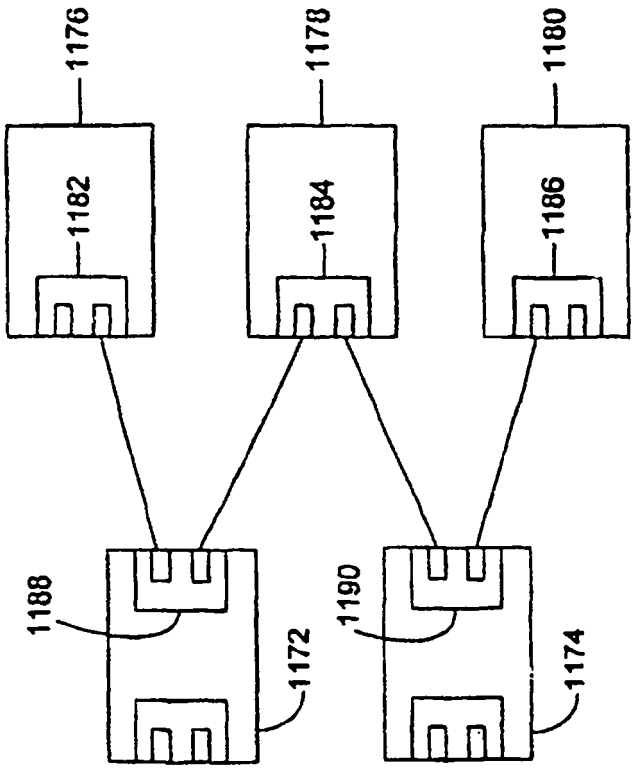

FIG. 11C

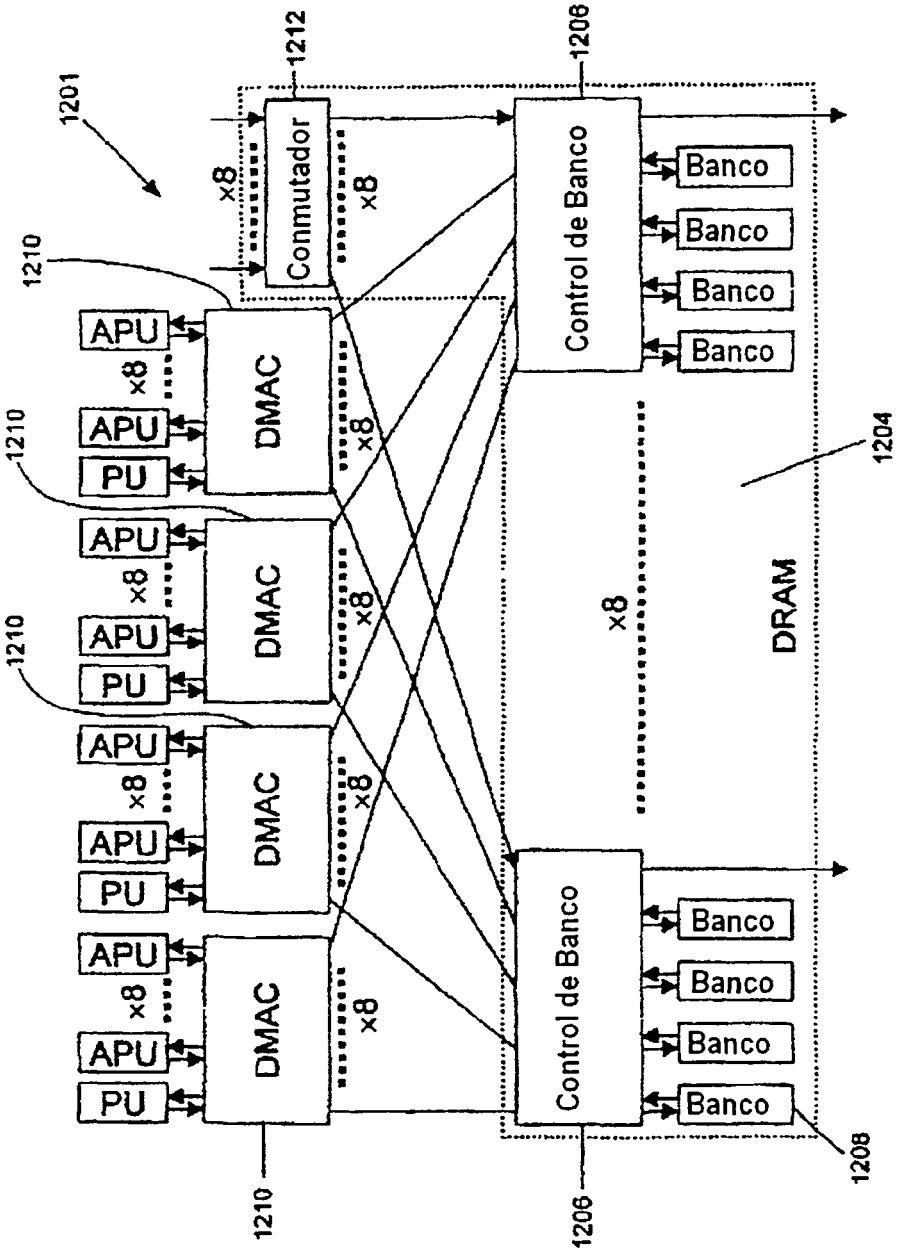
1201
1212
1208
Conmutador
Control de Banco
Banco
Banco
Banco
Banco
1210
APU
APU
PU
DMAC
x8
1204
DRAM
Control de Banco
Banco
Banco
Banco
Banco
1206
1208

FIG. 12A

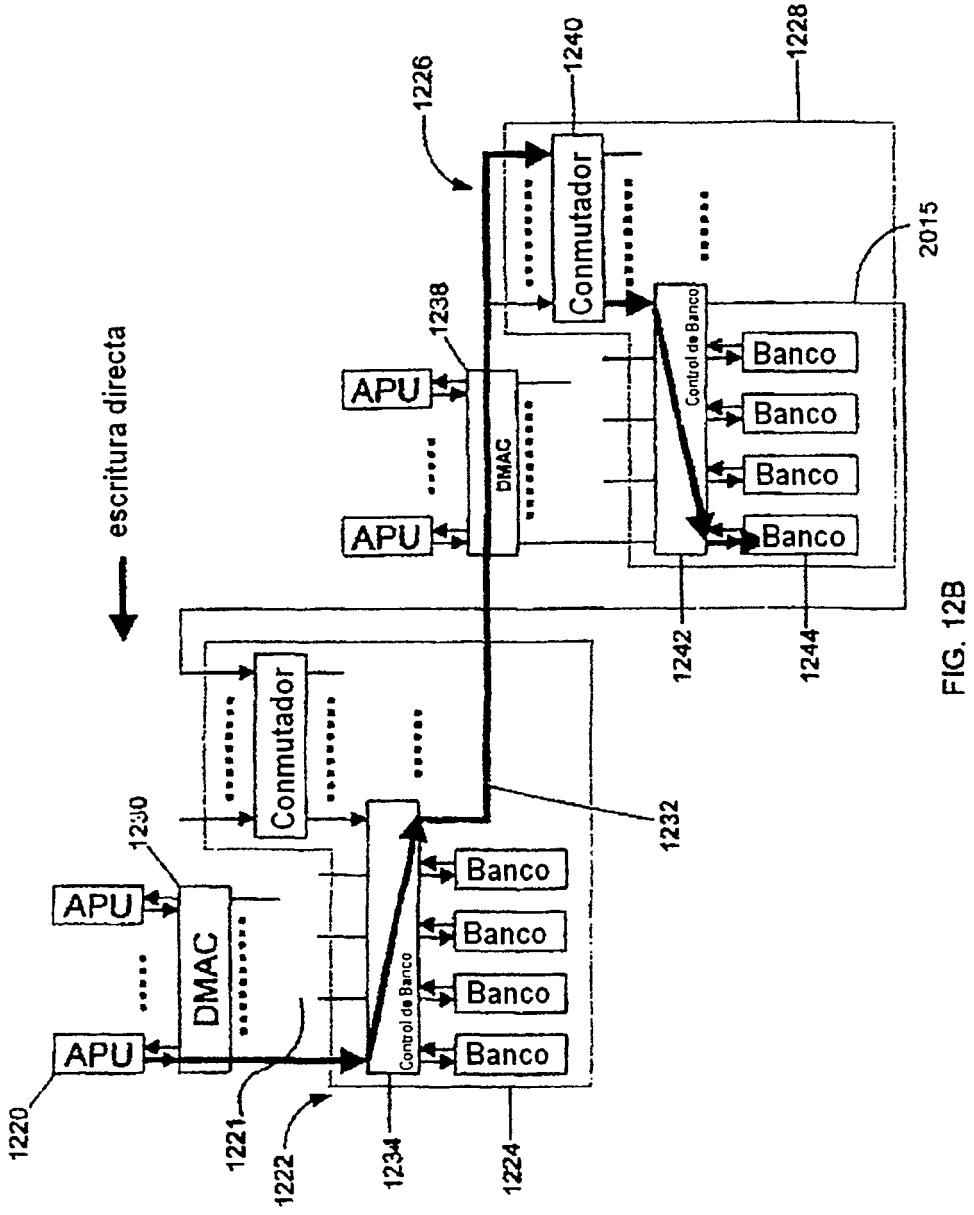
escritura directa
1220
APU
APU
1230
DMAC
1221
1222
Conmutador
1234
Control de Banco
Banco
Banco
Banco
Banco
1224
1232
1226
1240
1228
Conmutador
2015
1238
APU
APU
DMAC
Control de Banco
Banco
Banco
Banco
Banco
1242
1244

FIG. 12B

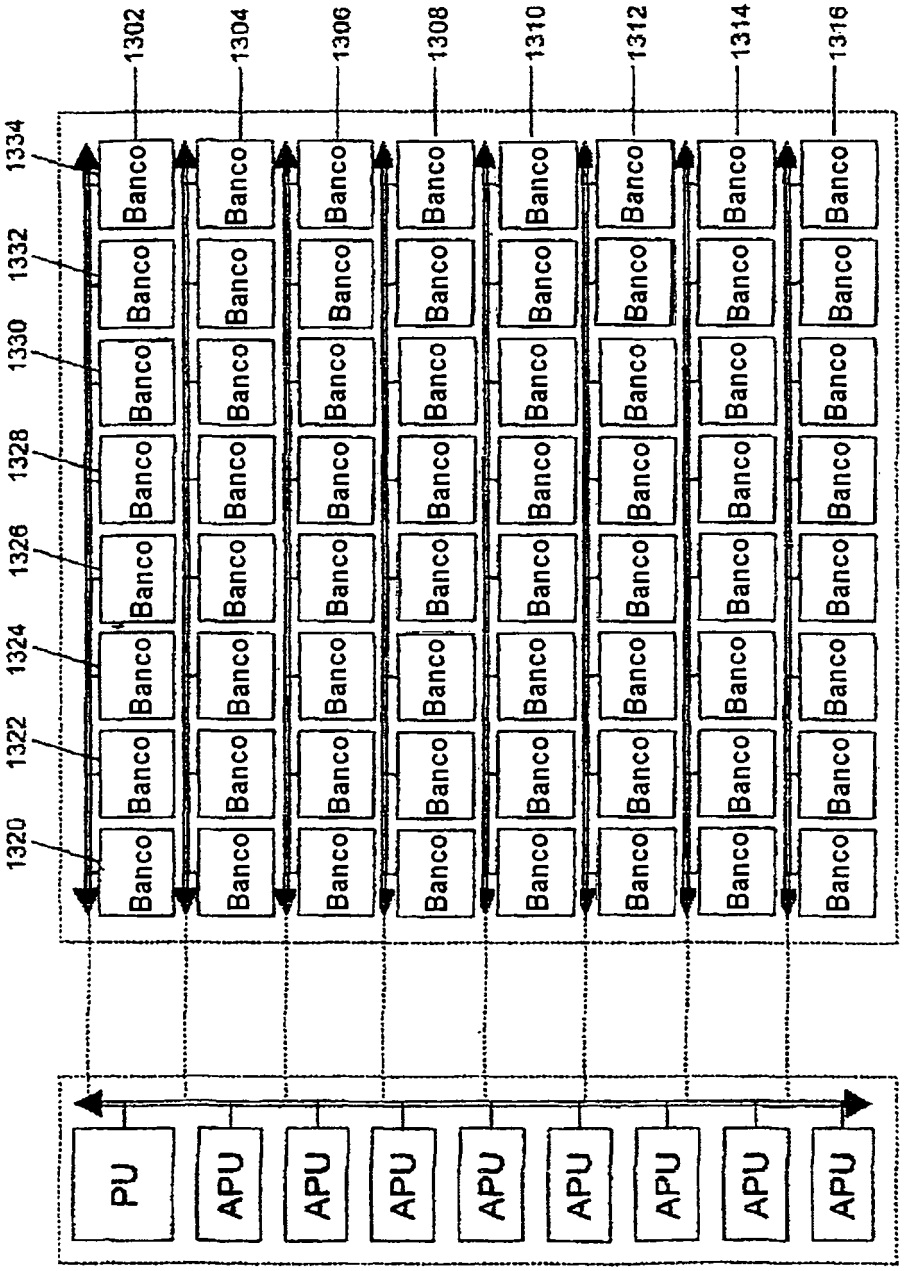
1302
1304
1306
1308
1310
1312
1314
1316
1334
1332
1330
1328
1326
1324
1322
1320
Banco
DRAM
PU
APU

FIG. 13

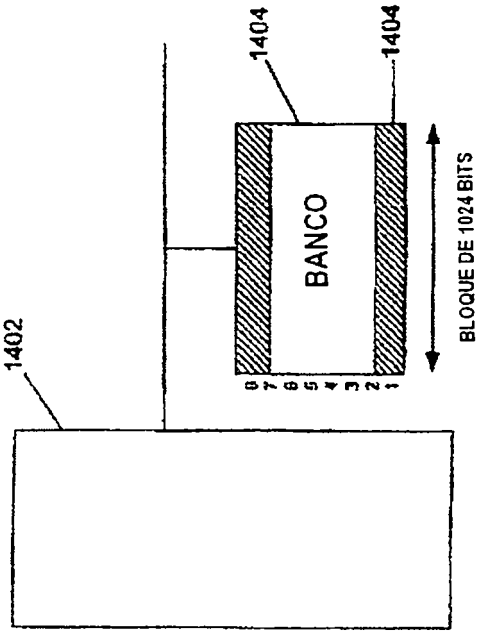

FIG. 14A

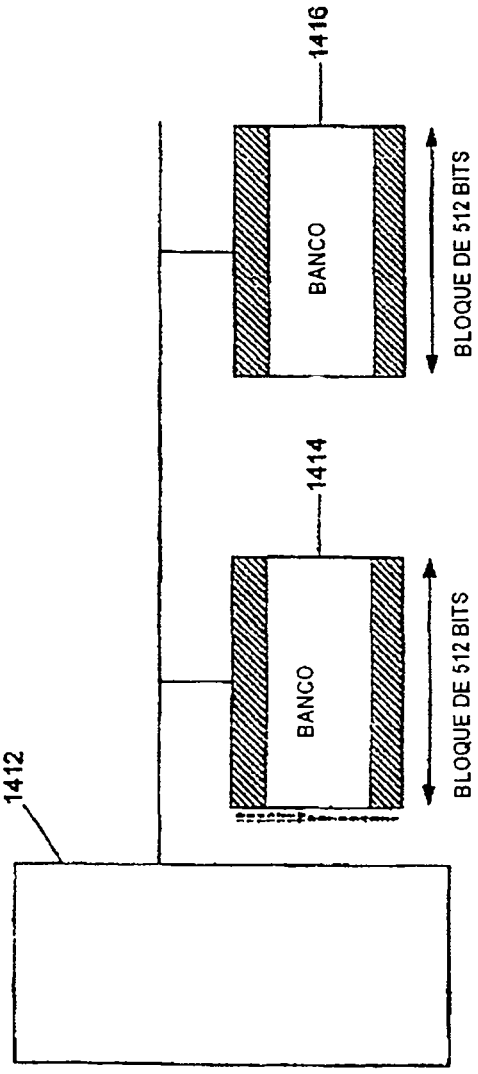
1416
BANCO
BLOQUE DE 512 BITS
1414
BANCO
BLOQUE DE 512 BITS
1412

FIG. 14B

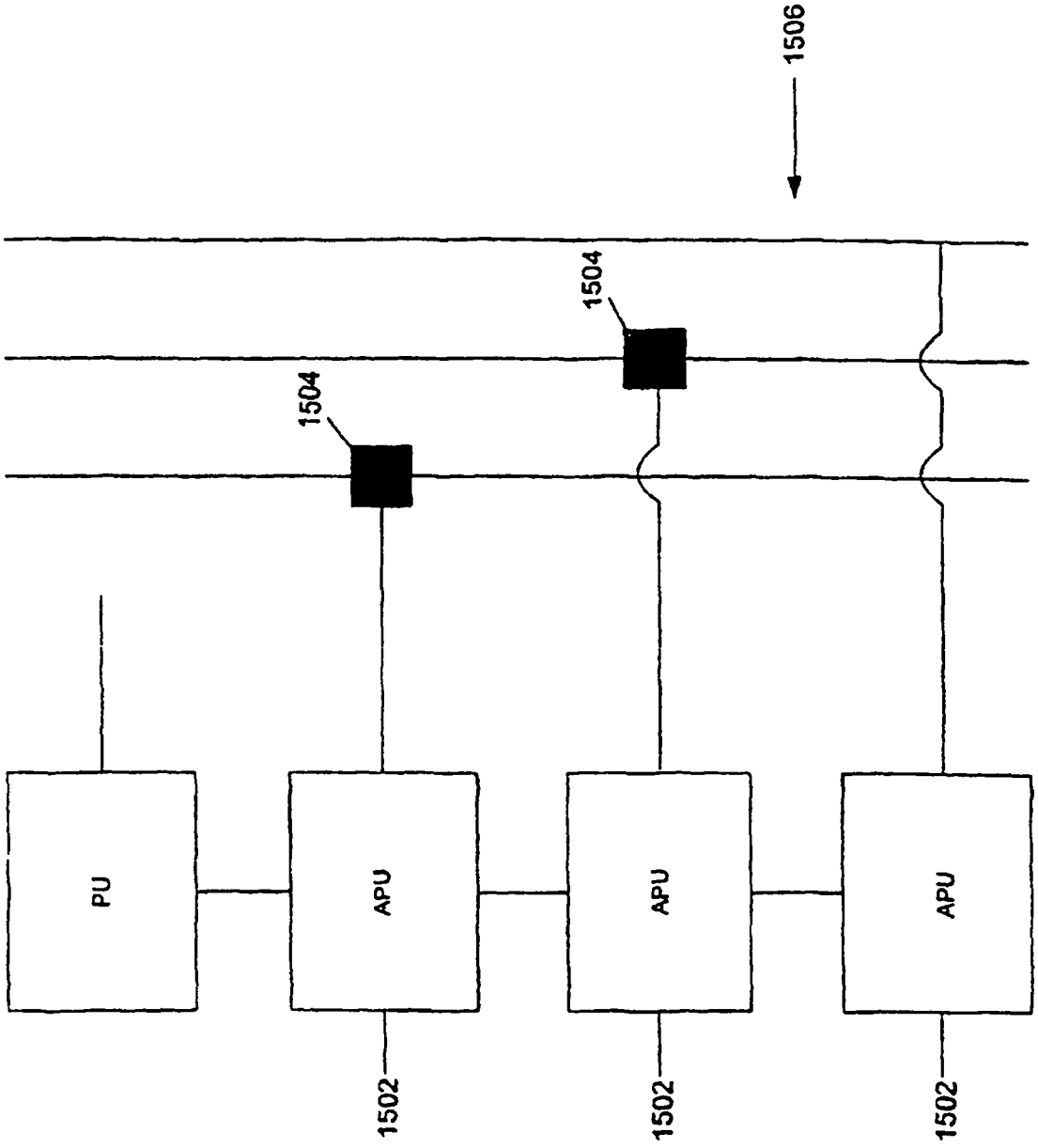

FIG. 15

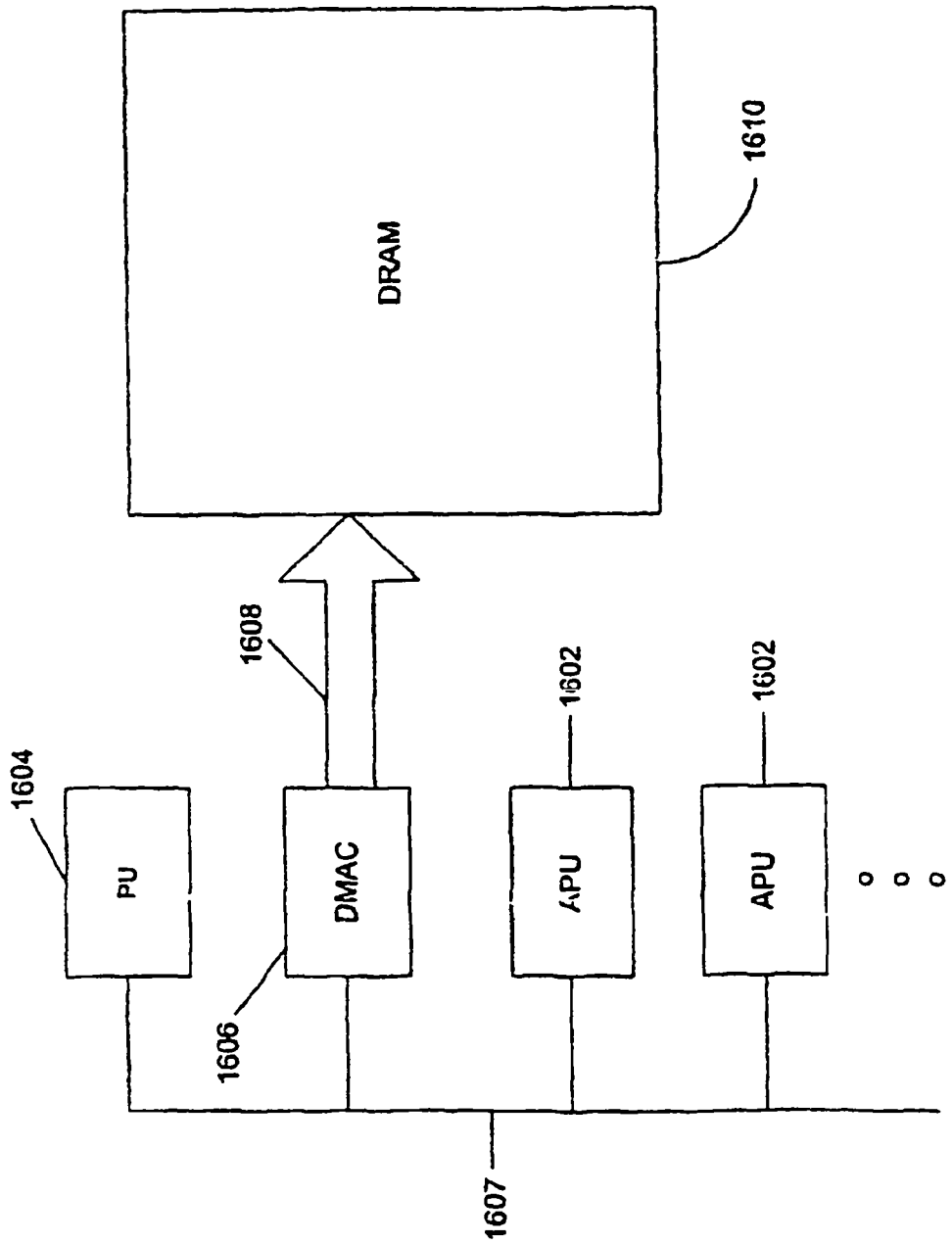

FIG. 16

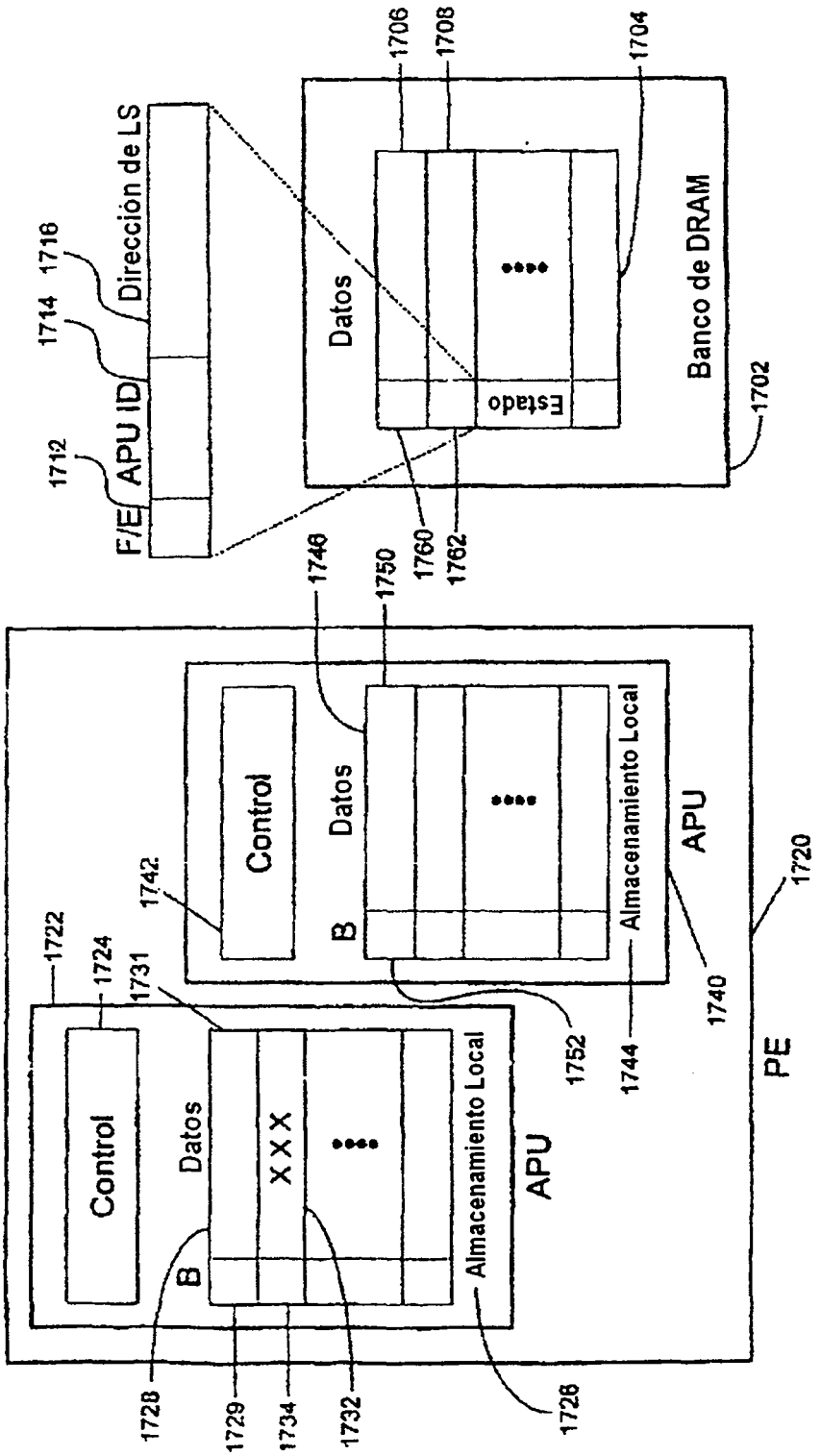
1706
1708
1704
Dirección de LS
1716
1714
APU ID
1712
F/E
Datos
Estado
Banco de DRAM
1702
1746
1750
1760
1762
Control
Datos
B
Almacenamiento Local
APU
1742
1722
1724
1731
1752
1744
1740
1720
PE
Control
Datos
B
XXX
Almacenamiento Local
APU
1728
1729
1734
1732
1728

FIG. 17A

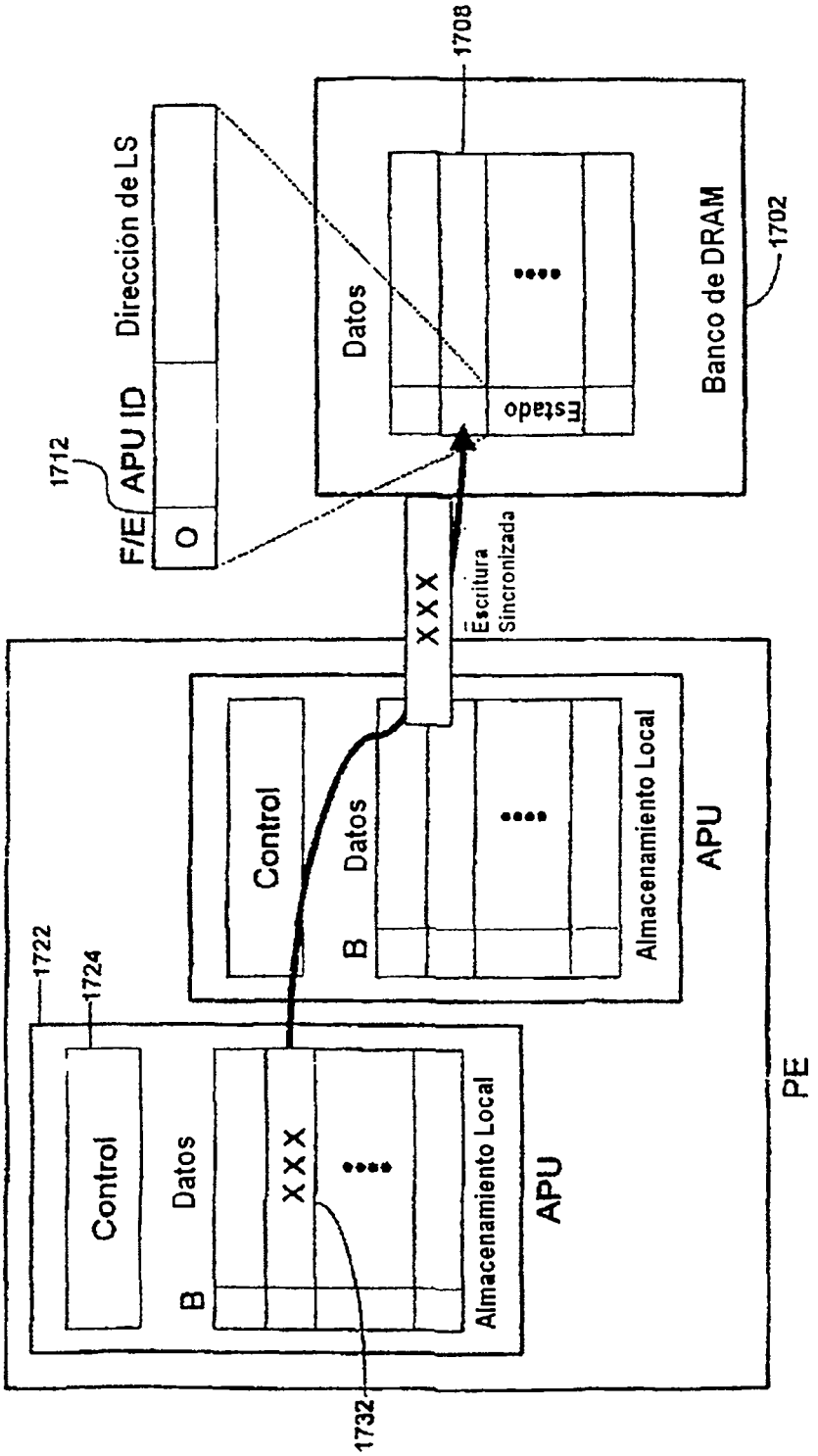
1708
Dirección de LS
APU ID
F/E
O
1712
Datos
Estado
Banco de DRAM
1702
XXX
Escritura Sincronizada
Control
Datos
B
Almacenamiento Local
APU
1722
1724
Control
Datos
B
XXX
Almacenamiento Local
APU
PE
1732

FIG. 17B

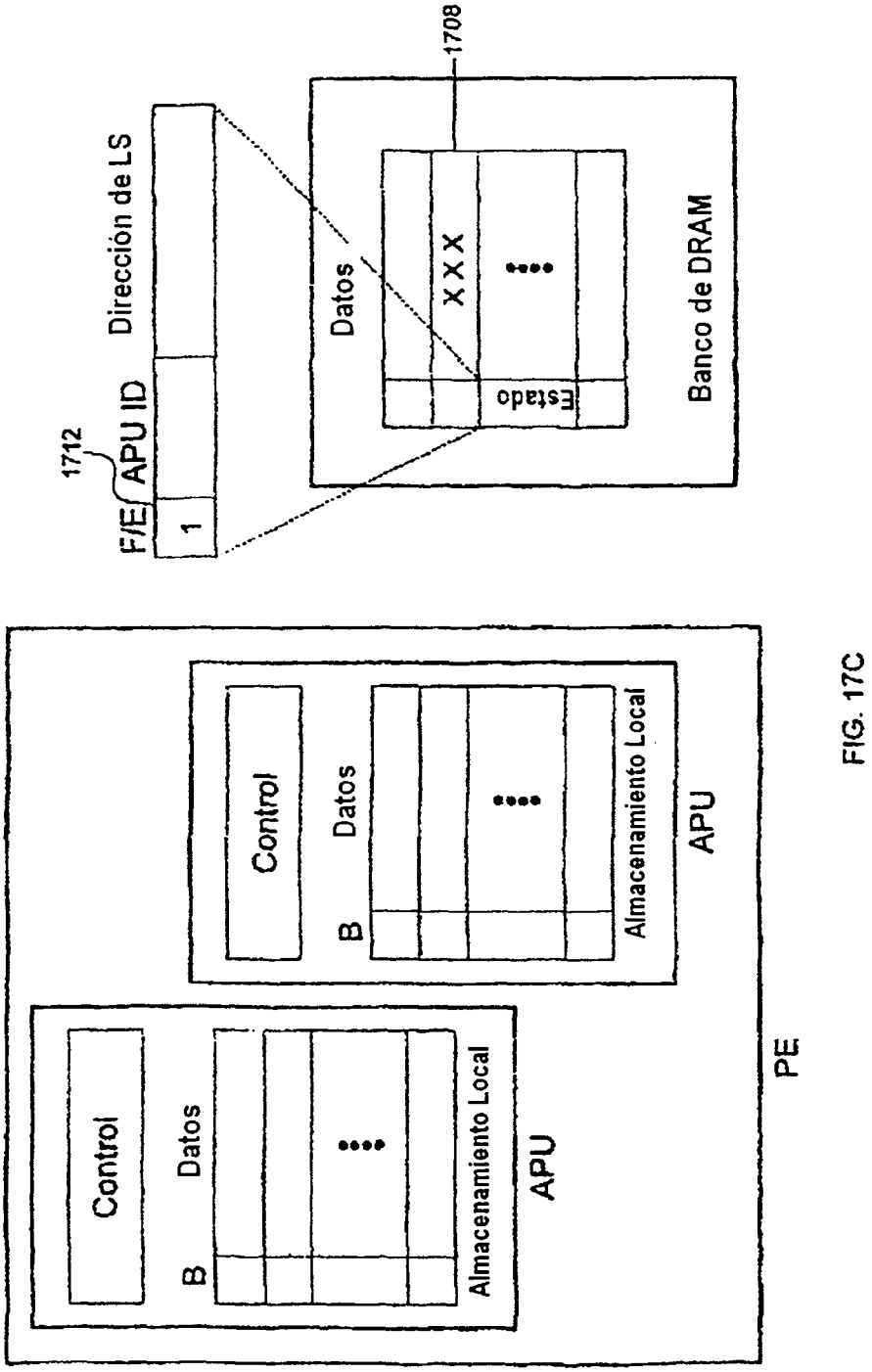
1708
Dirección de LS
1712
F/E
APU ID
1
Datos
XXX
Estado
Banco de DRAM
PE
Control
Datos
B
Almacenamiento Local
APU
Control
Datos
B
Almacenamiento Local
APU

FIG. 17C

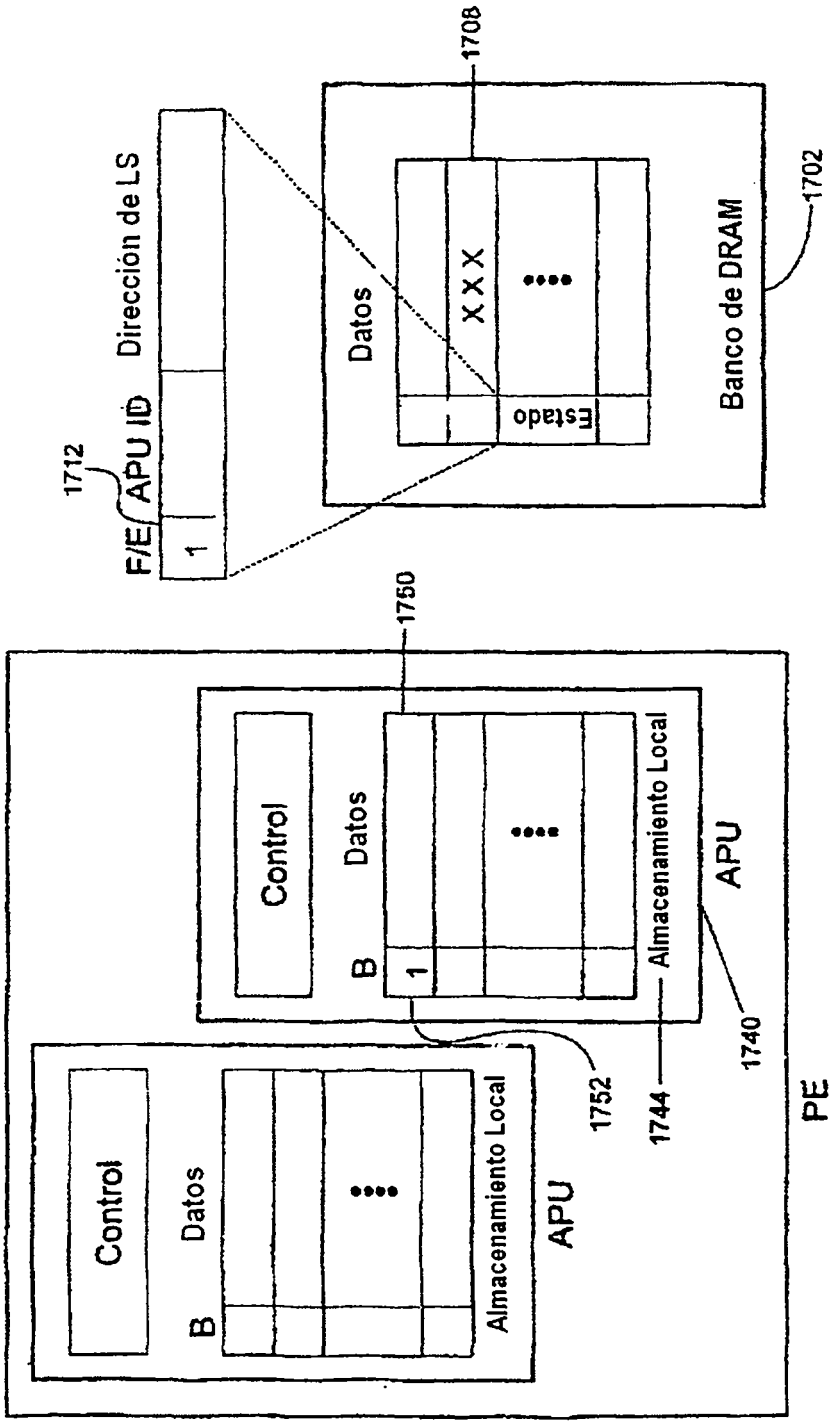
1708
Dirección de LS
F/E
APU ID
1712
1
Datos
XXX
Estado
Banco de DRAM
1702
1750
Control
Datos
B
1
Almacenamiento Local
APU
1740
1752
1744
Control
Datos
B
Almacenamiento Local
APU
PE

FIG. 17D

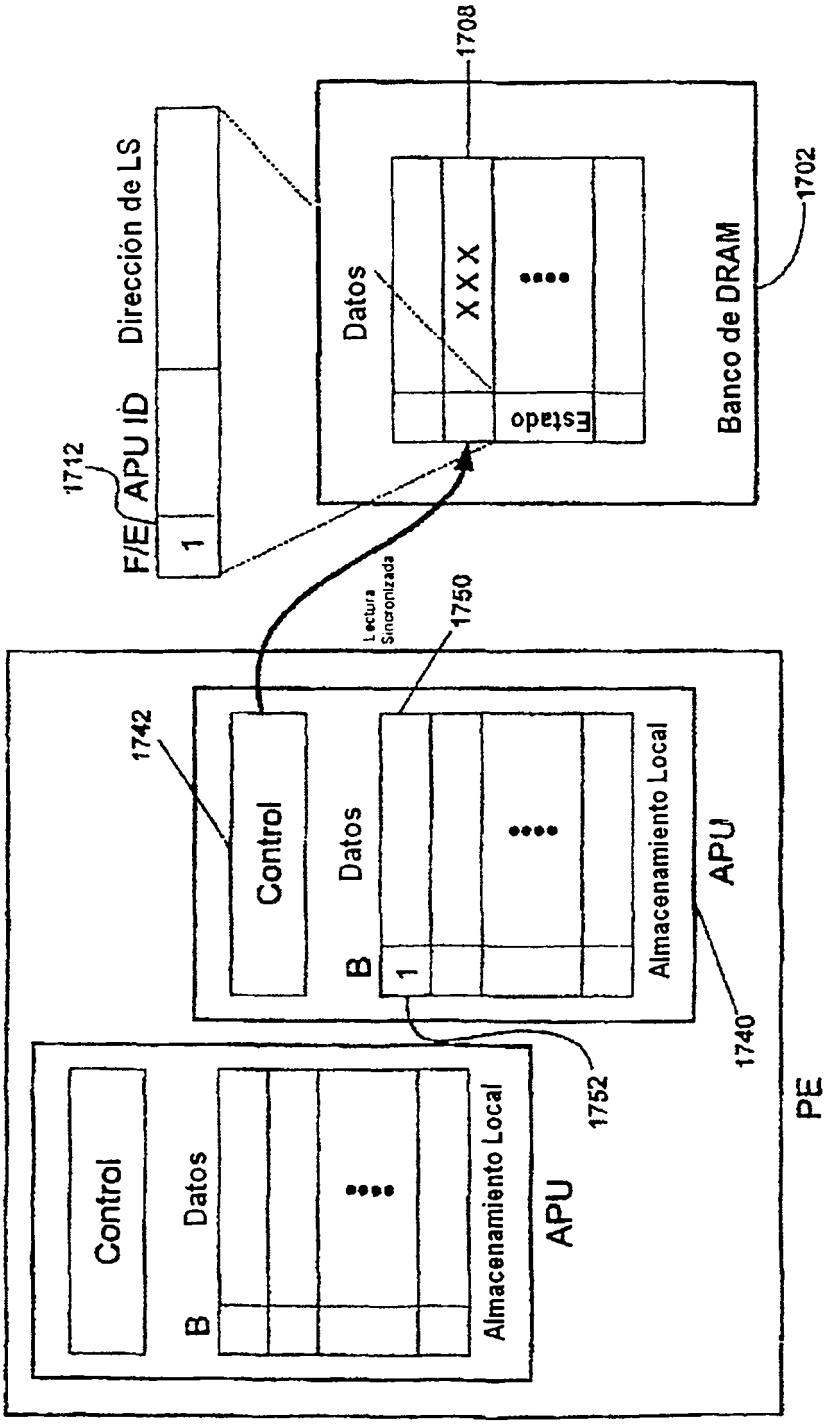
1708
Dirección de LS
APU ID
F/E
1712
1
Datos
XXX
....
Estado
Banco de DRAM
1702
Lectura Sincronizada
1750
1742
Control
Datos
B
1
....
Almacenamiento Local
APU
1740
1752
Control
Datos
B
....
Almacenamiento Local
APU
PE

FIG. 17E

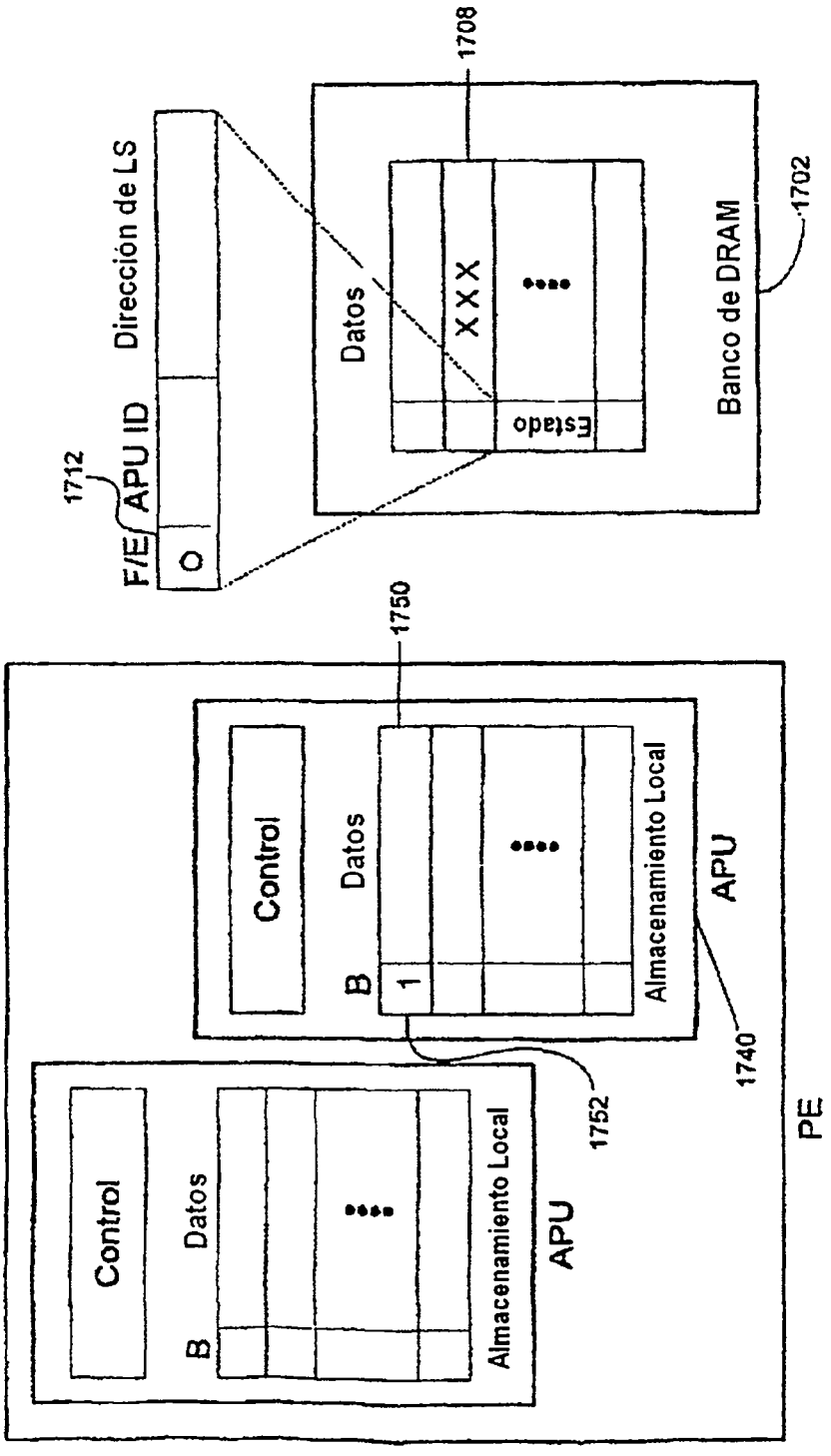
1708
Dirección de LS
F/E
APU ID
O
1712
Datos
XXX
Estado
Banco de DRAM
1702
1750
Control
Datos
B
1
Almacenamiento Local
APU
1740
1752
Control
Datos
B
Almacenamiento Local
APU
PE

FIG. 17F

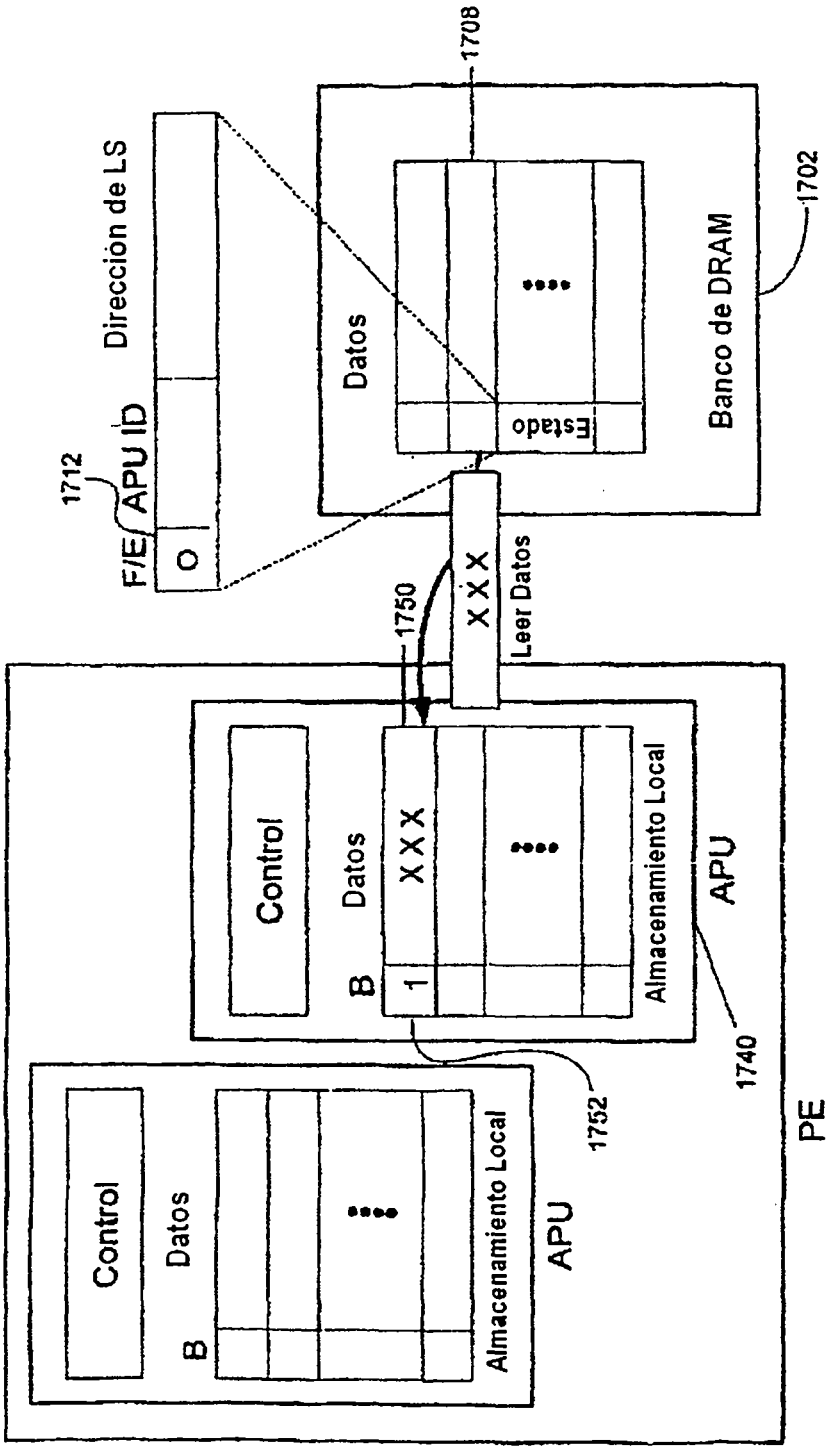
1708
Dirección de LS
Datos
Banco de DRAM
1702
Estado
1712
F/E
APU ID
O
1750
XXX
Leer Datos
Control
Datos
B
1
XXX
Almacenamiento Local
APU
1740
1752
PE

FIG. 17G

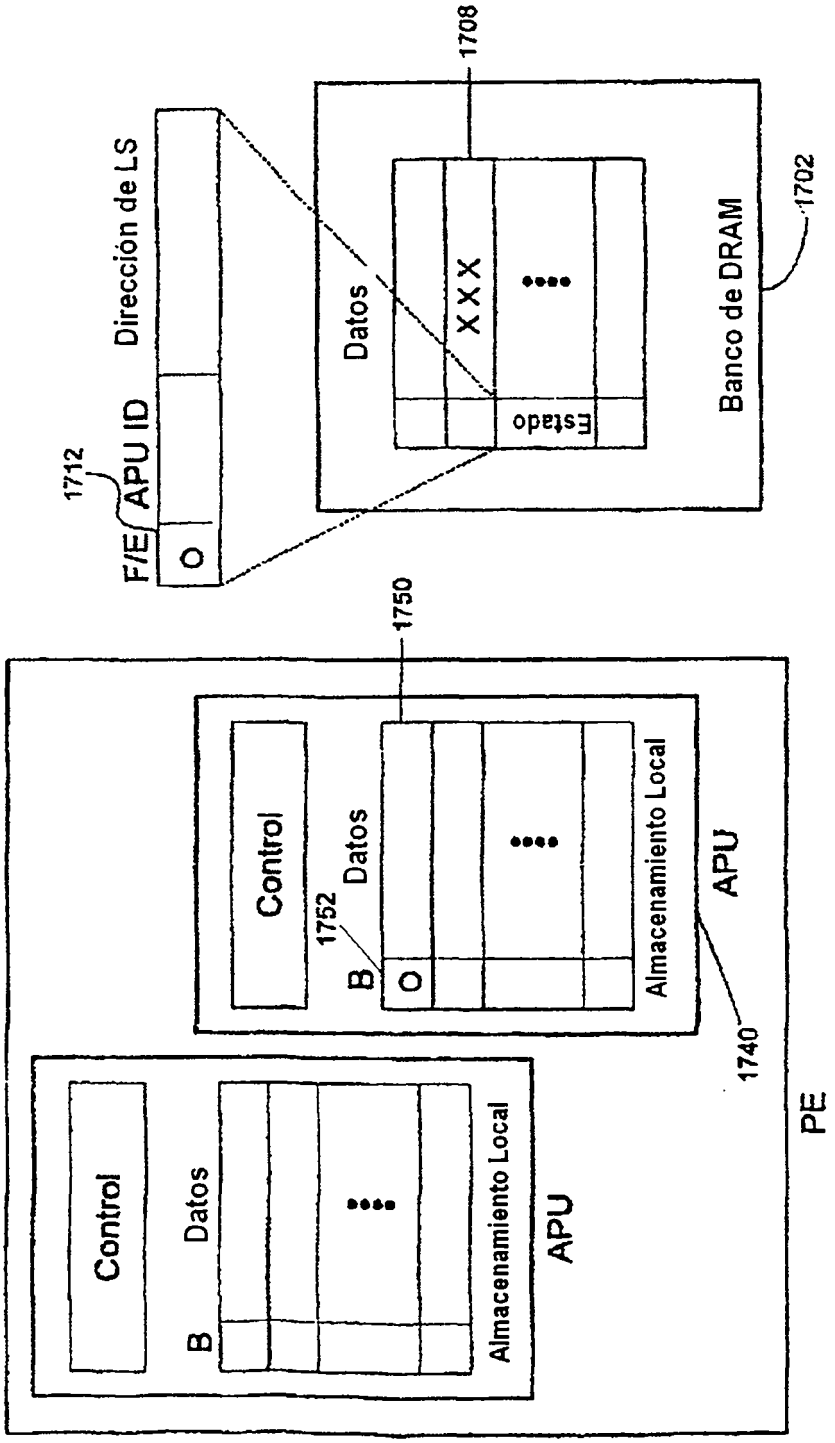
1708
Dirección de LS
APU ID
F/E
1712
O
Datos
XXX
Estado
Banco de DRAM
1702
1750
Control
Datos
1752
B
O
Almacenamiento Local
APU
1740
PE

FIG. 17H

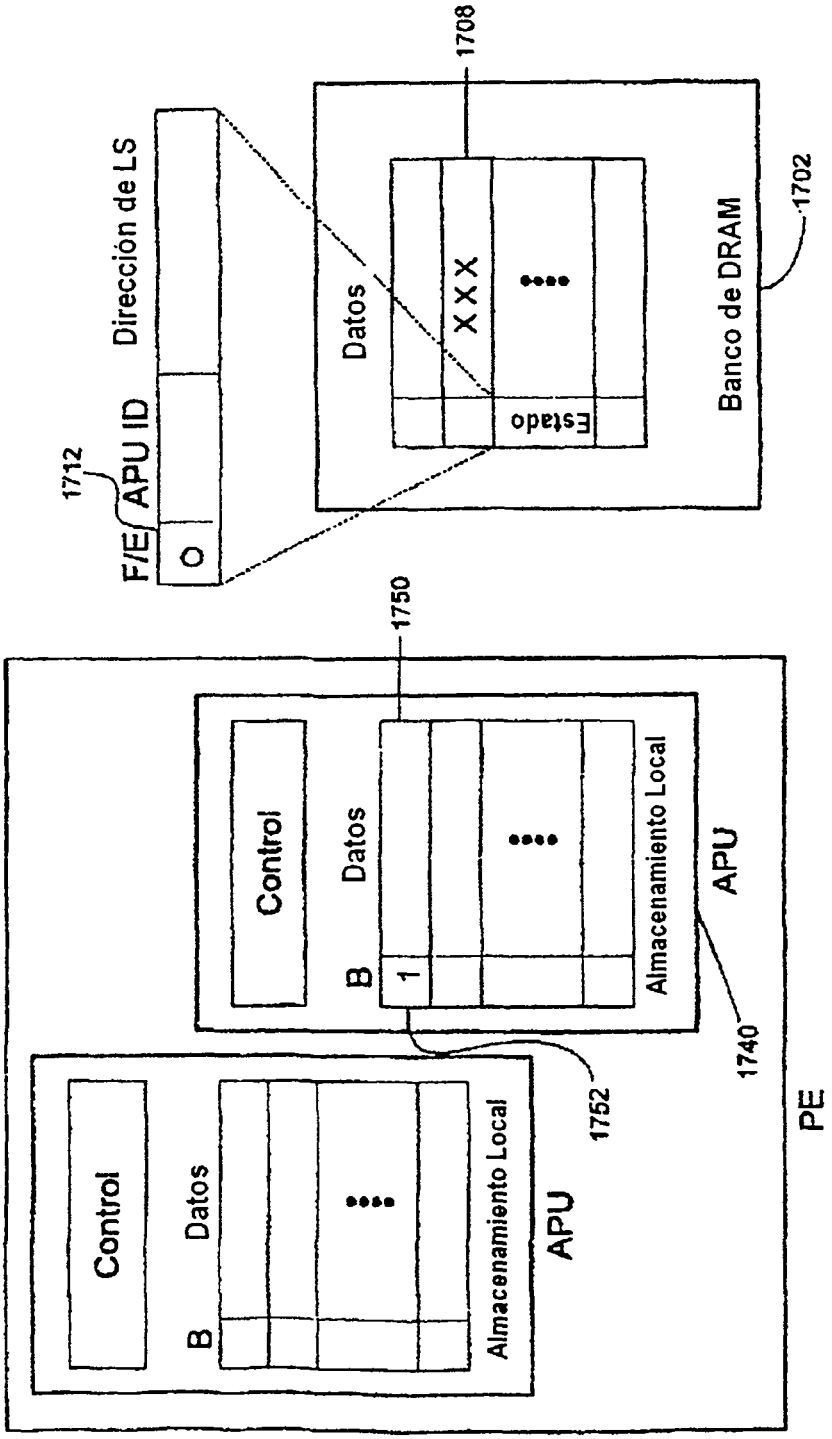
1708
Dirección de LS
F/E
APU ID
1712
O
Datos
XXX
Estado
Banco de DRAM
1702
1750
Control
Datos
B
1
Almacenamiento Local
APU
1740
1752
Control
Datos
B
Almacenamiento Local
APU
PE

FIG. 171

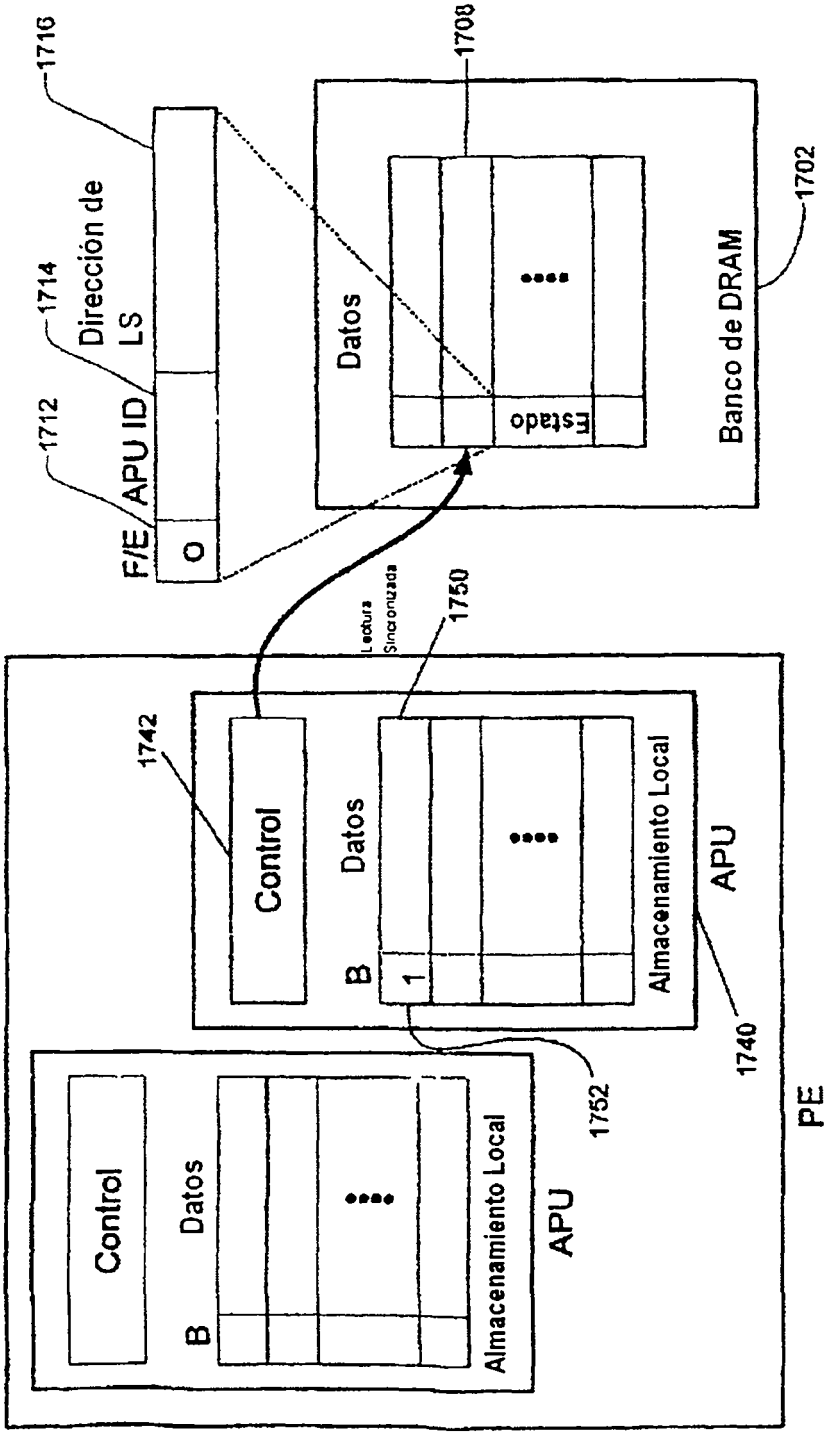
1716
1708
1714
Dirección de LS
1712
APU ID
F/E
O
Datos
Estado
Banco de DRAM
1702
Lectura Sincronizada
1750
1742
Control
Datos
B
1
Almacenamiento Local
APU
1740
1752
PE
Control
Datos
B
Almacenamiento Local
APU

FIG. 17J

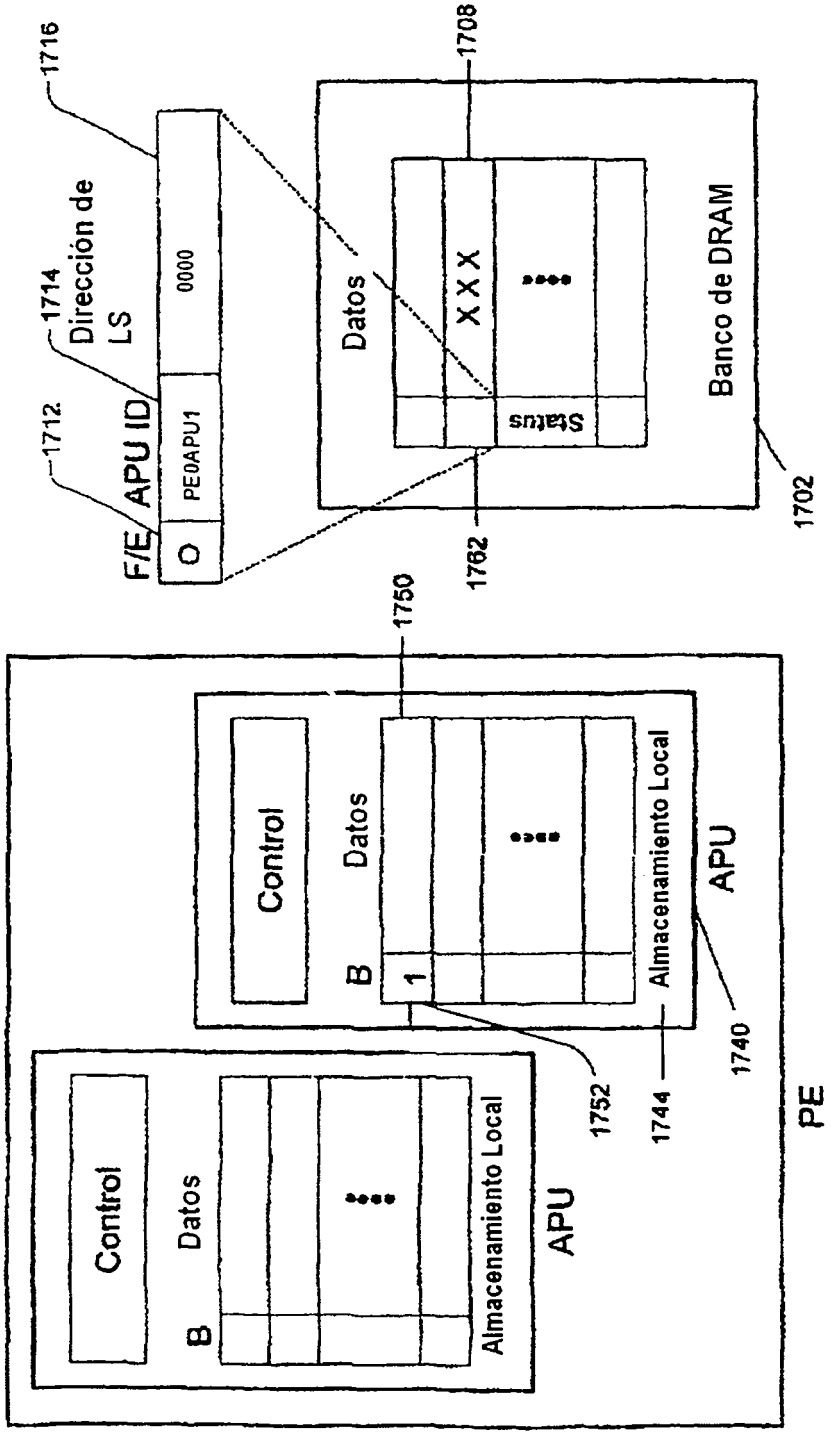
1716
1714
Dirección de LS
1712
APU ID
F/E
O
PE0APU1
0000
1708
Datos
X X X
Status
Banco de DRAM
1702
1762
1750
Control
Datos
B
1
Almacenamiento Local
APU
1740
1752
1744
PE

FIG. 17K

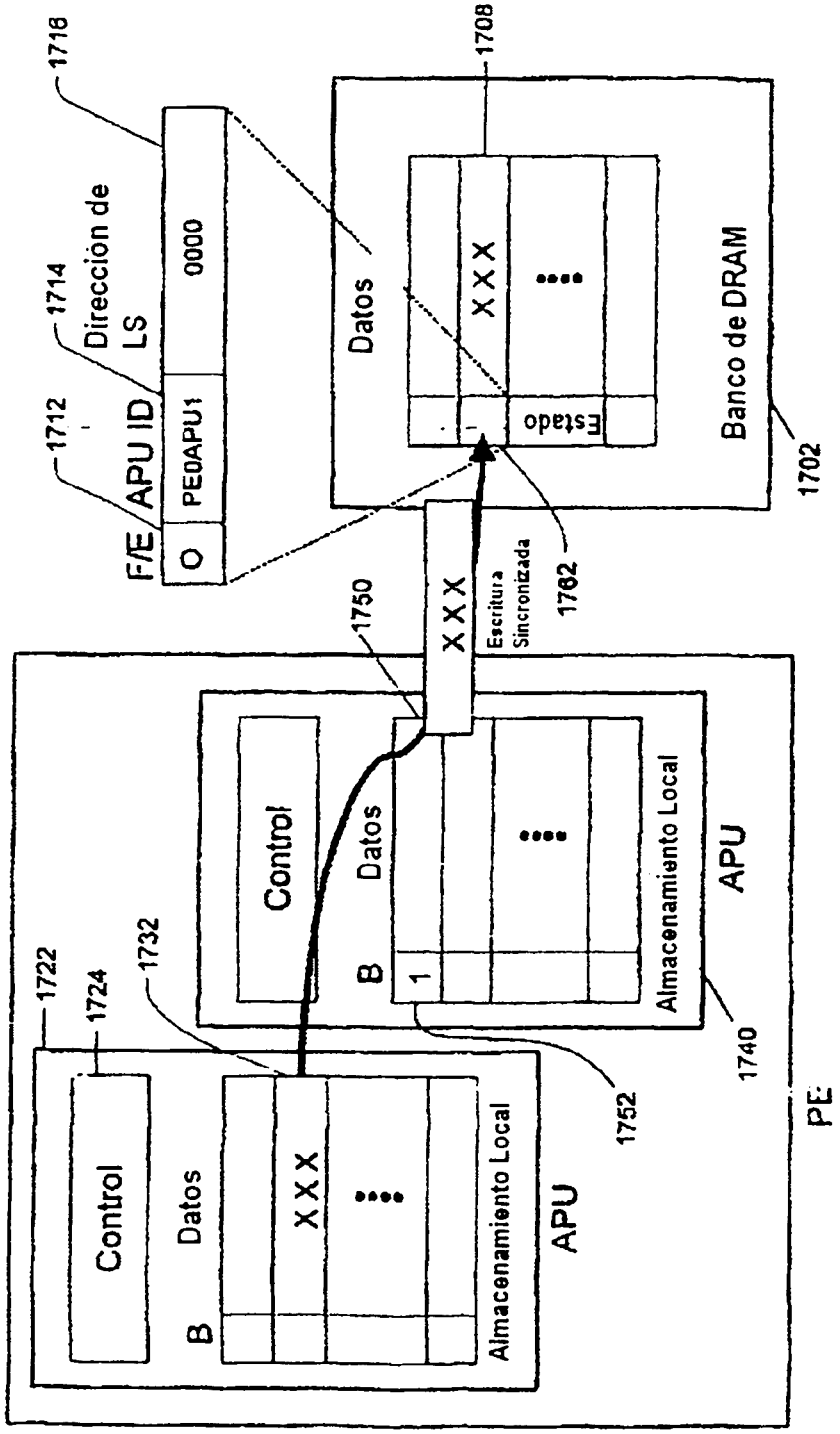
1716
1708
1714
Dirección de LS
Datos
XXX
....
Estado
Banco de DRAM
1702
1712
F/E
APU ID
0
PEDAPU1
0000
1762
Escritura Sincronizada
1750
XXX
Control
Datos
B
1
....
Almacenamiento Local
APU
1740
1722
1724
1732
Control
Datos
B
XXX
....
Almacenamiento Local
APU
1752
PE

FIG. 17L

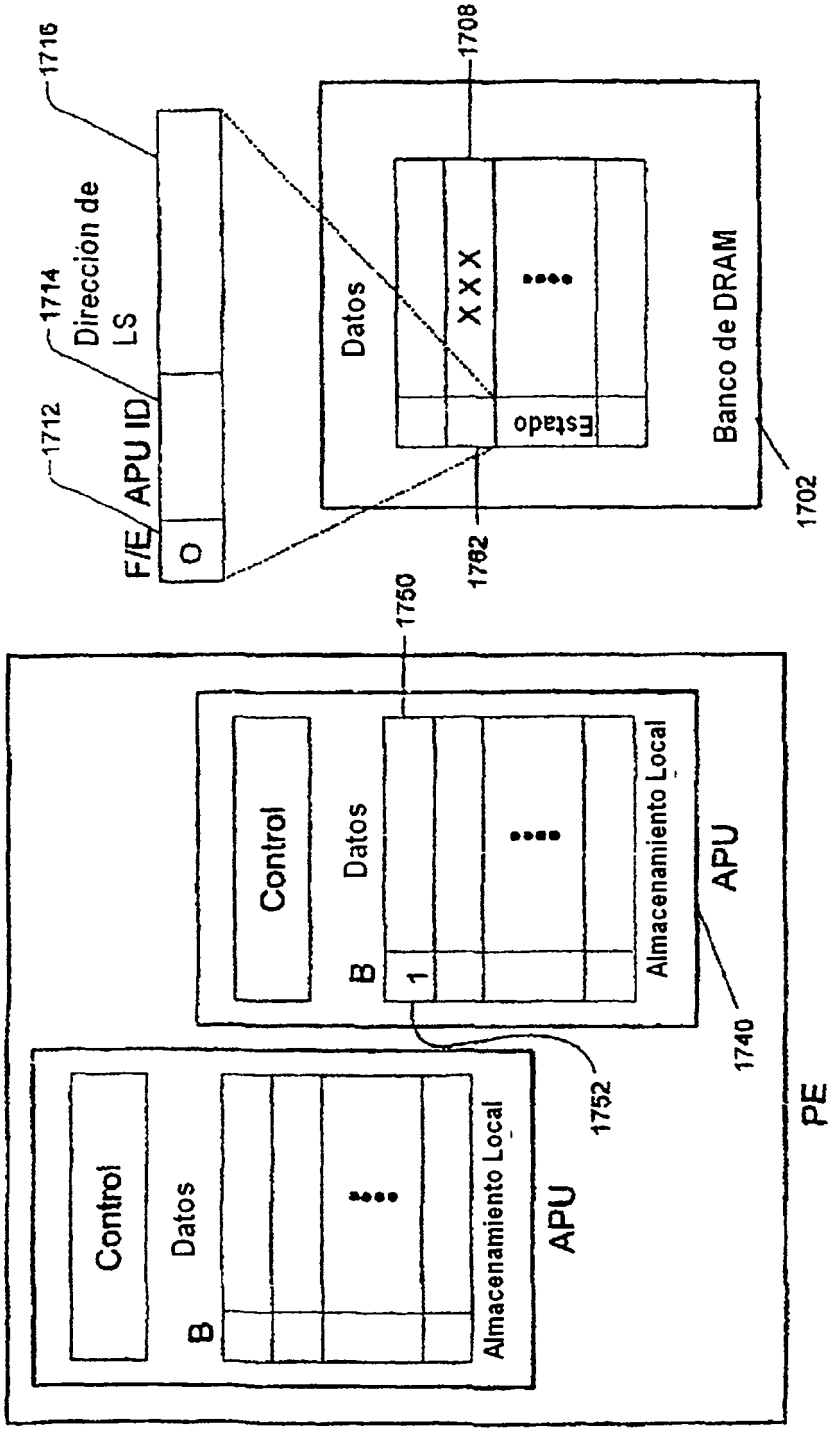
1716
1714
Dirección de LS
1712
APU ID
F/E
0
1708
Datos
XXX
Estado
Banco de DRAM
1702
1782
1750
Control
Datos
B
1
Almacenamiento Local
APU
1740
1752
Control
Datos
B
Almacenamiento Local
APU
PE

FIG. 17M

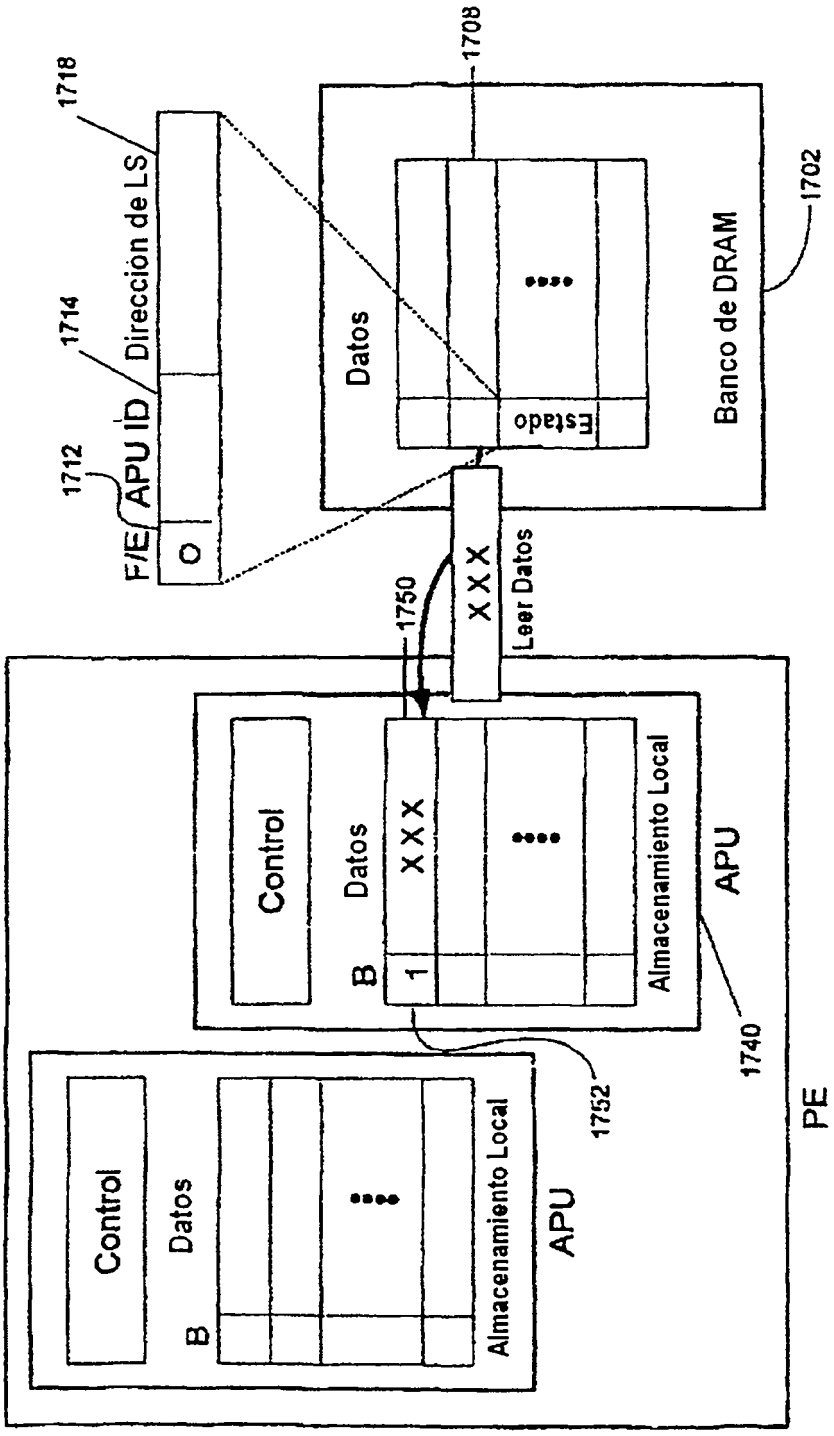
1718
1708
1714
Dirección de LS
1712
APU ID
F/E
O
Datos
Estado
Banco de DRAM
1702
1750
X X X
Leer Datos
Control
Datos
B
1
X X X
Almacenamiento Local
APU
1740
1752
Control
Datos
B
Almacenamiento Local
APU
PE

FIG. 17N

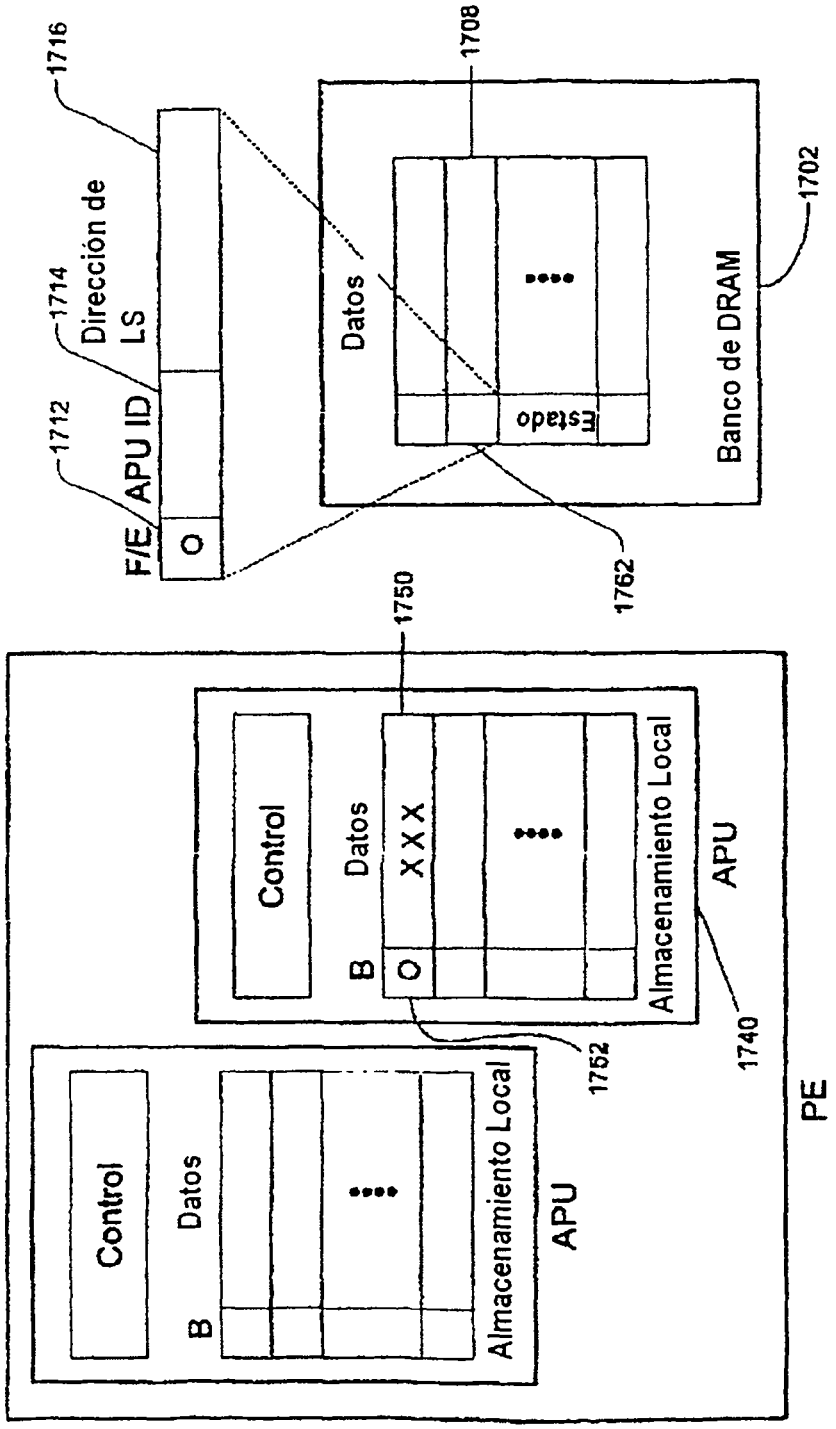
1716
1708
1714
Dirección de LS
1712
APU ID
F/E
O
Datos
Estado
Banco de DRAM
1702
1762
1750
Control
Datos
B
O
XXX
Almacenamiento Local
APU
1740
1752
Control
Datos
B
Almacenamiento Local
APU
PE

FIG. 17O

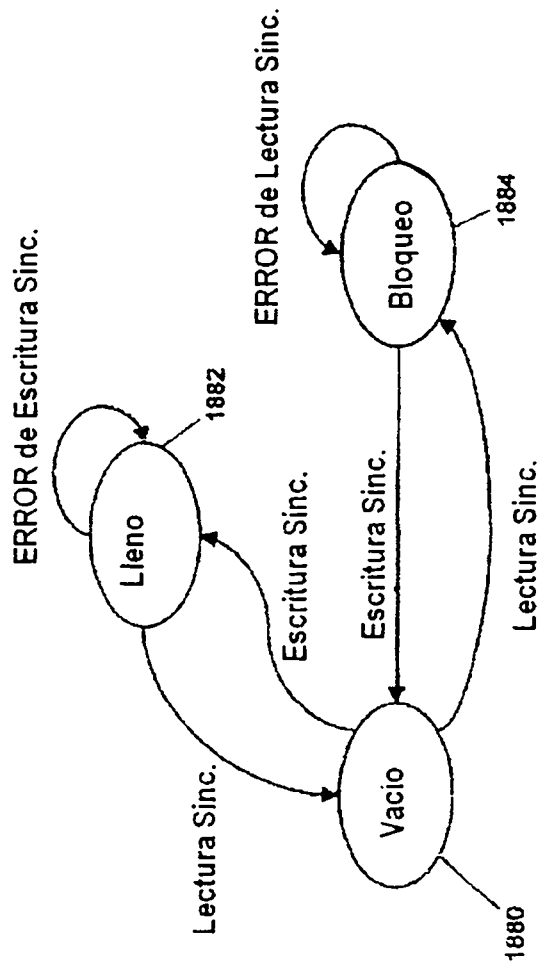
ERROR de Escritura Sinc.
Lleno
1882
ERROR de Lectura Sinc.
Bloqueo
1884
Escritura Sinc.
Escritura Sinc.
Lectura Sinc.
Lectura Sinc.
Vacio
1880

FIG. 18

Tabla de Control de Claves

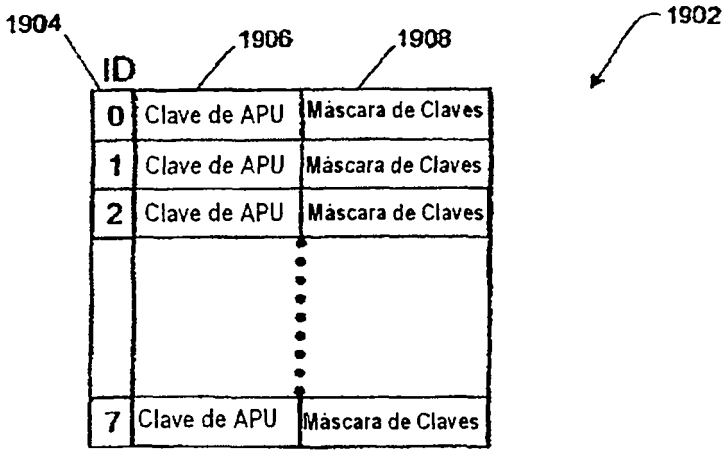

FIG. 19

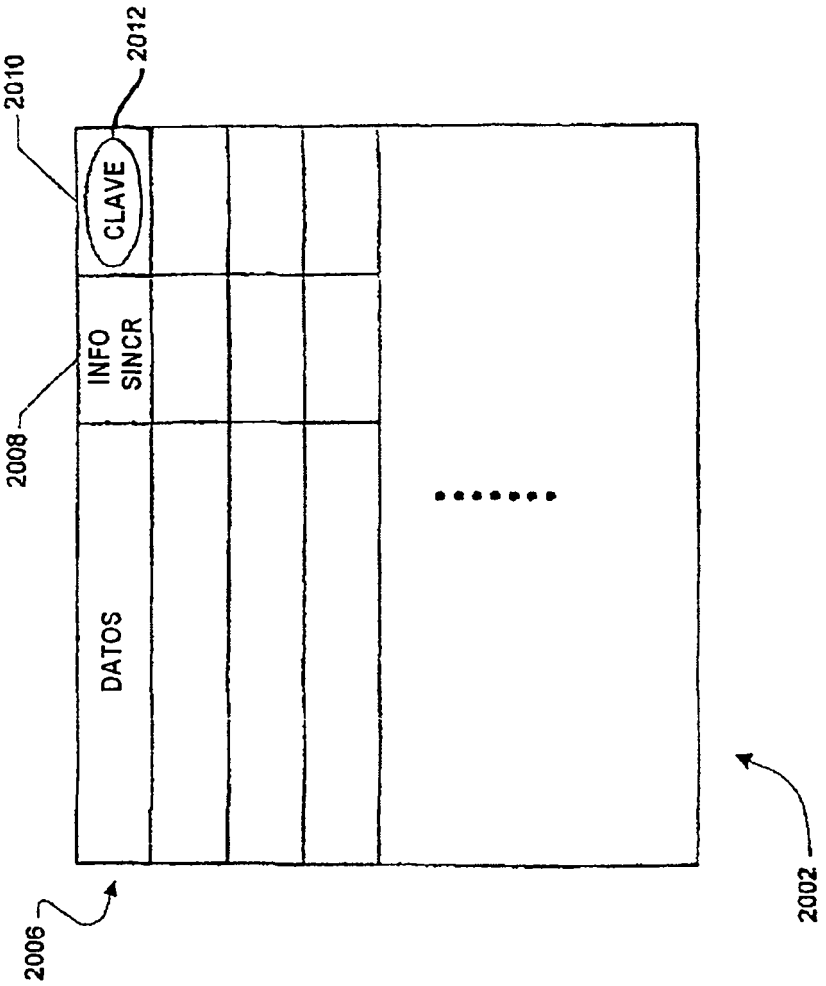

FIG. 20

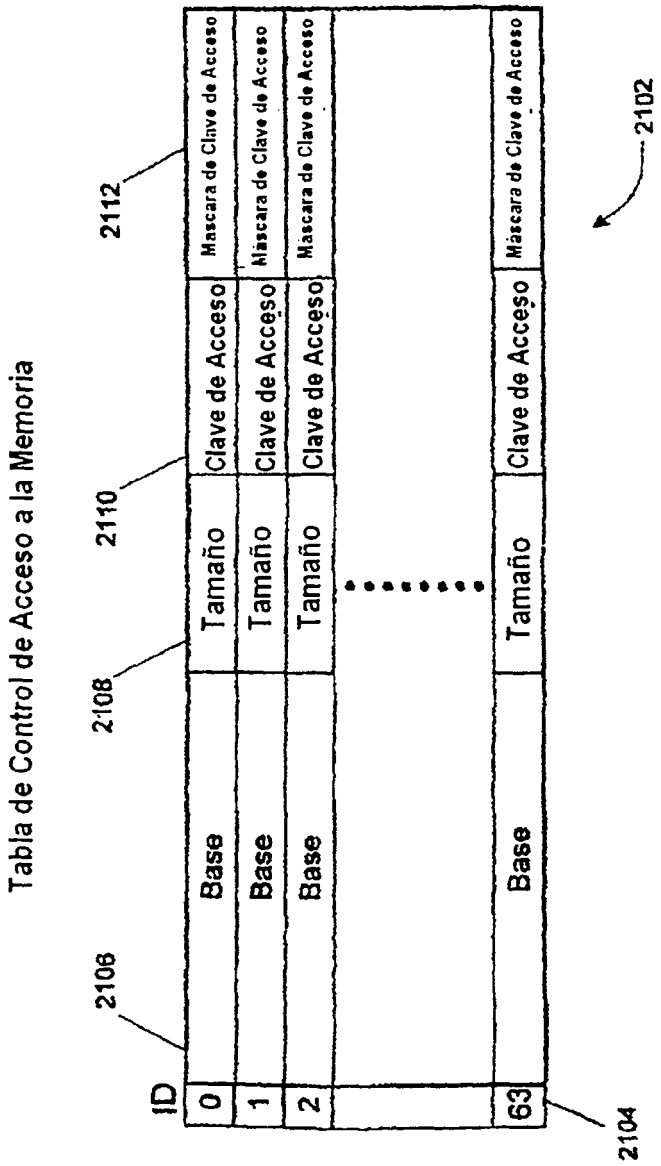
Tabla de Control de Acceso a la Memoria
2106
2108
2110
2112
2102
2104
ID
Base
Tamaño
Clave de Acceso
Mascara de Clave de Acceso
0
1
2
63

FIG. 21

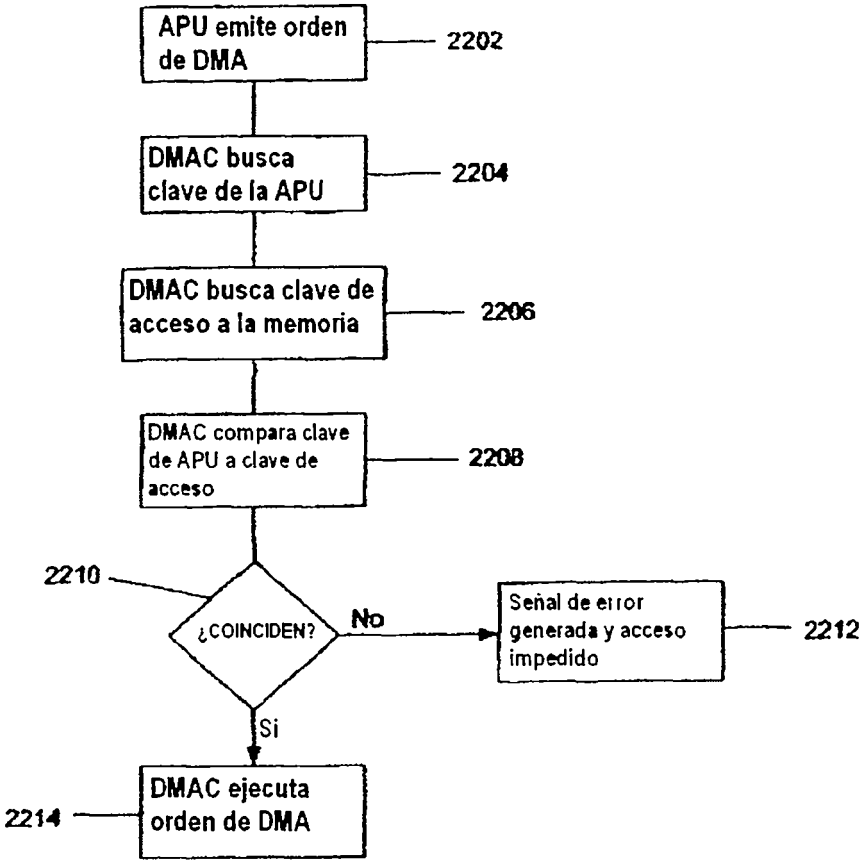
APU emite orden de DMA
2202
DMAC busca clave de la APU
2204
DMAC busca clave de acceso a la memoria
2206
DMAC compara clave de APU a clave de acceso
2208
2210
¿COINCIDEN?
No
Señal de error generada y acceso impedido
2212
Si
DMAC ejecuta orden de DMA
2214

FIG. 22

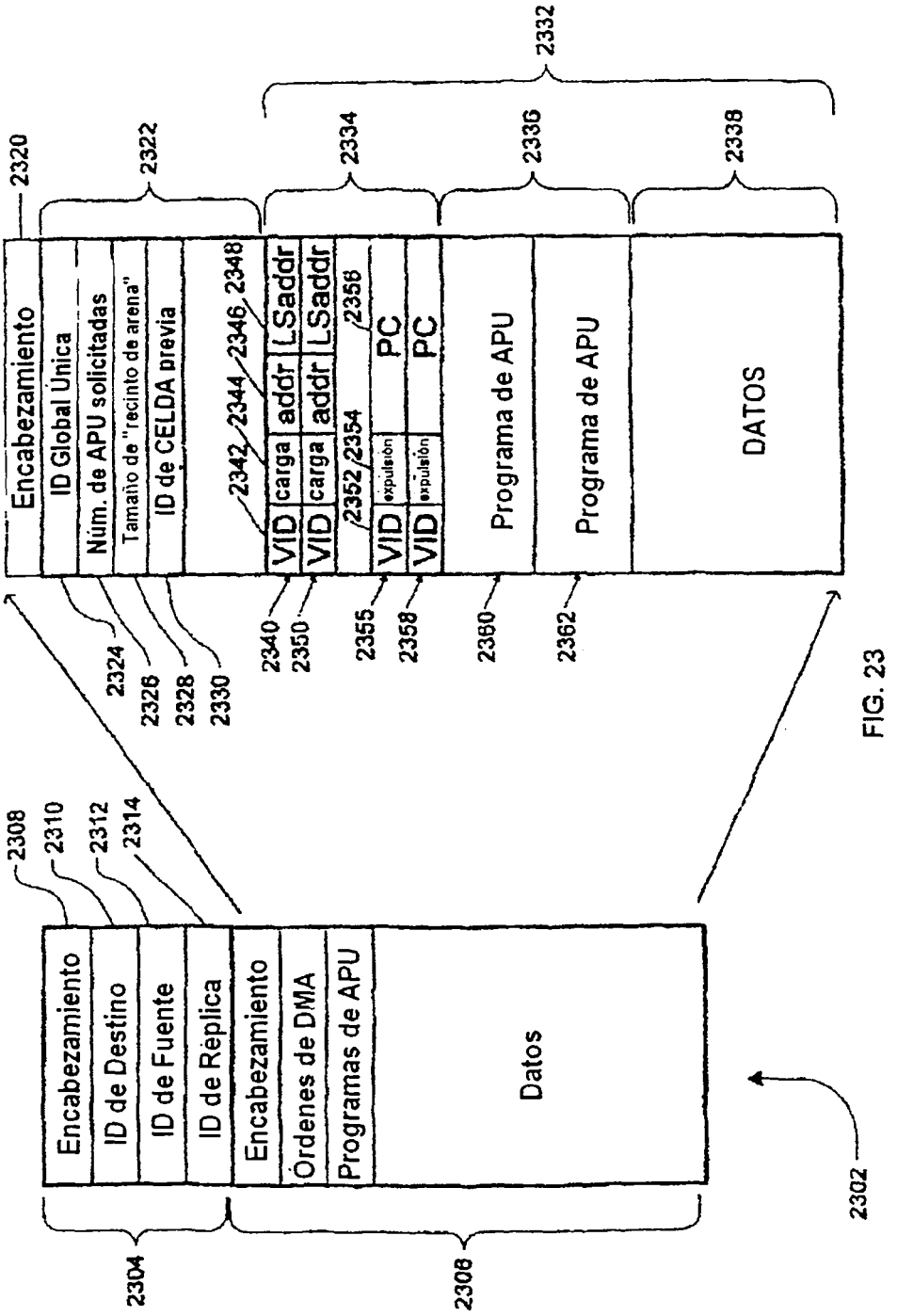
Encabezamiento
ID Global Única
Núm. de APU solicitadas
Tamaño de "recinto de arena"
ID de CELDA previa
VID carga addr LSaddr
VID carga addr LSaddr
VID expulsión PC
VID expulsión PC
Programa de APU
Programa de APU
DATOS
2320
2322
2332
2334
2336
2338
2324
2326
2328
2330
2340
2350
2355
2358
2360
2362
Encabezamiento
ID de Destino
ID de Fuente
ID de Réplica
Encabezamiento
Órdenes de DMA
Programas de APU
Datos
2308
2310
2312
2314
2304
2306
2302

FIG. 23

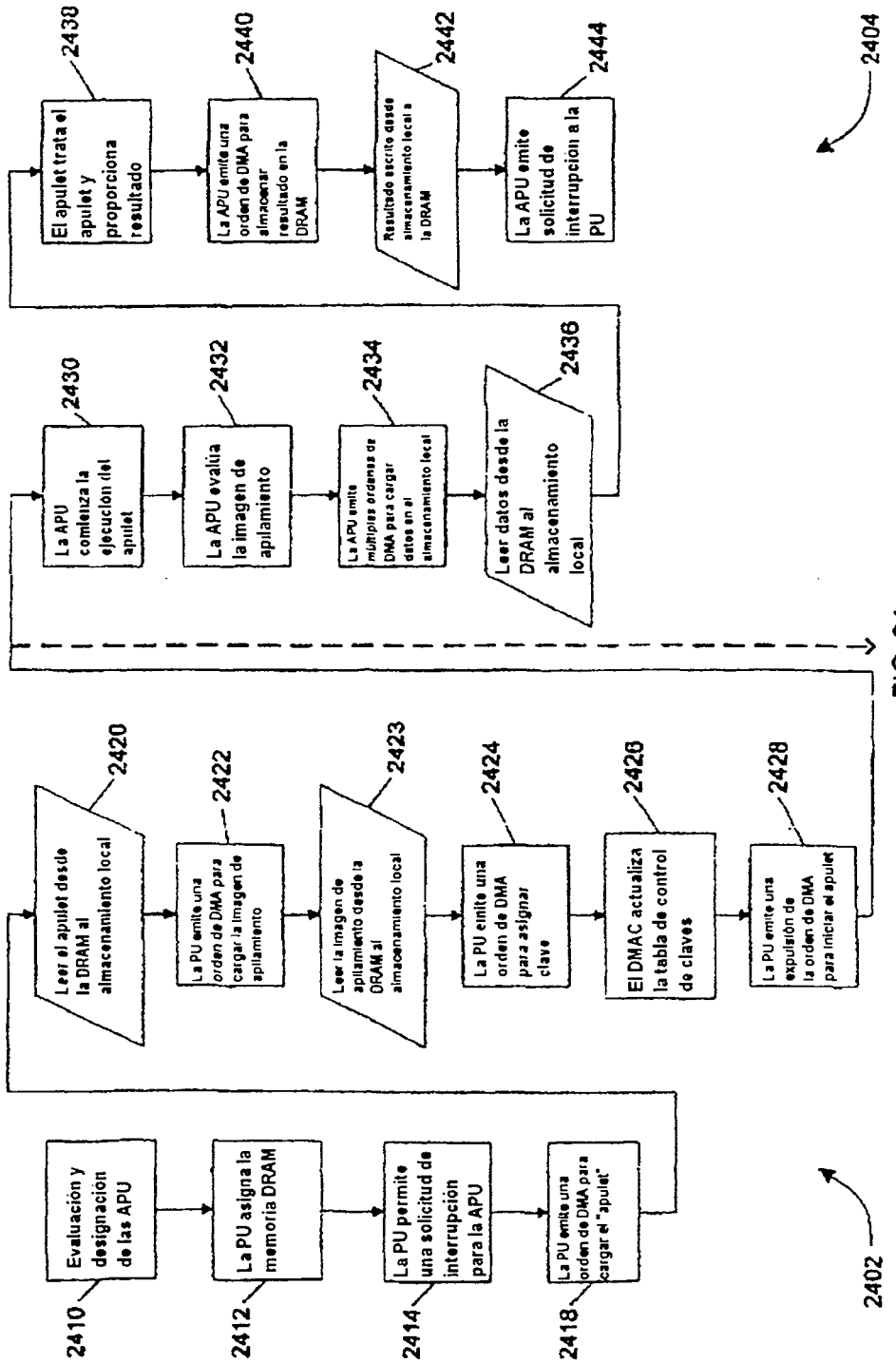
2410
Evaluación y designación de las APU
2412
La PU asigna la memoria DRAM
2414
La PU permite una solicitud de interrupción para la APU
2418
La PU emite una orden de DMA para cargar el "apulet"
2420
Leer el apulet desde la DRAM al almacenamiento local
2422
La PU emite una orden de DMA para cargar la imagen de apilamiento
2423
Leer la imagen de apilamiento desde la DRAM al almacenamiento local
2424
La PU emite una orden de DMA para asignar clave
2426
El DMAC actualiza la tabla de control de claves
2428
La PU emite una expulsión de la orden de DMA para iniciar el apulet
2402
2430
La APU comienza la ejecución del apulet
2432
La APU evalúa la imagen de apilamiento
2434
La APU emite múltiples órdenes de DMA para cargar datos en el almacenamiento local
2436
Leer datos desde la DRAM al almacenamiento local
2438
El apulet trata el apulet y proporciona resultado
2440
La APU emite una orden de DMA para almacenar resultado en la DRAM
2442
Resultado escrito desde almacenamiento local a la DRAM
2444
La APU emite solicitud de interrupción a la PU
2404

FIG. 24

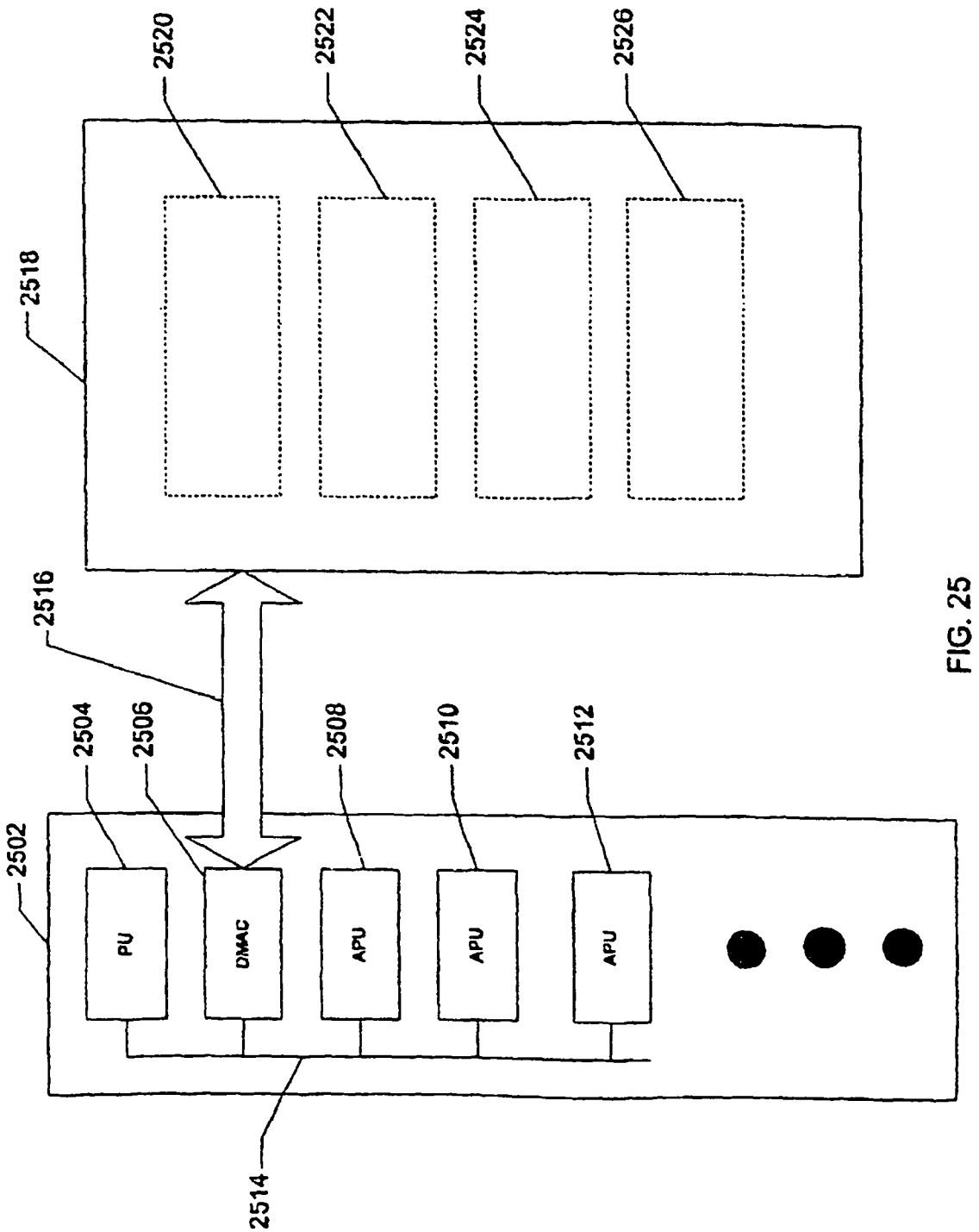
2520
2522
2524
2526
2518
2516
2504
2506
2508
2510
2512
2502
PU
DMAC
APU
APU
APU
2514

FIG. 25

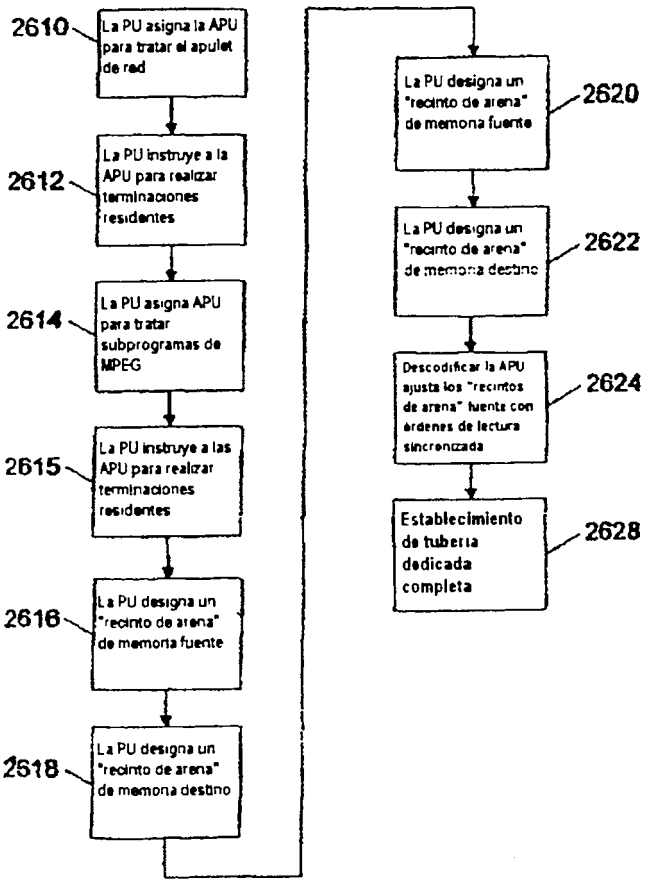
2610
La PU asigna la APU para tratar el apulet de red
2612
La PU instruye a la APU para realizar terminaciones residentes
2614
La PU asigna APU para tratar subprogramas de MPEG
2615
La PU instruye a las APU para realizar terminaciones residentes
2616
La PU designa un "recinto de arena" de memoria fuente
2618
La PU designa un "recinto de arena" de memoria destino
2620
La PU designa un "recinto de arena" de memoria fuente
2622
La PU designa un "recinto de arena" de memoria destino
2624
Descodificar la APU ajusta los "recintos de arena" fuente con órdenes de lectura sincronizada
2628
Establecimiento de tubería dedicada completa

FIG. 26A

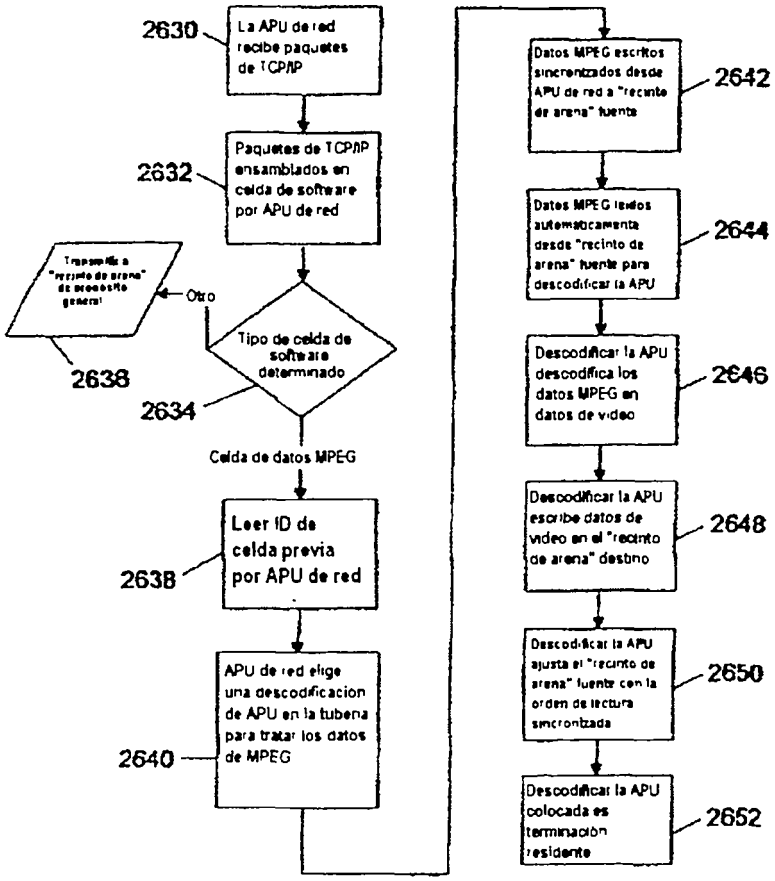
2630
La APU de red recibe paquetes de TCP/IP
2632
Paquetes de TCP/IP ensamblados en celda de software por APU de red
Otro
2638
Tipo de celda de software determinado
2634
Celda de datos MPEG
Leer ID de celda previa por APU de red
2636
APU de red elige una descodificación de APU en la tubería para tratar los datos de MPEG
2640
Datos MPEG escritos sincronizados desde APU de red a "recinto de arena" fuente
2642
Datos MPEG leídos automáticamente desde "recinto de arena" fuente para descodificar la APU
2644
Descodificar la APU descodifica los datos MPEG en datos de vídeo
2646
Descodificar la APU escribe datos de vídeo en el "recinto de arena" destino
2648
Descodificar la APU ajusta el "recinto de arena" fuente con la orden de lectura sincronizada
2650
Descodificar la APU colocada es terminación residente
2652

FIG. 26B

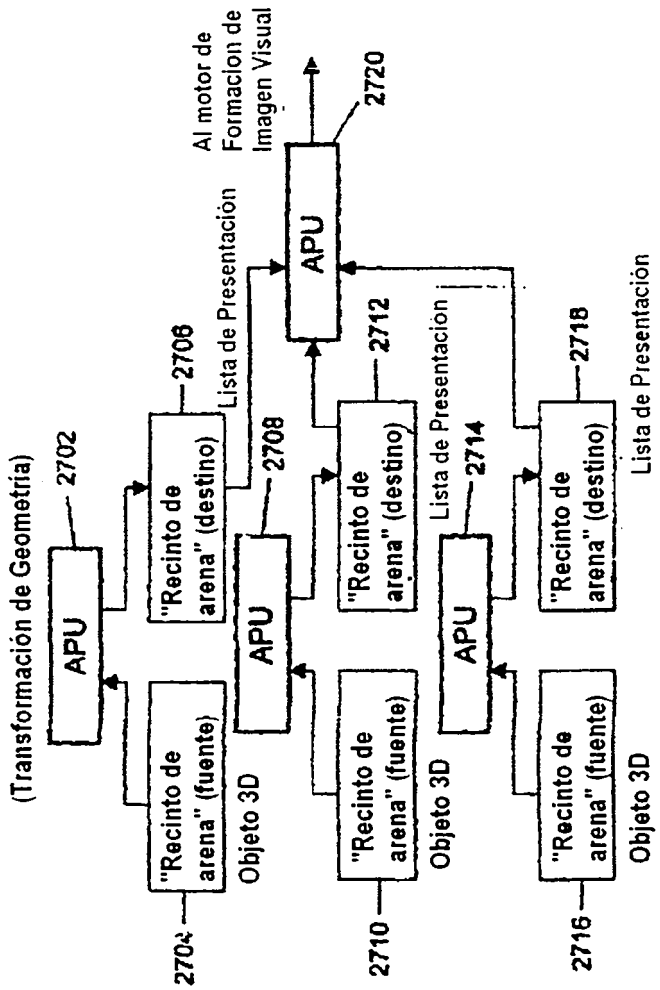
(Transformación de Geometría)
APU
2702
"Recinto de arena" (fuente)
2704
"Recinto de arena" (destino)
2706
Objeto 3D
Lista de Presentación
APU
2708
"Recinto de arena" (fuente)
2710
"Recinto de arena" (destino)
2712
Objeto 3D
Lista de Presentación
APU
2714
"Recinto de arena" (fuente)
2716
"Recinto de arena" (destino)
2718
Objeto 3D
Lista de Presentación
APU
2720
Al motor de Formacion de Imagen Visual

FIG. 27

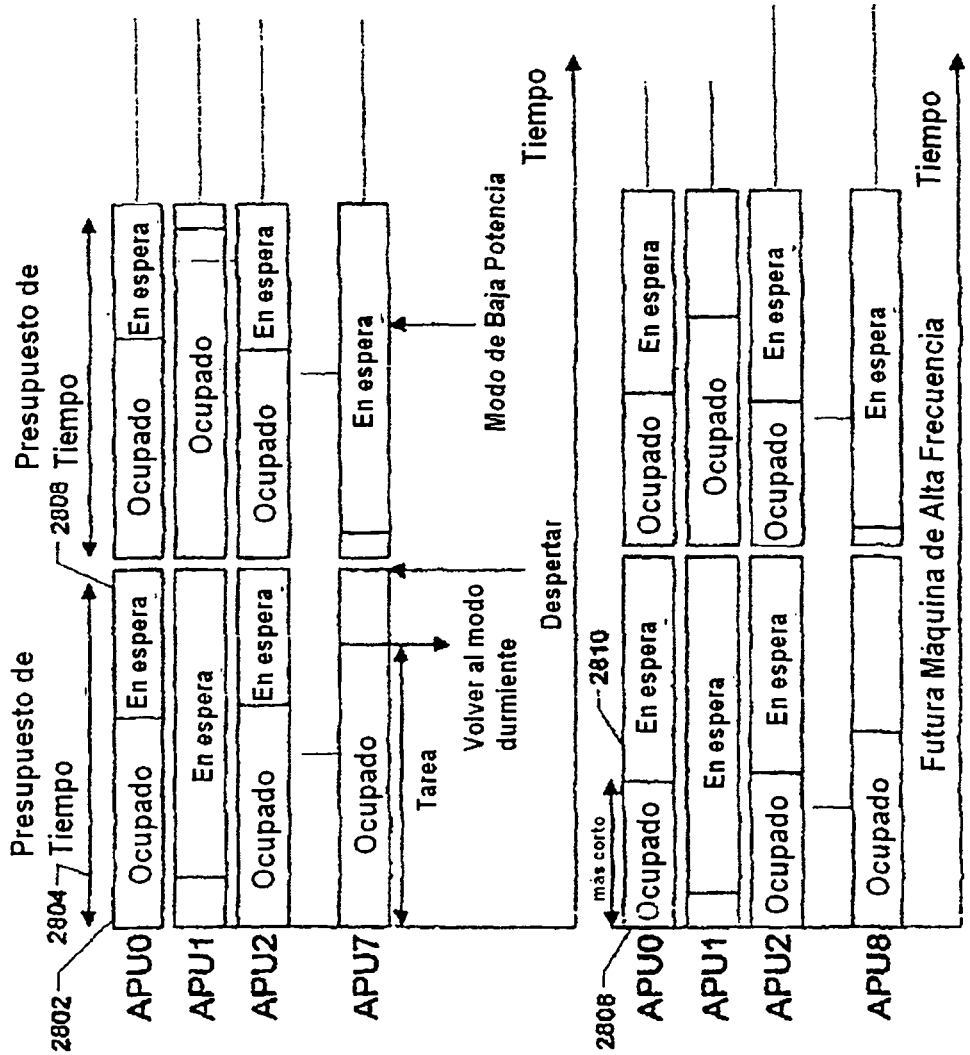
Presupuesto de Tiempo
2804
2802
2808
Presupuesto de Tiempo
APU0
APU1
APU2
APU7
Ocupado
En espera
Tarea
Volver al modo durmiente
Modo de Baja Potencia
Despertar
Tiempo
2810
más corto
2808
APU0
APU1
APU2
APU8
Futura Máquina de Alta Frecuencia
Tiempo

FIG. 28