(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 985 693**

(51) Int. Cl.:

G06T 1/20 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **02.03.2018 E 21181091 (6)**

(97) Fecha y número de publicación de la concesión europea: **21.02.2024 EP 3955203**

(54) Título: **Coordinación y mayor utilización de procesadores de gráficos durante una inferencia**

(30) Prioridad:

24.04.2017 US 201715495054

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
07.11.2024

(73) Titular/es:

INTEL CORPORATION (100.0%)
2200 Mission College Blvd.
Santa Clara, CA 95054, US

(72) Inventor/es:

APPU, ABHISHEK R.;
KOKER, ALTUG;
WEAST, JOHN C.;
MACPHERSON, MIKE B.;
HURD, LINDA L.;
BAGHSORKHI, SARA S.;
GOTTSCHLICH, JUSTIN E.;
SURTI, PRASOONKUMAR;
SAKTHIVEL, CHANDRASEKARAN;
MA, LIWEI;
OULD-AHMED-VALL, ELMOUSTAPHA;
SINHA, KAMAL;
RAY, JOYDEEP;
VEMBU, BALAJI;
JAHAGIRDAR, SANJEEV;
RANGANATHAN, VASANTH y
KIM, DUKHWAN

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Coordinación y mayor utilización de procesadores de gráficos durante una inferencia

CAMPO

Las realizaciones descritas en el presente documento se refieren en general al procesamiento de datos y, más particularmente, a facilitar una herramienta para facilitar la coordinación y una mayor utilización de los procesadores de gráficos durante una inferencia.

ANTECEDENTES

El procesamiento paralelo actual de datos gráficos incluye sistemas y métodos desarrollados para realizar operaciones específicas sobre datos gráficos, tales como, por ejemplo, interpolación lineal, teselación, rasterización, mapeo de textura, prueba de profundidad, etc. Tradicionalmente, los procesadores gráficos usan unidades computacionales de función fija para procesar datos gráficos; sin embargo, más recientemente, porciones de los procesadores gráficos se han hecho programables, lo que permite que tales procesadores soporten una gama más amplia de operaciones para procesar datos de vértice y de fragmento.

Para aumentar adicionalmente el rendimiento, los procesadores de gráficos típicamente implementan técnicas de procesamiento, tales como canalizaciones, que intentan procesar, en paralelo, tantos datos de gráficos como sea posible a lo largo de todas las diferentes partes de la canalización de gráficos. Los procesadores de gráficos paralelos con arquitecturas de múltiples hilos y única instrucción (SIMT) se diseñan para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura de SIMT, grupos de hilos paralelos intentan ejecutar instrucciones de programa de manera síncrona conjuntamente tan a menudo como sea posible para aumentar la eficiencia de procesamiento. Puede encontrarse una vista global general del software y hardware para arquitecturas SIMT en Shane Cook, CUDA Programming, capítulo 3, páginas 37-51 (2013) y/o Nicholas Wilt, CUDA Handbook, A Comprehensive Guide to GPU Programming, secciones 2.6.2 a 3.1.2 (junio de 2013).

El aprendizaje automático ha tenido éxito en la resolución de muchos tipos de tareas. Los cálculos que surgen cuando se entrenan y usan algoritmos de aprendizaje automático (por ejemplo, redes neuronales) se prestan naturalmente a implementaciones paralelas eficientes. En consecuencia, los procesadores paralelos, tales como las unidades de procesamiento de gráficos de propósito general (GPGPU), han desempeñado un papel importante en la implementación práctica de redes neuronales profundas. Los procesadores de gráficos paralelos con arquitecturas de múltiples hilos y única instrucción (SIMT) se diseñan para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura de SIMT, grupos de hilos paralelos intentan ejecutar instrucciones de programa de manera síncrona conjuntamente tan a menudo como sea posible para aumentar la eficiencia de procesamiento. La eficiencia proporcionada por las implementaciones paralelas de algoritmos de aprendizaje automático permite el uso de redes de alta capacidad y permite que esas redes se entrenen en conjuntos de datos más grandes.

Las técnicas convencionales no proporcionan la coordinación entre la salida de inferencia y los sensores responsables de proporcionar entradas; sin embargo, tales técnicas convencionales no proporcionan precisión en el resultado de la inferencia. Además, el uso de la inferencia a través de un procesador gráfico es bastante ligero, mientras que el resto del procesador gráfico permanece sin utilizar.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Las realizaciones se ilustran a modo de ejemplo, y no a modo de limitación, en las figuras de los dibujos adjuntos en los que números de referencia similares se refieren a elementos similares. Para que puedan entenderse en detalle las características antes citadas, se proporciona una descripción más particular, resumida anteriormente de manera breve, haciendo referencia a las realizaciones, algunas de las cuales se ilustran en los dibujos adjuntos. Sin embargo, cabe señalar que los dibujos adjuntos ilustran únicamente realizaciones típicas y, por lo tanto, no deben considerarse limitativos de su alcance, ya que los dibujos pueden ilustrar otras realizaciones igualmente eficaces.

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático configurado para implementar uno o más aspectos descritos en el presente documento.

Las **Figuras 2A-2D** ilustran componentes de procesador paralelo.

Las **Figuras 3A-3B** son diagramas de bloques de multiprocesadores de gráficos.

Las **Figuras 4A-4F** ilustran una arquitectura ilustrativa en la que una pluralidad de unidades de procesamiento de gráficos están acopladas de manera comunicativa a una pluralidad de procesadores de múltiples núcleos.

La **Figura 5** ilustra una canalización de procesamiento de gráficos.

La **Figura 6** ilustra un dispositivo informático que aloja un mecanismo de coordinación de inferencia y utilización de procesamiento.

La **Figura 7** ilustra un mecanismo de coordinación de inferencia y utilización de procesamiento.

La **Figura 8A** ilustra una estructura de transacción en procesadores de aplicaciones y/o gráficos para facilitar el entrenamiento previamente analizado.

La **Figura 8B** ilustra un procesador de gráficos para una mejor utilización de procesamiento.

La **Figura 8C** ilustra una secuencia de transacciones para mejorar la coordinación de salidas y sensores de inferencia.

La **Figura 8D** ilustra una secuencia de transacciones para mejorar la coordinación de salidas y sensores de inferencia.

Las **Figuras 9A** y **9B** ilustran secuencias de transacciones que muestran modelos de uso de acuerdo con una realización.

La **Figura 9C** ilustra un gráfico que muestra las opciones de priorización.

La **Figura 10** ilustra una pila de software de aprendizaje automático.

La **Figura 11** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela.

La **Figura 12** ilustra un sistema informático de múltiples GPU.

Las **Figuras 13A-13B** ilustran capas de redes neuronales profundas ilustrativas.

La **Figura 14** ilustra el entrenamiento y despliegue de una red neuronal profunda.

La **Figura 15** ilustra el entrenamiento y despliegue de una red neuronal profunda

La **Figura 16** es un diagrama de bloques que ilustra el aprendizaje distribuido.

La **Figura 17** ilustra un sistema de inferencia ilustrativo en un chip (SOC) adecuado para realizar inferencias utilizando un modelo entrenado.

La **Figura 18** es un diagrama de bloques de un sistema informático con un procesador que tiene uno o más núcleos de procesador y procesadores de gráficos.

La **Figura 19** es un diagrama de bloques de un procesador que tiene uno o más núcleos de procesador, un controlador de memoria integrado y un procesador de gráficos integrado.

La **Figura 20** es un diagrama de bloques de un procesador de gráficos, que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento.

La **Figura 21** es un diagrama de bloques de un motor de procesamiento de gráficos de un procesador de gráficos.

La **Figura 22** es un diagrama de bloques de otro procesador de gráficos.

La **Figura 23** es un diagrama de bloques de la lógica de ejecución de hilos que incluye una matriz de elementos de procesamiento.

La **Figura 24** ilustra un formato de instrucción de unidad de ejecución de procesador de gráficos.

La **Figura 25** es un diagrama de bloques de otro procesador de gráficos que incluye una canalización de gráficos, una canalización de medios, un motor de visualización, una lógica de ejecución de hilos y una canalización de salida de representación.

La **Figura 26A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos.

La **Figura 26B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos.

La **Figura 27** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos.

La **Figura 28** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP que se puede usar para fabricar un circuito integrado para realizar operaciones.

La **Figura 29** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ilustrativo que puede fabricarse usando uno o más núcleos de IP.

La **Figura 30** es un diagrama de bloques que ilustra un procesador de gráficos ilustrativo de un sistema en circuito integrado de chip.

La **Figura 31** es un diagrama de bloques que ilustra un procesador de gráficos ilustrativo adicional de un sistema en un circuito integrado de chip.

DESCRIPCIÓN DETALLADA

Las realizaciones proporcionan una técnica novedosa para facilitar la detección de valores de datos usados con frecuencia y, a continuación, acelerar las operaciones mediante el uso de una o más técnicas, tales como tablas de búsqueda, matemáticas reducidas, etc. Las realizaciones proporcionan además una técnica novedosa para introducir una máquina de estados finitos, donde, en una realización, esta máquina de estados finitos proporciona un puntero a una dirección base para A, B, mientras que la salida es una secuencia de C+.

Cabe señalar que, términos o acrónimos como "red neuronal convolucional", "CNN", "red neuronal", "NN", "red neuronal profunda", "DNN", "red neuronal recurrente", "RNN" y /o similares pueden ser referenciados de manera intercambiable a lo largo de todo este documento. Además, expresiones como "máquina autónoma" o simplemente "máquina", "vehículo autónomo" o simplemente "vehículo", "agente autónomo" o simplemente "agente", "dispositivo autónomo" o "dispositivo informático", "robot", y/o similares, se pueden hacer referencia de manera intercambiable a lo largo de todo este documento.

En algunos ejemplos, una unidad de procesamiento de gráficos (GPU) está acoplada de manera comunicativa a núcleos de anfitrión/de procesador para acelerar las operaciones de gráficos, las operaciones de aprendizaje automático, las operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede acoplarse de manera comunicativa al procesador/núcleos de anfitrión a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otros ejemplos, la GPU se puede integrar en el mismo paquete o chip que los núcleos y se puede acoplar de manera comunicativa a los núcleos a través de un bus/interconexión interna del procesador (es decir, interna al paquete o chip). Independientemente de la manera en la que esté conectada la GPU, los núcleos del procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU usa, a continuación, circuitería/lógica especializada para procesar de manera eficiente estos comandos/instrucciones.

En la siguiente descripción, se exponen numerosos detalles específicos. Sin embargo, los ejemplos, como se describen en el presente documento, pueden practicarse sin estos detalles específicos. En otros casos, no se han mostrado en detalle circuitos, estructuras y técnicas bien conocidos para no complicar la comprensión de esta descripción.

Descripción general del sistema I

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático 100 configurado para implementar uno o más aspectos descritos en el presente documento. El sistema informático 100 incluye un subsistema de procesamiento 101 que tiene uno o más procesador o procesadores 102 y una memoria de sistema 104 que se comunica mediante una ruta de interconexión que puede incluir un concentrador de memoria 105. El concentrador de memoria 105 puede ser un componente separado dentro de un componente de conjunto de chips o puede integrarse dentro del uno o más procesadores 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 mediante un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede habilitar al sistema informático 100 para recibir entrada desde uno o más dispositivo o dispositivos de entrada 108. Adicionalmente, el concentrador de E/S 107 puede permitir que un controlador de visualización, que puede incluirse en el uno o más procesador o procesadores 102, proporcione salidas para uno o más dispositivo o dispositivos de visualización 110A. En un ejemplo, el uno o más dispositivo o dispositivos de visualización 110A acoplados al concentrador de E/S 107 pueden incluir un dispositivo de visualización local, interno o integrado.

En un ejemplo, el subsistema de procesamiento 101 incluye uno o más procesador o procesadores paralelos 112 acoplados al concentrador de memoria 105 mediante un bus u otro enlace de comunicación 113. El enlace de comunicación 113 puede ser uno de cualquier número de tecnologías o protocolos de enlace de comunicación basados en normas, tales como, pero sin limitación, PCI Express, o puede ser una interfaz de comunicaciones o estructura de

comunicaciones específica del proveedor. En un ejemplo, el uno o más procesador o procesadores paralelos 112 forman un sistema de procesamiento paralelo o vectorial computacionalmente enfocado que incluye un gran número de núcleos de procesamiento y/o agrupaciones de procesamiento, tales como un procesador de muchos núcleos integrados (MIC). En un ejemplo, el uno o más procesador o procesadores paralelos 112 forman un subsistema de procesamiento de gráficos que puede emitir píxeles a uno del uno o más dispositivo o dispositivos de visualización 110A acoplados mediante el concentrador de E/S 107. El uno o más procesador o procesadores paralelos 112 también pueden incluir un controlador de visualización e interfaz de visualización (no mostrados) para permitir una conexión directa a uno o más dispositivo o dispositivos de visualización 110B.

Dentro del subsistema de E/S 111, una unidad de almacenamiento de sistema 114 puede conectarse al concentrador de E/S 107 para proporcionar un mecanismo de almacenamiento para el sistema informático 100. Se puede usar un conmutador de E/S 116 para proporcionar un mecanismo de interfaz para permitir conexiones entre el concentrador de E/S 107 y otros componentes, tales como un adaptador de red 118 y/o un adaptador de red inalámbrica 119 que pueden estar integrados en la plataforma, y otros diversos dispositivos que puedan añadirse por medio de uno o más dispositivo o dispositivos de adición 120. El adaptador de red 118 puede ser un adaptador de Ethernet u otro adaptador de red alámbrico. El adaptador de red inalámbrico 119 puede incluir uno o más de un dispositivo de red de Wi-Fi, de Bluetooth, de comunicación de campo cercano (NFC) o de otro tipo que incluye una o más radios inalámbricas.

El sistema informático 100 que puede incluir otros componentes no explícitamente mostrados, que incluyen USB u otras conexiones de puerto, unidades de almacenamiento óptico, dispositivos de captura de vídeo, y similares, puede conectarse también al concentrador de E/S 107. Las rutas de comunicación que interconectan los diversos componentes en la **Figura 1** se pueden implementar usando cualquier protocolo adecuado, tal como protocolos (por ejemplo, PCI-Express) basados en PCI (Interconexión de Componentes Periféricos), o cualesquiera otras interfaces de comunicación de bus o de punto a punto y/o protocolo o protocolos, tal como la interconexión de alta velocidad NV-Link o protocolos de interconexión conocidos en la técnica.

En un ejemplo, el uno o más procesador o procesadores paralelos 112 incorporan circuitos optimizados para procesamiento de gráficos y vídeo, incluyendo, por ejemplo, circuitería de salida de vídeo, y constituyen una unidad de procesamiento de gráficos (GPU). En otro ejemplo, el uno o más procesador o procesadores paralelos 112 incorporan circuitería optimizada para procesamiento de propósito general, mientras conservan la arquitectura computacional subyacente, descrita en mayor detalle en el presente documento. En otro ejemplo más, los componentes del sistema informático 100 pueden estar integrados con uno o más de otros elementos de sistema en un único circuito integrado. Por ejemplo, el uno o más procesadores paralelos 112, el concentrador de memoria 105, el o los procesadores 102 y el concentrador de E/S 107 se pueden integrar en un circuito integrado de sistema en chip (SoC). Como alternativa, los componentes del sistema informático 100 pueden integrarse en un único paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una porción de los componentes del sistema informático 100 puede integrarse en un módulo de múltiples microprocesadores (MCM), que puede interconectarse con otros módulos de múltiples microprocesadores para dar un sistema informático modular.

Se apreciará que, el sistema informático 100 mostrado en el presente documento es ilustrativo y que son posibles variaciones y modificaciones. La topología de conexión, incluyendo el número y disposición de puentes, el número de procesadores 102 y el número de procesadores paralelos 112 se puede modificar como se desee. Por ejemplo, en algunos ejemplos, la memoria de sistema 104 está conectada al procesador o procesadores 102 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria de sistema 104 mediante el concentrador de memoria 105 y el procesador o procesadores 102. En otras topologías alternativas, el procesador o procesadores paralelos 112 están conectados al concentrador de E/S 107 o directamente a uno del uno o más procesador o procesadores 102, en lugar de al concentrador de memoria 105. En otros ejemplos, el concentrador de E/S 107 y el concentrador de memoria 105 se pueden integrar en un único chip. Algunos ejemplos pueden incluir dos o más conjuntos de procesador o procesadores 102 anexados mediante múltiples zócalos, que pueden acoplarse con dos o más instancias del procesador o procesadores paralelos 112.

Algunos de los componentes particulares mostrados en el presente documento son opcionales y pueden no estar incluidos en todas las implementaciones del sistema informático 100. Por ejemplo, puede ser compatible con cualquier número de tarjetas o periféricos de adición o se pueden eliminar algunos componentes. Además, algunas arquitecturas pueden usar terminología diferente para componentes similares a los ilustrados en la **Figura 1**. Por ejemplo, el concentrador de memoria 105 puede denominarse puente norte en algunas arquitecturas, mientras que el concentrador de E/S 107 puede denominarse puente sur.

La **Figura 2A** ilustra un procesador paralelo 200. Los diversos componentes del procesador paralelo 200 se pueden implementar usando uno o más dispositivos de circuito integrado, tales como procesadores programables, circuitos integrados específicos de la aplicación (ASIC) o campos de matrices de puertas programables (FPGA). El procesador paralelo 200 ilustrado es una variante del uno o más procesadores paralelos 112 mostrados en la **Figura 1**.

En un ejemplo, el procesador paralelo 200 incluye una unidad de procesamiento paralelo 202. La unidad de procesamiento paralelo incluye una unidad de E/S 204 que permite una comunicación con otros dispositivos, que incluyen otras instancias de la unidad de procesamiento paralelo 202. La unidad de E/S 204 se puede conectar

directamente a otros dispositivos. En un ejemplo, la unidad de E/S 204 se conecta con otros dispositivos mediante el uso de una interfaz de concentrador o de conmutador, tal como un concentrador de memoria 105. Las conexiones entre el concentrador de memoria 105 y la unidad de E/S 204 forman un enlace de comunicaciones 113. Dentro de la unidad de procesamiento paralelo 202, la unidad de E/S 204 se conecta con una interfaz de anfitrión 206 y una barra transversal de memoria 216, donde la interfaz de anfitrión 206 recibe comandos dirigidos a realizar operaciones de procesamiento y la barra transversal de memoria 216 recibe comandos dirigidos a realizar operaciones de memoria.

Cuando la interfaz de anfitrión 206 recibe una memoria intermedia de comando mediante la unidad de E/S 204, la interfaz de anfitrión 206 puede dirigir operaciones de trabajo para realizar aquellos comandos a un extremo frontal 208. En un ejemplo, el extremo frontal 208 se acopla con un planificador 210, que está configurado para distribuir comandos u otros elementos de trabajo a una matriz de agrupaciones de procesamiento 212. En un ejemplo, el planificador 210 garantiza que la matriz de agrupaciones de procesamiento 212 está configurada apropiadamente y en un estado válido antes de que se distribuyan las tareas a las agrupaciones de procesamiento de la matriz de agrupaciones de procesamiento 212.

La matriz de agrupaciones de procesamiento 212 puede incluir hasta "N" agrupaciones de procesamiento (por ejemplo, de la agrupación 214A, agrupación 214B a la agrupación 214N). Cada agrupación 214A-214N de la matriz de agrupaciones de procesamiento 212 puede ejecutar un gran número de hilos concurrentes. El planificador 210 puede asignar trabajo a las agrupaciones 214A-214N de la matriz de agrupación de procesamiento 212 usando diversos algoritmos de programación y/o distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surge para cada tipo de programa o cálculo. La planificación puede manejarse dinámicamente por el planificador 210, o puede ser ayudada, en parte, por lógica de compilador durante la compilación de la lógica de programa configurada para la ejecución por la matriz de agrupación de procesamiento 212.

En un ejemplo, pueden asignarse diferentes agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 para procesar diferentes tipos de programas o para realizar diferentes tipos de cálculos.

La matriz de agrupaciones de procesamiento 212 se puede configurar para realizar diversos tipos de operaciones de procesamiento paralelo. En un ejemplo, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de cálculo paralelo de propósito general. Por ejemplo, la matriz de agrupaciones de procesamiento 212 puede incluir lógica para ejecutar tareas de procesamiento que incluye filtración de datos de vídeo y/o de audio, realización de operaciones de modelado, que incluye operaciones físicas y realización de transformaciones de datos.

En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de procesamiento de gráficos en paralelo. En ejemplos en los que el procesador paralelo 200 está configurado para realizar operaciones de procesamiento de gráficos, la matriz de agrupaciones de procesamiento 212 puede incluir lógica adicional para soportar la ejecución de tales operaciones de procesamiento de gráficos, que incluyen, pero sin limitación, lógica de muestreo de textura para realizar operaciones de textura, así como lógica de teselación y otra lógica de procesamiento de vértices. Adicionalmente, la matriz de agrupaciones de procesamiento 212 puede configurarse para ejecutar programas sombreadores relacionados con el procesamiento de gráficos tales como, pero sin limitación, sombreadores de vértices, sombreadores de teselación, sombreadores de geometría y sombreadores de píxeles. La unidad de procesamiento paralelo 202 puede transferir datos desde la memoria de sistema por medio de la unidad de E/S 204 para su procesamiento. Durante el procesamiento, los datos transferidos pueden almacenarse en una memoria en chip (por ejemplo, la memoria de procesador paralelo 222) durante el procesamiento y, a continuación, escribirse en diferido en la memoria del sistema.

En un ejemplo, cuando se usa la unidad de procesamiento paralelo 202 para realizar el procesamiento de gráficos, el planificador 210 puede estar configurado para dividir la carga de trabajo de procesamiento en tareas de tamaño aproximadamente igual, para permitir mejor la distribución de las operaciones de procesamiento de gráficos a múltiples agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212. En algunos ejemplos, porciones de la matriz de agrupaciones de procesamiento 212 se pueden configurar para realizar diferentes tipos de procesamiento. Por ejemplo, una primera porción puede configurarse para realizar un sombreado de vértices y una generación de topología, una segunda porción puede configurarse para realizar sombreado de teselación y de geometría, y una tercera porción puede configurarse para realizar sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen representada para su visualización. Los datos intermedios producidos por una o más de las agrupaciones 214A-214N pueden almacenarse en memorias intermedias para permitir que se transmitan los datos intermedios entre las agrupaciones 214A-214N para su procesamiento adicional.

Durante la operación, la matriz de agrupaciones de procesamiento 212 puede recibir tareas de procesamiento a ejecutar mediante el planificador 210, que recibe comandos que definen tareas de procesamiento desde el extremo frontal 208. Para operaciones de procesamiento gráfico, las tareas de procesamiento pueden incluir índices de datos que se van a procesar, por ejemplo, datos de superficie (parche), datos de primitivas, datos de vértices y/o datos de píxeles, así como parámetros de estado y comandos que definen cómo han de procesarse los datos (por ejemplo, qué programa ha de ejecutarse). El planificador 210 puede estar configurado para extraer los índices que corresponden a las tareas o puede recibir los índices desde el extremo frontal 208. El extremo frontal 208 puede estar configurado

para garantizar que la matriz de agrupaciones de procesamiento 212 esté configurada en un estado válido antes de que se inicie la carga de trabajo especificada en memorias intermedias de comandos entrantes (por ejemplo, memorias intermedias de lotes, memorias intermedias de carga, etc.).

Cada una de la una o más instancias de la unidad de procesamiento paralelo 202 puede acoplarse con la memoria de procesador paralelo 222. Puede accederse a la memoria de procesador paralelo 222 mediante la barra transversal de memoria 216, que puede recibir solicitudes de memoria desde la matriz de agrupaciones de procesamiento 212, así como de la unidad de E/S 204. La barra transversal de memoria 216 puede acceder a la memoria de procesador paralelo 222 mediante una interfaz de memoria 218. La interfaz de memoria 218 puede incluir múltiples unidades de división (por ejemplo, la unidad de división 220A, la unidad de división 220B a la unidad de división 220N), cada una acoplable a una porción (por ejemplo, la unidad de memoria) de la memoria de procesador paralelo 222. En una implementación, el número de unidades de subdivisión 220A-220N está configurado para que sea igual al número de unidades de memoria, de manera que una primera unidad de subdivisión 220A tiene una primera unidad de memoria 224A correspondiente, una segunda unidad de subdivisión 220B tiene una unidad de memoria 224B correspondiente y una N-ésima unidad de subdivisión 220N tiene una N-ésima unidad de memoria 224N correspondiente. En otros ejemplos, el número de unidades de subdivisión 220A-220N puede no ser igual al número de dispositivos de memoria.

En diversos ejemplos, las unidades de memoria 224A-224N pueden incluir diversos tipos de dispositivos de memoria, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria gráfica de acceso aleatorio, tal como una memoria gráfica de acceso aleatorio síncrona (SGRAM), que incluye una memoria gráfica de doble tasa de datos (GDDR). En un ejemplo, las unidades de memoria 224A-224N también pueden incluir memoria 3D apilada, que incluye, pero sin limitación, memoria de ancho de banda alto (HBM). Los expertos en la materia apreciarán que la implementación específica de las unidades de memoria 224A-224N puede variar, y puede seleccionarse de uno de diversos diseños convencionales. Los objetivos de representación, tales como las memorias intermedias de fotograma o los mapas de textura pueden almacenarse a través de las unidades de memoria 224A-224N, permitiendo que las unidades de subdivisión 220A-220N escriban porciones de cada objetivo de representación en paralelo para usar de manera efectiva el ancho de banda disponible de la memoria de procesador paralelo 222. En algunos ejemplos, se puede excluir una instancia local de la memoria de procesador paralelo 222 en favor de un diseño de memoria unificado que utiliza memoria de sistema junto con memoria caché local.

En un ejemplo, una cualquiera de las agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 224A-224N dentro de la memoria de procesador paralelo 222. La barra transversal de memoria 216 puede estar configurada para transferir la salida de cada agrupación 214A-214N a cualquier unidad de subdivisión 220A-220N o a otra agrupación 214A-214N, que puede realizar operaciones de procesamiento adicionales en la salida. Cada agrupación 214A-214N puede comunicarse con la interfaz de memoria 218 a través de la barra transversal de memoria 216 para leer desde o escribir en diversos dispositivos de memoria externos. En un ejemplo, la barra transversal de memoria 216 tiene una conexión a la interfaz de memoria 218 para comunicarse con la unidad de E/S 204, así como una conexión a una instancia local de la memoria de procesador paralelo 222, lo que posibilita que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 214A-214N se comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 202. En un ejemplo, la barra transversal de memoria 216 puede usar canales virtuales para separar flujos de tráfico entre las agrupaciones 214A-214N y las unidades de subdivisión 220A-220N.

Aunque se ilustra una única instancia de la unidad de procesamiento paralelo 202 dentro del procesador paralelo 200, se puede incluir cualquier número de instancias de la unidad de procesamiento paralelo 202. Por ejemplo, pueden proporcionarse múltiples instancias de la unidad de procesamiento paralelo 202 en una única tarjeta de adición, o pueden interconectarse múltiples tarjetas de adición. Las diferentes instancias de la unidad de procesamiento paralelo 202 pueden configurarse para interfuncionar incluso si las diferentes instancias tienen diferentes números de núcleos de procesamiento, diferentes cantidades de memoria de procesador paralelo local y/u otras diferencias de configuración. Por ejemplo, y en un ejemplo, algunas instancias de la unidad de procesamiento paralelo 202 pueden incluir unidades de coma flotante de precisión más alta con relación a otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 202 o el procesador paralelo 200 pueden implementarse en una diversidad de configuraciones y factores de forma, incluyendo, pero sin limitación, sobremesa, portátil u ordenadores personales portátiles, servidores, estaciones de trabajo, consolas de juegos y/o sistemas integrados.

La **Figura 2B** es un diagrama de bloques de una unidad de subdivisión 220. En un ejemplo, la unidad de subdivisión 220 es una instancia de una de las unidades de subdivisión 220A-220N de la **Figura 2A**. Como se ilustra, la unidad de subdivisión 220 incluye una caché L2 221, una interfaz de memoria intermedia de fotograma 225 y una ROP 226 (unidad de operaciones de rasterización). La memoria caché L2 221 es una caché de lectura/escritura que está configurada para realizar operaciones de carga y almacenamiento recibidas desde la barra transversal de memoria 216 y la ROP 226. Los fallos de lectura y las solicitudes de escritura urgentes se emiten por la memoria caché L2 221 a la interfaz de memoria intermedia de fotogramas 225 para su procesamiento. Pueden enviarse también actualizaciones sucias a la memoria intermedia de fotogramas por medio de la interfaz de memoria intermedia de fotogramas 225 para su procesamiento oportuno. En una realización, la interfaz de memoria intermedia de fotograma

225 interconecta con una de las unidades de memoria en la memoria de procesador paralelo, tal como las unidades de memoria 224A-224N de la **Figura 2A** (por ejemplo, dentro de la memoria de procesador paralelo 222).

En las aplicaciones de gráficos, la ROP 226 es una unidad de procesamiento que realiza operaciones de rasterización tales como estarcido, prueba z, mezcla y similares. La ROP 226 emite, a continuación, datos de gráficos procesados que se almacenan en memoria de gráficos. En algunos ejemplos, la ROP 226 incluye lógica de compresión para comprimir datos de profundidad o de color que se escriben en memoria y descomprimir datos de profundidad o de color que se leen desde la memoria. En algunos ejemplos, la ROP 226 está incluida dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 214A-214N de la **Figura 2A**) en lugar de dentro de la unidad de subdivisión 220. En tal ejemplo, las solicitudes de lectura y escritura de datos de píxeles se transmiten a través de la barra transversal de memoria 216 en lugar de datos de fragmentos de píxeles.

Los datos de gráficos procesados pueden visualizarse en un dispositivo de visualización, tal como uno del uno o más dispositivo o dispositivos de visualización 110 de la **Figura 1**, enrutarse para su procesamiento adicional por el procesador o procesadores 102, o enrutarse para su procesamiento adicional por una de las entidades de procesamiento dentro del procesador paralelo 200 de la **Figura 2A**.

La **Figura 2C** es un diagrama de bloques de una agrupación de procesamiento 214 dentro de una unidad de procesamiento paralelo. En un ejemplo, la agrupación de procesamiento es una instancia de una de las agrupaciones de procesamiento 214A-214N de la **Figura 2A.** La agrupación de procesamiento 214 puede estar configurada para ejecutar muchos hilos en paralelo, donde el término "hilo" se refiere a una instancia de un programa particular que se ejecuta en un conjunto particular de datos de entrada. En algunos ejemplos, se usan técnicas de emisión de instrucción de única instrucción de múltiples datos (SIMD) para soportar la ejecución en paralelo de un gran número de hilos sin proporcionar múltiples unidades de instrucción independientes. En otros ejemplos, se usan técnicas de única instrucción de múltiples hilos (SIMT) para soportar la ejecución paralela de un gran número de hilos generalmente sincronizados, usando una unidad de instrucciones común configurada para emitir instrucciones en un conjunto de motores de procesamiento dentro de cada una de las agrupaciones de procesamiento. A diferencia del régimen de ejecución de SIMD, donde todos los motores de procesamiento ejecutan típicamente instrucciones idénticas, la ejecución de SIMT permite que diferentes hilos sigan más fácilmente rutas de ejecución divergentes a través de un programa de hilos dado. Los expertos en la materia entenderán que un régimen de procesamiento de SIMD representa un subconjunto funcional de un régimen de procesamiento de SIMT.

La operación de la agrupación de procesamiento 214 puede controlarse mediante un gestor de canalizaciones 232 que distribuye las tareas de procesamiento a procesadores paralelos de SIMT. El gestor de canalización 232 recibe instrucciones del planificador 210 de la **Figura 2A** y gestiona la ejecución de esas instrucciones a través de un multiprocesador de gráficos 234 y/o una unidad de texturas 236. El multiprocesador de gráficos 234 ilustrado es una instancia ilustrativa de un procesador paralelo de SIMT. Sin embargo, se pueden incluir diversos tipos de procesadores paralelos de SIMT de arquitecturas diferentes dentro de la agrupación de procesamiento 214. Una o más instancias del multiprocesador de gráficos 234 se pueden incluir dentro de una agrupación de procesamiento 214. El multiprocesador de gráficos 234 puede procesar datos y puede usarse una barra transversal de datos 240 para distribuir los datos procesados a uno de múltiples posibles destinos, que incluyen otras unidades sombreadoras. El gestor de canalizaciones 232 puede facilitar la distribución de datos procesados especificando destinos para que se distribuyan datos procesados mediante la barra transversal de datos 240.

Cada multiprocesador de gráficos 234 dentro de la agrupación de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades aritmético-lógicas, unidades de carga-almacén, etc.). La lógica de ejecución funcional puede configurarse de una manera canalizada en la que pueden emitirse instrucciones nuevas antes de que se hayan completado instrucciones previas. Se puede proporcionar la lógica de ejecución funcional. La lógica funcional soporta una diversidad de operaciones que incluyen aritmética de números enteros y de coma flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y de cálculo de diversas funciones algebraicas. En un ejemplo, se puede aprovechar el mismo hardware funcional-unitario para realizar diferentes operaciones y puede estar presente cualquier combinación de unidades funcionales.

Las instrucciones transmitidas a la agrupación de procesamiento 214 constituyen un hilo. Un conjunto de hilos que se ejecutan a través del conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa en diferentes datos de entrada. Cada hilo dentro de un grupo de hilos se puede asignar a un motor de procesamiento diferente dentro de un multiprocesador de gráficos 234. Un grupo de hilos puede incluir menos hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando un grupo de hilos incluye menos hilos que el número de motores de procesamiento, uno o más de los motores de procesamiento pueden estar inactivos durante los ciclos en los que se está procesando ese grupo de hilos. Un grupo de hilos puede incluir también más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando el grupo de hilos incluye más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234, el procesamiento se puede realizar durante ciclos de reloj consecutivos. En un ejemplo, pueden ejecutarse múltiples grupos de hilos concurrentemente en un multiprocesador de gráficos 234.

En un ejemplo, el multiprocesador de gráficos 234 incluye una memoria caché interna para realizar operaciones de carga y de almacenamiento. En un ejemplo, el multiprocesador de gráficos 234 puede prescindir de una caché interna y usar una memoria caché (por ejemplo, la caché L1 308) dentro de la agrupación de procesamiento 214. Cada multiprocesador de gráficos 234 también tiene acceso a cachés L2 dentro de las unidades de subdivisión (por ejemplo, las unidades de subdivisión 220A-220N de la **Figura 2A**) que se comparten entre todas las agrupaciones de procesamiento 214 y pueden usarse para transferir datos entre hilos. El multiprocesador de gráficos 234 puede acceder también a memoria global fuera de chip, que puede incluir una o más de memoria de procesador paralelo local y/o memoria de sistema. Puede usarse cualquier memoria externa a la unidad de procesamiento paralelo 202 como memoria global. Los ejemplos en los que la agrupación de procesamiento 214 incluye múltiples instancias del multiprocesador de gráficos 234 pueden compartir instrucciones y datos comunes, que pueden almacenarse en la caché L1 308.

Cada agrupación de procesamiento 214 puede incluir una MMU 245 (unidad de gestión de memoria) que está configurada para mapear direcciones virtuales en direcciones físicas. En otros ejemplos, una o más instancias de la MMU 245 pueden residir dentro de la interfaz de memoria 218 de la **Figura 2A**. La MMU 245 incluye un conjunto de entradas de tabla de página (PTE) usadas para mapear una dirección virtual a una dirección física de un mosaico (más información sobre la generación de mosaicos) y, opcionalmente, un índice de línea de memoria caché. La MMU 245 puede incluir memorias intermedias de traducción adelantada (TLB) de direcciones o cachés que pueden residir dentro del multiprocesador de gráficos 234 o la caché L1 o la agrupación de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso de datos de superficie para permitir una intercalación de solicitud eficiente entre unidades de subdivisión. El índice de línea de memoria caché se puede usar para determinar si una solicitud para una línea de memoria caché es un acierto o un fallo.

En aplicaciones de gráficos e informática, se puede configurar una agrupación de procesamiento 214 de tal manera que cada multiprocesador de gráficos 234 esté acoplado a una unidad de textura 236 para realizar operaciones de mapeo de textura, por ejemplo, determinar posiciones de muestras de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen desde una caché L1 de textura interna (no mostrada) o, en algunos ejemplos, desde la caché L1 dentro del multiprocesador de gráficos 234 y se extraen desde una caché L2, memoria de procesador paralelo local o memoria de sistema, según sea necesario. Cada multiprocesador de gráficos 234 emite tareas procesadas a la barra transversal de datos 240 para proporcionar la tarea procesada a otra agrupación de procesamiento 214 para su procesamiento adicional o para almacenar la tarea procesada en una caché L2, memoria de procesador paralelo local o memoria de sistema mediante la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones previas a la rasterización) está configurada para recibir datos desde el multiprocesador de gráficos 234, dirigir datos a las unidades de ROP, que pueden estar ubicadas con unidades de subdivisión como se describe en el presente documento (por ejemplo, las unidades de subdivisión 220A-220N de la **Figura 2A**). La unidad de preROP 242 puede realizar optimizaciones para la mezcla de color, organizar datos de color de píxel y realizar traducciones de dirección.

Se apreciará que la arquitectura de núcleo descrita en el presente documento es ilustrativa y que son posibles variaciones y modificaciones. Puede incluirse cualquier número de unidades de procesamiento, por ejemplo, el multiprocesador de gráficos 234, las unidades de texturas 236, las preROP 242, etc., dentro de una agrupación de procesamiento 214. Además, aunque únicamente se muestra una agrupación de procesamiento 214, la unidad de procesamiento paralelo, como se describe en el presente documento, puede incluir cualquier número de instancias de la agrupación de procesamiento 214. En un ejemplo, cada agrupación de procesamiento 214 puede configurarse para operar independientemente de otras agrupaciones de procesamiento 214 usando unidades de procesamiento separadas y distintas, cachés L1, etc.

La **Figura 2D** muestra un multiprocesador de gráficos 234. En tal ejemplo, el multiprocesador de gráficos 234 está acoplado al gestor de canalización 232 de la agrupación de procesamiento 214. El multiprocesador de gráficos 234 tiene una canalización de ejecución que incluye, pero sin limitación, una caché de instrucciones 252, una unidad de instrucciones 254, una unidad de mapeo de direcciones 256, un archivo de registro 258, uno o más núcleos de unidad de procesamiento de gráficos de propósito general (GPGPU) 262 y una o más unidades de carga/almacenamiento 266. Los núcleos de GPGPU 262 y las unidades de carga/almacenamiento 266 se acoplan con la memoria caché 272 y la memoria compartida 270 mediante una interconexión de memoria y de caché 268.

En un ejemplo, la caché de instrucciones 252 recibe un flujo de instrucciones para ejecutarse desde el gestor de canalizaciones 232. Las instrucciones se almacenan en caché en la caché de instrucciones 252 y se despachan para su ejecución por la unidad de instrucciones 254. La unidad de instrucciones 254 puede despachar instrucciones como grupos de hilos (por ejemplo, envolventes), con cada hilo del grupo de hilos asignado a una unidad de ejecución diferente dentro del núcleo de GPGPU 262. Una instrucción puede acceder a cualquiera de un espacio de direcciones local, compartido o global, especificando una dirección dentro de un espacio de direcciones unificado. La unidad de mapeo de direcciones 256 puede usarse para traducir direcciones en el espacio de direcciones unificado a una dirección de memoria distinta a la que pueden acceder las unidades de carga/almacén 266.

El archivo de registro 258 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de gráficos 324. El archivo de registro 258 proporciona almacenamiento temporal para los operandos conectados a las

rutas de datos de las unidades funcionales (por ejemplo, núcleos de GPGPU 262, unidades de carga/almacén 266) del multiprocesador de gráficos 324. En un ejemplo, el archivo de registro 258 se divide entre cada una de las unidades funcionales de manera que cada unidad funcional está asignada a una porción especializada del archivo de registro 258. En un ejemplo, el archivo de registro 258 se divide entre las diferentes envolventes que se ejecutan por el multiprocesador de gráficos 324.

Cada núcleo de GPGPU 262 puede incluir unidades de coma flotante (FPU) y/o unidades aritmético-lógicas (ALU) de números enteros que se usan para ejecutar instrucciones del multiprocesador de gráficos 324. Los núcleos de GPGPU 262 pueden ser similares en arquitectura o pueden diferir en arquitectura, de acuerdo con ejemplos. Por ejemplo, una primera porción de los núcleos de GPGPU 262 incluye una FPU de precisión sencilla y una ALU de números enteros, mientras que una segunda porción de los núcleos de GPGPU incluye una FPU de precisión doble. En un ejemplo, las FPU pueden implementar la norma IEEE 754-2008 para aritmética de coma flotante o posibilitar aritmética de coma flotante de precisión variable. El multiprocesador de gráficos 324 puede incluir adicionalmente una o más unidades de función fija o de función especial para realizar funciones específicas, tales como operaciones de copia de rectángulo o de mezcla de píxeles. En un ejemplo, uno o más de los núcleos de GPGPU puede incluir también lógica de función fija o especial.

La interconexión de memoria y caché 268 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador de gráficos 324 al archivo de registro 258 y a la memoria compartida 270. En un ejemplo, la interconexión de memoria y caché 268 es una interconexión de barra transversal que permite que la unidad de carga/almacén 266 implemente operaciones de carga y almacén entre la memoria compartida 270 y el archivo de registro 258. El archivo de registro 258 puede operar a la misma frecuencia que los núcleos de GPGPU 262, por lo tanto, la transferencia de datos entre los núcleos de GPGPU 262 y el archivo de registro 258 es de latencia muy baja. La memoria compartida 270 se puede usar para permitir la comunicación entre hilos que se ejecutan en las unidades funcionales dentro del multiprocesador de gráficos 234. La memoria caché 272 puede usarse como una caché de datos, por ejemplo, para almacenar en caché datos de textura comunicados entre las unidades funcionales y la unidad de textura 236. La memoria compartida 270 también se puede usar como un programa gestionado en caché. Los hilos que se ejecutan en los núcleos de GPGPU 262 pueden almacenar datos mediante programación dentro de la memoria compartida además de los datos almacenados en memoria caché automáticamente que se almacenan dentro de la memoria caché 272.

Las **Figuras 3A-3B** ilustran multiprocesadores de gráficos adicionales. Los multiprocesadores de gráficos 325, 350 ilustrados son variantes del multiprocesador de gráficos 234 de la **Figura 2C**. Los multiprocesadores de gráficos 325, 350 ilustrados pueden estar configurados como un multiprocesador de envío por flujo continuo (SM) que puede realizar la ejecución simultánea de un gran número de hilos de ejecución.

La **Figura 3A** muestra un multiprocesador de gráficos 325. El multiprocesador de gráficos 325 incluye múltiples instancias adicionales de unidades de recursos de ejecución con respecto al multiprocesador de gráficos 234 de la **Figura 2D**. Por ejemplo, el multiprocesador de gráficos 325 puede incluir múltiples instancias de la unidad de instrucciones 332A-332B, del archivo de registro 334A-334B y de la unidad o unidades de textura 344A-344B. El multiprocesador de gráficos 325 también incluye múltiples conjuntos de gráficos o unidades de ejecución de cálculo (p. ej., núcleo GPGPU 336A-336B, núcleo GPGPU 337A-337B, núcleo GPGPU 338A-338B) y múltiples conjuntos de unidades de carga/almacenamiento 340A-340B. En un ejemplo, las unidades de recurso de ejecución tienen una caché de instrucciones común 330, memoria caché de textura y/o de datos 342 y una memoria compartida 346. Los diversos componentes pueden comunicarse mediante una estructura de interconexión 327. En una realización, la estructura de interconexión 327 incluye uno o más conmutadores de barra transversal para permitir la comunicación entre los diversos componentes del multiprocesador de gráficos 325.

La **Figura 3B** muestra un multiprocesador de gráficos 350. El procesador de gráficos incluye múltiples conjuntos de recursos de ejecución 356A-356D, donde cada conjunto de recursos de ejecución incluye múltiples unidades de instrucciones, archivos de registro, núcleos de GPGPU y unidades de carga-almacenamiento, como se ilustra en la **Figura 2D** y en la **Figura 3A**. Los recursos de ejecución 356A-356D pueden funcionar en conjunto con la unidad o unidades de texturas 360A-360D para operaciones de textura, mientras que comparten una caché de instrucciones 354 y la memoria compartida 362. En un ejemplo, los recursos de ejecución 356A-356D pueden compartir una memoria caché de instrucciones 354 y una memoria compartida 362, así como múltiples instancias de una memoria de textura y/o de caché de datos 358A-358B. Los diversos componentes pueden comunicarse mediante una estructura de interconexión 352 similar a la estructura de interconexión 327 de la **Figura 3A**.

Los expertos en la materia entenderán que la arquitectura descrita en las **Figuras 1**, 2A-2D y 3A-3B es descriptiva y no limitante en cuanto al alcance. Por lo tanto, las técnicas descritas en el presente documento pueden implementarse en cualquier unidad de procesamiento configurada apropiadamente, incluyendo, sin limitación, uno o más procesadores de aplicaciones móviles, una o más unidades centrales de procesamiento (CPU) de sobremesa o de servidor, incluyendo CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tales como la unidad de procesamiento paralelo 202 de la **Figura 2A**, así como uno o más procesadores de gráficos o unidades de procesamiento de propósito especial.

En algunos ejemplos, un procesador paralelo o GPGPU, como se describe en el presente documento, está acoplado de manera comunicativa a núcleos de anfitrión/procesador para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede acoplarse de manera comunicativa al procesador/núcleos de anfitrión a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otros ejemplos, la GPU se puede integrar en el mismo paquete o chip que los núcleos y se puede acoplar de manera comunicativa a los núcleos a través de un bus/interconexión interna del procesador (es decir, interna al paquete o chip). Independientemente de la manera en la que esté conectada la GPU, los núcleos del procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU usa, a continuación, circuitería/lógica especializada para procesar de manera eficiente estos comandos/instrucciones.

Técnicas para la interconexión de GPU a un procesador de anfitrión

La **Figura 4A** ilustra una arquitectura ilustrativa en la que una pluralidad de GPU 410-413 están acopladas de manera comunicativa a una pluralidad de procesadores de múltiples núcleos 405-406 a través de enlaces de alta velocidad 440-443 (por ejemplo, buses, interconexiones punto a punto, etc.). En un ejemplo, los enlaces de alta velocidad 440-443 soportan un caudal de comunicación de 4 GB/s, 30 GB/s, 80 GB/s o mayor, dependiendo de la implementación. Pueden usarse diversos protocolos de interconexión que incluyen, pero sin limitación, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no están limitados a ningún protocolo de comunicación o caudal particular.

Además, en un ejemplo, dos o más de las GPU 410-413 están interconectadas a través de enlaces de alta velocidad 444-445, que pueden implementarse usando protocolos/enlaces iguales o diferentes a los usados para los enlaces de alta velocidad 440-443. De manera similar, dos o más de los procesadores de múltiples núcleos 405-406 pueden conectarse a través del enlace de alta velocidad 433, que puede ser buses de múltiples procesadores simétricos (SMP) que operan a 20 GB/s, 30 GB/s, 120 GB/s o más. Como alternativa, toda la comunicación entre los diversos componentes de sistema que se muestran en la **Figura 4A** se puede lograr usando los mismos protocolos/enlaces (por ejemplo, a través de una estructura de interconexión común). Sin embargo, como se ha mencionado, los principios subyacentes de la invención no están limitados a ningún tipo particular de tecnología de interconexión.

En un ejemplo, cada procesador de múltiples núcleos 405-406 está acoplado de manera comunicativa a una memoria de procesador 401-402, mediante las interconexiones de memoria 430-431, respectivamente, y cada GPU 410-413 está acoplada de manera comunicativa a la memoria de la GPU 420-423 a través de las interconexiones de memoria de GPU 450-453, respectivamente. Las interconexiones de memoria 430-431 y 450-453 pueden utilizar las mismas tecnologías de acceso de memoria u otras diferentes. A modo de ejemplo, y no como limitación, las memorias de procesador 401-402 y las memorias de GPU 420-423 pueden ser memorias volátiles, tales como memorias de acceso aleatorio dinámicas (DRAM) (que incluyen DRAM apiladas), SDRAM DDR de gráficos (GDDR) (por ejemplo, GDDR5, GDDR6), o memoria de alto ancho de banda (HBM) y/o pueden ser memorias no volátiles, tales como 3D XPoint o Nano-Ram. En un ejemplo, alguna porción de las memorias puede ser memoria volátil y otra porción puede ser memoria no volátil (por ejemplo, usando una jerarquía de memoria de dos niveles (2LM)).

Como se describe a continuación, aunque los diversos procesadores 405-406 y las GPU 410-413 pueden estar físicamente acoplados a una memoria particular 401-402, 420-423, respectivamente, puede implementarse una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtual (también denominado espacio "de direcciones eficaces") está distribuido entre todas las diversas memorias físicas. Por ejemplo, cada una de las memorias de procesador 401-402 puede comprender 64 GB del espacio de direcciones de memoria del sistema y cada una de las memorias de GPU 420-423 puede comprender 32 GB del espacio de direcciones de memoria del sistema (dando como resultado un total de 256 GB de memoria direccionable en este ejemplo).

La **Figura 4B** ilustra detalles adicionales para una interconexión entre un procesador de múltiples núcleos 407 y un módulo de aceleración de gráficos 446. El módulo de aceleración de gráficos 446 puede incluir uno o más chips de GPU integrados en una tarjeta de línea que está acoplada al procesador 407 mediante el enlace de alta velocidad 440. Como alternativa, el módulo de aceleración de gráficos 446 puede estar integrado en el mismo paquete o chip que el procesador 407.

El procesador 407 ilustrado incluye una pluralidad de núcleos 460A-460D, cada uno con una memoria intermedia de traducción adelantada 461A-461D y una o más cachés 462A-462D. Los núcleos pueden incluir diversos otros componentes para ejecutar instrucciones y procesar datos, que no se han ilustrado para evitar oscurecer los principios subyacentes de la invención, (por ejemplo, unidades de extracción de instrucciones, unidades de predicción de ramificación, decodificadores, unidades de ejecución, memorias intermedias de reordenación, etc.). Las cachés 462A-462D pueden comprender cachés de nivel 1 (L1) y de nivel 2 (L2). Además, una o más memorias caché compartidas 426 pueden incluirse en la jerarquía de almacenamiento en caché y ser compartidas por conjuntos de núcleos 460A-460D. Por ejemplo, un ejemplo del procesador 407 incluye 24 núcleos, cada uno con su propia memoria caché L1, doce memorias caché L2 compartidas y doce memorias caché L3 compartidas. En este ejemplo, una de las memorias caché L2 y L3 están compartidas por dos núcleos adyacentes. El procesador 407 y el módulo de integración de

acelerador de gráficos 446 se conectan con la memoria de sistema 441, que puede incluir las memorias de procesador 401-402.

Se mantiene la coherencia para los datos e instrucciones almacenados en las diversas cachés 462A-462D, 456 y en la memoria de sistema 441 mediante la comunicación inter-núcleo a través de un bus de coherencia 464. Por ejemplo, cada caché puede tener una lógica/circuitería de coherencia de caché asociada con la misma para comunicarse a través del bus de coherencia 464 en respuesta a lecturas o escrituras detectadas en líneas de caché particulares. En una implementación, se implementa un protocolo de monitorización de caché a través del bus de coherencia 464 para monitorizar los accesos de caché. Los expertos en la materia entienden bien las técnicas de coherencia/monitorización de caché y no se describirán en el presente documento en detalle para evitar oscurecer los principios subyacentes de la invención.

En un ejemplo, un circuito proxy 425 acopla de manera comunicativa el módulo de aceleración de gráficos 446 al bus de coherencia 464, permitiendo al módulo de aceleración de gráficos 446 participar en el protocolo de coherencia de caché como un par de los núcleos. En particular, una interfaz 435 proporciona conectividad al circuito proxy 425 a través del enlace de alta velocidad 440 (por ejemplo, un bus PCIe, NVLink, etc.) y una interfaz 437 conecta el módulo de aceleración de gráficos 446 al enlace 440.

En una implementación, un circuito de integración de acelerador 436 proporciona servicios de gestión de caché, acceso a memoria, gestión de contexto y gestión de interrupciones en beneficio de una pluralidad de motores de procesamiento de gráficos 431, 432, N del módulo de aceleración de gráficos 446. Cada motor de procesamiento de gráficos 431, 432, N puede comprender una unidad de procesamiento de gráficos (GPU) separada. Como alternativa, los motores de procesamiento de gráficos 431, 432, N pueden comprender diferentes tipos de motores de procesamiento de gráficos dentro de una GPU, tales como unidades de ejecución de gráficos, motores de procesamiento de medios (por ejemplo, codificadores/decodificadores de vídeo), muestreadores y motores de BLIT. En otras palabras, el módulo de aceleración de gráficos puede ser una GPU con una pluralidad de motores de procesamiento de gráficos 431-432, N, o los motores de procesamiento de gráficos 431-432, N pueden ser GPU individuales integradas en un paquete, tarjeta de línea o chip común.

En un ejemplo, el circuito de integración de acelerador 436 incluye una unidad de gestión de memoria (MIW) 439 para realizar diversas funciones de gestión de memoria tales como traducciones de memoria virtual a física (también denominadas traducciones de memoria efectiva a real) y protocolos de acceso de memoria para acceder a la memoria de sistema 441. La MMU 439 puede incluir también una memoria intermedia de traducción adelantada (TLB) (no mostrada) para almacenar en caché las traducciones de dirección virtual/eficaz a física/real. En una implementación, una caché 438 almacena comandos y datos para un acceso eficiente por los motores de procesamiento de gráficos 431-432, N. En un ejemplo, los datos almacenados en la caché 438 y en las memorias de gráficos 433-434, N se mantienen coherentes con las cachés de núcleo 462A-462D, 456 y la memoria de sistema 411. Como se ha mencionado, esto puede conseguirse mediante el circuito proxy 425 que toma parte en el mecanismo de coherencia de caché en nombre de la caché 438 y las memorias 433-434, N (por ejemplo, enviando actualizaciones a la caché 438 relacionadas con modificaciones/accesos de líneas de caché en las cachés de procesador 462A-462D, 456 y recibiendo actualizaciones desde la caché 438).

Un conjunto de registros 445 almacenan datos de contexto para hilos ejecutados por los motores de procesamiento de gráficos 431-432, N y un circuito de gestión de contexto 448 gestiona los contextos de hilo. Por ejemplo, el circuito de gestión de contexto 448 puede realizar operaciones de guardado y restauración para guardar y restaurar contextos de los diversos hilos durante conmutaciones de contexto (por ejemplo, en donde se guarda un primer hilo y se almacena un segundo hilo de modo que el segundo hilo puede ejecutarse por un motor de procesamiento de gráficos). Por ejemplo, en un cambio de contexto, el circuito de gestión de contexto 448 puede almacenar valores de registro actuales en una región designada en memoria (por ejemplo, identificada por un puntero de contexto). A continuación, puede restablecer los valores de registro cuando se vuelve al contexto. En una realización, un circuito de gestión de interrupciones 447 recibe y procesa interrupciones recibidas desde los dispositivos de sistema.

En una implementación, las direcciones virtuales/efectivas de un motor de procesamiento de gráficos 431 se traducen a direcciones reales/físicas en la memoria de sistema 411 por la MMU 439. Una realización del circuito de integración de acelerador 436 admite múltiples (por ejemplo, 4, 8, 16) módulos de aceleración de gráficos 446 y/u otros dispositivos de aceleración. El módulo de acelerador de gráficos 446 puede estar especializado en una única aplicación ejecutada en el procesador 407 o puede compartirse entre múltiples aplicaciones. En un ejemplo, hay un entorno de ejecución de gráficos virtualizado en el que los recursos de los motores de procesamiento de gráficos 431-432, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos pueden subdividirse en "cortes" que se asignan a diferentes VM y/o aplicaciones basándose en los requisitos de procesamiento y las propiedades asociadas con las VM y/o las aplicaciones.

Por tanto, el circuito de integración de acelerador actúa como un puente al sistema para el módulo de aceleración de gráficos 446 y proporciona servicios de traducción de direcciones y de memoria caché de sistema. Además, el circuito de integración de acelerador 436 puede proporcionar instalaciones de virtualización para que el procesador de anfitrión gestione la virtualización de los motores de procesamiento de gráficos, las interrupciones y la gestión de memoria.

Debido a que los recursos de hardware de los motores de procesamiento de gráficos 431-432, N se asignan explícitamente al espacio de direcciones real visto por el procesador de anfitrión 407, cualquier procesador de anfitrión puede direccionar estos recursos directamente usando un valor de dirección efectivo. Una función del circuito de integración de acelerador 436, en un ejemplo, es la separación física de los motores de procesamiento de gráficos 431-432, N de modo que aparecen al sistema como unidades independientes.

Como se ha mencionado, en el ejemplo ilustrado, una o más memorias de gráficos 433-434, M están acopladas a cada uno de los motores de procesamiento de gráficos 431-432, N, respectivamente. Las memorias de gráficos 433-434, M almacenan instrucciones y datos que son procesados por cada uno de los motores de procesamiento de gráficos 431-432, N. Las memorias de gráficos 433-434, M pueden ser memorias volátiles, tales como DRAM (incluyendo DRAM apiladas), memoria GDDR (por ejemplo, GDDR5, GDDR6) o HBM y/o pueden ser memorias no volátiles, tales como 3D XPoint o Nano-Ram.

En un ejemplo, para reducir el tráfico de datos a través del enlace 440, se usan técnicas de desvío para garantizar que los datos almacenados en las memorias de gráficos 433-434, M sean datos que serán usados con mayor frecuencia por los motores de procesamiento de gráficos 431-432, N y, preferentemente, no usados por los núcleos 460A-460D (al menos no con frecuencia). De manera similar, el mecanismo de desvío intenta mantener los datos que necesitan los núcleos (y, preferentemente, no los motores de procesamiento de gráficos 431-432, N) dentro de las memorias caché 462A-462D, 456 de los núcleos y de la memoria de sistema 411.

La **Figura 4C** ilustra otro ejemplo en la que el circuito de integración de acelerador 436 está integrado dentro del procesador 407. En este ejemplo, los motores de procesamiento de gráficos 431-432, N se comunican directamente a través del enlace de alta velocidad 440 al circuito de integración de acelerador 436 mediante la interfaz 437 y la interfaz 435 (que, de nuevo, pueden utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración de acelerador 436 puede realizar las mismas operaciones que las descritas con respecto a la **Figura 4B,** pero potencialmente con un mayor rendimiento dada su proximidad al bus de coherencia 462 y las caché 462A-462D, 426.

Un ejemplo soporta diferentes modelos de programación que incluyen un modelo de programación de proceso especializado (sin virtualización de módulo de aceleración de gráficos) y modelos de programación compartida (con virtualización). Este último puede incluir modelos de programación que son controlados por el circuito de integración de acelerador 436 y modelos de programación que son controlados por el módulo de aceleración de gráficos 446.

En un ejemplo del modelo de proceso especializado, los motores de procesamiento de gráficos 431-432, N están especializados a una única aplicación o proceso bajo un único sistema operativo. La única aplicación puede canalizar otras solicitudes de aplicación a los motores de gráficos 431-432, N, lo que proporciona virtualización dentro de una VM/subdivisión.

En los modelos de programación de proceso especializado, los motores de procesamiento de gráficos 431-432, N pueden estar compartidos por múltiples VM/subdivisiones de aplicación. Los modelos compartidos requieren un hipervisor de sistema para virtualizar los motores de procesamiento de gráficos 431-432, N para permitir el acceso por cada sistema operativo. Para sistemas de subdivisión única sin un hipervisor, los motores de procesamiento de gráficos 431-432, N son propiedad del sistema operativo. En ambos casos, el sistema operativo puede virtualizar los motores de procesamiento de gráficos 431-432, N para proporcionar acceso a cada proceso o aplicación.

Para el modelo de programación compartida, el módulo de aceleración de gráficos 446 o un motor de procesamiento de gráficos individual 431-432, N selecciona un elemento de proceso usando un manejador de proceso. En un ejemplo, los elementos de proceso se almacenan en memoria de sistema 411 y son direccionables usando las técnicas de traducción de dirección efectivo a dirección real descritas en el presente documento. El manejador de proceso puede ser un valor específico de la implementación proporcionado al proceso de anfitrión cuando registra su contexto con el motor de procesamiento de gráficos 431-432, N (es decir, solicitando que el software de sistema añada el elemento de proceso a la lista enlazada de elementos de proceso). Los 16 bits inferiores del gestor de procesos pueden ser el desplazamiento del elemento de proceso dentro de la lista enlazada de elementos de proceso.

La **Figura 4D** ilustra un corte de integración de acelerador 490 ilustrativo. Como se usa en el presente documento, un "corte" comprende una porción específica de los recursos de procesamiento del circuito de integración de acelerador 436. El espacio de direcciones efectivo de la aplicación 482 dentro de la memoria de sistema 411 almacena elementos de proceso 483. En un ejemplo, los elementos de proceso 483 se almacenan en respuesta a invocaciones de GPU 481 desde las aplicaciones 480 ejecutadas en el procesador 407. Un elemento de proceso 483 contiene el estado de proceso para la correspondiente aplicación 480. Un descriptor de trabajo (WD) 484 contenido en el elemento de proceso 483 puede ser un único trabajo solicitado por una aplicación o puede contener un puntero a una cola de trabajos. En el último caso, el WD 484 es un puntero a la cola de solicitudes de trabajo en el espacio de direcciones de la aplicación 482.

El módulo de aceleración de gráficos 446 y/o los motores de procesamiento de gráficos individuales 431-432, N pueden compartirse por todos o un subconjunto de los procesos en el sistema. Los ejemplos invención incluyen una

infraestructura para configurar el estado de proceso y enviar un WD 484 a un módulo de aceleración de gráficos 446 para empezar un trabajo en un entorno virtualizado.

En una implementación, el modelo de programación de proceso especializado es específico de la implementación. En este modelo, un único proceso posee el módulo de aceleración de gráficos 446 o un motor de procesamiento de gráficos individual 431. Debido a que el módulo de aceleración de gráficos 446 es de propiedad de un único proceso, el hipervisor inicializa el circuito de integración de acelerador 436 para la subdivisión de propiedad y el sistema operativo inicializa el circuito de integración de acelerador 436 para el proceso de propiedad en el momento cuando se asigna el módulo de aceleración de gráficos 446.

Durante la operación, una unidad de extracción de WD 491 en el corte de integración de acelerador 490 extrae el siguiente WD 484 que incluye una indicación del trabajo a hacer por uno de los motores de procesamiento de gráficos del módulo de aceleración de gráficos 446. Los datos del WD 484 pueden almacenarse en los registros 445 y usarse por la MMU 439, el circuito de gestión de interrupciones 447 y/o el circuito de gestión de contexto 446 como se ilustra. Por ejemplo, una realización de la MMU 439 incluye circuitería de recorrido de páginas/segmentos para acceder a las tablas de segmentos/páginas 486 dentro del espacio de direcciones virtual del SO 485. El circuito de gestión de interrupciones 447 puede procesar los eventos de interrupción 492 recibidos desde el módulo de aceleración de gráficos 446. Cuando se realizan operaciones de gráficos, se traduce una dirección efectiva 493 generada por un motor de procesamiento de gráficos 431-432, N a una dirección real por la MMU 439.

En un ejemplo, el mismo conjunto de registros 445 se duplica para cada motor de procesamiento de gráficos 431-432, N y/o módulo de aceleración de gráficos 446, y puede inicializarse por el hipervisor o el sistema operativo. Cada uno de estos registros duplicados puede incluirse en un corte de integración de acelerador 490. Se muestran los registros ilustrativos que pueden inicializarse por el hipervisor en la **Tabla 1.**

Tabla 1 - Registros inicializados por el hipervisor

1	Registro de control de corte
2	Puntero de área de procesos planificados de dirección real (RA)
3	Registro de anulación de máscara de autoridad
4	Desplazamiento de entrada de tabla de vectores de interrupción
5	Límite de entrada de tabla de vectores de interrupción
6	Registro de estado
7	ID de división lógica
8	Puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

Se muestran los registros ilustrativos que pueden inicializarse por el sistema operativo en la **Tabla 2.**

Tabla 2 - Registros inicializados por sistema operativo

1	Identificación de proceso y de hilo
2	Puntero de grabación/restauración de contexto de dirección efectiva (EA)
3	Puntero de registro de utilización de acelerador de dirección virtual (VA)
4	Puntero de tabla de segmentos de almacenamiento de dirección virtual (VA)
5	Máscara de autoridad
6	Descriptor de trabajo

En un ejemplo, cada WD 484 es específico a un módulo de aceleración de gráficos particular 446 y/o a los motores de procesamiento de gráficos 431-432, N. Contiene toda la información que requiere un motor de procesamiento de gráficos 431-432, N para hacer su trabajo o puede ser un puntero a una ubicación de memoria donde la aplicación ha establecido una cola de comandos de trabajo para que se complete.

La **Figura 4E** ilustra detalles adicionales para un modelo compartido. Este ejemplo incluye un espacio de direcciones real de hipervisor 498 en el que se almacena una lista de elementos de proceso 499. El espacio de direcciones real de hipervisor 498 es accesible mediante un hipervisor 496 que virtualiza los motores de módulo de aceleración de gráficos para el sistema operativo 495.

Los modelos de programación compartida permiten que todos o un subconjunto de procesos de todas o un subconjunto de divisiones en el sistema usen un módulo de aceleración de gráficos 446. Hay dos modelos de programación donde el módulo de aceleración de gráficos 446 se comparte por múltiples procesos y subdivisiones: compartido en cortes de tiempo y compartido dirigido a gráficos.

En este modelo, el hipervisor de sistema 496 tiene propiedad del módulo de aceleración de gráficos 446 y hace que su función esté disponible para todos los sistemas operativos 495. Para que un módulo de aceleración de gráficos 446 soporte la virtualización por el hipervisor de sistema 496, el módulo de aceleración de gráficos 446 puede adherirse a los siguientes requisitos: 1) La solicitud de trabajo de una aplicación debe ser autónoma (es decir, no es necesario mantener el estado entre trabajos) o el módulo de aceleración de gráficos 446 debe proporcionar un mecanismo de grabación y restauración de contexto. 2) El módulo de aceleración de gráficos 446 garantiza que la solicitud de trabajo de una aplicación se completa en una cantidad especificada de tiempo, incluyendo cualquier fallo de traducción, o el módulo de aceleración de gráficos 446 proporciona la capacidad de dar prioridad al procesamiento del trabajo. 3) Se ha de garantizar al módulo de aceleración de gráficos 446 la equidad entre procesos cuando se opera en el modelo de programación compartido dirigido.

En un ejemplo, para el modelo compartido, se requiere que la aplicación 480 haga una llamada de sistema al sistema operativo 495 con un tipo de módulo de aceleración de gráficos 446, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de grabación/restauración de contexto (CSRP). El tipo del módulo de aceleración de gráficos 446 describe la función de aceleración dirigida como objetivo para la llamada de sistema. El tipo del módulo de aceleración de gráficos 446 puede ser un valor específico de sistema. El WD se formatea específicamente para el módulo de aceleración de gráficos 446 y puede estar en forma de un comando de módulo de aceleración de gráficos 446, un puntero de dirección efectiva a una estructura definida por el usuario, un puntero de dirección efectiva a una cola de comandos o cualquier otra estructura de datos para describir el trabajo que va a hacerse por el módulo de aceleración de gráficos 446. En un ejemplo, el valor de AMR es el estado de AMR que se debe usar para el proceso actual. El valor pasado al sistema operativo es similar a una aplicación que configura el AMR. Si las implementaciones del circuito de integración de acelerador 436 y del módulo de aceleración de gráficos 446 no soportan un registro de anulación de máscara de autoridad de usuario (UAMOR), el sistema operativo puede aplicar el valor de UAMOR actual al valor de AMR antes de pasar el AMR en la llamada de hipervisor. Opcionalmente, el hipervisor 496 puede aplicar el valor de registro de anulación de máscara de autoridad (AMOR) actual antes de colocar el AMR en el elemento de proceso 483. En un ejemplo, el CSRP es uno de los registros 445 que contienen la dirección efectiva de un área en el espacio de direcciones de la aplicación 482 para que el módulo de aceleración de gráficos 446 grabe y restablezca el estado de contexto. Este puntero es opcional si no se requiere que se grabe estado entre trabajos o cuando se da prioridad a un trabajo. El área de guardado/restauración de contexto puede estar fijada en la memoria de sistema.

Tras recibir la llamada de sistema, el sistema operativo 495 puede verificar que se ha registrado la aplicación 480 y que se le ha dado la autoridad para usar el módulo de aceleración de gráficos 446. El sistema operativo 495, a continuación, llama al hipervisor 496 con la información mostrada en la **Tabla 3.**

Tabla 3 - Parámetros de llamada de SO a hipervisor

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de grabación/restauración de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) e ID de hilo (TID) opcional
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de segmento de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)

Tras recibir la llamada de hipervisor, el hipervisor 496 verifica que se ha registrado el sistema operativo 495 y se le ha dado la autoridad para usar el módulo de aceleración de gráficos 446. A continuación, el hipervisor 496, pone el elemento de proceso 483 en la lista enlazada de elementos de proceso para el correspondiente tipo de módulo de aceleración de gráficos 446. El elemento de proceso puede incluir la información mostrada en la **Tabla 4**

Tabla 4 - Información de elemento de proceso

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de grabación/restauración de contexto (CSRP) de dirección efectiva (EA)

4	Un ID de proceso (PID) e ID de hilo (TID) opcional
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de segmento de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)
8	Tabla de vectores de interrupción, derivada de los parámetros de llamada de hipervisor.
9	Un valor de registro de estado (SR)
10	Un ID de subdivisión lógica (LPID)
11	Un puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
12	El registro de descriptor de almacenamiento (SDR)

En un ejemplo, el hipervisor inicializa una pluralidad de registros 445 de corte de integración de acelerador 490.

Como se ilustra en la **Figura 4F,** un ejemplo emplea una memoria unificada direccionable mediante un espacio de direcciones virtual de memoria común usado para acceder a las memorias de procesador físico 401-402 y a las memorias de GPU 420-423. En esta implementación, las operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de direcciones de memoria virtual/efectivo para acceder a las memorias de procesador 401-402 y viceversa, simplificando de esta manera la programabilidad. En un ejemplo, una primera porción del espacio de direcciones virtual/efectivo está asignada a la memoria de procesador 401, una segunda porción a la segunda memoria de procesador 402, una tercera porción a la memoria de GPU 420, y así sucesivamente. El espacio de memoria virtual/efectivo total (en ocasiones denominado el espacio de direcciones efectivo) está distribuido, de esta manera, a través de cada una de las memorias de procesador 401-402 y de las memorias de GPU 420-423, permitiendo que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual mapeada a esa memoria.

En un ejemplo, el circuito de gestión de desvío/coherencia 494A-494E dentro de una o más de las MMU 439A-439E garantiza la coherencia de caché entre las cachés de los procesadores de anfitrión (por ejemplo, 405) y las GPU 410-413 e implementa técnicas de desvío que indican las memorias físicas en las que deben almacenarse ciertos tipos de datos. Si bien se ilustran múltiples instancias de circuitería de gestión de desvío/coherencia 494A-494E en **Figura 4F,** la circuitería de desvío/coherencia puede implementarse dentro de la MMU de uno o más procesadores anfitriones 405 y/o dentro del circuito de integración de acelerador 436.

Un ejemplo permite que la memoria adjunta a la GPU 420-423 se mapee como parte de la memoria de sistema, y que se acceda usando tecnología de memoria virtual compartida (SVM), pero sin sufrir las desventajas de rendimiento típicas asociadas con la coherencia de caché de sistema completa. La capacidad de que se acceda a la memoria adjunta a la GPU 420-423 como memoria de sistema sin sobrecarga de coherencia de caché onerosa proporciona un entorno de operación beneficioso para la descarga de la GPU. Esta disposición permite que el software del procesador de anfitrión 405 establezca operandos y acceda a resultados de cálculo, sin la sobrecarga de copias de datos de DMA de E/S tradicionales. Tales copias tradicionales implican llamadas de controlador, interrupciones y accesos de E/S de memoria mapeada (MMIO) que son todos ineficaces con relación a los accesos de memoria sencillos. Al mismo tiempo, la capacidad de acceder a la memoria adjunta a la GPU 420-423 sin sobrecargas de coherencia de caché puede ser crítica para el tiempo de ejecución de un cálculo descargado. En casos con tráfico de memoria de escritura de envío por flujo continuo sustancial, por ejemplo, la sobrecarga de coherencia de caché puede reducir significativamente el ancho de banda de escritura eficaz observado por una GPU 410-413. La eficiencia de la configuración del operando, la eficiencia del acceso a los resultados y la eficiencia del cálculo de GPU, todos desempeñan una función al determinar la efectividad de la descarga de la GPU.

En una implementación, la selección entre el desvío de GPU y el desvío de procesador de anfitrión se controla por una estructura de datos de rastreador de desvío. Se puede usar una tabla de desvío, por ejemplo, que puede ser una estructura granular de página (es decir, controlada en la granularidad de una página de memoria) que incluye 1 o 2 bits por página de memoria adjunta a la GPU. La tabla de desvíos puede implementarse en un intervalo de memoria robado de una o más memorias adjuntas a la GPU 420-423, con o sin una caché de desvío en la GPU 410-413 (por ejemplo, para almacenar en caché entradas usadas de manera frecuente/reciente de la tabla de desvíos). Como alternativa, la tabla de desvío entera puede mantenerse dentro de la GPU.

En una implementación, se accede a la entrada de tabla de desvíos asociada a cada acceso a la memoria adjunta a la GPU 420-423 antes del acceso real a la memoria de GPU, lo que provoca las siguientes operaciones. En primer lugar, las solicitudes locales de la GPU 410-413 que encuentran su página en el desvío de GPU se reenvían directamente a una correspondiente memoria de GPU 420-423. Las solicitudes locales desde la GPU que encuentran su página en el desvío de anfitrión se reenvían al procesador 405 (por ejemplo, a través de un enlace de alta velocidad como se ha analizado anteriormente). En un ejemplo, las solicitudes del procesador 405 que encuentran la página

solicitada en el desvío de procesador de anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, las solicitudes dirigidas a una página con desvío de GPU pueden redirigirse a la GPU 410-413. A continuación, la GPU puede hacer que la página pase a un desvío de procesador anfitrión si no está usando actualmente la página.

El estado de desvío de una página puede cambiarse mediante un mecanismo basado en software, mediante un mecanismo basado en software asistido por hardware, o, para un conjunto limitado de casos, mediante un mecanismo basado puramente en hardware.

Un mecanismo para cambiar el estado de desvío emplea una llamada API (por ejemplo, OpenCL), que, a su vez, llama al controlador de dispositivo de la GPU que, a su vez, envía un mensaje (o pone en cola un descriptor de comando) a la GPU indicándole que cambie el estado de desvío y, para algunas transiciones, realice una operación de vaciado de caché en el anfitrión. Se requiere la operación de vaciado de caché para una transición desde el procesador de anfitrión 405 a un desvío de GPU, pero no se requiere para la transición opuesta.

En una realización, se mantiene la coherencia de caché representando temporalmente las páginas con desvío de GPU que no pueden almacenare en caché por el procesador de anfitrión 405. Para acceder a estas páginas, el procesador 405 puede solicitar acceso desde la GPU 410, que puede conceder o no acceso inmediato, dependiendo de la implementación. Por tanto, para reducir la comunicación entre el procesador 405 y la GPU 410 es beneficioso garantizar que las páginas con desvío por la GPU sean aquellas que requiere la GPU, pero no el procesador anfitrión 405 y viceversa.

Canalización de procesamiento de gráficos

La **Figura 5** ilustra una canalización de procesamiento de gráficos 500. En un ejemplo, un procesador de gráficos puede implementar la canalización de procesamiento de gráficos 500 ilustrada. El procesador de gráficos puede estar incluido dentro de los subsistemas de procesamiento paralelo como se describe en el presente documento, tal como el procesador paralelo 200 de la **Figura 2A**, que, es una variante del procesador o procesadores paralelos 112 de la **Figura 1.** Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento de gráficos 500 mediante una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 202 de la **Figura 2A**) como se describe en el presente documento. Por ejemplo, una unidad sombreadora (por ejemplo, el multiprocesador de gráficos 234 de la **Figura 2D**) puede configurarse para realizar las funciones de una o más de una unidad de procesamiento de vértices 504, una unidad de control de proceso de teselación 508, una unidad de procesamiento de evaluación de teselación 512, una unidad de procesamiento de geometría 516 y una unidad de procesamiento de fragmentos/píxeles 524. Las funciones del ensamblador de datos 502, los ensambladores de primitivas 506, 514, 518, la unidad de teselación 510, el rasterizador 522, y la unidad de operaciones de rasterización 526 pueden realizarse también por otros motores de procesamiento dentro de una agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 214 de la **Figura 3A**) y una unidad de subdivisión correspondiente (por ejemplo, la unidad de subdivisión 220A-220N de la **Figura 2A**). La canalización de procesamiento de gráficos 500 puede implementarse también usando unidades de procesamiento especializadas para una o más funciones. En un ejemplo, pueden realizarse una o más porciones de la canalización de procesamiento de gráficos 500 mediante lógica de procesamiento paralelo dentro de un procesador de propósito general (por ejemplo, la CPU). En un ejemplo, una o más porciones de la canalización de procesamiento de gráficos 500 pueden acceder a una memoria en chip (por ejemplo, la memoria de procesador paralelo 222 como en la **Figura 2A**) mediante una interfaz de memoria 528, que puede ser una instancia de la interfaz de memoria 218 de la **Figura 2A**.

En un ejemplo, el ensamblador de datos 502 es una unidad de procesamiento que recopila datos de vértices para superficies y primitivas. El ensamblador de datos 502, a continuación, emite los datos de vértices, que incluyen los atributos de vértices, a la unidad de procesamiento de vértices 504. La unidad de procesamiento de vértices 504 es una unidad de ejecución programable que ejecuta programas sombreadores de vértices, iluminando y transformando datos de vértice según lo especificado por los programas sombreadores de vértices. La unidad de procesamiento de vértices 504 lee datos que se almacenan en memoria caché, local o de sistema para su uso en el procesamiento de los datos de vértices y puede programarse para transformar los datos de vértices desde una representación de coordenadas basada en objetos hasta un espacio de coordenadas de espacio mundial o un espacio de coordenadas de dispositivo normalizado.

Una primera instancia de un ensamblador de primitivas 506 recibe atributos de vértices desde la unidad de procesamiento de vértices 504. El ensamblador de primitivas 506 lee atributos de vértices almacenados según sea necesario y construye primitivas de gráficos para su procesamiento por la unidad de procesamiento de control de teselación 508. Las primitivas de gráficos incluyen triángulos, segmentos de línea, puntos, parches y así sucesivamente de acuerdo con son soportados por varias interfaces de programación de aplicaciones (API) de procesamiento de gráficos.

La unidad de procesamiento de control de teselación 508 trata los vértices de entrada como puntos de control para un parche geométrico. Los puntos de control se transforman desde una representación de entrada desde el parche (por ejemplo, las bases del parche) a una representación que es adecuada para su uso en la evaluación superficial por la

unidad de procesamiento de evaluación de teselación 512. La unidad de procesamiento de control de teselación 508 también puede calcular factores de teselación para bordes de parches geométricos. Se aplica un factor de teselación a un único borde y cuantifica un nivel de detalle dependiente de la vista asociado con el borde. Una unidad de teselación 510 está configurada para recibir los factores de teselación para bordes de un parche y para teselar el parche en múltiples primitivas geométricas, tales como primitivas de línea, de triángulo o cuadrilaterales, que se transmiten a una unidad de procesamiento de evaluación de teselación 512. La unidad de procesamiento de evaluación de teselación 512 opera en coordenadas parametrizadas del parche subdividido para generar una representación superficial y atributos de vértice para cada vértice asociado con las primitivas geométricas.

Una segunda instancia de un ensamblador de primitivas 514 recibe atributos de vértices desde la unidad de procesamiento de evaluación de teselación 512, que lee los atributos de vértices almacenados de acuerdo con sea necesario y construye primitivas de gráficos para su procesamiento por la unidad de procesamiento de geometría 516. La unidad de procesamiento de geometría 516 es una unidad de ejecución programable que ejecuta programas sombreadores de geometría para transformar primitivas de gráficos recibidas desde el ensamblador de primitivas 514 según se especifica por los programas sombreadores de geometría. En un ejemplo, la unidad de procesamiento de geometría 516 está programada para subdividir las primitivas de gráficos en una o más primitivas de gráficos nuevas y calcular parámetros usados para rasterizar las nuevas primitivas de gráficos.

En algunos ejemplos, la unidad de procesamiento de geometría 516 puede añadir o borrar elementos en el flujo de geometría. La unidad de procesamiento de geometría 516 emite los parámetros y vértices que especifican nuevas primitivas de gráficos al ensamblador de primitivas 518. El ensamblador de primitivas 518 recibe los parámetros y vértices desde la unidad de procesamiento de geometría 516 y construye primitivas de gráficos para su procesamiento por una unidad de escala, selección y recorte de ventana gráfica 520. La unidad de procesamiento de geometría 516 lee datos que están almacenados en la memoria de procesador paralelo o en la memoria de sistema para su uso en el procesamiento de los datos de geometría. La unidad de escala, selección y recorte de ventana gráfica 520 realiza recorte, selección y escalado de ventana gráfica y emite primitivas de gráficos procesadas a un rasterizador 522.

El rasterizador 522 puede realizar optimizaciones de selección de profundidad y otras basadas en profundidad. El rasterizador 522 también realiza una conversión de exploración en las nuevas primitivas de gráficos para generar fragmentos y emitir esos fragmentos y datos de cobertura asociados a la unidad de procesamiento de fragmentos/píxeles 524.

La unidad de procesamiento de fragmentos/píxeles 524 es una unidad de ejecución programable que está configurada para ejecutar programas de sombreadores de fragmentos o programas de sombreadores de píxeles. Transformando la unidad de procesamiento de fragmentos/píxeles 524 fragmentos o píxeles recibidos desde el rasterizador 522, como se especifica por los programas sombreadores de fragmentos o de píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 puede programarse para realizar operaciones que incluyen, pero sin limitación, mapeo de textura, sombreado, mezcla, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se emiten a una unidad de operaciones de rasterización 526. La unidad de procesamiento de fragmentos/píxeles 524 puede leer datos que se almacenan en cualquiera de la memoria de procesador paralelo o la memoria de sistema para su uso cuando se procesan los datos de fragmento. Los programas de sombreado de fragmentos o de píxeles pueden estar configurados para sombrear a granularidad de muestra, de píxel, de mosaico u otras dependiendo de la tasa de muestreo configurada para las unidades de procesamiento.

La unidad de operaciones de rasterización 526 es una unidad de procesamiento que realiza operaciones de rasterización que incluyen, pero sin limitación, estarcido, prueba z, mezcla y similares, y emite datos de píxeles como datos de gráficos procesados para que se almacenen en la memoria de gráficos (por ejemplo, la memoria de procesador paralelo 222 como en la **Figura 2A**, y/o la memoria de sistema 104 como en la **Figura 1**, para que se visualicen en el uno o más dispositivo o dispositivos de visualización 110 o para su procesamiento adicional por uno del uno o más procesador o procesadores 102 o procesador o procesadores paralelos 112. En algunos ejemplos, la unidad de operaciones de rasterización 526 está configurada para comprimir datos z o de color que se escriben en memoria y descomprimir datos z o de color que se leen desde la memoria.

La **Figura 6** ilustra un dispositivo informático 600 que aloja un mecanismo de coordinación de inferencia y utilización de procesamiento ("mecanismo de coordinación/utilización") 610. El dispositivo informático 600 representa un dispositivo de comunicación y procesamiento de datos que incluye (pero sin limitación) dispositivos portátiles inteligentes, teléfonos inteligentes, dispositivos de realidad virtual (VR), pantallas montadas en la cabeza (HMD), ordenadores móviles, dispositivos del Internet de las cosas (IoT), ordenadores portátiles, ordenadores de sobremesa, ordenadores de servidor, etc., y ser similar o igual que el dispositivo informático 100 de la **Figura 1;** en consecuencia, por brevedad, claridad y facilidad de comprensión, muchos de los detalles establecidos anteriormente con referencia a las **Figuras 1-5** no se analizan ni se repiten adicionalmente más adelante.

El dispositivo informático 600 puede incluir además (sin limitaciones) una máquina autónoma o un agente artificialmente inteligente, tal como un agente o máquina mecánica, un agente o máquina electrónica, un agente o máquina virtual, un agente o máquina electromecánica, etc. Ejemplos de máquinas o agentes artificialmente inteligentes pueden incluir (sin limitación) robots, vehículos autónomos (por ejemplo, automóviles autónomos, aviones

autónomos, barcos autónomos, etc.), equipos autónomos (vehículos de construcción autónomos, equipos médicos autónomos, etc.), y/o similares. A través de todo este documento, "dispositivo informático" puede denominarse de manera intercambiable "máquina autónoma" o "agente artificialmente inteligente" o simplemente "robot".

Se contempla que, aunque a lo largo de este documento se hace referencia a "vehículo autónomo" y "conducción autónoma", los ejemplos no están limitados como tales. Por ejemplo, "vehículo autónomo" no se limita a un automóvil, sino que puede incluir cualquier número y tipo de máquinas autónomas, tales como robots, equipos autónomos, dispositivos domésticos autónomos y/o similares, y una o más tareas u operaciones relacionadas con tales máquinas autónomas pueden denominarse de manera intercambiable con la conducción autónoma.

El dispositivo informático 600 puede incluir además (sin limitaciones) grandes sistemas informáticos, tales como ordenadores de servidor, ordenadores de sobremesa, etc., y puede incluir además decodificadores de salón (por ejemplo, decodificadores de salón de televisión por cable basados en Internet, etc.), dispositivos basados en el sistema de posicionamiento global (GPS), etc. El dispositivo informático 600 puede incluir dispositivos informáticos móviles que dan servicio como dispositivos de comunicación, tales como teléfonos celulares, que incluyen teléfonos inteligentes, asistentes digitales personales (PDA), ordenadores de tabletas, ordenadores portátiles, lectores electrónicos, televisores inteligentes, plataformas de televisión, dispositivos llevables (por ejemplo, gafas, relojes, pulseras, tarjetas inteligentes, joyas, prendas de vestir, etc.), reproductores multimedia, etc. Por ejemplo, en un ejemplo, el dispositivo informático 600 puede incluir un dispositivo informático móvil que emplea una plataforma informática que aloja un circuito integrado ("CI"), tal como un sistema en un chip ("SoC" o "SOC"), que integra diversos componentes de hardware y/o software del dispositivo informático 600 en un único chip.

Como se ilustra, en un ejemplo, el dispositivo informático 600 puede incluir cualquier número y tipo de componentes de hardware y/o software, tales como (sin limitación) la unidad de procesamiento gráfico ("GPU" o simplemente "procesador de gráficos") 614, el controlador de gráficos (también denominado "controlador de GPU", "lógica de controlador de gráficos", "lógica de controlador", controlador de modo de usuario (UMD), UMD, estructura de controlador de modo de usuario (UMDF), UMDF, o simplemente "controlador") 616, unidad central de procesamiento ("CPU" o simplemente "procesador de aplicación") 612, memoria 608, dispositivos de red, controladores, o similares, así como fuentes de entrada/salida (E/S) 604, tales como pantallas táctiles, paneles táctiles, almohadillas táctiles, teclados virtuales o normales, ratones virtuales o normales, puertos, conectores, etc. El dispositivo informático 600 puede incluir un sistema operativo (SO) 606 que da servicio como interfaz entre el hardware y/o los recursos físicos del dispositivo informático 600 y un usuario. Se contempla que el procesador de gráficos 614 y el procesador de aplicaciones 612 pueden ser uno o más procesador o procesador 102 de la **Figura 1.**

Debe apreciarse que para determinadas implementaciones puede preferirse un sistema menos o más equipado que el ejemplo descrito anteriormente. Por lo tanto, la configuración del dispositivo informático 600 puede variar de una implementación a otra dependiendo de numerosos factores, tales como limitaciones de precio, requisitos de rendimiento, mejoras tecnológicas u otras circunstancias.

Los ejemplos pueden implementarse como cualquiera o una combinación de: uno o más microchips o circuitos integrados interconectados usando una placa base, lógica de cableado permanente, software almacenado por un dispositivo de memoria y ejecutado por un microprocesador, firmware, un circuito integrado de específico de la aplicación (ASIC), y/o una matriz de puertas programables en campo (FPGA). Los términos "lógica", "módulo", "componente", "motor" y "mecanismo" pueden incluir, a modo de ejemplo, software o hardware y/o combinaciones de software y hardware.

En un ejemplo, el mecanismo de coordinación/utilización 610 puede alojarse o facilitarse por el sistema operativo 606 del dispositivo informático 600. En otro ejemplo, el mecanismo de coordinación/utilización 610 puede estar alojado en o parte de la unidad de procesamiento de gráficos ("GPU" o simplemente "procesador de gráficos") 614 o firmware del procesador de gráficos 614. Por ejemplo, el mecanismo de coordinación/utilización 610 puede integrarse o implementarse como parte del hardware de procesamiento del procesador de gráficos 614. De manera similar, en otro ejemplo más, el mecanismo de coordinación/utilización 610 puede estar alojado en o ser parte de la unidad central de procesamiento ("CPU" o simplemente "procesador de aplicaciones") 612. Por ejemplo, el mecanismo de coordinación/utilización 610 puede integrarse o implementarse como parte del hardware de procesamiento del procesador de aplicaciones 612. En otro ejemplo más, el mecanismo de coordinación/utilización 610 puede estar alojado en o parte de cualquier número y tipo de componentes del dispositivo informático 600, tal como una porción del mecanismo de coordinación/utilización 610 puede estar alojado en o parte del sistema operativo 606, otra porción puede estar alojada en o parte del procesador de gráficos 614, otra porción puede estar alojada en o parte del procesador de aplicaciones 612, mientras que una o más porciones del mecanismo de coordinación/utilización 610 pueden estar alojadas en o parte del sistema operativo 606 y/o cualquier número y tipo de dispositivos del dispositivo informático 600. Se contempla que una o más porciones o componentes del mecanismo de coordinación/utilización 610 puedan emplearse como hardware, software y/o firmware.

Se contempla que los ejemplos no están limitados a ninguna implementación o alojamiento particular del mecanismo de coordinación/utilización 610 y que el mecanismo de coordinación/utilización 610 y uno o más de sus componentes pueden implementarse como hardware, software, firmware o cualquier combinación de los mismos.

El dispositivo informático 600 puede alojar interfaz o interfaces de red para proporcionar acceso a una red, tal como una LAN, una red de área extensa (WAN), una red de área metropolitana (MAN), una red de área personal (PAN), Bluetooth, una red en la nube, una red móvil (por ejemplo, 3ª generación (3G), 4ª generación (4G), etc.), una intranet, Internet, etc. La interfaz o interfaces de red pueden incluir, por ejemplo, una interfaz de red inalámbrica que tiene una antena, que puede representar una o más antenas. La interfaz o interfaces de red también pueden incluir, por ejemplo, una interfaz de red alámbrica para comunicarse con dispositivos remotos por medio de un cable de red, que puede ser, por ejemplo, un cable Ethernet, un cable coaxial, un cable de fibra óptica, un cable serie o un cable paralelo.

Se pueden proporcionar realizaciones, por ejemplo, como un producto de programa informático que puede incluir uno o más medios legibles por máquina que tienen almacenados en los mismos instrucciones ejecutables por máquina que, cuando se ejecutan por una o más máquinas tales como un ordenador, una red de ordenadores u otros dispositivos electrónicos, pueden dar como resultado que la una o más máquinas lleven a cabo operaciones de acuerdo con las realizaciones descritas en el presente documento. Un medio legible por máquina puede incluir, pero sin limitación, disquetes, discos ópticos, CD-ROM (memorias de sólo lectura en disco compacto) y discos magnetoópticos, ROM, RAM, EPROM (memorias de sólo lectura programables y borrables), EEPROM (memorias de sólo lectura programables y borrables eléctricamente), tarjetas magnéticas u ópticas, memoria flash u otro tipo de soporte/medio legible por máquina adecuado para almacenar instrucciones ejecutables por máquina.

Además, las realizaciones pueden descargarse como un producto de programa informático, en donde el programa puede transferirse desde un ordenador remoto (por ejemplo, un servidor) a un ordenador solicitante (por ejemplo, un cliente) por medio de una o más señales de datos incorporadas en y/o moduladas por una onda portadora u otro medio de propagación a través de un enlace de comunicación (por ejemplo, un módem y/o conexión de red).

A través de todo del documento, el término "usuario" puede denominarse de manera intercambiable "espectador", "observador", "persona", "individuo", "usuario final" y/o similares. Cabe señalar que, a lo largo de todo este documento, se puede hacer referencia a términos como "dominio de gráficos" de manera intercambiable con "unidad de procesamiento de gráficos", "procesador de gráficos" o simplemente "GPU" y, de manera similar, "dominio de CPU" o "dominio de anfitrión" se pueden hacer referencia de manera intercambiable a "unidad de procesamiento de ordenador", "procesador de aplicaciones" o simplemente "CPU".

Cabe señalar que, términos y expresiones como "nodo", "nodo informático", "servidor", "dispositivo de servidor", "ordenador en la nube", "servidor en la nube", "ordenador de servidor en la nube", "máquina", "máquina de anfitrión", "dispositivo", "dispositivo informático", "ordenador", "sistema informático" y similares, pueden usarse de manera intercambiable en este documento. Cabe señalar además que, términos como "aplicación", "aplicación de software", "programa", "programa de software", "paquete", "paquete de software" y similares, pueden usarse de manera intercambiable a lo largo de todo este documento. Además, términos como "trabajo", "entrada", "solicitud", "mensaje" y similares se pueden usar de manera intercambiable en este documento.

La **Figura 7** ilustra el mecanismo de coordinación/utilización 610 de la **Figura 6.** Para abreviar, muchos de los detalles ya analizados con referencia a las **Figuras 1-6** no se repiten ni se analizan a continuación. En un ejemplo, el mecanismo de coordinación/utilización 610 puede incluir cualquier número y tipo de componentes, tales como (sin limitaciones): lógica de detección/monitorización 701; lógica de entrenamiento previamente analizada 703; lógica de coordinación de inferencia 705; y lógica de comunicación/compatibilidad 707; lógica de fusión temprana 709; lógica de planificación de redes neuronales 711; y lógica de utilización de procesamiento 713.

El hardware de procesamiento de gráficos actual es más potente de lo que normalmente se necesita para realizar inferencias, tal como en términos de capacidades de precisión. Los ejemplos proporcionan una técnica novedosa para usar la lógica de detección/monitorización 701 para detectar y monitorizar conjuntos de datos de entrenamiento previamente analizados y, a continuación, activar la lógica de entrenamiento previamente analizada 703 para determinar un rango <X,Y> y configurar el hardware de gráficos para que sea colocado dentro de este rango de valores.

Los ejemplos proporcionan una técnica novedosa para añadir la capacidad de configurar el hardware de procesamiento, tal como el del procesador de gráficos 614, el procesador de aplicaciones 612, etc., para adaptarse a un conjunto de datos para mejorar la eficiencia energética del cálculo de inferencia. Por ejemplo, la precisión de los datos de inferencia/predicción puede determinarse detectando y monitorizando en primer lugar conjuntos de datos de según se facilita por la lógica de detección/monitorización 701 y simultáneamente o posteriormente, analizando la precisión asociada con tales conjuntos de datos, que, cuando se usan y aplican, pueden permitir mantener la eficiencia energética mientras se adaptan al hardware creado para capacidades de superconjunto.

En algunos ejemplos, el hardware de inferencia (tal como el de los procesadores de aplicaciones y/o gráficos 612, 614) puede diseñarse a priori para capacidades máximas, tales como precisión, etc. Por ejemplo, en tiempo de ejecución, puede ser necesario que se tenga una capacidad de precisión para un subconjunto de lo que soporta el hardware del procesador correspondiente. En un ejemplo, la información observada y obtenida de conjuntos de datos de entrenamiento puede usarse para configurar el hardware, tal como hardware de aplicación y/o los procesadores de

gráficos 612, 614. En un ejemplo, usar el hardware de superconjunto da como resultado una eficiencia energética subóptima, ya que la aplicación de software descarta o ignora las capacidades adicionales.

Como se ilustra con referencia a la **Figura 8A,** el hardware de inferencia, tal como el hardware de aplicación y/o los procesadores gráficos 612, 614, puede diseñarse para cubrir cualquier tamaño y precisión de datos previstos. Para mejorar la eficiencia en el momento de la inferencia, aquellas porciones o porciones del hardware que no son necesarias para los conjuntos de datos pueden desconectarse para ahorrar energía, etc., pero, en tales aplicaciones, sería de mayor interés maximizar el rendimiento del hardware. Además, para aumentar el número de operaciones realizadas por segundo, aquellos bloques de hardware necesarios para diversas operaciones, tales como sumar, multiplicar, acumular y/o similares, pueden reconfigurarse según se facilita por la lógica de entrenamiento previamente analizada 703, tal como información que se va a configurar que se puede generar en el momento del entrenamiento basándose en el conjunto de datos y, a continuación, se puede comunicar al controlador de configuración de hardware en el tiempo de ejecución según se facilitar por la lógica de entrenamiento previamente analizada 703.

Los ejemplos proporcionan una técnica novedosa para aumentar la utilización del procesador de gráficos durante una inferencia a través de contextos múltiples. Por ejemplo, usando la lógica de utilización de procesamiento 713, se añade soporte para ejecutar múltiples contextos en el procesador de gráficos 614, donde cada contexto (tal como el proceso de aplicación) puede usarse para resolver la inferencia para una red neuronal. Estos contextos pueden tener espacios de direcciones separados, que pueden aplicarse por el hardware pertinente, tal como el procesador de gráficos 614.

En un ejemplo, según se facilita por la lógica de utilización de procesamiento 713, un microcontrolador basado en hardware (por ejemplo, planificador de contexto) puede facilitarse por la lógica de detección/monitorización 701 para monitorizar qué parte del dispositivo de procesamiento, tal como el procesador de gráficos 614, se utilizar por el contexto actual, tal como determinar si hay más problemas de inferencia que necesitan resolución). Típicamente, los problemas de inferencia son más sencillos y pueden infrautilizar el procesador de gráficos 614 de modo que el procesador de gráficos 614 no esté infrautilizado en este caso y en otros casos similares. Esto se ilustra y describe con más detalle con referencia a la **Figura 8B.**

Los ejemplos proporcionan además una técnica novedosa para facilitar la coordinación de la salida de inferencia y los sensores (por ejemplo, cámaras, micrófonos, otros sensores, etc.). Por ejemplo, las técnicas convencionales no proporcionan coordinación entre la salida de inferencia y los sensores que proporcionan la entrada. Los ejemplos proporcionan una técnica novedosa que puede encontrar un sensor para realizar una tarea (por ejemplo, aplicar un filtro, activar un dispositivo, ajustar una cámara, etc.), lo que permite mejorar la precisión de la salida de inferencia. Por ejemplo, cuando la confianza de la inferencia cae por debajo de un umbral, se puede aplicar un filtro a una cámara para intentar mejorar la confianza de la inferencia capturando o enfocándose en ciertos objetos o escenas, mientras se ignoran otros objetos o escenas según se facilita por la lógica de coordinación de inferencia 705.

Esta novedosa técnica permite además una coordinación a nivel de sistema entre sensores de fuentes de E/S 604 de la **Figura 6** y algoritmos y técnicas de aprendizaje profundo que pueden dar sentido a los sensores en el corazón de la carrera hacia un súper ordenador centralizado en un vehículo autónomo, tal como la máquina autónoma 600. A medida que los sistemas avanzan hacia el procesamiento centralizado de sensores (pero no el sensor en sí), la coordinación entre sensores y los diversos filtros que puede aplicar se basan en el conocimiento que existe dentro del cerebro central del ordenador, lo que resalta la diferencia entre detectar un objeto o no. Esto se ilustra mejor con referencia a la **Figura 8C.**

Los ejemplos proporcionan además una técnica novedosa para ofrecer detección de objetos basada en conjuntos. Por ejemplo, poder tomar decisiones reales en un modelo en lugar de esperar a que el siguiente resultado sea externo al modelo, tal como en la conducción automatizada cuando se trata de sensores de tipos variables con diferentes velocidades de datos de series temporales.

En un ejemplo, se puede usar la lógica de fusión temprana 709 para facilitar la comunicación temprana entre un modelo de cámara basado en imágenes capturadas por una o más cámaras y otro modelo, tal como un modelo de sistema de detección y alcance de luz ("LiDAR", "LIDAR" o simplemente "lidar"). Esta comunicación temprana puede incluir el intercambio de aciertos tempranos que conducen a la planificación temprana de la ruta, la toma de decisiones, etc., a través del módulo de identificación (ID) de objetos fusionados combinado según se facilita por la lógica de fusión temprana 703. En un ejemplo, esta comunicación temprana permite la fusión temprana al compartir sugerencias entre modelos para reducir una fusión típicamente separada después de que cada modelo se haya completado por separado. Esta técnica novedosa puede combinarse o aplicarse en procesos de fusión tempranos y realizarse como reemplazo de la fusión de bajo nivel. Esto se ilustra y describe con más detalle con referencia a la **Figura 8D**.

Las realizaciones de la presente invención proporcionan una técnica novedosa para la planificación de redes neuronales (NN), donde dicha planificación puede incluir una planificación de NN tolerante a fallos para la criticidad del tiempo y la eficiencia energética, según se facilita por la lógica de planificación de NN 710. Además, múltiples aplicaciones coexisten en el procesador de gráficos 614 para el empleo de inferencia en el despliegue, donde se define un porcentaje de prioridad para cada proceso, de modo que el procesador de gráficos 614 planifica un proceso de acuerdo con el porcentaje del total de hilos disponibles según se facilita por la lógica de planificación NN 710.

En una realización, el porcentaje mencionado anteriormente se ajusta dinámicamente por un usuario u otras primitivas de resultados de perfil para que el usuario actualice el porcentaje, donde el usuario define el límite inferior y el porcentaje esperado. Además, se puede usar un microcontrolador con un sistema operativo en tiempo real (RTOS) que gestiona las entradas de los sensores para reactivarse y realizar un entrenamiento periódico con prioridades de entrenamiento basadas en la criticidad del tiempo. Se contempla que puede ser necesaria la centralización de súper ordenadores con respecto a vehículos autónomos, tales como la máquina autónoma 600, para poder virtualizar y, a continuación, priorizar cargas de trabajo para propósitos de protección y seguridad en tiempo real.

Como se ilustrará y describirá con más detalle con referencia a las **Figuras 9A** y **9B,** múltiples aplicaciones coexisten en el procesador de gráficos 614 en despliegue, donde se define un porcentaje de prioridad para cada proceso. El procesador de gráficos 614 se ve facilitado por la lógica de planificación NN 711 para planificar procesos de acuerdo con el porcentaje del total de hilos disponibles y otros recursos. Este porcentaje se ajusta dinámicamente por el usuario, donde se proporcionan primitivas para que los usuarios actualicen el porcentaje. Por ejemplo, los usuarios pueden definir el límite inferior y el porcentaje esperado, donde los usuarios pueden necesitar esta característica para ajustar la utilización del procesador de gráficos de acuerdo con las aplicaciones y capacidades de hardware actuales.

Las siguientes tablas muestran cómo se puede usar una GPU, tal como el procesador de gráficos 614, para almacenar información relacionada en hardware o memoria. Por ejemplo, puede haber primitivas para que los usuarios seleccionen y escriban un porcentaje deseado y de límite inferior para un proceso identificado por la derivada integral proporcional (PID), mientras que puede haber primitivas para que los usuarios lean el porcentaje asignado actual del sistema y el porcentaje deseado por el usuario y el porcentaje de límite inferior. Cualquier porcentaje asignado por el sistema puede gestionarse mediante hardware GPU o a través de un proceso de gestión con privilegios.

ID de proceso	Porcentaje asignado al sistema	Porcentaje deseado requerido por el usuario	Porcentaje de límite inferior requerido por el usuario
PID A	Sistema A%	Usuario A%	Menor A%
PID B	Sistema B%	Usuario B%	Menor B%
PID C	Sistema C%	Usuario C%	Menor %C

Con el uso cada vez mayor del aprendizaje profundo en aplicaciones críticas para la seguridad, también se puede considerar el aspecto "crítico para la seguridad" de estos casos de uso para garantizar que el procesamiento de inferencia se realice en una cantidad de tiempo determinista y garantizado, tal como el intervalo de tiempo tolerante a fallos (FTTI). Esto debe hacerse antes de que el fallo en el cálculo de los resultados de cualquier pasada de inferencia provoque que falle una aplicación de bucle de control crítico de seguridad en tiempo real y que a continuación pueda provocar daños o lesiones a los seres humanos.

Lo que hace que esta consideración sea algo problemática es que los elementos informáticos, tales como el procesador de gráficos 614, que realizan las operaciones de inferencia normalmente son responsables de realizar también otras tareas, tales como otras operaciones de inferencia que no son críticas para la seguridad. Por ejemplo, en un robot industrial, se puede usar un modelo entrenado para la detección de personas y evitar que el robot golpee a una persona, mientras que otro modelo entrenado, ejecutándose al mismo tiempo en el mismo elemento de cálculo, se puede usar para aplicar aspectos de personalización al comportamiento del robot.

Por lo tanto, es esencial que en estas aplicaciones de "criticidad mixta" el dispositivo informático 600 pueda tener conocimiento de la "criticidad de seguridad" de un modelo de inferencia particular (por ejemplo, ASIL-D frente a ASIL-B o SIL-4 frente a SIL-1) cuando se planifican y asignan recursos informáticos, incluyendo cualquier capacidad de interrumpir un modelo de criticidad más baja con un modelo de criticidad más alta, según se facilita por la lógica de planificación NN 711. Esto se ilustra y describe con más detalle con referencia a la **Figura 9C.**

Además, la lógica de comunicación/compatibilidad 707 se puede usar para facilitar la comunicación y compatibilidad necesarias entre cualquier número de dispositivos del dispositivo informático 600 y diversos componentes del mecanismo de coordinación/utilización 610.

La lógica de comunicación/compatibilidad 707 puede usarse para facilitar la comunicación dinámica y la compatibilidad entre el dispositivo informático 600 y cualquier número y tipo de otros dispositivos informáticos (tales como dispositivo informático móvil, ordenador de escritorio, dispositivo informático de servidor, etc.); dispositivos o componentes de procesamiento (tales como CPU, GPU, etc.); dispositivos de captura/localización/detección (tales como componentes de captura/detección que incluyen cámaras, cámaras de detección de profundidad, sensores de cámara, sensores de rojo, verde, azul ("RGB" o "rgb"), micrófonos, etc.); dispositivos de visualización (tales como componentes de salida, que incluyen pantallas de visualización, áreas de visualización, proyectores de visualización, etc.); componentes de reconocimiento de usuario/contexto y/o sensores/dispositivos de identificación/verificación (tales como

sensores/detectores biométricos, escáneres, etc.); base o bases de datos 730, tales como memoria o dispositivos de almacenamiento, bases de datos y/o fuentes de datos (tales como dispositivos de almacenamiento de datos, discos duros, unidades de estado sólido, discos duros, tarjetas o dispositivos de memoria, circuitos de memoria, etc.); medio o medios de comunicación 725, tales como uno o más canales o redes de comunicación (por ejemplo, redes en la nube, Internet, intranets, redes celulares, redes de proximidad, tales como Bluetooth, Bluetooth de baja energía (BLE), Bluetooth inteligente, Wi-Fi de proximidad, identificación por radiofrecuencia (RFID), comunicación de campo cercano (NFC), red de área corporal (BAN), etc.); comunicaciones inalámbricas o alámbricas y protocolos pertinentes (por ejemplo, Wi-Fi®, WiMAX, Ethernet, etc.); técnicas de conectividad y gestión de ubicación; aplicaciones de software/sitios web (por ejemplo, sitios web de redes sociales y/o comerciales, etc., aplicaciones comerciales, juegos y otras aplicaciones de entretenimiento, etc.); y lenguajes de programación, etc., garantizando al mismo tiempo la compatibilidad con tecnologías, parámetros, protocolos, normas, etc., cambiantes.

Además, cualquier uso de una marca, palabra, término, expresión, nombre y/o acrónimo particular, tal como "detectar", "observar", "decidir", "ruta normal", "desvío", "bloque de cálculo", "omisión", "valor de datos de uso frecuente", "FDV", "máquina de estados finitos", "conjunto de entrenamiento", "agente", "máquina", "vehículo", "robot", "conducción", "CNN", "DNN", "NN", "unidad de ejecución", "EU", "memoria local compartida", "SLM", "flujos de gráficos", "caché", "caché de gráficos", "GPU", "procesador de gráficos", "dominio de GPU", "GPGPU", "CPU", "procesador de aplicaciones", "dominio de CPU", "controlador de gráficos", "carga de trabajo", "aplicación", "canalización de gráficos", "procesos de canalización", "API", "API 3D", "OpenGL®", "DirectX®", "hardware", "software", "agente", "controlador de gráficos", "controlador de gráficos en modo de núcleo", "controlador en modo de usuario", "estructura de controlador en modo de usuario", "memoria intermedia", "memoria intermedia de gráficos", "tarea", "proceso", "operación", "aplicación de software", "juego", etc., no deben interpretarse como limitantes de las realizaciones al software o dispositivos que llevan esa etiqueta en productos o en bibliografía externa a este documento.

Se contempla que se puede añadir y/o eliminar cualquier número y tipo de componentes del mecanismo de coordinación/utilización 610 para facilitar diversas realizaciones que incluyen añadir, eliminar y/o mejorar ciertas características. Por brevedad, claridad y facilidad de comprensión del mecanismo de coordinación/utilización 610, muchos de los componentes convencionales y/o conocidos, tales como los de un dispositivo informático, no se muestran ni analizan en este punto. Se contempla que las realizaciones, como se describen en el presente documento, no se limiten a ninguna tecnología, topología, sistema, arquitectura y/o norma particular y sean lo suficientemente dinámicas para adoptar y adaptarse a cualquier cambio futuro.

La **Figura 8A** ilustra una estructura de transacción 800 en procesadores de aplicaciones y/o gráficos 612, 614 para facilitar el entrenamiento previamente analizado. Para abreviar, muchos de los detalles previamente analizados con referencia a las **Figuras 1-7** pueden no analizarse o repetirse posteriormente en este punto. Cualquier proceso relacionado con la estructura 800 puede realizarse mediante lógica de procesamiento que puede comprender hardware (por ejemplo, circuitería, lógica especializada, lógica programable, etc.), software (tal como instrucciones ejecutadas en un dispositivo de procesamiento) o una combinación de los mismos, según se facilite por el mecanismo de coordinación/utilización 610 de la **Figura** 6. Los procesos asociados con la estructura 800 pueden ilustrarse o indicarse en secuencias lineales para mayor brevedad y claridad en la presentación; sin embargo, se contempla que cualquier número de ellos pueda realizarse en paralelo, de forma asincrónica o en diferentes órdenes. Además, los ejemplos no están limitados a ninguna ubicación, estructura, configuración o estructura arquitectónica particular de procesos y/o componentes, tal como la estructura 800.

Como se ilustra, en un ejemplo, el hardware de inferencia, tal como el hardware de aplicación y/o los procesadores de gráficos 612, 614, puede desarrollarse de tal manera que pueda cubrir todos los tamaños y precisiones de datos previstos. Por ejemplo, para mejorar la eficiencia en los tiempos de inferencia, ciertas partes del hardware que no son necesarias para los conjuntos de datos pueden desconectarse para preservar energía, etc. Sin embargo, en algunas aplicaciones, se considera más esencial maximizar el rendimiento del hardware.

En el ejemplo ilustrado, la estructura 800 incluye los datos de entrenamiento 801, el bloque de aprendizaje 803, los datos de inferencia 805 y modelos de hardware configurables 807, donde los datos de entrenamiento 801 se muestran comunicados al bloque de aprendizaje 803 y uno o más de los modelos de hardware configurables 807 tal como comunicando información de configuración 809 desde los datos de entrenamiento 801 a los modelos de hardware configurables 807. Además, por ejemplo, al recibir entradas de los datos de entrenamiento 801, el bloque de aprendizaje 803 y los datos de inferencia 805, uno o más modelos de hardware configurables 807 producen inferencia/predicción 811, como se ilustra.

Por ejemplo, para aumentar el número de operaciones realizadas por segundo, aquellos bloques de hardware de procesamiento que se necesitan para la suma, multiplicación, acumulación, etc., pueden reconfigurarse usándose como parte de modelos de hardware configurables 807 usando información de configuración 809 de los datos de entrenamiento 801. Esta información de configuración 809 puede generarse en el momento del entrenamiento basándose en uno o más conjuntos de datos y comunicarse a un controlador de configuración de hardware en los procesadores de aplicaciones y/o de gráficos 612, 614, en el tiempo de ejecución, según se facilita por la lógica de entrenamiento previamente analizada 703 de la **Figura 7**.

La **Figura 8B** ilustra un procesador de gráficos 614 para una mejor utilización de procesamiento. Para abreviar, muchos de los detalles previamente analizados con referencia a las **Figuras 1-8A** pueden no analizarse o repetirse posteriormente en este punto. Cualquier proceso relacionado con el procesador de gráficos 614 puede realizarse mediante lógica de procesamiento que puede comprender hardware (por ejemplo, circuitería, lógica especializada, lógica programable, etc.), software (tal como instrucciones ejecutadas en un dispositivo de procesamiento) o una combinación de los mismos, según se facilite por el mecanismo de coordinación/utilización 610 de la **Figura 6**. Los procesos asociados con el procesador de gráficos 614 pueden ilustrarse o indicarse en secuencias lineales para mayor brevedad y claridad en la presentación; sin embargo, se contempla que cualquier número de ellos pueda realizarse en paralelo, de forma asincrónica o en diferentes órdenes. Además, los ejemplos no están limitados a ninguna ubicación, estructura, configuración o estructura arquitectónica particular de procesos y/o componentes, tal como la colocación arquitectónica dentro del procesador de gráficos 614 ilustrada.

En un ejemplo, como se ilustra, los bloques de unidad de ejecución (EU) 831A, 831B, 831C y 831D ilustrados ejecutan el contexto-0, mientras que los bloques EU 833A, 833B, 833C, 833D, 833E y 833F ejecutan el contexto-1. Como se ilustra, el procesador de gráficos 614 se muestra alojando procesadores de envío por flujo continuo (SMM0) 821 y SMM1 823, que incluyen además las barreras 835A, 835B, las cachés L1/L2 837A, 837B, la memoria local compartida (SLM) 839A, 839B, respectivamente.

Como se ilustra, el planificador de contexto 820 realiza la monitorización de la utilización del procesador, tal como la monitorización de la utilización del procesador de gráficos 614, a través del bloque de monitorización de utilización de GPU 825 según se facilita por la lógica de detección/monitorización 701 de la **Figura 7**. Como se ilustra adicionalmente, el contexto-0 y el contexto-1 representados por EU 831A-831D y EU833A-833F, respectivamente, tienen espacios de direcciones separados, donde, en un ejemplo, el planificador de contexto de microcontrolador 820 del procesador de gráficos 614 monitoriza cuánto del procesador de gráficos 614 se utiliza. Si la utilización se considera baja, el planificador de contexto 820 puede despachar más contextos, lo que permite resolver problemas de inferencia adicionales.

La **Figura 8C** ilustra una secuencia de transacciones 850 para mejorar la coordinación de salidas y sensores de inferencia. Para abreviar, muchos de los detalles previamente analizados con referencia a las **Figuras 1-8B** pueden no analizarse o repetirse posteriormente en este punto. Cualquier proceso relacionado con la secuencia de transacciones 850 puede realizarse mediante lógica de procesamiento que puede comprender hardware (por ejemplo, circuitería, lógica especializada, lógica programable, etc.), software (tal como instrucciones ejecutadas en un dispositivo de procesamiento) o una combinación de los mismos, según se facilite por el mecanismo de coordinación/utilización 610 de la **Figura 6**. Los procesos asociados con la secuencia de transacciones 850 pueden ilustrarse o indicarse en secuencias lineales para mayor brevedad y claridad en la presentación; sin embargo, se contempla que cualquier número de ellos pueda realizarse en paralelo, de forma asincrónica o en diferentes órdenes. Además, los ejemplos no están limitados a ninguna ubicación, estructura, configuración o estructura arquitectónica particular de procesos y/o componentes, tal como la colocación arquitectónica dentro de la secuencia de transacciones 850.

La secuencia de transacciones 850 comienza con el sensor 851 (por ejemplo, cámara inteligente) de las fuentes de E/S 604 de la **Figura 6**, donde el sensor 851 puede incluir cualquier número y tipo de sensores, tales como cámaras inteligentes con imagen integrada, procesador de señales, donde, por ejemplo, el proveedor de servicios de Internet (ISP) puede ser externo a la cámara. Como se ilustra, se captura una imagen por el sensor/cámara 851 y la imagen capturada original a continuación se transmite al modelo 853, que puede indicar (tal como en un 13%) que los resultados de inferencia de nodos tienen una probabilidad mucho menor de lo normal (o por debajo de algún umbral).

En 855, en un ejemplo, se solicita al sensor/cámara 851 y/o ISP que apliquen un filtro a la imagen original antes de completar cualquier operación de inferencia, de modo que los resultados de inferencia puedan mejorarse según se facilita por la lógica de coordinación de inferencia 705 de la **Figura 7**. Por ejemplo, se puede utilizar un filtro para reducir cualquier objeto no deseado de la escena, tal como árboles, personas, tiendas, animales, etc., lo que conduce a una mejor calidad de los resultados. En un ejemplo, la lógica de coordinación de inferencia 705 de la **Figura 7** para facilitar que el sensor/cámara 851 aplique un filtro a las imágenes y/o vídeo que captura de modo que el filtro se use para filtrar el tráfico no deseado en las imágenes y/o vídeos, lo que conduce a continuación a un modelo 857 mejorado o potenciado basado en resultados mejorados.

La **Figura 8D** ilustra una secuencia de transacciones 870 para mejorar la coordinación de salidas y sensores de inferencia. Para abreviar, muchos de los detalles previamente analizados con referencia a las **Figuras 1-8C** pueden no analizarse o repetirse posteriormente en este punto. Cualquier proceso relacionado con la secuencia de transacciones 850 puede realizarse mediante lógica de procesamiento que puede comprender hardware (por ejemplo, circuitería, lógica especializada, lógica programable, etc.), software (tal como instrucciones ejecutadas en un dispositivo de procesamiento) o una combinación de los mismos, según se facilite por el mecanismo de coordinación/utilización 610 de la **Figura 6**. Los procesos asociados con la secuencia de transacciones 870 pueden ilustrarse o indicarse en secuencias lineales para mayor brevedad y claridad en la presentación; sin embargo, se contempla que cualquier número de ellos pueda realizarse en paralelo, de forma asincrónica o en diferentes órdenes. Además, los ejemplos no están limitados a ninguna ubicación, estructura, configuración o estructura arquitectónica

particular de procesos y/o componentes, tal como la colocación arquitectónica dentro de la secuencia de transacciones 870

La secuencia de transacciones 870 comienza con el sensor (por ejemplo, la cámara inteligente) 851 que captura imágenes y/o vídeos de escenas, donde estas imágenes/vídeos, etc., se usan para crear un modelo, tal como el modelo de cámara 871. Como se ilustra en este punto, usando la lógica de fusión temprana 709 de la **Figura 7**, la comunicación temprana se facilita entre el modelo de cámara 871 y otro modelo 877, tal como el modelo Lidar, a través y usando el módulo de ID de objetos fusionados 873 combinado. En un ejemplo, el modelo 877 puede extraerse u obtenerse del almacenamiento 879, que puede ser parte de una o más base o bases de datos 730 de la **Figura 7.** Como se ilustra, además, en un ejemplo, esta comunicación puede incluir correspondencia de pistas tempranas entre los dos modelos 871, 877, donde esta comunicación se recopila, almacena o comunica por el módulo de ID de objeto fusionado 873 combinado para que, a continuación, pueda usarse para planificación de rutas, toma de decisiones y otros planes y predicciones similares según se facilitan por la lógica de fusión temprana 709 de la **Figura 7**.

Las **Figuras 9A**, **9B** ilustran secuencias de transacciones 900, 930 que ilustran modelos de uso de acuerdo con una realización de la presente invención. Para abreviar, muchos de los detalles previamente analizados con referencia a las **Figuras 1-8D** pueden no analizarse o repetirse posteriormente en este punto. Como se ilustra en las secuencias de transacciones 900, 930, hay dos modelos de uso básicos, donde uno, como se muestra en la **Figura 9A**, es cómo el hardware de GPU o un proceso de gestión con privilegios puede actualizar el porcentaje asignado por el sistema y ajustar cada proceso para respetar la asignación.

Por ejemplo, como se muestra en la secuencia de transacciones 900, se emplea el controlador de PID 909 para controlar y ajustar el porcentaje de asignación 911 del sistema de cada proceso para lograr un alto nivel de utilizaciones de GPU. En una realización, este controlador de PID 909 está alojado o integrado en el procesador de gráficos 614. Como se ilustra con más detalle, los requisitos de la aplicación del usuario 901 sirven como límites superior e inferior para restringir la salida del controlador de modo que permanezca dentro del rango límite comunicando los límites superior e inferior al controlador de PID 909.

Además, como se ilustra, los datos de la asignación de sistema 903 actual, la demanda instantánea del planificador 905 y el objetivo de control actual 907 también se comunican al controlador de PID 909 para permitir un mejor control y gestión según se facilita por la lógica de planificación NN 711. Se contempla que el controlador de PID 909 pueda ser de cualquier rango de complejidad que pueda usarse para la asignación de porcentajes, mientras que un controlador derivativo integral proporcional puede ser básico.

Haciendo referencia ahora a la secuencia de transacciones 930 de la **Figura 9B,** muestra cómo las aplicaciones de usuario actualizan su porcentaje deseado y límite inferior de acuerdo con los requisitos del sistema actuales. Por ejemplo, como se ilustra, similar a la secuencia de transacciones 900 de la **Figura 9A,** se emplea un controlador de PID 909 que puede recibir información pertinente como los requisitos de PID del proceso de aplicación del usuario 931, la asignación de sistema 933 actual, la demanda instantánea del planificador 933 y el objetivo de control actual 937 para a continuación procesar esta información individual y/o colectivamente para proporcionar un mejor control y ajuste de los requisitos de la aplicación del siguiente usuario 941.

La **Figura 9C** ilustra un gráfico 950 que ilustra opciones de priorización. Para abreviar, muchos de los detalles previamente analizados con referencia a las **Figuras 1-9B** pueden no analizarse o repetirse posteriormente en este punto. En el ejemplo ilustrado del gráfico 950, según se facilita por la lógica de planificación NN 711 de la **Figura 7**, la priorización se puede usar para asignar más unidades de ejecución a una NN frente a otras, lo que aún puede permitir que la red 953 no relacionada con la seguridad se ejecute siempre que la red crítica para la seguridad 951, 953 tenga el recurso que necesita. Estos recursos pueden incluir o hacer referencia a una cantidad de memoria, caché, memoria de borrador, un porcentaje de algún elemento de cálculo, etc. Esta novedosa técnica puede implementarse en software, hardware o cualquier combinación de los mismos.

Por ejemplo, la NN2 953 no relacionada con la seguridad se muestra como interrumpida debido a que la NN3 955 crítica para la seguridad se activa para ejecutarse debido a cierto evento externo o activador de temporización, donde, como se ilustra, la NN2 953 puede reanudarse después de que haya finalizado el evento o temporización en la NN3 955.

Vista general de aprendizaje automático

Un algoritmo de aprendizaje automático es un algoritmo que puede aprender basándose en un conjunto de datos. Ejemplos de algoritmos de aprendizaje automático pueden diseñarse para modelar abstracciones de alto nivel dentro de un conjunto de datos. Por ejemplo, pueden usarse algoritmos de reconocimiento de imágenes para determinar a cuál de varias categorías pertenece una entrada dada; los algoritmos de regresión pueden emitir un valor numérico dada una entrada; y pueden usarse los algoritmos de reconocimiento de patrones para generar texto traducido o para convertir texto en habla y/o reconocimiento de habla.

Un tipo ilustrativo de algoritmo de aprendizaje automático es una red neuronal. Hay muchos tipos de redes neuronales; un tipo sencillo de red neuronal es una red de realimentación prospectiva. Una red de realimentación prospectiva puede implementarse como un grafo acíclico en el que los nodos están dispuestos en capas. Típicamente, una topología de red de realimentación prospectiva incluye una capa de entrada y una capa de salida que están separadas por al menos una capa oculta. La capa oculta transforma la entrada recibida por la capa de entrada en una representación que es útil para generar la salida en la capa de salida. Los nodos de red están completamente conectados mediante bordes a los nodos en capas adyacentes, pero no hay bordes entre nodos dentro de cada capa. Los datos recibidos en los nodos de una capa de entrada de una red de realimentación prospectiva se propagan (es decir, "se realimentan prospectivamente") a los nodos de la capa de salida mediante una función de activación que calcula los estados de los nodos de cada capa sucesiva en la red basándose en coeficientes ("pesos") asociados, respectivamente, con cada uno de los bordes que conectan las capas. Dependiendo del modelo específico que se esté representando por el algoritmo que se está ejecutando, la salida del algoritmo de la red neuronal puede tomar diversas formas.

Antes de que pueda usarse un algoritmo de aprendizaje automático para modelar un problema particular, se entrena el algoritmo usando un conjunto de datos de entrenamiento. Entrenar una red neuronal implica seleccionar una topología de red, usar un conjunto de datos de entrenamiento que representa un problema que es modelado por la red, y ajustar los pesos hasta que el modelo de red rinde con un error mínimo para todas las instancias del conjunto de datos de entrenamiento. Por ejemplo, durante un proceso de entrenamiento de aprendizaje supervisado para una red neuronal, la salida producida por la red en respuesta a la entrada que representa una instancia en un conjunto de datos de entrenamiento se compara con la salida etiquetada "correcta" para esa instancia, se calcula una señal de error que representa la diferencia entre la salida y la salida etiquetada, y los pesos asociados con las conexiones se ajustan para minimizar ese error a medida que la señal de error se propaga hacia atrás a través de las capas de la red. La red se considera "entrenada" cuando se minimizan los errores para cada una de las salidas generadas a partir de las instancias del conjunto de datos de entrenamiento.

La precisión de un algoritmo de aprendizaje automático puede verse afectada significativamente por la calidad del conjunto de datos usado para entrenar el algoritmo. El proceso de entrenamiento puede ser computacionalmente intensivo y puede requerir una cantidad de tiempo significativa en un procesador de propósito general convencional. En consecuencia, se usa hardware de procesamiento paralelo para entrenar muchos tipos de algoritmos de aprendizaje automático. Esto es particularmente útil para optimizar el entrenamiento de redes neuronales, ya que los cálculos realizados al ajustar los coeficientes en las redes neuronales se prestan de manera natural a implementaciones paralelas. Específicamente, muchos algoritmos de aprendizaje automático y aplicaciones de software se han adaptado a hacer uso del hardware de procesamiento paralelo dentro de dispositivos de procesamiento de gráficos de propósito general.

La **Figura 10** es un diagrama generalizado de una pila de software de aprendizaje automático 1000. Una aplicación de aprendizaje automático 1002 puede configurarse para entrenar una red neuronal usando un conjunto de datos de entrenamiento o para usar una red neuronal profunda entrenada para implementar una inteligencia automática. La aplicación de aprendizaje automático 1002 puede incluir una funcionalidad de entrenamiento y de inferencia para una red neuronal y/o software especializado que puede usarse para entrenar una red neuronal antes del despliegue. La aplicación de aprendizaje automático 1002 puede implementar cualquier tipo de inteligencia automática incluyendo, pero sin limitación, reconocimiento de imágenes, mapeo y localización, navegación autónoma, síntesis de habla, formación de imágenes médicas o traducción de idioma.

Puede posibilitarse una aceleración de hardware para la aplicación de aprendizaje automático 1002 mediante una estructura de aprendizaje automático 1004. La estructura de aprendizaje automático 1004 puede proporcionar una biblioteca de primitivas de aprendizaje automático. Las primitivas de aprendizaje automático son operaciones básicas que se realizan comúnmente por algoritmos de aprendizaje automático. Sin la estructura de aprendizaje automático 1004, se requeriría que los desarrolladores de algoritmos de aprendizaje automático crearan y optimizaran la lógica computacional principal asociada con el algoritmo de aprendizaje automático, y volvieran a optimizar a continuación la lógica computacional a medida que se desarrollaran nuevos procesadores paralelos. En su lugar, la aplicación de aprendizaje automático puede estar configurada para realizar los cálculos necesarios usando las primitivas proporcionadas por la estructura de aprendizaje automático 1004. Las primitivas ilustrativas incluyen convoluciones de tipo tensor, funciones de activación y agrupamiento, que son operaciones computacionales que se realizan mientras se entrena una red neuronal convolucional (CNN). La estructura de aprendizaje automático 1004 puede proporcionar también primitivas para implementar subprogramas de álgebra lineal básicos realizados por muchos algoritmos de aprendizaje automático, tales como operaciones matriciales y vectoriales.

La estructura de aprendizaje automático 1004 puede procesar datos de entrada recibidos de la aplicación de aprendizaje automático 1002 y genera la entrada apropiada a una estructura de cálculo 1006. La estructura de cálculo 1006 puede abstraer las instrucciones subyacentes proporcionadas al controlador de GPGPU 1008 para posibilitar que la estructura de aprendizaje automático 1004 se aproveche de la aceleración de hardware mediante el hardware de GPGPU 1010 sin requerir que la estructura de aprendizaje automático 1004 tenga conocimiento íntimo de la arquitectura del hardware de GPGPU 1010. Adicionalmente, la estructura de cálculo 1006 puede posibilitar la

aceleración de hardware para la estructura de aprendizaje automático 1004 a través de una diversidad de tipos y generaciones del hardware de GPGPU 1010.

Aceleración de aprendizaje automático de GPGPU

La **Figura 11** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela 1100. En un ejemplo, la unidad de procesamiento de propósito general (GPGPU) 1100 puede estar configurada para ser particularmente eficiente al procesar el tipo de cargas de trabajo computacionales asociadas con el entrenamiento de las redes neuronales profundas. Adicionalmente, la GPGPU 1100 puede vincularse directamente a otras instancias de la GPGPU para crear una agrupación de múltiples GPU para mejorar la velocidad de entrenamiento para redes neuronales particularmente profundas.

La GPGPU 1100 incluye una interfaz de anfitrión 1102 para habilitar una conexión con un procesador de anfitrión. En una realización, la interfaz de anfitrión 1102 es una interfaz PCI Express. Sin embargo, la interfaz de anfitrión puede ser también una interfaz de comunicaciones o estructura de comunicaciones específica de proveedor. La GPGPU 1100 recibe comandos desde el procesador de anfitrión y usa un planificador global 1104 para distribuir hilos de ejecución asociados con estos comandos a un conjunto de agrupaciones de cálculo 1106A-H. Las agrupaciones de cálculo 1106A-H comparten una memoria caché 1108. La memoria caché 1108 puede servir como una caché de nivel superior para memorias caché dentro de las agrupaciones de cálculo 1106A-H.

La GPGPU 1100 incluye memoria 1114A-B acoplada con las agrupaciones de cálculo 1106A-H a través de un conjunto de controladores de memoria 1112A-B. En diversos ejemplos, la memoria 1114A-B puede incluir diversos tipos de dispositivos de memoria, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria gráfica de acceso aleatorio, tal como una memoria gráfica de acceso aleatorio síncrona (SGRAM), que incluye una memoria gráfica de doble tasa de datos (GDDR). En un ejemplo, las unidades de memoria 224A-N también pueden incluir memoria 3D apilada, que incluye, pero sin limitación, memoria de ancho de banda alto (HBM).

En un ejemplo, cada agrupación de cálculo GPLAB06A-H incluye un conjunto de multiprocesadores de gráficos, tal como el multiprocesador de gráficos 400 de la Figura 4A. Los multiprocesadores de gráficos de la agrupación de cálculo tienen múltiples tipos de unidades de lógica de enteros y de coma flotante que pueden realizar operaciones computacionales con un intervalo de precisiones que incluyen unas adecuadas para cálculos de aprendizaje automático. Por ejemplo, y en un ejemplo, al menos un subconjunto de las unidades de coma flotante en cada una de las agrupaciones de cálculo 1106A-H puede estar configurado para realizar operaciones de coma flotante de 16 bits o de 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante puede estar configurado para realizar operaciones de coma flotante de 64 bits.

Múltiples instancias de la GPGPU 1100 pueden configurarse para funcionar como una agrupación de cálculo. El mecanismo de comunicación usado por la agrupación de cálculo para la sincronización y el intercambio de datos varía a través de ejemplos. En un ejemplo, las múltiples instancias de la GPGPU 1100 se comunican a través de la interfaz de anfitrión 1102. En un ejemplo, la GPGPU 1100 incluye un concentrador de E/S 1108 que acopla la GPGPU 1100 con un enlace de GPU 1110 que posibilita una conexión directa a otras instancias de la GPGPU. En un ejemplo, el enlace de la GPU 1110 está acoplado a un puente de GPU a GPU especializado que posibilita la comunicación y sincronización entre múltiples instancias de la GPGPU 1100. En un ejemplo, el enlace de la GPU 1110 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En un ejemplo, las múltiples instancias de la GPGPU 1100 están ubicadas en sistemas de procesamiento de datos separados y se comunican mediante un dispositivo de red que es accesible mediante la interfaz de anfitrión 1102. En un ejemplo, el enlace de GPU 1110 puede estar configurado para posibilitar una conexión a un procesador de anfitrión además de o como una alternativa a la interfaz de anfitrión 1102.

Aunque la configuración ilustrada de la GPGPU 1100 puede configurarse para entrenar redes neuronales, un ejemplo proporciona una configuración alternativa de la GPGPU 1100, que puede configurarse para el despliegue dentro de una plataforma de inferenciación de alto rendimiento o de baja potencia. En una configuración de inferencia, la GPGPU 1100 incluye menos de las agrupaciones de cálculo 1106A-H con relación a la configuración de entrenamiento. Adicionalmente, una tecnología de memoria asociada con la memoria 1114A-B puede diferir entre las configuraciones de inferencia y de entrenamiento. En un ejemplo, la configuración de inferencia de la GPGPU 1100 puede soportar las instrucciones específicas de inferencia. Por ejemplo, una configuración de inferencia puede proporcionar soporte para una o más instrucciones de producto escalar de números enteros de 8 bits, que se usan comúnmente durante las operaciones de inferencia para redes neuronales desplegadas.

La **Figura 12** ilustra un sistema informático de múltiples GPU 1200. El sistema informático de múltiples GPU 1200 puede incluir un procesador 1202 acoplado a múltiples GPGPU 1206A-D mediante un conmutador de interfaz de anfitrión 1204. El conmutador de interfaz de anfitrión 1204, en un ejemplo, es un dispositivo de conmutador de PCI express que acopla el procesador 1202 a un bus de PCI express a través del que el procesador 1202 puede comunicarse con el conjunto de GPGPU 1206A-D. Cada una de las múltiples GPGPU 1206A-D puede ser una instancia de la GPGPU 1100 de la Figura 11. Las GPGPU 1206A-D pueden interconectarse mediante un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 1216. Los enlaces de GPU a GPU de alta velocidad se

pueden conectar a cada una de las GPGPU 1206A-D a través de un enlace de GPU especializado, tal como el enlace de GPU 1110 como en la Figura 11. Los enlaces de GPU de P2P 1216 posibilitan la comunicación directa entre cada una de las GPGPU 1206A-D sin requerir la comunicación a través del bus de interfaz de anfitrión a la que está conectado el procesador 1202. Con el tráfico de GPU a GPU dirigido a los enlaces de GPU de P2P, el bus de interfaz de anfitrión permanece disponible para el acceso a memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 1200, por ejemplo, mediante uno o más dispositivos de red. Aunque en el ejemplo ilustrado las GPGPU 1206A-D se conectan al procesador 1202 mediante el conmutador de interfaz de anfitrión 1204, en un ejemplo, el procesador 1202 incluye el soporte directo para los enlaces de GPU de P2P 1216 y puede conectarse directamente a las GPGPU 1206A-D.

Implementaciones de red neuronal de aprendizaje automático

La arquitectura informática descrita en el presente documento puede configurarse para realizar los tipos de procesamiento paralelo que son particularmente adecuados para entrenar y desplegar redes neuronales para un aprendizaje automático. Una red neuronal puede generalizarse como una red de funciones que tienen una relación de grafo. Como es bien conocido en la técnica, en el aprendizaje automático se usa una diversidad de tipos de implementaciones de red neuronal. Un tipo ilustrativo de red neuronal es la red de realimentación prospectiva, como se ha descrito previamente.

Un segundo tipo ilustrativo de red neuronal es la red neuronal convolucional (CNN). Una CNN es una red neuronal de realimentación prospectiva especializada para procesar datos que tienen una topología de tipo cuadrícula conocida, tales como datos de imagen. En consecuencia, las CNN se usan comúnmente para aplicaciones de reconocimiento de imágenes y de visión de cálculo, pero también pueden usarse para otros tipos de reconocimiento de patrones, tales como procesamiento de habla y de idioma. Los nodos en la capa de entrada de CNN están organizados en un conjunto de "filtros" (detectores de características inspirados por los campos receptivos encontrados en la retina), y la salida de cada conjunto de filtros se propaga a nodos en capas sucesivas de la red. Los cálculos para una CNN incluyen aplicar la operación matemática convolucional a cada filtro para producir la salida de ese filtro. La convolución es una clase especializada de operación matemática realizada por dos funciones para producir una tercera función que es una versión modificada de una de las dos funciones originales. En la terminología de red convolucional, la primera función a la convolución puede denominarse la entrada, mientras que la segunda función puede denominarse el núcleo de convolución. La salida puede denominarse mapa de características. Por ejemplo, la entrada a una capa de convolución puede ser una matriz multidimensional de datos que define los diversos componentes de color de una imagen de entrada. El núcleo de convolución puede ser una matriz multidimensional de parámetros, donde los parámetros están adaptados por el proceso de entrenamiento para la red neuronal.

Las redes neuronales recurrentes (RNN) son una familia de redes neuronales de realimentación prospectiva que incluyen conexiones de realimentación entre capas. Las RNN posibilitan el modelado de datos secuenciales compartiendo datos de parámetros a través de diferentes partes de la red neuronal. La arquitectura para una RNN incluye ciclos. Los ciclos representan la influencia de un valor presente de una variable sobre su propio valor en el futuro, ya que al menos una porción de los datos de salida de la RNN se usa como realimentación para procesar entrada posterior en una secuencia. Esta característica hace que las RNN sean particularmente útiles para el procesamiento de idioma debido a la naturaleza variable en la que pueden componerse los datos de idioma.

Las figuras descritas a continuación presentan redes de realimentación prospectiva, CNN y RNN ilustrativas, así como describen un proceso general para entregar y desplegar, respectivamente, cada uno de esos tipos de redes. Se entenderá que estas descripciones son ilustrativas y no limitantes en cuanto a cualquier ejemplo específico descrito en el presente documento y los conceptos ilustrados pueden aplicarse en general a redes neuronales profundas y técnicas de aprendizaje automático en general.

Las redes neuronales ilustrativas descritas anteriormente pueden usarse para realizar un aprendizaje profundo. El aprendizaje profundo es un aprendizaje automático que usa redes neuronales profundas. Las redes neuronales profundas usadas en el aprendizaje profundo son redes neuronales artificiales compuestas por múltiples capas ocultas, en contraposición a redes neuronales poco profundas que solo incluyen una única capa oculta. El entrenamiento de redes neuronales más profundas es, en general, más intensivo desde el punto de vista computacional. Sin embargo, las capas ocultas adicionales de la red habilitan un reconocimiento de patrones de múltiples etapas que da como resultado un error de salida reducido en relación con técnicas de aprendizaje automático poco profundo.

Las redes neuronales profundas usadas en aprendizaje automático incluyen típicamente una red de extremo frontal para realizar un reconocimiento de características acoplada a una red de extremo trasero que representa un modelo matemático que puede realizar operaciones (por ejemplo, clasificación de objetos, reconocimiento de habla, etc.) basándose en la representación de característica proporcionada en el modelo. Un aprendizaje profundo posibilita que se realice un aprendizaje automático sin requerir que se realice una ingeniería de características artesanal para el modelo. En su lugar, las redes neuronales profundas pueden aprender características basándose en una correlación o estructura estadística dentro de los datos de entrada. Las características aprendidas pueden proporcionarse a un modelo matemático que puede mapear características detectadas a una salida. El modelo matemático usado por la

red está especializado, en general, para la tarea específica a realizar, y se usarán diferentes modelos para realizar diferentes tareas.

Una vez que se ha estructurado la red neuronal, puede aplicarse un modelo de aprendizaje a la red para entrenar la red para realizar tareas específicas. El modelo de aprendizaje describe cómo ajustar los pesos dentro del modelo para reducir el error de salida de la red. La retropropagación de errores es un método común usado para entrenar redes neuronales. Se presenta un vector de entrada a la red para su procesamiento. La salida de la red se compara con la salida deseada usando una función de pérdida y se calcula un valor de error para cada una de las neuronas en la capa de salida. Los valores de error se retropropagan, a continuación, hasta que cada neurona tiene un valor de error asociado que representa aproximadamente su contribución a la salida original. La red puede aprender, a continuación, de esos errores usando un algoritmo, tal como el algoritmo de descenso de gradiente estocástico, para actualizar los pesos de la red neuronal.

Las Figuras 13A-B ilustran una red neuronal convolucional ilustrativa. La Figura 13A ilustra diversas capas dentro de una CNN. Como se muestra en la Figura 13A, una CNN ilustrativa usada para modelar el procesamiento de imagen puede recibir la entrada 1302 que describe los componentes rojo, verde y azul (RGB) de una imagen de entrada. La entrada 1302 puede procesarse por múltiples capas convolucionales (por ejemplo, la capa convolucional 1304, la capa convolucional 1306). La salida de las múltiples capas convolucionales puede procesarse opcionalmente por un conjunto de capas completamente conectadas 1308. Las neuronas en una capa completamente conectada tienen conexiones completas a todas las activaciones en la capa previa, como se ha descrito previamente para una red de realimentación prospectiva. La salida de las capas completamente conectadas 1308 puede usarse para generar un resultado de salida a partir de la red. Las activaciones dentro de las capas completamente conectadas 1308 pueden calcularse usando una multiplicación matricial en lugar de una convolución. No todas las implementaciones de CNN hacen uso de capas completamente conectadas DPLA08. Por ejemplo, en algunas implementaciones, la capa convolucional 1306 puede generar la salida de la CNN.

Las capas convolucionales están conectadas de manera dispersa, lo que difiere de la configuración de red neuronal tradicional encontrada en las capas completamente conectadas 1308. Las capas de red neuronal tradicionales están completamente conectadas, de manera que cada unidad de salida interacciona con cada unidad de entrada. Sin embargo, las capas convolucionales están conectadas de manera dispersa debido a que se introduce la salida de la convolución de un campo (en lugar del valor de estado respectivo de cada uno de los nodos en el campo) en los nodos de la capa subsiguiente, como se ha ilustrado. Los núcleos asociados con las capas convolucionales realizan operaciones de convolución, cuya salida se envía a la capa siguiente. La reducción de dimensionalidad realizada dentro de las capas convolucionales es un aspecto que posibilita que la CNN realice un ajuste a escala para procesar imágenes grandes.

La **Figura 13B** ilustra etapas de cálculo ilustrativas dentro de una capa convolucional de una CNN. La entrada a una capa convolucional 1312 de una CNN puede procesarse en tres etapas de una capa convolucional 1314. Las tres etapas pueden incluir una etapa de convolución 1316, una etapa de detector 1318 y una etapa de agrupamiento 1320. La capa convolucional 1314 puede emitir a continuación datos a una capa convolucional sucesiva. La capa convolucional final de la red puede generar datos de mapeo de características de salida o proporcionar entrada a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

En la etapa de convolución 1316 se realizan varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La etapa de convolución 1316 puede incluir una transformación afín, que es cualquier transformación que pueda especificarse como una transformación lineal más una traslación. Las transformaciones afines incluyen rotaciones, traslaciones, escalamiento y combinaciones de estas transformaciones. La etapa de convolución calcula la salida de funciones (por ejemplo, neuronas) que están conectadas a regiones específicas en la entrada, que puede determinarse como la región local asociada con la neurona. Las neuronas calculan un producto escalar entre los pesos de las neuronas y la región en la entrada local a la que están conectadas las neuronas. La salida de la etapa de convolución 1316 define un conjunto de activaciones lineales que se procesan por etapas sucesivas de la capa convolucional 1314.

Las activaciones lineales pueden procesarse por una etapa de detector 1318. En la etapa de detector 1318, cada activación lineal se procesa por una función de activación no lineal. La función de activación no lineal aumenta las propiedades no lineales de la red global sin afectar a los campos receptivos de la capa de convolución. Pueden usarse varios tipos de funciones de activación no lineal. Un tipo particular es la unidad lineal rectificada (ReLU), que usa una función de activación definida como $f(x) = \max(0, x)$, de manera que se fija un umbral de cero para la activación.

La etapa de agrupamiento 1320 usa una función de agrupación que sustituye la salida de la capa convolucional 1306 con un sumario estadístico de las salidas cercanas. La función de agrupamiento puede usarse para introducir la invarianza de traslación en la red neuronal, de manera que traslaciones pequeñas en la entrada no cambian las salidas agrupadas. La invarianza a la traslación local puede ser útil en situaciones donde la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Pueden usarse diversos tipos de funciones de agrupamiento durante la etapa de agrupamiento 1320, incluyendo agrupamiento máximo, agrupamiento promedio y agrupamiento de norma 12. Adicionalmente, algunas implementaciones de CNN no incluyen

una etapa de agrupación. En su lugar, tales implementaciones sustituyen una etapa de convolución adicional que tiene un paso aumentado en relación con etapas de convolución previas.

La salida de la capa convolucional 1314 puede procesarse a continuación por la siguiente capa 1322. La siguiente capa 1322 puede ser una capa convolucional adicional o una de las capas completamente conectadas 1308. Por ejemplo, la primera capa convolucional 1304 de la **Figura 13A** puede enviarse a la segunda capa convolucional 1306, mientras que la segunda capa convolucional puede enviarse a una primera capa de las capas completamente conectadas 1308.

La **Figura 14** ilustra una red neuronal recurrente ilustrativa 1400. En una red neuronal recurrente (RNN), el estado anterior de la red influye en la salida del estado actual de la red. Las RNN pueden crearse en una diversidad de maneras usando una diversidad de funciones. El uso de las RNN en general gira entorno al uso de modelos matemáticos para predecir el futuro basándose en una secuencia de entradas anterior. Por ejemplo, puede usarse una RNN para realizar modelado de idioma estadístico para predecir una palabra próxima dada en una secuencia de palabras anterior. La RNN 1400 ilustrada puede describirse como una que tiene una capa de entrada 1402 que recibe un vector de entrada, capas ocultas 1404 para implementar una función recurrente, un mecanismo de realimentación 1405 para habilitar una 'memoria' de estados previos y una capa de salida 1406 para emitir un resultado. La RNN 1400 opera basándose en etapas de tiempo. El estado de la RNN en un paso de tiempo dado se ve influenciado basándose en el paso de tiempo anterior mediante el mecanismo de realimentación 1405. Para una etapa de tiempo dada, se define el estado de las capas ocultas 1404 por el estado anterior y la entrada en la etapa de tiempo actual. Puede procesarse una entrada inicial (x_1) en una primera etapa de tiempo por la capa oculta 1404. Puede procesarse una segunda entrada (x_2) por la capa oculta 1404 usando información de estado que se determina durante el procesamiento de la entrada inicial (x_1). Un estado dado puede calcularse como $s_t = f(Ux_t + Ws_{t-1})$, donde U y W son matrices de parámetros. La función f es, en general, una no linealidad, tal como la función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x) = \text{máx}(0, x)$. Sin embargo, la función matemática específica usada en las capas ocultas 1404 puede variar dependiendo de los detalles de la implementación específica de la RNN 1400.

Además de las redes CNN y RNN básicas descritas, pueden habilitarse variaciones en esas redes. Una variante de RNN ilustrativa es la RNN de memoria a corto plazo larga (LSTM). Las RNN de LSTM pueden aprender dependencias a largo plazo que pueden ser necesarias para procesar secuencias de idioma más largas. Una variante en la CNN es una red de creencia profunda convolucional, que tiene una estructura similar a una CNN y se entrena de una manera similar a una red de creencia profunda. Una red de creencia profunda (DBN) es una red neuronal generativa que está compuesta de múltiples capas de variables estocásticas (aleatorias). Las DBN pueden entrenarse capa a capa usando aprendizaje no supervisado voraz. Los pesos aprendidos de la DBN pueden usarse a continuación para proporcionar redes neuronales de preentrenamiento determinando un conjunto inicial óptimo de pesos para la red neuronal.

La **Figura 15** ilustra el entrenamiento y despliegue de una red neuronal profunda. Una vez que se ha estructurado una red dada para una tarea, se entrena la red neuronal usando un conjunto de datos de entrenamiento 1502. Se han desarrollado diversas estructuras de entrenamiento 1504 para posibilitar la aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura de aprendizaje automático 1004 de la Figura 10 puede configurarse como una estructura de entrenamiento 1004. La estructura de entrenamiento 1004 puede engancharse a una red neuronal no entrenada 1506 y posibilitar que la red neuronal no entrenada se entrene usando los recursos de procesamiento paralelo descritos en el presente documento para generar una red neuronal entrenada 1508.

Para iniciar el proceso de entrenamiento, pueden elegirse los pesos iniciales aleatoriamente o mediante entrenamiento previo usando una red de creencia profunda. El ciclo de entrenamiento puede realizarse a continuación de una manera supervisada o no supervisada.

El aprendizaje supervisado es un método de aprendizaje en el que se realiza un entrenamiento como una operación mediada, tal como cuando el conjunto de datos de entrenamiento 1502 incluye una entrada emparejada con la salida deseada para la entrada, o donde el conjunto de datos de entrenamiento incluye una entrada que tiene una salida conocida, y la salida de la red neuronal se califica manualmente. La red procesa las entradas y compara las salidas resultantes contra un conjunto de salidas esperadas o deseadas. Los errores se retropropagan a continuación a través del sistema. La estructura de entrenamiento 1504 puede ajustarse para ajustar los pesos que controlan la red neuronal no entrenada 1506. La estructura de entrenamiento 1504 puede proporcionar herramientas para monitorizar cómo está convergiendo de bien la red neuronal no entrenada 1506 hacia un modelo adecuado para generar respuestas correctas basándose en datos de entrada conocidos. El proceso de entrenamiento tiene lugar repetidamente a medida que se ajustan los pesos de la red para perfeccionar la salida generada por la red neuronal. El proceso de entrenamiento puede continuar hasta que la red neuronal alcanza una precisión estadísticamente deseada asociada con una red neuronal entrenada 1508. La red neuronal entrenada 1508 puede desplegarse a continuación para implementar cualquier número de operaciones de aprendizaje automático.

El aprendizaje no supervisado es un método de aprendizaje en el que la red intenta entrenarse a sí misma usando datos no etiquetados. Por lo tanto, para un aprendizaje no supervisado, el conjunto de datos de entrenamiento 1502 incluirá datos de entrada sin ningún dato de salida asociado. La red neuronal no entrenada 1506 puede aprender agrupamientos dentro de la entrada no etiquetada y puede determinar cómo las entradas individuales están

relacionadas con el conjunto de datos global. El entrenamiento no supervisado puede usarse para generar un mapa de autoorganización, que es un tipo de red neuronal entrenada 1507 que puede realizar operaciones útiles en cuanto a la reducción de la dimensionalidad de los datos. El entrenamiento no supervisado también puede usarse para realizar una detección de anomalías, lo que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

Pueden emplearse también variaciones en el entrenamiento supervisado y no supervisado. El aprendizaje semisupervisado es una técnica en la que el conjunto de datos de entrenamiento 1502 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que se usan continuamente datos de entrada para entrenar adicionalmente el modelo. El aprendizaje incremental habilita la adaptación de la red neuronal entrenada 1508 a los datos nuevos 1512 sin olvidar el conocimiento inculcado dentro de la red durante el entrenamiento inicial.

Ya sea supervisado o no supervisado, el proceso de entrenamiento para redes neuronales particularmente profundas puede ser demasiado intensivo desde el punto de vista computacional para un único nodo de cálculo. En lugar de usar un único nodo de cálculo, puede usarse una red distribuida de nodos computacionales para acelerar el proceso de entrenamiento.

La **Figura 16** es un diagrama de bloques que ilustra el aprendizaje distribuido. El aprendizaje distribuido es un modelo de entrenamiento que usa múltiples nodos informáticos distribuidos para realizar un entrenamiento supervisado o no supervisado de una red neuronal. Cada uno de los nodos computacionales distribuidos puede incluir uno o más procesadores de anfitrión y uno o más de los nodos de procesamiento de propósito general, tales como la unidad de procesamiento de gráficos de propósito general altamente paralela 1100 como en la Figura 1100. Como se ha ilustrado, un aprendizaje distribuido puede realizarse con el paralelismo de modelo 1602, el paralelismo de datos 1604 o una combinación del paralelismo de modelo y de datos 1604.

En el paralelismo de modelo 1602, diferentes nodos computacionales en un sistema distribuido pueden realizar cálculos de entrenamiento para diferentes partes de una única red. Por ejemplo, cada capa de una red neuronal puede entrenarse por un nodo de procesamiento diferente del sistema distribuido. Los beneficios del paralelismo de modelo incluyen la capacidad de ajustar a escala a modelos particularmente grandes. La división de los cálculos asociados con diferentes capas de la red neuronal habilita el entrenamiento de redes neuronales muy grandes en las que los pesos de todas las capas no cabrían en la memoria de un único nodo computacional. En algunos casos, el paralelismo de modelo puede ser particularmente útil al realizar entrenamiento no supervisado de redes neuronales grandes.

En el paralelismo de datos 1604, los diferentes nodos de la red distribuida tienen una instancia completa del modelo y cada nodo recibe una porción diferente de los datos. Los resultados de los diferentes nodos, a continuación, se combinan. Aunque son posibles diferentes enfoques al paralelismo de datos, los enfoques de entrenamiento de datos paralelos requieren, todos ellos, una técnica de combinación de resultados y de sincronización de los parámetros de modelo entre cada nodo. Los enfoques ilustrativos de la combinación de datos incluyen el promediado de parámetros y el paralelismo de datos basado en actualizaciones. El promediado de parámetros entrena cada nodo en un subconjunto de los datos de entrenamiento y establece los parámetros globales (por ejemplo, pesos, desvíos) al promedio de los parámetros de cada nodo. El promediado de parámetros usa un servidor de parámetros central que mantiene los datos de parámetro. El paralelismo de datos basado en actualizaciones es similar al promediado de parámetros excepto en que, en lugar de transferir parámetros desde los nodos al servidor de parámetros, se transfieren las actualizaciones al modelo. Adicionalmente, el paralelismo de datos basado en actualizaciones puede realizarse de una manera descentralizada, donde las actualizaciones se comprimen y se transfieren entre nodos.

El paralelismo de modelo y de datos combinado 1606 puede implementarse, por ejemplo, en un sistema distribuido en el que cada nodo computacional incluye múltiples GPU. Cada nodo puede tener una instancia completa del modelo con GPU separadas dentro de cada nodo que se usan para entrenar diferentes porciones del modelo.

El entrenamiento distribuido ha aumentado la sobrecarga en relación con el entrenamiento en una única máquina. Sin embargo, cada uno de los procesadores paralelos y las GPGPU descritos en el presente documento pueden implementar diversas técnicas para reducir la sobrecarga del entrenamiento distribuido, incluyendo técnicas para posibilitar una transferencia de datos de GPU a GPU de alto ancho de banda y una sincronización de datos remota acelerada.

Aplicaciones de aprendizaje automático ilustrativas

El aprendizaje automático puede aplicarse para resolver una diversidad de problemas tecnológicos, incluyendo, pero sin limitación, visión informática, conducción y navegación autónoma, reconocimiento del habla y procesamiento del idioma. La visión informática ha sido tradicionalmente una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Las aplicaciones de visión informática varían desde la reproducción de capacidades visuales humanas, tales como el reconocimiento de caras, hasta la creación de nuevas categorías de capacidades visuales. Por ejemplo, las aplicaciones de visión informática pueden configurarse para reconocer ondas de sonido de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesador

paralelo habilita el entrenamiento de aplicaciones de visión informática usando un conjunto de datos de entrenamiento significativamente mayor que el previamente factible y habilita el desarrollo de sistemas de inferencia usando procesadores paralelos de baja potencia.

El aprendizaje automático acelerado por procesador paralelo tiene aplicaciones de conducción autónoma que incluyen reconocimiento de señales de carretera y de carril, evitación de obstáculos, navegación y control de conducción. Las técnicas de aprendizaje automático aceleradas pueden usarse para entrenar modelos de conducción basándose en conjuntos de datos que definen las respuestas apropiadas a una entrada de entrenamiento específica. Los procesadores paralelos descritos en el presente documento pueden habilitar el entrenamiento rápido de las redes neuronales cada vez más complejas usadas para soluciones de conducción autónoma y posibilita el despliegue de procesadores de inferencia de baja potencia en una plataforma móvil adecuada para su integración en vehículos autónomos.

Las redes neuronales profundas aceleradas por procesador paralelo han posibilitado enfoques de aprendizaje automático para un reconocimiento de habla automático (ASR). El ASR incluye la creación de una función que, dada una secuencia acústica de entrada, calcula la secuencia lingüística más probable. El aprendizaje automático acelerado usando redes neuronales profundas ha posibilitado la sustitución de los modelos ocultos de Markov (HMM) y los modelos de mezcla gaussiana (GMM) previamente usados para el ASR.

El aprendizaje automático acelerado por procesador paralelo puede usarse también para acelerar el procesamiento de lenguaje natural. Los procedimientos de aprendizaje automático pueden hacer uso de algoritmos de inferencia estadística para producir modelos que son robustos ante una entrada errónea o no familiar. Las aplicaciones de procesador de lenguaje natural ilustrativas incluyen la traducción mecánica automática entre idiomas humanos.

Las plataformas de procesamiento paralelo usadas para aprendizaje automático pueden dividirse en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento son, en general, altamente paralelas e incluyen optimizaciones para acelerar el entrenamiento de múltiples GPU y un único nodo y el entrenamiento de múltiples nodos y múltiples GPU. Los procesadores paralelos ilustrativos adecuados para entrenamiento incluyen la unidad de procesamiento de gráficos de propósito general 1100 de la Figura 1100 y el sistema informático de múltiples GPU 1200 de la Figura 1200. Por el contrario, las plataformas de aprendizaje automático desplegadas incluyen, en general, procesadores paralelos de potencia inferior adecuados para su uso en productos tales como cámaras, robots autónomos y vehículos autónomos.

La **Figura 17** ilustra un sistema de inferencia ilustrativo en un chip (SOC) 1700 adecuado para realizar inferencia usando un modelo entrenado. El SOC 1700 puede integrar componentes de procesamiento que incluyen un procesador de medios 1702, un procesador de visión 1704, una GPGPU 1706 y un procesador de múltiples núcleos 1708. El SOC 1700 puede incluir adicionalmente una memoria en chip 1705 que puede posibilitar una agrupación de datos en chip compartida que es accesible por cada uno de los componentes de procesamiento. Los componentes de procesamiento pueden optimizarse para la operación de baja potencia para posibilitar el despliegue a una diversidad de plataformas de aprendizaje automático, que incluyen vehículos autónomos y robots autónomos. Por ejemplo, puede usarse una implementación del SOC 1700 como una porción del sistema de control principal para un vehículo autónomo. Donde el SOC 1700 está configurado para su uso en vehículos autónomos, el SOC está diseñado y configurado para su cumplimiento con las normas de seguridad funcionales relevantes de la jurisdicción de despliegue.

Durante la operación, el procesador de medios 1702 y el procesador de visión 1704 pueden funcionar en conjunto para acelerar operaciones de visión informática. El procesador de medios 1702 puede posibilitar la decodificación de baja latencia de múltiples flujos de vídeo de alta resolución (por ejemplo, 4K, 8K). Los flujos de vídeo decodificados pueden escribirse en una memoria intermedia en la memoria en el chip 1705. El procesador de visión 1704 puede a continuación analizar el vídeo decodificado y realizar de manera preliminar las operaciones de procesamiento en los fotogramas del vídeo decodificado en preparación del procesamiento de los fotogramas usando un modelo de reconocimiento de imagen entrenado. Por ejemplo, el procesador de visión 1704 puede acelerar las operaciones convolucionales para una CNN que se usa para realizar el reconocimiento de imagen en los datos de vídeo de alta resolución, mientras se realizan cálculos de modelo de extremo trasero por la GPGPU 1706.

El procesador de múltiples núcleos 1708 puede incluir lógica de control para ayudar con la secuenciación y la sincronización de transferencias de datos y operaciones de memoria compartida realizadas por el procesador de medios 1702 y el procesador de visión 1704. El procesador de múltiples núcleos 1708 puede funcionar también como un procesador de aplicación para ejecutar aplicaciones de software que pueden hacer uso de la capacidad de cálculo de inferencia de la GPGPU 1706. Por ejemplo, puede implementarse al menos una porción de la lógica de navegación y de conducción en software que se ejecuta en el procesador de múltiples núcleos 1708. Tal software puede emitir directamente cargas de trabajo computacionales a la GPGPU 1706 o pueden emitirse las cargas de trabajo computacionales al procesador de múltiples núcleos 1708, que puede descargar al menos una porción de estas operaciones a la GPGPU 1706.

La GPGPU 1706 puede incluir agrupaciones de cálculo, tal como una configuración de baja potencia de las agrupaciones de cálculo 1106A-1106H dentro de la unidad de procesamiento de gráficos de propósito general

altamente paralela 1100. Las agrupaciones de cálculo dentro de la GPGPU 1706 pueden soportar instrucciones que están optimizadas específicamente para realizar cálculos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1706 puede soportar instrucciones para realizar cálculos de baja precisión tales como operaciones vectoriales de números enteros de 8 bits y 4 bits.

Vista general del sistema II

La **Figura 18** es un diagrama de bloques de un sistema de procesamiento 1800. En diversas realizaciones, el sistema 1800 incluye uno o más procesadores 1802 y uno o más procesadores de gráficos 1808, y puede ser un sistema de sobremesa de procesador único, un sistema de estación de trabajo de multiprocesador o un sistema de servidor que tiene un gran número de procesadores 1802 o núcleos de procesador 1807. En un ejemplo, el sistema 1800 es una plataforma de procesamiento incorporada dentro de un circuito integrado de sistema en un chip (SoC) para su uso en dispositivos móviles, portátiles o integrados.

Un ejemplo del sistema 1800 puede incluir o incorporarse dentro de: una plataforma de juegos basada en servidor, una consola de juegos, que incluye una consola de juegos y medios, una consola de juegos móvil, una consola de juegos portátil o una consola de juegos en línea. En algunos ejemplos, el sistema 1800 es un teléfono móvil, teléfono inteligente, dispositivo informático de tableta o dispositivo de internet móvil. El sistema de procesamiento de datos 1800 puede incluir también, estar acoplado con, o estar integrado dentro de un dispositivo llevable, tal como un dispositivo llevable de reloj inteligente, dispositivo de gafas inteligentes, dispositivo de realidad aumentada o dispositivo de realidad virtual. En algunos ejemplos, el sistema de procesamiento de datos 1800 es un televisor o dispositivo decodificador que tiene uno o más procesadores 1802 y una interfaz gráfica generada por uno o más procesadores de gráficos 1808.

En algunos ejemplos, cada uno del uno o más procesadores 1802 incluye uno o más núcleos de procesador 1807 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para software de usuario y sistema. En algunos ejemplos, cada uno del uno o más núcleos de procesador 1807 está configurado para procesar un conjunto de instrucciones 1809 específico. En algunos ejemplos, el conjunto de instrucciones 1809 puede facilitar el cálculo de conjunto de instrucciones complejo (CISC), el cálculo de conjunto de instrucciones reducido (RISC) o el cálculo mediante una palabra de instrucción muy larga (VLIW). Cada uno de múltiples núcleos de procesador 1807 puede procesar un conjunto de instrucciones diferente 1809, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 1807 puede incluir también otros dispositivos de procesamiento, tal como un procesador de señales digitales (DSP).

En algunos ejemplos, el procesador 1802 incluye la memoria caché 1804. Dependiendo de la arquitectura, el procesador 1802 puede tener una única caché interna o múltiples niveles de caché interna. En algunos ejemplos, la memoria caché se comparte entre diversos componentes del procesador 1802. En algunos ejemplos, el procesador 1802 también usa una memoria caché externa (por ejemplo, una memoria caché de nivel-3 (L3) o una memoria caché de último nivel (LLC)) (no mostrada), que puede estar compartida entre los núcleos de procesador 1807 usando técnicas de coherencia de caché conocidas. Un archivo de registro 1806 está incluido adicionalmente en el procesador 1802 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de coma flotante, registros de estado y un registro de puntero de instrucción). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos al diseño del procesador 1802.

En algunos ejemplos, el procesador 1802 está acoplado con un bus de procesador 1810 para transmitir señales de comunicación tales como señales de dirección, de datos o de control entre el procesador 1802 y otros componentes en el sistema 1800. En un ejemplo, el sistema 1800 usa una arquitectura de sistema de 'concentrador' ilustrativa, que incluye un concentrador de controlador de memoria 1816 y un concentrador de controlador de entrada y salida (E/S) 1830. Un concentrador de controlador de memoria 1816 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 1800, mientras que un concentrador de controlador de E/S (ICH) 1830 proporciona conexiones a los dispositivos de E/S mediante un bus de E/S local. En un ejemplo, la lógica del concentrador de controlador de memoria 1816 está integrada dentro del procesador.

El dispositivo de memoria 1820 puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria de acceso aleatorio estática (SRAM), dispositivo de memoria flash, dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para dar servicio como una memoria de proceso. En un ejemplo, el dispositivo de memoria 1820 puede operar como memoria de sistema para el sistema 1800, para almacenar datos 1822 e instrucciones 1821 para su uso cuando el uno o más procesadores 1802 ejecutan una aplicación o proceso. El concentrador de controlador de memoria 1816 también se acopla con un procesador gráfico externo 1812 opcional, que puede comunicarse con el uno o más procesadores de gráficos 1808 en los procesadores 1802 para realizar operaciones de gráficos y de medios.

En algunos ejemplos, el ICH 1830 posibilita que los periféricos se conecten al dispositivo de memoria 1820 y al procesador 1802 mediante un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, pero sin limitación, un controlador de audio 1846, una interfaz de firmware 1828, un transceptor inalámbrico 1826 (por ejemplo, Wi-Fi,

Bluetooth), un dispositivo de almacenamiento de datos 1824 (por ejemplo, unidad de disco duro, memoria flash, etc.), y un controlador de E/S heredado 1840 para acoplar dispositivos heredados (por ejemplo, dispositivos de sistema personal 2 (PS/2)) al sistema. Uno o más controladores de bus serie universal (USB) 1842 conectan dispositivos de entrada, tales como las combinaciones de teclado y ratón 1844. También se puede acoplar un controlador de red 1834 al ICH 1830. En algunos ejemplos, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de procesador 1810. Se apreciará que el sistema 1800 mostrado es ilustrativo y no limitante, ya que pueden usarse otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por ejemplo, el concentrador de controlador de E/S 1830 puede integrarse dentro del uno o más procesadores 1802, o el concentrador de controlador de memoria 1816 y el concentrador de controlador de E/S 1830 pueden estar integrados en un procesador de gráficos externo discreto, tal como el procesador de gráficos externo 1812.

La **Figura 19** es un diagrama de bloques de un procesador 1900 que tiene uno o más núcleos de procesador 1902A-1902N, un controlador de memoria integrado 1914 y un procesador de gráficos integrado 1908. Los elementos de la **Figura 19** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura del presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte del presente documento, pero no se limitan a ello. El procesador 1900 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 1902N representado por los recuadros de línea discontinua. Cada uno de los núcleos de procesador 1902A-1902N incluye una o más unidades de caché internas 1904A-1904N. En algunos ejemplos, cada núcleo de procesador también tiene acceso a una o más unidades de caché compartidas 1906.

Las unidades de caché internas 1904A-1904N y las unidades de caché compartidas 1906 representan una jerarquía de memoria caché dentro del procesador 1900. La jerarquía de memoria caché puede incluir al menos un nivel de caché de instrucciones y de datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel medio compartida, tal como una caché de Nivel 2 (L2), de Nivel 3 (L3), de Nivel 4 (L4) o de otros niveles, donde el nivel más alto de caché antes de la memoria externa se clasifica como LLC. En algunos ejemplos, la lógica de coherencia de caché mantiene la coherencia entre las diversas unidades de caché 1906 y 1904A-1904N.

En algunos ejemplos, el procesador 1900 puede incluir también un conjunto de una o más unidades de controlador de bus 1916 y un núcleo de agente de sistema 1910. La una o más unidades de controlador de bus 1916 gestionan un conjunto de buses periféricos, tal como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 1910 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunos ejemplos, el núcleo de agente de sistema 1910 incluye uno o más controladores de memoria integrados 1914 para gestionar el acceso a diversos dispositivos de memoria externos (no mostrados).

En algunos ejemplos, uno o más de los núcleos de procesador 1902A-1902N incluyen el soporte para múltiples hilos simultáneos. En un ejemplo de este tipo, el núcleo de agente de sistema 1910 incluye componentes para coordinar y operar los núcleos 1902A-1902N durante el procesamiento de múltiples hilos. El núcleo de agente de sistema 1910 puede incluir adicionalmente una unidad de control de potencia (PCU), que incluye una lógica y componentes para regular el estado de potencia de los núcleos de procesador 1902A-1902N y el procesador de gráficos 1908.

En algunos ejemplos, el procesador 1900 incluye adicionalmente el procesador de gráficos 1908 para ejecutar las operaciones de procesamiento de gráficos. En algunos ejemplos, el procesador de gráficos 1908 se acopla con el conjunto de unidades de caché compartida 1906 y el núcleo de agente de sistema 1910, incluyendo el uno o más controladores de memoria integrados 1914. En algunos ejemplos, un controlador de visualización 1911 está acoplado con el procesador de gráficos 1908 para controlar la salida de procesador de gráficos a una o más pantallas acopladas. En algunos ejemplos, el controlador de visualización 1911 puede ser un módulo separado acoplado con el procesador de gráficos mediante al menos una interconexión, o puede estar integrado dentro del procesador de gráficos 1908 o el núcleo de agente de sistema 1910.

En algunos ejemplos, se usa una unidad de interconexión basada en anillo 1912 para acoplar los componentes internos del procesador 1900. Sin embargo, se puede usar una unidad de interconexión alternativa, tal como una interconexión punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en la técnica. En algunos ejemplos, el procesador de gráficos 1908 se acopla con el anillo de interconexión 1912 mediante un enlace de E/S 1913.

El enlace de E/S 1913 ilustrativo representa al menos una de múltiples variedades de interconexiones de E/S, incluyendo una interconexión de E/S en paquete que facilita la comunicación entre diversos componentes de procesador y un módulo de memoria integrado de alto rendimiento 1918, tal como un módulo de eDRAM. En algunos ejemplos, cada uno de los núcleos de procesador 1902-1902N y del procesador de gráficos 1908 usan módulos de memoria integrados 1918 como una caché de último nivel compartida.

En algunos ejemplos, los núcleos de procesador 1902A-1902N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otro ejemplo, los núcleos de procesador 1902A-1902N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 1902A-N ejecutan un primer conjunto de instrucciones, mientras que al menos uno de los otros núcleos ejecuta un subconjunto

del primer conjunto de instrucciones o un conjunto de instrucciones diferente. En un ejemplo, los núcleos de procesador 1902A-1902N son heterogéneos en términos de microarquitectura, donde uno o más núcleos que tienen un consumo de potencia relativamente más alto se acoplan con uno o más núcleos de potencia que tienen un consumo de potencia más bajo. Adicionalmente, el procesador 1900 puede implementarse en uno o más chips o como un circuito de SoC integrado que tiene los componentes ilustrados, además de otros componentes.

La **Figura 20** es un diagrama de bloques de un procesador de gráficos 2000, que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento. En algunos ejemplos, el procesador de gráficos se comunica a través de una interfaz de E/S de memoria mapeada a registros en el procesador gráfico y con comandos colocados en la memoria de procesador. En algunos ejemplos, el procesador de gráficos 2000 incluye una interfaz de memoria 2014 para acceder a memoria. La interfaz de memoria 2014 puede ser una interfaz a memoria local, a una o más cachés internas, a una o más cachés externas compartidas y/o a memoria de sistema.

En algunos ejemplos, el procesador de gráficos 2000 también incluye un controlador de visualización 2002 para controlar los datos de salida de visualización a un dispositivo de visualización 2020. El controlador de visualización 2002 incluye hardware para uno o más planos de superposición para la visualización y la composición de múltiples capas de vídeo o elementos de interfaz de usuario. En algunos ejemplos, el procesador de gráficos 2000 incluye un motor de códec de vídeo 2006 para codificar, decodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, que incluyen, pero sin limitación formatos del Grupo de Expertos de Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como de la Sociedad de Ingenieros de Imágenes en Movimiento y Televisión (SMPTE) 421M/VC-1 y formatos del Grupo Mixto de Expertos en Fotografía (JPEG), tal como los formatos JPEG y Motion JPEG (MJPEG).

En algunos ejemplos, el procesador de gráficos 2000 incluye un motor de transferencia de imágenes en bloque (BLIT) 2004 para realizar operaciones de rasterizador bidimensionales (2D), incluyendo, por ejemplo, transferencias de bloque de límite de bits. Sin embargo, en un ejemplo, las operaciones de gráficos 2D se realizan usando uno o más componentes del motor de procesamiento de gráficos (GPE) 2010. En algunos ejemplos, el motor de procesamiento de gráficos 2010 es un motor de cálculo para realizar operaciones de gráficos, que incluyen operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunos ejemplos, el GPE 2010 incluye una canalización 3D 2012 para realizar operaciones en 3D, tales como representar imágenes y escenas tridimensionales usando funciones de procesamiento que actúan sobre formas de primitivas en 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización 3D 2012 incluye elementos de función programable y fija que realizan diversas tareas dentro del elemento y/o generan hilos de ejecución en un subsistema 3D/de medios 2015. Aunque puede usarse la canalización 3D 2012 para realizar operaciones de medios, un ejemplo del GPE 2010 también incluye una canalización de medios 2016 que se usa específicamente para realizar operaciones de medios, tales como post procesamiento de vídeo y mejora de imagen.

En algunos ejemplos, la canalización de medios 2016 incluye unidades de lógica programable o de función fija para realizar una o más operaciones de medios especializadas, tales como aceleración de decodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar o en nombre del motor de códec de vídeo 2006. En algunos ejemplos, la canalización de medios 2016 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema 3D/de medios 2015. Los hilos generados realizan cálculos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema 3D/de medios 2015.

En algunos ejemplos, el subsistema 3D/de medios 2015 incluye una lógica para ejecutar hilos generados por la canalización 3D 2012 y la canalización de medios 2016. En un ejemplo, las canalizaciones envían solicitudes de ejecución de hilos al subsistema de 3D/de medios 2015, que incluye una lógica de despacho de hilos para arbitrar y despachar las diversas solicitudes a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos de medios y 3D. En algunos ejemplos, el subsistema 3D/de medios 2015 incluye una o más cachés internas para instrucciones y datos de hilo. En algunos ejemplos, el subsistema también incluye memoria compartida, que incluye registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Procesamiento 3D/de medios

La **Figura 21** es un diagrama de bloques de un motor de procesamiento de gráficos 2110 de un procesador de gráficos. En un ejemplo, el motor de procesamiento de gráficos (GPE) 2110 es una versión del GPE 2010 mostrado en la **Figura 20.** Elementos de la **Figura 21** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura del presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte del presente documento, pero no se limitan a ello. Por ejemplo, se ilustra la canalización 3D 2012 y la canalización de medios 2016 de la **Figura 20**. La canalización de medios 2016 es opcional en algunos ejemplos del

GPE 2110 y puede no estar explícitamente incluida dentro del GPE 2110. Por ejemplo y, en al menos un ejemplo, un procesador de medios y/o de imágenes separado está acoplado al GPE 2110.

En algunos ejemplos, el GPE 2110 se acopla con o incluye un emisor por flujo continuo de comandos 2103, que proporciona un flujo de comandos a la canalización de 3D 2012 y/o a las canalizaciones de medios 2016. En algunos ejemplos, el emisor por flujo continuo de comandos 2103 está acoplado con memoria, que puede ser memoria de sistema, o una o más de memoria de caché interna y memoria de caché compartida. En algunos ejemplos, el emisor por flujo continuo de comandos 2103 recibe comandos desde la memoria y envía los comandos a la canalización 3D 2012 y/o a la canalización de medios 2016. Las órdenes son directivas extraídas de una memoria intermedia en anillo, que almacena órdenes para la canalización de 3D 2012 y la canalización de medios 2016. En un ejemplo, la memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de comandos en lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización de 3D 2012 pueden incluir también referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértices y geometría para la canalización de 3D 2012 y/o datos de imagen y objetos de memoria para la canalización de medios 2016. La canalización de 3D 2012 y la canalización de medios 2016 procesan los comandos y los datos realizando operaciones mediante la lógica dentro de las respectivas canalizaciones o despachando uno o más hilos de ejecución a una matriz de núcleo de gráficos 2114.

En diversos ejemplos, la canalización 3D 2012 puede ejecutar uno o más programas sombreadores, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculos u otros programas sombreadores, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleo de gráficos 2114. La matriz de núcleo de gráficos 2114 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución de múltiples fines (por ejemplo, las unidades de ejecución) dentro de la matriz de núcleo de gráficos 2114 incluye el soporte para diversos idiomas sombreadores de API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunos ejemplos, la matriz de núcleo de gráficos 2114 también incluye lógica de ejecución para realizar funciones de medios, tales como procesamiento de vídeo y/o de imagen. En un ejemplo, las unidades de ejecución incluyen adicionalmente lógica de fin general que es programable para realizar operaciones de cálculo de fin general paralelas, además de operaciones de procesamiento de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o junto con la lógica de propósito general dentro del núcleo o núcleos de procesador 1807 de la **Figura 18** o núcleo 1902A-1902N como en la **Figura 19.**

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleo de gráficos 2114 pueden emitir datos a memoria en una memoria intermedia de retorno unificada (URB) 2118. La URB 2118 puede almacenar datos para múltiples hilos. En algunos ejemplos, la URB 2118 puede usarse para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleo de gráficos 2114. En algunos ejemplos, la URB 2118 puede usarse adicionalmente para la sincronización entre hilos en la matriz de núcleo de gráficos y la lógica de función fija dentro de la lógica de función compartida 2120.

En algunos ejemplos, la matriz de núcleo de gráficos 2114 es escalable, de manera que la matriz incluye un número variable de núcleos de gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y el nivel de rendimiento del GPE 2110. En un ejemplo, los recursos de ejecución son dinámicamente escalables, de manera que pueden activarse o desactivarse los recursos de ejecución según sean necesarios.

La matriz de núcleo de gráficos 2114 se acopla con la lógica de función compartida 2120 que incluye múltiples recursos que se comparten entre los núcleos de gráficos en la matriz de núcleo de gráficos. Las funciones compartidas dentro de la lógica de función compartida 2120 son unidades de lógica de hardware que proporcionan funcionalidad complementaria especializada a la matriz de núcleo de gráficos 2114. En diversos ejemplos, lógica de función compartida 2120 incluye, pero sin limitación, el muestreador 2121, el cálculo matemático 2122 y la lógica de comunicación inter-hilo (ITC) 2123. Adicionalmente, algunos ejemplos implementan una o más caché o cachés 2125 dentro de la lógica de función compartida 2120. Se implementa una función compartida donde la demanda para una función especializada dada es insuficiente para la inclusión dentro de la matriz de núcleo de gráficos 2114. En su lugar, se implementa una única instanciación de esa función especializada como una entidad autónoma en la lógica de función compartida 2120 y se comparte entre los recursos de ejecución dentro de la matriz de núcleo de gráficos 2114. El conjunto preciso de funciones que se comparten entre la matriz de núcleo de gráficos 2114 y están incluidas dentro de la matriz de núcleo de gráficos 2114 varía entre ejemplos.

La **Figura 22** es un diagrama de bloques de otra realización de un procesador de gráficos 2200. Elementos de la **Figura 22** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura del presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte del presente documento, pero no se limitan a ello.

En algunos ejemplos, el procesador de gráficos 2200 incluye una interconexión de anillo 2202, un extremo frontal de canalización 2204, un motor de medios 2237 y núcleos de gráficos 2280A-2280N. En algunos ejemplos, la interconexión en anillo 2202 acopla el procesador de gráficos a otras unidades de procesamiento, que incluyen otros

procesadores de gráficos o uno o más núcleos de procesadores de propósito general. En algunos ejemplos, el procesador de gráficos es uno de muchos procesadores integrados dentro de un sistema de procesamiento de múltiples núcleos.

En algunos ejemplos, el procesador de gráficos 2200 recibe lotes de comandos a través de la interconexión en anillo 2202. Los comandos de entrada se interpretan por un emisor por flujo continuo de comandos 2203 en el extremo frontal de canalización 2204. En algunos ejemplos, el procesador de gráficos 2200 incluye una lógica de ejecución escalable para realizar procesamiento de geometría 3D y procesamiento de medios a través del núcleo o núcleos de gráficos 2280A-2280N. Para comandos de procesamiento de geometría 3D, el emisor por flujo continuo de comandos 2203 suministra comandos a la canalización de geometría 2236. Para al menos algunos comandos de procesamiento de medios, el emisor por flujo continuo de comandos 2203 suministra los comandos a un extremo frontal de vídeo 2234, que se acopla con un motor de medios 2237. En algunos ejemplos, el motor de medios 2237 incluye un motor de calidad de vídeo (VQE) 2230 para posprocesamiento de vídeo e imágenes y un motor de codificación/decodificación de múltiples formatos (MFX) 2233 para proporcionar codificación y decodificación de datos de medios acelerados por hardware. En algunos ejemplos, la canalización de geometría 2236 y el motor de medios 2237 generan, cada uno, hilos de ejecución para los recursos de ejecución de hilos proporcionados por al menos un núcleo de gráficos 2280A.

En algunos ejemplos, el procesador de gráficos 2200 incluye recursos de ejecución de hilos ajustables a escala que cuentan con los núcleos modulares 2280A-2280N (denominados, en ocasiones, cortes de núcleo), teniendo cada uno múltiples subnúcleos 2250A-2250N, 2260A-2260N (denominados, en ocasiones, subcortes de núcleo). En algunos ejemplos, el procesador de gráficos 2200 puede tener cualquier número de núcleos de gráficos de 2280A a 2280N. En algunos ejemplos, el procesador de gráficos 2200 incluye un núcleo de gráficos 2280A que tiene al menos un primer subnúcleo 2250A y un segundo subnúcleo de núcleo 2260A. En otros ejemplos, el procesador gráfico es un procesador de baja potencia con un único subnúcleo (por ejemplo, 2250A). En algunos ejemplos, el procesador de gráficos 2200 incluye múltiples núcleos de gráficos 2280A-2280N, incluyendo cada uno un conjunto de primeros subnúcleos 2250A-2250N y un conjunto de segundos subnúcleos 2260A-2260N. Cada subnúcleo en el conjunto de primeros subnúcleos 2250A-2250N incluye al menos un primer conjunto de unidades de ejecución 2252A-2252N y muestreadores de medios/texturas 2254A-2254N. Cada subnúcleo en el conjunto de segundos subnúcleos 2260A-2260N incluye al menos un segundo conjunto de unidades de ejecución 2262A-2262N y muestreadores 2264A-2264N. En algunos ejemplos, cada subnúcleo 2250A-2250N, 2260A-2260N comparte un conjunto de recursos compartidos 2270A-2270N. En algunos ejemplos, los recursos compartidos incluyen memoria de caché compartida y lógica de operación de píxel. Pueden incluirse también otros recursos compartidos en los diversos ejemplos del procesador de gráficos.

Lógica de ejecución

La **Figura 23** ilustra lógica de ejecución de hilos 2300 que incluye una matriz de elementos de procesamiento empleados en algunos ejemplos de un GPE. Elementos de la **Figura 23** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura del presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte del presente documento, pero no se limitan a ello.

En algunos ejemplos, la lógica de ejecución de hilos 2300 incluye un sombreador de píxeles 2302, un despachador de hilos 2304, una caché de instrucciones 2306, una matriz de unidades de ejecución escalable que incluye una pluralidad de unidades de ejecución 2308A-2308N, un muestreador 2310, una caché de datos 2312 y un puerto de datos 2314. En un ejemplo, los componentes incluidos están interconectados mediante una estructura de interconexión que se enlaza con cada uno de los componentes. En algunos ejemplos, la lógica de ejecución de hilos 2300 incluye una o más conexiones a memoria, tal como la memoria de sistema o memoria caché, a través de una o más de la caché de instrucciones 2306, el puerto de datos 2314, el muestreador 2310 y la matriz de unidades de ejecución 2308A-2308N. En algunos ejemplos, cada unidad de ejecución (por ejemplo, 2308A) es un procesador de vector individual que puede ejecutar múltiples hilos simultáneos y procesar múltiples elementos de datos en paralelo para cada hilo. En algunos ejemplos, la matriz de unidades de ejecución 2308A-2308N incluye cualquier número de unidades de ejecución individuales.

En algunos ejemplos, la matriz de unidades de ejecución 2308A-2308N se usa principalmente para ejecutar programas de "sombreador". En algunos ejemplos, las unidades de ejecución en la matriz 2308A-2308N ejecutan un conjunto de instrucciones que incluye el soporte nativo para muchas instrucciones de sombreador de gráficos 3D convencional, de manera que se ejecutan los programas de sombreador de las bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) con una traducción mínima. Las unidades de ejecución soportan procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y procesamiento de propósito general (por ejemplo, sombreadores de cálculo y de medios).

Cada unidad de ejecución en la matriz de unidades de ejecución 2308A-2308N opera en matrices de elementos de datos. El número de elementos de datos es el "tamaño de ejecución", o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso de elemento de datos, el enmascaramiento y el control de flujo dentro de las instrucciones. El número de canales puede ser independiente del número de Unidades

Aritmético-Lógicas (ALU) o Unidades de Coma Flotante (FPU) físicas para un procesador de gráficos particular. En algunos ejemplos, las unidades de ejecución 2308A-2308N soportan tipos de datos de números enteros y de coma flotante.

El conjunto de instrucciones de la unidad de ejecución incluye instrucciones de datos múltiples de instrucción única (SIMD) o instrucciones de hilos múltiples de instrucción única (SIMT). Los diversos elementos de datos se pueden almacenar como un tipo de datos empaquetado en un registro y la unidad de ejecución procesará los diversos elementos basándose en el tamaño de datos de los elementos. Por ejemplo, cuando se opera en un vector de 256 bits de ancho, los 256 bits del vector se almacenan en un registro y la unidad de ejecución opera en el vector como cuatro elementos de datos empaquetados de 64 bits separados (elementos de datos de tamaño de palabra cuádruple (QW)), ocho elementos de datos empaquetados de 32 bits separados (elementos de datos de tamaño de palabra doble (DW)), dieciséis elementos de datos empaquetados de 16 bits separados (elementos de datos de tamaño de palabra (W)), o treinta y dos elementos de datos de 8 bits separados (elementos de datos de tamaño de byte (B)). Sin embargo, son posibles diferentes anchuras de vector y tamaños de registro.

Una o más memorias caché de instrucciones internas (por ejemplo, 2306) se incluyen en la lógica de ejecución de hilos 2300 para almacenar en caché instrucciones de hilos para las unidades de ejecución. En algunos ejemplos, una o más cachés de datos (por ejemplo, 2312) están incluidas en datos de hilo de caché durante la ejecución de hilo. En algunos ejemplos, se incluye un muestreador 2310 para proporcionar un muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunos ejemplos, el muestreador 2310 incluye funcionalidad de textura especializada o muestreo de medios para procesar los datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las canalizaciones de gráficos y de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilos 2300 por medio de una lógica de generación y de despacho de hilos. En algunos ejemplos, la lógica de ejecución de hilos 2300 incluye un despachador de hilos local 2304 que arbitra las solicitudes de inicio de hilos de las canalizaciones de gráficos y medios y genera instancias a los hilos solicitados en una o más unidades de ejecución 2308A-2308N. Por ejemplo, la canalización de geometría (por ejemplo, 2236 de la **Figura 22**) despacha hilos de procesamiento de vértices, teselación o procesamiento de geometría a la lógica de ejecución de hilos 2300 (**Figura 23**). En algunos ejemplos, el despachador de hilos 2304 puede procesar también hilos en tiempo de ejecución que generan solicitudes desde los programas sombreadores de ejecución.

Una vez que un grupo de objetos geométricos ha sido procesado y rasterizado en datos de píxeles, se invoca el sombreador de píxeles 2302 para calcular además la información de salida y hacer que los resultados se escriban en las superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de estarcido, etc.). En algunos ejemplos, el sombreador de píxeles 2302 calcula los valores de los diversos atributos de vértice que han de interpolarse a través del objeto rasterizado. En algunos ejemplos, el sombreador de píxeles 2302 a continuación ejecuta un programa de sombreador de píxeles suministrado por la interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreador de píxeles, el sombreador de píxeles 2302 despacha hilos a una unidad de ejecución (por ejemplo, 2308A) mediante el despachador de hilos 2304. En algunos ejemplos, el sombreador de píxeles 2302 usa la lógica de muestreo de textura en el muestreador 2310 para acceder a datos de textura en mapas de textura almacenados en memoria. Las operaciones aritméticas en los datos de textura y los datos de geometría de entrada calculan datos de color de píxel para cada fragmento geométrico, o descartan uno o más píxeles del procesamiento posterior.

En algunos ejemplos, el puerto de datos 2314 proporciona un mecanismo de acceso de memoria para que la lógica de ejecución de hilos 2300 emita datos procesados a la memoria para su procesamiento en una canalización de salida de procesador de gráficos. En algunos ejemplos, el puerto de datos 2314 incluye o se acopla a una o más memorias caché (por ejemplo, la caché de datos 2312) para almacenar en caché datos para un acceso de memoria por medio del puerto de datos.

La **Figura 24** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos 2400. En uno o más ejemplos, las unidades de ejecución de procesador de gráficos soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los recuadros con línea continua ilustran los componentes que se incluyen en general en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que únicamente están incluidos en un subconjunto de las instrucciones. En algunos ejemplos, el formato de instrucción 2400 descrito e ilustrado son macro-instrucciones, en el sentido de que las mismas son instrucciones suministradas a la unidad de ejecución, en contraposición a micro-operaciones resultantes de la decodificación de instrucciones una vez que se ha procesado la instrucción.

En algunos ejemplos, las unidades de ejecución de procesador de gráficos soportan de manera nativa instrucciones en un formato de instrucción de 128 bits 2410. Un formato de instrucción compacta de 64 bits 2430 está disponible para algunas instrucciones basándose en la instrucción, las opciones de instrucción y el número de operandos seleccionados. El formato de instrucción de 128 bits nativo 2410 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de instrucción de 64 bits 2430. Las instrucciones nativas disponibles en el formato de instrucción de 64 bits 2430 varían según ejemplo. En

algunos ejemplos, la instrucción está compactada en parte usando un conjunto de valores de índice en un campo de índice 2413. El hardware de la unidad de ejecución hace referencia a un conjunto de tablas de compactación basándose en los valores de índice y usa las salidas de tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 2410.

Para cada formato, la operación de código de instrucción 2412 define la operación que ha de realizar la unidad de ejecución. Las unidades de ejecución ejecutan cada instrucción en paralelo a través de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de adición, la unidad de ejecución realiza una operación de adición simultánea en cada canal de color que representa un elemento de textura o elemento de imagen. Por defecto, la unidad de ejecución realiza cada instrucción a través de todos los canales de datos de los operandos. En algunos ejemplos, el campo de control de instrucciones 2414 permite el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicación) y el orden de los canales de datos (por ejemplo, mezcla). Para las instrucciones de 128 bits 2410, un campo de tamaño de ejecución 2416 limita el número de canales de datos que se ejecutarán en paralelo. En algunos ejemplos, el campo de tamaño de ejecución 2416 no está disponible para su uso en el formato de instrucción compacto de 64 bits 2430.

Algunas instrucciones de la unidad de ejecución tienen hasta tres operandos que incluyen dos operandos de origen, src0 2420, src1 2422, y un destino 2418. En algunos ejemplos, las unidades de ejecución soportan instrucciones de destino dual, donde está implicado uno de los destinos. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 2424), donde la operación de código de instrucción 2412 determina el número de operandos de origen. El último operando fuente de una instrucción puede ser un valor inmediato (por ejemplo, precodificado) pasado con la instrucción.

En algunos ejemplos, el formato de instrucción de 128 bits 2410 incluye una información de modo de acceso/dirección 2426 que especifica, por ejemplo, si se usa el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se usa el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos se proporciona directamente por los bits en la instrucción 2410.

En algunos ejemplos, el formato de instrucción de 128 bits 2410 incluye un campo de modo de acceso/dirección 2426, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En un ejemplo, el modo de acceso para definir una alineación de acceso de datos para la instrucción. Algunos ejemplos soportan modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, donde la alineación de bytes del modo de acceso determina la alineación de acceso de los operandos de instrucción. Por ejemplo, cuando está en un primer modo, la instrucción 2410 puede usar un direccionamiento alineado en bytes para operandos de origen y destino y, cuando está en un segundo modo, la instrucción 2410 puede usar direccionamiento alineado de 16 bytes para todos los operandos de origen y destino.

En un ejemplo, la porción de modo de dirección del campo de modo de acceso/dirección 2426 determina si la instrucción ha de usar el direccionamiento directo o indirecto. Cuando se usa el modo de direccionamiento de registro directo, los bits en la instrucción 2410 proporcionan directamente la dirección de registro de uno o más operandos. Cuando se usa un modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos puede calcularse basándose en un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

En algunos ejemplos, las instrucciones se agrupan basándose en los campos de bits del código de operación 2412 para simplificar la decodificación de código de operación 2440. Para un código de operación de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de código de operación. El agrupamiento del código de operación preciso mostrado es simplemente un ejemplo. En algunos ejemplos, un grupo de código de operación de movimiento y de lógica 2442 incluye instrucciones de movimiento y de lógica de datos (por ejemplo, mover (mov), comparar (cmp)). En algunos ejemplos, el grupo de movimiento y lógica 2442 comparte los cinco bits más significativos (MSB), donde las instrucciones mover (mov) están en forma de 0000xxxxb y las instrucciones de lógica están en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 2444 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones de miscelánea 2446 incluye una mezcla de instrucciones, que incluyen instrucciones de sincronización (por ejemplo, espera, envío) en forma de 001 1xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo matemático paralelo 2448 incluye instrucciones aritméticas a nivel de componente (por ejemplo, añadir, multiplicar (mult)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculo matemático paralelo 2448 realiza las operaciones aritméticas en paralelo a través de canales de datos. El grupo de cálculo matemático vectorial 2450 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cálculo matemático vectorial realiza aritmética tal como cálculos de producto escalar con operandos de vectores.

Canalización de gráficos

La **Figura 25** es un diagrama de bloques de otra realización de un procesador de gráficos 2500. Elementos de la **Figura 25** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura del presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte del presente documento, pero no se limitan a ello.

En algunos ejemplos, el procesador de gráficos 2500 incluye una canalización de gráficos 2520, una canalización de medios 2530, un motor de visualización 2540, una lógica de ejecución de hilos 2550 y una canalización de salida de representación 2570. En algunos ejemplos, el procesador de gráficos 2500 es un procesador de gráficos dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos se controla por las escrituras de registro en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 2500 mediante una interconexión en anillo 2502. En algunos ejemplos, la interconexión de anillo 2502 acopla el procesador de gráficos 2500 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de fin general. Los comandos desde la interconexión de anillo 2502 se interpretan por un transmisor de envío por flujo continuo de comandos 2503, que suministra instrucciones a componentes individuales de la canalización de gráficos 2520 o la canalización de medios 2530.

En algunos ejemplos, el emisor de envío por flujo continuo 2503 dirige la operación de un extractor de vértices 2505 que lee los datos de vértices de memoria y ejecuta comandos de procesamiento de vértices proporcionados por el emisor de envío por flujo continuo 2503. En algunos ejemplos, el extractor de vértices 2505 proporciona datos de vértices a un sombreador de vértices 2507, que realiza operaciones de transformación espacial de coordenadas y de iluminación en cada vértice. En algunos ejemplos, el extractor de vértices 2505 y el sombreador de vértices 2507 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a unidades de ejecución 2552A, 2552B mediante un despachador de hilos 2531.

En algunos ejemplos, las unidades de ejecución 2552A, 2552B son una matriz de procesadores vectoriales que tienen un conjunto de instrucciones para realizar operaciones de gráficos y de medios. En algunos ejemplos, las unidades de ejecución 2552A, 2552B tienen una caché L1 adjunta 2551 que es específica para cada matriz o está compartida entre las matrices. La caché puede configurarse como una caché de datos, una caché de instrucciones o una única caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.

En algunos ejemplos, la canalización de gráficos 2520 incluye componentes de teselación para realizar teselación acelerada por hardware de objetos 3D. En algunos ejemplos, un sombreador de casco programable 2511 configura las operaciones de teselación. Un sombreador de domino programable 2517 proporciona una evaluación de extremo trasero de la salida de teselación. Un teselador 2513 opera en la dirección del sombreador de casco 2511 y contiene lógica de propósito especial para generar un conjunto de objetos geométricos detallados basándose en un modelo geométrico aproximado que se proporciona como entrada a la canalización de gráficos 2520. En algunos ejemplos, si no se usa la teselación, pueden omitirse los componentes de teselación 2511, 2513, 2517.

En algunos ejemplos, pueden procesarse objetos geométricos completos por un sombreador de geometría 2519 mediante uno o más hilos despachados a unidades de ejecución 2552A, 2552B, o pueden continuar directamente al recortador 2529. En algunos ejemplos, el sombreador de geometría opera en objetos geométricos enteros, en lugar de en vértices o parches de vértices como en etapas anteriores de la canalización de gráficos. Si la teselación está deshabilitada, el sombreador de geometría 2519 recibe una entrada desde el sombreador de vértices 2507. En algunos ejemplos, el sombreador de geometría 2519 es programable mediante un programa de sombreado de geometría para realizar una teselación de geometría si las unidades de teselación están deshabilitadas.

Antes de la rasterización, un recortador 2529 procesa datos de vértices. El recortador 2529 puede ser un recortador de función fija o un recortador programable que tiene funciones de recortador y de sombreador de geometría. En algunos ejemplos, un componente de prueba de rasterizador y profundidad 2573 en la canalización de salida del representador 2570 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxeles. En algunos ejemplos, la lógica de sombreador de píxeles está incluida en la lógica de ejecución de hilos 2550. En algunos ejemplos, una aplicación puede omitir la rasterización y acceder a datos de vértices no rasterizados a través de una unidad de salida de flujo 2523.

El procesador de gráficos 2500 tiene un bus de interconexión, una estructura de interconexión o algún otro mecanismo de interconexión que permite el paso de datos y de mensajes entre los componentes principales del procesador. En algunos ejemplos, las unidades de ejecución 2552A, 2552B y la caché o cachés asociadas 2551, el muestreador de textura y de medios 2554 y la caché de textura/muestreador 2558 se interconectan mediante un puerto de datos 2556 para realizar el acceso a memoria y comunicarse con los componentes de canalización de salida de representación del procesador. En algunos ejemplos, cada uno del muestreador 2554, cachés 2551, 2558 y las unidades de ejecución 2552A, 2552B tienen rutas de acceso a memoria separadas.

En algunos ejemplos, la canalización de salida del representador 2570 contiene un componente de prueba de rasterizador y profundidad 2573 que convierte objetos basados en vértices en una representación basada en píxeles asociada. En algunos ejemplos, la canalización de salida de representador 2570 incluye una unidad generadora de ventanas/enmascaradora para realizar una rasterización de líneas y de triángulos de función fija. Una caché de representación 2578 y una caché de profundidad 2579 asociadas también están disponibles en algunos ejemplos. Un componente de operaciones de píxel 2577 realiza operaciones basadas en píxeles sobre los datos, aunque, en algunas instancias, las operaciones de píxel asociadas con operaciones 2D (por ejemplo, transferencias de imagen

de bloque de bits con mezcla) son realizadas por el motor 2D 2541, o son sustituidas en el momento de la visualización por el controlador de visualización 2543 usando planos de visualización de superposición. En algunos ejemplos, está disponible una caché L3 compartida 2575 para todos los componentes de gráficos, permitiendo compartir datos sin el uso de memoria de sistema principal.

En algunos ejemplos, la canalización de medios de procesador de gráficos 2530 incluye un motor de medios 2537 y un extremo frontal de vídeo 2534. En algunos ejemplos, el extremo frontal de vídeo 2534 recibe comandos de canalización desde el emisor por flujo continuo de comandos 2503. En algunos ejemplos, la canalización de medios 2530 incluye un emisor por flujo continuo de comandos separado. En algunos ejemplos, el extremo frontal de vídeo 2534 procesa comandos de medios antes de enviar el comando al motor de medios 2537. En algunos ejemplos, el motor de medios 2537 incluye funcionalidad de generación de hilos para generar hilos para su despacho a la lógica de ejecución de hilos 2550 a través del distribuidor de hilos 2531.

En algunos ejemplos, el procesador de gráficos 2500 incluye un motor de visualización 2540. En algunos ejemplos, el motor de visualización 2540 es externo al procesador 2500 y se acopla con el procesador de gráficos mediante el anillo de interconexión 2502, o algún otro bus o estructura de interconexión. En algunos ejemplos, el motor de visualización 2540 incluye un motor 2D 2541 y un controlador de visualización 2543. En algunos ejemplos, el motor de visualización 2540 contiene una lógica de propósito especial que puede operar independientemente de la canalización 3D. En algunos ejemplos, el controlador de visualización 2543 se acopla con un dispositivo de visualización (no mostrado), que puede ser un dispositivo de visualización integrado en sistema, como en un ordenador portátil, o un dispositivo de visualización externo adjunto mediante un conector de dispositivo de visualización.

En algunos ejemplos, la canalización de gráficos 2520 y la canalización de medios 2530 se pueden configurar para realizar operaciones basándose en múltiples interfaces de programación de gráficos y de medios y no son específicas de ninguna interfaz de programación de aplicaciones (API). En algunos ejemplos, el software de controlador para el procesador de gráficos traduce llamadas de API que son específicas de una biblioteca de medios o de gráficos particular a comandos que pueden ser procesados por el procesador de gráficos. En algunos ejemplos, se proporciona soporte para la biblioteca Open Graphics (OpenGL) y Open Computing Language (OpenCL) de Khronos Group, la biblioteca Direct3D de Microsoft Corporation, o se puede proporcionar soporte tanto para OpenGL como para D3D. Puede proporcionarse también soporte para la Biblioteca de Visión Informática de Código Abierto (OpenCV). También se soportaría una API futura con una canalización 3D compatible si pudiera hacerse un mapeo de la canalización de la API futura a la canalización del procesador de gráficos.

Programación de canalización de gráficos

La **Figura 26A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos 2600. La **Figura 26B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos 2610. Los recuadros con líneas continuas en la **Figura 26A** ilustran los componentes que generalmente se incluyen en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de los comandos de gráficos. El formato de comando de procesador de gráficos 2600 ilustrativo de la **Figura 26A** incluye campos de datos para identificar un cliente objetivo 2602 del comando, un código de operación de comando (código de operación) 2604 y los datos pertinentes 2606 para el comando. También se incluye un subcódigo de operación 2605 y un tamaño de comando 2608 en algunos comandos.

En algunos ejemplos, el cliente 2602 especifica la unidad de cliente del dispositivo de gráficos que procesa los datos de comando. En algunos ejemplos, un analizador de comandos de procesador de gráficos examina el campo de cliente de cada comando para acondicionar el procesamiento adicional del comando y enrutar los datos de comando a la unidad cliente apropiada. En algunos ejemplos, las unidades cliente de procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad del representador, una unidad 2D, una unidad 3D y una unidad de medios. Cada unidad de cliente tiene una canalización de procesamiento correspondiente que procesa los comandos. Una vez que se recibe el comando por la unidad de cliente, la unidad de cliente lee el código de operación 2604 y, si está presente, el subcódigo de operación 2605 para determinar la operación a realizar. La unidad cliente realiza el comando usando información en el campo de datos 2606. Para algunos comandos, se espera un tamaño de comando explícito 2608 para especificar el tamaño del comando. En algunos ejemplos, el analizador de comandos determina automáticamente el tamaño de al menos alguno de los comandos basándose en el código de operación del comando. En algunos ejemplos, se alinean los comandos mediante múltiplos de una palabra doble.

El diagrama de flujo de la **Figura 26B** muestra una secuencia de comandos de procesador de gráficos 2610 ilustrativa. En algunos ejemplos, el software o firmware de un sistema de procesamiento de datos que presenta una realización de un procesador de gráficos que usa una versión de la secuencia de comandos mostrada para establecer, ejecutar y terminar un conjunto de operaciones de gráficos. Se muestra una secuencia de comandos de muestra y se describe únicamente con propósitos ilustrativos, ya que los ejemplos no están limitados a estos comandos específicos o para esta secuencia de comandos. Además, pueden emitirse los comandos como un lote de comandos en una secuencia de comandos, de manera que el procesador de gráficos procesará la secuencia de comandos en concurrencia al menos parcialmente.

En algunos ejemplos, la secuencia de comandos del procesador de gráficos 2610 puede comenzar con un comando de vaciado de canalización 2612 para hacer que alguna canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunos ejemplos, la canalización 3D 2622 y la canalización de medios 2624 no operan concurrentemente. Se realiza la descarga de la canalización para hacer que la canalización de gráficos activa complete cualquier comando pendiente. En respuesta a un vaciado de canalización, el analizador de comandos del procesador gráfico pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. Opcionalmente, cualquier dato en la caché de representación que esté marcado como 'sucio' puede vaciarse a memoria. En algunos ejemplos, puede usarse el comando de vaciado de canalización 2612 para la sincronización de canalización o antes de colocar el procesador de gráficos en un estado de baja potencia.

En algunos ejemplos, se usa un comando de selección de canalización 2613 cuando una secuencia de comandos requiere que el procesador de gráficos conmute explícitamente entre canalizaciones. En algunos ejemplos, se requiere únicamente un comando de selección de canalización 2613 una vez dentro de un contexto de ejecución antes de emitir comandos de canalización a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunos ejemplos, se requiere un comando de vaciado de canalización 2612 inmediatamente antes de un conmutador de canalización mediante el comando de selección de canalización 2613.

En algunos ejemplos, un comando de control de canalización 2614 configura una canalización de gráficos para la operación y se usa para programar la canalización 3D 2622 y la canalización de medios 2624. En algunos ejemplos, el comando de control de canalización 2614 configura el estado de canalización para la canalización activa. En un ejemplo, se usa el comando de control de canalización 2614 para sincronización de canalización y para limpiar datos de una o más memorias de caché dentro de la canalización activa antes de procesar un lote de comandos.

En algunos ejemplos, los comandos para el estado de memoria intermedia de retorno 2616 se usan para configurar un conjunto de memorias intermedias de retorno para que las respectivas canalizaciones escriban datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunos ejemplos, el procesador de gráficos también usa una o más memorias intermedias de retorno para almacenar datos de salida y realizar comunicación de hilos cruzada. En algunos ejemplos, configurar el estado de memoria intermedia de retorno 2616 incluye seleccionar el tamaño y número de memorias intermedias de retorno para su uso para un conjunto de operaciones de canalización.

Los comandos restantes en la secuencia de comandos difieren basándose en la canalización activa para las operaciones. Basándose en una determinación de la canalización 2620, la secuencia de comandos se adapta a la canalización 3D 2622 que comienza con el estado de canalización 3D 2630, o a la canalización de medios 2624 que comienza en el estado de canalización de medios 2640.

Los comandos para el estado de canalización 3D 2630 incluyen los comandos de ajuste de estado 3D para el estado de memoria intermedia de vértice, estado de elemento de vértice, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que han de configurarse antes de que se procesen los comandos de primitiva 3D. Los valores de estos comandos se determinan, al menos en parte, basándose en la API 3D particular en uso. En algunos ejemplos, los comandos de estado de canalización de 3D 2630 también pueden desactivar o desviar selectivamente ciertos elementos de canalización si esos elementos no se usarán.

En algunos ejemplos, se usa el comando de primitiva 3D 2632 para enviar primitivas 3D para su procesamiento por la canalización 3D. Los comandos y parámetros asociados que se pasan al procesador de gráficos mediante el comando de primitiva 3D 2632 se reenvían a la función de extracción de vértices en la canalización de gráficos. La función de extracción de vértices usa los datos de comando de primitiva 3D 2632 para generar estructuras de datos de vértices. Las estructuras de datos de vértices se almacenan en una o más memorias intermedias de retorno. En algunos ejemplos, se usa el comando de primitiva 3D 2632 para realizar operaciones de vértice en primitivas 3D mediante sombreadores de vértices. Para procesar sombreadores de vértice, la canalización 3D 2622 despacha hilos de ejecución de sombreadores a unidades de ejecución de procesador de gráficos.

En algunos ejemplos, la canalización 3D 2622 se activa mediante un comando o evento de ejecución 2634. En algunos ejemplos, una escritura de registro activa la ejecución de comando. En algunos ejemplos, se activa la ejecución mediante un comando 'ir' o 'disparar' en la secuencia de comandos. En un ejemplo se activa la ejecución de comando usando un comando de sincronización de canalización para vaciar la secuencia de comandos a través de la canalización de gráficos. La canalización 3D realizará un procesamiento de geometría para las primitivas 3D. Una vez que están completadas las operaciones, se rasterizan los objetos geométricos resultantes y los colores de motor de píxel y los píxeles resultantes. Pueden incluirse también comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo trasero de píxeles para estas operaciones.

En algunos ejemplos, la secuencia de comandos de procesador de gráficos 2610 sigue la ruta de canalización de medios 2624 cuando se realizan operaciones de medios. En general, el uso y manera específicos de la programación para la canalización de medios 2624 depende de las operaciones de medios o de cálculo que van a realizarse. Las

operaciones de decodificación de medios específicas pueden descargarse hacia la canalización de medios durante la decodificación de medios. En algunos ejemplos, puede desviarse también la canalización de medios y puede realizarse la decodificación de medios, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. En un ejemplo, la canalización de medios también incluye elementos para las operaciones de la unidad de procesador de gráficos de propósito general (GPGPU), donde se usa el procesador de gráficos para realizar operaciones vectoriales SIMD usando programas de sombreador computacionales que no están relacionados explícitamente con la representación de primitivas de gráficos.

En algunos ejemplos, la canalización de medios 2624 se configura de una manera similar a la de la canalización de 3D 2622. Un conjunto de comandos para configurar el estado de canalización de medios 2640 se despacha o coloca en una cola de comandos antes de los comandos de objeto de medios 2642. En algunos ejemplos, los comandos para el estado de canalización de medios 2640 incluyen datos para configurar los elementos de canalización de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación y codificación de vídeo dentro de la canalización de medios, tal como el formato de codificación o decodificación. En algunos ejemplos, los comandos para el estado de canalización de medios 2640 también soportan el uso de uno o más punteros a elementos de estado "indirectos" que contienen un lote de configuraciones de estado.

En algunos ejemplos, los comandos de objeto de medios 2642 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias que contienen datos de vídeo que van a procesarse. En algunos ejemplos, todos los estados de canalización de medios deben ser válidos antes de emitir un comando de objeto de medios 2642. Una vez que se ha configurado el estado de la canalización y los comandos de objeto de medios 2642 se han puesto en cola, se activa la canalización de medios 2624 por medio de un comando de ejecución 2644 o un evento de ejecución equivalente (por ejemplo, una escritura de registro). La salida desde la canalización de medios 2624 puede posprocesarse, a continuación, mediante operaciones proporcionadas por la canalización 3D 2622 o la canalización de medios 2624. En algunos ejemplos, las operaciones de GPGPU se configuran y ejecutan de una manera similar a las operaciones de medios.

Arquitectura de software de gráficos

La **Figura 27** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos 2700. En algunos ejemplos, la arquitectura de software incluye una aplicación de gráficos 3D 2710, un sistema operativo 2720 y al menos un procesador 2730. En algunos ejemplos, el procesador 2730 incluye un procesador de gráficos 2732 y uno o más núcleo o núcleos de procesador de propósito general 2734. Cada uno de la aplicación de gráficos 2710 y el sistema operativo 2720 se ejecutan en la memoria de sistema 2750 del sistema de procesamiento de datos.

En algunos ejemplos, la aplicación de gráficos 3D 2710 contiene uno o más programas sombreadores que incluyen instrucciones de sombreador 2712. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el Lenguaje de Sombreador de Alto Nivel (HLSL) o el Lenguaje de Sombreador OpenGL (GLSL). La aplicación también incluye las instrucciones ejecutables 2714 en un lenguaje máquina adecuado para su ejecución por el núcleo o núcleos de procesador de propósito general 2734. La aplicación también incluye los objetos de gráficos 2716 definidos por datos de vértices.

En algunos ejemplos, el sistema operativo 2720 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo similar a UNIX propietario o un sistema operativo similar a UNIX de código abierto que usa una variante del núcleo Linux. El sistema operativo 2720 puede soportar una API de gráficos 2722 tal como la API Direct3D o la API OpenGL. Cuando está en uso la API Direct3D, el sistema operativo 2720 usa un compilador de sombreador de extremo frontal 2724 para compilar cualquier instrucción de sombreador 2712 en HLSL en un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede realizar una compilación previa de sombreador. En algunos ejemplos, los sombreadores de alto nivel se compilan en sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 2710.

En algunos ejemplos, el controlador de gráficos de modo de usuario 2726 contiene un compilador de sombreador de extremo trasero 2727 para convertir las instrucciones de sombreador 2712 en una representación específica de hardware. Cuando la API OpenGL está en uso, las instrucciones de sombreador 2712 en el lenguaje de alto nivel GLSL se pasan al controlador de gráficos de modo de usuario 2726 para su compilación. En algunos ejemplos, el controlador de gráficos de modo de usuario 2726 usa las funciones de modo de núcleo de sistema operativo 2728 para comunicarse con un controlador de gráficos de modo de núcleo 2729. En algunos ejemplos, el controlador de gráficos en modo de núcleo 2729 se comunica con el procesador de gráficos 2732 para despachar comandos e instrucciones.

Implementaciones de núcleo de IP

Uno o más aspectos pueden implementarse mediante un código representativo almacenado en un medio legible por máquina que representa y/o define una lógica dentro de un circuito integrado tal como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan diversa lógica dentro del procesador. Cuando

se leen por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en el presente documento. Tales representaciones, conocidas como "núcleos de IP", son unidades reutilizables de lógica para un circuito integrado que pueden almacenarse en un medio legible por máquina tangible como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware se puede suministrar a diversos clientes o instalaciones de fabricación, que cargan el modelo de hardware en máquinas de fabricación que fabrican el circuito integrado. El circuito integrado se puede fabricar de tal manera que el circuito realice las operaciones descritas en asociación con cualquiera de los ejemplos descritos en el presente documento.

La **Figura 28** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 2800 que se puede usar para fabricar un circuito integrado para realizar operaciones. El sistema de desarrollo de núcleo de IP 2800 puede usarse para generar diseños reutilizables modulares que pueden incorporarse en un diseño más grande o usarse para construir un circuito integrado entero (por ejemplo, un circuito de SOC integrado). Una instalación de diseño 2830 puede generar una simulación de software 2810 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 2810 puede usare para diseñar, probar y verificar el comportamiento del núcleo de IP usando un modelo de simulación 2812. El modelo de simulación 2812 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Puede crearse o sintetizarse, a continuación, un diseño de nivel de transferencia de registro (RTL) 2815 a partir del modelo de simulación 2812. El diseño de RTL 2815 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, que incluyen la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño de RTL 2815, los diseños de nivel inferior en el nivel de lógica o en el nivel de transistores también pueden crearse, diseñarse o sintetizarse. Por tanto, los detalles particulares del diseño y simulación inicial pueden variar.

El diseño de RTL 2815, o un equivalente, puede sintetizarse además por la instalación de diseño para obtener un modelo de hardware 2820, que puede estar en un lenguaje de descripción de hardware (HDL) o alguna otra representación de datos de diseño físico. El HDL puede simularse o probarse además para verificar el diseño de núcleo de IP. El diseño de núcleo de IP puede almacenarse para su entrega a una instalación de fabricación de 3os 2865 usando memoria no volátil 2840 (por ejemplo, disco duro, memoria flash o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño del núcleo IP se puede transmitir (por ejemplo, a través de Internet) a través de una conexión alámbrica 2850 o una conexión inalámbrica 2860. La instalación de fabricación 2865 puede fabricar a continuación un circuito integrado que se basa, al menos en parte, en el diseño de núcleo de IP. El circuito integrado fabricado puede estar configurado para realizar operaciones de acuerdo con al menos un ejemplo descrito en el presente documento.

Circuito integrado de sistema en chip ilustrativo

Las **Figuras 29-31** ilustran circuitos integrados ilustrativos y procesadores de gráficos asociados que pueden fabricarse usando uno o más núcleos de IP. Además de lo que se ilustra, se pueden incluir otros circuitos y lógica, incluyendo procesadores/núcleos de gráficos adicionales, controladores de interfaz de periféricos o núcleos de procesador de propósito general.

La **Figura 29** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip 2900 ilustrativo que puede fabricarse usando uno o más núcleos de IP. El circuito integrado 2900 ilustrativo incluye uno o más procesador o procesadores de aplicación 2905 (por ejemplo, las CPU), al menos un procesador de gráficos 2910, y puede incluir adicionalmente un procesador de imágenes 2915 y/o un procesador de vídeo 2920, cualquiera de los que puede ser un núcleo de IP modular desde las mismas o múltiples diferentes instalaciones de diseño. El circuito integrado 2900 incluye una lógica de bus o de periféricos que incluye un controlador de USB 2925, un controlador de UART 2930, un controlador de SPI/SDIO 2935 y un controlador de I^2S/I^2C 2940. Adicionalmente, el circuito integrado puede incluir un dispositivo de visualización 2945 acoplado a uno o más de un controlador de interfaz multimedia de alta definición (HDMI) 2950 y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 2955. El almacenamiento puede proporcionarse por un subsistema de memoria flash 2960 que incluye memoria flash y un controlador de memoria flash. La interfaz de memoria puede proporcionarse mediante un controlador de memoria 2965 para acceso a dispositivos de memoria de SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad embebido 2970.

La **Figura 30** es un diagrama de bloques que ilustra un procesador de gráficos 3010 ilustrativo de un circuito integrado de sistema en chip que se puede fabricar usando uno o más núcleos IP. El procesador de gráficos 3010 puede ser una variante del procesador de gráficos 2910 de la **Figura 29.** El procesador de gráficos 3010 incluye un procesador de vértices 3005 y uno o más procesador o procesadores de fragmentos 3015A-3015N (por ejemplo, 3015A, 3015B, 3015C, 3015D a 3015N-1 y 3015N). El procesador de gráficos 3010 puede ejecutar diferentes programas sombreadores mediante lógica separada, de manera que el procesador de vértices 3005 está optimizado para ejecutar operaciones para programas de sombreador de vértices, mientras que el uno o más procesador o procesadores de fragmentos 3015A-3015N ejecutan operaciones de sombreado de fragmentos (por ejemplo, de píxeles) para programas sombreadores de fragmentos o de píxeles. El procesador de vértices 3005 realiza la etapa de procesamiento de vértices de la canalización de gráficos 3D y genera datos de primitivas y de vértices. El procesador o procesadores de fragmentos 3015A-3015N usan los datos de primitiva y de vértice generados por el procesador de vértices 3005 para producir una memoria intermedia de fotogramas que se visualiza en un dispositivo de visualización.

En una realización, el procesador o procesadores de fragmentos 3015A-3015N están optimizados para ejecutar programas sombreadores de fragmento según se proporciona en la API de OpenGL, que pueden usarse para realizar operaciones similares como un programa sombreador de píxeles como se proporciona en la API de Direct 3D.

El procesador de gráficos 3010 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 3020A-3020B, caché o cachés 3025A-3025B e interconexión o interconexiones de circuito 3030A-3030B. La una o más MMU 3020A-3020B proporcionan el mapeo de direcciones virtuales a físicas para el procesador de gráficos 3010, incluyendo para el procesador de vértices 3005 y/o el procesador o procesadores de fragmentos 3015A-3015N, que pueden hacer referencia a datos de vértices o imágenes/texturas almacenados en memoria, además de datos de vértices o imágenes/texturas almacenados en la una o más cachés 3025A-3025B. En un ejemplo, la una o más MMU 3020A-3020B pueden estar sincronizadas con otras MMU dentro del sistema, que incluyen una o más MMU asociadas con el uno o más procesadores de aplicación 2905, el procesador de imágenes 2915 y/o el procesador de vídeo 2920 de la **Figura 29,** de manera que cada procesador 2905-2920 puede participar en un sistema de memoria virtual compartida o unificada. La una o más interconexión o interconexiones de circuito 3030A-3030B posibilitan que el procesador de gráficos 3010 interconecte con otros núcleos de IP dentro del SoC, mediante un bus interno del SoC o mediante una conexión directa, de acuerdo con ejemplos.

La **Figura 31** es un diagrama de bloques que ilustra un procesador de gráficos 3110 ilustrativo adicional ilustrativo de un circuito integrado de sistema en chip que se puede fabricar usando uno o más núcleos IP, de acuerdo con una realización. El procesador de gráficos 3110 puede ser una variante del procesador de gráficos 2910 de la **Figura 29.** El procesador de gráficos 3110 incluye las una o más MMU 3020A-3020B, caché o cachés 3025A-3025B e interconexión o interconexiones de circuito 3030A-3030B del circuito integrado 3000 de la **Figura 30**.

El procesador de gráficos 3110 incluye uno o más núcleo o núcleos de sombreador 3115A-3115N (por ejemplo, 3115A, 3115B, 3115C, 3115D, 3115E, 3115F, a 3015N-1 y 3015N), que proporciona una arquitectura de núcleo de sombreado unificada en la que un solo núcleo o tipo o núcleo puede ejecutar todo tipo de código de sombreado programable, incluyendo el código de programa de sombreador para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. El número exacto de núcleos de sombreado presentes puede variar entre ejemplos e implementaciones. Además, el procesador de gráficos 3110 incluye un gestor de tareas inter-núcleo 3105, que actúa como un despachador de hilos para enviar hilos de ejecución a uno o más núcleo o núcleos de sombreado 3115A-3115N. El procesador de gráficos 3110 incluye adicionalmente una unidad de mosaico 3118 para acelerar las operaciones de mosaico para la representación basada en mosaicos, en la que las operaciones de representación para una escena se subdividen en el espacio de la imagen. La representación basada en mosaicos se puede usar para aprovechar la coherencia espacial local dentro de una escena o para optimizar el uso de cachés internas.

En la siguiente descripción y las reivindicaciones, puede usarse el término "acoplado" junto con sus derivadas. "Acoplado" se usa para indicar que dos o más elementos cooperan o interactúan entre sí, pero pueden tener o no componentes físicos o eléctricos intermedios entre ellos.

Como se usa en las reivindicaciones, a menos que se especifique lo contrario, el uso de los adjetivos ordinales "primero", "segundo", "tercero", etc., para describir un elemento común, simplemente indica que se hace referencia a diferentes instancias de elementos similares, y no pretenden implicar que los elementos así descritos deban estar en una secuencia determinada, ya sea temporal, espacial, en clasificación o de cualquier otra manera.

Los dibujos y la descripción anterior dan ejemplos de realizaciones. Los expertos en la materia apreciarán que uno o más de los elementos descritos pueden combinarse en un único elemento funcional. Como alternativa, ciertos elementos pueden dividirse en múltiples elementos funcionales. Se pueden añadir elementos de una realización a otra realización.

Son posibles numerosas variaciones, ya sea que se indiquen explícitamente en la memoria descriptiva o no, tal como diferencias en la estructura, dimensión y uso del material. El alcance de las realizaciones es tan amplio como se proporciona la siguiente reivindicación.

REIVINDICACIONES

1. Un método (900) a realizar para un procesador de gráficos (614) que tiene múltiples procesos de aplicación coexistentes para el empleo de interferencia de redes neuronales, que comprende:
definir un porcentaje para cada proceso de aplicación (901);
planificar, mediante una lógica de planificación (820) del procesador de gráficos (614), los procesos de aplicación de acuerdo con el porcentaje de hilos y recursos disponibles total;
ajustar dinámicamente el porcentaje para cada proceso de aplicación por un usuario, en donde el ajuste incluye actualizar, usando primitivas proporcionadas al usuario, el porcentaje para afinar la utilización del procesador de gráficos (614),
en donde las primitivas incluyen primitivas para seleccionar y escribir un porcentaje deseado y un porcentaje de límite inferior para un proceso de aplicación identificado por un controlador proporcional integral derivativo, PID (909) y primitivas para leer un porcentaje asignado de sistema actual, el porcentaje deseado por el usuario y el porcentaje de límite inferior.

2. Un sistema que comprende:
un procesador de gráficos (614) que tiene múltiples procesos de aplicación coexistentes para el empleo de interferencia de redes neuronales;
una lógica de planificación (820) para planificar los procesos de aplicación de acuerdo con el porcentaje de los hilos y recursos disponibles total;
en donde el sistema es para definir un porcentaje para cada proceso de aplicación (901) y para ajustar dinámicamente el porcentaje para cada proceso de aplicación por un usuario, en donde el ajuste incluye actualizar, usando primitivas proporcionadas al usuario, el porcentaje para afinar la utilización del procesador de gráficos (614),
en donde las primitivas incluyen primitivas para seleccionar y escribir un porcentaje deseado y un porcentaje de límite inferior para un proceso de aplicación identificado por un controlador proporcional integral derivativo, PID (909) del sistema y primitivas para leer un porcentaje asignado de sistema actual, el porcentaje deseado por el usuario y el porcentaje de límite inferior.

3. Uno o más medios de almacenamiento legibles por máquina que tienen almacenadas en los mismos instrucciones de programa informático ejecutables que, cuando son ejecutadas por una o más máquinas, hacen que la una o más máquinas realicen el método de la reivindicación 1.

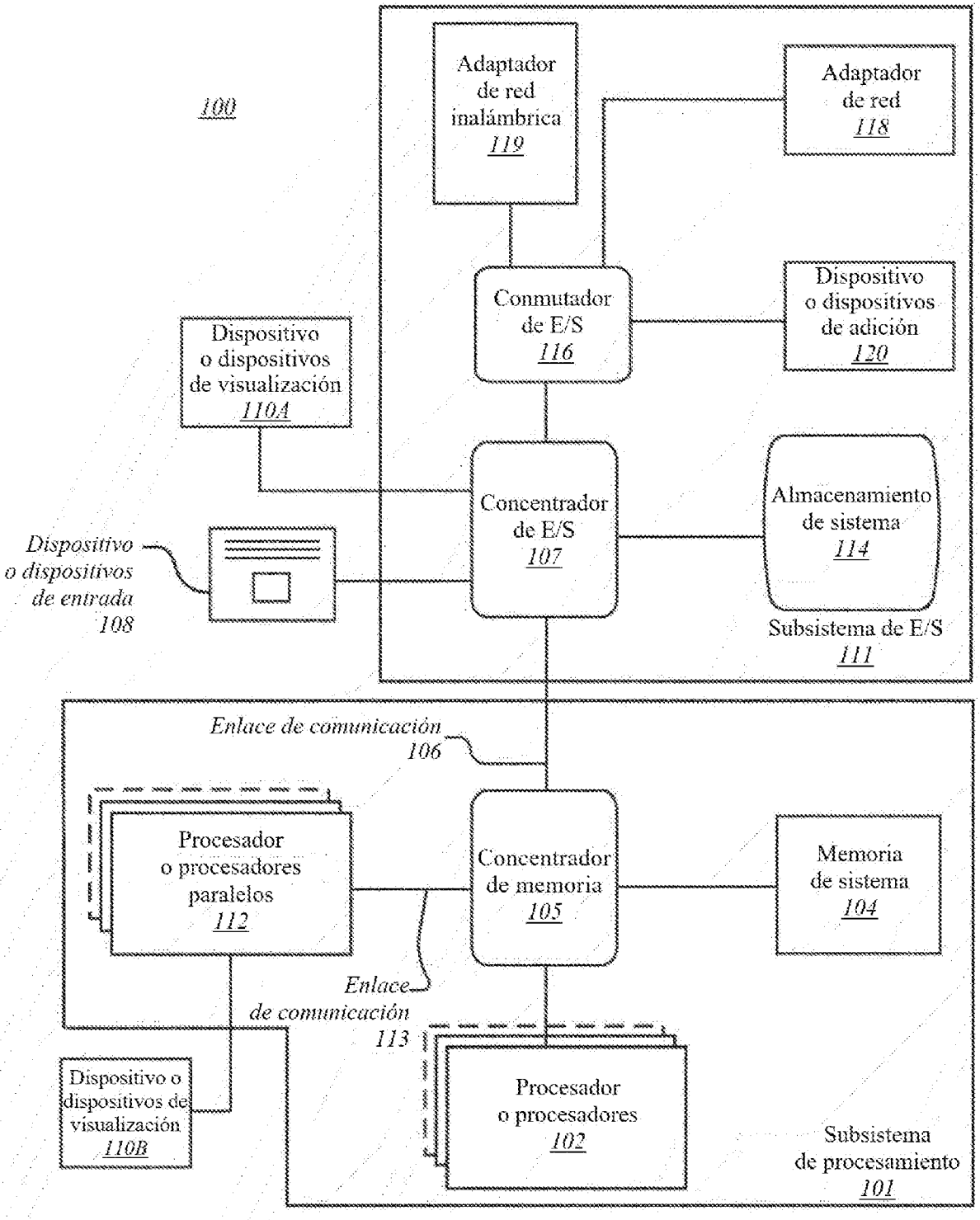

100
Adaptador de red inalámbrica 119
Adaptador de red 118
Conmutador de E/S 116
Dispositivo o dispositivos de adición 120
Dispositivo o dispositivos de visualización 110A
Concentrador de E/S 107
Almacenamiento de sistema 114
Dispositivo o dispositivos de entrada 108
Subsistema de E/S 111
Enlace de comunicación 106
Procesador o procesadores paralelos 112
Concentrador de memoria 105
Memoria de sistema 104
Enlace de comunicación 113
Dispositivo o dispositivos de visualización 110B
Procesador o procesadores 102
Subsistema de procesamiento 101

FIG. 1

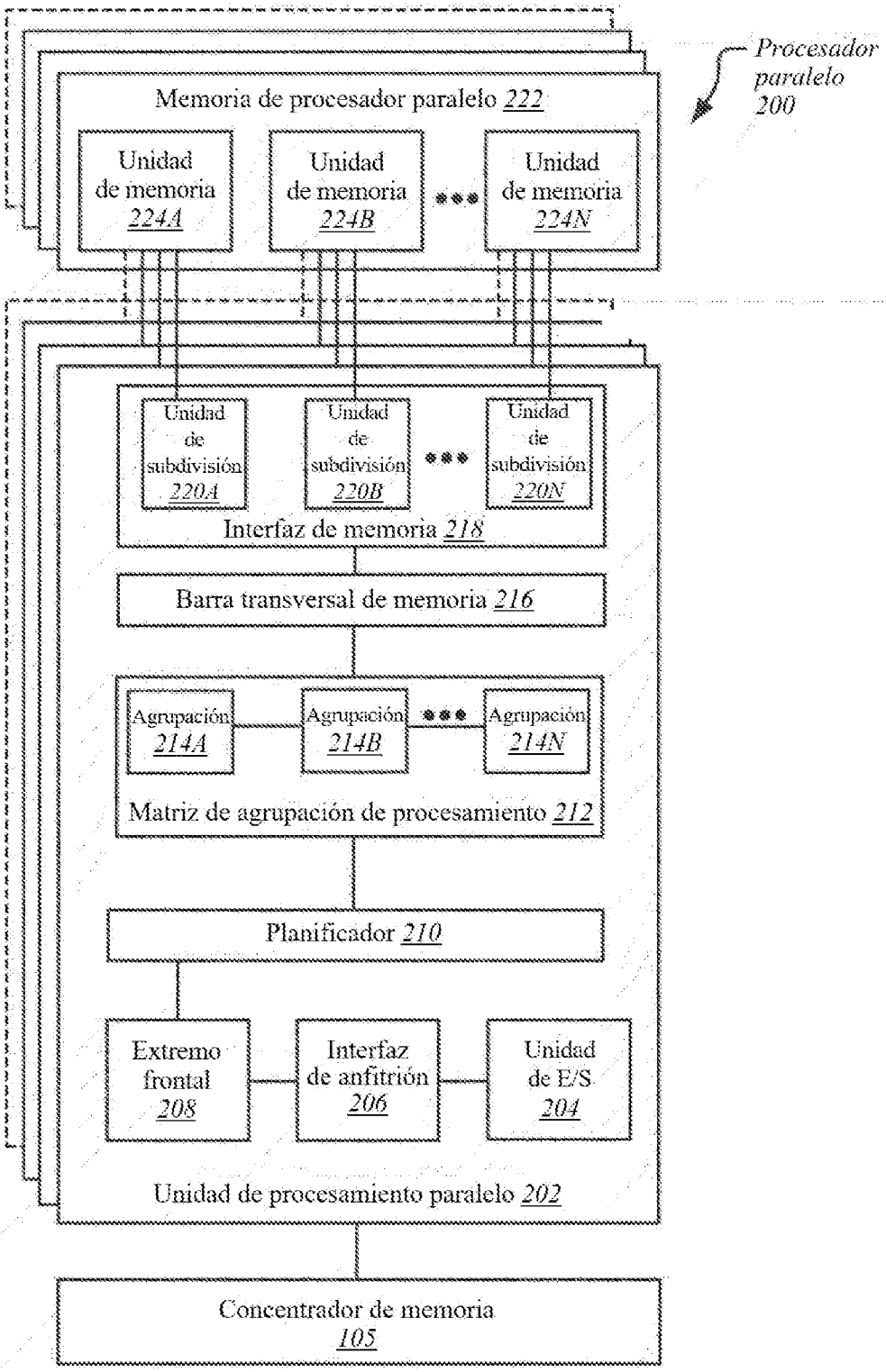

Procesador paralelo 200
Memoria de procesador paralelo 222
Unidad de memoria 224A
Unidad de memoria 224B
Unidad de memoria 224N
Unidad de subdivisión 220A
Unidad de subdivisión 220B
Unidad de subdivisión 220N
Interfaz de memoria 218
Barra transversal de memoria 216
Agrupación 214A
Agrupación 214B
Agrupación 214N
Matriz de agrupación de procesamiento 212
Planificador 210
Extremo frontal 208
Interfaz de anfitrión 206
Unidad de E/S 204
Unidad de procesamiento paralelo 202
Concentrador de memoria 105

FIG. 2A

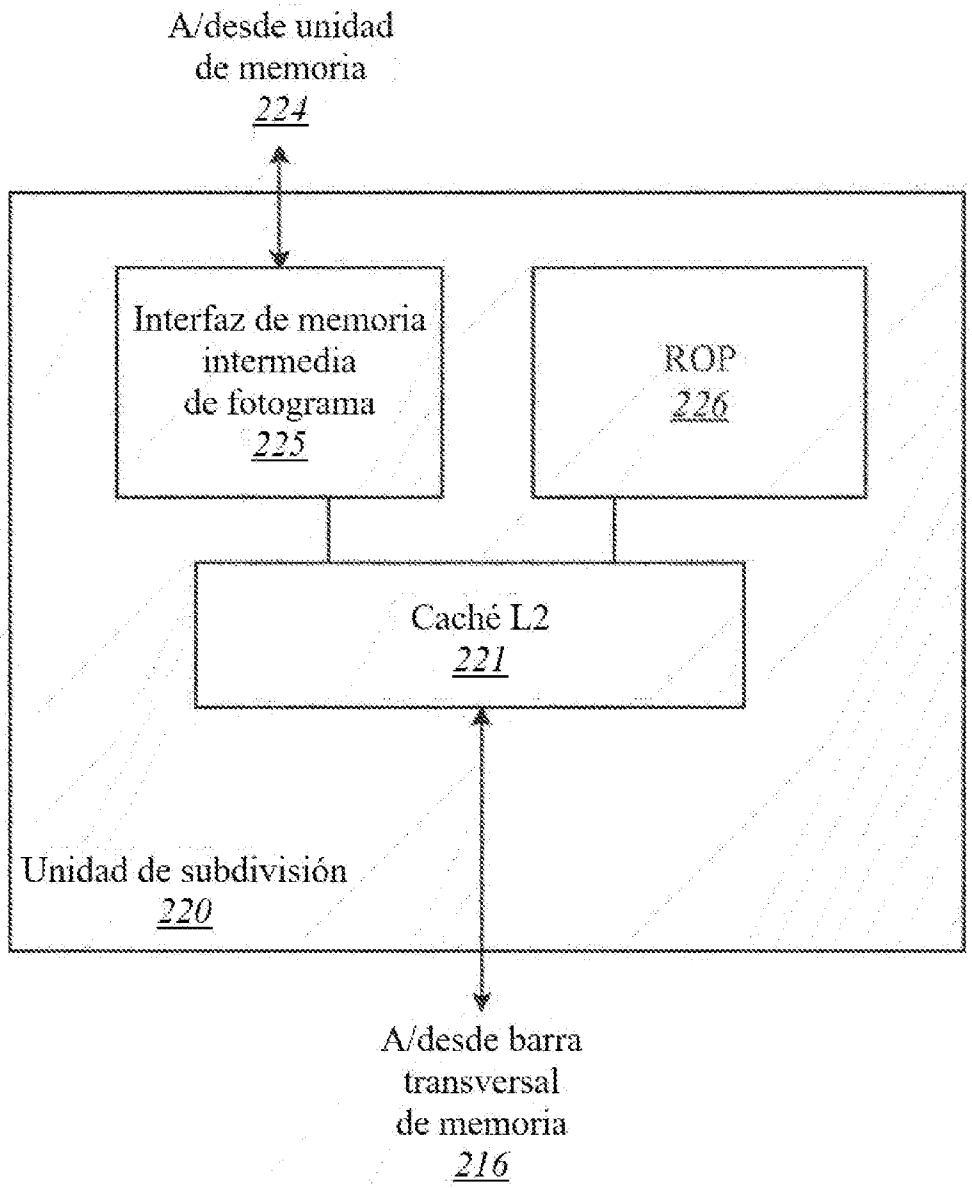

A/desde unidad
de memoria
224
Interfaz de memoria
intermedia
de fotograma
225
ROP
226
Caché L2
221
Unidad de subdivisión
220
A/desde barra
transversal
de memoria
216

FIG. 2B

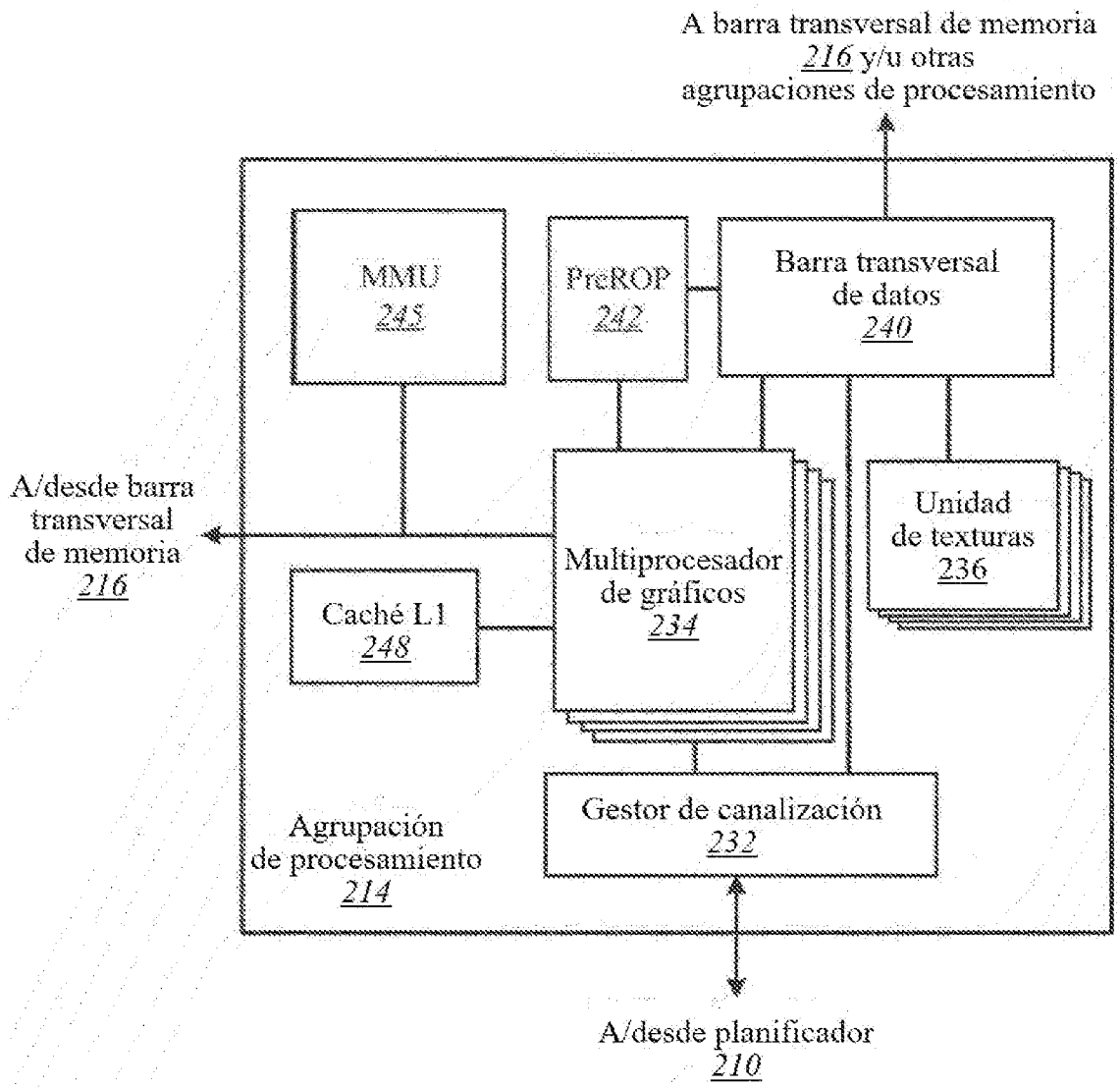

A barra transversal de memoria
216 y/u otras
agrupaciones de procesamiento
MMU
245
PreROP
242
Barra transversal
de datos
240
A/desde barra
transversal
de memoria
216
Multiprocesador
de gráficos
234
Unidad
de texturas
236
Caché L1
248
Gestor de canalización
232
Agrupación
de procesamiento
214
A/desde planificador
210

FIG. 2C

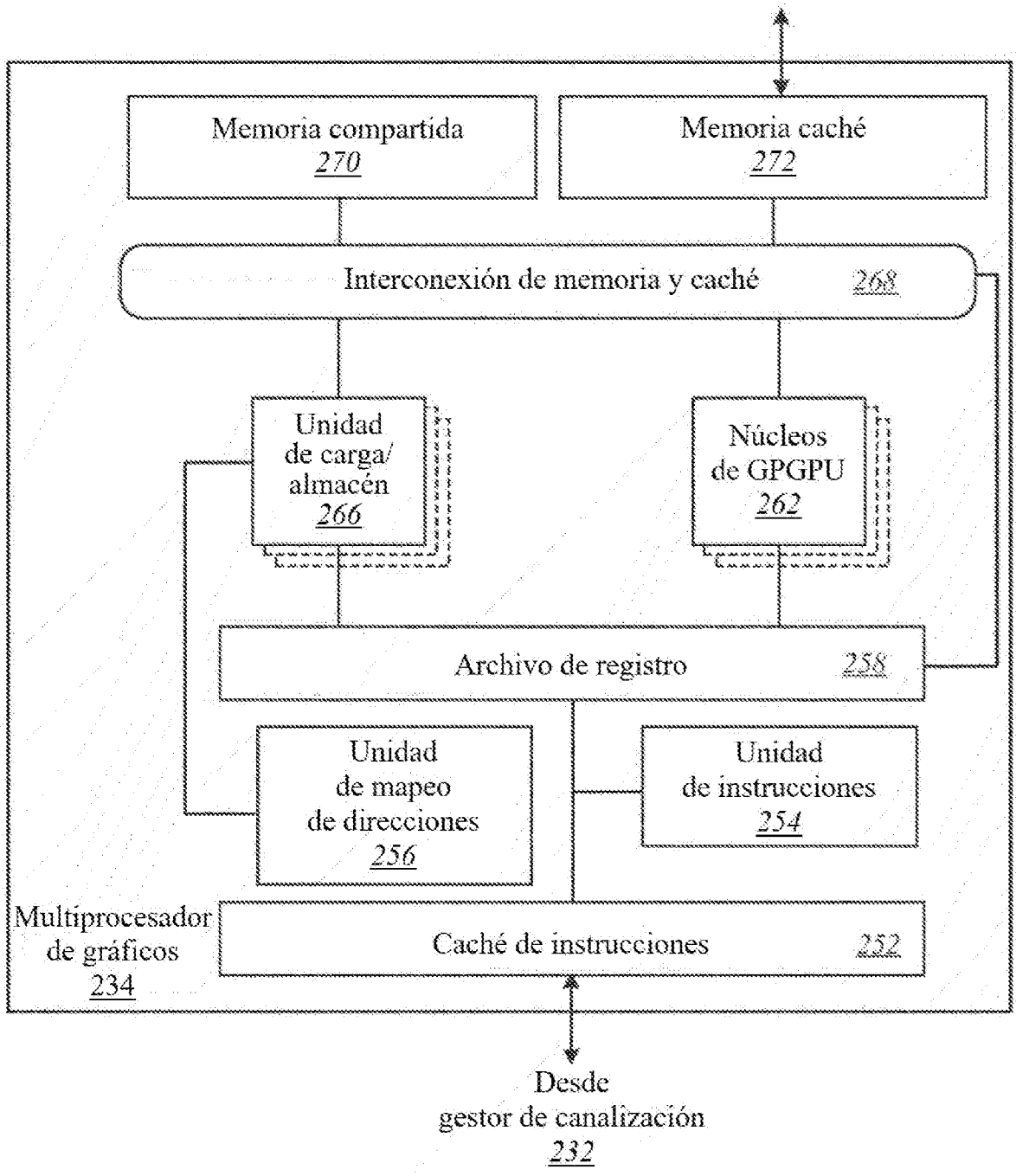

Memoria compartida
270
Memoria caché
272
Interconexión de memoria y caché
268
Unidad
de carga/
almacén
266
Núcleos
de GPGPU
262
Archivo de registro
258
Unidad
de mapeo
de direcciones
256
Unidad
de instrucciones
254
Multiprocesador
de gráficos
234
Caché de instrucciones
252
Desde
gestor de canalización
232

FIG. 2D

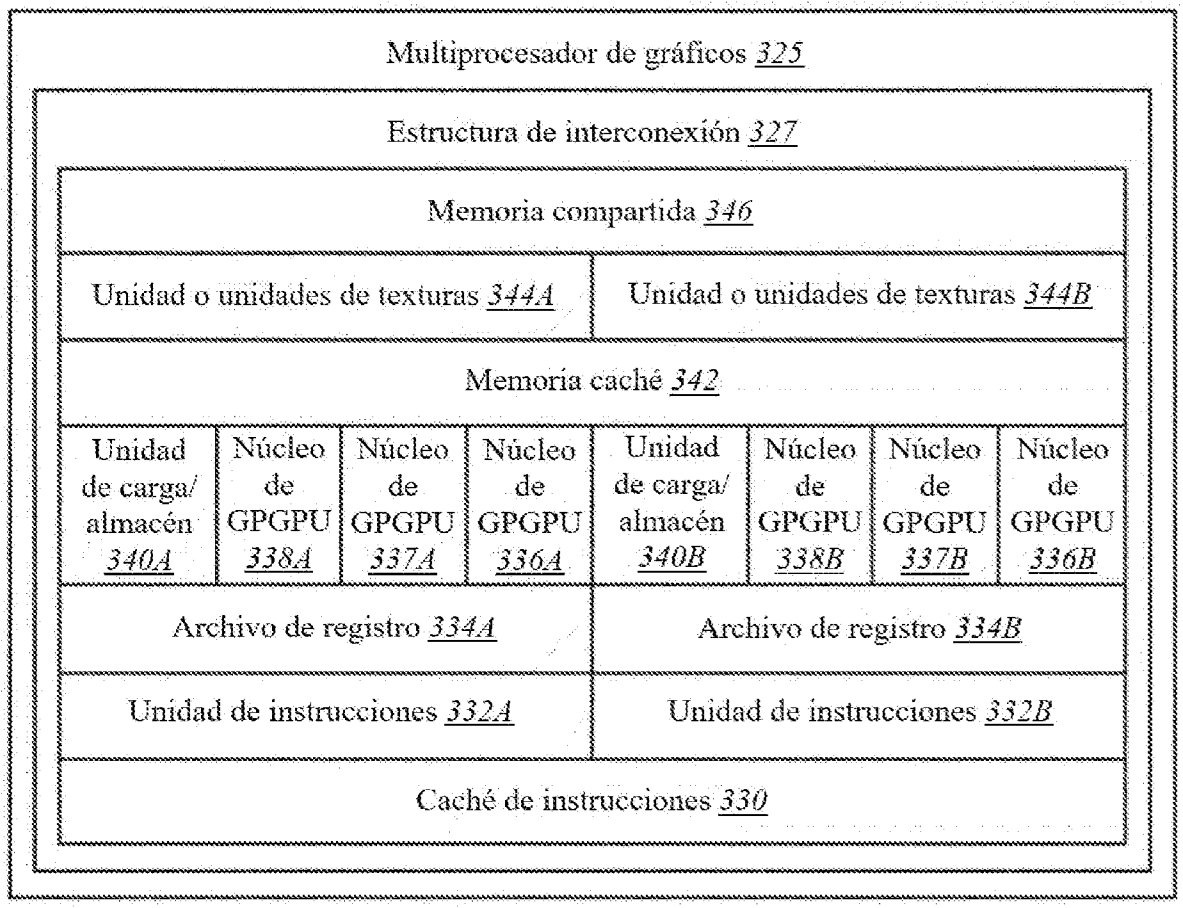

Multiprocesador de gráficos 325
Estructura de interconexión 327
Memoria compartida 346
Unidad o unidades de texturas 344A
Unidad o unidades de texturas 344B
Memoria caché 342
Unidad de carga/ almacén 340A
Núcleo de GPGPU 338A
Núcleo de GPGPU 337A
Núcleo de GPGPU 336A
Unidad de carga/ almacén 340B
Núcleo de GPGPU 338B
Núcleo de GPGPU 337B
Núcleo de GPGPU 336B
Archivo de registro 334A
Archivo de registro 334B
Unidad de instrucciones 332A
Unidad de instrucciones 332B
Caché de instrucciones 330

FIG. 3A


Multiprocesador de gráficos 350
Estructura de interconexión 352
Memoria compartida 362
Unidad o unidades de texturas 360A
Unidad o unidades de texturas 360B
Memoria caché 358A
Recursos de ejecución 356A
Recursos de ejecución 356B
Unidad o unidades de texturas 360C
Unidad o unidades de texturas 360D
Memoria caché 358B
Recursos de ejecución 356C
Recursos de ejecución 356D
Caché de instrucciones 354

FIG. 3B

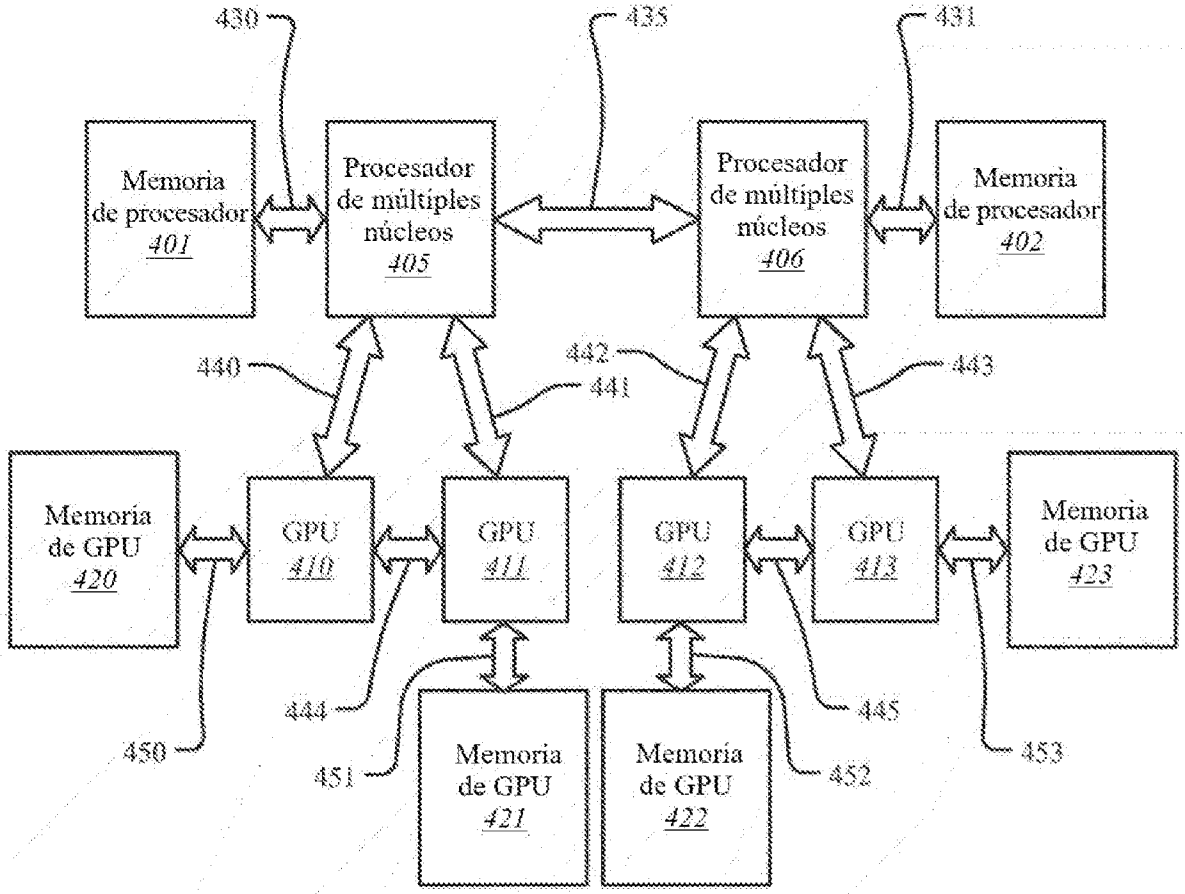

430
435
431
Memoria de procesador 401
Procesador de múltiples núcleos 405
Procesador de múltiples núcleos 406
Memoria de procesador 402
440
442
441
443
Memoria de GPU 420
GPU 410
GPU 411
GPU 412
GPU 413
Memoria de GPU 423
444
445
450
451
452
453
Memoria de GPU 421
Memoria de GPU 422

FIG. 4A

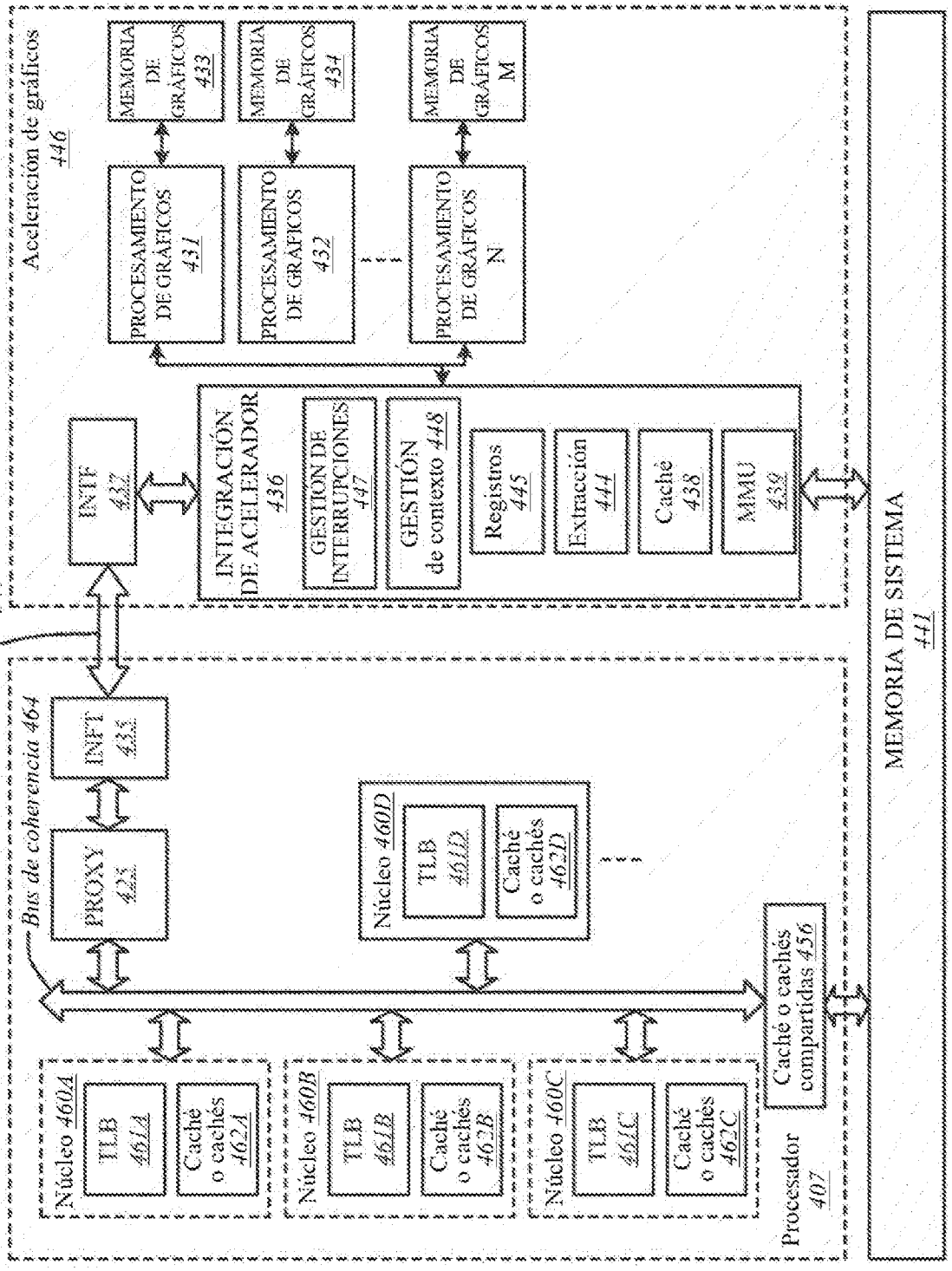

Aceleración de gráficos 446
MEMORIA DE GRÁFICOS 433
MEMORIA DE GRÁFICOS 434
MEMORIA DE GRÁFICOS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
INTEGRACIÓN DE ACELERADOR 436
GESTIÓN DE INTERRUPCIONES 447
GESTIÓN de contexto 448
Registros 445
Extracción 444
Caché 438
MMU 439
INTF 437
340
Bus de coherencia 464
PROXY 425
INFT 435
Núcleo 460A
TLB 461A
Caché o cachés 462A
Núcleo 460B
TLB 461B
Caché o cachés 462B
Núcleo 460C
TLB 461C
Caché o cachés 462C
Núcleo 460D
TLB 461D
Caché o cachés 462D
Caché o cachés compartidas 456
Procesador 407
MEMORIA DE SISTEMA 441

FIG. 4B

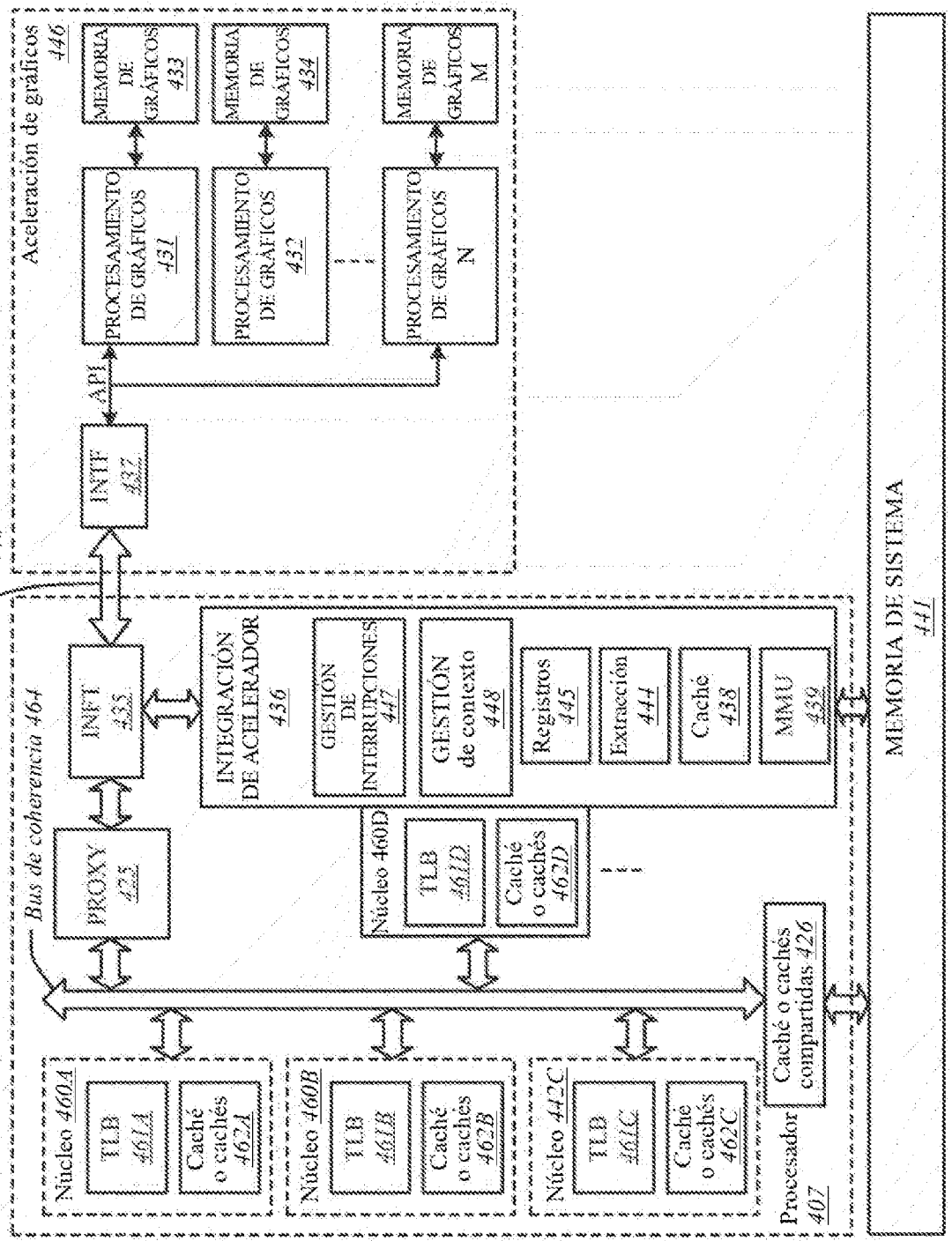

Aceleración de gráficos 446
MEMORIA DE GRÁFICOS 433
MEMORIA DE GRÁFICOS 434
MEMORIA DE GRÁFICOS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 427
449
INFT 425
PROXY 423
Bus de coherencia 464
INTEGRACIÓN DE ACELERADOR 436
GESTIÓN DE INTERRUPCIONES 447
GESTIÓN de contexto 448
Registros 445
Extracción 444
Caché 438
MMU 439
Núcleo 460D
TLB 461D
Caché o cachés 462D
Núcleo 460A
TLB 461A
Caché o cachés 462A
Núcleo 460B
TLB 461B
Caché o cachés 462B
Núcleo 442C
TLB 461C
Caché o cachés 462C
Procesador 407
Caché o cachés compartidas 426
MEMORIA DE SISTEMA 441

FIG. 4C

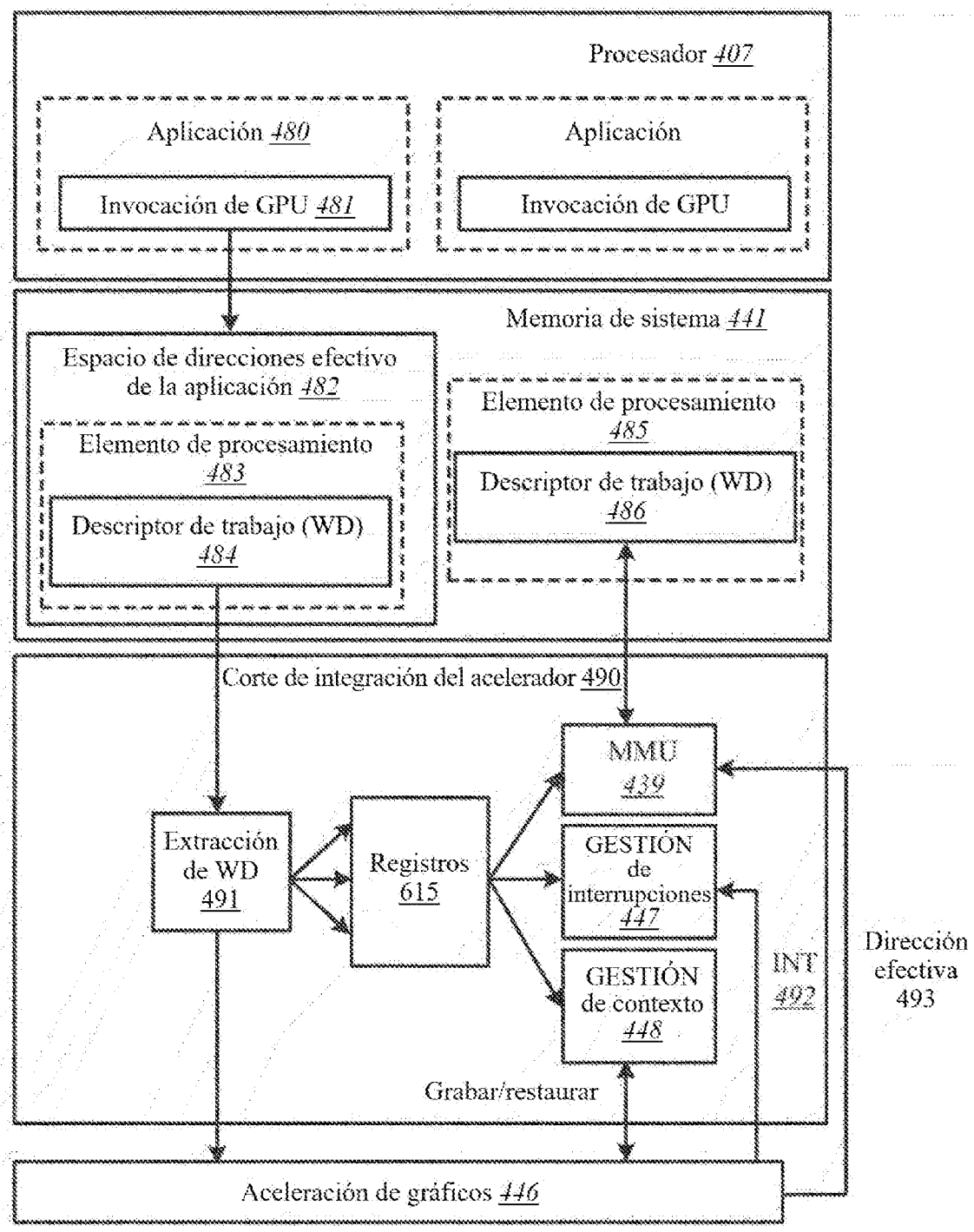

Procesador 407
Aplicación 480
Invocación de GPU 481
Aplicación
Invocación de GPU
Memoria de sistema 441
Espacio de direcciones efectivo de la aplicación 482
Elemento de procesamiento 483
Descriptor de trabajo (WD) 484
Elemento de procesamiento 485
Descriptor de trabajo (WD) 486
Corte de integración del acelerador 490
MMU 439
Extracción de WD 491
Registros 615
GESTIÓN de interrupciones 447
GESTIÓN de contexto 448
INT 492
Dirección efectiva 493
Grabar/restaurar
Aceleración de gráficos 446

FIG. 4D

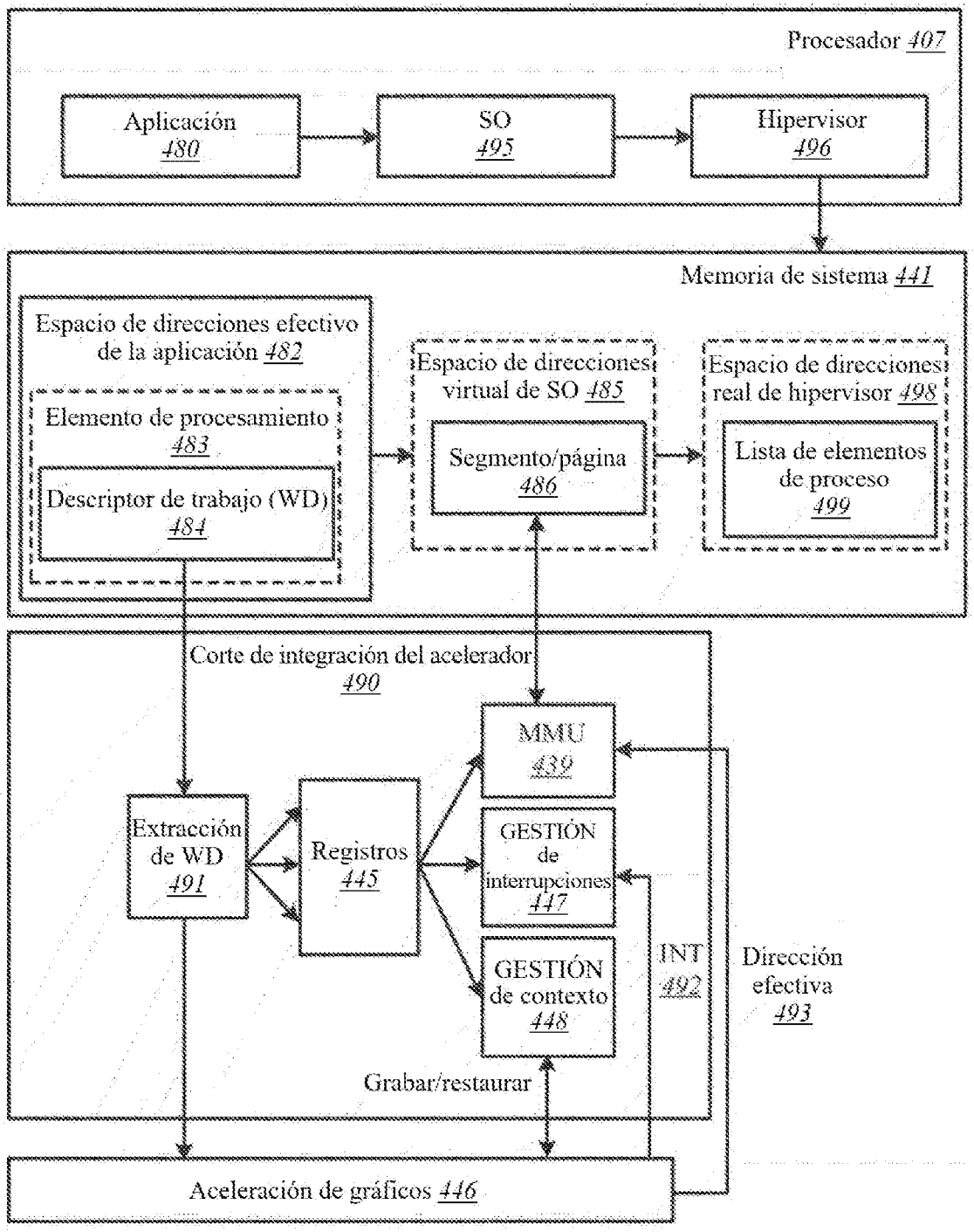

Procesador 407
Aplicación 480
SO 495
Hipervisor 496
Memoria de sistema 441
Espacio de direcciones efectivo de la aplicación 482
Elemento de procesamiento 483
Descriptor de trabajo (WD) 484
Espacio de direcciones virtual de SO 485
Segmento/página 486
Espacio de direcciones real de hipervisor 498
Lista de elementos de proceso 499
Corte de integración del acelerador 490
MMU 439
Extracción de WD 491
Registros 445
GESTIÓN de interrupciones 447
GESTIÓN de contexto 448
INT 492
Dirección efectiva 493
Grabar/restaurar
Aceleración de gráficos 446

FIG. 4E

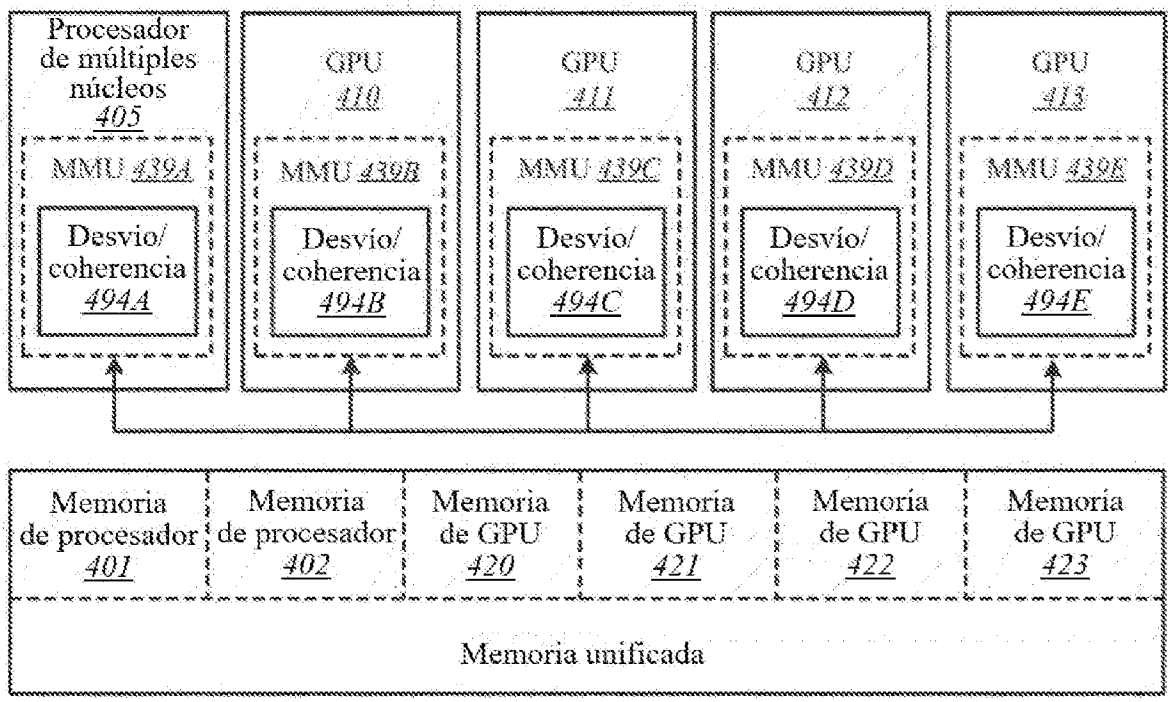

Procesador de múltiples núcleos 405
MMU 439A
Desvío/ coherencia 494A
GPU 410
MMU 439B
Desvío/ coherencia 494B
GPU 411
MMU 439C
Desvío/ coherencia 494C
GPU 412
MMU 439D
Desvío/ coherencia 494D
GPU 413
MMU 439E
Desvío/ coherencia 494E
Memoria de procesador 401
Memoria de procesador 402
Memoria de GPU 420
Memoria de GPU 421
Memoria de GPU 422
Memoria de GPU 423
Memoria unificada

FIG. 4F

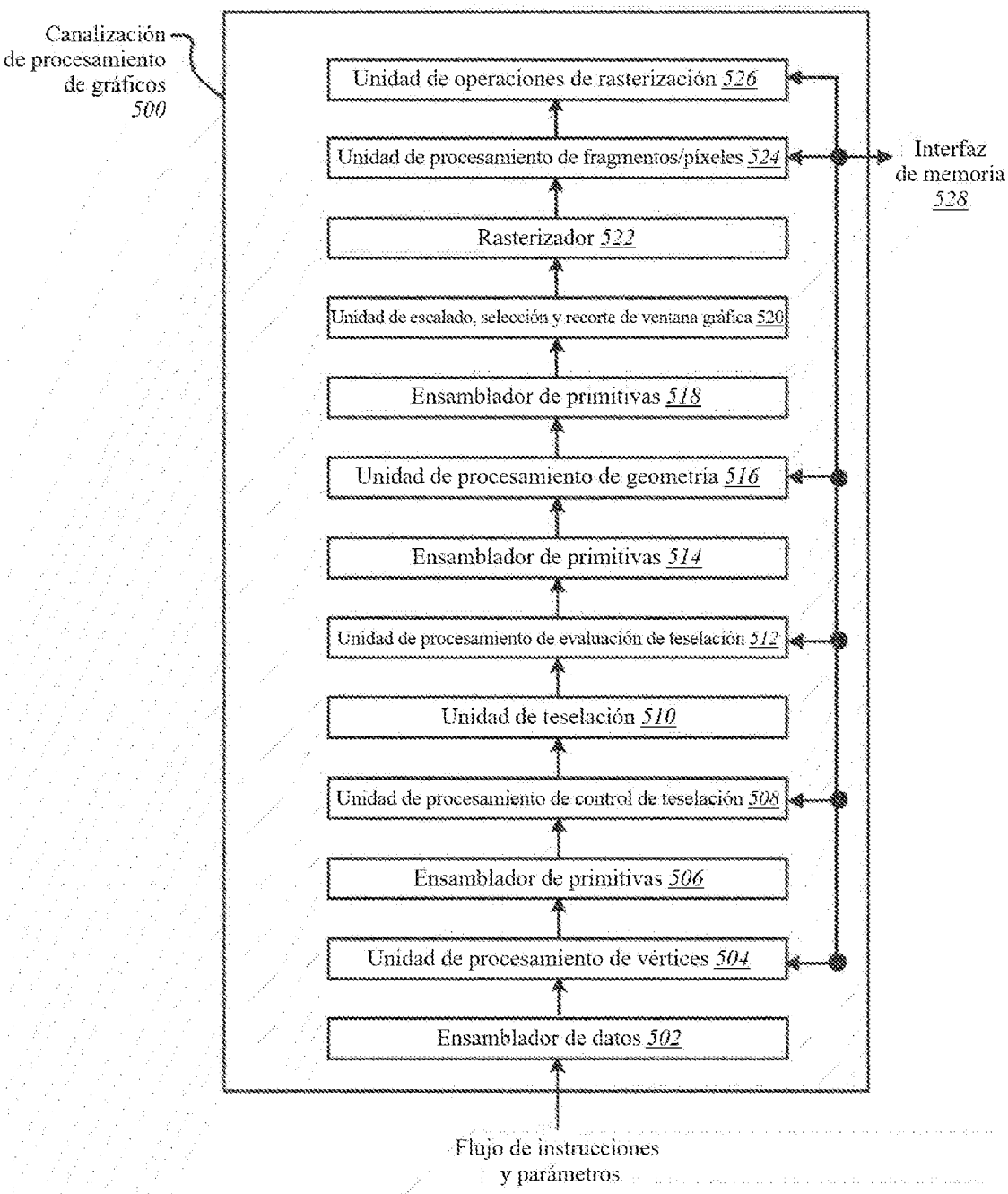

Canalización de procesamiento de gráficos 500
Unidad de operaciones de rasterización 526
Unidad de procesamiento de fragmentos/píxeles 524
Interfaz de memoria 528
Rasterizador 522
Unidad de escalado, selección y recorte de ventana gráfica 520
Ensamblador de primitivas 518
Unidad de procesamiento de geometría 516
Ensamblador de primitivas 514
Unidad de procesamiento de evaluación de teselación 512
Unidad de teselación 510
Unidad de procesamiento de control de teselación 508
Ensamblador de primitivas 506
Unidad de procesamiento de vértices 504
Ensamblador de datos 502
Flujo de instrucciones y parámetros

FIG. 5

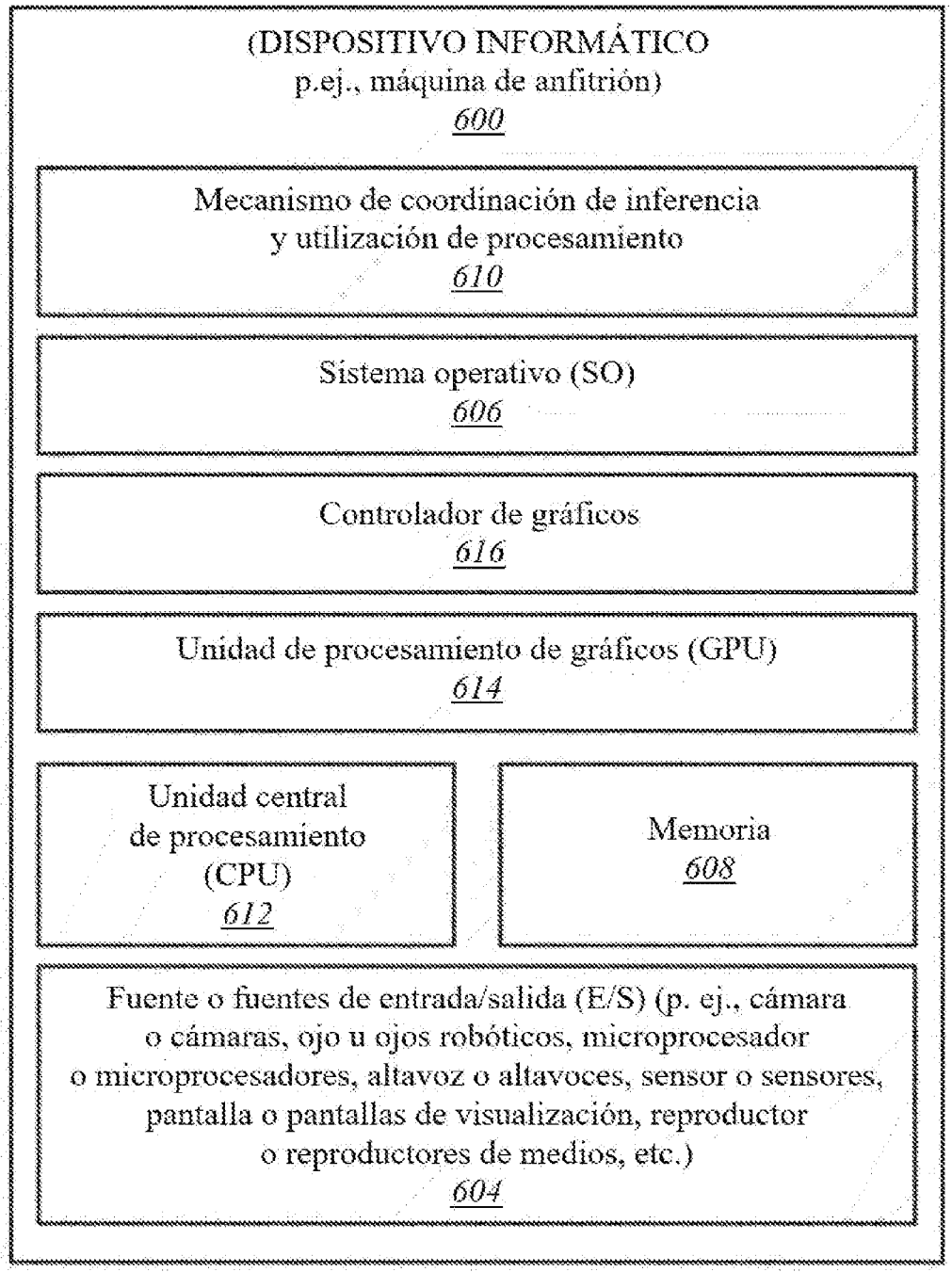

(DISPOSITIVO INFORMÁTICO
p.ej., máquina de anfitrión)
600
Mecanismo de coordinación de inferencia y utilización de procesamiento
610
Sistema operativo (SO)
606
Controlador de gráficos
616
Unidad de procesamiento de gráficos (GPU)
614
Unidad central de procesamiento (CPU)
612
Memoria
608
Fuente o fuentes de entrada/salida (E/S) (p. ej., cámara o cámaras, ojo u ojos robóticos, microprocesador o microprocesadores, altavoz o altavoces, sensor o sensores, pantalla o pantallas de visualización, reproductor o reproductores de medios, etc.)
604

FIG. 6

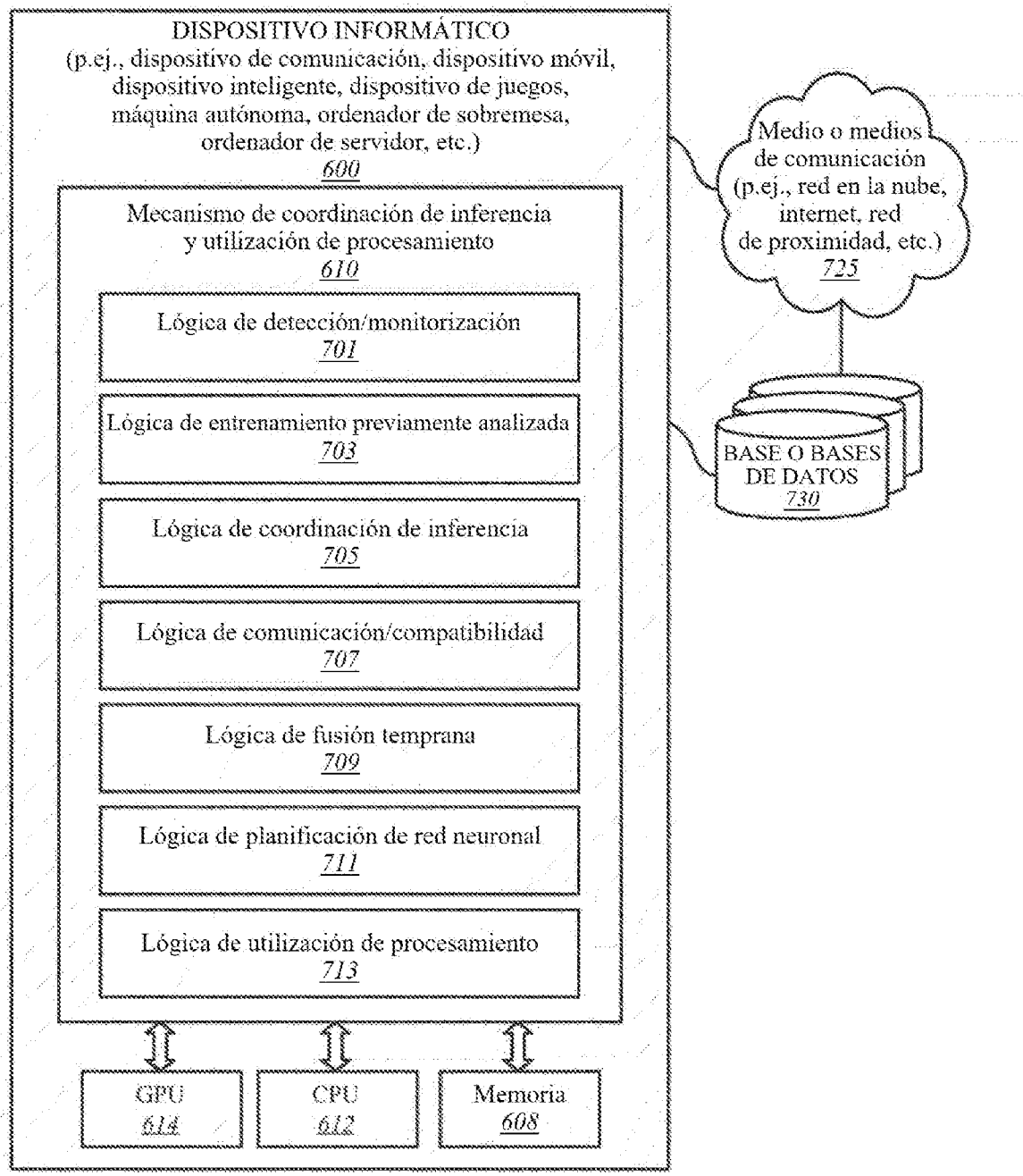

DISPOSITIVO INFORMÁTICO
(p.ej., dispositivo de comunicación, dispositivo móvil, dispositivo inteligente, dispositivo de juegos, máquina autónoma, ordenador de sobremesa, ordenador de servidor, etc.)
600
Mecanismo de coordinación de inferencia y utilización de procesamiento
610
Lógica de detección/monitorización
701
Lógica de entrenamiento previamente analizada
703
Lógica de coordinación de inferencia
705
Lógica de comunicación/compatibilidad
707
Lógica de fusión temprana
709
Lógica de planificación de red neuronal
711
Lógica de utilización de procesamiento
713
GPU
614
CPU
612
Memoria
608
Medio o medios de comunicación (p.ej., red en la nube, internet, red de proximidad, etc.)
725
BASE O BASES DE DATOS
730

FIG. 7

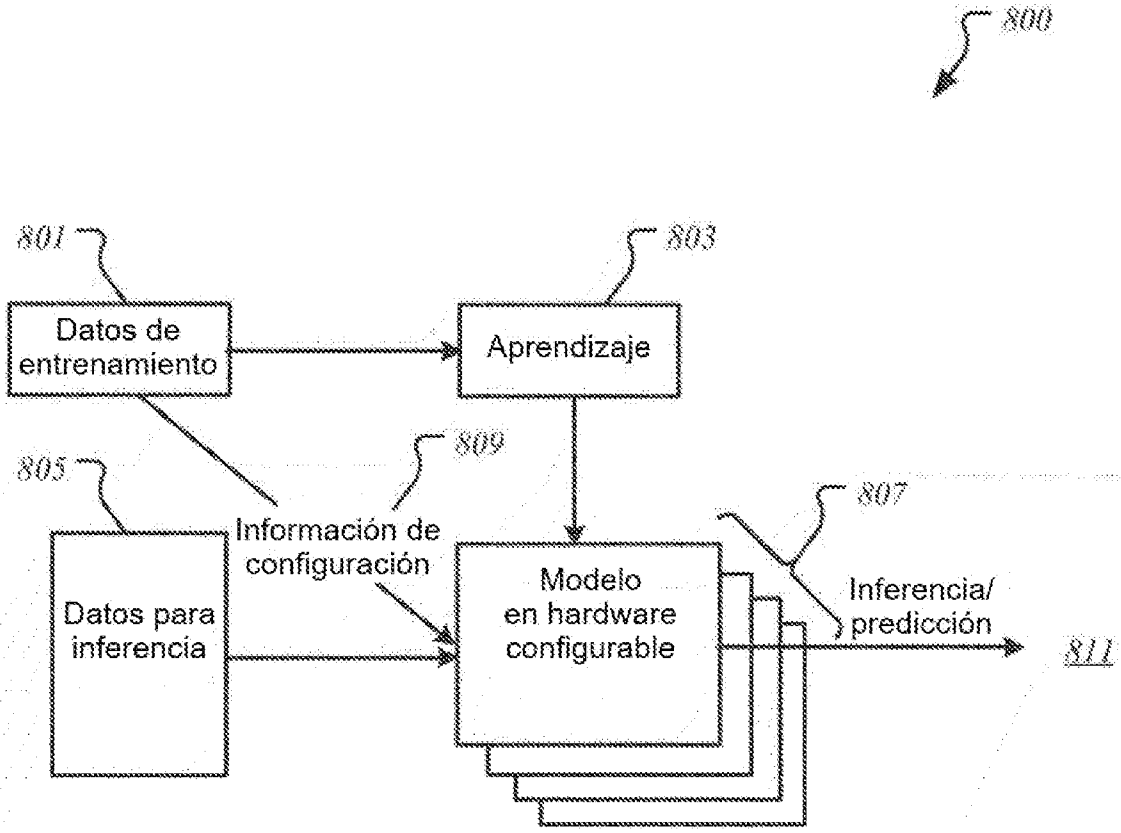

800
801
Datos de entrenamiento
803
Aprendizaje
805
Datos para inferencia
809
Información de configuración
807
Modelo en hardware configurable
Inferencia/ predicción
811

FIG. 8A

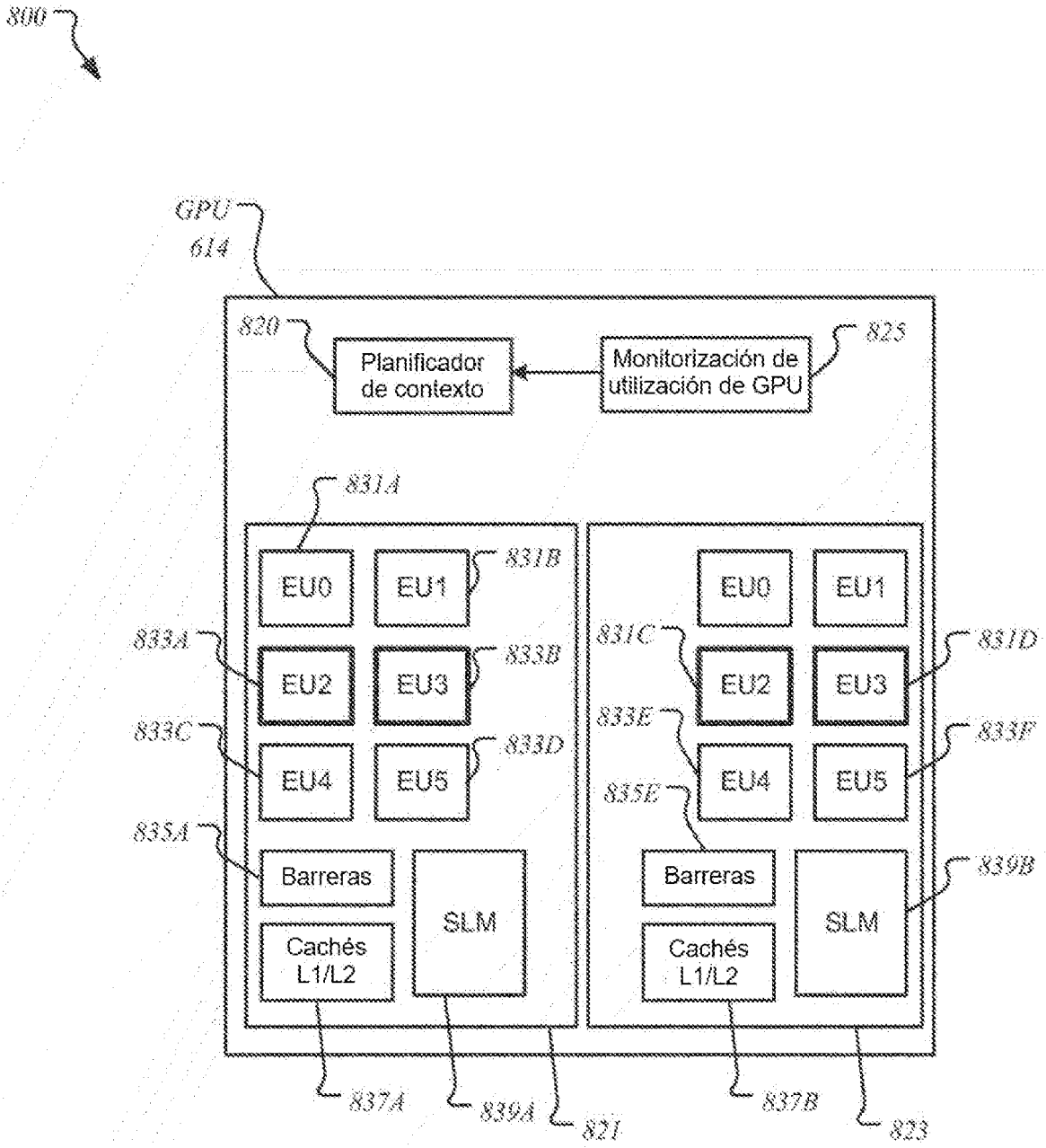

800
GPU
614
820
Planificador de contexto
825
Monitorización de utilización de GPU
831A
831B
831C
831D
833A
833B
833C
833D
833E
833F
835A
835E
839B
EU0
EU1
EU2
EU3
EU4
EU5
Barreras
SLM
Cachés L1/L2
837A
839A
821
837B
823

FIG. 8B

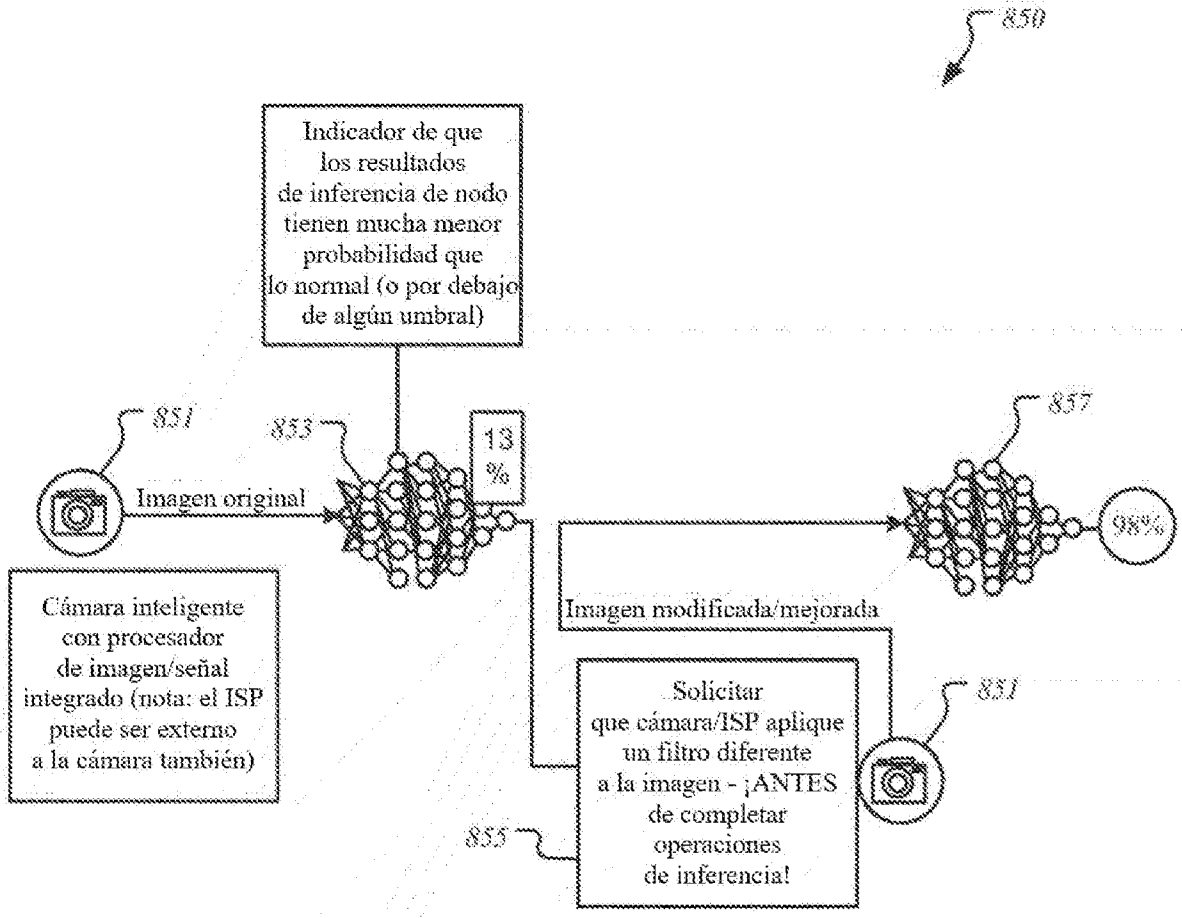

850
Indicador de que los resultados de inferencia de nodo tienen mucha menor probabilidad que lo normal (o por debajo de algún umbral)
851
853
13 %
Imagen original
857
98%
Cámara inteligente con procesador de imagen/señal integrado (nota: el ISP puede ser externo a la cámara también)
Imagen modificada/mejorada
Solicitar que cámara/ISP aplique un filtro diferente a la imagen - ¡ANTES de completar operaciones de inferencia!
851
855

FIG. 8C

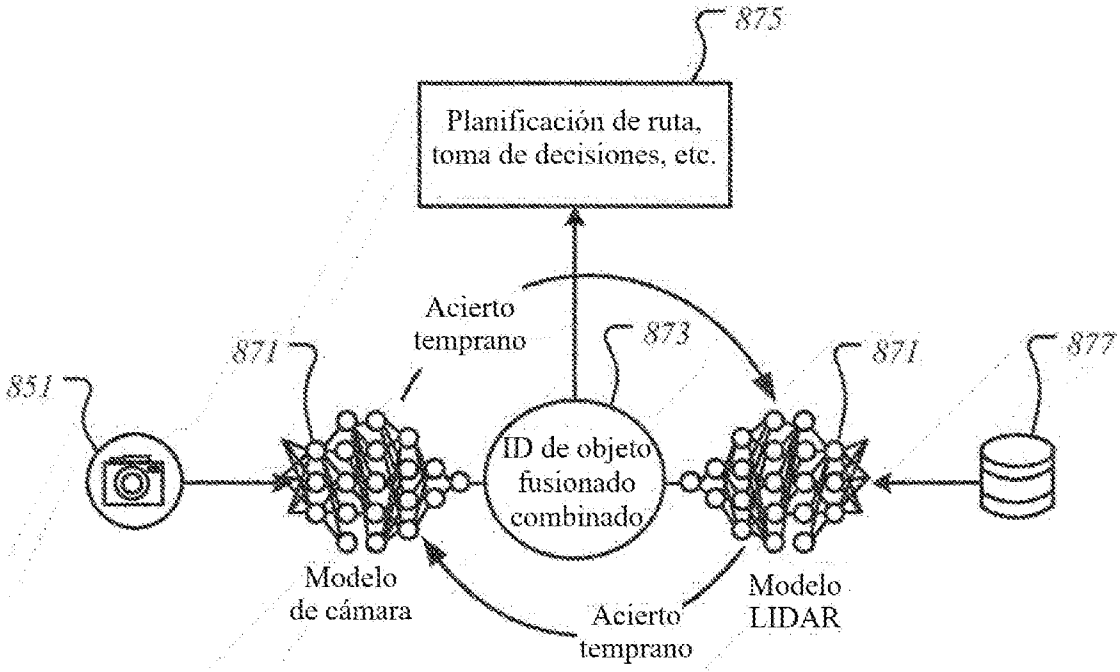

875
Planificación de ruta, toma de decisiones, etc.
Acierto temprano
873
871
871
877
851
ID de objeto fusionado combinado
Modelo de cámara
Modelo LIDAR
Acierto temprano

FIG. 8D

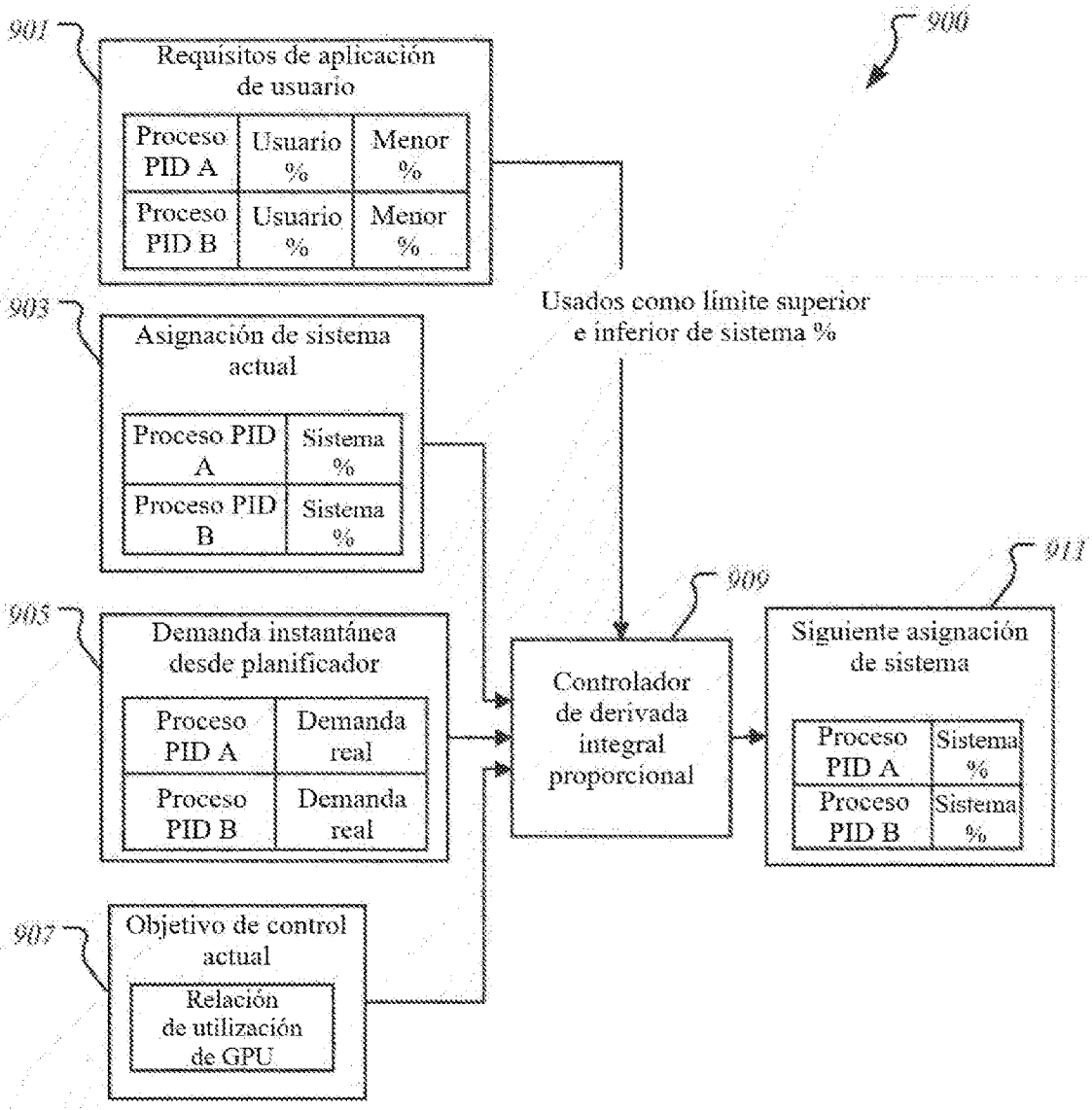

900
901
Requisitos de aplicación de usuario
Proceso PID A
Usuario %
Menor %
Proceso PID B
Usuario %
Menor %
Usados como límite superior e inferior de sistema %
903
Asignación de sistema actual
Proceso PID A
Sistema %
Proceso PID B
Sistema %
905
Demanda instantánea desde planificador
Proceso PID A
Demanda real
Proceso PID B
Demanda real
907
Objetivo de control actual
Relación de utilización de GPU
909
Controlador de derivada integral proporcional
911
Siguiente asignación de sistema
Proceso PID A
Sistema %
Proceso PID B
Sistema %

FIG. 9A

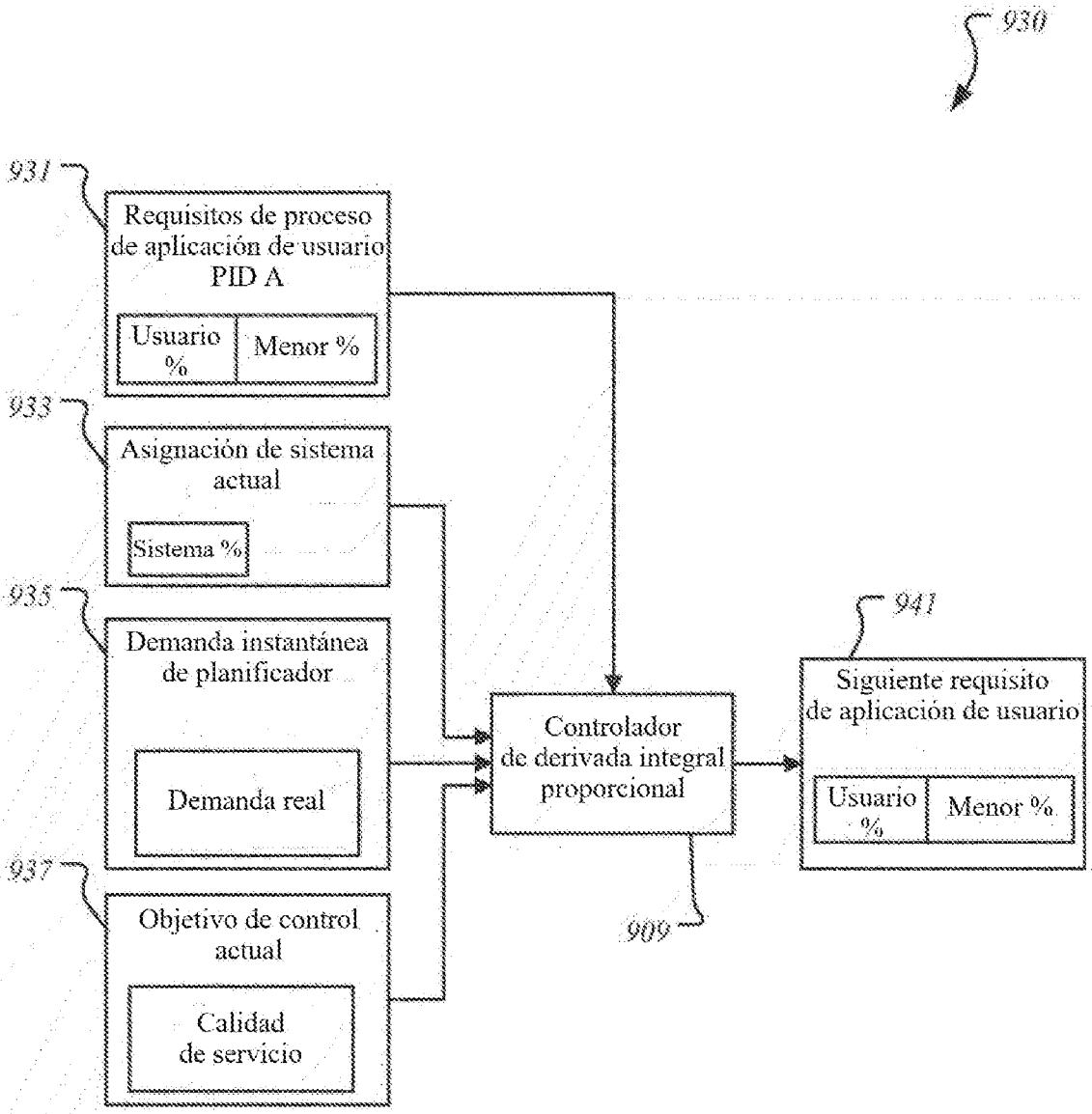

930
931
Requisitos de proceso de aplicación de usuario PID A
Usuario %
Menor %
933
Asignación de sistema actual
Sistema %
935
Demanda instantánea de planificador
Demanda real
937
Objetivo de control actual
Calidad de servicio
Controlador de derivada integral proporcional
909
941
Siguiente requisito de aplicación de usuario
Usuario %
Menor %

FIG. 9B

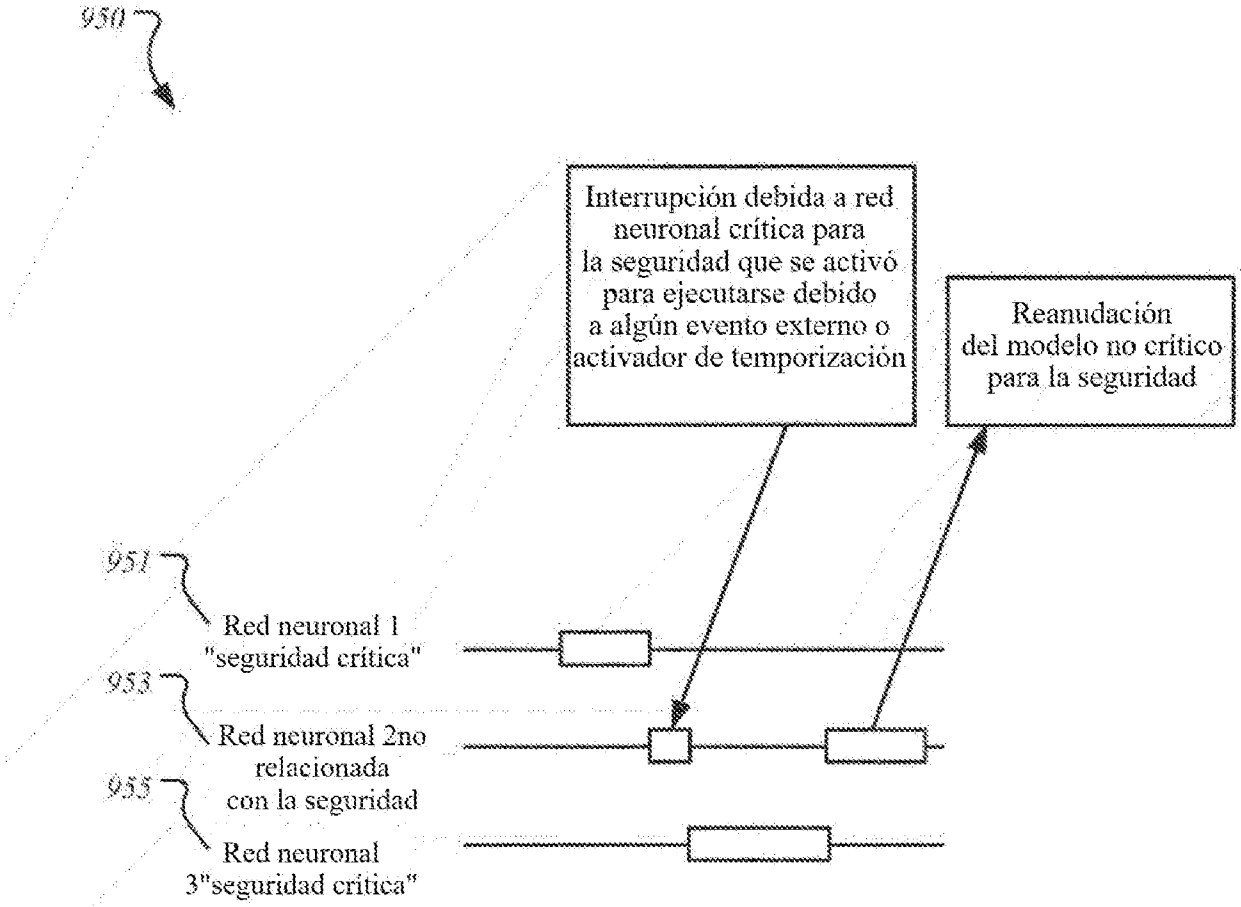

950
Interrupción debida a red neuronal crítica para la seguridad que se activó para ejecutarse debido a algún evento externo o activador de temporización
Reanudación del modelo no crítico para la seguridad
951
Red neuronal 1 "seguridad crítica"
953
Red neuronal 2no relacionada con la seguridad
955
Red neuronal 3"seguridad crítica"

FIG. 9C

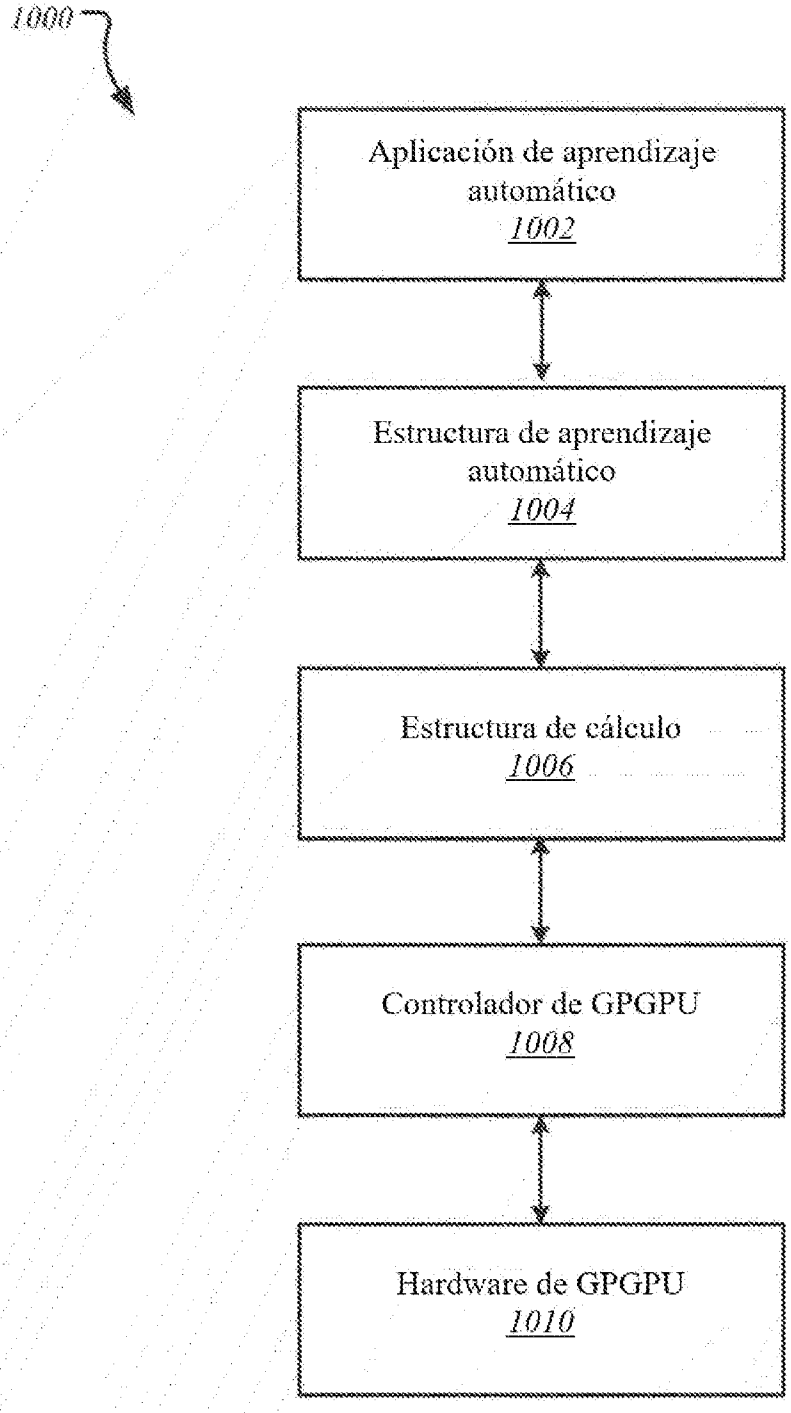

1000
Aplicación de aprendizaje automático
1002
Estructura de aprendizaje automático
1004
Estructura de cálculo
1006
Controlador de GPGPU
1008
Hardware de GPGPU
1010

FIG. 10

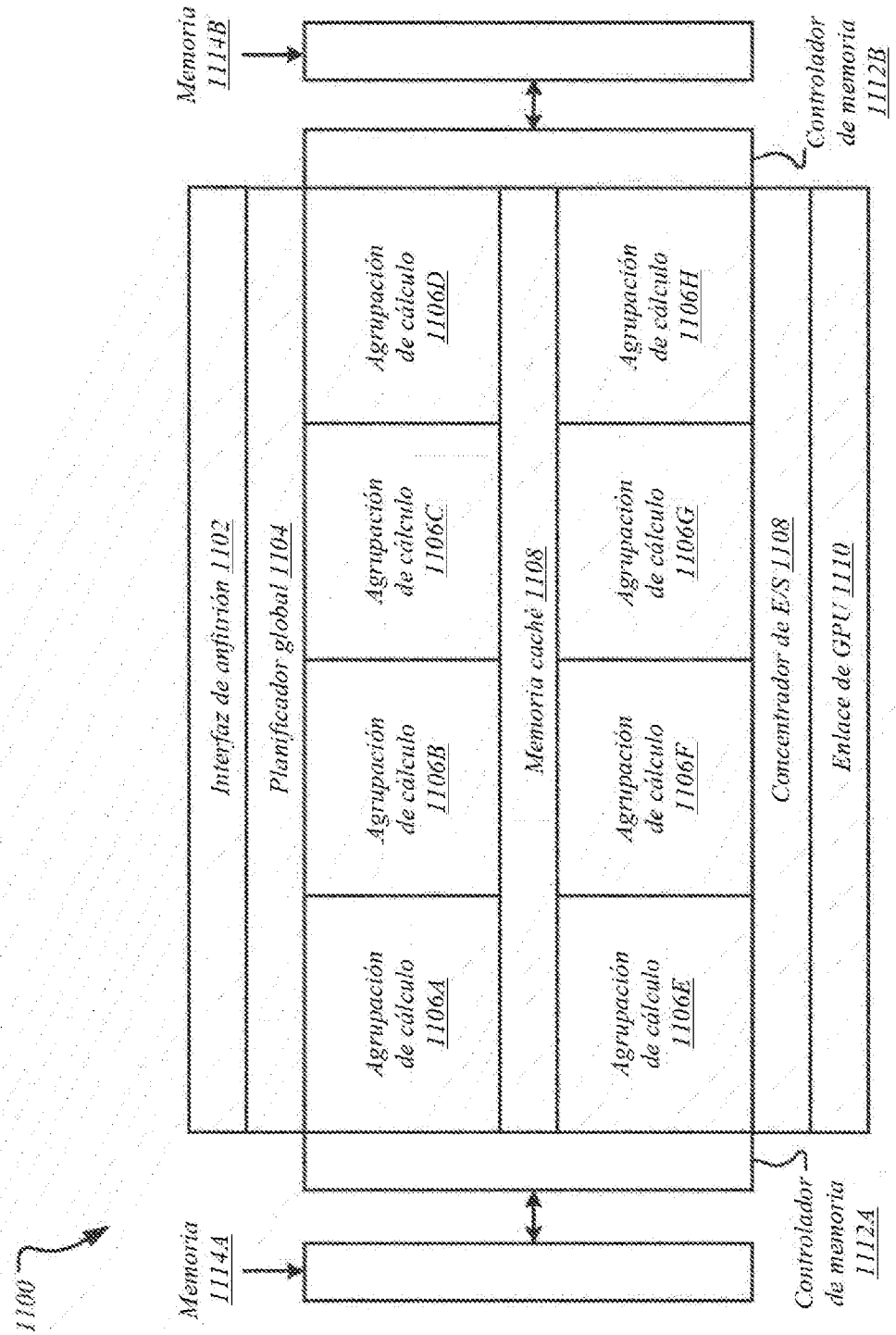

1100
Memoria 1114A
Memoria 1114B
Controlador de memoria 1112A
Controlador de memoria 1112B
Interfaz de anfitrión 1102
Planificador global 1104
Agrupación de cálculo 1106A
Agrupación de cálculo 1106B
Agrupación de cálculo 1106C
Agrupación de cálculo 1106D
Agrupación de cálculo 1106E
Agrupación de cálculo 1106F
Agrupación de cálculo 1106G
Agrupación de cálculo 1106H
Memoria caché 1108
Concentrador de E/S 1108
Enlace de GPU 1110

FIG. 11

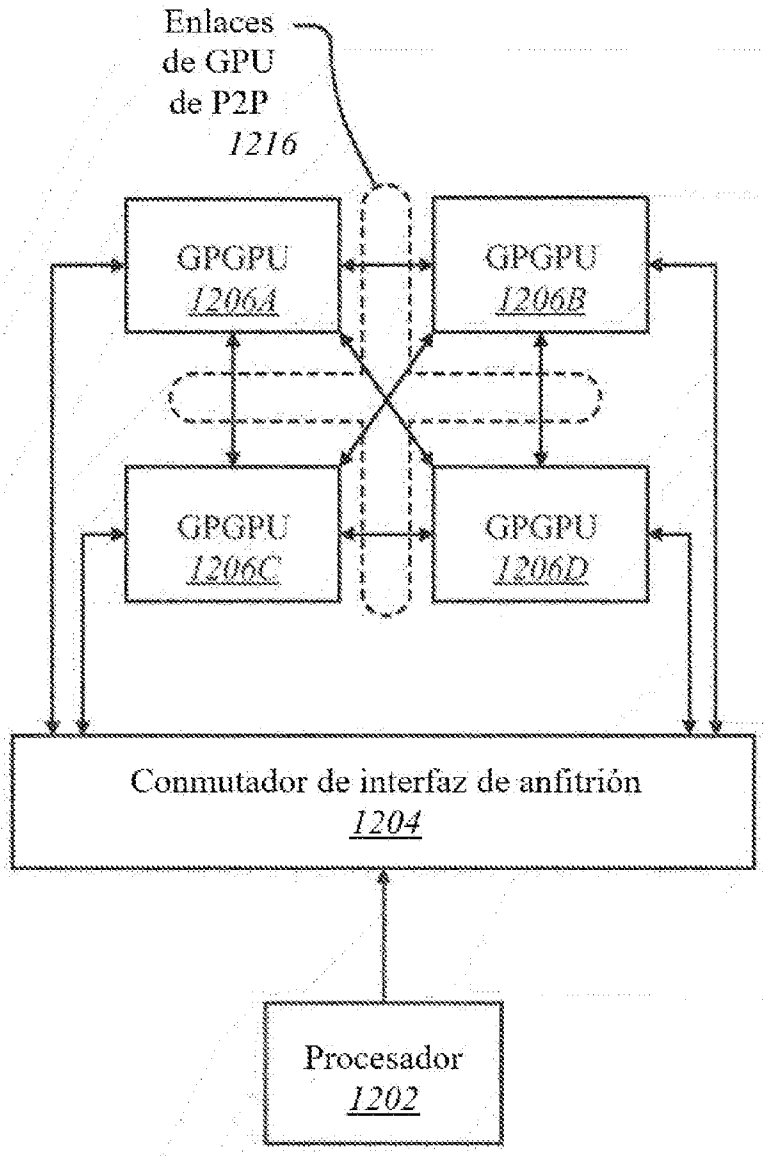

Enlaces
de GPU
de P2P
1216
GPGPU
1206A
GPGPU
1206B
GPGPU
1206C
GPGPU
1206D
Conmutador de interfaz de anfitrión
1204
Procesador
1202

FIG. 12

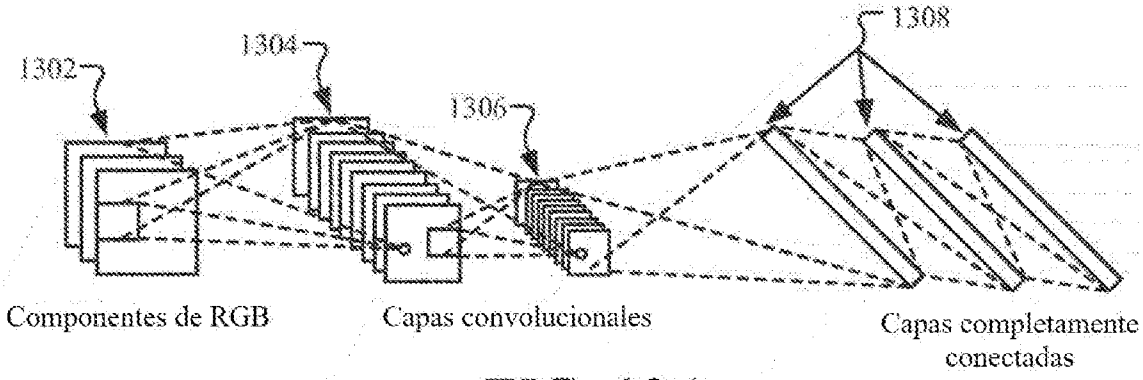


FIG. 13A

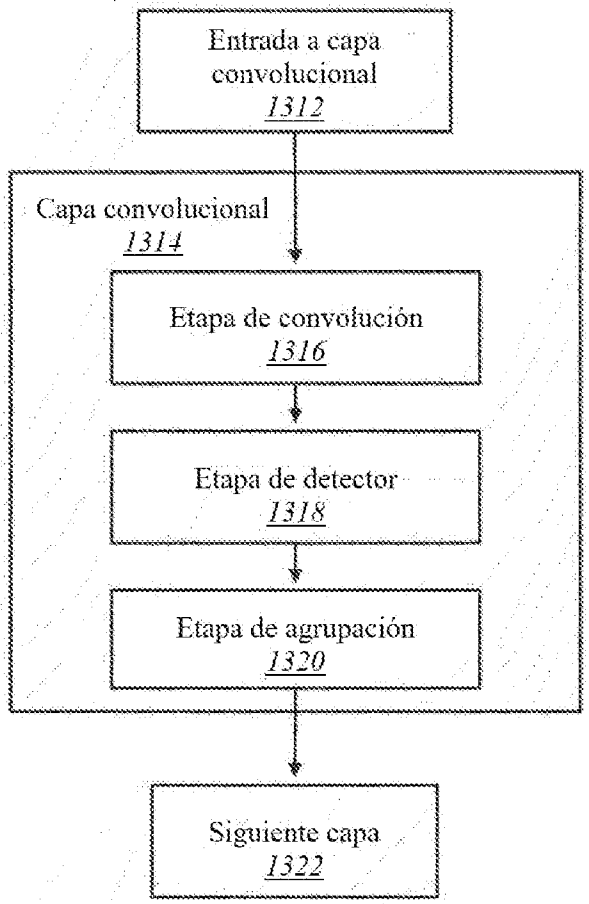


FIG. 13B

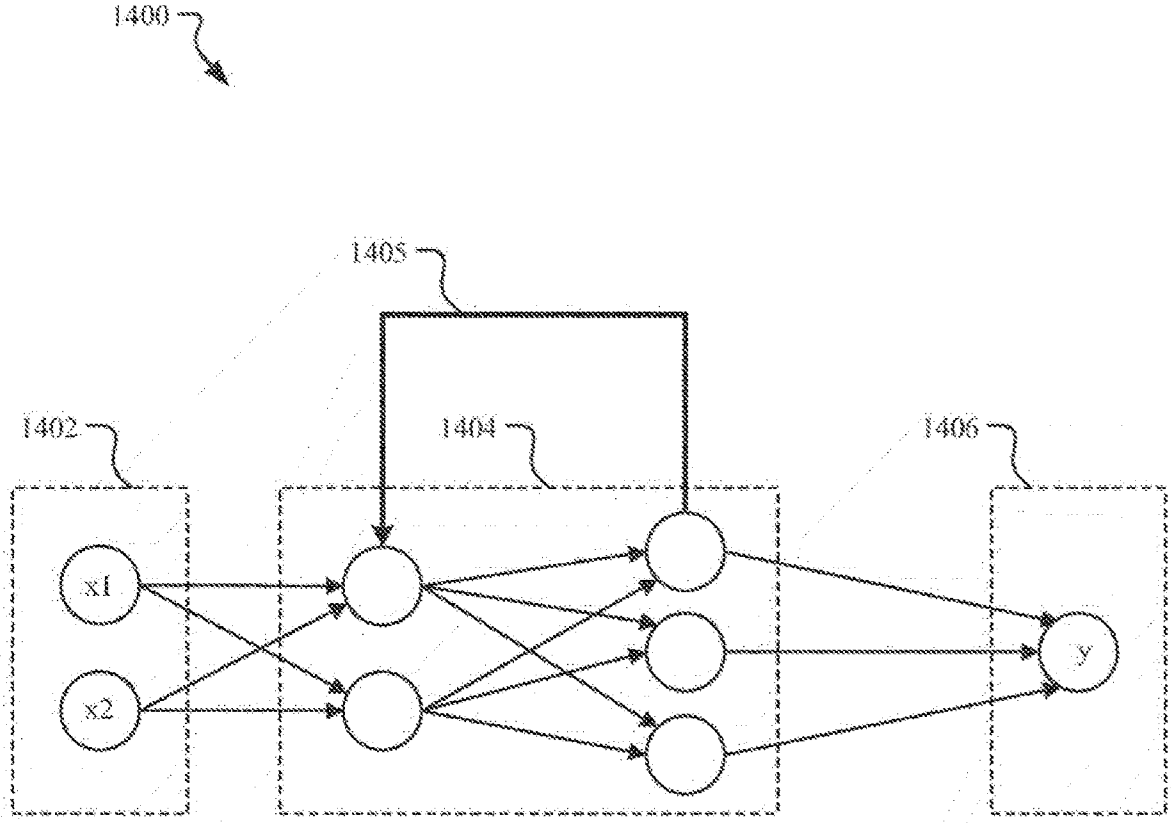


FIG. 14

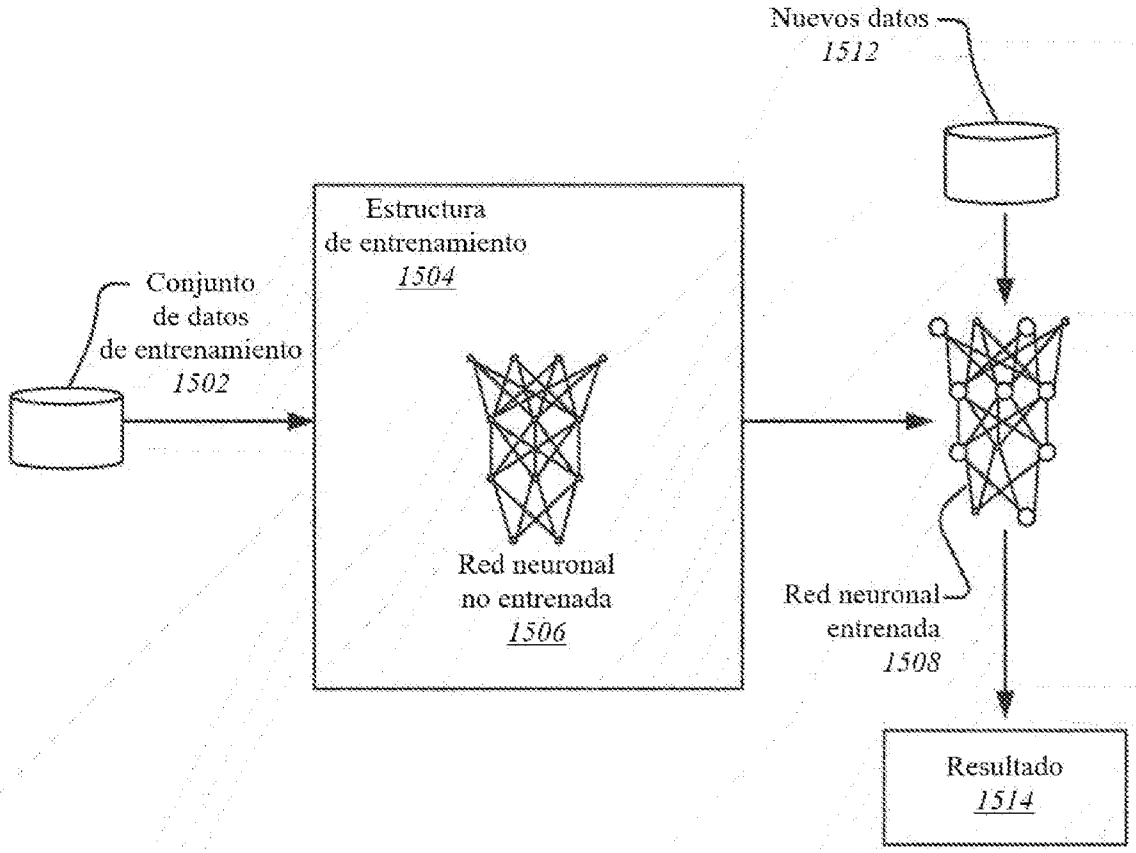

Nuevos datos
1512
Estructura
de entrenamiento
1504
Conjunto
de datos
de entrenamiento
1502
Red neuronal
no entrenada
1506
Red neuronal
entrenada
1508
Resultado
1514

FIG. 15

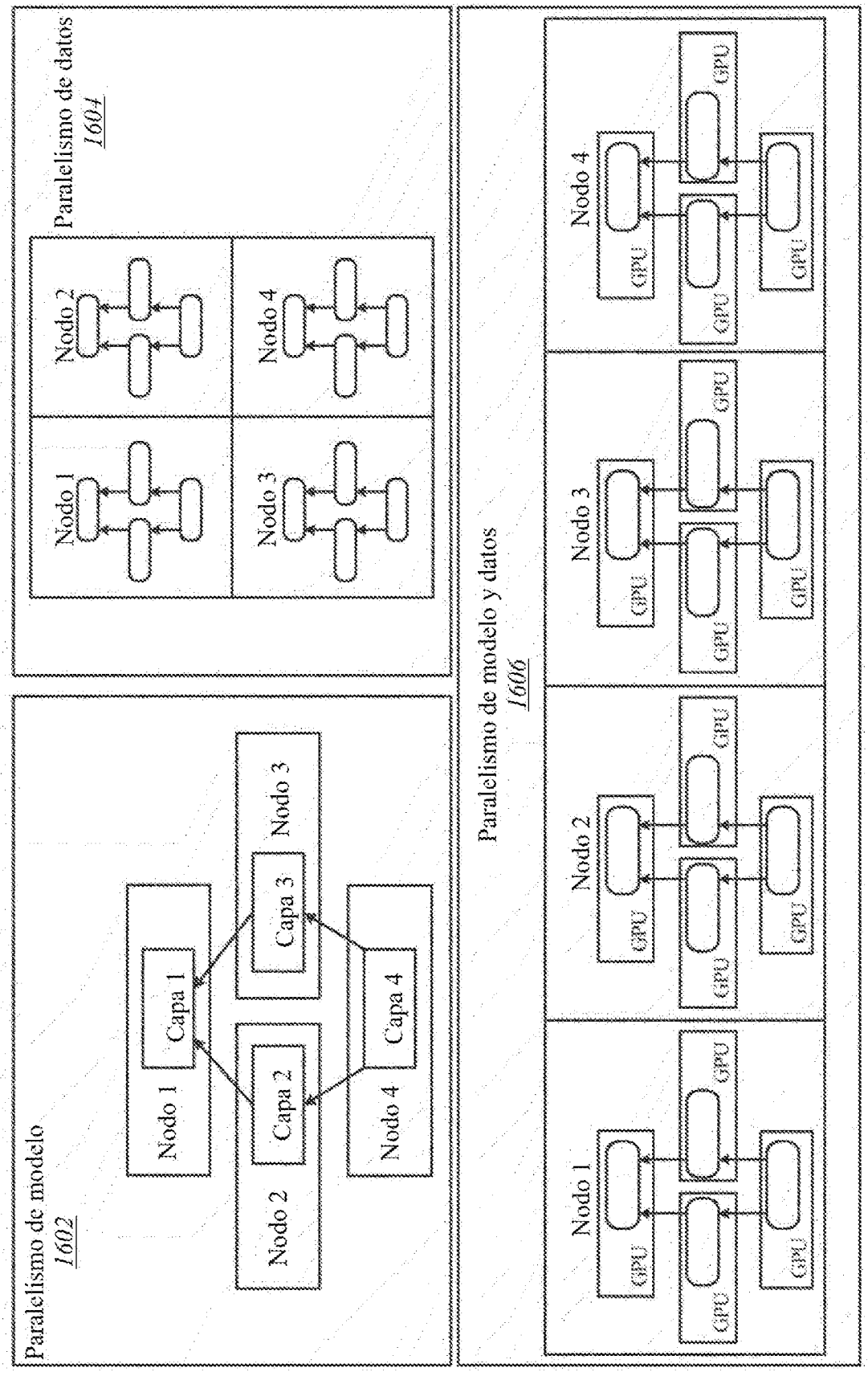

Paralelismo de modelo
1602
Nodo 1
Capa 1
Nodo 2
Capa 2
Nodo 3
Capa 3
Nodo 4
Capa 4
Paralelismo de datos
1604
Nodo 1
Nodo 2
Nodo 3
Nodo 4
Paralelismo de modelo y datos
1606
Nodo 1
Nodo 2
Nodo 3
Nodo 4
GPU

FIG. 16

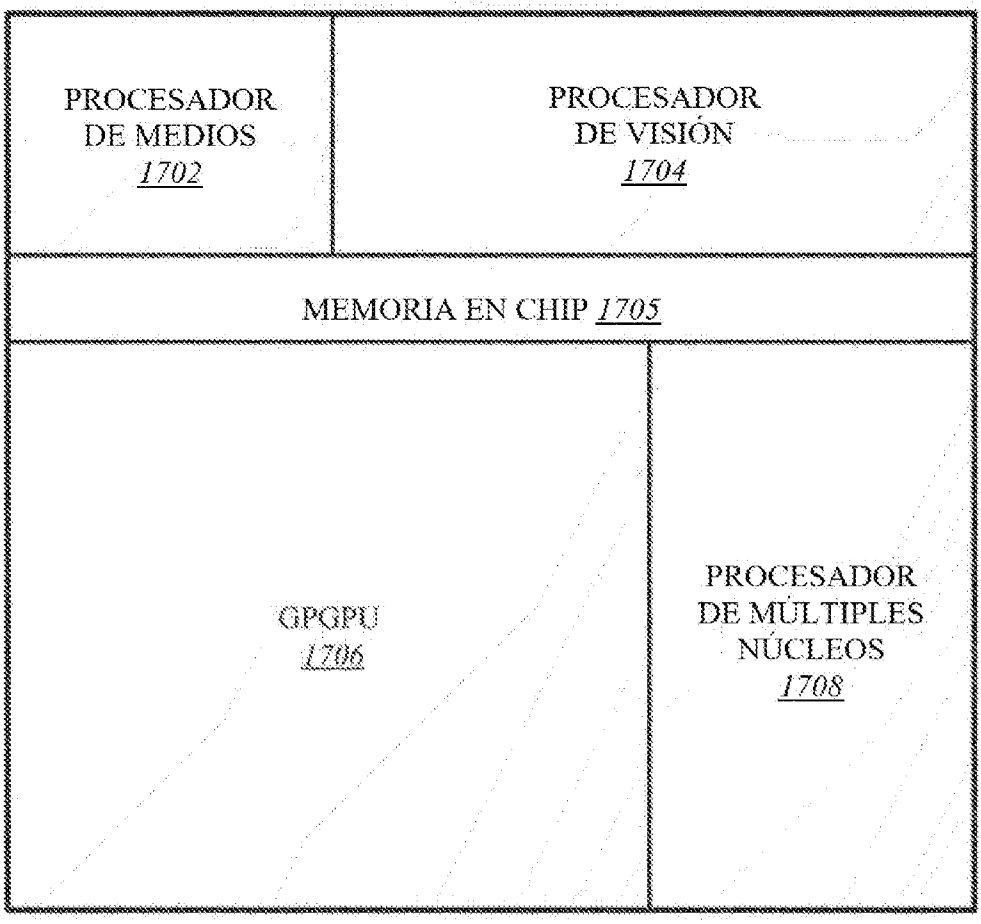

PROCESADOR DE MEDIOS 1702
PROCESADOR DE VISIÓN 1704
MEMORIA EN CHIP 1705
GPGPU 1706
PROCESADOR DE MÚLTIPLES NÚCLEOS 1708

FIG. 17

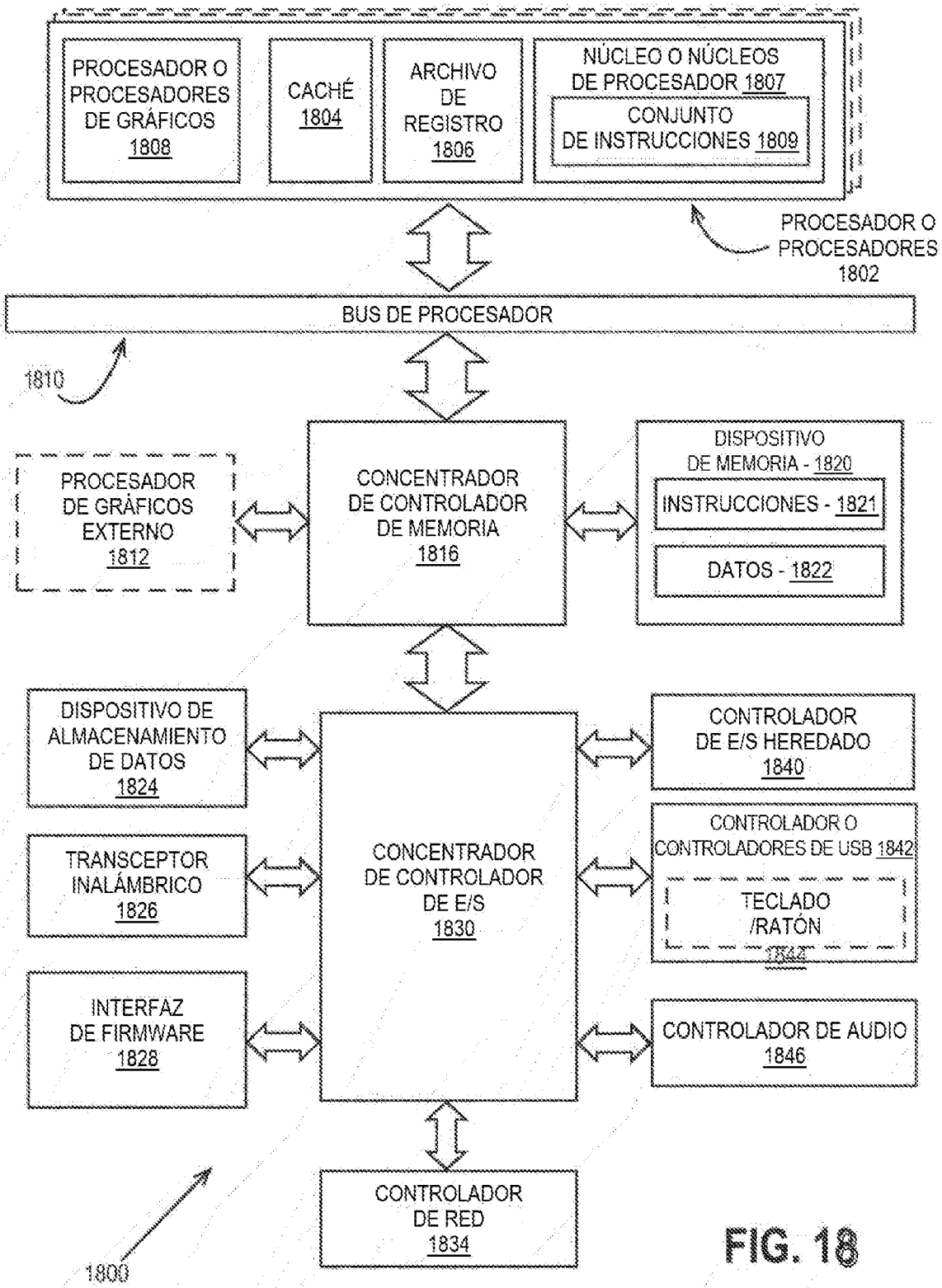

PROCESADOR O PROCESADORES DE GRÁFICOS 1808
CACHÉ 1804
ARCHIVO DE REGISTRO 1806
NÚCLEO O NÚCLEOS DE PROCESADOR 1807
CONJUNTO DE INSTRUCCIONES 1809
PROCESADOR O PROCESADORES 1802
BUS DE PROCESADOR
1810
PROCESADOR DE GRÁFICOS EXTERNO 1812
CONCENTRADOR DE CONTROLADOR DE MEMORIA 1816
DISPOSITIVO DE MEMORIA - 1820
INSTRUCCIONES - 1821
DATOS - 1822
DISPOSITIVO DE ALMACENAMIENTO DE DATOS 1824
TRANSCEPTOR INALÁMBRICO 1826
INTERFAZ DE FIRMWARE 1828
CONCENTRADOR DE CONTROLADOR DE E/S 1830
CONTROLADOR DE RED 1834
CONTROLADOR DE E/S HEREDADO 1840
CONTROLADOR O CONTROLADORES DE USB 1842
TECLADO /RATÓN
1844
CONTROLADOR DE AUDIO 1846
1800

FIG. 18

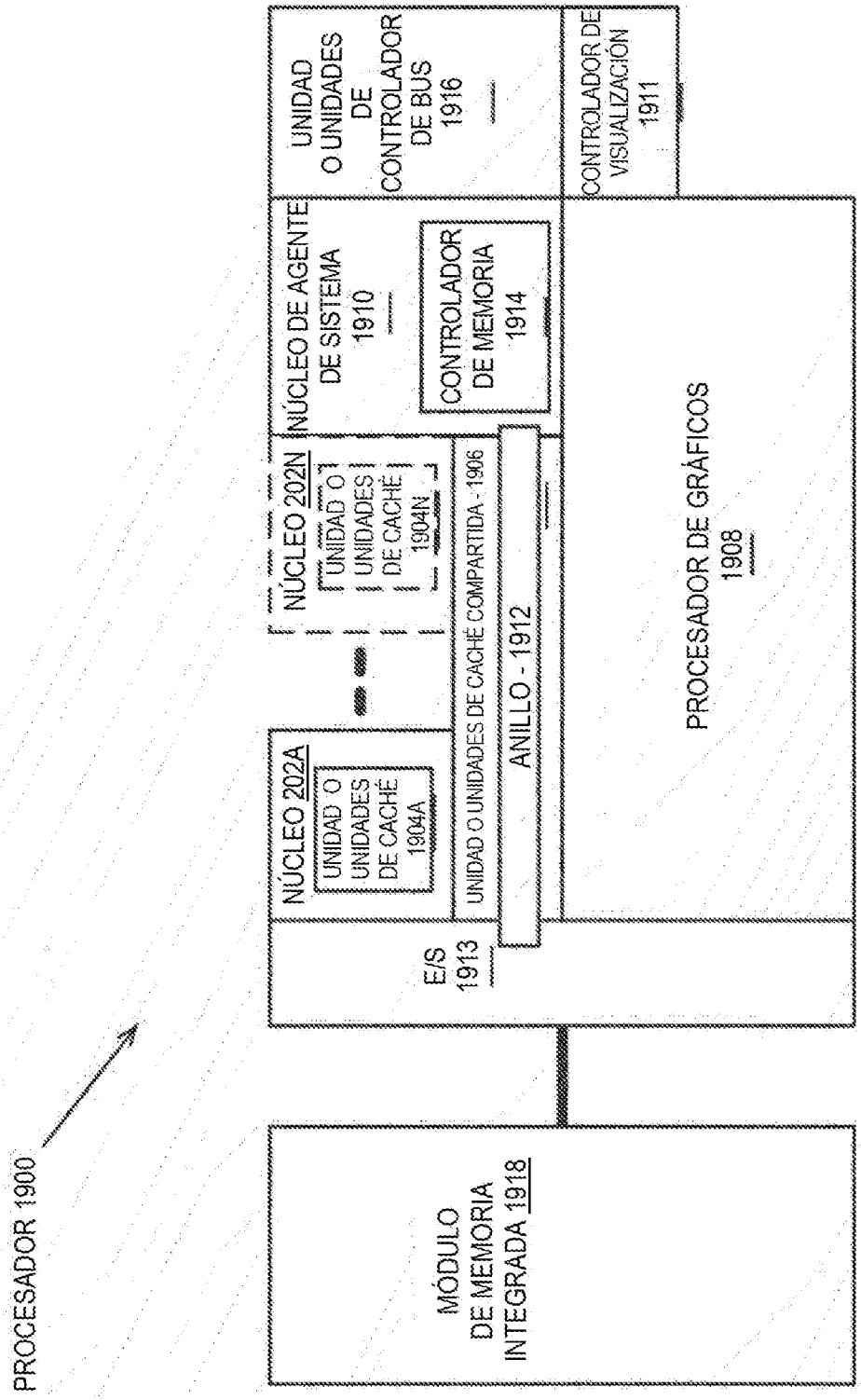

PROCESADOR 1900
MÓDULO DE MEMORIA INTEGRADA 1918
E/S 1913
NÚCLEO 202A
UNIDAD O UNIDADES DE CACHÉ 1904A
NÚCLEO 202N
UNIDAD O UNIDADES DE CACHÉ 1904N
UNIDAD O UNIDADES DE CACHÉ COMPARTIDA - 1906
ANILLO - 1912
NÚCLEO DE AGENTE DE SISTEMA 1910
CONTROLADOR DE MEMORIA 1914
UNIDAD O UNIDADES DE CONTROLADOR DE BUS 1916
CONTROLADOR DE VISUALIZACIÓN 1911
PROCESADOR DE GRÁFICOS 1908

FIG. 19

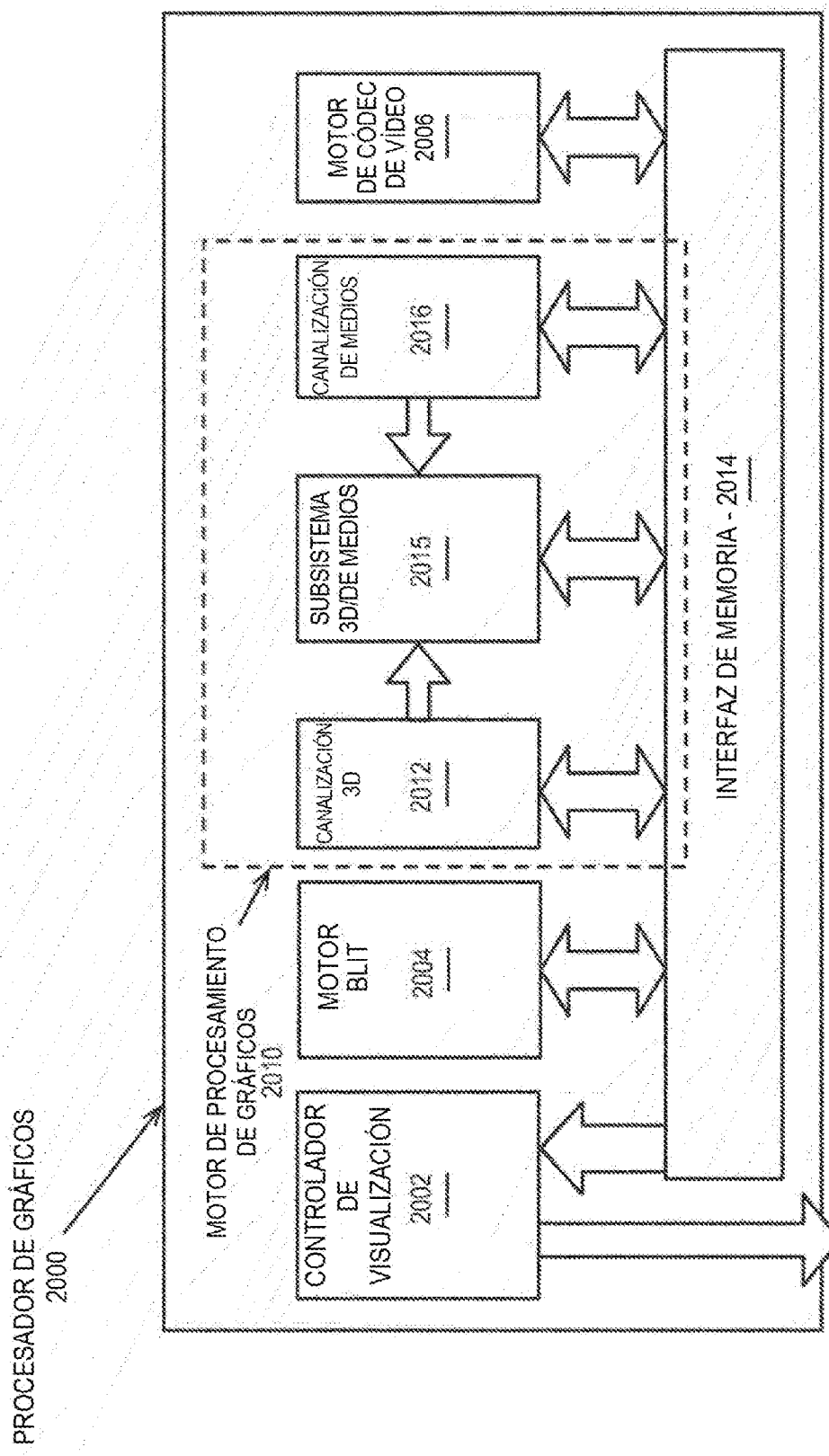

PROCESADOR DE GRÁFICOS
2000
MOTOR DE PROCESAMIENTO DE GRÁFICOS
2010
CONTROLADOR DE VISUALIZACIÓN
2002
MOTOR BLIT
2004
CANALIZACIÓN 3D
2012
SUBSISTEMA 3D/DE MEDIOS
2015
CANALIZACIÓN DE MEDIOS
2016
MOTOR DE CÓDEC DE VÍDEO
2006
INTERFAZ DE MEMORIA - 2014
DISPOSITIVO DE VISUALIZACIÓN
2020

FIG. 20

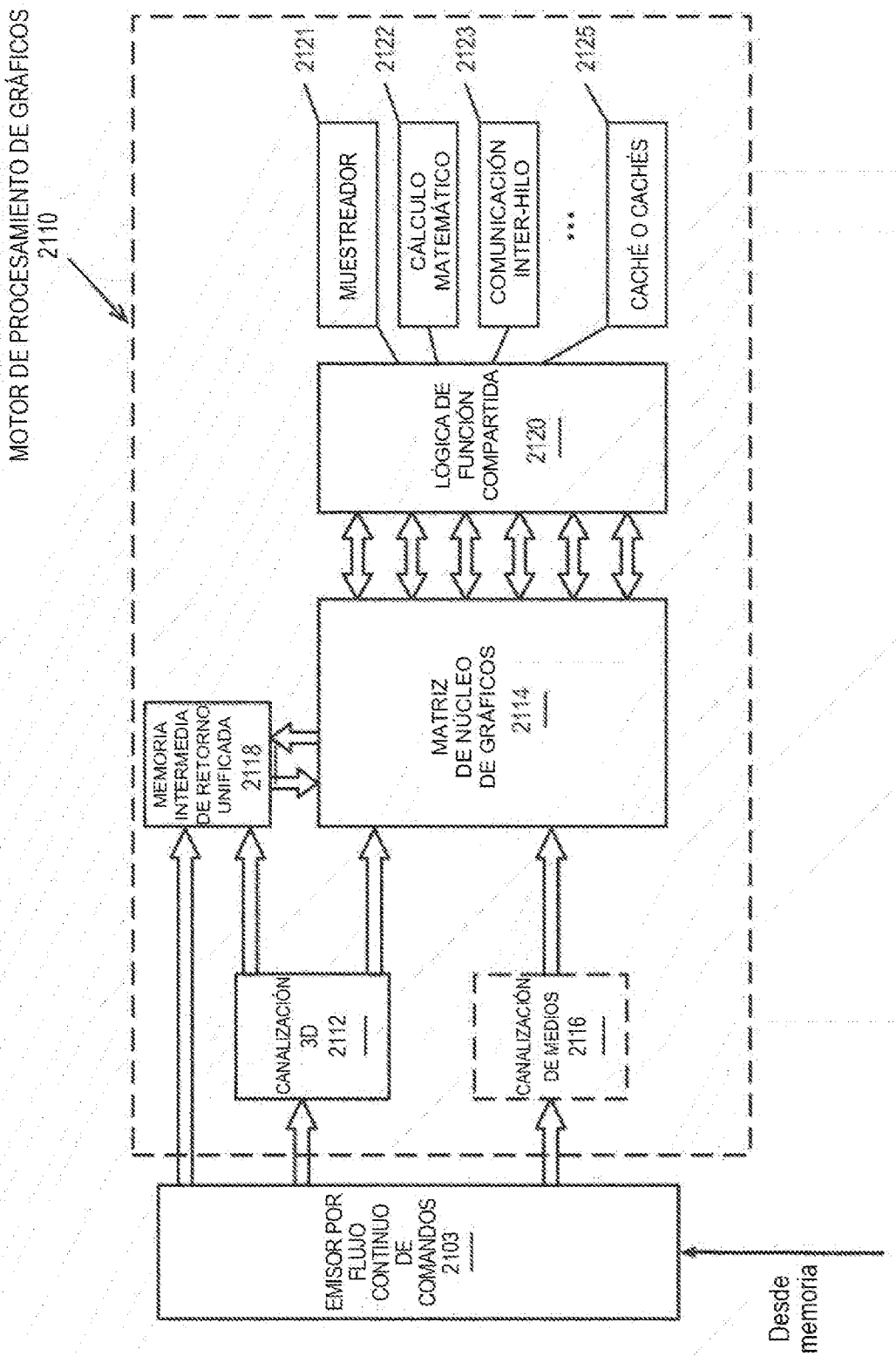

MOTOR DE PROCESAMIENTO DE GRÁFICOS
2110
MUESTREADOR
2121
CÁLCULO MATEMÁTICO
2122
COMUNICACIÓN INTER-HILO
2123
...
CACHÉ O CACHÉS
2125
LÓGICA DE FUNCIÓN COMPARTIDA
2120
MATRIZ DE NÚCLEO DE GRÁFICOS
2114
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
2118
CANALIZACIÓN 3D
2112
CANALIZACIÓN DE MEDIOS
2116
EMISOR POR FLUJO CONTINUO DE COMANDOS
2103
Desde memoria

FIG. 21

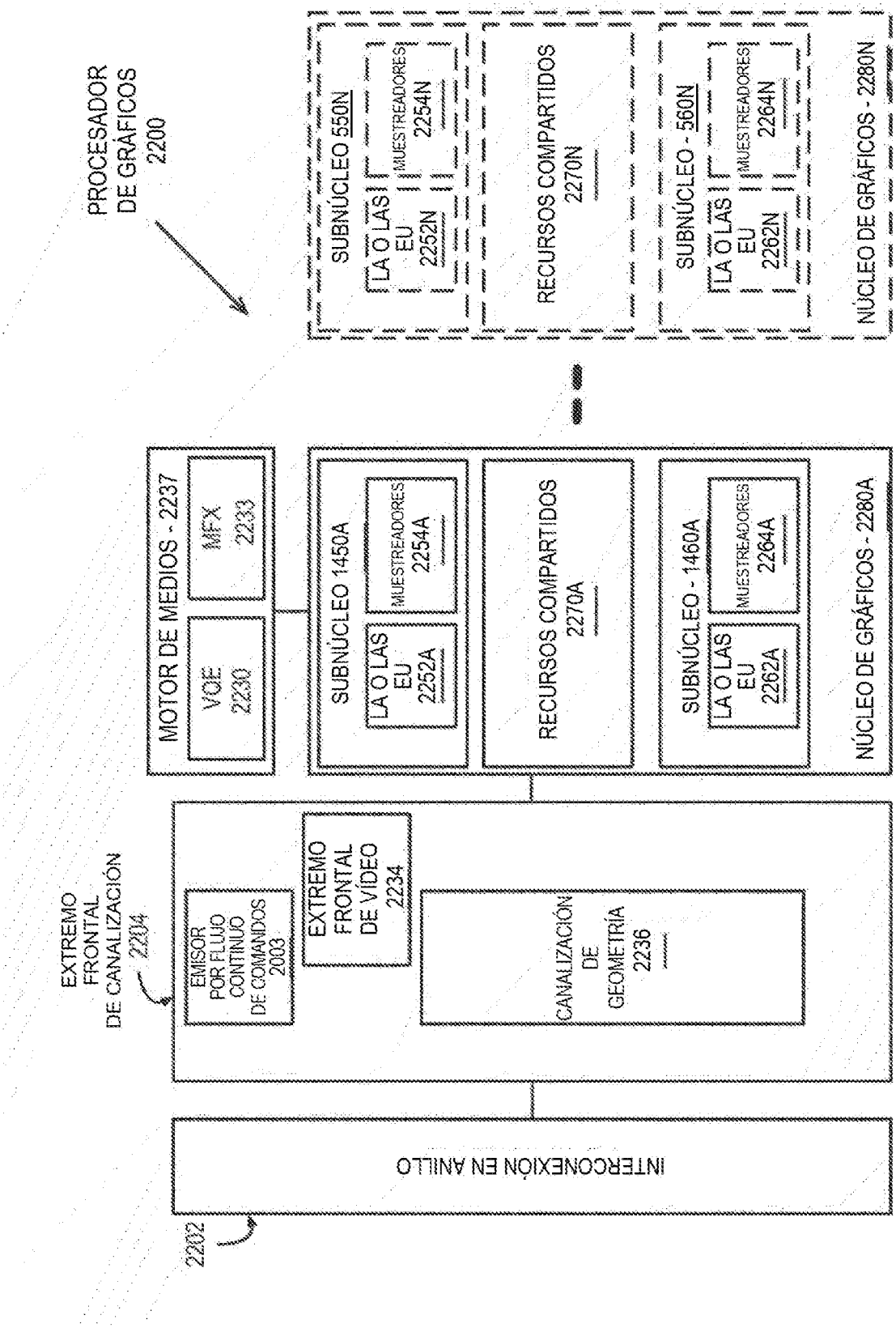

PROCESADOR DE GRÁFICOS 2200
INTERCONEXIÓN EN ANILLO
2202
EXTREMO FRONTAL DE CANALIZACIÓN 2204
EMISOR POR FLUJO CONTINUO DE COMANDOS 2003
EXTREMO FRONTAL DE VÍDEO 2234
CANALIZACIÓN DE GEOMETRÍA 2236
MOTOR DE MEDIOS - 2237
VQE 2230
MFX 2233
SUBNÚCLEO 1450A
LA O LAS EU 2252A
MUESTREADORES 2254A
RECURSOS COMPARTIDOS 2270A
SUBNÚCLEO - 1460A
LA O LAS EU 2262A
MUESTREADORES 2264A
NÚCLEO DE GRÁFICOS - 2280A
SUBNÚCLEO 550N
LA O LAS EU 2252N
MUESTREADORES 2254N
RECURSOS COMPARTIDOS 2270N
SUBNÚCLEO - 560N
LA O LAS EU 2262N
MUESTREADORES 2264N
NÚCLEO DE GRÁFICOS - 2280N

FIG. 22

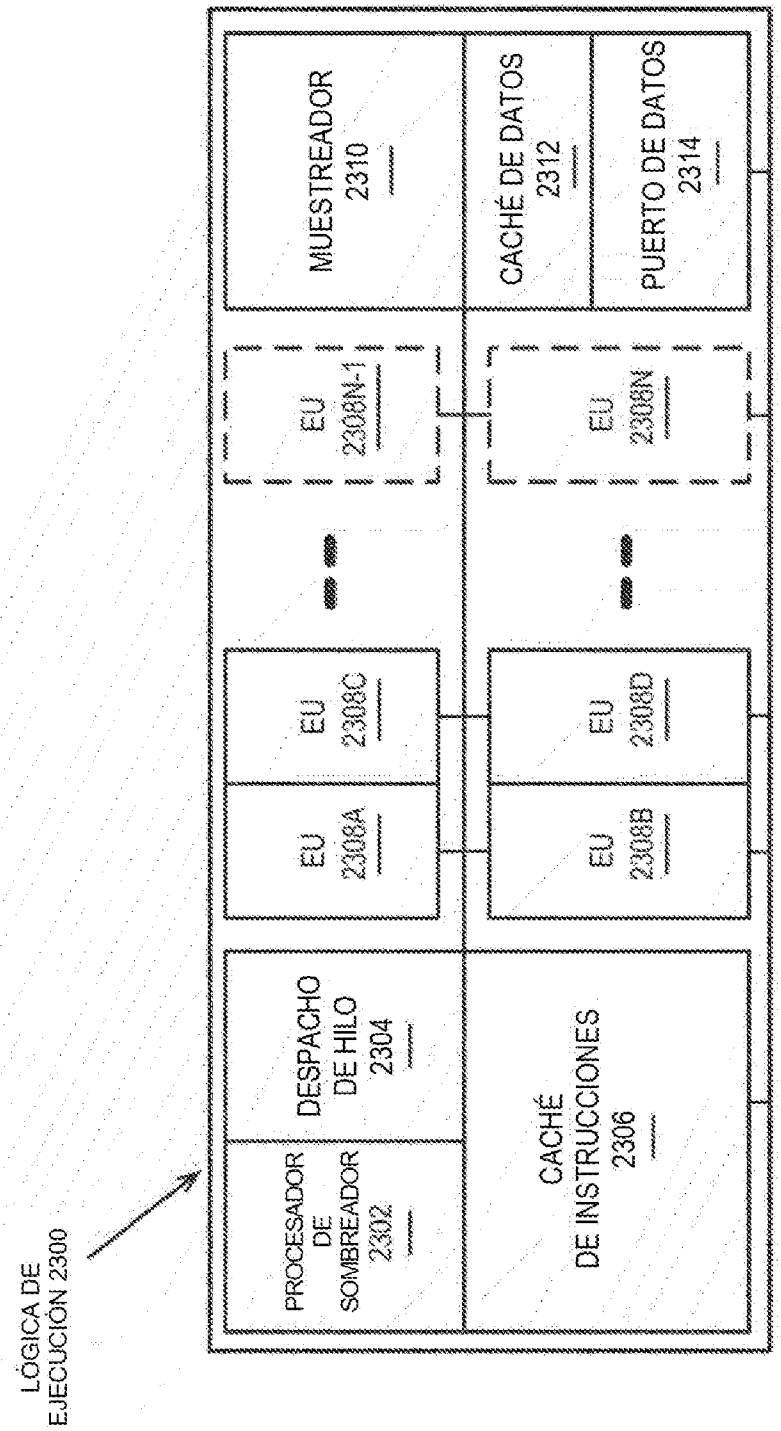

LÓGICA DE EJECUCIÓN 2300
PROCESADOR DE SOMBREADOR 2302
DESPACHO DE HILO 2304
CACHÉ DE INSTRUCCIONES 2306
EU 2308A
EU 2308B
EU 2308C
EU 2308D
EU 2308N-1
EU 2308N
MUESTREADOR 2310
CACHÉ DE DATOS 2312
PUERTO DE DATOS 2314

FIG. 23

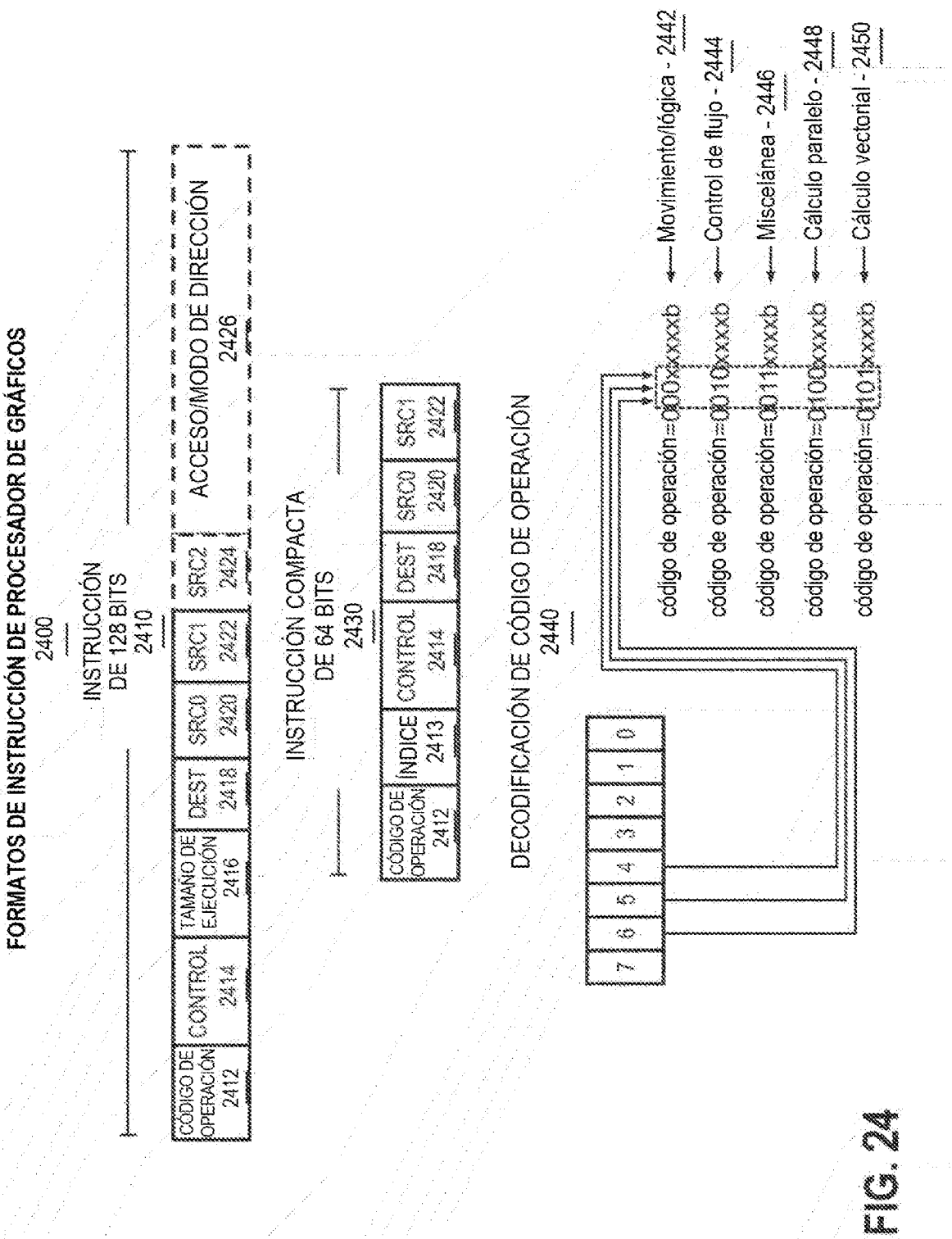

FORMATOS DE INSTRUCCIÓN DE PROCESADOR DE GRÁFICOS
2400
INSTRUCCIÓN DE 128 BITS
2410
CÓDIGO DE OPERACIÓN 2412
CONTROL 2414
TAMAÑO DE EJECUCIÓN 2416
DEST 2418
SRC0 2420
SRC1 2422
SRC2 2424
ACCESO/MODO DE DIRECCIÓN 2426
INSTRUCCIÓN COMPACTA DE 64 BITS
2430
CÓDIGO DE OPERACIÓN 2412
ÍNDICE 2413
CONTROL 2414
DEST 2418
SRC0 2420
SRC1 2422
DECODIFICACIÓN DE CÓDIGO DE OPERACIÓN
2440
7 6 5 4 3 2 1 0
código de operación=0x0xxxxb
código de operación=0x1xxxxb
código de operación=0x2xxxxb
código de operación=0x3xxxxb
código de operación=0x4xxxxb
código de operación=0x5xxxxb
Movimiento/lógica - 2442
Control de flujo - 2444
Misceláneа - 2446
Cálculo paralelo - 2448
Cálculo vectorial - 2450

FIG. 24

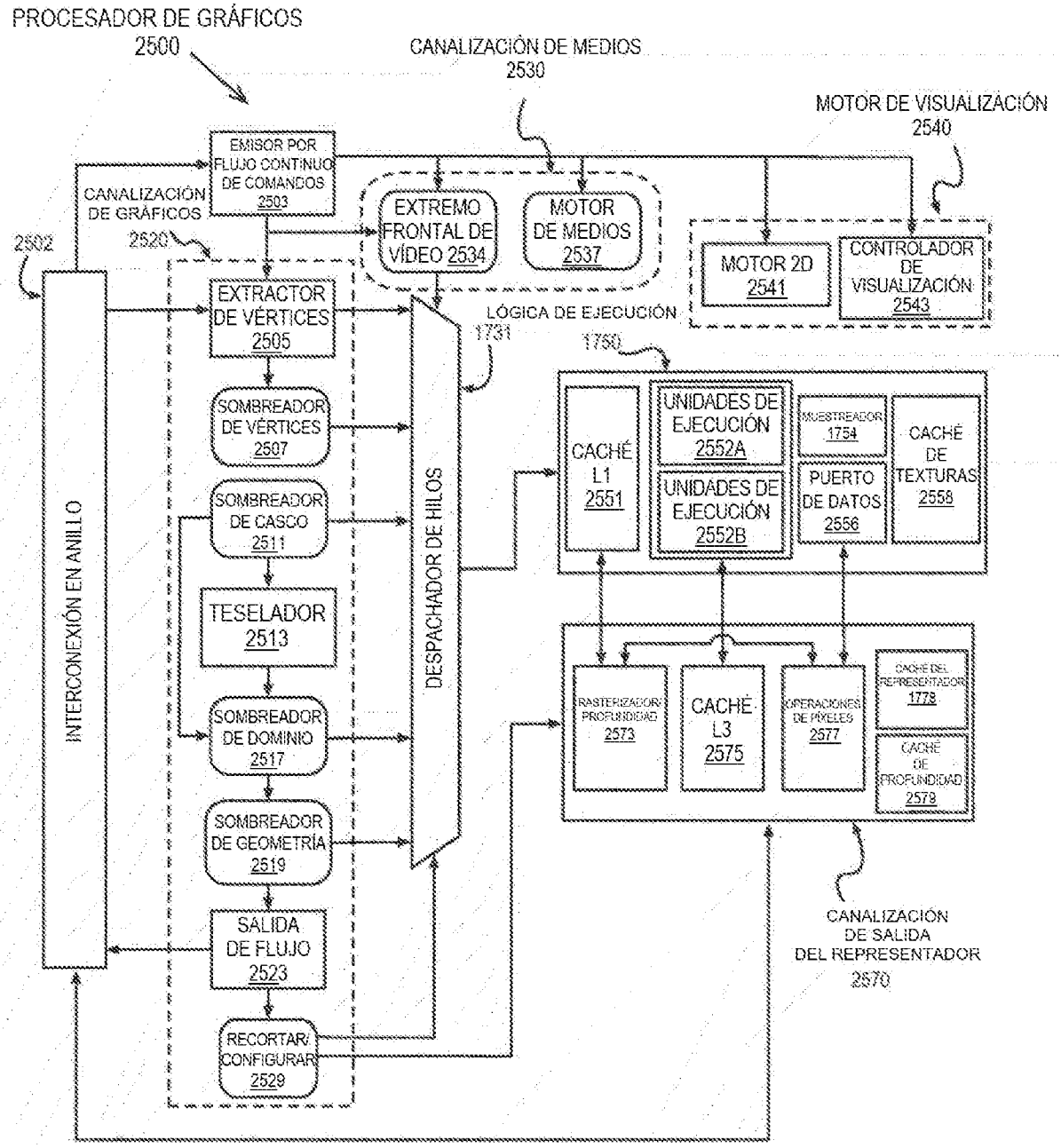

PROCESADOR DE GRÁFICOS
2500
CANALIZACIÓN DE MEDIOS
2530
MOTOR DE VISUALIZACIÓN
2540
EMISOR POR FLUJO CONTINUO DE COMANDOS
2503
CANALIZACIÓN DE GRÁFICOS
2520
2502
EXTREMO FRONTAL DE VÍDEO 2534
MOTOR DE MEDIOS 2537
MOTOR 2D 2541
CONTROLADOR DE VISUALIZACIÓN 2543
EXTRACTOR DE VÉRTICES 2505
LÓGICA DE EJECUCIÓN
1731
1750
SOMBREADOR DE VÉRTICES 2507
SOMBREADOR DE CASCO 2511
TESELADOR 2513
SOMBREADOR DE DOMINIO 2517
SOMBREADOR DE GEOMETRÍA 2519
SALIDA DE FLUJO 2523
RECORTAR/ CONFIGURAR 2529
INTERCONEXIÓN EN ANILLO
DESPACHADOR DE HILOS
CACHÉ L1 2551
UNIDADES DE EJECUCIÓN 2552A
UNIDADES DE EJECUCIÓN 2552B
MUESTREADOR 1754
PUERTO DE DATOS 2556
CACHÉ DE TEXTURAS 2558
RASTERIZADOR/ PROFUNDIDAD 2573
CACHÉ L3 2575
OPERACIONES DE PÍXELES 2577
CACHÉ DEL REPRESENTADOR 1778
CACHÉ DE PROFUNDIDAD 2579
CANALIZACIÓN DE SALIDA DEL REPRESENTADOR
2570

FIG. 25

FIG. 26A FORMATO DE COMANDO DE PROCESADOR DE GRÁFICOS
2600

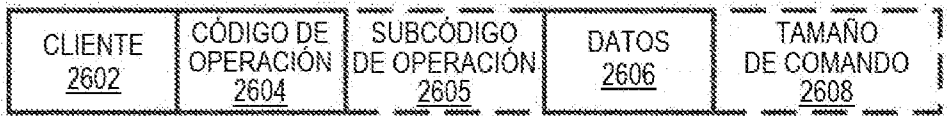
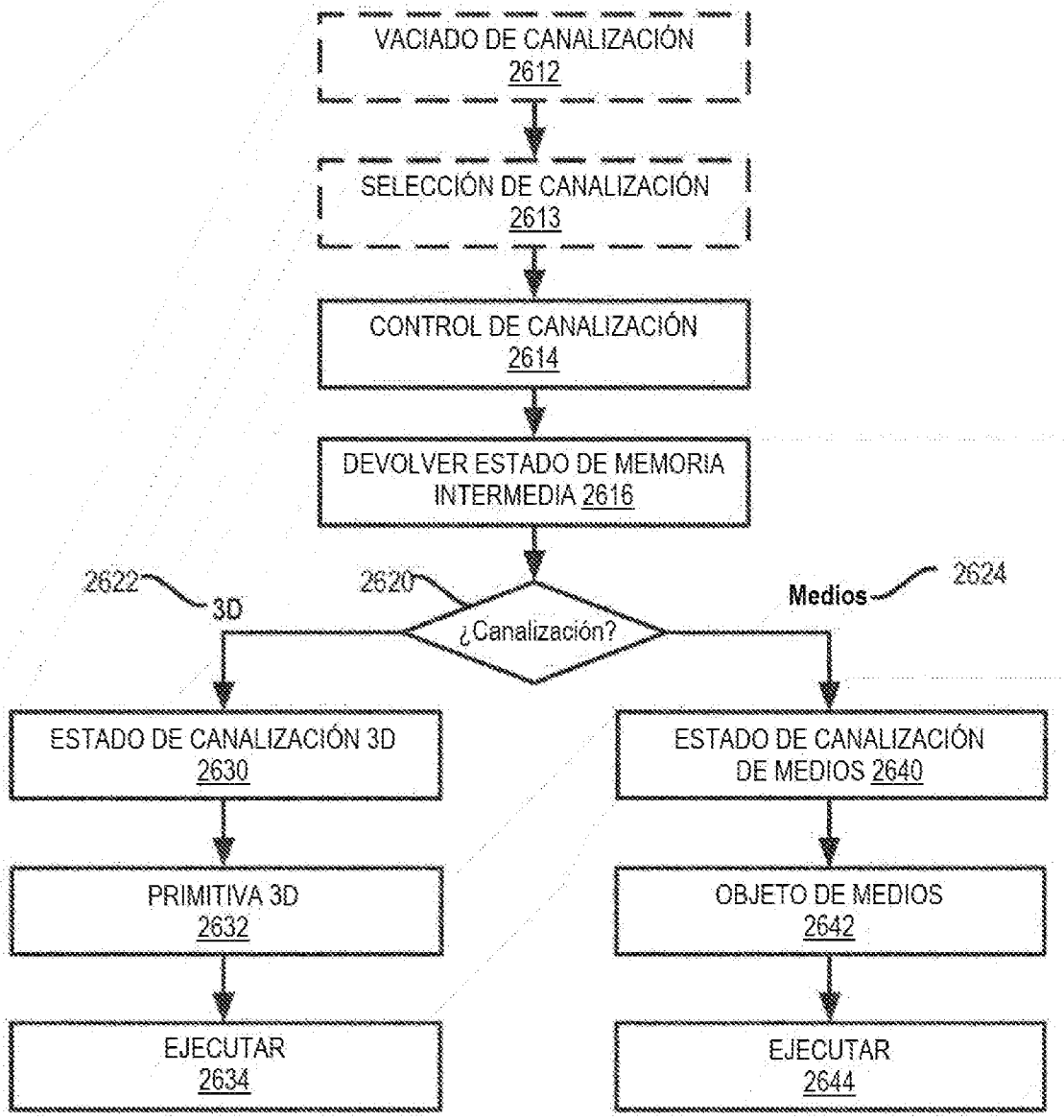


FIG. 26B SECUENCIA DE COMANDOS DE PROCESADOR DE GRÁFICOS
2610



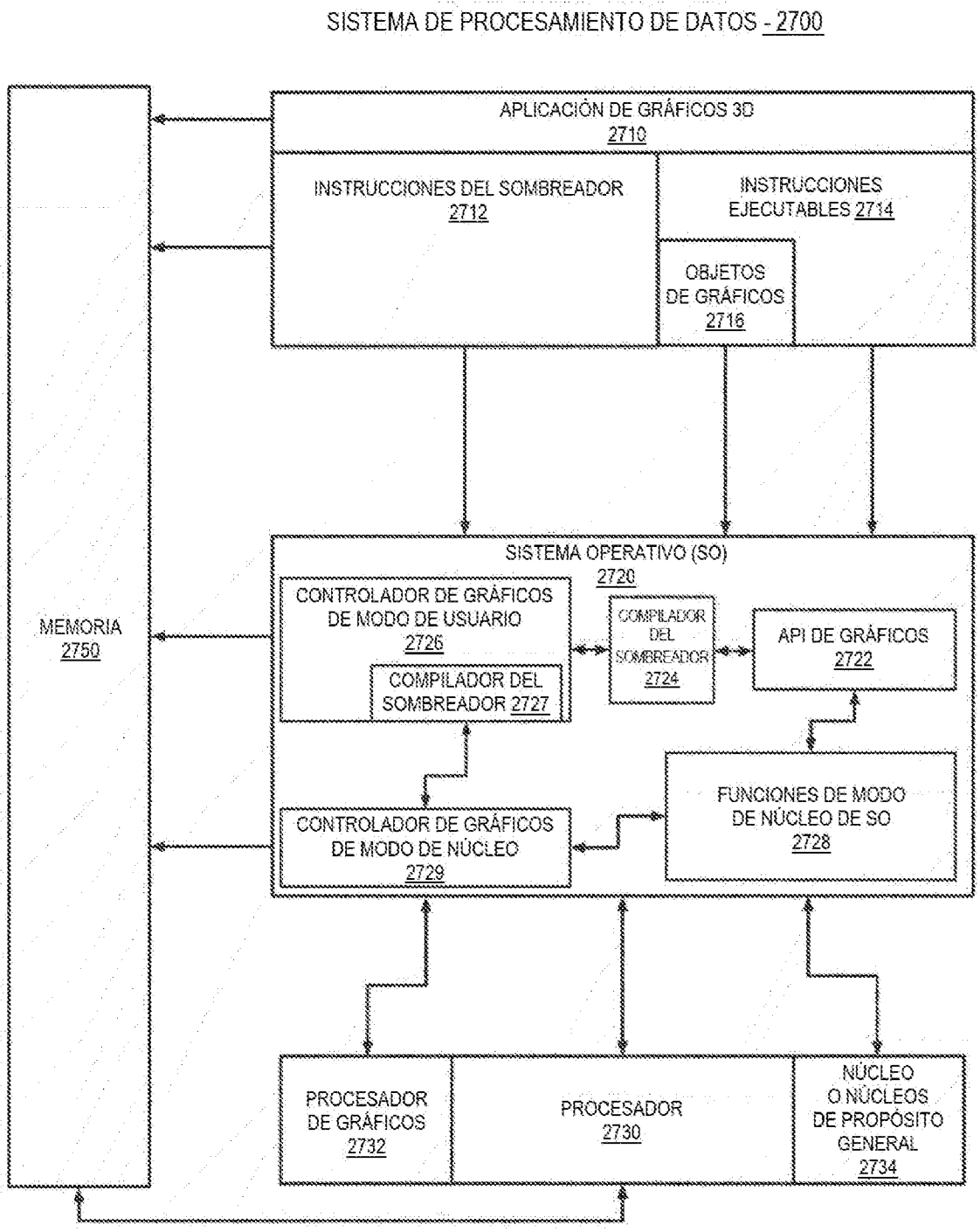

SISTEMA DE PROCESAMIENTO DE DATOS - 2700
APLICACIÓN DE GRÁFICOS 3D 2710
INSTRUCCIONES DEL SOMBREADOR 2712
INSTRUCCIONES EJECUTABLES 2714
OBJETOS DE GRÁFICOS 2716
MEMORIA 2750
SISTEMA OPERATIVO (SO) 2720
CONTROLADOR DE GRÁFICOS DE MODO DE USUARIO 2726
COMPILADOR DEL SOMBREADOR 2727
COMPILADOR DEL SOMBREADOR 2724
API DE GRÁFICOS 2722
FUNCIONES DE MODO DE NÚCLEO DE SO 2728
CONTROLADOR DE GRÁFICOS DE MODO DE NÚCLEO 2729
PROCESADOR DE GRÁFICOS 2732
PROCESADOR 2730
NÚCLEO O NÚCLEOS DE PROPÓSITO GENERAL 2734

FIG. 27

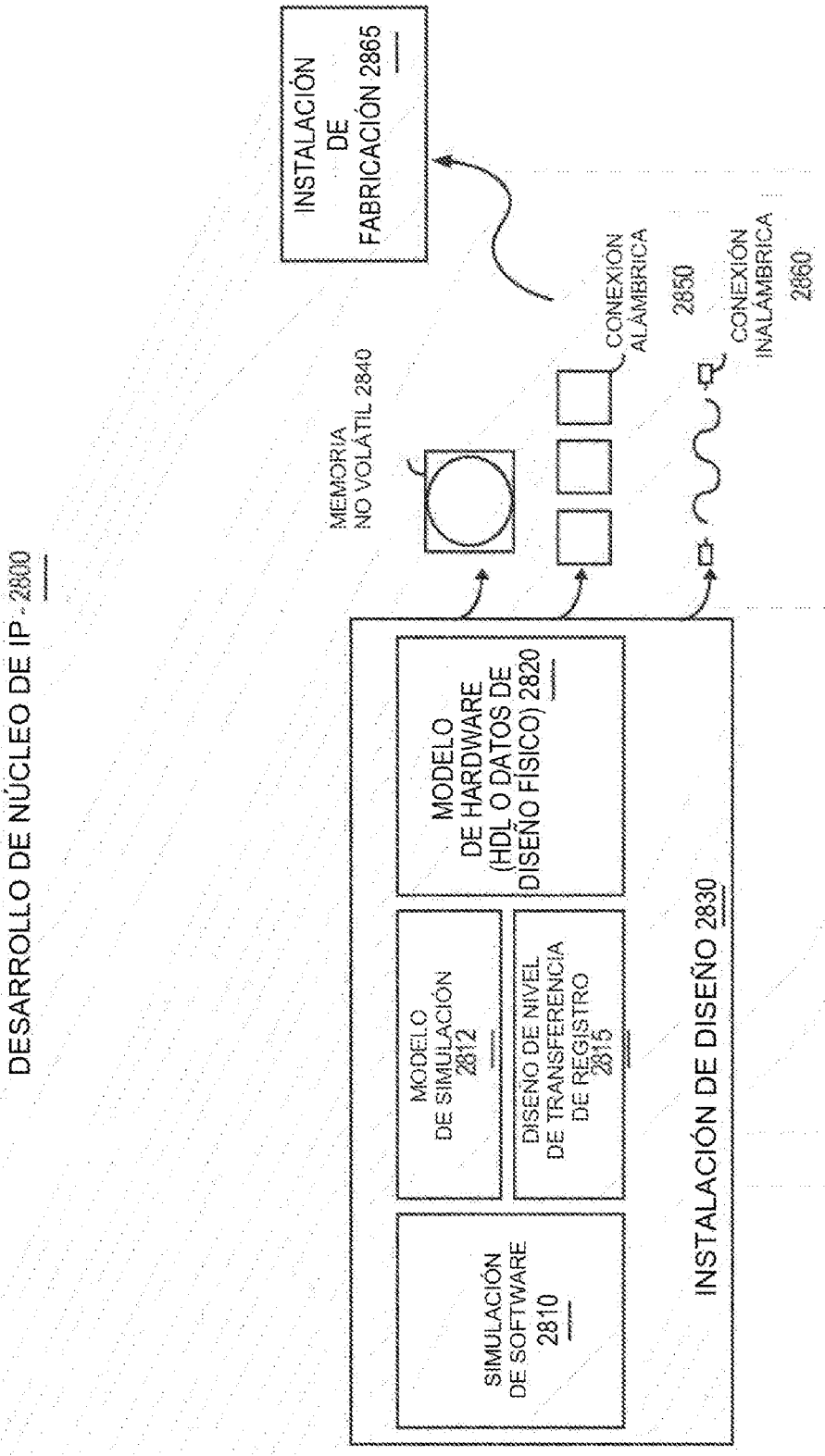

DESARROLLO DE NÚCLEO DE IP - 2800
INSTALACIÓN DE FABRICACIÓN 2865
MEMORIA NO VOLÁTIL 2840
CONEXIÓN ALÁMBRICA 2850
CONEXIÓN INALÁMBRICA 2860
SIMULACIÓN DE SOFTWARE 2810
MODELO DE SIMULACIÓN 2812
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 2815
MODELO DE HARDWARE (HDL O DATOS DE DISEÑO FÍSICO) 2820
INSTALACIÓN DE DISEÑO 2830

FIG. 28

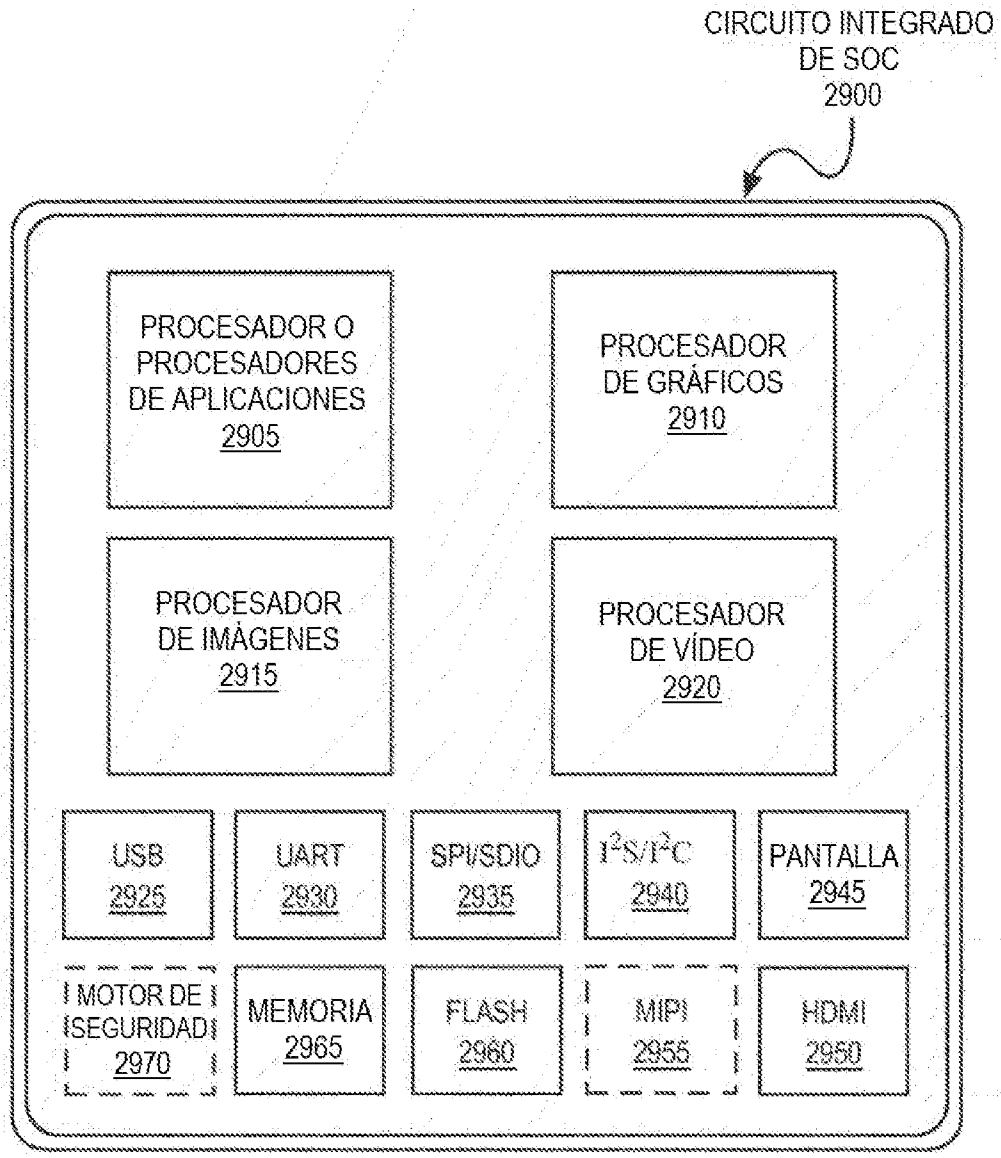

CIRCUITO INTEGRADO DE SOC 2900
PROCESADOR O PROCESADORES DE APLICACIONES 2905
PROCESADOR DE GRÁFICOS 2910
PROCESADOR DE IMÀGENES 2915
PROCESADOR DE VÍDEO 2920
USB 2925
UART 2930
SPI/SDIO 2935
I2S/I2C 2940
PANTALLA 2945
MOTOR DE SEGURIDAD 2970
MEMORIA 2965
FLASH 2960
MIPI 2955
HDMI 2950

FIG. 29

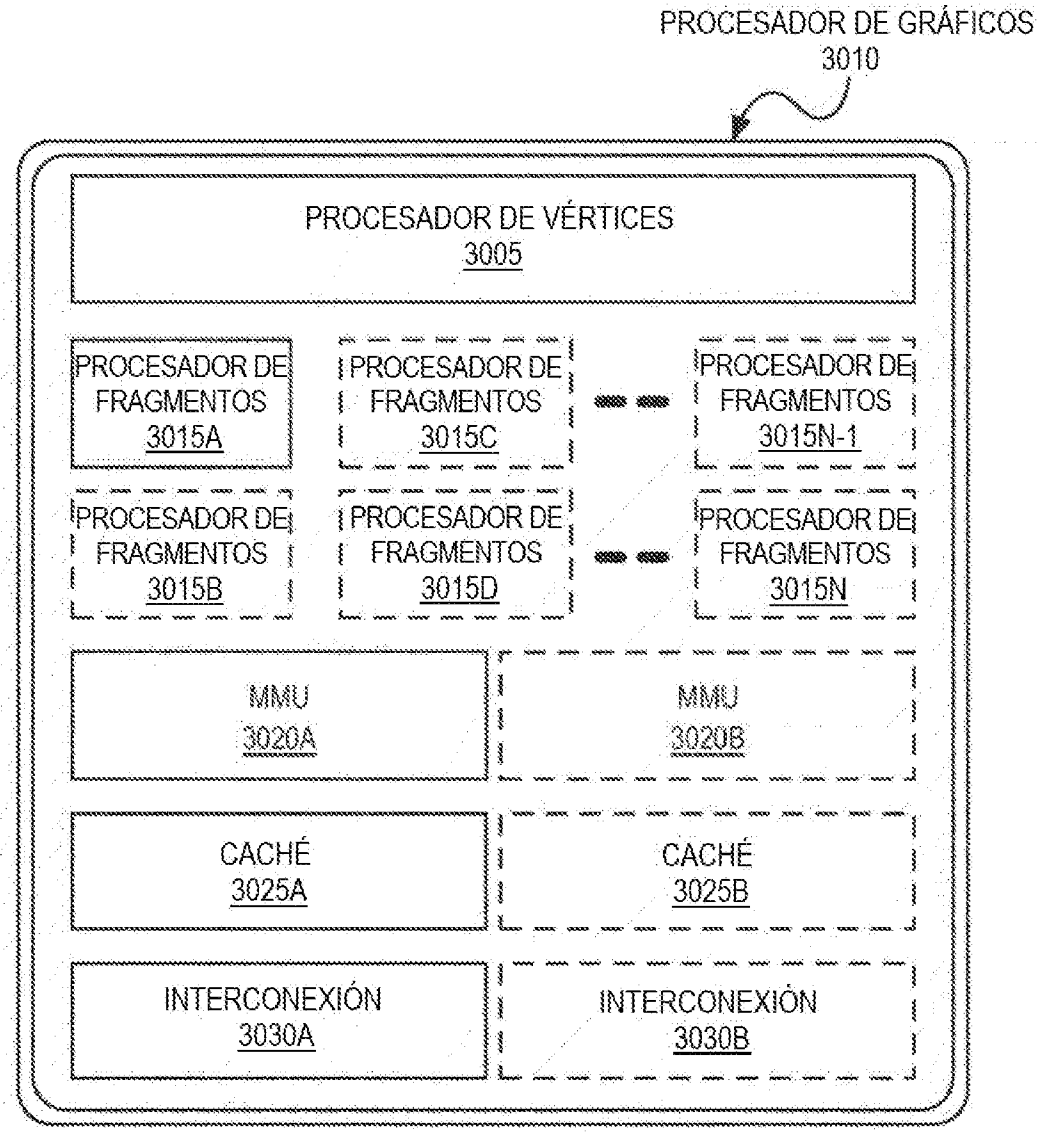

PROCESADOR DE GRÁFICOS
3010
PROCESADOR DE VÉRTICES
3005
PROCESADOR DE FRAGMENTOS
3015A
PROCESADOR DE FRAGMENTOS
3015C
PROCESADOR DE FRAGMENTOS
3015N-1
PROCESADOR DE FRAGMENTOS
3015B
PROCESADOR DE FRAGMENTOS
3015D
PROCESADOR DE FRAGMENTOS
3015N
MMU
3020A
MMU
3020B
CACHÉ
3025A
CACHÉ
3025B
INTERCONEXIÓN
3030A
INTERCONEXIÓN
3030B

FIG. 30

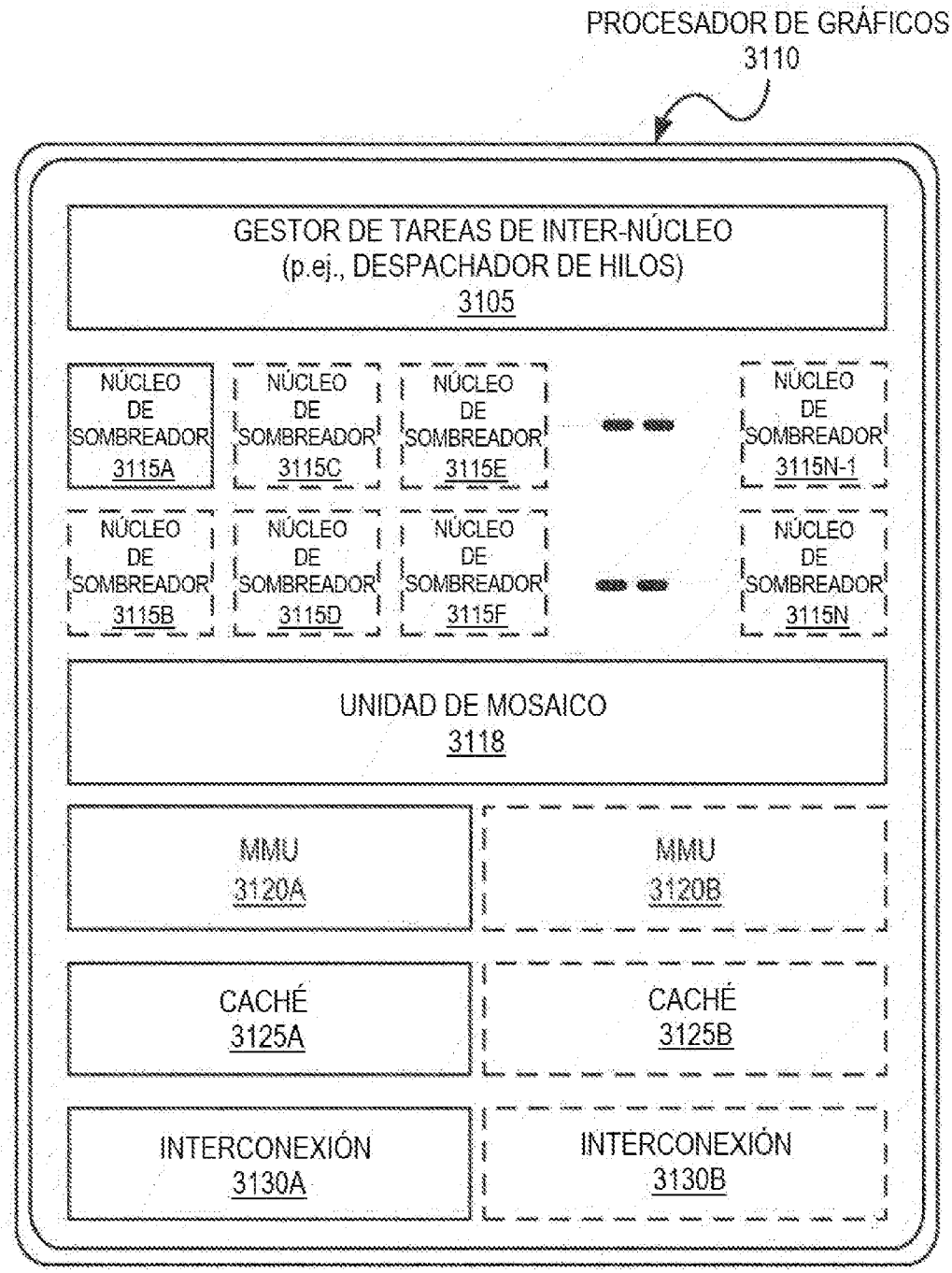

PROCESADOR DE GRÁFICOS
3110
GESTOR DE TAREAS DE INTER-NÚCLEO
(p.ej., DESPACHADOR DE HILOS)
3105
NÚCLEO DE SOMBREADOR 3115A
NÚCLEO DE SOMBREADOR 3115C
NÚCLEO DE SOMBREADOR 3115E
NÚCLEO DE SOMBREADOR 3115N-1
NÚCLEO DE SOMBREADOR 3115B
NÚCLEO DE SOMBREADOR 3115D
NÚCLEO DE SOMBREADOR 3115F
NÚCLEO DE SOMBREADOR 3115N
UNIDAD DE MOSAICO
3118
MMU
3120A
MMU
3120B
CACHÉ
3125A
CACHÉ
3125B
INTERCONEXIÓN
3130A
INTERCONEXIÓN
3130B

FIG. 31