(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 926 704**

(51) Int. CI.:

G06N 3/04 (2006.01)
G06N 3/063 (2006.01)
G06N 3/08 (2006.01)
G06T 1/20 (2006.01)
G06F 9/50 (2006.01)
G06T 15/00 (2011.01)
G06F 9/30 (2008.01)
G06F 9/38 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **26.03.2018 E 18164092 (1)**

(97) Fecha y número de publicación de la concesión europea: **27.07.2022 EP 3396547**

(54) Título: **Optimización de cálculos para operaciones de aprendizaje automático de baja precisión**

(30) Prioridad:

28.04.2017 US 201715581167

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
27.10.2022

(73) Titular/es:

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

(72) Inventor/es:

OULD-AHMED-VALL, ELMOUSTAPHA;
BAGHSORKHI, SARA S.;
YAO, ANBANG;
NEALIS, KEVIN;
CHEN, XIAOMING;
KOKER, ALTUG;
APPU, ABHISHEK R.;
WEAST, JOHN C.;
MACPHERSON, MIKE B.;
KIM, DUKHWAN;
HURD, LINDA L.;
ASHBAUGH, BEN J.;
LAKSHMANAN, BARATH;
MA, LIWEI;
RAY, JOYDEEP;
TANG, PING T. y
STRICKLAND, MICHAEL S.

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Optimización de cálculos para operaciones de aprendizaje automático de baja precisión

Campo

Las realizaciones se refieren, en general, a procesamiento de datos y, en particular, a procesamiento de datos a través de una unidad de procesamiento de gráficos para fines generales.

Antecedente de la descripción

El procesamiento de datos gráficos paralelo actual incluye sistemas y métodos desarrollados para realizar operaciones específicas en datos gráficos, tal y como, por ejemplo, interpolación lineal, teselación, rasterización, mapeo de textura, prueba de profundidad, etc. Tradicionalmente, los procesadores gráficos utilizan unidades computacionales de función fija para procesar datos gráficos; sin embargo, más recientemente, porciones de procesadores gráficos se han vuelto programables, lo que permite que dichos procesadores admitan una mayor variedad de operaciones para procesar datos de vértices y de fragmentos.

Para aumentar aún más el rendimiento, los procesadores gráficos generalmente implementan técnicas de procesamiento, tal y como canalizaciones que intentan procesar, en paralelo, tantos datos gráficos como sea posible a lo largo de las diferentes partes de la canalización de gráficos. Los procesadores gráficos paralelos con arquitecturas de instrucción única para múltiples hilos (SIMT) están diseñados para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura SIMT, los grupos de hilos paralelos intentan ejecutar juntos instrucciones de programa de manera sincrónica tan a menudo como sea posible para aumentar la eficiencia de procesamiento. Se puede encontrar una visión general de software y hardware para arquitecturas SIMT 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 documento US 2005/066205 A1 se refiere a una arquitectura de gráficos 3D de alta calidad y rendimiento adecuada para dispositivos portátiles.

El documento de Venkataramani Swagath et al. los "Procesadores vectoriales programables de calidad para computación aproximada" se refiere a procesadores programables de calidad en los que la noción de calidad está explícitamente codificada en el conjunto de instrucciones.

Compendio

La presente invención está definida mediante la reivindicación independiente 1. Las reivindicaciones dependientes definen realizaciones de la misma. Cualquier "realización" o "ejemplo" que se de a conocer en la siguiente descripción, pero que no esté cubierto por las reivindicaciones debe considerarse como presentado únicamente con fines ilustrativos.

Breve descripción de los dibujos

Para que las características de la presente invención se puedan comprender en detalle, se puede obtener una descripción más específica de la invención con referencia a las realizaciones, algunas de las cuales se ilustran en los dibujos adjuntos. Sin embargo, se ha de observar que los dibujos adjuntos ilustran solo las realizaciones típicas y, por tanto, no se han de considerar limitantes del alcance de todas las realizaciones.

La Figura 1 es un diagrama de bloque que ilustra un sistema informático configurado para implementar uno o más aspectos de los ejemplos descritos en la presente memoria;

la Figura 2A-2D ilustra componentes de procesador paralelo, según un ejemplo;

las Figuras 3A-3B son diagramas de bloque de multiprocesadores gráficos, según los ejemplos;

las Figuras 4A-4F ilustran una arquitectura de ejemplo en la que una pluralidad de GPU se acoplan en comunicación con una pluralidad de procesadores multinúcleo;

la Figura 5 ilustra una canalización de procesamiento de gráficos, según un ejemplo;

la Figura 6 ilustra una pila de software de aprendizaje automático, según un ejemplo;

la Figura 7 ilustra una unidad de procesamiento de gráficos para fines generales muy paralelos, según un ejemplo;

la Figura 8 ilustra un sistema informático multi-GPU, según un ejemplo;

las Figuras 9A-9B ilustran capas de redes neuronales profundas de ejemplo;

la Figura 10 ilustra una red neuronal recurrente de ejemplo;

la Figura 11 ilustra el entrenamiento y despliegue de una red neuronal profunda;

la Figura 12 es un diagrama de bloque que ilustra un aprendizaje distribuido;

la Figura 13 ilustra un sistema de inferencia de ejemplo en un chip (SOC) apropiado para llevar a cabo inferencias utilizando un modelo entrenado;

la Figura 14 ilustra componentes de una unidad de coma flotante de precisión dinámica, según la invención;

la Figura 15 proporciona detalles adicionales respecto de una unidad de coma flotante de precisión dinámica, según una realización;

la Figura 16 ilustra asignaciones de hilos para un sistema de procesamiento de precisión dinámica, según una realización;

la Figura 17 ilustra una lógica para llevar a cabo una operación numérica con una precisión menor a la solicitada, según una realización;

la Figura 18 ilustra una vectorización en bucle para unidades SIMD, según un ejemplo;

la Figura 19 ilustra un sistema de procesamiento de hilos, según un ejemplo;

la Figura 20 ilustra una lógica para asignar hilos para cálculos, según un ejemplo;

la Figura 21 ilustra una red neuronal profunda que se puede procesar utilizando una lógica de cálculo provista por los ejemplos descritos en la presente memoria;

la Figura 22 es un diagrama de flujo de lógica para evitar errores o pérdidas de precisión significativas cuando se llevan a cabo operaciones de precisión baja en aprendizaje automático, según una realización;

la Figura 23 es un diagrama de bloque de un sistema de procesamiento, según un ejemplo;

la Figura 24 es un diagrama de bloque de un ejemplo de un procesador con uno o más núcleos de procesador, un controlador de memoria integrado y un procesador gráfico integrado;

la Figura 25 es un diagrama de bloque de un procesador gráfico, que puede ser una unidad de procesamiento de gráficos discreta o puede ser un procesador gráfico integrado con una pluralidad de núcleos de procesamiento;

la Figura 26 es un diagrama de bloque de un motor de procesamiento de gráficos de un procesador gráfico según algunos ejemplos;

la Figura 27 es un diagrama de bloque de un procesador gráfico proporcionado por un ejemplo adicional;

la Figura 28 ilustra una lógica de ejecución de hilo que incluye una disposición de elementos de procesamiento empleada en algunos ejemplos;

la Figura 29 es un diagrama de bloque que ilustra formatos de instrucción del procesador gráfico según algunos ejemplos;

la Figura 30 es un diagrama de bloque de un procesador gráfico según otro ejemplo;

las Figuras 31A-31B ilustran un formato de comando y secuencia de comando del procesador gráfico, según algunos ejemplos;

la Figura 32 ilustra una arquitectura de software de gráficos de ejemplo para un sistema de procesamiento de datos según algunos ejemplos;

la Figura 33 es un diagrama de bloque que ilustra un sistema de desarrollo de núcleo PI, según un ejemplo;

la Figura 34 es un diagrama de bloque que ilustra un sistema de ejemplo en circuito de chip integrado, según un ejemplo;

la Figura 35 es un diagrama de bloque que ilustra un procesador gráfico adicional, según un ejemplo; y

la Figura 36 es un diagrama de bloque que ilustra un procesador gráfico de ejemplo adicional de un sistema en un circuito de chip integrado, según un ejemplo.

Descripción detallada

En algunos ejemplos, una unidad de procesamiento de gráficos (GPU) está acoplada en comunicación con núcleos anfitriones/procesadores para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de

análisis de patrones y diversas funciones de GPU para fines generales (GPGPU). La GPU puede estar acoplada en comunicación con el procesador anfitrión/núcleos a través de un bus u otra interconexión (p. ej., interconexión de alta velocidad, como PCIe o NVLink). En otros ejemplos, la GPU puede estar integrada en el mismo paquete o chip que los núcleos y estar acoplada en comunicación con los núcleos a través de un bus/interconexión de procesador interno (es decir, interno para el paquete o chip). Independientemente de la manera en la que la GPU está conectada, los núcleos de procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidos en un descriptor de trabajo. La GPU entonces utiliza un circuito/lógica dedicado para procesar de manera eficiente estos comandos/instrucciones.

En la siguiente descripción, se describen numerosos detalles específicos para proporcionar una comprensión más exhaustiva. Sin embargo, para un experto en la técnica será evidente que las realizaciones descritas en la presente memoria se pueden poner en práctica sin uno o más de estos detalles específicos. En otras instancias, no se describirán características bien conocidas para evitar dificultar los detalles de las realizaciones presentes.

Visión general del sistema

La Figura 1 es un diagrama de bloque que ilustra un sistema 100 informático configurado para implementar uno o más aspectos de los ejemplos descritos en la presente memoria. El sistema 100 informático incluye el subsistema 101 de procesamiento que presenta uno o más procesador(es) 102 y una memoria 104 de sistema que se comunica mediante un trayecto de interconexión que puede incluir un centro 105 de memoria. El centro 105 de memoria puede ser un componente separado dentro de un componente del conjunto de chips o puede estar integrado dentro del uno o más procesador(es) 102. El centro 105 de memoria se acopla con un subsistema 111 de E/S mediante un enlace 106 de comunicación. El subsistema 111 de E/S incluye un centro 107 de E/S que puede permitir al sistema 100 informático recibir entradas de uno o más dispositivo(s) 108 de entrada. Además, el centro 107 de E/S puede permitir que un controlador de visualización, que puede estar incluido en el uno o más procesador(es) 102, proporcione salidas a uno o más dispositivos 110A de visualización. En un ejemplo, el uno o más dispositivo(s) 110A de visualización acoplado con el centro 107 de E/S puede incluir un dispositivo de visualización local, interno o incorporado.

En un ejemplo, el subsistema 101 de procesamiento incluye uno o más procesador(es) 112 paralelo(s) acoplado(s) a un centro 105 de memoria mediante un bus u otro enlace 113 de comunicación. El enlace 113 de comunicación puede ser una de cualquier cantidad de estándares basados en tecnologías o protocolos de enlace de comunicación, tal y como, pero no limitado a, PCI Express, o puede ser una interfaz de comunicación o tejido de comunicación específico del proveedor. En un ejemplo, el uno o más procesador(es) 112 paralelos forma(n) un sistema de procesamiento paralelo o de vector focalizado computacionalmente que incluye una gran cantidad de núcleos de procesamiento y/o clústeres de procesamiento, tal y como un procesador MIC (muchos núcleos integrados). En un ejemplo, el uno o más procesador(es) 112 paralelo(s) forma un subsistema de procesamiento de gráficos que puede emitir píxeles a uno de los uno o más dispositivo(s) 110A de visualización acoplado(s) mediante el centro 107 de E/S. El uno o más procesador(es) 112 paralelo(s) también puede incluir un controlador de visualización y una interfaz de visualización (no se muestra) para permitir una conexión directa con uno o más dispositivo(s) 110B de visualización.

Dentro del subsistema 111 de E/S, se puede conectar una unidad 114 de almacenamiento de sistema con el centro 107 de E/S para proporcionar un mecanismo de almacenamiento para el sistema 100 informático. Se puede utilizar un conmutador 116 de E/S para proporcionar un mecanismo de interfaz para permitir conexiones entre el centro 107 de E/S y otros componentes, tal y como un adaptador 118 de red y/o un adaptador 119 de red inalámbrica que puede estar integrado en la plataforma, y otros diversos dispositivos que se pueden añadir mediante uno o más dispositivo(s) 120 de complemento. El adaptador 118 de red puede ser un adaptador Ethernet u otro adaptador de red cableado. El adaptador 119 de red inalámbrica puede incluir uno o más de Wi-Fi, Bluetooth, comunicación de campo cercano (NFC) u otro dispositivo de red que incluya una o más radios inalámbricas.

El sistema 100 informático puede incluir otros componentes que no se muestran explícitamente, que incluyen USB u otras conexiones de puerto, unidades de almacenamiento ópticas, dispositivos de captura de vídeo y componentes similares, que también pueden estar conectados al centro 107 de E/S. Los trayectos de comunicación que interconectan los diversos componentes de la Figura 1 pueden implementarse utilizando cualquier protocolo apropiado, tal y como protocolos basados en PCI (interconexión de componentes periféricos) (p. ej., PCI-Express), o cualquier otra interfaz y/o protocolo(s) de bus o de comunicación punto a punto, tal y como interconexión de alta velocidad NV-Link, o protocolos de interconexión conocidos en la técnica.

En un ejemplo, el uno o más procesador(es) 112 paralelos incorporan un circuito optimizado para procesamiento de gráficos y vídeo, que incluye, por ejemplo, un circuito de salida de vídeo, y constituye una unidad de procesamiento de gráficos (GPU). En otro ejemplo, el uno o más procesador(es) 112 paralelo(s) incorpora(n) un circuito optimizado para procesamiento para fines generales, a la vez que se preserva la arquitectura computacional subyacente, descrita en mayor detalle en la presente memoria. En incluso otro ejemplo, los componentes del sistema 100 informático pueden estar integrados con uno o más de otros elementos de sistema en un único circuito integrado. Por ejemplo, el uno o más procesador(es) 112 paralelo(s), un centro 105 de memoria, procesador(es) 102 y un centro 107 de E/S pueden estar integrados en un circuito integrado de sistema en un chip (SoC). De manera alternativa, los componentes del sistema 100 informático pueden estar integrados en un único paquete para formar una configuración de sistema en paquete (SIP). En un ejemplo al menos una porción de los componentes del sistema 100 informático puede estar

integrada en un módulo multichip (MCM), el cual puede estar interconectado con otros módulos multichip en un sistema informático modular.

Se podrá apreciar que el sistema 100 informático que se muestra en la presente memoria es ilustrativo y que es posible realizar variaciones y modificaciones. Esta topología de conexión, que incluye la cantidad y disposición de puentes, la cantidad de procesador(es) 102 y la cantidad de procesador(es) 112 paralelo(s), se puede modificar según se desee. Por ejemplo, en algunos ejemplos, la memoria 104 de sistema está conectada al(los) procesador(es) 102 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria 104 de sistema mediante el centro 105 de memoria y el(los) procesador(es) 102. En otras topologías alternativas, el(los) procesador(es) 112 paralelo(s) está(n) conectado(s) al centro 107 de E/S o directamente a uno del uno o más procesador(es) 102, en lugar de al centro 105 de memoria. En otros ejemplos, el centro 107 de E/S y el centro 105 de memoria puede estar integrado en un único chip. Algunos ejemplos pueden incluir dos o más conjuntos de procesador(es) 102 unidos mediante múltiples zócalos, que se pueden acoplar con dos o más instancias del(los) procesador(es) 112 paralelo(s).

Alguno de los componentes particulares que se muestran en la presente memoria son opcionales y pueden no estar incluidos en todas las implementaciones del sistema 100 informático. Por ejemplo, se admite cualquier cantidad de tarjetas o periféricos de complemento o se pueden eliminar algunos componentes. Además, algunas arquitecturas pueden utilizar diferente terminología para componentes similares a los ilustrados en la Figura 1. Por ejemplo, en algunas arquitecturas se puede hacer referencia al centro 105 de memoria como puente norte, mientras que se puede hacer referencia al centro 107 de E/S como puente sur.

La Figura 2A ilustra un procesador 200 paralelo, según un ejemplo. Los diversos componentes del procesador 200 paralelo se pueden implementar utilizando uno o más dispositivos de circuito integrado, tal y como procesadores programables, circuitos integrados de aplicación específica (ASIC) o disposición de puertos programables de campo (FPGA). El procesador 200 paralelo ilustrado es una variante del uno o más procesador(es) 112 paralelo(s) que se muestra(n) en la Figura 1, según un ejemplo.

En un ejemplo, el procesador 200 paralelo incluye una unidad 202 de procesamiento paralelo. La unidad de procesamiento paralelo incluye una unidad 204 de E/S que permite la comunicación con otros dispositivos, incluyendo otras instancias de la unidad 202 de procesamiento paralelo. La unidad 204 de E/S puede estar conectada directamente a otros dispositivos. En un ejemplo, la unidad 204 de E/S se conecta con otros dispositivos mediante el uso de un centro o interfaz de conmutación, tal y como un centro 105 de memoria. Las conexiones entre el centro 105 de memoria y la unidad 204 de E/S forman un enlace 113 de comunicación. Dentro de la unidad 202 de procesamiento paralelo, la unidad 204 de E/S se conecta con una interfaz 206 anfitriona y una barra cruzada 216 de memoria, donde la interfaz 206 anfitriona recibe comandos dirigidos a llevar a cabo operaciones de procesamiento y la barra cruzada 216 de memoria recibe comandos dirigidos a llevar a cabo operaciones de memoria.

Cuando la interfaz 206 anfitriona recibe una memoria intermedia de comandos mediante la unidad 204 de E/S, la interfaz 206 anfitriona puede dirigir operaciones de trabajo para llevar a cabo estos comandos en un *front end* 208. En un ejemplo, el *front end* 208 se acopla con un planificador 210, que está configurado para distribuir comandos u otros elementos de trabajo a una disposición 212 de clúster de procesamiento. En un ejemplo, el planificador 210 garantiza que la disposición 212 de clúster de procesamiento está configurada de manera apropiada y en un estado válido antes de que se distribuyan las tareas a los clústeres de procesamiento de la disposición 212 de clúster de procesamiento. En un ejemplo, el planificador 210 se implementa mediante una lógica de firmware que se ejecuta en un microcontrolador. El planificador 210 implementado en microcontrolador se puede configurar para llevar a cabo planificaciones y operaciones de distribución de trabajo complejas de granularidad gruesa o fina, lo que permite una apropiación rápida y una conmutación de contexto de hilos que ejecutan la disposición 212 de procesamiento. En un ejemplo, el software anfitrión puede verificar los volúmenes de trabajo para su planificación en la disposición 212 de procesamiento mediante uno de múltiples *doorbells* de procesamiento de gráficos. Los volúmenes de trabajo pueden entonces distribuirse automáticamente a lo largo de la disposición 212 de procesamiento mediante la lógica del planificador 210 dentro del microcontrolador del planificador.

La disposición 212 de clúster de procesamiento puede incluir hasta "N" clústeres de procesamiento (p. ej., clúster 214A, clúster 214B, hasta clúster 214N). Cada clúster 214A-214N de la disposición 212 de clúster de procesamiento puede ejecutar una gran cantidad de hilos concurrentes. El planificador 210 puede asignar trabajo a los clústeres 214A-214N de la disposición 212 de clúster de procesamiento utilizando diversas planificaciones y/o algoritmos de distribución de trabajo, los cuales pueden variar dependiendo del volumen de trabajo que surja para cada tipo de programa o cálculo. El planificador 210 puede manejar la planificación de manera dinámica, o puede estar asistido en parte por una lógica de compilación durante la compilación de lógica de programa configurada para ser ejecutada por la disposición 212 de clúster de procesamiento. En un ejemplo, los diferentes clústeres 214A-214N de la disposición 212 de clúster de procesamiento se pueden asignar para procesar diferentes tipos de programas o para llevar a cabo diferentes tipos de cálculos.

La disposición 212 de clúster de procesamiento puede estar configurada para llevar a cabo diversos tipos de operaciones de procesamiento paralelo. En un ejemplo, la disposición 212 de clúster de procesamiento está configurada para llevar a cabo operaciones de cálculos paralelos para fines generales. Por ejemplo, la disposición 212

de clúster de procesamiento puede incluir una lógica para ejecutar tareas de procesamiento que incluyen filtrado de datos de vídeo y/o audio, llevar a cabo operaciones de modelación, que incluyen operaciones de física, y llevar a cabo transformaciones de datos.

En un ejemplo, la disposición 212 de clúster de procesamiento está configurada para llevar a cabo operaciones de procesamiento de gráficos paralelo. En ejemplos en las que el procesador 200 paralelo está configurado para llevar a cabo operaciones de procesamiento de gráficos, la disposición 212 de clúster de procesamiento puede incluir una lógica adicional para admitir la ejecución de dichas operaciones de procesamiento de gráficos, que incluyen, pero no están limitadas a, lógica de muestreo de textura para realizar operaciones de textura, así como lógica de teselación y otras lógicas de procesamiento de vértices. Además, la disposición 212 de clúster de procesamiento puede configurarse para ejecutar procesamientos de gráficos relacionados con programas de sombreado tal y como, pero no limitado a, sombreadores de vértices, sombreadores de teselación, sombreadores de geometría y sombreadores de píxeles. La unidad 202 de procesamiento paralelo puede transferir datos de la memoria de sistema mediante la unidad 204 de E/S para su procesamiento. Durante el procesamiento, los datos transferidos se pueden almacenar en la memoria en chip (p. ej., memoria 222 de procesador paralelo) durante el procesamiento, para luego escribirse nuevamente en la memoria de sistema.

En un ejemplo, cuando la unidad 202 de procesamiento paralelo se utiliza para llevar a cabo procesamiento de gráficos, el planificador 210 puede configurarse para dividir el volumen de trabajo de procesamiento en tareas de aproximadamente el mismo tamaño, para permitir una mejor distribución de las operaciones de procesamiento de gráficos en múltiples clústeres 214A-214N de la disposición 212 de clúster de procesamiento. En algunos ejemplos, las porciones de la disposición 212 de clúster de procesamiento se pueden configurar para llevar a cabo diferentes tipos de procesamiento. Por ejemplo, una primera porción puede configurarse para llevar a cabo un sombreado de vértices y generación de topología, una segunda porción puede configurarse para llevar a cabo la teselación y sombreado de geometría, y una tercera porción puede configurarse para llevar a cabo un sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen representada para su visualización. Se pueden almacenar datos intermedios producidos por uno o más de los clústeres 214A-214N en memorias intermedias para permitir que los datos intermedios se transmitan entre clústeres 214A-214N para procesamientos adicionales.

Durante el funcionamiento, la disposición 212 de clúster de procesamiento puede recibir tareas de procesamiento a ejecutar a través del planificador 210, el cual recibe comandos que definen tareas de procesamiento del *front end* 208. Para operaciones de procesamiento de gráficos, las tareas de procesamiento pueden incluir índices de datos a procesar, p. ej., datos de superficie (parche), datos de primitiva, datos de vértices y/o datos de píxeles, así como también parámetros y comandos de estado que definen cómo se han de procesar los datos (p. ej., qué programa se ha de ejecutar). El planificador 210 puede configurarse para capturar los índices correspondientes a las tareas o puede recibir los índices del *front end* 208. El *front end* 208 puede configurarse para garantizar que la disposición 212 de clúster de procesamiento está configurada en un estado válido antes de que se inicie el volumen de trabajo especificado por las memorias intermedias de comandos entrantes (p. ej., memorias intermedias por lotes, memorias intermedias de empuje, etc.).

Cada una de la una o más instancias de la unidad 202 de procesamiento paralelo se puede acoplar con la memoria 222 de procesador paralelo. Se puede acceder a la memoria 222 de procesador paralelo mediante la barra cruzada 216 de memoria, la cual puede recibir solicitudes de memoria de la disposición 212 de clúster de procesamiento, así como también de la unidad 204 de E/S. La barra cruzada 216 de memoria puede acceder a la memoria 222 de procesador paralelo a través de una interfaz 218 de memoria. La interfaz 218 de memoria puede incluir múltiples unidades de partición (p. ej., unidad 220A de partición, unidad 220B de partición, hasta la unidad 220N de partición) cada una de las cuales se puede acoplar a una porción (p. ej., una unidad de memoria) de memoria 222 de procesador paralelo. En una implementación, la cantidad de unidades 220A-220N de partición está configurada para ser igual a la cantidad de unidades de memoria, de manera que una primera unidad 220A de partición tiene una primera unidad 224A de memoria correspondiente, una segunda unidad 220B de partición tiene una unidad 224B de memoria correspondiente y una Nésima unidad 220N de partición tiene una Nésima unidad 224N de memoria correspondiente. En otros ejemplos, la cantidad de unidades 220A-220N de partición puede no ser igual a la cantidad de dispositivos de memoria.

En diversos ejemplos, las unidades 224A-224N de memoria pueden incluir diversos tipos de dispositivos de memoria, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria gráfica de acceso aleatorio, tal y como una memoria gráfica de acceso aleatorio sincrónico (SGRAM), que incluye memoria gráfica de tasa de datos doble (GDDR). En un ejemplo, las unidades 224A-224N de memoria también pueden incluir una memoria apilada 3D, que incluye, pero no está limitada a, una memoria de ancho de banda alto (HBM). Los expertos en la técnica apreciarán que la implementación específica de las unidades 224A-224N de memoria puede variar y que se puede seleccionar uno de varios diseños convencionales. Los objetivos de representación, tal y como memorias intermedias de tramas o mapas de textura se pueden almacenar en todas las unidades 224A-224N de memoria, lo que permite que las unidades 220A-220N de partición escriban porciones de cada objetivo de representación en paralelo para utilizar de manera eficiente el ancho de banda disponible de la memoria 222 de procesador paralelo. En algunos ejemplos, una instancia local de la memoria 222 de procesador paralelo puede excluirse para favorecer un diseño de memoria unificado que utiliza una memoria de sistema en conjunto con una memoria caché local.

En un ejemplo, cualquiera de los clústeres 214A-214N de la disposición 212 de clúster de procesamiento puede procesar datos que se escribirán en cualquiera de las unidades 224A-224N de memoria dentro de la memoria 222 de procesador paralelo. La barra cruzada 216 de memoria puede configurarse para transferir la salida de cada clúster 214A-214N a cualquier unidad 220A-220N de partición o a cualquier otro clúster 214A-214N, que puede llevar a cabo operaciones de procesamiento adicionales sobre la salida. Cada clúster 214A-214N puede comunicarse con la interfaz 218 de memoria a través de la barra cruzada 216 de memoria para leer de o escribir en diversos dispositivos de memoria externos. En un ejemplo, la barra cruzada 216 de memoria tiene una conexión con la interfaz 218 de memoria para comunicarse con la unidad 204 de E/S, así como una conexión con una instancia local de la memoria 222 de procesador paralelo, lo que permite que las unidades de procesamiento dentro de los diferentes clústeres 214A-214N de procesamiento se comuniquen con la memoria de sistema u otra memoria que no es local a la unidad 202 de procesamiento paralelo. En un ejemplo, la barra cruzada 216 de memoria puede utilizar canales virtuales para separar flujos de tráfico entre los clústeres 214A-214N y las unidades 220A-220N de partición.

A pesar de que se ilustra una única instancia de la unidad 202 de procesamiento paralelo dentro del procesador 200 paralelo, se puede incluir cualquier cantidad de instancias de la unidad 202 de procesamiento paralelo. Por ejemplo, se pueden proporcionar múltiples instancias de la unidad 202 de procesamiento paralelo en una única tarjeta de complemento, o múltiples tarjetas de complemento pueden estar interconectadas. Las diferentes instancias de la unidad 202 de procesamiento paralelo se pueden configurar para interoperar incluso si las diferentes instancias tienen diferentes cantidades de núcleos de procesamiento, diferentes cantidades de memoria de procesador paralelo local y/u otras diferencias de configuración. Por ejemplo, algunas instancias de la unidad 202 de procesamiento paralelo pueden incluir unidades de coma flotante de mayor precisión con respecto a otras instancias. Se pueden implementar sistemas que incorporan una o más instancias de la unidad 202 de procesamiento paralelo o del procesador 200 paralelo en una variedad de configuraciones y factores de forma, que incluyen, pero no se limitan a, ordenadores de mesa, portátiles, ordenadores personales de bolsillo, servidores, estaciones de trabajo, consolas de videojuegos y/o sistemas incorporados.

La Figura 2B es un diagrama de bloque de una unidad 220 de partición, según un ejemplo. En un ejemplo, la unidad 220 de partición es una instancia de una de las unidades 220A-220N de partición de la Figura 2A. Tal y como se ilustra, la unidad 220 de partición incluye una caché 221 de L2, una interfaz 225 de memoria intermedia de trama y una ROP 226 (unidad de operaciones de ráster). La caché 221 de L2 es una caché de lectura/escritura que está configurada para llevar a cabo operaciones de carga y almacenamiento recibidas de la barra cruzada 216 de memoria y de la ROP 226. Los fallos de lectura y solicitudes de anulación urgentes se emiten por la caché 221 de L2 a la interfaz 225 de memoria intermedia de trama para su procesamiento. También se pueden enviar actualizaciones a la memoria intermedia de trama mediante la interfaz 225 de memoria intermedia de trama para su procesamiento. En un ejemplo, la interfaz 225 de memoria intermedia de trama hace interfaz con una de las unidades de memoria en la memoria de procesador paralelo, tal y como las unidades 224A-224N de memoria de la Figura 2 (p. ej., dentro de la memoria 222 de procesador paralelo).

En aplicaciones gráficas, la ROP 226 es una unidad de procesamiento que lleva a cabo operaciones de ráster, tal y como plantillas, prueba Z, combinación de imágenes y operaciones similares. Entonces, la ROP 226 emite los datos de gráficos procesados que se almacenan en la memoria gráfica. En algunos ejemplos, la ROP 226 incluye una lógica de compresión que comprime datos de profundidad o color que están escritos en la memoria y descomprimen datos de profundidad o color que se leen de la memoria. La lógica de compresión puede ser una lógica de compresión sin pérdida que hace uso de uno o más de múltiples algoritmos de compresión. El tipo de compresión que lleva a cabo la ROP 226 puede variar en base a las características estadísticas de los datos a comprimir. Por ejemplo, la compresión de color delta se lleva a cabo en datos de profundidad y color por cada mosaico.

En algunos ejemplos, la ROP 226 está incluida dentro de cada clúster de procesamiento (p. ej., clúster 214A-214N de la Figura 2) en lugar de dentro de la unidad 220 de partición. En un ejemplo tal, las solicitudes de lectura y escritura por dato de píxel se transmiten a través de la barra cruzada 216 de memoria en lugar de datos de fragmento de píxel. Los datos de gráficos procesados se pueden visualizar en un dispositivo de visualización, tal y como uno del uno o más dispositivo(s) 110 de visualización de la Figura 1, enrutados para procesamientos adicionales por el(los) procesador(es) 102, o enrutados para procesamientos adicionales por una de las entidades de procesamiento dentro del procesador 200 paralelo de la Figura 2A.

La Figura 2C es un diagrama de bloque de un clúster 214 de procesamiento dentro de una unidad de procesamiento paralelo según un ejemplo. En un ejemplo, el clúster de procesamiento es una instancia de uno de los clústeres 214A-214N de procesamiento de la Figura 2. El clúster 214 de procesamiento puede configurarse para ejecutar muchos hilos en paralelo, donde el término "hilo" se refiere a una instancia de un programa específico que se ejecuta en un conjunto específico de datos de entrada. En algunos ejemplos, se utilizan técnicas de emisión de instrucciones de instrucción única para múltiples datos (SIMD) para admitir la ejecución paralela de una gran cantidad de hilos sin proporcionar múltiples unidades de instrucción independientes. En otros ejemplos, las técnicas de instrucción única para múltiples hilos (SIMT) se utilizan para admitir la ejecución paralela de una gran cantidad de hilos generalmente sincronizados, utilizando una unidad de instrucción común configurada para emitir instrucciones a un conjunto de motores de procesamiento dentro de cada uno de los clústeres de procesamiento. A diferencia del régimen de ejecución SIMD, donde todos los motores de procesamiento generalmente ejecutan instrucciones idénticas, la ejecución SIMT permite que sea más fácil que hilos diferentes sigan trayectos de ejecución divergentes a través de

un programa de hilos determinado. Los expertos en la técnica comprenderán que un régimen de procesamiento SIMD representa un subconjunto funcional de un régimen de procesamiento SIMT.

La operación del clúster 214 de procesamiento se puede controlar mediante un gestor 232 de canalización que distribuye tareas de procesamiento a los procesadores paralelos SIMT. El gestor 232 de canalización recibe instrucciones del planificador 210 de la Figura 2 y gestiona la ejecución de estas instrucciones mediante un microprocesador 234 de gráficos y/o una unidad 236 de textura. El microprocesador 234 gráfico ilustrado es una instancia de ejemplo de un procesador paralelo SIMT. Sin embargo, se pueden incluir diversos tipos de procesadores paralelos SIMT de arquitecturas diferentes dentro del clúster 214 de procesamiento. Una o más instancias del multiprocesador 234 gráfico se puede(n) incluir dentro de un clúster 214 de procesamiento. El multiprocesador 234 gráfico puede procesar datos y se puede utilizar una barra cruzada 240 de datos para distribuir los datos procesados a uno de múltiples destinos posibles, que incluyen otras unidades sombreadoras. El gestor 232 de canalización puede facilitar la distribución de datos procesados especificando los destinos de los datos procesados a distribuir a través de la barra cruzada 240 de datos.

Cada multiprocesador 234 gráfico dentro del clúster 214 de procesamiento puede incluir un conjunto idéntico de lógica de ejecución funcional (p. ej., unidades aritmético-lógicas, unidades de almacenamiento de carga, etc.). La lógica de ejecución funcional se puede configurar de una forma canalizada en la que se pueden emitir nuevas instrucciones antes de que se completen las instrucciones anteriores. La lógica de ejecución funcional admite una variedad de operaciones que incluyen aritmética de enteros y de coma flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y cálculo de diversas funciones algebraicas. En un ejemplo, el mismo hardware de unidad funcional puede influenciarse para llevar a cabo diferentes operaciones y puede tener cualquier combinación de unidades funcionales.

Las instrucciones transmitidas al clúster 214 de procesamiento constituyen un hilo. Un conjunto de hilos que se ejecutan a lo largo de un conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa sobre diferentes datos de entrada. Se puede asignar cada hilo dentro de un grupo de hilos a un motor de procesamiento diferente dentro de un multiprocesador 234 gráfico. Un grupo de hilos puede incluir menos hilos que la cantidad de motores de procesamiento dentro del multiprocesador 234 gráfico. Cuando un grupo de hilos incluye menos hilos que la cantidad de motores de procesamiento, uno o más de los motores de procesamiento pueden estar inactivos durante los ciclos en los cuales dicho grupo de hilos está siendo procesado. Un grupo de hilos también puede incluir más hilos que la cantidad de motores de procesamiento dentro del multiprocesador 234 gráfico. Cuando el grupo de hilos incluye más hilos que la cantidad de motores de procesamiento dentro del multiprocesador 234 gráfico, el procesamiento se puede llevar a cabo en ciclos de reloj consecutivos. En un ejemplo, se pueden ejecutar múltiples grupos de hilos de manera concurrente en un multiprocesador 234 gráfico.

En un ejemplo, el multiprocesador 234 gráfico incluye una memoria caché interna para realizar operaciones de carga y almacenamiento. En un ejemplo, el multiprocesador 234 gráfico puede anteponer una caché interna y utilizar una memoria caché (p. ej., caché 308 de L1) dentro del clúster 214 de procesamiento. Cada multiprocesador 234 gráfico también tiene acceso a cachés de L2 dentro de las unidades de partición (p. ej., unidades 220A-220N de partición de la Figura 2) que se comparten entre todos los clústeres 214 de procesamiento y se pueden utilizar para transferir datos entre hilos. El multiprocesador 234 gráfico puede también acceder a la memoria global fuera de chip, que puede incluir una o más de una memoria de procesador paralelo local y/o una memoria de sistema. Cualquier memoria externa a la unidad 202 de procesamiento paralelo se puede utilizar como memoria global. Los ejemplos en las que el clúster 214 de procesamiento incluye múltiples instancias del multiprocesador 234 gráfico puede compartir instrucciones y datos comunes, que se pueden almacenar en la caché 308 de L1.

Cada clúster 214 de procesamiento puede incluir una MMU 245 (unidad de gestión de memoria) que está configurada para mapear direcciones virtuales con direcciones físicas. En otros ejemplos, una o más instancias de la MMU 245 pueden residir dentro de la interfaz 218 de memoria de la Figura 2. La MMU 245 incluye un conjunto de entradas de tabla de paginación (PTE) utilizadas para mapear una dirección virtual con una dirección física de un mosaico y, opcionalmente, con un índice de línea de caché. La MMU 245 puede incluir memorias intermedias de traducción anticipada (TLB) de dirección o cachés que pueden residir dentro del multiprocesador 234 gráfico o la caché de L1 o clúster 214 de procesamiento. La dirección física se procesa para distribuir la localidad de acceso de datos de superficie para permitir un intercalado eficiente de solicitudes entre unidades de partición. El índice de línea de caché se puede utilizar para determinar si una solicitud para una línea de caché es un acierto o un fallo.

En aplicaciones gráficas e informáticas, un clúster 214 de procesamiento se puede configurar de manera que cada multiprocesador 234 gráfico se acople a una unidad 236 de textura para llevar a cabo operaciones de mapeo de textura, p. ej., determinar posiciones de muestra de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen de una caché de L1 de textura interna (no se muestra) o, en algunos ejemplos, de la caché de L1 dentro del multiprocesador 234 gráfico y se captura de una caché de L2, memoria de procesador paralelo local o memoria de sistema, según sea necesario. Cada multiprocesador 234 gráfico emite las tareas procesadas a la barra cruzada 240 de datos para proporcionar la tarea procesada a otro clúster 214 de procesamiento para su procesamiento adicional o para almacenar la tarea procesada en una caché de L2 local, memoria de procesador paralelo local o memoria de sistema a través de la barra cruzada 216 de memoria. Una pre-ROP 242 (unidad de operaciones pre-ráster) está configurada para recibir datos del multiprocesador 234 gráfico, dirigir datos a unidades ROP, que pueden

estar ubicadas en unidades de partición como se describe en la presente memoria (p. ej., unidades 220A-220N de partición de la Figura 2). La unidad 242 pre-ROP puede llevar a cabo optimizaciones para combinación de colores, organizar datos de color de píxeles, y realizar traducciones de direcciones.

Se podrá apreciar que la arquitectura de núcleo descrita en la presente memoria es ilustrativa y que es posible realizar variaciones y modificaciones. Se puede incluir cualquier cantidad de unidades de procesamiento, p. ej., multiprocesador 234 gráfico, unidades 236 de textura, pre-ROP 242, etc., dentro de un clúster 214 de procesamiento. Además, a pesar de que solo se muestra un clúster 214 de procesamiento, una unidad de procesamiento paralelo tal y como se describe en la presente memoria puede incluir cualquier cantidad de instancias del clúster 214 de procesamiento. En un ejemplo, cada clúster 214 de procesamiento puede configurarse para operar de manera independiente de otros clústeres 214 de procesamiento utilizando unidades de procesamiento, cachés de L1, etc., separadas y distintas.

La Figura 2D muestra un multiprocesador 234 gráfico, según un ejemplo. En un ejemplo tal, el multiprocesador 234 gráfico se acopla con el gestor 232 de canalización del clúster 214 de procesamiento. El multiprocesador 234 gráfico tiene una canalización de ejecución que incluye, pero no está limitada a, una caché 252 de instrucción, una unidad 254 de instrucción, una unidad 256 de mapeo de direcciones, un fichero 258 de registro, uno o más núcleos 262 de unidad de procesamiento gráfico para fines generales (GPGPU) y una o más unidades 266 de carga/almacenamiento. Los núcleos 262 de GPGPU y unidades 266 de carga/almacenamiento están acoplados con una memoria 272 caché y memoria 270 compartida mediante una interconexión 268 de memoria y caché.

En un ejemplo, la caché 252 de instrucción recibe un flujo de instrucciones para ejecutar desde el gestor 232 de canalización. Las instrucciones se incluyen en la caché 252 de instrucción y se despachan para que las ejecute la unidad 254 de instrucción. La unidad 254 de instrucción puede despachar instrucciones como grupos de hilos (p. ej., urdiembres), con cada hilo del grupo de hilos asignado a una unidad de ejecución diferente dentro del núcleo 262 de GPGPU. Una instrucción puede acceder a cualquier espacio de direcciones locales, compartidas o globales al especificar una dirección dentro de un espacio de dirección unificada. La unidad 256 de mapeo de direcciones se puede utilizar para traducir direcciones en el espacio de direcciones unificadas en una dirección de memoria distinta a la que las unidades 266 de carga/almacenamiento pueden acceder.

El fichero 258 de registro proporciona un conjunto de registros para las unidades funcionales del multiprocesador 234 gráfico. El fichero 258 de registro proporciona almacenamiento temporal para operandos conectados a los trayectos de datos de las unidades funcionales (p. ej., núcleos 262 de GPGPU, unidades 266 de carga/almacenamiento) del multiprocesador 234 gráfico. En un ejemplo, el fichero 258 de registro está dividido entre cada una de las unidades funcionales de manera que cada unidad funcional está asignada a una porción dedicada del fichero 258 de registro. En un ejemplo, el fichero 258 de registro está dividido entre los diferentes urdiembres que el multiprocesador 234 gráfico está ejecutando.

Cada uno de los núcleos 262 de GPGPU puede incluir unidades de coma flotante (FPU) y/o unidades aritmético-lógicas (ALU) que se utilizan para ejecutar instrucciones del multiprocesador 234 gráfico. Los núcleos 262 de GPGPU pueden ser de arquitectura similar o de arquitectura diferente, según los ejemplos. Por ejemplo, una primera porción de los núcleos 262 de GPGPU incluye una FPU de simple precisión y una ALU de enteros, mientras que una segunda porción de los núcleos de GPGPU incluye una FPU de doble precisión. En un ejemplo, las FPU pueden implementar el estándar IEEE 754-2008 para aritmética en coma flotante o permitir aritmética en coma flotante de precisión variable. El multiprocesador 234 gráfico puede además incluir una o más unidades de función fija o función especial para llevar a cabo funciones específicas, tal y como copiar operaciones de combinación de rectángulo o de píxel. En un ejemplo, uno o más de los núcleos de GPGPU también pueden incluir una lógica de función fija o especial.

En un ejemplo, los núcleos 262 de GPGPU incluyen una lógica SIMD capaz de llevar a cabo una instrucción única en múltiples conjuntos de datos. En un ejemplo, los núcleos 262 de GPGPU pueden ejecutar físicamente instrucciones SIMD4, SIMD8 y SIMD16 y ejecutar lógicamente instrucciones SIMD1, SIMD2 y SIMD32. Las instrucciones SIMD para los núcleos de GPGPU se pueden generar en el momento de compilación mediante un compilador de sombreado o se pueden generar automáticamente cuando se ejecutan programas escritos y compilados por arquitecturas de programa único para múltiples datos (SPMD) o SIMT. Múltiples hilos de un programa configurado para el modelo de ejecución SIMT se pueden ejecutar a través de una instrucción SIMD única. Por ejemplo, ocho hilos SIMT que llevan a cabo la misma operación o similar se pueden ejecutar en paralelo mediante una unidad lógica SIMD8 única.

La interconexión 268 de memoria y caché es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador 234 gráfico con el fichero 258 de registro y con la memoria 270 compartida. En un ejemplo, la interconexión 268 de memoria y caché es una interconexión de barra cruzada que permite a la unidad 266 de carga/almacenamiento implementar operaciones de carga y almacenamiento entre la memoria 270 compartida y el fichero 258 de registro. El fichero 258 de registro puede operar en la misma frecuencia que los núcleos 262 de GPGPU, por tanto, la transferencia de datos entre los núcleos 262 de GPGPU y el fichero 258 de registro es de muy baja latencia. La memoria 270 compartida se puede utilizar para permitir la comunicación entre hilos que ejecutan las unidades funcionales dentro del multiprocesador 234 gráfico. La memoria 272 caché se puede utilizar como caché de datos, por ejemplo, para hacer la caché de los datos de textura comunicados entre las unidades funcionales y la unidad 236 de textura. La memoria 270 compartida también se puede utilizar como una caché gestionada mediante programa.

Los hilos que se ejecutan en los núcleos 262 de GPGPU pueden almacenar datos de manera programada dentro de la memoria compartida además de los datos automáticamente en caché que se almacenan dentro de la memoria 272 caché.

Las Figuras 3A-3B ilustran multiprocesadores gráficos adicionales, según los ejemplos. Los microprocesadores 325, 350 gráficos ilustrados son variantes de los multiprocesadores 234 gráficos de la Figura 2C. Los multiprocesadores 325, 350 gráficos ilustrados se pueden configurar como un multiprocesador de flujo (SM) capaz de ejecutar simultáneamente una gran cantidad de hilos de ejecución.

La Figura 3A muestra un multiprocesador 325 gráfico según un ejemplo adicional. El multiprocesador 325 gráfico incluye múltiples instancias adicionales de unidades de recursos de ejecución en relación con el microprocesador 234 gráfico de la Figura 2D. Por ejemplo, el multiprocesador 325 gráfico puede incluir múltiples instancias de la unidad 332A-332B de instrucción, del fichero 334A-334B de registro y de la(s) unidad(es) 344A-344B de textura. El multiprocesador 325 gráfico también incluye múltiples conjuntos de gráficos o unidades de ejecución de cálculos (p. ej., núcleo 336A-336B de GPGPU, núcleo 337A-337B de GPGPU, núcleo 338A-338B de GPGPU) y múltiples conjuntos de unidades 340A-340B de carga/almacenamiento. En un ejemplo las unidades de recursos de ejecución tienen una caché 330 de instrucción común, memoria 342 caché de textura y/o datos y memoria 346 compartida.

Los diversos componentes pueden comunicarse mediante un tejido 327 de interconexión. En un ejemplo, el tejido 327 de interconexión incluye uno o más conmutadores de barra cruzada para permitir la comunicación entre los diversos componentes del multiprocesador 325 gráfico. En un ejemplo, el tejido 327 de interconexión es una capa de tejido de red de alta velocidad separada sobre la cual se apila cada componente del multiprocesador 325 gráfico. Los componentes del multiprocesador 325 gráfico se comunican con los componentes remotos mediante el tejido 327 de interconexión. Por ejemplo, cada uno de los núcleos 336A-336B, 337A-337B y 3378A-338B de GPGPU se puede comunicar con la memoria 346 compartida mediante el tejido 327 de interconexión. El tejido 327 de interconexión puede mediar en la comunicación dentro del microprocesador 325 gráfico para garantizar una asignación justa del ancho de banda entre componentes.

La Figura 3B muestra un multiprocesador 350 gráfico según un ejemplo adicional. El procesador gráfico incluye múltiples conjuntos de recursos 356A-356D de ejecución, donde cada conjunto de recursos de ejecución incluye múltiples unidades de instrucción, ficheros de registro, núcleos de GPGPU y unidades de carga almacenamiento, según se ilustra en la Figura 2D y la Figura 3A. Los recursos 356A-356D de ejecución pueden trabajar en consonancia con la(s) unidad(es) 360A-360D de textura para operaciones de textura, a la vez que comparten una caché 354 de instrucción y una memoria 362 compartida. En un ejemplo, los recursos 356A-356D de ejecución pueden compartir una caché 354 de instrucción y una memoria 362 compartida, además de múltiples instancias de una memoria 358A-358B de caché de textura y/o datos. Los diversos componentes pueden comunicarse a través de un tejido 352 de interconexión similar al tejido 327 de interconexión de la Figura 3A.

Los expertos en la técnica comprenderán que la arquitectura descrita en las Figuras 1, 2A-2D y 3A-3B son descriptivas. Por tanto, las técnicas descritas en la presente memoria se pueden implementar en cualquier unidad de procesamiento configurada de manera apropiada, incluyendo, sin limitación, uno o más procesadores de aplicación móvil, una o más unidades de procesamiento central (CPU) de mesa o de servidor, que incluyen CPU multinúcleo, una o más unidades de procesamiento paralelo, tal y como la unidad 202 de procesamiento paralelo de la Figura 2, así como uno o más procesadores gráficos o unidades de procesamiento para fines específicos.

En algunos ejemplos, un procesador paralelo o GPGPU tal y como se describe en la presente memoria está acoplado en comunicación con núcleos anfitriones/procesadores para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrones, y diversas funciones de GPU para fines generales (GPGPU). La GPU puede estar acoplada en comunicación con el procesador anfitrión/núcleos a través de un bus u otra interconexión (p. ej., interconexión de alta velocidad, como PCIe o NVLink). En otros ejemplos, la GPU puede estar integrada en el mismo paquete o chip que los núcleos y puede estar acoplada en comunicación con los núcleos a través de un bus/interconexión de procesador interno (es decir, interno para el paquete o chip). Independientemente de la manera en la que la GPU está conectada, los núcleos de procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidos en un descriptor de trabajo. La GPU entonces utiliza un circuito/lógica dedicado para procesar de manera eficiente estos comandos/instrucciones.

Técnicas de GPU para interconexión de procesador anfitrión

La Figura 4A ilustra una arquitectura de ejemplo en la que una pluralidad de GPU 410-413 se acoplan en comunicación con una pluralidad de procesadores 405-406 multinúcleo a través de enlaces 440-443 de alta velocidad (p. ej., buses, interconexiones punto a punto, etc.). En un ejemplo, los enlaces 440-443 de alta velocidad admiten un rendimiento de comunicación de 4GB/s, 30GB/s, 80GB/s o más, dependiendo de la implementación. Se pueden utilizar diversos protocolos de interconexión que incluyen, pero no están limitados a, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes descritos en esta memoria no están limitados a ningún protocolo de comunicación o rendimiento en particular.

Además, en un ejemplo, dos o más de las GPU 410-413 están interconectadas a través de enlaces 444-445 de alta velocidad, que se pueden implementar utilizando los mismos protocolos/enlaces utilizados para enlaces 440-443 de alta velocidad o diferentes. De manera similar, se pueden conectar dos o más de los procesadores 405-406 multinúcleo a través de un enlace 433 de alta velocidad, que pueden ser buses de multiprocesador simétrico (SMP) que operan a 20GB/s, 30GB/s, 120GB/s o más. De manera alternativa, toda comunicación entre los diversos componentes de sistema que se muestran en la Figura 4A se puede lograr utilizando los mismos protocolos/enlaces (p. ej., sobre un tejido de interconexión común). Tal y como se menciona, sin embargo, los principios subyacentes descritos en esta memoria no están limitados a ningún tipo de tecnología de interconexión en particular.

En un ejemplo, cada procesador 405-406 multinúcleo se acopla en comunicación con una memoria 401-402 de procesador, a través de interconexiones 430-431 de memoria, respectivamente, y cada GPU 410-413 está acoplada en comunicación con una memoria 420-423 de GPU a través de interconexiones 450-453 de memoria de GPU, respectivamente. Las interconexiones 430-431 y 450-453 de memoria pueden utilizar las mismas tecnologías de acceso de memoria o diferentes. A modo de ejemplo, y no como limitación, las memorias 401-402 de procesador y las memorias 420-423 de GPU pueden ser memorias volátiles tal y como memorias de acceso aleatorio dinámico (DRAM) (que incluye DRAM apiladas), Graphics DDR SDRAM (GDDR) (p. ej., GDDR5, GDDR6) o memoria de ancho de banda alto (HBM) y/o pueden ser memorias no volátiles 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 (p. ej., utilizando una jerarquía de memoria de dos niveles [2LM]).

Tal y como se describe a continuación, a pesar de que los diversos procesadores 405-406 y GPU 410-413 puedan estar físicamente acoplados a una memoria 401-402, 420-423 particular, respectivamente, se puede implementar una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtual (también denominado espacio de "dirección efectiva") se distribuye entre todas las diversas memorias físicas. Por ejemplo, cada una de las memorias 401-402 de procesador puede comprender 64GB del espacio de direcciones de la memoria de sistema y cada una de las memorias 420-423 de GPU puede comprender 32GB del espacio de direcciones de la memoria de sistema (lo que da un total de 256GB de memoria direccionable en este ejemplo).

La Figura 4B ilustra detalles adicionales para una interconexión entre un procesador 407 multinúcleo y un módulo 446 de aceleración gráfica según un ejemplo. El módulo 446 de aceleración gráfica puede incluir uno o más chips de GPU integrados en una tarjeta de línea, que está acoplada al procesador 407 a través del enlace 440 de alta velocidad. De manera alternativa, el módulo 446 de aceleración gráfica 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 461A-461D de traducción anticipada y una o más cachés 462A-462D. Los núcleos pueden incluir otros diversos componentes para ejecutar instrucciones y procesar datos que no están ilustrados para evitar dificultar los principios subyacentes descritos en esta memoria (p. ej., unidades de captura de instrucción, unidades de predicción de salto, descodificadores, unidades de ejecución, memorias intermedias de reorden, etc.). Las cachés 462A-462D pueden comprender cachés de nivel 1 (L1) y de nivel 2 (L2). Además, se pueden incluir una o más cachés 426 compartidas en la jerarquía de caché y compartir por conjuntos de los núcleos 460A-460D. Por ejemplo, el procesador 407 incluye 24 núcleos, cada uno con su propio caché de L1, doce cachés de L2 compartidas y doce cachés de L3 compartidas. En este ejemplo, una de las cachés de L2 y de L3 está compartida por dos núcleos adyacentes. El procesador 407 y el módulo 446 de integración de acelerador gráfico se conectan con la memoria 441 de sistema, la cual puede incluir memorias 401-402 de procesador.

Se mantiene la coherencia de los datos e instrucciones almacenadas en las diversas cachés 462A-462D, 456 y la memoria 441 de sistema a través de la comunicación entre núcleos a través de un bus 464 de coherencia. Por ejemplo, cada caché puede tener una lógica/circuito de coherencia asociada entre ellos para comunicarse a través del bus 464 de coherencia, en respuesta a lecturas o escrituras detectadas para líneas de caché particulares. En una implementación, un protocolo de rastreo de caché se implementa a través del bus 464 de coherencia para rastrear accesos de caché. Los expertos en la técnica comprenden bien las técnicas de rastreo/coherencia de caché y no se describirán en detalle en la presente memoria para evitar dificultar los principios subyacentes descritos en esta memoria.

En un ejemplo, el circuito 425 de proxy acopla en comunicación al módulo 446 de aceleración gráfica con el bus 464 de coherencia, lo que permite que el módulo 446 de aceleración gráfica participe en el protocolo de coherencia de caché como par de los núcleos. En particular, una interfaz 435 proporciona conectividad al circuito 425 de proxy a través de un enlace 440 de alta velocidad (p. ej., un bus PCIe, NVLink, etc.) y una interfaz 437 conecta el módulo 446 de aceleración gráfica con el enlace 440 de alta velocidad.

En una implementación, un circuito 436 de integración de acelerador proporciona gestión de caché, acceso de memoria, gestión de contexto y servicios de gestión de interrupción en representación de una pluralidad de motores 431, 432, N de procesamiento de gráficos del módulo 446 de aceleración gráfica. Cada uno de los motores 431, 432, N de procesamiento de gráficos puede comprender una unidad de procesamiento de gráficos (GPU) separada. De manera alternativa, los motores 431, 432, N de procesamiento de gráficos pueden comprender diferentes tipos de motores de procesamiento de gráficos dentro de una GPU, tal y como unidades de ejecución de gráficos, motores de

procesamiento de medios (p. ej., codificadores/descodificadores de vídeo), sacamuestras y motores blit. Dicho de otro modo, el módulo de aceleración gráfica puede ser una GPU con una pluralidad de motores 431-432, N de procesamiento de gráficos o los motores 431-432, N de procesamiento de gráficos pueden ser GPU individuales integradas en un paquete, tarjeta de línea o chip común.

En un ejemplo, el circuito 436 de integración de acelerador incluye una unidad de gestión de memoria (MMU) 439 para llevar a cabo diversas funciones de gestión de memoria, tal y 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 441 de sistema. La MMU 439 también puede incluir una memoria intermedia de traducción anticipada (TLB) (no se muestra) para hacer la caché de las traducciones de direcciones virtuales/efectivas a físicas/reales. En un ejemplo, el circuito 436 de integración de acelerador incluye una unidad 491 de captura para capturar comandos, instrucciones, descriptores de trabajo, etc., que definen operaciones a realizar. En una implementación, una caché 438 almacena comandos y datos para que los motores 431-432, N de procesamiento de gráficos puedan acceder a ellos de manera eficiente. En un ejemplo, los datos almacenados en caché 438 y memorias 433-434, N gráficas se mantienen en coherencia con las cachés 462A-462D, 456 de núcleo y memoria 411 de sistema. Tal y como se menciona, esto se puede lograr mediante un circuito 425 proxy que forma parte del mecanismo de coherencia de caché en representación de la caché 438 y memorias 433-434, N (p. ej., enviando actualizaciones a la caché 438 en relación con modificaciones/accesos de líneas de caché en cachés 462A-462D, 456 de procesador y recibiendo actualizaciones de la caché 438).

Un conjunto de registros 449 almacenan datos de contexto para hilos ejecutados por los motores 431-432, N de procesamiento de gráficos y un circuito 448 de gestión de contexto que gestiona los contextos de hilo. Por ejemplo, el circuito 448 de gestión de contexto puede llevar a cabo operaciones de guardado y recuperación para guardar y recuperar contextos de los diversos hilos durante conmutaciones de contextos (p. ej., cuando se guarda un primer hilo y un segundo hilo se almacena para que un motor de procesamiento de gráficos ejecute el segundo hilo). Por ejemplo, en un conmutador de contexto, el circuito 448 de gestión de contexto puede almacenar valores de registro actuales en una región designada de la memoria (p. ej., identificada por un puntero de contexto). Puede más tarde recuperar los valores de registro cuando vuelve al contexto. En un ejemplo, un circuito 447 de gestión de interrupciones recibe y procesa interrupciones recibidas de dispositivos de sistema.

En una implementación, las direcciones virtuales/efectivas de un motor 431 de procesamiento de gráficos se traducen en direcciones reales/físicas en la memoria 411 de sistema mediante la MMU 439. Un ejemplo del circuito 436 de integración de acelerador admite múltiples (p. ej., 4, 8, 16) módulos 446 de acelerador de gráficos y/u otros dispositivos aceleradores. El módulo 446 de acelerador gráfico se puede dedicar a una única aplicación ejecutada en el procesador 407 o puede estar compartido entre múltiples aplicaciones. En un ejemplo, se presenta un entorno de ejecución de gráficos virtual en el que los recursos de los motores 431-432, N de procesamiento de gráficos se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos pueden estar subdivididos en "segmentos" (*slices*) que están asignados a diferentes VM y/o aplicaciones basadas en los requisitos de procesamiento y prioridades asociados con los VM y/o aplicaciones.

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

Debido a que los recursos de hardware de los motores 431-432, N de procesamiento de gráficos se mapean de manera explícita al espacio de direcciones reales que ve el procesador 407 anfitrión, cualquier procesador anfitrión puede direccionar estos recursos directamente utilizando un valor de dirección efectiva. Una función del circuito 436 de integración de acelerador, en un ejemplo, es la separación física de los motores 431-432, N de procesamiento de gráficos de manera que aparezcan en el sistema como unidades independientes.

Como se menciona, en el ejemplo ilustrado, una o más memorias 433-434, M gráficas están acopladas a cada uno de los motores 431-432, N de procesamiento de gráficos, respectivamente. Las memorias 433-434, M gráficas almacenan instrucciones y datos que cada motor 431-432, N de procesamiento de gráficos está procesando. Las memorias 433-434, M gráficas pueden ser memorias volátiles, tal y como DRAM (incluidas DRAM apiladas), memorias GDDR (p. ej., GDDR5, GDDR6) o HBM, y/o pueden ser memorias no volátiles, tal y como 3D XPoint o Nano-Ram.

En un ejemplo, para reducir el tráfico de datos a través del enlace 440 de alta velocidad, se utilizan técnicas de polarización para garantizar que los datos almacenados en las memorias 433-434, M gráficas son datos que los motores 431-432, N de procesamiento de gráficos utilizarán muy frecuentemente y, preferiblemente, no utilizarán los núcleos 460A-460D (al menos no frecuentemente). De manera similar, el mecanismo de polarización trata de mantener los datos que necesitan los núcleos (y, preferiblemente, no los motores 431-432, N de procesamiento de gráficos) dentro de las cachés 462A-462D, 456 de los núcleos y de la memoria 411 del sistema.

La Figura 4C ilustra otro ejemplo en la que el circuito 436 de integración de acelerador está integrado dentro del procesador 407. En este ejemplo, los motores 431-432, N de procesamiento de gráficos se comunican directamente a

través del enlace 440 de alta velocidad con el circuito 436 de integración de acelerador a través de la interfaz 437 y la interfaz 435 (que, nuevamente, puede utilizar cualquier forma de bus o protocolo de interfaz). El circuito 436 de integración de acelerador puede realizar las mismas operaciones que las descritas con respecto a la Figura 4B, pero posiblemente con un rendimiento mayor dada su gran proximidad con el bus 462 de coherencia y las cachés 462A-462D, 426.

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

En un ejemplo del modelo de proceso dedicado, los motores 431-432, N de procesamiento de gráficos están dedicados a una aplicación o proceso único en un sistema operativo único. La aplicación única puede canalizar otras solicitudes de aplicación a los motores 431-432, N gráficos, proporcionando una virtualización dentro de una VM/partición.

En los modelos de programación de procesos dedicados, los motores 431-432, N de procesamiento de gráficos pueden compartirse entre múltiples VM/particiones de aplicación. Los modelos compartidos requieren un hipervisor de sistema para virtualizar los motores 431-432, N de procesamiento de gráficos para permitir el acceso de cada sistema operativo. Para sistemas de partición única sin un hipervisor, los motores 431-432, N de procesamiento de gráficos pertenecen al sistema operativo. En ambos casos, el sistema operativo puede virtualizar los motores 431-432, N de procesamiento de gráficos para proporcionar acceso a cada proceso o aplicación.

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

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

El módulo 446 de aceleración de gráficos y/o los motores 431-432, N de procesamiento de gráficos individual pueden estar compartidos por todos o por un subconjunto de los procesos del sistema. Los ejemplos incluyen una infraestructura para configurar el estado de proceso y enviar un WD 484 a un módulo 446 de aceleración de gráficos para iniciar un trabajo en un entorno virtualizado.

En una implementación, el modelo de programación de proceso dedicado es específico de la implementación. En este modelo, un proceso único es poseedor del módulo 446 de aceleración de gráficos o de un motor 431 de procesamiento de gráficos individual. Debido a que el módulo 446 de aceleración de gráficos pertenece a un proceso único, el hipervisor inicializa el circuito 436 de integración de acelerador para la partición de pertenencia y el sistema operativo inicializa el circuito 436 de integración de acelerador para el proceso de pertenencia en el momento en el que se asigna el módulo 446 de aceleración de gráficos.

Durante el funcionamiento, una unidad 491 de captura de WD en el segmento 490 de integración de acelerador captura el siguiente WD 484 que incluye una indicación del trabajo a realizar por uno de los motores de procesamiento de gráficos del módulo 446 de aceleración de gráficos. Los datos del WD 484 se pueden almacenar en registros 449 y los puede utilizar la MMU 439, el circuito 447 de gestión de interrupciones y/o el circuito 446 de gestión de contexto según se ilustra. Por ejemplo, la MMU 439 incluye un circuito de recorrido de segmento/página para acceder a las tablas 486 de segmento/página dentro del espacio 485 de direcciones virtuales del OS. El circuito 447 de gestión de interrupciones puede procesar eventos 492 de interrupción recibidos del módulo 446 de aceleración de gráficos. Cuando se realizan operaciones de gráficos, la MMU 439 traduce una dirección 493 efectiva generada por un motor 431-432, N de procesamiento de gráficos en una dirección real.

En un ejemplo, el mismo conjunto de registros 449 se duplica para cada motor 431-432, N de procesamiento de gráficos y/o módulo 446 de aceleración de gráficos y se puede inicializar por el sistema hipervisor u operativo. Cada uno de estos registros duplicados puede estar incluido en un segmento 490 de integración de acelerador. En la Tabla 1 se muestran registros de ejemplo que el hipervisor puede inicializar.

Tabla 1 - Registros inicializados por Hipervisor

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

En la Tabla 2 se muestran registros de ejemplo que el sistema operativo puede inicializar.

Tabla 2 - Registros inicializados por sistema operativo

1	Identificación de proceso e hilo
2	Puntero de guardado/recuperación de contexto de dirección efectiva (EA)
3	Puntero de registro de utilización de acelerador de dirección virtual (VA)
4	Puntero de tabla de segmento de almacenamiento de dirección virtual (VA)
5	Máscara de autoridad
6	Descriptor de trabajo

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

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

Los modelos de programación compartida permiten que todos o un subconjunto de procesos de todos o de un subconjunto de particiones en el sistema utilicen un módulo 446 de aceleración de gráficos. Hay dos modelos de programación donde el módulo 446 de aceleración de gráficos está compartido por múltiples procesos y particiones: compartido por tiempo-segmento y compartido por dirección de gráficos.

En este modelo, el hipervisor 496 de sistema es poseedor del módulo 446 de aceleración de gráficos y hace que su función esté disponible para todos los sistemas 495 operativos. Para que un módulo 446 de aceleración de gráficos admita la virtualización mediante el hipervisor 496 de sistema, el módulo 446 de aceleración de gráficos puede adherirse a los siguientes requisitos: 1) Una solicitud de trabajo de una aplicación puede ser autónoma (es decir, no es necesario mantener el estado entre trabajos) o el módulo 446 de aceleración de gráficos debe proporcionar un mecanismo de guardado y recuperación de contexto. 2) Una solicitud de trabajo de una aplicación está garantizada por el módulo 446 de aceleración de gráficos para finalizarla en una cantidad de tiempo especificada, lo que incluye cualquier fallo de traducción, o el módulo 446 de aceleración de gráficos proporciona la capacidad de apropiarse del

procesamiento del trabajo. 3) Al módulo 446 de aceleración de gráficos se le debe garantizar imparcialidad entre procesos cuando operan en el modelo de programación compartida dirigida.

En un ejemplo, para el modelo compartido, se requiere que la aplicación 480 haga un llamado a un sistema 495 operativo con un tipo de módulo 446 de aceleración de gráficos, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de guardado/recuperación de contexto (CSRP). El tipo de módulo 446 de aceleración de gráficos describe la función de aceleración diana para la llamada de sistema. El tipo de módulo 446 de aceleración de gráficos puede ser un valor específico de sistema. El WD está formateado específicamente para el módulo 446 de aceleración de gráficos y puede tener forma de un comando de módulo 446 de aceleración de gráficos, un puntero de direcciones efectivas para una estructura definida por el usuario, un puntero de direcciones efectivas para una cola de comandos, o cualquier otra estructura de datos para describir el trabajo a realizar por el módulo 446 de aceleración de gráficos. En un ejemplo, el valor AMR es el estado AMR a utilizar para el proceso actual. El valor trasladado al sistema operativo es similar a una aplicación que configura el AMR. Si las implementaciones de circuito 436 de integración de acelerador y el módulo 446 de aceleración de gráficos no admiten un registro de anulación de máscara de autoridad de usuario (UAMOR), el sistema operativo puede aplicar el valor UAMOR actual al valor AMR antes de trasladar el AMR en la llamada de hipervisor. El hipervisor 496 puede opcionalmente aplicar el valor de registro de anulación de máscara de autoridad (AMOR) antes de colocar el AMR en el elemento 483 de proceso. En un ejemplo, el CSRP es uno de los registros 449 que contiene la dirección efectiva de un área en el espacio 482 de direcciones de la aplicación para el módulo 446 de aceleración de gráficos para guardar y recuperar el estado de contexto. El puntero es opcional si no se requiere que se guarde ningún estado entre trabajos o cuando se apropia un trabajo. El área de guardado/recuperación de contexto puede ser una memoria de sistema anclada.

Al recibir la llamada de sistema, el sistema 495 operativo puede verificar que la aplicación 480 ha registrado y dado la autoridad para utilizar el módulo 446 de aceleración de gráficos. El sistema 495 operativo después llama al hipervisor 496 con la información que se muestra en la tabla 3.

Tabla 3 - OS a parámetros de llamada de hipervisor

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (posiblemente enmascarado).
3	Un puntero de área de guardado/recuperación de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) e ID de hilo opcional (TID)
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)

Al recibir la llamada del hipervisor, el hipervisor 496 verifica que el sistema 495 operativo ha registrado y dado la autoridad para utilizar el módulo 446 de aceleración de gráficos. El hipervisor 496 después coloca el elemento 483 de proceso en la lista enlazada de elementos de proceso para el tipo de módulo 446 de aceleración de gráficos correspondiente. El elemento de proceso puede incluir la información que se muestra 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) (posiblemente enmascarado).
3	Un puntero de área de guardado/recuperación de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) e ID de hilo opcional (TID)
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 vector de interrupción, derivada de los parámetros de llamada del hipervisor
9	Un valor de registro de estado (SR)
10	Un ID de partición 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 449 del segmento 490 de integración de acelerador.

Tal y como se ilustra en la Figura 4F, un ejemplo emplea una memoria unificada direccionable a través de un espacio de direcciones de memoria virtual común utilizado para acceder a las memorias 401-402 de procesador físico y las memorias 420-423 de GPU. En esta implementación, las operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de direcciones de memoria virtual/efectiva para acceder a las memorias 401-402 de procesadores y viceversa, simplificando así su programabilidad. En un ejemplo, una primera porción del espacio de direcciones virtuales/efectivas está asignado a la memoria 401 de procesador, una segunda porción a la segunda memoria 402 de procesador, una tercera porción a la memoria 420 de GPU, y así sucesivamente. El espacio de memoria virtual/efectiva completo (a menudo denominado el espacio de direcciones efectivas) está, por tanto, distribuido a lo largo de cada una de las memorias 401-402 de procesador y memorias 420-423 de GPU, lo que permite que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual mapeada a dicha memoria.

En un ejemplo, el circuito 494A-494E de gestión de polarización/coherencia dentro de una o más de las MMU 439A-439E garantiza coherencia de la caché entre las cachés de los procesadores anfitriones (p. ej., 405) y las GPU 410-413 e implementa técnicas de polarización que indican las memorias físicas en las que se deben almacenar determinados tipos de datos. A pesar de que en la Figura 4F se ilustran múltiples instancias de circuitos 494A-494E de gestión de polarización/coherencia, el circuito de desviación/coherencia se puede implementar dentro de la MMU de uno o más procesadores 405 anfitriones y/o dentro del circuito 436 de integración de acelerador.

Un ejemplo permite que la memoria 420-423 fijada a la GPU se mapee como parte de la memoria de sistema y se acceda utilizando tecnología de memoria virtual compartida (SVM), pero sin tener las desventajas de rendimiento típicas asociadas con la coherencia de caché de sistema completo. La capacidad de que a la memoria 420-423 fijada a la GPU se acceda como memoria de sistema sin una tara de coherencia de caché onerosa proporciona un entorno operativo beneficioso para la descarga de la GPU. Esta disposición permite que el software de procesador 405 anfitrión configure operandos y resultados computacionales de acceso sin la tara de copias de datos DMA de E/S tradicionales. Dichas copias tradicionales involucran llamadas de controlador, interrupciones y accesos de E/S de mapeo de memoria (MMIO) que son todos ineficientes con respecto a simples accesos de memoria. Al mismo tiempo, la capacidad de acceder a la memoria 420-423 fijada a la GPU sin taras 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 flujo sustancial, por ejemplo, la tara de coherencia de caché puede reducir de manera significativa el ancho de banda de escritura efectivo que ve una GPU 410-413. La eficiencia de la configuración de operando, la eficiencia de acceso a resultados y la eficiencia del cálculo de GPU cumplen todos un papel al momento de determinar la efectividad de descarga de GPU.

En una implementación, la selección entre la polarización de GPU y la polarización de procesador anfitrión está impulsada por una estructura de datos de seguimiento de polarización. Se puede utilizar, por ejemplo, una tabla de polarización 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 fijada a la GPU. La tabla de polarización se puede implementar en un rango de memoria robada de una o más memorias 420-423 fijadas a la GPU, con o sin una caché de polarización en la GPU 410-413 (p. ej., para hacer la caché de entradas utilizadas frecuentemente/recientemente de la tabla de polarización). De manera alternativa, la tabla de polarización completa puede mantenerse dentro de la GPU.

En una implementación, la entrada de la tabla de polarización asociada con cada acceso a la memoria 420-423 acoplada a la GPU se accede antes del acceso real a la memoria de GPU, lo que provoca las siguientes operaciones. En primer lugar, solicitudes locales de la GPU 410-413 que encuentran su página en la polarización de GPU se reenvían directamente a una memoria 420-423 de GPU correspondiente. Las solicitudes locales de la GPU que encuentran su página en polarización de anfitrión se reenvían al procesador 405 (p. ej., a través de un enlace de alta velocidad tal y como se describe más arriba). En un ejemplo, las solicitudes del procesador 405 que encuentran la página solicitada en la polarización del procesador anfitrión finalizan la solicitud como una lectura de memoria normal. De manera alternativa, las solicitudes dirigidas a una página polarizada de GPU se pueden reenviar a la GPU 410-

413. La GPU puede entonces hacer la transición de la página a una polarización de procesador anfitrión si no está utilizando la página en ese momento.

El estado de polarización de una página se puede cambiar ya sea mediante un mecanismo basado en software, un mecanismo basado en software y asistido por hardware, o, para un conjunto limitado de casos, un mecanismo basado únicamente en hardware.

Un mecanismo para cambiar el estado de polarización emplea una llamada API (p. ej., OpenCL) que, a su vez, llama al controlador de dispositivo de 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 polarización y, para algunas transiciones, realice una operación de evacuación de caché en el anfitrión. Se requiere la operación de evacuación de caché para una transición de polarización de procesador 405 anfitrión a polarización de GPU, pero no se requiere para la transición opuesta.

En un ejemplo la coherencia de caché se mantiene al representar temporalmente que el procesador 405 anfitrión no puede hacer la caché de páginas polarizadas de GPU. Para acceder a estas páginas, el procesador 405 puede solicitar acceso de la GPU 410 que puede o no otorgar acceso inmediatamente, dependiendo de la implementación. Por tanto, para reducir la comunicación entre el procesador 405 y la GPU 410 es beneficioso garantizar que las páginas polarizadas de GPU sean aquellas requeridas por la GPU, pero no por el procesador 405 anfitrión, y viceversa.

Canalización de procesamiento de gráficos

La Figura 5 ilustra una canalización 500 de procesamiento de gráficos, según un ejemplo. En un ejemplo, un procesador gráfico puede implementar la canalización 500 de procesamiento de gráficos ilustrada. El procesador gráfico puede estar incluido dentro de los subsistemas de procesamiento paralelo tal y como se describe en la presente memoria, tal y como el procesador 200 paralelo de la Figura 2, que, en un ejemplo, es una variante del(los) procesador(es) 112 paralelo(s) de la Figura 1. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización 500 de procesamiento de gráficos a través de una o más instancias de la unidad de procesamiento paralelo (p. ej., unidad 202 de procesamiento paralelo de la Figura 2) tal y como se describe en la presente memoria. Por ejemplo, una unidad sombreadora (p. ej., multiprocesador 234 gráfico de la Figura 3) puede estar configurada para realizar las funciones de una o más de una unidad 504 de procesamiento de vértices, una unidad 508 de procesamiento de control de teselación, una unidad 512 de procesamiento de evaluación de teselación, una unidad 516 de procesamiento de geometría y una unidad 524 de procesamiento de fragmentos/píxeles. Las funciones de ensamblador 502 de datos, ensambladores 506, 514, 518 de primitivas, unidad 510 de teselación, rasterizador 522 y unidad 526 de operaciones de ráster también se pueden llevar a cabo mediante otros motores de procesamiento dentro de un clúster de procesamiento (p. ej., clúster 214 de procesamiento de la Figura 3) y una unidad de partición correspondiente (p. ej., unidad 220A-220N de partición de la Figura 2). La canalización 500 de procesamiento de gráficos también se puede implementar utilizando unidades de procesamiento dedicadas para una o más funciones. En un ejemplo, una o más porciones de la canalización 500 de procesamiento de gráficos pueden llevarse a cabo mediante lógica de procesamiento paralelo dentro de un procesador para fines generales (p. ej., CPU). En un ejemplo, una o más porciones de la canalización 500 de procesamiento de gráficos puede acceder a la memoria en chip (p. ej., memoria 222 de procesador paralelo como en la Figura 2) a través de una interfaz 528 de memoria, que puede ser una instancia de la interfaz 218 de memoria de la Figura 2.

En un ejemplo, el ensamblador 502 de datos es una unidad de procesamiento que recolecta datos de vértices para superficies y primitivas. El ensamblador 502 de datos después emite los datos de vértices, incluyendo los atributos de vértices, hacia la unidad 504 de procesamiento de vértices. La unidad 504 de procesamiento de vértices es una unidad de ejecución programable que ejecuta programas de sombreador de vértices, datos de vértices de iluminación y transformación según lo especifican los programas de sombreador de vértices. La unidad 504 de procesamiento de vértices lee datos que están almacenados en la memoria caché, local o de sistema para utilizarlos en el procesamiento de datos de vértices y puede estar programada para transformar los datos de vértices de una representación de coordenadas basada en objeto en un espacio de coordenadas de espacio global o en un espacio de coordenadas de dispositivo normalizado.

Una primera instancia de un ensamblador 506 de primitivas recibe atributos de vértices de la unidad 50 de procesamiento de vértices. Las lecturas del ensamblador 506 de primitivas almacena atributos de vértices según sea necesario y construye primitivas gráficas para que las procese la unidad 508 de procesamiento de control de teselación. Las primitivas gráficas incluyen triángulos, segmentos de línea, puntos, superficies a trozos, etc., tal y como admiten diversas interfaces de programación de aplicación (API) de procesamiento de gráficos.

La unidad 508 de procesamiento de control de teselación trata los vértices de entrada como puntos de control para una superficie a trozos geométrica. Los puntos de control se transforman de una representación de entrada desde la superficie a trozos (p. ej., las bases de superficies a trozos) a una representación que es apropiada para que utilice la unidad 512 de procesamiento de evaluación de teselación en la evaluación de superficie. La unidad 508 de procesamiento de control de teselación también puede calcular factores de teselación para bordes de superficies a trozos geométricas. Un factor de teselación aplica a un borde único y cuantifica un nivel de detalle que depende de la vista asociada al borde. Una unidad 510 de teselación está configurada para recibir los factores de teselación para bordes de una superficie a trozos y para teselar la superficie a trozos en múltiples primitivas geométricas tal y como

primitivas de línea, de triángulo o cuadrilateral, que se transmiten a una unidad 512 de procesamiento de evaluación de teselación. La unidad 512 de procesamiento de evaluación de teselación opera con coordenadas parametrizadas de la superficie a trozos subdividida para generar una representación de superficie y atributos de vértices para cada vértice asociado a primitivas geométricas.

Una segunda instancia de un ensamblador 514 de primitivas recibe atributos de vértices de la unidad 512 de procesamiento de evaluación de teselación, donde lee los atributos de vértices almacenados según sea necesario, y construye primitivas de gráficas para su procesamiento por parte de la unidad 516 de procesamiento geométrico. La unidad 516 de procesamiento geométrico es una unidad de ejecución programable que ejecuta programas de sombreador geométrico para transformar primitivas gráficas recibidas de un ensamblador 514 de primitivas, según lo especifican los programas de sombreador de vértices. En un ejemplo, la unidad 516 de procesamiento geométrico está programada para subdividir las primitivas gráficas en una o más primitivas gráficas y calcular parámetros utilizados para rasterizar las nuevas primitivas gráficas.

En algunos ejemplos, la unidad 516 de procesamiento geométrico puede añadir o eliminar elementos en el flujo de geometrías. La unidad 516 de procesamiento geométrico emite los parámetros y vértices especificando nuevas primitivas gráficas al ensamblador 518 de primitivas. El ensamblador 518 de primitivas recibe los parámetros y vértices de la unidad 516 de procesamiento geométrico y construye primitivas gráficas para su procesamiento mediante una unidad 520 de escala de sector de visualización, selección y recorte. La unidad 516 de procesamiento geométrico lee datos que están almacenados en una memoria de procesador paralelo o memoria de sistema para su uso en el procesamiento de datos geométricos. La unidad 520 de escala de sector de visualización, selección y recorte realiza recortes, selecciones y escalas de sectores de visualización y emite las primitivas gráficas procesadas a un rasterizador 522.

El rasterizador 522 puede realizar selecciones de profundidad y otras optimizaciones basadas en profundidad. El rasterizador 522 también realiza una conversión de escaneo sobre las nuevas primitivas gráficas para generar fragmentos y emitir dichos fragmentos y datos de cobertura asociados a la unidad 524 de procesamiento de fragmentos/pixeles. La unidad 524 de procesamiento de fragmentos/píxeles es una unidad de ejecución programable que está configurada para ejecutar programas de sombreador de fragmentos o programas de sombreador de píxeles. La unidad 524 de procesamiento de fragmentos/píxeles transforma fragmentos o píxeles recibidos del rasterizador 522, según lo especifican los programas de sombreador de fragmentos o píxeles. Por ejemplo, la unidad 524 de procesamiento de fragmentos/píxeles puede estar programada para llevar a cabo operaciones que incluyen, pero no están limitadas a, mapeo de texturas, sombreados, combinaciones de imágenes, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se emiten a una unidad 526 de operaciones de ráster. La unidad 524 de procesamiento de fragmentos/píxeles puede leer datos que están almacenados ya sea en la memoria de procesador paralelo o en la memoria de sistema para su uso cuando se procesan los datos de fragmento. Los programas de sombreador de fragmentos o píxeles se pueden configurar para sombrear en granularidades de muestra, píxel, mosaico u otras granularidades dependiendo de la velocidad de muestreo configurada para las unidades de procesamiento.

La unidad 526 de operaciones de ráster es una unidad de procesamiento que realiza operaciones de ráster que incluyen, pero no están limitadas a, plantillas, pruebas z, combinaciones de imágenes y operaciones similares, y emite datos de píxeles como datos de gráficos procesados a almacenar en la memoria de gráficos (p. ej., memoria 222 de procesador paralelo como en la Figura 2, y/o memoria 104 de sistema como en la Figura 1, que se visualizará en el uno más dispositivo(s) 110 de visualización o para que uno o más de los procesador(es) 102 o procesador(es) 112 paralelo(s) lleven a cabo procesamientos adicionales. En algunos ejemplos, la unidad 526 de operaciones de ráster está configurada para comprimir datos z o de color que están escritos en la memoria y descomprimir datos z o de color que se leen de la memoria.

Visión general del aprendizaje automático

Un algoritmo de aprendizaje automático es un algoritmo que puede aprender en base a un conjunto de datos. Los ejemplos de algoritmos de aprendizaje automático se pueden diseñar para modelar abstracciones de alto nivel dentro de un conjunto de datos. Por ejemplo, los algoritmos de reconocimiento de imágenes se pueden utilizar para determinar a cuál de diversas categorías pertenece una entrada determinada; algoritmos de regresión pueden emitir un valor numérico dependiendo de una entrada; y se pueden utilizar algoritmos de reconocimiento de patrones para generar texto traducido o llevar a cabo un reconocimiento de texto a voz y/o de voz.

Un ejemplo de tipo de algoritmo de aprendizaje automático es una red neuronal. Hay muchos tipos de redes neuronales; un tipo de red neuronal simple es una red predictiva. Una red predictiva se puede implementar como un gráfico acíclico en el que los nodos están dispuestos en capas. Generalmente, una topología de red predictiva 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 salidas 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 predictiva se propagan (es decir, "se predicen") a los nodos de la capa de salida a través de una función de activación que calcula los estados de los nodos de cada capa sucesiva en la red en base a coeficientes ("pesos")

asociados respectivamente con cada uno de los bordes que conectan las capas. Dependiendo del modelo específico que representa el algoritmo que se está ejecutando, la salida del algoritmo de red neuronal puede adquirir diversas formas.

Antes de que se pueda utilizar un algoritmo de aprendizaje automático para modelar un problema particular, el algoritmo se entrena utilizando un conjunto de datos de entrenamiento. Entrenar una red neuronal implica seleccionar una topología de red, utilizar un conjunto de datos de entrenamiento que representa un problema que la red está modelando, y ajustar los pesos hasta que el modelo red se lleva a cabo 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 se ajustan los pesos asociados con las conexiones para minimizar el 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 de manera significativa por la calidad del conjunto de datos utilizados para entrenar al algoritmo. El proceso de entrenamiento puede ser computacionalmente intensivo y puede requerir una cantidad de tiempo significativo en un procesador para fines generales convencional. Por consiguiente, se utiliza un hardware de procesamiento paralelo para entrenar diversos 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 redes neuronales se prestan naturalmente para implementaciones paralelas. Específicamente, muchos algoritmos de aprendizaje automático y aplicaciones de software se han adaptado para hacer uso del hardware de procesamiento paralelo dentro de dispositivos de procesamiento de gráficos para fines generales.

La Figura 6 es un diagrama generalizado de una pila 600 de software de aprendizaje automático. Una aplicación 602 de aprendizaje automático se puede configurar para entrenar una red neuronal utilizando un conjunto de datos de entrenamiento o para utilizar una red neuronal profunda para implementar inteligencia artificial. La aplicación 602 de aprendizaje automático puede incluir una funcionalidad de entrenamiento e inferencia para una red neuronal y/o software especializado que se puede utilizar para entrenar una red neuronal antes de su despliegue. La aplicación 602 de aprendizaje automático puede implementar cualquier tipo de inteligencia artificial que incluye, pero no está limitada a, reconocimiento de imágenes, mapeo y localización, navegación autónoma, síntesis de voz, imágenes médicas o traducción de idiomas.

La aceleración de hardware para la aplicación 602 de aprendizaje automático se puede permitir a través de una estructura 604 de aprendizaje automático. La estructura 604 de aprendizaje automático puede proporcionar una biblioteca de primitivas de aprendizaje automático. Las primitivas de aprendizaje automático son operaciones básicas que se llevan a cabo comúnmente por algoritmos de aprendizaje automático. Sin la estructura 604 de aprendizaje automático, a los desarrolladores de algoritmos de aprendizaje automático se les requeriría que creasen y optimizasen la lógica computacional principal asociada con el algoritmo de aprendizaje automático, y después que reoptimizasen la lógica computacional a medida que se desarrollan procesadores paralelos nuevos. En cambio, la aplicación de aprendizaje automático se puede configurar para realizar los cálculos necesarios utilizando las primitivas proporcionadas por la estructura 604 de aprendizaje automático. Las primitivas de ejemplo incluyen convoluciones de tensor, funciones de activación y reducciones, que son operaciones computacionales que se llevan a cabo mientras se entrena una red neuronal convolucional (CNN). La estructura 604 de aprendizaje automático también puede proporcionar primitivas para implementar subprogramas algebraicos lineales básicos realizados por muchos algoritmos de aprendizaje automático, tal y como operaciones de matriz y de vector.

La estructura 604 de aprendizaje automático puede procesar datos de entrada recibidos de la aplicación 602 de aprendizaje automático y generar la entrada apropiada para una estructura 606 de cálculo. La estructura 606 de cálculo puede abstraer las instrucciones subyacentes proporcionadas al controlador 608 de GPGPU para permitir que la estructura 604 de aprendizaje automático aproveche la aceleración de hardware a través del hardware 610 de GPGPU sin requerir que la estructura 604 de aprendizaje automático tenga un conocimiento profundo de la arquitectura del hardware 610 de GPGPU. Además, la estructura 606 de cálculo puede permitir la aceleración de hardware para la estructura 604 de aprendizaje automático a largo de una variedad de tipos y generaciones de hardware 610 de GPGPU.

Aceleración de aprendizaje automático de GPGPU

La Figura 7 ilustra una unidad 700 de procesamiento de gráficos para fines generales muy paralelos, según un ejemplo. En un ejemplo, la unidad 700 de procesamiento para fines generales (GPGPU) puede estar configurada para ser particularmente eficiente al procesar el tipo de volúmenes de trabajo computacionales asociados con el entrenamiento de redes neuronales profundas. Además, la GPGPU 700 puede estar enlazada directamente a otras instancias de la GPGPU para crear un clúster multi-GPU para mejorar la velocidad de capacitación para, en particular, redes neuronales profundas.

La GPGPU 700 incluye una interfaz 702 anfitriona para permitir una conexión con un procesador anfitrión. En un ejemplo, la interfaz 702 anfitriona es una interfaz PCI Express. Sin embargo, la interfaz anfitriona también puede ser una interfaz de comunicación o tejido de comunicación específico del proveedor. La GPGPU 700 recibe comandos del procesador anfitrión y utiliza un planificador 704 global para distribuir los hilos de ejecución asociados con dichos comandos a un conjunto de clústeres 706A-706H de cálculos. Los clústeres 706A-706H de cálculos comparten una memoria 708 caché. La memoria 708 caché puede servir como una caché de mayor nivel para memorias caché dentro de los clústeres 706A-706H de cálculos.

La GPGPU 700 incluye una memoria 714A-714B acoplada con los clústeres 706A-H de cálculo a través de un conjunto de controladores 712A-712B de memoria. En diversos ejemplos, la memoria 714A-714B puede incluir diversos tipos de dispositivos de memoria, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria gráfica de acceso aleatorio, tal y como memoria gráfica de acceso aleatorio sincrónico (SGRAM), que incluye memoria gráfica de tasa de datos doble (GDDR) o memoria apilada 3D, que incluye, pero no está limitada a, una memoria de ancho de banda alto (HBM).

En un ejemplo, cada clúster 706A-706H de cálculo incluye un conjunto de multiprocesadores gráficos, tal y como el multiprocesador 400 gráfico de la Figura 4A. Los multiprocesadores gráficos de los múltiples tipos de clúster de cálculo de enteros y unidades lógicas de coma flotante que pueden realizar operaciones computacionales en un rango de precisiones que incluye el apropiado para cálculos de aprendizaje automático. Como ejemplo y en un ejemplo, al menos un subconjunto de las unidades de coma flotante en cada uno de los clústeres 706A-H de cálculos se pueden configurar para llevar a cabo operaciones de coma flotante de 16 bits o 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante se puede configurar para llevar a cabo operaciones de coma flotante de 64 bits.

Se pueden configurar múltiples instancias de la GPGPU 700 para operar como un clúster de cálculo. El mecanismo de comunicación utilizado por el clúster de cálculo para sincronización e intercambio de datos varía en los ejemplos. En un ejemplo, las múltiples instancias de la GPGPU 700 se comunican a través de la interfaz 702 anfitriona. En un ejemplo, la GPGPU 700 incluye un centro 709 de E/S que acopla la GPGPU 700 con un enlace 710 de GPU que permite una conexión directa con otras instancias de la GPGPU. En un ejemplo, el enlace 710 de GPU está acoplado a un puente de GPU a GPU dedicado que permite la comunicación y sincronización entre múltiples instancias de la GPGPU 700. En un ejemplo, el enlace 710 de GPU 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 700 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red que es accesible a través de la interfaz 702 anfitriona. En un ejemplo, el enlace 710 de GPU puede estar configurado para permitir una conexión con un procesador anfitrión además de o como alternativa a la interfaz 702 anfitriona.

A pesar de que la configuración ilustrada de la GPGPU 700 se puede configurar para entrenar redes neuronales, un ejemplo proporciona una configuración alternativa de la GPGPU 700 que se puede configurar para despliegue dentro de una plataforma de alto rendimiento o de inferencia de baja potencia. En una configuración de inferencia, la GPGPU 700 incluye menos de los clústeres 706A-H de cálculo en relación con la configuración de entrenamiento. Además, la tecnología de memoria asociada con la memoria 714A-714B puede variar entre configuraciones de inferencia y de entrenamiento. En un ejemplo la configuración de inferencia de la GPGPU 700 puede admitir instrucciones de inferencia específicas. Por ejemplo, una configuración de inferencia da soporte a una o más instrucciones de producto escalar entero de 8 bits, que se utilizan comúnmente durante las operaciones de inferencia para redes neuronales desplegadas.

La Figura 8 ilustra un sistema 800 informático multi-GPU, según un ejemplo. El sistema 800 informático multi-GPU puede incluir un procesador 802 acoplado a múltiples GPGPU 806A-806D a través de un conmutador 804 de interfaz anfitriona. En un ejemplo, el conmutador 804 de interfaz anfitriona es un dispositivo conmutador PCI Express que acopla el procesador 802 a un bus PCI Express a través del cual el procesador 802 se puede comunicar con el conjunto de GPGPU 806A-D. Cada una de las múltiples GPGPU 806A-806D puede ser una instancia de la GPGPU 700 de la Figura 7. Las GPGPU 806A-D pueden interconectarse a través de un conjunto de GPU de punto a punto de alta velocidad a enlaces 816 de GPU. La GPU de alta velocidad a enlaces de GPU pueden conectarse con cada una de las GPGPU 806A-806D a través de un enlace de GPU dedicado, tal y como el enlace 710 de GPU como se ve en la Figura 7. Los enlaces 816 de GPU de P2P permiten la comunicación directa entre cada una de las GPGPU 806A-806D sin requerir comunicaciones a través del bus de interfaz anfitriona al que está conectado el procesador 802. Con el tráfico de GPU a GPU dirigido a los enlaces de GPU de P2P, el bus de interfaz anfitriona permanece disponible para acceso a la memoria de sistema o para comunicarse con otras instancias del sistema 800 informático multi-GPU, por ejemplo, a través de uno o más dispositivos de red. A pesar de que en el ejemplo ilustrado las GPGPU 806A-806D se conectan al procesador 802 a través del conmutador 804 de interfaz anfitriona, en un ejemplo el procesador 802 incluye soporte directo para los enlaces 816 de GPU de P2P y se puede conectar directamente a las GPGPU 806A-806D.

Implementaciones de red neuronal de aprendizaje automático

La arquitectura informática proporcionada por los ejemplos descritos en la presente memoria se puede configurar para llevar a cabo los tipos de procesamiento paralelo que son particularmente apropiados para entrenar y desplegar redes neuronales para aprendizaje automático. Una red neuronal se puede generalizar como una red de funciones que

tienen una relación gráfica. Como es bien conocido en la técnica, hay una variedad de tipos de implementaciones de red neuronal utilizadas en aprendizaje automático. Un tipo ejemplar de red neuronal es la red predictiva, tal y como se ha descrito anteriormente.

Un segundo tipo de ejemplo de red neuronal es la red neuronal convolucional (CNN). Una CNN es una red neuronal predictiva especializada para procesamiento de datos que presenta una topología tipo grilla conocida, tal y como datos de imagen. Por consiguiente, las CNN se utilizan comúnmente para aplicaciones de visión artificial y reconocimiento de imágenes, pero también se pueden utilizar para otros tipos de reconocimiento de patrones, tal y como procesamiento de voz y de lenguaje. Los nodos en la capa de entrada de CNN están organizados en un conjunto de "filtros" (detectores de características inspirados en los campos receptivos que se encuentran en la retina), y en la salida de cada conjunto de filtros se propagan a nodos en capas sucesivas de la red. Los cálculos para una CNN incluyen aplicar la operación matemática de convolución para que cada filtro produzca la salida de dicho filtro. La convolución es un tipo especializado de operación matemática realizada por dos funciones para que produzca una tercera función que es una versión modificada de una de las dos funciones originales. En terminología de red convolucional, se hace referencia a la primera función de la convolución como la entrada, mientras que se hace referencia a la segunda función como el *kernel* de convolución. Se puede hacer referencia a la salida como el mapa de características. Por ejemplo, la entrada a una capa de convolución puede ser una disposición multidimensional de datos que define los diversos componentes de color como una imagen de entrada. El *kernel* de convolución puede ser una disposición multidimensional de parámetros, donde los parámetros están adaptados por el proceso de entrenamiento de la red neuronal.

Las redes neuronales recurrentes (RNN) son una familia de redes neuronales predictivas que incluyen conexiones de retroalimentación entre capas. Las RNN permiten la modelación de datos secuenciales al compartir datos de parámetros entre partes diferentes 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 un tiempo futuro, ya que al menos una porción de los datos de salida de la RNN se utilizan como retroalimentación para procesar entradas posteriores en una secuencia. Esta característica hace que las RNN sean particularmente útiles para el procesamiento de lenguaje debido a la naturaleza variable en la que se pueden componer los datos de lenguaje.

Las figuras descritas a continuación muestran redes predictivas, CNN y RNN de ejemplo, además de describir procesos generales para entrenar y desplegar respectivamente cada uno de esos tipos de redes. Se comprenderá que estas descripciones son de ejemplo y no limitantes en cuanto a que cualquier ejemplo específico descrita en la presente memoria y los conceptos ilustrados se pueden aplicar, en general, a redes neuronales profundas y técnicas de aprendizaje automático en general.

Las redes neuronales de ejemplo descritas anteriormente se pueden utilizar para realizar aprendizaje profundo. El aprendizaje profundo es aprendizaje automático que utiliza redes neuronales profundas. Las redes neuronales profundas utilizadas en aprendizaje profundo son redes neuronales artificiales compuestas por múltiples capas ocultas, en contraposición con redes neuronales superficiales que incluyen solo una única capa oculta. Las redes neuronales profundas requieren, generalmente, un entrenamiento intensivo en términos computacionales. Sin embargo, las capas ocultas adicionales de la red permiten el reconocimiento de patrones multietapas que resultan en un error de salida reducido en relación con las técnicas de aprendizaje automático superficial.

Las redes neuronales profundas utilizadas en aprendizaje profundo generalmente incluyen una red *front-end* para llevar a cabo un reconocimiento de características acoplado a una red *back-end* que representa un modelo matemático que puede llevar a cabo operaciones (p. ej., clasificación de objetos, reconocimiento de voz, etc.) en base a la representación de características provista en el modelo. El aprendizaje profundo permite que el aprendizaje automático se pueda realizar sin requerir que se lleve a cabo una ingeniería de características realizada a mano para el modelo. En cambio, las redes neuronales profundas pueden aprender características en base a una estructura o correlación estadística dentro de los datos de entrada. Se pueden proporcionar las características aprendidas a un modelo matemático que puede mapear las características detectadas hacia una salida. El modelo matemático utilizado por la red generalmente está especializado para la tarea específica a realizar y se pueden utilizar diferentes modelos para llevar a cabo tareas diferentes.

Una vez que la red neuronal está estructurada, se puede aplicar un modelo de aprendizaje para la red con el fin de entrenar la red para llevar a cabo tareas específicas. El modelo de aprendizaje describe cómo ajustar los pesos dentro del modelo para reducir el error de salida de la red. La propagación hacia atrás de errores es un método común que se utiliza 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 utilizando una función de pérdida y se calcula el valor de error para cada una de las neuronas en la capa de salida. Los valores de error después se propagan hacia atrás hasta que cada neurona tiene un valor de error asociado que representa en líneas generales su contribución a la salida original. La red puede entonces aprender de esos errores utilizando un algoritmo, tal y como un algoritmo de descenso de gradiente estocástico, para actualizar los pesos de la red neuronal.

La Figura 9A-9B ilustra una red neuronal convolucional de ejemplo. La Figura 9A ilustra diversas capas dentro de una CNN. Tal y como se muestra en la Figura 9A, una CNN de ejemplo utilizada para modelar el procesamiento de imágenes puede recibir una entrada 902 que describe los componentes rojo, verde y azul (RGB) de una imagen de

entrada. La entrada 902 se puede procesar mediante múltiples capas convolucionales (p. ej., capa 904 convolucional, capa 906 convolucional). La salida de las múltiples capas convolucionales puede procesarse de manera opcional mediante un conjunto de capas 908 completamente conectadas. Las neuronas en una capa completamente conectada tienen conexiones completas para todas las activaciones en la capa previa, tal y como se ha descrito anteriormente para una red predictiva. La salida de las capas 908 completamente conectadas se puede utilizar para generar un resultado de salida de la red. Las activaciones dentro de las capas 908 completamente conectadas se pueden calcular utilizando la multiplicación de matriz en lugar de la convolución. No todas las implementaciones de CNN hacen uso de las capas 908 completamente conectadas. Por ejemplo, en algunas implementaciones de la capa 906 convolucional puede generar una salida para la CNN.

Las capas convolucionales se conectan de manera dispersa, lo cual difiere de la configuración de red neuronal tradicional que se encuentra en las capas 908 completamente conectadas. Las capas de red neuronal tradicional están completamente conectadas, de manera que cada unidad de salida interactúe con cada unidad de entrada. Sin embargo, las capas convolucionales están conectadas de manera dispersa debido a que la salida de la convolución de un campo es entrada (en lugar del valor de estado respectivo de cada uno de los nodos en el campo) para los nodos de la capa posterior, tal y como se ilustra. Los *kernels* asociados con las capas convolucionales llevan a cabo operaciones de convolución, la salida de los cuales se envía a la capa siguiente. La reducción de dimensionalidad llevada a cabo dentro de las capas convolucionales es un aspecto que permite que la CNN avance para procesar imágenes grandes.

La Figura 9B ilustra etapas de cálculos de ejemplo dentro de una capa convolucional de una CNN. La entrada a una capa 912 convolucional de una CNN se puede procesar en tres etapas de una capa 914 convolucional. Las tres etapas pueden incluir una etapa 916 convolucional, una etapa 918 de detector, y una etapa 920 de reducción. La capa 914 de convolución puede entonces emitir datos a una capa convolucional sucesiva. La capa convolucional final de la red puede generar datos de mapa de características de salida o proporcionar entradas a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

La etapa 916 de convolución lleva a cabo varias convoluciones paralelas para producir un conjunto de activaciones lineales. La etapa 916 de convolución puede incluir una transformación afín, que es cualquier transformación que se puede especificar como una transformación lineal más una traducción. Las transformaciones afines incluyen rotaciones, traducciones, escalas y combinaciones de estas transformaciones. La etapa de convolución calcula la salida de funciones (p. ej., neuronas) que están conectadas a regiones específicas de la entrada, que se pueden determinar cómo 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 cual se conectan las neuronas. El resultado de la etapa 916 de convolución define un conjunto de activaciones lineales que se procesan en etapas sucesivas de la capa 914 convolucional.

Una etapa 918 de detector puede procesar las activaciones lineales. En la etapa 918 de detector, cada activación lineal es procesada por una función de activación no lineal. La función de activación no lineal aumenta las propiedades no lineales de la red total sin afectar los campos receptivos de la capa de convolución. Se pueden utilizar diversos tipos de funciones de activación no lineales. Un tipo específico es la unidad lineal rectificada (ReLU), que utiliza una función de activación definida como $f(x) = \text{máx}(0, x)$, de manera que la activación tiene el umbral en cero.

La etapa 920 de reducción utiliza una función de reducción que reemplaza la salida de la capa 906 convolucional con una estadística de resumen de las salidas cercanas. La función de reducción se puede utilizar para introducir invariancias en la traducción en la red neuronal, de manera que pequeñas traducciones en la entrada no cambien las salidas reducidas. Invariancias en la traducción local pueden ser útiles en entornos en los que la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Durante la etapa 920 de reducción se pueden utilizar diversos tipos de funciones de reducción, incluyendo reducción máxima, reducción promedio y reducción de 12 estándares. Además, algunas implementaciones de CNN no incluyen una etapa de reducción. En cambio, dichas implementaciones sustituyen una etapa de convolución adicional que presenta un segmento aumentado en relación con etapas de convolución previas.

La salida de la capa 914 convolucional puede entonces ser procesada por la capa 922 siguiente. La capa 922 siguiente puede ser una capa convolucional adicional o una de las capas 908 completamente conectadas. Por ejemplo, la primera capa 904 convolucional de la Figura 9A puede emitir hacia la segunda capa 906 convolucional, mientras que la segunda capa convolucional puede emitir hacia una primera capa de las capas 908 completamente conectadas.

La Figura 10 ilustra una red 1000 neuronal recurrente de ejemplo. En una red neuronal recurrente (RNN), el estado previo de la red influye a la salida del estado actual de la red. Las RNN pueden estar construidas de una variedad de formas utilizando una variedad de funciones. El uso de RNN generalmente gira entorno al uso de modelos matemáticos para predecir el futuro en base a una secuencia previa de entradas. Por ejemplo, una RNN se puede utilizar para llevar a cabo un modelado de lenguaje estadístico para predecir una palabra de próxima aparición dada una secuencia de palabras previa. La RNN 1000 ilustrada se puede describir con una capa 1002 de entrada que recibe un vector de entrada, capas 1004 ocultas para implementar una función recurrente, un mecanismo 1005 de retroalimentación para permitir una "memoria" de estados previos y una capa 1006 de salida para emitir un resultado. La RNN 1000 opera en base a etapas temporales. El estado de la RNN en una etapa temporal determinada está influenciado según la etapa

temporal previa a través del mecanismo 1005 de retroalimentación. Para una etapa temporal determinada, el estado de las capas 1004 ocultas está definido por el estado previo y la entrada en la etapa temporal actual. Una entrada inicial (x_1) en una primera etapa temporal puede ser procesada por la capa 1004 oculta. Una segunda entrada (x_2) puede ser procesada por la capa 1004 oculta utilizando una información de estado que se determina durante el procesamiento de la entrada (x_1) inicial. Un estado determinado puede computarse como $s_t = f(Ux_t + Ws_{t-1})$, donde U y W son matrices de parámetro. La función f es una no linealidad, tal y como una función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x)$ = máx(0, x). Sin embargo, la función matemática específica utilizada en las capas 1004 ocultas puede variar dependiendo de los detalles de implementación específicos de la RNN 1000.

Además de las redes CNN y RNN básicas descritas, se pueden permitir variaciones sobre esas redes. Un ejemplo de variante de RNN es la RNN de memoria de corto-largo plazo (LSTM). Las RNN de LSTM son capaces de aprender dependencias de largo plazo que pueden ser necesarias para procesar secuencias de lenguajes más largas. Una variante de la CNN es una red de creencias profundas convolucionales, que tiene una estructura similar a una CNN y se entrena de una manera similar a una red de creencia profunda. Una red de creencias profundas (DBN) es una red neuronal generativa que está compuesta de múltiples capas de variables (aleatorias) estocásticas. Las DBN pueden entrenarse capa por capa utilizando un aprendizaje ambicioso sin supervisión. Entonces, los pesos aprendidos de la DBN se pueden utilizar para proporcionar redes neuronales preentrenadas al determinar un conjunto de pesos óptimos inicial para la red neuronal.

La Figura 11 ilustra el entrenamiento y despliegue de una red neuronal profunda. Una vez que se ha estructurado una red determinada para una tarea la red neuronal se entrena utilizando un conjunto de datos 1102 de entrenamiento. Se han desarrollado varias estructuras 1104 de entrenamiento para permitir una aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura 604 de aprendizaje automático de la Figura 6 se puede configurar como una estructura 604 de entrenamiento. La estructura 604 de entrenamiento se puede enganchar a una red 1106 neuronal sin entrenar y permitir que la red neuronal sin entrenar se entrene utilizando los recursos de procesamiento paralelo descritos en la presente memoria para generar una red 1108 neuronal entrenada.

Para iniciar los procesos de entrenamiento los pesos iniciales se pueden seleccionar aleatoriamente o mediante pre-entrenamiento utilizando una red de creencias profundas. El ciclo de entrenamiento se lleva a cabo entonces de manera supervisada o no supervisada.

El aprendizaje supervisado es un método de aprendizaje en el que el entrenamiento se lleva a cabo como una operación mediada, como cuando el conjunto de datos 1102 de entrenamiento incluye una entrada emparejada con la salida deseada para la entrada, o donde el conjunto de datos de entrenamiento incluye una entrada que presenta una salida conocida y la salida de la red neuronal se clasifica manualmente. La red procesa las entradas y compara las salidas resultantes con un conjunto de salidas esperadas o deseadas. Los errores después se propagan de nuevo a través del sistema. La estructura 1104 de entrenamiento puede ajustarse para ajustar los pesos que controlan la red 1106 neuronal sin entrenar. La estructura 1104 de entrenamiento puede proporcionar herramientas para monitorizar qué tan bien la red 1106 neuronal sin entrenar está convergiendo hacia un modelo apropiado para generar respuestas correctas en base a datos de entrada conocidos. El proceso de entrenamiento ocurre de manera repetitiva a medida que los pesos de la red se ajustan para afinar la salida generada por la red neuronal. Los procesos de entrenamiento pueden continuar hasta que la red neuronal alcanza una precisión deseada en términos estadísticos asociada con una red 1108 neuronal entrenada. La red 1108 neuronal entrenada puede entonces desplegarse para implementar cualquier cantidad de operaciones de aprendizaje automático.

El aprendizaje sin supervisión es un método de aprendizaje en el que la red intenta entrenarse a sí misma utilizando datos sin etiquetar. Por lo tanto, para el aprendizaje sin supervisión, el conjunto 1102 de datos de entrenamiento incluirá datos de entrada sin ningún dato de salida asociado. La red 1106 neuronal sin entrenar puede aprender agrupamientos dentro de la entrada sin etiquetar y puede determinar cómo se relacionan las entradas individuales al conjunto de datos total. El entrenamiento sin supervisión se puede utilizar para generar un mapa de autoorganización, que es un tipo de red 1107 neuronal entrenada capaz de llevar a cabo operaciones útiles para reducir la dimensionalidad de datos. El entrenamiento sin supervisión también se puede utilizar para llevar a cabo detecciones de anomalías, lo que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvía de los patrones normales de los datos.

También se pueden utilizar variaciones de entrenamiento supervisado o sin supervisión. El aprendizaje semi supervisado es una técnica en la que el conjunto 1102 de datos de entrenamiento incluye una mezcla de datos etiquetados y sin etiquetar de la misma distribución. El aprendizaje incremental es una variante de aprendizaje supervisado en el que los datos de entrada se utilizan continuamente para entrenar aún más el modelo. El aprendizaje incremental permite que la red 1108 neuronal entrenada se adapte a los nuevos datos 1112 sin olvidar el conocimiento instilado dentro de la red durante el entrenamiento inicial.

Ya sea con supervisión o sin supervisión, el proceso de entrenamiento para redes neuronales particularmente profundas puede ser demasiado intensivo computacionalmente para un único nodo de cálculos. En lugar de utilizar un único nodo de cálculo, se puede utilizar una red distribuida de nodos computacionales para acelerar el proceso de entrenamiento.

La Figura 12 es un diagrama de bloque que ilustra un aprendizaje distribuido. El aprendizaje distribuido es un modelo de entrenamiento que utiliza múltiples nodos de cálculos distribuidos para llevar a cabo un entrenamiento supervisado o sin supervisión de una red neuronal. Cada uno de los nodos computacionales distribuidos pueden incluir uno o más procesadores anfitriones y uno o más de los nodos de procesamiento para fines generales, tales como la unidad 700 de procesamiento de gráficos para fines generales muy paralelos como en la Figura 700. Tal y como se ilustra, el aprendizaje distribuido se puede llevar a cabo como paralelismo 1202 modelo, paralelismo 1204 de datos o una combinación de paralelismo 1204 de modelo y datos.

En el paralelismo 1202 modelo, diferentes nodos computacionales en un sistema distribuido pueden llevar a cabo cálculos de entrenamiento para partes diferentes de una red única. Por ejemplo, un nodo de procesamiento diferente del sistema distribuido puede entrenar cada capa de una red neuronal. Los beneficios del paralelismo modelo incluyen la capacidad de avanzar a modelos particularmente grandes. Dividir los cálculos asociados con diferentes capas de la red neuronal permite el entrenamiento de redes neuronales muy grandes en las que los pesos de todas las capas no entrarían en la memoria de un único nodo computacional. En algunas instancias, el paralelismo modelo puede ser particularmente útil para llevar a cabo entrenamientos sin supervisión de grandes redes neuronales.

En el paralelismo 1204 de datos, 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 después se combinan. A pesar de que son posibles diferentes enfoques para el paralelismo de datos, los enfoques de entrenamiento paralelo de datos requieren todos una técnica que combine resultados y sincronización de los parámetros modelo entre cada nodo. Enfoques de ejemplo para datos combinados incluye promediar y actualizar parámetros en base al paralelismo de datos. La promediación de parámetros entrena cada nodo en un subconjunto de los datos de entrenamiento y configura los parámetros globales (p. ej., pesos, polarizaciones) para el promedio de los parámetros de cada nodo. La promediación de parámetros utiliza un servidor de parámetros central que mantiene los datos de parámetros. La actualización en base a paralelismo de datos es similar a la promediación de parámetros excepto en que en lugar de transferir parámetros de los nodos al servidor de parámetros, se transfieren las actualizaciones al modelo. Además, las actualizaciones en base al paralelismo de datos se pueden llevar a cabo de una manera descentralizada, donde las actualizaciones se comprimen y transfieren entre nodos.

El modelo combinado y paralelismo 1206 de datos se pueden implementar, por ejemplo, en un sistema distribuido en el que cada nodo computacional incluye múltiples GPU. Cada nodo puede presentar una instancia completa del modelo con GPU separadas dentro de cada nodo que se utilizan para entrenar diferentes porciones del modelo.

El entrenamiento distribuido tiene taras aumentadas en relación con el entrenamiento de una única máquina. Sin embargo, los procesadores paralelos y las GPGPU descritos en la presente memoria pueden cada uno implementar diversas técnicas para reducir la tara de entrenamiento distribuido, incluyendo técnicas para permitir transferencia de datos de GPU a GPU de ancho de banda alto y sincronización de datos remotos acelerada.

Aplicaciones de aprendizaje automático de ejemplo

El aprendizaje automático se puede aplicar para resolver una variedad de problemas tecnológicos, que incluye, pero no está limitado a visión artificial, conducción y navegación autónoma, reconocimiento de voz y procesamiento de lenguaje. La visión artificial ha sido, tradicionalmente, una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Aplicaciones de visión artificial que van desde reproducir capacidades visuales humanas, tal y como reconocer caras, hasta crear nuevas categorías de capacidades visuales. Por ejemplo, las aplicaciones de visión artificial se pueden configurar para reconocer ondas de sonido de las vibraciones inducidas en objetos visibles en un vídeo. El aprendizaje automático acelerado en procesador paralelo permite que se entrenen aplicaciones de visión artificial utilizando un conjunto de datos de entrenamiento significativamente más grande que lo que era posible antes y permite que se desplieguen sistemas de inferencia utilizando procesadores paralelos de baja potencia.

El aprendizaje automático acelerado en procesador paralelo tiene aplicaciones para la conducción autónoma que incluyen el reconocimiento de señales de tráfico, evitación de obstáculos, navegación y control de la conducción. Las técnicas de aprendizaje automático acelerado se pueden utilizar para entrenar modelos de conducción basados en conjuntos de datos que definen las respuestas adecuadas para entradas de entrenamientos específicos. Los procesadores paralelos descritos en la presente memoria pueden permitir un rápido entrenamiento de las redes neuronales cada vez más complejas que se utilizan en soluciones de conducción autónoma y permiten el despliegue de procesadores de inferencia de baja potencia en una plataforma móvil apropiada para integrarse en vehículos autónomos.

Las redes neuronales profundas aceleradas en procesadores paralelos han permitido aplicar enfoques del aprendizaje automático en el reconocimiento automático de voz (ASR,). El ASR incluye crear una función que calcula la secuencia lingüística más probable dada una secuencia acústica de entrada. El aprendizaje automático acelerado que utiliza redes neuronales profundas ha permitido que se sustituyan los modelos ocultos de Márkov (HMM) y los modelos de mezcla gaussiana (GMM) que anteriormente se utilizaban para el ASR.

El aprendizaje automático acelerado en procesador paralelo también se puede utilizar para acelerar el procesamiento del lenguaje natural. Los procedimientos de aprendizaje automático pueden utilizar algoritmos de inferencia estadística

para producir modelos que sean robustos ante entradas erróneas o desconocidas. Las aplicaciones de procesador de lenguaje natural de ejemplo incluyen la traducción automática entre lenguajes humanos.

Las plataformas de procesamiento paralelo utilizadas para el aprendizaje automático se pueden dividir en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento son generalmente muy paralelas e incluyen optimizaciones para acelerar el entrenamiento multi-GPU de nodo único y el entrenamiento multi-GPU de múltiples nodos. Los procesadores paralelos de ejemplo adecuados para el entrenamiento incluyen la unidad 700 de procesamiento de gráficos muy paralela para fines generales de la Figura 700 y el sistema 800 informático multi-GPU de la Figura 800. Por el contrario, las plataformas de aprendizaje automático desplegadas generalmente incluyen procesadores paralelos de menor potencia apropiados para utilizar en productos tal y como cámaras, robots autónomos y vehículos autónomos.

La Figura 13 ilustra un sistema de inferencia de ejemplo en un chip (SOC) 1300 apropiado para llevar a cabo inferencias que utilizan un modelo entrenado. El SOC 1300 puede integrar componentes de procesamiento, incluidos un procesador 1302 de medios, un procesador 1304 de visión, una GPGPU 1306 y un procesador 1308 multinúcleo. El SOC 1300 puede además incluir una memoria 1305 en chip que puede habilitar un grupo de datos compartido en chip que es accesible por cada uno de los componentes de procesamiento. Los componentes de procesamiento se pueden optimizar para que la operación de baja potencia permita el despliegue de una variedad de plataformas de aprendizaje automático, incluidos los vehículos autónomos y los robots autónomos. Por ejemplo, una implementación del SOC 1300 se puede utilizar como una porción del sistema de control principal para un vehículo autónomo. Donde el SOC 1300 está configurado para utilizarse en vehículos autónomos el SOC está diseñado y configurado para cumplir con las normas de seguridad funcional relevantes de la jurisdicción de despliegue.

Durante el funcionamiento, el procesador 1302 de medios y el procesador 1304 de visión pueden trabajar en conjunto para acelerar las operaciones de visión artificial. El procesador 1302 de medios puede permitir la descodificación de latencia baja de múltiples secuencias de vídeo de alta resolución (p. ej., 4K, 8K). Las secuencias de vídeo descodificadas se pueden escribir en una memoria intermedia en la memoria 1305 en chip. Entonces, el procesador 1304 de visión puede analizar el vídeo descodificado y llevar a cabo operaciones de procesamiento preliminares en las tramas del vídeo descodificado al preparar el procesamiento de las tramas utilizando un modelo de reconocimiento de imágenes entrenado. Por ejemplo, el procesador 1304 de visión pude acelerar las operaciones de convolución para una CNN que se utiliza para llevar a cabo el reconocimiento de imágenes en los datos de vídeo de alta resolución, mientras que la GPGPU 1306 lleva a cabo cálculos del modelo *back end*.

El procesador 1308 multinúcleo puede incluir lógica de control para ayudar a secuenciar y sincronizar las transferencias de datos y las operaciones de memoria compartida llevadas a cabo por el procesador 1302 de medios y el procesador 1304 de visión. El procesador 1308 multinúcleo puede también funcionar como un procesador de aplicación para ejecutar aplicaciones de software que pueden utilizar la capacidad de cálculo de inferencia de la GPGPU 1306. Por ejemplo, se puede implementar al menos una porción de la lógica de navegación y conducción en el software que se ejecuta en el procesador 1308 multinúcleo. Dicho software puede emitir directamente volúmenes de trabajo de cálculo a la GPGPU 1306 o los volúmenes de trabajo de cálculo se pueden emitir al procesador 1308 multinúcleo, que puede descargar al menos una porción de dichas operaciones a la GPGPU 1306.

La GPGPU 1306 puede incluir clústeres de cálculo tal y como una configuración de baja potencia de los clústeres de cálculo 706A-706H dentro de la unidad 700 de procesamiento de gráficos muy paralela para fines generales. Los clústeres de cálculo dentro de la GPGPU 1306 pueden admitir instrucciones que están específicamente optimizadas para realizar cálculos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1306 puede admitir instrucciones para llevar a cabo cálculos de baja precisión tal y como operaciones de vectores enteros de 8 bits y 4 bits.

Reducción de precisión de la unidad dinámica de coma flotante para operaciones de aprendizaje automático

El formato de coma flotante binario de precisión única del IEEE 754 especifica una representación binaria de 32 bits con una señal de 1 bit, un exponente de 8 bits y un significando de 24 bits, del cual se almacenan explícitamente 23 bits. El formato de coma flotante binario de precisión media del IEEE 754 especifica una representación binaria de 16 bits con una señal de 1 bit, un exponente de 5 bits y un significando de 11 bits, del cual se almacenan explícitamente 10 bits. Los bits de significando implícito se definen como uno para los valores de exponente distintos a cero y como cero cuando todos los bits de exponente son cero. Las unidades de coma flotante capaces de llevar a cabo operaciones aritméticas con precisión única y media son conocidas en la técnica. Por ejemplo, las unidades de coma flotante existentes pueden realizar operaciones (FP32) de coma flotante de precisión única de 32 bits u operaciones (FP16) de coma flotante de precisión media dobles de 16 bits.

Las realizaciones descritas en la presente memoria extienden dicha capacidad al proporcionar apoyo a las instrucciones y a la lógica asociada para permitir operaciones de precisión variables. Las instrucciones de coma flotante que permiten operaciones de precisión variables pueden aumentar dinámicamente el rendimiento al llevar a cabo operaciones con una precisión más baja cuando sea posible. En una realización se proporciona un conjunto de instrucciones y lógica asociada en la que se aumenta el rendimiento al realizar operaciones de coma flotante a la precisión más baja posible sin pérdidas significativas de datos. En una realización se proporciona un conjunto de

instrucciones y lógica asociada en la que la lógica de coma flotante verificará resultados de baja precisión contra resultados llevados a cabo con una precisión más alta para determinar si ha tenido lugar alguna pérdida de datos significativa.

La Figura 14 ilustra componentes de una unidad 1400 de coma flotante de precisión dinámica, según la invención. La unidad 1400 de coma flotante de precisión dinámica incluye una unidad 1402 de control, un conjunto de registros 1404 internos, un bloque 1406 de exponente y un bloque 1408 de significando. Además de la lógica de control de coma flotante conocida en la técnica, en una realización la unidad 1402 de control incluye además lógica 1412 de seguimiento de precisión y una unidad 1422 de transformación numérica.

La lógica 1412 de seguimiento de precisión es lógica de hardware configurada para llevar a cabo el seguimiento de un número disponible de bits de precisión para datos calculados en relación con una precisión diana. La lógica 1412 de seguimiento de precisión puede llevar a cabo el seguimiento de registros de precisión en el bloque 1406 de exponente y el bloque 1408 de significando para llevar a cabo el seguimiento de las métricas de precisión tal y como el número mínimo de bits de precisión requerido para almacenar valores calculados generados por el bloque 1406 de exponente y el bloque 1408 de significando. En una realización las métricas de precisión incluyen una media móvil de precisión numérica requerida para representar datos sobre un conjunto de cálculos. En una realización las métricas de precisión incluyen una precisión requerida máxima dentro de un conjunto de datos determinado. En una realización la unidad 1400 de coma flotante de precisión dinámica admite instrucciones para leer o reiniciar los datos registrados utilizados por la lógica 1412 de seguimiento de precisión para generar las métricas de precisión descritas en la presente memoria. En una realización la unidad de cálculo que alberga la unidad de coma flotante de precisión dinámica admite instrucciones para configurar o reiniciar los datos registrados utilizados por la lógica 1412 de seguimiento de precisión. En una realización la lógica 1412 de seguimiento de precisión monitoriza un acumulador 1434 de errores en el conjunto de registros 1404 internos. El acumulador de errores se puede utilizar para llevar a cabo el seguimiento de un error acumulado (p. ej., error de redondeo) sobre un conjunto de operaciones de coma flotante. En una realización la unidad 1400 de coma flotante de precisión dinámica admite un conjunto de instrucciones que incluyen una instrucción para reiniciar el acumulador 1434 de errores y una instrucción para leer el acumulador 1434 de errores. En una realización el acumulador de errores se puede reiniciar en respuesta a un bit o una bandera suministrada como un operando para una instrucción.

En una realización la unidad 1422 de transformación numérica se puede utilizar para realizar transformaciones numéricas inmediatas en datos cuando se llevan a cabo operaciones de precisión más baja para evitar o mitigar la posibilidad de desbordamiento o subdesbordamiento mientras se llevan a cabo operaciones. Por ejemplo, cuando se aproxima a los límites de precisión de un tipo de datos determinado, la unidad 1422 de transformación numérica puede llevar a cabo operaciones de multiplicación o división al utilizar logaritmos y puede transformar el valor resultante a través de la exponenciación. Se proporcionan detalles adicionales con respecto a la lógica 1412 de seguimiento de precisión y a la unidad 1422 de transformación numérica en la Figura 22.

Los registros 1404 internos incluyen un conjunto de registros 1414 de operandos que almacenan valores de entrada para la unidad 1400 de coma flotante de precisión dinámica. En una realización los registros 1414 de operandos incluyen dos operandos (A, B). Para los datos de entrada de coma flotante, los valores de datos de entrada se pueden dividir en porciones de exponente (EXA, EXB) y porciones de significando (SIGA, SIGB). En diversas realizaciones los registros 1414 de operandos no se limitan a admitir dos entradas de coma flotante. En una realización los registros 1414 de operandos incluyen tres operandos de entrada, por ejemplo, para admitir la multiplicación-adición fusionada, la multiplicación-resta, la multiplicación-acumulación u operaciones relacionadas. En una realización los registros 1414 de operandos pueden también almacenar valores enteros, así como en una realización la unidad de coma flotante de precisión dinámica admite operaciones enteras de 32 bits, 16 bits y 8 bits. La precisión de tipo de datos específicos y de línea de base se puede configurar, en una realización, mediante una entrada a la unidad 1402 de control.

Las operaciones de coma flotante se realizan con precisión dinámica al utilizar el bloque 1406 de exponente y el bloque 1408 de significando. En una realización, las operaciones enteras se pueden llevar a cabo mediante el bloque 1408 de significando. En una realización, las operaciones enteras de 8 bits dobles se pueden llevar a cabo utilizando el bloque 1406 de exponente y el bloque 1408 de significando.

En una realización el bloque 1406 de exponente incluye un comparador 1416 y un sumador 1426 de exponente de precisión dinámica. El comparador determina la diferencia entre exponentes y determina el exponente más pequeño de los dos. Durante la adición de coma flotante, el exponente del número más pequeño se ajusta para coincidir con el exponente del número más grande. El sumador 1426 de exponente de precisión dinámica se puede utilizar para la adición de valores de exponente para valores de FP16 o FP32. El bloque 1408 de significando incluye un multiplicador 1418 de precisión dinámica, una unidad 1428 de desplazamiento, un sumador 1438 de significando de precisión dinámica y un registro 1448 de acumuladores.

En una realización se puede especificar un tipo de datos de FP16 o FP32 para una operación. Donde se especifica FP16, la unidad 1400 de coma flotante de precisión dinámica puede activar elementos de control que son innecesarios para llevar a cabo operaciones FP32 mientras que se mantiene la lógica para llevar a cabo el seguimiento de errores o pérdida de precisión (p. ej., mediante un acumulador 1434 de errores). Por ejemplo y en una realización, el acumulador 1434 de errores se puede utilizar para llevar a cabo el seguimiento de un número de operaciones de

redondeo dentro de un período de instrucciones. En una realización el acumulador de errores mantiene un valor de los errores de redondeo acumulados totales sobre un conjunto de instrucciones. La unidad 1400 de coma flotante de precisión dinámica puede permitir la admisión de una instrucción para despejar o leer el acumulador 1434 de errores del software. Donde se especifica FP32, la unidad 1400 de coma flotante de precisión dinámica puede tratar de realizar operaciones de FP32 con una precisión FP16, mientras que activa elementos de control y componentes más allá de los requeridos para realizar operaciones con una precisión FP16. En base a la entrada o a los valores intermedios, donde se solicita que la unidad 1400 de coma flotante de precisión dinámica realice operaciones con una FP32, la unidad 1400 de coma flotante de precisión dinámica puede inicialmente tratar de realizar operaciones a una FP16 y expandir la precisión lo que se necesite hasta una FP32. Donde se pueden realizar operaciones de FP32 con una precisión FP16, se reduce la demanda de potencia por operación, lo que permite que se habilite simultáneamente una gran cantidad de elementos de cálculo. Por ejemplo, la capacitancia dinámica y/o las limitaciones de presupuesto de potencia para una configuración dada, tal y como una configuración con batería o una configuración de enfriamiento solo pasivo, pueden no permitir que todas las unidades de coma flotante u otros elementos de cálculo dentro de una GPGPU se habiliten simultáneamente. Reducir la potencia dinámica de un conjunto de unidades de coma flotante al habilitar el cálculo de menor precisión dinámica puede aumentar el rendimiento general de las unidades de cálculo de una GPGPU dentro de un envolvente de potencia dado, y se puede procesar un mayor número de hilos según cada ciclo sin exceder limitaciones de potencia dinámica.

La Figura 15 proporciona detalles adicionales sobre la unidad 1400 de coma flotante de precisión dinámica de la Figura 14, según una realización. En una realización el multiplicador 1418 de precisión dinámica incluye un conjunto de memorias intermedias 1502 de entrada para almacenar datos de significandos. En una realización el conjunto de memorias intermedias de entrada incluye dos memorias intermedias para almacenar dos valores de entrada para una operación de multiplicación o división. Para una operación fusionada (p. ej., multiplicación-adición, multiplicación-resta) el producto de la operación se puede añadir a una tercera entrada mediante un sumador y/o almacenar en un registro de acumuladores.

En una realización algunas configuraciones del multiplicador 1418 de precisión dinámica incluyen memorias intermedias de entrada que son entradas de 24 bits que pueden almacenar explícitamente 24 bits de datos de significando para entradas de coma flotante de precisión única o datos de significando de 11 bits para valores de coma flotante de precisión media. En algunas configuraciones las memorias intermedias 1502 de entrada pueden también ser memorias intermedias de 32 bits para permitir la multiplicación de valores enteros de 32 bits. En una realización está presente una única configuración de las memorias intermedias 1502 de entrada que es seleccionable o configurable entre 32 bits y 24 bits. En una realización la memoria intermedia 1510 de salida es configurable o seleccionable de forma similar entre 24 bits y 32 bits para permitir de forma selectiva el almacenamiento de un entero de 32 bits de precisión completa o un valor de significando de 24 bits y/o de 11 bits para un número de coma flotante de 32 bits o 16 bits.

En una realización el multiplicador 1418 de precisión dinámica incluye un multiplicador 1506 y un multiplicador 1504 de desbordamiento. El multiplicador 1506 es configurable para realizar una operación de multiplicación o división con una precisión media para un tipo de datos. Por ejemplo, el multiplicador 1506 puede realizar una operación de multiplicación de 11 bits para el significando de un valor de coma flotante de FP16 y/o una operación de multiplicación de 16 bits para una operación entera de 16 bits. El multiplicador 1506 puede también realizar una operación de multiplicación de 8 bits para un valor entero de INT8. Para un valor de coma flotante de 32 bits o un valor entero de 32 bits, el multiplicador 1506 puede realizar una operación de multiplicación para un significando de 24 bits a 11 bits (p. ej., una precisión FP16). El multiplicador 1506 puede, en caso necesario, llevar a cabo un valor de multiplicación de 16 bits de precisión de significando para un significando de FP16 de 24 bits. En una realización se puede llevar a cabo el seguimiento de la precisión requerida y resultante para una operación en un conjunto dado de entradas mediante el registro 1508 de precisión. En una realización la precisión solicitada y resultante se puede representar en el registro 1508 de precisión mediante una pérdida de precisión que resultaría si la respuesta del multiplicador 1506 se emite mediante la memoria intermedia 1510 de salida. En dicha realización el registro 1508 de precisión puede llevar a cabo el seguimiento de la pérdida de precisión asociada con el uso de tipos de datos de menor precisión así como la pérdida de precisión asociada con la ejemplo de operaciones con una precisión menor de la solicitada.

En una realización la lógica de control asociada con el multiplicador 1418 de precisión dinámica (p. ej., en la unidad 1402 de control de la Figura 14), puede monitorizar la pérdida de precisión asociada con la ejemplo de operaciones para operaciones de una mayor precisión (p. ej., FP32, INT32) con una menor precisión (p. ej., FP16, INT16, INT8). Si la pérdida de precisión es significativa la lógica de control puede permitir que el multiplicador 1504 de desbordamiento realice operaciones para los bits adicionales de precisión. Adicionalmente, si la lógica de control determina que ocurrirá un desbordamiento o un subdesbordamiento en base a las entradas actuales, el multiplicador 1504 de desbordamiento se habilita y se realiza la operación de multiplicación al utilizar el multiplicador 1504 de desbordamiento y el multiplicador 1506.

Se realizan operaciones de control similares para el sumador 1426 de exponente de precisión dinámica y el sumador 1438 de significando de precisión dinámica. El sumador 1426 de exponente de precisión dinámica incluye un conjunto de memorias intermedias de entrada de 8 bits que puede almacenar datos de exponente para una FP32 (8 bits) y FP16 (5 bits). Las memorias intermedias 1512 de entrada de 8 bits pueden también almacenar un conjunto de entradas de INT-8. Se puede configurar de manera similar la memoria intermedia 1520 de salida para el sumador 1426 de

exponente de precisión dinámica. El sumador 1438 de significando de precisión dinámica incluye un conjunto de memorias intermedias 1522 de entrada que se puede seleccionar de una de un conjunto de memorias intermedias de 24 bits y 32 bits o se puede configurar dinámicamente para almacenar datos de entrada de o bien 24 bits o 32 bits. En una realización las memorias intermedias 1522 de entrada son simplemente memorias intermedias de 32 bits que también pueden almacenar datos de entrada de 24 bits. Se puede configurar de manera similar la memoria intermedia 1530 de salida para el sumador 1438 de significando de precisión dinámica. El registro 1518 de precisión en el sumador 1426 de exponente de precisión dinámica y el registro 1528 de precisión en el sumador 1438 de significando de precisión dinámica se pueden configurar para llevar a cabo el seguimiento de la pérdida de precisión para las operaciones realizadas. La lógica de control puede habilitar el sumador 1514 de desbordamiento y/o el sumador 1524 de desbordamiento según se necesite para evitar condiciones de desbordamiento o subdesbordamiento o para evitar que la pérdida de precisión exceda un umbral.

Volviendo a la Figura 14, en una realización, la unidad 1400 de coma flotante de precisión dinámica puede realizar las operaciones de INT8 dobles al utilizar el sumador 1426 de exponente de precisión dinámica y el sumador 1438 de significando de precisión dinámica. Por ejemplo, en lugar de deshabilitar el bloque 1406 de exponente durante operaciones enteras, el bloque 1406 de exponente se puede configurar para realizar una operación en un primer conjunto de operandos enteros de 8 bits mientras que el bloque 1408 de significando se puede configurar para realizar una operación en un segundo conjunto de operandos de 8 bits. Para permitir la admisión de la multiplicación doble de 8 bits, la multiplicación-adición fusionada doble, la multiplicación-resta fusionada doble y/u otras operaciones basadas en multiplicaciones, en una realización el bloque 1406 de exponente puede incluir un multiplicador 1436 adicional. El multiplicador puede ser un multiplicador de 8 bits fijo para permitir operaciones de multiplicación de 8 bits dobles simultáneas utilizando el bloque 1406 de exponente y el bloque 1408 de significando.

La Figura 16 ilustra asignaciones de hilos para un sistema 1600 de procesamiento de precisión dinámica, según una realización. En una realización el sistema 1600 de procesamiento de precisión dinámica incluye un conjunto de unidades 1608A-1608D de coma flotante dinámicas. Las unidades 1608A-1608D de coma flotante dinámicas pueden ejecutar un conjunto de hilos 1606A-1606D de operaciones que pueden realizar operaciones de precisión mixtas y generar datos de salida con precisiones variables. En una realización un primer hilo 1606A de operaciones puede realizar una primera operación (p. ej., adición, resta, multiplicación, división, etc.) en una primera unidad 1608A de coma flotante dinámica, donde el primer hilo 1606A de operaciones acepta como entrada dos valores 1602A-1602B de coma flotante de 16 bits y emite un valor FP16 de coma flotante de 16 bits. La primera operación se puede llevar a cabo como una operación doble en la que una única instrucción ejecutada por una GPGPU permite una operación doble de precisión FP16/FP32 mixta. La segunda operación de la operación doble la puede llevar a cabo un segundo hilo 1606B de operaciones que es llevado a cabo por una segunda unidad 1608B de coma flotante dinámica, lo que puede generar una segunda salida 1612 que es una salida de coma flotante de 32 bits. El segundo hilo 1606B de operaciones configura la segunda unidad 1608B de coma flotante dinámica para recibir dos valores 1603A-1603B de entrada de coma flotante de 32 bits. En una realización la operación en las dos operaciones de coma flotante de 32 bits se puede llevar a cabo a 16 bits de precisión si la operación se puede llevar a cabo sin subdesbordamiento, desbordamiento, o pérdida de precisión excesiva no tendrá lugar al llevar a cabo la operación con una precisión menor.

En una realización el sistema 1600 de procesamiento de precisión dinámica puede ejecutar una única instrucción con un operando 1604A de 16 bits y un operando 1604B de 32 bits. El hilo 1606C de operaciones se puede ejecutar en una unidad 1608C de coma flotante dinámica. La unidad 1608C de coma flotante dinámica tratará de realizar una operación de 16 o 32 bits de precisión mixta con una precisión de 16 bits a menos que tenga lugar una pérdida de precisión significativa o un error. En una realización el sistema 1600 de procesamiento de precisión dinámica puede estar también configurado para realizar operaciones enteras. Por ejemplo, una operación en un par de entradas 1605A-1605B enteras de 8 bits se puede realizar mediante un hilo 1606D de operaciones a través de una unidad 1608D de coma flotante dinámica para generar una salida 1616 entera de 8 bits. En una realización la unidad 1608D de coma flotante dinámica se puede configurar para llevar a cabo operaciones enteras de 8 bits dobles en las que dos operaciones enteras de 8 bits se pueden realizar en un único ciclo.

La Figura 17 ilustra una lógica 1700 para llevar a cabo una operación numérica con menos de una precisión solicitada, según una realización. En una realización la lógica 1700 se implementa a través del hardware integrado en la unidad 1400 de coma flotante de precisión dinámica de la Figura 14. En una realización la lógica 1700 se lleva a cabo en parte a través de la unidad 1402 de control en la unidad 1400 de coma flotante de precisión dinámica de la Figura 14.

En una realización la lógica 1700 puede recibir una solicitud para llevar a cabo una operación numérica con una primera precisión, tal y como se muestra en el bloque 1702. La operación numérica puede ser una operación de coma flotante o una operación entera. La primera precisión puede ser, por ejemplo, una precisión de 32 bits. En una realización la operación numérica puede ser una operación en la primera precisión que se realiza después de otras operaciones con precisión mixta. Entonces, la lógica 1700 puede llevar a cabo la operación numérica al utilizar un número de bits asociado con una segunda precisión que es menor que la primera precisión, tal y como se muestra en el bloque 1704. Por ejemplo y en una realización el número de bits utilizado para realizar la operación puede ser un número de bits asociado con una operación de 16 bits, mientras que la primera precisión es una precisión de 32 bits. La lógica 1700 puede generar un resultado intermedio con la segunda precisión en el bloque 1706. Entonces, la lógica 1700 puede determinar una pérdida de precisión del resultado intermedio con respecto a la primera precisión. La

pérdida de precisión se puede leer desde un registro que almacena un indicador de pérdida de precisión que se almacena durante la operación.

La lógica 1700 puede determinar si la pérdida de precisión es menor que un umbral en el bloque 1709. En una realización el umbral asociado con la pérdida de precisión se puede configurar a través del software, a pesar de que en algunas realizaciones se utiliza un umbral de hardware por defecto. En una realización el grado de pérdida de precisión también se puede determinar al ejecutar operaciones de precisión completa en paralelo en unidades de cálculo no utilizadas. Entonces, los resultados de precisión reducidos se pueden comparar con los resultados de precisión completa. Si la pérdida de precisión es menor que el umbral, la lógica 1700 puede emitir el resultado con la segunda precisión, tal y como se muestra en el bloque 1712. Si la perdida de precisión no es menor que el umbral en el bloque 1709, la lógica 1700 puede calcular los bits restantes del resultado en el bloque 1710 y emitir el resultado con la primera precisión, tal y como se muestra en el bloque 1714. En una realización se pueden calcular los bits restantes del resultado en el bloque 1710 mediante unidades lógicas de desbordamiento, tal y como el multiplicador 1504 de desbordamiento, el sumador 1514 de desbordamiento y/o el sumador 1524 de desbordamiento, tal y como en la Figura 15.

Apilado vertical de operaciones para operaciones de coma flotante de 16 bits

Cuando se llevan a cabo operaciones de instrucción única para múltiples hilos (SIMT) con una precisión menor, en algunas circunstancias puede resultar difícil mantener pleno uso de lógica de instrucción única y flujo de datos múltiple (SIMD) subyacente debido al mayor número de elementos requeridos para llenar todos los carriles SIMD. Por ejemplo, una unidad lógica SIMD configurada para operaciones de FP32 en registros de entrada de 128 bits puede realizar una única operación en cuatro conjuntos de datos de entrada. Si dicha unidad lógica está configurada para realizar operaciones de FP16 en los mismos cuatro conjuntos de datos de entrada, el rendimiento subyacente para las operaciones se puede ver aumentado debido a la menor precisión de la operación, pero el uso de SIMD se reduce a la mitad. Una solución para la subutilización de SIMD es realizar la operación en ocho conjuntos de datos de entrada. Sin embargo, el software que se ejecuta en las unidades lógicas puede no requerir tanto paralelismo como el que puede proporcionar el hardware subyacente.

Por ejemplo, un bucle que realiza operaciones iterativas en disposiciones de entrada se puede vectorizar de modo que cada iteración de la disposición se lleva a cabo en paralelo como un hilo SIMT separado. Los hilos SIMT separados se pueden llevar a cabo en una única operación en lógica SIMD/vector subyacente en una unidad de cálculo. Cuando se llevan a cabo instrucciones paralelas derivadas de lógica de vectorización en bucle de compilación, un bucle más corto que ocho iteraciones no llenará los ocho carriles SIMD disponibles para ejecutar los hilos generados para dichas operaciones, reduciendo el uso general de las unidades de cálculo. Adicionalmente, donde el hardware subyacente tiene *N* carriles SIMD, cualquier número de iteraciones vectorizadas que no sea múltiplo de *N* requerirá que se ejecuten las iteraciones restantes en una unidad SIMD menos que llena. Además, la vectorización puede requerir la ejecución separada de bucles desenroscados antes de ejecutar el cuerpo principal de las operaciones vectorizadas.

Algunos ejemplos descritos en la presente memoria pueden aumentar el uso de SIMD al apilar múltiples operaciones de FP16 no relacionadas en una única unidad de SIMD para ejecutarla. Donde una unidad SIMD tiene 8 carriles disponibles para su ejecución, la lógica de programación de hilos puede despachar hilos en unidades de *N*/2 o *N*/4, y permitir que los conjuntos de hilos no relacionados que van a llevar a cabo las mismas operaciones (o similares) compartan una única unidad SIMD. Adicionalmente, un ejemplo habilita la planificación de carriles SIMD que permite que los grupos de hilos SIMT ensamblados dinámicamente se entremezclen con los hilos SIMD de vector.

La Figura 18 ilustra una vectorización en bucle para unidades SIMD, según un ejemplo. En un ejemplo, la lógica de software puede incluir un bucle que se vectoriza automáticamente al ejecutar un software de compilación en un sistema de procesamiento de datos. El bucle puede incluir un bucle 1802 desenroscado, un bucle 1804 principal vectorizado y un bucle 1806 restante. En algunas configuraciones la vectorización de bucles es más eficiente cuando se realiza en datos que tienen acceso a la memoria alineada. Por ejemplo, una GPGPU puede estar configurada de modo que los accesos a la memoria de vector se pueden llevar a cabo de forma más eficiente en fragmentos 1801A-1801F de 64 bytes. En dicha configuración, un bucle 1802 desenroscado incluye un subconjunto de iteraciones de bucle que se desenroscan del bucle principal para permitir que los accesos a la memoria no alineada se separen del bucle principal. El bucle 1804 principal vectorizado incluye la mayoría de las iteraciones del bucle. Cada iteración del bucle principal vectorizado se puede llevar a cabo en paralelo y los accesos a la memoria para cada elemento se alinean en un límite de la memoria específico. El bucle 1806 restante incluye el conjunto de iteraciones que siguen al bucle 1804 principal vectorizado. Generalmente, las iteraciones en el bucle 1806 restante pueden no llevarse a cabo en paralelo de una forma tan eficiente como en el bucle principal.

En un ejemplo el bucle 1802 desenroscado y el bucle 1806 restante también se pueden vectorizar. En un ejemplo cada uno del bucle 1802 desenroscado, el bucle 1804 principal y el bucle 1806 restante se pueden ejecutar en unidades SIMD8 de FP16, donde se pueden llevar a cabo ocho instancias de la misma operación en paralelo. Las iteraciones de bucle se pueden ejecutar en paralelo en hardware SIMD (p. ej., unidades 1801A-1808C SIMD8 de FP16) al utilizar la máscara 1812 de ejecución, la máscara 1814 de ejecución y la máscara 1816 de ejecución, cada una de las cuales habilita y deshabilita los carriles SIMD para un ciclo de operaciones. Para el bucle 1802 desenroscado ilustrado y el bucle 1806 restante, se selecciona un subconjunto de elementos en la máscara 1812 de

ejecución y en la máscara 1816 de ejecución. Todos los carriles se seleccionan en la máscara 1814 de ejecución del bucle 1804 principal vectorizado.

En un ejemplo, una unidad SIMD con carriles inactivos se puede configurar para realizar otras operaciones en dichos carriles inactivos. Para un ciclo dado, donde la lógica de planificación configura un conjunto de carriles inactivos para una unidad SIMD (p. ej., SIMD8 1808A de FP16, SIMD8 108C de FP16), en lugar de dejar en reposo dichos carriles durante un ciclo, el planificador puede agrupar otros hilos SIMD multielemento o asignar hilos SIMT a los carriles SIMD de lo contrario en reposo.

La Figura 19 ilustra un sistema 1900 de procesamiento de hilos, según un ejemplo. En un ejemplo el sistema 1900 de procesamiento de hilos incluye una unidad de cálculo SIMD, tal y como una unidad 1920 de coma flotante SIMD8 que incluye múltiples unidades 1922A-1922H de coma flotante dinámicas. Dependiendo de la operación, la unidad 1920 de coma flotante SIMD8 puede ejecutar ocho o más de las mismas operaciones (o similares) en un único ciclo. Por ejemplo y en un ejemplo, cada una de las unidades 1922A-1922H de coma flotante dinámicas puede ejecutar una única operación con una precisión FP16. En un ejemplo, cada una de las ocho unidades 1922A-1922H de coma flotante dinámicas puede realizar dos operaciones INT8 emparejadas en un único ciclo.

En algunas circunstancias, tal y como con bucles desenroscados o restantes tal y como se ilustra en la Figura 18, no todos los carriles de una unidad de coma flotante SIMD estarán activos durante un ciclo. Para aumentar el uso, se pueden asignar ranuras SIMD a granularidades menores, para permitir que se utilicen los carriles SIMD de lo contrario inutilizados. Por ejemplo, a la unidad 1920 de coma flotante SIMD8 se le asignarían generalmente hilos u operaciones a una granularidad de ocho operaciones, donde menos de ocho operaciones presentan una pérdida potencial de eficiencia computacional. En un ejemplo, los carriles SIMD pueden ser ocupados por un único hilo SIMD de vectores que incluye una máscara de ejecución que selecciona al menos ocho elementos o un grupo de hilos SIMT con al menos ocho elementos.

Para aumentar el uso de SIMD, un ejemplo divide ocho carriles SIMD en dos ranuras SIMD4 (p. ej., ranura 1910 SIMD4, ranura 1912 SIMD4). Las ranuras SIMD4 se pueden llenar de distintas maneras. En un ejemplo, dos hilos SIMD separados (hilo 1902 SIMD, hilo 1904 SIMD) que se combinan para cubrir un total de cuatro carriles SIMD se asignan a una ranura SIMD4 (p. ej., ranura 1910 SIMD4). En un ejemplo, el grupo 1906 de hilos SIMT se puede asignar a una ranura 1912 SIMD4. El grupo 1906 de hilos SIMT puede incluir cualquier número de hilos que sea múltiplo de cuatro hilos (p. ej., 4, 8, 12, 16, etc.). Con respecto a los hilos en el grupo 1906 de hilos SIMT, se pueden procesar cuatro hilos cada vez, con el número de ciclos requeridos para procesar todos los hilos en el grupo 1906 de hilos SIMT dependiendo del número de hilos en el grupo.

La Figura 20 ilustra una lógica 2000 para asignar hilos para cálculos, según un ejemplo. En un ejemplo la lógica 2000 se lleva a cabo mediante un sistema 1900 de procesamiento de hilos tal y como en la Figura 19. En un ejemplo la lógica 2000 puede recibir un primer conjunto de hilos en una unidad SIMD con un primer número de carriles, tal y como se muestra en el bloque 2002. Entonces, la lógica 2000 puede determinar si el primer conjunto de hilos llena todos los carriles SIMD de la unidad SIMD, tal y como se muestra en el bloque 2003. Si el primer conjunto de hilos incluye suficientes hilos SIMT o los hilos del primer conjunto de hilos incluyen suficientes elementos de vector SIMD para llenar todos los carriles SIMD, la lógica 2000 puede asignar el primer conjunto de hilos a la unidad SIMD, tal y como se muestra en el bloque 2004.

Si el primer conjunto de hilos no llena todos los carriles SIMD, tal y como se determina en el bloque 2003, la lógica 2000 puede asignar un primer conjunto de hilos a un segundo número de carriles que es menor que el primer número de carriles en el bloque 2006. La asignación se puede llevar a cabo al asignar un hilo SIMD a la unidad SIMD y enmascarar los carriles inactivos. La asignación también se puede llevar a cabo al asignar un conjunto de hilos SIMT a la unidad SIMD. Entonces, la lógica puede apilar uno o más conjuntos adicionales de hilos para llenar todos los carriles SIMD, tal y como se muestra en el bloque 2008. El conjunto adicional de hilos puede especificar carriles SIMD activos que ocupan carriles que no están ocupados por los hilos iniciales.

Sistema para habilitar la normalización y transformaciones para datos de baja precisión

Cuando se llevan a cabo operaciones con tipos de datos de baja precisión, se ha de tener cuidado para evitar el desbordamiento o subdesbordamiento de datos durante operaciones numéricas. Esta responsabilidad suele corresponder al científico de datos que está desarrollando el algoritmo de baja precisión. Debido a las limitaciones de la aritmética de baja precisión, muchas redes neuronales se han adaptado para utilizar valores binarios y/o ternarios que únicamente ocupan uno o dos bits por elemento. Sin embargo, existe una necesidad de unidades de lógica aritmética de coma flotante y enteras que pueden habilitar la aritmética de baja precisión de N bits con lógica de guarda para advertir o tratar de evitar una pérdida significativa de precisión durante operaciones aritméticas. En un ejemplo las unidades de coma flotante de precisión dinámica descritas en la presente memoria incluyen lógica para advertir cuando los cálculos numéricos se acercan a los límites de cálculos de baja precisión.

Tal y como se muestra en la Figura 14, una unidad 1400 de coma flotante de precisión dinámica puede incluir lógica 1412 de seguimiento de precisión y una unidad 1422 de transformación numérica. En una realización la lógica 1412 de seguimiento de precisión lleva a cabo el seguimiento de los bits disponibles de precisión restantes para datos

calculados en relación con una precisión diana. Si se puede llevar a cabo el seguimiento de la precisión para datos intermedios para determinar si un valor de datos intermedios, el cual en una realización se calcula con una precisión más alta en relación con datos de entrada o datos de salida, los bits disponibles se pueden almacenar con una precisión de salida sin pérdida significativa de precisión o error de redondeo. Por ejemplo y en una realización, las operaciones de precisión baja específicas se pueden llevar a cabo de forma eficiente con una precisión más alta y la lógica 1412 de seguimiento de precisión puede determinar si un resultado de un cálculo desbordaría una precisión de salida dada. En una realización las unidades lógicas descritas en la presente memoria pueden emitir información de estado que indique el grado de precisión perdida debido a un error de redondeo. En una realización las unidades lógicas pueden realizar transformaciones numéricas intermedias en datos para evitar la pérdida significativa de datos. Entonces, las unidades lógicas pueden emitir el valor transformado. En una realización un valor de salida de precisión completa o casi completa se puede derivar de manera programada en base a la salida y a la información de estado proporcionada con la salida.

La Figura 21 ilustra una red 2100 neuronal profunda que se puede procesar utilizando una lógica de cálculo proporcionada por los ejemplos descritos en la presente memoria. La red neuronal profunda (DNN) es una red neuronal artificial que incluye múltiples capas 2102A-2102N de red neuronales. Cada capa representa un conjunto de operaciones de cálculo no lineales para llevar a cabo la extracción y transformación de características de una manera coherente con las redes neuronales de aprendizaje automático descritas en la presente memoria. Cada capa sucesiva utiliza la salida de la capa anterior como entrada. En el caso de una red neuronal convolucional, la lógica de multiplicación-adición fusionada (p. ej., la lógica 2104A, 2104B FMA) se puede utilizar para calcular productos escalares entre un mapa de características y datos de filtro para generar datos de mapa de activación que se proporcionan como entrada para una capa sucesiva.

Las redes neuronales de baja precisión se pueden implementar utilizando pesos binarios o ternarios en combinación con mapas de características binarios, ternarios o de N bits. Algunas redes neuronales pueden incluso beneficiarse de la precisión añadida de cálculos que utilizan mapas de características de N bits y filtros de N bits. En algunas implementaciones, las características y pesos de N bits para una red neuronal se pueden procesar con una precisión baja sin una reducción significativa en el error de salida. Sin embargo, un científico de datos que implemente una red neuronal de N bits de baja precisión (p. ej., FP16, INT8) generalmente debe ser consciente de errores de redondeo o de datos fuera de límites que pueden surgir debido a cálculos sucesivos a baja precisión. Si la lógica de seguimiento de precisión (p. ej. la lógica 1412 de seguimiento de precisión de la Figura 14) en la lógica 2104A-2106B FMA determina que los datos de peso o de mapa de características se acercan a los límites de la precisión disponible del tipo de datos, la lógica 2104A-2015B FMA puede configurar un bit de estado. El bit de estado puede servir de indicador para un científico de datos que esté desarrollando el modelo de red neuronal que está presente en las capas 2102A-2012N de red neuronal de que el modelo pueda requerir optimización o de que se pueda garantizar una precisión numérica mayor.

En un ejemplo se puede habilitar la lógica 2106A-2106B de transformación y normalización para llevar a cabo la normalización de peso o transformaciones numéricas en datos de mapa de características antes de proporcionar los datos de mapa de características a la siguiente capa de red neuronal como entrada. La aplicación de la lógica 2106A-2106B de transformación y normalización es opcional en cada etapa y se puede llevar a cabo únicamente si las condiciones de pérdida de precisión significativa, desbordamiento o subdesbordamiento son probables durante el procesamiento de una próxima capa. En un ejemplo los pesos o los mapas de características que se emiten de una capa de una red neuronal se pueden normalizar automáticamente mediante una instancia de la lógica 2106A-2106B de transformación y normalización.

En un ejemplo la lógica 2106A-2016B de transformación y normalización puede utilizar la unidad 1422 de transformación numérica de la Figura 14 para transformar datos de mapa de características o datos de peso. Los datos de mapa de características que se emiten de una capa neuronal pueden estar basados en un conjunto de salida de datos de un conjunto de funciones. En dicha ejemplo se proporciona un conjunto específico de instrucciones de baja precisión que permiten que el ajuste automático de los datos de red neuronal de N bits evite la pérdida catastrófica de precisión. Las transformaciones o normalizaciones de ejemplo que la lógica 2106A-2016B de transformación y normalización puede llevar a cabo incluyen la normalización de peso a un rango de valores o un conjunto de transformación de datos de características persistente y reversible. En un ejemplo, la normalización de peso se puede llevar a cabo para comprimir el rango dinámico de un conjunto de pesos de filtro a dentro de un rango predeterminado. Los datos de peso se pueden normalizar, por ejemplo, en un rango de [-1,1], el cual puede conservar las diferencias relativas entre valores de peso mientras que reduce la magnitud general de los valores de peso. En un ejemplo, los datos de peso o de mapas de características de redes neuronales se pueden normalizar mediante el valor medio del conjunto de datos.

En un ejemplo, los cálculos de redes neuronales que utilizan datos que se acercan a los límites del rango del tipo de datos se pueden transformar antes de que se utilicen los datos en cálculos. Por ejemplo, una operación de multiplicación que utilice grandes valores que pueden resultar en un desbordamiento se puede llevar a cabo como una adición de logaritmos en lugar de una operación de multiplicación. A pesar de que dichas transformaciones pueden resultar en cierto grado de pérdida de precisión, los cálculos se podrán realizar sin desbordar el número de bits asignados para realizar la operación. Por ejemplo, una serie de operaciones se puede presentar como en la ecuación (1).

$$f = \frac{A \times B \times C}{D \times E} \quad (1)$$

Si la lógica de seguimiento de precisión en una unidad de cálculo determina que dicha operación puede resultar en desbordamiento o subdesbordamiento, la operación se puede transformar en la ecuación (2).

$$f = 10^{(\log(A) + \log(B) + \log(C) - \log(D) - \log(E))} \quad (2)$$

La ecuación (2) se puede llevar a cabo para producir un resultado sin causar un desbordamiento del tipo de datos. En un ejemplo, la lógica 2106A-2016B de transformación y normalización puede transformar los valores de salida en valores logarítmicos para el almacenamiento y transformar los valores mediante exponenciación antes de utilizar los valores para los cálculos de aprendizaje automático descritos en la presente memoria.

La Figura 22 es un diagrama de flujo de lógica 2200 para evitar errores o pérdidas de precisión significativas cuando se llevan a cabo operaciones de precisión baja en aprendizaje automático, según una realización. En una realización la lógica 2200 se puede implementar mediante la lógica 1412 de seguimiento de precisión y una unidad 1422 de transformación numérica en una unidad 1400 de coma flotante de precisión dinámica tal y como se muestra en la Figura 14.

En una realización la lógica 2200 puede calcular un mapa de activación basado en datos de mapa de características y filtros asociado con una capa de una red neuronal, tal y como se muestra en el bloque 2202. Entonces, la lógica 2200 puede llevar a cabo el seguimiento de pérdida de precisión que tiene lugar durante el cálculo de un mapa de activación para la capa de la red neuronal. Entonces, la lógica 2200 puede determinar si la pérdida de precisión se acerca a un umbral en el bloque 2205. Si la pérdida de precisión no se acerca al umbral por defecto o configurado en el bloque 2205, la lógica 2200 puede continuar calculando mapas de activación (y aplicar funciones de activación) para próximas capas hasta que , a menos que en el bloque 2205 ocurra una pérdida de precisión que se acerque a un umbral. Cuando la pérdida de precisión se acerca al umbral, la lógica 2200 puede determinar si se habilitan transformaciones numéricas automáticas en el bloque 2207. Si se habilitan transformaciones automáticas en el bloque 2207, por ejemplo, mediante las instrucciones utilizadas para realizar el conjunto de operaciones numéricas, la lógica 2200 puede transformar los datos de red neuronal para reducir los errores debidos a la pérdida de precisión en el bloque 2208. La lógica 2200 puede llevar a cabo cualquiera de las transformaciones numéricas descritas en la presente memoria, incluida la normalización de datos dentro de un rango o mediante un valor medio. Independientemente de si las transformaciones automáticas se habilitan en el bloque 2207, la lógica 2200 puede emitir estados que indiquen que la pérdida de precisión se está acercando al umbral en el bloque 2210. El estado se puede emitir como una bandera de estado que se emite desde una unidad de cálculo como resultado de una operación realizada. Un programador puede configurar una lógica de software para responder a dicho estado al realizar ajustes algorítmicos a un programa en ejecución o al ajustar el modelo de red neuronal utilizado para llevar a cabo el aprendizaje automático.

Sistema de procesamiento de gráficos de ejemplo adicional

Se pueden incorporar detalles de los ejemplos descritos anteriormente en sistemas y dispositivos de procesamiento de gráficos descritos a continuación. Los dispositivos y el sistema de procesamiento de gráficos de la Figura 23 a la Figura 36 ilustran sistemas alternativos y hardware de procesamiento de gráficos que pueden implementar cualquiera y todas las técnicas descritas anteriormente.

Visión general del sistema de procesamiento de gráficos de ejemplo adicional

La Figura 23 es un diagrama de bloque de un sistema 2300 de procesamiento, según un ejemplo. En diversos ejemplos el sistema 2300 incluye uno o más procesadores 2302 y uno o más procesadores 2308 gráficos, y puede ser un único sistema procesador de sobremesa, un sistema multiprocesador de estación de trabajo o un sistema servidor con un gran número de procesadores 2302 o núcleos 2307 del procesador. En un ejemplo, el sistema 2300 es una plataforma de procesamiento incorporada en un circuito integrado de sistema contenido en un solo chip (SoC) para utilizar en dispositivos móviles, de mano o incorporados.

Un ejemplo del sistema 2300 puede incluir, o tener incorporado en una plataforma de juego basada en servidor, una consola de juego, incluidas una consola de juego y multimedia, una consola de juego móvil, una consola de juego de mano o una consola de juego en línea. En algunos ejemplos el sistema 2300 es un teléfono móvil, un *smartphone*, un dispositivo informático de tableta o un dispositivo móvil con conectividad a Internet. El sistema 2300 de procesamiento de datos puede también incluir, acoplarse con, o estar integrado en un dispositivo ponible, tal y como un dispositivo ponible de *smart watch*, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunos ejemplos, el sistema 2300 de procesamiento de datos es un televisor o un dispositivo descodificador con uno o más procesadores 2302 y una interfaz gráfica generada por uno o más procesadores 2308 gráficos.

En algunos ejemplos, cada uno del uno o más procesadores 2302 incluye uno o más núcleos 2307 del procesador para procesar instrucciones las cuales, cuando se ejecutan, realizan operaciones para el software de usuario y sistema. En algunos ejemplos, cada uno del uno o más núcleos 2307 del procesador está configurado para procesar

un juego 2309 de instrucciones específicas. En algunos ejemplos, el juego 2309 de instrucciones puede facilitar el tratamiento con juego de instrucciones complejo (CISC), el tratamiento con juego de instrucciones reducido (RISC) o el cálculo mediante una palabra de instrucción muy larga (VLIW). Múltiples núcleos 2307 del procesador pueden cada uno procesar un juego 2309 de instrucciones diferente, el cual puede incluir instrucciones para facilitar la emulación de otros juegos de instrucciones. El núcleo 2307 del procesador puede también incluir otros dispositivos de procesamiento, tal y como un procesador de señal digital (DSP).

En algunos ejemplos, el procesador 2302 incluye memoria 2304 caché. Dependiendo de la arquitectura, el procesador 2302 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 2302. En algunos ejemplos, el procesador 2302 también utiliza una caché externa (p. ej., una caché de nivel 3 [L3] o una caché de último nivel [LLC]) (no se muestra), que se puede compartir entre otros núcleos 2307 del procesador al utilizar técnicas de coherencia de caché conocidas. Un fichero 2306 de registro se incluye adicionalmente en el procesador 2302 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (p. ej., registros enteros, registros de coma flotante, registros de estado y un registro de puntero de instrucciones). Algunos registros pueden ser registros para fines generales, mientras que otros registros pueden ser específicos para el diseño del procesador 2302.

En algunos ejemplos, el procesador 2302 está acoplado con un bus 2310 del procesador para transmitir señales de comunicación tal y como señales de dirección, datos o control entre el procesador 2302 y otros componentes en el sistema 2300. En un ejemplo el sistema 2300 utiliza una arquitectura de sistema "centro" de ejemplo, que incluye un centro 2316 controlador de memoria y un centro 2330 controlador de entrada y salida (E/S). Un centro 2316 controlador de memoria facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 2300, mientras que un centro 2330 controlador de E/S (ICH) proporciona conexiones a dispositivos de E/S mediante un bus E/S local. En un ejemplo, la lógica del centro 2316 controlador de memoria está integrada en el procesador.

El dispositivo 2320 de memoria puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria estática de acceso aleatorio (SRAM), un dispositivo de memoria flash, un dispositivo de memoria de cambio de fase u otro dispositivo de memoria con un rendimiento adecuado para servir como memoria de proceso. En un ejemplo el dispositivo 2320 de memoria puede funcionar como memoria de sistema para el sistema 2300, para almacenar datos 2322 e instrucciones 2321 para utilizarlos cuando el uno o más procesadores 2302 ejecutan una aplicación o proceso. El centro 2316 controlador de memoria también se acopla con un procesador 2312 gráfico externo opcional, el cual se puede comunicar con el uno o más procesadores 2308 gráficos en los procesadores 2302 para llevar a cabo operaciones gráficas y de medios.

En algunos ejemplos, el ICH 2330 permite que los periféricos se conecten a un dispositivo 2320 de memoria y al procesador 2302 mediante un bus E/S de alta velocidad. Los periféricos de E/S incluyen, pero no se limitan a, un controlador 2346 de audio, una interfaz 2328 de firmware, un transceptor 2326 inalámbrico (p. ej., Wi-Fi, Bluetooth), un dispositivo 2324 de almacenamiento de datos (p. ej., unidad de disco duro, memoria flash, etc.) y un controlador 2340 de E/S heredado para acoplar dispositivos heredados (p. ej., sistema personal 2 [PS/2]) al sistema. Uno o más controladores 2342 de bus serial universal (USB) conectan dispositivos de entrada, tal y como combinaciones de teclado y ratón 2344. Un controlador 2334 de red puede también acoplarse con el ICH 2330. En algunos ejemplos, un controlador de red de alto rendimiento (no se muestra) se acopla con el bus 2310 del procesador. Se apreciará que el sistema 2300 que se muestra es de ejemplo y no limitante, ya que también se pueden utilizar otros tipos de sistemas de procesamiento de datos que se configuran de otra manera. Por ejemplo, el centro 2330 controlador de E/S puede estar integrado en el uno o más procesadores 2302, o el centro 2316 controlador de memoria y el centro 2330 controlador de E/S pueden estar integrados en un procesador gráfico externo discreto, tal y como el procesador 2312 gráfico externo.

La Figura 24 es un diagrama de bloque de un ejemplo de un procesador 2400 con uno o más núcleos 2402A-2402N del procesador, un controlador 2414 de memoria integrado, y un procesador 2408 gráfico integrado. Dichos elementos de la Figura 24 con los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en la presente memoria pueden funcionar de cualquier manera similar a la descrita en otro lugar de la presente memoria, pero no se limitan a ella. El procesador 2400 puede incluir núcleos adicionales hasta e incluido el núcleo 2402N adicional representado por las casillas de líneas discontinuas. Cada uno de los núcleos 2402A-2402N del procesador incluye una o más unidades 2404A-2404N de caché internas. En algunos ejemplos cada núcleo del procesador tiene también acceso a una o más unidades 2406 de caché compartida.

Las unidades 2404A-2404N de caché internas y las unidades 2406 de caché compartida representan una jerarquía de memoria caché en el procesador 2400. La jerarquía de memoria caché puede incluir al menos un nivel de instrucción y caché de datos en cada núcleo del procesador y uno o más niveles de caché de nivel medio compartidos, tal y como un nivel 2 (L2), nivel 3 (L3), nivel 4 (L4) u otros niveles de caché, donde el nivel de caché más alto antes de la memoria externa se clasifica como el LLC. En algunos ejemplos, la lógica de coherencia de caché mantiene coherencia entre las diversas unidades 2406 y 2404A-2404N de caché.

En algunos ejemplos, el procesador 2400 puede también incluir un conjunto de una o más unidades 2416 de controlador de bus y un núcleo 2410 de agente de sistema. La una o más unidades 2416 de controlador de bus gestionan un conjunto de buses periféricos, tal y como uno o más buses de interconexión de componentes periféricos

(p. ej., PCI, PCI Express). El núcleo 2410 de agente de sistema proporciona una funcionalidad de gestión para los diversos componentes del procesador. En algunos ejemplos, el núcleo 2410 de agente de sistema incluye uno o más controladores 2414 de memoria integrados para gestionar el acceso a diversos dispositivos de memoria externa (no se muestra).

En algunos ejemplos, uno o más núcleos 2402A-2402N del procesador incluyen apoyo para un multitratamiento simultáneo. En dicho ejemplo, el núcleo 2410 de agente de sistema incluye componentes para núcleos 2402A-2402N de coordinación y funcionamiento durante el procesamiento de multitratamiento. El núcleo 2410 de agente de sistema puede además incluir una unidad de control de potencia (PCU), la cual incluye lógica y componentes para regular el estado de potencia de los núcleos 2402A-2402N del procesador y los procesadores 2408 gráficos.

En algunos ejemplos, el procesador 2400 incluye además un procesador 2408 gráfico para ejecutar operaciones de procesamiento de gráficos. En algunos ejemplos, el procesador 2408 gráfico se acopla con el conjunto de unidades 2406 de caché compartida y el núcleo 2410 de agente de sistema, incluido el uno o más controladores 2414 de memoria integrados. En algunos ejemplos, un controlador 2411 de visualización está acoplado con el procesador 2408 gráfico para dirigir la salida del procesador gráfico a una o más visualizadores. En algunos ejemplos, el controlador 2411 de visualización puede ser un módulo separado acoplado con el procesador gráfico mediante al menos una interconexión, o puede estar integrado en el procesador 2408 gráfico o núcleo 2410 de agente de sistema.

En algunos ejemplos, una unidad 2412 de interconexión en anillo se utiliza para acoplar los componentes internos del procesador 2400. Sin embargo, se puede utilizar una unidad de interconexión alternativa, tal y como una interconexión punto a punto, una interconexión de conmutación u otras técnicas, incluidas las técnicas bien conocidas en la técnica. En algunos ejemplos, el procesador 2408 gráfico se acopla con la interconexión 2412 de anillo mediante un enlace 2413 de E/S.

El enlace 2413 de E/S de ejemplo representa al menos una de múltiples variedades de interconexiones de E/S, incluida una interconexión de E/S en paquete la cual facilita la comunicación entre diversos componentes del procesador y un módulo 2418 de memoria incorporada de alto rendimiento, tal y como un módulo eDRAM. En algunos ejemplos, cada uno de los núcleos 2402A-2402N del procesador y el procesador 2408 gráfico utiliza los módulos 2418 de memoria incorporada como una caché de último nivel compartida.

En algunos ejemplos, los núcleos 2402A-2402N del procesador son núcleos homogéneos que ejecutan la misma arquitectura del conjunto de instrucciones. En otro ejemplo, los núcleos 2402A-2402N del procesador son heterogéneos en términos de arquitectura del conjunto de instrucciones (ISA), donde uno o más núcleos 2402A-2402N del procesador 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 2402A-2402N del procesador son heterogéneos en términos de microarquitectura, donde uno o más núcleos con un consumo de energía relativamente más alto se acoplan con uno o más núcleos de potencia con un consumo de energía más bajo. Adicionalmente, el procesador 2400 se puede implementar en uno o más chips o como un circuito integrado SoC con los componentes ilustrados, además de otros componentes.

La Figura 25 es un diagrama de bloque de un procesador 2500 gráfico, que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador gráfico integrado con una pluralidad de núcleos de procesamiento. En algunos ejemplos, el procesador gráfico se comunica a través de una interfaz de E/S mapeada en memoria con los registros en el procesador gráfico y con comandos colocados en la memoria del procesador. En algunos ejemplos, el procesador 2500 gráfico incluye una interfaz 2514 de memoria para acceder a la memoria. La interfaz 2514 de memoria puede ser una interfaz para la memoria local, una o más cachés internas, una o más cachés externas compartidas y/o para la memoria de sistema.

En algunos ejemplos, el procesador 2500 gráfico también incluye un controlador 2502 de visualización para dirigir datos de salida a un dispositivo 2520 de visualización. El controlador 2502 de visualización incluye hardware para uno o más planos superpuestos para la visualización y composición de múltiples capas de elementos de interfaz de vídeo o usuario. En algunos ejemplos, el procesador 2500 gráfico incluye un motor 2506 de códec de vídeo para codificar, descodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluidos pero no limitados a formatos de Grupo de expertos de Imagen en Movimiento (MPEG) tal y como MPEG-2, formatos de codificación de vídeo avanzada (AVC) tal y como H.264/MPEG-4 AVC, así como la Sociedad de Ingenieros de Imágenes en Movimiento y Televisión (SMPTE) 421M/VC-1, y formatos de Grupo Mixto de Expertos en Fotografía (JPEG) y formatos Motion JPEG (MJPEG).

En algunos ejemplos, el procesador 2500 gráfico incluye un motor 2504 de transferencia de imágenes en bloque (BLIT) para realizar operaciones de rasterización de dos dimensiones (2D) que incluyen, por ejemplo, transferencias de bloques en frontera de bits. Sin embargo, en un ejemplo, las operaciones gráficas 2D se llevan a cabo al utilizar uno o más componentes del motor 2510 de procesamiento de gráficos (GPE). En algunos ejemplos, el GPE 2510 es un motor de cálculo para realizar operaciones gráficas, incluidas las operaciones gráficas de tres dimensiones (3D) y las operaciones de medios.

En algunos ejemplos, el GPE 310 incluye una canalización 2512 3D para realizar operaciones 3D, tal y como representar imágenes y escenas de tres dimensiones que utilizan funciones de procesamiento que actúan sobre formas primitivas 3D (p. ej., rectángulo, triángulo, etc.). La canalización 2512 3D incluye elementos de función programable y fija que realizan diversas tareas en el elemento y/o generan hilos de ejecución a un subsistema 2515 3D/Medios. Mientras que la canalización 2512 3D se puede utilizar para realizar operaciones de medios, un ejemplo de GPE 2510 también incluye una canalización 2516 de medios que se utiliza específicamente para llevar a cabo operaciones de medios, tal y como el postprocesamiento de vídeos y el realce de imágenes.

En algunos ejemplos, la canalización 2516 de medios incluye unidades de lógica programable o de función fija para realizar una o más operaciones de medios especializadas, tal y como la aceleración de la descodificación de vídeos, el desentrelazado de vídeos y la aceleración de la codificación de vídeos en lugar de, o en nombre de, el motor 2506 de códec de vídeo. En algunos ejemplos, la canalización 2516 de medios además incluye una unidad generadora de hilos para generar hilos para ejecutar en el subsistema 2515 3D/Medios. Los hilos generados llevan a cabo cálculos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema 2515 3D/Medios.

En algunos ejemplos, el subsistema 2515 3D/Medios incluye lógica para ejecutar hilos generados por la canalización 2512 3D y la canalización 2516 de medios. En un ejemplo, las canalizaciones envían solicitudes de ejecución de hilos al subsistema 2515 3D/Medios, que incluye lógica de despacho de hilos para mediar y despachar las diversas solicitudes a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una disposición de unidades de ejecución de gráficos para procesar los hilos 3D y de medios. En algunos ejemplos, el subsistema 2515 3D/Medios incluye una o más cachés internas para datos e instrucciones de hilos. En algunos ejemplos, el subsistema también incluye una memoria compartida, incluidos registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Motor de procesamiento de gráficos adicional de ejemplo

La Figura 26 es un diagrama de bloque de un motor 2610 de procesamiento de gráficos de un procesador gráfico según algunos ejemplos. En un ejemplo, el motor de procesamiento de gráficos (GPE) 2610 es una versión del GPE 2510 que se muestra en la Figura 25**.** Los elementos de la Figura 26 con los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en la presente memoria pueden funcionar de cualquier manera similar a la descrita en otro lugar de la presente memoria, pero no se limitan a ella. Por ejemplo, se ilustran la canalización 2512 3D y la canalización 2516 de medios de la figura 25. La canalización 2516 de medios es opcional en algunas reivindicaciones del GPE 2610 y puede no estar incluida explícitamente en el GPE 2610. Por ejemplo y en al menos un ejemplo, un medio separado y/o un procesador de imágenes está acoplado al GPE 2610.

En algunos ejemplos, el GPE 2610 se acopla con o incluye un transmisor de tren 2603 de comandos, que proporciona un tren de comandos a la canalización 2512 3D y/o a las canalizaciones 2516 de medios. En algunos ejemplos, el transmisor de tren 2603 de comandos está acoplado con la memoria, que puede ser una memoria de sistema, o una o más memorias caché internas y memorias caché compartidas. En algunos ejemplos, el transmisor de tren 2603 de comandos recibe comandos de la memoria y envía los comandos a la canalización 2512 3D y/o la canalización 2516 de medios. Los comandos son directivas capturadas de una memoria intermedia de anillo, que almacena comandos para la canalización 2512 3D y la canalización 2516 de medios. En un ejemplo, la memoria intermedia de anillo puede además incluir memorias intermedias de comando por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización 2512 3D pueden también incluir referencias a datos almacenados en la memoria, tal y como pero no limitado a datos de vértices y geometría para la canalización 2512 3D y/o datos de imagen y objetos de memoria para la canalización 2516 de medios. La canalización 2512 3D y la canalización 2516 de medios procesan los comandos y los datos al realizar operaciones a través de lógica en las respectivas canalizaciones o al despachar uno o más hilos de ejecución a una disposición 2614 de núcleo gráfico.

En diversos ejemplos la canalización 2512 3D puede ejecutar uno o más programas de sombreado, tal y como sombreadores de vértices, sombreadores de geometrías, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculos u otros programas de sombreado, al procesar las instrucciones y despachar hilos de ejecución a la disposición 2614 de núcleo gráfico. La disposición 2614 de núcleo gráfico proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución para múltiples fines (p. ej., unidades de ejecución) en la disposición 2614 de núcleo gráfico incluye apoyo para diversos lenguajes de sombreado 3D API y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunos ejemplos la disposición 2614 de núcleo gráfico también incluye lógica de ejecución para llevar a cabo funciones de medios, tal y como el procesamiento de vídeos y/o imágenes. En un ejemplo, las unidades de ejecución además incluyen lógica para fines generales que es programable para realizar operaciones de cálculo para fines generales, además de operaciones de procesamiento de gráficos. La lógica para fines generales puede realizar operaciones de procesamiento en paralelo o en conjunto con lógica para fines generales en el(los) núcleo(s) 2307 de la Figura 23, núcleo 2402A-2402N tal y como en la Figura 24, o cualquier otro procesador descrito en la presente memoria.

Los datos de salida generados por hilos ejecutados en la disposición 2614 de núcleo gráfico pueden emitir datos a la memoria en una memoria intermedia 2618 de retorno unificada (URB). La URB 2618 puede almacenar datos para múltiples hilos. En algunos ejemplos la URB 2618 se puede utilizar para enviar datos entre diferentes hilos que se ejecutan en la disposición 2614 de núcleo gráfico. En algunos ejemplos la URB 2618 se puede utilizar además para la sincronización entre hilos en la disposición de núcleo gráfico y en la lógica de función fija en la lógica 2620 de función compartida.

En algunos ejemplos, la disposición 2614 de núcleo gráfico es escalable, de modo que la disposición incluye un número variable de núcleos gráficos, cada uno con un número variable de unidades de ejecución basadas en la potencia diana y el nivel de rendimiento del GPE 2610. En un ejemplo los recursos de ejecución son dinámicamente escalables, de modo que los recursos de ejecución se pueden habilitar o deshabilitar según sea necesario.

La disposición 2614 de núcleo gráfico se acopla con la lógica 2620 de función compartida que incluye múltiples recursos que se comparten entre los núcleos gráficos en la disposición de núcleo gráfico. Las funciones compartidas en la lógica 2620 de función compartida son unidades de lógica de hardware que proporcionan funcionalidad complementaria especializada a la disposición 2614 de núcleo gráfico. En diversos ejemplos, la lógica 2620 de función compartida incluye, pero no se limita, a sacamuestras 2621, matemáticas 2622 y lógica 2623 de comunicación entre hilos (ITC). Adicionalmente, algunos ejemplos implementan una o más cachés 2625 en la lógica 2620 de función compartida. Una función compartida se implementa donde la demanda de una función especializada dada es insuficiente para incluirla en la disposición 2614 de núcleo gráfico. En cambio, se implementa una única instanciación de dicha función especializada como una entidad autónoma en la lógica 2620 de función compartida y se comparte entre los recursos de ejecución en la disposición 2614 de núcleo gráfico. El conjunto preciso de funciones que se comparten entre la disposición 2614 de núcleo gráfico y que se incluyen en la disposición 2614 de núcleo gráfico varía entre ejemplos.

La Figura 27 es un diagrama de bloque de otro ejemplo de un procesador 2700 gráfico. Los elementos de la Figura 27 con los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en la presente memoria pueden funcionar de cualquier manera similar a la descrita en otro lugar de la presente memoria, pero no se limitan a ella.

En algunos ejemplos, el procesador 2700 gráfico incluye una interconexión 2702 de anillo, un *front-end* 2704 de la canalización, un motor 2737 de medios y núcleos 2780A-2780N gráficos. En algunos ejemplos, la interconexión 2702 de anillo acopla el procesador gráfico con otras unidades de procesamiento, incluidos otros procesadores gráficos o uno o más núcleos de procesador para fines generales. En algunos ejemplos, el procesador gráfico es uno de muchos procesadores integrados en un sistema de procesamiento multinúcleo.

En algunos ejemplos, el procesador 2700 gráfico recibe lotes de comandos a través de la interconexión 2702 de anillo. Un transmisor de tren 2703 de comandos interpreta los comandos entrantes en el *front end* 2704 de la canalización. En algunos ejemplos, el procesador 2700 gráfico incluye una lógica de ejecución escalable para llevar a cabo el procesamiento de geometría 3D y el procesamiento de medios a través del(los) núcleo(s) 2780A-2780N gráfico(s). Para los comandos de procesamiento de geometría 3D, el transmisor de tren 2703 de comandos proporciona comandos a la canalización 2736 de geometría. Para al menos algunos comandos de procesamiento de medios, el transmisor de tren 2703 de comandos proporciona los comandos a un *front end* 2734 de vídeo, que se acopla con un motor 2737 de medios. En algunos ejemplos, el motor 2737 de medios incluye un motor 2730 de calidad de vídeo (VQE) para el postprocesamiento de vídeos e imágenes y un motor 2733 codificador/descodificador multiformato (MFX) para proporcionar codificación y descodificación de datos de medios acelerados por hardware. En algunos ejemplos, la canalización 2736 de geometría y el motor 2737 de medios generan cada uno hilos de ejecución para los recursos de ejecución proporcionados por al menos un núcleo 2780A gráfico.

En algunos ejemplos, el procesador 2700 gráfico incluye recursos de ejecución de hilos escalables que presentan núcleos 2780A-2780N modulares (a veces denominados segmentos de núcleo), cada uno con múltiples subnúcleos 2750A-550N, 2760A-2760N (a veces denominados subsegmentos de núcleo). En algunos ejemplos, el procesador 2700 gráfico puede tener cualquier número de núcleos gráficos 2780A hasta 2780N. En algunos ejemplos, el procesador 2700 gráfico incluye un núcleo 2780A gráfico con al menos un primer subnúcleo 2750A y un segundo subnúcleo 2760A. En otros ejemplos, el procesador gráfico es un procesador de baja potencia con un único subnúcleo (p. ej., 2750A). En algunos ejemplos, el procesador 2700 gráfico incluye múltiples núcleos 2780A-2780N gráficos, cada uno con un conjunto de primeros subnúcleos 2750A-2750N y un conjunto de segundos subnúcleos 2760A-2760N. Cada subnúcleo en el conjunto de primeros subnúcleos 2750A-2750N incluye al menos un primer conjunto de unidades 2752A-2752N de ejecución y sacamuestras 2754A-2754N de medios/textura. Cada subnúcleo en el conjunto de segundos subnúcleos 2760A-2760N incluye al menos un segundo conjunto de unidades 2762A-2762N de ejecución y sacamuestras 2764A-2764N. En algunos ejemplos, cada subnúcleo 2750A-2750N, 2760A-2760N comparte un conjunto de recursos 2770A-2770N compartidos. En algunos ejemplos, los recursos compartidos incluyen memoria caché compartida y lógica de operación de píxeles. Otros recursos compartidos pueden también estar incluidos en las diversos ejemplos del procesador gráfico.

Unidades de ejecución adicionales de ejemplo

La Figura 28 ilustra una lógica 2800 de ejecución de hilo que incluye una disposición de elementos de procesamiento empleado en algunos ejemplos de un GPE. Los elementos de la Figura 28 con los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en la presente memoria pueden funcionar de cualquier manera similar a la descrita en otro lugar de la presente memoria, pero no se limitan a ella.

En algunos ejemplos, la lógica 2800 de ejecución de hilos incluye un procesador 2802 de sombreado, un despachador 2804 de hilos, una caché 2806 de instrucciones, una disposición de unidad de ejecución escalable que incluye una pluralidad de unidades 2808A-2808N de ejecución, un sacamuestras 2810, una caché 2812 de datos y un puerto 2814 de datos. En un ejemplo la disposición de unidad de ejecución escalable puede escalar dinámicamente al activar o desactivar una o más unidades de ejecución (p. ej., cualquiera de las unidades de ejecución 2808A, 2808B, 2808C, 2808D, hasta 2808N-1 y 2808N) basadas en los requerimientos de cálculo de un volumen de trabajo. En un ejemplo los componentes incluidos están interconectados a través de un tejido de interconexión que se conecta a cada uno de los componentes. En algunos ejemplos, la lógica 2800 de ejecución de hilos incluye una o más conexiones a una memoria, tal y como una memoria de sistema o una memoria caché, a través de una o más cachés 2806 de instrucciones, puertos 2814 de datos, sacamuestras 2810 y unidades 2808A-2808N de ejecución. En algunos ejemplos, cada unidad de ejecución (p. ej., 2808A) es una unidad de cálculo para fines generales programable y autónoma que es capaz de ejecutar múltiples hilos de hardware simultáneos mientras que procesa múltiples elementos de datos en paralelo para cada hilo. En diversos ejemplos, la disposición de unidades 2808A-2808N de ejecución es escalable para incluir cualquier número de unidades de ejecución individuales.

En algunos ejemplos, las unidades 2808A-2808N de ejecución se utilizan principalmente para ejecutar programas de sombreado. Un procesador 2802 de sombreado puede procesar los diversos programas de sombreado y despachar hilos de ejecución asociados con los programas de sombreado mediante un despachador 2804 de hilos. En un ejemplo el despachador de hilos incluye lógica para mediar las solicitudes de iniciación de hilos de las canalizaciones gráficas y de medios e instanciar los hilos solicitados en la una o más unidades de ejecución en las unidades 2808A-2808N de ejecución. Por ejemplo, la canalización de geometrías (p. ej., el 2736 de la Figura 27) puede despachar sombreadores de vértices, teselación o geometría a la lógica 2800 de ejecución de hilos (Figura 28) para su procesamiento. En algunos ejemplos, el despachador 2804 de hilos puede también procesar solicitudes de generación de hilos de ejecución de los programas de sombreado en ejecución.

En algunos ejemplos, las unidades 2808A-2808N de ejecución admiten un conjunto de instrucciones que incluye soporte nativo para gran cantidad de instrucciones de sombreado gráfico 3D estándares, de modo que los programas de sombreado de las bibliotecas gráficas (p. ej., Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución admiten el procesamiento de vértices y geometrías (p. ej., programas de vértices, programas de geometrías, sombreadores de vértices), el procesamiento de píxeles (p.ej., sombreadores de píxeles, sombreadores de fragmentos) y el procesamiento para fines generales (p. ej., sombreadores de cálculos y medios). Cada una de las unidades 2808A-2808N de ejecución es capaz de emitir múltiples ejecuciones de instrucción única para datos múltiples (SIMD) y la operación de multitratamiento permite un entorno de ejecución eficiente frente a accesos de memoria de latencia más altos. Cada hilo de hardware en cada unidad de ejecución tiene un fichero de registro con ancho de banda alto dedicado y un hilo-estado independiente asociado. La ejecución de emisión múltiple por reloj a canalizaciones capaces de realizar operaciones de coma flotante de precisión doble, entera y simple, con capacidad de ramificar SIMD, operaciones lógicas, operaciones transcendentales y otras operaciones varias. Mientras que se espera a los datos de la memoria o de una de las funciones compartidas, la lógica de dependencia en las unidades 2808A-2808N de ejecución hace que un hilo en espera se duerma hasta que se hayan devuelto los datos solicitados. Mientras que el hilo en espera duerme, los recursos de hardware pueden estar destinados a procesar otros hilos. Por ejemplo, durante un retraso asociado con una operación de sombreado de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, sombreador de fragmentos u otro tipo de programa de sombreado, incluido un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades 2808A-2808N de ejecución opera en disposiciones 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 acceder a elementos de datos, enmascarar y controlar el flujo en las instrucciones. El número de canales puede ser independiente del número de unidades aritmético-lógicas (ALU) o unidades de coma flotante (FPU) para un procesador gráfico particular. En algunos ejemplos, las unidades 2808A-2808N de ejecución admiten tipos de datos de coma flotante y enteros.

El conjunto de instrucciones de unidad de ejecución incluye instrucciones SIMD. Los diversos elementos de datos se pueden almacenar como un tipo de datos empaquetados 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, los 256 bits del vector se almacenan en un registro y la unidad de ejecución opera en el vector como cuatro elementos de datos de 64 bits empaquetados separados (elementos de datos de tamaño Quad Word [QW]), ocho elementos de datos de 32 bits empaquetados separados (elementos de datos de tamaño Double Word [DW]), dieciséis elementos de datos de 16 bits empaquetados separados (elementos de datos de tamaño Word [W]) o treinta y dos elementos de datos de 8 bits separados (elementos de datos de tamaño byte [B]). Sin embargo, son posibles diferentes anchuras de vector y tamaños de registro.

Se incluyen una o más cachés de instrucción internas (p. ej., 2806) en la lógica 2800 de ejecución de hilos a las instrucciones de hilo de caché para las unidades de ejecución. En algunos ejemplos, se incluyen una o más cachés de datos (p. ej., 2812) a los datos de hilo de caché durante la ejecución de hilos. En algunos ejemplos, se incluye un sacamuestras 2810 para proporcionar muestras de textura para operaciones 3D y muestras de medios para operaciones de medios. En algunos ejemplos, el sacamuestras 2810 incluye funcionalidad de muestras de texturas o de medios especializada para procesar datos de medios o texturas durante el proceso de muestreo antes de proporcionar los datos de muestra a una unidad de ejecución.

Durante la ejecución, las canalizaciones gráficas y de medios envían solicitudes de iniciación de hilos a la lógica 2800 de ejecución de hilos a través de la generación de hilos y la lógica de despacho. Una vez se ha procesado un grupo de objetos geométricos y se ha rasterizado en datos de píxeles, la lógica de procesador de píxeles (p. ej., lógica de sombreado de píxeles, lógica de sombreado de fragmentos, etc.) en el procesador 2802 de sombreado se invoca para además calcular la información de salida y hacer que los resultados estén escritos en superficies de salida (p. ej., memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de plantillas, etc.). En algunos ejemplos, un sombreador de píxeles o sombreador de fragmentos calcula los valores de los diversos atributos de vértices que se van a interpolar a lo largo del objeto rasterizado. En algunos ejemplos, la lógica de procesador de píxeles en el procesador 2802 de sombreado ejecuta entonces un programa de sombreado de píxeles o de fragmentos con una interfaz de programación de aplicación (API). Para ejecutar el programa de sombreado, el procesador 2802 despacha hilos a una unidad de ejecución (p. ej., 2808A) a través de un despachador 2804 de hilos. En algunas aplicaciones, el sombreador 2802 de píxeles utiliza lógica de muestreo de textura en el sacamuestras 2810 para acceder a los datos de texturas en mapas de textura almacenados en la memoria. Las operaciones aritméticas en los datos de texturas y los datos de geometría de entrada calculan datos de color de píxeles para cada fragmento geométrico, o descartan uno o más píxeles de un procesamiento adicional.

En algunos ejemplos, el puerto 2814 de datos proporciona un mecanismo de acceso de memoria para los datos procesados de salida de la lógica 2800 de ejecución de hilos a la memoria para procesar en una canalización de salida de procesador gráfico. En algunos ejemplos, el puerto 2814 de datos incluye o se acopla a una o más memorias caché (p. ej., caché 2812 de datos) a datos de caché para el acceso de memoria a través del puerto de datos.

La Figura 29 es un diagrama de bloque que ilustra formatos 2900 de instrucción del procesador gráfico según algunos ejemplos. En una o más ejemplos, las unidades de ejecución del procesador gráfico admiten un conjunto de instrucciones con instrucciones en múltiples formatos. Las casillas de líneas continuas ilustran los componentes que están generalmente incluidos en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que se incluyen únicamente en un subconjunto de las instrucciones. En algunos ejemplos, el formato 2900 de instrucción descrito e ilustrado son macroinstrucciones, en cuanto a que son instrucciones proporcionadas a la unidad de ejecución, en oposición a las microoperaciones que resultan de la descodificación de instrucciones una vez se procesa la instrucción.

En algunos ejemplos, las unidades de ejecución de procesador gráfico admiten de forma nativa instrucciones en un formato 2910 de instrucción de 128 bits. Un formato 2930 de instrucción compacto de 64 bits está disponible para algunas instrucciones en base a la instrucción, opciones de instrucción y número de operandos seleccionados. El formato 2910 de instrucción nativo de 128 bits proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato 2930 de 64 bits. Las instrucciones nativas disponibles en el formato 2930 de 64 bits varían según el ejemplo. En algunos ejemplos, la instrucción se compacta en parte al utilizar un conjunto de valores índice en un campo 2913 índice. El hardware de unidad de ejecución menciona un conjunto de tablas de compactación basado en los valores índices y utiliza las salidas de las tablas de compactación para reconstruir una instrucción nativa en el formato 2910 de instrucción de 128 bits.

Para cada formato, el código de operación 2912 de instrucción define la operación que la unidad de ejecución va a realizar. Las unidades de ejecución ejecutan cada instrucción en paralelo a lo largo 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 a lo largo de cada canal de color que representa un elemento de textura o elemento de imagen. Por defecto, la unidad de ejecución lleva a cabo cada instrucción a lo largo de todos los canales de datos de los operandos. En algunos ejemplos, el campo 2914 de control de instrucción habilita el control sobre determinadas opciones de ejecución, tal y como la selección de canales (p. ej., la predicación) y el orden de canales de datos (p. ej., *swizzle*). Para instrucciones en el formato 2910 de instrucción de 128 bits un campo 2916 de tamaño de ejecución limita el número de canales de datos que se ejecutarán en paralelo. En algunos ejemplos, el campo 2916 de tamaño de ejecución no está disponible para utilizarlo en el formato 2930 de instrucción compacto de 64 bits.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos que incluyen dos operandos fuente, src0 2920, src1 2922 y un destino 2918. En algunos ejemplos, las unidades de ejecución admiten instrucciones de destino dobles, donde está implicado uno de los destinos. Las instrucciones de manipulación de datos pueden tener un tercer operando fuente (p. ej., SRC2 2924), donde el código de operaciones 2912 de instrucción determina el número de operandos fuente. Un último operando fuente de una instrucción puede ser un valor inmediato (p. ej., codificado de forma fija) pasado con la instrucción.

En algunos ejemplos, el formato 2910 de instrucción de 128 bits incluye un campo 2926 de modo de acceso/direccionamiento que especifica, por ejemplo, si se utiliza el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, el direccionamiento de registro de uno o más operandos se proporciona directamente por bits en la instrucción.

En algunos ejemplos, el formato 2910 de instrucción de 128 bits incluye un campo 2926 de modo de acceso/direccionamiento, que especifica un modo de direccionamiento y/o un modo de acceso para la instrucción. En un ejemplo se utiliza el modo de acceso para definir un alineamiento de acceso de datos para la instrucción. Algunos ejemplos admiten modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, donde el alineamiento de bytes del modo de acceso determina el alineamiento de acceso de los operandos de instrucción. Por ejemplo, cuando está en un primer modo, la instrucción puede utilizar el direccionamiento de bytes alineados para operandos fuente y destino y cuando está en un segundo modo, la instrucción puede utilizar el direccionamiento alineado de 16 bytes para todos los operandos fuente y destino.

En un ejemplo, la porción del modo de direccionamiento del campo 2926 de modo de acceso/direccionamiento determina si la instrucción es para utilizar direccionamiento directo o indirecto. Cuando se utiliza el modo de direccionamiento de registro directo los bits en la instrucción directamente proporcionan al direccionamiento de registro de uno o más operandos. Cuando se utiliza el modo de direccionamiento de registro indirecto, el direccionamiento de registro de uno o más operandos se puede calcular en base a un valor de registro de direccionamiento y un campo inmediato de direccionamiento en la instrucción.

En algunos ejemplos las instrucciones se agrupan en base a campos de bits de código de operaciones 2912 para simplificar la descodificación 2940 de código de operaciones. 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 de código de operación preciso que se muestra es meramente un ejemplo. En algunos ejemplos, un grupo 2942 de código de operación lógico incluye movimiento de datos e instrucciones lógicas (p. ej., mover [mov], comparar [comp]). En algunos ejemplos, el grupo 2942 de mover y de lógica comparte los cinco bits más significativos (MSB), donde instrucciones de mover (mov) están en la forma de 0000xxxxb y las instrucciones lógicas están en la forma de 0001xxxxb. Un grupo 2944 de instrucción de control de flujo (p. ej., llamar, saltar [jmp]) incluye instrucciones en la forma de 0010xxxxb (p. ej., 0x20). Un grupo 2946 de instrucciones varias incluye una mezcla de instrucciones, incluidas instrucciones de sincronización (p. ej., esperar, enviar) en la forma de 0011xxxxb (p. ej., 0x30). Un grupo 2948 de instrucción de matemáticas paralelo incluye instrucciones aritméticas por componentes (p. ej., adición, multiplicación [mul]) en la forma de 0100xxxxb (p. ej., 0x40). El grupo 2948 de matemáticas paralelo realiza las operaciones aritméticas en paralelo a lo largo de canales de datos. Un grupo 2950 de matemáticas de vector incluye instrucciones aritméticas (p. ej., dp4) en la forma de 0101xxxxb (p. ej., 0x50). El grupo de matemáticas de vector lleva a cabo la aritmética tal y como cálculos de producto escalar en operandos de vector.

Canalización gráfica adicional de ejemplo

La Figura 30 es un diagrama de bloque de otro ejemplo de un procesador 3000 gráfico. Los elementos de la Figura 30 con los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en la presente memoria pueden funcionar de cualquier manera similar a la descrita en otro lugar de la presente memoria, pero no se limitan a ella.

En algunos ejemplos, el procesador 3000 gráfico incluye una canalización 3020 gráfica, una canalización 3030 de medios, un motor 3040 de visualización, lógica 3050 de ejecución de hilos y una canalización 3070 de salida de representación. En algunos ejemplos, el procesador 3000 gráfico es un procesador gráfico en un sistema de procesamiento multinúcleo que incluye uno o más núcleos de procesamiento para fines generales. El procesador gráfico se controla por registros de escritura a uno o más registros de control (no se muestran) o a través de comandos emitidos al procesador 3000 gráfico mediante interconexiones 3002 de anillo. En algunos ejemplos, la interconexión 3002 de anillo acopla el procesador 3000 gráfico con otros componentes de procesamiento, tal y como otros procesadores gráficos o procesadores para fines generales. Los comandos de la interconexión 3002 de anillo son interpretados por un transmisor de tren 3003 de comandos, que proporciona instrucciones a componentes individuales de la canalización 3020 gráfica o canalización 3030 de medios.

En algunos ejemplos, el transmisor de tren 3003 de comandos dirige la operación de un capturador 3005 de vértices que lee datos de vértices de la memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor de tren 3003 de comandos. En algunos ejemplos, el capturador 3005 de vértices proporciona datos de vértices al sombreador 3007 de vértices, que lleva a cabo una transformación de espacio de coordenada y operaciones de iluminación a cada vértice. En algunos ejemplos, el capturador 3005 de vértices y el sombreador 3007 de vértices ejecuta instrucciones de procesamiento de vértices al despachar hilos de ejecución a las unidades 3052A-3052B de ejecución mediante un despachador 3031 de hilos.

En algunos ejemplos, las unidades 3052A-3052B de ejecución son una disposición de procesadores de vectores con un conjunto de instrucciones para realizar operaciones gráficas y de medios. En algunos ejemplos, las unidades 3052A-3052B de ejecución tienen una caché 3051 de L1 fijada que es específica para cada disposición o se comparte

entre las disposiciones. La caché puede estar configurada como una caché de datos, una caché de instrucción o una caché única que está particionada para contener datos e instrucciones en diferentes particiones.

En algunos ejemplos, la canalización 3020 gráfica incluye componentes de teselación para llevar a cabo la teselación acelerada por hardware de objetos 3D. En algunos ejemplos, un sombreador 811 de casco programable configura las operaciones de teselación. Un sombreador 817 de dominio programable proporciona la evaluación *back end* de la salida de teselación. Un teselador 3013 opera en la dirección del sombreador 3011 de casco y contiene lógica para fines específicos para generar un conjunto de objetos geométricos detallados en base a un modelo geométrico aproximado que se proporciona como entrada a la canalización 3020 gráfica. En algunos ejemplos, si no se utiliza la teselación, los componentes de teselación (p. ej., el sombreador 3011 de casco, el teselador 3013 y el sombreador 3017 de dominio) se pueden evitar.

En algunos ejemplos, los objetos geométricos completos pueden ser procesados por un sombreador 3019 de geometría mediante uno o más hilos despachados a unidades 3052A-3052B de ejecución, o pueden proceder directamente al recortador 3029. En algunos ejemplos, el sombreador de geometría opera en objetos geométricos enteros, en lugar de en vértices o partes de vértices como en etapas previas de la canalización gráfica. Si se deshabilita la teselación el sombreador 3019 de geometría recibe entradas del sombreador 3007 de vértices. En algunos ejemplos, el sombreador 3019 de geometría es programable por un sombreador de geometría para llevar a cabo la teselación de geometría si se deshabilitan las unidades de teselación.

Antes de la rasterización, un recortador 3029 procesa los datos de vértices. El recortador 3029 puede ser un recortador de función fija o un recortador programable con funciones de sombreador de geometría y de recorte. En algunos ejemplos, un rasterizador y componente 3073 de prueba de profundidad en la canalización 3070 de salida de representación despacha los sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunos ejemplos, la lógica de sombreador de píxeles está incluida en la lógica 3050 de ejecución de hilos. En algunos ejemplos, una aplicación puede evitar el rasterizador y componente 3073 de prueba de profundidad y acceder a datos de vértices no rasterizados a través de una unidad 3023 de salida de tren.

El procesador 3000 gráfico tiene un bus de interconexión, un tejido de interconexión u otros mecanismos de interconexión que permiten que los datos y mensajes pasen entre los componentes principales del procesador. En algunos ejemplos, las unidades 3052A-3052B de ejecución y la(s) caché(s) 3051 asociada(s), el sacamuestras 3054 de medios y texturas y la caché 3058 de texturas/sacamuestras se interconectan mediante un puerto 3056 de datos para llevar a cabo el acceso de memoria y comunicarse con componentes de canalización de salida de representación del procesador. En algunos ejemplos, el sacamuestras 3054, las cachés 3051, 3058 y las unidades 3052A-3052B de ejecución tienen cada uno trayectos de acceso de memoria separados.

En algunos ejemplos, la canalización 3070 de salida de representación contiene un rasterizador y componente 3073 de prueba de profundidad que convierte objetos basados en vértice en una representación basada en píxeles asociada. En algunos ejemplos, la lógica de rasterizador incluye una unidad de ventana/máscara para llevar a cabo la rasterización de línea y triángulo de función fija. Una caché 3078 de representación asociada y una caché 3079 de profundidad también están disponibles en algunos ejemplos. Un componente 3077 de operaciones de píxeles lleva a cabo operaciones basadas en píxeles en los datos, a pesar de que en algunas instancias, las operaciones de píxeles asociadas con operaciones 2D (p. ej., transferencias de imagen de bloque de bits con combinación) son realizadas por el motor 3041 2D, o sustituidas a tiempo de visualización por el controlador 3043 de visualización al utilizar planos de visualización superpuestos. En algunos ejemplos, una caché 3075 de L3 compartida está disponible para todos los componentes gráficos, permitiendo que se compartan datos sin utilizar la memoria de sistema principal.

En algunos ejemplos, la canalización 3030 de medios de procesador gráfico incluye un motor 3037 de medios y un *front end* 3034 de vídeo. En algunos ejemplos, el *front end* 3034 de vídeo recibe comandos de canalización del transmisor de tren 3003 de comandos. En algunos ejemplos, la canalización 3030 de medios incluye un transmisor de tren de comandos separado. En algunos ejemplos, el *front end* 3034 de vídeo procesa comandos de medios antes de enviar el comando al motor 3037 de medios. En algunos ejemplos, el motor 3037 de medios incluye una funcionalidad de generación de hilos para generar hilos para despachar a la lógica 3050 de ejecución de hilos a través del despachador 3031 de hilos.

En algunos ejemplos, el procesador 3000 gráfico incluye un motor 3040 de visualización. En algunos ejemplos, el motor 3040 de visualización es externo al procesador 3000 y se acopla con el procesador gráfico mediante la interconexión 3002 de anillo u otros buses o tejidos de interconexión. En algunos ejemplos, el motor 3040 de visualización incluye un motor 3041 2D y un controlador 3043 de visualización. En algunos ejemplos, el motor 3040 contiene una lógica para fines específicos capaz de operar independientemente de la canalización 3D. En algunos ejemplos, el controlador 3043 de visualización se acopla con un dispositivo de visualización (no se muestra), que puede ser un dispositivo de visualización integrado en el sistema, tal y como en un ordenador portátil, o un dispositivo de visualización externo fijado a través de un conector de dispositivo de visualización.

En algunos ejemplos, la canalización 3020 gráfica y la canalización 3030 de medios son configurables para realizar operaciones basadas en múltiples interfaces de programación gráfica y de medios y no son específicas para ninguna interfaz de programación de aplicación (API). En algunos ejemplos, el software de controlador para el procesador

gráfico traduce las llamadas de API que son específicas para una biblioteca gráfica o de medios particular en comandos que el procesador gráfico puede procesar. En algunos ejemplos, se proporciona apoyo para la biblioteca Open Graphics (OpenGL), lenguaje Open Computing (OpenCL) y/o la API gráfica y de cálculo Vulkan, todos del Grupo Khronos. En algunos ejemplos, también se puede proporcionar apoyo para la biblioteca Direct3D desde Microsoft Corporation. En algunos ejemplos, se puede admitir una combinación de estas bibliotecas. También se puede proporcionar apoyo a la biblioteca Open Source Computer Vision (OpenCV). Una futura API con una canalización 3D compatible también se admitiría si se puede llevar a cabo un mapeo desde la canalización de la futura API a la canalización del procesador gráfico.

Programación de canalización gráfica

La Figura 31A es un diagrama de bloque que ilustra un formato 3100 de comandos del procesador gráfico según algunos ejemplos. La Figura 31B es un diagrama de bloque que ilustra una secuencia 3110 de comandos del procesador gráfico según un ejemplo. Las casillas de líneas continuas en la Figura 31A ilustran los componentes que están generalmente incluidos en un comando gráfico, mientras que las líneas discontinuas incluyen componentes que son opcionales o que se incluyen únicamente en un subconjunto de los comandos gráficos. El formato 3100 de comandos del procesador gráfico de ejemplo de la Figura 31A, incluye campos de datos para identificar un cliente 3102 diana del comando, un código 3104 de operación de comandos y los datos 3106 relevantes para el comando. También se incluyen un subcódigo de operación 3105 y un tamaño 3108 de comando en algunos comandos.

En algunos ejemplos, el cliente 3102 especifica la unidad de cliente del dispositivo gráfico que procesa los datos de comandos. En algunos ejemplos, un analizador de comandos de procesador gráfico examina el campo de cliente de cada comando para condicionar el procesamiento adicional del comando y encaminar los datos de comandos a la unidad de cliente adecuada. En algunos ejemplos, las unidades de cliente de procesador gráfico incluyen una unidad de interfaz de memoria, una unidad de representación, 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 la unidad de cliente recibe el comando, la unidad de cliente lee el código de operación 3104 y, si está presente, el subcódigo de operación 3105 para determinar la operación a llevar a cabo. La unidad de cliente lleva a cabo el comando al utilizar información en el campo 3106 de datos. Para algunos comandos, se espera que un tamaño 3108 de comandos explícito especifique el tamaño del comando. En algunos ejemplos, el analizador de comandos automáticamente determina el tamaño de al menos algunos de los comandos en base al código de operación del comando. En algunos ejemplos los comandos se alinean a través de múltiples de un Double Word.

El diagrama de flujo en la Figura 31B muestra una secuencia 3110 de comandos de procesador gráfico de ejemplo. En algunos ejemplos, el software o el firmware de un sistema de procesamiento de datos que caracteriza un ejemplo de un procesador gráfico utiliza una versión de la secuencia de comandos que se muestra para establecer, ejecutar y finalizar un conjunto de operaciones gráficas. Una secuencia de comandos de muestra se muestra y describe únicamente a efectos de ejemplo ya que los ejemplos no se limitan a estos comandos específicos o a esta secuencia de comandos. Además, los comandos se pueden emitir como lote de comandos en una secuencia de comandos, de modo que el procesador gráfico procesará la secuencia de comandos simultáneamente al menos en parte.

En algunos ejemplos, la secuencia 3110 de comandos del procesador gráfico puede comenzar con un comando 3112 de evacuación de canalización para hacer que cualquier canalización gráfica activa complete los comandos pendientes en el momento para la canalización. En algunos ejemplos, la canalización 3122 3D y la canalización 3124 de medios no operan simultáneamente. La evacuación de canalización se lleva a cabo para hacer que la canalización gráfica activa complete cualquier comando pendiente. En respuesta a una evacuación de canalización, el analizador de comandos para el procesador gráfico pausará el procesamiento de comandos hasta que los motores de representación activos completen las operaciones pendientes y las cachés de lectura relevantes se invaliden. Opcionalmente, cualquier dato en la caché de representación que está marcada como "sucia" se puede evacuar a la memoria. En algunos ejemplos, el comando 3112 de evacuación de canalización se puede utilizar para sincronizar las canalizaciones o antes de colocar el procesador gráfico en un estado de baja potencia.

En algunos ejemplos, un comando 3113 de selección de canalización se utiliza cuando una secuencia de comandos requiere que el procesador gráfico conmute de forma explícita entre canalizaciones. En algunos ejemplos, un comando 3113 de selección de canalización se requiere únicamente en un contexto de ejecución antes de emitir comandos de canalización a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunos ejemplos, un comando 3112 de evacuación de canalización se requiere inmediatamente antes de que una canalización conmute a través del comando 3113 de selección de canalización.

En algunos ejemplos, un comando 3114 de control de canalización configura una canalización gráfica para su operación y se utiliza para programar la canalización 3122 3D y la canalización 3124 de medios. En algunos ejemplos, el comando 3114 de control de canalización configura el estado de canalización para la canalización activa. En un ejemplo, el comando 3114 de control de canalización se utiliza para sincronizar las canalizaciones y para despejar datos de una o más memorias caché en la canalización activa antes de procesar un lote de comandos.

En algunos ejemplos, los comandos 3116 de estado de memoria intermedia de retorno se utilizan para configurar un conjunto de memorias intermedias de retorno para las canalizaciones respectivas para escribir datos. Algunas

operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las cuales las operaciones escriben datos intermedios durante el procesamiento. En algunos ejemplos, el procesador gráfico también utiliza una o más memorias intermedias de retorno para almacenar datos de salida y para llevar a cabo una comunicación entre hilos. En algunos ejemplos, el estado 3116 de memoria intermedia de retorno incluye seleccionar el tamaño y número de las memorias intermedias de retorno para un conjunto de operaciones de canalización.

Los comandos restantes en la secuencia de comandos difieren en base a las canalizaciones activas para operaciones. En base a la determinación 3120 de canalización, la secuencia de comandos está adaptada a la canalización 3122 3D que comienza con el estado 3130 de canalización 3D o la canalización 3124 de medios que comienza en el estado 3140 de canalización de medios.

Los comandos para configurar el estado 3130 de canalización 3D incluyen comandos de ajuste del estado 3D para el estado de la memoria intermedia de vértices, estado de elemento de vértices, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que se han de configurar antes de que se procesen los comandos de primitivas 3D. Los valores de estos comandos se determinan al menos en parte en base a la API 3D particular en uso. En algunos ejemplos, los comandos de estado 3130 de canalización 3D son también capaces de deshabilitar o evitar de forma selectiva determinados elementos de canalización si dichos elementos no se van a utilizar.

En algunos ejemplos, el comando de primitiva 3132 3D se utiliza para ingresar primitivas 3D a procesar por la canalización 3D. Los comandos y los parámetros asociados que se pasan al procesador gráfico a través del comando de primitiva 3132 3D se envían a la función de captura de vértices en la canalización gráfica. La función de captura de vértices utiliza los datos del comando de primitiva 3132 3D 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, el comando de primitiva 3132 3D se utiliza para realizar operaciones de vértices en primitivas 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la canalización 3122 3D despacha hilos de ejecución de sombreado a las unidades de ejecución del procesador gráfico.

En algunos ejemplos, la canalización 3122 3D se activa mediante un evento o comando de ejecución 3134. En algunos ejemplos, un registro de escritura activa la ejecución de comandos. En algunos ejemplos la ejecución se activa a través de un comando "iniciar" o "impulsar" en la secuencia de comandos. En un ejemplo, la ejecución de comandos se activa al utilizar un comando de sincronización de canalizaciones para evacuar la secuencia de comandos a través de la canalización gráfica. La canalización 3D llevará a cabo el procesamiento de geometría para las primitivas 3D. Una vez se completan las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles colorea los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones *back end* de píxeles.

En algunos ejemplos, la secuencia 3110 de comandos de procesador gráfico sigue el trayecto de la canalización 3124 de medios cuando lleva a cabo operaciones de medios. En general, el uso y manera específicos de programación para la canalización 3124 de medios depende de las operaciones de medios o de cálculos a realizar. Las operaciones de descodificación de medios específicas se pueden descargar a la canalización de medios durante la descodificación de medios. En algunos ejemplos, la canalización de medios se puede también evitar y la descodificación de medios se puede llevar a cabo por completo o en parte al utilizar recursos proporcionados por uno o más núcleos de procesamiento para fines generales. En un ejemplo, la canalización de medios también incluye elementos para operaciones de la unidad de procesador gráfico para fines generales (GPGPU), donde el procesador gráfico se utiliza para realizar operaciones de vector SIMD al utilizar programas de sombreado de cálculo que no están relacionados de forma explícita con la representación de primitivas gráficas.

En algunos ejemplos, la canalización 3124 de medios está configurada de una manera similar a la canalización 3122 3D. Un conjunto de comandos para configurar el estado 3140 de la canalización de medios se despacha o coloca en una cola de comandos antes de los comandos 3142 de objeto de medios. En algunos ejemplos, los comandos 3140 de estado de canalización de medios incluyen datos para configurar los elementos de canalización de medios que se utilizarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de codificación de vídeo y descodificación de vídeo en la canalización de medios, tal y como el formato de codificación o descodificación. En algunos ejemplos, los comandos 3140 de estado de canalización de medios también apoyan el uso de uno o más punteros para elementos de estado "indirectos" que contienen un lote de configuraciones de estado.

En algunos ejemplos, los comandos 3142 de objeto de medios proporcionan punteros a los objetos de medios para que la canalización de medios lleve a cabo el procesamiento. Los objetos de medios incluyen memorias intermedias que contienen datos de vídeo a procesar. En algunos ejemplos, todos los estados de canalización de medios deben ser válidos antes de emitir un comando 3142 de objeto de medios. Una vez se ha configurado el estado de canalización y se han puesto en cola los comandos 3142 de objeto de medios, la canalización 3124 de medios se activa a través de un comando 3144 de ejecución o un evento de ejecución equivalente (p. ej., registro de escritura). Entonces, se puede postprocesar la salida de la canalización 3124 de medios mediante operaciones proporcionadas por la canalización 3122 3D o la canalización 3124 de medios. En algunos ejemplos, las operaciones de GPGPU se configuran y ejecutan de una manera similar a las operaciones de medios.

Arquitectura de software gráfica

La Figura 32 ilustra una arquitectura de software grafica de ejemplo para un sistema 3200 de procesamiento de datos según algunos ejemplos. En algunos ejemplos, la arquitectura de software incluye una aplicación 3210 gráfica 3D, un sistema 3220 operativo y al menos un procesador 3230. En algunos ejemplos, el procesador 3230 incluye un procesador 3232 gráfico y uno o más núcleo(s) 3234 de procesador para fines generales. La aplicación 3210 gráfica y el sistema 3220 operativo se ejecutan cada uno en la memoria 3250 de sistema del sistema de procesamiento de datos.

En algunos ejemplos, la aplicación 3210 gráfica 3D contiene uno o más programas de sombreado que incluyen instrucciones 3212 de sombreado. Las instrucciones de lenguaje de sombreado pueden estar en un lenguaje de sombreado de alto nivel, tal y como el lenguaje de sombreado de alto nivel (HLSL) o el lenguaje de sombreado OpenGL (GLSL). La aplicación también incluye instrucciones 3214 ejecutables en un lenguaje automático apropiado para que el núcleo 3234 de procesador para fines generales lo ejecute. La aplicación también incluye objetos 3216 gráficos definidos por datos de vértices.

En algunos ejemplos, el sistema 3220 operativo es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo tipo UNIX privativo, o un sistema operativo tipo UNIX de fuente abierta que utiliza una variante del *kernel* Linux. El sistema 3220 operativo puede admitir una API 3222 gráfica tal y como la API Direct3D, la API OpenGL o la API Vulkan. Cuando la API Direct3D está en uso, el sistema 3220 operativo utiliza un compilador 3224 de sombreado *front end* para compilar cualquier instrucción 3212 de sombreado en HLSL en un lenguaje de sombreado de menor nivel. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede llevar a cabo una precompilación de sombreado. En algunos ejemplos, los sombreadores de alto nivel se compilan en sombreadores de bajo nivel durante la compilación de la aplicación 3210 gráfica 3D. En algunos ejemplos, las instrucciones 3212 de sombreado se proporcionan en una forma intermedia, tal y como una versión de representación intermedia portable estándar (SPIR) utilizada por la API Vulkan.

En algunos ejemplos, el controlador 3226 gráfico de modo usuario contiene un compilador 3227 de sombreado *back end* para convertir las instrucciones 3212 de sombreado en una representación específica de hardware. Cuando la API OpenGL está en uso, las instrucciones 3212 de sombreado en el lenguaje de alto nivel GLSL se pasan a un controlador 3226 gráfico de modo usuario para su compilación. En algunos ejemplos, el controlador 3226 gráfico de modo usuario utiliza funciones 3228 del modo sistema operativo *kernel* para comunicarse con un controlador 3229 gráfico de modo *kernel*. En algunos ejemplos, el controlador 3229 gráfico de modo *kernel* se comunica con el procesador 3232 gráfico para despachar comandos e instrucciones.

Implementaciones del núcleo PI

Uno o más aspectos de al menos un ejemplo se pueden implementar por el código representativo almacenado en un medio legible por máquina que representa y/o define lógica en un circuito integrado tal y como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan diversas lógicas en el procesador. Cuando las lee una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para que lleve a cabo las técnicas descritas en la presente memoria. Dichas representaciones, conocidas como "núcleos PI", son unidades reutilizables de lógica para un circuito integrado que puede estar almacenado en un medio tangible y legible por máquina como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware se puede proporcionar 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 puede estar fabricado de modo que el circuito lleva a cabo operaciones descritas en asociación con cualquiera de los ejemplos descritos en la presente memoria.

La Figura 33 es un diagrama de bloque que ilustra un sistema 3300 de desarrollo de núcleo PI que se puede utilizar para fabricar un circuito integrado para que lleve a cabo operaciones según un ejemplo. El sistema 3300 de desarrollo de núcleo PI se puede utilizar para generar diseños modulares y reutilizables que se pueden incorporar en un diseño mayor o se pueden utilizar para construir un circuito integrado completo (p. ej., un circuito integrado SOC). Una instalación 3330 de diseño puede generar una simulación 3310 de software de un diseño de núcleo PI en un lenguaje de programación de alto nivel (p. ej., C/C++). La simulación 3310 de software se puede utilizar para diseñar, probar y verificar el comportamiento del núcleo PI al utilizar un modelo 3312 de simulación. El modelo 3312 de simulación puede incluir simulaciones funcionales, de comportamiento y/o de tiempo. Entonces, se puede crear o sintetizar un diseño 3315 de nivel de transferencia de registro (RTL) desde el modelo 3312 de simulación. El diseño 3315 de RTL es una abstracción del comportamiento del circuito integrado que modela el flujo de las señales digitales entre los registros de hardware, incluida la lógica asociada llevada a cabo al utilizar las señales digitales modeladas. Además de un diseño 3315 de RTL, también se pueden crear, diseñar o sintetizar diseños de menor nivel en el nivel de lógica o nivel de transistor. Por tanto, los detalles particulares del diseño inicial y la simulación pueden variar.

El diseño 3315 de RTL o equivalente puede además ser sintetizado por la instalación de diseño en un modelo 3320 de hardware, que puede estar en un lenguaje de descripción de hardware (HDL), o alguna otra representación de datos de diseño físicos. El HDL puede además simularse o probarse para verificar el diseño del núcleo PI. El diseño del núcleo PI se puede almacenar para administrarlo a una instalación 3365 de fabricación de terceros al utilizar una memoria 3340 no volátil (p. ej., disco duro, memoria flash o cualquier medio de almacenamiento no volátil). De manera

alternativa, el diseño de núcleo PI se puede transmitir (p. ej., a través de Internet) por una conexión 3350 por cable o por una conexión 3360 inalámbrica. La instalación 3365 de fabricación puede entonces fabricar un circuito integrado que se basa al menos en parte en el diseño del núcleo PI. El circuito integrado fabricado puede estar configurado para realizar operaciones según al menos un ejemplo descrito en la presente memoria.

Sistema en un circuito de chip integrado de ejemplo

Las Figuras 34-36 ilustran circuitos integrados de ejemplo y procesadores gráficos asociados que se pueden fabricar al utilizar uno o más núcleos PI, según diversos ejemplos descritos en la presente memoria. Además de lo que se ilustra, se pueden incluir otras lógicas y circuitos, incluidos núcleos/procesadores gráficos adicionales, controladores de interfaz periféricos o núcleos de procesador para fines generales.

La Figura 34 es un diagrama de bloque que ilustra un sistema de ejemplo en circuito 3400 de chip integrado que se puede fabricar utilizando uno o más núcleos PI, según un ejemplo. El circuito 3400 integrado de ejemplo incluye uno o más procesadores 3405 de aplicación (p. ej., CPU), al menos un procesador 3410 gráfico y puede además incluir un procesador 3415 de imágenes y/o un procesador 3420 de vídeo, cualquiera de los cuales puede ser un núcleo PI modular de la misma o de múltiples instalaciones de diseño diferentes. El circuito 3400 integrado incluye lógica periférica o de bus que incluye un controlador 3425 de USB, un controlador 3430 UART, un controlador 3435 SPI/SDIO y un controlador 3440 I^2S/I^2C. Adicionalmente, el circuito integrado puede incluir un dispositivo 3445 de visualización acoplado a uno o más de un controlador 3450 de interfaz multimedia de alta definición (HDMI) y una interfaz 3455 de visualización de interfaz de procesador de industria móvil (MIPI). El almacenamiento puede proporcionarlo un subsistema 3460 de memoria flash que incluye una memoria flash y un controlador de memoria flash. La interfaz de memoria puede estar proporcionada a través de un controlador 3465 de memoria para acceder a los dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen además un motor 3470 de seguridad incorporado.

La Figura 35 es un diagrama de bloque que ilustra un procesador 3510 gráfico de ejemplo de un sistema en un circuito de chip integrado que se puede fabricar utilizando uno o más núcleos PI, según un ejemplo. El procesador 3510 gráfico puede ser una variante del procesador 3410 gráfico de la Figura 34**.** El procesador 3510 gráfico incluye un procesador 3505 de vértices y uno o más procesadores 3515A-3515N de fragmentos (p. ej., 3515A, 3515B, 3515C, 3515D, hasta 3515N-1, y 3515N). El procesador 3510 gráfico puede ejecutar diferentes programas de sombreado a través de una lógica separada, de modo que el procesador 3505 de vértices se optimiza para ejecutar operaciones para programas de sombreado de vértices, mientras que el uno o más procesadores 3515A-3515N de fragmentos ejecutan operaciones de sombreado de fragmento (p. ej., píxeles) para programas de sombreado de píxeles o fragmentos. El procesador 3505 de vértices lleva a cabo la etapa de procesamiento de vértices de la canalización gráfica 3D y genera datos de vértices y primitivas. El(los) procesador(es) 3515A-3515N de fragmentos utiliza(n) los datos de vértices y primitivas generados por el procesador 3505 de vértices para producir un *framebuffer* que se muestra en el dispositivo de visualización. En un ejemplo, el(los) procesador(es) 3515A-3515N de fragmentos se optimiza(n) para ejecutar programas de sombreado de fragmentos tal y como se permite en la API OpenGL, que se puede utilizar para llevar a cabo operaciones similares como un programa de sombreado de píxeles tal y como se proporciona en la API Direct 3D.

El procesador 3510 gráfico además incluye una o más unidades 3520A-3520B de gestión de memoria (MMU), cachés 3525A-3525B e interconexiones 3530A-3530B de circuito. La una o más MMU 3520A-3520B permiten el mapeo de direcciones virtuales a físicas para el procesador 3510 gráfico, incluido para el procesador 3505 de vértices y/o procesador(es) 3515A-3515N de fragmentos, que pueden referenciar datos de imágenes/texturas o de vértices almacenados en la memoria, además de datos de imágenes/texturas o de vértices almacenados en la una o más cachés 3525A-3525B. En un ejemplo la una o más MMU 3520A-3520B pueden estar sincronizadas con otras MMU en el sistema, incluidas una o más MMU asociadas con el uno o más procesadores 3405 de aplicaciones, procesador 3415 de imágenes y/o procesador 3420 de vídeos de la Figura 34, de modo que cada procesador 3405-3420 puede participar en un sistema de memoria virtual compartido o unificado. La una o más interconexiones 3530A-3530B de circuito permiten que el procesador 3510 gráfico interactúe con otros núcleos PI en el SoC, o bien a través de un bus interno del SoC o a través de conexión directa, según los ejemplos.

La Figura 36 es un diagrama de bloque que ilustra un procesador 3610 gráfico de ejemplo adicional de un sistema en un circuito de chip integrado que se puede fabricar utilizando uno o más núcleos PI, según un ejemplo. El procesador 3610 gráfico puede ser una variante del procesador 3410 gráfico de la Figura 34. El procesador 3610 gráfico incluye la una o más MMU 3520A-3520B, cachés 3525A-3525B e interconexiones 3530A-3530B de circuito del circuito 3500 integrado de la Figura 35.

El procesador 3610 gráfico incluye uno o más núcleos 3615A-3615N de sombreado (p. ej., 3615A, 3615B, 3615C, 3615D, 3615E, 3615F, hasta 3615N-1 y 3615N), que proporciona a una arquitectura de núcleo de sombrado unificada en la cual un único núcleo o tipo o núcleo puede ejecutar todos los tipos de códigos de sombreado programables, incluido el código de programa de sombreado 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. Adicionalmente, el procesador 3610 gráfico incluye un gestor 3605 de tareas entre núcleos, que actúa como un despachador de hilos para despachar los hilos de ejecución a uno o más núcleos 3615A-3615N de sombreado y una unidad 3618 de enlosamiento para acelerar las operaciones de enlosamiento para la

representación basada en losa, en la cual las operaciones de reproducción para una escena se subdividen en un espacio de imagen, por ejemplo para explotar la coherencia espacial local en una escena o para optimizar el uso de cachés internas.

Los ejemplos pueden incluir materia tal y como un método, medios para realizar actuaciones del método, al menos un medio legible por máquina que incluya instrucciones que, cuando se llevan a cabo por una máquina, hacen que la máquina lleve a cabo actuaciones del método, o de un aparato o sistema según las realizaciones y ejemplos descritos en la presente memoria. Diversos componentes pueden ser un medio para llevar a cabo las operaciones o funciones descritas.

Las realizaciones descritas en la presente memoria se refieren a configuraciones de hardware específicas, tal y como circuitos integrados específicos de la aplicación (ASIC), configurados para llevar a cabo determinadas operaciones o con una funcionalidad predeterminada. Dichos dispositivos electrónicos suelen incluir un conjunto de uno o más procesadores acoplados a uno o más componentes distintos, tal y como uno o más dispositivos de almacenamiento (medios de almacenamiento legibles por máquina no transitorios), dispositivos de entrada/salida de usuario (p. ej., un teclado, una pantalla táctil y/o una pantalla) y conexiones de red. El acople del conjunto de procesadores y otros componentes suele ser a través de uno o más buses o puentes (también denominados controladores de bus). El dispositivo de almacenamiento y las señales que llevan el tráfico de red representan respectivamente uno o más medios de almacenamiento legibles por máquina y medios de comunicación legibles por máquina. Por tanto, los dispositivos de almacenamiento de un dispositivo electrónico dado suelen almacenar códigos y/o datos para su ejecución en el conjunto de uno o más procesadores de dicho dispositivo electrónico.

Por supuesto, una o más partes de una realización pueden implementarse utilizando diferentes combinaciones de software, firmware y/o hardware. A lo largo de esta descripción detallada, con fines explicativos, se establecieron numerosos detalles específicos para proporcionar una comprensión completa de la presente invención. Será evidente, sin embargo, para un experto en la técnica que las realizaciones pueden ponerse en práctica sin algunos de estos detalles específicos. En determinados casos, las estructuras y funciones bien conocidas no se describieron con detalles elaborados para evitar oscurecer el tema inventivo de las realizaciones.

REIVINDICACIONES

1. Una unidad de procesamiento de gráficos de fines generales que comprende:

una unidad de coma flotante de precisión dinámica (1400) que incluye una unidad de control (1402) que tiene lógica de hardware de seguimiento de precisión (1412) para rastrear un número disponible de bits de precisión para datos calculados en relación con una precisión objetivo, en donde unidad de coma flotante de precisión dinámica (1400) incluye lógica computacional para generar datos con múltiples precisiones,

caracterizada porque la unidad de coma flotante de precisión dinámica (1400) incluye:

un conjunto de registros (1404) para almacenar datos de entrada y datos intermedios en múltiples precisiones,

un bloque de significando (1408) para realizar una porción de significando de un cálculo de coma flotante, incluyendo el bloque de significando (1408) que incluye un sumador de precisión dinámica (1438) configurable para sumar o restar datos de entrada en múltiples precisiones, y

un bloque de exponente (1406) para realizar una porción de exponente de un cálculo de coma flotante, incluyendo el bloque de exponente (1406) que un sumador de precisión dinámica (1426) configurable para sumar o restar exponentes de datos de entrada en múltiples precisiones,

en donde el bloque de exponente (1406) y el bloque de significando (1408) deben realizar una primera operación de coma flotante para emitir un primer valor de salida que tiene 16 bits de precisión,

en donde el bloque de exponente (1406) y el bloque de significando (1408) deben realizar una segunda operación para emitir un segundo valor de salida con 32 bits de precisión, y

en donde el bloque de exponente (1406) y el bloque de significando (1408) deben realizar una tercera operación de coma flotante en datos de entrada que tienen valores de 32 bits para emitir un tercer valor de salida que tiene un tipo de datos de 32 bits, siendo el tercer valor de salida generado en 16 bits de precisión.

2. La unidad de procesamiento de gráficos de fines generales según la reivindicación 1, en donde el conjunto de registros (1404) incluye un acumulador de errores (1434) para rastrear un error acumulado en un conjunto de operaciones de coma flotante.

3. La unidad de procesamiento de gráficos de fines generales según la reivindicación 1, incluyendo el bloque de significando (1408) un multiplicador de precisión dinámica (1418) configurable para sumar, multiplicar o dividir datos de entrada en múltiples precisiones.

4. La unidad de procesamiento de gráficos de fines generales según la reivindicación 1, incluyendo el bloque de exponente (1406) un multiplicador de 8 bits, y en donde el bloque de exponente (1406) y el bloque de significando (1408) son configurables para realizar una operación de enteros de 8 bits dual.

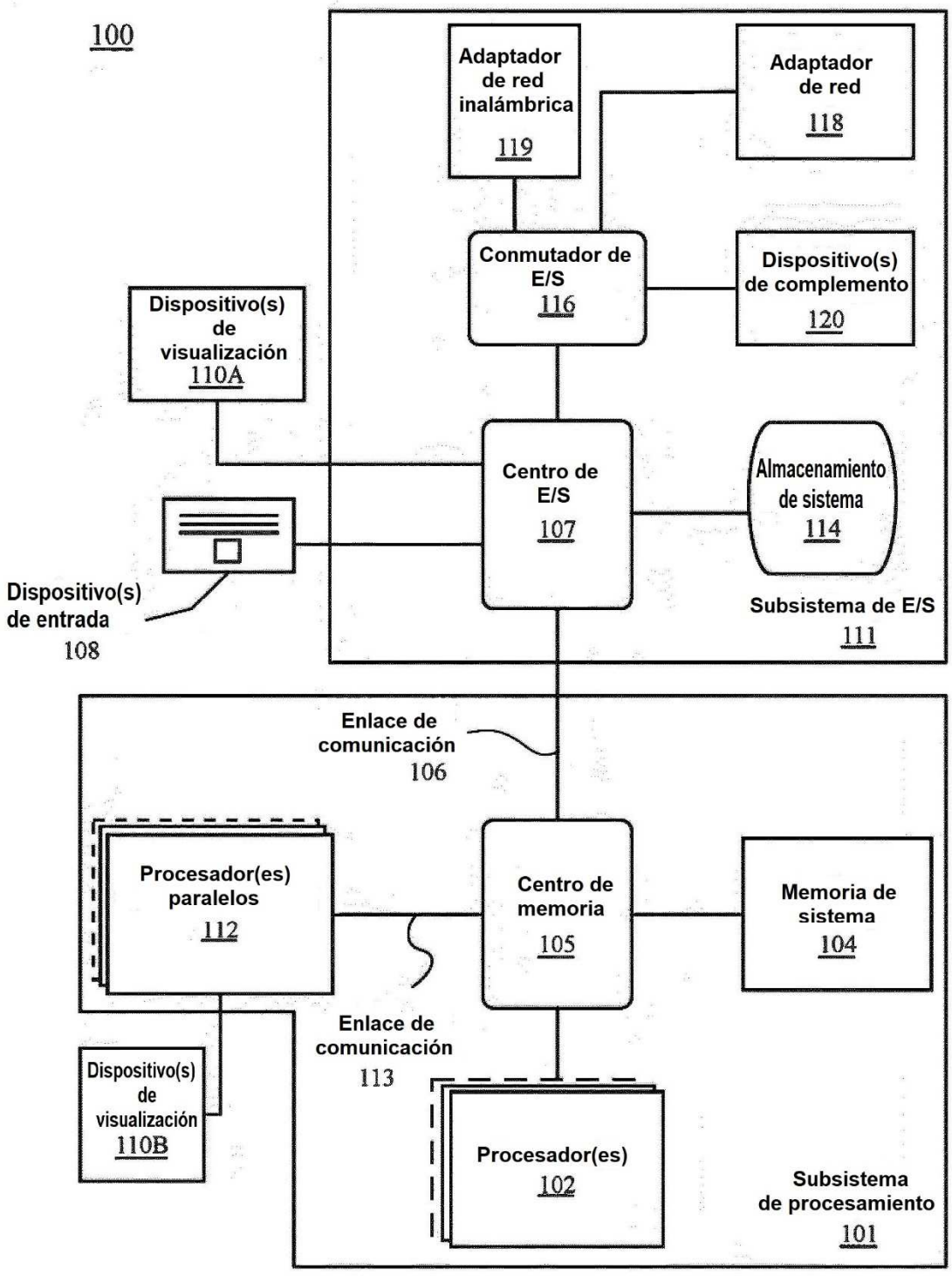

100
Adaptador de red inalámbrica 119
Adaptador de red 118
Conmutador de E/S 116
Dispositivo(s) de complemento 120
Dispositivo(s) de visualización 110A
Centro de E/S 107
Almacenamiento de sistema 114
Dispositivo(s) de entrada 108
Subsistema de E/S 111
Enlace de comunicación 106
Procesador(es) paralelos 112
Centro de memoria 105
Memoria de sistema 104
Enlace de comunicación 113
Dispositivo(s) de visualización 110B
Procesador(es) 102
Subsistema de procesamiento 101

FIG. 1

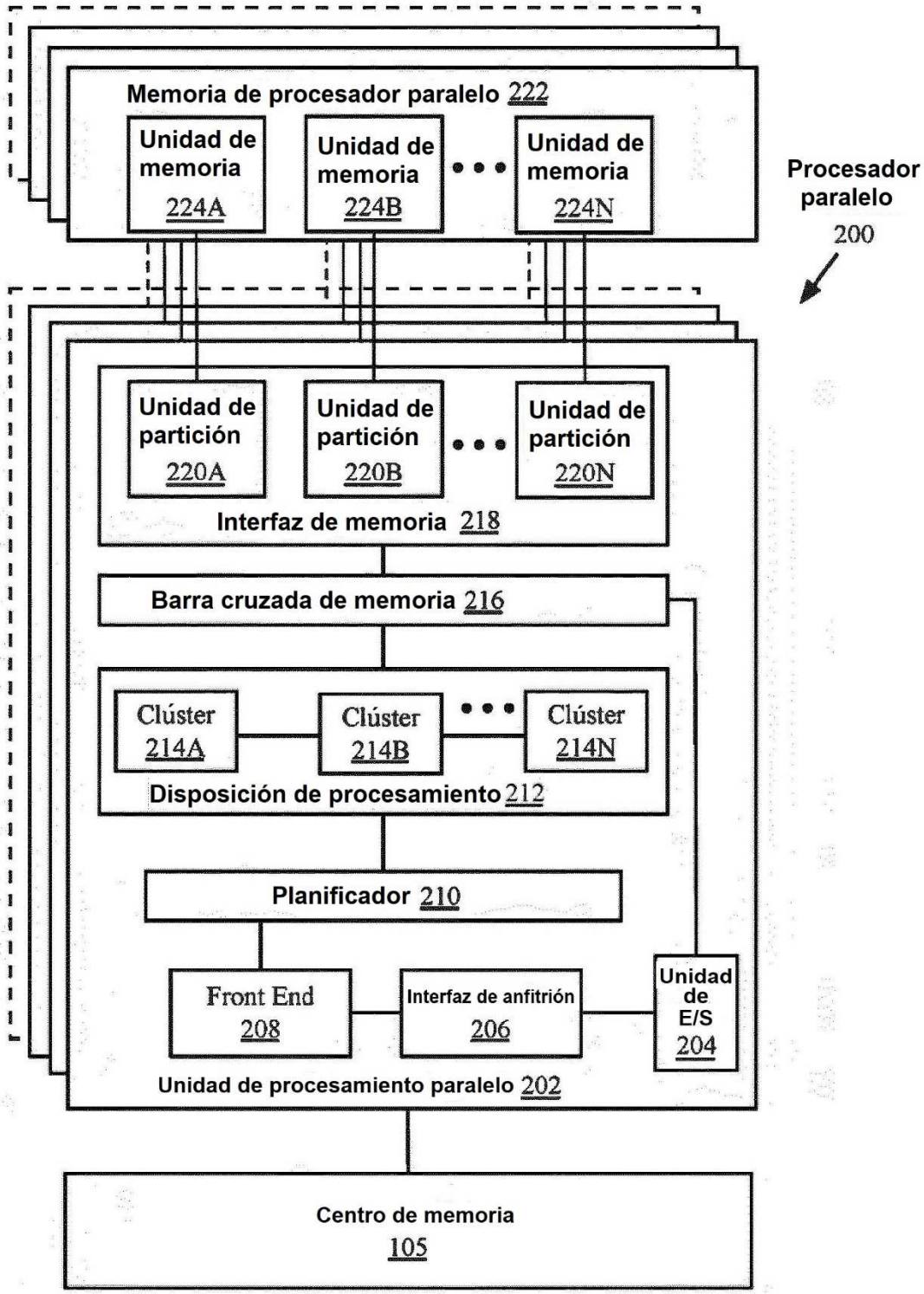

Memoria de procesador paralelo 222
Unidad de memoria 224A
Unidad de memoria 224B
Unidad de memoria 224N
Procesador paralelo 200
Unidad de partición 220A
Unidad de partición 220B
Unidad de partición 220N
Interfaz de memoria 218
Barra cruzada de memoria 216
Clúster 214A
Clúster 214B
Clúster 214N
Disposición de procesamiento 212
Planificador 210
Front End 208
Interfaz de anfitrión 206
Unidad de E/S 204
Unidad de procesamiento paralelo 202
Centro de memoria 105

FIG. 2A

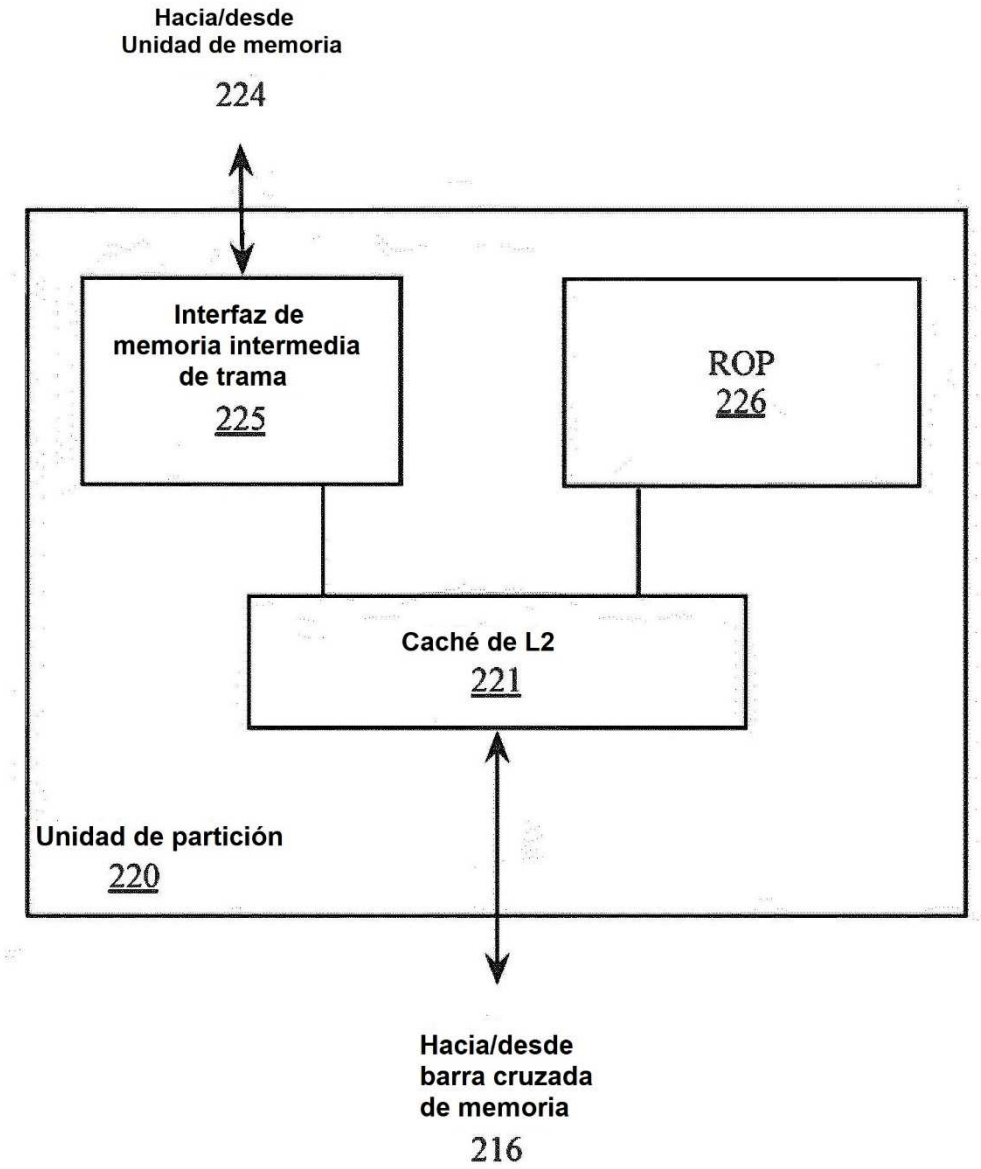

Hacia/desde
Unidad de memoria
224
Interfaz de
memoria intermedia
de trama
225
ROP
226
Caché de L2
221
Unidad de partición
220
Hacia/desde
barra cruzada
de memoria
216

FIG. 2B

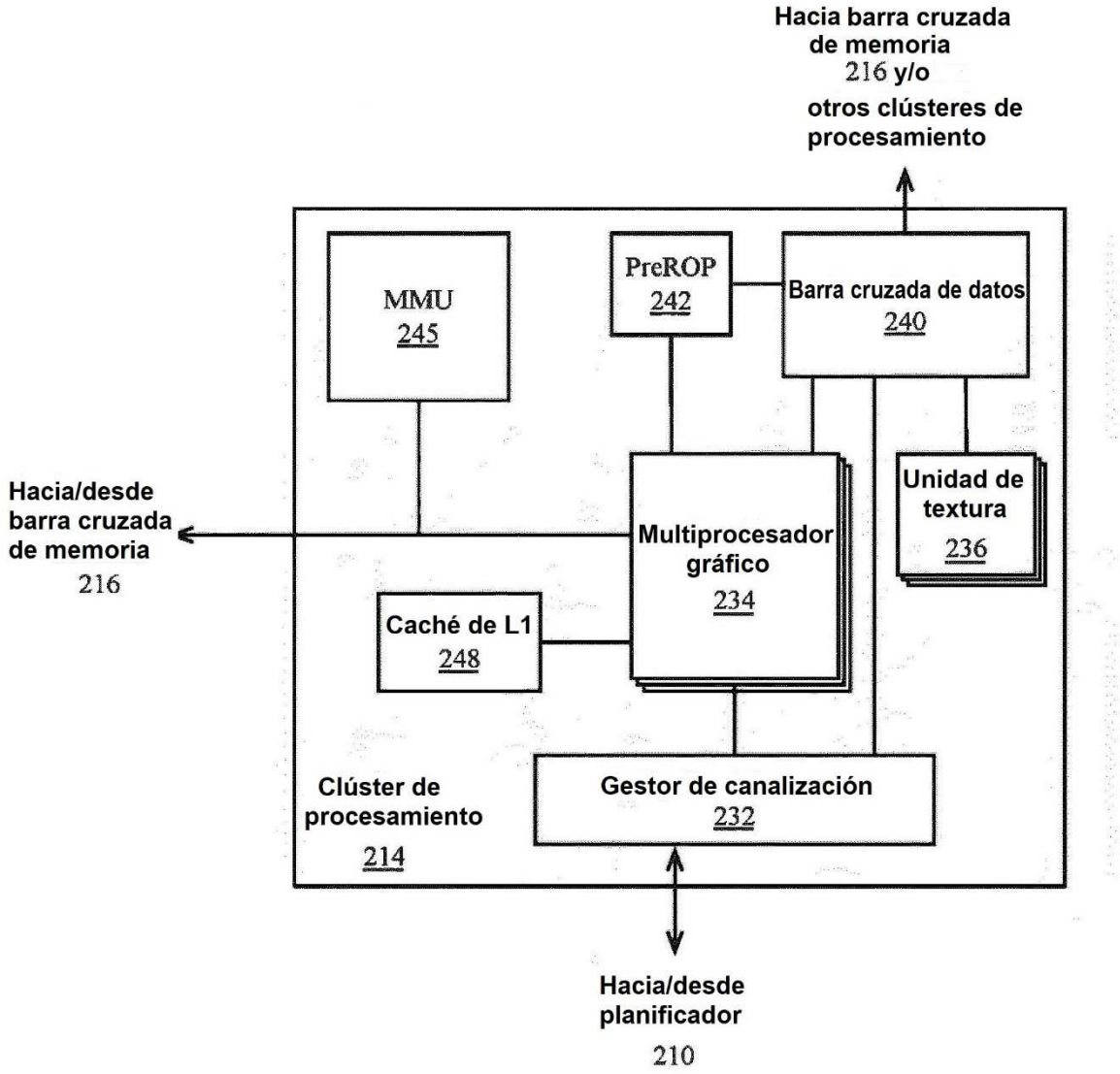

Hacia barra cruzada de memoria 216 y/o otros clústeres de procesamiento
MMU 245
PreROP 242
Barra cruzada de datos 240
Hacia/desde barra cruzada de memoria 216
Multiprocesador gráfico 234
Unidad de textura 236
Caché de L1 248
Clúster de procesamiento 214
Gestor de canalización 232
Hacia/desde planificador 210

FIG. 2C

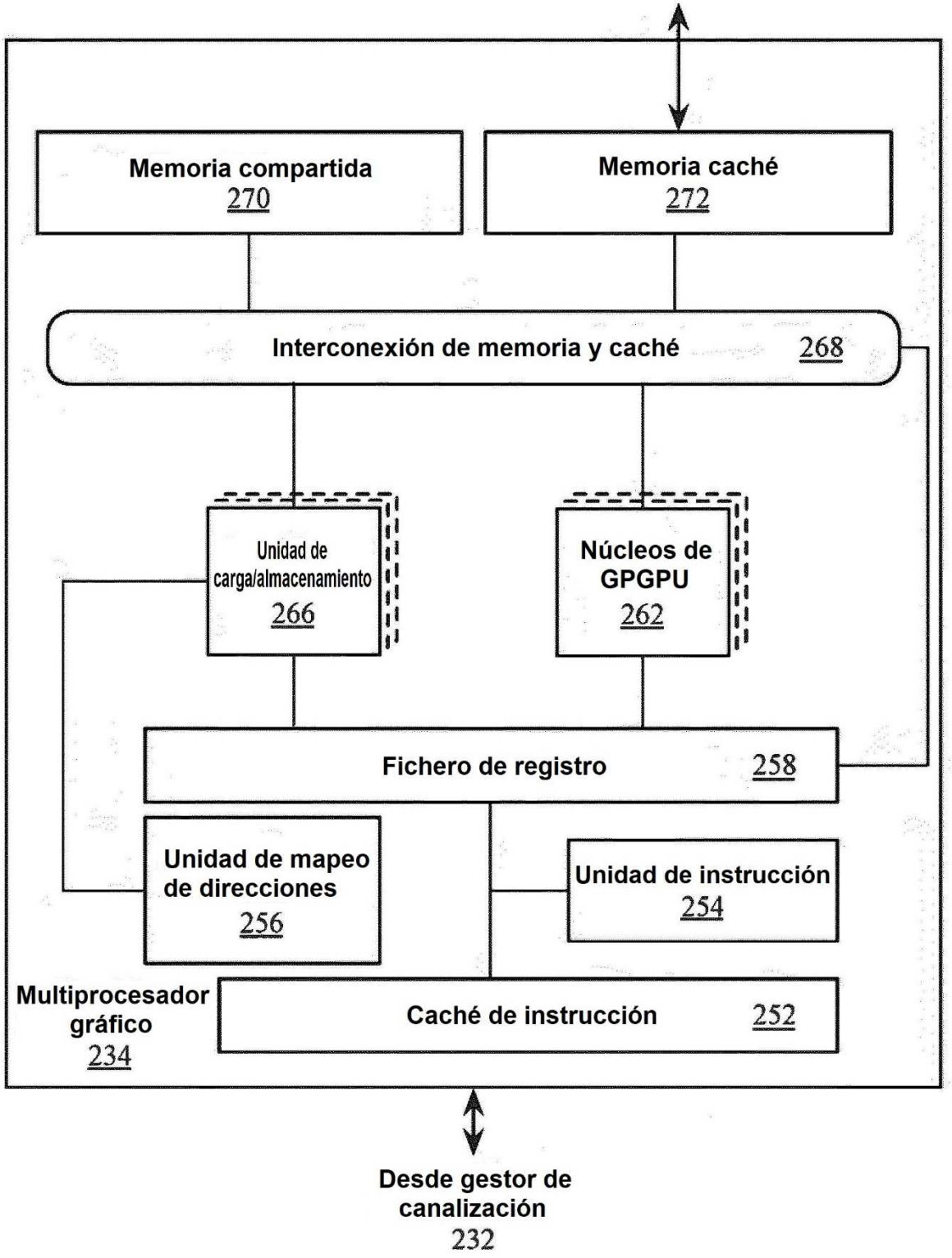

Memoria compartida
270
Memoria caché
272
Interconexión de memoria y caché
268
Unidad de carga/almacenamiento
266
Núcleos de GPGPU
262
Fichero de registro
258
Unidad de mapeo de direcciones
256
Unidad de instrucción
254
Multiprocesador gráfico
234
Caché de instrucción
252
Desde gestor de canalización
232

FIG. 2D


Multiprocesador gráfico 325
Tejido de interconexión 327
Memoria compartida 346
Unidad(es) de textura 344A
Unidad(es) de textura 344B
Memoria caché 342
Unidad de carga/almacenamiento 340A
Núcleo de GPGPU 338A
Núcleo de GPGPU 337A
Núcleo de GPGPU 336A
Unidad de carga/almacenamiento 340B
Núcleo de GPGPU 338B
Núcleo de GPGPU 337B
Núcleo de GPGPU 336B
Fichero de registro 334A
Fichero de registro 334B
Unidad de instrucción 332A
Unidad de instrucción 332B
Caché de instrucción 330

FIG. 3A

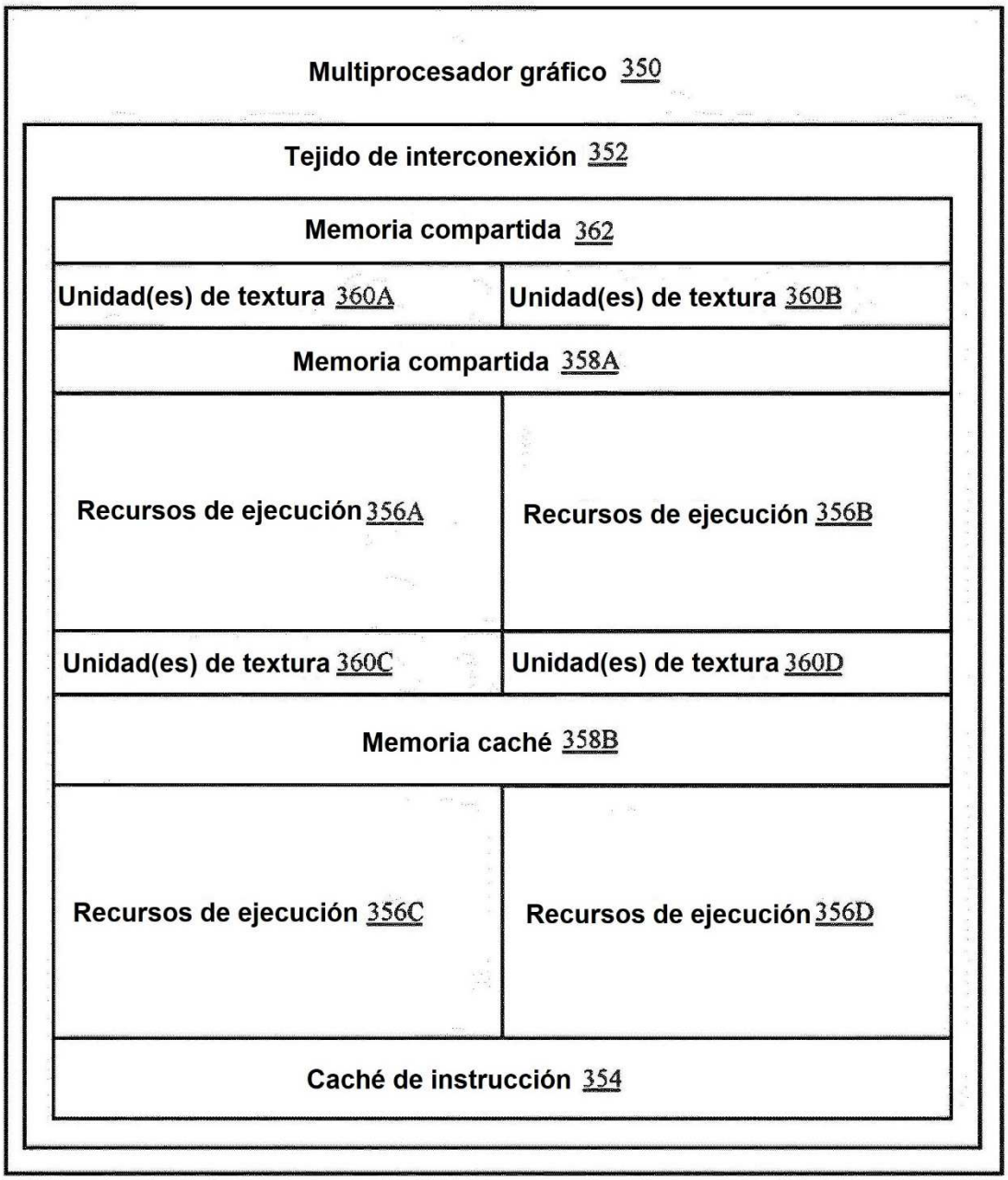

Multiprocesador gráfico 350
Tejido de interconexión 352
Memoria compartida 362
Unidad(es) de textura 360A
Unidad(es) de textura 360B
Memoria compartida 358A
Recursos de ejecución 356A
Recursos de ejecución 356B
Unidad(es) de textura 360C
Unidad(es) de textura 360D
Memoria caché 358B
Recursos de ejecución 356C
Recursos de ejecución 356D
Caché de instrucción 354

FIG. 3B

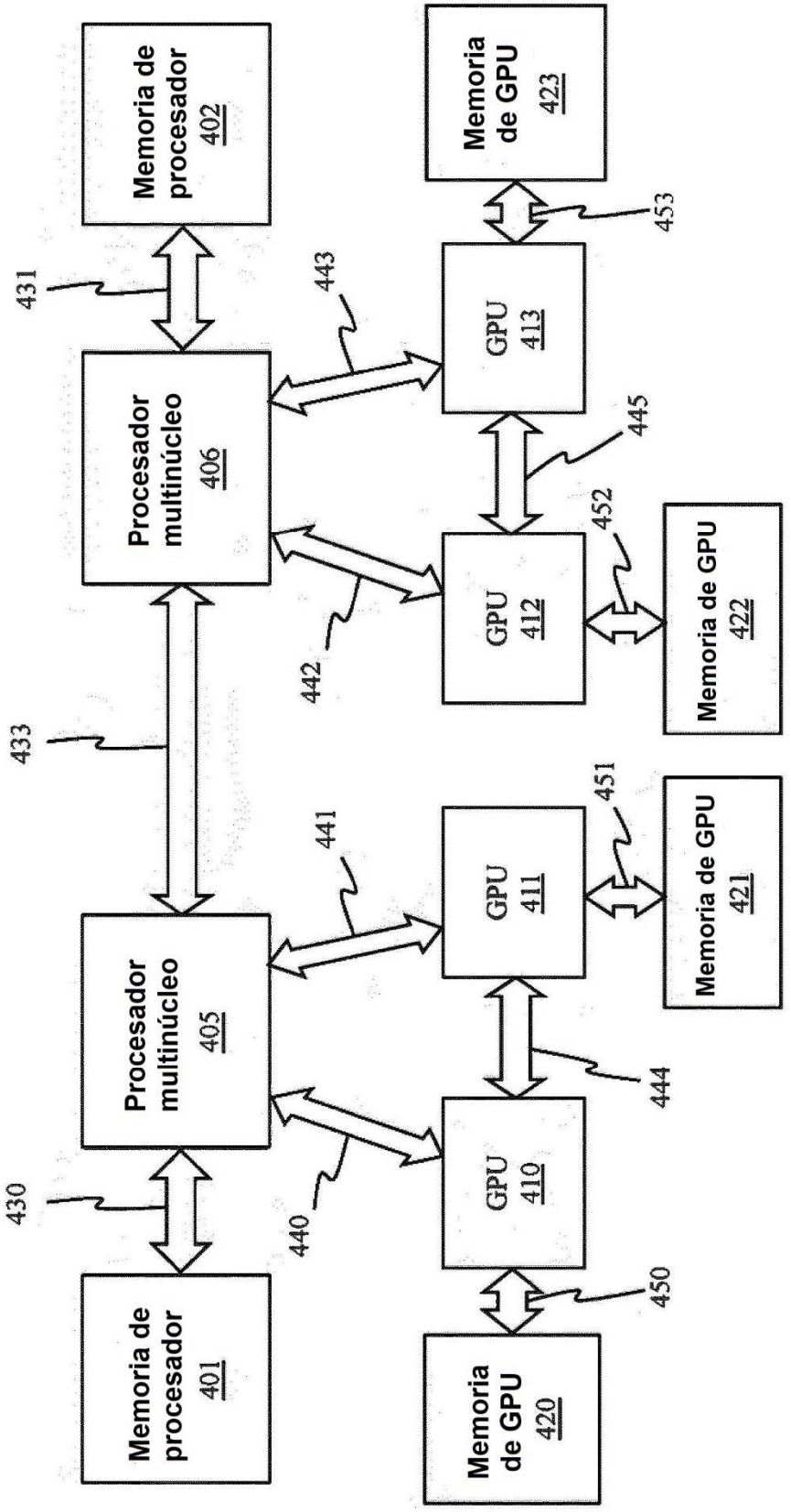

Memoria de procesador 402
Memoria de GPU 423
431
443
453
Procesador multinúcleo 406
GPU 413
445
452
442
GPU 412
Memoria de GPU 422
433
441
451
Procesador multinúcleo 405
GPU 411
Memoria de GPU 421
444
430
440
GPU 410
450
Memoria de procesador 401
Memoria de GPU 420

FIG. 4A

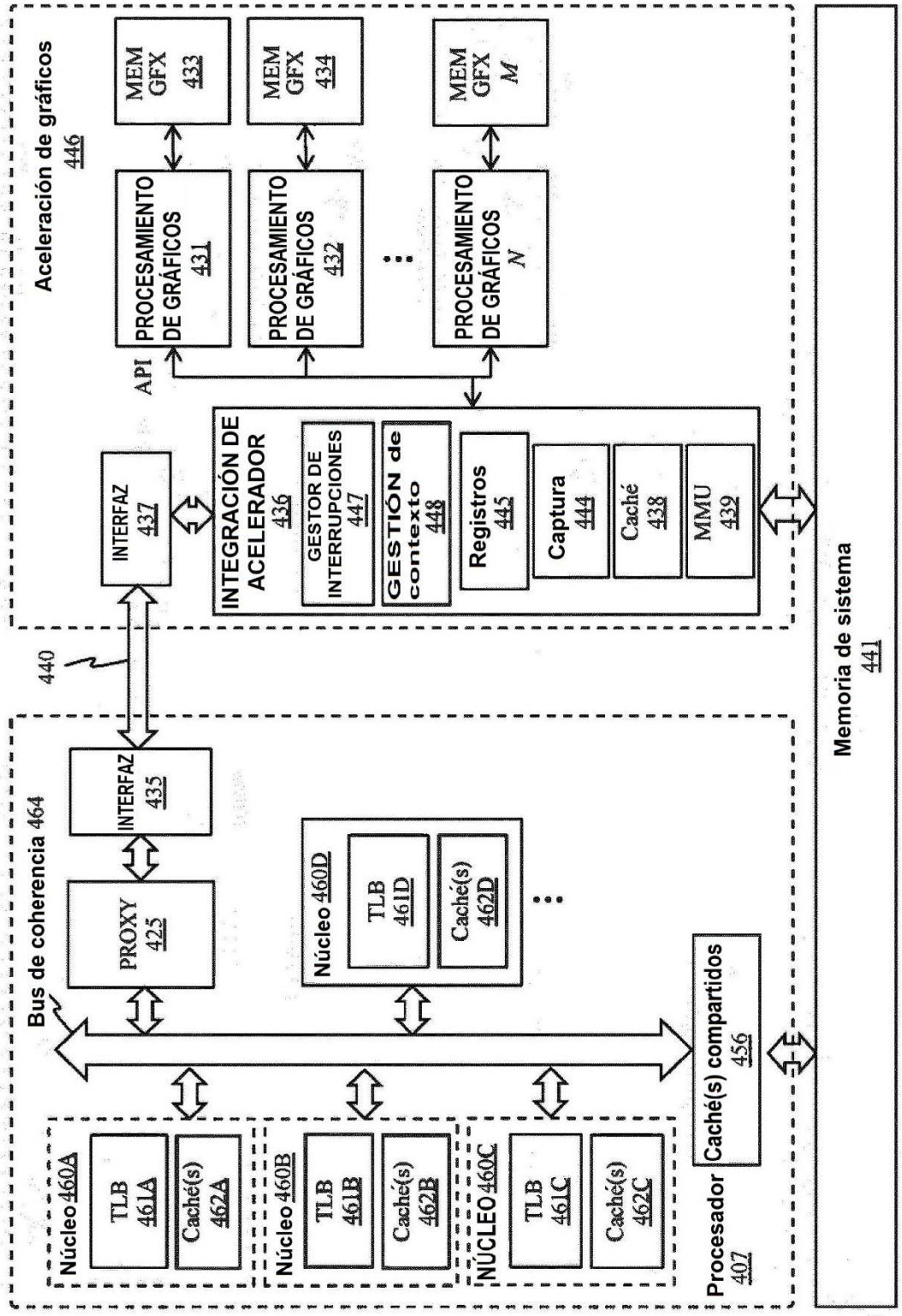

Aceleración de gráficos 446
MEM GFX 433
MEM GFX 434
MEM GFX M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTERFAZ 437
INTEGRACIÓN DE ACELERADOR 436
GESTOR DE INTERRUPCIONES 447
GESTIÓN de contexto 448
Registros 445
Captura 444
Caché 438
MMU 439
440
Memoria de sistema 441
Bus de coherencia 464
INTERFAZ 435
PROXY 425
Núcleo 460D
TLB 461D
Caché(s) 462D
Caché(s) compartidos 456
Núcleo 460A
TLB 461A
Caché(s) 462A
Núcleo 460B
TLB 461B
Caché(s) 462B
NÚCLEO 460C
TLB 461C
Caché(s) 462C
Procesador 407

FIG. 4B

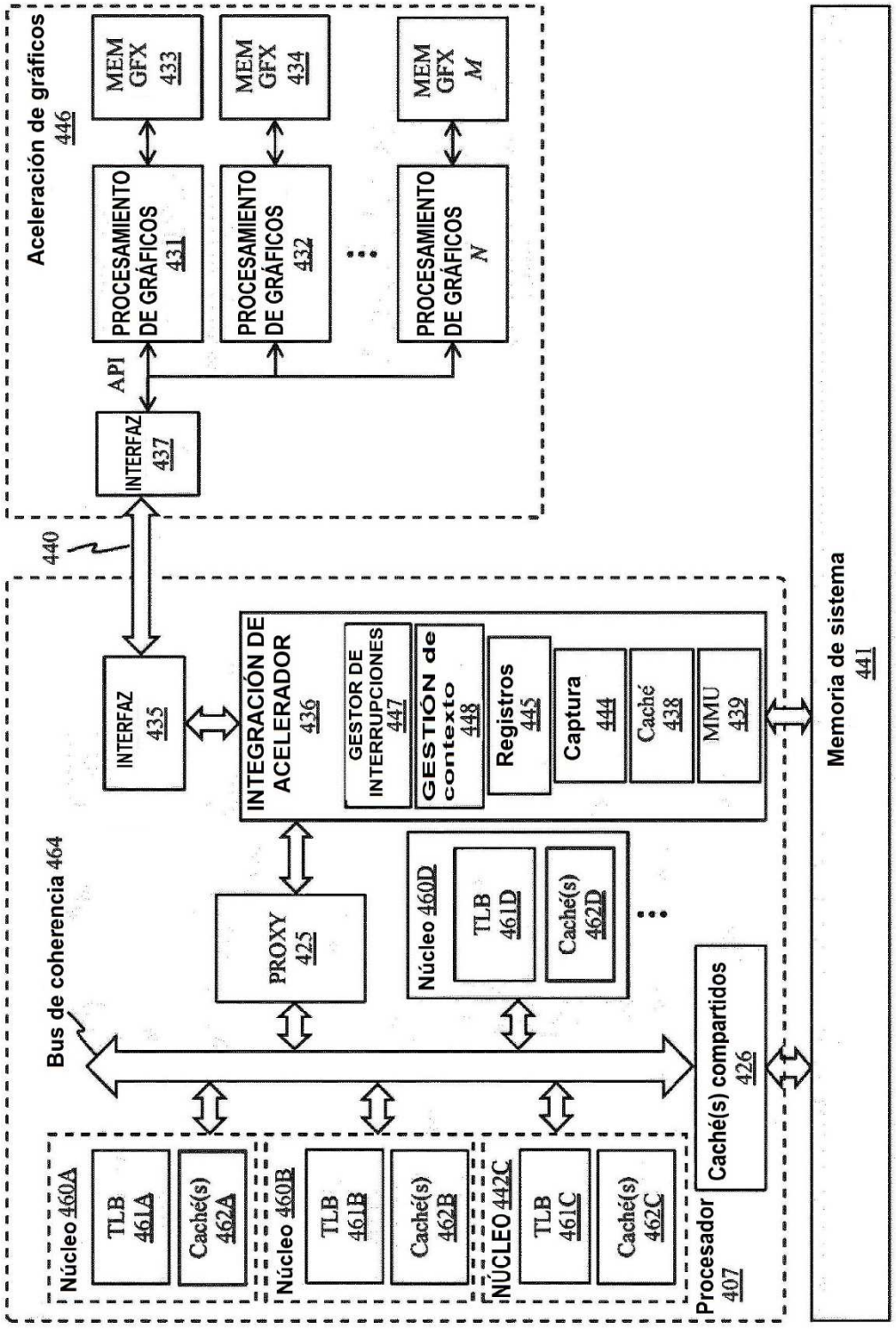

Aceleración de gráficos 446
MEM GFX 433
MEM GFX 434
MEM GFX M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTERFAZ 437
440
INTERFAZ 435
INTEGRACIÓN DE ACELERADOR 436
GESTOR DE INTERRUPCIONES 447
GESTIÓN de contexto 448
Registros 445
Captura 444
Caché 438
MMU 439
Memoria de sistema 441
Bus de coherencia 464
PROXY 425
Núcleo 460D
TLB 461D
Caché(s) 462D
Caché(s) compartidos 426
Procesador 407
Núcleo 460A
TLB 461A
Caché(s) 462A
Núcleo 460B
TLB 461B
Caché(s) 462B
NÚCLEO 442C
TLB 461C
Caché(s) 462C

FIG. 4C

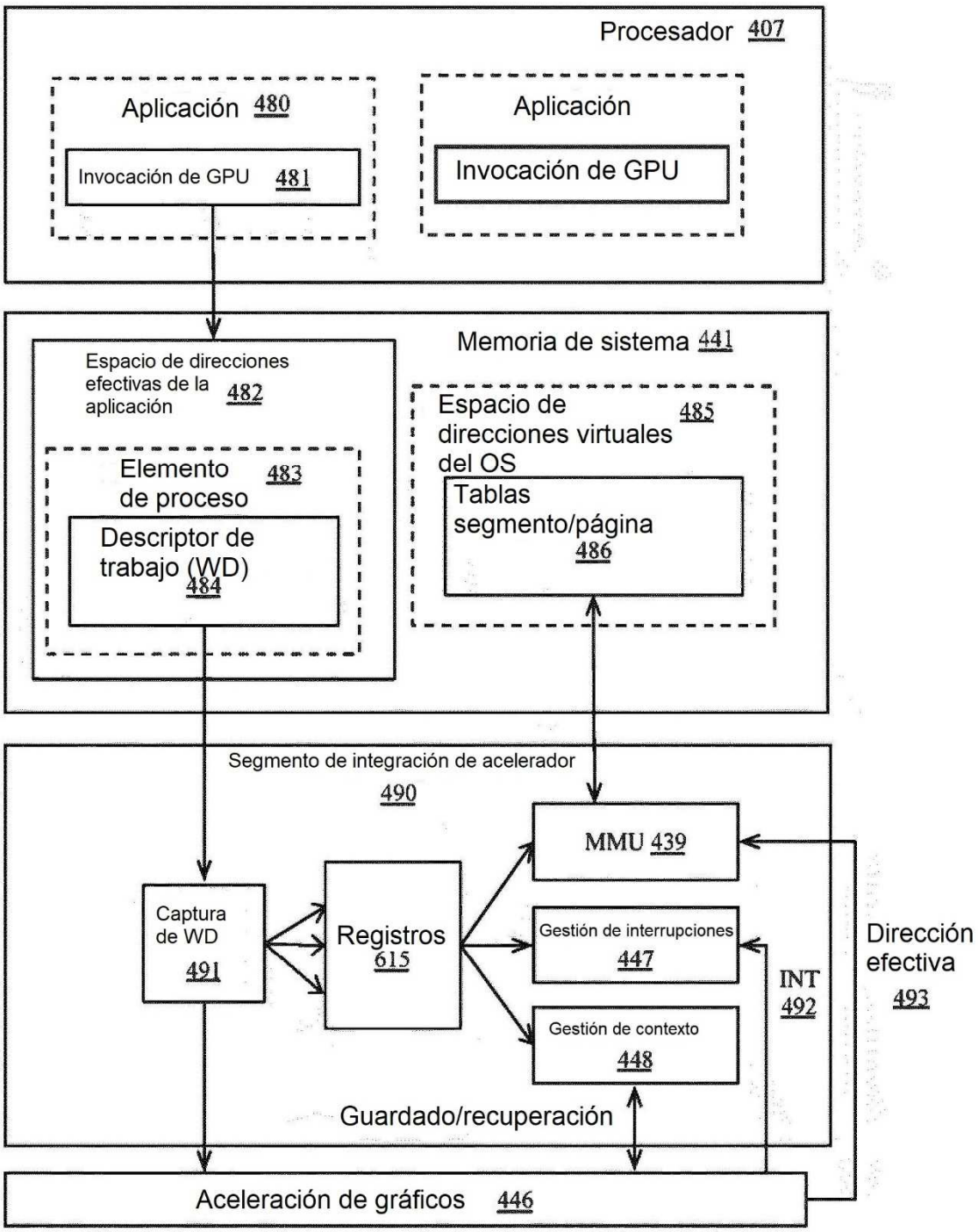

Procesador 407
Aplicación 480
Invocación de GPU 481
Aplicación
Invocación de GPU
Memoria de sistema 441
Espacio de direcciones efectivas de la aplicación 482
Elemento de proceso 483
Descriptor de trabajo (WD) 484
Espacio de direcciones virtuales del OS 485
Tablas segmento/página 486
Segmento de integración de acelerador 490
MMU 439
Captura de WD 491
Registros 615
Gestión de interrupciones 447
Gestión de contexto 448
INT 492
Dirección efectiva 493
Guardado/recuperación
Aceleración de gráficos 446

FIG. 4D

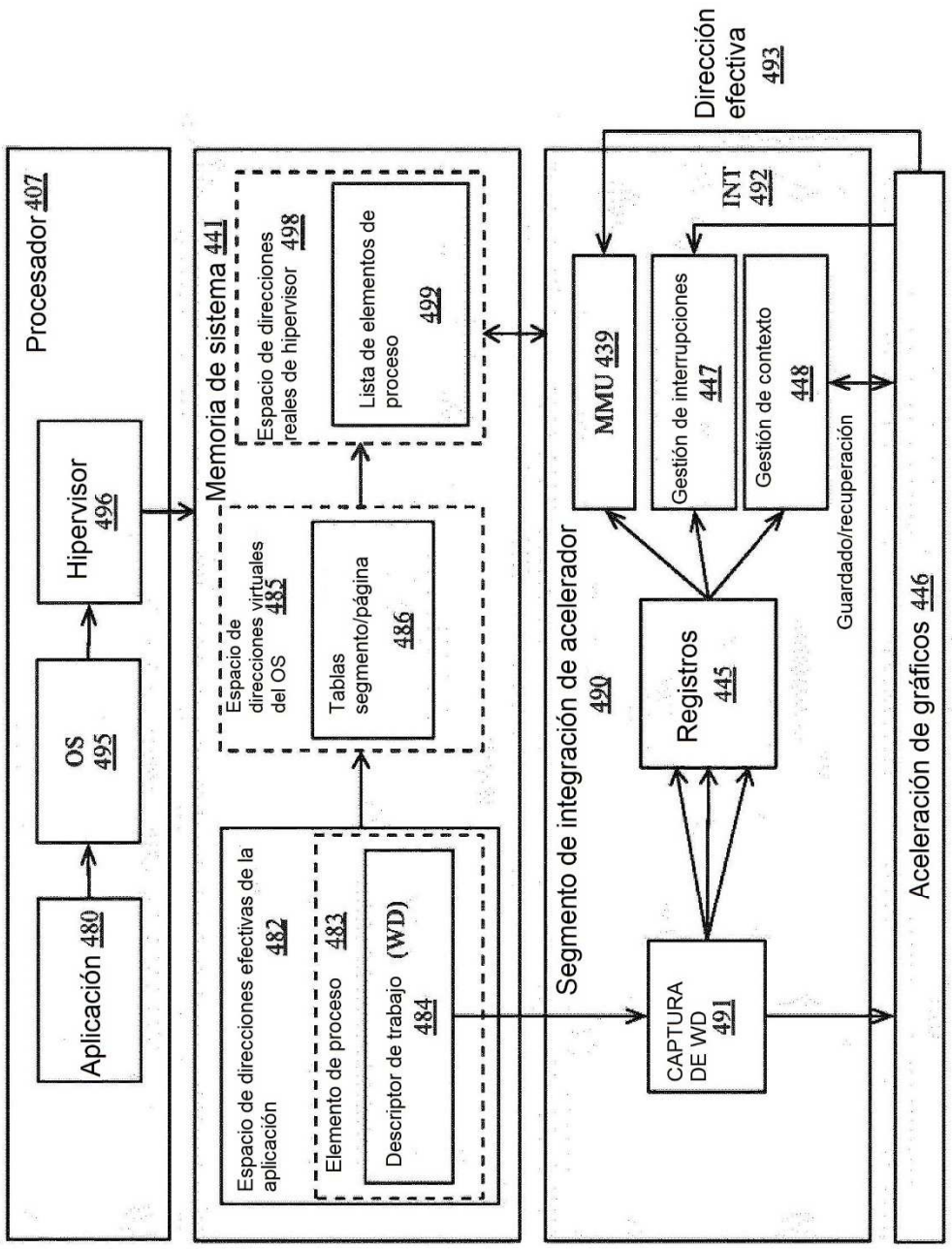

Procesador 407
Aplicación 480
OS 495
Hipervisor 496
Memoria de sistema 441
Espacio de direcciones efectivas de la aplicación 482
Elemento de proceso 483
Descriptor de trabajo (WD) 484
Espacio de direcciones virtuales del OS 485
Tablas segmento/página 486
Espacio de direcciones reales de hipervisor 498
Lista de elementos de proceso 499
Segmento de integración de acelerador 490
CAPTURA DE WD 491
Registros 445
MMU 439
Gestión de interrupciones 447
Gestión de contexto 448
Guardado/recuperación
INT 492
Dirección efectiva 493
Aceleración de gráficos 446

FIG. 4E

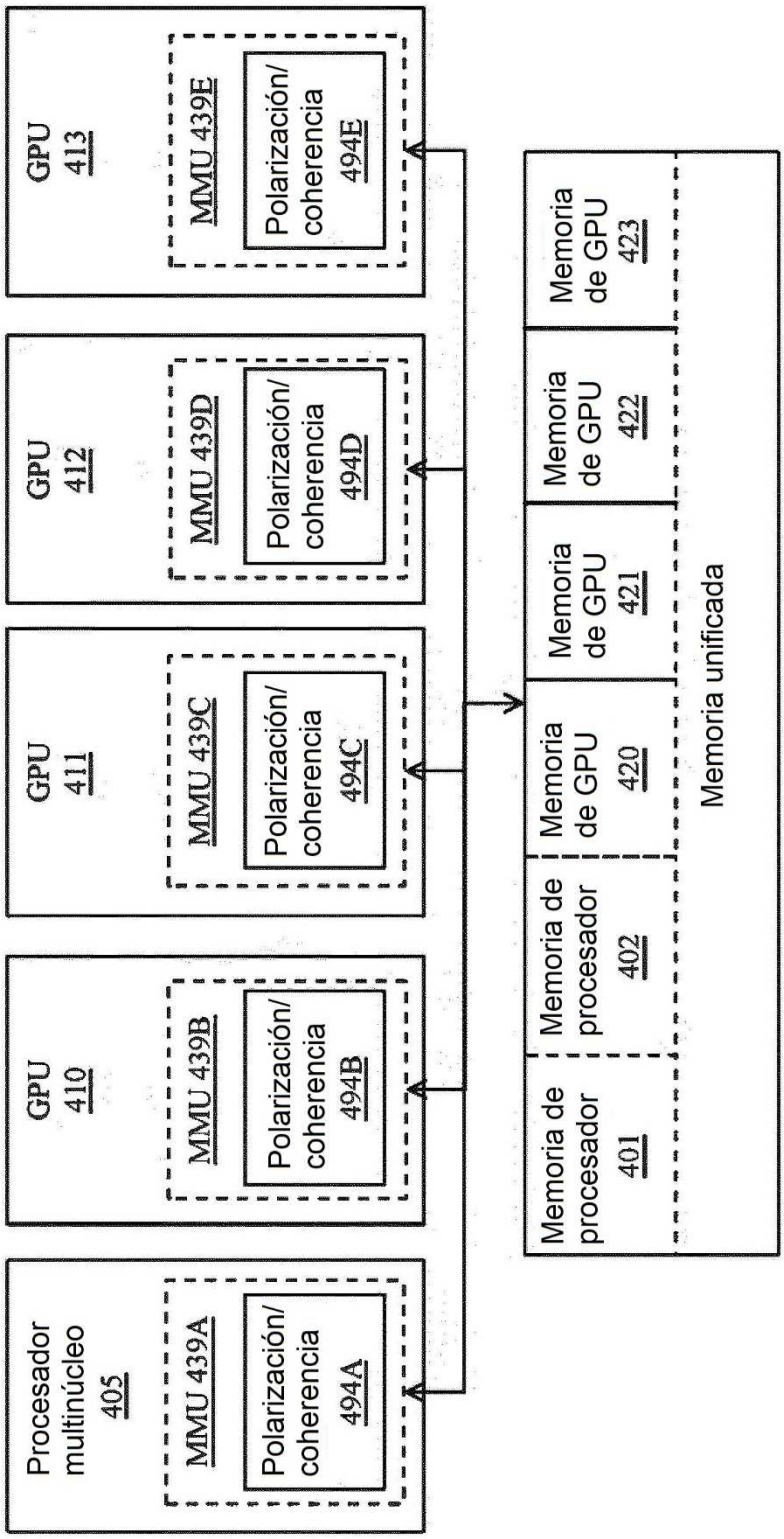

Procesador multinúcleo 405
MMU 439A
Polarización/ coherencia 494A
GPU 410
MMU 439B
Polarización/ coherencia 494B
GPU 411
MMU 439C
Polarización/ coherencia 494C
GPU 412
MMU 439D
Polarización/ coherencia 494D
GPU 413
MMU 439E
Polarización/ coherencia 494E
Memoria de procesador 401
Memoria de procesador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

FIG. 4F

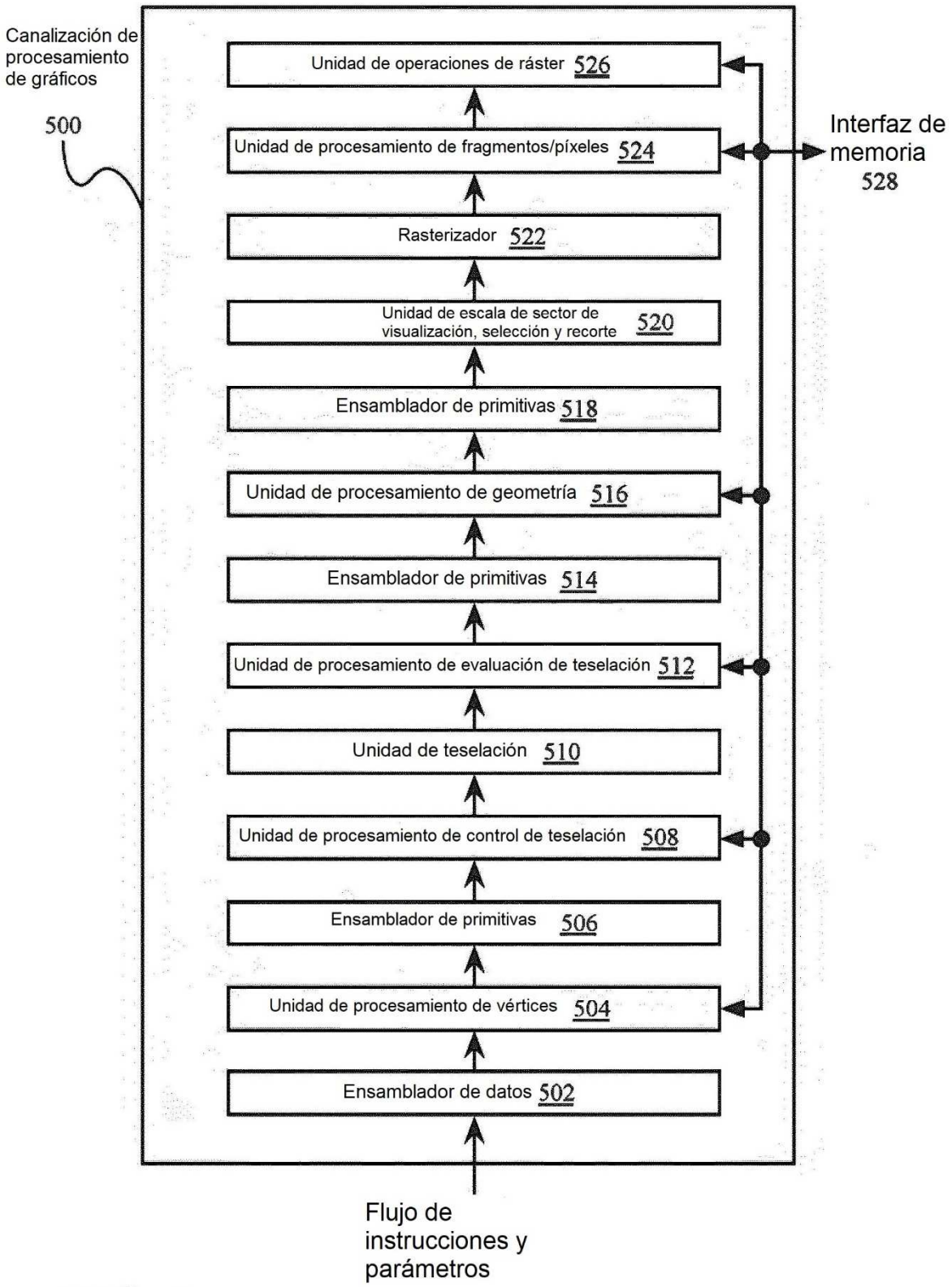

Canalización de procesamiento de gráficos
500
Unidad de operaciones de ráster 526
Unidad de procesamiento de fragmentos/píxeles 524
Interfaz de memoria 528
Rasterizador 522
Unidad de escala de sector de visualización, selección y recorte 520
Ensamblador de primitivas 518
Unidad de procesamiento de geometría 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

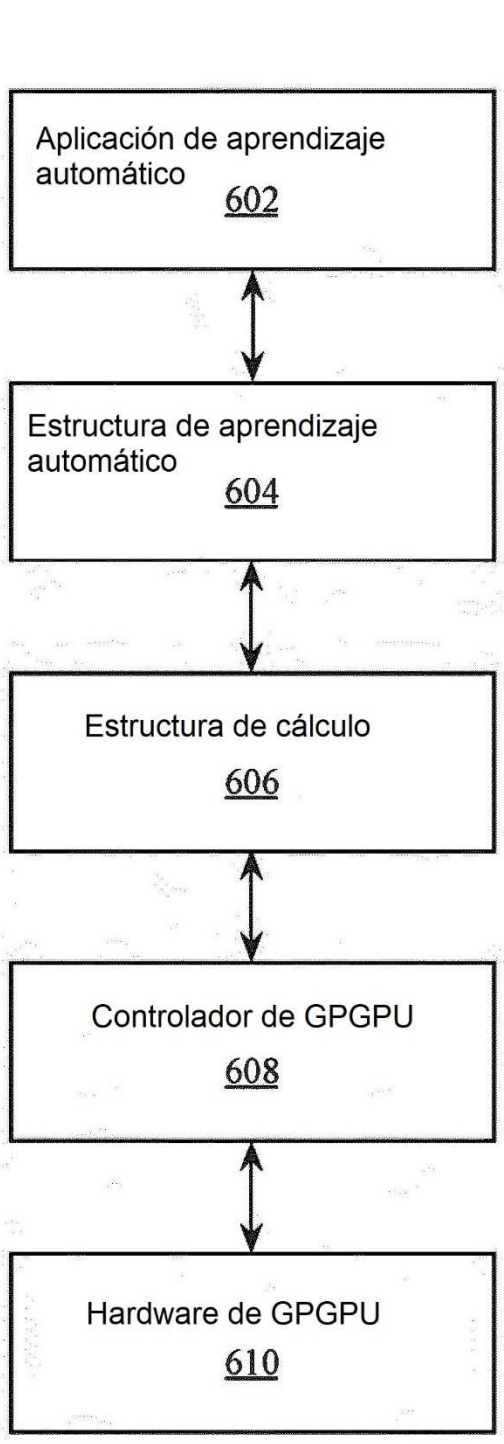

600
Aplicación de aprendizaje automático
602
Estructura de aprendizaje automático
604
Estructura de cálculo
606
Controlador de GPGPU
608
Hardware de GPGPU
610

FIG. 6

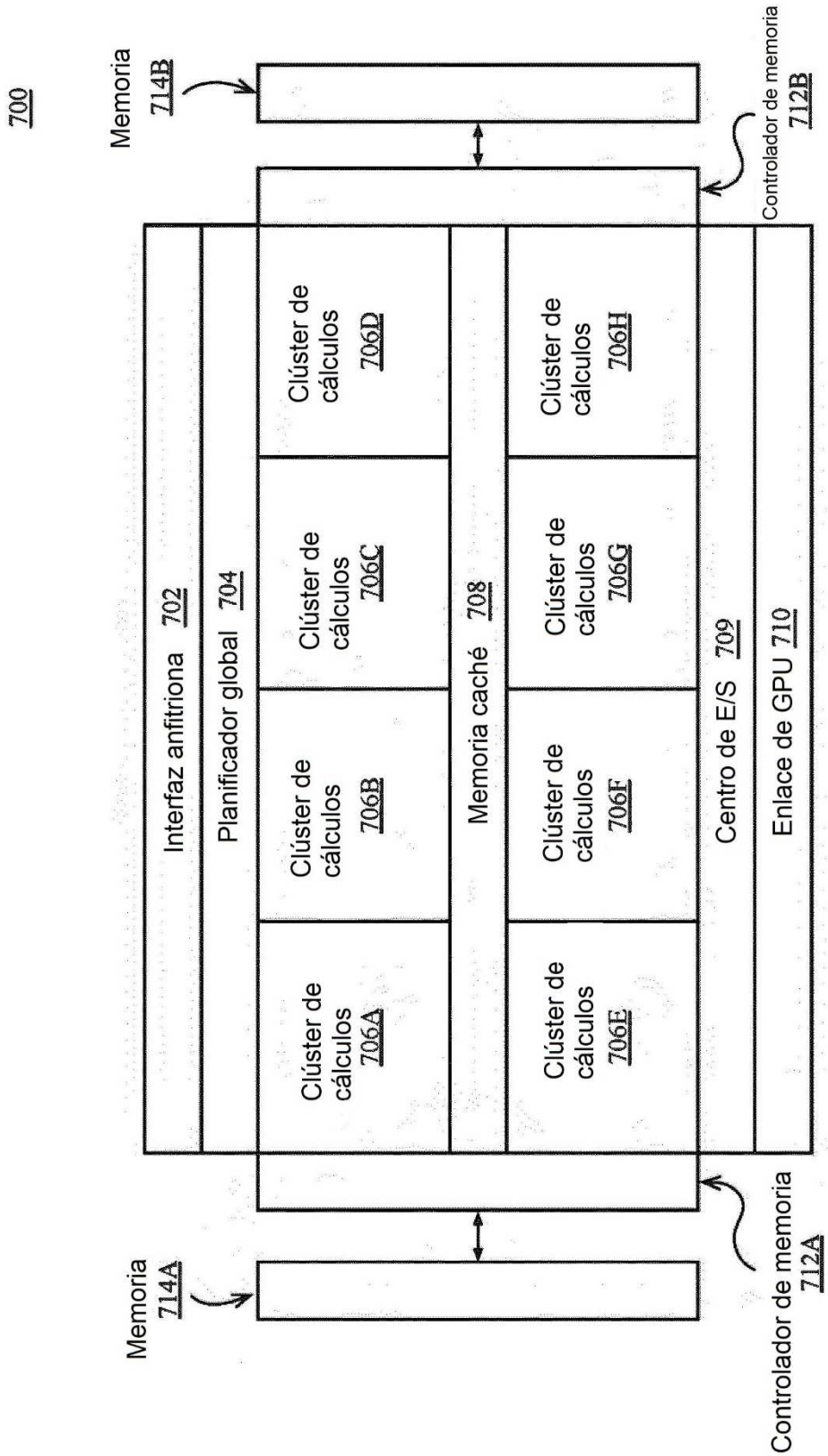

700
Memoria 714B
Controlador de memoria 712B
Interfaz anfitriona 702
Planificador global 704
Clúster de cálculos 706A
Clúster de cálculos 706B
Clúster de cálculos 706C
Clúster de cálculos 706D
Memoria caché 708
Clúster de cálculos 706E
Clúster de cálculos 706F
Clúster de cálculos 706G
Clúster de cálculos 706H
Centro de E/S 709
Enlace de GPU 710
Memoria 714A
Controlador de memoria 712A

FIG. 7

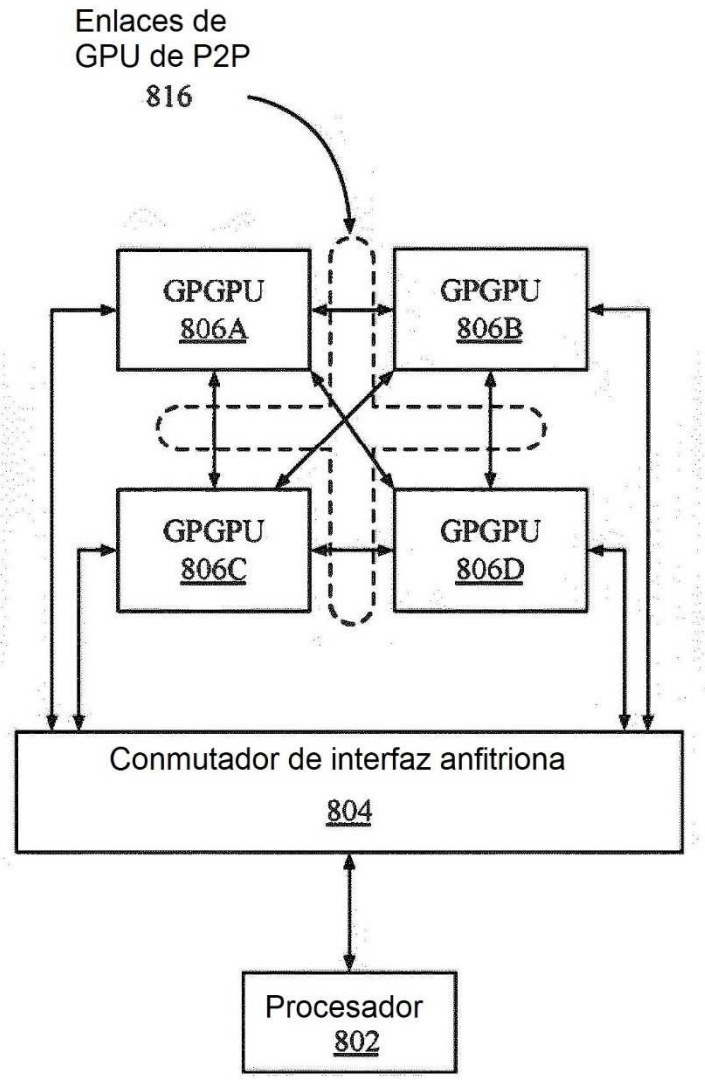

Enlaces de
GPU de P2P
816
GPGPU
806A
GPGPU
806B
GPGPU
806C
GPGPU
806D
Conmutador de interfaz anfitriona
804
Procesador
802

FIG. 8

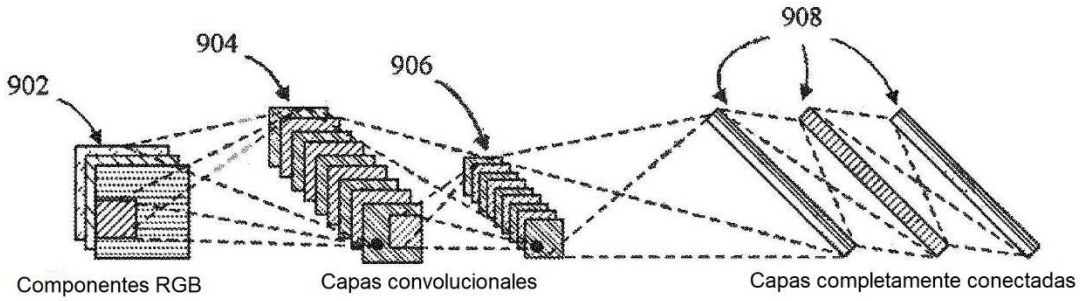

902
904
906
908
Componentes RGB
Capas convolucionales
Capas completamente conectadas

FIG. 9A

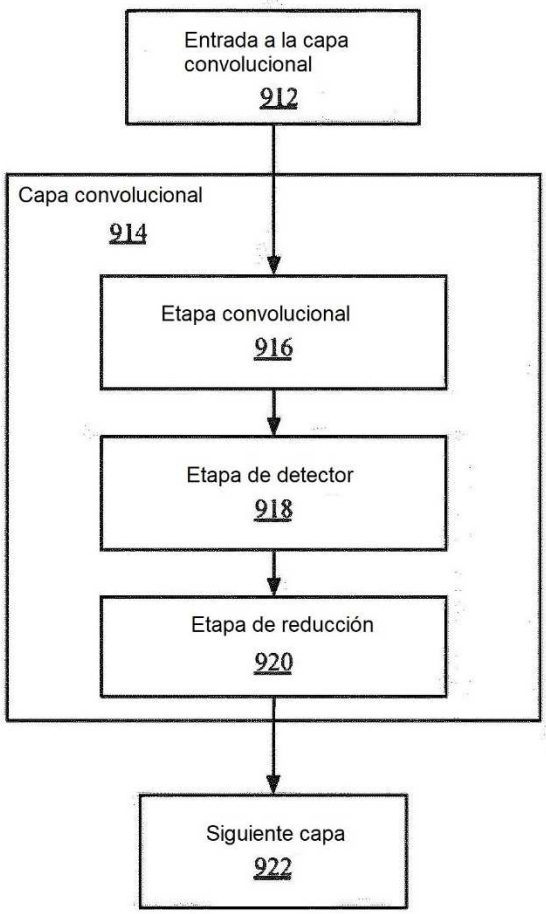

Entrada a la capa convolucional
912
Capa convolucional
914
Etapa convolucional
916
Etapa de detector
918
Etapa de reducción
920
Siguiente capa
922

FIG. 9B

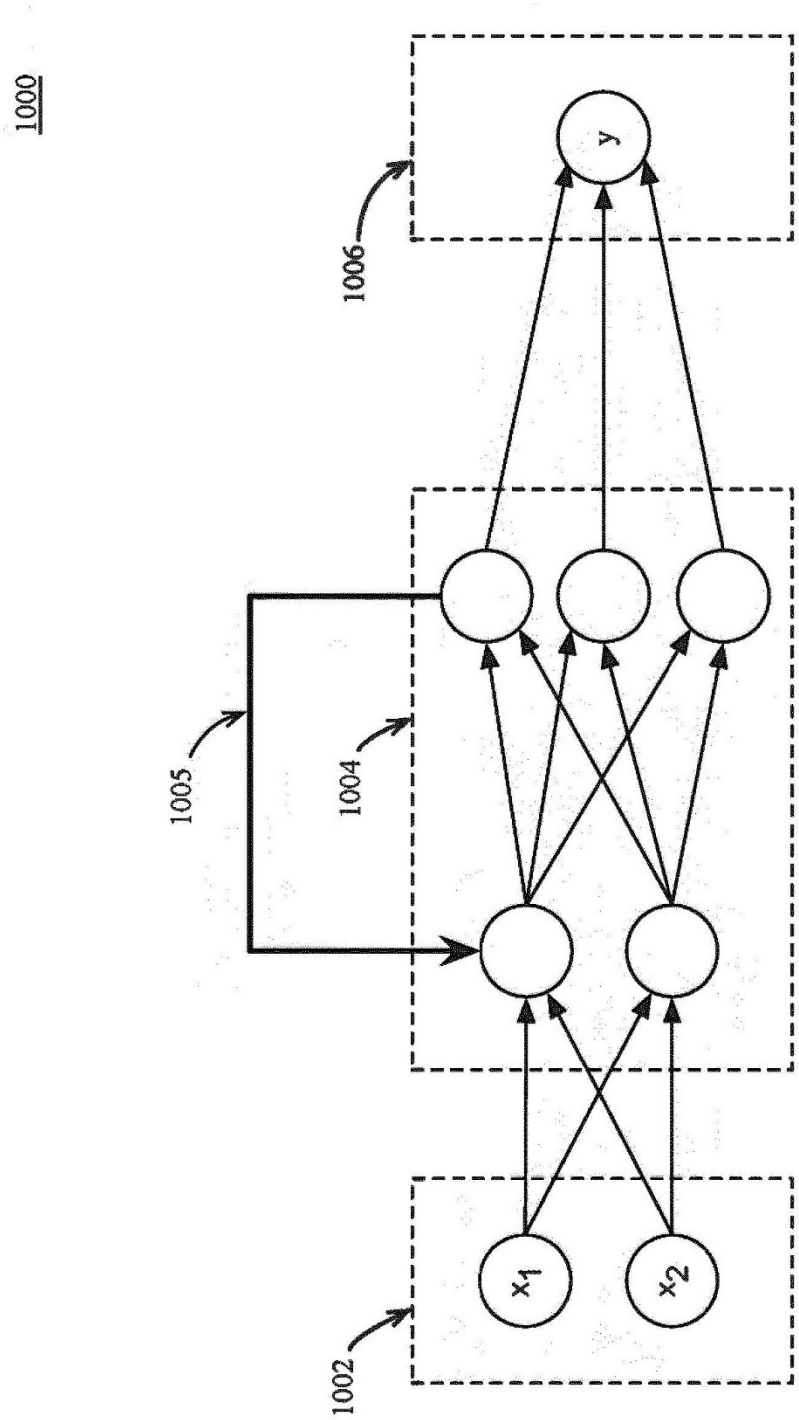


FIG. 10

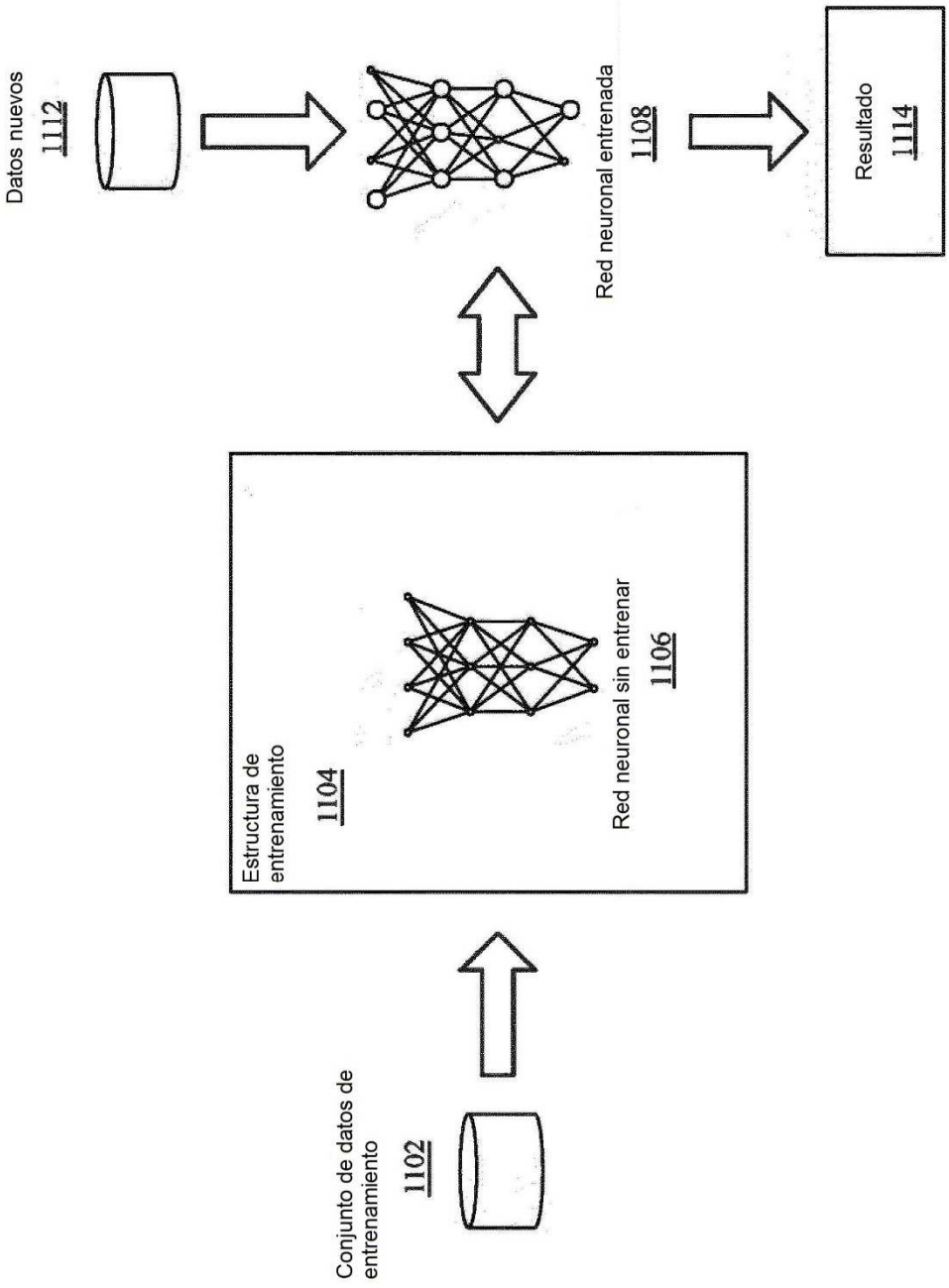

Datos nuevos
1112
Red neuronal entrenada
1108
Resultado
1114
Estructura de entrenamiento
1104
Red neuronal sin entrenar
1106
Conjunto de datos de entrenamiento
1102

FIG. 11

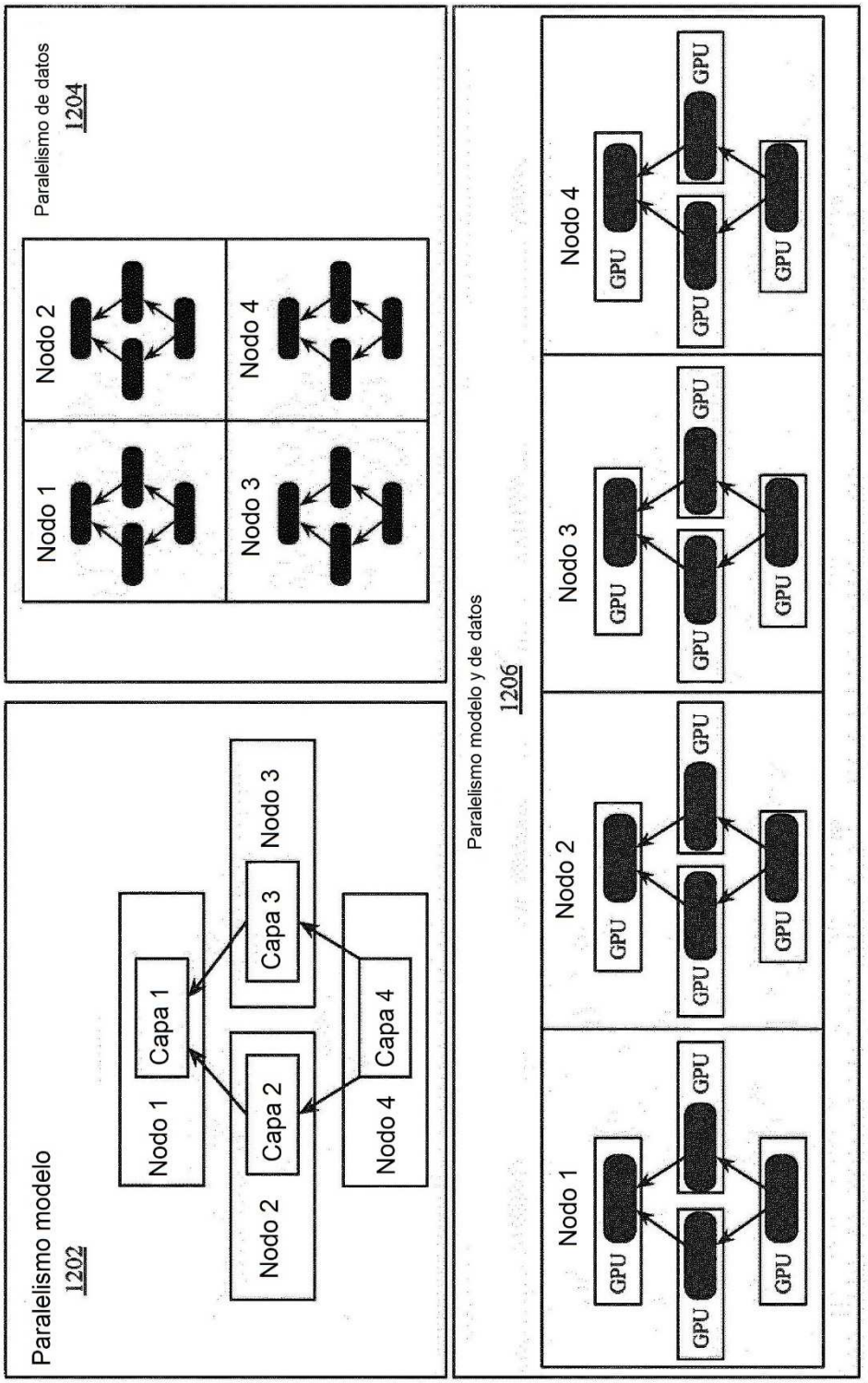

Paralelismo modelo
1202
Nodo 1
Capa 1
Nodo 2
Capa 2
Capa 3
Nodo 3
Nodo 4
Capa 4
Paralelismo de datos
1204
Nodo 1
Nodo 2
Nodo 3
Nodo 4
Paralelismo modelo y de datos
1206
GPU

FIG. 12

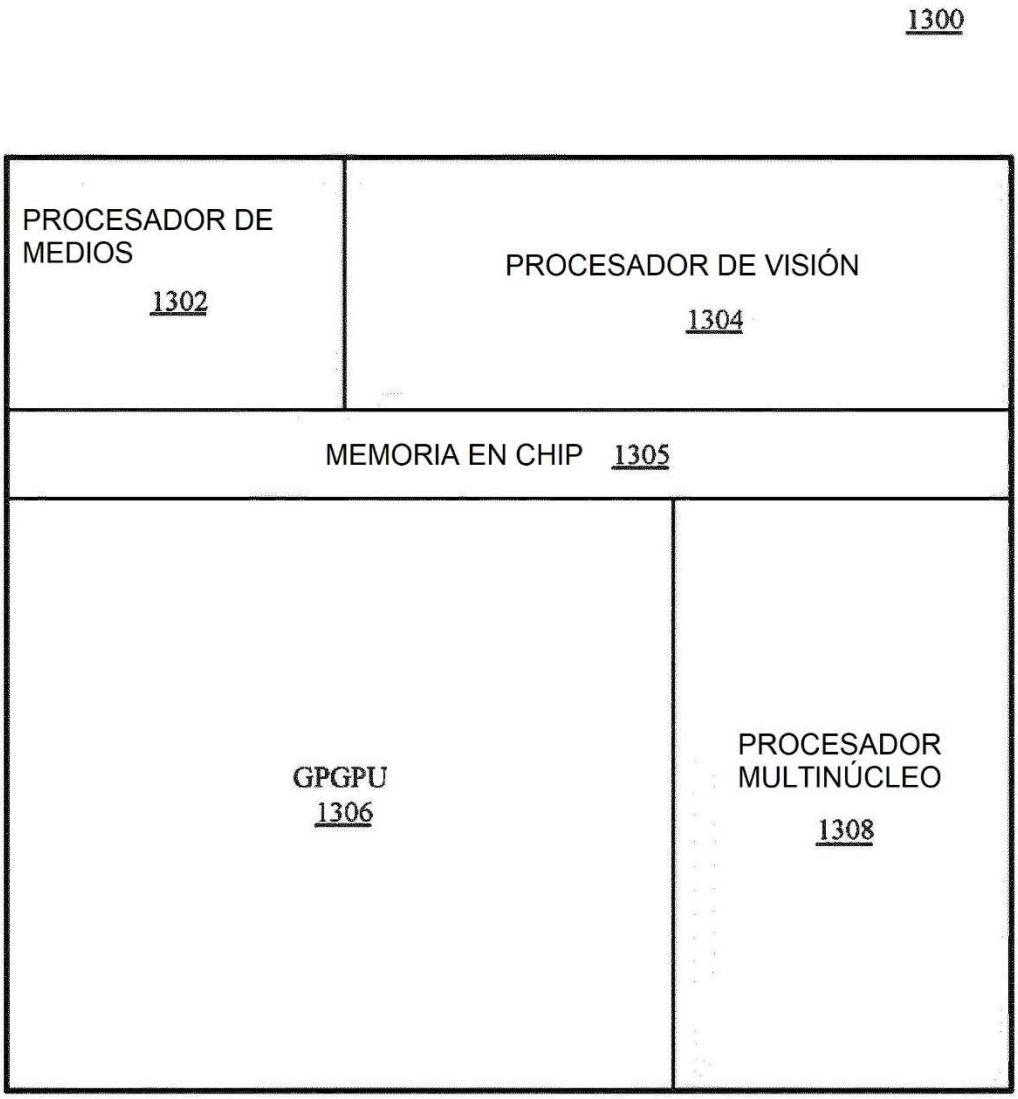

1300
PROCESADOR DE MEDIOS
1302
PROCESADOR DE VISIÓN
1304
MEMORIA EN CHIP 1305
GPGPU
1306
PROCESADOR MULTINÚCLEO
1308

FIG. 13

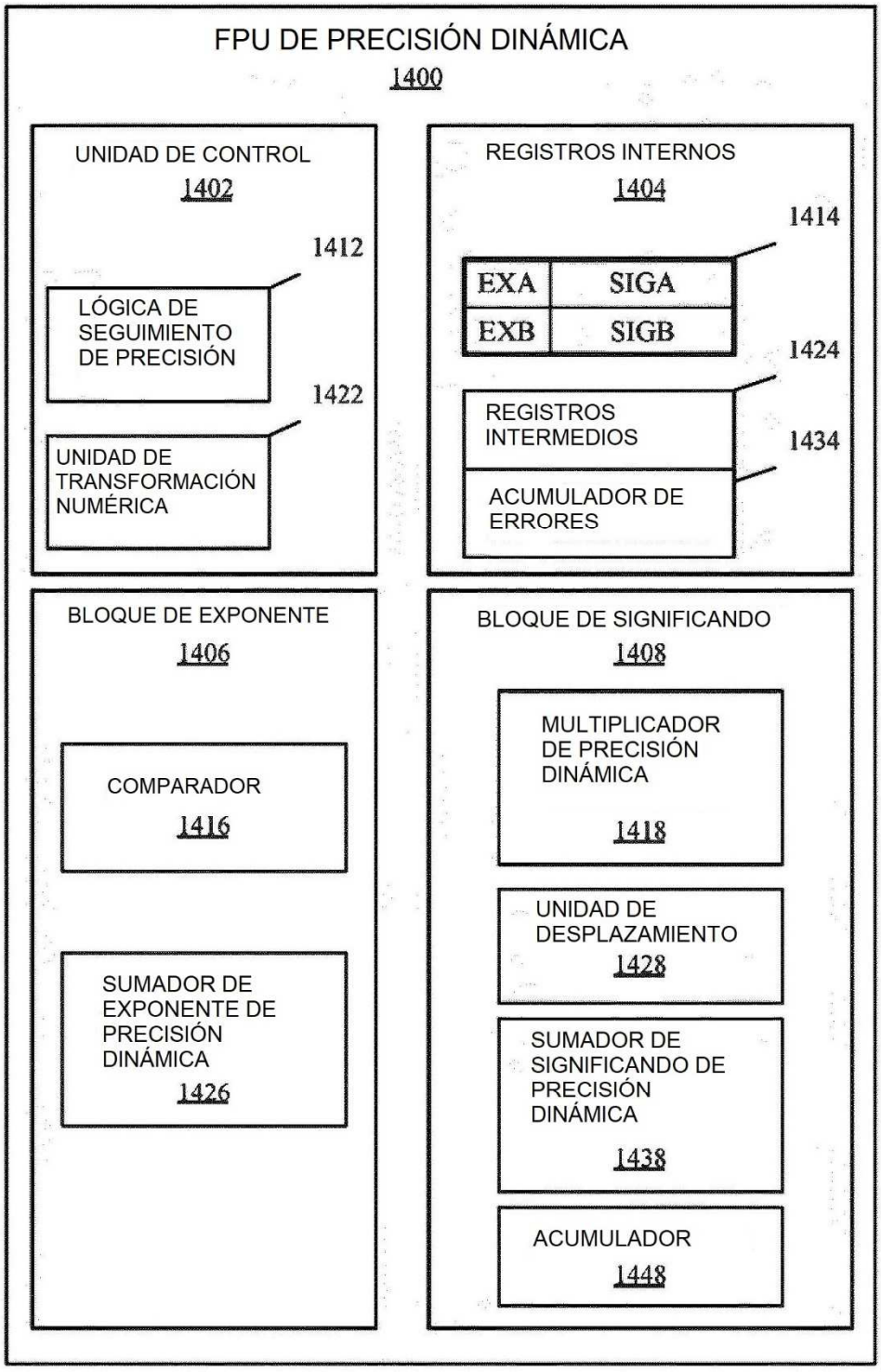

FPU DE PRECISIÓN DINÁMICA
1400
UNIDAD DE CONTROL
1402
1412
LÓGICA DE SEGUIMIENTO DE PRECISIÓN
1422
UNIDAD DE TRANSFORMACIÓN NUMÉRICA
REGISTROS INTERNOS
1404
1414
EXA
SIGA
EXB
SIGB
1424
REGISTROS INTERMEDIOS
1434
ACUMULADOR DE ERRORES
BLOQUE DE EXPONENTE
1406
COMPARADOR
1416
SUMADOR DE EXPONENTE DE PRECISIÓN DINÁMICA
1426
BLOQUE DE SIGNIFICANDO
1408
MULTIPLICADOR DE PRECISIÓN DINÁMICA
1418
UNIDAD DE DESPLAZAMIENTO
1428
SUMADOR DE SIGNIFICANDO DE PRECISIÓN DINÁMICA
1438
ACUMULADOR
1448

FIG. 14

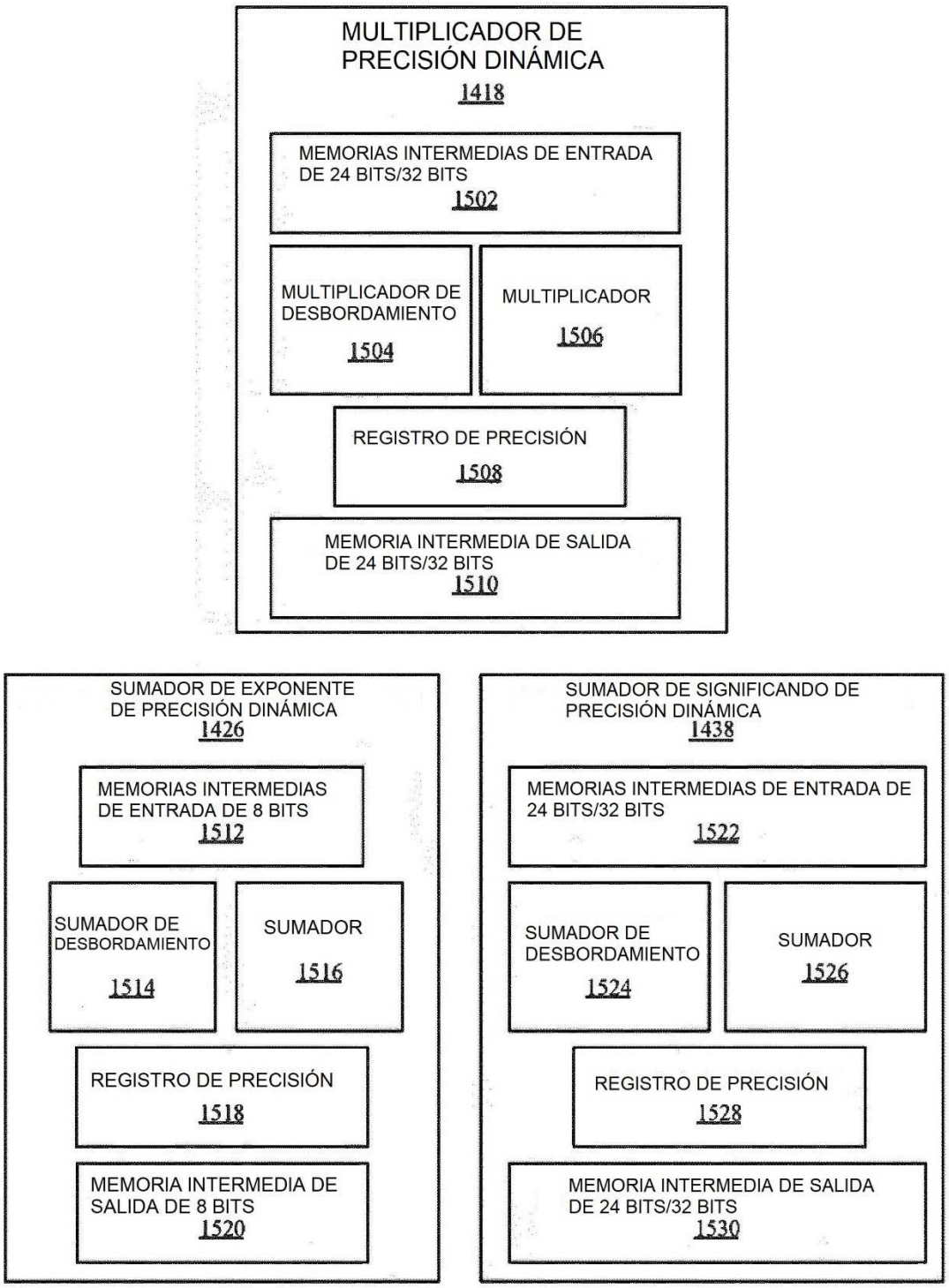

MULTIPLICADOR DE PRECISIÓN DINÁMICA
1418
MEMORIAS INTERMEDIAS DE ENTRADA DE 24 BITS/32 BITS
1502
MULTIPLICADOR DE DESBORDAMIENTO
1504
MULTIPLICADOR
1506
REGISTRO DE PRECISIÓN
1508
MEMORIA INTERMEDIA DE SALIDA DE 24 BITS/32 BITS
1510
SUMADOR DE EXPONENTE DE PRECISIÓN DINÁMICA
1426
MEMORIAS INTERMEDIAS DE ENTRADA DE 8 BITS
1512
SUMADOR DE DESBORDAMIENTO
1514
SUMADOR
1516
REGISTRO DE PRECISIÓN
1518
MEMORIA INTERMEDIA DE SALIDA DE 8 BITS
1520
SUMADOR DE SIGNIFICANDO DE PRECISIÓN DINÁMICA
1438
MEMORIAS INTERMEDIAS DE ENTRADA DE 24 BITS/32 BITS
1522
SUMADOR DE DESBORDAMIENTO
1524
SUMADOR
1526
REGISTRO DE PRECISIÓN
1528
MEMORIA INTERMEDIA DE SALIDA DE 24 BITS/32 BITS
1530

FIG. 15

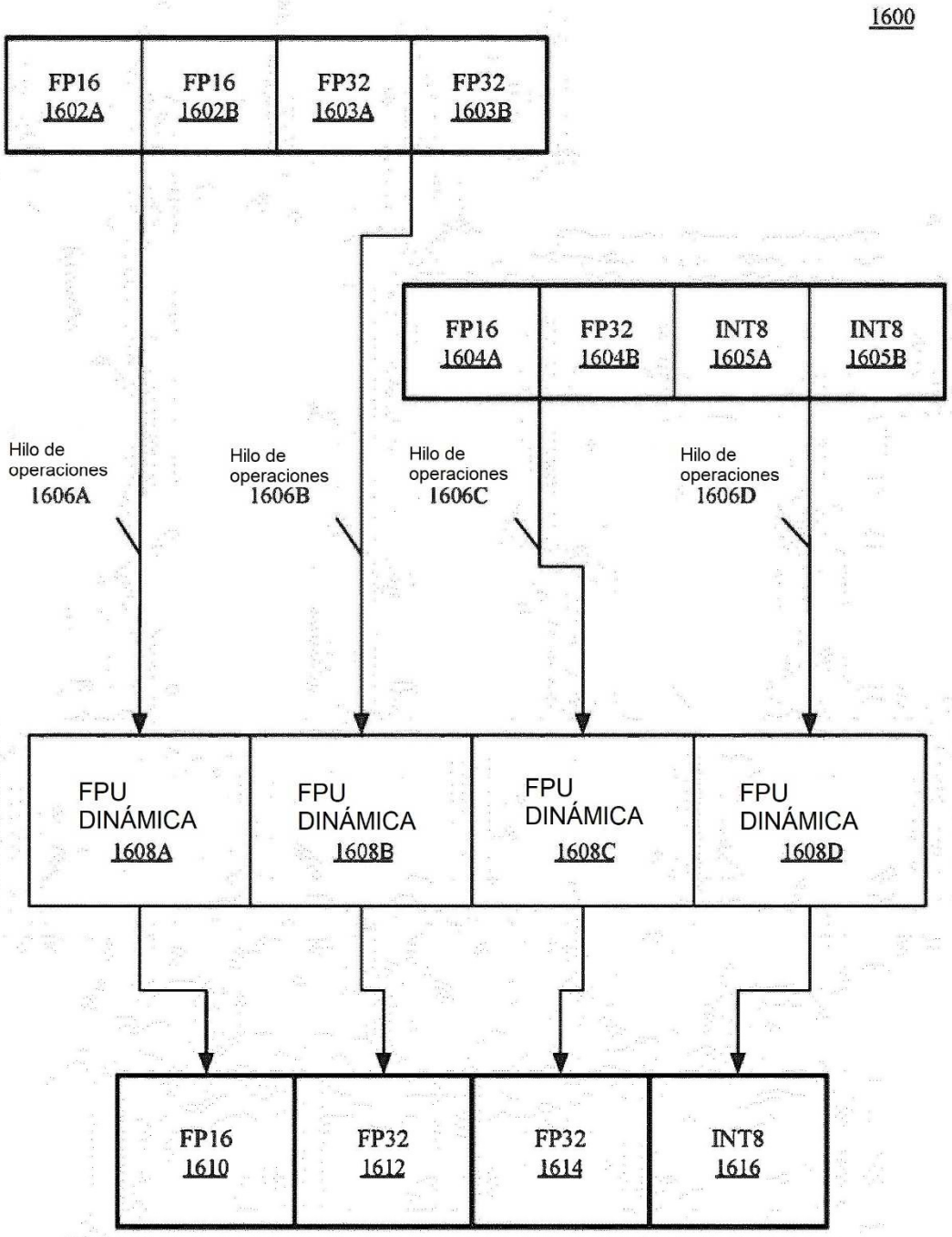

1600
FP16 1602A
FP16 1602B
FP32 1603A
FP32 1603B
FP16 1604A
FP32 1604B
INT8 1605A
INT8 1605B
Hilo de operaciones 1606A
Hilo de operaciones 1606B
Hilo de operaciones 1606C
Hilo de operaciones 1606D
FPU DINÁMICA 1608A
FPU DINÁMICA 1608B
FPU DINÁMICA 1608C
FPU DINÁMICA 1608D
FP16 1610
FP32 1612
FP32 1614
INT8 1616

FIG. 16

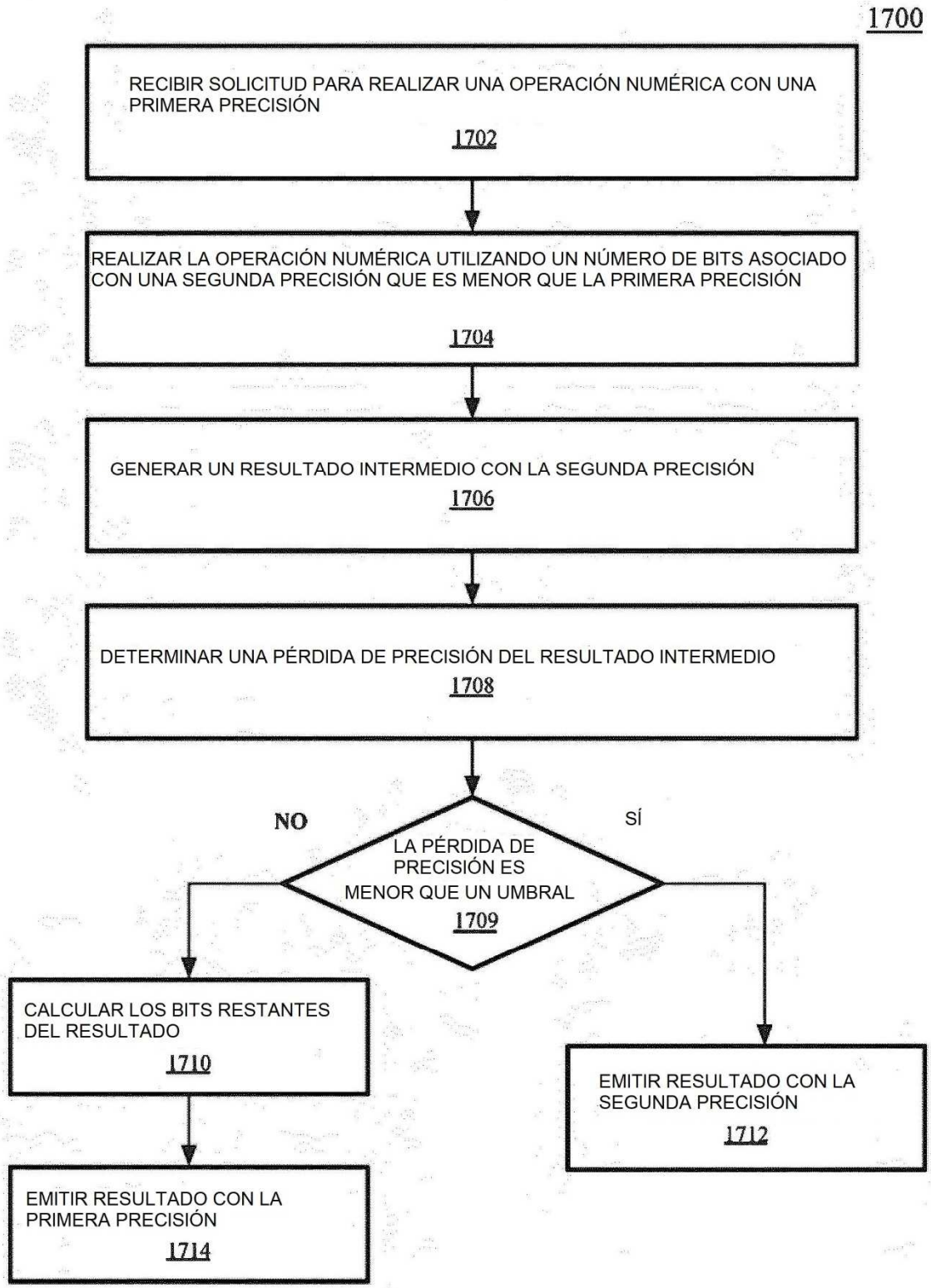

1700
RECIBIR SOLICITUD PARA REALIZAR UNA OPERACIÓN NUMÉRICA CON UNA PRIMERA PRECISIÓN
1702
REALIZAR LA OPERACIÓN NUMÉRICA UTILIZANDO UN NÚMERO DE BITS ASOCIADO CON UNA SEGUNDA PRECISIÓN QUE ES MENOR QUE LA PRIMERA PRECISIÓN
1704
GENERAR UN RESULTADO INTERMEDIO CON LA SEGUNDA PRECISIÓN
1706
DETERMINAR UNA PÉRDIDA DE PRECISIÓN DEL RESULTADO INTERMEDIO
1708
NO
SÍ
LA PÉRDIDA DE PRECISIÓN ES MENOR QUE UN UMBRAL
1709
CALCULAR LOS BITS RESTANTES DEL RESULTADO
1710
EMITIR RESULTADO CON LA SEGUNDA PRECISIÓN
1712
EMITIR RESULTADO CON LA PRIMERA PRECISIÓN
1714

FIG. 17

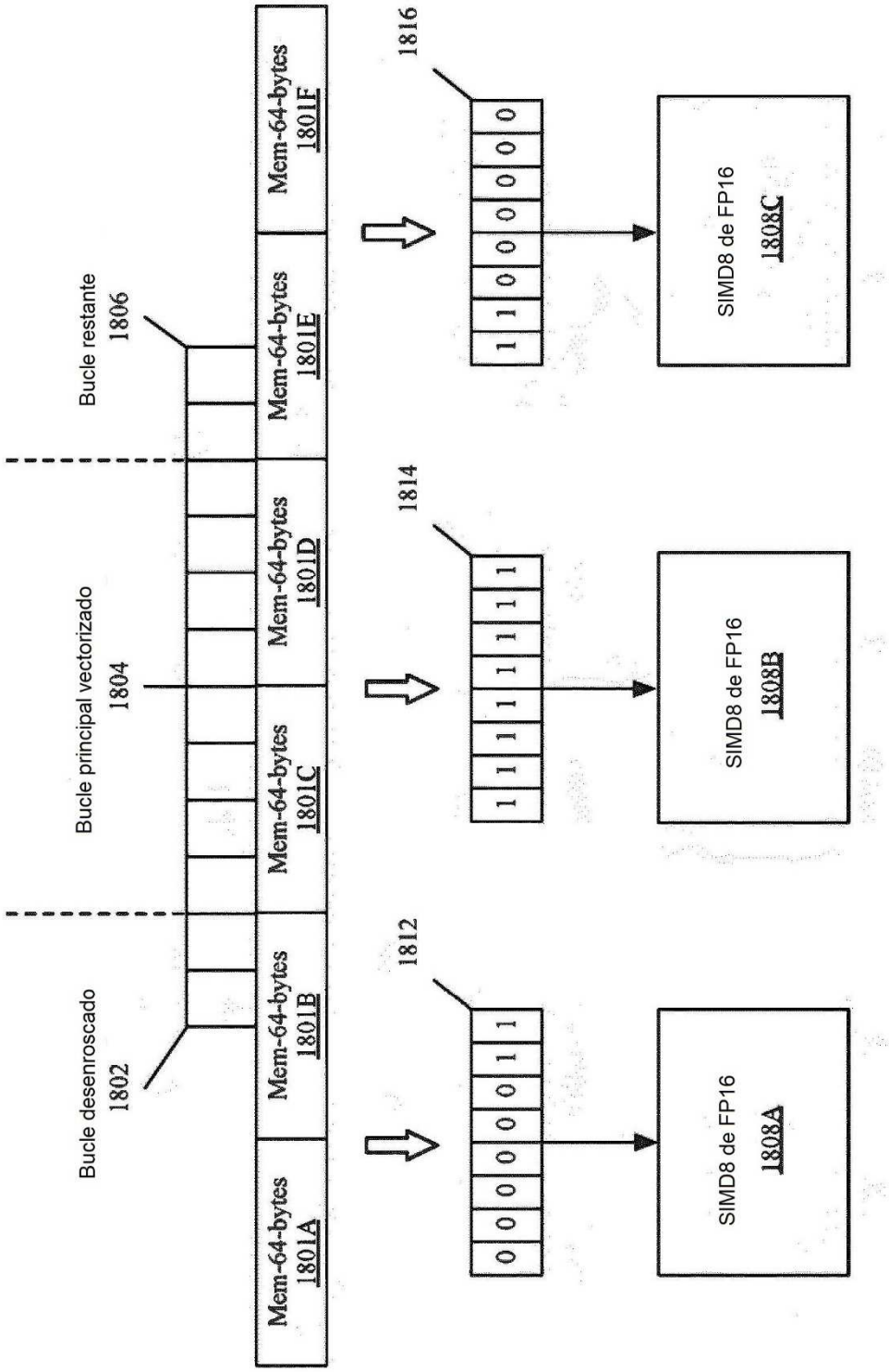

Bucle desenroscado
1802
Bucle principal vectorizado
1804
Bucle restante
1806
Mem-64-bytes 1801A
Mem-64-bytes 1801B
Mem-64-bytes 1801C
Mem-64-bytes 1801D
Mem-64-bytes 1801E
Mem-64-bytes 1801F
1812
0 0 0 0 0 1 1 1
1814
1 1 1 1 1 1 1 1
1816
1 1 0 0 0 0 0 0
SIMD8 de FP16 1808A
SIMD8 de FP16 1808B
SIMD8 de FP16 1808C

FIG. 18

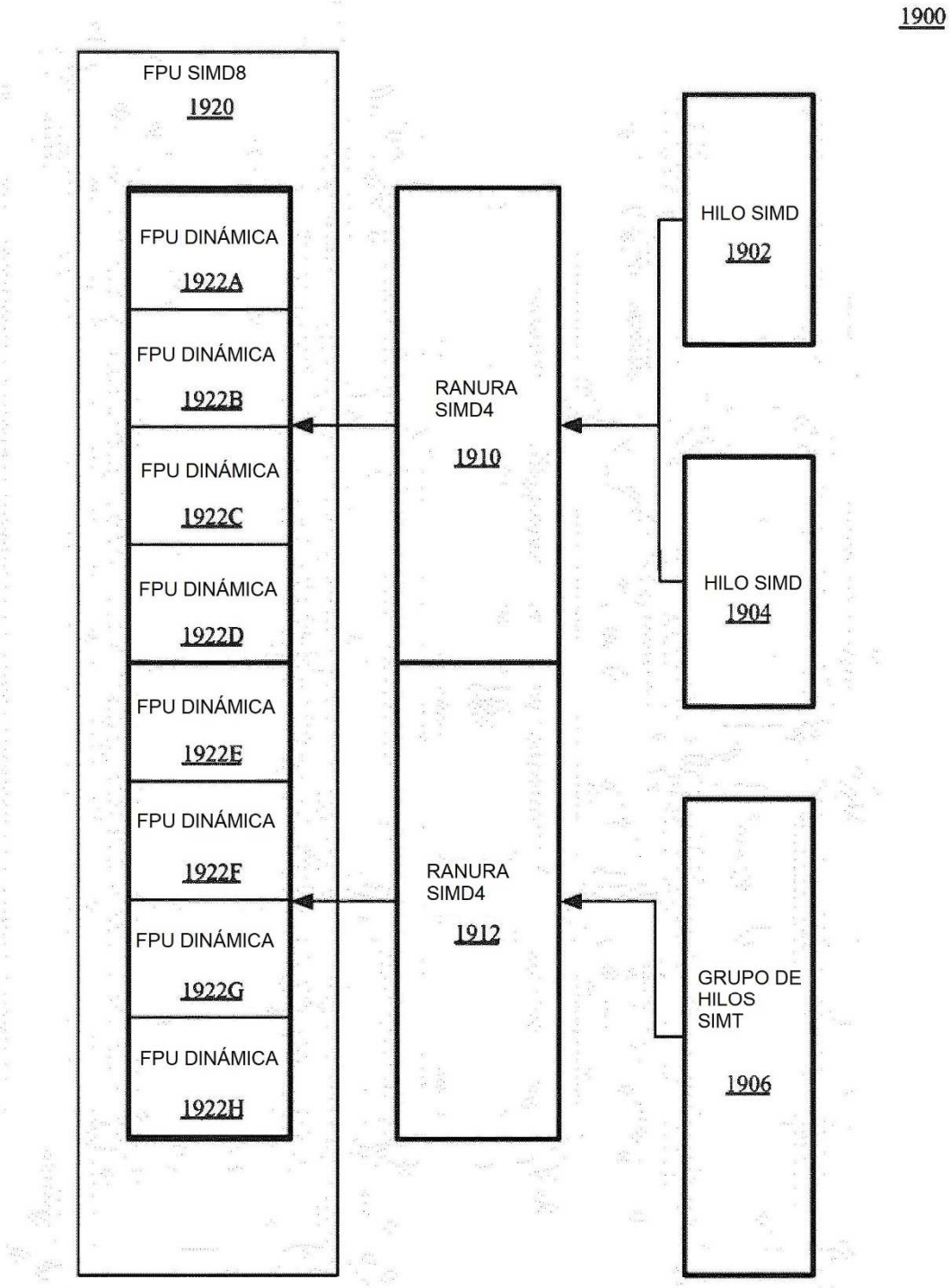

1900
FPU SIMD8
1920
FPU DINÁMICA
1922A
FPU DINÁMICA
1922B
FPU DINÁMICA
1922C
FPU DINÁMICA
1922D
FPU DINÁMICA
1922E
FPU DINÁMICA
1922F
FPU DINÁMICA
1922G
FPU DINÁMICA
1922H
RANURA SIMD4
1910
RANURA SIMD4
1912
HILO SIMD
1902
HILO SIMD
1904
GRUPO DE HILOS SIMT
1906

FIG. 19

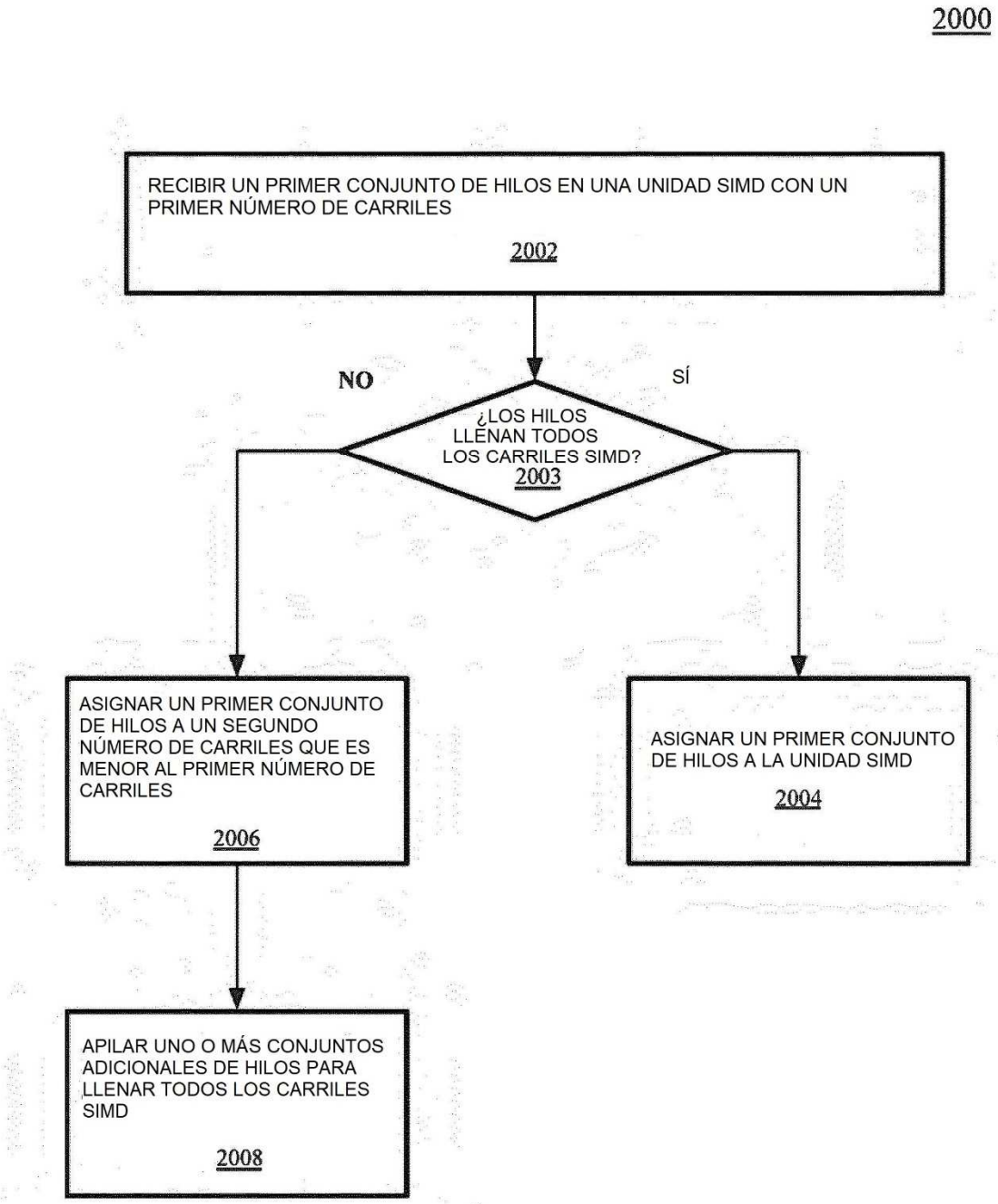

2000
RECIBIR UN PRIMER CONJUNTO DE HILOS EN UNA UNIDAD SIMD CON UN PRIMER NÚMERO DE CARRILES
2002
NO
SÍ
¿LOS HILOS LLENAN TODOS LOS CARRILES SIMD?
2003
ASIGNAR UN PRIMER CONJUNTO DE HILOS A UN SEGUNDO NÚMERO DE CARRILES QUE ES MENOR AL PRIMER NÚMERO DE CARRILES
2006
ASIGNAR UN PRIMER CONJUNTO DE HILOS A LA UNIDAD SIMD
2004
APILAR UNO O MÁS CONJUNTOS ADICIONALES DE HILOS PARA LLENAR TODOS LOS CARRILES SIMD
2008

FIG. 20

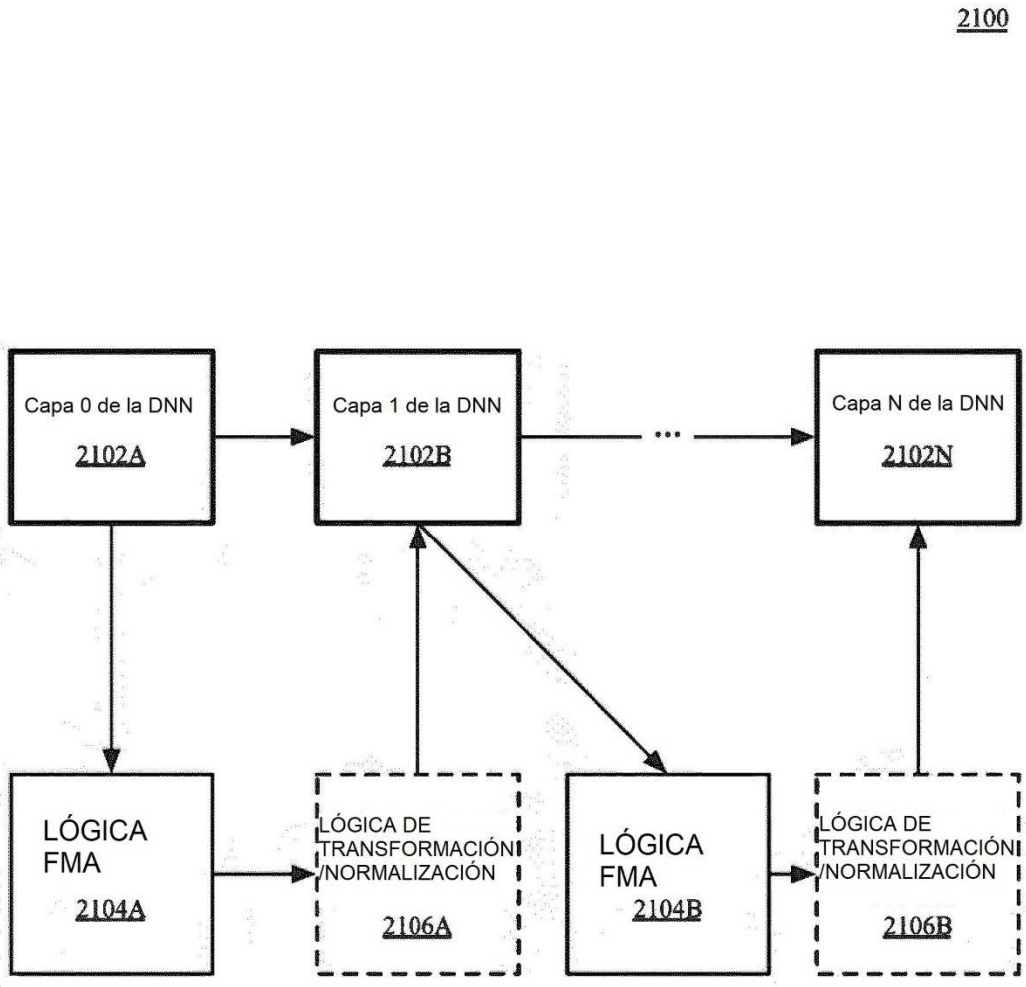

2100
Capa 0 de la DNN
2102A
Capa 1 de la DNN
2102B
Capa N de la DNN
2102N
LÓGICA FMA
2104A
LÓGICA DE TRANSFORMACIÓN /NORMALIZACIÓN
2106A
LÓGICA FMA
2104B
LÓGICA DE TRANSFORMACIÓN /NORMALIZACIÓN
2106B

FIG. 21

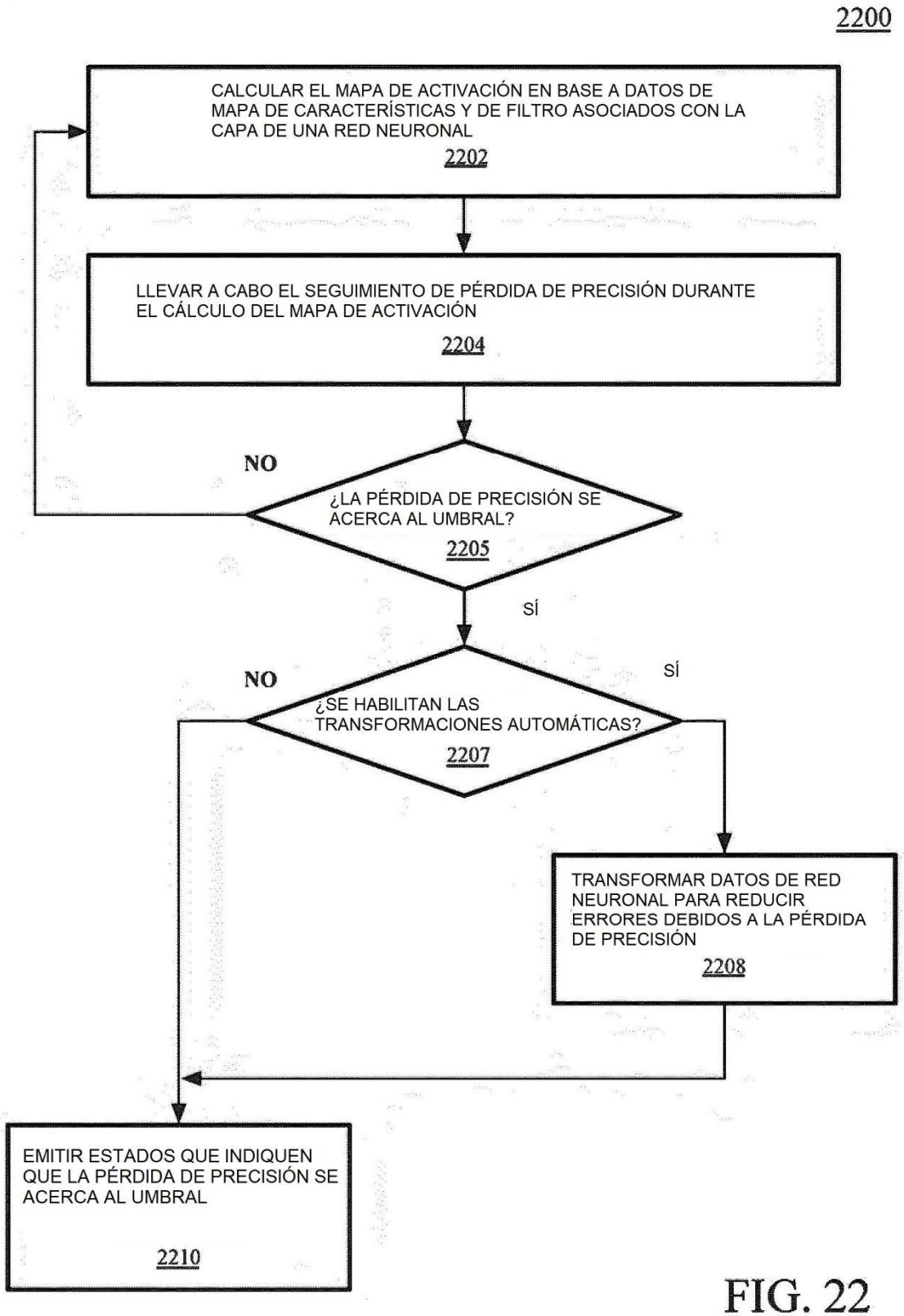

2200
CALCULAR EL MAPA DE ACTIVACIÓN EN BASE A DATOS DE MAPA DE CARACTERÍSTICAS Y DE FILTRO ASOCIADOS CON LA CAPA DE UNA RED NEURONAL
2202
LLEVAR A CABO EL SEGUIMIENTO DE PÉRDIDA DE PRECISIÓN DURANTE EL CÁLCULO DEL MAPA DE ACTIVACIÓN
2204
NO
¿LA PÉRDIDA DE PRECISIÓN SE ACERCA AL UMBRAL?
2205
SÍ
NO
¿SE HABILITAN LAS TRANSFORMACIONES AUTOMÁTICAS?
2207
SÍ
TRANSFORMAR DATOS DE RED NEURONAL PARA REDUCIR ERRORES DEBIDOS A LA PÉRDIDA DE PRECISIÓN
2208
EMITIR ESTADOS QUE INDIQUEN QUE LA PÉRDIDA DE PRECISIÓN SE ACERCA AL UMBRAL
2210

FIG. 22

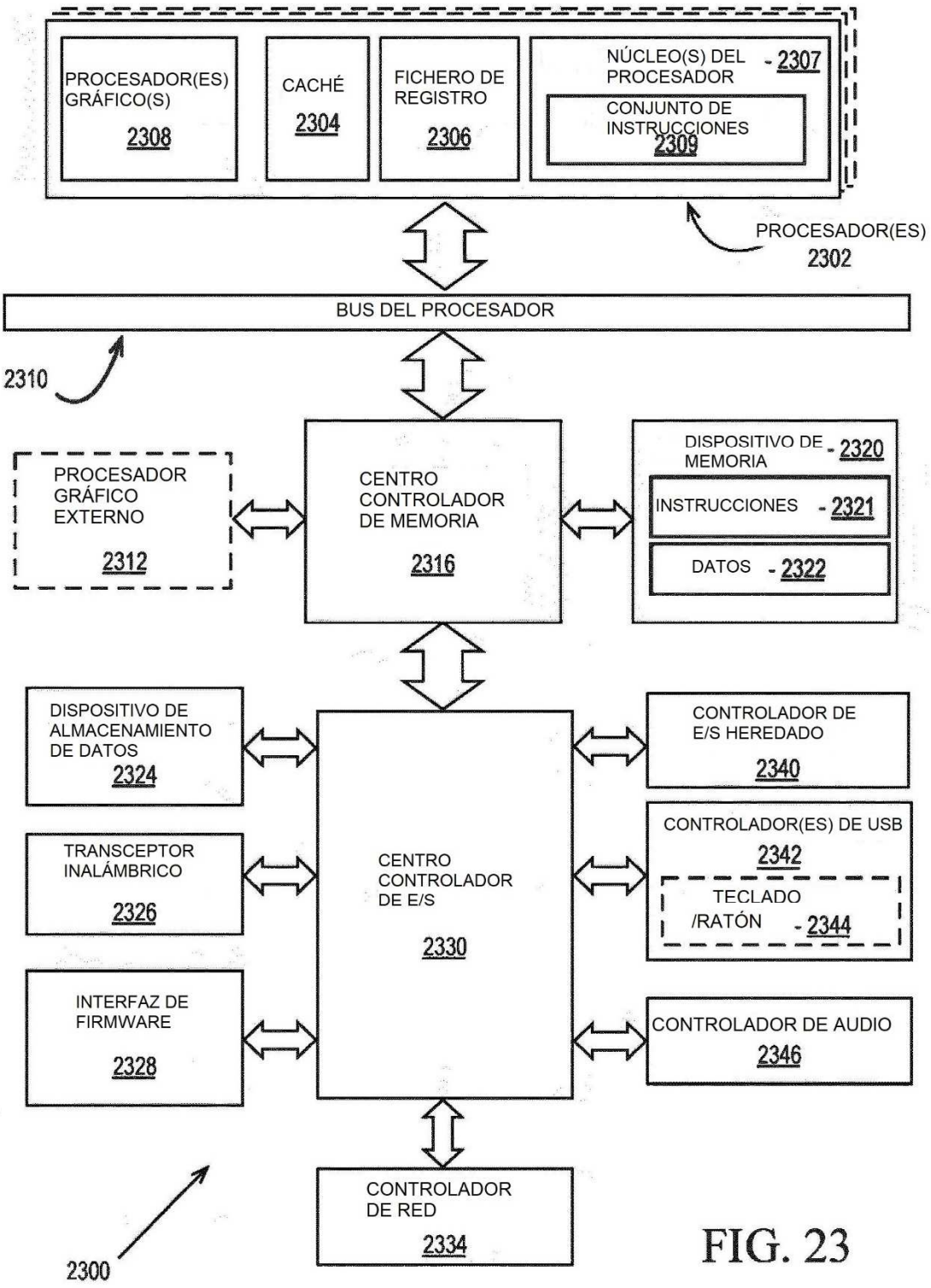

PROCESADOR(ES) GRÁFICO(S) 2308
CACHÉ 2304
FICHERO DE REGISTRO 2306
NÚCLEO(S) DEL PROCESADOR - 2307
CONJUNTO DE INSTRUCCIONES 2309
PROCESADOR(ES) 2302
BUS DEL PROCESADOR
2310
PROCESADOR GRÁFICO EXTERNO 2312
CENTRO CONTROLADOR DE MEMORIA 2316
DISPOSITIVO DE MEMORIA - 2320
INSTRUCCIONES - 2321
DATOS - 2322
DISPOSITIVO DE ALMACENAMIENTO DE DATOS 2324
TRANSCEPTOR INALÁMBRICO 2326
INTERFAZ DE FIRMWARE 2328
CENTRO CONTROLADOR DE E/S 2330
CONTROLADOR DE E/S HEREDADO 2340
CONTROLADOR(ES) DE USB 2342
TECLADO /RATÓN - 2344
CONTROLADOR DE AUDIO 2346
CONTROLADOR DE RED 2334
2300

FIG. 23

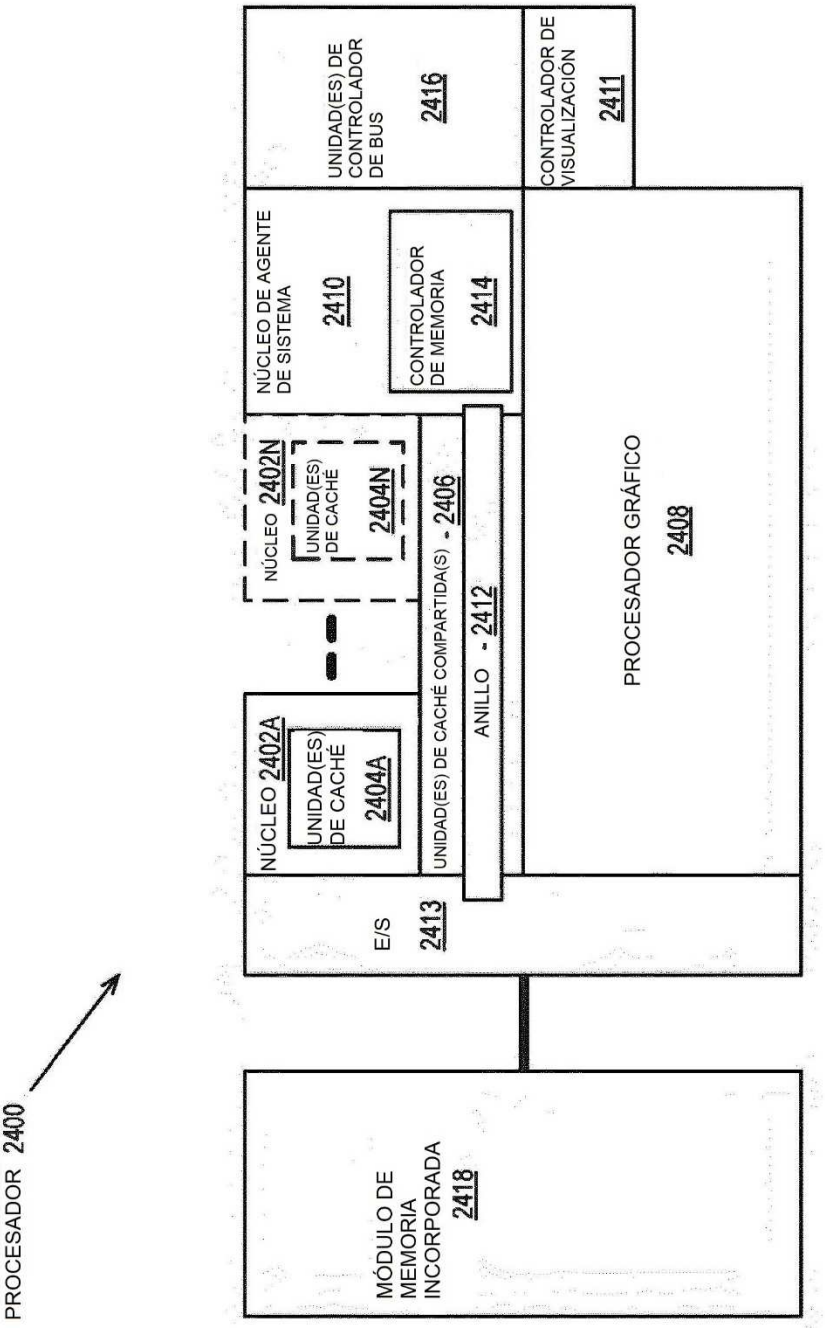

PROCESADOR 2400
MÓDULO DE MEMORIA INCORPORADA 2418
E/S 2413
NÚCLEO 2402A
UNIDAD(ES) DE CACHÉ 2404A
NÚCLEO 2402N
UNIDAD(ES) DE CACHÉ 2404N
UNIDAD(ES) DE CACHÉ COMPARTIDA(S) - 2406
ANILLO - 2412
NÚCLEO DE AGENTE DE SISTEMA 2410
CONTROLADOR DE MEMORIA 2414
PROCESADOR GRÁFICO 2408
UNIDAD(ES) DE CONTROLADOR DE BUS 2416
CONTROLADOR DE VISUALIZACIÓN 2411

FIG. 24

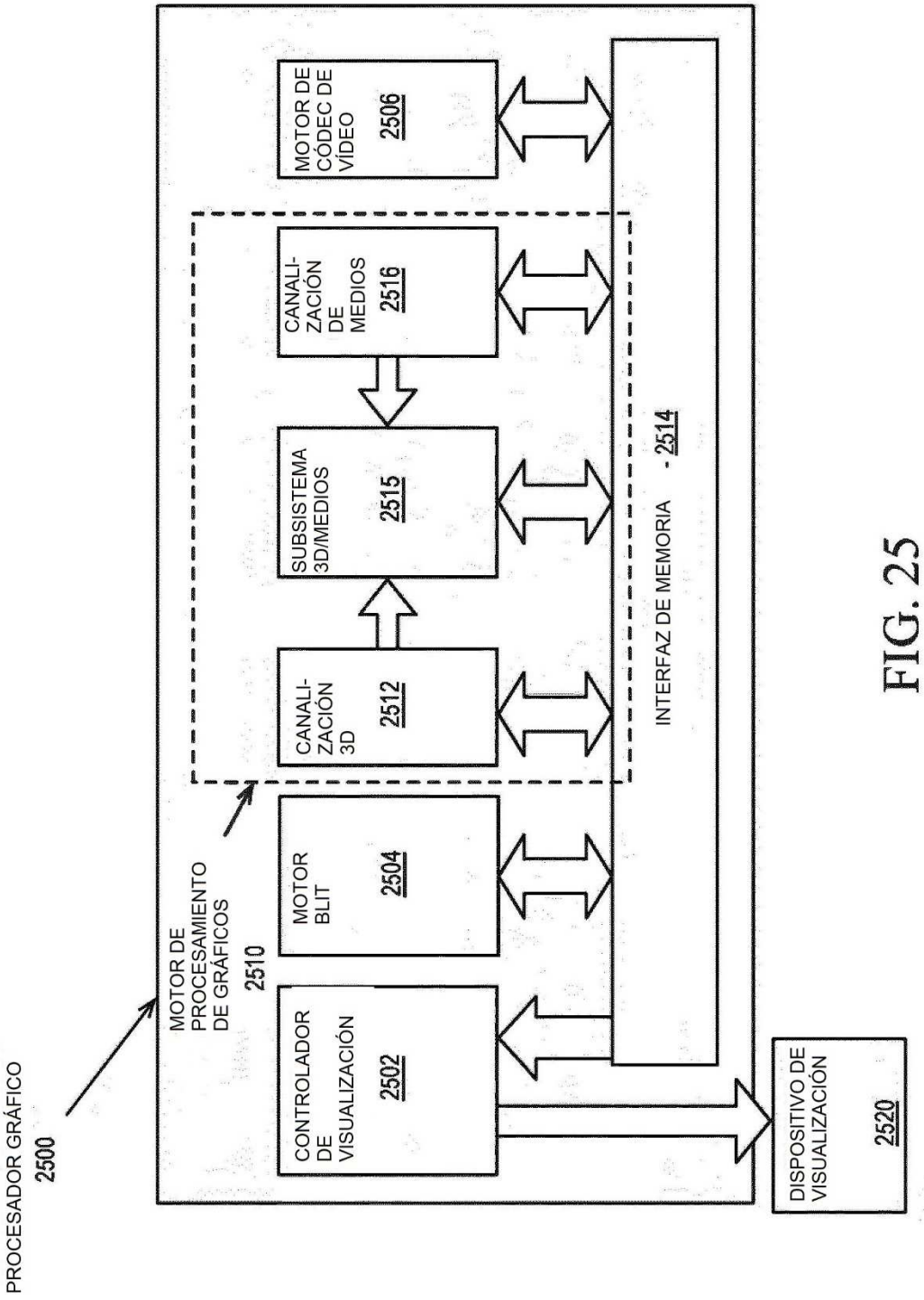

PROCESADOR GRÁFICO
2500
MOTOR DE PROCESAMIENTO DE GRÁFICOS
2510
CONTROLADOR DE VISUALIZACIÓN
2502
MOTOR BLIT
2504
CANALI-ZACIÓN 3D
2512
SUBSISTEMA 3D/MEDIOS
2515
CANALI-ZACIÓN DE MEDIOS
2516
MOTOR DE CÓDEC DE VÍDEO
2506
INTERFAZ DE MEMORIA - 2514
DISPOSITIVO DE VISUALIZACIÓN
2520

FIG. 25

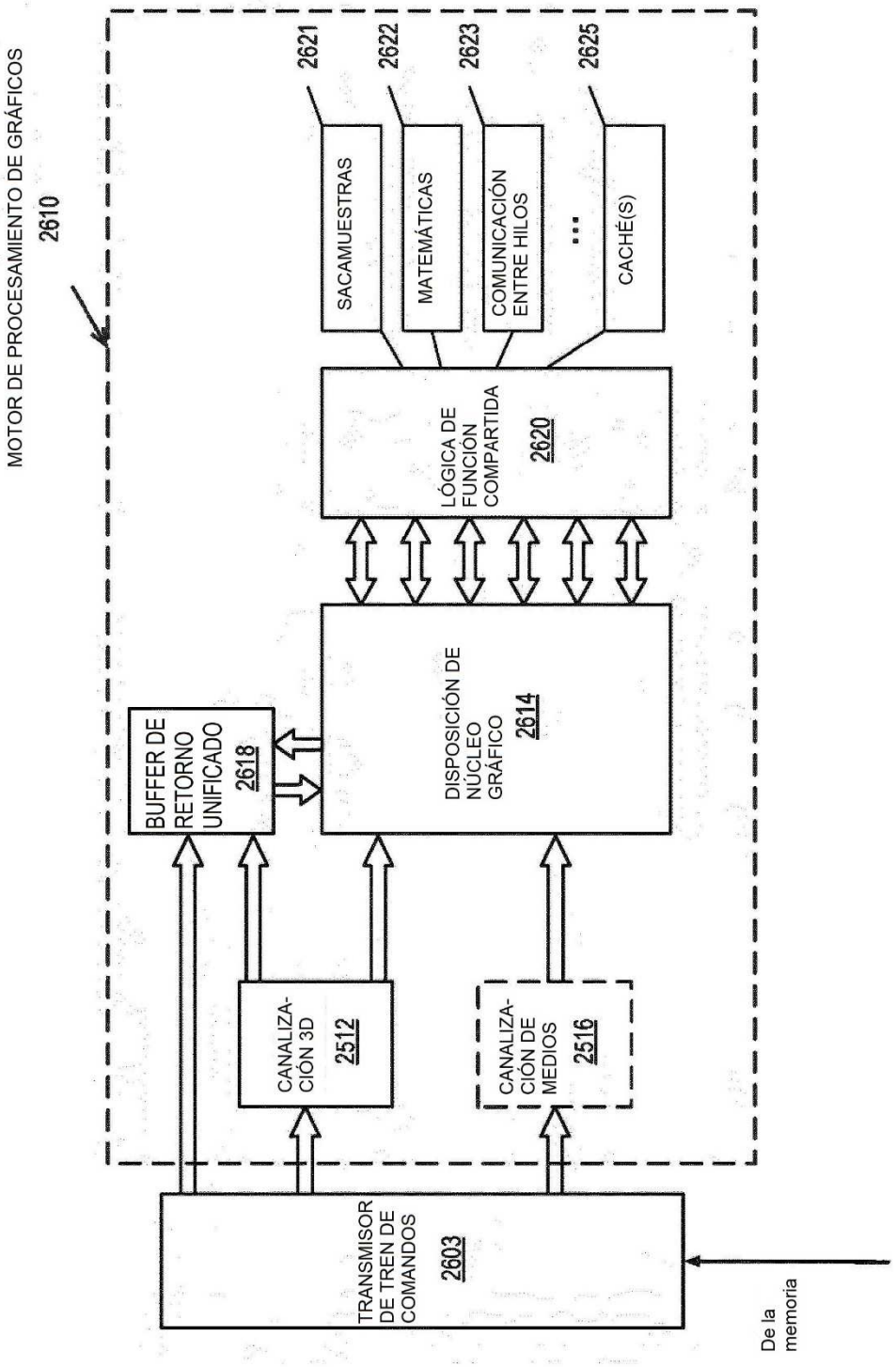

MOTOR DE PROCESAMIENTO DE GRÁFICOS
2610
2621
2622
2623
2625
SACAMUESTRAS
MATEMÁTICAS
COMUNICACIÓN ENTRE HILOS
...
CACHÉ(S)
LÓGICA DE FUNCIÓN COMPARTIDA
2620
DISPOSICIÓN DE NÚCLEO GRÁFICO
2614
BUFFER DE RETORNO UNIFICADO
2618
CANALIZA-CIÓN 3D
2512
CANALIZA-CIÓN DE MEDIOS
2516
TRANSMISOR DE TREN DE COMANDOS
2603
De la memoria

FIG. 26

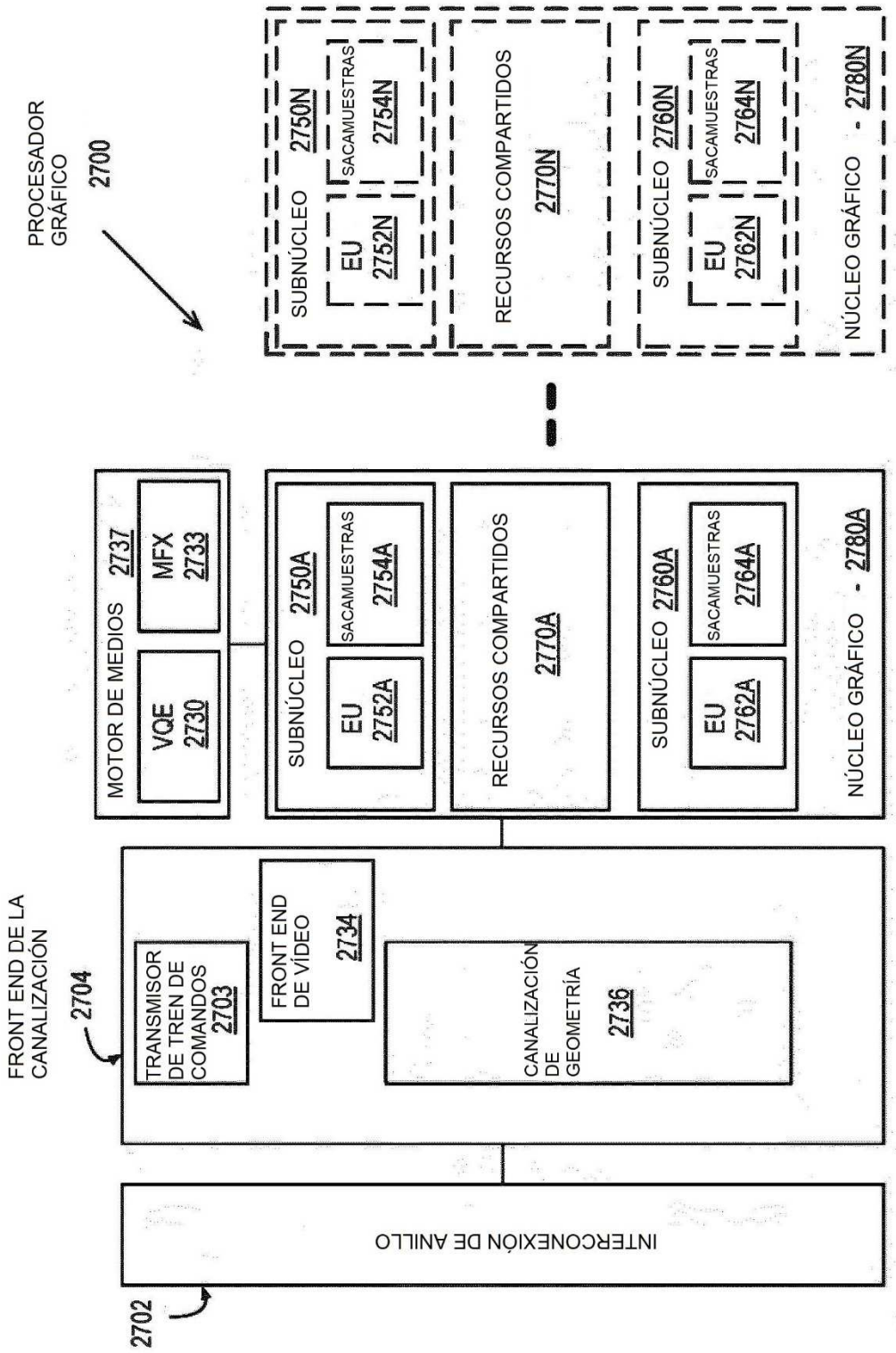

PROCESADOR GRÁFICO
2700
FRONT END DE LA CANALIZACIÓN
2704
TRANSMISOR DE TREN DE COMANDOS 2703
FRONT END DE VÍDEO 2734
CANALIZACIÓN DE GEOMETRÍA 2736
MOTOR DE MEDIOS 2737
VQE 2730
MFX 2733
SUBNÚCLEO 2750A
EU 2752A
SACAMUESTRAS 2754A
RECURSOS COMPARTIDOS 2770A
SUBNÚCLEO 2760A
EU 2762A
SACAMUESTRAS 2764A
NÚCLEO GRÁFICO - 2780A
SUBNÚCLEO 2750N
EU 2752N
SACAMUESTRAS 2754N
RECURSOS COMPARTIDOS 2770N
SUBNÚCLEO 2760N
EU 2762N
SACAMUESTRAS 2764N
NÚCLEO GRÁFICO - 2780N
INTERCONEXIÓN DE ANILLO
2702

FIG. 27

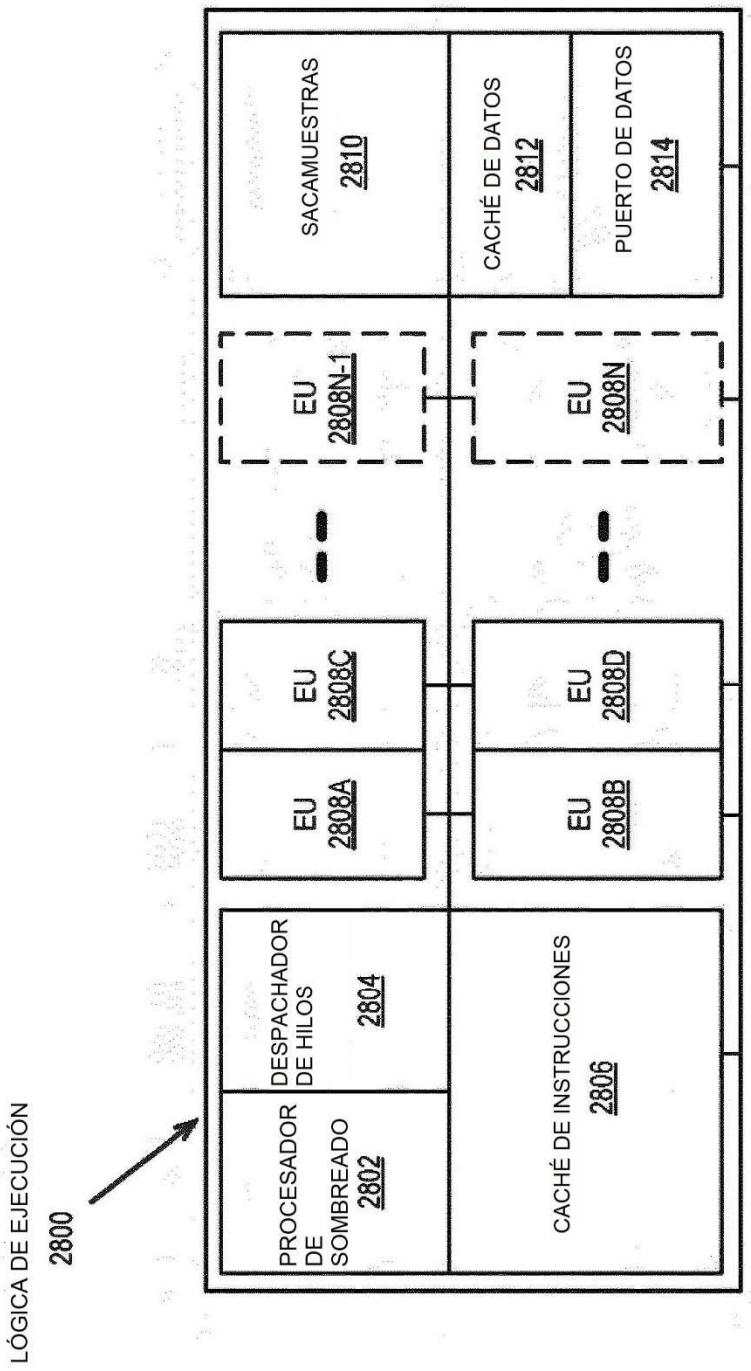

LÓGICA DE EJECUCIÓN
2800
PROCESADOR DE SOMBREADO
2802
DESPACHADOR DE HILOS
2804
CACHÉ DE INSTRUCCIONES
2806
EU 2808A
EU 2808B
EU 2808C
EU 2808D
EU 2808N-1
EU 2808N
SACAMUESTRAS
2810
CACHÉ DE DATOS
2812
PUERTO DE DATOS
2814

FIG. 28

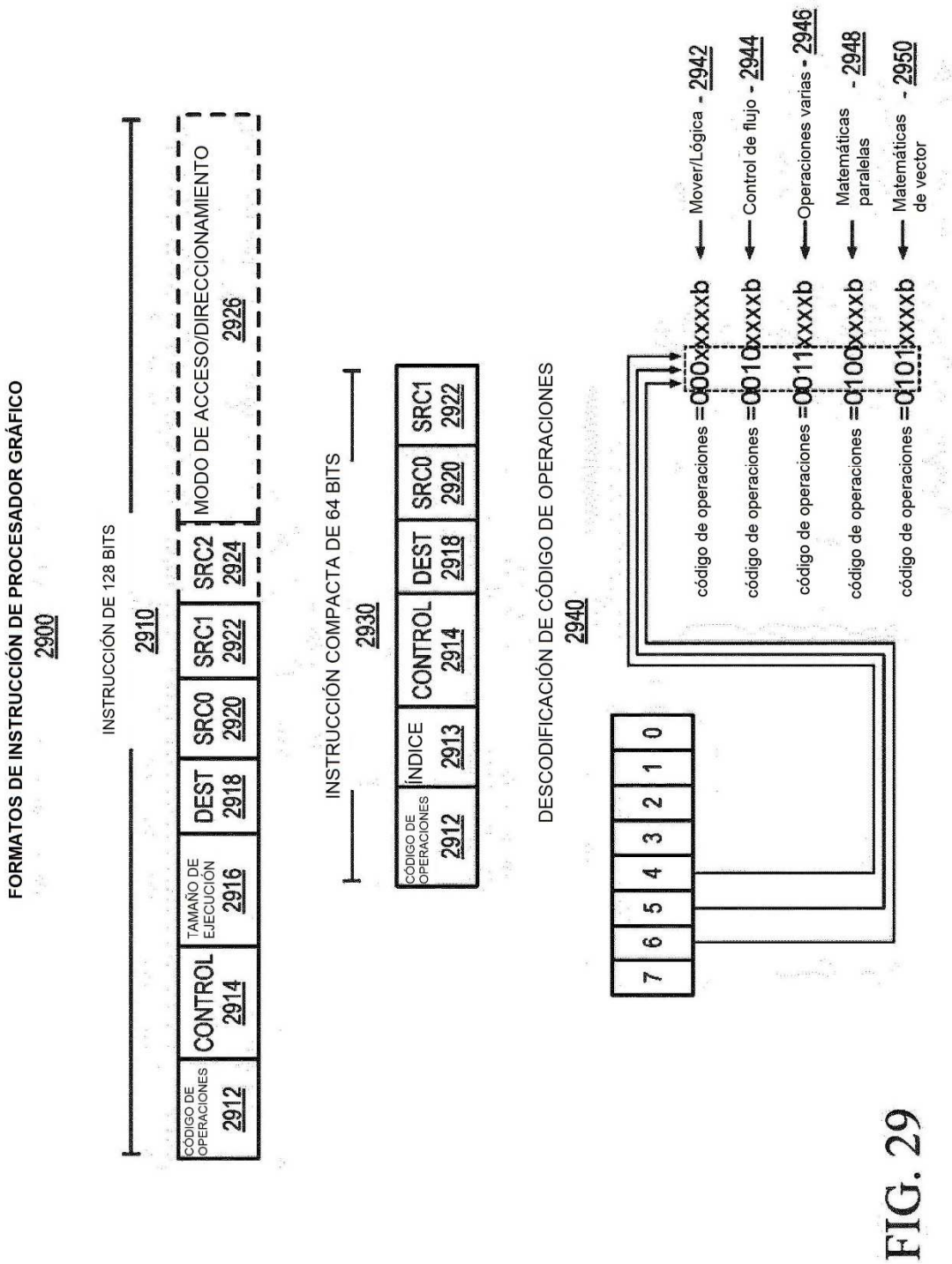

FORMATOS DE INSTRUCCIÓN DE PROCESADOR GRÁFICO
2900
INSTRUCCIÓN DE 128 BITS
2910
CÓDIGO DE OPERACIONES 2912
CONTROL 2914
TAMAÑO DE EJECUCIÓN 2916
DEST 2918
SRC0 2920
SRC1 2922
SRC2 2924
MODO DE ACCESO/DIRECCIONAMIENTO 2926
INSTRUCCIÓN COMPACTA DE 64 BITS
2930
CÓDIGO DE OPERACIONES 2912
ÍNDICE 2913
CONTROL 2914
DEST 2918
SRC0 2920
SRC1 2922
DESCODIFICACIÓN DE CÓDIGO DE OPERACIONES
2940
7 6 5 4 3 2 1 0
código de operaciones =000xxxxb
código de operaciones =001xxxxb
código de operaciones =010xxxxb
código de operaciones =011xxxxb
código de operaciones =100xxxxb
código de operaciones =101xxxxb
Mover/Lógica - 2942
Control de flujo - 2944
Operaciones varias - 2946
Matemáticas paralelas - 2948
Matemáticas de vector - 2950

FIG. 29

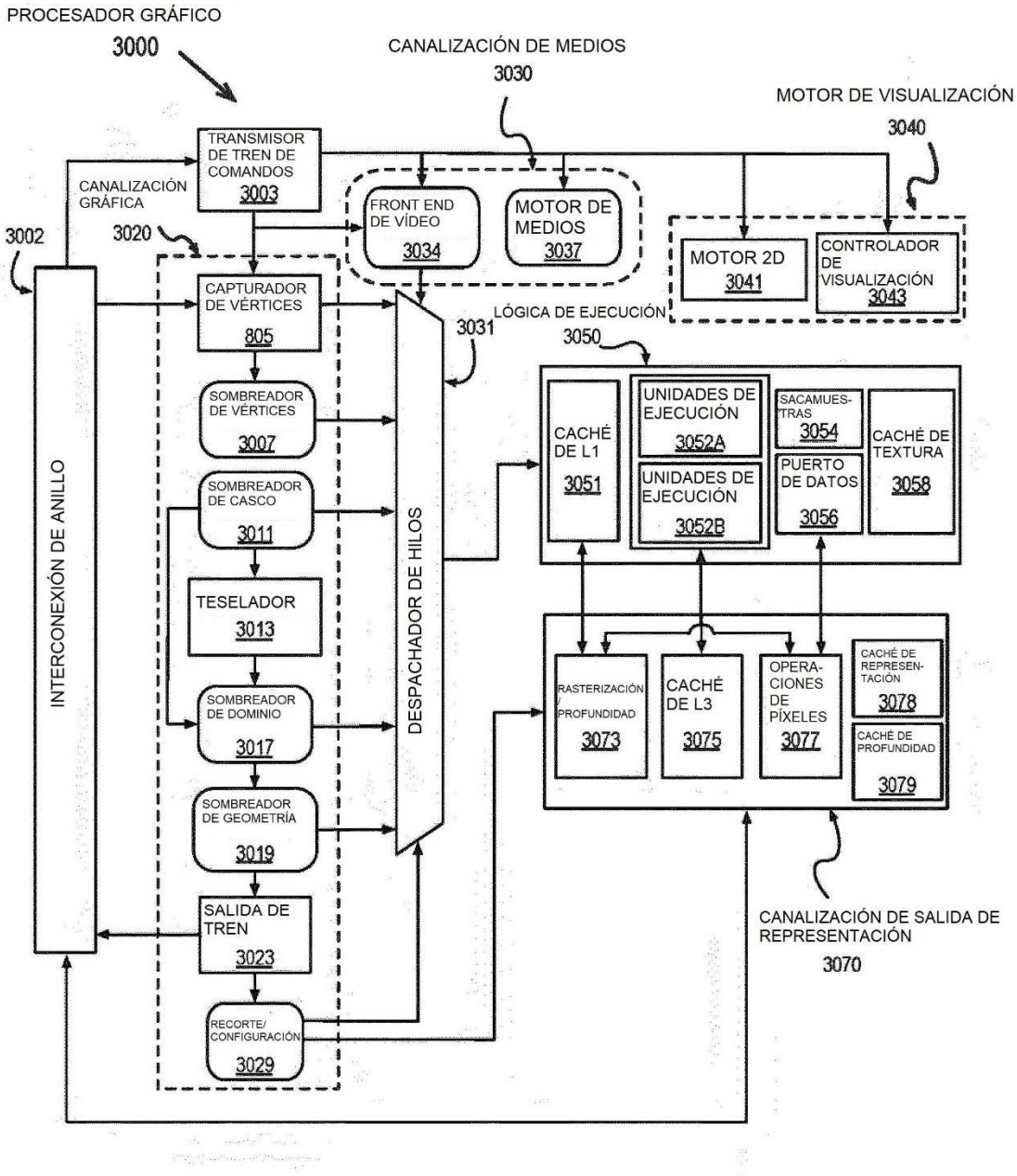
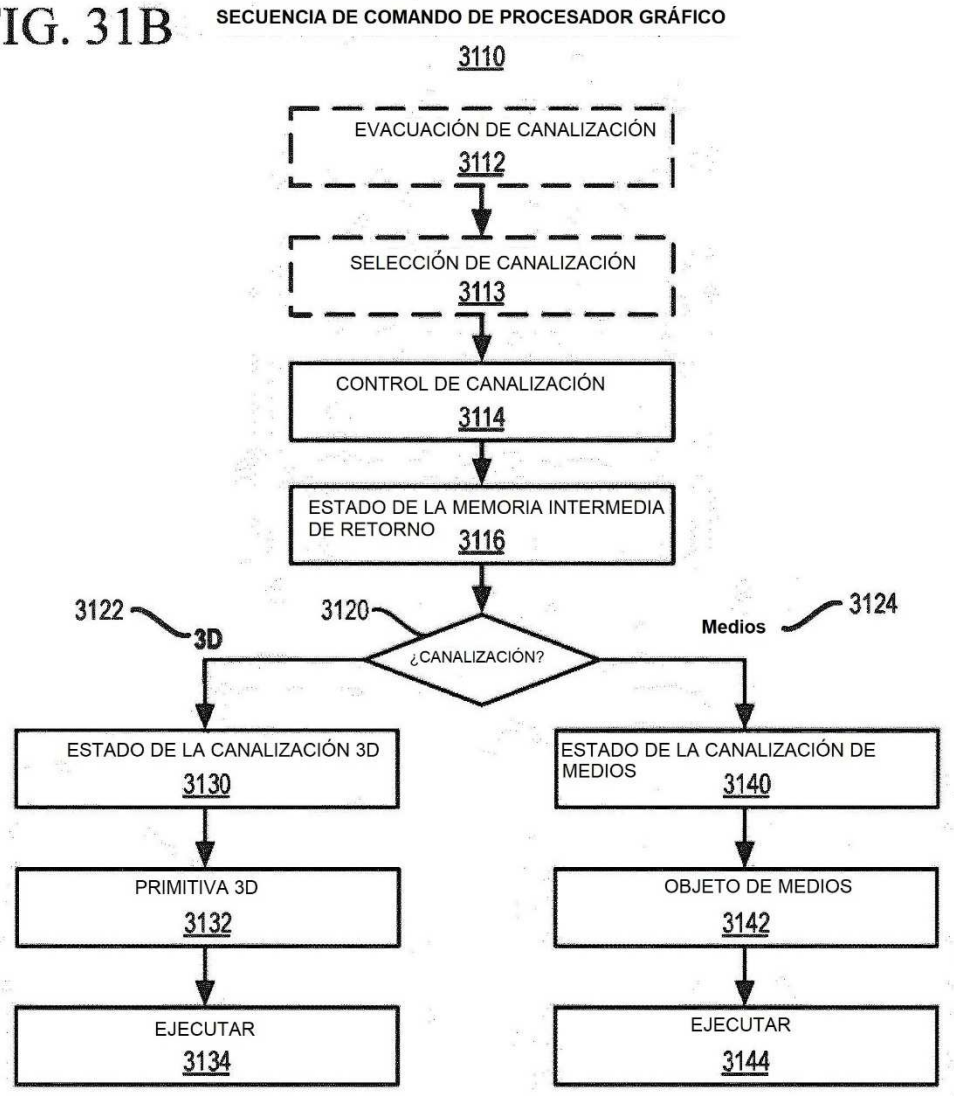

PROCESADOR GRÁFICO
3000
CANALIZACIÓN DE MEDIOS
3030
MOTOR DE VISUALIZACIÓN
3040
TRANSMISOR DE TREN DE COMANDOS
3003
CANALIZACIÓN GRÁFICA
3002
3020
FRONT END DE VÍDEO
3034
MOTOR DE MEDIOS
3037
MOTOR 2D
3041
CONTROLADOR DE VISUALIZACIÓN
3043
CAPTURADOR DE VÉRTICES
805
LÓGICA DE EJECUCIÓN
3031
3050
SOMBREADOR DE VÉRTICES
3007
SOMBREADOR DE CASCO
3011
TESELADOR
3013
SOMBREADOR DE DOMINIO
3017
SOMBREADOR DE GEOMETRÍA
3019
SALIDA DE TREN
3023
RECORTE/ CONFIGURACIÓN
3029
INTERCONEXIÓN DE ANILLO
DESPACHADOR DE HILOS
CACHÉ DE L1
3051
UNIDADES DE EJECUCIÓN
3052A
UNIDADES DE EJECUCIÓN
3052B
SACAMUESTRAS
3054
PUERTO DE DATOS
3056
CACHÉ DE TEXTURA
3058
RASTERIZACIÓN / PROFUNDIDAD
3073
CACHÉ DE L3
3075
OPERACIONES DE PÍXELES
3077
CACHÉ DE REPRESENTACIÓN
3078
CACHÉ DE PROFUNDIDAD
3079
CANALIZACIÓN DE SALIDA DE REPRESENTACIÓN
3070

FIG. 30

FIG. 31A

FIG. 31B



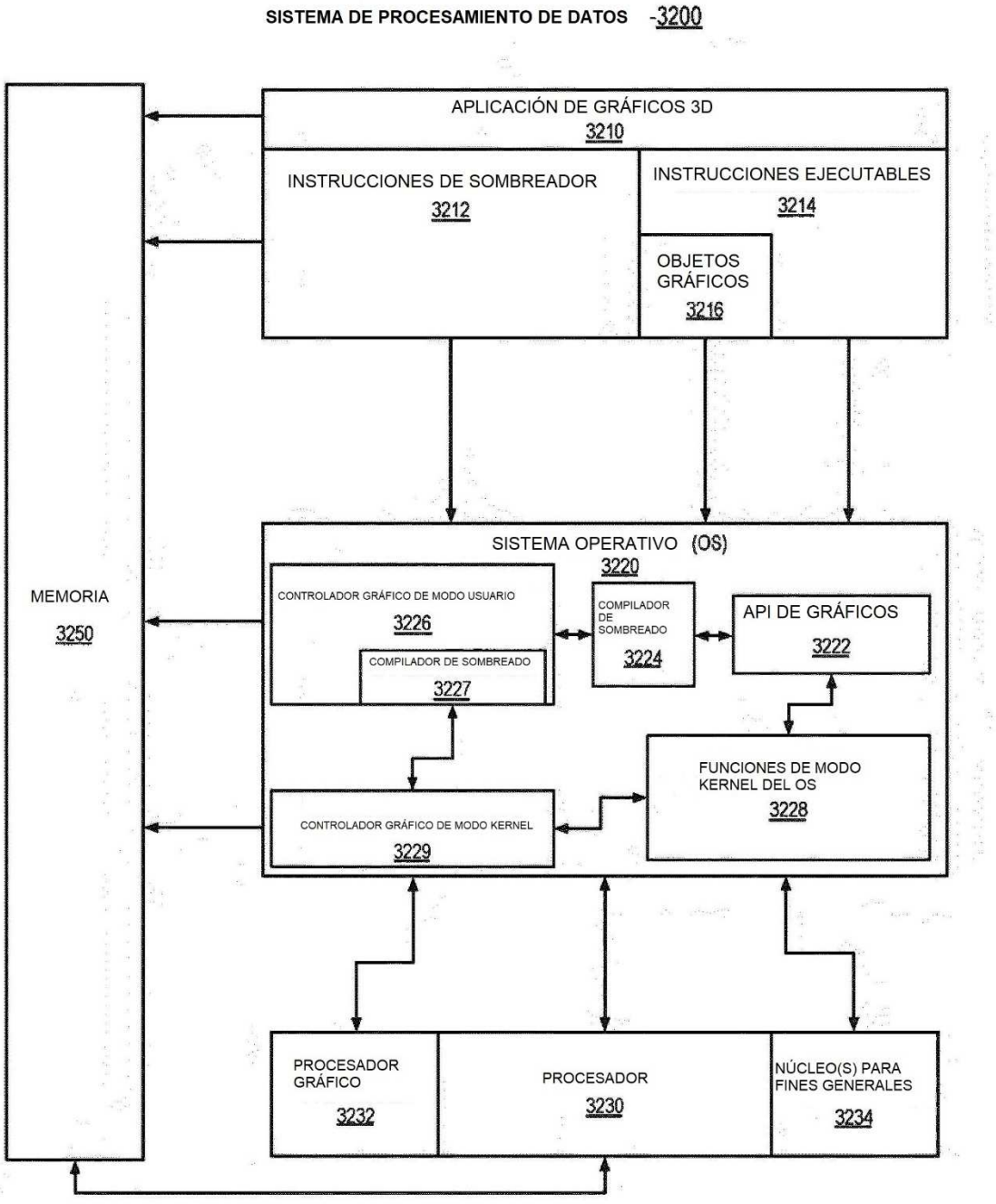

SISTEMA DE PROCESAMIENTO DE DATOS -3200
APLICACIÓN DE GRÁFICOS 3D
3210
INSTRUCCIONES DE SOMBREADOR
3212
INSTRUCCIONES EJECUTABLES
3214
OBJETOS GRÁFICOS
3216
MEMORIA
3250
SISTEMA OPERATIVO (OS)
3220
CONTROLADOR GRÁFICO DE MODO USUARIO
3226
COMPILADOR DE SOMBREADO
3227
COMPILADOR DE SOMBREADO
3224
API DE GRÁFICOS
3222
FUNCIONES DE MODO KERNEL DEL OS
3228
CONTROLADOR GRÁFICO DE MODO KERNEL
3229
PROCESADOR GRÁFICO
3232
PROCESADOR
3230
NÚCLEO(S) PARA FINES GENERALES
3234

FIG. 32

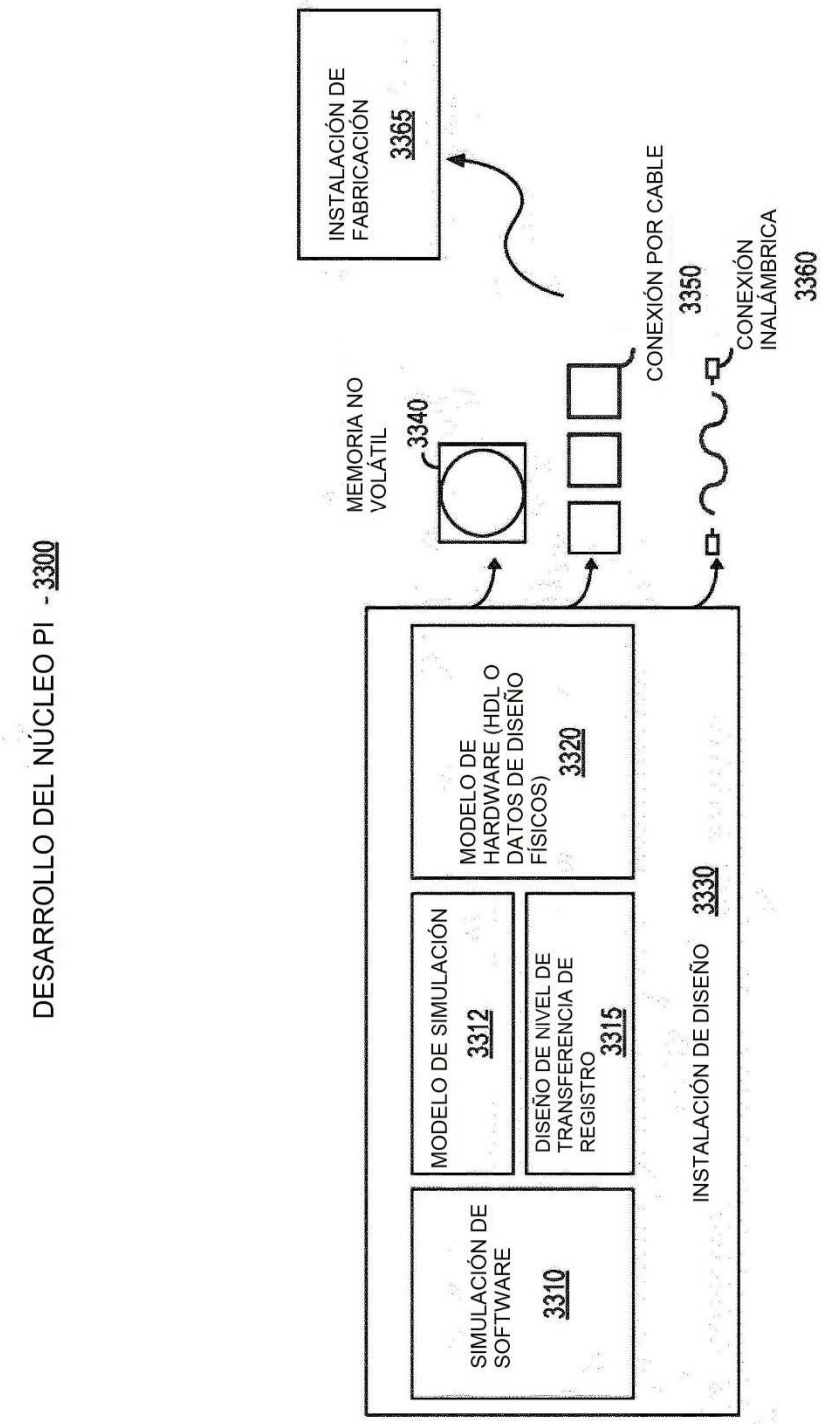

DESARROLLO DEL NÚCLEO PI - 3300
SIMULACIÓN DE SOFTWARE
3310
MODELO DE SIMULACIÓN
3312
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO
3315
MODELO DE HARDWARE (HDL O DATOS DE DISEÑO FÍSICOS)
3320
INSTALACIÓN DE DISEÑO 3330
MEMORIA NO VOLÁTIL
3340
CONEXIÓN POR CABLE
3350
CONEXIÓN INALÁMBRICA
3360
INSTALACIÓN DE FABRICACIÓN
3365

FIG. 33

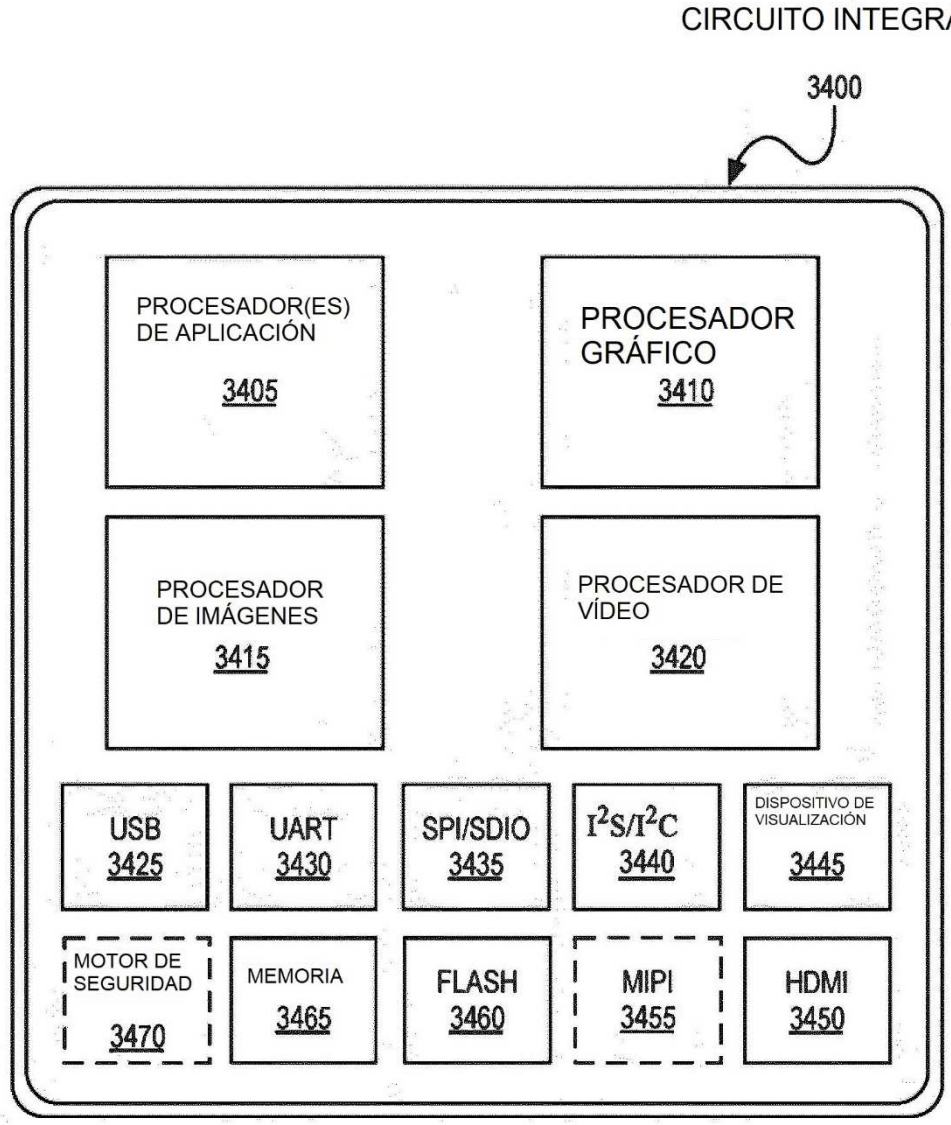

CIRCUITO INTEGRADO SOC
3400
PROCESADOR(ES) DE APLICACIÓN
3405
PROCESADOR GRÁFICO
3410
PROCESADOR DE IMÁGENES
3415
PROCESADOR DE VÍDEO
3420
USB
3425
UART
3430
SPI/SDIO
3435
I2S/I2C
3440
DISPOSITIVO DE VISUALIZACIÓN
3445
MOTOR DE SEGURIDAD
3470
MEMORIA
3465
FLASH
3460
MIPI
3455
HDMI
3450

FIG. 34

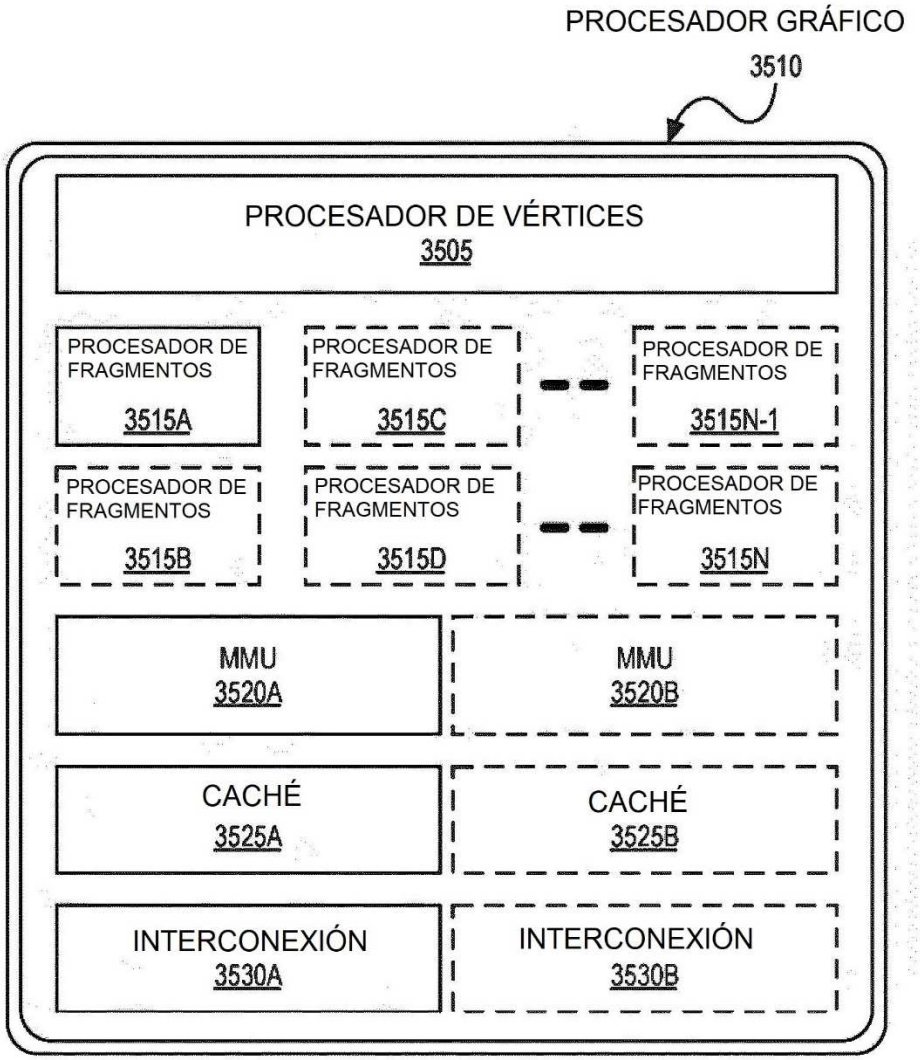

PROCESADOR GRÁFICO
3510
PROCESADOR DE VÉRTICES
3505
PROCESADOR DE FRAGMENTOS
3515A
PROCESADOR DE FRAGMENTOS
3515C
PROCESADOR DE FRAGMENTOS
3515N-1
PROCESADOR DE FRAGMENTOS
3515B
PROCESADOR DE FRAGMENTOS
3515D
PROCESADOR DE FRAGMENTOS
3515N
MMU
3520A
MMU
3520B
CACHÉ
3525A
CACHÉ
3525B
INTERCONEXIÓN
3530A
INTERCONEXIÓN
3530B

FIG. 35

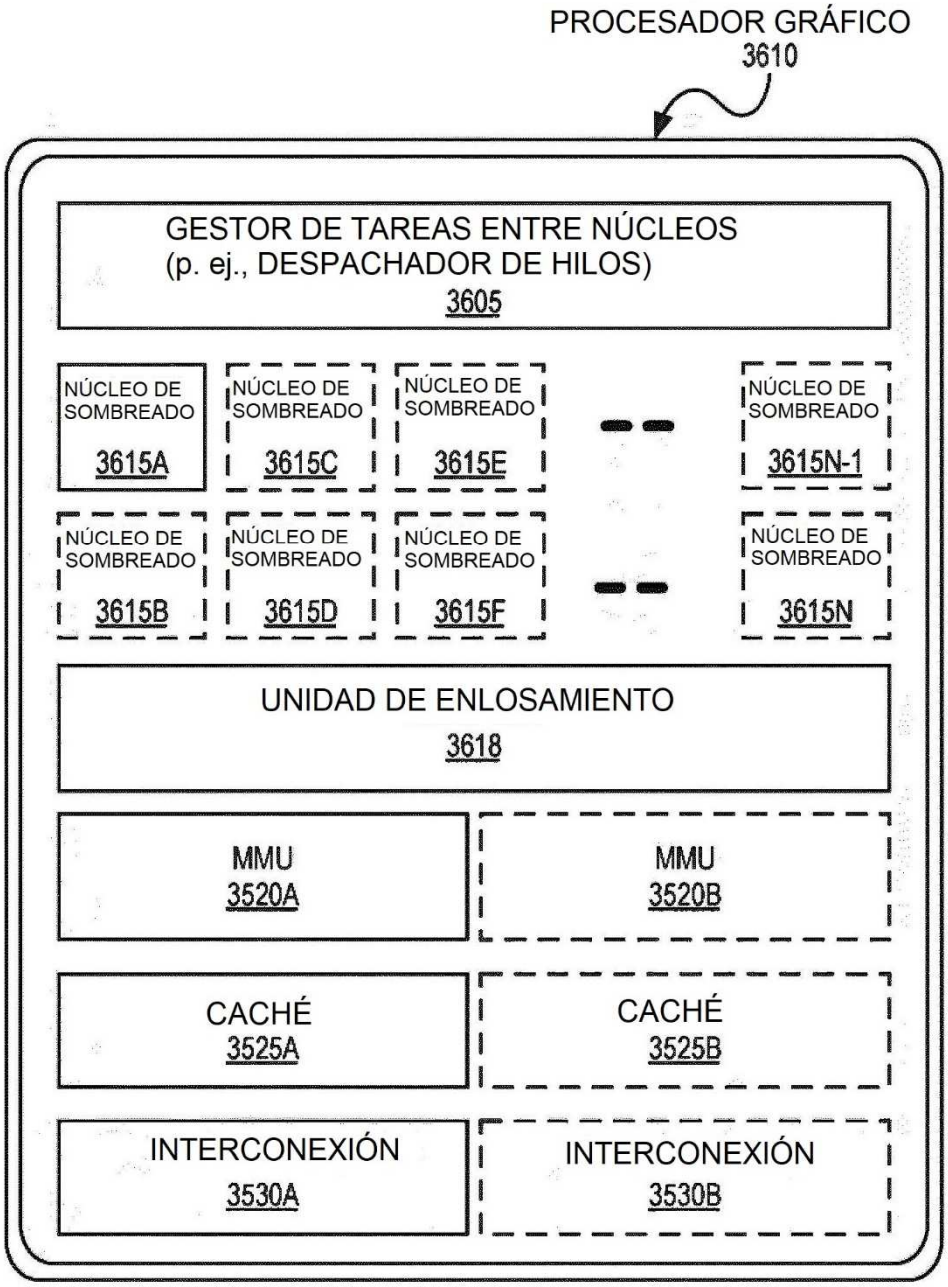

PROCESADOR GRÁFICO
3610
GESTOR DE TAREAS ENTRE NÚCLEOS
(p. ej., DESPACHADOR DE HILOS)
3605
NÚCLEO DE SOMBREADO
3615A
NÚCLEO DE SOMBREADO
3615C
NÚCLEO DE SOMBREADO
3615E
NÚCLEO DE SOMBREADO
3615N-1
NÚCLEO DE SOMBREADO
3615B
NÚCLEO DE SOMBREADO
3615D
NÚCLEO DE SOMBREADO
3615F
NÚCLEO DE SOMBREADO
3615N
UNIDAD DE ENLOSAMIENTO
3618
MMU
3520A
MMU
3520B
CACHÉ
3525A
CACHÉ
3525B
INTERCONEXIÓN
3530A
INTERCONEXIÓN
3530B

FIG. 36