(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 929 797**

(51) Int. CI.:

G06F 7/483 (2006.01)
G06F 7/544 (2006.01)
G06F 9/30 (2008.01)
G06N 3/04 (2006.01)
G06N 3/063 (2006.01)
G06N 3/08 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **26.03.2018 E 19214829 (4)**

(97) Fecha y número de publicación de la concesión europea: **17.08.2022 EP 3637247**

(54) Título: **Instrucciones y lógica para realizar operaciones de coma flotante y de números enteros para el aprendizaje automático**

(30) Prioridad:

28.04.2017 US 201762491699 P
18.10.2017 US 201715787129

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
01.12.2022

(73) Titular/es:

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

(72) Inventor/es:

KAUL, HIMANSHU;
ANDERS, MARK A.;
MATHEW, SANU K.;
YAO, ANBANG;
RAY, JOYDEEP;
TANG, PING T.;
STRICKLAND, MICHAEL S.;
CHEN, XIAOMING;
SHPEISMAN, TATIANA;
APPU, ABHISHEK R.;
KOKER, ALTUG;
SINHA, KAMAL;
VEMBU, BALAJI;
NURVITADHI, ERIKO;
BARIK, RAJKISHORE;
LIN, TSUNG-HAN;
RANGANATHAN, VASANTH;
JAHAGIRDAR, SANJEEV y
GALOPPO VON BORRIES, NICOLAS

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Instrucciones y lógica para realizar operaciones de coma flotante y de números enteros para el aprendizaje automático

Campo

Las realizaciones se refieren, en general, al procesamiento de datos y, más particularmente, al procesamiento de datos a través de una unidad de procesamiento de gráficos de propósito general.

Antecedentes de la descripción

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

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

El documento XP 055509917 "*Mixed-Precision Programming with CUDA-8*" se refiere a una arquitectura de GPU NVIDIA Pascal que tiene capacidades de cómputo nuevas. Asimismo, se proporcionan detalles acerca de la coma flotante de precisión media y acerca del desempeño que se puede lograr en las GPU Pascal usando el cómputo vectorial FP16 e INT8. Además, se analizan las capacidades de cómputo de precisión mixta proporcionadas por diversas API y bibliotecas de plataforma CUDA.

Breve descripción de los dibujos

De modo que las características de la presente invención se puedan entender con detalle, se puede tener una descripción más particular de la invención por referencia a realizaciones, algunas de las cuales se ilustran en los dibujos adjuntos. Sin embargo, se ha de observar que los dibujos adjuntos solo ilustran realizaciones habituales y, por lo tanto, no han de considerarse limitantes del alcance de todas las realizaciones.

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

las **Figuras 2A-2D** ilustran componentes de procesador paralelo, de acuerdo con una realización;

las **Figuras 3A-3B** son diagramas de bloques de multiprocesadores de gráficos, de acuerdo con realizaciones;

las **Figuras 4A-4F** ilustran una arquitectura ilustrativa en la que una pluralidad de GPU están acopladas de manera comunicativa a una pluralidad de procesadores de múltiples núcleos;

la **Figura 5** ilustra una canalización de procesamiento de gráficos, de acuerdo con una realización;

la **Figura 6** ilustra una pila de software de aprendizaje automático, de acuerdo con una realización;

la **Figura 7** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela, de acuerdo con una realización;

la **Figura 8** ilustra un sistema informático de múltiples GPU, de acuerdo con una realización;

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

la **Figura 10** ilustra una red neuronal recurrente ilustrativa;

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

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

la **Figura 13** ilustra un sistema en un chip (SOC) de inferenciación ilustrativo adecuado para realizar una inferenciación usando un modelo entrenado;

la **Figura 14** es un diagrama de bloques de una unidad de multiprocesador, de acuerdo con una realización;

las **Figuras 15A-15B** ilustran diseños para unidades de lógica para realizar operaciones de multiplicación - suma fusionada de números enteros y de coma flotante, de acuerdo con una realización;

la **Figura 16** ilustra una unidad de lógica de multiplicación - suma fusionada que tiene una ruta de datos de coma flotante y de números enteros combinada, de acuerdo con una realización;

las **Figuras 17A-17B** ilustran unidades de lógica que incluyen circuitos de cómputo combinado para realizar operaciones de multiplicación - acumulación fusionada de coma flotante y de números enteros, de acuerdo con una realización;

las **Figuras 18A-18B** ilustran un sistema de procesamiento de datos y unidades de cómputo y de lógica asociadas que realizan operaciones de entrenamiento y de inferenciación aceleradas para el aprendizaje automático;

la **Figura 19** ilustra detalles del módulo de instrucción de activación, de acuerdo con una realización;

la **Figura 20** ilustra la unidad de cuantificación estocástica, de acuerdo con una realización;

la **Figura 21** ilustra un módulo de codificación y de configuración de FPU, de acuerdo con una realización;

la **Figura 22** ilustra una lógica para procesar una instrucción usando una unidad de cómputo configurable dinámicamente, de acuerdo con una realización;

las **Figuras 23A-23B** son diagramas de flujo que ilustran una lógica para realizar operaciones de cómputo dispersas dentro de una GPGPU proporcionada por realizaciones descritas en el presente documento;

la **Figura 24** es un diagrama de bloques de un sistema de procesamiento, de acuerdo con una realización;

la **Figura 25** es un diagrama de bloques de un procesador de acuerdo con una realización;

la **Figura 26** es un diagrama de bloques de un procesador de gráficos, de acuerdo con una realización;

la **Figura 27** es un diagrama de bloques de un motor de procesamiento de gráficos de un procesador de gráficos de acuerdo con algunas realizaciones;

la **Figura 28** es un diagrama de bloques de un procesador de gráficos proporcionado por una realización adicional;

la **Figura 29** ilustra una lógica de ejecución de hilos que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones;

la **Figura 30** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos de acuerdo con algunas realizaciones;

la **Figura 31** es un diagrama de bloques de un procesador de gráficos de acuerdo con otra realización.

Las **Figuras 32A-32B** ilustran un formato de comando de procesador de gráficos y una secuencia de comandos, de acuerdo con algunas realizaciones;

la **Figura 33** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos de acuerdo con algunas realizaciones;

la **Figura 34** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP, de acuerdo con una realización;

la **Figura 35** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ilustrativo, de acuerdo con una realización;

la **Figura 36** es un diagrama de bloques que ilustra un procesador de gráficos adicional, de acuerdo con una realización; y

la **Figura 37** es un diagrama de bloques que ilustra un procesador de gráficos ilustrativo adicional de un circuito integrado de sistema en un chip, de acuerdo con una realización.

Descripción detallada

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

En la descripción siguiente, se exponen numerosos detalles específicos para proporcionar un entendimiento más minucioso. Sin embargo, será evidente para un experto en la materia que las realizaciones descritas en el presente documento se pueden poner en práctica sin uno o más de estos detalles específicos. En otras instancias, no se han descrito características bien conocidas para evitar complicar los detalles de las presentes realizaciones. La invención se define por las reivindicaciones adjuntas.

Vista global de sistema

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático 100 configurado para implementar uno o más aspectos de las realizaciones descritas en el presente documento. El sistema informático 100 incluye un subsistema de procesamiento 101 que tiene uno o más procesadores 102 y una memoria de sistema 104 que se comunica a través de una ruta de interconexión que puede incluir un concentrador de memoria 105. El concentrador de memoria 105 puede ser un componente separado dentro de un componente de conjunto de chips o se puede integrar dentro de los uno o más procesadores 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 a través de un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede habilitar que el sistema informático 100 reciba una entrada desde uno o más dispositivos de entrada 108. Adicionalmente, el concentrador de E/S 107 puede habilitar que un controlador de visualización, que se puede incluir en los uno o más procesadores 102, proporcione salidas a uno o más dispositivos de visualización 110A. En una realización, los uno o más dispositivos de visualización 110A acoplados con el concentrador de E/S 107 pueden incluir un dispositivo de visualización local, interno o embebido.

En una realización, el subsistema de procesamiento 101 incluye uno o más procesadores paralelos 112 acoplados al concentrador de memoria 105 a través de un bus u otro enlace de comunicación 113. El enlace de comunicación 113 puede ser uno de cualquier número de tecnologías o protocolos de enlace de comunicación basados en normas, tales como, pero sin limitación, PCI Express, o puede ser una interfaz de comunicaciones o tejido de comunicaciones específico de distribuidor. En una realización, los uno o más procesadores paralelos 112 forman un sistema de procesamiento paralelo o vectorial de enfoque computacional que incluye un gran número de núcleos de procesamiento y/o agrupaciones de procesamiento, tal como un procesador de muchos núcleos integrados (MIC). En una realización, los uno o más procesadores paralelos 112 forman un subsistema de procesamiento de gráficos que puede emitir píxeles a uno de los uno o más dispositivos de visualización 110A acoplados a través del concentrador de E/S 107. Los uno o más procesadores paralelos 112 pueden incluir también un controlador de visualización y una interfaz de visualización (no mostrados) para habilitar una conexión directa a uno o más dispositivos de visualización 110B.

Dentro del subsistema de E/S 111, una unidad de almacenamiento de sistema 114 se puede conectar al concentrador de E/S 107 para proporcionar un mecanismo de almacenamiento para el sistema informático 100. Se puede usar un conmutador de E/S 116 para proporcionar un mecanismo de interfaz para habilitar conexiones entre el concentrador de E/S 107 y otros componentes, tales como un adaptador de red 118 y/o un adaptador de red inalámbrico 119 que se pueden integrar en la plataforma, y diversos otros dispositivos que se pueden añadir a través de uno o más dispositivos de complemento 120. El adaptador de red 118 puede ser un adaptador de Ethernet u otro adaptador de red cableado. El adaptador de red inalámbrico 119 puede incluir uno o más de un dispositivo de red de Wi-Fi, de Bluetooth, de comunicación de campo cercano (NFC) o de otro tipo que incluye una o más radios inalámbricas.

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

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

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

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

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

En una realización, el procesador paralelo 200 incluye una unidad de procesamiento paralelo 202. La unidad de procesamiento paralelo incluye una unidad de E/S 204 que habilita la comunicación con otros dispositivos, incluyendo otras instancias de la unidad de procesamiento paralelo 202. La unidad de E/S 204 se puede conectar directamente a otros dispositivos. En una realización, la unidad de E/S 204 se conecta con otros dispositivos a través del uso de una interfaz de concentrador o de conmutador, tal como un concentrador de memoria 105. Las conexiones entre el concentrador de memoria 105 y la unidad de E/S 204 forman un enlace de comunicación 113. Dentro de la unidad de procesamiento paralelo 202, la unidad de E/S 204 se conecta con una interfaz de anfitrión 206 y una barra transversal de memoria 216, en donde la interfaz de anfitrión 206 recibe comandos dirigidos a realizar operaciones de procesamiento y la barra transversal de memoria 216 recibe comandos dirigidos a realizar operaciones de memoria.

Cuando la interfaz de anfitrión 206 recibe una memoria intermedia de comandos a través de la unidad de E/S 204, la interfaz de anfitrión 206 puede dirigir operaciones de trabajo para realizar esos comandos a un extremo frontal 208. En una realización, el extremo frontal 208 se acopla con un planificador 210, que está configurado para distribuir comandos u otros elementos de trabajo a una matriz de agrupaciones de procesamiento 212. En una realización, el planificador 210 garantiza que la matriz de agrupaciones de procesamiento 212 está configurada apropiadamente y en un estado válido antes de que las tareas se distribuyan a las agrupaciones de procesamiento de la matriz de agrupaciones de procesamiento 212. En una realización, el planificador 210 se implementa a través de lógica de firmware que se ejecuta en un microcontrolador. El planificador implementado por microcontrolador 210 se puede configurar para realizar operaciones de planificación y de distribución de trabajo complejas con granularidad gruesa y fina, lo que habilita un rápido otorgamiento de prioridad y conmutación de contexto de hilos que se ejecutan en la matriz de procesamiento 212. En una realización, el software de anfitrión puede probar cargas de trabajo para su planificación en la matriz de procesamiento 212 a través de uno de múltiples llamadores de procesamiento de gráficos. Las cargas de trabajo se pueden distribuir entonces automáticamente a lo largo de la matriz de procesamiento 212 por la lógica del planificador 210 dentro del microcontrolador de planificador.

La matriz de agrupaciones de procesamiento 212 puede incluir hasta "N" agrupaciones de procesamiento (por ejemplo, de la agrupación 214A, la agrupación 214B a la agrupación 214N). Cada agrupación 214A-214N de la matriz de

agrupaciones de procesamiento 212 puede ejecutar un gran número de hilos concurrentes. El planificador 210 puede asignar trabajo a las agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 usando diversos algoritmos de planificación y/o de distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surja para cada tipo de programa o cómputo. La planificación puede ser manejada dinámicamente por el planificador 210, o puede ser asistida, en parte, por lógica de compilador durante la compilación de lógica de programa configurada para su ejecución por la matriz de agrupaciones de procesamiento 212. En una realización, se pueden asignar diferentes agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 para procesar diferentes tipos de programas o para realizar diferentes tipos de cómputos.

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

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

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

Durante el funcionamiento, la matriz de agrupaciones de procesamiento 212 puede recibir tareas de procesamiento a ejecutar a través del planificador 210, que recibe comandos que definen tareas de procesamiento desde el extremo frontal 208. Para operaciones de procesamiento de gráficos, las tareas de procesamiento pueden incluir índices de datos a procesar, por ejemplo, datos de superficie (parche), datos de primitiva, datos de vértice y/o datos de píxel, así como parámetros de estado y comandos que definen cómo se han de procesar los datos (por ejemplo, qué programa se va a ejecutar). El planificador 210 se puede configurar para extraer los índices que corresponden a las tareas o puede recibir los índices desde el extremo frontal 208. El extremo frontal 208 se puede configurar para garantizar que la matriz de agrupaciones de procesamiento 212 está configurada en un estado válido antes de que se inicie la carga de trabajo especificada por memorias intermedias de comando de entrada (por ejemplo, memorias intermedias de lotes, memorias intermedias de inserción, etc.).

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

En diversas realizaciones, las unidades de memoria 224A-224N pueden incluir diversos tipos de dispositivos de

memoria, incluyendo memoria de acceso aleatorio dinámica (DRAM) o memoria de acceso aleatorio de gráficos, tal como memoria de acceso aleatorio de gráficos síncrona (SGRAM), incluyendo memoria de tasa de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 224A-224N pueden incluir también memoria apilada 3D, incluyendo, pero sin limitación, memoria de ancho de banda alto (HBM). Los expertos en la materia apreciarán que la implementación específica de las unidades de memoria 224A-224N puede variar, y se puede seleccionar de uno de diversos diseños convencionales. Se pueden almacenar objetivos de representación, tales como memorias intermedias de tramas o correlaciones de textura a lo largo de las unidades de memoria 224A-224N, permitiendo que las unidades de subdivisión 220A-220N escriban porciones de cada objetivo de representación en paralelo para usar de manera eficiente el ancho de banda disponible de la memoria de procesador paralelo 222. En algunas realizaciones, se puede excluir una instancia local de la memoria de procesador paralelo 222 en favor de un diseño de memoria unificado que utiliza memoria de sistema junto con memoria caché local.

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

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

La **Figura 2B** es un diagrama de bloques de una unidad de subdivisión 220, de acuerdo con una realización. En una realización, la unidad de subdivisión 220 es una instancia de una de las unidades de subdivisión 220A-220N de la Figura 2A. Como se ilustra, la unidad de subdivisión 220 incluye una caché de L2 221, una interfaz de memoria intermedia de tramas 225 y una ROP 226 (unidad de operaciones de rasterización). La caché de L2 221 es una caché de lectura/escritura que está configurada para realizar operaciones de carga y de almacenamiento recibidas desde la barra transversal de memoria 216 y la ROP 226. Los desaciertos de lectura y las solicitudes de escritura diferida urgente son emitidas por la caché de L2 221 a la interfaz de memoria intermedia de tramas 225 para su procesamiento. También se pueden enviar actualizaciones a la memoria intermedia de tramas a través de la interfaz de memoria intermedia de tramas 225 para su procesamiento. En una realización, la interfaz de memoria intermedia de tramas 225 interacciona con una de las unidades de memoria en memoria de procesador paralelo, tales como las unidades de memoria 224A-224N de la Figura 2 (por ejemplo, dentro de la memoria de procesador paralelo 222).

En aplicaciones de gráficos, la ROP 226 es una unidad de procesamiento que realiza operaciones de rasterización tales como estarcido, prueba z, mezcla y similares. La ROP 226 emite entonces datos de gráficos procesados que se almacenan en memoria de gráficos. En algunas realizaciones, la ROP 226 incluye lógica de compresión para comprimir datos de profundidad o de color que se escriben en memoria y descomprimir datos de profundidad o de color que se leen desde memoria. La lógica de compresión puede ser lógica de compresión sin pérdidas que hace uso de uno o más de múltiples algoritmos de compresión. El tipo de compresión que es realizado por la ROP 226 puede variar basándose en las características estadísticas de los datos a comprimir. Por ejemplo, en una realización, se realiza una compresión de color delta sobre datos de profundidad y de color de una manera por tesela.

En algunas realizaciones, la ROP 226 se incluye dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 214A-214N de la Figura 2) en lugar de dentro de la unidad de subdivisión 220. En tal realización, se transmiten solicitudes de lectura y de escritura de datos de píxel a través de la barra transversal de memoria 216 en lugar de datos de fragmento de píxel. Los datos de gráficos procesados se pueden visualizar en un dispositivo de visualización, tal como uno de los uno o más dispositivos de visualización 110 de la Figura 1, encaminarse para su procesamiento adicional por el procesador o procesadores 102, o encaminarse para su procesamiento adicional por

una de las entidades de procesamiento dentro del procesador paralelo 200 de la Figura 2A.

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

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

Cada multiprocesador de gráficos 234 dentro de la agrupación de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades aritméticas lógicas, unidades de carga-almacenamiento, etc.). La lógica de ejecución funcional puede configurase de una manera en canalización en la que se pueden emitir instrucciones nuevas antes de que se estén completadas instrucciones previas. La lógica de ejecución funcional soporta una diversidad de operaciones, incluyendo aritmética de números enteros y de coma flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y cómputo de diversas funciones algebraicas. En una realización, se puede aprovechar el mismo hardware de unidades funcionales para realizar diferentes operaciones, y puede estar presente cualquier combinación de unidades funcionales.

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

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

Cada agrupación de procesamiento 214 puede incluir una MMU 245 (unidad de gestión de memoria) que está configurada para correlacionar direcciones virtuales en direcciones físicas. En otras realizaciones, una o más instancias de la MMU 245 pueden residir dentro de la interfaz de memoria 218 de la Figura 2. La MMU 245 incluye un conjunto de entradas de tabla de página (PTE) usadas para correlacionar una dirección virtual con una dirección física

de una tesela y, opcionalmente, un índice de líneas de caché. La MMU 245 puede incluir memorias intermedias de traducción adelantada (TLB) de direcciones o cachés que pueden residir dentro del multiprocesador de gráficos 234 o la caché de L1 o la agrupación de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso de datos de superficie para permitir una intercalación de solicitud eficiente entre unidades de subdivisión. El índice de líneas de caché se puede usar para determinar si una solicitud de una línea de caché es un acierto o un desacierto.

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

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

La Figura 2D muestra un multiprocesador de gráficos 234, de acuerdo con una realización. En tal realización, el multiprocesador de gráficos 234 se acopla con el gestor de canalizaciones 232 de la agrupación de procesamiento 214. El multiprocesador de gráficos 234 tiene una canalización de ejecución que incluye, pero sin limitación, una caché de instrucciones 252, una unidad de instrucción 254, una unidad de correlación de direcciones 256, un archivo de registro 258, uno o más núcleos de unidad de procesamiento de gráficos de propósito general (GPGPU) 262 y una o más unidades de carga/almacenamiento 266. Los núcleos de GPGPU 262 y las unidades de carga/almacenamiento 266 están acoplados con la memoria caché 272 y la memoria compartida 270 a través de una interconexión de memoria y de caché 268.

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

El archivo de registro 258 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de gráficos 324. El archivo de registro 258 proporciona almacenamiento temporal para operandos conectados a las rutas de datos de las unidades funcionales (por ejemplo, los núcleos de GPGPU 262, las unidades de carga/almacenamiento 266) del multiprocesador de gráficos 324. En una realización, el archivo de registro 258 se divide entre cada una de las unidades funcionales de manera que cada unidad funcional está asignada a una porción dedicada del archivo de registro 258. En una realización, el archivo de registro 258 se divide entre las diferentes urdimbres que son ejecutadas por el multiprocesador de gráficos 324.

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

realización, uno o más de los núcleos de GPGPU pueden incluir también lógica de función fija o especial.

En una realización, los núcleos de GPGPU 262 incluyen lógica de SIMD capaz de realizar una única instrucción sobre múltiples conjuntos de datos. En una realización, los núcleos de GPGPU 262 pueden ejecutar físicamente instrucciones de SIMD4, de SIMD8 y de SIMD16 y ejecutar lógicamente instrucciones de SIMD1, de SIMD2 y de SIMD32. Las instrucciones de SIMD para los núcleos de GPGPU pueden ser generadas en tiempo de compilación por un compilador sombreador o se pueden generar automáticamente cuando se ejecutan programas escritos y compilados para arquitecturas de único programa-múltiples datos (SPMD) o de SIMT. Múltiples hilos de un programa configurado para el modelo de ejecución de SIMT se pueden ejecutar a través de una única instrucción de SIMD. Por ejemplo, y en una realización, ocho hilos de SIMT que realizan las mismas operaciones, o unas similares, se pueden ejecutar en paralelo a través de una única unidad de lógica de SIMD8.

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

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

La Figura 3A muestra un multiprocesador de gráficos 325 de acuerdo con una realización adicional. El multiprocesador de gráficos 325 incluye múltiples instancias adicionales de unidades de recurso de ejecución relativas al multiprocesador de gráficos 234 de la Figura 2D. Por ejemplo, el multiprocesador de gráficos 325 puede incluir múltiples instancias de la unidad de instrucción 332A-332B, el archivo de registro 334A-334B y la unidad o unidades de textura 344A-344B. El multiprocesador de gráficos 325 también incluye múltiples conjuntos de unidades de ejecución de cómputo o de gráficos (por ejemplo, el núcleo de GPGPU 336A-336B, el núcleo de GPGPU 337A-337B, el núcleo de GPGPU 338A-338B) y múltiples conjuntos de unidades de carga/almacenamiento 340A-340B. En una realización, las unidades de recurso de ejecución tienen una caché de instrucciones común 330, una memoria caché de textura y/o de datos 342 y una memoria compartida 346.

Los diversos componentes se pueden comunicar a través de un tejido de interconexión 327. En una realización, el tejido de interconexión 327 incluye uno o más conmutadores de barra transversal para habilitar la comunicación entre los diversos componentes del multiprocesador de gráficos 325. En una realización, el tejido de interconexión 327 es una capa de tejido de red de alta velocidad separada sobre la que se apila cada componente del multiprocesador de gráficos 325. Los componentes del multiprocesador de gráficos 325 se comunican con componentes remotos a través del tejido de interconexión 327. Por ejemplo, cada uno de los núcleos de GPGPU 336A-336B, 337A-337By 3378A-338B se puede comunicar con la memoria compartida 346 a través del tejido de interconexión 327. El tejido de interconexión 327 puede arbitrar la comunicación dentro del multiprocesador de gráficos 325 para garantizar una asignación de ancho de banda equitativa entre componentes.

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

Los expertos en la materia entenderán que la arquitectura descrita en las Figuras 1, 2A-2D y 3A-3B es descriptiva y no limitante en cuanto al alcance de las presentes realizaciones. Por lo tanto, las técnicas descritas en el presente documento se pueden implementar en cualquier unidad de procesamiento configurada apropiadamente, incluyendo, sin limitación, uno o más procesadores de aplicación móvil, una o más unidades centrales de procesamiento (CPU) de sobremesa o de servidor, incluyendo CPU de múltiples núcleos, una o más unidades de procesamiento paralelo,

tales como la unidad de procesamiento paralelo 202 de la Figura 2, así como uno o más procesadores de gráficos o unidades de procesamiento de propósito especial, sin apartarse del alcance de las realizaciones descritas en el presente documento.

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

Técnicas para interconexión de GPU a procesador de anfitrión

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

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

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

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

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

El procesador 407 ilustrado incluye una pluralidad de núcleos 460A-460D, cada uno con una memoria intermedia de traducción adelantada 461A-461D y una o más cachés 462A-462D. Los núcleos pueden incluir diversos otros componentes para ejecutar instrucciones y procesar datos que no se ilustran para evitar complicar los principios subyacentes de la invención (por ejemplo, unidades de extracción de instrucción, unidades de predicción de bifurcaciones, descodificadores, unidades de ejecución, memorias intermedias de reordenación, etc.). Las cachés 462A-462D pueden comprender cachés de nivel 1 (L1) y de nivel 2 (L2). Además, una o más cachés compartidas 426

se pueden incluir en la jerarquía de almacenamiento en caché y pueden ser compartidas por conjuntos de los núcleos 460A-460D. Por ejemplo, una realización del procesador 407 incluye 24 núcleos, cada uno con su propia caché de L1, doce cachés de L2 compartidas y doce cachés de L3 compartidas. En esta realización, una de las cachés de L2 y de L3 es compartida por dos núcleos adyacentes. El procesador 407 y el módulo de integración de acelerador de gráficos 446 se conectan con la memoria de sistema 441, que puede incluir las memorias de procesador 401-402

Se mantiene la coherencia para datos e instrucciones almacenados en las diversas cachés 462A-462D, 456 y la memoria de sistema 441 a través de comunicación entre núcleos a través de un bus de coherencia 464. Por ejemplo, cada caché puede tener una lógica/circuitería de coherencia de caché asociada con la misma con la que comunicarse a través del bus de coherencia 464 en respuesta a lecturas o escrituras detectadas en líneas de caché particulares. En una implementación, se implementa un protocolo de fisgoneo de caché a través del bus de coherencia 464 para fisgar accesos de caché. Las técnicas de fisgoneo/coherencia de caché son bien entendidas por los expertos en la materia y no se describirán con detalle en el presente caso para evitar complicar los principios subyacentes de la invención.

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

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

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

Un conjunto de registros 449 almacenan datos de contexto para hilos ejecutados por los motores de procesamiento de gráficos 431-432, N y un circuito de gestión de contexto 448 gestiona los contextos de hilo. Por ejemplo, el circuito de gestión de contexto 448 puede realizar operaciones de guardado y de restablecimiento para guardar y restablecer contextos de los diversos hilos durante conmutaciones de contexto (por ejemplo, en donde se guarda un primer hilo y se almacena un segundo hilo de modo que el segundo hilo puede ser ejecutado por un motor de procesamiento de gráficos). Por ejemplo, en una conmutación de contexto, el circuito de gestión de contexto 448 puede almacenar valores de registro actuales en una región designada en memoria (por ejemplo, identificada por un puntero de contexto). Este puede restablecer entonces los valores de registro cuando se vuelve al contexto. En una realización, un circuito de gestión de interrupciones 447 recibe y procesa interrupciones recibidas desde dispositivos de sistema.

En una implementación, direcciones virtuales/eficaces desde un motor de procesamiento de gráficos 431 son traducidas, por la MMU 439, a direcciones reales/físicas en la memoria de sistema 411. Una realización del circuito de integración de acelerador 436 soporta múltiples (por ejemplo, 4, 8, 16) módulos de acelerador de gráficos 446 y/u otros dispositivos aceleradores. El módulo acelerador de gráficos 446 se puede dedicar a una única aplicación ejecutada en el procesador 407 o se puede compartir entre múltiples aplicaciones. En una realización, se presenta un entorno de ejecución de gráficos virtualizado en el que los recursos de los motores de procesamiento de gráficos 431-432, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos se pueden subdividir en "cortes" que se asignan a diferentes VM y/o aplicaciones basándose en los requisitos de procesamiento y prioridades asociados con las VM y/o las aplicaciones.

Por lo tanto, el circuito de integración de acelerador actúa como un puente al sistema para el módulo de aceleración

de gráficos 446 y proporciona servicios de traducción de direcciones y de caché de memoria de sistema. Además, el circuito de integración de acelerador 436 puede proporcionar instalaciones de virtualización para que el procesador de anfitrión gestione la virtualización de los motores de procesamiento de gráficos, las interrupciones y la gestión de memoria.

Debido a que los recursos de hardware de los motores de procesamiento de gráficos 431-432, N se correlacionan explícitamente con el espacio de direcciones real observado por el procesador de anfitrión 407, cualquier procesador de anfitrión puede dirigir estos recursos directamente usando un valor de dirección eficaz. Una función del circuito de integración de acelerador 436, en una realización, es la separación física de los motores de procesamiento de gráficos 431-432, N de modo que aparecen ante el sistema como unidades independientes.

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

En una realización, para reducir el tráfico de datos a través del enlace de alta velocidad 440, se usan técnicas de desviación para garantizar que los datos almacenados en las memorias de gráficos 433-434, M son datos que serán usados de la manera más frecuente por los motores de procesamiento de gráficos 431-432, N y preferiblemente no serán usados por los núcleos 460A-460D (al menos, no de manera frecuente). De manera similar, el mecanismo de desviación intenta mantener datos que son necesitados por los núcleos (y, preferiblemente, no por los motores de procesamiento de gráficos 431-432, N) dentro de las cachés 462A-462D, 456 de los núcleos y la memoria de sistema 411.

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

Una realización soporta 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 compartida (con virtualización). Este último puede incluir modelos de programación que son controlados por el circuito de integración de acelerador 436 y modelos de programación que son controlados por el módulo de aceleración de gráficos 446.

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

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

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

La **Figura 4D** ilustra un corte de integración de acelerador 490 ilustrativo. Como se usa en el presente documento, un "corte" comprende una porción especificada de los recursos de procesamiento del circuito de integración de acelerador 436. El espacio de direcciones eficaces de aplicación 482 dentro de la memoria de sistema 411 almacena los elementos de proceso 483. En una realización, los elementos de proceso 483 se almacenan en respuesta a las invocaciones de GPU 481 desde las aplicaciones 480 ejecutadas en el procesador 407. Un elemento de proceso 483 contiene el estado de proceso para la aplicación 480 correspondiente. Un descriptor de trabajo (WD) 484 contenido

en el elemento de proceso 483 puede ser un único trabajo solicitado por una aplicación o puede contener un puntero a una cola de trabajos. En este último caso, el WD 484 es un puntero a la cola de solicitudes de trabajo en el espacio de direcciones 482 de la aplicación.

El módulo de aceleración de gráficos 446 y/o los motores de procesamiento de gráficos 431-432, N individuales pueden ser compartidos por todos, o por un subconjunto de, los procesos en el sistema. Las realizaciones de la invención incluyen una infraestructura para establecer el estado de proceso y enviar un WD 484 a un módulo de aceleración de gráficos 446 para empezar un trabajo en un entorno virtualizado.

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

Durante el funcionamiento, una unidad de extracción de WD 491 en el corte de integración de acelerador 490 extrae el WD 484 siguiente que incluye una indicación del trabajo a hacer por uno de los motores de procesamiento de gráficos del módulo de aceleración de gráficos 446. Datos desde el WD 484 se pueden almacenar en los registros 449 y ser usados por la MMU 439, el circuito de gestión de interrupciones 447 y/o el circuito de gestión de contexto 446 como se ilustra. Por ejemplo, una realización de la MMU 439 incluye circuitería de recorrido de segmentos/páginas para acceder a las tablas de segmentos/páginas 486 dentro del espacio de direcciones virtuales de SO 485. El circuito de gestión de interrupciones 447 puede procesar los eventos de interrupción 492 recibidos del módulo de aceleración de gráficos 446. Cuando se realizan operaciones de gráficos, una dirección eficaz 493 generada por un motor de procesamiento de gráficos 431-432, N es traducida a una dirección real por la MMU 439.

En una realización, el mismo conjunto de registros 449 se duplica para cada motor de procesamiento de gráficos 431-432, N y/o módulo de aceleración de gráficos 446, y puede ser inicializado por el hipervisor o el sistema operativo. Cada uno de estos registros duplicados se puede incluir en un corte de integración de acelerador 490. En la **Tabla 1** se muestran registros ilustrativos que pueden ser inicializados por el hipervisor.

Tabla 1 - Registros inicializados por hipervisor

1	Registro de control de corte
2	Puntero de área de procesos planificados de dirección real (RA)
3	Registro de anulación de máscara de autoridad
4	Desplazamiento de entrada de tabla de vectores de interrupción
5	Límite de entrada de tabla de vectores de interrupción
6	Registro de estado
7	ID de subdivisión 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 ilustrativos que pueden ser inicializados por el sistema operativo.

Tabla 2 - Registros inicializados por sistema operativo

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

En una realización, cada WD 484 es específico de un módulo de aceleración de gráficos 446 y/o de un motor de procesamiento de gráficos 431-432, N particular. Este contiene toda la información que requiere un motor de procesamiento de gráficos 431-432, N para hacer su trabajo, o puede ser un puntero a una ubicación de memoria en la que la aplicación ha establecido una cola de comandos de trabajo a completar.

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

Los modelos de programación compartida prevén que todos los procesos, o un subconjunto de los mismos, de todas las subdivisiones en el sistema, o de un subconjunto de las mismas, usen un módulo de aceleración de gráficos 446. Hay dos modelos de programación en los que el módulo de aceleración de gráficos 446 es compartido por múltiples procesos y particiones: compartido en cortes de tiempo y compartido dirigido a gráficos.

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

En una realización, para el modelo compartido, se requiere que la aplicación 480 haga una llamada de sistema al sistema operativo 495 con un tipo del módulo de aceleración de gráficos 446, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de guardado/restablecimiento de contexto (CSRP). El tipo del módulo de aceleración de gráficos 446 describe la función de aceleración seleccionada como objetivo para la llamada de sistema. El tipo del módulo de aceleración de gráficos 446 puede ser un valor específico del sistema. Al WD se le da formato específicamente para el módulo de aceleración de gráficos 446, y puede estar en forma de un comando del módulo de aceleración de gráficos 446, un puntero de dirección eficaz a una estructura definida por el usuario, un puntero de dirección eficaz a una cola de comandos, o cualquier otra estructura de datos para describir el trabajo a hacer por el módulo de aceleración de gráficos 446. En una realización, el valor de AMR es el estado de AMR a usar para el proceso actual. El valor pasado al sistema operativo es similar a que una aplicación establezca el AMR. Si las implementaciones del circuito de integración de acelerador 436 y del módulo de aceleración de gráficos 446 no soportan un Registro de Anulación de Máscara de Autoridad de Usuario (UAMOR), el sistema operativo puede aplicar el valor de UAMOR actual al valor de AMR antes de pasar el AMR en la llamada de hipervisor. El hipervisor 496 puede aplicar opcionalmente el valor de registro de anulación de máscara de autoridad (AMOR) actual antes de colocar el AMR en el elemento de proceso 483. En una realización, el CSRP es uno de los registros 449 que contienen la dirección eficaz de un área en el espacio de direcciones 482 de la aplicación para que el módulo de aceleración de gráficos 446 guarde y restablezca el estado de contexto. Este puntero es opcional si no se requiere que se guarde estado alguno entre trabajos o cuando se da prioridad a un trabajo. El área de guardado/restablecimiento de contexto puede ser una memoria de sistema anclada.

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

Tabla 3 - Parámetros de llamada de SO a hipervisor

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

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

Tabla 4 - Información de elemento de proceso

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de guardado/restablecimiento de contexto (CSRP) de dirección eficaz (EA)
4	Un ID de proceso (PID) e ID de hilo (TID) opcional
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de segmentos de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)
8	Tabla de vectores de interrupción, derivada de los parámetros de llamada de hipervisor.
9	Un valor de registro de estado (SR)
10	Un ID de subdivisión lógica (LPID)
11	Un puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
12	El registro de descriptor de almacenamiento (SDR)

En una realización, el hipervisor inicializa una pluralidad de registros 490 del corte de integración de acelerador 449.

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

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

Una realización permite que la memoria anexada a GPU 420-423 se correlacione como parte de memoria de sistema, y que se acceda a la misma usando tecnología de memoria virtual compartida (SVM), pero sin adolecer de las desventajas de desempeño habituales asociadas con la coherencia de caché de sistema completa. La capacidad de que se acceda a la memoria anexada a GPU 420-423 como memoria de sistema sin una tara de coherencia de caché onerosa proporciona un entorno de operación beneficioso para la descarga de GPU. Esta disposición permite que el software del procesador de anfitrión 405 establezca operandos y acceda a resultados de cómputo, sin la tara de copias de datos de acceso de memoria directo (DMA) de E/S tradicionales. Tales copias tradicionales implican llamadas de controlador, interrupciones y accesos de E/S correlacionados con memoria (MMIO) que son, todos ellos, ineficientes en relación con accesos de memoria sencillos. Al mismo tiempo, la capacidad de acceder a la memoria anexada a GPU 420-423 sin taras de coherencia de caché puede ser crítica para el tiempo de ejecución de un cómputo descargado. En casos con tráfico de memoria de escritura de transmisión por flujo continuo sustancial, por ejemplo, la tara de coherencia de caché puede reducir significativamente el ancho de banda de escritura eficaz observado por una GPU 410-413. La eficiencia del establecimiento de operandos, la eficiencia del acceso a resultados y la eficiencia del cómputo de GPU desempeñan, todas ellas, un papel en la determinación de la eficacia de la descarga de GPU.

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

En una implementación, se accede a la entrada de tabla de desvíos asociada con cada acceso a la memoria anexada a GPU 420-423 antes del acceso real a la memoria de GPU, provocando las operaciones siguientes. En primer lugar, solicitudes locales desde la GPU 410-413 que encuentran su página en el desvío de GPU se reenvían directamente a

una memoria de GPU 420-423 correspondiente. Las solicitudes locales de la GPU que encuentran su página en la desviación del anfitrión se reenvían al procesador 405 (por ejemplo, a través de un enlace de alta velocidad como se ha analizado anteriormente). En una realización, las solicitudes del procesador 405 que encuentran la página solicitada en una desviación de procesador de anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, solicitudes dirigidas a una página con desvío de GPU se pueden redirigir a la GPU 410-413. La GPU puede hacer entonces que la página realice una transición a una desviación de procesador de anfitrión si no está usando actualmente la página.

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

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

En una realización, la coherencia de caché se mantiene haciendo temporalmente que las páginas con desvío de GPU no puedan ser almacenadas en caché por el procesador de anfitrión 405. Para acceder a estas páginas, el procesador 405 puede solicitar acceso desde la GPU 410 que puede conceder, o no, acceso de manera inmediata, dependiendo de la implementación. Por lo tanto, para reducir la comunicación entre el procesador 405 y la GPU 410, es beneficioso garantizar que las páginas con desvío de GPU son aquellas que son requeridas por la GPU, pero no por el procesador de anfitrión 405, y viceversa.

Canalización de procesamiento de gráficos

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

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

Una primera instancia de un ensamblador de primitivas 506 recibe atributos de vértice desde la unidad de procesamiento de vértices 50. El ensamblador de primitivas 506 lee atributos de vértice almacenados según sea necesario y construye primitivas de gráficos para su procesamiento por la unidad de procesamiento de control de teselación 508. Las primitivas de gráficos incluyen triángulos, segmentos de línea, puntos, parches y así sucesivamente, según sea soportado por diversas interfaces de programación de aplicaciones (API) de procesamiento de gráficos.

La unidad de procesamiento de control de teselación 508 trata los vértices de entrada como puntos de control para un parche geométrico. Los puntos de control se transforman de una representación de entrada a partir del parche (por ejemplo, las bases del parche) a una representación que es adecuada para su uso en una evaluación superficial por la unidad de procesamiento de evaluación de teselación 512. La unidad de procesamiento de control de teselación 508 también puede computar factores de teselación para bordes de parches geométricos. Un factor de teselación es de aplicación a un único borde y cuantifica un nivel de detalle, dependiente de la vista, asociado con el borde. Una unidad de teselación 510 está configurada para recibir los factores de teselación para bordes de un parche y para teselar el parche en múltiples primitivas geométricas, tales como primitivas de línea, de triángulo o cuadrilaterales, que se transmiten a una unidad de procesamiento de evaluación de teselación 512. La unidad de procesamiento de evaluación de teselación 512 opera sobre coordenadas parametrizadas del parche subdividido para generar una representación superficial y atributos de vértice para cada vértice asociado con las primitivas geométricas.

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

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

El rasterizador 522 puede realizar optimizaciones de selección de profundidad y otras basadas en profundidad. El rasterizador 522 también realiza una conversión de exploración sobre las nuevas primitivas de gráficos para generar fragmentos y emitir esos fragmentos y datos de cobertura asociados a la unidad de procesamiento de fragmentos/píxeles 524. La unidad de procesamiento de fragmentos/píxeles 524 es una unidad de ejecución programable que está configurada para ejecutar programas de sombreado de fragmentos o programas de sombreado de píxeles. Transformando, la unidad de procesamiento de fragmentos/píxeles 524, fragmentos o píxeles recibidos desde el rasterizador 522, según sea especificado por los programas de sombreado de fragmentos o de píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 se puede programar para realizar operaciones que incluyen, pero sin limitación, correlación de textura, sombreado, mezcla, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se emiten a una unidad de operaciones de rasterización 526. La unidad de procesamiento de fragmentos/píxeles 524 puede leer datos que se almacenan o bien en la memoria de procesador paralelo o bien en la memoria de sistema para su uso cuando se procesan los datos de fragmento. Se pueden configurar programas de sombreado de fragmentos o de píxeles para sombrear con granularidades de muestra, de píxel, de tesela u otras dependiendo de las tasas de muestreo configuradas para las unidades de procesamiento.

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

Vista global de aprendizaje automático

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

Un tipo ilustrativo de algoritmo de aprendizaje automático es una red neuronal. Hay muchos tipos de redes neuronales; un tipo sencillo de red neuronal es una red de realimentación prospectiva. Una red de realimentación prospectiva se

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

Antes de que se pueda usar un algoritmo de aprendizaje automático para modelar un problema particular, se entrena el algoritmo usando un conjunto de datos de entrenamiento. Entrenar una red neuronal implica seleccionar una topología de red, usar un conjunto de datos de entrenamiento que representa un problema que es modelado por la red, y ajustar los pesos hasta que el modelo de red rinde con un error mínimo para todas las instancias del conjunto de datos de entrenamiento. Por ejemplo, durante un proceso de entrenamiento de aprendizaje supervisado para una red neuronal, la salida producida por la red en respuesta a la entrada que representa una instancia en un conjunto de datos de entrenamiento se compara con la salida etiquetada "correcta" para esa instancia, se calcula una señal de error que representa la diferencia entre la salida y la salida etiquetada, y se ajustan los pesos asociados con las conexiones para minimizar ese error a medida que la señal de error se retropropaga 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 se puede ver afectada significativamente por la calidad del conjunto de datos usado para entrenar el algoritmo. El proceso de entrenamiento puede ser intensivo desde el punto de vista computacional y puede requerir una cantidad de tiempo significativa en un procesador de propósito general convencional. En consecuencia, se usa hardware de procesamiento paralelo para entrenar muchos tipos de algoritmos de aprendizaje automático. Esto es particularmente útil para optimizar el entrenamiento de redes neuronales, debido a que los cómputos realizados en el ajuste de los coeficientes en redes neuronales se prestan de manera natural a 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 de propósito general.

La **Figura 6** es un diagrama generalizado de una pila de software de aprendizaje automático 600. Una aplicación de aprendizaje automático 602 se puede configurar para entrenar una red neuronal usando un conjunto de datos de entrenamiento o para usar una red neuronal profunda entrenada para implementar una inteligencia automática. La aplicación de aprendizaje automático 602 puede incluir una funcionalidad de entrenamiento y de inferencia para una red neuronal y/o software especializado que se puede usar para entrenar una red neuronal antes del despliegue. La aplicación de aprendizaje automático 602 puede implementar cualquier tipo de inteligencia automática incluyendo, pero sin limitación, reconocimiento de imágenes, correlación y localización, navegación autónoma, síntesis de habla, formación de imágenes médicas o traducción de idioma.

Se puede habilitar una aceleración de hardware para la aplicación de aprendizaje automático 602 a través de una estructura de aprendizaje automático 604. La estructura de aprendizaje automático 604 puede proporcionar una biblioteca de primitivas de aprendizaje automático. Las primitivas de aprendizaje automático son operaciones básicas que se realizan comúnmente por algoritmos de aprendizaje automático. Sin la estructura de aprendizaje automático 604, se requeriría que los desarrolladores de algoritmos de aprendizaje automático crearan y optimizaran la lógica computacional principal asociada con el algoritmo de aprendizaje automático, y que reoptimizaran entonces la lógica computacional a medida que se desarrollan nuevos procesadores paralelos. En su lugar, la aplicación de aprendizaje automático se puede configurar para realizar los cómputos necesarios usando las primitivas proporcionadas por la estructura de aprendizaje automático 604. Las primitivas ilustrativas incluyen convoluciones tensoriales, funciones de activación y agrupación, que son operaciones computacionales que se realizan mientras se entrena una red neuronal convolucional (CNN). La estructura de aprendizaje automático 604 también puede proporcionar primitivas para implementar subprogramas de álgebra lineal básicos realizados por muchos algoritmos de aprendizaje automático, tales como operaciones matriciales y vectoriales.

La estructura de aprendizaje automático 604 puede procesar datos de entrada recibidos desde la aplicación de aprendizaje automático 602 y generar la entrada apropiada a una estructura de cómputo 606. La estructura de cómputo 606 puede abstraer las instrucciones subyacentes proporcionadas al controlador de GPGPU 608 para habilitar que la estructura de aprendizaje automático 604 se aproveche de la aceleración de hardware a través del hardware de GPGPU 610 sin requerir que la estructura de aprendizaje automático 604 tenga un conocimiento íntimo de la arquitectura del hardware de GPGPU 610. Adicionalmente, la estructura de cómputo 606 puede habilitar la aceleración de hardware para la estructura de aprendizaje automático 604 a lo largo de una diversidad de tipos y generaciones del hardware de GPGPU 610.

Aceleración de aprendizaje automático de GPGPU

La **Figura 7** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela 700, de acuerdo con una realización. En una realización la unidad de procesamiento de propósito general (GPGPU) 700 se puede configurar para ser particularmente eficiente en el procesamiento del tipo de cargas de trabajo computacionales asociadas con el entrenamiento de redes neuronales profundas. Adicionalmente, la GPGPU 700 se puede vincular directamente a otras instancias de la GPGPU para crear una agrupación de múltiples GPU para mejorar la velocidad de entrenamiento para redes neuronales particularmente profundas.

La GPGPU 700 incluye una interfaz de anfitrión 702 para habilitar una conexión con un procesador de anfitrión. En una realización, la interfaz de anfitrión 702 es una interfaz PCI Express. Sin embargo, la interfaz de anfitrión puede ser también una interfaz de comunicaciones o tejido de comunicaciones específico de proveedor. La GPGPU 700 recibe comandos desde el procesador de anfitrión y usa un planificador global 704 para distribuir hilos de ejecución asociados con esos comandos a un conjunto de agrupaciones de cómputo 706A-706H. Las agrupaciones de cómputo 706A-706H comparten una memoria caché 708. La memoria caché 708 puede servir como una caché de nivel superior para memorias caché dentro de las agrupaciones de cómputo 706A-706H.

La GPGPU 700 incluye la memoria 714A-714B acoplada con las agrupaciones de cómputo 706A-H a través de un conjunto de controladores de memoria 712A-712B. En diversas realizaciones, la memoria 714A-714B puede incluir diversos tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámica (DRAM) o memoria de acceso aleatorio de gráficos, tal como memoria de acceso aleatorio de gráficos síncrona (SGRAM), incluyendo memoria de tasa de datos doble de gráficos (GDDR), o memoria apilada 3D, incluyendo, pero sin limitación, memoria de ancho de banda alto (HBM).

En una realización, cada agrupación de cómputo 706A-706H incluye un conjunto de multiprocesadores de gráficos, tales como el multiprocesador de gráficos 400 de la Figura 4A. Los multiprocesadores de gráficos de la agrupación de cómputo múltiples tipos de unidades de lógica de números enteros y de coma flotante que pueden realizar operaciones computacionales con un rango de precisiones que incluyen unas adecuadas para cómputos de aprendizaje automático. Por ejemplo, y en una realización, al menos un subconjunto de las unidades de coma flotante en cada una de las agrupaciones de cómputo 706A-706H se puede configurar para realizar operaciones de coma flotante de 16 bits o de 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante se puede configurar para realizar operaciones de coma flotante de 64 bits.

Múltiples instancias de la GPGPU 700 se pueden configurar para funcionar como una agrupación de cómputo. El mecanismo de comunicación usado por la agrupación de cómputo para la sincronización y el intercambio de datos varía a lo largo de las realizaciones. En una realización, las múltiples instancias de la GPGPU 700 se comunican a través de la interfaz de anfitrión 702. En una realización, la GPGPU 700 incluye un concentrador de E/S 709 que acopla la GPGPU 700 con un enlace de GPU 710 que habilita una conexión directa a otras instancias de la GPGPU. En una realización, el enlace de GPU 710 está acoplado a un puente de GPU a GPU dedicado que habilita la comunicación y la sincronización entre múltiples instancias de la GPGPU 700. En una realización, el enlace de GPU 710 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En una realización, 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 de anfitrión 702. En una realización, el enlace de GPU 710 se puede configurar para habilitar una conexión a un procesador de anfitrión además de o como una alternativa a la interfaz de anfitrión 702.

Aunque la configuración ilustrada de la GPGPU 700 se puede configurar para entrenar redes neuronales, una realización proporciona una configuración alternativa de la GPGPU 700 que se puede configurar para el despliegue dentro de una plataforma de inferenciación de alto desempeño o de baja potencia. En una configuración de inferenciación, la GPGPU 700 incluye menos de las agrupaciones de cómputo de las agrupaciones de cómputo 706A-706H en relación con la configuración de entrenamiento. Adicionalmente, una tecnología de memoria asociada con la memoria 714A-714B puede diferir entre configuraciones de inferenciación y de entrenamiento. En una realización, la configuración de inferenciación de la GPGPU 700 puede soportar instrucciones específicas de inferenciación. Por ejemplo, una configuración de inferenciación puede proporcionar soporte para una o más instrucciones de producto escalar de números enteros de 8 bits, que se usan comúnmente durante operaciones de inferenciación para redes neuronales desplegadas.

La **Figura 8** ilustra un sistema informático de múltiples GPU 800, de acuerdo con una realización. El sistema informático de múltiples GPU 800 puede incluir un procesador 802 acoplado a múltiples GPGPU 806A-806D a través de un conmutador de interfaz de anfitrión 804. El conmutador de interfaz de anfitrión 804, en una realización, es un dispositivo de conmutador de PCI express que acopla el procesador 802 a un bus de PCI express a través del que el procesador 802 se puede comunicar con el conjunto de GPGPU 806A-806D. Cada una de las múltiples GPGPU 806A-806D puede ser una instancia de la GPGPU 700 de la Figura 7. Las GPGPU 806A-806D se pueden interconectar a través de un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 816. Los enlaces de GPU a GPU de alta velocidad se pueden conectar a cada una de las GPGPU 806A-806D a través de un enlace de GPU dedicado, tal como el enlace de GPU 710 como en la Figura 7. Los enlaces de GPU de P2P 816 habilitan una comunicación

directa entre cada una de las GPGPU 806A-806D sin requerir una comunicación a través del bus de interfaz de anfitrión 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 de anfitrión permanece disponible para el acceso de memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 800, por ejemplo, a través de uno o más dispositivos de red. Aunque, en la realización ilustrada, las GPGPU 806A-806D se conectan al procesador 802 a través del conmutador de interfaz de anfitrión 804, en una realización, el procesador 802 incluye un soporte directo para los enlaces de GPU de P2P 816 y se puede conectar directamente a las GPGPU 806A-806D.

Implementaciones de red neuronal de aprendizaje automático

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

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

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

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

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

Las redes neuronales profundas usadas en el aprendizaje automático incluyen habitualmente una red de extremo frontal para realizar un reconocimiento de características, acoplada a una red de extremo trasero que representa un modelo matemático que puede realizar operaciones (por ejemplo, clasificación de objetos, reconocimiento de habla, etc.) basándose en la representación de características proporcionada al modelo. Un aprendizaje profundo habilita que se realice un aprendizaje automático sin requerir que se realice una ingeniería de características artesanal para el modelo. En su lugar, las redes neuronales profundas pueden aprender características basándose en una correlación o estructura estadística dentro de los datos de entrada. Las características aprendidas se pueden proporcionar a un modelo matemático que puede correlacionar características detectadas con una salida. El modelo matemático usado por la red está especializado, en general, para la tarea específica a realizar, y se usarán diferentes modelos para realizar diferentes tareas.

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

Las **Figuras 9A-9B** ilustran una red neuronal convolucional ilustrativa. La Figura 9A ilustra diversas capas dentro de una CNN. Como se muestra en la Figura 9A, una CNN ilustrativa usada para modelar el procesamiento de imagen puede recibir la entrada 902 que describe los componentes de rojo, verde y azul (RGB) de una imagen de entrada. La entrada 902 puede ser procesada por múltiples capas convolucionales (por ejemplo, la capa convolucional 904, la capa convolucional 906). La salida desde las múltiples capas convolucionales puede ser procesada opcionalmente por un conjunto de capas completamente conectadas 908. Las neuronas en una capa completamente conectada tienen conexiones completas a todas las activaciones en la capa previa, como se ha descrito previamente para una red de realimentación prospectiva. La salida desde las capas completamente conectadas 908 se puede usar para generar un resultado de salida a partir de la red. Las activaciones dentro de las capas completamente conectadas 908 se pueden computar usando una multiplicación matricial en lugar de una convolución. No todas las implementaciones de CNN hacen uso de las capas completamente conectadas 908. Por ejemplo, en algunas implementaciones, la capa convolucional 906 puede generar una salida para la CNN.

Las capas convolucionales están conectadas de manera dispersa, lo que difiere de la configuración de red neuronal tradicional encontrada en las capas completamente conectadas 908. Las capas de red neuronal tradicionales están completamente conectadas, de manera que cada unidad de salida interacciona con cada unidad de entrada. Sin embargo, las capas convolucionales están conectadas de manera dispersa debido a que se introduce la salida de la convolución de un campo (en lugar del valor de estado respectivo de cada uno de los nodos en el campo) en los nodos de la capa subsiguiente, como se ilustra. Los núcleos asociados con las capas convolucionales realizan operaciones de convolución, la salida de los cuales se envía a la capa siguiente. La reducción de dimensionalidad realizada dentro de las capas convolucionales es un aspecto que habilita que la CNN realice un ajuste a escala para procesar imágenes grandes.

La Figura 9B ilustra fases de cómputo ilustrativas dentro de una capa convolucional de una CNN. La entrada a una capa convolucional 912 de una CNN se puede procesar en tres fases de una capa convolucional 914. Las tres fases pueden incluir una fase de convolución 916, una fase de detección 918 y una fase de agrupación 920. La capa de convolución 914 puede emitir entonces datos a una capa convolucional sucesiva. La capa convolucional final de la red puede generar datos de correlación de características de salida o proporcionar una entrada a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

En la fase de convolución 916 se realizan varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La fase de convolución 916 puede incluir una transformación afín, que es cualquier transformación que se pueda especificar como una transformación lineal más una traslación. Las transformaciones afines incluyen rotaciones, traslaciones, ajuste a escala y combinaciones de estas transformaciones. La fase de convolución computa la salida de funciones (por ejemplo, neuronas) que están conectadas a regiones específicas en la entrada, lo que se puede determinar como la región local asociada con la neurona. Las neuronas computan un producto escalar entre los pesos de las neuronas y la región en la entrada local a la que están conectadas las neuronas. La salida desde la fase de convolución 916 define un conjunto de activaciones lineales que son procesadas por fases sucesivas de la capa convolucional 914.

Las activaciones lineales pueden ser procesadas por una fase de detección 918. En la fase de detección 918, 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 global sin afectar a los campos receptivos de la capa de convolución. Se pueden usar varios tipos de funciones de activación no lineal. Un tipo particular es la unidad lineal rectificada (ReLU), que usa una función de activación definida como $f(x) = \text{máx}(0, x)$, de manera que se fija un umbral de cero para la activación.

La fase de agrupación 920 usa una función de agrupación que sustituye la salida de la capa convolucional 906 con una estadística de resumen de las salidas cercanas. La función de agrupación se puede usar para introducir la invarianza de traslación en la red neuronal, de manera que traslaciones pequeñas a la entrada no cambian las salidas agrupadas. La invarianza a la traslación local puede ser útil en escenarios donde la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Se pueden usar diversos tipos de funciones de agrupación durante la fase de agrupación 920, incluyendo agrupación máxima, agrupación promedio y agrupación de norma l2. Adicionalmente, algunas implementaciones de CNN no incluyen una fase de agrupación. En su lugar, tales implementaciones sustituyen una fase de convolución adicional que tiene un paso aumentado en relación con fases de convolución previas.

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

La **Figura 10** ilustra una red neuronal recurrente 1000 ilustrativa. En una red neuronal recurrente (RNN), el estado previo de la red influye sobre la salida del estado actual de la red. Las RNN se pueden construir de una diversidad de maneras usando una diversidad de funciones. El uso de las RNN pivota, en general, alrededor del uso de modelos matemáticos para predecir el futuro basándose en una secuencia anterior de entradas. Por ejemplo, una RNN se puede usar para realizar un modelado de idioma estadístico para predecir una palabra venidera, dada en una secuencia previa de palabras. La RNN 1000 ilustrada se puede describir como que tiene una capa de entrada 1002 que recibe un vector de entrada, las capas ocultas 1004 para implementar una función recurrente, un mecanismo de realimentación 1005 para habilitar una 'memoria' de estados previos y una capa de salida 1006 para emitir un resultado. La RNN 1000 opera basándose en escalones de tiempo. El estado de la RNN en un escalón de tiempo dado se ve influenciado basándose en el escalón de tiempo previo a través del mecanismo de realimentación 1005. Para un escalón de tiempo dado, el estado de las capas ocultas 1004 se define por el estado previo y la entrada en el escalón de tiempo actual. Una entrada inicial (x_1) en un primer escalón de tiempo puede ser procesada por la capa oculta 1004. Una segunda entrada (x_2) puede ser procesada por la capa oculta 1004 usando información de estado que se determina durante el procesamiento de la entrada inicial (x_1). Un estado dado se puede computar como $s_t = f(Ux_t + Ws_{t-1})$, en donde U y W son matrices de parámetros. La función f es, en general, una no linealidad, tal como la función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x) = máx(0, x)$. Sin embargo, la función matemática específica usada en las capas ocultas 1004 puede variar dependiendo de los detalles de implementación específicos de la RNN 1000.

Además de las redes CNN y RNN básicas descritas, se pueden habilitar variaciones a esas redes. Una variante de RNN ilustrativa es la RNN de memoria a corto plazo larga (LSTM). Las RNN de LSTM son capaces de aprender dependencias a largo plazo que pueden ser necesarias para procesar secuencias de idioma más largas. Una variante de la CNN es una red de creencia profunda convolucional, que tiene una estructura similar a una CNN y se entrena de una manera similar a una red de creencia profunda. Una red de creencia profunda (DBN) es una red neuronal generativa que está compuesta por múltiples capas de variables estocásticas (aleatorias). Las DBN se pueden entrenar capa a capa usando aprendizaje no supervisado voraz. Los pesos aprendidos de la DBN pueden usase entonces para proporcionar redes neuronales de preentrenamiento determinando un conjunto inicial óptimo de pesos 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 dada para una tarea, la red neuronal se entrena usando un conjunto de datos de entrenamiento 1102. Se han desarrollado diversas estructuras de entrenamiento 1104 para habilitar la aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura de aprendizaje automático 604 de la Figura 6 se puede configurar como una estructura de entrenamiento 604. La estructura de entrenamiento 604 se puede enganchar a una red neuronal no entrenada 1106 y habilitar que la red neuronal no entrenada se entrene usando los recursos de procesamiento paralelo descritos en el presente documento para generar una red neuronal entrenada 1108.

Para iniciar el proceso de entrenamiento, los pesos iniciales se pueden elegir aleatoriamente o mediante preentrenamiento usando una red de creencia profunda. El ciclo de entrenamiento se puede realizar entonces de una manera o bien supervisada o bien no supervisada.

El aprendizaje supervisado es un método de aprendizaje en el que un entrenamiento se realiza como una operación mediada, tal como cuando el conjunto de datos de entrenamiento 1102 incluye una entrada emparejada con la salida deseada para la entrada, o donde el conjunto de datos de entrenamiento incluye una entrada que tiene una salida conocida, y la salida de la red neuronal se califica manualmente. La red procesa las entradas y compara las salidas resultantes contra un conjunto de salidas esperadas o deseadas. Los errores se retropropagan entonces a través del sistema. La estructura de entrenamiento 1104 se puede ajustar para ajustar los pesos que controlan la red neuronal no entrenada 1106. La estructura de entrenamiento 1104 puede proporcionar herramientas para supervisar cómo está convergiendo de bien la red neuronal no entrenada 1106 hacia un modelo adecuado para generar respuestas correctas basándose en datos de entrada conocidos. El proceso de entrenamiento tiene lugar repetidamente a medida que se ajustan los pesos de la red para perfeccionar la salida generada por la red neuronal. El proceso de entrenamiento puede continuar hasta que la red neuronal alcanza una precisión estadísticamente deseada asociada con una red neuronal entrenada 1108. La red neuronal entrenada 1108 se puede desplegar entonces para implementar cualquier número de operaciones de aprendizaje automático.

El aprendizaje no supervisado es un método automático en el que la red intenta entrenarse a sí misma usando datos no etiquetados. Por lo tanto, para un aprendizaje no supervisado, el conjunto de datos de entrenamiento 1102 incluirá datos de entrada sin dato de salida asociado alguno. La red neuronal no entrenada 1106 puede aprender agrupamientos dentro de la entrada no etiquetada y puede determinar cómo las entradas individuales están relacionadas con el conjunto de datos global. El entrenamiento no supervisado se puede usar para generar una

correlación de autoorganización, que es un tipo de red neuronal entrenada 1107 que puede realizar operaciones útiles en cuanto a la reducción de la dimensionalidad de los datos. El entrenamiento no supervisado se puede usar también para realizar una detección de anomalías, lo que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

También se pueden emplear variaciones al entrenamiento supervisado y no supervisado. El aprendizaje semisupervisado es una técnica en la que el conjunto de datos de entrenamiento 1102 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que se usan continuamente datos de entrada para entrenar adicionalmente el modelo. El aprendizaje incremental habilita que la red neuronal entrenada 1108 se adapte a los datos nuevos 1112 sin olvidar el conocimiento inculcado dentro de la red durante el entrenamiento inicial.

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

La **Figura 12** es un diagrama de bloques que ilustra un aprendizaje distribuido. El aprendizaje distribuido es un modelo de entrenamiento que usa múltiples nodos informáticos distribuidos para realizar un entrenamiento supervisado o no supervisado de una red neuronal. Cada uno de los nodos computacionales distribuidos puede incluir uno o más procesadores de anfitrión y uno o más de los nodos de procesamiento de propósito general, tales como la unidad de procesamiento de gráficos de propósito general altamente paralela 700, como en la Figura 700. Como se ilustra, un aprendizaje distribuido se puede realizar con el paralelismo de modelo 1202, el paralelismo de datos 1204 o una combinación del paralelismo de modelo y de datos 1204.

En el paralelismo de modelo 1202, diferentes nodos computacionales en un sistema distribuido pueden realizar cómputos de entrenamiento para diferentes partes de una única red. Por ejemplo, cada capa de una red neuronal puede ser entrenada por un nodo de procesamiento diferente del sistema distribuido. Los beneficios del paralelismo de modelo incluyen la capacidad de ajustar a escala a modelos particularmente grandes. La división de los cómputos asociados con diferentes capas de la red neuronal habilita el entrenamiento de redes neuronales muy grandes en las que los pesos de todas las capas no encajarían en la memoria de un único nodo computacional. En algunas instancias, el paralelismo de modelo puede ser particularmente útil en la ejecución de un entrenamiento no supervisado de redes neuronales grandes.

En el paralelismo de datos 1204, 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 desde los diferentes nodos se combinan entonces. Aunque son posibles diferentes enfoques al paralelismo de datos, los enfoques de entrenamiento de datos paralelos requieren, todos ellos, una técnica de combinación de resultados y de sincronización de los parámetros de modelo entre cada nodo. Los enfoques ilustrativos a la combinación de datos incluyen promediado de parámetros y paralelismo de datos basado en actualizaciones. El promediado de parámetros entrena cada nodo en un subconjunto de los datos de entrenamiento y establece los parámetros globales (por ejemplo, pesos, desviaciones) al promedio de los parámetros desde cada nodo. El promediado de parámetros usa un servidor de parámetros central que mantiene los datos de parámetro. El paralelismo de datos basado en actualizaciones es similar al promediado de parámetros excepto en que, en lugar de transferir parámetros desde los nodos al servidor de parámetros, se transfieren las actualizaciones al modelo. Adicionalmente, el paralelismo de datos basado en actualizaciones se puede realizar de una manera descentralizada, en donde las actualizaciones se comprimen y se transfieren entre nodos.

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

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

Aplicaciones de aprendizaje automático ilustrativas

El aprendizaje automático se puede aplicar a resolver una diversidad de problemas tecnológicos, incluyendo, pero sin limitación, visión por ordenador, conducción y navegación autónoma, reconocimiento de habla y procesamiento de idioma. La visión por ordenador ha sido tradicionalmente una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Las aplicaciones de visión por ordenador varían de reproducir capacidades visuales humanas, tales como reconocer caras, a crear nuevas categorías de capacidades visuales. Por ejemplo, las aplicaciones de visión por ordenador se pueden configurar para reconocer ondas de sonido de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesador paralelo posibilita que se entrenen aplicaciones de visión por ordenador usando un conjunto de datos de entrenamiento

significativamente más grande que el previamente factible y habilita que se desarrollen sistemas de inferenciación usando procesadores paralelos de baja potencia.

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

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

El aprendizaje automático acelerado por procesador paralelo se puede usar también para acelerar el procesamiento de lenguaje natural. Los procedimientos de aprendizaje automático pueden hacer uso de algoritmos de inferencia estadística para producir modelos que son robustos ante una entrada errónea o extraña. Las aplicaciones de procesador de lenguaje natural ilustrativas incluyen la traducción mecánica automática entre idiomas humanos.

Las plataformas de procesamiento paralelo usadas para el aprendizaje automático se pueden dividir en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento son, en general, altamente paralelas e incluyen optimizaciones para acelerar el entrenamiento de múltiples GPU y un único nodo y el entrenamiento de múltiples nodos y múltiples GPU. Los procesadores paralelos ilustrativos adecuados para el entrenamiento incluyen la unidad de procesamiento de gráficos de propósito general altamente paralela 700 de la Figura 700 y el sistema informático de múltiples GPU 800 de la Figura 800. Por el contrario, las plataformas de aprendizaje automático desplegadas incluyen, en general, procesadores paralelos de potencia inferior adecuados para su uso en productos tales como cámaras, robots autónomos y vehículos autónomos.

La **Figura 13** ilustra un sistema en un chip (SOC) de inferenciación 1300 ilustrativo adecuado para realizar una inferenciación usando un modelo entrenado. El SOC 1300 puede integrar componentes de procesamiento que incluyen un procesador de medios 1302, un procesador de visión 1304, una GPGPU 1306 y un procesador de múltiples núcleos 1308. El SOC 1300 puede incluir adicionalmente una memoria en chip 1305 que puede habilitar una agrupación de datos en chip compartida que es accesible por cada uno de los componentes de procesamiento. Los componentes de procesamiento se pueden optimizar para un funcionamiento de baja potencia para habilitar el despliegue en una diversidad de plataformas de aprendizaje automático, incluyendo vehículos autónomos y robots autónomos. Por ejemplo, una implementación del SOC 1300 se puede usar como una porción del sistema de control principal para un vehículo autónomo. Donde el SOC 1300 está configurado para su uso 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 de medios 1302 y el procesador de visión 1304 pueden trabajar conjuntamente para acelerar operaciones de visión por ordenador. El procesador de medios 1302 puede habilitar la descodificación de latencia baja de múltiples flujos de vídeo de alta resolución (por ejemplo, 4K, 8K). Los flujos de vídeo descodificados se pueden escribir en una memoria intermedia en la memoria en chip 1305. El procesador de visión 1304 puede analizar entonces el vídeo descodificado y realizar operaciones de procesamiento preliminares sobre las tramas del vídeo descodificado como preparación al procesamiento de las tramas usando un modelo de reconocimiento de imágenes entrenado. Por ejemplo, el procesador de visión 1304 puede acelerar operaciones de convolución para una CNN que se usa para realizar un reconocimiento de imágenes sobre los datos de vídeo de alta resolución, mientras que los cómputos de modelo de extremo trasero son realizados por la GPGPU 1306.

El procesador de múltiples núcleos 1308 puede incluir lógica de control para asistir con la secuenciación y la sincronización de transferencias de datos y operaciones de memoria compartida realizadas por el procesador de medios 1302 y el procesador de visión 1304. El procesador de múltiples núcleos 1308 también puede funcionar como un procesador de aplicaciones para ejecutar aplicaciones de software que pueden hacer uso de la capacidad de cómputo de inferenciación de la GPGPU 1306. Por ejemplo, al menos una porción de la lógica de navegación y de conducción se puede implementar en software que se ejecuta en el procesador de múltiples núcleos 1308. Tal software puede emitir directamente cargas de trabajo computacionales a la GPGPU 1306 o las cargas de trabajo computacionales se pueden emitir al procesador de múltiples núcleos 1308, que puede descargar al menos una porción de esas operaciones a la GPGPU 1306.

La GPGPU 1306 puede incluir agrupaciones de cómputo, tales como una configuración de baja potencia de las agrupaciones de cómputo 706A-706H dentro de la unidad de procesamiento de gráficos de propósito general

altamente paralela 700. Las agrupaciones de cómputo dentro de la GPGPU 1306 pueden soportar instrucciones que se optimizan específicamente para realizar cómputos de inferenciación sobre una red neuronal entrenada. Por ejemplo, la GPGPU 1306 puede soportar instrucciones para realizar cómputos de precisión baja, tales como operaciones vectoriales de números enteros de 8 bits y de 4 bits.

Hardware especializado para operaciones de aprendizaje automático eficientes

Las realizaciones descritas en el presente documento proporcionan primitivas computacionales de aprendizaje automático de alto nivel que se pueden usar para abstraer muchos de los detalles computacionales subyacentes de realizar cálculos de aprendizaje automático. Las primitivas de alto nivel descritas en el presente documento habilitan que una lógica de software solicite operaciones de aprendizaje automático de alto nivel al tiempo que se abstraen los detalles de implementación subyacentes de esas operaciones. Por ejemplo, y en una realización, una lógica de software puede solicitar una operación de convolución para una imagen usando un conjunto dado de filtros. Se puede ejecutar una única instrucción de alto nivel que tiene operandos para definir direcciones de memoria intermedia de entrada y de salida y direcciones para memorias intermedias que almacenan datos de filtro y/o de núcleo. La GPGPU puede dividir entonces la instrucción de convolución de alto nivel en múltiples suboperaciones que son realizadas por las unidades de cómputo subyacentes de la GPGPU. En una realización, se proporciona soporte de hardware directo para una o más subrutinas de los subprogramas de algoritmos lineales básicos (BLAS), aunque las realizaciones pueden proporcionar soporte de hardware para otras bibliotecas de subrutinas. La lógica de compilador y las bibliotecas de tiempo de ejecución asociadas pueden compilar código fuente que hace uso de subrutinas de cómputo de alto nivel soportadas y emitir código fuente compilado que llama a una unidad de macroinstrucción de aprendizaje automático.

Instrucciones y lógica para realizar operaciones de cómputo para aprendizaje automático

Los aceleradores de hardware para la visión por ordenador y el aprendizaje automático pueden mejorar la eficiencia energética para aplicaciones tales como el reconocimiento de objetos, de rostros y de habla en órdenes de magnitud. Estos aceleradores usan matrices de elementos de procesamiento (PE) interconectadas, con circuitos de multiplicación - suma que son dominantes en cuanto a desempeño, área y energía para correlacionar algoritmos clave usados para operaciones de cómputo de CNN. Por ejemplo, algunos aceleradores de hardware de aprendizaje automático usan bloques de construcción de ruta de datos de multiplicación - suma de coma fija de anchura de bits estrecha (16 b) para cumplir con los presupuestos estrictos de memoria, de área y de energía para los SoC en el espacio integrado o de baja potencia. Se puede lograr una calidad mejor de los resultados para algunos conjuntos de datos y algoritmos con el rango dinámico superior ofrecido por los números/cómputos de coma flotante al tiempo que se sigue manteniendo la misma huella de memoria (operandos de 16 b). Soluciones de hardware anteriores para dar cabida a ambos tipos de cómputos numéricos emplean unos PE o rutas de datos de coma fija y de coma flotante separados, dando como resultado un coste de área alto para lograr esta flexibilidad. En su lugar, una realización descrita en el presente documento proporciona una ruta de datos de multiplicación - suma y de multiplicación - acumulación fusionada de números enteros/coma flotante combinada que utiliza circuitos de multiplicación - suma de números enteros con signo existentes para conseguir operaciones de multiplicación - suma de mantisa de coma flotante. En una realización, al añadir solo los circuitos requeridos para desplazamientos de alineación/normalización y una unidad de exponente, se habilita un soporte de coma flotante en una unidad de coma flotante/números enteros combinada sin aumentar la anchura de datos de entrada/salida y la huella de memoria de datos. Se utiliza una única señal de control para conmutar, de una manera por ciclo, entre los modos de cómputo de coma flotante y de números enteros.

Las unidades de números enteros/coma flotante combinadas proporcionadas por realizaciones se complementan con múltiples tipos de unidades de aceleración de aprendizaje automático que se pueden integrar en una GPGPU. Las realizaciones descritas en el presente documento proporcionan una lógica para habilitar instrucciones adicionales que combina una operación de multiplicación y de suma fusionada con una función de activación de red neuronal, tal como la función de unidad lineal rectificada (RELU), la función sigmoide o la función sigmoide rígida.

Una realización habilita una ampliación de la codificación de coma flotante de 16 bits para soportar codificaciones alternativas con respecto al formato de coma flotante de precisión media de la norma IEEE 754. El formato de coma flotante de precisión media de IEEE especifica un signo de 1 bit, un exponente de 5 bits y una porción fraccionaria de 10 bits. Las realizaciones descritas en el presente documento pueden soportar, de manera selectiva, codificaciones alternativas de datos FP16 basándose en el patrón de datos a codificar. En una realización, un formato alternativo soportado especifica un signo de 1 bit, con un exponente de 8 bits y un componente fraccionario de 7 bits. Una realización permite una codificación con un signo de 1 bit, un exponente de 3 bits y un componente fraccionario de 12 bits. En tales realizaciones, diferentes conjuntos de instrucciones soportan diferentes codificaciones de coma flotante, permitiendo que un desarrollador seleccione una codificación basándose en la instrucción especificada en un código de programa. En una realización, se pueden usar diferentes codificaciones de coma flotante cuando se redondean o se muestrean en sentido descendente datos de coma flotante, por ejemplo, desde un valor de coma flotante de 32 bits acumulado a un valor de 16 bits.

Las unidades de coma flotante combinadas descritas en el presente documento pueden realizar, de manera selectiva,

operaciones de números enteros o de coma flotante de 16 bits de una manera por ciclo. Una realización habilita una reconfiguración dinámica de las unidades de coma flotante descritas en el presente documento para habilitar un soporte de múltiples formatos. Por ejemplo, usando una configuración de múltiples pasadas, las unidades de números enteros o de coma flotante de 16 bits se pueden configurar para realizar una operación de 32 bits de dos pasadas o una operación de 64 bits de cuatro pasadas. Tal lógica habilita que una lógica de coma flotante que se optimiza para operaciones de inferenciación de precisión inferior se agrupe para su uso en operaciones de entrenamiento de precisión superior.

Una realización proporciona una unidad de redondeo estocástico y un acumulador de estadísticas para redes de precisión baja. El redondeo estocástico habilita una precisión aumentada con respecto a la cuantificación y el redondeo clásicos para redes neuronales profundas de precisión baja. La unidad de redondeo puede trabajar en diferentes modos. Un primer modo es un modo aleatorio que usa un generador de números aleatorios para controlar la unidad de redondeo. Un segundo modo usa una distribución de probabilidad de salidas a lo largo de entradas subsiguientes y hace uso de una unidad de estimación de estadística de datos cercanos acoplada a una memoria GPGPU.

Las técnicas descritas en el presente documento se pueden implementar dentro de un sistema computacional de propósito general con optimizaciones de aprendizaje automático proporcionadas a través de unidades de aceleración de aprendizaje automático. En la Figura 14 se muestra un multiprocesador proporcionado por realizaciones descritas en el presente documento.

La **Figura 14** es un diagrama de bloques de una unidad de multiprocesador 1400, de acuerdo con una realización. La unidad de multiprocesador 1400 puede ser una variante de un multiprocesador de gráficos 234 de la Figura 2D. La unidad de multiprocesador 1400 incluye una unidad de extracción y de descodificación 1402, una unidad de bifurcación 1404, un archivo de registro 1406, un gestor de hilos 1406, una unidad de una única instrucción - múltiples hilos (la unidad de SIMT 1410) y un gestor de voltaje y de frecuencia 1420. La unidad de extracción y de descodificación 1402 puede extraer una instrucción para su ejecución por la unidad de multiprocesador 1400. La unidad de bifurcación 1404 puede computar ajustes de puntero de instrucción basándose en una instrucción de salto ejecutada. El archivo de registro 1406 puede almacenar registros arquitectónicos y de propósito general usados por la unidad de SIMT 1410. El gestor de hilos 1406 puede distribuir y redistribuir hilos entre las unidades de cómputo de la unidad de SIMT 1410. En una realización, la unidad de SIMT 1410 está configurada para ejecutar una única instrucción como múltiples hilos, con cada hilo de la instrucción ejecutado por una unidad de cómputo separada. En una realización, cada una de la unidad de cómputo 1411 a la unidad de cómputo 1418 incluye una ALU de números enteros (por ejemplo, la ALU 1411A-1418A) y una unidad de coma flotante (por ejemplo, la FPU 1411B-1418B). El voltaje y la frecuencia de cada unidad de cómputo 1411-1418 dentro de la unidad de SIMT 1410 pueden ser gestionados dinámicamente por el gestor de voltaje y de frecuencia 1420, que puede aumentar o disminuir el voltaje y la frecuencia de reloj suministrados a las diversas unidades de cómputo cuando se habilitan y se deshabilitan componentes de las unidades de cómputo.

En algunas configuraciones previamente habilitadas, cada unidad de cómputo puede ejecutar un único hilo o bien de una instrucción de números enteros o bien de una instrucción de coma flotante. Si a alguna de las ALU 1411A-1418A se le asigna la tarea de ejecutar un hilo de una instrucción de números enteros, la FPU 1411B-FPU1418B respectiva no está disponible para su uso para ejecutar un hilo de una instrucción de coma flotante y se puede controlar por alimentación durante el funcionamiento de la ALU 1411A-ALU 1418A correspondiente. Por ejemplo, aunque la ALU 1411A puede ejecutar un hilo de una instrucción de números enteros mientras la FPU 1413B ejecuta un hilo de una instrucción de coma flotante, la FPU 1411B se controla por alimentación mientras la ALU 1411A está activa. Las realizaciones descritas en el presente documento superan tales limitaciones habilitando, por ejemplo, que la ALU 1411A ejecute un hilo de una instrucción mientras la FPU 1411B ejecuta un hilo de una instrucción diferente. Además, una realización proporciona soporte para operandos de precisión mixta o de tipo de datos mixto, de manera que una única unidad de cómputo puede realizar simultáneamente operaciones para una instrucción que tiene operandos de coma flotante y de números enteros y/u operandos que tienen diferentes precisiones.

Las realizaciones descritas en el presente documento habilitan un caudal operativo aumentado para un grupo de unidades de cómputo al hacer que todas las unidades de lógica dentro de cada unidad de cómputo estén disponibles para realizar cómputos. En tales realizaciones, las unidades de lógica dentro de una unidad de cómputo que están diseñadas para realizar cómputos de manera selectiva con una de múltiples precisiones o múltiples tipos de datos se pueden configurar para realizar múltiples operaciones simultáneas para cada precisión o tipo de datos soportado por la unidad de cómputo. Para una unidad de cómputo 1411-1418 dada, las ALU 1411A-1418A pueden realizar operaciones de números enteros, mientras que las FPU 1411B-1418B realizan operaciones de coma flotante. Estas operaciones se pueden realizar para una única instrucción o para múltiples instrucciones. En una realización, se habilita una clase nueva de instrucción de precisión mixta en la que uno o más operandos son de un tipo de datos o de una precisión, mientras que uno o más operandos diferentes son de un tipo de datos o de una precisión diferente. Por ejemplo, una instrucción puede aceptar dos o más operandos de múltiples elementos que incluyen tipos de datos de números enteros y de coma flotante y una única instrucción se ejecuta de una manera por tipo de datos o de una manera por precisión.

Una unidad de multiplicación - suma fusionada de coma flotante/de números enteros de 16 bits reconfigurable

Los diseños de unidad de lógica proporcionados por realizaciones descritas en el presente documento tienen una latencia de un único ciclo y de múltiples ciclos, al tiempo que se mantiene un caudal de un único ciclo tanto para la multiplicación - suma fusionada (por ejemplo, una entrada de 3 operandos sin dependencia alguna a lo largo de ciclos) como para la multiplicación - acumulación (por ejemplo, una entrada de 2 operandos con dependencia de datos a lo largo de ciclos). En contraposición, los diseños de unidad de lógica conocidos en la técnica implementan una multiplicación - suma fusionada sin tener en cuenta operaciones de multiplicación - acumulación de latencia de múltiples ciclos y de caudal de un único ciclo, lo que puede ser un factor limitante para el desempeño para operaciones de aprendizaje automático clave, tales como la operación de producto escalar.

Una realización descrita en el presente documento proporciona una ruta de datos de multiplicación - suma fusionada de números enteros/coma flotante combinada que utiliza los circuitos de multiplicación - suma de números enteros con signo existentes para conseguir también operaciones de multiplicación - suma de mantisa de coma flotante. Con la adición de solo los circuitos requeridos para los desplazamientos de alineación/normalización y la unidad de exponente, se habilita un soporte de coma flotante. Las anchuras de datos de entrada/salida y la huella de memoria de datos siguen siendo iguales, con solo una única señal de control requerida para conmutar de una manera por ciclo entre los dos modos de cómputo.

Una realización proporciona un diseño de multiplicación - suma fusionada de números enteros/coma flotante de 16 bits combinada que mejora los diseños convencionales de un único ciclo con rutas de datos de números enteros/coma flotante separadas. Los diseños descritos en el presente documento logran circuitos de multiplicación - suma para una ruta de datos int16/float16 combinada que reduce el área total hasta en un 29 %. Una realización proporciona una ruta de datos de coma flotante mejorada con alineación solo para un sumando junto con un incrementador de negación y de redondeo combinado que contribuye a un 11 % de la reducción de área total. Una realización proporciona una variante de multiplicación - acumulación con dos entradas y un caudal de latencia de dos ciclos de un único ciclo. Una realización proporciona un circuito alternativo que aumenta significativamente la precisión de acumulación duplicando la anchura de acumulador con un coste en el aumento de área solo un 11 %.

Las **Figuras 15A-15B** ilustran diseños para unidades de lógica para realizar operaciones de multiplicación - suma fusionada de números enteros y de coma flotante, de acuerdo con una realización. La **Figura 15A** muestra un diseño convencional para una unidad de lógica 1500 que habilita una operación de multiplicación - suma fusionada al tiempo que se mantienen el rango y la precisión completos del producto intermedio. La operación de multiplicación - suma fusionada (o = a * b + c) se realiza sobre tres operandos de entrada de 16 bits 1501 en los modos o bien de coma flotante de precisión media (float16) o bien de números enteros de 16 b (int16) con signo de IEEE. Las entradas se proporcionan o bien a una ruta de datos de coma flotante de 16 bits 1510 o bien a una ruta de datos de números enteros de 16 bits 1520, con el puerto de salida (o 1530) seleccionando el resultado apropiado (f16o 1518 o i16o 1528) basándose en el modo de funcionamiento 1532. El resultado int16 (i16o 1528) selecciona y redondea al más cercano la mitad superior del resultado entero con signo de 32 b (isum 1525) que es generado por un multiplicador de 16 b × 16 b con signo 1521 y un sumador de 32 b 1522. La ruta de datos float16 1510 desplaza a la derecha (1511) la mantisa del más pequeño del producto de un multiplicador de 11 b × 11 b sin signo 1617 y desplaza a la derecha el sumando para la alineación en un desplazador de alineación 1512A antes de procesar el producto a través de un sumador de mantisa de 22 bits 1513. Un anticipador de cero inicial de 22 bits (LZA 1519) predice la ubicación de la ubicación de bits más significativos del resultado de la suma de coma flotante realizada por el sumador de mantisa de 22 bits 1513 basándose en las entradas al sumador. Un desplazamiento a la izquierda (1514) es realizado por un desplazador de normalización 1515 antes de que el resultado intermedio se proporcione a la lógica de redondeo 1516.

La **Figura 15B** es un diagrama de bloques de una unidad de lógica de multiplicación - suma 1540, de acuerdo con una realización. La unidad de lógica 1540 de la Figura 15B mantiene circuitos de coma flotante/números enteros de 16 bits separados al tiempo que se mejora la ruta de datos de coma flotante de la unidad de lógica 1500. En una realización, el diseño de la unidad de lógica 1540 quita el desplazador de alineación 1512B de la ruta crítica realizando la alineación solo sobre el sumando, en paralelo con la operación de multiplicación (1541). La suma de 33 bits más amplia solo requiere un incrementador de 11 bits para los bits superiores. Adicionalmente, para operaciones de resta, la salida del sumador se puede negar para producir una mantisa sin signo. En una realización, el incrementador se quita de la ruta crítica de la ruta de datos de la unidad de lógica 1540 combinando la operación de incremento con el incrementador de redondeo final (1542). Por el contrario, la unidad de lógica 1500 de la Figura 15A requiere un incrementador para completar cualquier operación de negación de complemento a dos requerida después del sumador. La reducción de ruta crítica con la ruta de datos de coma flotante de 16 bits de la unidad de lógica 1540 da como resultado unas puertas más pequeñas y permite una reducción de área de un 11 % en relación con la unidad de lógica 1500 al tiempo que se mantiene la misma latencia de un único ciclo.

La **Figura 16** ilustra la unidad de lógica de multiplicación - suma fusionada 1600 que tiene una ruta de datos de coma flotante y de números enteros combinada, de acuerdo con una realización. Un multiplicador con signo de 16 bits × 16 bits 1602A y un sumador de 32 bits 1604 de la ruta de datos de números enteros se reutilizan para operaciones de mantisa de coma flotante con bits de operando superiores controlados para producir resultados para mantisas de 11 bits (1602B). Los conmutadores de entrada 1601A-1601C se usan para redirigir los 6 bits superiores de los operandos de entrada (a, b, c) a una unidad de exponente 1608 cuando se habilita el modo de coma flotante. Los valores de signo y de exponente a partir de las entradas se empaquetan y se proporcionan a una unidad de exponente 1608 a

través de un bus de operandos de signo de 3 bits fijo 1609A y un bus de exponente de 15 bits 1609B. Para operaciones de coma flotante de 16 bits, el sumador de 32 bits compartido usa un incrementador de 1 bit 1605 para crear el bit o bits superiores 1606 de la suma de 33 bits. Los circuitos de derivación (1610A, 1610B) dentro de la unidad de exponente 1608, así como en el desplazador de alineación 1612 y el desplazador de normalización 1613, aseguran una alineación/normalización fija con una actividad de conmutación mínima en esas unidades para el modo de números enteros, mientras que los bits de mantisa superior cero aseguran la ausencia de conmutación actividad dentro de las porciones no usadas del multiplicador en el modo de coma flotante. La lógica de redondeo 1616 y el incrementador de la ruta de datos de coma flotante se reutilizan para el modo de números enteros para computar los 10 bits inferiores del resultado entero i16o con redondeo. Los 6 bits superiores de i16o se computan correlacionando esa operación en el incrementador de exponente 1611 existente, que también realiza cualquier operación de desbordamiento de redondeo a partir de la ruta de datos de mantisa en el modo de coma flotante. Se puede proporcionar un valor entero o de coma flotante de 16 bits a través de una salida 1630 cuando se completa el procesamiento.

La **Figura 17A** ilustra una unidad de lógica 1700 que incluye circuitos de cómputo combinado para realizar operaciones de multiplicación - acumulación fusionada de coma flotante y de números enteros, de acuerdo con una realización. La unidad de lógica 1700 incluye una unidad de exponente 1708 y una unidad de mantisa 1709, dos puertos de entrada de 16 bits 1701 y un puerto de salida de 16 bits 1730. Los puertos de entrada 1701 incluyen conmutadores para conmutar bits de signo y de exponente de los datos de entrada a la unidad de exponente 1708. La unidad de exponente 1708 y la unidad de mantisa 1709 se usan cuando se realizan operaciones de números enteros. En una realización, la unidad de lógica soporta un formato de entrada de 8,8 y de salida de 16,0 para un modo de coma fija de 16 bits. La unidad de lógica 1700 soporta unos requisitos de latencia de dos ciclos y de caudal de un único ciclo. Algunos de los circuitos ilustrados se comparten entre modos operativos, incluyendo el multiplicador con signo 1702A-1702B y el sumador de 32 bits 1704, que se usan para los modos tanto de números enteros como de coma flotante. Se impone una entrada de acumulador de 16 bits 1703A durante una acumulación en el segundo ciclo, en donde el valor del acumulador se proporciona al sumador de 32 bits 1704. Los 10 bits superiores de la entrada de acumulador 1703A (por ejemplo, c[15:6]) son exclusivos de una operación de números enteros de 16 bits. Para ambos modos de cómputo, la multiplicación se realiza en el primer ciclo y la suma/redondeo en el segundo ciclo.

La unidad de lógica 1700 de la Figura 17A usa tres técnicas clave para habilitar un diseño combinado eficiente. En primer lugar, una canalización directa del diseño combinado de un único ciclo de la Figura 16 para operaciones de acumulación o bien reduciría el caudal a la mitad con la alineación de sumando en el primer ciclo o bien aumenta el tiempo de ciclo con el cómputo de desplazamiento a la derecha y la alineación de 33 b en la ruta crítica para el segundo ciclo. En su lugar, el diseño de la unidad de lógica 1700 aprovecha la no criticidad de temporización/área de la unidad de exponente 1708 para precomputar la mantisa más grande (o más pequeña) y la cantidad de desplazamiento a la derecha para el desplazador de alineación 1713. En una realización, la unidad de lógica 1700 realiza una operación de dos ciclos al tiempo que se mantiene un caudal de un único ciclo realimentando la salida al segundo ciclo como entrada de sumando, escogiendo una mantisa más pequeña para una alineación de solo 22 bits y precomputando la mantisa/cantidad de desplazamiento a la derecha más pequeña en el primer ciclo usando la salida de multiplicador y el exponente de acumulador computado previamente por la segunda fase.

En segundo lugar, una operación de redondeo al más cercano en el modo de números enteros de 16 bits aprovecha el formato de coma fija 8,8 y elimina la necesidad de correlacionar el redondeo de números enteros en el incrementador de redondeo de coma flotante. Un uno es insertado en lugar de un cero en la posición de bit 15 por la lógica de multiplexor 1705 antes del sumador para lograr la misma operación de redondeo.

En tercer lugar, se reutilizan biestables para señales mutuamente excluyentes, tales como el cómputo de exponentes (por ejemplo, el Eun 1707, el Desplazamiento de la derecha 1710) y 10 b superiores del producto (1711) entre los dos modos. La reducción de ruta de temporización en el segundo ciclo también se logra combinando los incrementadores de negación/redondeo y usando optimizaciones basadas en ruta lejana/cercana para reducir la ruta crítica a través del desplazador de alineación 1713 y el desplazador de normalización 1714.

La precisión del diseño de multiplicación - acumulación de dos ciclos se aumenta significativamente duplicando la anchura de solo el acumulador a 32 bits, como se muestra en la **Figura 17B**. El acumulador puede acumular resultados enteros de 16 bits en un formato de coma fija 16,16 y resultados de coma flotante de 16 bits basándose en un resultado intermedio que tiene un exponente de 5 bits y una mantisa de 22 bits (1 inicial implícito no almacenado). La mantisa de 22 bits del resultado intermedio, en diversas realizaciones, se puede redondear, truncar o cuantificar a una mantisa según una norma IEEE. El diseño de la unidad de lógica 1740 limita el coste del acumulador duplicado principalmente a los biestables de salida y al incrementador final en la ruta de datos de mantisa, debido a que la ruta de datos restante después del multiplicador ya da cabida al ancho adicional para el producto. En una realización, la precisión superior habilita simplificar el redondeo a un truncamiento simple para generar una salida de 16 bits 1750 a partir del acumulador de 32 bits. El incrementador de normalización posterior al exponente se quita de la unidad de exponente 1708 en la unidad de lógica 1740. En su lugar, un incrementador de negación 1742 realiza un incremento final en la mantisa para computar el complemento a dos cuando se va a negar la salida del sumador. Se impone una entrada de acumulador de 32 bits 1703B durante una acumulación en el segundo ciclo, en donde el valor del acumulador se proporciona al sumador de 32 bits 1704. Los 10 bits superiores de la entrada de acumulador 1703B (por ejemplo,

c[31:22]) son exclusivos de una operación de números enteros de 16 bits. El área total sintetizada de este diseño presenta solo un aumento del área de un 11 % en relación con el diseño de la unidad de lógica 1700 de la Figura 17A al tiempo que se duplica la precisión de acumulador.

Aunque las descripciones anteriores se proporcionan para operandos de 16 bits, estas técnicas se pueden extender fácilmente a anchuras de datos más grandes para lograr objetivos similares. Adicionalmente, aunque se describe la salida de precisión media de IEEE, los diseños descritos en el presente documento también se pueden ajustar para soportar formatos de coma flotante no convencionales. Adicionalmente, se pueden usar diferentes formatos de coma flotante no convencionales para valores intermedios, como se describe a continuación.

Las realizaciones descritas anteriormente proporcionan diversas implementaciones de una Unidad de Multiplicación - Suma Fusionada de Coma Flotante/Números Enteros de 16 bits reconfigurable que proporciona múltiples ventajas frente a diseños existentes. El diseño propuesto no afecta a la huella de memoria para el almacenamiento de números enteros o de coma flotante. Los diseños propuestos solo aumentan el área de multiplicador sin cambiar el resto de la ruta de datos de coma flotante. En contraposición, los diseños lógicos conocidos en la técnica extienden toda la mantisa/significante de coma flotante a la misma anchura que el número entero, mientras que el área de almacenamiento adicional para el signo y el exponente está separada y es exclusiva solo de los números de coma flotante, dando como resultado un aumento en la huella y en el tamaño de archivo de registro para el almacenamiento de números de coma flotante. Los diseños existentes también aumentan la anchura de toda la ruta de datos de mantisa, lo que puede dar como resultado un aumento de área significativo. Se proporcionan diseños de un único ciclo (por ejemplo, la unidad de lógica 1600 de la Figura 16) y de múltiples ciclos (por ejemplo, la unidad de lógica 1700 de la Figura 17A y la unidad de lógica 1740 de la Figura 17B), en donde los múltiples ciclos, después de una latencia inicial, generan una salida cada ciclo. La unidad de lógica 1740 de la figura 17B proporciona un diseño de multiplicación - acumulación de coma flotante/números enteros combinado con una anchura de acumulador local que es el doble de ancha que los operandos de entrada. Esto habilita una precisión de acumulación muy superior para operaciones como productos escalares sin influir en la huella de almacenamiento de memoria de los operandos de entrada, y afecta a una pequeña porción del diseño para un impacto de área total de solo un 11 %. Además, cada unidad de lógica correlaciona una porción de la operación de números enteros en la ruta de datos de exponente existente para maximizar la reutilización de circuitos cuando se reconfigura para el modo de números enteros. Adicionalmente, para operaciones de coma flotante con operaciones de resta, las unidades de lógica 1540 de la Figura 15B y 1700 de la figura 17A combinan el incremento de complemento a 2 para dar el incremento de redondeo para un retardo y un área reducidos.

Sistema de procesamiento de datos y lógica de aceleración de aprendizaje automático

Una realización usa la unidad de multiprocesador 1400 de la Figura 14 y una o más unidades de lógica de números enteros/coma flotante de las Figuras 15A-17B se pueden usar como bloques de construcción para un sistema de procesamiento de datos de aprendizaje automático que incluye hardware, software y firmware que se optimiza para realizar el tipo de operaciones de cómputo realizadas comúnmente cuando se realiza un entrenamiento o inferenciación usando redes neuronales profundas. Las **Figuras 18A-18B** ilustran un sistema de procesamiento de datos y unidades de cómputo y de lógica asociadas que realizan operaciones de entrenamiento y de inferenciación aceleradas para el aprendizaje automático, por ejemplo, a través del uso de redes neuronales profundas. La **Figura 18A** ilustra un sistema de procesamiento de datos de aprendizaje automático ilustrativo proporcionado por realizaciones descritas en el presente documento. La **Figura 18B** ilustra componentes de un acelerador de aprendizaje automático, de acuerdo con una realización.

El sistema de procesamiento de datos 1800 de la Figura 18A es un sistema de procesamiento heterogéneo que tiene un procesador 1802, una memoria unificada 1810 y una GPGPU 1820 que incluye lógica de aceleración de aprendizaje automático. El procesador 1802 y la GPGPU 1820 pueden ser cualquiera de los procesadores y GPGPU/procesadores paralelos como se describe en el presente documento. El procesador 1802 puede ejecutar instrucciones para un compilador 1815 almacenado en la memoria de sistema 1812. El compilador 1815 se ejecuta en el procesador 1802 para compilar el código fuente 1814A para dar el código compilado 1814B. El código compilado 1814B puede incluir código que puede ser ejecutado por el procesador 1802 y/o código que puede ser ejecutado por la GPGPU 1820. Durante la compilación, el compilador 1815 puede realizar operaciones para insertar metadatos, incluyendo sugerencias en cuanto al nivel de paralelismo de datos presente en el código compilado 1814B y/o sugerencias con respecto a la localidad de datos asociada con hilos a despachar basándose en el código compilado 1814B. El compilador 1815 puede incluir la información necesaria para realizar tales operaciones o las operaciones se pueden realizar con la asistencia de una biblioteca de tiempo de ejecución 1816. La biblioteca de tiempo de ejecución 1816 también puede facilitar al compilador 1815 la compilación del código fuente 1814A y también puede incluir instrucciones que están vinculadas en tiempo de ejecución con el código compilado 1814B para facilitar la ejecución de las instrucciones compiladas en la GPGPU 1820.

La memoria unificada 1810 representa un espacio de direcciones unificado al que pueden acceder el procesador 1802 y la GPGPU 1820. La memoria unificada incluye la memoria de sistema 1812 así como la memoria de GPGPU 1818. La memoria de GPGPU 1818 incluye la memoria local de GPGPU 1834A-1834B dentro de la GPGPU 1820 y también puede incluir parte de o toda la memoria de sistema 1812. Por ejemplo, el código compilado 1814B almacenado en la

memoria de sistema 1812 se puede correlacionar también en la memoria de GPGPU 1818 para el acceso por la GPGPU 1820.

La GPGPU 1820 incluye múltiples bloques de cómputo 1824A-1824N, que pueden ser instancias de la agrupación de procesamiento 214A-214N de la **Figura 2A** y puede incluir una o más instancias del multiprocesador de gráficos 234 descrito en el presente documento. En diversas realizaciones, los bloques de cómputo 1824A-1824N incluyen unidades de cómputo que tienen una o más de las unidades de lógica de las **figuras 15B-17B**. La GPGPU 1820 también incluye un conjunto de registros 1825, una memoria caché 1827 y un módulo de potencia y de desempeño 1826 que se pueden usar como recursos compartidos para los bloques de cómputo 1824A-1824N. En una realización, los registros 1825 incluyen registros accesibles directa e indirectamente, en donde los registros accesibles indirectamente se pueden optimizar para su uso en operaciones de cómputo matricial. El módulo de potencia y de desempeño 1826 se puede configurar para ajustar la entrega de potencia y las frecuencias de reloj para los bloques de cómputo 1824A-1824N para controlar por alimentación componentes en espera dentro de los bloques de cómputo 1824A-1824N bajo cargas de trabajo intensas. La GPGPU 1820 incluye la memoria local de GPGPU 1828, que son módulos de memoria física que comparten una tarjeta de gráficos o un módulo de múltiples chips con la GPGPU 1820.

En una realización, la GPGPU 1820 incluye lógica de hardware que incluye una unidad de extracción y de descodificación 1821, un controlador de planificador 1822 y un acelerador de aprendizaje automático 1823. La unidad de extracción y de descodificación de instrucciones 1821 es una unidad de extracción y de descodificación que incluye lógica para extraer y descodificar instrucciones, incluyendo instrucciones específicas de aprendizaje automático, que pueden definir un comportamiento complejo y personalizable. Las instrucciones pueden hacer que la lógica de cómputo planifique, a través del controlador de planificador 1822, un conjunto de operaciones a realizar a través de uno o más de los bloques de cómputo 1824A-1824N. En una realización, el controlador de planificador 1822 es un ASIC configurable para realizar operaciones de planificación avanzadas. En una realización, el controlador de planificador 1822 es un microcontrolador o un núcleo de procesamiento de baja energía por instrucción capaz de ejecutar instrucciones cargadas desde un módulo de firmware.

En una realización, algunas funciones a realizar por los bloques de cómputo 1824A-1824N se pueden planificar o descargar directamente en el acelerador de aprendizaje automático 1823. El acelerador de aprendizaje automático 1823 incluye una lógica de elemento de procesamiento configurada para realizar eficientemente operaciones matriciales y otras operaciones de cómputo realizadas comúnmente durante un aprendizaje automático.

En algunas realizaciones, la GPGPU 1820 incluye adicionalmente una unidad de estadística 1829 que se puede configurar como una unidad de cómputo de datos cercanos. Por ejemplo, la unidad de estadística 1829 se puede integrar en o distribuir a lo largo de uno o más controladores de memoria para la memoria local de GPGPU 1828. En una realización, la unidad de estadística 1829, cuando es habilitada por el acelerador de aprendizaje automático 1823, se puede usar para determinar una distribución de probabilidad para datos de correlación de activación o peso cuando se realizan operaciones de aprendizaje automático que escriben en o leen desde la memoria local de GPGPU 1828. La unidad de estadística 1829 incluye lógica para determinar, basándose en patrones de dirección y de datos durante un acceso de memoria, si los datos a los que se accede en la memoria local de GPGPU 1828 encajan dentro de una o más distribuciones estadísticas (por ejemplo, gaussiana, uniforme, de Poisson, etc.). En una realización, se puede recopilar información estadística (por ejemplo, media, mediana, moda, desviación típica, etc.) durante un período de muestra para al menos un subconjunto de accesos de memoria. La unidad de estadística 1829 se puede configurar de manera que la recopilación de la información estadística no aumenta significativamente la latencia de accesos de memoria realizados a través de un controlador de memoria que alberga la unidad de estadística 1829. La información estadística se puede proporcionar periódicamente al acelerador de aprendizaje automático 1823, o el acelerador de aprendizaje automático 1823 puede solicitar los datos a partir de la unidad de estadística. En una realización, la unidad de estadística 1829 puede comprobar datos asociados con accesos de memoria frente a un conjunto de distribuciones probables-conocidas. Se puede proporcionar un vector que incluye un conjunto de probabilidades asociadas con cada una de las distribuciones probables-conocidas al acelerador de aprendizaje automático 1823 de una manera periódica o a petición. En diversas realizaciones, el acelerador de aprendizaje automático 1823 puede usar la información de probabilidad y/o estadística proporcionada por la unidad de estadística 1829 para una diversidad de operaciones. En una realización, como se describe adicionalmente en la Figura 18B y en la Figura 20, el acelerador de aprendizaje automático 1823 puede usar datos proporcionados por la unidad de estadística 1829 para realizar un redondeo estocástico durante una cuantificación para redes neuronales de precisión baja.

El acelerador de aprendizaje automático 1823 de la Figura 18A se ilustra con detalle adicional en la Figura 18B. En una realización, el acelerador de aprendizaje automático 1823 incluye un módulo de instrucción de activación 1832, un módulo de codificación y de configuración de FPU 1834, una unidad de cuantificación estocástica 1838 y una memoria caché 1836 que se comparte entre los diversos módulos dentro del acelerador de aprendizaje automático 1823.

El módulo de instrucción de activación 1832 incluye lógica para secuenciar la ejecución de una multiplicación - suma fusionada combinada y activación en respuesta a una única instrucción. En respuesta a la descodificación de una función de FMAC o FMADD más activación en la GPGPU 1820, la unidad de planificación 1822 puede planificar una operación a través del acelerador de aprendizaje automático 1823. El acelerador de aprendizaje automático 1823, a

través del módulo de instrucción de activación 1832, puede realizar un conjunto de operaciones de multiplicación - suma fusionada o de multiplicación - acumulación fusionada en dos o tres operandos de entrada por hilo o elemento de vector y, para cada hilo o elemento, proporcionar la salida a una lógica de hardware configurada para realizar una de múltiples funciones de activación seleccionables. Se puede asociar una función de activación diferente con instrucciones diferentes, o una única instrucción puede incluir un campo para habilitar la selección de una función de activación. En una realización, el módulo de instrucción de activación puede realizar una operación vectorial o de urdimbre para generar un resultado de FMADD o de FMAC intermedio y almacenar los resultados intermedios en la memoria caché 1836. El módulo de instrucción de activación 1832 puede aplicar entonces la función de activación a los datos intermedios. Las funciones de activación soportadas ilustrativas incluyen la función de unidad lineal rectificada (RELU) de la ecuación (1), la función sigmoide de la ecuación (2) o la función sigmoide rígida de la ecuación (3).

$$f(x) = \text{máx}(0, x) \qquad (1)$$

$$\sigma(\text{x}) = \frac{1}{(1+e^{-x})} \qquad (2)$$

$$\sigma(\text{x}) = \text{máx}\left(0, \text{mín}\left(1, \frac{x+1}{2}\right)\right) \qquad (3)$$

El módulo de codificación y de configuración de FPU 1834 incluye lógica para definir parámetros para la configuración dinámica de unidades de coma flotante dentro de los bloques de cómputo 1824A-1824N de la GPGPU 1820. En una realización, ciertos aspectos dinámicos de las unidades de números enteros/coma flotante combinadas de la **Figura 16** y las **Figuras 17A-17B** se pueden configurar a través del módulo de codificación y de configuración de FPU 1834. Por ejemplo, los bloques de cómputo 1825A-1824N se pueden aprovisionar en exceso para contener más unidades de cómputo de las que pueden estar activas de manera máxima en un momento cualquiera, dado el presupuesto de energía de la GPGPU 1820. Sin embargo, el módulo de codificación y de configuración de FPU 1834 puede configurar las unidades de coma flotante dinámicas para controlar ciertos bloques de lógica para operar con una precisión reducida y un gasto de energía reducido. Los requisitos reducidos de precisión y de potencia de cada unidad pueden habilitar que esté en línea un número más grande de unidades, permitiendo que se realice un número más grande de hilos para operaciones de precisión inferior. Por ejemplo, y en una realización, las unidades de lógica que se pueden configurar para realizar operaciones de números enteros de 16 bits se pueden configurar para realizar operaciones de números enteros de 8 bits, reduciendo los requisitos de potencia. En una realización, se pueden realizar operaciones de números enteros de 8 bits duales, aumentando de principio a fin sin aumentar significativamente el gasto de energía. En una realización, múltiples unidades de lógica de precisión media pueden trabajar en paralelo para realizar operaciones de coma flotante de precisión sencilla o de precisión doble. En una realización, las operaciones de precisión superior se pueden realizar a través de múltiples pasadas a través de la unidad de lógica.

En una realización, el módulo de codificación y de configuración de FPU 1834 también puede configurar los métodos de codificación de coma flotante soportados por las unidades de coma flotante. Además de las normas de coma flotante de IEEE 754 para la codificación de precisión media, sencilla y doble para valores de coma flotante, se puede soportar una miríada de formatos de codificación alternativos basándose en el rango dinámico de los datos que se están procesando actualmente. Por ejemplo, basándose en la distribución y/o rango dinámico de un conjunto de datos dado, los datos se pueden cuantificar de forma más precisa de una precisión superior a inferior usando más o menos bits para datos de exponente o de mantisa. En una realización, un formato alternativo soportado especifica un signo de 1 bit, con un exponente de 8 bits y un componente fraccionario de 7 bits. Una realización permite una codificación con un signo de 1 bit, un exponente de 3 bits y un componente fraccionario de 12 bits. En tales realizaciones, diferentes conjuntos de instrucciones soportan diferentes codificaciones de coma flotante, permitiendo que un desarrollador seleccione una codificación basándose en la instrucción especificada en un código de programa. En una realización, se pueden usar diferentes codificaciones de coma flotante cuando se redondean o se muestrean en sentido descendente datos de coma flotante, por ejemplo, desde un valor de coma flotante de 32 bits acumulado a un valor de 16 bits. En una realización, se puede aprovechar la unidad de estadística 1829 para determinar qué codificación de 16 bits es la más adecuada para un bloque dado de datos.

En una realización, el acelerador de aprendizaje automático 1823 incluye adicionalmente la unidad de cuantificación estocástica 1838 para habilitar una cuantificación estocástica para operaciones de aprendizaje automático. La unidad de cuantificación estocástica 1838 se puede usar para habilitar un redondeo estocástico durante operaciones de cuantificación. Una realización habilita un redondeo estocástico usando un generador de números aleatorios, en donde se puede usar un valor fraccionario para determinar una probabilidad de redondeo. Una realización hace uso de la unidad de estadística 1829 para determinar una distribución de probabilidad asociada con el conjunto de datos de salida a partir de una capa dada de una red neuronal. Para cada capa, se puede determinar una densidad de probabilidad de los valores de datos, en donde la densidad de probabilidad es determinada por características estadísticas que incluyen la media, la desviación típica y la varianza de los datos determinados para cada capa de la red neuronal. Usando tales datos, un redondeo estocástico se puede realizar de una manera que no altera la distribución de probabilidad de los datos dentro de cada capa de la red neuronal.

La **Figura 19** ilustra detalles del módulo de instrucción de activación 1832, de acuerdo con una realización. El módulo de instrucción de activación 1832 incluye lógica para secuenciar la ejecución de una multiplicación - suma fusionada combinada y activación en respuesta a una única instrucción. En respuesta a descodificar una función de FMAC/FMADD + Activación por la unidad de extracción y de descodificación de instrucciones 1821 de la Figura 18A, la ejecución de instrucciones se puede despachar al módulo de instrucción de activación 1832 a través del acelerador de aprendizaje automático 1823. El acelerador de aprendizaje automático 1823, tras la recepción de la instrucción, puede usar una unidad de planificación de hilos de multiplicación - suma fusionada/multiplicación - acumulación fusionada 1902 para planificar un conjunto de operaciones de multiplicación - suma fusionada o de multiplicación - acumulación fusionada para computar unidades dentro de los bloques de cómputo 1824A-1824N. En una realización, datos intermedios emitidos desde los bloques de cómputo 1824A-1824N se pueden almacenar en la memoria caché 1836 dentro del acelerador de aprendizaje automático 1823. En una realización, fragmentos de datos intermedios se pueden procesar de una manera por flujo continuo dentro del módulo de instrucción de activación 1832. Los datos intermedios, en una realización, pueden representar una correlación de activación a la que se aplicará la no linealidad de la función de activación. Una seleccionada de las funciones de activación puede ser aplicada por la lógica de función de activación 1904A-1904N. La función de activación se puede seleccionar basándose en la instrucción específica procesada por el módulo de instrucción de activación 1832 o por parámetros suministrados con la instrucción. Se puede dar formato a la instrucción específica basándose en cualquiera de los formatos de instrucción descritos en el presente documento.

Las operaciones de coma flotante, en diversos puntos, incluyen una operación de redondeo. Se usa redondeo en los cálculos de coma flotante debido a que los números de coma flotante tienen un número limitado de dígitos y no pueden representar todos los números reales de forma precisa. Por lo tanto, cuando se asigna a un número la tarea de representar un valor que requiere más dígitos de los que permite el formato de coma flotante seleccionado, se omiten los dígitos sobrantes y el número se redondea al valor más cercano que pueda ser representado por el formato de coma flotante. Los números específicos que se pueden representar dependen del formato de coma flotante seleccionado.

Se pueden realizar diversos enfoques para un redondeo durante cálculos de coma flotante. Las realizaciones descritas en el presente documento incluyen lógica de hardware para realizar un redondeo estocástico para operaciones de aprendizaje automático. En contraposición a otros enfoques de redondeo que redondean al número más cercano o estrictamente hacia arriba y hacia abajo, el enfoque estocástico redondea los números aleatoriamente. Las realizaciones descritas en el presente documento habilitan un redondeo estocástico para la cuantificación de valores de datos para redes neuronales profundas. Se proporciona una unidad de redondeo que habilita un redondeo estocástico de hardware usando uno de múltiples modos de redondeo. Una realización habilita un redondeo estocástico usando un generador de números aleatorios. Se puede usar un valor fraccionario para determinar una probabilidad de redondeo. El número aleatorio se puede comparar con la probabilidad de redondeo para determinar cuál de los valores representables más cercanos redondear durante una cuantificación. Como alternativa, una realización hace uso de una lógica de acumulador/estimador de estadística para determinar una distribución de probabilidad asociada con el conjunto de datos de salida a partir de una capa dada de una red neuronal. Para cada capa, se puede determinar una densidad de probabilidad de la distribución de los valores de datos, en donde la densidad de probabilidad se define por la media, la desviación típica y la varianza de los datos determinados para cada capa de la red neuronal. Usando tales datos, un redondeo estocástico se puede realizar de una manera que no altera la distribución de probabilidad para cada capa de la red neuronal.

La **Figura 20** ilustra la unidad de cuantificación estocástica 1838, de acuerdo con una realización. En una realización, la unidad de cuantificación estocástica 1838 se usa para cuantificar datos de salida sin procesar generados dentro de una capa de una red neuronal en el formato usado por la capa siguiente de la red neuronal. Por ejemplo, las operaciones de cómputo usadas para generar datos de salida se pueden procesar con una precisión superior y los resultados se pueden cuantificar a una precisión inferior antes de proporcionarse como entrada a la capa siguiente. En una realización, la salida 2002B desde una capa n dada se procesa, por ejemplo, a 32 bits y es cuantificada por la unidad de cuantificación 2004 a un tipo de datos de 16 bits. La operación de cuantificación puede hacer uso de un redondeo estocástico, que se puede implementar a través de una unidad de redondeo estocástico 2009. Los valores cuantificados y redondeados se pueden proporcionar entonces a la capa siguiente (Capa N + 1) 2010 de la red neuronal.

En diversas realizaciones, la unidad de cuantificación estocástica 1838 puede realizar un redondeo estocástico a través del uso de un generador de números aleatorios 2006. En la aritmética de coma flotante, el redondeo tiene como objetivo convertir un valor x dado en un valor z con un número especificado de dígitos significativos, en donde z es un múltiplo de un número *m* que depende de la magnitud de *x*. El número *m* es una potencia de la base (habitualmente, 2 o 10) de la representación de coma flotante. El número z es un valor representable que es próximo al valor *x*. Si el valor *x* se redondea hacia arriba o hacia abajo para lograr el valor z se basa en un valor aleatorio que es seleccionado por el generador de números aleatorios 2006. El valor aleatorio que se genera se compara con la porción fraccionaria entre representaciones válidas. La porción fraccionaria se puede usar como la probabilidad de redondear hacia arriba o hacia abajo al valor representable más cercano. La separación entre los valores representables durante una cuantificación depende del formato de codificación para la representación de coma flotante que esté teniendo lugar.

Como un ejemplo, si la cuantificación se ha de redondear a un valor entero y el valor fraccionario es 0,3, la probabilidad de redondear hacia arriba se puede equiparar a un 30 %, mientras que la probabilidad de redondear hacia abajo se puede equiparar a un 70 %. En tal escenario, en donde el generador de números aleatorios 2006 es un generador de números aleatorios auténtico y validado de forma apropiada, la unidad de redondeo estocástico 2009 redondeará hacia arriba o hacia abajo en proporción al valor fraccionario.

Como alternativa, la unidad de redondeo estocástico 2009 puede hacer uso de un acumulador/estimador de estadística 2008 que, en una realización, es una unidad de estadística de datos cercanos 1829 como en la Figura 18A. El acumulador/estimador de estadística 2008 puede analizar una salida a partir de las capas 2002A-2002B previas para determinar la distribución asociada con los datos de red neuronal. La unidad de redondeo estocástico 2009 puede redondear entonces datos durante una cuantificación de manera que los datos cuantificados tienen una distribución similar a la de los datos precuantificados.

La **Figura 21** ilustra el módulo de codificación y de configuración de FPU 1834, de acuerdo con una realización. En una realización, el módulo de codificación y de configuración de GPU 1834 incluye un módulo de configuración de FPU 2102 y un módulo de codificación de FPU 2104. El módulo de configuración de FPU 2102 se puede usar para configurar para realizar unidades de lógica de números enteros de 16 bits para realizar operaciones de números enteros de 8 bits, incluyendo operaciones de números enteros de 8 bits duales. En una realización, múltiples unidades de lógica de precisión media pueden trabajar en paralelo para realizar operaciones de coma flotante de precisión sencilla o de precisión doble. El módulo de codificación de FPU 2104 se puede usar para configurar el formato de codificación de coma flotante específico para su uso dentro de los bloques de cómputo 1824A-1824N durante cómputos de datos. En una realización, el módulo de codificación de FPU 2104 puede configurar uno o más de los bloques de cómputo 1824A-1824N en respuesta a una instrucción que especifica que se van a almacenar datos de entrada o de salida en un formato de coma flotante no convencional. Los bloques de cómputo para ejecutar la instrucción se pueden configurar entonces para interpretar datos en el formato no convencional antes de que se ejecuten operaciones de las instrucciones. En una realización, el módulo de codificación de FPU 2104 ha de configurar uno o más de los bloques de cómputo para usar un formato de codificación de coma flotante que puede almacenar de la manera más eficiente los datos a procesar. Tal determinación se puede realizar en parte basándose en la información de probabilidad y de estadística proporcionada por la unidad de estadística 1829, que puede funcionar como una unidad de cómputo de datos cercanos que está situada dentro de un controlador de memoria 2106 de la memoria local de GPGPU 1828.

La **Figura 22** ilustra la lógica 2200 para procesar una instrucción usando una unidad de cómputo configurable dinámicamente, de acuerdo con una realización. La lógica 2200 puede ser lógica de hardware o de firmware dentro de una GPGPU y/o un multiprocesador de GPGPU como se describe en el presente documento, tal como la unidad de multiprocesador 1400 como en la Figura 14, o la GPGPU 1820 de la Figura 18. La lógica 2200 está configurada para extraer y descodificar una única instrucción para realizar una operación de multiplicación - suma combinada sobre un conjunto de operandos, como se muestra en el bloque 2202. La lógica 2200 puede emitir entonces la única instrucción para su ejecución por una unidad de cómputo para su ejecución por una unidad de cómputo configurable dinámicamente, como se muestra en el bloque 2204. La lógica 2200 puede configurar entonces una o más unidades de lógica de la unidad de cómputo para realizar operaciones con la precisión y el tipo de datos de los operandos, como se muestra en el bloque 2206. La lógica 2200 puede ejecutar entonces la única instrucción en la unidad de cómputo para generar una salida basándose en una operación de multiplicación y de suma, como se muestra en el bloque 2208.

En una realización, la operación de multiplicación y de suma combinada realizada en el bloque 2202 puede ser una operación de coma flotante fusionada que incluye un único redondeo. Por ejemplo, la operación de multiplicación y de suma puede ser una operación de multiplicación - suma fusionada o de multiplicación - acumulación fusionada. La operación de multiplicación y de suma combinada también puede ser una operación de números enteros. La operación de números enteros puede incluir una operación de redondeo entre la multiplicación y la suma. El redondeo se puede realizar insertando un cero en la posición de bit más alta del tipo de datos de números enteros a través de un multiplexor dentro de la unidad de lógica. El multiplexor se puede situar después del multiplicador y antes del sumador dentro de la unidad de lógica.

En una realización, la unidad de lógica configurable dinámicamente del bloque 2204 es una unidad de lógica de coma flotante y de números enteros combinada que es configurable para realizar operaciones de números enteros o de coma flotante. Por ejemplo, la unidad de lógica configurable dinámicamente puede ser una de la unidad de lógica 1600 de la Figura 16, 1700 de la Figura 17A o 1740 de la Figura 17B. La unidad de cómputo puede incluir múltiples instancias diferentes de tales unidades de lógica. En una realización, la unidad de lógica es configurable de una manera por ciclo. En una realización, la unidad de lógica es una primera unidad de lógica configurada para realizar una operación de multiplicación - suma fusionada de un único ciclo usando un multiplicador y un sumador que se comparten entre rutas de datos de coma flotante y de números enteros. En una realización, la unidad de lógica es una segunda unidad de lógica configurada para realizar una operación de multiplicación - acumulación fusionada de dos ciclos que tiene un caudal de un único ciclo. En una realización, la unidad de lógica es una tercera unidad de lógica configurada para realizar una operación de multiplicación - acumulación fusionada de dos ciclos, en donde la tercera lógica incluye un acumulador que tiene el doble de la anchura de bits de los operandos de entrada y de salida. En una realización, el área de pastilla de la tercera unidad de lógica es como máximo un once por ciento más grande que el área de pastilla

de la segunda unidad de lógica.

Las unidades de lógica configurables dinámicamente descritas en el presente documento se pueden configurar para realizar operaciones de números enteros o de coma flotante. En una realización, una o más de las unidades de lógica se pueden configurar para realizar operaciones con múltiples precisiones diferentes. En una realización, las unidades de lógica se pueden usar para realizar operaciones con múltiples precisiones diferentes a través de operaciones de múltiples ciclos. En una realización, se pueden seleccionar diferentes codificaciones de coma flotante, incluyendo el formato de coma flotante de precisión media, el formato de coma flotante de precisión sencilla y el formato de coma flotante de precisión doble de IEEE 754. También se pueden usar formatos de coma flotante no convencionales en los que se usan diferentes asignaciones de bits para el exponente y la mantisa de los valores de coma flotante.

En una realización, la salida basándose en la operación de multiplicación y de suma se puede procesar entonces adicionalmente mediante una función de activación. Por ejemplo, en respuesta a una única instrucción, una operación de FMADD o de FMAC puede ser planificada por una unidad de planificación de hilos de FMADD/FMAC, como se muestra en la Figura 19. La salida de tales operaciones pueden ser datos de correlación de activación que se pueden proporcionar a la lógica de función de activación (por ejemplo, la lógica de función de activación 1904 como en la Figura 19) para generar datos de activación de neuronas.

La **Figura 23A** ilustra la lógica 2300 para ejecutar una instrucción de aprendizaje automático, de acuerdo con una realización. La lógica 2300 puede ser lógica de hardware o de firmware dentro de una GPGPU y/o un multiprocesador de GPGPU como se describe en el presente documento, tal como la unidad de multiprocesador 1400 como en la Figura 14, o la GPGPU 1820 de la Figura 18. La lógica 2300 está configurada para extraer y descodificar una única instrucción para realizar un conjunto de operaciones de aprendizaje automático a través de una unidad de aceleración de aprendizaje automático, como se muestra en el bloque 2302. La unidad de aceleración de aprendizaje automático incluye un elemento del acelerador de aprendizaje automático 1823 descrito en el presente documento, incluyendo el módulo de instrucción de activación 1832, el módulo de codificación y de configuración de FPU 1834 y la unidad de cuantificación estocástica 1838 de la Figura 18B. La lógica 2300 puede emitir entonces la única instrucción para su ejecución por un conjunto de unidades de cómputo configurables dinámicamente, como se muestra en el bloque 2304. La lógica puede configurar entonces el conjunto de unidades de cómputo para realizar el conjunto de operaciones de aprendizaje automático con una precisión superior que las entradas y salidas de las operaciones, como se muestra en el bloque 2306. En una realización, la configuración es realizada por un módulo de configuración de FPU como se describe en el presente documento. El módulo de configuración de FPU puede configurar las unidades de cómputo para realizar, por ejemplo, una operación de convolución sobre datos de matriz de coma flotante de 16 bits usando datos intermedios de 32 bits. Como se muestra en el bloque 2308, la lógica 2300 puede cuantificar entonces los valores intermedios de precisión superior a una precisión inferior antes de la salida a través de una lógica de redondeo estocástico dentro del acelerador de aprendizaje automático. Por ejemplo, datos intermedios de 32 bits se pueden cuantificar a 16 bits para su emisión usando un redondeo estocástico.

La **Figura 23B** ilustra la lógica 2310 para configurar operaciones de coma flotante basándose en una distribución de datos de red neuronal, de acuerdo con una realización. En una realización, la lógica 2300 incluye lógica de hardware y de firmware y unidades de lógica descritas en el presente documento, incluyendo una unidad de cuantificación estocástica 1838 de la Figura 18B y la Figura 20, el módulo de codificación y de configuración de FPU 1834 de la Figura 18B. El acumulador/estimador de estadística 2008 de la figura 20, en una realización, se incluye dentro de la unidad de estadística 1829 de la Figura 18A. La unidad de estadística 1829 puede ser una unidad de cómputo de datos cercanos incluida dentro de un controlador de memoria para la GPGPU, como se muestra en la Figura 21.

Usando la unidad de estadística, la lógica 2310 puede determinar un conjunto de métricas estadísticas para datos de red neuronal almacenados en memoria, como se muestra en el bloque 2312. La lógica 2310 puede determinar entonces, a través de las métricas estadísticas, una distribución para los datos de red neuronal en memoria, como se muestra en el bloque 2314. En una realización, la lógica 2310 puede configurar una codificación de coma flotante para las unidades de cómputo para su uso en la ejecución de un conjunto de operaciones de aprendizaje automático, como se muestra en el bloque 2316. La lógica 2310 puede configurar entonces la lógica de redondeo estocástico dentro del acelerador de aprendizaje automático para redondear basándose en la distribución, como se muestra en el bloque 2318. La lógica de redondeo estocástico se puede configurar para redondear basándose en la distribución de manera que la distribución de probabilidad de los datos de red neuronal cuantificados está más cerca de los datos precuantificados de lo que puede ser posible usando técnicas de redondeo estocástico basadas en un generador de números aleatorios.

Sistema de procesamiento de gráficos ilustrativo adicional

Los detalles de las realizaciones descritas anteriormente se pueden incorporar dentro de sistemas y dispositivos de procesamiento de gráficos descritos a continuación. Los dispositivos y el sistema de procesamiento de gráficos de la **Figura 24** a la **Figura 37** ilustran hardware de procesamiento de gráficos y sistemas alternativos que pueden implementar todas y cada una de las técnicas descritas anteriormente.

Vista global de sistema de procesamiento de gráficos ilustrativa adicional

La **Figura 24** es un diagrama de bloques de un sistema de procesamiento 2400, de acuerdo con una realización. En diversas realizaciones, el sistema 2400 incluye uno o más procesadores 2402 y uno o más procesadores de gráficos 2408, y puede ser un sistema de sobremesa de procesador único, un sistema de estación de trabajo de multiprocesador o un sistema de servidor que tiene un gran número de procesadores 2402 o núcleos de procesador 2407. En una realización, el sistema 2400 es una plataforma de procesamiento incorporada dentro de un circuito integrado de sistema en un chip (SoC) para su uso en dispositivos móviles, de mano o integrados.

Una realización del sistema 2400 puede incluir, o incorporarse dentro de, una plataforma de juegos basada en servidor, una consola de juegos, incluyendo una consola de juegos y de medios, una consola de juegos móvil, una consola de juegos de mano o una consola de juegos en línea. En algunas realizaciones, el sistema 2400 es un teléfono móvil, un teléfono inteligente, un dispositivo informático de tipo tableta o un dispositivo de Internet móvil. El sistema de procesamiento de datos 2400 también puede incluir, acoplarse con o integrarse dentro de un dispositivo ponible, tal como un dispositivo ponible de reloj inteligente, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 2400 es un dispositivo de televisión o de descodificador de salón que tiene uno o más procesadores 2402 y una interfaz gráfica generada por uno o más procesadores de gráficos 2408.

En algunas realizaciones, cada uno de los uno o más procesadores 2402 incluye uno o más núcleos de procesador 2407 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para software de usuario y sistema. En algunas realizaciones, cada uno de los uno o más núcleos de procesador 2407 está configurado para procesar un conjunto de instrucciones 2409 específico. En algunas realizaciones, el conjunto de instrucciones 2409 puede facilitar el cómputo de conjunto de instrucciones complejo (CISC), el cómputo de conjunto de instrucciones reducido (RISC) o el cómputo a través de una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 2407 pueden procesar, cada uno, un conjunto de instrucciones 2409 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 2407 también puede incluir otros dispositivos de procesamiento, tales como un procesador de señales digitales (DSP).

En algunas realizaciones, el procesador 2402 incluye la memoria caché 2404. Dependiendo de la arquitectura, el procesador 2402 puede tener una única caché interna o múltiples niveles de caché interna. En algunas realizaciones, la memoria caché se comparte entre diversos componentes del procesador 2402. En algunas realizaciones, el procesador 2402 también usa una caché externa (por ejemplo, una caché de nivel 3 (L3) o una caché de último nivel (LLC)) (no mostrada), que se puede compartir entre los núcleos de procesador 2407 usando técnicas de coherencia de caché conocidas. Se incluye adicionalmente, en el procesador 2402, un archivo de registro 2406 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de coma flotante, registros de estado y un registro de puntero de instrucción). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos del diseño del procesador 2402.

En algunas realizaciones, el procesador 2402 está acoplado con un bus de procesador 2410 para transmitir señales de comunicación tales como señales de dirección, de datos o de control entre el procesador 2402 y otros componentes en el sistema 2400. En una realización, el sistema 2400 usa una arquitectura de sistema de 'concentrador' ilustrativa, incluyendo un concentrador de controlador de memoria 2416 y un concentrador de controlador de entrada-salida (E/S) 2430. Un concentrador de controlador de memoria 2416 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 2400, mientras que un concentrador de controlador de E/S (ICH) 2430 proporciona conexiones a dispositivos de E/S a través de un bus de E/S local. En una realización, la lógica del concentrador de controlador de memoria 2416 está integrada dentro del procesador.

El dispositivo de memoria 2420 puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria de acceso aleatorio estática (SRAM), un dispositivo de memoria flash, un dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un desempeño adecuado para servir como memoria de proceso. En una realización, el dispositivo de memoria 2420 puede funcionar como memoria de sistema para el sistema 2400, para almacenar los datos 2422 y las instrucciones 2421 para su uso cuando los uno o más procesadores 2402 ejecutan una aplicación o proceso. El concentrador de controlador de memoria 2416 también se acopla con un procesador de gráficos externo 2412 opcional, que se puede comunicar con los uno o más procesadores de gráficos 2408 en los procesadores 2402 para realizar operaciones de gráficos y de medios.

En algunas realizaciones, el ICH 2430 habilita que los periféricos se conecten al dispositivo de memoria 2420 y al procesador 2402 a través de un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, pero sin limitación, un controlador de audio 2446, una interfaz de firmware 2428, un transceptor inalámbrico 2426 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 2424 (por ejemplo, unidad de disco duro, memoria flash, etc.) y un controlador de E/S heredado 2440 para acoplar dispositivos heredados (por ejemplo, de sistema personal 2 (PS/2)) al sistema. Uno o más controladores de bus serie universal (USB) 2442 conectan dispositivos de entrada, tales como combinaciones de teclado y de ratón 2444. Un controlador de red 2434 también se puede acoplar con el ICH 2430. En algunas realizaciones, un controlador de red de alto desempeño (no mostrado) se acopla con el bus de procesador 2410. Se apreciará que el sistema 2400 mostrado es ilustrativo y no limitante, debido a que también se pueden usar otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por

ejemplo, el concentrador de controlador de E/S 2430 se puede integrar dentro de los uno o más procesadores 2402, o el concentrador de controlador de memoria 2416 y el concentrador de controlador de E/S 2430 se pueden integrar en un procesador de gráficos externo discreto, tal como el procesador de gráficos externo 2412.

La **Figura 25** es un diagrama de bloques de una realización de un procesador 2500 que tiene uno o más núcleos de procesador 2502A-2502N, un controlador de memoria integrado 2514 y un procesador de gráficos integrado 2508. Aquellos elementos de la **Figura 25** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa. El procesador 2500 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 2502N representado por los recuadros con línea discontinua. Cada uno de los núcleos de procesador 2502A-2502N incluye una o más unidades de caché internas 2504A-2504N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades almacenadas en caché compartidas 2506.

Las unidades de caché internas 2504A-2504N y las unidades de caché compartidas 2506 representan una jerarquía de memoria caché dentro del procesador 2500. La jerarquía de memoria caché puede incluir al menos un nivel de caché de instrucciones y de datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel medio compartida, tal como una caché de Nivel 2 (L2), de Nivel 3 (L3), de Nivel 4 (L4) o de otros niveles, en donde el nivel más alto de caché antes de la memoria externa se clasifica como LLC. En algunas realizaciones, la lógica de coherencia de caché mantiene la coherencia entre las diversas unidades de caché 2506 y 2504A-2504N.

En algunas realizaciones, el procesador 2500 también puede incluir un conjunto de una o más unidades de controlador de bus 2516 y un núcleo de agente de sistema 2510. Las una o más unidades controladoras de bus 2516 gestionan un conjunto de buses de periféricos, tales como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 2510 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunas realizaciones, el núcleo de agente de sistema 2510 incluye uno o más controladores de memoria integrados 2514 para gestionar el acceso a diversos dispositivos de memoria externos (no mostrados).

En algunas realizaciones, uno o más de los núcleos de procesador 2502A-2502N incluyen soporte para múltiples hilos simultáneos. En tal realización, el núcleo de agente de sistema 2510 incluye componentes para coordinar y hacer funcionar los núcleos 2502A-2502N durante un procesamiento de múltiples hilos. El núcleo de agente de sistema 2510 puede incluir adicionalmente una unidad de control de potencia (PCU), que incluye lógica y componentes para regular el estado de potencia de los núcleos de procesador 2502A-2502N y el procesador de gráficos 2508.

En algunas realizaciones, el procesador 2500 incluye adicionalmente un procesador de gráficos 2508 para ejecutar operaciones de procesamiento de gráficos. En algunas realizaciones, el procesador de gráficos 2508 se acopla con el conjunto de unidades de caché compartidas 2506 y el núcleo de agente de sistema 2510, incluyendo los uno o más controladores de memoria integrados 2514. En algunas realizaciones, un controlador de visualización 2511 está acoplado con el procesador de gráficos 2508 para controlar una salida de procesador de gráficos a una o más pantallas acopladas. En algunas realizaciones, el controlador de visualización 2511 puede ser un módulo separado acoplado con el procesador de gráficos a través de al menos una interconexión, o se puede integrar dentro del procesador de gráficos 2508 o el núcleo de agente de sistema 2510.

En algunas realizaciones, se usa una unidad de interconexión basada en anillo 2512 para acoplar los componentes internos del procesador 2500. Sin embargo, se puede usar una unidad de interconexión alternativa, tal como una interconexión de punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador de gráficos 2508 se acopla con la interconexión en anillo 2512 a través de un enlace de E/S 2513.

El enlace de E/S ilustrativo 2513 representa al menos una de múltiples diversidades de interconexiones de E/S, incluyendo una interconexión de E/S en paquete que facilita la comunicación entre diversos componentes de procesador y un módulo de memoria integrado de alto desempeño 2518, tal como un módulo de eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 2502A-2502N y el procesador de gráficos 2508 usan módulos de memoria integrados 2518 como una caché de último nivel compartida.

En algunas realizaciones, los núcleos de procesador 2502A-2502N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 2502A-2502N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), en donde uno o más de los núcleos de procesador 2502A-2502N 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 una realización, los núcleos de procesador 2502A-2502N son heterogéneos en términos de microarquitectura, en donde uno o más núcleos que tienen un consumo de energía relativamente superior se acoplan con uno o más núcleos de potencia que tienen un consumo de energía inferior. Adicionalmente, el procesador 2500 se puede implementar en uno o más chips o como un circuito integrado de SoC que tiene los componentes ilustrados, además de otros componentes.

La **Figura 26** es un diagrama de bloques de un procesador de gráficos 2600, que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento. En algunas realizaciones, el procesador de gráficos se comunica, a través de una interfaz de E/S correlacionada con memoria, con registros en el procesador de gráficos y con comandos colocados en la memoria de procesador. En algunas realizaciones, el procesador de gráficos 2600 incluye una interfaz de memoria 2614 para acceder a memoria. La interfaz de memoria 2614 puede ser una interfaz a memoria local, una o más cachés internas, una o más cachés externas compartidas y/o a memoria de sistema.

En algunas realizaciones, el procesador de gráficos 2600 también incluye un controlador de visualización 2602 para controlar unos datos de salida de visualización a un dispositivo de visualización 2620. El controlador de visualización 2602 incluye hardware para uno o más planos de superposición para la visualización y composición de múltiples capas de elementos de interfaz de usuario o de vídeo. En algunas realizaciones, el procesador de gráficos 2600 incluye un motor de códec de vídeo 2606 para codificar, descodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluyendo, pero sin limitación, formatos del Grupo de Expertos en Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como de la Sociedad de Ingenieros de Imágenes en Movimiento y de Televisión (SMPTE) 421M/VC-1 y formatos del Grupo Conjunto de Expertos en Fotografía (JPEG) tales como los formatos JPEG y Motion JPEG (MJPEG).

En algunas realizaciones, el procesador de gráficos 2600 incluye un motor de transferencia de imágenes en bloque (BLIT) 2604 para realizar operaciones de rasterizador bidimensionales (2D), incluyendo, por ejemplo, transferencias de bloque de frontera de bits. Sin embargo, en una realización, se realizan operaciones de gráficos 2D usando uno o más componentes del motor de procesamiento de gráficos (GPE) 2610. En algunas realizaciones, el GPE 2610 es un motor de cómputo para realizar operaciones de gráficos, incluyendo operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 310 incluye una canalización de 3D 2612 para realizar operaciones 3D, tales como representar imágenes y escenas tridimensionales usando funciones de procesamiento que actúan sobre formas de primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización de 3D 2612 incluye elementos de función programable y fija que realizan diversas tareas dentro del elemento y/o generan hilos de ejecución en un subsistema de 3D/de medios 2615. Aunque la canalización de 3D 2612 se puede usar para realizar operaciones de medios, una realización del GPE 2610 también incluye una canalización de medios 2616 que se usa específicamente para realizar operaciones de medios, tales como post-procesamiento de vídeo y potenciación de imagen.

En algunas realizaciones, la canalización de medios 2616 incluye unidades de lógica programable o de función fija para realizar una o más operaciones de medios especializadas, tales como aceleración de descodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar o en nombre del motor de códec de vídeo 2606. En algunas realizaciones, la canalización de medios 2616 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema de 3D/de medios 2615. Los hilos generados realizan cómputos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema de 3D/de medios 2615.

En algunas realizaciones, el subsistema de 3D/de medios 2615 incluye lógica para ejecutar hilos generados por la canalización de 3D 2612 y la canalización de medios 2616. En una realización, las canalizaciones envían solicitudes de ejecución de hilos al subsistema de 3D/de medios 2615, incluyendo lógica de despacho de hilos para arbitrar y despachar las diversas solicitudes a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos de 3D y de medios. En algunas realizaciones, el subsistema de 3D/de medios 2615 incluye una o más cachés internas para datos e instrucciones de hilo. En algunas realizaciones, el subsistema también incluye memoria compartida, incluyendo registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Motor de procesamiento de gráficos ilustrativo adicional

La **Figura 27** es un diagrama de bloques de un motor de procesamiento de gráficos 2710 de un procesador de gráficos de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento de gráficos (GPE) 2710 es una versión del GPE 2610 mostrado en la Figura 26. Los elementos de la **Figura 27** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa. Por ejemplo, se ilustran la canalización de 3D 2612 y la canalización de medios 2616 de la Figura 26. La canalización de medios 2616 es opcional en algunas realizaciones del GPE 2710 y puede no incluirse explícitamente dentro del GPE 2710. Por ejemplo, y en al menos una realización, un procesador de medios y/o de imágenes separado se acopla al GPE 2710.

En algunas realizaciones, el GPE 2710 se acopla con o incluye un transmisor por flujo continuo de comandos 2703, que proporciona un flujo de comandos a la canalización de 3D 2612 y/o a las canalizaciones de medios 2616. En algunas realizaciones, el transmisor por flujo continuo de comandos 2703 está acoplado con memoria, que puede ser

memoria de sistema, o una o más de memoria caché interna y memoria caché compartida. En algunas realizaciones, el transmisor por flujo continuo de comandos 2703 recibe comandos desde la memoria y envía los comandos a la canalización de 3D 2612 y/o a la canalización de medios 2616. Los comandos son directivas extraídas de una memoria intermedia en anillo, que almacena comandos para la canalización de 3D 2612 y la canalización de medios 2616. En una realización, la memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización de 3D 2612 también pueden incluir referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértice y de geometría para la canalización de 3D 2612 y/o datos de imagen y objetos de memoria para la canalización de medios 2616. La canalización de 3D 2612 y la canalización de medios 2616 procesan los comandos y datos realizando operaciones a través de lógica dentro de las canalizaciones respectivas o despachando uno o más hilos de ejecución a una matriz de núcleo de gráficos 2714.

En diversas realizaciones, la canalización de 3D 2612 puede ejecutar uno o más programas de sombreado, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cómputo u otros programas de sombreado, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleo de gráficos 2714. La matriz de núcleo de gráficos 2714 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución de múltiples propósitos (por ejemplo, unidades de ejecución) dentro de la matriz de núcleo de gráficos 2714 incluye soporte para diversos lenguajes de sombreador de API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunas realizaciones, la matriz de núcleo de gráficos 2714 también incluye lógica de ejecución para realizar funciones de medios, tales como procesamiento de vídeo y/o de imagen. En una realización, las unidades de ejecución incluyen adicionalmente lógica de propósito general que es programable para realizar operaciones computacionales de propósito general paralelas, además de operaciones de procesamiento de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o junto con lógica de propósito general dentro del núcleo o núcleos de procesador 2407 de la **Figura 24** o del núcleo 2502A-2502N, como en la **Figura 25**.

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleo de gráficos 2714 pueden emitir datos a memoria en una memoria intermedia de retorno unificada (URB) 2718. La URB 2718 puede almacenar datos para múltiples hilos. En algunas realizaciones, la URB 2718 se puede usar para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleo de gráficos 2714. En algunas realizaciones, la URB 2718 se puede usar adicionalmente para la sincronización entre hilos en la matriz de núcleo de gráficos y la lógica de función fija dentro de la lógica de funciones compartidas 2720.

En algunas realizaciones, la matriz de núcleos de gráficos 2714 es ajustable a escala, de manera que la matriz incluye un número variable de núcleos de gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y en el nivel de desempeño del GPE 2710. En una realización, los recursos de ejecución son dinámicamente ajustables a escala, de manera que los recursos de ejecución se pueden habilitar o deshabilitar según sea necesario.

La matriz de núcleo de gráficos 2714 se acopla con la lógica de funciones compartidas 2720 que incluye múltiples recursos que se comparten entre los núcleos de gráficos en la matriz de núcleo de gráficos. Las funciones compartidas dentro de la lógica de funciones compartidas 2720 son unidades de lógica de hardware que proporcionan una funcionalidad complementaria especializada a la matriz de núcleo de gráficos 2714. En diversas realizaciones, la lógica de funciones compartidas 2720 incluye, pero sin limitación, la lógica del muestreador 2721, del cálculo matemático 2722 y de la comunicación entre hilos (ITC) 2723. Adicionalmente, algunas realizaciones implementan una o más cachés 2725 dentro de la lógica de funciones compartidas 2720. Se implementa una función compartida donde la demanda de una función especializada dada es insuficiente para su inclusión dentro de la matriz de núcleo de gráficos 2714. En su lugar, una única instanciación de esa función especializada se implementa como una entidad autónoma en la lógica de funciones compartidas 2720 y se comparte entre los recursos de ejecución dentro de la matriz de núcleo de gráficos 2714. El conjunto preciso de funciones que se comparten entre la matriz de núcleo de gráficos 2714 y se incluyen dentro de la matriz de núcleo de gráficos 2714 varía entre realizaciones.

La **Figura 28** es un diagrama de bloques de otra realización de un procesador de gráficos 2800. Los elementos de la **Figura 28** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

En algunas realizaciones, el procesador de gráficos 2800 incluye una interconexión en anillo 2802, un extremo frontal de canalización 2804, un motor de medios 2837 y unos núcleos de gráficos 2880A-2880N. En algunas realizaciones, la interconexión en anillo 2802 acopla el procesador de gráficos a otras unidades de procesamiento, incluyendo otros procesadores de gráficos o uno o más núcleos de procesador de propósito general. En algunas realizaciones, el procesador de gráficos es uno de muchos procesadores integrados dentro de un sistema de procesamiento de múltiples núcleos.

En algunas realizaciones, el procesador de gráficos 2800 recibe lotes de comandos a través de la interconexión en

anillo 2802. Los comandos entrantes son interpretados por un transmisor por flujo continuo de comandos 2803 en el extremo frontal de canalización 2804. En algunas realizaciones, el procesador de gráficos 2800 incluye lógica de ejecución ajustable a escala para realizar un procesamiento de geometría 3D y un procesamiento de medios a través del núcleo o núcleos de gráficos 2880A-2880N. Para los comandos de procesamiento de geometría 3D, el transmisor por flujo continuo de comandos 2803 suministra comandos a la canalización de geometría 2836. Para al menos algunos comandos de procesamiento de medios, el transmisor por flujo continuo de comandos 2803 suministra los comandos a un extremo frontal de vídeo 2834, que se acopla con un motor de medios 2837. En algunas realizaciones, el motor de medios 2837 incluye un motor de calidad de vídeo (VQE) 2830 para el post-procesamiento de vídeo y de imagen y un motor de codificación/descodificación (MFX) de múltiples formatos 2833 para proporcionar una codificación y descodificación de datos de medios acelerada por hardware. En algunas realizaciones, la canalización de geometría 2836 y el motor de medios 2837 generan, cada uno, hilos de ejecución para los recursos de ejecución de hilos proporcionados por al menos un núcleo de gráficos 2880A.

En algunas realizaciones, el procesador de gráficos 2800 incluye recursos de ejecución de hilos ajustables a escala que cuentan con los núcleos modulares 2880A-2880N (denominados, en ocasiones, cortes de núcleo), teniendo cada uno múltiples subnúcleos 2850A-550N, 2860A-2860N (denominados, en ocasiones, subcortes de núcleo). En algunas realizaciones, el procesador de gráficos 2800 puede tener cualquier número de núcleos de gráficos 2880A a 2880N. En algunas realizaciones, el procesador de gráficos 2800 incluye un núcleo de gráficos 2880A que tiene al menos un primer subnúcleo 2850A y un segundo subnúcleo 2860A. En otras realizaciones, el procesador de gráficos es un procesador de baja potencia con un único subnúcleo (por ejemplo, 2850A). En algunas realizaciones, el procesador de gráficos 2800 incluye múltiples núcleos de gráficos 2880A-2880N, incluyendo cada uno un conjunto de primeros subnúcleos 2850A-2850N y un conjunto de segundos subnúcleos 2860A-2860N. Cada subnúcleo en el conjunto de primeros subnúcleos 2850A-2850N incluye al menos un primer conjunto de unidades de ejecución 2852A-2852N y muestreadores de medios/texturas 2854A-2854N. Cada subnúcleo en el conjunto de segundos subnúcleos 2860A-2860N incluye al menos un segundo conjunto de unidades de ejecución 2862A-2862N y muestreadores 2864A-2864N. En algunas realizaciones, cada subnúcleo 2850A-2850N, 2860A-2860N comparte un conjunto de recursos compartidos 2870A-2870N. En algunas realizaciones, los recursos compartidos incluyen memoria caché compartida y lógica de operación de píxel. También se pueden incluir otros recursos compartidos en las diversas realizaciones del procesador de gráficos.

Unidades de ejecución ilustrativas adicionales

La **Figura 29** ilustra la lógica de ejecución de hilos 2900 que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones de un GPE. Los elementos de la **Figura 29** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

En algunas realizaciones, la lógica de ejecución de hilos 2900 incluye un procesador de sombreado 2902, un despachador de hilos 2904, una caché de instrucciones 2906, una matriz de unidades de ejecución ajustable a escala que incluye una pluralidad de unidades de ejecución 2908A-2908N, un muestreador 2910, una caché de datos 2912 y un puerto de datos 2914. En una realización, la matriz de unidades de ejecución ajustable a escala puede realizar un ajuste a escala dinámico habilitando o deshabilitando una o más unidades de ejecución (por ejemplo, cualquiera de las unidades de ejecución 2908A, 2908B, 2908C, 2908D a 2908N-1 y 2908N) basándose en los requisitos computacionales de una carga de trabajo. En una realización, los componentes incluidos están interconectados a través de un tejido de interconexión que se enlaza con cada uno de los componentes. En algunas realizaciones, la lógica de ejecución de hilos 2900 incluye una o más conexiones a memoria, tales como memoria de sistema o memoria caché, a través de una o más de la caché de instrucciones 2906, el puerto de datos 2914, el muestreador 2910 y las unidades de ejecución 2908A-2908N. En algunas realizaciones, cada unidad de ejecución (por ejemplo, 2908A) es una unidad computacional de propósito general programable autónoma que es capaz de ejecutar múltiples hilos de hardware simultáneos mientras se procesan múltiples elementos de datos en paralelo para cada hilo. En diversas realizaciones, la matriz de unidades de ejecución 2908A-2908N es ajustable a escala para incluir cualquier número de unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 2908A-2908N se usan principalmente para ejecutar programas de sombreado. Un procesador de sombreado 2902 puede procesar los diversos programas de sombreado y despachar hilos de ejecución asociados con los programas de sombreado a través de un despachador de hilos 2904. En una realización, el despachador de hilos incluye lógica para arbitrar solicitudes de iniciación de hilo desde las canalizaciones de gráficos y de medios e instanciar los hilos solicitados en una o más unidades de ejecución en las unidades de ejecución 2908A-2908N. Por ejemplo, la canalización de geometría (por ejemplo, 2836 de la **Figura 28**) puede despachar sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilos 2900 (la **Figura 29**) para su procesamiento. En algunas realizaciones, el despachador de hilos 2904 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas de sombreado en ejecución.

En algunas realizaciones, las unidades de ejecución 2908A-2908N soportan un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreador de gráficos 3D convencionales, de manera que programas

de sombreado desde bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución soportan un procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), un procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y un procesamiento de propósito general (por ejemplo, sombreadores de cómputo y de medios). Cada una de las unidades de ejecución 2908A-2908N es capaz de múltiples emisiones de una ejecución de una única instrucción - múltiples datos (SIMD), y un funcionamiento de múltiples hilos habilita un entorno de ejecución eficiente frente a accesos de memoria de latencia superior. Cada hilo de hardware dentro de cada unidad de ejecución tiene un archivo de registro de ancho de banda alto dedicado y un estado de hilo independiente asociado. La ejecución es de múltiples emisiones por reloj a canalizaciones capaces de realizar operaciones de números enteros, de coma flotante de precisión sencilla y doble, capacidad de bifurcación de SIMD, operaciones lógicas, operaciones trascendentales y otras operaciones misceláneas. Mientras se esperan datos desde memoria o una de las funciones compartidas, una lógica de dependencia dentro de las unidades de ejecución 2908A-2908N hace que un hilo en espera pase a estar inactivo hasta que se hayan devuelto los datos solicitados. Mientras el hilo en espera está inactivo, se pueden dedicar recursos de hardware a procesar otros hilos. Por ejemplo, durante un retardo asociado con una operación de sombreador de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, un sombreador de fragmentos u otro tipo de programa de sombreado, incluyendo un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 2908A-2908N opera sobre matrices de elementos de datos. El número de elementos de datos es el "tamaño de ejecución" o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso, enmascaramiento y control de flujo de elementos de datos dentro de las instrucciones. El número de canales puede ser independiente del número de Unidades Aritméticas Lógicas (ALU) o Unidades de Coma Flotante (FPU) físicas para un procesador de gráficos particular. En algunas realizaciones, las unidades de ejecución 2908A-2908N soportan tipos de datos de números enteros y de coma flotante.

El conjunto de instrucciones de unidad de ejecución incluye instrucciones de 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 sobre un vector de 256 bits de ancho, los 256 bits del vector se almacenan en un registro y la unidad de ejecución opera sobre el vector como cuatro elementos de datos empaquetados de 64 bits separados (elementos de datos de tamaño de palabra cuádruple (QW)), ocho elementos de datos empaquetados de 32 bits separados (elementos de datos de tamaño de palabra doble (DW)), dieciséis elementos de datos empaquetados de 16 bits separados (elementos de datos de tamaño de palabra (W)) o treinta y dos elementos de datos de 8 bits separados (elementos de datos de tamaño de byte (B)). Sin embargo, son posibles diferentes anchuras de vector y tamaños de registro.

Una o más cachés de instrucciones internas (por ejemplo, 2906) se incluyen en la lógica de ejecución de hilos 2900 para almacenar en caché instrucciones de hilo para las unidades de ejecución. En algunas realizaciones, se incluyen una o más cachés de datos (por ejemplo, 2912) para almacenar en caché datos de hilo durante la ejecución de hilo. En algunas realizaciones, se incluye un muestreador 2910 para proporcionar un muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunas realizaciones, el muestreador 2910 incluye una funcionalidad de muestreo de textura o de medios especializada para procesar datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las canalizaciones de gráficos y de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilos 2900 a través de lógica de generación y de despacho de hilos. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos para dar datos de píxel, se invoca lógica de procesador de píxeles (por ejemplo, lógica de sombreado de píxeles, lógica de sombreado de fragmentos, etc.) dentro del procesador de sombreado 2902 para computar adicionalmente información de salida y hacer que se escriban resultados para emitir superficies (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de estarcido, etc.). En algunas realizaciones, un sombreador de píxeles o un sombreador de fragmentos calcula los valores de los diversos atributos de vértice que se van a interpolar a lo largo del objeto rasterizado. En algunas realizaciones, una lógica de procesador de píxeles dentro del procesador de sombreado 2902 ejecuta entonces un programa de sombreado de píxeles o de fragmentos suministrado por interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreado, el procesador de sombreado 2902 despacha hilos a una unidad de ejecución (por ejemplo, 2908A) a través del despachador de hilos 2904. En algunas realizaciones, el sombreador de píxeles 2902 usa una lógica de muestreo de textura en el muestreador 2910 para acceder a datos de textura en correlaciones de textura almacenadas en memoria. Operaciones aritméticas sobre los datos de textura y los datos de geometría de entrada computan datos de color de píxel para cada fragmento geométrico, o descartan el procesamiento adicional de uno o más píxeles.

En algunas realizaciones, el puerto de datos 2914 proporciona un mecanismo de acceso de memoria para que la lógica de ejecución de hilos 2900 emita datos procesados a memoria para su procesamiento en una canalización de salida de procesador de gráficos. En algunas realizaciones, el puerto de datos 2914 incluye o se acopla a una o más memorias caché (por ejemplo, la caché de datos 2912) para almacenar en caché datos para un acceso de memoria a través del puerto de datos.

La **Figura 30** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos 3000 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución de procesador de gráficos soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los cuadros con línea continua ilustran los componentes que se incluyen, en general, en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de las instrucciones. En algunas realizaciones, el formato de instrucción 3000 descrito e ilustrado son macro-instrucciones, en el sentido de que las mismas son instrucciones suministradas a la unidad de ejecución, en contraposición a micro-operaciones resultantes de la descodificación de instrucciones una vez que se ha procesado la instrucción.

En algunas realizaciones, las unidades de ejecución de procesador de gráficos soportan de manera nativa instrucciones en un formato de instrucción de 128 bits 3010. Un formato de instrucción compactado de 64 bits 3030 está disponible para algunas instrucciones basándose en la instrucción, las opciones de instrucción y el número de operandos seleccionados. El formato de instrucción de 128 bits nativo 710 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de 64 bits 3030. Las instrucciones nativas disponibles en el formato de 64 bits 3030 varían según la realización. En algunas realizaciones, la instrucción se compacta en parte usando un conjunto de valores de índice en un campo de índice 3013. El hardware de unidad de ejecución consulta un conjunto de tablas de compactación basándose en los valores de índice y usa las salidas de tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 3010.

Para cada formato, el código de operación de instrucción 3012 define la operación que ha de realizar la unidad de ejecución. 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 suma, la unidad de ejecución realiza una operación de suma simultánea a lo largo de cada canal de color que representa un elemento de textura o un elemento de imagen. Por defecto, la unidad de ejecución ejecuta cada instrucción a lo largo de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 3014 habilita el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicación) y el orden de canal de datos (por ejemplo, referenciación). Para instrucciones en el formato de instrucción de 128 bits 3010, un campo de tamaño de ejecución 3016 limita el número de canales de datos que se ejecutarán en paralelo. En algunas realizaciones, el campo de tamaño de ejecución 3016 no está disponible para su uso en el formato de instrucción compacto de 64 bits 3030.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos, incluyendo dos operandos de origen, src0 3020, src1 3022 y un destino 3018. En algunas realizaciones, las unidades de ejecución soportan instrucciones de destino dual, en donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 3024), en donde el código de operación de instrucción 3012 determina el número de operandos de origen. El último operando de origen de una instrucción puede ser un valor inmediato (por ejemplo, codificado de manera rígida) pasado con la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 3010 incluye un campo de modo de acceso/dirección 3026 que especifica, por ejemplo, si se usa el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se usa el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos es proporcionada directamente por bits en la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 3010 incluye un campo de modo de dirección/acceso 3026, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En una realización, el modo de acceso se usa para definir una alineación de acceso de datos para la instrucción. Algunas realizaciones soportan modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, en donde la alineación de bytes del modo de acceso determina la alineación de acceso de los operandos de instrucción. Por ejemplo, cuando está en un primer modo, la instrucción puede usar un direccionamiento alineado por byte para los operandos de origen y de destino y, cuando está en un segundo modo, la instrucción puede usar un direccionamiento alineado por 16 bytes para todos los operandos de origen y de destino.

En una realización, la porción de modo de dirección del campo de modo de acceso/dirección 3026 determina si la instrucción va a usar un direccionamiento directo o indirecto. Cuando se usa el modo de direccionamiento de registro directo, bits en la instrucción proporcionan directamente la dirección de registro de uno o más operandos. Cuando se usa un modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos se puede computar basándose en un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

En algunas realizaciones, las instrucciones se agrupan basándose en los campos de bits del código de operación 3012 para simplificar la descodificación de código de operación 3040. 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. La agrupación de código de operación precisa mostrada es simplemente un ejemplo. En algunas realizaciones, un grupo de código de operación de movimiento y de lógica 3042 incluye instrucciones de movimiento y de lógica de datos (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y de lógica 3042 comparte los cinco bits más significativos (MSB), en donde las instrucciones de movimiento (mov) están en forma de 0000xxxxb y las instrucciones de lógica están en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 3044 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas

3046 incluye una mezcla de instrucciones, incluyendo instrucciones de sincronización (por ejemplo, espera, envío) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo matemático paralelo 3048 incluye instrucciones aritméticas a nivel de componente (por ejemplo, suma, multiplicación (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculo matemático paralelo 3048 realiza las operaciones aritméticas en paralelo a lo largo de canales de datos. El grupo de cálculo matemático vectorial 3050 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cálculo matemático vectorial realiza aritmética tal como cálculos de producto escalar sobre operandos de vectores.

Canalización de gráficos ilustrativa adicional

La **Figura 31** es un diagrama de bloques de otra realización de un procesador de gráficos 3100. Los elementos de la **Figura 31** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

En algunas realizaciones, el procesador de gráficos 3100 incluye una canalización de gráficos 3120, una canalización de medios 3130, un motor de visualización 3140, una lógica de ejecución de hilos 3150 y una canalización de salida de representación 3170. En algunas realizaciones, el procesador de gráficos 3100 es un procesador de gráficos dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos es controlado por escrituras de registro en uno o más registros de control (no mostrados) o a través de comandos emitidos al procesador de gráficos 3100 a través de una interconexión en anillo 3102. En algunas realizaciones, la interconexión en anillo 3102 acopla el procesador de gráficos 3100 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de propósito general. Los comandos desde la interconexión en anillo 3102 son interpretados por un transmisor por flujo continuo de comandos 3103, que suministra instrucciones a componentes individuales de la canalización de gráficos 3120 o la canalización de medios 3130.

En algunas realizaciones, el transmisor por flujo continuo de comandos 3103 dirige el funcionamiento de un extractor de vértices 3105 que lee datos de vértice desde memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor por flujo continuo de comandos 3103. En algunas realizaciones, el extractor de vértices 3105 proporciona datos de vértice a un sombreador de vértices 3107, que realiza operaciones de transformación y de iluminación de espacio de coordenadas en cada vértice. En algunas realizaciones, el extractor de vértices 3105 y el sombreador de vértices 3107 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a las unidades de ejecución 3152A-3152B a través de un despachador de hilos 3131.

En algunas realizaciones, las unidades de ejecución 3152A-3152B son una matriz de procesadores de vectores que tienen un conjunto de instrucciones para realizar operaciones de gráficos y de medios. En algunas realizaciones, las unidades de ejecución 3152A-3152B tienen una caché de L1 3151 anexada que es específica para cada matriz o que se comparte entre las matrices. La caché se puede configurar como una caché de datos, una caché de instrucciones o una única caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.

En algunas realizaciones, la canalización de gráficos 3120 incluye componentes de teselado para realizar un teselado acelerado por hardware de objetos 3D. En algunas realizaciones, un sombreador de casco programable 811 configura las operaciones de teselado. Un sombreador de dominio programable 817 proporciona una evaluación de extremo trasero de la salida de teselado. Un teselador 3113 opera en la dirección del sombreador de casco 3111 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados basándose en un modelo geométrico grueso que se proporciona como entrada a la canalización de gráficos 3120. En algunas realizaciones, si no se usa el teselado, se pueden sortear los componentes de teselado (por ejemplo, el sombreador de casco 3111, el teselador 3113 y el sombreador de dominio 3117).

En algunas realizaciones, objetos geométricos completos pueden ser procesados por un sombreador de geometría 3119 a través de uno o más hilos despachados a las unidades de ejecución 3152A-3152B, o puede avanzar directamente al recortador 3129. En algunas realizaciones, el sombreador de geometría opera sobre objetos geométricos enteros, en lugar de vértices o parches de vértices como en fases previas de la canalización de gráficos. Si la teselación está deshabilitada, el sombreador de geometría 3119 recibe una entrada desde el sombreador de vértices 3107. En algunas realizaciones, el sombreador de geometría 3119 se puede programar mediante un programa de sombreado de geometría para realizar un teselado de geometría si las unidades de teselado están deshabilitadas.

Antes de la rasterización, un recortador 3129 procesa datos de vértice. El recortador 3129 puede ser un recortador de función fija o un recortador programable que tiene funciones de recorte y de sombreador de geometría. En algunas realizaciones, un componente de prueba de rasterizador y de profundidad 3173 en la canalización de salida de representación 3170 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunas realizaciones, la lógica de sombreado de píxeles se incluye en la lógica de ejecución de hilos 3150. En algunas realizaciones, una aplicación puede sortear el componente de prueba de rasterizador y de profundidad 3173 y acceder a datos de vértice sin rasterizar a través de una unidad de salida de flujo 3123.

El procesador de gráficos 3100 tiene un bus de interconexión, un tejido de interconexión o algún otro mecanismo de interconexión que permite el paso de datos y de mensajes entre los componentes principales del procesador. En algunas realizaciones, las unidades de ejecución 3152A-3152B y la caché o cachés 3151 asociadas, el muestreador de textura y de medios 3154 y la caché de textura/muestreador 3158 se interconectan a través de un puerto de datos 3156 para realizar un acceso de memoria y comunicarse con componentes de canalización de salida de representación del procesador. En algunas realizaciones, el muestreador 3154, las cachés 3151, 3158 y las unidades de ejecución 3152A-3152B tienen, cada uno, rutas de acceso de memoria separadas.

En algunas realizaciones, la canalización de salida de representación 3170 contiene un componente de prueba de rasterizador y de profundidad 3173 que convierte objetos basados en vértices en una representación basada en píxeles asociada. En algunas realizaciones, la lógica de rasterizador incluye una unidad generadora de ventanas/enmascaradora para realizar una rasterización de líneas y de triángulos de función fija. Una caché de representación 3178 y una caché de profundidad 3179 asociadas también están disponibles en algunas realizaciones. Un componente de operaciones de píxel 3177 realiza operaciones basadas en píxeles sobre los datos, aunque, en algunas instancias, las operaciones de píxel asociadas con operaciones 2D (por ejemplo, transferencias de imagen de bloque de bits con mezcla) son realizadas por el motor 2D 3141, o son sustituidas en el momento de la visualización por el controlador de visualización 3143 usando planos de visualización de superposición. En algunas realizaciones, está disponible una caché de L3 compartida 3175 para todos los componentes de gráficos, permitiendo la compartición de datos sin el uso de memoria de sistema principal.

En algunas realizaciones, la canalización de medios de procesador de gráficos 3130 incluye un motor de medios 3137 y un extremo frontal de vídeo 3134. En algunas realizaciones, el extremo frontal de vídeo 3134 recibe comandos de canalización desde el transmisor por flujo continuo de comandos 3103. En algunas realizaciones, la canalización de medios 3130 incluye un transmisor por flujo continuo de comandos separado. En algunas realizaciones, el extremo frontal de vídeo 3134 procesa comandos de medios antes de enviar el comando al motor de medios 3137. En algunas realizaciones, el motor de medios 3137 incluye una funcionalidad de generación de hilos para generar hilos para despacharlos a la lógica de ejecución de hilos 3150 a través del despachador de hilos 3131.

En algunas realizaciones, el procesador de gráficos 3100 incluye un motor de visualización 3140. En algunas realizaciones, el motor de visualización 3140 es externo al procesador 3100 y se acopla con el procesador de gráficos a través de la interconexión en anillo 3102, o algún otro bus o tejido de interconexión. En algunas realizaciones, el motor de visualización 3140 incluye un motor 2D 3141 y un controlador de visualización 3143. En algunas realizaciones, el motor de visualización 3140 contiene una lógica de propósito especial capaz de funcionar independientemente de la canalización de 3D. En algunas realizaciones, el controlador de visualización 3143 se acopla con un dispositivo de visualización (no mostrado), que puede ser un dispositivo de visualización integrado en sistema, como en un ordenador portátil, o un dispositivo de visualización externo anexado a través de un conector de dispositivo de visualización.

En algunas realizaciones, la canalización de gráficos 3120 y la canalización de medios 3130 se pueden configurar para realizar operaciones basándose en múltiples interfaces de programación de gráficos y de medios y no son específicas de ninguna interfaz de programación de aplicaciones (API) concreta. En algunas realizaciones, software de controlador para el procesador de gráficos traduce llamadas de API que son específicas de una biblioteca de medios o de gráficos particular a comandos que pueden ser procesados por el procesador de gráficos. En algunas realizaciones, se proporciona soporte para la Biblioteca de Gráficos Abierta (OpenGL), Lenguaje Informático Abierto (OpenCL) y/o API de gráficos y de cómputo Vulkan, todas ellas del grupo Khronos. En algunas realizaciones, también se puede proporcionar soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, se puede soportar una combinación de estas bibliotecas. También se puede proporcionar soporte para la Biblioteca de Visión por Ordenador de Código Abierto (OpenCV). También se soportaría una API futura con una canalización de 3D compatible si se puede hacer una correlación desde la canalización de la API futura a la canalización del procesador de gráficos.

Programación de canalización de gráficos

La **Figura 32A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos 3200 de acuerdo con algunas realizaciones. La **Figura 32B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos 3210 de acuerdo con una realización. Los cuadros con línea continua en la **Figura 32A** ilustran los componentes que se incluyen, en general, en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de los comandos de gráficos. El formato de comando de procesador de gráficos 3200 ilustrativo de la **Figura 32A** incluye campos de datos para identificar un cliente objetivo 3202 del comando, un código de operación (código de op.) de comando 3204 y los datos 3206 relevantes para el comando. También se incluyen un subcódigo de operación 3205 y un tamaño de comando 3208 en algunos comandos.

En algunas realizaciones, el cliente 3202 especifica la unidad de cliente del dispositivo de gráficos que procesa los datos de comando. En algunas realizaciones, un analizador de comandos de procesador de gráficos examina el campo de cliente de cada comando para acondicionar el procesamiento adicional del comando y encaminar los datos de

comando a la unidad de cliente apropiada. En algunas realizaciones, las unidades de cliente de procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de representación, una unidad de 2D, una unidad de 3D y una unidad de medios. Cada unidad de cliente tiene una canalización de procesamiento correspondiente que procesa los comandos. Una vez que el comando ha sido recibido por la unidad de cliente, la unidad de cliente lee el código de operación 3204 y, si está presente, el subcódigo de operación 3205 para determinar la operación a realizar. La unidad de cliente realiza el comando usando información en el campo de datos 3206. Para algunos comandos, se espera que un tamaño de comando explícito 3208 especifique el tamaño del comando. En algunas realizaciones, el analizador de comandos determina automáticamente el tamaño de al menos algunos de los comandos basándose en el código de operación de comando. En algunas realizaciones, los comandos se alinean a través de múltiplos de una palabra doble.

El diagrama de flujo en la **Figura 32B** muestra una secuencia de comandos de procesador de gráficos 3210 ilustrativa. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que cuenta con una realización de un procesador de gráficos usa una versión de la secuencia de comandos mostrada para establecer, ejecutar y terminar un conjunto de operaciones de gráficos. Se muestra y se describe una secuencia de comandos de muestra solo con fines de ejemplo, debido a que las realizaciones no se limitan a estos comandos específicos o a esta secuencia de comandos. Además, los comandos se pueden emitir como un lote de comandos en una secuencia de comandos, de manera que el procesador de gráficos procesará la secuencia de comandos de manera al menos parcialmente concurrente.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 3210 puede comenzar con un comando de vaciado de canalización 3212 para hacer que cualquier canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunas realizaciones, la canalización de 3D 3222 y la canalización de medios 3224 no funcionan de manera concurrente. El vaciado de canalización se realiza para hacer que la canalización de gráficos activa complete cualquier comando pendiente. En respuesta a un vaciado de canalización, el analizador de comandos para el procesador de gráficos pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. Opcionalmente, cualquier dato en la caché de representación que se marque como 'sucio' se puede vaciar a memoria. En algunas realizaciones, el comando de vaciado de canalización 3212 se puede usar para la sincronización de canalización o antes de poner el procesador de gráficos en un estado de baja potencia.

En algunas realizaciones, se usa un comando de selección de canalización 3213 cuando una secuencia de comandos requiere que el procesador de gráficos conmute explícitamente entre canalizaciones. En algunas realizaciones, se requiere un comando de selección de canalización 3213 solo una vez dentro de un contexto de ejecución antes de emitir comandos de canalización, a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunas realizaciones, se requiere un comando de vaciado de canalización 3212 inmediatamente antes de una conmutación de canalización a través del comando de selección de canalización 3213.

En algunas realizaciones, un comando de control de canalización 3214 configura una canalización de gráficos para su funcionamiento y se usa para programar la canalización de 3D 3222 y la canalización de medios 3224. En algunas realizaciones, el comando de control de canalización 3214 configura el estado de canalización para la canalización activa. En una realización, el comando de control de canalización 3214 se usa para la sincronización de canalización y para borrar datos de una o más memorias caché dentro de la canalización activa antes de procesar un lote de comandos.

En algunas realizaciones, se usan comandos específicos del estado de memoria intermedia de retorno 3216 para configurar un conjunto de memorias intermedias de retorno para que las canalizaciones respectivas escriban datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas realizaciones, el procesador de gráficos también usa una o más memorias intermedias de retorno para almacenar datos de salida y para realizar una comunicación a través de hilos. En algunas realizaciones, el estado de memoria intermedia de retorno 3216 incluye seleccionar el tamaño y el número de memorias intermedias de retorno a usar para un conjunto de operaciones de canalización.

Los comandos restantes en la secuencia de comandos difieren basándose en la canalización activa para las operaciones. Basándose en una determinación de canalización 3220, la secuencia de comandos se adapta a la canalización de 3D 3222 comenzando con el estado de canalización de 3D 3230, o a la canalización de medios 3224 comenzando en el estado de canalización de medios 3240.

Los comandos para configurar el estado de canalización de 3D 3230 incluyen comandos de ajuste de estado de 3D para estado de memoria intermedia de vértice, estado de elemento de vértice, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que han de configurarse antes de que se procesen los comandos de primitiva 3D. Los valores de estos comandos se determinan, al menos en parte, basándose en la API 3D particular en uso. En algunas realizaciones, comandos del estado de canalización de 3D 3230 también son capaces de deshabilitar o sortear selectivamente ciertos elementos de canalización si esos elementos no se van a usar.

En algunas realizaciones, el comando de la primitiva 3232 3D se usa para enviar primitivas 3D para que sean

procesadas por la canalización de 3D. Los comandos y parámetros asociados que se pasan al procesador de gráficos a través del comando de la primitiva 3232 3D se reenvían a la función de extracción de vértices en la canalización de gráficos. La función de extracción de vértices usa los datos de comando de la primitiva 3D 3232 para generar estructuras de datos de vértice. Las estructuras de datos de vértice se almacenan en una o más memorias intermedias de retorno. En algunas realizaciones, el comando de la primitiva 3D 3232 se usa para realizar operaciones de vértice sobre primitivas 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la canalización de 3D 3222 despacha hilos de ejecución de sombreador a unidades de ejecución de procesador de gráficos.

En algunas realizaciones, la canalización de 3D 3222 se desencadena a través de un comando o evento de la ejecución 3234. En algunas realizaciones, una escritura de registro desencadena una ejecución de comando. En algunas realizaciones, la ejecución se desencadena a través de un comando 'ir' o 'poner en marcha' en la secuencia de comandos. En una realización, la ejecución de comando se desencadena usando un comando de sincronización de canalización para vaciar la secuencia de comandos a través de la canalización de gráficos. La canalización de 3D realizará un procesamiento de geometría para las primitivas 3D. Una vez que se han completado las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles da color a los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo trasero de píxeles para esas operaciones.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 3210 sigue la ruta de la canalización de medios 3224 cuando se realizan operaciones de medios. En general, el uso específico y manera específicos de la programación para la canalización de medios 3224 depende de las operaciones de medios o de cómputo a realizar. Operaciones de descodificación de medios específicas se pueden descargar a la canalización de medios durante la descodificación de medios. En algunas realizaciones, la canalización de medios también se puede sortear y la descodificación de medios se puede realizar, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. En una realización, la canalización de medios también incluye elementos para operaciones de unidad de procesador de gráficos de propósito general (GPGPU), en donde el procesador de gráficos se usa para realizar operaciones vectoriales de SIMD usando programas de sombreado computacional que no están relacionados explícitamente con la representación de primitivas de gráficos.

En algunas realizaciones, la canalización de medios 3224 se configura de una manera similar a la de la canalización de 3D 3222. Un conjunto de comandos para configurar el estado de canalización de medios 3240 se despachan o se colocan en una cola de comandos antes de los comandos de objeto de medios 3242. En algunas realizaciones, los comandos de estado de canalización de medios 3240 incluyen datos para configurar los elementos de canalización de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de descodificación de vídeo y de codificación de vídeo dentro de la canalización de medios, tal como el formato de codificación o de descodificación. En algunas realizaciones, los comandos de estado de canalización de medios 3240 también soportan el uso de uno o más punteros a elementos de estado "indirectos" que contienen un lote de ajustes de estado.

En algunas realizaciones, los comandos de objeto de medios 3242 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo a procesar. En algunas realizaciones, todos los estados de canalización de medios han de ser válidos antes de emitir un comando de objeto de medios 3242. Una vez que se ha configurado el estado de canalización y los comandos de objeto de medios 3242 se han puesto en cola, la canalización de medios 3224 se desencadena a través de un comando de ejecución 3244 o un evento de ejecución equivalente (por ejemplo, una escritura de registro). La salida desde la canalización de medios 3224 se puede post-procesar entonces mediante operaciones proporcionadas por la canalización de 3D 3222 o la canalización de medios 3224. En algunas realizaciones, las operaciones de GPGPU se configuran y se ejecutan de una manera similar a la de las operaciones de medios.

Arquitectura de software de gráficos

La **Figura 33** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos 3300 de acuerdo con algunas realizaciones. En algunas realizaciones, la arquitectura de software incluye una aplicación de gráficos 3D 3310, un sistema operativo 3320 y al menos un procesador 3330. En algunas realizaciones, el procesador 3330 incluye un procesador de gráficos 3332 y uno o más núcleos de procesador de propósito general 3334. La aplicación de gráficos 3310 y el sistema operativo 3320 se ejecutan, cada uno, en la memoria de sistema 3350 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación de gráficos 3D 3310 contiene uno o más programas de sombreado que incluyen las instrucciones de sombreador 3312. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el lenguaje de sombreador de alto nivel (HLSL) o el lenguaje de sombreador de OpenGL (GLSL). La aplicación también incluye las instrucciones ejecutables 3314 en un lenguaje máquina adecuado para su ejecución por el núcleo de procesador de propósito general 3334. La aplicación también incluye los objetos de gráficos 3316 definidos por datos de vértice.

En algunas realizaciones, el sistema operativo 3320 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo de tipo UNIX de propiedad exclusiva o un sistema operativo de tipo UNIX de código abierto que usa una variante del núcleo de Linux. El sistema operativo 3320 puede soportar una API de gráficos 3322 tal como la API de Direct3D, la API de OpenGL o la API de Vulkan. Cuando está en uso la API de Direct3D, el sistema operativo 3320 usa un compilador de sombreador de extremo frontal 3324 para compilar cualquier instrucción de sombreador 3312 en HLSL a un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede realizar una precompilación de sombreador. En algunas realizaciones, sombreadores de alto nivel se compilan a sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 3310. En algunas realizaciones, las instrucciones de sombreador 3312 se proporcionan en una forma intermedia, tal como una versión de la representación intermedia portátil convencional (SPIR) usada por la API de Vulkan.

En algunas realizaciones, el controlador de gráficos de modo de usuario 3326 contiene un compilador de sombreador de extremo trasero 3327 para convertir las instrucciones de sombreador 3312 en una representación específica de hardware. Cuando está en uso la API de OpenGL, las instrucciones de sombreador 3312 en el lenguaje de alto nivel GLSL se pasan a un controlador de gráficos de modo de usuario 3326 para su compilación. En algunas realizaciones, el controlador de gráficos de modo de usuario 3326 usa las funciones de modo de núcleo de sistema operativo 3328 para comunicarse con un controlador de gráficos de modo de núcleo 3329. En algunas realizaciones, el controlador de gráficos de modo de núcleo 3329 se comunica con el procesador de gráficos 3332 para despachar comandos e instrucciones.

Implementaciones de núcleo de IP

Uno o más aspectos de al menos una realización se pueden implementar mediante un código representativo almacenado en un medio legible por máquina que representa y/o define una lógica dentro de un circuito integrado tal como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan una lógica diversa dentro del procesador. Cuando son leídas por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en el presente documento. Tales representaciones, conocidas como "núcleos de IP", son unidades reutilizables de lógica para un circuito integrado que se pueden almacenar en un medio legible por máquina tangible como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware se puede suministrar a diversos clientes o instalaciones de fabricación, que cargan el modelo de hardware en máquinas de fabricación que fabrican el circuito integrado. El circuito integrado se puede fabricar de manera que el circuito realiza operaciones descritas en asociación con cualquiera de las realizaciones descritas en el presente documento.

La **Figura 34** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 3400 que se puede usar para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo de IP 3400 se puede usar para generar diseños reutilizables modulares que se pueden incorporar en un diseño más grande o usarse para construir todo un circuito integrado (por ejemplo, un circuito integrado de SoC). Una instalación de diseño 3430 puede generar una simulación de software 3410 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). El software de simulación 3410 se puede usar para diseñar, someter a prueba y verificar el comportamiento del núcleo de IP usando un modelo de simulación 3412. El modelo de simulación 3412 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Se puede crear o sintetizar entonces un diseño de nivel de transferencia de registro (RTL) 3415 a partir del modelo de simulación 3412. El diseño de RTL 3415 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, incluyendo la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño de RTL 3415, también se pueden crear, diseñar o sintetizar diseños de nivel inferior a nivel de lógica o a nivel de transistores. Por lo tanto, los detalles particulares del diseño y simulación inicial pueden variar.

El diseño de RTL 3415, o un equivalente, puede ser sintetizado adicionalmente por la instalación de diseño para dar un modelo de hardware 3'0, que puede estar en un lenguaje de descripción de hardware (HDL) o alguna otra representación de datos de diseño físico. El HDL se puede simular o someter a prueba adicionalmente para verificar el diseño de núcleo de IP. El diseño de núcleo de IP se puede almacenar para su entrega a una instalación de fabricación de terceros 3465 usando la memoria no volátil 3440 (por ejemplo, disco duro, memoria flash o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño de núcleo de IP se puede transmitir (por ejemplo, a través de Internet) a través de una conexión cableada 3450 o una conexión inalámbrica 3460. La instalación de fabricación 3465 puede fabricar entonces un circuito integrado que se basa, al menos en parte, en el diseño de núcleo de IP. El circuito integrado fabricado se puede configurar para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Circuito integrado de sistema en un chip ilustrativo

Las **Figuras 35-37** ilustran circuitos integrados ilustrativos y procesadores de gráficos asociados que se pueden fabricar usando uno o más núcleos de IP, de acuerdo con diversas realizaciones descritas en el presente documento. Además de lo que se ilustra, se pueden incluir otros circuitos y lógica, incluyendo procesadores/núcleos de gráficos

adicionales, controladores de interfaz de periféricos o núcleos de procesador de propósito general.

La **Figura 35** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ilustrativo 3500 que se puede fabricar usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 3500 ilustrativo incluye uno o más procesadores de aplicaciones 3505 (por ejemplo, unas CPU), al menos un procesador de gráficos 3510, y puede incluir adicionalmente un procesador de imágenes 3515 y/o un procesador de vídeo 3520, cualquiera de los cuales puede ser un núcleo de IP modular desde las mismas o múltiples instalaciones de diseño diferentes. El circuito integrado 3500 incluye lógica de bus o de periféricos que incluye un controlador de USB 3525, un controlador de UART 3530, un controlador de SPI/SDIO 3535 y un controlador de I^2S/I^2C 3540. Adicionalmente, el circuito integrado puede incluir un dispositivo de visualización 3545 acoplado a uno o más de un controlador de interfaz multimedios de alta definición (HDMI) 3550 y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 3555. El almacenamiento puede ser proporcionado por un subsistema de memoria flash 3560 que incluye memoria flash y un controlador de memoria flash. La interfaz de memoria se puede proporcionar a través de un controlador de memoria 3565 para el acceso a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 3570.

La **Figura 36** es un diagrama de bloques que ilustra un procesador de gráficos 3610 ilustrativo de un circuito integrado de sistema en un chip que se puede fabricar usando uno o más núcleos de IP, de acuerdo con una realización. El procesador de gráficos 3610 puede ser una variante del procesador de gráficos 3610 de la **Figura 36**. El procesador de gráficos 3610 incluye un procesador de vértices 3605 y uno o más procesadores de fragmentos 3615A-3615N (por ejemplo, 3615A, 3615B, 3615C, 3615D a 3615N-1 y 3615N). El procesador de gráficos 3610 puede ejecutar diferentes programas de sombreado a través de lógica separada, de manera que el procesador de vértices 3605 se optimiza para ejecutar operaciones para programas de sombreado de vértices, mientras que los uno o más procesadores de fragmentos 3615A-3615N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreado de fragmentos o de píxeles. El procesador de vértices 3605 realiza la fase de procesamiento de vértices de la canalización de gráficos 3D y genera primitivas y datos de vértice. El procesador o procesadores de fragmentos 3615A-3615N usan los datos de primitiva y de vértice generados por el procesador de vértices 3605 para producir una memoria intermedia de tramas que se visualiza en un dispositivo de visualización. En una realización, el procesador o procesadores de fragmentos 3615A-3615N se optimizan para ejecutar programas de sombreado de fragmentos según lo previsto en la API de OpenGL, que se pueden usar para realizar operaciones similares como un programa de sombreado de píxeles según lo previsto en la API de Direct 3D.

El procesador de gráficos 3610 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 3620A-3620B, caché o cachés 3625A-3625B e interconexión o interconexiones de circuito 3630A-3630B. Las una o más MMU 3620A-3620B prevén una correlación de dirección virtual a física para el circuito integrado 3610, incluyendo para el procesador de vértices 3605 y/o el procesador o procesadores de fragmentos 3615A-3615N, que pueden hacer referencia a datos de vértice o de imagen/textura almacenados en memoria, además de datos de vértice o de imagen/textura almacenados en las una o más cachés 3625A-3625B. En una realización, una o más MMU 3625A-3625B se pueden sincronizar con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con uno o más procesadores de aplicaciones 3605, procesadores de imágenes 3615 y/o procesador de vídeo 3620 de la **Figura 36**, de manera que cada procesador 3605-3620 puede participar en un sistema de memoria virtual compartida o unificada. Las una o más interconexiones de circuito 3630A-3630B habilitan que el procesador de gráficos 3610 interaccione con otros núcleos de IP dentro del SoC, o bien a través de un bus interno del SoC o bien a través de una conexión directa, de acuerdo con realizaciones.

La **Figura 37** es un diagrama de bloques que ilustra un procesador de gráficos 3710 ilustrativo adicional de un circuito integrado de sistema en un chip que se puede fabricar usando uno o más núcleos de IP, de acuerdo con una realización. El procesador de gráficos 3710 puede ser una variante del procesador de gráficos 3510 de la **Figura 35**. El procesador de gráficos 3710 incluye las una o más MMU 3520A-3520B, cachés 3525A-3525B e interconexiones de circuito 3530A-3530B del circuito integrado 3500 de la **Figura 35**.

El procesador de gráficos 3710 incluye uno o más núcleos de sombreador 3715A-3715N (por ejemplo, 3715A, 3715B, 3715C, 3715D, 3715E, 3715F a 3715N-1 y 3715N), lo que prevé una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo o núcleo puede ejecutar todos los tipos de código de sombreado programable, incluyendo código de programa de sombreado para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cómputo. El número exacto de núcleos de sombreador presentes puede variar entre realizaciones e implementaciones. Adicionalmente, el procesador de gráficos 3710 incluye un gestor de tareas entre núcleos 3705, que actúa como un despachador de hilos para despachar hilos de ejecución a uno o más núcleos de sombreador 3715A-3715N y una unidad de teselado 3718 para acelerar operaciones de teselado para una representación basada en teselas, en la que operaciones de representación para una escena se subdividen en el espacio de imágenes, por ejemplo, para aprovechar la coherencia espacial local dentro de una escena o para optimizar el uso de cachés internas.

Las realizaciones descritas en el presente documento proporcionan una unidad de lógica que incluye una ruta de datos de números enteros/coma flotante combinada para operaciones tanto de multiplicación y de suma (por ejemplo, a * b + c) como de multiplicación - acumulación (por ejemplo, c = c + a * b). En una realización, un sumando para la

operación de suma se basa en una acumulación de operaciones previas. En una realización, la ruta de datos de números enteros de la unidad de lógica se combina para dar una ruta de datos de coma flotante que tiene una operación de alineación de sumando en paralelo con una operación de multiplicación. En una realización, la ruta de datos de números enteros se combina para dar una ruta de datos de coma flotante que tiene una operación de alineación de sumando después de la operación de multiplicación. Las rutas de datos de multiplicación - suma y de multiplicación - acumulación descritas en el presente documento pueden ser de un único ciclo o de múltiples ciclos.

En una realización, durante una multiplicación - acumulación de coma flotante de dos ciclos, la unidad de lógica no compara las mantisas al comienzo de la segunda fase (por ejemplo, la fase de sumador). En su lugar, la unidad de lógica precomputa una mantisa más grande (o más pequeña) basándose en un exponente de acumulador a partir de la segunda fase y la salida de multiplicador computada durante la primera fase.

En una realización, la anchura de bits de mantisa de sumando o de acumulador es más grande que la anchura de bits de mantisa de las entradas de multiplicador. En una realización, las operaciones de números enteros se correlacionan en una unidad de coma flotante. Algunas de las operaciones de números enteros también se correlacionan en circuitos de exponente existentes además de en los circuitos de mantisa de una unidad de coma flotante. En una realización, las unidades de lógica descritas en el presente documento incluyen una unidad de multiplicación y una unidad de suma que se comparten entre operaciones de coma flotante y de números enteros y se usan para realizar operaciones tanto de coma flotante como de números enteros.

Las cláusulas y/o ejemplos siguientes se refieren a realizaciones específicas o a ejemplos de las mismas. Detalles específicos en los ejemplos se pueden usar en cualquier lugar en una o más realizaciones. Las diversas características de las diferentes realizaciones o ejemplos se pueden combinar de manera diversa con algunas características incluidas y otras excluidas para adecuarse a una diversidad de aplicaciones diferentes. Los ejemplos pueden incluir materia objeto tal como un método, medios para realizar actos del método, al menos un medio legible por máquina que incluye instrucciones que, cuando son realizadas por una máquina, hacen que la máquina realice actos del método, o de un aparato o sistema de acuerdo con realizaciones y ejemplos descritos en el presente documento. Diversos componentes pueden ser un medio para realizar las operaciones o funciones descritas.

Una realización proporciona un acelerador de hardware de aprendizaje automático que comprende una unidad de cómputo que tiene un sumador y un multiplicador que se comparten entre una ruta de datos de números enteros y una ruta de datos de coma flotante, con los bits superiores de los operandos de entrada al multiplicador controlándose durante una operación de coma flotante. En una realización, el sumador y el multiplicador son configurables para realizar una operación de coma flotante y una operación de números enteros. En una realización, la unidad de cómputo ha de realizar una operación de multiplicación - suma a través del multiplicador y el sumador. En una realización, la unidad de cómputo acepta al menos dos operandos de entrada. Una realización proporciona una unidad de cómputo para realizar una operación de multiplicación - acumulación usando operandos de dos entradas y un valor acumulado. Una realización proporciona una unidad de cómputo para realizar una operación de multiplicación - suma usando tres operandos de entrada. En una realización, la unidad de cómputo ha de realizar una operación de multiplicación - acumulación o una operación de multiplicación - suma dentro de un único ciclo. En una realización, la unidad de cómputo ha de realizar una operación de multiplicación - suma de dos ciclos o una operación de multiplicación - acumulación de dos ciclos. En una realización, el multiplicador dentro de la unidad de cómputo ha de producir una salida durante un primer ciclo y el sumador ha de producir una salida durante un segundo ciclo. En una realización, la unidad de cómputo ha de realizar una operación de multiplicación - acumulación de dos ciclos en la que el primer ciclo está asociado con una primera fase de lógica, el segundo ciclo está asociado con una segunda fase de lógica y la unidad de cómputo incluye una unidad de exponente para precomputar una mantisa y un desplazamiento de alineación más grandes para la segunda fase a través de una salida acumulada de un ciclo previo de la segunda fase y una salida de multiplicador a partir de la primera fase.

En una realización, la ruta de datos de números enteros se combina para dar la ruta de datos de coma flotante que tiene una operación de alineación de sumando en paralelo con la operación de multiplicación. En una realización, la ruta de datos de números enteros se combina para dar la ruta de datos de coma flotante que tiene una operación de alineación de sumando después de la operación de multiplicación. La unidad de cómputo puede tener una entrada de modo para conmutar la unidad de cómputo entre una operación de números enteros y una operación de coma flotante. En una realización, la unidad de cómputo es configurable para una entrada de coma fija de 8,8 y una salida de coma fija de 16,0.

Una realización proporciona un sistema de procesamiento de datos que comprende un medio legible por máquina no transitorio para almacenar instrucciones para su ejecución por uno o más procesadores del sistema de procesamiento de datos; y una unidad de procesamiento de gráficos de propósito general que comprende un acelerador de hardware de aprendizaje automático y una unidad de cómputo de precisión dinámica, incluyendo el acelerador de hardware de aprendizaje automático una lógica de hardware para realizar múltiples operaciones de cómputo de aprendizaje automático en respuesta a una única instrucción. En una realización, la unidad de cómputo de precisión dinámica se puede conmutar entre una operación de números enteros y una operación de coma flotante. En una realización, la unidad de cómputo de precisión dinámica incluye una ruta de datos de números enteros y una ruta de datos de coma flotante que comparten un multiplicador y un sumador, en donde el multiplicador ha de realizar una operación de

multiplicación para la ruta de datos de números enteros y la ruta de datos de coma flotante. En una realización, la ruta de datos de coma flotante incluye una operación de alineación de sumando realizada en paralelo con la operación de multiplicación. En una realización, la ruta de datos de coma flotante incluye una operación de alineación de sumando realizada después de la operación de multiplicación. En una realización, la unidad de cómputo de precisión dinámica está configurada para una operación de multiplicación - suma fusionada de un único ciclo o una operación de multiplicación - acumulación fusionada de dos ciclos.

Una realización proporciona un método para acelerar operaciones de aprendizaje automático, comprendiendo el método extraer y descodificar una única instrucción para realizar una operación de multiplicación y de suma combinada sobre un conjunto de operandos; emitir la única instrucción para su ejecución por una unidad de cómputo configurable dinámicamente; configurar una o más unidades de lógica de la unidad de cómputo para realizar operaciones con la precisión y el tipo de datos del conjunto de operandos; y ejecutar al menos una porción de la única instrucción en la unidad de cómputo configurable dinámicamente para generar una salida basándose en la operación de multiplicación y de suma.

Las realizaciones descritas en el presente documento se refieren a configuraciones específicas de hardware, tales como circuitos integrados específicos de la aplicación (ASIC), configuradas para realizar ciertas operaciones o que tienen una funcionalidad predeterminada. Tales dispositivos electrónicos incluyen habitualmente un conjunto de uno o más procesadores acoplados a otros uno o más componentes, tales como uno o más dispositivos de almacenamiento (medios de almacenamiento legibles por máquina no transitorios), dispositivos de entrada/salida de usuario (por ejemplo, un teclado, una pantalla táctil y/o una pantalla) y conexiones de red. El acoplamiento del conjunto de procesadores y otros componentes se produce habitualmente a través de uno o más buses y puentes (también denominados controladores de bus). El dispositivo de almacenamiento y las señales que portan el tráfico de red representan, respectivamente, uno o más medios de almacenamiento legibles por máquina y medios de comunicación legibles por máquina. Por lo tanto, los dispositivos de almacenamiento de un dispositivo electrónico dado almacenan habitualmente código y/o datos para su ejecución en el conjunto de uno o más procesadores de ese dispositivo electrónico.

Por supuesto, una o más partes de una realización se pueden implementar usando diferentes combinaciones de software, firmware y/o hardware. A lo largo de toda esta descripción detallada, con fines de explicación, se expusieron numerosos detalles específicos para proporcionar un entendimiento completo de la presente invención. Sin embargo, será evidente para un experto en la materia que las realizaciones se pueden poner en práctica sin algunos de estos detalles específicos. En ciertas instancias, estructuras y funciones bien conocidas no se describieron con todo lujo de detalles para evitar complicar la materia objeto inventiva de las realizaciones. En consecuencia, el alcance de la invención debería evaluarse en términos de las reivindicaciones que se dan a continuación.

REIVINDICACIONES

1. Una unidad de procesamiento de gráficos para acelerar operaciones de aprendizaje automático, comprendiendo la unidad de procesamiento de gráficos:

un multiprocesador (1400) que incluye una arquitectura de múltiples hilos y única instrucción, el multiprocesador para ejecutar una única instrucción usando múltiples hilos;
una unidad de cómputo (1411-1418) incluida dentro del multiprocesador (1400) en donde
la única instrucción es para hacer que la unidad de cómputo (1411-1418) realice al menos una operación de acumulación y multiplicación de matrices bidimensionales, en donde realizar la operación de acumulación y multiplicación de matrices bidimensionales incluye realizar una operación de acumulación y multiplicación de comas flotantes en operandos indicados por la única instrucción que está en un formato de coma flotante no convencional: y
en donde la unidad de cómputo (1411-1418) soporta múltiples precisiones que incluyen precisiones para los formatos de coma flotante y formatos de coma flotante no convencionales de IEEE 754, en donde los formatos de coma flotante no convencionales usan asignaciones de bit diferentes para exponente y mantisa como los formatos de coma flotante de IEEE 754 y está diseñado para realizar la operación de acumulación y multiplicación de comas flotantes de manera selectiva en una de las múltiples precisiones basándose en el formato de coma no convencional indicado por la única instrucción

2. La unidad de procesamiento de gráficos según la reivindicación 1, el multiprocesador (1400) para ejecutar hilos paralelos de un grupo de hilos, teniendo cada hilo del grupo de hilos un estado de hilo independiente.

3. La unidad de procesamiento de gráficos según la reivindicación 2, en donde la unidad de procesamiento de gráficos incluye múltiples unidades de cómputo (1411-1418) y un planificador para planificar los hilos paralelos a múltiples unidades de cómputo (1411-1418).

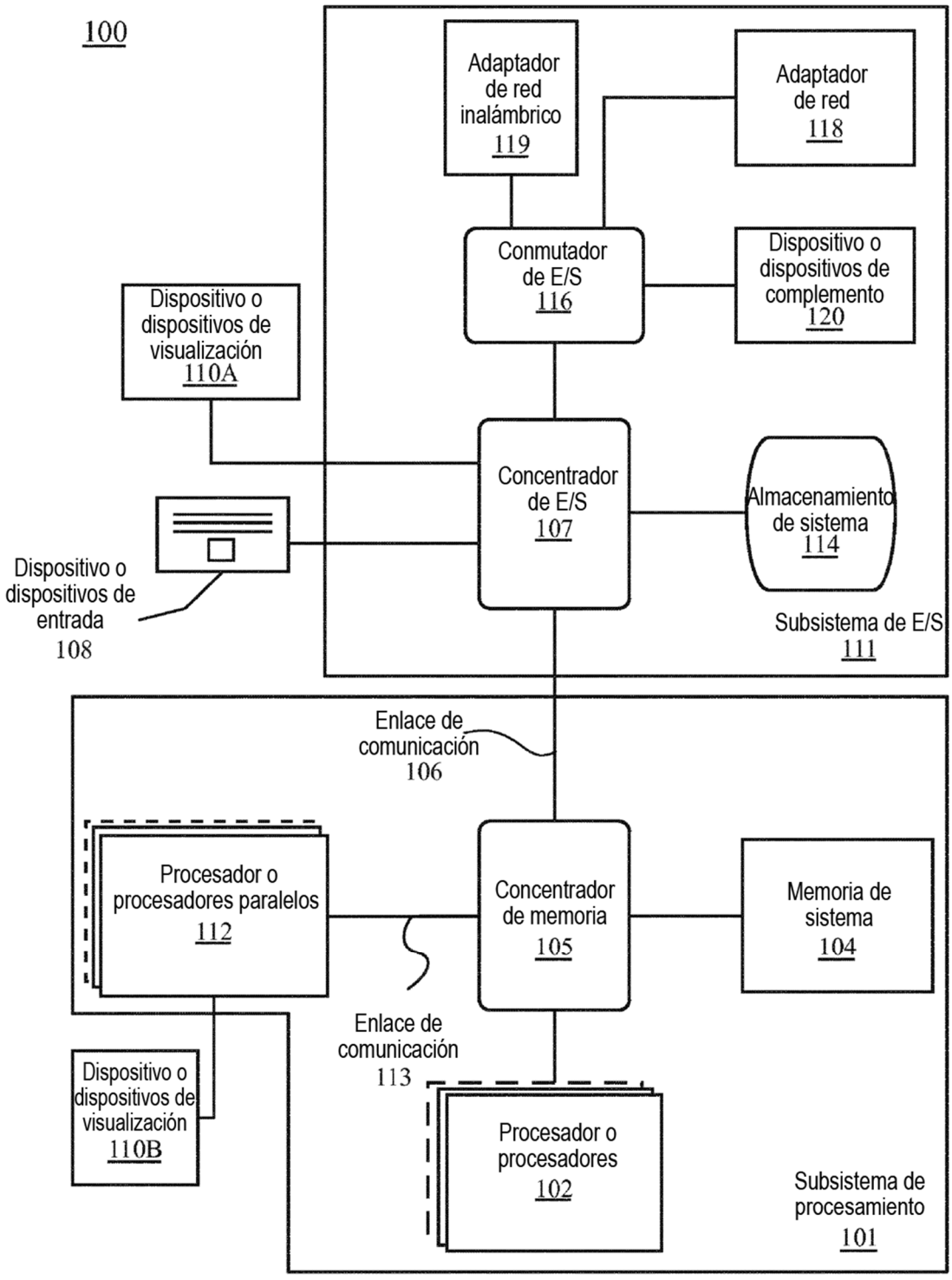

100
Adaptador de red inalámbrico 119
Adaptador de red 118
Conmutador de E/S 116
Dispositivo o dispositivos de complemento 120
Dispositivo o dispositivos de visualización 110A
Concentrador de E/S 107
Almacenamiento de sistema 114
Dispositivo o dispositivos de entrada 108
Subsistema de E/S 111
Enlace de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

FIG. 1

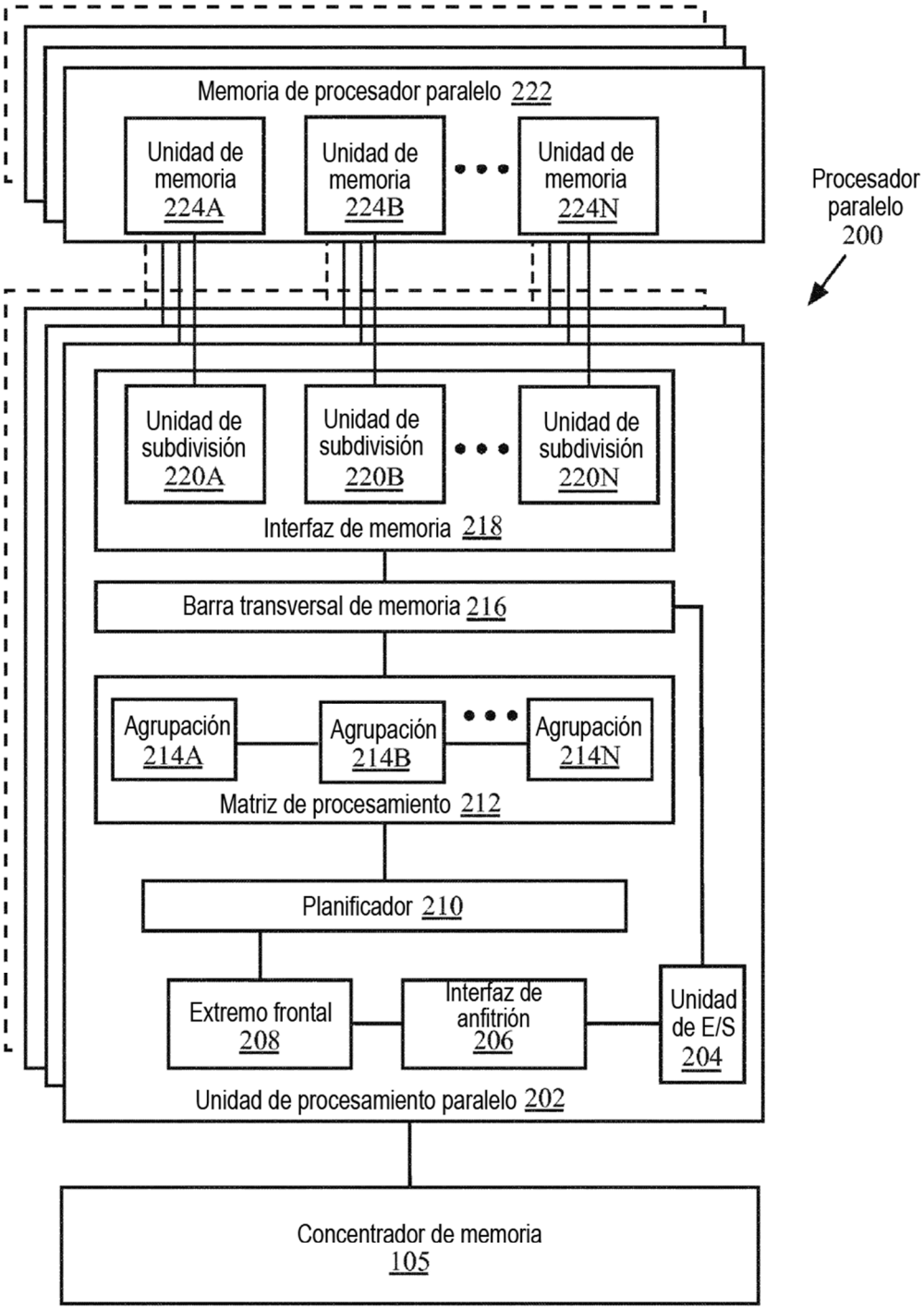

Memoria de procesador paralelo 222
Unidad de memoria 224A
Unidad de memoria 224B
Unidad de memoria 224N
Procesador paralelo 200
Unidad de subdivisión 220A
Unidad de subdivisión 220B
Unidad de subdivisión 220N
Interfaz de memoria 218
Barra transversal de memoria 216
Agrupación 214A
Agrupación 214B
Agrupación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

FIG. 2A

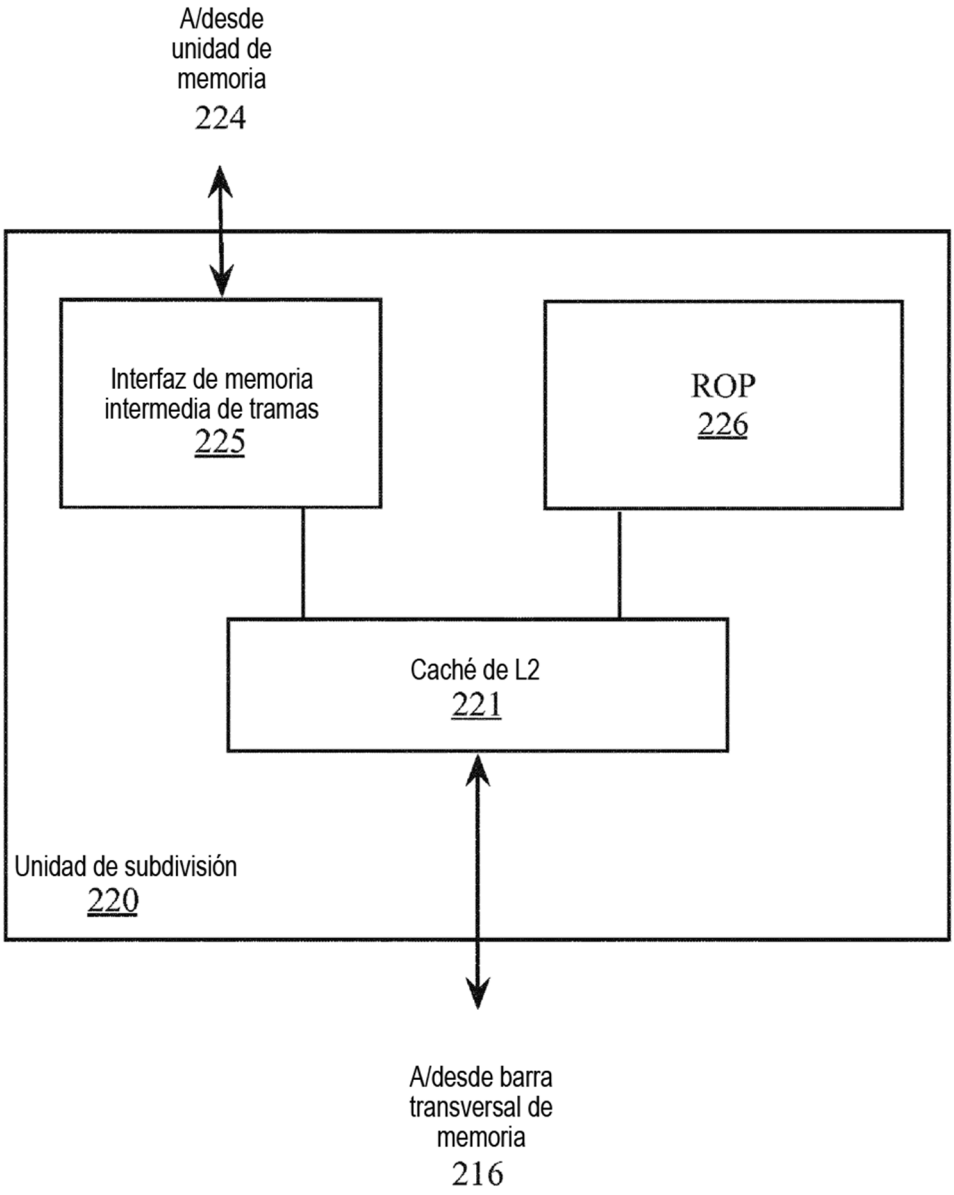

A/desde
unidad de
memoria
224
Interfaz de memoria
intermedia de tramas
225
ROP
226
Caché de L2
221
Unidad de subdivisión
220
A/desde barra
transversal de
memoria
216

FIG. 2B

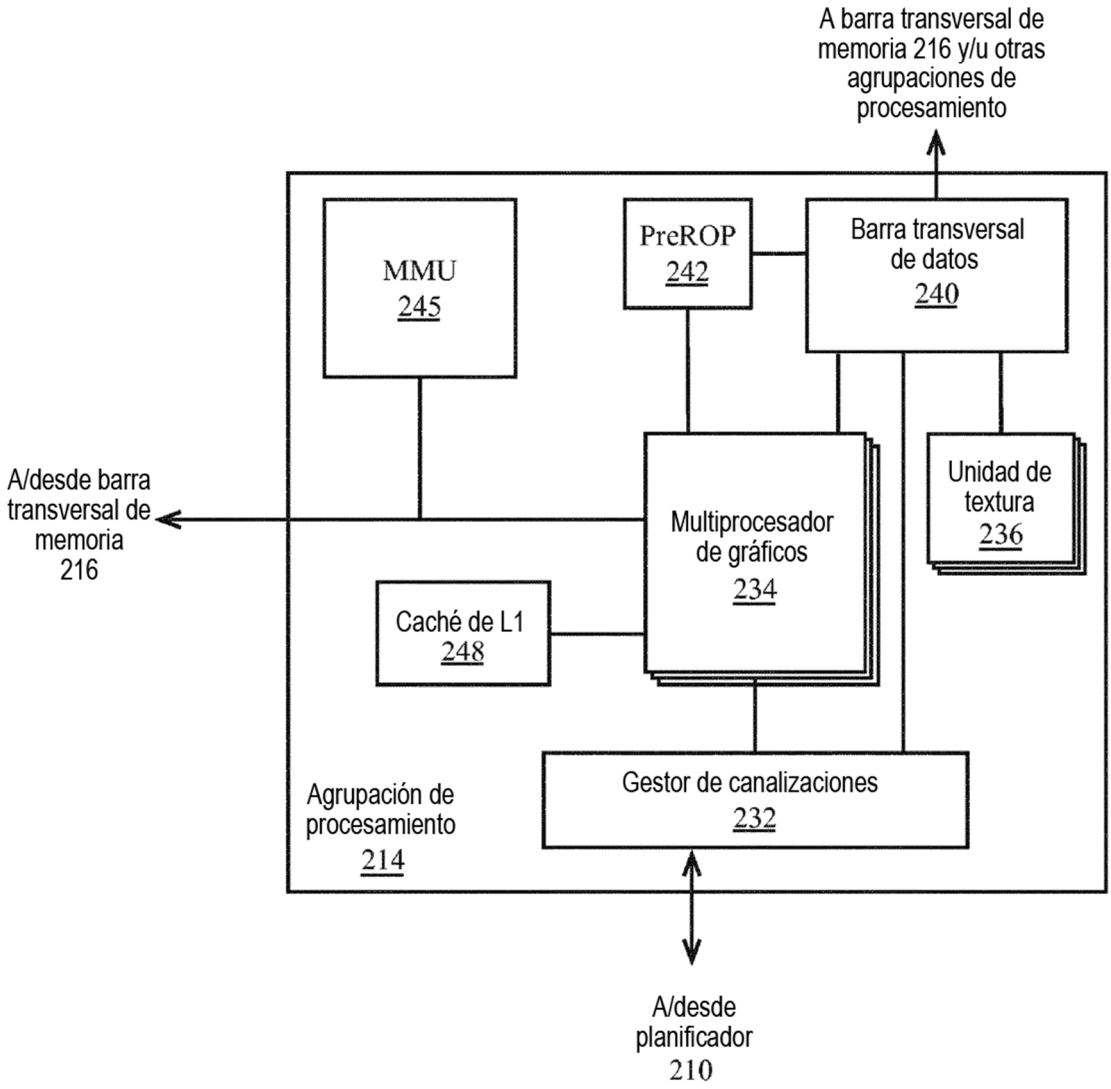

A barra transversal de memoria 216 y/u otras agrupaciones de procesamiento
MMU 245
PreROP 242
Barra transversal de datos 240
A/desde barra transversal de memoria 216
Multiprocesador de gráficos 234
Unidad de textura 236
Caché de L1 248
Agrupación de procesamiento 214
Gestor de canalizaciones 232
A/desde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

FIG. 2C

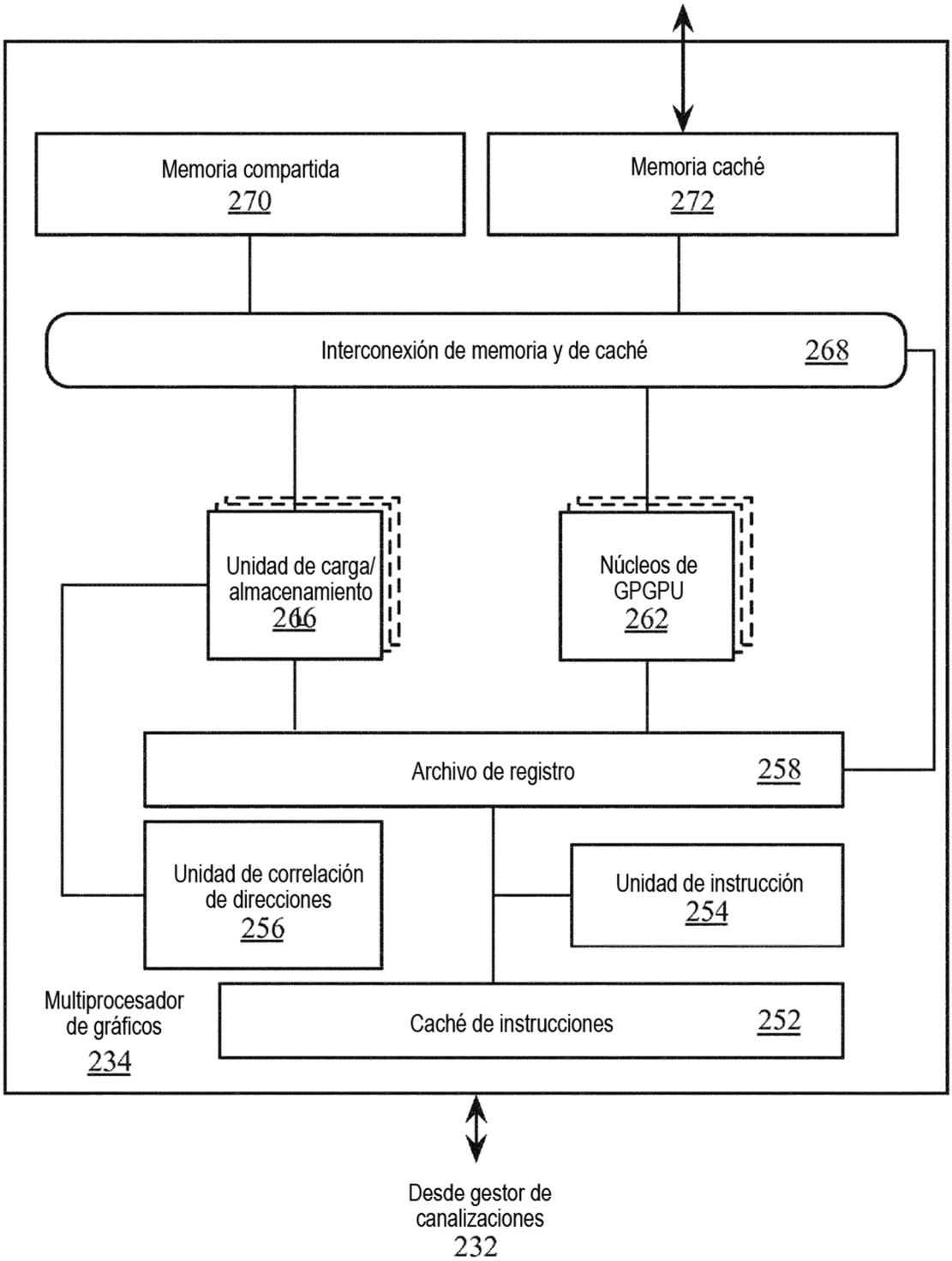

Memoria compartida
270
Memoria caché
272
Interconexión de memoria y de caché
268
Unidad de carga/
almacenamiento
266
Núcleos de
GPGPU
262
Archivo de registro
258
Unidad de correlación
de direcciones
256
Unidad de instrucción
254
Multiprocesador
de gráficos
234
Caché de instrucciones
252
Desde gestor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

FIG. 2D

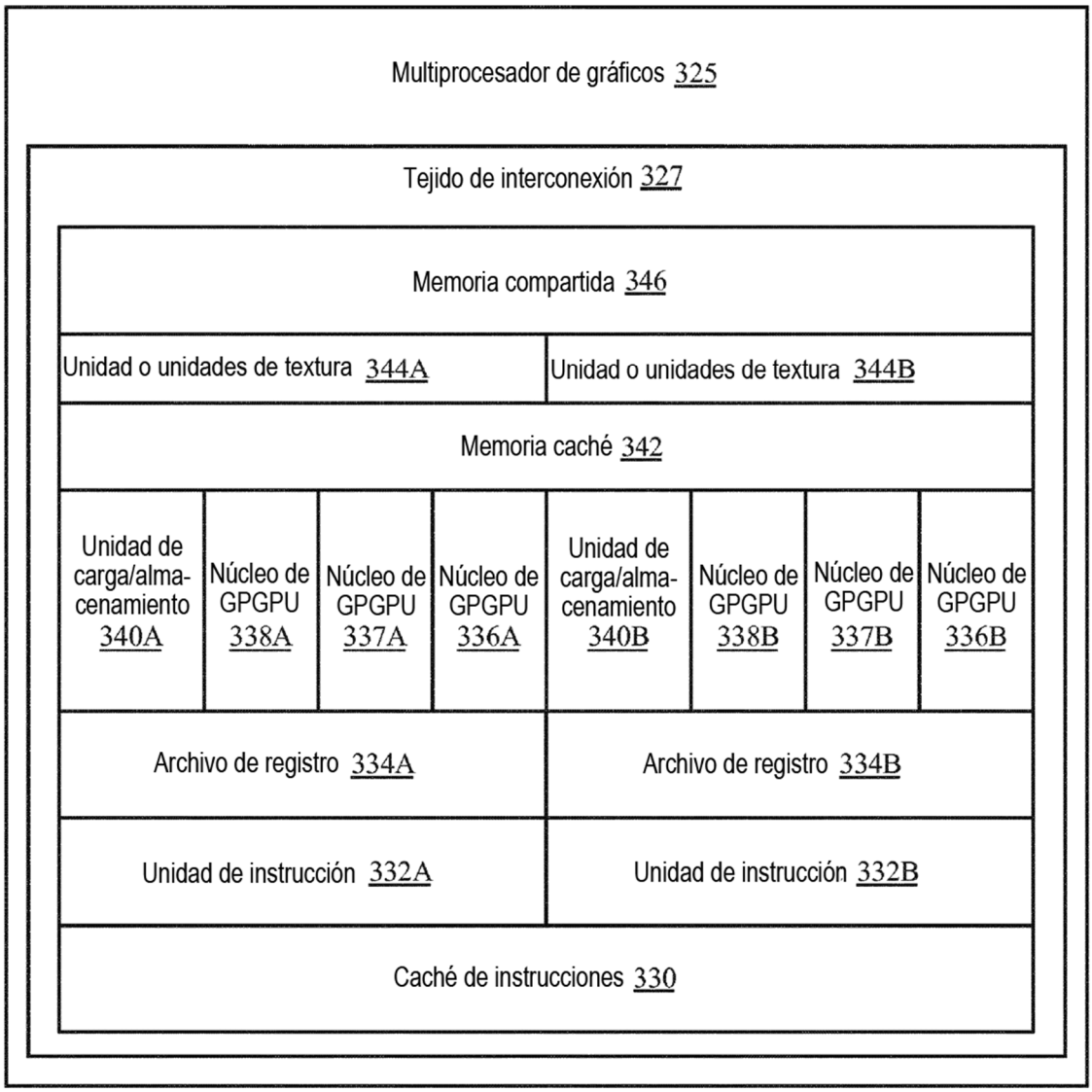

Multiprocesador de gráficos 325
Tejido de interconexión 327
Memoria compartida 346
Unidad o unidades de textura 344A
Unidad o unidades de textura 344B
Memoria caché 342
Unidad de carga/alma-cenamiento 340A
Núcleo de GPGPU 338A
Núcleo de GPGPU 337A
Núcleo de GPGPU 336A
Unidad de carga/alma-cenamiento 340B
Núcleo de GPGPU 338B
Núcleo de GPGPU 337B
Núcleo de GPGPU 336B
Archivo de registro 334A
Archivo de registro 334B
Unidad de instrucción 332A
Unidad de instrucción 332B
Caché de instrucciones 330

FIG. 3A

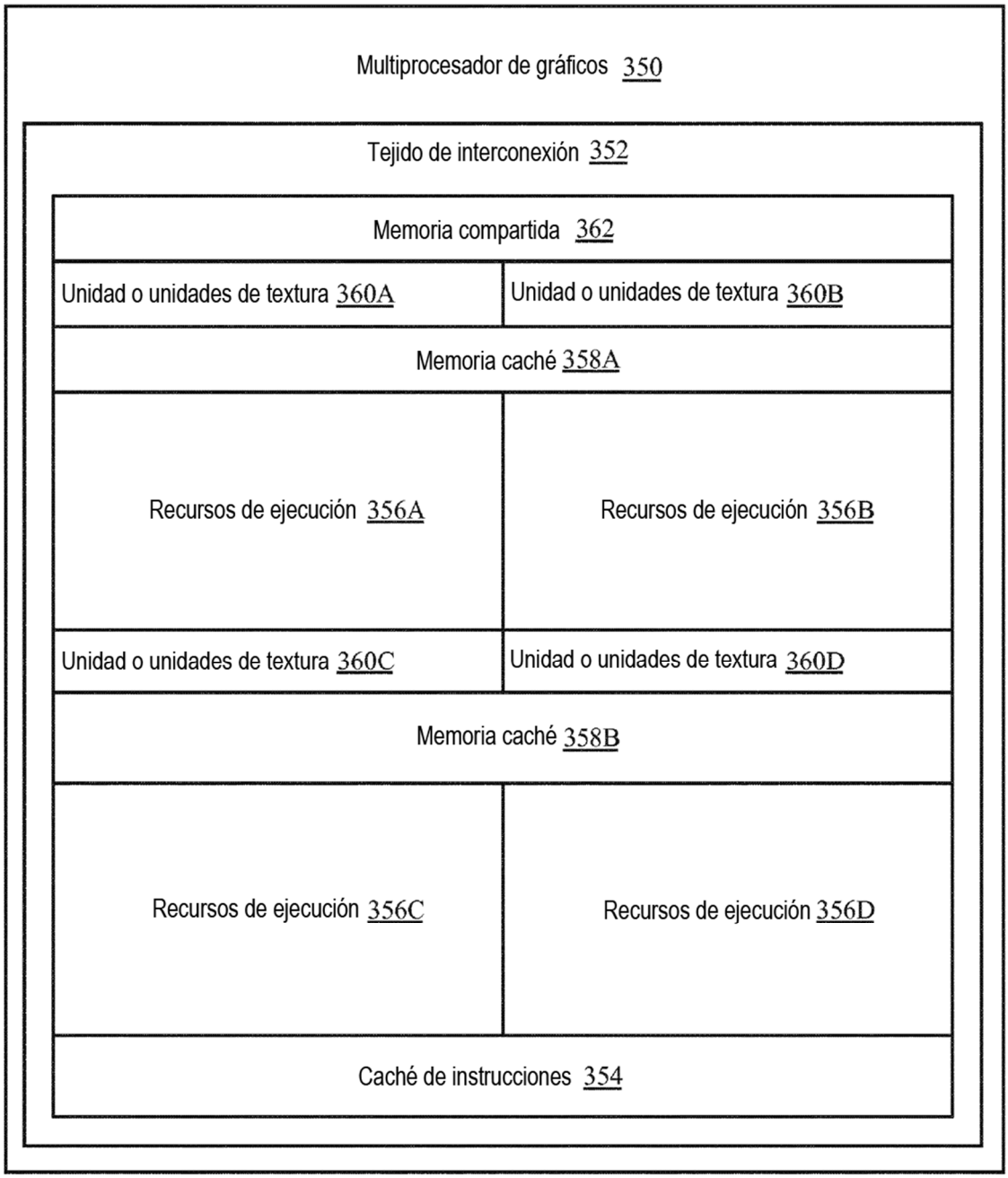

Multiprocesador de gráficos 350
Tejido de interconexión 352
Memoria compartida 362
Unidad o unidades de textura 360A
Unidad o unidades de textura 360B
Memoria caché 358A
Recursos de ejecución 356A
Recursos de ejecución 356B
Unidad o unidades de textura 360C
Unidad o unidades de textura 360D
Memoria caché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

FIG. 3B

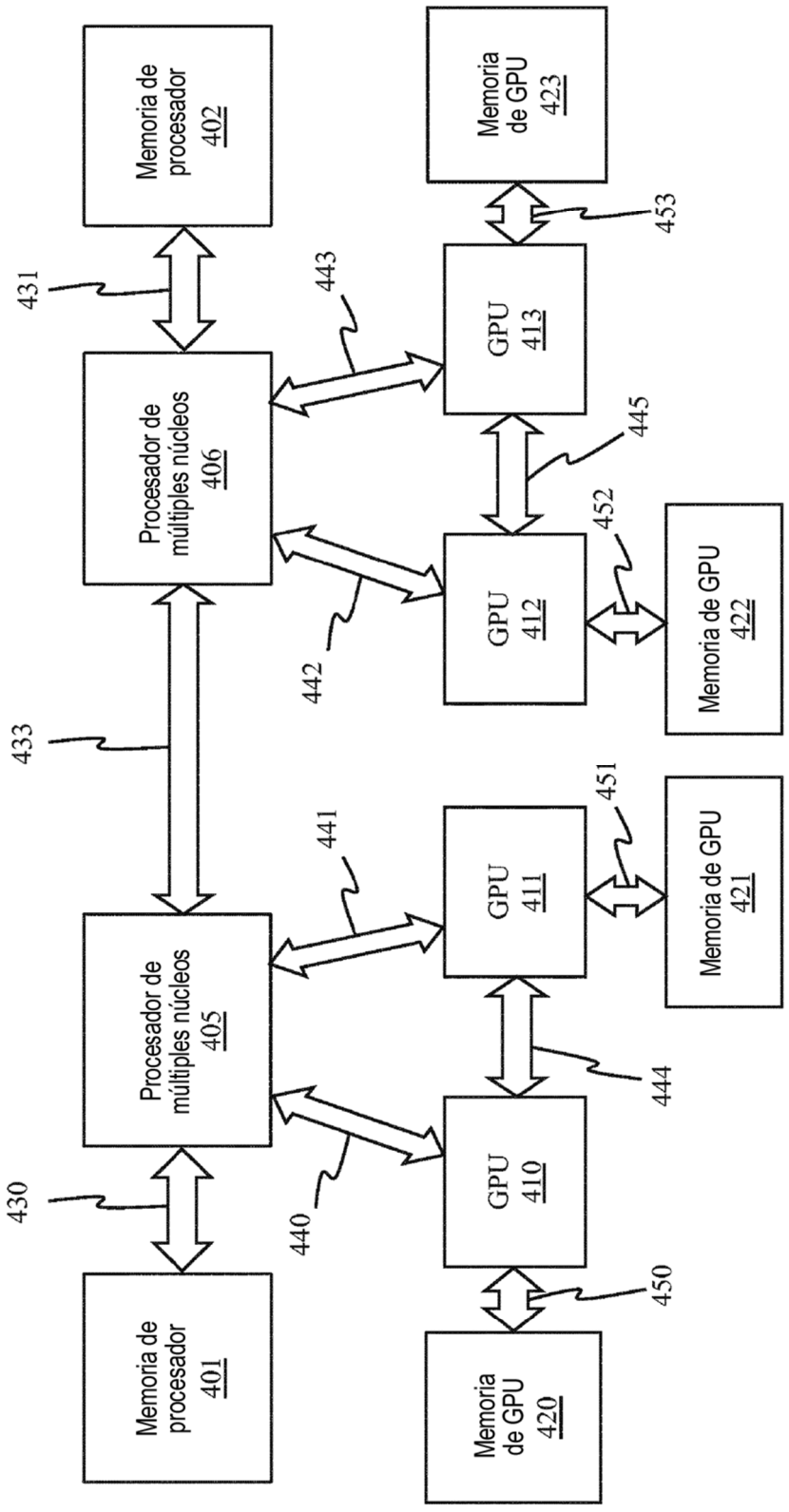

Memoria de procesador 402
Memoria de GPU 423
431
443
453
GPU 413
Procesador de múltiples núcleos 406
445
452
433
442
GPU 412
Memoria de GPU 422
441
451
GPU 411
Memoria de GPU 421
Procesador de múltiples núcleos 405
444
430
440
GPU 410
450
Memoria de procesador 401
Memoria de GPU 420

FIG. 4A

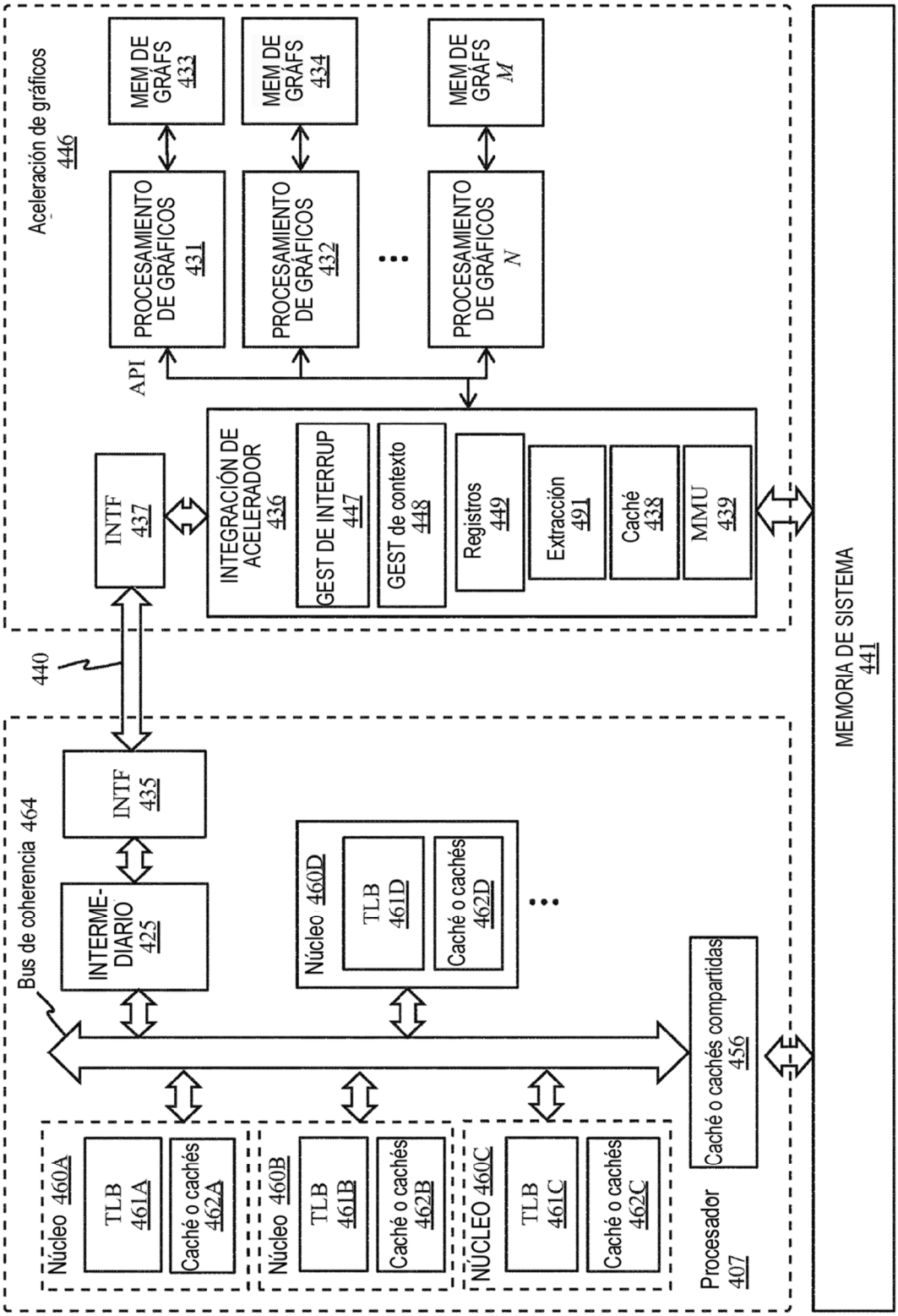

Aceleración de gráficos 446
MEM DE GRÁFS 433
MEM DE GRÁFS 434
MEM DE GRÁFS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 437
INTEGRACIÓN DE ACELERADOR 436
GEST DE INTERRUP 447
GEST de contexto 448
Registros 449
Extracción 491
Caché 438
MMU 439
440
Bus de coherencia 464
INTF 435
INTERME-DIARIO 425
Núcleo 460D
TLB 461D
Caché o cachés 462D
Caché o cachés compartidas 456
Núcleo 460A
TLB 461A
Caché o cachés 462A
Núcleo 460B
TLB 461B
Caché o cachés 462B
NÚCLEO 460C
TLB 461C
Caché o cachés 462C
Procesador 407
MEMORIA DE SISTEMA 441

FIG. 4B

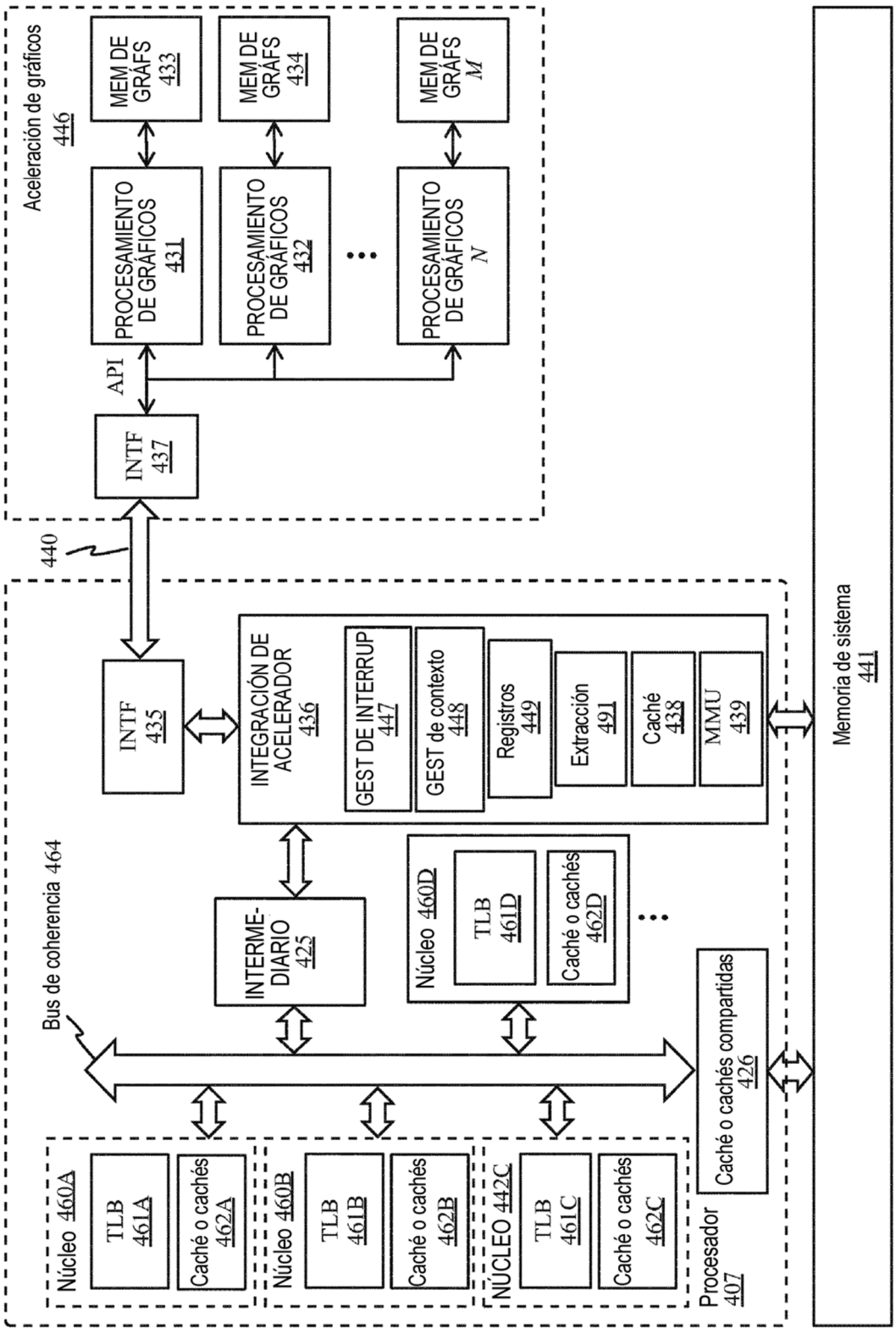

Aceleración de gráficos 446
MEM DE GRÁFS 433
MEM DE GRÁFS 434
MEM DE GRÁFS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 437
440
INTF 435
INTEGRACIÓN DE ACELERADOR 436
GEST DE INTERRUP 447
GEST de contexto 448
Registros 449
Extracción 491
Caché 438
MMU 439
Memoria de sistema 441
Bus de coherencia 464
INTERME-DIARIO 425
Núcleo 460D
TLB 461D
Caché o cachés 462D
Caché o cachés compartidas 426
Núcleo 460A
TLB 461A
Caché o cachés 462A
Núcleo 460B
TLB 461B
Caché o cachés 462B
NÚCLEO 442C
TLB 461C
Caché o cachés 462C
Procesador 407

FIG. 4C

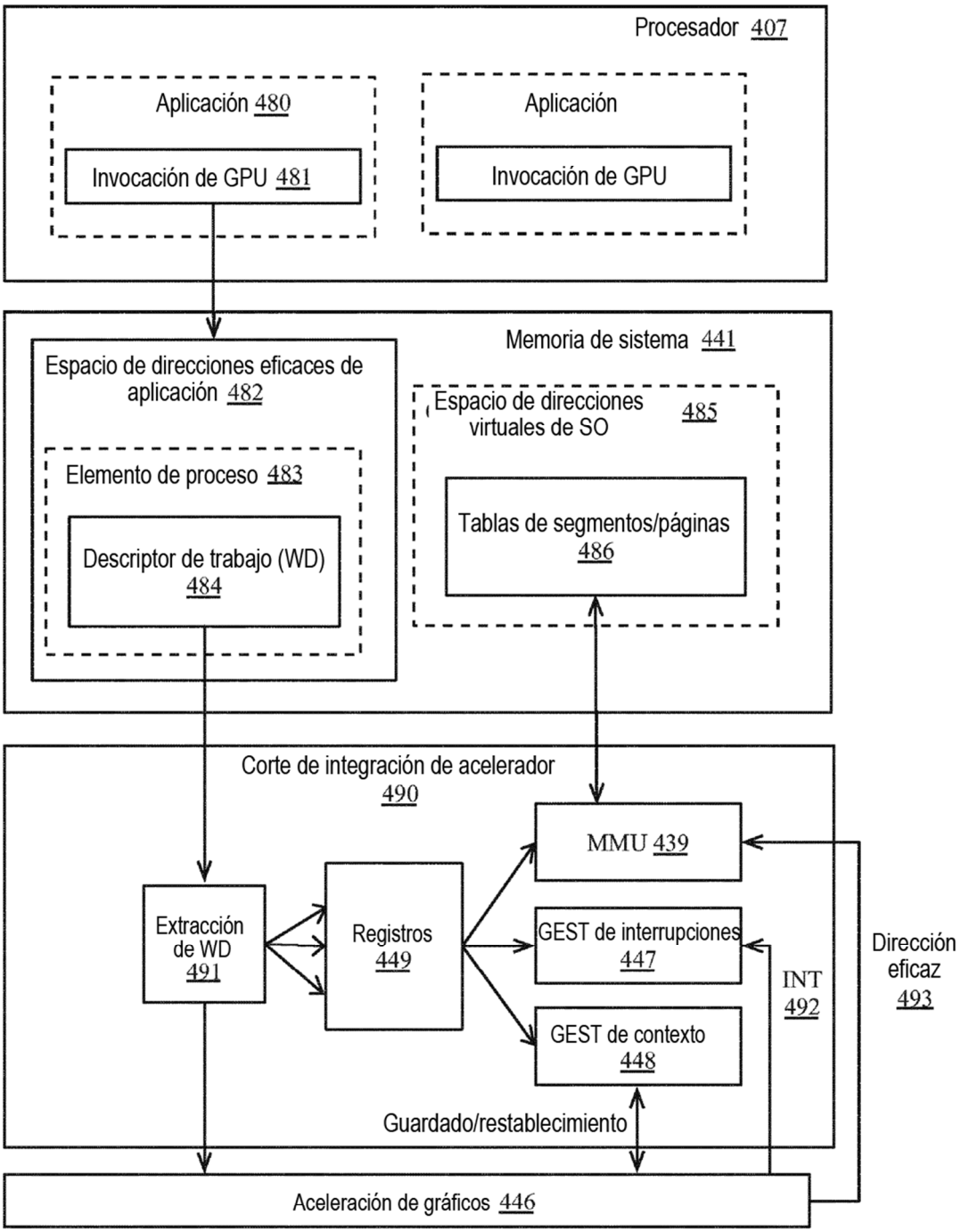

Procesador 407
Aplicación 480
Invocación de GPU 481
Aplicación
Invocación de GPU
Memoria de sistema 441
Espacio de direcciones eficaces de aplicación 482
Elemento de proceso 483
Descriptor de trabajo (WD) 484
Espacio de direcciones virtuales de SO 485
Tablas de segmentos/páginas 486
Corte de integración de acelerador 490
MMU 439
Extracción de WD 491
Registros 449
GEST de interrupciones 447
GEST de contexto 448
INT 492
Dirección eficaz 493
Guardado/restablecimiento
Aceleración de gráficos 446

FIG. 4D

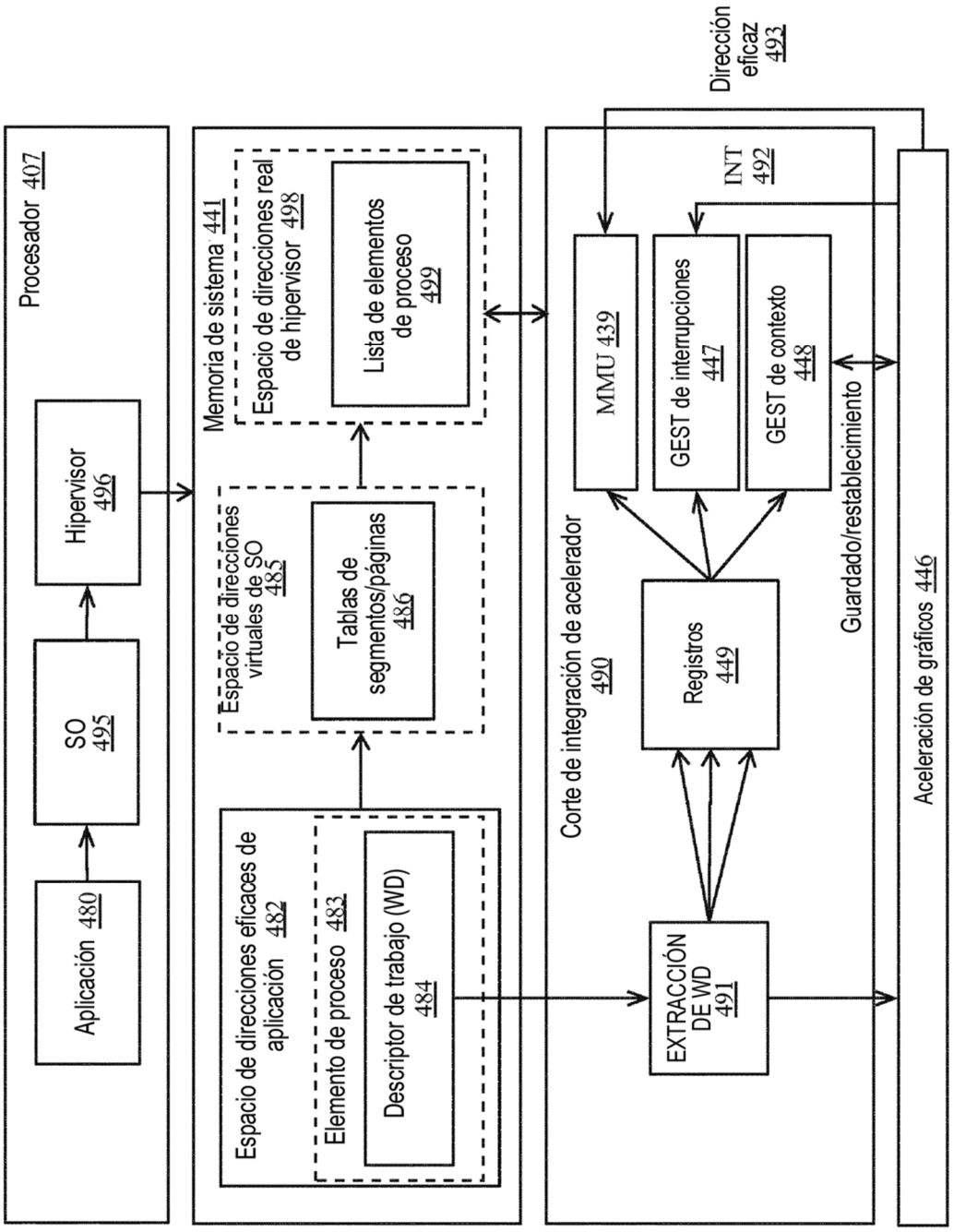

Procesador 407
Aplicación 480
SO 495
Hipervisor 496
Memoria de sistema 441
Espacio de direcciones real de hipervisor 498
Lista de elementos de proceso 499
Espacio de direcciones virtuales de SO 485
Tablas de segmentos/páginas 486
Espacio de direcciones eficaces de aplicación 482
Elemento de proceso 483
Descriptor de trabajo (WD) 484
Corte de integración de acelerador 490
MMU 439
GEST de interrupciones 447
GEST de contexto 448
Registros 449
EXTRACCIÓN DE WD 491
Guardado/restablecimiento
Dirección eficaz 493
INT 492
Aceleración de gráficos 446

FIG. 4E

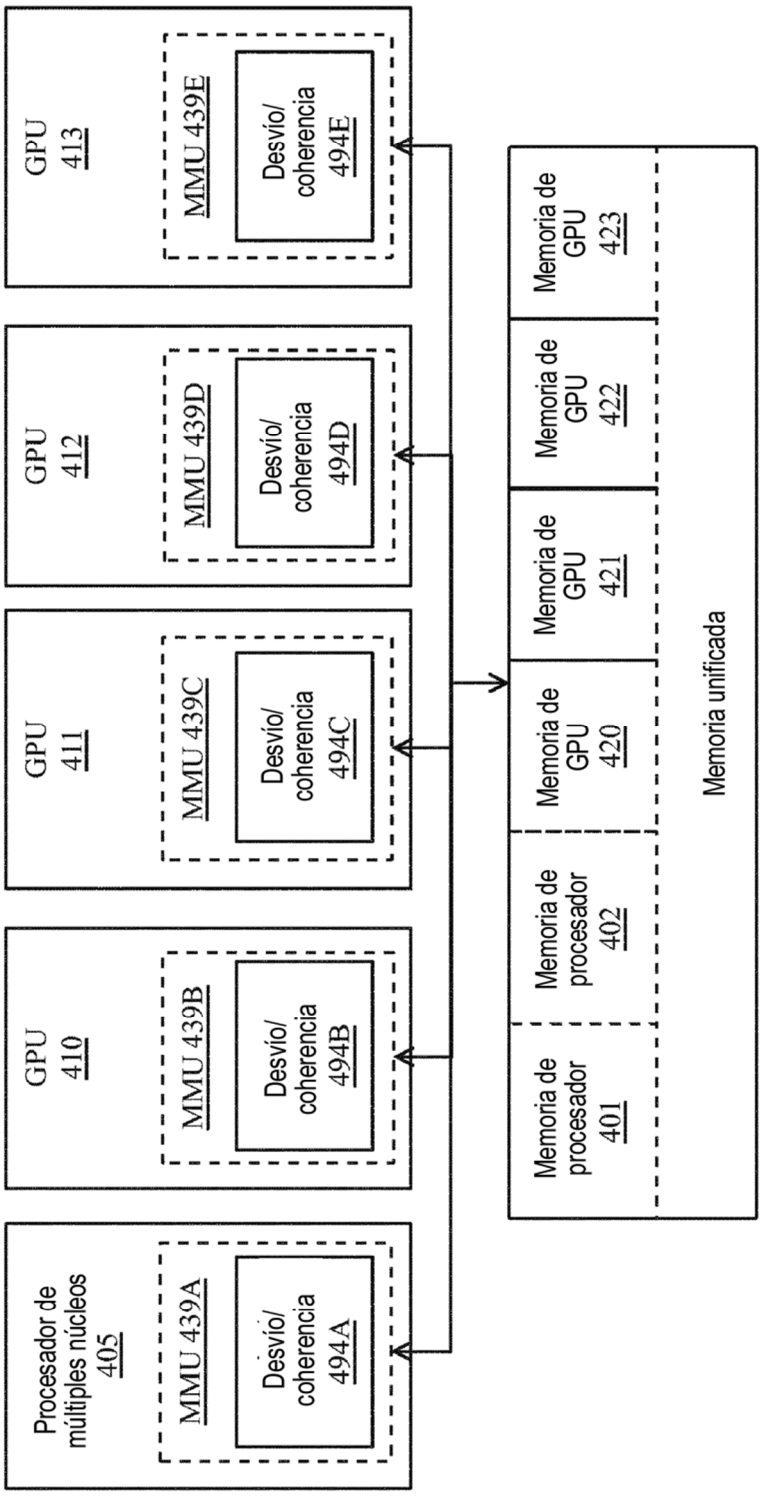

Procesador de múltiples núcleos 405
MMU 439A
Desvío/ coherencia 494A
GPU 410
MMU 439B
Desvío/ coherencia 494B
GPU 411
MMU 439C
Desvío/ coherencia 494C
GPU 412
MMU 439D
Desvío/ coherencia 494D
GPU 413
MMU 439E
Desvío/ coherencia 494E
Memoria de procesador 401
Memoria de procesador 402
Memoria de GPU 420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

FIG. 4F

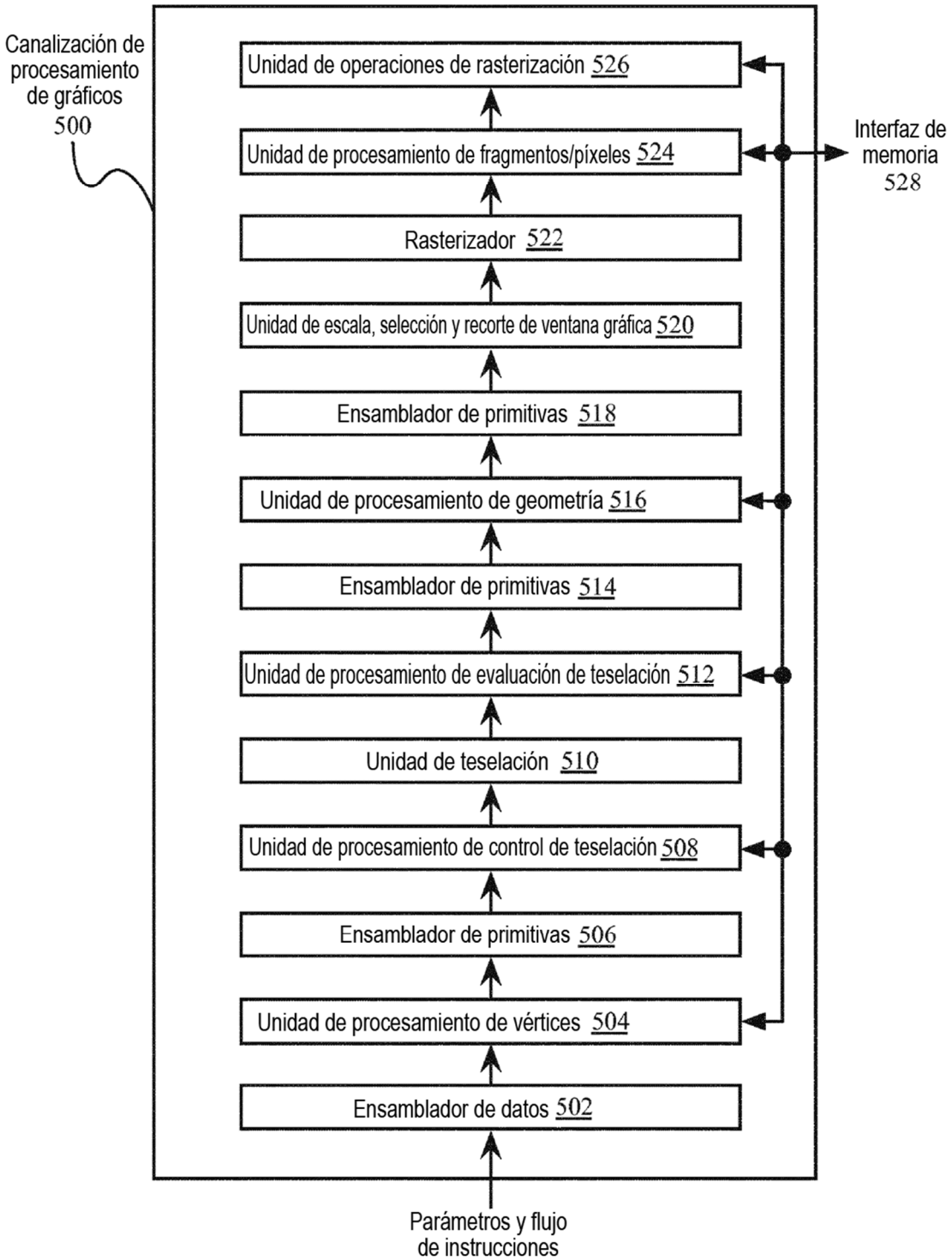

Canalización de procesamiento de gráficos 500
Unidad de operaciones de rasterización 526
Unidad de procesamiento de fragmentos/píxeles 524
Rasterizador 522
Unidad de escala, selección y recorte de ventana gráfica 520
Ensamblador de primitivas 518
Unidad de procesamiento de geometría 516
Ensamblador de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

FIG. 5

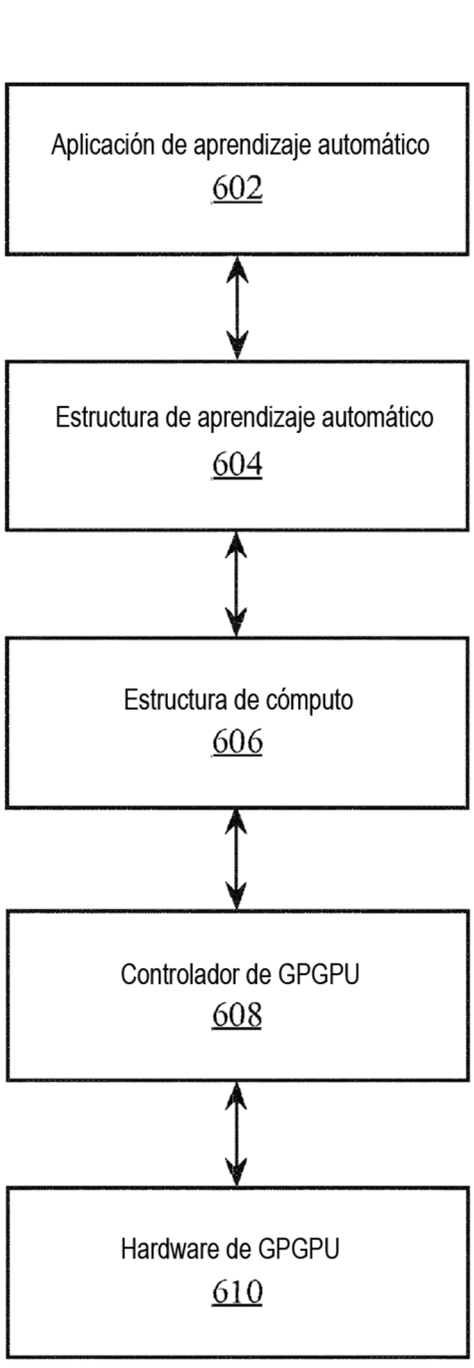

600
Aplicación de aprendizaje automático
602
Estructura de aprendizaje automático
604
Estructura de cómputo
606
Controlador de GPGPU
608
Hardware de GPGPU
610

FIG. 6

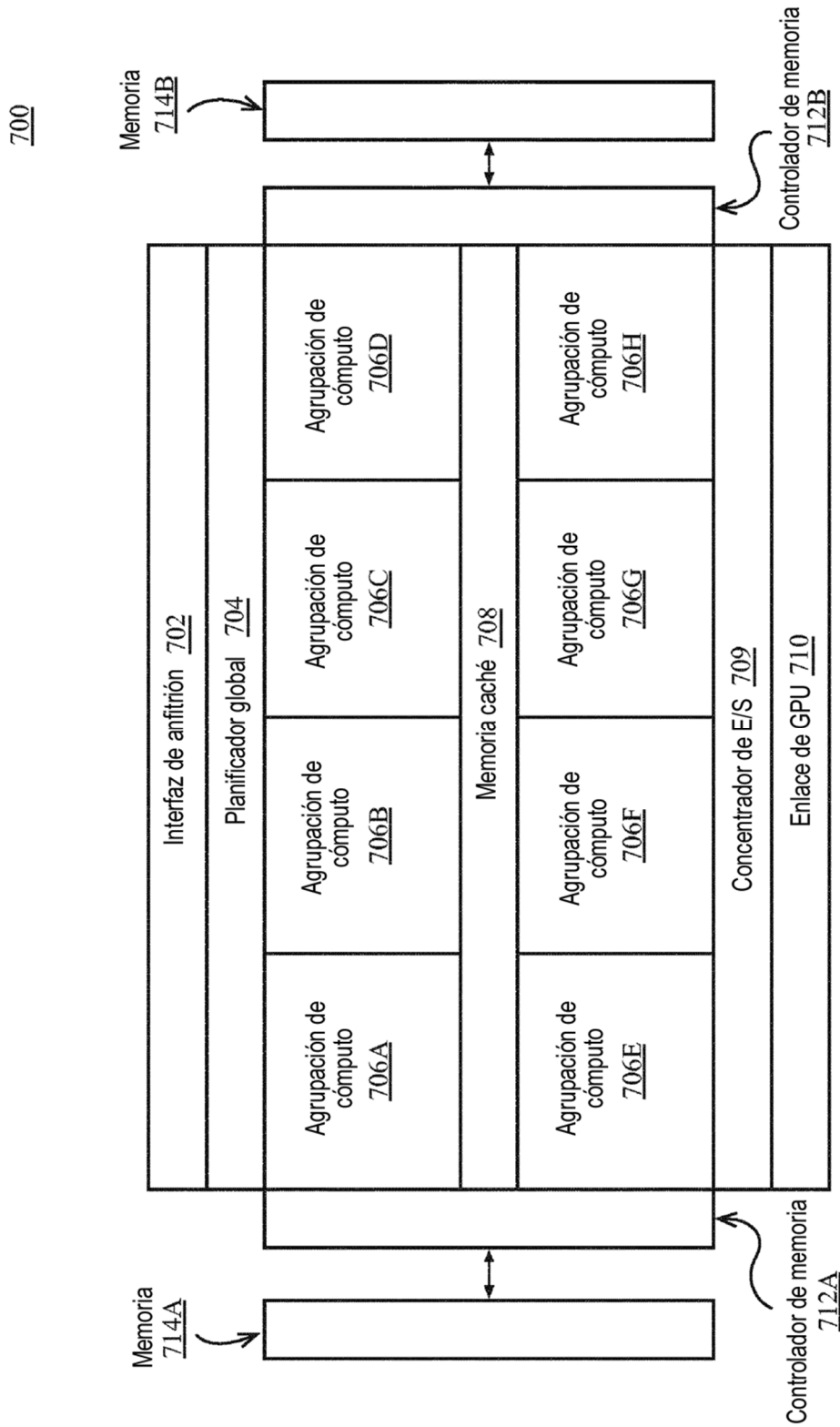

700
Memoria 714B
Controlador de memoria 712B
Interfaz de anfitrión 702
Planificador global 704
Agrupación de cómputo 706A
Agrupación de cómputo 706B
Agrupación de cómputo 706C
Agrupación de cómputo 706D
Memoria caché 708
Agrupación de cómputo 706E
Agrupación de cómputo 706F
Agrupación de cómputo 706G
Agrupación de cómputo 706H
Concentrador de E/S 709
Enlace de GPU 710
Memoria 714A
Controlador de memoria 712A

FIG. 7

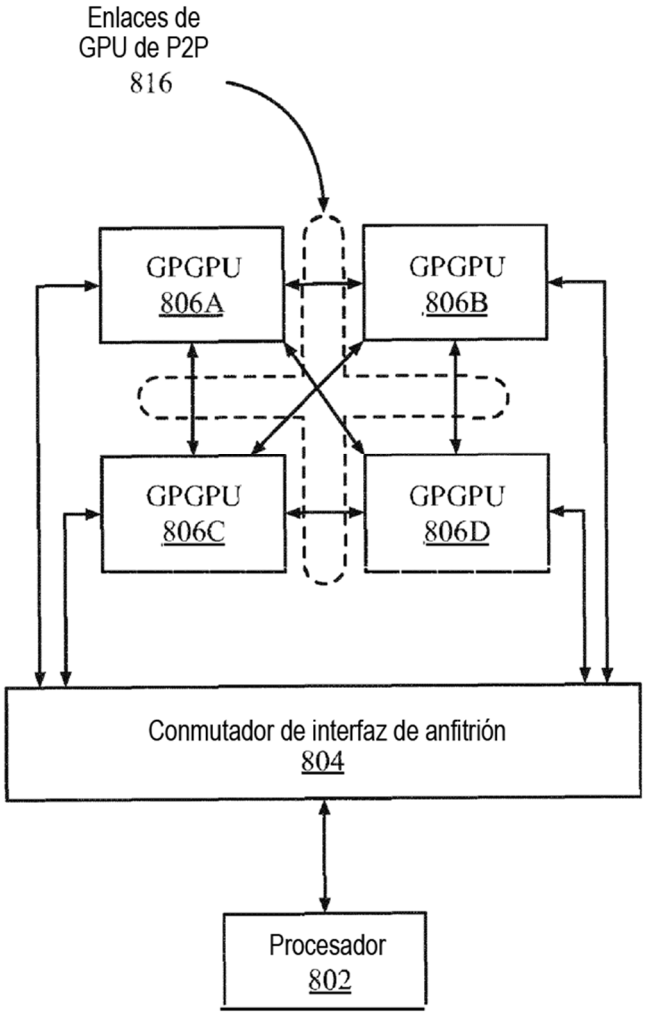

Enlaces de
GPU de P2P
816
GPGPU
806A
GPGPU
806B
GPGPU
806C
GPGPU
806D
Conmutador de interfaz de anfitrión
804
Procesador
802

FIG. 8

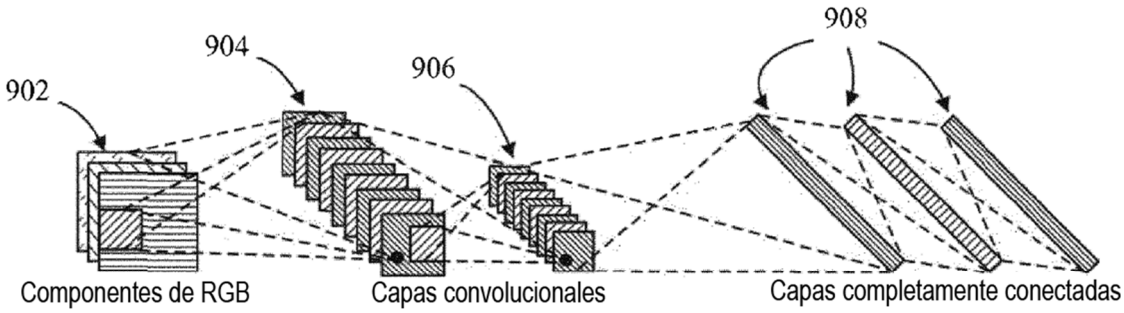

902
904
906
908
Componentes de RGB
Capas convolucionales
Capas completamente conectadas

FIG. 9A

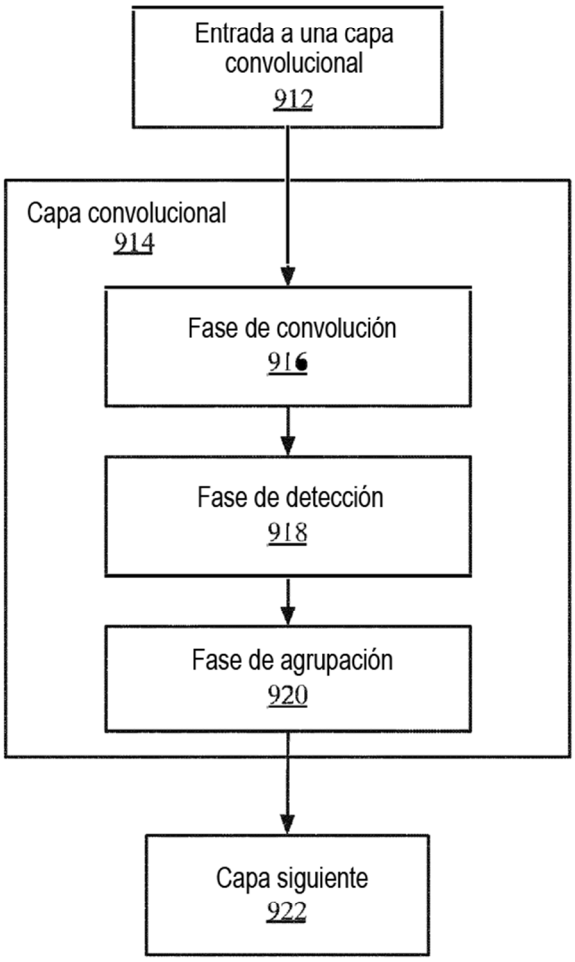

Entrada a una capa convolucional
912
Capa convolucional
914
Fase de convolución
916
Fase de detección
918
Fase de agrupación
920
Capa siguiente
922

FIG. 9B

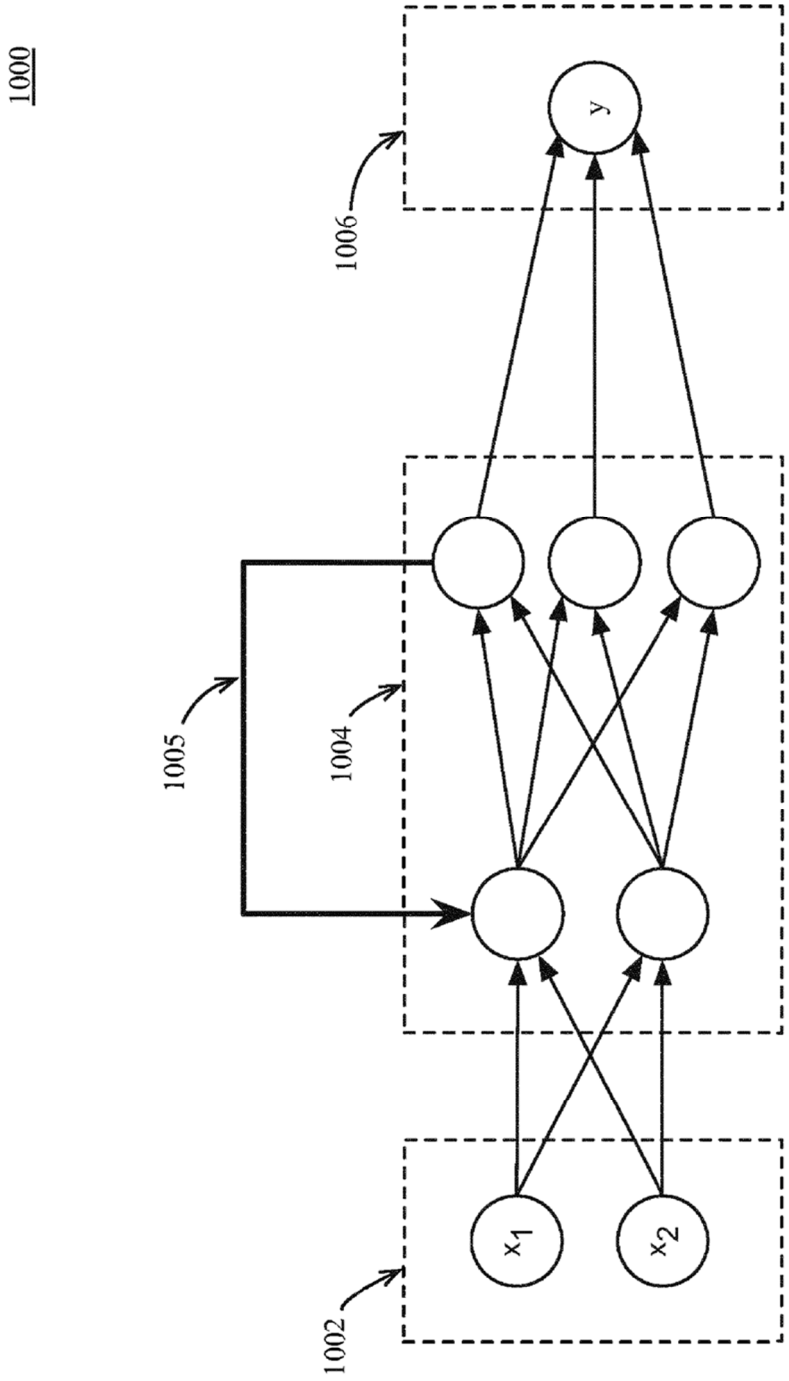


FIG. 10

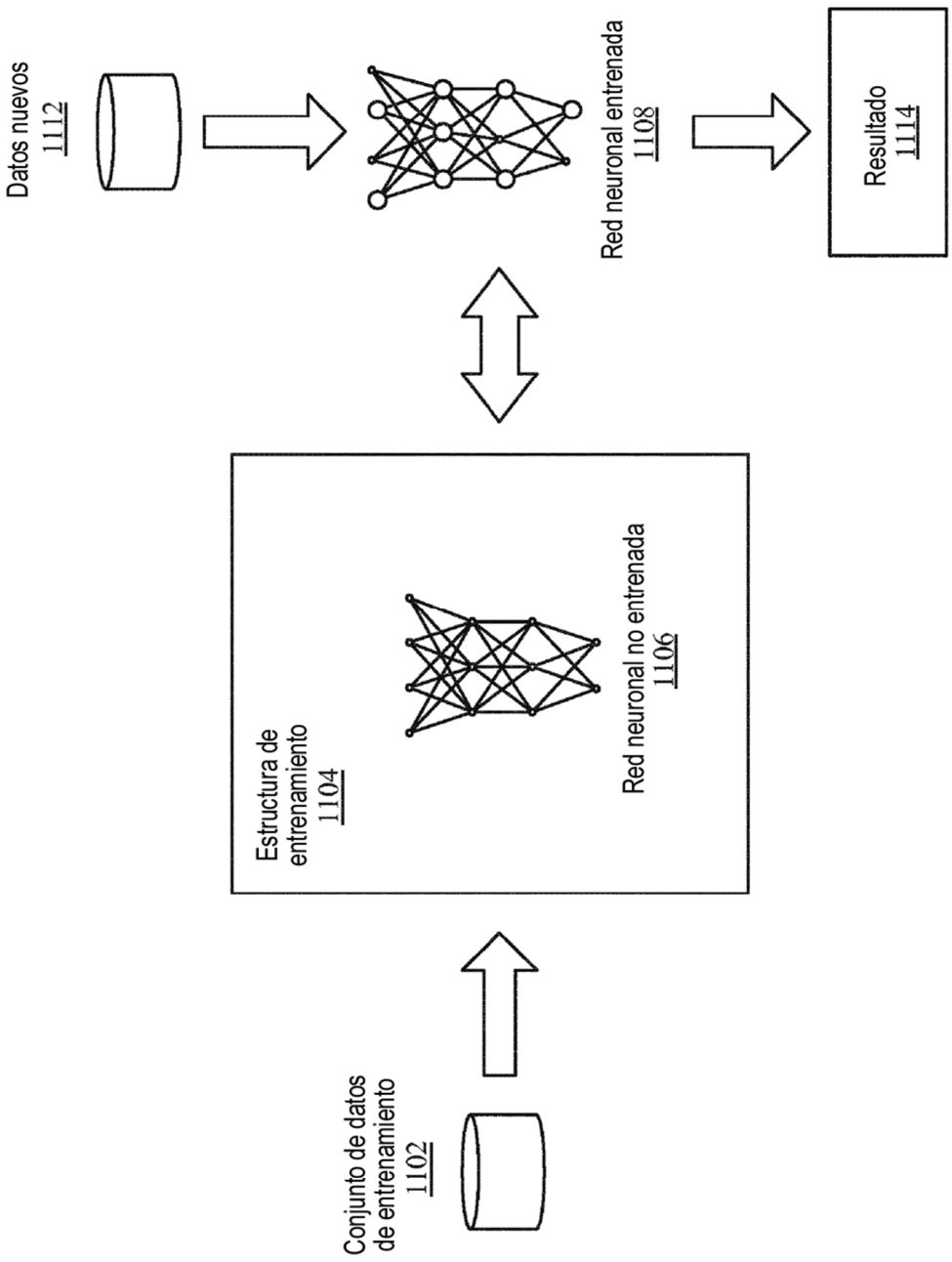

Datos nuevos
1112
Red neuronal entrenada
1108
Resultado
1114
Estructura de entrenamiento
1104
Red neuronal no entrenada
1106
Conjunto de datos de entrenamiento
1102

FIG. 11

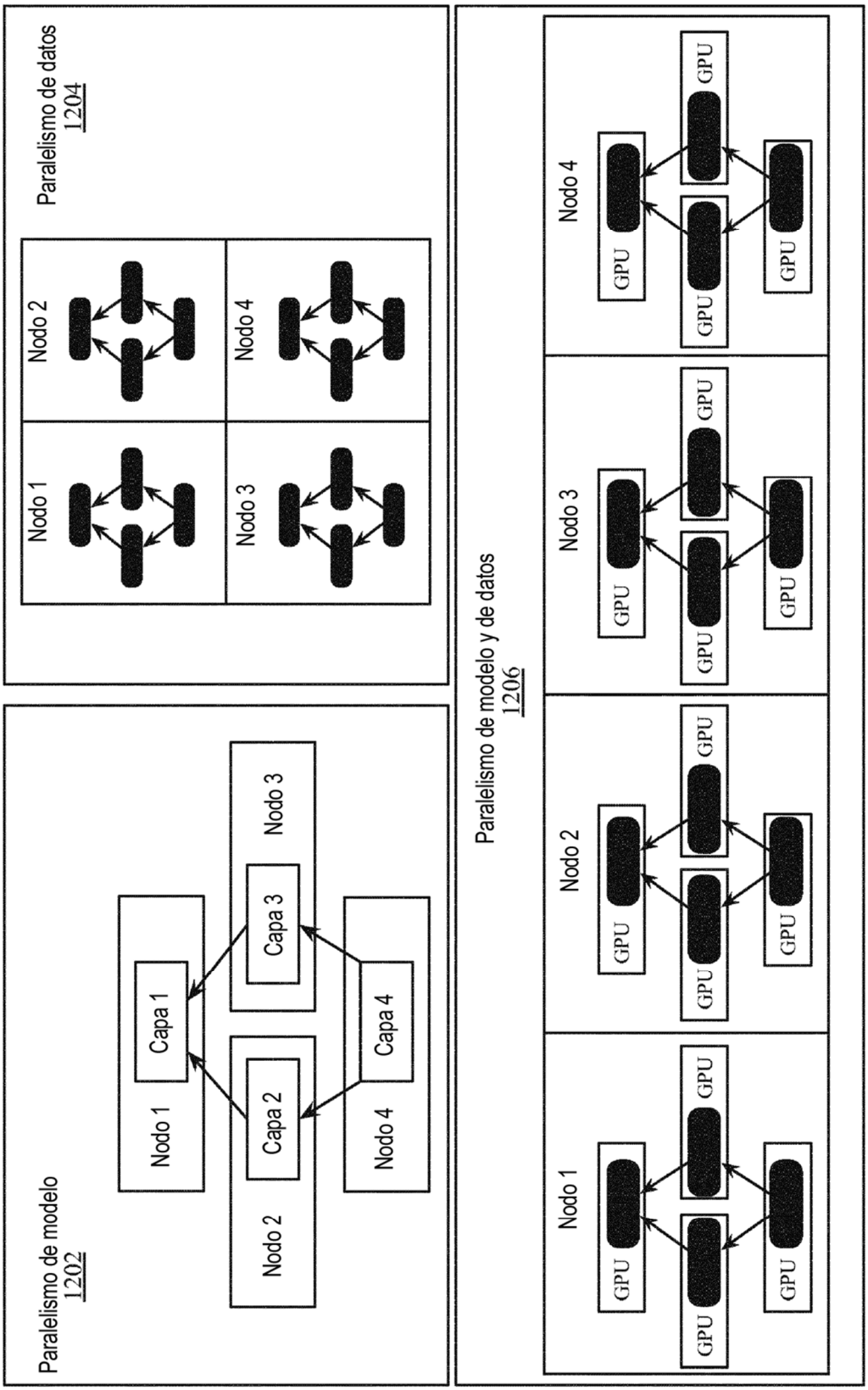

Paralelismo de modelo
1202
Nodo 1
Capa 1
Nodo 2
Capa 2
Nodo 3
Capa 3
Nodo 4
Capa 4
Paralelismo de datos
1204
Nodo 1
Nodo 2
Nodo 3
Nodo 4
Paralelismo de modelo y de datos
1206
GPU

FIG. 12

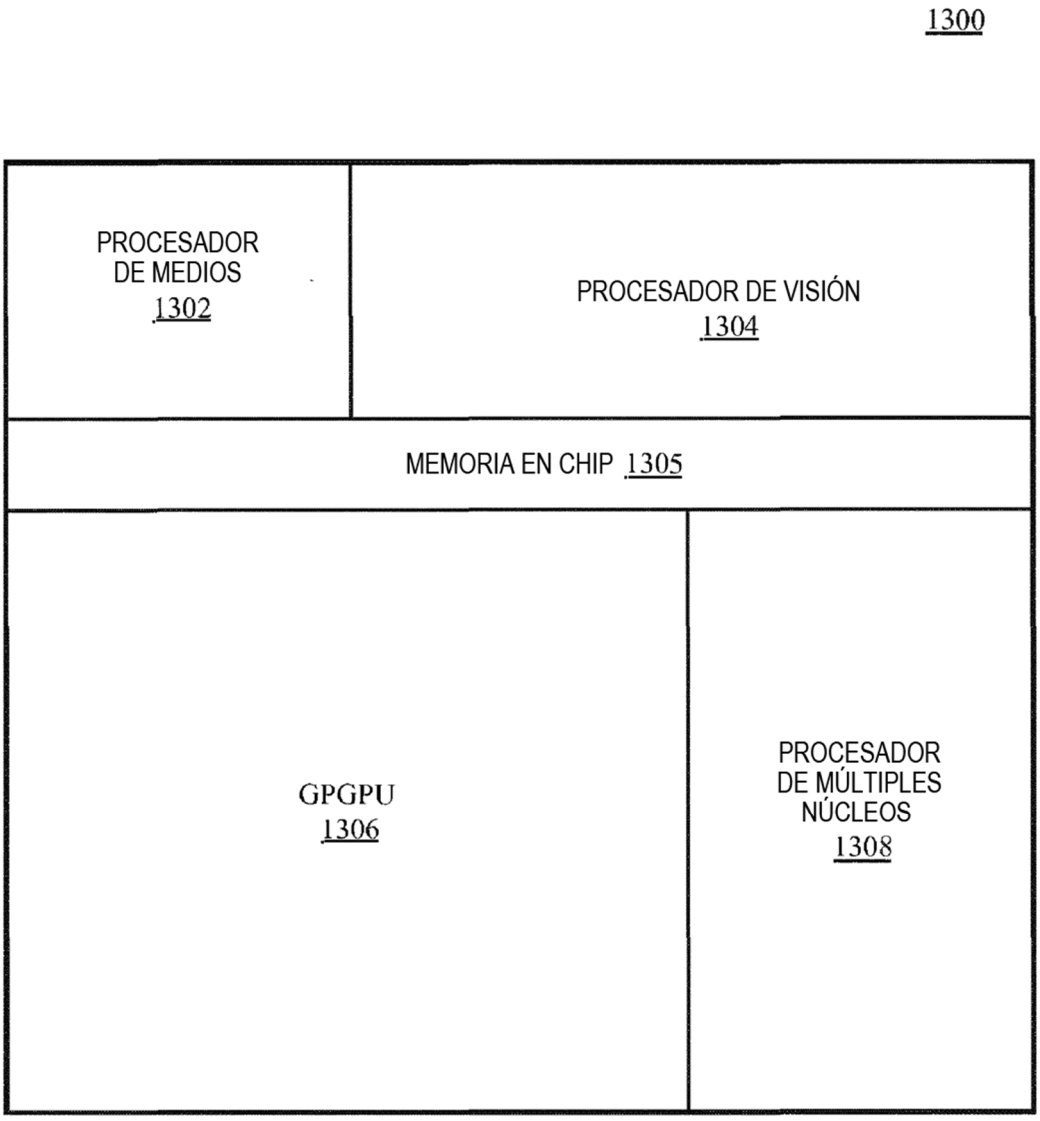

1300
PROCESADOR DE MEDIOS 1302
PROCESADOR DE VISIÓN 1304
MEMORIA EN CHIP 1305
GPGPU 1306
PROCESADOR DE MÚLTIPLES NÚCLEOS 1308

FIG. 13

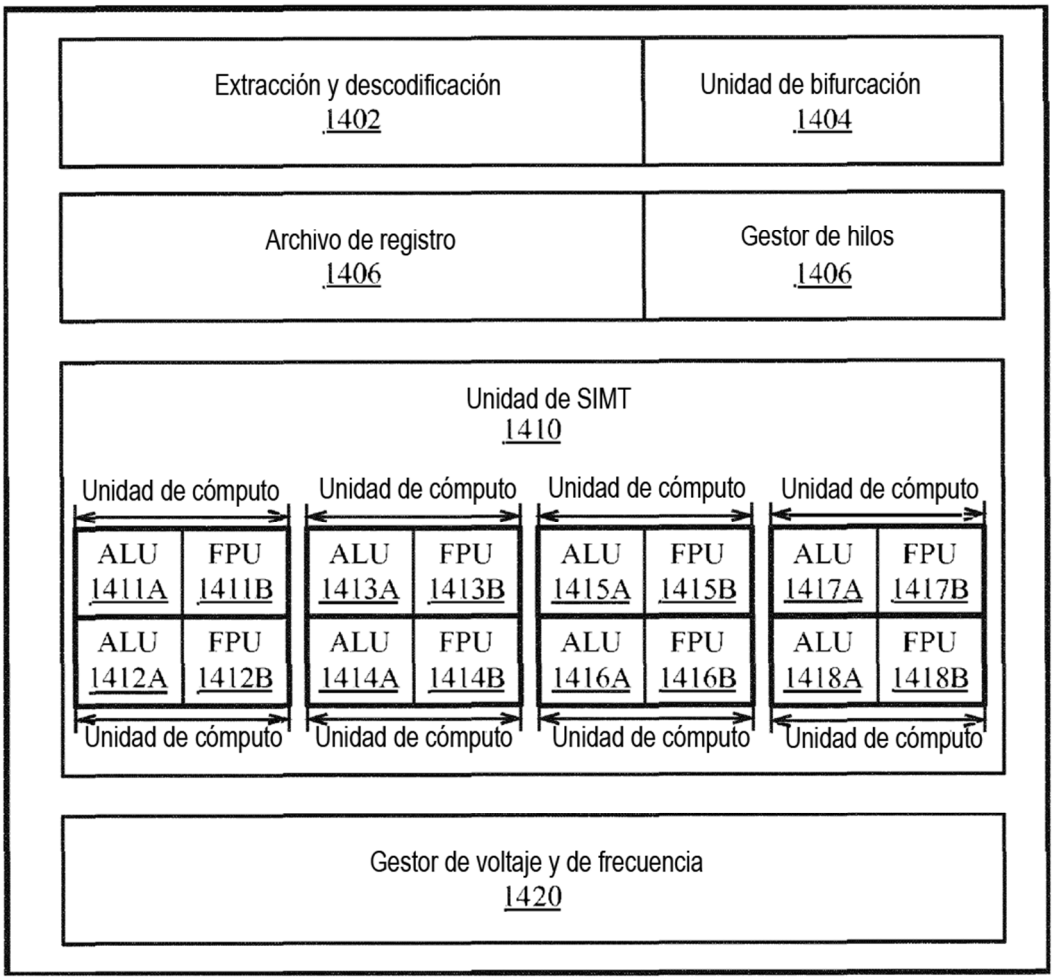

1400
Extracción y descodificación
1402
Unidad de bifurcación
1404
Archivo de registro
1406
Gestor de hilos
1406
Unidad de SIMT
1410
Unidad de cómputo
Unidad de cómputo
Unidad de cómputo
Unidad de cómputo
ALU 1411A
FPU 1411B
ALU 1412A
FPU 1412B
ALU 1413A
FPU 1413B
ALU 1414A
FPU 1414B
ALU 1415A
FPU 1415B
ALU 1416A
FPU 1416B
ALU 1417A
FPU 1417B
ALU 1418A
FPU 1418B
Unidad de cómputo
Unidad de cómputo
Unidad de cómputo
Unidad de cómputo
Gestor de voltaje y de frecuencia
1420

FIG. 14

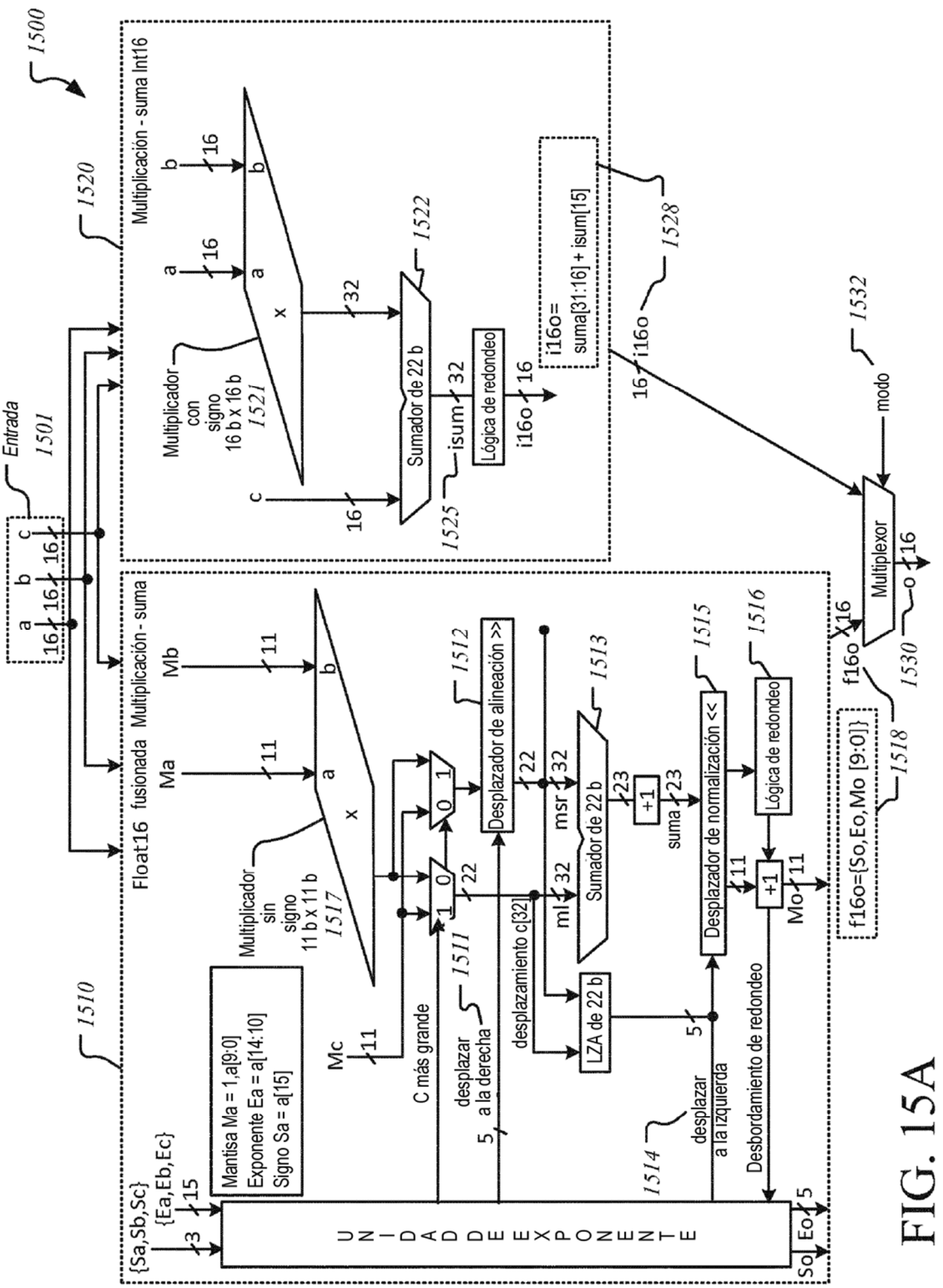

1500
1520
Multiplicación - suma Int16
Entrada
1501
a b c
16 16 16
Multiplicador con signo 16 b x 16 b
1521
a b
16 16
x
32
Sumador de 22 b
1522
isum 32
Lógica de redondeo
i16o 16
i16o= suma[31:16] + isum[15]
1528
1525
c
16
16 i16o
1532
modo
Multiplexor
o 16
f16o 16
1530
1518
f16o={So,Eo,Mo [9:0]}
1510
Float16 fusionada Multiplicación - suma
Ma
11
Mb
11
Multiplicador sin signo 11 b x 11 b
1517
a b
x
1512
Desplazador de alineación >>
22
msr 32
1513
Sumador de 22 b
23
+1
suma 23
1515
Desplazador de normalización <<
11
1516
Lógica de redondeo
+1
Mo 11
1 0
0 1
22
ml 32
Mc
11
C más grande
desplazar 5 a la derecha
1511
desplazamiento c[32]
LZA de 22 b
5
1514
desplazar a la izquierda
Desbordamiento de redondeo
{Sa,Sb,Sc}
3
{Ea,Eb,Ec}
15
Mantisa Ma = 1,a[9:0]
Exponente Ea = a[14:10]
Signo Sa = a[15]
UNIDAD DE EXPONENTE
So Eo 5

FIG. 15A

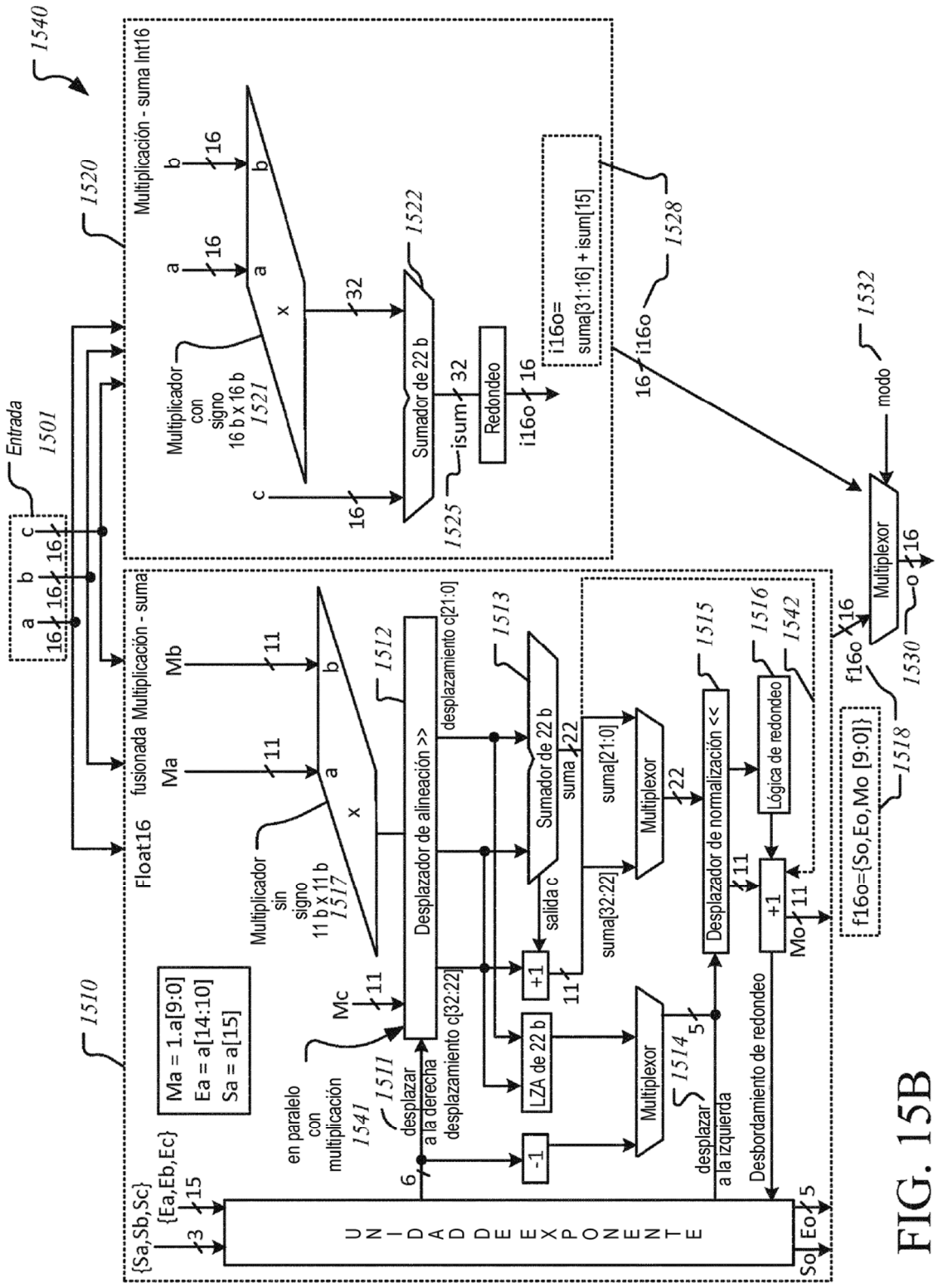

1540
1520
Multiplicación - suma Int16
Entrada
1501
a b c
16 16 16
Multiplicador con signo 16 b x 16 b
1521
a b
16 16
x
32
Sumador de 22 b
1522
isum
32
Redondeo
16
i16o
i16o= suma[31:16] + isum[15]
1528
16
i16o
1525
c
16
1532
modo
1510
Float16 fusionada Multiplicación - suma
Ma Mb
11 11
Multiplicador sin signo 11 b x 11 b
1517
a b
x
1512
Desplazador de alineación >>
desplazamiento c[21:0]
1513
Sumador de 22 b
suma
22
salida c
+1
11
suma[21:0]
suma[32:22]
Multiplexor
22
1515
Desplazador de normalización <<
11
1516
1542
Lógica de redondeo
+1
11
Mo
Mc
11
Ma = 1.a[9:0]
Ea = a[14:10]
Sa = a[15]
en paralelo con multiplicación
1541
1511
desplazar a la derecha
desplazamiento c[32:22]
6
-1
LZA de 22 b
Multiplexor
1514
5
desplazar a la izquierda
Desbordamiento de redondeo
{Sa,Sb,Sc}
3
{Ea,Eb,Ec}
15
UNIDAD DE EXPONENTE
So Eo
5
f16o={So,Eo,Mo [9:0]}
1518
f16o
16
Multiplexor
o
16
1530

FIG. 15B

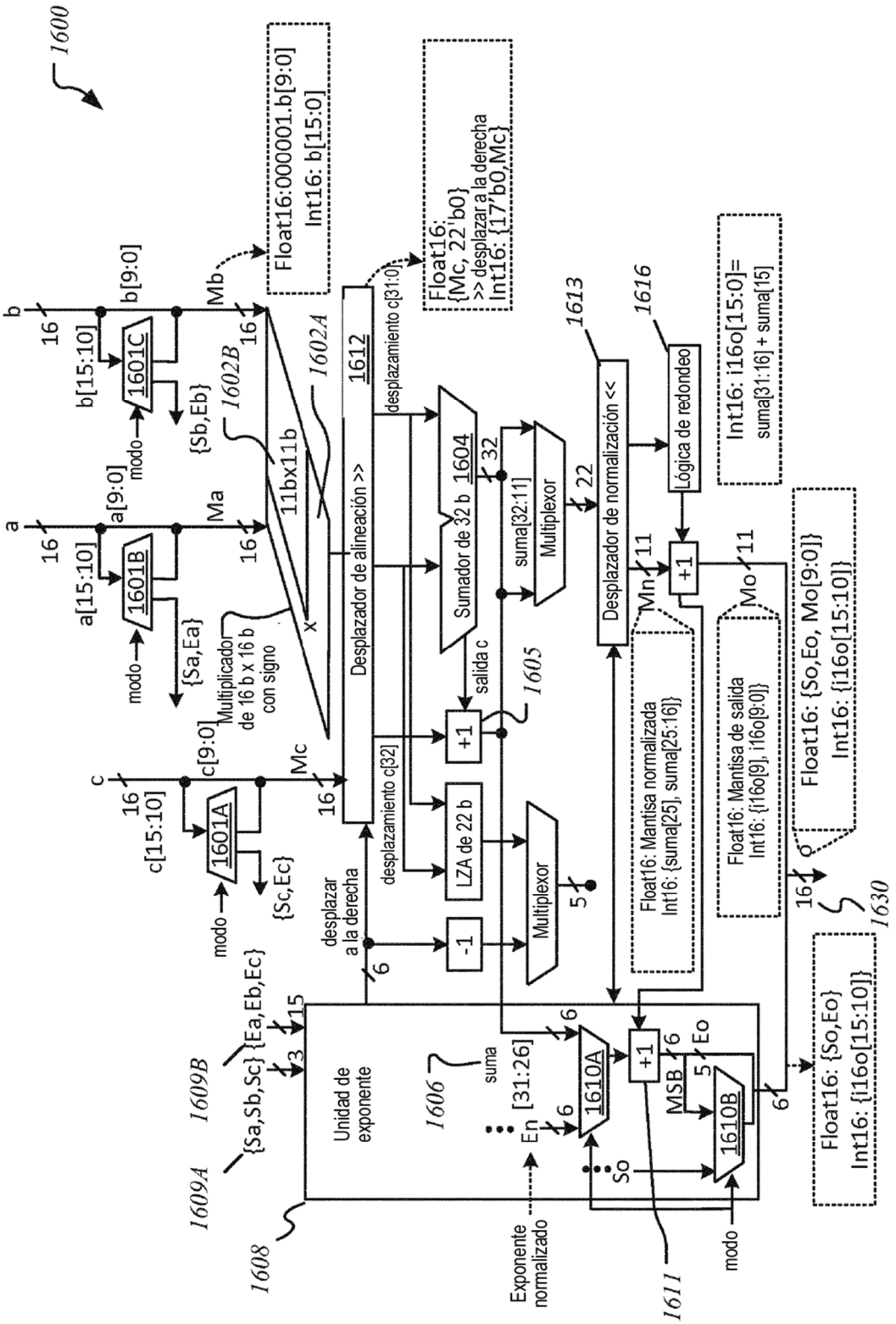

1600
a
16
a[15:10]
1601B
a[9:0]
modo
{Sa,Ea}
Ma
16
b
16
b[15:10]
1601C
b[9:0]
modo
{Sb,Eb}
Mb
16
c
16
c[15:10]
1601A
c[9:0]
modo
{Sc,Ec}
Mc
16
Multiplicador de 16 b x 16 b con signo
11bx11b
1602A
1602B
Float16:000001.b[9:0]
Int16: b[15:0]
1612
desplazamiento c[31:0]
Float16:
{Mc, 22'b0}
>> desplazar a la derecha
Int16: {17'b0,Mc}
Desplazador de alineación >>
desplazamiento c[32]
desplazar a la derecha
6
Sumador de 32 b 1604
32
salida c
1605
+1
-1
LZA de 22 b
Multiplexor
5
suma[32:11]
Multiplexor
22
1613
Desplazador de normalización <<
1616
Lógica de redondeo
Mn
11
+1
Mo
11
Float16: Mantisa normalizada
Int16: {suma[25], suma[25:16]}
Float16: Mantisa de salida
Int16: {i16o[9], i16o[9:0]}
Int16: i16o[15:0]=
suma[31:16] + suma[15]
16
Float16: {So,Eo, Mo[9:0]}
Int16: {i16o[15:10]}
1630
Float16: {So,Eo}
Int16: {i16o[15:10]}
1609A
1609B
{Sa,Sb,Sc}
{Ea,Eb,Ec}
3
15
Unidad de exponente
1608
1606
suma
En [31:26]
6
1610A
+1
6
MSB
6
Eo
5
So
1610B
6
Exponente normalizado
1611
modo

FIG. 16

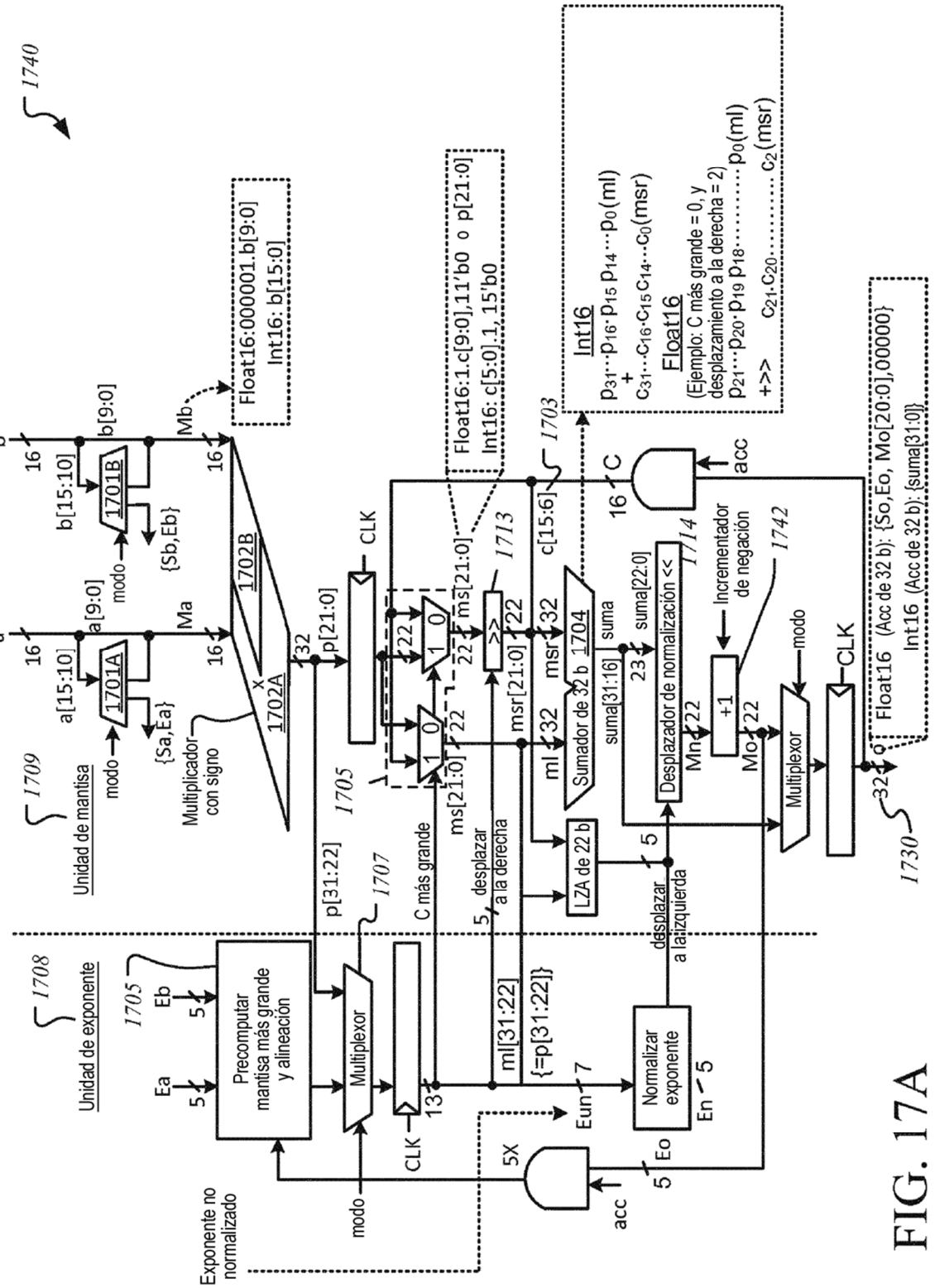

1740
1709
Unidad de mantisa
1708
Unidad de exponente
1705
1707
1703
1713
1714
1742
1730
a
b
a[15:10]
a[9:0]
b[15:10]
b[9:0]
modo
1701A
1701B
{Sa,Ea}
{Sb,Eb}
Ma
Mb
Multiplicador con signo
1702A
1702B
Float16:000001.b[9:0]
Int16: b[15:0]
Float16:1.c[9:0],11'b0 o p[21:0]
Int16: c[5:0],1, 15'b0
p[21:0]
p[31:22]
CLK
C más grande
ms[21:0]
msr[21:0]
ml[31:22]
{=p[31:22]}
desplazar a la derecha
desplazar a la izquierda
Precomputar mantisa más grande y alineación
Multiplexor
Normalizar exponente
Exponente no normalizado
Ea
Eb
Eun
En
Eo
5X
acc
LZA de 22 b
Sumador de 32 b
suma
suma[31:16]
suma[22:0]
Desplazador de normalización <<
Mn
Mo
+1
Incrementador de negación
Multiplexor
c[15:6]
ml
msr
C
Int16
p31···p16·p15 p14···p0(ml)
+
c31···c16·c15 c14···c0(msr)
Float16
(Ejemplo: C más grande = 0, y desplazamiento a la derecha = 2)
p21···p20·p19 p18·····p0(ml)
+>> c21·c20·····c2(msr)
Float16 (Acc de 32 b): {So,Eo, Mo[20:0],00000}
Int16 (Acc de 32 b): {suma[31:0]}

FIG. 17A

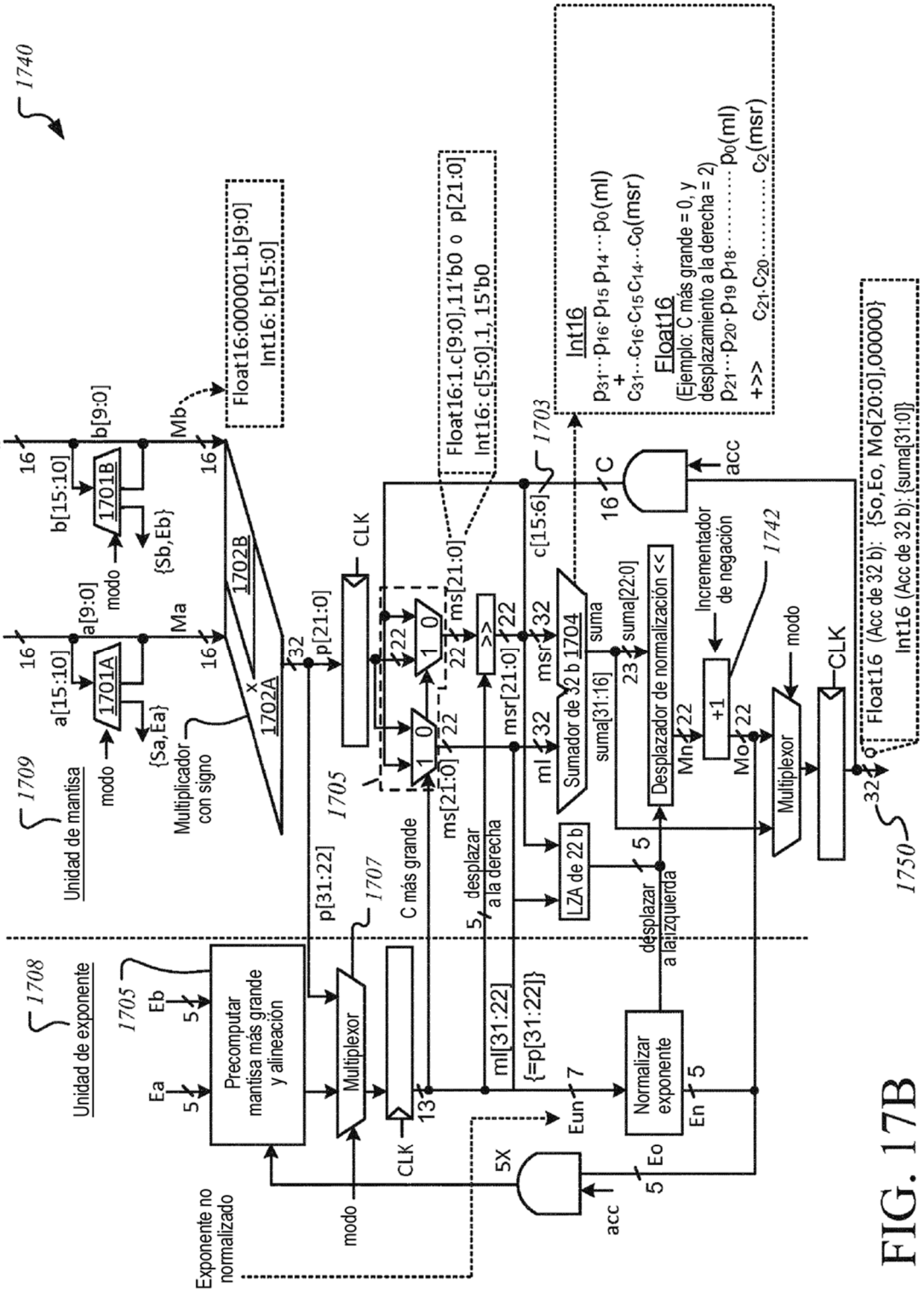

1740
1709
Unidad de mantisa
1708
Unidad de exponente
1705
1707
1703
1742
1750
a
b
16
a[15:10]
b[15:10]
a[9:0]
b[9:0]
modo
1701A
1701B
{Sa,Ea}
{Sb,Eb}
Ma
Mb
Multiplicador con signo
1702A
1702B
x
32
p[21:0]
p[31:22]
CLK
Float16:000001.b[9:0]
Int16: b[15:0]
Float16:1.c[9:0],11'b0 o p[21:0]
Int16: c[5:0],1, 15'b0
ms[21:0]
msr[21:0]
ml[31:22]
{=p[31:22]}
C más grande
desplazar a la derecha
desplazar a la izquierda
Precomputar mantisa más grande y alineación
Multiplexor
Ea
Eb
5X
Eun
En
Eo
Normalizar exponente
Exponente no normalizado
LZA de 22 b
Sumador de 32 b 1704
suma
suma[31:16]
suma[22:0]
c[15:6]
ml
msr
C
acc
Desplazador de normalización <<
Incrementador de negación
Mn
Mo
+1
Multiplexor
Int16
Float16
(Ejemplo: C más grande = 0, y desplazamiento a la derecha = 2)
+>>
Float16 (Acc de 32 b): {So,Eo, Mo[20:0],00000}
Int16 (Acc de 32 b): {suma[31:0]}

FIG. 17B

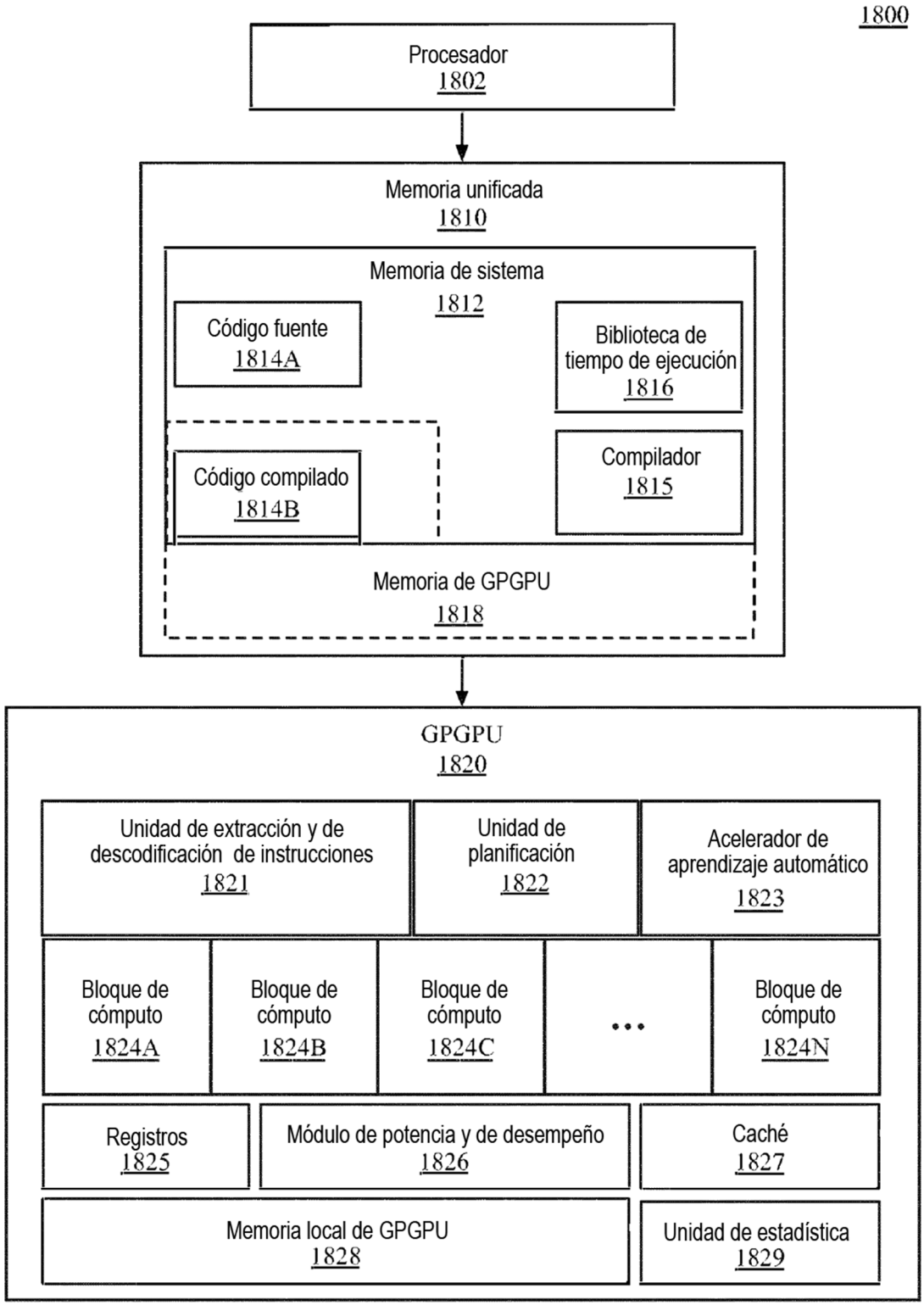

1800
Procesador
1802
Memoria unificada
1810
Memoria de sistema
1812
Código fuente
1814A
Biblioteca de tiempo de ejecución
1816
Código compilado
1814B
Compilador
1815
Memoria de GPGPU
1818
GPGPU
1820
Unidad de extracción y de descodificación de instrucciones
1821
Unidad de planificación
1822
Acelerador de aprendizaje automático
1823
Bloque de cómputo
1824A
Bloque de cómputo
1824B
Bloque de cómputo
1824C
...
Bloque de cómputo
1824N
Registros
1825
Módulo de potencia y de desempeño
1826
Caché
1827
Memoria local de GPGPU
1828
Unidad de estadística
1829

FIG. 18A

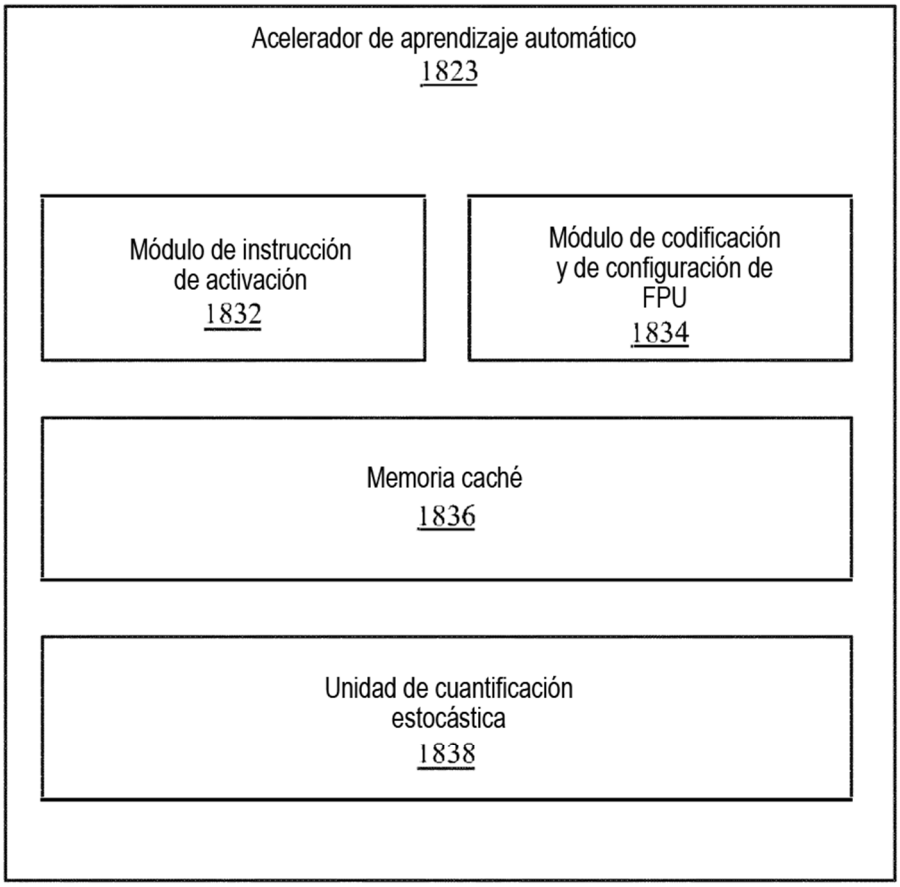

Acelerador de aprendizaje automático
1823
Módulo de instrucción de activación
1832
Módulo de codificación y de configuración de FPU
1834
Memoria caché
1836
Unidad de cuantificación estocástica
1838

FIG. 18B

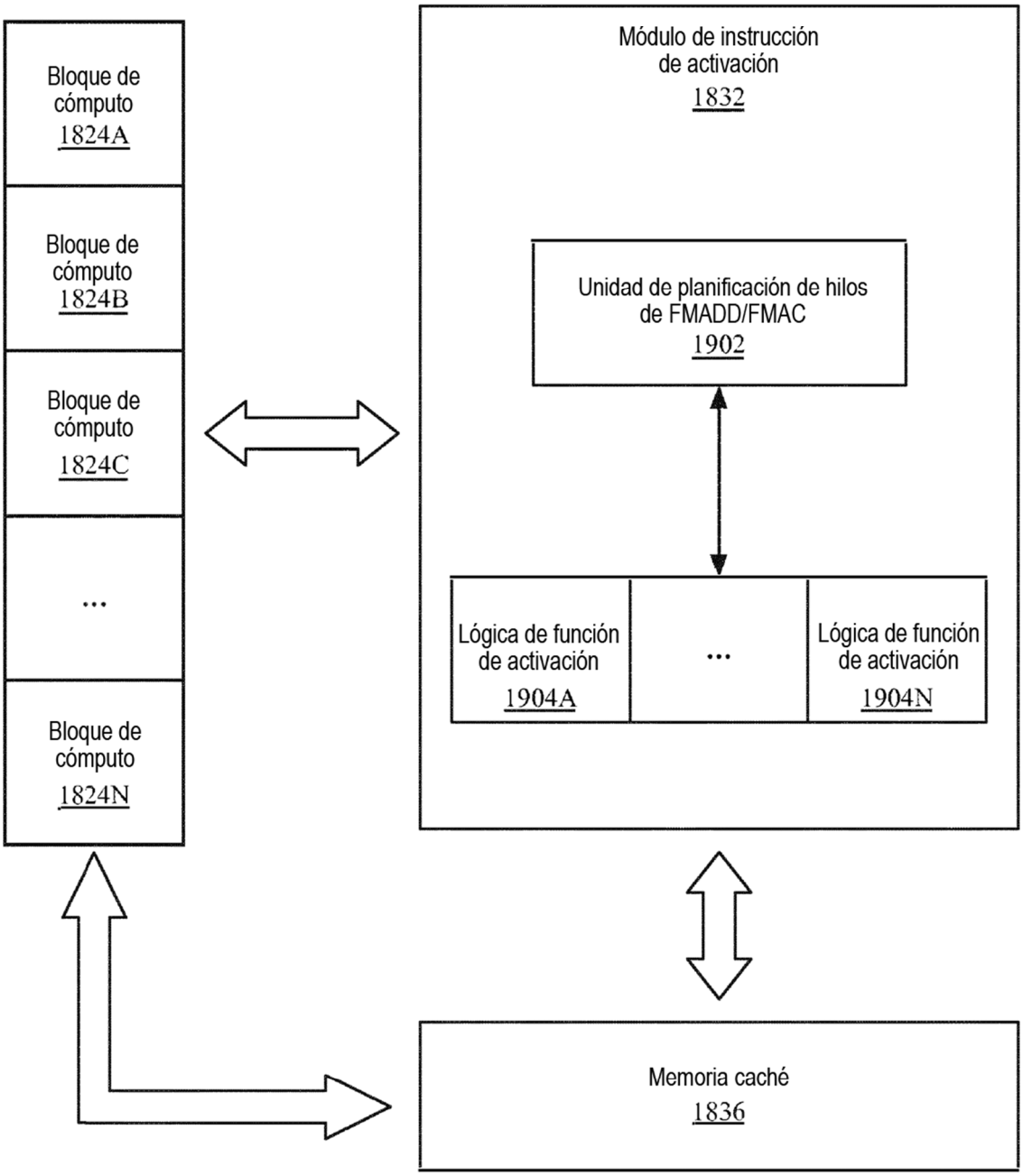

Bloque de cómputo 1824A
Bloque de cómputo 1824B
Bloque de cómputo 1824C
...
Bloque de cómputo 1824N
Módulo de instrucción de activación 1832
Unidad de planificación de hilos de FMADD/FMAC 1902
Lógica de función de activación 1904A
...
Lógica de función de activación 1904N
Memoria caché 1836

FIG. 19

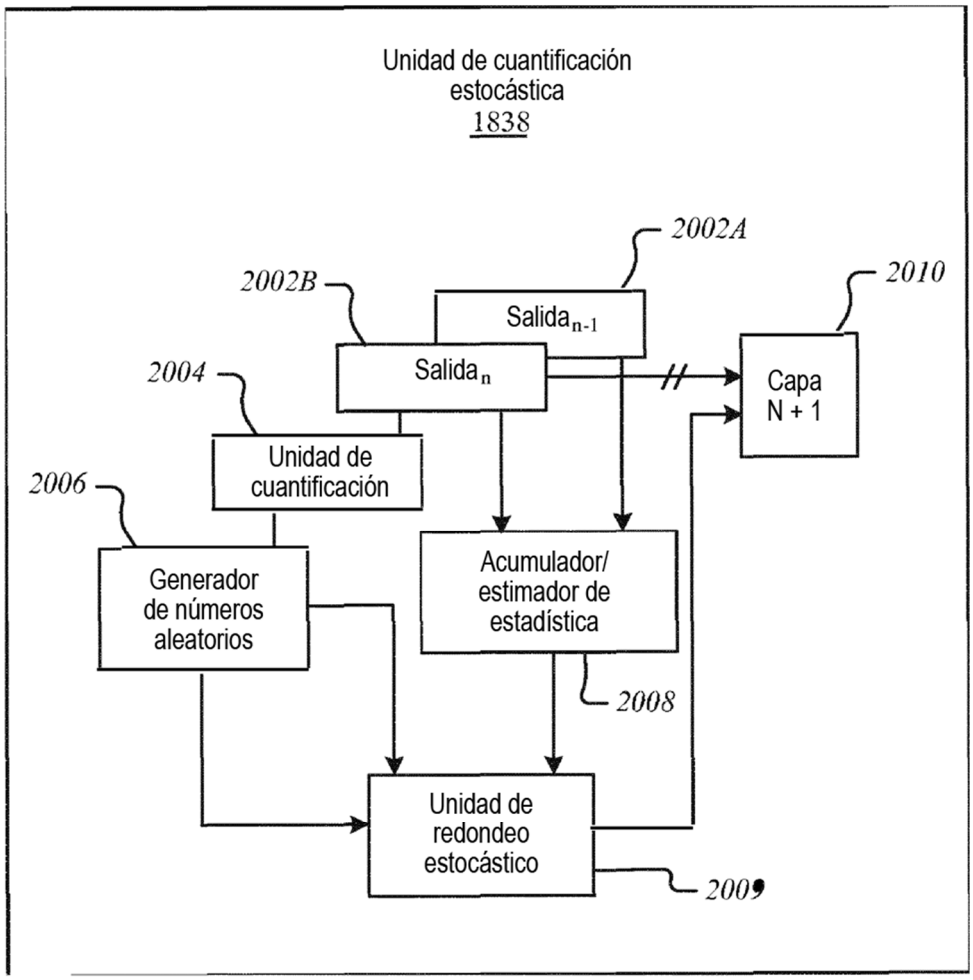

Unidad de cuantificación estocástica
1838
2002A
2002B
2010
Salida n-1
Salida n
2004
Capa N + 1
Unidad de cuantificación
2006
Generador de números aleatorios
Acumulador/ estimador de estadística
2008
Unidad de redondeo estocástico
2009

FIG. 20

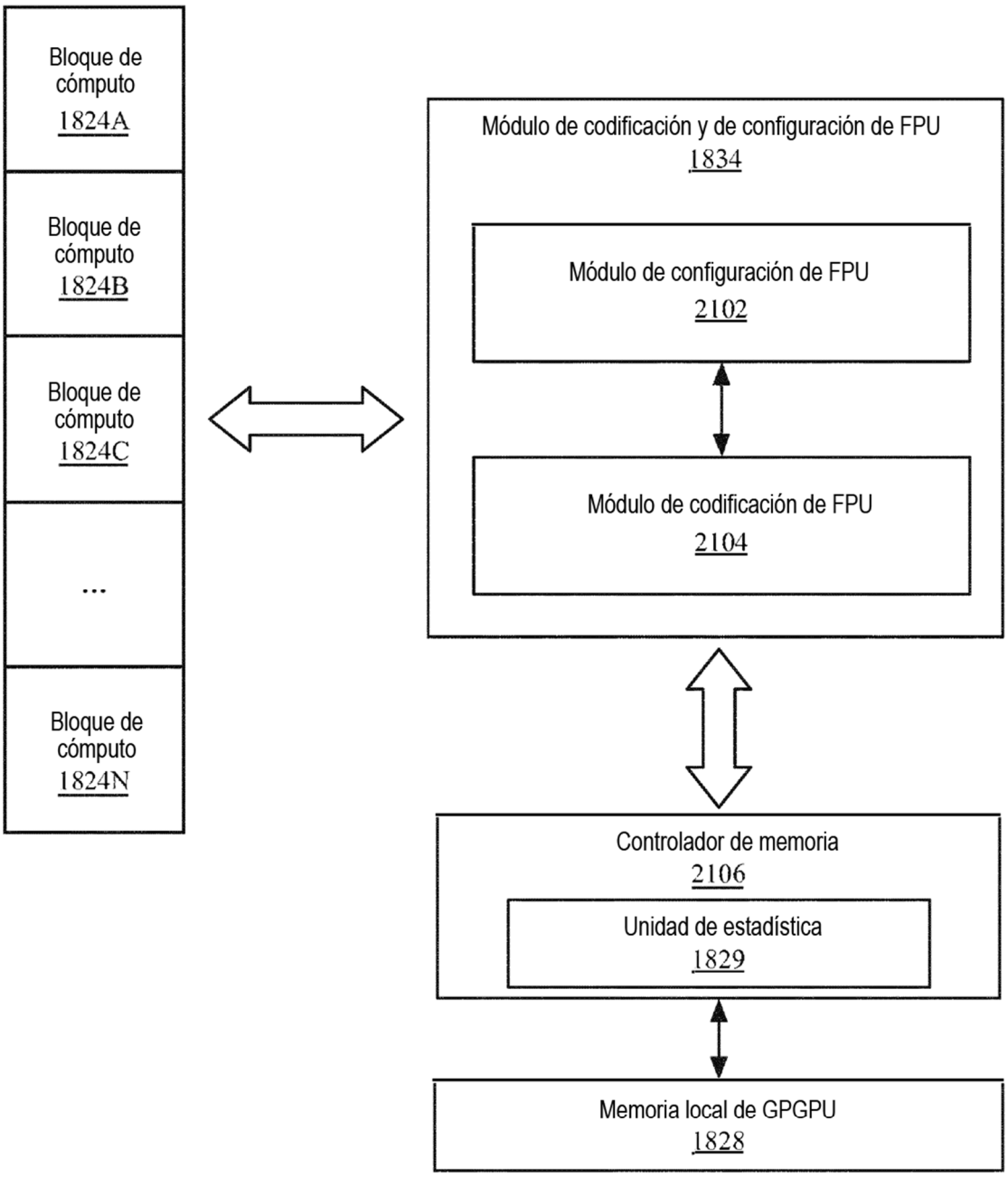

Bloque de cómputo
1824A
Bloque de cómputo
1824B
Bloque de cómputo
1824C
...
Bloque de cómputo
1824N
Módulo de codificación y de configuración de FPU
1834
Módulo de configuración de FPU
2102
Módulo de codificación de FPU
2104
Controlador de memoria
2106
Unidad de estadística
1829
Memoria local de GPGPU
1828

FIG. 21

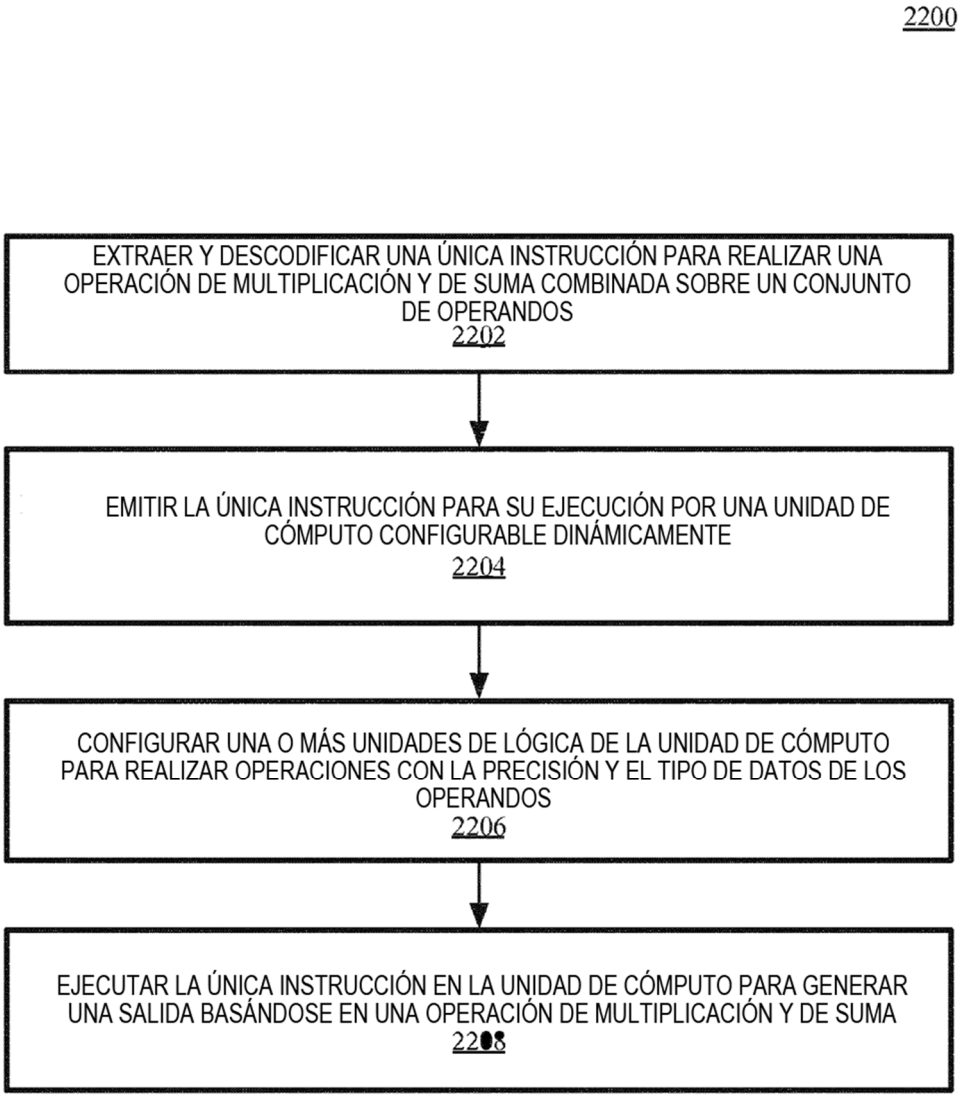

2200
EXTRAER Y DESCODIFICAR UNA ÚNICA INSTRUCCIÓN PARA REALIZAR UNA OPERACIÓN DE MULTIPLICACIÓN Y DE SUMA COMBINADA SOBRE UN CONJUNTO DE OPERANDOS
2202
EMITIR LA ÚNICA INSTRUCCIÓN PARA SU EJECUCIÓN POR UNA UNIDAD DE CÓMPUTO CONFIGURABLE DINÁMICAMENTE
2204
CONFIGURAR UNA O MÁS UNIDADES DE LÓGICA DE LA UNIDAD DE CÓMPUTO PARA REALIZAR OPERACIONES CON LA PRECISIÓN Y EL TIPO DE DATOS DE LOS OPERANDOS
2206
EJECUTAR LA ÚNICA INSTRUCCIÓN EN LA UNIDAD DE CÓMPUTO PARA GENERAR UNA SALIDA BASÁNDOSE EN UNA OPERACIÓN DE MULTIPLICACIÓN Y DE SUMA
2208

FIG. 22

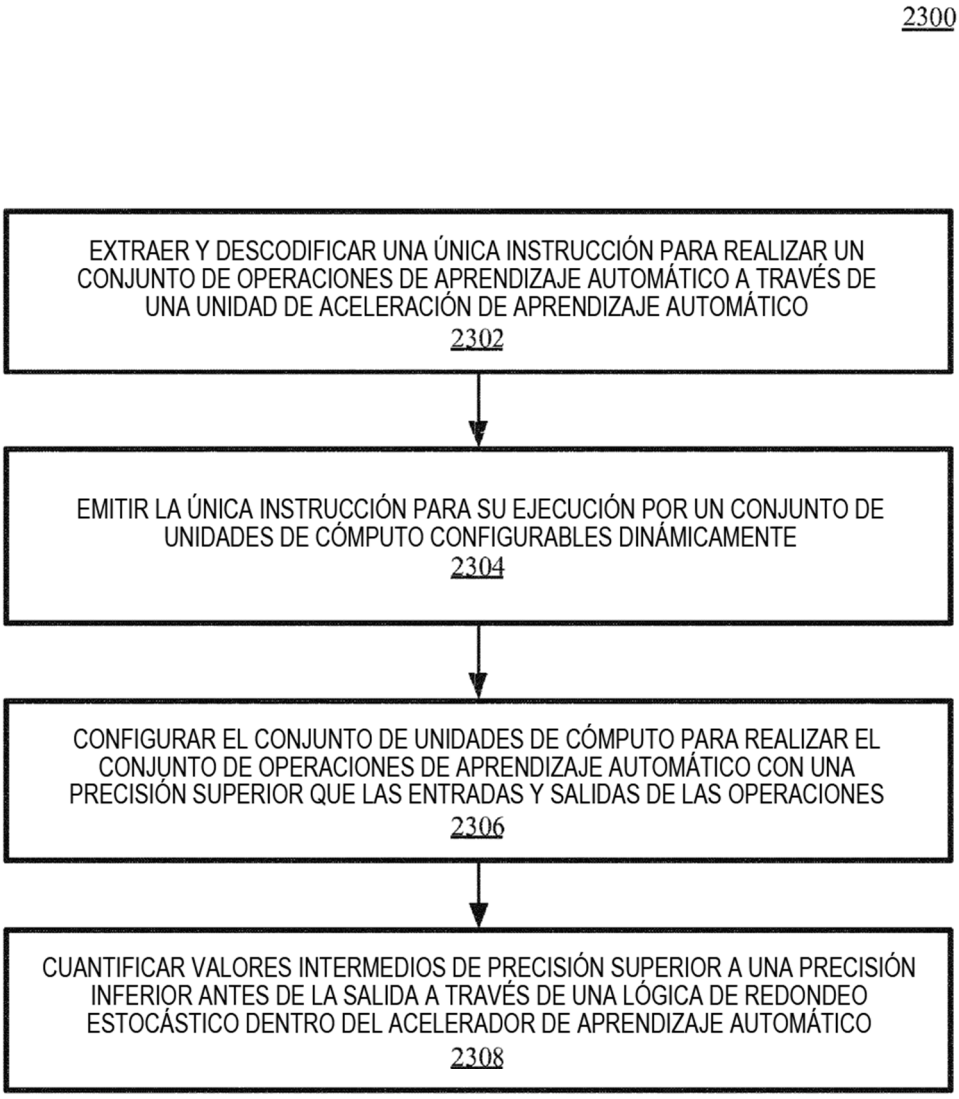

2300
EXTRAER Y DESCODIFICAR UNA ÚNICA INSTRUCCIÓN PARA REALIZAR UN CONJUNTO DE OPERACIONES DE APRENDIZAJE AUTOMÁTICO A TRAVÉS DE UNA UNIDAD DE ACELERACIÓN DE APRENDIZAJE AUTOMÁTICO
2302
EMITIR LA ÚNICA INSTRUCCIÓN PARA SU EJECUCIÓN POR UN CONJUNTO DE UNIDADES DE CÓMPUTO CONFIGURABLES DINÁMICAMENTE
2304
CONFIGURAR EL CONJUNTO DE UNIDADES DE CÓMPUTO PARA REALIZAR EL CONJUNTO DE OPERACIONES DE APRENDIZAJE AUTOMÁTICO CON UNA PRECISIÓN SUPERIOR QUE LAS ENTRADAS Y SALIDAS DE LAS OPERACIONES
2306
CUANTIFICAR VALORES INTERMEDIOS DE PRECISIÓN SUPERIOR A UNA PRECISIÓN INFERIOR ANTES DE LA SALIDA A TRAVÉS DE UNA LÓGICA DE REDONDEO ESTOCÁSTICO DENTRO DEL ACELERADOR DE APRENDIZAJE AUTOMÁTICO
2308

FIG. 23A

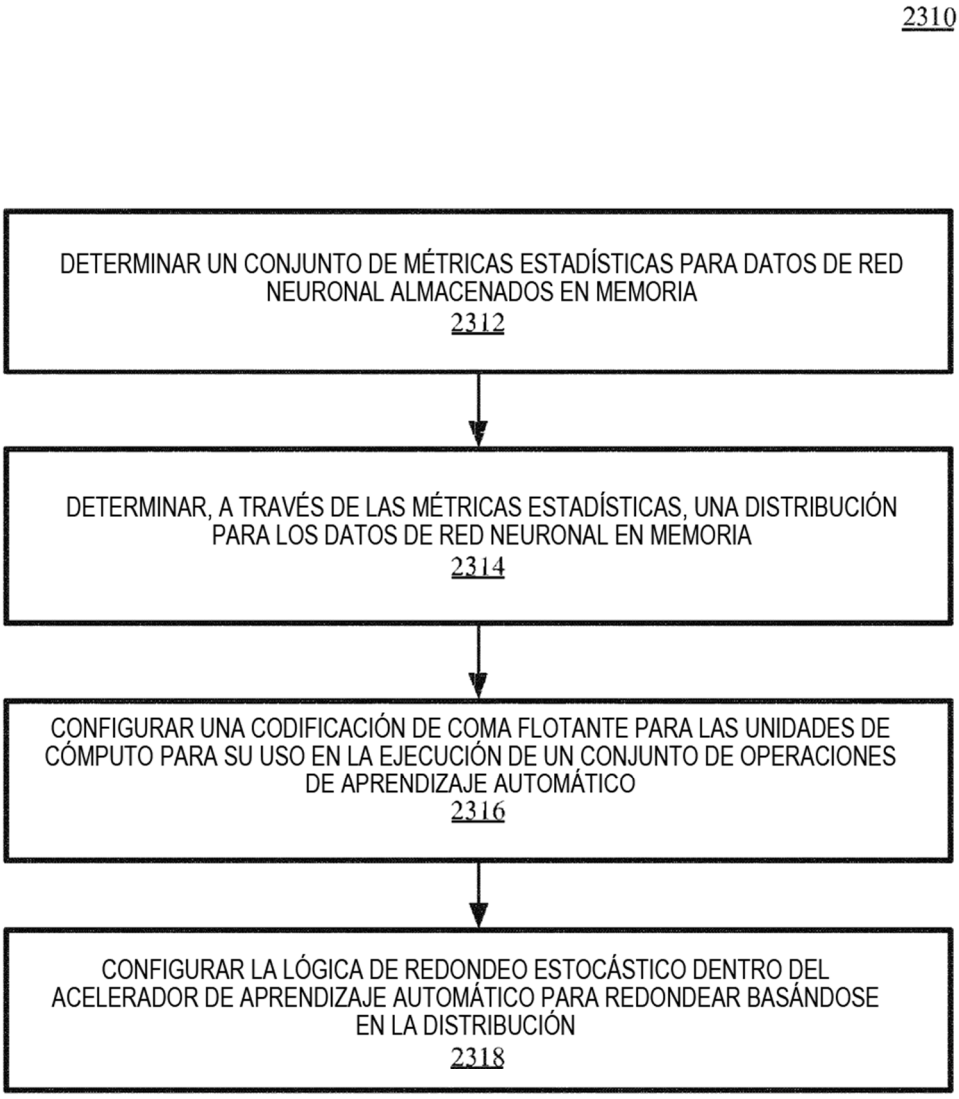

2310
DETERMINAR UN CONJUNTO DE MÉTRICAS ESTADÍSTICAS PARA DATOS DE RED NEURONAL ALMACENADOS EN MEMORIA
2312
DETERMINAR, A TRAVÉS DE LAS MÉTRICAS ESTADÍSTICAS, UNA DISTRIBUCIÓN PARA LOS DATOS DE RED NEURONAL EN MEMORIA
2314
CONFIGURAR UNA CODIFICACIÓN DE COMA FLOTANTE PARA LAS UNIDADES DE CÓMPUTO PARA SU USO EN LA EJECUCIÓN DE UN CONJUNTO DE OPERACIONES DE APRENDIZAJE AUTOMÁTICO
2316
CONFIGURAR LA LÓGICA DE REDONDEO ESTOCÁSTICO DENTRO DEL ACELERADOR DE APRENDIZAJE AUTOMÁTICO PARA REDONDEAR BASÁNDOSE EN LA DISTRIBUCIÓN
2318

FIG. 23B

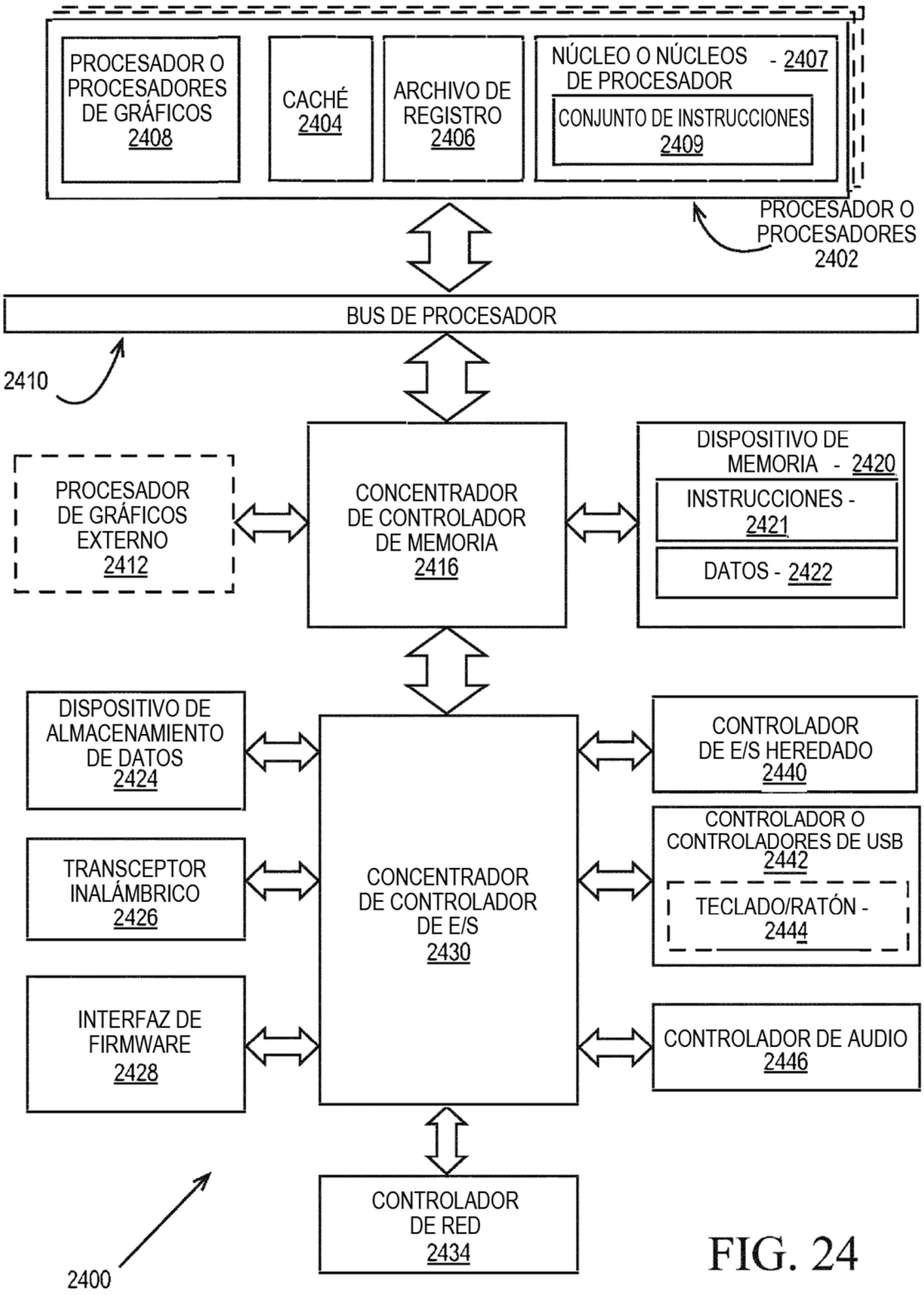

PROCESADOR O PROCESADORES DE GRÁFICOS 2408
CACHÉ 2404
ARCHIVO DE REGISTRO 2406
NÚCLEO O NÚCLEOS DE PROCESADOR - 2407
CONJUNTO DE INSTRUCCIONES 2409
PROCESADOR O PROCESADORES 2402
BUS DE PROCESADOR
2410
PROCESADOR DE GRÁFICOS EXTERNO 2412
CONCENTRADOR DE CONTROLADOR DE MEMORIA 2416
DISPOSITIVO DE MEMORIA - 2420
INSTRUCCIONES - 2421
DATOS - 2422
DISPOSITIVO DE ALMACENAMIENTO DE DATOS 2424
TRANSCEPTOR INALÁMBRICO 2426
INTERFAZ DE FIRMWARE 2428
CONCENTRADOR DE CONTROLADOR DE E/S 2430
CONTROLADOR DE E/S HEREDADO 2440
CONTROLADOR O CONTROLADORES DE USB 2442
TECLADO/RATÓN - 2444
CONTROLADOR DE AUDIO 2446
CONTROLADOR DE RED 2434
2400

FIG. 24

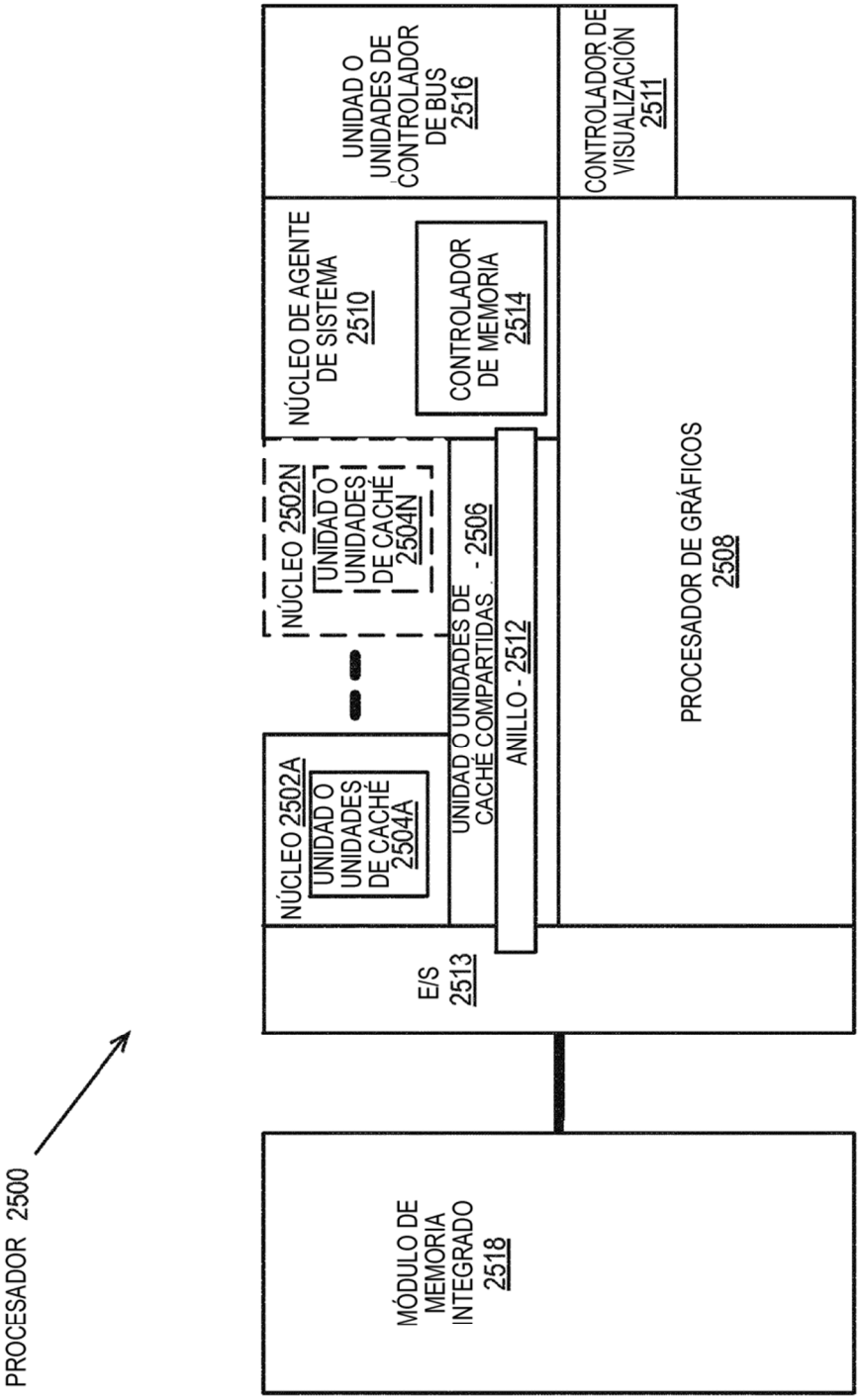

PROCESADOR 2500
MÓDULO DE MEMORIA INTEGRADO 2518
E/S 2513
NÚCLEO 2502A
UNIDAD O UNIDADES DE CACHÉ 2504A
NÚCLEO 2502N
UNIDAD O UNIDADES DE CACHÉ 2504N
UNIDAD O UNIDADES DE CACHÉ COMPARTIDAS - 2506
ANILLO - 2512
NÚCLEO DE AGENTE DE SISTEMA 2510
CONTROLADOR DE MEMORIA 2514
UNIDAD O UNIDADES DE CONTROLADOR DE BUS 2516
CONTROLADOR DE VISUALIZACIÓN 2511
PROCESADOR DE GRÁFICOS 2508

FIG. 25

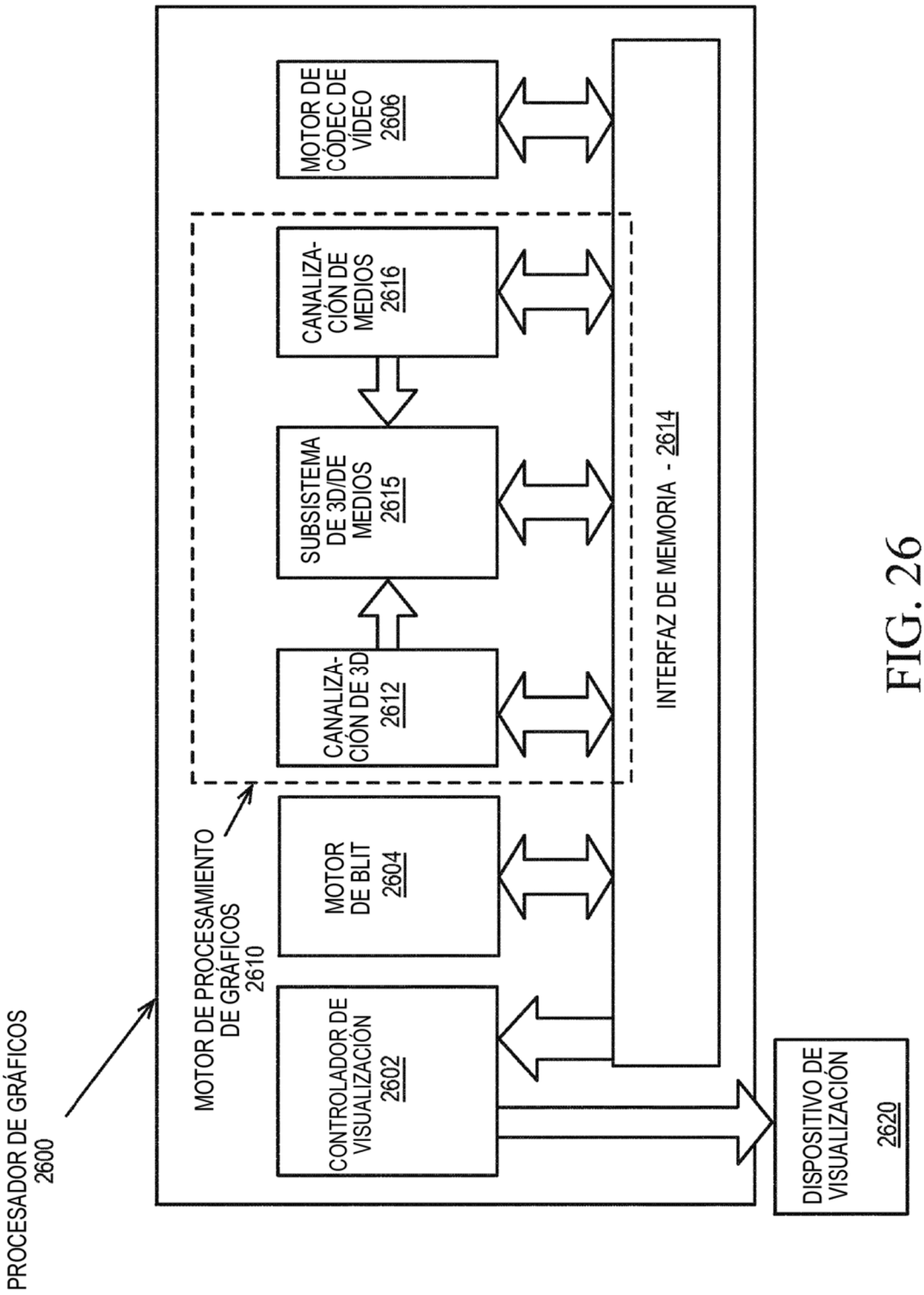

PROCESADOR DE GRÁFICOS 2600
MOTOR DE PROCESAMIENTO DE GRÁFICOS 2610
CONTROLADOR DE VISUALIZACIÓN 2602
MOTOR DE BLIT 2604
CANALIZACIÓN DE 3D 2612
SUBSISTEMA DE 3D/DE MEDIOS 2615
CANALIZACIÓN DE MEDIOS 2616
MOTOR DE CÓDEC DE VÍDEO 2606
INTERFAZ DE MEMORIA - 2614
DISPOSITIVO DE VISUALIZACIÓN 2620

FIG. 26

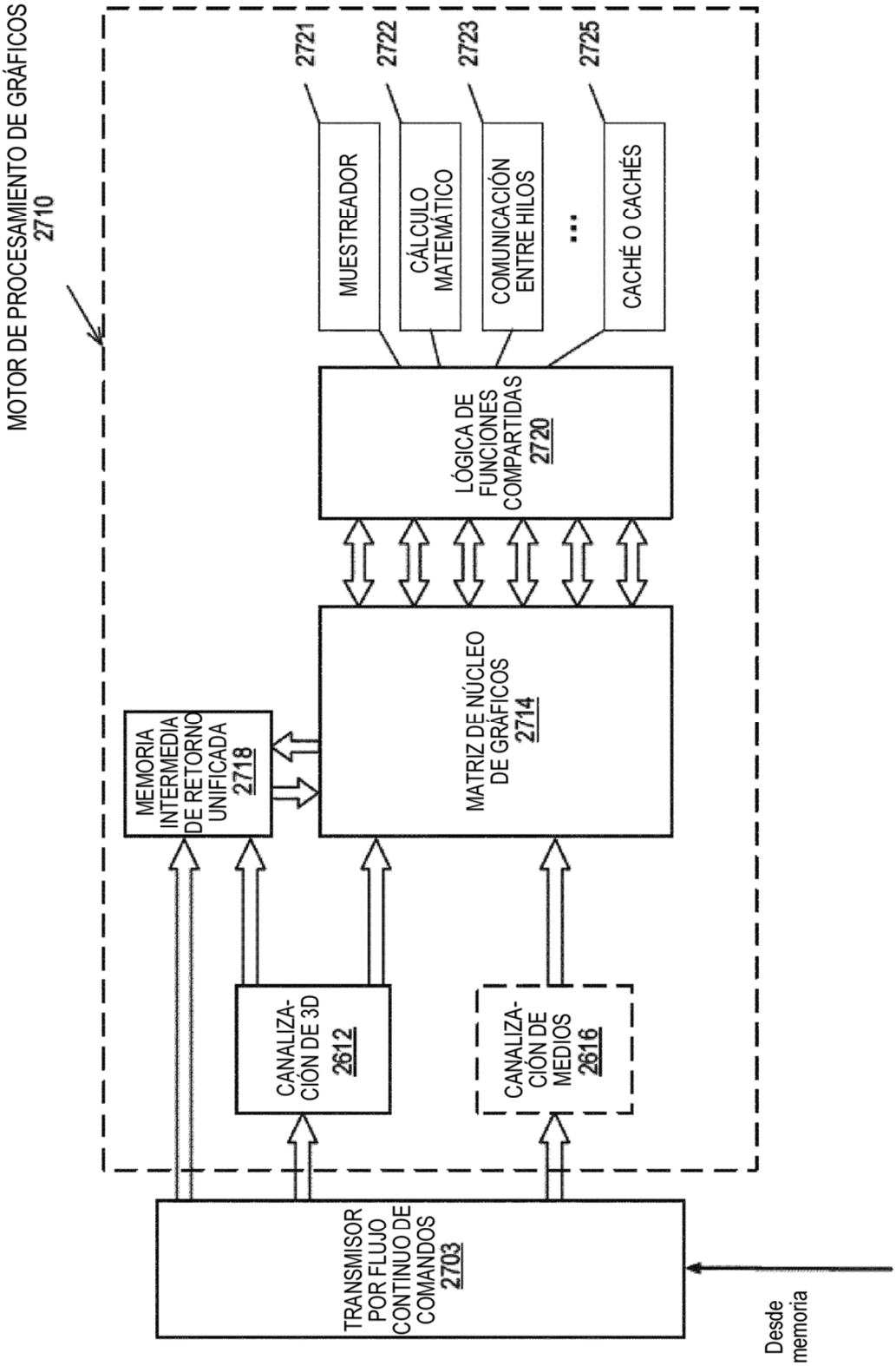

MOTOR DE PROCESAMIENTO DE GRÁFICOS
2710
2721
2722
2723
2725
MUESTREADOR
CÁLCULO MATEMÁTICO
COMUNICACIÓN ENTRE HILOS
...
CACHÉ O CACHÉS
LÓGICA DE FUNCIONES COMPARTIDAS
2720
MATRIZ DE NÚCLEO DE GRÁFICOS
2714
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
2718
CANALIZA-CIÓN DE 3D
2612
CANALIZA-CIÓN DE MEDIOS
2616
TRANSMISOR POR FLUJO CONTINUO DE COMANDOS
2703
Desde memoria

FIG. 27

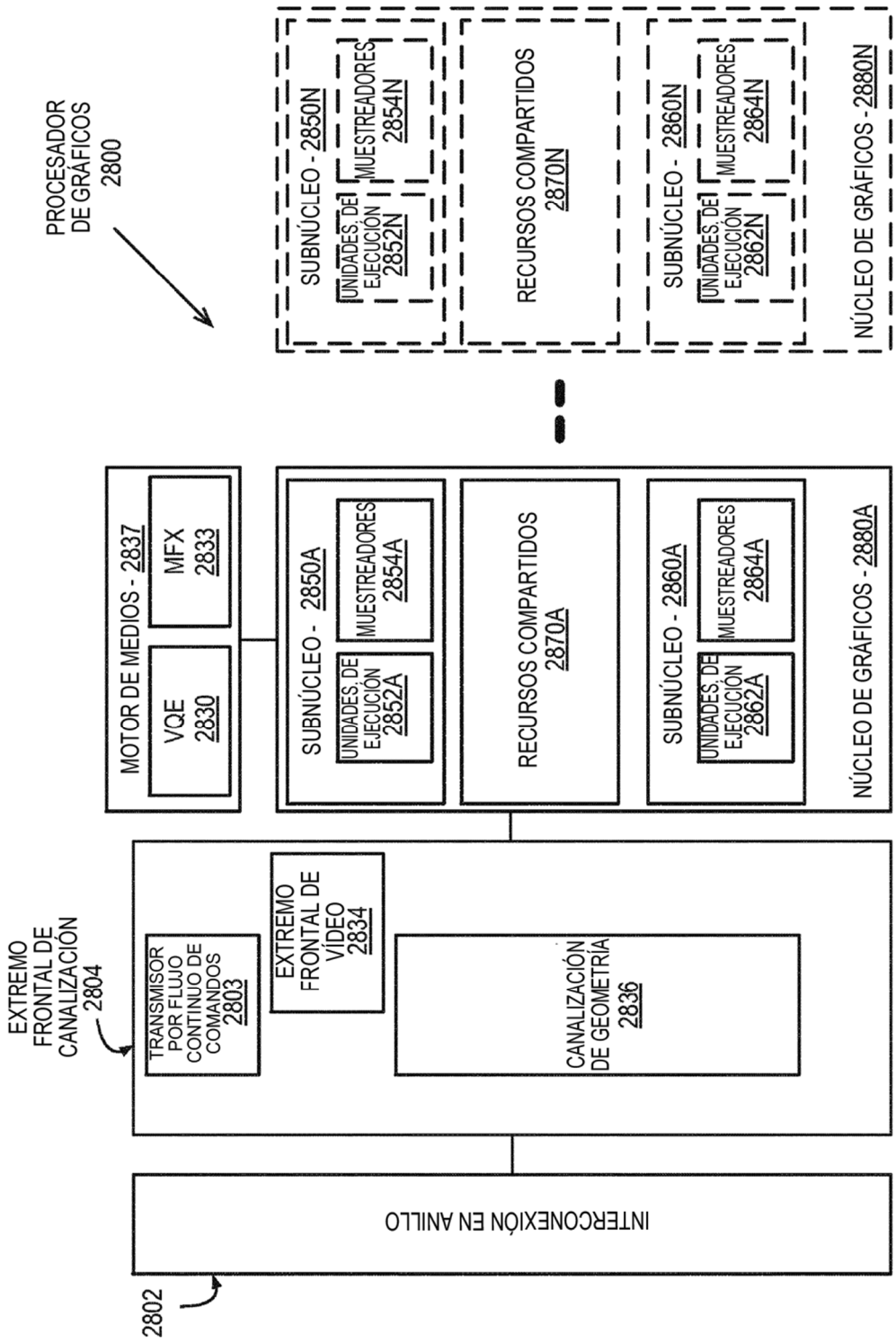

PROCESADOR DE GRÁFICOS 2800
SUBNÚCLEO - 2850N
UNIDADES DE EJECUCIÓN 2852N
MUESTREADORES 2854N
RECURSOS COMPARTIDOS 2870N
SUBNÚCLEO - 2860N
UNIDADES DE EJECUCIÓN 2862N
MUESTREADORES 2864N
NÚCLEO DE GRÁFICOS - 2880N
MOTOR DE MEDIOS - 2837
VQE 2830
MFX 2833
SUBNÚCLEO - 2850A
UNIDADES DE EJECUCIÓN 2852A
MUESTREADORES 2854A
RECURSOS COMPARTIDOS 2870A
SUBNÚCLEO - 2860A
UNIDADES DE EJECUCIÓN 2862A
MUESTREADORES 2864A
NÚCLEO DE GRÁFICOS - 2880A
EXTREMO FRONTAL DE CANALIZACIÓN 2804
TRANSMISOR POR FLUJO CONTINUO DE COMANDOS 2803
EXTREMO FRONTAL DE VÍDEO 2834
CANALIZACIÓN DE GEOMETRÍA 2836
INTERCONEXIÓN EN ANILLO
2802

FIG. 28

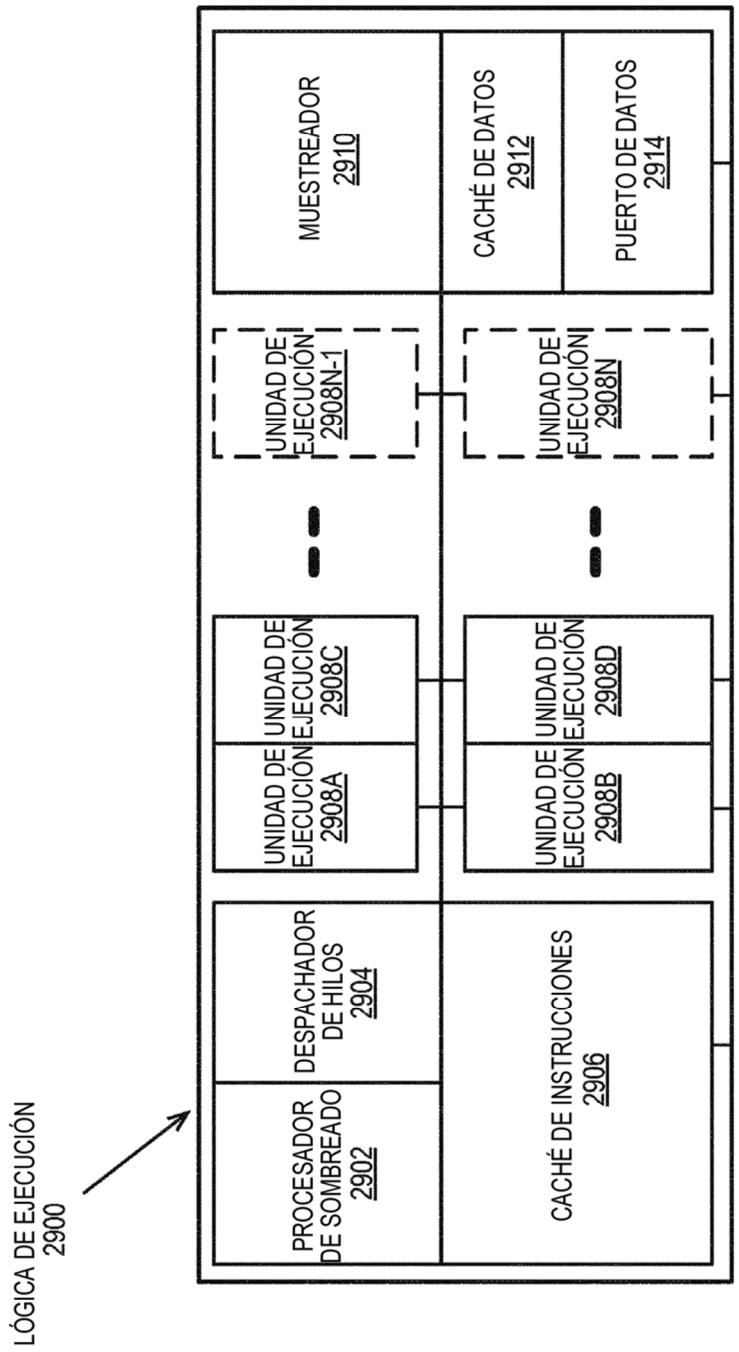

LÓGICA DE EJECUCIÓN
2900
PROCESADOR DE SOMBREADO
2902
DESPACHADOR DE HILOS
2904
CACHÉ DE INSTRUCCIONES
2906
UNIDAD DE EJECUCIÓN
2908A
UNIDAD DE EJECUCIÓN
2908B
UNIDAD DE EJECUCIÓN
2908C
UNIDAD DE EJECUCIÓN
2908D
UNIDAD DE EJECUCIÓN
2908N-1
UNIDAD DE EJECUCIÓN
2908N
MUESTREADOR
2910
CACHÉ DE DATOS
2912
PUERTO DE DATOS
2914

FIG. 29

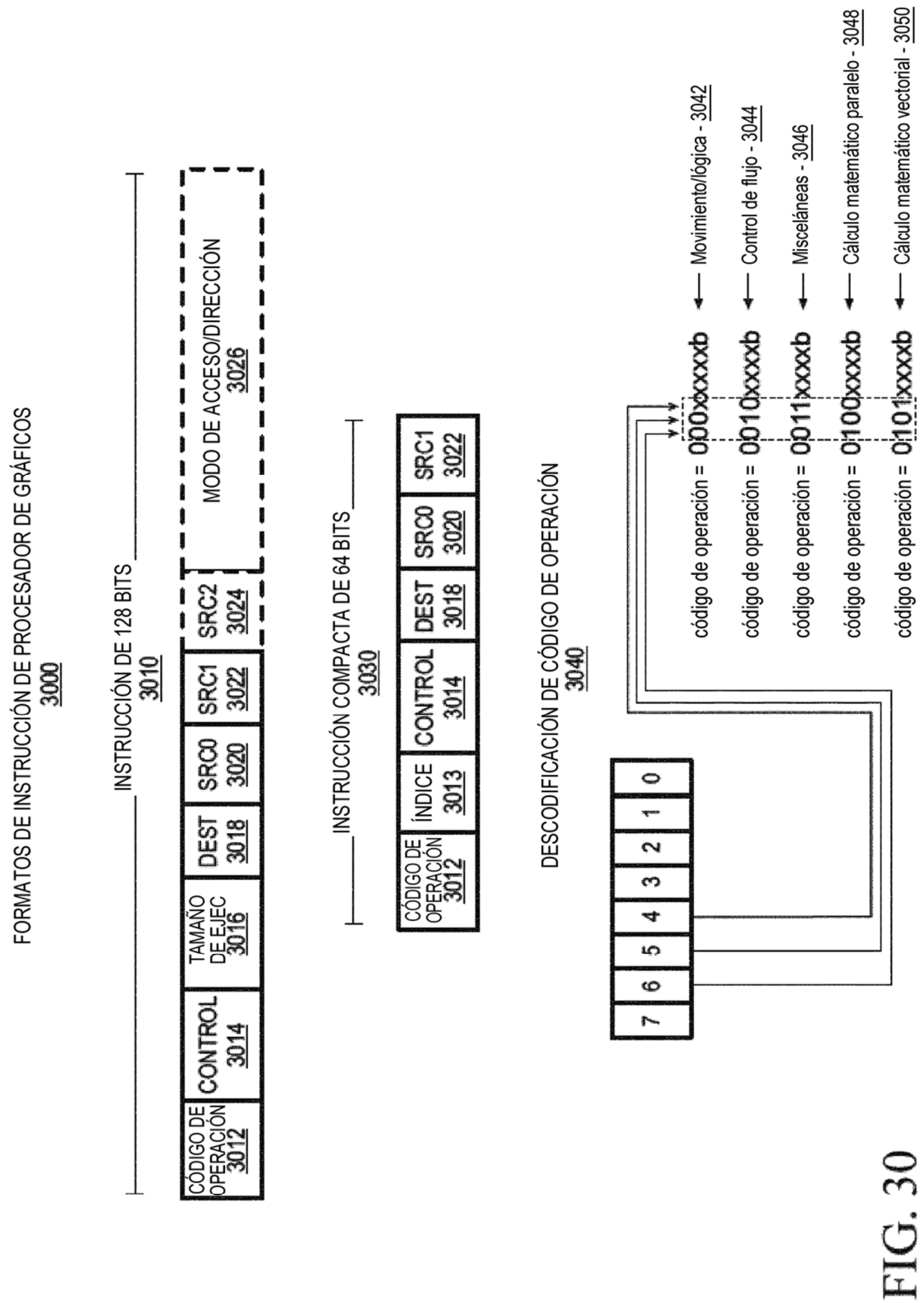

FORMATOS DE INSTRUCCIÓN DE PROCESADOR DE GRÁFICOS
3000
INSTRUCCIÓN DE 128 BITS
3010
CÓDIGO DE OPERACIÓN 3012
CONTROL 3014
TAMAÑO DE EJEC 3016
DEST 3018
SRC0 3020
SRC1 3022
SRC2 3024
MODO DE ACCESO/DIRECCIÓN 3026
INSTRUCCIÓN COMPACTA DE 64 BITS
3030
CÓDIGO DE OPERACIÓN 3012
ÍNDICE 3013
CONTROL 3014
DEST 3018
SRC0 3020
SRC1 3022
DESCODIFICACIÓN DE CÓDIGO DE OPERACIÓN
3040
7 6 5 4 3 2 1 0
código de operación = 000xxxxb
código de operación = 0010xxxxb
código de operación = 0011xxxxb
código de operación = 0100xxxxb
código de operación = 0101xxxxb
Movimiento/lógica - 3042
Control de flujo - 3044
Misceláneas - 3046
Cálculo matemático paralelo - 3048
Cálculo matemático vectorial - 3050

FIG. 30

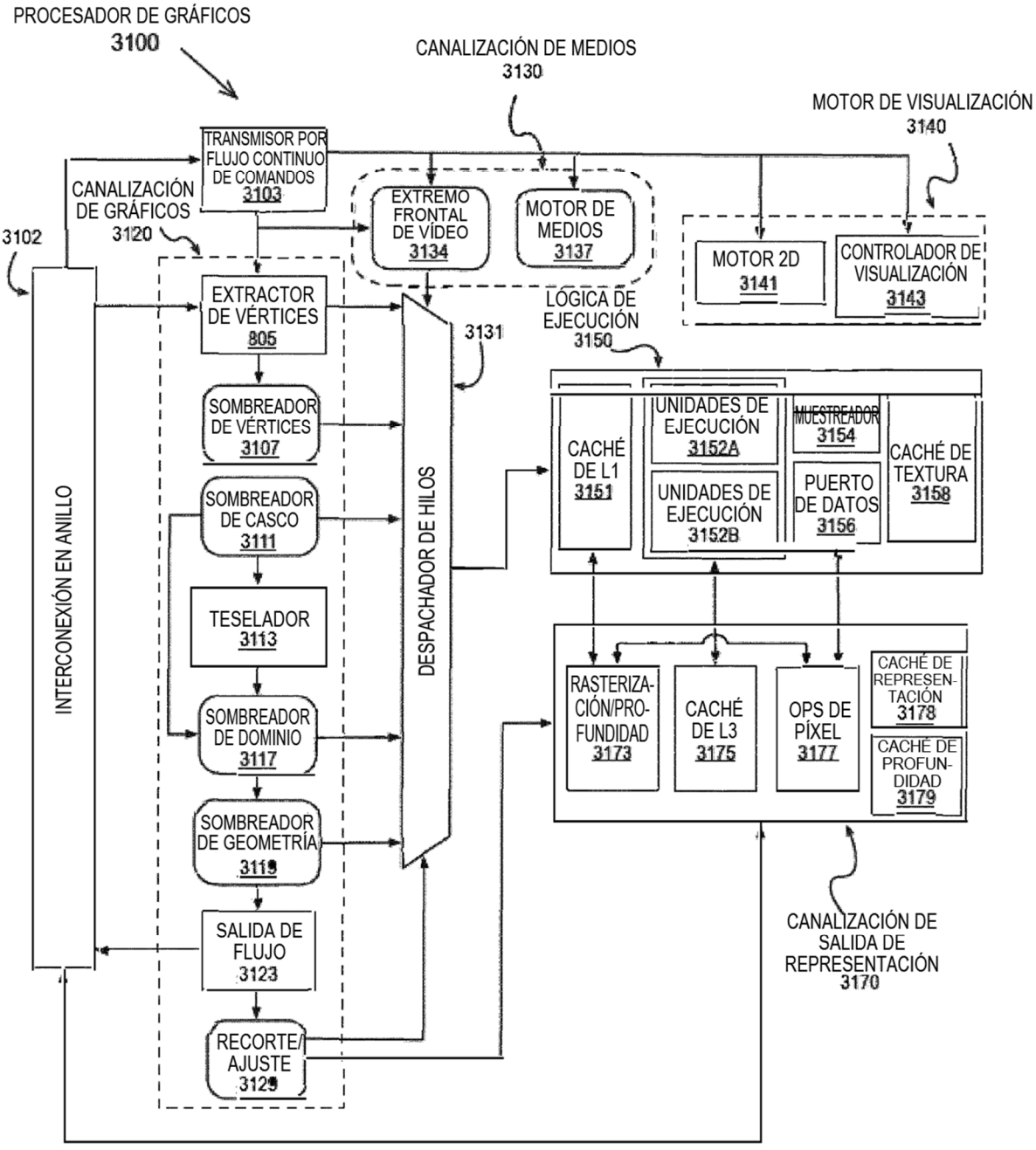
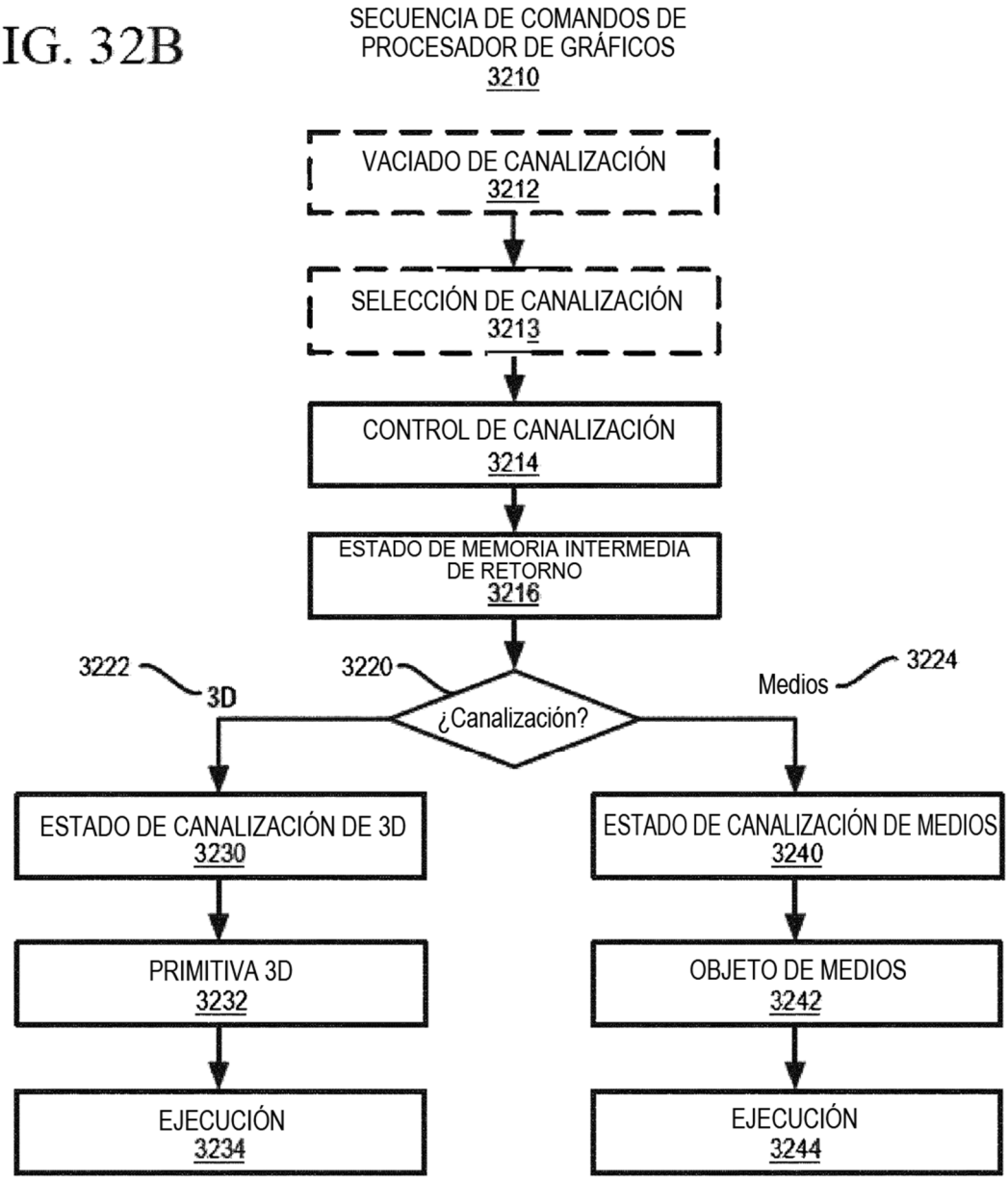

PROCESADOR DE GRÁFICOS
3100
CANALIZACIÓN DE MEDIOS
3130
MOTOR DE VISUALIZACIÓN
3140
TRANSMISOR POR FLUJO CONTINUO DE COMANDOS
3103
CANALIZACIÓN DE GRÁFICOS
3120
3102
EXTREMO FRONTAL DE VÍDEO
3134
MOTOR DE MEDIOS
3137
MOTOR 2D
3141
CONTROLADOR DE VISUALIZACIÓN
3143
LÓGICA DE EJECUCIÓN
3150
3131
EXTRACTOR DE VÉRTICES
805
SOMBREADOR DE VÉRTICES
3107
SOMBREADOR DE CASCO
3111
TESELADOR
3113
SOMBREADOR DE DOMINIO
3117
SOMBREADOR DE GEOMETRÍA
3119
SALIDA DE FLUJO
3123
RECORTE/AJUSTE
3129
INTERCONEXIÓN EN ANILLO
DESPACHADOR DE HILOS
CACHÉ DE L1
3151
UNIDADES DE EJECUCIÓN
3152A
UNIDADES DE EJECUCIÓN
3152B
MUESTREADOR
3154
PUERTO DE DATOS
3156
CACHÉ DE TEXTURA
3158
RASTERIZACIÓN/PROFUNDIDAD
3173
CACHÉ DE L3
3175
OPS DE PÍXEL
3177
CACHÉ DE REPRESENTACIÓN
3178
CACHÉ DE PROFUNDIDAD
3179
CANALIZACIÓN DE SALIDA DE REPRESENTACIÓN
3170

FIG. 31

FIG. 32A

FIG. 32B



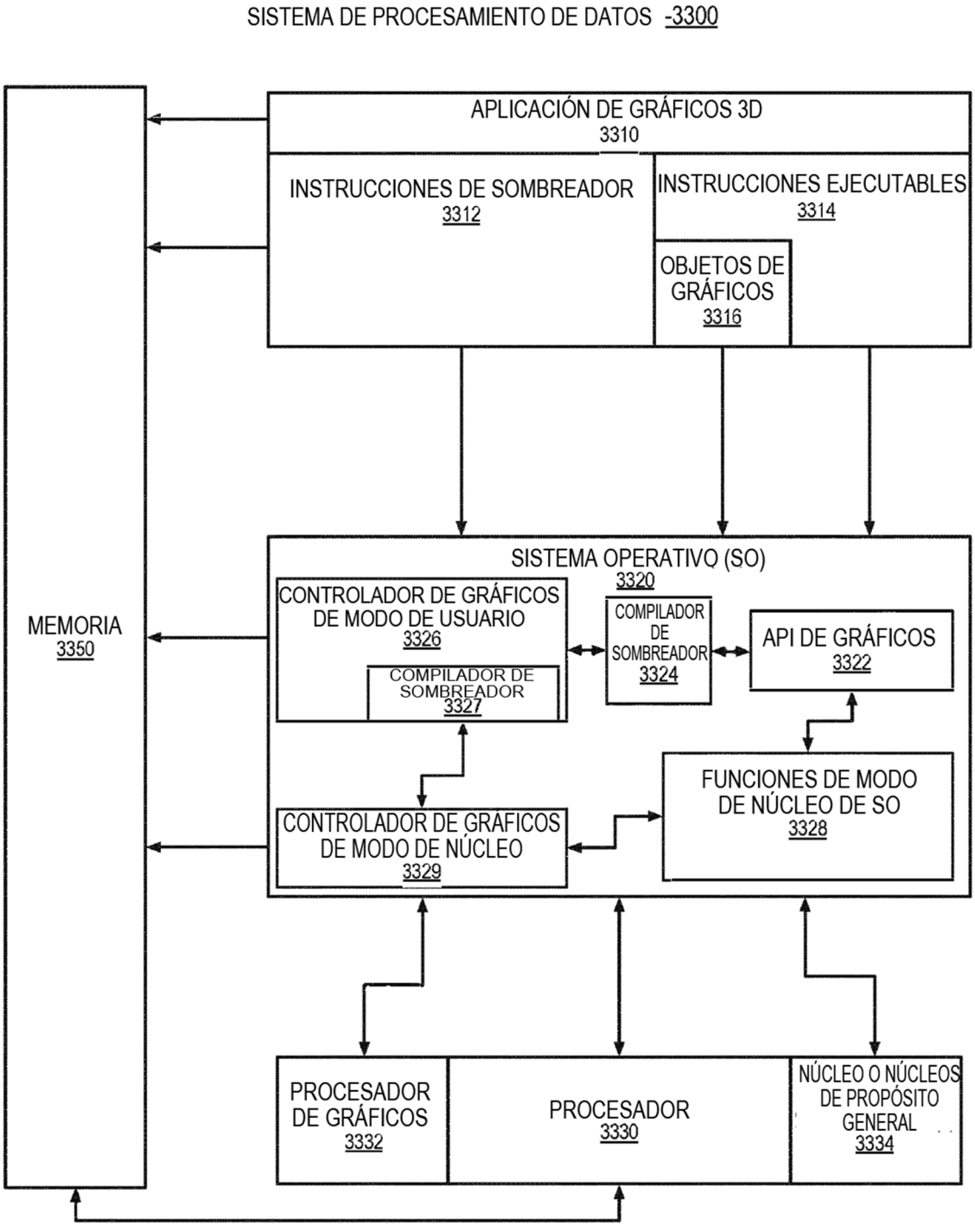

SISTEMA DE PROCESAMIENTO DE DATOS -3300
APLICACIÓN DE GRÁFICOS 3D
3310
INSTRUCCIONES DE SOMBREADOR
3312
INSTRUCCIONES EJECUTABLES
3314
OBJETOS DE GRÁFICOS
3316
MEMORIA
3350
SISTEMA OPERATIVO (SO)
3320
CONTROLADOR DE GRÁFICOS DE MODO DE USUARIO
3326
COMPILADOR DE SOMBREADOR
3327
COMPILADOR DE SOMBREADOR
3324
API DE GRÁFICOS
3322
FUNCIONES DE MODO DE NÚCLEO DE SO
3328
CONTROLADOR DE GRÁFICOS DE MODO DE NÚCLEO
3329
PROCESADOR DE GRÁFICOS
3332
PROCESADOR
3330
NÚCLEO O NÚCLEOS DE PROPÓSITO GENERAL
3334

FIG. 33

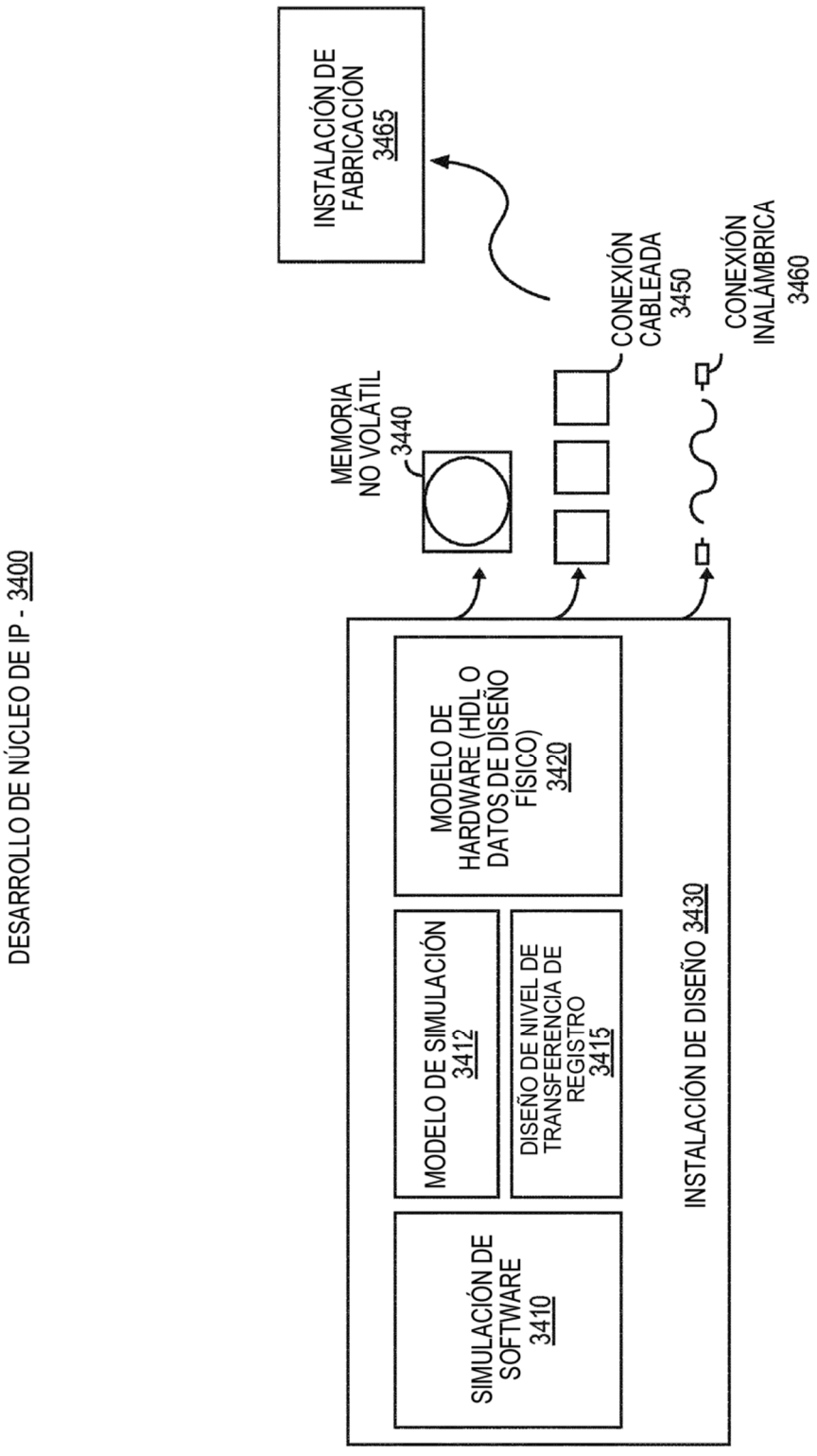

DESARROLLO DE NÚCLEO DE IP - 3400
INSTALACIÓN DE FABRICACIÓN 3465
MEMORIA NO VOLÁTIL 3440
CONEXIÓN CABLEADA 3450
CONEXIÓN INALÁMBRICA 3460
SIMULACIÓN DE SOFTWARE 3410
MODELO DE SIMULACIÓN 3412
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 3415
MODELO DE HARDWARE (HDL O DATOS DE DISEÑO FÍSICO) 3420
INSTALACIÓN DE DISEÑO 3430

FIG. 34

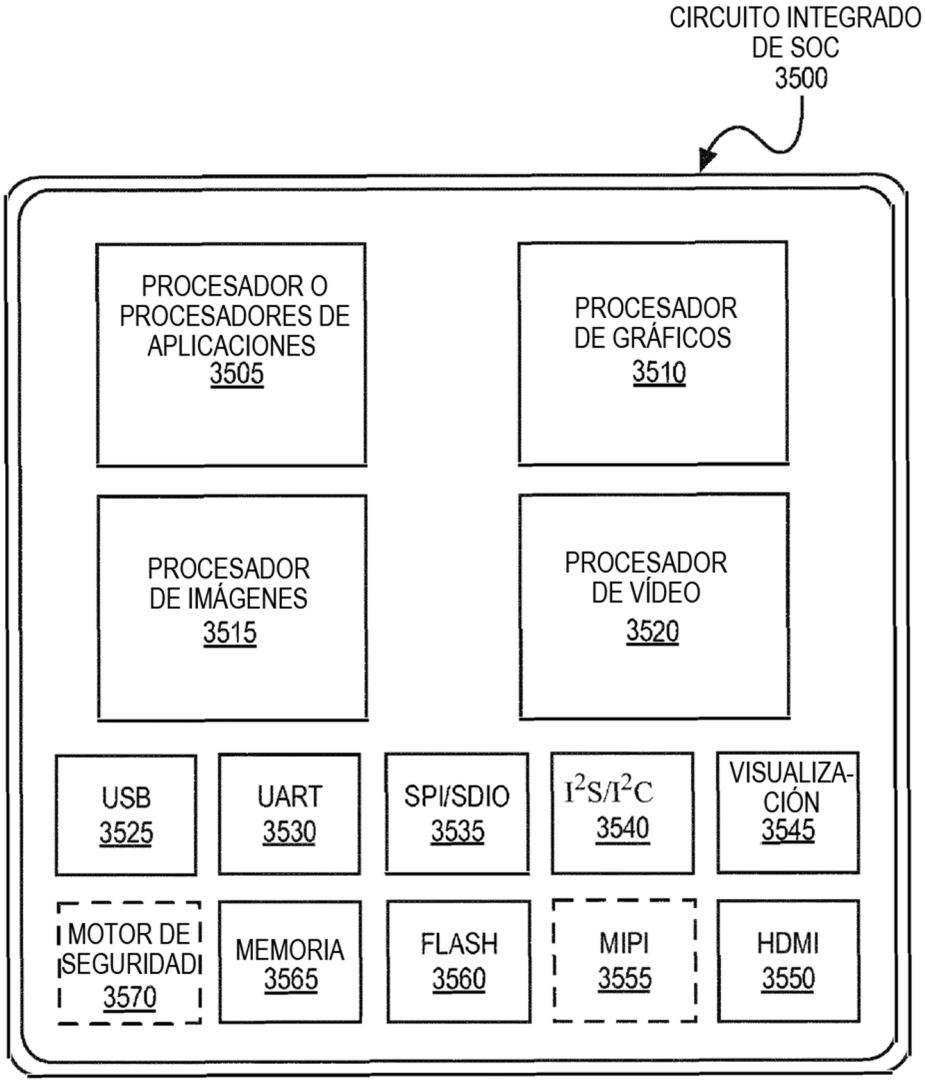

CIRCUITO INTEGRADO
DE SOC
3500
PROCESADOR O
PROCESADORES DE
APLICACIONES
3505
PROCESADOR
DE GRÁFICOS
3510
PROCESADOR
DE IMÁGENES
3515
PROCESADOR
DE VÍDEO
3520
USB
3525
UART
3530
SPI/SDIO
3535
I2S/I2C
3540
VISUALIZA-
CIÓN
3545
MOTOR DE
SEGURIDAD
3570
MEMORIA
3565
FLASH
3560
MIPI
3555
HDMI
3550

FIG. 35

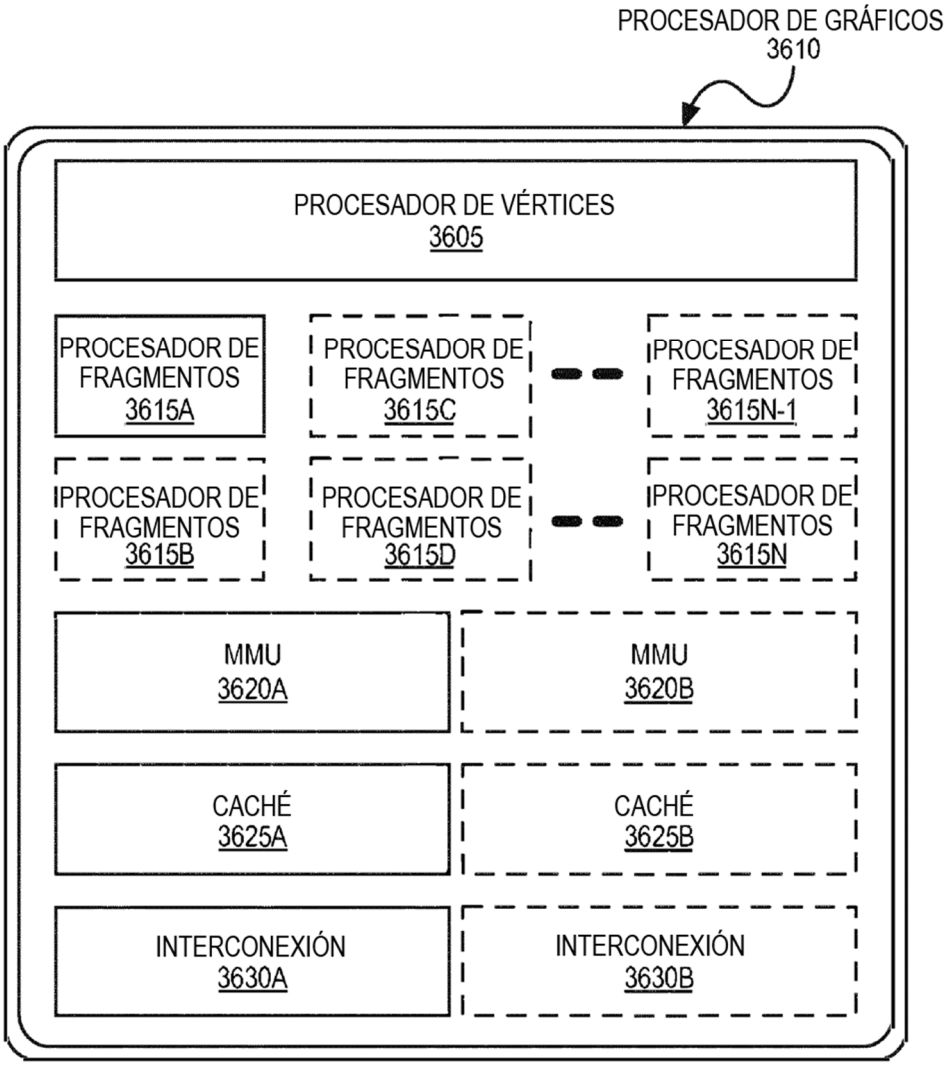

PROCESADOR DE GRÁFICOS
3610
PROCESADOR DE VÉRTICES
3605
PROCESADOR DE FRAGMENTOS
3615A
PROCESADOR DE FRAGMENTOS
3615C
PROCESADOR DE FRAGMENTOS
3615N-1
PROCESADOR DE FRAGMENTOS
3615B
PROCESADOR DE FRAGMENTOS
3615D
PROCESADOR DE FRAGMENTOS
3615N
MMU
3620A
MMU
3620B
CACHÉ
3625A
CACHÉ
3625B
INTERCONEXIÓN
3630A
INTERCONEXIÓN
3630B

FIG. 36

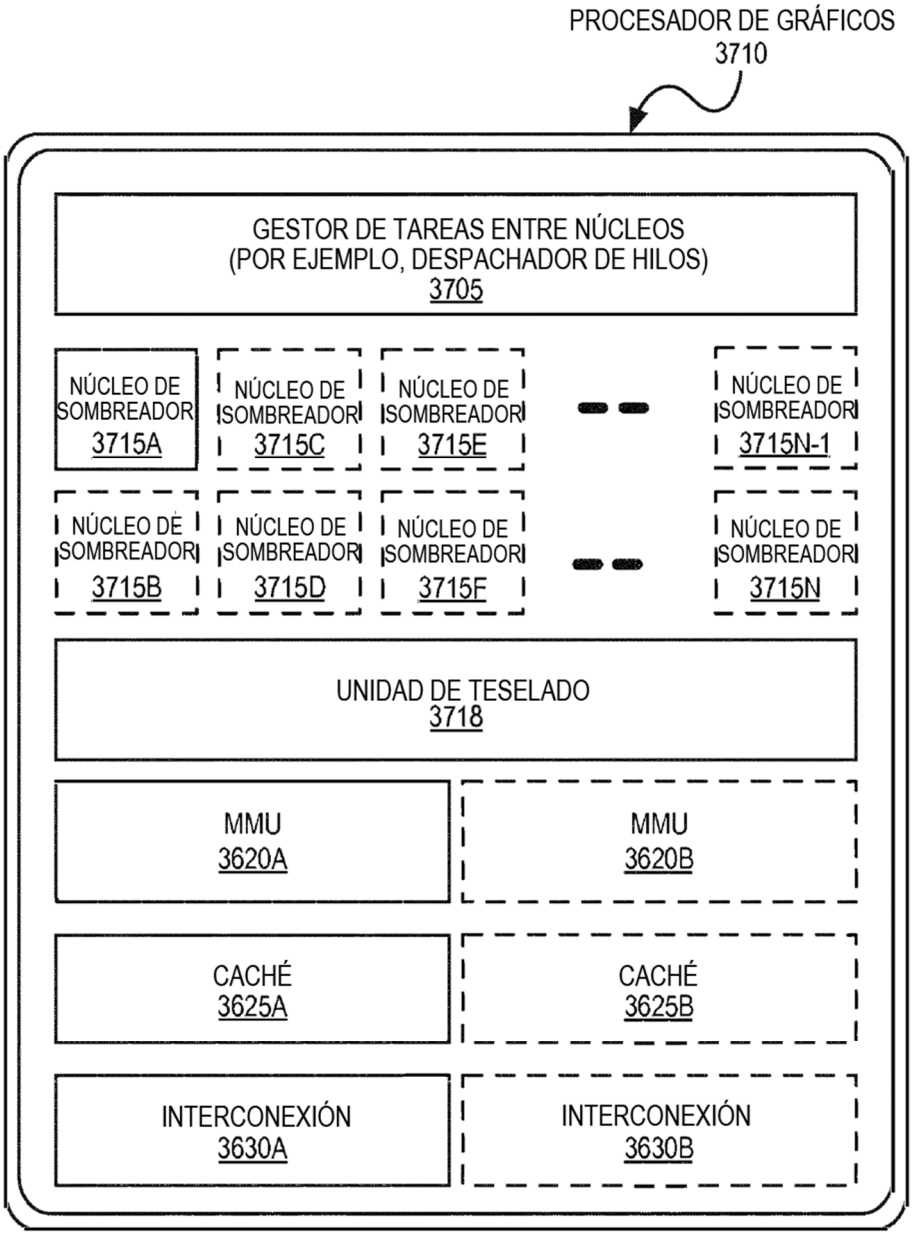

PROCESADOR DE GRÁFICOS
3710
GESTOR DE TAREAS ENTRE NÚCLEOS
(POR EJEMPLO, DESPACHADOR DE HILOS)
3705
NÚCLEO DE SOMBREADOR
3715A
NÚCLEO DE SOMBREADOR
3715C
NÚCLEO DE SOMBREADOR
3715E
NÚCLEO DE SOMBREADOR
3715N-1
NÚCLEO DE SOMBREADOR
3715B
NÚCLEO DE SOMBREADOR
3715D
NÚCLEO DE SOMBREADOR
3715F
NÚCLEO DE SOMBREADOR
3715N
UNIDAD DE TESELADO
3718
MMU
3620A
MMU
3620B
CACHÉ
3625A
CACHÉ
3625B
INTERCONEXIÓN
3630A
INTERCONEXIÓN
3630B

FIG. 37