(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 913 992**

(51) Int. CI.:

G06F 9/30 (2008.01)
G06F 9/38 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **19.03.2018 E 18162635 (9)**

(97) Fecha y número de publicación de la concesión europea: **09.03.2022 EP 3396533**

(54) Título: **Hardware programable de cálculo de matriz dispersa y de grano grueso con planificación avanzada**

(30) Prioridad:

28.04.2017 US 201715581182

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
07.06.2022

(73) Titular/es:

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

(72) Inventor/es:

NURVITADHI, ERIKO;
VEMBU, BALAJI;
GALOPPO VON BORRIES, NICOLAS C.;
BARIK, RAJKISHORE;
LIN, TSUNG-HAN;
SINHA, KAMAL;
SATISH, NADATHUR RAJAGOPALAN;
BOTTLESON, JEREMY;
AKHBARI, FARSHAD;
KOKER, ALTUG;
SRINIVASA, NARAYAN;
KIM, DUKHWAN;
BAGHSORKHI, SARA S.;
GOTTSCHLICH, JUSTIN E.;
CHEN, FENG;
OULD-AHMED-VALL, ELMOUSTAPHA;
NEALIS, KEVIN;
CHEN, XIAOMING y
YAO, ANBANG

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Hardware programable de cálculo de matriz dispersa y de grano grueso con planificación avanzada

Campo técnico

Las realizaciones se refieren en general al procesamiento de datos y, más particularmente, al procesamiento de datos mediante una unidad de procesamiento de gráficos de fin general.

Antecedentes de la descripción

El procesamiento de datos de gráficos paralelo actual incluye sistemas y métodos desarrollados para realizar operaciones específicas en datos de gráficos, tales como, por ejemplo, interpolación lineal, teselación, rasterización, mapeo de texturas, 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, se han hecho programables porciones de los procesadores de gráficos, lo que posibilita que tales procesadores soporten una gama más amplia de operaciones para procesar datos de vértices y de fragmentos.

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

El documento US 2011/029471 A1 se refiere a un coprocesador y a un método para procesar redes neuronales convolucionales que incluye un conmutador de entrada configurable acoplado a una entrada. Se posibilita una pluralidad de elementos de convolución de acuerdo con el conmutador de entrada. Un conmutador de salida está configurado para recibir salidas del conjunto de elementos de convolución para proporcionar datos a los ramales de salida. Un controlador está configurado para proporcionar señales de control al conmutador de entrada y al conmutador de salida de manera que se presenta activo el conjunto de elementos de convolución y se selecciona un número de ramales de salida para un ciclo dado de acuerdo con las señales de control.

Heehoon Kim et al "Performance analysis of CNNframeworks for GPUS″ se refiere a un análisis de cinco estructuras de aprendizaje profundo conocidas.

PEDRAM ARDAVAN ET AL: "A Highly Efficient Multicore Floating-Point FFT Architecture based on Hybrid Linear Algebra/FFT Cores", se refiere a la selección de una de múltiples unidades de ejecución.

El documento US 2009/111413 se refiere a la determinación de un tipo de operación para realizar una instrucción basándose en parámetros.

El documento US 2006/200810 se refiere a la determinación de un tipo de operación para realizar una instrucción basándose en parámetros.

La invención se define por un aparato de cálculo, un método y al menos un medio legible por máquina de acuerdo con las reivindicaciones independientes. Se definen realizaciones preferidas en las reivindicaciones dependientes. Específicamente, la invención es como se expone en las Figuras 19A y 19B y en la descripción pertinente.

Breve descripción de los dibujos

De modo que las características de la presente invención puedan entenderse en detalle, puede obtenerse una descripción de la invención más particular por referencia a las realizaciones, algunas de las cuales se ilustran en los dibujos adjuntos. Sin embargo, se ha de observar que los dibujos adjuntos ilustran únicamente realizaciones típicas 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;

La **Figura 2A-2D** ilustra unos componentes de procesador paralelo, de acuerdo con una realización;

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

La **Figura 4A-4F** ilustra una arquitectura ilustrativa en la que una pluralidad de GPU están comunicativamente acopladas a una pluralidad de procesadores de múltiples núcleos;

La **Figura 5** ilustra una tubería 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 fin general altamente paralela, de acuerdo con una realización;

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

La **Figura 9A-9B** ilustra 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 inferencia ilustrativo adecuado para realizar la inferencia usando un modelo entrenado;

La **Figura 14** es un diagrama de bloques de un sistema de procesamiento de datos, de acuerdo con una realización;

La **Figura 15A** ilustra detalles de una unidad de instrucción y de extracción de aprendizaje automático, de acuerdo con una realización;

La **Figura 15B** ilustra detalles de un controlador de planificador de aprendizaje automático, de acuerdo con una realización;

La **Figura 16** ilustra operaciones convolucionales ilustrativas, de acuerdo con las realizaciones;

La **Figura 17** es un diagrama de flujo de lógica para realizar planificación de grado basto de operaciones de aprendizaje automático en una tubería de cálculo, de acuerdo con una realización;

La **Figura 18** es un diagrama de bloques que ilustra un sistema de cálculo de memoria híbrida, de acuerdo con una realización

Las Figuras 19A-19B son diagramas de flujo que ilustran lógica para realizar operaciones de cálculo de datos cercanos mediante las realizaciones descritas en el presente documento;

La **Figura 20** ilustra lógica de multiplicación-adición ilustrativa dentro de las realizaciones descritas en el presente documento;

La **Figura 21** ilustra una arquitectura de acelerador de cálculo disperso, de acuerdo con una realización;

La **Figura 22** ilustra una arquitectura de cálculo disperso adicional para operaciones matriciales dispersas, de acuerdo con una realización;

Las Figuras 23A-23B son diagramas de flujo que ilustran la lógica 2300, 2310 para realizar operaciones de cálculo disperso dentro de una GPGPU proporcionada por las 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 lógica de ejecución de hilo 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.

La **Figura 32A-32B** ilustra un formato de comando de procesador de gráficos y 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 sistema ilustrativo en un circuito de chip integrado, 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 sistema en un circuito de chip integrado, 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/de procesador para acelerar las operaciones de gráficos, operaciones de aprendizaje automático, operaciones de análisis de patrones y diversas funciones de GPU de fin general (GPGPU). La GPU puede estar acoplada 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 puede estar integrada 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, interna 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, a continuación, usa circuitería/lógica especializada para el procesamiento de manera eficaz de estos comandos/instrucciones.

En la siguiente descripción, 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 pueden ponerse en práctica sin uno o más de estos detalles específicos. En otras instancias, no se han descrito características bien conocidas para evitar oscurecer los detalles de las presentes realizaciones.

Vista general del 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 procesador o procesadores 102 y una memoria de sistema 104 que se comunica mediante una ruta de interconexión que puede incluir un concentrador de memoria 105. El concentrador de memoria 105 puede ser un componente separado dentro de un componente de conjunto de chips o puede estar integrado dentro del uno o más procesador o procesadores 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 mediante un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede posibilitar que el sistema informático 100 reciba entrada desde uno o más dispositivo o dispositivos de entrada 108. Adicionalmente, el concentrador de E/S 107 puede posibilitar un controlador de visualización, que puede estar incluido en el uno o más procesador o procesadores 102, para proporcionar salidas a uno o más dispositivo o dispositivos de visualización 110A. En una realización el uno o más dispositivo o 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 procesador o procesadores paralelos 112 acoplados al concentrador de memoria 105 mediante un bus u otro enlace de comunicación 113. El enlace de comunicación 113 puede ser uno de cualquier número de tecnologías o protocolos de enlace de comunicación

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

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

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

En una realización, el uno o más procesador o procesadores paralelos 112 incorporan circuitería optimizada para procesamiento de gráficos y vídeo, que incluye, por ejemplo, circuitería de salida de vídeo y constituye una unidad de procesamiento de gráficos (GPU). En otra realización, el uno o más procesador o procesadores paralelos 112 incorporan circuitería optimizada para procesamiento de fin general, mientras que conservan la arquitectura computacional subyacente, descrita en mayor detalle en el presente documento. En otra realización más, los componentes del sistema informático 100 pueden estar integrados con uno o más otros elementos de sistema en un único circuito integrado. Por ejemplo, el uno o más procesador o procesadores paralelos 112, el concentrador de memoria 105, el procesador o procesadores 102, y el concentrador de E/S 107 pueden estar integrados un circuito integrado de sistema en chip (SoC). Como alternativa, los componentes del sistema informático 100 pueden estar integrados en un único paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una porción de los componentes del sistema informático 100 puede estar integrada en un módulo de múltiples chips (MCM), que puede estar interconectado con otros módulos de múltiples chips en 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, que incluye el número y disposición de puentes, el número del procesador y procesadores 102, y el número de procesador y procesadores paralelos 112, puede modificarse 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 mediante el concentrador de memoria 105 y el procesador o procesadores 102. En otras topologías alternativas, el procesador o procesadores paralelos 112 están conectados al concentrador de E/S 107 o directamente a uno del uno o más procesador o procesadores 102, en lugar de al concentrador de memoria 105. En otras realizaciones, el concentrador de E/S 107 y el concentrador de memoria 105 pueden estar integrados en un único chip. Algunas realizaciones pueden incluir dos o más conjuntos del procesador o procesadores 102 adjuntos mediante múltiples zócalos, que pueden acoplarse con dos o más instancias del procesador o procesadores paralelos 112.

Alguno de los componentes particulares mostrados en el presente documento es opcional y puede no estar incluido en todas las implementaciones del sistema informático 100. Por ejemplo, puede soportarse cualquier número de tarjetas o periféricos de adición, o pueden eliminarse algunos componentes. Adicionalmente, algunas arquitecturas pueden usar diferente terminología para componentes similares a aquellos ilustrados en la Figura 1. Por ejemplo, el concentrador de memoria 105 puede denominarse un puente norte en algunas arquitecturas, mientras que el concentrador de E/S 107 puede denominarse un puente sur.

La **Figura 2A** ilustra un procesador paralelo 200, de acuerdo con una realización. Los diversos componentes del procesador paralelo 200 pueden implementarse usando uno o más dispositivos de circuito integrado, tal como procesadores programables, circuitos integrados específicos de la aplicación (ASIC) o campos de matrices de puertas programables (FPGA). El procesador paralelo 200 ilustrado es una variante del uno o más procesador o 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 posibilita la comunicación con otros dispositivos, que incluyen otras instancias de la unidad de procesamiento paralelo 202. La unidad de E/S 204 puede estar conectada directamente a otros dispositivos. En una realización, la unidad de E/S 204 se conecta con otros dispositivos mediante el uso de una interfaz de concentrador o de conmutador, tal como un concentrador de memoria 105. Las conexiones entre el concentrador de memoria 105 y la unidad de E/S 204 forman un enlace de 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, donde la interfaz de anfitrión 206 recibe comandos dirigidos a realizar las operaciones de procesamiento y la barra transversal de memoria 216 recibe comandos dirigidos a realizar operaciones de memoria.

Cuando la interfaz de anfitrión 206 recibe una memoria intermedia de comando mediante la unidad de E/S 204, la interfaz de anfitrión 206 puede dirigir operaciones de trabajo para realizar aquellos comandos a un extremo delantero 208. En una realización, el extremo delantero 208 se acopla con un planificador 210, que está configurado para distribuir comandos u otros elementos de trabajo a una matriz de agrupación de procesamiento 212. En una realización, el planificador 210 garantiza que la matriz de agrupación de procesamiento 212 está configurada apropiadamente y en un estado válido antes de que se distribuyan las tareas a las agrupaciones de procesamiento de la matriz de agrupación de procesamiento 212. En una realización, el planificador 210 se implementa mediante lógica de firmware que se ejecuta en un microcontrolador. El planificador implementado por microcontrolador 210 puede configurarse para realizar operaciones de planificación compleja y distribución de trabajo en granularidad basta y precisa, lo que posibilita la anticipación rápida y la conmutación de contexto de hilos que se ejecutan en la matriz de procesamiento 212. En una realización, el software de anfitrión puede demostrar cargas de trabajo para la planificación en la matriz de procesamiento 212 mediante uno de múltiples timbres de procesamiento de tráfico. Las cargas de trabajo, a continuación, pueden distribuirse automáticamente a través de la matriz de procesamiento 212 por la lógica del planificador 210 dentro del microcontrolador planificador.

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

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

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

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

Durante la operación, la matriz de agrupación de procesamiento 212 puede recibir tareas de procesamiento para que se ejecuten mediante el 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 que van a procesarse, por ejemplo, datos de superficie (parche), datos de primitivas, datos de vértices y/o datos de píxeles, así como parámetros de estado y comandos que definen cómo han de procesarse los datos (por ejemplo, qué programa ha de ejecutarse). El planificador 210 puede estar configurado para extraer los índices que corresponden a las tareas o puede recibir los índices desde el extremo delantero 208. El extremo delantero 208 puede estar configurado para garantizar que la matriz de agrupación de procesamiento 212 está configurada en un estado válido antes de que se inicie la carga de trabajo especificada por las memorias intermedias de comando de entrada (por ejemplo, memorias intermedias de lote, memorias intermedias de inserción, etc.).

Cada una de la una o más instancias de la unidad de procesamiento paralelo 202 puede acoplarse con memoria de procesador paralelo 222. Puede accederse a la memoria de procesador paralelo 222 mediante la barra transversal de memoria 216, que puede recibir solicitudes de memoria desde la matriz de agrupación de procesamiento 212, así como de la unidad de E/S 204. La barra transversal de memoria 216 puede acceder a la memoria de procesador paralelo 222 mediante una interfaz de memoria 218. La interfaz de memoria 218 puede incluir múltiples unidades de subdivisión (por ejemplo, la unidad de subdivisión 220A, la unidad de subdivisión 220B a la unidad de subdivisión 220N) que cada una puede acoplarse a una porción (por ejemplo, la unidad de memoria) de la memoria de procesador paralelo 222. En una implementación, el número de unidades de subdivisión 220A-220N está configurado para que sea igual al número de unidades de memoria, de manera que una primera unidad de partición 220A tiene una correspondiente primera unidad de memoria 224A, una segunda unidad de partición 220B tiene una correspondiente unidad de memoria 224B y una unidad de subdivisión de orden N 220N tiene una correspondiente unidad de memoria de orden N 224N. 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, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria de acceso aleatorio de gráficos, tal como la memoria de acceso aleatorio de gráficos síncrona (SGRAM), que incluye la memoria de tasa de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 224A-224N pueden incluir también memoria 3D apilada, que incluye, pero sin limitación, memoria de ancho de banda alto (HBM). Los expertos en la materia apreciarán que la implementación específica de las unidades de memoria 224A-224N puede variar, y puede seleccionarse de uno de diversos diseños convencionales. Los objetivos de representación, tales como las memorias intermedias de fotograma o los mapas de textura pueden almacenarse a través de las unidades de memoria 224A-224N, permitiendo que las unidades de subdivisión 220A-220N escriban porciones de cada objetivo de representación en paralelo para usar de manera efectiva el ancho de banda disponible de la memoria de procesador paralelo 222. En algunas realizaciones, puede excluirse una instancia local de la memoria de procesador paralelo 222 en favor de un diseño de memoria unificado que utiliza memoria de sistema en conjunto con memoria caché local.

En una realización, una cualquiera de las agrupaciones 214A-214N de la matriz de agrupación de procesamiento 212 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 224A-224N dentro de la memoria de procesador paralelo 222. La barra transversal de memoria 216 puede estar configurada para transferir la salida de cada agrupación 214A-214N en cualquier unidad de subdivisión 220A-220N o en otra agrupación 214A-214N, que puede realizar operaciones de procesamiento adicionales en la salida. Cada agrupación 214A-214N puede comunicarse con la interfaz de memoria 218 a través de la barra transversal de memoria 216 para leer desde o escribir en diversos dispositivos de memoria externos. En 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, lo que posibilita que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 214A-214N se comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 202. En 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, puede incluirse cualquier número de instancias de la unidad de procesamiento paralelo 202. Por ejemplo, pueden proporcionarse múltiples instancias de la unidad de procesamiento paralelo 202 en una única tarjeta de adición, o pueden interconectarse múltiples tarjetas de adición. Las diferentes instancias de la unidad de procesamiento paralelo 202 pueden estar configuradas para inter-operar 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 más alta con relación a otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 202 o el procesador paralelo 200 pueden implementarse en una diversidad de configuraciones y factores de forma, incluyendo, pero sin limitación, sobremesa, portátil u ordenadores personales portátiles, servidores, estaciones de trabajo, consolas de juegos y/o sistemas integrados.

La **Figura 2B** es un diagrama de bloques de una unidad de subdivisión 220, 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 L2 221, una interfaz de memoria intermedia de fotograma 225 y una ROP 226 (unidad de operaciones de rasterización). La caché L2 221 es una caché de lectura/escritura que está configurada para realizar operaciones de carga y almacén recibidas desde la barra transversal de memoria 216 y la ROP 226. Los fallos de lectura y las solicitudes de escritura urgentes se emiten por la caché L2 221 a la interfaz de memoria intermedia de fotograma 225 para su procesamiento. Pueden enviarse también las actualizaciones a la memoria intermedia de fotograma mediante la interfaz de memoria intermedia de fotograma 225 para su procesamiento. En una realización, la interfaz de memoria intermedia de fotograma 225 interconecta con una de las unidades de memoria en la memoria de procesador paralelo, tal como las unidades de memoria 224A-224N de la Figura 2 (por ejemplo, dentro de la memoria de procesador paralelo 222).

En las aplicaciones de gráficos, la ROP 226 es una unidad de procesamiento que realiza operaciones de rasterización tales como estarcido, prueba z, mezcla y similares. La ROP 226 a continuación emite los datos de gráficos procesados que se almacenan en la 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 la 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 se realiza por la ROP 226 puede variar basándose en las características estadísticas de los datos que van a comprimirse. Por ejemplo, en una realización, se realiza compresión de color delta en datos de profundidad y de color en una base por pieza.

En algunas realizaciones, la ROP 226 está incluida 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, las solicitudes de lectura y escritura para datos de píxeles se transmiten a través de la barra transversal de memoria 216 en lugar de los datos de fragmento de píxel. Los datos de gráficos procesados pueden visualizarse en un dispositivo de visualización, tal como uno del uno o más dispositivo o dispositivos de visualización 110 de la Figura 1, 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 puede estar configurada para ejecutar muchos hilos en paralelo, donde el término "hilo" se refiere a una instancia de un programa particular que se ejecuta en un conjunto particular de datos de entrada. En algunas realizaciones, se usan técnicas de emisión de instrucción 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, donde todos los motores de procesamiento típicamente ejecutan instrucciones idénticas, la ejecución de SIMT permite que diferentes hilos sigan más fácilmente rutas de ejecución divergentes a través de un programa de hilos dado. Los expertos en la materia entenderán que un régimen de procesamiento de SIMD representa un subconjunto funcional de un régimen de procesamiento de SIMT.

La operación de la agrupación de procesamiento 214 puede controlarse mediante un gestor de tubería 232 que distribuye las tareas de procesamiento a procesadores paralelos de SIMT. El gestor de tubería 232 recibe instrucciones desde el planificador 210 de la Figura 2 y gestiona la ejecución de estas instrucciones mediante 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, pueden incluirse diversos tipos de procesadores paralelos de SIMT de arquitecturas diferentes dentro de la agrupación de procesamiento 214. Puede incluirse 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 puede usarse una barra transversal de datos 240 para distribuir los datos procesados a uno de múltiples posibles destinos, que incluyen otras unidades sombreadoras. El gestor de tubería 232 puede facilitar la distribución de datos procesados especificando destinos para que se distribuyan datos procesados mediante la barra transversal de datos 240.

Cada multiprocesador de gráficos 234 dentro de la agrupación de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades aritmético-lógicas, unidades de carga-almacén, etc.). La lógica de ejecución funcional puede configurase en forma en tubería en la que pueden emitirse nuevas instrucciones antes de que estén completadas instrucciones anteriores. La lógica de ejecución funcional soporta una diversidad de operaciones que incluyen aritmética de números enteros y de coma flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y del cálculo de diversas funciones algebraicas. En una realización, puede aprovecharse el mismo hardware funcional-unitario para realizar diferentes operaciones y puede estar presente cualquier combinación de unidades funcionales.

Las instrucciones transmitidas a la agrupación de procesamiento 214 constituyen un hilo. Un conjunto de hilos que se ejecutan a través del conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa en diferentes datos de entrada. Cada hilo dentro de un grupo de hilos puede asignarse 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 puede encontrarse en espera 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, puede realizarse el procesamiento a través de ciclos de reloj consecutivos. En una realización, pueden ejecutarse múltiples grupos de hilos concurrentemente en un multiprocesador de gráficos 234.

En una realización, el multiprocesador de gráficos 234 incluye una memoria de caché interna para realizar operaciones de carga y almacén. En una realización, el multiprocesador de gráficos 234 puede prescindir de una caché interna y usar una memoria caché (por ejemplo, la caché L1 308) dentro de la agrupación de procesamiento 214. Cada multiprocesador de gráficos 234 también tiene acceso a cachés 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 pueden usarse 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. Puede usarse cualquier memoria externa a la unidad de procesamiento paralelo 202 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 pueden almacenarse en la caché L1 308.

Cada agrupación de procesamiento 214 puede incluir una MMU 245 (unidad de gestión de memoria) que está configurada para mapear direcciones virtuales en direcciones físicas. En 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 mapear una dirección virtual a una dirección física de una pieza (más información sobre la aplicación de piezas) y, opcionalmente, un índice de línea de caché. La MMU 245 puede incluir memorias intermedias de traducción adelantada (TLB) de dirección o cachés que pueden residir dentro del multiprocesador de gráficos 234 o la caché L1 o la agrupación de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso de datos de superficie para permitir una intercalación de solicitud eficaz entre unidades de subdivisión. El índice de línea de caché puede usarse para determinar si una solicitud para una línea de caché es un acierto o un fallo.

En aplicaciones de gráficos e informáticas, puede configurarse una agrupación de procesamiento 214 de manera que cada multiprocesador de gráficos 234 esté acoplado a una unidad de textura 236 para realizar operaciones de mapeo de textura, por ejemplo, determinar posiciones de muestra de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen desde una caché L1 de textura interna (no mostrada) o, en algunas realizaciones, desde la caché L1 dentro del multiprocesador de gráficos 234 y se extraen desde una caché L2, memoria de procesador paralelo local o memoria de sistema, según sea necesario. Cada multiprocesador de gráficos 234 emite tareas procesadas a la barra transversal de datos 240 para proporcionar la tarea procesada a otra agrupación de procesamiento 214 para su procesamiento adicional o para almacenar la tarea procesada en una caché L2, memoria de procesador paralelo local o memoria de sistema mediante la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones previas a la rasterización) está configurada para recibir datos desde el multiprocesador de gráficos 234, dirigir datos a las unidades de ROP, que pueden estar ubicadas con unidades de subdivisión como se describe en el presente documento (por ejemplo, las unidades de subdivisión 220A-220N de la Figura 2). La unidad 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 modificaciones. Puede incluirse 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 únicamente se muestra una agrupación de procesamiento 214, la unidad de procesamiento paralelo, como se describe en el presente documento, puede incluir cualquier número de instancias de la agrupación de procesamiento 214. En una realización, cada agrupación de procesamiento 214 puede estar configurada para operar independientemente de otras agrupaciones de procesamiento 214 usando unidades de procesamiento separadas y distintas, cachés L1, etc.

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

En una realización, la caché de instrucciones 252 recibe un flujo de instrucciones para ejecutarse desde el gestor de tubería 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, envolturas), 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 mapeo de direcciones 256 puede usarse para traducir direcciones en el espacio de direcciones unificado en una dirección de memoria distinta que puede accederse por las unidades de carga/almacén 266.

El fichero de registro 258 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de gráficos 324. El fichero 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/almacén 266) del multiprocesador de gráficos 324. En una realización, el fichero de registro 258 se divide entre cada una de las unidades funcionales de manera que cada unidad funcional está asignada a una porción especializada del fichero de registro 258. En una realización, el fichero de registro 258 se divide entre las diferentes envolventes que se ejecutan por el multiprocesador de gráficos 324.

Los núcleos de GPGPU 262 puede cada uno incluir unidades de coma flotante (FPU) y/o unidades aritmético-lógicas (ALU) de números enteros que se usan para ejecutar instrucciones del multiprocesador de gráficos 324. Los núcleos de GPGPU 262 pueden ser similares en arquitectura o pueden diferir en arquitectura, de acuerdo con las 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 posibilitar aritmética de coma flotante de precisión variable. El multiprocesador de gráficos 324 puede incluir adicionalmente una o más unidades de función fija o de función especial para realizar funciones específicas, tales como operaciones de copia de rectángulo o de mezcla de píxeles. En una realización, uno o más de los núcleos de GPGPU puede incluir también lógica de función fija o especial.

En una realización, los núcleos de GPGPU 262 incluyen lógica de SIMD que puede realizar una instrucción sencilla en múltiples conjuntos de datos. En una realización, los núcleos de GPGPU 262 pueden ejecutar físicamente instrucciones SIMD4, SIMD8 y SIMD16 y ejecutar lógicamente instrucciones SIMD1, SIMD2 y SIMD32. Las instrucciones SIMD para los núcleos de GPGPU pueden generarse en el momento de compilación por un compilador de sombreador o generarse automáticamente cuando se ejecutan programas escritos y compilados para arquitecturas de datos de múltiples programas sencillos (SPMD) o SIMT. Pueden ejecutarse múltiples hilos de un programa configurado para el modelo de ejecución de SIMT mediante una instrucción de SIMD sencilla. Por ejemplo, y en una realización, ocho hilos SIMT pueden realizar las mismas operaciones o similares que pueden ejecutarse en paralelo mediante una unidad lógica SIMD8 sencilla.

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

Las Figuras 3A-3B ilustran multiprocesadores de gráficos adicionales, de acuerdo con las 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 pueden estar configurados como un multiprocesador de envío por flujo continuo (SM) que puede realizar la ejecución simultánea de un gran número de hilos de ejecución.

La Figura 3A muestra un multiprocesador de gráficos 325 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 fichero 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 gráficos o de cálculo (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/almacén 340A-340B. En una realización,

las unidades de recurso de ejecución tienen una caché de instrucciones común 330, memoria caché de textura y/o de datos 342 y memoria compartida 346.

Los diversos componentes pueden comunicarse mediante un tejido de interconexión 327. En una realización, el tejido de interconexión 327 incluye uno o más conmutadores de barra transversal para posibilitar 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 en la que se apila cada componente del multiprocesador de gráficos 325. Los componentes del multiprocesador de gráficos 325 se comunican con componentes remotos mediante el tejido de interconexión 327. Por ejemplo, cada uno de los núcleos de GPGPU 336A-336B, 337A-337By 3378A-338B puede comunicarse con la memoria compartida 346 mediante el 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 los 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, donde cada conjunto de recursos de ejecución incluye múltiples unidades de instrucción, ficheros de registro, núcleos de GPGPU y unidades de carga-almacén, como se ilustra en la Figura 2D y en la Figura 3A. Los recursos de ejecución 356A-356D pueden funcionar en conjunto con la unidad o unidades de textura 360A-360D para operaciones de textura, mientras que comparten una caché de instrucciones 354 y la 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 pueden comunicarse mediante 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 pueden implementarse en cualquier unidad de procesamiento configurada apropiadamente, que incluye, sin limitación, uno o más procesadores de aplicación móvil, una o más unidades de procesamiento central (CPU) de sobremesa o servidor que incluyen CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tal como la unidad de procesamiento paralelo 202 de la Figura 2, así como uno o más procesadores de gráficos o unidades de procesamiento de fin especial, sin alejarse 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 fin general (GPGPU). La GPU puede estar acoplada 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 puede estar integrada 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, interna 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, a continuación, usa circuitería/lógica especializada para el procesamiento de manera eficaz de 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 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 mayor, dependiendo de la implementación. Pueden usarse diversos protocolos de interconexión que incluyen, pero sin limitación, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no están limitados a ningún protocolo o caudal de comunicación particular.

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

En una realización, cada procesador de múltiples núcleos 405-406 está acoplado de manera comunicativa a una memoria de procesador 401-402, mediante las interconexiones de memoria 430-431, respectivamente, y cada GPU

410-413 está acoplada de manera comunicativa a la memoria de la GPU 420-423 a través de las interconexiones de memoria de GPU 450-453, respectivamente. Las interconexiones de memoria 430-431 y 450-453 pueden utilizar las mismas tecnologías de acceso a memoria o diferentes. A modo de ejemplo, y no como limitación, las memorias de procesador 401-402 y las memorias de GPU 420-423 pueden ser memorias volátiles, tal como memorias de acceso aleatorio dinámicas (DRAM) (que incluyen 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 GPU 410-413 pueden estar físicamente acoplados a una memoria particular 401-402, 420-423, respectivamente, puede implementarse una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtual (también denominado espacio "de direcciones eficaz") está distribuido entre todas las diversas memorias físicas. Por ejemplo, las memorias de procesador 401-402 puede cada una comprender 64 GB de la memoria del espacio de direcciones de sistema y las memorias de GPU 420-423 puede cada una comprender 32 GB de la memoria del espacio de direcciones de sistema (dando como resultado un total de 256 GB de memoria direccionable en este ejemplo).

La **Figura 4B** ilustra detalles adicionales para una interconexión entre un procesador de múltiples núcleos 407 y un módulo de aceleración de gráficos 446 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 mediante el enlace de alta velocidad 440. Como alternativa, el módulo de aceleración de gráficos 446 puede estar integrado en el mismo paquete o chip que el procesador 407.

El procesador ilustrado 407 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 oscurecer los principios subyacentes de la invención (por ejemplo, unidades de extracción de instrucción, unidades de predicción de ramal, decodificadores, unidades de ejecución, memorias intermedias de reordenación, etc.). Las cachés 462A-462D pueden comprender cachés de nivel 1 (LI) y de nivel 2 (L2). Además, puede incluirse una o más cachés compartidas 426 en la jerarquía de almacenamiento en caché y compartirse por conjuntos de los núcleos 460A-460D. Por ejemplo, una realización del procesador 407 incluye 24 núcleos, cada uno con su propia caché L1, doce cachés L2 compartidas y doce cachés L3 compartidas. En esta realización, una de las cachés L2 y L3 está 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 los datos e instrucciones almacenados en las diversas cachés 462A-462D, 456 y en la memoria de sistema 441 mediante la comunicación inter-núcleo a través de un bus de coherencia 464. Por ejemplo, cada caché puede tener una lógica/circuitería de coherencia de caché asociada con la misma para comunicarse a través del bus de coherencia 464 en respuesta a lecturas o escrituras detectadas en líneas de caché particulares. En una implementación, se implementa un protocolo de monitorización de caché a través del bus de coherencia 464 para monitorizar los accesos de caché. Las técnicas de monitorización/coherencia de caché son bien entendidas por los expertos en la materia y no se describirán en detalle en este punto para evitar oscurecer los principios subyacentes de la invención.

En una realización, un circuito de intermediario 425 se acopla comunicativamente al módulo de aceleración de gráficos 446 al bus de coherencia 464, lo que permite que el módulo de aceleración de gráficos 446 participe en el protocolo de coherencia de caché como un par de los núcleos. En particular, una interfaz 435 proporciona conectividad al circuito de 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 440.

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

En 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 efectiva a real) y protocolos de acceso a 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/efectiva 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 ha mencionado, esto puede conseguirse mediante el circuito de 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 las modificaciones/accesos de líneas de caché en las cachés del procesador 462A-462D, 456 y recibiendo actualizaciones de la caché 438).

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

En una implementación, las direcciones virtuales/efectivas de un motor de procesamiento de gráficos 431 se traducen a direcciones reales/físicas en memoria de sistema 411 por la MMU 439. 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 puede estar especializado a una única aplicación ejecutada en el procesador 407 o puede compartirse 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 pueden subdividirse en "cortes" que se asignan a diferentes VM y/o aplicaciones basándose en los requisitos de procesamiento y las propiedades asociadas con las VM y/o las aplicaciones.

Por 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 mapean explícitamente al 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 efectivo. 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 al sistema como unidades independientes.

Como se ha mencionado, 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 se procesan 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 440, se usan técnicas de desvío para garantizar que los datos almacenados en las memorias de gráficos 433-434, M son datos que se usarán de manera más frecuente por los motores de procesamiento de gráficos 431-432, N y, preferentemente, no se usarán por los núcleos 460A-460D (al menos no de manera frecuente). De manera similar, el mecanismo de desvío intenta mantener los datos necesarios por los núcleos (y, preferentemente, no 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 mediante la interfaz 437 y la interfaz 435 (que, de nuevo, puede utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración de acelerador 436 puede realizar las mismas operaciones que aquellas descritas con respecto a la **Figura 4B,** pero potencialmente a un caudal superior dada su proximidad cercana 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 especializado (sin virtualización de módulo de aceleración de gráficos) y modelos de programación compartida (con virtualización). El último puede incluir modelos de programación que se controlan por el circuito de integración de acelerador 436 y modelos de programación que se controlan por el módulo de aceleración de gráficos 446.

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

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

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

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

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

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

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

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

Tabla 1 - Registros inicializados por el hipervisor

1	Registro de control de corte
2	Puntero de área de procesos planificados de dirección real (RA)
3	Registro de anulación de máscara de autoridad

4	Desplazamiento de entrada de tabla de vector de interrupción
5	Límite de entrada de tabla de vector de interrupción
6	Registro de estado
7	ID de partición lógica
8	Puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

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

Tabla 2 - Registros inicializados por el sistema operativo

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

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

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

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

En este modelo, el hipervisor de sistema 496 tiene propiedad del módulo de aceleración de gráficos 446 y pone a disposición su función a 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 adherirse a los siguientes requisitos: 1) Una solicitud de trabajo de la aplicación debe ser autónoma (es decir, el estado no necesita mantenerse entre trabajos), o el módulo de aceleración de gráficos 446 debe proporcionar un mecanismo de grabación y restauración de contexto. 2) Se garantiza una solicitud de trabajo de la aplicación por el módulo de aceleración de gráficos 446 para completar en una cantidad especificada de tiempo, que incluye cualquier fallo de traducción, o el módulo de aceleración de gráficos 446 proporciona la capacidad de anticiparse al procesamiento del trabajo. 3) El módulo de aceleración de gráficos 446 debe garantizar equidad entre procesos cuando 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 a un sistema operativo 495 con un tipo de módulo de aceleración de gráficos 446, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de grabación/restauración de contexto (CSRP). El tipo de módulo de aceleración de gráficos 446 describe la función de aceleración dirigida para la llamada de sistema. El tipo de módulo de aceleración de gráficos 446 puede ser un valor específico de sistema. El WD se formatea específicamente para el módulo de aceleración de gráficos 446 y puede estar en forma de un comando de módulo de aceleración de gráficos 446, un puntero de dirección efectiva a una estructura definida por el usuario, un puntero de dirección efectiva a una cola de comandos o cualquier otra estructura de datos para describir el trabajo que va a hacerse por el módulo de aceleración de gráficos 446. En una realización, el valor de AMR es el estado de AMR para su uso para el proceso actual. El valor pasado al sistema operativo es similar a una aplicación que configura el AMR. Si las implementaciones del circuito de integración del 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 del 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 445 que contiene la dirección

efectiva de un área en el espacio de direcciones de la aplicación 482 para que el módulo de aceleración de gráficos 446 grabe y restaure el estado de contexto. Este puntero es opcional si no se requiere que se grabe estado entre trabajos o cuando se anticipa un trabajo. El área de grabación/restauración de contexto puede estar fijada en la memoria de sistema.

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

Tabla 3 - Parámetros de llamada de SO al hipervisor

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

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

Tabla 4 - Información de elemento de proceso

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

En una realización, el hipervisor inicializa una pluralidad de registros 445 de corte de integración del acelerador 490.

Como se ilustra en la **Figura 4F,** una realización de la invención emplea una memoria unificada direccionable mediante un espacio de direcciones virtual de memoria común usado para acceder a las memorias de procesador físico 401-402 y a las memorias de GPU 420-423. En esta implementación, las operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de direcciones de memoria virtual/efectiva 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/efectivo está asignada a la memoria del procesador 401, una segunda porción a la memoria del segundo procesador 402, una tercera porción a la memoria de GPU 420 y así sucesivamente. El espacio de memoria virtual/efectivo total (en ocasiones denominado el espacio de direcciones efectivo) está distribuido de esta manera a través de cada una de las memorias de procesador 401-402 y las memorias de GPU 420-423, lo que permite que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual mapeada a 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 desvío que indican las memorias físicas en las que deben almacenarse ciertos tipos de datos. Aunque se ilustran múltiples casos de circuitería de gestión de desvío/coherencia 494A-494E en la **Figura 4F,** la circuitería de desvío/coherencia puede implementarse 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 adjunta a la GPU 420-423 se mapee como parte de memoria de sistema, y se acceda usando tecnología de memoria virtual compartida (SVM), pero sin sufrir las desventajas de rendimiento típicas asociadas con la coherencia de caché de sistema completa. La capacidad de que se acceda a la memoria adjunta a la GPU 420-423 como memoria de sistema sin sobrecarga de coherencia de caché onerosa proporciona un entorno de operación beneficioso para la descarga de la GPU. Esta disposición permite que el software de procesador de anfitrión 405 configure operandos y acceda a resultados de cálculo, sin la sobrecarga de las copias de datos de acceso a memoria directo (DMA) de E/S tradicionales. Tales copias tradicionales implican llamadas de controlador, interrupciones y accesos de E/S de memoria mapeada (MMIO) que son todos ineficaces con relación a los accesos de memoria sencillos. Al mismo tiempo, la capacidad de acceder a la memoria adjunta a la GPU 420-423 sin sobrecargas de coherencia de caché puede ser critica para el tiempo de ejecución de un cálculo descargado. En casos con tráfico de memoria de escritura de envío por flujo continuo sustancial, por ejemplo, la sobrecarga de coherencia de caché puede reducir significativamente el ancho de banda de escritura efectivo observado por una GPU 410-413. La eficacia de la configuración del operando, la eficacia del acceso a los resultados y la eficacia del cálculo de GPU, todos desempeñan un papel al determinar la efectividad de la descarga de la GPU.

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

En una implementación, la entrada de tabla de desvío asociada con cada acceso a la memoria adjunta a la GPU 420-423 se accede antes del acceso real a la memoria de GPU, lo que provoca las siguientes operaciones. En primer lugar, las solicitudes locales de la GPU 410-413 que encuentran su página en el desvío de la GPU se reenvían directamente a una correspondiente memoria de GPU 420-423. Las solicitudes locales de la GPU que encuentran su página en el desvío 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 el desvío del procesador de anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, las solicitudes dirigidas a una página desviada de la GPU pueden redirigirse a la GPU 410-413. La GPU puede pasar, a continuación, la página a un desvío de procesador de anfitrión si no está usando actualmente la página.

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

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

En una realización, se mantiene la coherencia de caché representando temporalmente las páginas adjuntas en la GPU que no pueden almacenare en caché por el procesador de anfitrión 405. Para acceder a estas páginas, el procesador 405 puede solicitar acceso desde la GPU 410 que puede conceder o no el acceso de inmediato, 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 GPU de desvío sean aquellas que se requieren por la GPU, pero no por el procesador de anfitrión 405 y viceversa.

Tubería de procesamiento de gráficos

La **Figura 5** ilustra una tubería de procesamiento de gráficos 500, de acuerdo con una realización. En una realización, un procesador de gráficos puede implementar la tubería de procesamiento de gráficos 500 ilustrada. El procesador de gráficos puede estar incluido dentro de los subsistemas de procesamiento paralelo como se describe en el presente documento, tal como el procesador paralelo 200 de la Figura 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 tubería de procesamiento de gráficos 500 mediante una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 202 de la Figura 2) como se describe en el presente documento. Por ejemplo, una unidad sombreadora (por ejemplo, el multiprocesador de gráficos 234 de la Figura 3) puede estar configurada para realizar las funciones de una o más de una unidad de procesamiento de vértices 504, una unidad de

control de proceso de teselación 508, una unidad de procesamiento de evaluación de teselación 512, una unidad de procesamiento de geometría 516 y una unidad de procesamiento de fragmento/píxel 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 del rasterizador 526 pueden realizarse también por otros motores de procesamiento dentro de una agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 214 de la Figura 3) y una correspondiente unidad de subdivisión (por ejemplo, la unidad de subdivisión 220A-220N de la Figura 2). La tubería de procesamiento de gráficos 500 puede implementarse también usando unidades de procesamiento especializadas para una o más funciones. En una realización, puede realizarse una o más porciones de la tubería de procesamiento de gráficos 500 mediante lógica de procesamiento paralelo dentro de un procesador de fin general (por ejemplo, la CPU). En una realización, una o más porciones de la tubería de procesamiento de gráficos 500 pueden acceder a memoria en el chip (por ejemplo, la memoria de procesador paralelo 222 como en la Figura 2) mediante 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 vértices para superficies y primitivas. El ensamblador de datos 502, a continuación, emite los datos de vértice, que incluyen los atributos de vértices, a la unidad de procesamiento de vértices 504. La unidad de procesamiento de vértices 504 es una unidad de ejecución programable que ejecuta programas del sombreador de vértices, datos de vértices de iluminación y transformación como se especifica por los programas del sombreador de vértices. La unidad de procesamiento de vértices 504 lee datos que se almacenan en caché, en memoria local o de sistema para su uso al procesar los datos de vértice y puede estar programada para transformar los datos de vértice de una representación de coordenadas basada en objeto a un espacio de coordenadas del 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 son soportados por diversas interfaces de programación de aplicación (API) de procesamiento de gráficos.

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

Una segunda instancia de un ensamblador de primitivas 514 recibe atributos de vértices desde la unidad de procesamiento de evaluación de teselación 512, que lee los atributos de vértices almacenados 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 sombreadores de geometría para transformar primitivas de gráficos recibidas desde el ensamblador de primitivas 514 como se especifica por los programas sombreadores 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 nuevas primitivas de gráficos al ensamblador de primitivas 518. El ensamblador de primitivas 518 recibe los parámetros y vértices desde la unidad de procesamiento de geometría 516 y construye primitivas de gráficos para su procesamiento por una unidad de escala, selección y recorte de ventana gráfica 520. La unidad de procesamiento de geometría 516 lee datos que están almacenados en la memoria de procesador paralelo o en la memoria de sistema para su uso al procesar los datos de geometría. La unidad de escala, selección y recorte de ventana gráfica 520 realiza el recorte, selección y escalado de ventana gráfica y emite las primitivas de gráficos procesados a un rasterizador 522.

El rasterizador 522 puede realizar optimizaciones de selección de profundidad y otras basadas en la profundidad. El rasterizador 522 también realiza la conversión de exploración en las nuevas primitivas de gráficos para generar fragmentos y emitir aquellos 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 sombreadores de fragmentos o programas sombreadores de píxeles. La unidad de procesamiento de fragmentos/píxeles 524 que transforma fragmentos o píxeles recibidos

desde el rasterizador 522, como se especifica por los programas sombreadores de fragmentos o de píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 puede estar programada para realizar operaciones que incluyen, pero sin limitación, mapeo de textura, sombreado, mezcla, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se emiten a una unidad de operaciones de rasterización 526. La unidad de procesamiento de fragmentos/píxeles 524 puede leer datos que se almacenan en cualquiera de la memoria de procesador paralelo o la memoria de sistema para su uso cuando se procesan los datos de fragmento. Los programas sombreadores de fragmento o de píxeles pueden estar configurados para sombrear a granularidad de muestra, de píxel, de pieza 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 píxeles como datos de gráficos procesados para que se almacenen en la memoria de gráficos (por ejemplo, la memoria de procesador paralelo 222 como en la Figura 2, y/o la memoria de sistema 104 como en la Figura 1, para que se visualicen en el uno o más dispositivo o dispositivos de visualización 110 o para su procesamiento adicional por uno del uno o más procesador o procesadores 102 o procesador o procesadores paralelos 112. En 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 la memoria.

Vista general de aprendizaje automático

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

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

Antes de que pueda usarse un algoritmo de aprendizaje automático para modelar un problema particular, se entrena el algoritmo usando un conjunto de datos de entrenamiento. Entrenar una red neuronal implica seleccionar una topología de red, usando un conjunto de datos de entrenamiento que representan un problema que se está modelando por la red, y ajustando los pesos hasta que el modelo de red rinda con un error mínimo para todos los casos 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, representando una señal de error la diferencia entre la salida y se calcula la salida etiquetada, y se ajustan los pesos asociados con las conexiones para minimizar ese error a medida que la señal de error se propaga hacia atrás a través de las capas de la red. La red se considera "entrenada" cuando se minimizan los errores para cada una de las salidas generadas a partir de las instancias del conjunto de datos de entrenamiento.

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

La **Figura 6** es un diagrama generalizado de una pila de software de aprendizaje automático 600. Una aplicación de aprendizaje automático 602 puede estar configurada para entrenar una red neuronal usando un conjunto de datos de entrenamiento o para usar una red neuronal profunda entrenada para implementar la inteligencia automática. La aplicación de aprendizaje automático 602 puede incluir funcionalidad de entrenamiento e inferencia para una red

neuronal y/o software especializado que puede usarse para entrenar una red neuronal antes del despliegue. La aplicación de aprendizaje automático 602 puede implementar cualquier tipo de inteligencia automática que incluye, pero sin limitación, reconocimiento de imágenes, mapeo y ubicación, navegación autónoma, síntesis de voz, formación de imágenes médicas o traducción de idioma.

Puede posibilitarse la aceleración de hardware para la aplicación de aprendizaje automático 602 mediante 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 volvieran a optimizar a continuación la lógica computacional a medida que se desarrollaran nuevos procesadores paralelos. En su lugar, la aplicación de aprendizaje automático puede estar configurada para realizar los cálculos necesarios usando las primitivas proporcionadas por la estructura de aprendizaje automático 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 puede proporcionar también primitivas para implementar subprogramas de álgebra lineal básicos realizados por muchos algoritmos de aprendizaje automático, tales como operaciones matriciales y vectoriales.

La estructura de aprendizaje automático 604 puede procesar datos de entrada recibidos de la aplicación de aprendizaje automático 602 y genera la entrada apropiada a una estructura de cálculo 606. La estructura de cálculo 606 puede abstraer las instrucciones subyacentes proporcionadas al controlador de la GPGPU 608 para posibilitar que la estructura de aprendizaje automático 604 se aproveche de la aceleración de hardware mediante el hardware de GPGPU 610 sin requerir que la estructura de aprendizaje automático 604 tenga conocimiento íntimo de la arquitectura del hardware de GPGPU 610. Adicionalmente, la estructura de cálculo 606 puede posibilitar la aceleración de hardware para la estructura de aprendizaje automático 604 a través 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 fin general (GPGPU) 700 puede estar configurada para ser particularmente eficiente al procesar el tipo de cargas de trabajo computacionales asociadas con el entrenamiento de las redes neuronales profundas. Adicionalmente, la GPGPU 700 puede estar vinculada 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 posibilitar 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 estos comandos a un conjunto de agrupaciones de cálculo 706A-706H. Las agrupaciones de cálculo 706A-706H comparten una memoria caché 708. La memoria caché 708 puede servir como una caché de nivel más alto para memorias de caché dentro de las agrupaciones de cálculo 706A-706H.

La GPGPU 700 incluye la memoria 714A-714B acoplada con las agrupaciones de cálculo 706A-H mediante un conjunto de controladores de memoria 712A-712B. En diversas realizaciones, la memoria 714A-714B puede incluir diversos tipos de dispositivos de memoria, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria de acceso aleatorio de gráficos, tal como memoria de acceso aleatorio de gráficos síncrona (SGRAM), que incluye memoria de tasa de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 224A-224N pueden incluir también memoria 3D apilada, que incluye, pero sin limitación, memoria de ancho de banda alto (HBM).

En una realización, cada agrupación de cálculo 706A-706H incluye un conjunto de multiprocesadores de gráficos, tal como el multiprocesador de gráficos 400 de la Figura 4A. Los multiprocesadores de gráficos de la agrupación de cálculo tienen múltiples tipos de unidades de números enteros y de coma flotante que pueden realizar operaciones computacionales a un rango de precisiones que incluyen las adecuadas para los cálculos 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 cálculo 706A-H puede estar configurado para realizar operaciones de coma flotante de 16 bits o de 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante puede estar configurado para realizar operaciones de coma flotante de 64 bits.

Pueden configurarse múltiples instancias de la GPGPU 700 para operar como una agrupación de cálculo. El mecanismo de comunicación usado por la agrupación de cálculo para la sincronización y el intercambio de datos varía a través de 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 708 que acopla la GPGPU 700 con un enlace de GPU 710 que posibilita una conexión directa a otras instancias de la GPGPU. En una

realización, el enlace de la GPU 710 está acoplado a un puente de GPU a GPU especializado que posibilita la comunicación y sincronización entre múltiples instancias de la GPGPU 700. En una realización, el enlace de la 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 mediante un dispositivo de red que es accesible mediante la interfaz de anfitrión 702. En una realización, el enlace de la GPU 710 puede estar configurado para posibilitar 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 puede configurarse para entrenar redes neuronales, una realización proporciona una configuración alternativa de la GPGPU 700 que puede configurarse para el despliegue dentro de una plataforma de inferencia de alto rendimiento o de baja potencia. En una configuración de inferencia, la GPGPU 700 incluye menos de las agrupaciones de cálculo de las agrupaciones de cálculo 706A-H con relación a la configuración de entrenamiento. Adicionalmente, la tecnología de memoria asociada con la memoria 714A- 714B puede diferir entre las configuraciones de inferencia y entrenamiento. En una realización, la configuración de inferencia de la GPGPU 700 puede soportar las instrucciones específicas de inferencia. Por ejemplo, una configuración de inferencia puede proporcionar soporte para una o más instrucciones de producto vectorial de números enteros de 8 bits, que se usan comúnmente durante las operaciones de inferencia 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-D mediante 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 puede comunicarse con el conjunto de GPGPU 806A-D. Cada una de las múltiples GPGPU 806A-806D puede ser una instancia de la GPGPU 700 de la Figura 7. Las GPGPU 806A-D pueden interconectarse mediante 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 pueden conectarse a cada una de las GPGPU 806A-806D mediante un enlace de GPU especializado, tal como el enlace de GPU 710 como en la Figura 7. Los enlaces de GPU de P2P 816 posibilitan la comunicación directa entre cada una de las GPGPU 806A-D sin requerir la comunicación a través del bus de interfaz de anfitrión a la 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 a memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 800, por ejemplo, mediante uno o más dispositivos de red. Aunque en la realización ilustrada las GPGPU 806A-D se conectan al procesador 802 mediante el conmutador de interfaz de anfitrión 804, en una realización, el procesador 802 incluye el soporte directo para los enlaces de GPU de P2P 816 y puede conectarse directamente a las GPGPU 806A-806D.

Implementaciones de red neuronal de aprendizaje automático

La arquitectura informática proporcionada por las realizaciones descritas en el presente documento puede estar configurada para realizar los tipos de procesamiento paralelo que son particularmente adecuados para entrenar y desplegar redes neuronales para aprendizaje automático. Una red neuronal puede generalizarse como una red de funciones que tienen una relación de grafo. Como es bien conocido en la técnica, hay una diversidad de tipos de implementaciones de red neuronal usadas en el aprendizaje automático. Un tipo ilustrativo de red neuronal es la red proactiva, como se ha descrito anteriormente.

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

Las redes neuronales recurrentes (RNN) son una familia de las redes neuronales de proalimentación que incluyen conexiones de realimentación entre capas. Las RNN posibilitan el modelado de datos secuenciales compartiendo datos de parámetros a través de diferentes partes de la red neuronal. La arquitectura para una RNN incluye ciclos. Los ciclos representan la influencia de un valor presente de una variable de su propio valor en un tiempo futuro, ya que se usan al menos una porción de los datos de salida de la RNN como realimentación para su procesamiento de

entrada posterior en una secuencia. Esta característica hace a las RNN particularmente útiles para procesamiento de idioma debido a la naturaleza variable en la que pueden estar compuestos los datos de idioma.

Las figuras descritas a continuación presentan redes de proalimentación, CNN y RNN ilustrativas, así como describen un proceso general para entrenar y desplegar respectivamente cada uno de estos 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 pueden aplicarse en general a redes neuronales profundas y técnicas de aprendizaje automático en general.

Las redes neuronales ilustrativas anteriormente descritas pueden usarse para realizar aprendizaje profundo. El aprendizaje profundo es aprendizaje automático que usa redes neuronales profundas. Las redes neuronales profundas usadas en aprendizaje profundo son redes neuronales artificiales compuestas de múltiples capas ocultas, a diferencia de redes neuronales poco profundas que incluyen únicamente una sola capa oculta. Las redes neuronales más profundas son en general más intensivas computacionalmente de entrenar. Sin embargo, las capas ocultas adicionales de la red posibilitan un reconocimiento de patrón de múltiples etapas que da como resultado un error de salida reducido con relación a técnicas de aprendizaje automático poco profundas.

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

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

Las **Figuras 9A-B** 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 procesarse por múltiples capas convolucionales (por ejemplo, la capa convolucional 904, la capa convolucional 906). La salida de las múltiples capas convolucionales puede procesarse 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 anterior, como se ha descrito anteriormente para una red de proalimentación. La salida de las capas completamente conectadas 908 puede usarse para generar un resultado de salida de la red. Las activaciones dentro de las capas completamente conectadas 908 pueden calcularse usando una multiplicación matricial en lugar de la convolución. No todas las implementaciones de CNN hacen uso de capas completamente conectadas 908. Por ejemplo, en algunas implementaciones, la capa convolucional 906 puede generar la salida de la CNN.

Las capas convolucionales están conectadas de manera dispersa, 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 interactúa con cada unidad de entrada. Sin embargo, las capas convolucionales están conectadas de manera dispersa, puesto que se introduce la salida de la convolución de un campo (en lugar del respectivo valor de estado de cada uno de los nodos en el campo) a los nodos de la capa posterior, como se ilustra. Los núcleos asociados con las capas convolucionales realizan operaciones convolucionales, la salida de los que se envía a la siguiente capa. La reducción de la dimensionalidad realizada dentro de las capas convolucionales es un aspecto que posibilita que la CNN escale para procesar imágenes grandes.

La Figura 9B ilustra etapas de cálculo ilustrativas dentro de una capa convolucional de una CNN. La entrada a una capa convolucional 912 de una CNN puede procesarse en tres etapas de una capa convolucional 914. Las tres etapas pueden incluir una etapa convolucional 916, una etapa de detector 918 y una etapa de agrupación 920. La capa de convolución 914 puede emitir a continuación datos a una capa convolucional sucesiva. La capa convolucional final de la red puede generar datos de mapeo de característica de salida o proporcionar entrada a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

En la etapa de convolución 916 se realizan varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La etapa de convolución 916 puede incluir una transformación afín, que es cualquier transformación que puede especificarse como una transformación lineal más una traducción. Las transformaciones afines incluyen rotaciones, traducciones, escalamiento y combinaciones de estas transformaciones. La etapa de convolución calcula la salida de funciones (por ejemplo, neuronas) que están conectadas a regiones específicas en la entrada, que puede determinarse como la región local asociada con la neurona. Las neuronas calculan un producto vectorial entre los pesos de las neuronas y la región en la entrada local a la que están conectadas las neuronas. La salida de la etapa de convolución 916 define un conjunto de activaciones lineales que se procesan por etapas sucesivas de la capa de convolución 914.

Las activaciones lineales pueden procesarse por una etapa de detector 918. En la etapa de detector 918, cada activación lineal se procesa por una función de activación no lineal. La función de activación no lineal aumenta las propiedades no lineales de la red global sin afectar a los campos receptivos de la capa de convolución. Pueden usarse varios tipos de funciones de activación no lineal. Un tipo particular es la unidad lineal rectificada (ReLU), que usa una función de activación definida como $f(x) = \text{máx}(0, x)$, de manera que la activación tiene un umbral de cero.

La etapa de agrupación 920 usa una función de agrupación que sustituye la salida de la capa convolucional 906 con un sumario estadístico de las salidas cercanas. La función de agrupación puede usarse para introducir la invarianza de la traducción en la red neuronal, de manera que las traducciones pequeñas a la entrada no cambian las salidas agrupadas. La invarianza a la traducción 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. Pueden usarse diversos tipos de funciones de agrupación durante la etapa de agrupación 920, que incluye agrupación máxima, agrupación promedio y agrupación de norma l2. Adicionalmente, algunas implementaciones de CNN no incluyen una etapa de agrupación. En su lugar, tales implementaciones sustituyen una etapa de convolución adicional que tiene un paso mayor en relación con las etapas de convolución anteriores.

La salida de la capa convolucional 914 puede procesarse a continuación por la siguiente capa 922. La siguiente capa 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 emitirse a la segunda capa convolucional 906, mientras que la segunda capa convolucional puede emitirse 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 anterior de la red influencia la salida del estado actual de la red. Las RNN pueden crearse en una diversidad de maneras usando una diversidad de funciones. El uso de las RNN en general gira entorno al uso de modelos matemáticos para predecir el futuro basándose en una secuencia de entradas anterior. Por ejemplo, puede usarse una RNN para realizar modelado de idioma estadístico para predecir una palabra próxima dada en una secuencia de palabras anterior. La RNN ilustrada 1000 puede describirse 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 posibilitar una 'memoria' de estados anteriores y una capa de salida 1006 para emitir un resultado. La RNN 1000 opera basándose en pasos de tiempo. El estado de la RNN en un paso de tiempo dado se ve influenciada basándose en el paso de tiempo anterior mediante el mecanismo de realimentación 1005. Para un paso de tiempo dado, se define el estado de las capas ocultas 1004 por el estado anterior y la entrada en el paso de tiempo actual. Puede procesarse una entrada inicial (x_1) en un primer paso de tiempo por la capa oculta 1004. Puede procesarse una segunda entrada (x_2) por la capa oculta 1004 usando información de estado que se determina durante el procesamiento de la entrada inicial (x_1). Puede calcularse un estado dado como $s_t = f(Ux_t + Ws_{t-1})$, donde U y W son matrices de parámetros. La función f es en general una no linealidad, tal como la función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x) = \text{máx}(0, x)$. Sin embargo, la función matemática específica usada en las capas ocultas 1004 puede variar dependiendo de los detalles de la implementación específica de la RNN 1000.

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

La **Figura 11** ilustra el entrenamiento y despliegue de una red neuronal profunda. Una vez que se ha estructurado una red dada para una tarea, se entrena la red neuronal usando un conjunto de datos de entrenamiento 1102. Se han desarrollado diversas estructuras de entrenamiento 1104 para posibilitar la aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura de aprendizaje automático 604 de la Figura 6 puede estar configurada como una estructura de entrenamiento 604. La estructura de entrenamiento 604 puede engancharse a una red neuronal no

entrenada 1106 y posibilita que se entrene la red neuronal no entrenada usando los recursos de procesamiento paralelo descritos en el presente documento para generar una red neuronal entrenada 1108.

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

El aprendizaje supervisado es un método de aprendizaje en el que se realiza entrenamiento como una operación mediada, tal como cuando el conjunto de datos de entrenamiento 1102 incluye la entrada emparejada con la salida deseada para la entrada, o cuando el conjunto de datos de entrenamiento incluye la entrada que tiene la salida conocida y se clasifica manualmente la salida de la red neuronal. La red procesa las entradas y compara las salidas resultantes contra un conjunto de salidas esperadas o deseadas. Los errores a continuación se propagan de vuelta a través del sistema. La estructura de entrenamiento 1104 puede ajustar los pesos que controlan la red neuronal no entrenada 1106. La estructura de entrenamiento 1104 puede proporcionar herramientas para monitorizar 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 de manera repetitiva 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 puede a continuación desplegarse 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 incluirán datos de entrada sin ningún dato de salida asociado. 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 puede usarse para generar un mapa de autoorganización, que es un tipo de red neuronal entrenada 1107 que puede realizar operaciones útiles al reducir la dimensionalidad de los datos. El entrenamiento no supervisado puede usarse también para realizar detección de anomalías, que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

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

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

La **Figura 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 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 fin general, tales como la unidad de procesamiento de gráficos de fin general altamente paralela 700 como en la Figura 700. Como se ilustra, el aprendizaje distribuido puede realizarse en 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 cálculos de entrenamiento para diferentes partes de una única red. Por ejemplo, cada capa de una red neuronal puede entrenarse por un nodo de procesamiento diferente del sistema distribuido. Los beneficios del paralelismo de modelo incluyen la capacidad de escalar a modelos particularmente grandes. La división de los cálculos asociados con diferentes capas de la red neuronal posibilita el entrenamiento de redes neuronales muy grandes en las que los pesos para todas las capas no se ajustarían en la memoria de un único nodo computacional. En algunos casos, el paralelismo de modelo puede ser particularmente útil al realizar entrenamiento no supervisado de redes neuronales grandes.

En el paralelismo de datos 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 de los diferentes nodos a continuación se combinan. Aunque son posibles diferentes enfoques al paralelismo de datos, los enfoques de entrenamiento de datos paralelos todos requieren una técnica de combinación de resultados y de sincronización de los parámetros de modelo entre cada nodo. Los enfoques ilustrativos para combinar datos incluyen promedio de parámetros y paralelismo de datos basado en actualización. El promedio 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 de cada nodo. El promedio de parámetros usa un servidor de parámetros central que mantiene los datos de parámetros.

El paralelismo de datos basado en la actualización es similar al promedio de parámetros excepto 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 la actualización puede realizarse de una manera descentralizada, donde se comprimen las actualizaciones y se transfieren entre nodos.

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

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

Aplicaciones de aprendizaje automático ilustrativas

El aprendizaje automático puede aplicarse para resolver una diversidad de problemas tecnológicos, incluyendo, pero sin limitación, visión informática, conducción y navegación autónoma, reconocimiento del habla y procesamiento del idioma. La visión informática ha sido tradicionalmente una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Las aplicaciones de visión informática varían de reproducir capacidades visuales humanas, tales como reconocer caras, a crear nuevas categorías de capacidades visuales. Por ejemplo, las aplicaciones de visión informática pueden configurarse para reconocer ondas de sonido de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesador paralelo posibilita que se entrenen aplicaciones de visión informática usando un conjunto de datos de entrenamiento significativamente mayor que el previamente factible y posibilita que se desarrollen sistemas de inferencia usando procesadores paralelos de baja potencia.

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

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

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

Las plataformas de procesamiento paralelo usadas para aprendizaje automático pueden dividirse en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento son, en general, altamente paralelas e incluyen optimizaciones para acelerar el entrenamiento de nodo sencillo de múltiples GPU y entrenamiento de múltiples nodos de múltiples GPU. Los procesadores paralelos ilustrativos adecuados para entrenamiento incluyen la unidad de procesamiento de gráficos de fin 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 inferencia 1300 ilustrativo adecuado para realizar la inferencia 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 memoria en el chip 1305 que puede posibilitar una agrupación de datos en chip compartida que es accesible por cada uno de los componentes de procesamiento. Los componentes de procesamiento pueden optimizarse para la operación de baja potencia para posibilitar el despliegue a una diversidad de plataformas de aprendizaje automático, que incluyen vehículos autónomos y robots autónomos. Por ejemplo, puede usarse una implementación del SOC 1300 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 su cumplimiento con las normas de seguridad funcionales relevantes de la jurisdicción de despliegue.

Durante la operación, el procesador de medios 1302 y el procesador de visión 1304 pueden funcionar en conjunto para acelerar operaciones de visión informática. El procesador de medios 1302 puede posibilitar la decodificación de baja latencia de múltiples flujos de vídeo de alta resolución (por ejemplo, 4K, 8K). Los flujos de vídeo decodificados pueden escribirse en una memoria intermedia en la memoria en el chip 1305. El procesador de visión 1304 puede a continuación analizar el vídeo decodificado y realizar de manera preliminar las operaciones de procesamiento en los fotogramas del vídeo decodificado en preparación del procesamiento de los fotogramas usando un modelo de reconocimiento de imagen entrenado. Por ejemplo, el procesador de visión 1304 puede acelerar las operaciones convolucionales para una CNN que se usa para realizar el reconocimiento de imagen en los datos de vídeo de alta resolución, mientras se realizan cálculos de modelo de extremo trasero por la GPGPU 1306.

El procesador de múltiples núcleos 1308 puede incluir lógica de control para ayudar con la secuenciación y 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 puede funcionar también como un procesador de aplicación para ejecutar aplicaciones de software que pueden hacer uso de la capacidad de cálculo de inferencia de la GPGPU 1306. Por ejemplo, puede implementarse al menos una porción de la lógica de navegación y de conducción en software que se ejecuta en el procesador de múltiples núcleos 1308. Tal software puede emitir directamente cargas de trabajo computacionales a la GPGPU 1306 o pueden emitirse las cargas de trabajo computacionales al procesador de múltiples núcleos 1308, que puede descargar al menos una porción de estas operaciones a la GPGPU 1306.

La GPGPU 1306 puede incluir agrupaciones de cálculo, tal como una configuración de baja potencia de las agrupaciones de cálculo 706A-706H dentro de la unidad de procesamiento de gráficos de fin general altamente paralela 700. Las agrupaciones de cálculo dentro de la GPGPU 1306 pueden soportar instrucciones que están optimizadas específicamente para realizar cálculos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1306 puede soportar instrucciones para realizar cálculos de baja precisión tales como operaciones vectoriales de números enteros de 8 bits y 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 pueden usarse para abstraer muchos de los detalles computacionales subyacentes de la realización de los cálculos de aprendizaje automático. Las primitivas de alto nivel descritas en el presente documento posibilitan que la lógica de software solicite operaciones de aprendizaje automático de alto nivel mientras que se abstraen los detalles de implementación subyacentes de estas operaciones. Por ejemplo, y en una realización, la lógica de software puede solicitar una operación de convolución para una imagen usando un conjunto dado de filtros. Puede ejecutarse una única instrucción de alto nivel que tiene operandos para definir direcciones de memoria intermedia de entrada y salida y direcciones para memorias intermedias que almacenan datos de filtro y/o de núcleo. La GPGPU puede dividir, a continuación, la instrucción de convolución de alto nivel en múltiples suboperaciones que se realizan por las unidades de cálculo subyacentes de la GPGPU. En una realización, se proporciona soporte de hardware directo para una o más subrutinas de los subprogramas de algoritmo lineal básico (BLAS), aunque las realizaciones pueden proporcionar soporte de hardware para otras bibliotecas de subrutinas. La lógica del compilador y las bibliotecas de tiempo de ejecución asociadas pueden compilar código fuente que puede hacer uso de subrutinas de cálculo de alto nivel y emitir código fuente compilado que llama a una unidad de macroinstrucción de aprendizaje automático.

Lógica de aceleración de aprendizaje automático con operaciones de tubería de grano basto personalizadas

La **Figura 14** es un diagrama de bloques de un sistema de procesamiento de datos 1400, de acuerdo con una realización. El sistema de procesamiento de datos 1400 es un sistema de procesamiento heterogéneo que tiene procesador 1402, memoria unificada 1410 y una GPGPU 1420 que incluye lógica de aceleración de aprendizaje automático. El procesador 1402 y la GPGPU 1420 pueden ser cualquiera de los procesadores y GPGPU/procesadores paralelos como se describe en el presente documento. El procesador 1402 puede ejecutar instrucciones para un compilador 1415 almacenadas en memoria de sistema 1412. El compilador 1415 se ejecuta en el procesador 1402 para compilar el código fuente 1414A en código compilado 1414B. El código compilado 1414B puede incluir código que puede ejecutarse por el procesador 1402 y/o código que puede ejecutarse por la GPGPU 1420. Durante la compilación, el compilador 1415 puede realizar operaciones para insertar metadatos, que incluyen sugerencias en cuanto al nivel de paralelismo de datos presente en el código compilado 1414B y/o sugerencias con respecto a la localidad de datos asociada con los hilos que van a despacharse basándose en el código compilado 1414B. El compilador 1415 puede incluir la información necesaria para realizar tales operaciones o las operaciones pueden realizarse con la asistencia de una biblioteca de tiempo de ejecución 1416. La biblioteca de tiempo de ejecución 1416 puede facilitar también al compilador 1415 en la compilación del código fuente 1414A y puede incluir también instrucciones que están vinculadas en tiempo de ejecución con el código compilado 1414B para facilitar la ejecución de las instrucciones compiladas en la GPGPU 1420.

La memoria unificada 1410 representa un espacio de direcciones unificado al que puede accederse por el procesador 1402 y la GPGPU 1420. La memoria unificada incluye la memoria de sistema 1412, así como la memoria de GPGPU

1418. La memoria de GPGPU 1418 incluye memoria local de GPGPU 1434A-1434B dentro de la GPGPU 1420 y puede incluir también alguna o toda la memoria de sistema 1412. Por ejemplo, el código compilado 1414B almacenado en la memoria de sistema 1412 puede mapearse también en la memoria de GPGPU 1418 para su acceso por la GPGPU 1420.

La GPGPU 1420 incluye múltiples bloques de cálculo 1424A-1424N, que pueden ser instancias de la agrupación de cálculo 214A-214N de la **Figura 2A.** La GPGPU 1420 también incluye un conjunto de registros 1425, memoria caché 1427 y un módulo de potencia y rendimiento 1426 que pueden usarse como recursos compartidos para los bloques de cálculo 1424A-1424N. En una realización, los registros 1425 incluyen registros directa e indirectamente accesibles, donde los registros indirectamente accesibles están optimizados para su uso en operaciones de cálculo de matriz dispersa. El módulo de potencia y rendimiento 1426 puede estar configurado para ajustar la entrega de potencia y las frecuencias de reloj para los bloques de cálculo 1424A-1424N para alimentar los componentes de puerta en espera dentro de los bloques de cálculo 1424A-1424N bajo cargas de trabajo intensas. La GPGPU 1420 incluye la memoria local de GPGPU 1434A-1434B, que son módulos de memoria física que comparten una tarjeta de gráficos o un módulo de múltiples chips con la GPGPU 1420.

En una realización, la memoria local de GPGPU 1434A-1434B reside en un módulo de memoria híbrida 1430. El módulo de memoria híbrida 1430 incluye un conjunto de unidades de controlador de cálculo y de memoria 1432A-1432B que proporcionan tanto funcionalidad de controlador de memoria como de cálculo. Las unidades de controlador de cálculo y de memoria 1432A-1432B incluyen módulos lógicos que pueden realizar operaciones de cálculo de datos cercanos en datos directamente dentro de la memoria local de la GPGPU 1434A-1434B. Las unidades de controlador de cálculo y memoria 1432A-1432B pueden recibir la planificación directa de operaciones de límite de memoria desde el controlador planificador de aprendizaje automático 1422 o pueden recibir la descarga de estas operaciones desde la unidad del acelerador de cálculo disperso 1423 o los bloques de cálculo 1424A-1424B. En una realización, los controladores de cálculo y memoria 1432A-1432B incluyen lógica informática que puede realizar un subconjunto de las operaciones de cálculo que pueden realizarse por los bloques de cálculo 1424A-1424N. Por ejemplo, y en una realización, las unidades de controlador de cálculo y de memoria 1432A-1432B, además de las operaciones requeridas para realizar las operaciones del controlador de memoria, pueden estar configuradas para realizar un subconjunto de operaciones de cálculo que son específicamente útiles para operaciones de límite de memoria significativas, u operaciones en las que el ancho de banda de la memoria es más determinista del rendimiento que el caudal de cálculo. En una realización, la lógica de cálculo de las unidades de controlador de cálculo y de memoria 1432A-1432B pueden ser procesadores que tienen un conjunto de instrucciones diferente con relación al conjunto de instrucciones soportado por el bloque de cálculo 1424A-1424N. En una realización, el módulo de memoria híbrida 1430 se implementa mediante tecnología de apilamiento 3D para posibilitar que la memoria local de la GPGPU 1434A-1434B se apile verticalmente en la parte superior de las unidades de controlador de cálculo/memoria 1432A-1432B, que están acopladas mediante una interconexión a través de silicio de alto ancho de banda. La conexión de alto ancho de banda entre las unidades de controlador de cálculo y de memoria 1432A-1432B y la memoria local de la GPGPU 1434A-1434B puede posibilitar que se realicen de manera eficiente las operaciones de límite de memoria dentro del módulo de memoria híbrida 1430 usando unidades de cálculo de potencia inferior, a diferencia de ciclos de grandes cantidades de datos de la memoria local de la GPGPU 1434A-1434B dentro y fuera de la caché 1427 para su procesamiento mediante los bloques de cálculo 1424A-1424N.

En una realización, se posibilita el soporte de descarga de cálculo de datos cercanos mediante una extensión a la ISA soportada por los bloques de cálculo 1424A-1424N. En una realización, se posibilita el soporte de descarga de cálculo de datos cercanos por lógica de firmware ejecutada por el controlador planificador de aprendizaje automático 1422. Antes de ejecutar las operaciones de cálculo de datos cercanos, por ejemplo, mediante un núcleo de cálculo de datos cercanos, puede traducirse el conjunto de direcciones virtuales que va a accederse por el núcleo de cálculo de datos cercanos en direcciones físicas que son reconocibles por las unidades de controlador de cálculo y de memoria 1432A-1432B. La traducción de dirección puede realizarse por el núcleo antes de que el núcleo se despache o descargue a las unidades de controlador de memoria de cálculo 1432A-1432B.

En una realización, la GPGPU 1420 incluye lógica de aceleración de aprendizaje automático que incluye una unidad de extracción y decodificación de instrucciones de aprendizaje automático 1421, un controlador de planificador de aprendizaje automático 1422 y una unidad de acelerador de cálculo disperso 1423. La unidad de extracción y decodificación de instrucciones de aprendizaje automático 1421 es una unidad de extracción y decodificación que incluye lógica para extraer y decodificar instrucciones de aprendizaje automático que definen un comportamiento complejo personalizable. Las instrucciones pueden secuenciar y/o serializar, mediante el controlador planificador de aprendizaje automático 1422, un conjunto de instrucciones que van a realizarse mediante uno o más de los bloques de cálculo 1424A-1424N. En una realización, el controlador planificador de aprendizaje automático 1422 es un ASIC configurable para realizar operaciones de planificación avanzadas. En una realización, el controlador planificador de aprendizaje automático 1422 es un microcontrolador o un núcleo de procesamiento de baja energía por instrucción que puede realizar instrucciones cargadas desde un módulo de firmware.

En una realización, algunas funciones que van a realizarse por los bloques de cálculo 1424A-1424N pueden planificarse directamente o descargarse a la unidad del acelerador de cálculo disperso 1423. La unidad de acelerador de cálculo disperso 1423 incluye la lógica de elemento de procesamiento configurada para realizar de manera eficiente

operaciones en matrices dispersas. En una realización, la unidad del acelerador de cálculo disperso 1423 está configurada para realizar multiplicaciones matriciales para redes neuronales que tienen valores de peso dispersos. En una realización, la unidad del acelerador de cálculo disperso 1423 es un circuito integrado específico de la aplicación explícitamente configurado para realizar unas operaciones de multiplicación matriciales paralelas en las que uno o más operandos son matrices dispersas o muy dispersas. En una realización, la unidad del acelerador de cálculo disperso 1423 es un campo de matrices de puertas programables (FPGA) que proporciona lógica de función fija que puede actualizarse entre cargas de trabajo.

La **Figura 15A** ilustra detalles de la unidad de instrucción y extracción de aprendizaje automático 1421, de acuerdo con una realización. En una realización, la unidad de extracción y decodificación de instrucción de aprendizaje automático 1421 incluye una memoria caché 1502, una unidad de extracción de instrucción de aprendizaje automático 1504 y una unidad de decodificación de instrucción de aprendizaje automático 1506. La unidad de extracción de instrucción de aprendizaje automático 1504 puede extraer una o más macroinstrucciones de aprendizaje automático y almacenar las macroinstrucciones en la memoria caché 1502. La unidad de decodificación de instrucción de aprendizaje automático 1506 puede decodificar las macroinstrucciones de aprendizaje automático y, en respuesta, determinar un conjunto de operaciones para realizar. En una realización, la unidad de extracción y decodificación de instrucción de aprendizaje automático 1421 incluye un microcontrolador 1510 para posibilitar operaciones complejas, tales como seleccionar una de una pluralidad técnicas para usar para realizar una operación de aprendizaje automático específica, tal como una operación de convolución. El microcontrolador 1510 puede determinar también si realizar operaciones para una macroinstrucción mediante lógica programable dentro de la GPGPU o mediante lógica de aprendizaje automático de fin especial dentro de la GPGPU.

En una realización, el microcontrolador 1510 puede cargar lógica de firmware desde un módulo de firmware de aprendizaje automático 1508 para definir las operaciones para realizar en respuesta a una macroinstrucción de aprendizaje automático. En una realización, el módulo de firmware de aprendizaje automático 1508 puede actualizase mediante lógica de controlador de la GPGPU para expandir el conjunto de operaciones que se soportan mediante macroinstrucciones de aprendizaje automático y/o para expandir la capacidad de macroinstrucciones soportadas. En una realización, el microcontrolador 1510 posibilita el soporte explícito para operaciones convolucionales u otras operaciones relacionadas con la matriz o la red neuronal mediante lógica de aceleración de aprendizaje automático 1516.

Los cálculos para una CNN incluyen aplicar una operación matemática convolucional a cada filtro para producir la salida de ese filtro. Cada filtro es un núcleo con pesos entrenables que se convolucionan a través de la anchura y altura de un volumen de entrada para calcular productos vectoriales entre las entradas del filtro y la salida en cualquier posición. A medida que se convoluciona el filtro a través del volumen de entrada, se genera un mapa de activación bidimensional para indicar la respuesta del filtro en cada posición espacial. Se genera un mapa de activación para cada filtro aplicado al volumen de entrada. Los tamaños de filtro usados dentro de una CNN pueden variar basándose en los detalles de la implementación de la red neuronal.

En una realización, la lógica de análisis de parámetro 1512 puede analizar los parámetros para una operación de convolución solicitada. Una operación de convolución tiene dos entradas, los datos de entrada y el filtro convolucional. Los datos de entrada incluyen un lote de datos de imagen de $H \times W$ píxeles y el número C de mapas de características de entrada. El filtro convolucional tiene R filas y S columnas. En una realización, la lógica de análisis de parámetro 1512 determina, basándose en la dimensión $R \times S$ del filtro convolucional, si realizar al menos una porción de la convolución mediante lógica de convolución de fin especial, por ejemplo, si el tamaño del filtro de convolución indica que la convolución se realizaría de manera menos eficiente mediante la lógica programable de la GPGPU. En una realización, basándose en los parámetros convolucionales que incluyen la dimensión de filtro convolucional, la imagen de entrada o la dimensión de mapa de característica y las métricas operacionales actuales de la GPGPU, la lógica de aceleración de aprendizaje automático 1516 puede seleccionar un algoritmo para usar para realizar una operación convolucional solicitada.

Por ejemplo, la lógica de aceleración de aprendizaje automático 1516 puede estar configurada para seleccionar uno de varios posibles algoritmos para usar para implementar la convolución. En una realización, se realiza la convolución mediante una convolución basada en la Transformada Rápida de Fourier (FFT). La convolución de FFT usa el principio de que la multiplicación en el dominio de la frecuencia corresponde a la convolución en el dominio del tiempo. Por lo tanto, la transformada de Fourier de una convolución de dos funciones es el producto de las transformadas de Fourier de estas funciones. Los datos de entrada pueden transformarse en el dominio de la frecuencia usando una Transformada de Fourier discreta (DFT), multiplicada por la respuesta de frecuencia del filtro, y, a continuación, transformarse de vuelta en el dominio del tiempo usando la DFT inversa. Por ejemplo, y en una realización, para la convolución usando tamaños de filtro pequeños (por ejemplo, 1×1, 3×3), puede usarse el algoritmo de filtración mínimo de Winograd para realizar la convolución. Pueden realizarse tamaños de filtro mayores (por ejemplo, 4x4) mediante otros algoritmos FFT. La convolución para tamaños de filtro incluso mayores (5×5, 7×7) puede realizarse mediante hardware de convolución de función fija especializado. Como alternativa, puede realizarse la convolución directa en el dominio original de los datos usando operaciones de matriz en lotes mediante aceleración de hardware de subrutinas de multiplicación de matriz a matriz generales (GEMM).

La **Figura 15B** ilustra detalles de un controlador de planificador de aprendizaje automático, de acuerdo con una realización. El controlador planificador de aprendizaje automático 1422, en una realización, incluye un microcontrolador 1520 configurado para ejecutar instrucciones o comandos para posibilitar la lógica de gestión de planificación y tareas de aprendizaje automático 1526. La lógica de gestión de planificación y tareas de aprendizaje automático 1526 puede facilitar la planificación y anticipación de los diversos comandos de tubería e instrucciones que implementan las operaciones de aceleración de aprendizaje automático complejas descritas en el presente documento. La lógica de gestión de planificación y tareas de aprendizaje automático 1526 puede posibilitarse mediante instrucciones almacenadas en un módulo de firmware del planificador de aprendizaje automático 1518. Las instrucciones almacenadas en el módulo de firmware de planificador de aprendizaje automático 1518 pueden actualizarse en el campo para posibilitar la mejora y expansión de la capacidad del controlador planificador de aprendizaje automático 1422. El controlador planificador de aprendizaje automático 1422 puede incluir adicionalmente un controlador de interrupción 1519 para posibilitar que el controlador planificador de aprendizaje automático 1422 reciba y procese interrupciones de elementos de cálculo dentro del procesador de gráficos de fin general. Aunque se ilustra el controlador planificador de aprendizaje automático 1422 como que incluye un microcontrolador 1520, en una realización, el controlador de planificación de aprendizaje automático se implementa mediante un módulo de FPGA integrado dentro de la GPGPU.

La **Figura 16** ilustra operaciones convolucionales ilustrativas, de acuerdo con las realizaciones. Una memoria intermedia de volumen de entrada 1604 representa un canal 2D de datos de entrada. Aunque se ilustra la convolución 2D, la convolución puede realizarse también en un volumen tridimensional usando tres filtros dimensionales. Una pieza de campo receptivo 1602 destaca una porción del volumen de entrada. Se realiza un producto vectorial entre los datos dentro de la pieza de campo receptivo 1602 y un filtro convolucional para generar un punto de datos dentro de la memoria intermedia de salida 1606. La combinación de los puntos de datos dentro de la memoria intermedia de salida 1606 representa un mapa de activación generado por la convolución. Cada punto dentro del mapa de activación se genera deslizando la pieza de campo receptivo a través de la memoria intermedia de volumen de entrada 1604. Los datos de mapa de activación pueden introducirse a una función de activación para determinar un valor de activación de salida.

En una realización, se realiza la convolución de la memoria intermedia de volumen de entrada 1604 mediante un conjunto de operaciones de matriz de alto nivel 1605. Las operaciones de matriz de alto nivel pueden realizarse mediante operaciones primitivas, tales como una operación BLAS, que se acelera mediante macroinstrucciones que pueden decodificarse mediante la unidad de extracción y decodificación de instrucciones de aprendizaje automático 1421. La unidad de extracción y decodificación de instrucciones de aprendizaje automático 1421 puede despachar operaciones al controlador planificador de aprendizaje automático 1422 para su planificación. Las operaciones pueden planificarse, a continuación, al uno o más bloques de cálculo 1424A-1424N. Los bloques de cálculo 1424A-1424N pueden comunicarse con el módulo de memoria híbrida 1430 para almacenar datos en la memoria de gráficos local. Los bloques de cálculo 1424A-1424N pueden también descargar operaciones intensivas en memoria a los procesadores de cálculo de datos cercanos dentro del módulo de memoria híbrida 1430. En una realización, el controlador planificador de aprendizaje automático 1422 puede despachar operaciones de cálculo directamente al módulo de memoria híbrida 1430.

La **Figura 17** es un diagrama de flujo de lógica 1700 para realizar planificación de grano basto de operaciones de aprendizaje automático en una tubería de cálculo, de acuerdo con una realización. En una realización, la lógica 1700 puede implementarse mediante hardware dentro de la unidad de extracción y decodificación de instrucciones de aprendizaje automático 1421 y el controlador planificador de aprendizaje automático 1422 como en la Figura 14-Figura 15. La lógica 1700 puede extraer y decodificar una instrucción de cálculo de aprendizaje automático para que se ejecute dentro de la GPGPU, como se muestra en el bloque 1702. La instrucción de aprendizaje automático es una instrucción que especifica un conjunto de múltiples operaciones que van a realizarse por una tubería de cálculo dentro de un procesador de gráficos descrito en el presente documento. La instrucción de aprendizaje automático se decodifica en una instrucción de aprendizaje automático decodificada que está asociada con un conjunto de operaciones relacionadas con el aprendizaje automático. La instrucción de aprendizaje automático decodificada hace que la GPGPU realice una operación de aprendizaje automático compleja mediante las unidades de cálculo y los elementos de procesamiento del procesador de gráficos de fin general.

La lógica 1700 puede determinar un conjunto de comandos de tubería para realizar para ejecutar la instrucción de aprendizaje automático decodificada, como se muestra en el bloque 1704. Por ejemplo, la lógica de análisis de parámetro 1512 de la Figura 15 puede determinar un tipo o subtipo de operaciones de aprendizaje automático para realizar para la instrucción, mientras que lógica de aceleración de aprendizaje automático 1516 puede determinar un conjunto preciso de operaciones para realizar para ejecutar la instrucción de aprendizaje automático decodificada. Por ejemplo, y en una realización, la lógica 1700 puede determinar que la instrucción de aprendizaje automático es una instrucción de convolución para procesar una CNN. La lógica de aceleración de aprendizaje automático 1516 puede determinar, a continuación, las operaciones requeridas para realizar para posibilitar una implementación de convolución específica, así como el conjunto específico de comandos de tubería que puede usarse para implementar tales operaciones. Por ejemplo, el conjunto de operaciones puede ser un lote de operaciones de primitivas de multiplicación para ejecutar una operación de convolución a través de un conjunto de matrices.

Basándose en el conjunto de comandos de tubería determinado en el bloque 1704, la lógica 1700 puede planificar el conjunto de comandos de tubería a través de un conjunto de bloques de cálculo a la tubería de cálculo de la unidad de procesamiento de fin general para posibilitar la ejecución de la instrucción de aprendizaje automático decodificada, como se muestra en el bloque 1706. La lógica 1700 puede planificar el conjunto de comandos de tubería mediante una unidad planificadora, tal como el controlador planificador de aprendizaje automático 1422 de la Figura 14 y la Figura 15. La planificación puede incluir diversos comandos o instrucciones asociadas a diversos elementos de cálculo dentro de la tubería de cálculo. Los comandos pueden implementarse como instrucciones que van a ejecutarse mediante elementos de cálculo dentro de los bloques de cálculo (por ejemplo, los bloques de cálculo 1424A-1424N de la Figura 14) de la GPGPU. Los comandos también pueden ejecutarse como instrucciones realizadas mediante una unidad de acelerador de cálculo disperso 1423 o un módulo de memoria híbrida 1430 como en la Figura 14. Como alternativa, los comandos o instrucciones realizados mediante los bloques de cálculo pueden activar una descarga de instrucciones secundarias o comandos al uno o más de la unidad del acelerador de cálculo disperso 1423 y el módulo de memoria híbrida 1430, basándose en el tipo de operación que va a realizarse. Como se muestra en el bloque 1708, la lógica 1700 puede a continuación retirar la instrucción de aprendizaje automático decodificada en respuesta a una finalización del conjunto de comandos de tubería planificados en el bloque 1706.

Aceleración de aprendizaje automático usando cálculo de datos cercano

El cálculo de datos cercano es un paradigma computacional que puede implementarse en sistemas de procesamiento en los que un subconjunto de elementos de procesamiento dentro del sistema está configurado para tener ancho de banda de memoria significativamente superior con relación a otros sistemas informáticos dentro del sistema. Puede mejorarse significativamente el rendimiento para las operaciones de límite de memoria realizando tales operaciones en los elementos de cálculo que están 'cerca' de los datos en memoria, incluso si los elementos de cálculo de datos cercanos son menos complejos que otros elementos de cálculo. En algunas realizaciones, se posibilita el cálculo de datos cercano mejorando la lógica de controlador de memoria con la capacidad de realizar al menos un subconjunto de las operaciones de cálculo soportadas por los elementos de cálculo primarios dentro del sistema. En una realización, se posibilita el cálculo de datos cercano aumentando los controladores de memoria con núcleos de procesador de baja potencia eficientes que proporcionan una ISA de cálculo de datos cercanos. Los núcleos de procesador de baja potencia pueden recibir instrucciones desde una unidad planificadora y/o recibir descarga de instrucciones desde otros elementos de cálculo dentro de la unidad de procesador de gráficos de fin general. En una realización, el paradigma de cálculo de datos cercano puede ser particularmente útil para realizar o acelerar operaciones de matriz dispersa, que tienen intensidad aritmética baja.

La **Figura 18** es un diagrama de bloques que ilustra un sistema de cálculo de memoria híbrida 1800, de acuerdo con una realización. En una realización, el sistema de cálculo de memoria híbrida 1800 ilustra una implementación del módulo de memoria híbrida 1430 de la Figura 14, que incluye unidades de controlador de cálculo y de memoria 1432A-1432B y la memoria local de la GPGPU 1432A-1432B. En el sistema de cálculo de memoria híbrida 1800 ilustrado, el módulo de memoria híbrida 1430 incluye adicionalmente un procesador de control 1802 y un controlador de memoria primario 1805. El procesador de control 1802 y el controlador de memoria primario 1805 pueden trabajar juntos con un controlador de DMA 1803 para posibilitar una transferencia de memoria de DMA de datos a, desde y entre módulos de la memoria local de la GPGPU 1434A-1434B.

En una realización, el procesador de control 1802 recibe solicitudes para operaciones de cálculo de entrada 1801 para que se satisfagan por la lógica computacional dentro de una o más de las unidades de controlador de cálculo y de memoria 1432A-1432B. El procesador de control 1802 puede despachar a continuación las operaciones de cálculo a la unidad de controlador de cálculo y de memoria 1432A-1432B apropiada basándose en el conjunto de direcciones a las que se va a acceder por las operaciones de cálculo. Las operaciones de cálculo pueden recibirse en forma de un núcleo de cálculo de datos cercano. En una realización, las direcciones de memoria a las que accederá un núcleo de cálculo de datos cercano que van a ejecutarse en las unidades de controlador de cálculo y de memoria 1432A-1432B se traducen desde direcciones virtuales a una dirección física antes de que se reciba el núcleo en el módulo de memoria híbrida 1430, ya que las unidades de controlador de cálculo y de memoria 1432A-1432B se subdividen basándose en la dirección física, con diferentes unidades asociadas con diferentes rangos de direcciones. En una realización, cuando va a realizarse una operación de cálculo en un conjunto de direcciones físicas que se manejan por múltiples controladores de memoria, puede usarse el controlador de DMA 1803 para transferir los datos asociados con el rango de direcciones de los diferentes módulos de la memoria local de la GPGPU 1434A-1434B a un único módulo, almacenándose al menos una porción de los datos en una o más memorias caché 1806A-1806B dentro del controlador de memoria primario 1805. Las unidades de controlador de cálculo y de memoria 1432A-1432B pueden a continuación realizar las operaciones aritméticas requeridas a los datos almacenados en las memorias caché 1806A-1806B, que pueden a continuación desalojarse de vuelta a la memoria local de la GPGPU 1434A-1434B.

Para operaciones de acceso a memoria, el controlador de memoria primario 1805 puede recibir operaciones de memoria de entrada 1807, encaminar las operaciones de memoria a la unidad de controlador de cálculo y de memoria 1432A-1432B apropiada basándose en las direcciones físicas a las que va a accederse. Cuando se recibe una solicitud de un rango de direcciones que cruza un límite de dirección física que divide múltiples unidades de controlador de cálculo y de memoria 1432A-1432B, pueden despacharse y darse servicio en paralelo a múltiples solicitudes de memoria. Las unidades de controlador de cálculo y de memoria 1432A-1432B pueden intercambiar datos entre

módulos asociados de la memoria local de la GPGPU 1434A-1434B y un conjunto de memorias intermedias gestionadas por el controlador de DMA 1803. Por ejemplo, y en una realización, para operaciones de lectura y escritura, una operación de DMA puede estar configurada por el controlador de DMA 1803 para transmitir datos desde la memoria local de la GPGPU 1434A-1434B mediante una memoria intermedia de E/S 1804.

Aunque se ilustran interfaces separadas para operaciones de cálculo de entrada 1801 y operaciones de memoria de entrada 1807, en una realización, se proporciona una interfaz de memoria y cálculo unificada en la que los comandos de acceso a memoria son un subconjunto de operaciones de cálculo. Por ejemplo, puede recibirse una operación de carga o almacén por el procesador de control 1802. El comando de carga o almacén puede a continuación ejecutarse por el controlador de memoria primario. En tales realizaciones, pueden ejecutarse accesos a memoria complejos tales como operaciones de dispersión/recopilación directamente mediante el sistema de cálculo de memoria híbrida 1800.

En una realización, se implementa el módulo de memoria híbrida 1430 como un cubo de memoria híbrida en el que se apila la memoria local de la GPGPU 1434A-1434B en la parte superior de una capa lógica que incluye las unidades de controlador de cálculo y de memoria 1432A-1432B, el procesador de control 1802 y el controlador de memoria primario 1805. Sin embargo, las realizaciones no están limitadas a implementaciones de cubo de memoria híbrida, ya que el módulo de memoria híbrida 1430 puede implementarse mediante cualquier sistema de memoria que tenga uno o más controladores de memoria que puedan realizar operaciones aritméticas.

Los elementos de cálculo y procesamiento de las unidades de procesamiento de gráficos de fin general descritos en el presente documento pueden incluir diversos tipos de unidades aritmético-lógicas, que incluyen unidades de coma flotante y de lógica de números enteros. Una matriz grande de tales unidades de procesamiento puede incluirse en los bloques de cálculo 1424A-1424N de una GPGPU 1420 como en la Figura 14. Una matriz más pequeña de tales unidades de procesamiento puede incluirse en las unidades de controlador de cálculo y de memoria 1432A-1432B mostradas, por ejemplo, en la Figura 14 y la Figura 18.

El ancho de banda de la memoria entre los elementos de memoria y de cálculo permanece casi constante independientemente de la capacidad de la memoria debido a la limitación de recuento de patillas por chip. Tales limitaciones de ancho de banda pueden introducir un problema de escalabilidad para cargas de trabajo intensivas en memoria, tal como el entrenamiento de red neuronal. El problema del escalamiento puede exacerbarse particularmente cuando se entrenan redes neuronales dispersas. El entrenamiento neuronal disperso no es aritméticamente intenso, sino que puede estar limitado gravemente por el ancho de banda de memoria sin hardware especializado que está adaptado para la operación en redes neuronales dispersas. La unidad de procesamiento de gráficos de fin general proporcionada por las realizaciones descritas en el presente documento incluye una unidad de acelerador de cálculo disperso, tal como la unidad del acelerador de cálculo disperso 1423 de la Figura 14, que se describe adicionalmente en las Figuras 21-22 a continuación. El entrenamiento de redes neuronales dispersas puede realizarse también de manera eficiente usando recursos de cálculo de datos cercanos proporcionados por el módulo de memoria híbrida 1430.

Se muestra un ejemplo de pseudocódigo de gradiente conjugado disperso en la **Tabla 5.**

Tabla 5 - Pseudocódigo de gradiente conjugado disperso

```
0  for(....){
1     for(krow = 0; krow < 8; krow++ ){
2        for(kcol = 0; kcol < 8; kcol++ ){
3           a[node[krow] + node[kcol]*n] += coeff * em[krow+kcol*8];
4        }
5     }
6  }
```

El pseudocódigo ilustrado en la Tabla 5 realiza un gradiente de conjugado disperso que es aplicable a sistemas de matriz dispersa. El recuento de viajes para los dos bucles más internos (que iteran a través de "krow" y "kcol") es normalmente pequeño. Por consiguiente, cualquier paralelización o vectorización que puede realizarse sería más eficiente cuando se aplican a los bucles más externos. Las operaciones realizadas en la línea 3 provocan carga/almacenes indirectos (por ejemplo, recopilación/dispersión en el código de vector) y con poco cálculo presente en el bucle más interno, puede ser más eficiente descargar el cálculo realizado dentro de los dos bucles más internos a procesadores de cálculo de datos cercanos dentro del controlador de memoria. La descarga de las operaciones de cálculo de datos cercanos puede realizarse de una manera similar a descargar una operación de cálculo de un procesador de fin general (por ejemplo, la CPU) a una GPU. Sin embargo, en lugar de descargar las operaciones de cálculo a través de los dispositivos, la descarga de cálculo de datos cercanos descargará las operaciones de cálculo para calcular recursos dentro de un controlador de memoria, ya que los recursos de cálculo dentro del controlador de memoria tendrán un ancho de banda de comunicación significativamente superior a la memoria.

La **Tabla 6** a continuación ilustra un núcleo inspector y ejecutor que puede usarse para posibilitar la descarga de cálculo de datos cercanos.

Tabla 6 - Núcleos inspector y ejecutor

```
0  for(....){
1        //inspector fija páginas de memoria física, empaqueta las direcciones físicas
2     inspector_kernel(...); //realiza copia de datos desde el anfitrión al dispositivo
3     //se ejecuta el ejecutor por el controlador o controladores de memoria - nota, los datos
    pueden haberse
4     //subdividido a través de múltiples controladores de memoria por el núcleo del inspector
5     <<<executor_kernel(...)>>> //invocación del núcleo
6  }
```

Como se muestra en la Tabla 6, el inspector fija las páginas de memoria física y empaqueta las direcciones de memoria física. El núcleo ejecutor se invoca a continuación en el dispositivo. Las operaciones realizadas por el inspector son similares a una copia de datos desde el anfitrión a un dispositivo en el lenguaje de programación paralelo de alto nivel de CUDA. La invocación de núcleo de ejecutor puede compararse con una invocación de núcleo de CUDA.

La **Tabla 7** ilustra un núcleo de inspector ilustrativo.

Tabla 7 - núcleo de inspector

```
0    my_inspector_kernel(addr_a, addr_em, ...){
1       for(krow = 0; krow < 8; krow++ ){
2          for(kcol = 0; kcol < 8; kcol++ ){
3             //fija páginas y calcula/empaqueta direcciones físicas
4             get_pa(&a[node[krow]+node[kcol]*n], pa_addr_a, mem_ctrl_id_a);
5             addr_a[mem_ctrl_id].append(pa_addr_a);
6             get_pa(&em[krow+kcol*8], pa_addr_em, mem_ctrl_id_em);
7             //si se manejan las direcciones físicas mediante controladores de memoria diferentes
8             //DMA y proporciona todos los datos al controlador de memoria principal
9          if(mem_ctrl_id_a == mem_ctrl_id_em){
10            //en este punto puesto que mem_ctrl_a maneja almacenes y cargas, el resto
11            //de datos cargados se transfieren a su región de memoria
12            pa_addr_em = dma_pa(buff[mem_ctrl_id_a], pa_addr_em);
13         }
14         addr_em[mem_ctrl_id].append(pa_addr_em);
15         }
16      }
17   }
```

El núcleo de inspector determina las direcciones de memoria física relevantes para que se accedan por los cálculos mostrados en la Tabla 5. Estas direcciones, a continuación, se fijan y empaquetan en una estructura de datos. Si las direcciones físicas se manejan por controladores de memoria diferentes, puede realizarse un DMA para mover los datos a un controlador de memoria principal, que es el controlador de memoria que realizará los cálculos.

La **Tabla 8** ilustra un núcleo ejecutor ilustrativo.

Tabla 8 - Núcleo ejecutor ilustrativo

```
0  my_executor_kernel(my_addr_a, my_addr_em, coeff) {
1     //cada controlador de memoria procesa direcciones en sus cercanías
2     //my_addr_a es addr_a[mem_ctrl_id], etc.
3     for(i=0; i<len(my_addr_a);i++){
4      *(my_addr_a) += coeff * *(my_addr_em);
5     }
6  }
```

Como se muestra en la Tabla 8, la función del ejecutor acepta dos clases de argumentos: 1) aquellos argumentos que se calculan/preparan por el núcleo/función de inspector (por ejemplo, las direcciones) y 2) argumentos que se pasan desde el núcleo original y se usan por el inspector (por ejemplo, coeff). Los argumentos que se pasan desde el núcleo original se denominan valores vivos. El ejecutor recibe una lista de argumentos que incluyen valores vivos (por ejemplo, coeff) y direcciones (my_addr_a, my_addr_em) a la función de cálculo de memoria cercana. Las realizaciones descritas en el presente documento proporcionan soporte para una ISA de procesamiento heterogéneo que posibilita la descarga entre procesadores. El núcleo del ejecutor puede encapsularse dentro de una función para posibilitar la descarga del núcleo. Puede proporcionarse una llamada de descarga por la ISA de procesamiento heterogéneo que puede usarse para descargar el cálculo a los elementos de cálculo de memoria cercana desde los elementos de cálculo de fin general dentro de la GPGPU.

Las Figuras 19A-19B son diagramas de flujo que ilustran lógica para realizar operaciones de cálculo de datos cercanos mediante las realizaciones descritas en el presente documento. La **Figura 19A** ilustra la lógica 1900 para marcar cargas de trabajo que pueden realizarse de manera óptima mediante lógica de cálculo de datos cercanos. La **Figura 19B** ilustra la lógica 1910 para despachar una carga de trabajo de cálculo de datos cercanos a un controlador de memoria que tiene lógica de cálculo. En diversas realizaciones, la lógica 1900, 1910 ilustrada puede proporcionarse por unidades de software o hardware dentro de un sistema de procesamiento de datos descrito en el presente documento, tal como, pero sin limitación, el sistema de procesamiento de datos 1400 como en la Figura 14.

Como se muestra en la Figura 19A, una realización proporciona la lógica 1900 implementada mediante un sistema de procesamiento de datos que incluye lógica de compilación para compilar una carga de trabajo de la GPGPU para su ejecución, como se muestra en el bloque 1902. La lógica de compilación puede proporcionarse por un compilador y una o más bibliotecas de tiempo de compilación y/o de tiempo de ejecución, tal como las bibliotecas del compilador 1415 y de tiempo de ejecución 1416 como en la Figura 14. Durante o después de la compilación realizada en el bloque 1902, la lógica 1900 puede perfilar la carga de trabajo para determinar una complejidad de cálculo y de memoria de la carga de trabajo, como se muestra en el bloque 1904. Las cargas de trabajo que son más adecuadas para el cálculo de datos cercano son cargas de trabajo que tienen complejidad de acceso a memoria alta y complejidad aritmética o computacional baja. Por ejemplo, la carga de trabajo de gradiente conjugado disperso mostrada en la Tabla 5 realiza un número limitado de operaciones matemáticas (por ejemplo, multiplicación, adición), el patrón de acceso a memoria es complejo. Para cargas de trabajo que tienen complejidad de cálculo baja y complejidad de acceso a memoria alta, como se determinan en el bloque 1905, la lógica 1900 puede marcar aquellas cargas de trabajo para cálculo de datos cercanos, como se muestra en el bloque 1908. Para datos con complejidad de cálculo alta y/o complejidad de acceso a memoria baja, la lógica 1900 puede marcar la carga de trabajo para su ejecución en los recursos de cálculo principales de la GPGPU.

En una realización, marcar la carga de trabajo puede realizarse marcando información de sugerencias o metadatos asociada con la carga de trabajo. Por ejemplo, y en una realización, los datos compilados para calcular núcleos dentro de la carga de trabajo pueden tener sugerencias de procesador o metadatos que identifican el núcleo de cálculo como un núcleo de cálculo de datos cercanos. En una realización, los núcleos de cálculo de datos cercanos pueden planificarse directamente a lógica de cálculo dentro de las unidades de controlador de cálculo y de memoria 1432A-1432B descritas en el presente documento. En una realización, las cargas de trabajo se planifican a los bloques de cálculo 1424A-1424N descritos en el presente documento y pueden descargarse a los recursos de cálculo de datos cercanos en tiempo de ejecución.

Como se muestra en la Figura 19B, una realización proporciona la lógica 1910 implementada mediante un sistema de procesamiento de datos que incluye la lógica de ejecución de carga de trabajo para cargar una carga de cálculo de datos cercanos para su ejecución en la GPGPU, como se muestra en el bloque 1912. En una realización, la carga de trabajo es un núcleo de cálculo paralelo (por ejemplo, el núcleo ejecutor como en la Tabla 8) para el que se ejecutan múltiples instancias mediante lógica de procesamiento paralelo. La lógica 1910 puede inspeccionar el conjunto de direcciones de memoria que se accederán por la carga de trabajo en el bloque 1914. Cuando el hardware de la GPGPU y el modelo de programación posibilitan el uso de direcciones de memoria virtual, según se determina en el bloque 1915, la lógica 1910 puede traducir las direcciones virtuales a direcciones físicas, como se muestra en el bloque 1916. En una realización, por ejemplo, cuando los datos que van a accederse son dispersos, el conjunto de direcciones

físicas a las que se va a acceder puede empaquetarse en una estructura de datos. La lógica 1910 puede determinar también el controlador de memoria o controladores de memoria asociados con las direcciones físicas. La memoria a la que va a accederse puede abarcar múltiples regiones de memoria que se controlan por múltiples controladores de memoria. Si la memoria accedida se controla por múltiples controladores de memoria, según se determina en el bloque 1919, la lógica 1910 puede configurar una operación de DMA para transferir los datos a una región de memoria que se controla por un único controlador de memoria, como se muestra en el bloque 1920.

En diversas realizaciones, pueden posibilitarse diferentes enfoques para la consolidación de memoria para minimizar la cantidad de transferencia de datos o para minimizar la latencia asociada con cualquier transferencia de datos para que se realice antes del cálculo de memoria cercano. Por ejemplo, y en una realización, los datos se transfieren a la región de memoria asociada con el conjunto primario de cargas y almacenes que van a realizarse por la carga de trabajo, como se muestra en la Tabla 7. Cuando el conjunto de datos está situado de manera apropiada, la lógica 1900 puede despachar la carga de trabajo a un controlador de memoria. Las operaciones de lógica y los accesos a memoria para las cargas de trabajo pueden realizarse a continuación dentro del controlador de memoria. En una realización, en lugar de consolidar datos dentro de una región de memoria única como se muestra en la Figura 19B y el pseudocódigo enumerado anteriormente, en algunos casos, las operaciones de lógica pueden dividirse entre los controladores de memoria. Por ejemplo, ciertas cargas de trabajo perfectamente paralelas pueden subdividirse y ejecutarse simultáneamente en múltiples controladores de memoria.

La lógica de cálculo implementada dentro de las unidades de controlador de cálculo y de memoria 1432A-1432B descritas en el presente documento puede variar a través de las realizaciones. En una realización, las unidades de cálculo a nivel de arquitectura sencillas y de cálculo de baja potencia pueden incorporarse en cada controlador de memoria y la ISA de la GPGPU se extiende para posibilitar la planificación o descarga de un subconjunto específico de operaciones a controladores de memoria para cálculo de datos cercanos. En una realización, la lógica del controlador de memoria puede incluir las ALU y/o la lógica de FPU 2000 configuradas para realizar operaciones de multiplicación-adición fusionadas paralelas, como se muestra en la **Figura 20.**

La lógica de multiplicación-adición 2001 ilustrativa de la Figura 20 se describe en general con respecto a operaciones de coma flotante. Sin embargo, la lógica 2001 puede estar configurada para realizar de manera selectiva operaciones de coma de números enteros y fija. Las operaciones de multiplicación-adición pueden ejecutarse en múltiples elementos de datos en el mismo número de ciclos de reloj como una única multiplicación en datos desempaquetados. La lógica de multiplicación-adición acepta múltiples entradas que incluyen la Fuente1[63:0] 2031, Fuente2[63:0] 2033 y la activación 2080. El control de operación 2002 procesa unas señales de control de entrada para la lógica de multiplicación-adición 2001 y proporciona la activación 2080 de la entrada para activar la lógica de multiplicación-adición 2011. La lógica de multiplicación-adición 2001 incluye cuatro circuitos multiplicadores de 16×16 (por ejemplo, el multiplicador A de 16×16 2010A, el multiplicador B de 16×16 2010B, el multiplicador C de 16×16 2010C, el multiplicador D de 16×16 2010D). Los resultados intermedios de 32 bits generados por el multiplicador A de 16×16 2010A y el multiplicador B de 16×16 2010B se reciben por el sumador 2020A, mientras que los resultados intermedios de 32 bits generados por el multiplicador C de 16×16 2010C y el multiplicador D de 16×16 2010D se reciben por el sumador 2020B. La salida del sumador 2020B (es decir, los bits 31 a 0 del resultado) y la salida del sumador 2020A (es decir, los bits 63 a 32 del resultado) se combinan en el resultado de 64 bits y se comunican al registro de resultado 2030. En una realización, cada uno del sumador 2020A y el sumador 2020B está compuesto de cuatro sumadores de 8 bits con los retardos de propagación apropiados. Sin embargo, realizaciones alternativas podrían implementar el sumador 2020A-2020B en cualquier número de maneras (por ejemplo, dos sumadores de 32 bits y/o circuitería de compresión aritmética redundante).

Aceleración de cálculo disperso

Las operaciones de matriz dispersa se hallan comúnmente en muchos dominios de aplicación, incluyendo el aprendizaje automático. Por consiguiente, las optimizaciones al hardware para posibilitar un procesamiento más eficiente de las operaciones de matriz dispersa pueden ser de uso particular en el hardware de GPGPU que está optimizado para operaciones de aprendizaje automático. Los conjuntos de datos de matriz dispersa pueden tener una distribución sesgada de valores distintos de cero, donde una porción de la matriz es dispersa, con un número razonable de valores distintos de cero por columna o fila, mientras que otras porciones de la matriz son muy dispersas, con únicamente unos pocos valores distintos de cero por columna o fila, o híper dispersa, con todas las filas o columnas estando vacías. En una matriz híper dispersa, el número de valores distintos de cero puede ser menor que el número de filas y columnas en la matriz. Puede surgir una distribución sesgada a partir de grafos naturales que siguen la distribución de la ley de la potencia, con unos pocos nodos conocidos que tienen muchos bordes a otros nodos y muchos otros nodos que únicamente tienen unos pocos bordes. En conjuntos de datos de aprendizaje automático, las columnas y filas de la matriz representan características y muestras, respectivamente, ocurriendo algunas características de manera más frecuente que otras, dando como resultado valores distintos de cero sesgados a través de las columnas.

Las realizaciones descritas en el presente documento proporcionan una arquitectura de acelerador de hardware que puede mejorar la eficacia de procesamiento del hardware de la GPGPU cuando se procesan datos de matriz dispersa sesgados. En una realización, la arquitectura de acelerador de hardware se implementa dentro de la unidad del

acelerador de cálculo disperso 1423 de la Figura 14. Los elementos de la arquitectura de acelerador de hardware de cálculo disperso se ilustran en las Figuras 21-22.

La **Figura 21** ilustra una arquitectura de acelerador de cálculo disperso 2100, de acuerdo con una realización. En una realización, la arquitectura de acelerador de cálculo disperso 2100 está configurada para operar en un conjunto arbitrariamente grande de datos de entrada (por ejemplo, matriz, vector) que reside en memoria externa (por ejemplo, fuera del chip), tal como la memoria local de la GPGPU 1434A-1434B como en la Figura 14. En una realización, la arquitectura de acelerador de cálculo disperso 2100 puede operar también directamente en datos almacenados en memoria no volátil de alto ancho de banda, tal como 3D XPoint o Nano-RAM. La arquitectura de acelerador de cálculo disperso 2100 puede comunicarse independientemente con la memoria para leer datos de entrada y escribir de vuelta los resultados del cálculo sin requerir el uso de los recursos de cálculo primarios dentro de una GPGPU de anfitrión.

En una realización, la arquitectura de acelerador de cálculo disperso 2100 incluye la unidad del acelerador de cálculo disperso 1423 y una porción de la unidad del planificador de aprendizaje automático 1422 de la Figura 14. El controlador planificador de aprendizaje automático 1422 puede incluir una unidad de pre-extracción dispersa 2130 que está configurada para pre-extraer direcciones que contienen valores distintos de cero de una matriz dispersa. Pre-extraer y almacenar el número limitado de valores distintos de cero de la matriz dispersa dentro de las memorias caché y pre-extraer memorias intermedias de la arquitectura de acelerador de cálculo disperso 2100 puede activar fallos de página para cualquier dirección de memoria virtual que no está residente en memoria física. Pre-activar fallos de página reducirá la latencia de acceso para las direcciones pre-extraídas incluso si los datos asociados con estas direcciones no se almacenan en una memoria caché dentro de la arquitectura de acelerador de cálculo disperso 2100 cuando se accede a la dirección de memoria por un núcleo de cálculo que se ejecuta en la arquitectura.

La unidad de pre-extracción dispersa 2130, en una realización, se acopla con una unidad de gestión de datos 2120 dentro del controlador planificador de aprendizaje automático 1422. En una realización, la unidad de gestión de datos 2120 incluye una unidad de lectura y una unidad de escritura, incluyendo la unidad de lectura un planificador de elemento de procesamiento (PE) 2121, un comparador NxN 2122 y una memoria intermedia de lectura 2123. La unidad de escritura, en una realización, incluye una memoria intermedia de escritura 2124, aunque la unidad de escritura puede incluir otros componentes en diversas realizaciones dependiendo del caso de uso objetivo de la arquitectura de acelerador de cálculo disperso 2100. Adicionalmente, aunque se ilustra la unidad de gestión de datos 2120 como un componente del controlador planificador de aprendizaje automático 1422, no todas las realizaciones están limitadas a una configuración de este tipo, ya que la unidad de gestión de datos 2120 puede ser un módulo separado del planificador de aprendizaje automático 1422 y/o puede estar integrada en la lógica de hardware de la unidad del acelerador de cálculo disperso 1423.

La unidad del acelerador de cálculo disperso 1423, en una realización, incluye múltiples elementos de procesamiento (por ejemplo, el PE 2110A-2110N). En una realización, los elementos de procesamiento 2110A-2110N puede cada uno incluir lógica similar a las ALU y/o a la lógica de FPU 2000 de la Figura 20, y pueden estar configurados para procesar operandos de vector para operación de SIMD. En una realización, los elementos de procesamiento 2110A-2110N incluyen memoria intermedia de entrada y las unidades de desempaquetamiento 2111A-2111N, la memoria de acceso aleatorio 2112A-2112N y una memoria intermedia de salida 2113A-2113N. Las memorias intermedias dentro de los elementos de procesamiento 2110A-2110N pueden ser memorias intermedias de memoria de acceso aleatorio estática, mientras que la RAM 2112A-2112N puede ser cualquier memoria de acceso aleatorio descrita en el presente documento, que incluye RAM estática o dinámica. La memoria intermedia de entrada y la unidad de desempaquetamiento 2111A-2111N soportan un formato de matriz densa, formatos de matriz dispersa comprimidos, así como optimizaciones de formato de matriz dispersa adicionales, tales como compresión de valor único. Los elementos de procesamiento 2110A-2110N pueden incluir lógica de multiplicación-adición que incluye un multiplicador y un sumador como se describe en el presente documento, donde la lógica de multiplicación y de adición puede estar configurada para realizar operaciones de multiplicación-adición fusionadas o combinadas. La lógica de multiplicación y adición es configurable para aceptar la entrada de la RAM 2112A-2112N o desde memoria externa mediante la memoria intermedia de entrada y las unidades de desempaquetamiento 2111A-2111N. La salida puede escribirse en un registro de suma o en la RAM 2112A-2112N. Los datos almacenados en la RAM 2112A-2112N o en las memorias intermedias de salida 2113A-2113N pueden emitirse a una memoria intermedia de escritura 2124 dentro de la unidad de gestión de datos 2120.

Aunque existen soluciones de arquitectura de hardware para acelerar las operaciones de matriz y vector, tales soluciones de arquitectura no soportan operaciones de matriz y vector para algoritmos de aprendizaje automático que operan en conjuntos de datos dispersos (por ejemplo, texto), tal como una multiplicación contra un vector disperso, el soporte para formatos de datos tanto orientados a fila como orientados a columna y las operaciones de escalamiento y actualización. La arquitectura de acelerador de cálculo disperso 2100, descrita en el presente documento, soporta tanto formatos de datos orientados a fila como orientados a columna, así como otras operaciones de matriz y vector comúnmente usadas soportadas por los aceleradores existentes. Por ejemplo, una realización proporciona una arquitectura de acelerador de cálculo disperso 2100 configurada para realizar de manera eficiente operaciones que incluyen la operación de multiplicación (matriz, vector) en formatos orientados tanto a filas como orientados a columnas para cualquier combinación de una matriz dispersa o densa y un vector disperso o denso (por ejemplo, matriz dispersa, vector disperso; matriz dispersa, vector denso; matriz densa, vector disperso; matriz densa, vector denso. La

arquitectura de acelerador de cálculo disperso 2100 puede soportar adicionalmente operaciones de producto vectorial de vectores (por ejemplo, vector, vector) que incluyen vector disperso, vector disperso; vector disperso, vector denso; y vector denso, vector denso. La arquitectura de acelerador de cálculo disperso 2100 puede soportar adicionalmente unas operaciones de escalamiento y actualización (ScaleAndUpdate) que tienen operandos de matriz dispersa, vector denso. Se pretende que la arquitectura de acelerador de cálculo disperso 2100 opere, en general, en datos de matriz grandes, donde el rendimiento típicamente está limitado por el ancho de banda de memoria disponible para acceder a tales datos. Por consiguiente, la arquitectura del acelerador se ha diseñado para escalar y aprovechar la mejor ventaja de todo el ancho de banda de memoria disponible. En una realización, se maximiza el ancho de banda de memoria disponible implementando la arquitectura de acelerador de cálculo disperso 2100 como una arquitectura de cálculo de datos cercanos, tal como en el sistema de cálculo de memoria híbrido 1800 de la Figura 18.

Un desafío clave presentado para el desarrollo de un acelerador de vector de matriz dispersa es el desarrollo de la lógica para reducir la latencia asociada con accesos aleatorios y/o irregulares a un vector denso. Los accesos aleatorios y/o irregulares pueden conducir a problemas de rendimiento cuando el vector denso está en memoria. Por ejemplo, el acceso puede requerir que se realice una operación de recopilación o dispersión para leer o escribir los datos con patrones irregularmente a y desde la memoria. Para tratar tal problema, el acelerador descrito en el presente documento está configurado para operar en unos datos de matriz que están bloqueados de modo que el vector denso que corresponde a cada bloque de matriz se ajusta en la RAM de PE. Durante la operación, la unidad del acelerador de cálculo disperso 1423 puede enviar por flujo continuo datos de matriz distintos de cero en los elementos de procesamiento para su procesamiento contra los datos de vector almacenados en la RAM interna 2112A-2112N de cada elemento de procesamiento 2110A-2110N. Los accesos aleatorios para los datos de vector almacenados se realizan en la RAM local 2112A-2112N dentro de los elementos de procesamiento 2110A-2110N, lo que evita accesos irregulares a memoria durante la ejecución de carga de trabajo de cálculo.

La **Figura 22** ilustra una arquitectura de cálculo disperso adicional 2200 para operaciones matriciales dispersas, de acuerdo con una realización. Una realización proporciona una arquitectura heterogénea que posibilita el procesamiento eficiente de matrices sesgadas que contienen bloques de matriz dispersa, así como bloques de matriz muy dispersa y/o híper dispersa. Los datos de matriz de entrada 2202 almacenados en memoria se leen por un módulo de subdivisión de matriz 2210, que emite un conjunto de bloques dispersos 2220 y, si está presente, un conjunto de bloques muy densos o híper densos 2222. Los bloques dispersos 2220 se almacenan en memoria 2230 que está optimizada para ancho de banda en bruto, mientras que los bloques muy dispersos o híper dispersos 2222 se almacenan en memoria 2232 que está optimizada para posibilitar baja latencia para ráfagas breves de accesos paralelos. Los diversos tipos de memoria 2230, 2232, se acoplan mediante una interconexión 2233 para calcular recursos de la arquitectura de cálculo disperso 2200.

En algunas realizaciones, los recursos informáticos de la arquitectura de cálculo disperso 2200 incluyen una pieza de cálculo disperso 2234 y una pieza de cálculo muy/híper disperso 2236. Un conjunto de planificadores 2235 se configuran para planificar tareas para su ejecución en las piezas de cálculo disperso 2234 y piezas de cálculo muy o híper dispersas 2236. La pieza de cálculo disperso 2234 puede incluir elementos ilustrados en la unidad del acelerador de cálculo disperso 1423 de la Figura 14 y la Figura 21, exceptuando que, en una realización, la unidad de gestión de datos (DMU) está integrada dentro de la pieza de cálculo disperso 2234 en lugar del controlador planificador de aprendizaje automático 1422. Los datos distintos de cero en los bloques dispersos 2220 almacenados en memoria 2230 pueden enviarse por flujo continuo en la RAM en chip de la pieza de cálculo disperso 2234. Las técnicas que posibilitan la pieza de cálculo disperso 2234 para procesar eficientemente datos dispersos son menos eficientes en matrices dispersas muy dispersas e híper dispersas. Una matriz muy/híper dispersa tiene muy pocos números distintos de cero. Por consiguiente, procesar tales matrices incurre en una sobrecarga de bloqueo relativamente superior (por ejemplo, punteros de fila o columna). La sobrecarga de bloqueo superior significa que se consumió más tiempo de cálculo y ancho de banda de memoria en el procesamiento de datos de la contabilidad en relación con el procesamiento de los elementos de matriz distintos de cero reales. Adicionalmente, las matrices muy/híper dispersas tienen muy pocos números distintos de cero por columna y fila, y acceder a las columnas y filas implica accesos a memoria más pequeños y más breves. Adicionalmente, se reutiliza una cantidad menor de los datos accedidos durante el procesamiento. La pieza de cálculo muy/híper disperso 2236 supera estas ineficacias mediante ajustes a la arquitectura de la pieza de cálculo disperso 2234.

Para aumentar la eficacia de la pieza de cálculo muy/híper disperso 2236, el módulo de subdivisión de matriz 2210 genera bloques más grandes de la matriz dispersa. Los bloques más grandes dan como resultado una sobrecarga de bloqueo reducida con relación a los datos distintos de cero que van a procesarse. El bloque de matriz más grande tiene un subconjunto de vectores asociado más grande que se procesará contra los bloques muy/híper dispersos 2222. En lugar de almacenar el subconjunto de vectores en la RAM en chip, como en la pieza de cálculo disperso 2234, el subconjunto de vectores se almacena en la memoria optimizada paralela 2232. La pieza de cálculo muy/híper disperso 2236 usa una unidad de gestión de datos optimizada para operaciones de recopilación/dispersión (por ejemplo, la DMU de G/S 2237). En una realización, la DMU de G/S 2237 incluye una caché 2238 para capturar la modesta reutilización de datos para los datos del subconjunto de vectores. En algunas realizaciones, la pieza de cálculo muy/híper disperso 2236 puede incluir también menos elementos de procesamiento con relación a la pieza de cálculo disperso 2234. En algunas realizaciones, cualquiera o ambas de la pieza de cálculo disperso 2234 y/o la pieza de cálculo muy/híper disperso 2236 pueden estar integradas en las unidades de controlador de cálculo o de memoria

1432A-1432B del módulo de memoria híbrida 1430 descritas en el presente documento para optimizar la capacidad de cálculo disperso de los módulos de cálculo de datos cercanos.

En una realización, el módulo de subdivisión de matriz 2210 incluye una unidad de análisis de propiedad de matriz 2211, una unidad de determinación de subdivisión de bloque 2212 y una unidad de optimización de matriz 2213. La unidad de análisis de propiedad de matriz 2211 está configurada para analizar diversas propiedades de la matriz, tal como el número de números distintos de cero por columnas o filas. Se proporcionan las métricas determinadas por la unidad de análisis de propiedad de matriz 2211 a la unidad de determinación de subdivisión de bloque, que determina la técnica apropiada para usar para subdividir la matriz en bloques. La unidad de determinación de subdivisión de bloque, a continuación, determina límites de bloque de matriz, de manera que se colocan partes de la matriz con propiedades similares dentro del mismo bloque. La unidad de optimización de matriz 2213 a continuación aplica diversas optimizaciones para mejorar la eficacia de procesamiento de las unidades de cálculo cuando se procesan los bloques. Por ejemplo, y en una realización, la unidad de optimización de matriz 2213 puede optimizar el formato de matriz para cada bloque, de manera que el bloque híper disperso usa un formato doblemente comprimido, mientras que un bloque de matriz delgado y alto usa un formato orientado a filas para evitar la dispersión de la memoria. La unidad de optimización de matriz 2213 puede optimizar también la planificación de los bloques para su procesamiento por los planificadores 2235 produciendo sugerencias de planificación para su uso cuando se planifican cargas de trabajo a los elementos de procesamiento.

Las Figuras 23A-23B son diagramas de flujo que ilustran la lógica 2300, 2310 para realizar operaciones de cálculo disperso dentro de una GPGPU proporcionada por las realizaciones descritas en el presente documento. La lógica 2300 puede implementarse por una unidad de acelerador de cálculo disperso 1423 como en la Figura 14 y en la Figura 21. En una realización, la unidad del acelerador de cálculo disperso 1423 incluye aspectos de la arquitectura de cálculo disperso 2200 de la Figura 22. La lógica 2310 puede implementarse mediante hardware dentro de la unidad de extracción y decodificación de instrucciones de aprendizaje automático 1421 y el controlador planificador de aprendizaje automático 1422 como en la Figura 14-Figura 15. En una realización, al menos una porción de la lógica 2310 puede implementarse dentro de elementos de cálculo de un módulo de memoria híbrida, tal como el módulo de memoria híbrida 1430 de la Figura 14 y/o la Figura 18.

Como se muestra en la Figura 23A, la lógica 2300 provoca que el hardware dentro de la GPGPU lea una matriz de entrada en una arquitectura de cálculo disperso, como se muestra en el bloque 2302. La lógica 2300 puede procesar a continuación la matriz mediante un módulo de subdivisión, como se muestra en el bloque 2304. El módulo de subdivisión puede usar una instancia del módulo de subdivisión de matriz 2210 de la Figura 22, y puede realizar operaciones que incluyen el análisis de matriz, la determinación de partición y la optimización de la matriz. Procesar la matriz en el bloque 2304 proporciona a la lógica 2300 con información para determinar si la matriz es una matriz muy dispersa o híper dispersa en el bloque 2305, donde una muy dispersa tiene pocos valores de datos distintos de cero por columna o fila y una matriz híper dispersa tiene filas o columnas enteras de valores de datos de cero. Como se muestra en el bloque 2306, cuando la matriz de entrada es únicamente dispersa y no muy o híper dispersa, la lógica 2300 puede emitir un conjunto de bloques de matriz dispersa a la memoria de ancho de banda optimizado, tal como la memoria 2230 de la Figura 22. Como se muestra en el bloque 2308, la lógica 2300 puede procesar a continuación el bloque de matriz dispersa mediante un fichero de cálculo de matriz dispersa, tal como la pieza de cálculo disperso 2234 de la Figura 22. Como se muestra en el bloque 2307, cuando la matriz de entrada es muy dispersa o híper dispersa, la lógica 2300 puede emitir un conjunto de bloques de matriz muy dispersa de híper dispersa a memoria de latencia optimizada, tal como la memoria 2232 de la Figura 22. Como se muestra en el bloque 2309, la lógica 2300 puede procesar a continuación el bloque de matriz muy o híper dispersa mediante una pieza de cálculo de matriz muy/híper dispersa, tal como la pieza de cálculo muy/híper disperso 2236 de la Figura 22

Como se muestra en la Figura 23B, la lógica 2310 posibilita una arquitectura de cálculo disperso descrita en el presente documento para que se integre en una microarquitectura optimizada para aprendizaje automático dentro de una GPGPU. La lógica 2310 puede determinar un conjunto de comandos de tubería para realizar, en respuesta a una instrucción de aprendizaje automático decodificada en una GPGPU, como se muestra en el bloque 2312. La instrucción de aprendizaje automático decodificada puede ser la instrucción de aprendizaje automático decodificada del bloque 1702 en la Figura 17. Como se muestra en el bloque 2314, en una realización, la lógica 2310 puede procesar el conjunto de comandos de tubería mediante lógica programable dentro de un planificador basado en hardware, tal como el controlador planificador de aprendizaje automático 1422 descrito en el presente documento. La lógica 2310 puede determinar a continuación, en el bloque 2315, si los comandos de tubería especifican alguna operación de matriz dispersa. Si han de realizarse operaciones de matriz dispersa, la lógica 2310 puede planificar las operaciones de matriz dispersa en un acelerador de matriz dispersa dentro de la GPGPU, como se muestra en el bloque 2317, donde el acelerador de matriz dispersa es un acelerador de cálculo disperso, tal como la unidad del acelerador de cálculo disperso 1423 como en la Figura 14 y la Figura 21 o la pieza de cálculo disperso 2234 y/o pieza de cálculo muy/híper disperso 2236 de la Figura 22. Si no se especifican las operaciones dispersas, la lógica 2310 puede planificar operaciones de comando a bloques de cálculo de fin general dentro de la GPGPU, como se muestra en el bloque 2316. En algunas realizaciones, puede realizarse también la matriz dispersa y un subconjunto de las operaciones de fin general mediante elementos de cálculo de datos cercanos, donde las operaciones de cálculo son sensibles a ancho de banda de memoria.

Sistema de procesamiento de gráficos ilustrativo adicional

Los detalles de las realizaciones anteriormente descritas pueden incorporarse dentro de los sistemas de procesamiento de gráficos y los dispositivos descritos a continuación. El sistema y los dispositivos de procesamiento de gráficos de la Figura 24 a la Figura 37 ilustran sistemas alternativos y hardware de procesamiento de gráficos que pueden implementar cualquiera y todas las técnicas anteriormente descritas.

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, portátiles o integrados.

Una realización del sistema 2400 puede incluir, o estar incorporada dentro de una plataforma de juegos basada en servidor, una consola de juegos, que incluye una consola de juegos y medios, una consola de juegos móvil, una consola de juegos portátil o una consola de juegos en línea. En algunas realizaciones, el sistema 2400 es un teléfono móvil, teléfono inteligente, dispositivo informático de tableta o dispositivo de internet móvil. El sistema de procesamiento de datos 2400 puede incluir también, estar acoplado con, o estar integrado dentro de un dispositivo llevable, tal como un dispositivo llevable de reloj inteligente, dispositivo de gafas inteligentes, dispositivo de realidad aumentada o dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 2400 es un dispositivo de televisión o decodificador 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 del uno o más procesadores 2402 incluye uno o más núcleos de procesador 2407 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para el sistema y el software de usuario. En algunas realizaciones, cada uno del uno o más núcleos de procesador 2407 está configurado para procesar un conjunto de instrucciones específico 2409. En algunas realizaciones, el conjunto de instrucciones 2409 puede facilitar el cálculo de conjunto de instrucciones complejo (CISC), el cálculo de conjunto de instrucciones reducido (RISC) o el cálculo mediante una palabra de instrucción muy larga (VLIW). Cada uno de múltiples núcleos de procesador 2407 puede procesar un conjunto de instrucciones diferente 2409, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 2407 puede incluir también otros dispositivos de procesamiento, tal como un procesador de señales digitales (DSP).

En algunas realizaciones, el procesador 2402 incluye 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 caché de último nivel (LLC)) (no mostrada), que puede compartirse entre núcleos de procesador 2407 usando técnicas de coherencia de caché conocidas. Un fichero de registro 2406 está incluido adicionalmente en el procesador 2402 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 fin general, mientras que otros registros pueden ser específicos al 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 direcciones, 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, que incluye un concentrador de controlador de memoria 2416 y un concentrador de controlador de entrada y 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 los dispositivos de E/S mediante 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), dispositivo de memoria flash, dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para servir como una memoria de proceso. En una realización, el dispositivo de memoria 2420 puede operar como memoria de sistema para el sistema 2400, para almacenar datos 2422 e instrucciones 2421 para su uso cuando el 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 opcional 2412, que puede comunicarse con el 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 posibilita que los periféricos se conecten al dispositivo de memoria 2420 y al procesador 2402 mediante 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 las combinaciones de teclado y ratón 2444. Un controlador de red 2434 puede acoplarse también con el ICH 2430. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de procesador 2410. Se apreciará que, el sistema 2400 mostrado es ilustrativo y no limitante, ya que pueden usarse otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por ejemplo, el concentrador de controlador de E/S 2430 puede integrarse dentro del uno o más procesadores 2402, o el concentrador de controlador de memoria 2416 y el concentrador de controlador de E/S 2430 pueden estar integrados 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 (o nombres) de referencia 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 sin limitación a esto. 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 de 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) u otros niveles, donde el nivel más alto de caché antes de la memoria externa se clasifica como la 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 puede incluir también un conjunto de una o más unidades de controlador de bus 2516 y un núcleo de agente de sistema 2510. La una o más unidades de controlador de bus 2516 gestionan un conjunto de buses periféricos, tal como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 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 externa (no mostrados).

En algunas realizaciones, uno o más de los núcleos de procesador 2502A-2502N incluyen el soporte para múltiples hilos simultáneos. En una realización de este tipo, el núcleo de agente de sistema 2510 incluye componentes para coordinar y operar los núcleos 2502A-2502N durante el 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 el procesador de gráficos 2508 para ejecutar las 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, que incluye el 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 la salida del 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 mediante al menos una interconexión, o puede estar integrado 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, puede usarse una unidad de interconexión alternativa, tal como una interconexión de punto a punto, una interconexión conmutada u otras técnicas, que incluyen técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador de gráficos 2508 se acopla con el anillo de interconexión 2512 mediante un enlace de E/S 2513.

El enlace de E/S 2513 ilustrativo representa al menos una de múltiples variedades de interconexiones de E/S, que incluyen una interconexión de E/S de paquete, que facilita la comunicación entre diversos componentes de procesador y un módulo de memoria integrado de alto rendimiento 2518, tal como un módulo de eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 2502A-2502N y del 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), 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, donde uno o más núcleos que tienen un consumo de potencia relativamente más alto se acoplan con uno o más núcleos de potencia que tienen un consumo de potencia más bajo. Adicionalmente, el procesador 2500 puede implementarse en uno o más chips o como un circuito de SoC integrado que tiene los componentes ilustrados, además de otros componentes.

La **Figura 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 mediante una interfaz de E/S de memoria mapeada a 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, a una o más cachés internas, a 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 la composición de múltiples capas de vídeo o elementos de interfaz de usuario. En algunas realizaciones, el procesador de gráficos 2600 incluye un motor de códec de vídeo 2606 para codificar, decodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, que incluyen, pero sin limitación formatos del Grupo de Expertos de Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como de la Sociedad de Ingenieros de Imágenes en Movimiento y Televisión (SMPTE) 421M/VC-1 y formatos del Grupo Mixto de Expertos en Fotografía (JPEG), tal como los formatos JPEG y Motion JPEG (MJPEG).

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

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

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

En algunas realizaciones, el subsistema 3D/de medios 2615 incluye lógica para ejecutar hilos generados por la tubería en 3D 2612 y la tubería de medios 2616. En una realización, las tuberías envían solicitudes de ejecución de hilos al subsistema 3D/de medios 2615, que incluye lógica de despacho de hilo para arbitrar y despachar las diversas solicitudes a recursos de ejecución de hilo disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos 3D y los medios. En algunas realizaciones, el subsistema 3D/de medios 2615 incluye una o más cachés internas para instrucciones y datos de hilo. En algunas realizaciones, el subsistema también incluye memoria compartida, que incluye 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 (o nombres) de referencia 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 sin limitación a esto. Por ejemplo, se ilustra la tubería 3D 2612 y la tubería de medios 2616 de la **Figura 26**. La tubería 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 está acoplado al GPE 2710.

En algunas realizaciones, el GPE 2710 se acopla con o incluye un transmisor de envío por flujo continuo de comandos 2703, que proporciona un flujo de comandos a la tubería 3D 2612 y/o a las tuberías 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 de caché interna y memoria de caché compartida. En algunas realizaciones, el transmisor por flujo continuo de comandos 2703 recibe comandos desde la memoria y envía los comandos a la tubería 3D 2612 y/o a la tubería de medios 2616. Los comandos son directivas extraídas desde una memoria intermedia de anillo, que almacena comandos para la tubería 3D 2612 y la tubería de medios 2616. En una realización, la memoria intermedia de anillo puede incluir adicionalmente unas memorias intermedias de comandos en lotes que almacenan lotes de múltiples comandos. Los comandos para la tubería 3D 2612 pueden incluir también referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértices y geometría para la tubería 3D 2612 y/o datos de imagen y objetos de memoria para la tubería de medios 2616. La tubería 3D 2612 y la tubería de medios 2616 procesan los comandos y los datos realizando operaciones mediante la lógica dentro de las respectivas tuberías o despachando uno o más hilos de ejecución a una matriz de núcleo de gráficos 2714.

En diversas realizaciones, la tubería 3D 2612 puede ejecutar uno o más programas sombreadores, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculos u otros programas sombreadores, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleo de gráficos 2714. La matriz de núcleo de gráficos 2714 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución de múltiples fines (por ejemplo, las unidades de ejecución) dentro de la matriz de núcleo de gráficos 2714 incluye el soporte para diversos idiomas sombreadores 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 fin general que es programable para realizar operaciones de cálculo de fin general paralelas, además de operaciones de procesamiento de gráficos. La lógica de fin general puede realizar operaciones de procesamiento en paralelo o en conjunto con la lógica de fin general dentro del núcleo o núcleos de procesador 2407 de la **Figura 24** o el 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 puede usarse para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleo de gráficos 2714. En algunas realizaciones, la URB 2718 puede usarse adicionalmente para la sincronización entre hilos en la matriz de núcleo de gráficos y la lógica de función fija dentro de la lógica de función compartida 2720.

En algunas realizaciones, la matriz de núcleo de gráficos 2714 es escalable, de manera que la matriz incluye un número variable de núcleos de gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y el nivel de rendimiento del GPE 2710. En una realización, los recursos de ejecución son dinámicamente escalables, de manera que pueden activarse o desactivarse los recursos de ejecución según sean necesarios.

La matriz de núcleo de gráficos 2714 se acopla con la lógica de función compartida 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 función compartida 2720 son unidades de lógica de hardware que proporcionan funcionalidad complementaria especializada a la matriz de núcleo de gráficos 2714. En diversas realizaciones, lógica de función compartida 2720 incluye, pero sin limitación, el muestreador 2721, el cálculo 2722 y la lógica de comunicación inter-hilo (ITC) 2723. Adicionalmente, algunas realizaciones implementan una o más caché o cachés 2725 dentro de la lógica de función compartida 2720. Se implementa una función compartida donde la demanda para una función especializada dada es insuficiente para la inclusión dentro de la matriz de núcleo de gráficos 2714. En su lugar, se implementa una única instanciación de esa función especializada como una entidad autónoma en la lógica de función compartida 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 están incluidas 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 (o nombres) de referencia 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 sin limitación a esto.

En algunas realizaciones, el procesador de gráficos 2800 incluye una interconexión de anillo 2802, un extremo frontal de tubería 2804, un motor de medios 2837 y núcleos de gráficos 2880A-2880N. En algunas realizaciones, la interconexión de anillo 2802 acopla el procesador de gráficos a otras unidades de procesamiento, que incluyen otros procesadores de gráficos o uno o más núcleos de procesadores de fin 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 mediante la interconexión de anillo 2802. Los comandos de entrada se interpretan por un transmisor de envío por flujo continuo de comandos 2803 en el extremo frontal de la tubería 2804. En algunas realizaciones, el procesador de gráficos 2800 incluye una lógica de ejecución escalable para realizar procesamiento de geometría 3D y procesamiento de medios mediante el núcleo o núcleos de gráficos 2880A-2880N. Para comandos de procesamiento de geometría 3D, el transmisor por flujo continuo de comandos 2803 suministra comandos a la tubería 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 post procesamiento de vídeo y de imagen y un motor de codificación/decodificación de múltiples formatos (MFX) 2833 para proporcionar codificación y decodificación de datos de medios acelerados por hardware. En algunas realizaciones, cada uno de la tubería de geometría 2836 y el motor de medios 2837 generan 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 escalables que presentan núcleos modulares 2880A-2880N (en ocasiones denominados cortes de núcleo), teniendo cada uno múltiples subnúcleos 2850A-550N, 2860A-2860N (en ocasiones denominados 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 de caché compartida y lógica de operación de píxel. Pueden incluirse también otros recursos compartidos en las diversas realizaciones del procesador de gráficos.

Unidades de ejecución ilustrativas adicionales

La **Figura 29** ilustra lógica de ejecución de hilo 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 (o nombres) de referencia 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 sin limitación a esto.

En algunas realizaciones, la lógica de ejecución de hilo 2900 incluye un procesador sombreador 2902, un despachador de hilo 2904, una caché de instrucciones 2906, una matriz de unidad de ejecución escalable 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 unidad de ejecución escalable puede escalar dinámicamente activando o desactivando una o más unidades de ejecución (por ejemplo, cualquiera de la unidad de ejecución 2908A, 2908B, 2908C, 2908D, a 2908N-1 y 2908N) basándose en los requisitos de cálculo de una carga de trabajo. En una realización, los componentes incluidos están interconectados mediante un tejido de interconexión que se enlaza a cada uno de los componentes. En algunas realizaciones, la lógica de ejecución de hilo 2900 incluye una o más conexiones a memoria, tal como la memoria de sistema o memoria caché, a través de una o más de la caché de instrucciones 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 de cálculo de fin general programable autónoma que puede 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 escalable para incluir cualquier número de unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 2908A-2908N se usan principalmente para ejecutar programas sombreadores. Un procesador sombreador 2902 puede procesar los diversos programas sombreadores y despachar los hilos de ejecución asociados con los programas sombreadores mediante un despachador de hilo 2904. En una realización, el despachador de hilo incluye lógica para arbitrar las solicitudes de iniciación de hilo desde las tuberías 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 tubería de geometría (por ejemplo, 2836 de la **Figura 28**) puede despachar los sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilo 2900 (**Figura 29**) para su

procesamiento. En algunas realizaciones, el despachador de hilo 2904 puede procesar también hilos en tiempo de ejecución que abarcan solicitudes desde los programas sombreadores de ejecución.

En algunas realizaciones, las unidades de ejecución 2908A-2908N soportan un conjunto de instrucciones que incluye el soporte nativo para muchas instrucciones del sombreador de gráficos de 3D convencional, de manera que se ejecutan los programas sombreadores de las bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) con una traducción mínima. Las unidades de ejecución soportan procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y procesamiento de fin general (por ejemplo, sombreadores de cálculo y de medios). Cada una de las unidades de ejecución 2908A-2908N puede emitir de manera múltiple la ejecución de datos de múltiples instrucciones sencillas (SIMD) y la operación de múltiples hilos posibilita un entorno de ejecución eficaz frente a los accesos a memoria de latencia más alta. 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 tuberías aptas para operaciones de números enteros, y de coma flotante de precisión sencilla y doble, capacidad de ramal de SIMD, operaciones lógicas, operaciones transcendentales y otras operaciones misceláneas. Mientras se esperan los datos de la memoria o una de las funciones compartidas, la lógica de dependencia dentro de las unidades de ejecución 2908A-2908N hace que un hilo en espera pase a inactividad hasta que se devuelvan los datos solicitados. Mientras el hilo en espera estaba en inactividad, los recursos de hardware pueden dedicarse 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, sombreador de fragmentos u otro tipo de programa sombreador, que incluye un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 2908A-2908N opera en matrices de elementos de datos. El número de elementos de datos es el "tamaño de ejecución", o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso de elemento de datos, el enmascaramiento y el control de flujo dentro de las instrucciones. El número de canales puede ser independiente del número de Unidades 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 la unidad de ejecución incluye instrucciones SIMD. Los diversos elementos de datos pueden almacenarse como un tipo de datos empaquetado en un registro y la unidad de ejecución procesará los diversos elementos basándose en el tamaño de datos de los elementos. Por ejemplo, cuando se opera en un vector de 256 bits de ancho, los 256 bits del vector se almacenan en un registro y la unidad de ejecución opera en el vector como cuatro elementos de datos empaquetados de 64 bits separados (elementos de datos de tamaño de palabra cuádruple (QW)), ocho elementos de datos empaquetados de 32 bits separados (elementos de datos de tamaño de palabra doble (DW)), dieciséis elementos de datos empaquetados de 16 bits separados (elementos de datos de 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 tamaños de anchuras de vector y registros.

Una o más cachés de instrucción internas (por ejemplo, 2906) están incluidas en la lógica de ejecución de hilo 2900 a las instrucciones de hilo de caché para las unidades de ejecución. En algunas realizaciones, una o más cachés de datos (por ejemplo, 2912) están incluidas en datos de hilo de caché 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 funcionalidad de textura especializada o muestreo de medios para procesar los datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las tuberías de gráficos y de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilo 2900 mediante lógica de generación y despacho de hilo. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos en datos de píxeles, se invoca la lógica de procesador de píxel (por ejemplo, lógica de sombreador de píxeles, lógica de sombreador de fragmentos, etc.) dentro del procesador de sombreador 2902 para calcular adicionalmente información de salida y hace que se escriban los 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 sombreador de fragmentos calcula los valores de los diversos atributos de vértice que han de interpolarse a través del objeto rasterizado. En algunas realizaciones, la lógica de procesador de píxel dentro del procesador de sombreador 2902 ejecuta a continuación un píxel suministrado por la interfaz de programación de aplicación (API) o programa sombreador de fragmentos. Para ejecutar el programa sombreador, el procesador de sombreador 2902 despacha hilos a una unidad de ejecución (por ejemplo, 2908A) mediante el despachador de hilo 2904. En algunas realizaciones, el sombreador de píxeles 2902 usa la lógica de muestreo de textura en el muestreador 2910 para acceder a datos de textura en mapas de textura almacenados en memoria. Las operaciones aritméticas en los datos de textura y los datos de geometría de entrada calculan datos de color de píxel para cada fragmento geométrico, o descarta uno o más píxeles de su procesamiento adicional.

En algunas realizaciones, el puerto de datos 2914 proporciona un mecanismo de acceso a memoria para que la lógica de ejecución de hilo 2900 emita datos procesados a memoria para su procesamiento en una tubería de salida de procesador de gráficos. En algunas realizaciones, el puerto de datos 2914 incluye o se acopla a una o más memorias

de caché (por ejemplo, la caché de datos 2912) para almacenar en caché datos para el acceso a memoria mediante el 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 tienen instrucciones en múltiples formatos. Los recuadros con línea continua ilustran los componentes que se incluyen en general en una unidad de instrucción de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que únicamente están incluidos en un subconjunto de instrucciones. En algunas realizaciones, el formato de instrucción 3000 descrito e ilustrado son macroinstrucciones, en que son instrucciones suministradas a la unidad de ejecución, a diferencia de las microoperaciones resultantes de la decodificación de la instrucción una vez que se procesa la instrucción.

En algunas realizaciones, las unidades de ejecución de procesador de gráficos soportan de manera nativa las instrucciones en un formato de instrucción de 128 bits 3010. Un formato de instrucción de 64 bits compacto 3030 está disponible para algunas instrucciones basándose en la instrucción seleccionada, las opciones de instrucción y el número de operandos. 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 por realización. En algunas realizaciones, la instrucción está compactada en parte usando un conjunto de valores de índice en un campo de índice 3013. El hardware de la unidad de ejecución hace referencia a un conjunto de tablas de compactación basándose en el valor de índices 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, la operación de código de instrucción 3012 define la operación que ha de realizar la unidad de ejecución. Las unidades de ejecución ejecutan cada instrucción en paralelo a través de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de adición la unidad de ejecución realiza una operación de adición simultánea a través de cada canal de color que representa un elemento de textura o elemento de imagen. Por defecto, la unidad de ejecución realiza cada instrucción a través de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 3014 posibilita el control a través de ciertas opciones de ejecución, tal como la selección de canales (por ejemplo, predicación) y orden de canal de datos (por ejemplo, mezcla). Para las 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 la unidad de ejecución tienen hasta tres operandos que incluyen dos operandos de origen, src0 3020, src1 3022, y un destino 3018. En algunas realizaciones, las unidades de ejecución soportan instrucciones de destino dual, donde está implicado uno de los destinos. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 3024), donde la operación de código de instrucción 3012 determina el número de operandos de origen. Un último operando de origen de la instrucción puede ser un valor inmediato (por ejemplo, precodificado) 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 se proporciona directamente por los bits en la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 3010 incluye un campo de modo de acceso/dirección 3026, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En una realización, se usa el modo de acceso 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, 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 en bytes para operandos de origen y destino y, cuando está en un segundo modo, la instrucción puede usar direccionamiento alineado de 16 bytes para todos los operandos de origen y destino.

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

En algunas realizaciones, las instrucciones se agrupan basándose en los campos de bits del código de operación 3012 para simplificar la decodificación 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 lógica 3042 incluye instrucciones de movimiento y lógica de datos (por ejemplo, mover (mov),

comparar (cmp)). En algunas realizaciones, el grupo de movimiento y lógica 3042 comparte los cinco bits más significativos (MSB), donde las instrucciones mover (mov) son en forma de OOOOxxxxb y las instrucciones de lógica son en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 3044 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en forma de OOlOxxxxb (por ejemplo, 0x20). Un grupo de instrucciones de miscelánea 3046 incluye una mezcla de instrucciones, que incluyen instrucciones de sincronización (por ejemplo, espera, envío) en forma de 001 1xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo paralelo 3048 incluye instrucciones aritméticas a nivel de componente (por ejemplo, añadir, multiplicar (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculos paralelos 3048 realiza las operaciones aritméticas en paralelo a través de canales de datos. El grupo de cálculos vectoriales 3050 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de OlOlxxxxb (por ejemplo, 0x50). El grupo de cálculos vectoriales realiza la aritmética tal como los cálculos de producto vectorial en operandos vectoriales.

Tubería 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 (o nombres) de referencia 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 sin limitación a esto.

En algunas realizaciones, el procesador de gráficos 3100 incluye una tubería de gráficos 3120, una tubería de medios 3130, un motor de visualización 3140, lógica de ejecución de hilo 3150 y una tubería de salida del representador 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 fin general. El procesador de gráficos se controla por las escrituras de registro en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 3100 mediante una interconexión de anillo 3102. En algunas realizaciones, la interconexión de anillo 3102 acopla el procesador de gráficos 3100 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de fin general. Los comandos desde la interconexión de anillo 3102 se interpretan por un transmisor de envío por flujo continuo de comandos 3103, que suministra instrucciones a componentes individuales de la tubería de gráficos 3120 o la tubería de medios 3130.

En algunas realizaciones, el transmisor de envío por flujo continuo 3103 dirige la operación de un extractor de vértice 3105 que lee los datos de vértices de memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor de envío por flujo continuo 3103. En algunas realizaciones, el extractor de vértices 3105 proporciona datos de vértices a un sombreador de vértices 3107, que realiza operaciones de transformación espacial de coordenadas y de iluminación 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 unidades de ejecución 3152A-3152B mediante un despachador de hilo 3131.

En algunas realizaciones, las unidades de ejecución 3152A-3152B son una matriz de procesadores vectoriales 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 L1 adjunta 3151 que es específica para cada matriz o está compartida entre las matrices. La caché puede estar configurada como una caché de datos, una caché de instrucciones o una única caché que está subdividida para contener datos e instrucciones en diferentes particiones.

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

En algunas realizaciones, pueden procesarse objetos geométricos completos por un sombreador de geometría 3119 mediante uno o más hilos despachados a unidades de ejecución 3152A-3152B, o pueden continuar directamente al recortador 3129. En algunas realizaciones, el sombreador de geometría opera en objetos geométricos enteros, en lugar de en vértices o parches de vértices como en etapas anteriores de la tubería de gráficos. Si se desactiva la teselación, el sombreador de geometría 3119 recibe entrada desde el sombreador de vértices 3107. En algunas realizaciones, el sombreador de geometría 3119 es programable por un programa sombreador de geometría para realizar teselación de geometría si las unidades de teselación están desactivadas.

Antes de la rasterización, un recortador 3129 procesa datos de vértices. El recortador 3129 puede ser un recortador de función fija o un recortador programable que tiene funciones de recortador y de sombreado de geometría. En algunas realizaciones, un componente de prueba de rasterizador y profundidad 3173 en la tubería de salida del representador 3170 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxeles. En algunas realizaciones, la lógica de sombreador de píxeles está incluida en la lógica

de ejecución de hilo 3150. En algunas realizaciones, una aplicación puede omitir el componente de prueba de rasterizador y profundidad 3173 y accede a datos de vértice no rasterizados mediante una unidad de salida de flujo 3123.

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

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

En algunas realizaciones, la tubería de medios del 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 tubería desde el transmisor de envío por flujo continuo 3103. En algunas realizaciones, la tubería de medios 3130 incluye un transmisor de envío por flujo continuo 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 funcionalidad de generación de hilo para generar hilos para despachar a lógica de ejecución de hilo 3150 mediante el despachador de hilo 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 mediante el anillo de interconexión 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 lógica de fin especial que puede operar independientemente de la tubería 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 adjunto mediante un conector de dispositivo de visualización.

En algunas realizaciones, la tubería de gráficos 3120 y la tubería de medios 3130 son configurables para realizar operaciones basándose en múltiples gráficos e interfaces de programación de medios y no son específicas a ninguna interfaz de programación de aplicación (API). En algunas realizaciones, el software del controlador para el procesador de gráficos traduce llamadas API que son específicas a gráficos o a bibliotecas de medios particulares en comandos que pueden procesarse por el procesador de gráficos. En algunas realizaciones, se proporciona soporte para la Biblioteca de Gráficos Abierta (OpenGL), Lenguaje Informático Abierto (OpenCL) y/o gráficos Vulkan y API de cálculos, todas a partir del grupo Khronos. En algunas realizaciones, puede proporcionarse también soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, puede soportarse una combinación de estas bibliotecas. Puede proporcionarse también soporte para la Biblioteca de Visión Informática de Código Abierto (OpenCV). También se soportaría una API futura con una tubería 3D compatible si pudiera hacerse un mapeo de la tubería de la API futura a la tubería del procesador de gráficos.

Programación de tubería de gráficos

La **Figura 32A** es un diagrama de bloques que ilustra un formato de orden de procesador de gráficos 3200 de acuerdo con algunas realizaciones. La **Figura 32B** es un diagrama de bloques que ilustra una secuencia de orden de procesador de gráficos 3210 de acuerdo con una realización. Los recuadros de línea continua en la **Figura 32A** ilustran los componentes que están incluidos en general en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que están incluidos únicamente en un subconjunto de 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 del comando (código de operación) 3204 y los datos relevantes 3206 para el comando. También se incluye 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 comando 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 representadora, una unidad 2D, una unidad 3D y una unidad de medios. Cada unidad de cliente tiene una tubería de procesamiento correspondiente que procesa los comandos. Una vez que se recibe el comando por la unidad de cliente, la unidad de cliente lee el código de operación 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 un tamaño de comando explícito 3208 para especificar el tamaño del comando. En algunas realizaciones, el analizador de comando determina automáticamente el tamaño de al menos alguno de los comandos basándose en el código de operación del comando. En algunas realizaciones, se alinean los comandos mediante múltiplos de una palabra doble.

El diagrama de flujo en la **Figura 32B** muestra una secuencia de comandos de procesador de gráficos ilustrativo 3210. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que presenta 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 una secuencia de comandos de muestra y se describe para los fines de ejemplo únicamente ya que las realizaciones no están limitadas a estos comandos específicos o para esta secuencia de comandos. Además, pueden emitirse los comandos como un lote de comandos en una secuencia de comandos, de manera que el procesador de gráficos procesará la secuencia de comandos en al menos parcialmente concurrencia.

En algunas realizaciones, la secuencia de comandos del procesador de gráficos 3210 puede comenzar con un comando de vaciado de tubería 3212 para hacer que alguna tubería de gráficos activa complete los comandos actualmente pendientes para la tubería. En algunas realizaciones, la tubería 3D 3222 y la tubería de medios 3224 no operan concurrentemente. Se realiza el vaciado de la tubería para hacer que la tubería de gráficos activa complete algún comando pendiente. En respuesta a un vaciado de tubería, el analizador de comando 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 pertinentes. Opcionalmente, cualquier dato en la caché del representador que se marca 'sucio' puede vaciarse a memoria. En algunas realizaciones, puede usarse el comando de vaciado de tubería 3212 para la sincronización de tubería o antes de colocar el procesador de gráficos en un estado de baja potencia.

En algunas realizaciones, se usa un comando de selección de tubería 3213 cuando una secuencia de comandos requiere el procesador de gráficos para conmutar explícitamente entre tuberías. En algunas realizaciones, se requiere únicamente un comando de selección de tubería 3213 una vez dentro de un contexto de ejecución antes de emitir comandos de tubería a menos que el contexto sea emitir comandos para ambas tuberías. En algunas realizaciones, se requiere un comando de vaciado de tubería 3212 inmediatamente antes de un conmutador de tubería mediante el comando de selección de tubería 3213.

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

En algunas realizaciones, se usan comandos de estado de memoria intermedia de retorno 3216 para configurar un conjunto de memorias intermedias de retorno para que las respectivas tuberías escriban datos. Algunas operaciones de tubería 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 realizar comunicación de hilo cruzada. En algunas realizaciones, el estado de la memoria intermedia de retorno 3216 incluye seleccionar el tamaño y número de memorias intermedias de retorno para usar para un conjunto de operaciones de tubería.

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

Los comandos para configurar el estado de tubería 3D 3230 incluyen los comandos de ajuste de estado 3D para el estado de memoria intermedia de vértice, estado de elemento de vértice, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que han de configurarse antes de que se procesen los comandos de primitiva 3D. Los valores de estos comandos se determinan, al menos en parte, basándose en la API 3D particular en uso. En algunas realizaciones, los comandos de estado de tubería 3D 3230 también pueden desactivar o desviar selectivamente ciertos elementos de tubería si no se usarán estos elementos.

En algunas realizaciones, se usa el comando de primitiva 3D 3232 para enviar que se procesen primitivas 3D por la tubería 3D. Los comandos y parámetros asociados que se pasan al procesador de gráficos mediante el comando de primitiva 3D 3232 se reenvían a la función de extracción de vértice en la tubería de gráficos. La función de extracción de vértice usa los datos de comando de 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, se usa el comando de primitiva 3D 3232 para realizar operaciones de vértice en primitivas 3D mediante sombreadores de vértice. Para procesar sombreadores de vértice, la tubería 3D 3222 despacha hilos de ejecución de sombreador a las unidades de ejecución de procesador de gráficos.

En algunas realizaciones, se activa la tubería 3D 3222 mediante un comando o evento de ejecución 3234. En algunas realizaciones, una escritura de registro activa la ejecución de comando. En algunas realizaciones, se activa la ejecución mediante un comando 'ir' o 'disparar' en la secuencia de comandos. En una realización, se activa la ejecución de comando usando un comando de sincronización de tubería para vaciar la secuencia de comandos a través de la tubería de gráficos. La tubería 3D realizará un procesamiento de geometría para las primitivas 3D. Una vez que están completadas las operaciones, se rasterizan los objetos geométricos resultantes y los colores de motor de píxel y los píxeles resultantes. Pueden incluirse también comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo trasero de píxeles para estas operaciones.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 3210 sigue la ruta de tubería de medios 3224 cuando se realizan operaciones de medios. En general, el uso y manera específicos de la programación para la tubería de medios 3224 depende de las operaciones de medios o de cálculo que van a realizarse. Las operaciones de decodificación de medios específicas pueden descargarse en la tubería de medios durante la decodificación de medios. En algunas realizaciones, puede desviarse también la tubería de medios y puede realizarse la decodificación de medios, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de fin general. En una realización, la tubería de medios también incluye elementos para las operaciones de la unidad de procesador de gráficos de fin general (GPGPU), donde se usa el procesador de gráficos para realizar operaciones vectoriales SIMD usando programas sombreadores computacionales que no están relacionados explícitamente con la representación de las primitivas de gráficos.

En algunas realizaciones, se configura la tubería de medios 3224 de una manera similar que la tubería 3D 3222. Un conjunto de comandos para configurar el estado de tubería de medios 3240 se despacha o coloca en una cola de comandos antes de los comandos de objeto de medios 3242. En algunas realizaciones, los comandos de estado de tubería de medios 3240 incluyen datos para configurar los elementos de tubería de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación de vídeo y codificación de vídeo dentro de la tubería de medios, tal como el formato de codificación o decodificación. En algunas realizaciones, los comandos de estado de tubería de medios 3240 también soportan el uso de uno o más punteros a elementos de estado "indirecto" 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 tubería de medios. Los objetos de medios incluyen memorias intermedias que contienen datos de vídeo que van a procesarse. En algunas realizaciones, todos los estados de tubería de medios deben ser válidos antes de que se emita un comando de objeto de medios 3242. Una vez que está configurado el estado de la tubería y se ponen en cola los comandos de objeto de medios 3242, se activa la tubería de medios 3224 mediante un comando de ejecución 3244 o un evento de ejecución equivalente (por ejemplo, escritura de registro). La salida de la tubería de medios 3224 puede post procesarse a continuación por las operaciones proporcionadas por la tubería 3D 3222 o la tubería de medios 3224. En algunas realizaciones, las operaciones de GPGPU se configuran y ejecutan de una manera similar a 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 núcleo o núcleos de procesador de fin general 3334. Cada uno de la aplicación de gráficos 3310 y el sistema operativo 3320 se ejecutan 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 sombreadores que incluyen instrucciones del 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 OpenGL (GLSL). La aplicación también incluye instrucciones ejecutables 3314 en un lenguaje máquina adecuado para su ejecución por el núcleo de procesador de fin general 3334. La aplicación también incluye los objetos de gráficos 3316 definidos por los datos de vértices.

En algunas realizaciones, el sistema operativo 3320 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo similar a UNIX propietario o un sistema operativo similar a UNIX de código abierto que usa una variante del núcleo Linux. El sistema operativo 3320 puede soportar una API de gráficos 3322 tal como la API Direct3D, la API OpenGL o la API Vulkan. Cuando está en uso la API Direct3D, el sistema operativo 3320 usa un compilador de sombreador de extremo frontal 3324 para compilar cualquier instrucción de sombreador 3312 en HLSL en un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede realizar una compilación previa de sombreador. En algunas realizaciones, los sombreadores de alto nivel se compilan en sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 3310. En algunas realizaciones, las instrucciones del sombreador 3312 se proporcionan en una forma intermedia, tal como una versión de la Representación Intermedia Portátil Convencional (SPIR) usada por la API 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 del sombreador 3312 en una representación específica de hardware. Cuando está en uso la API OpenGL, las instrucciones del 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 pueden implementarse por código representativo almacenado en un medio legible por máquina que representa y/o define lógica dentro de un circuito integrado tal como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan diversa lógica dentro del procesador. Cuando se leen por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en el presente documento. Tales representaciones, conocidas como "núcleos de IP", son unidades reutilizables de lógica para un circuito integrado que pueden almacenarse en un medio legible por máquina tangible como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware puede suministrarse a diversos clientes o instalaciones de fabricación, que cargan el modelo de hardware en máquinas de fabricación que fabrican el circuito integrado. El circuito integrado puede fabricarse de manera que el circuito realiza las 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 puede usarse para fabricar un circuito integrado para realizar las operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo de IP 3400 puede usarse para generar diseños reutilizables modulares que pueden incorporarse en un diseño más grande o usarse para construir un circuito integrado entero (por ejemplo, un circuito de SOC integrado). Una instalación de diseño 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++). La simulación de software 3410 puede usare para diseñar, probar 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. A continuación, puede crearse o sintetizarse 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, que incluyen lógica asociada realizada usando las señales digitales modeladas. Además de un diseño de RTL 3415, los diseños de nivel inferior en el nivel de lógica o en el nivel de transistores también pueden crearse, diseñarse o sintetizarse. Por lo tanto, los detalles particulares del diseño y simulación inicial pueden variar.

El diseño de RTL 3415 o equivalente puede sintetizarse adicionalmente por la instalación de diseño en 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 puede simularse o probarse adicionalmente para verificar el diseño de núcleo de IP. El diseño de núcleo de IP puede almacenarse para su entrega a una instalación de fabricación de 3os 3465 usando 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 puede transmitirse (por ejemplo, mediante Internet) a través de una conexión alámbrica 3450 o conexión inalámbrica 3460. La instalación de fabricación 3465 puede a continuación fabricar un circuito integrado que está basado al menos en parte en el diseño de núcleo de IP. El circuito integrado fabricado puede estar configurado para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Sistema ilustrativo en un circuito de chip integrado

Las Figuras 35-37 ilustran circuitos integrados de manera ilustrativa y procesadores de gráficos asociados que pueden fabricarse usando uno o más núcleos de IP, de acuerdo con diversas realizaciones descritas en el presente documento. Además de lo que se ilustra, puede incluirse otra lógica y circuitos, que incluyen procesadores/núcleos de gráficos adicionales, controladores de interfaz periférica o núcleos de procesador de fin general.

La **Figura 35** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ilustrativo 3500 que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 3500 ilustrativo incluye uno o más procesador o procesadores de aplicación 3505 (por ejemplo, las 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 que puede ser un núcleo de IP modular desde las mismas o múltiples diferentes instalaciones de diseño. El circuito integrado 3500 incluye lógica de periférico o de bus que incluye un controlador de USB 3525, controlador de UART 3530, un controlador de SPI/SDIO 3535 y un controlador de I2S/I2C 3540. Adicionalmente, el circuito integrado puede incluir un dispositivo de visualización 3545 acoplado a uno o más de un controlador de interfaz multimedia de alta definición (HDMI) 3550 y una interfaz de pantalla de procesador industrial móvil (MIPI) 3555. El almacenamiento puede proporcionarse por un subsistema de memoria flash 3560 que incluye la memoria flash y un controlador de memoria flash. La interfaz de memoria puede proporcionarse mediante un controlador de memoria 3565 para acceso a dispositivos de memoria de 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 puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El procesador de gráficos 3610 puede tener 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 procesador o 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 sombreadores mediante lógica separada, de manera que el procesador de vértices 3605 está optimizado para ejecutar operaciones para programas de sombreador de vértices, mientras que el uno o más procesador o procesadores de fragmentos 3615A-3615N ejecutan operaciones de sombreado de fragmento (por ejemplo, de píxeles) para programas sombreadores de fragmento o de píxel. El procesador de vértices 3605 realiza la etapa de procesamiento de vértices de la tubería de gráficos 3D y genera datos de primitivas y de vértices. El procesador o procesadores de fragmentos 3615A-3615N usan datos de primitiva y de vértice generados por el procesador de vértices 3605 para producir una memoria intermedia de fotograma que se visualiza en un dispositivo de visualización. En una realización, el procesador o procesadores de fragmentos 3615A-3615N están optimizados para ejecutar programas sombreadores de fragmento según se proporciona en la API OpenGL, que pueden usarse para realizar operaciones similares como un programa sombreador de píxeles como se proporciona en la API 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. La una o más MMU 3620A-3620B proporcionan mapeo 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 los datos de vértice o de imagen/textura almacenados en memoria, además de los datos de vértice o imagen/textura almacenados en la una o más caché o cachés 3625A-3625B. En una realización, la una o más MMU 3625A-3625B pueden estar sincronizadas con otras MMU dentro del sistema, que incluyen una o más MMU asociadas con el uno o más procesador o procesadores de aplicación 3605, el procesador de imagen 3615 y/o el procesador de vídeo 3620 de la **Figura 36,** de manera que cada procesador 3605-3620 puede participar en un sistema virtual de memoria compartida o unificada. La una o más interconexión o interconexiones de circuito 3630A-3630B posibilitan que el procesador de gráficos 3610 interconecte con otros núcleos de IP dentro del SoC, mediante un bus interno del SoC o mediante una conexión directa, de acuerdo con las 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 puede fabricarse 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 la una o más MMU 3520A-3520B, las cachés 3525A-3525B y las interconexiones de circuito 3530A-3530B del circuito integrado 3500 de la **Figura 35.**

El procesador de gráficos 3710 incluye uno o más núcleo o núcleos de sombreador 3715A-3715N (por ejemplo, 3715A, 3715B, 3715C, 3715D, 3715E, 3715F a 3715N-1, y 3715N), que proporcionan una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo o núcleo puede ejecutar todos los tipos de código sombreador programable, que incluyen código de programa sombreador para implementar sombreadores de vértice, sombreadores de fragmento y/o sombreadores de cálculo. 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 inter-núcleo 3705, que actúa como un despachador de hilo para despachar hilos de ejecución a uno o más núcleos de sombreador 3715A-3715N y una unidad de mosaico 3718 para acelerar las operaciones de mosaico para la representación basada en mosaico, en las que las operaciones de representación para una escena se subdividen en el espacio de imágenes, por ejemplo, para aprovechar la coherencia de espacio local dentro de una escena o para optimizar el uso de cachés internas.

Las siguientes cláusulas y/o ejemplos pertenecen a realizaciones específicas o ejemplos de las mismas. Los detalles específicos en los ejemplos pueden usarse en cualquier parte en una o más realizaciones. Las diversas características de las diferentes realizaciones o ejemplos pueden combinarse de manera variable con algunas características

incluidas y otras excluidas para adecuarse a una diversidad de diferentes aplicaciones. Los ejemplos pueden incluir la materia objeto tal como un método, medios para realizar actos del método, al menos un medio legible por máquina que incluye instrucciones que, cuando se realizan por una máquina, hacen que la máquina realice actos del método, o de un aparato o sistema de acuerdo con las realizaciones y ejemplos descritos en el presente documento. Diversos componentes pueden ser un medio para realizar las operaciones o funciones descritas.

Las realizaciones descritas en el presente documento hacen referencia a configuraciones de hardware específicas, tales como circuitos integrados específicos de la aplicación (ASIC), configuradas para realizar ciertas operaciones o que tienen una funcionalidad predeterminada. Tales dispositivos electrónicos típicamente incluyen un conjunto de uno o más procesadores acoplados a uno o más otros 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 es típicamente a través de uno o más buses y puentes (también denominado controladores de bus). El dispositivo de almacenamiento y las señales que llevan el tráfico de red representan respectivamente uno o más medios de almacenamiento legibles por máquina y medios de comunicación legibles por máquina. Por lo tanto, los dispositivos de almacenamiento de un dispositivo electrónico dado típicamente almacenan 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 pueden implementarse usando diferentes combinaciones de software, firmware y/o hardware. A través de toda esta descripción detallada, para los fines de explicación, se exponen numerosos detalles específicos para proporcionar un entendimiento minucioso de la presente invención. Sin embargo, será evidente para un experto en la materia que las realizaciones pueden ponerse en práctica sin algunos de estos detalles específicos. En ciertos casos, no se describen estructuras y funciones bien conocidas en detalle elaborado para evitar oscurecer la materia objeto inventiva de las realizaciones. Por consiguiente, el alcance de la invención debe determinarse en términos de las reivindicaciones que siguen.

REIVINDICACIONES

1.Un aparato de cálculo (1400) para realizar operaciones de aprendizaje automático, comprendiendo el aparato de cálculo:

una unidad de decodificación (1421) para decodificar una instrucción sencilla en una instrucción decodificada, la instrucción decodificada para hacer que el aparato de cálculo realice una operación de cálculo de aprendizaje automático compleja, en donde la operación de cálculo de aprendizaje automático compleja es para realizar una convolución para una red neuronal convolucional, en donde la convolución incluye múltiples operaciones matriciales;
una unidad de extracción (1421) para extraer la instrucción sencilla;
lógica de análisis de parámetro (1512) para determinar un tipo de operaciones de aprendizaje automático para realizar para la instrucción sencilla basándose en parámetros que incluyen dimensiones de filtro convolucional;
lógica de aceleración de aprendizaje automático (1516) para determinar un conjunto de operaciones para realizar la instrucción decodificada;
un controlador del planificador (1422) para planificar las múltiples operaciones matriciales a uno o más de múltiples tipos de unidades de cálculo, en donde los múltiples tipos de unidades de cálculo incluyen una unidad de cálculo de gráficos de fin general y una unidad de cálculo de datos cercanos; y
un microcontrolador (1510) para ejecutar instrucciones de firmware, las instrucciones de firmware para posibilitar la lógica de análisis de parámetro y la lógica de aceleración de aprendizaje automático.

2.El aparato de cálculo de acuerdo con la reivindicación 1, en donde los múltiples tipos de unidades de cálculo incluyen adicionalmente una unidad de cálculo disperso.

3.Un método, que comprende:

decodificar, por una unidad de decodificación (1421) de un aparato de cálculo, una instrucción sencilla en una instrucción decodificada, la instrucción decodificada para hacer que el aparato de cálculo realice una operación de cálculo de aprendizaje automático compleja, en donde la operación de cálculo de aprendizaje automático compleja es para realizar una convolución para una red neuronal convolucional, en donde la convolución incluye múltiples operaciones matriciales;
extraer, por una unidad de extracción (1421) del aparato de cálculo, la instrucción sencilla;
determinar, por la lógica de análisis de parámetro (1512) del aparato de cálculo, un tipo de operaciones de aprendizaje automático para realizar para la instrucción sencilla basándose en parámetros que incluyen dimensiones de filtro convolucional;
determinar, por una lógica de aceleración de aprendizaje automático (1516), un conjunto de operaciones para realizar la instrucción decodificada;
planificar, por un controlador del planificador (1422), las múltiples operaciones matriciales a uno o más de múltiples tipos de unidades de cálculo, en donde los múltiples tipos de unidades de cálculo incluyen una unidad de cálculo de gráficos de fin general y una unidad de cálculo de datos cercanos; y
ejecutar, por un microcontrolador (1510), instrucciones de firmware, las instrucciones de firmware para posibilitar la lógica de análisis de parámetro y la lógica de aceleración de aprendizaje automático.

4.Al menos un medio legible por máquina que incluye instrucciones que, cuando se realizan por una máquina, hacen que la máquina realice un método de acuerdo con la reivindicación 3.

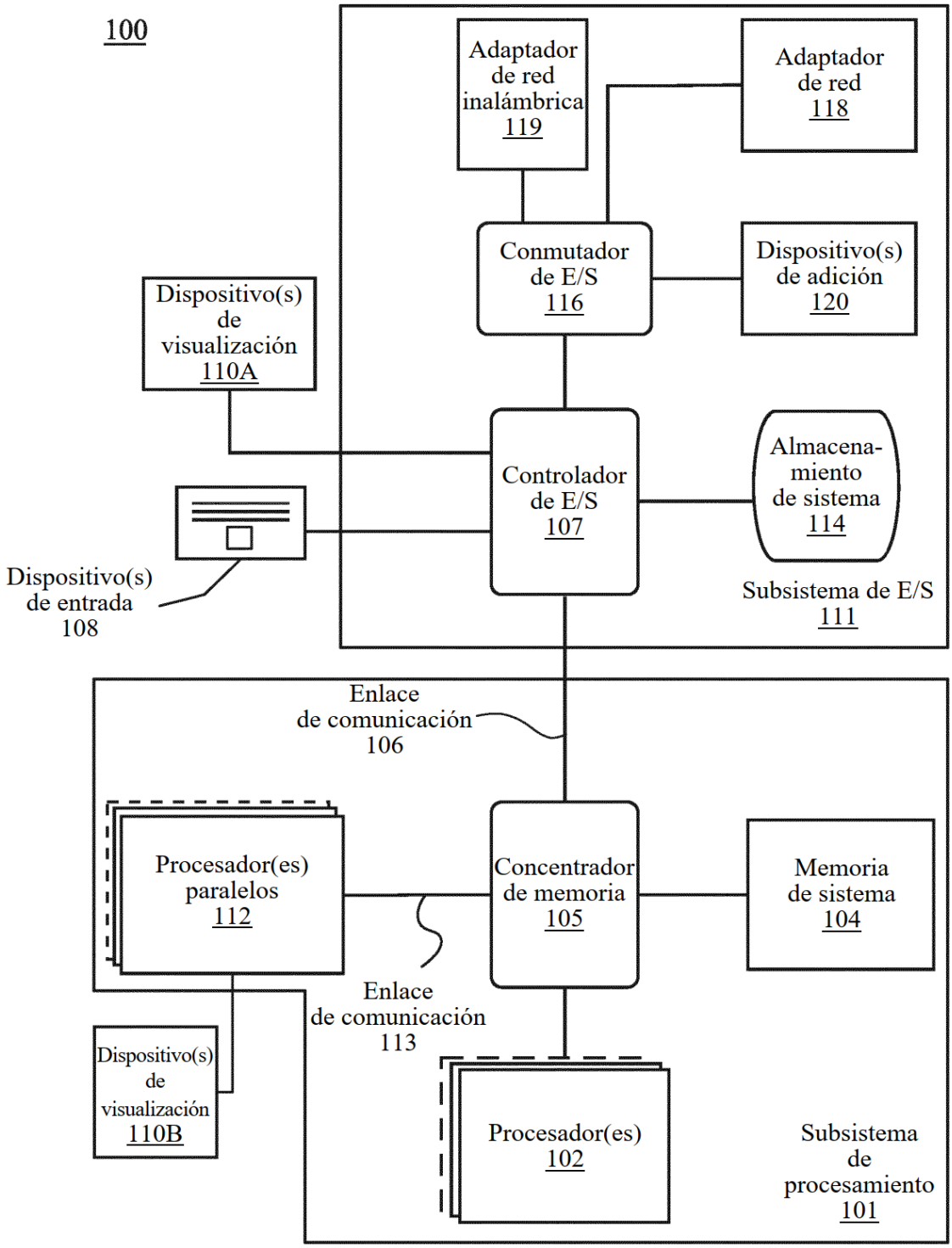

100
Adaptador de red inalámbrica 119
Adaptador de red 118
Conmutador de E/S 116
Dispositivo(s) de adición 120
Dispositivo(s) de visualización 110A
Controlador de E/S 107
Almacena-miento de sistema 114
Dispositivo(s) de entrada 108
Subsistema de E/S 111
Enlace de comunicación 106
Procesador(es) paralelos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

FIG. 1

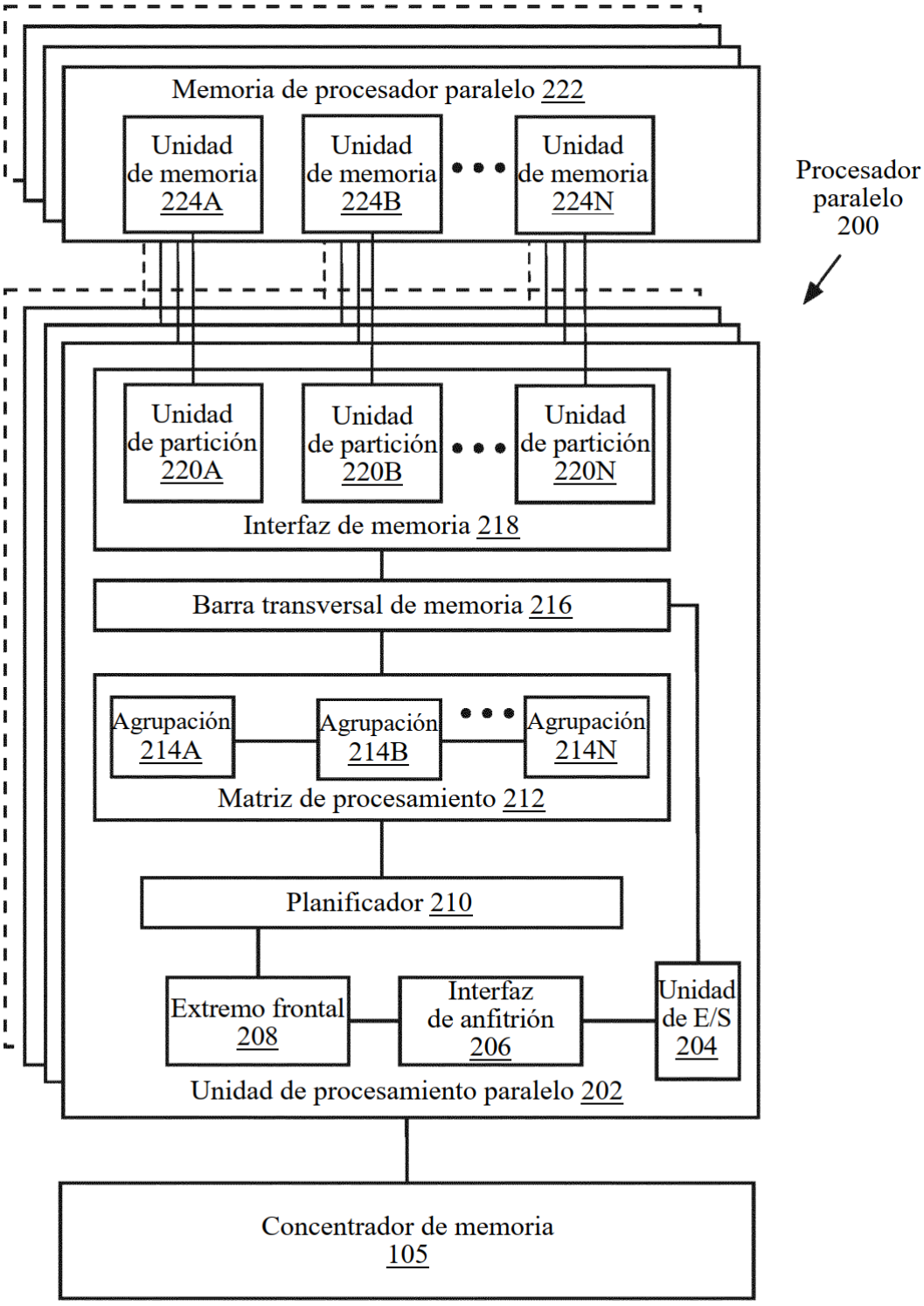

Memoria de procesador paralelo 222
Unidad de memoria 224A
Unidad de memoria 224B
Unidad de memoria 224N
Procesador paralelo 200
Unidad de partición 220A
Unidad de partición 220B
Unidad de partición 220N
Interfaz de memoria 218
Barra transversal de memoria 216
Agrupación 214A
Agrupación 214B
Agrupación 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

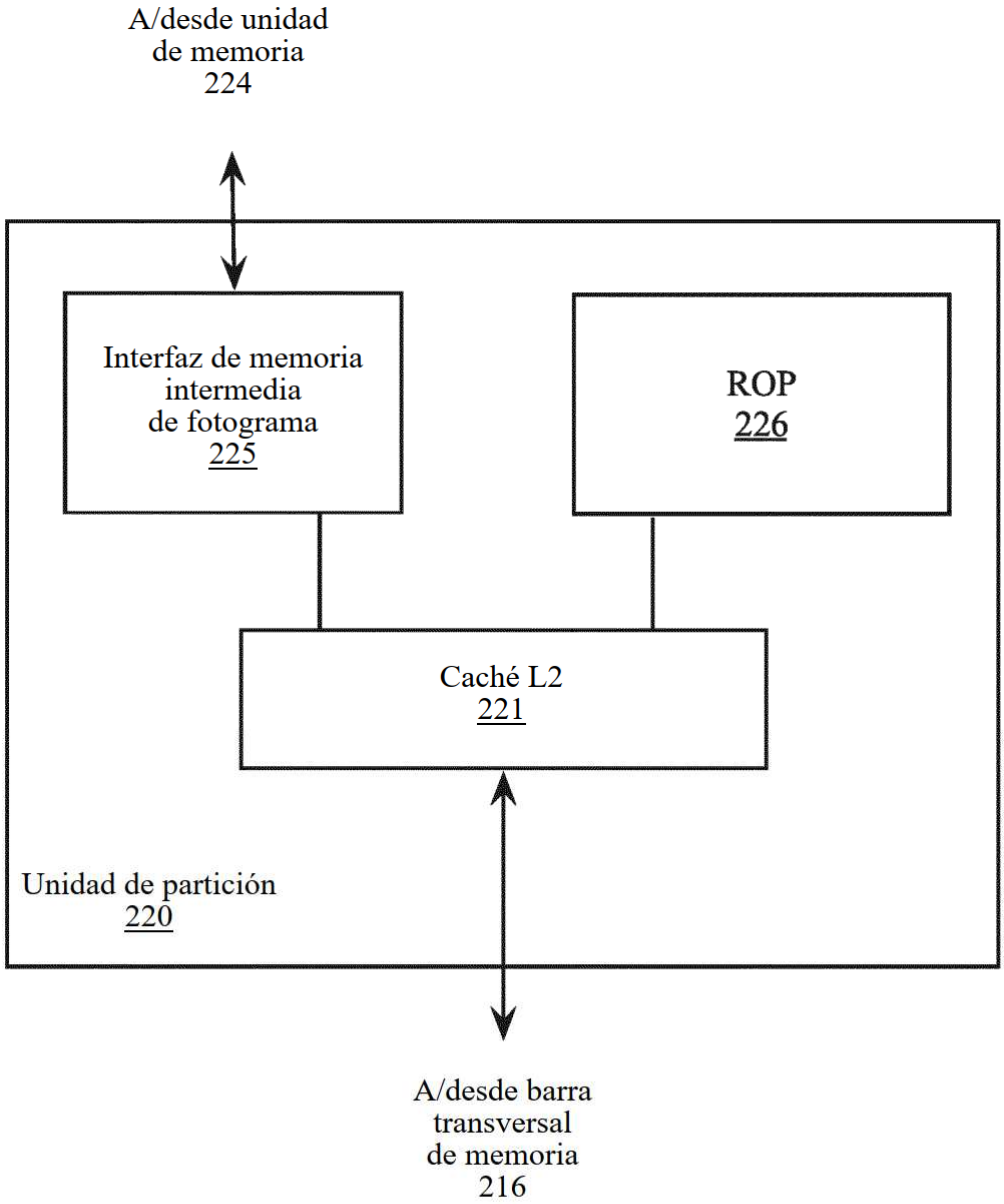

A/desde unidad
de memoria
224
Interfaz de memoria
intermedia
de fotograma
225
ROP
226
Caché L2
221
Unidad de partición
220
A/desde barra
transversal
de memoria
216

FIG. 2B

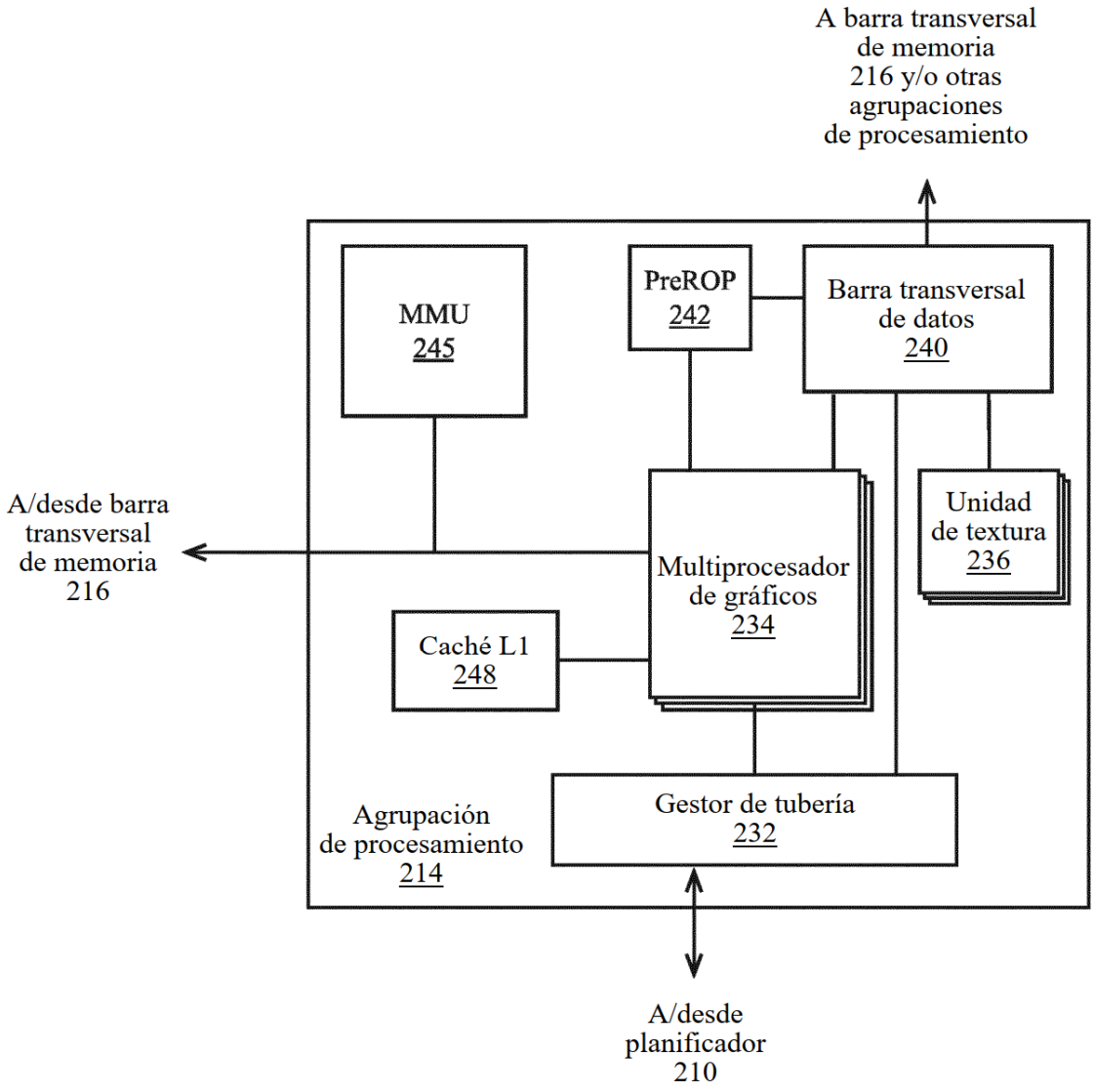

A barra transversal de memoria 216 y/o otras agrupaciones de procesamiento
MMU 245
PreROP 242
Barra transversal de datos 240
A/desde barra transversal de memoria 216
Multiprocesador de gráficos 234
Unidad de textura 236
Caché L1 248
Agrupación de procesamiento 214
Gestor de tubería 232
A/desde planificador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

FIG. 2C

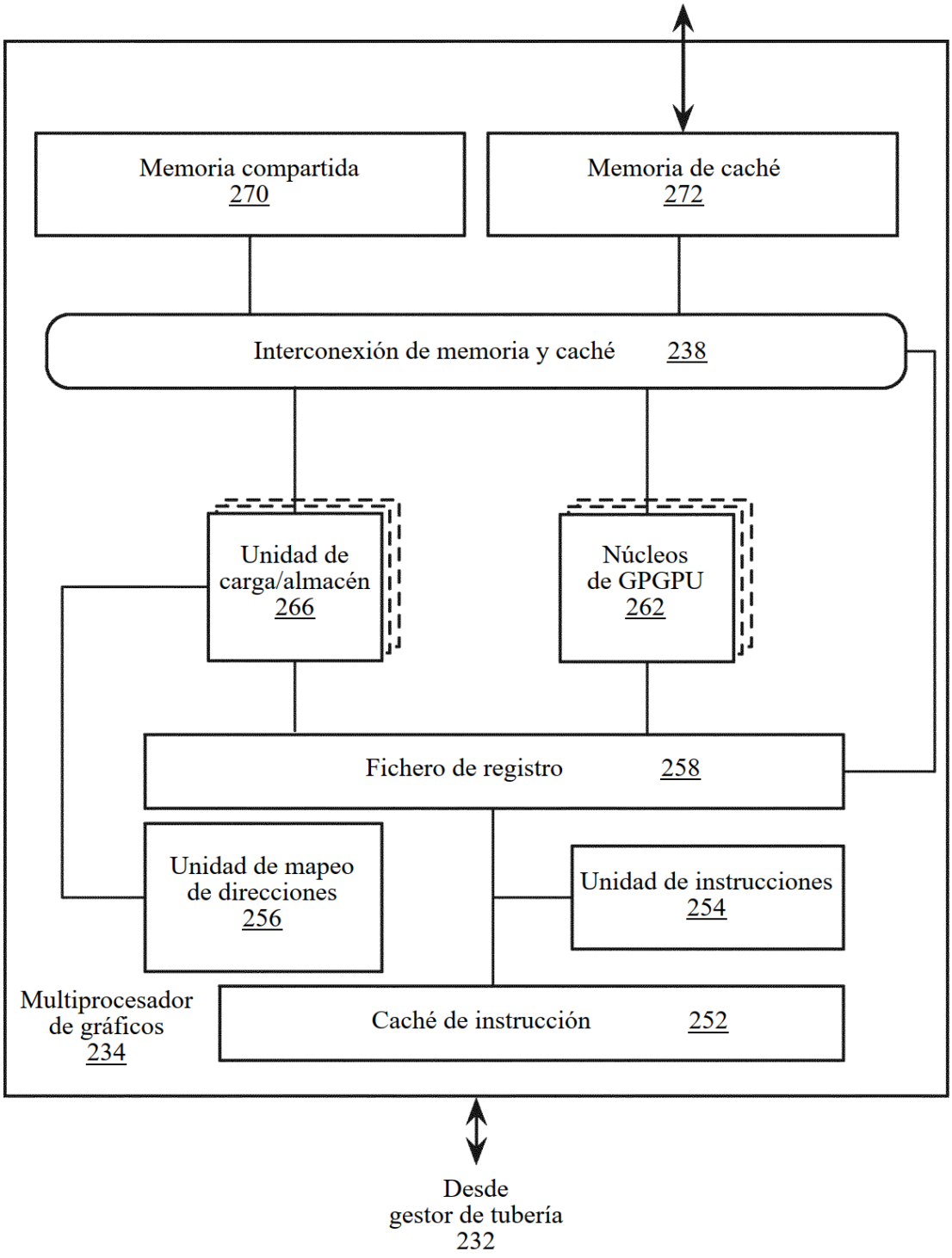

Memoria compartida
270
Memoria de caché
272
Interconexión de memoria y caché
238
Unidad de
carga/almacén
266
Núcleos
de GPGPU
262
Fichero de registro
258
Unidad de mapeo
de direcciones
256
Unidad de instrucciones
254
Multiprocesador
de gráficos
234
Caché de instrucción
252
Desde
gestor de tubería
232

FIG. 2D

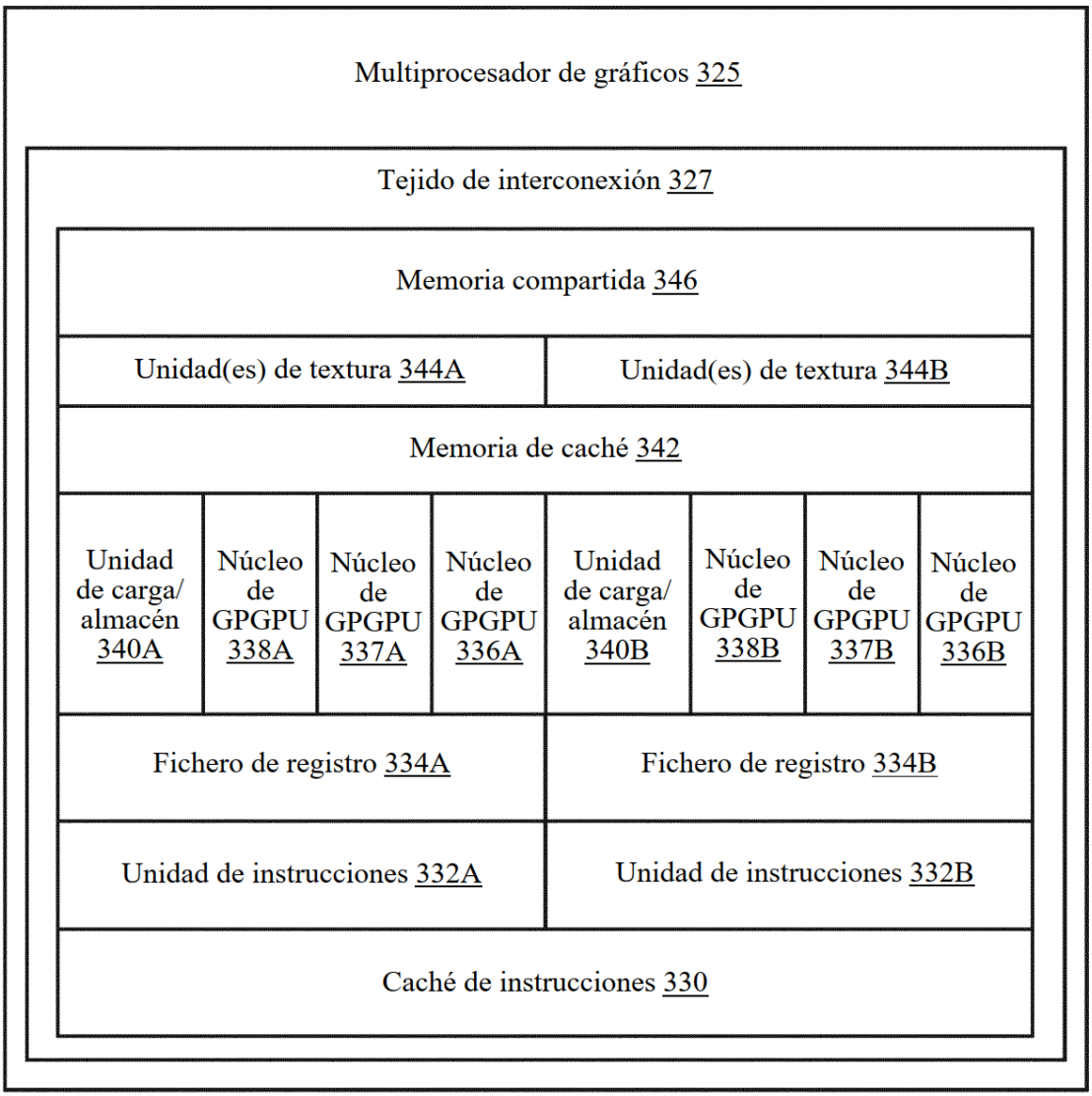

Multiprocesador de gráficos 325
Tejido de interconexión 327
Memoria compartida 346
Unidad(es) de textura 344A
Unidad(es) de textura 344B
Memoria de caché 342
Unidad de carga/ almacén 340A
Núcleo de GPGPU 338A
Núcleo de GPGPU 337A
Núcleo de GPGPU 336A
Unidad de carga/ almacén 340B
Núcleo de GPGPU 338B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

FIG. 3A

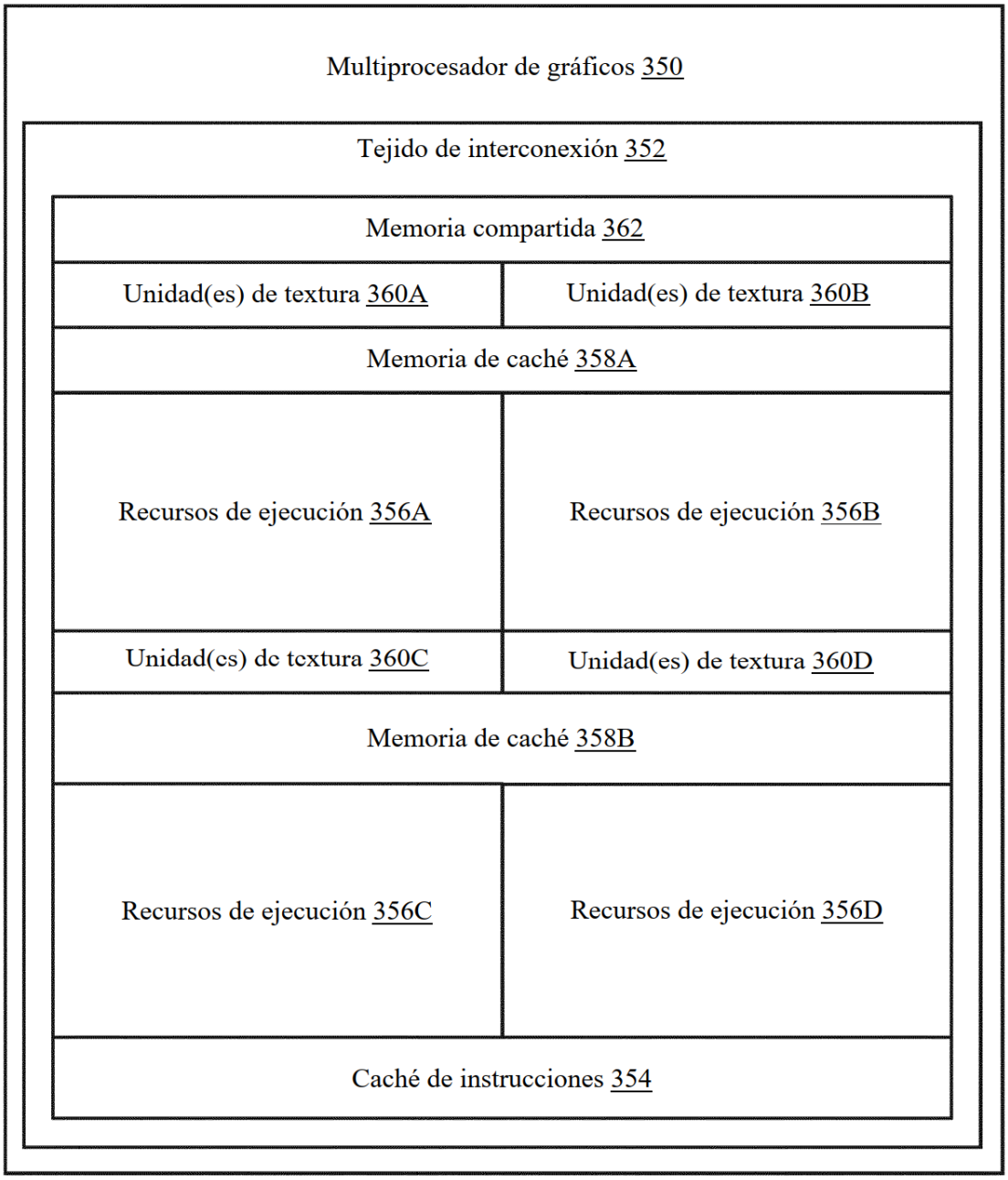

Multiprocesador de gráficos 350
Tejido de interconexión 352
Memoria compartida 362
Unidad(es) de textura 360A
Unidad(es) de textura 360B
Memoria de caché 358A
Recursos de ejecución 356A
Recursos de ejecución 356B
Unidad(es) de textura 360C
Unidad(es) de textura 360D
Memoria de caché 358B
Recursos de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

FIG. 3B

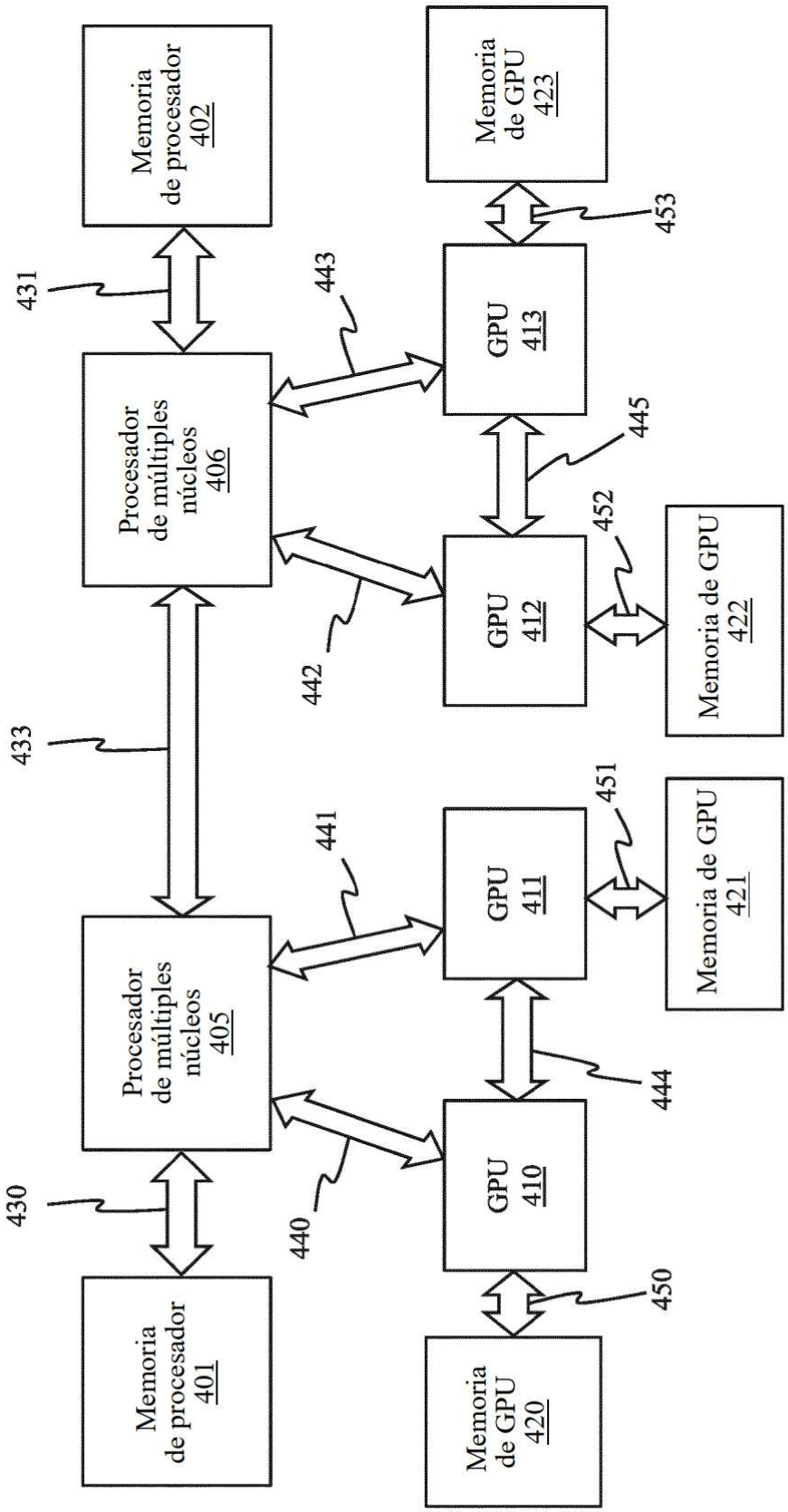

Memoria de procesador 402
Memoria de GPU 423
431
443
453
GPU 413
Procesador de múltiples núcleos 406
445
452
433
442
GPU 412
Memoria de GPU 422
441
451
GPU 411
Memoria de GPU 421
Procesador de múltiples núcleos 405
444
430
440
GPU 410
450
Memoria de procesador 401
Memoria de GPU 420

FIG. 4A

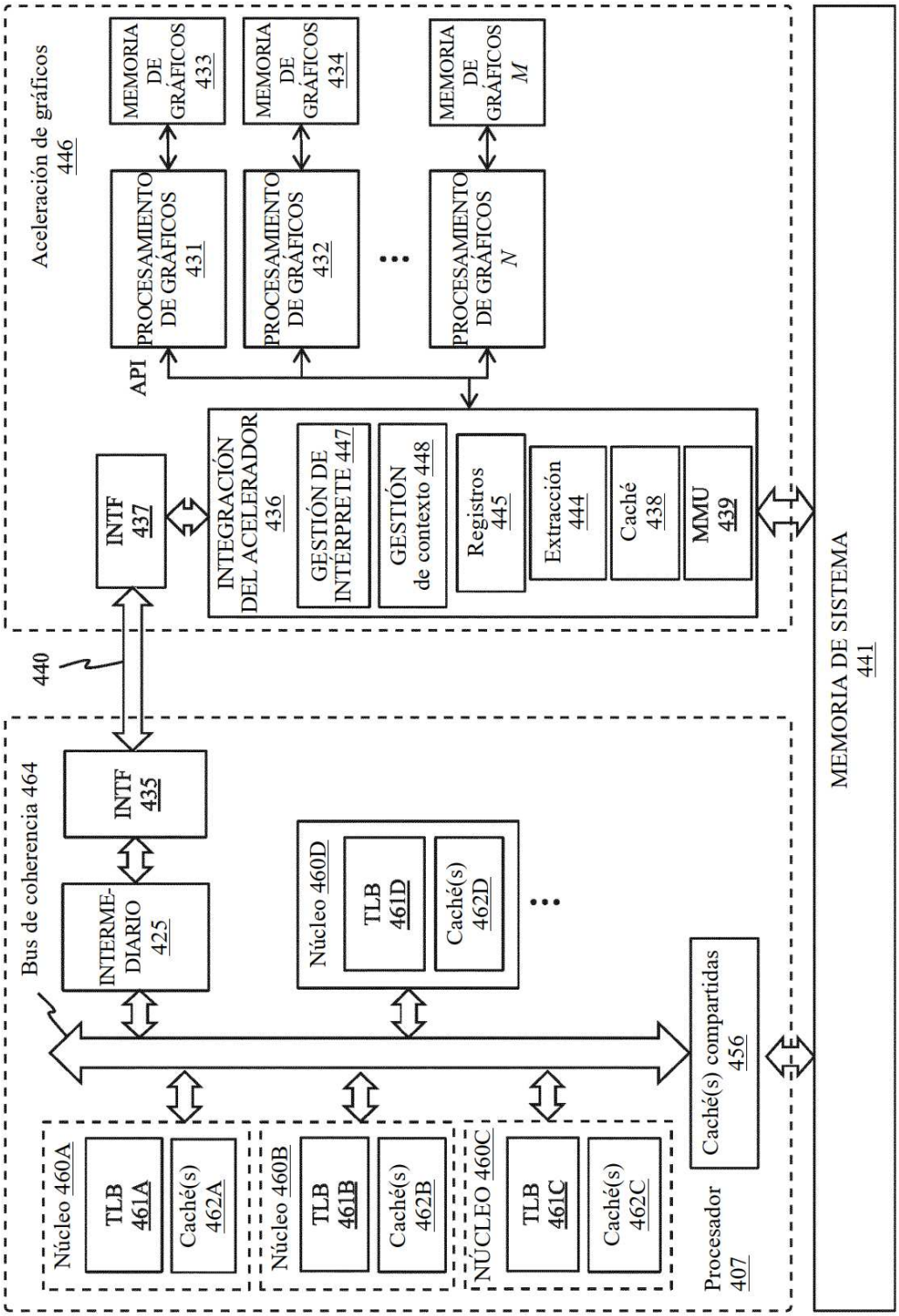

Aceleración de gráficos 446
MEMORIA DE GRÁFICOS 433
MEMORIA DE GRÁFICOS 434
MEMORIA DE GRÁFICOS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 437
INTEGRACIÓN DEL ACELERADOR 436
GESTIÓN DE INTÉRPRETE 447
GESTIÓN de contexto 448
Registros 445
Extracción 444
Caché 438
MMU 439
440
Bus de coherencia 464
INTF 435
INTERME-DIARIO 425
Núcleo 460D
TLB 461D
Caché(s) 462D
Caché(s) compartidas 456
Núcleo 460A
TLB 461A
Caché(s) 462A
Núcleo 460B
TLB 461B
Caché(s) 462B
NÚCLEO 460C
TLB 461C
Caché(s) 462C
Procesador 407
MEMORIA DE SISTEMA 441

FIG. 4B

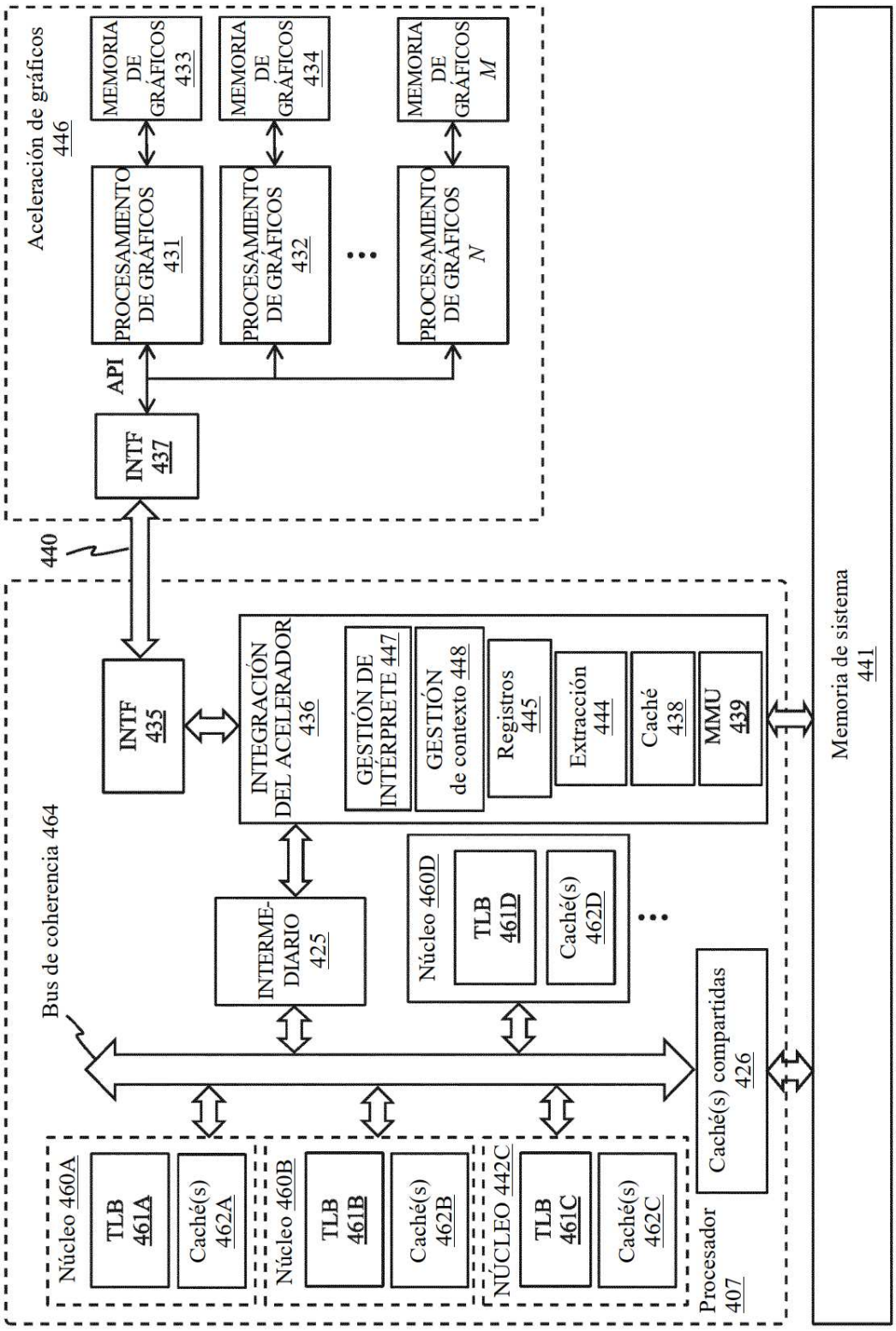

Aceleración de gráficos 446
MEMORIA DE GRÁFICOS 433
MEMORIA DE GRÁFICOS 434
MEMORIA DE GRÁFICOS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 437
440
INTF 435
INTEGRACIÓN DEL ACELERADOR 436
GESTIÓN DE INTÉRPRETE 447
GESTIÓN de contexto 448
Registros 445
Extracción 444
Caché 438
MMU 439
Bus de coherencia 464
INTERME-DIARIO 425
Núcleo 460D
TLB 461D
Caché(s) 462D
Núcleo 460A
TLB 461A
Caché(s) 462A
Núcleo 460B
TLB 461B
Caché(s) 462B
NÚCLEO 442C
TLB 461C
Caché(s) 462C
Caché(s) compartidas 426
Procesador 407
Memoria de sistema 441

FIG. 4C

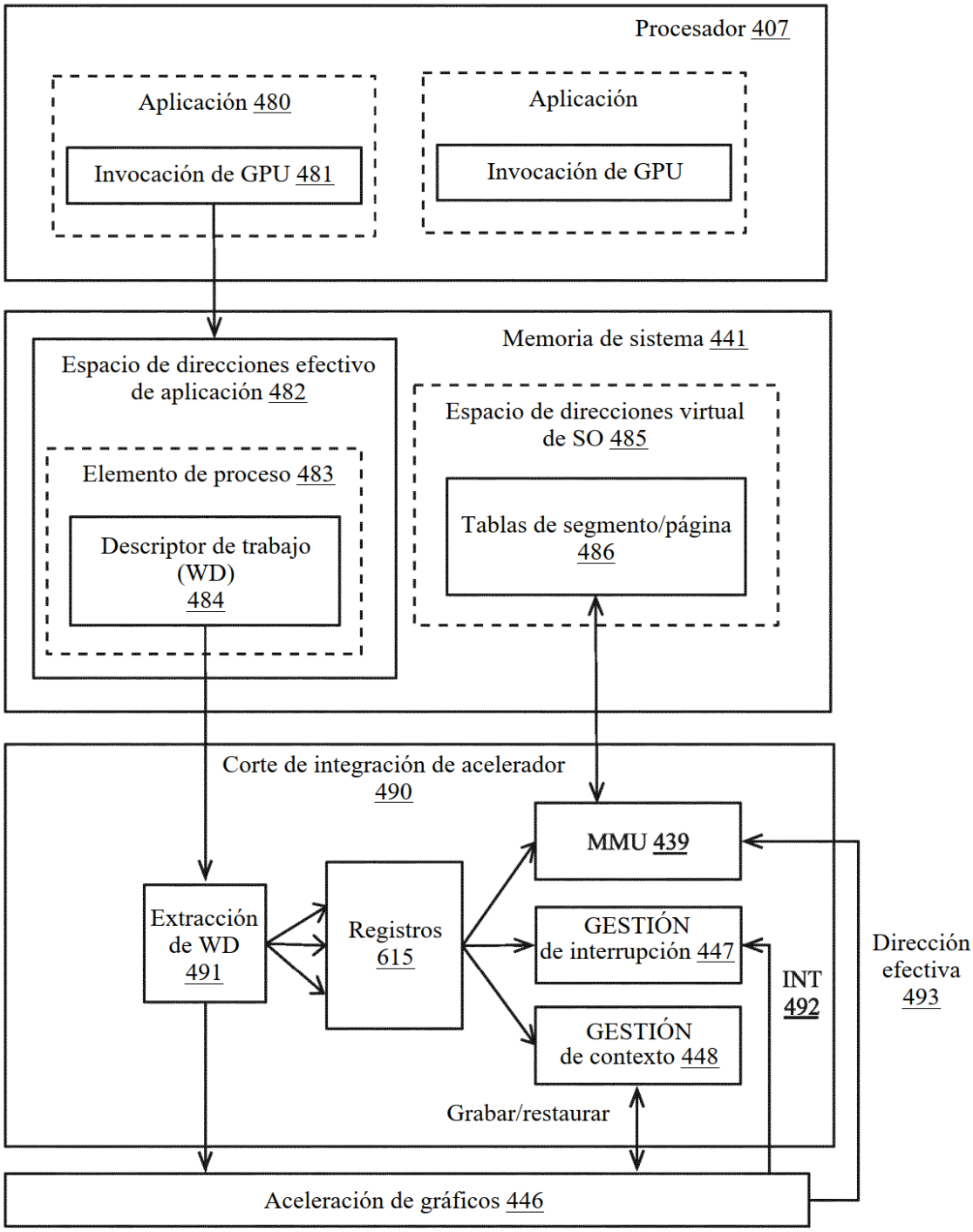

Procesador 407
Aplicación 480
Invocación de GPU 481
Aplicación
Invocación de GPU
Memoria de sistema 441
Espacio de direcciones efectivo de aplicación 482
Elemento de proceso 483
Descriptor de trabajo (WD) 484
Espacio de direcciones virtual de SO 485
Tablas de segmento/página 486
Corte de integración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

FIG. 4D

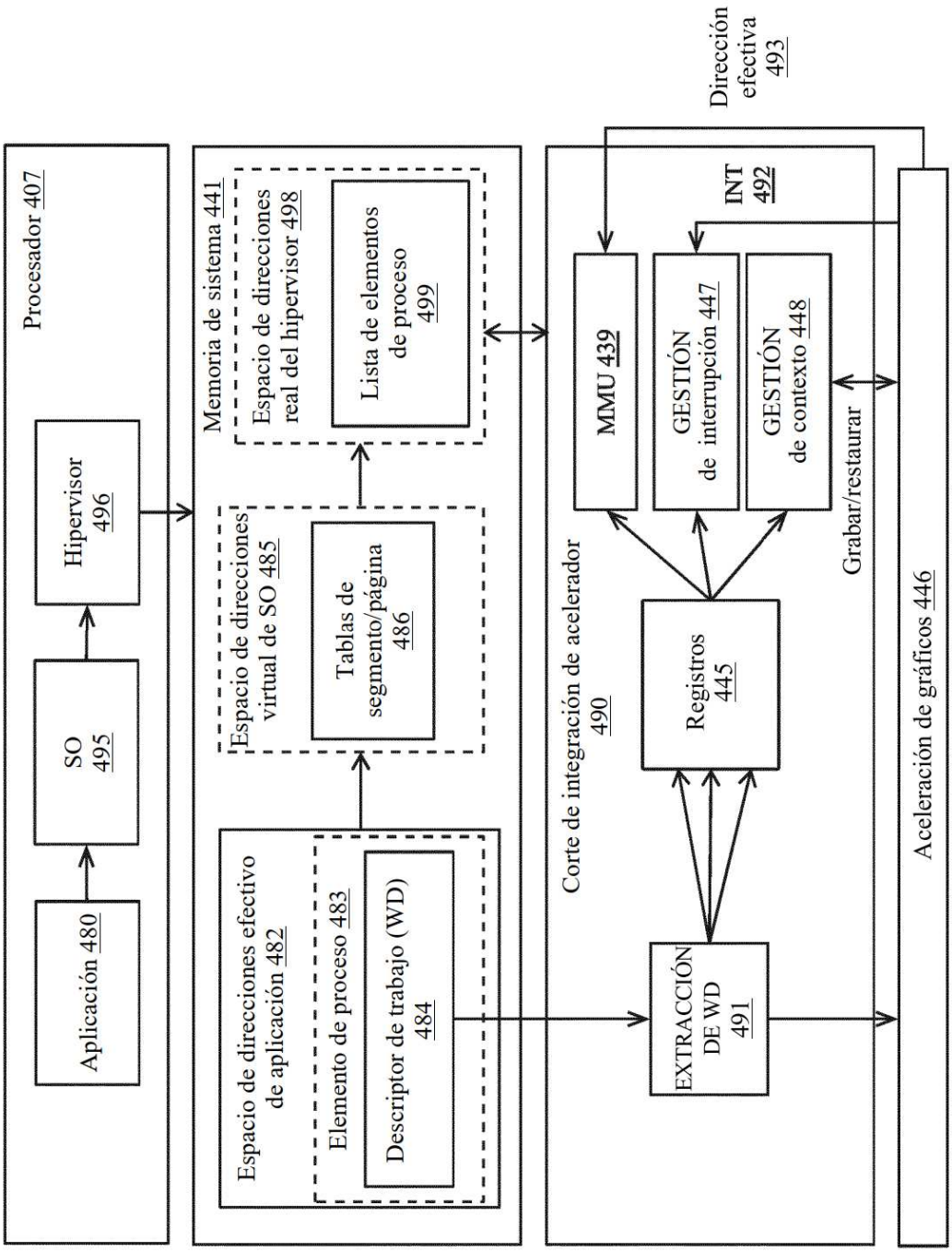

Procesador 407
Aplicación 480
SO 495
Hipervisor 496
Memoria de sistema 441
Espacio de direcciones efectivo de aplicación 482
Elemento de proceso 483
Descriptor de trabajo (WD) 484
Espacio de direcciones virtual de SO 485
Tablas de segmento/página 486
Espacio de direcciones real del hipervisor 498
Lista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

FIG. 4E

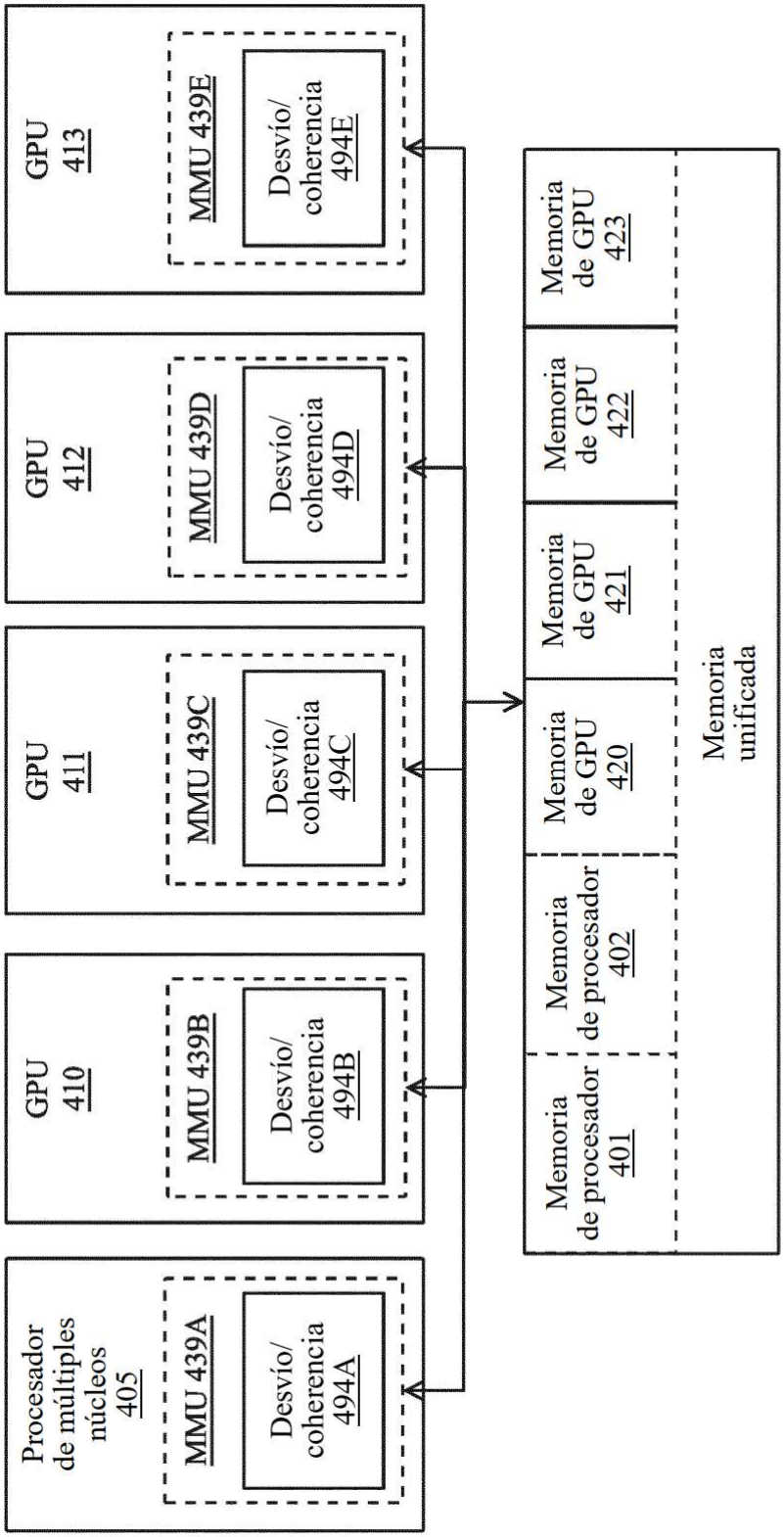

Procesador de múltiples núcleos 405
MMU 439A
Desvío/ coherencia 494A
GPU 410
MMU 439B
Desvío/ coherencia 494B
GPU 411
MMU 439C
Desvío/ coherencia 494C
GPU 412
MMU 439D
Desvío/ coherencia 494D
GPU 413
MMU 439E
Desvío/ coherencia 494E
Memoria de procesador 401
Memoria de procesador 402
Memoria de GPU 420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

FIG. 4F

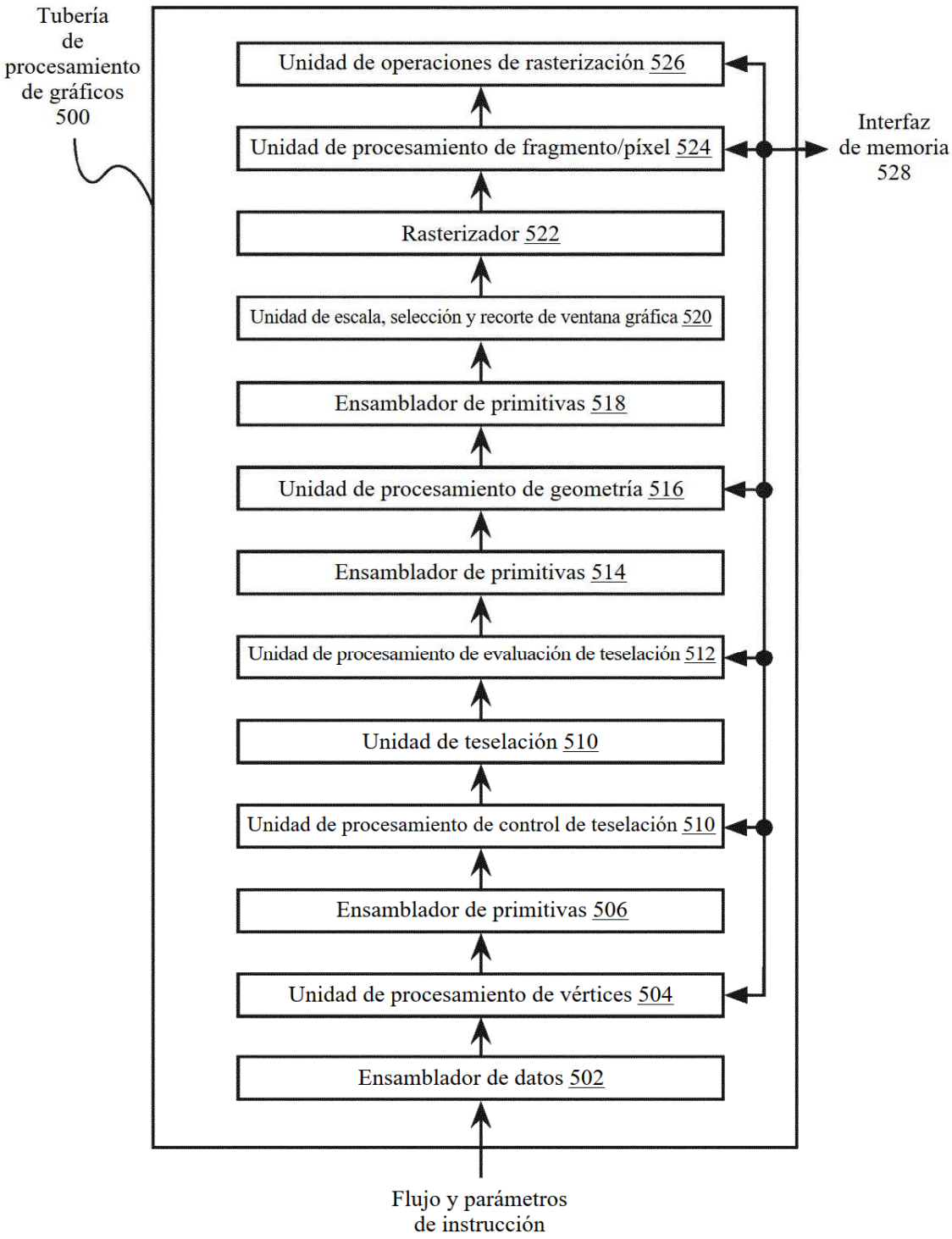

Tubería de procesamiento de gráficos 500
Unidad de operaciones de rasterización 526
Unidad de procesamiento de fragmento/píxel 524
Interfaz de memoria 528
Rasterizador 522
Unidad de escala, selección y recorte de ventana gráfica 520
Ensamblador de primitivas 518
Unidad de procesamiento de geometría 516
Ensamblador de primitivas 514
Unidad de procesamiento de evaluación de teselación 512
Unidad de teselación 510
Unidad de procesamiento de control de teselación 510
Ensamblador de primitivas 506
Unidad de procesamiento de vértices 504
Ensamblador de datos 502
Flujo y parámetros de instrucción

FIG. 5

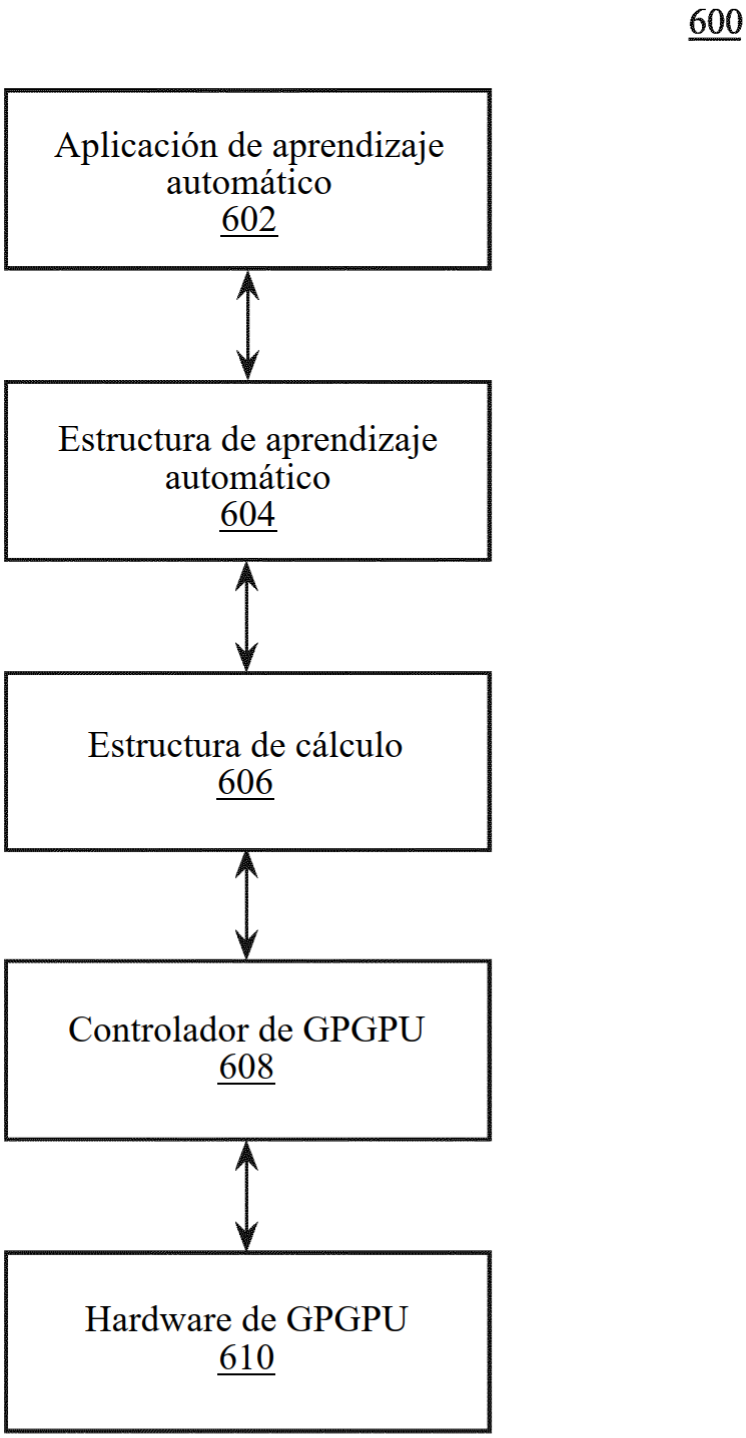

600
Aplicación de aprendizaje automático
602
Estructura de aprendizaje automático
604
Estructura de cálculo
606
Controlador de GPGPU
608
Hardware de GPGPU
610

FIG. 6

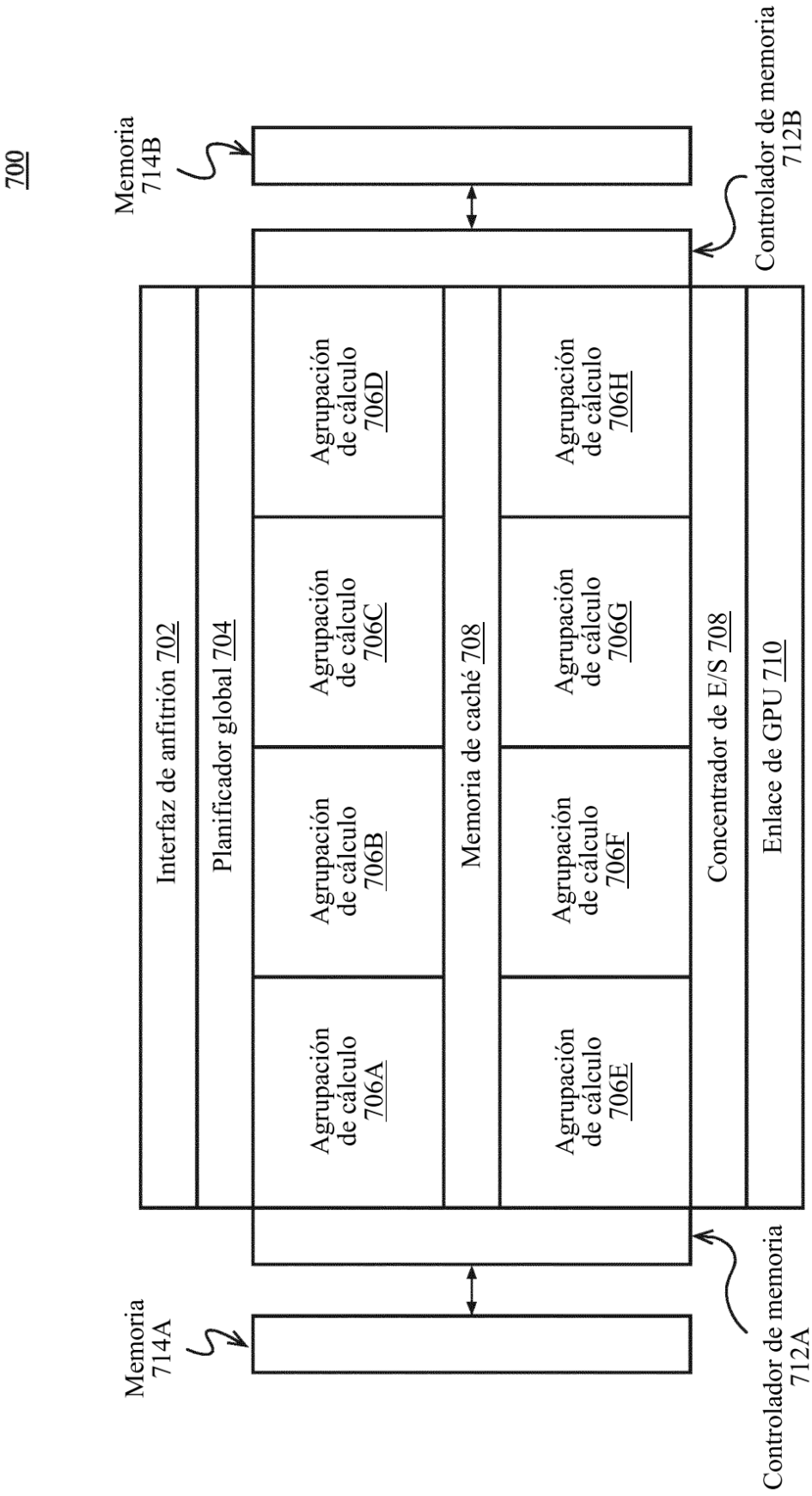

700
Memoria 714B
Controlador de memoria 712B
Interfaz de anfitrión 702
Planificador global 704
Agrupación de cálculo 706A
Agrupación de cálculo 706B
Agrupación de cálculo 706C
Agrupación de cálculo 706D
Memoria de caché 708
Agrupación de cálculo 706E
Agrupación de cálculo 706F
Agrupación de cálculo 706G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

FIG. 7

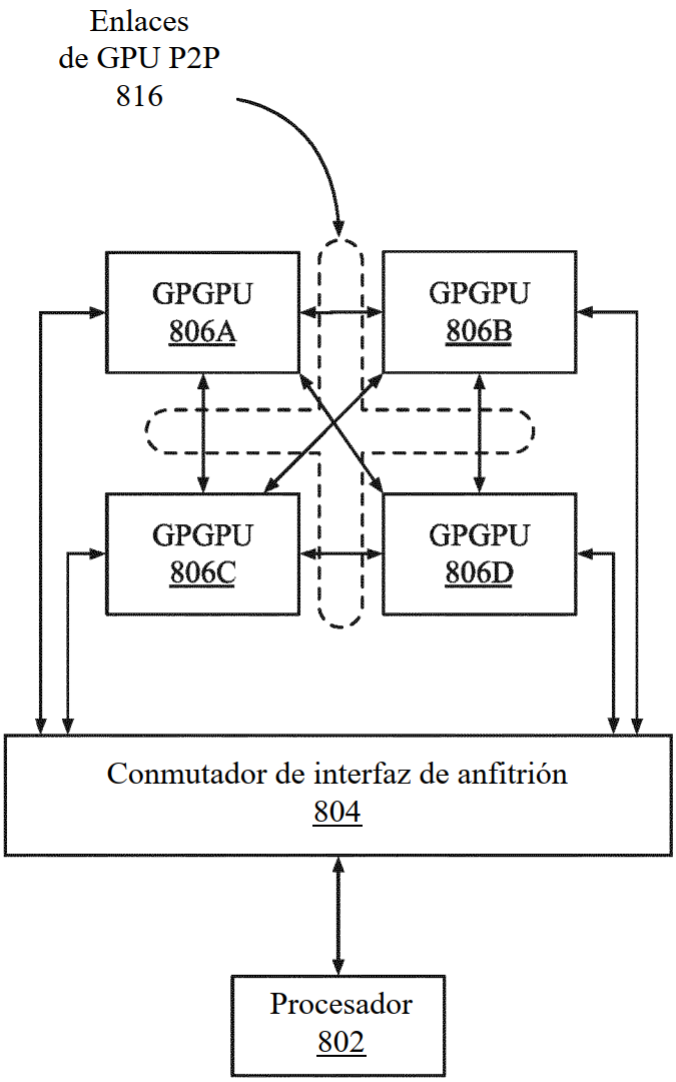

Enlaces
de GPU P2P
816
GPGPU
806A
GPGPU
806B
GPGPU
806C
GPGPU
806D
Conmutador de interfaz de anfitrión
804
Procesador
802

FIG. 8

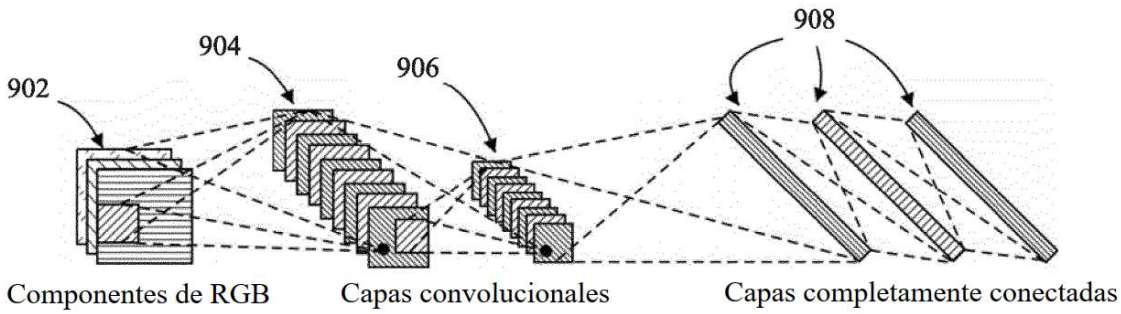

902
904
906
908
Componentes de RGB
Capas convolucionales
Capas completamente conectadas

FIG. 9A

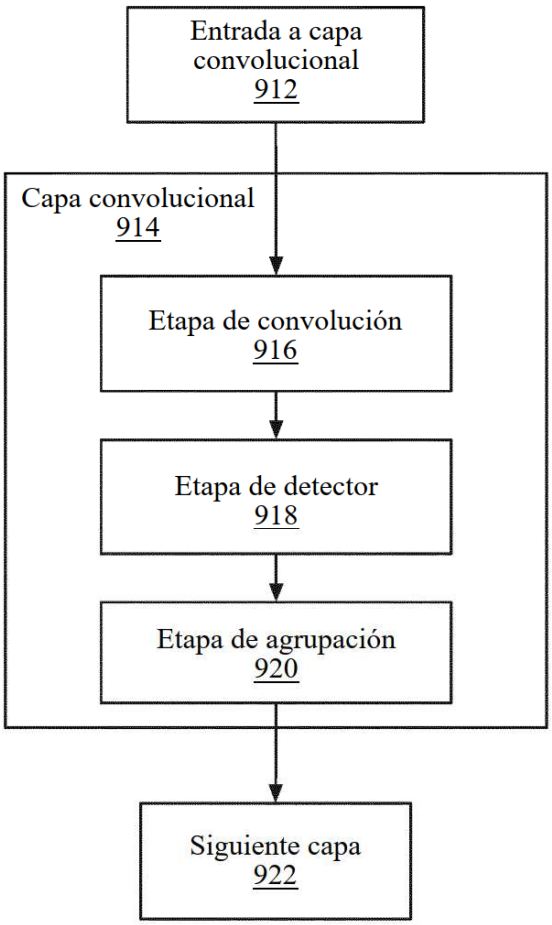

Entrada a capa convolucional 912
Capa convolucional 914
Etapa de convolución 916
Etapa de detector 918
Etapa de agrupación 920
Siguiente capa 922

FIG. 9B

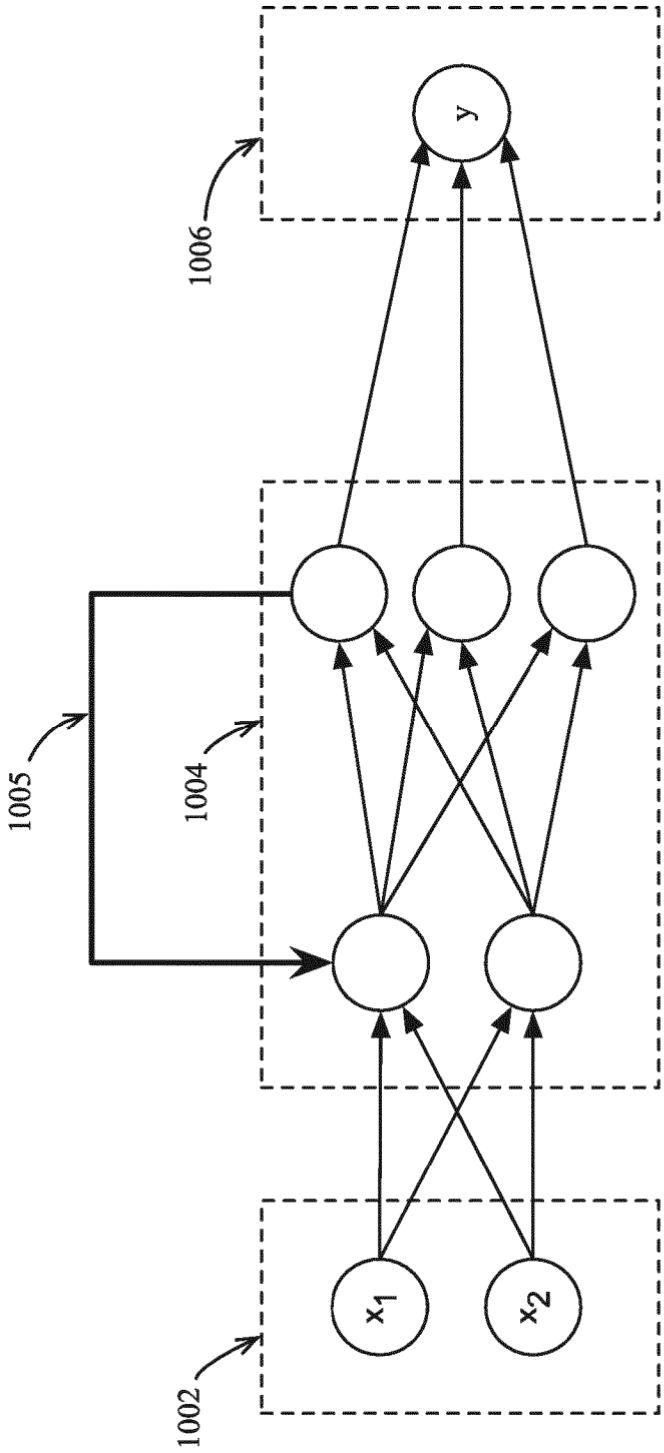


FIG. 10

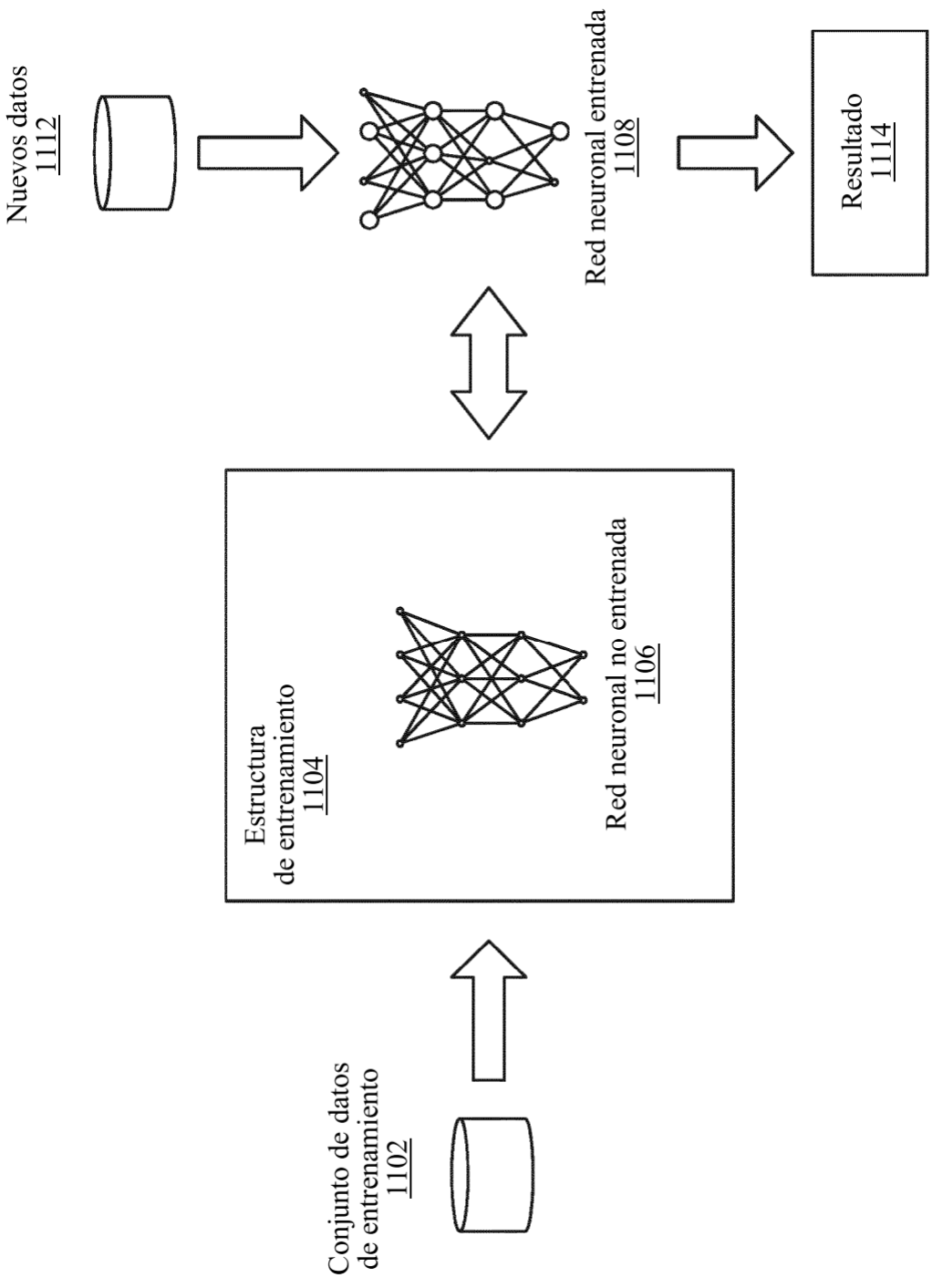

Nuevos datos
1112
Red neuronal entrenada
1108
Resultado
1114
Estructura
de entrenamiento
1104
Red neuronal no entrenada
1106
Conjunto de datos
de entrenamiento
1102

FIG. 11

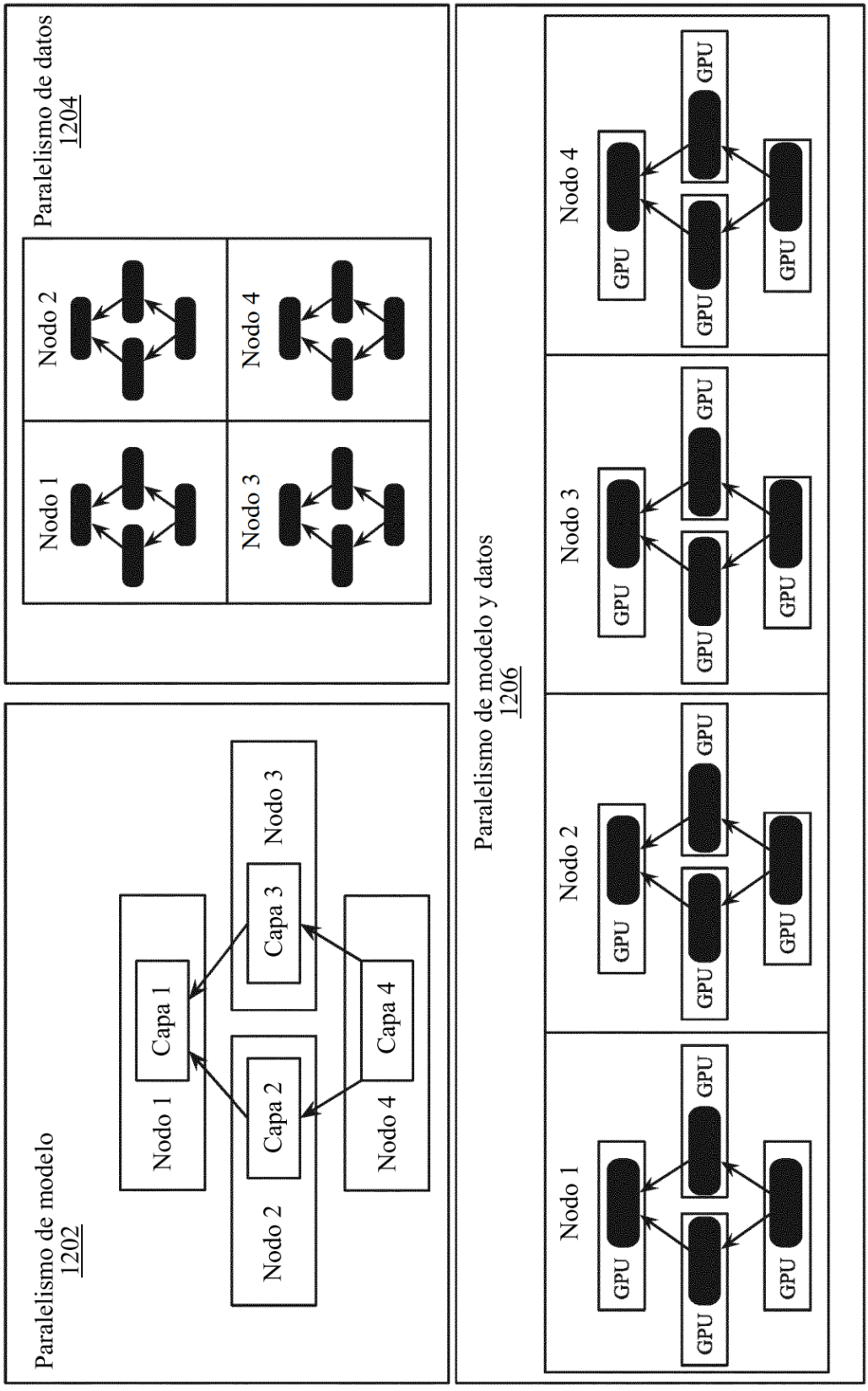

Paralelismo de modelo
1202
Nodo 1
Capa 1
Nodo 2
Capa 2
Nodo 3
Capa 3
Nodo 4
Capa 4
Paralelismo de datos
1204
Nodo 1
Nodo 2
Nodo 3
Nodo 4
Paralelismo de modelo y datos
1206
Nodo 1
Nodo 2
Nodo 3
Nodo 4
GPU

FIG. 12

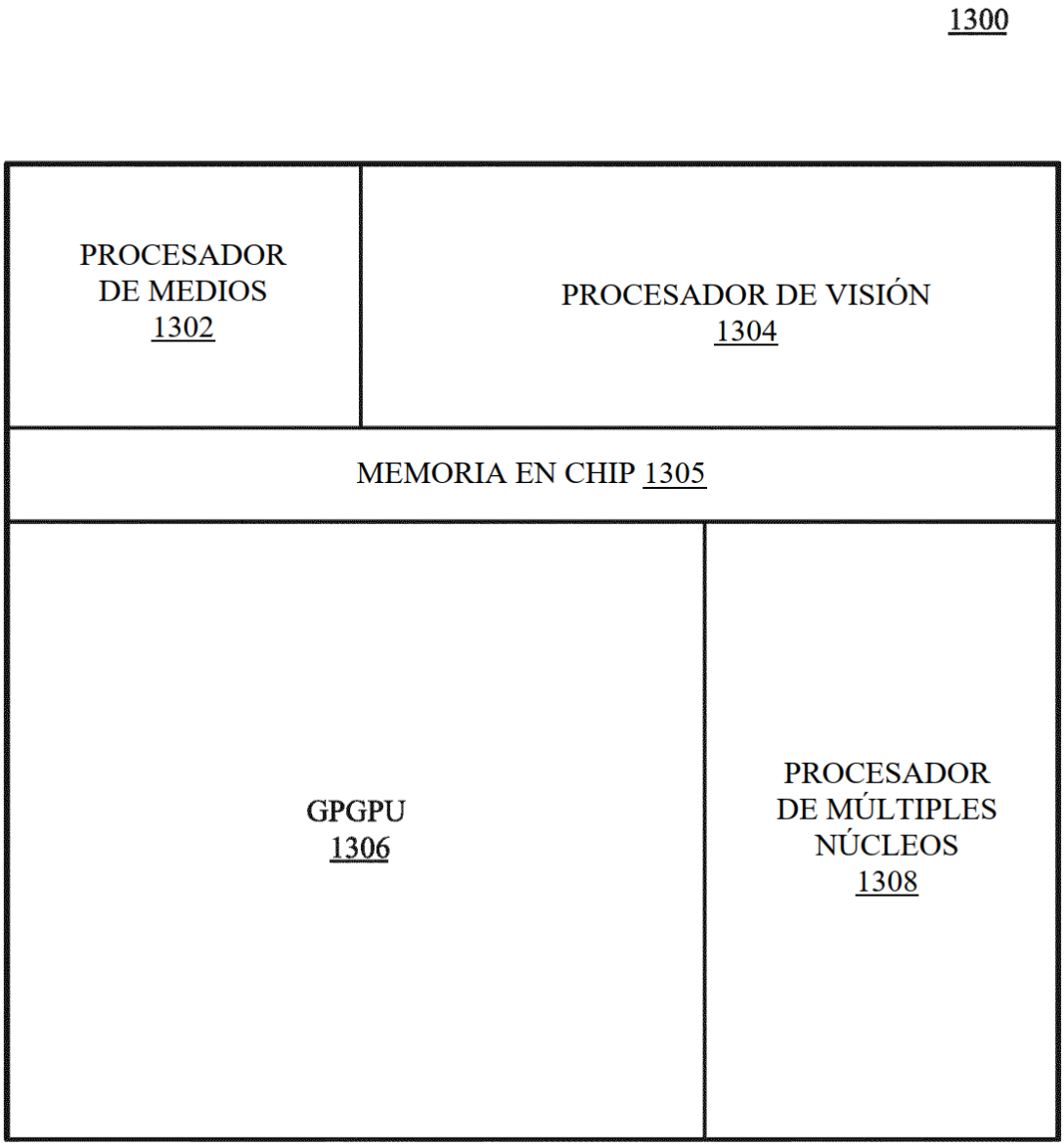

1300
PROCESADOR DE MEDIOS 1302
PROCESADOR DE VISIÓN 1304
MEMORIA EN CHIP 1305
GPGPU 1306
PROCESADOR DE MÚLTIPLES NÚCLEOS 1308

FIG. 13

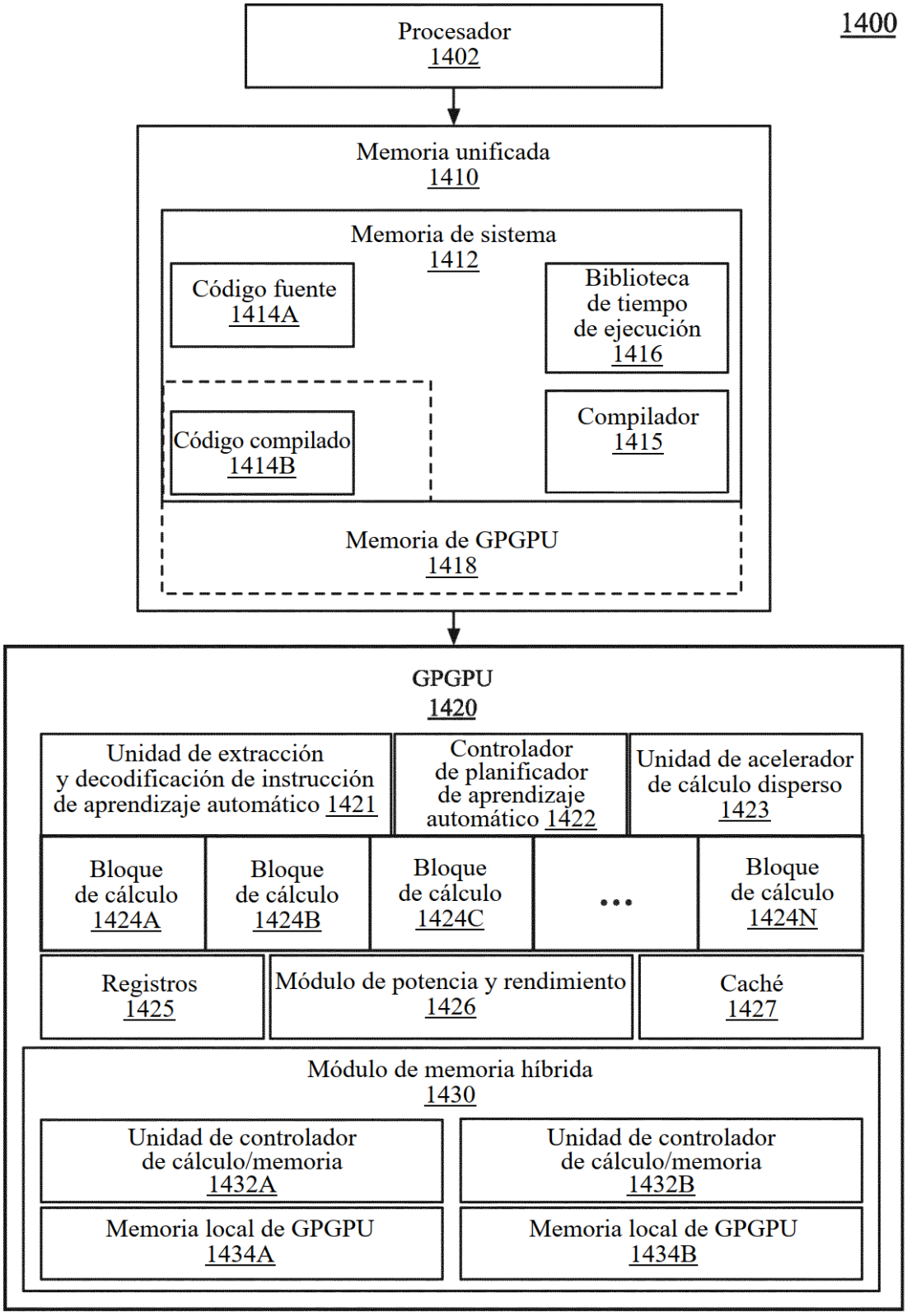

1400
Procesador
1402
Memoria unificada
1410
Memoria de sistema
1412
Código fuente
1414A
Biblioteca
de tiempo
de ejecución
1416
Código compilado
1414B
Compilador
1415
Memoria de GPGPU
1418
GPGPU
1420
Unidad de extracción
y decodificación de instrucción
de aprendizaje automático 1421
Controlador
de planificador
de aprendizaje
automático 1422
Unidad de acelerador
de cálculo disperso
1423
Bloque
de cálculo
1424A
Bloque
de cálculo
1424B
Bloque
de cálculo
1424C
...
Bloque
de cálculo
1424N
Registros
1425
Módulo de potencia y rendimiento
1426
Caché
1427
Módulo de memoria híbrida
1430
Unidad de controlador
de cálculo/memoria
1432A
Unidad de controlador
de cálculo/memoria
1432B
Memoria local de GPGPU
1434A
Memoria local de GPGPU
1434B

FIG. 14

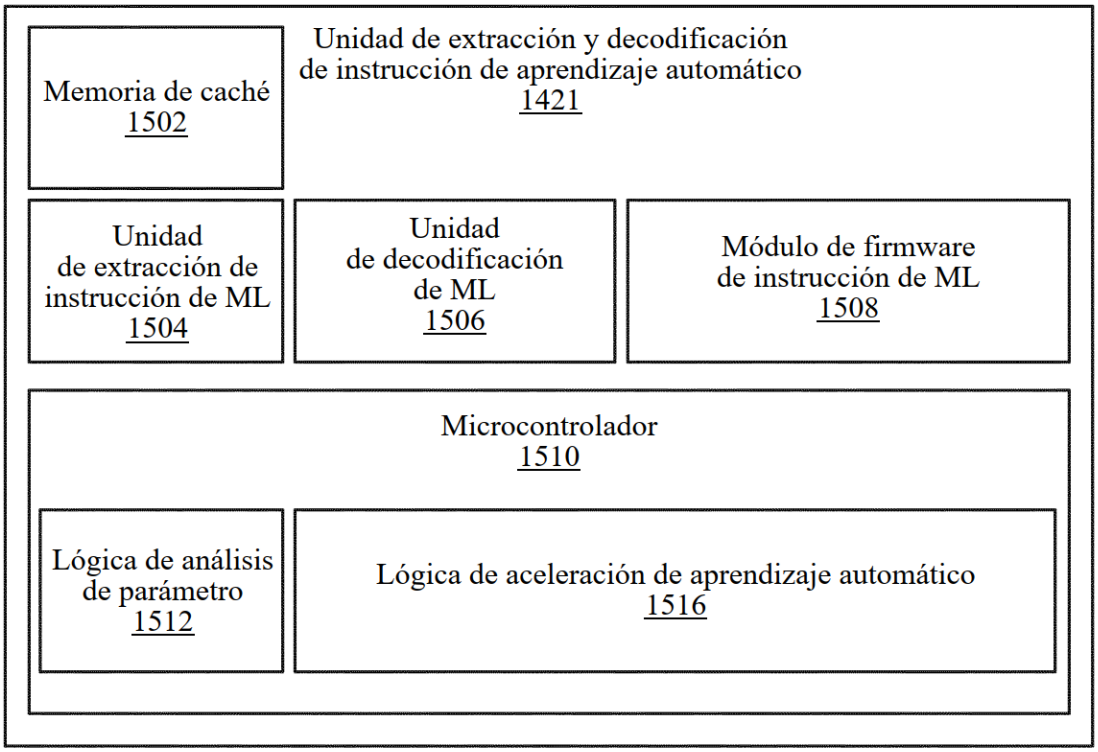


FIG. 15A

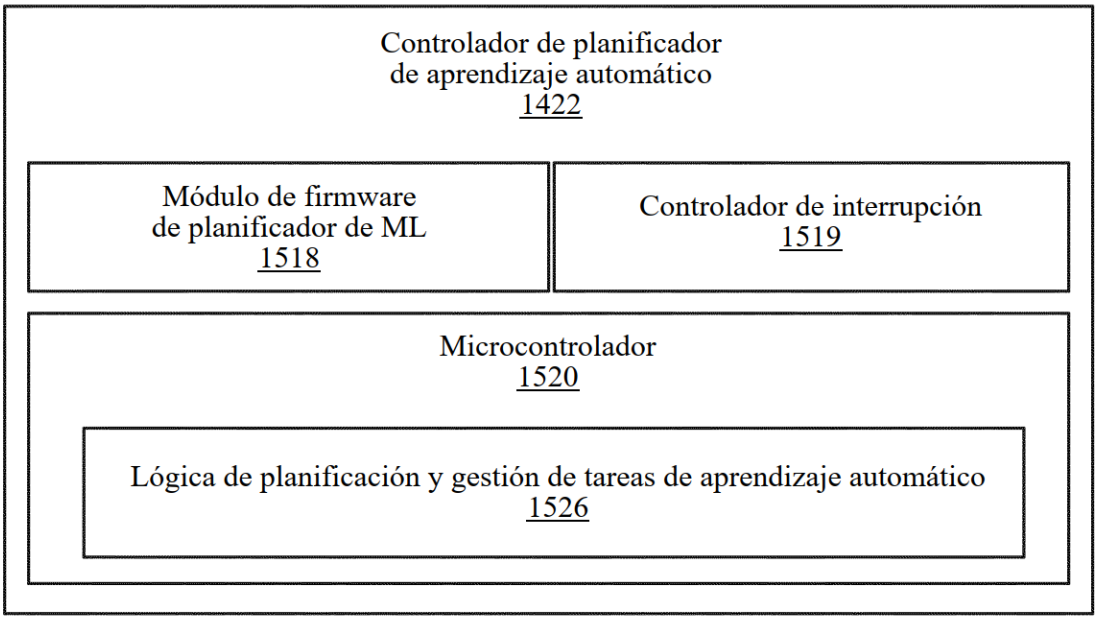


FIG. 15B

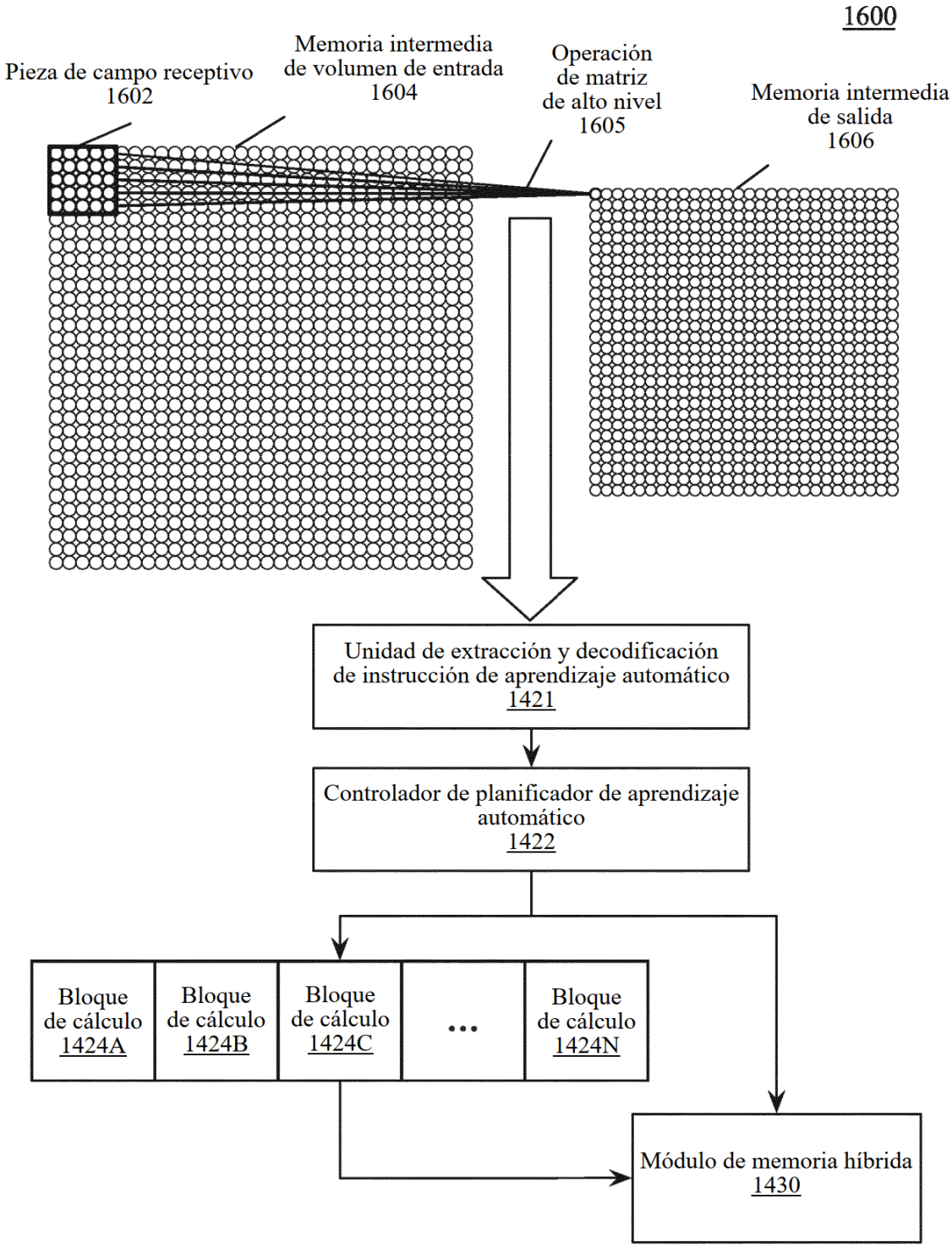

1600
Pieza de campo receptivo
1602
Memoria intermedia
de volumen de entrada
1604
Operación
de matriz
de alto nivel
1605
Memoria intermedia
de salida
1606
Unidad de extracción y decodificación
de instrucción de aprendizaje automático
1421
Controlador de planificador de aprendizaje
automático
1422
Bloque
de cálculo
1424A
Bloque
de cálculo
1424B
Bloque
de cálculo
1424C
...
Bloque
de cálculo
1424N
Módulo de memoria híbrida
1430

FIG. 16

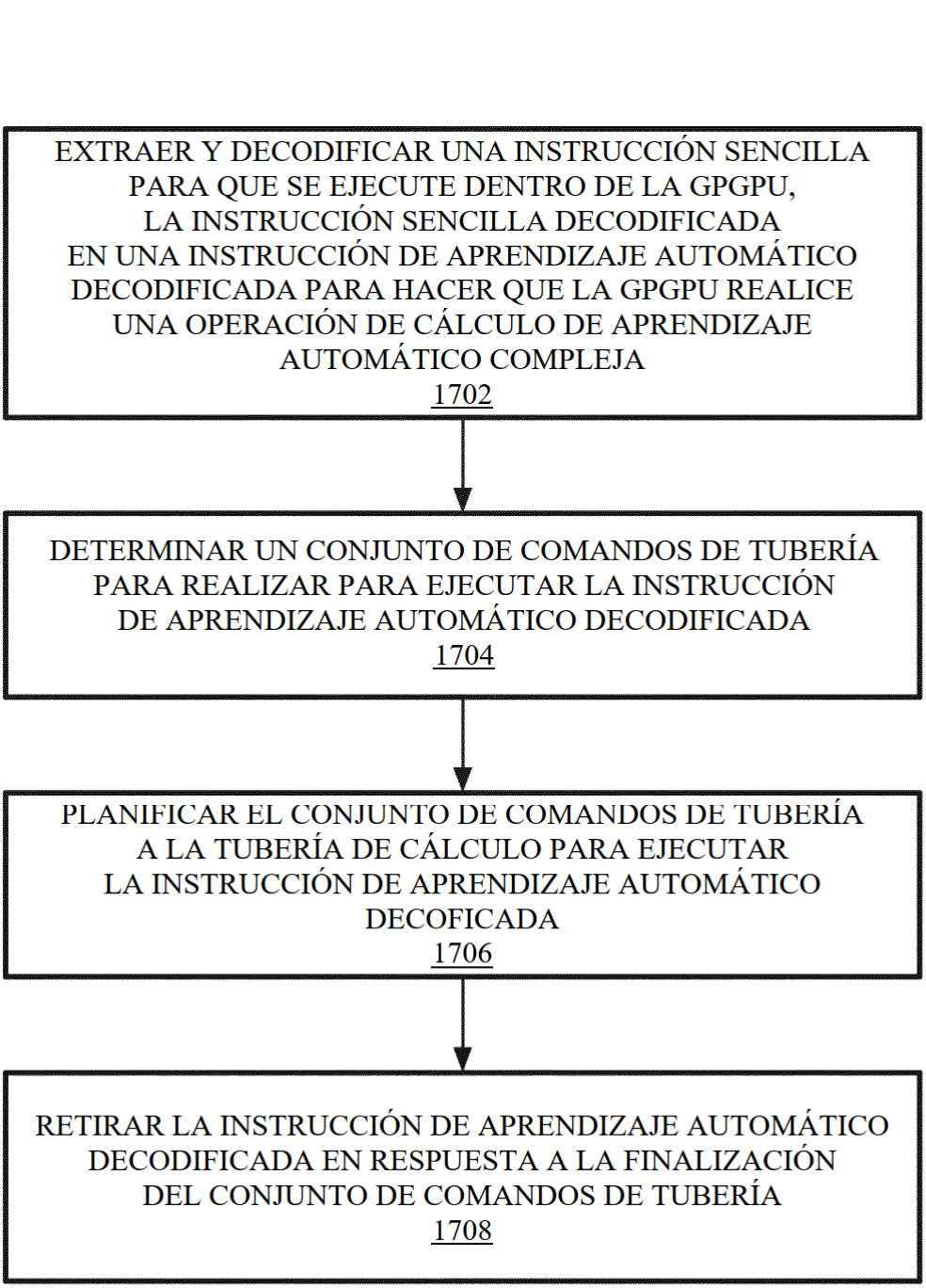

1700
EXTRAER Y DECODIFICAR UNA INSTRUCCIÓN SENCILLA PARA QUE SE EJECUTE DENTRO DE LA GPGPU, LA INSTRUCCIÓN SENCILLA DECODIFICADA EN UNA INSTRUCCIÓN DE APRENDIZAJE AUTOMÁTICO DECODIFICADA PARA HACER QUE LA GPGPU REALICE UNA OPERACIÓN DE CÁLCULO DE APRENDIZAJE AUTOMÁTICO COMPLEJA
1702
DETERMINAR UN CONJUNTO DE COMANDOS DE TUBERÍA PARA REALIZAR PARA EJECUTAR LA INSTRUCCIÓN DE APRENDIZAJE AUTOMÁTICO DECODIFICADA
1704
PLANIFICAR EL CONJUNTO DE COMANDOS DE TUBERÍA A LA TUBERÍA DE CÁLCULO PARA EJECUTAR LA INSTRUCCIÓN DE APRENDIZAJE AUTOMÁTICO DECOFICADA
1706
RETIRAR LA INSTRUCCIÓN DE APRENDIZAJE AUTOMÁTICO DECODIFICADA EN RESPUESTA A LA FINALIZACIÓN DEL CONJUNTO DE COMANDOS DE TUBERÍA
1708

FIG. 17

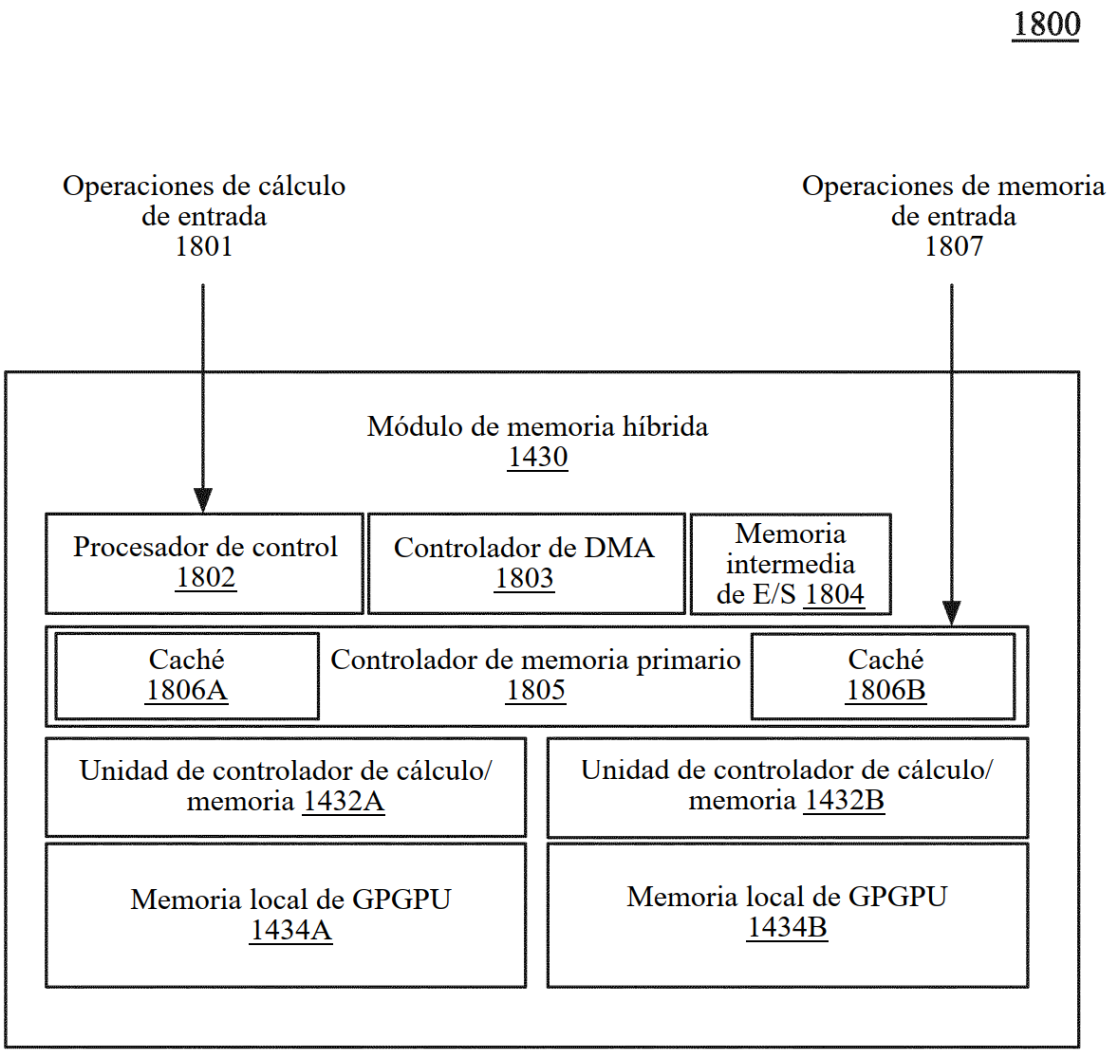

1800
Operaciones de cálculo de entrada 1801
Operaciones de memoria de entrada 1807
Módulo de memoria híbrida 1430
Procesador de control 1802
Controlador de DMA 1803
Memoria intermedia de E/S 1804
Caché 1806A
Controlador de memoria primario 1805
Caché 1806B
Unidad de controlador de cálculo/ memoria 1432A
Unidad de controlador de cálculo/ memoria 1432B
Memoria local de GPGPU 1434A
Memoria local de GPGPU 1434B

FIG. 18

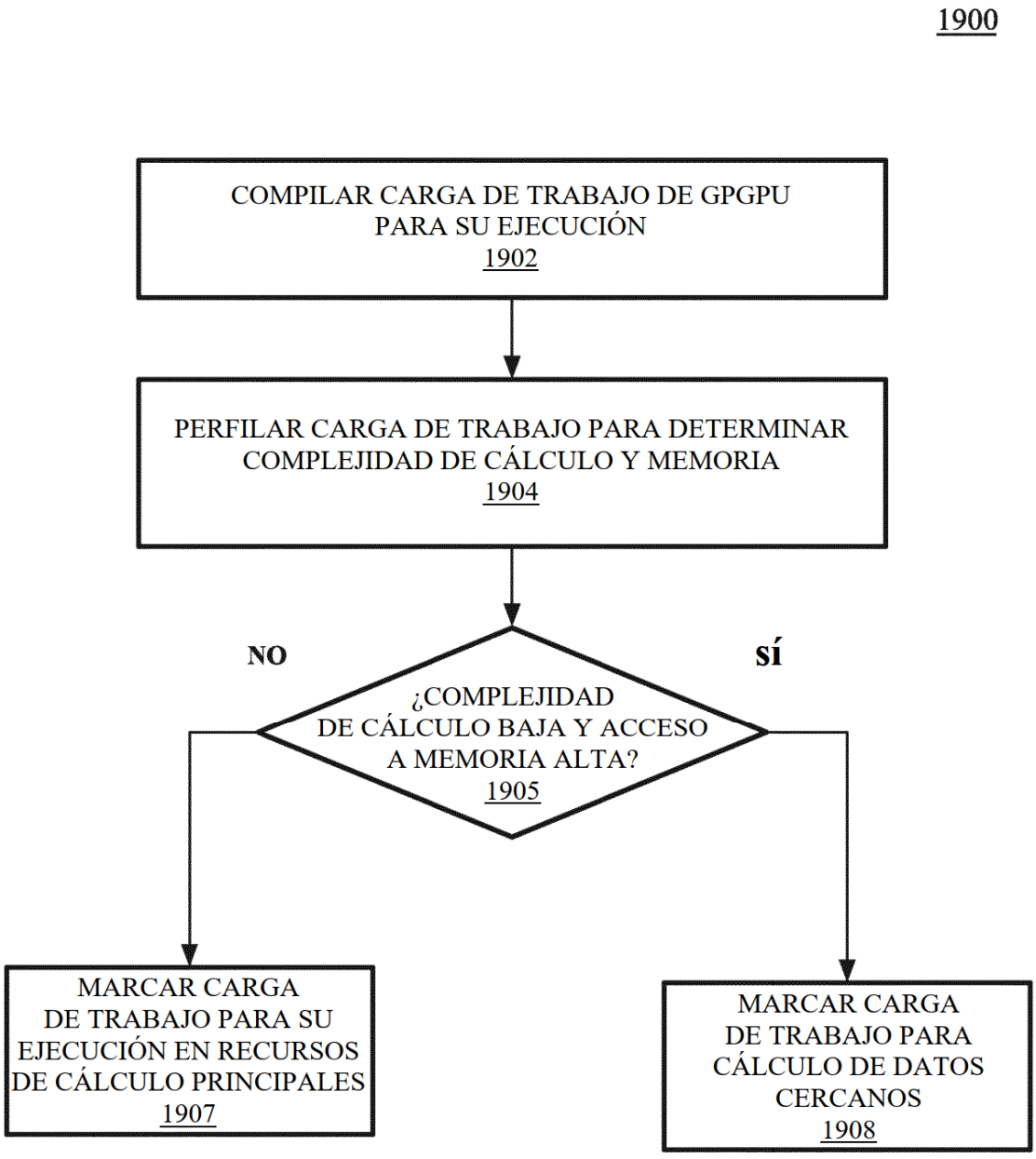

1900
COMPILAR CARGA DE TRABAJO DE GPGPU PARA SU EJECUCIÓN
1902
PERFILAR CARGA DE TRABAJO PARA DETERMINAR COMPLEJIDAD DE CÁLCULO Y MEMORIA
1904
NO
SÍ
¿COMPLEJIDAD DE CÁLCULO BAJA Y ACCESO A MEMORIA ALTA?
1905
MARCAR CARGA DE TRABAJO PARA SU EJECUCIÓN EN RECURSOS DE CÁLCULO PRINCIPALES
1907
MARCAR CARGA DE TRABAJO PARA CÁLCULO DE DATOS CERCANOS
1908

FIG. 19A

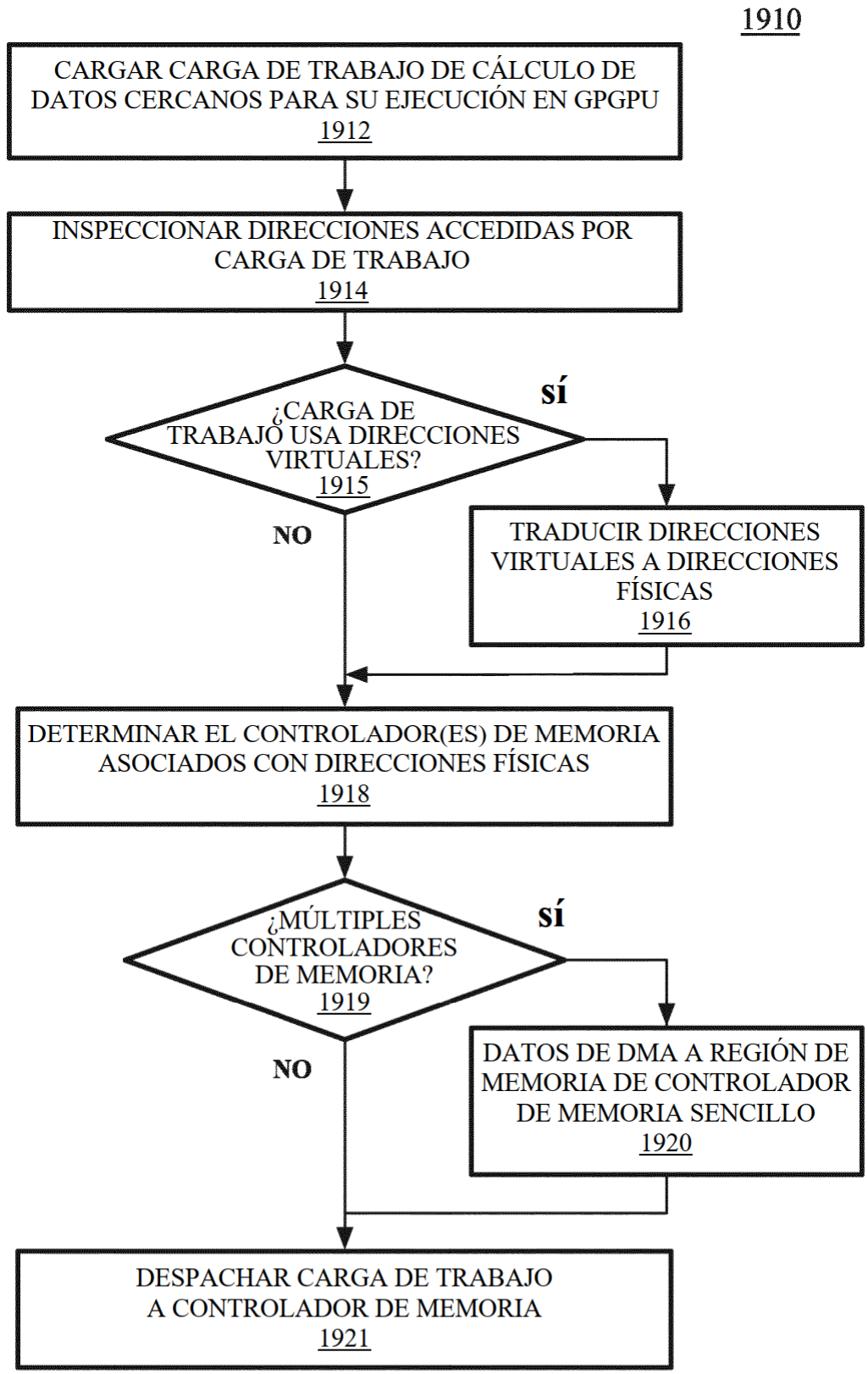

1910
CARGAR CARGA DE TRABAJO DE CÁLCULO DE DATOS CERCANOS PARA SU EJECUCIÓN EN GPGPU
1912
INSPECCIONAR DIRECCIONES ACCEDIDAS POR CARGA DE TRABAJO
1914
¿CARGA DE TRABAJO USA DIRECCIONES VIRTUALES?
1915
SÍ
NO
TRADUCIR DIRECCIONES VIRTUALES A DIRECCIONES FÍSICAS
1916
DETERMINAR EL CONTROLADOR(ES) DE MEMORIA ASOCIADOS CON DIRECCIONES FÍSICAS
1918
¿MÚLTIPLES CONTROLADORES DE MEMORIA?
1919
SÍ
NO
DATOS DE DMA A REGIÓN DE MEMORIA DE CONTROLADOR DE MEMORIA SENCILLO
1920
DESPACHAR CARGA DE TRABAJO A CONTROLADOR DE MEMORIA
1921

FIG. 19B

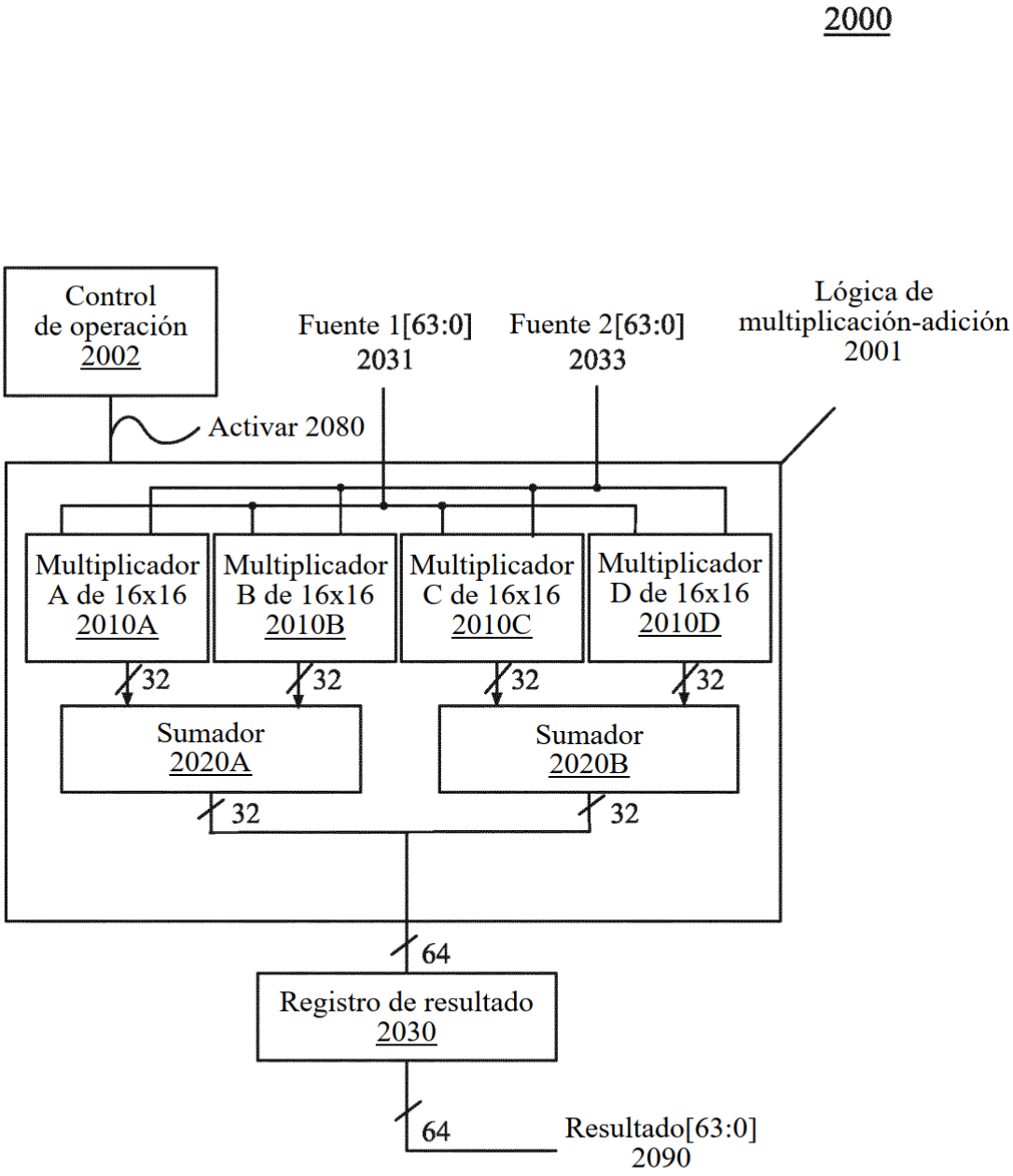

2000
Control de operación 2002
Fuente 1[63:0] 2031
Fuente 2[63:0] 2033
Lógica de multiplicación-adición 2001
Activar 2080
Multiplicador A de 16x16 2010A
Multiplicador B de 16x16 2010B
Multiplicador C de 16x16 2010C
Multiplicador D de 16x16 2010D
32
32
32
32
Sumador 2020A
Sumador 2020B
32
32
64
Registro de resultado 2030
64
Resultado[63:0] 2090

FIG. 20

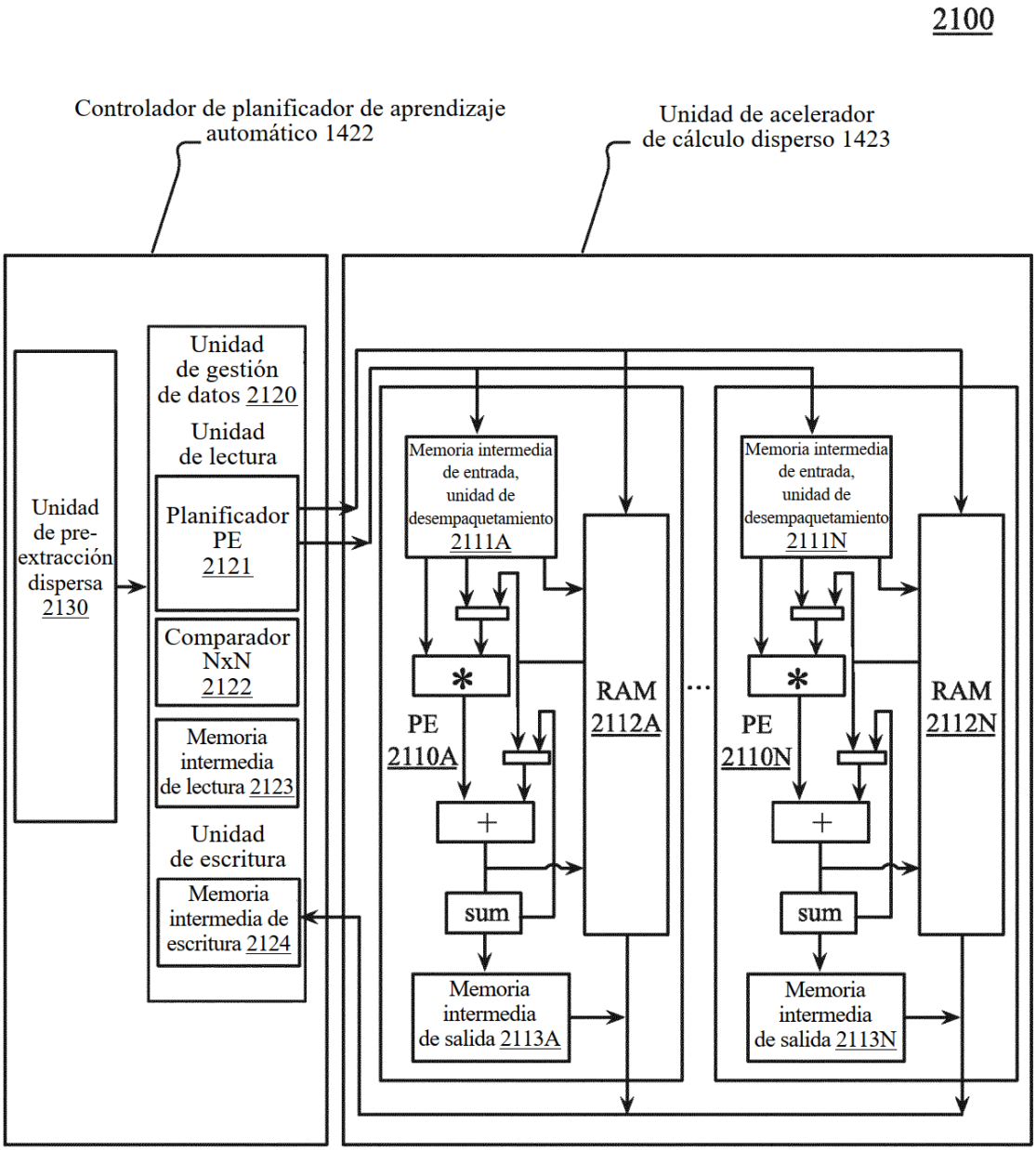

2100
Controlador de planificador de aprendizaje automático 1422
Unidad de acelerador de cálculo disperso 1423
Unidad de pre-extracción dispersa 2130
Unidad de gestión de datos 2120
Unidad de lectura
Planificador PE 2121
Comparador NxN 2122
Memoria intermedia de lectura 2123
Unidad de escritura
Memoria intermedia de escritura 2124
Memoria intermedia de entrada, unidad de desempaquetamiento 2111A
Memoria intermedia de entrada, unidad de desempaquetamiento 2111N
PE 2110A
PE 2110N
RAM 2112A
RAM 2112N
sum
Memoria intermedia de salida 2113A
Memoria intermedia de salida 2113N

FIG. 21

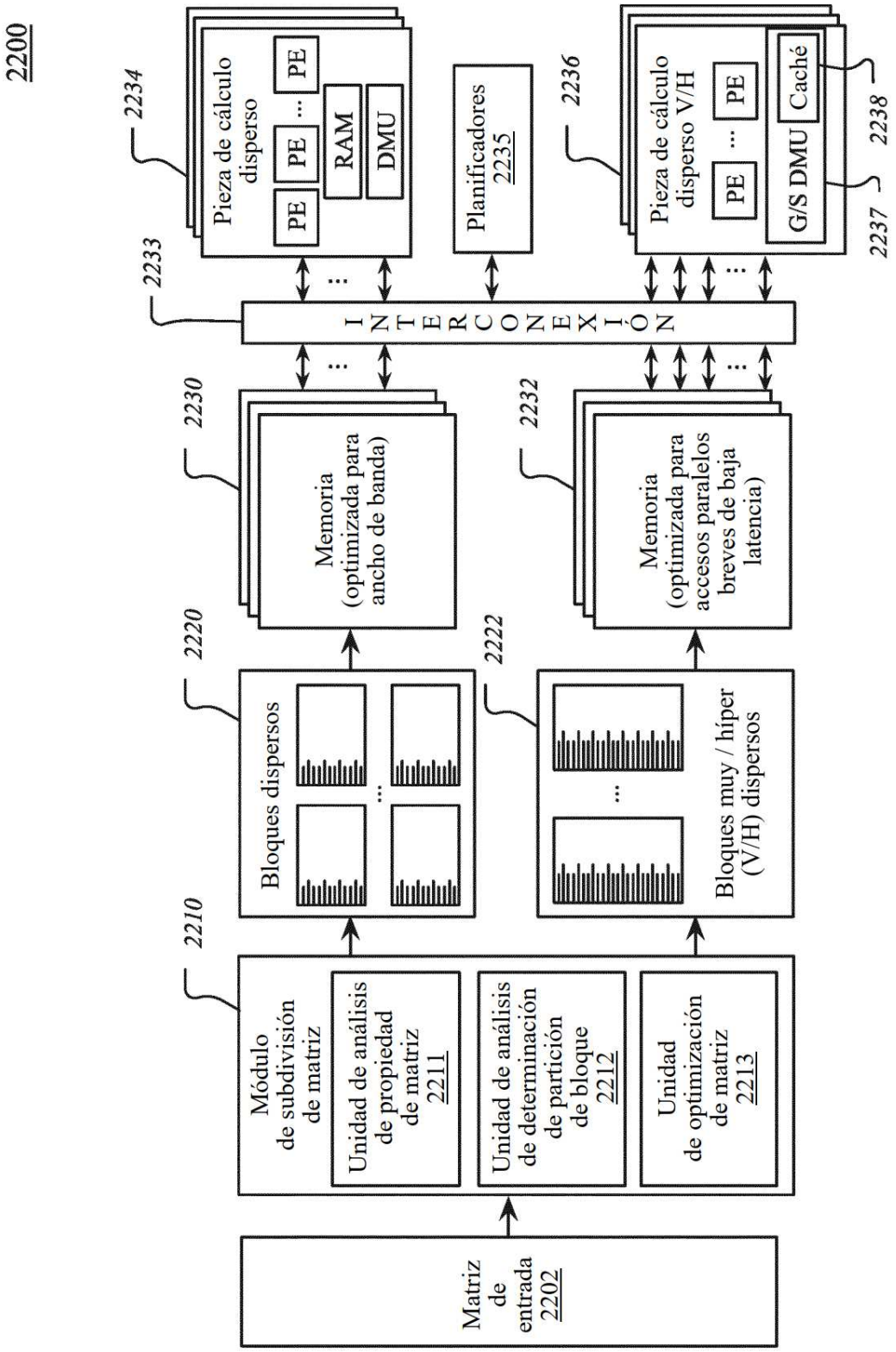

2200
Matriz de entrada 2202
2210
Módulo de subdivisión de matriz
Unidad de análisis de propiedad de matriz 2211
Unidad de análisis de determinación de partición de bloque 2212
Unidad de optimización de matriz 2213
2220
Bloques dispersos
2222
Bloques muy / híper (V/H) dispersos
2230
Memoria (optimizada para ancho de banda)
2232
Memoria (optimizada para accesos paralelos breves de baja latencia)
2233
INTERCONEXIÓN
2234
Pieza de cálculo disperso
PE
PE
PE
RAM
DMU
Planificadores 2235
2236
Pieza de cálculo disperso V/H
PE
PE
G/S DMU
Caché
2237
2238

FIG. 22

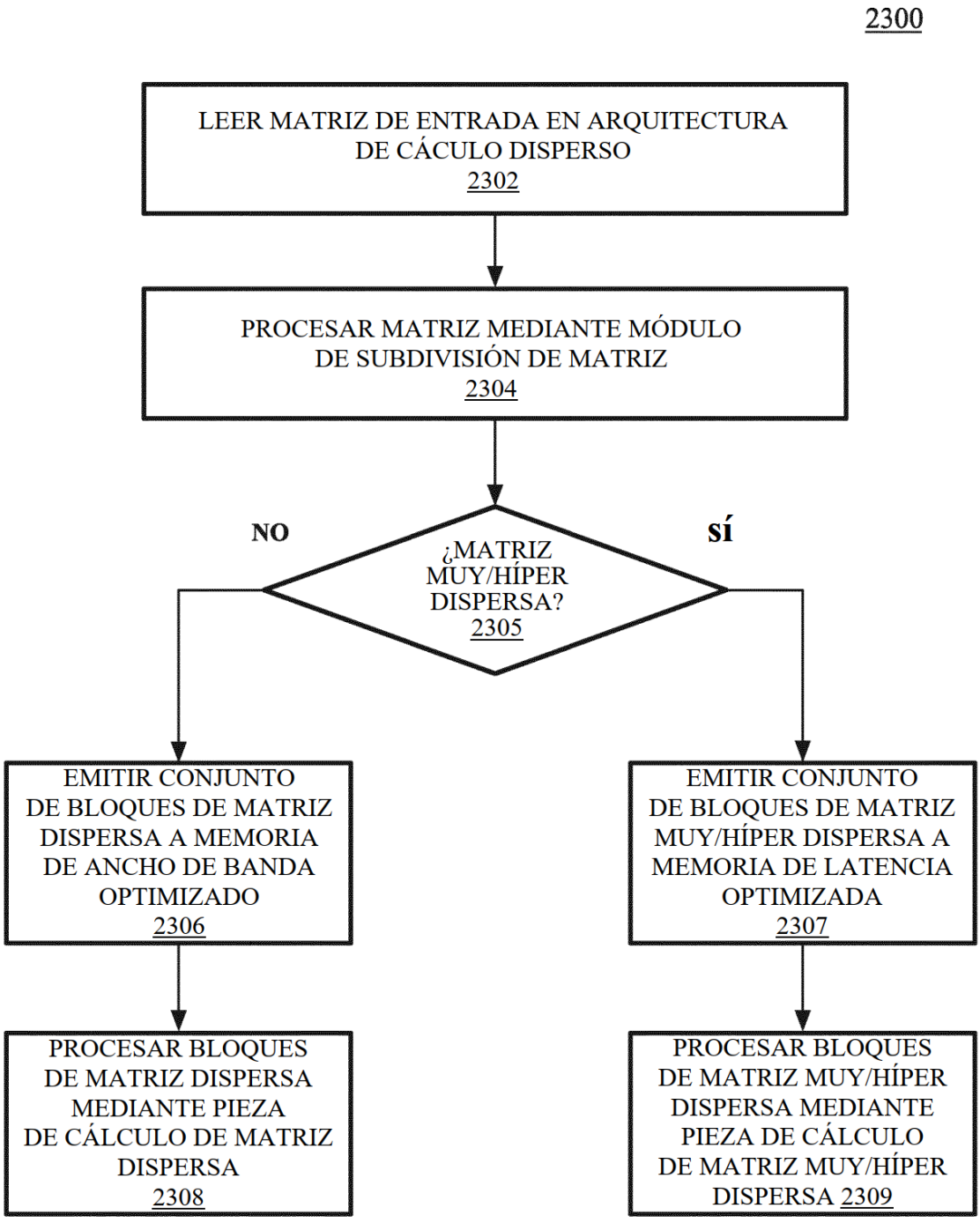

2300
LEER MATRIZ DE ENTRADA EN ARQUITECTURA DE CÁCULO DISPERSO
2302
PROCESAR MATRIZ MEDIANTE MÓDULO DE SUBDIVISIÓN DE MATRIZ
2304
NO
¿MATRIZ MUY/HÍPER DISPERSA?
2305
SÍ
EMITIR CONJUNTO DE BLOQUES DE MATRIZ DISPERSA A MEMORIA DE ANCHO DE BANDA OPTIMIZADO
2306
EMITIR CONJUNTO DE BLOQUES DE MATRIZ MUY/HÍPER DISPERSA A MEMORIA DE LATENCIA OPTIMIZADA
2307
PROCESAR BLOQUES DE MATRIZ DISPERSA MEDIANTE PIEZA DE CÁLCULO DE MATRIZ DISPERSA
2308
PROCESAR BLOQUES DE MATRIZ MUY/HÍPER DISPERSA MEDIANTE PIEZA DE CÁLCULO DE MATRIZ MUY/HÍPER DISPERSA 2309

FIG. 23A

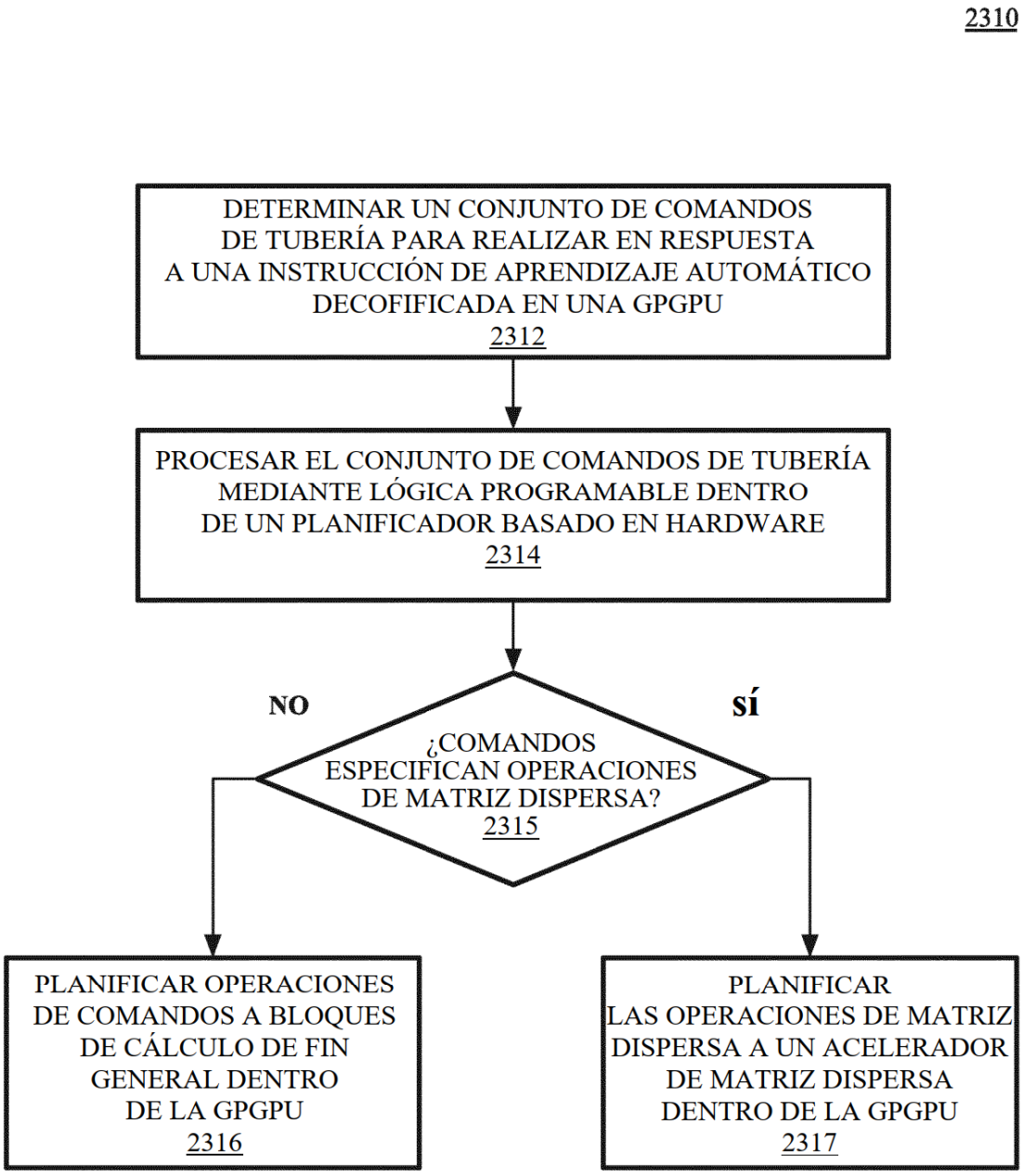

2310
DETERMINAR UN CONJUNTO DE COMANDOS DE TUBERÍA PARA REALIZAR EN RESPUESTA A UNA INSTRUCCIÓN DE APRENDIZAJE AUTOMÁTICO DECOFIFICADA EN UNA GPGPU
2312
PROCESAR EL CONJUNTO DE COMANDOS DE TUBERÍA MEDIANTE LÓGICA PROGRAMABLE DENTRO DE UN PLANIFICADOR BASADO EN HARDWARE
2314
¿COMANDOS ESPECIFICAN OPERACIONES DE MATRIZ DISPERSA?
2315
NO
SÍ
PLANIFICAR OPERACIONES DE COMANDOS A BLOQUES DE CÁLCULO DE FIN GENERAL DENTRO DE LA GPGPU
2316
PLANIFICAR LAS OPERACIONES DE MATRIZ DISPERSA A UN ACELERADOR DE MATRIZ DISPERSA DENTRO DE LA GPGPU
2317

FIG. 23B

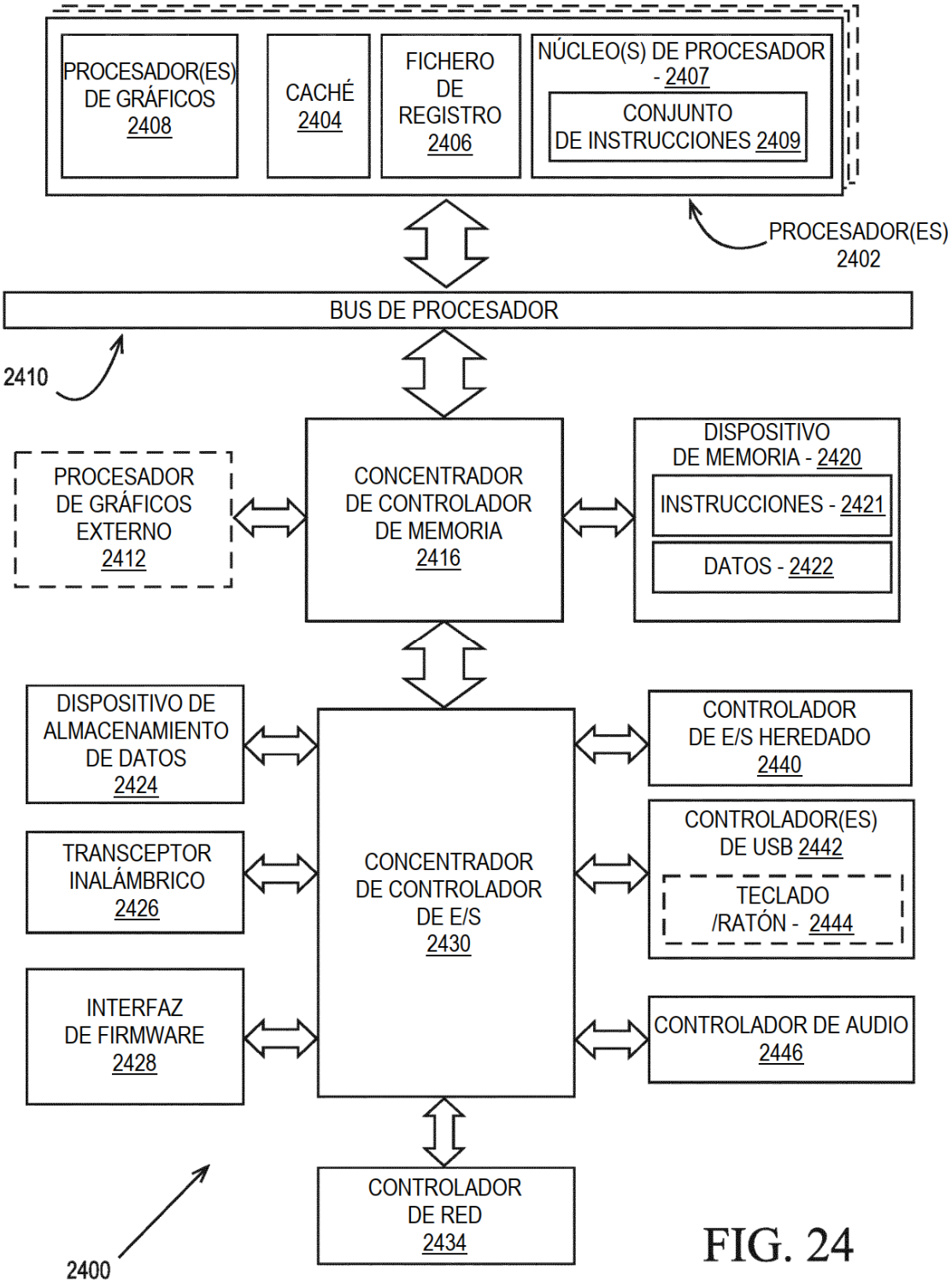

PROCESADOR(ES) DE GRÁFICOS 2408
CACHÉ 2404
FICHERO DE REGISTRO 2406
NÚCLEO(S) DE PROCESADOR - 2407
CONJUNTO DE INSTRUCCIONES 2409
PROCESADOR(ES) 2402
BUS DE PROCESADOR
2410
PROCESADOR DE GRÁFICOS EXTERNO 2412
CONCENTRADOR DE CONTROLADOR DE MEMORIA 2416
DISPOSITIVO DE MEMORIA - 2420
INSTRUCCIONES - 2421
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(ES) DE USB 2442
TECLADO /RATÓN - 2444
CONTROLADOR DE AUDIO 2446
CONTROLADOR DE RED 2434
2400

FIG. 24

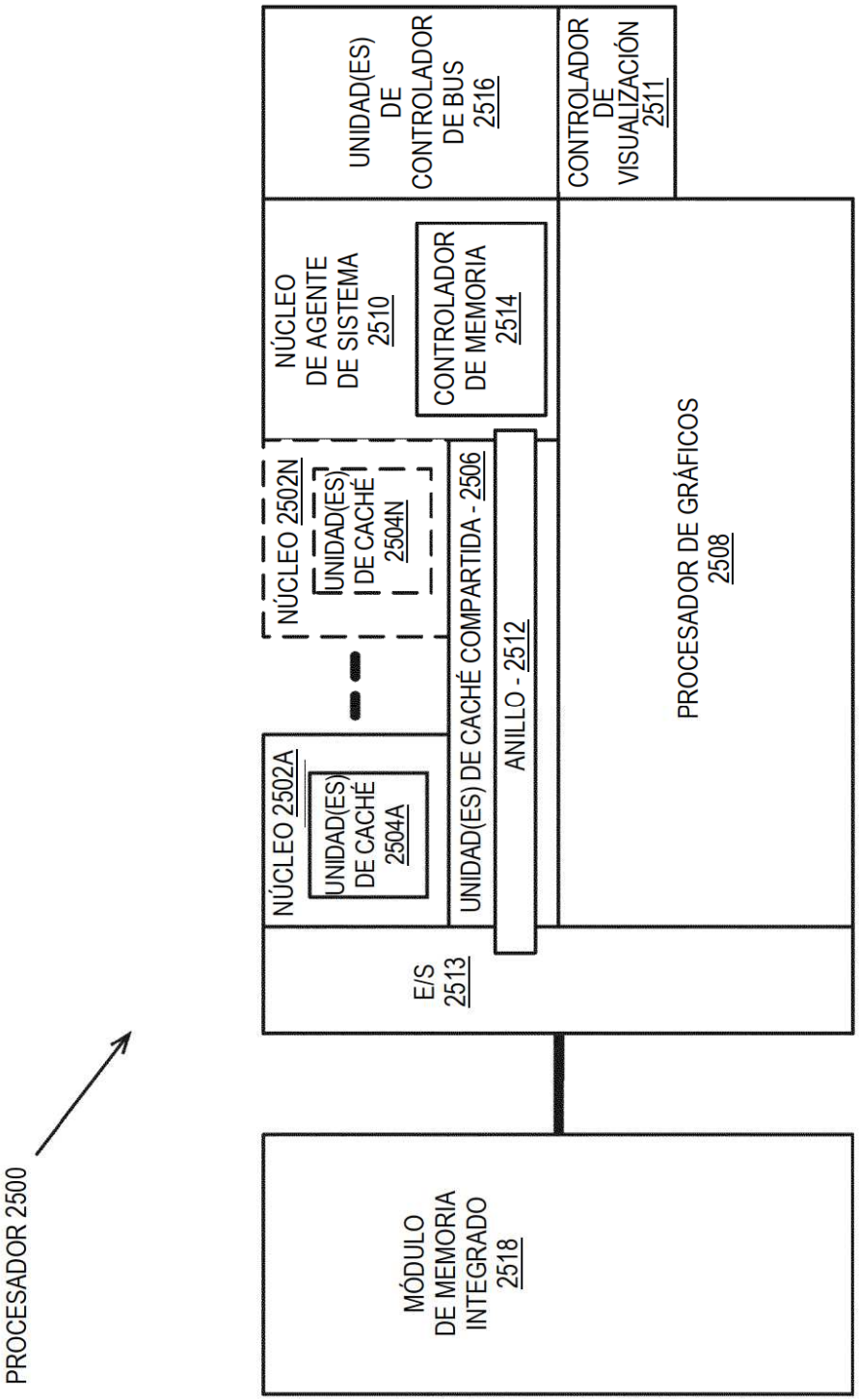

PROCESADOR 2500
MÓDULO DE MEMORIA INTEGRADO 2518
E/S 2513
NÚCLEO 2502A
UNIDAD(ES) DE CACHÉ 2504A
NÚCLEO 2502N
UNIDAD(ES) DE CACHÉ 2504N
UNIDAD(ES) DE CACHÉ COMPARTIDA - 2506
ANILLO - 2512
NÚCLEO DE AGENTE DE SISTEMA 2510
CONTROLADOR DE MEMORIA 2514
PROCESADOR DE GRÁFICOS 2508
UNIDAD(ES) DE CONTROLADOR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

FIG. 25

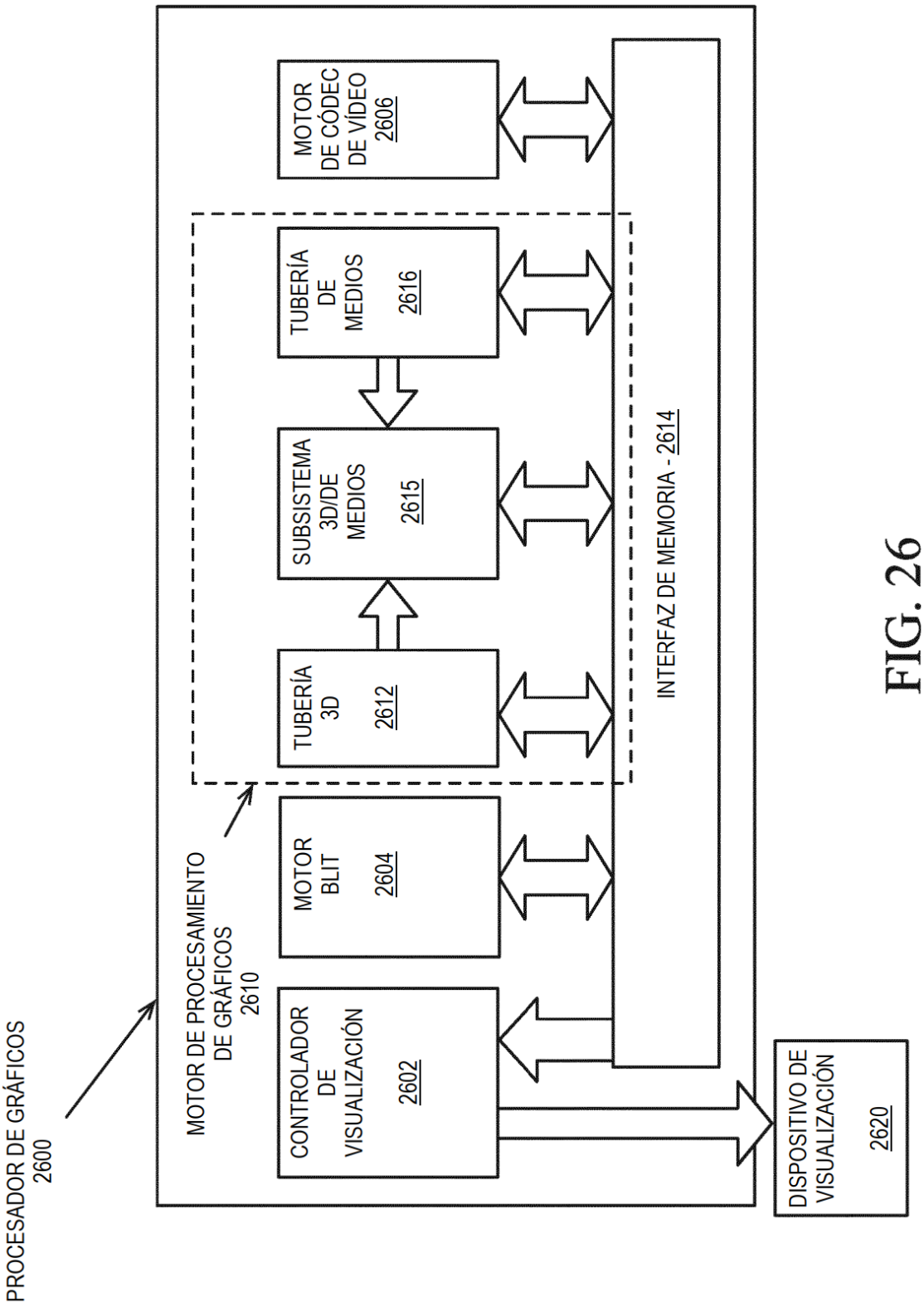

PROCESADOR DE GRÁFICOS 2600
MOTOR DE PROCESAMIENTO DE GRÁFICOS 2610
CONTROLADOR DE VISUALIZACIÓN 2602
MOTOR BLIT 2604
TUBERÍA 3D 2612
SUBSISTEMA 3D/DE MEDIOS 2615
TUBERÍA DE MEDIOS 2616
MOTOR DE CÓDEC DE VÍDEO 2606
INTERFAZ DE MEMORIA - 2614
DISPOSITIVO DE VISUALIZACIÓN 2620

FIG. 26

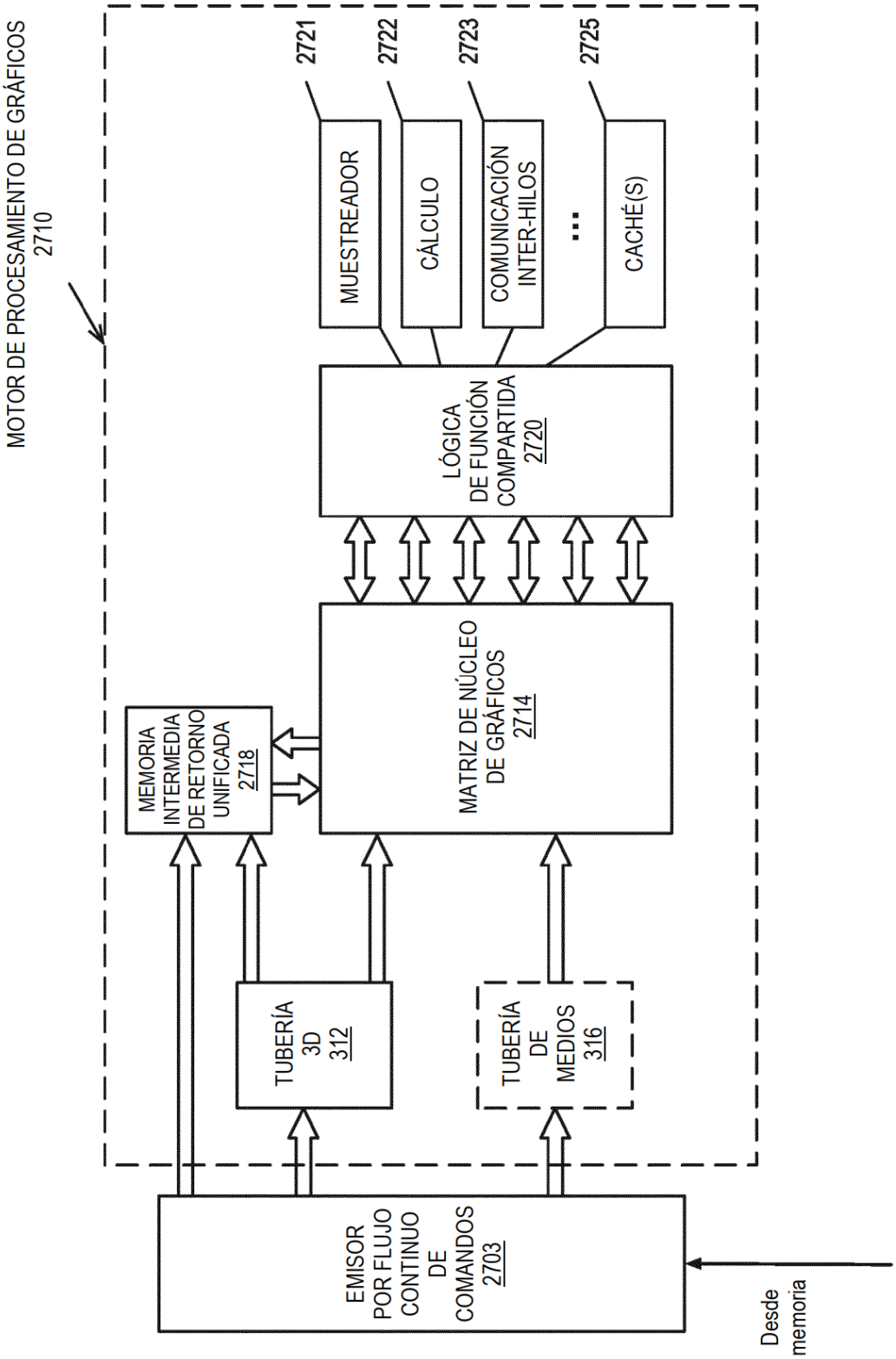

MOTOR DE PROCESAMIENTO DE GRÁFICOS
2710
2721
2722
2723
2725
MUESTREADOR
CÁLCULO
COMUNICACIÓN INTER-HILOS
...
CACHÉ(S)
LÓGICA DE FUNCIÓN COMPARTIDA
2720
MATRIZ DE NÚCLEO DE GRÁFICOS
2714
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
2718
TUBERÍA 3D
312
TUBERÍA DE MEDIOS
316
EMISOR POR FLUJO CONTINUO DE COMANDOS
2703
Desde memoria

FIG. 27

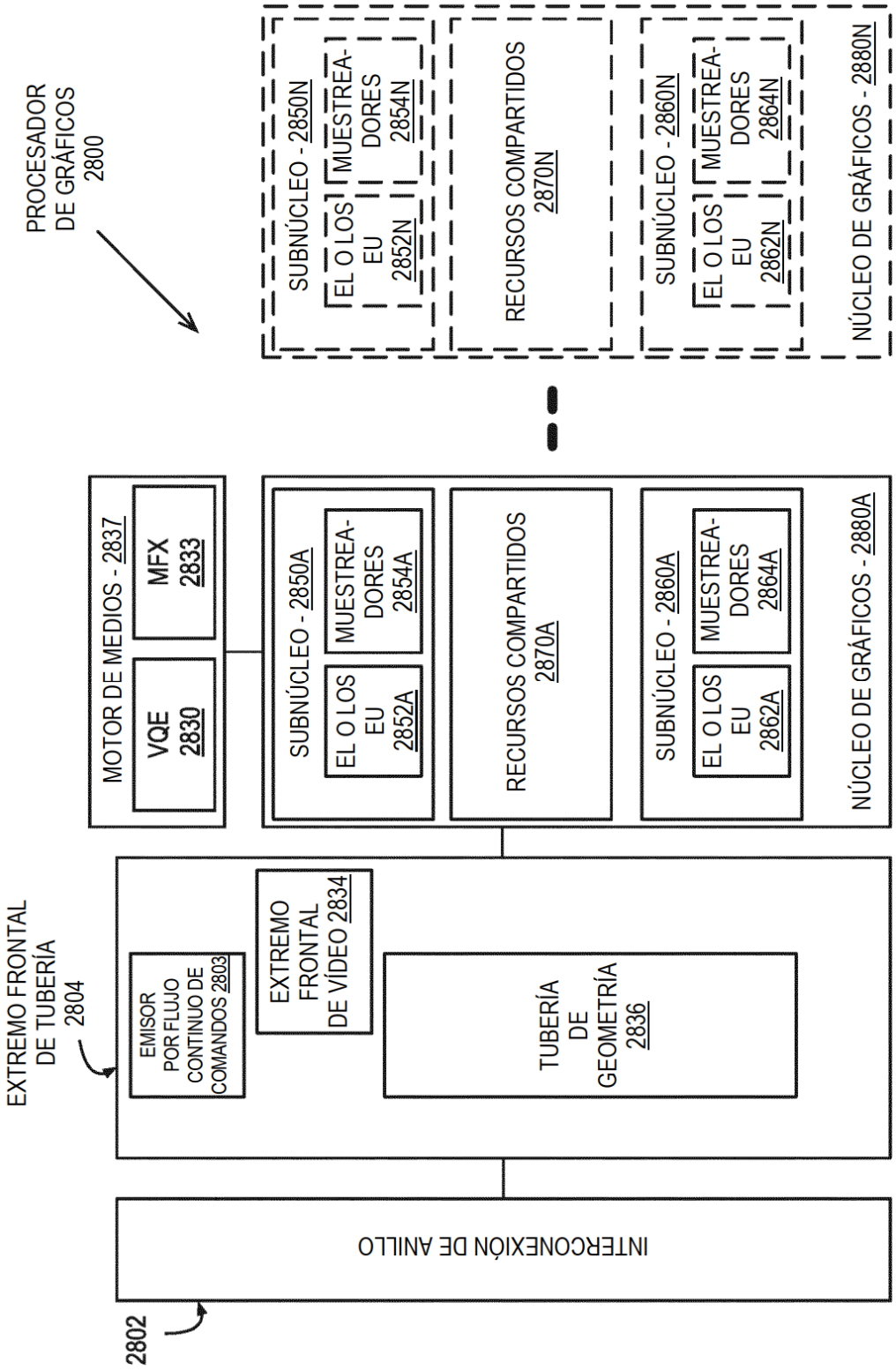

PROCESADOR DE GRÁFICOS 2800
SUBNÚCLEO - 2850N
EL O LOS EU 2852N
MUESTREA-DORES 2854N
RECURSOS COMPARTIDOS 2870N
SUBNÚCLEO - 2860N
EL O LOS EU 2862N
MUESTREA-DORES 2864N
NÚCLEO DE GRÁFICOS - 2880N
MOTOR DE MEDIOS - 2837
MFX 2833
VQE 2830
SUBNÚCLEO - 2850A
EL O LOS EU 2852A
MUESTREA-DORES 2854A
RECURSOS COMPARTIDOS 2870A
SUBNÚCLEO - 2860A
EL O LOS EU 2862A
MUESTREA-DORES 2864A
NÚCLEO DE GRÁFICOS - 2880A
EXTREMO FRONTAL DE TUBERÍA 2804
EMISOR POR FLUJO CONTINUO DE COMANDOS 2803
EXTREMO FRONTAL DE VÍDEO 2834
TUBERÍA DE GEOMETRÍA 2836
INTERCONEXIÓN DE ANILLO
2802

FIG. 28

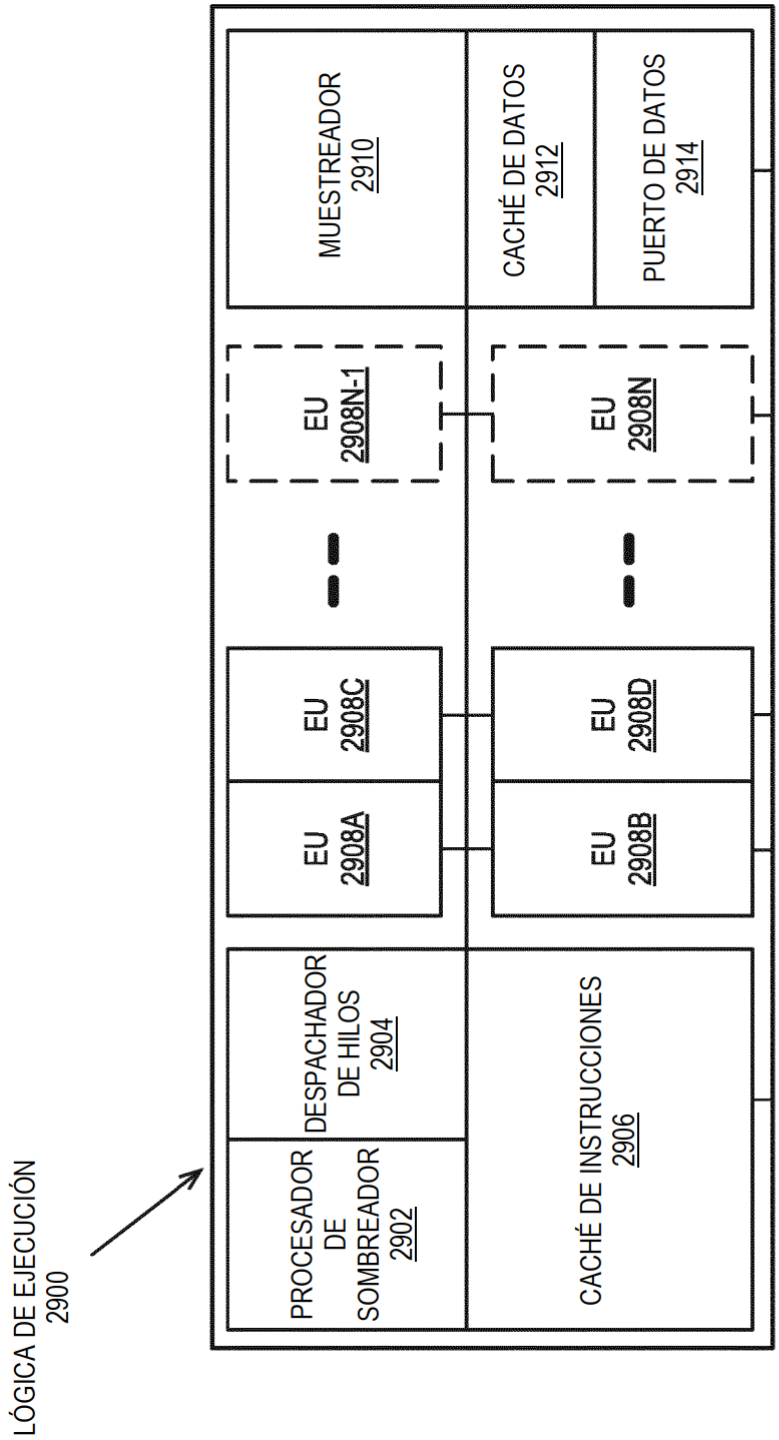

LÓGICA DE EJECUCIÓN
2900
PROCESADOR DE SOMBREADOR
2902
DESPACHADOR DE HILOS
2904
CACHÉ DE INSTRUCCIONES
2906
EU
2908A
EU
2908C
EU
2908B
EU
2908D
EU
2908N-1
EU
2908N
MUESTREADOR
2910
CACHÉ DE DATOS
2912
PUERTO DE DATOS
2914

FIG. 29

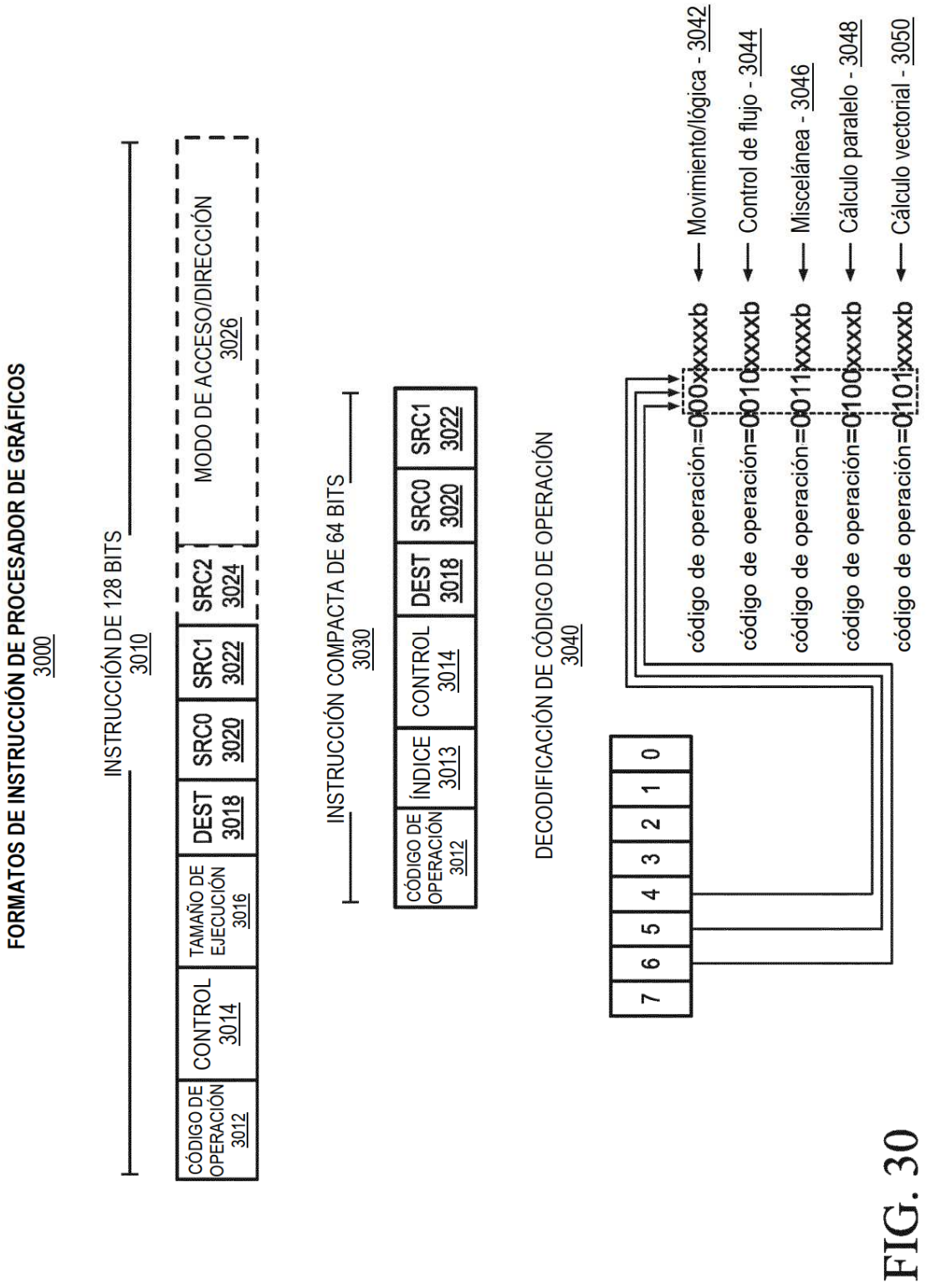

FORMATOS DE INSTRUCCIÓN DE PROCESADOR DE GRÁFICOS
3000
INSTRUCCIÓN DE 128 BITS
3010
CÓDIGO DE OPERACIÓN 3012
CONTROL 3014
TAMAÑO DE EJECUCIÓN 3016
DEST 3018
SRC0 3020
SRC1 3022
SRC2 3024
MODO DE ACCESO/DIRECCIÓN 3026
INSTRUCCIÓN COMPACTA DE 64 BITS
3030
CÓDIGO DE OPERACIÓN 3012
ÍNDICE 3013
CONTROL 3014
DEST 3018
SRC0 3020
SRC1 3022
DECODIFICACIÓN DE CÓDIGO DE OPERACIÓN
3040
7 6 5 4 3 2 1 0
código de operación=0000xxxxb
código de operación=0010xxxxb
código de operación=0011xxxxb
código de operación=0100xxxxb
código de operación=0101xxxxb
Movimiento/lógica - 3042
Control de flujo - 3044
Miscelánea - 3046
Cálculo paralelo - 3048
Cálculo vectorial - 3050

FIG. 30

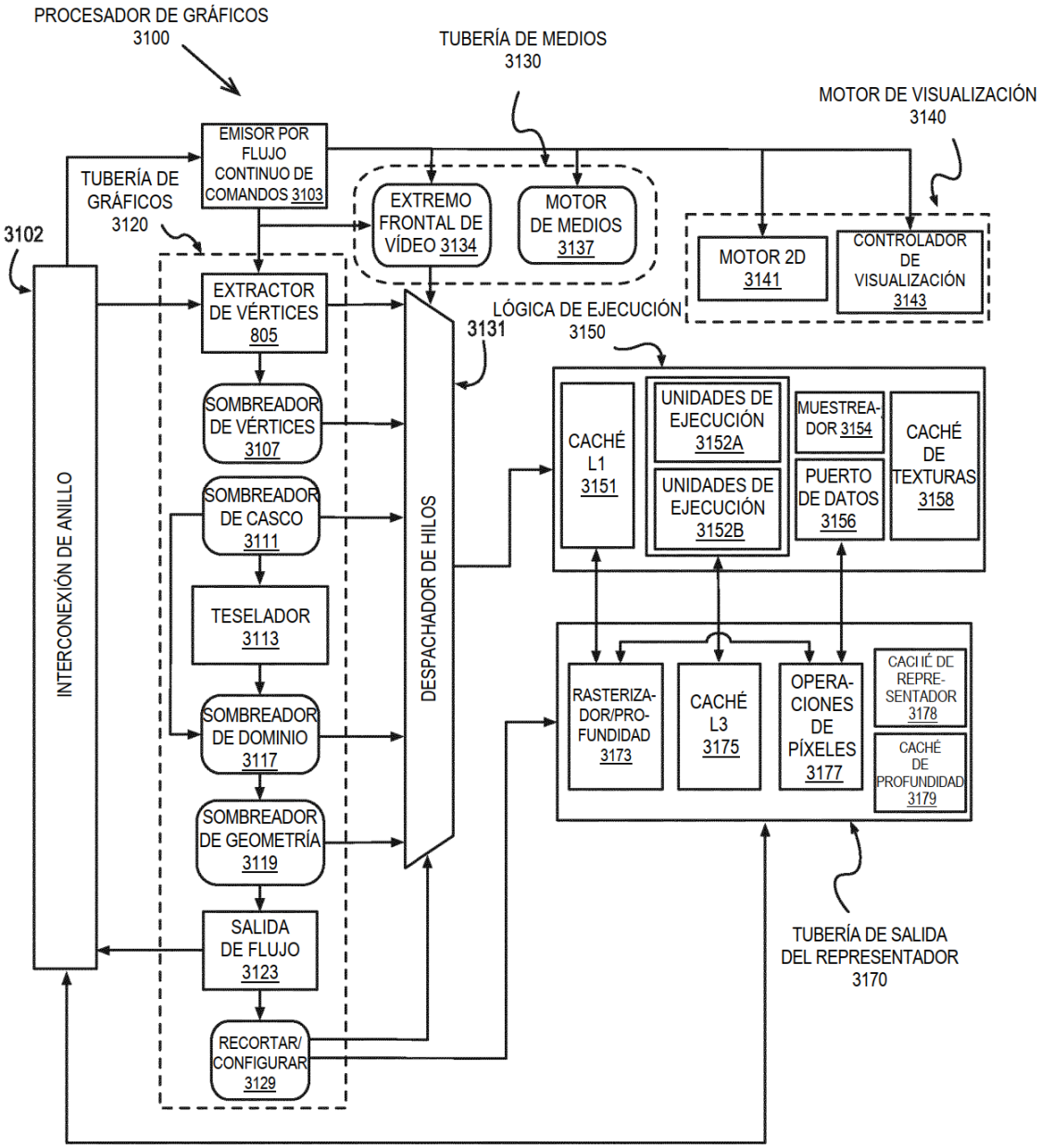

PROCESADOR DE GRÁFICOS 3100
TUBERÍA DE MEDIOS 3130
MOTOR DE VISUALIZACIÓN 3140
EMISOR POR FLUJO CONTINUO DE COMANDOS 3103
TUBERÍA DE GRÁFICOS 3120
3102
EXTREMO FRONTAL DE VÍDEO 3134
MOTOR DE MEDIOS 3137
MOTOR 2D 3141
CONTROLADOR DE VISUALIZACIÓN 3143
EXTRACTOR DE VÉRTICES 805
LÓGICA DE EJECUCIÓN 3150
3131
SOMBREADOR DE VÉRTICES 3107
SOMBREADOR DE CASCO 3111
TESELADOR 3113
SOMBREADOR DE DOMINIO 3117
SOMBREADOR DE GEOMETRÍA 3119
SALIDA DE FLUJO 3123
RECORTAR/ CONFIGURAR 3129
INTERCONEXIÓN DE ANILLO
DESPACHADOR DE HILOS
CACHÉ L1 3151
UNIDADES DE EJECUCIÓN 3152A
UNIDADES DE EJECUCIÓN 3152B
MUESTREA-DOR 3154
PUERTO DE DATOS 3156
CACHÉ DE TEXTURAS 3158
RASTERIZA-DOR/PRO-FUNDIDAD 3173
CACHÉ L3 3175
OPERA-CIONES DE PÍXELES 3177
CACHÉ DE REPRE-SENTADOR 3178
CACHÉ DE PROFUNDIDAD 3179
TUBERÍA DE SALIDA DEL REPRESENTADOR 3170

FIG. 31

FIG. 32A **FORMATO DE COMANDO DE PROCESADOR DE GRÁFICOS**
3200

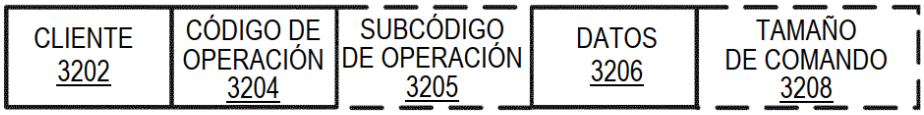
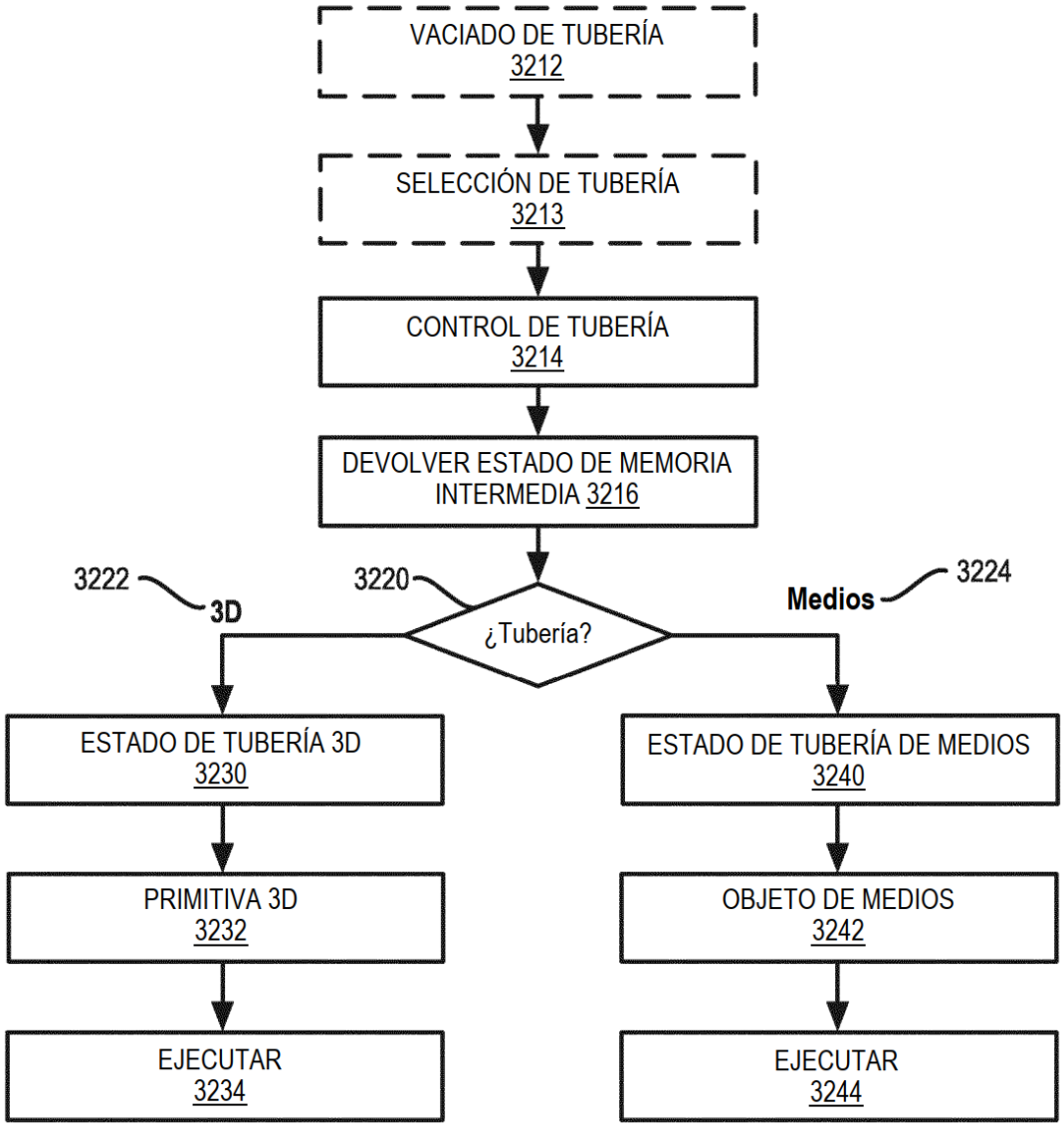


FIG. 32B **SECUENCIA DE COMANDOS DE PROCESADOR DE GRÁFICOS**
3210



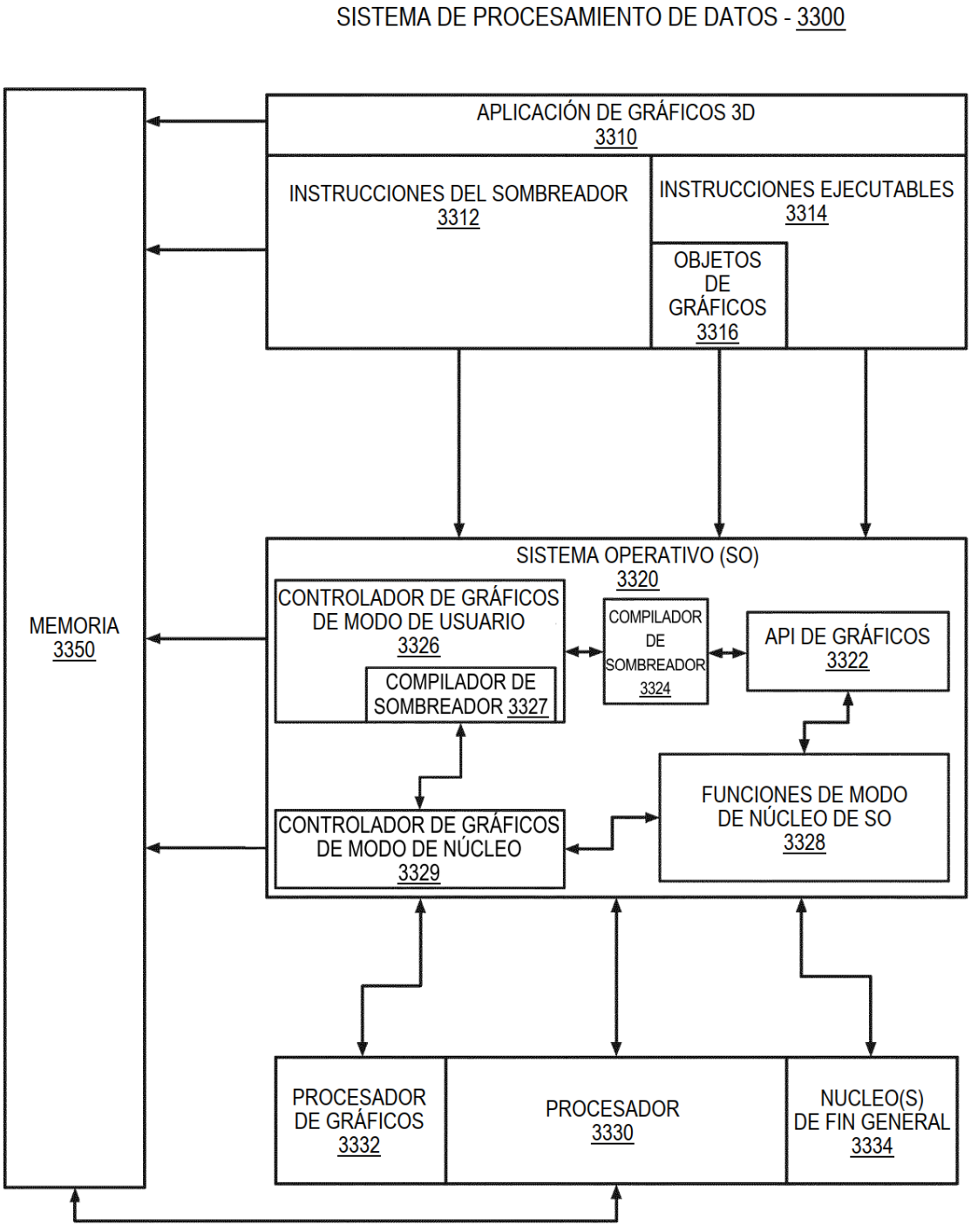

SISTEMA DE PROCESAMIENTO DE DATOS - 3300
APLICACIÓN DE GRÁFICOS 3D 3310
INSTRUCCIONES DEL SOMBREADOR 3312
INSTRUCCIONES EJECUTABLES 3314
OBJETOS DE GRÁFICOS 3316
MEMORIA 3350
SISTEMA OPERATIVO (SO) 3320
CONTROLADOR DE GRÁFICOS DE MODO DE USUARIO 3326
COMPILADOR DE SOMBREADOR 3327
COMPILADOR DE SOMBREADOR 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
NUCLEO(S) DE FIN GENERAL 3334

FIG. 33

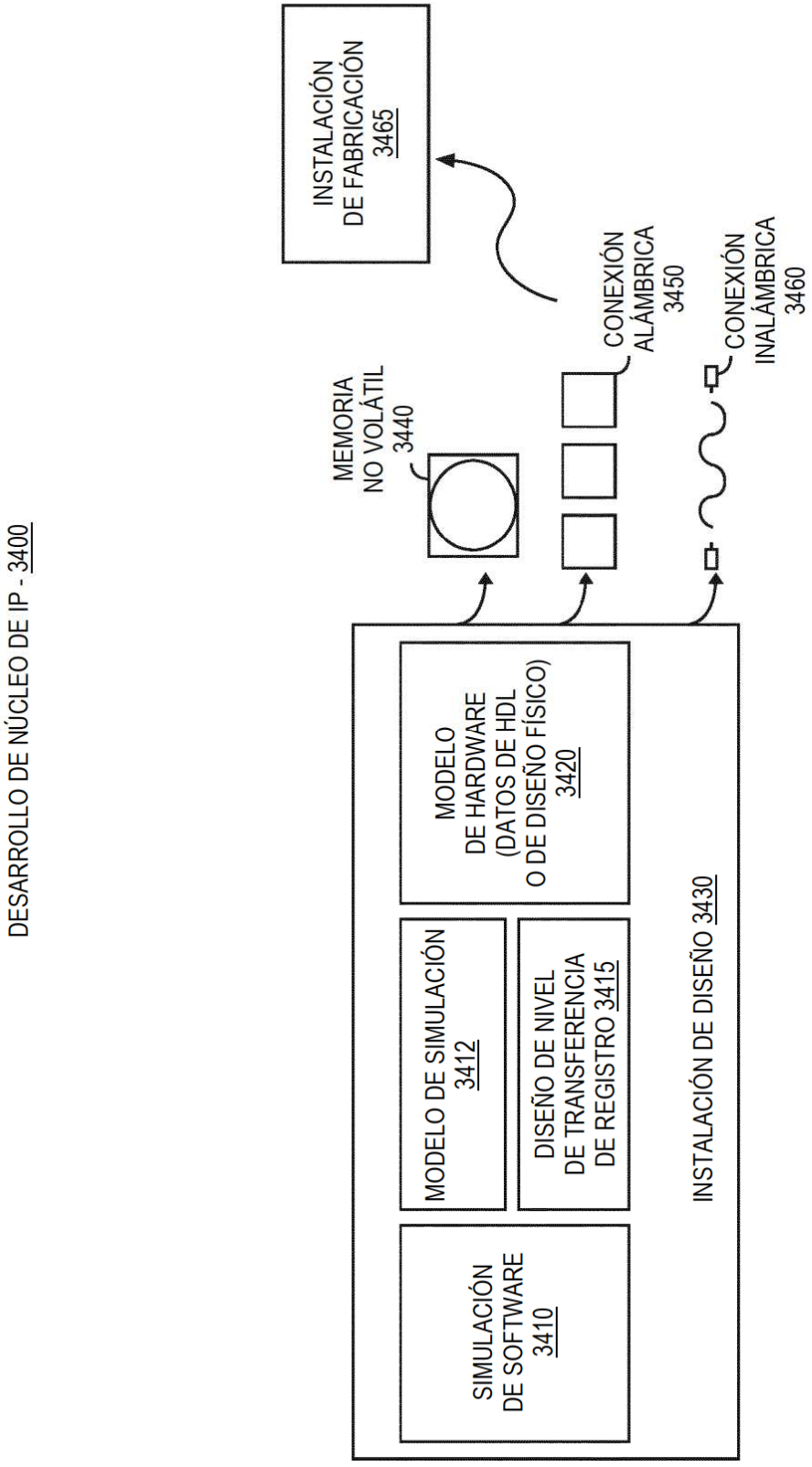

DESARROLLO DE NÚCLEO DE IP - 3400
SIMULACIÓN DE SOFTWARE 3410
MODELO DE SIMULACIÓN 3412
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 3415
MODELO DE HARDWARE (DATOS DE HDL O DE DISEÑO FÍSICO) 3420
INSTALACIÓN DE DISEÑO 3430
MEMORIA NO VOLÁTIL 3440
CONEXIÓN ALÁMBRICA 3450
CONEXIÓN INALÁMBRICA 3460
INSTALACIÓN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

FIG. 34

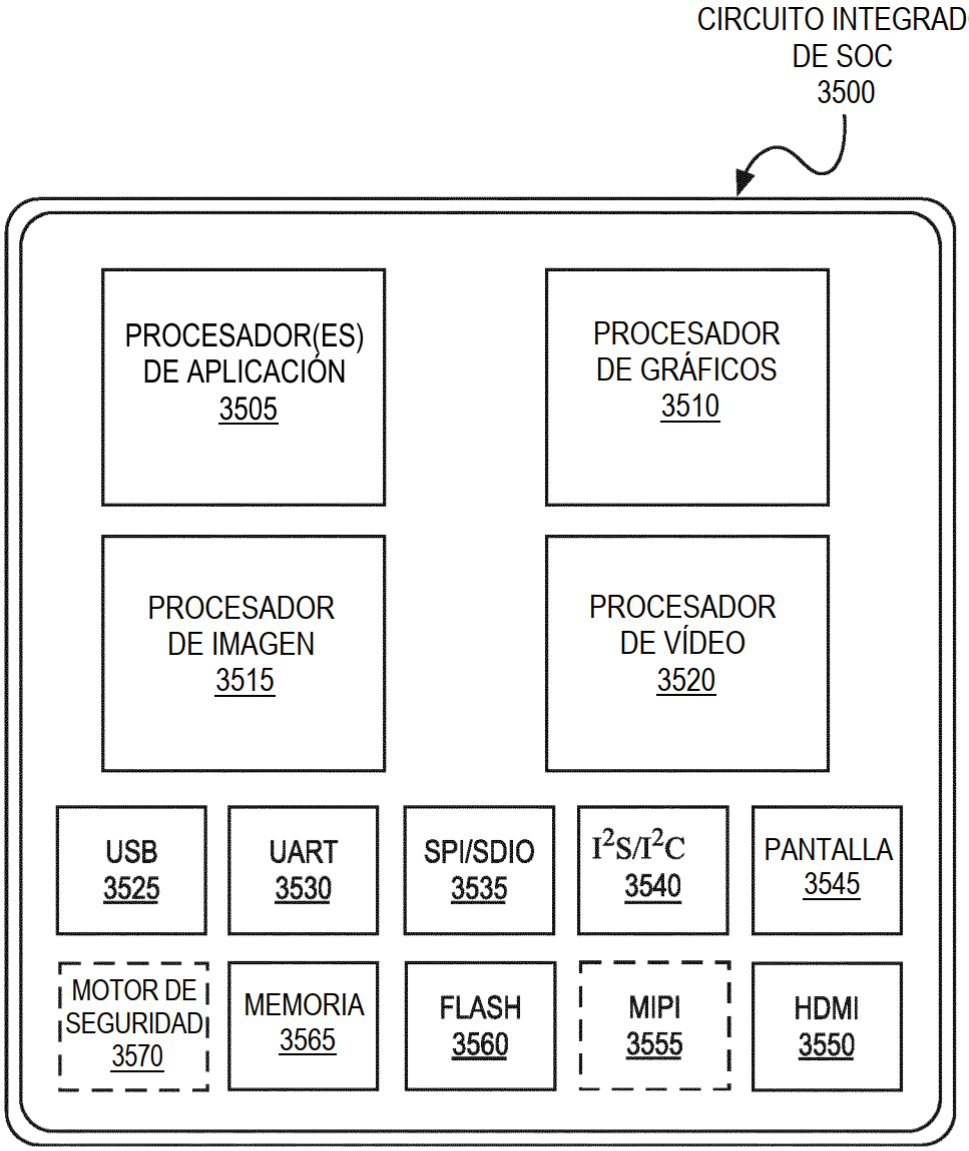

CIRCUITO INTEGRADO
DE SOC
3500
PROCESADOR(ES)
DE APLICACIÓN
3505
PROCESADOR
DE GRÁFICOS
3510
PROCESADOR
DE IMAGEN
3515
PROCESADOR
DE VÍDEO
3520
USB
3525
UART
3530
SPI/SDIO
3535
I2S/I2C
3540
PANTALLA
3545
MOTOR DE
SEGURIDAD
3570
MEMORIA
3565
FLASH
3560
MIPI
3555
HDMI
3550

FIG. 35

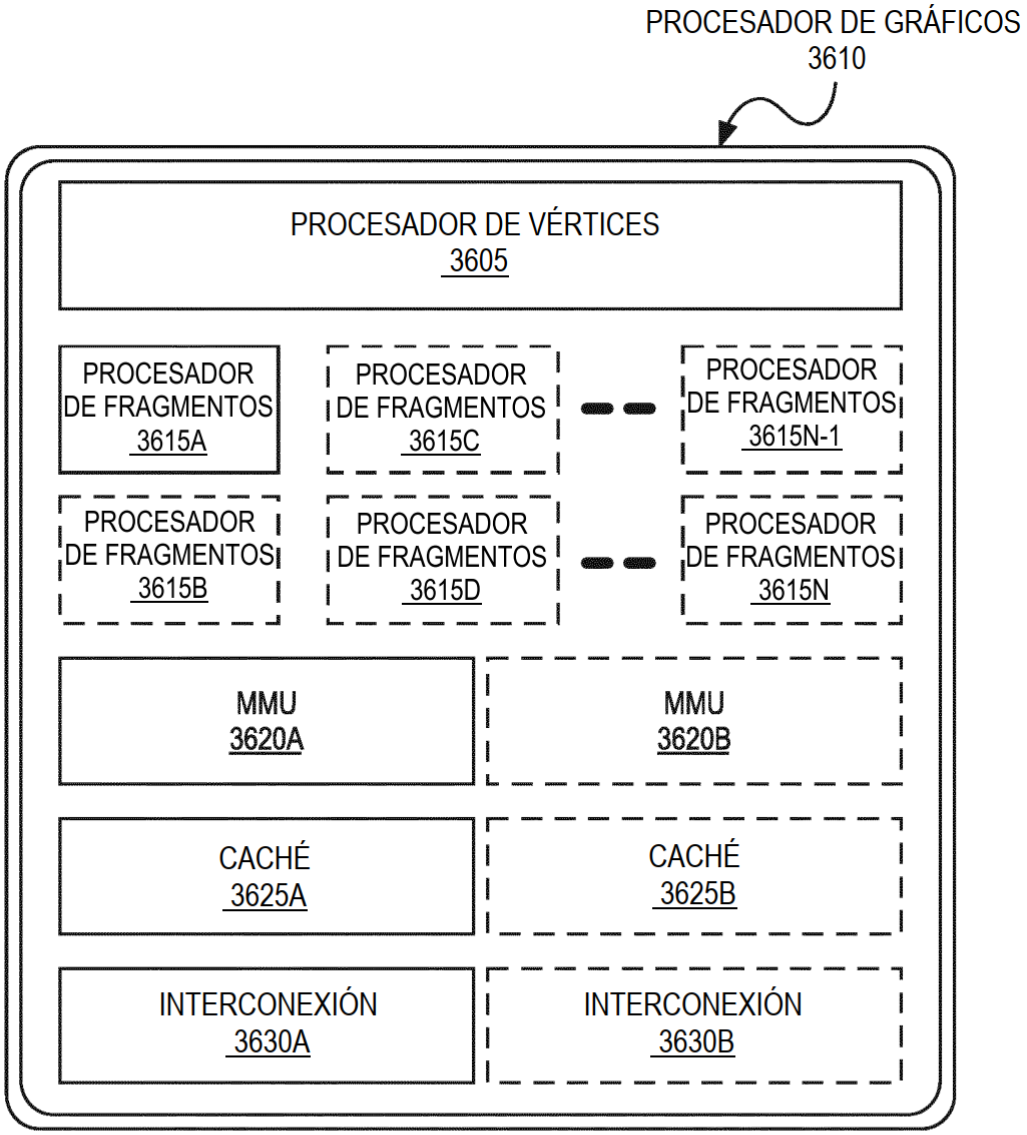

PROCESADOR DE GRÁFICOS
3610
PROCESADOR DE VÉRTICES
3605
PROCESADOR DE FRAGMENTOS
3615A
PROCESADOR DE FRAGMENTOS
3615C
PROCESADOR DE FRAGMENTOS
3615N-1
PROCESADOR DE FRAGMENTOS
3615B
PROCESADOR DE FRAGMENTOS
3615D
PROCESADOR DE FRAGMENTOS
3615N
MMU
3620A
MMU
3620B
CACHÉ
3625A
CACHÉ
3625B
INTERCONEXIÓN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

FIG. 36

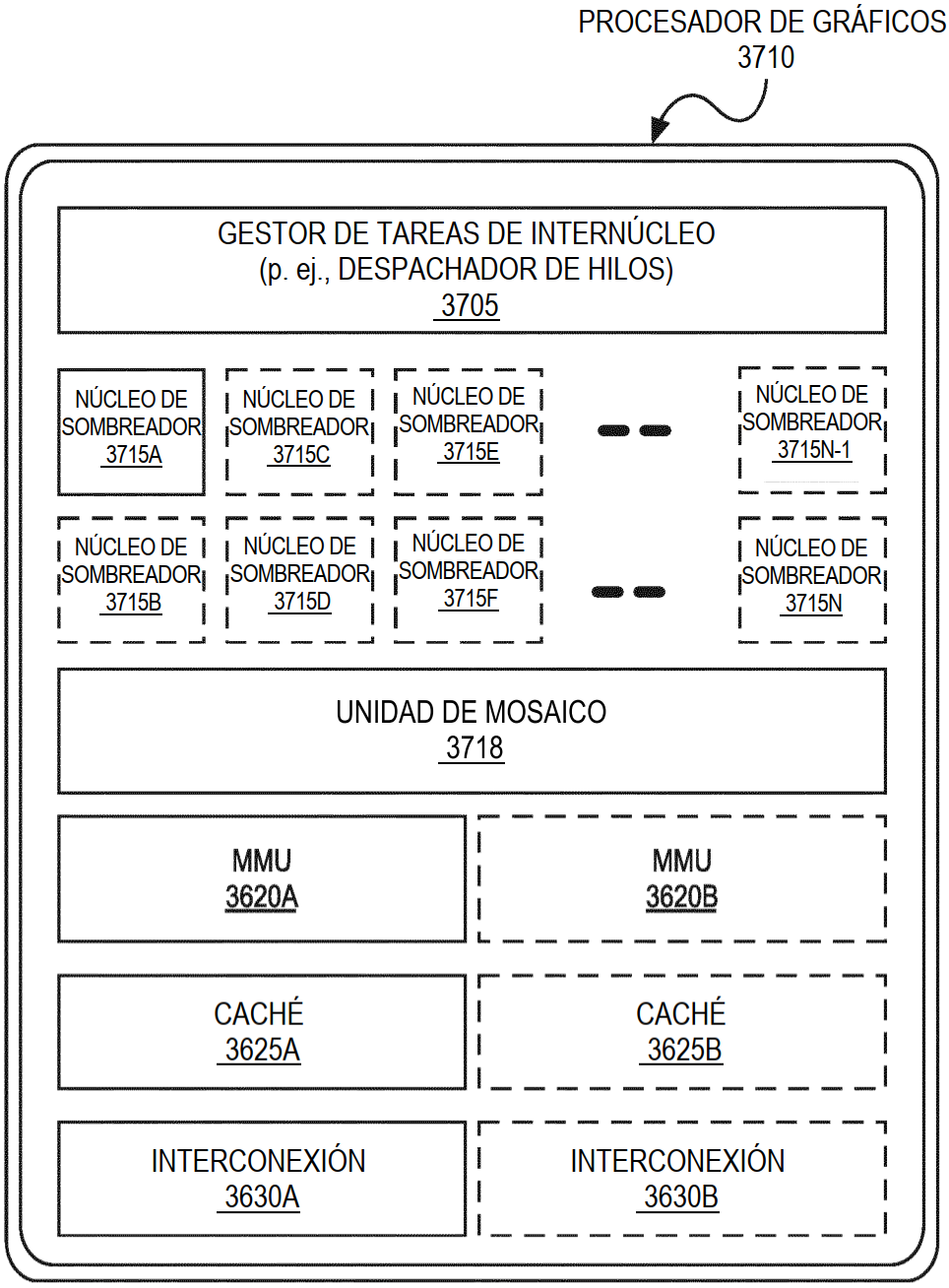

PROCESADOR DE GRÁFICOS
3710
GESTOR DE TAREAS DE INTERNÚCLEO
(p. ej., DESPACHADOR DE HILOS)
3705
NÚCLEO DE SOMBREADOR 3715A
NÚCLEO DE SOMBREADOR 3715C
NÚCLEO DE SOMBREADOR 3715E
NÚCLEO DE SOMBREADOR 3715N-1
NÚCLEO DE SOMBREADOR 3715B
NÚCLEO DE SOMBREADOR 3715D
NÚCLEO DE SOMBREADOR 3715F
NÚCLEO DE SOMBREADOR 3715N
UNIDAD DE MOSAICO
3718
MMU
3620A
MMU
3620B
CACHÉ
3625A
CACHÉ
3625B
INTERCONEXIÓN
3630A
INTERCONEXIÓN
3630B

FIG. 37