

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 982 454**

51 Int. Cl.:

G06F 9/38 (2008.01)

G06F 9/30 (2008.01)

G06F 7/523 (2006.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **23.03.2018** **E 20155873 (1)**

97 Fecha y número de publicación de la concesión europea: **10.04.2024** **EP 3671439**

54 Título: **Optimizaciones de cálculo para redes neuronales**

30 Prioridad:

24.04.2017 US 201715494710

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

16.10.2024

73 Titular/es:

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

72 Inventor/es:

**NEALIS, KEVIN;
YAO, ANBANG;
CHEN, XIAOMING;
OULD-AHMED-VALL, ELMOUSTAPHA;
BAGHSORKHI, SARA S.;
NURVITADHI, ERIKO;
VEMBU, BALAJI;
GALOPPO VON BORRIES, NICOLAS C.;
BARIK, RAJKISHORE;
LIN, TSUNG-HAN y
SINHA, KAMAL**

74 Agente/Representante:

LEHMANN NOVO, María Isabel

ES 2 982 454 T3

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

DESCRIPCIÓN

Optimizaciones de cálculo para redes neuronales

5 **SECTOR**

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 propósito general.

10 **ANTECEDENTES DE LA DESCRIPCIÓN**

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

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

Renzo Andri y col.: "YodaNN: An Architecture for Ultra-Power Binary-Weight CNN Acceleration", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 24 de febrero de 2017, divulga redes neuronales convolucionales (CNN) con pesos binarios (+1/-1). Los pesos se almacenan y se actualizan con total precisión, pero se binarizan para su propagación hacia delante y hacia atrás. El documento hace referencia a la binarización determinista y estocástica. Una multiplicación de 12x12 se sustituye por operaciones de complemento a dos y multiplexores. Además, se divulga un multiplicador y un árbol sumador.

En lo sucesivo, las referencias a "realizaciones" han de interpretarse como "ejemplos no cubiertos por las reivindicaciones" si no se refieren o bien a un aparato de cálculo o bien a un método como se define en las reivindicaciones.

40 **BREVE DESCRIPCIÓN DE LOS DIBUJOS**

De modo que las características de la presente invención puedan entenderse en detalle, puede tenerse una descripción más particular de la invención 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 solo ilustran realizaciones habituales y, por lo tanto, no han de considerarse limitantes del alcance de todas las realizaciones.

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

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

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

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

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

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

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

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

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

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

5 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;

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

la **Figura 14** ilustra una unidad aritmético-lógica, de acuerdo con una realización;

15 la **Figura 15** ilustra una lógica dentro de una ALU para realizar un desplazamiento-acumulación de cañón fusionado, de acuerdo con una realización;

la **Figura 16** es un diagrama de flujo de una lógica para una instrucción de desplazamiento-acumulación de cañón fusionado, de acuerdo con una realización;

20 la **Figura 17** ilustra cálculos para una red neuronal ponderada binaria que tiene características de N bits, de acuerdo con una realización;

la **Figura 18** ilustra una lógica dentro de una ALU para realizar una operación de multiplicación-acumulación de N bits por 1 bit fusionada, de acuerdo con una realización;

25 la **Figura 19** ilustra multiplicadores ilustrativos, de acuerdo con una realización;

la **Figura 20** ilustra un diagrama de flujo de una lógica para una instrucción de desplazamiento-acumulación de cañón fusionado, de acuerdo con una realización;

30 la **Figura 21** ilustra una lógica para realizar una operación de XNOR y recuento de población fusionado, de acuerdo con una realización;

la **Figura 22** es un diagrama de bloques de un sistema de procesamiento, de acuerdo con una realización;

35 la **Figura 23** es un diagrama de bloques de un procesador de acuerdo con una realización;

la **Figura 24** es un diagrama de bloques de un procesador de gráficos, de acuerdo con una realización;

40 la **Figura 25** es un diagrama de bloques de un motor de procesamiento de gráficos de un procesador de gráficos de acuerdo con algunas realizaciones;

la **Figura 26** es un diagrama de bloques de un procesador de gráficos proporcionado por una realización adicional;

45 la **Figura 27** ilustra la lógica de ejecución de hilos que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones;

50 la **Figura 28** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos de acuerdo con algunas realizaciones;

la **Figura 29** es un diagrama de bloques de un procesador de gráficos de acuerdo con otra realización.

55 La **Figura 30A-30B** ilustra un formato de comando de procesador de gráficos y secuencia de comandos, de acuerdo con algunas realizaciones;

la **Figura 31** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos de acuerdo con algunas realizaciones;

60 la **Figura 32** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP, de acuerdo con una realización;

la **Figura 33** es un diagrama de bloques que ilustra un sistema ilustrativo en un circuito de chip integrado, de acuerdo con una realización;

65 la **Figura 34** es un diagrama de bloques que ilustra un procesador de gráficos ilustrativo adicional; y

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

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

En la 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 complicar 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 procesadores 102 y una memoria de sistema 104 que se comunica por medio de una ruta de interconexión que puede incluir un concentrador de memoria 105. El concentrador de memoria 105 puede ser un componente separado dentro de un componente de conjunto de chips o puede integrarse dentro de los uno o más procesadores 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 mediante un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede posibilitar que el sistema informático 100 reciba una entrada desde uno o más dispositivo(s) de entrada 108. Adicionalmente, el concentrador de E/S 107 puede posibilitar que un controlador de visualización, que puede incluirse en el/los uno o más procesadores 102, proporcione salidas a uno o más dispositivos de visualización 110A. En una realización, los uno o más dispositivos de visualización 110A acoplados con el concentrador de E/S 107 pueden incluir un dispositivo de visualización local, interno o embebido.

En una realización, el subsistema de procesamiento 101 incluye uno o más procesadores paralelos 112 acoplados al concentrador de memoria 105 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 específica del proveedor. En una realización, los uno o más procesadores paralelos 112 forman un sistema de procesamiento paralelo o vectorial de enfoque computacional que incluye un gran número de núcleos de procesamiento y/o agrupaciones de procesamiento, tal como un procesador de muchos núcleos integrados (MIC). En una realización, los uno o más procesadores paralelos 112 forman un subsistema de procesamiento de gráficos que puede proporcionar píxeles a uno de los uno o más dispositivos de visualización 110A acoplados por medio del concentrador de E/S 107. Los uno o más procesadores paralelos 112 pueden incluir también un controlador de visualización y una interfaz de visualización (no mostrados) para posibilitar una conexión directa a uno o más dispositivos de visualización 110B.

Dentro del subsistema de E/S 111, una unidad de almacenamiento de sistema 114 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, tales como un adaptador de red 118 y/o un adaptador de red inalámbrica 119 que pueden estar integrados en la plataforma, y otros diversos dispositivos que puedan añadirse por medio de uno o más dispositivo(s) de adición 120. El adaptador de red 118 puede ser un adaptador de Ethernet u otro adaptador de red cableado. El adaptador de red inalámbrico 119 puede incluir uno o más de un dispositivo de red de Wi-Fi, de Bluetooth, de comunicación de campo cercano (NFC) o de otro tipo que incluya una o más radios inalámbricas.

El sistema informático 100 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 que también puede conectarse al concentrador de E/S 107. Las rutas de comunicaciones 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 comunicaciones de bus o de punto a punto y/o protocolo(s), tal como la interconexión de alta velocidad NVLink, o protocolos de interconexión conocidos en la técnica.

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

Se apreciará que el sistema informático 100 mostrado en el presente documento es ilustrativo y que son posibles variaciones y modificaciones. La topología de conexión, incluyendo el número y disposición de puentes, el número de procesador(es) 102 y el número de procesador(es) paralelo(s) 112 puede modificarse como se desee. Por ejemplo, en algunas realizaciones, la memoria de sistema 104 está conectada a los procesador(es) 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/los procesador(es) 102. En otras topologías alternativas, el/los procesador(es) paralelo(s) 112 se conecta(n) al concentrador de E/S 107 o directamente a uno del/de los uno o más procesador(es) 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 integrarse en un único chip. Algunas realizaciones pueden incluir dos o más conjuntos de procesadores 102 conectados mediante múltiples zócalos, que pueden acoplarse con dos o más instancias del/de los procesador(es) paralelo(s) 112.

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

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

En una realización, el procesador paralelo 200 incluye una unidad de procesamiento paralelo 202. La unidad de procesamiento paralelo incluye una unidad de E/S 204 que posibilita la comunicación con otros dispositivos, incluidas otras instancias de la unidad de procesamiento paralelo 202. La unidad de E/S 204 puede conectarse 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 operaciones de procesamiento y la barra transversal de memoria 216 recibe comandos dirigidos a realizar operaciones de memoria.

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

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

agrupaciones de procesamiento 212 puede ejecutar un gran número de hilos concurrentes. El planificador 210 puede asignar trabajo a las agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 usando diversos algoritmos de planificación y/o de distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surja para cada tipo de programa o 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 agrupaciones de procesamiento 212. En una realización, pueden asignarse diferentes agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 para procesar diferentes tipos de programas o para realizar diferentes tipos de cálculos.

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

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

En 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 agrupaciones de procesamiento 212. En algunas realizaciones, porciones de la matriz de agrupaciones de procesamiento 212 pueden configurarse para realizar diferentes tipos de procesamiento. Por ejemplo, una primera porción puede configurarse para realizar un sombreado de vértices y una generación de topología, una segunda porción puede configurarse para realizar sombreado de teselación y de geometría, y una tercera porción puede configurarse para realizar sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen representada para su visualización. Los datos intermedios producidos por una o más de las agrupaciones 214A-214N pueden almacenarse en memorias intermedias para permitir que se transmitan los datos intermedios entre las agrupaciones 214A-214N para su procesamiento adicional.

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

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

En diversas realizaciones, las unidades de memoria 224A-224N pueden incluir diversos tipos de dispositivos de memoria, 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 alto ancho de banda (HBM). Los expertos en la materia apreciarán que la implementación específica de las unidades de memoria 224A-224N puede variar y puede seleccionarse a partir 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 lo largo de las unidades de memoria 224A-224N, permitiendo que las unidades de subdivisión 220A-220N escriban porciones de cada objetivo de representación en paralelo para usar de manera eficiente el ancho de banda disponible de la memoria de procesador paralelo 222. En algunas realizaciones, 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 junto con memoria caché local.

En una realización, una cualquiera de las agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 224A-224N dentro de la memoria de procesador paralelo 222. La barra transversal de memoria 216 puede configurarse para transferir la salida de cada agrupación 214A-214N a cualquier unidad de subdivisión 220A-220N o a otra agrupación 214A-214N, que puede realizar operaciones de procesamiento adicionales sobre la salida. Cada agrupación 214A-214N 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 configurarse para interfundar incluso si las diferentes instancias tienen diferentes números de núcleos de procesamiento, diferentes cantidades de memoria de procesador paralelo local y/u otras diferencias de configuración. Por ejemplo, y en una realización, algunas instancias de la unidad de procesamiento paralelo 202 pueden incluir unidades de coma flotante de precisión superior en relación con otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 202 o el procesador paralelo 200 pueden implementarse en una diversidad de configuraciones y factores de forma, incluyendo, pero sin limitación, ordenadores personales de sobremesa, portátiles o de mano, servidores, estaciones de trabajo, consolas de juegos y/o sistemas embebidos.

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 fotogramas 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 de almacenamiento recibidas desde la barra transversal de memoria 216 y la ROP 226. Los desaciertos de lectura y las solicitudes de escritura diferida urgente son emitidas por la caché L2 221 a la interfaz de memoria intermedia de fotogramas 225 para su procesamiento. También pueden enviarse actualizaciones a la memoria intermedia de fotogramas mediante la interfaz de memoria intermedia de fotogramas 225 para su procesamiento. En una realización, la interfaz de memoria intermedia de fotogramas 225 interactúa 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 emite entonces datos de gráficos procesados que se almacenan en una memoria de gráficos. En algunas realizaciones, la ROP 226 incluye lógica de compresión para comprimir datos de profundidad o de color que se escriben en memoria y descomprimir datos de profundidad o de color que se leen desde memoria. La lógica de compresión puede ser lógica de compresión sin pérdidas que hace uso de uno o más de múltiples algoritmos de compresión. El tipo de compresión que es realizado por la ROP 226 puede variar basándose en las características estadísticas de los datos a comprimir. Por ejemplo, en una realización, se realiza una compresión de color delta sobre datos de profundidad y de color por tesela.

En algunas realizaciones, la ROP 226 se incluye dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 214A-214N de la Figura 2) en lugar de dentro de la unidad de subdivisión 220. En tal realización, se transmiten solicitudes de lectura y de escritura de datos de píxel a través de la barra transversal de memoria 216 en lugar de datos de fragmento de píxel. Los datos de gráficos procesados pueden visualizarse en un dispositivo de

visualización, tal como uno de los uno o más dispositivo(s) de visualización 110 de la Figura 1, encaminarse para su procesamiento adicional por medio del/de los procesador(es) 102, o encaminarse para su procesamiento adicional por medio de 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 configurarse 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 instrucciones de única instrucción de 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 de múltiples hilos (SIMT) para soportar la ejecución paralela de un gran número de hilos generalmente sincronizados, usando una unidad de instrucciones común configurada para emitir instrucciones en un conjunto de motores de procesamiento dentro de cada una de las agrupaciones de procesamiento. A diferencia del régimen de ejecución de SIMD, donde todos los motores de procesamiento ejecutan habitualmente instrucciones idénticas, la ejecución de SIMT permite que diferentes hilos sigan más fácilmente rutas de ejecución divergentes a través de un programa de hilos dado. Los expertos en la materia entenderán que un régimen de procesamiento de SIMD representa un subconjunto funcional de un régimen de procesamiento de SIMT.

El funcionamiento de la agrupación de procesamiento 214 puede controlarse mediante un gestor de canalizaciones 232 que distribuye tareas de procesamiento a procesadores paralelos de SIMT. El gestor de canalizaciones 232 recibe instrucciones desde el planificador 210 de la Figura 2 y gestiona la ejecución de esas instrucciones 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. Una o más instancias del multiprocesador de gráficos 234 pueden incluirse dentro de una agrupación de procesamiento 214. El multiprocesador de gráficos 234 puede procesar datos y puede usarse una barra transversal de datos 240 para distribuir los datos procesados a uno de múltiples posibles destinos, que incluyen otras unidades sombreadoras. El gestor de canalizaciones 232 puede facilitar la distribución de datos procesados especificando destinos para que se distribuyan datos procesados mediante la barra transversal de datos 240.

Cada multiprocesador de gráficos 234 dentro de la agrupación de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades aritmético-lógicas, unidades de carga-almacenamiento, etc.). La lógica de ejecución funcional puede configurarse de una manera canalizada en la que pueden emitirse nuevas instrucciones antes de que se hayan completado instrucciones previas. 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 cálculo de diversas funciones algebraicas. En una realización, puede aprovecharse el mismo hardware de unidades funcionales para realizar diferentes operaciones, y puede estar presente cualquier combinación de unidades funcionales.

Las instrucciones transmitidas a la agrupación de procesamiento 214 constituyen un hilo. Un conjunto de hilos que se ejecutan a lo largo del conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa sobre diferentes datos de entrada. Cada hilo dentro de un grupo de hilos 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 pueden estar inactivos durante los ciclos en los que se está procesando ese grupo de hilos. Un grupo de hilos puede incluir también más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando el grupo de hilos incluye más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234, puede realizarse un procesamiento a lo largo 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 caché interna para realizar operaciones de carga y de almacenamiento. 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 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 una o más de memoria de procesador paralelo local y/o memoria de sistema. Cualquier memoria externa a la unidad de procesamiento paralelo 202 puede usarse 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 páginas (PTE) usadas para mapear una dirección virtual con una dirección física de una tesela y, opcionalmente, un índice de líneas de caché. La MMU 245 puede incluir memorias intermedias de traducción adelantada (TLB) de direcciones o cachés que pueden residir dentro del multiprocesador de gráficos 234 o la caché L1 o la agrupación de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso de datos de superficie para permitir una intercalación de solicitud eficiente entre unidades de subdivisión. El índice de líneas de caché puede usarse para determinar si una solicitud de una línea de caché es un acierto o un desacierto.

En aplicaciones de gráficos e informáticas, una agrupación de procesamiento 214 puede configurarse de manera que cada multiprocesador de gráficos 234 se acopla 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 prerrasterización) está configurada para recibir datos desde el multiprocesador de gráficos 234, dirigir datos a unidades de ROP, que pueden ubicarse 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 variaciones y 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 solo se muestra una agrupación de procesamiento 214, una unidad de procesamiento paralelo como se describe en el presente documento puede incluir cualquier número de instancias de la agrupación de procesamiento 214. En una realización, cada agrupación de procesamiento 214 puede configurarse para funcionar 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 canalización 232 de la agrupación de procesamiento 214. El multiprocesador de gráficos 234 tiene una canalización de ejecución que incluye, pero sin limitación, una caché de instrucciones 252, una unidad de instrucción 254, una unidad de mapeo de direcciones 256, un archivo de registro 258, uno o más núcleos de unidad de procesamiento de gráficos de propósito general (GPGPU) 262 y una o más unidades de carga/almacenamiento 266. Los núcleos de GPGPU 262 y las unidades de carga/almacenamiento 266 se acoplan con la memoria caché 272 y la memoria compartida 270 mediante una interconexión de memoria y caché 268.

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

El archivo de registro 258 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de gráficos 234. El archivo de registro 258 proporciona almacenamiento temporal para los operandos conectados a las rutas de datos de las unidades funcionales (por ejemplo, núcleos de GPGPU 262, unidades de carga/almacenamiento 266) del multiprocesador de gráficos 234. En una realización, el archivo de registro 258 se divide entre cada una de las unidades funcionales de manera que cada unidad funcional está asignada a una porción especializada del archivo de registro 258. En una realización, el archivo de registro 258 se divide entre las diferentes urdimbres que se ejecutan mediante el multiprocesador de gráficos 234.

Cada núcleo de GPGPU 262 puede incluir unidades de coma flotante (FPU) y/o unidades aritmético-lógicas (ALU) de números enteros que se usan para ejecutar instrucciones del multiprocesador de gráficos 234. 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 234 puede incluir adicionalmente una o más unidades de función fija o de función especial para realizar funciones específicas tales

como operaciones de copiar rectángulo o de mezcla de píxeles. En una realización, uno o más de los núcleos de GPGPU puede incluir también lógica de función fija o especial.

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

La interconexión de memoria y caché 268 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador de gráficos 234 al archivo 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/almacenamiento 266 implemente operaciones de carga y de almacenamiento entre la memoria compartida 270 y el archivo de registro 258. El archivo de registro 258 puede funcionar a la misma frecuencia que los núcleos de GPGPU 262, por lo tanto, la transferencia de datos entre los núcleos de GPGPU 262 y el archivo de registro 258 es de 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 una caché gestionada por programa. Los hilos que se ejecutan en los núcleos de GPGPU 262 pueden almacenar, de manera programática, datos dentro de la memoria compartida además de los datos almacenados automáticamente en caché que se almacenan dentro de la memoria caché 272.

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

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

Los diversos componentes 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 sobre la que se apila cada componente del multiprocesador de gráficos 325. Los componentes del multiprocesador de gráficos 325 se comunican con componentes remotos a través del tejido de interconexión 327. Por ejemplo, cada uno de los núcleos de GPGPU 336A-336B, 337A-337B y 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 instrucciones, archivos de registro, núcleos de GPGPU y unidades de carga-almacenamiento, como se ilustra en la Figura 2D y en la Figura 3A. Los recursos de ejecución 356A-356D pueden funcionar en conjunto con la(s) unidad(es) de textura 360A-360D para operaciones de textura, mientras comparten una caché de instrucciones 354 y una memoria compartida 362. En una realización, los recursos de ejecución 356A-356D pueden compartir una caché de instrucciones 354 y una memoria compartida 362, así como múltiples instancias de una memoria caché de textura y/o de datos 358A-358B. Los diversos componentes pueden comunicarse a través de un tejido de interconexión 352 similar al tejido de interconexión 327 de la Figura 3A.

Los expertos en la materia entenderán que la arquitectura descrita en las Figuras 1, 2A-2D y 3A-3B es descriptiva y no limitante en cuanto al alcance de las presentes realizaciones. Por lo tanto, las técnicas descritas en el presente documento pueden implementarse en cualquier unidad de procesamiento configurada apropiadamente, incluyendo, sin limitación, uno o más procesadores de aplicaciones móviles, una o más unidades centrales de procesamiento

(CPU) de sobremesa o de servidor, incluyendo CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tales como la unidad de procesamiento paralelo 202 de la Figura 2, así como uno o más procesadores de gráficos o unidades de procesamiento de propósito especial, sin apartarse del alcance de las realizaciones descritas en el presente documento.

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

Técnicas para interconexión de GPU a procesador de anfitrión

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

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

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

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

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

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

incluirse en la jerarquía de almacenamiento en caché y ser compartidas por conjuntos de 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 es compartida por dos núcleos adyacentes. El procesador 407 y el módulo de integración de acelerador de gráficos 446 se conectan con la memoria de sistema 441, que puede incluir las memorias de procesador 401-402

La coherencia se mantiene para los datos e instrucciones almacenados en las diversas cachés 462A-462D, 456 y la memoria de sistema 441 a través de comunicación entre núcleos a través de un bus de coherencia 464. Por ejemplo, cada caché puede tener una lógica/circuitería de coherencia de caché asociada con la misma 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 coherencia/monitorización de caché son bien entendidas por los expertos en la técnica y no se describirán en el presente caso en detalle para evitar complicar los principios subyacentes de la invención.

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

En una implementación, un circuito de integración de acelerador 436 proporciona servicios de gestión de caché, de acceso de memoria, de gestión de contexto y de gestión de interrupciones en nombre de una pluralidad de motores de procesamiento de gráficos 431, 432, N del módulo de aceleración de gráficos 446. Cada uno de los motores de procesamiento de gráficos 431, 432, N puede comprender una unidad de procesamiento de gráficos (GPU) separada. Como alternativa, los motores de procesamiento de gráficos 431, 432, N pueden comprender diferentes tipos de 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 de memoria para acceder a la memoria de sistema 441. La MMU 439 puede incluir también una memoria intermedia de traducción adelantada (TLB) (no mostrada) para almacenar en caché las traducciones de dirección virtual/eficaz a física/real. En una realización, el circuito de integración de acelerador 436 incluye una unidad de extracción 491 para extraer comandos, instrucciones, descriptores de trabajo, etc., que definen operaciones que hay que realizar. En una implementación, una caché 438 almacena comandos y datos para un acceso eficiente por los motores de procesamiento de gráficos 431-432, N. En una realización, los datos almacenados en la caché 438 y en las memorias de gráficos 433-434, N se mantienen coherentes con las cachés de núcleo 462A-462D, 456 y la memoria de sistema 411. Como se menciona, esto puede conseguirse mediante el circuito intermediario 425 que toma parte en el mecanismo de coherencia de caché en nombre de la caché 438 y las memorias 433-434, N (por ejemplo, enviando actualizaciones a la caché 438 relacionadas con modificaciones/accesos de líneas de caché en las cachés de procesador 462A-462D, 456 y recibiendo actualizaciones desde la caché 438).

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

En una implementación, direcciones virtuales/eficaces desde un motor de procesamiento de gráficos 431 son traducidas, por la MMU 439, a direcciones reales/físicas en la memoria de sistema 411. Una realización del circuito de integración de acelerador 436 soporta múltiples (por ejemplo, 4, 8, 16) módulos de aceleración de gráficos 446 y/u otros dispositivos de aceleración. El módulo de acelerador de gráficos 446 puede dedicarse 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 "segmentos" que se asignan a diferentes VM y/o aplicaciones basándose en los requisitos de procesamiento y las propiedades asociadas con las VM y/o las aplicaciones.

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

Debido a que los recursos de hardware de los motores de procesamiento de gráficos 431-432, N se mapean explícitamente al espacio de direcciones real visto por el procesador de anfitrión 407, cualquier procesador de anfitrión puede direccionar estos recursos directamente usando un valor de dirección efectivo. Una función del circuito de integración de acelerador 436, en 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 menciona, en la realización ilustrada, una o más memorias de gráficos 433-434, M están acopladas a cada uno de los motores de procesamiento de gráficos 431-432, N, respectivamente. Las memorias de gráficos 433-434, M almacenan instrucciones y datos que son procesados por cada uno de los motores de procesamiento de gráficos 431-432, N. Las memorias de gráficos 433-434, M pueden ser memorias volátiles, tales como DRAM (incluyendo DRAM apiladas), memoria 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 sean datos que serán usados con mayor frecuencia por los motores de procesamiento de gráficos 431-432, N y, preferentemente, no usados por los núcleos 460A-460D (al menos no con frecuencia). De manera similar, el mecanismo de desvío intenta mantener datos que necesitan 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, pueden utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración de acelerador 436 puede realizar las mismas operaciones que las descritas con respecto a la **Figura 4B**, pero potencialmente a un caudal superior dada su proximidad estrecha al bus de coherencia 462 y a las cachés 462A-462D, 426.

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

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

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

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

La **Figura 4D** ilustra un segmento de integración de acelerador 490 ilustrativo. Como se usa en el presente documento, un "segmento" comprende una porción especificada de los recursos de procesamiento del circuito de integración de acelerador 436. El espacio de direcciones efectivo de 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 las invocaciones de GPU 481 desde las aplicaciones 480 ejecutadas en el procesador 407. Un elemento de proceso 483 contiene el estado de proceso para la aplicación 480 correspondiente. Un descriptor de trabajo (WD) 484 contenido

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

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

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

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

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

Tabla 1 - Registros inicializados por hipervisor

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

35 En la **Tabla 2** se muestran los registros ilustrativos que pueden inicializarse por el sistema operativo.

Tabla 2 - Registros inicializados por sistema operativo

1	Identificación de procesos e hilos
2	Puntero de guardado/restauración de contexto de dirección eficaz (EA)
3	Puntero de registro de utilización de acelerador de dirección virtual (VA)
4	Puntero de tabla de segmentos de almacenamiento de dirección virtual (VA)
5	Máscara de autoridad
6	Descriptor de trabajo

40

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

La **Figura 4E** ilustra detalles adicionales para una realización de un modelo compartido. Esta realización incluye un espacio de direcciones real de hipervisor 498 en el que se almacena una lista de elementos de proceso 499. El espacio de direcciones real de hipervisor 498 es accesible 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 subdivisiones en el sistema usen un módulo de aceleración de gráficos 446. Hay dos modelos de programación donde el módulo de aceleración de gráficos 446 se comparte por múltiples procesos y subdivisiones: compartido en segmentos de tiempo y compartido dirigido a gráficos.

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

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

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

Tabla 3 - Parámetros de llamada de SO a hipervisor

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

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

5

Tabla 4 - Información de elemento de proceso

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

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

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

25 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 instancias de la 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.

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

45

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

En una implementación, se accede a la entrada de tabla de desvíos asociada a cada acceso a la memoria adjunta a la GPU 420-423 antes del acceso real a la memoria de GPU, lo que provoca las siguientes operaciones. En primer lugar, las solicitudes locales desde la GPU 410-413 que encuentran su página en el desvío de GPU se reenvían directamente a una memoria de GPU 420-423 correspondiente. Las solicitudes locales desde la GPU que encuentran su página en el desvío de anfitrión se reenvían al procesador 405 (por ejemplo, a través de un enlace de alta velocidad como se ha analizado anteriormente). En una realización, las solicitudes del procesador 405 que encuentran la página solicitada en el desvío de procesador de anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, solicitudes dirigidas a una página con desvío de GPU pueden redirigirse a la GPU 410-413. A continuación, la GPU puede hacer que la página pase a un desvío de procesador anfitrión si no está usando actualmente la página.

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

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

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

Canalización de procesamiento de gráficos

La **Figura 5** ilustra una canalización de procesamiento de gráficos 500, de acuerdo con una realización. En una realización, un procesador de gráficos puede implementar la canalización de procesamiento de gráficos 500 ilustrada. El procesador de gráficos puede incluirse 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/de los procesador(es) paralelo(s) 112 de la Figura 1. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento de gráficos 500 mediante una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 202 de la Figura 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 configurarse para realizar las funciones de una o más de una unidad de procesamiento de vértices 504, una unidad de procesamiento de control de teselación 508, una unidad de procesamiento de evaluación de teselación 512, una unidad de procesamiento de geometría 516 y una unidad de procesamiento de fragmentos/píxeles 524. Las funciones del ensamblador de datos 502, los ensambladores de primitivas 506, 514, 518, la unidad de teselación 510, el rasterizador 522 y la unidad de operaciones de rasterización 526 también pueden ser realizadas por otros motores de procesamiento dentro de una agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 214 de la Figura 3) y una unidad de subdivisión correspondiente (por ejemplo, la unidad de subdivisión 220A-220N de la Figura 2). La canalización de procesamiento de gráficos 500 puede implementarse también usando unidades de procesamiento especializadas para una o más funciones. En una realización, una o más porciones de la canalización de procesamiento de gráficos 500 pueden realizarse mediante lógica de procesamiento paralelo dentro de un procesador de propósito general (por ejemplo, CPU). En una realización, una o más porciones de la canalización de procesamiento de gráficos 500 pueden acceder a una memoria en chip (por ejemplo, la memoria de procesador paralelo 222 como en la Figura 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 vértice para superficies y primitivas. El ensamblador de datos 502 emite entonces los datos de vértice, incluyendo los atributos de vértice, a la unidad de procesamiento de vértices 504. La unidad de procesamiento de vértices 504 es una unidad de ejecución programable que ejecuta programas sombreadores de vértices, iluminando y transformando datos de vértice según lo especificado por los programas sombreadores de vértices. La unidad de procesamiento de vértices 504 lee

datos que se almacenan en caché, local o de sistema para su uso en el procesamiento de los datos de vértice y puede programarse para transformar los datos de vértice desde una representación de coordenadas basada en objetos hasta un espacio de coordenadas de espacio mundial o un espacio de coordenadas de dispositivo normalizado.

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

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

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

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

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

La unidad de operaciones de rasterización 526 es una unidad de procesamiento que realiza operaciones de rasterización que incluyen, pero sin limitación, estarcido, prueba z, mezcla y similares, y emite datos de píxel como datos de gráficos procesados para almacenarse en memoria de gráficos (por ejemplo, la memoria de procesador paralelo 222 como en la Figura 2, y/o la memoria de sistema 104 como en la Figura 1), para visualizarse en el/los uno o más dispositivo(s) de visualización 110 o para su procesamiento adicional por uno del/los uno o más procesador(es) 102 o procesador(es) paralelo(s) 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. Pueden diseñarse realizaciones de algoritmos de aprendizaje automático 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 habla y/o reconocimiento de habla.

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

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

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

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

Puede posibilitarse una aceleración de hardware para la aplicación de aprendizaje automático 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 que reoptimizaran, a continuación, la lógica computacional a medida que se desarrollan nuevos procesadores paralelos. En su lugar, la aplicación de aprendizaje automático puede configurarse 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 agrupamiento, 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 desde la aplicación de aprendizaje automático 602 y generar la entrada apropiada a una estructura de cálculo 606. La estructura de cálculo 606 puede abstraer las instrucciones subyacentes proporcionadas al controlador de 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 un 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 lo largo de una diversidad de tipos y generaciones del hardware de GPGPU 610.

Aceleración de aprendizaje automático de GPGPU

La **Figura 7** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela 700, de acuerdo con una realización. En una realización, la unidad de procesamiento de propósito general (GPGPU) 700 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 vincularse directamente a otras instancias de la GPGPU para crear una agrupación de múltiples GPU para mejorar la velocidad de entrenamiento para redes neuronales particularmente profundas.

La GPGPU 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 también puede ser 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 superior para memorias 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, incluyendo una memoria de acceso aleatorio dinámica (DRAM) o una memoria de acceso aleatorio de gráficos, tal como una memoria de acceso aleatorio de gráficos sincrónica (SGRAM), incluyendo memoria de tasa de datos doble de gráficos (GDDR), y también puede incluir una memoria apilada 3D incluyendo, pero sin limitación, una memoria de alto ancho de banda (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 lógica de enteros y de coma flotante que pueden realizar operaciones computacionales con un intervalo de precisiones que incluyen unas adecuadas para cálculos de aprendizaje automático. Por ejemplo, y en 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.

Múltiples instancias de la GPGPU 700 pueden configurarse 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 lo largo de las realizaciones. En una realización, las múltiples instancias de la GPGPU 700 se comunican a través de la interfaz de anfitrión 702. En una realización, la GPGPU 700 incluye un concentrador de E/S 709 que acopla la GPGPU 700 con un enlace de GPU 710 que posibilita una conexión directa a otras instancias de la GPGPU. En una realización, el enlace de GPU 710 está acoplado a un puente de GPU a GPU dedicado que posibilita la comunicación y sincronización entre múltiples instancias de la GPGPU 700. En una realización, el enlace de GPU 710 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En una realización, las múltiples instancias de la GPGPU 700 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red al que puede accederse a través de la interfaz de anfitrión 702. En una realización, el enlace de 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 706A-H en relación con 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 escalar 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 dedicado, tal como el enlace de GPU 710 como en la Figura 7. Los enlaces de GPU de P2P 816 posibilitan una comunicación directa entre cada una de las GPGPU 806A-D sin requerir una comunicación a través del bus de interfaz de anfitrión al que se conecta el procesador 802. Con el tráfico de GPU a GPU dirigido a los enlaces de GPU de P2P, el bus de interfaz de anfitrión permanece disponible para el acceso de memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 800, por ejemplo, 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 realizaciones descritas en el presente documento puede configurarse para realizar los tipos de procesamiento paralelo que son particularmente adecuados para entrenar y desplegar redes neuronales para un aprendizaje automático. Una red neuronal puede generalizarse como una red de funciones que tienen una relación de grafo. Como es bien conocido en la técnica, hay una diversidad de tipos de implementaciones de red neuronal usadas en el aprendizaje automático. Un tipo ilustrativo de red neuronal es la red de realimentación prospectiva, como se ha descrito previamente.

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

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

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

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

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

Las **Figuras 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 las componentes de rojo, verde y azul (RGB) de una imagen de entrada. La entrada 902 puede ser procesada por múltiples capas convolucionales (por ejemplo, la capa convolucional 904, la capa convolucional 906). La salida desde las múltiples capas convolucionales puede ser procesada opcionalmente por un conjunto de capas completamente conectadas 908. Las neuronas en una capa completamente conectada tienen conexiones completas a todas las activaciones en la capa previa, como se ha descrito previamente para una red de realimentación prospectiva. La salida desde las capas completamente conectadas 908 puede usarse para generar un resultado de salida a partir de la red. Las activaciones dentro de las capas completamente conectadas 908 pueden calcularse usando una multiplicación matricial en lugar de una convolución. No todas las implementaciones de CNN hacen uso de las capas completamente conectadas 908. Por ejemplo, en algunas implementaciones, la capa convolucional 906 puede generar una salida para la CNN.

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

La Figura 9B ilustra fases 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 fases de una capa convolucional 914. Las tres fases pueden incluir una fase de convolución 916, una fase de detector 918 y una fase de agrupamiento 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 mapa de características de salida o proporcionar una entrada a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

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

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

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

La salida desde la capa convolucional 914 puede ser procesada, 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 emitir a la segunda capa convolucional 906, mientras que la segunda capa convolucional puede emitir a una primera capa de las capas completamente conectadas 908.

La **Figura 10** ilustra una red neuronal recurrente 1000 ilustrativa. En una red neuronal recurrente (RNN), el estado previo de la red influye sobre la salida del estado actual de la red. Las RNN pueden construirse de una diversidad de maneras usando una diversidad de funciones. El uso de las RNN pivota, en general, alrededor del uso de modelos matemáticos para predecir el futuro basándose en una secuencia anterior de entradas. Por ejemplo, una RNN puede usarse para realizar un modelado de idioma estadístico para predecir una palabra venidera, dada una secuencia previa de palabras. La RNN 1000 ilustrada 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 previos y una capa de salida 1006 para emitir un resultado. La RNN 1000 opera basándose en escalones de tiempo. El estado de la RNN en un escalón de tiempo dado se ve influenciado basándose en el escalón de tiempo previo mediante el mecanismo de realimentación 1005. Para un escalón de tiempo dado, el estado de las capas ocultas 1004 se define por el estado previo y la entrada en el escalón de tiempo actual. Una entrada inicial (x_1) en un primer escalón de tiempo puede ser procesada por la capa oculta 1004. Una segunda entrada (x_2) puede ser procesada por la capa oculta 1004 usando información de estado que se determina durante el procesamiento de la entrada inicial (x_1). Un estado dado puede calcularse como $s_t = f(Ux_t + Ws_{t-1})$, donde U y W son matrices de parámetros. La función f es, en general, una no linealidad, tal como la función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x) = \max(0, x)$. Sin embargo, la función matemática específica usada en las capas ocultas 1004 puede variar dependiendo de los detalles de implementación específicos de la RNN 1000.

Además de las redes CNN y RNN básicas descritas, 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 pueden aprender dependencias a largo plazo que pueden ser necesarias para procesar secuencias de idioma más largas. Una variante en la CNN es una red de creencia profunda convolucional, que tiene una estructura similar a una CNN y se entrena de una manera similar a una red de creencia profunda. Una red de creencia profunda (DBN) es una red neuronal generativa que está compuesta por 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, la red neuronal se entrena 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 configurarse como una estructura de entrenamiento 604. La estructura de entrenamiento 604 puede engancharse a una red neuronal no entrenada 1106 y posibilitar que la red neuronal no entrenada se entrene usando los recursos de procesamiento paralelo descritos en el presente documento para generar una red neuronal entrenada 1108.

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

El aprendizaje supervisado es un método de aprendizaje en el que se realiza un entrenamiento como una operación mediada, tal como cuando el conjunto de datos de entrenamiento 1102 incluye una entrada emparejada con la salida deseada para la entrada, o donde el conjunto de datos de entrenamiento incluye una entrada que tiene una salida conocida, y la salida de la red neuronal se califica manualmente. La red procesa las entradas y compara las salidas resultantes contra un conjunto de salidas esperadas o deseadas. Los errores se retropropagan, a continuación, a través del sistema. La estructura de entrenamiento 1104 puede ajustarse para 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 repetidamente a medida que se ajustan los pesos de la red para perfeccionar la salida generada por la red neuronal.

El proceso de entrenamiento puede continuar hasta que la red neuronal alcanza una precisión estadísticamente deseada asociada con una red neuronal entrenada 1108. La red neuronal entrenada 1108 puede desplegarse a continuación para implementar cualquier número de operaciones de aprendizaje automático.

5 El aprendizaje no supervisado es un método de aprendizaje en el que la red intenta entrenarse a sí misma usando datos no etiquetados. Por lo tanto, para un aprendizaje no supervisado, el conjunto de datos de entrenamiento 1102 incluirá 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 en cuanto a la reducción de la dimensionalidad de los datos. El entrenamiento no supervisado puede usarse también para realizar una detección de anomalías, lo que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

15 También pueden emplearse variaciones al entrenamiento supervisado y no supervisado. El aprendizaje semisupervisado es una técnica en la que el conjunto de datos de entrenamiento 1102 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que se usan continuamente datos de entrada para entrenar adicionalmente el modelo. El aprendizaje incremental 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 sea supervisado o no supervisado, el proceso de entrenamiento para redes neuronales particularmente profundas puede ser demasiado intensivo desde el punto de vista computacional para un único nodo de cálculo. En lugar de usar un único nodo de cálculo, puede usarse una red distribuida de nodos computacionales para acelerar el proceso de entrenamiento.

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

35 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 ajustar a escala 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 de todas las capas no cabrían en la memoria de un único nodo computacional. En algunas instancias, el paralelismo de modelo puede ser particularmente útil en la ejecución de un entrenamiento no supervisado de redes neuronales grandes.

45 En el paralelismo de datos 1204, los diferentes nodos de la red distribuida tienen una instancia completa del modelo y cada nodo recibe una porción diferente de los datos. Los resultados desde los diferentes nodos se combinan a continuación. Aunque son posibles diferentes enfoques para el paralelismo de datos, los enfoques de entrenamiento de datos paralelos requieren, todos ellos, una técnica de combinación de resultados y de sincronización de los parámetros de modelo entre cada nodo. Los enfoques ilustrativos para la combinación de datos incluyen promediado de parámetros y paralelismo de datos basado en actualizaciones. El promediado de parámetros entrena cada nodo en un subconjunto de los datos de entrenamiento y establece los parámetros globales (por ejemplo, pesos, desvíos) al promedio de los parámetros desde cada nodo. El promediado de parámetros usa un servidor de parámetros central que mantiene los datos de parámetro. El paralelismo de datos basado en actualizaciones es similar al promediado de parámetros excepto

55 que, en lugar de transferir parámetros desde los nodos al servidor de parámetros, se transfieren las actualizaciones al modelo. Adicionalmente, el paralelismo de datos basado en actualizaciones puede realizarse de una manera descentralizada, donde las actualizaciones se comprimen y se transfieren entre nodos.

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

El entrenamiento distribuido ha aumentado la sobrecarga en relación con el entrenamiento en una única máquina. Sin embargo, cada uno de los procesadores paralelos y las GPGPU descritas en el presente documento puede implementar diversas técnicas para reducir la sobrecarga del entrenamiento distribuido, incluyendo técnicas para

posibilitar una transferencia de datos de GPU a GPU de alto ancho de banda y una sincronización de datos remota acelerada.

Aplicaciones de aprendizaje automático ilustrativas

El aprendizaje automático puede aplicarse a resolver una diversidad de problemas tecnológicos, incluyendo, pero sin limitación, visión informática, conducción y navegación autónoma, reconocimiento de habla y procesamiento de idioma. La visión informática ha sido tradicionalmente una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Las aplicaciones de visión informática varían desde la reproducción de capacidades visuales humanas, tales como el reconocimiento de caras, hasta la creación de nuevas categorías de capacidades visuales. Por ejemplo, las aplicaciones de visión informática pueden configurarse para reconocer ondas de sonido de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesador paralelo 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 reconocimiento de señales de carretera y de carril, evitación de obstáculos, navegación y control de conducción. Las técnicas de aprendizaje automático aceleradas pueden usarse para entrenar modelos de conducción basándose en conjuntos de datos que definen las respuestas apropiadas a una entrada de entrenamiento específica. Los procesadores paralelos descritos en el presente documento pueden posibilitar el entrenamiento rápido de las redes neuronales cada vez más complejas usadas para soluciones de conducción autónoma y posibilita el despliegue de procesadores de inferencia de baja potencia en una plataforma móvil adecuada para su integración en vehículos autónomos.

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

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

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

La **Figura 13** ilustra un sistema en un chip (SOC) de 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 la memoria en chip 1305 que puede posibilitar un agrupamiento de datos en chip compartida a la que puede acceder cada uno de los componentes de procesamiento. Los componentes de procesamiento pueden optimizarse para una operación de baja potencia para posibilitar el despliegue en una diversidad de plataformas de aprendizaje automático, incluyendo vehículos autónomos y robots autónomos. Por ejemplo, una implementación del SOC 1300 puede usarse 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 se diseña y está configurado para cumplir con las normas de seguridad funcional relevantes de la jurisdicción de despliegue.

Durante el funcionamiento, el procesador de medios 1302 y el procesador de visión 1304 pueden trabajar conjuntamente para acelerar las operaciones de visión informática. El procesador de medios 1302 puede posibilitar la decodificación de latencia baja 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 analizar, a continuación, el vídeo decodificado y realizar operaciones de procesamiento preliminares sobre los fotogramas del vídeo decodificado como preparación al procesamiento de los fotogramas usando un modelo de reconocimiento de imágenes entrenado. Por ejemplo, el procesador de visión 1304 puede acelerar las operaciones de convolución para una CNN que se usa para realizar un reconocimiento de imágenes sobre los datos de vídeo de alta resolución, mientras que los cálculos de modelo de extremo posterior son realizados por la GPGPU 1306.

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

La GPGPU 1306 puede incluir agrupaciones de cálculo, tal como una configuración de baja potencia de las agrupaciones de cálculo 706A-706H dentro de la unidad de procesamiento de gráficos de propósito general altamente paralela 700. Las agrupaciones de cálculo dentro de la GPGPU 1306 pueden soportar instrucciones que se optimizan específicamente para realizar cálculos de inferencia sobre una red neuronal entrenada. Por ejemplo, la GPGPU 1306 puede soportar instrucciones para realizar cálculos de precisión baja, tales como operaciones vectoriales de números enteros de 8 bits y de 4 bits.

Optimizaciones de cálculo para redes neuronales

Diversas optimizaciones de cálculo son proporcionadas por realizaciones descritas en el presente documento para mejorar la eficiencia de las unidades de procesamiento de gráficos de propósito general cuando se realizan operaciones para el aprendizaje automático a través de redes neuronales. Una realización prevé una instrucción de desplazamiento-acumulación de cañón fusionado. Una realización prevé una indexación de pesos para empaquetado denso de pesos en redes neuronales ponderadas binarias. Una realización proporciona una arquitectura para operaciones de 1 bit x N bits. Una realización prevé una arquitectura de procesamiento para redes neuronales de precisión extremadamente baja.

Operación de desplazamiento-acumulación de cañón fusionado

La investigación sobre redes neuronales actual examina la cuantificación de pesos a valores de potencia de dos (por ejemplo, 2^x) para reducir la operación múltiple para los pesos a un desplazamiento de cañón. Para acelerar tales redes, las realizaciones proporcionadas en el presente documento implementan una instrucción de desplazamiento-acumulación de cañón fusionada, que puede acelerar significativamente las operaciones de procesamiento para tales redes cuando se ejecuta una GPGPU descrita en el presente documento.

Un desplazador de cañón es un bloque de lógica combinacional que acepta un valor de entrada de N bits y proporciona un valor de N bits que es el valor de entrada desplazado P bits a la izquierda o a la derecha. Un desplazador de cañón puede realizar operaciones en un único ciclo que de otro modo requeriría P ciclos para realizarse sin el uso de un desplazador de cañón. Pueden usarse desplazadores de cañón para realizar rápidamente operaciones de multiplicación de potencia de dos. Desplazar a la izquierda P una entrada equivale a multiplicar la entrada por 2^P . Para una red neuronal que usa valores de peso que se cuantifican a valores de 2^x , los cálculos pueden realizarse muy rápidamente a través de una operación de desplazamiento-acumulación de cañón fusionado, que se proporciona mediante realizaciones descritas en el presente documento.

La **Figura 14** ilustra una unidad aritmético-lógica, de acuerdo con una realización. La unidad aritmético-lógica (ALU 1400) está configurada para recibir un primer operando 1401 y un segundo operando 1402. En el procesamiento de redes neuronales, un valor de entrada puede multiplicarse por un valor de peso para generar un valor de resultado. El valor de resultado puede aplicarse a continuación a una función de activación para determinar un valor de salida para un nodo. La ALU 1400 acelera esas operaciones en redes neuronales cuantificadas posibilitando una operación de desplazamiento y acumulación de barril fusionado. En una realización, el primer operando 1401 puede ser un valor de entrada que va a multiplicarse por un valor de peso, mientras que el segundo operando 1402 es un valor de peso que se ha cuantificado como una potencia de dos. Por ejemplo, el primer operando 1401 puede representar datos de característica dentro de una red neuronal. La ALU 1400 puede recibir un código de operación de desplazamiento-acumulación de cañón fusionado 1403, lo que hace que la ALU 1400 realice una operación de desplazamiento-acumulación de cañón fusionado sobre los operandos de entrada. La ALU 1400 puede generar un resultado 1406 de la operación y un estado 1404 de la operación. El estado 1404 puede indicar diversas señales individuales que transmiten información complementaria acerca del resultado de la operación realizada en la ALU 1400. El estado 1404 puede almacenarse en un registro de estado para su uso posterior.

En una realización, el resultado 1406 es el mismo valor que se emitiría a partir de una operación de multiplicación-acumulación fusionada sobre el primer operando y 2^x , donde X = el segundo operando 1402. Sin embargo, la lógica interna de la ALU 1400 se implementa usando un desplazador de cañón en lugar de la lógica de multiplicación convencional y puede realizar el cálculo significativamente más rápido que usando una lógica de multiplicación convencional.

La **Figura 15** ilustra una lógica dentro de una ALU para realizar un desplazamiento-acumulación de cañón fusionado, de acuerdo con una realización. En una realización, una primera fase 1500 incluye un registro de entrada de N bits 1501, un registro de peso cuantificado 1502, un desplazador de cañón 1504 y un registro intermedio 1506. Una segunda fase 1510 incluye un sumador de N bits 1512, un registro de acumulador 1514 y un registro de salida de N bits 1516.

En una realización, el registro de entrada de N bits 1501 almacena un valor de entrada de N bits (por ejemplo, 4, 8, 16, 32, 64), donde el tamaño específico de la entrada varía a lo largo de las realizaciones. El registro de peso cuantificado 1503 puede almacenar un valor de exponente para un peso de red neuronal que se ha cuantificado a una potencia de dos (por ejemplo, 0, 1, 2, 4, 8, 16, 32, 64, etc.). El registro de peso cuantificado 1503 puede proporcionar un valor de desplazamiento para el desplazador de cañón 1504 para indicar la cantidad que hay que desplazar el valor en el registro de entrada de N bits 1501. El desplazador de cañón 1504 puede generar una salida para almacenarse en el registro intermedio 1506. En una realización, el desplazador de cañón puede enviar un resultado al registro intermedio 1506 en un único ciclo y sin realizar una operación de redondeo.

Un sumador de N bits 1512 en la segunda fase 1510 puede leer datos del registro intermedio de N bits 1506 como un primer operando para una operación de suma. El segundo operando de la suma es el valor leído desde un registro de acumulador 1514 que almacena una salida desde un ciclo previo. La salida de la suma puede escribirse entonces en el registro de salida de N bits 1516.

La **Figura 16** es un diagrama de flujo de una lógica 1600 para una instrucción de desplazamiento-acumulación de cañón fusionado, de acuerdo con una realización. La lógica 1600 para la instrucción ilustrada, en una realización, se realiza dentro de una unidad de cálculo de una unidad de procesamiento de gráficos de propósito general como se describe en el presente documento para acelerar las operaciones de aprendizaje automático y de redes neuronales. En una realización, la lógica 1600 decodifica una única instrucción que especifica múltiples operandos que incluyen un valor de entrada y un valor de peso de una red neuronal, como se muestra en el bloque 1602. La lógica 1600 puede emitir entonces la única instrucción para su ejecución dentro de una unidad de cálculo de una unidad de procesamiento de gráficos de propósito general, como se muestra en el bloque 1604. En respuesta a la ejecución de la única instrucción, la lógica 1600 puede generar un resultado basándose en desplazar el valor de entrada por el valor de peso de la red neuronal y sumar el valor desplazado a un valor almacenado en un registro de acumulación, como se muestra en el bloque 1606.

Indexación de pesos para empaquetamiento denso de pesos en redes neuronales ponderadas binarias

Las implementaciones de aprendizaje automático pueden implementar redes neuronales que hacen uso de datos de característica de coma flotante o de coma fija de alta precisión en combinación con pesos binarios. Estas redes pueden usarse para posibilitar una alta precisión mientras se usa una lógica computacional de precisión inferior. Sin embargo, tales implementaciones no pueden usar instrucciones de manipulación de bits para posibilitar un uso eficiente de pesos binarios debido al uso de datos de característica de coma flotante o de coma fija. Para empaquetar densamente y acceder eficientemente a los pesos, una indexación a nivel de bits de palabras dentro de un registro puede implementarse dentro de una lógica de GPGPU descrita en el presente documento. Tal implementación puede usarse en combinación con un soporte de hardware para operaciones binarias de multiplicación-acumulación.

La **Figura 17** ilustra cálculos para una red neuronal ponderada binaria que tiene características de N bits, de acuerdo con una realización. Algunos modelos de redes neuronales usan pesos de baja precisión en conjunto con datos de característica de N bits. Los datos de característica de N bits pueden ser una potencia de dos o un número arbitrario de bits, mientras que los datos de peso se representan como un único bit. Las redes de peso binario han encontrado uso en implementaciones de inferencia, donde valores de precisión inferiores pueden producir resultados similares en relación con valores de precisión superiores. Una red neuronal entrenada usando pesos de N bits puede binarizar esos pesos para realizar inferencias, sustituyendo el valor de peso con uno de dos valores posibles basándose en si el valor de peso es mayor que o igual a algún umbral.

Para posibilitar un almacenamiento eficiente de pesos empaquetados densamente, múltiples pesos binarios 1702 se empaquetan en un único registro de N bits. La indexación de bits se usa para hacer referencia a pesos individuales para un procesamiento simultáneo a lo largo de múltiples características de N bits 1704A-1704N. Múltiples operaciones de multiplicación-acumulación binarias fusionadas paralelas 1706A-1706N pueden realizarse en paralelo dentro de las unidades de cálculo del procesador de propósito general. El operando de peso introducido en las operaciones de multiplicación-acumulación binaria 1706A-1706N incluye un registro y un índice a una ubicación dentro del registro para el valor de peso que usar para el cálculo. Para un registro de N bits, pueden almacenarse N pesos binarios. En una realización, las unidades de cálculo de la GPGPU proporcionan soporte a una instrucción de multiplicación-acumulación binaria de vectores en la que los N bits binarios dentro del registro de N bits pueden multiplicarse por el N número de las características de N bits 1704A-1704N, que pueden almacenarse en un registro vectorial de anchura NxN. Por ejemplo, y en una realización, se proporciona una única instrucción en la que un primer operando es un registro de entrada que almacena ocho pesos de 1 bit, donde cada peso se almacena en una ubicación de índice [0:7] dentro del registro. Ocho valores de característica de 8 bits pueden empaquetarse en un segundo registro. Cada uno de los ocho pesos de 1 bit se multiplicará por un valor de característica de 8 bits asociado y los productos se

acumularán para generar un valor de salida de M bits. En diversas realizaciones, la salida de M bits para un producto de 8 bits por 1 bit puede ser un valor de 8 bits con saturación, o un valor mayor que uno de 8 bits, tal como un valor de 16 bits. Para un N de 8, el registro de acumulador usado para almacenar el valor acumulado será mayor que 8 bits.

- 5 La binarización puede realizarse de forma determinista o transformar estocásticamente pesos de N bits en los valores binarios bipolares 1701 que tienen unos valores bipolares (-1, 1). La binarización determinista se muestra en la ecuación (1). La binarización estocástica se muestra en la ecuación (2).

$$10 \quad x^b = \text{Signo}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & \text{si no} \end{cases} \quad (1)$$

$$x^b = \text{Signo}(x) = \begin{cases} +1 & \text{probabilidad} = \sigma(x) \\ -1 & \text{probabilidad} = 1 - \sigma(x) \end{cases} \quad (2)$$

En la ecuación (2), $\sigma(x)$ es la función sigmoidea rígida mostrada en la ecuación (3).

$$15 \quad \sigma(x) = \max(0, \min(1, \frac{x+1}{2})) \quad (3)$$

La **Figura 18** ilustra una lógica dentro de una ALU para realizar una operación de multiplicación-acumulación de N bits por 1 bit fusionada, de acuerdo con una realización. En una realización, una primera fase 1800 incluye un registro de N bits que almacena la entrada de característica 1801, un peso indexado 1802 dentro de un registro de peso empaquetado, un multiplicador 1804 y un registro intermedio 1806. La segunda fase 1810 incluye un sumador de N bits 1812, un registro de acumulador 1814 y un registro de salida de M bits 1816.

Si el peso indexado 1802 tiene un valor de peso bipolar de uno (0b1), el valor de la entrada de característica de N bits puede pasarse al otro lado sin modificación. El multiplicador 1804 incluye una unidad de inversión de signo 1805 para realizar una multiplicación por un valor de peso bipolar de uno negativo (-1), que se representa por el valor de bit 0b0. No puede usarse un multiplicador convencional, debido a que el valor resultante sería cero. En su lugar, la unidad de inversión de signo 1805 se usa para realizar la operación de multiplicación sobre la entrada de característica de n bits 1801. El valor puede almacenarse en el registro intermedio 1806 para usarlo como una entrada a un sumador de M bits 1812, que suma el producto a un valor en el registro de acumulador 1814. El valor en el registro de acumulador puede actualizarse con el resultado de la suma y el valor resultante puede emitirse a través de un registro de salida de M bits 1816.

Aunque se ilustran valores de peso binario bipolar, una realización soporta pesos tanto binarios bipolares como ternarios. Con pesos ternarios, pueden usarse dos bits para representar los valores (-1, 0, +1). El mapeo entre valores binarios y valores ternarios puede variar, y las realizaciones pueden configurarse para usar selectivamente diferentes mapeos. Por ejemplo, los valores binarios bipolares y ternarios pueden mapearse como se muestra en la Tabla 5 a continuación.

40 **Tabla 5** - Mapeo de pesos ilustrativo

Valor máquina	Valor binario	Valor ternario
0b00	-1	-1
0b01	1	1
0b1X	N/A	0

45 También pueden implementarse mapeos alternativos entre valores binarios y ternarios. Por ejemplo, una realización puede usar un mapeo como se muestra en la Tabla 6.

Tabla 6 - Mapeo de pesos ilustrativo adicional

Valor máquina	Valor binario	Valor ternario
0b00	-1	0
0b01	1	1
0b11	N/A	-1
0b10	N/A	0 o N/A

En la configuración de la Tabla 6, el bit [1] del valor máquina es un bit de signo, de manera que 0b11 representa (-1). En una realización, pueden soportarse pesos cuaternarios en la configuración que se muestra en la Tabla 6 mapeando 0b10 a un cuarto valor usado por una red neuronal de pesos cuaternarios.

La **Figura 19** ilustra multiplicadores ilustrativos, de acuerdo con la invención. Se proporciona un multiplicador binario 1910 que está configurado para posibilitar la multiplicación de 1 bit por N bits, con pesos bipolares binarios de 1 bit. El peso indexado 1802 se acopla con una puerta NO 1902, de manera que un valor de peso de 0b0 (-1) activa la unidad de inversión de signo 1805 para invertir el signo de la entrada de característica de N bits 1801. La naturaleza de la inversión de signo varía basándose en el tipo de datos de la entrada de característica de N bits. Para las representaciones de números enteros y de coma fija, la unidad de inversión de signo 1805 realiza una operación de complemento a 2 para invertir el signo de la característica de entrada. Para las representaciones de coma flotante, se invierte el bit de signo del valor. Como una alternativa al uso de la puerta NO 1902, en una realización, la unidad de inversión de signo 1805 está configurada con una entrada activa a valor bajo.

Una realización proporciona un multiplicador ternario 1920 para posibilitar la multiplicación frente a un peso indexado ternario. El multiplicador ternario 1920 incluye un multiplexor 1904 y una unidad de inversión de signo 1805. En una realización, el multiplexor 1904 acepta la entrada de característica de N bits en una entrada y una segunda entrada está vinculada a cero. El multiplicador ternario 1920 es controlado por los dos bits del valor ternario. En la realización ilustrada, el bit cero [0] del peso indexado 1802A representa el valor de peso y el bit uno [1] 1802B es un bit de signo. Durante la operación, el bit de peso indexado [0] 1802A determina si se proporciona el valor de la entrada de característica de N bits 1801 o un valor de cero, a través del multiplexor 1904 a la unidad de inversión de signo 1805. Un valor de bit de peso indexado [0] 1802 de 0b1 puede pasar el valor de característica de N bits, mientras que un valor de 0b0 pasará una entrada de cero. El bit de peso indexado [1] 1802B determina si la unidad de inversión de signo 1805 realiza una inversión de signo o pasa el valor de entrada sin modificación (por ejemplo, inversión de bit de signo o complemento a 2). Como una alternativa al multiplicador ternario 1920 ilustrado, en una realización puede implementarse una lógica de multiplicación-suma fusionada en la que el valor de cero se maneja usando el bit de peso indexado [0] como bit de habilitación para el acumulador. En tal realización, cuando se recibe un valor de entrada de cero, el valor de la entrada de característica de N bits 1801 no se añade al registro de acumulador.

La **Figura 20** ilustra un diagrama de flujo de una lógica 2000 para una instrucción de desplazamiento-acumulación de cañón fusionado, de acuerdo con una realización. La lógica 2000 para la instrucción ilustrada, en una realización, se realiza dentro de una unidad de cálculo de una unidad de procesamiento de gráficos de propósito general como se describe en el presente documento para acelerar las operaciones de aprendizaje automático y de redes neuronales. En una realización, la lógica 2000 decodifica una única instrucción que especifica múltiples operandos que incluyen un valor de entrada y una referencia a un valor de peso binario o ternario de una red neuronal, como se muestra en el bloque 2002. La referencia al valor de peso binario o ternario de la red neuronal, en una realización, es un registro de entrada y un índice a una ubicación dentro del registro de entrada. La ubicación indexada puede ser un único bit para un peso binario o dos bits para un peso ternario. La lógica 2000 puede emitir entonces la única instrucción para su ejecución dentro de una unidad de cálculo de una unidad de procesamiento de gráficos de propósito general, como se muestra en el bloque 2004. En respuesta a la ejecución de la única instrucción, la lógica 2000 puede generar un resultado basándose en un producto del valor de entrada de característica y el peso.

Arquitectura de procesamiento para redes neuronales de precisión extremadamente baja.

Además de los pesos de 1 bit con características de N bits, pueden implementarse redes neuronales totalmente binarias en las que datos tanto de peso como de característica se almacenan como valores binarios. Una binarización determinista o estocástica a través de la ecuación (1) o la ecuación (2) anteriores puede usarse para binarizar datos de peso y de característica. En algunas instancias, las redes neuronales totalmente binarias pueden lograr una precisión de inferencia similar a la de redes de precisión superior con una reducción significativa en los requisitos de almacenamiento de memoria y ancho de banda y en la complejidad computacional. Específicamente, el producto escalar para redes neuronales binarias puede realizarse mediante una operación de XNOR y recuento de población. En la lógica de procesamiento de gráficos convencional, las operaciones de XNOR y recuento de población son operaciones de separación. Adicionalmente, no todos los elementos de procesamiento o unidades de cálculo pueden soportar la función de recuento de población. Para acelerar las operaciones de redes neuronales binarias, las realizaciones descritas en el presente documento proporcionan una función de XNOR y recuento de población fusionado para posibilitar una convolución eficiente de caudal alto para redes neuronales binarias.

La **Figura 21** ilustra una lógica para realizar una operación de XNOR y recuento de población fusionado, de acuerdo con una realización. En una realización, la lógica de XNOR y recuento de población fusionado se incluye en una primera fase 2100 de una unidad de multiplicación-acumulación de entrada binaria y de salida de M bits dentro binaria de un elemento de procesamiento de una GPGPU como se describe en el presente documento. La primera fase 2100 incluye una entrada de característica de 1 bit 2101 y una entrada de peso de 1 bit 2102, donde la entrada de característica y la entrada de peso se han binarizado a valores binarios bipolares de (-1, 1) con -1 representado como 0b0. La lógica de XNOR y recuento de población fusionado se incluye dentro de una unidad de XNOR y recuento de

población fusionado 2104 que incluye una unidad de XNOR 2103 y una unidad de recuento de población 2105. La unidad de XNOR y recuento de población fusionado 2104 emite a un registro intermedio 2106, que sirve como una entrada a la segunda fase 1810, que se ha descrito anteriormente en la Figura 18. En una realización, la unidad de XNOR y recuento de población fusionado 2104 se incluye dentro de todos los elementos de procesamiento con la GPGPU. En una realización, solo un subconjunto de los elementos de procesamiento dentro de la GPGPU incluye la unidad de XNOR y recuento de población fusionado 2104.

Sistema de procesamiento de gráficos ilustrativo adicional

Los detalles de las realizaciones descritas anteriormente pueden incorporarse dentro de los sistemas y dispositivos de procesamiento de gráficos descritos a continuación. Los dispositivos y el sistema de procesamiento de gráficos de las Figuras 22-35 ilustran hardware de procesamiento de gráficos y sistemas alternativos que pueden implementar todas y cada una de las técnicas descritas anteriormente.

Los detalles de las realizaciones descritas anteriormente pueden incorporarse dentro de los sistemas y dispositivos de procesamiento de gráficos descritos a continuación. Los dispositivos y el sistema de procesamiento de gráficos de las Figuras 24-37 ilustran hardware de procesamiento de gráficos y sistemas alternativos que pueden implementar todas y cada una de las técnicas descritas anteriormente.

La **Figura 22** es un diagrama de bloques de un sistema de procesamiento 2200, de acuerdo con una realización. En diversas realizaciones, el sistema 2200 incluye uno o más procesadores 2202 y uno o más procesadores de gráficos 2208, 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 2202 o núcleos de procesador 2207. En una realización, el sistema 2200 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 2200 puede incluir o incorporarse dentro de una plataforma de juegos basada en servidor, una consola de juegos, incluyendo 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 2200 es un teléfono móvil, un teléfono inteligente, un dispositivo informático de tipo tableta o un dispositivo de Internet móvil. El sistema de procesamiento de datos 2200 también puede incluir, acoplarse o estar integrado dentro de un dispositivo portátil, tal como un dispositivo ponible tipo reloj inteligente, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 2200 es un televisor o dispositivo decodificador que tiene uno o más procesadores 2202 y una interfaz gráfica generada por uno o más procesadores de gráficos 2208.

En algunas realizaciones, cada uno de los uno o más procesadores 2202 incluye uno o más núcleos de procesador 2207 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para software de usuario y sistema. En algunas realizaciones, cada uno de los uno o más núcleos de procesador 2207 está configurado para procesar un conjunto de instrucciones 2209 específico. En algunas realizaciones, el conjunto de instrucciones 2209 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). Múltiples núcleos de procesador 2207 pueden procesar, cada uno, un conjunto de instrucciones 2209 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 2207 también puede incluir otros dispositivos de procesamiento, tales como un procesador de señales digitales (DSP).

En algunas realizaciones, el procesador 2202 incluye la memoria caché 2204. Dependiendo de la arquitectura, el procesador 2202 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 2202. En algunas realizaciones, el procesador 2202 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 2207 usando técnicas de coherencia de caché conocidas. Se incluye adicionalmente un archivo de registro 2206 en el procesador 2202 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de coma flotante, registros de estado y un registro de puntero de instrucción). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos del diseño del procesador 2202.

En algunas realizaciones, el procesador 2202 está acoplado con un bus de procesador 2210 para transmitir señales de comunicación tales como señales de dirección, de datos o de control entre el procesador 2202 y otros componentes en el sistema 2200. En una realización, el sistema 2200 usa una arquitectura de sistema de 'concentrador' ilustrativa, incluyendo un concentrador de controlador de memoria 2216 y un concentrador de controlador de entrada-salida (E/S) 2230. Un concentrador de controlador de memoria 2216 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 2200, mientras que un concentrador de controlador de E/S (ICH) 2230 proporciona conexiones a dispositivos de E/S mediante un bus de E/S local. En una realización, la lógica del concentrador de controlador de memoria 2216 está integrada dentro del procesador.

El dispositivo de memoria 2220 puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria de acceso aleatorio estática (SRAM), un dispositivo de memoria flash, un dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para servir como memoria de proceso. En una realización, el dispositivo de memoria 2220 puede operar como memoria de sistema para el sistema 2200, para almacenar datos 2222 e instrucciones 2221 para su uso cuando los uno o más procesadores 2202 ejecutan una aplicación o proceso. El concentrador de controlador de memoria 2216 también se acopla con un procesador de gráficos externo 2212 opcional, que puede comunicarse con los uno o más procesadores de gráficos 2208 en los procesadores 2202 para realizar operaciones de gráficos y de medios.

En algunas realizaciones, el ICH 2230 posibilita que los periféricos se conecten al dispositivo de memoria 2220 y al procesador 2202 mediante un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, pero sin limitación, un controlador de audio 2246, una interfaz de firmware 2228, un transceptor inalámbrico 2226 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 2224 (por ejemplo, unidad de disco duro, memoria flash, etc.), y un controlador de E/S heredado 2240 para acoplar dispositivos heredados (por ejemplo, dispositivos de sistema personal 2 (PS/2)) al sistema. Uno o más controladores de bus serie universal (USB) 2242 conectan dispositivos de entrada, tales como combinaciones de teclado y ratón 2244. Un controlador de red 2234 también puede acoplarse con el ICH 2230. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de procesador 2210. Se apreciará que el sistema 2200 mostrado es ilustrativo y no limitante, debido a que también 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 2230 puede integrarse dentro de los uno o más procesadores 2202, o el concentrador de controlador de memoria 2216 y el concentrador de controlador de E/S 2230 pueden integrarse en un procesador de gráficos externo discreto, tal como el procesador de gráficos externo 2212.

La **Figura 23** es un diagrama de bloques de un procesador 2300 que tiene uno o más núcleos de procesador 2302A-2302N, un controlador de memoria integrado 2314 y un procesador de gráficos integrado 2308. Aquellos elementos de la **Figura 23** 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 2300 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 2302N representado por los recuadros de línea discontinua. Cada uno de los núcleos de procesador 2302A-2302N incluye una o más unidades de caché internas 2304A-2304N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades almacenadas en caché compartidas 2306.

Las unidades de caché internas 2304A-2304N y las unidades de caché compartidas 2306 representan una jerarquía de memoria caché dentro del procesador 2300. La jerarquía de memoria caché puede incluir al menos un nivel de caché de instrucciones y de datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel medio compartida, tal como una caché de Nivel 2 (L2), de Nivel 3 (L3), de Nivel 4 (L4) o de otros niveles, donde el nivel más alto de caché antes de la memoria externa se clasifica como LLC. En algunas realizaciones, la lógica de coherencia de caché mantiene la coherencia entre las diversas unidades de caché 2306 y 2304A-2304N.

En algunas realizaciones, el procesador 2300 también puede incluir un conjunto de una o más unidades de controlador de bus 2316 y un núcleo de agente de sistema 2310. Las una o más unidades de controlador de bus 2316 gestionan un conjunto de buses de periféricos, tales como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 2310 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunas realizaciones, el núcleo de agente de sistema 2310 incluye uno o más controladores de memoria integrados 2314 para gestionar el acceso a diversos dispositivos de memoria externos (no mostrados).

En algunas realizaciones, uno o más de los núcleos de procesador 2302A-2302N incluyen el soporte para múltiples hilos simultáneos. En una realización de este tipo, el núcleo de agente de sistema 2310 incluye componentes para coordinar y operar los núcleos 2302A-2302N durante el procesamiento de múltiples hilos. El núcleo de agente de sistema 2310 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 2302A-2302N y el procesador de gráficos 2308.

En algunas realizaciones, el procesador 2300 incluye adicionalmente el procesador de gráficos 2308 para ejecutar las operaciones de procesamiento de gráficos. En algunas realizaciones, el procesador de gráficos 2308 se acopla con el conjunto de unidades de caché compartidas 2306 y el núcleo de agente de sistema 2310, incluyendo los uno o más controladores de memoria integrados 2314. En algunas realizaciones, un controlador de visualización 2311 está acoplado con el procesador de gráficos 2308 para controlar la salida de procesador de gráficos a una o más pantallas acopladas. En algunas realizaciones, el controlador de visualización 2311 puede ser un módulo separado acoplado con el procesador de gráficos a través de al menos una interconexión, o puede estar integrado dentro del procesador de gráficos 2308 o el núcleo de agente de sistema 2310.

En algunas realizaciones, se usa una unidad de interconexión basada en anillo 2312 para acoplar los componentes internos del procesador 2300. Sin embargo, puede usarse una unidad de interconexión alternativa, tal como una interconexión de punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en

la técnica. En algunas realizaciones, el procesador de gráficos 2308 se acopla con la interconexión en anillo 2312 a través de un enlace de E/S 2313.

El enlace de E/S 2313 ilustrativo representa al menos una de múltiples diversidades de interconexiones de E/S, incluyendo una interconexión de E/S en paquete que facilita la comunicación entre diversos componentes de procesador y un módulo de memoria integrado de alto rendimiento 2318, tal como un módulo de eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 2302A-2302N y el procesador de gráficos 2308 usan módulos de memoria integrados 2318 como una caché de último nivel compartida.

En algunas realizaciones, los núcleos de procesador 2302A-2302N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 2302A-2302N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 2302A-2302N 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 2302A-2302N son heterogéneos en términos de microarquitectura, donde uno o más núcleos que tienen un consumo de energía relativamente superior se acoplan con uno o más núcleos de potencia que tienen un consumo de energía inferior. Adicionalmente, el procesador 2300 puede implementarse en uno o más chips o como un circuito integrado de SoC que tiene los componentes ilustrados, además de otros componentes.

La **Figura 24** es un diagrama de bloques de un procesador de gráficos 2400, 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 por medio de una interfaz de E/S mapeada en memoria con registros del procesador de gráficos y con los comandos almacenados en la memoria de procesador. En algunas realizaciones, el procesador de gráficos 2400 incluye una interfaz de memoria 2414 para acceder a memoria. La interfaz de memoria 2414 puede ser una interfaz a una memoria local, una o más cachés internas, una o más cachés externas compartidas y/o a una memoria de sistema.

En algunas realizaciones, el procesador de gráficos 2400 también incluye un controlador de visualización 2402 para enviar datos de salida de visualización a un dispositivo de visualización 2420. El controlador de visualización 2402 incluye hardware para uno o más planos de superposición para la visualización y composición de múltiples capas de elementos de interfaz de usuario o de vídeo. En algunas realizaciones, el procesador de gráficos 2400 incluye un motor de códec de vídeo 2406 para codificar, decodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluyendo, pero sin limitación, formatos del Grupo de Expertos en Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como de la Sociedad de Ingenieros de Imágenes en Movimiento y de Televisión (SMPTE) 421M/VC-1 y formatos del Grupo Conjunto de Expertos en Fotografía (JPEG) tales como los formatos JPEG y Motion JPEG (MJPEG).

En algunas realizaciones, el procesador de gráficos 2400 incluye un motor de transferencia de imágenes en bloque (BLIT) 2404 para realizar operaciones de rasterizador bidimensionales (2D), incluyendo, por ejemplo, transferencias de bloque de frontera de bits. Sin embargo, en una realización, se realizan operaciones de gráficos 2D usando uno o más componentes del motor de procesamiento de gráficos (GPE) 2410. En algunas realizaciones, el GPE 2410 es un motor de cálculo para realizar operaciones de gráficos, incluyendo operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 310 incluye una canalización 3D 2412 para realizar operaciones 3D, tales como representar imágenes y escenas tridimensionales usando funciones de procesamiento que actúan sobre formas de primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización 3D 2412 incluye elementos de función programable y fija que realizan diversas tareas dentro del elemento y/o generan hilos de ejecución en un subsistema 3D/de medios 2415. Aunque la canalización 3D 2412 puede usarse para realizar operaciones de medios, una realización del GPE 2410 también incluye una canalización de medios 2416 que se usa específicamente para realizar operaciones de medios, tales como post-procesamiento de vídeo y potenciación de imagen.

En algunas realizaciones, la canalización de medios 2416 incluye unidades de lógica programable o de función fija para realizar una o más operaciones de medios especializadas, tales como aceleración de descodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar o en nombre del motor de códec de vídeo 2406. En algunas realizaciones, la canalización de medios 2416 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema 3D/de medios 2415. 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 2415.

En algunas realizaciones, el subsistema 3D/de medios 2415 incluye una lógica para ejecutar hilos generados por la canalización 3D 2412 y la canalización de medios 2416. En una realización, las canalizaciones envían solicitudes de ejecución de hilos al subsistema 3D/de medios 2415, que incluye lógica de despacho de hilo para arbitrar y despachar las diversas solicitudes a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos 3D y de medios. En algunas realizaciones, el subsistema 3D/de medios 2415 incluye una o más cachés internas para datos e instrucciones 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 25** es un diagrama de bloques de un motor de procesamiento de gráficos 2510 de un procesador de gráficos de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento de gráficos (GPE) 2510 es una versión del GPE 2410 mostrado en la **Figura 24**. 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. Por ejemplo, se ilustra la canalización 3D 2412 y la canalización de medios 2416 de la **Figura 24**. La canalización de medios 2416 es opcional en algunas realizaciones del GPE 2510 y puede no incluirse explícitamente dentro del GPE 2510. Por ejemplo, y en al menos una realización, un procesador de medios y/o de imágenes separado se acopla al GPE 2510.

En algunas realizaciones, el GPE 2510 se acopla con o incluye un transmisor por flujo continuo de comandos 2503, que proporciona un flujo de comandos a la canalización 3D 2412 y/o a las canalizaciones de medios 2416. En algunas realizaciones, el transmisor por flujo continuo de comandos 2503 está acoplado con memoria, que puede ser memoria de sistema, o una o más de memoria caché interna y memoria caché compartida. En algunas realizaciones, el transmisor por flujo continuo de comandos 2503 recibe comandos desde la memoria y envía los comandos a la canalización 3D 2412 y/o a la canalización de medios 2416. Los comandos son directivas extraídas de una memoria intermedia en anillo, que almacena comandos para la canalización 3D 2412 y la canalización de medios 2416. En una realización, la memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización 3D 2412 también pueden incluir referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértice y de geometría para la canalización 3D 2412 y/o datos de imagen y objetos de memoria para la canalización de medios 2416. La canalización 3D 2412 y la canalización de medios 2416 procesan los comandos y datos realizando operaciones mediante lógica dentro de las canalizaciones respectivas o despachando uno o más hilos de ejecución a la matriz de núcleos de gráficos 2514.

En diversos ejemplos, la canalización 3D 2412 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 núcleos de gráficos 2514. La matriz de núcleos de gráficos 2514 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución de múltiples propósitos (por ejemplo, unidades de ejecución) dentro de la matriz de núcleos de gráficos 2514 incluye un soporte para diversos lenguajes de sombreador de API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunas realizaciones, la matriz de núcleos de gráficos 2514 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 una lógica de propósito general que es programable para realizar operaciones computacionales de propósito general paralelas, además de operaciones de procesamiento de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o en conjunto con la lógica de propósito general dentro del/de los núcleo(s) de procesador 2207 de la **Figura 22** o el núcleo 2302A-2302N como en la **Figura 23**.

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleos de gráficos 2514 pueden proporcionar datos a memoria en una memoria intermedia de retorno unificado (URB) 2518. La URB 2518 puede almacenar datos para múltiples hilos. En algunas realizaciones, la URB 2518 puede usarse para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleos de gráficos 2514. En algunas realizaciones, la URB 2518 puede usarse adicionalmente para la sincronización entre hilos en la matriz de núcleos de gráficos y la lógica de función fija dentro de la lógica de funciones compartidas 2520.

En algunas realizaciones, la matriz de núcleos de gráficos 2514 es ajustable a escala, de manera que la matriz incluye un número variable de núcleos de gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y en el nivel de rendimiento del GPE 2510. En una realización, los recursos de ejecución son dinámicamente ajustables a escala, de manera que los recursos de ejecución pueden habilitarse o deshabilitarse según sea necesario.

La matriz de núcleos de gráficos 2514 se acopla con la lógica de funciones compartidas 2520 que incluye múltiples recursos que se comparten entre los núcleos de gráficos en la matriz de núcleos de gráficos. Las funciones compartidas dentro de la lógica de funciones compartidas 2520 son unidades de lógica de hardware que proporcionan una funcionalidad complementaria especializada a la matriz de núcleos de gráficos 2514. En diversas realizaciones, la lógica de funciones compartidas 2520 incluye, pero sin limitación, la lógica del muestreador 2521, del cálculo matemático 2522 y de la comunicación entre hilos (ITC) 2523. Adicionalmente, algunas realizaciones implementan una o más cachés 2525 dentro de la lógica de funciones compartidas 2520. Se implementa una función compartida

donde la demanda de una función especializada dada es insuficiente para su inclusión dentro de la matriz de núcleos de gráficos 2514. En su lugar, una única instanciación de esa función especializada se implementa como una entidad autónoma en la lógica de funciones compartidas 2520 y se comparte entre los recursos de ejecución dentro de la matriz de núcleos de gráficos 2514. El conjunto preciso de funciones que se comparten entre la matriz de núcleos de gráficos 2514 y se incluyen dentro de la matriz de núcleos de gráficos 2514 varía entre realizaciones.

La **Figura 26** es un diagrama de bloques de un procesador de gráficos 2600 proporcionado por una realización adicional. Los elementos de la **Figura 26** 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 2600 incluye una interconexión en anillo 2602, un extremo frontal de canalización 2604, un motor de medios 2637 y unos núcleos de gráficos 2680A-2680N. En algunas realizaciones, la interconexión en anillo 2602 acopla el procesador de gráficos a otras unidades de procesamiento, que incluyen otros procesadores de gráficos o uno o más núcleos de procesadores de propósito general. En 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 2600 recibe lotes de comandos mediante la interconexión en anillo 2602. Los comandos entrantes son interpretados por un transmisor por flujo continuo de comandos 2603 en el extremo frontal de canalización 2604. En algunas realizaciones, el procesador de gráficos 2600 incluye una lógica de ejecución ajustable a escala para realizar procesamiento de geometría 3D y procesamiento de medios a través del/de los núcleo(s) de gráficos 2680A-2680N. Para los comandos de procesamiento de geometría 3D, el transmisor por flujo continuo de comandos 2603 suministra comandos a la canalización de geometría 2636. Para al menos algunos comandos de procesamiento de medios, el transmisor por flujo continuo de comandos 2603 suministra los comandos a un extremo frontal de vídeo 2634, que se acopla con un motor de medios 2637. En algunas realizaciones, el motor de medios 2637 incluye un motor de calidad de vídeo (VQE) 2630 para post-procesamiento de vídeo y de imagen y un motor de codificación/decodificación de múltiples formatos (MFX) 2633 para proporcionar codificación y decodificación de datos de medios acelerados por hardware. En algunas realizaciones, cada uno de la canalización de geometría 2636 y el motor de medios 2637 generan hilos de ejecución para los recursos de ejecución de hilos proporcionados por al menos un núcleo de gráficos 2680A.

En algunas realizaciones, el procesador de gráficos 2600 incluye recursos de ejecución de hilos ajustables a escala que cuentan con los núcleos modulares 2680A-2680N (denominados, en ocasiones, segmentos de núcleo), teniendo cada uno múltiples subnúcleos 2650A-550N, 2660A-2660N (denominados, en ocasiones, subsegmentos de núcleo). En algunas realizaciones, el procesador de gráficos 2600 puede tener cualquier número de núcleos de gráficos 2680A a 2680N. En algunas realizaciones, el procesador de gráficos 2600 incluye un núcleo de gráficos 2680A que tiene al menos un primer subnúcleo 2650A y un segundo subnúcleo 2660A. En otras realizaciones, el procesador de gráficos es un procesador de baja potencia con un único subnúcleo (por ejemplo, 2650A). En algunas realizaciones, el procesador de gráficos 2600 incluye múltiples núcleos de gráficos 2680A-2680N, incluyendo cada uno un conjunto de primeros subnúcleos 2650A-2650N y un conjunto de segundos subnúcleos 2660A-2660N. Cada subnúcleo en el conjunto de primeros subnúcleos 2650A-2650N incluye al menos un primer conjunto de unidades de ejecución 2652A-2652N y muestreadores de medios/texturas 2654A-2654N. Cada subnúcleo en el conjunto de segundos subnúcleos 2660A-2660N incluye al menos un segundo conjunto de unidades de ejecución 2662A-2662N y muestreadores 2664A-2664N. En algunas realizaciones, cada subnúcleo 2650A-2650N, 2660A-2660N comparte un conjunto de recursos compartidos 2670A-2670N. En algunas realizaciones, los recursos compartidos incluyen memoria caché compartida y lógica de operación de píxeles. También pueden incluirse otros recursos compartidos en las diversas realizaciones del procesador de gráficos.

Unidades de ejecución ilustrativas adicionales

La **Figura 27** ilustra una lógica de ejecución de hilos 2700 que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones. Aquellos 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.

En algunas realizaciones, la lógica de ejecución de hilos 2700 incluye un procesador de sombreador 2702, un despachador de hilos 2704, una caché de instrucciones 2706, una matriz de unidades de ejecución ajustable a escala que incluye una pluralidad de unidades de ejecución 2708A-2708N, un muestreador 2710, una caché de datos 2712 y un puerto de datos 2714. En una realización, la matriz de unidades de ejecución ajustable a escala puede realizar un ajuste a escala dinámico habilitando o deshabilitando una o más unidades de ejecución (por ejemplo, cualquiera de las unidades de ejecución 2708A, 2708B, 2708C, 2708D a 2708N-1 y 2708N) basándose en los requisitos computacionales 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 hilos 2700 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 2706, el puerto de datos 2714, el muestreador 2710 y las

unidades de ejecución 2708A-2708N. En algunas realizaciones, cada unidad de ejecución (por ejemplo, 2708A) es una unidad computacional de propósito general programable autónoma que es capaz de ejecutar múltiples hilos de hardware simultáneos mientras se procesan múltiples elementos de datos en paralelo para cada hilo. En diversas realizaciones, la matriz de unidades de ejecución 2708A-2708N es ajustable a escala para incluir cualquier número de unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 2708A-2708N se usan principalmente para ejecutar programas sombreadores. Un procesador de sombreador 2702 puede procesar los diversos programas sombreadores y despachar los hilos de ejecución asociados con los programas sombreadores mediante un despachador de hilos 2704. En una realización, el despachador de hilos incluye lógica para arbitrar las solicitudes de iniciación de hilo desde las canalizaciones de gráficos y de medios e instanciar los hilos solicitados en una o más unidades de ejecución en las unidades de ejecución 2708A-2708N. Por ejemplo, la canalización de geometría (por ejemplo, 2636 de la **Figura 26**) puede despachar sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilos 2700 (la **Figura 27**) para su procesamiento. En algunas realizaciones, el despachador de hilos 2704 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas sombreadores en ejecución.

En algunas realizaciones, las unidades de ejecución 2708A-2708N soportan un conjunto de instrucciones que incluye un soporte nativo para muchas instrucciones de sombreador de gráficos 3D convencionales, de manera que los programas sombreadores desde bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución soportan procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y procesamiento de propósito general (por ejemplo, sombreadores de cálculo y de medios). Cada una de las unidades de ejecución 2708A-2708N es capaz de múltiples emisiones de ejecución de múltiples datos de única instrucción (SIMD), y una operación de múltiples hilos posibilita un entorno de ejecución eficiente frente a accesos de memoria de latencia superior. Cada hilo de hardware dentro de cada unidad de ejecución tiene un archivo de registro de alto ancho de banda dedicado y un estado de hilo independiente asociado. La ejecución es de múltiples emisiones por reloj a canalizaciones capaces de realizar operaciones de números enteros, de coma flotante de precisión sencilla y doble, capacidad de bifurcación de SIMD, operaciones lógicas, operaciones trascendentales y otras operaciones misceláneas. Mientras se esperan datos desde memoria o una de las funciones compartidas, una lógica de dependencia dentro de las unidades de ejecución 2708A-2708N hace que un hilo en espera pase a estar inactivo hasta que se hayan devuelto 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, un sombreador de fragmentos u otro tipo de programa sombreador, incluyendo un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 2708A-2708N 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, enmascaramiento y control de flujo de elementos de datos dentro de las instrucciones. El número de canales puede ser independiente del número de unidades aritmético-lógicas (ALU) o unidades de coma flotante (FPU) físicas de un procesador de gráficos en particular. En algunas realizaciones, las unidades de ejecución 2708A-2708N soportan tipos de datos de números enteros y de coma flotante.

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

Una o más cachés de instrucciones internas (por ejemplo, 2706) se incluyen en la lógica de ejecución de hilos 2700 para almacenar en caché instrucciones de hilo para las unidades de ejecución. En algunas realizaciones, una o más cachés de datos (por ejemplo, 2712) se incluyen para almacenar en caché datos de hilo durante la ejecución de hilos. En algunas realizaciones, se incluye un muestreador 2710 para proporcionar un muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunas realizaciones, el muestreador 2710 incluye una funcionalidad de muestreo de textura o de medios especializada para procesar datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las canalizaciones de gráficos y de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilos 2700 mediante lógica de generación y de despacho de hilos. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos para dar datos de píxel, se invoca lógica de procesador de píxeles (por ejemplo, lógica de sombreador de píxeles, lógica de sombreador de fragmentos, etc.) dentro del procesador de sombreador 2702 para calcular adicionalmente información de salida y hacer que se escriban resultados para emitir

superficies (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de estarcido, etc.). En algunas realizaciones, un sombreador de píxeles o un sombreador de fragmentos calcula los valores de los diversos atributos de vértice que se van a interpolar a lo largo del objeto rasterizado. En algunas realizaciones, una lógica de procesador de píxeles dentro del procesador de sombreador 2702 ejecuta entonces un programa sombreador de píxeles o de fragmentos suministrado por una interfaz de programación de aplicaciones (API). Para ejecutar el programa sombreador, el procesador de sombreador 2702 despacha hilos a una unidad de ejecución (por ejemplo, 2708A) mediante el despachador de hilos 2704. En algunas realizaciones, el sombreador de píxeles 2702 usa la lógica de muestreo de textura en el muestreador 2710 para acceder a datos de textura en mapas de textura almacenados en memoria. Operaciones aritméticas sobre los datos de textura y los datos de geometría de entrada calculan datos de color de píxel para cada fragmento geométrico, o descartan el procesamiento adicional de uno o más píxeles.

En algunas realizaciones, el puerto de datos 2714 proporciona un mecanismo de acceso de memoria para que la lógica de ejecución de hilos 2700 emita datos procesados a la memoria para su procesamiento en una canalización de salida de procesador de gráficos. En algunas realizaciones, el puerto de datos 2714 incluye o se acopla a una o más memorias caché (por ejemplo, la caché de datos 2712) para almacenar en caché datos para un acceso de memoria por medio del puerto de datos.

La **Figura 28** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos 2800 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución de procesador de gráficos soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los recuadros con línea continua ilustran los componentes que se incluyen, en general, en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de las instrucciones. En algunas realizaciones, el formato de instrucción 2800 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 instrucciones en un formato de instrucción de 128 bits 2810. Un formato de instrucción compactado de 64 bits 2830 está disponible para algunas instrucciones basándose en la instrucción, las opciones de instrucción y el número de operandos seleccionados. El formato de instrucción de 128 bits nativo 710 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de 64 bits 2830. Las instrucciones nativas disponibles en el formato de 64 bits 2830 varían según la realización. En algunas realizaciones, la instrucción se compacta en parte usando un conjunto de valores de índice en un campo de índice 2813. El hardware de unidad de ejecución consulta un conjunto de tablas de compactación basándose en los valores de índice y usa las salidas de tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 2810.

Para cada formato, el código de operación de instrucción 2812 define la operación que ha de realizar la unidad de ejecución. Las unidades de ejecución ejecutan cada instrucción en paralelo a lo largo de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de suma, la unidad de ejecución realiza una operación de suma simultánea a lo largo de cada canal de color que representa un elemento de textura o un elemento de imagen. Por defecto, la unidad de ejecución ejecuta cada instrucción a lo largo de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 2814 posibilita el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicación) y el orden de canal de datos (por ejemplo, mezcla). Para las instrucciones en el formato de instrucción de 128 bits 2810, un campo de tamaño de ejecución 2816 limita el número de canales de datos que se ejecutarán en paralelo. En algunas realizaciones, el campo de tamaño de ejecución 2816 no está disponible para su uso en el formato de instrucción compacto de 64 bits 2830.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos, incluyendo dos operandos de origen, src0 2820, src1 2822 y un destino 2818. En algunas realizaciones, las unidades de ejecución soportan instrucciones de destino dual, donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 2824), donde el código de operación de instrucción 2812 determina el número de operandos de origen. El último operando de origen de una instrucción puede ser un valor inmediato (por ejemplo, codificado de manera fija) pasado con la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 2810 incluye un campo de modo de acceso/dirección 2826 que especifica, por ejemplo, si se usa el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se usa el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos es proporcionada directamente por bits en la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 2810 incluye un campo de modo de acceso/dirección 2826, 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 por byte para

los operandos de origen y de destino y, cuando está en un segundo modo, la instrucción puede usar un direccionamiento alineado por 16 bytes para todos los operandos de origen y de destino.

En una realización, la porción de modo de dirección del campo de modo de acceso/dirección 2826 determina si la instrucción va a usar direccionamiento directo o indirecto. Cuando se usa el modo de direccionamiento de registro directo, bits en la instrucción proporcionan directamente la dirección de registro de uno o más operandos. Cuando se usa un modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos puede calcularse basándose en un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

En algunas realizaciones, las instrucciones se agrupan basándose en los campos de bits del código de operación 2812 para simplificar la decodificación de código de operación 2840. Para un código de operación de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de código de operación. El agrupamiento del código de operación preciso mostrado es simplemente un ejemplo. En algunas realizaciones, un grupo de código de operación de movimiento y lógica 2842 incluye instrucciones de movimiento y lógica de datos (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y lógica 2842 comparte los cinco bits más significativos (MSB), donde las instrucciones de movimiento (mov) están tienen la forma 0000xxxxb y las instrucciones lógicas tienen la forma 0001xxxxb. Un grupo de instrucciones de control de flujo 2844 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 2846 incluye una mezcla de instrucciones, incluyendo instrucciones de sincronización (por ejemplo, espera, envío) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo matemático paralelo 2848 incluye instrucciones aritméticas a nivel de componente (por ejemplo, añadir, multiplicar (mult)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculo matemático paralelo 2848 realiza las operaciones aritméticas en paralelo a lo largo de canales de datos. El grupo de cálculo matemático vectorial 2850 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cálculo matemático vectorial realiza una aritmética tal como cálculos de producto escalar sobre operandos de vectores.

Canalización de gráficos ilustrativa adicional

La **Figura 29** es un diagrama de bloques de un procesador de gráficos 2900 de acuerdo con otra realización. 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, el procesador de gráficos 2900 incluye una canalización de gráficos 2920, una canalización de medios 2930, un motor de visualización 2940, una lógica de ejecución de hilos 2950 y una canalización de salida de representación 2970. En algunas realizaciones, el procesador de gráficos 2900 es un procesador de gráficos dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos es controlado por escrituras de registro en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 2900 mediante una interconexión en anillo 2902. En algunas realizaciones, la interconexión en anillo 2902 acopla el procesador de gráficos 2900 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de propósito general. Los comandos desde la interconexión en anillo 2902 se interpretan por un transmisor de envío por flujo continuo de comandos 2903, que suministra instrucciones a componentes individuales de la canalización de gráficos 2920 o la canalización de medios 2930.

En algunas realizaciones, el transmisor por flujo continuo de comandos 2903 dirige la operación de un extractor de vértices 2905 que lee datos de vértice desde memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor por flujo continuo de comandos 2903. En algunas realizaciones, el extractor de vértices 2905 proporciona datos de vértice a un sombreador de vértices 2907, que realiza operaciones de transformación y de iluminación de espacio de coordenadas en cada vértice. En algunas realizaciones, el extractor de vértices 2905 y el sombreador de vértices 2907 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a unidades de ejecución 2952A-2952B mediante un despachador de hilos 2931.

En algunas realizaciones, las unidades de ejecución 2952A-2952B son una matriz de procesadores de vectores que tienen un conjunto de instrucciones para realizar operaciones de gráficos y de medios. En algunas realizaciones, las unidades de ejecución 2952A-2952B tienen una caché L1 2951 adjunta que es específica para cada matriz o que se comparte entre las matrices. La caché puede configurarse como una caché de datos, una caché de instrucciones o una única caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.

En algunas realizaciones, la canalización de gráficos 2920 incluye componentes de teselación para realizar una teselación acelerada por hardware de objetos 3D. En algunas realizaciones, un sombreador de casco programable 811 configura las operaciones de teselación. Un sombreador de dominio programable 817 proporciona una evaluación de extremo posterior del resultado de la teselación. Un teselador 2913 opera en la dirección del sombreador de casco 2911 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados basándose en un modelo geométrico grueso que se proporciona como entrada a la canalización de gráficos 2920. En

algunas realizaciones, si no se usa la teselación, pueden omitirse los componentes de teselación (por ejemplo, el sombreador de casco 2911, el teselador 2913 y el sombreador de dominio 2917).

En algunas realizaciones, unos objetos geométricos completos pueden ser procesados por un sombreador de geometría 2919 mediante uno o más hilos despachados a las unidades de ejecución 2952A-2952B, o puede avanzar directamente al recortador 2929. En algunas realizaciones, el sombreador de geometría opera sobre objetos geométricos enteros, en lugar de vértices o parches de vértices como en fases previas de la canalización de gráficos. Si la teselación está deshabilitada, el sombreador de geometría 2919 recibe una entrada desde el sombreador de vértices 2907. En algunas realizaciones, el sombreador de geometría 2919 puede programarse mediante un programa sombreador de geometría para realizar una teselación de geometría si las unidades de teselación están deshabilitadas.

Antes de la rasterización, un recortador 2929 procesa datos de vértice. El recortador 2929 puede ser un recortador de función fija o un recortador programable que tiene funciones de recorte y de sombreador de geometría. En algunas realizaciones, un componente de prueba de rasterizador y de profundidad 2973 en la canalización de salida de representación 2970 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunas realizaciones, la lógica de sombreador de píxeles está incluida en la lógica de ejecución de hilos 2950. En algunas realizaciones, una aplicación puede omitir el componente de rasterización y prueba de profundidad 2973 y acceder a datos de vértice sin rasterizar mediante una unidad de salida de flujo 2923.

El procesador de gráficos 2900 tiene un bus de interconexión, un tejido de interconexión o algún otro mecanismo de interconexión que permite que datos y mensajes pasen entre los componentes principales del procesador. En algunas realizaciones, las unidades de ejecución 2952A-2952B y la(s) caché(s) 2951 asociada(s), el muestreador de textura y de medios 2954 y la caché de textura/muestreador 2958 se interconectan mediante un puerto de datos 2956 para realizar accesos a memoria y comunicarse con componentes de canalización de salida de representación del procesador. En algunas realizaciones, el muestreador 2954, las memorias caché 2951, 2958 y las unidades de ejecución 2952A-2952B tienen, cada uno, rutas de acceso de memoria separadas.

En algunas realizaciones, la canalización de salida de representación 2970 contiene un componente de rasterización y prueba de profundidad 2973 que convierte objetos basados en vértices en una representación asociada basada en píxeles. En algunas realizaciones, la lógica de rasterizador incluye una unidad generadora de ventanas/enmascaradora para realizar una rasterización de líneas y de triángulos de función fija. En algunas realizaciones también están disponibles una caché de representación 2978 y una caché de profundidad 2979 asociadas. Un componente de operaciones de píxel 2977 realiza operaciones basadas en píxeles sobre los datos, aunque, en algunas instancias, las operaciones de píxel asociadas con operaciones 2D (por ejemplo, transferencias de imagen de bloque de bits con mezcla) son realizadas por el motor 2D 2941, o son sustituidas en el momento de la visualización por el controlador de visualización 2943 usando planos de visualización de superposición. En algunas realizaciones, está disponible una caché L3 compartida 2975 para todos los componentes de gráficos, permitiendo compartir datos sin el uso de memoria de sistema principal.

En algunas realizaciones, la canalización de medios de procesador de gráficos 2930 incluye un motor de medios 2937 y un extremo frontal de vídeo 2934. En algunas realizaciones, el extremo frontal de vídeo 2934 recibe comandos de canalización desde el transmisor por flujo continuo de comandos 2903. En algunas realizaciones, la canalización de medios 2930 incluye un transmisor por flujo continuo de comandos separado. En algunas realizaciones, el extremo frontal de vídeo 2934 procesa comandos de medios antes de enviar el comando al motor de medios 2937. En algunas realizaciones, el motor de medios 2937 incluye una funcionalidad de generación de hilos para generar hilos para despacharlos a la lógica de ejecución de hilos 2950 mediante el despachador de hilos 2931.

En algunas realizaciones, el procesador de gráficos 2900 incluye un motor de visualización 2940. En algunas realizaciones, el motor de visualización 2940 es externo al procesador 2900 y se acopla con el procesador de gráficos mediante la interconexión en anillo 2902, o algún otro bus o tejido de interconexión. En algunas realizaciones, el motor de visualización 2940 incluye un motor 2D 2941 y un controlador de visualización 2943. En algunas realizaciones, el motor de visualización 2940 contiene una lógica de propósito especial capaz de operar independientemente de la canalización 3D. En algunas realizaciones, el controlador de visualización 2943 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 canalización de gráficos 2920 y la canalización de medios 2930 pueden configurarse para realizar operaciones basándose en múltiples interfaces de programación de gráficos y de medios y no son específicas de ninguna interfaz de programación de aplicaciones (API). En algunas realizaciones, el software de controlador para el procesador de gráficos traduce llamadas de API que son específicas de una biblioteca de medios o de gráficos particular a comandos que pueden ser procesados por el procesador de gráficos. En algunas realizaciones, se proporciona soporte para la Biblioteca de Gráficos Abierta (OpenGL), Lenguaje Informático Abierto (OpenCL) y/o API de gráficos y de cálculo Vulkan, todas ellas del grupo Khronos. En algunas realizaciones, también puede proporcionarse soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, puede soportarse una combinación de estas bibliotecas. También puede proporcionarse soporte para la Biblioteca de Visión

Informática de Código Abierto (OpenCV). También se soportaría una API futura con una canalización 3D compatible si puede hacerse un mapeo desde la canalización de la API futura a la canalización del procesador de gráficos.

Programación de canalización de gráficos ilustrativa adicional

La **Figura 30A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos 3000 de acuerdo con algunas realizaciones. La **Figura 30B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos 3010 de acuerdo con una realización. Los recuadros con línea continua en la **Figura 30A** ilustran los componentes que se incluyen, en general, en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de los comandos de gráficos. El formato de comando de procesador gráfico 3000 ilustrativo de la **Figura 30A** incluye campos de datos para identificar un cliente objetivo 3002 del comando, un código de operación (cód. de ope.) de comando 3004 y los datos 3006 relevantes para el comando. También se incluyen un subcódigo de operación 3005 y un tamaño de comando 3008 en algunos comandos.

En algunas realizaciones, el cliente 3002 especifica la unidad de cliente del dispositivo de gráficos que procesa los datos de comando. En algunas realizaciones, un analizador de comandos de procesador de gráficos examina el campo de cliente de cada comando para acondicionar el procesamiento adicional del comando y encaminar los datos de comando a la unidad de cliente apropiada. En algunas realizaciones, las unidades de cliente del procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de representación, una unidad 2D, una unidad 3D y una unidad multimedia. Cada unidad de cliente tiene una canalización de procesamiento correspondiente que procesa los comandos. Una vez que el comando es recibido por la unidad de cliente, la unidad de cliente lee el código de operación 3004 y, si está presente, el sub-código de operación 3005 para determinar la operación que hay que realizar. La unidad de cliente realiza el comando usando información en el campo de datos 3006. Para algunos comandos, se espera que un tamaño de comando explícito 3008 especifique el tamaño del comando. En algunas realizaciones, el analizador de comandos determina automáticamente el tamaño de al menos algunos de los comandos basándose en el código de operación de comando. En algunas realizaciones, los comandos se alinean mediante múltiplos de una palabra doble.

El diagrama de flujo en la **Figura 30B** muestra una secuencia de comandos de procesador de gráficos 3010 ilustrativa. 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 y se describe una secuencia de comandos de muestra solo a fines de ejemplo, debido a que las realizaciones no se limitan a estos comandos específicos o a esta secuencia de comandos. Además, los comandos pueden emitirse como un lote de comandos en una secuencia de comandos, de manera que el procesador de gráficos procesará la secuencia de comandos de manera al menos parcialmente concurrente.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 3010 puede comenzar con un comando de vaciado de canalización 3012 para hacer que alguna canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunas realizaciones, la canalización 3D 3022 y la canalización de medios 3024 no operan concurrentemente. El vaciado de canalización se realiza para hacer que la canalización de gráficos activa complete cualquier comando pendiente. En respuesta a un vaciado de canalización, el analizador de comandos del procesador gráfico pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las memorias caché de lectura relevantes. Opcionalmente, cualquier dato en la caché de representación que se marque como 'sucio' puede vaciarse a memoria. En algunas realizaciones, el comando de vaciado de canalización 3012 puede usarse para la sincronización de canalización o antes de colocar el procesador gráfico en un estado de bajo consumo de energía.

En algunas realizaciones, se usa un comando de selección de canalización 3013 cuando una secuencia de comandos requiere que el procesador de gráficos conmute explícitamente entre canalizaciones. En algunas realizaciones, un comando de selección de canalización 3013 solo se requiere una vez dentro de un contexto de ejecución antes de emitir comandos de canalización a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunas realizaciones, un comando de vaciado de canalización 3012 se requiere inmediatamente antes de un conmutador de canalización por medio del comando de selección de canalización 3013.

En algunas realizaciones, un comando de control de canalización 3014 configura una canalización de gráficos para la operación y se usa para programar la canalización 3D 3022 y la canalización de medios 3024. En algunas realizaciones, el comando de control de canalización 3014 configura el estado de canalización para la canalización activa. En una realización, el comando de control de canalización 3014 se usa para la sincronización de canalización y para limpiar datos de una o más memorias caché dentro de la canalización activa antes de procesar un lote de comandos.

En algunas realizaciones, los comandos de estado de memoria intermedia de retorno 3016 se usan para configurar un conjunto de memorias intermedias de retorno para que las canalizaciones respectivas escriban datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas realizaciones,

el procesador de gráficos también usa una o más memorias intermedias de retorno para almacenar datos de salida y realizar comunicación de hilos cruzada. En algunas realizaciones, el estado de memoria intermedia de retorno 3016 incluye seleccionar el tamaño y número de memorias intermedias de retorno que usar para un conjunto de operaciones de canalización.

Los comandos restantes en la secuencia de comandos difieren basándose en la canalización activa para las operaciones. Basándose en una determinación de canalización 3020, la secuencia de comandos se adapta a la canalización 3D 3022 comenzando con el estado de canalización 3D 3030, o a la canalización de medios 3024 comenzando en el estado de canalización de medios 3040.

Los comandos para configurar el estado de canalización 3D 3030 incluyen comandos de ajuste de estado 3D para estado de memoria intermedia de vértice, estado de elemento de vértice, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que han de configurarse antes de que se procesen los comandos de primitiva 3D. Los valores de estos comandos se determinan, al menos en parte, basándose en la API 3D particular en uso. En algunas realizaciones, los comandos de estado de canalización 3D 3030 también pueden deshabilitar u omitir selectivamente ciertos elementos de la canalización si no se van a usar estos elementos.

En algunas realizaciones, el comando de primitivas 3D 3032 se usa para enviar primitivas 3D que serán procesadas mediante la canalización 3D. Los comandos y parámetros asociados que se pasan al procesador de gráficos a través del comando de primitiva 3D 3032 se reenvían a la función de extracción de vértices en la canalización de gráficos. La función de extracción de vértices usa los datos de comando de la primitiva 3D 3032 para generar estructuras de datos de vértice. Las estructuras de datos de vértice se almacenan en una o más memorias intermedias de retorno. En algunas realizaciones, el comando de la primitiva 3D 3032 se usa para realizar operaciones de vértice sobre primitivas 3D mediante sombreadores de vértices. Para procesar sombreadores de vértices, la canalización 3D 3022 despacha hilos de ejecución de sombreador a unidades de ejecución de procesador de gráficos.

En algunas realizaciones, la canalización 3D 3022 se desencadena a través de un comando o evento de ejecución 3034. En algunas realizaciones, una escritura de registro desencadena una ejecución de comando. En algunas realizaciones, la ejecución se desencadena por medio de un comando 'ir' o 'poner en marcha' en la secuencia de comandos. En una realización, la ejecución del comando se desencadena usando un comando de sincronización de canalización para vaciar la secuencia de comandos a través de la canalización de gráficos. La canalización 3D realizará un procesamiento de geometría para las primitivas 3D. Una vez que se han completado las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles da color a los píxeles resultantes. También pueden incluirse comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo posterior de píxeles para esas operaciones.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 3010 sigue la ruta de canalización de medios 3024 cuando se llevan a cabo operaciones de medios. En general, el uso específico y la forma de programación para la canalización de medios 3024 depende de los medios o de las operaciones de cálculo que se van a realizar. Operaciones de decodificación de medios específicas pueden descargarse a la canalización de medios durante la decodificación de medios. En algunas realizaciones, puede omitirse también la canalización de medios y puede realizarse la decodificación de medios, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. En una realización, la canalización de medios también incluye elementos para operaciones de unidad de procesador de gráficos de propósito general (GPGPU), donde el procesador de gráficos se usa para realizar operaciones de vector SIMD usando programas sombreadores computacionales que no están relacionados explícitamente con la representación de primitivas de gráficos.

En algunas realizaciones, se configura la canalización de medios 3024 de una manera similar que la canalización 3D 3022. Un conjunto de comandos para configurar el estado de canalización de medios 3040 se despacha o coloca en una cola de comandos antes de los comandos de objeto de medios 3042. En algunas realizaciones, los comandos de estado de canalización de medios 3040 incluyen datos para configurar los elementos de canalización de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación y codificación de vídeo dentro de la canalización de medios, tal como el formato de codificación o decodificación. En algunas realizaciones, los comandos de estado de canalización de medios 3040 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 3042 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo que hay que procesar. En algunas realizaciones, todos los estados de canalización de medios deben ser válidos antes de emitir un comando de objeto de medios 3042. Una vez que se ha configurado el estado de canalización y los comandos de objeto de medios 3042 se han puesto en cola, la canalización de medios 3024 se desencadena mediante un comando de ejecución 3044 o un evento de ejecución equivalente (por ejemplo, una escritura de registro). La salida desde la canalización de medios 3024 puede post-procesarse, a continuación, mediante operaciones proporcionadas por la canalización 3D 3022 o la canalización de medios 3024. En algunas realizaciones, las operaciones de GPGPU se configuran y se ejecutan de una manera similar a la de las operaciones de medios.

Arquitectura de software de gráficos ilustrativa adicional

La **Figura 31** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos 3100 de acuerdo con algunas realizaciones. En algunas realizaciones, la arquitectura de software incluye una aplicación de gráficos 3D 3110, un sistema operativo 3120 y al menos un procesador 3130. En algunas realizaciones, el procesador 3130 incluye un procesador de gráficos 3132 y uno o más núcleos de procesador de propósito general 3134. Cada uno de la aplicación de gráficos 3110 y el sistema operativo 3120 se ejecutan en la memoria de sistema 3150 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación de gráficos 3D 3110 contiene uno o más programas sombreadores que incluyen instrucciones de sombreador 3112. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el Lenguaje de Sombreador de Alto Nivel (HLSL) o el Lenguaje de Sombreador OpenGL (GLSL). La aplicación también incluye las instrucciones ejecutables 3114 en un lenguaje máquina adecuado para su ejecución por el núcleo o núcleos de procesador de propósito general 3134. La aplicación también incluye los objetos de gráficos 3116 definidos por datos de vértice.

En algunas realizaciones, el sistema operativo 3120 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo similar a UNIX patentado o un sistema operativo similar a UNIX de código abierto que usa una variante del núcleo Linux. El sistema operativo 3120 puede soportar una API de gráficos 3122 tal como la API de Direct3D, la API de OpenGL o la API de Vulkan. Cuando está en uso la API de Direct3D, el sistema operativo 3120 usa un compilador de sombreador de extremo frontal 3124 para compilar cualquier instrucción de sombreador 3112 en HLSL a un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede llevar a cabo una precompilación de sombreador. En algunas realizaciones, los sombreadores de alto nivel se compilan a sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 3110. En algunas realizaciones, las instrucciones de sombreador 3112 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 en modo de usuario 3126 contiene un compilador de sombreador de extremo posterior 3127 para convertir las instrucciones de sombreador 3112 en una representación específica de hardware. Cuando está en uso la API de OpenGL, las instrucciones de sombreador 3112 en el lenguaje de alto nivel GLSL se pasan a un controlador de gráficos de modo de usuario 3126 para su compilación. En algunas realizaciones, el controlador de gráficos de modo de usuario 3126 usa las funciones de modo de núcleo de sistema operativo 3128 para comunicarse con un controlador de gráficos de modo de núcleo 3129. En algunas realizaciones, el controlador de gráficos de modo de núcleo 3129 se comunica con el procesador de gráficos 3132 para despachar comandos e instrucciones.

Implementaciones de núcleo de IP ilustrativas adicionales

Uno o más aspectos de al menos una realización pueden implementarse mediante un código representativo almacenado en un medio legible por máquina que representa y/o define la lógica dentro de un circuito integrado, tal como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan una lógica diversa dentro del procesador. Cuando son leídas por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en el presente documento. Tales representaciones, conocidas como "núcleos de IP", son unidades reutilizables de lógica para un circuito integrado que 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 operaciones descritas en asociación con cualquiera de las realizaciones descritas en el presente documento.

La **Figura 32** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 3200 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 3200 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 3230 puede generar una simulación de software 3210 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 3210 puede usarse para diseñar, someter a prueba y verificar el comportamiento del núcleo de IP usando un modelo de simulación 3212. El modelo de simulación 3212 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Puede crearse o sintetizarse, a continuación, un diseño de nivel de transferencia de registro (RTL) 3215 a partir del modelo de simulación 3212. El diseño de RTL 3215 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, incluyendo la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño de RTL 3215, también pueden crearse, diseñarse o sintetizarse diseños de nivel inferior al nivel de lógica o al nivel de transistores. Por tanto, los detalles particulares del diseño y simulación inicial pueden variar.

El diseño de RTL 3215, o un equivalente, puede ser sintetizado adicionalmente por la instalación de diseño para dar un modelo de hardware 3220, 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 someterse a prueba 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 terceros 3265 usando la memoria no volátil 3240 (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, por Internet) a través de una conexión cableada 3250 o una conexión inalámbrica 3260. La instalación de fabricación 3265 puede fabricar, a continuación, un circuito integrado que se basa, al menos en parte, en el diseño de núcleo de IP. El circuito integrado fabricado puede configurarse para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Circuito integrado de sistema en un chip ilustrativo adicional

Las **Figuras 33-35** ilustran circuitos integrados ilustrativos 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, pueden incluirse otros circuitos y lógica, incluyendo procesadores/núcleos de gráficos adicionales, controladores de interfaz de periféricos o núcleos de procesador de propósito general.

La **Figura 33** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip 3300 ilustrativo que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 3300 ilustrativo incluye uno o más procesadores de aplicaciones 3305 (por ejemplo, una CPU), al menos un procesador de gráficos 3310, y puede incluir adicionalmente un procesador de imágenes 3315 y/o un procesador de vídeo 3320, cualquiera de los cuales puede ser un núcleo de IP modular desde las mismas o múltiples instalaciones de diseño diferentes. El circuito integrado 3300 incluye una lógica de bus o de periféricos que incluye un controlador de USB 3325, un controlador de UART 3330, un controlador de SPI/SDIO 3335 y un controlador de I²S/I²C 3340. Adicionalmente, el circuito integrado puede incluir un dispositivo de visualización 3345 acoplado a uno o más de un controlador de interfaz multimedia de alta definición (HDMI) 3350 y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 3355. El almacenamiento puede ser proporcionado por un subsistema de memoria flash 3360 que incluye memoria flash y un controlador de memoria flash. La interfaz de memoria puede proporcionarse mediante un controlador de memoria 3365 para el acceso a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 3370.

La **Figura 34** es un diagrama de bloques que ilustra un procesador de gráficos 3410 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 3410 puede ser una variante del procesador de gráficos 3310 de la **Figura 33**. El procesador de gráficos 3410 incluye un procesador de vértices 3405 y uno o más procesador(es) de fragmentos 3415A-3415N (por ejemplo, 3415A, 3415B, 3415C, 3415D a 3415N-1 y 3415N). El procesador de gráficos 3410 puede ejecutar diferentes programas sombreadores por medio de una lógica diferente, de tal forma que el procesador de vértices 3405 se optimiza para ejecutar operaciones para programas sombreadores de vértices, mientras que los uno o más procesadores de fragmentos 3415A-3415N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas sombreadores de fragmentos o de píxeles. El procesador de vértices 3405 realiza la fase de procesamiento de vértices de la canalización de gráficos 3D y genera primitivas y datos de vértice. El/los procesador(es) de fragmentos 3415A-3415N usa(n) los datos de primitiva y de vértice generados por el procesador de vértices 3405 para producir una memoria intermedia de fotogramas que se visualiza en un dispositivo de visualización. En una realización, el/los procesador(es) de fragmentos 3415A-3415N se optimiza(n) para ejecutar programas sombreadores de fragmentos según lo previsto en la API de OpenGL, que pueden usarse para llevar a cabo operaciones similares como un programa sombreador de píxeles según lo previsto en la API de Direct 3D.

El procesador de gráficos 3410 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 3420A-3420B, caché(s) 3425A-3425B e interconexión(es) de circuito 3430A-3430B. La(s) una o más MMU 3420A-3420B prevé(n) un mapeo de dirección virtual a física para el circuito integrado 3410, incluyendo para el procesador de vértices 3405 y/o el/los procesador(es) de fragmentos 3415A-3415N, que pueden hacer referencia a datos de vértice o de imagen/textura almacenados en memoria, además de datos de vértice o de imagen/textura almacenados en la(s) una o más caché(s) 3425A-3425B. En una realización, las una o más MMU 3425A-3425B pueden sincronizarse con otras MMU dentro del sistema, incluyendo una o más MMU asociadas a los uno o más procesadores de aplicación 3305, procesador de imagen 3315 y/o procesador de vídeo 3320 de la **Figura 33**, de manera que cada procesador 3305-3320 puede participar en un sistema de memoria virtual compartido o unificado. Las una o más interconexiones de circuito 3430A-3430B posibilitan que el procesador de gráficos 3410 interaccione con otros núcleos de IP dentro del SoC, o bien mediante un bus interno del SoC o bien mediante una conexión directa, de acuerdo con unas realizaciones.

La **Figura 35** es un diagrama de bloques que ilustra un procesador de gráficos 3510 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 3510 puede ser una variante del procesador de gráficos 3310 de la **Figura 33**. El procesador

de gráficos 3510 incluye las una o más MMU 3420A-3420B, las cachés 3425A-3425B y las interconexiones de circuito 3430A-3430B del circuito integrado 3400 de la **Figura 34**.

El procesador de gráficos 3510 incluye uno o más núcleos de sombreador 3515A-3515N (por ejemplo, 3515A, 3515B, 3515C, 3515D, 3515E, 3515F a 3515N-1 y 3515N), lo que prevé una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo o núcleo puede ejecutar todos los tipos de código sombreador programable, que incluyen código de programa sombreador para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. El número exacto de núcleos de sombreador presentes puede variar entre realizaciones e implementaciones. Adicionalmente, el procesador de gráficos 3510 incluye un gestor de tareas entre núcleos 3505, que actúa como un despachador de hilos para despachar hilos de ejecución a uno o más núcleos de sombreador 3515A-3515N y una unidad de teselado 3518 para acelerar operaciones de teselado para una representación basada en teselas, en la que las operaciones de representación para una escena se subdividen en el espacio de imágenes, por ejemplo, para aprovechar la coherencia espacial local dentro de una escena o para optimizar el uso de cachés internas.

Las realizaciones descritas en el presente documento se refieren a configuraciones específicas de hardware, tales como circuitos integrados específicos de la aplicación (ASIC), configuradas para realizar ciertas operaciones o que tienen una funcionalidad predeterminada. Tales dispositivos electrónicos habitualmente incluyen un conjunto de uno o más procesadores acoplados a uno o más de 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 habitualmente se produce a través de uno o más buses y puentes (también denominados controladores de bus). El dispositivo de almacenamiento y las señales que portan el tráfico de red representan, respectivamente, uno o más medios de almacenamiento legibles por máquina y medios de comunicación legibles por máquina. Por lo tanto, los dispositivos de almacenamiento de un dispositivo electrónico dado 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 lo largo de toda esta descripción detallada, con fines explicativos, se expusieron numerosos detalles específicos para proporcionar un entendimiento completo de la presente invención. Sin embargo, será evidente para un experto en la materia que las realizaciones pueden ponerse en práctica sin algunos de estos detalles específicos. En ciertas instancias, no se describieron estructuras y funciones bien conocidas con todo lujo de detalles para evitar complicar la materia objeto inventiva de las realizaciones.

REIVINDICACIONES

1. Un aparato de cálculo que comprende:

- 5 una unidad de decodificación para decodificar una única instrucción para dar una instrucción decodificada que especifica múltiples operandos que incluyen un valor de entrada de múltiples bits (1801) y un peso binario bipolar asociado con una red neuronal; y
 una unidad aritmético-lógica que incluye un peso indexado (1802), un multiplicador (1804), un sumador (1812) y un registro de acumulador (1814), en donde, para ejecutar la instrucción decodificada, el multiplicador (1804) tiene que
 10 realizar una operación de multiplicación sobre el valor de entrada de múltiples bits (1801) basándose en el peso binario bipolar proporcionado por el peso indexado (1802) para generar un producto intermedio y el sumador es para sumar el producto intermedio a un valor almacenado en el registro de acumulador y actualizar el valor almacenado en el registro de acumulador;
 en donde el peso binario bipolar representa un valor de peso de uno positivo y uno negativo, en donde el multiplicador (1804) incluye una unidad de inversión de signo (1805) para generar el producto intermedio, la unidad de inversión de
 15 signo (1805) para invertir el signo del valor de entrada de múltiples bits (1801) para un valor de peso bipolar de uno negativo y pasar al otro lado el valor de entrada de múltiples bits (1801) para un valor de peso bipolar de uno positivo, en donde el peso binario bipolar representa un valor de peso de uno negativo como un cero binario, y en donde el multiplicador (1804) incluye una puerta NO (1902) entre una entrada para el peso binario bipolar y la unidad de
 20 inversión de signo (1805), en donde el peso indexado (1802) se acopla a la puerta NO (1902) de manera que un valor de peso de uno negativo activa la unidad de inversión de signo (1805) para invertir el signo del valor de entrada de múltiples bits (1801).

25 2. El aparato de cálculo según la reivindicación 1, en donde se hace referencia al valor del peso binario bipolar a través de un índice en un registro de múltiples bits.

3. El aparato de cálculo según la reivindicación 1, que incluye adicionalmente un registro de salida para almacenar un valor de salida de la única instrucción.

30 4. El aparato de cálculo según la reivindicación 1, en donde el valor de entrada de múltiples bits (1801) tiene un número de bits que es una potencia de dos.

5. El aparato de cálculo según la reivindicación 1, que comprende además:

35 una unidad de procesamiento de gráficos de propósito general que comprende la unidad de decodificación; y una memoria acoplada con la unidad de procesamiento de gráficos de propósito general.

6. Un método que comprende:

- 40 decodificar una única instrucción que especifica múltiples operandos, incluyendo los operandos un valor de entrada de múltiples bits y un peso binario bipolar asociado con una red neuronal;
 emitir la única instrucción para su ejecución dentro de una unidad de cálculo de una unidad de procesamiento de gráficos de propósito general; y
 en respuesta a la ejecución de la única instrucción, generar un resultado realizando una operación de multiplicación
 45 sobre el valor de entrada de múltiples bits basándose en el peso binario bipolar para generar un producto intermedio y actualizar un valor almacenado en un registro de acumulador sumando el producto intermedio al valor almacenado en un registro de acumulador; en donde el peso binario bipolar representa un valor de peso de uno positivo y uno negativo, en donde el producto intermedio se genera a través de una unidad de inversión de signo (1805), en donde la unidad de inversión de signo (1805) es para invertir el signo del valor de entrada de múltiples bits (1801) para un
 50 valor de peso bipolar de uno negativo y pasar al otro lado el valor de entrada de múltiples bits (1801) para un valor de peso bipolar de uno positivo, en donde el peso binario bipolar representa un valor de peso de uno negativo como un cero binario, y
 en donde un valor de peso de uno negativo activa la unidad de inversión de signo (1805) a través de una puerta NO (1902) para invertir el signo del valor de entrada de múltiples entradas (1801).

55 7. El método según la reivindicación 6, en donde se hace referencia al valor del peso binario bipolar a través de un índice en un registro de múltiples bits.

60 8. El método según la reivindicación 6, en donde el valor de entrada de múltiples bits tiene un número de bits que es una potencia de dos.

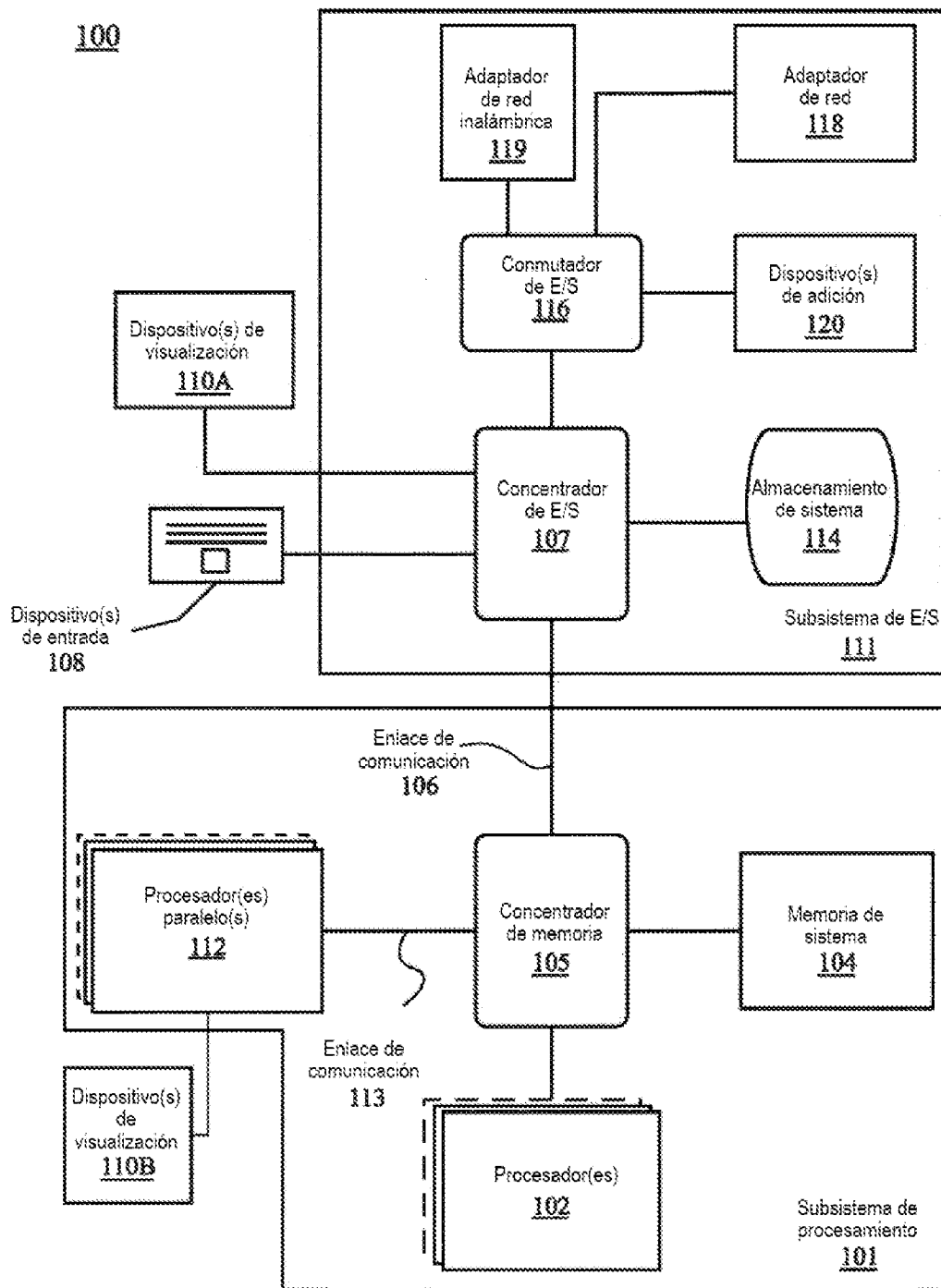


FIG. 1

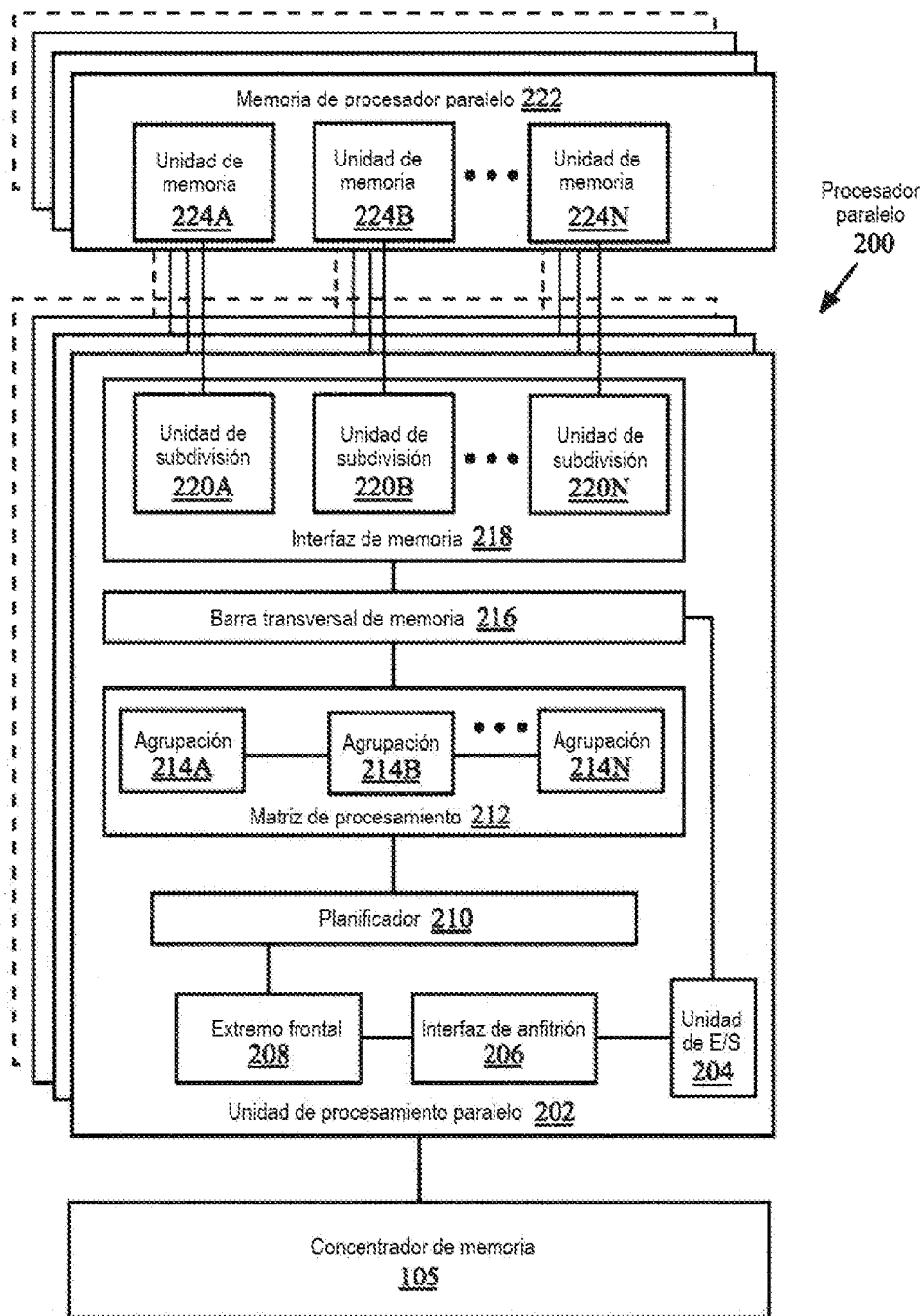


FIG. 2A

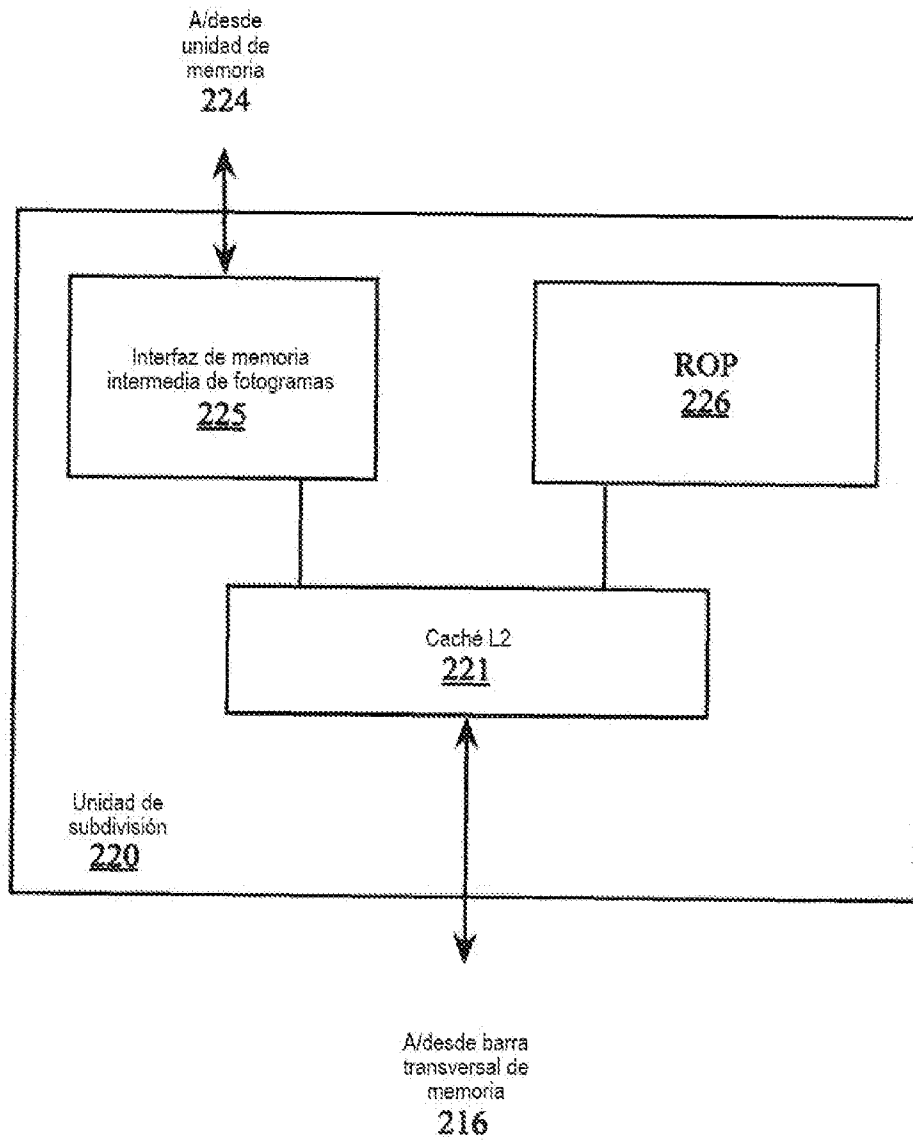


FIG. 2B

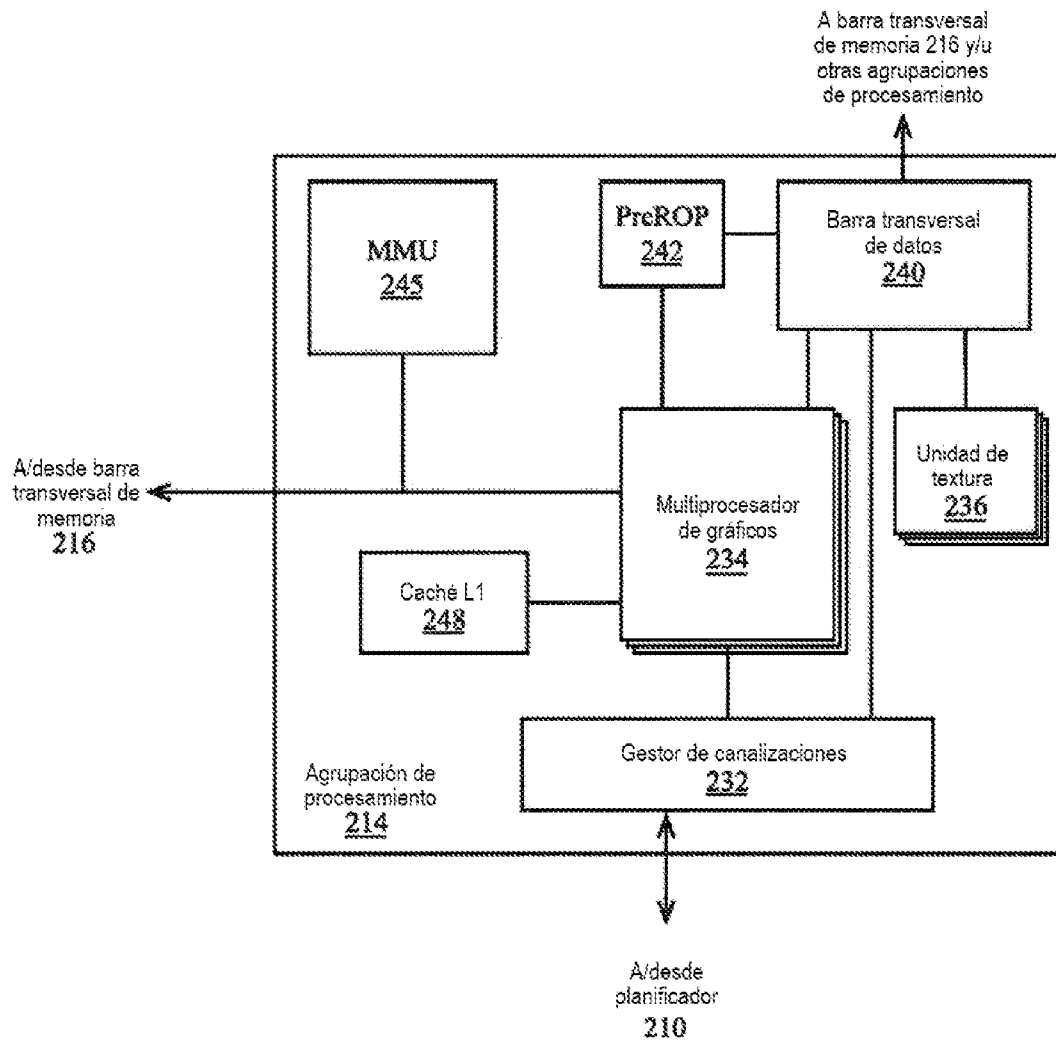


FIG. 2C

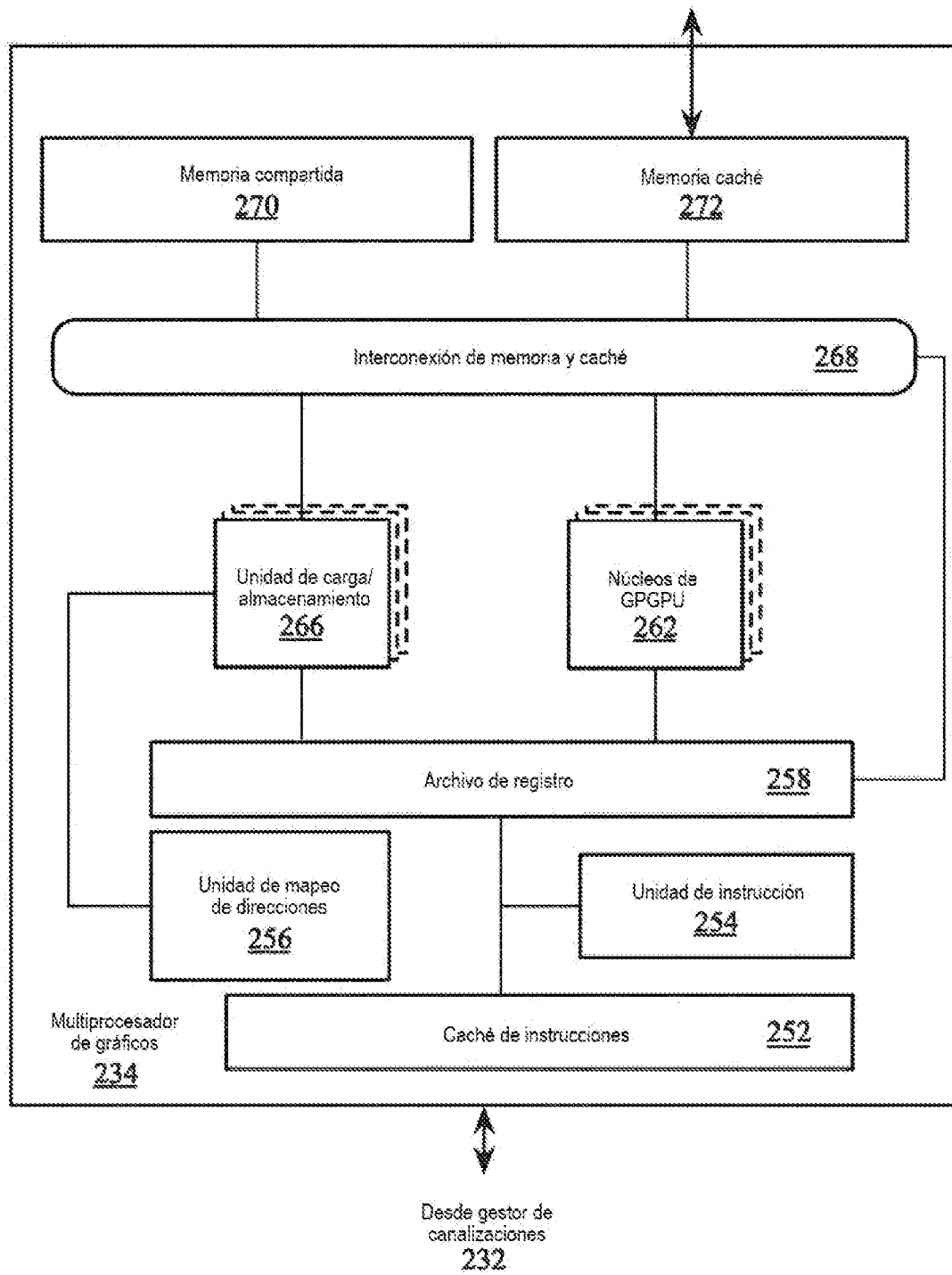


FIG. 2D

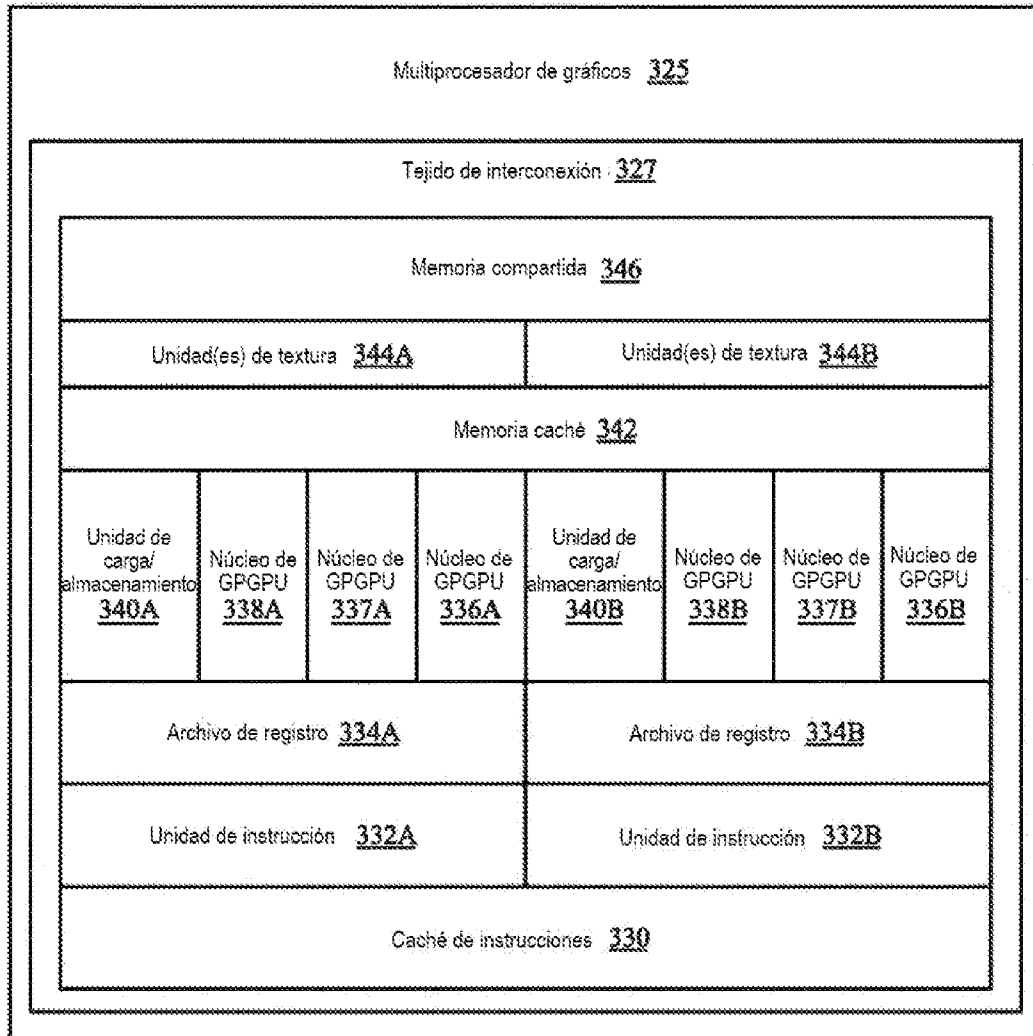


FIG. 3A

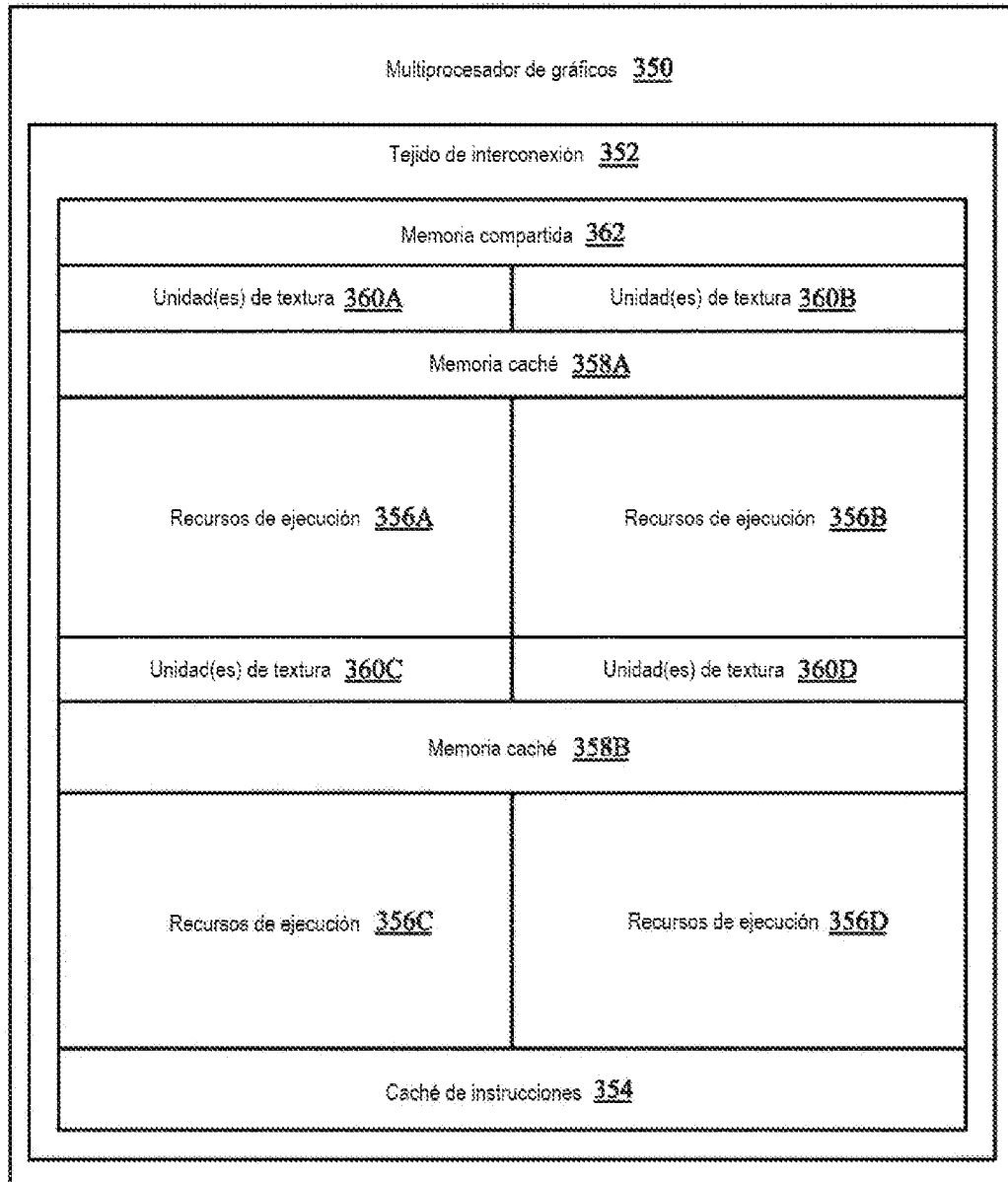


FIG. 3B

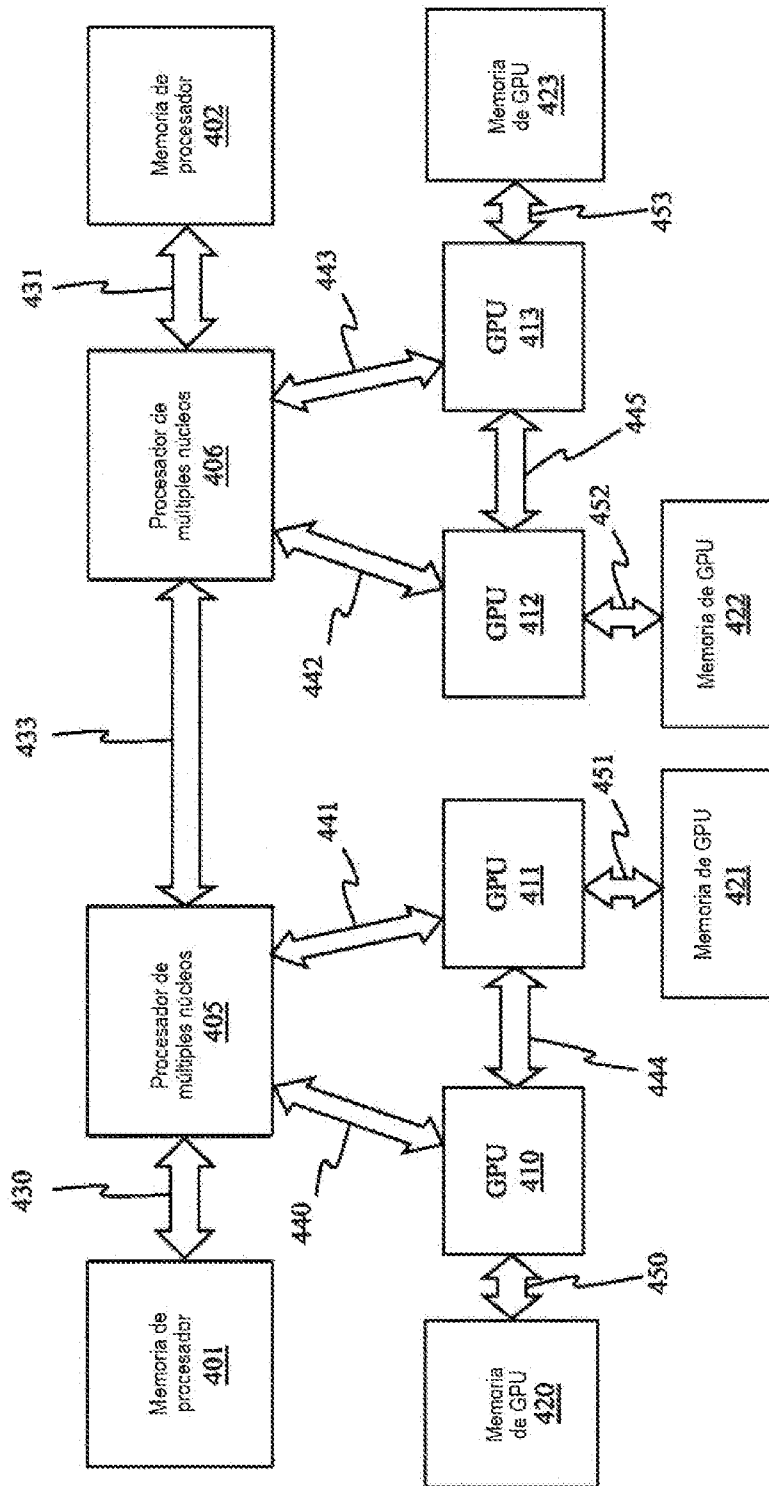


FIG. 4A

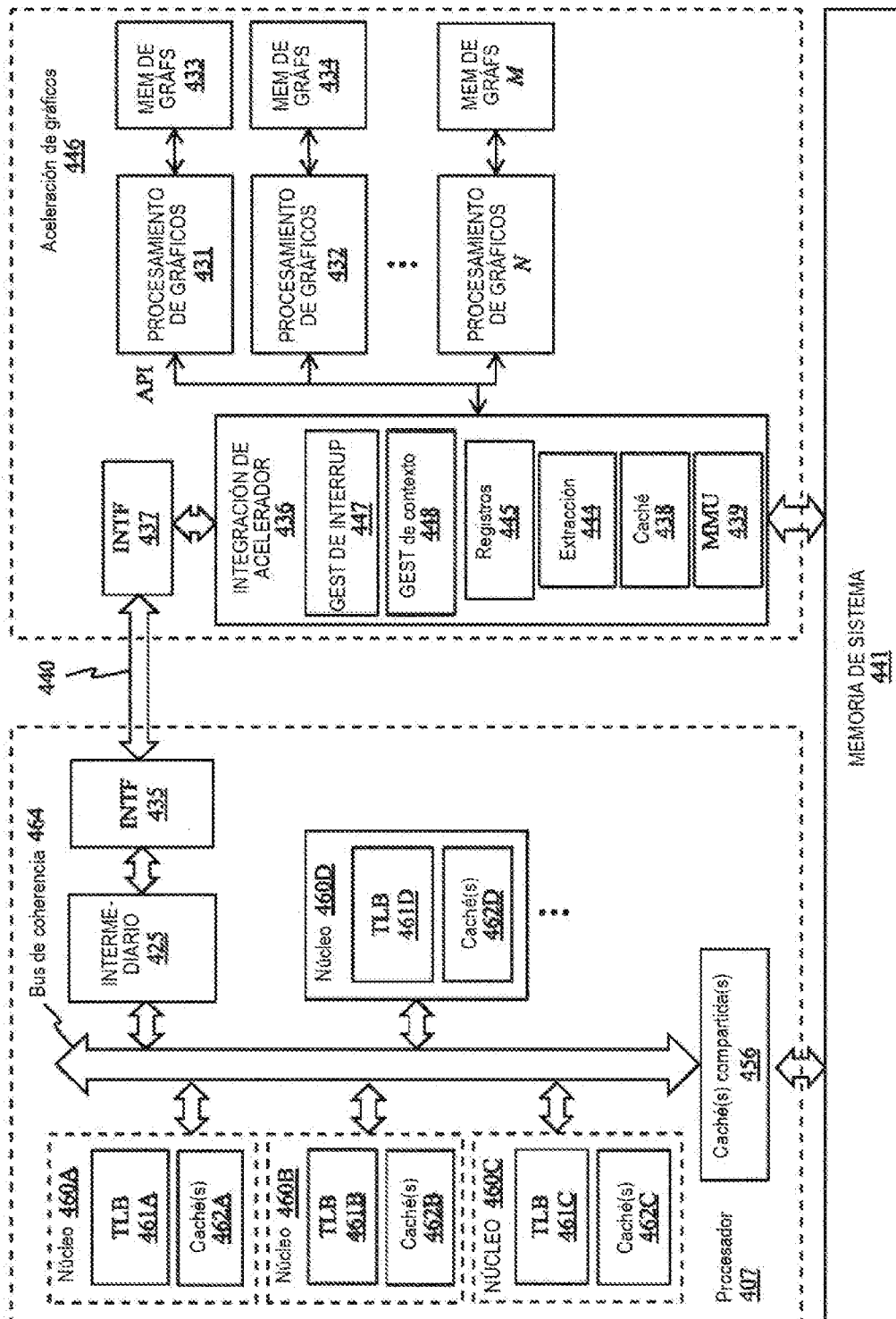


FIG. 4B

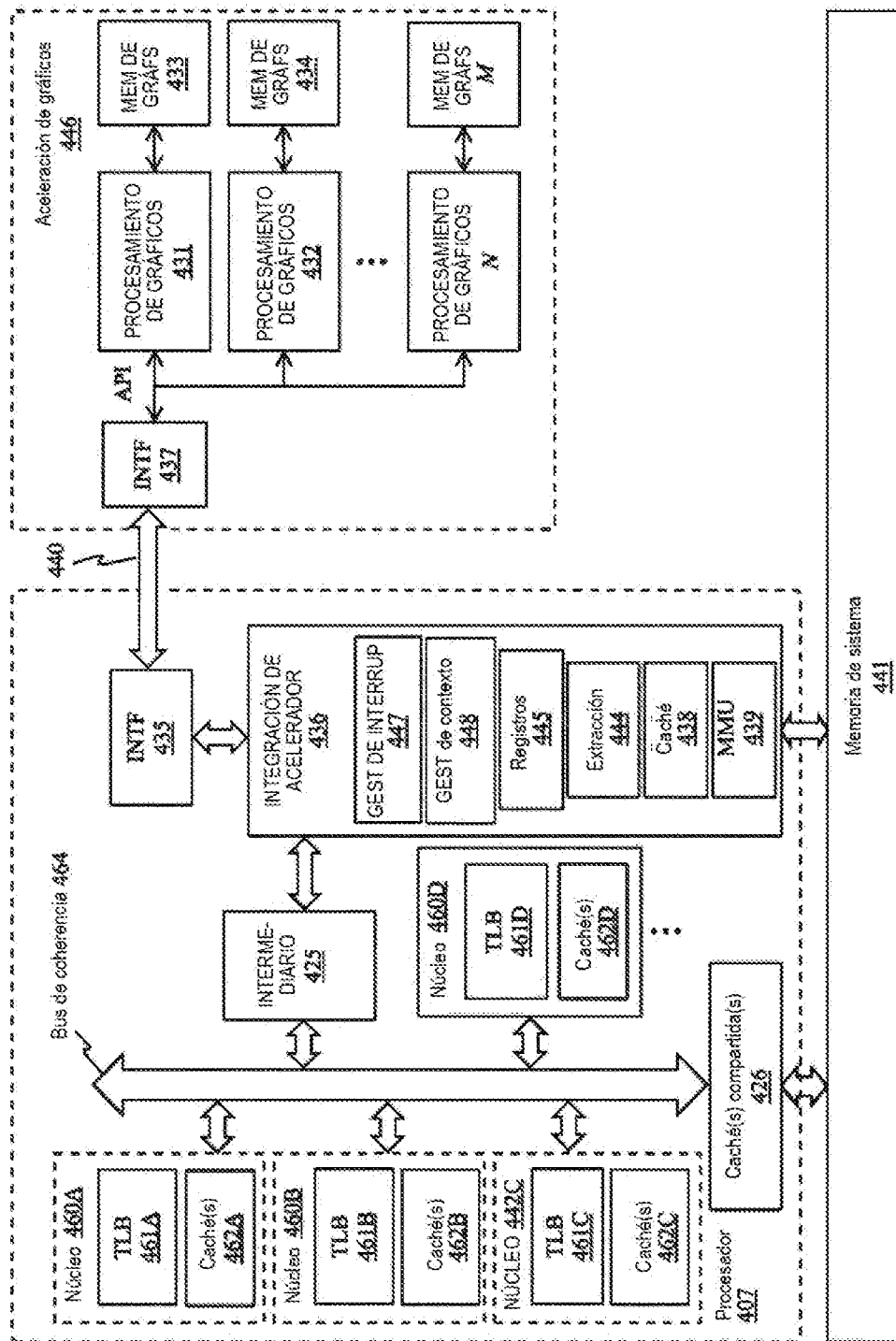


FIG. 4C

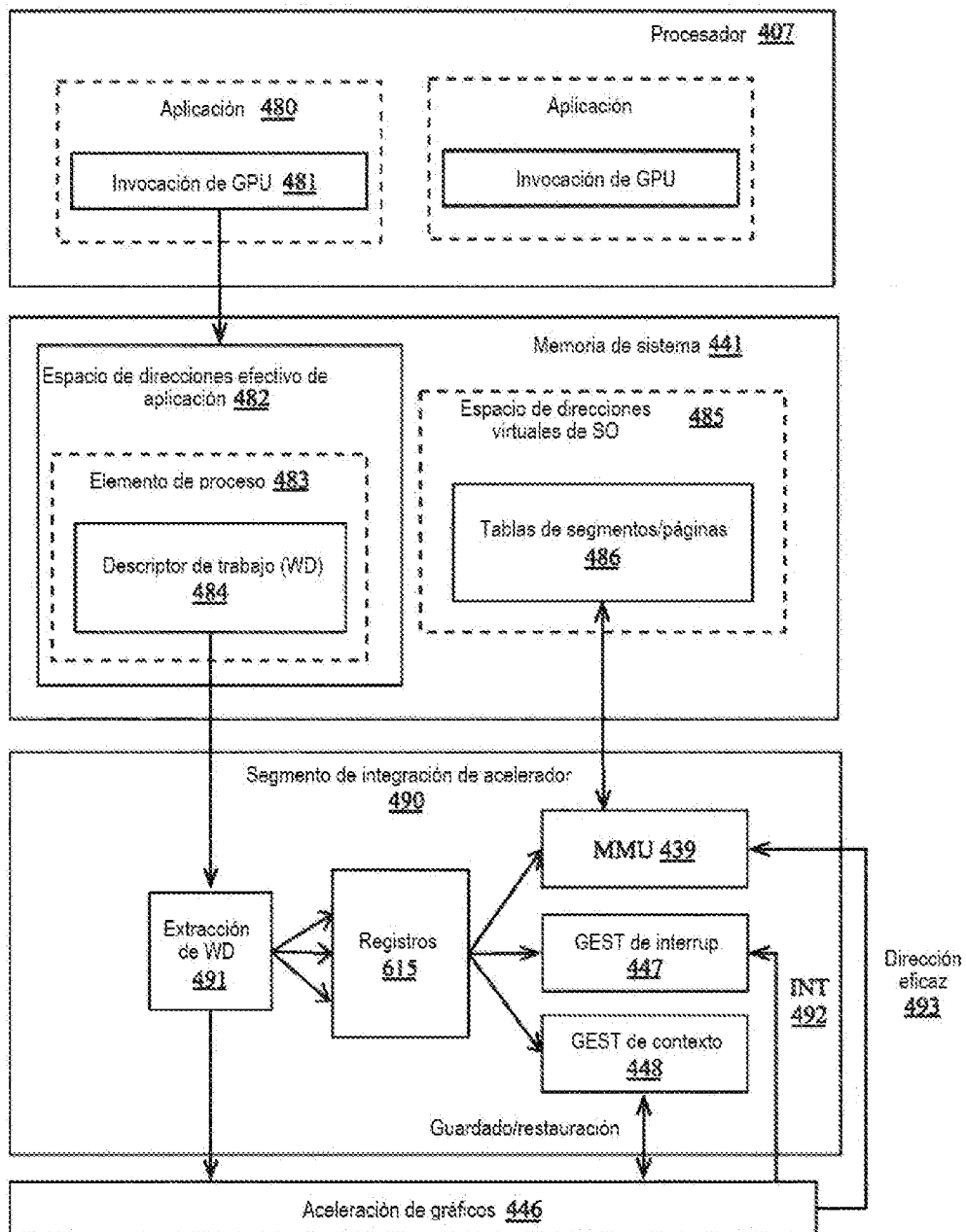


FIG. 4D

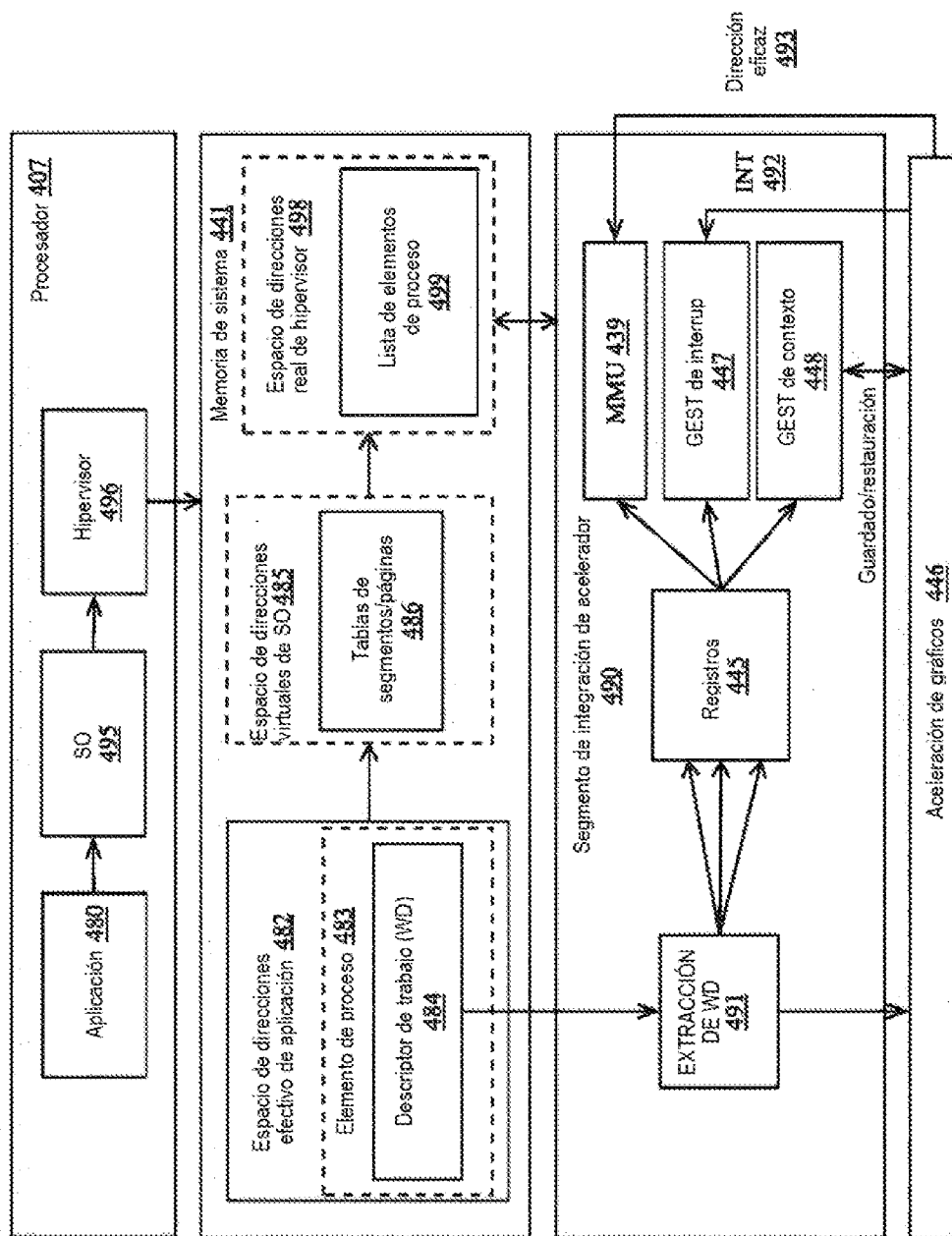


FIG. 4E

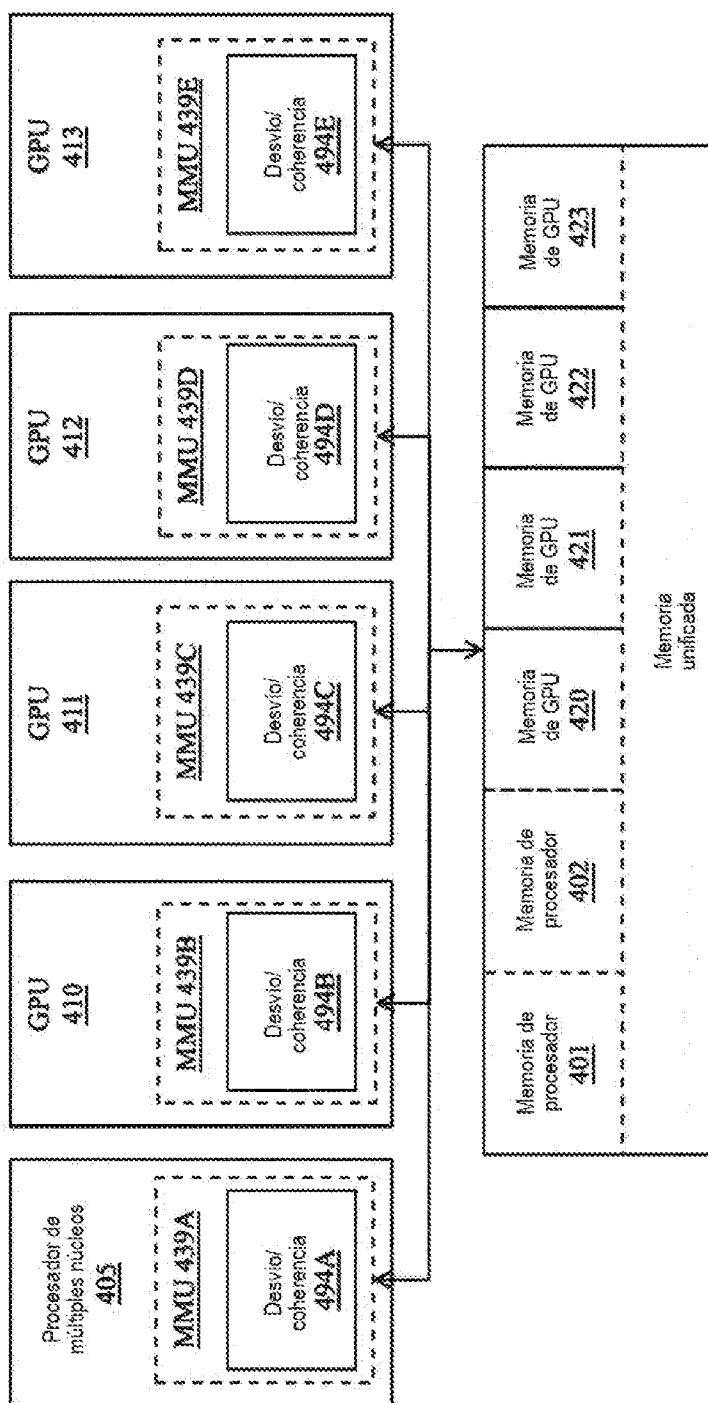


FIG. 4F

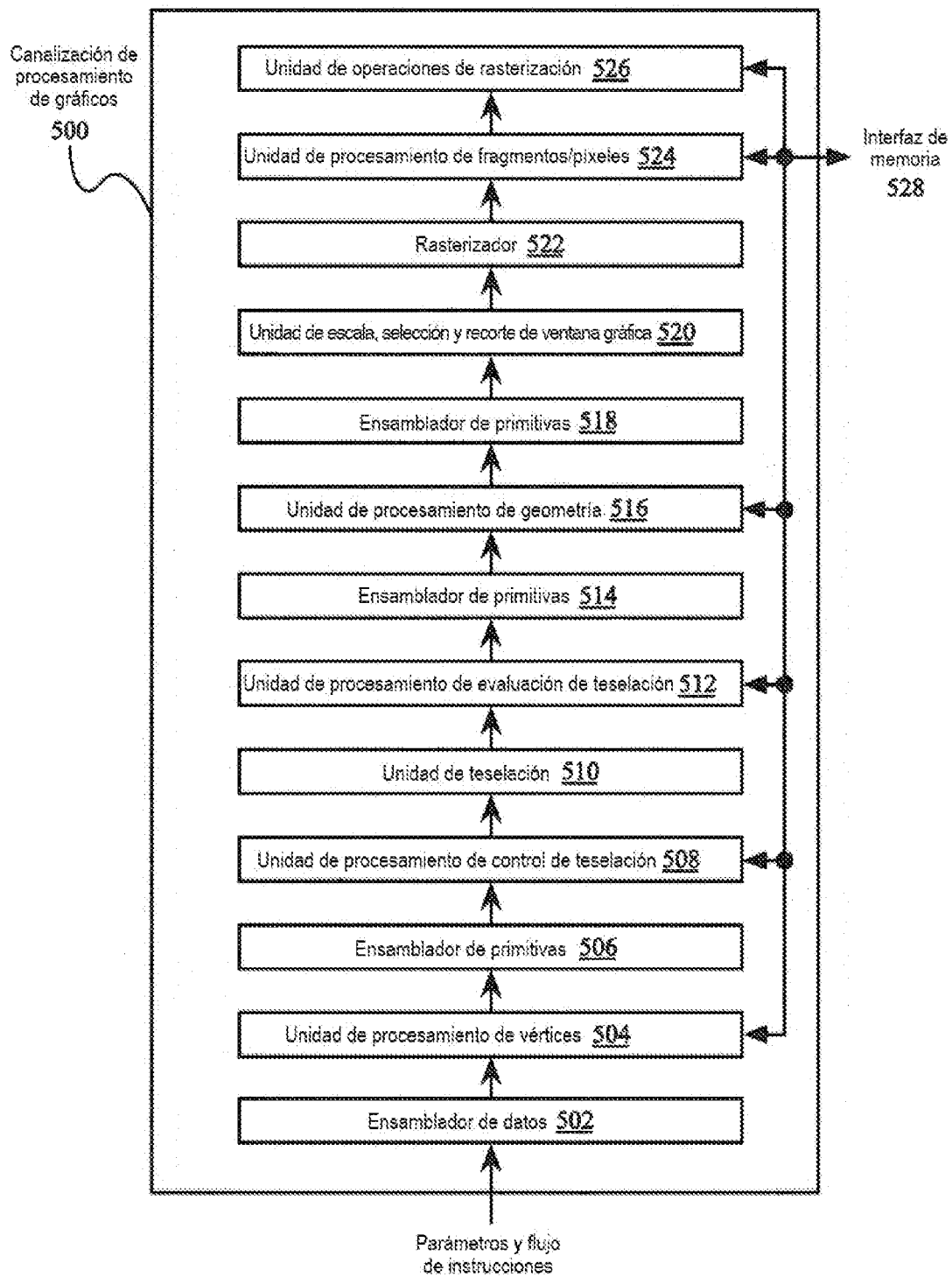


FIG. 5

600

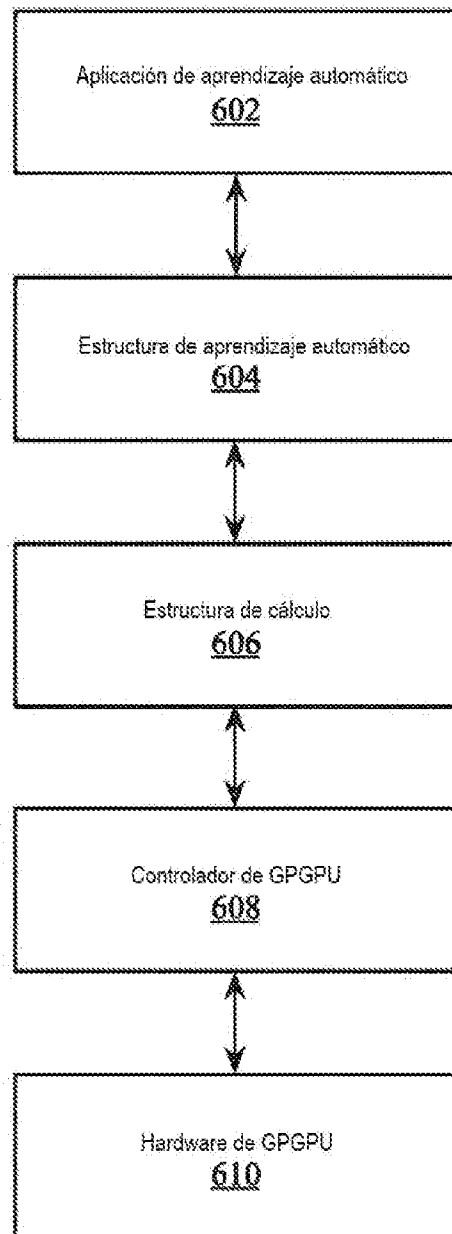


FIG. 6

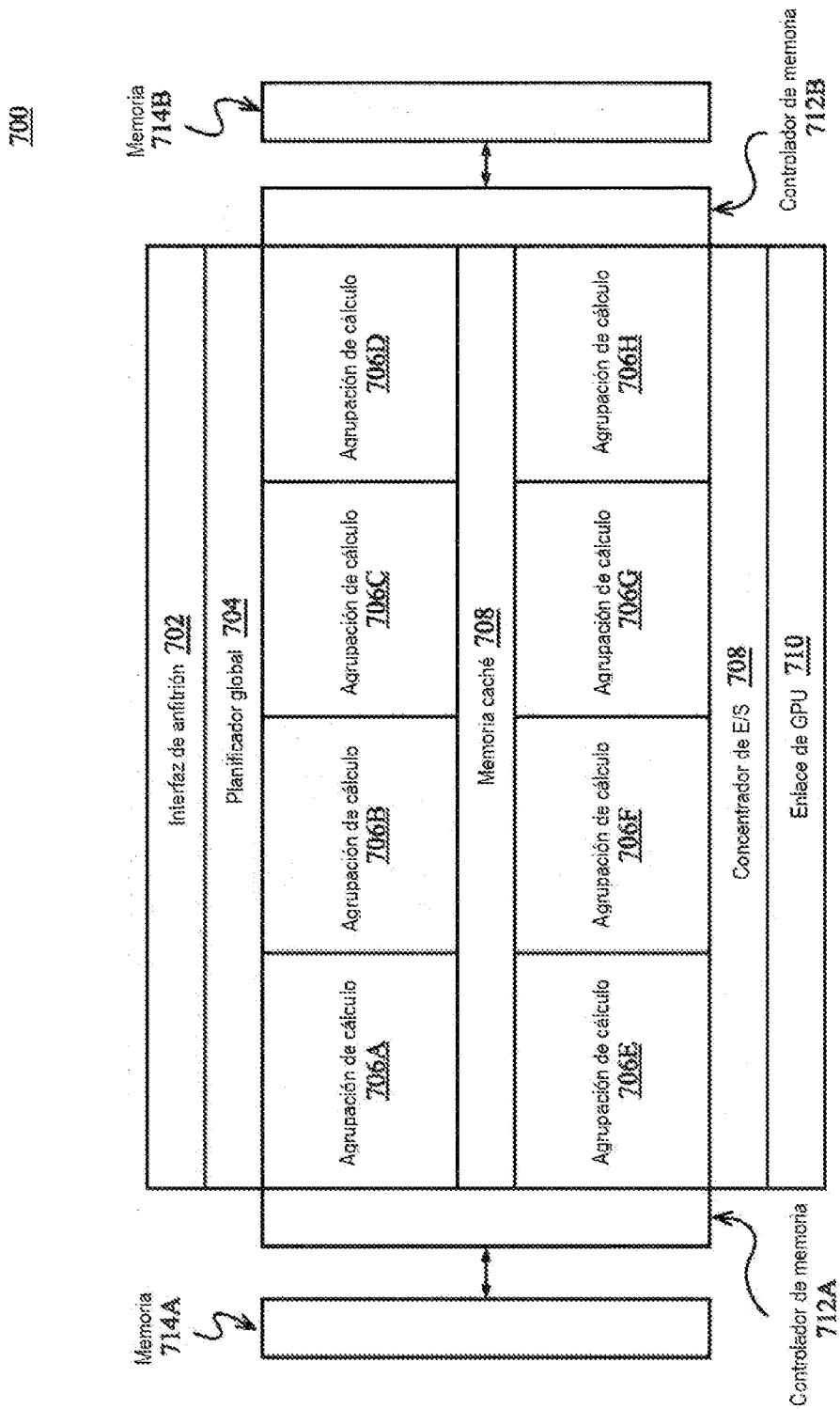


FIG. 7

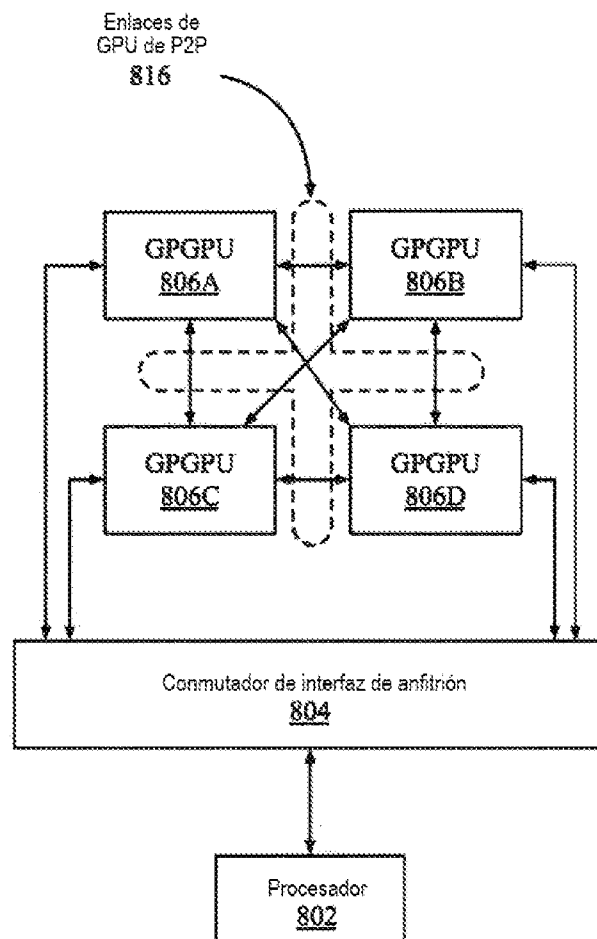


FIG. 8

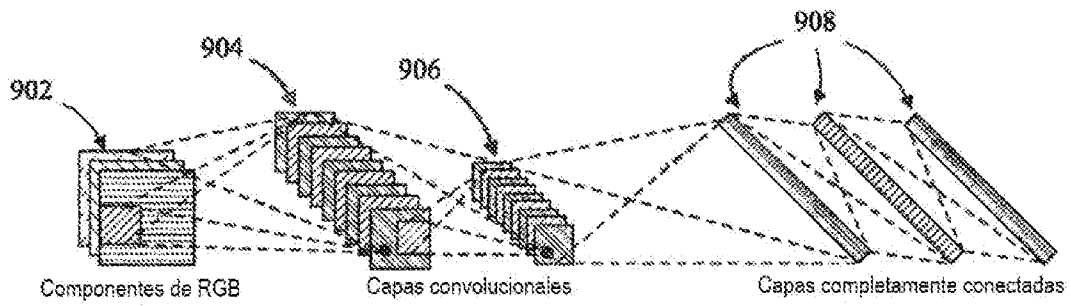


FIG. 9A

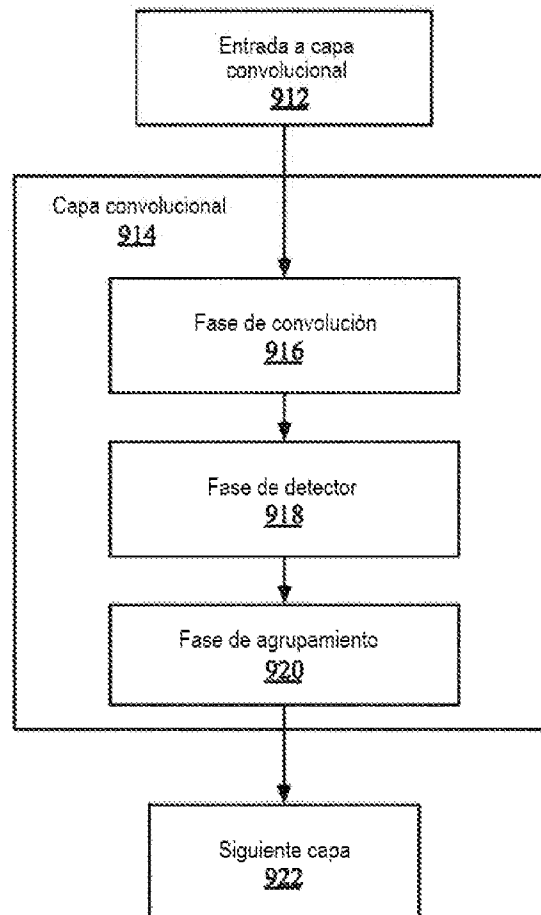


FIG. 9B

1000

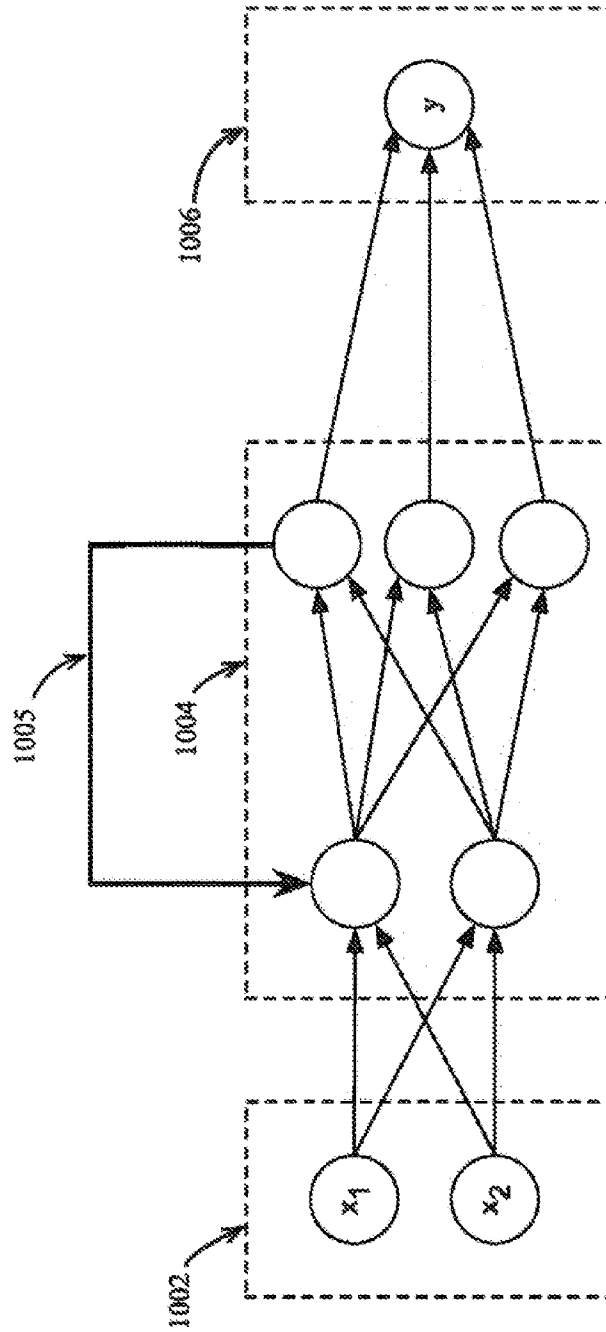


FIG. 10

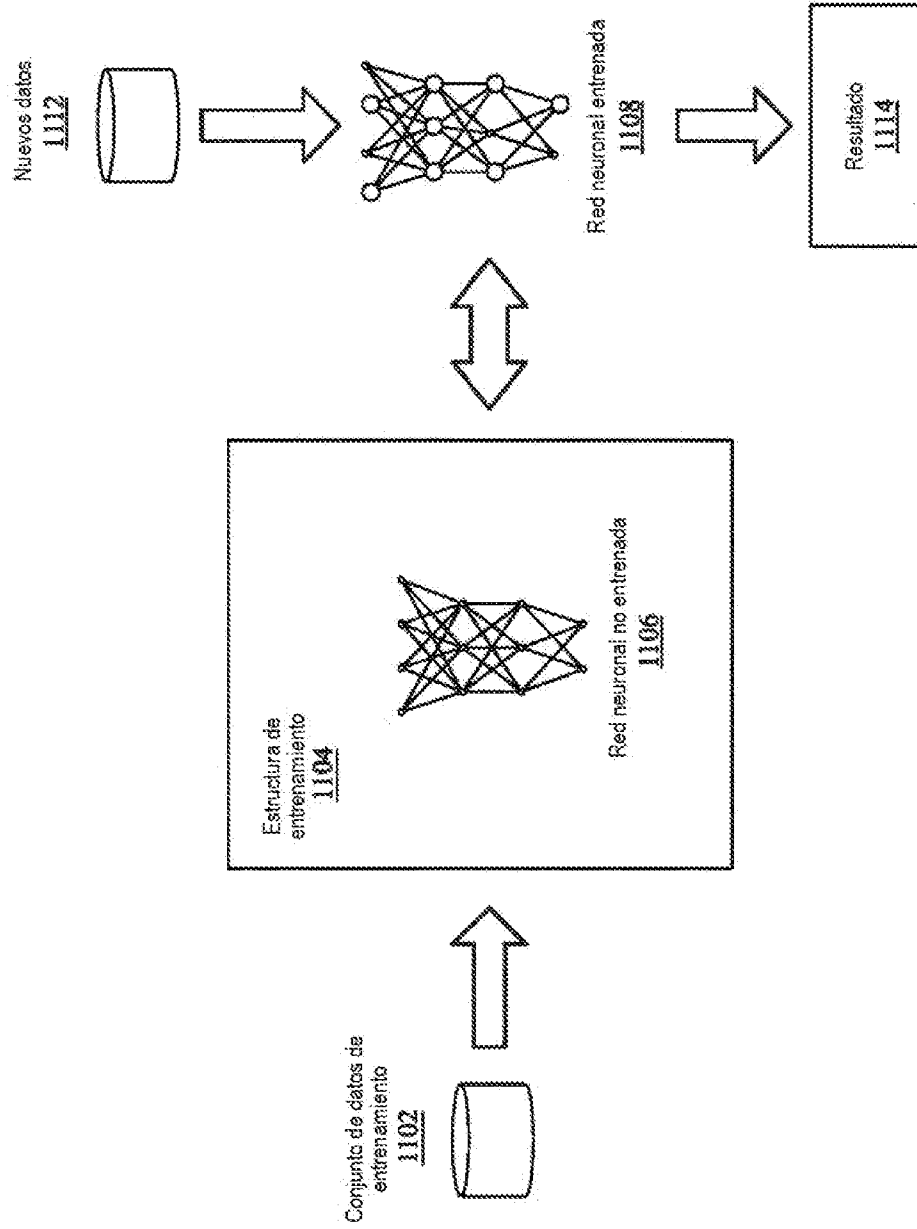


FIG. 11

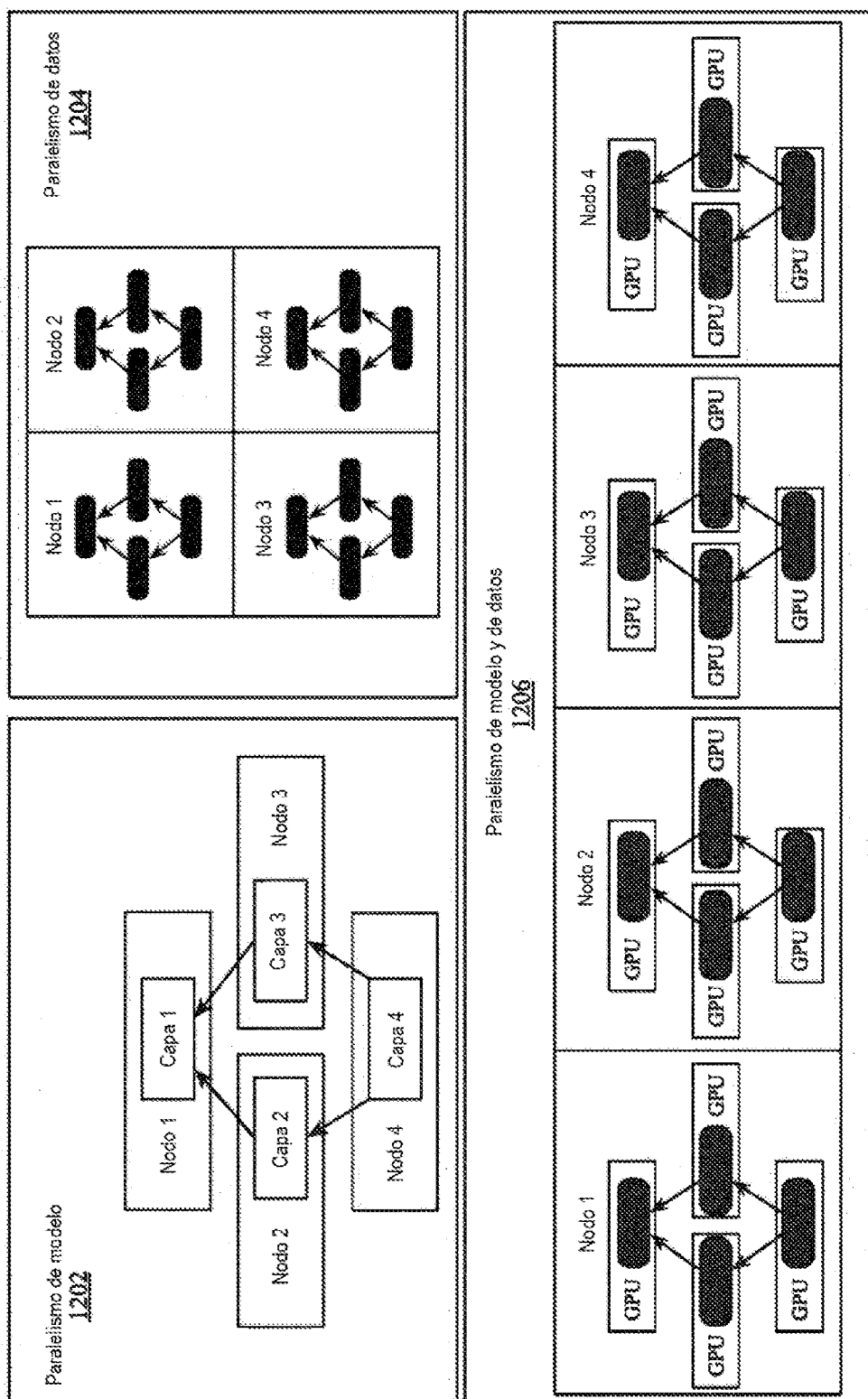


FIG. 12

1300

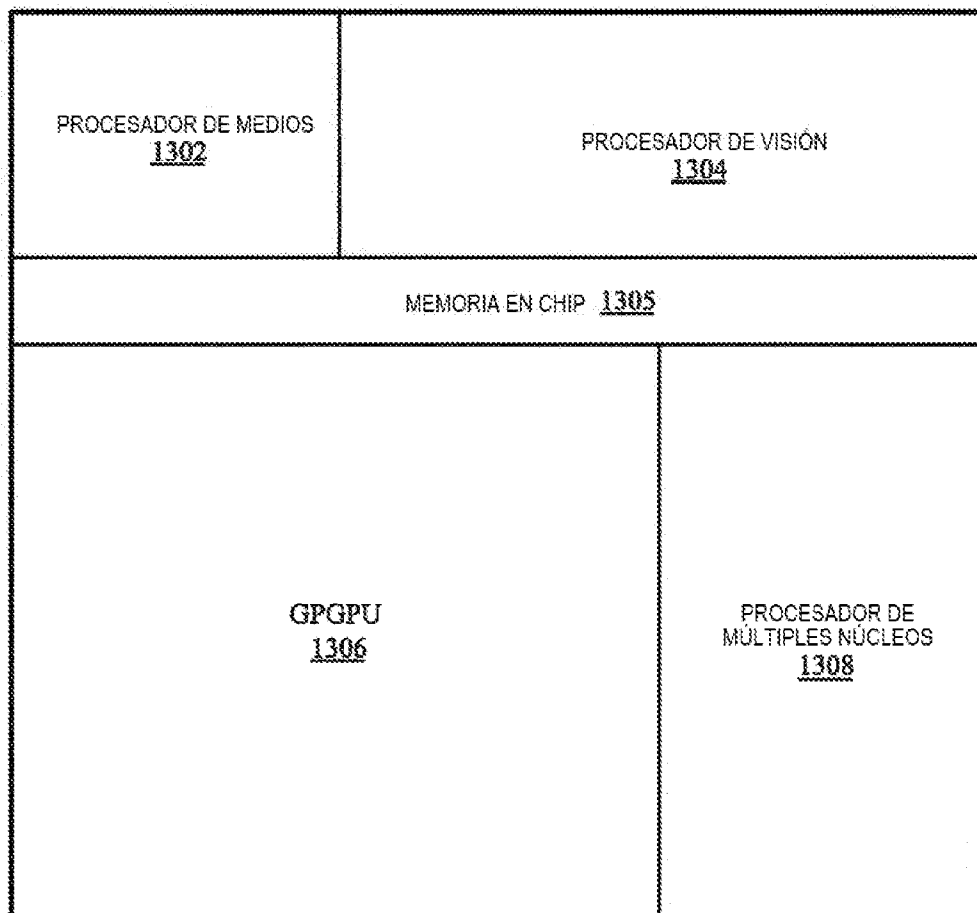


FIG. 13

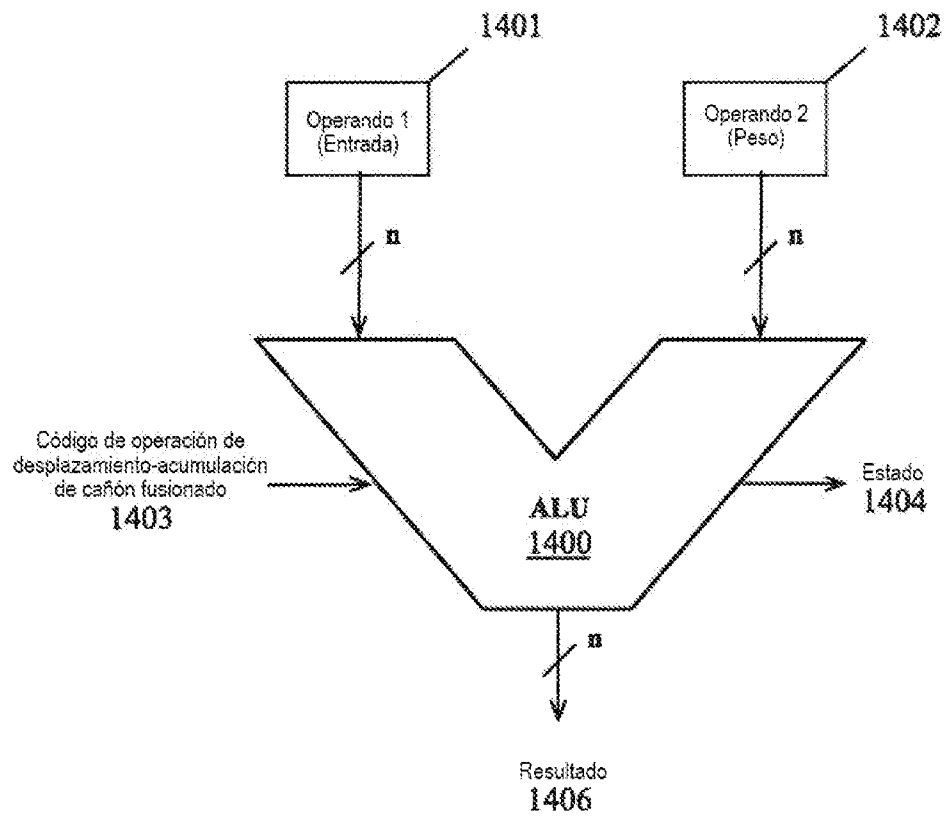


FIG. 14

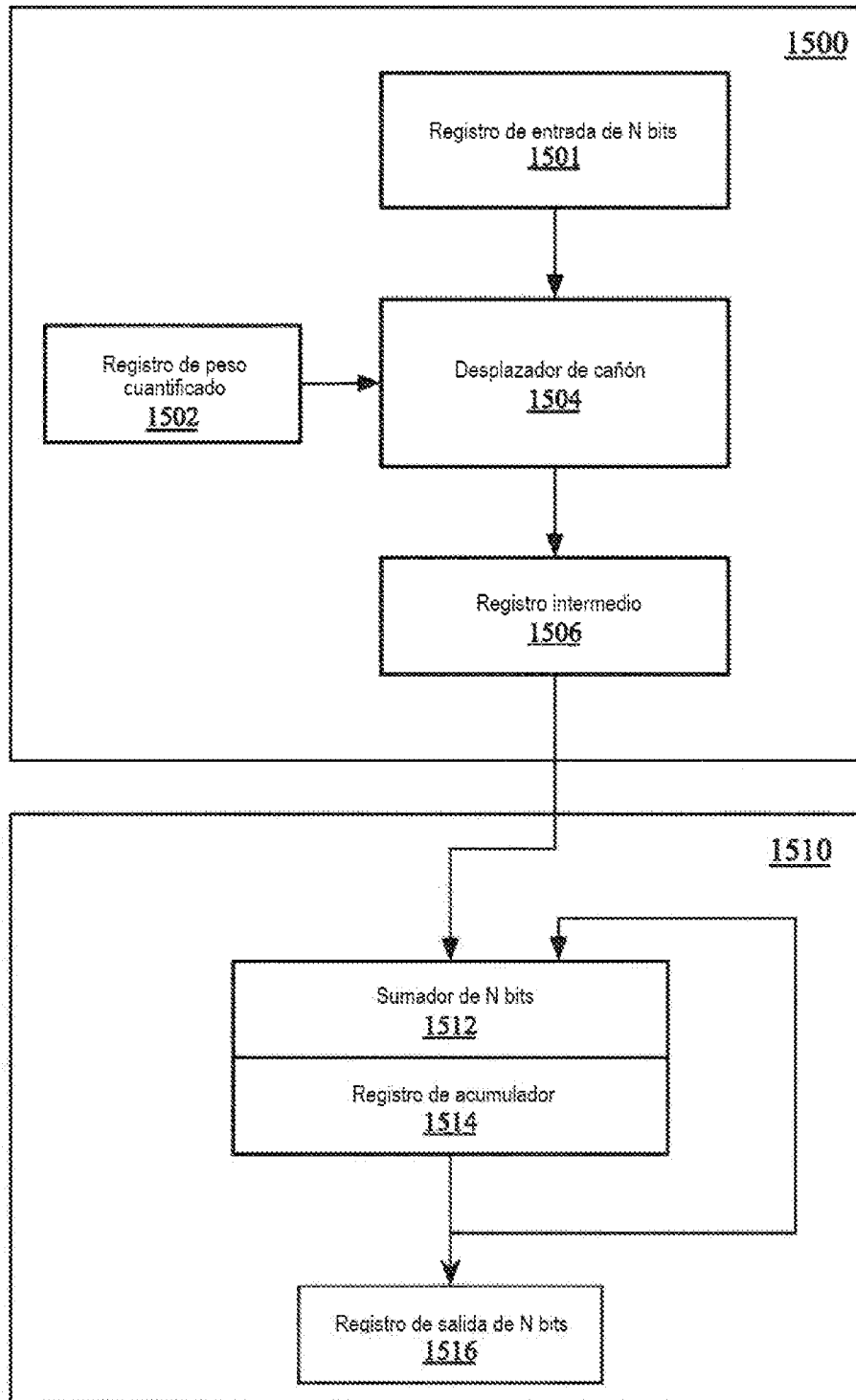


FIG. 15

1600

FIG. 16

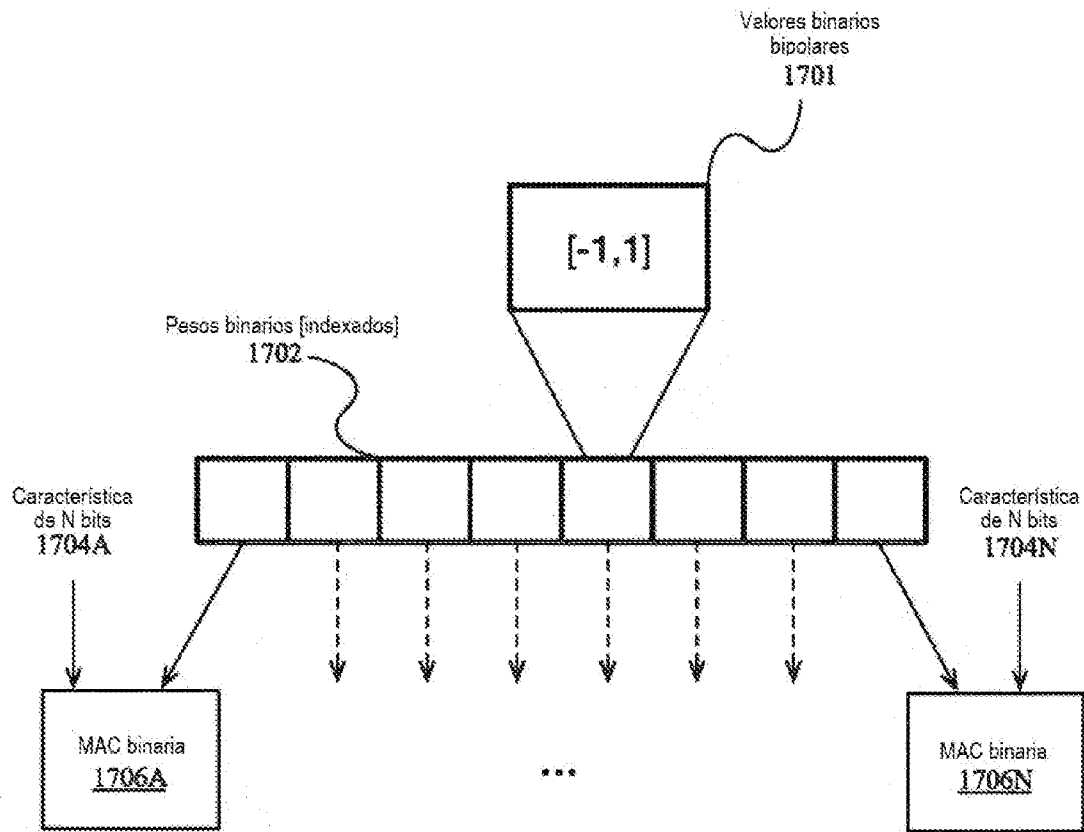


FIG. 17

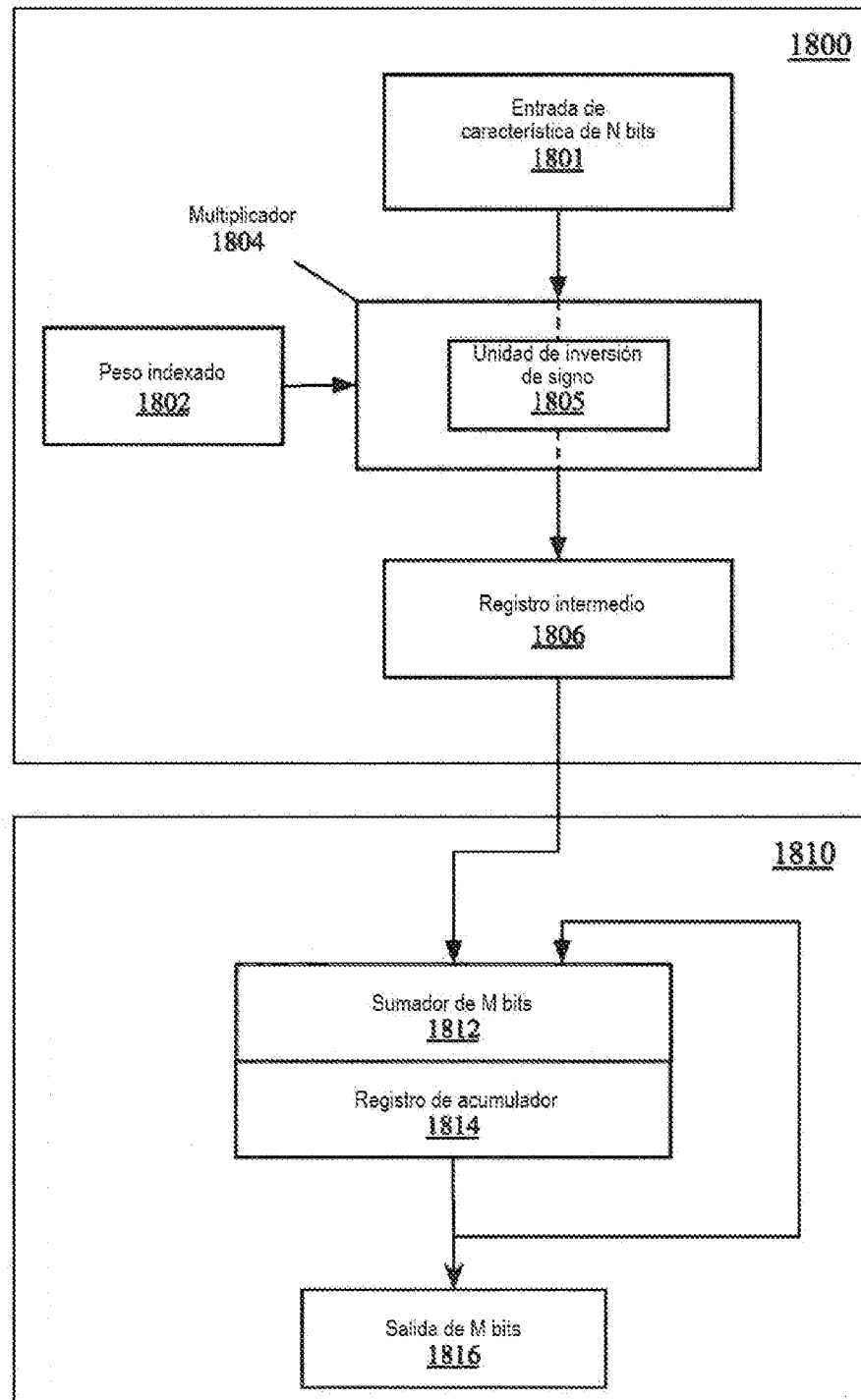


FIG. 18

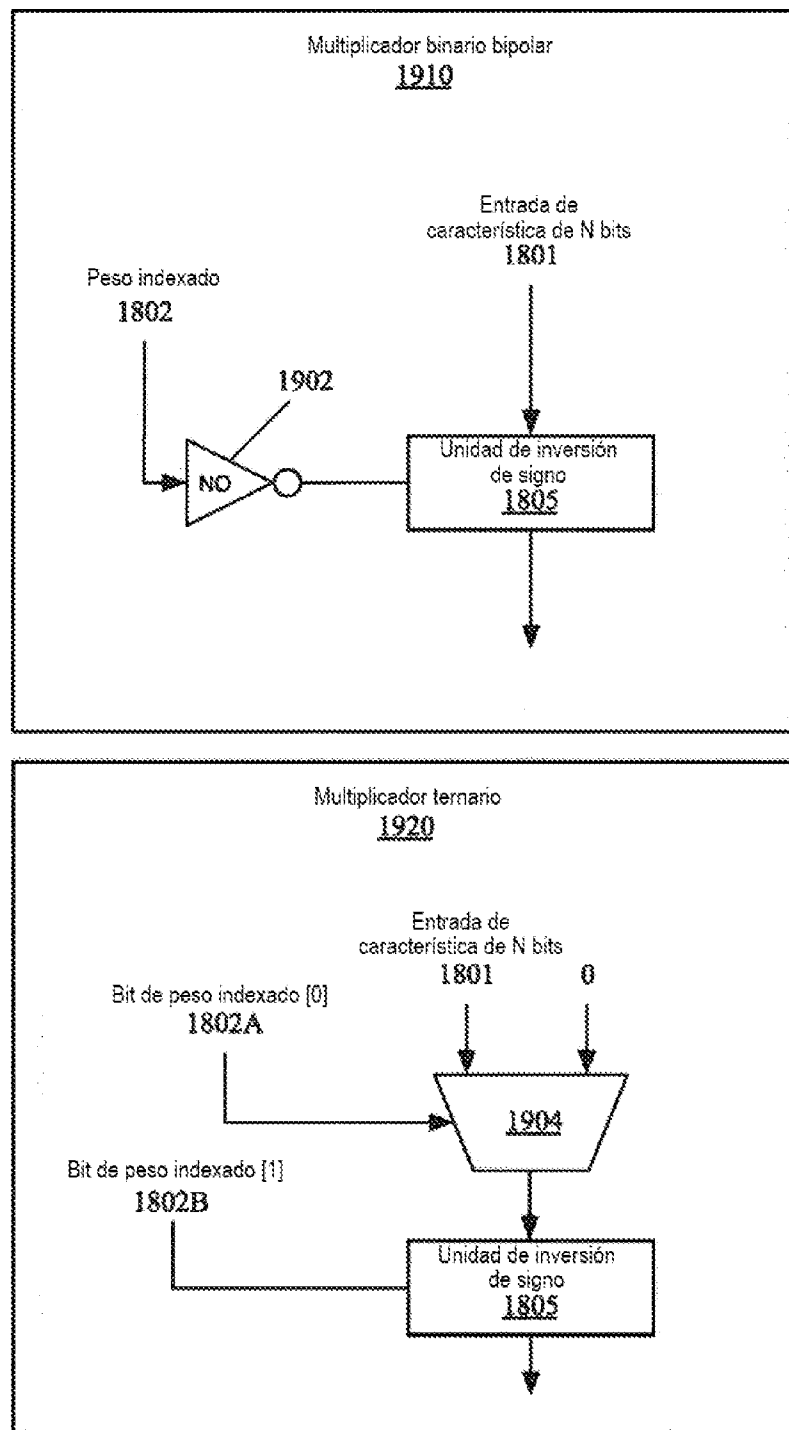


FIG. 19

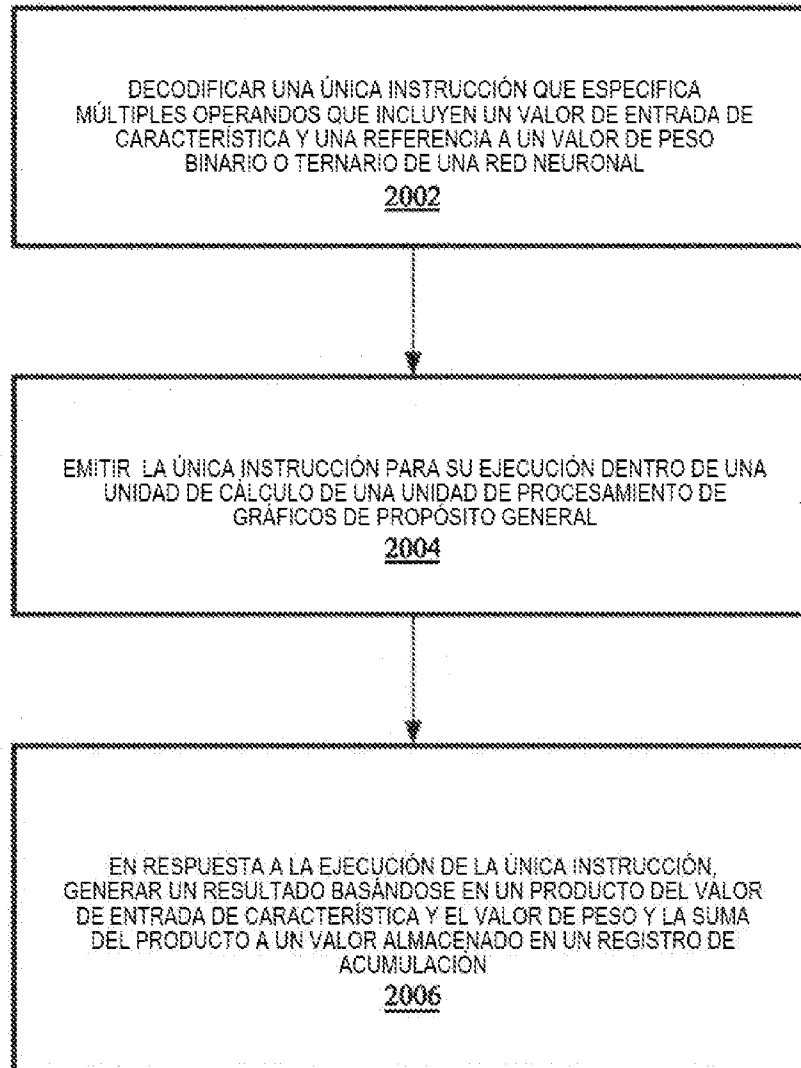
2000

FIG. 20

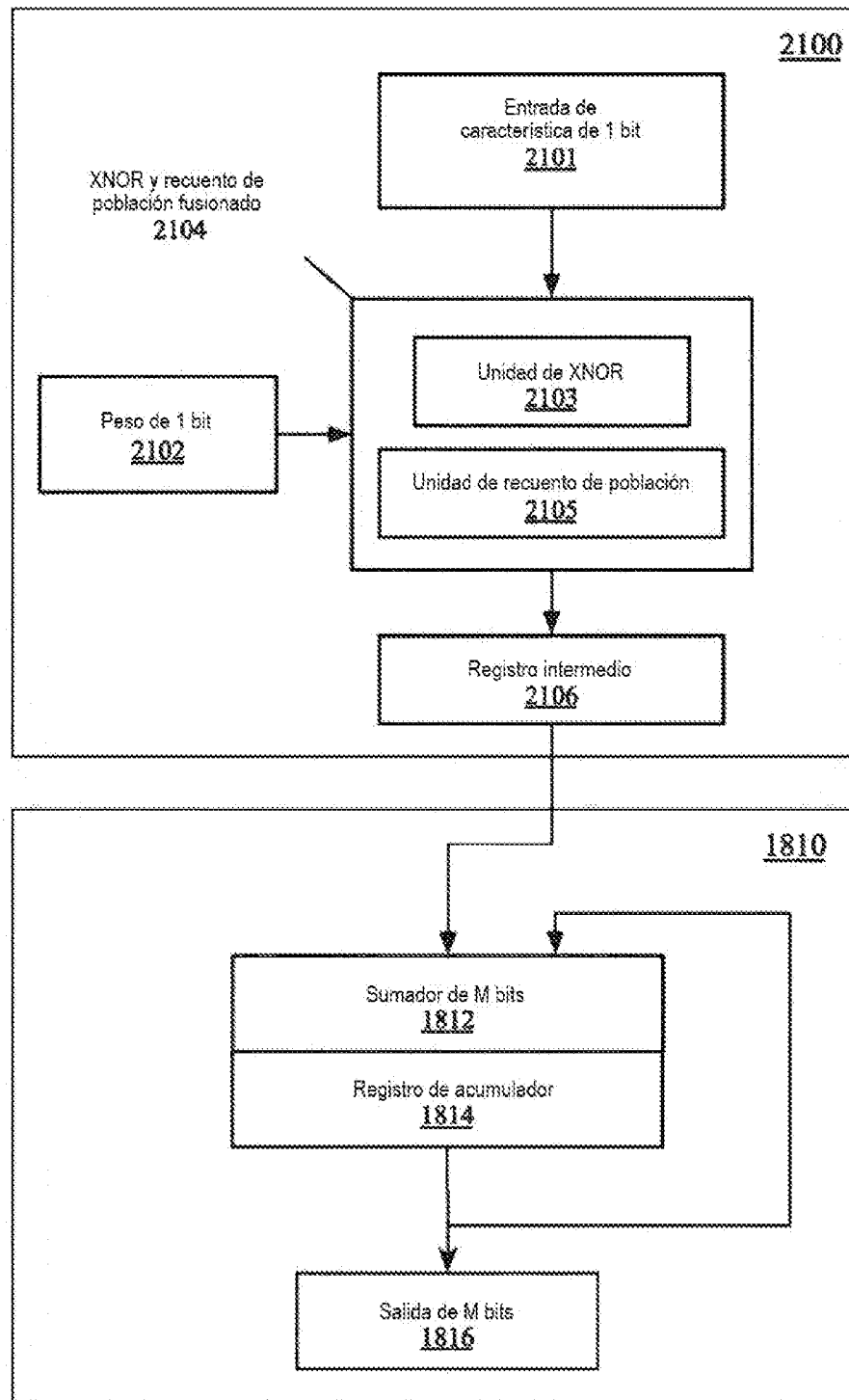


FIG. 21

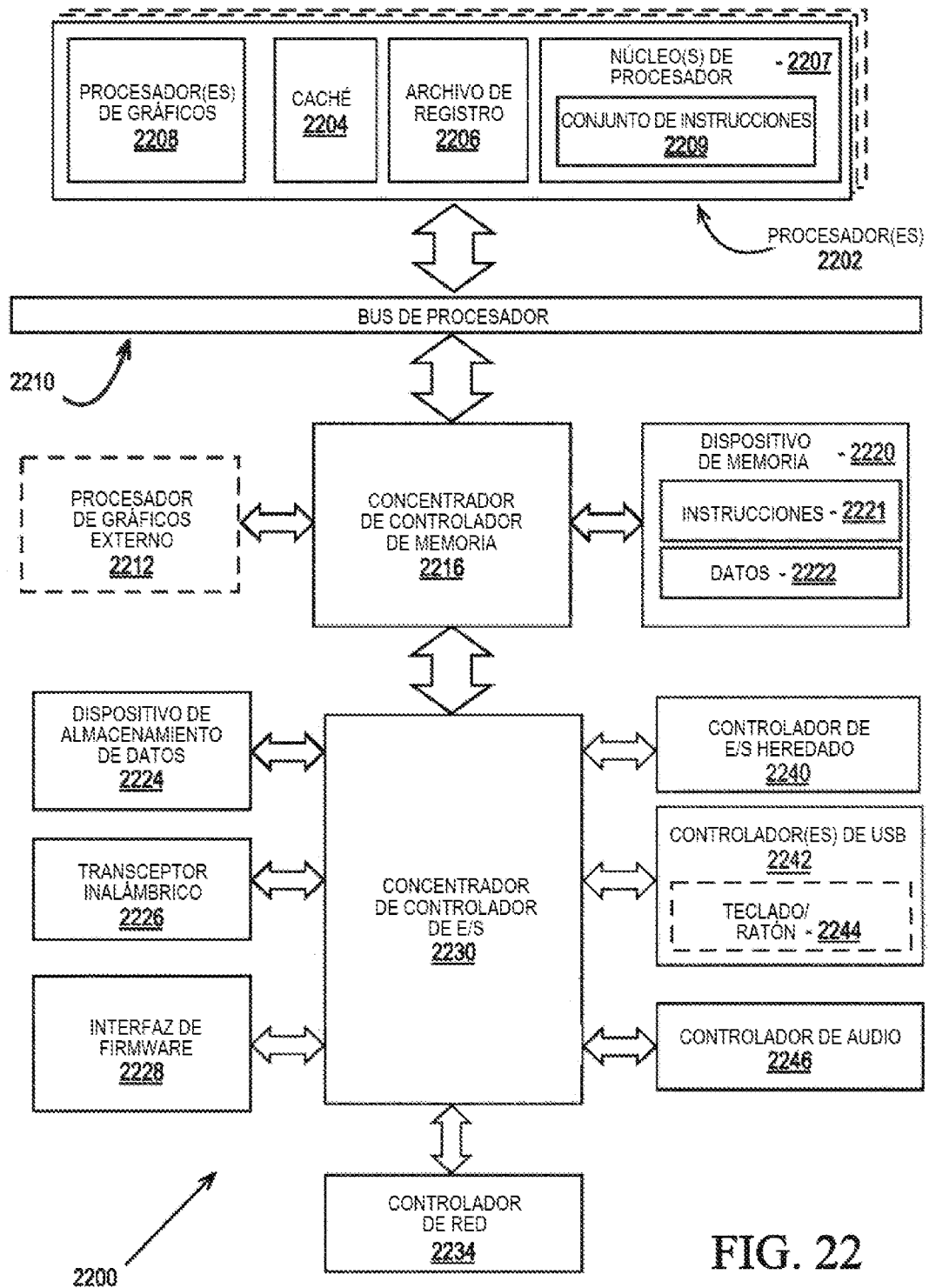


FIG. 22

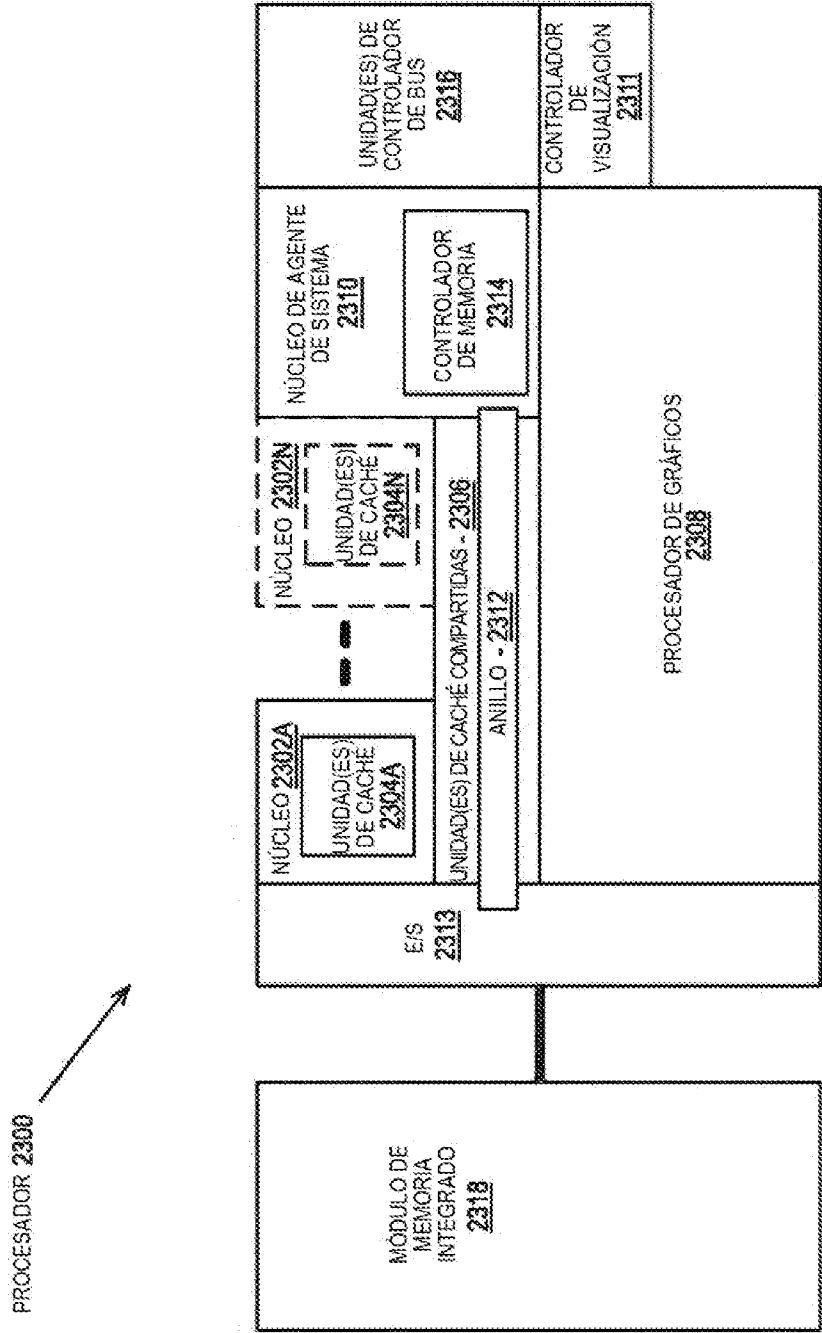


FIG. 23

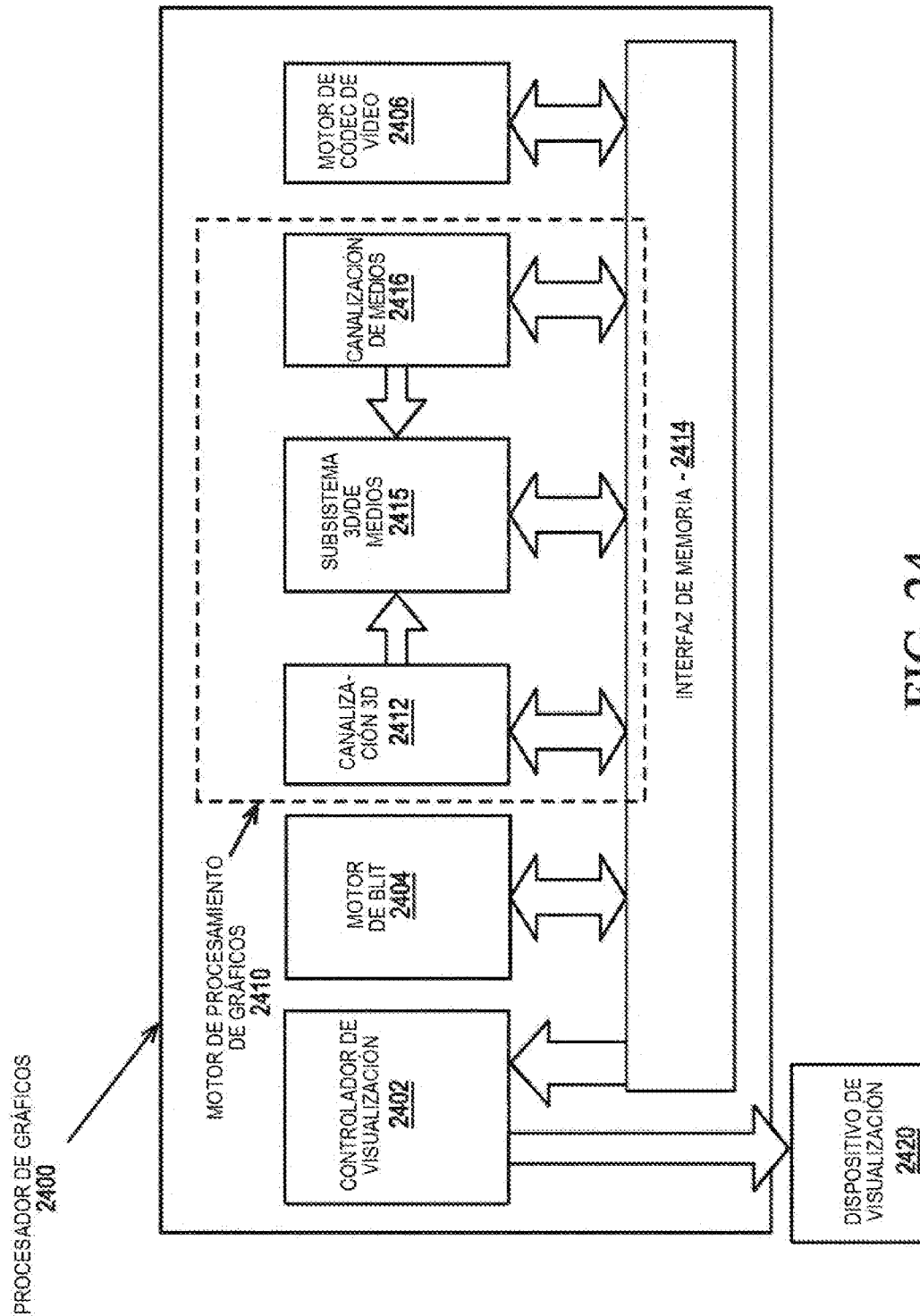


FIG. 24

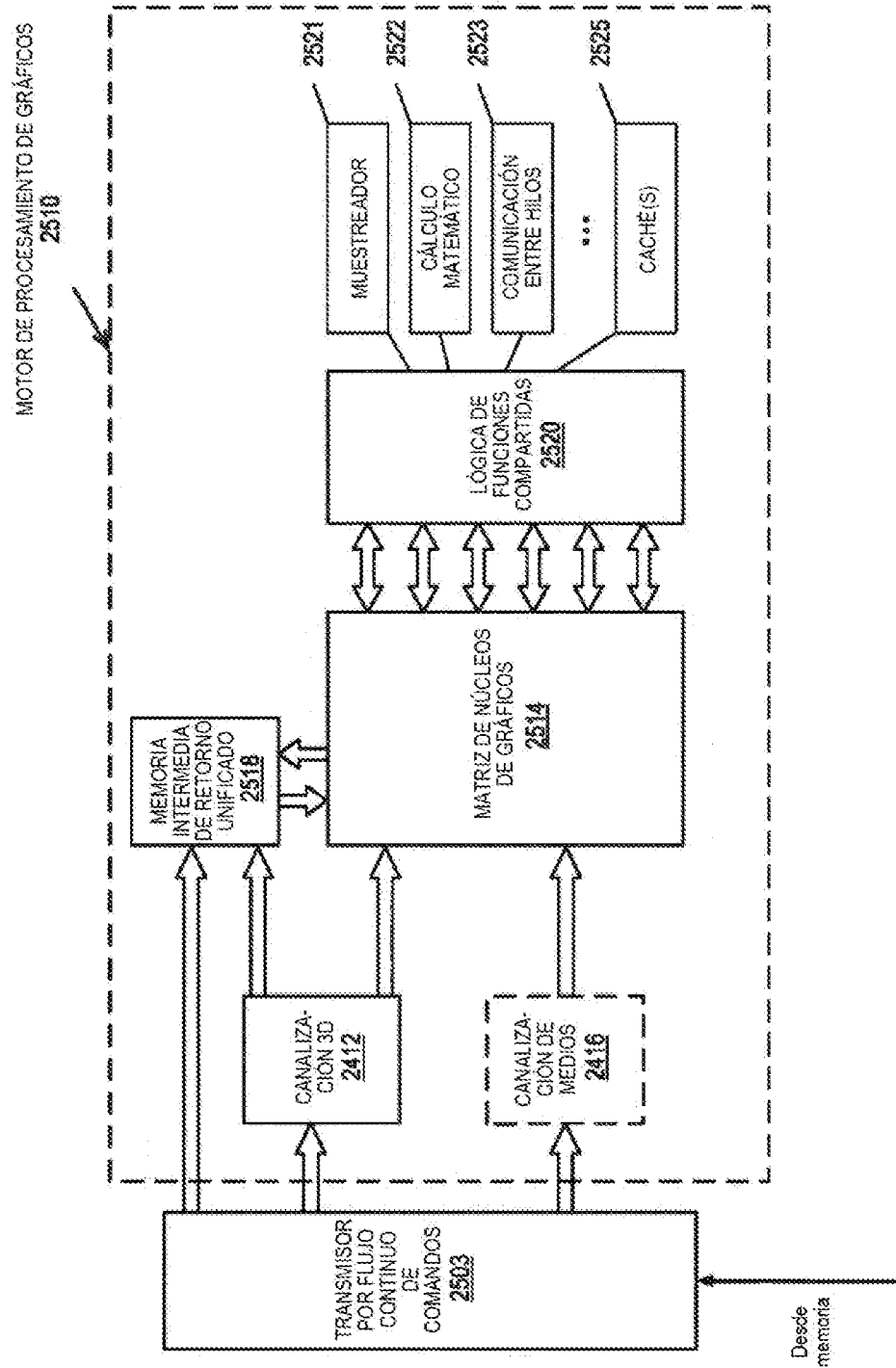


FIG. 25

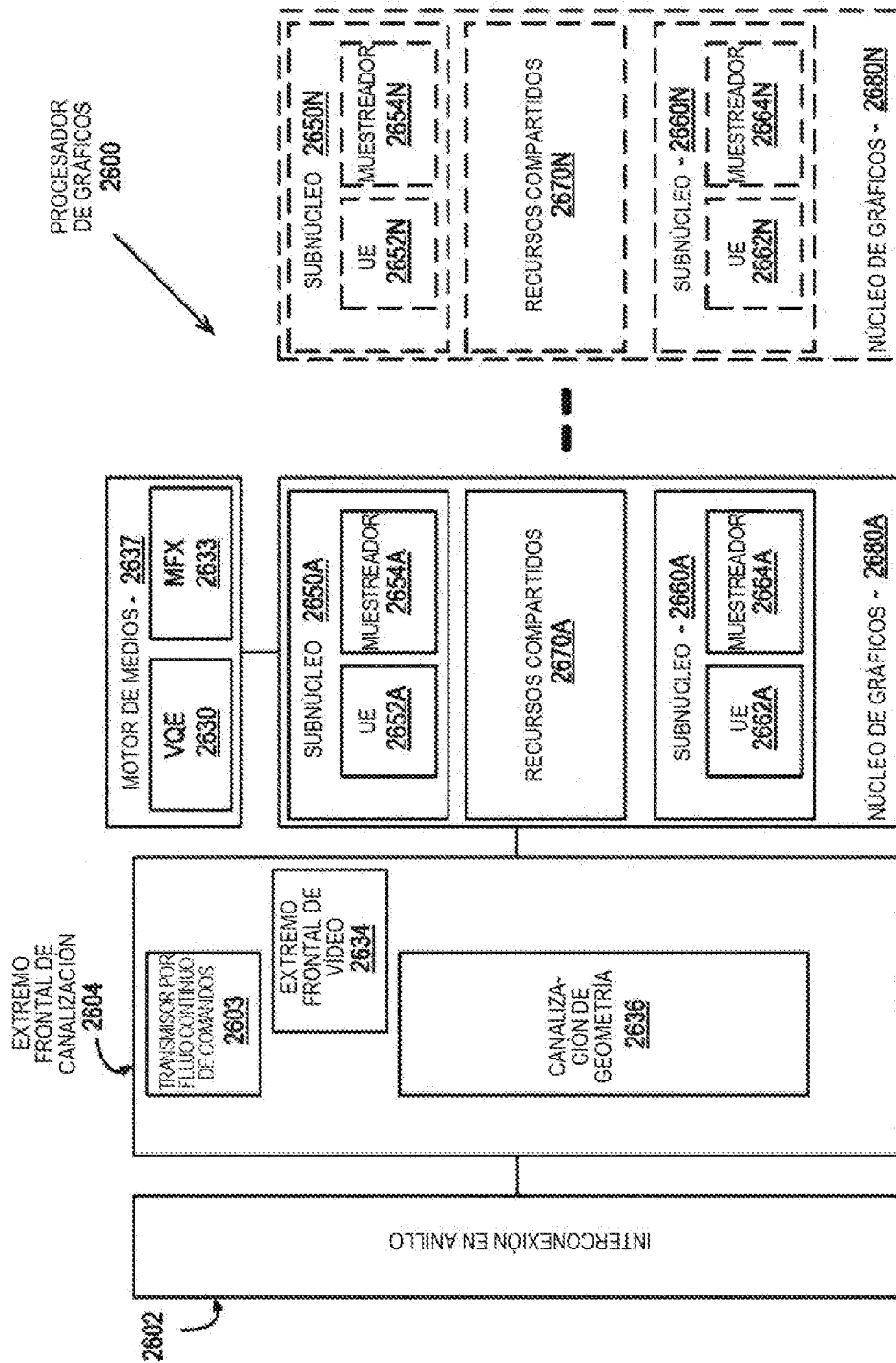


FIG. 26

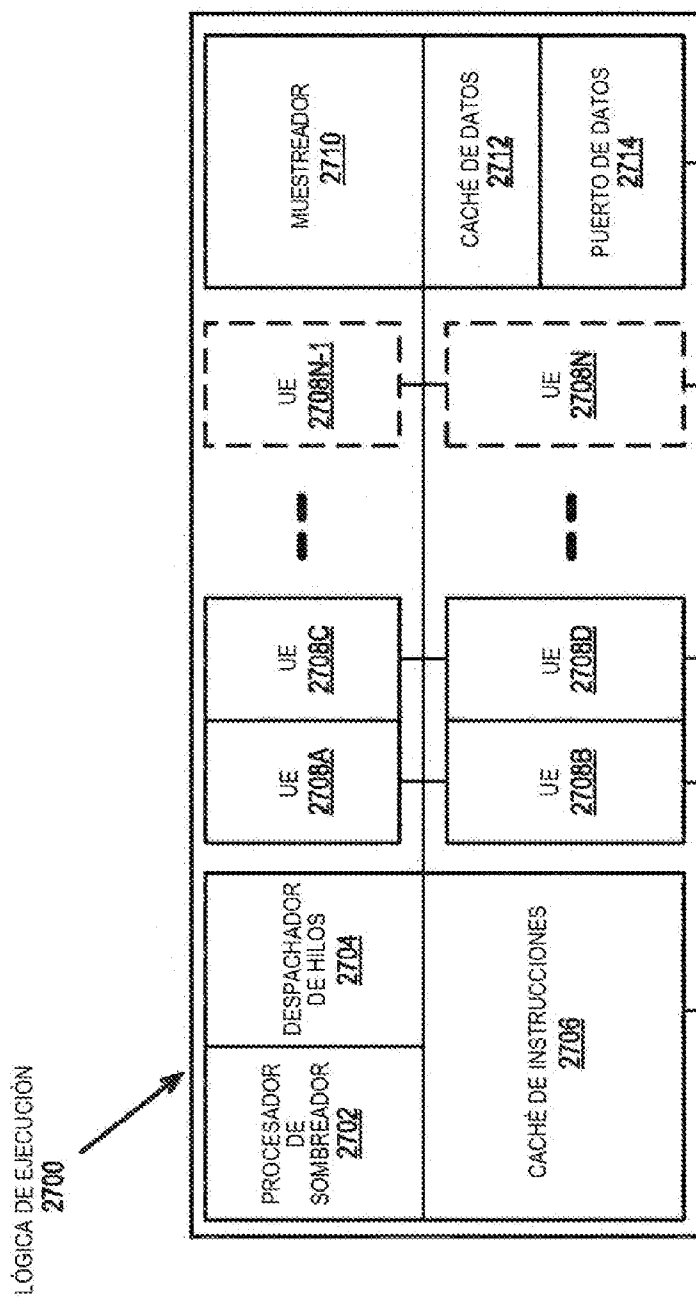
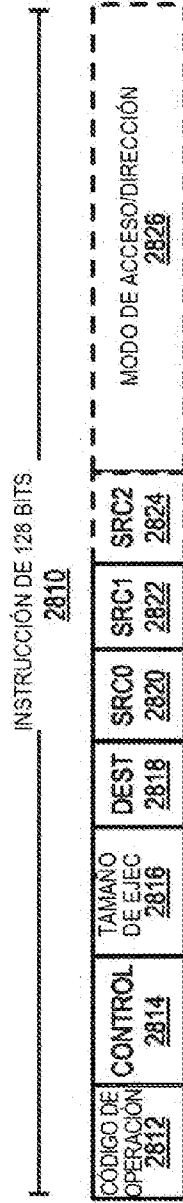


FIG. 27

FORMATOS DE INSTRUCCIÓN DE PROCESADOR DE GRÁFICOS

2800



DESCODIFICACIÓN DE CÓDIGO DE OPERACIÓN

2840

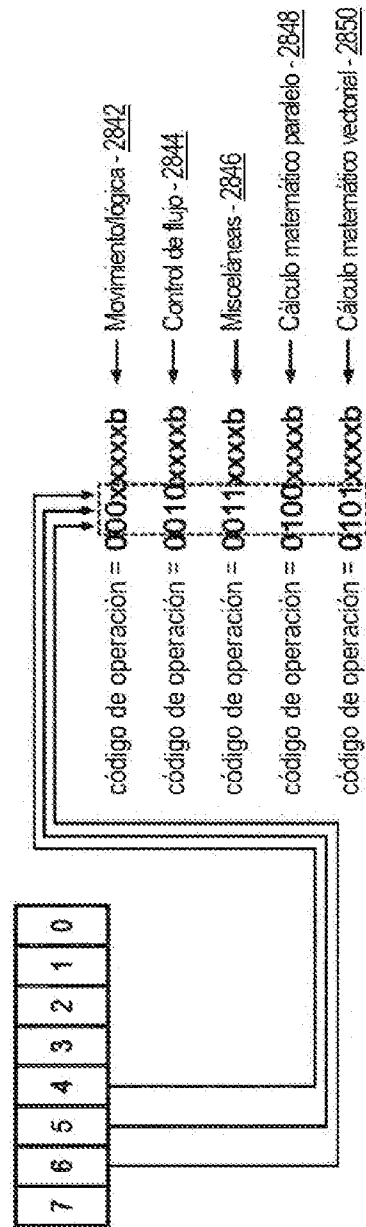


FIG. 28

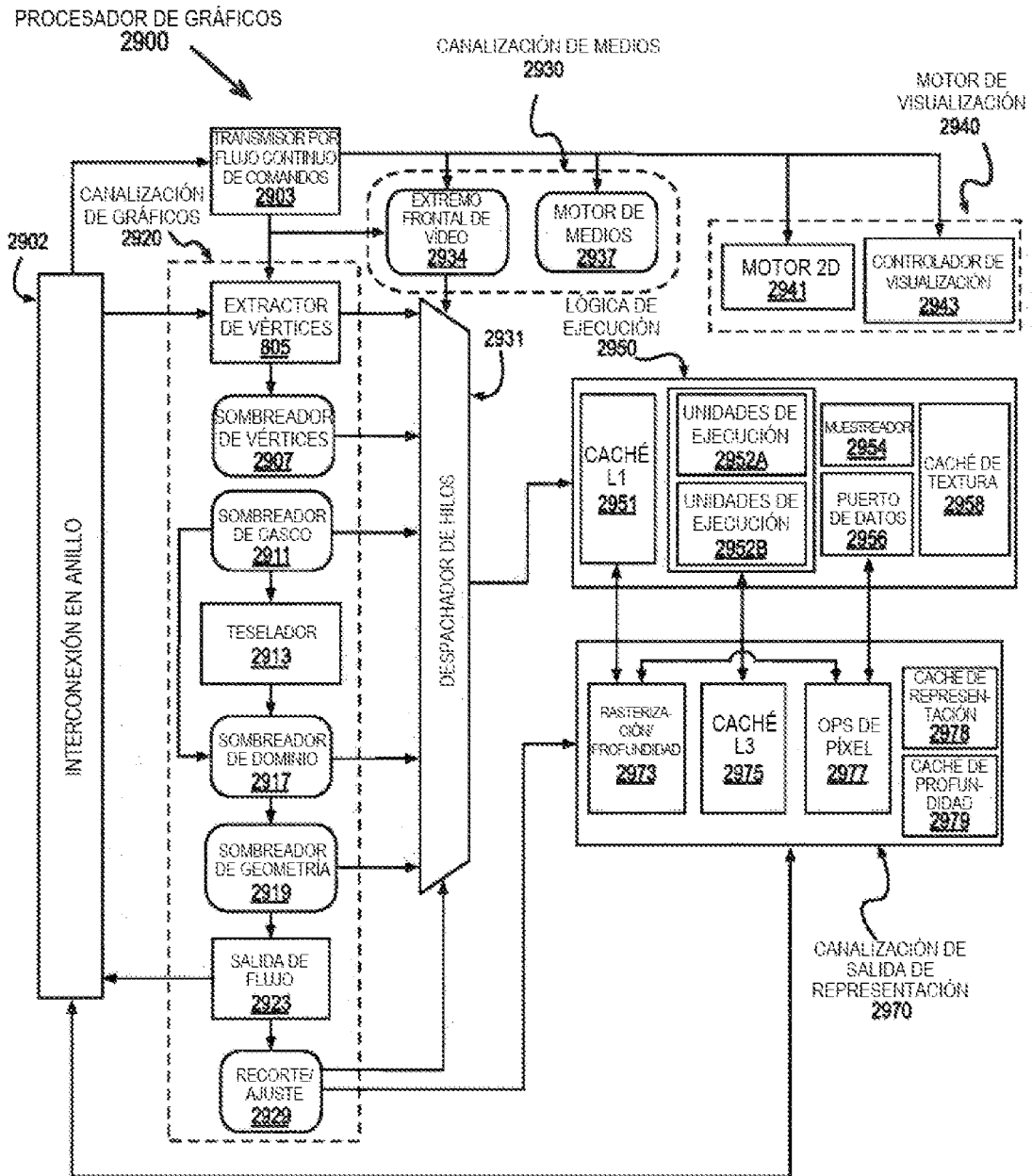
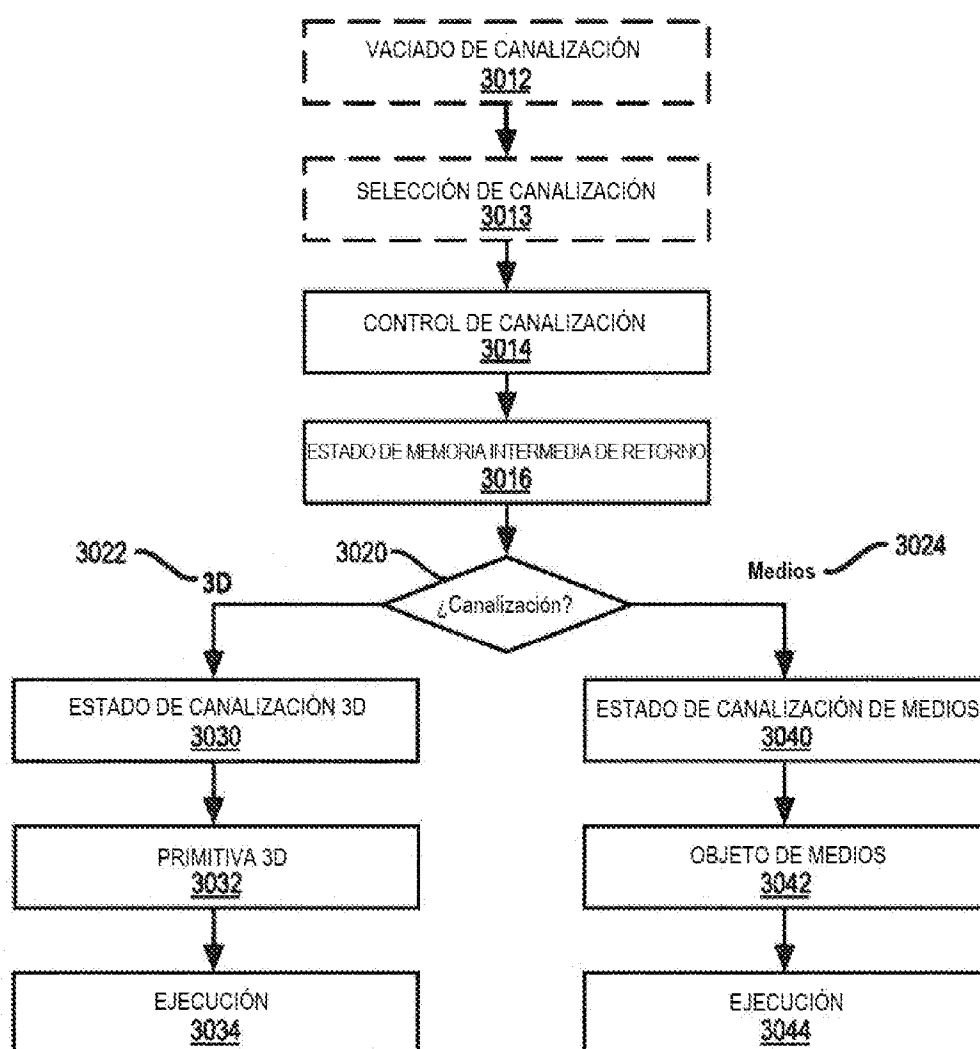


FIG. 29

FIG. 30A FORMATO DE COMANDO DE PROCESADOR DE GRÁFICOS



FIG. 30B SECUENCIA DE COMANDOS DE PROCESADOR DE GRÁFICOS



SISTEMA DE PROCESAMIENTO DE DATOS-3100

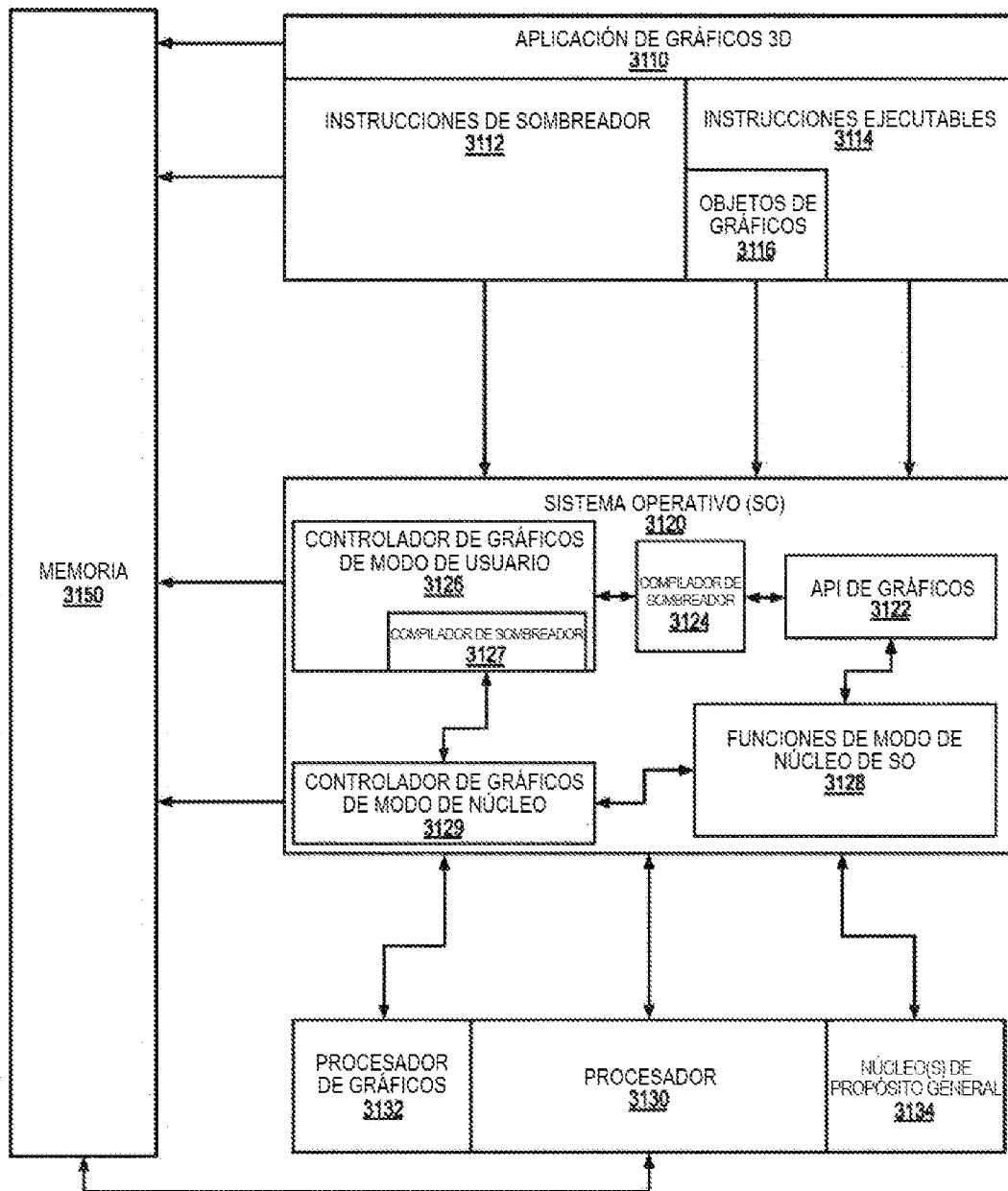


FIG. 31

DESARROLLO DE NÚCLEO DE IP - 3200

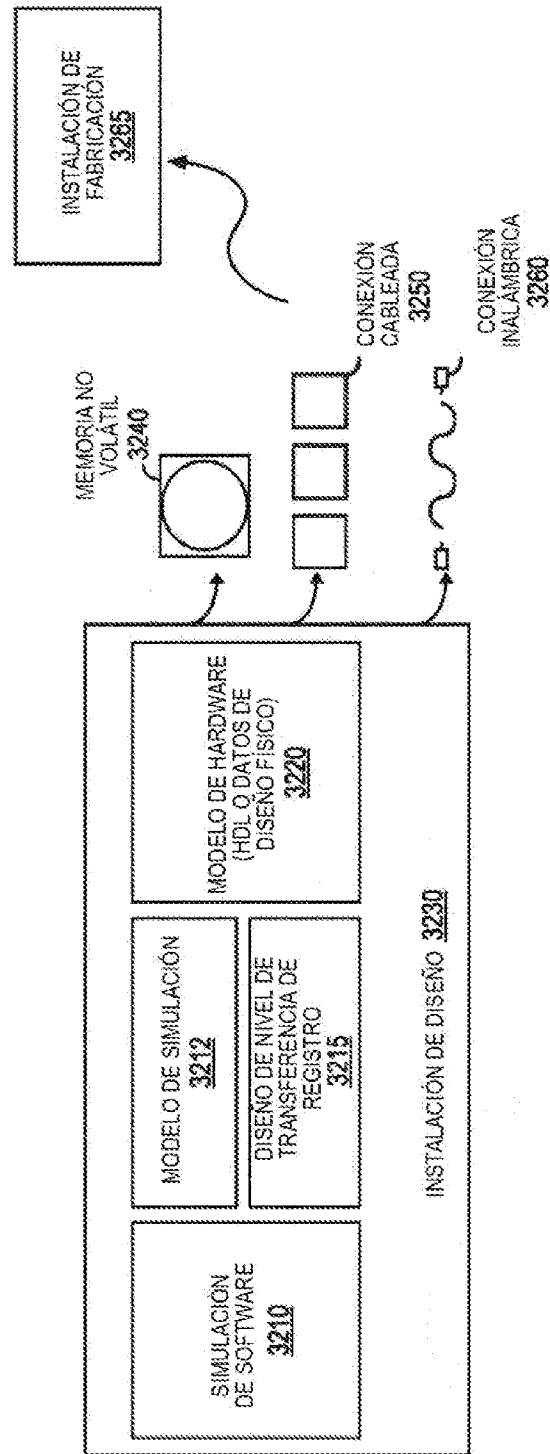


FIG. 32

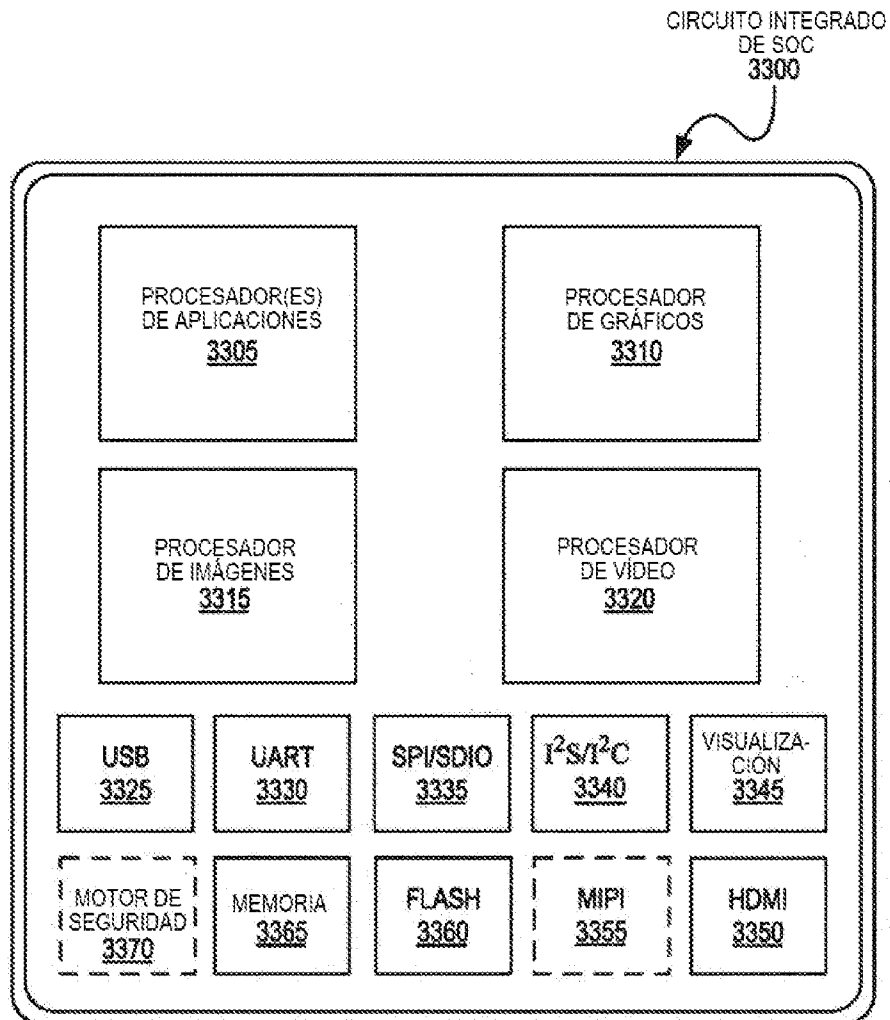


FIG. 33

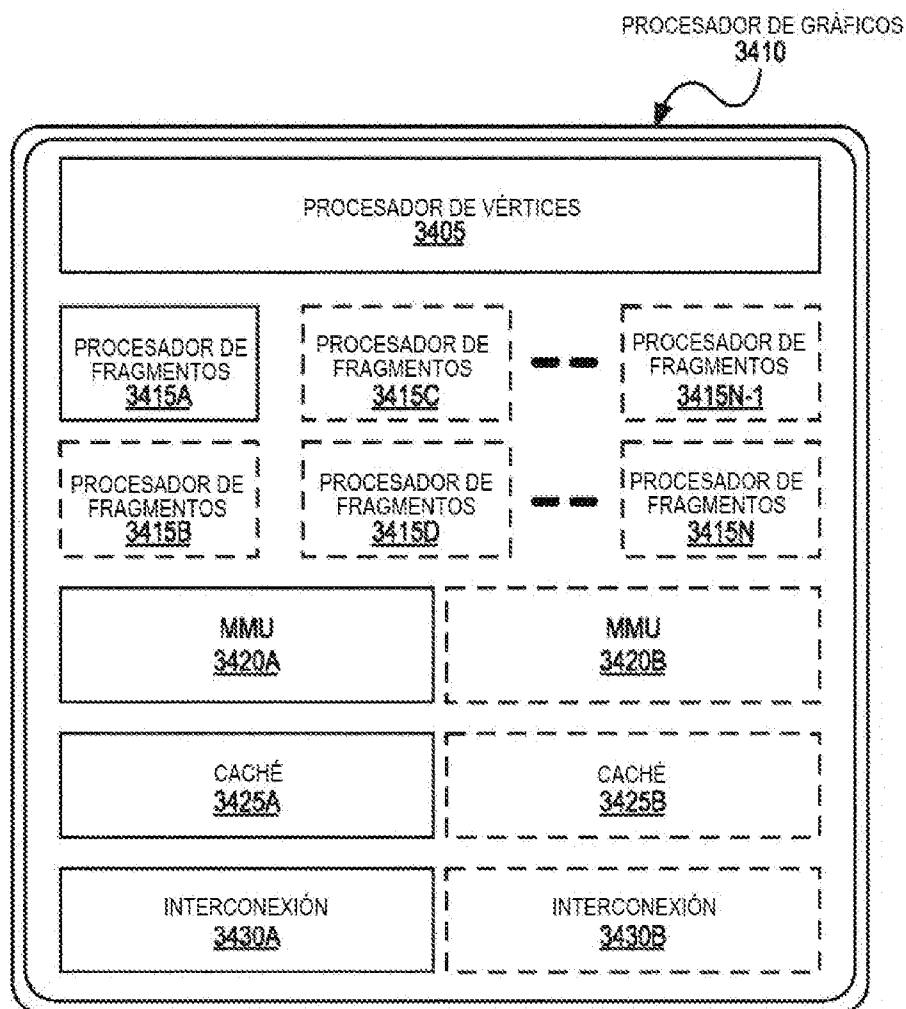


FIG. 34

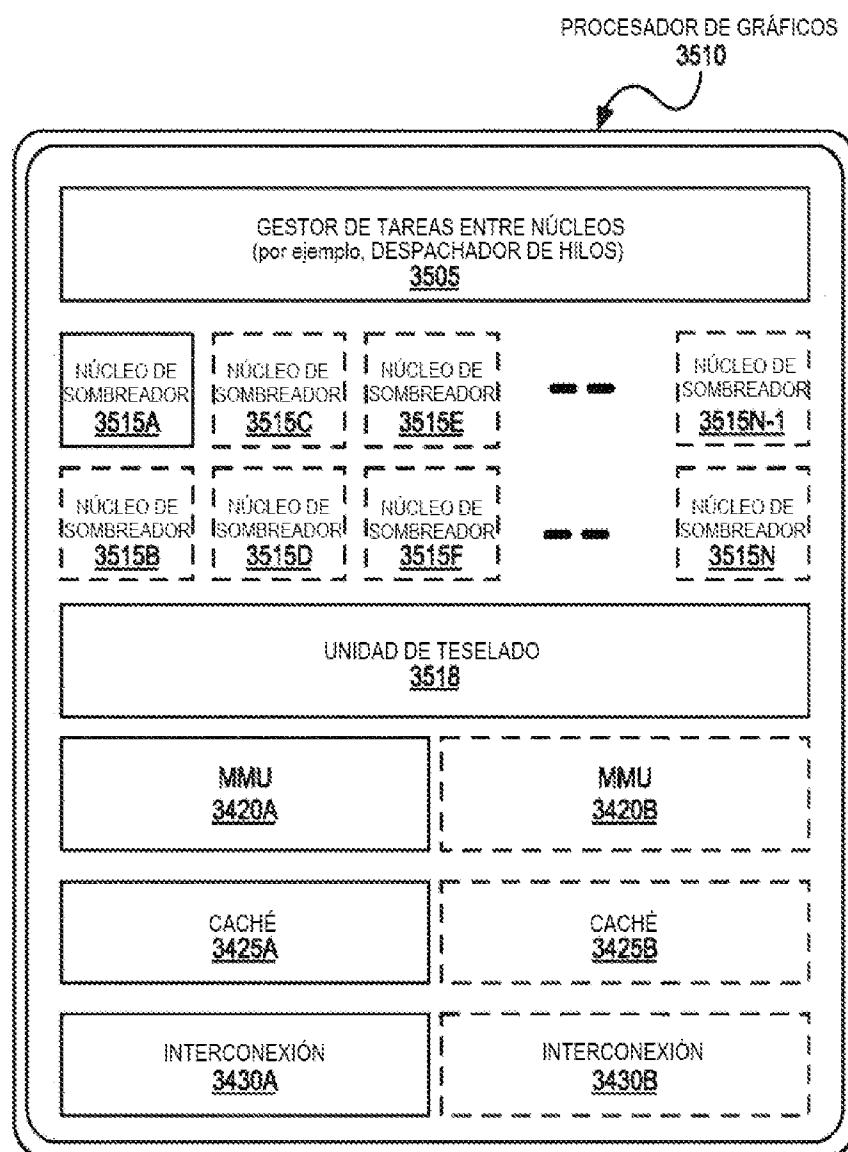


FIG. 35