(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **3 011 182**

(51) Int. Cl.:

G06F 9/52 (2006.01)
G06N 3/063 (2013.01)
G06N 3/084 (2013.01)
G06N 3/044 (2013.01)
G06N 3/045 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **23.03.2018 E 21178579 (5)**

(97) Fecha y número de publicación de la concesión europea: **30.10.2024 EP 3901774**

(54) Título: **Barreras y sincronización para el aprendizaje automático en máquinas autónomas**

(30) Prioridad:

24.04.2017 US 201715495112

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
07.04.2025

(73) Titular/es:

INTEL CORPORATION (100.00%)
2200 Mission College Blvd.
Santa Clara, CA 95054, US

(72) Inventor/es:

APPU, ABHISHEK R.;
KOKER, ALTUG;
RAY, JOYDEEP;
VEMBU, BALAJI;
WEAST, JOHN C.;
MACPHERSON, MIKE B.;
KIM, DUKHWAN;
HURD, LINDA L.;
JAHAGIRDAR, SANJEEV y
RANGANATHAN, VASANTH

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Barreras y sincronización para el aprendizaje automático en máquinas autónomas

CAMPO

Las realizaciones descritas en el presente documento se refieren en general con el procesamiento de datos y más particularmente con la facilitación de barreras y sincronización para el aprendizaje automático en máquinas autónomas.

ANTECEDENTES

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

Para aumentar aún más el rendimiento, los procesadores gráficos generalmente implementan técnicas de procesamiento tales como canalizaciones, que intentan procesar, en paralelo, la mayor cantidad posible de datos gráficos a lo largo de las diferentes partes de la canalización de gráficos. Los procesadores gráficos paralelos con arquitecturas de instrucción única para múltiples hilos (SIMT) están diseñados para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura SIMT, los grupos de hilos paralelos intentan ejecutar instrucciones de programa de forma sincrónica con la mayor frecuencia posible para aumentar la eficiencia de procesamiento. Se puede encontrar una descripción general del software y hardware para arquitecturas SIMT en Shane Cook, CUDA Programming, Capítulo 3, páginas 37-51 (2013) y/o Nicholas Wilt, CUDA Handbook, A Comprehensive Guide to GPU Programming, Secciones 2.6.2 a 3.1.2 (junio de 2013).

El aprendizaje automático ha tenido éxito resolviendo muchos tipos de tareas. Los cálculos que surgen durante el entrenamiento y uso de algoritmos de aprendizaje automático (por ejemplo, redes neuronales) se prestan naturalmente a implementaciones paralelas eficientes. En consecuencia, los procesadores paralelos, tales como las unidades de procesamiento gráfico de propósito general (GPGPU), han desempeñado un papel importante en la implementación práctica de redes neuronales profundas. Los procesadores gráficos paralelos con arquitecturas de instrucción única, hilo múltiple (SIMT) están diseñados para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura SIMT, los grupos de hilos paralelos intentan ejecutar instrucciones de programa de forma sincrónica con la mayor frecuencia posible para aumentar la eficiencia de procesamiento. La eficiencia que brindan las implementaciones de algoritmos de aprendizaje automático en paralelo permite el uso de redes de alta capacidad y permite que esas redes se entrenen con conjuntos de datos más grandes.

Las técnicas convencionales de barrera y sincronización están severamente limitadas por varias razones, tal como la dependencia total del software, la falta de acceso para atravesar múltiples grupos de hilos y la falta de funcionalidades universales, etc., y por lo tanto, dichas técnicas son ineficientes en términos de recursos del sistema, tales como tiempo, memoria, energía y ancho de banda.

El documento US 2016/321 777 A1 se refiere a un método de procesamiento de datos en paralelo basado en múltiples unidades de procesamiento gráfico, GPU, que incluye: crear, en una unidad de procesamiento central, CPU, una pluralidad de hilos de trabajo para controlar una pluralidad de grupos de trabajo respectivamente, incluyendo los grupos de trabajo una o más GPU; vincular cada hilo de trabajo a una GPU correspondiente; cargar una pluralidad de lotes de datos de entrenamiento desde una memoria no volátil a memorias de vídeo de GPU en la pluralidad de grupos de trabajo; y controlar la pluralidad de GPU para realizar el procesamiento de datos en paralelo a través de los hilos de trabajo.

Tyler Sorensen et. al.: "Portable inter-workgroup barrier synchronisation for GPUs", ACM SIGPLAN Notices, vol. 51,núm. 10, 19 de octubre de 2016, págs. 39-58, se refiere a la sincronización de barrera para GPU.

SUMARIO DE LA INVENCIÓN

La invención se define en las reivindicaciones independientes. En las reivindicaciones dependientes se exponen modificaciones ventajosas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

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

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 un procesador paralelo, de acuerdo con una realización.

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

Las Figuras 4A-4F ilustran una arquitectura ejemplar en la que una pluralidad de unidades de procesamiento gráfico están acopladas comunicativamente a una pluralidad de procesadores multinúcleo.

La Figura 5 es un diagrama conceptual de una canalización de procesamiento gráfico, de acuerdo con una realización.

La Figura 6 ilustra un dispositivo informático que aloja un mecanismo de barrera y sincronización de acuerdo con una realización.

La Figura 7 ilustra un dispositivo informático que aloja un mecanismo de barrera y sincronización de acuerdo con una realización.

La Figura 8A ilustra una configuración arquitectónica para emplear y utilizar una barrera de múltiples matrices para el aprendizaje automático de acuerdo con una realización.

La Figura 8B ilustra un marco de trabajo para facilitar la sincronización de grupos de hilos en el aprendizaje automático de acuerdo con una realización.

La Figura 8C ilustra un marco de trabajo para facilitar la compartición de datos entre procesadores gráficos utilizando una biblioteca de superficies en el aprendizaje automático de acuerdo con una realización.

La Figura 9A ilustra un procesador gráfico para facilitar la programación optimizada de grupos de hilos sin barreras o memoria local compartida en el aprendizaje automático de acuerdo con una realización.

La Figura 9B ilustra un marco de trabajo para facilitar la preemisión de grupos de hilos en función de barreras en el aprendizaje automático de acuerdo con una realización.

La Figura 10 ilustra una pila de software de aprendizaje automático, de acuerdo con una realización.

La Figura 11 ilustra una unidad de procesamiento gráfico de propósito general altamente paralela, de acuerdo con una realización.

La Figura 12 ilustra un sistema informático de múltiples GPU, de acuerdo con una realización.

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

La Figura 14 ilustra el entrenamiento y la implementación de una red neuronal profunda.

La Figura 15 ilustra el entrenamiento y la implementación de una red neuronal profunda.

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

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

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

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

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

La Figura 21 es un diagrama de bloques de una realización de un motor de procesamiento gráfico para un procesador gráfico.

La Figura 22 es un diagrama de bloques de otra realización de un procesador gráfico.

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

La Figura 24 ilustra un formato de instrucción de unidad de ejecución de procesador gráfico de acuerdo con una realización.

La Figura 25 es un diagrama de bloques de otra realización de un procesador gráfico que incluye una canalización de gráficos, una canalización de medios, un motor de visualización, una lógica de ejecución de hilos y una canalización de salida de renderizado.

La Figura 26A es un diagrama de bloques que ilustra un formato de comando de procesador gráfico de acuerdo con una realización.

La Figura 26B es un diagrama de bloques que ilustra una secuencia de comandos de procesador gráfico de acuerdo con una realización.

La Figura 27 ilustra una arquitectura de software de gráficos ejemplar para un sistema de procesamiento de datos de acuerdo con una realización.

La Figura 28 es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo IP que puede usarse para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización.

La Figura 29 es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ejemplar que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con una realización.

La Figura 30 es un diagrama de bloques que ilustra un procesador gráfico ejemplar de un circuito integrado de sistema en un chip.

La Figura 31 es un diagrama de bloques que ilustra un procesador gráfico ejemplar adicional de un circuito integrado de sistema en un chip.

DESCRIPCIÓN DETALLADA

Las realizaciones proporcionan una técnica novedosa para ofrecer una barrera de múltiples matrices para el aprendizaje automático y con sincronización por hilo dentro de grupos de hilos. Las realizaciones también facilitan la sincronización entre grupos de hilos mediante barreras, donde para aquellos grupos de hilos que no tienen memoria local compartida (SLM) o uso de barrera, se planifican entre multiprocesadores de transmisión (SM) y/o procesadores gráficos. La realización también permite prescindir de hilos basados de barreras.

Se debe tener en cuenta que términos o acrónimos como "red neuronal convolucional", "RNC", "red neuronal", "RN", "red neuronal profunda", "RNP", "red neuronal recurrente", "RNR" y/o similares pueden usarse como referencia indistinta en todo este documento. Además, expresiones como "máquina autónoma" o simplemente "máquina", "vehículo autónomo" o simplemente "vehículo", "agente autónomo" o simplemente "agente", "dispositivo autónomo" o "dispositivo informático", "robot", y/o similares, se pueden hacer referencia de manera intercambiable a lo largo de todo este documento.

En algunas realizaciones, una unidad de procesamiento gráfico (GPU) está acoplada comunicativamente a núcleos anfitrión/procesador para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede estar acoplada comunicativamente al procesador/núcleos anfitrión a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otras realizaciones, la GPU puede estar integrada en el mismo paquete o chip que los núcleos y acoplada comunicativamente a los núcleos a través de un bus/interconexión de procesador interno (es decir, interno al paquete o chip). Independientemente de la forma en 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. Luego, la GPU utiliza circuitos/lógica dedicados para procesar de manera eficiente estos comandos/instrucciones.

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

Visión General del Sistema I

La Figura 1 es un diagrama de bloques que ilustra un sistema informático 100 configurado para implementar uno o más aspectos 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 comunican a través de una ruta de interconexión que puede incluir un concentrador de memoria 105. El concentrador de memoria 105 puede ser un componente separado dentro de un componente de conjunto de chips o puede estar integrado dentro de uno o más procesadores 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 a través de un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede permitir que el sistema informático 100 reciba entrada de uno o más dispositivos de entrada 108. Adicionalmente, el concentrador de E/S 107 puede habilitar un controlador de visualización, que puede estar incluido en uno o más procesadores 102, para proporcionar salidas a uno o más dispositivos de visualización 110A. En una realización, el uno o más dispositivos de visualización 110A acoplados con el concentrador de E/S 107 pueden incluir un dispositivo de visualización local, interno o integrado.

En una realización, el subsistema de procesamiento 101 incluye uno o más procesadores paralelos 112 acoplados al concentrador de memoria 105 a través de un bus u otro enlace de comunicación 113. El enlace de comunicación 113 puede ser uno de cualquier número de tecnologías o protocolos de enlace de comunicación basados en estándares, tales como, entre otros, PCI Express, o puede ser una interfaz de comunicaciones o una estructura de comunicaciones específica del proveedor. En una realización, el uno o más procesadores paralelos 112 forman un sistema de procesamiento paralelo o vectorial enfocado computacionalmente, que puede incluir una gran cantidad de núcleos de procesamiento y/o agrupaciones de procesamiento, tal como un procesador de muchos núcleos integrados (MIC). En una realización, el uno o más procesadores paralelos 112 forman un subsistema de procesamiento gráfico que puede proporcionar píxeles a uno del uno o más dispositivos de visualización 110A acoplados por medio del concentrador de E/S 107. El o los procesadores paralelos 112 también pueden incluir un controlador de visualización y una interfaz de visualización (no mostrada) para permitir 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. Se puede utilizar un conmutador de E/S 116 para proporcionar un mecanismo de interfaz para permitir conexiones entre el concentrador de E/S 107 y otros componentes, tales como un adaptador de red 118 y/o un adaptador de red inalámbrica 119 que pueden estar integrados en la plataforma, y varios otros dispositivos que pueden agregarse a través de uno o más dispositivos complementarios 120. El adaptador de red 118 puede ser un adaptador Ethernet u otro adaptador de red cableado. El adaptador de red inalámbrica 119 puede incluir uno o más de un dispositivo de red Wi-Fi, Bluetooth, comunicación de campo cercano (NFC) u otro dispositivo de red que incluya una o más radios inalámbricas.

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

En una realización, el uno o más procesadores paralelos 112 incorporan circuitos optimizados para el procesamiento gráfico y vídeo, incluyendo, por ejemplo, circuitos de salida de vídeo, y constituyen una unidad de procesamiento gráfico (GPU). En otra realización, el uno o más procesadores paralelos 112 incorporan circuitos optimizados para el procesamiento de propósito general, al tiempo que preservan la arquitectura computacional subyacente, descrita con mayor detalle en el presente documento. En otra realización más, los componentes del sistema informático 100 pueden

estar integrados con uno o más elementos del sistema en un único circuito integrado. Por ejemplo, el uno o más procesadores paralelos, el concentrador de memoria 112 105, procesador(es) 102 y el concentrador de E/S 107 se pueden integrar en un circuito integrado de sistema en un chip (SoC). Alternativamente, los componentes del sistema informático 100 pueden integrarse en un solo paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una parte de los componentes del sistema informático 100 se pueden integrar en un módulo multichip (MCM), que se puede interconectar con otros módulos multichip en un sistema informático modular.

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

Algunos de los componentes particulares que se muestran en el presente documento son opcionales y pueden no estar incluidos en todas las implementaciones del sistema informático 100. Por ejemplo, se puede soportar cualquier cantidad de tarjetas complementarias o periféricos, o se pueden eliminar algunos componentes. Además, algunas arquitecturas pueden utilizar una terminología diferente para componentes similares a los ilustrados en la Figura 1. Por ejemplo, el concentrador de memoria 105 puede denominarse Puente Norte (Northbridge) en algunas arquitecturas, mientras que el concentrador de E/S 107 puede denominarse Puente Sur (Southbridge).

La Figura 2A ilustra un procesador paralelo 200, de acuerdo con una realización. Los diversos componentes del procesador paralelo 200 pueden implementarse utilizando uno o más dispositivos de circuitos integrados, tales como procesadores programables, circuitos integrados de aplicación específica (ASIC) o matrices de puertas programables en campo (FPGA). El procesador paralelo 200 ilustrado es una variante de 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 permite 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 conmutador, tal como el 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 anfitrión 206 y una barra transversal de memoria 216, donde la interfaz 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 anfitrión 206 recibe una memoria intermedia de comandos a través de la unidad de E/S 204, la interfaz anfitrión 206 puede dirigir operaciones de trabajo para realizar esos comandos a un frontal (front end) 208. En una realización, el 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 esté configurada correctamente y en un estado válido antes de que las tareas se distribuyan a las agrupaciones de procesamiento de la matriz de agrupaciones de procesamiento 212.

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 a 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 utilizando varios algoritmos de planificación y/o distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surja para cada tipo de programa o cálculo. La planificación puede ser manejada dinámicamente por el planificador 210, o puede ser asistida en parte por la lógica del compilador durante la compilación de la lógica del programa configurada para su ejecución por la matriz de agrupaciones de procesamiento 212.

En una realización, se pueden asignar diferentes agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 para procesar diferentes tipos de programas o para realizar diferentes tipos de cálculos.

La matriz de agrupaciones de procesamiento 212 se puede configurar para realizar diversos tipos de operaciones de procesamiento paralelo. En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar

operaciones de cálculo paralelo de propósito general. Por ejemplo, la matriz de agrupaciones de procesamiento 212 puede incluir lógica para ejecutar tareas de procesamiento que incluyen el filtrado de datos de vídeo y/o audio, la realización de operaciones de modelado, incluidas operaciones físicas, y la realización de transformaciones de datos.

En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de procesamiento gráfico paralelo. En realizaciones en las que el procesador paralelo 200 está configurado para realizar operaciones de procesamiento gráfico, la matriz de agrupaciones de procesamiento 212 puede incluir lógica adicional para soportar la ejecución de dichas operaciones de procesamiento gráfico, incluyendo, pero sin limitarse a, lógica de muestreo de textura para realizar operaciones de textura, así como lógica de teselación y otra lógica de procesamiento de vértices. Además, la matriz de agrupaciones de procesamiento 212 se puede configurar para ejecutar programas de sombreador relacionados con el procesamiento gráfico, tales como, entre otros, 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 a través de la unidad de E/S 204 para su procesamiento. Durante el procesamiento, los datos transferidos se pueden almacenar en la memoria en chip (por ejemplo, la memoria de procesador paralelo 222) durante el procesamiento y luego volver a escribirse en la memoria de sistema.

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

Durante el funcionamiento, la matriz de agrupaciones de procesamiento 212 puede recibir tareas de procesamiento que se ejecutarán a través del planificador 210, que recibe comandos que definen tareas de procesamiento desde el frontal 208. Para las operaciones de procesamiento gráfico, las tareas de procesamiento pueden incluir índices de datos a procesar, por ejemplo, datos de superficie (parche), datos de primitivas, datos de vértice y/o datos de píxeles, así como parámetros de estado y comandos que definen cómo se procesarán los datos (por ejemplo, qué programa se ejecutará). El planificador 210 puede configurarse para obtener los índices correspondientes a las tareas o puede recibir los índices desde el frontal 208. El frontal 208 se puede configurar para garantizar que la matriz de agrupaciones de procesamiento 212 esté configurada en un estado válido antes de que se inicie la carga de trabajo especificada por las memorias intermedias de comandos entrantes (por ejemplo, memorias intermedias de lote, memorias intermedias de inserción, etc.).

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

En diversas realizaciones, las unidades de memoria 224A-224N pueden incluir varios tipos de dispositivos de memoria, incluyendo memoria dinámica de acceso aleatorio (DRAM) o memoria de acceso aleatorio de gráficos, tal como memoria de acceso aleatorio de gráficos síncrónica (SGRAM), incluyendo memoria de velocidad de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 224A-224N también pueden incluir memoria apilada 3D, que incluye, entre otras, 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 entre uno de varios diseños convencionales. Los objetivos de renderizado, tales como memorias intermedias de tramas o mapas de textura, se pueden almacenar en las unidades de memoria 224A-224N, lo que permite que las unidades de partición 220A-220N escriban porciones de cada objetivo de renderizado en paralelo para utilizar de manera eficiente el ancho de banda disponible de la memoria de procesador paralelo 222. En algunas realizaciones, se puede excluir una instancia local de la memoria de procesador paralelo 222 en favor de un diseño de memoria unificada que utiliza la memoria de sistema junto con la memoria caché local.

En una realización, cualquiera de las agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 224A-224N dentro de la memoria de procesador paralelo 222. La barra transversal de memoria 216 se puede configurar para transferir la salida de cada grupo 214A-214N a cualquier unidad de partición 220A-220N o a otra agrupación 214A-214N, que puede realizar operaciones de procesamiento adicionales en la salida. Cada agrupación 214A-214N puede comunicarse con la interfaz de memoria 218 a través de la barra transversal de memoria 216 para leer o escribir en varios dispositivos de memoria externa. 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 permite 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 para 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 partición 220A-220N.

Si bien se ilustra una única instancia de la unidad de procesamiento paralelo 202 dentro del procesador paralelo 200, se puede incluir cualquier número de instancias de la unidad de procesamiento paralelo 202. Por ejemplo, se pueden proporcionar múltiples instancias de la unidad de procesamiento paralelo 202 en una sola tarjeta complementaria, o se pueden interconectar múltiples tarjetas complementarias. Las diferentes instancias de la unidad de procesamiento paralelo 202 se pueden configurar para interoperar incluso si las diferentes instancias tienen diferentes cantidades de núcleos de procesamiento, diferentes cantidades de memoria de procesador paralelo local y/u otras diferencias de configuración. Por ejemplo, y en una realización, algunas instancias de la unidad de procesamiento paralelo 202 pueden incluir unidades de punto flotante de mayor precisión en relación con otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 202 o del procesador paralelo 200 se pueden implementar en una variedad de configuraciones y factores de forma, incluyendo, entre otros, computadoras personales de escritorio, portátiles o de mano, servidores, estaciones de trabajo, consolas de juegos y/o sistemas integrados.

La Figura 2B es un diagrama de bloques de una unidad de partición 220, de acuerdo con una realización. En una realización, la unidad de partición 220 es una instancia de una de las unidades de partición 220A-220N de la Figura 2A. Como se ilustra, la unidad de partición 220 incluye una caché L2 221, una interfaz de memoria intermedia de tramas 225 y una ROP 226 (unidad de operaciones de rasterizado). La caché L2 221 es una caché de lectura/escritura que está configurada para realizar operaciones de carga y almacenamiento recibidas desde la barra transversal de memoria 216 y la ROP 226. Los errores de lectura y las solicitudes de escritura urgentes se envían desde la caché L2 221 a la interfaz de memoria intermedia de tramas 225 para su procesamiento. También se pueden enviar actualizaciones sucias a la memoria intermedia de tramas a través de la interfaz de memoria intermedia de tramas 225 para un procesamiento oportunista. En una realización, la interfaz de memoria intermedia de tramas 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 2A (por ejemplo, dentro de la memoria de procesador paralelo 222).

En aplicaciones de gráficos, la ROP 226 es una unidad de procesamiento que realiza operaciones de tramas, tales como plantilla, prueba z, combinación y similares. Luego, la ROP 226 emite datos gráficos procesados que se almacenan en la memoria gráfica. En algunas realizaciones, la ROP 226 incluye lógica de compresión para comprimir datos z o de color que se escriben en la memoria y descomprimir datos z o de color que se leen desde la memoria. En algunas realizaciones, la ROP 226 está incluida dentro de cada agrupación de procesamiento (por ejemplo, el grupo 214A-214N de la Figura 2A) en lugar de dentro de la unidad de partición 220. En dicha realización, las solicitudes de lectura y escritura de datos de píxeles se transmiten a través de la barra transversal de memoria 216 en lugar de datos de fragmentos de píxeles.

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

La Figura 2C es un diagrama de bloques de una agrupación de procesamiento 214 dentro de una unidad de procesamiento paralelo, de acuerdo con una realización. En una realización, la agrupación de procesamiento es una instancia de uno de las agrupaciones de procesamiento 214A-214N de la Figura 2A. La agrupación de procesamiento 214 se puede configurar 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 utilizan técnicas de emisión de instrucciones de instrucción única, datos múltiples (SIMD) para soportar la ejecución paralela de una gran cantidad de hilos sin proporcionar múltiples unidades de instrucción independientes. En otras realizaciones, se utilizan técnicas de instrucción única, hilo múltiple (SIMT) para soportar la ejecución paralela de una gran cantidad de hilos generalmente sincronizados, utilizando una unidad de instrucción común configurada para emitir instrucciones a un conjunto de motores de procesamiento dentro de cada uno de las agrupaciones de procesamiento. A diferencia de un régimen de ejecución SIMD, donde todos los motores de procesamiento generalmente ejecutan instrucciones idénticas, la ejecución SIMT permite que diferentes hilos sigan más fácilmente caminos de ejecución divergentes a través de un programa de hilos determinado. Los expertos en la materia comprenderán que un régimen de procesamiento SIMD representa un subconjunto funcional de un régimen de procesamiento SIMT.

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

Cada multiprocesador gráfico 234 dentro de la agrupación de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades de lógica aritmética, unidades de carga y almacenamiento, etc.). La lógica de ejecución funcional se puede configurar de manera segmentada, de modo que se puedan emitir nuevas instrucciones antes de que se completen las instrucciones anteriores. Se puede proporcionar lógica de ejecución funcional. La lógica funcional soporta una variedad de operaciones, incluidas operaciones de comparación aritmética de números enteros y de punto flotante, operaciones booleanas de 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 en el conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa en diferentes datos de entrada. Cada hilo dentro de un grupo de hilos se puede asignar a un motor de procesamiento diferente dentro de un multiprocesador gráfico 234. Un grupo de hilos puede incluir menos hilos que la cantidad de motores de procesamiento dentro del multiprocesador gráfico 234. Cuando un grupo de hilos incluye menos hilos que la cantidad de motores de procesamiento, uno o más de los motores de procesamiento pueden estar inactivos durante los ciclos en los que se procesa ese grupo de hilos. Un grupo de hilos también puede incluir más hilos que la cantidad de motores de procesamiento dentro del multiprocesador gráfico 234. Cuando el grupo de hilos incluye más hilos que la cantidad de motores de procesamiento dentro del multiprocesador gráfico 234, el procesamiento se puede realizar durante ciclos de reloj consecutivos. En una realización, se pueden ejecutar múltiples grupos de hilos simultáneamente en un multiprocesador gráfico 234.

En una realización, el multiprocesador gráfico 234 incluye una memoria caché interna para realizar operaciones de carga y almacenamiento. En una realización, el multiprocesador gráfico 234 puede prescindir de una caché interna y utilizar una memoria caché (por ejemplo, caché L1 308) dentro de la agrupación de procesamiento 214. Cada multiprocesador gráfico 234 también tiene acceso a cachés L2 dentro de las unidades de partición (por ejemplo, unidades de partición 220A-220N de la Figura 2A) que se comparten entre todos las agrupaciones de procesamiento 214 y pueden usarse para transferir datos entre hilos. El multiprocesador gráfico 234 también puede acceder a la memoria global fuera de chip, que puede incluir una o más memorias de procesador paralelo local y/o memoria de sistema. Cualquier memoria externa a la unidad de procesamiento paralelo 202 puede utilizarse como memoria global. Las realizaciones en las que la agrupación de procesamiento 214 incluye múltiples instancias del multiprocesador gráfico 234 pueden compartir instrucciones y datos comunes, que pueden almacenarse en la memoria 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 2A. La MMU 245 incluye un conjunto de entradas de tabla de páginas (PTE) que se utilizan para asignar una dirección virtual a una dirección física de un mosaico (hablaremos más sobre mosaico) y, opcionalmente, un índice de línea de caché. La MMU 245 puede incluir memorias intermedias de traducción de direcciones (TLB) o cachés que pueden residir dentro del multiprocesador gráfico 234 o la caché L1 o la agrupación de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso a datos de superficie para permitir un intercalado eficiente de solicitudes entre las unidades de partición. El índice de línea de caché se puede utilizar para determinar si una solicitud de una línea de caché es un éxito o un fracaso.

En aplicaciones de gráficos y computación, una agrupación de procesamiento 214 puede configurarse de tal manera que cada multiprocesador gráfico 234 esté acoplado a una unidad de textura 236 para realizar operaciones de mapeo de textura, por ejemplo, determinar posiciones de muestras de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen desde una caché L1 de textura interna (no mostrada) o en algunas realizaciones desde la caché L1 dentro del multiprocesador gráfico 234 y se obtienen de una caché L2, una memoria de procesador paralelo local o una memoria de sistema, según sea necesario. Cada multiprocesador gráfico 234 envía tareas procesadas a la barra transversal de datos 240 para proporcionar la tarea procesada a otra agrupación de procesamiento 214 para su posterior procesamiento o para almacenar la tarea procesada en una memoria caché L2, una memoria de procesador paralelo local o una memoria de sistema a través de la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones de pre-rasterizado) está configurada para recibir datos del multiprocesador gráfico 234, dirigir datos a unidades ROP, que pueden estar ubicadas con unidades de partición como se describe en el presente documento (por ejemplo, unidades de

partición 220A-220N de la Figura 2A). La unidad preROP 242 puede realizar optimizaciones para la combinación de colores, organizar datos de color de píxeles y realizar traducciones de direcciones.

Se apreciará que la arquitectura central descrita en el presente documento es ilustrativa y que son posibles variaciones y modificaciones. Cualquier número de unidades de procesamiento, por ejemplo, multiprocesador gráfico 234, unidades de textura 236, preROP 242, etc., se puede incluir 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 utilizando unidades de procesamiento separadas y distintas, cachés L1, etc.

La Figura 2D muestra un multiprocesador gráfico 234, de acuerdo con una realización. En tal realización, el multiprocesador gráfico 234 se acopla con el gestor de canalización 232 de la agrupación de procesamiento 214. El multiprocesador gráfico 234 tiene una canalización de ejecución que incluye, entre otros, 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 gráfico de propósito general (GPGPU) 262 y una o más unidades de carga/almacenamiento 266. Los núcleos GPGPU 262 y las unidades de carga/almacenamiento 266 están acopiados con la memoria caché 272 y la memoria compartida 270 a través de una interconexión de memoria y caché 268.

En una realización, la memoria caché de instrucciones 252 recibe un flujo de instrucciones para ejecutar desde el gestor de canalización 232. Las instrucciones se almacenan en caché en la caché de instrucciones 252 y se despachan para su ejecución por la unidad de instrucción 254. La unidad de instrucción 254 puede despachar instrucciones como grupos de hilos (por ejemplo, warps), con cada hilo del grupo de hilos asignado a una unidad de ejecución diferente dentro del núcleo GPGPU 262. Una instrucción puede acceder a cualquier espacio de dirección local, compartido o global especificando una dirección dentro de un espacio de direcciones unificado. La unidad de mapeo de direcciones 256 se puede utilizar para traducir direcciones en el espacio de direcciones unificado en 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 gráfico 324. El archivo de registro 258 proporciona almacenamiento temporal para los operandos conectados a las rutas de datos de las unidades funcionales (por ejemplo, núcleos GPGPU 262, unidades de carga/almacenamiento 266) del multiprocesador gráfico 324. En una realización, el archivo de registro 258 se divide entre cada una de las unidades funcionales de modo que a cada unidad funcional se le asigna una porción dedicada del archivo de registro 258. En una realización, el archivo de registro 258 se divide entre las diferentes urdidumbres que ejecuta el multiprocesador gráfico 324.

Los núcleos GPGPU 262 pueden incluir cada uno unidades de punto flotante (FPU) y/o unidades lógicas aritméticas (ALU) de enteros que se utilizan para ejecutar instrucciones del multiprocesador gráfico 324. Los núcleos 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 GPGPU 262 incluye una FPU de precisión simple y una ALU entera, mientras que una segunda porción de los núcleos GPGPU incluye una FPU de precisión doble. En una realización, las FPU pueden implementar el estándar IEEE 754-2008 para aritmética de punto flotante o habilitar aritmética de punto flotante de precisión variable. El multiprocesador gráfico 324 puede incluir adicionalmente una o más unidades de función fija o de función especial para realizar funciones específicas tales como operaciones de copia de rectángulos o de fusión de píxeles. En una realización, uno o más de los núcleos GPGPU también pueden incluir lógica de función fija o especial.

La interconexión de memoria y caché 268 es una estructura de interconexión que conecta cada una de las unidades funcionales del multiprocesador gráfico 324 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 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 GPGPU 262, por lo que la transferencia de datos entre los núcleos GPGPU 262 y el archivo de registro 258 tiene una latencia muy baja. La memoria compartida 270 se puede utilizar para permitir la comunicación entre hilos que se ejecutan en las unidades funcionales dentro del multiprocesador gráfico 234. La memoria caché 272 se puede utilizar como caché de datos, por ejemplo, para almacenar en caché datos de textura comunicados entre las unidades funcionales y la unidad de textura 236. La memoria compartida 270 también se puede utilizar como caché gestionada por programas. Los hilos que se ejecutan en los núcleos GPGPU 262 pueden almacenar datos de manera programática dentro de la memoria compartida además de los datos almacenados automáticamente en caché dentro de la memoria caché 272.

Las Figuras 3A-3B ilustran multiprocesadores gráficos adicionales, de acuerdo con realizaciones. Los multiprocesadores gráficos 325, 350 ilustrados son variantes del multiprocesador gráfico 234 de la Figura 2C. Los multiprocesadores gráficos 325, 350 ilustrados se pueden configurar como un multiprocesador de transmisión (SM) capaz de ejecutar simultáneamente una gran cantidad de hilos de ejecución.

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

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

Las personas expertas en la materia entenderán que la arquitectura descrita en las Figuras 1, 2A-2D y 3A-3B es descriptiva y no limitativa en cuanto al alcance de las presentes realizaciones. Por lo tanto, las técnicas descritas en el presente documento se pueden implementar en cualquier unidad de procesamiento configurada adecuadamente, incluyendo, sin limitación, uno o más procesadores de aplicaciones móviles, una o más unidades de procesamiento central (CPU) de escritorio o servidor, incluyendo CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tal como la unidad de procesamiento paralelo 202 de la Figura 2A, así como uno o más procesadores 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 comunicativamente a núcleos anfitrión/procesador para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede acoplarse de manera comunicativa al procesador/núcleos anfitrión a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otras realizaciones, la GPU puede estar integrada en el mismo paquete o chip que los núcleos y acoplada comunicativamente a los núcleos a través de un bus/interconexión de procesador interno (es decir, interno al paquete o chip). Independientemente de la forma en 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. Luego, la GPU utiliza circuitos/lógica dedicados para procesar de manera eficiente estos comandos/instrucciones.

Técnicas para la Interconexión de la GPU al Procesador Anfitrión

La Figura 4A ilustra una arquitectura ejemplar en la que una pluralidad de GPU 410-413 están acopladas comunicativamente a una pluralidad de procesadores multinúcleo 405-406 a través de enlaces de alta velocidad 440-443 (por ejemplo, buses, interconexiones punto a punto, etc.). En una realización, los enlaces de alta velocidad 440-443 soportan un rendimiento de comunicación de 4 GB/s, 30 GB/s, 80 GB/s o superior, dependiendo de la implementación. Se pueden utilizar diversos protocolos de interconexión, incluidos, entre otros, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no están limitados a un protocolo de comunicación o rendimiento en 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 utilizando los mismos o diferentes protocolos/enlaces que los utilizados para los enlaces de alta velocidad 440-443. De manera similar, dos o más de los procesadores multinúcleo 405-406 pueden estar conectados a través de un enlace de alta velocidad 433 que pueden ser buses multiprocesador simétrico (SMP) que funcionan a 20 GB/s, 30 GB/s, 120 GB/s o más. Como alternativa, toda la comunicación entre los diversos componentes del sistema que se muestran en la Figura 4A puede lograrse utilizando los mismos protocolos/enlaces (por ejemplo, a través de una estructura de interconexión común). Sin embargo, como se mencionó, los principios subyacentes de la invención no están limitados a un tipo particular de tecnología de interconexión.

En una realización, cada procesador multinúcleo 405-406 está acoplado comunicativamente a una memoria de procesador 401-402, a través de interconexiones de memoria 430-431, respectivamente, y cada GPU 410-413 está acoplada comunicativamente a la memoria de GPU 420-423 a través de interconexiones de memoria de GPU 450-453, respectivamente. Las interconexiones de memoria 430-431 y 450-453 pueden utilizar las mismas o diferentes tecnologías

de acceso a memoria. A modo de ejemplo, y sin limitación, las memorias de procesador 401-402 y las memorias de GPU 420-423 pueden ser memorias volátiles tales como memorias dinámicas de acceso aleatorio (DRAM) (incluyendo DRAM apiladas), memoria DDR SDRAM 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, utilizando una jerarquía de memoria de dos niveles (2LM)).

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

La Figura 4B ilustra detalles adicionales para una interconexión entre un procesador multinúcleo 407 y un módulo de aceleración gráfica 446 de acuerdo con una realización. El módulo de aceleración gráfica 446 puede incluir uno o más chips GPU integrados en una tarjeta de línea que está acoplada al procesador 407 a través del enlace de alta velocidad 440. Alternativamente, el módulo de aceleración gráfica 446 puede estar integrado en el mismo paquete o chip que el procesador 407.

El procesador 407 ilustrado incluye una pluralidad de núcleos 460A-460D, cada uno con una memoria intermedia de búsqueda de traducción 461A-461D y una o más cachés 462A-462D. Los núcleos pueden incluir varios otros componentes para ejecutar instrucciones y procesar datos que no están ilustrados para evitar oscurecer los principios subyacentes de la invención (por ejemplo, unidades de búsqueda de instrucciones, unidades de predicción de ramificaciones, decodificadores, unidades de ejecución, memorias intermedias de reordenamiento, etc.). Las cachés 462A-462D pueden comprender cachés de nivel 1 (L1) y de nivel 2 (L2). Además, pueden incluirse una o más cachés compartidas 426 en la jerarquía de almacenamiento en caché y compartirse por conjuntos de los núcleos 460A-460D. Por ejemplo, una realización del procesador 407 incluye 24 núcleos, cada uno con su propia caché L1, doce cachés L2 compartidas y doce cachés L3 compartidas. En esta realización, una de las cachés L2 y L3 son compartidas por dos núcleos adyacentes. El procesador 407 y el módulo de integración de acelerador de gráficos 446 se conectan con la memoria de sistema 441, que puede incluir las memorias de procesador 401-402.

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 la comunicación entre núcleos a través de un bus de coherencia 464. Por ejemplo, cada caché puede tener una lógica/circuito de coherencia de caché asociado 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 vigilancia de caché a través del bus de coherencia 464 para vigilar los accesos a caché. Las técnicas de coherencia/vigilancia de caché son bien entendidas por aquellos expertos en la materia y no se describirán en detalle aquí para evitar oscurecer los principios subyacentes de la invención.

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

En una implementación, un circuito de integración de acelerador 436 proporciona gestión de caché, acceso a memoria, gestión de contexto y servicios de gestión de interrupciones en nombre de una pluralidad de motores de procesamiento gráfico 431, 432, N del módulo de aceleración gráfica 446. Los motores de procesamiento gráfico 431, 432, N pueden comprender cada uno una unidad de procesamiento gráfico (GPU) independiente. Como alternativa, los motores de procesamiento gráfico 431, 432, N pueden comprender diferentes tipos de motores de procesamiento gráfico dentro de una GPU, tales como unidades de ejecución de gráficos, motores de procesamiento de medios (por ejemplo, codificadores/decodificadores de vídeo), muestreadores y motores blit. En otras palabras, el módulo de aceleración gráfica puede ser una GPU con una pluralidad de motores de procesamiento gráfico 431-432, N o los motores de procesamiento gráfico 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 varias funciones de gestión de memoria tales como traducciones de memoria virtual a física (también denominadas traducciones de memoria efectiva a real) y protocolos de acceso a memoria para acceder a la memoria de sistema 441. La MMU 439 también puede incluir una memoria intermedia de búsqueda de traducción (TLB) (no mostrada) para almacenar en caché las traducciones de direcciones virtuales/efectivas a físicas/reales. En una implementación, una memoria caché 438 almacena comandos y datos para un acceso eficiente por parte de los motores de procesamiento

gráfico 431-432, N. En una realización, los datos almacenados en la memoria caché 438 y las memorias gráficas 433-434, N se mantienen coherentes con las memorias caché centrales 462A-462D, 456 y la memoria de sistema 411. Como se mencionó, esto se puede lograr a través del circuito proxy 425 que participa 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 a líneas de caché en las cachés de procesador 462A-462D, 456 y recibiendo actualizaciones de la caché 438).

Un conjunto de registros 445 almacena datos de contexto para hilos ejecutados por los motores de procesamiento gráfico 431-432, N y un circuito de gestión de contexto 448 gestiona los contextos de hilos. Por ejemplo, el circuito de gestión de contexto 448 puede realizar operaciones de guardar y restaurar para guardar y restaurar contextos de los diversos hilos durante cambios de contexto (por ejemplo, donde se guarda un primer hilo y se almacena un segundo hilo de modo que el segundo hilo pueda ser ejecutado por un motor de procesamiento gráfico). Por ejemplo, en un cambio de contexto, el circuito de gestión de contexto 448 puede almacenar valores de registro actuales en una región designada en la memoria (por ejemplo, identificada por un puntero de contexto). Luego puede restaurar los valores del registro al regresar al contexto. En una realización, un circuito de gestión de interrupciones 447 recibe y procesa las interrupciones recibidas de los dispositivos del sistema.

En una implementación, las direcciones virtuales/efectivas de un motor de procesamiento gráfico 431 se traducen a direcciones reales/físicas en la memoria de sistema 411 por la MMU 439. Una realización del circuito de integración de acelerador 436 soporta múltiples (por ejemplo, 4, 8, 16) módulos aceleradores de gráficos 446 y/u otros dispositivos aceleradores. El módulo acelerador de gráficos 446 puede estar dedicado a una sola 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 gráfico 431-432, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos se pueden subdividir en "segmentos" que se asignan a diferentes máquinas virtuales o aplicaciones en función de los requisitos de procesamiento y las prioridades asociadas con las máquinas virtuales o aplicaciones.

De este modo, el circuito de integración de acelerador actúa como un puente hacia el sistema para el módulo de aceleración gráfica 446 y proporciona servicios de traducción de direcciones y de caché de memoria de sistema. Además, el circuito de integración de acelerador 436 puede proporcionar facilidades de virtualización para que el procesador anfitrión gestione la virtualización de los motores de procesamiento gráfico, las interrupciones y la gestión de memoria.

Debido a que los recursos de hardware de los motores de procesamiento gráfico 431-432, N se asignan explícitamente al espacio de dirección real visto por el procesador anfitrión 407, cualquier procesador anfitrión puede direccionar estos recursos directamente utilizando 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 gráfico 431-432, N de modo que aparezcan en el sistema como unidades independientes.

Como se mencionó, en la realización ilustrada, una o más memorias gráficas 433-434, M están acopladas a cada uno de los motores de procesamiento gráfico 431-432, N, respectivamente. Las memorias gráficas 433-434, M almacenan instrucciones y datos que son procesados por cada uno de los motores de procesamiento gráfico 431-432, N. Las memorias gráficas 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 utilizan técnicas de polarización para garantizar que los datos almacenados en las memorias gráficas 433-434, M sean datos que serán utilizados con mayor frecuencia por los motores de procesamiento gráfico 431-432, N y preferiblemente no utilizados por los núcleos 460A-460D (al menos no con frecuencia). De manera similar, el mecanismo de polarización intenta mantener los datos que necesitan los núcleos (y preferiblemente no los motores de procesamiento gráfico 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 gráfico 431-432, N se comunican directamente a través del enlace de alta velocidad 440 con el circuito de integración de acelerador 436 mediante la interfaz 437 y la interfaz 435 (que, nuevamente, pueden utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración de acelerador 436 puede realizar las mismas operaciones que las descritas con respecto a la Figura 4B, pero potencialmente con un mayor rendimiento dada su proximidad al bus de coherencia 462 y 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 gráfica) y modelos de programación compartidos (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 gráfica 446.

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

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

Para el modelo de programación compartida, el módulo de aceleración gráfica 446 o un motor de procesamiento gráfico 431-432, N individual selecciona un elemento de proceso utilizando un manejador de proceso. En una realización, los elementos de proceso se almacenan en la memoria de sistema 411 y son direccionables utilizando las técnicas de traducción de dirección efectiva a dirección real descritas en el presente documento. El identificador de proceso puede ser un valor específico de la implementación proporcionado al proceso anfitrión cuando se registra su contexto con el motor de procesamiento gráfico 431-432, N (es decir, se llama al software del sistema para agregar 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 ejemplar. Tal como se utiliza en el presente documento, un "segmento" comprende una porción específica de los recursos de procesamiento del circuito de integración de acelerador 436. El espacio de dirección efectiva 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 invocaciones de GPU 481 desde 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 solo trabajo solicitado por una aplicación o puede contener un puntero a una cola de trabajos. En el último caso, el WD 484 es un puntero a la cola de solicitudes de trabajo en el espacio de direcciones de aplicación 482.

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

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

En funcionamiento, una unidad de búsqueda de WD 491 en el segmento de integración de acelerador 490 busca el siguiente WD 484 que incluye una indicación del trabajo que debe realizar uno de los motores de procesamiento gráfico del módulo de aceleración gráfica 446. Los datos del WD 484 pueden almacenarse en registros 445 y ser utilizados 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 circuitos de recorrido de segmento/página para acceder a las tablas de segmento/página 486 dentro del espacio de dirección virtual del sistema operativo 485. El circuito de gestión de interrupciones 447 puede procesar eventos de interrupción 492 recibidos desde el módulo de aceleración gráfica 446. Al realizar operaciones gráficas, una dirección efectiva 493 generada por un motor de procesamiento gráfico 431-432, N se traduce a una dirección real por la MMU 439.

En una realización, el mismo conjunto de registros 445 se duplica para cada motor de procesamiento gráfico 431-432, N y/o módulo de aceleración gráfica 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 de 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 partición lógica
8	Puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

Los registros ejemplares que pueden ser inicializados por el sistema operativo se muestran en la Tabla 2.

Tabla 2 - Registros inicializados por sistema operativo

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

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

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

Los modelos de programación compartidos permiten que todos o un subconjunto de procesos de todas o un subconjunto de particiones en el sistema utilicen un módulo de aceleración gráfica 446. Hay dos modelos de programación en los que el módulo de aceleración gráfica 446 es compartido por múltiples procesos y particiones: compartido por intervalos de tiempo y compartido dirigido por gráficos.

En este modelo, el hipervisor de sistema 496 posee el módulo de aceleración gráfica 446 y pone su función a disposición de todos los sistemas operativos 495. Para que un módulo de aceleración gráfica 446 soporte la virtualización mediante el hipervisor de sistema 496, el módulo de aceleración gráfica 446 puede cumplir 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 gráfica 446 debe proporcionar un mecanismo de guardado y restauración de contexto. 2) el módulo de aceleración gráfica 446 garantiza que la solicitud de trabajo de una aplicación se complete en un período de tiempo específico, incluidos los errores de traducción, o bien el módulo de aceleración gráfica 446 proporciona la capacidad de anticipar el procesamiento del trabajo. 3) al módulo de aceleración gráfica 446 se le debe garantizar equidad entre procesos cuando opera en el modelo de programación compartida dirigida.

En una realización, para el modelo compartido, se requiere que la aplicación 480 realice una llamada de sistema al sistema operativo 495 con un tipo de módulo de aceleración gráfica 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 de módulo de aceleración gráfica 446 describe la función de aceleración específica para la llamada de sistema. El tipo de módulo de aceleración gráfica 446 puede ser un valor específico del sistema. El WD está formateado específicamente para el módulo de aceleración gráfica 446 y puede tener la forma de un comando del módulo de aceleración gráfica 446, un puntero de dirección efectivo a una estructura definida por el usuario, un puntero de dirección efectivo a una cola de comandos o cualquier otra estructura de datos para describir el trabajo que debe realizar el módulo de aceleración gráfica 446. En una realización, el valor AMR es el estado AMR que se utilizará para el proceso actual. El valor que se pasa al sistema operativo es similar al de una aplicación que configura el AMR. Si las implementaciones del circuito de integración de acelerador 436 y del módulo de aceleración gráfica 446 no soportan un registro de anulación de máscara de autoridad de usuario (UAMOR), el sistema operativo puede aplicar el valor UAMOR actual al valor AMR antes de pasar el AMR en la llamada de hipervisor. El hipervisor 496 puede aplicar opcionalmente el valor actual del registro de anulación de máscara de autoridad (AMOR) antes de colocar el AMR en el elemento de proceso 483. En una realización, el CSRP es uno de los registros 445 que contienen la dirección efectiva de un área en el espacio de direcciones de aplicación 482 para que el

módulo de aceleración gráfica 446 guarde y restaure el estado de contexto. Este puntero es opcional si no se requiere guardar un estado entre trabajos o cuando se interrumpe un trabajo. El área de guardado/restauración de contexto puede estar anclada a la memoria de sistema.

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

Tabla 3 - Parámetros de llamada de SO a hipervisor

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de restauración/guardado de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) y un 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 haya registrado y se le haya otorgado la autoridad para utilizar el módulo de aceleración gráfica 446. A continuación, el hipervisor 496 coloca el elemento de proceso 483 en la lista vinculada de elementos de proceso para el tipo de módulo de aceleración gráfica 446 correspondiente. El elemento de proceso puede incluir la información que se muestra en la Tabla 4

Tabla 4 - Información de elemento de proceso

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de restauración/guardado de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) y un 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 partición lógica (LPID)
11	Un puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
12	El registro de descriptor de almacenamiento (SDR)

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

Como se ilustra en la Figura 4F, una realización de la invención emplea una memoria unificada direccionable a través de un espacio de dirección de memoria virtual común utilizado 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 dirección de memoria virtual/efectiva para acceder a las memorias de procesador 401-402 y viceversa, simplificando así la programabilidad. En una realización, una primera porción del espacio de dirección virtual/efectiva se asigna 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. De este modo, todo el espacio de memoria virtual/efectiva (a veces denominado espacio de dirección efectiva) se distribuye entre cada una de las memorias de procesador 401-402 y las memorias de GPU 420-423, lo que permite que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual asignada a esa memoria.

En una realización, los circuitos de gestión de polarización/coherencia 494A-494E dentro de una o más de las MMU 439A-439E aseguran la coherencia de caché entre las memorias caché de los procesadores anfitriones (por ejemplo, 405) y las GPU 410-413 e implementan técnicas de polarización que indican las memorias físicas en las que se deben almacenar ciertos tipos de datos. Si bien en la Figura 4F se ilustran múltiples instancias de circuitos de gestión de polarización/coherencia 494A-494E, los circuitos de polarización/coherencia se pueden implementar dentro de la MMU de uno o más procesadores anfitriones 405 y/o dentro del circuito de integración de acelerador 436.

Una realización permite que la memoria adjunta a la GPU 420-423 se mapee como parte de la memoria de sistema y se acceda a ella utilizando tecnología de memoria virtual compartida (SVM), pero sin sufrir los inconvenientes de rendimiento típicos asociados con la coherencia total de la caché de sistema. La capacidad de acceder a la memoria 420-423 conectada a la GPU como memoria de sistema sin una onerosa sobrecarga de coherencia de caché proporciona un entorno operativo beneficioso para la descarga de la GPU. Esta disposición permite que el software del procesador anfitrión 405 configure operandos y acceda a los resultados de los cálculos, sin la sobrecarga de las copias de datos DMA de E/S tradicionales. Estas copias tradicionales implican llamadas a controladores, interrupciones y accesos de E/S mapeados en memoria (MMIO), que son todos ineficientes en comparación con los accesos a memoria simples. Al mismo tiempo, la capacidad de acceder a la memoria 420-423 conectada a la GPU sin sobrecarga de coherencia de caché puede ser fundamental para el tiempo de ejecución de un cálculo descargado. En casos con tráfico de memoria de escritura en transmisión sustancial, por ejemplo, la sobrecarga de coherencia de caché puede reducir significativamente el ancho de banda de escritura efectivo observado por una GPU 410-413. La eficiencia de la configuración de operandos, la eficiencia del acceso a resultados y la eficiencia del cálculo de GPU juegan un papel en la determinación de la efectividad de la descarga de GPU.

En una implementación, la selección entre la polarización de GPU y la polarización de procesador anfitrión está impulsada por una estructura de datos de seguimiento de polarización. Se puede utilizar una tabla de polarización, por ejemplo, que puede ser una estructura granular de página (es decir, controlada en la granularidad de una página de memoria) que incluye 1 o 2 bits por página de memoria adjunta a la GPU. La tabla de polarización se puede implementar en un rango de memoria robada de una o más memorias adjuntas a la GPU 420-423, con o sin una caché de polarización en la GPU 410-413 (por ejemplo, para almacenar en caché entradas utilizadas frecuente/recientemente de la tabla de polarización). Alternativamente, toda la tabla de polarización se puede mantener dentro de la GPU.

En una implementación, se accede a la entrada de tabla de polarización asociada con cada acceso a la memoria adjunta a la GPU 420-423 antes del acceso real a la memoria de GPU, lo que provoca las siguientes operaciones. En primer lugar, las solicitudes locales de la GPU 410-413 que encuentran su página en la polarización de GPU se reenvían directamente a una memoria GPU 420-423 correspondiente. Las solicitudes locales de la GPU que encuentran su página en la polarización de anfitrión se reenvían al procesador 405 (por ejemplo, a través de un enlace de alta velocidad como se explicó anteriormente). En una realización, las solicitudes del procesador 405 que encuentran la página solicitada en el procesador anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, las solicitudes dirigidas a una página orientada a la GPU pueden reenviarse a la GPU 410-413. La GPU puede luego pasar la página a un procesador anfitrión si no está usando la página en ese momento.

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

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

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

Canalización de procesamiento gráfico

La Figura 5 ilustra una canalización de procesamiento gráfico 500, de acuerdo con una realización. En una realización, un procesador gráfico puede implementar la canalización de procesamiento gráfico 500 ilustrada. El procesador gráfico 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 2A, que, en una realización, es una variante del o de los procesadores paralelos 112 de la Figura 1. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento

gráfico 500 a través de una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 202 de la Figura 2A) como se describe en el presente documento. Por ejemplo, una unidad de sombreador (por ejemplo, el multiprocesador gráfico 234 de la Figura 2D) puede configurarse para realizar las funciones de una o más de una unidad de procesamiento de vértices 504, una unidad de 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 rasterizado 526 también pueden ser realizadas por otros motores de procesamiento dentro de un agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 214 de la Figura 3A) y una unidad de partición correspondiente (por ejemplo, la unidad de partición 220A-220N de la Figura 2C). La canalización de procesamiento gráfico 500 también puede implementarse utilizando unidades de procesamiento dedicadas para una o más funciones. En una realización, una o más porciones de la canalización de procesamiento gráfico 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 gráfico 500 pueden acceder a la memoria en chip (por ejemplo, la memoria de procesador paralelo 222 como en la Figura 2A) a través de una interfaz de memoria 528, que puede ser una instancia de la interfaz de memoria 218 de la Figura 2A.

En una realización, el ensamblador de datos 502 es una unidad de procesamiento que recopila datos de vértices para superficies y primitivas. Luego, el ensamblador de datos 502 envía los datos de vértice, incluidos los atributos de vértice, a la unidad de procesamiento de vértices 504. La unidad de procesamiento de vértices 504 es una unidad de ejecución programable que ejecuta programas de sombreador de vértices, iluminando y transformando datos de vértices según lo especificado por los programas de sombreador de vértices. La unidad de procesamiento de vértices 504 lee datos almacenados en memoria caché, local o de sistema para su uso en el procesamiento de los datos de vértices y puede programarse para transformar los datos de vértices desde una representación de coordenadas basada en objetos a un espacio de coordenadas del espacio mundial o un espacio de coordenadas de dispositivo normalizado.

Una primera instancia de un ensamblador de primitivas 506 recibe atributos de vértice de la unidad de procesamiento de vértices 504. El ensamblador de primitivas 506 lee los atributos de vértice almacenados según sea necesario y construye primitivas gráficas para su procesamiento por la unidad de procesamiento de control de teselación 508. Las primitivas gráficas incluyen triángulos, segmentos de línea, puntos, parches, etc., tal como lo soportan varias interfaces de programación de aplicaciones (API) de procesamiento gráfico.

La unidad de procesamiento de control de teselación 508 trata los vértices de entrada como puntos de control para un parche geométrico. Los puntos de control se transforman desde una representación de entrada del parche (por ejemplo, las bases del parche) a una representación que es adecuada para su uso en la evaluación de la superficie por la unidad de procesamiento de evaluación de teselación 512. La unidad de procesamiento de control de teselación 508 también puede calcular factores de teselación para los bordes de parches geométricos. Un factor de teselación se aplica a un solo 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 los bordes de un parche y para teselar el parche en múltiples primitivas geométricas tales como primitivas de línea, triángulo o cuadrilátero, que se transmiten a una unidad de procesamiento de evaluación de teselación 512. La unidad de procesamiento de evaluación de teselación 512 opera sobre coordenadas parametrizadas del parche subdividido para generar una representación de superficie y atributos de vértice para cada vértice asociado con las primitivas geométricas.

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

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

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

La unidad de procesamiento de fragmentos/píxeles 524 es una unidad de ejecución programable que está configurada para ejecutar programas de sombreador de fragmentos o programas de sombreador de píxeles. La unidad de procesamiento de fragmentos/píxeles 524 transforma los fragmentos o píxeles recibidos desde el rasterizador 522, según lo especificado por los programas de sombreador de fragmentos o píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 puede estar programada para realizar operaciones que incluyen, entre otras, mapeo de textura, sombreado, combinación, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se envían a una unidad de operaciones de rasterizado 526. La unidad de procesamiento de fragmentos/píxeles 524 puede leer datos almacenados en la memoria de procesador paralelo o en la memoria de sistema para su uso al procesar los datos de fragmentos. Los programas de sombreador de fragmentos o píxeles se pueden configurar para sombrear en granularidades de muestra, píxel, mosaico u otras, dependiendo de la frecuencia de muestreo configurada para las unidades de procesamiento.

La unidad de operaciones de rasterizado 526 es una unidad de procesamiento que realiza operaciones de rasterizado que incluyen, pero no se limitan a, plantilla, prueba z, combinación y similares, y genera datos de píxeles como datos gráficos procesados para almacenarse en la memoria de gráficos, por ejemplo, la memoria de procesador paralelo 222 como en la Figura 2A, y/o la memoria de sistema 104 como en la Figura 1, para mostrarse en el uno o más dispositivos de visualización 110 o para su posterior procesamiento por uno del uno o más procesadores 102 o procesadores paralelos 112. En algunas realizaciones, la unidad de operaciones de rasterización 526 está configurada para comprimir datos z o de color que se escriben en memoria y descomprimir datos z o de color que se leen desde la memoria.

La Figura 6 ilustra un dispositivo informático 600 que aloja un mecanismo de barrera y sincronización 610 de acuerdo con una realización. El dispositivo informático 600 representa un dispositivo de comunicación y procesamiento de datos que incluye (pero no se limita a) dispositivos portátiles inteligentes, teléfonos inteligentes, dispositivos de realidad virtual (VR), pantallas montadas en la cabeza (HMD), computadoras móviles, dispositivos de Internet de las cosas (IoT), computadoras portátiles, computadoras de escritorio, computadoras servidor, etc., y es similar o igual al dispositivo informático 100 de la Figura 1; en consecuencia, por brevedad, claridad y facilidad de comprensión, muchos de los detalles indicados anteriormente con referencia a las Figuras 1-5 no se analizan ni se repiten en lo sucesivo.

El dispositivo informático 600 puede incluir además (sin limitaciones) una máquina autónoma o un agente de inteligencia artificial, tal como un agente o máquina mecánicos, un agente o máquina electrónicos, un agente o máquina virtuales, un agente o máquina electromecánicos, etc. Los ejemplos de máquinas autónomas o agentes de inteligencia artificial pueden incluir (sin limitación) robots, vehículos autónomos (por ejemplo, automóviles autónomos, aviones autónomos, barcos autónomos, etc.), equipos autónomos (vehículos de construcción autónomos, equipos médicos autónomos, etc.) y/o similares. A través de todo este documento, "dispositivo informático" puede denominarse de manera intercambiable "máquina autónoma" o "agente artificialmente inteligente" o simplemente "robot".

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

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

Como se ilustra, en una realización, el dispositivo informático 600 puede incluir cualquier número y tipo de componentes de hardware y/o software, tales como (sin limitación) unidad de procesamiento gráfico ("GPU" o simplemente "procesador gráfico") 614, controlador gráfico (también denominado "controlador de GPU", "lógica de controlador gráfico", "lógica de controlador", controlador de modo de usuario (UMD), UMD, marco de controlador de modo de usuario (UMDF), UMDF o simplemente "controlador") 616, unidad central de procesamiento ("CPU" o simplemente "procesador de aplicaciones") 612, memoria 608, dispositivos de red, controladores o similares, así como fuentes de entrada/salida (E/S) 604, tales como

pantallas táctiles, paneles táctiles, alfombrillas táctiles, teclados virtuales o regulares, ratones virtuales o regulares, puertos, conectores, etc. El dispositivo informático 600 puede incluir un sistema operativo (OS) 606 que sirve como una interfaz entre el hardware y/o los recursos físicos del dispositivo informático 600 y un usuario. Se contempla que el procesador gráfico 614 y el procesador de aplicaciones 612 pueden ser uno o más de los procesadores 102 de la Figura 1.

Se debe apreciar que un sistema menor o más equipado que el ejemplo descrito anteriormente puede ser preferible para ciertas implementaciones. Por lo tanto, la configuración del dispositivo informático 600 puede variar de una implementación a otra dependiendo de numerosos factores, tales como limitaciones de precio, requisitos de rendimiento, mejoras tecnológicas u otras circunstancias.

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

En una realización, el mecanismo de barrera y sincronización 610 puede estar alojado o facilitado por el sistema operativo 606 del dispositivo informático 600. En otra realización, el mecanismo de barrera y sincronización 610 puede estar alojado en o ser parte de la unidad de procesamiento gráfico ("GPU" o simplemente "procesador gráfico") 614 o del firmware del procesador gráfico 614. Por ejemplo, el mecanismo de barrera y sincronización 610 puede estar integrado o implementado como parte del hardware de procesamiento del procesador gráfico 614. De manera similar, en otra realización más, el mecanismo de almacenamiento 610 puede estar alojado en o ser parte de la unidad central de procesamiento ("CPU" o simplemente "procesador de aplicaciones") 612. Por ejemplo, el mecanismo de barrera y sincronización 610 puede estar integrado o implementado como parte del hardware de procesamiento del procesador de aplicaciones 612. En otra realización más, el mecanismo de barrera y sincronización 610 puede estar alojado en o ser parte de cualquier número y tipo de componentes del dispositivo informático 600, tal como una porción del mecanismo de barrera y sincronización 610 puede estar alojada en o ser parte del sistema operativo 606, otra porción puede estar alojada en o ser parte del procesador gráfico 614, otra porción puede estar alojada en o ser parte del procesador de aplicaciones 612, mientras que una o más porciones del mecanismo de barrera y sincronización 610 pueden estar alojadas en o ser parte del sistema operativo 606 y/o cualquier número y tipo de dispositivos del dispositivo informático 600. Se contempla que una o más porciones o componentes del mecanismo de barrera y sincronización 610 pueden emplearse como hardware, software y/o firmware.

Se contempla que las realizaciones no están limitadas a una implementación o alojamiento particular del mecanismo de barrera y sincronización 610 y que el mecanismo de barrera y sincronización 610 y uno o más de sus componentes pueden implementarse como hardware, software, firmware o cualquier combinación de los mismos.

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

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

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

A lo largo del documento, el término "usuario" puede denominarse indistintamente "espectador", "observador", "persona", "individuo", "usuario final" y/o similares. Cabe señalar que, a lo largo de todo este documento, se puede hacer referencia a términos como "dominio de gráficos" de manera intercambiable con "unidad de procesamiento gráfico", "procesador

gráfico" o simplemente "GPU" y, de manera similar, "dominio de CPU" o "dominio anfitrión" se pueden hacer referencia de manera intercambiable a "unidad de procesamiento de ordenador", "procesador de aplicaciones" o simplemente "CPU".

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

La Figura 7 ilustra el mecanismo de barrera y sincronización 610 de la Figura 6 de acuerdo con una realización. Para abreviar, muchos de los detalles ya analizados con referencia a las Figuras 1-6 no se repiten ni se analizan a continuación. En una realización, el mecanismo de barrera y sincronización 610 puede incluir cualquier número y tipo de componentes, tales como (sin limitaciones): lógica de detección/observación 701; lógica de barrera 703; lógica de sincronización 705; lógica de comunicación/compatibilidad 707; y lógica de anticipación y planificación 709.

Con respecto a las barreras, las técnicas convencionales están severamente limitadas ya que se centran principalmente en la sincronización basada en software, que tiene sus limitaciones e ineficiencias en términos de costo y recursos, tales como energía, tiempo, memoria, etc. Además, dichas técnicas no prevén un enfoque global para todos los procesadores, tales como las GPU.

Las realizaciones proporcionan una técnica novedosa para emplear barreras de múltiples matrices para el aprendizaje automático donde se proporciona sincronización entre GPU, tal como se ofrece aceleración de hardware para barrera entre matrices de GPU.

Se contempla que las barreras se utilizan típicamente para sincronizar hilos dentro de un grupo de hilos. Sin embargo, cualquier hilo en un grupo de hilos que ejecute una instrucción de barrera se detiene hasta que todos los hilos en el grupo lleguen al mismo punto señalado por ellos al ejecutar la instrucción de barrera. Dicho de otra manera, las instrucciones de barrera requieren hardware de señalización y sincronización, lo que las limita a un agrupación o bloque de elementos de cálculo, porque cualquier grupo de hilos necesitaría programarse dentro de su bloque/agrupación correspondiente de elementos de cálculo. Dado que el aprendizaje automático puede tener una cantidad bastante grande de hilos, esta restricción de hilos a una agrupación/bloque de recursos computacionales más pequeño limita el rendimiento general.

En una realización, utilizando la lógica de barrera 703, se proporciona una nueva barrera de múltiples matrices para hilos en un grupo de hilos, de modo que los hilos se puedan planificar en un conjunto más grande de elementos de cálculo como se ilustra con respecto a la Figura 8A. Por ejemplo, la lógica de detección/observación 701 se puede utilizar para detectar cualquier número y tipo de hilos y grupos de hilos que necesiten planificación o despacho que se puede realizar a través de una barrera de múltiples matrices.

En una realización, utilizando la lógica de barrera 703, se emplea una barrera de múltiples matrices para facilitar la planificación de hilos y/o grupos de hilos en todos los elementos de cálculo sin estar atado a un bloque particular de elementos de cálculo. Además, para garantizar el correcto funcionamiento de esta técnica, la lógica de barrera 703 facilita el control y la sincronización necesarios para permitir una comunicación fluida de datos/información entre todos los chips asociados con la barrera de múltiples matrices, de modo que se pueda mantener la sincronización y el orden entre varias GPU, como el procesador gráfico 614.

Las realizaciones proporcionan además una técnica novedosa para la sincronización previa al hilo dentro y entre grupos de hilos, tal como lo facilita la lógica de sincronización 705. En una realización, la lógica de sincronización 705 se puede utilizar para mantener la sincronización no solo dentro de un grupo de hilos, sino también a través de múltiples grupos de hilos extendiendo una valla a través de esos grupos de hilos sin tener la necesidad de requerir una barrera.

Las interfaces de programación de aplicaciones (API) actuales permiten la sincronización solo dentro de un grupo de hilos (por ejemplo, mediante el uso de barreras de grupo de hilos). En una realización, utilizando la lógica de sincronización 705, se agrega un nuevo comando de barrera para la sincronización de grupos de hilos cruzados, de modo que se puedan sincronizar múltiples grupos de hilos. Por ejemplo, un comando de barrera de hilo X puede estar asociado con una lista de identificaciones (ID) de grupos de hilos como parte del argumento, donde estos grupos de hilos enumerados en el argumento e identificados por sus ID pueden luego sincronizarse. Además, dado que diferentes grupos de hilos pueden ejecutarse en diferentes multiprocesadores de transmisión (SM), el hardware de la GPU, tal como el hardware del procesador gráfico 614, puede agregar una barra transversal o una estructura DSS para comunicar la sincronización de barrera a través de esos SM. Esto se ilustra en la Figura 8B.

Las realizaciones proporcionan además la sincronización de grupos de hilos utilizando barreras, tal como utilizando una barrera global para sincronizar entre grupos de hilos como lo facilita la lógica de sincronización 705. Por ejemplo, como se ilustra con respecto a la Figura 8C, la compartición de datos entre GPU, tal como el procesador gráfico 614, se puede realizar utilizando una biblioteca de superficies, tal como las GPU que trabajan en redes neuronales convolucionales (RNC) que pueden guardar los datos de la RN intermedia como una superficie de datos en una base de datos en la nube. Otros sistemas de aprendizaje profundo que trabajan en los mismos problemas o en problemas similares pueden usar los datos de RN intermedios, tales como los vehículos autónomos, como la máquina autónoma 600, que están uno al lado del otro para tener una vista similar y pueden compartir los datos de RN.

Además, en una realización, utilizando la lógica de sincronización 705, cualquier superficie producida de esta manera puede ser comprimida opcionalmente para un tiempo de transmisión más rápido, donde las superficies compartidas pueden usarse para verificar los resultados mediante dos o más sistemas de aprendizaje profundo, tal como la máquina autónoma 600. Esto puede resultar en una mayor capacidad de tolerancia a fallas, por ejemplo, dos vehículos que están uno al lado del otro pueden verificar que sus superficies de RN intermedias estén lo suficientemente cerca.

Las realizaciones proporcionan además una nueva técnica para planificar grupos de hilos a través de SM (o DSS) o GPU donde estos grupos de hilos no tienen acceso a una memoria local compartida (SLM).

En las implementaciones de GPU actuales, todos los hilos en un grupo de hilos se ejecutan en el mismo SM (o DSS) ya que comparten el mismo SLM y/o usan barreras para sincronizarse. Por ejemplo, las barreras de hardware y SLM se implementan por SM y, por lo tanto, para que las barreras y SLM funcionen, todos los hilos del grupo de hilos se despachan en el mismo SM.

En una realización, se utilizan paradas en los despachos de tal manera que, asumiendo que una GPU tiene 2 SM, entonces en cualquier momento dado, SM0 tiene 10 hilos libres y SM1 tiene 6 hilos libres, donde un nuevo grupo de hilos puede entonces estar listo para ser planificado con 16 hilos. En este caso, actualmente, ninguno de los 2 SM puede encajar en el hueco de hilos ya que no tienen 16 hilos libres. Sin embargo, entre 2SM, hay una cantidad suficiente de hilos libres para planificar el nuevo grupo de hilos.

En una realización, utilizando la lógica de anticipación y planificación 709, la política de planificador de hilos se puede cambiar de manera tal que un compilador y/o un controlador, tal como el controlador 616 de la Figura 6, pueda pasar y determinar si un grupo de hilos está utilizando alguna barrera junto con la detección del tamaño de SLM que utiliza el grupo de hilos. Ahora bien, si un grupo de hilos no utiliza una barrera o SLM, entonces se facilita un planificador de hilos mediante la lógica de anticipación y planificación 709 para planificar ese grupo de hilos en múltiples SM.

Sin embargo, si el grupo de hilos utiliza una pequeña cantidad de SLM que no se considera crítica para el rendimiento del sistema por la lógica de anticipación y planificación 709, entonces el planificador de hilos se ve facilitado por la lógica de anticipación y planificación 709 para mapear ese espacio de SLM a la memoria global y posteriormente planificar ese grupo de hilos en múltiples SM. En algunas realizaciones, el compilador puede verse facilitado por la lógica de anticipación y planificación 709 para estimar si es aceptable (por ejemplo, en términos de rendimiento) mapear el espacio SLM a la memoria global y pasar esa sugerencia al planificador de hilos. Estas sugerencias pueden luego convertirse en parte de un estado de grupo de hilos en la memoria.

Las realizaciones proporcionan además una técnica novedosa para la preferencia de grupos de hilos basada en barreras, tal como lo facilita la lógica de anticipación y programación 709.

Se contempla que se utilicen barreras para sincronizar hilos dentro de un grupo de hilos. Cualquier hilo en un grupo de hilos determinado que ejecute una instrucción de barrera puede detenerse hasta que todos los hilos en el grupo alcancen ese punto según lo indicado por los grupos de hilos que ejecutan la instrucción de barrera. Esta instrucción de barrera puede necesitar hardware de señalización y sincronización, donde un hilo en un grupo de hilos ejecuta barreras y espera a todos los demás hilos. Si bien la mayoría de los hilos pueden haber completado la ejecución al alcanzar la barrera, el proceso puede aún estancarse hasta que todos los demás hilos o grupos de hilos hayan alcanzado ese punto. La necesidad de sincronización puede provocar una utilización subóptima de los recursos de hardware y una reducción del rendimiento de la carga de trabajo.

En una realización, la lógica de anticipación y programación 709 se puede utilizar para permitir que se anticipen grupos de hilos, mientras se permite que otros grupos de hilos continúen progresando. Por ejemplo, cuando el segundo grupo de hilos completa la ejecución o entra en un estado de bloqueo de barrera similar, el primer grupo de hilos puede cambiar a ejecución. En una realización, la lógica de anticipación y programación 709 puede permitir que el planificador de hilos mantenga la lista y el estado del grupo de hilos anticipados, si bien puede tener una limitación máxima en cuanto a cuántos grupos de hilos pueden anticiparse en un momento dado para conservar y mantener los controles de seguimiento a un tamaño manejable.

Además, la lógica de comunicación/compatibilidad 707 se puede utilizar para facilitar la comunicación y compatibilidad necesarias entre cualquier número de dispositivos del dispositivo informático 600 y varios componentes del mecanismo de barrera y sincronización 610.

La lógica de comunicación/compatibilidad 707 se puede utilizar para facilitar la comunicación dinámica y la compatibilidad entre el dispositivo informático 600 y cualquier número y tipo de otros dispositivos informáticos (tales como un dispositivo informático móvil, una computadora de escritorio, un dispositivo informático de servidor, etc.); dispositivos o componentes de procesamiento (tales como CPU, GPU, etc.); dispositivos de captura/detección (tales como componentes de captura/detección que incluyen cámaras, cámaras de detección de profundidad, sensores de cámara, sensores rojo-verde-azul ("RGB" o "rgb"), micrófonos, etc.); dispositivos de visualización (tales como componentes de salida que incluyen pantallas de visualización, áreas de visualización, proyectores de visualización, etc.); componentes de conocimiento del usuario/contexto y/o sensores/dispositivos de identificación/verificación (tales como sensores/detectores biométricos, escáneres, etc.); bases de datos 730, tales como dispositivos de memoria o almacenamiento, bases de datos y/o fuentes de datos (tales como dispositivos de almacenamiento de datos, unidades de disco duro, unidades de estado sólido, discos duros, tarjetas o dispositivos de memoria, circuitos de memoria, etc.); medios de comunicación 725, tales como uno o más canales o redes de comunicación (por ejemplo, redes en la nube, Internet, intranets, redes móviles, redes de proximidad, tales como Bluetooth, Bluetooth de baja energía (BLE), Bluetooth Smart, proximidad Wi-Fi, identificación por radiofrecuencia (RFID), comunicación de campo cercano (NFC), red de área corporal (BAN), etc.); comunicaciones inalámbricas o cableadas y protocolos relevantes (por ejemplo, Wi-Fi®, WiMAX, Ethernet, etc.); técnicas de gestión de conectividad y ubicación; aplicaciones/sitios web de software (por ejemplo, sitios web de redes sociales y/o comerciales, etc., aplicaciones comerciales, juegos y otras aplicaciones de entretenimiento, etc.); y lenguajes de programación, etc., al tiempo que se garantiza la compatibilidad con tecnologías, parámetros, protocolos, estándares, etc. cambiantes.

Además, cualquier uso de una marca, palabra, término, frase, nombre y/o acrónimo en particular, como "detectar", "observar", "hilo", "grupo de hilos", "memoria", "recurso informático", "planificación", "sincronizar", "anticipación", "multiprocesador de transmisión", "barrera", "barrera de múltiples matrices", "conjunto de entrenamiento", "máquina autónoma", "agente", "máquina", "vehículo", "robot", "conducción", "RNC", "RNP", "RN", "unidad de ejecución", "UE", "memoria local compartida", "SLM", "flujos de gráficos", "caché", "caché de gráficos", "GPU", "procesador gráfico", "dominio de GPU", "GPGPU", "CPU", "procesador de aplicaciones", "dominio de CPU", "controlador gráfico", "carga de trabajo", "aplicación", "canalización de gráficos", "procesos de canalización", "API", "API 3D", "OpenGL®", "DirectX®", "hardware", "software", "agente", "controlador gráfico", "controlador gráfico en modo núcleo", "controlador de modo usuario", "marco de controlador de modo usuario", "memoria intermedia", "memorias intermedia de gráficos", "tarea", "proceso", "operación", "aplicación de software", "juego", etc., no deben interpretarse como que limitan las realizaciones a software o dispositivos que llevan esa etiqueta en productos o en literatura externa a este documento.

Se contempla que se puede agregar y/o quitar cualquier cantidad y tipo de componentes del mecanismo de barrera y sincronización 610 para facilitar diversas realizaciones que incluyen agregar, quitar y/o mejorar ciertas características. Para mayor brevedad, claridad y facilidad de comprensión del mecanismo de barrera y sincronización 610, muchos de los componentes estándar y/o conocidos, tales como los de un dispositivo informático, no se muestran ni se analizan aquí. Se contempla que las realizaciones, como se describen en el presente documento, no se limiten a ninguna tecnología, topología, sistema, arquitectura y/o norma particular y sean lo suficientemente dinámicas para adoptar y adaptarse a cualquier cambio futuro.

La Figura 8A ilustra una configuración arquitectónica 800 para emplear y utilizar una barrera de múltiples matrices para el aprendizaje automático de acuerdo con una realización. Para mayor brevedad, muchos de los detalles discutidos previamente con referencia a las Figuras 1-7 pueden no ser discutidos o repetidos en adelante. Además, las realizaciones no están limitadas a ninguna ubicación arquitectónica, entorno, configuración o estructura particular de procesos y/o componentes, tal como la configuración 800.

Como se discutió previamente, las técnicas convencionales son ineficientes y no son adecuadas para el aprendizaje automático, ya que estas técnicas a menudo conducen al estancamiento de los hilos o grupos de hilos hasta que todos los hilos o grupos de hilos han alcanzado el mismo punto, lo cual es señalado por esos hilos o grupos de hilos que ejecutan la misma instrucción de barrera. En otras palabras, en las técnicas convencionales, los hilos o grupos de hilos están limitados a ser despachados o planificados a través de un bloque particular de elementos de cálculo y no más allá, lo que es particularmente problemático para el aprendizaje automático, ya que normalmente implicaría una gran cantidad de hilos.

Como se ilustra, en una realización, utilizando la lógica de barrera 703, se emplea una barrera de múltiples matrices en una máquina autónoma, tal como la máquina autónoma 600 de la Figura 6, para permitir que los hilos en los grupos de hilos se planifiquen en todos los elementos de cálculo y no se restrinjan a una agrupación o grupo particular de elementos de cálculo. Además, en una realización, como se ilustra, la lógica de barrera 703 proporciona comunicación con y entre cualquier número y tipo de procesadores, tales como GPU0 801, GPU1 803, GPU2 805, GPU3 807, junto con facilitar la sincronización de barrera entre las GPU 801-807 de modo que el control y la sincronización de la información se puedan mantener en todos los dispositivos de procesamiento, tales como las GPU 801-807, asociados con múltiples matrices. En una realización, la barrera de múltiples matrices puede ser una barrera de hardware para ofrecer la aceleración de

hardware que es necesaria para que una barrera funcione en las matrices de GPU 801-807. Sin embargo, las realizaciones no están limitadas como tales.

La Figura 8B ilustra un marco de trabajo 820 para facilitar la sincronización de grupos de hilos en el aprendizaje automático de acuerdo con una realización. Para mayor brevedad, muchos de los detalles discutidos previamente con referencia a las Figuras 1-8A pueden no ser discutidos o repetidos en adelante. Además, las realizaciones no se limitan a ninguna ubicación, estructura, configuración o estructura arquitectónica particular de procesos y/o componentes, tal como la estructura 820.

Como se discutió previamente con referencia a la Figura 7 y se ilustra aquí, el comando de barrera 831 que tiene el argumento 833 se puede utilizar para facilitar la sincronización entre múltiples grupos de hilos, tales como TG0 821, TG1 823, TG2 825 y TGN 827, que pueden ejecutarse en múltiples SM diferentes. Por ejemplo, la lógica de sincronización 705 de la Figura 7 se puede utilizar para facilitar un comando de barrera de hilo X, tal como el comando de barrera 831 que tiene el argumento 833 que enumera los ID de grupo de hilos. En una realización, el comando de barrera 831 se ejecuta para facilitar y mantener la sincronización de datos y la comunicación entre múltiples grupos de hilos 821-827 correspondientes a los ID de grupo de hilos del argumento 833. De esta manera, los grupos de hilos 821 a 827 se sincronizan incluso si se ejecutan en diferentes SM; el hardware de la GPU puede agregar una estructura de SM cruzado (o DSS) para comunicar la sincronización de barrera entre los SM.

La Figura 8C ilustra un marco de trabajo 840 para facilitar la compartición de datos entre GPU utilizando una biblioteca de superficies en el aprendizaje automático de acuerdo con una realización. Para mayor brevedad, muchos de los detalles discutidos previamente con referencia a las Figuras 1-8B pueden no ser discutidos o repetidos en adelante. Además, las realizaciones no se limitan a ninguna ubicación, estructura, configuración o estructura arquitectónica particular de procesos y/o componentes, tal como la estructura 840.

Como se ilustra aquí y se analiza con referencia a la Figura 7, cualquier dato de RN intermedio puede ser guardado por cada sistema DL 841, 843, 845 como datos de superficie en una o más bases de datos 730, tales como una base de datos en la nube, a través de medios de comunicación 725, tales como una red en la nube. Se contempla que los sistemas DL 841, 843, 845 pueden representar cada uno una máquina autónoma, tal como un vehículo autónomo, y pueden estar muy cerca uno del otro, tal como una cantidad de vehículos autónomos que están juntos en la misma carretera.

En algunas realizaciones, cualquier dato de superficie producido por los sistemas DL 841, 843, 845 se puede comprimir para una transmisión más rápida entre los sistemas DL 841, 843, 845 y cualquier otro sistema o entidad DL que sea o se convierta en parte del marco de trabajo 840. De esta manera, cualquier número de sistemas DL 841, 843, 845 pueden comunicar datos/información entre sí a través de los medios de comunicación 725, tal como por ejemplo verificando que sus superficies de RN intermedias estén lo suficientemente cerca.

La Figura 9A ilustra un procesador gráfico 614 para facilitar la planificación optimizada de grupos de hilos sin barreras o memoria local compartida en el aprendizaje automático de acuerdo con una realización. Para mayor brevedad, muchos de los detalles discutidos previamente con referencia a las Figuras 1-8C pueden no ser discutidos o repetidos en adelante.

Como se ilustra aquí y se discutió previamente con respecto a la Figura 7, el procesador gráfico 614 se muestra como si tuviera múltiples SM, SMM o DSS, tales como SMM0 905 y SMM1 907. La realización ilustrada muestra además los SMM 905 y 907 en comunicación con el planificador de hilos 901 que es capaz de recibir información o sugerencias 903 relacionadas con SLM, barreras, etc., desde un compilador y/o un controlador como lo facilita la lógica de anticipación y programación 709 de la Figura 7.

Por ejemplo, en una realización, EU0 911A, EU1 911B, EU2 913A y EU3 913B se muestran como pertenecientes al mismo grupo de hilos y planificados en SM, tales como SMM 907, 907, ya que este grupo de hilos en particular puede no utilizar una barrera 915, 917 o SLM 921, 923. Por ejemplo, si el grupo de hilos utiliza una pequeña cantidad de SLM que no se determina como crítica para el rendimiento, entonces el planificador de hilos 901 puede simplemente mapear ese espacio de SLM a la memoria global, lo que le permite planificar el grupo de hilos en múltiples SM, tales como los SMM 907, 907.

La Figura 9B ilustra un marco de trabajo 950 para facilitar la anticipación de grupos de hilos en función de barreras en el aprendizaje automático de acuerdo con una realización. Para mayor brevedad, muchos de los detalles discutidos previamente con referencia a las Figuras 1-9A pueden no ser discutidos o repetidos en adelante.

Como se ilustra aquí y se discutió previamente con referencia a la Figura 7, una cantidad de grupos de hilos, tales como TG0 951, TG1 953 y TG2 955, se muestran como estando en comunicación con recursos computacionales (por ejemplo, elementos computacionales, bloques computacionales, etc.) y el planificador de hilos 959 (tal como el planificador de hilos de hardware) como lo facilita la lógica de anticipación y programación 709 de la Figura 7.

Como se mencionó anteriormente, utilizando la lógica de anticipación y programación 709 de la Figura 7, por ejemplo, si se deben despachar y ejecutar tres TG 951, 953, 955 y si TG1 953 completa la ejecución o ingresa en un estado de bloqueo de barrera similar, entonces, por ejemplo, TG0 951 puede ser conmutado a ejecución por el planificador de hilos 959 sin tener que detener todo el proceso utilizando recursos computacionales 957. De manera similar, al utilizar esta nueva técnica y la disponibilidad de recursos computacionales, cuando un hilo en un grupo de hilos ejecuta una barrera, no tiene que entrar en una parada y esperar hasta que todos los demás hilos alcancen ese punto de barrera. Esto se logra mediante el uso del planificador de hilos 959, como lo facilita la lógica de anticipación y programación 709 de la Figura 7.

Visión General del Aprendizaje Automático

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

Un tipo ejemplar 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 propagación hacia adelante se puede implementar como un grafo acíclico en el que los nodos están dispuestos en capas. Normalmente, una topología de red de propagación hacia adelante 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 representado por el algoritmo que se está ejecutando, la salida del algoritmo de red neuronal puede tomar varias formas.

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

La precisión de un algoritmo de aprendizaje automático puede verse afectada significativamente por la calidad del conjunto de datos utilizado para entrenar el algoritmo. El proceso de entrenamiento puede requerir un gran esfuerzo computacional y una cantidad significativa de tiempo en un procesador convencional de propósito general. En consecuencia, se utiliza hardware de procesamiento paralelo para entrenar muchos tipos de algoritmos de aprendizaje automático. Esto es particularmente útil para optimizar el entrenamiento de redes neuronales, ya que los cálculos realizados para ajustar los coeficientes en las redes neuronales se prestan naturalmente a implementaciones paralelas. En concreto, muchos algoritmos de aprendizaje automático y aplicaciones de software se han adaptado para utilizar el hardware de procesamiento paralelo dentro de dispositivos de procesamiento gráfico de propósito general.

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

La aceleración de hardware para la aplicación de aprendizaje automático 1002 se puede habilitar a través de un marco de trabajo de aprendizaje automático 1004. El marco de trabajo de aprendizaje automático 1004 puede proporcionar una biblioteca de primitivas de aprendizaje automático. Las primitivas de aprendizaje automático son operaciones básicas que

comúnmente realizan los algoritmos de aprendizaje automático. Sin el marco de trabajo de aprendizaje automático 1004, los desarrolladores de algoritmos de aprendizaje automático tendrían que crear y optimizar la lógica computacional principal asociada con el algoritmo de aprendizaje automático y luego volver a optimizar la lógica computacional a medida que se desarrollan nuevos procesadores paralelos. En cambio, la aplicación de aprendizaje automático se puede configurar para realizar los cálculos necesarios utilizando las primitivas proporcionadas por el marco de trabajo de aprendizaje automático 1004. Las primitivas ejemplares incluyen convoluciones tensoriales, funciones de activación y agrupamiento, que son operaciones computacionales que se realizan durante el entrenamiento de una red neuronal convolucional (RNC). El marco de trabajo de aprendizaje automático 1004 también puede proporcionar primitivas para implementar subprogramas de álgebra lineal básica realizados por muchos algoritmos de aprendizaje automático, tales como operaciones matriciales y vectoriales.

El marco de trabajo de aprendizaje automático 1004 puede procesar datos de entrada recibidos desde la aplicación de aprendizaje automático 1002 y generar la entrada apropiada para un marco de trabajo de cálculo 1006. El marco de trabajo de cálculo 1006 puede abstraer las instrucciones subyacentes proporcionadas al controlador GPGPU 1008 para permitir que el marco de trabajo de aprendizaje automático 1004 aproveche la aceleración de hardware a través del hardware GPGPU 1010 sin requerir que el marco de trabajo de aprendizaje automático 1004 tenga un conocimiento profundo de la arquitectura del hardware GPGPU 1010. Además, el marco de trabajo de cálculo 1006 puede habilitar la aceleración de hardware para el marco de trabajo de aprendizaje automático 1004 en una variedad de tipos y generaciones de hardware GPGPU 1010.

Aceleración de Aprendizaje Automático de GPGPU

La Figura 11 ilustra una unidad de procesamiento gráfico de propósito general altamente paralela 1100, de acuerdo con una realización. En una realización, la unidad de procesamiento de propósito general (GPGPU) 1100 se puede configurar para ser particularmente eficiente en el procesamiento del tipo de cargas de trabajo computacionales asociadas con el entrenamiento de redes neuronales profundas. Además, la GPGPU 1100 se puede vincular directamente a otras instancias de la GPGPU para crear una agrupación de múltiples GPU para mejorar la velocidad de entrenamiento para redes neuronales particularmente profundas.

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

La GPGPU 1100 incluye la memoria 1114A-B acoplada con las agrupaciones de cálculo 1106A-H a través de un conjunto de controladores de memoria 1112A-B. En varias realizaciones, la memoria 1114A-B puede incluir varios tipos de dispositivos de memoria, incluyendo memoria dinámica de acceso aleatorio (DRAM) o memoria de acceso aleatorio de gráficos, tal como memoria de acceso aleatorio de gráficos síncrónica (SGRAM), incluyendo memoria de velocidad de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 224A-N también pueden incluir memoria apilada 3D, que incluye, entre otras, memoria de alto ancho de banda (HBM).

En una realización, cada agrupación de cálculo GPLAB06A-H incluye un conjunto de multiprocesadores gráficos, como el multiprocesador gráfico 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 1106A-H puede estar configurado para realizar operaciones de coma flotante de 16 bits o de 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante puede estar configurado para realizar operaciones de coma flotante de 64 bits.

Se pueden configurar múltiples instancias de la GPGPU 1100 para operar como una agrupación de cálculo. El mecanismo de comunicación utilizado por la agrupación de cálculo para la sincronización y el intercambio de datos varía de acuerdo con las realizaciones. En una realización, las múltiples instancias de la GPGPU 1100 se comunican a través de la interfaz anfitrión 1102. En una realización, la GPGPU 1100 incluye un concentrador de E/S 1108 que acopla la GPGPU 1100 con un enlace de GPU 1110 que permite una conexión directa a otras instancias de la GPGPU. En una realización, el enlace de GPU 1110 está acoplado a un puente de GPU a GPU dedicado que permite la comunicación y sincronización entre múltiples instancias de la GPGPU 1100. En una realización, el enlace GPU 1110 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 1100 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red al que se puede acceder a través de la interfaz anfitrión 1102. En una realización, el enlace de

GPU 1110 se puede configurar para permitir una conexión a un procesador anfitrión además de o como alternativa a la interfaz anfitrión 1102.

Si bien la configuración ilustrada de la GPGPU 1100 se puede configurar para entrenar redes neuronales, una realización proporciona una configuración alternativa de la GPGPU 1100 que se puede configurar para su implementación dentro de una plataforma de inferencia de alto rendimiento o de bajo consumo. En una configuración de inferencia, la GPGPU 1100 incluye menos agrupaciones de cálculo 1106A-H en relación con la configuración de entrenamiento. Además, la tecnología de memoria asociada con la memoria 1114A-B puede diferir entre las configuraciones de inferencia y entrenamiento. En una realización, la configuración de inferencia de la GPGPU 1100 puede soportar la inferencia de instrucciones específicas. 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 utilizan comúnmente durante operaciones de inferencia para redes neuronales implementadas.

La Figura 12 ilustra un sistema informático de múltiples GPU 1200, de acuerdo con una realización. El sistema informático de múltiples GPU 1200 puede incluir un procesador 1202 acoplado a múltiples GPGPU 1206A-D mediante un conmutador de interfaz anfitrión 1204. El conmutador de interfaz anfitrión 1204, en una realización, es un dispositivo de conmutación PCI Express que acopla el procesador 1202 a un bus PCI Express a través del cual el procesador 1202 puede comunicarse con el conjunto de GPGPU 1206A-D. Cada una de las múltiples GPGPU 1206A-D puede ser una instancia de la GPGPU 1100 de la Figura 11. Las GPGPU 1206A-D pueden interconectarse mediante un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 1216. Los enlaces de GPU a GPU de alta velocidad pueden conectarse a cada una de las GPGPU 1206A-D a través de un enlace de GPU dedicado, tal como el enlace de GPU 1110 como en la Figura 11. Los enlaces GPU P2P 1216 permiten la comunicación directa entre cada una de las GPGPU 1206A-D sin requerir comunicación a través del bus de interfaz anfitrión al que está conectado el procesador 1202. Con el tráfico de GPU a GPU dirigido a los enlaces de GPU P2P, el bus de interfaz anfitrión permanece disponible para el acceso a memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 1200, por ejemplo, a través de uno o más dispositivos de red. Mientras que en la realización ilustrada las GPGPU 1206A-D se conectan al procesador 1202 a través del conmutador de interfaz anfitrión 1204, en una realización el procesador 1202 incluye soporte directo para los enlaces GPU P2P 1216 y puede conectarse directamente a las GPGPU 1206A-D.

Implementaciones de Red Neuronal de Aprendizaje Automático

La arquitectura informática proporcionada por las realizaciones descritas en el presente documento se puede configurar para realizar los tipos de procesamiento paralelo que son particularmente adecuados para el entrenamiento y la implementación de redes neuronales para el aprendizaje automático. Una red neuronal puede generalizarse como una red de funciones que tienen una relación de grafo. Como es bien sabido en la técnica, existen diversos tipos de implementaciones de redes neuronales utilizadas en el aprendizaje automático. Un tipo ejemplar de red neuronal es la red de propagación hacia adelante, como se describió anteriormente.

Un segundo tipo ejemplar de red neuronal es la red neuronal convolucional (RNC). Una RNC es una red neuronal de propagación hacia adelante especializada para procesar datos que tienen una topología conocida tipo cuadrícula, tal como datos de imágenes. En consecuencia, las RNC se utilizan comúnmente para aplicaciones de visión computacional y reconocimiento de imágenes, pero también pueden usarse para otros tipos de reconocimiento de patrones, tal como el procesamiento del habla y del lenguaje. Los nodos de la capa de entrada de la RNC están organizados en un conjunto de "filtros" (detectores de características inspirados en los campos receptivos que se encuentran en la retina), y la salida de cada conjunto de filtros se propaga a los nodos en capas sucesivas de la red. Los cálculos para una RNC incluyen la aplicación de 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 de la convolución puede denominarse entrada, mientras que la segunda función puede denominarse núcleo de convolución. El resultado puede denominarse mapa de características. Por ejemplo, la entrada a una capa de convolución puede ser una matriz multidimensional de datos que define los diversos componentes de color de una imagen de entrada. El núcleo de convolución puede ser una matriz multidimensional de parámetros, donde los parámetros están adaptados por el proceso de entrenamiento para la red neuronal.

Las redes neuronales recurrentes (RNR) son una familia de redes neuronales de propagación hacia adelante que incluyen conexiones de retroalimentación entre capas. Las RNR permiten modelar datos secuenciales al compartir datos de parámetros entre 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 momento futuro, ya que al menos una parte de los datos de salida de la RNR se utiliza como retroalimentación para procesar la entrada posterior en una secuencia. Esta característica hace que las RNR sean particularmente útiles para el procesamiento del lenguaje debido a la naturaleza variable en que se pueden componer los datos del lenguaje.

Las figuras descritas a continuación presentan redes de propagación hacia adelante, RNC y RNR a modo de ejemplo, y describen además un proceso general para entrenar e implementar respectivamente cada uno de esos tipos de redes. Se entenderá que estas descripciones son ejemplares y no limitativas en cuanto a cualquier realización específica descrita en el presente documento y los conceptos ilustrados se pueden aplicar de manera general a redes neuronales profundas y técnicas de aprendizaje automático en general.

Las redes neuronales ejemplares descritas anteriormente se pueden utilizar para realizar aprendizaje profundo. El aprendizaje profundo es el aprendizaje automático que utiliza redes neuronales profundas. Las redes neuronales profundas utilizadas en el aprendizaje profundo son redes neuronales artificiales compuestas de múltiples capas ocultas, a diferencia de las redes neuronales superficiales que incluyen solo 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 permiten el reconocimiento de patrones de múltiples pasos, lo que resulta en un error de salida reducido en relación con las técnicas de aprendizaje automático superficiales.

Las redes neuronales profundas utilizadas en el aprendizaje profundo normalmente incluyen una red frontal para realizar el reconocimiento de características acoplada a una red servidor (back-end) que representa un modelo matemático que puede realizar operaciones (por ejemplo, clasificación de objetos, reconocimiento de voz, etc.) en función de la representación de características proporcionada al modelo. El aprendizaje profundo permite realizar aprendizaje automático sin necesidad de realizar ingeniería de características manualmente para el modelo. En cambio, las redes neuronales profundas pueden aprender características basadas en la estructura estadística o la correlación dentro de los datos de entrada. Las características aprendidas se pueden proporcionar a un modelo matemático que puede asignar las características detectadas a una salida. El modelo matemático utilizado por la red generalmente está especializado para la tarea específica a realizar, y se utilizarán diferentes modelos para realizar diferentes tareas.

Una vez estructurada la red neuronal, se puede aplicar un modelo de aprendizaje a la red para entrenarla para que realice 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 utilizado para entrenar redes neuronales. Se presenta un vector de entrada a la red para su procesamiento. La salida de la red se compara con la salida deseada utilizando una función de pérdida y se calcula un valor de error para cada una de las neuronas en la capa de salida. Luego, los valores de error se propagan hacia atrás hasta que cada neurona tiene un valor de error asociado que representa aproximadamente su contribución a la salida original. La red puede luego aprender de esos errores utilizando un algoritmo, tal como el algoritmo de descenso de gradiente estocástico, para actualizar los pesos de la red neuronal.

Las Figuras 13A-B ilustran una red neuronal convolucional ejemplar. La Figura 13A ilustra diversas capas dentro de una CNN. Como se muestra en la Figura 13A, una RNC ejemplar utilizada para modelar el procesamiento de imágenes puede recibir la entrada 1302 que describe los componentes rojo, verde y azul (RGB) de una imagen de entrada. La entrada 1302 puede ser procesada por múltiples capas convolucionales (por ejemplo, capa convolucional 1304, capa convolucional 1306). La salida de las múltiples capas convolucionales puede ser procesada opcionalmente por un conjunto de capas completamente conectadas 1308. Las neuronas en una capa completamente conectada tienen conexiones completas con todas las activaciones en la capa anterior, como se describió anteriormente para una red de propagación hacia adelante. La salida de las capas completamente conectadas 1308 se puede utilizar para generar un resultado de salida de la red. Las activaciones dentro de las capas completamente conectadas 1308 se pueden calcular utilizando la multiplicación de matrices en lugar de convolución. No todas las implementaciones de RNC utilizan capas completamente conectadas DPLA08. Por ejemplo, en algunas implementaciones la capa convolucional 1306 puede generar salida para la RNC.

Las capas convolucionales están escasamente conectadas, lo que difiere de la configuración de red neuronal tradicional que se encuentra en las capas completamente conectadas 1308. Las capas de redes neuronales tradicionales están completamente conectadas, de modo que cada unidad de salida interactúa con cada unidad de entrada. Sin embargo, las capas convolucionales están escasamente conectadas porque la salida de la convolución de un campo se utiliza como entrada (en lugar del valor de estado respectivo de cada uno de los nodos del campo) para los nodos de la capa subsiguiente, como se ilustra. Los núcleos asociados a 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 permite que la RNC escale para procesar imágenes grandes.

La Figura 13B ilustra etapas de cálculo ejemplares dentro de una capa convolucional de una RNC. La entrada a una capa convolucional 1312 de una RNC se puede procesar en tres etapas de una capa convolucional 1314. Las tres etapas pueden incluir una etapa de convolución 1316, una etapa de detector 1318 y una etapa de agrupamiento 1320. La capa de convolución 1314 puede luego enviar datos a una capa convolucional sucesiva. La capa convolucional final de la red puede generar datos de mapas de características de salida o proporcionar entrada a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la RNC.

En la etapa de convolución 1316 se realizan varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La etapa de convolución 1316 puede incluir una transformación afín, que es cualquier transformación que pueda

especificarse como una transformación lineal más una traslación. Las transformaciones afines incluyen rotaciones, traslaciones, escalado y combinaciones de estas transformaciones. La etapa de convolución calcula la salida de funciones (por ejemplo, neuronas) que están conectadas a regiones específicas en la entrada, que pueden determinarse como la región local asociada con la neurona. Las neuronas calculan un producto escalar entre los pesos de las neuronas y la región en la entrada local a la que están conectadas las neuronas. La salida de la etapa de convolución 1316 define un conjunto de activaciones lineales que son procesadas por etapas sucesivas de la capa convolucional 1314.

Las activaciones lineales pueden ser procesadas por una etapa de detector 1318. En la etapa de detector 1318, 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 utiliza una función de activación definida como f(x) = max(0, x), de modo que la activación tiene un umbral de cero.

La etapa de agrupamiento 1320 utiliza una función de agrupamiento que reemplaza la salida de la capa convolucional 1306 con una estadística de resumen de las salidas cercanas. La función de agrupamiento se puede utilizar para introducir invariancia de traducción en la red neuronal, de modo que pequeñas traducciones a la entrada no cambien las salidas agrupadas. La invariancia a la traducción local puede ser útil en escenarios donde la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Se pueden utilizar varios tipos de funciones de agrupamiento durante la etapa de agrupamiento 1320, incluidos el agrupamiento máximo, el agrupamiento promedio y el agrupamiento de norma l2. Además, algunas implementaciones de RNC no incluyen una etapa de agrupamiento. En lugar de ello, dichas implementaciones sustituyen una etapa de convolución adicional que tiene un paso mayor en relación con las etapas de convolución anteriores.

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

La Figura 14 ilustra una red neuronal recurrente 1400 ejemplar. En una red neuronal recurrente (RNR), el estado anterior de la red influye en la salida del estado actual de la red. Las RNR se pueden construir de distintas maneras utilizando una variedad de funciones. El uso de RNR generalmente gira en torno al uso de modelos matemáticos para predecir el futuro basándose en una secuencia previa de entradas. Por ejemplo, una RNR puede usarse para realizar modelos estadísticos del lenguaje para predecir una próxima palabra dada una secuencia previa de palabras. La RNR 1400 ilustrada se puede describir como que tiene una capa de entrada 1402 que recibe un vector de entrada, capas ocultas 1404 para implementar una función recurrente, un mecanismo de retroalimentación 1405 para habilitar una "memoria" de estados anteriores y una capa de salida 1406 para generar un resultado. La RNR 1400 funciona en base a pasos de tiempo. El estado de la RNR en un paso de tiempo determinado se ve influenciado en función del paso de tiempo anterior a través del mecanismo de retroalimentación 1405. Para un paso de tiempo determinado, el estado de las capas ocultas 1404 está definido por el estado anterior y la entrada en el paso de tiempo actual. Una entrada inicial (x1) en un primer paso puede ser procesada por la capa oculta 1404. La capa oculta 1404 puede procesar una segunda entrada (x2) utilizando información de estado que se determina durante el procesamiento de la entrada inicial (x1). Un estado dado se puede calcular como st = f(Uxt + Wst-1), donde U y W son matrices de parámetros. La función f es generalmente 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 utilizada en las capas ocultas 1404 puede variar dependiendo de los detalles de implementación específicos de la RNR 1400.

Además de las redes RNC y RNR básicas descritas, se pueden habilitar variaciones en esas redes. Una variante de RNN ilustrativa es la RNN de memoria a corto plazo larga (LSTM). Las RNR LSTM son capaces de aprender dependencias a largo plazo que pueden ser necesarias para procesar secuencias de lenguaje más largas. Una variante de la RNC es una red convolucional de creencias profundas, que tiene una estructura similar a una RNC y se entrena de manera similar a una red de creencias profundas. Una red de creencias profundas (RCP) es una red neuronal generativa que se compone de múltiples capas de variables estocásticas (aleatorias). Las RCP pueden entrenarse capa a capa usando aprendizaje no supervisado voraz. Los pesos aprendidos de la RCP se pueden utilizar luego para proporcionar redes neuronales entrenadas previamente determinando un conjunto inicial óptimo de pesos para la red neuronal.

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

Para iniciar el proceso de entrenamiento, los pesos iniciales se pueden elegir aleatoriamente o mediante entrenamiento previo utilizando una red de creencias profundas. El ciclo de entrenamiento se puede realizar de forma supervisada o no supervisada.

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

El aprendizaje no supervisado es un método de aprendizaje en el que la red intenta entrenarse a sí misma utilizando datos no etiquetados. Por lo tanto, para el aprendizaje no supervisado, el conjunto de datos de entrenamiento 1502 incluirá datos de entrada sin un dato de salida asociado. La red neuronal no entrenada 1506 puede aprender agrupaciones dentro de la entrada no etiquetada y puede determinar cómo se relacionan las entradas individuales con el conjunto de datos general. El entrenamiento no supervisado se puede utilizar para generar un mapa autoorganizado, que es un tipo de red neuronal entrenada 1507 capaz de realizar operaciones útiles para reducir la dimensionalidad de los datos. El entrenamiento no supervisado también se puede utilizar para realizar la detección de anomalías, lo que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

También se pueden emplear variaciones en el entrenamiento supervisado y no supervisado. El aprendizaje semisupervisado es una técnica en la que el conjunto de datos de entrenamiento 1502 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que los datos de entrada se utilizan continuamente para entrenar aún más el modelo. El aprendizaje incremental permite que la red neuronal entrenada 1508 se adapte a los nuevos datos 1512 sin olvidar el conocimiento inculcado en 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 en términos computacionales para un solo nodo de cálculo. En lugar de usar un único nodo de cálculo, puede usarse una red distribuida de nodos de cálculo para acelerar el proceso de entrenamiento.

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

En el paralelismo de modelos 1602, diferentes nodos computacionales en un sistema distribuido pueden realizar cálculos de entrenamiento para diferentes partes de una sola red. Por ejemplo, cada capa de una red neuronal puede ser entrenada por un nodo de procesamiento diferente del sistema distribuido. Los beneficios del paralelismo de modelos incluyen la capacidad de escalar a modelos particularmente grandes. Dividir los cálculos asociados con diferentes capas de la red neuronal permite el entrenamiento de redes neuronales muy grandes en las que los pesos de todas las capas no cabrían en la memoria de un solo nodo computacional. En algunos casos, el paralelismo de modelos puede ser particularmente útil para realizar entrenamiento no supervisado de redes neuronales grandes.

En el paralelismo de datos 1604, los diferentes nodos de la red distribuida tienen una instancia completa del modelo y cada nodo recibe una porción diferente de los datos. Luego se combinan los resultados de los diferentes nodos. Si bien son posibles diferentes enfoques para el paralelismo de datos, todos los enfoques de entrenamiento paralelo de datos requieren una técnica para combinar resultados y sincronizar los parámetros de modelo entre cada nodo. Los enfoques ejemplares para combinar datos incluyen el promedio de parámetros y el paralelismo de datos basado en actualizaciones. El promedio de parámetros entrena cada nodo en un subconjunto de los datos de entrenamiento y establece los parámetros globales (por ejemplo, pesos, polarizaciones) en el promedio de los parámetros de cada nodo. El promedio de parámetros utiliza un servidor de parámetros central que mantiene los datos de parámetros. El paralelismo de datos basado en actualizaciones es similar al promedio de parámetros, excepto que en lugar de transferir parámetros desde los nodos al servidor de

parámetros, se transfieren las actualizaciones al modelo. Además, el paralelismo de datos basado en actualizaciones se puede realizar de manera descentralizada, donde las actualizaciones se comprimen y transfieren entre nodos.

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

El entrenamiento distribuido tiene una sobrecarga mayor en relación con el entrenamiento en una sola máquina. Sin embargo, los procesadores paralelos y las GPGPU descritos en el presente documento pueden implementar varias técnicas para reducir la sobrecarga del entrenamiento distribuido, incluidas técnicas para permitir la transferencia de datos de GPU a GPU de alto ancho de banda y la sincronización de datos remota acelerada.

Aplicaciones Ejemplares de Aprendizaje Automático

El aprendizaje automático se puede aplicar para resolver una variedad de problemas tecnológicos, incluidos, entre otros, visión por computadora, conducción y navegación autónomas, reconocimiento de voz y procesamiento del lenguaje. La visión por computadora ha sido tradicionalmente una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Las aplicaciones de la visión por computadora varían desde la reproducción de capacidades visuales humanas, como el reconocimiento de rostros, hasta la creación de nuevas categorías de habilidades visuales. Por ejemplo, las aplicaciones de visión por computadora pueden configurarse para reconocer ondas de sonido de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesadores paralelos permite entrenar aplicaciones de visión por computadora utilizando conjuntos de datos de entrenamiento significativamente más grandes que los que eran posibles anteriormente y permite implementar sistemas de inferencia utilizando procesadores paralelos de bajo consumo.

El aprendizaje automático acelerado por procesadores paralelos tiene aplicaciones de conducción autónoma que incluyen reconocimiento de carriles y señales de tráfico, evitación de obstáculos, navegación y control de conducción. Se pueden utilizar técnicas de aprendizaje automático acelerado para entrenar modelos de conducción basados en conjuntos de datos que definen las respuestas apropiadas a una entrada de entrenamiento específica. Los procesadores paralelos descritos en el presente documento pueden permitir el entrenamiento rápido de las redes neuronales cada vez más complejas utilizadas para soluciones de conducción autónoma y permiten la implementación de procesadores de inferencia de bajo consumo en una plataforma móvil adecuada para la integración en vehículos autónomos.

Las redes neuronales profundas aceleradas por procesadores paralelos han permitido enfoques de aprendizaje automático para el reconocimiento automático de voz (ASR). El ASR incluye la creación de una función que calcula la secuencia lingüística más probable dada una secuencia acústica de entrada. El aprendizaje automático acelerado mediante redes neuronales profundas ha permitido reemplazar los modelos ocultos de Markov (HMM) y los modelos de mezcla gaussiana (GMM) utilizados anteriormente para el ASR.

El aprendizaje automático acelerado por procesador paralelo también se puede utilizar para acelerar el procesamiento de lenguaje natural. Los procedimientos de aprendizaje automático pueden utilizar algoritmos de inferencia estadística para producir modelos que sean robustos ante entradas erróneas o desconocidas. Entre las aplicaciones ejemplares de procesadores de lenguaje natural se incluye la traducción automática entre idiomas humanos.

Las plataformas de procesamiento paralelo utilizadas para el aprendizaje automático se pueden dividir en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento generalmente son muy paralelas e incluyen optimizaciones para acelerar el entrenamiento de un solo nodo con múltiples GPU y el entrenamiento de múltiples nodos y múltiples GPU. Los procesadores paralelos a modo de ejemplo adecuados para entrenamiento incluyen la unidad de procesamiento gráfico de propósito general altamente paralela 1100 de la Figura 1100 y el sistema informático de múltiples GPU 1200 de la Figura 1200. Por el contrario, las plataformas de aprendizaje automático desplegadas incluyen, en general, procesadores paralelos de potencia inferior adecuados para su uso en productos tales como cámaras, robots autónomos y vehículos autónomos.

La Figura 17 ilustra un sistema de inferencia ejemplar en chip (SOC) 1700 adecuado para realizar inferencias utilizando un modelo entrenado. El SOC 1700 puede integrar componentes de procesamiento que incluyen un procesador de medios 1702, un procesador de visión 1704, una GPGPU 1706 y un procesador de múltiples núcleos 1708. El SOC 1700 puede incluir además una memoria en chip 1705 que puede habilitar un grupo de datos en chip compartido al que pueden acceder cada uno de los componentes de procesamiento. Los componentes de procesamiento se pueden optimizar para un funcionamiento de bajo consumo para permitir la implementación en una variedad de plataformas de aprendizaje automático, incluidos vehículos autónomos y robots autónomos. Por ejemplo, una implementación del SOC 1700 se puede utilizar como parte del sistema de control principal de un vehículo autónomo. Cuando el SOC 1700 está configurado para su uso en vehículos autónomos, está diseñado y configurado para cumplir con los estándares de seguridad funcional pertinentes de la jurisdicción de despliegue.

Durante el funcionamiento, el procesador de medios 1702 y el procesador de visión 1704 pueden trabajar en conjunto para acelerar las operaciones de visión por computadora. El procesador de medios 1702 puede permitir la decodificación de baja latencia de múltiples transmisiones de vídeo de alta resolución (por ejemplo, 4K, 8K). Los flujos de vídeo decodificados se pueden escribir en una memoria intermedia en la memoria del chip 1705. El procesador de visión 1704 puede luego analizar el vídeo decodificado y realizar operaciones de procesamiento preliminar en los fotogramas del vídeo decodificado como preparación para el procesamiento de los fotogramas utilizando un modelo de reconocimiento de imágenes entrenado. Por ejemplo, el procesador de visión 1704 puede acelerar las operaciones de convolución para una RNC que se utiliza para realizar el reconocimiento de imágenes en los datos de vídeo de alta resolución, mientras que los cálculos del modelo servidor los realiza la GPGPU 1706.

El procesador multinúcleo 1708 puede incluir lógica de control para ayudar con la secuenciación y sincronización de transferencias de datos y operaciones de memoria compartida realizadas por el procesador de medios 1702 y el procesador de visión 1704. El procesador multinúcleo 1708 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 1706. Por ejemplo, al menos una parte de la lógica de navegación y conducción se puede implementar en un software que se ejecuta en el procesador multinúcleo 1708. Dicho software puede emitir directamente cargas de trabajo computacionales a la GPGPU 1706 o las cargas de trabajo computacionales pueden emitirse al procesador multinúcleo 1708, que puede descargar al menos una parte de esas operaciones a la GPGPU 1706.

La GPGPU 1706 puede incluir agrupaciones de cálculo tales como una configuración de bajo consumo de energía de los agrupaciones de cálculo 1106A-1106H dentro de la unidad de procesamiento gráfico de propósito general altamente paralela 1100. Las agrupaciones de cálculo dentro de GPGPU 1706 pueden soportar instrucciones específicamente optimizadas para realizar cálculos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1706 puede soportar instrucciones para realizar cálculos de baja precisión, tales como operaciones con vectores enteros de 8 y 4 bits.

Vista General del Sistema II

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

Una realización del sistema 1800 puede incluir, o estar incorporada dentro de una plataforma de juego basada en servidor, una consola de juegos, incluyendo una consola de juegos y medios, una consola de juegos móvil, una consola de juegos de mano o una consola de juegos en línea. En algunas realizaciones, el sistema 1800 es un teléfono móvil, un teléfono inteligente, un dispositivo informático tipo tableta o un dispositivo de Internet móvil. El sistema de procesamiento de datos 1800 también puede incluir, acoplarse con o integrarse dentro de un dispositivo llevable, tal como un dispositivo llevable de reloj inteligente, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 1800 es un dispositivo de televisión o decodificador que tiene uno o más procesadores 1802 y una interfaz gráfica generada por uno o más procesadores gráficos 1808.

En algunas realizaciones, los uno o más procesadores 1802 incluyen en cada caso uno o más núcleos de procesador 1807 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para el sistema y el software de usuario. En algunas realizaciones, cada uno de los uno o más núcleos de procesador 1807 está configurado para procesar un conjunto de instrucciones 1809 específico. En algunas realizaciones, el conjunto de instrucciones 1809 puede facilitar la computación de conjunto de instrucciones complejo (CISC), la computación de conjunto de instrucciones reducido (RISC) o la computación a través de una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 1807 pueden procesar cada uno un conjunto de instrucciones 1809 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 1807 también puede incluir otros dispositivos de procesamiento, tal como un procesador de señal digital (DSP).

En algunas realizaciones, el procesador 1802 incluye una memoria caché 1804. Dependiendo de la arquitectura, el procesador 1802 puede tener una sola caché interna o múltiples niveles de caché interna. En algunas realizaciones, la memoria caché se comparte entre varios componentes del procesador 1802. En algunas realizaciones, el procesador 1802 también utiliza una caché externa (por ejemplo, una caché de nivel 3 (L3) o una caché de último nivel (LLC)) (no mostrada), que puede compartirse entre los núcleos de procesador 1807 utilizando técnicas de coherencia de caché conocidas. Además, en el procesador 1802 se incluye un archivo de registro 1806 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de punto flotante, registros de estado y un registro de puntero de instrucciones). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos del diseño del procesador 1802.

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

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

En algunas realizaciones, el ICH 1830 permite que los periféricos se conecten al dispositivo de memoria 1820 y al procesador 1802 a través de un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, entre otros, un controlador de audio 1846, una interfaz de firmware 1828, un transceptor inalámbrico 1826 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 1824 (por ejemplo, unidad de disco duro, memoria flash, etc.) y un controlador de E/S heredado 1840 para acoplar dispositivos heredados (por ejemplo, Sistema Personal 2 (PS/2)) al sistema. Uno o más controladores de bus serie universal (USB) 1842 conectan dispositivos de entrada, tales como combinaciones de teclado y ratón 1844. Un controlador de red 1834 también puede acoplarse al ICH 1830. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla al bus de procesador 1810. Se apreciará que el sistema 1800 mostrado es ejemplar y no limitativo, ya que también se pueden utilizar otros tipos de sistemas de procesamiento de datos que estén configurados de manera diferente. Por ejemplo, el concentrador de controlador de E/S 1830 puede estar integrado dentro de uno o más procesadores 1802, o el concentrador de controlador de memoria 1816 y el concentrador de controlador de E/S 1830 pueden estar integrados en un procesador gráfico externo discreto, tal como el procesador gráfico externo 1812.

La Figura 19 es un diagrama de bloques de una realización de un procesador 1900 que tiene uno o más núcleos de procesador 1902A-1902N, un controlador de memoria integrado 1914 y un procesador gráfico integrado 1908. Aquellos elementos de la Figura 19 que tengan los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento, pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella. El procesador 1900 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 1902N representado por los recuadros de línea discontinua. Cada uno de los núcleos de procesador 1902A-1902N incluye una o más unidades de caché interna 1904A-1904N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades de caché compartidas 1906.

Las unidades de caché interna 1904A-1904N y las unidades de caché compartida 1906 representan una jerarquía de memoria caché dentro del procesador 1900. La jerarquía de memoria caché puede incluir al menos un nivel de caché de instrucciones y datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel medio compartido, tal como un Nivel 2 (L2), Nivel 3 (L3), Nivel 4 (L4) u otros niveles de caché, 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 1906 y 1904A-1904N.

En algunas realizaciones, el procesador 1900 también puede incluir un conjunto de una o más unidades de controlador de bus 1916 y un núcleo de agente de sistema 1910. Las una o más unidades de controlador de bus 1916 gestionan un conjunto de buses periféricos, tales como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 1910 proporciona funcionalidad de gestión para los distintos componentes del procesador. En algunas realizaciones, el núcleo de agente de sistema 1910 incluye uno o más controladores de memoria integrados 1914 para gestionar el acceso a varios dispositivos de memoria externa (no mostrados).

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

En algunas realizaciones, el procesador 1900 incluye adicionalmente un procesador gráfico 1908 para ejecutar operaciones de procesamiento gráfico. En algunas realizaciones, el procesador gráfico 1908 se acopla con el conjunto de

unidades de caché compartidas 1906 y el núcleo de agente de sistema 1910, incluidos uno o más controladores de memoria integrados 1914. En algunas realizaciones, un controlador de visualización 1911 está acoplado con el procesador gráfico 1908 para impulsar la salida del procesador gráfico a una o más pantallas acopladas. En algunas realizaciones, el controlador de visualización 1911 puede ser un módulo separado acoplado con el procesador gráfico a través de al menos una interconexión, o puede estar integrado dentro del procesador gráfico 1908 o el núcleo de agente de sistema 1910.

En algunas realizaciones, se utiliza una unidad de interconexión basada en anillo 1912 para acoplar los componentes internos del procesador 1900. Sin embargo, se puede utilizar una unidad de interconexión alternativa, tal como una interconexión punto a punto, una interconexión conmutada u otras técnicas, incluidas técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador gráfico 1908 se acopla con la interconexión en anillo 1912 a través de un enlace de E/S 1913.

El enlace de E/S 1913 ejemplar representa al menos una de múltiples variedades de interconexiones de E/S, incluyendo una interconexión de E/S en paquete que facilita la comunicación entre varios componentes de procesador y un módulo de memoria integrado de alto rendimiento 1918, tal como un módulo eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 1902-1902N y el procesador gráfico 1908 utilizan módulos de memoria integrados 1918 como una caché de último nivel compartida.

En algunas realizaciones, los núcleos de procesador 1902A-1902N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 1902A-1902N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 1902A-N ejecutan un primer conjunto de instrucciones, mientras que al menos uno de los otros núcleos ejecuta un subconjunto del primer conjunto de instrucciones o un conjunto de instrucciones diferente. En una realización, los núcleos de procesador 1902A-1902N son heterogéneos en términos de microarquitectura, donde uno o más núcleos que tienen un consumo de energía relativamente mayor se acoplan con uno o más núcleos de potencia que tienen un consumo de energía menor. Además, el procesador 1900 se puede implementar en uno o más chips o como un circuito integrado SoC que tenga los componentes ilustrados, además de otros componentes.

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

En algunas realizaciones, el procesador gráfico 2000 también incluye un controlador de visualización 2002 para impulsar datos de salida de visualización a un dispositivo de visualización 2020. El controlador de visualización 2002 incluye hardware para uno o más planos de superposición para la visualización y composición de múltiples capas de vídeo o elementos de interfaz de usuario. En algunas realizaciones, el procesador gráfico 2000 incluye un motor de códec de vídeo 2006 para codificar, decodificar o transcodificar medios hacia, desde o entre uno o más formatos de codificación de medios, incluidos, entre otros, los formatos Moving Picture Experts Group (MPEG) como MPEG-2, los formatos Advanced Vídeo Coding (AVC) como H.264/MPEG-4 AVC, así como los formatos Society of Motion Picture & Television Engineers (SMPTE) 421M/VC-1 y Joint Photographic Experts Group (JPEG) como JPEG y Motion JPEG (MJPEG).

En algunas realizaciones, el procesador gráfico 2000 incluye un motor de transferencia de imágenes en bloques (BLIT) 2004 para realizar operaciones de rasterización bidimensionales (2D) que incluyen, por ejemplo, transferencias de bloques de límite de bits. Sin embargo, en una realización, las operaciones gráficas 2D se realizan utilizando uno o más componentes del motor de procesamiento gráfico (GPE) 2010. En algunas realizaciones, el motor de procesamiento gráfico 2010 es un motor de cálculo para realizar operaciones gráficas, incluidas operaciones gráficas tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 2010 incluye una canalización 3D 2012 para realizar operaciones 3D, tales como renderizar imágenes y escenas tridimensionales utilizando funciones de procesamiento que actúan sobre formas primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización 3D 2012 incluye elementos de función fija y programables que realizan varias tareas dentro del elemento y/o generan hilos de ejecución en un subsistema 3D/Medios 2015. Si bien la canalización 3D 2012 se puede utilizar para realizar operaciones de medios, una realización de GPE 2010 también incluye una canalización de medios 2016 que se utiliza específicamente para realizar operaciones de medios, tales como posprocesamiento de vídeo y mejora de imágenes.

En algunas realizaciones, la canalización de medios 2016 incluye unidades lógicas programables o de función fija para realizar una o más operaciones de medios especializadas, tales como aceleración de decodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar de, o en nombre de, el motor de códec de vídeo

2006. En algunas realizaciones, la canalización de medios 2016 incluye además una unidad de generación de hilos para generar hilos para su ejecución en el subsistema 3D/Medios 2015. Los hilos generados realizan cálculos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema 3D/Medios 2015.

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

Procesamiento 3D/Medios

La Figura 21 es un diagrama de bloques de un motor de procesamiento gráfico 2110 de un procesador gráfico de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento gráfico (GPE) 2110 es una versión del GPE 2010 que se muestra en la Figura 20. Los elementos de la Figura 21 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella. Por ejemplo, se ilustran la canalización 3D 2012 y la canalización de medios 2016 de la Figura 20. La canalización de medios 2016 es opcional en algunas realizaciones del GPE 2110 y es posible que no esté incluida explícitamente dentro del GPE 2110. Por ejemplo, y en al menos una realización, un procesador de medios y/o imágenes separado está acoplado al GPE 2110.

En algunas realizaciones, GPE 2110 se acopla con o incluye un transmisor de comandos 2103, que proporciona un flujo de comandos a la canalización 3D 2012 y/o a las canalizaciones de medios 2016. En algunas realizaciones, el transmisor de comandos 2103 está acoplado a una memoria, que puede ser una memoria de sistema o una o más de las siguientes: memoria caché interna y memoria caché compartida. En algunas realizaciones, el transmisor de comandos 2103 recibe comandos de la memoria y envía los comandos a la canalización 3D 2012 y/o a la canalización de medios 2016. Los comandos son directivas obtenidas de una memoria intermedia en anillo, que almacena comandos para la canalización 3D 2012 y la canalización de medios 2016. En una realización, la memoria intermedia en anillo puede incluir además memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización 3D 2012 también pueden incluir referencias a datos almacenados en la memoria, como por ejemplo, entre otros, datos de vértices y geometría para la canalización 3D 2012 y/o datos de imágenes y objetos de memoria para la canalización de medios 2016. La canalización 3D 2012 y la canalización de medios 2016 procesan los comandos y los datos realizando operaciones a través de la lógica dentro de las respectivas canalizaciones o despachando uno o más hilos de ejecución a una matriz de núcleos gráficos 2114.

En varias realizaciones, la canalización 3D 2012 puede ejecutar uno o más programas de sombreador, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculo u otros programas de sombreado, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleos gráficos 2114. La matriz de núcleos gráficos 2114 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 gráficos 2114 incluye soporte para varios 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 gráficos 2114 también incluye lógica de ejecución para realizar funciones de medios, tales como procesamiento de video y/o imágenes. En una realización, las unidades de ejecución incluyen además lógica de propósito general que es programable para realizar operaciones computacionales de propósito general paralelas, además de operaciones de procesamiento gráfico. La lógica de propósito general puede realizar operaciones de procesamiento paralelo o en conjunto con la lógica de propósito general dentro de los núcleos de procesador 1807 de la Figura 18 o los núcleos 1902A-1902N como en la Figura 19.

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

En algunas realizaciones, la matriz de núcleos gráficos 2114 es escalable, de modo que la matriz incluye una cantidad variable de núcleos gráficos, cada uno de los cuales tiene una cantidad variable de unidades de ejecución basadas en el

nivel de potencia y rendimiento objetivo del GPE 2110. En una realización, los recursos de ejecución son escalables dinámicamente, de modo que los recursos de ejecución pueden habilitarse o deshabilitarse según sea necesario.

La matriz de núcleos gráficos 2114 se acopla con la lógica de función compartida 2120 que incluye múltiples recursos que se comparten entre los núcleos gráficos en la matriz de núcleos gráficos. Las funciones compartidas dentro de la lógica de función compartida 2120 son unidades lógicas de hardware que proporcionan una funcionalidad complementaria especializada a la matriz de núcleos gráficos 2114. En diversas realizaciones, la lógica de función compartida 2120 incluye, entre otras cosas, la lógica del muestreador 2121, la lógica matemática 2122 y la lógica de comunicación entre hilos (ITC) 2123. Además, algunas realizaciones implementan una o más memorias caché 2125 dentro de la lógica de función compartida 2120. Una función compartida se implementa cuando la demanda de una función especializada dada es insuficiente para su inclusión en la matriz de núcleos gráficos 2114. En su lugar, se implementa una única instanciación de dicha función especializada como una entidad independiente en la lógica de función compartida 2120 y se comparte entre los recursos de ejecución dentro de la matriz de núcleos gráficos 2114. El conjunto preciso de funciones que se comparten entre la matriz de núcleos gráficos 2114 y se incluyen dentro de la matriz de núcleos gráficos 2114 varía entre las realizaciones.

La Figura 22 es un diagrama de bloques de otra realización de un procesador gráfico 2200. Los elementos de la Figura 22 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella.

En algunas realizaciones, el procesador gráfico 2200 incluye una interconexión en anillo 2202, un frontal de canalización 2204, un motor de medios 2237 y núcleos gráficos 2280A-2280N. En algunas realizaciones, la interconexión en anillo 2202 acopla el procesador gráfico a otras unidades de procesamiento, incluidos otros procesadores gráficos o uno o más núcleos de procesador de propósito general. En algunas realizaciones, el procesador gráfico es uno de los muchos procesadores integrados dentro de un sistema de procesamiento de múltiples núcleos.

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

En algunas realizaciones, el procesador gráfico 2200 incluye recursos de ejecución de hilos escalables que presentan núcleos modulares 2280A-2280N (a veces denominados segmentos de núcleo), cada uno de los cuales tiene múltiples subnúcleos 2250A-2250N, 2260A-2260N (a veces denominados subsegmentos de núcleo). En algunas realizaciones, el procesador gráfico 2200 puede tener cualquier número de núcleos gráficos 2280A-2280N. En algunas realizaciones, el procesador gráfico 2200 incluye un núcleo gráfico 2280A que tiene al menos un primer subnúcleo 2250A y un segundo subnúcleo de núcleo 2260A. En otras realizaciones, el procesador gráfico es un procesador de baja potencia con un único subnúcleo (por ejemplo, 2250A). En algunas realizaciones, el procesador gráfico 2200 incluye múltiples núcleos gráficos 2280A-2280N, incluyendo cada uno un conjunto de primeros subnúcleos 2250A-2250N y un conjunto de segundos subnúcleos 2260A-2260N. Cada subnúcleo del conjunto de primeros subnúcleos 2250A-2250N incluye al menos un primer conjunto de unidades de ejecución 2252A-2252N y muestreadores de medios/textura 2254A-2254N. Cada subnúcleo del conjunto de segundos subnúcleos 2260A-2260N incluye al menos un segundo conjunto de unidades de ejecución 2262A-2262N y muestreadores 2264A-2264N. En algunas realizaciones, cada subnúcleo 2250A-2250N, 2260A-2260N comparte un conjunto de recursos compartidos 2270A-2270N. En algunas realizaciones, los recursos compartidos incluyen memoria caché compartida y lógica de operación de píxeles. También se pueden incluir otros recursos compartidos en las diversas realizaciones del procesador gráfico.

Lógica de ejecución

La Figura 23 ilustra la lógica de ejecución de hilos 2300 que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones de un GPE. Los elementos de la Figura 23 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella.

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

En algunas realizaciones, la matriz de unidades de ejecución 2308A-2308N se utiliza principalmente para ejecutar programas de "sombreadores". En algunas realizaciones, las unidades de ejecución en la matriz 2308A-2308N ejecutan un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreado de gráficos 3D estándar, de modo que los programas de sombreado de bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una conversión mínima. Las unidades de ejecución soportan el procesamiento de vértices y 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 medios y de cálculo).

Cada unidad de ejecución en la matriz de unidades de ejecución 2308A-2308N opera sobre matrices de elementos de datos. La cantidad de elementos de datos es el "tamaño de ejecución" o la cantidad de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso a elementos de datos, el enmascaramiento y el control de flujo dentro de las instrucciones. La cantidad de canales puede ser independiente de la cantidad de unidades lógicas aritméticas (ALU) o unidades de punto flotante (FPU) físicas para un procesador gráfico en particular. En algunas realizaciones, las unidades de ejecución 2308A-2308N admiten tipos de datos de números enteros y de coma flotante.

El conjunto de instrucciones de la unidad de ejecución incluye instrucciones de instrucción única, múltiples datos (SIMD) o instrucciones de instrucción única, múltiples hilos (SIMT). Los diversos elementos de datos se pueden almacenar como un tipo de datos empaquetado en un registro y la unidad de ejecución procesará los diversos elementos en función del 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 separados de 64 bits (elementos de datos de tamaño Quad-Word (QW)), ocho elementos de datos empaquetados separados de 32 bits (elementos de datos de tamaño Double Word (DW)), dieciséis elementos de datos empaquetados separados de 16 bits (elementos de datos de tamaño Word (W)) o treinta y dos elementos de datos separados de 8 bits (elementos de datos de tamaño byte (B)). Sin embargo, son posibles diferentes anchos de vector y tamaños de registro.

Una o más cachés de instrucciones internas (por ejemplo, 2306) se incluyen en la lógica de ejecución de hilos 2300 para almacenar en caché las instrucciones de hilos para las unidades de ejecución. En algunas realizaciones, se incluyen una o más cachés de datos (por ejemplo, 2312) para almacenar en caché los datos de hilo durante la ejecución de hilo. En algunas realizaciones, se incluye un muestreador 2310 para proporcionar un muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunas realizaciones, el muestreador 2310 incluye una funcionalidad especializada de muestreo de textura o medios para procesar datos de textura o 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 medios envían solicitudes de iniciación de hilos a la lógica de ejecución de hilos 2300 a través de la lógica de generación y despacho de hilos. En algunas realizaciones, la lógica de ejecución de hilos 2300 incluye un despachador de hilos local 2304 que arbitra las solicitudes de inicio de hilos de las canalizaciones de gráficos y medios y genera instancias a los hilos solicitados en una o más unidades de ejecución 2308A-2308N. Por ejemplo, la canalización de geometría (por ejemplo, 2236 de la Figura 22) despacha hilos de procesamiento de vértices, teselación o procesamiento de geometría a la lógica de ejecución de hilos 2300 (Figura 23). En algunas realizaciones, el despachador de hilos 2304 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas de sombreadores en ejecución.

Una vez que un grupo de objetos geométricos ha sido procesado y rasterizado en datos de píxeles, se invoca el sombreador de píxeles 2302 para calcular más información de salida y hacer que los resultados se escriban en superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de plantilla, etc.). En algunas realizaciones, el sombreador de píxeles 2302 calcula los valores de los diversos atributos de vértice que se interpolarán en el objeto rasterizado. En algunas realizaciones, el sombreador de píxeles 2302 a continuación ejecuta un programa de sombreador de píxeles suministrado por la interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreado de píxeles, el sombreador de píxeles 2302 despacha hilos a una unidad de ejecución (por ejemplo, 2308A) a través del despachador de hilos 2304. En algunas realizaciones, el sombreador de píxeles 2302 utiliza la lógica de muestreo de textura en el muestreador 2310 para acceder a datos de textura en mapas de

textura almacenados en memoria. Las operaciones aritméticas en los datos de textura y los datos de geometría de entrada calculan datos de color de píxel para cada fragmento geométrico o descartan uno o más píxeles del procesamiento posterior.

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

La Figura 24 es un diagrama de bloques que ilustra los formatos de instrucción de procesador gráfico 2400 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución de procesador gráfico soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Las cajas con líneas continuas ilustran los componentes que generalmente se incluyen 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 2400 descrito e ilustrado son macroinstrucciones, en el sentido de que son instrucciones suministradas a la unidad de ejecución, a diferencia de las microoperaciones que resultan de la decodificación de la instrucción una vez que se procesa la instrucción.

En algunas realizaciones, las unidades de ejecución de procesador gráfico soportan de forma nativa instrucciones en un formato de instrucción de 128 bits 2410. Un formato de instrucción compactado de 64 bits 2430 está disponible para algunas instrucciones según la instrucción seleccionada, las opciones de instrucción y la cantidad de operandos. El formato de instrucción de 128 bits nativo 2410 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de instrucción de 64 bits 2430. Las instrucciones nativas disponibles en el formato de instrucción de 64 bits 2430 varían según la realización. En algunas realizaciones, la instrucción se compacta en parte utilizando un conjunto de valores de índice en un campo de índice 2413. El hardware de la unidad de ejecución hace referencia a un conjunto de tablas de compactación basadas en los valores de índice y utiliza las salidas de la tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 2410.

Para cada formato, el opcode de instrucción 2412 define la operación que la unidad de ejecución debe realizar. Las unidades de ejecución ejecutan cada instrucción en paralelo en los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de adición, la unidad de ejecución realiza una operación de adición simultánea en cada canal de color que representa un elemento de textura o un elemento de imagen. De forma predeterminada, la unidad de ejecución ejecuta cada instrucción en todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 2414 permite el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicción) y el orden de los canales de datos (por ejemplo, swizzle). Para las instrucciones de 128 bits 2410, un campo exec-size 2416 limita la cantidad de canales de datos que se ejecutarán en paralelo. En algunas realizaciones, el campo exec-size 2416 no está disponible para su uso en el formato de instrucción compacta de 64 bits 2430.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos que incluyen dos operandos de origen, src0 2420, src1 2422 y uno de destino 2418. 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 2424), donde el opcode de instrucción 2412 determina la cantidad de operandos de origen. El último operando de origen de una instrucción puede ser un valor inmediato (por ejemplo, codificado) que se pasa con la instrucción.

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

En algunas realizaciones, el formato de instrucción de 128 bits 2410 incluye un campo de modo de acceso/dirección 2426, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En una realización, el modo de acceso define una alineación de acceso a 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 2410 puede usar direccionamiento alineado a bytes para los operandos de origen y destino y cuando está en un segundo modo, la instrucción 2410 puede usar direccionamiento alineado a 16 bytes para todos los operandos de origen y destino.

En una realización, la porción de modo de dirección del campo de modo de acceso/dirección 2426 determina si la instrucción debe utilizar direccionamiento directo o indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, los bits de la instrucción 2410 proporcionan directamente la dirección de registro de uno o más operandos. Cuando

se utiliza el modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos se puede calcular basándose en un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

En algunas realizaciones, las instrucciones se agrupan en función de los campos de bits del opcode 2412 para simplificar la decodificación de opcode 2440. Para un opcode de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de opcode. La agrupación precisa de opcode que se muestra es simplemente un ejemplo. En algunas realizaciones, un grupo de opcode de movimiento y lógica 2442 incluye instrucciones de movimiento de datos y lógica (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y lógica 2442 comparte los cinco bits más significativos (MSB), donde las instrucciones de movimiento (mov) tienen el formato 0000xxxxb y las instrucciones lógicas tienen el formato 0001xxxxb. Un grupo de instrucciones de control de flujo 2444 (por ejemplo, llamada (call), salto (jmp)) incluye instrucciones en la forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 2446 incluye una combinación de instrucciones, incluidas instrucciones de sincronización (por ejemplo, esperar, enviar) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones matemáticas paralelas 2448 incluye instrucciones aritméticas componente por componente (por ejemplo, sumar (add), multiplicar (mul)) en la forma de 0100xxxxb (por ejemplo, 0x40). El grupo de matemáticas paralelas 2448 realiza las operaciones aritméticas en paralelo a través de canales de datos. El grupo de matemáticas vectoriales 2450 incluye instrucciones aritméticas (por ejemplo, dp4) en la forma de 0101xxxxb (por ejemplo, 0x50). El grupo de matemáticas vectoriales realiza operaciones aritméticas como cálculos de producto escalar en operandos vectoriales.

Canalización de gráficos

La Figura 25 es un diagrama de bloques de otra realización de un procesador gráfico 2500. Los elementos de la Figura 25 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella.

En algunas realizaciones, el procesador gráfico 2500 incluye una canalización de gráficos 2520, una canalización de medios 2530, un motor de visualización 2540, una lógica de ejecución de hilos 2550 y una canalización de salida de renderizado 2570. En algunas realizaciones, el procesador gráfico 2500 es un procesador gráfico dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador gráfico se controla mediante escrituras de registros en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador gráfico 2500 a través de una interconexión en anillo 2502. En algunas realizaciones, la interconexión en anillo 2502 acopla el procesador gráfico 2500 a otros componentes de procesamiento, tales como otros procesadores gráficos o procesadores de propósito general. Los comandos de la interconexión en anillo 2502 son interpretados por un transmisor de comandos 2503, que suministra instrucciones a los componentes individuales de la canalización de gráficos 2520 o de la canalización de medios 2530.

En algunas realizaciones, el transmisor de comandos 2503 dirige la operación de un extractor de vértices 2505 que lee datos de vértices de la memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor de comandos 2503. En algunas realizaciones, el extractor de vértices 2505 proporciona datos de vértices a un sombreador de vértices 2507, que realiza operaciones de transformación de espacio de coordenadas e iluminación en cada vértice. En algunas realizaciones, el extractor de vértices 2505 y el sombreador de vértices 2507 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a las unidades de ejecución 2552A, 2552B a través de un despachador de hilos 2531.

En algunas realizaciones, las unidades de ejecución 2552A, 2552B son una matriz de procesadores vectoriales que tienen un conjunto de instrucciones para realizar operaciones gráficas y de medios. En algunas realizaciones, las unidades de ejecución 2552A, 2552B tienen una caché L1 2551 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 2520 incluye componentes de teselación para realizar teselación acelerada por hardware de objetos 3D. En algunas realizaciones, un sombreador de casco programable 2511 configura las operaciones de teselación. Un sombreador de dominio programable 2517 proporciona una evaluación de servidor de la salida de teselación. Un teselador 2513 opera bajo la dirección del sombreador de casco 2511 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados basados en un modelo geométrico grueso que se proporciona como entrada a la canalización de gráficos 2520. En algunas realizaciones, si no se utiliza teselación, se pueden omitir los componentes de teselación 2511, 2513, 2517.

En algunas realizaciones, los objetos geométricos completos pueden ser procesados por un sombreador de geometría 2519 a través de uno o más hilos despachados a las unidades de ejecución 2552A, 2552B, o pueden proceder directamente al recortador 2529. En algunas realizaciones, el sombreador de geometría opera sobre objetos geométricos completos, en lugar de vértices o parches de vértices como en etapas anteriores de la canalización de gráficos. Si la

teselación está deshabilitada, el sombreador de geometría 2519 recibe entrada del sombreador de vértices 2507. En algunas realizaciones, el sombreador de geometría 2519 es programable mediante un programa de sombreador de geometría para realizar teselación de geometría si las unidades de teselación están deshabilitadas.

Antes de la rasterización, un recortador 2529 procesa datos de vértice. El recortador 2529 puede ser un recortador de función fija o un recortador programable que tenga funciones de recorte y sombreador de geometría. En algunas realizaciones, un componente rasterizador y de prueba de profundidad 2573 en la canalización de salida de renderizado 2570 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 2550. En algunas realizaciones, una aplicación puede omitir la rasterización y acceder a datos de vértices no rasterizados a través de una unidad de salida de flujo 2523.

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

En algunas realizaciones, la canalización de salida de renderizado 2570 contiene un componente rasterizador y de prueba de profundidad 2573 que convierte objetos basados en vértices en una representación basada en píxeles asociada. En algunas realizaciones, la canalización de salida de renderizado 2570 incluye una unidad de enmascaramiento/ventana para realizar una rasterización de triángulos y líneas con funciones fijas. En algunas realizaciones también están disponibles una caché de renderizado 2578 y una caché de profundidad 2579 asociadas. Un componente de operaciones de píxeles 2577 realiza operaciones basadas en píxeles sobre los datos, aunque en algunos casos, las operaciones de píxeles asociadas con operaciones 2D (por ejemplo, transferencias de imágenes de bloques de bits con combinación) son realizadas por el motor 2D 2541, o sustituidas en el momento de la visualización por el controlador de visualización 2543 utilizando planos de visualización superpuestos. En algunas realizaciones, una caché L3 compartida 2575 está disponible para todos los componentes gráficos, lo que permite compartir datos sin el uso de la memoria de sistema principal.

En algunas realizaciones, la canalización de medios de procesador gráfico 2530 incluye un motor de medios 2537 y un frontal de vídeo 2534. En algunas realizaciones, el frontal de vídeo 2534 recibe comandos de canalización desde el transmisor de comandos 2503. En algunas realizaciones, la canalización de medios 2530 incluye un transmisor de comandos separado. En algunas realizaciones, el frontal de vídeo 2534 procesa comandos de medios antes de enviar el comando al motor de medios 2537. En algunas realizaciones, el motor de medios 2537 incluye una funcionalidad de generación de hilos para generar hilos para su envío a la lógica de ejecución de hilos 2550 a través del despachador de hilos 2531.

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

En algunas realizaciones, la canalización de gráficos 2520 y la canalización de medios 2530 son configurables para realizar operaciones basadas en múltiples interfaces de programación de gráficos y medios y no son específicas de una interfaz de programación de aplicaciones (API). En algunas realizaciones, el software controlador del procesador gráfico traduce las llamadas API que son específicas de una biblioteca de gráficos o medios en particular en comandos que pueden ser procesados por el procesador gráfico. En algunas realizaciones, se proporciona soporte para Open Graphics Library (OpenGL) y Open Computing Language (OpenCL) del Grupo Khronos, la biblioteca Direct3D de Microsoft Corporation, o se puede proporcionar soporte tanto para OpenGL como para D3D. También se podrá prestar soporte para la biblioteca de Open Soure Computer Vision (OpenCV). También se soportará una API futura con un canalización 3D compatible si se pudiera realizar un mapeo desde la canalización de la API futura a la canalización del procesador gráfico.

Programación de Canalización de Gráficos

La Figura 26A es un diagrama de bloques que ilustra un formato de comando de procesador gráfico 2600 de acuerdo con algunas realizaciones. La Figura 26B es un diagrama de bloques que ilustra una secuencia de comandos de procesador gráfico 2610 de acuerdo con una realización. Los cajas con líneas continuas en la Figura 26A ilustran los componentes

que generalmente se incluyen en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de los comandos de gráficos. El formato de orden de procesador gráfico de ejemplo 2600 de la Figura 26A incluye campos de datos para identificar un cliente objetivo 2602 de la orden, un código de operación de la orden (opcode) 2604, y los datos relevantes 2606 para la orden. También se incluye un subcódigo de operación 2605 y un tamaño de orden 2608 en algunas órdenes.

En algunas realizaciones, el cliente 2602 especifica la unidad cliente del dispositivo gráfico que procesa los datos de comando. En algunas realizaciones, un analizador de comandos de procesador gráfico examina el campo de cliente de cada comando para condicionar el procesamiento posterior del comando y enrutar los datos de comando a la unidad cliente adecuada. En algunas realizaciones, las unidades cliente de procesador gráfico incluyen una unidad de interfaz de memoria, una unidad de renderizado, una unidad 2D, una unidad 3D y una unidad de medios. Cada unidad cliente tiene un canal de procesamiento correspondiente que procesa los comandos. Una vez que la unidad cliente recibe el comando, lee el opcode 2604 y, si está presente, el subopcode 2605 para determinar la operación a realizar. La unidad cliente ejecuta el comando utilizando la información del campo de datos 2606. Para algunos comandos se espera un tamaño de comando 2608 explícito para especificar el tamaño de comando. En algunas realizaciones, el analizador de comandos determina automáticamente el tamaño de al menos algunos de los comandos en función del opcode del comando. En algunas realizaciones, los comandos se alinean mediante múltiplos de una palabra doble.

El diagrama de flujo de la Figura 26B muestra una secuencia de comandos de procesador gráfico 2610 ejemplar. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que presenta una realización de un procesador gráfico utiliza una versión de la secuencia de comandos mostrada para configurar, ejecutar y finalizar un conjunto de operaciones gráficas. Se muestra y describe una secuencia de comandos de muestra solo con fines de ejemplo, ya que las realizaciones no están limitadas a estos comandos específicos ni a esta secuencia de comandos. Además, los comandos pueden emitirse como un lote de comandos en una secuencia de comandos, de modo que el procesador gráfico procesará la secuencia de comandos al menos parcialmente en concurrencia.

En algunas realizaciones, la secuencia de comandos de procesador gráfico 2610 puede comenzar con un comando de vaciado de canalización 2612 para hacer que cualquier canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunas realizaciones, la canalización 3D 2622 y la canalización de medios 2624 no funcionan concurrentemente. El vaciado de canalización se realiza para hacer que la canalización de gráficos activa complete todos los comandos pendientes. En respuesta a un vaciado de canalización, el analizador de comandos del procesador gráfico pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. De manera opcional, cualquier dato en la caché de renderizado que esté marcado como "sucio" se puede vaciar a la memoria. En algunas realizaciones, el comando de vaciado de canalización 2612 se puede utilizar para la sincronización de canalización o antes de colocar el procesador gráfico en un estado de bajo consumo de energía.

En algunas realizaciones, se utiliza un comando de selección de canalización 2613 cuando una secuencia de comandos requiere que el procesador gráfico cambie explícitamente entre canalizaciones. En algunas realizaciones, se requiere un comando de selección de canalización 2613 solo una vez dentro de un contexto de ejecución antes de emitir comandos de canalización a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunas realizaciones, se requiere un comando de vaciado de canalización 2612 inmediatamente antes de un cambio de canalización a través del comando de selección de canalización 2613.

En algunas realizaciones, un comando de control de canalización 2614 configura una canalización de gráficos para su funcionamiento y se utiliza para programar la canalización 3D 2622 y la canalización de medios 2624. En algunas realizaciones, el comando de control de canalización 2614 configura el estado de canalización para la canalización activa. En una realización, el comando de control de canalización 2614 se utiliza para la sincronización de canalización y para borrar datos de una o más memorias caché dentro de la canalización activa antes de procesar un lote de comandos.

En algunas realizaciones, los comandos para el estado de memoria intermedia de retorno 2616 se utilizan para configurar un conjunto de memorias intermedias de retorno para que las respectivas canalizaciones escriban datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas realizaciones, el procesador gráfico también utiliza una o más memorias intermedias de retorno para almacenar datos de salida y realizar comunicación entre hilos. En algunas realizaciones, la configuración del estado de memoria intermedia de retorno 2616 incluye la selección del tamaño y la cantidad de memorias intermedias de retorno a utilizar para un conjunto de operaciones de canalización.

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

Los comandos para el estado de canalización 3D 2630 incluyen comandos de configuración de estado 3D para el estado de memoria intermedia de vértice, el estado de elemento de vértice, el estado de color constante, el estado de memoria intermedia de profundidad y otras variables de estado que se deben configurar antes de que se procesen los comandos de primitivas 3D. Los valores de estos comandos se determinan, al menos en parte, basándose en la API 3D particular en utilización. En algunas realizaciones, los comandos de estado de canalización 3D 2630 también pueden inhabilitar o eludir selectivamente ciertos elementos de la canalización si esos elementos no se utilizarán.

En algunas realizaciones, el comando de primitiva 3D 2632 se utiliza para enviar primitivas 3D para que sean procesadas por la canalización 3D. Los comandos y los parámetros asociados que se pasan al procesador gráfico a través del comando de primitiva 3D 2632 se reenvían a la función de obtención de vértices en la canalización de gráficos. La función de obtención de vértices utiliza los datos de comando de primitiva 3D 2632 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 primitiva 3D 2632 se utiliza para realizar operaciones de vértice en primitivas 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la canalización 3D 2622 despacha hilos de ejecución de sombreadores a unidades de ejecución de procesadores gráficos.

En algunas realizaciones, la canalización 3D 2622 se activa a través de un comando o evento de ejecución 2634. En algunas realizaciones, una escritura de registro desencadena la ejecución de comando. En algunas realizaciones, la ejecución se activa a través de un comando "go" o "kick" en la secuencia de comandos. En una realización, la ejecución de comando se activa utilizando un comando de sincronización de canalización para vaciar la secuencia de comandos a través de la canalización de gráficos. La canalización 3D realizará el procesamiento de geometría para las primitivas 3D. Una vez completadas las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles colorea los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones de servidor de píxeles para esas operaciones.

En algunas realizaciones, la secuencia de comandos de procesador gráfico 2610 sigue la ruta de la canalización de medios 2624 cuando realiza operaciones de medios. En general, el uso específico y la manera de programación de la canalización de medios 2624 dependen de los medios o las operaciones de cálculo que se realizarán. Es posible que se descarguen operaciones de decodificación de medios específicas en la canalización de medios durante la decodificación de medios. En algunas realizaciones, la canalización de medios también se puede omitir y la decodificación de medios se puede realizar total o parcialmente utilizando 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 gráfico de propósito general (GPGPU), donde el procesador gráfico se utiliza para realizar operaciones de vector SIMD utilizando programas de sombreador computacionales que no están relacionados explícitamente con la representación de primitivas gráficas.

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

En algunas realizaciones, los comandos de objeto de medios 2642 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias que contienen datos de vídeo para ser procesados. En algunas realizaciones, todos los estados de canalización de medios deben ser válidos antes de emitir un comando de objeto de medios 2642. Una vez que se configura el estado de canalización y se ponen en cola los comandos de objeto de medios 2642, la canalización de medios 2624 se activa a través de un comando de ejecución 2644 o un evento de ejecución equivalente (por ejemplo, escritura de registro). La salida del canal de medios 2624 puede luego posprocesarse mediante operaciones proporcionadas por el canal 3D 2622 o el canal de medios 2624. En algunas realizaciones, las operaciones GPGPU se configuran y ejecutan de manera similar a las operaciones de medios.

Arquitectura de software de gráficos

La Figura 27 ilustra una arquitectura de software de gráficos ejemplar para un sistema de procesamiento de datos 2700 de acuerdo con algunas realizaciones. En algunas realizaciones, la arquitectura de software incluye una aplicación de gráficos 3D 2710, un sistema operativo 2720 y al menos un procesador 2730. En algunas realizaciones, el procesador 2730 incluye un procesador gráfico 2732 y uno o más núcleos de procesador de propósito general 2734. La aplicación de gráficos 2710 y el sistema operativo 2720 se ejecutan cada uno en la memoria de sistema 2750 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación de gráficos 3D 2710 contiene uno o más programas de sombreador que incluyen instrucciones de sombreador 2712. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el lenguaje de sombreador de alto nivel (HLSL) o el lenguaje de sombreador OpenGL (GLSL). La aplicación también incluye instrucciones ejecutables 2714 en un lenguaje máquina adecuado para ser ejecutado por el o los núcleos de procesador de propósito general 2734. La aplicación también incluye objetos gráficos 2716 definidos por datos de vértice.

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

En algunas realizaciones, el controlador gráfico en modo de usuario 2726 contiene un compilador de sombreador de servidor 2727 para convertir las instrucciones de sombreador 2712 en una representación específica de hardware. Cuando se utiliza la API OpenGL, las instrucciones de sombreador 2712 en el lenguaje de alto nivel GLSL se pasan a un controlador gráfico en modo de usuario 2726 para su compilación. En algunas realizaciones, el controlador gráfico en modo de usuario 2726 utiliza funciones de modo núcleo de sistema operativo 2728 para comunicarse con un controlador gráfico en modo núcleo 2729. En algunas realizaciones, el controlador gráfico en modo núcleo 2729 se comunica con el procesador gráfico 2732 para despachar comandos e instrucciones.

Implementaciones de IP Core

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 diversa lógica dentro del procesador. Cuando las lee una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en el presente documento. Estas representaciones, conocidas como "núcleos IP", son unidades lógicas reutilizables para un circuito integrado que pueden almacenarse en un medio tangible y legible por máquina 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 realice las operaciones descritas en asociación con cualquiera de las realizaciones descritas en el presente documento.

La Figura 28 es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo IP 2800 que puede usarse para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo IP 2800 se puede utilizar para generar diseños modulares y reutilizables que se pueden incorporar en un diseño más grande o utilizar para construir un circuito integrado completo (por ejemplo, un circuito integrado SOC). Una instalación de diseño 2830 puede generar una simulación de software 2810 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 2810 se puede utilizar para diseñar, probar y verificar el comportamiento del núcleo IP utilizando un modelo de simulación 2812. El modelo de simulación 2812 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Luego se puede crear o sintetizar un diseño de nivel de transferencia de registro (RTL) 2815 a partir del modelo de simulación 2812. El diseño RTL 2815 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, incluida la lógica asociada realizada utilizando las señales digitales modeladas. Además de un diseño RTL 2815, también se pueden crear, diseñar o sintetizar diseños de nivel inferior a nivel lógico o a nivel de transistor. Por lo tanto, los detalles particulares del diseño inicial y la simulación pueden variar.

El diseño RTL 2815 o equivalente puede ser sintetizado además por la instalación de diseño en un modelo de hardware 2820, que puede estar en un lenguaje de descripción de hardware (HDL), o alguna otra representación de datos de diseño físico. El HDL se puede simular o probar más a fondo para verificar el diseño de núcleo IP. El diseño de núcleo de IP se puede almacenar para su entrega a una instalación de fabricación de 3rd 2865 usando la memoria no volátil 2840 (por ejemplo, disco duro, memoria flash o cualquier medio de almacenamiento no volátil). Alternativamente, el diseño de núcleo IP puede transmitirse (por ejemplo, a través de Internet) a través de una conexión cableada 2850 o una conexión inalámbrica 2860. La instalación de fabricación 2865 puede entonces fabricar un circuito integrado que esté basado al menos en parte en el diseño de núcleo IP. El circuito integrado fabricado se puede configurar para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Ejemplo de Circuito Integrado de Sistema en Chip

Las Figuras 29-31 ilustran circuitos integrados ejemplares y procesadores gráficos asociados que pueden fabricarse utilizando uno o más núcleos IP, de acuerdo con diversas realizaciones descritas en el presente documento. Además de lo ilustrado, se pueden incluir otros circuitos y lógica, incluidos procesadores/núcleos gráficos adicionales, controladores de interfaz periférica o núcleos de procesador de propósito general.

La Figura 29 es un diagrama de bloques que ilustra un sistema ejemplar en un circuito integrado en chip 2900 que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con una realización. El circuito integrado ejemplar 2900 incluye uno o más procesadores de aplicaciones 2905 (por ejemplo, CPU), al menos un procesador gráfico 2910, y puede incluir adicionalmente un procesador de imágenes 2915 y/o un procesador de vídeo 2920, cualquiera de los cuales puede ser un núcleo IP modular de la misma o múltiples instalaciones de diseño diferentes. El circuito integrado 2900 incluye una lógica de bus o de periféricos que incluye un controlador de USB 2925, un controlador de UART 2930, un controlador de SPI/SDIO 2935 y un controlador de I^2S/I^2C 2940. Además, el circuito integrado puede incluir un dispositivo de visualización 2945 acoplado a uno o más de un controlador de interfaz de medios de alta definición (HDMI) 2950 y una interfaz de visualización de interfaz de procesador de la industria móvil (MIPI) 2955. El almacenamiento puede ser proporcionado por un subsistema de memoria flash 2960 que incluye memoria flash y un controlador de memoria flash. Se puede proporcionar una interfaz de memoria a través de un controlador de memoria 2965 para acceder a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 2970.

La Figura 30 es un diagrama de bloques que ilustra un procesador gráfico 3010 ejemplar de un circuito integrado de sistema en chip que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con una realización. El procesador gráfico 3010 puede ser una variante del procesador gráfico 2910 de la Figura 29. El procesador gráfico 3010 incluye un procesador de vértices 3005 y uno o más procesadores de fragmentos 3015A-3015N (por ejemplo, 3015A, 3015B, 3015C, 3015D, hasta 3015N-1 y 3015N). El procesador gráfico 3010 puede ejecutar diferentes programas de sombreado a través de una lógica separada, de modo que el procesador de vértices 3005 está optimizado para ejecutar operaciones para programas de sombreado de vértices, mientras que el uno o más procesadores de fragmentos 3015A-3015N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreado de fragmentos o píxeles. El procesador de vértices 3005 realiza la etapa de procesamiento de vértices de la canalización de gráficos 3D y genera primitivas y datos de vértices. El procesador o procesadores de fragmentos 3015A-3015N usan los datos de primitiva y de vértice generados por el procesador de vértices 3005 para producir una memoria intermedia de fotogramas que se visualiza en un dispositivo de visualización. En una realización, el procesador o procesadores de fragmentos 3015A-3015N están optimizados para ejecutar programas de sombreado de fragmentos según se proporciona en la API de OpenGL, que pueden usarse para realizar operaciones similares como un programa de sombreado de píxeles como se proporciona en la API de Direct 3D.

El procesador gráfico 3010 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 3020A-3020B, cachés 3025A-3025B e interconexiones de circuitos 3030A-3030B. La una o más MMU 3020A-3020B proporcionan un mapeo de direcciones virtuales a físicas para el procesador gráfico 3010, incluyendo el procesador de vértices 3005 y/o los procesadores de fragmentos 3015A-3015N, que pueden hacer referencia a datos de vértices o de imagen/textura almacenados en memoria, además de los datos de vértices o de imagen/textura almacenados en la una o más cachés 3025A-3025B. En una realización, la una o más MMU 3020A-3020B pueden estar sincronizadas con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con el uno o más procesadores de aplicaciones 2905, procesador de imágenes 2915 y/o procesador de vídeo 2920 de la Figura 29, de modo que cada procesador 2905-2920 pueda participar en un sistema de memoria virtual compartido o unificado. El uno o más interconectores de circuitos 3030A-3030B permiten que el procesador gráfico 3010 interactúe con otros núcleos IP dentro del SoC, ya sea a través de un bus interno del SoC o a través de una conexión directa, de acuerdo con las realizaciones.

La Figura 31 es un diagrama de bloques que ilustra un procesador gráfico 3110 ejemplar adicional de un circuito integrado de sistema en chip que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con una realización. El procesador gráfico 3110 puede ser una variante del procesador gráfico 2910 de la Figura 29. El procesador gráfico 3110 incluye una o más MMU 3020A-3020B, cachés 3025A-3025B e interconexiones de circuito 3030A-3030B del circuito integrado 3000 de la Figura 30.

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

Las referencias a "una sola realización", "una realización", "una realización de ejemplo", "varias realizaciones", etc., indican que la o las realizaciones así descritas pueden incluir características, estructuras o rasgos particulares, pero no todas las realizaciones incluyen necesariamente las características, estructuras o rasgos particulares. Además, algunas realizaciones pueden tener algunas, todas o ninguna de las características descritas para otras realizaciones.

En la memoria descriptiva anterior, se han descrito realizaciones con referencia a realizaciones ejemplares específicas de las mismas. Será evidente, sin embargo, que podrán introducirse diversas modificaciones y cambios. Por consiguiente, la memoria descriptiva y los dibujos deben considerarse en sentido ilustrativo y no restrictivo.

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

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

Las siguientes cláusulas y/o ejemplos pertenecen a realizaciones o ejemplos adicionales. Los detalles de los ejemplos se pueden utilizar en cualquier parte de una o más realizaciones. Las diversas características de las diferentes realizaciones o ejemplos se pueden combinar de diversas formas, incluyéndose algunas características y excluyéndose otras para adaptarse a una variedad de aplicaciones diferentes. Los ejemplos pueden incluir materia tal como un método, medios para realizar actos del método, al menos un medio legible por máquina que incluye instrucciones que, cuando son ejecutadas por una máquina, hacen que la máquina realice actos del método, o de un aparato o sistema para facilitar la comunicación híbrida de acuerdo con las realizaciones y los ejemplos descritos en el presente documento.

REIVINDICACIONES

1. Un aparato para facilitar la sincronización para el aprendizaje automático en máquinas autónomas, el aparato que comprende:

uno o más dispositivos de procesamiento que incluyen un procesador gráfico que comprende un planificador de hilos y múltiples multiprocesadores de transmisión, SM, teniendo cada SM una pluralidad de elementos de cálculo y una memoria local compartida, SLM;

múltiples matrices, en donde cada matriz representa un dispositivo de procesamiento de uno o más dispositivos de procesamiento;

lógica de detección configurada para detectar grupos de hilos relacionados con el aprendizaje automático asociado con uno o más dispositivos de procesamiento;

lógica de barrera configurada para controlar la sincronización de barrera de los grupos de hilos a través de las múltiples matrices de tal manera que cada hilo en un grupo de hilos está planificado a través de un conjunto de elementos de cálculo asociados con las múltiples matrices; y

lógica de anticipación y programación configurada para planificar un grupo de hilos en múltiples multiprocesadores de transmisión, SM, si el grupo de hilos no utiliza una barrera o una memoria local compartida, SLM, y

si el grupo de hilos utiliza una cantidad de espacio SLM que no se considera crítica para el rendimiento del sistema, la lógica de anticipación y programación se configura para facilitar que el planificador de hilos mapee dicho espacio SLM a una memoria global y posteriormente planifique dicho grupo de hilos en los múltiples SM.

2. El aparato de la reivindicación 1, que comprende además una lógica de sincronización configurada para facilitar la sincronización de los grupos de hilos utilizando un comando de barrera basado en identificaciones de grupos de hilos, ID, correspondientes a los grupos de hilos, en donde el comando de barrera se ejecuta para sincronizar los grupos de hilos tal como se identifican por las ID de grupos de hilos insertadas en el comando de barrera como parte de un argumento, en donde los grupos de hilos se sincronizan a través de los múltiples multiprocesadores de transmisión, SM, y a través de los múltiples procesadores gráficos utilizando una o más barras transversales de estructura.

3. El aparato de la reivindicación 2, en donde la lógica de sincronización está configurada además para facilitar el compartición de una red neuronal intermedia entre una pluralidad de máquinas autónomas a través de una base de datos sobre una red de comunicación, en donde los datos de la red neuronal intermedia se almacenan en la base de datos por una o más de la pluralidad de máquinas autónomas y se comprimen para una transmisión más rápida, en donde la base de datos incluye una base de datos en la nube y la red de comunicación incluye una red en la nube.

4. El aparato de la reivindicación 1, en donde la lógica de anticipación y programación está configurada además para anticipar el grupo de hilos mientras permite que otros grupos de hilos continúen para evitar que el grupo de hilos entre en un estado de bloqueo, en donde esta anticipación se basa en una lista de grupos de hilos y un estado mantenidos por un planificador de hilos, en donde la lista de grupos de hilos indica un número de grupos de hilos que se permite anticipar durante un período de tiempo, en donde el procesador gráfico está ubicado junto con un procesador de aplicaciones en un paquete de semiconductores común.

5. Un método implementado por computadora para facilitar la sincronización para el aprendizaje automático en máquinas autónomas, el método que comprende:

detectar grupos de hilos relacionados con el aprendizaje automático asociado con uno o más dispositivos de procesamiento;

sincronización de barrera de control de los grupos de hilos a través de múltiples matrices de tal manera que cada hilo en un grupo de hilos está planificado a través de un conjunto de elementos de cálculo asociados con las múltiples matrices, en donde cada matriz representa un dispositivo de procesamiento de uno o más dispositivos de procesamiento, incluyendo el dispositivo de procesamiento un procesador gráfico; y

planificar un grupo de hilos en varios multiprocesadores de transmisión, SM, si el grupo de hilos no utiliza una barrera o una memoria local compartida, SLM, y

si el grupo de hilos utiliza una cantidad de espacio SLM que no se considera crítica para el rendimiento del sistema, lo que facilita que el planificador de hilos asigne dicho espacio SLM a una memoria global y posteriormente planifique dicho grupo de hilos en los múltiples SM.

6. El método de la reivindicación 5, que comprende además facilitar la sincronización de los grupos de hilos utilizando un comando de barrera basado en identificaciones de grupos de hilos, ID, correspondientes a los grupos de hilos, en donde el comando de barrera se ejecuta para sincronizar los grupos de hilos tal como se identifican por las ID de grupos de hilos insertadas en el comando de barrera como parte de un argumento, en donde los grupos de hilos se sincronizan a través de los múltiples multiprocesadores de transmisión, SM, y a través de los múltiples procesadores gráficos utilizando una o más barras transversales de estructura.

7. El método de la reivindicación 6, que comprende además facilitar la compartición de una red neuronal intermedia entre una pluralidad de máquinas autónomas a través de una base de datos sobre una red de comunicación, en donde los datos de la red neuronal intermedia se almacenan en la base de datos por una o más de la pluralidad de máquinas autónomas y se comprimen para una transmisión más rápida, en donde la base de datos incluye una base de datos en la nube y la red de comunicación incluye una red en la nube.

8. El método de la reivindicación 5, que comprende además anticipar el grupo de hilos mientras se permite que otros grupos de hilos continúen para evitar que el grupo de hilos entre en un estado de bloqueo, en donde esta anticipación se basa en una lista de grupos de hilos y un estado mantenidos por un planificador de hilos, en donde la lista de grupos de hilos indica un número de grupos de hilos que se permite anticipar durante un período de tiempo, en donde el procesador gráfico está ubicado junto con un procesador de aplicaciones en un paquete de semiconductores común.

9. Al menos un medio legible por máquina que comprende una pluralidad de instrucciones, que cuando se ejecutan en el aparato de la reivindicación 1, hacen que el aparato lleve a cabo un método como se reivindica en cualquiera de las reivindicaciones 5-8.

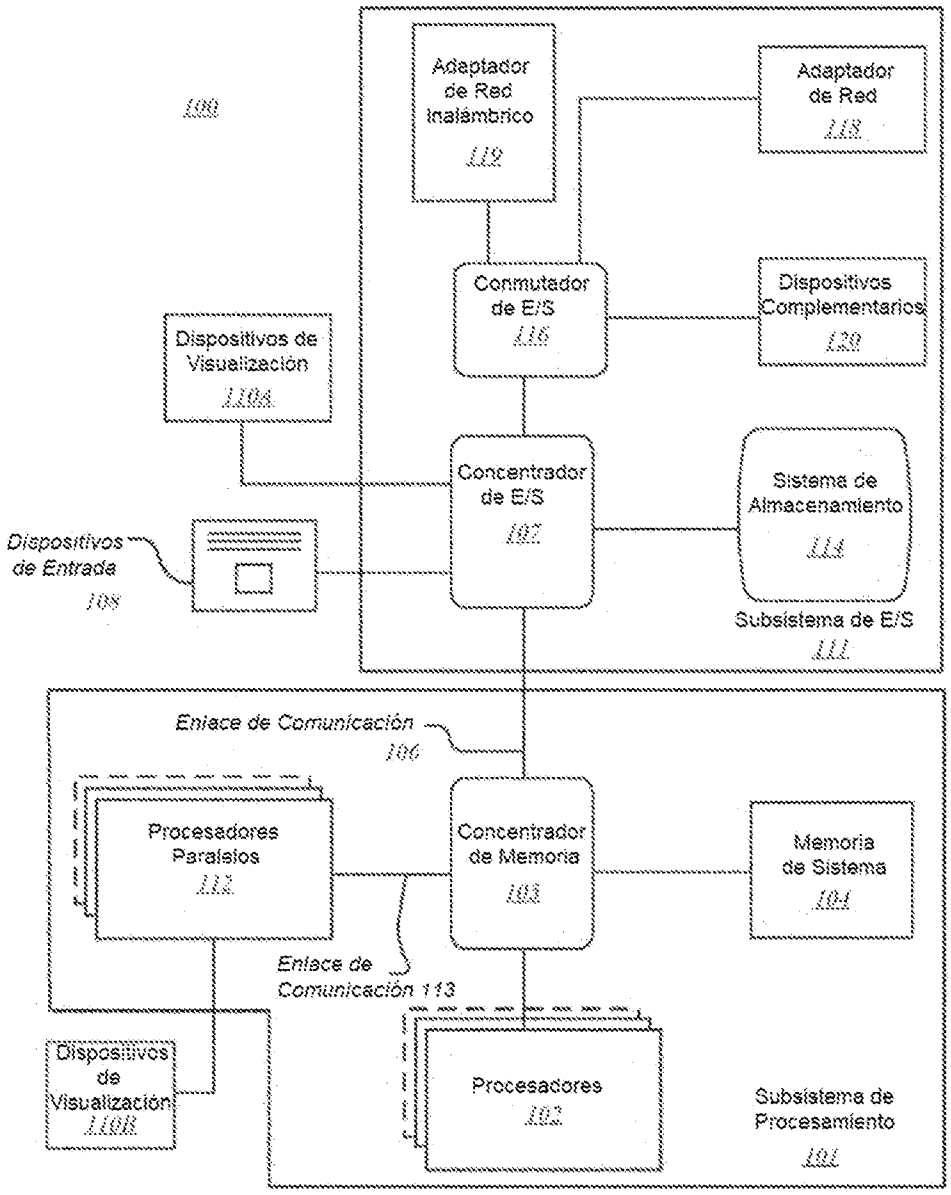

100
Adaptador de Red Inalámbrico 112
Adaptador de Red 118
Conmutador de E/S 116
Dispositivos Complementarios 120
Dispositivos de Visualización 110A
Concentrador de E/S 102
Sistema de Almacenamiento 114
Dispositivos de Entrada 108
Subsistema de E/S 111
Enlace de Comunicación 106
Procesadores Paralelos 112
Concentrador de Memoria 103
Memoria de Sistema 104
Enlace de Comunicación 113
Dispositivos de Visualización 110B
Procesadores 107
Subsistema de Procesamiento 101

FIG. 1

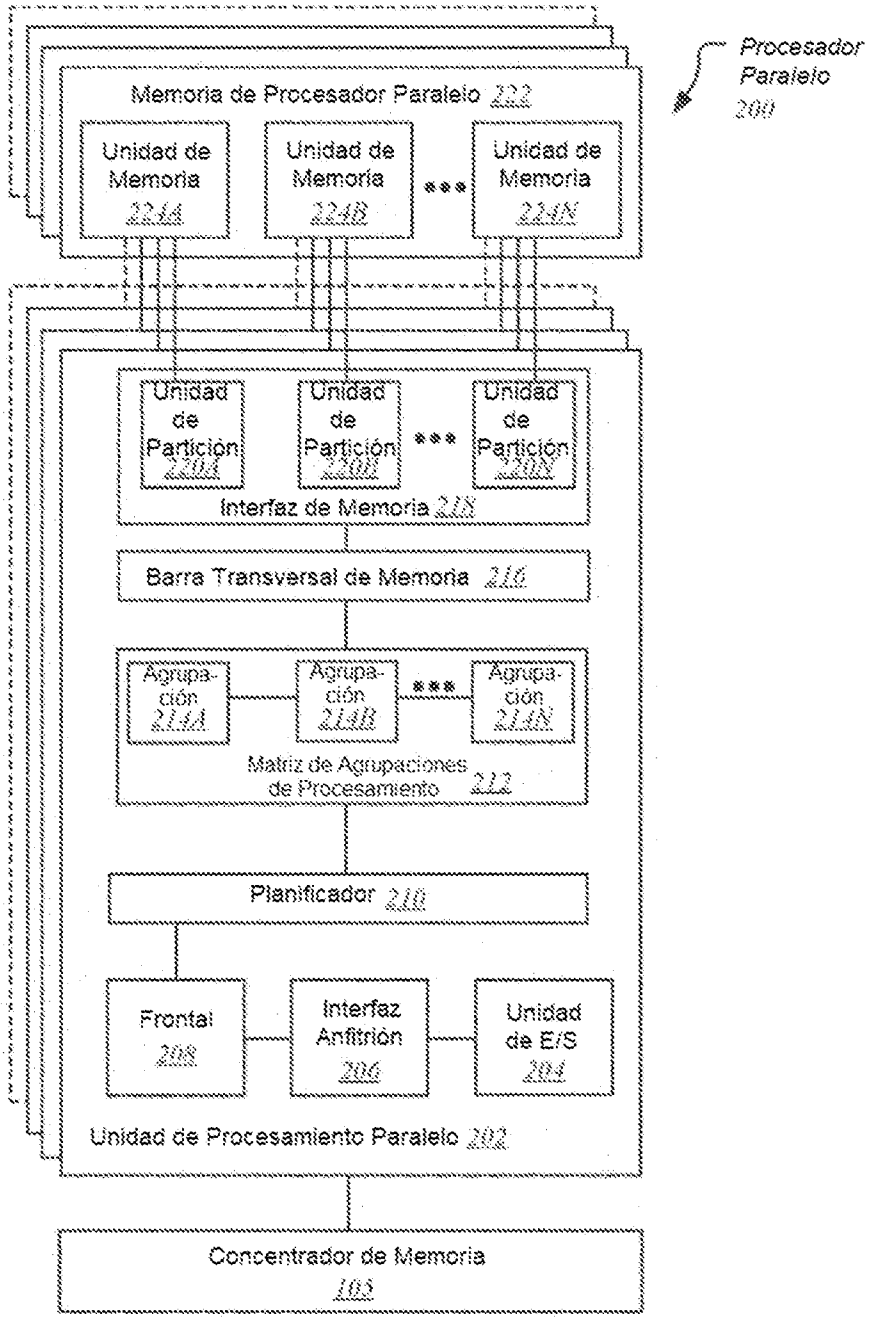

Procesador Paralelo 200
Memoria de Procesador Paralelo 222
Unidad de Memoria 224A
Unidad de Memoria 224B
Unidad de Memoria 224N
Unidad de Partición 220A
Unidad de Partición 220B
Unidad de Partición 220N
Interfaz de Memoria 218
Barra Transversal de Memoria 216
Agrupación 214A
Agrupación 214B
Agrupación 214N
Matriz de Agrupaciones de Procesamiento 212
Planificador 210
Frontal 208
Interfaz Anfitrión 206
Unidad de E/S 204
Unidad de Procesamiento Paralelo 202
Concentrador de Memoria 103

FIG. 2A

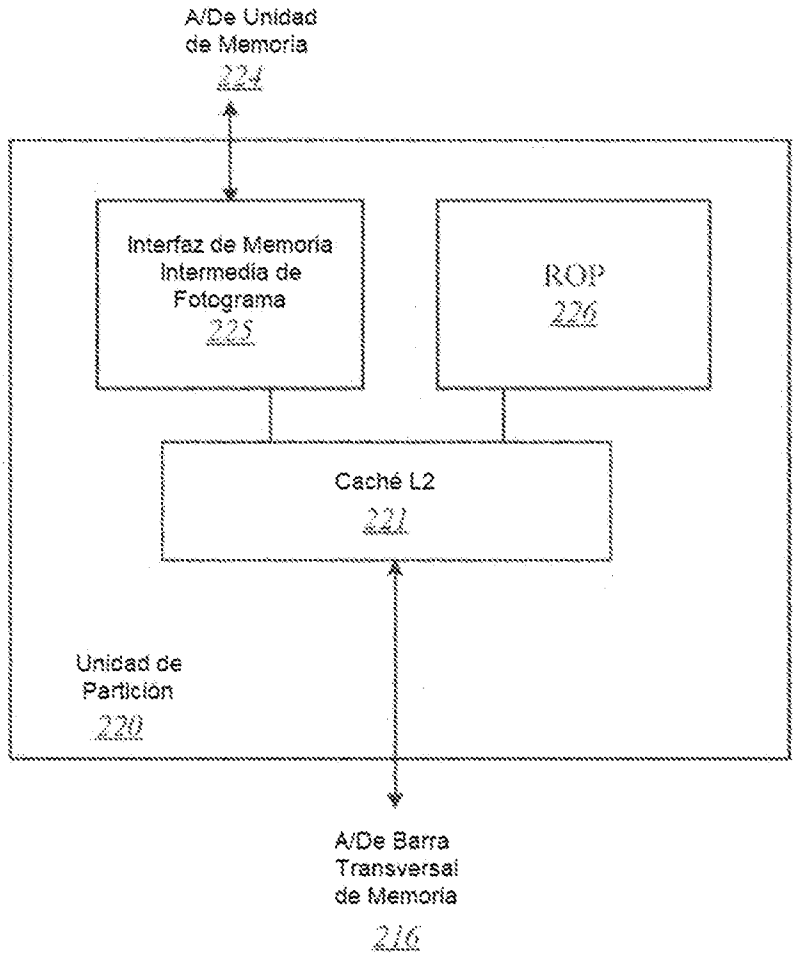

A/De Unidad de Memoria
224
Interfaz de Memoria Intermedia de Fotograma
225
ROP
226
Caché L2
221
Unidad de Partición
220
A/De Barra Transversal de Memoria
216

FIG. 2B

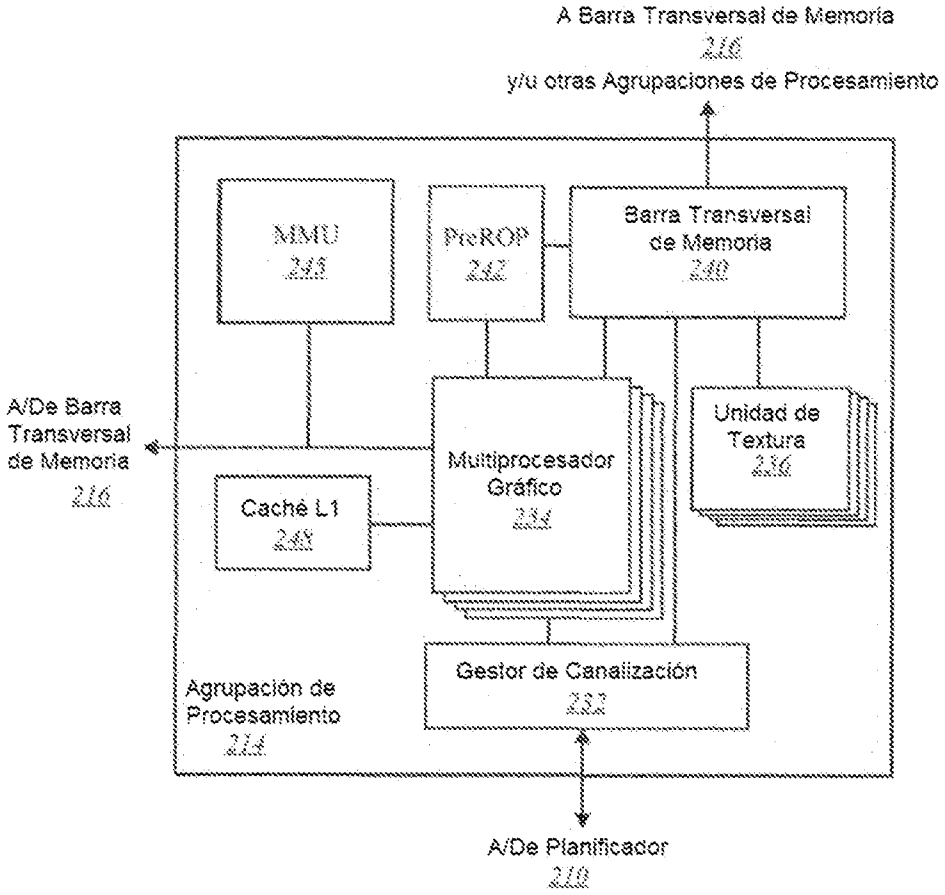

A Barra Transversal de Memoria
216
y/u otras Agrupaciones de Procesamiento
MMU
218
PreROP
242
Barra Transversal de Memoria
240
A/De Barra Transversal de Memoria
216
Multiprocesador Gráfico
234
Unidad de Textura
236
Caché L1
238
Gestor de Canalización
232
Agrupación de Procesamiento
214
A/De Planificador
210

FIG. 2C

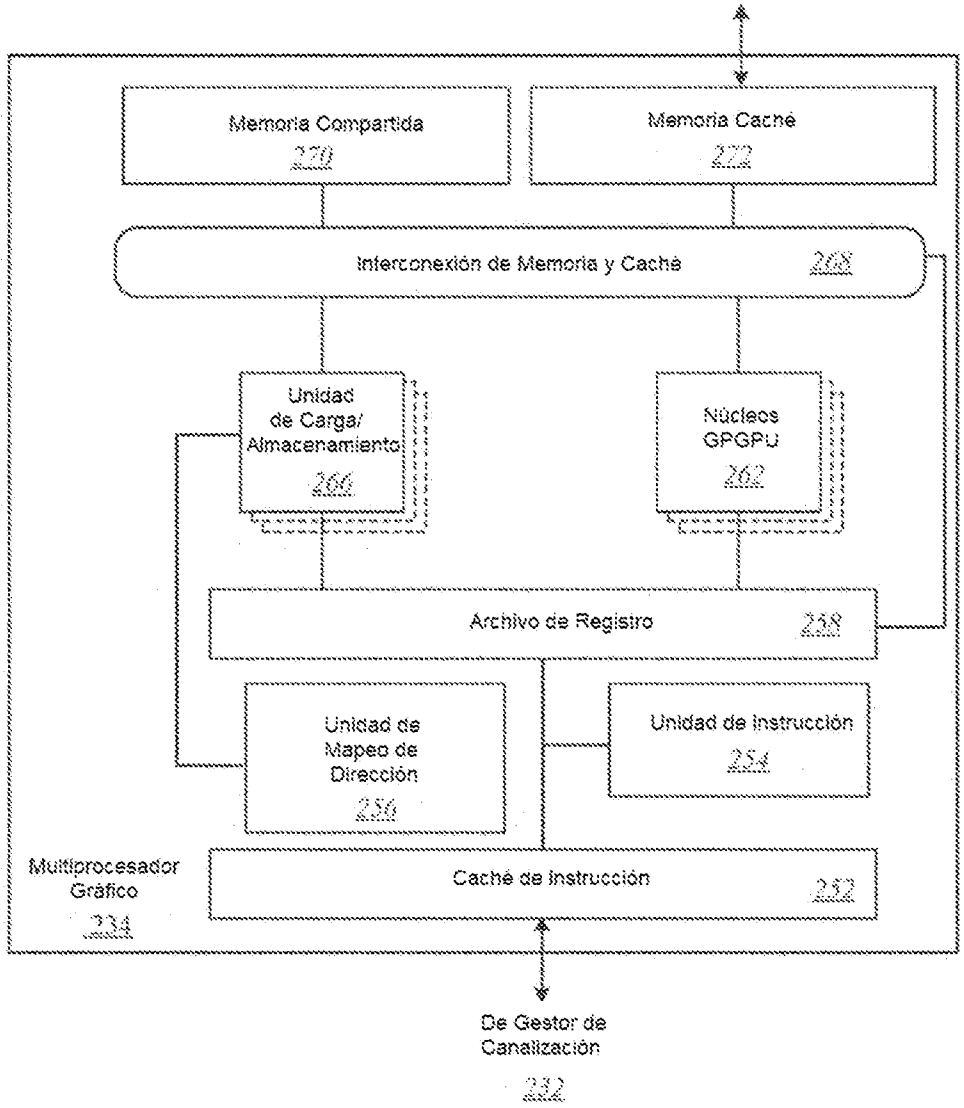

Memoria Compartida
270
Memoria Caché
272
Interconexión de Memoria y Caché
268
Unidad de Carga/ Almacenamiento
266
Núcleos GPGPU
262
Archivo de Registro
258
Unidad de Mapeo de Dirección
256
Unidad de Instrucción
254
Multiprocesador Gráfico
234
Caché de Instrucción
252
De Gestor de Canalización
232

FIG. 2D

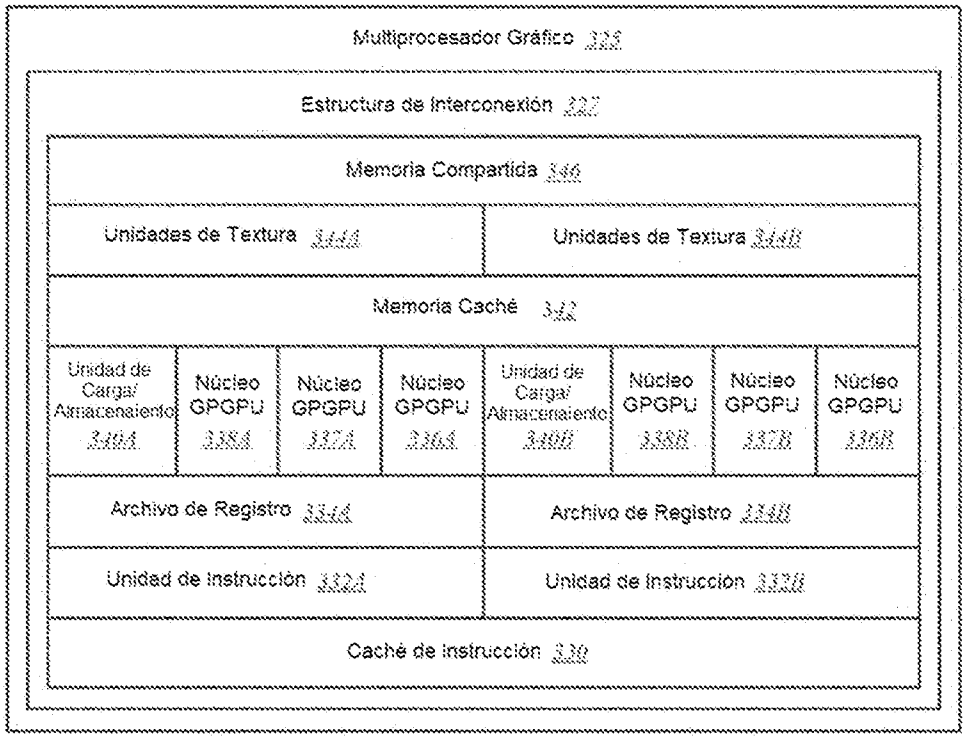

Multiprocesador Gráfico 325
Estructura de Interconexión 327
Memoria Compartida 346
Unidades de Textura 344A
Unidades de Textura 344B
Memoria Caché 342
Unidad de Carga/ Almacenaiento 340A
Núcleo GPGPU 335A
Núcleo GPGPU 337A
Núcleo GPGPU 339A
Unidad de Carga/ Almacenaiento 340B
Núcleo GPGPU 335B
Núcleo GPGPU 337B
Núcleo GPGPU 339B
Archivo de Registro 334A
Archivo de Registro 334B
Unidad de Instrucción 332A
Unidad de Instrucción 332B
Caché de Instrucción 330

FIG. 3A

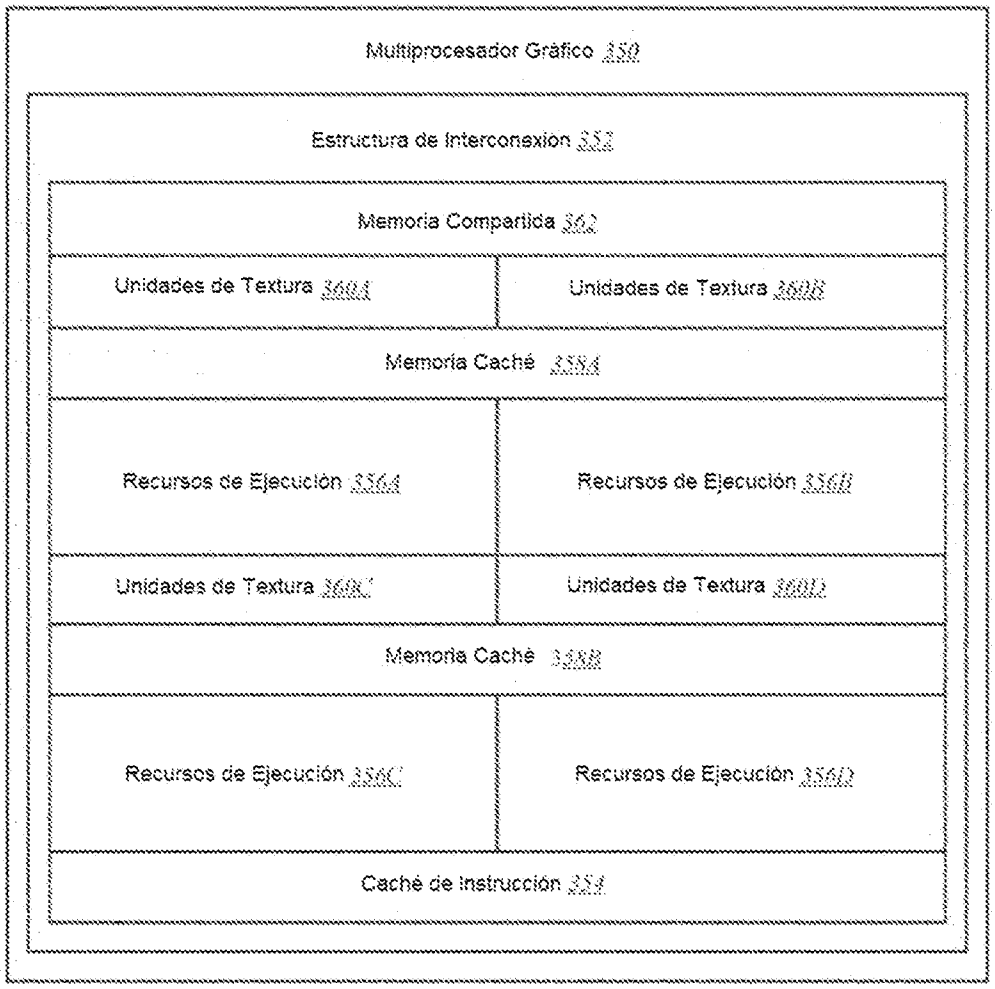

Multiprocesador Gráfico 350
Estructura de Interconexión 352
Memoria Compartida 362
Unidades de Textura 360A
Unidades de Textura 360B
Memoria Caché 358A
Recursos de Ejecución 356A
Recursos de Ejecución 356B
Unidades de Textura 360C
Unidades de Textura 360D
Memoria Caché 358B
Recursos de Ejecución 356C
Recursos de Ejecución 356D
Caché de Instrucción 354

FIG. 3B

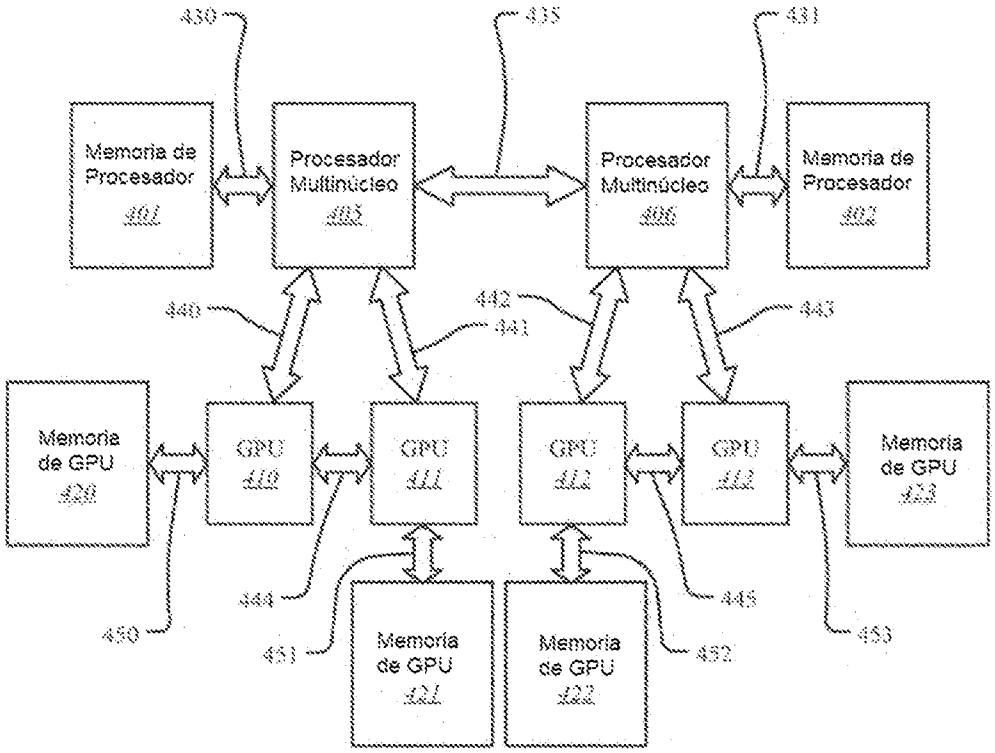

430
435
431
Memoria de Procesador 401
Procesador Multinúcleo 405
Procesador Multinúcleo 406
Memoria de Procesador 402
440
442
441
443
Memoria de GPU 420
GPU 410
GPU 411
GPU 412
GPU 413
Memoria de GPU 423
444
445
450
451
452
453
Memoria de GPU 421
Memoria de GPU 422

FIG. 4A

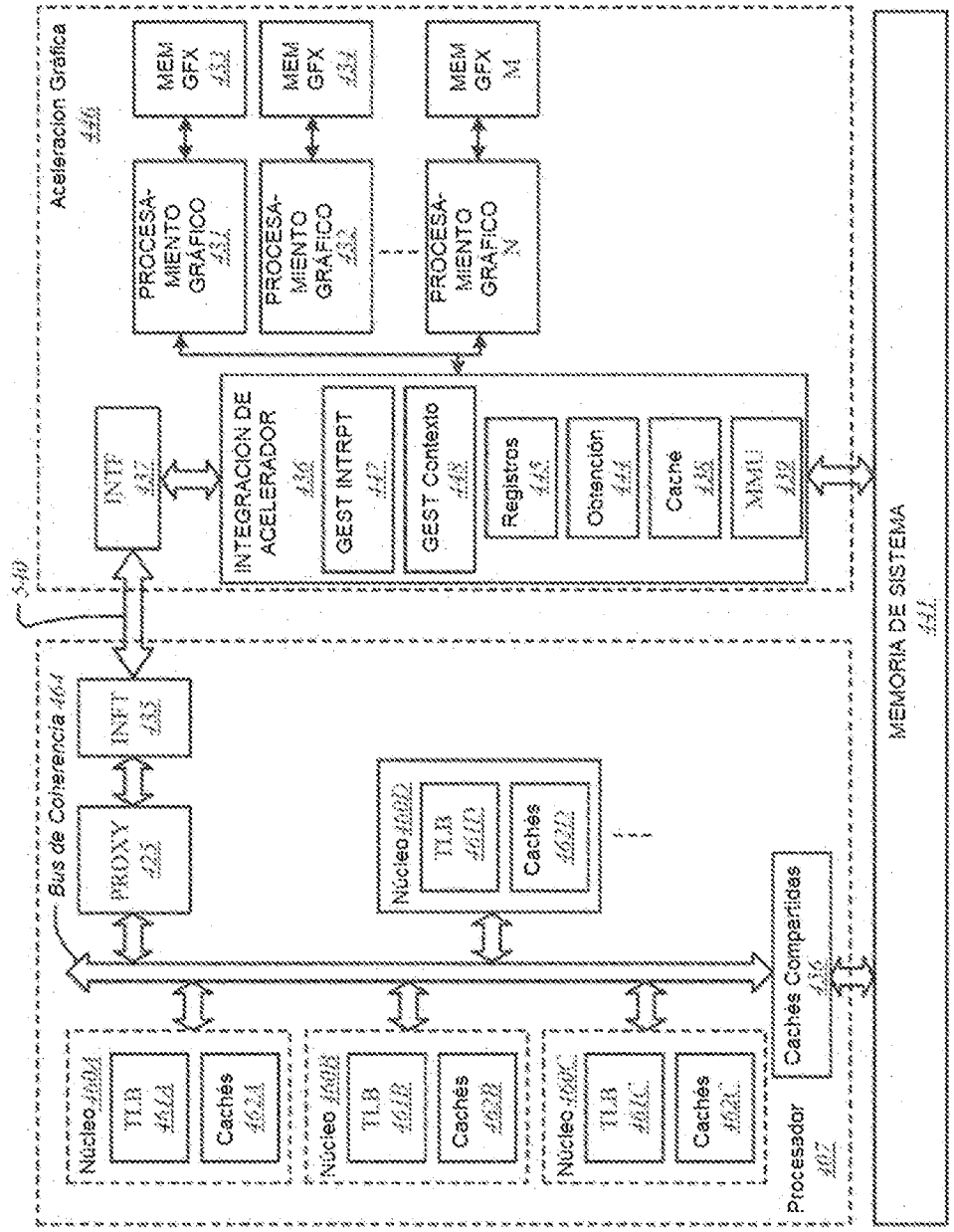

Aceleracion Gráfica
446
MEM GFX 442
MEM GFX 443
MEM GFX M
PROCESA-MIENTO GRÁFICO 441
PROCESA-MIENTO GRÁFICO 442
PROCESA-MIENTO GRÁFICO N
INTEGRACION DE ACELERADOR
446
GEST INTRPT 447
GEST Contexto 448
Registros 445
Obtención 444
Cache 443
MMU 442
INTF 447
449
Bus de Coherencia 464
INFT 435
PROXY 425
Núcleo 460D
TLB 461D
Cachés 462D
Núcleo 460A
TLB 461A
Cachés 462A
Núcleo 460B
TLB 461B
Cachés 462B
Núcleo 460C
TLB 461C
Cachés 462C
Procesador 467
Cachés Compartidas 465
MEMORIA DE SISTEMA
441

FIG. 4B

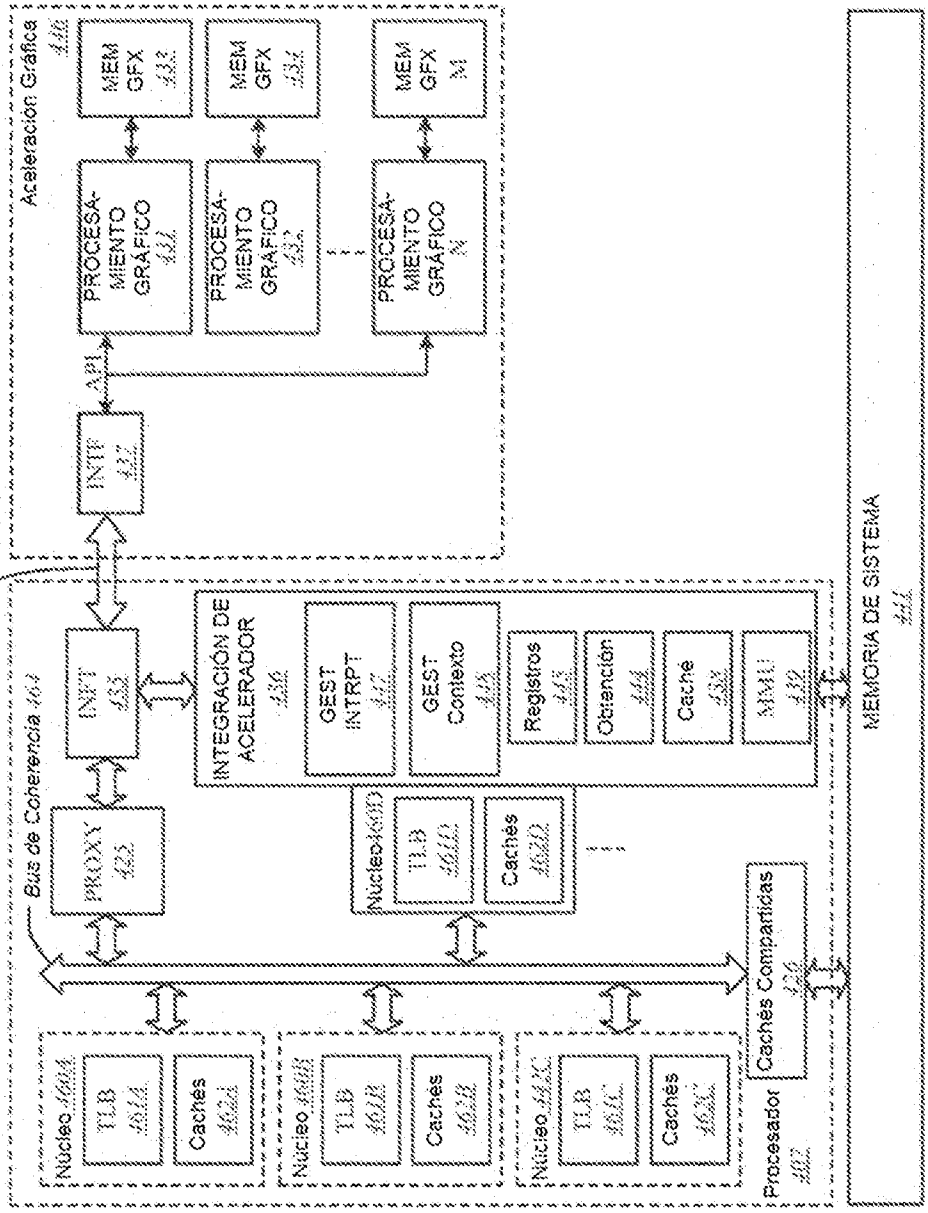

Aceleración Gráfica
446
MEM GFX 432
MEM GFX 434
MEM GFX N
PROCESAMIENTO GRÁFICO 442
PROCESAMIENTO GRÁFICO 432
PROCESAMIENTO GRÁFICO N
API
INTF 427
440
Bus de Coherencia 464
INTF 453
PROXY 425
INTEGRACIÓN DE ACELERADOR 436
GEST INTRPT 447
GEST Contexto 448
Registros 445
Obtención 444
Caché 438
MMU 439
Núcleo 460D
TLB 460D
Cachés 462D
Núcleo 460A
TLB 460A
Cachés 462A
Núcleo 460B
TLB 460B
Cachés 462B
Núcleo 460C
TLB 460C
Cachés 462C
Procesador 402
Cachés Compartidas 426
MEMORIA DE SISTEMA 441

FIG. 4C

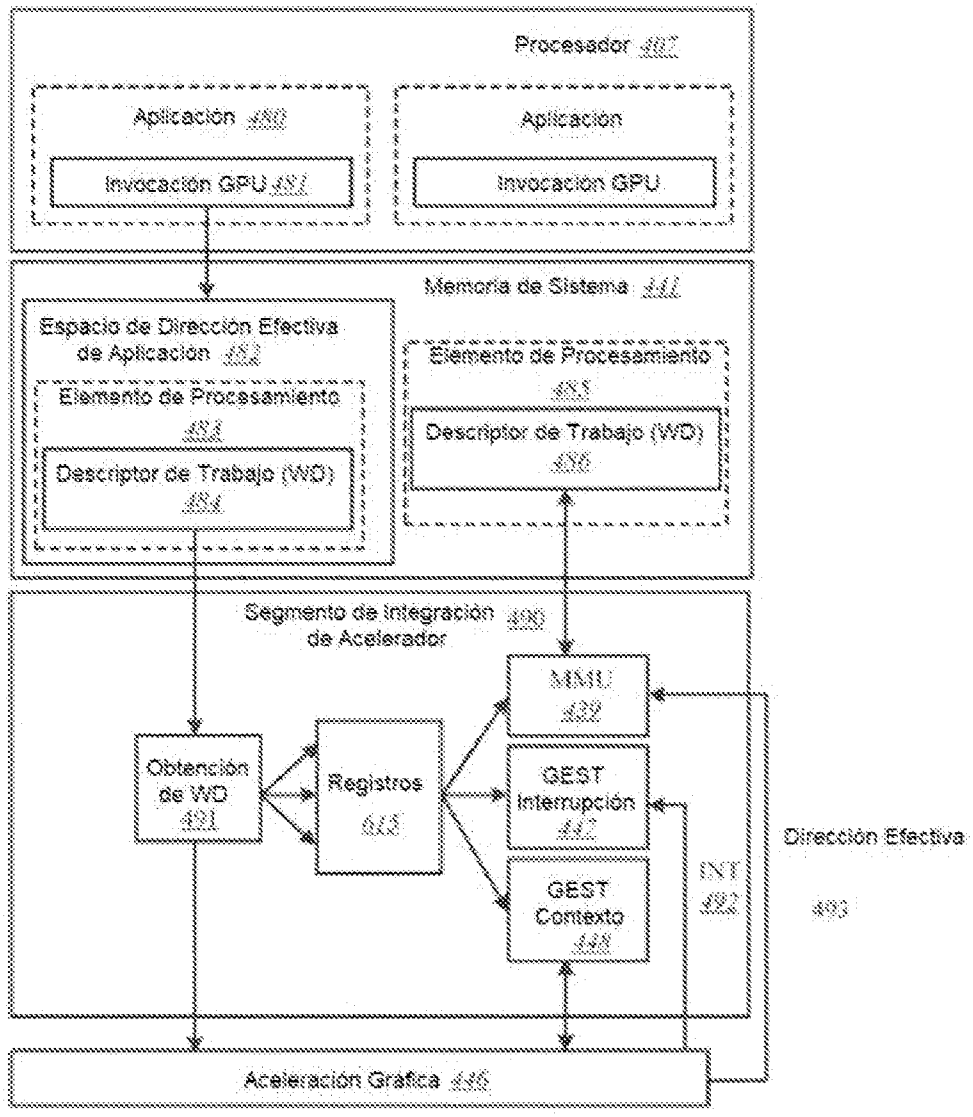

Procesador 407
Aplicación 480
Invocación GPU 481
Aplicación
Invocación GPU
Memoria de Sistema 411
Espacio de Dirección Efectiva de Aplicación 482
Elemento de Procesamiento 491
Descriptor de Trabajo (WD) 484
Elemento de Procesamiento 485
Descriptor de Trabajo (WD) 486
Segmento de Integración de Acelerador 490
MMU 439
Obtención de WD 431
Registros 445
GEST Interrupción 447
GEST Contexto 448
INT 492
Dirección Efectiva 493
Aceleración Gráfica 446

FIG. 4D

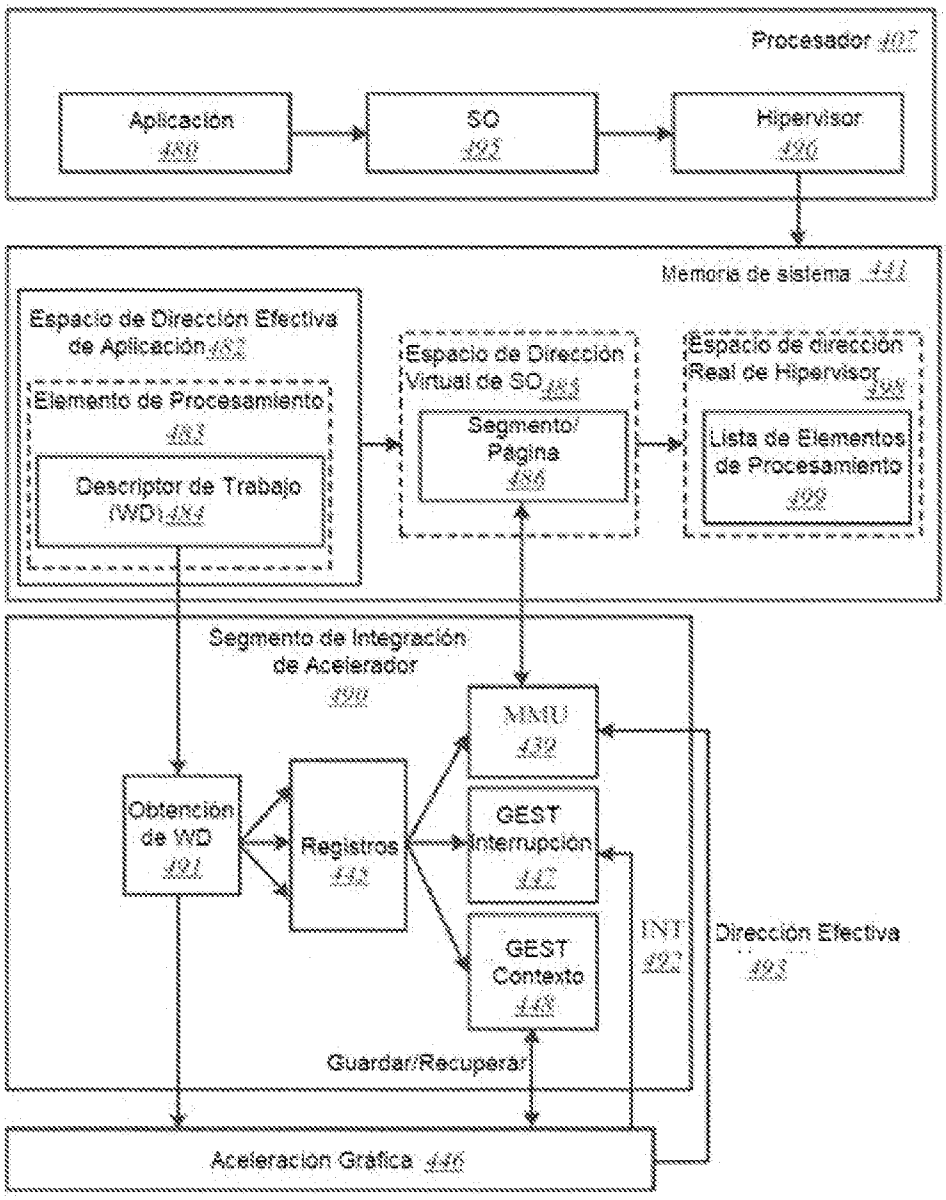

Procesador 407
Aplicación 460
SO 422
Hipervisor 426
Memoria de sistema 441
Espacio de Dirección Efectiva de Aplicación 462
Elemento de Procesamiento 491
Descriptor de Trabajo (WD) 484
Espacio de Dirección Virtual de SO 482
Segmento/ Página 486
Espacio de dirección Real de Hipervisor 488
Lista de Elementos de Procesamiento 499
Segmento de Integración de Acelerador 600
MMU 432
Obtención de WD 421
Registros 445
GEST Interrupción 447
GEST Contexto 448
INT 427
Dirección Efectiva 493
Guardar/Recuperar
Aceleración Gráfica 446

FIG. 4E

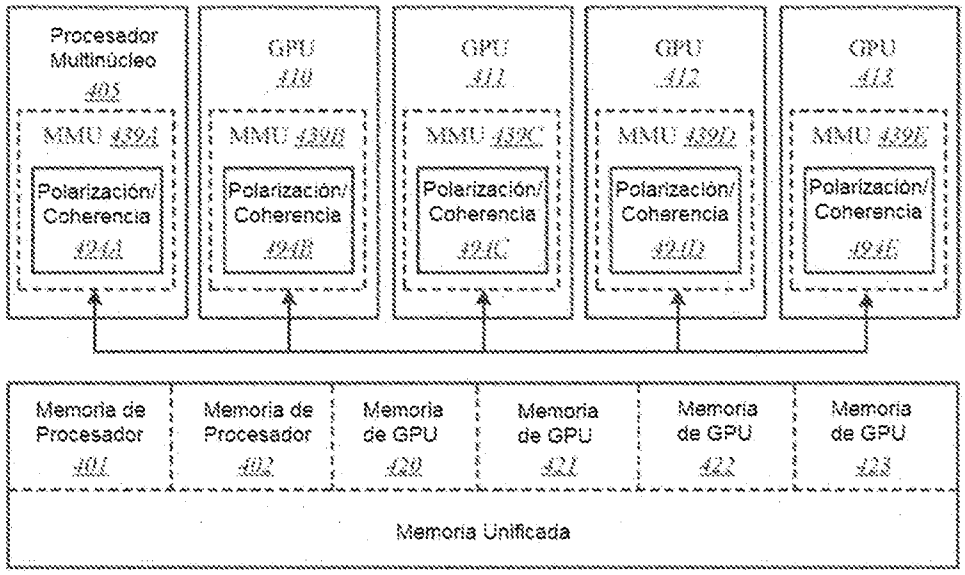

Procesador Multinúcleo
405
MMU 439A
Polarización/ Coherencia
439A
GPU
410
MMU 439B
Polarización/ Coherencia
439B
GPU
411
MMU 439C
Polarización/ Coherencia
439C
GPU
412
MMU 439D
Polarización/ Coherencia
439D
GPU
413
MMU 439E
Polarización/ Coherencia
439E
Memoria de Procesador
401
Memoria de Procesador
402
Memoria de GPU
420
Memoria de GPU
421
Memoria de GPU
422
Memoria de GPU
423
Memoria Unificada

FIG. 4F

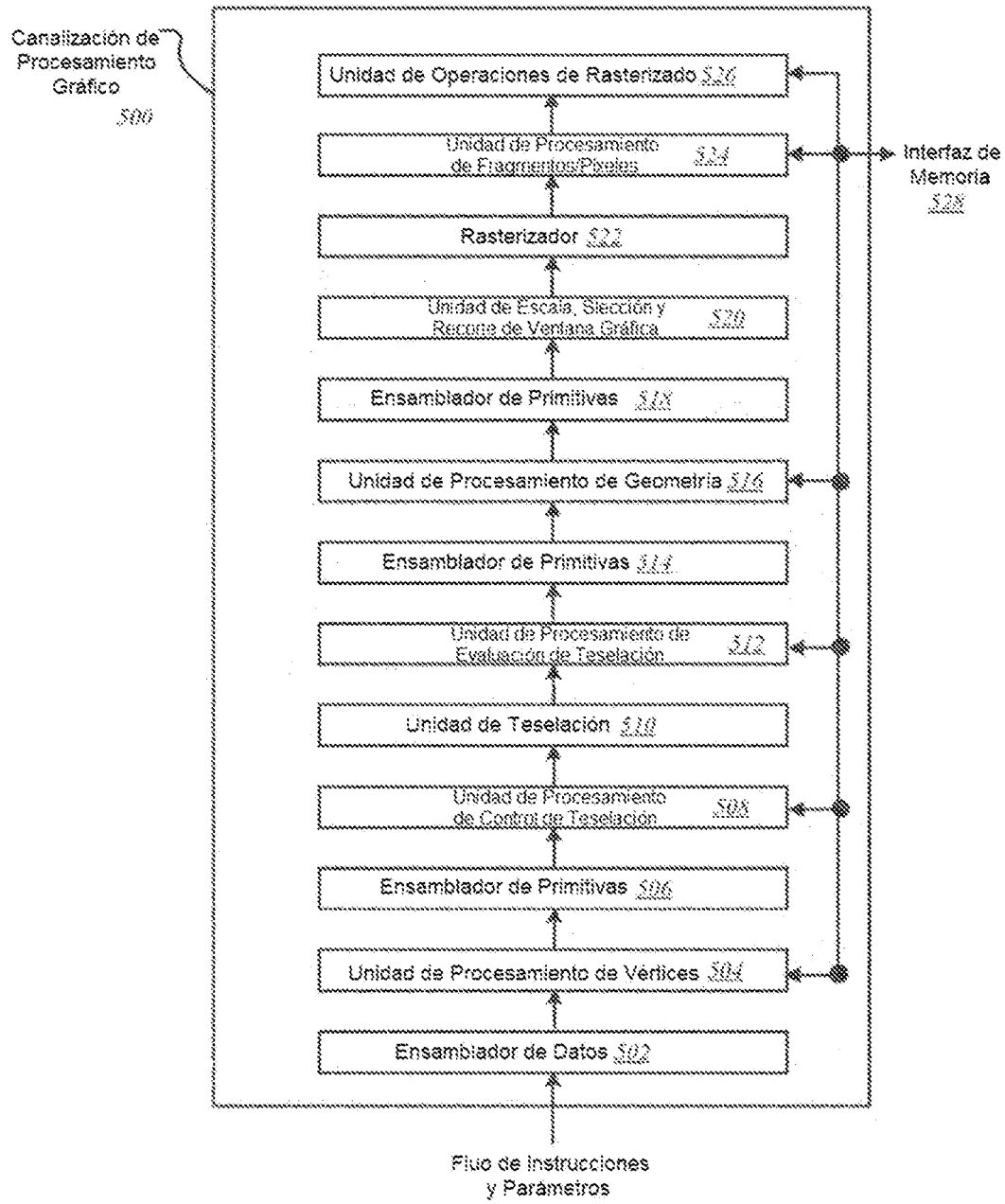

Canalización de Procesamiento Gráfico 500
Unidad de Operaciones de Rasterizado 526
Unidad de Procesamiento de Fragmentos/Pixeles 524
Rasterizador 522
Unidad de Escala, Sección y Recorte de Ventana Gráfica 520
Ensamblador de Primitivas 518
Unidad de Procesamiento de Geometría 516
Ensamblador de Primitivas 514
Unidad de Procesamiento de Evaluación de Teselación 512
Unidad de Teselación 510
Unidad de Procesamiento de Control de Teselación 508
Ensamblador de Primitivas 506
Unidad de Procesamiento de Vértices 504
Ensamblador de Datos 502
Interfaz de Memoria 528
Flujo de Instrucciones y Parámetros

FIG. 5

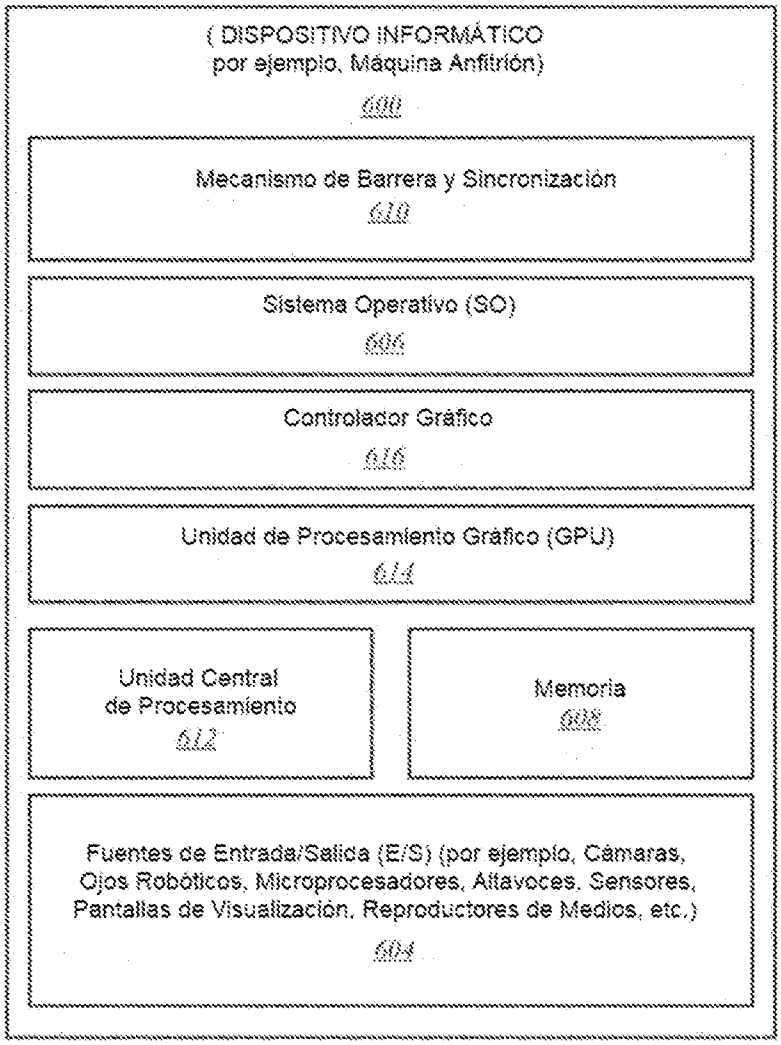

(DISPOSITIVO INFORMÁTICO
por ejemplo, Máquina Anfitrión)
600
Mecanismo de Barrera y Sincronización
610
Sistema Operativo (SO)
606
Controlador Gráfico
616
Unidad de Procesamiento Gráfico (GPU)
614
Unidad Central
de Procesamiento
612
Memoria
608
Fuentes de Entrada/Salida (E/S) (por ejemplo, Cámaras,
Ojos Robóticos, Microprocesadores, Altavoces, Sensores,
Pantallas de Visualización, Reproductores de Medios, etc.)
604

FIG. 6

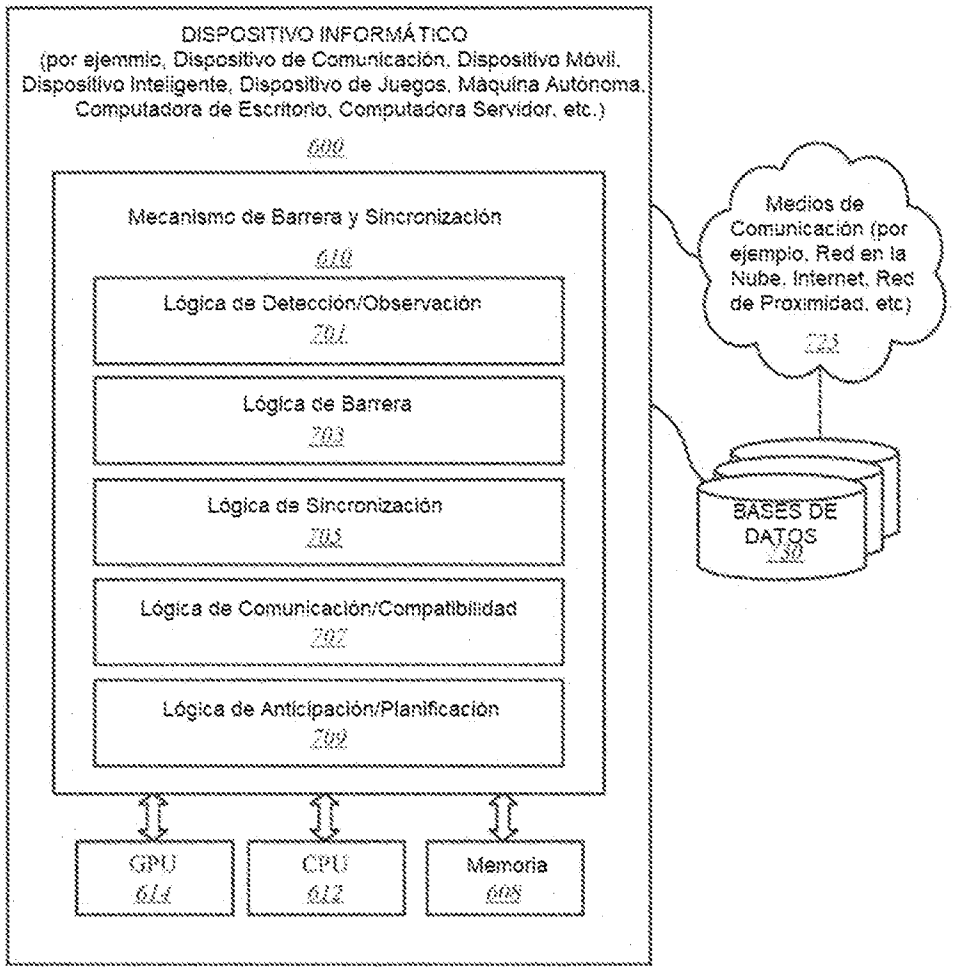

DISPOSITIVO INFORMÁTICO
(por ejemmio, Dispositivo de Comunicación, Dispositivo Móvil, Dispositivo Inteligente, Dispositivo de Juegos, Máquina Autónoma, Computadora de Escritorio, Computadora Servidor, etc.)
600
Mecanismo de Barrera y Sincronización
610
Lógica de Detección/Observación
201
Lógica de Barrera
203
Lógica de Sincronización
205
Lógica de Comunicación/Compatibilidad
207
Lógica de Anticipación/Planificación
209
GPU
614
CPU
612
Memoria
608
Medios de Comunicación (por ejemplo, Red en la Nube, Internet, Red de Proximidad, etc)
223
BASES DE DATOS
230

FIG. 7

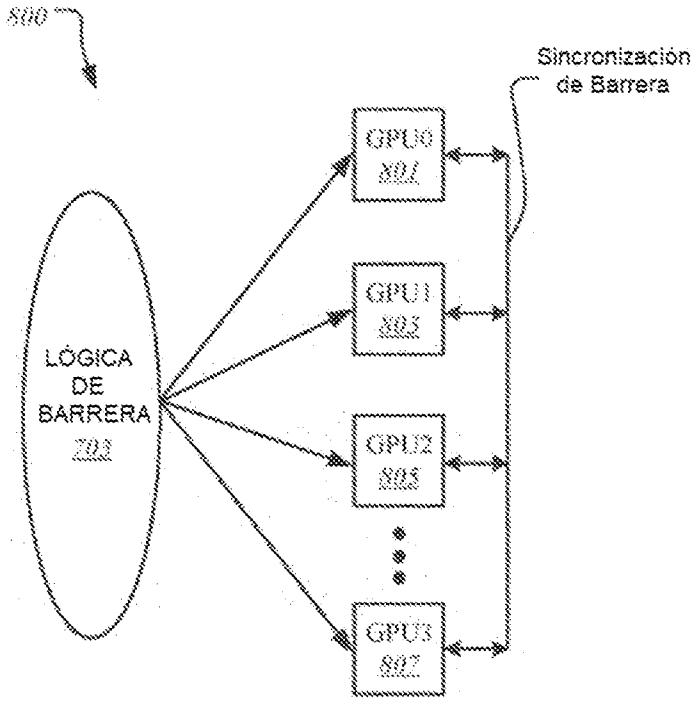

800
Sincronización de Barrera
GPU0
801
GPU1
803
GPU2
805
GPU3
807
LÓGICA DE BARRERA
203

FIG. 8A

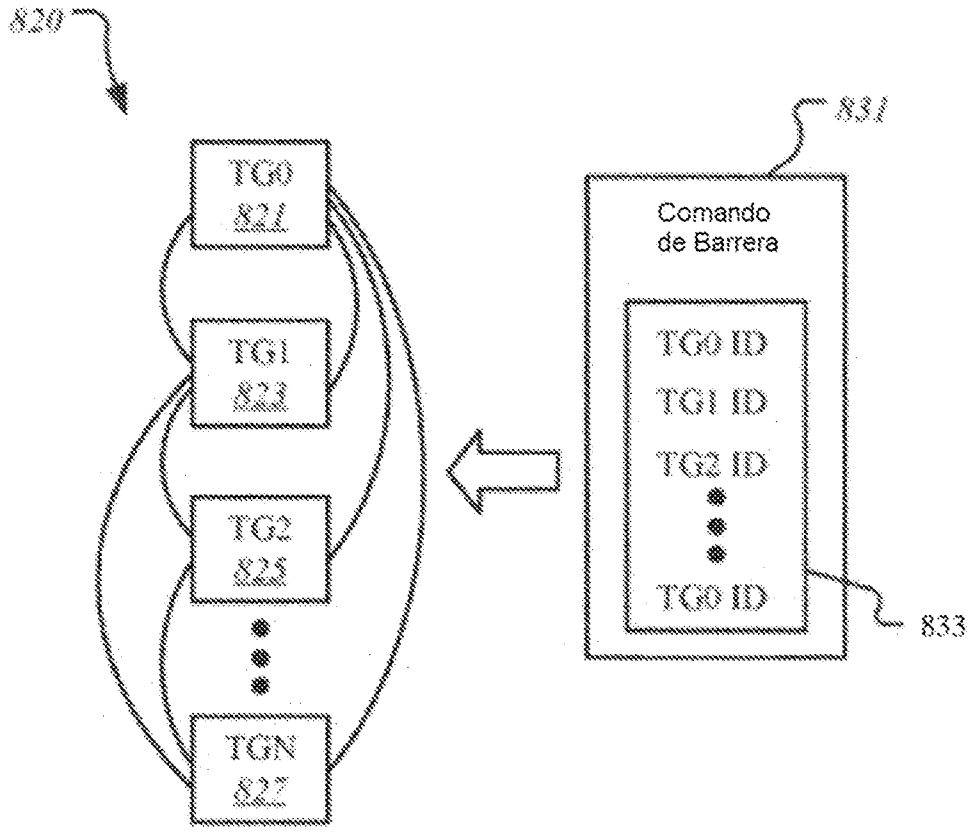

820
831
TG0
821
TG1
823
TG2
825
TGN
827
Comando
de Barrera
TG0 ID
TG1 ID
TG2 ID
TG0 ID
833

Fig. 8B

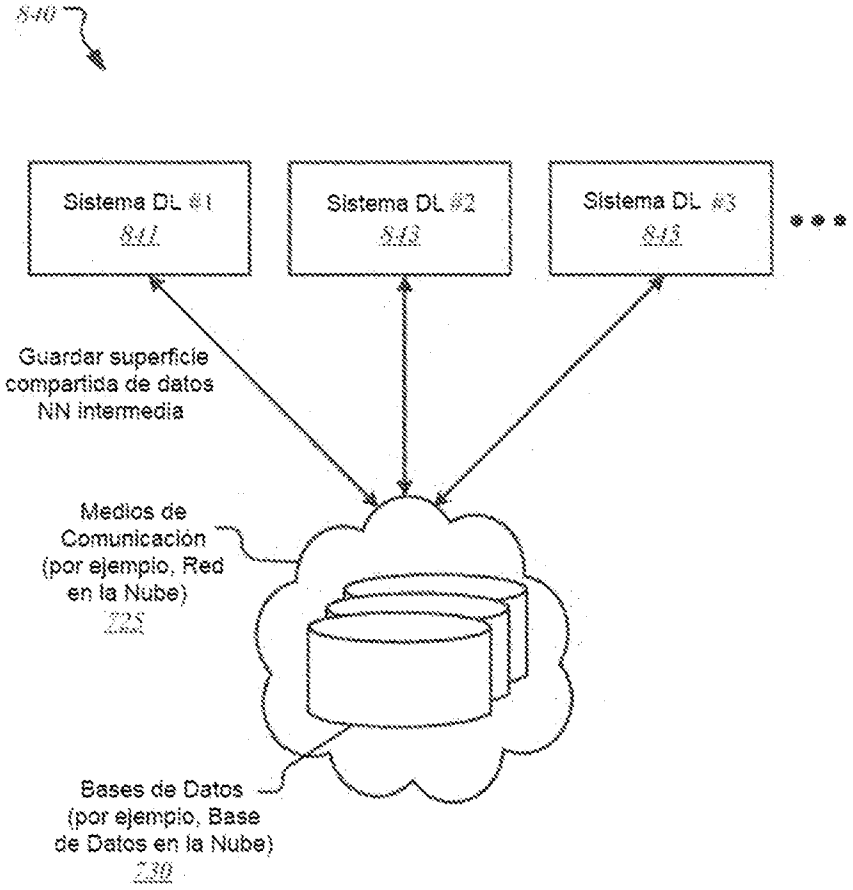

840
Sistema DL #1
841
Sistema DL #2
843
Sistema DL #3
845
Guardar superficie compartida de datos NN intermedia
Medios de Comunicación (por ejemplo, Red en la Nube)
225
Bases de Datos (por ejemplo, Base de Datos en la Nube)
230

FIG. 8C

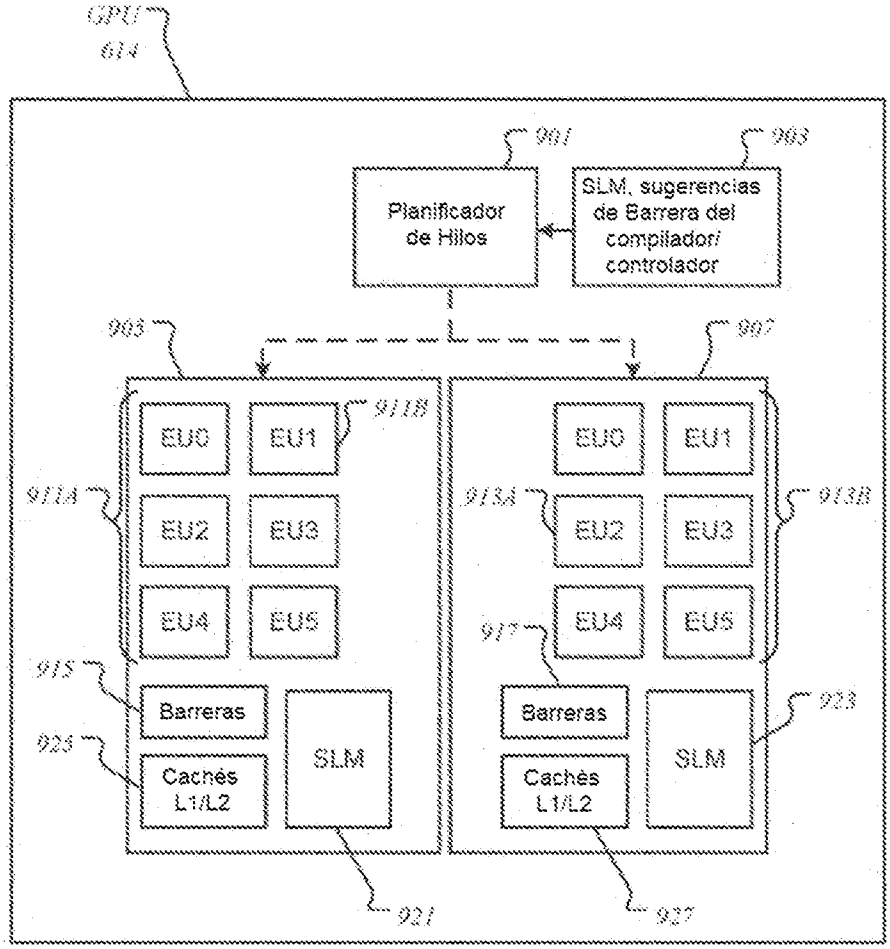

GPU
614
901
903
Planificador de Hilos
SLM, sugerencias de Barrera del compilador/ controlador
905
907
911B
911A
913A
913B
EU0
EU1
EU2
EU3
EU4
EU5
915
917
923
925
Barreras
Cachés L1/L2
SLM
921
927

FIG. 9A

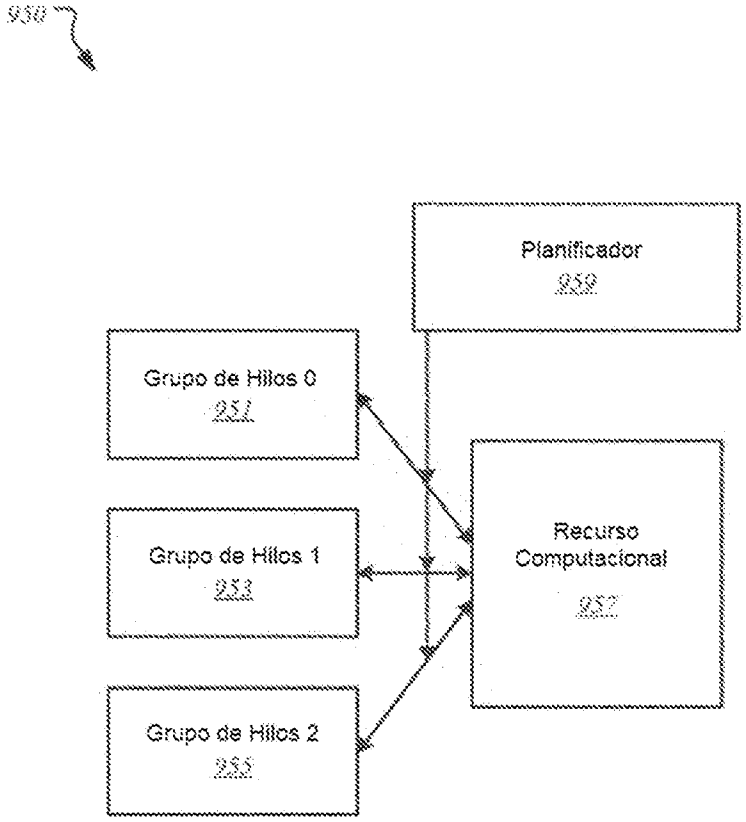

950
Planificador
959
Grupo de Hilos 0
251
Grupo de Hilos 1
253
Grupo de Hilos 2
255
Recurso
Computacional
257

FIG. 9B

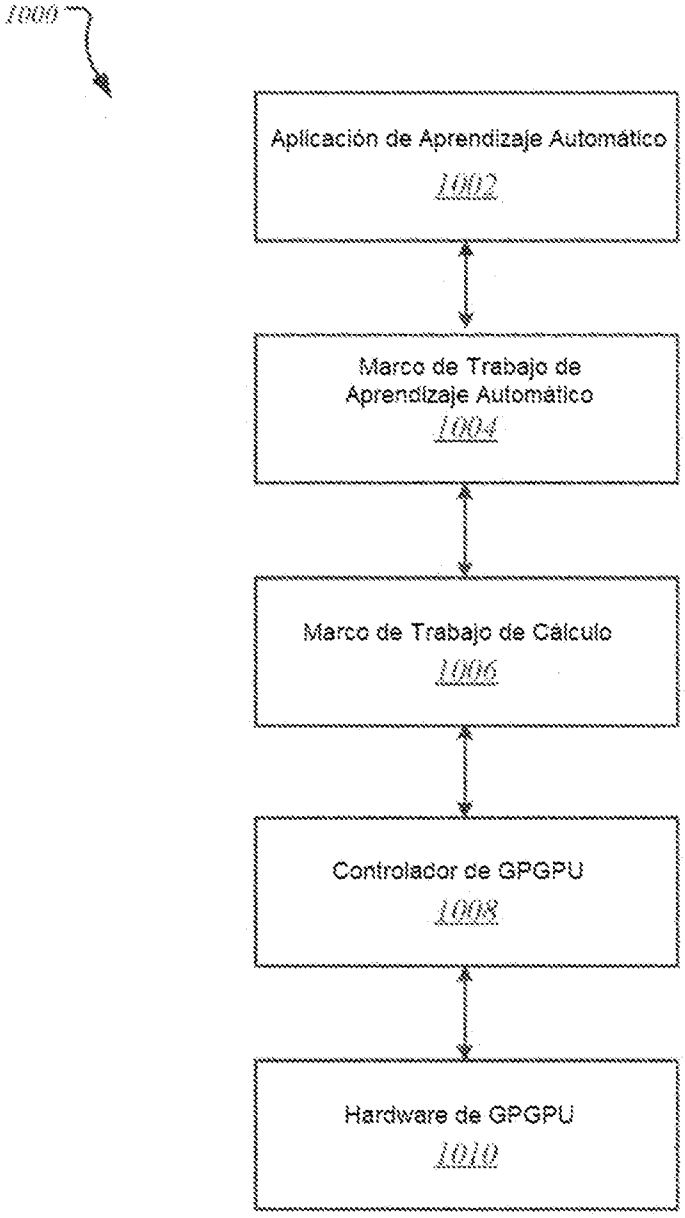

1000
Aplicación de Aprendizaje Automático
1002
Marco de Trabajo de Aprendizaje Automático
1004
Marco de Trabajo de Cálculo
1006
Controlador de GPGPU
1008
Hardware de GPGPU
1010

FIG. 10

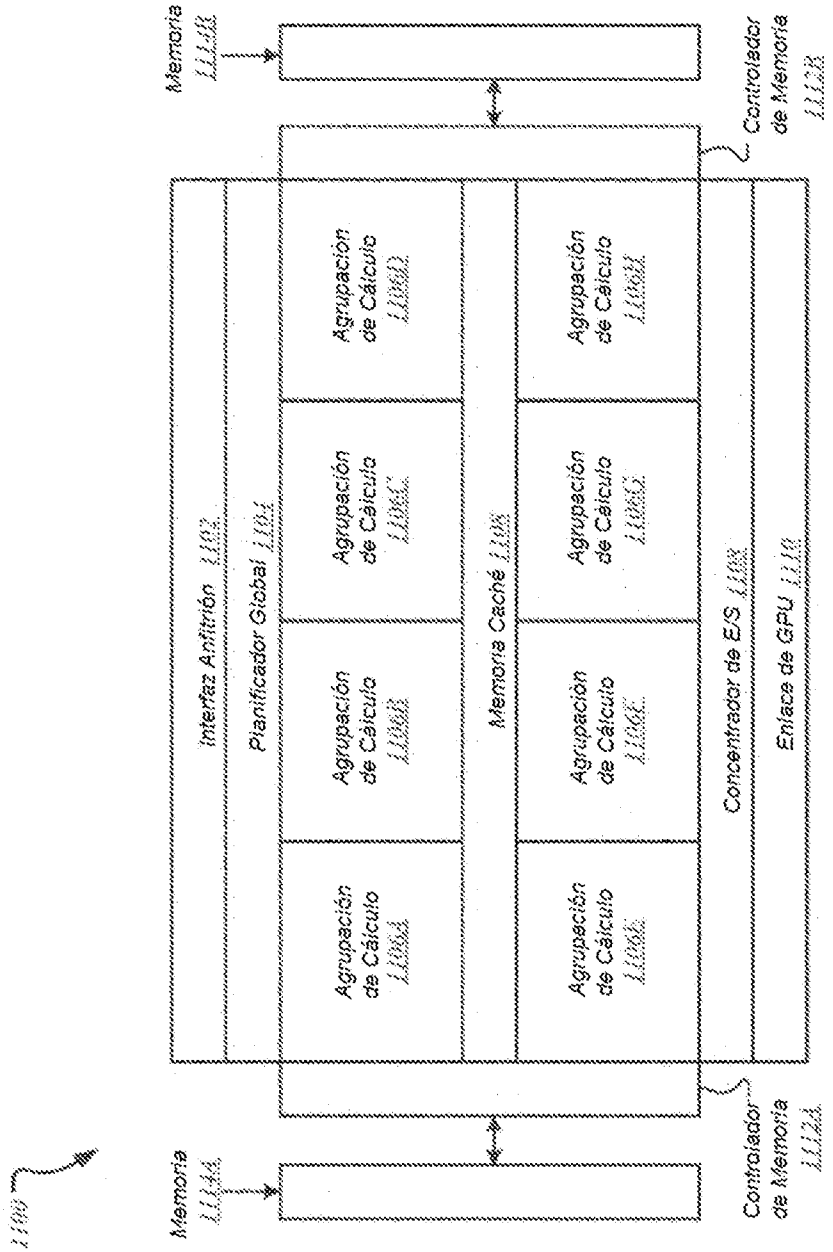

1100
Memoria
1144
Controlador de Memoria
1124
Interfaz Anfitrión 1102
Planificador Global 1104
Agrupación de Cálculo 1106A
Agrupación de Cálculo 1106B
Agrupación de Cálculo 1106C
Agrupación de Cálculo 1106D
Agrupación de Cálculo 1106E
Agrupación de Cálculo 1106F
Agrupación de Cálculo 1106G
Agrupación de Cálculo 1106H
Memoria Caché 1108
Concentrador de E/S 1118
Enlace de GPU 1120
Controlador de Memoria
1122
Memoria
1146

FIG. 11

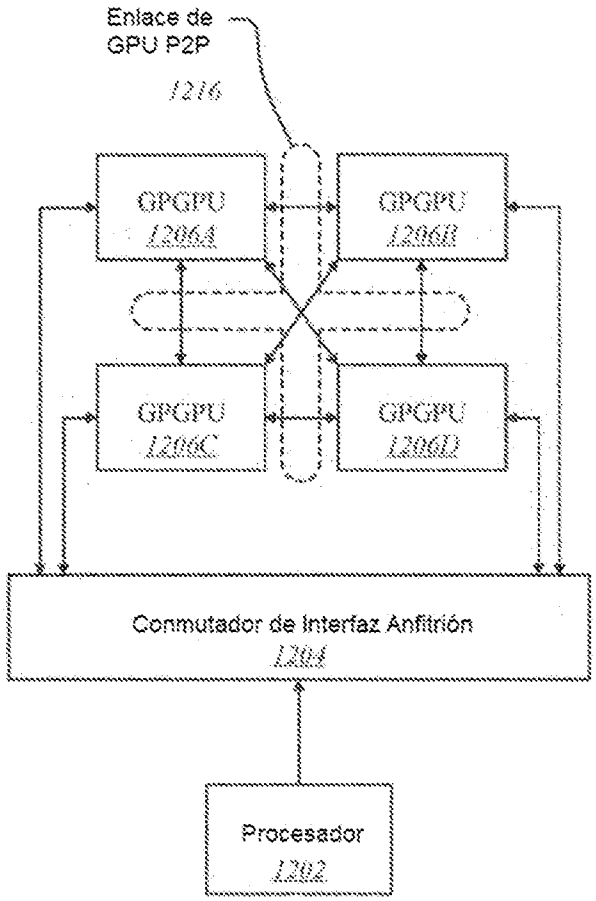

Enlace de
GPU P2P
1216
GPGPU
1206A
GPGPU
1206B
GPGPU
1206C
GPGPU
1206D
Conmutador de Interfaz Anfitrión
1204
Procesador
1202

FIG. 12

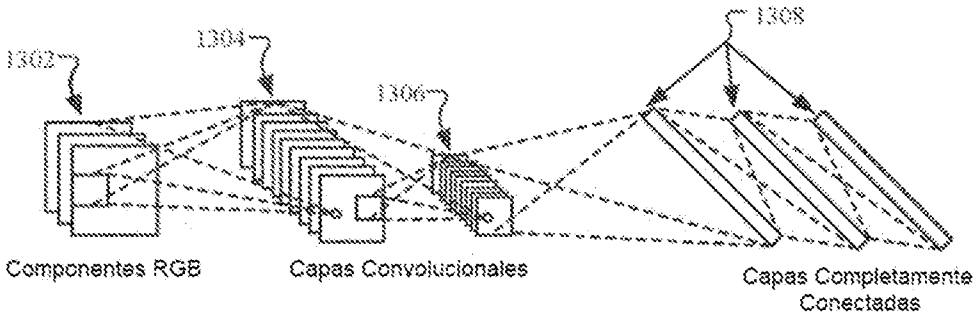


FIG. 13A

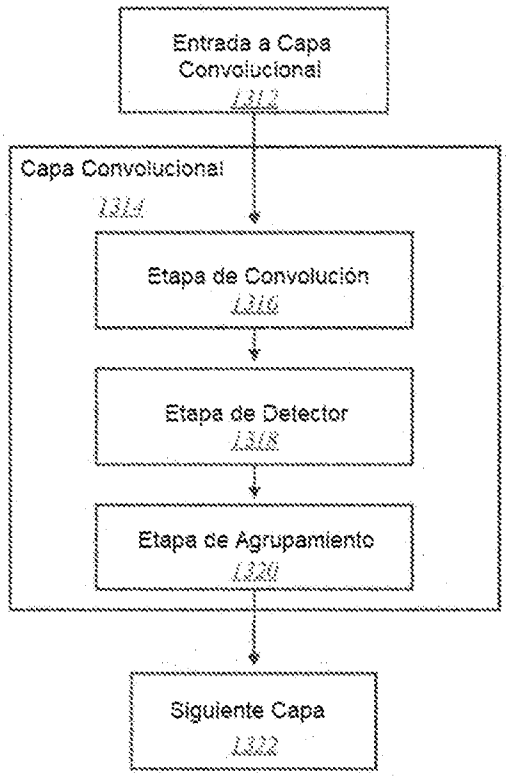


FIG. 13B

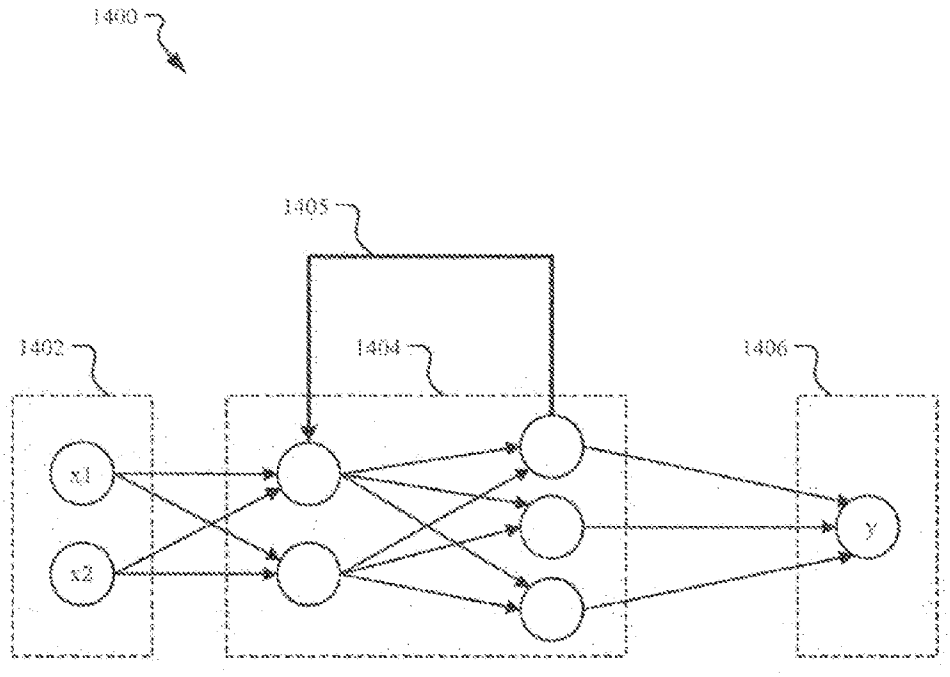


FIG. 14

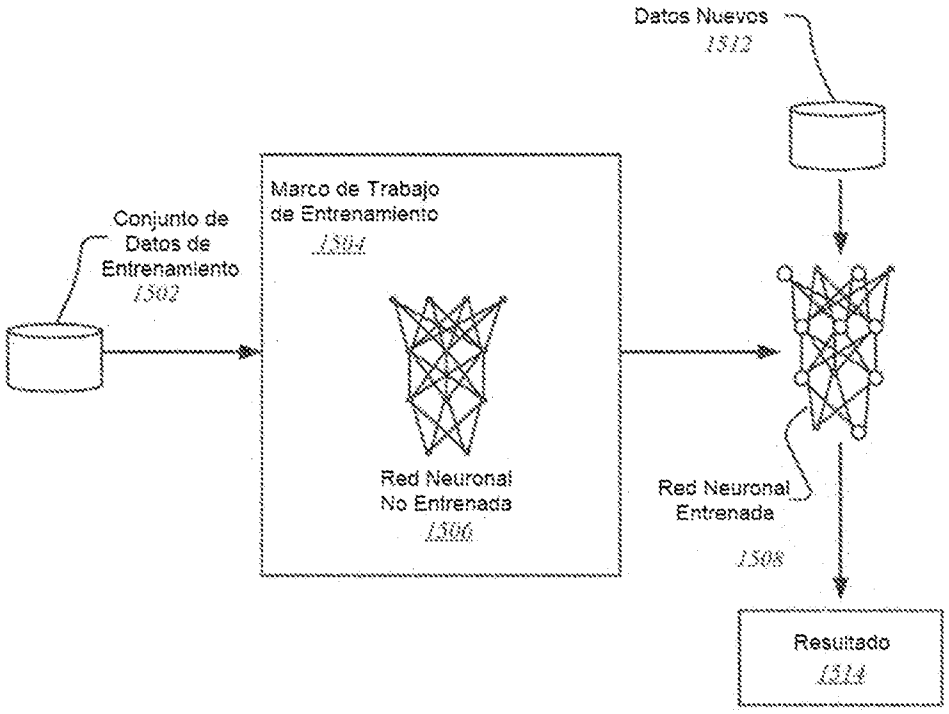

Datos Nuevos
1512
Marco de Trabajo
de Entrenamiento
1504
Conjunto de
Datos de
Entrenamiento
1502
Red Neuronal
No Entrenada
1506
Red Neuronal
Entrenada
1508
Resultado
1514

FIG. 15

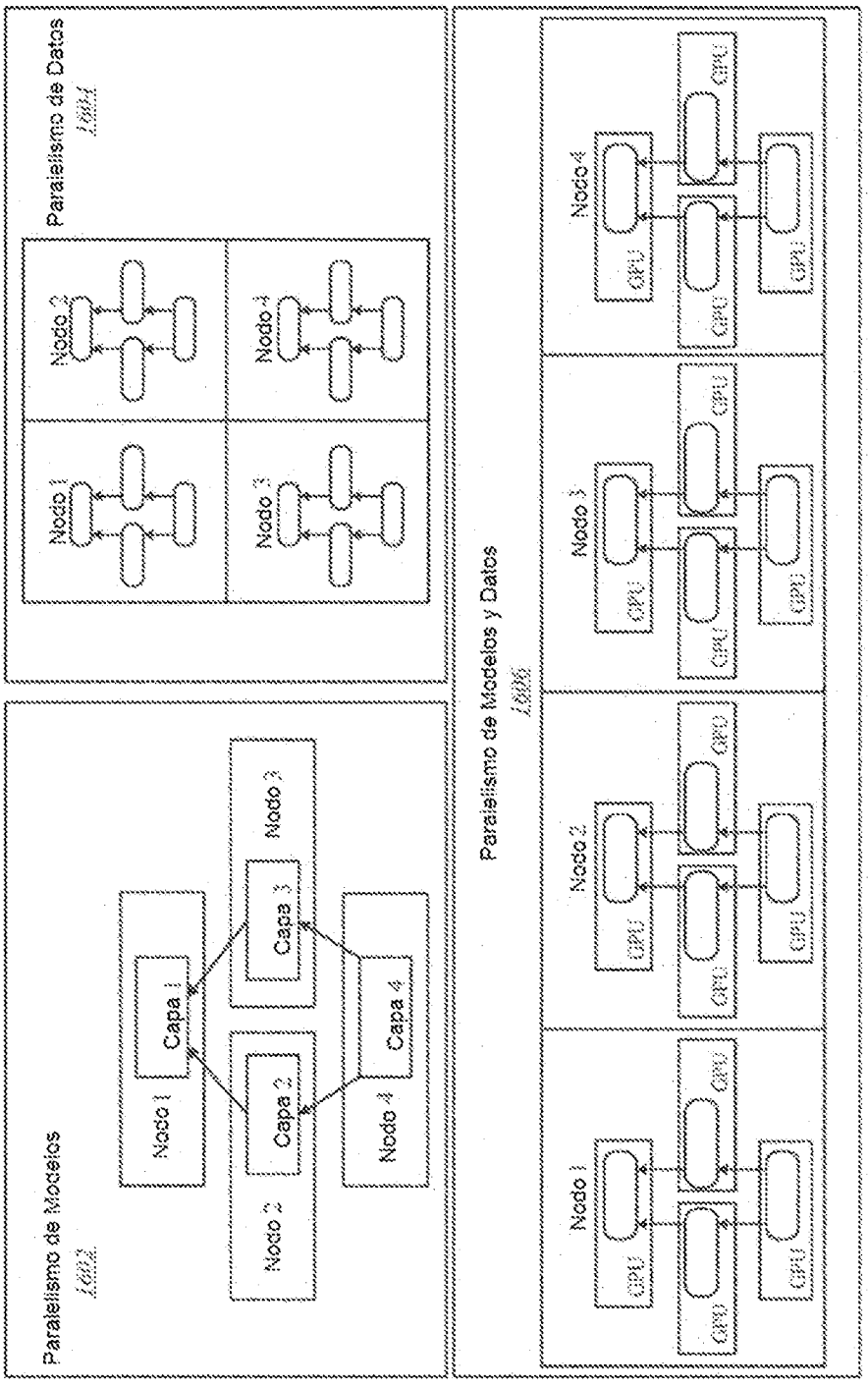

Paralelismo de Datos
1004
Nodo 1
Nodo 2
Nodo 3
Nodo 4
Paralelismo de Modelos
1002
Nodo 1
Capa 1
Nodo 2
Capa 2
Nodo 3
Capa 3
Nodo 4
Capa 4
Paralelismo de Modelos y Datos
1006
Nodo 1
Nodo 2
Nodo 3
Nodo 4
GPU

FIG. 16

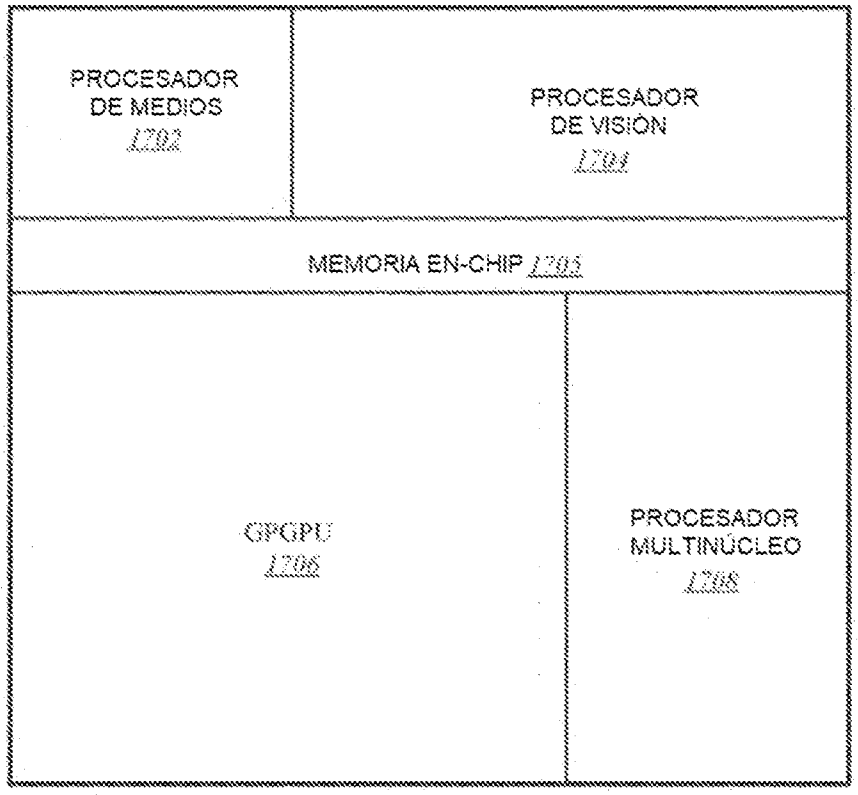

PROCESADOR DE MEDIOS
1702
PROCESADOR DE VISIÓN
1704
MEMORIA EN-CHIP 1705
GPGPU
1706
PROCESADOR MULTINÚCLEO
1708

FIG. 17

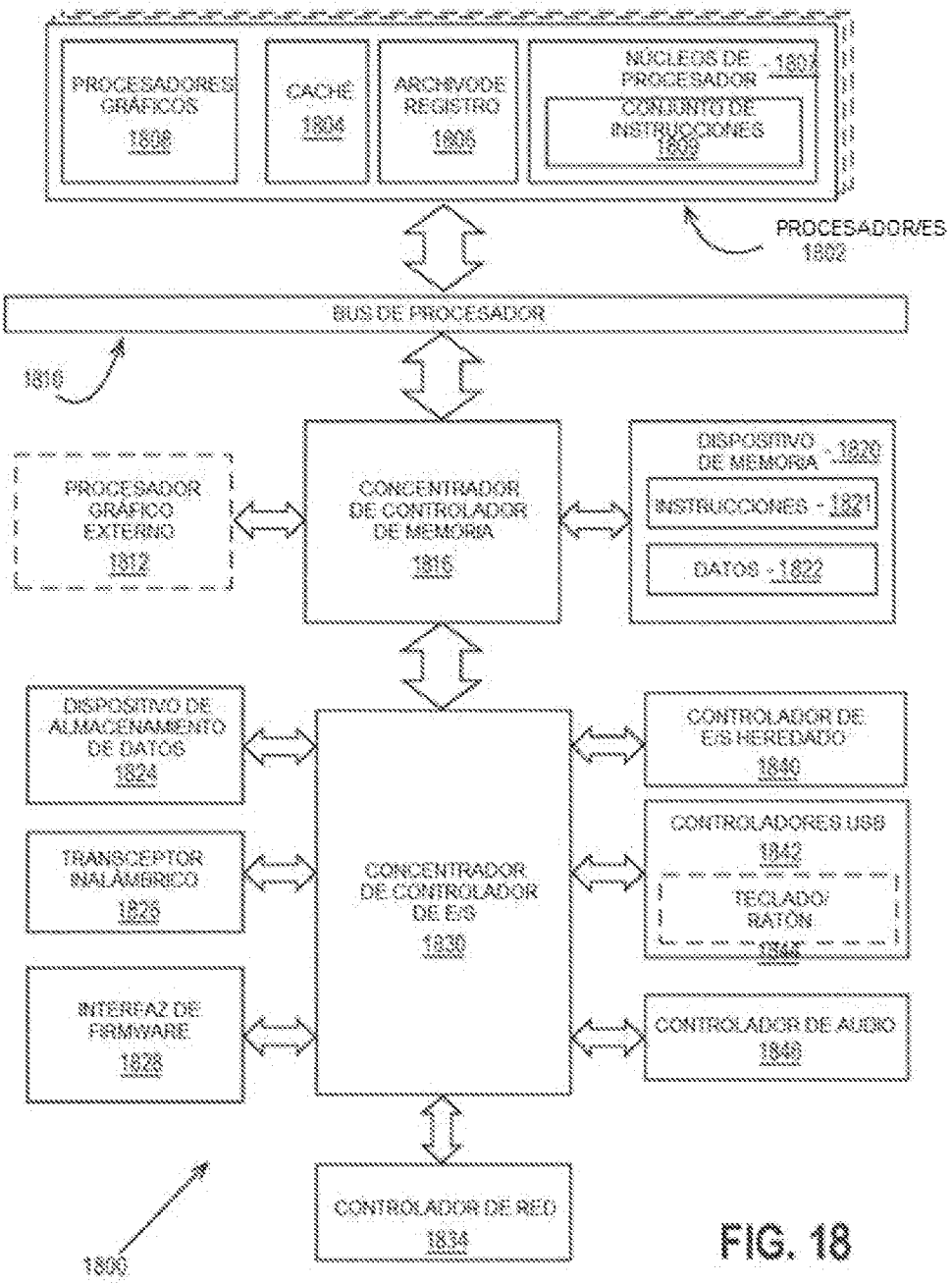

PROCESADORES GRÁFICOS 1838
CACHÉ 1804
ARCHIVO DE REGISTRO 1806
NÚCLEOS DE PROCESADOR 1801
CONJUNTO DE INSTRUCCIONES 1809
PROCESADORES 1802
BUS DE PROCESADOR
1816
PROCESADOR GRÁFICO EXTERNO 1812
CONCENTRADOR DE CONTROLADOR DE MEMORIA 1816
DISPOSITIVO DE MEMORIA 1820
INSTRUCCIONES 1821
DATOS 1822
DISPOSITIVO DE ALMACENAMIENTO DE DATOS 1824
TRANSCEPTOR INALÁMBRICO 1826
INTERFAZ DE FIRMWARE 1828
CONCENTRADOR DE CONTROLADOR DE E/S 1830
CONTROLADOR DE E/S HEREDADO 1840
CONTROLADORES USB 1842
TECLADO/ RATÓN
CONTROLADOR DE AUDIO 1846
CONTROLADOR DE RED 1834
1800

FIG. 18

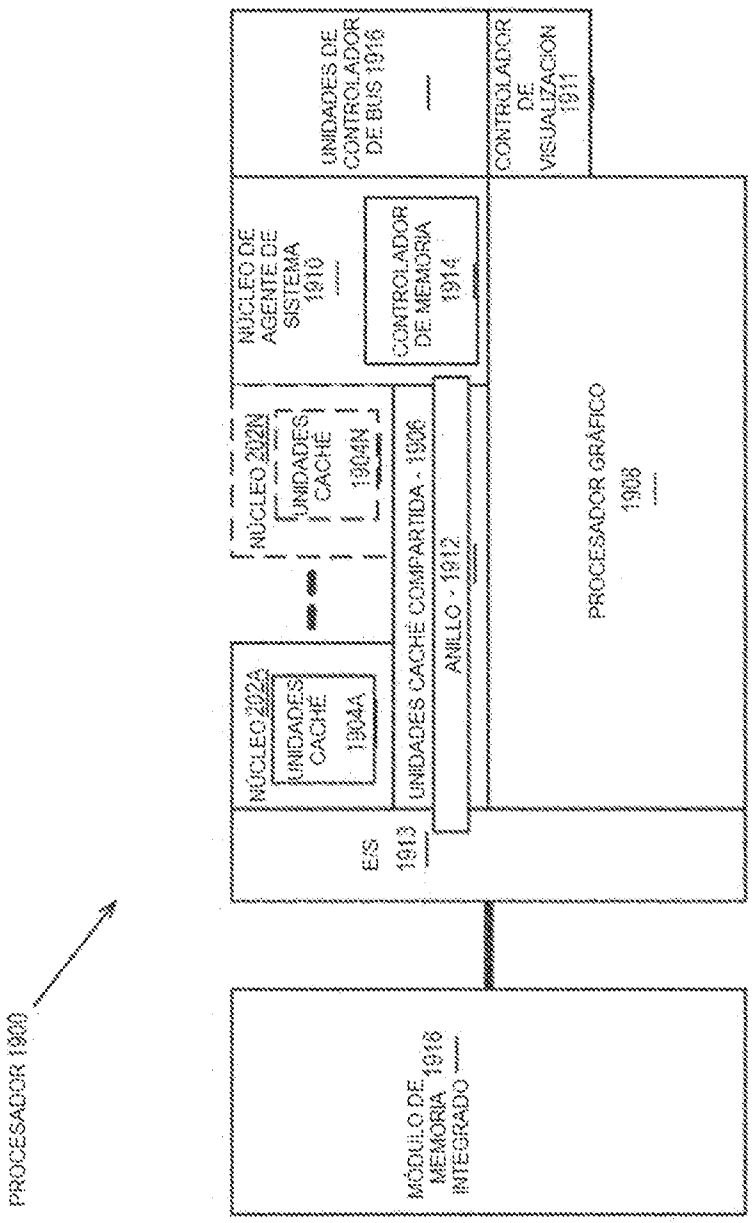

PROCESADOR 1900
MÓDULO DE MEMORIA INTEGRADO 1918
ES 1913
NÚCLEO 202A
UNIDADES CACHÉ 1904A
NÚCLEO 202N
UNIDADES CACHÉ 1904N
UNIDADES CACHÉ COMPARTIDA - 1906
ANILLO - 1912
NÚCLEO DE AGENTE DE SISTEMA 1910
CONTROLADOR DE MEMORIA 1914
UNIDADES DE CONTROLADOR DE BUS 1916
CONTROLADOR DE VISUALIZACIÓN 1911
PROCESADOR GRÁFICO 1908

FIG. 19

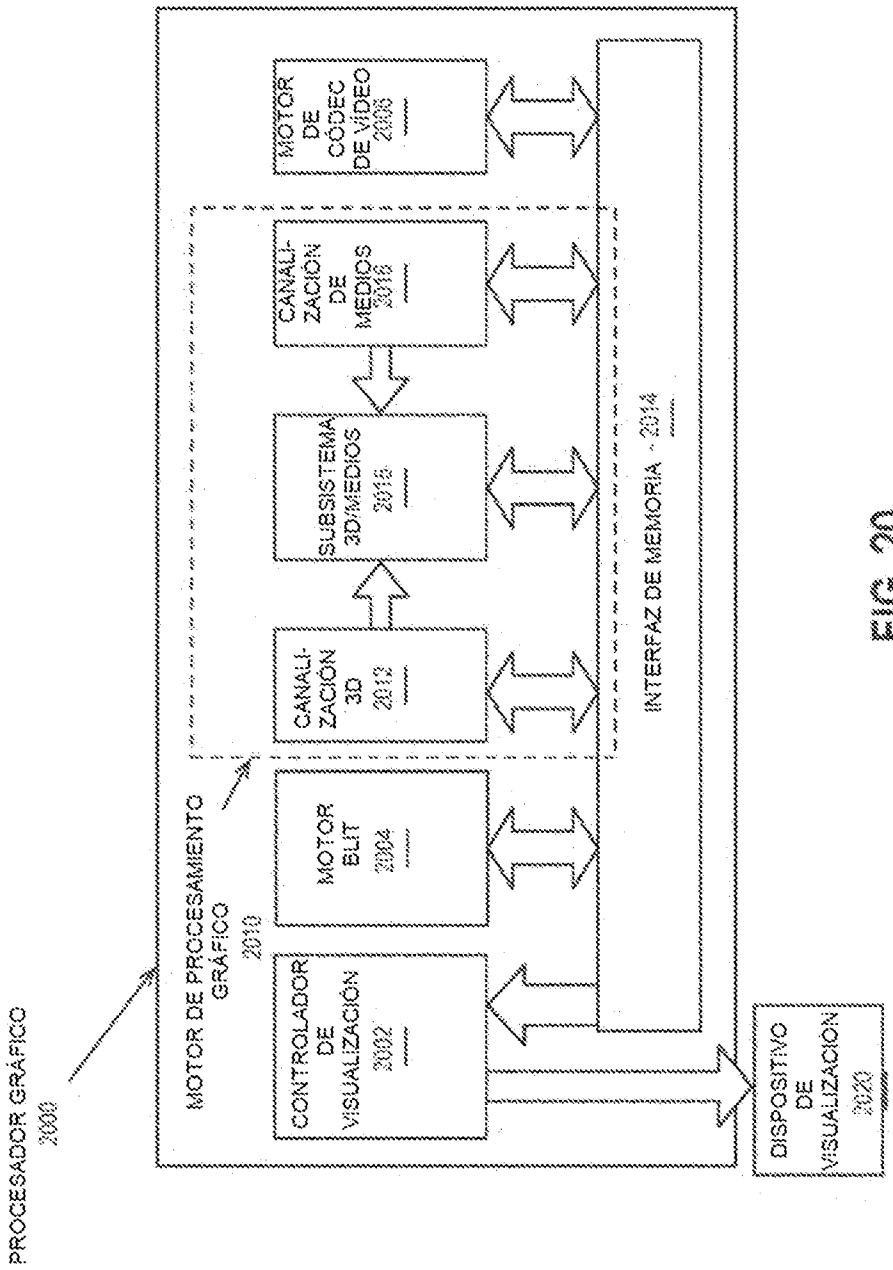

PROCESADOR GRÁFICO
2000
MOTOR DE PROCESAMIENTO GRÁFICO
2010
CONTROLADOR DE VISUALIZACIÓN
2002
MOTOR BLIT
2004
CANALI-ZACIÓN 3D
2012
SUBSISTEMA 3D/MEDIOS
2016
CANALI-ZACIÓN DE MEDIOS
2018
MOTOR DE CÓDEC DE VÍDEO
2006
INTERFAZ DE MEMORIA - 2014
DISPOSITIVO DE VISUALIZACIÓN
2020

FIG. 20

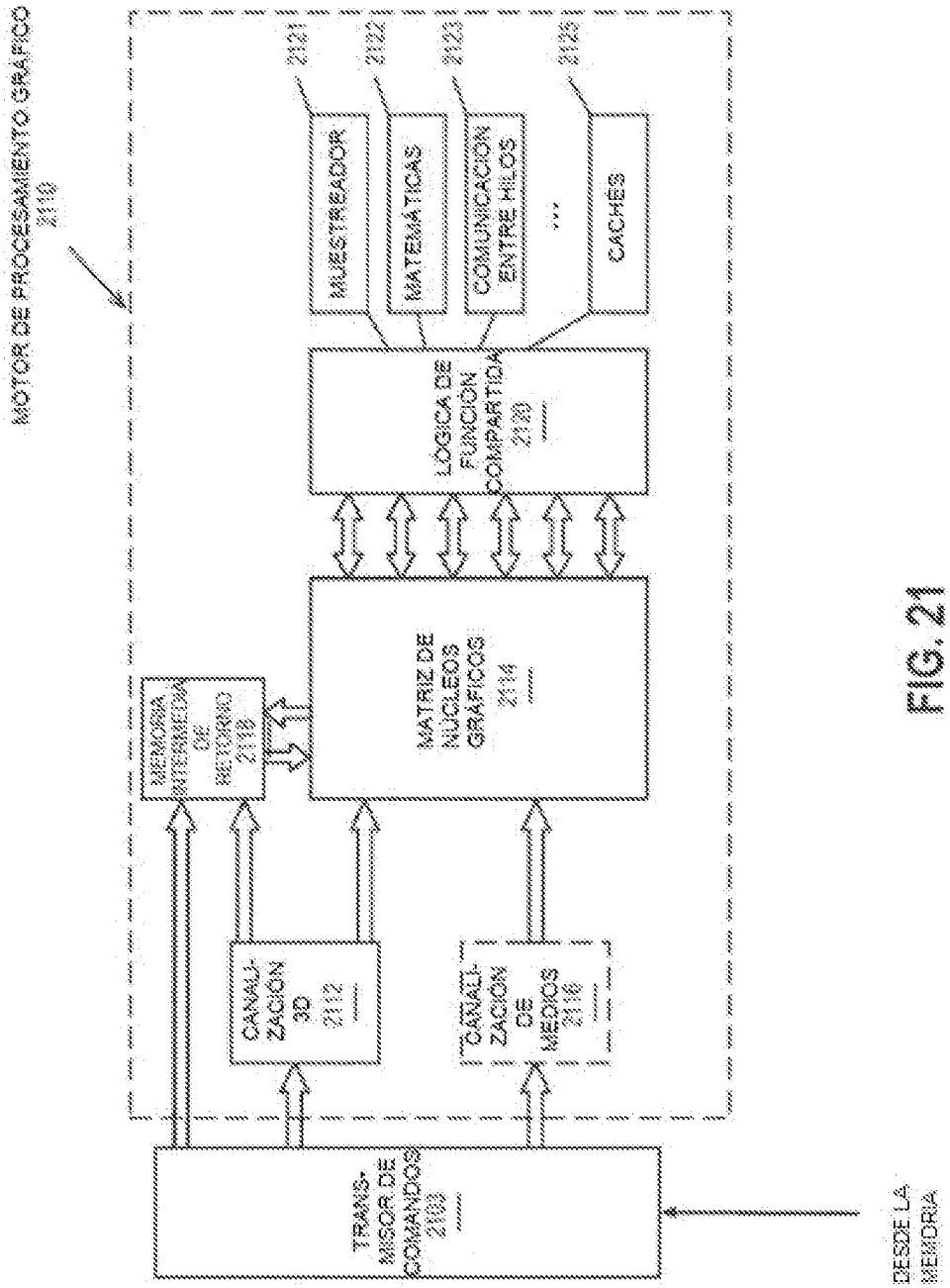

MOTOR DE PROCESAMIENTO GRÁFICO
2110
2121
2122
2123
2125
MUESTREADOR
MATEMÁTICAS
COMUNICACIÓN ENTRE HILOS
...
CACHÉS
LÓGICA DE FUNCIÓN COMPARTIDA
2120
MATRIZ DE NÚCLEOS GRÁFICOS
2114
MEMORIA INTERMEDIA DE RETORNO
2118
CANALI-ZACIÓN 3D
2112
CANALI-ZACIÓN DE MEDIOS
2116
TRANS-MISOR DE COMANDOS
2103
DESDE LA MEMORIA

FIG. 21

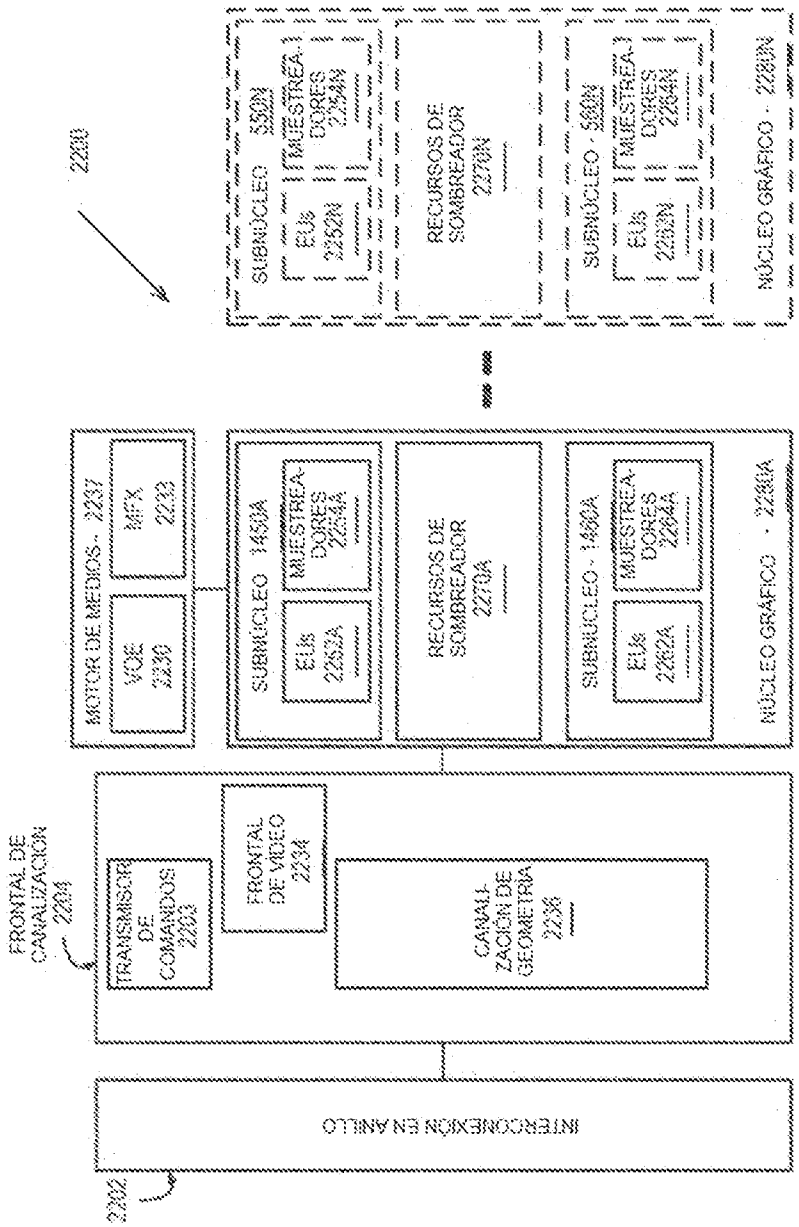

2200
2202
INTERCONEXIÓN EN ANILLO
FRONTAL DE CANALIZACIÓN 2204
TRANSMISOR DE COMANDOS 2203
FRONTAL DE VÍDEO 2234
CANALIZACIÓN DE GEOMETRÍA 2238
MOTOR DE MEDIOS - 2207
VOE 2230
MFX 2232
SUBNÚCLEO 1450A
ELs 2262A
MUESTREADORES 2264A
RECURSOS DE SOMBREADOR 2270A
SUBNÚCLEO - 1460A
ELs 2262A
MUESTREADORES 2264A
NÚCLEO GRÁFICO - 2280A
SUBNÚCLEO 1450N
ELs 2262N
MUESTREADORES 2264N
RECURSOS DE SOMBREADOR 2270N
SUBNÚCLEO - 1460N
ELs 2262N
MUESTREADORES 2264N
NÚCLEO GRÁFICO - 2280N

FIG. 22

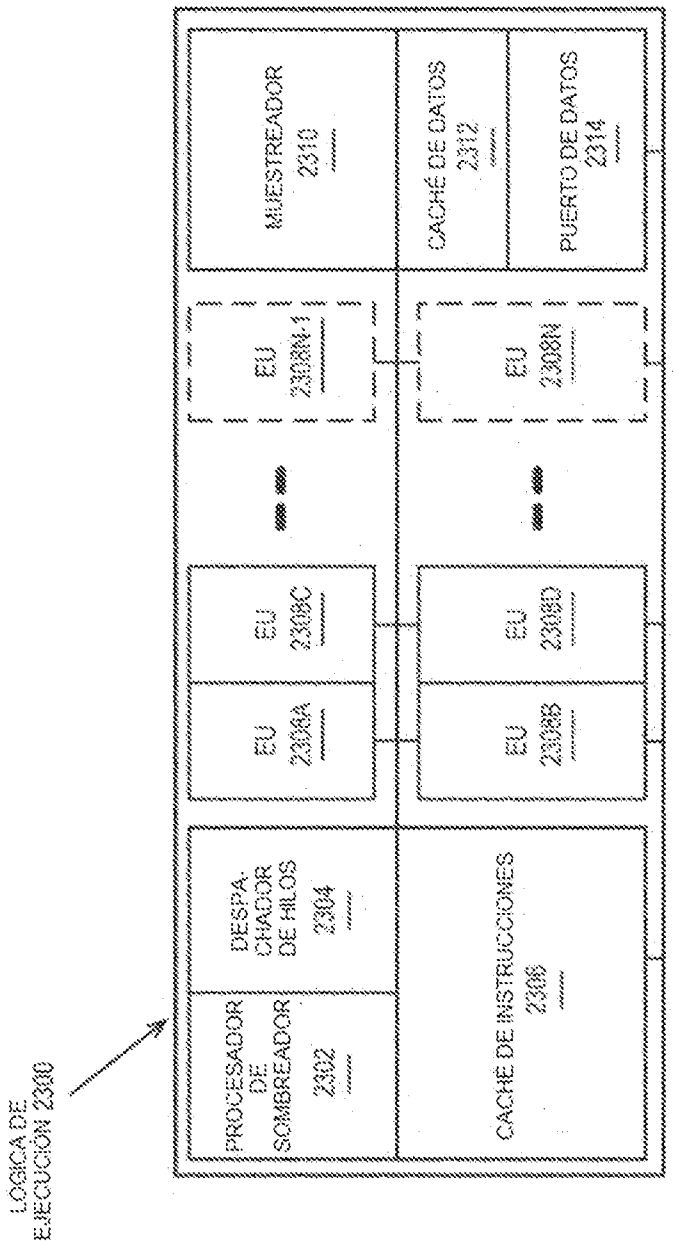

LOGICA DE EJECUCIÓN 2300
PROCESADOR DE SOMBREADOR 2302
DESPACHADOR DE HILOS 2304
CACHÉ DE INSTRUCCIONES 2306
EU 2308A
EU 2308B
EU 2308C
EU 2308D
EU 2308N-1
EU 2308N
MUESTREADOR 2310
CACHÉ DE DATOS 2312
PUERTO DE DATOS 2314

FIG. 23

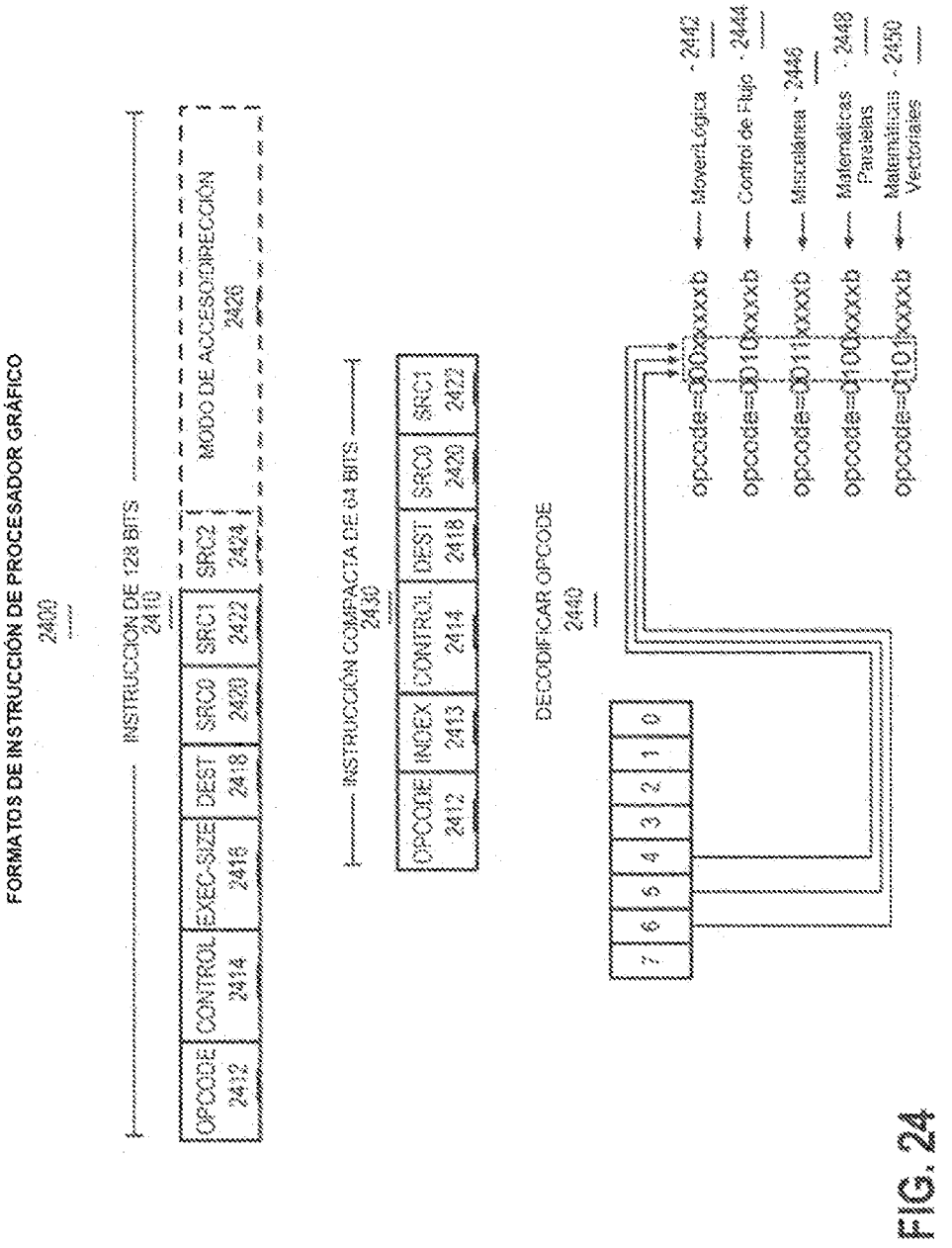

FORMATOS DE INSTRUCCIÓN DE PROCESADOR GRÁFICO
2400
INSTRUCCIÓN DE 128 BITS
2410
OPCODE 2412
CONTROL 2414
EXEC-SIZE 2416
DEST 2418
SRC0 2420
SRC1 2422
SRC2 2424
MODO DE ACCESO/DIRECCIÓN 2426
INSTRUCCIÓN COMPACTA DE 64 BITS
2430
OPCODE 2412
INDEX 2413
CONTROL 2414
DEST 2418
SRC0 2420
SRC1 2422
DECODIFICAR OPCODE
2440
7 6 5 4 3 2 1 0
opcode=00xxxxb
opcode=01xxxxb
opcode=10xxxxb
opcode=11xxxxb
opcode=100xxxxb
opcode=101xxxxb
Mover/Lógica 2442
Control de Flujo 2444
Miscelánea 2446
Matemáticas Paralelas 2448
Matemáticas Vectoriales 2450

FIG. 24

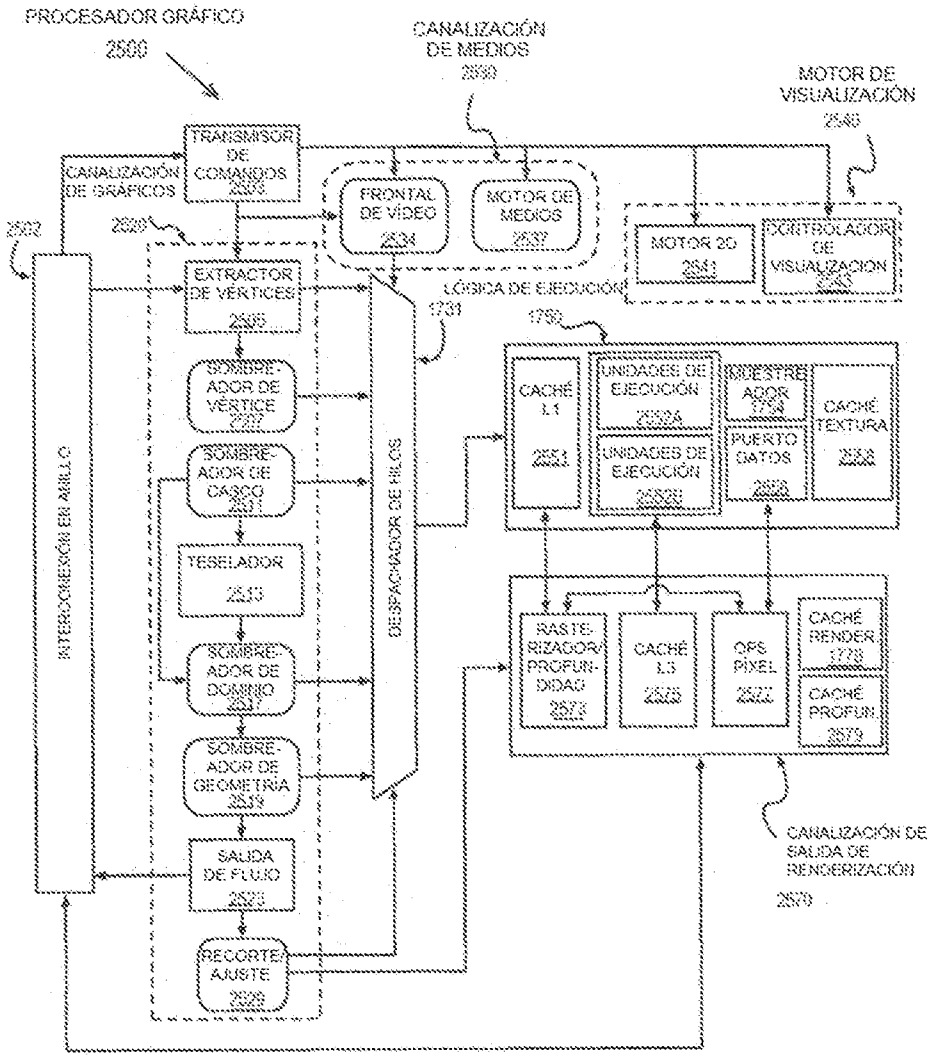

PROCESADOR GRÁFICO
2500
CANALIZACIÓN DE MEDIOS
2530
MOTOR DE VISUALIZACIÓN
2540
TRANSMISOR DE COMANDOS
2503
CANALIZACIÓN DE GRÁFICOS
2520
FRONTAL DE VÍDEO
2534
MOTOR DE MEDIOS
2537
MOTOR 2D
2541
CONTROLADOR DE VISUALIZACIÓN
2543
2502
EXTRACTOR DE VÉRTICES
2505
LÓGICA DE EJECUCIÓN
1731
1750
SOMBREADOR DE VÉRTICE
2507
SOMBREADOR DE CASCO
2511
TESELADOR
2513
SOMBREADOR DE DOMINIO
2517
SOMBREADOR DE GEOMETRÍA
2519
SALIDA DE FLUJO
2523
RECORTE/AJUSTE
2529
INTERCONEXIÓN EN ANILLO
DESPACHADOR DE HILOS
CACHÉ L1
1752
UNIDADES DE EJECUCIÓN
1702A
UNIDADES DE EJECUCIÓN
1702B
MUESTREADOR
1754
PUERTO DATOS
1758
CACHÉ TEXTURA
1756
RASTERIZADOR/PROFUNDIDAD
1573
CACHÉ L3
1775
OPS PÍXEL
1577
CACHÉ RENDER
1778
CACHÉ PROFUN
1579
CANALIZACIÓN DE SALIDA DE RENDERIZACIÓN
2570

FIG. 25

FIG. 26A FORMATO DE COMANDO DE PROCESADOR GRÁFICO

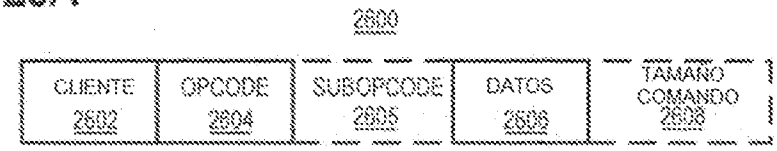
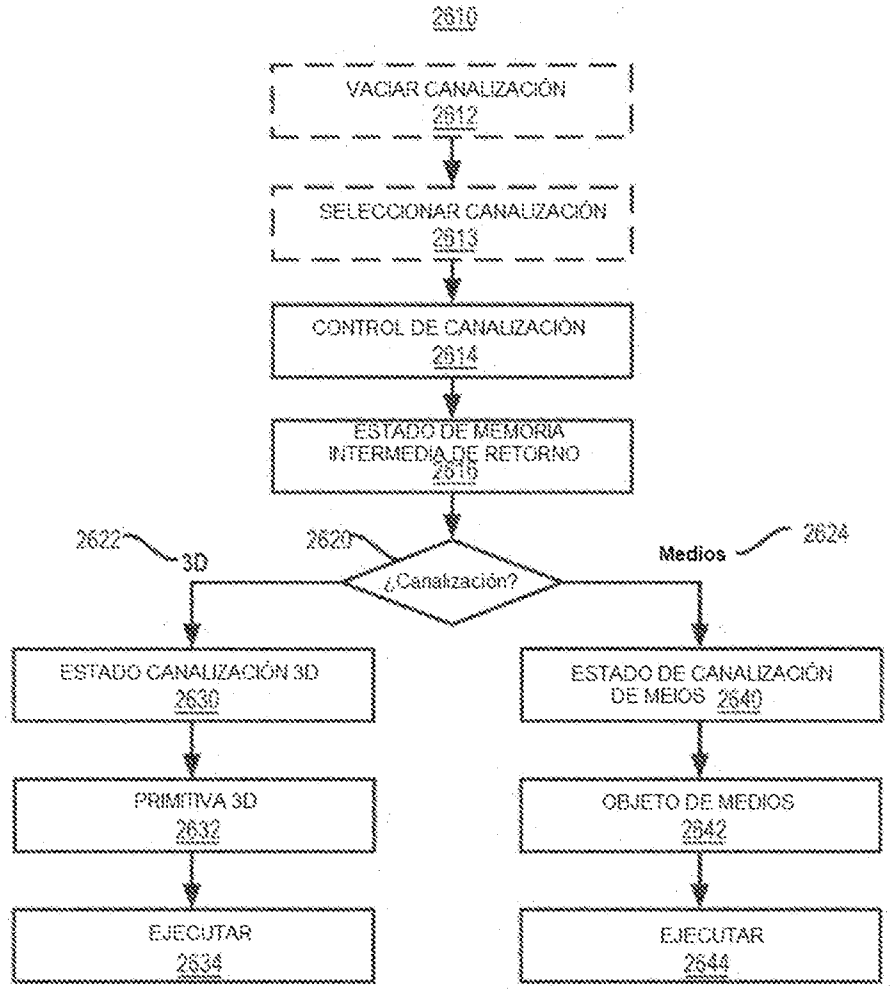


FIG. 26B SECUENCIA DE COMANDOS DE PROCESADOR GRÁFICO



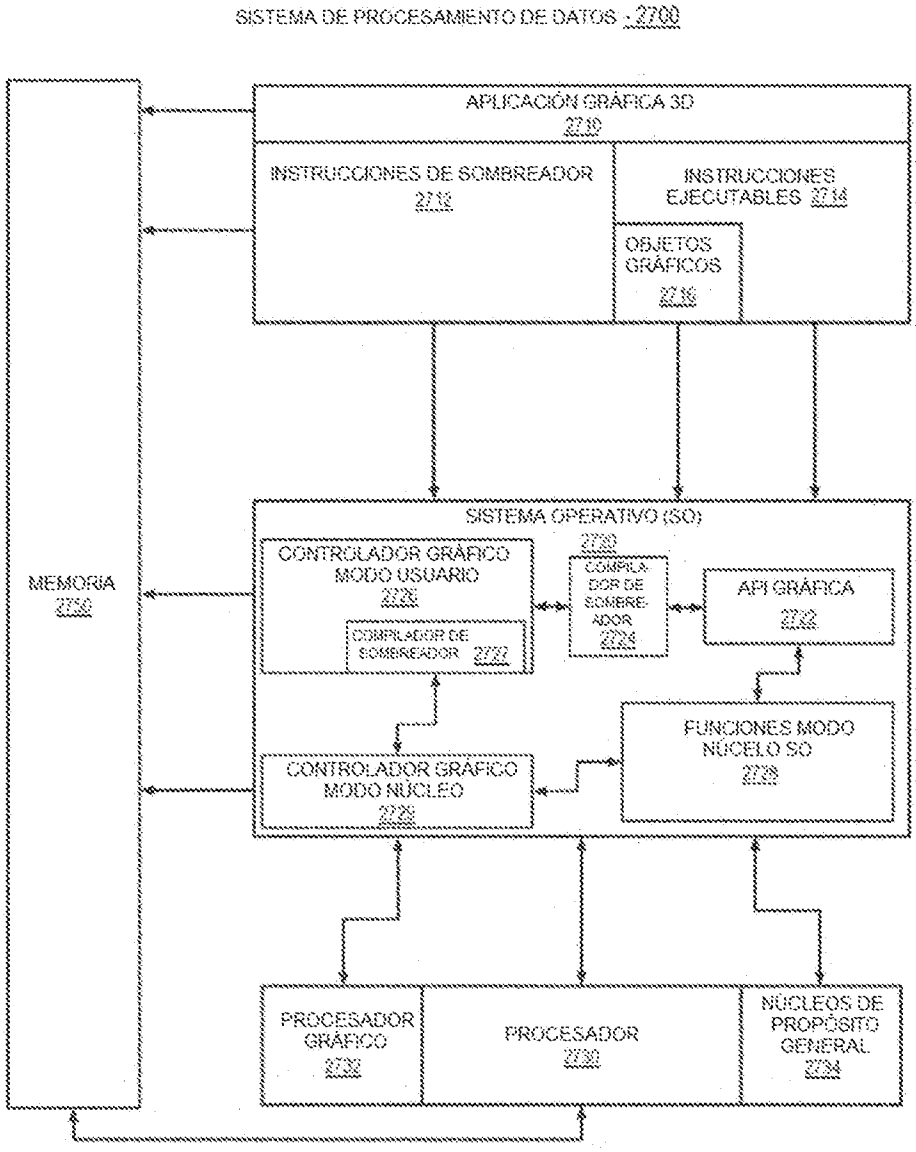

SISTEMA DE PROCESAMIENTO DE DATOS 2700
APLICACIÓN GRÁFICA 3D
2710
INSTRUCCIONES DE SOMBREADOR
2712
INSTRUCCIONES EJECUTABLES 2714
OBJETOS GRÁFICOS
2716
MEMORIA
2750
SISTEMA OPERATIVO (SO)
2720
CONTROLADOR GRÁFICO MODO USUARIO
2726
COMPILADOR DE SOMBREADOR 2722
COMPILADOR DE SOMBREADOR
2724
API GRÁFICA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

FIG. 27

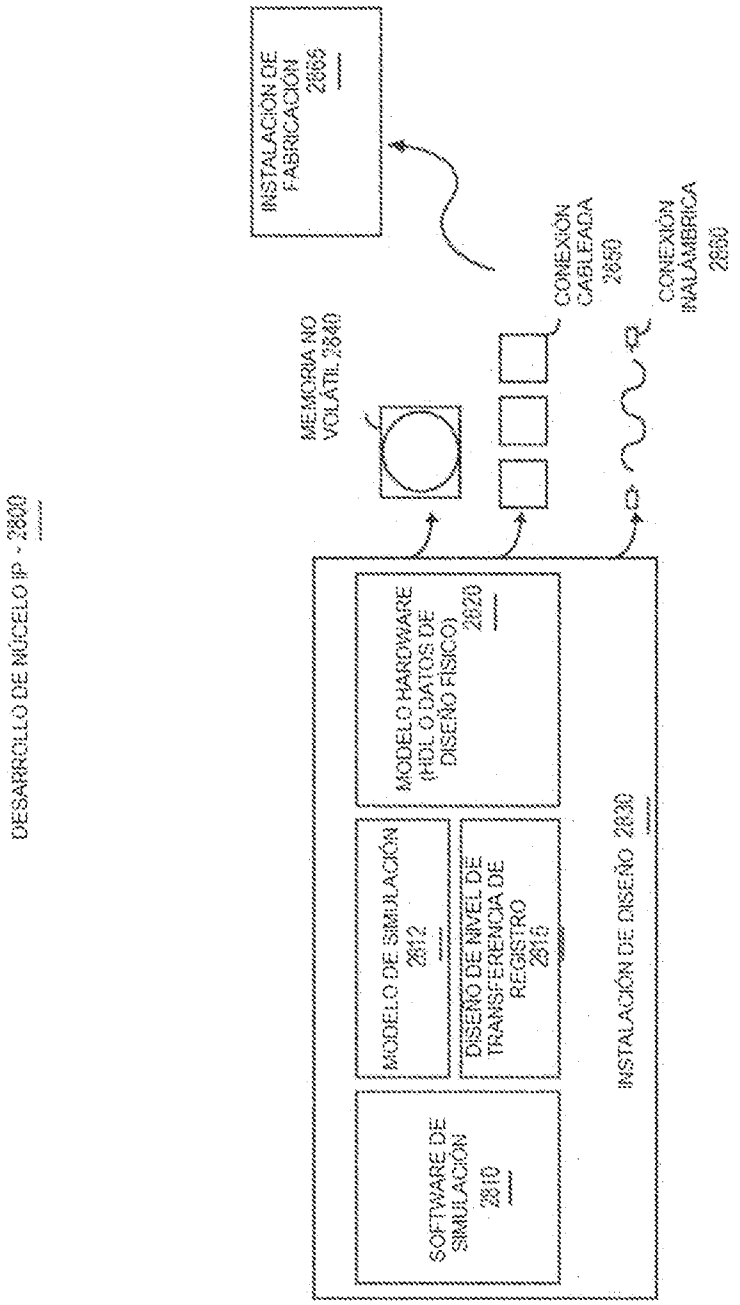

DESARROLLO DE NÚCLEO IP - 2800
INSTALACIÓN DE FABRICACIÓN 2885
MEMORIA NO VOLÁTIL 2840
CONEXIÓN CABLEADA 2850
CONEXIÓN INALÁMBRICA 2860
SOFTWARE DE SIMULACIÓN 2810
MODELO DE SIMULACIÓN 2812
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 2815
MODELO HARDWARE (HDL O DATOS DE DISEÑO FÍSICO) 2820
INSTALACIÓN DE DISEÑO 2830

FIG. 28

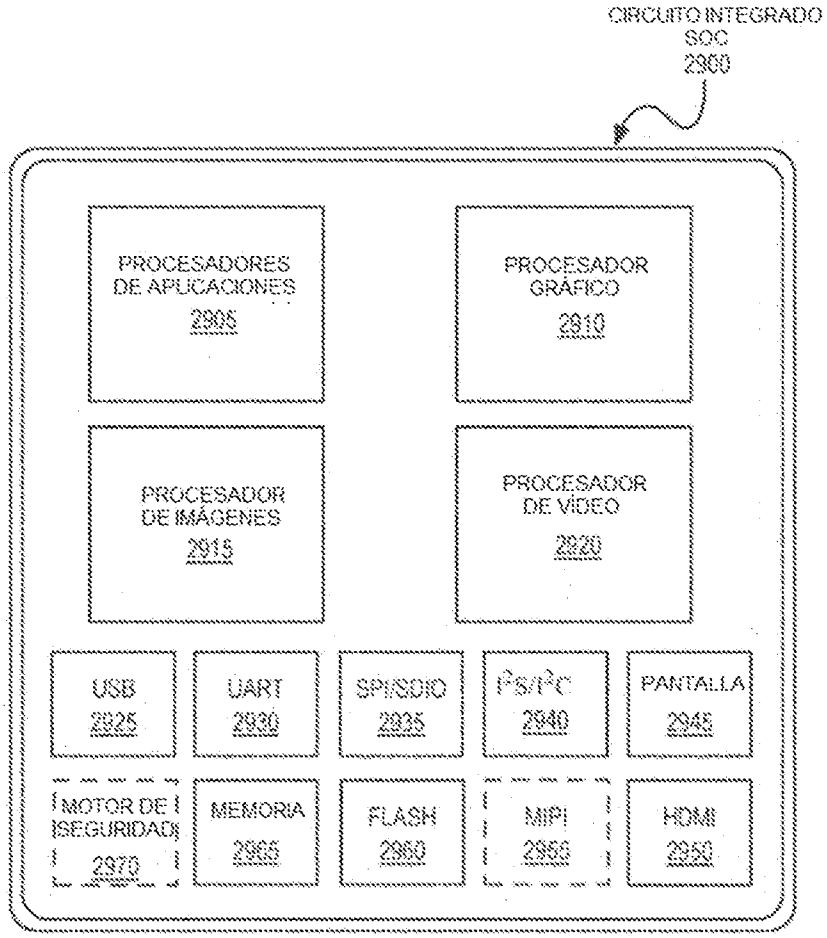

CIRCUITO INTEGRADO SOC 2900
PROCESADORES DE APLICACIONES 2905
PROCESADOR GRÁFICO 2910
PROCESADOR DE IMÁGENES 2915
PROCESADOR DE VIDEO 2920
USB 2925
UART 2930
SPI/SDIO 2935
PS/I²C 2940
PANTALLA 2945
MOTOR DE SEGURIDAD 2970
MEMORIA 2955
FLASH 2960
MIPI 2956
HDMI 2950

FIG. 29

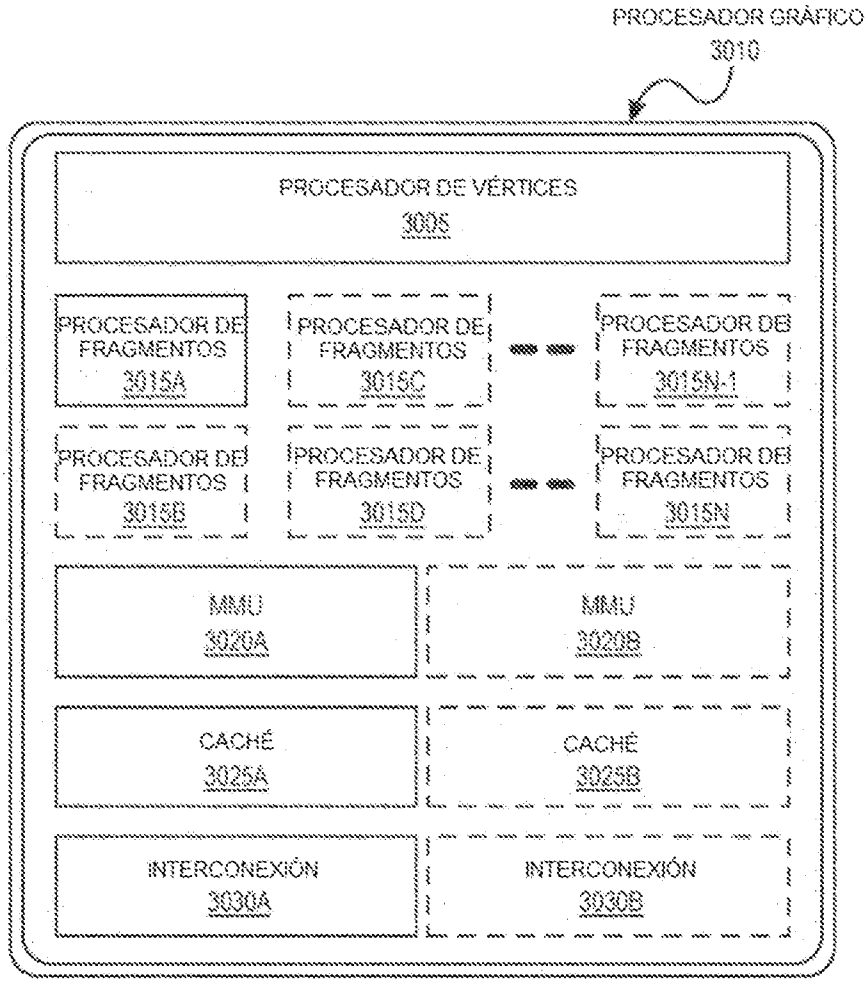

PROCESADOR GRÁFICO
3010
PROCESADOR DE VÉRTICES
3005
PROCESADOR DE FRAGMENTOS
3015A
PROCESADOR DE FRAGMENTOS
3015C
PROCESADOR DE FRAGMENTOS
3015N-1
PROCESADOR DE FRAGMENTOS
3015B
PROCESADOR DE FRAGMENTOS
3015D
PROCESADOR DE FRAGMENTOS
3015N
MMU
3020A
MMU
3020B
CACHÉ
3025A
CACHÉ
3025B
INTERCONEXIÓN
3030A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

FIG. 30

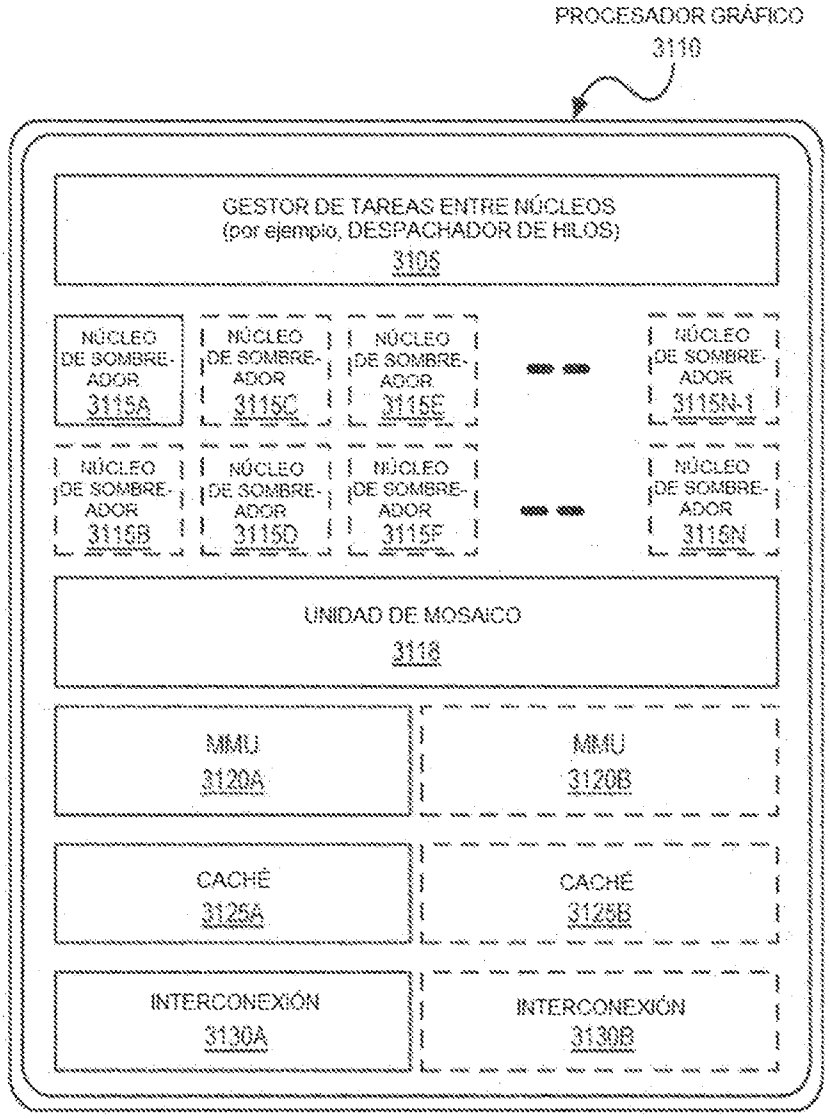

PROCESADOR GRÁFICO
3110
GESTOR DE TAREAS ENTRE NÚCLEOS
(por ejemplo, DESPACHADOR DE HILOS)
3105
NÚCLEO DE SOMBREADOR 3115A
NÚCLEO DE SOMBREADOR 3115C
NÚCLEO DE SOMBREADOR 3115E
NÚCLEO DE SOMBREADOR 3115N-1
NÚCLEO DE SOMBREADOR 3115B
NÚCLEO DE SOMBREADOR 3115D
NÚCLEO DE SOMBREADOR 3115F
NÚCLEO DE SOMBREADOR 3115N
UNIDAD DE MOSAICO
3118
MMU
3120A
MMU
3120B
CACHÉ
3125A
CACHÉ
3125B
INTERCONEXIÓN
3130A
INTERCONEXIÓN
3130B

FIG. 31