(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **3 025 359**

(51) Int. Cl.:

G06N 3/063 (2013.01)
G06N 3/0464 (2013.01)
G06N 3/045 (2013.01)
G06N 20/20 (2009.01)
G06V 10/764 (2012.01)
G06V 10/82 (2012.01)
G06V 10/44 (2012.01)
G06V 10/94 (2012.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **07.04.2017 PCT/CN2017/079682**

(87) Fecha y número de publicación internacional: **11.10.2018 WO18184194**

(96) Fecha de presentación y número de la solicitud europea: **07.04.2017 E 17904390 (6)**

(97) Fecha y número de publicación de la concesión europea: **12.02.2025 EP 3607488**

(54) Título: **Procedimientos y sistemas que utilizan redes neuronales convolucionales mejoradas para el procesamiento de imágenes**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.06.2025

(73) Titular/es:

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

(72) Inventor/es:

WANG, SHANDONG;
GUO, YIWEN;
YAO, ANBANG;
CAI, DONGQI;
WANG, LIBIN;
XU, LIN;
HU, PING;
CHENG, WENHUA y
CHEN, YURONG

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Procedimientos y sistemas que utilizan redes neuronales convolucionales mejoradas para el procesamiento de imágenes

SECTOR

Las realizaciones de la invención están en el sector del procesamiento de datos, incluido el procesamiento de imágenes, el procesamiento de gráficos y el aprendizaje automático. Más particularmente, las realizaciones de la invención se refieren a procedimientos y sistemas que utilizan redes neuronales convolucionales (CNN) mejoradas para el procesamiento de imágenes.

ESTADO DE LA TÉCNICA ANTERIOR

El procesamiento actual de datos gráficos en paralelo incluye sistemas y procedimientos 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 de gráficos utilizaban unidades computacionales de función fija para procesar datos de gráficos; sin embargo, más recientemente, partes de los procesadores de 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 de gráficos generalmente implementan técnicas de procesamiento tales como canalización, 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 de 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 en paralelo 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).

El aprendizaje automático ha logrado resolver 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 de 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 en paralelo 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 paralelo permite el uso de redes de alta capacidad y permite que esas redes se entrenen en conjuntos de datos más grandes.

Un tipo de red neuronal es la red neuronal convolucional (CNN), que puede realizar aprendizaje automático profundo. Los nodos de la capa de entrada de la CNN están organizados en un conjunto de "filtros" que pueden actuar como detectores de características. La salida de cada conjunto de filtros se propaga a los nodos en capas sucesivas de la red. Por lo tanto, la CNN es útil en aplicaciones de visión artificial y reconocimiento de imágenes debido a sus capacidades de detección de características. Sin embargo, el procesamiento de CNN puede requerir un gran esfuerzo computacional, ya que cada capa tiene una cantidad de nodos con una cantidad de parámetros a calcular para aplicaciones de detección y procesamiento de imágenes. Esto se vuelve aún más problemático si la potencia computacional y la memoria son limitadas en un dispositivo informático. Por lo tanto, lo que se necesita es una CNN mejorada, que pueda reducir el número de nodos y parámetros utilizados para el cálculo de la CNN.

El documento US 2016/321784 A1 da a conocer: Un procedimiento para reducir la resolución de la imagen en una red convolucional profunda (DCN) incluye la selección dinámica de un factor de reducción que se aplicará a una imagen de entrada. El factor de reducción se puede seleccionar en cada capa de la DCN. El procedimiento también incluye el ajuste de la DCN en función del factor de reducción seleccionado para cada capa.

CARACTERÍSTICAS

La invención se expone en el conjunto de reivindicaciones adjunto.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Los dibujos adjuntos ilustran ejemplos y son, por lo tanto, realizaciones ejemplares y no se consideran limitativos en su alcance.

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 este documento.

Las **Figuras 2A-2D** ilustran componentes de un procesador paralelo, de acuerdo con una realización ejemplar.

Las **Figuras 3A-3B** son diagramas de bloques de multiprocesadores gráficos según realizaciones ejemplares.

Las **Figuras 4A-4F** ilustran una arquitectura ejemplar en la que una pluralidad de Unidades de procesamiento de gráficos (GPU) están acopladas comunicativamente a una pluralidad de procesadores multinúcleo.

La **Figura 5** ilustra una canalización de procesamiento de gráficos, de acuerdo con una realización ejemplar.

La **Figura 6** ilustra una pila de software de aprendizaje automático según una realización ejemplar.

La **Figura 7** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela según una realización ejemplar.

La **Figura 8** ilustra un sistema informático de múltiples GPU según una realización ejemplar.

Las **Figuras 9A-9B** ilustran capas de redes neuronales profundas ejemplares, que no están cubiertas por la materia objeto de las reivindicaciones.

La **Figura 10** ilustra una red neuronal recurrente ejemplar, que no está cubierta por la materia objeto de las reivindicaciones. La **Figura 11** ilustra una realización ejemplar del entrenamiento y despliegue de una red neuronal profunda, que no está cubierta por la materia objeto de las reivindicaciones.

La **Figura 12** es un diagrama de bloques ejemplar que ilustra el aprendizaje distribuido, que no está cubierto por la materia objeto de las reivindicaciones.

La **Figura 13** ilustra un sistema de inferencia ejemplar en un chip (SOC) adecuado para realizar inferencias utilizando un modelo entrenado.

La **Figura 14** es un diagrama de bloques ejemplar de un sistema de procesamiento de imágenes que tiene un sistema de red neuronal convolucional (CNN) con una CNN mejorada para procesar una imagen de entrada.

Las **Figuras 15A-15B** ilustran un sistema de procesamiento de imágenes que procesa una imagen de entrada submuestreada utilizando una CNN mejorada con nodos de capa reducidos de acuerdo con realizaciones ejemplares.

La **Figura 16A** ilustra un diagrama de flujo ejemplar de una operación para procesar una imagen de entrada utilizando una CNN mejorada con nodos de capa reducidos de acuerdo con una realización ejemplar.

La **Figura 16B** ilustra un diagrama de flujo ejemplar de una operación para proporcionar salidas para una CNN mejorada utilizando nodos de capa reducidos de acuerdo con una realización ejemplar.

La **Figura 17** ilustra una CNN mejorada con redes CNN superficiales que imitan una red neuronal profunda según una realización ejemplar.

La **Figura 18A** es un diagrama de bloques ejemplar para generar una CNN mejorada con redes CNN superficiales que imitan una red neuronal profunda de acuerdo con una realización ejemplar, que no está cubierta por la materia objeto de las reivindicaciones.

La **Figura 18B** es un diagrama de flujo ejemplar para generar una CNN mejorada con redes CNN superficiales de acuerdo con una realización ejemplar, que no está cubierta por la materia objeto de las reivindicaciones.

La **Figura 19** ilustra un diagrama de bloques de un sistema de procesamiento según una realización ejemplar.

La **Figura 20** ilustra un diagrama de bloques ejemplar de una realización de un procesador que tiene uno o más núcleos de procesador, un controlador de memoria integrado y un procesador de gráficos integrado.

La **Figura 21** ilustra un diagrama de bloques ejemplar de un procesador de gráficos.

La **Figura 22** ilustra un diagrama de bloques de un motor de procesamiento de gráficos de un procesador de gráficos según realizaciones ejemplares.

La **Figura 23** ilustra un diagrama de bloques de otra realización ejemplar de un procesador de gráficos.

La **Figura 24** ilustra la lógica de ejecución de hilos que incluye una matriz de elementos de procesamiento empleados en realizaciones ejemplares de un motor de procesamiento de gráficos (GPE).

La **Figura 25** ilustra un diagrama de bloques de formatos de instrucciones de un procesador de gráficos según realizaciones ejemplares.

La **Figura 26** ilustra un diagrama de bloques de una realización ejemplar de un procesador de gráficos.

La **Figura 27A** ilustra un diagrama de bloques de un formato de comando de procesador de gráficos según una realización ejemplar.

La **Figura 27B** ilustra un diagrama de bloques de una secuencia de comandos de procesador de gráficos según una realización ejemplar.

La **Figura 28** ilustra una arquitectura de software de gráficos ejemplar para un sistema de procesamiento de datos según realizaciones ejemplares.

La **Figura 29** ilustra un diagrama de bloques de un sistema de desarrollo de núcleo PI que puede usarse para fabricar un circuito integrado (CI) para realizar operaciones de acuerdo con una realización ejemplar.

La **Figura 30** ilustra un diagrama de bloques de un sistema ejemplar en un circuito integrado de chip que puede fabricarse utilizando uno o más núcleos PI de acuerdo con una realización ejemplar.

La **Figura 31** ilustra un diagrama de bloques de un procesador de gráficos ejemplar en un circuito integrado de sistema en chip que puede fabricarse utilizando uno o más núcleos PI de acuerdo con una realización ejemplar.

La **Figura 32** ilustra un diagrama de bloques de un procesador de gráficos adicional ejemplar de un circuito integrado de sistema en chip que puede fabricarse utilizando uno o más núcleos PI de acuerdo con una realización ejemplar.

DESCRIPCIÓN DETALLADA

En algunas realizaciones, una unidad de procesamiento de gráficos (GPU) está acoplada de manera comunicativa a núcleos de anfitrión/de procesador para acelerar las operaciones de gráficos, las operaciones de aprendizaje automático, las operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede estar acoplada comunicativamente al procesador/núcleos anfitriones 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 del procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU luego utiliza circuitos/lógica dedicados para procesar de manera eficiente estos comandos/instrucciones.

En algunas realizaciones, un dispositivo de captura de imágenes es un dispositivo independiente para capturar una imagen de entrada. Sin embargo, el dispositivo de captura de imágenes puede ser parte o un subcomponente de otro dispositivo informático que requiera capacidades de captura de imágenes, tal como un dispositivo informático portátil o de mano con una cámara digital para capturar imágenes.

En la siguiente descripción se exponen numerosos detalles específicos para proporcionar una comprensión más completa. Sin embargo, será evidente que las realizaciones descritas en este documento pueden llevarse a la práctica sin uno o más de estos detalles específicos. En otros casos, no se han descrito características bien conocidas para evitar oscurecer los detalles de las realizaciones ejemplares.

Descripción general del sistema informático

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático 100 configurado para implementar uno o más aspectos de las realizaciones ejemplares descritas en este 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 del 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 la entrada de uno o más dispositivos de entrada 108. Además, el concentrador de E/S 107 puede permitir que un controlador de pantalla, que puede estar incluido en el uno o más procesadores 102, proporcione 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 una de cualquier cantidad 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 específicas del proveedor. En una realización, el uno o más procesadores paralelos 112 forman un sistema de procesamiento paralelo o vectorial centrado en la computación que puede incluir una gran cantidad de núcleos de procesamiento y/o agrupaciones de procesamiento, tales como un procesador de núcleos integrados múltiples (MIC). En una realización, el uno o más procesadores paralelos 112 forman un subsistema de procesamiento de gráficos que puede enviar píxeles a uno del uno o más dispositivos de visualización 110A acoplados a través del concentrador de E/S 107. El uno o más procesadores paralelos 112 también pueden incluir un controlador de visualización y una interfaz de visualización (no se muestra) para permitir una conexión directa a uno o más dispositivos de visualización 110B.

Dentro del subsistema de E/S 111, una unidad de almacenamiento del 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 que permita conexiones entre el concentrador de E/S 107 y otros componentes, tales como un adaptador de red 118 y/o un adaptador de red inalámbrico 119 que se pueden integrar en la plataforma, y varios otros dispositivos que se pueden agregar 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 inalámbrico 119 puede incluir uno o más dispositivos de red Wi-Fi, Bluetooth, de 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 USB u otros puertos, unidades de almacenamiento óptico, dispositivos de captura de vídeo y similares, que también pueden estar conectados al concentrador de E/S 107. Las rutas de comunicación que interconectan los diversos componentes de la Figura 1 pueden implementarse 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, tales 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 de gráficos y vídeo, incluyendo, por ejemplo, circuitos de salida de vídeo, y constituyen una unidad de procesamiento de gráficos (GPU). En otra realización, el uno o más procesadores paralelos 112 incorporan circuitos optimizados para procesamiento de propósito general, mientras se conserva la arquitectura computacional subyacente, descrita con mayor detalle en este documento. En otra realización más, los componentes del sistema informático 100 pueden integrarse con uno o más elementos del sistema en un único circuito integrado. Por ejemplo, el uno o más procesadores paralelos 112, el concentrador de memoria 105, el procesador o procesadores 102 y el concentrador de E/S 107 pueden integrarse en un circuito integrado de sistema en chip (SoC). Alternativamente, los componentes del sistema informático 100 pueden integrarse en un único paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una parte de los componentes del sistema informático 100 puede integrarse en un módulo multichip (MCM), que puede interconectarse con otros módulos multichip en un sistema informático modular.

Se apreciará que el sistema informático 100 que se muestra en este documento es ilustrativo y que son posibles variaciones y modificaciones. La topología de conexión, que incluye 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 del sistema 104 está conectada al procesador o procesadores 102 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria del sistema 104 a través del concentrador de memoria 105 y el procesador o procesadores 102. En otras topologías alternativas, el procesador o procesadores paralelos 112 están conectados al concentrador de E/S 107 o directamente a uno del 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 conectores, que pueden acoplarse con dos o más instancias del procesador o procesadores paralelos 112.

Algunos de los componentes particulares que se muestran en este documento son opcionales y pueden no estar incluidos en todas las implementaciones del sistema informático 100. Por ejemplo, se puede admitir cualquier número de tarjetas o periféricos complementarios, 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 Northbridge en algunas arquitecturas, mientras que el concentrador de E/S 107 puede denominarse Southbridge.

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

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 estar conectada directamente a otros dispositivos. En una realización, la unidad de E/S 204 se conecta con otros dispositivos mediante el uso de un concentrador o una interfaz de conmutación, 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 de anfitrión 206 y una barra transversal de memoria 216, donde la interfaz de anfitrión 206 recibe comandos dirigidos a realizar operaciones de procesamiento y la barra transversal de memoria 216 recibe comandos dirigidos a realizar operaciones de memoria.

Cuando la interfaz de anfitrión 206 recibe una memoria intermedia de comandos a través de la unidad de E/S 204, la interfaz de anfitrión 206 puede dirigir las operaciones de trabajo para ejecutar esos comandos a un extremo frontal 208. En una realización, el extremo frontal 208 se acopla con un planificador 210, que está configurado para distribuir comandos u otros elementos de trabajo a una matriz de agrupaciones de procesamiento 212. En una realización, el planificador 210 garantiza que la matriz de agrupaciones de procesamiento 212 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. En una realización, el planificador 210 se implementa a través de una lógica de firmware que se ejecuta en un microcontrolador. El planificador 210 implementado en el microcontrolador se puede configurar para realizar operaciones complejas de planificación y distribución de trabajo con granularidad gruesa y fina, lo que permite una rápida interrupción y cambio de contexto de los hilos que se ejecutan en la matriz de procesamiento 212. En una realización, el software anfitrión puede proporcionar cargas de trabajo para la planificación en la matriz de procesamiento 212 a través de uno de los múltiples timbres de procesamiento de gráficos. Las cargas de trabajo se pueden distribuir automáticamente a través de la matriz de procesamiento 212 mediante la lógica del planificador 210 dentro del microcontrolador del planificador.

La matriz de agrupaciones de procesamiento 212 puede incluir hasta "N" agrupaciones de procesamiento (por ejemplo, la agrupación 214A, la 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 la ejecución por la matriz de agrupaciones de procesamiento 212. En una realización, diferentes agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 pueden asignarse para procesar diferentes tipos de programas o para realizar diferentes tipos de cálculos.

La matriz de agrupaciones de procesamiento 212 puede configurarse para realizar varios tipos de operaciones de procesamiento en paralelo. En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de cálculo en 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 de física, y la realización de transformaciones de datos.

En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de procesamiento de gráficos en paralelo. En realizaciones en las que el procesador paralelo 200 está configurado para realizar operaciones de procesamiento de gráficos, la matriz de agrupaciones de procesamiento 212 puede incluir lógica adicional para soportar la ejecución de dichas operaciones de procesamiento de gráficos, incluyendo, pero no limitado a, lógica de muestreo de textura para realizar operaciones de textura, así como lógica de teselación y otra lógica de procesamiento de vértices. Adicionalmente, la matriz de agrupaciones de procesamiento 212 puede configurarse para ejecutar programas de sombreado relacionados con el procesamiento de gráficos tales como, de forma no limitativa, 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 del sistema a través de la unidad de E/S 204 para su procesamiento. Durante el procesamiento, los datos transferidos pueden almacenarse en la memoria en chip (por ejemplo, la memoria del procesador paralelo 222) durante el procesamiento, y luego escribirse nuevamente en la memoria del sistema.

En una realización, cuando la unidad de procesamiento paralelo 202 se utiliza para realizar el procesamiento de gráficos, el planificador 210 puede configurarse para dividir la carga de trabajo de procesamiento en tareas de tamaño aproximadamente igual, para permitir mejor la distribución de las operaciones de procesamiento de gráficos a múltiples agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212. En algunas realizaciones, partes de la matriz de agrupaciones de procesamiento 212 pueden configurarse para realizar diferentes tipos de procesamiento. Por ejemplo, una primera parte puede configurarse para realizar sombreado de vértices y generación de topología, una segunda parte puede configurarse para realizar teselación y sombreado de geometría, y una tercera parte puede configurarse para realizar sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen renderizada para visualización. Los datos intermedios producidos por una 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 las tareas de procesamiento desde el extremo frontal 208. Para las operaciones de procesamiento de gráficos, las tareas de procesamiento pueden incluir índices de datos a procesar, por ejemplo, datos de superficie (parche), datos de primitiva, datos de vértice y/o datos de 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 buscar los índices correspondientes a las tareas o puede recibir los índices desde el extremo frontal 208. El extremo frontal 208 se puede configurar para garantizar que la matriz de agrupaciones de procesamiento 212 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 lotes, memorias intermedias de inserción, etc.).

Cada una de la una o más instancias de la unidad de procesamiento paralelo 202 puede acoplarse con la memoria del procesador paralelo 222. Se puede acceder a la memoria del 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 del 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 parte (por ejemplo, unidad de memoria) de la memoria del 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 varias realizaciones, las unidades de memoria 224A-224N pueden incluir varios tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámico (DRAM) o memoria de acceso aleatorio de gráficos, tales como memoria de acceso aleatorio de gráficos sincrónicos (SGRAM), incluyendo memoria de doble velocidad de datos de gráficos (GDDR). En una realización, las unidades de memoria 224A-224N también pueden incluir memoria apilada 3D, incluyendo, 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 de uno de varios diseños convencionales. Los objetivos de renderizado, tales como las memorias intermedias de fotogramas o los mapas de textura, se pueden almacenar en las unidades de memoria 224A-224N, lo que permite que las unidades de partición 220A-220N escriban partes 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 a favor de un diseño de memoria unificada que utiliza la memoria del 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 agrupación 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 en paralelo 222, lo que permite que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 214A a 214N se comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento en 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 única 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 números de núcleos de procesamiento, diferentes cantidades de memoria de procesador paralelo local y/u otras diferencias de configuración. Por ejemplo y en una realización, algunas instancias de la unidad de procesamiento paralelo 202 pueden incluir unidades de 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 el procesador paralelo 200 se pueden implementar en una variedad de configuraciones y factores de forma, incluyendo, de forma no limitativa, ordenadores 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 ejemplar. 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 memoria caché L2 221, una interfaz de memoria intermedia de fotogramas 225 y una ROP 226 (unidad de operaciones de rasterización). La memoria caché L2, 221, es una memoria caché de lectura/escritura que está configurada para realizar operaciones de carga y de almacenamiento recibidas desde la barra transversal de memoria 216 y la ROP 226. Los fallos de lectura y las solicitudes de escritura urgentes son enviadas por la memoria caché L2 221 a la interfaz de memoria intermedia de fotogramas 225 para su procesamiento. Las actualizaciones también se pueden enviar a la memoria intermedia de fotogramas a través de la interfaz de memoria intermedia de fotogramas 225 para su procesamiento. En una realización, la interfaz de memoria intermedia de fotogramas 225 interactúa con una de las unidades de memoria en la memoria del procesador paralelo, tales como las unidades de memoria 224A-224N de la **Figura 2A** (por ejemplo, dentro de la memoria del procesador paralelo 222).

En aplicaciones gráficas, la ROP 226 es una unidad de procesamiento que realiza operaciones de rasterización tales como estarcido, prueba z, combinación y similares. La ROP 226 luego genera datos gráficos procesados que se almacenan en la memoria gráfica. En algunas realizaciones, la ROP 226 incluye una lógica de compresión para comprimir datos de profundidad o de color que se escriben en la memoria y descomprimir datos de profundidad o de color que se leen desde la memoria. La lógica de compresión puede ser lógica de compresión sin pérdida que hace uso de uno o más de múltiples algoritmos de compresión. El tipo de compresión que realiza la ROP 226 puede variar en función de las características estadísticas de los datos que se van a comprimir. Por ejemplo, en una realización, la compresión de color delta se realiza en datos de profundidad y color sobre una base por mosaico.

En algunas realizaciones, la ROP 226 está incluida dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 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 visualizarse en un dispositivo de visualización, tal como uno del uno o más dispositivos de visualización 110 de la Figura 1, enrutarse para su posterior procesamiento por el procesador o procesadores 102, o enrutarse para su posterior procesamiento 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 ejemplar. En una realización, la agrupación de procesamiento es una instancia de una de las agrupaciones de procesamiento 214A-214N de la **Figura 2A**. La agrupación de procesamiento 214 puede configurarse para ejecutar muchos hilos en paralelo, donde el término "hilo" se refiere a una instancia de un programa particular que se ejecuta en un conjunto particular de datos de entrada. En algunas realizaciones, se utilizan técnicas de emisión de instrucciones de instrucción única, múltiples datos (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, múltiples hilos (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 normalmente ejecutan instrucciones idénticas, la ejecución SIMT permite que diferentes hilos sigan más fácilmente rutas de ejecución divergentes a través de un programa de hilos determinado. Los expertos en la materia entenderán que un régimen de procesamiento SIMD representa un subconjunto funcional de un régimen de procesamiento SIMT.

El funcionamiento de las agrupaciones de procesamiento 214 se puede controlar a través de un administrador de canalizaciones 232 que distribuye tareas de procesamiento a procesadores paralelos SIMT. El administrador de canalizaciones 232 recibe instrucciones del planificador 210 de la **Figura 2A** y gestiona la ejecución de esas instrucciones a través de un multiprocesador de gráficos 234 y/o una unidad de textura 236. El multiprocesador de gráficos 234 ilustrado es una instancia ejemplar de un procesador paralelo SIMT. Sin embargo, se pueden incluir varios tipos de procesadores paralelos SIMT de diferentes arquitecturas dentro de las agrupaciones de procesamiento 214. Se pueden incluir una o más instancias del multiprocesador de gráficos 234 dentro de una agrupación de procesamiento 214. El multiprocesador de gráficos 234 puede procesar datos y se puede utilizar una barra transversal de datos 240 para distribuir los datos procesados a uno de los múltiples destinos posibles, incluidas otras unidades de sombreado. El gestor de canales 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 de gráficos 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-almacenamiento, etc.). La lógica de ejecución funcional se puede configurar de manera canalizada en la que se pueden emitir nuevas instrucciones antes de que se completen las instrucciones anteriores. La lógica de ejecución funcional admite una variedad de operaciones que incluyen aritmética de números enteros y de punto flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y cálculo de varias 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 puede asignarse a un motor de procesamiento diferente dentro de un multiprocesador de gráficos 234. Un grupo de hilos puede incluir menos hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando un grupo de hilos incluye menos hilos que el número de motores de procesamiento, uno o más de los motores de procesamiento pueden estar inactivos durante los ciclos en los que se procesa ese grupo de hilos. Un grupo de hilos también puede incluir más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando el grupo de hilos incluye más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234, se puede realizar el procesamiento a través de ciclos de reloj consecutivos. En una realización, se pueden ejecutar múltiples grupos de hilos simultáneamente en un multiprocesador de gráficos 234.

En una realización, el multiprocesador de gráficos 234 incluye una memoria caché interna para realizar operaciones de carga y almacenamiento. En una realización, el multiprocesador de gráficos 234 puede prescindir de una memoria caché interna y utilizar una memoria caché (por ejemplo, la memoria caché L1 248) dentro de la agrupación de procesamiento 214. Cada multiprocesador de gráficos 234 también tiene acceso a las memorias caché L2 dentro de las unidades de partición (por ejemplo, las unidades de partición 220A-220N de la **Figura 2A**) que se comparten entre todas las agrupaciones de procesamiento 214 y que pueden utilizarse para transferir datos entre hilos. El multiprocesador de gráficos 234 también puede acceder a la memoria global fuera del chip, que puede incluir una o más de las memorias de procesador paralelo local y/o memoria del 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 de gráficos 234 pueden compartir instrucciones y datos comunes, que pueden almacenarse en la memoria caché L1 248.

Cada agrupación de procesamiento 214 puede incluir una MMU 245 (unidad de administración de memoria) que está configurada para mapear direcciones virtuales en direcciones físicas. En otras realizaciones, una o más instancias de la MMU 245 pueden residir dentro de la interfaz de memoria 218 de la Figura 2. La MMU 245 incluye un conjunto de entradas de tabla de páginas (PTE) utilizadas para mapear una dirección virtual a una dirección física de un mosaico (más información sobre el mosaico) y, opcionalmente, un índice de línea de caché. La MMU 245 puede incluir memorias intermedias de búsqueda de traducción de direcciones (TLB) o memorias caché que pueden residir dentro del multiprocesador de gráficos 234 o la memoria 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 de solicitudes eficiente entre 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 se puede configurar de manera que cada multiprocesador de gráficos 234 esté acoplado a una unidad de textura 236 para realizar operaciones de mapeo de textura, por ejemplo, determinar posiciones de muestreo de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen desde una memoria caché L1 de textura interna (no se muestra) o en algunas realizaciones desde la memoria caché L1 dentro del multiprocesador de gráficos 234 y se obtienen de una memoria caché L2, la memoria del procesador paralelo local o la memoria del sistema, según sea necesario. Cada multiprocesador de gráficos 234 envía las 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 del sistema a través de la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones de prerasterización) está configurada para recibir datos del multiprocesador de gráficos 234, dirigir los datos a las unidades ROP, que pueden estar ubicadas con unidades de partición como se describe en este documento (por ejemplo, las unidades de partición 220A-220N de la **Figura 2A**). La unidad preROP 242 puede realizar optimizaciones para la combinación de colores, organizar los datos de color de los píxeles y realizar traducciones de direcciones.

Se apreciará que la arquitectura central descrita en este documento es ilustrativa y que son posibles variaciones y modificaciones. Cualquier número de unidades de procesamiento, por ejemplo, multiprocesador de gráficos 234, unidades de textura 236, preROP 242, etc., pueden incluirse 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 este documento se puede incluir cualquier número de instancias de las agrupaciones de procesamiento 214. En una realización, cada agrupación de procesamiento 214 puede configurarse para funcionar de forma independiente de otras agrupaciones de procesamiento 214 utilizando unidades de procesamiento separadas y distintas, memorias caché L1, etc.

La **Figura 2D** muestra un multiprocesador de gráficos 234, según una realización ejemplar. 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 acoplados 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 administrador de canalizaciones 232. Las instrucciones se almacenan en memoria caché en la memoria caché de instrucciones 252 y se envían para su ejecución por la unidad de instrucciones 254. La unidad de instrucciones 254 puede enviar 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 cualquiera de un espacio de direcciones local, compartido o global especificando una dirección dentro de un espacio de direcciones unificado. La unidad de mapeo de direcciones 256 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 registros 258 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de gráficos 234. El archivo de registros 258 proporciona almacenamiento temporal para operandos conectados a las rutas de datos de las unidades funcionales (por ejemplo, núcleos GPGPU 262, unidades de carga/almacenamiento 266) del multiprocesador de gráficos 234. En una realización, el archivo de registros 258 se divide entre cada una de las unidades funcionales de modo que a cada unidad funcional se le asigna una parte dedicada del archivo de registros 258. En una realización, el archivo de registros 258 se divide entre los diferentes *warps* que ejecuta el multiprocesador de gráficos 234.

Los núcleos GPGPU 262 pueden incluir cada uno unidades de punto flotante (FPU) y/o unidades de lógica aritmética de enteros (ALU) que se utilizan para ejecutar instrucciones de los gráficos multiprocesador 234. Los núcleos GPGPU 262 pueden ser similares en arquitectura o pueden diferir en arquitectura, según las realizaciones. Por ejemplo y en una realización, una primera parte de los núcleos GPGPU 262 incluye una FPU de precisión simple y una ALU de números enteros, mientras que una segunda parte 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 de gráficos 234 puede incluir adicionalmente una o más unidades de función fija o de función especial para realizar funciones específicas, tales como operaciones de copia de rectángulo o fusión de píxeles. En una realización, uno o más de los núcleos GPGPU también pueden incluir lógica de función fija o especial.

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

La interconexión de memoria y memoria caché 268 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador de gráficos 234 al archivo de registros 258 y a la memoria compartida 270. En una realización, la interconexión de memoria y memoria 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 registros 258. El archivo de registros 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 registros 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 de gráficos 234. La memoria caché 272 se puede utilizar como memoria caché de datos, por ejemplo, para almacenar en memoria caché los datos de textura comunicados entre las unidades funcionales y la unidad de textura 236. La memoria compartida 270 también se puede utilizar como memoria caché gestionada por programas. Los hilos que se ejecutan en los núcleos GPGPU 262 pueden almacenar programáticamente datos dentro de la memoria compartida, además de los datos almacenados automáticamente en memoria caché que se almacenan dentro de la memoria caché 272.

Las Figuras 3A-3B ilustran multiprocesadores de gráficos adicionales, según las realizaciones ejemplares. Los multiprocesadores de gráficos 325, 350 ilustrados son variantes del multiprocesador de gráficos 234 de la **Figura 2C**. Los multiprocesadores de gráficos 325, 350 ilustrados pueden configurarse como un multiprocesador de transmisión (SM) capaz de ejecutar simultáneamente un gran número de hilos de ejecución.

La **Figura 3A** muestra un multiprocesador de gráficos 325 de acuerdo con una realización adicional ejemplar. El multiprocesador de gráficos 325 incluye múltiples instancias adicionales de unidades de recurso de ejecución relativas al multiprocesador de gráficos 234 de la **Figura 2C-2D**. Por ejemplo, el multiprocesador de gráficos 325 puede incluir múltiples instancias de la unidad de instrucciones 332A-332B, archivo de registros 334A-334B y unidad(es) de textura

344A-344B. El multiprocesador de gráficos 325 también incluye múltiples conjuntos de unidades de ejecución de gráficos o de cálculo (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 memoria caché de instrucciones común 330, una memoria caché de texturas 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 de gráficos 325. En una realización, la estructura de interconexión 327 es una capa de estructura de red de alta velocidad independiente sobre la que se apila cada componente del multiprocesador de gráficos 325. Los componentes del multiprocesador de gráficos 325 se comunican con componentes remotos mediante la estructura de interconexión 327. Por ejemplo, los núcleos GPGPU 336A-336B, 337A-337B y 338A-338B pueden comunicarse cada uno con la memoria compartida 346 a través de la estructura de interconexión 327. La estructura de interconexión 327 puede arbitrar la comunicación dentro del multiprocesador de gráficos 325 para asegurar una asignación justa de ancho de banda entre los componentes.

La **Figura 3B** muestra un multiprocesador de gráficos 350 de acuerdo con una realización adicional ejemplar. El procesador de gráficos incluye múltiples conjuntos de recursos de ejecución 356A-356D, donde cada conjunto de recursos de ejecución incluye múltiples unidades de instrucciones, archivos de registros, núcleos GPGPU y unidades de almacenamiento y carga, como se ilustra en la **Figura 2D** y la **Figura 3A**. Los recursos de ejecución 356A-356D pueden trabajar en conjunto con la(s) unidad(es) de textura 360A-360D para operaciones de textura, mientras comparten una memoria caché de instrucciones 354 y una memoria compartida 362. En una realización, los recursos de ejecución 356A a 356D pueden compartir una memoria 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 mediante una estructura de interconexión 352 similar a la estructura de interconexión 327 de la **Figura 3A**.

Los expertos en la materia entenderán que la arquitectura descrita en las **Figuras 1, 2A-2D** y **3A-3B** son descriptivas y no limitativas en cuanto al alcance de las presentes realizaciones, que son ejemplares. Por lo tanto, las técnicas descritas en este documento pueden implementarse 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 multinúcleo, 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 de gráficos o unidades de procesamiento de propósito especial, sin apartarse del alcance de las realizaciones descritas en este documento.

En algunas realizaciones, un procesador paralelo o GPGPU como se describe en este documento está acoplado comunicativamente a núcleos de anfitrión/procesador para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrones y varias 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 del procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU luego utiliza circuitos/lógica dedicados para procesar de manera eficiente estos comandos/instrucciones.

Técnicas para la interconexión de GPU a 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, 441A, 442 y 443 (denominados enlaces de alta velocidad 440-443) (por ejemplo, buses, interconexiones punto a punto, etc.). En una realización, los enlaces de alta velocidad 440-443 admiten un rendimiento de comunicación de 4 GB/s, 30 GB/s, 80 GB/s o superior, según la implementación. Se pueden utilizar varios 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 se limitan a ningún protocolo de comunicación o rendimiento en particular.

Además, y 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 del enlace de alta velocidad 433, que puede ser un bus multiprocesador simétrico (SMP) que funciona 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 llevarse a cabo utilizando los mismos protocolos/enlaces (por ejemplo, sobre una estructura de interconexión común). Sin embargo, como se mencionó, los principios subyacentes de la invención no se limitan a ningún 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 GPU 420-423 a través de interconexiones de memoria GPU 450-453, respectivamente. Las interconexiones de memoria 430-431 y 450-453 pueden utilizar las mismas o diferentes tecnologías de acceso a la memoria. A modo de ejemplo, y sin carácter limitativo, las memorias de procesador 401-402 y las memorias de GPU 420-423 pueden ser memorias volátiles, tales como memorias de acceso aleatorio dinámico (DRAM) (incluidas las DRAM apiladas), memorias gráficas DDR SDRAM (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 parte de las memorias puede ser memoria volátil y otra parte 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 las GPU 410-413 pueden estar acoplados físicamente a una memoria particular 401-402, 420-423, respectivamente, se puede implementar una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtual (también denominado espacio de "direcciones efectivas") 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 del sistema y las memorias de GPU 420-423 pueden comprender cada una 32 GB del espacio de direcciones de memoria del 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 de gráficos 446 de acuerdo con una realización ejemplar. El módulo de aceleración de gráficos 446 puede incluir uno o más chips de GPU integrados en una tarjeta de línea que está acoplada al procesador 407 a través del enlace de alta velocidad 440. Alternativamente, el módulo de aceleración de gráficos 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 memorias caché 462A-462D. Los núcleos pueden incluir varios otros componentes para ejecutar instrucciones y procesar datos que no se ilustran para evitar oscurecer los principios subyacentes de la invención (por ejemplo, unidades de extracción de instrucciones, unidades de predicción de bifurcaciones, decodificadores, unidades de ejecución, memorias intermedias de reordenamiento, etc.). Las memorias caché 462A-462D pueden comprender memorias caché de nivel 1 (L1) y nivel 2 (L2). Además, pueden incluirse una o más memorias caché 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 memoria caché L1, doce memorias caché L2 compartidos y doce memorias caché L3 compartidas. En esta realización, una de las memorias caché L2 y L3 son compartidas por dos núcleos adyacentes. El procesador 407 y el módulo de integración del acelerador de gráficos 446 se conectan con la memoria del sistema 441, que puede incluir las memorias del procesador 401-402.

La coherencia se mantiene para los datos e instrucciones almacenados en las diversas memorias caché 462A-462D, 456 y la memoria del sistema 441 a través de la comunicación entre núcleos a través de un bus de coherencia 464. Por ejemplo, cada memoria caché puede tener lógica/circuitos de coherencia de memoria caché asociados 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 espionaje de caché a través del bus de coherencia 464 para espiar los accesos a la memoria caché. Las técnicas de espionaje de caché/coherencia son bien entendidas por los 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 de gráficos 446 al bus de coherencia 464, lo que permite que el módulo de aceleración de gráficos 446 participe en el protocolo de coherencia de caché como un par de los núcleos. En particular, una interfaz 435 proporciona conectividad al circuito proxy 425 a través de un enlace de alta velocidad 440 (por ejemplo, un bus PCIe, NVLink, etc.) y una interfaz 437 conecta el módulo de aceleración de gráficos 446 al enlace 440.

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

En una realización, el circuito de integración del acelerador 436 incluye una unidad de administración 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 del sistema 441. La MMU 439 también puede incluir una memoria intermedia de búsqueda de traducción (TLB) (no mostrada) para almacenar en memoria 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 de gráficos 431-432, N. En una realización, los datos almacenados en la memoria caché 438 y las memorias de gráficos 433-434, N se mantienen coherentes con las memorias caché centrales 462A-462D, 456 y la memoria del sistema 441. Como se mencionó, esto se puede lograr a través del circuito proxy 425 que participa en el mecanismo de coherencia de la memoria caché en nombre de la memoria caché 438 y las memorias 433-434, N (por ejemplo, enviando actualizaciones a la memoria caché 438 relacionadas con modificaciones/accesos de líneas de memoria caché en las memorias caché del procesador 462A-462D, 456 y recibiendo actualizaciones de la memoria caché 438).

Un conjunto de registros 445 almacena datos de contexto para hilos ejecutados por los motores de procesamiento de gráficos 431-432, N y un circuito de gestión de contexto 448 gestiona los contextos de hilos. Por ejemplo, el circuito de gestión de contexto 448 puede realizar operaciones de guardar y recuperar para guardar y recuperar contextos de los diversos hilos durante cambios de contexto (por ejemplo, cuando 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 de gráficos). Por ejemplo, en un cambio de contexto, el circuito de administración de contexto 448 puede almacenar valores de registro actuales en una región designada en la memoria (por ejemplo, identificada por un puntero de contexto). A continuación, puede recuperar los valores de registro al volver al contexto. En una realización, un circuito de administración de interrupciones 447 recibe y procesa interrupciones recibidas de dispositivos del sistema.

En una implementación, las direcciones virtuales/efectivas de un motor de procesamiento de gráficos 431 se traducen a direcciones reales/físicas en la memoria del sistema 441 por la MMU 439. Una realización del circuito de integración del acelerador 436 admite 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 única aplicación ejecutada en el procesador 407 o puede ser compartido entre múltiples aplicaciones. En una realización, se presenta un entorno de ejecución de gráficos virtualizado en el que los recursos de los motores de procesamiento de gráficos 431-432, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos pueden subdividirse en "secciones" que se asignan a diferentes VM y/o aplicaciones en función de los requisitos de procesamiento y las prioridades asociadas con las VM y/o aplicaciones.

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

Debido a que los recursos de hardware de los motores de procesamiento de gráficos 431-432, N se mapean explícitamente al espacio de direcciones reales que ve el procesador anfitrión 407, cualquier procesador anfitrión puede direccionar estos recursos directamente utilizando un valor de dirección efectivo. Una función del circuito de integración del acelerador 436, en una realización, es la separación física de los motores de procesamiento de gráficos 431-432, N de modo que aparezcan ante 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 de gráficos 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 de gráficos 431-432, N. Las memorias gráficas 433-434, M pueden ser memorias volátiles tales como DRAM (incluidas las 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 de gráficos 431-432, N y preferentemente 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 preferentemente no los motores de procesamiento de gráficos 431-432, N) dentro de las memorias caché 462A-462D, 456 de los núcleos y la memoria del sistema 441.

La **Figura 4C** ilustra otra realización ejemplar en la que el circuito de integración del acelerador 436 está integrado dentro del procesador 407. En esta realización, los motores de procesamiento de gráficos 431-432, N se comunican directamente a través del enlace de alta velocidad 440 con el circuito de integración del acelerador 436 a través de 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 caudal superior dada su estrecha proximidad al bus de coherencia 464 y a las memorias caché 462A-462D, 426.

Una realización admite diferentes modelos de programación que incluyen un modelo de programación de proceso dedicado (sin virtualización del módulo de aceleración de gráficos) y modelos de programación compartidos (con virtualización). Estos últimos pueden incluir modelos de programación que están controlados por el circuito de integración del acelerador 436 y modelos de programación que están controlados por el módulo de aceleración de gráficos 446.

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

En los modelos de programación de procesos dedicados, los motores de procesamiento de gráficos 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 de gráficos 431-432, N para permitir el acceso por parte de cada sistema operativo. Para sistemas de una sola partición sin un hipervisor, los motores de procesamiento de gráficos 431-432, N son propiedad del sistema operativo. En ambos casos, el sistema operativo puede virtualizar los motores de procesamiento de gráficos 431-432, N para proporcionar acceso a cada proceso o aplicación.

Para el modelo de programación compartido, el módulo de aceleración de gráficos 446 o un motor de procesamiento de gráficos individual 431-432, N selecciona un elemento de proceso utilizando un identificador de proceso. En una realización, los elementos de proceso se almacenan en la memoria del sistema 441 y son direccionables utilizando las técnicas de traducción de dirección efectiva a dirección real descritas en este 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 de gráficos 431-432, N (es decir, se llama al software del sistema para agregar el elemento de proceso a la lista enlazada de elementos de proceso). Los 16 bits inferiores del identificador de proceso pueden ser el desplazamiento del elemento de proceso dentro de la lista enlazada de elementos de proceso.

La **Figura 4D** ilustra una sección de integración de acelerador 490 ejemplar. Como se utiliza en este documento, una "sección" comprende una porción especificada de los recursos de procesamiento del circuito de integración de acelerador 436. El espacio de direcciones efectivas de la aplicación 482 dentro de la memoria del sistema 441 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 del proceso para la aplicación correspondiente 480. Un descriptor de trabajo (WD) 484 contenido en el elemento de proceso 483 puede ser un trabajo único solicitado por una aplicación o puede contener un puntero a una cola de trabajos. En el último caso, el WD 484 es un puntero a la cola de solicitudes de trabajo en el espacio de direcciones de la aplicación 482.

El módulo de aceleración de gráficos 446 y/o los motores de procesamiento de gráficos individuales 431-432, N pueden 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 del proceso y enviar un WD 484 a un módulo de aceleración de gráficos 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 único proceso posee el módulo de aceleración de gráficos 446 o un motor de procesamiento de gráficos individual 431. Debido a que el módulo de aceleración de gráficos 446 es propiedad de un único proceso, el hipervisor inicializa el circuito de integración del acelerador 436 para la partición propietaria y el sistema operativo inicializa el circuito de integración del acelerador 436 para el proceso propietario en el momento en que se asigna el módulo de aceleración de gráficos 446.

En funcionamiento, una unidad de recuperación de WD 491 en la sección de integración del acelerador 490 recupera el siguiente WD 484 que incluye una indicación del trabajo que debe realizar uno de los motores de procesamiento de gráficos del módulo de aceleración de gráficos 446. Los datos del WD 484 pueden almacenarse en registros 445 y usarse por la MMU 439, el circuito de administración de interrupciones 447 y/o el circuito de administración de contexto 448 como se ilustra. Por ejemplo, una realización de la MMU 439 incluye un circuito de recorrido de segmentos/páginas para acceder a las tablas de segmentos/páginas 486 dentro del espacio de direcciones virtuales del SO 485. El circuito de administración de interrupciones 447 puede procesar eventos de interrupción 492 recibidos desde el módulo de aceleración de gráficos 446. Al realizar operaciones gráficas, una dirección efectiva 493 generada por un motor de procesamiento de gráficos 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 de gráficos 431-432, N y/o módulo de aceleración de gráficos 446 y puede ser inicializado por el hipervisor o el sistema operativo. Cada uno de estos registros duplicados puede incluirse en una sección de integración del acelerador 490. En la **Tabla 1** se muestran registros de ejemplo que pueden ser inicializados por el hipervisor.

Tabla 1 - Registros inicializados por el hipervisor

1	Registro de control de sección
2	Puntero de área de procesos programados 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 del acelerador del hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

En la **Tabla 2** se muestran registros de ejemplo que pueden ser inicializados por el sistema operativo.

Tabla 2 - Registros inicializados por el sistema operativo

1	Identificación de proceso e hilo
2	Puntero de guardado/recuperación de contexto de dirección efectiva (EA)
3	Puntero de registro de utilización de acelerador de dirección virtual (VA)
4	Puntero de tabla de 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 de gráficos 446 y/o motor de procesamiento de gráficos 431-432, N en particular. Contiene toda la información que un motor de procesamiento de gráficos 431-432, N requiere para hacer su trabajo o puede ser un puntero a una ubicación de memoria donde la aplicación ha configurado una cola de comandos de trabajo.

La **Figura 4E** ilustra detalles adicionales para una realización de un modelo compartido. Esta realización incluye un espacio de direcciones reales de hipervisor 498 en el que se almacena una lista de elementos de proceso 499. El espacio de direcciones reales de hipervisor 498 es accesible a través de un hipervisor 496 que virtualiza los motores del módulo de aceleración de gráficos 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 de gráficos 446. Hay dos modelos de programación en los que el módulo de aceleración de gráficos 446 es compartido por múltiples procesos y particiones: compartido por intervalos de tiempo y compartido dirigido por gráficos.

En este modelo, el hipervisor del sistema 496 posee el módulo de aceleración de gráficos 446 y pone su función a disposición de todos los sistemas operativos 495. Para que un módulo de aceleración de gráficos 446 admita la virtualización por parte del hipervisor del sistema 496, el módulo de aceleración de gráficos 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 de gráficos 446 debe proporcionar un mecanismo de guardado y recuperación de contexto. 2) El módulo de aceleración de gráficos 446 garantiza que la solicitud de trabajo de una aplicación se complete en una cantidad de tiempo especificada, incluidos los errores de traducción, o el módulo de aceleración de gráficos 446 proporciona la capacidad de adelantarse al procesamiento del trabajo. 3) Se debe garantizar la equidad entre procesos del módulo de aceleración de gráficos 446 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 del sistema operativo 495 con un tipo de módulo de aceleración de gráficos 446, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de guardado/recuperación de contexto (CSRP). El tipo de módulo de aceleración de gráficos 446 describe la función de aceleración de destino para la llamada del sistema. El tipo de módulo de aceleración de gráficos 446 puede ser un valor específico del sistema. El WD está formateado específicamente para el módulo de aceleración de gráficos 446 y puede tener la forma de un comando del módulo de aceleración de gráficos 446, un puntero de dirección efectiva a una estructura definida por el usuario, un

puntero de dirección efectiva a una cola de comandos o cualquier otra estructura de datos para describir el trabajo que debe realizar el módulo de aceleración de gráficos 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 a una aplicación que configura el AMR. Si las implementaciones del circuito de integración del acelerador 436 y del módulo de aceleración de gráficos 446 no admiten un Registro de anulación de máscara de autoridad de usuario (UAMOR), el sistema operativo puede aplicar el valor UAMOR actual al valor AMR antes de pasar el AMR en la llamada del 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 la aplicación 482 para que el módulo de aceleración de gráficos 446 guarde y recupere el estado del contexto. Este puntero es opcional si no se requiere guardar ningún estado entre trabajos o cuando se prioriza un trabajo. El área de guardado/recuperación del contexto puede estar anclada en la memoria del sistema.

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

Tabla 3 - Parámetros de llamada del SO al hipervisor

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

Al recibir la llamada del hipervisor, el hipervisor 496 verifica que el sistema operativo 495 se haya registrado y haya recibido la autorización para utilizar el módulo de aceleración de gráficos 446. El hipervisor 496 coloca entonces el elemento de proceso 483 en la lista enlazada de elementos de proceso para el tipo de módulo de aceleración de gráficos 446 correspondiente. El elemento de proceso puede incluir la información mostrada en la **Tabla 4**.

Tabla 4 - Información de elemento de proceso

1	Un descriptor de trabajo (WD)
2	Un valor del Registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de guardado/recuperación de contexto (CSRP) de dirección efectiva (EA)
4	Un identificador de proceso (PID) y un identificador de hilo opcional (TID)
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de 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 identificador 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 la sección de integración del acelerador 490.

Como se ilustra en la **Figura 4F**, una realización ejemplar de la invención emplea una memoria unificada direccionable a través de un espacio de direcciones de memoria virtual común utilizado para acceder a las memorias de procesador físicas 401-402 y las memorias de GPU 420-423. En esta implementación, las operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de direcciones de memoria virtual/efectiva para acceder a las memorias de

procesador 401-402 y viceversa, simplificando así la programabilidad. En una realización, una primera parte del espacio de direcciones virtuales/efectivas se asigna a la memoria de procesador 401, una segunda parte a la segunda memoria de procesador 402, una tercera parte a la memoria de GPU 420, y así sucesivamente. De este modo, todo el espacio de memoria virtual/efectiva (al que a veces se hace referencia como el espacio de direcciones efectivas) se distribuye entre cada una de las memorias de procesador 401-402 y las memorias GPU 420-423, lo que permite que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual mapeada a esa memoria.

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

Una realización permite que la memoria conectada a la GPU 420-423 se mapee como parte de la memoria del sistema y se acceda a ella utilizando la tecnología de memoria virtual compartida (SVM), pero sin sufrir los inconvenientes de rendimiento habituales asociados con la coherencia de caché de todo el sistema. La capacidad de acceder a la memoria conectada a la GPU 420-423 como memoria del sistema sin una sobrecarga onerosa de coherencia de caché proporciona un entorno operativo beneficioso para la descarga de 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 de controlador, interrupciones y accesos de E/S mapeadas en memoria (MMIO), que son todos ineficientes en relación con los accesos de memoria simples. Al mismo tiempo, la capacidad de acceder a la memoria conectada a la GPU 420-423 sin sobrecargas de coherencia de caché puede ser crítica para el tiempo de ejecución de un cálculo descargado. En casos con un tráfico de memoria de escritura de transmisión sustancial, por ejemplo, la sobrecarga de coherencia de caché puede reducir significativamente el ancho de banda de escritura efectivo visto por una GPU 410-413. La eficiencia de la configuración de operandos, la eficiencia del acceso a los resultados y la eficiencia del cálculo de la GPU juegan un papel en la determinación de la efectividad de la descarga de la GPU.

En una implementación, la selección entre la polarización de la GPU y la polarización del 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 conectada a la GPU. La tabla de polarización se puede implementar en un intervalo de memoria robada de una o más memorias conectadas a la GPU 420-423, con o sin una memoria caché de polarización en la GPU 410-413 (por ejemplo, para almacenar en memoria caché las entradas de la tabla de polarización utilizadas con frecuencia/recientemente). Como alternativa, la tabla de polarización completa se puede mantener dentro de la GPU.

En una implementación, se accede a la entrada de la tabla de polarización asociada con cada acceso a la memoria conectada a la GPU 420-423 antes del acceso real a la memoria de la 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 la GPU se reenvían directamente a una memoria de la GPU correspondiente 420-423. Las solicitudes locales de la GPU que encuentran su página en la polarización del 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 la polarización del procesador anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, las solicitudes dirigidas a una página con polarización de GPU pueden reenviarse a la GPU 410-413. La GPU puede entonces realizar la transición de la página a una polarización del procesador anfitrión si no está utilizando actualmente la página.

El estado de polarización de una página puede modificarse 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 la memoria caché en el anfitrión. La operación de vaciado de memoria 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 la memoria caché se mantiene haciendo que las páginas polarizadas a la GPU no puedan almacenarse temporalmente en memoria caché por el procesador anfitrión 405. Para acceder a estas páginas, el procesador 405 puede solicitar acceso a la GPU 410, que puede otorgar o no acceso de inmediato, según la implementación. Por lo tanto, para reducir la comunicación entre el procesador 405 y la GPU 410, es beneficioso garantizar que las páginas polarizadas a la GPU sean las que requiere la GPU, pero no el procesador anfitrión 405 y viceversa.

Canalización de procesamiento de gráficos

La **Figura 5** ilustra una canalización de procesamiento de gráficos 500, según una realización ejemplar. En una realización, un procesador de gráficos puede implementar la canalización de procesamiento de gráficos 500 ilustrada. El procesador de gráficos puede incluirse dentro de los subsistemas de procesamiento paralelo como se describe en este documento, tales como el procesador paralelo 200 de la **Figura 2A**, que, en una realización, es una variante del procesador o procesadores paralelos 112 de la **Figura 1**. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento de gráficos 500 a través de una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 202 de la **Figura 2A**) como se describe en este documento. Por ejemplo, una unidad de sombreado (por ejemplo, el multiprocesador de gráficos 234 de la **Figura 2C-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 rasterización 526 también pueden ser realizadas por otros motores de procesamiento dentro de una agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 214 de la **Figura 2A**) y una unidad de partición correspondiente (por ejemplo, la unidad de partición 220A-220N de la **Figura 2A**). La canalización de procesamiento de gráficos 500 también puede implementarse utilizando unidades de procesamiento dedicadas para una o más funciones. En una realización, una o más partes de la canalización de procesamiento de gráficos 500 pueden ser realizadas por lógica de procesamiento paralelo dentro de un procesador de propósito general (por ejemplo, CPU). En una realización, una o más partes de la canalización de procesamiento de gráficos 500 pueden acceder a la memoria en chip (por ejemplo, la memoria del 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. El ensamblador de datos 502 luego envía los datos de vértices, 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 sombreado de vértices, iluminando y transformando datos de vértices según lo especificado por los programas de sombreado de vértices. La unidad de procesamiento de vértices 504 lee datos almacenados en memoria caché, memoria local o memoria del sistema para su uso en el procesamiento de los datos de vértices y puede programarse para transformar los datos de vértices de 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 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 de gráficos para su procesamiento por la unidad de procesamiento de control de teselación 508. Las primitivas de gráficos incluyen triángulos, segmentos de línea, puntos, parches, etc., tal como lo admiten varias interfaces de programación de aplicaciones (API) de procesamiento de gráficos.

La unidad de procesamiento de control de teselación 508 trata los vértices de entrada como puntos de control para un parche geométrico. Los puntos de control se transforman de una representación de entrada 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 la 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 de gráficos para su procesamiento por la unidad de procesamiento de geometría 516. La unidad de procesamiento de geometría 516 es una unidad de ejecución programable que ejecuta programas de sombreado de geometría para transformar primitivas de gráficos recibidas del ensamblador de primitivas 514 según lo especificado por los programas de sombreado de geometría. En una realización, la unidad de procesamiento de geometría 516 está programada para subdividir las primitivas de gráficos en una o más primitivas de gráficos nuevas y calcular los parámetros utilizados para rasterizar las nuevas primitivas de gráficos.

En algunas realizaciones, la unidad de procesamiento de geometría 516 puede añadir o eliminar elementos en el flujo de geometría. La unidad de procesamiento de geometría 516 envía los parámetros y vértices que especifican las primitivas de gráficos nuevas 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 de gráficos para su procesamiento por una unidad de escala, selección y recorte de ventana gráfica 520. La unidad de procesamiento de geometría 516 lee datos que están almacenados en la memoria del procesador paralelo o en la memoria del sistema para su uso en el procesamiento de los datos de geometría. La unidad de escala, selección y recorte de la ventana gráfica 520 realiza el recorte, la selección y el escalado de la ventana gráfica y envía las primitivas de gráficos procesadas a un rasterizador 522.

El rasterizador 522 puede realizar la selección de profundidad y otras optimizaciones basadas en la profundidad. El rasterizador 522 también realiza la conversión de escaneo en las nuevas primitivas de gráficos para generar fragmentos y enviar 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 sombreado de fragmentos o programas de sombreado de píxeles. La unidad de procesamiento de fragmentos/píxeles 524 transforma fragmentos o píxeles recibidos del rasterizador 522, según lo especificado por los programas de sombreado de fragmentos o píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 puede programarse para realizar operaciones que incluyen, entre otras, mapeo de texturas, sombreado, combinación, corrección de texturas y corrección de perspectiva para producir fragmentos o píxeles sombreados que se envían a una unidad de operaciones de fotogramas 526. La unidad de procesamiento de fragmentos/píxeles 524 puede leer datos que se almacenan en la memoria del procesador paralelo o en la memoria del sistema para su uso al procesar los datos de fragmentos. Los programas de sombreado de fragmentos o píxeles pueden configurarse 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 rasterización 526 es una unidad de procesamiento que realiza operaciones de rasterización que incluyen, entre otras, estarcido, prueba z, combinación y similares, y genera datos de píxeles como datos gráficos procesados que se almacenarán en la memoria gráfica (por ejemplo, la memoria del procesador paralelo 222 como en la **Figura 2A**, y/o la memoria del sistema 104 como en la **Figura 1**), para visualizarse 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 de color o z que se escriben en la memoria y descomprimir datos de color o z que se leen desde la memoria.

Descripción 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 reconocimiento de texto a voz y/o 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 realimentación prospectiva puede implementarse como un gráfico acíclico en el que los nodos están dispuestos en capas. Normalmente, una topología de red realimentación prospectiva incluye una capa de entrada y una capa de salida que están separadas por al menos una capa oculta. La capa oculta transforma la entrada recibida por la capa de entrada en una representación que es útil para generar la salida en la capa de salida. Los nodos de red están completamente conectados mediante bordes a los nodos en capas adyacentes, pero no hay bordes entre nodos dentro de cada capa. Los datos recibidos en los nodos de una capa de entrada de una red de realimentación prospectiva se propagan (es decir, "se realimentan prospectivamente") a los nodos de la capa de salida mediante una función de activación que calcula los estados de los nodos de cada capa sucesiva en la red basándose en coeficientes ("pesos") asociados, respectivamente, con cada uno de los bordes que conectan las capas. Dependiendo del modelo específico 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, utilizar un conjunto de datos de entrenamiento que representan un problema modelado por la red y ajustar los pesos hasta que el modelo de red funcione 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 usado 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 de gráficos de propósito general.

La **Figura 6** es un diagrama generalizado de una pila de software de aprendizaje automático 600. Se puede configurar una aplicación de aprendizaje automático 602 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 602 puede incluir funcionalidad de entrenamiento e inferencia para una red neuronal y/o software especializado que se puede utilizar para entrenar una red neuronal antes de su implementación. La aplicación de aprendizaje automático 602 puede implementar cualquier tipo de inteligencia de máquina, incluidos, entre otros, reconocimiento de imágenes, mapeo y localización, navegación autónoma, síntesis de voz, imágenes médicas o traducción de idiomas.

La aceleración de hardware para la aplicación de aprendizaje automático 602 se puede habilitar a través de un entorno de aprendizaje automático 604. El entorno de aprendizaje automático 604 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 entorno de aprendizaje automático 604, 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 entorno de aprendizaje automático 604. Las primitivas de ejemplo incluyen convoluciones tensoriales, funciones de activación y agrupamiento, que son operaciones computacionales que se realizan durante el entrenamiento de una red neuronal convolucional (CNN). El entorno de aprendizaje automático 604 también puede proporcionar primitivas para implementar subprogramas de álgebra lineal básica realizados por muchos algoritmos de aprendizaje automático, tales como operaciones con matrices y vectores.

El entorno de aprendizaje automático 604 puede procesar datos de entrada recibidos desde la aplicación de aprendizaje automático 602 y generar la entrada adecuada para un entorno de computación 606. El entorno de computación 606 puede abstraer las instrucciones subyacentes proporcionadas al controlador GPGPU 608 para permitir que el entorno de aprendizaje automático 604 aproveche la aceleración de hardware a través del hardware GPGPU 610 sin requerir que el entorno de aprendizaje automático 604 tenga un conocimiento profundo de la arquitectura del hardware GPGPU 610. Además, el entorno de computación 606 puede habilitar la aceleración de hardware para el entorno de aprendizaje automático 604 en una variedad de tipos y generaciones de hardware GPGPU 610.

Aceleración de Aprendizaje Automático de GPGPU

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

La GPGPU 700 incluye una interfaz de anfitrión 702 para permitir una conexión con un procesador anfitrión. En una realización, la interfaz anfitrión 702 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 700 recibe comandos del procesador anfitrión y utiliza un planificador global 704 para distribuir los hilos de ejecución asociados con esos comandos a un conjunto de agrupaciones de computación 706AH. Las agrupaciones de computación 706A-H comparten una memoria caché 708. La memoria caché 708 puede servir como una memoria caché de nivel superior para memorias caché dentro de las agrupaciones de computación 706A-H.

La GPGPU 700 incluye memoria 714A-B acoplada con las agrupaciones de computación 706A-H a través de un conjunto de controladores de memoria 712A-B. En varias formas de realización, la memoria 714A-B puede incluir varios tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámico (DRAM) o memoria de acceso aleatorio gráfico, como memoria de acceso aleatorio gráfico sincrónico (SGRAM), incluyendo memoria de doble velocidad de datos gráficos (GDDR). En una forma de realización, las unidades de memoria 224A-N también pueden incluir memoria apilada 3D, incluyendo, entre otras, memoria de alto ancho de banda (HBM).

En una realización, cada agrupación de computación 706A-H incluye un conjunto de multiprocesadores gráficos, tal como el multiprocesador gráfico 400 de la Figura 4A. Los multiprocesadores gráficos de la agrupación de computación tienen múltiples tipos de unidades lógicas de punto flotante y entero que pueden realizar operaciones computacionales con un intervalo de precisiones, incluso adecuadas para cálculos de aprendizaje automático. Por ejemplo, y en una realización, al menos un subconjunto de las unidades de punto flotante en cada una de las agrupaciones de computación 706A-H se puede configurar para realizar operaciones de punto flotante de 16 bits o 32 bits, mientras que un subconjunto diferente de unidades de punto flotante se puede configurar para realizar operaciones de punto flotante de 64 bits.

Se pueden configurar varias instancias de GPGPU 700 para que funcionen como una agrupación de computación. El mecanismo de comunicación utilizado por la agrupación de computación para la sincronización y el intercambio de datos varía según las implementaciones. En una realización, las múltiples instancias de la GPGPU 700 se comunican a través de la interfaz anfitrión 702. En una realización, la GPGPU 700 incluye un concentrador de E/S 708 que acopla la GPGPU 700 con un enlace de GPU 710 que permite una conexión directa a otras instancias de la GPGPU. En una realización, el enlace de GPU 710 está acoplado a un puente de GPU a GPU dedicado que permite la comunicación y sincronización entre múltiples instancias de la GPGPU 700. En una realización, el enlace de GPU 710 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En una realización, las múltiples instancias de la GPGPU 700 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red al que se puede acceder mediante la interfaz anfitrión 702. En una realización, el enlace de GPU 710 se puede configurar para permitir una conexión a un procesador anfitrión, además, de o como alternativa a la interfaz anfitrión 702.

Si bien la configuración ilustrada de la GPGPU 700 se puede configurar para entrenar redes neuronales, una realización proporciona una configuración alternativa de la GPGPU 700 que se puede configurar para su implementación dentro de una plataforma de inferencia de alto rendimiento o bajo consumo. En una configuración de inferencia, la GPGPU 700 incluye menos agrupaciones de computación 706A-H en relación con la configuración de entrenamiento. Además, la tecnología de memoria asociada con la memoria 714A-B puede diferir entre las configuraciones de inferencia y entrenamiento. En una realización, la configuración de inferencia de la GPGPU 700 puede admitir 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 8** ilustra un sistema informático de múltiples GPU 800 según una realización ejemplar. El sistema informático de múltiples GPU 800 puede incluir un procesador 802 acoplado a múltiples GPGPU 806A-D mediante un conmutador de interfaz de anfitrión 804. El conmutador de interfaz anfitrión 804, en una realización, es un dispositivo de conmutación PCI Express que acopla el procesador 802 a un bus PCI Express mediante el cual el procesador 802 puede comunicarse con el conjunto de GPGPU 806A-D. Cada una de las múltiples GPGPU 806A-D puede ser una instancia de la GPGPU 700 de la **Figura 7**. Las GPGPU 806A-D se pueden interconectar mediante un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 816. Los enlaces de GPU a GPU de alta velocidad pueden conectarse a cada una de las GPGPU 806A-D a través de un enlace de GPU dedicado, tal como el enlace de GPU 710 como en la Figura 7. Los enlaces de GPU P2P 816 permiten la comunicación directa entre cada una de las GPGPU 806A-D sin requerir comunicación a través del bus de interfaz anfitrión al que está conectado el procesador 802. 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 la memoria del sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 800, por ejemplo, a través de uno o más dispositivos de red. Aunque en la realización ilustrada las GPGPU 806A-D se conectan al procesador 802 mediante el conmutador de interfaz de anfitrión 804, en una realización, el procesador 802 incluye el soporte directo para los enlaces de GPU de P2P 816 y puede conectarse directamente a las GPGPU 806A-D.

Implementaciones de Red Neuronal de Aprendizaje Automático

La arquitectura informática proporcionada por las formas de realización descritas en la presente memoria se puede configurar para llevar a cabo los tipos de procesamiento paralelo que son particularmente adecuados para el entrenamiento y despliegue 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 realimentación prospectiva , como se describió anteriormente.

Un segundo tipo ejemplar de red neuronal es la red neuronal convolucional (CNN). Una CNN es una red neuronal de realimentación prospectiva especializada en el procesamiento de datos que tienen una topología conocida tipo cuadrícula, tal como datos de imágenes. En consecuencia, las CNN se utilizan comúnmente para aplicaciones de visión artificial y reconocimiento de imágenes, pero también pueden emplearse para otros tipos de reconocimiento de patrones, tales como el procesamiento del habla y del lenguaje. Los nodos de la capa de entrada de la CNN se organizan 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 CNN 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 distintos componentes de color de una imagen de entrada. El núcleo de convolución puede ser una matriz multidimensional de parámetros, donde los parámetros son adaptados por el proceso de entrenamiento de la red neuronal.

Las redes neuronales recurrentes (RNN) son una familia de redes neuronales de realimentación prospectiva que incluyen conexiones de retroalimentación entre capas. Las RNN 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 RNN se utiliza como retroalimentación para procesar la entrada posterior en una secuencia. Esta característica hace que las RNN sean particularmente útiles para el procesamiento del lenguaje debido a la naturaleza variable en que se pueden componer los datos del lenguaje.

Las figuras que se describen a continuación presentan ejemplos de redes realimentación prospectiva, CNN y RNN, y, además, describen 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 este 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 aprendizaje automático que utiliza redes neuronales profundas. Las redes neuronales profundas utilizadas en el aprendizaje profundo son redes neuronales artificiales compuestas por múltiples capas ocultas, a diferencia de las redes neuronales superficiales que incluyen solo una única capa oculta. Las redes neuronales más profundas generalmente requieren un entrenamiento computacional más intensivo. Sin embargo, las capas ocultas adicionales de la red permiten el reconocimiento de patrones de múltiples pasos que resultan en un error de salida reducido en relación con las técnicas de aprendizaje automático superficiales.

Las redes neuronales profundas utilizadas en el aprendizaje profundo generalmente incluyen una red de extremo frontal para realizar el reconocimiento de características acoplada a una red de extremo posterior que representa un modelo matemático que puede realizar operaciones (por ejemplo, clasificación de objetos, reconocimiento de 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 mapear 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 a realizar tareas específicas. El modelo de aprendizaje describe cómo ajustar los pesos dentro del modelo para reducir el error de salida de la red. La propagación hacia atrás de errores es un procedimiento 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 9A-B** ilustran una red neuronal convolucional ejemplar. La **Figura 9A** ilustra varias capas dentro de una CNN. Como se muestra en la **Figura 9A**, una CNN ejemplar utilizada para modelar el procesamiento de imágenes puede recibir la entrada 902 que describe los componentes rojo, verde y azul (RGB) de una imagen de entrada. La entrada 902 puede ser procesada por múltiples capas convolucionales (por ejemplo, capa convolucional 904, capa convolucional 906). La salida de las múltiples capas convolucionales puede ser procesada opcionalmente por un conjunto de capas completamente conectadas 908. Las neuronas en una capa completamente conectada tienen conexiones completas con todas las activaciones en la capa anterior, como se describió previamente para una red de realimentación prospectiva. La salida de las capas completamente conectadas 908 se puede utilizar para generar un resultado de salida de la red. Las activaciones dentro de las capas completamente conectadas 908 se pueden calcular utilizando la multiplicación de matrices en lugar de convolución. No todas las implementaciones de CNN utilizan capas completamente conectadas 906. Por ejemplo, en algunas implementaciones, la capa convolucional 906 puede generar salida para la CNN.

Las capas convolucionales están escasamente conectadas, lo que difiere de la configuración de red neuronal tradicional que se encuentra en las capas completamente conectadas 908. 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 es la entrada (en lugar del valor de estado respectivo de cada uno de los nodos del campo) a 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 a la CNN escalar para procesar imágenes grandes.

La **Figura 9B** ilustra etapas de computación ejemplares dentro de una capa convolucional de una CNN. La entrada a una capa convolucional 912 de una CNN se puede procesar en tres etapas de una capa convolucional 914. Las tres etapas pueden incluir una etapa de convolución 916, una etapa de detector 918 y una etapa de agrupamiento 920. La capa de convolución 914 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 CNN.

En la etapa de convolución 916, la capa convolucional 914 puede realizar varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La etapa de convolución 916 puede incluir una transformación afín, que es cualquier transformación que 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 916 define un conjunto de activaciones lineales que son procesadas por etapas sucesivas de la capa convolucional 914.

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

La etapa de agrupamiento 920 utiliza una función de agrupamiento que reemplaza la salida de la capa convolucional 906 con una estadística de resumen de las salidas cercanas. La función de agrupamiento se puede utilizar para introducir invariancia de traslación en la red neuronal, de modo que pequeñas traslaciones a la entrada no cambien las salidas agrupadas. La invariancia a la traslación local puede ser útil en escenarios donde la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Se pueden utilizar varios tipos de funciones de agrupamiento durante la etapa de agrupamiento 920, incluidas el agrupamiento máximo, el agrupamiento promedio y el agrupamiento de norma l2. Además, algunas implementaciones de CNN 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 914 puede luego ser procesada por la siguiente capa 922. La siguiente capa 922 puede ser una capa convolucional adicional o una de las capas completamente conectadas 908. Por ejemplo, la primera capa convolucional 904 de la **Figura 9A** puede generar salida a la segunda capa convolucional 906, mientras que la segunda capa convolucional puede generar salida a una primera capa de las capas completamente conectadas 908.

La **Figura 10** ilustra una red neuronal recurrente 1000 ejemplar. En una red neuronal recurrente (RNN), el estado anterior de la red influye en la salida del estado actual de la red. Las RNN se pueden construir de distintas maneras utilizando una variedad de funciones. El uso de RNN generalmente gira en torno al uso de modelos matemáticos para predecir el futuro basándose en una secuencia previa de entradas. Por ejemplo, una RNN puede usarse para realizar modelos estadísticos del lenguaje para predecir una próxima palabra dada una secuencia previa de palabras. La RNN 1000 ilustrada se puede describir como teniendo una capa de entrada 1002 que recibe un vector de entrada, capas ocultas 1004 para implementar una función recurrente, un mecanismo de retroalimentación 1005 para habilitar una 'memoria' de estados anteriores y una capa de salida 1006 para emitir un resultado. La RNN 1000 funciona en base a pasos de tiempo. El estado de la RNN en un paso de tiempo determinado se ve influenciado en función del paso de tiempo anterior a través del mecanismo de retroalimentación 1005. Para un paso de tiempo determinado, el estado de las capas ocultas 1004 está definido por el estado anterior y la entrada en el paso de tiempo actual. Una entrada inicial (x_1) en un primer paso de tiempo puede ser procesada por la capa oculta 1004. La capa oculta 1004 puede procesar una segunda entrada (x_2) utilizando información de estado que se determina durante el procesamiento de la entrada inicial (x_1). Un estado dado se puede calcular como $s_t = f(Ux_t + Ws_{t-1})$, 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 1004 puede variar dependiendo de los detalles de implementación específicos de la RNN 1000.

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

La **Figura 11** ilustra un entrenamiento y despliegue ejemplar de una red neuronal profunda. Una vez que una red determinada se ha estructurado para una tarea, la red neuronal se entrena utilizando un conjunto de datos de entrenamiento 1102. Se han desarrollado varios entornos de entrenamiento 1104 para permitir la aceleración de hardware del proceso de entrenamiento. Por ejemplo, el entorno de aprendizaje automático 604 de la **Figura 6** puede configurarse como un entorno de entrenamiento 604. El entorno de entrenamiento 604 puede conectarse a una red neuronal no entrenada 1106 y permitir que la red neuronal no entrenada se entrene utilizando los recursos de procesamiento paralelo descritos en este documento para generar una red neuronal entrenada 1108.

Para iniciar el proceso de entrenamiento, los pesos iniciales pueden elegirse de forma aleatoria o mediante un 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 procedimiento de aprendizaje en el que el entrenamiento se realiza como una operación mediada, tal como cuando el conjunto de datos de entrenamiento 1102 incluye una entrada emparejada con la salida deseada para la entrada, o donde el conjunto de datos de entrenamiento incluye una entrada que tiene una salida conocida y la salida de la red neuronal se califica manualmente. La red procesa las entradas y compara las salidas resultantes contra un conjunto de salidas esperadas o deseadas. Los errores luego se propagan a través del sistema. El entorno de entrenamiento 1104 puede ajustarse para ajustar los pesos que controlan la red neuronal no entrenada 1106. El entorno de entrenamiento 1104 puede proporcionar herramientas para monitorizar lo bien que la red neuronal no entrenada 1106 está convergiendo 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 1108. La red neuronal entrenada 1108 puede luego desplegarse para implementar cualquier cantidad de operaciones de aprendizaje automático.

El aprendizaje no supervisado es un procedimiento 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 1102 incluirá datos de entrada sin ningún dato de salida asociado. La red neuronal no entrenada 1106 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 1107 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 1102 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que los datos de entrada se utilizan continuamente para seguir entrenando el modelo. El aprendizaje incremental permite que la red neuronal entrenada 1108 se adapte a los nuevos datos 1112 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 computacional. En lugar de utilizar un único nodo computacional, se puede utilizar una red distribuida de nodos computacionales para acelerar el proceso de entrenamiento.

La **Figura 12** es un diagrama de bloques ejemplar que ilustra el aprendizaje distribuido. El aprendizaje distribuido es un modelo de entrenamiento que utiliza múltiples nodos de computación distribuida para realizar un entrenamiento supervisado o no supervisado de una red neuronal. Los nodos computacionales distribuidos pueden incluir, cada uno, uno o más procesadores anfitrión y uno o más de los nodos de procesamiento de propósito general, tales como la unidad de procesamiento de gráficos de propósito general altamente paralela 700 como en la **Figura 7**. Como se ilustra, el aprendizaje distribuido se puede realizar mediante paralelismo de modelos 1202, paralelismo de datos 1204 o una combinación de paralelismo de modelos y datos 1206.

En el paralelismo de modelos 1202, 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 1204, 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 del 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, desviaciones) en el promedio de los parámetros de cada nodo. El promedio de parámetros utiliza un servidor de parámetros central que mantiene los datos de los 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 se transfieren entre nodos.

El paralelismo combinado de modelos y datos 1206 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 que el entrenamiento en una sola máquina. Sin embargo, los procesadores paralelos y las GPGPU descritos en este 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 de aprendizaje automático ejemplares

El aprendizaje automático se puede aplicar para resolver una variedad de problemas tecnológicos, incluidos, entre otros, la visión artificial, la conducción y la navegación autónomas, el reconocimiento de voz y el 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 artificial pueden configurarse para reconocer ondas sonoras a partir de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesador paralelo permite entrenar aplicaciones de visión artificial utilizando conjuntos de datos de entrenamiento significativamente más grandes que lo que era posible antes y permite implementar sistemas de inferencia utilizando procesadores paralelos de bajo consumo.

El aprendizaje automático acelerado por procesador paralelo tiene aplicaciones de conducción autónoma que incluyen reconocimiento de carriles y señales de tráfico, evitación de obstáculos, navegación y control de la 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 este documento pueden permitir el entrenamiento rápido de las redes neuronales cada vez más complejas utilizadas para soluciones de conducción autónoma y permiten el despliegue 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 hecho posible el uso de enfoques de aprendizaje automático para el reconocimiento automático de voz (ASR). 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 ASR.

El aprendizaje automático acelerado por procesador paralelo también se puede utilizar para acelerar el procesamiento del lenguaje natural. Los procedimientos de aprendizaje automático pueden utilizar algoritmos de inferencia estadística para producir modelos que sean robustos ante entradas erróneas o desconocidas. 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 implementación. Las plataformas de entrenamiento generalmente son altamente 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 ejemplares adecuados para el

entrenamiento incluyen la unidad de procesamiento de gráficos de propósito general altamente paralela 700 de la Figura 7 y el sistema informático de múltiples GPU 800 de la Figura 8. Por el contrario, las plataformas de aprendizaje automático implementadas generalmente incluyen procesadores paralelos de menor potencia, adecuados para su uso en productos tales como cámaras, robots autónomos y vehículos autónomos.

La **Figura 13** ilustra un sistema de inferencia ejemplar en un chip (SOC), 1300, adecuado para realizar inferencias utilizando un modelo entrenado. El SOC 1300 puede integrar componentes de procesamiento que incluyen un procesador de medios 1302, un procesador de visión 1304, una GPGPU 1306 y un procesador de múltiples núcleos 1308. El SOC 1300 puede incluir, además, una memoria en chip 1305 que puede habilitar un grupo de datos en chip compartido al que puede 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 1300 se puede utilizar como parte del sistema de control principal de un vehículo autónomo. Cuando el SOC 1300 está configurado para su uso en vehículos autónomos, el SOC está diseñado y configurado para cumplir con los estándares de seguridad funcional pertinentes de la jurisdicción de implementación.

Durante el funcionamiento, el procesador de medios 1302 y el procesador de visión 1304 pueden trabajar en conjunto para acelerar las operaciones de visión por ordenador. El procesador de medios 1302 puede posibilitar la decodificación de latencia baja de múltiples flujos de vídeo de alta resolución (por ejemplo, 4K, 8K). Los flujos de vídeo decodificados se pueden escribir en una memoria intermedia en la memoria del chip 1305. El procesador de visión 1304 puede luego analizar el vídeo decodificado y realizar operaciones de procesamiento preliminar en las tramas del vídeo decodificado como preparación para el procesamiento de las tramas utilizando un modelo de reconocimiento de imágenes entrenado. Por ejemplo, el procesador de visión 1304 puede acelerar las operaciones de convolución para una CNN que se utiliza para realizar el reconocimiento de imágenes en datos de vídeo de alta resolución, mientras que los cálculos del modelo de extremo posterior los realiza la GPGPU 1306.

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

La GPGPU 1306 puede incluir agrupaciones de computación tales como una configuración de bajo consumo de las agrupaciones de computación 706A-706H dentro de la unidad de procesamiento de gráficos de propósito general altamente paralela 700. Las agrupaciones de computación dentro de la GPGPU 1306 pueden admitir instrucciones específicamente optimizadas para realizar cálculos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1306 puede admitir instrucciones para realizar cálculos de baja precisión, tales como operaciones con vectores enteros de 8 y 4 bits.

Sistemas de procesamiento de imágenes que utilizan CNN mejorada

La **Figura 14** es un diagrama de bloques ejemplar de un sistema de procesamiento de imágenes 1400 que tiene un sistema de red neuronal convolucional (CNN) 1404 con una CNN mejorada para procesar una imagen de entrada 1402. En una realización, el sistema de CNN 1404 implementa una CNN mejorada según las **Figuras 15A-15B** y **16A-16B** con nodos de capa reducidos. Para estas realizaciones ejemplares, al utilizar nodos de capa reducidos, se puede reducir significativamente el número de conexiones y parámetros para el cálculo de convolución. En otra realización, el sistema de CNN 1404 implementa una CNN mejorada según las **Figuras 17**, **18A** y **18B** que tiene redes CNN superficiales que imitan una red neuronal profunda (DNN). En estas realizaciones ejemplares, las redes CNN superficiales y eficientes pueden imitar una DNN utilizando menos computación con menores requisitos de memoria.

En algunas realizaciones, el sistema de CNN 1404 puede incluir o ser implementado por o con los sistemas y procesadores dados a conocer y descritos en las **Figuras 1-8** y **19-32**. En otras realizaciones, el sistema de CNN, 1404, se puede implementar utilizando aceleración de hardware como se describe en las **Figuras 6** y **7**. En una realización, la imagen de entrada, 1402, es una imagen capturada por un conjunto de sensores (no se muestra) y puede estar en un formato Rojo Verde Azul (RGB) que tiene valores R, G y B y, en otras realizaciones, la imagen de entrada 1402 está en formato de píxeles de espacio de color (YUV) que tiene valores de brillo, luminancia y crominancia de color. Se puede utilizar cualquier tipo de dispositivo de captura de imágenes para capturar la imagen de entrada 1402 y la memoria (no se muestra) en el sistema de CNN 1404 para almacenar la imagen de entrada 1402. En realizaciones ejemplares, el sistema de CNN 1404 procesa la imagen de entrada 1402 utilizando CNN mejoradas como se da a conocer con respecto a las **Figuras 15A-15B**, **16A-17** y **18A-18B**, que puede reducir significativamente los cálculos de convolución para sistemas de procesamiento con memoria y recursos de cálculo limitados.

(CNN con nodos de capa reducida)

Las **Figuras 15A-15B** ilustran un sistema de procesamiento de imágenes 1500 que procesa una imagen de entrada submuestreada 1504 (por ejemplo, cuatro imágenes más pequeñas de la imagen de entrada 1502) con una CNN mejorada 1508 que utiliza nodos de capa CNN reducidos de acuerdo con realizaciones ejemplares. La arquitectura de CNN 1508 mejorada puede reducir el entrenamiento y mejorar las pruebas al utilizar menos conexiones (es decir, nodos de capa de CNN) para reducir el cálculo de CNN de los parámetros respectivos.

En referencia a la **Figura 15A**, la imagen de entrada 1502 puede tener cualquier tipo de resolución o dimensión de matriz de píxeles definida por ancho (W) y alto (H). En una realización, la imagen de entrada 1502 proporciona valores de color RGB y, en otras realizaciones, la imagen de entrada 1502 proporciona valores de color YUV. Se puede entrenar una CNN tradicional con conjuntos de datos para procesar la resolución completa W x H de la imagen de entrada 1502. Esta CNN puede tener tres capas principales: capas de convolución, capas de agrupamiento y las capas restantes denominadas capas completamente conectadas o capas de salida. La capa de salida final aplica una función llamada Softmax que ayuda a clasificar la imagen de entrada 1502. Por ejemplo, la capa de salida puede generar puntuaciones de clase y puede proporcionar N funciones Softmax que proporcionen una distribución en N etiquetas. Las etiquetas se pueden utilizar para identificar píxeles o imágenes. Una CNN de este tipo se puede entrenar completamente conteniendo una gran cantidad de conexiones y parámetros de capa nodal. En las realizaciones ejemplares dadas a conocer en este documento, se describe un sistema y una red CNN mejorados que pueden utilizar la capacidad de una CNN completa con sistemas y dispositivos que tienen potencia de cálculo y memoria limitadas.

En realizaciones ejemplares, para el sistema de CNN mejorado, la imagen de entrada 1502 se submuestrea primero en imágenes más pequeñas como imagen de entrada submuestreada 1504. En este ejemplo, la imagen de entrada submuestreada 1504 incluye cuatro imágenes más pequeñas, cada una de estas imágenes tiene una resolución de W/2 x H/2. Esta resolución es menor que la resolución completa W x H de la imagen de entrada 1502. En otras realizaciones, se pueden utilizar más de cuatro imágenes para la imagen de entrada submuestreada 1504. Como se muestra en la **Figura 15A**, la resolución de la imagen de entrada 1502 se divide en una pluralidad de bloques de 4 píxeles 1501. Cada una de las cuatro imágenes más pequeñas para la imagen de entrada submuestreada 1504 puede tomar un píxel de entre cuatro píxeles en los bloques de 4 píxeles 1501 para formar la imagen de entrada con tas de muestreo reducida 1504. En una realización, las cuatro imágenes más pequeñas se pueden concatenar entre sí. Cada una de las imágenes más pequeñas es similar a la imagen de entrada 1502 al compartir un píxel en los bloques de 4 píxeles 1501. La imagen de entrada submuestreada 1504 (por ejemplo, cuatro imágenes con resolución W/2 x H/2) se alimenta o introduce a la CNN mejorada 1508. Al usar imágenes más pequeñas, la CNN 1508 mejorada requiere núcleos de CNN más pequeños para realizar operaciones de convolución en la imagen de entrada submuestreada 1504 en comparación con el procesamiento de la imagen de entrada completa 1502 con una resolución completa de W x H.

La CNN 1506 mejorada incluye capas de agrupamiento convolucional 1506. En una realización, las capas de agrupamiento convolucional 1506 incluyen tres etapas de procesamiento como se da a conocer en la **Figura 9B**, tales como la capa de convolución 914, la etapa del detector 918 y la etapa o capa de agrupamiento 912. Las capas de agrupación convolucional 1506 pueden incluir cualquier número de capas de convolución y agrupamiento. Cada capa de convolución incluye una serie de nodos. Los nodos pueden actuar como filtros y sus salidas se propagan a los nodos en capas de CNN sucesivas. Los cálculos para un nodo incluyen la aplicación de una operación matemática de convolución a cada filtro para producir la salida de ese filtro. Esta operación 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. Para una CNN, la primera función de convolución puede denominarse entrada, mientras que la segunda función puede denominarse núcleo de convolución. El resultado puede denominarse mapa de características. Las salidas de los nodos en una capa convolucional se pueden introducir en una capa de agrupamiento que reemplaza las salidas con un resumen estadístico para reducir el tamaño espacial de las salidas de los nodos convolucionales. En este ejemplo, una matriz multidimensional de datos para cada una de las cuatro imágenes más pequeñas de la imagen de entrada submuestreada 1504 se introduce a capas de agrupamiento convolucional 1506 de la CNN 1508 mejorada en la que se realizan operaciones de convolución y agrupamiento en las imágenes más pequeñas de la imagen de entrada 1502.

En realizaciones ejemplares, las capas de agrupamiento convolucional 1506 proporcionan salidas al módulo de capa completamente conectada-1 (1510-1) a través del módulo de capa completamente conectada-K (1510-K). Cada módulo de capa completamente conectada-1 (1510-1) a módulo de capa completamente conectada-K (1510-K) incluye una pluralidad de capas completamente conectadas y una capa de etiqueta final mostrada como etiqueta-1 (1511-1) a etiqueta-K (1511-K) que puede ayudar a identificar píxeles e imágenes. En este ejemplo, los módulos de capa completamente conectadas (1510-1) a (1510-K) se basan en un subconjunto de nodos de convolución utilizados en una red CNN completa, como se describió anteriormente con respecto a una red CNN para procesar la imagen de entrada 1502 con su resolución completa (W x H). Por ejemplo, en referencia a la **Figura 15B**, en una realización, la última capa de agrupamiento convolucional (1506-L) puede utilizar un número reducido de nodos en comparación con la última capa de una CNN completa que utiliza todos los nodos. Como se muestra, se utilizan ciertos subconjuntos de los nodos en la última capa de agrupamiento convolucional (1506-L) y cada subconjunto se agrupa en un respectivo módulo de capa completamente conectada del 1 (1510-1) al K (1510-K).

En este ejemplo, se utiliza una función aleatoria para seleccionar ciertos nodos en la última capa de agrupamiento convolucional (1506-L) como nodos aleatoriamente seleccionados del 1 (1507-1) al K (1507-K) para obtener un subconjunto de nodos en la última capa de agrupamiento convolucional 1506-L. Por ejemplo, con referencia a los nodos aleatoriamente seleccionados 1 (1507-1), se seleccionan cuatro de los siete nodos de la última capa, cuyas salidas se proporcionan como entrada a la capa 1 (1510-1) del módulo de capa completamente conectada. Y en referencia a los nodos seleccionados aleatoriamente (1507-K), se seleccionan tres nodos de los siete nodos de la última capa cuyas salidas se proporcionan como entrada a la capa-K (1510-K) del módulo de capa completamente conectada. En otras realizaciones, se puede utilizar un promedio de la cantidad de nodos para dividir los nodos en K subconjuntos o grupos cuyas salidas se proporcionan como entrada a las respectivas capas de módulo completamente conectadas (1510-1) a (1510-K). La salida de la última capa completamente conectada incluye funciones Softmax de N vías (N funciones de clasificación) que producen una distribución sobre las N etiquetas de clase. De esta manera, dados los conjuntos de datos de entrenamiento, se puede entrenar una DNN con todos los parámetros aprendidos. En estos ejemplos, K y N pueden ser números enteros.

Cabe señalar que los parámetros utilizados para capas completamente conectadas en una CNN completa pueden ocupar más del 90 % de los parámetros totales de la CNN. En las realizaciones ejemplares, al utilizar módulos de capa completamente conectada 1-K (1510-1 a 1510-K) que reciben entradas de subconjuntos o grupos seleccionados de nodos de la última capa de agrupamiento convolucional (1506-L), cada módulo puede dar un resultado de predicción que se muestra como salida 1 (1515-1) a salida K (1515-K). En una realización, se puede tomar un promedio de las salidas 1 (1515-1) a K (1515-K) para obtener un resultado de predicción o salida final que puede mejorar el rendimiento de la prueba de una predicción final. Por lo tanto, en las realizaciones ejemplares de las **Figuras 15A** y **15B**, se pueden utilizar menos parámetros en los que se necesitan menos nodos en la capa de convolución final en comparación con el uso de todos los nodos en la última capa de convolución de una CNN completa, proporcionando así eficiencias de cálculo a un sistema de CNN.

La **Figura 16A** ilustra un diagrama de flujo ejemplar de una operación 1600 para procesar una imagen de entrada (por ejemplo, la imagen de entrada 1502) utilizando una CNN mejorada (por ejemplo, la CNN mejorada 1508) con nodos de capas CNN reducidas de acuerdo con una realización ejemplar. En la operación 1602, una imagen de entrada (por ejemplo, la imagen de entrada 1502) se submuestrea en imágenes más pequeñas (por ejemplo, cuatro imágenes más pequeñas de la imagen de entrada submuestreada 1504). En la operación 1604, las imágenes más pequeñas son procesadas por una CNN mejorada (por ejemplo, CNN mejorada 1508) utilizando nodos de capa reducidos en una última capa de convolución (por ejemplo, última capa de agrupamiento convolucional 1506-L). En la operación 1606 se emiten los resultados de las imágenes más pequeñas procesadas mediante la CNN mejorada. En un ejemplo, la salida puede ser la salida de capas de módulo completamente conectadas (por ejemplo, capa de módulo completamente conectada 1 (1510-1) a K (1510-K)). La salida es la clasificación final de las imágenes procesadas al detectar o identificar características de una imagen de entrada. La salida de la última capa completamente conectada incluye funciones Softmax de N vías (N funciones de clasificación) que producen una distribución sobre las N etiquetas de clase. En estos ejemplos, K y N pueden ser números enteros.

La **Figura 16B** ilustra un diagrama de flujo ejemplar de una operación 1620 para proporcionar salidas de una CNN mejorada (por ejemplo, CNN mejorada 1508) utilizando nodos de capa reducida en una última capa (por ejemplo, última capa de agrupamiento convolucional 1506-L) de acuerdo con una realización ejemplar. En la operación 1622, se selecciona un subconjunto de nodos de capa en una última capa de una CNN completa. En una realización, se seleccionan de manera aleatoria subconjuntos de nodos de capa en la última capa de una CNN completa (por ejemplo, nodos seleccionados aleatoriamente del 1 (1507-1) al K (1507-K)) para procesar imágenes más pequeñas (por ejemplo, imagen de entrada submuestreada 1504) de una imagen de entrada (por ejemplo, imagen de entrada 1502). En la operación 1624, las salidas de los subconjuntos seleccionados de nodos de capa se envían o se introducen a las respectivas capas de módulo completamente conectadas (por ejemplo, capa de módulo completamente conectada 1 (1510-1) a K (1510-K)). En la operación 1626, se emite una clasificación final basada en los resultados de las capas del módulo completamente conectadas. La salida de la última capa completamente conectada incluye funciones Softmax de N vías (N funciones de clasificación) que producen una distribución sobre las N etiquetas de clase. En estos ejemplos, K y N pueden ser números enteros.

(Redes CNN superficiales)

La **Figura 17** ilustra una CNN mejorada con redes CNN superficiales que imitan redes neuronales profundas (DNN) de acuerdo con una realización ejemplar. Como se muestra, las redes CNN superficiales 1702 están configuradas para imitar la red neuronal profunda 1701 utilizando menos capas y nodos y conexiones internodales. En realizaciones ejemplares, las redes CNN superficiales incluyen cada una redes CNN más pequeñas con 3 capas y K nodos ocultos 1703 en la segunda capa, donde K es un número entero producido por un generador aleatorio uniforme. Como se muestra, la red CNN superficial superior incluye 2 nodos ocultos en la segunda capa y la CNN superficial inferior incluye 1 nodo oculto en la segunda capa. Las redes CNN superficiales 1702 son más eficientes en comparación con las redes neuronales profundas 1701, en las que se necesitan menos cálculos de recursos de procesamiento y memoria. Es decir, la red neuronal profunda 1701 requiere un uso intensivo de recursos y su profundidad es expansiva e implica interconexiones complejas.

Las **Figuras 18A-18B** son diagramas de bloques ejemplares para ilustrar el proceso de creación de un conjunto de redes CNN superficiales más pequeñas y más eficientes para imitar una DNN. Haciendo referencia a la **Figura 18A**, se da a conocer un sistema de CNN simplificado 1800 para ilustrar el proceso de generación de un conjunto de redes CNN superficiales. Como se muestra, el arquitecto de red 1804 puede representar la estructura de una DNN en la que un entrenador 1806 obtiene datos de entrenamiento 1802 para construir la DNN de acuerdo con la arquitectura de red 1804. La información de referencia profunda 1810 puede ser utilizada por el entrenador 1806 y un sintonizador fino 1808 para ajustar la DNN. Haciendo referencia a las Figuras 18A y 18B, ahora se explicará el proceso de combinación de redes CNN superficiales 1812 por el combinador 1814 para crear un conjunto de redes 1816 para imitar una DNN.

Por ejemplo, haciendo referencia a la **Figura 18B**, en la operación 1852, se diseña aleatoriamente una estructura CNN de redes CNN superficiales (pequeñas). En realizaciones ejemplares, como se muestra en la **Figura 17**, las redes CNN superficiales diseñadas aleatoriamente constan de 3 capas y K nodos ocultos 1703 en la segunda capa y K se genera mediante un generador aleatorio uniforme. En la operación 1854 se realiza el entrenamiento inicial de las redes CNN superficiales. Por ejemplo, el entrenador 1806 puede entrenar las redes CNN pequeñas o superficiales 1812 utilizando los datos de entrenamiento 1802 y la información de referencia profunda 1810 del entrenador 1806 para DNN 1701. En la operación 1856, las redes CNN superficiales 1812 se ajustan de manera incremental. Por ejemplo, se puede sintonizar una segunda red CNN superficial para mejorar una primera red CNN superficial. Si las muestras son clasificadas erróneamente por una red CNN superficial, pero clasificadas correctamente por una DNN, las redes CNN superficiales 1812 se pueden ajustar en consecuencia. En la operación 1858 se combinan las salidas de las redes CNN superficiales 1812. En una realización, las salidas de las redes CNN superficiales 1812 se pueden combinar en un conjunto de redes 1816 utilizando estrategias de conjunto como la votación. En las realizaciones ejemplares, el conjunto de redes 1816 de redes CNN superficiales más pequeñas 1812 puede imitar y ser tan eficaz como DNN1701, que utilizan menos recursos de computación y memoria en las etapas de entrenamiento y clasificación.

Descripción general del sistema gráfico

La **Figura 19** es un diagrama de bloques de un sistema de procesamiento 1900 según una realización ejemplar. En diversas realizaciones, el sistema 1900 incluye uno o más procesadores 1902 y uno o más procesadores de gráficos 1908, 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 1902 o núcleos de procesador 1907. En una realización, el sistema 1900 es una plataforma de procesamiento incorporada dentro de un circuito integrado de sistema en un chip (SoC) para su uso en dispositivos móviles, portátiles o integrados.

Una realización del sistema 1900 puede incluir, o estar incorporada dentro de una plataforma de juego basada en servidor, una consola de juegos, que incluye una consola de juegos y multimedia, una consola de juegos móvil, una consola de juegos portátil o una consola de juegos en línea. En algunas realizaciones, el sistema 1900 es un teléfono móvil, un teléfono inteligente, un dispositivo informático tipo tableta o un dispositivo de Internet móvil. El sistema de procesamiento de datos 1900 también puede incluir, acoplarse con, o integrarse dentro de un dispositivo ponible, tal como un dispositivo ponible de reloj inteligente, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 1900 es un dispositivo de televisión o decodificador que tiene uno o más procesadores 1902 y una interfaz gráfica generada por uno o más procesadores de gráficos 1908.

En algunas realizaciones, el uno o más procesadores 1902 incluyen, cada uno, uno o más núcleos de procesador 1907 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para el sistema y el software del usuario. En algunas realizaciones, cada uno del uno o más núcleos de procesador 1907 está configurado para procesar un conjunto de instrucciones específico 1909. En algunas realizaciones, el conjunto de instrucciones 1909 puede facilitar la computación de conjunto de instrucciones complejas (CISC), la computación de conjunto de instrucciones reducidas (RISC) o la computación mediante una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 1907 pueden procesar cada uno un conjunto de instrucciones diferente 1909, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 1907 también puede incluir otros dispositivos de procesamiento, tal como un procesador de señal digital (DSP).

En algunas realizaciones, el procesador 1902 incluye memoria caché 1904. Dependiendo de la arquitectura, el procesador 1902 puede tener una única memoria caché interna o múltiples niveles de memoria caché interna. En algunas realizaciones, la memoria caché se comparte entre varios componentes del procesador 1902. En algunas realizaciones, el procesador 1902 también utiliza una memoria caché externa (por ejemplo, una memoria caché de nivel 3 (L3) o una memoria caché de último nivel (LLC)) (no se muestra), que puede compartirse entre los núcleos de procesador 1907 utilizando técnicas de coherencia de caché conocidas. Además, se incluye un archivo de registros 1906 en el procesador 1902 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 1902.

En algunas realizaciones, el procesador 1902 está acoplado a un bus de procesador 1910 para transmitir señales de comunicación, tales como direcciones, datos o señales de control, entre el procesador 1902 y otros componentes del

sistema 1900. En una realización, el sistema 1900 utiliza una arquitectura de sistema de "concentrador" a modo de ejemplo, que incluye un concentrador de controlador de memoria 1916 y un concentrador de controlador de entrada y salida (E/S) 1930. Un concentrador de controlador de memoria 1916 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 1900, mientras que un concentrador del controlador de E/S (ICH) 1930 proporciona conexiones a dispositivos de E/S a través de un bus de E/S local. En una realización, la lógica del concentrador del controlador de memoria 1916 está integrada dentro del procesador.

El dispositivo de memoria 1920 puede ser un dispositivo de memoria de acceso aleatorio dinámico (DRAM), un dispositivo de memoria de acceso aleatorio estático (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 1920 puede funcionar como memoria del sistema para el sistema 1900, para almacenar datos 1922 e instrucciones 1921 para su uso cuando el uno o más procesadores 1902 ejecutan una aplicación o un proceso. El concentrador del controlador de memoria 1916 también se acopla con un procesador de gráficos externo opcional 1912, que puede comunicarse con el uno o más procesadores de gráficos 1908 en los procesadores 1902 para realizar operaciones gráficas y multimedia.

En algunas realizaciones, el ICH 1930 permite que los periféricos se conecten al dispositivo de memoria 1920 y al procesador 1902 a través de un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, entre otros, un controlador de audio 1946, una interfaz de firmware 1928, un transceptor inalámbrico 1926 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 1924 (por ejemplo, unidad de disco duro, memoria flash, etc.) y un controlador de E/S heredado 1940 para acoplar dispositivos heredados (por ejemplo, Sistema Personal 2 (PS/2)) al sistema. Uno o más controladores de bus serie universal (USB) 1942 conectan dispositivos de entrada, tales como combinaciones de teclado y ratón 1944. Un controlador de red 1934 también puede acoplarse al ICH 1930. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla al bus de procesador 1910. Se apreciará que el sistema 1900 mostrado es ejemplar y no limitativo, ya que también se pueden utilizar otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por ejemplo, el concentrador de controlador de E/S 1930 puede estar integrado dentro del uno o más procesadores 1902, o el concentrador de controlador de memoria 1916 y el concentrador de controlador de E/S 1930 pueden estar integrados en un procesador de gráficos externo discreto, como el procesador de gráficos externo 1912.

La **Figura 20** es un diagrama de bloques de una realización ejemplar de un procesador 2000 que tiene uno o más núcleos de procesador 2002A-2002N, un controlador de memoria integrado 2014 y un procesador de gráficos integrado 2008. Aquellos elementos de la **Figura 20** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto. El procesador 2000 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 2002N representado por los recuadros en líneas discontinuas. Cada uno de los núcleos de procesador 2002A-2002N incluye una o más unidades de caché internas 2004A-2004N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades de caché compartidas 2006.

Las unidades de caché internas 2004A-2004N y las unidades de caché compartidas 2006 representan una jerarquía de memoria caché dentro del procesador 2000. La jerarquía de memoria caché puede incluir al menos un nivel de memoria caché de instrucciones y datos dentro de cada núcleo de procesador y uno o más niveles de memoria caché de nivel medio compartido, tales como un Nivel 2 (L2), Nivel 3 (L3), Nivel 4 (L4) u otros niveles de memoria caché, donde el nivel más alto de memoria 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 2006 y 2004A-2004N.

En algunas realizaciones, el procesador 2000 también puede incluir un conjunto de una o más unidades de controlador de bus 2016 y un núcleo de agente de sistema 2010. La una o más unidades de controlador de bus 2016 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 2010 proporciona una funcionalidad de administración para los diversos componentes del procesador. En algunas realizaciones, el núcleo de agente de sistema 2010 incluye uno o más controladores de memoria integrados 2014 para administrar el acceso a varios dispositivos de memoria externa (no se muestran).

En algunas realizaciones, uno o más de los núcleos de procesador 2002A-2002N incluyen soporte para hilos múltiples simultáneos. En una realización de este tipo, el núcleo de agente de sistema 2010 incluye componentes para coordinar y hacer funcionar los núcleos 2002A-2002N durante el procesamiento de múltiples hilos. El núcleo de agente de sistema 2010 puede incluir adicionalmente una unidad de control de energía (PCU), que incluye una lógica y componentes para regular el estado de energía de los núcleos de procesador 2002A-2002N y el procesador de gráficos 2008.

En algunas realizaciones, el procesador 2000 incluye, además, un procesador de gráficos 2008 para ejecutar operaciones de procesamiento de gráficos. En algunas realizaciones, el procesador de gráficos 2008 se acopla con el conjunto de unidades de memoria caché compartidas 2006 y el núcleo del agente del sistema 2010, incluidos uno o

más controladores de memoria integrados 2014. En algunas realizaciones, un controlador de visualización 2011 está acoplado con el procesador de gráficos 2008 para impulsar la salida del procesador de gráficos a una o más pantallas acopladas. En algunas realizaciones, el controlador de visualización 2011 puede ser un módulo separado acoplado con el procesador de gráficos a través de al menos una interconexión, o puede estar integrado dentro del procesador de gráficos 2008 o el núcleo del agente del sistema 2010.

En algunas realizaciones, se utiliza una unidad de interconexión basada en anillo 2012 para acoplar los componentes internos del procesador 2000. 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 de gráficos 2008 se acopla con la interconexión de anillo 2012 a través de un enlace de E/S 2013.

El enlace de E/S 2013 de ejemplo representa al menos una de múltiples variedades de interconexiones de E/S, incluida una interconexión de E/S en paquete que facilita la comunicación entre varios componentes del procesador y un módulo de memoria integrado de alto rendimiento 218, como un módulo eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 2002A-2002N y del procesador de gráficos 2008 utiliza módulos de memoria embebida 2018, tal como una memoria caché compartida de último nivel.

En algunas realizaciones, los núcleos de procesador 2002A-2002N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 2002A-2002N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 2002A-2002N 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 2002A-2002N son heterogéneos en términos de microarquitectura, donde uno o más núcleos que tienen un consumo de potencia relativamente más alto se acoplan con uno o más núcleos de potencia que tienen un consumo de potencia más bajo. Adicionalmente, el procesador 2000 puede implementarse en uno o más chips o como un circuito de SoC integrado que tiene los componentes ilustrados, además de otros componentes.

La **Figura 21** es un diagrama de bloques de un procesador de gráficos 2100, que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento. En algunas realizaciones, el procesador de gráficos se comunica a través de una interfaz de E/S mapeada en memoria con registros en el procesador de gráficos y con comandos colocados en la memoria del procesador. En algunas realizaciones, el procesador de gráficos 2100 incluye una interfaz de memoria 2114 para acceder a la memoria. La interfaz de memoria 2114 puede ser una interfaz a la memoria local, a una o más memorias caché internas, a una o más memorias caché externas compartidas y/o a la memoria del sistema.

En algunas realizaciones, el procesador de gráficos 2100 también incluye un controlador de visualización 2102 para dirigir los datos de salida de pantalla a un dispositivo de pantalla 2120. El controlador de visualización 2102 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 de gráficos 2100 incluye un motor de códec de vídeo 2106 para codificar, decodificar o transcodificar medios hacia, desde o entre uno o más formatos de codificación de medios, incluidos, entre otros, los formatos del Grupo de expertos en imágenes en movimiento (MPEG) tales como MPEG-2, los formatos de codificación de vídeo avanzada (AVC) tales como H.264/MPEG-4 AVC, así como los formatos 421M/VC-1 de la Sociedad de ingenieros de cine y televisión (SMPTE) y del Grupo de expertos fotográficos conjuntos (JPEG) tales como JPEG y Motion JPEG (MJPEG).

En algunas realizaciones, el procesador de gráficos 2100 incluye un motor de transferencia de imágenes en bloque (BLIT) 2104 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 de gráficos 2D se realizan utilizando uno o más componentes del motor de procesamiento de gráficos (GPE) 2110. En algunas realizaciones, el motor de procesamiento de gráficos 2110 es un motor de cómputo para realizar operaciones de gráficos, incluidas operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 2110 incluye una canalización 3D 2112 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 2112 incluye elementos de función programables y fijos que realizan varias tareas dentro del elemento y/o generan hilos de ejecución en un subsistema 3D/Medios 2115. Si bien la canalización 3D 2112 se puede utilizar para realizar operaciones de medios, una realización del GPE 2110 también incluye una canalización de medios 2116 que se utiliza específicamente para realizar operaciones de medios, tales como postprocesamiento de vídeo y mejora de imagen.

En algunas realizaciones, la canalización de medios 2116 incluye funciones fijas o unidades lógicas programables para realizar una o más operaciones de medios especializadas, tales como aceleración de decodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar de, o en nombre del motor de códec de vídeo 2106. En algunas realizaciones, la canalización de medios 2116 incluye, además, una unidad de generación de hilos

para generar hilos para su ejecución en el subsistema 3D/Medios 2115. 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 2115.

En algunas realizaciones, el subsistema 3D/Medios 2115 incluye lógica para ejecutar hilos generados por la canalización 3D 2112 y la canalización de medios 2116. En una realización, las canalizaciones envían solicitudes de ejecución de hilos al subsistema 3D/Medios 2115, que incluye una lógica de despacho de hilos para arbitrar y enviar las diversas solicitudes a los recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos 3D y multimedia. En algunas realizaciones, el subsistema 3D/Medios 2115 incluye una o más memorias caché 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 para almacenar datos de salida.

Motor de procesamiento de gráficos

La **Figura 22** es un diagrama de bloques de un motor de procesamiento de gráficos 2210 de un procesador de gráficos de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento de gráficos (GPE) 2210 es una versión del GPE 2110 que se muestra en la **Figura 21**. Los elementos de la **Figura 22** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto. Por ejemplo, se ilustra la canalización de 3D 2112 y la canalización de medios 2116 de la **FIG. 21**. La canalización de medios 2116 es opcional en algunas realizaciones del GPE 2210 y puede no estar explícitamente incluida dentro del GPE 2210. Por ejemplo y en al menos una realización, se acopla al GPE 2210 un procesador de medios y/o imágenes independiente.

En algunas realizaciones, el GPE 2210 se acopla con, o incluye un transmisor de comandos 2203, que proporciona un flujo de comandos a la canalización 3D 2112 y/o a las canalizaciones de medios 2116. En algunas realizaciones, el transmisor de comandos 2203 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 2203 recibe comandos de la memoria y envía los comandos a la canalización 3D 2112 y/o a la canalización de medios 2116. Los comandos son directivas extraídas de una memoria intermedia en anillo, que almacena comandos para la canalización 3D 2112 y la canalización de medios 2116. En una realización, la memoria intermedia de anillo puede incluir, además, memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización 3D 2112 también pueden incluir referencias a datos almacenados en la memoria, tales como, entre otros, datos de vértices y geometría para la canalización 3D 2112 y/o datos de imagen y objetos de memoria para la canalización de medios 2116. La canalización 3D 2112 y la canalización de medios 2116 procesan los comandos y los datos realizando operaciones mediante lógica dentro de las respectivas canalizaciones o despachando uno o más hilos de ejecución a una matriz de núcleos gráficos 2214.

En varias realizaciones, la canalización 3D 2112 puede ejecutar uno o más programas de sombreado, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de computación u otros programas de sombreado, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleos gráficos 2214. La matriz de núcleos gráficos 2214 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución multipropósito (por ejemplo, unidades de ejecución) dentro de la matriz de núcleos gráficos 2214 incluye soporte para varios lenguajes de sombreado de API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunas realizaciones, la matriz de núcleos gráficos 2214 también incluye lógica de ejecución para realizar funciones multimedia, tales como procesamiento de vídeo 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 de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o en conjunto con la lógica de propósito general dentro del núcleo o núcleos del procesador 1907 de la **Figura 19** o el núcleo 2002A-2002N como en la **Figura 20**.

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleos gráficos 2214 pueden enviar datos a la memoria en una memoria intermedia de retorno unificada (URB) 2218. La URB 2218 puede almacenar datos para múltiples hilos. En algunas realizaciones, la URB 2218 se puede utilizar para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleos gráficos 2214. En algunas realizaciones, la URB 2218 puede usarse adicionalmente para la sincronización entre hilos en la matriz del núcleo de gráficos y la lógica de función fija dentro de la lógica de función compartida 2220.

En algunas realizaciones, la matriz de núcleos gráficos 2214 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 2210. 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 2214 se acopla con la lógica de función compartida 2220 que incluye múltiples recursos que se comparten entre los núcleos de gráficos en la matriz de núcleos gráficos. Las funciones compartidas dentro de la lógica de función compartida 2220 son unidades lógicas de hardware que proporcionan una funcionalidad complementaria especializada a la matriz de núcleos gráficos 2214. En diversas realizaciones, la lógica de funciones compartidas 2220 incluye, pero sin limitación, la lógica del muestreador 2221, del cálculo matemático 2222 y de la comunicación entre hilos (ITC) 2223. Adicionalmente, algunas realizaciones implementan una o más memorias caché 2225 dentro de la lógica de funciones compartidas 2220. Una función compartida se implementa cuando la demanda de una función especializada dada es insuficiente para su inclusión en la matriz de núcleos gráficos 2214. En su lugar, se implementa una única instanciación de dicha función especializada como una entidad independiente en la lógica de función compartida 2220 y se comparte entre los recursos de ejecución dentro de la matriz de núcleos gráficos 2214. El conjunto preciso de funciones que se comparten entre la matriz de núcleos gráficos 2214 y se incluyen dentro de la matriz de núcleos gráficos 2214 varía entre las realizaciones.

La **Figura 23** ilustra un diagrama de bloques de otra realización ejemplar de un procesador de gráficos 2300. Los elementos de la **Figura 23** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto.

En algunas realizaciones, el procesador de gráficos 2300 incluye una interconexión en anillo 2302, un extremo frontal de canalización 2305, un motor de medios 2337 y núcleos gráficos 2380A-2380N. En algunas realizaciones, la interconexión en anillo 2302 acopla el procesador de gráficos a otras unidades de procesamiento, incluidos otros procesadores de gráficos o uno o más núcleos de procesador de propósito general. En algunas realizaciones, el procesador de gráficos es uno de los muchos procesadores integrados dentro de un sistema de procesamiento multinúcleo.

En algunas realizaciones, el procesador de gráficos 2300 recibe lotes de comandos a través de la interconexión en anillo 2302. Los comandos entrantes son interpretados por un transmisor de comandos 2303 en el extremo frontal de la canalización 2305. En algunas realizaciones, el procesador de gráficos 2300 incluye una lógica de ejecución escalable para realizar el procesamiento de geometría 3D y el procesamiento de medios a través del núcleo o núcleos gráficos 2380A-2380N. Para los comandos de procesamiento de geometría 3D, el transmisor de comandos 2303 suministra comandos a la canalización de geometría 2336. Para al menos algunos comandos de procesamiento de medios, el transmisor de comandos 2303 suministra los comandos a un extremo frontal de vídeo 2334, que se acopla con un motor de medios 2337. En algunas realizaciones, el motor de medios 2337 incluye un motor de calidad de vídeo (VQE) 2330 para el postprocesamiento de vídeo e imágenes y un motor de codificación/decodificación multiformato (MFX) 2333 para proporcionar codificación y decodificación de datos de medios acelerada por hardware. En algunas realizaciones, la canalización de geometría 2336 y el motor de medios 2337 generan cada uno hilos de ejecución para los recursos de ejecución de hilos proporcionados por al menos un núcleo gráfico 2380A.

En algunas realizaciones, el procesador de gráficos 2300 incluye recursos de ejecución de hilos escalables que presentan núcleos modulares 2380A-2380N (a veces denominados secciones de núcleo), cada uno de los cuales tiene múltiples subnúcleos 2350A-2350N, 2360A-2360N (a veces denominados porciones de núcleo). En algunas realizaciones, el procesador de gráficos 2300 puede tener cualquier número de núcleos gráficos 2380A a 2380N. En algunas realizaciones, el procesador de gráficos 2300 incluye un núcleo gráfico 2380A que tiene al menos un primer subnúcleo 2350A y un segundo subnúcleo 2360A. En otras realizaciones, el procesador de gráficos es un procesador de baja potencia con un único subnúcleo (por ejemplo, 2350A). En algunas realizaciones, el procesador de gráficos 2300 incluye múltiples núcleos gráficos 2380A-2380N, incluyendo cada uno un conjunto de primeros subnúcleos 2350A-2350N y un conjunto de segundos subnúcleos 2360A-2360N. Cada subnúcleo del conjunto de primeros subnúcleos 2350A-2350N incluye al menos un primer conjunto de unidades de ejecución 2352A-2352N y muestreadores de medios/texturas 2354A-2354N. Cada subnúcleo del conjunto de segundos subnúcleos 2360A-2360N incluye al menos un segundo conjunto de unidades de ejecución 2362A-2362N y muestreadores 2364A-2364N. En algunas realizaciones, cada subnúcleo 2350A-2350N, 2360A-2360N comparte un conjunto de recursos compartidos 2370A-2370N. 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 de gráficos.

Unidades de ejecución

La **Figura 24** ilustra la lógica de ejecución de hilos 2400 que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones ejemplares de un GPE. Los elementos de la **Figura 24** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto.

En algunas realizaciones, la lógica de ejecución de hilos 2400 incluye un procesador de sombreado 2402, un despachador de hilos 2404, memoria caché de instrucciones 2406, una matriz de unidades de ejecución escalables que incluye una pluralidad de unidades de ejecución 2408A-2408N, un muestreador 2410, una memoria caché de

datos 2412 y un puerto de datos 2414. En una realización, la matriz de unidades de ejecución escalable puede escalarse dinámicamente habilitando o inhabilitando una o más unidades de ejecución (por ejemplo, cualquiera de las unidades de ejecución 2408A, 2408B, 2408C, 2408D a 2408N-1 y 2408N) basándose en los requisitos computacionales de una carga de trabajo. 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 2400 incluye una o más conexiones a la memoria, tales como la memoria del sistema o la memoria caché, a través de uno o más de la memoria caché de instrucciones 2406, el puerto de datos 2414, el muestreador 2410 y las unidades de ejecución 2408A-2408N. En algunas realizaciones, cada unidad de ejecución (por ejemplo, 2408A) es una unidad computacional de propósito general programable e independiente que es capaz de ejecutar múltiples hilos de hardware simultáneos mientras procesa múltiples elementos de datos en paralelo para cada hilo. En diversas realizaciones, la matriz de unidades de ejecución 2408A-2408N puede escalarse para incluir cualquier número de unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 2408A-2408N se utilizan principalmente para ejecutar programas de sombreado. Un procesador de sombreado 2402 puede procesar los diversos programas de sombreado y despachar hilos de ejecución asociados con los programas de sombreado a través de un despachador de hilos 2404. En una realización, el distribuidor de hilos incluye lógica para arbitrar solicitudes de inicio de hilo a partir de las canalizaciones de gráficos y de medios e instanciar los hilos solicitados en una o más unidades de ejecución de las unidades de ejecución 2408A-2408N. Por ejemplo, la canalización de geometría (por ejemplo, 2336 de la **Figura 23**) puede despachar los sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilo 2400 (**Figura 24**) para su procesamiento. En algunas realizaciones, el despachador de hilos 2404 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas de sombreado en ejecución.

En algunas realizaciones, las unidades de ejecución 2408A-2408N admiten 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 las bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución admiten el procesamiento de vértices y geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), el procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y el procesamiento de propósito general (por ejemplo, sombreadores de medios y de cómputo). Cada una de las unidades de ejecución 2408A-2408N es capaz de ejecutar múltiples instrucciones únicas y múltiples datos (SIMD) y la operación multiproceso permite un entorno de ejecución eficiente frente a accesos a memoria de mayor latencia. Cada hilo de hardware dentro de cada unidad de ejecución tiene un archivo de registro de alto ancho de banda dedicado y un estado de hilo independiente asociado. La ejecución es de múltiples emisiones por reloj a canalizaciones capaces de realizar operaciones de números enteros, de coma flotante de precisión sencilla y doble, capacidad de bifurcación de SIMD, operaciones lógicas, operaciones trascendentales y otras operaciones misceláneas. Mientras se esperan datos desde memoria o una de las funciones compartidas, una lógica de dependencia dentro de las unidades de ejecución 2408A-2408N hace que un hilo en espera pase a estar inactivo hasta que se hayan devuelto los datos solicitados. Mientras el hilo en espera está inactivo, los recursos de hardware pueden dedicarse a procesar otros hilos. Por ejemplo, durante un retraso asociado con una operación de sombreado de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, un sombreador de fragmentos u otro tipo de programa de sombreado, incluido un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 2408A-2408N opera sobre matrices de elementos de datos. El número de elementos de datos es el "tamaño de ejecución" o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso 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 coma flotante (FPU) físicas para un procesador de gráficos en particular. En algunas realizaciones, las unidades de ejecución 2408A-2408N admiten tipos de datos enteros y de coma flotante.

El conjunto de instrucciones de la unidad de ejecución incluye instrucciones SIMD. Los diversos elementos de datos se pueden almacenar como un tipo de datos empaquetados en un registro y la unidad de ejecución procesará los diversos elementos en función del tamaño de los 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 cuádruple palabra (QW, Quad-Word)), ocho elementos de datos empaquetados separados de 32 bits (elementos de datos de tamaño doble palabra (DW, Double Word)), dieciséis elementos de datos empaquetados separados de 16 bits (elementos de datos de tamaño palabra (W, Word)) 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.

Se incluyen una o más memorias caché de instrucciones internas (por ejemplo, 2406) en la lógica de ejecución de hilos 2400 para almacenar en memoria caché las instrucciones de hilos para las unidades de ejecución. En algunas realizaciones, se incluyen una o más memorias caché de datos (por ejemplo, 2412) para almacenar en memoria caché los datos de los hilos durante la ejecución de los mismos. En algunas realizaciones, se incluye un muestreador 2410 para proporcionar un muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunas realizaciones, el muestreador 2410 incluye una funcionalidad especializada de muestreo de texturas o

medios para procesar los datos de texturas 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 inicio de hilos a la lógica de ejecución de hilos 2400 a través de la lógica de generación y envío de hilos. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos en datos de píxeles, se invoca la lógica del procesador de píxeles (por ejemplo, lógica de sombreado de píxeles, lógica de sombreado de fragmentos, etc.) dentro del procesador de sombreado 2402 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 estarcido, etc.). En algunas realizaciones, el sombreador de píxeles o fragmentos calcula los valores de los diversos atributos de vértice que se van a interpolar en el objeto rasterizado. En algunas realizaciones, la lógica del procesador de píxeles dentro del procesador de sombreado 2402 a continuación ejecuta un programa de sombreador de píxeles o fragmentos suministrado por la interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreado de píxeles, el procesador de sombreado 2402 despacha hilos a una unidad de ejecución (por ejemplo, 2408A) a través del despachador de hilos 2404. En algunas realizaciones, el sombreador de píxeles 2402 utiliza la lógica de muestreo de textura en el muestreador 2410 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 2414 proporciona un mecanismo de acceso a la memoria para que la lógica de ejecución de hilos 2400 envíe datos procesados a la memoria para su procesamiento en una canalización de salida del procesador de gráficos. En algunas realizaciones, el puerto de datos 2414 incluye o se acopla a una o más memorias caché (por ejemplo, memoria caché de datos 2412) para almacenar en memoria caché de datos para el acceso a la memoria a través del puerto de datos.

La **Figura 25** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos 2500 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución del procesador de gráficos admiten un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los recuadros de línea continua 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 2500 descrito e ilustrado son macroinstrucciones, en que son instrucciones suministradas a la unidad de ejecución, a diferencia de las microoperaciones resultantes de la decodificación de la instrucción una vez que se procesa la instrucción.

En algunas realizaciones, las unidades de ejecución de procesador de gráficos soportan de manera nativa instrucciones en un formato de instrucción de 128 bits 2510. Un formato de instrucción compactado de 64 bits 2530 está disponible para algunas instrucciones basándose en la instrucción, las opciones de instrucción y el número de operandos seleccionados. El formato de instrucción de 128 bits nativo 2510 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de instrucción de 64 bits 2530. Las instrucciones nativas disponibles en el formato de instrucción de 64 bits 2530 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 2513. El hardware de la unidad de ejecución hace referencia a un conjunto de tablas de compactación en función de 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 2510.

Para cada formato, el código de operación de instrucción 2512 define la operación que la unidad de ejecución debe realizar. Las unidades de ejecución ejecutan cada instrucción en paralelo a través de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de suma, la unidad de ejecución realiza una operación de suma simultánea a través de cada canal de color que representa un elemento de textura o un elemento de imagen. De manera predeterminada, la unidad de ejecución realiza cada instrucción a través de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 2514 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 instrucciones en el formato de instrucción de 128 bits 2510, un campo de tamaño de ejecución 2516 limita la cantidad de canales de datos que se ejecutarán en paralelo. En algunas realizaciones, el campo de tamaño de ejecución 2516 no está disponible para su uso en el formato de instrucción compacta de 64 bits 2530.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos que incluyen dos operandos fuente, src0 2520, src1 2522 y un destino 2518. En algunas realizaciones, las unidades de ejecución admiten instrucciones de destino dual, donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando fuente (por ejemplo, SRC2 2524), donde el código de operación de instrucción 2512 determina la cantidad de operandos fuente. El último operando fuente de una instrucción puede ser un valor inmediato (por ejemplo, codificado de forma rígida) que se pasa con la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 2510 incluye un campo de modo de acceso/dirección 2526 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.

En algunas realizaciones, el formato de instrucción de 128 bits 2510 incluye un campo de modo de acceso/dirección 2526, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En una realización, el modo de acceso se utiliza para definir una alineación de acceso a datos para la instrucción. Algunas realizaciones admiten modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, donde la alineación de bytes del modo de acceso determina la alineación de acceso de los operandos de instrucción. Por ejemplo, cuando está en un primer modo, la instrucción puede utilizar direccionamiento alineado por bytes para los operandos fuente y destino y cuando está en un segundo modo, la instrucción puede utilizar direccionamiento alineado por 16 bytes para todos los operandos fuente y destino.

En una realización, la parte de modo de dirección del campo de modo de acceso/dirección 2526 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 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 en función de 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 código de operación 2512 para simplificar la decodificación del código de operación 2540. Para un código de operación de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de código de operación. La agrupación precisa de códigos de operación que se muestra es simplemente un ejemplo. En algunas realizaciones, un grupo de código de operación de movimiento y lógica 2542 incluye instrucciones lógicas y de movimiento de datos (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y lógica 2542 comparte los cinco bits más significativos (MSB), donde las instrucciones de movimiento (mov) tienen la forma 0000xxxxb y las instrucciones lógicas tienen la forma 0001xxxxb. Un grupo de instrucciones de control de flujo 2544 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en la forma 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 2546 incluye una mezcla de instrucciones, incluidas instrucciones de sincronización (por ejemplo, esperar, enviar) en la forma 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones matemáticas paralelas 2548 incluye instrucciones aritméticas por componente (por ejemplo, suma, multiplicación (mul)) en la forma 0100xxxxb (por ejemplo, 0x40). El grupo matemático paralelo 2548 realiza las operaciones aritméticas en paralelo a través de los canales de datos. El grupo matemático vectorial 2550 incluye instrucciones aritméticas (por ejemplo, dp4) en la forma de 0101xxxxb (por ejemplo, 0x50). El grupo de matemáticas vectoriales realiza operaciones aritméticas, tales como cálculos de producto escalar, sobre operandos vectoriales.

Canalización de gráficos

La **Figura 26** es un diagrama de bloques de otra realización de un procesador de gráficos 2600. Los elementos de la **Figura 26** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto.

En algunas realizaciones, el procesador de gráficos 2600 incluye una canalización de gráficos 2620, una canalización de medios 2630, un motor de visualización 2640, una lógica de ejecución de subprocesos 2650 y una canalización de salida de renderizado 2670. En algunas realizaciones, el procesador de gráficos 2600 es un procesador de gráficos dentro de un sistema de procesamiento multinúcleo que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos se controla mediante escrituras de registros en uno o más registros de control (no se muestran) o mediante comandos emitidos al procesador de gráficos 2600 a través de una interconexión de anillo 2602. En algunas realizaciones, la interconexión de anillo 2602 acopla el procesador de gráficos 2600 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de propósito general. Los comandos de la interconexión de anillo 2602 son interpretados por un transmisor de comandos 2603, que suministra instrucciones a componentes individuales de la canalización de gráficos 2620 o la canalización de medios 2630.

En algunas realizaciones, el transmisor de comandos 2603 dirige la operación de un extractor de vértices 2605 que lee datos de vértices de la memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor de comandos 2603. En algunas realizaciones, el extractor de vértices 2605 proporciona datos de vértices a un sombreador de vértices 2607, que realiza operaciones de transformación de espacio de coordenadas e iluminación para cada vértice. En algunas realizaciones, el extractor de vértices 2605 y el sombreador de vértices 2607 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a unidades de ejecución 2652A, 2652B mediante un despachador de hilos 2631.

En algunas realizaciones, las unidades de ejecución 2652A-2652B son una matriz de procesadores vectoriales que tienen un conjunto de instrucciones para realizar operaciones gráficas y multimedia. En algunas realizaciones, las unidades de ejecución 2652A-2652B tienen una memoria caché de L1 acoplada 2651 que es específica para cada matriz o que se comparte entre las matrices. La memoria caché se puede configurar como una memoria caché de

datos, una memoria caché de instrucciones o una única memoria caché que está dividida para contener datos e instrucciones en diferentes particiones.

En algunas realizaciones, la canalización de gráficos 2620 incluye componentes de teselación para realizar la teselación acelerada por hardware de objetos 3D. En algunas realizaciones, un sombreador de casco programable 2611 configura las operaciones de teselación. Un sombreador de dominio programable 2617 proporciona una evaluación de extremo posterior de la salida de la teselación. Un teselador 2613 opera bajo la dirección del sombreador de casco 2611 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 2620. En algunas realizaciones, si no se utiliza la teselación, se pueden omitir los componentes de teselación (por ejemplo, sombreador de casco 2611, teselador 2613, sombreador de dominio 2617).

En algunas realizaciones, un sombreador de geometría 2619 puede procesar objetos geométricos completos a través de uno o más hilos despachados a las unidades de ejecución 2652A-2652B, o puede proceder directamente al recortador 2629. 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 2619 recibe la entrada del sombreador de vértices 2607. En algunas realizaciones, el sombreador de geometría 2619 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 2629 procesa los datos de vértice. El recortador 2629 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 de rasterizador y prueba de profundidad 2673 en la canalización de salida de renderizado 2670 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxeles. En algunas realizaciones, la lógica de sombreador de píxeles está incluida en la lógica de ejecución de hilos 2650. En algunas realizaciones, una aplicación puede omitir el componente de rasterizador y prueba de profundidad 2673 y acceder a datos de vértices no rasterizados mediante una unidad de salida de flujo 2623.

El procesador de gráficos 2600 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 2652A-2652B y la(s) memoria(s) caché asociada(s) 2651, el muestreador de texturas y medios 2654 y la memoria caché de texturas/muestreadores 2658 se interconectan a través de un puerto de datos 2656 para realizar el acceso a la memoria y comunicarse con los componentes de la canalización de salida de renderizado del procesador. En algunas realizaciones, el muestreador 2654, las memorias caché 2651, 2658 y las unidades de ejecución 2652A-2652B tienen cada uno rutas de acceso a memoria independientes.

En algunas realizaciones, la canalización de salida de renderizado 2670 contiene un componente de rasterizador y prueba de profundidad 2673 que convierte objetos basados en vértices en una representación asociada basada en píxeles. En algunas realizaciones, la lógica del rasterizador incluye una unidad de enmascaramiento/ventana para realizar la rasterización de triángulos y líneas de función fija. En algunas realizaciones también están disponibles una memoria caché de renderizado 2678 asociada y una memoria caché de profundidad 2679. Un componente de operaciones de píxeles 2677 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 mezcla) son realizadas por el motor 2D 2641, o sustituidas en el momento de la visualización por el controlador de visualización 2643 utilizando planos de visualización superpuestos. En algunas realizaciones, una memoria caché L3 compartida 2675 está disponible para todos los componentes gráficos, lo que permite compartir datos sin el uso de la memoria principal del sistema.

En algunas realizaciones, la canalización de medios 2630 del procesador de gráficos incluye un motor de medios 2637 y un extremo frontal de vídeo 2634. En algunas realizaciones, el extremo frontal de vídeo 2634 recibe comandos de canalización del transmisor de comandos 2603. En algunas realizaciones, la canalización de medios 2630 incluye un transmisor de comandos independiente. En algunas realizaciones, el extremo frontal de vídeo 2634 procesa comandos de medios antes de enviar el comando al motor de medios 2637. En algunas realizaciones, el motor de medios 2637 incluye una funcionalidad de generación de hilos para generar hilos para su envío a la lógica de ejecución de hilos 2650 a través del despachador de hilos 2631.

En algunas realizaciones, el procesador de gráficos 2600 incluye un motor de visualización 2640. En algunas realizaciones, el motor de visualización 2640 es externo al procesador 2600 y se acopla con el procesador de gráficos a través de la interconexión de anillo 2602, o algún otro bus o estructura de interconexión. En algunas realizaciones, el motor de visualización 2640 incluye un motor 2D 2641 y un controlador de visualización 2643. En algunas realizaciones, el motor de visualización 2640 contiene una lógica de propósito especial capaz de funcionar independientemente de la secuencia de procesamiento 3D. En algunas realizaciones, el controlador de visualización 2643 se acopla con un dispositivo de visualización (no mostrado), que puede ser un dispositivo de visualización integrado en el sistema, como un ordenador portátil, o un dispositivo de visualización externo conectado a través de un conector de dispositivo de visualización.

En algunas realizaciones, la canalización de gráficos 2620 y la canalización de medios 2630 se pueden configurar para realizar operaciones basadas en múltiples interfaces de programación de gráficos y medios y no son específicas de ninguna interfaz de programación de aplicaciones (API). En algunas realizaciones, el software del controlador para el procesador de gráficos traduce las llamadas de API que son específicas de una biblioteca de gráficos o medios en particular en comandos que pueden ser procesados por el procesador de gráficos. En algunas realizaciones, se proporciona soporte para Open Graphics Library (OpenGL) y Open Computing Language (OpenCL) y/o API de gráficos y computación Vulkan, todas del Grupo Khronos. En algunas realizaciones, también se puede proporcionar soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, se puede proporcionar soporte a una combinación de estas bibliotecas. También se puede proporcionar soporte para Open Source Computer Vision Library (OpenCV). Una futura API con una canalización 3D compatible también sería compatible si se puede realizar un mapeo desde la canalización de la futura API a la canalización del procesador de gráficos.

Programación de canalización de gráficos

La **Figura 27A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos 2700 de acuerdo con algunas realizaciones. La **Figura 27B** es un diagrama de bloques que ilustra una secuencia de orden de procesador de gráficos 2710 de acuerdo con una realización. Los cuadros con línea continua en la **Figura 27A** ilustran los componentes que se incluyen, en general, en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de los comandos de gráficos. El formato de comando de procesador de gráficos 2700 ejemplar de la **Figura 27A** incluye campos de datos para identificar un cliente objetivo 2702 del comando, un código de operación del comando (opcode) 2704 y los datos relevantes 2706 para el comando. En algunos comandos también se incluyen un subcódigo de operación 2705 y un tamaño de comando 2708.

En algunas realizaciones, el cliente 2702 especifica la unidad cliente del dispositivo gráfico que procesa los datos del comando. En algunas realizaciones, un analizador de comandos del procesador de gráficos examina el campo cliente de cada comando para condicionar el procesamiento posterior del comando y enrutar los datos del comando a la unidad cliente adecuada. En algunas realizaciones, las unidades cliente del procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de renderizado, una unidad 2D, una unidad 3D y una unidad multimedia. Cada unidad cliente tiene una canalización de procesamiento correspondiente que procesa los comandos. Una vez que la unidad cliente recibe el comando, la unidad cliente lee el código de operación 2704 y, si está presente, el subcódigo de operación 2705 para determinar la operación a realizar. La unidad cliente ejecuta el comando utilizando información en el campo de datos 2706. Para algunos comandos, se espera que un tamaño de comando explícito 2708 especifique el tamaño del comando. En algunas realizaciones, el analizador sintáctico de comandos determina automáticamente el tamaño de al menos algunos de los comandos en función del código de operación del comando. En algunas realizaciones, los comandos se alinean mediante múltiplos de una palabra doble.

El diagrama de flujo en la **Figura 27B** muestra una secuencia de comandos de procesador de gráficos 2710 ilustrativo. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que presenta una realización de un procesador de gráficos 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 se limitan a estos comandos específicos ni a esta secuencia de comandos. Además, los comandos se pueden emitir como lotes de comandos en una secuencia de comandos, de modo que el procesador de gráficos procesará la secuencia de comandos al menos parcialmente de forma simultánea.

En algunas realizaciones, la secuencia de comandos del procesador de gráficos 2710 puede comenzar con un comando de vaciado de canalización 2712 para hacer que cualquier canalización de gráficos activa complete los comandos pendientes actualmente para la canalización. En algunas realizaciones, la canalización 3D 2722 y la canalización de medios 2724 no funcionan simultáneamente. El vaciado de canalización se realiza para hacer que la canalización de gráficos activa complete los comandos pendientes. En respuesta a un vaciado de canalización, el analizador sintáctico de comandos para el procesador de gráficos pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las memorias caché de lectura relevantes. Opcionalmente, cualquier dato en la memoria caché de renderizado que esté marcado como "sucio" puede ser descargado a memoria. En algunas formas de realización, la orden de descarga de canal 2712 se puede utilizar para la sincronización de canales o antes de colocar el procesador de gráficos en un estado de bajo consumo.

En algunas formas de realización, una orden de selección de canal 2713 se utiliza cuando una secuencia de órdenes requiere que el procesador de gráficos cambie explícitamente entre canales. En algunas formas de realización, una orden de selección de canal 2713 se requiere sólo una vez dentro de un contexto de ejecución antes de emitir órdenes de canal a menos que el contexto sea para emitir órdenes para ambas tuberías. En algunas formas de realización, se requiere una orden de descarga de canal 2712 inmediatamente antes de un cambio de canal a través de la orden de selección de canal 2713.

En algunas formas de realización, una orden de control de canal 2714 configura un canal de gráficos para su operación y se utiliza para programar el canal 3D 2722 y el canal de medios 2724. En algunas formas de realización, la orden de

control de canal 2714 configura el estado de canal para el canal activa. En una forma de realización, la orden de control de canal 2714 se utiliza para la sincronización de canal y para borrar datos de una o más memorias caché dentro del canal activa antes de procesar un lote de órdenes.

En algunas realizaciones, los comandos para el estado de la memoria intermedia de retorno 2716 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 de gráficos también utiliza una o más memorias intermedias de retorno para almacenar datos de salida y para realizar comunicación entre hilos. En algunas realizaciones, la configuración del estado de la memoria intermedia de retorno 2716 incluye la selección del tamaño y la cantidad de memorias intermedias de retorno que se utilizarán para un conjunto de operaciones de canalización.

Los comandos restantes en la secuencia de comandos difieren en función de la canalización activa para las operaciones. En función de una determinación de canalización 2720, la secuencia de comandos se adapta a la canalización 3D 2722 comenzando con el estado de canalización 3D 2730, o la canalización de medios 2724 comenzando en el estado de canalización de medios 2740.

Los comandos para configurar el estado de canalización 3D 2730 incluyen comandos de configuración de estado 3D para el estado de la memoria intermedia de vértice, el estado del elemento de vértice, el estado de color constante, el estado de la 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 2730 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 2732 se utiliza para enviar primitivas 3D para que sean procesados por la canalización 3D. Los comandos y los parámetros asociados que se pasan al procesador de gráficos a través del comando de primitiva 3D 2732 se reenvían a la función de búsqueda de vértices en la canalización de gráficos. La función de extracción de vértices utiliza los datos del comando de primitiva 3D 2732 para generar estructuras de datos de vértices. Las estructuras de datos de vértices se almacenan en una o más memorias intermedias de retorno. En algunas realizaciones, el comando de primitiva 3D 2732 se utiliza para realizar operaciones de vértice en primitivas 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la canalización 3D 2722 despacha hilos de ejecución de sombreadores a unidades de ejecución de procesadores de gráficos.

En algunas realizaciones, la canalización 3D 2722 se activa a través de un comando o evento de ejecución 2734. En algunas realizaciones, una escritura de registro activa la ejecución del 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 del 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 que se completan las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles colorea los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo posterior de píxeles para esas operaciones.

En algunas realizaciones, la secuencia de comandos del procesador de gráficos 2710 sigue la ruta de la canalización de medios 2724 al realizar operaciones de medios. En general, el uso específico y la manera de programar para la canalización de medios 2724 dependen de los medios o las operaciones de cómputo que se realizarán. Las operaciones de decodificación de medios específicas se pueden descargar a 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 en su totalidad o en parte 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 la unidad de procesador de gráficos de propósito general (GPGPU), donde el procesador de gráficos se utiliza para realizar operaciones vectoriales SIMD utilizando programas de sombreado computacional que no están relacionados explícitamente con la representación de primitivas de gráficos.

En algunas realizaciones, la canalización de medios 2724 está configurada de una manera similar a la canalización 3D 2722. Se despacha o se coloca un conjunto de comandos para configurar el estado de canalización de medios 2740 en una cola de comandos antes de los comandos de objeto de medios 2742. En algunas realizaciones, los comandos para el estado de canalización de medios 2740 incluyen datos para configurar los elementos de canalización de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación y codificación de vídeo dentro de la canalización de medios, tal como el formato de codificación o decodificación. En algunas realizaciones, los comandos para el estado de canalización de medios 2740 también soportan el uso de uno o más punteros a elementos de estado "indirecto" que contienen un lote de ajustes de estado.

En algunas realizaciones, los comandos de objetos de medios 2742 proporcionan punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo que se procesarán. En algunas realizaciones, todos los estados de la canalización de medios deben ser válidos antes de emitir un comando de objeto de medios 2742. Una vez que el estado de la canalización está configurado y los comandos de objeto de medios 2742 están en cola, la canalización de medios 2724 se activa a través de un comando de ejecución 2744 o un evento de ejecución equivalente (por ejemplo, escritura de registro). La salida de la canalización de medios 2724 puede entonces ser procesada posteriormente por operaciones proporcionadas por la canalización 3D 2722 o la canalización de medios 2724. En algunas realizaciones, las operaciones GPGPU se configuran y ejecutan de una manera similar a las operaciones de medios.

Arquitectura de software de gráficos

La **Figura 28** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos 2800 de acuerdo con algunas realizaciones. En algunas realizaciones, la arquitectura de software incluye una aplicación de gráficos 3D 2810, un sistema operativo 2820 y al menos un procesador 2830. En algunas realizaciones, el procesador 2830 incluye un procesador de gráficos 2832 y uno o más núcleos de procesador de propósito general 2834. La aplicación de gráficos 2810 y el sistema operativo 2820 se ejecutan cada uno en la memoria del sistema 2850 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación de gráficos 3D 2810 contiene uno o más programas de sombreado que incluyen instrucciones de sombreado 2812. Las instrucciones del lenguaje de sombreado pueden estar en un lenguaje de sombreado de alto nivel, tal como el lenguaje de sombreado de alto nivel (HLSL) o el lenguaje de sombreado OpenGL (GLSL). La aplicación también incluye instrucciones ejecutables 2814 en un lenguaje de máquina adecuado para su ejecución por el núcleo de procesador de propósito general 2834. La aplicación también incluye objetos gráficos 2816 definidos por datos de vértice.

En algunas realizaciones, el sistema operativo 2820 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 2820 puede soportar una API de gráficos 2822 como la API Direct3D, la API OpenGL o la API Vulkan. Cuando se utiliza la API Direct3D, el sistema operativo 2820 utiliza un compilador de sombreado de extremo frontal 2824 para compilar cualquier instrucción de sombreado 2812 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 sombreado. En algunas realizaciones, los sombreadores de alto nivel se compilan en sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 2810. En algunas realizaciones, las instrucciones de sombreado 2812 se proporcionan en una forma intermedia, tal como una versión de la Representación Intermedia Portátil Estándar (SPIR) utilizada por la API de Vulkan.

En algunas realizaciones, el controlador gráfico de modo de usuario 2826 contiene un compilador de sombreadores de extremo posterior 2827 para convertir las instrucciones de sombreado 2812 en una representación específica de hardware. Cuando se utiliza la API OpenGL, las instrucciones de sombreado 2812 en el lenguaje de alto nivel GLSL se pasan a un controlador de gráficos de modo de usuario 2826 para su compilación. En algunas realizaciones, el controlador de gráficos de modo de usuario 2826 utiliza funciones de modo kernel del sistema operativo 2828 para comunicarse con un controlador de gráficos de modo kernel 2829. En algunas realizaciones, el controlador de gráficos de modo kernel 2829 se comunica con el procesador de gráficos 2832 para enviar comandos e instrucciones.

Implementaciones de núcleo PI

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 diversas lógicas dentro del procesador. Cuando las lee una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en este documento. Dichas representaciones, conocidas como "núcleos PI", son unidades reutilizables de lógica para un circuito integrado que se pueden almacenar en un medio tangible 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 se puede fabricar de manera que el circuito realice operaciones descritas en asociación con cualquiera de las realizaciones descritas en este documento.

La **Figura 29** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo PI 2900 que puede usarse para fabricar un circuito integrado para realizar las operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo PI 2900 puede usarse para generar diseños reutilizables modulares que pueden incorporarse en un diseño más grande o usarse para construir un circuito integrado entero (por ejemplo, un circuito de SOC integrado). Una instalación de diseño 2930 puede generar una simulación de software 2910 de un diseño de núcleo PI en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 2910 se puede utilizar para diseñar, probar y verificar el comportamiento del núcleo PI utilizando un modelo de simulación 2912. El modelo de

simulación 2912 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. A continuación, se puede crear o sintetizar un diseño de nivel de transferencia de registro (RTL) 2915 a partir del modelo de simulación 2912. El diseño RTL 2915 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 2915, también se pueden crear, diseñar o sintetizar diseños de nivel inferior a nivel lógico o nivel de transistor. Por lo tanto, los detalles particulares del diseño y simulación iniciales pueden variar.

El diseño RTL 2915 o equivalente puede ser sintetizado adicionalmente por la instalación de diseño en un modelo de hardware 2920, que puede estar en un lenguaje de descripción de hardware (HDL), o alguna otra representación de datos de diseño físico. El HDL puede ser simulado o probado adicionalmente para verificar el diseño de núcleo PI. El diseño de núcleo PI se puede almacenar para su entrega a una instalación de fabricación de terceros 2965 usando la memoria no volátil 2940 (por ejemplo, disco duro, memoria flash o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño del núcleo PI se puede transmitir (por ejemplo, a través de Internet) a través de una conexión por cable 2950 o una conexión inalámbrica 2960. La instalación de fabricación 2965 puede entonces fabricar un circuito integrado que se basa al menos en parte en el diseño del núcleo PI. El circuito integrado fabricado se puede configurar para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Circuito integrado de sistema en chip ejemplar

Las **Figuras 30-32** ilustran circuitos integrados ilustrativos y procesadores de gráficos asociados que pueden fabricarse usando uno o más núcleos PI, 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 30** es un diagrama de bloques que ilustra un circuito integrado de sistema en chip 3000 ejemplar que puede fabricarse utilizando uno o más núcleos PI, según una realización. El circuito integrado 3000 ejemplar incluye uno o más procesadores de aplicaciones 3005 (por ejemplo, CPU), al menos un procesador de gráficos 3010, y puede incluir adicionalmente un procesador de imágenes 3015 y/o un procesador de vídeo 3020, cualquiera de los cuales puede ser un núcleo PI modular de la misma o de múltiples instalaciones de diseño diferentes. El circuito integrado 3000 incluye una lógica de bus o de periféricos que incluye un controlador de USB 3025, un controlador de UART 3030, un controlador de SPI/SDIO 3035 y un controlador de I^2S/I^2C 3040. Además, el circuito integrado puede incluir un dispositivo de visualización 3045 acoplado a uno o más de un controlador de interfaz multimedia de alta definición (HDMI) 3050 y una interfaz de visualización de interfaz de procesador de la industria móvil (MIPI) 3055. El almacenamiento puede ser proporcionado por un subsistema de memoria flash 3060 que incluye memoria flash y un controlador de memoria flash. La interfaz de memoria puede ser proporcionada a través de un controlador de memoria 3065 para acceder a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 3070.

La **Figura 31** es un diagrama de bloques que ilustra un procesador de gráficos 3110 ejemplar de un circuito integrado de sistema en un chip que puede fabricarse utilizando uno o más núcleos PI, de acuerdo con una realización. El procesador de gráficos 3110 puede ser una variante del procesador de gráficos 3010 de la **Figura 30**. El procesador de gráficos 3110 incluye un procesador de vértices 3105 y uno o más procesadores de fragmentos 3115A-3115N (por ejemplo, 3115A, 3115B, 3115C, 3115D a 3115N-1 y 3115N). El procesador de gráficos 3110 puede ejecutar diferentes programas de sombreado a través de una lógica separada, de modo que el procesador de vértices 3105 está optimizado para ejecutar operaciones para programas de sombreado de vértices, mientras que uno o más procesadores de fragmentos 3115A-3115N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreado de fragmentos o píxeles. El procesador de vértices 3105 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 3115A-3115N utilizan los datos de primitivas y de vértice generados por el procesador de vértices 3105 para producir una memoria intermedia de tramas que se muestra en un dispositivo de visualización. En una realización, el procesador o procesadores de fragmentos 3115A-3115N están optimizados para ejecutar programas de sombreado de fragmentos según lo previsto en la API OpenGL, que pueden usarse para realizar operaciones similares a las de un programa de sombreado de píxeles según lo previsto en la API Direct 3D.

El procesador de gráficos 3110 incluye adicionalmente una o más unidades de administración de memoria (MMU) 3120A-3120B, memoria o memorias caché 3125A-3125B e interconexión o interconexiones de circuitos 3130A-3130B. La una o más MMU 3120A-3120B proporcionan un mapeo de direcciones virtuales a físicas para el procesador de gráficos 3110, incluyendo el procesador de vértices 3105 y/o los procesadores de fragmentos 3115A-3115N, 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 memorias caché 3125A-3125B. En una realización, la una o más MMU 3120A-3120B pueden estar sincronizadas con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con el uno o más procesadores de aplicación 3005, procesador de imagen 3015 y/o procesador de vídeo 3020 de la **Figura 30**, de modo que cada procesador 3005-3020 pueda participar en un sistema de memoria virtual compartido o unificado. La una o más interconexiones de circuito 3130A-3130B permiten que el procesador de gráficos 3110 interactúe con otros núcleos PI 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 32** es un diagrama de bloques que ilustra un procesador de gráficos 3210 ejemplar de un circuito integrado de sistema en un chip que puede fabricarse utilizando uno o más núcleos PI, de acuerdo con una realización. El procesador de gráficos 3210 puede ser una variante del procesador de gráficos 3010 de la **Figura 30**. El procesador de gráficos 3210 incluye la una o más MMU 3120A-3120B, memoria o memorias caché 3125A-3125B e interconexión o interconexiones de circuito 3130A-3130B del circuito integrado 3100 de la **Figura 31**.

El procesador de gráficos 3210 incluye uno o más núcleo o núcleos de sombreador 3215A-3215N (por ejemplo, 3215A, 3215B, 3215C, 3215D, 3215E, 3215F, a 3215N-1 y 3115N), que proporciona una arquitectura de núcleo de sombreado unificada en la que un solo núcleo o tipo o núcleo puede ejecutar todo tipo de código de sombreado programable, incluyendo 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 de gráficos 3210 incluye un administrador de tareas entre núcleos 3205, que actúa como un despachador de hilos para despachar hilos de ejecución a uno o más núcleos de sombreado 3215A-3215N y una unidad de mosaico 3218 para acelerar las operaciones de mosaico para el renderizado basado en mosaicos, en el que las operaciones de renderizado para una escena se subdividen en el espacio de la imagen, por ejemplo para explotar la coherencia espacial local dentro de una escena o para optimizar el uso de memorias caché internas.

REIVINDICACIONES

1. Procedimiento de procesamiento de imágenes ejecutado por un procesador de gráficos, que comprende:
submuestrear una imagen de entrada (1402, 1502, 1504) en imágenes más pequeñas con una resolución menor que la imagen de entrada (1402, 1502, 1504);
concatenar las imágenes más pequeñas;
introducir las imágenes más pequeñas concatenadas en una red neuronal convolucional (CNN) que comprende una pluralidad de capas de convolución y agrupamiento y una pluralidad de grupos de capas completamente conectadas; en el que
la pluralidad de capas de convolución y agrupamiento incluye una última capa de convolución y agrupamiento que tiene una pluralidad de subconjuntos de nodos, siendo las salidas de cada subconjunto de nodos enviadas a un respectivo grupo de capas completamente conectadas, proporcionando cada grupo de capas completamente conectadas una salida que incluye funciones Softmax de N vías que producen una distribución sobre N etiquetas de clase;
combinar las salidas de cada uno de los grupos de capas completamente conectadas para proporcionar un resultado de clasificación final; y
generar el resultado de la clasificación final.

2. Almacenamiento legible por máquina que incluye instrucciones legibles por máquina que, cuando las ejecuta un procesador de gráficos, hacen que el procesador de gráficos implemente un procedimiento como se reivindica en la reivindicación 1.

3. Sistema (100, 800, 1020, 1100, 1400, 1500, 1800, 1900, 2800), que comprende:
un núcleo de procesamiento que incluye almacenamiento de memoria para almacenar una imagen de entrada (1402, 1502, 1504) con una resolución completa;
un concentrador de controlador de E/S acoplado al núcleo de procesamiento para proporcionar acceso a la red y al almacenamiento de datos al núcleo de procesamiento; y
un procesador de gráficos (208, 300, 500, 800, 1908, 1912, 2008, 2100, 2300, 2600, 2832, 3010, 3110, 3210) acoplado al concentrador del controlador de E/S (1930) y configurado para ejecutar el procedimiento de la reivindicación 1.

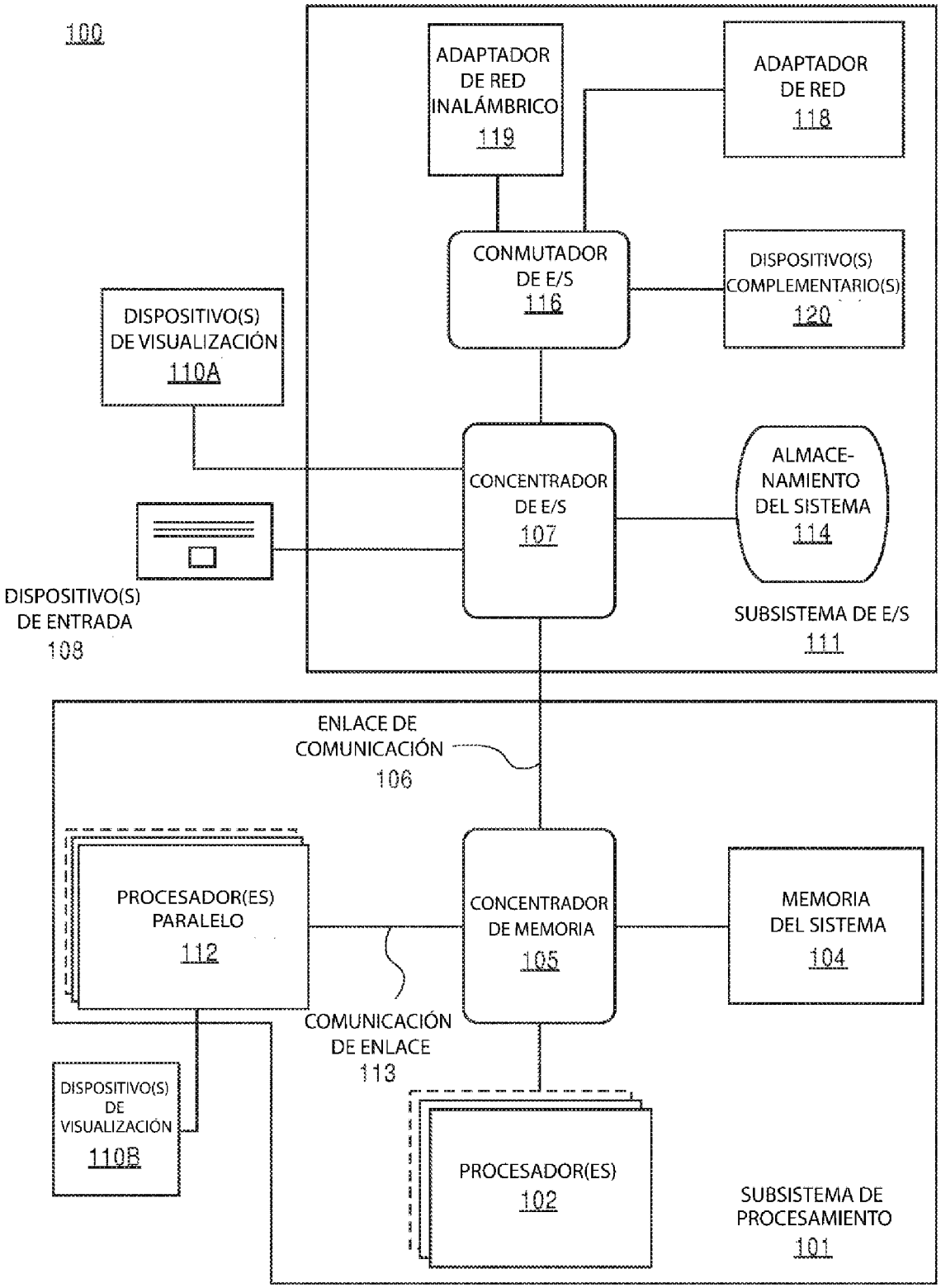

100
ADAPTADOR DE RED INALÁMBRICO 119
ADAPTADOR DE RED 118
CONMUTADOR DE E/S 116
DISPOSITIVO(S) COMPLEMENTARIO(S) 120
DISPOSITIVO(S) DE VISUALIZACIÓN 110A
CONCENTRADOR DE E/S 107
ALMACE-NAMIENTO DEL SISTEMA 114
DISPOSITIVO(S) DE ENTRADA 108
SUBSISTEMA DE E/S 111
ENLACE DE COMUNICACIÓN 106
PROCESADOR(ES) PARALELO 112
CONCENTRADOR DE MEMORIA 105
MEMORIA DEL SISTEMA 104
COMUNICACIÓN DE ENLACE 113
DISPOSITIVO(S) DE VISUALIZACIÓN 110B
PROCESADOR(ES) 102
SUBSISTEMA DE PROCESAMIENTO 101

FIG. 1

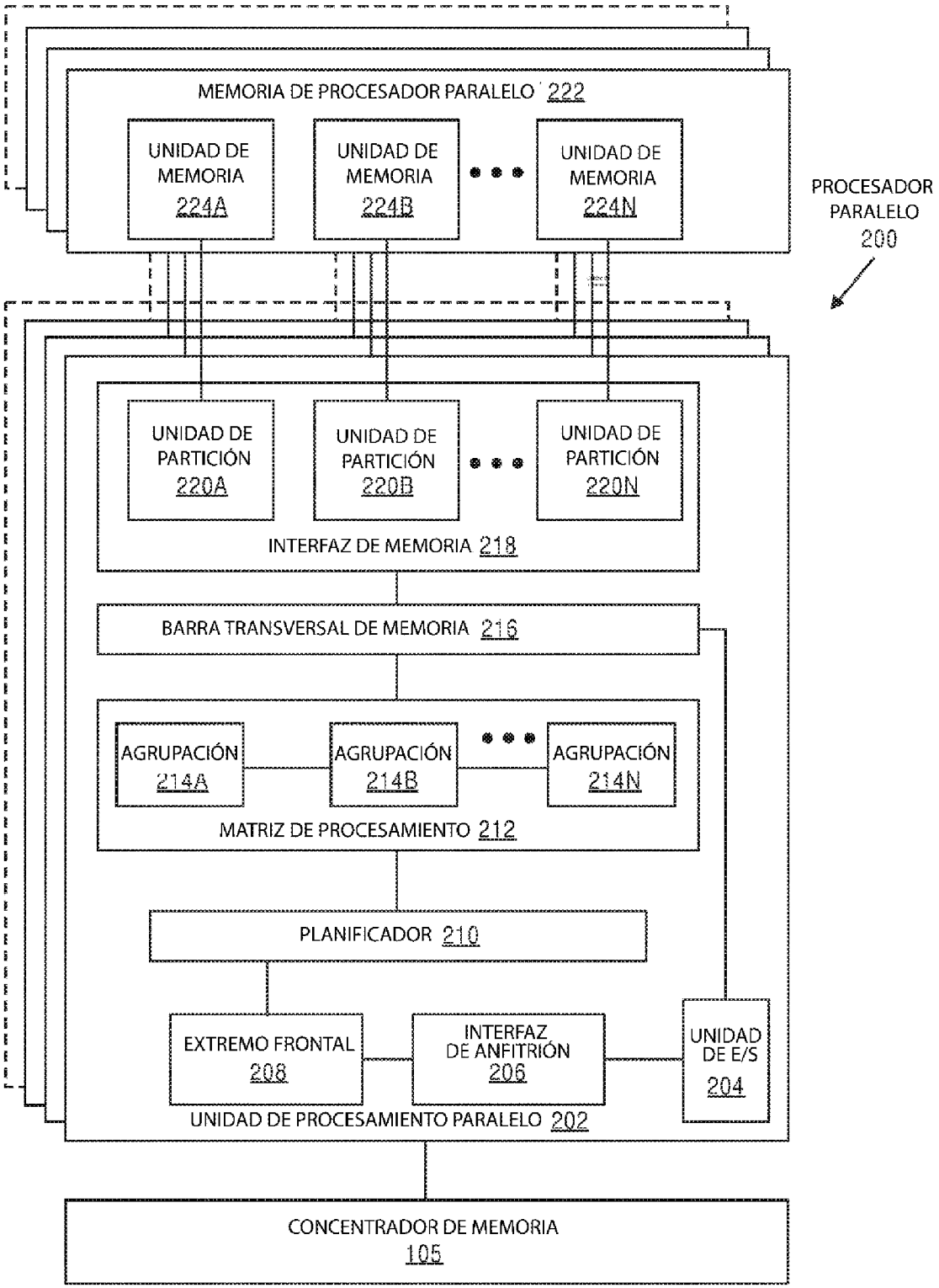

MEMORIA DE PROCESADOR PARALELO 222
UNIDAD DE MEMORIA 224A
UNIDAD DE MEMORIA 224B
UNIDAD DE MEMORIA 224N
PROCESADOR PARALELO 200
UNIDAD DE PARTICIÓN 220A
UNIDAD DE PARTICIÓN 220B
UNIDAD DE PARTICIÓN 220N
INTERFAZ DE MEMORIA 218
BARRA TRANSVERSAL DE MEMORIA 216
AGRUPACIÓN 214A
AGRUPACIÓN 214B
AGRUPACIÓN 214N
MATRIZ DE PROCESAMIENTO 212
PLANIFICADOR 210
EXTREMO FRONTAL 208
INTERFAZ DE ANFITRIÓN 206
UNIDAD DE E/S 204
UNIDAD DE PROCESAMIENTO PARALELO 202
CONCENTRADOR DE MEMORIA 105

FIG. 2A

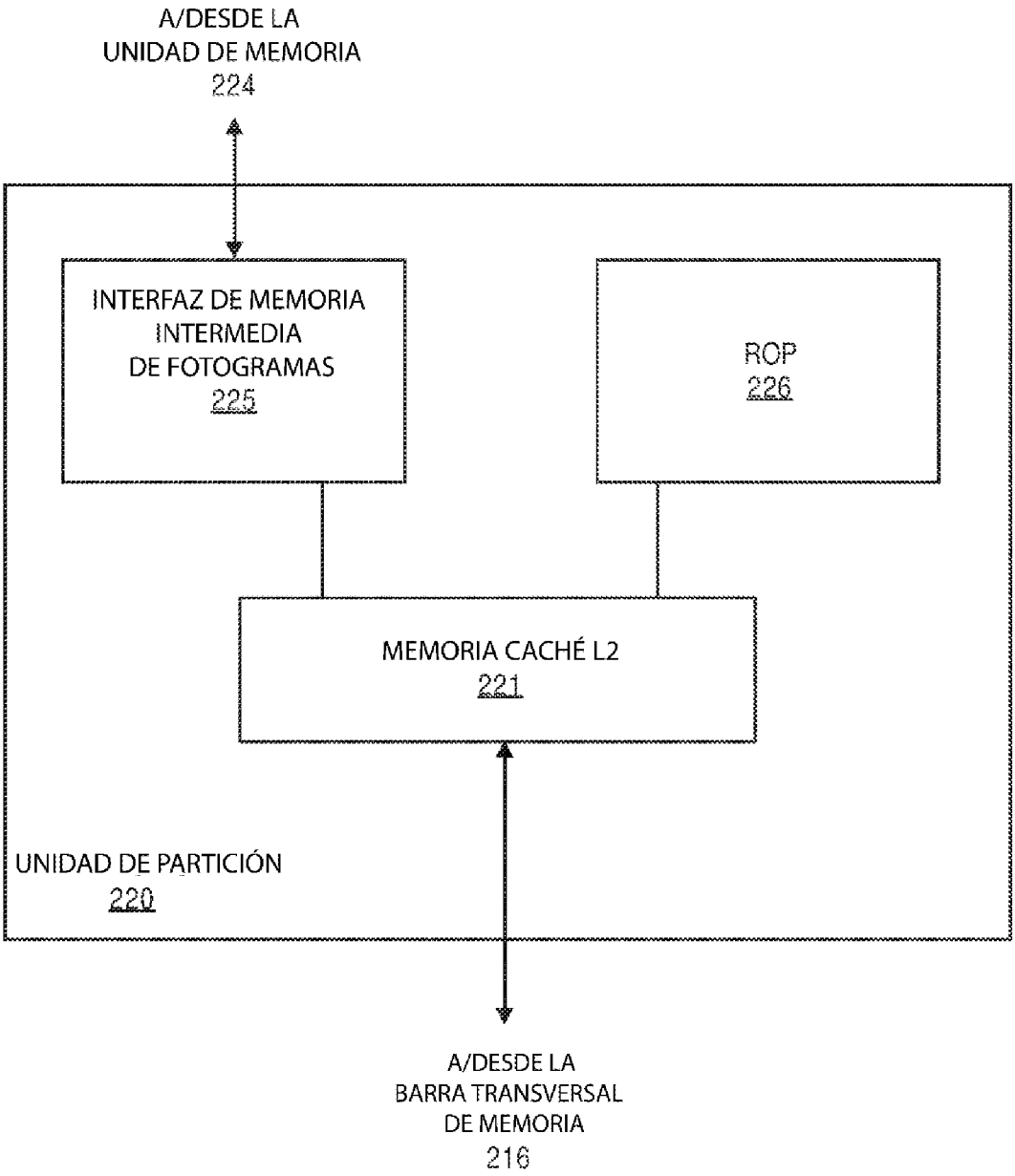

A/DESDE LA
UNIDAD DE MEMORIA
224
INTERFAZ DE MEMORIA
INTERMEDIA
DE FOTOGRAMAS
225
ROP
226
MEMORIA CACHÉ L2
221
UNIDAD DE PARTICIÓN
220
A/DESDE LA
BARRA TRANSVERSAL
DE MEMORIA
216

FIG. 2B

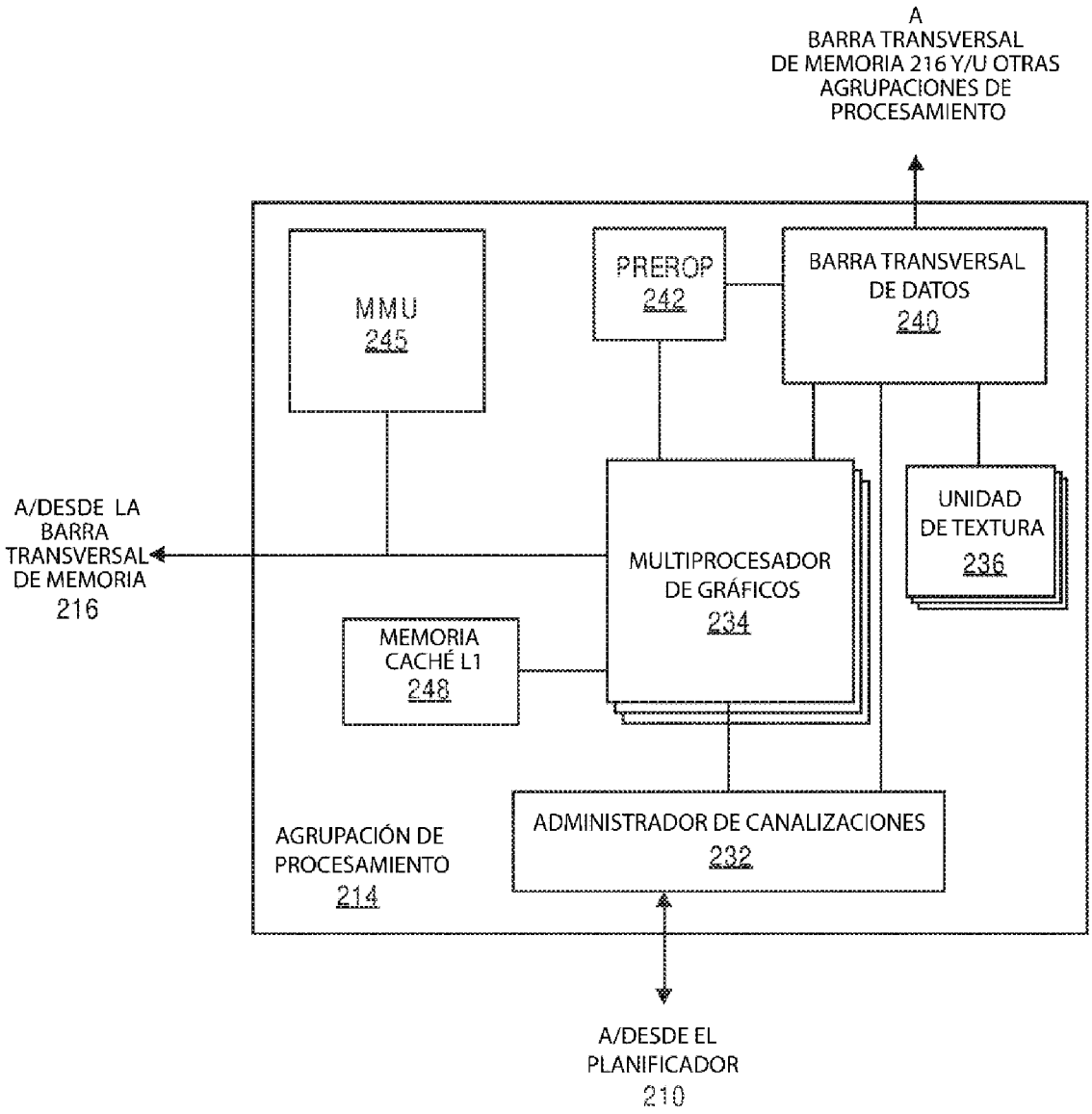

A
BARRA TRANSVERSAL
DE MEMORIA 216 Y/U OTRAS
AGRUPACIONES DE
PROCESAMIENTO
MMU
245
PREROP
242
BARRA TRANSVERSAL
DE DATOS
240
A/DESDE LA
BARRA
TRANSVERSAL
DE MEMORIA
216
MULTIPROCESADOR
DE GRÁFICOS
234
UNIDAD
DE TEXTURA
236
MEMORIA
CACHÉ L1
248
AGRUPACIÓN DE
PROCESAMIENTO
214
ADMINISTRADOR DE CANALIZACIONES
232
A/DESDE EL
PLANIFICADOR
210

FIG. 2C

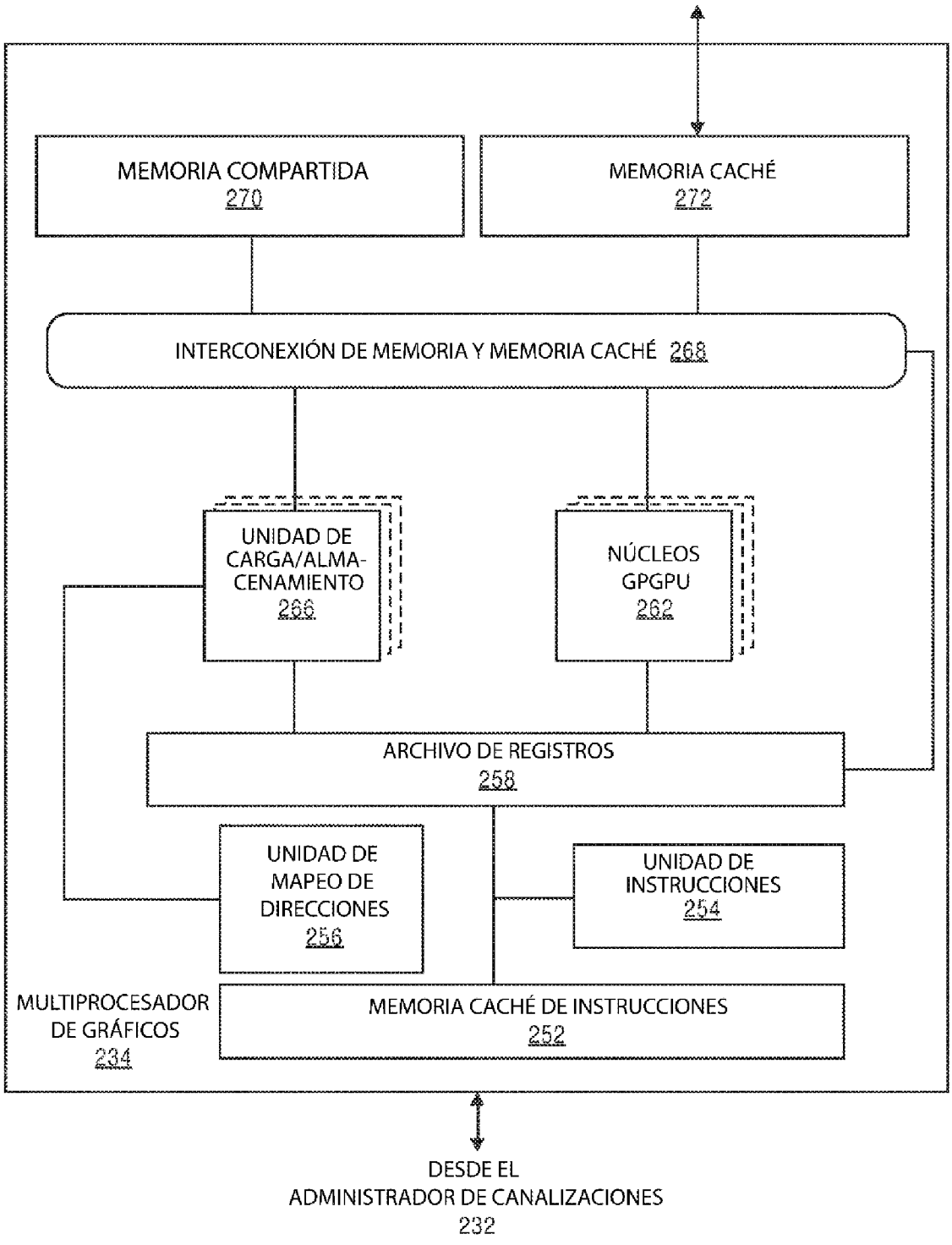

MEMORIA COMPARTIDA
270
MEMORIA CACHÉ
272
INTERCONEXIÓN DE MEMORIA Y MEMORIA CACHÉ 268
UNIDAD DE CARGA/ALMA-CENAMIENTO
266
NÚCLEOS GPGPU
262
ARCHIVO DE REGISTROS
258
UNIDAD DE MAPEO DE DIRECCIONES
256
UNIDAD DE INSTRUCCIONES
254
MULTIPROCESADOR DE GRÁFICOS
234
MEMORIA CACHÉ DE INSTRUCCIONES
252
DESDE EL ADMINISTRADOR DE CANALIZACIONES
232

FIG. 2D

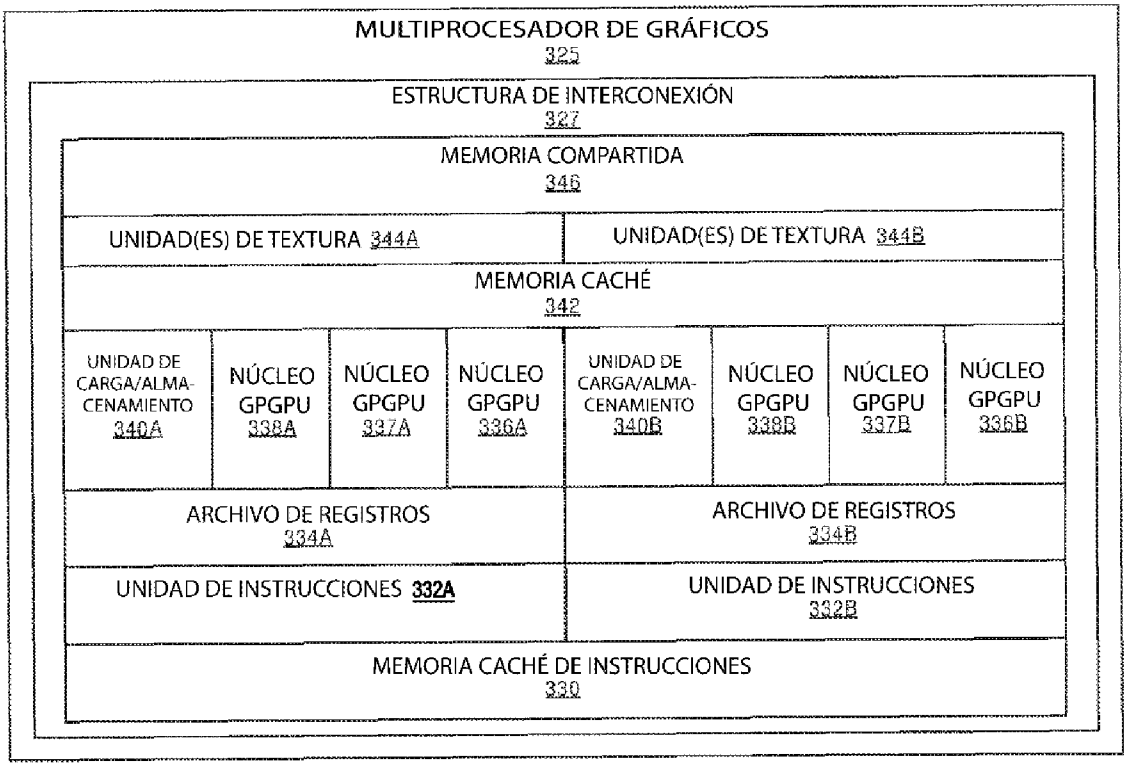

MULTIPROCESADOR DE GRÁFICOS
325
ESTRUCTURA DE INTERCONEXIÓN
327
MEMORIA COMPARTIDA
346
UNIDAD(ES) DE TEXTURA 344A
UNIDAD(ES) DE TEXTURA 344B
MEMORIA CACHÉ
342
UNIDAD DE CARGA/ALMACENAMIENTO 340A
NÚCLEO GPGPU 338A
NÚCLEO GPGPU 337A
NÚCLEO GPGPU 336A
UNIDAD DE CARGA/ALMACENAMIENTO 340B
NÚCLEO GPGPU 338B
NÚCLEO GPGPU 337B
NÚCLEO GPGPU 336B
ARCHIVO DE REGISTROS 334A
ARCHIVO DE REGISTROS 334B
UNIDAD DE INSTRUCCIONES 332A
UNIDAD DE INSTRUCCIONES 332B
MEMORIA CACHÉ DE INSTRUCCIONES
330

FIG. 3A

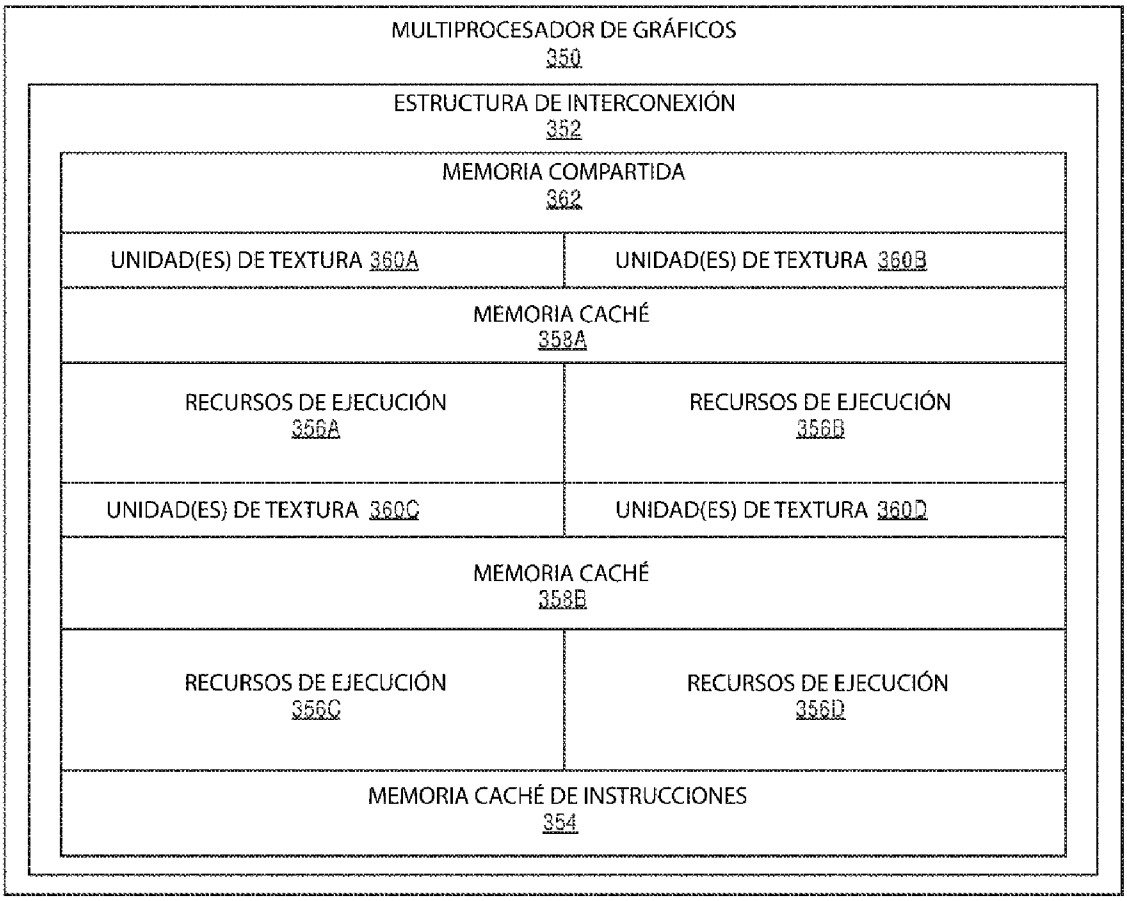

MULTIPROCESADOR DE GRÁFICOS
350
ESTRUCTURA DE INTERCONEXIÓN
352
MEMORIA COMPARTIDA
362
UNIDAD(ES) DE TEXTURA 360A
UNIDAD(ES) DE TEXTURA 360B
MEMORIA CACHÉ
358A
RECURSOS DE EJECUCIÓN
356A
RECURSOS DE EJECUCIÓN
356B
UNIDAD(ES) DE TEXTURA 360C
UNIDAD(ES) DE TEXTURA 360D
MEMORIA CACHÉ
358B
RECURSOS DE EJECUCIÓN
356C
RECURSOS DE EJECUCIÓN
356D
MEMORIA CACHÉ DE INSTRUCCIONES
354

FIG. 3B

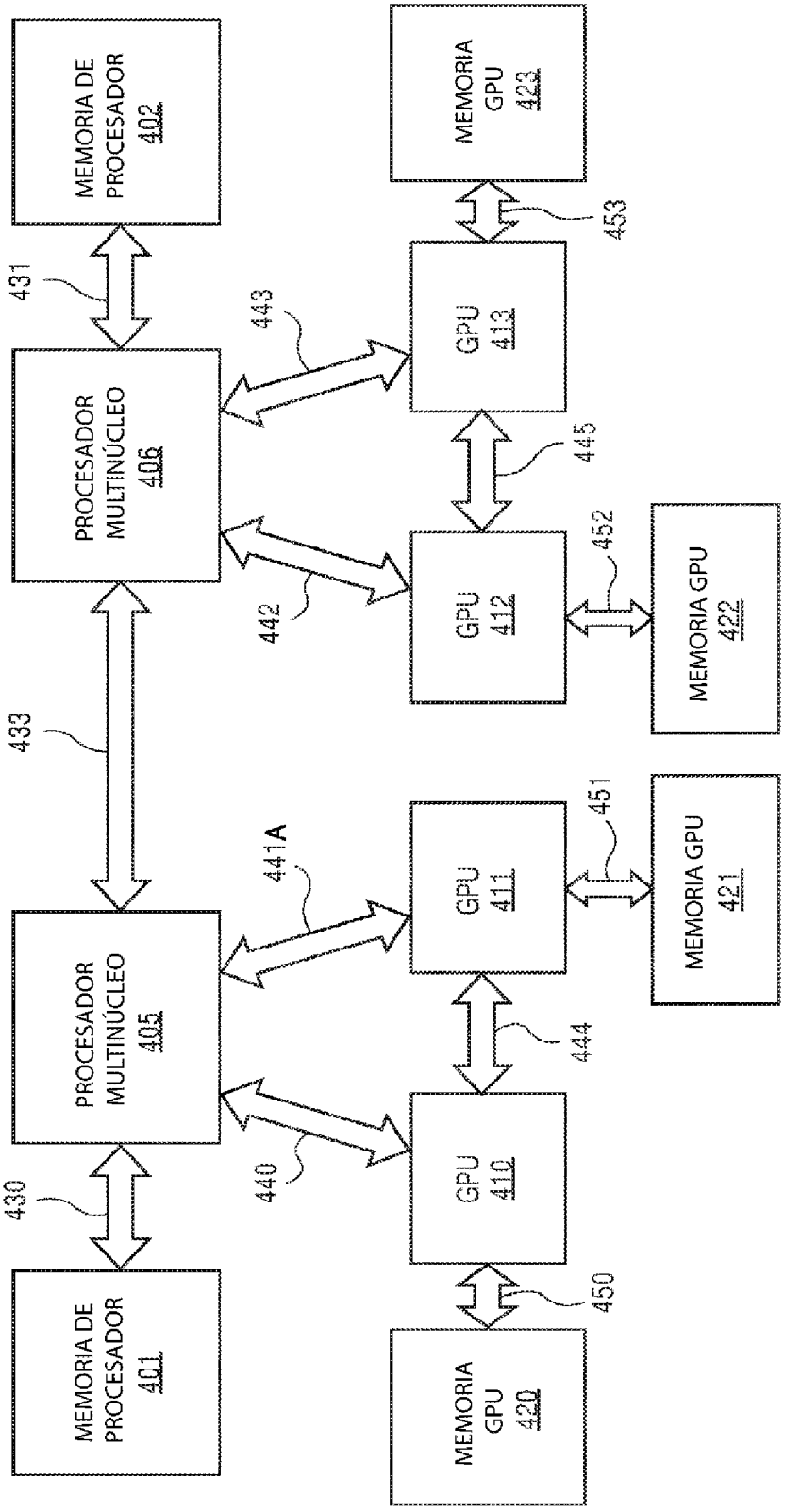

MEMORIA DE PROCESADOR 402
MEMORIA GPU 423
431
453
443
GPU 413
PROCESADOR MULTINÚCLEO 406
445
452
442
GPU 412
MEMORIA GPU 422
433
441A
451
GPU 411
MEMORIA GPU 421
PROCESADOR MULTINÚCLEO 405
444
440
GPU 410
430
450
MEMORIA DE PROCESADOR 401
MEMORIA GPU 420

FIG. 4A

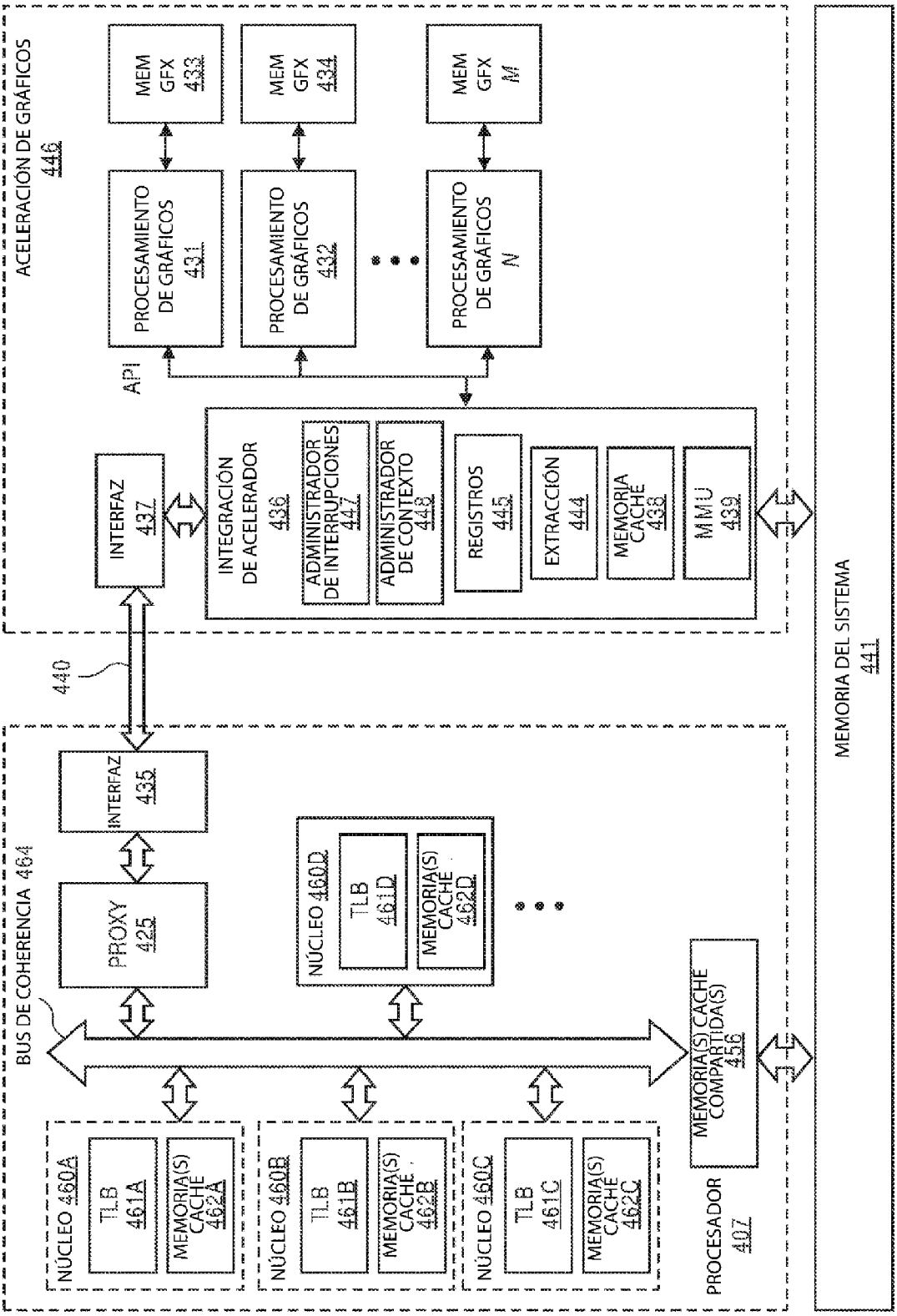

ACELERACIÓN DE GRÁFICOS 446
MEM GFX 433
MEM GFX 434
MEM GFX M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTERFAZ 437
INTEGRACIÓN DE ACELERADOR 436
ADMINISTRADOR DE INTERRUPCIONES 447
ADMINISTRADOR DE CONTEXTO 448
REGISTROS 445
EXTRACCIÓN 444
MEMORIA CACHÉ 438
MMU 439
440
INTERFAZ 435
PROXY 425
BUS DE COHERENCIA 464
NÚCLEO 460D
TLB 461D
MEMORIA(S) CACHÉ 462D
NÚCLEO 460A
TLB 461A
MEMORIA(S) CACHÉ 462A
NÚCLEO 460B
TLB 461B
MEMORIA(S) CACHÉ 462B
NÚCLEO 460C
TLB 461C
MEMORIA(S) CACHÉ 462C
MEMORIA(S) CACHÉ COMPARTIDA(S) 456
PROCESADOR 407
MEMORIA DEL SISTEMA 441

FIG. 4B

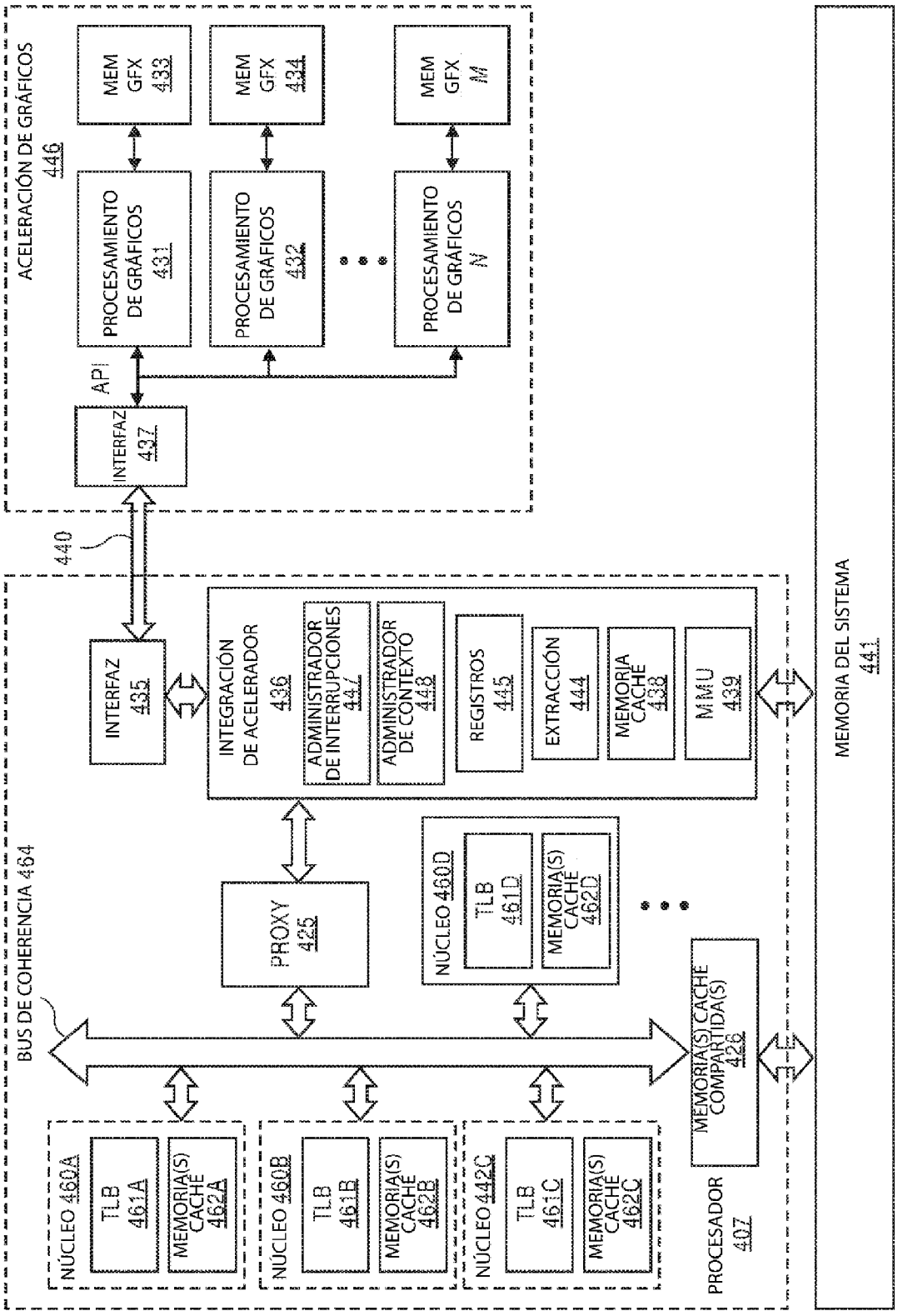

ACELERACIÓN DE GRÁFICOS 446
MEM GFX 433
MEM GFX 434
MEM GFX M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTERFAZ 437
440
BUS DE COHERENCIA 464
INTERFAZ 435
INTEGRACIÓN DE ACELERADOR 436
ADMINISTRADOR DE INTERRUPCIONES 447
ADMINISTRADOR DE CONTEXTO 448
REGISTROS 445
EXTRACCIÓN 444
MEMORIA CACHE 438
MMU 439
PROXY 425
NÚCLEO 460D
TLB 461D
MEMORIA(S) CACHE 462D
NÚCLEO 460A
TLB 461A
MEMORIA(S) CACHE 462A
NÚCLEO 460B
TLB 461B
MEMORIA(S) CACHE 462B
NÚCLEO 442C
TLB 461C
MEMORIA(S) CACHE 462C
MEMORIA(S) CACHE COMPARTIDA(S) 426
PROCESADOR 407
MEMORIA DEL SISTEMA 441

FIG. 4C

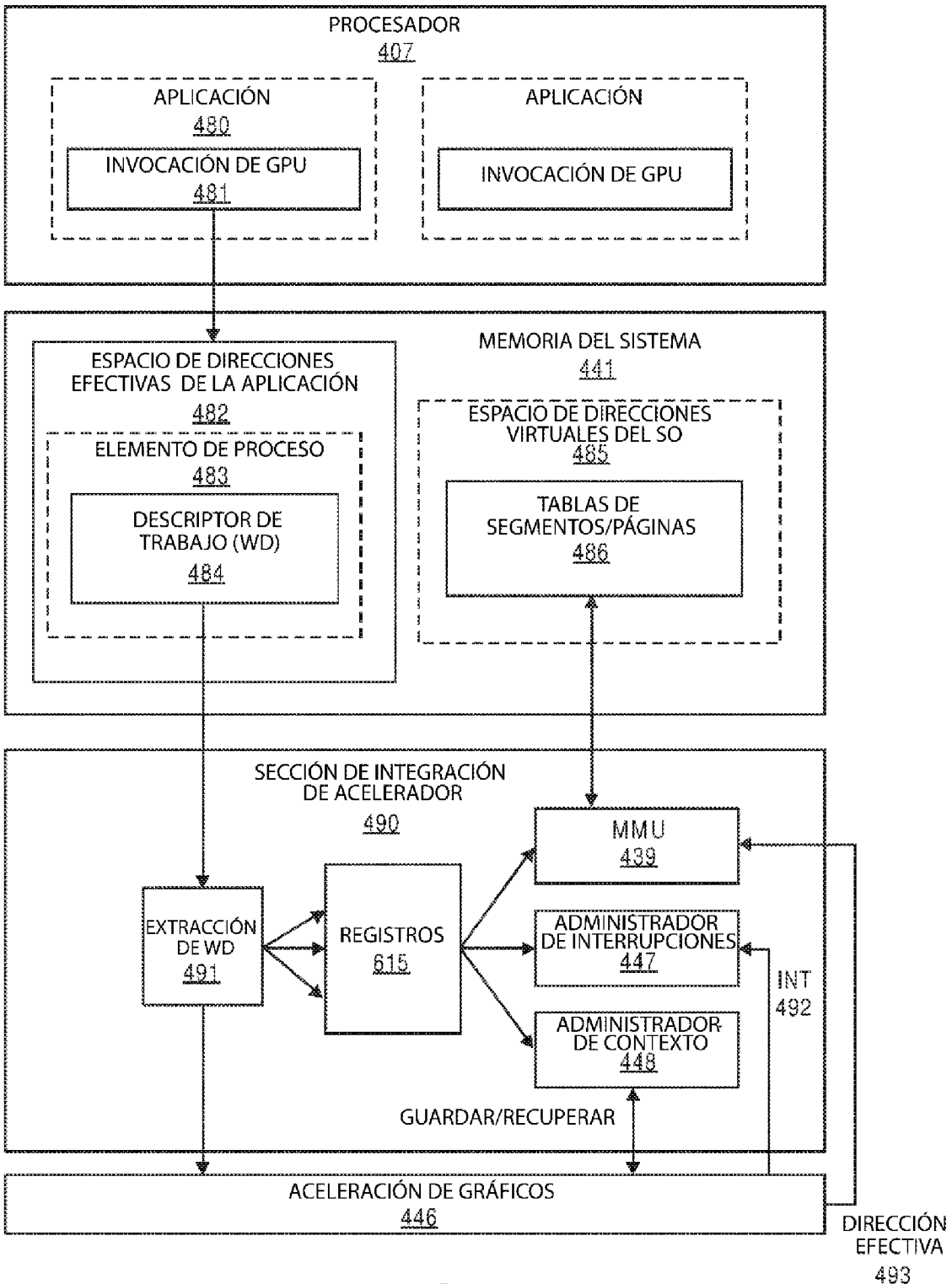

PROCESADOR
407
APLICACIÓN
480
INVOCACIÓN DE GPU
481
APLICACIÓN
INVOCACIÓN DE GPU
ESPACIO DE DIRECCIONES EFECTIVAS DE LA APLICACIÓN
482
ELEMENTO DE PROCESO
483
DESCRIPTOR DE TRABAJO (WD)
484
MEMORIA DEL SISTEMA
441
ESPACIO DE DIRECCIONES VIRTUALES DEL SO
485
TABLAS DE SEGMENTOS/PÁGINAS
486
SECCIÓN DE INTEGRACIÓN DE ACELERADOR
490
MMU
439
EXTRACCIÓN DE WD
491
REGISTROS
615
ADMINISTRADOR DE INTERRUPCIONES
447
INT
492
ADMINISTRADOR DE CONTEXTO
448
GUARDAR/RECUPERAR
ACELERACIÓN DE GRÁFICOS
446
DIRECCIÓN EFECTIVA
493

FIG. 4D

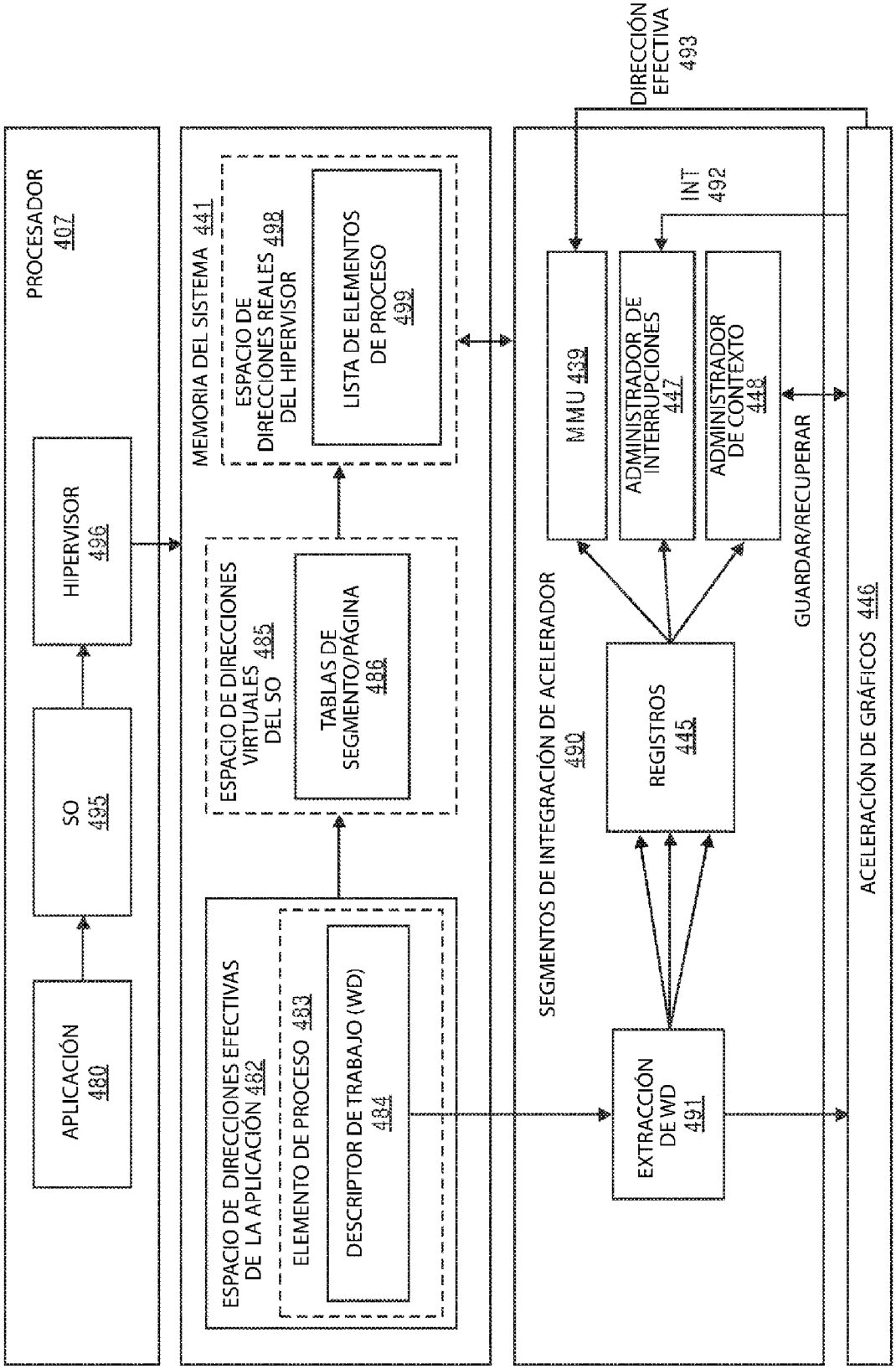

PROCESADOR 407
APLICACIÓN 480
SO 495
HIPERVISOR 496
MEMORIA DEL SISTEMA 441
ESPACIO DE DIRECCIONES EFECTIVAS DE LA APLICACIÓN 482
ELEMENTO DE PROCESO 483
DESCRIPTOR DE TRABAJO (WD) 484
ESPACIO DE DIRECCIONES VIRTUALES DEL SO 485
TABLAS DE SEGMENTO/PÁGINA 486
ESPACIO DE DIRECCIONES REALES DEL HIPERVISOR 498
LISTA DE ELEMENTOS DE PROCESO 499
SEGMENTOS DE INTEGRACIÓN DE ACELERADOR 490
EXTRACCIÓN DE WD 491
REGISTROS 445
MMU 439
ADMINISTRADOR DE INTERRUPCIONES 447
ADMINISTRADOR DE CONTEXTO 448
GUARDAR/RECUPERAR
DIRECCIÓN EFECTIVA 493
INT 492
ACELERACIÓN DE GRÁFICOS 446

FIG. 4E

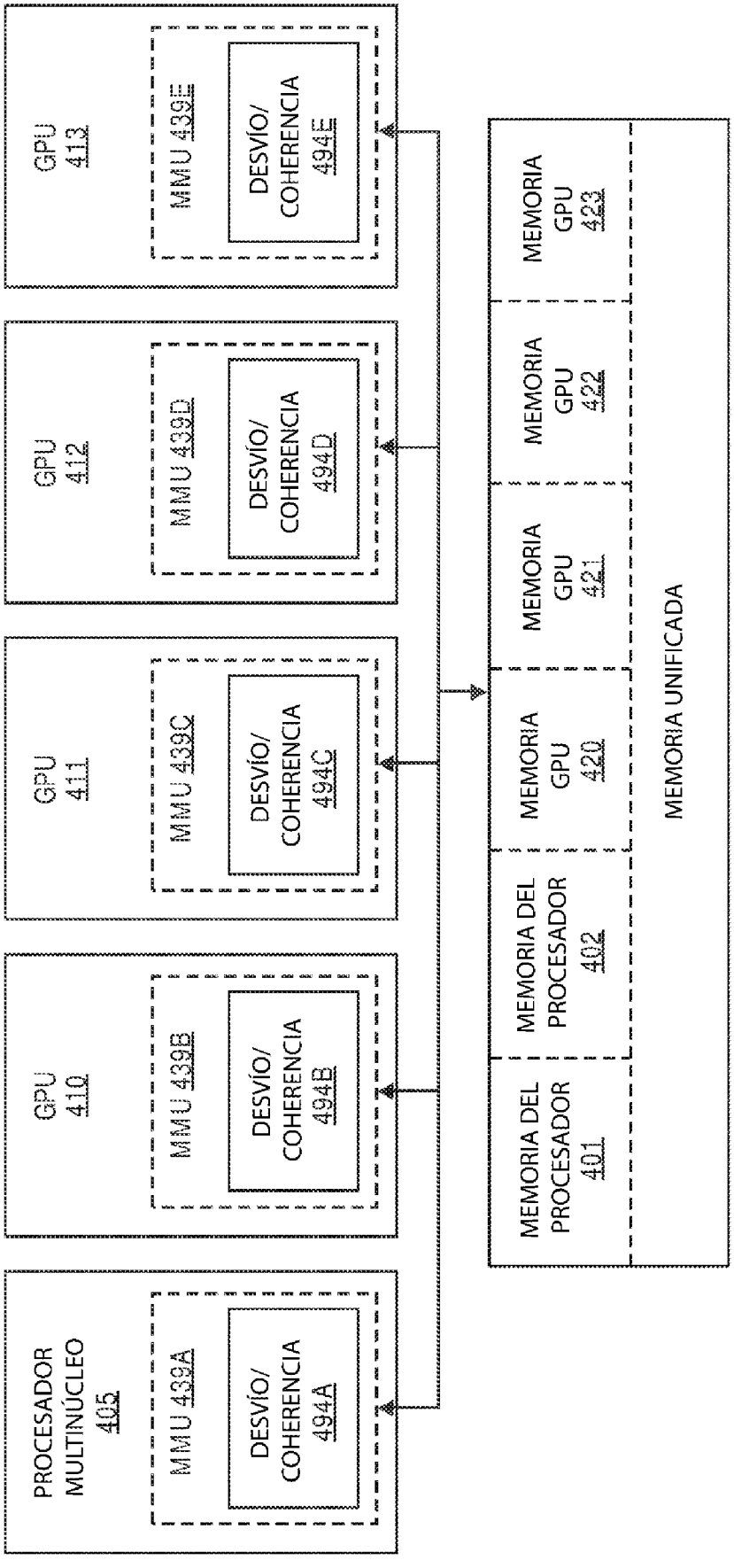

PROCESADOR MULTINÚCLEO 405
MMU 439A
DESVÍO/ COHERENCIA 494A
GPU 410
MMU 439B
DESVÍO/ COHERENCIA 494B
GPU 411
MMU 439C
DESVÍO/ COHERENCIA 494C
GPU 412
MMU 439D
DESVÍO/ COHERENCIA 494D
GPU 413
MMU 439E
DESVÍO/ COHERENCIA 494E
MEMORIA DEL PROCESADOR 401
MEMORIA DEL PROCESADOR 402
MEMORIA GPU 420
MEMORIA GPU 421
MEMORIA GPU 422
MEMORIA GPU 423
MEMORIA UNIFICADA

FIG. 4F

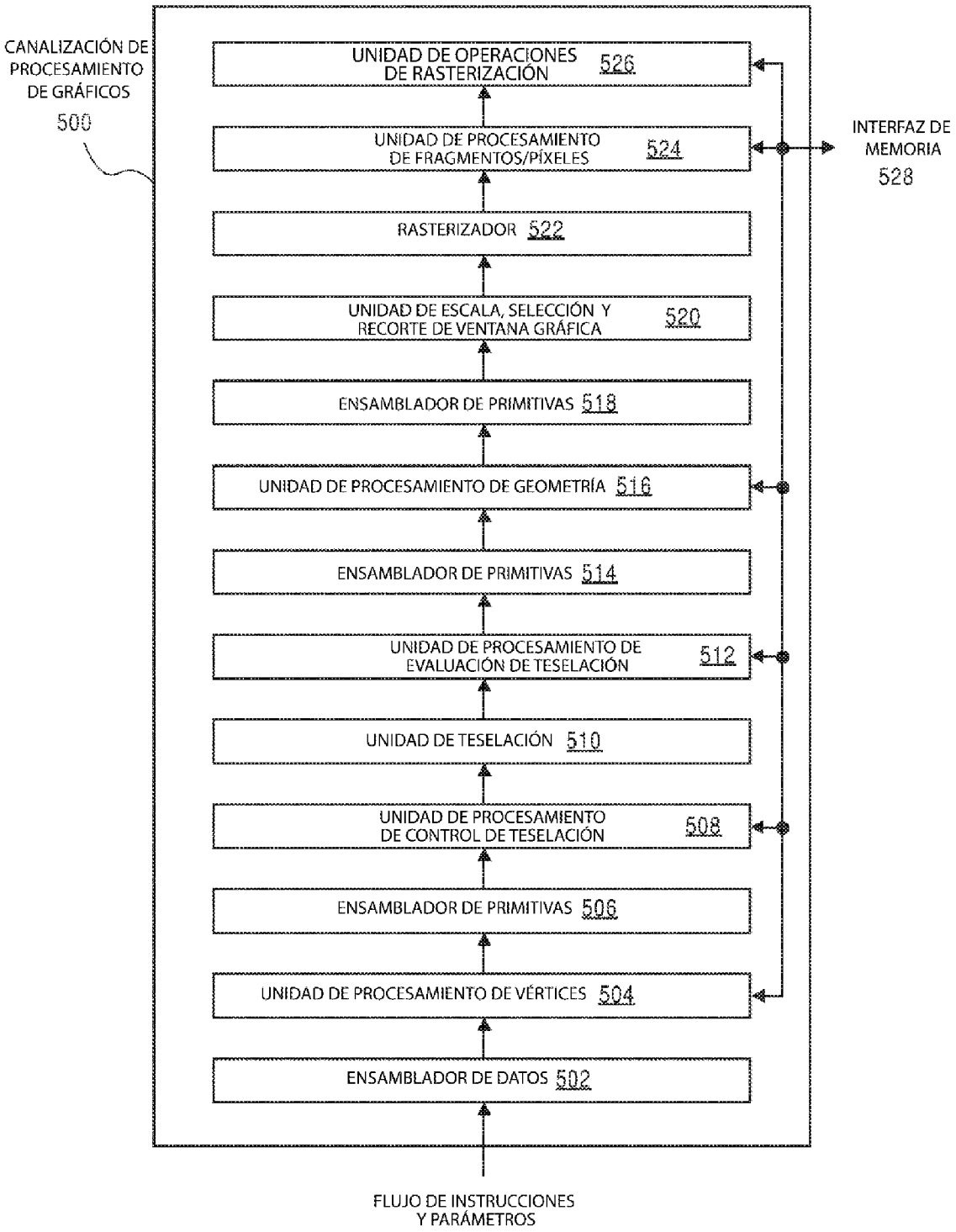

CANALIZACIÓN DE PROCESAMIENTO DE GRÁFICOS
500
UNIDAD DE OPERACIONES DE RASTERIZACIÓN 526
UNIDAD DE PROCESAMIENTO DE FRAGMENTOS/PÍXELES 524
RASTERIZADOR 522
UNIDAD DE ESCALA, SELECCIÓN Y RECORTE DE VENTANA GRÁFICA 520
ENSAMBLADOR DE PRIMITIVAS 518
UNIDAD DE PROCESAMIENTO DE GEOMETRÍA 516
ENSAMBLADOR DE PRIMITIVAS 514
UNIDAD DE PROCESAMIENTO DE EVALUACIÓN DE TESELACIÓN 512
UNIDAD DE TESELACIÓN 510
UNIDAD DE PROCESAMIENTO DE CONTROL DE TESELACIÓN 508
ENSAMBLADOR DE PRIMITIVAS 506
UNIDAD DE PROCESAMIENTO DE VÉRTICES 504
ENSAMBLADOR DE DATOS 502
INTERFAZ DE MEMORIA
528
FLUJO DE INSTRUCCIONES Y PARÁMETROS

FIG. 5

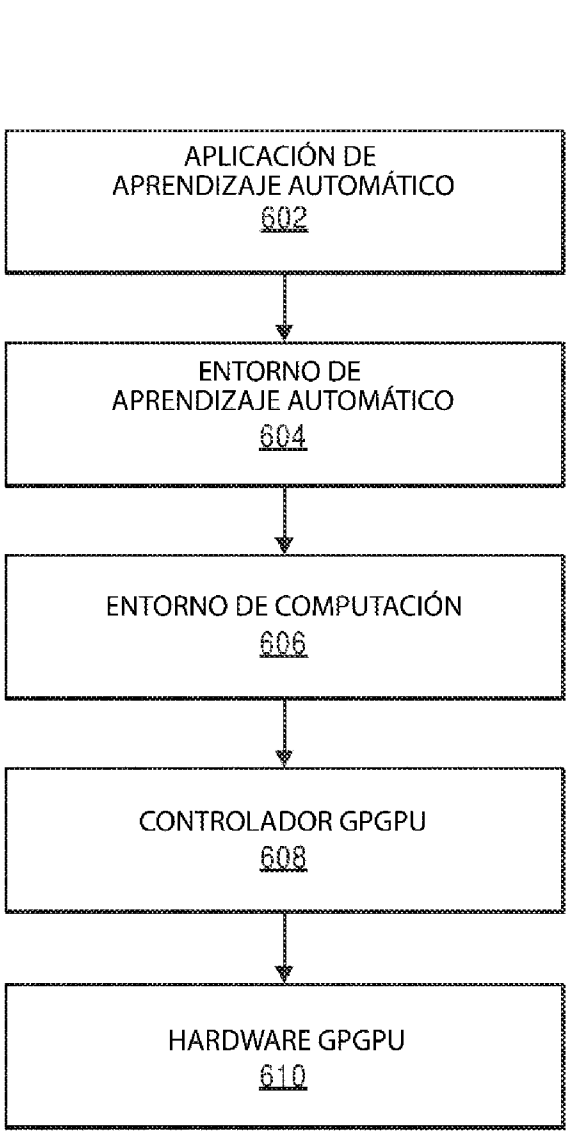

600
APLICACIÓN DE APRENDIZAJE AUTOMÁTICO
602
ENTORNO DE APRENDIZAJE AUTOMÁTICO
604
ENTORNO DE COMPUTACIÓN
606
CONTROLADOR GPGPU
608
HARDWARE GPGPU
610

FIG. 6

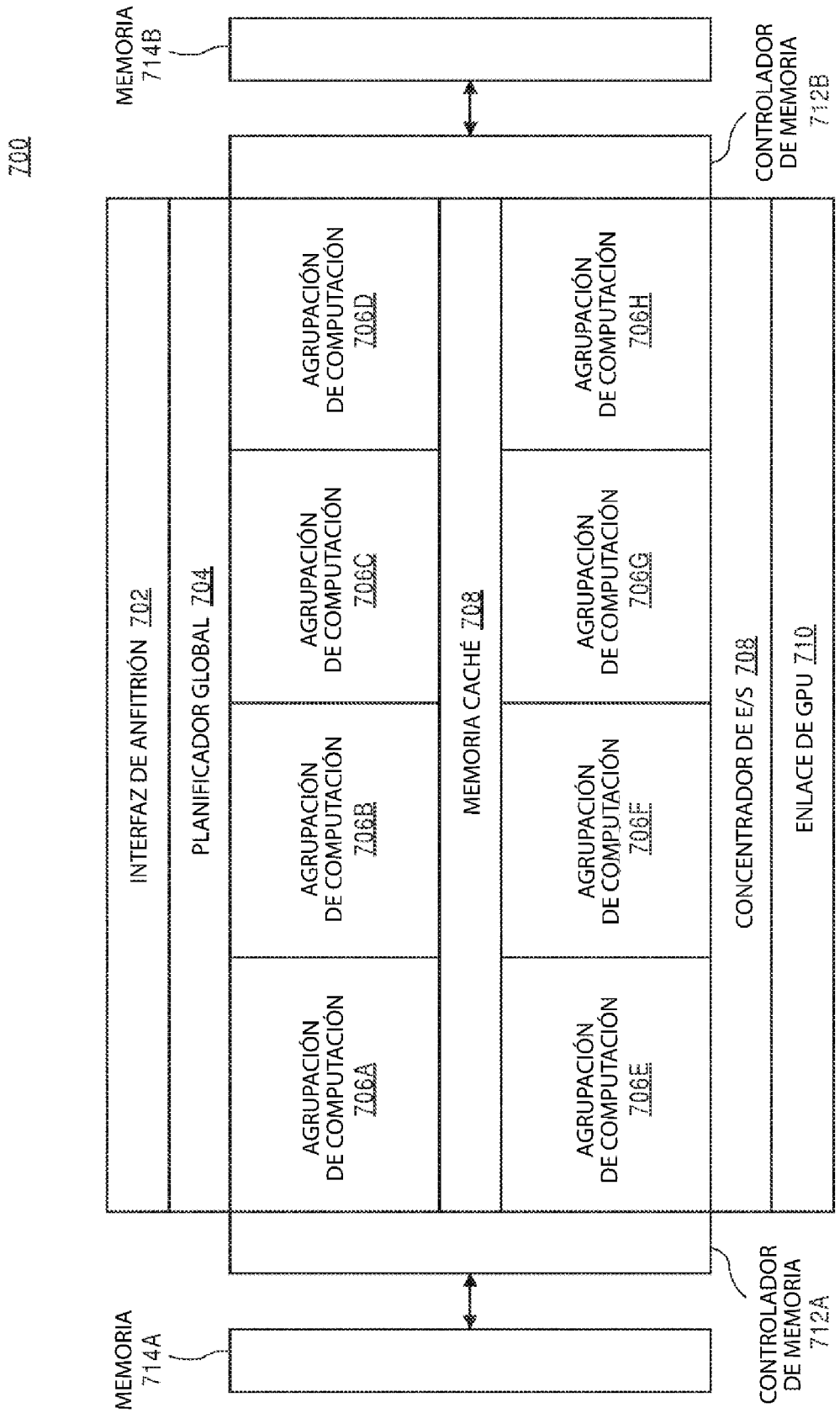

700
MEMORIA 714B
CONTROLADOR DE MEMORIA 712B
INTERFAZ DE ANFITRIÓN 702
PLANIFICADOR GLOBAL 704
AGRUPACIÓN DE COMPUTACIÓN 706A
AGRUPACIÓN DE COMPUTACIÓN 706B
AGRUPACIÓN DE COMPUTACIÓN 706C
AGRUPACIÓN DE COMPUTACIÓN 706D
MEMORIA CACHÉ 708
AGRUPACIÓN DE COMPUTACIÓN 706E
AGRUPACIÓN DE COMPUTACIÓN 706F
AGRUPACIÓN DE COMPUTACIÓN 706G
AGRUPACIÓN DE COMPUTACIÓN 706H
CONCENTRADOR DE E/S 708
ENLACE DE GPU 710
MEMORIA 714A
CONTROLADOR DE MEMORIA 712A

FIG. 7

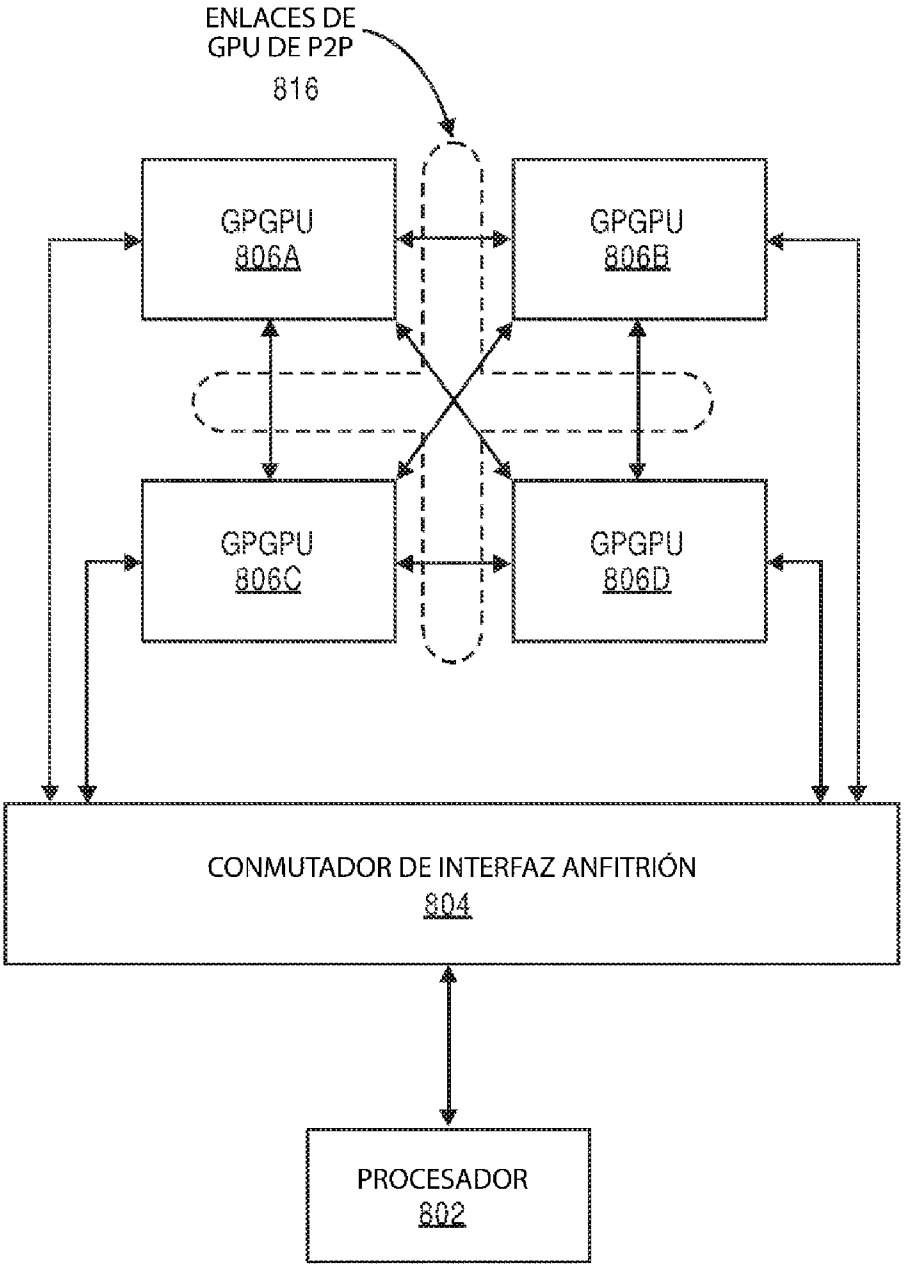

ENLACES DE
GPU DE P2P
816
GPGPU
806A
GPGPU
806B
GPGPU
806C
GPGPU
806D
CONMUTADOR DE INTERFAZ ANFITRIÓN
804
PROCESADOR
802

FIG. 8

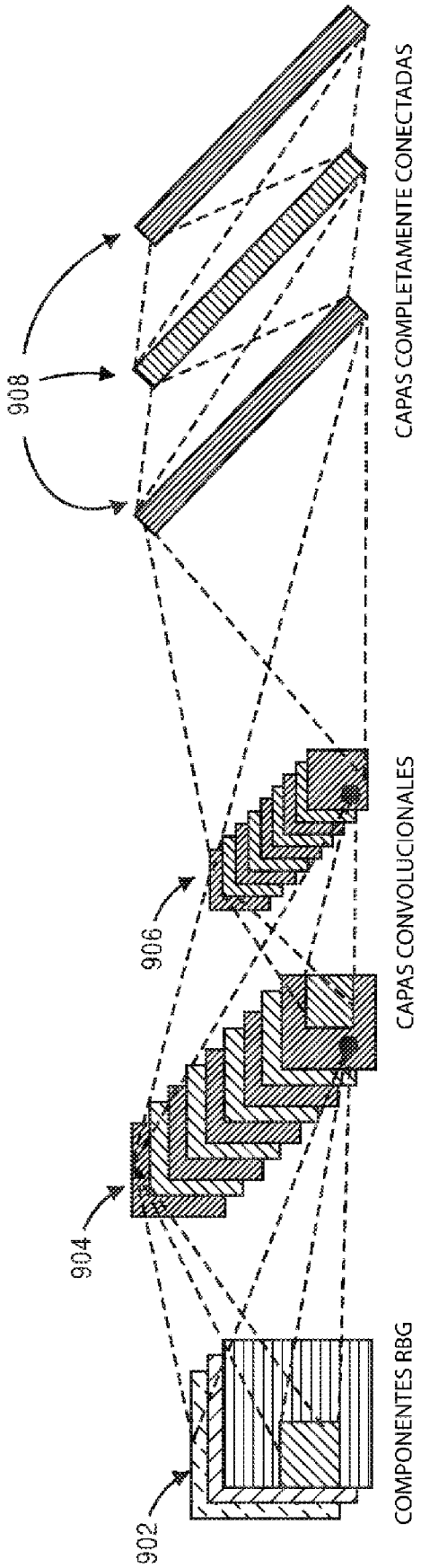

908
906
904
902
CAPAS COMPLETAMENTE CONECTADAS
CAPAS CONVOLUCIONALES
COMPONENTES RBG

FIG. 9A

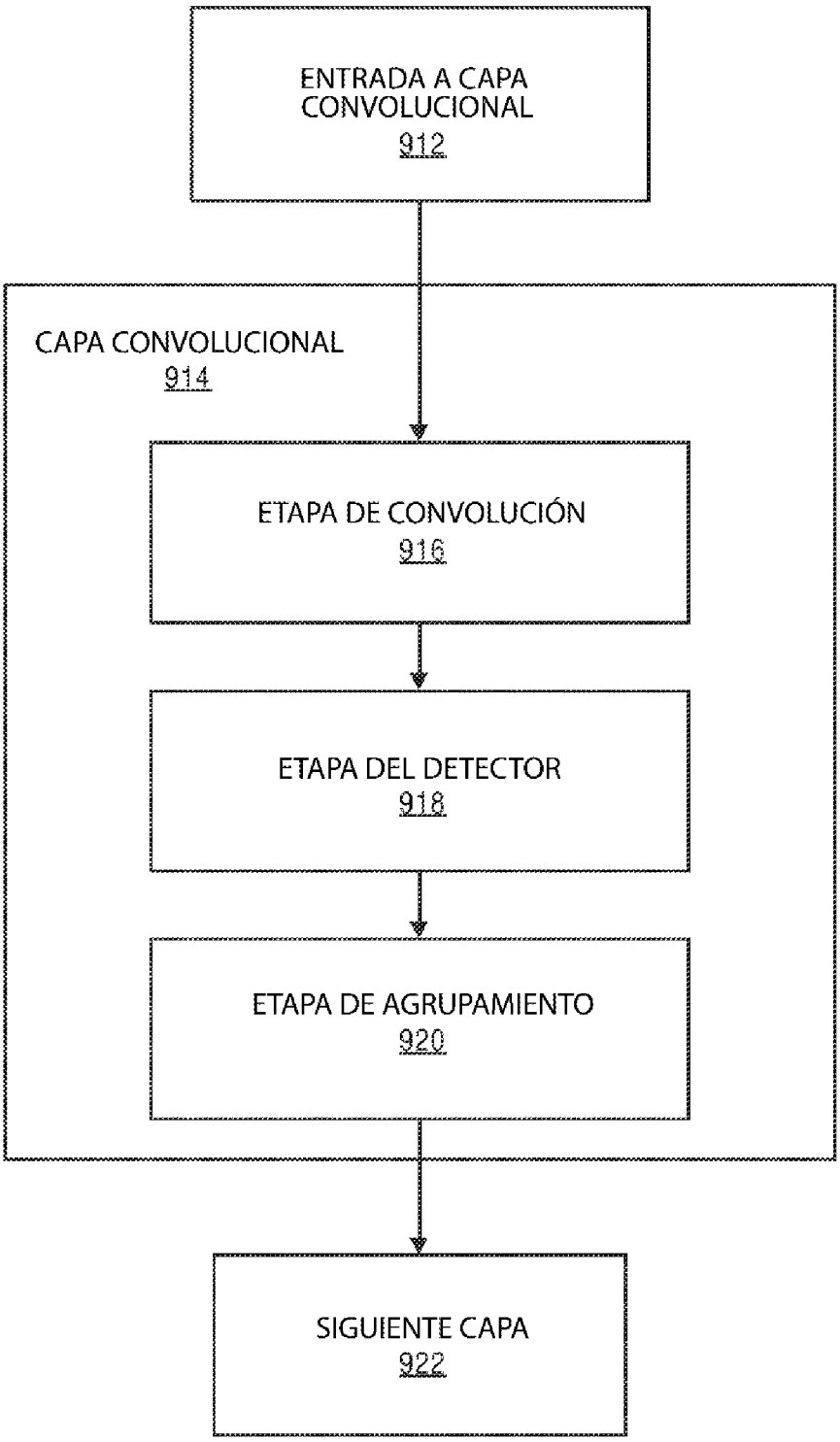

ENTRADA A CAPA CONVOLUCIONAL
912
CAPA CONVOLUCIONAL
914
ETAPA DE CONVOLUCIÓN
916
ETAPA DEL DETECTOR
918
ETAPA DE AGRUPAMIENTO
920
SIGUIENTE CAPA
922

FIG. 9B

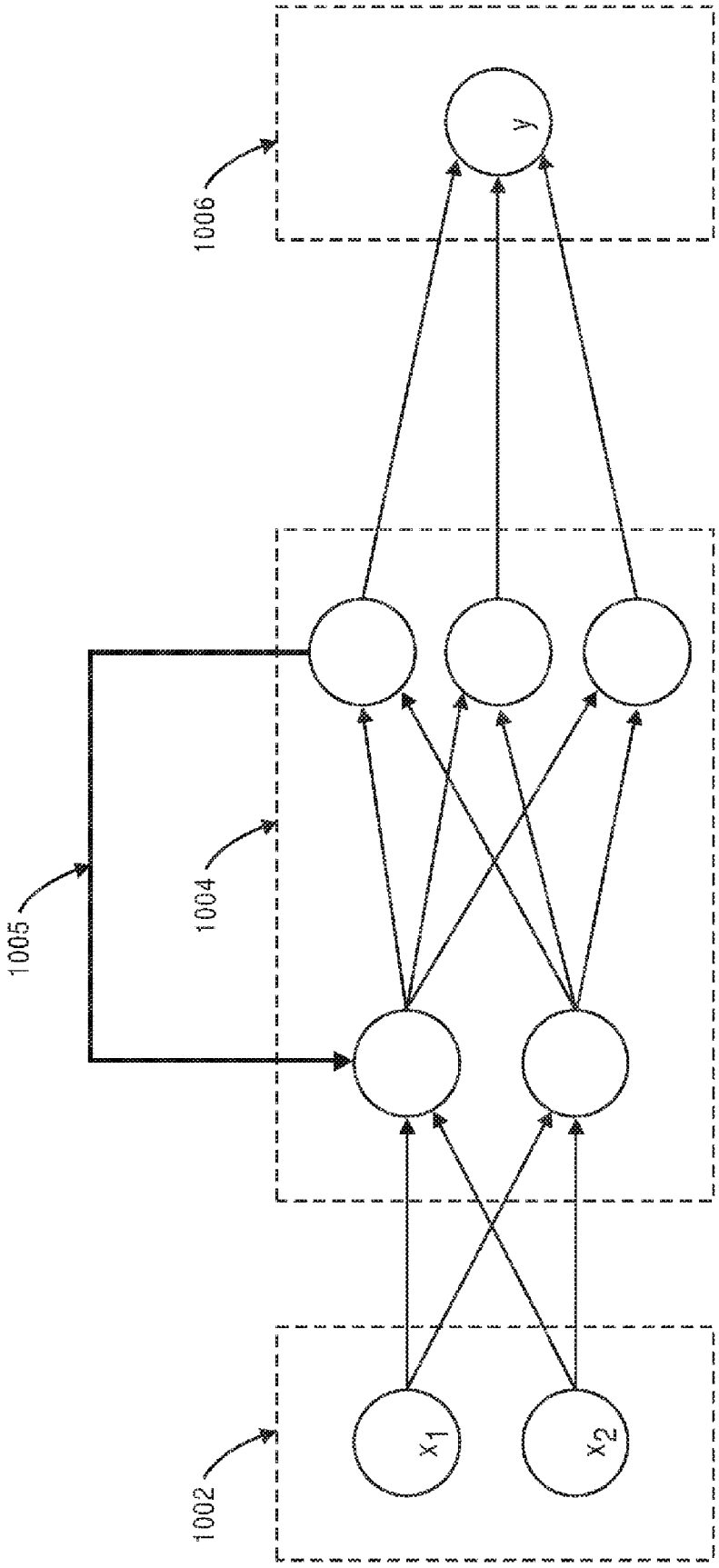


FIG. 10

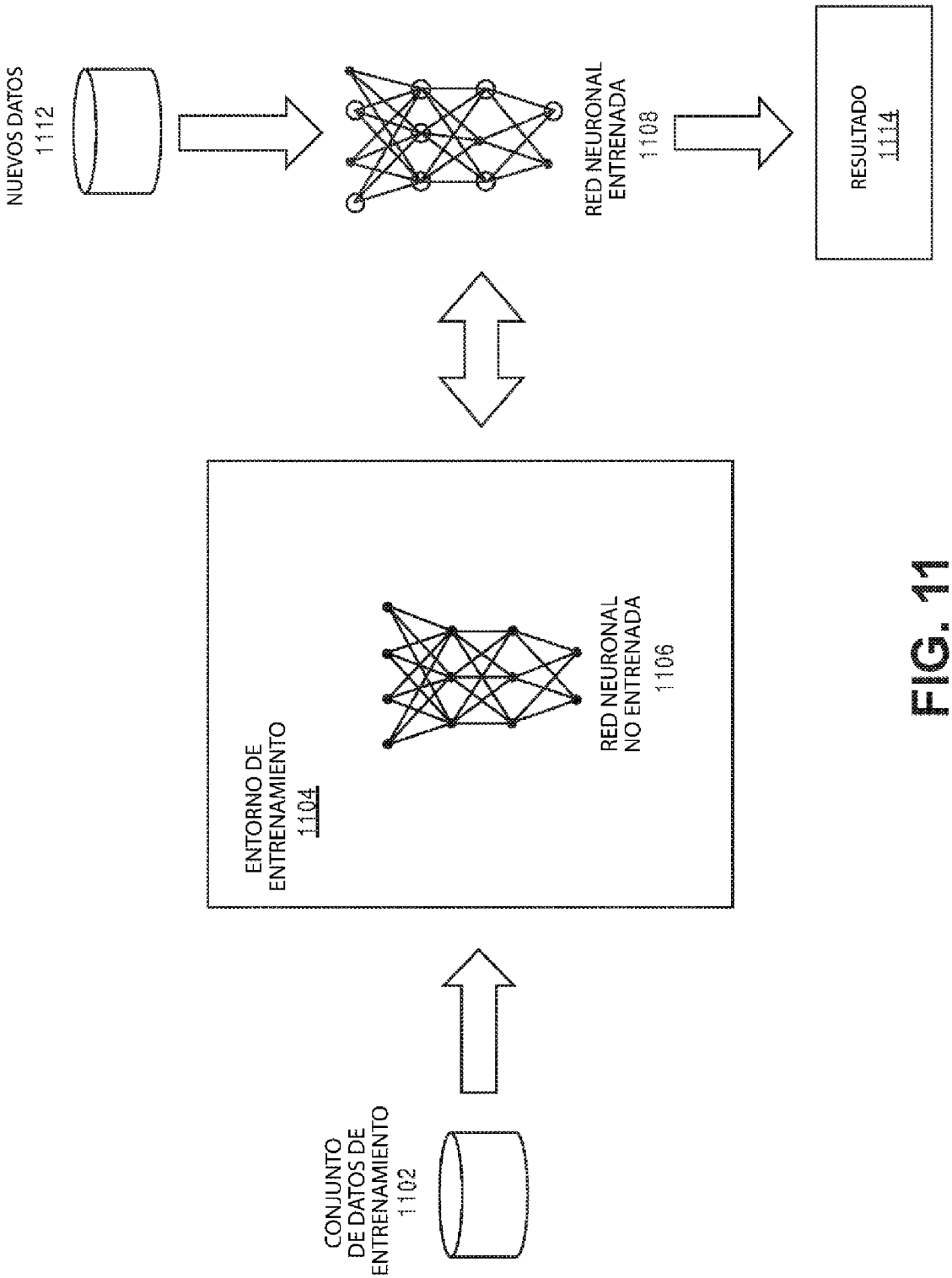

NUEVOS DATOS
1112
RED NEURONAL ENTRENADA
1108
RESULTADO
1114
ENTORNO DE ENTRENAMIENTO
1104
RED NEURONAL NO ENTRENADA
1106
CONJUNTO DE DATOS DE ENTRENAMIENTO
1102

FIG. 11

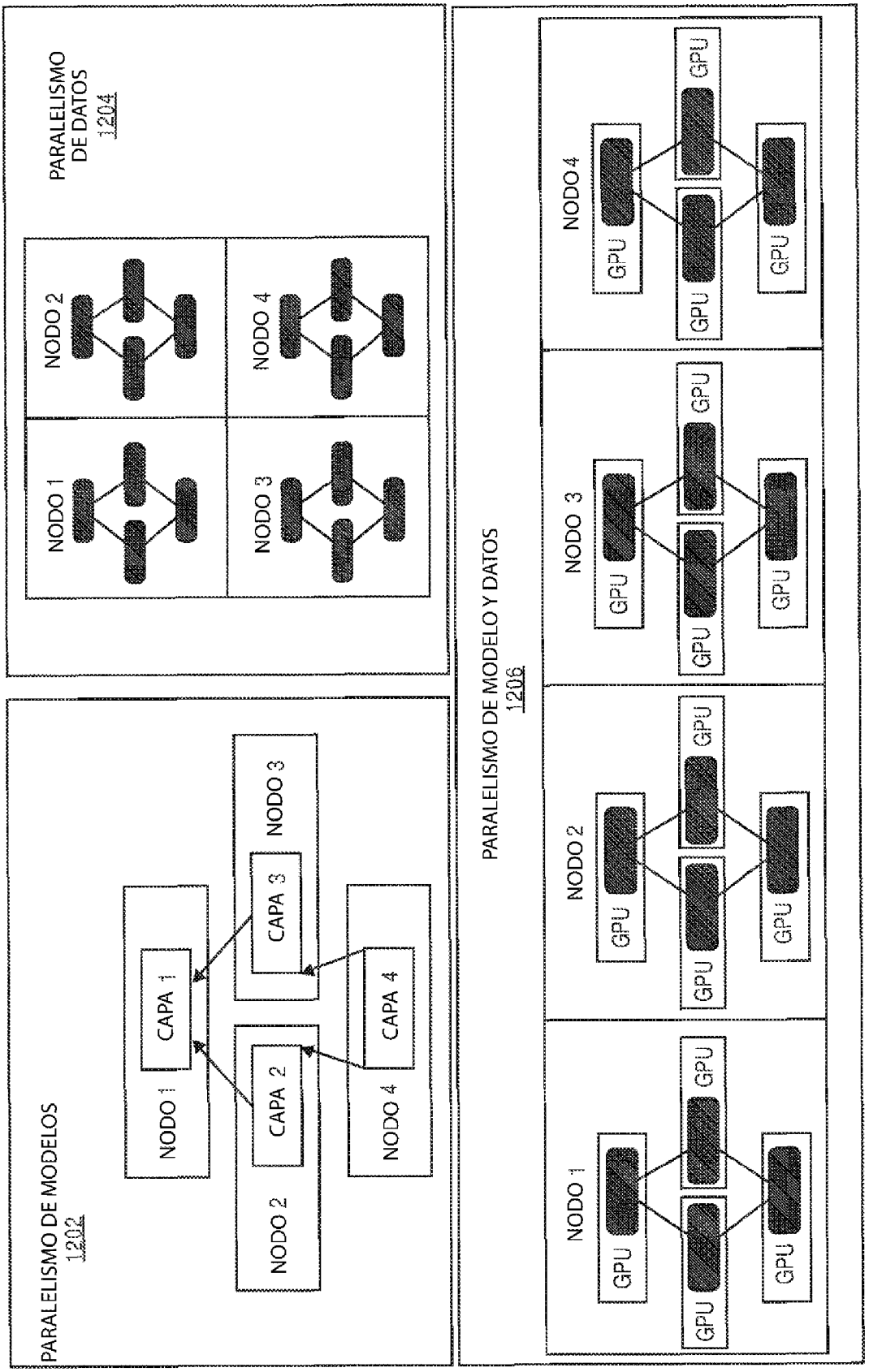

PARALELISMO DE MODELOS
1202
NODO 1
CAPA 1
NODO 2
CAPA 2
NODO 3
CAPA 3
NODO 4
CAPA 4
PARALELISMO DE DATOS
1204
NODO 1
NODO 2
NODO 3
NODO 4
PARALELISMO DE MODELO Y DATOS
1206
NODO 1
NODO 2
NODO 3
NODO 4
GPU

FIG. 12

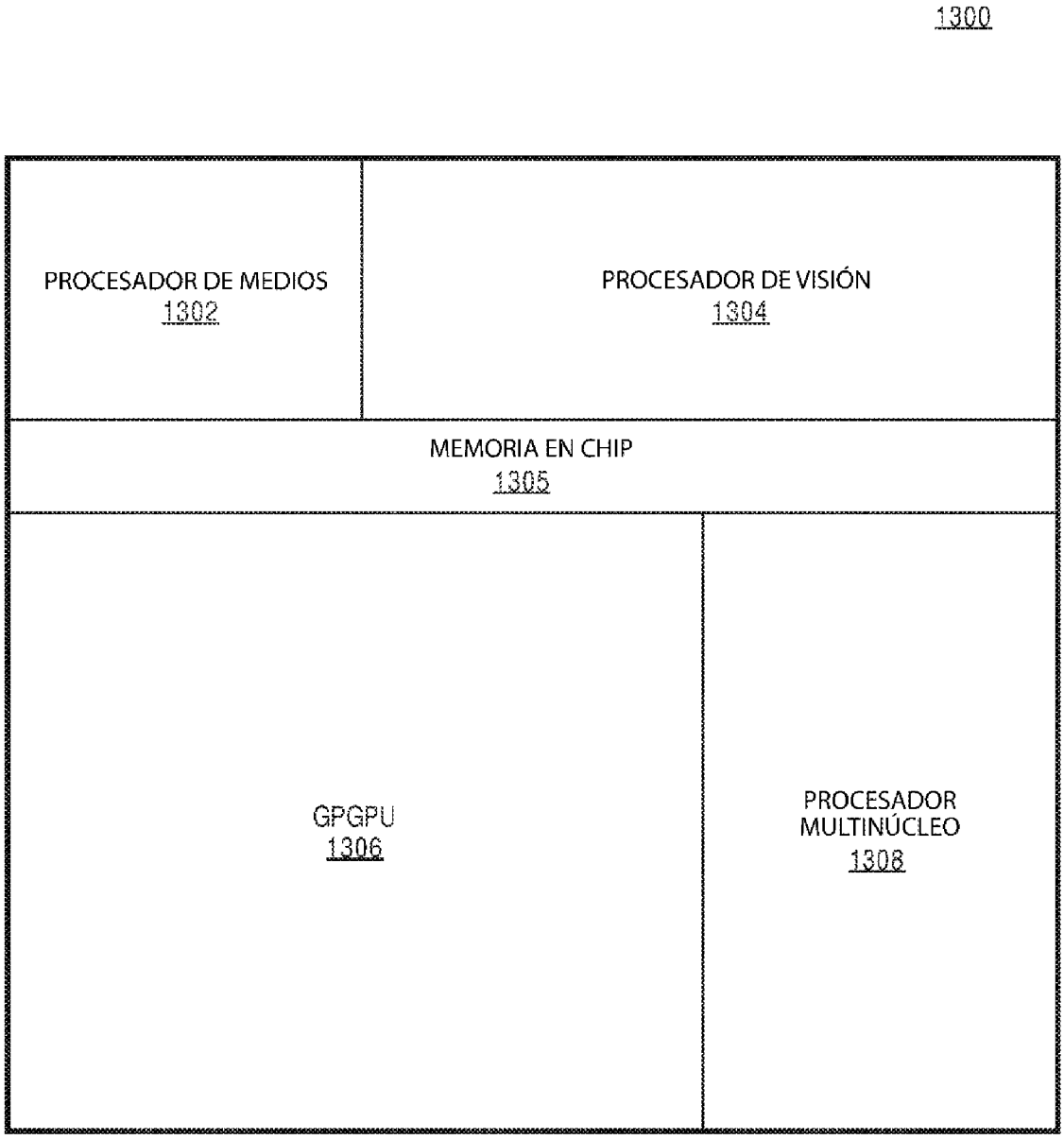

1300
PROCESADOR DE MEDIOS
1302
PROCESADOR DE VISIÓN
1304
MEMORIA EN CHIP
1305
GPGPU
1306
PROCESADOR
MULTINÚCLEO
1308

FIG. 13

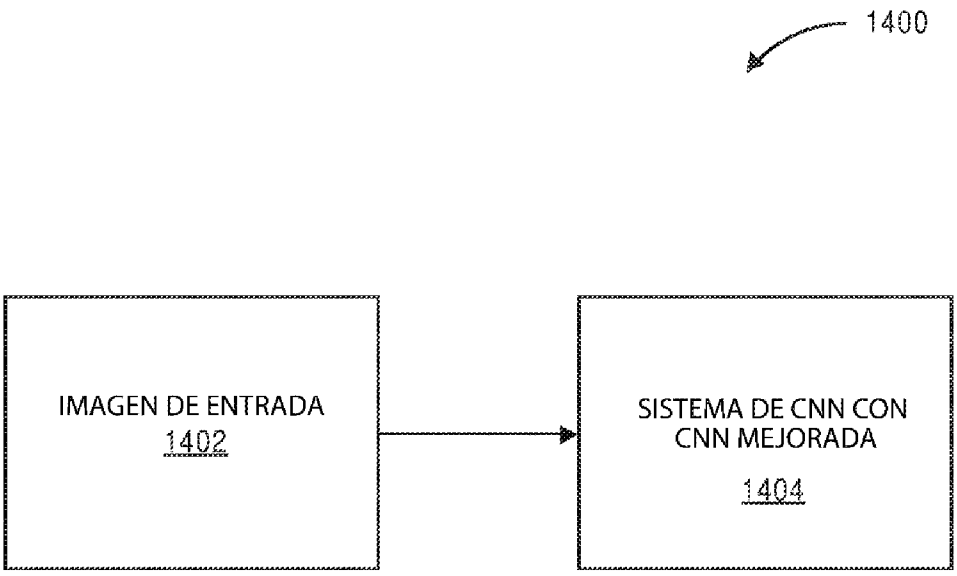

1400
IMAGEN DE ENTRADA
1402
SISTEMA DE CNN CON CNN MEJORADA
1404

FIG. 14

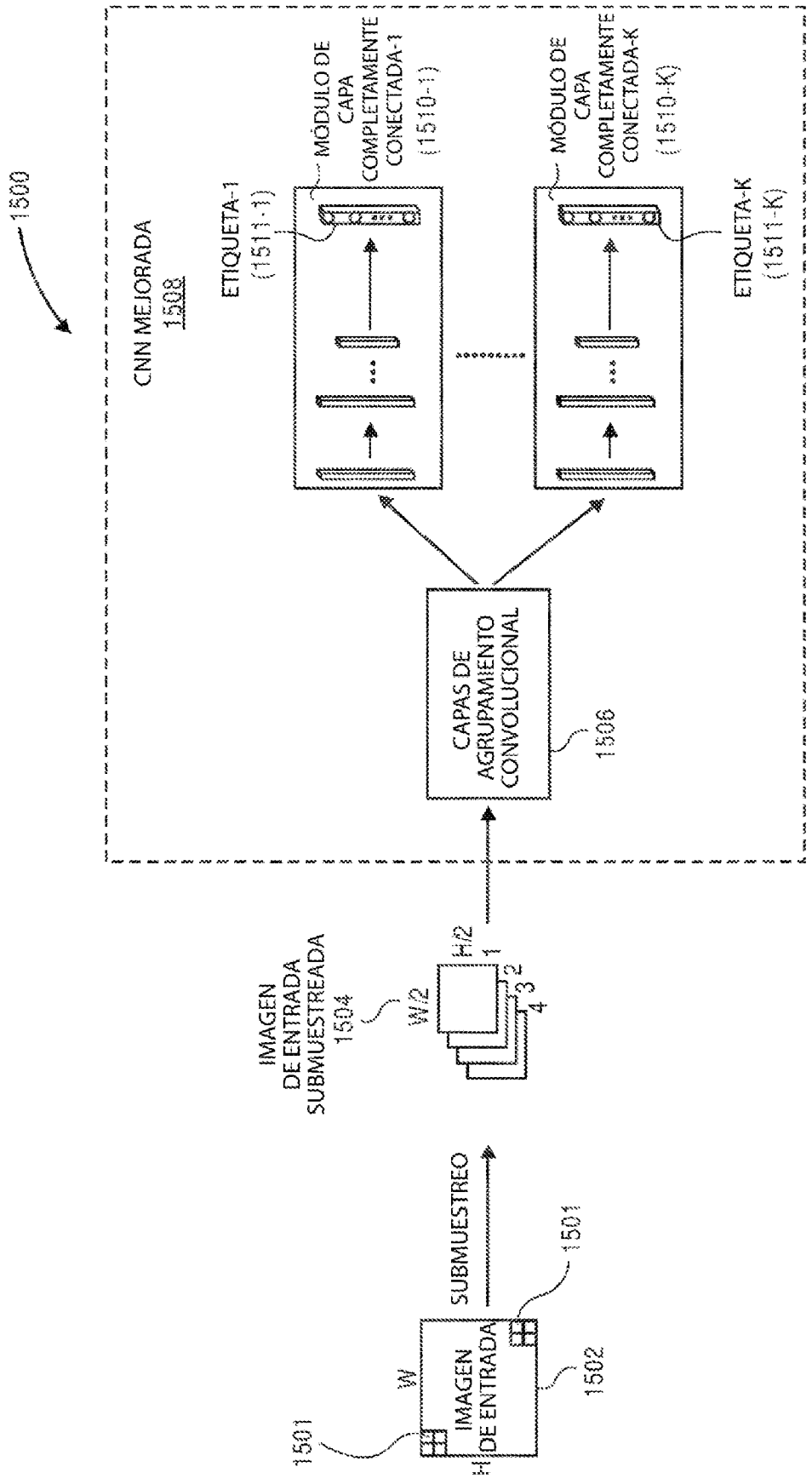

1500
CNN MEJORADA
1508
ETIQUETA-1
(1511-1)
MÓDULO DE CAPA COMPLETAMENTE CONECTADA-1
(1510-1)
MÓDULO DE CAPA COMPLETAMENTE CONECTADA-K
(1510-K)
ETIQUETA-K
(1511-K)
CAPAS DE AGRUPAMIENTO CONVOLUCIONAL
1506
IMAGEN DE ENTRADA SUBMUESTREADA
1504
W/2
H/2
1
2
3
4
SUBMUESTREO
1501
IMAGEN DE ENTRADA
W
H
1502
1501

FIG. 15A

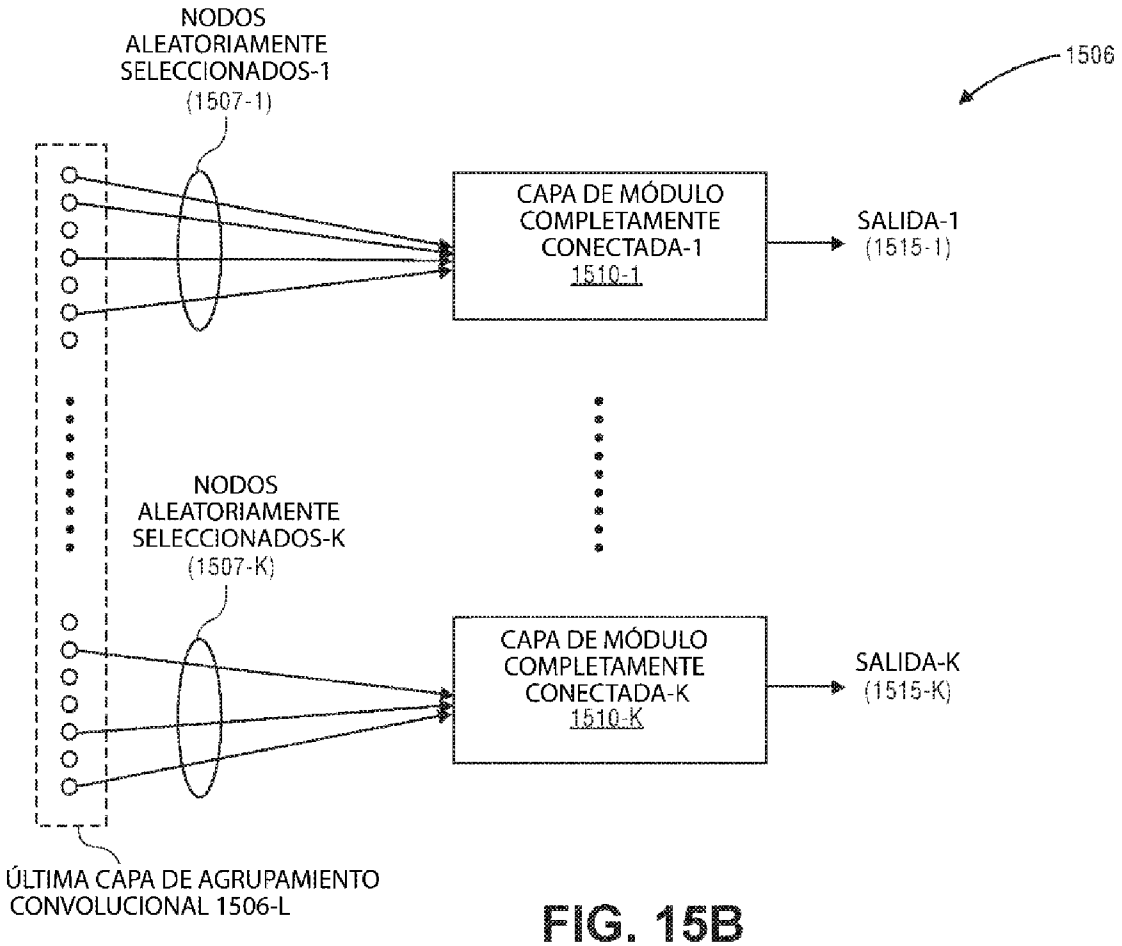

NODOS ALEATORIAMENTE SELECCIONADOS-1 (1507-1)
1506
CAPA DE MÓDULO COMPLETAMENTE CONECTADA-1 1510-1
SALIDA-1 (1515-1)
NODOS ALEATORIAMENTE SELECCIONADOS-K (1507-K)
CAPA DE MÓDULO COMPLETAMENTE CONECTADA-K 1510-K
SALIDA-K (1515-K)
ÚLTIMA CAPA DE AGRUPAMIENTO CONVOLUCIONAL 1506-L

FIG. 15B

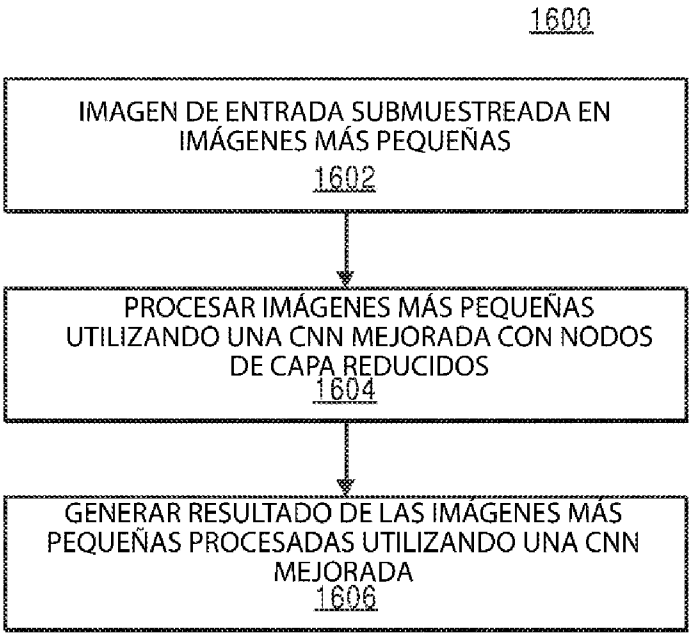


FIG. 16A

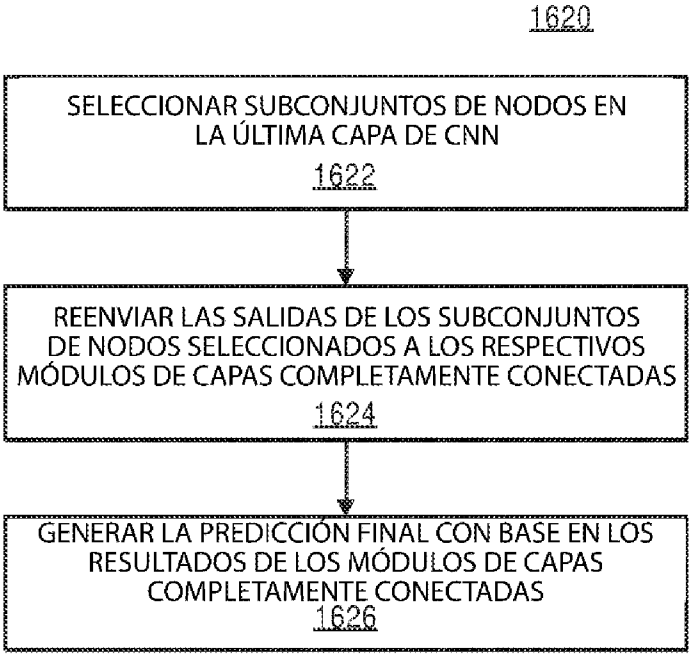


FIG. 16B

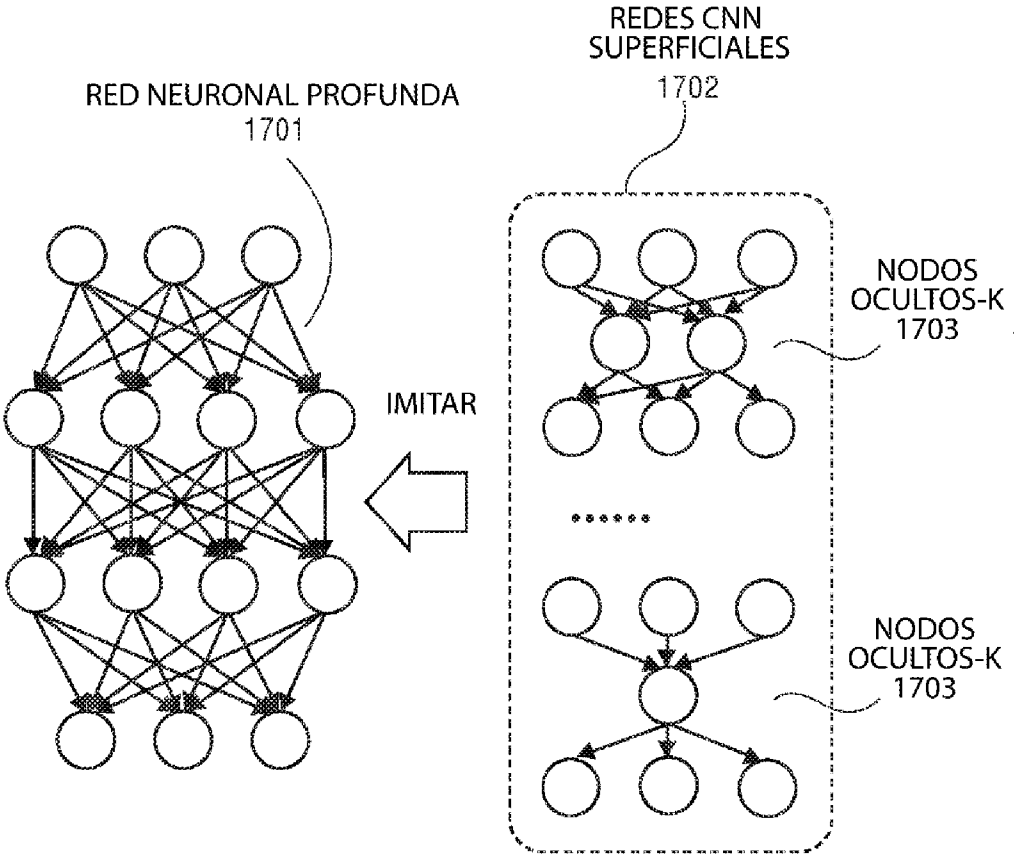

REDES CNN
SUPERFICIALES
1702
RED NEURONAL PROFUNDA
1701
NODOS
OCULTOS-K
1703
IMITAR
NODOS
OCULTOS-K
1703

FIG. 17

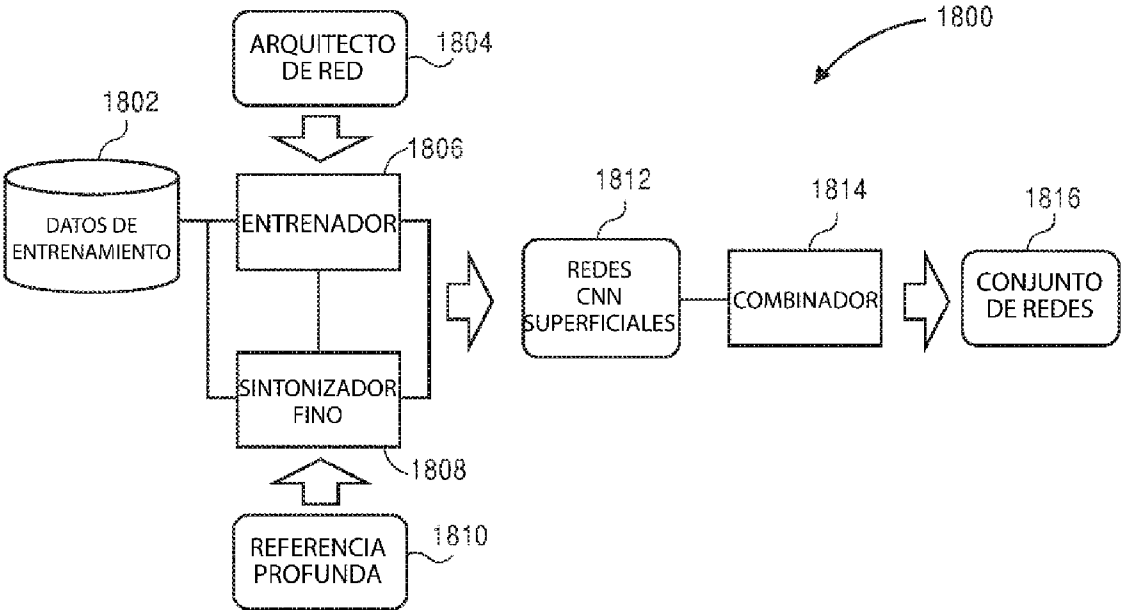


FIG. 18A

FIG. 18B

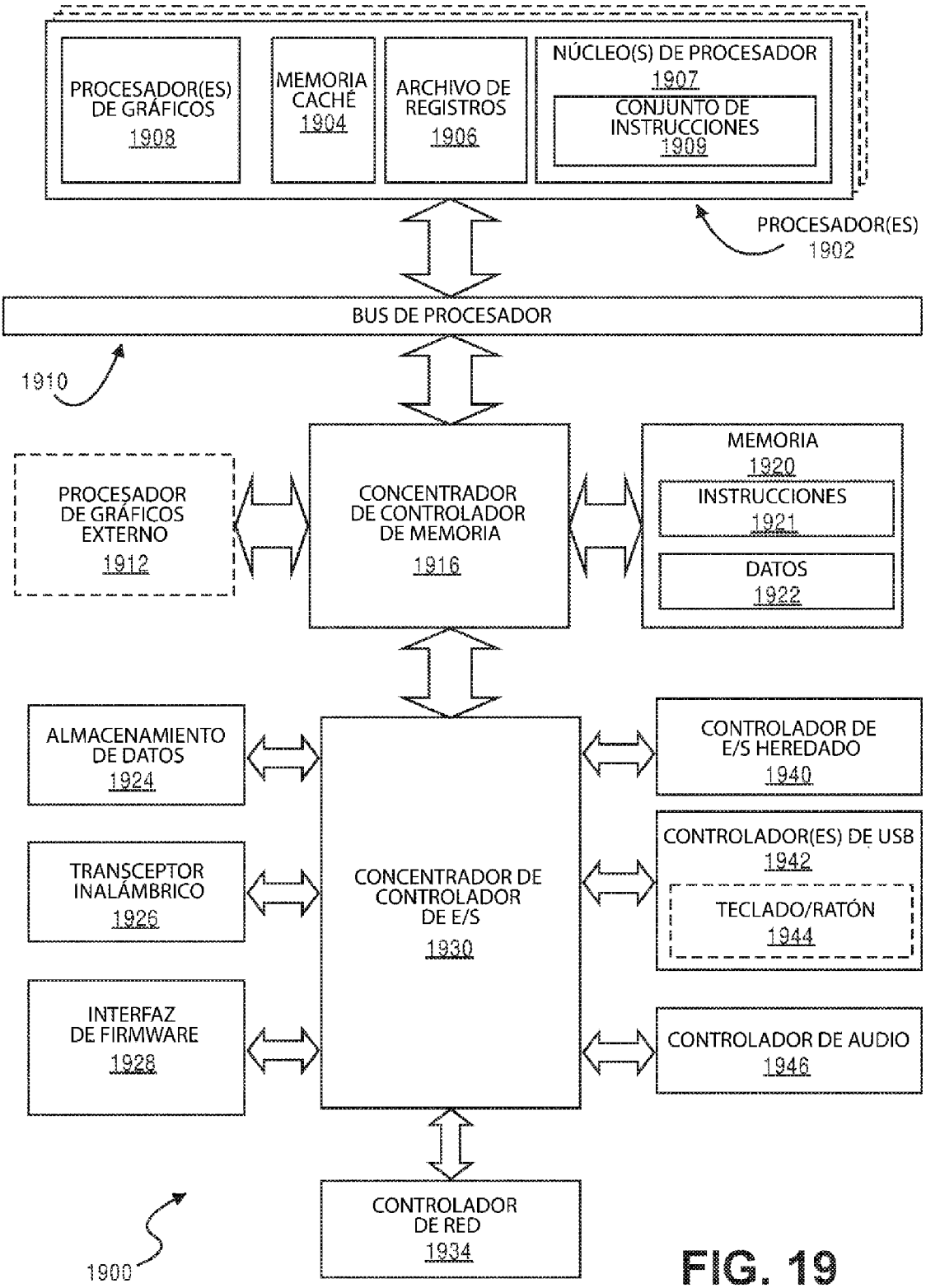

PROCESADOR(ES) DE GRÁFICOS 1908
MEMORIA CACHÉ 1904
ARCHIVO DE REGISTROS 1906
NÚCLEO(S) DE PROCESADOR 1907
CONJUNTO DE INSTRUCCIONES 1909
PROCESADOR(ES) 1902
BUS DE PROCESADOR
1910
PROCESADOR DE GRÁFICOS EXTERNO 1912
CONCENTRADOR DE CONTROLADOR DE MEMORIA 1916
MEMORIA 1920
INSTRUCCIONES 1921
DATOS 1922
ALMACENAMIENTO DE DATOS 1924
TRANSCEPTOR INALÁMBRICO 1926
INTERFAZ DE FIRMWARE 1928
CONCENTRADOR DE CONTROLADOR DE E/S 1930
CONTROLADOR DE E/S HEREDADO 1940
CONTROLADOR(ES) DE USB 1942
TECLADO/RATÓN 1944
CONTROLADOR DE AUDIO 1946
CONTROLADOR DE RED 1934
1900

FIG. 19

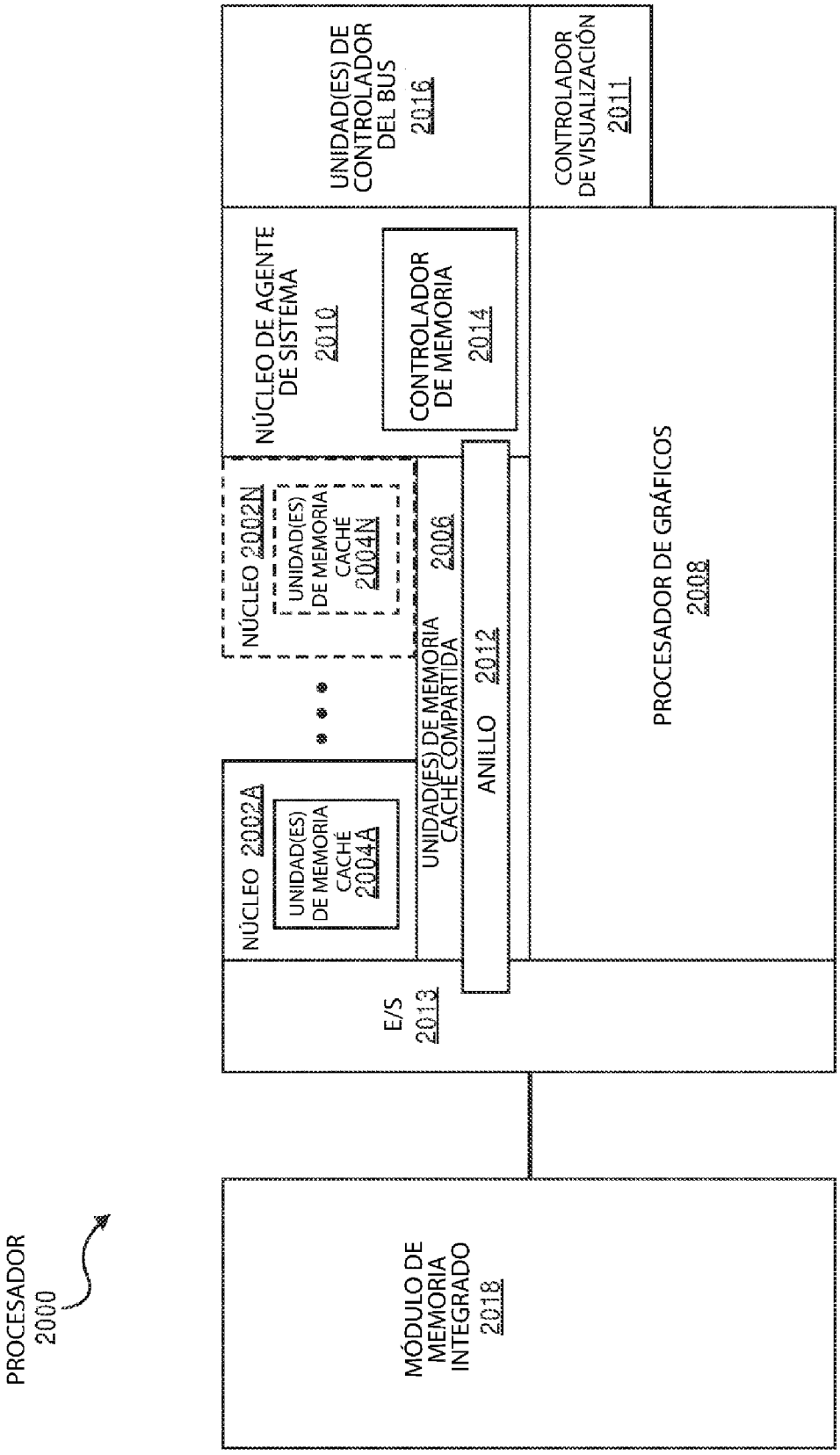

PROCESADOR
2000
MÓDULO DE MEMORIA INTEGRADO
2018
E/S
2013
NÚCLEO 2002A
UNIDAD(ES) DE MEMORIA CACHÉ 2004A
NÚCLEO 2002N
UNIDAD(ES) DE MEMORIA CACHÉ 2004N
UNIDAD(ES) DE MEMORIA CACHÉ COMPARTIDA 2006
ANILLO 2012
NÚCLEO DE AGENTE DE SISTEMA 2010
CONTROLADOR DE MEMORIA 2014
UNIDAD(ES) DE CONTROLADOR DEL BUS 2016
PROCESADOR DE GRÁFICOS 2008
CONTROLADOR DE VISUALIZACIÓN 2011

FIG. 20

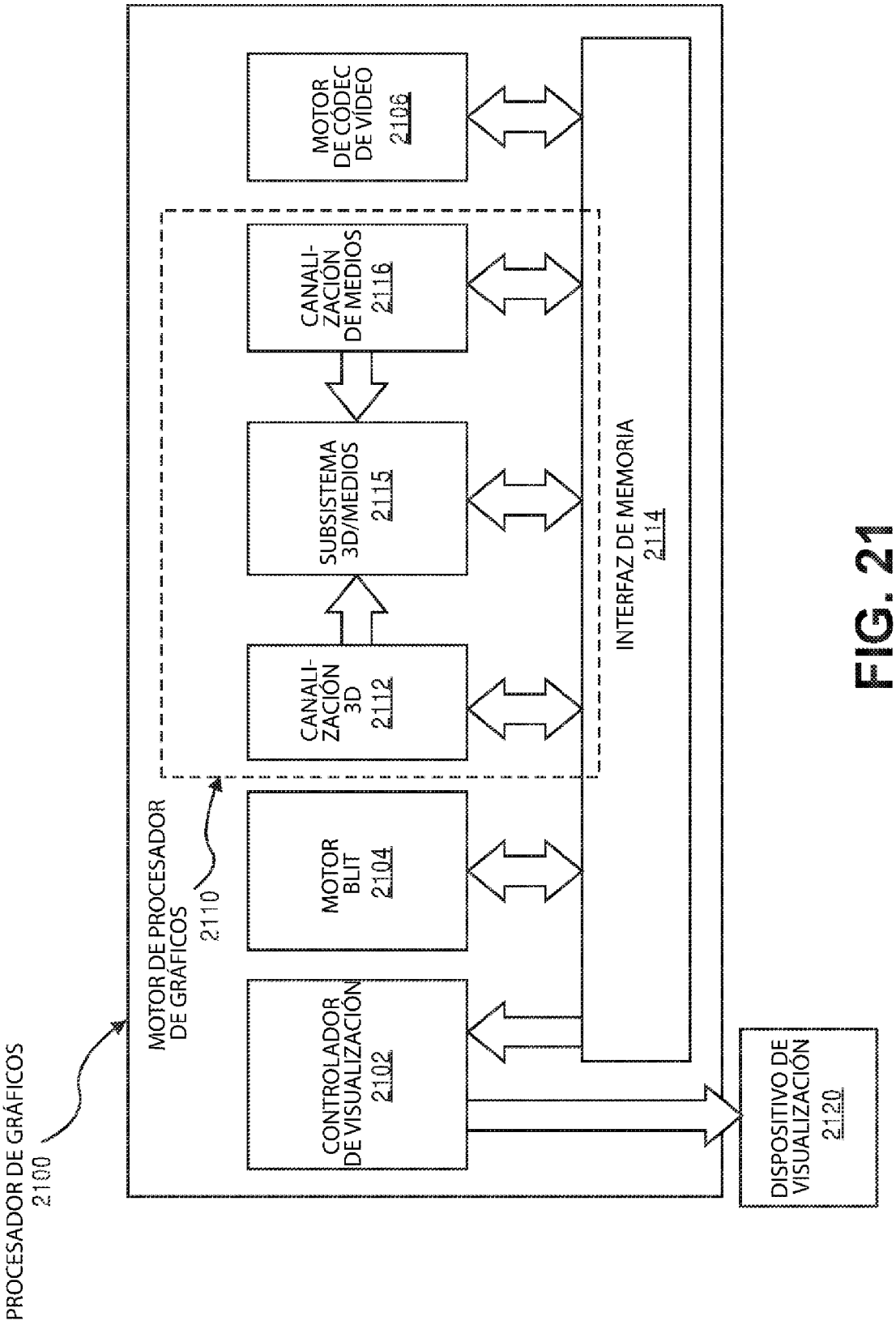

PROCESADOR DE GRÁFICOS
2100
MOTOR DE PROCESADOR DE GRÁFICOS
2110
CONTROLADOR DE VISUALIZACIÓN
2102
MOTOR BLIT
2104
CANALI-ZACIÓN 3D
2112
SUBSISTEMA 3D/MEDIOS
2115
CANALI-ZACIÓN DE MEDIOS
2116
MOTOR DE CÓDEC DE VÍDEO
2106
INTERFAZ DE MEMORIA
2114
DISPOSITIVO DE VISUALIZACIÓN
2120

FIG. 21

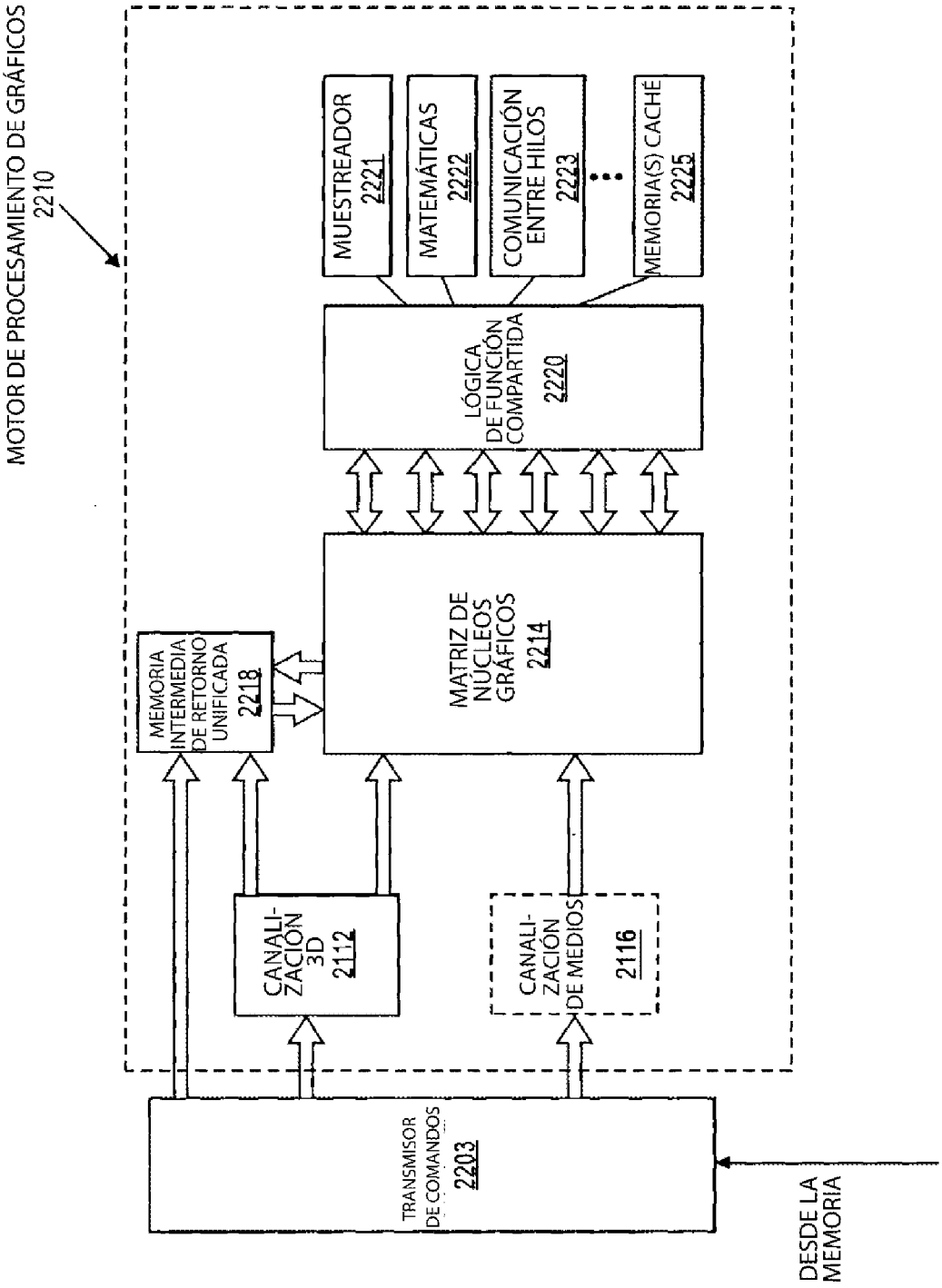

MOTOR DE PROCESAMIENTO DE GRÁFICOS
2210
MUESTREADOR
2221
MATEMÁTICAS
2222
COMUNICACIÓN ENTRE HILOS
2223
MEMORIA(S) CACHÉ
2225
LÓGICA DE FUNCIÓN COMPARTIDA
2220
MATRIZ DE NÚCLEOS GRÁFICOS
2214
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
2218
CANALI-ZACIÓN 3D
2112
CANALI-ZACIÓN DE MEDIOS
2116
TRANSMISOR DE COMANDOS
2203
DESDE LA MEMORIA

FIG. 22

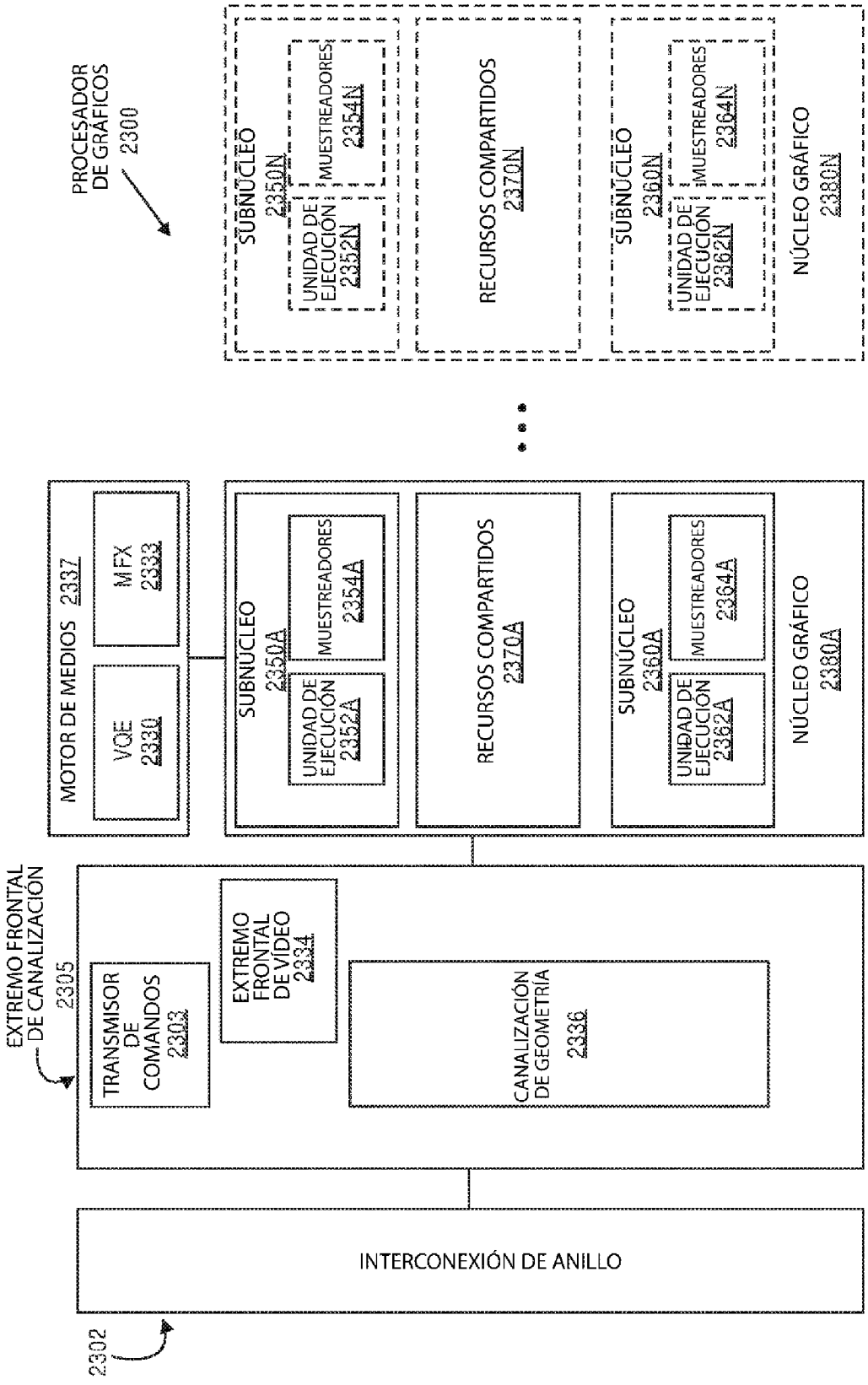

PROCESADOR DE GRÁFICOS 2300
SUBNÚCLEO 2350N
UNIDAD DE EJECUCIÓN 2352N
MUESTREADORES 2354N
RECURSOS COMPARTIDOS 2370N
SUBNÚCLEO 2360N
UNIDAD DE EJECUCIÓN 2362N
MUESTREADORES 2364N
NÚCLEO GRÁFICO 2380N
MOTOR DE MEDIOS 2337
VQE 2330
MFX 2333
SUBNÚCLEO 2350A
UNIDAD DE EJECUCIÓN 2352A
MUESTREADORES 2354A
RECURSOS COMPARTIDOS 2370A
SUBNÚCLEO 2360A
UNIDAD DE EJECUCIÓN 2362A
MUESTREADORES 2364A
NÚCLEO GRÁFICO 2380A
EXTREMO FRONTAL DE CANALIZACIÓN 2305
TRANSMISOR DE COMANDOS 2302
EXTREMO FRONTAL DE VÍDEO 2334
CANALIZACIÓN DE GEOMETRÍA 2336
INTERCONEXIÓN DE ANILLO
2302

FIG. 23

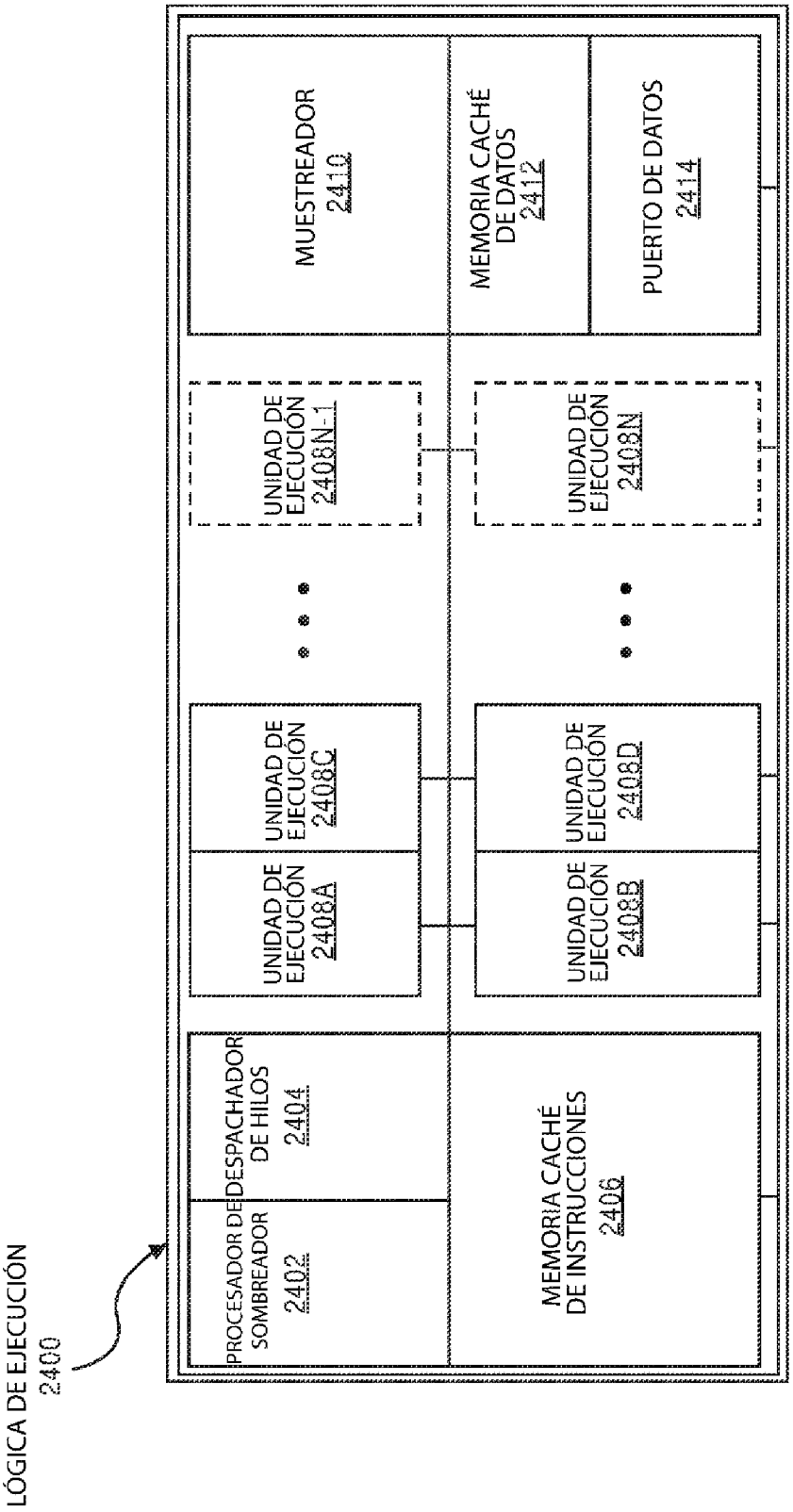

LÓGICA DE EJECUCIÓN
2400
PROCESADOR DE SOMBREADOR
2402
DESPACHADOR DE HILOS
2404
MEMORIA CACHÉ DE INSTRUCCIONES
2406
UNIDAD DE EJECUCIÓN
2408A
UNIDAD DE EJECUCIÓN
2408B
UNIDAD DE EJECUCIÓN
2408C
UNIDAD DE EJECUCIÓN
2408D
UNIDAD DE EJECUCIÓN
2408N-1
UNIDAD DE EJECUCIÓN
2408N
MUESTREADOR
2410
MEMORIA CACHÉ DE DATOS
2412
PUERTO DE DATOS
2414

FIG. 24

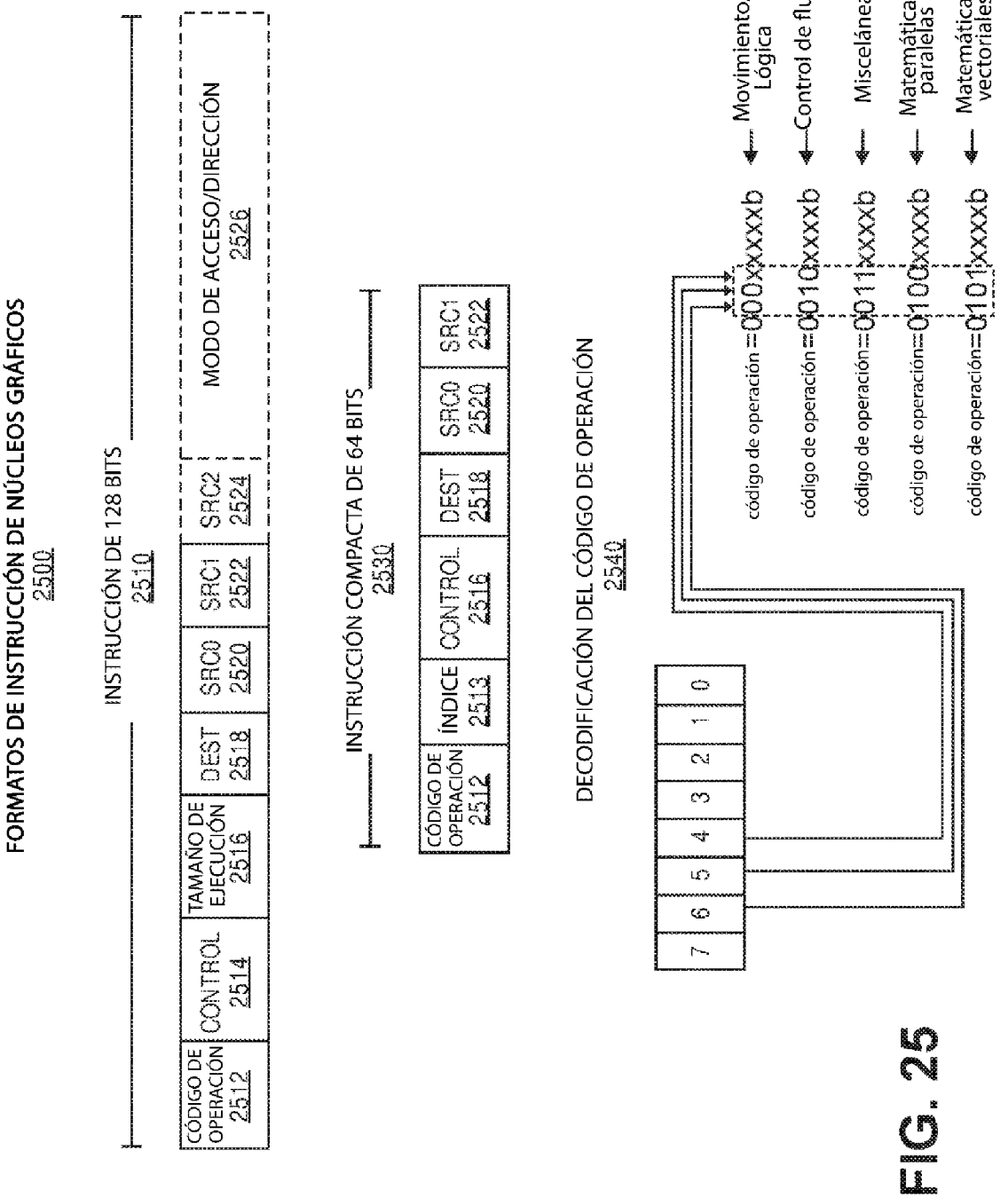

FORMATOS DE INSTRUCCIÓN DE NÚCLEOS GRÁFICOS
2500
INSTRUCCIÓN DE 128 BITS
2510
CÓDIGO DE OPERACIÓN 2512
CONTROL 2514
TAMAÑO DE EJECUCIÓN 2516
DEST 2518
SRC0 2520
SRC1 2522
SRC2 2524
MODO DE ACCESO/DIRECCIÓN 2526
INSTRUCCIÓN COMPACTA DE 64 BITS
2530
CÓDIGO DE OPERACIÓN 2512
ÍNDICE 2513
CONTROL 2516
DEST 2518
SRC0 2520
SRC1 2522
DECODIFICACIÓN DEL CÓDIGO DE OPERACIÓN
2540
7 6 5 4 3 2 1 0
código de operación = 00 0xxxxb
código de operación = 00 10xxxb
código de operación = 00 11xxxb
código de operación = 01 00xxxb
código de operación = 01 01xxxb
Movimiento/ Lógica
2542
Control de flujo
2544
Miscelánea
2546
Matemáticas paralelas
2548
Matemáticas vectoriales
2550

FIG. 25

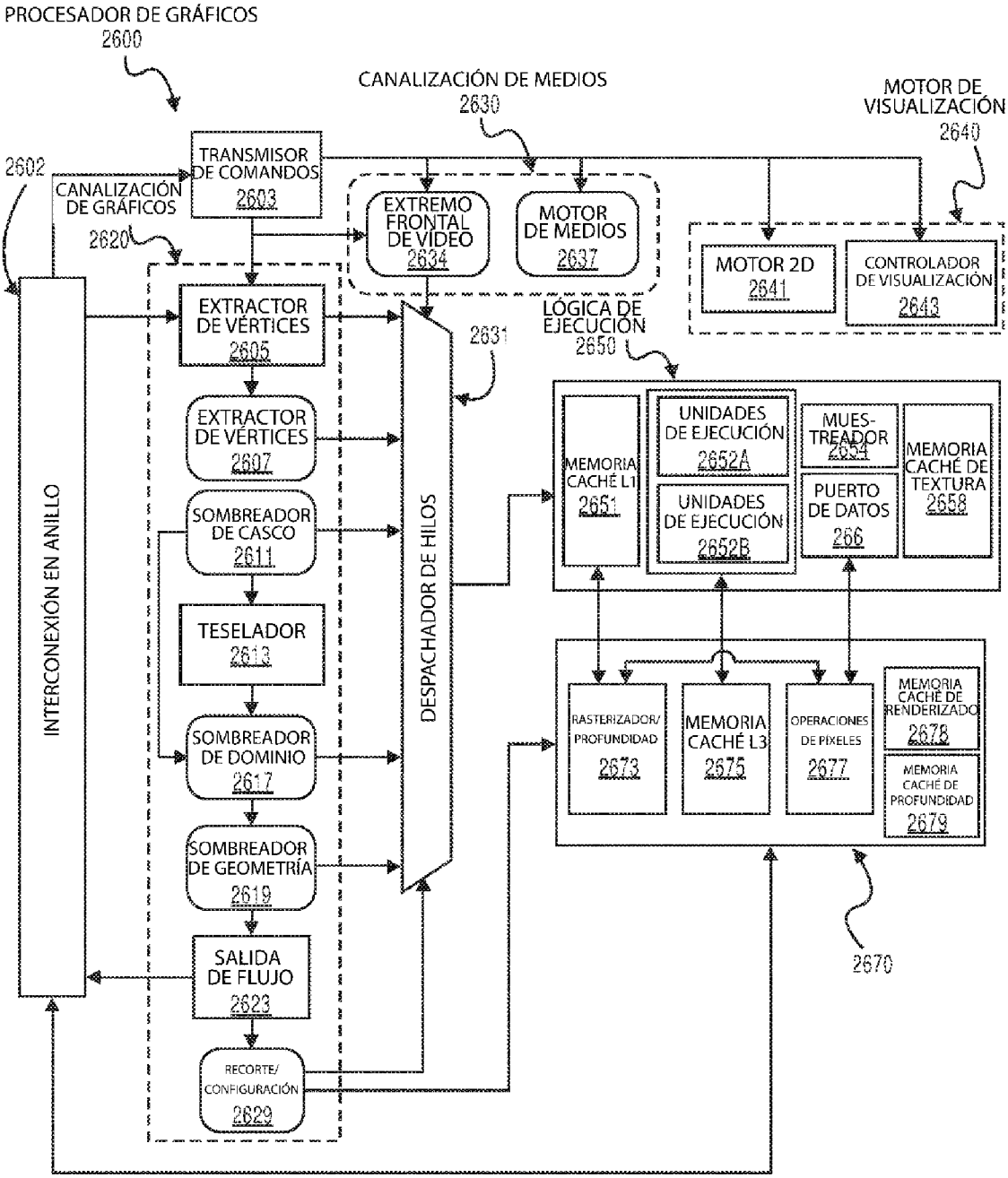

PROCESADOR DE GRÁFICOS
2600
CANALIZACIÓN DE MEDIOS
2630
MOTOR DE VISUALIZACIÓN
2640
2602
TRANSMISOR DE COMANDOS
2603
CANALIZACIÓN DE GRÁFICOS
2620
EXTREMO FRONTAL DE VÍDEO
2634
MOTOR DE MEDIOS
2637
MOTOR 2D
2641
CONTROLADOR DE VISUALIZACIÓN
2643
EXTRACTOR DE VÉRTICES
2605
LÓGICA DE EJECUCIÓN
2650
2631
EXTRACTOR DE VÉRTICES
2607
SOMBREADOR DE CASCO
2611
TESELADOR
2613
SOMBREADOR DE DOMINIO
2617
SOMBREADOR DE GEOMETRÍA
2619
SALIDA DE FLUJO
2623
RECORTE/ CONFIGURACIÓN
2629
INTERCONEXIÓN EN ANILLO
DESPACHADOR DE HILOS
MEMORIA CACHÉ L1
2651
UNIDADES DE EJECUCIÓN
2652A
UNIDADES DE EJECUCIÓN
2652B
MUES-TREADOR
2654
PUERTO DE DATOS
266
MEMORIA CACHÉ DE TEXTURA
2658
RASTERIZADOR/ PROFUNDIDAD
2673
MEMORIA CACHÉ L3
2675
OPERACIONES DE PÍXELES
2677
MEMORIA CACHÉ DE RENDERIZADO
2678
MEMORIA CACHÉ DE PROFUNDIDAD
2679
2670

FIG. 26

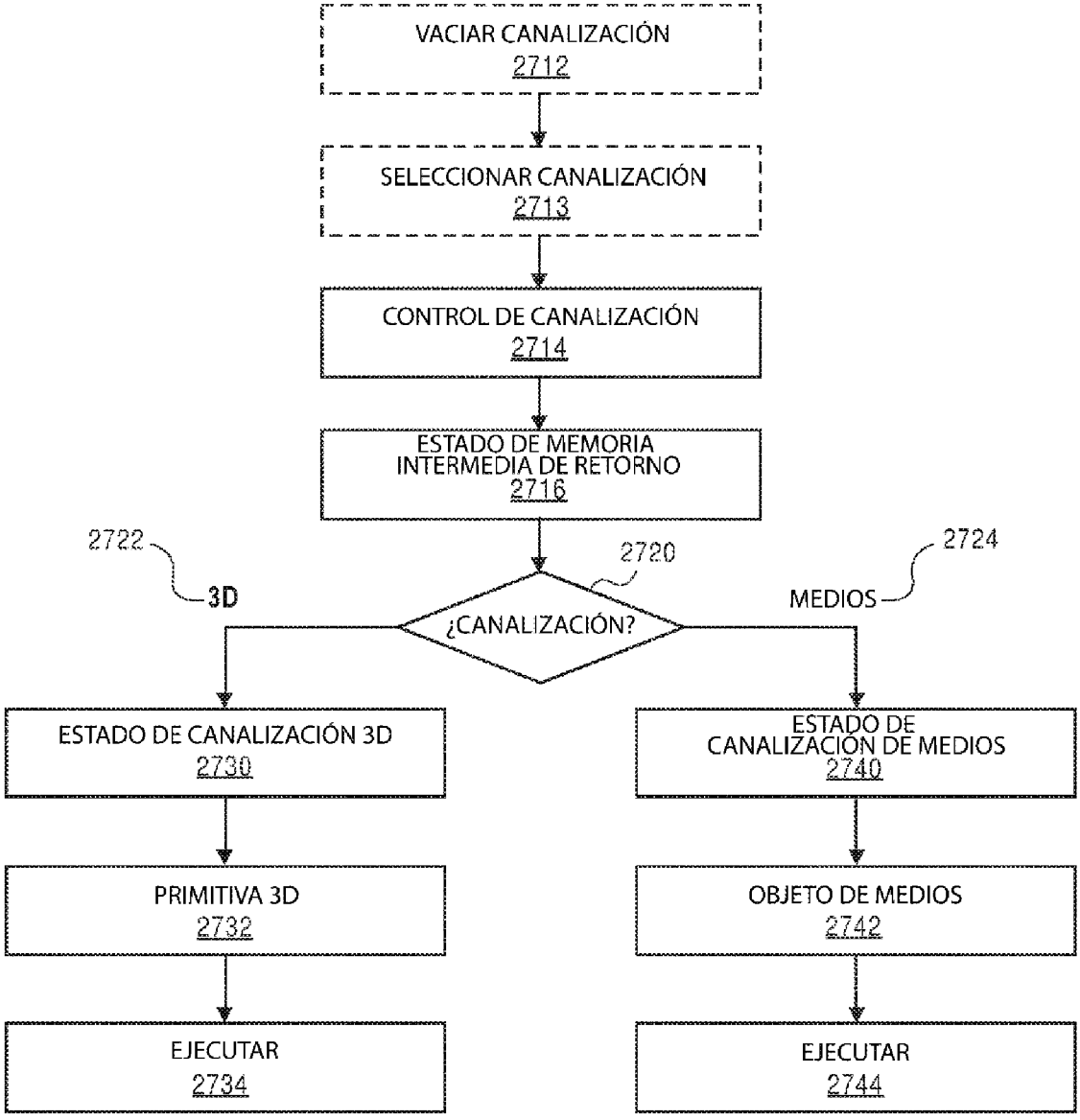
FIG. 27A

FORMATO DE COMANDO DE MUESTRA
2700

CLIENTE 2702	CÓDIGO DE OPERACIÓN 2704	SUBCÓDIGO DE OPERACIÓN 2705	DATOS 2706	TAMAÑO DE COMANDO 2708

FIG. 27B

SECUENCIA DE COMANDOS DE MUESTRA
2710



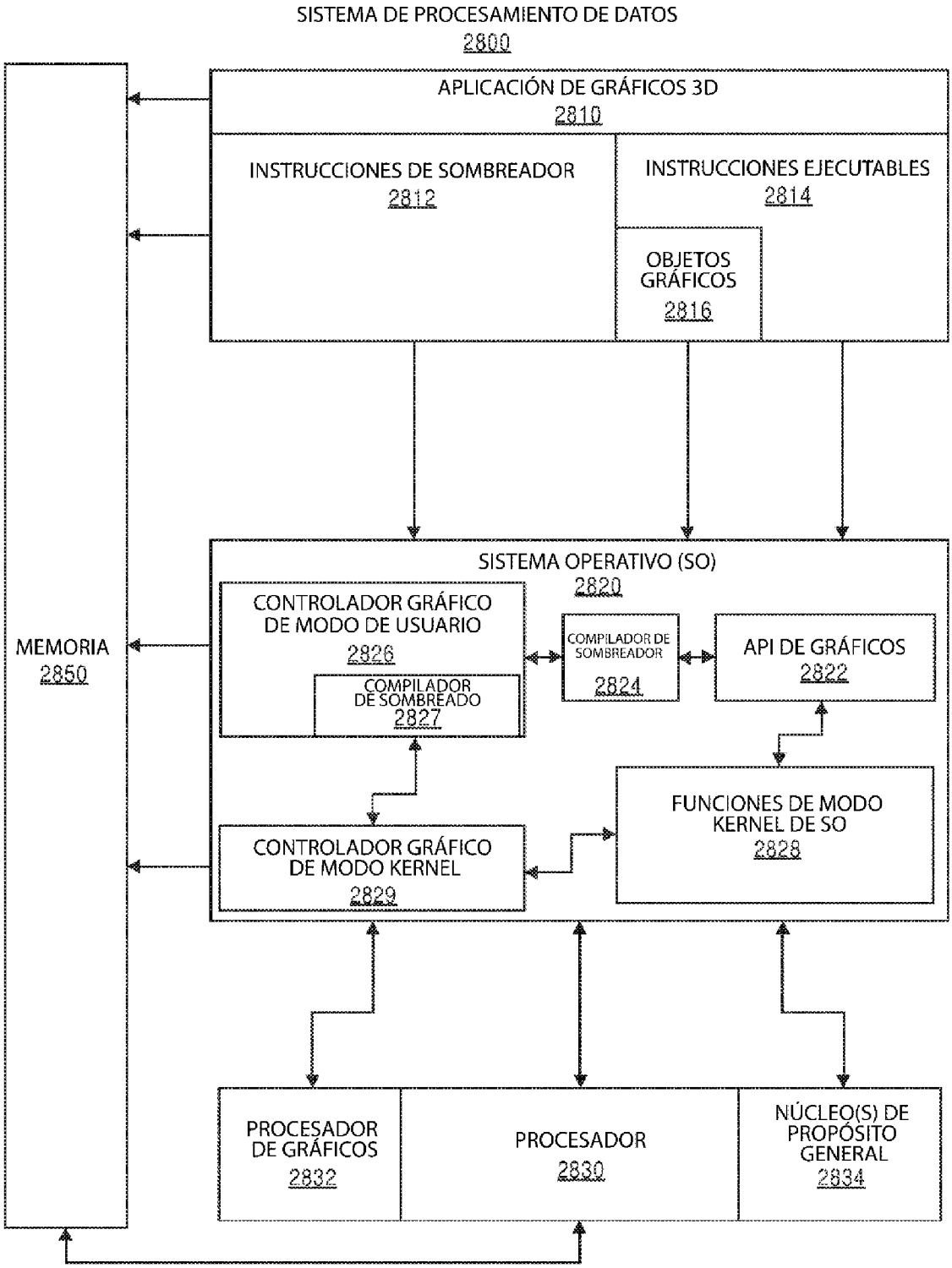

SISTEMA DE PROCESAMIENTO DE DATOS
2800
APLICACIÓN DE GRÁFICOS 3D
2810
INSTRUCCIONES DE SOMBREADOR
2812
INSTRUCCIONES EJECUTABLES
2814
OBJETOS GRÁFICOS
2816
MEMORIA
2850
SISTEMA OPERATIVO (SO)
2820
CONTROLADOR GRÁFICO DE MODO DE USUARIO
2826
COMPILADOR DE SOMBREADO
2827
COMPILADOR DE SOMBREADOR
2824
API DE GRÁFICOS
2822
FUNCIONES DE MODO KERNEL DE SO
2828
CONTROLADOR GRÁFICO DE MODO KERNEL
2829
PROCESADOR DE GRÁFICOS
2832
PROCESADOR
2830
NÚCLEO(S) DE PROPÓSITO GENERAL
2834

FIG. 28

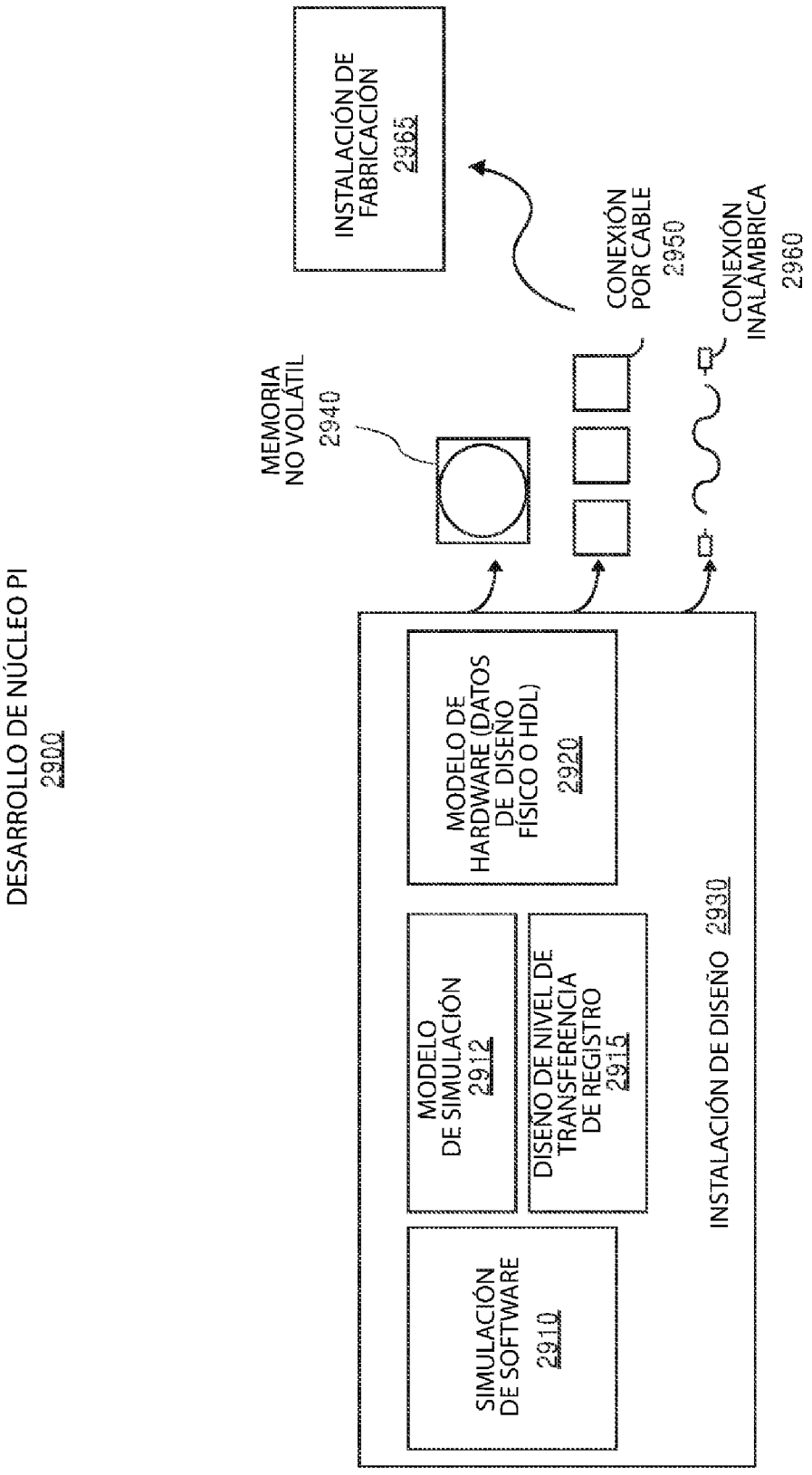

DESARROLLO DE NÚCLEO PI
2900
INSTALACIÓN DE DISEÑO 2930
SIMULACIÓN DE SOFTWARE 2910
MODELO DE SIMULACIÓN 2912
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 2915
MODELO DE HARDWARE (DATOS DE DISEÑO FÍSICO O HDL) 2920
MEMORIA NO VOLÁTIL 2940
CONEXIÓN POR CABLE 2950
CONEXIÓN INALÁMBRICA 2960
INSTALACIÓN DE FABRICACIÓN 2965

FIG. 29

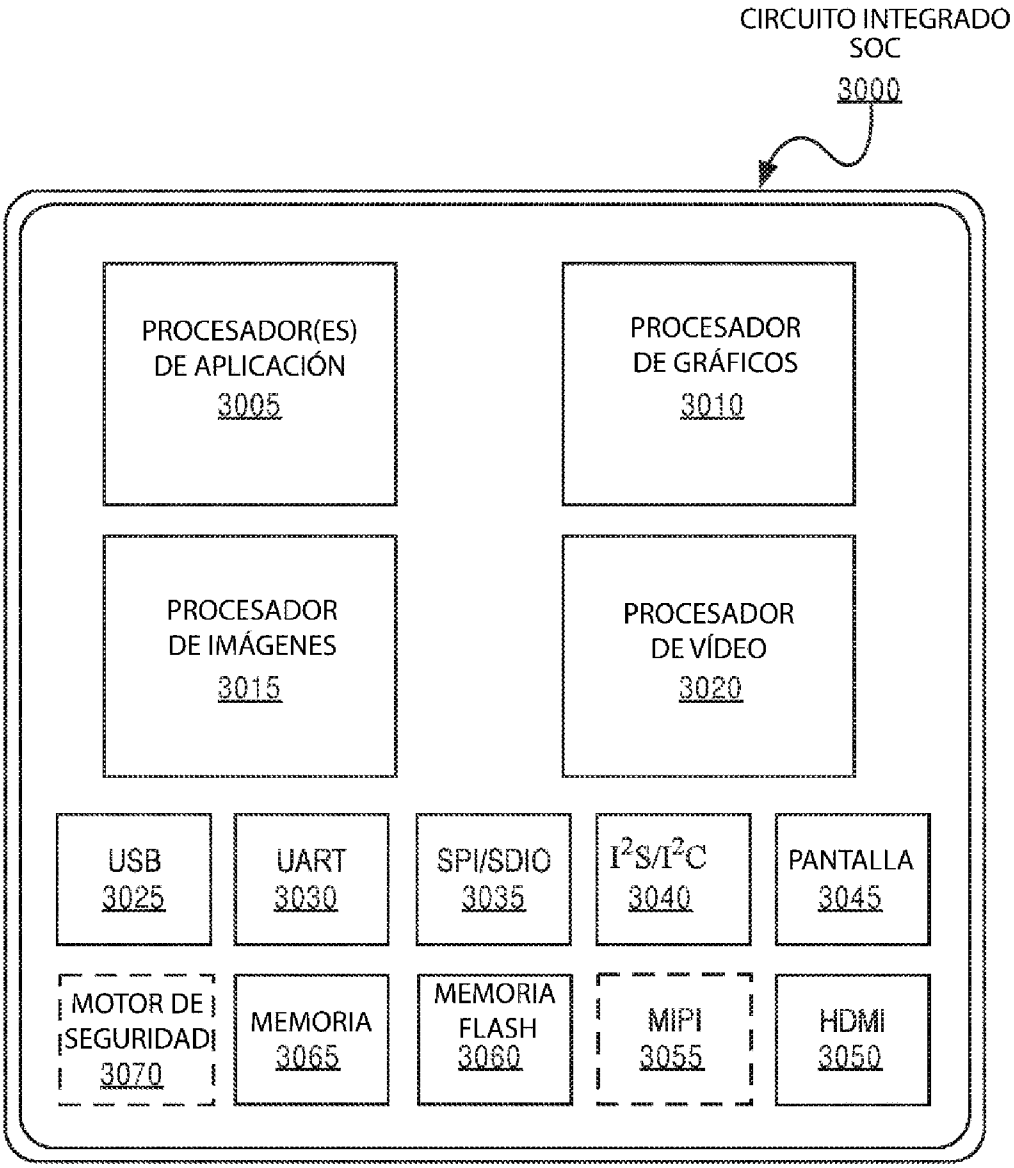

CIRCUITO INTEGRADO SOC
3000
PROCESADOR(ES) DE APLICACIÓN
3005
PROCESADOR DE GRÁFICOS
3010
PROCESADOR DE IMÁGENES
3015
PROCESADOR DE VÍDEO
3020
USB
3025
UART
3030
SPI/SDIO
3035
I2S/I2C
3040
PANTALLA
3045
MOTOR DE SEGURIDAD
3070
MEMORIA
3065
MEMORIA FLASH
3060
MIPI
3055
HDMI
3050

FIG. 30

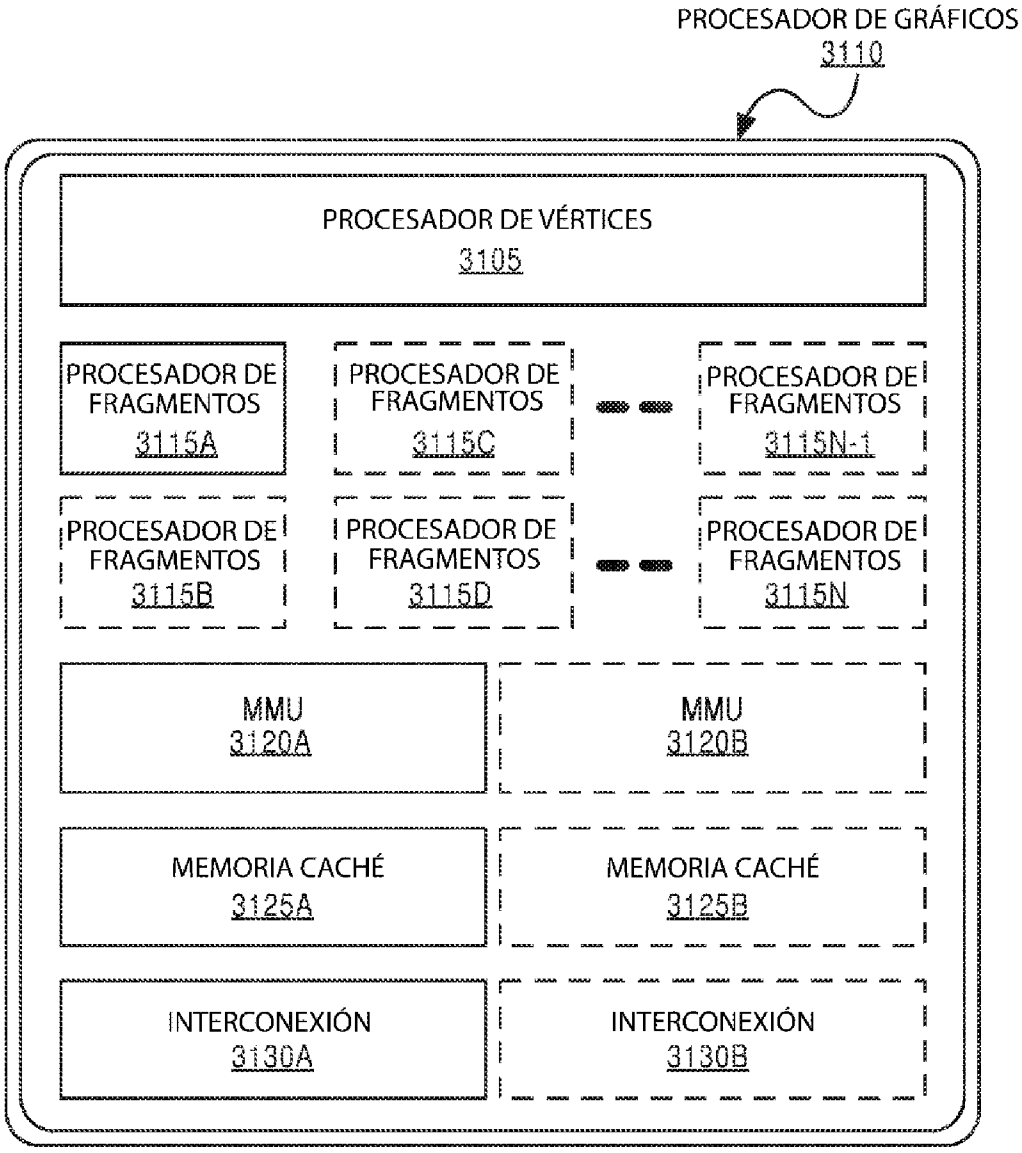

PROCESADOR DE GRÁFICOS
3110
PROCESADOR DE VÉRTICES
3105
PROCESADOR DE FRAGMENTOS
3115A
PROCESADOR DE FRAGMENTOS
3115C
PROCESADOR DE FRAGMENTOS
3115N-1
PROCESADOR DE FRAGMENTOS
3115B
PROCESADOR DE FRAGMENTOS
3115D
PROCESADOR DE FRAGMENTOS
3115N
MMU
3120A
MMU
3120B
MEMORIA CACHÉ
3125A
MEMORIA CACHÉ
3125B
INTERCONEXIÓN
3130A
INTERCONEXIÓN
3130B

FIG. 31

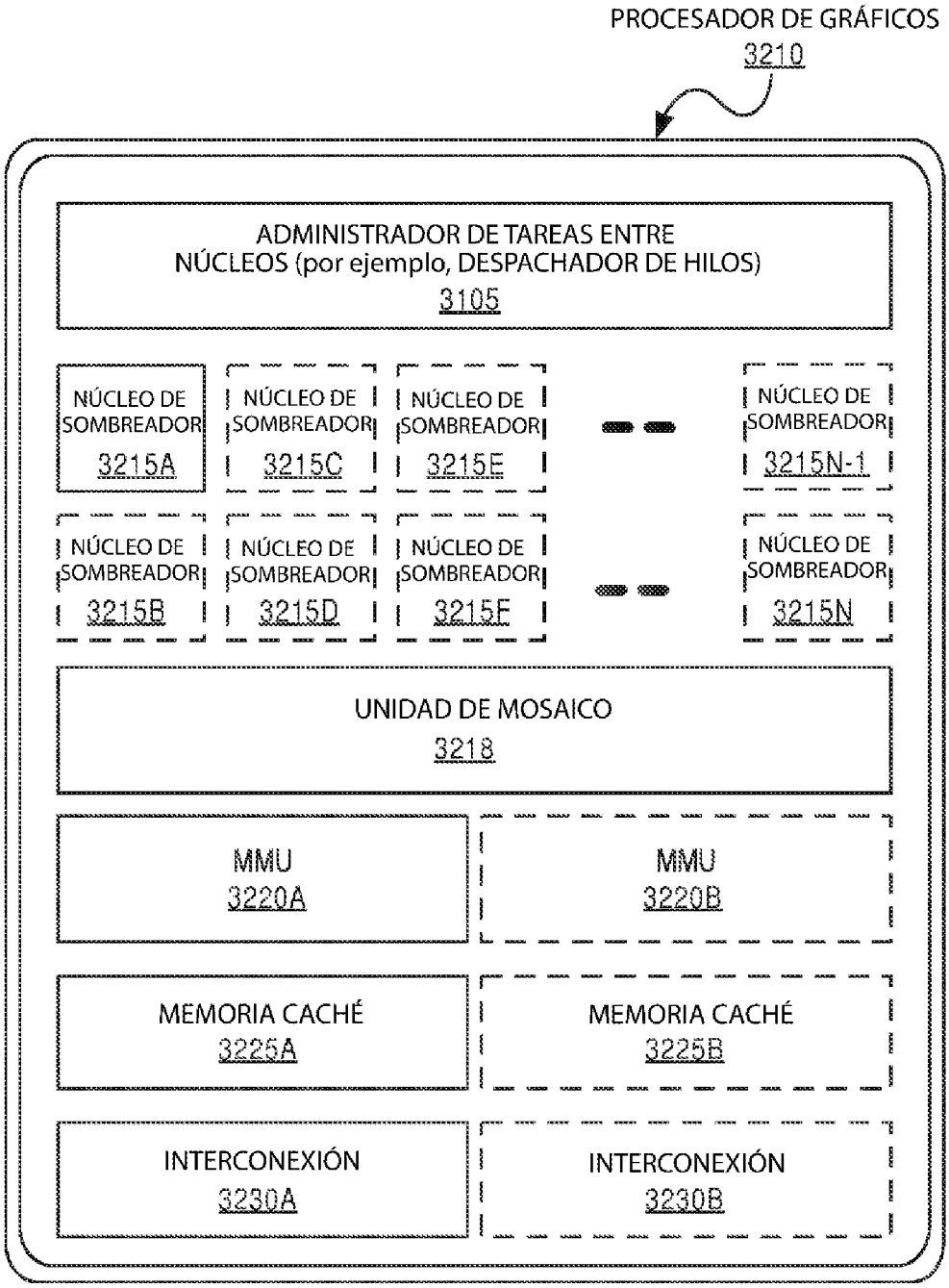

PROCESADOR DE GRÁFICOS
3210
ADMINISTRADOR DE TAREAS ENTRE NÚCLEOS (por ejemplo, DESPACHADOR DE HILOS)
3105
NÚCLEO DE SOMBREADOR
3215A
NÚCLEO DE SOMBREADOR
3215C
NÚCLEO DE SOMBREADOR
3215E
NÚCLEO DE SOMBREADOR
3215N-1
NÚCLEO DE SOMBREADOR
3215B
NÚCLEO DE SOMBREADOR
3215D
NÚCLEO DE SOMBREADOR
3215F
NÚCLEO DE SOMBREADOR
3215N
UNIDAD DE MOSAICO
3218
MMU
3220A
MMU
3220B
MEMORIA CACHÉ
3225A
MEMORIA CACHÉ
3225B
INTERCONEXIÓN
3230A
INTERCONEXIÓN
3230B

FIG. 32