(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 930 550**

(51) Int. CI.:

G06F 15/76 (2006.01)
G06T 1/20 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **07.04.2017 PCT/CN2017/079726**

(87) Fecha y número de publicación internacional: **11.10.2018 WO18184208**

(96) Fecha de presentación y número de la solicitud europea: **07.04.2017 E 17904717 (0)**

(97) Fecha y número de publicación de la concesión europea: **31.08.2022 EP 3607453**

(54) Título: **Métodos y aparatos para canalización de ejecución de red de aprendizaje profundo en plataforma multiprocesador**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
16.12.2022

(73) Titular/es:

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

(72) Inventor/es:

YANG, LIU y
YAO, ANBANG

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

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

DESCRIPCIÓN

Métodos y aparatos para canalización de ejecución de red de aprendizaje profundo en plataforma multiprocesador

CAMPO

Las realizaciones se refieren, en general, al procesamiento de datos y, más particularmente, al procesamiento de datos a través de una unidad de procesamiento de gráficos de propósito general. En particular, las realizaciones se refieren a sistemas y métodos para ejecutar una red de aprendizaje profundo en una canalización de un sistema multiprocesador.

ANTECEDENTES

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

Para aumentar adicionalmente el desempeño, los procesadores de gráficos habitualmente implementan técnicas de procesamiento tales como encauzamiento en canalizaciones, que intentan procesar, en paralelo, tantos datos de gráficos como sea posible a lo largo de todas las diferentes partes de la canalización de gráficos. Los procesadores de gráficos paralelos con arquitecturas de múltiples hilos y única instrucción (SIMT) están diseñados para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura de SIMT, grupos de hilos paralelos intentan ejecutar instrucciones de programa de manera síncrona conjuntamente tan a menudo como sea posible para aumentar la eficiencia de procesamiento. Puede encontrarse una vista global general del software y el hardware para las arquitecturas SIMT en Shane Cook, CUDA Programming capítulo 3, páginas 37-51 (2013).

El aprendizaje automático ha tenido éxito en la resolución de muchos tipos de tareas. Los cálculos que surgen al entrenar y usar 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 las redes neuronales profundas. Los procesadores de gráficos paralelos con arquitecturas de múltiples hilos y única instrucción (SIMT) están diseñados para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura de SIMT, grupos de hilos paralelos intentan ejecutar instrucciones de programa de manera síncrona conjuntamente tan a menudo como sea posible para aumentar la eficiencia de procesamiento. La eficiencia proporcionada por 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.

Una red neuronal de aprendizaje profundo (DNN) típicamente está estructurada como un tipo de red neuronal convolucional y se usa para realizar tareas asociativas complejas. Después de una fase de entrenamiento con entradas conocidas, la DNN puede reconocer nuevas entradas que son similares a la entrada de entrenamiento original. Esto es útil para tecnologías de detección de objetos, reconocimiento automático de voz, autenticación de usuarios, comprensión de imágenes y usos de visión artificial, entre otras cosas. Se puede usar una secuencia de vídeo para el seguimiento de objetos, así como para el reconocimiento.

DNN presenta una demanda de procesamiento computacional significativa en un sistema, pero muchas de las tareas son repetitivas. Con una canalización de múltiples núcleos, el procesamiento de una carga de trabajo de DNN puede acelerarse. Una DNN normalmente está formada como una red con múltiples nodos en un grafo acíclico. Los datos fluyen en una dirección de un nodo al siguiente con divisiones y uniones apropiadas a través del grafo.

El documento US 2012/304194 A1 desvela: Se proporcionan un aparato y un método de procesamiento de datos para procesar una carga de trabajo recibida para generar datos de resultados. Un generador de grupos de hilos genera a partir de la carga de trabajo recibida una pluralidad de grupos de hilos que se ejecutarán para procesar la carga de trabajo recibida. Cada grupo de hilos consiste en una pluralidad de hilos, y al menos un grupo de hilos tiene una dependencia entre hilos que existe entre la pluralidad de hilos. Cada hilo puede ser un hilo activo cuya salida se requiere para formar los datos de resultado, o un hilo ficticio requerido para resolver la dependencia entre hilos para uno de los hilos activos pero cuya salida no se requiere para formar los datos de resultado. El generador de grupos de hilos identifica para cada grupo de hilos cualquier hilo ficticio dentro de ese grupo de hilos. A continuación, una unidad de ejecución de hilos ejecuta cada hilo dentro de un grupo de hilos recibido del generador de grupos de hilos mediante la ejecución de un programa predeterminado que comprende una pluralidad de instrucciones de programa. La circuitería de modificación del flujo de ejecución responde al grupo de hilos recibido que tiene al menos un hilo ficticio, para hacer que la unidad de ejecución de hilos omita selectivamente al menos parte de la ejecución de al menos una de la pluralidad de instrucciones cuando se ejecuta cada hilo ficticio, en dependencia de la información de

control asociada con el programa predeterminado. En una realización particular, la carga de trabajo recibida es una carga de trabajo de representación de gráficos y la unidad de ejecución de hilos realiza operaciones de representación de gráficos para generar como resultado valores de píxeles de datos y valores de control asociados. Este enfoque puede generar mejoras significativas en el rendimiento, así como reducir el consumo de energía.

SUMARIO

La invención está definida por las reivindicaciones independientes. Realizaciones ventajosas son objeto de las reivindicaciones dependientes.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

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

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

Las **Figuras 2A-2D** ilustran los componentes de un procesador paralelo para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

Las **Figuras 3A-3B** son diagramas de bloques de multiprocesadores gráficos para implementar uno o más aspectos de realizaciones ilustrativas descritas en el presente documento.

Las **Figuras 4A-4F** ilustran una arquitectura ilustrativa en la que una pluralidad de Unidades de Procesamiento de Gráficos (GPU) se acopla comunicativamente a una pluralidad de procesadores de múltiples núcleos.

La **Figura 5** ilustra una canalización de procesamiento de gráficos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 6** ilustra una pila de software de aprendizaje automático para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 7** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 8** ilustra un sistema informático multi-GPU para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

Las **Figuras 9A-9B** ilustran capas de redes neuronales profundas ilustrativas.

La **Figura 10** ilustra una red neuronal recurrente ilustrativa.

La **Figura 11** ilustra el entrenamiento y despliegue de una red neuronal profunda.

La **Figura 12** es un diagrama de bloques ilustrativo que ilustra un aprendizaje distribuido.

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

La **Figura 14** es un diagrama ilustrativo de almacenamiento de datos de canal RGB y CNN como parte de una imagen de canal profundo que es un diagrama de bloques ilustrativo de un sistema de captura de imágenes que almacena una imagen de canal profundo usando una red neuronal convolucional (CNN).

La **Figura 15** es un diagrama de nodos secuenciales de una red de aprendizaje profundo.

La **Figura 16** es un diagrama de agrupación de los nodos de la Figura 15.

La **Figura 17** es un diagrama de una plataforma de procesamiento de múltiples núcleos configurada para ejecutar una red de aprendizaje profundo de acuerdo con una realización de la invención.

La **Figura 18** es un diagrama de flujo de proceso de la ejecución de una red de aprendizaje profundo en una plataforma de procesamiento de múltiples núcleos de acuerdo con una realización de la invención.

La **Figura 19** ilustra un diagrama de bloques de un sistema de procesamiento para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 20** ilustra un diagrama de bloques ilustrativo 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 ilustrativo 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 para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 23** ilustra un diagrama de bloques de otro procesador de gráficos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 24** ilustra la lógica de ejecución de hilos que incluye una matriz de elementos de procesamiento empleados de un motor de procesamiento de gráficos (GPE) para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 25** ilustra un diagrama de bloques de los formatos de instrucción de un procesador de gráficos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 26** ilustra un diagrama de bloques de una realización ilustrativa de un procesador de gráficos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 27A** ilustra un diagrama de bloques de un formato de comando de procesador de gráficos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 27B** ilustra un diagrama de bloques de una secuencia de comandos del procesador de gráficos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 28** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 29** ilustra un diagrama de bloques de un sistema de desarrollo de núcleo de IP que puede usarse para fabricar un circuito integrado (CI) para realizar operaciones para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 30** ilustra un diagrama de bloques de un sistema ilustrativo en un CI de chip que puede fabricarse usando uno o más núcleos IP para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 31** ilustra un diagrama de bloques de un procesador de gráficos ilustrativo en un sistema en un CI de chip que puede fabricarse usando uno o más núcleos de IP para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

La **Figura 32** ilustra un diagrama de bloques de un procesador de gráficos adicional ilustrativo de un sistema en un CI de chip que puede fabricarse usando uno o más núcleos de IP para implementar uno o más aspectos de las realizaciones ilustrativas descritas en el presente documento.

DESCRIPCIÓN DETALLADA

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

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

En la descripción siguiente, se exponen numerosos detalles específicos para proporcionar un entendimiento más

minucioso. Sin embargo, será evidente que las realizaciones descritas en el presente documento pueden ponerse en práctica sin uno o más de estos detalles específicos. En otras instancias, no se han descrito características bien conocidas para evitar oscurecer los detalles de las realizaciones ilustrativas.

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

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

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

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

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

Se apreciará que el sistema informático 100 mostrado en el presente documento es ilustrativo y que son posibles variaciones y modificaciones. La topología de conexión, incluyendo el número y disposición de puentes, el número del procesador o procesadores 102, y el número del procesador o procesadores paralelos 112, se puede modificar como se desee. Por ejemplo, en algunas realizaciones, la memoria de sistema 104 está conectada al procesador o

procesadores 102 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria de sistema 104 a través del concentrador de memoria 105 y el procesador o procesadores 102. En otras topologías alternativas, el procesador o procesadores paralelos 112 están conectados al concentrador de E/S 107 o directamente a uno de los uno o más procesadores 102, en lugar de al concentrador de memoria 105. En otras realizaciones, el concentrador de E/S 107 y el concentrador de memoria 105 se pueden integrar en un único chip. Algunas realizaciones pueden incluir dos o más conjuntos del procesador o procesadores 102 anexados a través de múltiples zócalos, que se pueden acoplar con dos o más instancias del procesador o procesadores paralelos 112.

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

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

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

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

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

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

En una realización, la matriz de agrupación de procesamiento 212 está configurada para realizar operaciones de procesamiento de gráficos paralelas. En realizaciones en las que el procesador paralelo 200 está configurado para realizar operaciones de procesamiento de gráficos, la matriz de agrupaciones de procesamiento 212 puede incluir

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

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

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

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

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

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

en diversos dispositivos de memoria externos. En una realización, la barra transversal de memoria 216 tiene una conexión a la interfaz de memoria 218 para comunicarse con la unidad de E/S 204, así como una conexión a una instancia local de la memoria de procesador paralelo 222, lo que posibilita que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 214A-214N se comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 202. En una realización, la barra transversal de memoria 216 puede usar canales virtuales para separar flujos de tráfico entre las agrupaciones 214A-214N y las unidades de subdivisión 220A-220N.

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

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

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

En algunas realizaciones, la ROP 226 se incluye dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 214A-214N de la Figura 2) en lugar de dentro de la unidad de subdivisión 220. En tal realización, se transmiten solicitudes de lectura y de escritura de datos de píxel a través de la barra transversal de memoria 216 en lugar de datos de fragmento de píxel. Los datos de gráficos procesados se pueden visualizar en un dispositivo de visualización, tal como uno de los uno o más dispositivos de visualización 110 de la Figura 1, encaminarse para su procesamiento adicional por el procesador o procesadores 102, o encaminarse para su procesamiento adicional por una de las entidades de procesamiento dentro del procesador paralelo 200 de la **Figura 2A.**

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

El funcionamiento de la agrupación de procesamiento 214 se puede controlar a través de un gestor de canalizaciones

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

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

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

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

Cada agrupación de procesamiento 214 puede incluir una MMU 245 (unidad de gestión de memoria) que está configurada para correlacionar direcciones virtuales en direcciones físicas. En otras realizaciones, una o más instancias de la MMU 245 pueden residir dentro de la interfaz de memoria 218 de la Figura 2. La MMU 245 incluye un conjunto de entradas de tabla de página (PTE) usadas para correlacionar una dirección virtual a una dirección física de un mosaico (más información sobre la aplicación de mosaico) y opcionalmente, una línea de índice de caché. La MMU 245 puede incluir memorias intermedias de traducción adelantada (TLB) de direcciones o cachés que pueden residir dentro del multiprocesador de gráficos 234 o la caché de L1 o la agrupación de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso de datos de superficie para permitir una intercalación de solicitud eficiente entre unidades de subdivisión. El índice de líneas de caché se puede usar para determinar si una solicitud de una línea de caché es un acierto o un desacierto.

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

como se describe en el presente documento (por ejemplo, las unidades de subdivisión 220A-220N de la Figura 2). La unidad de preROP 242 puede realizar optimizaciones para la mezcla de color, organizar datos de color de píxel y realizar traducciones de dirección.

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

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

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

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

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

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

La interconexión de memoria y de caché 268 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador de gráficos 324 al archivo de registro 258 y a la memoria compartida 270. En una realización, la interconexión de memoria y de caché 268 es una interconexión de barra transversal que permite que la unidad de carga/almacenamiento 266 implemente operaciones de carga y de almacenamiento entre la memoria compartida 270 y el archivo de registro 258. El archivo de registro 258 puede funcionar a la misma frecuencia que los núcleos de GPGPU 262, por lo tanto, la transferencia de datos entre los núcleos de GPGPU 262 y el archivo de registro

258 es de latencia muy baja. La memoria compartida 270 se puede usar para habilitar la comunicación entre hilos que se ejecutan en las unidades funcionales dentro del multiprocesador de gráficos 234. La memoria caché 272 se puede usar como una caché de datos, por ejemplo, para almacenar en caché datos de textura comunicados entre las unidades funcionales y la unidad de textura 236. La memoria compartida 270 se puede usar también como una caché gestionada por programa. Los hilos que se ejecutan en los núcleos de GPGPU 262 pueden almacenar, de manera programática, datos dentro de la memoria compartida además de los datos almacenados automáticamente en caché que se almacenan dentro de la memoria caché 272.

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

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

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

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

Los expertos en la materia entenderán que la arquitectura descrita en las Figuras 1, 2A-2D y **3A-3B** es descriptiva y no limitante en cuanto al alcance de las presentes realizaciones, que son ilustrativas. Por lo tanto, las técnicas descritas en el presente documento se pueden implementar en cualquier unidad de procesamiento configurada apropiadamente, incluyendo, sin limitación, uno o más procesadores de aplicación móvil, una o más unidades centrales de procesamiento (CPU) de sobremesa o de servidor, incluyendo CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tales como la unidad de procesamiento paralelo 202 de la Figura 2, así como uno o más procesadores de gráficos o unidades de procesamiento de propósito especial, sin apartarse del alcance de las realizaciones descritas en el presente documento.

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

Técnicas para interconexión de GPU a procesador de anfitrión

La **Figura 4A** ilustra una arquitectura ilustrativa en la que una pluralidad de GPU 410-413 están acopladas de manera

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

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

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

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

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

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

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

En una realización, un circuito intermediario 425 acopla de manera comunicativa el módulo de aceleración de gráficos 446 al bus de coherencia 464, permitiendo que el módulo de aceleración de gráficos 446 participe en el protocolo de coherencia de caché como un homólogo de los núcleos. En particular, una interfaz 435 proporciona conectividad al

circuito de intermediario 425 a través del enlace de alta velocidad 440 (por ejemplo, un bus PCIe, NVLink, etc.) y una interfaz 437 conecta el módulo de aceleración de gráficos 446 al enlace 440.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Durante el funcionamiento, una unidad de extracción de WD 491 en el corte de integración de acelerador 490 extrae el WD 484 siguiente que incluye una indicación del trabajo a hacer por uno de los motores de procesamiento de gráficos del módulo de aceleración de gráficos 446. Los datos del WD 484 pueden almacenarse en registros 445 y

usarse por la MMU 439, el circuito de gestión de interrupción 447 y/o el circuito de gestión de contexto 446 como se ilustra. Por ejemplo, una realización de la MMU 439 incluye circuitería de recorrido de segmentos/páginas para acceder a las tablas de segmentos/páginas 486 dentro del espacio de direcciones virtuales de SO 485. El circuito de gestión de interrupciones 447 puede procesar los eventos de interrupción 492 recibidos del módulo de aceleración de gráficos 446. Cuando se realizan operaciones de gráficos, una dirección eficaz 493 generada por un motor de procesamiento de gráficos 431-432, N es traducida a una dirección real por la MMU 439.

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

Tabla 1 - Registros de hipervisor inicializados

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

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

Tabla 2 - Registros inicializados por sistema operativo

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

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

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

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

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

En una realización, para el modelo compartido, se requiere que la aplicación 480 haga una llamada de sistema al sistema operativo 495 con un tipo del módulo de aceleración de gráficos 446, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de guardado/restablecimiento de contexto (CSRP). El tipo del módulo de aceleración de gráficos 446 describe la función de aceleración seleccionada como objetivo para la llamada de sistema. El tipo del módulo de aceleración de gráficos 446 puede ser un valor específico del sistema. Al WD se le da formato específicamente para el módulo de aceleración de gráficos 446, y puede estar en forma de un comando del módulo de aceleración de gráficos 446, un puntero de dirección eficaz a una estructura definida por el usuario, un puntero de dirección efectiva a una cola de comandos, o cualquier otra estructura de datos para describir el trabajo a hacer por el módulo de aceleración de gráficos 446. En una realización, el valor de AMR es el estado de AMR a usar para el proceso actual. El valor pasado al sistema operativo es similar a que una aplicación establezca el AMR. Si las implementaciones del circuito de integración de acelerador 436 y del módulo de aceleración de gráficos 446 no soportan un Registro de Anulación de Máscara de Autoridad de Usuario (UAMOR), el sistema operativo puede aplicar el valor de UAMOR actual al valor de AMR antes de pasar el AMR en la llamada de hipervisor. El hipervisor 496 puede aplicar opcionalmente el valor de registro de anulación de máscara de autoridad (AMOR) actual antes de colocar el AMR en el elemento de proceso 483. En una realización, el CSRP es uno de los registros 445 que contienen la dirección eficaz de un área en el espacio de direcciones de la aplicación 482 para que el módulo de aceleración de gráficos 446 grabe y restaure el estado de contexto. Este puntero es opcional si no se requiere que se guarde estado alguno entre trabajos o cuando se da prioridad a un trabajo. El área de guardado/restablecimiento de contexto puede ser una memoria de sistema anclada.

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

Tabla 3 - Parámetros de llamada de SO a hipervisor

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

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

Tabla 4 - Información de elemento de proceso

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

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

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

401-402, y viceversa, simplificando de ese modo la programabilidad. En una realización, una primera porción del espacio de direcciones virtual/eficaz está asignada a la memoria de procesador 401, una segunda porción a la segunda memoria de procesador 402, una tercera porción a la memoria de GPU 420, y así sucesivamente. El espacio de memoria virtual/eficaz total (denominado, en ocasiones, el espacio de direcciones eficaces) está distribuido, por lo tanto, a lo largo de cada una de las memorias de procesador 401-402 y las memorias de GPU 420-423, permitiendo que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual correlacionada con esa memoria.

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

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

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

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

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

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

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

Canalización de procesamiento de gráficos

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

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

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

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

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

En algunas realizaciones, la unidad de procesamiento de geometría 516 puede añadir o borrar elementos en el flujo de geometría. La unidad de procesamiento de geometría 516 emite los parámetros y vértices que especifican primitivas de gráficos nuevas al ensamblador de primitivas 518. El ensamblador de primitivas 518 recibe los parámetros y vértices desde la unidad de procesamiento de geometría 516 y construye primitivas de gráficos para su procesamiento por una

unidad de escala, selección y recorte de ventana gráfica 520. La unidad de procesamiento de geometría 516 lee datos que están almacenados en memoria de procesador paralelo o memoria de sistema para su uso en el procesamiento de los datos de geometría. La unidad de escala, selección y recorte de ventana gráfica 520 realiza el recorte, la selección y el ajuste a escala de ventana gráfica y emite primitivas de gráficos procesadas a un rasterizador 522.

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

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

Descripción general de aprendizaje automático

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

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

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

La precisión de un algoritmo de aprendizaje automático se puede ver afectada significativamente por la calidad del

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

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

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

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

Aceleración de aprendizaje automático de GPGPU

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

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

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

En una realización, cada agrupación de cómputo 706A-H incluye un conjunto de multiprocesadores de gráficos, tales

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

Múltiples instancias de la GPGPU 700 se pueden configurar para funcionar como una agrupación de cómputo. El mecanismo de comunicación usado por la agrupación de cómputo para la sincronización y el intercambio de datos varía a lo largo de las realizaciones. En una realización, las múltiples instancias de la GPGPU 700 se comunican a través de la interfaz de anfitrión 702. En una realización, la GPGPU 700 incluye un concentrador de E/S 708 que acopla la GPGPU 700 con un enlace de GPU 710 que habilita una conexión directa a otras instancias de la GPGPU. En una realización, el enlace de GPU 710 está acoplado a un puente de GPU a GPU dedicado que habilita la comunicación y la sincronización entre múltiples instancias de la GPGPU 700. En una realización, el enlace de GPU 710 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En una realización, las múltiples instancias de la GPGPU 700 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red que es accesible a través de la interfaz de anfitrión 702. En una realización, el enlace de GPU 710 se puede configurar para habilitar una conexión a un procesador de anfitrión además de o como una alternativa a la interfaz de anfitrión 702.

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

La **Figura 8** ilustra un sistema informático de múltiples GPU 800, de acuerdo con una realización ilustrativa. El sistema informático de múltiples GPU 800 puede incluir un procesador 802 acoplado a múltiples GPGPU 806A-D a través de un conmutador de interfaz de anfitrión 804. El conmutador de interfaz de anfitrión 804, en una realización, es un dispositivo conmutador PCI Express que acopla el procesador 802 a un bus PCI Express a través del 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 a través de un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 816. Los enlaces de GPU a GPU de alta velocidad se pueden conectar a cada una de las GPGPU 806A-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 de P2P 816 habilitan una comunicación directa entre cada una de las GPGPU 806A-D sin requerir una comunicación a través del bus de interfaz de anfitrión al que está conectado el procesador 802. Con el tráfico de GPU a GPU dirigido a los enlaces de GPU de P2P, el bus de interfaz de anfitrión permanece disponible para el acceso de memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 800, por ejemplo, a través de uno o más dispositivos de red. Aunque, en la realización ilustrada, las GPGPU 806A-D se conectan al procesador 802 a través del conmutador de interfaz de anfitrión 804, en una realización, el procesador 802 incluye un soporte directo para los enlaces de GPU de P2P 816 y se puede conectar directamente a las GPGPU 806A-D.

Implementaciones de red neuronal de aprendizaje automático

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

Un segundo tipo ilustrativo de red neuronal es la red neuronal convolucional (CNN). Una CNN es una red neuronal de realimentación prospectiva especializada para procesar datos que tienen una topología de tipo cuadrícula conocida, tales como datos de imagen. En consecuencia, las CNN se usan comúnmente para aplicaciones de reconocimiento de imágenes y de visión artificial, pero se pueden usar también para otros tipos de reconocimiento de patrones, tales como procesamiento de habla y de idioma. Los nodos en la capa de entrada de CNN están organizados en un conjunto de "filtros" (detectores de características inspirados por los campos receptivos encontrados en la retina), y la salida de cada conjunto de filtros se propaga a nodos en capas sucesivas de la red. Los cómputos para una CNN incluyen aplicar la operación matemática de convolución a cada filtro para producir la salida de ese filtro. La convolución es un

tipo especializado de operación matemática realizada por dos funciones para producir una tercera función que es una versión modificada de una de las dos funciones originales. En la terminología de redes convolucionales, la primera función para la convolución se puede denominar entrada, mientras que la segunda función se puede denominar núcleo de convolución. La salida se puede denominar correlación de características. Por ejemplo, la entrada a una capa de convolución puede ser una matriz multidimensional de datos que definen los diversos componentes de color de una imagen de entrada. El núcleo de convolución puede ser una matriz multidimensional de parámetros, en donde los parámetros están adaptados por el proceso de entrenamiento para la red neuronal.

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

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

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

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

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

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

Las capas convolucionales están conectadas de manera dispersa, lo que difiere de la configuración de red neuronal

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

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

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

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

La fase de agrupación 920 usa una función de agrupación que sustituye la salida de la capa convolucional 906 con una estadística de resumen de las salidas cercanas. La función de agrupación se puede usar para introducir la invarianza de traslación en la red neuronal, de manera que traslaciones pequeñas a la entrada no cambian las salidas agrupadas. La invarianza a la traslación local puede ser útil en escenarios donde la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Se pueden usar diversos tipos de funciones de agrupación durante la fase de agrupación 920, incluyendo agrupación máxima, agrupación promedio y agrupación de norma 12. Adicionalmente, algunas implementaciones de CNN no incluyen una fase de agrupación. En su lugar, tales implementaciones sustituyen una fase de convolución adicional que tiene un paso aumentado en relación con fases de convolución previas.

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

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

Además de las redes CNN y RNN básicas descritas, se pueden habilitar variaciones a esas redes. Una variante de RNN ilustrativa es la RNN de memoria a corto plazo larga (LSTM). Las RNN de LSTM son capaces de aprender dependencias a largo plazo que pueden ser necesarias para procesar secuencias de idioma más largas. Una variante de la CNN es una red de creencia profunda convolucional, que tiene una estructura similar a una CNN y se entrena de una manera similar a una red de creencia profunda. Una red de creencia profunda (DBN) es una red neuronal generativa que está compuesta por múltiples capas de variables estocásticas (aleatorias). Las DBN se pueden entrenar capa a capa usando aprendizaje no supervisado voraz. Los pesos aprendidos de la DBN pueden usase entonces para proporcionar redes neuronales de preentrenamiento determinando un conjunto inicial óptimo de pesos para la red neuronal.

La **Figura 11** ilustra el entrenamiento y despliegue ilustrativos de una red neuronal profunda. Una vez que se ha estructurado una red dada para una tarea, la red neuronal se entrena usando un conjunto de datos de entrenamiento 1102. Se han desarrollado diversas estructuras de entrenamiento 1104 para habilitar la aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura de aprendizaje automático 604 de la **Figura 6** se puede configurar como una estructura de entrenamiento 604. La estructura de entrenamiento 604 se puede enganchar a una red neuronal no entrenada 1106 y habilitar que la red neuronal no entrenada se entrene usando los recursos de procesamiento paralelo descritos en el presente documento para generar una red neuronal entrenada 1108.

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

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

El aprendizaje no supervisado es un método automático en el que la red intenta entrenarse a sí misma usando datos no etiquetados. Por lo tanto, para un aprendizaje no supervisado, el conjunto de datos de entrenamiento 1102 incluirá datos de entrada sin dato de salida asociado alguno. La red neuronal no entrenada 1106 puede aprender agrupamientos dentro de la entrada no etiquetada y puede determinar cómo las entradas individuales están relacionadas con el conjunto de datos global. El entrenamiento no supervisado se puede usar para generar una correlación de autoorganización, que es un tipo de red neuronal entrenada 1107 que puede realizar operaciones útiles en cuanto a la reducción de la dimensionalidad de los datos. El entrenamiento no supervisado se puede usar también para realizar una detección de anomalías, lo que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

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

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

La **Figura 12** es un diagrama de bloques ilustrativo que ilustra un aprendizaje distribuido. El aprendizaje distribuido es un modelo de entrenamiento que usa múltiples nodos informáticos distribuidos para realizar un entrenamiento supervisado o no supervisado de una red neuronal. Cada uno de los nodos computacionales distribuidos puede incluir uno o más procesadores de anfitrión y uno o más de los nodos de procesamiento de propósito general, tales como la unidad de procesamiento de gráficos de propósito general altamente paralela 700, como en la **Figura 7.** Como se ilustra, un aprendizaje distribuido se puede realizar con el paralelismo de modelo 1202, el paralelismo de datos 1204 o una combinación del paralelismo de modelo y de datos 1204.

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

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

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

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

Aplicaciones de aprendizaje automático ilustrativas

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

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

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

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

Las plataformas de procesamiento paralelo usadas para el aprendizaje automático se pueden dividir en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento son, en general, altamente paralelas

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

La **Figura 13** ilustra un sistema en un chip (SOC) de inferenciación 1300 ilustrativo adecuado para realizar una inferenciación usando un modelo entrenado. El SOC 1300 puede integrar componentes de procesamiento que incluyen un procesador de medios 1302, un procesador de visión 1304, una GPGPU 1306 y un procesador de múltiples núcleos 1308. El SOC 1300 puede incluir adicionalmente una memoria en chip 1305 que puede habilitar una agrupación de datos en chip compartida que es accesible por cada uno de los componentes de procesamiento. Los componentes de procesamiento se pueden optimizar para un funcionamiento de baja potencia para habilitar el despliegue en una diversidad de plataformas de aprendizaje automático, incluyendo vehículos autónomos y robots autónomos. Por ejemplo, una implementación del SOC 1300 se puede usar como una porción del sistema de control principal para un vehículo autónomo. Donde el SOC 1300 está configurado para su uso en vehículos autónomos, el SOC está diseñado y configurado para cumplir con las normas de seguridad funcional relevantes de la jurisdicción de despliegue.

Durante el funcionamiento, el procesador de medios 1302 y el procesador de visión 1304 pueden trabajar conjuntamente para acelerar operaciones de visión por ordenador. El procesador de medios 1302 puede habilitar la descodificación de latencia baja de múltiples flujos de vídeo de alta resolución (por ejemplo, 4K, 8K). Los flujos de vídeo descodificados se pueden escribir en una memoria intermedia en la memoria en chip 1305. El procesador de visión 1304 puede analizar entonces el vídeo descodificado y realizar operaciones de procesamiento preliminares sobre las tramas del vídeo descodificado como preparación al procesamiento de las tramas usando un modelo de reconocimiento de imágenes entrenado. Por ejemplo, el procesador de visión 1304 puede acelerar operaciones de convolución para una CNN que se usa para realizar un reconocimiento de imágenes sobre los datos de vídeo de alta resolución, mientras que los cómputos de modelo de extremo trasero son realizados por la GPGPU 1306.

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

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

La **Figura 14** ilustra un sistema de captura de imágenes ilustrativo para un dispositivo de captura de imágenes. En algunas realizaciones, el dispositivo de captura de imágenes es un dispositivo independiente, tal como una cámara digital. En otras realizaciones, el dispositivo de captura de imágenes es parte o un subcomponente de otro dispositivo informático que requiere capacidades de captura de imágenes, tal como un dispositivo informático portátil o de mano con una cámara digital para capturar imágenes. En realizaciones ilustrativas, un dispositivo de captura de imágenes está integrado con un procesamiento de computación de red neuronal convolucional (CNN), tal como una red neuronal de aprendizaje profundo (DNN) que puede proporcionar datos de DNN en tiempo real o simultáneamente con o en el momento de la captura y almacenamiento de imágenes por el dispositivo de captura de imágenes. Por lo tanto, el dispositivo de captura de imágenes desvelado en el presente documento puede proporcionar tanto información de imagen como de color, así como información de aprendizaje automático profundo, tal como la detección de características usando una CNN.

La **Figura 14** ilustra un diagrama ilustrativo del almacenamiento de datos de canal RGB y CNN como parte de una imagen de canal profundo. El sistema ilustrativo se muestra con una cámara de canal profundo 1400 y una memoria de canal profundo 1420. La cámara de canal profundo 1400 incluye un sensor de color 1402 y una unidad FPGA 1404. El sensor de color 1402 incluye una pluralidad de sensores dispuestos en una matriz para capturar datos de color para una imagen 1403A. La imagen 1403A puede ser una imagen RGB o una imagen YUV. El sensor de color 1402 emite valores de imagen o datos a la memoria de canal profundo 1420 y a la unidad FPGA 1404.

El sensor de color puede ser cualquier tipo de fotosensor, fotodetector, sensor CMOS o matriz de píxeles que tiene

una matriz de sensores de luz para detectar la energía luminosa que forma imágenes en color capturadas por la matriz de sensores de color. Como alternativa, el sensor de color se puede configurar para detectar imágenes sin color. Los datos o valores para cada componente de color forman canales de salida separados, tales como los canales de color RGB 1414.

En una realización, las salidas de la matriz de sensores de color 1402 son entradas a una red neuronal 1406 programada en FPGA 1404. La red puede ser cualquier tipo de red con cualquier número de capas convolucionales y nodos en cada capa que puedan realizar procesamiento de redes neuronales como se desvela en realizaciones ilustrativas con respecto a las **Figuras 9A** y **9B.** En otros ejemplos, puede usarse circuitería de cableado permanente o matrices lógicas programables (PLA) en lugar de FPGA 1406 con la red neuronal programada o cableada permanentemente en la respectiva circuitería o matrices de la misma manera. En otras realizaciones, la FPGA 1404 se puede programar con aceleración de hardware tal como se describe en la **Figura 6** con respecto a la aplicación de aprendizaje automático 602, la estructura de aprendizaje automático 604 y la estructura de trabajo informático 606.

En una realización, la unidad FPGA 1404 está programada con una CNN muy profunda convencional, tal como la CNN del Grupo de Geometría Visual (VGG) etiquetada como red VGG convencional 1406. En otras realizaciones, se pueden programar otros tipos de CNN, tal como una DNN, en la unidad FPGA 1404 como se desvela en el presente documento. En este ejemplo, la red VGG 1406 está programada con cinco (5) capas de convolución etiquetadas como "Conv1" a "Conv5". Cada capa de convolución Conv2 a Conv5 incluye un "grupo" de datos que proporciona una estadística de resumen de las salidas cercanas. En otras realizaciones, se puede utilizar cualquier número de capas convolucionales, incluyendo una sola capa de convolución. Cada capa de Conv1 a Conv5 realiza operaciones de convolución que proporcionan imágenes filtradas o datos de mapas de características etiquetados como "Datos de Conv1" a "Datos de Conv5". En esta red CNN de avance, la salida de cada capa convolucional pasa a la siguiente capa hasta una capa de predicción final 1425. En la capa de predicción 1425, se puede realizar una clasificación 1430 tal como determinar un tipo de imagen basándose en el procesamiento de CNN y los datos de CNN.

La memoria de canal profundo 1420 almacena una imagen 1403B para canales RGB 1414 derivados de la imagen 1403A. Los datos de Conv1 (1410A) a Conv5 (1410E) se almacenan para los canales CNN 1418. En esta realización, cada canal de convolución Conv1 a Conv5 representa un canal de CNN que tiene cinco canales para los canales de CNN 1418. Los datos de Conv1 (1410A) a Conv5 (1410E) pueden ser diferentes imágenes filtradas o mapas de características de la imagen 1403A dependiendo del procesamiento convolucional realizado en cada capa. En la memoria de canal profunda 1420, las imágenes para los canales de color RGB 1414 y los canales de CNN se almacenan en una memoria adyacente o se concatenan entre sí.

Canalización de ejecución de DNN en plataforma multiprocesador

Como se describe en el contexto de las **Figuras 12 y 14,** una red de aprendizaje profundo tiene una pluralidad de nodos. La **Figura 15** es un diagrama de bloques de un ejemplo de una red neuronal de aprendizaje profundo 1502 con seis nodos etiquetados del 1 al 6. Si bien se muestran seis nodos, esto es con fines ilustrativos. Puede haber muchos más nodos, dependiendo de la implementación particular. Las conexiones entre los nodos también pueden ser más complejas que las que se muestran en el ejemplo. En muchas redes de aprendizaje profundo, la red original es un grafo acíclico. En este ejemplo, el flujo de datos fluye en solo una dirección, que se muestra de izquierda a derecha.

Como se describe en el presente documento, una estructura de canalización puede ejecutar una red de aprendizaje profundo en una plataforma de múltiples núcleos de una manera más eficiente que con los sistemas de planificación automática de tareas. En el sistema de hilos o procesadores múltiples descrito, la eficacia se mejora equilibrando la carga de trabajo entre los recursos de procesamiento. Esto evita que la acumulación, la latencia o el retardo en un recurso de procesamiento afecten el flujo de trabajo general. Como se describe en el presente documento, el flujo de trabajo se distribuye usando la distribución de computación en los nodos de red. La estructura de canalización se mejora mediante dos componentes. El primero es un analizador de carga de trabajo de red 1708 y el segundo es un ejecutor de red 1712 como se muestra en la **Figura 17**.

La **Figura 16** es un diagrama de agrupación de nodos de red por un analizador de red. Los mismos seis nodos de red 1-6 se agrupan en tres grupos, etiquetados Grupo 1, Grupo 2, Grupo 3. Estos grupos corresponden a recursos de procesamiento como se describe a continuación. Supongamos que la plataforma de destino tiene N núcleos, entonces, dada una red de aprendizaje profundo, el analizador de carga de trabajo de red analiza la distribución de cómputo y agrupa apropiadamente los nodos de red en N grupos. La agrupación se realiza de modo que cada grupo tenga casi la misma cantidad de cómputo.

De acuerdo con una realización de la invención, la cantidad de cómputo para una red conocida se puede determinar con precisión usando el tamaño de sus datos de entrada. Cada nodo de red de aprendizaje profundo ejecuta una operación de aprendizaje profundo conocida, tal como la convolución, normalización por lotes, softmax, etc., por lo que la cantidad de cómputo de cada nodo se puede calcular con precisión basándose en el tamaño de entrada del primer nodo. Al mismo tiempo, una red de aprendizaje profundo tiene una característica noble que una vez que se determina el tamaño de entrada de la red, se puede determinar el tamaño de entrada de todos los nodos de red porque

todas son operaciones convencionales cuyo tamaño de entrada y tamaño de salida se determinan por fórmulas convencionales.

En consecuencia, para cualquier red particular y tarea particular o tipo de carga de trabajo se puede determinar la complejidad cómputo de cada nodo. Por ejemplo, la complejidad de cálculo del primer nodo puede ser 5, del segundo nodo 3 y del tercer nodo 2, esto es lo que se denomina en el presente documento distribución de cálculo. En este caso sencillo de tres nodos, puede haber dos grupos, el grupo 1 que tiene solo el primer nodo, el grupo 2 que tiene el segundo y el tercer nodo. Dado que las complejidades de computación del segundo y tercer nodo son 3 y 2, respectivamente, el total para el grupo 2 es 3 + 2 = 5. Esto es lo mismo que para el grupo 1, por lo que la estrategia de grupo del analizador de red sigue las instrucciones de la medición de distribución de cálculo.

En el ejemplo ilustrado, hay tres grupos, uno para cada uno de los tres núcleos de procesamiento donde N=3. El primer grupo tiene los nodos de red 1 y 2, el segundo grupo tiene los nodos de red 3 y 4 y el tercer grupo tiene los nodos de red 5 y 6. Dado que la red original es un grafo acíclico y el flujo de datos sigue solo una dirección, el flujo de datos atraviesa en la misma dirección dentro de cada grupo y entre grupos.

En este ejemplo, el analizador de red distribuye uniformemente los nodos de red entre los grupos para que haya el mismo número de nodos en cada grupo. En muchos casos, esta puede ser la mejor o casi la mejor solución. En otros casos, puede haber algunos nodos que requieran más cómputo que otros nodos. Para una red de este tipo, puede haber menos nodos en algunos grupos que en otros. La agrupación se realiza, no basándose en el número de nodos, sino basándose en la distribución del cálculo.

De acuerdo con una realización de la invención, los nodos se distribuyen a cada grupo secuencialmente. Como se muestra, la entrada para el nodo 1 va al primer grupo. La salida del nodo 1 va al nodo 2 y a continuación al segundo grupo. La agrupación establece una dirección de flujo de procesamiento no solo del nodo 1 al nodo 6 en secuencia, sino también del grupo 1 al grupo 3 en secuencia. Esto proporciona un flujo de datos más fluido a través de la red.

El analizador de carga de trabajo de red 1708 puede ser un componente, hilo, módulo o rutina en el procesador de múltiples núcleos que analiza una carga de trabajo entrante en tiempo real a medida que se recibe la carga de trabajo. Esto permite modificar la agrupación para adaptarse a diferentes cargas de trabajo. En algunas aplicaciones, la carga de trabajo es una secuencia de imágenes de vídeo. Según realizaciones de la invención, las imágenes de la secuencia están normalizadas para tener un tamaño fijo. El tamaño fijo de las imágenes puede ser el tamaño capturado o un tamaño reducido o con reducción de escala. Dado que cada imagen tiene el mismo tamaño, la distribución de cálculo para cada imagen será aproximadamente la misma. En un escenario de este tipo, las agrupaciones pueden hacerse una vez y a continuación fijarse hasta que se reciba una imagen de tamaño diferente o un tipo de imagen diferente como la carga de trabajo. La distribución uniforme de los cálculos reduce o evita los vacíos en el flujo de trabajo.

La **Figura 17** es un diagrama de un sistema de procesamiento de múltiples núcleos con un analizador de red 1708 y un ejecutor de red 1712 de acuerdo con una realización de la invención. Después de agrupar los nodos, a continuación, como se muestra, el ejecutor de la red asigna cada grupo a un procesador respectivo que maneja la tarea de cálculo para ese grupo especialmente. El procesador puede ser un núcleo de un chip de múltiples núcleos. Los procesadores también pueden incluir hilos de un sistema de múltiples hilos. Los procesadores también pueden ser núcleos de procesamiento en más de un chip. Con la asignación de grupos a los procesadores, para una secuencia de imágenes consecutiva, los N procesadores forman una canalización. Si un solo núcleo debe pasar un tiempo t para una imagen, entonces la canalización con N núcleos solo requiere un tiempo t/N.

Como se muestra, el procesador de múltiples núcleos 1712 tiene tres núcleos de procesamiento. A cada núcleo se le asigna uno de los tres grupos. Cada núcleo también está acoplado a una memoria 1714. La carga de trabajo se aplica a la memoria o a través de una memoria intermedia de entrada 1704 tal como se muestra. La memoria intermedia de entrada puede estar en una porción asignada de la memoria 1714. La entrada también se aplica al analizador de carga de trabajo de red 1708 para su uso en la determinación de la distribución de cálculo. El analizador de red distribuye los nodos de red a los diversos núcleos como se muestra, por ejemplo, en la figura.

Después de que el analizador de carga de trabajo de red ha agrupado los nodos, comienza una fase de configuración de ejecución de red. El ejecutor de red asigna cada grupo a un procesador o núcleo específico y asigna un bloque de memoria de salida, identificado como P1, P2 y P3 en este ejemplo, para cada procesador. A continuación, el siguiente procesador o núcleo accede a esta memoria de salida para realizar la siguiente etapa en la secuencia de etapas. El ejecutor de la red también puede asignar un bloque de memoria 1704 para la carga de trabajo de entrada. Esto puede ser una salida de una aplicación 1702 u otra fuente. A continuación, se ordena o configura la aplicación u otra fuente para que escriba nuevos datos en la memoria de entrada asignada.

En este ejemplo, el ejecutor también ordena a cada uno de los tres procesadores, 1, 2, 3, cuyo espacio de memoria asignado, P1, P2, P3, debe usarse para leer y escribir las entradas y salidas respectivas. En este ejemplo, el procesador 1 que ejecuta los nodos del grupo 1, lee su entrada de la memoria intermedia de la memoria de entrada 1704 y escribe su salida en la porción configurada de la memoria de salida P1. El procesador 2 que ejecuta los nodos del grupo 2, lee su entrada de P1 y escribe su salida en la memoria de salida P2. El procesador 3 lee su entrada de

P2 y escribe su salida en la memoria de salida P3. En cada caso de un grafo secuencial para la DNN, la salida de un procesador o grupo sirve como entrada para el siguiente. El resultado final de la red se escribe en P3 por el procesador 3. La asignación de memoria para P3 puede ser igual o diferente de una memoria intermedia de salida 1706. La memoria intermedia de salida proporciona los resultados a otros procesos aguas abajo.

Después de la configuración por el ejecutor de la red, el ejecutor inicia la ejecución. Una aplicación 1702, u otra fuente, genera continuamente entrada en la memoria intermedia de entrada o la memoria 1704 y toda la canalización sigue funcionando y escribe continuamente la salida correspondiente en P3 y en la memoria intermedia de salida 1706 para cada parche de datos de entrada.

Cada núcleo de procesamiento realiza el cálculo para sus nodos de red asignados y a continuación almacena los resultados en la memoria. El segundo y tercer núcleos de procesamiento obtienen sus entradas de la memoria y almacenan resultados y salidas intermedios en la memoria. Los resultados finales se proporcionan como una salida del procesador de múltiples núcleos. El procesador de múltiples núcleos puede tener muchos más núcleos que están asignados a diferentes tareas o que están asignados a más nodos de red neuronal profunda que los tres nodos mostrados en las figuras.

La **Figura 18** es un diagrama de flujo de proceso de las operaciones realizadas por el analizador y el ejecutor como se han descrito anteriormente. En 1822, se recibe una carga de trabajo para que una estructura de canalización se ejecute en una red de aprendizaje profundo. La estructura de canalización tiene múltiples recursos de procesamiento, por ejemplo, en una plataforma de múltiples núcleos. La red de aprendizaje profundo tiene múltiples nodos.

En 1824, un analizador de carga de trabajo de red analiza la distribución de cálculo de la carga de trabajo a través de los nodos de red. En 1826, el analizador de carga de trabajo de red agrupa los nodos de red en grupos. Esta agrupación se basa en la distribución de cálculo para distribuir la carga de trabajo de manera uniforme a través de los grupos. En 1828, un ejecutor de red, u otro componente, asigna cada grupo a un núcleo de procesamiento u otro tipo de recurso de la plataforma de múltiples núcleos.

En 1830, después de configurar los nodos de red, el ejecutor de la red ejecuta la carga de trabajo. Cada núcleo de procesamiento respectivo maneja las tareas de cálculo de la carga de trabajo recibida para el grupo respectivo de nodos de red. En 1832, la salida puede aplicarse a continuación a otra tarea, tal como tecnologías de detección, reconocimiento y seguimiento, reconocimiento automático de voz, autenticación de usuarios, comprensión de imágenes y visión artificial, etc.

Descripción general del sistema de gráficos

La **Figura 19** es un diagrama de bloques de un sistema de procesamiento 1900 de acuerdo con una realización ilustrativa. En diversas realizaciones, el sistema 1900 incluye uno o más procesadores 1902 y uno o más procesadores de gráficos 1908, y puede ser un único sistema de escritorio de procesador, un sistema de estación de trabajo multiprocesador o un sistema de servidor que tiene un gran número de procesadores 1902 o núcleos de procesador 107. 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 incorporarse dentro de, una plataforma de juegos basada en servidor, una consola de juegos, que incluye una consola de juegos y de medios, 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 de 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 de descodificador de salón que tiene uno o más procesadores 1902 y una interfaz gráfica generada por uno o más procesadores de gráficos 1908.

En algunas realizaciones, cada uno de los uno o más procesadores 1902 incluye uno o más núcleos de procesador 1907 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para software de usuario y sistema. En algunas realizaciones, cada uno de los uno o más núcleos de procesador 1907 está configurado para procesar un conjunto de instrucciones 1909 específico. En algunas realizaciones, el conjunto de instrucciones 1909 puede facilitar el cómputo de conjunto de instrucciones complejo (CISC), el cómputo de conjunto de instrucciones reducido (RISC) o el cómputo a través de una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 1907 pueden procesar, cada uno, un conjunto de instrucciones 1909 diferente, 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, tales como un procesador de señales digitales (DSP).

En algunas realizaciones, el procesador 1902 incluye la memoria caché 1904. Dependiendo de la arquitectura, el procesador 1902 puede tener una única caché interna o múltiples niveles de caché interna. En algunas realizaciones, la memoria caché se comparte entre diversos componentes del procesador 1902. En algunas realizaciones, el

procesador 1902 también usa una caché externa (por ejemplo, una caché de nivel 3 (L3) o una caché de último nivel (LLC)) (no mostrada), que se puede compartir entre los núcleos de procesador 1907 usando técnicas de coherencia de caché conocidas. Se incluye adicionalmente, en el procesador 1902, un archivo de registro 1906 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de coma flotante, registros de estado y un registro de puntero de instrucción). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos del diseño del procesador 1902.

En algunas realizaciones, el procesador 1902 está acoplado con un bus de procesador 1910 para transmitir señales de comunicación tales como señales de dirección, de datos o de control entre el procesador 1902 y otros componentes en el sistema 1900. En una realización, el sistema 100 usa una arquitectura de sistema de 'concentrador' ilustrativa, que incluye un concentrador de controlador de memoria 1916 y un concentrador de controlador de Entrada 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 mediante un bus de E/S local. En una realización, la lógica del concentrador de controlador de memoria 1916 está integrada dentro del procesador.

El dispositivo de memoria 1920 puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria de acceso aleatorio estática (SRAM), un dispositivo de memoria flash, dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para servir como una memoria de proceso. En una realización, el dispositivo de memoria 1920 puede operar como memoria de 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 proceso. El concentrador de 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 de gráficos y medios.

En algunas realizaciones, el ICH 1930 posibilita que los periféricos se conecten al dispositivo de memoria 1920 y al procesador 1902 mediante un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, pero sin limitación, 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, una unidad de disco duro, memoria flash, etc.), y un controlador de E/S heredado 1940 para acoplar dispositivos heredados (por ejemplo, de tipo sistema personal 2 (PS/2)) al sistema. Uno o más controladores de Bus Serie Universal (USB) 1942 conectan los dispositivos de entrada, tales como las combinaciones de teclado y ratón 1944. Un controlador de red 1934 puede acoplarse también con el ICH 1930. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus del procesador 1910. Se apreciará que el sistema 1900 mostrado es ilustrativo y no limitante, ya que pueden usarse también otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por ejemplo, el concentrador del 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, tal como el procesador de gráficos externo 1912.

La **Figura 20** es un diagrama de bloques de una realización ilustrativa 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 de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa. El procesador 2000 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 2002N representado por los recuadros con línea discontinua. 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 en 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 caché de instrucciones y de datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel medio compartida, tal como una caché de Nivel 2 (L2), de Nivel 3 (L3), de Nivel 4 (L4) o de otros niveles, en donde el nivel más alto de caché antes de la memoria externa se clasifica como LLC. En algunas realizaciones, la lógica de coherencia de caché mantiene la coherencia entre las diversas unidades de caché 2006 y 2004A-2004N.

En algunas realizaciones, el procesador 2000 también puede incluir un conjunto de una o más unidades de controlador de bus 216 y un núcleo de agente de sistema 2010. Las una o más unidades controladoras de bus 216 gestionan un conjunto de buses de periféricos, tales como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 2010 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunas realizaciones, el núcleo de agente de sistema 2010 incluye uno o más controladores de memoria integrados 2014 para gestionar el acceso a diversos dispositivos de memoria externos (no mostrados).

En algunas realizaciones, uno o más de los núcleos de procesador 2002A-2002N incluyen soporte para múltiples hilos simultáneos. En tal realización, el núcleo de agente de sistema 210 incluye componentes para coordinar y hacer

funcionar los núcleos 2002A-2002N durante un procesamiento de múltiples hilos. El núcleo de agente de sistema 210 puede incluir adicionalmente una unidad de control de potencia (PCU), que incluye lógica y componentes para regular el estado de potencia de los núcleos de procesador 2002A-2002N y el procesador de gráficos 2008.

En algunas realizaciones, el procesador 2000 incluye adicionalmente 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 caché compartidas 2006 y el núcleo de agente de sistema 2010, incluyendo los 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 controlar una salida de 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 se puede integrar dentro del procesador de gráficos 2008 o el núcleo de agente de sistema 2010.

En algunas realizaciones, se usa una unidad de interconexión basada en anillo 2012 para acoplar los componentes internos del procesador 2000. Sin embargo, se puede usar una unidad de interconexión alternativa, tal como una interconexión de punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador de gráficos 208 se acopla con la interconexión en anillo 2012 mediante un enlace de E/S 2013.

El enlace de E/S ilustrativo 2013 representa al menos una de múltiples variedades de interconexiones de E/S, que incluyen una interconexión de E/S en el paquete que facilita la comunicación entre diversos componentes de procesador y un módulo de memoria integrado de alto rendimiento 218, tal como un módulo eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 202A-202N y del procesador de gráficos 208 usan módulos de memoria integrados 218 como una caché de último nivel compartida.

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), en 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, en donde uno o más núcleos que tienen un consumo de energía relativamente superior se acoplan con uno o más núcleos de potencia que tienen un consumo de energía inferior. Adicionalmente, el procesador 200 se puede implementar en uno o más chips o como un circuito integrado de SoC que tiene los componentes ilustrados, además de otros componentes.

La **Figura 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 correlacionada con memoria, con registros en el procesador de gráficos y con comandos colocados en la memoria de procesador. En algunas realizaciones, el procesador de gráficos 300 incluye una interfaz de memoria 2114 para acceder a memoria. La interfaz de memoria 314 puede ser una interfaz a memoria local, una o más cachés internas, una o más cachés externas compartidas y/o a memoria de sistema.

En algunas realizaciones, el procesador de gráficos 2100 también incluye un controlador de visualización 2102 para controlar unos datos de salida de visualización a un dispositivo de visualización 2120. El controlador de visualización 2102 incluye hardware para uno o más planos de superposición para la visualización y composición de múltiples capas de elementos de interfaz de usuario o de vídeo. En algunas realizaciones, el procesador de gráficos 2100 incluye un motor de códec de vídeo 306 para codificar, descodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluyendo, pero sin limitación, formatos del Grupo de Expertos en Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como de la Sociedad de Ingenieros de Imágenes en Movimiento y de Televisión (SMPTE) 421M/VC-1 y formatos del Grupo Conjunto de Expertos en Fotografía (JPEG) tales como los formatos JPEG y Motion JPEG (MJPEG).

En algunas realizaciones, el procesador de gráficos 2100 incluye un motor de transferencia de imágenes en bloque (BLIT) 2104 para realizar operaciones de rasterizador bidimensionales (2D), incluyendo, por ejemplo, transferencias de bloque de frontera de bits. Sin embargo, en una realización, se realizan operaciones de gráficos 2D usando uno o más componentes del motor de procesamiento de gráficos (GPE) 310. En algunas realizaciones, el GPE 2110 es un motor de cómputo para realizar operaciones de gráficos, incluyendo operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 2110 incluye una canalización de 3D 2112 para realizar operaciones 3D, tales como representar imágenes y escenas tridimensionales usando funciones de procesamiento que actúan sobre formas de primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización de 3D 2112 incluye elementos de función programable y fija que realizan diversas tareas dentro del elemento y/o generan hilos de ejecución en un subsistema de 3D/de medios 315. Aunque la canalización de 3D 2112 se puede usar para realizar operaciones de medios, una realización del GPE 310 también incluye una canalización de medios 2116 que se usa específicamente para realizar

operaciones de medios, tales como post-procesamiento de vídeo y potenciación de imagen.

En algunas realizaciones, la canalización de medios 2116 incluye unidades de lógica programable o de función fija para realizar una o más operaciones de medios especializadas, tales como aceleración de descodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar o en nombre del motor de códec de vídeo 2106. En algunas realizaciones, la canalización de medios 2116 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema de 3D/de medios 2115. Los hilos generados realizan cómputos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema de 3D/de medios 2115.

En algunas realizaciones, el subsistema de 3D/de medios 2115 incluye lógica para ejecutar hilos generados por la canalización de 3D 2112 y la canalización de medios 2116. En una realización, las canalizaciones envían solicitudes de ejecución de hilos al subsistema de 3D/de medios 2115, incluyendo lógica de despacho de hilos para arbitrar y despachar las diversas solicitudes a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos de 3D y de medios. En algunas realizaciones, el subsistema de 3D/de medios 2115 incluye una o más cachés internas para datos e instrucciones de hilo. En algunas realizaciones, el subsistema también incluye memoria compartida, incluyendo registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Motor de procesamiento de gráficos

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 2210 mostrado en la **Figura 21.** Los elementos de la **Figura 22** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa. Por ejemplo, se ilustra la canalización 3D 2212 y la canalización de medios 2216 de la **Figura 3**. La canalización de medios 2216 es opcional en algunas realizaciones del GPE 2210 y puede no incluirse explícitamente dentro del GPE 410. Por ejemplo, y en al menos una realización, un procesador de medios y/o de imagen separado está acoplado al GPE 2210.

En algunas realizaciones, el GPE 2210 se acopla con o incluye un emisor de envío por flujo continuo de comando 2203, que proporciona un flujo de comandos a la canalización 3D 2112 y/o a las canalizaciones de medios 2116. En algunas realizaciones, el emisor de flujo continuo de comando 2203 está acoplado con memoria, que puede ser memoria de sistema, o una o más de memoria caché interna y memoria caché compartida. En algunas realizaciones, el emisor de flujo continuo de comando 2203 recibe comandos desde la memoria y envía los comandos a la canalización 3D 2112 y/o a la canalización de medios 2116. Los comandos son directivas extraídas desde 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 en anillo puede incluir adicionalmente memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización de 3D 2112 también pueden incluir referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértice y de geometría para la canalización de 3D 2112 y/o datos de imagen y objetos de memoria para la canalización de medios 2116. La canalización de 3D 2112 y la canalización de medios 2116 procesan los comandos y datos realizando operaciones a través de lógica dentro de las canalizaciones respectivas o despachando uno o más hilos de ejecución a una matriz de núcleo de gráficos 2214.

En diversas realizaciones, la canalización de 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 cálculo u otros programas de sombreado, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleo de gráficos 2214. La matriz de núcleo de gráficos 2214 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución de múltiples propósitos (por ejemplo, unidades de ejecución) dentro de la matriz de núcleo de gráficos 2214 incluye soporte para diversos lenguajes de sombreador de API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunas realizaciones, la matriz de núcleo de gráficos 2214 también incluye la lógica de ejecución para realizar funciones de medios, tales como el procesamiento de vídeo y/o de imagen. En una realización, las unidades de ejecución incluyen adicionalmente lógica de propósito general que es programable para realizar operaciones computacionales de propósito general paralelas, además de operaciones de procesamiento de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o junto con lógica de propósito general dentro del núcleo o núcleos de procesador 1907 de la **Figura 19** o del núcleo 2002A-2002N, como en la **Figura 20**.

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

función compartida 2220.

En algunas realizaciones, la matriz de núcleo de gráficos 2214 es escalable, de manera que la matriz incluye un número variable de núcleos de gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y en el nivel de rendimiento del GPE 2210. En una realización, los recursos de ejecución son dinámicamente escalables, de manera que los recursos de ejecución pueden activarse o desactivarse según sea necesario.

La matriz de núcleo de 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 núcleo de gráficos. Las funciones compartidas dentro de la lógica de función compartida 2220 son unidades de lógica de hardware que proporcionan funcionalidad complementaria especializada a la matriz de núcleo de 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 cachés 2225 dentro de la lógica de funciones compartidas 2220. Se implementa una función compartida donde la demanda de una función especializada dada es insuficiente para su inclusión dentro de la matriz de núcleo de gráficos 2214. En su lugar, una única instanciación de esa función especializada se implementa como una entidad autónoma en la lógica de funciones compartidas 2220 y se comparte entre los recursos de ejecución dentro de la matriz de núcleo de gráficos 2214. El conjunto preciso de funciones que se comparten entre la matriz de núcleo de gráficos 2214 y se incluyen dentro de la matriz de núcleo de gráficos 2214 varía entre realizaciones.

La **Figura 23** es un diagrama de bloques de otra realización ilustrativa de un procesador de gráficos 500. Los elementos de la **Figura 23** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

En algunas realizaciones, el procesador de gráficos 2300 incluye una interconexión en anillo 2302, un extremo frontal de canalización 2304, un motor de medios 2337 y unos núcleos de gráficos 2380A-2380N. En algunas realizaciones, la interconexión en anillo 2302 acopla el procesador de gráficos a otras unidades de procesamiento, incluyendo otros procesadores de gráficos o uno o más núcleos de procesador de propósito general. En algunas realizaciones, el procesador de gráficos es uno de muchos procesadores integrados dentro de un sistema de procesamiento de múltiples núcleos.

En algunas realizaciones, el procesador de gráficos 2300 recibe lotes de comandos a través de la interconexión en anillo 2302. Los comandos entrantes son interpretados por un transmisor por flujo continuo de comandos 2303 en el extremo frontal de canalización 2304. En algunas realizaciones, el procesador de gráficos 2300 incluye lógica de ejecución ajustable a escala para realizar un procesamiento de geometría 3D y un procesamiento de medios a través del núcleo o núcleos de gráficos 2380A-2380N. Para los comandos de procesamiento de geometría 3D, el transmisor por flujo continuo de comandos 2303 suministra comandos a la canalización de geometría 2336. Para al menos algunos comandos de procesamiento de medios, el transmisor por flujo continuo 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 post procesamiento de vídeo y de imagen y un motor de codificación/decodificación multiformato (MFX) 2333 para proporcionar la codificación y decodificación de datos de medios acelerados por hardware. En algunas realizaciones, la canalización de geometría 2336 y el motor de medios 2337 cada uno genera hilos de ejecución para los recursos de ejecución de hilo proporcionados por al menos un núcleo de gráficos 2380A.

En algunas realizaciones, el procesador de gráficos 2300 incluye recursos de ejecución de hilo escalables que presentan núcleos modulares 2380A-2380N (en ocasiones denominados cortes de núcleo), teniendo cada uno múltiples subnúcleos 2350A-2350N, 2360A-2360N (en ocasiones denominados subcortes de núcleo). En algunas realizaciones, el procesador de gráficos 2300 puede tener cualquier número de núcleos de gráficos 2380A a 2380N. En algunas realizaciones, el procesador de gráficos 2300 incluye un núcleo de gráficos 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 de gráficos 2380A-2380N, incluyendo cada uno un conjunto de primeros subnúcleos 2350A-2350N y un conjunto de segundos subnúcleos 2360A-2360N. Cada subnúcleo en el 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 en el conjunto de segundos subnúcleos 2360A-2360N incluye al menos un segundo conjunto de unidades de ejecución 2362A-562N 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 píxel. 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 ilustrativas de un GPE. Los elementos de la **Figura 24** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

En algunas realizaciones, la lógica de ejecución de hilos 2400 incluye un procesador de sombreado 2402, un despachador de hilos 2404, una caché de instrucciones 2406, una matriz de unidades de ejecución ajustable a escala que incluye una pluralidad de unidades de ejecución 2408A-2408N, un muestreador 2410, una caché de datos 2412 y un puerto de datos 2414. En una realización, la matriz de unidad de ejecución escalable puede escalar dinámicamente activando o desactivando una o más unidades de ejecución (por ejemplo, cualquiera de la unidad 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 mediante un tejido de interconexión que enlaza a cada uno de los componentes. En algunas realizaciones, la lógica de ejecución de hilo 2400 incluye una o más conexiones a memoria, tal como la memoria de sistema o memoria caché, a través de uno o más de la 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 fin general programable independiente que puede 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 es escalable para incluir cualquier número de unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 2408A-2408N se usan principalmente para ejecutar programas sombreadores. Un procesador del sombreador 2402 puede procesar los diversos programas sombreadores y despachar hilos de ejecución asociados con los programas sombreadores mediante un despachador de hilo 2404. En una realización, el despachador de hilo incluye lógica para arbitrar las solicitudes de iniciación de hilo de las canalizaciones de gráficos y de medios e instanciar los hilos solicitados en una o más unidades de ejecución en las unidades de ejecución 2408A-2408N. Por ejemplo, la canalización de geometría (por ejemplo, 2336 de la **Figura 23**) puede despachar sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilos 2400 (la **Figura 24**) para su procesamiento. En algunas realizaciones, el despachador de hilos 604 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 soportan un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreador de gráficos 3D convencionales, de manera que programas de sombreado desde bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución soportan un procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), un procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y un procesamiento de propósito general (por ejemplo, sombreadores de cómputo y de medios). Cada una de las unidades de ejecución 2408A-2408N es capaz de múltiples emisiones de una ejecución de una única instrucción - múltiples datos (SIMD), y un funcionamiento de múltiples hilos habilita un entorno de ejecución eficiente frente a accesos de memoria de latencia superior. Cada hilo de hardware dentro de cada unidad de ejecución tiene un archivo de registro de ancho de banda alto dedicado y un estado de hilo independiente asociado. La ejecución es de múltiples emisiones por reloj a canalizaciones capaces de realizar operaciones de números enteros, de coma flotante de precisión sencilla y doble, capacidad de bifurcación de SIMD, operaciones lógicas, operaciones trascendentales y otras operaciones misceláneas. Mientras se esperan datos desde memoria o una de las funciones compartidas, una lógica de dependencia dentro de las unidades de ejecución 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, se pueden dedicar recursos de hardware a procesar otros hilos. Por ejemplo, durante un retardo asociado con una operación de sombreador de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, un sombreador de fragmentos u otro tipo de programa de sombreado, incluyendo un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 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, enmascaramiento y control de flujo de elementos de datos dentro de las instrucciones. El número de canales puede ser independiente del número de Unidades Aritméticas Lógicas (ALU) o Unidades de Coma Flotante (FPU) físicas para un procesador de gráficos particular. En algunas realizaciones, las unidades de ejecución 608A-608N soportan tipos de datos de números enteros y de coma flotante.

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

tamaño de byte (B)). Sin embargo, son posibles diferentes anchuras de vector y tamaños de registro.

Una o más cachés de instrucciones internas (por ejemplo, 2406) se incluyen en la lógica de ejecución de hilos 2400 para almacenar en caché instrucciones de hilo para las unidades de ejecución. En algunas realizaciones, se incluyen una o más cachés de datos (por ejemplo, 2412) para almacenar en caché datos de hilo durante la ejecución de hilo. 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 de muestreo de textura o de medios especializada para procesar datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, los gráficos y las canalizaciones de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilo 2400 mediante lógica de generación y despacho de hilo. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos en datos de píxeles, se invoca la lógica de procesador de píxel (por ejemplo, lógica de sombreador de píxel, lógica de sombreador de fragmento, etc.) dentro del procesador del sombreador 2402 para que calcule adicionalmente información de salida y haga que se escriban los resultados en las superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de estarcido, etc.). En algunas realizaciones, un sombreador de píxeles o un sombreador de fragmentos calcula los valores de los diversos atributos de vértice que se van a interpolar a lo largo del objeto rasterizado. En algunas realizaciones, una lógica de procesador de píxeles dentro del procesador de sombreado 2402 ejecuta entonces un programa de sombreado de píxeles o de fragmentos suministrado por interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreado, el procesador de sombreado 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 usa una lógica de muestreo de textura en el muestreador 2410 para acceder a datos de textura en correlaciones de textura almacenadas en memoria. Operaciones aritméticas sobre los datos de textura y los datos de geometría de entrada computan datos de color de píxel para cada fragmento geométrico, o descartan el procesamiento adicional de uno o más píxeles.

En algunas realizaciones, el puerto de datos 2414 proporciona un mecanismo de acceso de memoria para que la lógica de ejecución de hilos 2400 emita datos procesados a memoria para su procesamiento en una canalización de salida de procesador de gráficos. En algunas realizaciones, el puerto de datos 614 incluye o se acopla a una o más memorias caché (por ejemplo, la caché de datos 2412) para almacenar en caché datos para un acceso de 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 de procesador de gráficos soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los cuadros con línea continua ilustran los componentes que se incluyen, en general, en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de las instrucciones. En algunas realizaciones, el formato de instrucción 2500 descrito e ilustrado son macroinstrucciones, en el sentido de que las mismas son instrucciones suministradas a la unidad de ejecución, en contraposición a microoperaciones resultantes de la decodificación de instrucciones una vez que se ha procesado la instrucción.

En algunas realizaciones, las unidades de ejecución de procesador de gráficos soportan de manera nativa instrucciones en un formato de instrucción de 128 bits 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 instrucciones de 64 bits 2530. Las instrucciones nativas disponibles en el formato de instrucciones 64 bits 2530 varían de acuerdo con la realización. En algunas realizaciones, la instrucción se compacta en parte usando un conjunto de valores de índice en un campo de índice 2513. El hardware de unidad de ejecución consulta un conjunto de tablas de compactación basándose en los valores de índice y usa las salidas de tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 2510.

Para cada formato, el código de operación de instrucción 2512 define la operación que ha de realizar la unidad de ejecución. Las unidades de ejecución ejecutan cada instrucción en paralelo a lo largo de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de suma, la unidad de ejecución realiza una operación de suma simultánea a lo largo de cada canal de color que representa un elemento de textura o un elemento de imagen. Por defecto, la unidad de ejecución ejecuta cada instrucción a lo largo de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 2514 habilita el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicación) y el orden de canal de datos (por ejemplo, referenciación). Para instrucciones en el formato de instrucción de 128 bits 2510, un campo de tamaño de ejecución 2516 limita el número 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 compacto de 64 bits 2530.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos, incluyendo dos operandos de origen, src0 2520, src1 2522 y un destino 2518. En algunas realizaciones, las unidades de ejecución soportan instrucciones de

destino dual, en donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 2524), en donde el código de operación de instrucción 2512 determina el número de operandos de origen. El último operando de origen de una instrucción puede ser un valor inmediato (por ejemplo, codificado de manera rígida) pasado con la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 2510 incluye un campo de modo de acceso/dirección 2526 que especifica, por ejemplo, si se usa el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se usa el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos es proporcionada directamente por bits en la instrucción.

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

En una realización, la porción de modo de dirección del campo de modo de acceso/dirección 726 determina si la instrucción va a usar un direccionamiento directo o indirecto. Cuando se usa el modo de direccionamiento de registro directo, bits en la instrucción proporcionan directamente la dirección de registro de uno o más operandos. Cuando se usa un modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos se puede computar basándose en un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

En algunas realizaciones, las instrucciones se agrupan basándose en los campos de bits del código de operación 2512 para simplificar la decodificación de 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 de código de operación precisa mostrada es simplemente un ejemplo. En algunas realizaciones, un grupo de código de operación de movimiento y de lógica 2542 incluye instrucciones de movimiento y de lógica de datos (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y de lógica 2542 comparte los cinco bits más significativos (MSB), en donde las instrucciones de movimiento (mov) están en forma de 0000xxxxb y las instrucciones de lógica están en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 2544 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 2546 incluye una mezcla de instrucciones, incluyendo instrucciones de sincronización (por ejemplo, espera, envío) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo matemático paralelo 2548 incluye instrucciones aritméticas a nivel de componente (por ejemplo, suma, multiplicación (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculo matemático paralelo 2548 realiza las operaciones aritméticas en paralelo a lo largo de canales de datos. El grupo de cálculo matemático vectorial 750 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cálculo matemático vectorial realiza aritmética tal como cálculos de producto escalar sobre operandos de vectores.

Canalización de gráficos

La **Figura 26** es un diagrama de bloques de otra realización de un procesador de gráficos 800. Los elementos de la **Figura 26** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

En algunas realizaciones, el procesador de gráficos 2600 incluye una canalización de gráficos 2620, una canalización de medios 2630, un motor de visualización 2640, una lógica de ejecución de hilos 2650 y una canalización de salida de representación 2670. En algunas realizaciones, el procesador de gráficos 2600 es un procesador de gráficos dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos se controla por escrituras de registro en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 2600 mediante una interconexión en anillo 2602. En algunas realizaciones, la interconexión en anillo 802 acopla el procesador de gráficos 2600 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de fin general. Los comandos desde la interconexión en anillo 802 son interpretados por un transmisor por flujo continuo 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 por flujo continuo de comandos 2603 dirige el funcionamiento de un extractor de vértices 2605 que lee datos de vértice desde memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor por flujo continuo de comandos 2603. En algunas realizaciones, el extractor de vértices 805 proporciona datos de vértice a un sombreador de vértices 2607, que realiza operaciones de transformación y de iluminación de espacio de coordenadas en cada vértice. En algunas realizaciones, el extractor de vértices 805 y el sombreador de vértices 2607 ejecutan instrucciones de procesamiento de vértices despachando hilos

de ejecución a las unidades de ejecución 2652A-2652B a través de un despachador de hilos 2631.

En algunas realizaciones, las unidades de ejecución 2652A-2652B son una matriz de procesadores de vectores que tienen un conjunto de instrucciones para realizar operaciones de gráficos y de medios. En algunas realizaciones, las unidades de ejecución 2652A-2652B tienen una caché de L1 2651 anexada que es específica para cada matriz o que se comparte entre las matrices. La caché se puede configurar como una caché de datos, una caché de instrucciones o una única caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.

En algunas realizaciones, la canalización de gráficos 2620 incluye componentes de teselación para realizar una 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 trasero de la salida de teselación. Un teselador 2613 opera en la dirección del sombreador de casco 2611 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados basándose en un modelo geométrico aproximado que se proporciona como entrada a la canalización de gráficos 2620. En algunas realizaciones, si no se usa la teselación, se pueden sortear los componentes de teselación (por ejemplo, el sombreador de casco 2611, el teselador 2613 y el sombreador de dominio 2617).

En algunas realizaciones, pueden procesarse los objetos geométricos completos por un sombreador de geometría 2619 mediante uno o más hilos despachados a unidades de ejecución 2652A-2652B o puede continuarse directamente al recortador 2629. En algunas realizaciones, el sombreador de geometría opera sobre objetos geométricos enteros, en lugar de vértices o parches de vértices como en fases previas de la canalización de gráficos. Si la teselación está inhabilitada, el sombreador de geometría 2619 recibe una entrada desde el sombreador de vértices 2607. En algunas realizaciones, el sombreador de geometría 2619 se puede programar mediante un programa de sombreado de geometría para realizar una teselación de geometría si las unidades de teselación están inhabilitadas.

Antes de la rasterización, un recortador 2629 procesa datos de vértice. El recortador 2629 puede ser un recortador de función fija o un recortador programable que tiene funciones de recorte y de sombreador de geometría. En algunas realizaciones, un componente de prueba de rasterizador y de profundidad 2673 en la canalización de salida de representación 2670 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunas realizaciones, la lógica de sombreado de píxeles se incluye en la lógica de ejecución de hilos 2650. En algunas realizaciones, una aplicación puede sortear el componente de prueba de rasterizador y de profundidad 2673 y acceder a datos de vértice sin rasterizar a través de una unidad de salida de flujo 2623.

El procesador de gráficos 2600 tiene un bus de interconexión, un tejido de interconexión o algún otro mecanismo de interconexión que permite el paso de datos y de mensajes entre los componentes principales del procesador. En algunas realizaciones, las unidades de ejecución 2652A-2652B y la caché o cachés 2651 asociadas, el muestreador de textura y de medios 2654 y la caché de textura/muestreador 2658 se interconectan a través de un puerto de datos 2656 para realizar un acceso de memoria y comunicarse con componentes de canalización de salida de representación del procesador. En algunas realizaciones, el muestreador 2654, las cachés 2651, 2658 y las unidades de ejecución 2652A-2652B tienen, cada uno, rutas de acceso de memoria separadas.

En algunas realizaciones, la canalización de salida de representación 2670 contiene un componente de prueba de rasterizador y de profundidad 2673 que convierte objetos basados en vértices en una representación basada en píxeles asociada. En algunas realizaciones, la lógica de rasterizador incluye una unidad generadora de ventanas/enmascaradora para realizar una rasterización de líneas y de triángulos de función fija. Una caché de representación 2678 y una caché de profundidad 2679 asociadas también están disponibles en algunas realizaciones. Un componente de operaciones de píxel 2677 realiza operaciones basadas en píxeles sobre los datos, aunque, en algunas instancias, las operaciones de píxel asociadas con operaciones 2D (por ejemplo, transferencias de imagen de bloque de bits con mezcla) son realizadas por el motor 2D 2641, o son sustituidas en el momento de la visualización por el controlador de visualización 2643 usando planos de visualización de superposición. En algunas realizaciones, está disponible una caché de L3 compartida 2675 para todos los componentes de gráficos, permitiendo la compartición de datos sin el uso de memoria de sistema principal.

En algunas realizaciones, la canalización de medios del procesador de gráficos 2630 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 desde el emisor por flujo continuo de comando 2603. En algunas realizaciones, la canalización de medios 2630 incluye un emisor por flujo continuo de comando separado. 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 funcionalidad de generación de hilos para abarcar hilos para despachar a la lógica de ejecución de hilo 2650 mediante el despachador de hilo 2631.

En algunas realizaciones, el procesador de gráficos 2600 incluye un motor de visualización 840. 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 en anillo 2602, o algún otro bus o tejido 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 canalización de 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 sistema, como en un ordenador portátil, o un dispositivo de visualización externo anexado a través de un conector de dispositivo de visualización.

En algunas realizaciones, la canalización de gráficos 2620 y la canalización de medios 2630 se pueden configurar para realizar operaciones basándose en múltiples interfaces de programación de gráficos y de medios y no son específicas de ninguna interfaz de programación de aplicaciones (API) concreta. En algunas realizaciones, software de controlador para el procesador de gráficos traduce llamadas de API que son específicas de una biblioteca de medios o de gráficos particular a comandos que pueden ser procesados por el procesador de gráficos. En algunas realizaciones, se proporciona soporte para la Biblioteca de Gráficos Abierta (OpenGL), Lenguaje Informático Abierto (OpenCL) y/o API de gráficos y de cómputo Vulkan, todas ellas del grupo Khronos. En algunas realizaciones, también se puede proporcionar soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, se puede soportar una combinación de estas bibliotecas. También se puede proporcionar soporte para la Biblioteca de Visión por Ordenador de Código Abierto (OpenCV). También se soportaría una API futura con una canalización de 3D compatible si se puede hacer una correlación desde la canalización de la API futura a la canalización del procesador de gráficos.

Programación de canalización de gráficos

La **Figura 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 comandos 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 ilustrativo de la **Figura 27A** incluye campos de datos para identificar un cliente objetivo 2702 del comando, un código de operación (código de op.) de comando 2704 y los datos 2706 relevantes para el comando. También se incluyen un subcódigo de operación 2705 y un tamaño de comando 2708 en algunos comandos.

En algunas realizaciones, el cliente 2702 especifica la unidad de cliente del dispositivo de gráficos que procesa los datos de comando. En algunas realizaciones, un analizador de comandos de procesador de gráficos examina el campo de cliente de cada comando para acondicionar el procesamiento adicional del comando y encaminar los datos de comando a la unidad de cliente apropiada. En algunas realizaciones, las unidades de cliente de procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de representación, una unidad de 2D, una unidad de 3D y una unidad de medios. Cada unidad de cliente tiene una canalización de procesamiento correspondiente que procesa los comandos. Una vez que el comando ha sido recibido por la unidad de cliente, la unidad de cliente lee el código de operación 2704 y, si está presente, el subcódigo de operación 2705 para determinar la operación a realizar. La unidad de cliente realiza el comando usando información en el campo de datos 2706. Para algunos comandos, se espera que un tamaño de comando explícito 908 especifique el tamaño del comando. En algunas realizaciones, el analizador de comandos determina automáticamente el tamaño de al menos algunos de los comandos basándose en el código de operación de comando. En algunas realizaciones, los comandos se alinean a través de múltiplos de una palabra doble.

El diagrama de flujo en la **Figura 27B** muestra una secuencia de comandos de procesador de gráficos 2710 ilustrativa. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que cuenta con una realización de un procesador de gráficos usa una versión de la secuencia de comandos mostrada para establecer, ejecutar y terminar un conjunto de operaciones de gráficos. Se muestra y se describe una secuencia de comandos de muestra solo con fines de ejemplo, debido a que las realizaciones no se limitan a estos comandos específicos o a esta secuencia de comandos. Además, los comandos se pueden emitir como un lote de comandos en una secuencia de comandos, de manera que el procesador de gráficos procesará la secuencia de comandos de manera al menos parcialmente concurrente.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 910 puede comenzar con un comando de vaciado de canalización 2712 para hacer que cualquier canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunas realizaciones, la canalización de 3D 2722 y la canalización de medios 2724 no funcionan de manera concurrente. El vaciado de canalización se realiza para hacer que la canalización de gráficos activa complete cualquier comando pendiente. En respuesta a un vaciado de canalización, el analizador de comandos para el procesador de gráficos pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. Opcionalmente, cualquier dato en la caché de representación que se marque como 'sucio' se puede vaciar a memoria. En algunas realizaciones, el comando de vaciado de canalización 2712 se puede usar para la sincronización de canalización o antes de poner el procesador de gráficos en un estado de baja potencia.

En algunas realizaciones, se usa un comando de selección de canalización 2713 cuando una secuencia de comandos requiere que el procesador de gráficos conmute explícitamente entre canalizaciones. En algunas realizaciones, se requiere un comando de selección de canalización 2713 solo una vez dentro de un contexto de ejecución antes de

emitir comandos de canalización, a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunas realizaciones, se requiere un comando de vaciado de canalización 2712 inmediatamente antes de una conmutación de canalización a través del comando de selección de canalización 2713.

En algunas realizaciones, un comando de control de canalización 2714 configura una canalización de gráficos para su funcionamiento y se usa para programar la canalización de 3D 2722 y la canalización de medios 2724. En algunas realizaciones, el comando de control de canalización 2714 configura el estado de canalización para la canalización activa. En una realización, el comando de control de canalización 2714 se usa para la sincronización de canalización y para borrar datos de una o más memorias caché dentro de la canalización activa antes de procesar un lote de comandos.

En algunas realizaciones, se usan comandos del estado de memoria intermedia de retorno 2716 para configurar un conjunto de memorias intermedias de retorno para que las canalizaciones respectivas escriban datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas realizaciones, el procesador de gráficos también usa una o más memorias intermedias de retorno para almacenar datos de salida y para realizar una comunicación a través de hilos. En algunas realizaciones, configurar el estado de memoria intermedia de retorno 2716 incluye seleccionar el tamaño y el número de memorias intermedias de retorno a usar para un conjunto de operaciones de canalización.

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

Los comandos para configurar el estado de canalización de 3D 930 incluyen comandos de ajuste de estado de 3D para estado de memoria intermedia de vértice, estado de elemento de vértice, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que han de configurarse antes de que se procesen los comandos de primitiva 3D. Los valores de estos comandos se determinan, al menos en parte, basándose en la API 3D particular en uso. En algunas realizaciones, los comandos del estado de canalización de 3D 2730 también son capaces de inhabilitar o sortear selectivamente ciertos elementos de canalización si esos elementos no se van a usar.

En algunas realizaciones, el comando de la primitiva 3D 2732 se usa para enviar primitivas 3D para que sean procesadas por la canalización de 3D. Los comandos y parámetros asociados que se pasan al procesador de gráficos a través del comando de la primitiva 3D 2732 se reenvían a la función de extracción de vértices en la canalización de gráficos. La función de extracción de vértices usa los datos de comando de la primitiva 3D 2732 para generar estructuras de datos de vértice. Las estructuras de datos de vértice se almacenan en una o más memorias intermedias de retorno. En algunas realizaciones, el comando de la primitiva 3D 2732 se usa para realizar operaciones de vértice sobre primitivas 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la canalización de 3D 2722 despacha hilos de ejecución de sombreador a unidades de ejecución de procesador de gráficos.

En algunas realizaciones, la canalización de 3D 2722 se desencadena a través de un comando o evento de la ejecución 2734. En algunas realizaciones, una escritura de registro desencadena una ejecución de comando. En algunas realizaciones, la ejecución se desencadena a través de un comando 'ir' o 'poner en marcha' en la secuencia de comandos. En una realización, la ejecución de comando se desencadena usando un comando de sincronización de canalización para vaciar la secuencia de comandos a través de la canalización de gráficos. La canalización de 3D realizará un procesamiento de geometría para las primitivas 3D. Una vez que se han completado las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles da color a los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo trasero de píxeles para esas operaciones.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 2710 sigue la ruta de la canalización de medios 2724 cuando se realizan operaciones de medios. En general, el uso específico y manera específicos de la programación para la canalización de medios 2724 depende de las operaciones de medios o de cálculo a realizar. Operaciones de descodificación de medios específicas se pueden descargar a la canalización de medios durante la descodificación de medios. En algunas realizaciones, la canalización de medios también se puede sortear y la descodificación de medios se puede realizar, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. En una realización, la canalización de medios también incluye elementos para operaciones de unidad de procesador de gráficos de propósito general (GPGPU), en donde el procesador de gráficos se usa para realizar operaciones vectoriales de SIMD usando programas de sombreado computacional que no están relacionados explícitamente con la representación de primitivas de gráficos.

En algunas realizaciones, la canalización de medios 2724 se configura de una manera similar a la de la canalización de 3D 2722. Un conjunto de comandos para configurar el estado de canalización de medios 2740 se despachan o se colocan 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 descodificación de vídeo y de codificación de vídeo dentro de la canalización de medios, tal como el formato de codificación o de descodificación. En algunas realizaciones, los comandos para el estado de canalización de medios 2740 también soportan el uso de uno o más punteros a elementos de estado "indirectos" que contienen un lote de ajustes de estado.

En algunas realizaciones, los comandos de objeto de medios 2742 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo a procesar. En algunas realizaciones, todos los estados de canalización de medios han de ser válidos antes de emitir un comando de objeto de medios 2742. Una vez que se ha configurado el estado de canalización y los comandos de objeto de medios 2742 se han puesto en cola, la canalización de medios 924 se desencadena a través de un comando de ejecución 2744 o un evento de ejecución equivalente (por ejemplo, una escritura de registro). La salida desde la canalización de medios 2724 se puede post-procesar a continuación mediante operaciones proporcionadas por la canalización de 3D 2722 o la canalización de medios 2724. En algunas realizaciones, las operaciones de GPGPU se configuran y se ejecutan de una manera similar a la de las operaciones de medios.

Arquitectura de software de gráficos

La **Figura 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 de sistema 1050 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación de gráficos 3D 2810 contiene uno o más programas de sombreado que incluyen las instrucciones de sombreador 2812. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el lenguaje de sombreador de alto nivel (HLSL) o el lenguaje de sombreador de OpenGL (GLSL). La aplicación también incluye las instrucciones ejecutables 2814 en un lenguaje máquina adecuado para su ejecución por el núcleo de procesador de propósito general 2834. La aplicación también incluye los objetos de gráficos 1016 definidos por datos de vértice.

En algunas realizaciones, el sistema operativo 2820 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo de tipo UNIX de propiedad exclusiva o un sistema operativo de tipo UNIX de código abierto que usa una variante del núcleo de Linux. El sistema operativo 1020 puede soportar una API de gráficos 2822 tal como la API de Direct3D, la API de OpenGL o la API de Vulkan. Cuando está en uso la API de Direct3D, el sistema operativo 2820 usa un compilador de sombreador de extremo frontal 2824 para compilar cualquier instrucción de sombreador 2812 en HLSL a un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede realizar una precompilación de sombreador. En algunas realizaciones, sombreadores de alto nivel se compilan a sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 2810. En algunas realizaciones, las instrucciones de sombreador 2812 se proporcionan en una forma intermedia, tal como una versión de la representación intermedia portátil convencional (SPIR) usada por la API de Vulkan.

En algunas realizaciones, el controlador de gráficos de modo de usuario 2826 contiene un compilador de sombreador de extremo trasero 2827 para convertir las instrucciones de sombreador 2812 en una representación específica de hardware. Cuando está en uso la API de OpenGL, las instrucciones de sombreador 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 usa las funciones de modo de núcleo de sistema operativo 2828 para comunicarse con un controlador de gráficos de modo de núcleo 2829. En algunas realizaciones, el controlador de gráficos de modo de núcleo 1029 se comunica con el procesador de gráficos 2832 para despachar comandos e instrucciones.

Implementaciones de núcleo de IP

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

presente documento.

La **Figura 29** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 1100 que se puede usar para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo de IP 1100 se puede usar para generar diseños reutilizables modulares que se pueden incorporar en un diseño más grande o usarse para construir todo un circuito integrado (por ejemplo, un circuito integrado de SoC). Una instalación de diseño 2930 puede generar una simulación de software 2910 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). El software de simulación 2910 se puede usar para diseñar, someter a prueba y verificar el comportamiento del núcleo de IP usando un modelo de simulación 2912. El modelo de simulación 2912 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. A continuación, puede crearse un diseño de nivel de transferencia de registro (RTL) 2915 o sintetizarse 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 registro de hardware, que incluye la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño RTL 2915, pueden crearse, diseñarse, o sintetizarse también diseños de nivel inferior al nivel de lógica o al nivel de transistores. Por lo tanto, los detalles particulares del diseño y simulación inicial pueden variar.

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

Circuito integrado de sistema en un chip ilustrativo

Las **Figuras 30-32** ilustran circuitos integrados ilustrativos y procesadores de gráficos asociados que se pueden fabricar usando uno o más núcleos de IP, de acuerdo con diversas realizaciones descritas en el presente documento. Además de lo que se ilustra, se pueden incluir otros circuitos y lógica, incluyendo procesadores/núcleos de gráficos adicionales, controladores de interfaz de periféricos o núcleos de procesador de propósito general.

La **Figura 30** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ilustrativo 3000 que se puede fabricar usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 1200 ilustrativo incluye uno o más procesadores de aplicaciones 3005 (por ejemplo, unas 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 de IP modular desde las mismas o múltiples instalaciones de diseño diferentes. El circuito integrado 3000 incluye lógica de bus o de periféricos que incluye un controlador de USB 1225, un controlador de UART 3030, un controlador de SPI/SDIO 3035 y un controlador de I^2S/I^2C 3040. Adicionalmente, el circuito integrado puede incluir un dispositivo de visualización 3045 acoplado a uno o más de un controlador de interfaz multimedios de alta definición (HDMI) 1250 y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 3055. El almacenamiento puede proporcionarse por un subsistema de memoria flash 3060 que incluye memoria flash y un controlador de memoria flash. La interfaz de memoria puede proporcionarse a través de un controlador de memoria 1265 para el acceso 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 ilustrativo de un circuito integrado de sistema en un chip que se puede fabricar usando uno o más núcleos de IP, de acuerdo con una realización. El procesador de gráficos 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 lógica separada, de manera que el procesador de vértices 3105 se optimiza para ejecutar operaciones para programas de sombreado de vértices, mientras que los uno o más procesadores de fragmentos 3115A-3115N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreado de fragmentos o de píxeles. El procesador de vértices 3105 realiza la fase de procesamiento de vértices de la canalización de gráficos 3D y genera primitivas y datos de vértice. El procesador o procesadores de fragmentos 3115A-3115N usan los datos de primitiva y de vértice generados por el procesador de vértices 3105 para producir una memoria intermedia de tramas que se visualiza en un dispositivo de visualización. En una realización, el procesador o procesadores de fragmentos 3115A-3115N se optimizan para ejecutar programas de sombreado de fragmentos de acuerdo con lo previsto en la API de OpenGL, que se pueden usar para realizar operaciones similares como un programa de sombreado de píxeles de acuerdo con lo previsto en la API de Direct 3D.

El procesador de gráficos 3110 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 3120A-

3120B, caché o cachés 3125A-3125B e interconexión o interconexiones de circuito 3130A-3130B. Las una o más MMU 3120A-3120B prevén una correlación de dirección virtual a física para el procesador de gráficos 3110, incluyendo para el procesador de vértices 1305 y/o el procesador o procesadores de fragmentos 3115A-3115N, que pueden hacer referencia a datos de vértice o de imagen/textura almacenados en memoria, además de datos de vértice o de imagen/textura almacenados en las una o más cachés 3125A-3125B. Por ejemplo, la una o más MMU 3120A-3120B se pueden sincronizar con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con uno o más procesadores de aplicaciones 3005, procesadores de imágenes 3015 y/o procesador de vídeo 3020 de la **Figura 30**, de manera que cada procesador 3005-3020 puede participar en un sistema de memoria virtual compartida o unificada. Las una o más interconexiones de circuito 3130A-3130B habilitan que el procesador de gráficos 3110 interaccione con otros núcleos de IP dentro del SoC, o bien a través de un bus interno del SoC o bien a través de una conexión directa, de acuerdo con realizaciones.

La **Figura 32** es un diagrama de bloques que ilustra un procesador de gráficos ilustrativo adicional 3210 de un sistema en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP, 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 las una o más MMU 3120A-3120B, la caché o cachés 3125A-3125B y la interconexión o interconexiones de circuito 3130A-1330B del circuito integrado 3100 de la **Figura 31**.

El procesador de gráficos 3210 incluye uno o más núcleos de sombreador 3215A-3215N (por ejemplo, 3215A, 3215B, 3215C, 3215D, 3215E, 3215F a 3215N-1 y 3115N), lo que proporciona una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo o núcleo puede ejecutar todos los tipos de código de sombreado programable, incluyendo código de programa de sombreado para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. El número exacto de núcleos de sombreador presentes puede variar entre realizaciones e implementaciones. Adicionalmente, el procesador de gráficos 3210 incluye un gestor de tareas entre núcleos 3205, que actúa como un despachador de hilos para despachar hilos de ejecución a uno o más núcleo o núcleos de sombreador 3215A-3215N y una unidad de teselación 3218 para acelerar operaciones de teselación para una representación basada en teselas, en la que las operaciones de representación para una escena se subdividen en el espacio de imágenes, por ejemplo, para aprovechar la coherencia espacial local dentro de una escena o para optimizar el uso de cachés internas.

REIVINDICACIONES

1. Un sistema de aprendizaje profundo para su uso con una estructura de canalización para ejecutar una red de aprendizaje profundo (1406, 1502) que tiene una pluralidad de nodos de red en una plataforma de múltiples núcleos, comprendiendo el sistema:

una matriz de sensores (1400, 1402) configurada para capturar una secuencia de imágenes (1403A);
una memoria intermedia de entrada (1420) configurada para recibir la secuencia de imágenes de tamaño fijo (1403A);
un ejecutor de red (1712) que tiene una pluralidad de núcleos, en donde el ejecutor de red (1712) está configurado para:

ejecutar la red de aprendizaje profundo (1406, 1502) en la pluralidad de núcleos que forman una canalización, en donde la red de aprendizaje profundo (1406, 1502) está configurada como un grafo acíclico y el flujo de datos a través de cada nodo de red fluye en una sola dirección a través del grafo, y
asignar una memoria compartida (1714) como una salida (1706) a uno de la pluralidad de núcleos y una entrada (1704) a otro de los núcleos respectivos para la ejecución secuencial de cada grupo respectivo de nodos de red; y

un analizador de carga de trabajo de red (1708) configurado para

recibir la secuencia de imágenes (1403A, 1403B) como una carga de trabajo para ser procesada por la red de aprendizaje profundo (1406, 1502) en la pluralidad de núcleos,
agrupar los nodos de red en grupos, en donde los nodos de red se distribuyen a los grupos secuencialmente y basándose en una cantidad de cómputo de cada nodo de red para que el flujo de datos a través de cada grupo sea en la misma dirección para cada grupo, en donde la cantidad de cómputo de cada nodo de red se determina por el tamaño de la imagen de la secuencia de imágenes (1403A) y su operación de aprendizaje profundo,
en donde el ejecutor de red (1712) está además configurado para asignar cada grupo de nodos de red a un núcleo respectivo de dicha pluralidad de núcleos, y para ejecutar los grupos de nodos de red usando el núcleo respectivo.

2. Un método de aprendizaje profundo realizado por un sistema para su uso con una estructura de canalización para ejecutar una red de aprendizaje profundo (1406, 1502) que tiene una pluralidad de nodos de red en una plataforma de múltiples núcleos, comprendiendo el método:

capturar una secuencia de imágenes (1403A) usando una matriz de sensores (1400, 1402);
recibir la secuencia de imágenes de tamaño fijo (1403A) en una memoria intermedia de entrada (1420);
ejecutar una red de aprendizaje profundo (1406, 1502) en una pluralidad de núcleos de un ejecutor de red (1712), formando la pluralidad de núcleos una canalización, en donde la red de aprendizaje profundo (1406, 1502) está configurada como un grafo acíclico y el flujo de datos a través de cada nodo de red fluye en una sola dirección a través del grafo; y
asignar, por parte del ejecutor de red (1712), una memoria compartida (1714) como una salida (1706) a uno de la pluralidad de núcleos y una entrada (1704) a otro de los núcleos respectivos para la ejecución secuencial de cada grupo respectivo de nodos de red;
usar un analizador de carga de trabajo de red (1708) para

recibir la secuencia de imágenes (1403A, 1403B) como una carga de trabajo para ser procesada por la red de aprendizaje profundo (1406, 1502) en la pluralidad de núcleos, y
agrupar los nodos de red en grupos, en donde los nodos de red se distribuyen a los grupos secuencialmente y basándose en una cantidad de cómputo de cada nodo de red para que el flujo de datos a través de cada grupo sea en la misma dirección para cada grupo, en donde la cantidad de cómputo de cada nodo de red se determina por el tamaño de la imagen de la secuencia de imágenes (1403A) y su operación de aprendizaje profundo; y

usar el ejecutor de red (1712) para asignar cada grupo de nodos de red a un núcleo respectivo de dicha pluralidad de núcleos, y para ejecutar los grupos de nodos de red usando el núcleo respectivo.

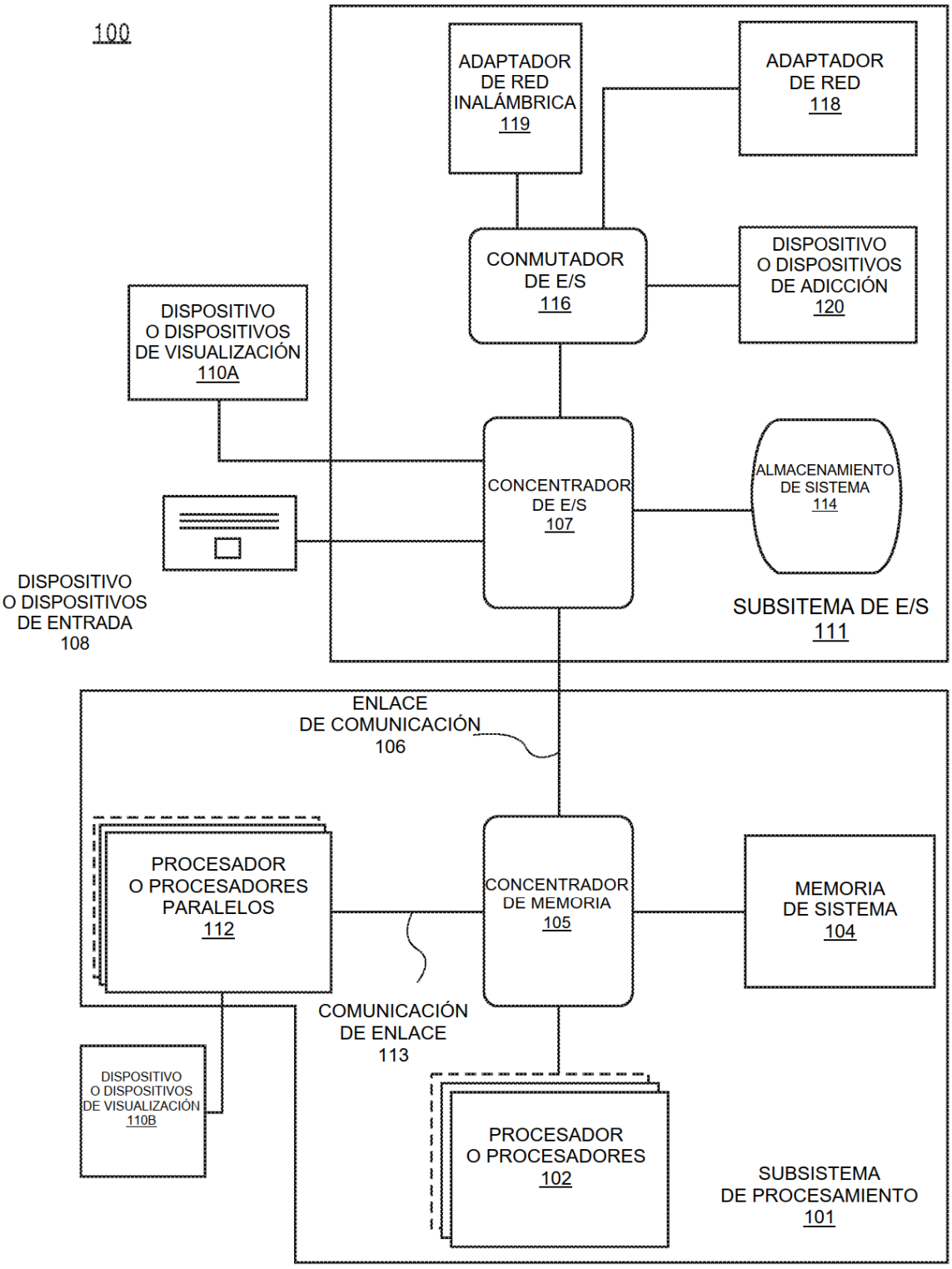

100
ADAPTADOR DE RED INALÁMBRICA 119
ADAPTADOR DE RED 118
CONMUTADOR DE E/S 116
DISPOSITIVO O DISPOSITIVOS DE ADICCIÓN 120
DISPOSITIVO O DISPOSITIVOS DE VISUALIZACIÓN 110A
CONCENTRADOR DE E/S 107
ALMACENAMIENTO DE SISTEMA 114
DISPOSITIVO O DISPOSITIVOS DE ENTRADA 108
SUBSITEMA DE E/S 111
ENLACE DE COMUNICACIÓN 106
PROCESADOR O PROCESADORES PARALELOS 112
CONCENTRADOR DE MEMORIA 105
MEMORIA DE SISTEMA 104
COMUNICACIÓN DE ENLACE 113
DISPOSITIVO O DISPOSITIVOS DE VISUALIZACIÓN 110B
PROCESADOR O PROCESADORES 102
SUBSISTEMA DE PROCESAMIENTO 101

FIG. 1

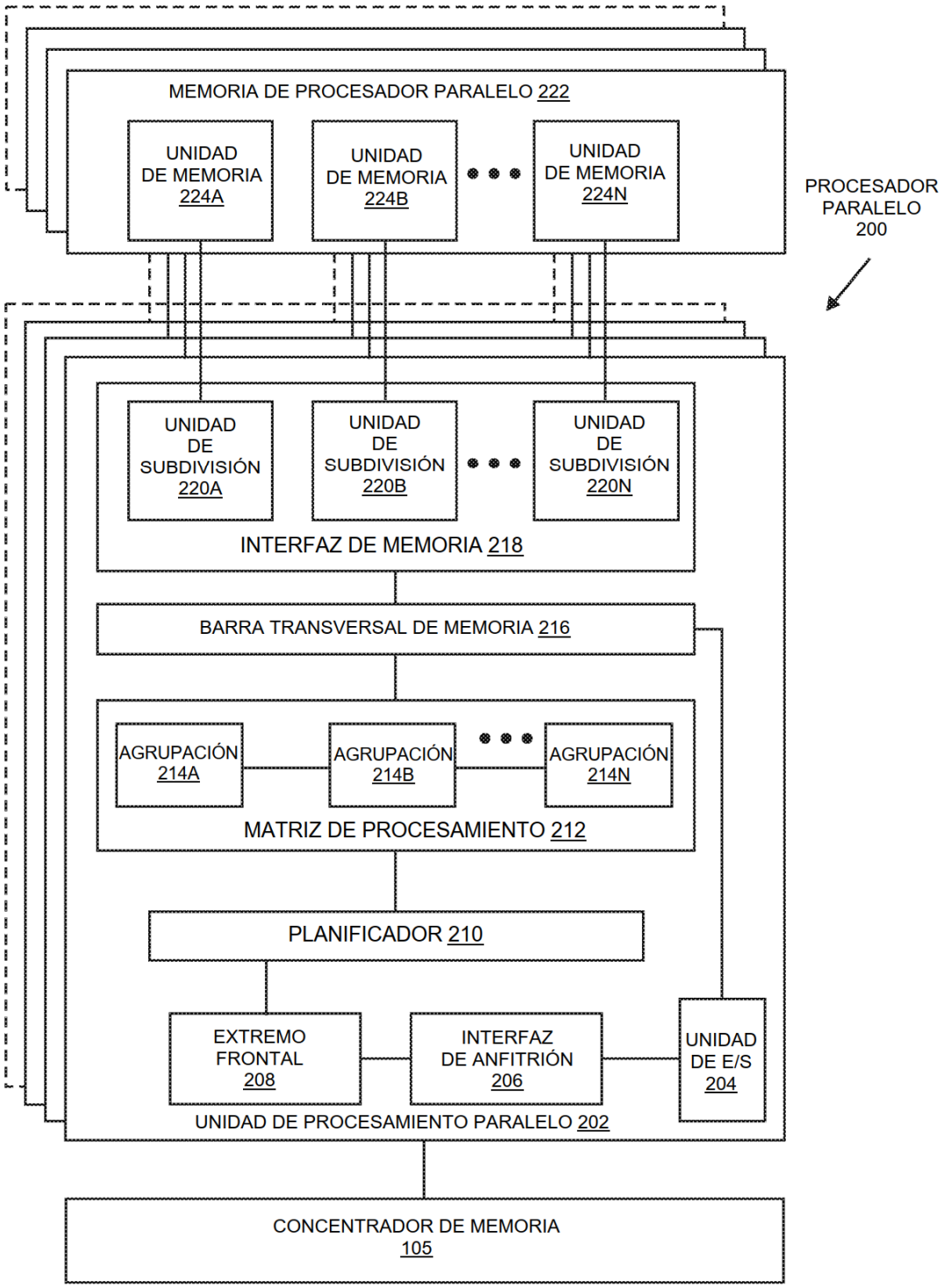

MEMORIA DE PROCESADOR PARALELO 222
UNIDAD DE MEMORIA 224A
UNIDAD DE MEMORIA 224B
UNIDAD DE MEMORIA 224N
PROCESADOR PARALELO 200
UNIDAD DE SUBDIVISIÓN 220A
UNIDAD DE SUBDIVISIÓN 220B
UNIDAD DE SUBDIVISIÓN 220N
INTERFAZ DE MEMORIA 218
BARRA TRANSVERSAL DE MEMORIA 216
AGRUPACIÓN 214A
AGRUPACIÓN 214B
AGRUPACIÓN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

FIG. 2A

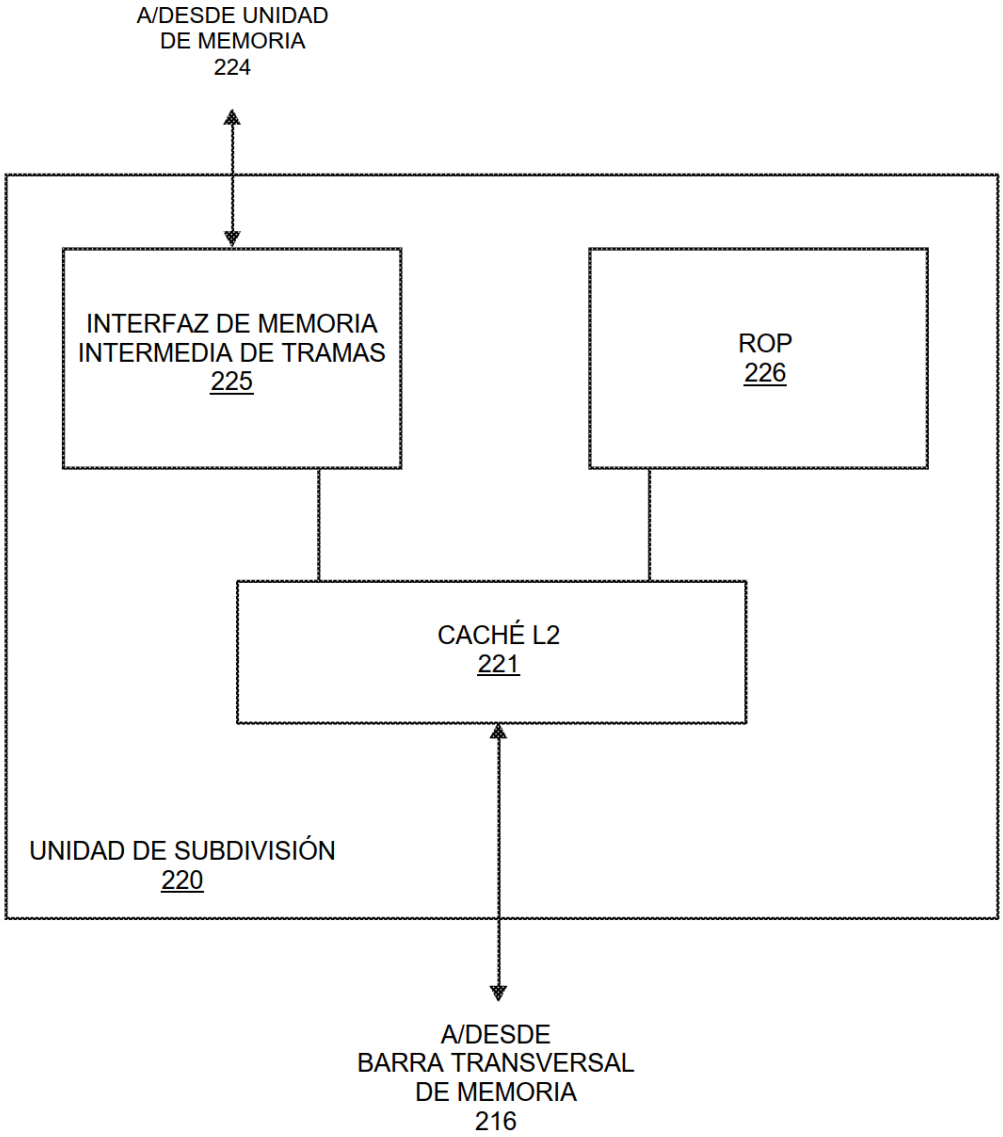

A/DESDE UNIDAD
DE MEMORIA
224
INTERFAZ DE MEMORIA
INTERMEDIA DE TRAMAS
225
ROP
226
CACHÉ L2
221
UNIDAD DE SUBDIVISIÓN
220
A/DESDE
BARRA TRANSVERSAL
DE MEMORIA
216

FIG. 2B

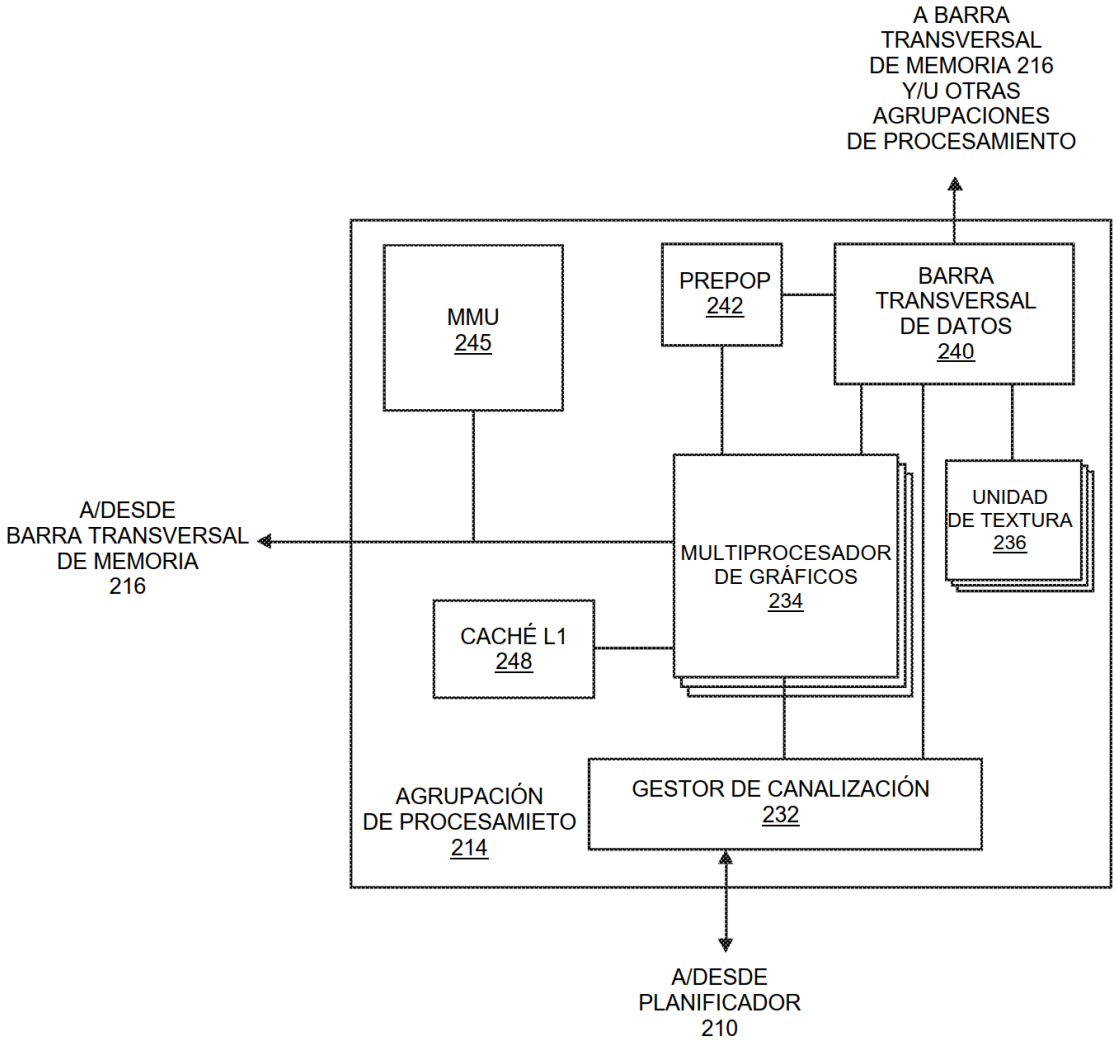

A BARRA TRANSVERSAL DE MEMORIA 216 Y/U OTRAS AGRUPACIONES DE PROCESAMIENTO
MMU 245
PREPOP 242
BARRA TRANSVERSAL DE DATOS 240
UNIDAD DE TEXTURA 236
A/DESDE BARRA TRANSVERSAL DE MEMORIA 216
MULTIPROCESADOR DE GRÁFICOS 234
CACHÉ L1 248
AGRUPACIÓN DE PROCESAMIETO 214
GESTOR DE CANALIZACIÓN 232
A/DESDE PLANIFICADOR 210

FIG. 2C

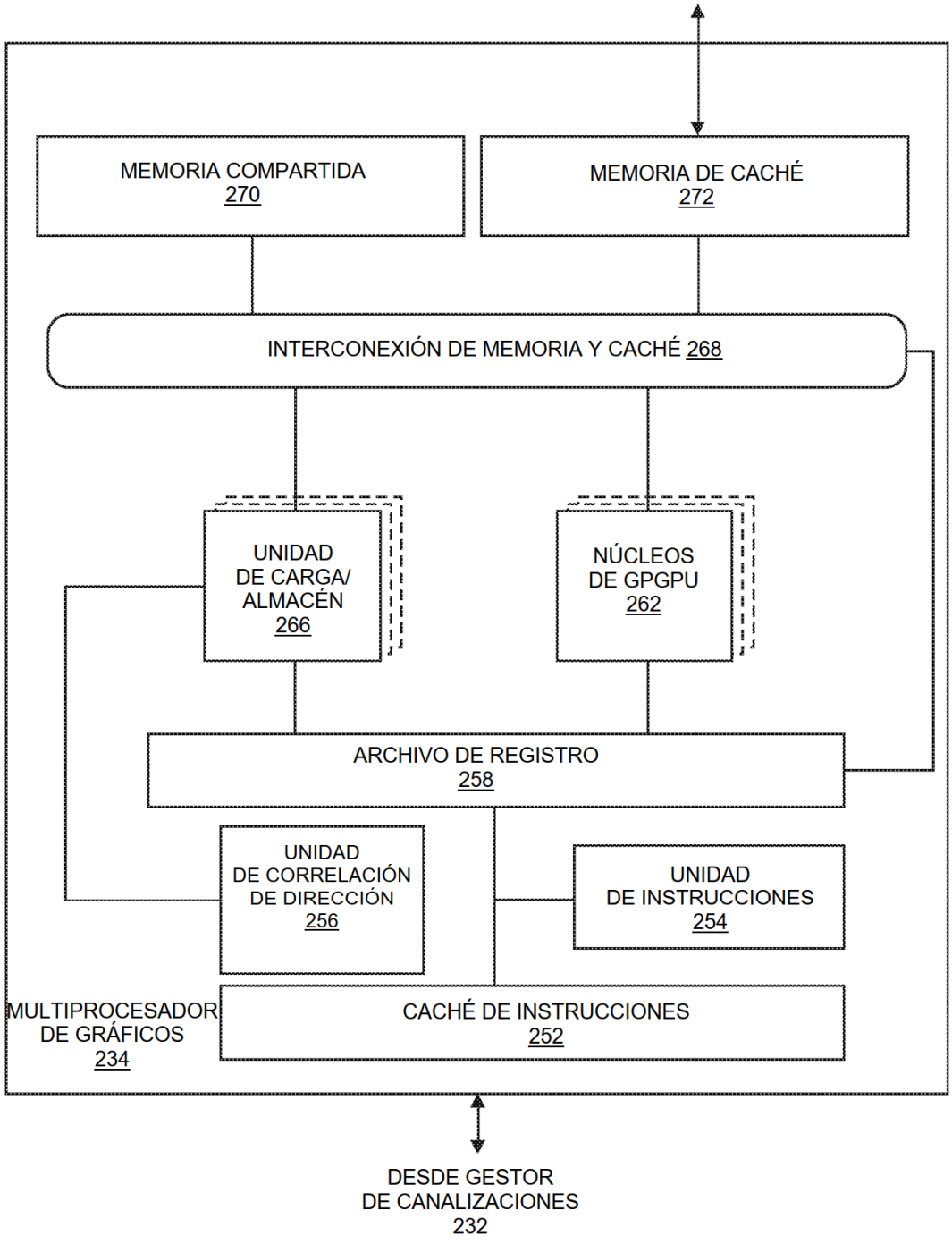

MEMORIA COMPARTIDA
270
MEMORIA DE CACHÉ
272
INTERCONEXIÓN DE MEMORIA Y CACHÉ 268
UNIDAD
DE CARGA/
ALMACÉN
266
NÚCLEOS
DE GPGPU
262
ARCHIVO DE REGISTRO
258
UNIDAD
DE CORRELACIÓN
DE DIRECCIÓN
256
UNIDAD
DE INSTRUCCIONES
254
MULTIPROCESADOR
DE GRÁFICOS
234
CACHÉ DE INSTRUCCIONES
252
DESDE GESTOR
DE CANALIZACIONES
232

FIG. 2D

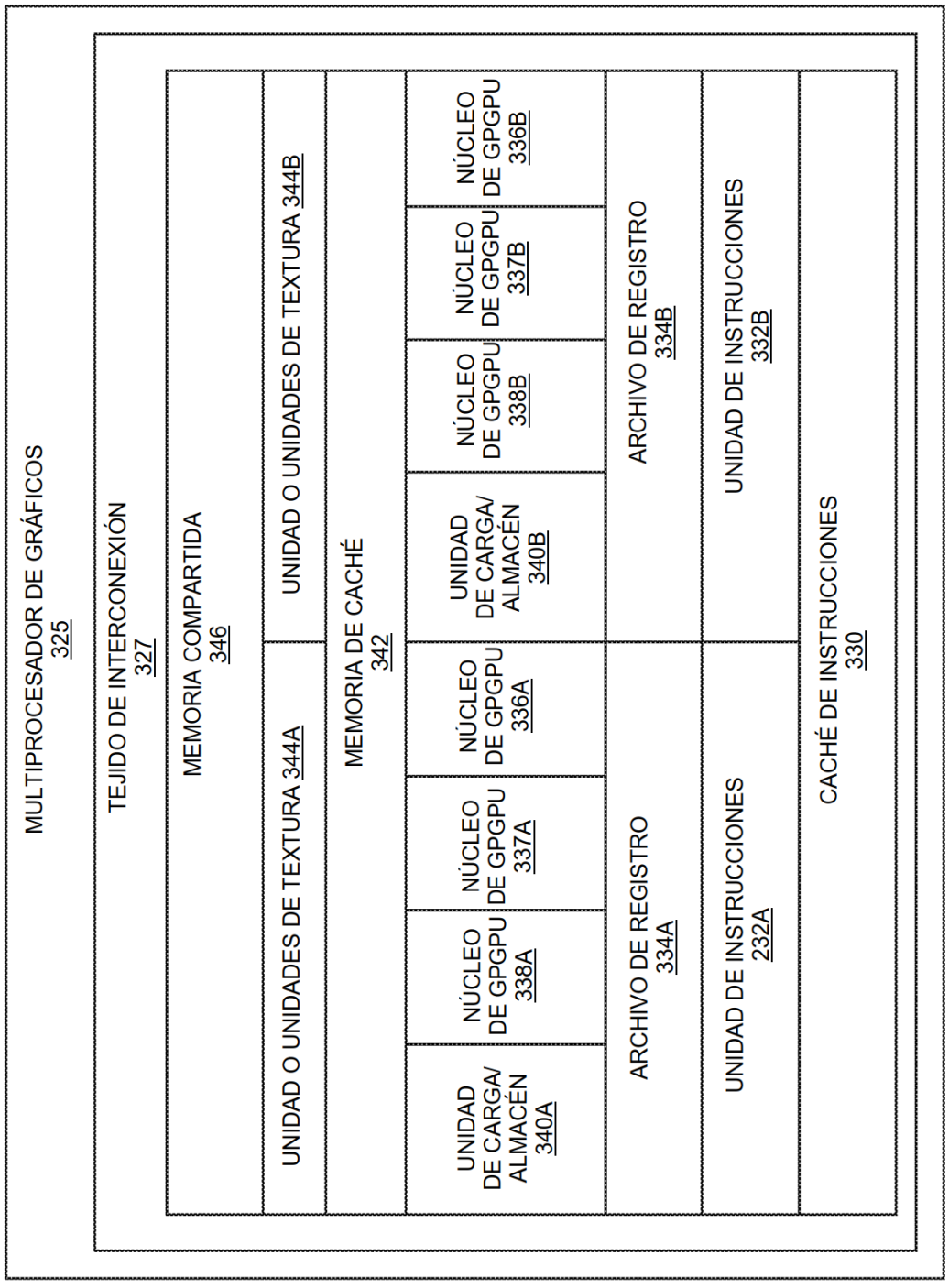

MULTIPROCESADOR DE GRÁFICOS
325
TEJIDO DE INTERCONEXIÓN
327
MEMORIA COMPARTIDA
346
UNIDAD O UNIDADES DE TEXTURA 344A
UNIDAD O UNIDADES DE TEXTURA 344B
MEMORIA DE CACHÉ
342
UNIDAD DE CARGA/ ALMACÉN 340A
NÚCLEO DE GPGPU 338A
NÚCLEO DE GPGPU 337A
NÚCLEO DE GPGPU 336A
UNIDAD DE CARGA/ ALMACÉN 340B
NÚCLEO DE GPGPU 338B
NÚCLEO DE GPGPU 337B
NÚCLEO DE GPGPU 336B
ARCHIVO DE REGISTRO 334A
ARCHIVO DE REGISTRO 334B
UNIDAD DE INSTRUCCIONES 232A
UNIDAD DE INSTRUCCIONES 332B
CACHÉ DE INSTRUCCIONES 330

FIG. 3A

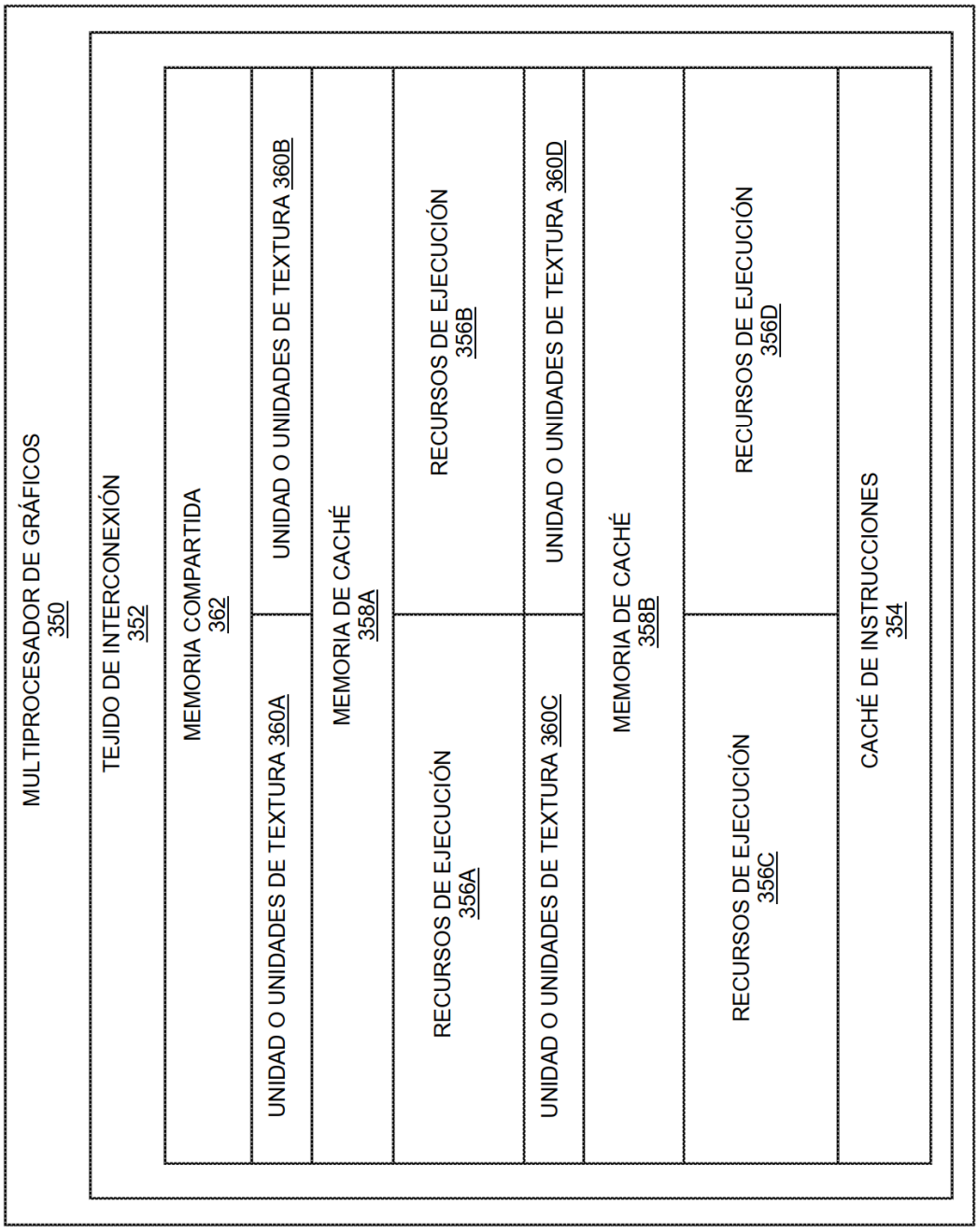

MULTIPROCESADOR DE GRÁFICOS
350
TEJIDO DE INTERCONEXIÓN
352
MEMORIA COMPARTIDA
362
UNIDAD O UNIDADES DE TEXTURA 360A
UNIDAD O UNIDADES DE TEXTURA 360B
MEMORIA DE CACHÉ
358A
RECURSOS DE EJECUCIÓN
356A
RECURSOS DE EJECUCIÓN
356B
UNIDAD O UNIDADES DE TEXTURA 360C
UNIDAD O UNIDADES DE TEXTURA 360D
MEMORIA DE CACHÉ
358B
RECURSOS DE EJECUCIÓN
356C
RECURSOS DE EJECUCIÓN
356D
CACHÉ DE INSTRUCCIONES
354

FIG. 3B

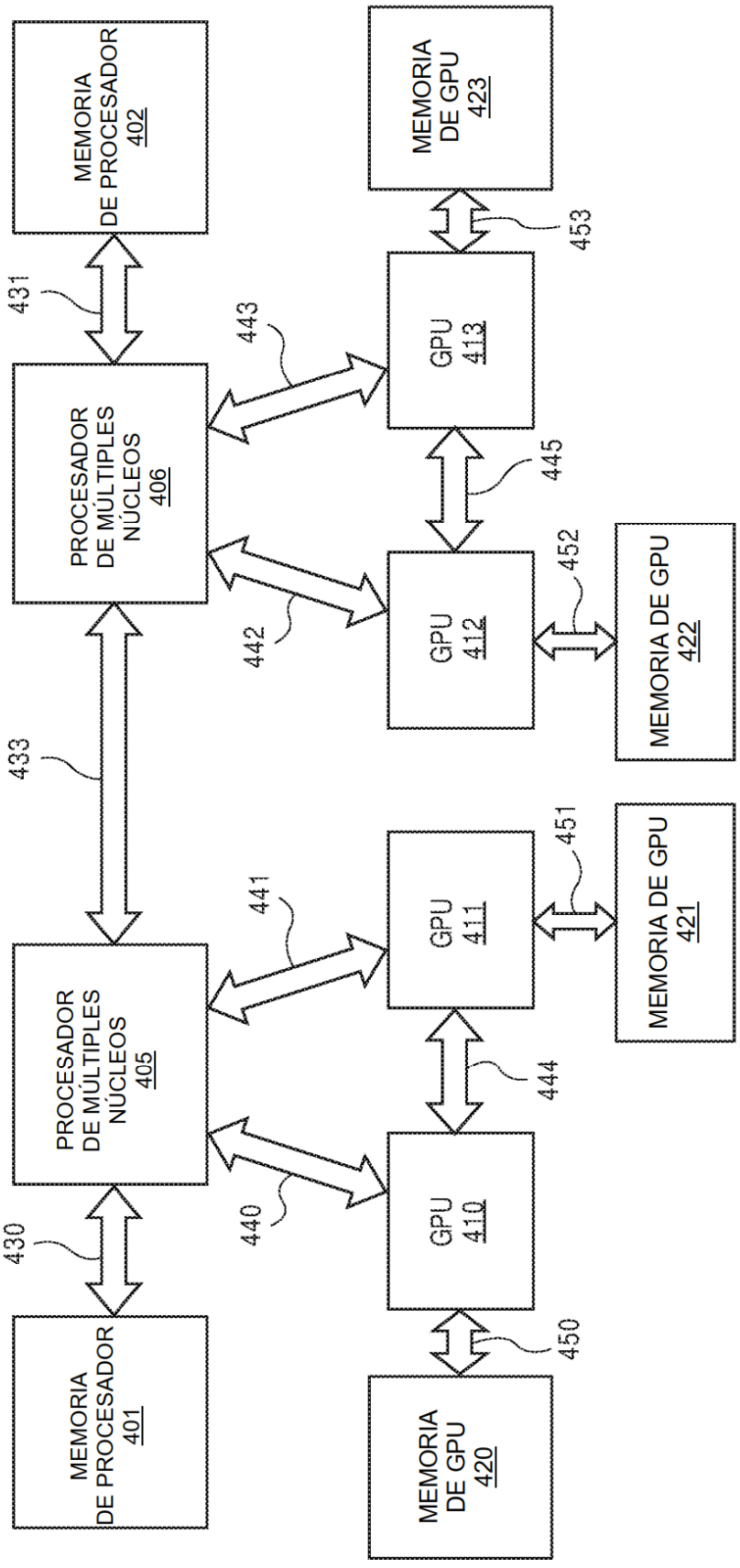

MEMORIA DE PROCESADOR 402
MEMORIA DE GPU 423
431
453
443
GPU 413
PROCESADOR DE MÚLTIPLES NÚCLEOS 406
445
452
442
GPU 412
MEMORIA DE GPU 422
433
451
441
GPU 411
MEMORIA DE GPU 421
PROCESADOR DE MÚLTIPLES NÚCLEOS 405
444
430
440
GPU 410
450
MEMORIA DE PROCESADOR 401
MEMORIA DE GPU 420

FIG. 4A

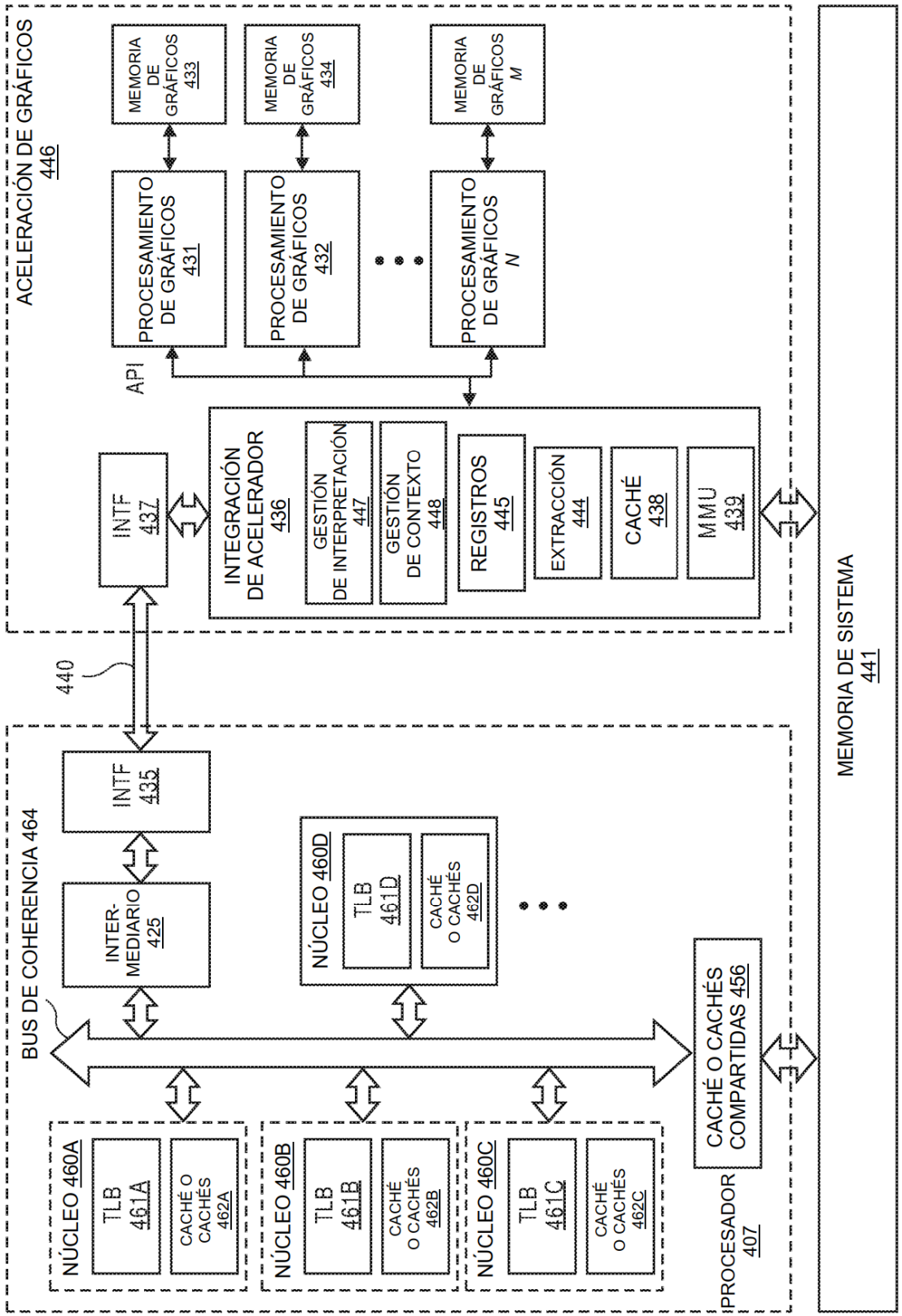

ACELERACIÓN DE GRÁFICOS 446
MEMORIA DE GRÁFICOS 433
MEMORIA DE GRÁFICOS 434
MEMORIA DE GRÁFICOS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 437
INTEGRACIÓN DE ACELERADOR 436
GESTIÓN DE INTERPRETACIÓN 447
GESTIÓN DE CONTEXTO 448
REGISTROS 445
EXTRACCIÓN 444
CACHÉ 438
MMU 439
440
INTF 435
INTER-MEDIARIO 425
BUS DE COHERENCIA 464
NÚCLEO 460D
TLB 461D
CACHÉ O CACHÉS 462D
NÚCLEO 460A
TLB 461A
CACHÉ O CACHÉS 462A
NÚCLEO 460B
TLB 461B
CACHÉ O CACHÉS 462B
NÚCLEO 460C
TLB 461C
CACHÉ O CACHÉS 462C
CACHÉ O CACHÉS COMPARTIDAS 456
PROCESADOR 407
MEMORIA DE SISTEMA 441

FIG. 4B

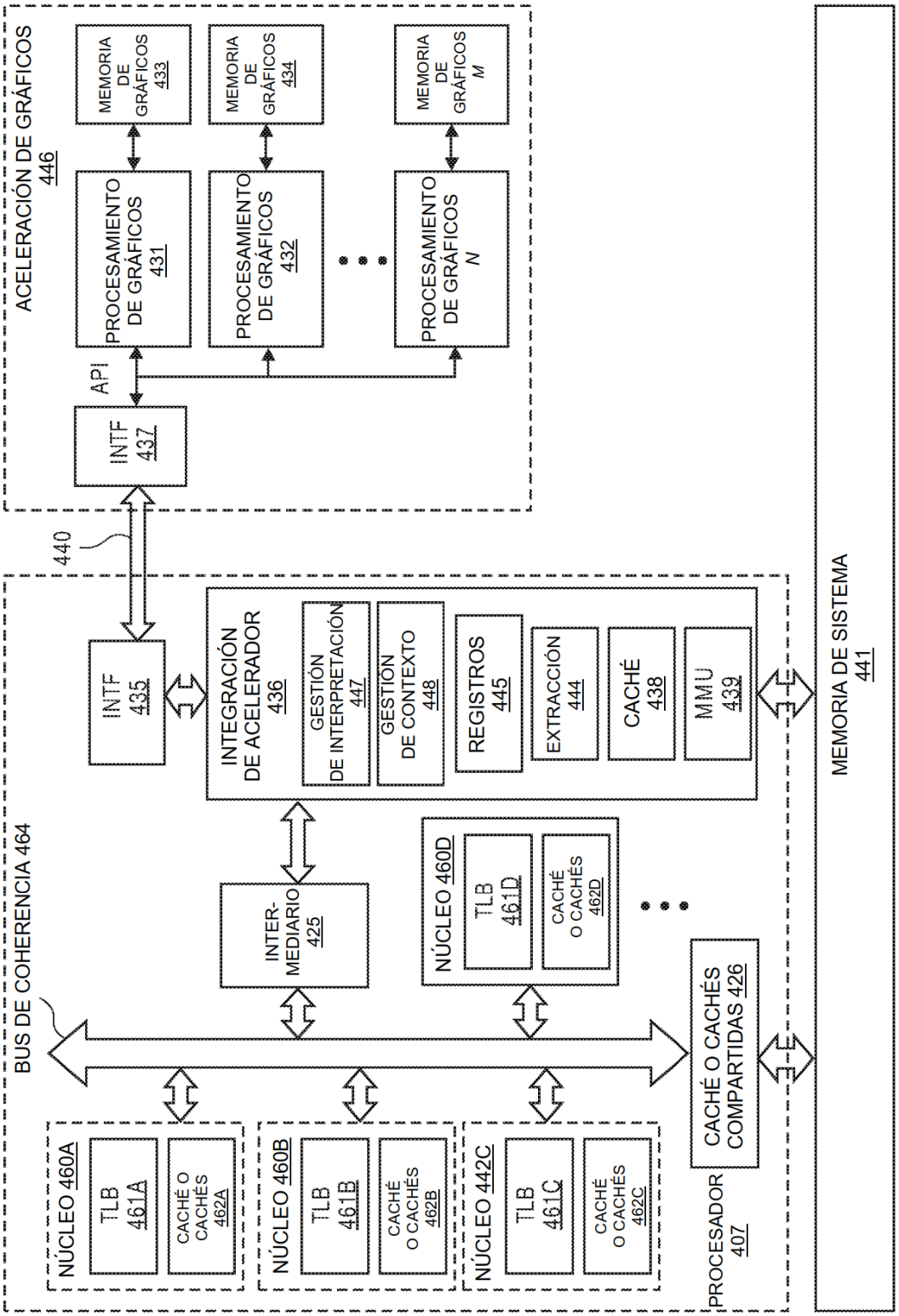

ACELERACIÓN DE GRÁFICOS 446
MEMORIA DE GRÁFICOS 433
MEMORIA DE GRÁFICOS 434
MEMORIA DE GRÁFICOS M
PROCESAMIENTO DE GRÁFICOS 431
PROCESAMIENTO DE GRÁFICOS 432
PROCESAMIENTO DE GRÁFICOS N
API
INTF 437
440
BUS DE COHERENCIA 464
INTF 435
INTEGRACIÓN DE ACELERADOR 436
GESTIÓN DE INTERPRETACIÓN 447
GESTIÓN DE CONTEXTO 448
REGISTROS 445
EXTRACCIÓN 444
CACHÉ 438
MMU 439
INTER-MEDIARIO 425
NÚCLEO 460D
TLB 461D
CACHÉ O CACHÉS 462D
NÚCLEO 460A
TLB 461A
CACHÉ O CACHÉS 462A
NÚCLEO 460B
TLB 461B
CACHÉ O CACHÉS 462B
NÚCLEO 442C
TLB 461C
CACHÉ O CACHÉS 462C
CACHÉ O CACHÉS COMPARTIDAS 426
PROCESADOR 407
MEMORIA DE SISTEMA 441

FIG. 4C

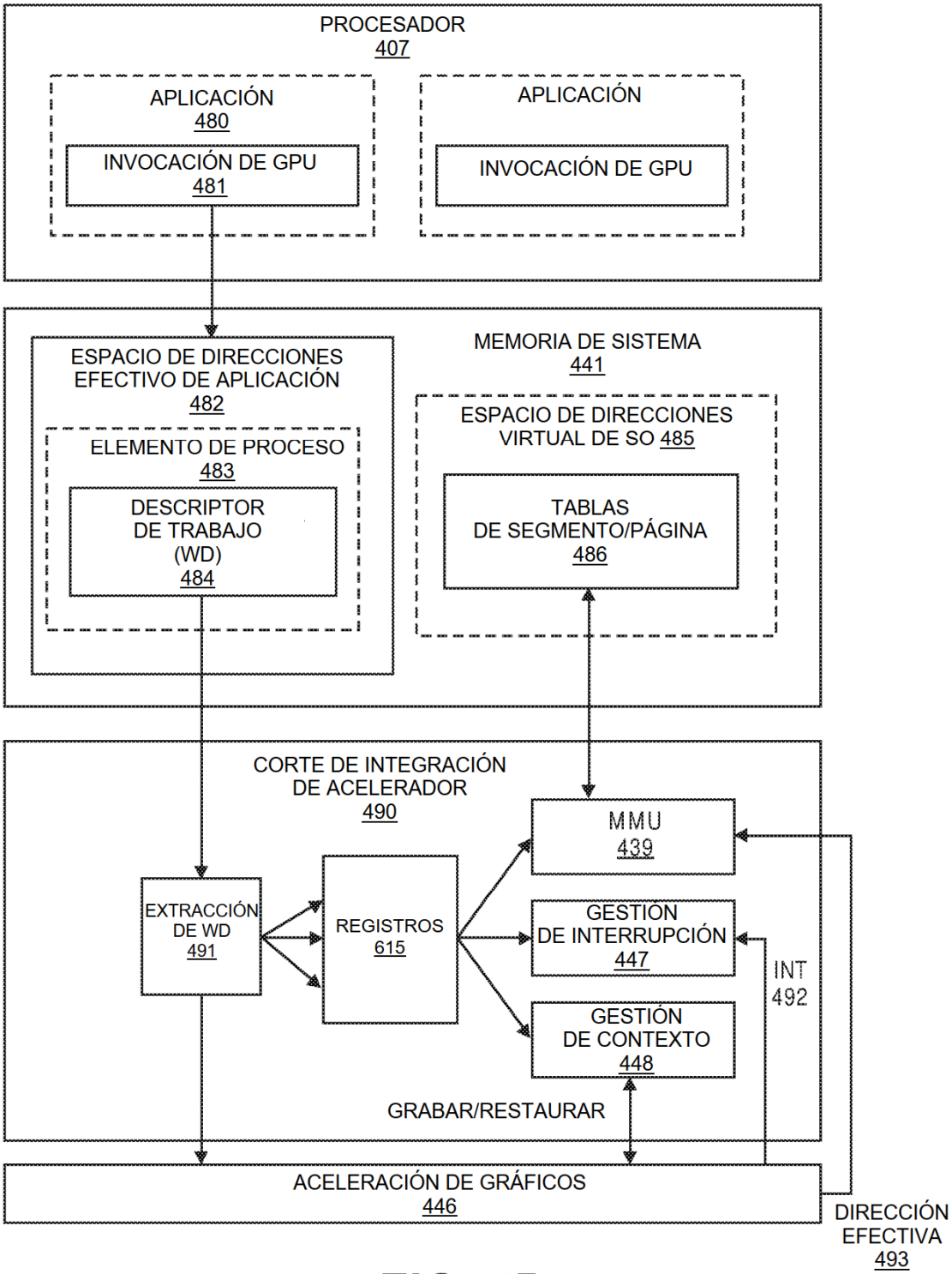

PROCESADOR
407
APLICACIÓN
480
INVOCACIÓN DE GPU
481
APLICACIÓN
INVOCACIÓN DE GPU
ESPACIO DE DIRECCIONES
EFECTIVO DE APLICACIÓN
482
ELEMENTO DE PROCESO
483
DESCRIPTOR
DE TRABAJO
(WD)
484
MEMORIA DE SISTEMA
441
ESPACIO DE DIRECCIONES
VIRTUAL DE SO 485
TABLAS
DE SEGMENTO/PÁGINA
486
CORTE DE INTEGRACIÓN
DE ACELERADOR
490
MMU
439
EXTRACCIÓN
DE WD
491
REGISTROS
615
GESTIÓN
DE INTERRUPCIÓN
447
INT
492
GESTIÓN
DE CONTEXTO
448
GRABAR/RESTAURAR
ACELERACIÓN DE GRÁFICOS
446
DIRECCIÓN
EFECTIVA
493

FIG. 4D

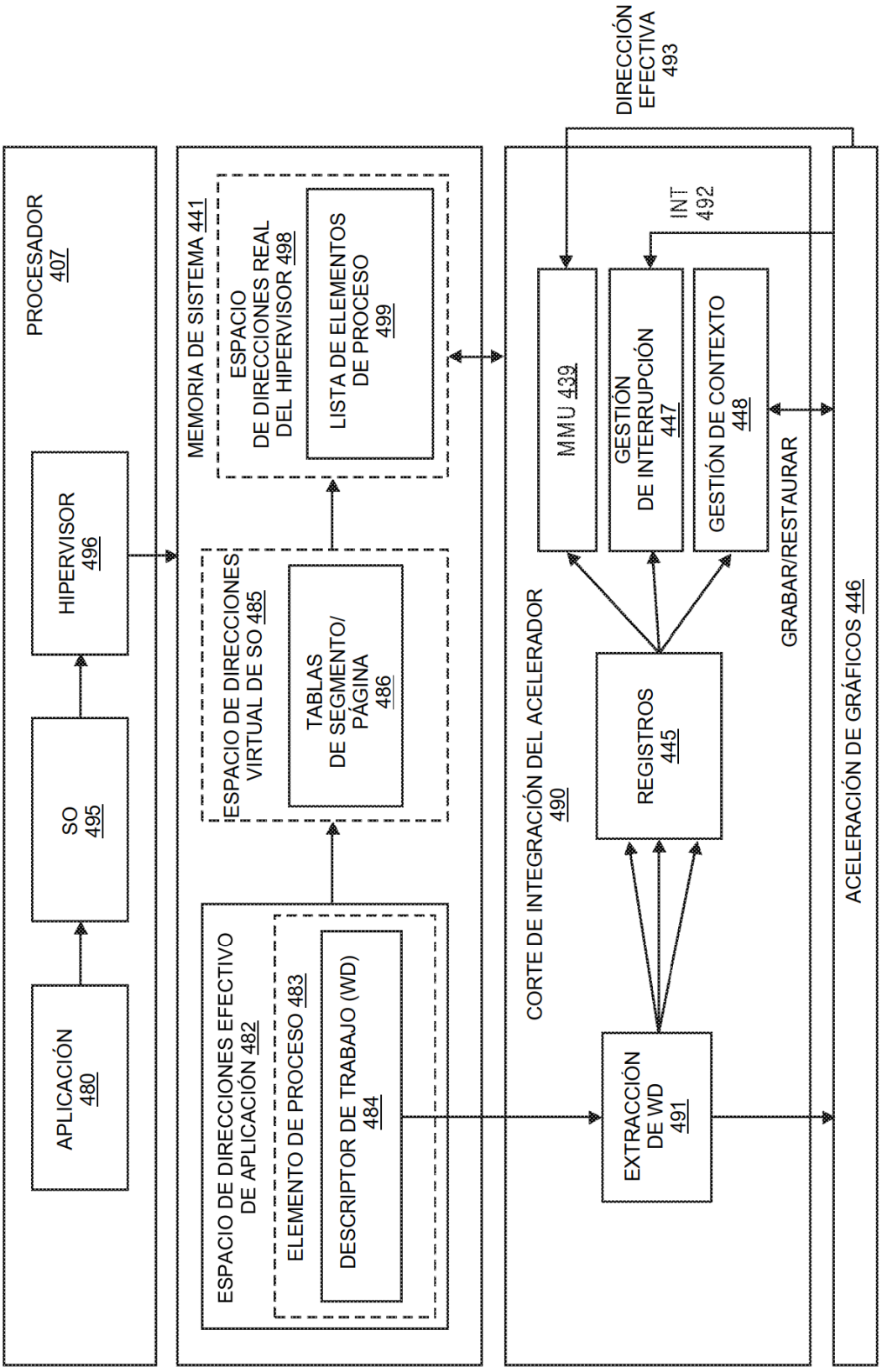

PROCESADOR 407
APLICACIÓN 480
SO 495
HIPERVISOR 496
MEMORIA DE SISTEMA 441
ESPACIO DE DIRECCIONES EFECTIVO DE APLICACIÓN 482
ELEMENTO DE PROCESO 483
DESCRIPTOR DE TRABAJO (WD) 484
ESPACIO DE DIRECCIONES VIRTUAL DE SO 485
TABLAS DE SEGMENTO/ PÁGINA 486
ESPACIO DE DIRECCIONES REAL DEL HIPERVISOR 498
LISTA DE ELEMENTOS DE PROCESO 499
DIRECCIÓN EFECTIVA 493
CORTE DE INTEGRACIÓN DEL ACELERADOR 490
MMU 439
GESTIÓN DE INTERRUPCIÓN 447
GESTIÓN DE CONTEXTO 448
INT 492
REGISTROS 445
EXTRACCIÓN DE WD 491
GRABAR/RESTAURAR
ACELERACIÓN DE GRÁFICOS 446

FIG. 4E

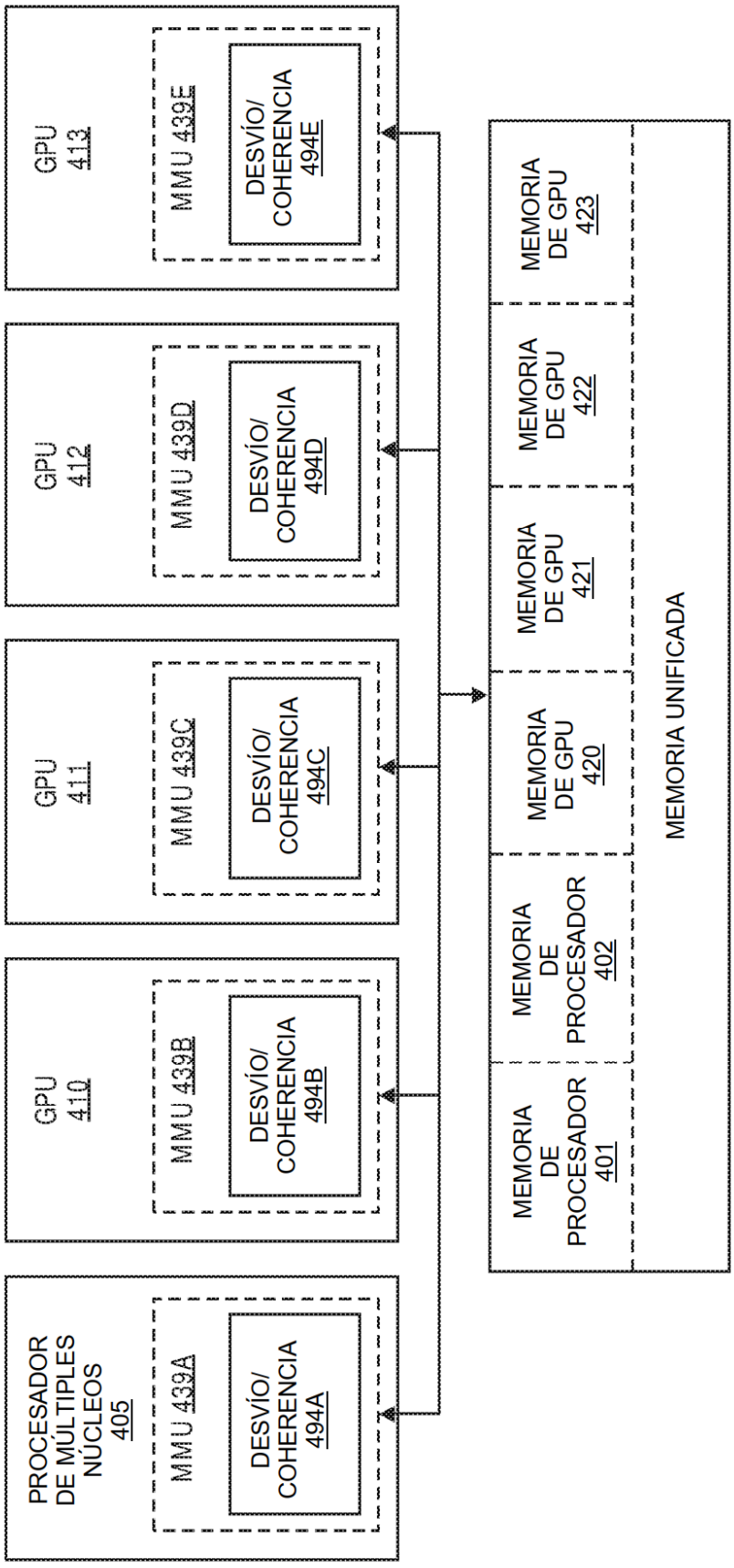

PROCESADOR DE MÚLTIPLES NÚCLEOS 405
MMU 439A
DESVÍO/ COHERENCIA 494A
GPU 410
MMU 439B
DESVÍO/ COHERENCIA 494B
GPU 411
MMU 439C
DESVÍO/ COHERENCIA 494C
GPU 412
MMU 439D
DESVÍO/ COHERENCIA 494D
GPU 413
MMU 439E
DESVÍO/ COHERENCIA 494E
MEMORIA DE PROCESADOR 401
MEMORIA DE PROCESADOR 402
MEMORIA DE GPU 420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

FIG. 4F

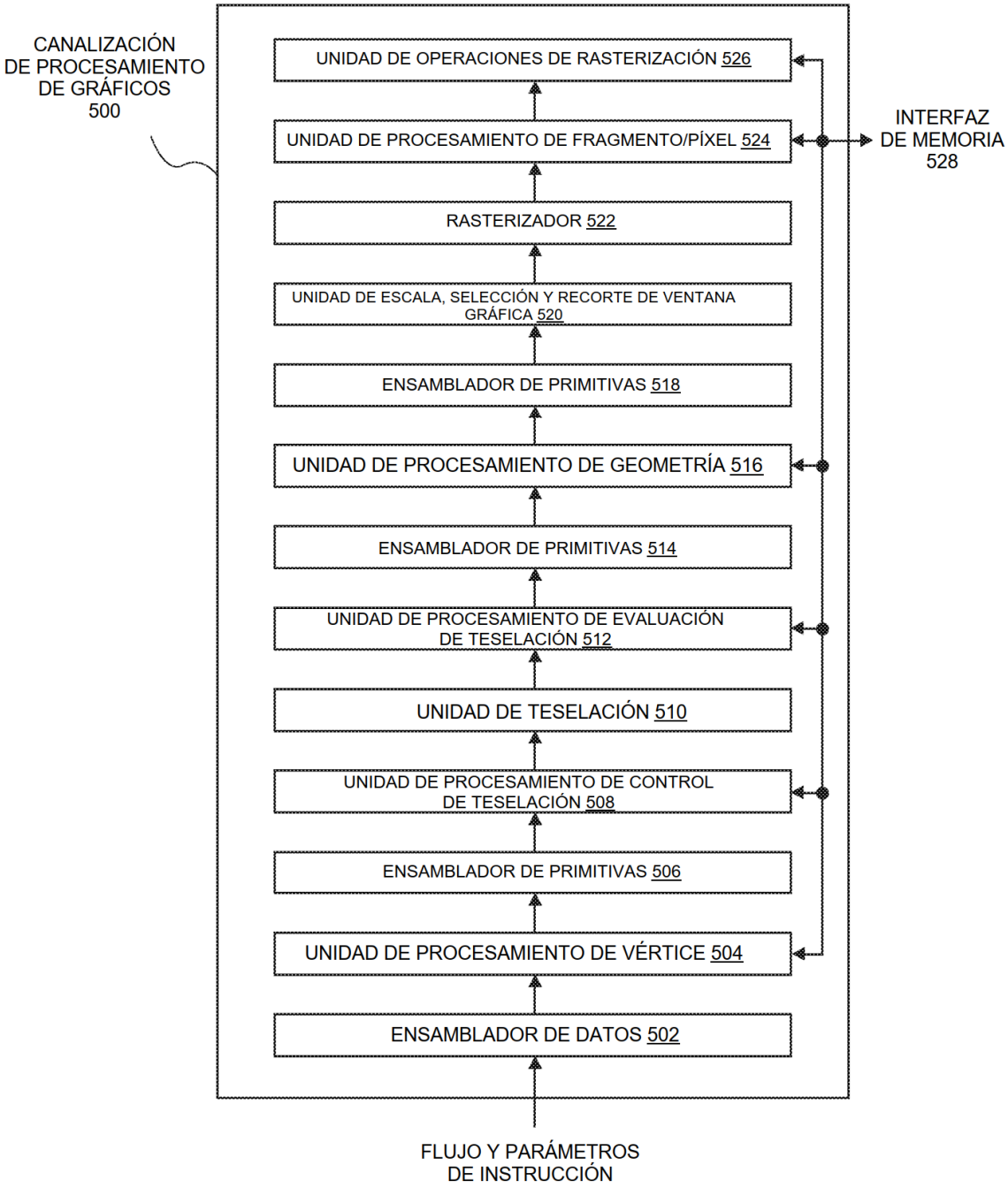

CANALIZACIÓN DE PROCESAMIENTO DE GRÁFICOS 500
UNIDAD DE OPERACIONES DE RASTERIZACIÓN 526
UNIDAD DE PROCESAMIENTO DE FRAGMENTO/PÍXEL 524
INTERFAZ DE MEMORIA 528
RASTERIZADOR 522
UNIDAD DE ESCALA, SELECCIÓN Y RECORTE DE VENTANA GRÁFICA 520
ENSAMBLADOR DE PRIMITIVAS 518
UNIDAD DE PROCESAMIENTO DE GEOMETRÍA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

FIG. 5

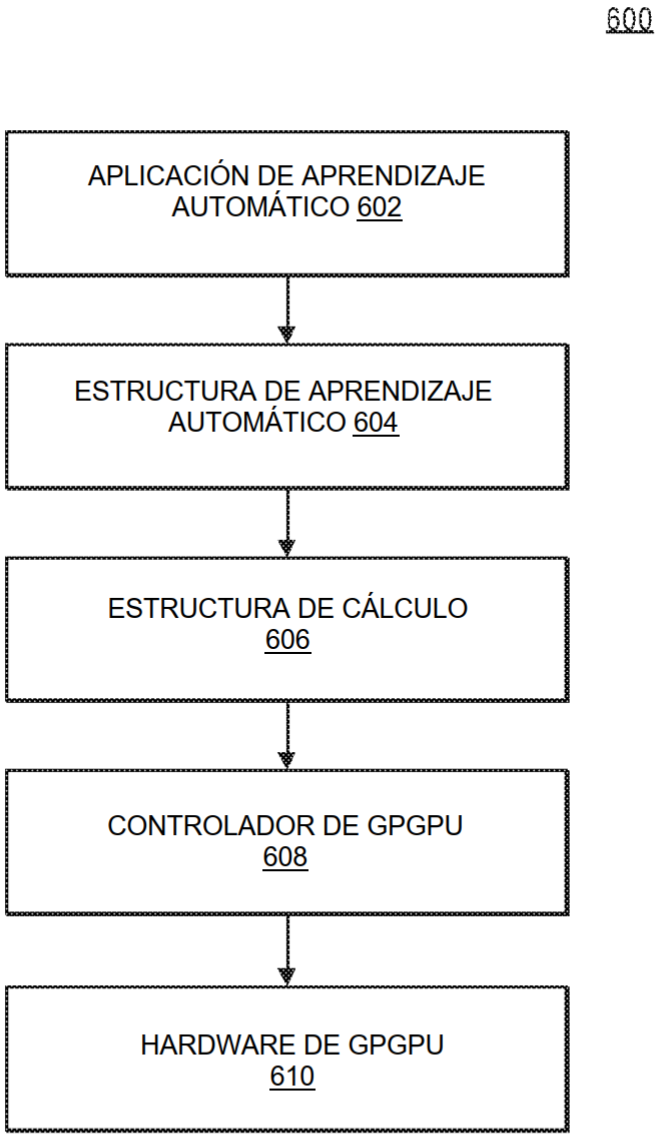

600
APLICACIÓN DE APRENDIZAJE AUTOMÁTICO 602
ESTRUCTURA DE APRENDIZAJE AUTOMÁTICO 604
ESTRUCTURA DE CÁLCULO 606
CONTROLADOR DE GPGPU 608
HARDWARE DE GPGPU 610

FIG. 6

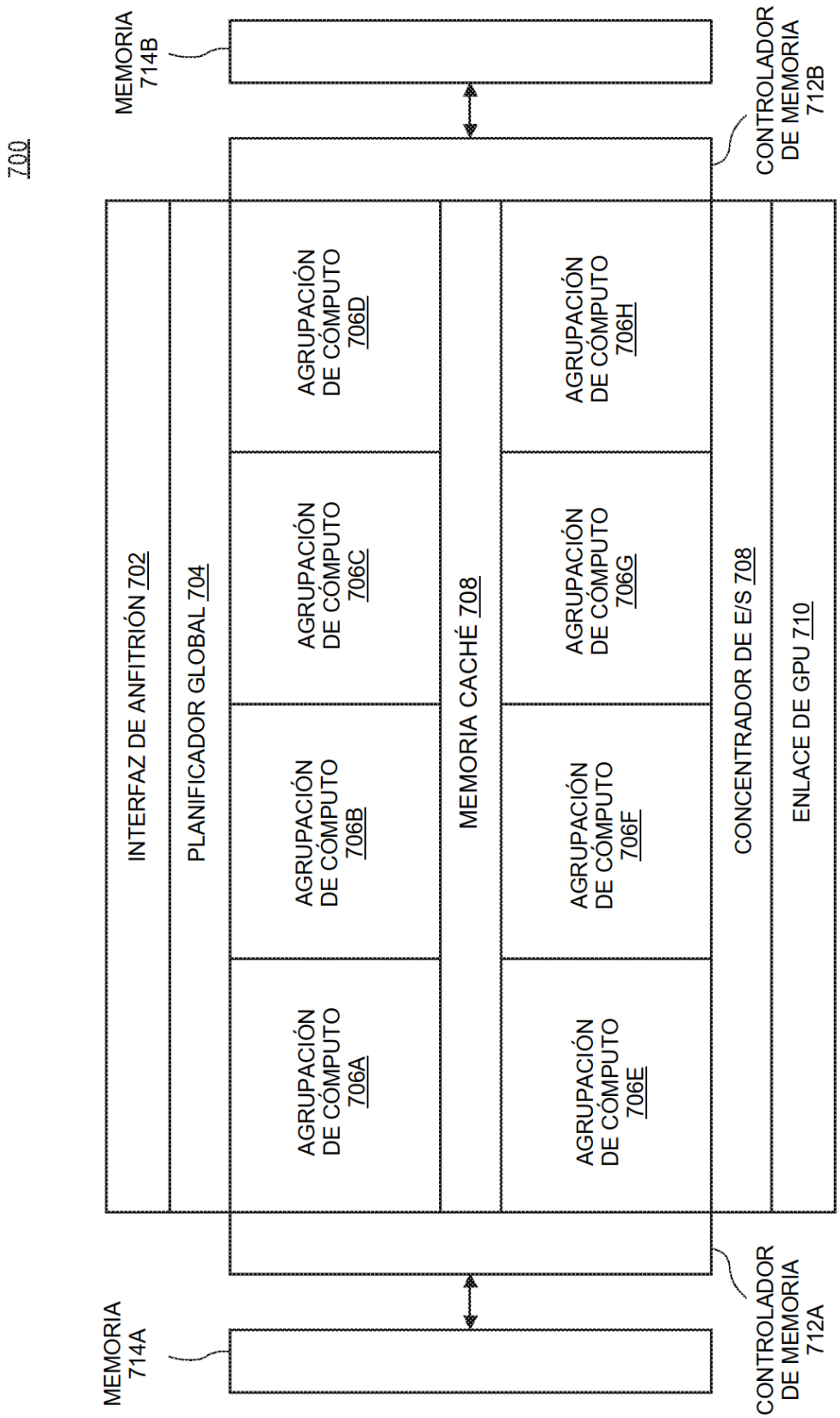

700
MEMORIA 714B
CONTROLADOR DE MEMORIA 712B
INTERFAZ DE ANFITRIÓN 702
PLANIFICADOR GLOBAL 704
AGRUPACIÓN DE CÓMPUTO 706D
AGRUPACIÓN DE CÓMPUTO 706C
AGRUPACIÓN DE CÓMPUTO 706B
AGRUPACIÓN DE CÓMPUTO 706A
MEMORIA CACHÉ 708
AGRUPACIÓN DE CÓMPUTO 706H
AGRUPACIÓN DE CÓMPUTO 706G
AGRUPACIÓN DE CÓMPUTO 706F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

FIG. 7

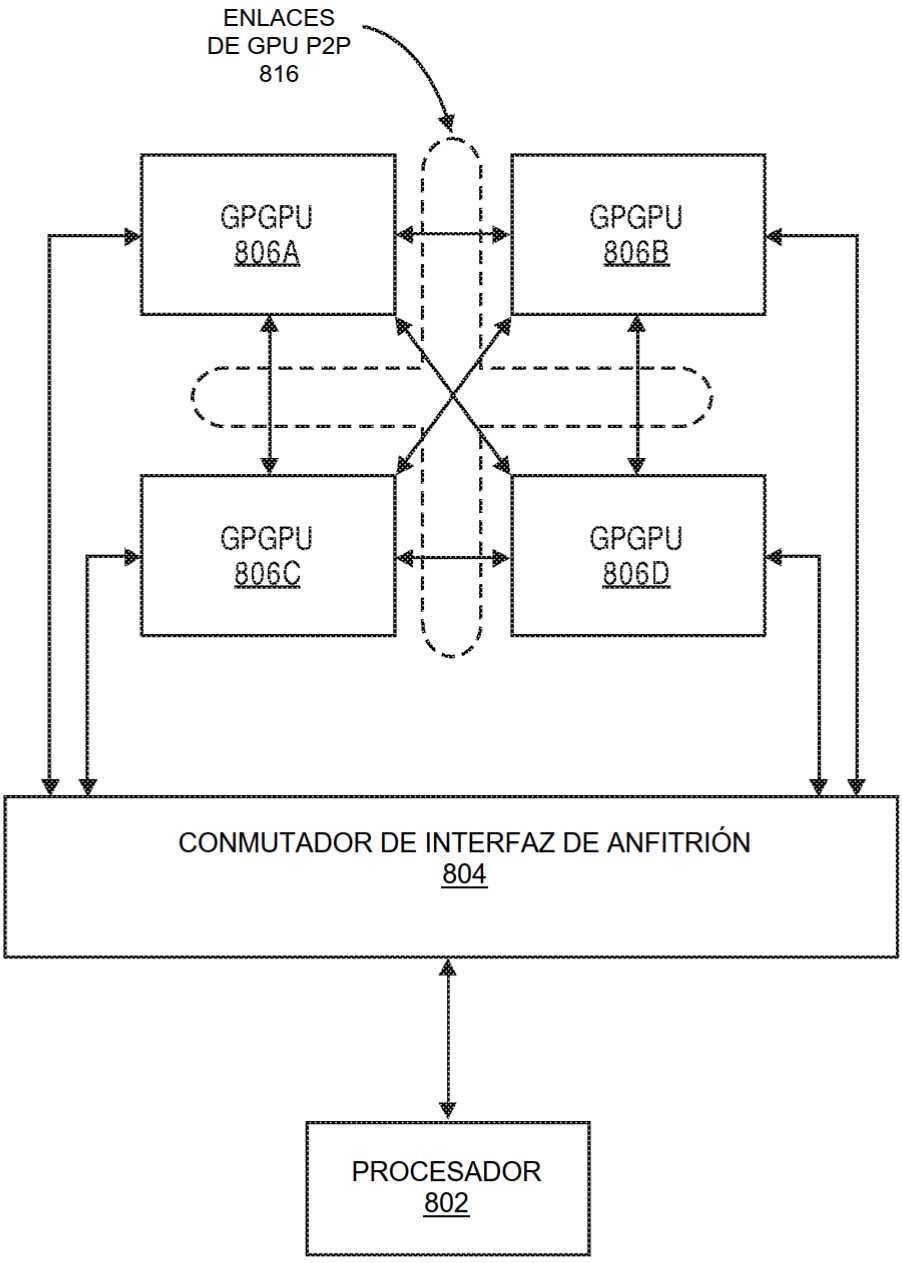

ENLACES
DE GPU P2P
816
GPGPU
806A
GPGPU
806B
GPGPU
806C
GPGPU
806D
CONMUTADOR DE INTERFAZ DE ANFITRIÓN
804
PROCESADOR
802

FIG. 8

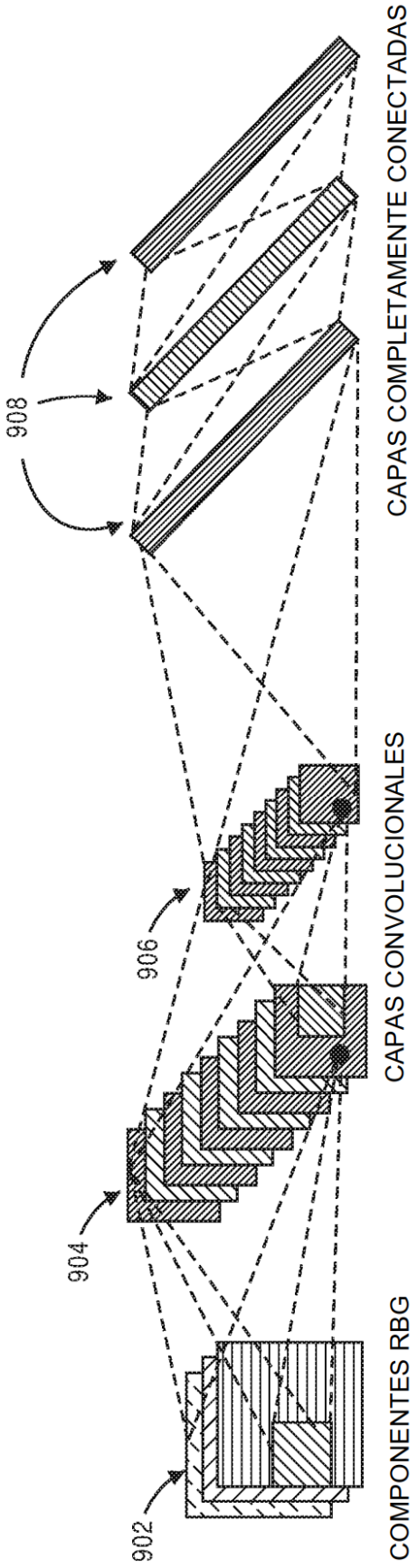

908
906
904
902
CAPAS COMPLETAMENTE CONECTADAS
CAPAS CONVOLUCIONALES
COMPONENTES RBG

FIG. 9A

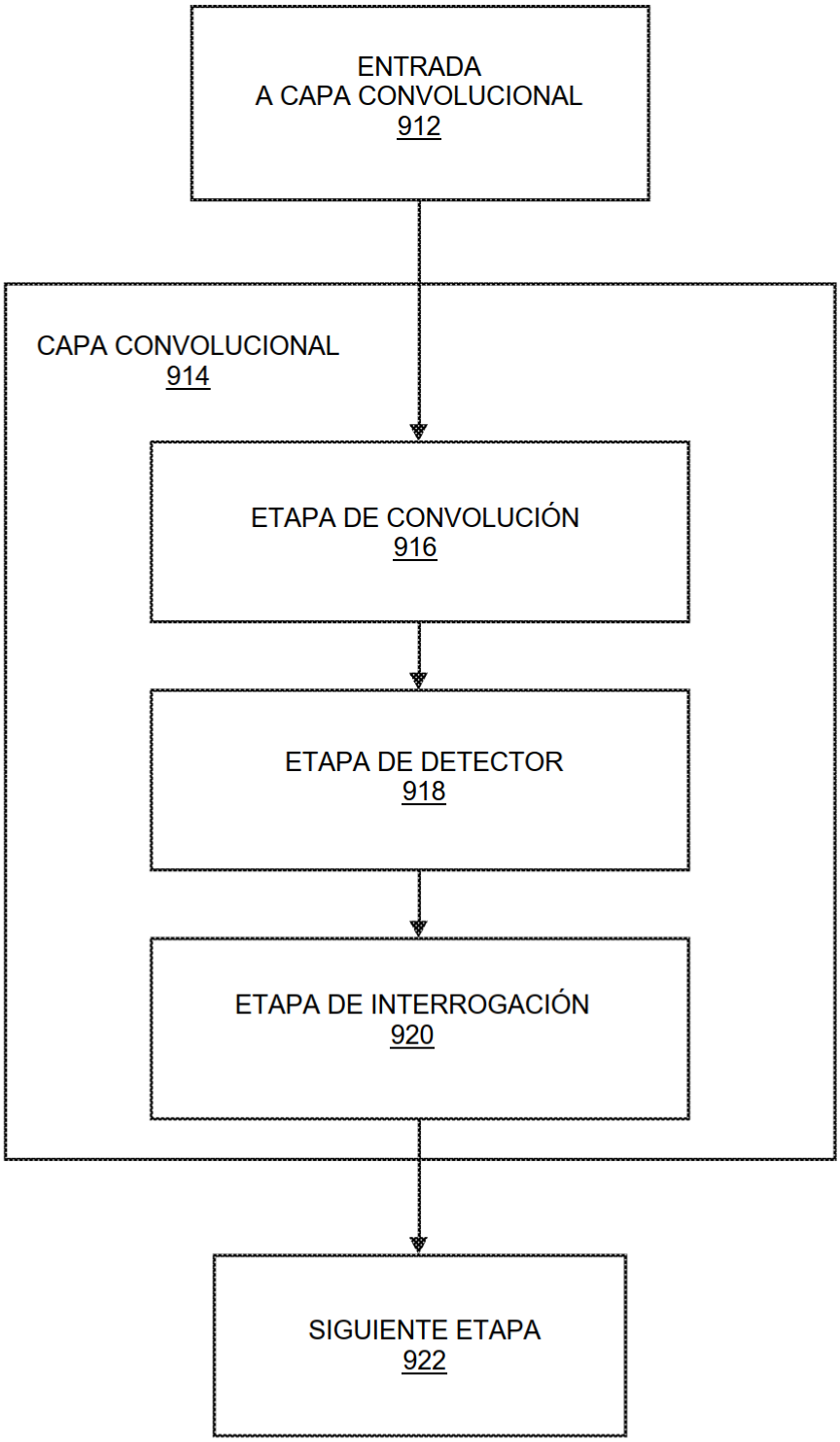

ENTRADA
A CAPA CONVOLUCIONAL
912
CAPA CONVOLUCIONAL
914
ETAPA DE CONVOLUCIÓN
916
ETAPA DE DETECTOR
918
ETAPA DE INTERROGACIÓN
920
SIGUIENTE ETAPA
922

FIG. 9B

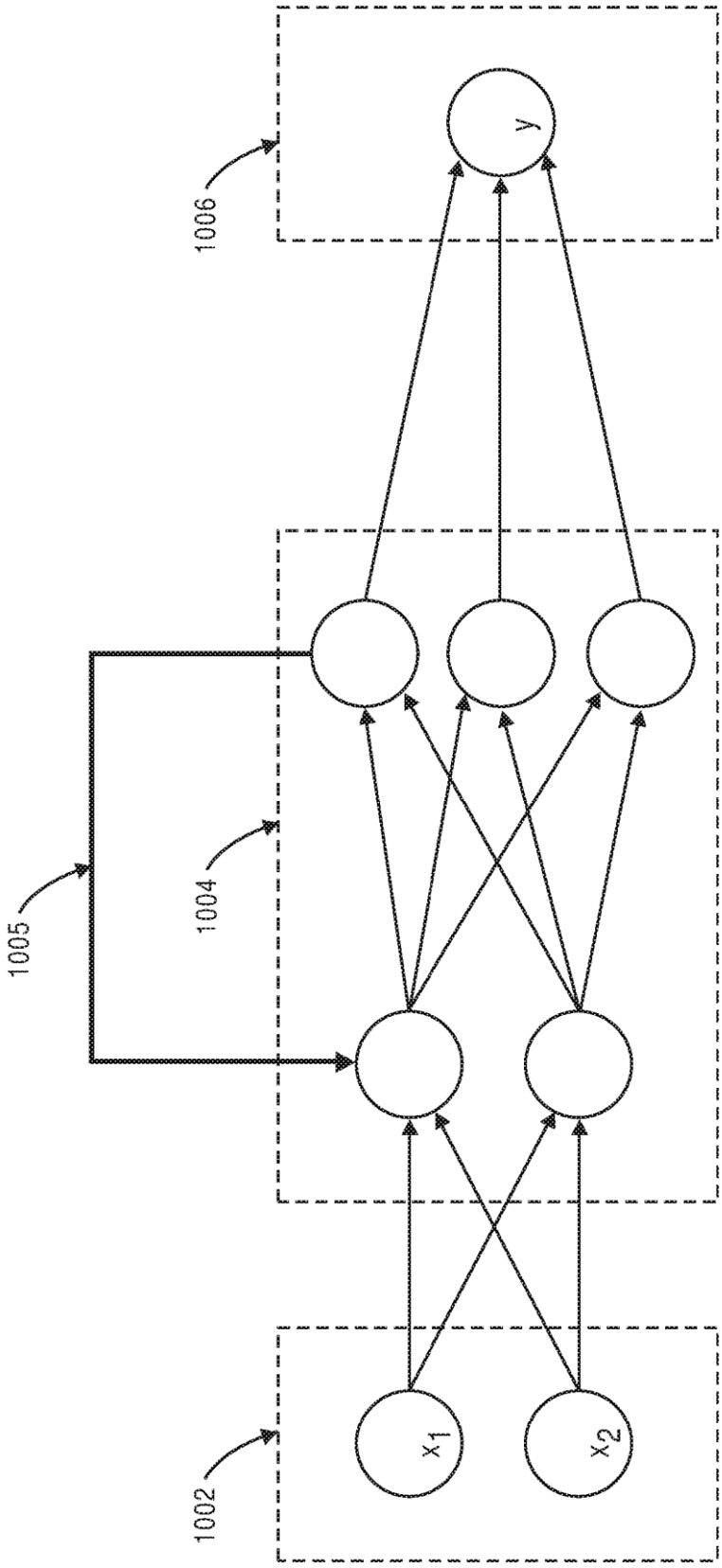


FIG. 10

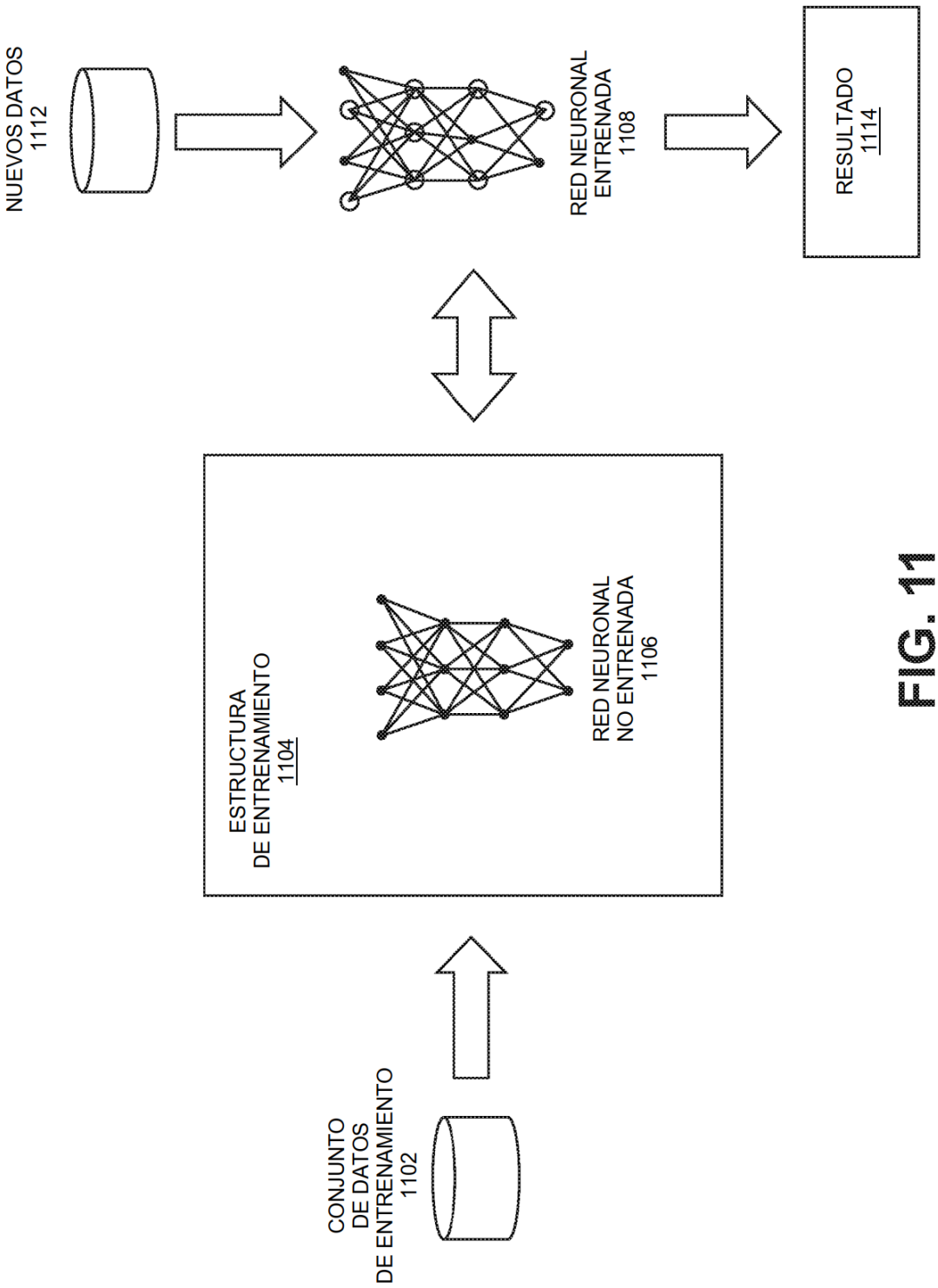

NUEVOS DATOS
1112
RED NEURONAL
ENTRENADA
1108
RESULTADO
1114
ESTRUCTURA
DE ENTRENAMIENTO
1104
RED NEURONAL
NO ENTRENADA
1106
CONJUNTO
DE DATOS
DE ENTRENAMIENTO
1102

FIG. 11

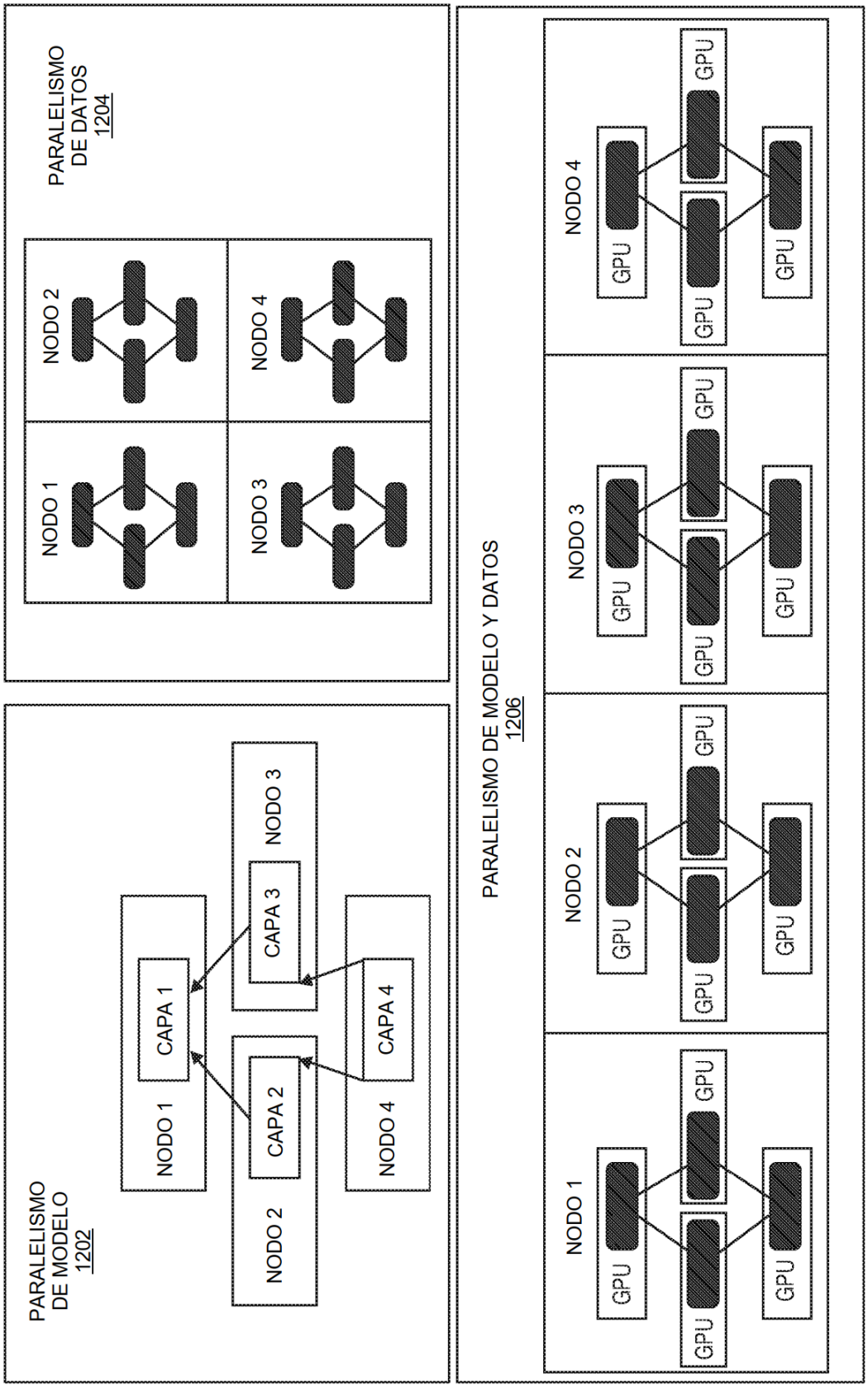

PARALELISMO DE MODELO 1202
NODO 1
CAPA 1
NODO 2
CAPA 2
NODO 3
CAPA 3
NODO 4
CAPA 4
PARALELISMO DE DATOS 1204
NODO 1
NODO 2
NODO 3
NODO 4
PARALELISMO DE MODELO Y DATOS 1206
NODO 1
NODO 2
NODO 3
NODO 4
GPU

FIG. 12

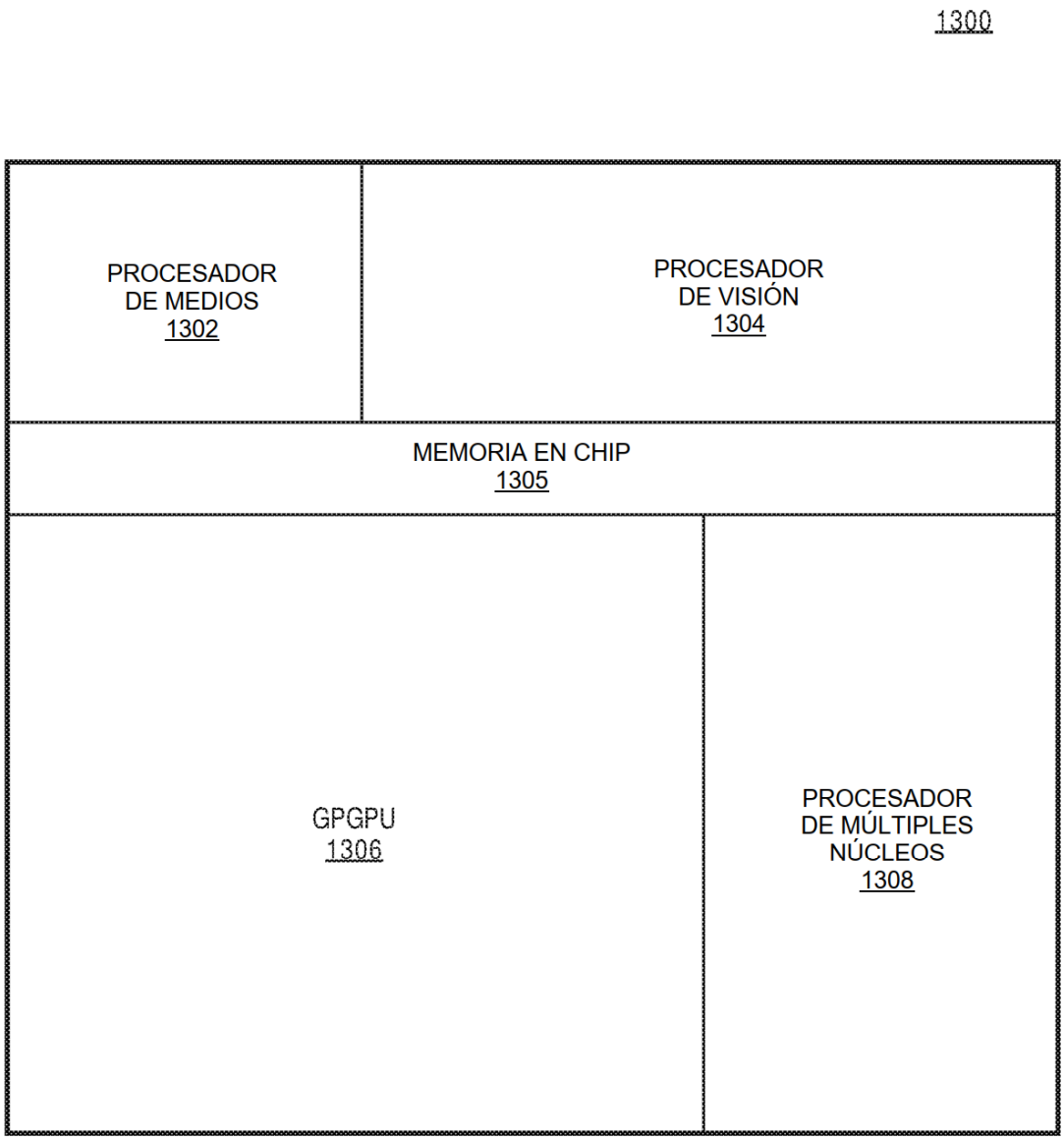

1300
PROCESADOR DE MEDIOS 1302
PROCESADOR DE VISIÓN 1304
MEMORIA EN CHIP 1305
GPGPU 1306
PROCESADOR DE MÚLTIPLES NÚCLEOS 1308

FIG. 13

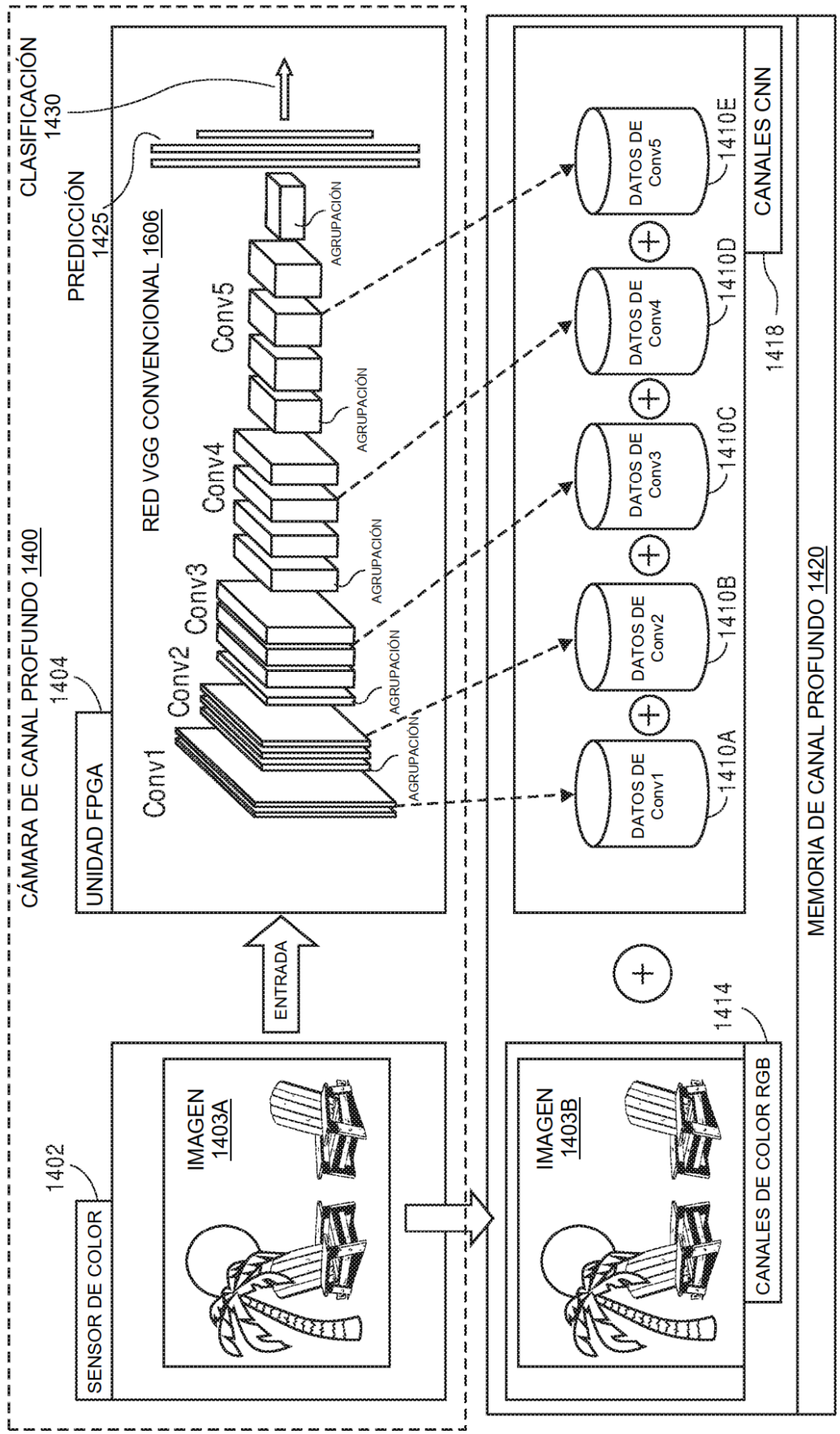

CÁMARA DE CANAL PROFUNDO 1400
CLASIFICACIÓN 1430
PREDICCIÓN 1425
UNIDAD FPGA
1404
RED VGG CONVENCIONAL 1606
Conv1
Conv2
Conv3
Conv4
Conv5
AGRUPACIÓN
AGRUPACIÓN
AGRUPACIÓN
AGRUPACIÓN
AGRUPACIÓN
ENTRADA
SENSOR DE COLOR
1402
IMAGEN 1403A
IMAGEN 1403B
CANALES DE COLOR RGB
1414
DATOS DE Conv1
1410A
DATOS DE Conv2
1410B
DATOS DE Conv3
1410C
DATOS DE Conv4
1410D
DATOS DE Conv5
1410E
CANALES CNN
1418
MEMORIA DE CANAL PROFUNDO 1420

FIG. 14

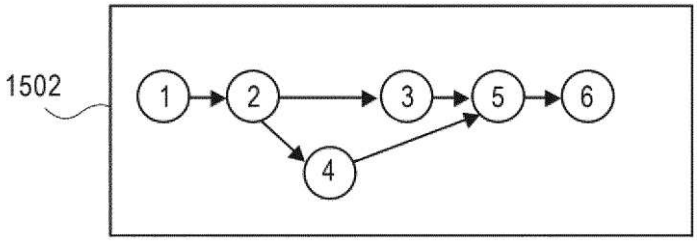


FIG. 15

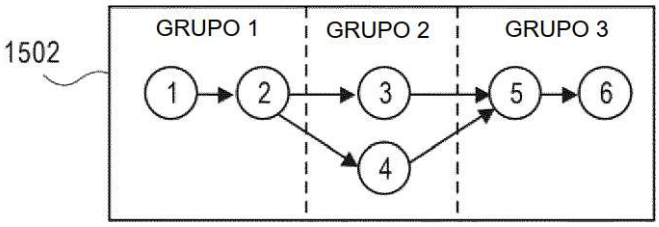


FIG. 16

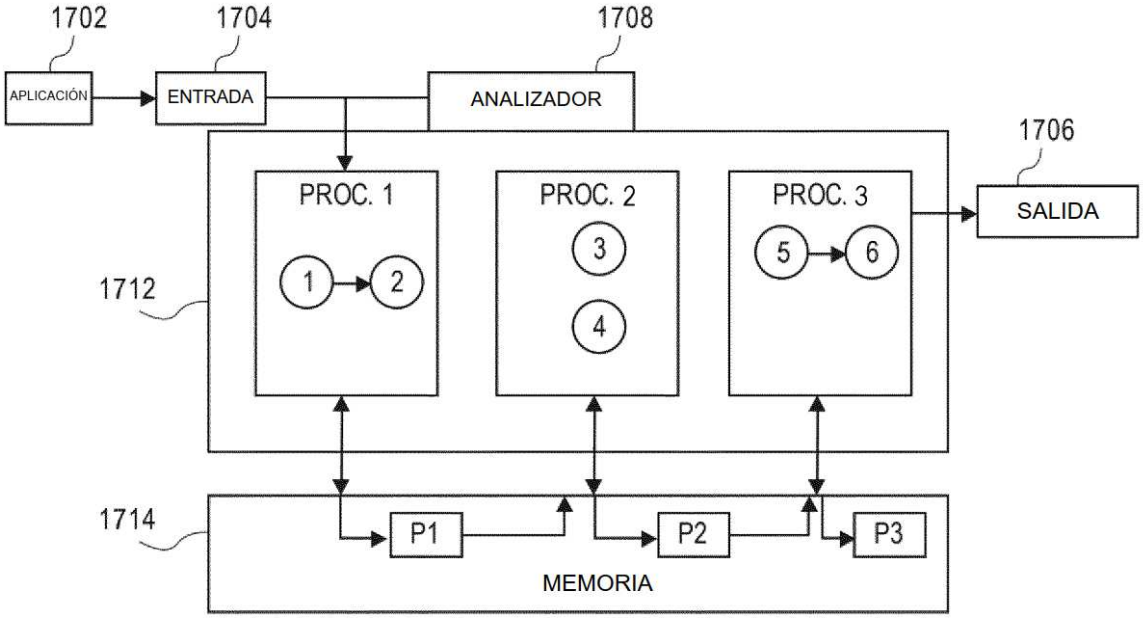


FIG. 17

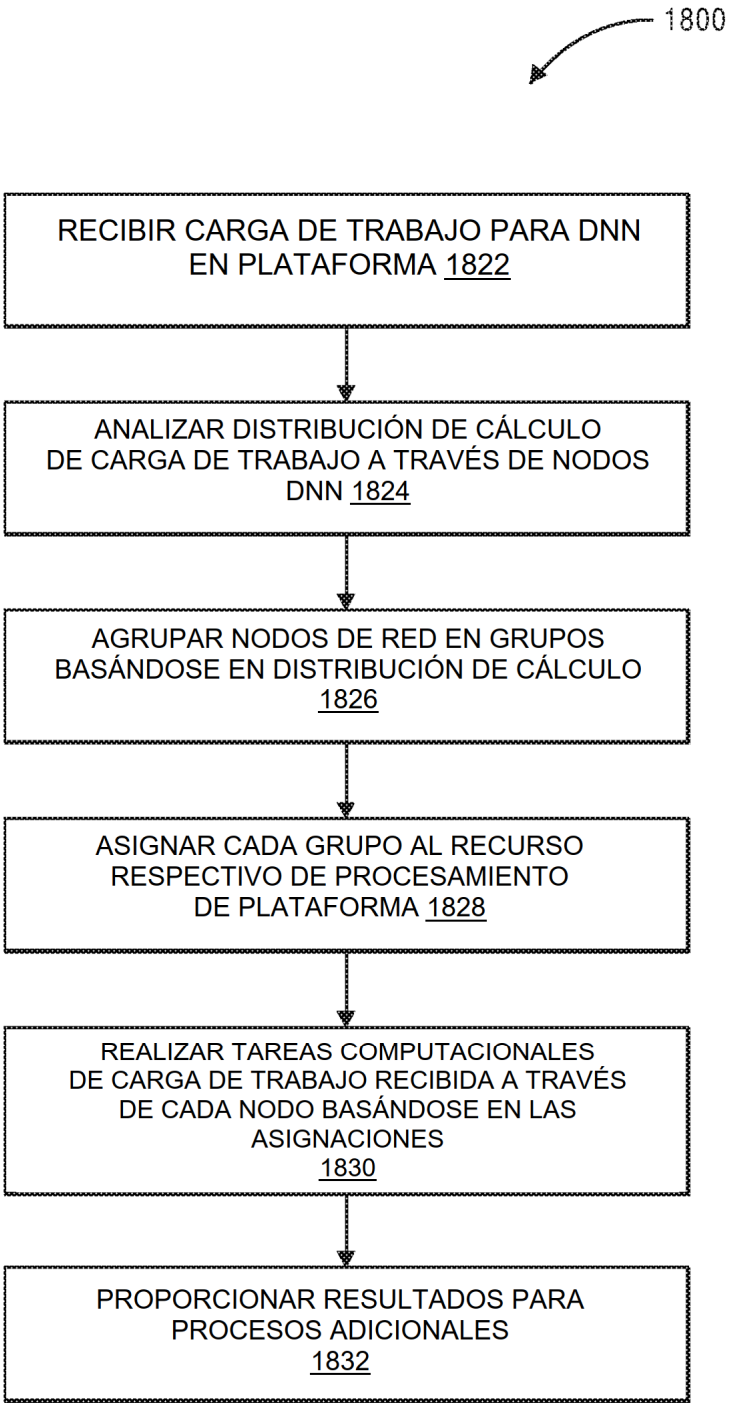

1800
RECIBIR CARGA DE TRABAJO PARA DNN EN PLATAFORMA 1822
ANALIZAR DISTRIBUCIÓN DE CÁLCULO DE CARGA DE TRABAJO A TRAVÉS DE NODOS DNN 1824
AGRUPAR NODOS DE RED EN GRUPOS BASÁNDOSE EN DISTRIBUCIÓN DE CÁLCULO 1826
ASIGNAR CADA GRUPO AL RECURSO RESPECTIVO DE PROCESAMIENTO DE PLATAFORMA 1828
REALIZAR TAREAS COMPUTACIONALES DE CARGA DE TRABAJO RECIBIDA A TRAVÉS DE CADA NODO BASÁNDOSE EN LAS ASIGNACIONES 1830
PROPORCIONAR RESULTADOS PARA PROCESOS ADICIONALES 1832

FIG. 18

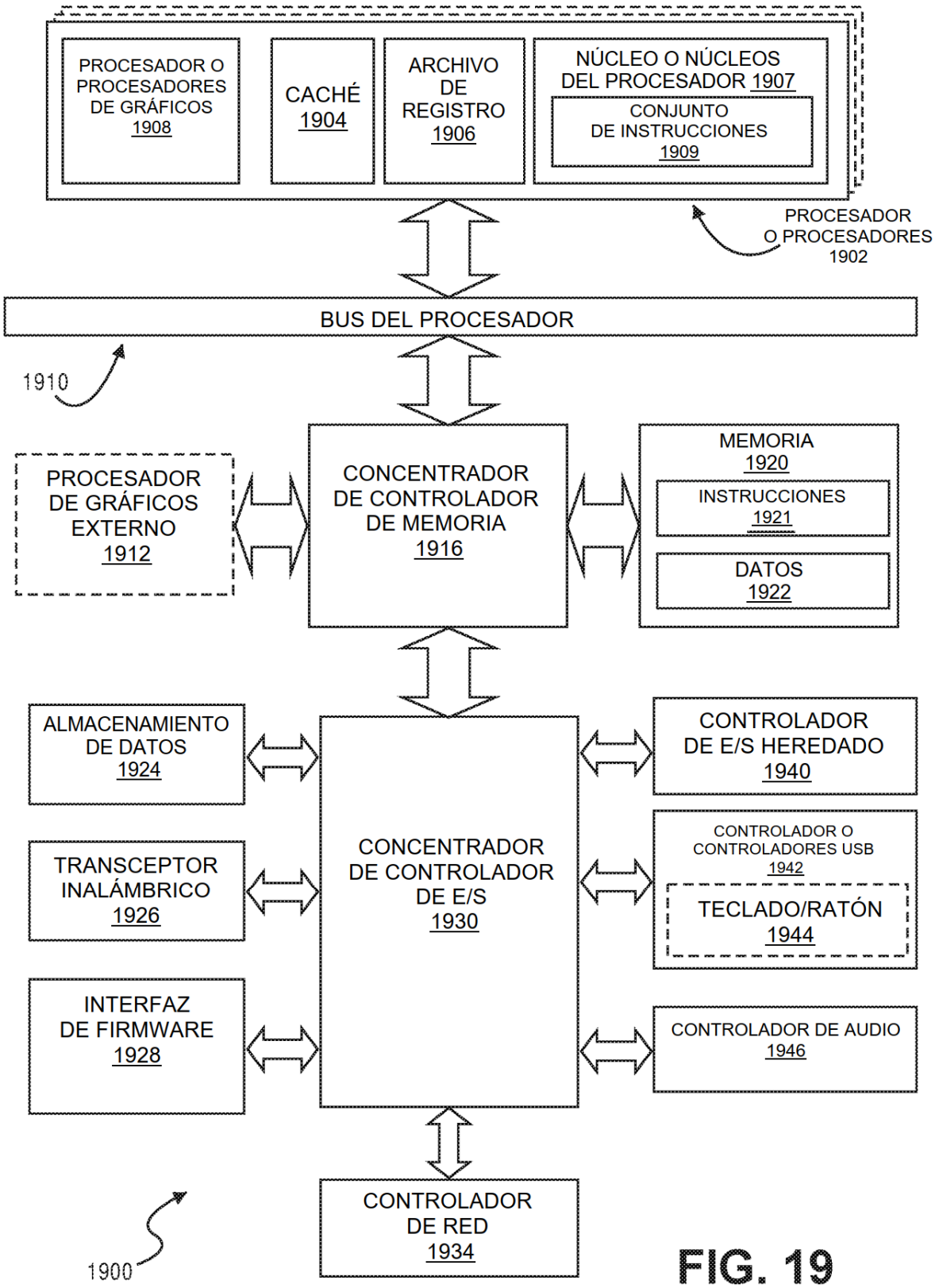

PROCESADOR O PROCESADORES DE GRÁFICOS 1908
CACHÉ 1904
ARCHIVO DE REGISTRO 1906
NÚCLEO O NÚCLEOS DEL PROCESADOR 1907
CONJUNTO DE INSTRUCCIONES 1909
PROCESADOR O PROCESADORES 1902
BUS DEL PROCESADOR
1910
PROCESADOR DE GRÁFICOS EXTERNO 1912
CONCENTRADOR DE CONTROLADOR DE MEMORIA 1916
MEMORIA 1920
INSTRUCCIONES 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 O CONTROLADORES USB 1942
TECLADO/RATÓN 1944
CONTROLADOR DE AUDIO 1946
CONTROLADOR DE RED 1934
1900

FIG. 19

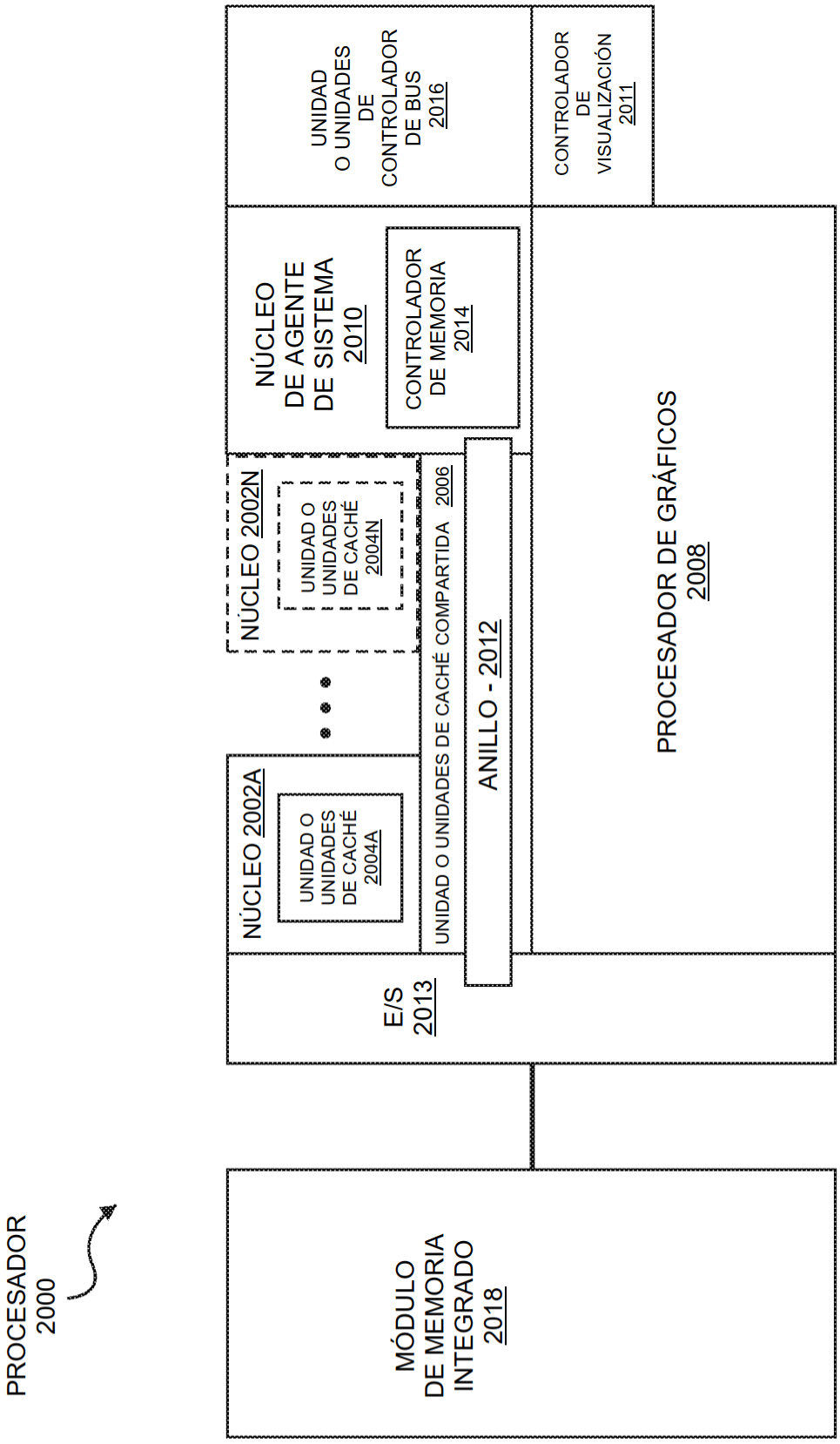

PROCESADOR
2000
MÓDULO
DE MEMORIA
INTEGRADO
2018
E/S
2013
NÚCLEO 2002A
UNIDAD O
UNIDADES
DE CACHÉ
2004A
NÚCLEO 2002N
UNIDAD O
UNIDADES
DE CACHÉ
2004N
UNIDAD O UNIDADES DE CACHÉ COMPARTIDA 2006
ANILLO - 2012
NÚCLEO
DE AGENTE
DE SISTEMA
2010
CONTROLADOR
DE MEMORIA
2014
UNIDAD
O UNIDADES
DE
CONTROLADOR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

FIG. 20

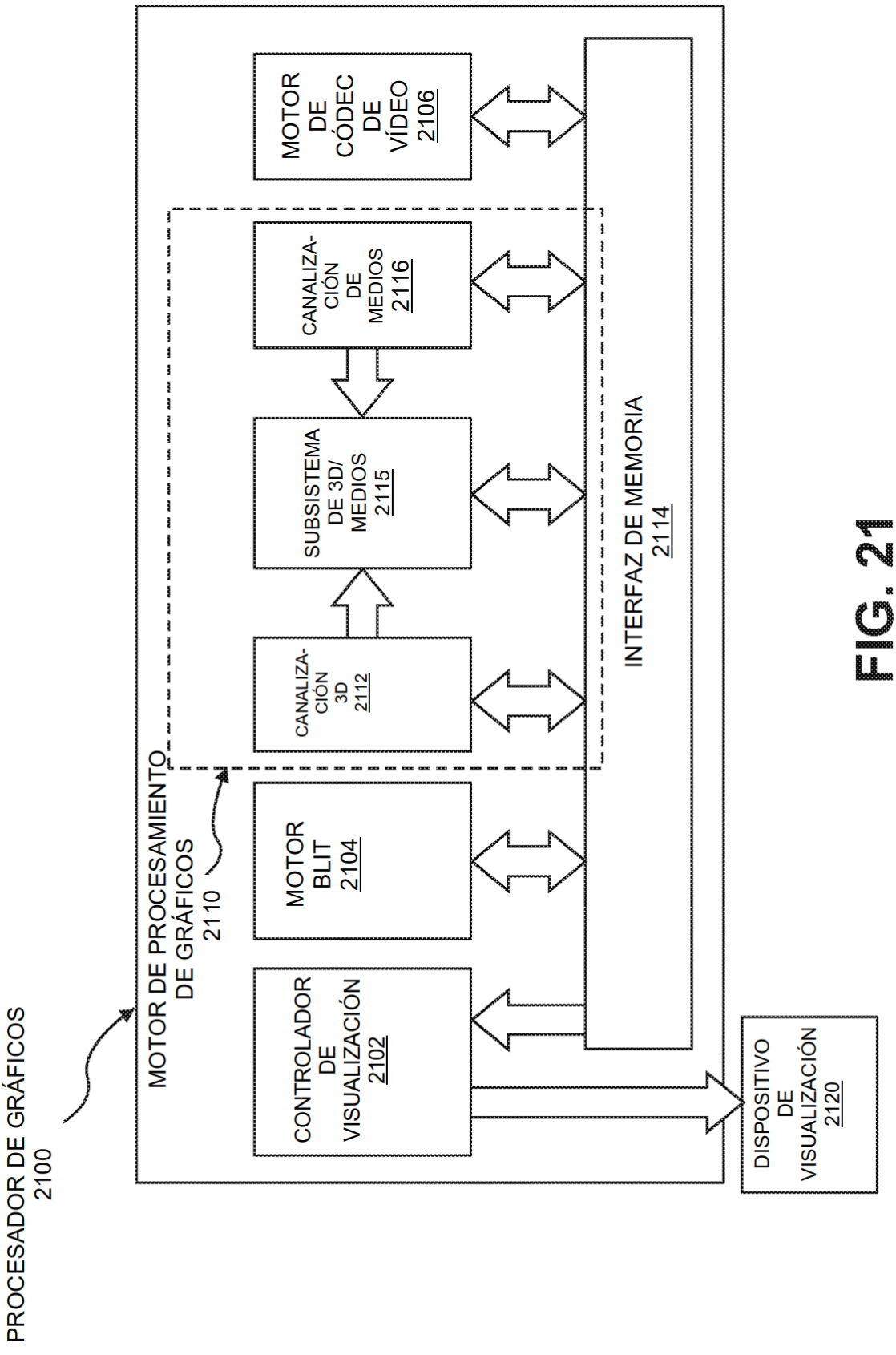

PROCESADOR DE GRÁFICOS
2100
MOTOR DE PROCESAMIENTO DE GRÁFICOS
2110
CONTROLADOR DE VISUALIZACIÓN
2102
MOTOR BLIT
2104
CANALIZA-CIÓN 3D
2112
SUBSISTEMA DE 3D/ MEDIOS
2115
CANALIZA-CIÓN DE MEDIOS
2116
MOTOR DE CÓDEC DE VÍDEO
2106
INTERFAZ DE MEMORIA
2114
DISPOSITIVO DE VISUALIZACIÓN
2120

FIG. 21

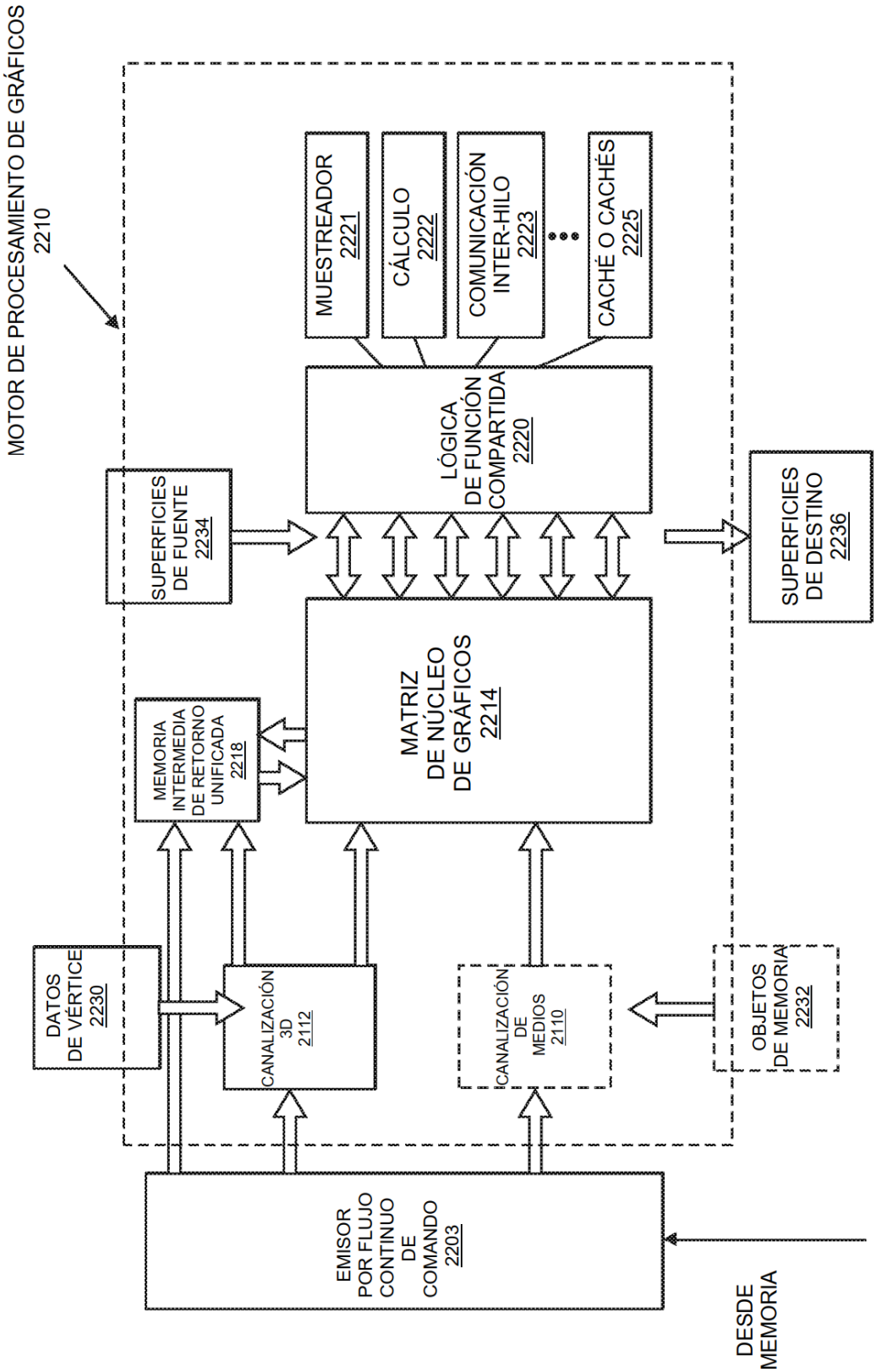

MOTOR DE PROCESAMIENTO DE GRÁFICOS
2210
MUESTREADOR
2221
CÁLCULO
2222
COMUNICACIÓN INTER-HILO
2223
CACHÉ O CACHÉS
2225
LÓGICA DE FUNCIÓN COMPARTIDA
2220
SUPERFICIES DE FUENTE
2234
SUPERFICIES DE DESTINO
2236
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
2218
MATRIZ DE NÚCLEO DE GRÁFICOS
2214
DATOS DE VÉRTICE
2230
CANALIZACIÓN 3D
2112
CANALIZACIÓN DE MEDIOS
2110
OBJETOS DE MEMORIA
2232
EMISOR POR FLUJO CONTINUO DE COMANDO
2203
DESDE MEMORIA

FIG. 22

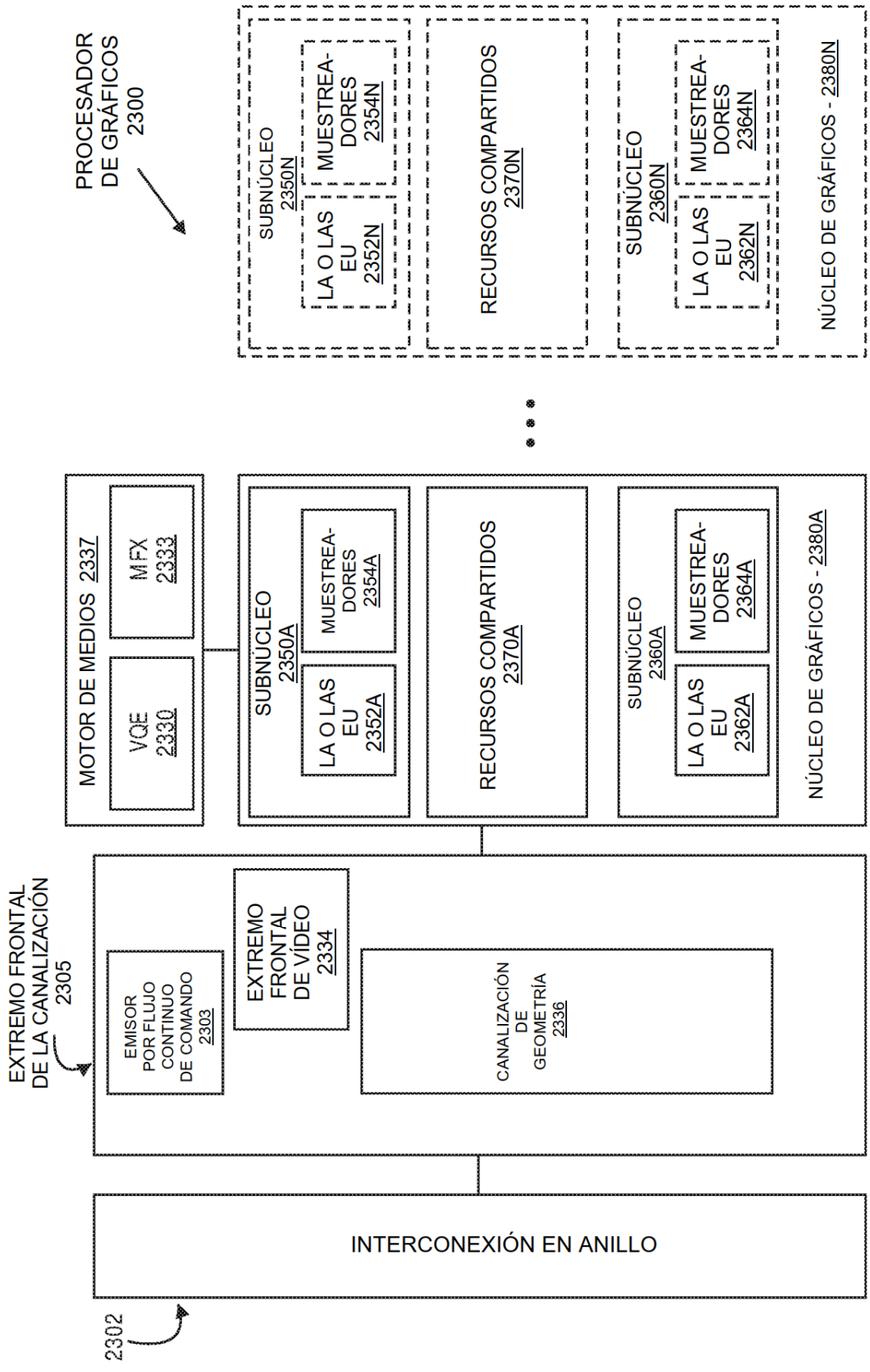

PROCESADOR DE GRÁFICOS 2300
2302
INTERCONEXIÓN EN ANILLO
EXTREMO FRONTAL DE LA CANALIZACIÓN 2305
EMISOR POR FLUJO CONTINUO DE COMANDO 2303
EXTREMO FRONTAL DE VÍDEO 2334
CANALIZACIÓN DE GEOMETRÍA 2336
MOTOR DE MEDIOS 2337
VQE 2330
MFX 2333
NÚCLEO DE GRÁFICOS - 2380A
SUBNÚCLEO 2350A
LA O LAS EU 2352A
MUESTREA-DORES 2354A
RECURSOS COMPARTIDOS 2370A
SUBNÚCLEO 2360A
LA O LAS EU 2362A
MUESTREA-DORES 2364A
SUBNÚCLEO 2350N
LA O LAS EU 2352N
MUESTREA-DORES 2354N
RECURSOS COMPARTIDOS 2370N
SUBNÚCLEO 2360N
LA O LAS EU 2362N
MUESTREA-DORES 2364N
NÚCLEO DE GRÁFICOS - 2380N

FIG. 23

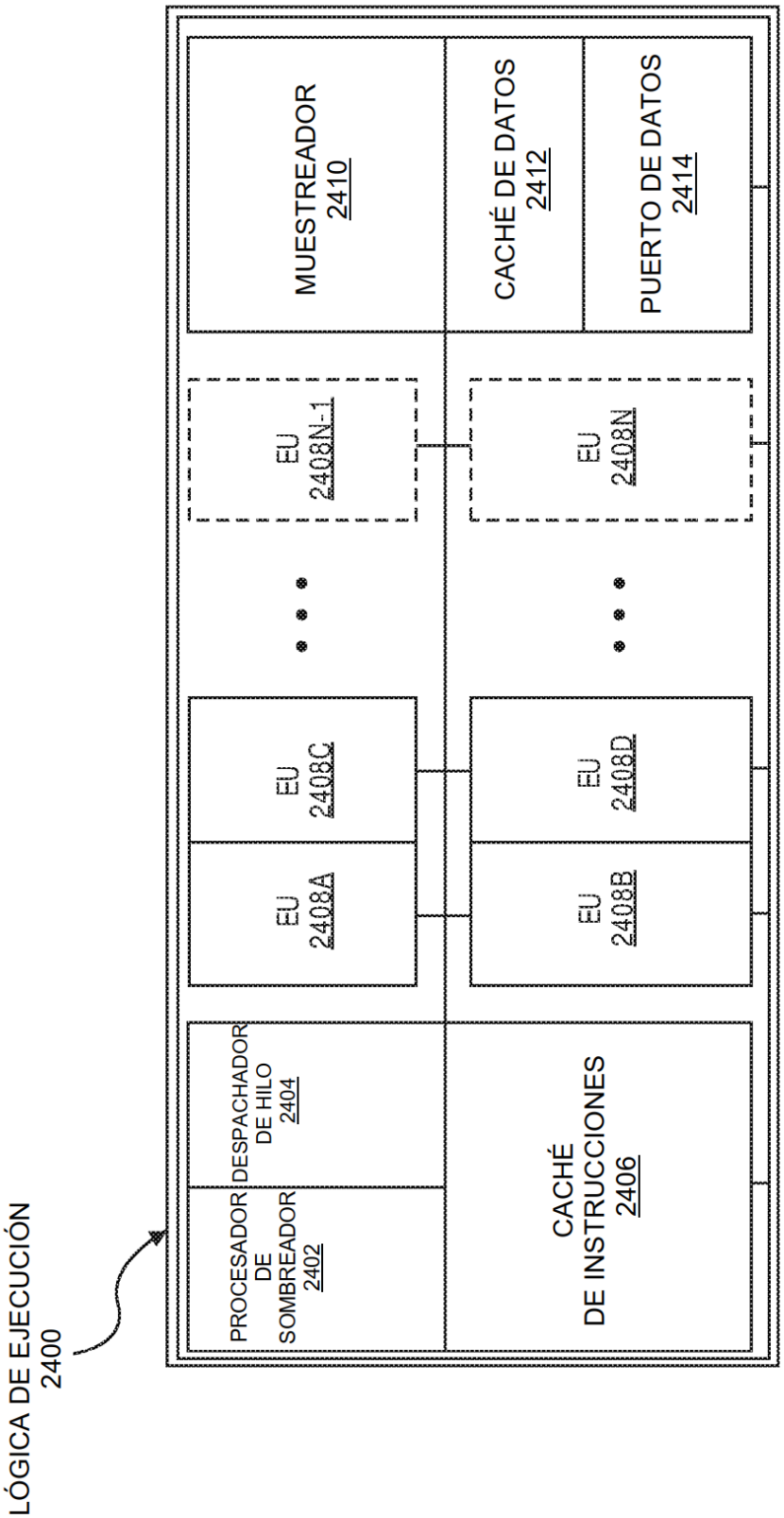

LÓGICA DE EJECUCIÓN
2400
PROCESADOR DE SOMBREADOR
2402
DESPACHADOR DE HILO
2404
CACHÉ DE INSTRUCCIONES
2406
EU
2408A
EU
2408C
EU
2408B
EU
2408D
EU
2408N-1
EU
2408N
MUESTREADOR
2410
CACHÉ DE DATOS
2412
PUERTO DE DATOS
2414

FIG. 24

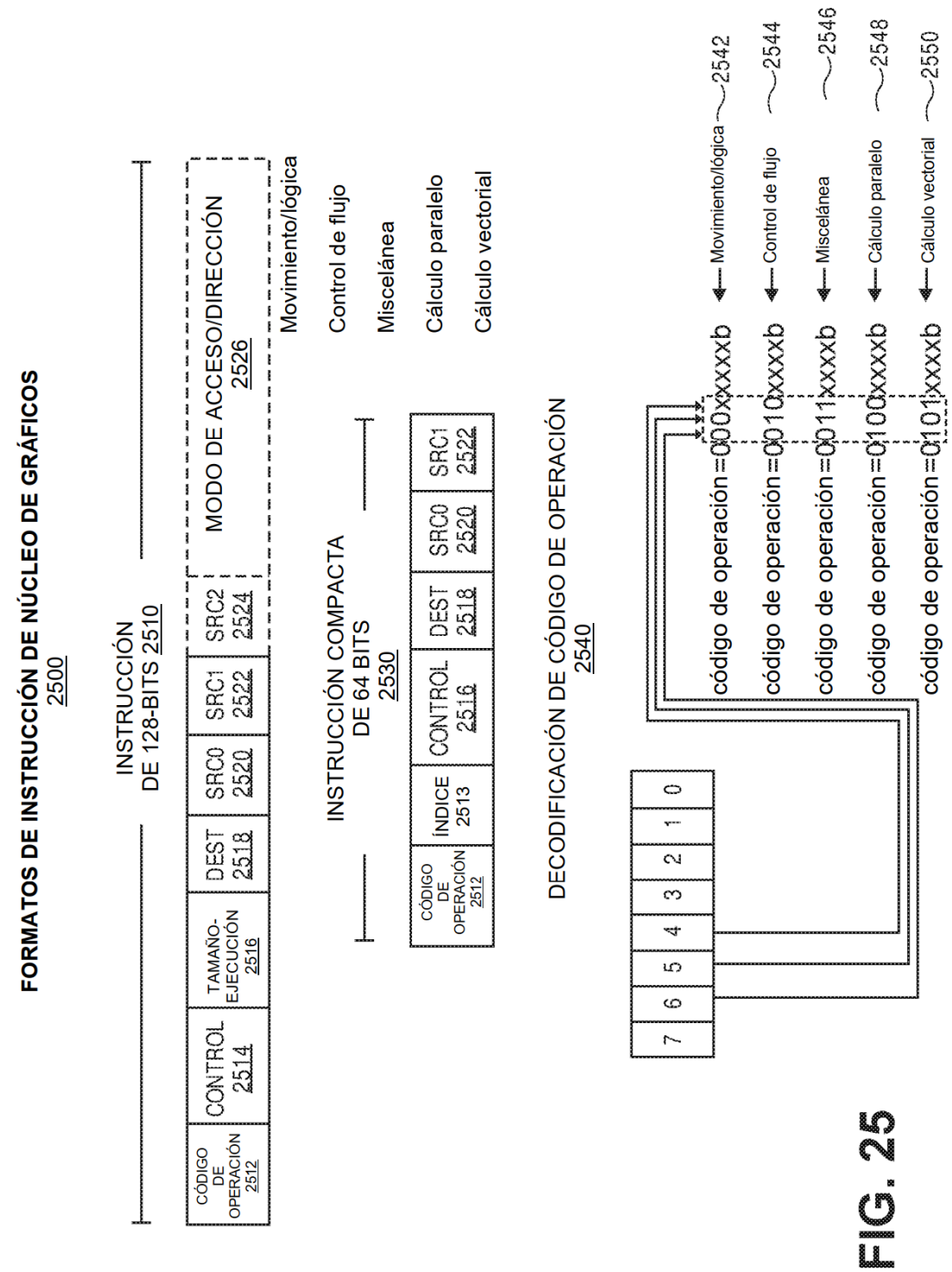

FORMATOS DE INSTRUCCIÓN DE NÚCLEO DE GRÁFICOS
2500
INSTRUCCIÓN DE 128-BITS 2510
CÓDIGO DE OPERACIÓN 2512
CONTROL 2514
TAMAÑO-EJECUCIÓN 2516
DEST 2518
SRC0 2520
SRC1 2522
SRC2 2524
MODO DE ACCESO/DIRECCIÓN 2526
Movimiento/lógica
INSTRUCCIÓN COMPACTA DE 64 BITS 2530
CÓDIGO DE OPERACIÓN 2512
ÍNDICE 2513
CONTROL 2516
DEST 2518
SRC0 2520
SRC1 2522
Control de flujo
Miscelánea
Cálculo paralelo
Cálculo vectorial
DECODIFICACIÓN DE CÓDIGO DE OPERACIÓN 2540
7 6 5 4 3 2 1 0
código de operación = 000xxxxb
código de operación = 001xxxxb
código de operación = 010xxxxb
código de operación = 011xxxxb
código de operación = 100xxxxb
código de operación = 101xxxxb
Movimiento/lógica 2542
Control de flujo 2544
Miscelánea 2546
Cálculo paralelo 2548
Cálculo vectorial 2550

FIG. 25

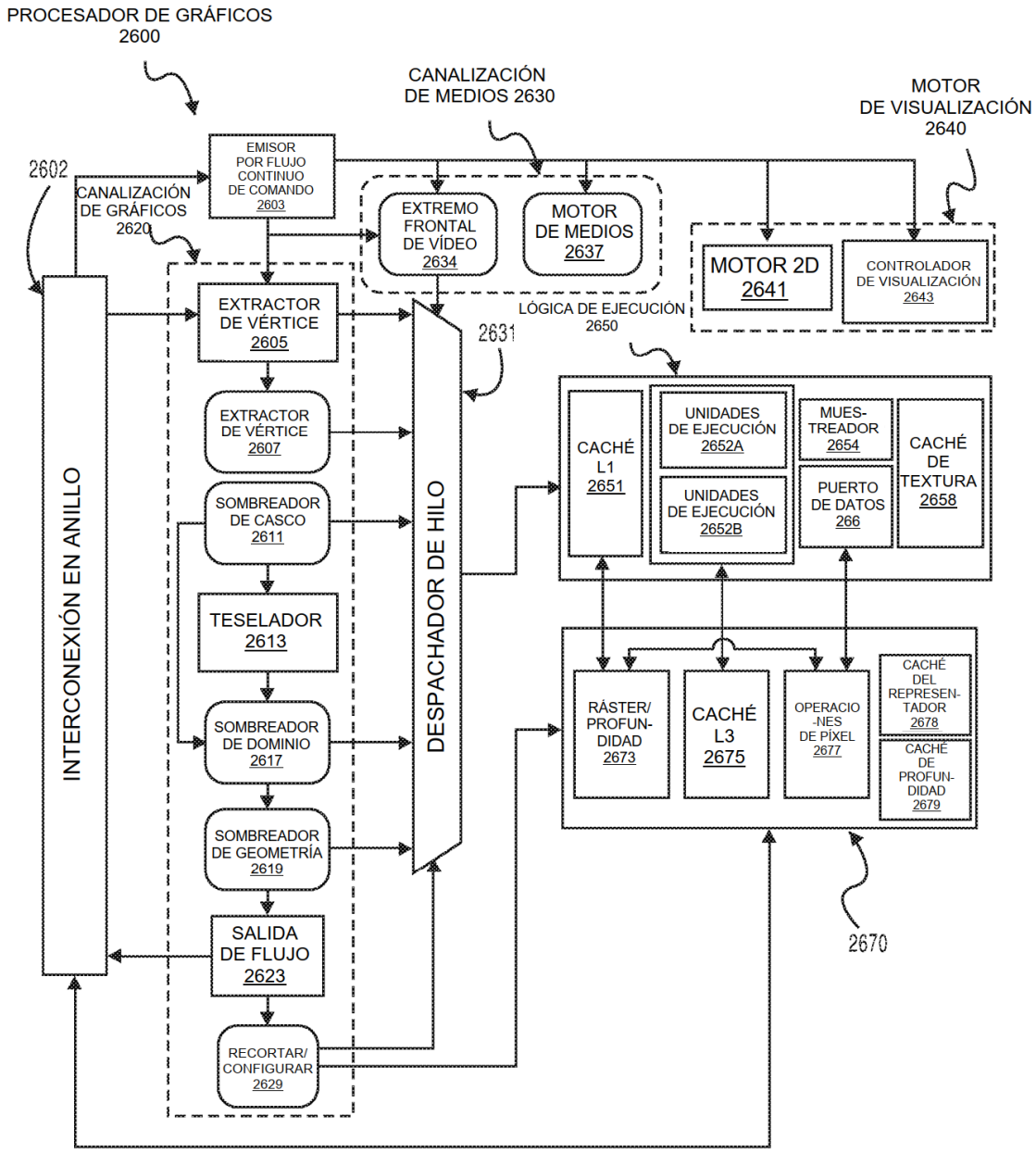

PROCESADOR DE GRÁFICOS 2600
CANALIZACIÓN DE MEDIOS 2630
MOTOR DE VISUALIZACIÓN 2640
2602
CANALIZACIÓN DE GRÁFICOS 2620
EMISOR POR FLUJO CONTINUO DE COMANDO 2603
EXTREMO FRONTAL DE VÍDEO 2634
MOTOR DE MEDIOS 2637
MOTOR 2D 2641
CONTROLADOR DE VISUALIZACIÓN 2643
LÓGICA DE EJECUCIÓN 2650
2631
EXTRACTOR DE VÉRTICE 2605
EXTRACTOR DE VÉRTICE 2607
SOMBREADOR DE CASCO 2611
TESELADOR 2613
SOMBREADOR DE DOMINIO 2617
SOMBREADOR DE GEOMETRÍA 2619
SALIDA DE FLUJO 2623
RECORTAR/ CONFIGURAR 2629
INTERCONEXIÓN EN ANILLO
DESPACHADOR DE HILO
CACHÉ L1 2651
UNIDADES DE EJECUCIÓN 2652A
UNIDADES DE EJECUCIÓN 2652B
MUES-TREADOR 2654
PUERTO DE DATOS 266
CACHÉ DE TEXTURA 2658
RÁSTER/ PROFUN-DIDAD 2673
CACHÉ L3 2675
OPERACIO-NES DE PÍXEL 2677
CACHÉ DEL REPRESEN-TADOR 2678
CACHÉ DE PROFUN-DIDAD 2679
2670

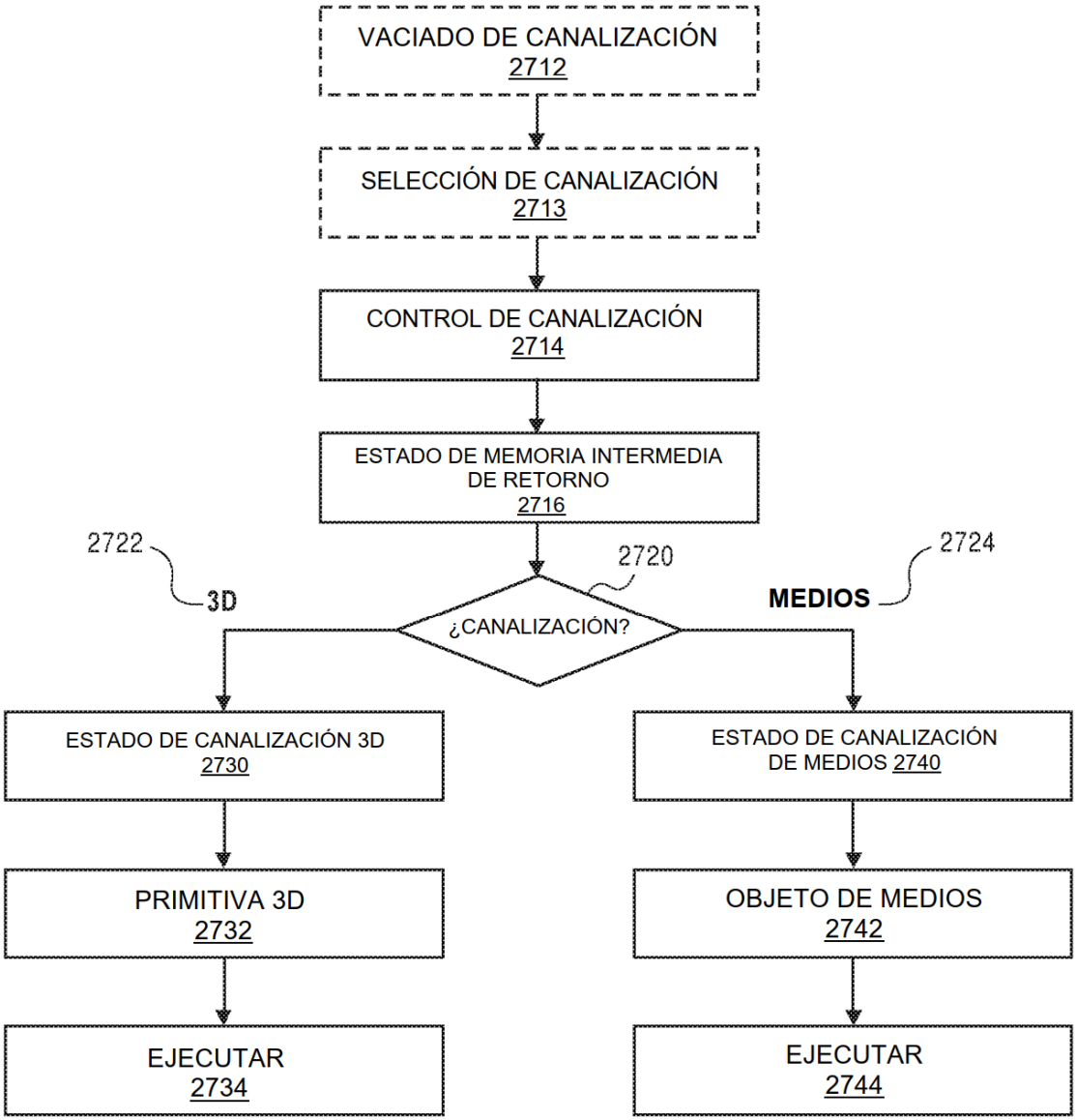
FIG. 26

FIG. 27A

FORMATO DE COMANDO DE MUESTRA
2700

FIG. 27B

SECUENCIA DE COMANDOS DE MUESTRA
2710



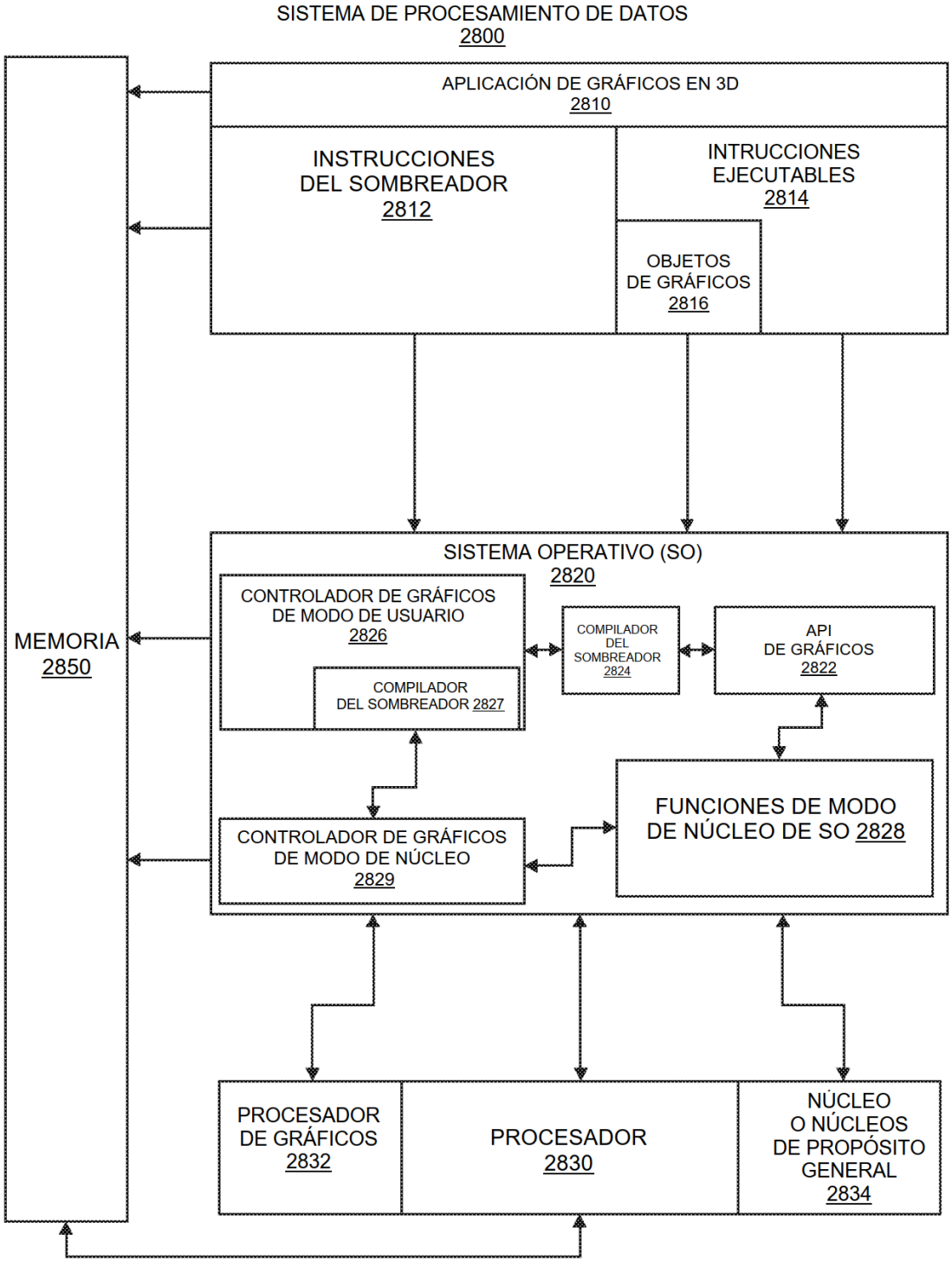

SISTEMA DE PROCESAMIENTO DE DATOS
2800
APLICACIÓN DE GRÁFICOS EN 3D
2810
INSTRUCCIONES DEL SOMBREADOR
2812
INTRUCCIONES EJECUTABLES
2814
OBJETOS DE GRÁFICOS
2816
MEMORIA
2850
SISTEMA OPERATIVO (SO)
2820
CONTROLADOR DE GRÁFICOS DE MODO DE USUARIO
2826
COMPILADOR DEL SOMBREADOR 2827
COMPILADOR DEL SOMBREADOR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

FIG. 28

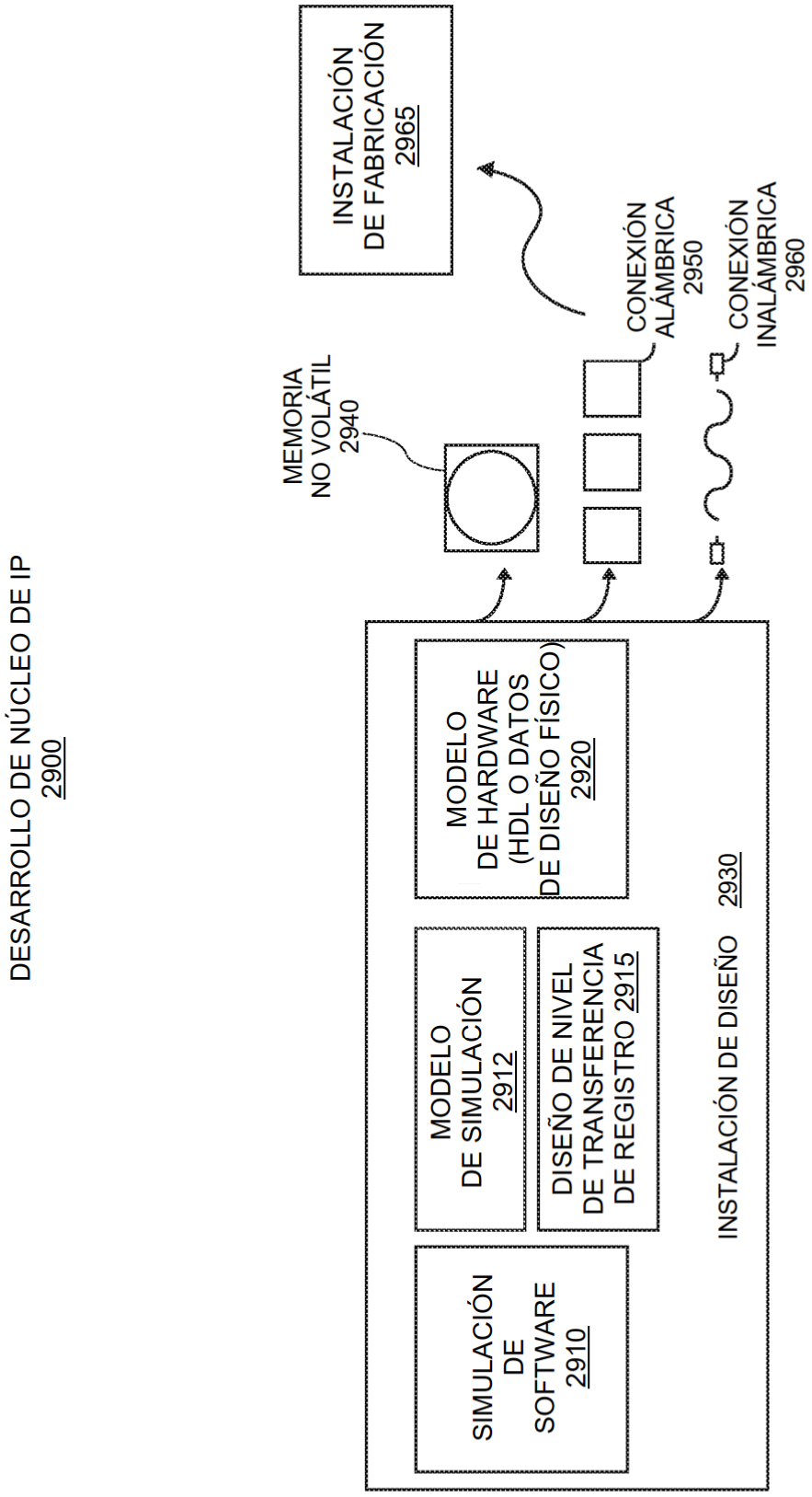

DESARROLLO DE NÚCLEO DE IP
2900
SIMULACIÓN DE SOFTWARE 2910
MODELO DE SIMULACIÓN 2912
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 2915
MODELO DE HARDWARE (HDL O DATOS DE DISEÑO FÍSICO) 2920
INSTALACIÓN DE DISEÑO 2930
MEMORIA NO VOLÁTIL 2940
CONEXIÓN ALÁMBRICA 2950
CONEXIÓN INALÁMBRICA 2960
INSTALACIÓN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

FIG. 29

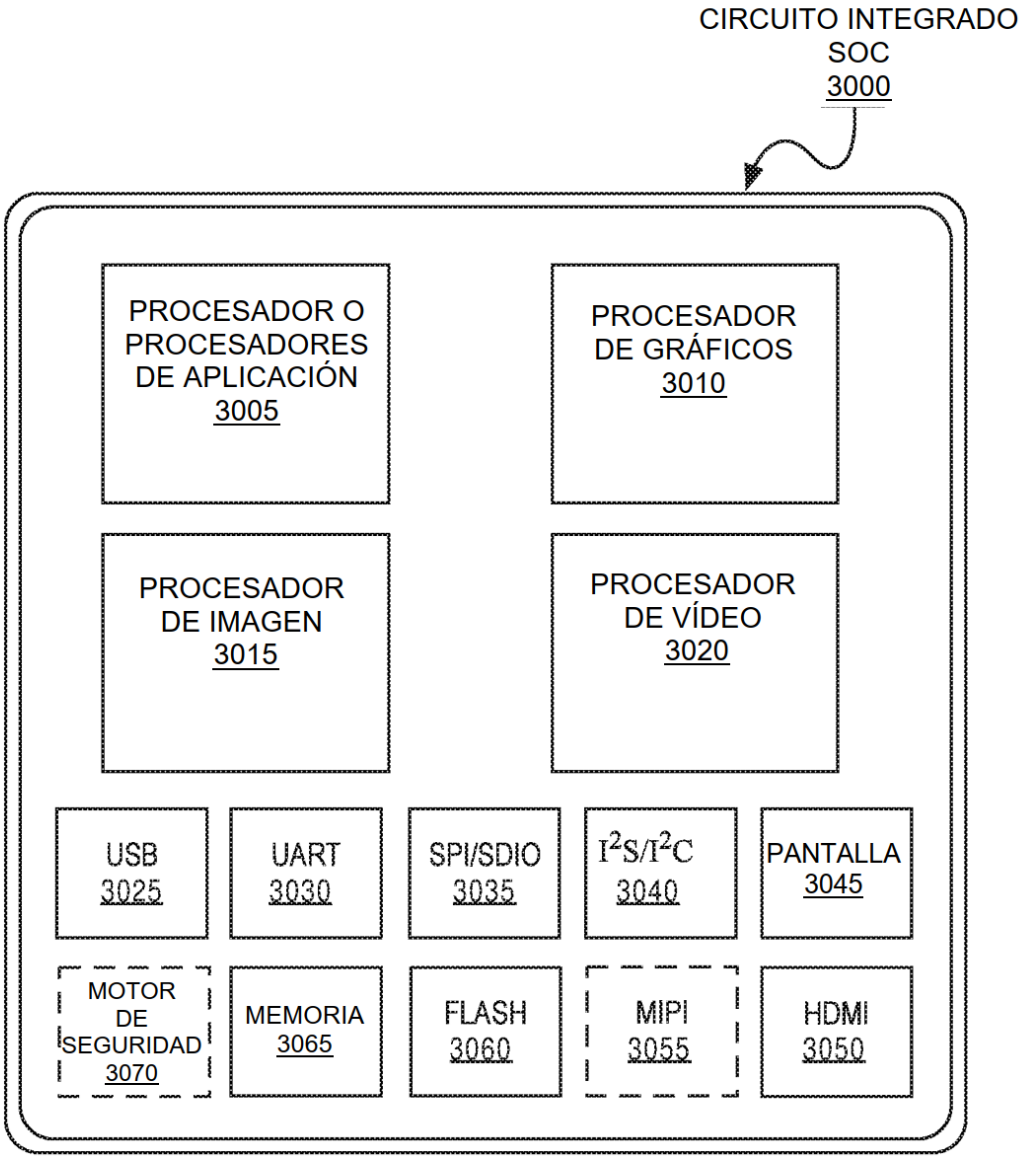

CIRCUITO INTEGRADO SOC
3000
PROCESADOR O PROCESADORES DE APLICACIÓN
3005
PROCESADOR DE GRÁFICOS
3010
PROCESADOR DE IMAGEN
3015
PROCESADOR DE VÍDEO
3020
USB
3025
UART
3030
SPI/SDIO
3035
I2S/I2C
3040
PANTALLA
3045
MOTOR DE SEGURIDAD
3070
MEMORIA
3065
FLASH
3060
MIPI
3055
HDMI
3050

FIG. 30

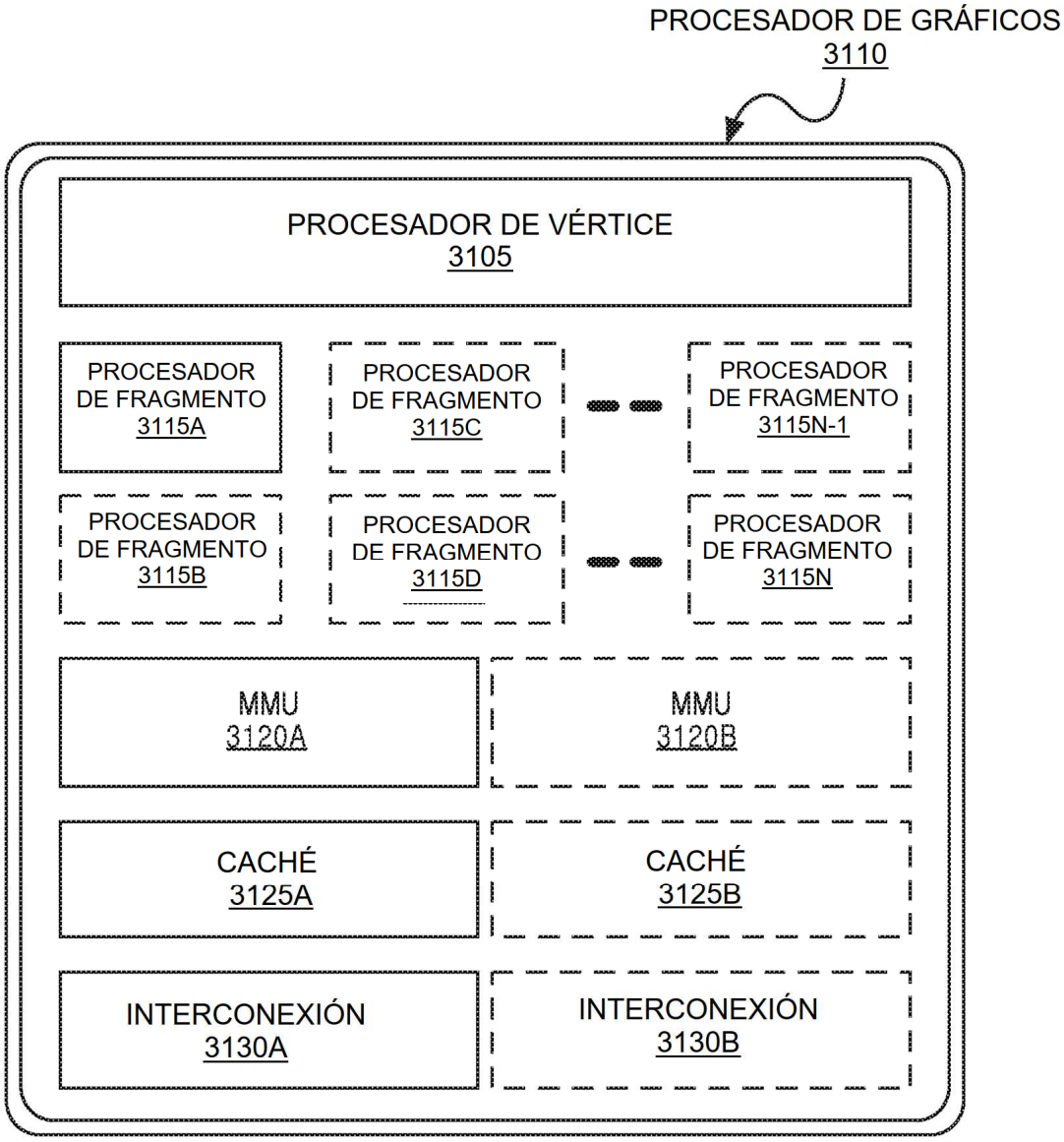

PROCESADOR DE GRÁFICOS
3110
PROCESADOR DE VÉRTICE
3105
PROCESADOR DE FRAGMENTO
3115A
PROCESADOR DE FRAGMENTO
3115C
PROCESADOR DE FRAGMENTO
3115N-1
PROCESADOR DE FRAGMENTO
3115B
PROCESADOR DE FRAGMENTO
3115D
PROCESADOR DE FRAGMENTO
3115N
MMU
3120A
MMU
3120B
CACHÉ
3125A
CACHÉ
3125B
INTERCONEXIÓN
3130A
INTERCONEXIÓN
3130B

FIG. 31

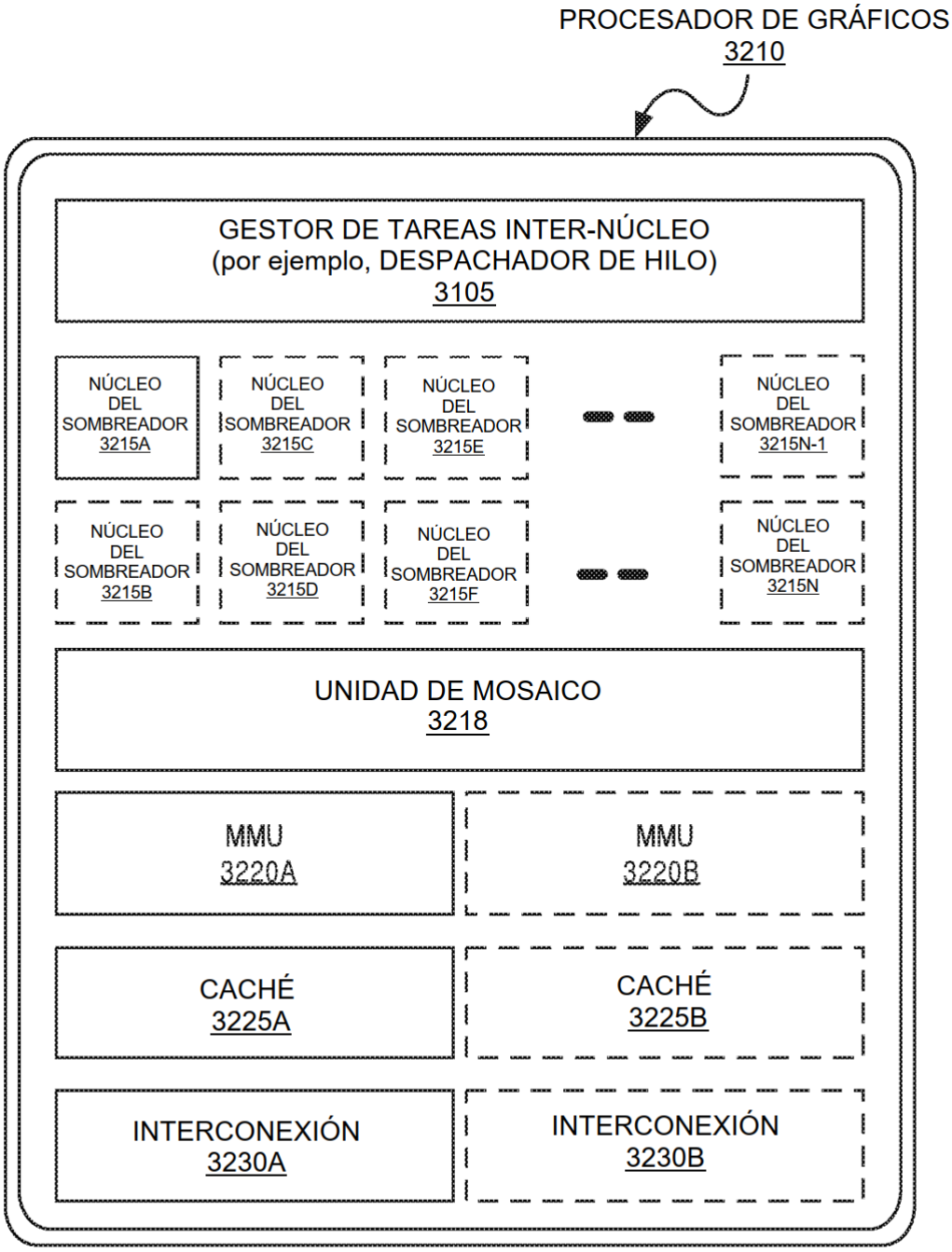

PROCESADOR DE GRÁFICOS
3210
GESTOR DE TAREAS INTER-NÚCLEO
(por ejemplo, DESPACHADOR DE HILO)
3105
NÚCLEO DEL SOMBREADOR 3215A
NÚCLEO DEL SOMBREADOR 3215C
NÚCLEO DEL SOMBREADOR 3215E
NÚCLEO DEL SOMBREADOR 3215N-1
NÚCLEO DEL SOMBREADOR 3215B
NÚCLEO DEL SOMBREADOR 3215D
NÚCLEO DEL SOMBREADOR 3215F
NÚCLEO DEL SOMBREADOR 3215N
UNIDAD DE MOSAICO
3218
MMU
3220A
MMU
3220B
CACHÉ
3225A
CACHÉ
3225B
INTERCONEXIÓN
3230A
INTERCONEXIÓN
3230B

FIG. 32