(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **3 011 834**

(51) Int. Cl.:

G06T 1/20 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **08.03.2018 E 18160825 (8)**

(97) Fecha y número de publicación de la concesión europea: **04.12.2024 EP 3385901**

(54) Título: **Mecanismo de cálculo disperso del aprendizaje automático**

(30) Prioridad:

09.04.2017 US 201715482791

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
08.04.2025

(73) Titular/es:

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

(72) Inventor/es:

NURVITADHI, ERIKO;
VEMBU, BALAJI;
LIN, TSUNG-HAN;
SINHA, KAMAL;
BARIK, RAJKISHORE y
GALOPPO VON BORRIES, NICOLAS C.

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Mecanismo de cálculo disperso del aprendizaje automático

CAMPO

Las formas de realización hacen referencia en general al procesamiento de datos y, más particularmente, al procesamiento de datos por medio de una unidad de procesamiento gráfico de propósito general.

ANTECEDENTES DE LA DESCRIPCIÓN

El procesamiento de datos gráficos paralelo actual incluye sistemas y métodos desarrollados para llevar a cabo operaciones específicas sobre datos gráficos tales como, por ejemplo, interpolación lineal, teselado, rasterización, asignación de texturas, pruebas de profundidad, etc. Tradicionalmente, los procesadores gráficos utilizaban unidades de cálculo de función fija para procesar los datos gráficos; sin embargo, más recientemente, partes de los procesadores gráficos se han hecho programables, permitiendo a dichos procesadores soportar una mayor variedad de operaciones para procesar datos de vértices y fragmentos.

Para aumentar aún más el rendimiento, los procesadores gráficos normalmente utilizan técnicas de procesamiento como el pipelining, que intenta procesar, en paralelo, tantos datos gráficos como sea posible a lo largo de las distintas partes del canal de gráficos. Los procesadores gráficos paralelos con arquitecturas de una sola instrucción y múltiples subprocesos (SIMT) están diseñados para maximizar la cantidad de procesamiento paralelo en el canal de gráficos. En una arquitectura SIMT, los grupos de subprocesos paralelos intentan ejecutar las instrucciones del programa de forma sincronizada con la mayor frecuencia posible para aumentar la eficiencia de procesamiento. En Shane Cook, CUDA Programming Chapter 3, páginas 37-51 (2013), se ofrece una visión general del software y el hardware de las arquitecturas SIMT.

CHEN YU-HSIN ET AL, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks", IEEE JOURNAL OF SOLID-STATE CIRCUITS, IEEE SERVICE CENTER, PISCATAWAY, NJ, USA, (20170101), vol. 52, nº 1, páginas 127 - 138, 1-15 * página 135, columna 1, párrafo 1; figura 12 * página 127, columna r, párrafo 2 * da a conocer un acelerador para redes convolucionales profundas de última generación (CNN). Optimiza la eficiencia energética de todo el sistema, incluido el microprocesador acelerador y la DRAM fuera del microprocesador, para varias formas de CNN reconfigurando la arquitectura.

Anónimo, "MULT - Multiply", Z8000 CPU User's Reference Manual, Englewood Cliffs, New Jersey 07632, USA, (19820101), páginas 141 - 143, 1-15 * Última tabla: "When the multiplier is zero..."; página 141 * que para la operación MULT, cuando el multiplicador es cero, el tiempo de ejecución de Multiplicar se reduce.

Knuth D.E, "Multiple-Precision Arithmetic", The Art of Computer Programming: Combinatorial Algorithms, Upper Saddle River, NJ, Addison-Wesley, (19980101), páginas 265 - 271 describe que una prueba puede ahorrar tiempo si existe una posibilidad razonable de que un argumento sea cero.

El documento US 5.262.973 describe que cuando se va a llevar a cabo una operación aritmética, los operandos se envían simultáneamente a la unidad aritmética para llevar a cabo la operación aritmética compleja y a un mecanismo de comprobación de operandos que determina si uno o ambos operandos son una instancia específica de un operando trivial. Si uno de los operandos es una instancia específica de un operando trivial, las operaciones aritméticas complejas se detienen y el mecanismo de comprobación emite rápidamente el resultado de la operación aritmética de acuerdo con el operando trivial detectado. En consecuencia, se evita la necesidad de llevar a cabo operaciones aritméticas complejas sobre operandos triviales.

El documento US 7.945.765 B2 describe que un aparato electrónico incluye varias fases interconectadas en serie como un canal para llevar a cabo procesamientos secuenciales en los operandos de entrada. Un circuito de acortamiento asociado con al menos una fase del canal reconoce cuando uno o más de los operandos de entrada para la fase ha sido predeterminado como apropiado para acortar y ejecutar el acortamiento según convenga.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

De modo que, para entender en detalle las características de las formas de realización indicadas anteriormente, se puede tomar como referencia una descripción más particular de dichas formas, brevemente resumida anteriormente, que se ilustra en los dibujos adjuntos. Sin embargo, se debe tener en cuenta que, los dibujos adjuntos ilustran sólo formas de realización típicas y por consiguiente no se deben considerar restrictivos de su alcance. La invención se define por las reivindicaciones adjuntas.

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático configurado para implementar uno o más aspectos de las formas de realización descritas en la presente memoria;

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

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

Las **Figuras 4A-4F** ilustran una arquitectura de ejemplo en la que varias GPU se acoplan con capacidad de comunicación a varios procesadores de múltiples núcleos;

La **Figura 5** ilustra un canal de procesamiento gráfico, de acuerdo con una forma de realización;

La **Figura 6** ilustra un dispositivo informático que emplea un mecanismo de cálculo disperso, de acuerdo con una forma de realización;

La **Figura 7A** ilustra una multiplicación de conjuntos de ejemplo;

La **Figura 7B** ilustra una forma de realización de un elemento de procesamiento que tiene un planificador disperso;

La **Figura 7C** ilustra una forma de realización de un elemento de procesamiento que tiene un seguidor disperso;

Las **Figuras 7D&7E** ilustran formas de realización de un procesador de gráficos;

La **Figura 8** ilustra una pila de software de aprendizaje automático, de acuerdo con una forma de realización;

La **Figura 9** ilustra una unidad de procesamiento gráfico de fin general altamente paralela, de acuerdo con una forma de realización;

La **Figura 10** ilustra un sistema informático de múltiples GPU, de acuerdo con una forma de realización;

Las **Figuras 11A-11B** ilustran capas de redes neuronales profundas de ejemplo;

La **Figura 12** ilustra una red neuronal recurrente de ejemplo;

La **Figura 13** ilustra el entrenamiento y despliegue de una red neuronal profunda.

La **Figura 14** es un diagrama de bloques que ilustra un aprendizaje distribuido;

La **Figura 15** ilustra un sistema de inferencia de ejemplo en un microprocesador (SOC) adecuado para llevar a cabo inferencias utilizando un modelo entrenado;

La **Figura 16** es un diagrama de bloques de un sistema de procesamiento, de acuerdo con una forma de realización;

La **Figura 17** es un diagrama de bloques de un procesador de acuerdo con una forma de realización;

La **Figura 18** es un diagrama de bloques de un procesador de gráficos, de acuerdo con una forma de realización;

La **Figura 19** es un diagrama de bloques de un motor de procesamiento gráfico de un procesador de gráficos de acuerdo con algunas formas de realización;

La **Figura 20** es un diagrama de bloques de un procesador de gráficos proporcionado por una forma de realización adicional;

La **Figura 21** ilustra la lógica de ejecución de subprocesos que incluye un conjunto de elementos de procesamiento empleados en algunas formas de realización;

La **Figura 22** es un diagrama de bloques que ilustra formatos de instrucción de procesador de gráficos de acuerdo con algunas formas de realización;

La **Figura 23** es un diagrama de bloques de un procesador de gráficos, de acuerdo con otra realización;

Las **Figuras 24A-24B** ilustran un formato de orden de procesador de gráficos y una secuencia de órdenes, de acuerdo con algunas formas de realización;

La **Figura 25** ilustra una arquitectura de software de gráficos de ejemplo para un sistema de procesamiento de datos de acuerdo con algunas formas de realización;

La **Figura 26** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP, de acuerdo con una forma de realización;

La **Figura 27** es un diagrama de bloques que ilustra un circuito integrado en un microprocesador de sistema de ejemplo, de acuerdo con una forma de realización;

La **Figura 28** es un diagrama de bloques que ilustra un procesador de gráficos de ejemplo adicional; y

La **Figura 29** es un diagrama de bloques que ilustra un procesador de gráficos de un circuito integrado en un microprocesador de sistema de ejemplo adicional, de acuerdo con una forma de realización.

DESCRIPCIÓN DETALLADA

En formas de realización, se describen mecanismos para llevar a cabo un mecanismo de procesamiento de matriz dispersa. En algunas formas de realización, el mecanismo de procesamiento incluye elementos de procesamiento que incluyen un planificador para identificar operandos que tienen un valor cero y evitar la programación de los operandos que tienen el valor cero en la unidad de multiplicación. En otras formas de realización, el mecanismo de procesamiento incluye lógica de seguimiento de patrones para detectar uno o más segmentos de datos dispersos en un bloque de datos almacenado y registrar una ubicación de dirección para cada segmento detectado de datos dispersos. Todavía en otras formas de realización, el mecanismo de procesamiento comprime matrices dispersas y almacena una o más matrices dispersas utilizadas con frecuencia en una memoria intermedia comprimida dispersa para su ejecución para el procesamiento. En una forma de realización adicional, el mecanismo de procesamiento divide varias unidades de ejecución (EU) y asigna cada subdivisión de EU para ejecutar subprocesos asociados con una capa de red neuronal.

En la siguiente descripción, se exponen numerosos detalles específicos para proporcionar una comprensión más minuciosa. Sin embargo, será evidente para un experto en la técnica que las formas de realización descritas en la presente memoria se pueden llevar a la práctica sin uno o más de estos detalles específicos. En otras instancias, no se han descrito características bien conocidas para evitar oscurecer los detalles de las presentes formas de realización.

Visión general del sistema

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático 100 configurado para implementar uno o más aspectos de las formas de realización descritas en la presente memoria. El sistema informático 100 incluye un subsistema de procesamiento 101 que tiene uno o más procesador(es) 102 y una memoria de sistema 104 que se comunican por medio de una trayectoria 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 microprocesadores o puede estar integrado dentro de uno o más procesador(es) 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 a través de un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede permitir que el sistema informático 100 reciba entradas de uno o más dispositivo(s) de entrada 108. Además, el concentrador de E/S 107 puede permitir a un controlador de visualización, que puede estar incluido en uno o más procesador(es) 102, proporcionar salidas a uno o más dispositivo(s) de visualización 110A. En una forma de realización, el uno o más dispositivo(s) de visualización 110A acoplado(s) con el concentrador de E/S 107 puede(n) incluir un dispositivo de visualización local, interno o integrado.

En una forma de realización, el subsistema de procesamiento 101 incluye uno o más procesadores paralelos 112 acoplados al concentrador de memoria 105 por medio de un bus u otro enlace de comunicación 113. El enlace de comunicación 113 puede ser uno de cualquier número de tecnologías o protocolos de enlace de comunicación basados en estándares, tales como, entre otros, una PCI Express, o puede ser una interfaz de comunicaciones o tejido de comunicaciones específico de un proveedor. En una forma de realización, el uno o más procesador(es) paralelo(s) 112 forma(n) un sistema de procesamiento vectorial o paralelo enfocado desde el punto de vista del cálculo que puede incluir un gran número de núcleos de procesamiento y/o clústeres de procesamiento, tales como un procesador de muchos núcleos integrados (MIC). En una forma de realización, el uno o más procesadores paralelos 112 forman un subsistema de procesamiento gráfico que puede proporcionar píxeles a uno del uno o más dispositivos de visualización 110A acoplados por medio del concentrador de E/S 107. El uno o más procesadores paralelos 112 también pueden incluir un controlador de visualización y una interfaz de visualización (no mostrada) para permitir una conexión directa a uno o más dispositivos de visualización 110B.

Dentro del subsistema de E/S 111, una unidad de almacenamiento de sistema 114 se puede conectar al concentrador de E/S 107 para proporcionar un mecanismo de almacenamiento para el sistema informático 100. Un conmutador de E/S 116 se puede utilizar para proporcionar un mecanismo de interfaz que permita conexiones entre el concentrador de E/S 107 y otros componentes, tales como un adaptador de red 118 y/o un adaptador de red inalámbrico 119 que se pueden integrar en la plataforma, y varios dispositivos que se pueden añadir por medio de uno o más dispositivos

complementarios 120. El adaptador de red 118 puede ser un adaptador Ethernet u otro adaptador de red cableado. El adaptador de red inalámbrico 119 puede incluir uno o más de un Wi-Fi, Bluetooth, comunicación de campo cercano (NFC), u otro dispositivo de red que incluya una o más radios inalámbricas.

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

En una forma de realización, el uno o más procesadores paralelos 112 incorporan circuitos optimizados para el procesamiento gráfico y vídeo, incluyendo, por ejemplo, circuitos de salida de vídeo, y constituyen una unidad de procesamiento gráfico (GPU). En otra forma de realización, el uno o más procesadores paralelos 112 incorporan circuitos optimizados para el procesamiento de propósito general, preservando la arquitectura de cálculo subyacente, descrita con mayor detalle en la presente memoria. En otra forma de realización, los componentes del sistema informático 100 se pueden integrar con uno o más elementos del sistema en un único circuito integrado. Por ejemplo, el uno o más procesadores paralelos, el concentrador de memoria 112 105, el (los) procesador(es) 102 y el concentrador de E/S 107 se pueden integrar en un circuito integrado en un microprocesador de sistema (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 forma de realización, al menos una parte de los componentes del sistema informático 100 se puede integrar en un módulo multimicroprocesador (MCM), que se puede interconectar con otros módulos multimicroprocesador para formar un sistema informático modular.

Se apreciará que el sistema informático 100 mostrado en la presente memoria es de ejemplo y que son posibles variaciones y modificaciones. La topología de conexión, incluyendo el número y disposición de los puentes, el número de procesador(es) 102, y el número de procesador(es) paralelo(s) 112, se puede modificar según se desee. Por ejemplo, en algunas formas de realización, la memoria del sistema 104 se conecta al procesador(es) 102 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria del sistema 104 por medio del concentrador de memoria 105 y el(los) procesador(es) 102. En otras topologías alternativas, el(los) procesador(es) paralelo(s) 112 se conecta(n) al concentrador de E/S 107 o directamente a uno del uno o más procesadores 102, en lugar de al concentrador de memoria 105. En otras formas de realización, el concentrador de E/S 107 y el concentrador de memoria 105 se pueden integrar en un único microprocesador. Algunas formas de realización pueden incluir dos o más conjuntos de procesador(es) 102 conectados por medio de múltiples zócalos, que se pueden acoplar con dos o más instancias del(de los) procesador(es) 112 paralelo(s).

Algunos de los componentes que se muestran en la presente memoria son opcionales y pueden no incluirse en todas las implementaciones del sistema informático 100. Por ejemplo, puede admitirse cualquier número de tarjetas o periféricos complementarios, o se pueden eliminar algunos componentes. Adicionalmente, algunas arquitecturas pueden utilizar diferente terminología para componentes similares a aquellos ilustrados en la Figura 1. Por ejemplo, el concentrador de memoria 105 se puede denominar Northbridge en algunas arquitecturas, mientras que el concentrador de E/S 107 se puede denominar Southbridge.

La **Figura 2A** ilustra un procesador paralelo 200, de acuerdo con una forma de realización. Los varios componentes del procesador paralelo 200 se pueden implementar utilizando uno o más dispositivos de circuitos integrados, como procesadores programables, circuitos integrados de aplicación específica (ASIC) o conjuntos de puertas programables en campo (FPGA). El procesador paralelo 200 ilustrado es una variante del uno o más procesadores paralelos 112 mostrados en la Figura 1, de acuerdo con una forma de realización.

En una forma de realización, el procesador paralelo 200 incluye una unidad de procesamiento paralelo 202. La unidad de procesamiento paralelo incluye una unidad de E/S 204 que permite la comunicación con otros dispositivos, incluidas otras instancias de la unidad de procesamiento paralelo 202. La unidad de E/S 204 se puede conectar directamente a otros dispositivos. En una forma de realización, la unidad de E/S 204 se conecta con otros dispositivos mediante la utilización de una interfaz de concentrador o conmutador, como el concentrador de memoria 105. Las conexiones entre el concentrador de memoria 105 y la unidad de E/S 204 forman un enlace de comunicación 113. Dentro de la unidad de procesamiento paralelo 202, la unidad de E/S 204 se conecta con una interfaz anfitrión 206 y una barra transversal de memoria 216, donde la interfaz anfitrión 206 recibe órdenes dirigidas a llevar a cabo operaciones de procesamiento y la barra transversal de memoria 216 recibe órdenes dirigidas a llevar a cabo operaciones de memoria.

Cuando la interfaz anfitrión 206 recibe una memoria intermedia de órdenes por medio de la unidad de E/S 204, la interfaz anfitrión 206 puede dirigir operaciones de trabajo para llevar a cabo esas órdenes a un front-end 208. En una forma de realización, el front-end 208 se acopla con un planificador 210, que se configura para distribuir órdenes u otros elementos de trabajo a un conjunto de clústeres de procesamiento 212. En una forma de realización, el planificador 210 asegura que el conjunto de clústeres de procesamiento 212 se configura correctamente y en un estado

válido antes de que las tareas se distribuyan a los clústeres de procesamiento del conjunto de clústeres de procesamiento 212.

El conjunto de clústeres de procesamiento 212 puede incluir hasta "N" clústeres de procesamiento (por ejemplo, clúster 214A, clúster 214B, hasta clúster 214N). Cada clúster 214A a 214N del conjunto de clústeres de procesamiento 212 puede ejecutar un gran número de subprocesos concurrentes. El planificador 210 puede asignar trabajo a los clústeres 214A-214N del conjunto de clústeres de procesamiento 212 utilizando varios algoritmos de planificación y/o distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surja para cada tipo de programa o cálculo. La planificación puede ser manejada dinámicamente por el planificador 210, o puede ser asistida en parte por la lógica del compilador durante la compilación de la lógica del programa configurada para su ejecución por el conjunto de clústeres de procesamiento 212.

En una forma de realización, se pueden asignar diferentes clústeres 214A a 214N del conjunto de clústeres de procesamiento 212 para procesar diferentes tipos de programas o para llevar a cabo diferentes tipos de cálculos.

El conjunto de clústeres de procesamiento 212 se puede configurar para llevar a cabo varios tipos de operaciones de procesamiento paralelo. En una forma de realización, el conjunto de clústeres de procesamiento 212 se configura para llevar a cabo operaciones de cálculo paralelo de propósito general. Por ejemplo, el conjunto de clústeres de procesamiento 212 puede incluir lógica para ejecutar tareas de procesamiento que incluyen el filtrado de datos de vídeo y/o audio y/u operaciones de modelado, incluidas operaciones físicas y la realización de transformaciones de datos.

En una forma de realización, el conjunto de clústeres de procesamiento 212 se configura para llevar a cabo operaciones de procesamiento gráfico paralelo de gráficos. En formas de realización en las que el procesador paralelo 200 se configura para llevar a cabo operaciones de procesamiento gráfico, el conjunto de clústeres de procesamiento 212 puede incluir lógica adicional para admitir la ejecución de dichas operaciones de procesamiento gráfico, incluidas, entre otras, lógica de muestreo de textura para llevar a cabo operaciones de textura, así como lógica de teselado y otra lógica de procesamiento de vértices. Además, el conjunto de clústeres de procesamiento 212 se puede configurar para ejecutar programas de sombreado relacionados con el procesamiento gráfico tales como, entre otros, sombreadores de vértices, sombreadores de teselado, sombreadores de geometría y sombreadores de píxeles. La unidad de procesamiento paralelo 202 puede transferir datos desde la memoria del sistema por medio de la unidad de E/S 204 para su procesamiento. Durante el procesamiento, los datos transferidos se pueden almacenar en la memoria en microprocesador (por ejemplo, la memoria del procesador paralelo 222) durante el procesamiento y, a continuación, escribirse de nuevo en la memoria del sistema.

En una forma de realización, cuando la unidad de procesamiento paralelo 202 se utiliza para llevar a cabo el procesamiento gráfico, el planificador 210 se puede configurar para dividir la carga de trabajo de procesamiento en tareas de tamaño aproximadamente igual, para permitir mejor la distribución de las operaciones de procesamiento gráfico a múltiples clústeres 214A-214N del conjunto de clústeres de procesamiento 212. En algunas formas de realización, partes del conjunto de clústeres de procesamiento 212 se pueden configurar para llevar a cabo diferentes tipos de procesamiento. Por ejemplo, una primera parte se puede configurar para llevar a cabo sombreado de vértices y generación de topología, una segunda parte se puede configurar para llevar a cabo teselado y sombreado de geometría, y una tercera parte se puede configurar para llevar a cabo sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen renderizada para su visualización. Los datos intermedios producidos por uno o más de los clústeres 214A-214N se pueden almacenar en memorias intermedias para permitir que los datos intermedios sean transmitidos entre los clústeres 214A-214N para su posterior procesamiento.

Durante la operación, el conjunto de clústeres de procesamiento 212 puede recibir tareas de procesamiento para ser ejecutadas por medio del planificador 210, que recibe órdenes que definen las tareas de procesamiento desde el front-end 208. Para las operaciones de procesamiento gráfico, las tareas de procesamiento pueden incluir índices de datos a procesar, por ejemplo, datos de superficie (parche), datos primitivos, datos de vértice y/o datos de píxel, así como parámetros de estado y órdenes que definen cómo se van a procesar los datos (por ejemplo, qué programa se va a ejecutar). El planificador 210 se puede configurar para buscar los índices correspondientes a las tareas o puede recibir los índices del front-end 208. El front-end 208 se puede configurar para asegurar que el conjunto de clústeres de procesamiento 212 se configura en un estado válido antes de que se inicie la carga de trabajo especificada por las memorias intermedias de órdenes entrantes (por ejemplo, memorias intermedias de lotes, memorias intermedias de empuje, etc.).

Cada una de la una o más instancias de la unidad de procesamiento paralelo 202 se puede acoplar con la memoria del procesador paralelo 222. Se puede acceder a la memoria del procesador paralelo 222 por medio de la barra transversal de memoria 216, que puede recibir solicitudes de memoria del conjunto de clústeres de procesamiento 212 así como de la unidad de E/S 204. La barra transversal de memoria 216 puede acceder a la memoria del procesador paralelo 222 por medio de la interfaz de memoria 218. La interfaz de memoria 218 puede incluir múltiples unidades de subdivisión (por ejemplo, unidad de subdivisión 220A, unidad de subdivisión 220B, a través de la unidad de subdivisión 220N) que se pueden acoplar cada una a una parte (por ejemplo, unidad de memoria) de la memoria del procesador paralelo 222. En una implementación, el número de unidades de subdivisión 220A-220N se configura

para ser igual al número de unidades de memoria, de tal forma que una primera unidad de subdivisión 220A tenga una primera unidad de memoria 224A correspondiente, una segunda unidad de subdivisión 220B tenga una unidad de memoria 224B correspondiente, y una enésima unidad de subdivisión 220N tenga una enésima unidad de memoria 224N correspondiente. En otras formas de realización, el número de unidades de subdivisión 220A-220N puede no ser igual al número de dispositivos de memoria.

En varias formas de realización, las unidades de memoria 224A-224N pueden incluir varios tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámico (DRAM) o memoria de acceso aleatorio gráfico, como memoria de acceso aleatorio gráfico síncrona (SGRAM), incluyendo memoria de doble velocidad de datos gráficos (GDDR). En una forma de realización, las unidades de memoria 224A-224N también pueden incluir memoria apilada 3D, incluyendo, entre otras, memoria de ancho de banda alto (HBM). Los expertos en la técnica apreciarán que la implementación específica de las unidades de memoria 224A-224N puede variar, y se puede seleccionar a partir de uno de varios diseños convencionales. Los objetivos de renderizado, como las memorias intermedias de fotogramas o los mapas de texturas, se pueden almacenar a través de las unidades de memoria 224A-224N, permitiendo a las unidades de subdivisión 220A-220N escribir partes de cada objetivo de renderizado en paralelo para utilizar de forma eficiente el ancho de banda disponible de la memoria del procesador paralelo 222. En algunas formas de realización, se puede excluir una instancia local de la memoria del procesador paralelo 222 en favor de un diseño de memoria unificado que utilice la memoria del sistema junto con la memoria caché local.

En una forma de realización, uno cualquiera de los clústeres 214A-214N del conjunto de clústeres de procesamiento 212 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 224A-224N dentro de la memoria del procesador paralelo 222. La barra transversal de memoria 216 se puede configurar para transferir la salida de cada clúster 214A-214N a cualquier unidad de subdivisión 220A-220N o a otro clúster 214A-214N, que puede llevar a cabo operaciones de procesamiento adicionales en la salida. Cada clúster 214A-214N se puede comunicar con la interfaz de memoria 218 a través de la barra transversal de memoria 216 para leer o escribir en varios dispositivos de memoria externos. En una forma de 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 del procesador paralelo 222, lo que permite que las unidades de procesamiento dentro de los diferentes clústeres de procesamiento 214A a 214N comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 202. En una forma de realización, la barra transversal de memoria 216 puede utilizar canales virtuales para separar secuencias de tráfico entre los clústeres 214A a 214N y las unidades de subdivisión 220A a 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 complementaria o se pueden interconectar múltiples tarjetas complementarias. Las diferentes instancias de la unidad de procesamiento paralelo 202 se pueden configurar para interoperar incluso si las diferentes instancias tienen diferentes números de núcleos de procesamiento, diferentes cantidades de memoria del procesador paralelo local y/u otras diferencias de configuración. Por ejemplo, y en una forma de realización, algunas instancias de la unidad de procesamiento paralelo 202 pueden incluir unidades de coma flotante de mayor precisión en relación con otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 202 o del procesador paralelo 200 se pueden implementar en una variedad de configuraciones y factores de forma, incluyendo, entre otros, 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 forma de realización. En una forma de realización, la unidad de subdivisión 220 es una instancia de una de las unidades de subdivisión 220A-220N de la Figura 2A. Según se ilustra, la unidad de subdivisión 220 incluye una memoria caché L2 221, una interfaz de memoria intermedia de fotogramas 225 y una ROP 226 (unidad de operaciones de rasterización). La memoria caché L2 221 es una memoria caché de lectura/escritura que se configura para llevar a cabo operaciones de carga y de almacenamiento recibidas desde la barra cruzada de memoria 216 y la ROP 226. Los fallos de lectura y las solicitudes de escritura urgentes se emiten por la memoria caché L2 221 a la interfaz de memoria intermedia de fotogramas 225 para su procesamiento. Actualizaciones sucias también se pueden enviar a la memoria intermedia de fotogramas por medio de la interfaz de la memoria intermedia de fotogramas 225 para procesamiento oportunista. En una forma de realización, la interfaz de memoria intermedia de fotogramas 225 interactúa con una de las unidades de memoria de la memoria del procesador paralelo, tal como las unidades de memoria 224A-224N de la Figura 2 (por ejemplo, dentro de la memoria del procesador paralelo 222).

En las aplicaciones de gráficos, la ROP 226 es una unidad de procesamiento que lleva a cabo operaciones de rasterización tal como estarcido, prueba z, mezcla y similares. A continuación, la ROP 226 proporciona datos de gráficos procesados que se almacenan en la memoria gráfica. En algunas formas de realización, la ROP 226 incluye lógica de compresión para comprimir los datos z o de color que se escriben en la memoria y descomprimir los datos z o de color que se leen de la memoria. En algunas formas de realización, la ROP 226 está incluida dentro de cada clúster de procesamiento (por ejemplo, clúster 214A a 214N de la figura 2) en lugar de dentro de la unidad de subdivisión 220. En una forma de realización de este tipo, las solicitudes de lectura y escritura de datos de píxeles se transmiten a través de la barra transversal de memoria 216 en lugar de datos de fragmentos de píxeles.

Los datos de gráficos procesados se pueden visualizar en un dispositivo de visualización, tal como uno del uno o más dispositivos de visualización 110 de la Figura 1, encaminarse para su procesamiento adicional por el/los procesador(es) 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 un clúster de procesamiento 214 dentro de una unidad de procesamiento paralelo, de acuerdo con una forma de realización. En una forma de realización, el clúster de procesamiento es una instancia de uno de los clústeres de procesamiento 214A-214N de la Figura 2. El clúster de procesamiento 214 se puede configurar para ejecutar muchos subprocesos en paralelo, donde el término "subproceso" se refiere a una instancia de un programa particular que se ejecuta en un conjunto particular de datos de entrada. En algunas formas de realización, se utilizan técnicas de emisión de instrucciones de una sola instrucción y múltiples datos (SIMD) para soportar la ejecución en paralelo de un gran número de subprocesos sin proporcionar múltiples unidades de instrucciones independientes. En otras formas de realización, se utilizan técnicas de una sola instrucción, múltiples subprocesos (SIMT) para soportar la ejecución paralela de un gran número de subprocesos generalmente sincronizados, utilizando una unidad de instrucción común configurada para emitir instrucciones a un conjunto de motores de procesamiento dentro de cada uno de los clústeres de procesamiento. A diferencia de un régimen de ejecución SIMD, en el que todos los motores de procesamiento ejecutan normalmente instrucciones idénticas, la ejecución SIMT permite que diferentes subprocesos sigan más fácilmente trayectorias de ejecución divergentes a través de un programa de subprocesos dado. Los expertos en la técnica entenderán que un régimen de procesamiento SIMD representa un subconjunto funcional de un régimen de procesamiento SIMT.

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

Cada multiprocesador de gráficos 234 dentro del clúster de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades de lógica aritmética, unidades de carga-almacenamiento, etc.). La lógica de ejecución funcional se puede configurar con una configuración de canales en la cual las nuevas instrucciones se pueden emitir antes de que las instrucciones previas sean completadas. La lógica de ejecución funcional soporta una variedad de operaciones, incluyendo aritmética de enteros y coma flotante, operaciones de comparación, operaciones booleanas, cambio de bits y cálculo de varias funciones algebraicas. En una forma de realización, se puede hacer uso del mismo hardware de unidad funcional para llevar a cabo diferentes operaciones y puede estar presente cualquier combinación de unidades funcionales.

Las instrucciones transmitidas al clúster de procesamiento 214 constituyen un subproceso. Un conjunto de subprocesos ejecutándose a través del conjunto de motores de procesamiento paralelo es un grupo de subprocesos. Un grupo de subprocesos ejecuta el mismo programa en diferentes datos de entrada. Cada subproceso dentro de un grupo de subprocesos se puede asignar a un motor de procesamiento diferente dentro de un multiprocesador de gráficos 234. Un grupo de subprocesos puede incluir menos subprocesos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando un grupo de subprocesos incluye menos subprocesos que el número de motores de procesamiento, uno o más de los motores de procesamiento pueden estar inactivos durante los ciclos en los que se está procesando ese grupo de subprocesos. Un grupo de subprocesos también puede incluir más subprocesos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234. Cuando el grupo de subprocesos incluye más subprocesos que el número de motores de procesamiento dentro del multiprocesador de gráficos 234, el procesamiento se puede llevar a cabo durante ciclos de reloj consecutivos. En una forma de realización, múltiples grupos de subprocesos se puede ejecutar simultáneamente en un multiprocesador de gráficos 234.

En una forma de realización, el multiprocesador de gráficos 234 incluye una memoria caché interna para llevar a cabo operaciones de carga y de almacenamiento. En una forma de realización, el multiprocesador de gráficos 234 puede renunciar a una memoria caché interna y utilizar una memoria caché (por ejemplo, la memoria caché L1 308) dentro del clúster de procesamiento 214. Cada multiprocesador de gráficos 234 también tiene acceso a memorias caché L2 dentro de las unidades de subdivisión (por ejemplo, unidades de subdivisión 220A-220N de la Figura 2) que se comparten entre todos los clústeres de procesamiento 214 y se pueden utilizar para transferir datos entre subprocesos. El multiprocesador de gráficos 234 también puede acceder a la memoria global fuera del microprocesador, que puede incluir una o más de la memoria del procesador paralelo local y/o la memoria del sistema. Cualquier memoria externa a la unidad de procesamiento paralelo 202 se puede utilizar como memoria global. Las formas de realización en las

que el clúster de procesamiento 214 incluye múltiples instancias del multiprocesador de gráficos 234 pueden compartir instrucciones y datos comunes, que se pueden almacenar en la memoria caché L1 308.

Cada clúster de procesamiento 214 puede incluir una MMU 245 (unidad de gestión de memoria) que se configura para asignar direcciones virtuales en direcciones físicas. En otras formas de realización, una o más instancias de la MMU 245 pueden residir dentro de la interfaz de memoria 218 de la Figura 2. La MMU 245 incluye un conjunto de entradas de tabla de páginas (PTE) utilizadas para asignar una dirección virtual a una dirección física de una tesela (más información sobre teselado) y opcionalmente un índice de línea de memoria caché. La MMU 245 puede incluir memorias intermedias de traducción de direcciones (TLB) o memorias caché que pueden residir dentro del multiprocesador de gráficos 234 o la memoria caché L1 o el clúster de procesamiento 214. La dirección física se procesa para distribuir la localidad de acceso a datos de superficie para permitir un intercalado eficiente de solicitudes entre las unidades de subdivisión. El índice de línea de memoria caché se puede utilizar para determinar si una solicitud de una línea de memoria caché es un acierto o error.

En aplicaciones gráficas y de cálculo, un clúster de procesamiento 214 se puede configurar de tal forma que cada multiprocesador de gráficos 234 se acopla a una unidad de textura 236 para llevar a cabo operaciones de asignación de textura, por ejemplo, determinar posiciones de muestra de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen de una memoria caché L1 de textura interna (no mostrada) o, en algunas formas de realización, de la memoria caché L1 dentro del multiprocesador de gráficos 234 y se obtienen de una memoria caché L2, de la memoria del procesador paralelo local o de la memoria del sistema, según sea necesario. Cada multiprocesador de gráficos 234 envía las tareas procesadas a la barra transversal de datos 240 para proporcionar la tarea procesada a otro clúster de procesamiento 214 para su posterior procesamiento o para almacenar la tarea procesada en una memoria caché L2, memoria del procesador paralelo local o memoria del sistema a través de la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones de rasterización previa) se configura para recibir datos del multiprocesador de gráficos 234, dirigir los datos a unidades ROP, que se pueden ubicar junto con unidades de subdivisión descritas en la presente memoria (por ejemplo, unidades de subdivisión 220A-220N de la Figura 2). La unidad preROP 242 puede llevar a cabo optimizaciones para la mezcla de colores, organizar datos de color de píxeles y llevar a cabo conversiones de direcciones.

Se apreciará que la arquitectura principal descrita en la presente memoria es de ejemplo y que se pueden llevar a cabo variaciones y modificaciones. Cualquier número de unidades de procesamiento, por ejemplo, multiprocesador de gráficos 234, unidades de textura 236, preROP 242, etc., se puede incluir dentro de un clúster de procesamiento 214. Además, mientras que sólo se muestra un clúster de procesamiento 214, una unidad de procesamiento paralelo según se describe en la presente memoria puede incluir cualquier número de instancias del clúster de procesamiento 214. En una forma de realización, cada clúster de procesamiento 214 se puede configurar para operar de forma independiente de otros clústeres de procesamiento 214 utilizando unidades de procesamiento separadas y distintas, memorias caché L1, etc.

La **Figura 2D** muestra un multiprocesador de gráficos 234, de acuerdo con una forma de realización. En formas de realización de este tipo, el multiprocesador de gráficos 234 se acopla al gestor de canales 232 del clúster de procesamiento 214. El multiprocesador de gráficos 234 tiene un canal de ejecución que incluye, entre otros, una memoria caché de instrucciones 252, una unidad de instrucciones 254, una unidad de asignación de direcciones 256, un archivo de registro 258, uno o más núcleos de unidad de procesamiento gráfico de propósito general (GPGPU) 262, y una o más unidades de carga/almacenamiento 266. Los núcleos GPGPU 262 y las unidades de carga/almacenamiento 266 están acoplados con la memoria caché 272 y la memoria compartida 270 a través de una interconexión de memoria y memoria caché 268.

En una forma de realización, la memoria caché de instrucciones 252 recibe una secuencia de instrucciones para ejecutar desde el gestor de canales 232. Las instrucciones se almacenan en la memoria caché de instrucciones 252 y se envían para su ejecución por la unidad de instrucciones 254. La unidad de instrucciones 254 puede enviar las instrucciones como grupos de subprocesos (por ejemplo, warps), con cada subproceso del grupo de subprocesos asignado a una unidad de ejecución diferente dentro del núcleo GPGPU 262. Una instrucción puede acceder a cualquier espacio de direcciones local, compartido o global especificando una dirección dentro de un espacio de direcciones unificado. La unidad de asignación de direcciones 256 se puede utilizar para traducir las direcciones en el espacio de direcciones unificado en una dirección de memoria distinta que pueda ser accedida por 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 los operandos conectados a las trayectorias de datos de las unidades funcionales (por ejemplo, núcleos GPGPU 262, unidades de carga/almacenamiento 266) del multiprocesador de gráficos 324. En una forma de realización, el archivo de registro 258 se divide entre cada una de las unidades funcionales de tal forma que a cada unidad funcional se le asigna una parte dedicada del archivo de registro 258. En una forma de realización, el archivo de registro 258 se divide entre los diferentes warps que están siendo ejecutados por el multiprocesador de gráficos 324.

Cada uno de los núcleos GPGPU 262 puede incluir unidades de coma flotante (FPU) y/o unidades aritméticas lógicas de enteros (ALU) que se utilizan para ejecutar instrucciones del multiprocesador de gráficos 324. Los núcleos GPGPU 262 pueden ser similares en arquitectura o pueden diferir en arquitectura, de acuerdo con las formas de realización. Por ejemplo, y en una forma de realización, una primera parte de los núcleos GPGPU 262 incluye una FPU de precisión simple y una ALU de enteros, mientras que una segunda parte de los núcleos GPGPU incluye una FPU de precisión doble. En una forma de realización, las FPU pueden implementar el estándar IEEE 754-2008 para aritmética de coma flotante o permitir aritmética de coma flotante de precisión variable. El multiprocesador de gráficos 324 puede incluir adicionalmente una o más unidades de función fija o especial para llevar a cabo funciones específicas tales como copiar rectángulos u operaciones de mezcla de píxeles. En una forma de realización, uno o más de los núcleos GPGPU también pueden incluir lógica de función fija o especial.

La interconexión de memoria y memoria caché 268 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador de gráficos 324 al archivo de registro 258 y a la memoria compartida 270. En una forma de realización, la interconexión de memoria y memoria caché 268 es una interconexión de barra transversal que permite que la unidad de carga/almacenamiento 266 implemente operaciones de carga y almacenamiento entre la memoria compartida 270 y el archivo de registro 258. El archivo de registro 258 puede operar a la misma frecuencia que los núcleos GPGPU 262, por lo que la transferencia de datos entre los núcleos GPGPU 262 y el archivo de registro 258 es de muy baja latencia. La memoria compartida 270 se puede utilizar para permitir la comunicación entre los subprocesos que se ejecutan en las unidades funcionales dentro del multiprocesador de gráficos 234. La memoria caché 272 se puede utilizar como una memoria caché de datos, por ejemplo, para almacenar en memoria caché los datos de textura comunicados entre las unidades funcionales y la unidad de textura 236. La memoria compartida 270 también se puede utilizar como memoria caché gestionada por programa. Los subprocesos que se ejecutan en los núcleos GPGPU 262 pueden almacenar en forma de programa datos dentro de la memoria compartida, además de los datos almacenados de forma automática en memoria caché que se almacenan dentro de la memoria caché 272.

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

La **Figura 3A** muestra un multiprocesador de gráficos 325 de acuerdo con una forma de realización adicional. El multiprocesador de gráficos 325 incluye múltiples instancias adicionales de unidades de recursos de ejecución con respecto 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, del archivo de registros 334A-334B y de la(s) unidad(es) de textura 344A-344B. El multiprocesador de gráficos 325 también incluye múltiples conjuntos de gráficos o unidades de ejecución de cálculo (por ejemplo, núcleo de GPGPU 336A-336B, núcleo de GPGPU 337A-337B, núcleo de GPGPU 338A-338B) y múltiples conjuntos de unidades de carga/almacenamiento 340A-340B. En una forma de realización, las unidades de recursos de ejecución tienen una memoria caché de instrucciones común 330, una memoria caché de textura y/o datos 342 y una memoria compartida 346. Los varios componentes pueden comunicarse por medio de una estructura de interconexión 327. En una forma de realización, el tejido de interconexión 327 incluye uno o más conmutadores de barra transversal para permitir la comunicación entre los diversos componentes del multiprocesador de gráficos 325.

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

Los expertos en la técnica entenderán que la arquitectura descrita en las Figuras 1, 2A-2D y 3A-3B es descriptiva. Por lo tanto, las técnicas descritas en la presente memoria se pueden implementar en cualquier unidad de procesamiento configurada adecuadamente, incluyendo, sin limitación, uno o más procesadores de aplicaciones móviles, una o más unidades centrales de procesamiento (CPU) de escritorio o servidor, incluyendo CPU multinúcleo, 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 gráficos o unidades de procesamiento de propósito especial, sin apartarse del alcance de las formas de realización descritas en la presente memoria.

En algunas formas de realización, un procesador paralelo o GPGPU, tal y según se describe en la presente memoria, se acopla con capacidad de comunicación a los núcleos del procesador/anfitrión para acelerar las operaciones gráficas, las operaciones de aprendizaje automático, las operaciones de análisis de patrones y diversas funciones de propósito general de la GPU (GPGPU). La GPU se puede acoplar con capacidad de comunicación al

procesador/núcleos anfitrión a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otras formas de realización, la GPU se puede integrar en el mismo paquete o microprocesador que los núcleos y se puede acoplar con capacidad de comunicación a los núcleos a través de un bus/interconexión de procesador interno (es decir, interno al paquete o microprocesador). 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 órdenes/instrucciones contenidas en un descriptor de trabajo. A continuación, la GPU utiliza circuitos/lógica dedicados para procesar de forma eficiente estas órdenes/instrucciones.

Técnicas para la Interconexión de la GPU al Procesador Anfitrión

La **Figura 4A** ilustra una arquitectura a modo de ejemplo en la que varias GPU 410-413 se acoplan con capacidad de comunicación a varios procesadores de múltiples núcleos 405-406 a través de enlaces de alta velocidad 440-443 (por ejemplo, buses, interconexiones punto a punto, etc.). En una forma de realización, los enlaces de alta velocidad 440-443 admiten un rendimiento de comunicación de 4 GB/s, 30 GB/s, 80 GB/s o superior, dependiendo de la implementación. Se pueden utilizar varios protocolos de interconexión, incluidos, entre otros, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no se limitan a ningún protocolo o rendimiento de comunicación particular.

Además, en una forma de realización, dos o más de las GPU 410-413 están interconectadas a través de enlaces de alta velocidad 444-445, que se pueden implementar utilizando los mismos o diferentes protocolos/enlaces que los utilizados para los enlaces de alta velocidad 440-443. Del mismo modo, dos o más de los procesadores multinúcleo 405-406 pueden estar conectados a través de enlaces de alta velocidad 433, que pueden ser buses multiprocesador simétricos (SMP) que funcionan a 20 GB/s, 30 GB/s, 120 GB/s o más. Como alternativa, toda la comunicación entre los varios componentes de sistema que se muestran en la **Figura 4A** se puede lograr utilizando los mismos protocolos/enlaces (por ejemplo, a través de una estructura de interconexión común). Sin embargo, según se ha mencionado, los principios subyacentes de la invención no se limitan a ningún tipo particular de tecnología de interconexión.

En una forma de realización, cada procesador multinúcleo 405-406 se acopla con capacidad de comunicación a una memoria de procesador 401-402, a través de interconexiones de memoria 430-431, respectivamente, y cada GPU 410-413 está acoplada con capacidad de comunicación a la memoria GPU 420-423 a través de interconexiones de memoria GPU 450-453, respectivamente. Las interconexiones de memoria 430-431 y 450-453 pueden utilizar las mismas o diferentes tecnologías de acceso a memoria. A modo de ejemplo, y no de limitación, las memorias de procesador 401-402 y las memorias de GPU 420-423 pueden ser memorias volátiles tales como memorias de acceso aleatorio dinámico (DRAM) (incluyendo DRAM apiladas), SDRAM DDR gráfica (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 forma de realización, una parte de las memorias puede ser memoria volátil y otra parte puede ser memoria no volátil (por ejemplo, utilizando una jerarquía de memoria de dos niveles (2LM)).

Según se describe a continuación, aunque los distintos procesadores 405-406 y GPU 410-413 pueden estar físicamente acoplados a una determinada memoria 401-402, 420-423, respectivamente, puede implementarse una arquitectura de memoria unificada en la que el mismo espacio de direcciones virtual del sistema (también denominado espacio de "direcciones efectivas") se distribuye entre todas las distintas memorias físicas. Por ejemplo, las memorias de procesador 401-402 pueden comprender cada una 64GB del espacio de direcciones de memoria del sistema y las memorias GPU 420-423 pueden comprender cada una 32GB del espacio de direcciones de memoria del sistema (resultando en un total de 256GB de memoria direccionable en este ejemplo).

La **Figura 4B** ilustra detalles adicionales para una interconexión entre un procesador multinúcleo 407 y un módulo de aceleración de gráficos 446 de acuerdo con una forma de realización. El módulo de aceleración de gráficos 446 puede incluir uno o más microprocesadores de GPU integrados en una tarjeta de línea que se acopla al procesador 407 a través del enlace de alta velocidad 440. Como alternativa, el módulo de aceleración de gráficos 446 puede estar integrado en el mismo paquete o microprocesador que el procesador 407.

El procesador 407 ilustrado incluye una pluralidad de núcleos 460A-460D, cada uno con una memoria intermedia de traducción anticipada 461A-461D y una o varias memorias caché 462A-462D. Los núcleos pueden incluir varios 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 búsqueda de instrucciones, unidades de predicción de ramal, decodificadores, unidades de ejecución, memoria intermedia de reordenamiento, etc.). Las memorias caché 462A-462D pueden comprender memorias caché de nivel 1 (L1) y de nivel 2 (L2). Además, pueden incluirse una o más memorias caché compartidas 426 en la jerarquía de almacenamiento en memoria caché y compartirse por conjuntos de los núcleos 460A-460D. Por ejemplo, una forma de realización del procesador 407 incluye 24 núcleos, cada uno con su propia memoria caché L1, doce memorias caché L2 compartidas y doce memorias caché L3 compartidas. En esta realización, una de las memorias caché L2 y L3 es compartida por dos núcleos adyacentes. El procesador 407 y el módulo de integración del acelerador gráfico 446 se conectan con la memoria del sistema 441, que puede incluir las memorias del procesador 401-402

La coherencia se mantiene para los datos y las instrucciones almacenadas en las distintas memorias caché 462A-462D, 456 y la memoria del sistema 441 mediante la comunicación entre núcleos a través de un bus de coherencia 464. Por ejemplo, cada memoria caché puede tener lógica/circuitos de coherencia de memoria caché asociados para comunicarse a través del bus de coherencia 464 en respuesta a lecturas o escrituras detectadas en líneas de memoria caché particulares. En una implementación, se implementa un protocolo de monitorización de memoria caché a través del bus de coherencia 464 para monitorizar los accesos a la memoria caché. Las técnicas monitorización/coherencia de memoria caché se comprenden bien por los expertos en la técnica y no se describirán en detalle en este caso para evitar oscurecer los principios subyacentes de la invención.

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

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

En una forma de realización, el circuito de integración acelerador 436 incluye una unidad de gestión de memoria (MMU) 439 para llevar a cabo varias funciones de gestión de memoria, tales como traducciones de memoria virtual a física (también denominadas traducciones de memoria efectiva a real) y protocolos de acceso a memoria para acceder a la memoria del sistema 441. La MMU 439 también puede incluir una memoria intermedia de traducción anticipada (TLB) (no mostrada) para almacenar en memoria caché las traducciones de direcciones virtuales/efectivas a físicas/reales. En una implementación, una memoria caché 438 almacena órdenes y datos para un acceso efectivo por parte de los motores de procesamiento gráfico 431-432, N. En una forma de realización, los datos almacenados en la memoria caché 438 y las memorias de gráficos 433-434, N se mantienen coherentes con las memorias caché de núcleo 462A-462D, 456 y la memoria de sistema 411. Según se ha mencionado, esto puede lograrse mediante el circuito proxy 425 que participa en el mecanismo de coherencia de memoria caché en nombre de la memoria caché 438 y las memorias 433-434, N (por ejemplo, enviando actualizaciones a la memoria caché 438 relacionadas con modificaciones/accesos de líneas de memoria caché en las memorias caché de procesador 462A-462D, 456 y recibiendo actualizaciones de la memoria caché 438).

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

En una implementación, las direcciones virtual/efectiva de un motor de procesamiento gráfico 431 se convierten en direcciones real/física en la memoria de sistema 411 por la unidad de gestión de memoria 439. Una forma de realización del circuito de integración acelerador 436 soporta múltiples (por ejemplo, 4, 8, 16) módulos aceleradores de gráficos 446 y/u otros dispositivos aceleradores. El módulo acelerador de gráficos 446 puede ser dedicado a una única aplicación ejecutada en el procesador 407 o puede ser compartido entre múltiples aplicaciones. En una forma de realización, se presenta un entorno de ejecución de gráficos virtualizado en el que los recursos de los motores de procesamiento gráfico 431-432, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos se pueden subdividir en "rebanadas" que se asignan a diferentes máquinas virtuales y/o aplicaciones en función de los requisitos y prioridades de procesamiento asociados a las máquinas virtuales y/o aplicaciones.

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

Debido a que los recursos de hardware de los motores de procesamiento gráfico 431-432, N se asignan explícitamente al espacio de direcciones real visto por el procesador anfitrión 407, cualquier procesador anfitrión puede dirigirse a estos recursos directamente utilizando un valor de dirección efectiva. Una función del circuito de integración acelerador 436, en una forma de realización, es la separación física de los motores de procesamiento gráfico 431-432, N de modo que aparezcan en el sistema como unidades independientes.

Según se ha mencionado, en la forma de realización ilustrada, una o más memorias gráficas 433-434, M se acoplan a cada uno de los motores de procesamiento gráfico 431-432, N, respectivamente. Las memorias gráficas 433-434, M almacenan instrucciones y datos que están siendo procesados por cada uno de los motores de procesamiento gráfico 431-432, N. Las memorias gráficas 433-434, M pueden ser memorias volátiles tales como DRAM (incluyendo DRAM apiladas), memoria GDDR (por ejemplo, GDDR5, GDDR6), o HBM, y/o pueden ser memorias no volátiles tales como 3D XPoint o Nano-Ram.

En una forma de realización, para reducir el tráfico de datos a través del enlace 440, se utilizan técnicas de desvío para asegurar que los datos almacenados en las memorias gráficas 433-434, M son datos que serán utilizados con mayor frecuencia por los motores de procesamiento gráfico 431-432, N y preferiblemente no utilizados por los núcleos 460A-460D (al menos no con frecuencia). Del mismo modo, el mecanismo de desvío intenta mantener los datos necesarios para los núcleos (y preferiblemente no los motores de procesamiento gráfico 431-432, N) dentro de las memorias caché 462A-462D, 456 de los núcleos y la memoria del sistema 411.

La **Figura 4C** ilustra otra forma de realización en la que el circuito de integración acelerador 436 se integra dentro del procesador 407. En esta forma de realización, los motores de procesamiento gráfico 431-432, N se comunican directamente a través del enlace de alta velocidad 440 con el circuito de integración acelerador 436 a través de la interfaz 437 y la interfaz 435 (que, de nuevo, puede utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración acelerador 436 puede llevar a cabo las mismas operaciones que las descritas con respecto a la **Figura 4B**, pero posiblemente con un mayor rendimiento dada su proximidad al bus de coherencia 462 y a las memorias caché 462A-462D, 426.

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

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

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

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

La **Figura 4D** ilustra una rebanada de integración del acelerador 490 de ejemplo. Según se utiliza en la presente memoria, una "rebanada" comprende una parte específica de los recursos de procesamiento del circuito de integración acelerador 436. El espacio de dirección efectivo de aplicación 482 dentro de la memoria de sistema 411 almacena elementos de proceso 483. En una forma de realización, los elementos de proceso 483 se almacenan en respuesta a invocaciones de GPU 481 desde aplicaciones 480 ejecutadas en el procesador 407. Un elemento de proceso 483 contiene el estado de proceso para la aplicación 480 correspondiente. Un descriptor de trabajo (WD) 484 contenido en el elemento de proceso 483 puede ser un 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 petición de trabajos en el espacio de direcciones 482 de la aplicación.

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

En una implementación, el modelo de programación de proceso dedicado es específico de la implementación. En este modelo, un único proceso es propietario del módulo de aceleración de gráficos 446 o de un motor de procesamiento gráfico 431 individual. Dado que el módulo de aceleración de gráficos 446 es propiedad de un único proceso, el hipervisor inicializa el circuito de integración 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 que se asigna el módulo de aceleración de gráficos 446.

En operación, una unidad de búsqueda de WD 491 en la rebanada de integración del acelerador 490 obtiene el siguiente WD 484 que incluye una indicación del trabajo a llevar a cabo por uno de los motores de procesamiento gráfico del módulo de aceleración gráfica 446. Los datos del WD 484 se pueden almacenar en registros 445 y ser utilizados por la MMU 439, el circuito de gestión de interrupciones 447 y/o el circuito de gestión de contexto 446, según se ilustra. Por ejemplo, una forma de realización de la MMU 439 incluye circuitos de recorrido de segmento/página para acceder a tablas de segmento/página 486 dentro del espacio de direcciones virtual 485 del sistema operativo. El circuito de gestión de interrupciones 447 puede procesar eventos de interrupción 492 recibidos desde el módulo de aceleración de gráficos 446. Cuando se llevan a cabo operaciones gráficas, la MMU 439 traduce una dirección efectiva 493 generada por un motor de procesamiento gráfico 431-432, N a una dirección real.

En una forma de realización, el mismo conjunto de registros 445 son duplicados para cada motor de procesamiento gráfico 431-432, N y/o módulo de aceleración gráfica 446 y pueden ser inicializados por el hipervisor o sistema operativo. Cada uno de estos registros duplicados puede ser incluido en una rebanada de integración del acelerador 490. En la **Tabla 1.** se muestran registros de ejemplo que pueden ser inicializados por el hipervisor.

Tabla 1 - Registros inicializados por el hipervisor

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

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

Tabla 2 - Registros inicializados por el sistema operativo

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

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

La **Figura 4E** ilustra detalles adicionales para una forma de realización de un modelo compartido. Esta realización incluye un espacio de dirección real de hipervisor 498 en el cual una lista de elementos de proceso 499 es almacenada.

El espacio de direcciones reales del hipervisor 498 es accesible a través de un hipervisor 496 que virtualiza los motores del módulo de aceleración de gráficos para el sistema operativo 495.

Los modelos de programación compartida permiten que todos o un subconjunto de procesos de todas o un subconjunto de subdivisiones en el sistema utilicen un módulo de aceleración de gráficos 446. Existen dos modelos de programación en los que el módulo de aceleración de gráficos 446 es compartido por múltiples procesos y subdivisiones: compartido por tiempo dividido y compartido dirigido por gráficos.

En este modelo, el hipervisor de sistema 496 es propietario del módulo de aceleración de gráficos 446 y pone su función a disposición de todos los sistemas operativos 495. Para que un módulo de aceleración de gráficos 446 soporte la virtualización por el hipervisor de sistema 496, el módulo de aceleración de gráficos 446 puede adherirse a los siguientes requisitos: 1) La solicitud de trabajo de una aplicación debe ser autónoma (es decir, el estado no necesita ser mantenido entre trabajos), o el módulo de aceleración de gráficos 446 debe proporcionar un mecanismo de registro/restauración de contexto. 2) El módulo de aceleración de gráficos 446 garantizará que la solicitud de trabajo de una aplicación se complete en un tiempo determinado, incluyendo cualquier fallo de traducción, o bien el módulo de aceleración de gráficos 446 proporcionará la capacidad de adelantarse al procesamiento del trabajo. 3) El módulo de aceleración de gráficos 446 debe tener garantizada la equidad entre procesos cuando opera en el modelo de programación compartida dirigida.

En una forma de realización, para el modelo compartido, la aplicación 480 debe llevar a cabo una llamada al sistema 495 del sistema operativo con un tipo de módulo de aceleración de gráficos 446, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de registro/restauración de contexto (CSRP). El tipo de módulo de aceleración de gráficos 446 describe la función de aceleración objetivo para la llamada al sistema. El tipo de módulo de aceleración de gráficos 446 puede ser un valor específico del sistema. El WD se formatea específicamente para el módulo de aceleración de gráficos 446 y puede tener la forma de una orden del módulo de aceleración de gráficos 446, un puntero de dirección efectiva a una estructura definida por el usuario, un puntero de dirección efectiva a una cola de órdenes, o cualquier otra estructura de datos para describir el trabajo a llevar a cabo por el módulo de aceleración de gráficos 446. En una forma de realización, el valor AMR es el estado AMR a utilizar para el proceso actual. El valor pasado al sistema operativo es similar a una aplicación configurando el AMR. Si las implementaciones del circuito de integración 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 UAMOR actual al valor AMR antes de pasar el AMR en la llamada al hipervisor. El hipervisor 496 puede opcionalmente aplicar el valor actual del Registro de Anulación de Máscara de Autoridad (AMOR) antes de colocar el AMR en el elemento de proceso 483. En una forma de realización, el CSRP es uno de los registros 445 que contiene la dirección efectiva de un área en el espacio de direcciones de la aplicación 482 para que el módulo de aceleración de gráficos 446 guarde y restaure el estado de contexto. Este puntero es opcional si no se requiere guardar el estado entre trabajos o cuando se adelanta un trabajo. El área de registro/restauración de contexto puede ser memoria de sistema anclada.

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

Tabla 3 - Parámetros de llamada del SO al Hipervisor

1	Un descriptor de trabajo (WD)
2	Un valor del Registro de Máscara de Autoridad (AMR) (posiblemente enmascarado).
3	Un puntero de área de registro/restauración de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) y un ID de subproceso opcional (TID)
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	Dirección virtual del puntero de la tabla de segmentos de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)

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

Tabla 4 - Información sobre los elementos de proceso

1	Un descriptor de trabajo (WD)

2	Un valor del Registro de Máscara de Autoridad (AMR) (posiblemente enmascarado).
3	Un puntero de área de registro/restauración de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) y un ID de subproceso opcional (TID)
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	Dirección virtual del puntero de la 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 forma de realización, el hipervisor inicializa varios registros 445 de la rebanada de integración del acelerador 490.

Según se ilustra en la **Figura 4F**, una forma de realización emplea una memoria unificada direccionable a través de un espacio de dirección de memoria virtual común utilizado para acceder a las memorias de procesador físicas 401-402 y las memorias de GPU 420-423. En esta implementación, las operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de dirección de memoria virtual/efectiva para acceder a las memorias de procesador 401-402 y viceversa, simplificando de este modo la programabilidad. En una forma de realización, una primera parte del espacio de direcciones virtual/efectivo se asigna a la memoria del procesador 401, una segunda parte a la memoria del segundo procesador 402, una tercera parte a la memoria de la GPU 420, y de este modo sucesivamente. De este modo, todo el espacio de memoria virtual/efectiva (a veces denominado espacio de direcciones efectivo) se distribuye entre cada una de las memorias de procesador 401-402 y las memorias de GPU 420-423, lo que permite a cualquier procesador o GPU acceder a cualquier memoria física con una dirección virtual asignada a dicha memoria.

En una forma de realización, los circuitos de gestión de sesgado/coherencia 494A-494E dentro de una o más de las MMU 439A-439E garantiza la coherencia de memoria caché entre las memorias caché de los procesadores anfitriones (por ejemplo, 405) y las GPU 410-413 y técnicas de desvío que indican las memorias físicas en las que deben almacenarse determinados tipos de datos. Si bien se ilustran múltiples instancias de circuitos de gestión de desvío/coherencia 494A-494E en la **Figura 4F**, los circuitos de desvío/coherencia se puede implementar dentro de la MMU de uno o más procesadores anfitriones 405 y/o dentro del circuito de integración acelerador 436.

Una forma de realización permite asignar la memoria 420-423 unida a la GPU como parte de la memoria del sistema y acceder a ella utilizando la tecnología de memoria virtual compartida (SVM), pero sin sufrir los típicos inconvenientes de rendimiento asociados a la coherencia total de la memoria caché del sistema. La posibilidad de acceder a las memorias 420-423 unidas a la GPU como memoria del sistema sin la onerosa sobrecarga de la coherencia de memoria caché proporciona un entorno operativo beneficioso para la descarga de la GPU. Esta disposición permite que el software del procesador anfitrión 405 configure los operandos y acceda a los resultados de los cálculos sin la sobrecarga de las copias de datos DMA de E/S tradicionales. Estas copias tradicionales implican llamadas a controladores, interrupciones y accesos de E/S asignados en memoria (MMIO), todos ellos ineficientes con respecto a los simples accesos a memoria. Al mismo tiempo, la capacidad de acceder a la memoria 420-423 conectada a la GPU sin sobrecargas de coherencia de memoria caché puede ser fundamental para el tiempo de ejecución de un cálculo descargado. Por ejemplo, en las instancias en las que existe un flujo de escritura considerable en la memoria, la sobrecarga de coherencia de memoria caché puede reducir significativamente el ancho de banda de escritura efectivo que ve una GPU 410-413. La eficiencia de la configuración de los operandos, la eficiencia del acceso a los resultados y la eficiencia del cálculo de la GPU desempeñan todas una función en la determinación de la eficacia de la descarga de la GPU.

En una implementación, la selección entre el desvío de la GPU y el desvío del procesador anfitrión se lleva a cabo mediante una estructura de datos de seguimiento del desvío. Puede utilizarse una tabla de desvío, por ejemplo, que puede ser una estructura granular de página (es decir, controlada a la granularidad de una página de memoria) que incluye 1 o 2 bits por página de memoria unida a la GPU. La tabla de desvío se puede implementar en un rango de memoria robado de una o más memorias unidas a la GPU 420-423, con o sin una memoria caché de desvío en la GPU 410-413 (por ejemplo, para almacenar en memoria caché entradas de la tabla de desvío utilizadas con frecuencia/recientemente). Como alternativa, toda la tabla de desvío puede mantenerse dentro de la GPU.

En una implementación, se accede a la entrada de la tabla de desvío asociada a cada acceso a la memoria unida a la GPU 420-423 antes del acceso real a la memoria de la GPU, lo que provoca las siguientes operaciones. En primer

lugar, las solicitudes locales de la GPU 410-413 que encuentran su página en el desvío de la GPU se reenvían directamente a la memoria de la GPU 420-423 correspondiente. Las solicitudes locales de la GPU que encuentran su página en el desvío de anfitrión se reenvían al procesador 405 (por ejemplo, a través de un enlace de alta velocidad según se describió anteriormente). En una forma de realización, las solicitudes del procesador 405 que encuentran la página solicitada en el desvío del procesador anfitrión completan la petición como una lectura de memoria normal. Como alternativa, las solicitudes dirigidas a una página desviada a la GPU se pueden reenviar a la GPU 410-413. A continuación, la GPU puede pasar la página a un desvío del procesador anfitrión si no está utilizando la página en ese momento.

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

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

En una forma de realización, la coherencia de la memoria caché se mantiene haciendo que temporalmente el procesador anfitrión 405 no pueda almacenar en la memoria caché las páginas desviadas a la GPU. Para acceder a estas páginas, el procesador 405 puede solicitar acceso desde la GPU 410, que puede otorgar o no acceso de inmediato, dependiendo de la implementación. Por lo tanto, para reducir la comunicación entre el procesador 405 y la GPU 410 es beneficioso asegurarse de que las páginas desviadas a la GPU son aquellas que son requeridas por la GPU pero no por el procesador anfitrión 405 y viceversa.

Canal de procesamiento gráfico

La **Figura 5** ilustra un canal de procesamiento gráfico 500, de acuerdo con una forma de realización. En una forma de realización, un procesador gráfico puede implementar el canal de procesamiento gráfico 500 ilustrada. El procesador de gráficos se puede incluir dentro de los subsistemas de procesamiento paralelo según se describe en la presente memoria, tal como el procesador paralelo 200 de la Figura 2, que, en una forma de realización, es una variante del/de los procesador(es) paralelo(s) 112 de la Figura 1. Los varios sistemas de procesamiento paralelo pueden implementar el canal de procesamiento gráfico 500 por medio de una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 202 de la Figura 2) según se describe en la presente memoria. Por ejemplo, una unidad de sombreado (por ejemplo, el multiprocesador de gráficos 234 de la Figura 3) se puede configurar para llevar a cabo las funciones de una o más de una unidad de procesamiento de vértices 504, una unidad de procesamiento de control de teselado 508, una unidad de procesamiento de evaluación de teselado 512, una unidad de procesamiento de geometría 516 y una unidad de procesamiento de fragmentos/píxeles 524. Las funciones de ensamblador de datos 502, de ensambladores de primitivas 506, 514, 518, de unidad de teselado 510, de rasterizador 522 y de unidad de operaciones de rasterización 526 también se pueden llevar a cabo mediante otros motores de procesamiento dentro de un clúster de procesamiento (por ejemplo, clúster de procesamiento 214 de la Figura 3) y una unidad de subdivisión correspondiente (por ejemplo, unidad de subdivisión 220A-220N de la Figura 2). La canal de procesamiento gráfico 500 también se puede implementar utilizando unidades de procesamiento dedicadas para una o más funciones. En una forma de realización, una o más partes del canal de procesamiento gráfico 500 se pueden llevar a cabo mediante una lógica de procesamiento paralelo dentro de un procesador de propósito general (por ejemplo, una CPU). En una forma de realización, una o más partes del canal de procesamiento gráfico 500 pueden acceder a memoria en microprocesador (por ejemplo, la memoria del procesador paralelo 222 como en la Figura 2) por medio de una interfaz de memoria 528, que puede ser una instancia de la interfaz de memoria 218 de la Figura 2.

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

Una primera instancia de un ensamblador de primitivas 506 recibe atributos de vértices de la unidad de procesamiento de vértices 50. El ensamblador de primitivas 506 lee los atributos de vértice almacenados según sea necesario y construye primitivas gráficas para su procesamiento por la unidad de procesamiento de control de teselado 508. Las primitivas gráficas incluyen triángulos, segmentos de línea, puntos, parches, etc., tal y como soportan varias interfaces de programación de aplicaciones (API) de procesamiento gráfico.

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

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

En algunas formas de realización, la unidad de procesamiento de geometría 516 puede añadir o borrar elementos en la secuencia de geometría. La unidad de procesamiento de geometría 516 envía los parámetros y vértices que especifican las nuevas primitivas gráficas al ensamblador de primitivas 518. El ensamblador de primitivas 518 recibe los parámetros y vértices de la unidad de procesamiento de geometría 516 y construye primitivas gráficas para ser procesadas por una unidad de escalado, selección y recorte de la ventana gráfica 520. La unidad de procesamiento de geometría 516 lee los datos almacenados en la memoria del procesador paralelo o en la memoria del sistema para su utilización en el procesamiento de los datos de geometría. La unidad de escalado, selección y recorte de la ventana gráfica 520 lleva a cabo el recorte, la selección y el escalado de la ventana y envía los gráficos primitivos procesados a un rasterizador 522. El rasterizador 522 puede llevar a cabo el recorte, la selección y el escalado de la ventana. El rasterizador 522 también lleva a cabo la conversión de barrido en las nuevas primitivas gráficas para generar fragmentos y enviar dichos fragmentos y los datos de cobertura asociados a la unidad de procesamiento de fragmentos/píxeles 524. El rasterizador 522 lleva a cabo la conversión de barrido de las nuevas primitivas gráficas y envía datos de fragmentos y cobertura a la unidad de procesamiento de fragmentos/píxeles 524.

La unidad de procesamiento de fragmentos/píxeles 524 es una unidad de ejecución programable que se configura para ejecutar programas de sombreado de fragmentos o programas de sombreado de píxeles. La unidad de procesamiento de fragmentos/píxeles 524 transforma fragmentos o píxeles recibidos del rasterizador 522, según lo especificado por los programas de sombreado de fragmentos o píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 se puede programar para llevar a cabo operaciones que incluyen, entre otras, asignación de texturas, sombreado, mezcla, corrección de texturas y corrección de perspectiva para producir fragmentos o píxeles sombreados que se envían a una unidad de operaciones de rasterización 526. La unidad de procesamiento de fragmentos/píxeles 524 puede leer datos almacenados en la memoria del procesador paralelo o en la memoria del sistema para su utilización en el procesamiento de los datos de fragmentos. Los programas de sombreado de fragmentos o píxeles se pueden configurar para sombrear en muestras, píxeles, teselas u otras granularidades dependiendo de la frecuencia de muestreo configurada para las unidades de procesamiento.

La unidad de operaciones de rasterización 526 es una unidad de procesamiento que lleva a cabo operaciones de rasterización que incluyen, entre otras, estarcido, prueba z, mezcla y similares, y proporciona datos de píxel como datos de gráficos procesados que se almacenarán en una memoria gráfica (por ejemplo, la memoria del procesador paralelo 222 de la Figura 1), para su visualización en el uno o más dispositivos de visualización 110 o para su procesamiento adicional por uno del uno o más procesadores 102 o procesadores paralelos 112. En algunas formas de realización, la unidad de operaciones de rasterización 526 se configura para comprimir datos z o de color que se escriben en memoria y descomprimir datos z o de color que se leen desde la memoria.

La **Figura 6** ilustra una forma de realización de un dispositivo informático 600 que emplea un mecanismo de procesamiento de matriz dispersa. El dispositivo informático 600 (por ejemplo, dispositivos para llevar puestos inteligentes, dispositivos de realidad virtual (VR), dispositivos de visualización colocados en la cabeza (HMD), ordenadores móviles, dispositivos de Internet de las cosas (IoT), ordenadores portátiles, ordenadores de escritorio, ordenadores de tipo servidor, etc.) puede ser el mismo que el sistema de procesamiento de datos 100 de la Figura 1 y, en consecuencia, por razones de brevedad, claridad y facilidad de comprensión, muchos de los detalles indicados anteriormente con referencia a las Figuras 1-5 no se analizan ni repiten en lo sucesivo en la presente memoria. Según se ilustra, en una forma de realización, el dispositivo informático 600 se muestra alojando un mecanismo de procesamiento de matriz dispersa 610.

Según se ilustra, en una forma de realización, el mecanismo de procesamiento de matriz dispersa 610 se puede alojar en la GPU 614. Sin embargo, en otras formas de realización, el mecanismo de procesamiento de matriz dispersa 610 se puede alojar en el controlador de gráficos 616. En otras formas de realización, el mecanismo de procesamiento de matrices dispersas 610 se puede alojar o formar parte del firmware de la unidad central de procesamiento ("CPU" o "procesador de aplicación") 612. Por brevedad, claridad y facilidad de comprensión, durante todo el resto del presente documento, el mecanismo de procesamiento de matrices dispersas 610 se puede describir como parte del controlador gráfico 616; sin embargo, las formas de realización no están limitadas de este modo.

En otra realización más, el mecanismo de procesamiento de matriz dispersa 610 se puede alojar como lógica de software o firmware mediante el sistema operativo 606. En otra forma de realización, el mecanismo de procesamiento de matrices dispersas 610 se puede alojar de forma parcial y simultánea por múltiples componentes del dispositivo informático 600, tales como uno o más de los controladores gráficos 616, GPU 614, firmware de GPU, CPU 612, firmware de CPU, sistema operativo 606, y/o similares. Se contempla que el mecanismo de procesamiento de matriz dispersa 610 o uno o más de sus componentes puedan implementarse como hardware, software y/o firmware.

Durante todo el documento, el término "usuario" se puede denominar indistintamente "espectador", "observador", "persona", "individuo", "usuario final" y/o similares. Se debe tener en cuenta que, durante todo este documento, se puede hacer referencia a términos como "dominio gráfico" indistintamente con "unidad de procesamiento gráfico", "procesador gráfico" o simplemente "GPU" y, de manera similar, se puede hacer referencia a términos como "dominio CPU" o "dominio anfitrión" indistintamente con "unidad de procesamiento informático", "procesador de aplicaciones" o simplemente "CPU".

El dispositivo informático 600 puede incluir cualquier número y tipo de dispositivos de comunicación, como grandes sistemas informáticos, ordenadores servidores, ordenadores de sobremesa, etc., y también puede incluir descodificadores (por ejemplo, descodificadores de televisión por cable basados en Internet, etc.), dispositivos basados en el sistema de posicionamiento global (GPS), etc. El dispositivo informático 600 puede incluir dispositivos informáticos móviles que sirvan como dispositivos de comunicación, como teléfonos móviles, incluidos teléfonos inteligentes, asistentes personales digitales (PDA), tabletas, ordenadores portátiles, lectores electrónicos, televisores inteligentes, plataformas de televisión, dispositivos para llevar puestos (por ejemplo, gafas, relojes, pulseras, tarjetas inteligentes, joyas, prendas de vestir, etc.), reproductores multimedia, etc. Por ejemplo, en una forma de realización, el dispositivo informático 600 puede incluir un dispositivo informático móvil que emplea una plataforma informática que aloja un circuito integrado ("IC"), como un sistema en un microprocesador ("SoC" o "SOC"), que integra diversos componentes de hardware y/o software del dispositivo informático 600 en un único microprocesador.

Según se ilustra, en una forma de realización, el dispositivo informático 600 puede incluir cualquier número y tipo de componentes de hardware y/o software, tales como (sin limitación) la GPU 614, el controlador de gráficos (también denominado "controlador GPU", "lógica de controlador de gráficos", "lógica de controlador", controlador de modo de usuario (UMD), UMD, user-mode driver framework (UMDF), UMDF, o simplemente "driver") 616, CPU 612, memoria 608, dispositivos de red, controladores, o similares, de este modo como fuentes de entrada/salida (E/S) 604, tales como pantallas táctiles, paneles táctiles, almohadillas táctiles, teclados virtuales o normales, ratones virtuales o normales, puertos, conectores, etc.

El dispositivo informático 600 puede incluir un sistema operativo (SO) 606 que sirve como interfaz entre el hardware y/o los recursos físicos del dispositivo informático 600 y un usuario. Se contempla que la CPU 612 puede incluir uno o más procesadores, tales como el(los) procesador(es) 102 de la Figura 1, mientras que la GPU 614 puede incluir uno o más procesadores gráficos (o multiprocesadores).

Se debe tener en cuenta que, términos como "nodo", "nodo informático", "servidor", "dispositivo de servidor", "ordenador en la nube", "servidor en la nube", "ordenador de servidor en la nube", "máquina", "máquina anfitrión", "dispositivo", "dispositivo informático", "ordenador", "sistema informático" y similares, se pueden utilizar indistintamente durante todo este documento. Se debe tener en cuenta además que, términos como "aplicación", "aplicación de software", "programa", "programa de software", "paquete", "paquete de software" y similares, se pueden utilizar indistintamente durante todo este documento. Además, términos como "trabajo", "entrada", "solicitud", "mensaje" y similares se pueden utilizar indistintamente en este documento.

Se contempla, y según se ha descrito anteriormente más detalladamente con referencia a las Figuras 1-5, que algunos procesos del canal de gráficos descritos anteriormente se implementan en software, mientras que el resto se implementa en hardware. Se puede implementar un canal de gráficos en un diseño de coprocesador de gráficos, donde la CPU 612 se diseña para trabajar con la GPU 614, que se puede incluir o situar junto con la CPU 612. En una forma de realización, la GPU 614 puede emplear cualquier número y tipo de lógica de software y hardware convencional para llevar a cabo las funciones convencionales relacionadas con el renderizado de gráficos, de este modo como lógica de software y hardware novedosa para ejecutar cualquier número y tipo de instrucciones.

Según se mencionó anteriormente, la memoria 608 puede incluir una memoria de acceso aleatorio (RAM) que comprende una base de datos de aplicaciones que tiene información de objetos. Un concentrador controlador de memoria, como el concentrador de memoria 105 de la Figura 1, puede acceder a los datos de la RAM y enviarlos a la

GPU 614 para su procesamiento en el canal de gráficos. La RAM puede incluir RAM de doble velocidad de datos (RAM DDR), RAM de salida de datos extendida (RAM EDO), etc. La CPU 612 interactúa con un canal de gráficos de hardware para compartir la funcionalidad de canalización de gráficos.

Los datos procesados se almacenan en una memoria intermedia en el canal de gráficos de hardware y la información de estado se almacena en la memoria 608. A continuación, la imagen resultante se transfiere a fuentes de E/S 604, tales como un componente de visualización para visualizar la imagen. Se contempla que el dispositivo de visualización pueda ser de varios tipos, tales como tubo de rayos catódicos (CRT), transistor de película delgada (TFT), pantalla de cristal líquido (LCD), conjunto de diodos orgánicos emisores de luz (OLED), etc., para mostrar información a un usuario.

La memoria 608 puede comprender una región preasignada de una memoria intermedia (por ejemplo, memoria intermedia de fotogramas); sin embargo, un experto en la técnica debería entender que las formas de realización no están limitadas a esto y que se puede utilizar cualquier memoria accesible a el canal de gráficos inferior. El dispositivo informático 600 puede incluir además un concentrador de control de entrada/salida (E/S) (ICH) 107 como se hace referencia en la Figura 1, como una o más fuentes de E/S 604, etc.

La CPU 612 puede incluir uno o más procesadores para ejecutar instrucciones con el fin de llevar a cabo cualquier rutina de software que implemente el sistema informático. Las instrucciones implican frecuentemente algún tipo de operación llevada a cabo sobre los datos. Tanto los datos como las instrucciones se pueden almacenar en la memoria de sistema 608 y en cualquier memoria caché asociada. La memoria caché se diseña normalmente para tener tiempos de latencia más cortos que la memoria de sistema 608; por ejemplo, la memoria caché se puede integrar en el/los mismo(s) microprocesador(es) de silicio que el/los procesador(es) y/o construirse con celdas de RAM estática (SRAM) más rápidas, mientras que la memoria de sistema 608 puede construirse con celdas de RAM dinámica (DRAM) más lentas. Al tender a almacenar instrucciones y datos usados con mayor frecuencia en la memoria caché en lugar de en la memoria de sistema 608, mejora la eficiencia del rendimiento general del dispositivo informático 600. Se contempla que en algunas formas de realización, la GPU 614 puede existir como parte de la CPU 612 (tal como parte de un paquete físico de CPU), en cuyo caso, la memoria 608 puede ser compartida por la CPU 612 y la GPU 614 o mantenerse separada.

La memoria de sistema 608 puede estar disponible para otros componentes dentro del dispositivo informático 600. Por ejemplo, cualquier dato (por ejemplo, datos de gráficos de entrada) recibido desde varias interfaces al dispositivo informático 600 (por ejemplo, teclado y ratón, puerto de impresora, puerto de red de área local (LAN), puerto de módem, etc.) o recuperado de un elemento de almacenamiento interno del dispositivo informático 600 (por ejemplo, unidad de disco duro) a menudo se pone en cola temporalmente en la memoria de sistema 608 antes de ser manejado por el uno o más procesadores en la implementación de un programa de software. Del mismo modo, los datos que un programa de software determina que deben enviarse desde el dispositivo informático 600 a una entidad externa a través de una de las interfaces del sistema informático, o almacenarse en un elemento de almacenamiento interno, a menudo se ponen en cola temporalmente en la memoria de sistema 608 antes de transmitirse o almacenarse.

Además, por ejemplo, se puede utilizar un ICH para garantizar que dichos datos se pasen adecuadamente entre la memoria del sistema 608 y su interfaz de sistema informático correspondiente apropiada (y el dispositivo de almacenamiento interno si el sistema informático está diseñado de este modo) y puede tener enlaces punto a punto bidireccionales entre sí mismo y las fuentes/dispositivos de E/S observados 604. Del mismo modo, se puede utilizar un MCH para gestionar las varias solicitudes en contienda para los accesos a la memoria de sistema 608 entre la CPU 612 y la GPU 614, las interfaces y los elementos de almacenamiento interno que puedan surgir en el tiempo próximo de una con la otra.

Las fuentes de E/S 604 pueden incluir uno o más dispositivos de E/S que se implementan para transferir datos hacia y/o desde el dispositivo informático 600 (por ejemplo, un adaptador de red); o, para un almacenamiento no volátil a gran escala dentro del dispositivo informático 600 (por ejemplo, unidad de disco duro). Se puede utilizar un dispositivo de entrada de usuario, incluidas claves alfanuméricas y de otro tipo, para comunicar información y ordenar selecciones a la GPU 614. Otro tipo de dispositivo de entrada de usuario es el control del cursor, tal como un ratón, una bola de seguimiento, una pantalla táctil, un panel táctil o teclas de dirección del cursor para comunicar información de dirección y selecciones de órdenes a la GPU 614 y para controlar el movimiento del cursor en el dispositivo de visualización. Se pueden emplear conjuntos de cámaras y micrófonos del dispositivo informático 600 para observar gestos, grabar audio y vídeo y recibir y transmitir órdenes visuales y de audio.

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

La(s) interfaz(s) de red puede(n) proporcionar acceso a una LAN, por ejemplo, conforme a las normas IEEE 802.1 1b y/o IEEE 802.11g, y/o la interfaz de red inalámbrica puede proporcionar acceso a una red de área personal, por ejemplo, conforme a las normas Bluetooth. También pueden ser compatibles otras interfaces y/o protocolos de red inalámbrica, incluidas versiones anteriores y posteriores de las normas. Además de, o en lugar de, la comunicación por medio de las normas de LAN inalámbrica, la(s) interfaz(es) de red puede(n) proporcionar comunicación inalámbrica utilizando, por ejemplo, protocolos de acceso múltiple por división de tiempo (TDMA), protocolos de sistemas globales para comunicaciones móviles (GSM), protocolos de acceso múltiple por división de código (CDMA) y/o cualquier otro tipo de protocolos de comunicación inalámbrica.

La(s) interfaz(es) de red puede(n) incluir una o más interfaces de comunicación, tales como un módem, una tarjeta de interfaz de red u otros dispositivos de interfaz conocidos, tales como los usados para la conexión a Ethernet, anillo con paso de testigo u otros tipos de uniones físicas cableadas o inalámbricas con fines de proporcionar un enlace de comunicación para admitir una LAN o una WAN, por ejemplo. De esta manera, el sistema informático también puede estar acoplado a una serie de dispositivos periféricos, clientes, superficies de control, consolas o servidores por medio de una infraestructura de red convencional, incluida una Intranet o Internet, por ejemplo.

Se debe apreciar que para determinadas implementaciones puede preferirse un sistema menos o más equipado que el ejemplo descrito anteriormente. Por consiguiente, la configuración del dispositivo informático 600 puede variar de una implementación a otra dependiendo de numerosos factores, tales como limitaciones de precio, requisitos de rendimiento, mejoras tecnológicas u otras circunstancias. Ejemplos del dispositivo electrónico o sistema informático 600 pueden incluir (sin limitación) un dispositivo móvil, un asistente digital personal, un dispositivo informático móvil, un teléfono inteligente, un teléfono celular, un auricular, un buscapersonas unidireccional, un buscapersonas bidireccional, un dispositivo de mensajería, un ordenador, un ordenador personal (PC), un ordenador de escritorio, un ordenador portátil, un ordenador manual, un ordenador portátil, una tableta, un servidor, un conjunto de servidores o granja de servidores, un servidor web, un servidor de red, un servidor de Internet, una estación de trabajo, un miniordenador, un ordenador principal, un superordenador, un dispositivo de red, un dispositivo web, un sistema informático distribuido, sistemas multiprocesador, sistemas basados en procesadores, electrónica de consumo, electrónica de consumo programable, televisión, televisión digital, decodificador, punto de acceso inalámbrico, estación base, estación de abonado, centro de abonado móvil, controlador de red de radio, enrutador, concentrador, pasarela, puente, conmutador, máquina o combinaciones de los mismos.

Las formas de realización se pueden implementar como cualquiera o una combinación de: uno o más microchips o circuitos integrados interconectados utilizando una placa base, lógica cableada, software almacenado por un dispositivo de memoria y ejecutado por un microprocesador, firmware, un circuito integrado de aplicación específica (ASIC), y/o un conjunto de puertas programables en campo (FPGA). El término "lógica" puede incluir, a modo de ejemplo, software o hardware y/o combinaciones de software y hardware.

Se pueden proporcionar formas de realización, por ejemplo, como un producto de programa informático que puede incluir uno o más medios legibles por máquina que tienen almacenados en los mismos instrucciones ejecutables por máquina que, cuando se ejecutan por una o más máquinas tales como un ordenador, una red de ordenadores u otros dispositivos electrónicos, pueden dar como resultado que la una o más máquinas lleven a cabo operaciones de acuerdo con las formas de realización descritas en la presente memoria. Un medio legible por máquina puede incluir, entre otros, disquetes, discos ópticos, CD-ROM (memorias de sólo lectura en disco compacto) y discos magnetoópticos, ROM, RAM, EPROM (memorias de sólo lectura programables y borrables), EEPROM (memorias de sólo lectura programables y borrables eléctricamente), tarjetas magnéticas u ópticas, memoria flash u otro tipo de soporte/medio legible por máquina adecuado para almacenar instrucciones ejecutables por máquina.

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

Las operaciones de multiplicación de matrices dispersas son importantes en diversas aplicaciones, incluidas las redes neuronales. Una matriz dispersa es un conjunto en la que la mayoría de los elementos son cero (o algún otro valor matemáticamente irrelevante). Las matrices dispersas suelen ser el resultado de datos de imagen recibidos que indican que una imagen (o región de imagen) incluye información que no es útil. Una multiplicación del conjunto normalmente se lleva a cabo utilizando un método bloqueado, según se muestra en la **Figura 7A**. Por lo tanto, una GPU tradicional toma dos bloques de matriz de entrada como entradas y produce un bloque de matriz de salida. Sin embargo, cuando se opera con matrices dispersas, estos bloques de entrada incluyen en su mayoría valores cero que no contribuyen a los resultados acumulados en el conjunto de salida (por ejemplo, multiplicar por cero produce cero). Según una forma de realización, el mecanismo 610 de procesamiento de matrices dispersas incluye un planificador 613 que identifica dinámicamente operandos que tienen valores cero en un conjunto que se está procesando.

La **Figura 7B** ilustra una forma de realización de dicho planificador 613 incluido dentro de un elemento de procesamiento de GPU 700. Como se muestra en la **Figura 7B,** El elemento de procesamiento 700 incluye lógica 701 para leer operandos incluidos en instrucciones recibidas. También se incluyen la unidad de cálculo 702 y la lógica de

resultados de escritura 703. En una forma de realización, el planificador 613 detecta e identifica ubicaciones de memoria de operandos que tienen valores cero. En una forma de realización de este tipo, el planificador 613 recupera valores de operandos almacenados de la memoria (o memoria caché) a medida que se reciben instrucciones en la GPU 614.

Una vez recuperado, se determina si el valor de un operando es cero. Tras la determinación de que el valor de un operando es cero, el planificador 613 impide la programación de multiplicación de esos operandos en la unidad de multiplicación 702. En consecuencia, sólo se programan y procesan operandos distintos de cero en la unidad de cálculo 702, mientras que el planificador 703 escribe un valor cero para escribir la lógica de resultados 703 para operandos cero. Aunque se muestra que reside dentro de la lógica 701, otras formas de realización pueden presentar un planificador 703 que reside externo a la lógica 701.

En una forma de realización adicional, el mecanismo de procesamiento de matriz dispersa 610 incluye además un rastreador de patrones dispersos 615 para detectar uno o más segmentos dispersos de datos (por ejemplo, patrones de dispersión) dentro de un bloque de datos almacenado, y utiliza los patrones para convertir cálculos de conjuntos posiblemente densos en cálculos dispersos. En una forma de realización, el rastreador de patrones dispersos 615 detecta patrones de dispersión en datos (por ejemplo, datos de imágenes) que están almacenados en la memoria/caché.

Se espera que los futuros sistemas de aprendizaje profundo almacenen miles de millones de imágenes que serán procesadas. Normalmente, una imagen se puede dividir en segmentos que representan información útil frente otra sin importancia. Por ejemplo, si la mitad de la imagen está vacía, es posible rastrear esta información en el nivel de memoria (por ejemplo, a través de un controlador de memoria), en el nivel de página (en el sistema operativo) o en el nivel de jerarquía de memoria caché). Esta información es útil durante la ejecución de una aplicación para eliminar la realización de operaciones de cálculo para segmentos vacíos sin importancia (o dispersos).

La **Figura 7C** ilustra una forma de realización de un rastreador de patrones dispersos 615, que incluye lógica de reconocimiento de patrones 708 y segmentos dispersos (o lógica de segmentos) 709. De acuerdo con una forma de realización, la lógica de reconocimiento de patrones 708 lleva a cabo una operación de cuadro delimitador en un bloque de datos (por ejemplo, datos de imágenes) buscando datos de imágenes almacenados en la memoria para determinar una similitud de varios segmentos dentro del bloque de datos. Los segmentos de datos dentro de un cuadro delimitador que tienen el mismo valor se pueden considerar datos dispersos.

En una forma de realización, la lógica de reconocimiento de patrones 708 se coordina con un controlador de memoria para rastrear los datos almacenados en un dispositivo de memoria. En otras formas de realización, la lógica de reconocimiento de patrones 708 rastrea la información en el nivel de jerarquía de memoria caché. Aún en otras formas de realización, la lógica de reconocimiento de patrones 708 rastrea la información en el nivel de tabla de páginas a través del sistema operativo 606. En una forma de realización adicional, la lógica de reconocimiento de patrones 708 se puede implementar para analizar grandes volúmenes de datos densos para determinar segmentos que pueden procesarse como operaciones dispersas. Como resultado, la lógica de segmento 709 registra las ubicaciones de dirección de segmentos dispersos identificados por la lógica de reconocimiento de patrones 708. En una forma de realización, los segmentos dispersos 709 comprenden punteros a los componentes del segmento disperso. Según se ha descrito anteriormente, las multiplicaciones de conjuntos para operaciones dispersas se pueden omitir, reduciendo de este modo la carga de procesamiento en la GPU 614.

Aún en una forma de realización adicional, el mecanismo de procesamiento de matrices dispersas 610 incluye lógica de compresión 617 para comprimir matrices dispersas. En una forma de realización de este tipo, las representaciones de matrices dispersas comprimidas se generan dinámicamente basándose en un índice disperso (por ejemplo, definido por un % de entradas distintas de cero en el conjunto). En esta forma de realización, el formato comprimido de matriz dispersa se puede representar con un valor distinto de cero señalado por el índice de fila y columna.

Según una forma de realización, la lógica de compresión 617 recibe el segmento disperso determinado por la lógica de reconocimiento de patrones 708 y determina si los datos cumplen un umbral predeterminado para ser considerados dispersos. Por ejemplo, un conjunto MxN puede considerarse escasa si se determina que un número Y de entradas dentro del conjunto son valores cero. La lógica de compresión 617 comprime conjuntos que se determina que son dispersos y almacena los conjuntos comprimidos en una memoria intermedia comprimida dispersa para su ejecución en la GPU 614.

La **Figura 7D** ilustra una forma de realización de la GPU 614 que incluye una memoria intermedia comprimida dispersa 712 y varias unidades de ejecución (EU) 710. En una forma de realización, la memoria intermedia 712 comprimida dispersa incluye entradas de almacenamiento de matriz dispersa comprimidas 712(0) - 712(n) que son procesadas por los EU 710. En una forma de realización de este tipo, la lógica de compresión 717 almacena matrices dispersas utilizadas con frecuencia. Antes del procesamiento por parte de los EU 710, la lógica de compresión 617 descomprime un conjunto comprimido nuevamente a su formato original. En una forma de realización, todos los EU 710 pueden utilizar los mismos conjuntos comprimidos para los subprocesos que se van a ejecutar. Sin embargo, en otras formas de realización, cada EU 710 puede utilizar un matriz dispersa única para los cálculos. Por lo tanto, las matrices

dispersas a las que se accede con frecuencia se almacenan y leen localmente para evitar la transmisión de datos desde la memoria caché a través de una interconexión larga.

La GPU 614 se puede implementar para llevar a cabo otras operaciones de aprendizaje profundo. Por ejemplo, la GPU 614 puede llevar a cabo procesamiento de capas de redes neuronales. Un patrón que se ejecuta con frecuencia en casi todas las redes neuronales profundas es que una capa de convolución (C) sigue una capa de desvío (B) seguida de una capa de unidad lineal rectificada (ReLu (R)) seguida de una capa de agrupación (P). Hoy en día, la mayoría de los sistemas ejecutan estas capas una tras otra (por ejemplo, en las GPU, C, B, R y P se asignan como núcleos individuales) o se asignan fusionando CBR seguido de P como dos núcleos separados.

En ambas instancias, se necesita más de una invocación al núcleo, lo que supone una sobrecarga adicional en la transferencia de datos. De acuerdo con una forma de realización, la GPU 614 se configura de tal forma que los UE se subdividen y asignan para llevar a cabo ciertas operaciones, y tienen resultados intermedios reenviados entre ellos para conseguir un alto rendimiento. La **Figura 7E** ilustra una forma de realización de la GPU 614 que tiene EU subdividido 720.

Según se muestra en la **Figura 7E,** los EU 720(1) - 720(10) se asignan para ejecutar subprocesos de capa de convolución, mientras que los EU 720(11) - 720(13), EU 720(14) - 720(16) y EU 720(17) - 720(19) llevan a cabo, desvían, ReLu y agrupan, respectivamente, la ejecución de subprocesos en capas. Además, se reenvían los datos entre la capa EU 720. Por ejemplo, los datos de C pueden enviarse a la jerarquía de memoria caché de B tan pronto como se complete, configurando un canal.

Según una forma de realización, la subdivisión y asignación de EU 720 se puede establecer de antemano basándose en el conocimiento del dominio. En una forma de realización de este tipo, los mecanismos de cálculo EU 720 se pueden dividir estáticamente, de tal forma que la asignación de EU permanezca igual durante la vida útil de una aplicación específica. En otras formas de realización, los EU 720 se pueden subdividir de forma óptima para cada invocación de ejecución de la GPU 614. Aún en otras formas de realización, la configuración puede ser dinámica de tal forma que se cambie por grupo de subprocesos durante el envío. En otras formas de realización más, la subdivisión se puede implementar para llevar a cabo el procesamiento de otros tipos de capas de red neuronal, además de las capas C, B, R y P, determinando un patrón común y configurando un canal para ejecutarlas más rápido en la GPU en lugar de realizarlos de forma individual.

Visión general del aprendizaje automático

Un algoritmo de aprendizaje automático es un algoritmo que puede aprender basándose en un conjunto de datos. Los algoritmos de aprendizaje automático pueden diseñarse para modelar abstracciones de alto nivel dentro de un conjunto de datos. Por ejemplo, los algoritmos de reconocimiento de imágenes pueden utilizarse para determinar a cuál de varias categorías pertenece una entrada dada; los algoritmos de regresión pueden dar como resultado un valor numérico dada una entrada; y los algoritmos de reconocimiento de patrones pueden utilizarse para generar texto traducido o llevar a cabo traducción de texto a voz y/o reconocimiento de voz.

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

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

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

La **Figura 8** es un diagrama generalizado de una pila de software de aprendizaje automático 800. Una aplicación de aprendizaje automático 802 se puede configurar para entrenar una red neuronal utilizando un conjunto de datos de entrenamiento o para utilizar una red neuronal profunda entrenada para implementar inteligencia automática. La aplicación de aprendizaje automático 802 puede incluir una funcionalidad de entrenamiento y de inferencia para una red neuronal y/o software especializado que se puede utilizar para entrenar una red neuronal antes del despliegue. La aplicación de aprendizaje automático 802 puede implementar cualquier tipo de inteligencia automática incluyendo, entre otros, reconocimiento de imágenes, asignación y localización, navegación autónoma, síntesis de habla, formación de imágenes médicas o traducción de idioma.

La aceleración de hardware para la aplicación de aprendizaje automático 802 se puede habilitar a través de una estructura de aprendizaje automático 804. La estructura de trabajo de aprendizaje automático 804 puede proporcionar una biblioteca de primitivas de aprendizaje automático. Las primitivas de aprendizaje automático son operaciones básicas que comúnmente llevan a cabo los algoritmos de aprendizaje automático. Sin la estructura de aprendizaje automático 804, los desarrolladores de algoritmos de aprendizaje automático tendrían que crear y optimizar la lógica de cálculo principal asociada con el algoritmo de aprendizaje automático y a continuación, volver a optimizar la lógica de cálculo a medida que se desarrollan nuevos procesadores paralelos. En su lugar, la aplicación de aprendizaje automático se puede configurar para llevar a cabo los cálculos necesarios utilizando las primitivas proporcionadas por la estructura de aprendizaje automático 804. Las primitivas de ejemplo incluyen convoluciones tensoriales, funciones de activación y agrupación, que son operaciones de cálculo que se llevan a cabo mientras se entrena una red neuronal convolucional (CNN). La estructura de aprendizaje automático 804 también puede proporcionar primitivas para implementar subprogramas básicos de álgebra lineal llevados a cabo por muchos algoritmos de aprendizaje automático, como operaciones matriciales y vectoriales.

La estructura de aprendizaje automático 804 puede procesar los datos de entrada recibidos de la aplicación de aprendizaje automático 802 y generar la entrada apropiada para una estructura de cálculo 806. La estructura de cálculo 806 puede abstraer las instrucciones subyacentes proporcionadas al controlador GPGPU 808 para permitir que la estructura de aprendizaje automático 804 aproveche la aceleración de hardware a través del hardware GPGPU 810 sin requerir que la estructura de aprendizaje automático 804 tenga un conocimiento íntimo de la arquitectura del hardware GPGPU 810. Además, la estructura de cálculo 806 puede permitir la aceleración de hardware para la estructura de aprendizaje automático 804 a través de una variedad de tipos y generaciones del hardware GPGPU 810.

Aceleración de Aprendizaje Automático de GPGPU

La **Figura 9** ilustra una unidad de procesamiento gráfico de propósito general altamente paralela 900, de acuerdo con una forma de realización. En una forma de realización, la unidad de procesamiento de propósito general (GPGPU) 900 se puede configurar para ser particularmente eficiente en el procesamiento del tipo de cargas de trabajo de cálculo asociadas con el entrenamiento de redes neuronales profundas. Además, la GPGPU 900 puede conectarse directamente a otras instancias de la GPGPU para crear un clúster multi-GPU que mejore la velocidad de entrenamiento de redes neuronales particularmente profundas.

La GPGPU 900 incluye una interfaz anfitrión 902 para permitir una conexión con un procesador anfitrión. En una forma de realización, la interfaz anfitrión 902 es una interfaz PCI Express. Sin embargo, la interfaz anfitrión también puede ser una interfaz de comunicaciones o una estructura de comunicaciones específica del proveedor. La GPGPU 900 recibe órdenes del procesador anfitrión y utiliza un planificador global 904 para distribuir los subprocesos de ejecución asociados con esas órdenes a un conjunto de clústeres de cálculo 906AH. Los clústeres de cálculo 906A-H comparten una memoria caché 908. La memoria caché 908 puede servir como una memoria caché de nivel superior para memorias caché dentro de los clústeres de cálculo 906A-H.

La GPGPU 900 incluye memoria 914A-B acoplada con los clústeres de cálculo 906AH a través de un conjunto de controladores de memoria 912A-B. En varias formas de realización, la memoria 914A-B puede incluir varios tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámico (DRAM) o memoria de acceso aleatorio gráfico, como memoria de acceso aleatorio gráfico síncrono (SGRAM), incluyendo memoria de doble velocidad de datos gráficos (GDDR). En una forma de realización, las unidades de memoria 224A-N también pueden incluir memoria apilada 3D, incluyendo, entre otras, memoria de alto ancho de banda (HBM).

En una forma de realización, cada clúster de cálculo GPLAB06A-H incluye un conjunto de multiprocesadores de gráficos, como el multiprocesador de gráficos 400 de la Figura 4A. Los multiprocesadores de gráficos del clúster de cálculo tienen múltiples tipos de unidades de lógica de enteros y de coma flotante que pueden llevar a cabo operaciones de cálculo con un intervalo de precisiones que incluyen unas adecuadas para cálculos de aprendizaje automático. Por ejemplo, y en una forma de realización, al menos un subconjunto de las unidades de coma flotante en cada uno de los clústeres de cálculo 906A-H se puede configurar para llevar a cabo operaciones de coma flotante de 16 o 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante se puede configurar para llevar a cabo operaciones de coma flotante de 64 bits.

Se pueden configurar varias instancias de GPGPU 900 para que funcionen como un clúster de cálculo. El mecanismo de comunicación utilizado por el clúster de cálculo para la sincronización y el intercambio de datos varía según la forma de realización. En una forma de realización, las múltiples instancias de la GPGPU 900 se comunican a través de la interfaz anfitrión 902. En una forma de realización, la GPGPU 900 incluye un concentrador de E/S 908 que acopla la GPGPU 900 con un enlace de GPU 910 que permite una conexión directa a otras instancias de la GPGPU. En una forma de realización, el enlace de GPU 910 se acopla a un puente de GPU a GPU dedicado que permite la comunicación y sincronización entre múltiples instancias de la GPGPU 900. En una forma de realización, el enlace GPU 910 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En una forma de realización, las múltiples instancias de la GPGPU 900 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red al que se puede acceder a través de la interfaz de anfitrión 902. En una forma de realización, el enlace GPU 910 se puede configurar para permitir una conexión a un procesador principal además de o como alternativa a la interfaz de anfitrión 902.

Mientras que la configuración ilustrada de la GPGPU 900 se puede configurar para entrenar redes neuronales, una forma de realización proporciona una configuración alternativa de la GPGPU 900 que se puede configurar para su despliegue dentro de una plataforma de inferencias de alto rendimiento o bajo consumo. En una configuración de inferenciación, la GPGPU 900 incluye menos de los clústeres de cálculo de los clústeres de cálculo 906A-H en relación con la configuración de entrenamiento. Además, la tecnología de memoria asociada con la memoria 914A-B puede diferir entre las configuraciones de inferencia y entrenamiento. En una forma de realización, la configuración de inferencia de la GPGPU 900 puede soportar instrucciones específicas de inferencia. Por ejemplo, una configuración de inferencia puede proporcionar soporte para una o más instrucciones de producto de punto entero de 8 bits, que se utilizan comúnmente durante las operaciones de inferencia para redes neuronales desplegadas.

La **Figura 10** ilustra un sistema informático multi-GPU 1000, de acuerdo con una forma de realización. El sistema informático de múltiples GPU 1000 puede incluir un procesador 1002 acoplado a múltiples GPGPU 1006A-D mediante un conmutador de interfaz anfitrión 1004. El conmutador de interfaz anfitrión 1004, en una forma de realización, es un dispositivo de conmutación PCI Express que acopla el procesador 1002 a un bus PCI Express a través del cual el procesador 1002 se puede comunicar con el conjunto de GPGPU 1006A-D. Cada una de las múltiples GPGPU 1006A-D puede ser una instancia de la GPGPU 900 de la Figura 9. Las GPGPU 1006A-D se pueden interconectar mediante un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 1016. Los enlaces GPU a GPU de alta velocidad pueden conectarse a cada una de las GPGPUs 1006A-D a través de un enlace GPU dedicado, tal como el enlace GPU 910 de la Figura 9. Los enlaces GPU P2P 1016 permiten la comunicación directa entre cada una de las GPGPUs 1006A-D sin necesidad de comunicación a través del bus de interfaz anfitrión al que está conectado el procesador 1002. Con el tráfico GPU-a-GPU dirigido a los enlaces GPU P2P, el bus de interfaz anfitrión permanece disponible para el acceso a la memoria del sistema o para comunicarse con otras instancias del sistema informático multi-GPU 1000, por ejemplo, a través de uno o más dispositivos de red. Aunque en la realización ilustrada las GPGPU 1006A-D se conectan al procesador 1002 mediante el conmutador de interfaz de anfitrión 1004, en una forma de realización, el procesador 1002 incluye el soporte directo para los enlaces de GPU de P2P 1016 y puede conectarse directamente a las GPGPU 1006A-D.

Implementaciones de Red Neuronal de Aprendizaje Automático

La arquitectura informática proporcionada por las formas de realización descritas en la presente memoria se puede configurar para llevar a cabo los tipos de procesamiento paralelo que son particularmente adecuados para el entrenamiento y despliegue de redes neuronales para el aprendizaje automático. Una red neuronal puede generalizarse como una red de funciones que tienen una relación de grafo. Como es bien conocido en el arte, hay una variedad de tipos de implementaciones de redes neuronales utilizadas en el aprendizaje automático. Un tipo de ejemplo de red neuronal es la red feedforward, como se ha descrito anteriormente.

Un segundo tipo de ejemplo de red neuronal es la red neuronal convolucional (CNN). Una CNN es una red neuronal feedforward especializada en el procesamiento de datos con una topología cuadriculada conocida, como los datos de imágenes. En consecuencia, las CNN se utilizan habitualmente para aplicaciones de visión de cálculo y reconocimiento de imágenes, pero también pueden utilizarse para otros tipos de reconocimiento de patrones, como el procesamiento del habla y del lenguaje. Los nodos de la capa de entrada de la CNN se organizan en un conjunto de "filtros" (detectores de características inspirados en los campos receptivos que se encuentran en la retina), y la salida de cada conjunto de filtros se propaga a los nodos de las capas sucesivas de la red. Los cálculos de una CNN incluyen la aplicación de la operación matemática de convolución a cada filtro para producir la salida de ese filtro. La convolución es un tipo

especializado de operación matemática llevada a cabo por dos funciones para producir una tercera función que es una versión modificada de una de las dos funciones originales. En la terminología de las redes convolucionales, la primera función de la convolución se puede denominar entrada, mientras que la segunda función puede denominarse núcleo de convolución. La salida se puede denominar mapa de características. Por ejemplo, la entrada a una capa de convolución puede ser un conjunto multidimensional de datos que define los distintos componentes de color de una imagen de entrada. El núcleo de convolución puede ser un conjunto multidimensional de parámetros, donde los parámetros están adaptados por el proceso de entrenamiento para la red neuronal.

Las redes neuronales recurrentes (RNN) son una familia de redes neuronales feedforward que incluyen conexiones de retroalimentación entre capas. Las RNN permiten modelar datos secuenciales compartiendo datos de parámetros entre distintas partes de la red neuronal. La arquitectura para una RNN incluye ciclos. Los ciclos representan la influencia de un valor actual de una variable sobre su propio valor en un momento futuro, ya que al menos una parte de los datos de salida de la RNN se utiliza como retroalimentación para procesar entradas posteriores en una secuencia. Esta característica hace que las RNN sean especialmente útiles para el procesamiento del lenguaje debido a la naturaleza variable en la que se pueden componer los datos del lenguaje.

Las figuras que se describen a continuación presentan redes feedforward, CNN y RNN de ejemplo, y describen un proceso general para entrenar y implementar respectivamente cada uno de esos tipos de redes. Se entenderá que estas descripciones son de ejemplo y no restrictivas en cuanto a cualquier realización específica descrita en la presente memoria y los conceptos ilustrados se pueden aplicar en general a las redes neuronales profundas y técnicas de aprendizaje automático en general.

Las redes neuronales de ejemplo descritas anteriormente pueden utilizarse para llevar a cabo aprendizaje profundo. El aprendizaje profundo es un aprendizaje automático que utiliza redes neuronales profundas. Las redes neuronales profundas utilizadas en el aprendizaje profundo son redes neuronales artificiales compuestas por múltiples capas ocultas, a diferencia de las redes neuronales superficiales que incluyen una sola capa oculta. El entrenamiento de redes neuronales más profundas es, en general, más intensivo desde el punto de vista del cálculo. Sin embargo, las capas ocultas adicionales de la red permiten el reconocimiento de patrones en varios pasos, lo que reduce el error de salida con respecto a las técnicas de aprendizaje automático poco profundas.

Las redes neuronales profundas utilizadas en el aprendizaje profundo suelen incluir una red front-end para llevar a cabo el reconocimiento de características acopladas a una red back-end que representa un modelo matemático que puede llevar a cabo operaciones (por ejemplo, clasificación de objetos, reconocimiento del habla, etc.) basándose en la representación de características proporcionada al modelo. El aprendizaje profundo permite que el aprendizaje automático se realice sin necesidad de llevar a cabo ingeniería de características a mano para el modelo. En su lugar, las redes neuronales profundas pueden aprender características basadas en la estructura estadística o la correlación dentro de los datos de entrada. Las características aprendidas pueden proporcionarse a un modelo matemático que puede asignar las características detectadas a una salida. El modelo matemático utilizado por la red suele estar especializado para la tarea específica que se va a llevar a cabo, y se utilizarán diferentes modelos para llevar a cabo diferentes tareas.

Una vez estructurada la red neuronal, se le puede aplicar un modelo de aprendizaje para entrenarla para llevar a cabo tareas específicas. El modelo de aprendizaje describe cómo ajustar los pesos dentro del modelo para reducir el error de salida de la red. La retropropagación de errores es un método habitual para entrenar redes neuronales. Se presenta un vector de entrada a la red para su procesamiento. La salida de la red se compara con la salida deseada mediante una función de pérdida y se calcula un valor de error para cada una de las neuronas de la capa de salida. A continuación, los valores de error se propagan hacia atrás hasta que cada neurona tiene un valor de error asociado que representa aproximadamente su contribución a la salida original. A continuación, la red puede aprender de esos errores mediante un algoritmo, como el de descenso de gradiente estocástico, para actualizar los pesos de la red neuronal.

Las **Figuras 11A y 11B** ilustran una red neuronal convolucional de ejemplo. La **Figura 11A** ilustra varias capas dentro de una CNN. Según se muestra en la **Figura 11A**, una CNN de ejemplo utilizada para modelar el procesamiento de imágenes puede recibir una entrada 1102 que describe los componentes rojo, verde y azul (RGB) de una imagen de entrada. La entrada 1102 puede ser procesada por múltiples capas convolucionales (por ejemplo, capa convolucional 1104, capa convolucional 1106). La salida de las múltiples capas convolucionales puede opcionalmente ser procesada por un conjunto de capas totalmente conectadas 1108. Las neuronas en una capa completamente conectada tienen conexiones completas a todas las activaciones en la capa previa, como se describió previamente para una red de alimentación directa. La salida de las capas totalmente conectadas 1108 se puede utilizar para generar un resultado de salida de la red. Las activaciones dentro de las capas totalmente conectadas 908 pueden ser calculadas utilizando multiplicación de conjuntos en lugar de convolución. No todas las implementaciones de CNN hacen uso de capas 1108 totalmente conectadas. Por ejemplo, en algunas implementaciones la capa convolucional 1106 puede generar la salida para la CNN.

Las capas convolucionales están escasamente conectadas, lo que difiere de la configuración de red neuronal tradicional que se encuentra en las capas 1108 totalmente conectadas. Las capas de redes neuronales tradicionales

están totalmente conectadas, de tal forma que cada unidad de salida interactúa con cada unidad de entrada. Sin embargo, las capas convolucionales están escasamente conectadas porque la salida de la convolución de un campo es la entrada (en lugar del valor de estado respectivo de cada uno de los nodos en el campo) a los nodos de la capa subsiguiente, según se ilustra. Los núcleos asociados a las capas convolucionales llevan a cabo operaciones de convolución, cuya salida se envía a la capa siguiente. La reducción de la dimensionalidad llevada a cabo dentro de las capas convolucionales es un aspecto que permite a la CNN escalar para procesar imágenes de gran tamaño.

La **Figura 11B** ilustra fases de cálculo de ejemplo dentro de una capa convolucional de una CNN. La entrada a una capa convolucional 1112 de una CNN puede procesarse en tres fases de una capa convolucional 1114. Las tres fases pueden incluir una fase de convolución 1116, una fase de detector 1118 y una fase de agrupación 1120. La capa convolucional 1114 puede, a continuación, enviar datos a una capa convolucional sucesiva. La capa convolucional final de la red puede generar datos de mapa de características de salida o proporcionar entrada a una capa totalmente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

En la fase de convolución 1116 lleva a cabo varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La fase de convolución 1116 puede incluir una transformación afín, que es cualquier transformación que pueda especificarse como una transformación lineal más una traslación. Las transformaciones afines incluyen rotaciones, traslaciones, escalas y combinaciones de estas transformaciones. La fase de convolución calcula la salida de funciones (por ejemplo, neuronas) que están conectadas a regiones específicas de la entrada, que puede determinarse como la región local asociada a la neurona. Las neuronas calculan un producto punto entre los pesos de las neuronas y la región en la entrada local a la que las neuronas están conectadas. La salida de la fase de convolución 1116 define un conjunto de activaciones lineales que son procesadas por fases sucesivas de la capa convolucional 1114.

Las activaciones lineales pueden ser procesadas por una fase detectora 1118. En la fase de detección 1118, cada activación lineal es procesada por una función de activación no lineal. La función de activación no lineal aumenta las propiedades no lineales de la red global sin afectar a los campos receptivos de la capa de convolución. Pueden usarse varios tipos de funciones de activación no lineal. Un tipo particular es la unidad lineal rectificada (ReLU), que utiliza una función de activación definida como $f(x) = \max(0, x)$, de tal forma que la activación tiene un umbral en cero.

La fase de agrupación 1120 utiliza una función de clúster que reemplaza la salida de la capa convolucional 1106 con un resumen estadístico de las salidas cercanas. La función de agrupación se puede utilizar para introducir invariancia de traducción en la red neuronal, de tal forma que pequeñas traducciones de la entrada no cambien las salidas agrupadas. La invariancia a la traducción local puede ser útil en escenarios en los que la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Se pueden utilizar varios tipos de funciones de agrupación durante la fase de agrupación 1120, incluyendo agrupación máxima, agrupación promedio y agrupación de norma 12. Además, algunas implementaciones de CNN no incluyen una fase de agrupación. En su lugar, dichas implementaciones sustituyen una fase de convolución adicional que tiene un paso aumentado en relación con las fases de convolución anteriores.

A continuación, la salida de la capa convolucional 1114 puede ser procesada por la siguiente capa 1122. La siguiente capa 1122 puede ser una capa convolucional adicional o una de las capas completamente conectadas 1108. Por ejemplo, la primera capa convolucional 1104 de la **Figura 11A** puede enviarse a la segunda capa convolucional 1106, mientras que la segunda capa convolucional puede enviarse a una primera capa de las capas completamente conectadas 1108.

La **Figura 12** ilustra una red neuronal recurrente de ejemplo 1200. En una red neuronal recurrente (RNN), el estado anterior de la red influye en la salida del estado actual de la red. Las RNN pueden crearse en una variedad de maneras utilizando una diversidad de funciones. El uso de las RNN generalmente gira en torno al uso de modelos matemáticos para predecir el futuro basándose en una secuencia previa de entradas. Por ejemplo, una RNN se puede utilizar para llevar a cabo un modelado estadístico del lenguaje para predecir una palabra futura dada una secuencia previa de palabras. La RNN 1200 ilustrada se puede describir con una capa de entrada 1202 que recibe un vector de entrada, capas ocultas 1204 para implementar una función recurrente, un mecanismo de retroalimentación 1205 para habilitar una "memoria" de estados previos, y una capa de salida 1206 para emitir un resultado. La RNN 1200 opera por etapas de tiempo. El estado de la RNN en una etapa de tiempo dada se ve influido por el paso de tiempo anterior a través del mecanismo de retroalimentación 1205. Para una etapa de tiempo dada, el estado de las capas ocultas 1204 se define por el estado previo y la entrada en la etapa de tiempo actual. Una entrada inicial x_1) en una primera etapa de tiempo puede ser procesada por la capa oculta 1204. Una segunda entrada (x_2) puede ser procesada por la capa oculta 1204 utilizando información de estado que es determinada durante el procesamiento de la entrada inicial (x_1). Un estado dado se puede calcular como $s_t = f(Ux_t + Ws_{t-1})$, donde U y W son conjuntos de parámetros. La función f suele ser una no linealidad, como la función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x) = \max(0, x)$. Sin embargo, la función matemática específica utilizada en las capas ocultas 1004 puede variar dependiendo de los detalles específicos de implementación de la RNN 1200.

Además de las redes CNN y RNN básicas descritas, se pueden habilitar variaciones de dichas redes. Una variante de RNN de ejemplo es la RNN de memoria a corto plazo larga (LSTM). Las RNN LSTM son capaces de aprender

dependencias a largo plazo que pueden ser necesarias para procesar secuencias de lenguaje más largas. Una variante de la CNN es una red convolucional de creencia profunda, que tiene una estructura similar a una CNN y se entrena de manera similar a una red de creencia profunda. Una red profunda de creencias (DBN) es una red neuronal generativa compuesta por múltiples capas de variables estocásticas (aleatorias). Las RCP pueden entrenarse capa a capa utilizando aprendizaje no supervisado voraz. A continuación, los pesos aprendidos de la RCP se pueden utilizar para proporcionar redes neuronales entrenadas previamente determinando un conjunto inicial óptimo de pesos para la red neuronal.

La **Figura 13** ilustra el entrenamiento y despliegue de una red neuronal profunda. Una vez que se ha estructurado una red determinada para una tarea, la red neuronal se entrena utilizando un conjunto de datos de entrenamiento 1302. Se han desarrollado varias estructuras de entrenamiento 1304 para posibilitar la aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura de aprendizaje de máquina 804 de la Figura 8 se puede configurar como una estructura de entrenamiento 1304. La estructura de entrenamiento 1304 puede conectarse a una red neuronal no entrenada 1306 y permitir que la red neuronal no entrenada sea entrenada utilizando los recursos de procesamiento paralelo descritos en la presente memoria para generar una red neuronal entrenada 1308.

Para iniciar el proceso de entrenamiento, los pesos iniciales se pueden elegir de forma aleatoria o mediante preentrenamiento utilizando una red de creencias profunda. A continuación, el ciclo de entrenamiento se puede llevar a cabo de manera supervisada o no supervisada.

El aprendizaje supervisado es un método de aprendizaje en el que el entrenamiento se lleva a cabo como una operación mediada, como cuando el conjunto de datos de entrenamiento 1302 incluye una entrada emparejada con la salida deseada para la entrada, o cuando 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. A continuación, los errores se propagan a través del sistema. La estructura de entrenamiento 1304 puede ajustar para ajustar los pesos que controlan la red neuronal no entrenada 1306. La estructura de entrenamiento 1304 puede proporcionar herramientas para supervisar lo bien que la red neuronal no entrenada 1306 está convergiendo hacia un modelo adecuado para generar respuestas correctas basadas en datos de entrada conocidos. El proceso de entrenamiento ocurre de forma repetitiva a medida que los pesos de la red se ajustan para refinar la salida generada por la red neuronal. El proceso de entrenamiento puede continuar hasta que la red neuronal alcanza una precisión estadísticamente deseada asociada con una red neuronal entrenada 1308. A continuación, la red neuronal entrenada 1308 se puede desplegar para implementar cualquier número de operaciones de aprendizaje automático.

El aprendizaje no supervisado es un método de aprendizaje en el que la red intenta entrenarse utilizando datos no etiquetados. Por lo tanto, para el aprendizaje no supervisado el conjunto de datos de entrenamiento 1302 incluirá datos de entrada sin ningún dato de salida asociado. La red neuronal no entrenada 1306 puede aprender clústeres dentro de la entrada no etiquetada y puede determinar cómo se relacionan las entradas individuales con el conjunto de datos global. El entrenamiento no supervisado se puede utilizar para generar un mapa autoorganizado, que es un tipo de red neuronal entrenada 1307 capaz de llevar a cabo operaciones útiles para reducir la dimensionalidad de los datos. El entrenamiento no supervisado también se puede utilizar para llevar a cabo la detección de anomalías, que permite la identificación de puntos de datos en un conjunto de datos de entrada que se desvían de los patrones normales de los datos.

También pueden emplearse variaciones del entrenamiento supervisado y no supervisado. El aprendizaje semisupervisado es una técnica en la que el conjunto de datos de entrenamiento 1302 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que los datos de entrada se utilizan de forma continua para seguir entrenando el modelo. El aprendizaje incremental permite a la red neuronal entrenada 1308 adaptarse a los nuevos datos 1312 sin olvidar el conocimiento inculcado en la red durante el entrenamiento inicial.

Ya sea supervisado o no supervisado, el proceso de entrenamiento de redes neuronales especialmente profundas puede ser demasiado intensivo desde el punto de vista del cálculo para un único nodo de cálculo. En lugar de utilizar un único nodo de cálculo, se puede utilizar una red distribuida de nodos de cálculo para acelerar el proceso de entrenamiento.

La **Figura 14** es un diagrama de bloques que ilustra el aprendizaje distribuido. El aprendizaje distribuido es un modelo de entrenamiento que utiliza múltiples nodos informáticos distribuidos para llevar a cabo un entrenamiento supervisado o no supervisado de una red neuronal. Los nodos de cálculo distribuidos pueden incluir cada uno, uno o más procesadores anfitrión y uno o más de los nodos de procesamiento de propósito general, tales como la unidad de procesamiento gráfico de propósito general altamente paralela 900 como en la Figura 9. Según se ilustra, el aprendizaje distribuido se puede llevar a cabo con el paralelismo de modelo 1402, el paralelismo de datos 1404 o una combinación de modelo y paralelismo de datos 1204.

En el modelo de paralelismo 1402, diferentes nodos de cálculo en un sistema distribuido pueden llevar a cabo 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. Las ventajas del paralelismo de modelos incluyen la capacidad de escalar a modelos particularmente grandes. Dividir los cálculos asociados a las diferentes capas de la red neuronal permite entrenar redes neuronales muy grandes en las que los pesos de todas las capas no cabrían en la memoria de un único nodo de cálculo. En algunas instancias, el paralelismo de modelos puede ser especialmente útil para el entrenamiento no supervisado de grandes redes neuronales.

En el paralelismo de datos 1404, los diferentes nodos de la red distribuida tienen una instancia completa del modelo y cada nodo recibe una parte diferente de los datos. A continuación, se combinan los resultados de los distintos nodos. Aunque son posibles diferentes enfoques para el paralelismo de datos, todos los enfoques de entrenamiento en paralelo de datos requieren una técnica de combinación de resultados y sincronización de los parámetros del modelo entre cada nodo. Algunos ejemplos de enfoques para combinar datos son el promediado de parámetros y el paralelismo de datos basado en actualizaciones. El promediado de parámetros entrena a cada nodo en un subconjunto de los datos de entrenamiento y establece los parámetros globales (por ejemplo, pesos, desvíos) a la media de los parámetros de cada nodo. El promediado de parámetros utiliza un servidor de parámetros central que mantiene los datos de los parámetros. El paralelismo de datos basado en actualizaciones es similar al promediado de parámetros, salvo que en lugar de transferir parámetros de los nodos al servidor de parámetros, se transfieren las actualizaciones del modelo. Además, el paralelismo de datos basado en actualizaciones se puede llevar a cabo de manera descentralizada, donde las actualizaciones se comprimen y transfieren entre nodos.

El paralelismo combinado de modelo y datos 1406 se puede implementar, por ejemplo, en un sistema distribuido en el que cada nodo de cálculo incluye múltiples GPU. Cada nodo puede tener una instancia completa del modelo con GPU separadas dentro de cada nodo que se utilizan para entrenar diferentes partes del modelo.

El entrenamiento distribuido tiene una sobrecarga mayor que el entrenamiento en una sola máquina. Sin embargo, los procesadores paralelos y las GPGPU descritos en la presente memoria pueden implementar diversas técnicas para reducir la sobrecarga del entrenamiento distribuido, incluidas técnicas para permitir la transferencia de datos de GPU a GPU con un gran ancho de banda y la sincronización acelerada de datos remotos.

Aplicaciones de aprendizaje automático de ejemplo

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

El aprendizaje automático acelerado mediante procesadores paralelos tiene aplicaciones de conducción autónoma, como el reconocimiento de carriles y señales de tráfico, la evitación de obstáculos, la navegación y el control de la conducción. Las técnicas de aprendizaje automático acelerado pueden utilizarse para entrenar modelos de conducción basados en conjuntos de datos que definen las respuestas adecuadas a entradas de entrenamiento específicas. Los procesadores paralelos descritos en la presente memoria pueden permitir un entrenamiento rápido de las redes neuronales cada vez más complejas utilizadas para las soluciones de conducción autónoma y permite el despliegue de procesadores de inferencias de baja potencia en una plataforma móvil adecuada para su integración en vehículos autónomos.

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

El aprendizaje automático acelerado por procesador paralelo se puede utilizar 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 sean robustos frente a entradas erróneas o desconocidas. Entre las aplicaciones de ejemplo de los procesadores de lenguaje natural se incluye la traducción automática entre lenguas humanas.

Las plataformas de procesamiento paralelo utilizadas para el aprendizaje automático pueden dividirse en plataformas de entrenamiento y plataformas de despliegue. Las plataformas de entrenamiento suelen ser altamente paralelas e incluyen optimizaciones para acelerar la formación en un único nodo multi-GPU y la formación en múltiples nodos y multi-GPU. Los procesadores paralelos de ejemplo adecuados para el entrenamiento incluyen la unidad de

procesamiento gráfico de uso general altamente paralelo y el sistema informático multi-GPU. Por el contrario, las plataformas de aprendizaje automático implementadas incluyen, en general, procesadores paralelos de potencia inferior adecuados para su utilización en productos tales como cámaras, robots autónomos y vehículos autónomos.

La **Figura 15** ilustra un sistema de ejemplo de inferencia en un microprocesador (SOC) 1500 adecuado para llevar a cabo inferencias utilizando un modelo entrenado. El SOC 1500 puede integrar componentes de procesamiento que incluyen un procesador de medios 1502, un procesador de visión 1504, una GPGPU 1506 y un procesador de múltiples núcleos 1508. El SOC 1500 puede incluir adicionalmente una memoria en microprocesador 1505 que puede permitir una agrupación de datos en microprocesador compartida que es accesible por cada uno de los componentes de procesamiento. Los componentes de procesamiento pueden optimizarse para un funcionamiento de bajo consumo que permita su despliegue en una variedad de plataformas de aprendizaje automático, incluyendo vehículos autónomos y robots autónomos. Por ejemplo, una implementación del SOC 1500 se puede utilizar como parte del sistema de control principal de un vehículo autónomo. Cuando el SOC 1500 se configura para su utilización en vehículos autónomos, el SOC se diseña y configura para cumplir las normas de seguridad funcional pertinentes de la jurisdicción de despliegue.

Durante el funcionamiento, el procesador de medios 1502 y el procesador de visión 1504 pueden trabajar conjuntamente para acelerar las operaciones de visión por ordenador. El procesador de medios 1502 puede permitir la decodificación de baja latencia de múltiples secuencias de vídeo de alta resolución (por ejemplo, 4K, 8K). Las secuencias de vídeo descodificadas se pueden escribir en una memoria intermedia en la memoria en microprocesador 1505. A continuación, el procesador de visión 1304 puede analizar el vídeo decodificado y llevar a cabo operaciones de procesamiento preliminares en los fotogramas del vídeo decodificado en preparación para procesar los fotogramas utilizando un modelo de reconocimiento de imagen entrenado. Por ejemplo, el procesador de visión 1504 puede acelerar las operaciones de convolución para una CNN que se utiliza para llevar a cabo el reconocimiento de imágenes en los datos de vídeo de alta resolución, mientras que los cálculos del modelo back-end son realizados por la GPGPU 1506.

El procesador multinúcleo 1508 puede incluir lógica de control para ayudar a secuenciar y sincronizar las transferencias de datos y las operaciones de memoria compartida llevadas a cabo por el procesador de medios 1502 y el procesador de visión 1504. El procesador multinúcleo 1308 también puede funcionar como un procesador de aplicaciones para ejecutar aplicaciones de software que pueden hacer uso de la capacidad de cálculo de inferencia de la GPGPU 1506. Por ejemplo, al menos una parte de la lógica de navegación y conducción se puede implementar en el software que se ejecuta en el procesador multinúcleo 1508. Dicho software puede emitir directamente cargas de trabajo de cálculo a la GPGPU 1506 o las cargas de trabajo de cálculo se pueden emitir al procesador multinúcleo 1508, que puede descargar al menos una parte de dichas operaciones a la GPGPU 1506.

La GPGPU 1506 puede incluir clústeres de cálculo, tales como una configuración de baja potencia de los grupos de cálculo 906A-906H dentro de la unidad de procesamiento gráfico de propósito general altamente paralela 900. Los clústeres de cálculo dentro de la GPGPU 1506 pueden soportar instrucción que está específicamente optimizada para llevar a cabo cómputos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1506 puede soportar instrucciones para llevar a cabo cálculos de baja precisión tales como operaciones vectoriales de enteros de 8 y 4 bits.

Sistema de procesamiento gráfico de ejemplo adicional

Los detalles de las formas de realización descritas anteriormente se pueden incorporar a los sistemas y dispositivos de procesamiento gráfico que se describen a continuación. El sistema y dispositivos de procesamiento gráfico de las **Figuras 16-29** ilustran sistemas y hardware de procesamiento gráfico alternativos que pueden implementar todas y cada una de las técnicas descritas anteriormente.

Vista general de otros sistemas de procesamiento gráfico de ejemplo

La **Figura 16** es un diagrama de bloques de un sistema de procesamiento 1600, de acuerdo con una forma de realización. En diversas formas de realización, el sistema 1600 incluye uno o más procesadores 1602 y uno o más procesadores gráficos 1608, y puede ser un sistema de escritorio de procesador único, un sistema de estación de trabajo multiprocesador, o un sistema de servidor que tiene un gran número de procesadores 1602 o núcleos de procesador 1607. En una forma de realización, el sistema 1600 es una plataforma de procesamiento incorporada dentro de un circuito integrado de sistema en un microprocesador (SoC) para su utilización en dispositivos móviles, de mano o integrados.

Una forma de realización del sistema 1600 puede incluir, o estar incorporado dentro de una plataforma de juegos basada en servidor, una consola de juegos, incluyendo una consola de juegos y multimedia, una consola de juegos móvil, una consola de juegos portátil o una consola de juegos en línea. En algunas formas de realización, el sistema 1600 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 1600 también puede incluir, acoplarse o integrarse en un dispositivo para llevar puesto, como un dispositivo para llevar puesto de reloj inteligente, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunas formas de realización, el sistema de

procesamiento de datos 1600 es un dispositivo de televisión o descodificador que tiene uno o más procesadores 1602 y una interfaz gráfica generada por uno o más procesadores gráficos 1608.

En algunas formas de realización, el uno o más procesadores 1602 incluyen cada uno, uno o más núcleos de procesador 1607 para procesar instrucciones que, cuando se ejecutan, llevan a cabo operaciones para el sistema y el software de usuario. En algunas formas de realización, cada uno del uno o más núcleos de procesador 1607 se configura para procesar un conjunto de instrucciones 1609 específico. En algunas formas de realización, el conjunto de instrucciones 1609 puede facilitar el cálculo del conjunto de instrucciones complejas (CISC), la cálculo de conjunto de instrucciones reducidas (RISC), o la cálculo a través de una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 1607 pueden procesar cada uno un conjunto de instrucciones 1609 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo del procesador 1607 también puede incluir otros dispositivos de procesamiento, como un procesador de señales digitales (DSP).

En algunas formas de realización, el procesador 1602 incluye memoria caché 1604. Dependiendo de la arquitectura, el procesador 1602 puede tener una única memoria caché interna o múltiples niveles de memoria caché interna. En algunas formas de realización, la memoria caché se comparte entre varios componentes del procesador 1602. En algunas formas de realización, el procesador 1602 también utiliza una memoria caché externa (por ejemplo, una memoria caché de nivel 3 (L3) o una memoria caché de último nivel (LLC)) (no mostrada), que puede ser compartida entre los núcleos de procesador 1607 utilizando técnicas de coherencia de memoria caché conocidas. Un archivo de registro 1606 se incluye adicionalmente en el procesador 1602 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de 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 para el diseño del procesador 1602.

En algunas formas de realización, el procesador 1602 se acopla con un bus de procesador 1610 para transmitir señales de comunicación tales como señales de dirección, datos o control entre el procesador 1602 y otros componentes del sistema 1600. En una forma de realización, el sistema 1600 utiliza una arquitectura de sistema "concentrador" de ejemplo, que incluye un concentrador controlador de memoria 1616 y un concentrador controlador de entrada-salida (E/S) 1630. Un concentrador controlador de memoria 1616 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 1600, mientras que un concentrador controlador de E/S (ICH) 1630 proporciona conexiones a dispositivos de E/S por medio de un bus de E/S local. En una forma de realización, la lógica del concentrador controlador de memoria 1616 se integra dentro del procesador.

El dispositivo de memoria 1620 puede ser un dispositivo de memoria de acceso aleatorio dinámico (DRAM), un dispositivo de memoria de acceso aleatorio estático (SRAM), un dispositivo de memoria flash, un dispositivo de memoria de cambio de fase, o algún otro dispositivo de memoria que tenga un rendimiento adecuado para servir como memoria de proceso. En una forma de realización, el dispositivo de memoria 1620 puede operar como memoria de sistema para el sistema 1600, para almacenar datos 1622 e instrucciones 1621 para su utilización cuando el uno o más procesadores 1602 ejecutan una aplicación o proceso. El concentrador controlador de memoria 1616 también se acopla con un procesador gráfico externo opcional 1612, que se puede comunicar con el uno o más procesadores gráficos 1608 en los procesadores 1602 para llevar a cabo operaciones gráficas y de medios.

En algunas formas de realización, el ICH 1630 permite a los periféricos conectarse al dispositivo de memoria 1620 y al procesador 1602 a través de un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, entre otros, un controlador de audio 1646, una interfaz de firmware 1628, un transceptor inalámbrico 1626 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 1624 (por ejemplo, unidad de disco duro, memoria flash, etc.), y un controlador de E/S heredado 1640 para acoplar dispositivos heredados (por ejemplo, Personal System 2 (PS/2)) al sistema. Uno o más controladores de Bus Serial Universal (USB) 1642 conectan dispositivos de entrada, tales como combinaciones de teclado y ratón 1644. Un controlador de red 1634 también se puede acoplar con el ICH 1630. En algunas formas de realización, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de procesador 1610. Se apreciará que el sistema 1600 mostrado sea de ejemplo y no restrictivo, ya que también se pueden utilizar otros tipos de sistemas de procesamiento de datos configurados de forma diferente. Por ejemplo, el concentrador controlador de E/S 1630 puede estar integrado dentro del uno o más procesadores 1602, o el concentrador controlador de memoria 1616 y el concentrador controlador de E/S 1630 se pueden integrar en un procesador gráfico externo discreto, tal como el procesador gráfico externo 1612.

La **Figura 17** es un diagrama de bloques de una forma de realización de un procesador 1700 que tiene uno o más núcleos de procesador 1702A-1702N, un controlador de memoria integrado 1714 y un procesador gráfico integrado 1708. Aquellos elementos de la **Figura 17** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura de la presente memoria pueden operar o funcionar de cualquier manera similar a la descrita en otra parte de la presente memoria, pero no se limitan a los mismas. El procesador 1700 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 1702N representado por los recuadros rayados. Cada uno de los núcleos de procesador 1702A-1702N incluye una o más unidades de memoria caché internas 1704A-1704N. En algunas formas de realización, cada núcleo de procesador también tiene acceso a una o más unidades de memoria caché compartidas 1706.

Las unidades de memoria caché internas 1704A-1704N y las unidades de memoria caché compartidas 1706 representan una jerarquía de memoria caché dentro del procesador 1700. La jerarquía de memoria caché puede incluir al menos un nivel de memoria caché de instrucciones y datos dentro de cada núcleo de procesador y uno o más niveles de memoria caché de nivel medio compartida, tal como un Nivel 2 (L2), Nivel 3 (L3), Nivel 4 (L4) u otros niveles de memoria caché, donde el nivel más alto de memoria caché antes de la memoria externa se clasifica como LLC. En algunas formas de realización, la lógica de coherencia de memoria caché mantiene la coherencia entre las varias unidades de memoria caché 1706 y 1704A-1704N.

En algunas formas de realización, el procesador 1700 también puede incluir un conjunto de una o más unidades de controlador de bus 1716 y un núcleo de agente de sistema 1710. Las una o más unidades controladoras de bus 1716 gestionan un conjunto de buses periféricos, tales como uno o más buses de Interconexión de Componentes Periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 1710 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunas formas de realización, el núcleo de agente de sistema 1710 incluye uno o más controladores de memoria integrados 1714 para gestionar el acceso a varios dispositivos de memoria externos (no mostrados).

En algunas formas de realización, uno o varios de los núcleos de procesador 1702A a 1702N incluyen la admisión de múltiples subprocesos simultáneos. En una forma de realización de este tipo, el núcleo de agente de sistema 1710 incluye componentes para coordinar y hacer funcionar los núcleos 1702A-1702N durante el procesamiento de múltiples subprocesos. El núcleo de agente de sistema 1710 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 1702A-1702N y el procesador gráfico 1708.

En algunas formas de realización, el procesador 1700 incluye adicionalmente el procesador gráfico 1708 para ejecutar operaciones de procesamiento gráfico. En algunas formas de realización, el procesador de gráficos 1708 se acopla con el conjunto de unidades de memoria caché compartidas 1706, y el núcleo de agente de sistema 1710, incluyendo el uno o más controladores de memoria integrados 1714. En algunas formas de realización, un controlador de visualización 1711 se acopla con el procesador de gráficos 1708 para conducir la salida del procesador de gráficos a una o más pantallas acopladas. En algunas formas de realización, el controlador de visualización 1711 puede ser un módulo separado acoplado con el procesador de gráficos a través de al menos una interconexión, o puede estar integrado dentro del procesador de gráficos 1708 o del núcleo de agente de sistema 1710.

En algunas formas de realización, se utiliza una unidad de interconexión 1712 basada en anillo para acoplar los componentes internos del procesador 1700. Sin embargo, se puede utilizar una unidad de interconexión alternativa, como una interconexión punto a punto, una interconexión conmutada u otras técnicas, incluidas técnicas bien conocidas en la técnica. En algunas formas de realización, el procesador gráfico 1708 se acopla con la interconexión en anillo 1712 a través de un enlace de E/S 1713.

El enlace de E/S de ejemplo 1713 representa al menos una de las múltiples variedades de interconexiones de E/S, incluyendo una interconexión de E/S en paquete que facilita la comunicación entre varios componentes del procesador y un módulo de memoria integrado de alto rendimiento 1718, tal como un módulo eDRAM. En algunas formas de realización, cada uno de los núcleos de procesador 1702A-1702N y el procesador gráfico 1708 utilizan módulos de memoria integrados 1718 como una memoria caché de último nivel compartida.

En algunas formas de realización, los núcleos de procesador 1702A-1702N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 1702A-1702N son heterogéneos en lo que respecta a la arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 1702A-1702N 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 forma de realización, los núcleos de procesador 1702A-1702N son heterogéneos en lo que respecta a la microarquitectura, donde uno o más núcleos que tienen un consumo de energía relativamente superior están acoplados a uno o más núcleos de potencia que tienen un menor consumo de energía. Además, el procesador 1700 se puede implementar en uno o varios microprocesadores o como un circuito integrado de SoC que tiene los componentes ilustrados, además de otros componentes.

La **Figura 18** es un diagrama de bloques de un procesador de gráficos 1800, que puede ser una unidad de procesamiento gráfico discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento. En algunas formas de realización, el procesador gráfico se comunica por medio de una interfaz de E/S asignada en memoria con registros en el procesador gráfico y con órdenes colocados en la memoria del procesador. En algunas formas de realización, el procesador gráfico 1800 incluye una interfaz de memoria 1814 para acceder a la memoria. La interfaz de memoria 1814 puede ser una interfaz a la memoria local, una o más memorias caché internas, una o más memorias caché externas compartidas, y/o a la memoria del sistema.

En algunas formas de realización, el procesador gráfico 1800 también incluye un controlador de visualización 1802 para conducir los datos de salida de pantalla a un dispositivo de visualización 1820. El controlador de visualización 1802 incluye hardware para uno o más planos de superposición para la visualización y composición de múltiples capas

de vídeo o elementos de interfaz de usuario. En algunas formas de realización, el procesador gráfico 1800 incluye un motor de códec de vídeo 1806 para codificar, decodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluyendo, entre otros, formatos del Grupo de Expertos en Imágenes en Movimiento (Moving Picture Experts Group, MPEG) como MPEG-2, formatos de Codificación de Vídeo Avanzada (Advanced Video Coding, AVC) como H.264/MPEG-4 AVC, de este modo como los formatos 421M/VC-1 de la Sociedad de Ingenieros de Cine y Televisión (Society of Motion Picture & Television Engineers, SMPTE), y los formatos del Grupo Conjunto de Expertos en Fotografía (Joint Photographic Experts Group, JPEG), como JPEG y Motion JPEG (MJPEG).

En algunas formas de realización, el procesador gráfico 1800 incluye un motor de transferencia de imagen de bloque (BLIT) 1804 para llevar a cabo operaciones de rasterización bidimensionales (2D) incluyendo, por ejemplo, transferencias de bloques de bandas de bits. Sin embargo, en una forma de realización, las operaciones de gráficos 2D se llevan a cabo utilizando uno o más componentes del motor de procesamiento gráfico (GPE) 1810. En algunas formas de realización, GPE 1810 es un motor de cálculo para llevar a cabo operaciones gráficas, incluyendo operaciones gráficas tridimensionales (3D) y operaciones de medios.

En algunas formas de realización, el GPE 1810 incluye un canal 3D 1812 para llevar a cabo operaciones 3D, como la renderización de imágenes y escenas tridimensionales utilizando funciones de procesamiento que actúan sobre formas primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). El canal 3D 1812 incluye elementos de función programables y fijos que llevan a cabo varias tareas dentro del elemento y/o generan subprocesos de ejecución a un subsistema de 3D/medios 1815. Mientras que el canal 3D 1812 se puede utilizar para llevar a cabo operaciones de medios, una forma de realización de GPE 1810 también incluye un canal de medios 1816 que se utiliza específicamente para llevar a cabo operaciones de medios, como el postprocesamiento de vídeo y la mejora de imágenes.

En algunas formas de realización, el canal de medios 1816 incluye unidades lógicas programables o de función fija para llevar a cabo una o más operaciones de medios especializadas, como aceleración de decodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar de, o en nombre del motor de códec de vídeo 1806. En algunas formas de realización, el canal de medios 1816 incluye adicionalmente una unidad de generación de subprocesos para generar subprocesos para su ejecución en el subsistema de 3D/medios 1815. Los subprocesos generados llevan a cabo cálculos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema de 3D/Medios 1815.

En algunas formas de realización, el subsistema de 3D/medios 1815 incluye lógica para ejecutar subprocesos generados por el canal 3D 1812 y el canal de medios 1816. En una forma de realización, las canalizaciones envían solicitudes de ejecución de subprocesos al subsistema de 3D/Medios 1815, que incluye lógica de envío de subprocesos para arbitrar y enviar las diversas solicitudes a los recursos de ejecución de subprocesos disponibles. Los recursos de ejecución incluyen un conjunto de unidades de ejecución gráfica para procesar los subprocesos 3D y multimedia. En algunas formas de realización, el subsistema de 3D/Medios 1815 incluye una o más memorias caché internas para instrucciones y datos de subprocesos. En algunas formas de realización, el subsistema también incluye memoria compartida, incluyendo registros y memoria direccionable, para compartir datos entre subprocesos y para almacenar datos de salida.

Motor de procesamiento gráfico

La **Figura 19** es un diagrama de bloques de un motor de procesamiento gráfico 1910 de un procesador de gráficos de acuerdo con algunas formas de realización. En una forma de realización, el motor de procesamiento gráfico (GPE) 1910 es una versión del GPE 1810 mostrado en la **Figura 18**. Los elementos de la **Figura 19** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura de la presente memoria pueden operar o funcionar de cualquier manera similar a la descrita en otras partes del presente documento, pero no se limitan a los mismas. Por ejemplo, se ilustran el canal 3D 1812 y el canal de medios 1816 de la **Figura 18**. La canal de medios 1816 es opcional en algunas formas de realización del GPE 1910 y puede no estar incluida explícitamente dentro del GPE 1910. Por ejemplo, en al menos una forma de realización, se acopla al GPE 1910 un procesador de medios y/o imágenes separado.

En algunas formas de realización, el GPE 1910 se acopla con o incluye un transmisor de órdenes 1903, que proporciona un flujo de órdenes al canal 3D 1812 y/o a los canales de medios 1816. En algunas formas de realización, el transmisor de órdenes 1903 se acopla con memoria, que puede ser memoria de sistema, o una o más de memoria caché interna y memoria caché compartida. En algunas formas de realización, el transmisor de órdenes 1903 recibe órdenes de la memoria y envía las órdenes al canal 3D 1812 y/o al canal de medios 1816. Las órdenes son directivas obtenidas de una memoria intermedia en anillo, que almacena órdenes para el canal 3D 1812 y el canal de medios 1816. En una forma de realización, la memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de órdenes por lotes que almacenan lotes de múltiples órdenes. Las órdenes para el canal 3D 1812 también pueden incluir referencias a datos almacenados en memoria como, por ejemplo, entre otros, datos de vértices y geometría para el canal 3D 1812 y/o datos de imagen y objetos de memoria para el canal de medios 1816. La canal 3D 1812 y el canal de medios 1816 procesan las órdenes y datos llevando a cabo operaciones a través de la lógica dentro de las canalizaciones respectivas o enviando uno o más subprocesos de ejecución a un conjunto de núcleos gráficos 1914.

En varias formas de realización, el canal 3D 1812 puede ejecutar uno o más programas de sombreado, 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 enviando subprocesos de ejecución al conjunto de núcleos gráficos 1914. El conjunto de núcleos gráficos 1914 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución multipropósito (por ejemplo, unidades de ejecución) dentro del conjunto de núcleos gráficos 1914 incluye soporte para varios lenguajes de sombreado 3D API y puede ejecutar múltiples subprocesos de ejecución simultáneos asociados con múltiples sombreadores.

En algunas formas de realización, el conjunto de núcleos gráficos 1914 también incluye lógica de ejecución para llevar a cabo funciones de medios, como procesamiento de vídeo y/o imágenes. En una forma de realización, las unidades de ejecución incluyen además lógica de propósito general que es programable para llevar a cabo operaciones de cálculo paralelas de propósito general, además de operaciones de procesamiento gráfico. La lógica de propósito general puede llevar a cabo operaciones de procesamiento paralelo o en conjunción con la lógica de propósito general dentro del núcleo(s) de procesador 1607 de la **Figura 16** o el núcleo 1702A-1702N como en la Figura **Figura 17**.

Los datos de salida generados por los subprocesos que se ejecutan en el conjunto de núcleos gráficos 1914 pueden enviar datos a la memoria en una memoria intermedia de retorno unificado (URB) 1918. El URB 1918 puede almacenar datos para múltiples subprocesos. En algunas formas de realización, el URB 1918 se puede utilizar para enviar datos entre diferentes subprocesos que se ejecutan en el conjunto de núcleos gráficos 1914. En algunas formas de realización, la URB 1918 se puede utilizar adicionalmente para la sincronización entre subprocesos en el conjunto de núcleos gráficos y la lógica de función fija dentro de la lógica de función compartida 1920.

En algunas formas de realización, el conjunto de núcleos gráficos 1914 se puede escalar, de tal forma que el conjunto incluye un número variable de núcleos gráficos, cada uno de los cuales tiene un número variable de unidades de ejecución basado en la potencia objetivo y el nivel de rendimiento del GPE 1910. En una forma de realización, los recursos de ejecución se pueden escalar dinámicamente, de tal forma que los recursos de ejecución se pueden activar o desactivar según sea necesario.

El conjunto de núcleos gráficos 1914 se acopla con la lógica de funciones compartidas 1920 que incluye múltiples recursos que son compartidos entre los núcleos gráficos en el conjunto de núcleos gráficos. Las funciones compartidas dentro de la lógica de funciones compartidas 1920 son unidades de lógica de hardware que proporcionan una funcionalidad complementaria especializada al conjunto de núcleos gráficos 1914. En diversas formas de realización, la lógica de función compartida 1920 incluye, entre otros, el muestreador 1921, los cálculos 1922 y la lógica de comunicación entre subprocesos (ITC) 1923. Además, algunas formas de realización implementan una o más memoria(s) caché 1925 dentro de la lógica de función compartida 1920. Una función compartida se implementa cuando la solicitud de una función especializada dada es insuficiente para su inclusión en el conjunto de núcleos gráficos 1914. En su lugar, se implementa una única instanciación de dicha función especializada como una entidad autónoma en la lógica de función compartida 1920 y se comparte entre los recursos de ejecución dentro del conjunto de núcleos gráficos 1914. El conjunto preciso de funciones que se comparten entre el conjunto de núcleos gráficos 1914 y se incluyen dentro del conjunto de núcleos gráficos 1914 varía entre las distintas formas de realización.

La **Figura 20** es un diagrama de bloques de otra forma de realización de un procesador gráfico 2000. Los elementos de la **Figura 20** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura de la presente memoria pueden operar o funcionar de cualquier manera similar a la descrita en otras partes del presente documento, pero no se limitan a los mismas.

En algunas formas de realización, el procesador gráfico 2000 incluye una interconexión en anillo 2002, un front-end del canal 2004, un motor de medios 2037 y núcleos gráficos 2080A-2080N. En algunas formas de realización, la interconexión en anillo 2002 acopla el procesador gráfico a otras unidades de procesamiento, incluyendo otros procesadores gráficos o uno o más núcleos de procesador de propósito general. En algunas formas de realización, el procesador gráfico es uno de los muchos procesadores integrados en un sistema de procesamiento de múltiples núcleos.

En algunas formas de realización, el procesador gráfico 2000 recibe lotes de órdenes a través de la interconexión en anillo 2002. Las órdenes entrantes se interpretan por un transmisor de órdenes 2003 en el front-end del canal 2004. En algunas formas de realización, el procesador gráfico 2000 incluye lógica de ejecución escalable para llevar a cabo el procesamiento de geometría 3D y el procesamiento de medios a través de los núcleos gráficos 2080A-2080N. Para las órdenes de procesamiento de geometría 3D, el transmisor de órdenes 2003 suministra órdenes al canal de geometría 2036. Para al menos algunas órdenes de procesamiento de medios, el transmisor de órdenes 2003 suministra las órdenes a un front-end de vídeo 2034, que se acopla con un motor de medios 2037. En algunas formas de realización, el motor de medios 2037 incluye un motor de calidad de vídeo (VQE) 2030 para el postprocesamiento de vídeo e imágenes y un motor de codificación/decodificación multiformato (MFX) 2033 para proporcionar codificación y decodificación de datos de medios acelerada por hardware. En algunas formas de realización, el canal de geometría 2036 y el motor de medios 2037 generan cada uno subprocesos de ejecución para los recursos de ejecución de subprocesos proporcionados por al menos un núcleo de gráficos 2080A.

En algunas formas de realización, el procesador gráfico 2000 incluye recursos de ejecución de subprocesos escalables que presentan núcleos modulares 2080A-2080N (denominados, en ocasiones, rebanadas de núcleo), cada una de las cuales tiene múltiples subnúcleos 2050A-550N, 2060A-2060N (denominados, en ocasiones, subrebanadas de núcleo). En algunas formas de realización, el procesador de gráficos 2000 puede tener cualquier número de núcleos gráficos 2080A a 2080N. En algunas formas de realización, el procesador de gráficos 2000 incluye un núcleo de gráficos 2080A que tiene al menos un primer subnúcleo 2050A y un segundo subnúcleo 2060A. En otras formas de realización, el procesador de gráficos es un procesador de baja potencia con un único subnúcleo (por ejemplo, 2050A). En algunas formas de realización, el procesador de gráficos 2000 incluye múltiples núcleos gráficos 2080A-2080N, incluyendo cada uno un conjunto de primeros subnúcleos 2050A-2050N y un conjunto de segundos subnúcleos 2060A-2060N. Cada subnúcleo del conjunto de primeros subnúcleos 2050A-2050N incluye al menos un primer conjunto de unidades de ejecución 2052A-2052N y muestreadores de medios/texturas 2054A-2054N. Cada subnúcleo del conjunto de segundos subnúcleos 2060A-2060N incluye al menos un segundo conjunto de unidades de ejecución 2062A-2062N y muestreadores 2064A-2064N. En algunas formas de realización, cada subnúcleo 2050A-2050N, 2060A-2060N comparte un conjunto de recursos compartidos 2070A-2070N. En algunas formas de realización, los recursos compartidos incluyen memoria caché compartida y lógica de operación de píxeles. También pueden incluirse otros recursos compartidos en las diversas formas de realización del procesador gráfico.

Unidades de ejecución

La **Figura 21** ilustra la lógica de ejecución de subprocesos 2100, que incluye un conjunto de elementos de procesamiento empleados en algunas formas de realización de un GPE. Los elementos de la **Figura 21** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura de la presente memoria pueden operar o funcionar de cualquier manera similar a la descrita en otras partes del presente documento, pero no se limitan a los mismas.

En algunas formas de realización, la lógica de ejecución de subprocesos 2100 incluye un procesador de sombreado 2102, un distribuidor de subprocesos 2104, una memoria caché de instrucciones 2106, un conjunto de unidades de ejecución que se puede escalar que incluye varias unidades de ejecución 2108A-2108N, un muestreador 2110, una memoria caché de datos 2112 y un puerto de datos 2114. En una forma de realización, el conjunto de unidades de ejecución que se puede escalar puede escalarse dinámicamente habilitando o deshabilitando una o más unidades de ejecución (por ejemplo, cualquiera de las unidades de ejecución 2108A, 2108B, 2108C, 2108D, hasta 2108N-1 y 2108N) basándose en las necesidades de cálculo de una carga de trabajo. En una forma de realización, los componentes incluidos están interconectados a través de un tejido de interconexión que enlaza a cada uno de los componentes. En algunas formas de realización, la lógica de ejecución de subprocesos 2100 incluye una o más conexiones a memoria, tal como memoria de sistema o memoria caché, a través de una o más de la memoria caché de instrucciones 2106, puerto de datos 2114, muestreador 2110, y unidades de ejecución 2108A-2108N. En algunas formas de realización, cada unidad de ejecución (por ejemplo 2108A) es una unidad de cálculo de propósito general programable autónoma que es capaz de ejecutar múltiples subprocesos de hardware simultáneos mientras procesa múltiples elementos de datos en paralelo para cada subproceso. En varias formas de realización, el conjunto de unidades de ejecución 2108A-2108N se puede escalar para incluir cualquier número de unidades de ejecución individuales.

En algunas formas de realización, las unidades de ejecución 2108A-2108N se usan principalmente para ejecutar programas de sombreador. Un procesador de sombreado 2102 puede procesar los diversos programas de sombreado y enviar subprocesos de ejecución asociados con los programas de sombreado a través de un distribuidor de subprocesos 2104. En una forma de realización, el distribuidor de subprocesos incluye lógica para arbitrar solicitudes de inicio de subproceso a partir de los canales de gráficos y de medios e instanciar los subprocesos solicitados en una o más unidades de ejecución de las unidades de ejecución 2108A-2108N. Por ejemplo, el canal de geometría (por ejemplo, 2036 de la **Figura 20)** puede enviar sombreadores de vértices, teselado o geometría a la lógica de ejecución de subprocesos 2100 **(Figura 21)** para procesar. En algunas formas de realización, el distribuidor de subprocesos 2104 también puede procesar solicitudes de generación de subprocesos en tiempo de ejecución desde los programas de sombreado en ejecución.

En algunas formas de realización, las unidades de ejecución 2108A-2108N soportan un conjunto de instrucciones que incluye un soporte nativo para muchas instrucciones de sombreador de gráficos 3D convencionales, de tal forma que los programas de sombreado desde bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución soportan procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y procesamiento de propósito general (por ejemplo, sombreadores de cálculo y de medios). Cada una de las unidades de ejecución 2108A-2108N es capaz de múltiples emisiones de ejecución de instrucción única, datos múltiples (SIMD) y la operación multisubproceso permite un entorno de ejecución eficiente frente a accesos de memoria de latencia más alta. Cada subproceso de hardware dentro de cada unidad de ejecución tiene un archivo de registro dedicado de gran ancho de banda y un estado de subproceso independiente asociado. La ejecución es de múltiples emisiones por reloj a canales capaces de llevar a cabo operaciones con enteros, operaciones de coma flotante de precisión simple y doble, capacidad de bifurcación SIMD,

operaciones lógicas, operaciones trascendentales y otras operaciones diversas. Mientras espera datos de la memoria o de una de las funciones compartidas, la lógica de dependencia de las unidades de ejecución 2108A-2108N hace que un subproceso en espera quede en suspensión hasta que se devuelvan los datos solicitados. Mientras el subproceso en espera está en suspensión, los recursos de hardware pueden dedicarse a procesar otros subprocesos. Por ejemplo, durante un retardo asociado con una operación de sombreado de vértices, una unidad de ejecución puede llevar a cabo operaciones para un sombreado de píxeles, sombreado de fragmentos u otro tipo de programa de sombreado, incluyendo un sombreado de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 2108A-2108N opera sobre conjuntos de elementos de datos. El número de elementos de datos es el "tamaño de ejecución", o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso a elementos de datos, enmascaramiento y control de flujo dentro de las instrucciones. El número de canales puede ser independiente del número de unidades aritméticas lógicas (ALUs) o unidades de coma flotante (FPUs) de un procesador gráfico en particular. En algunas formas de realización, las unidades de ejecución 2108A-2108N soportan tipos de datos de números enteros y de coma flotante.

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

Una o mas memorias caché de instrucciones internas (por ejemplo, 2106) se incluyen en la lógica de ejecución de subprocesos 2100 para almacenar en memoria caché instrucciones de subprocesos para las unidades de ejecución. En algunas formas de realización, se incluyen una o más memorias caché de datos (por ejemplo, 2112) para almacenar en memoria caché los datos del subproceso durante la ejecución del subproceso. En algunas formas de realización, se incluye un muestreador 2110 para proporcionar muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunas formas de realización, el muestreador 2110 incluye funcionalidad especializada de muestreo de textura o de medios para procesar datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las canales de gráficos y medios envían solicitudes de iniciación de subprocesos a la lógica de ejecución de subprocesos 2100 a través de la lógica de generación y envío de subprocesos. Una vez que un grupo de objetos geométricos ha sido procesado y rasterizado en datos de píxeles, la lógica del procesador de píxeles (por ejemplo, lógica de sombreado de píxeles, lógica de sombreado de fragmentos, etc.) dentro del procesador de sombreado 2102 es invocada para seguir calculando la información de salida y hacer que los resultados se escriban en las superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de plantilla, etc.). En algunas formas de realización, un sombreador de píxeles o un sombreador de fragmentos calcula los valores de los diversos atributos de vértice que deben interpolarse a través del objeto rasterizado. En algunas formas de realización, la lógica del procesador de píxeles dentro del procesador de sombreado 2102 ejecuta a continuación un programa de sombreado de píxeles o fragmentos suministrado por una interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreado, el procesador de sombreado 2102 envía subprocesos a una unidad de ejecución (por ejemplo, 2108A) a través del distribuidor de subprocesos 2104. En algunas formas de realización, el sombreador de píxeles 2102 utiliza lógica de muestreo de textura en el muestreador 2110 para acceder a datos de textura en mapas de textura almacenados en memoria. Las operaciones aritméticas en los datos de textura y los datos de geometría de entrada calculan los datos de color de píxel para cada fragmento geométrico, o descartan uno o más píxeles del procesamiento posterior.

En algunas formas de realización, el puerto de datos 2114 proporciona un mecanismo de acceso a memoria para la salida de datos procesados de la lógica de ejecución de subprocesos 2100 a la memoria para su procesamiento en un canal de salida del procesador gráfico. En algunas formas de realización, el puerto de datos 2114 incluye o se acopla a una o más memorias caché (por ejemplo, memoria caché de datos 2112) para almacenar en memoria caché los datos para el acceso a memoria a través del puerto de datos.

La **Figura 22** es un diagrama de bloques que ilustra los formatos de instrucción 2200 del procesador de gráficos de acuerdo con algunas formas de realización. En una o más formas de realización, las unidades de ejecución del procesador de gráficos admiten un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los recuadros de líneas continuas ilustran los componentes que se incluyen generalmente en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que sólo se incluyen en un subconjunto de las instrucciones. En algunas formas de realización, el formato de instrucción 2200 descrito e ilustrado son macroinstrucciones, en el sentido de que son instrucciones suministradas a la unidad de ejecución, en oposición a microoperaciones resultantes de la decodificación de instrucciones una vez que se procesa la instrucción.

En algunas formas de realización, las unidades de ejecución del procesador de gráficos admiten de forma nativa instrucciones en un formato de instrucción de 128 bits 2210. Un formato de instrucción compactado de 64 bits 2230 está disponible para algunas instrucciones basadas en la instrucción seleccionada, las opciones de instrucción y el número de operandos. El formato de instrucción de 128 bits nativo 710 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de 64 bits 2230. Las instrucciones nativas disponibles en el formato de 64 bits 2230 varían según la forma de realización. En algunas formas de realización, la instrucción se compacta en parte utilizando un conjunto de valores de índice en un campo de índice 2213. El hardware de la unidad de ejecución hace referencia a un conjunto de tablas de compactación basadas en los valores de índice y utiliza las salidas de la tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 2210.

Para cada formato, la instrucción opcode 2212 define la operación que debe llevar a cabo la unidad de ejecución. Las unidades de ejecución ejecutan cada instrucción en paralelo a través de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de suma, la unidad de ejecución lleva a cabo una operación de suma simultánea a través de cada canal de color que representa un elemento de textura o un elemento de imagen. Por defecto, la unidad de ejecución lleva a cabo cada instrucción a través de todos los canales de datos de los operandos. En algunas formas de realización, el campo de control de instrucción 2214 permite controlar ciertas opciones de ejecución, como la selección de canales (por ejemplo, predicación) y el orden de los canales de datos (por ejemplo, mezcla). Para instrucciones en el formato de instrucción de 128 bits 2210 un campo exec-size 2216 limita el número de canales de datos que serán ejecutados en paralelo. En algunas formas de realización, el campo exec-size 2216 no está disponible para su utilización en el formato de instrucción compacto de 64 bits 2230.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos, incluyendo dos operandos fuente, src0 2220, src1 2222, y un destino 2218. En algunas formas de realización, las unidades de ejecución soportan instrucciones de doble destino, donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando fuente (por ejemplo, SRC2 2224), donde el opcode 2212 de la instrucción determina el número de operandos fuente. El último operando fuente de una instrucción puede ser un valor inmediato (por ejemplo, codificado) pasado con la instrucción.

En algunas formas de realización, el formato de instrucción de 128 bits 2210 incluye un campo de modo de acceso/dirección 2226 que especifica, por ejemplo, si se utiliza el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos se proporciona directamente mediante bits en la instrucción.

En algunas formas de realización, el formato de instrucción de 128 bits 2210 incluye un campo de modo de acceso/dirección 2226, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En una forma de realización, el modo de acceso se utiliza para definir una alineación de acceso a datos para la instrucción. Algunas formas de realización soportan modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, donde la alineación de bytes del modo de acceso determina la alineación de acceso de los operandos de la instrucción. Por ejemplo, cuando está en un primer modo, la instrucción puede utilizar direccionamiento alineado con bytes para los operandos fuente y destino y cuando está en un segundo modo, la instrucción puede utilizar direccionamiento alineado con 16 bytes para todos los operandos fuente y destino.

En una forma de realización, la parte de modo de dirección del campo de modo de acceso/dirección 2226 determina si la instrucción va a utilizar direccionamiento directo o indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, los bits de la instrucción proporcionan directamente la dirección de registro de uno o más operandos. Cuando se utiliza el modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos puede calcularse basándose en un valor de registro de dirección y un campo de dirección inmediata en la instrucción.

En algunas formas de realización, las instrucciones se agrupan basándose en los campos de bits del opcode 2212 para simplificar la decodificación del opcode 2240. Para un opcode de 8 bits, los bits 4, 5 y 6 permiten a la unidad de ejecución determinar el tipo de opcode. El clúster preciso de opcode mostrado es meramente un ejemplo. En algunas formas de realización, un grupo de opcode de movimiento y lógica 2242 incluye instrucciones de movimiento de datos y lógica (por ejemplo, mover (mov), comparar (cmp)). En algunas formas de realización, el grupo de movimiento y lógica 2242 comparte los cinco bits más significativos (MSB), donde las instrucciones de movimiento (mov) están en la forma de 0000xxxxb y las instrucciones de lógica están en la forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 2244 (por ejemplo, call, jump (jmp)) incluye instrucciones en la forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 2246 incluye una mezcla de instrucciones, incluyendo instrucciones de sincronización (por ejemplo, wait, send) en la forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo paralelas 2248 incluye instrucciones aritméticas por componentes (por ejemplo, sumar, multiplicar (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculo paralelo 2248 lleva a cabo las operaciones aritméticas en paralelo a través de los canales de datos. El grupo de cálculo vectorial 2250 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cálculo vectorial lleva a cabo operaciones aritméticas tales como cálculos de producto punto en operandos vectoriales.

Canal de gráficos

La **Figura 23** es un diagrama de bloques de otra forma de realización de un procesador gráfico 2300. Los elementos de la **Figura 23** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura de la presente memoria pueden operar o funcionar de cualquier manera similar a la descrita en otras partes del presente documento, pero no se limitan a los mismas.

En algunas formas de realización, el procesador gráfico 2300 incluye un canal de gráficos 2320, un canal de medios 2330, un motor de visualización 2340, una lógica de ejecución de subprocesos 2350 y un canal de salida de renderizado 2370. En algunas formas de realización, el procesador gráfico 2300 es un procesador gráfico dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador gráfico se controla mediante escrituras de registro en uno o más registros de control (no mostrados) o mediante órdenes emitidos al procesador gráfico 2300 a través de una interconexión en anillo 2302. En algunas formas de realización, la interconexión en anillo 2302 acopla el procesador gráfico 2300 a otros componentes de procesamiento, tales como otros procesadores gráficos o procesadores de propósito general. Las órdenes de la interconexión en anillo 2302 son interpretados por un transmisor de órdenes 2303, que suministra instrucciones a componentes individuales del canal de gráficos 2320 o canal de medios 2330.

En algunas formas de realización, el transmisor de órdenes 2303 dirige la operación de un recuperador de vértices 2305 que lee datos de vértices de la memoria y ejecuta órdenes de procesamiento de vértices proporcionados por el transmisor de órdenes 2303. En algunas formas de realización, el recuperador de vértices 2305 proporciona datos de vértices a un sombreador de vértices 2307, que lleva a cabo la transformación del espacio de coordenadas y las operaciones de iluminación de cada vértice. En algunas formas de realización, el recuperador de vértices 2305 y el sombreador de vértices 2307 ejecutan instrucciones de procesamiento de vértices enviando subprocesos de ejecución a las unidades de ejecución 2352A-2352B a través de un distribuidor de subprocesos 2331.

En algunas formas de realización, las unidades de ejecución 2352A-2352B son un conjunto de procesadores de vectores que tienen un conjunto de instrucciones para llevar a cabo operaciones de gráficos y de medios. En algunas formas de realización, las unidades de ejecución 2352A-2352B tienen una memoria caché L1 acoplada 2351 que es específica para cada matriz o que se comparte entre los conjuntos. La memoria caché se puede configurar como una memoria caché de datos, una memoria caché de instrucciones, o una única memoria caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.

En algunas formas de realización, el canal de gráficos 2320 incluye componentes de teselado para llevar a cabo el teselado acelerado por hardware de objetos 3D. En algunas formas de realización, un sombreador de casco programable 811 configura las operaciones de teselado. Un sombreador de dominio programable 817 proporciona una evaluación back-end del resultado de la teselado. Un teselador 2313 opera en la dirección del sombreador de casco 2311 y contiene lógica de propósito especial para generar un conjunto de objetos geométricos detallados basados en un modelo geométrico grueso que se proporciona como entrada al canal de gráficos 2320. En algunas formas de realización, si no se utiliza el teselado, los componentes de teselado (por ejemplo, el sombreador de casco 2311, el teselador 2313 y el sombreador de dominio 2317) se pueden omitir.

En algunas formas de realización, unos objetos geométricos completos pueden ser procesados por un sombreador de geometría 2319 mediante uno o más subprocesos enviados a las unidades de ejecución 2352A-2352B, o pueden dirigirse directamente a un recortador 2329. En algunas formas de realización, el sombreador de geometría opera sobre objetos geométricos enteros, en lugar de vértices o parches de vértices como en fases anteriores del canal de gráficos. Si la teselado está desactivada, el sombreador de geometría 2319 recibe la entrada del sombreador de vértices 2307. En algunas formas de realización, el sombreador de geometría 2319 es programable por un programa de sombreado de geometría para llevar a cabo el teselado de geometría si las unidades de teselado están deshabilitadas.

Antes de la rasterización, un recortador 2329 procesa los datos de vértices. El recortador 2329 puede ser un recortador de función fija o un recortador programable con funciones de recorte y sombreado geométrico. En algunas formas de realización, un rasterizador y componente de prueba de profundidad 2373 en el canal de salida de renderizado 2370 envía sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunas formas de realización, la lógica de sombreado de píxeles se incluye en la lógica de ejecución de subprocesos 2350. En algunas formas de realización, una aplicación puede omitir el rasterizador y el componente de prueba de profundidad 2373 y acceder a los datos de vértices no rasterizados por medio de una unidad de salida de secuencias 2323.

El procesador gráfico 2300 tiene un bus de interconexión, un tejido de interconexión o algún otro mecanismo de interconexión que permite el paso de datos y mensajes entre los principales componentes del procesador. En algunas formas de realización, las unidades de ejecución 2352A-2352B y la memoria caché o memorias caché asociadas 2351, el muestreador de textura y de medios 2354 y la memoria caché de textura/muestreador 2358 se interconectan mediante un puerto de datos 2356 para llevar a cabo el acceso a memoria y comunicarse con los componentes del canal de salida de renderizado del procesador. En algunas formas de realización, el muestreador 2354, las memorias caché 2351, 2358 y las unidades de ejecución 2352A-2352B tienen trayectorias de acceso a memoria separadas.

En algunas formas de realización, el canal de salida de renderizado 2370 contiene un componente rasterizador y de prueba de profundidad 2373 que convierte objetos basados en vértices en una representación asociada basada en píxeles. En algunas formas de realización, la lógica del rasterizador incluye una unidad de ventana/máscara para llevar a cabo la rasterización de triángulos y líneas de función fija. Una memoria caché de renderizado asociada 2378 y una memoria caché de profundidad 2379 también están disponibles en algunas formas de realización. Un componente de operaciones de píxeles 2377 lleva a cabo operaciones basadas en píxeles sobre los datos, aunque en algunas instancias, las operaciones de píxeles asociadas con operaciones 2D (por ejemplo, transferencias de imágenes de bloques de bits con mezcla) son realizadas por el motor 2D 2341, o sustituidas en tiempo de visualización por el controlador de visualización 2343 utilizando planos de visualización superpuestos. En algunas formas de realización, una memoria caché L3 compartida 2375 está disponible para todos los componentes gráficos, permitiendo compartir datos sin la utilización de la memoria de sistema principal.

En algunas formas de realización, el canal de medios del procesador gráfico 2330 incluye un motor de medios 2337 y un front-end de vídeo 2334. En algunas formas de realización, el front-end de vídeo 2334 recibe órdenes de canal desde el transmisor de órdenes 2303. En algunas formas de realización, el canal de medios 2330 incluye un transmisor de órdenes separado. En algunas formas de realización, el front-end de vídeo 2334 procesa las órdenes de medios antes de enviar la orden al motor de medios 2337. En algunas formas de realización, el motor de medios 2337 incluye funcionalidad de generación de subprocesos para generar subprocesos para su envío a la lógica de ejecución de subprocesos 2350 a través del distribuidor de subprocesos 2331.

En algunas formas de realización, el procesador de gráficos 2300 incluye un motor de visualización 2340. En algunas formas de realización, el motor de visualización 2340 es externo al procesador 2300 y se acopla con el procesador gráfico a través de la interconexión en anillo 2302, o algún otro bus o tejido de interconexión. En algunas formas de realización, el motor de visualización 2340 incluye un motor 2D 2341 y un controlador de visualización 2343. En algunas formas de realización, el motor de visualización 2340 contiene lógica de propósito especial capaz de operar de forma independiente del canal 3D. En algunas formas de realización, el controlador de visualización 2343 se acopla con un dispositivo de pantalla (no mostrado), que puede ser un dispositivo de pantalla integrado en el sistema, como en un ordenador portátil, o un dispositivo de pantalla externo conectado a través de un conector de dispositivo de pantalla.

En algunas formas de realización, el canal de gráficos 2320 y el canal de medios 2330 son configurables para llevar a cabo operaciones basadas en múltiples interfaces de programación de gráficos y medios y no son específicos de ninguna interfaz de programación de aplicaciones (API). En algunas formas de realización, el software controlador del procesador gráfico traduce las llamadas a la API que son específicas de una determinada biblioteca de gráficos o de medios en órdenes que pueden ser procesados por el procesador gráfico. En algunas formas de realización, se proporciona soporte para la Open Graphics Library (OpenGL), Open Computing Language (OpenCL), y/o Vulkan graphics and compute API, todas del Khronos Group. En algunas formas de realización, también se puede proporcionar soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas formas de realización, puede ser compatible una combinación de estas bibliotecas. También puede ser compatible con la biblioteca Open Source Computer Vision Library (OpenCV). Una futura API con un canal 3D compatible también sería compatible si se puede hacer una asignación desde el canal de la futura API al canal del procesador gráfico.

Programación del canal de gráficos

La **Figura 24A** es un diagrama de bloques que ilustra un formato de orden de procesador de gráficos 2400 de acuerdo con algunas formas de realización. La **Figura 24B** es un diagrama de bloques que ilustra una secuencia de órdenes del procesador de gráficos 2410 de acuerdo con una forma de realización. Las cajas de líneas continuas en la **Figura 24A** ilustran los componentes que generalmente se incluyen en una orden de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que sólo se incluyen en un subconjunto de órdenes de gráficos. El formato de orden de procesador de gráficos 2400 de ejemplo de la **Figura 24A** incluye campos de datos para identificar un cliente objetivo 2402 de la orden, un código de operación de la orden (opcode) 2404, y los datos relevantes 2406 para la orden. Un sub-opcode 2405 y un tamaño de orden 2408 también se incluyen en algunas órdenes.

En algunas formas de realización, el cliente 2402 especifica la unidad cliente del dispositivo gráfico que procesa los datos de la orden. En algunas formas de realización, un analizador de órdenes del procesador gráfico examina el campo de cliente de cada comando para condicionar el procesamiento posterior de la orden y dirigir los datos de la orden a la unidad de cliente apropiada. En algunas formas de realización, las unidades cliente del procesador gráfico incluyen una unidad de interfaz de memoria, una unidad de renderizado, una unidad 2D, una unidad 3D y una unidad multimedia. Cada unidad cliente tiene un canal de procesamiento correspondiente que procesa las órdenes. Una vez que la orden es recibido por la unidad cliente, la unidad cliente lee el opcode 2404 y, si está presente, el sub-opcode 2405 para determinar la operación a llevar a cabo. La unidad cliente lleva a cabo la orden utilizando la información en el campo de datos 2406. Para algunas órdenes se espera un tamaño de comando explícito 2408 para especificar el tamaño de la orden. En algunas formas de realización, el analizador de órdenes determina de forma automática el

tamaño de al menos algunas de las órdenes basándose en el opcode de la orden. En algunas formas de realización las órdenes son alineados a través de múltiplos de una palabra doble.

El diagrama de flujo de la **Figura 24B** muestra una secuencia de órdenes del procesador de gráficos 2410 a modo de ejemplo. En algunas formas de realización, el software o firmware de un sistema de procesamiento de datos que caracteriza una forma de realización de un procesador gráfico utiliza una versión de la secuencia de órdenes mostrada para configurar, ejecutar y terminar un conjunto de operaciones gráficas. Se muestra y describe una secuencia de órdenes a modo de ejemplo, ya que las formas de realización no se limitan a estas órdenes específicas o a esta secuencia de órdenes. Además, las órdenes pueden ser emitidos como un lote de órdenes en una secuencia de órdenes, de tal forma que el procesador gráfico procesará la secuencia de órdenes en al menos parcialmente concurrencia.

En algunas formas de realización, la secuencia de órdenes del procesador de gráficos 2410 puede comenzar con una orden de descarga de canal 2412 para hacer que cualquier canal de gráficos activo complete las órdenes actualmente pendientes para el canal. En algunas formas de realización, el canal 3D 2422 y el canal de medios 2424 no operan simultáneamente. La descarga del canal se lleva a cabo para provocar que el canal de gráficos activo complete cualesquiera órdenes pendientes. En respuesta a una descarga del canal, el analizador de órdenes para el procesador gráfico pausará el procesamiento de órdenes hasta que los motores de dibujo activos completen las operaciones pendientes y las memorias caché de lectura relevantes sean invalidadas. Opcionalmente, cualquier dato en la memoria caché de renderizado que esté marcado como "sucio" puede ser descargado a memoria. En algunas formas de realización, la orden de descarga de canal 2412 se puede utilizar para la sincronización de canales o antes de colocar el procesador gráfico en un estado de bajo consumo.

En algunas formas de realización, una orden de selección de canal 2413 se utiliza cuando una secuencia de órdenes requiere que el procesador gráfico cambie explícitamente entre canales. En algunas formas de realización, una orden de selección de canal 2413 se requiere sólo una vez dentro de un contexto de ejecución antes de emitir órdenes de canal a menos que el contexto sea para emitir órdenes para ambas tuberías. En algunas formas de realización, se requiere una orden de descarga de canal 2412 inmediatamente antes de un cambio de canal a través de la orden de selección de canal 2413.

En algunas formas de realización, una orden de control de canal 2414 configura un canal de gráficos para su operación y se utiliza para programar el canal 3D 2422 y el canal de medios 2424. En algunas formas de realización, la orden de control de canal 2414 configura el estado de canal para el canal activa. En una forma de realización, la orden de control de canal 2414 se utiliza para la sincronización de canal y para borrar datos de una o más memorias caché dentro del canal activa antes de procesar un lote de órdenes.

En algunas formas de realización, las órdenes de estado de la memoria intermedia de retorno 2416 se utilizan para configurar un conjunto de memorias intermedias de retorno para que las respectivas canalizaciones escriban datos. Algunas operaciones de canal requieren la asignación, selección o configuración de una o varias memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas formas de realización, el procesador gráfico también utiliza uno o más memorias intermedias de retorno para almacenar datos de salida y para llevar a cabo la comunicación entre subprocesos. En algunas formas de realización, el estado de la memoria intermedia de retorno 2416 incluye la selección del tamaño y número de memorias intermedias de retorno a utilizar para un conjunto de operaciones de canal.

Las órdenes restantes de la secuencia de órdenes difieren en función del canal activa para las operaciones. Basándose en una determinación del canal 2420, la secuencia de órdenes se adapta al canal 3D 2422 empezando por el estado de canal 3D 2430 o al canal de medios 2424 empezando por el estado de canal de medios 2440.

Las órdenes para configurar el estado del canal 3D 2430 incluyen órdenes de configuración de estado 3D para el estado de la memoria intermedia de vértices, el estado de los elementos de vértices, el estado de color constante, el estado de la memoria intermedia de profundidad y otras variables de estado que deben configurarse antes de que se procesen las órdenes primitivas 3D. Los valores de estas órdenes se determinan, al menos en parte, en función de la API 3D en uso. En algunas formas de realización, las órdenes de estado de canal 3D 2430 también son capaces de desactivar u omitir de forma selectiva ciertos elementos de canal si dichos elementos no se van a utilizar.

En algunas formas de realización, la orden 3D primitiva 2432 se utiliza para presentar primitivas 3D para ser procesadas por el canal 3D. Las órdenes y parámetros asociados que se pasan al procesador gráfico a través de la orden 3D primitiva 2432 se reenvían a la función de obtención de vértices del canal de gráficos. La función de obtención de vértices utiliza los datos de la orden 3D primitiva 2432 para generar estructuras de datos de vértices. Las estructuras de datos de vértices se almacenan en uno o más memorias intermedias de retorno. En algunas formas de realización, la orden 3D primitiva 2432 se utiliza para llevar a cabo operaciones de vértices en primitivas 3D a través de sombreadores de vértices. Para procesar los sombreadores de vértices, el canal 3D 2422 envía subprocesos de ejecución de sombreadores a las unidades de ejecución del procesador gráfico.

En algunas formas de realización, el canal 3D 2422 se activa a través de una orden o evento de ejecución 2434. En algunas formas de realización, un registro de escritura activa la ejecución de la orden. En algunas formas de realización, la ejecución se activa a través de una orden "go" o "kick" en la secuencia de órdenes. En una forma de realización, la ejecución de órdenes se activa utilizando una orden de sincronización de canal para descargar la secuencia de órdenes a través del canal de gráficos. La canal 3D llevará a cabo el procesamiento geométrico de las primitivas 3D. Una vez completadas las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles colorea los píxeles resultantes. También pueden incluirse órdenes adicionales para controlar el sombreado de píxeles y las operaciones de back-end de píxeles para esas operaciones.

En algunas formas de realización, la secuencia de órdenes del procesador de gráficos 2410 sigue la trayectoria del canal de medios 2424 cuando se llevan a cabo operaciones de medios. En general, la utilización específica y la manera de programar el canal de medios 2424 depende de los medios o de las operaciones de cálculo que se vayan a llevar a cabo. Operaciones específicas de decodificación de medios pueden ser descargadas al canal de medios durante la decodificación de medios. En algunas formas de realización, el canal de medios también se puede omitir y la decodificación de medios se puede llevar a cabo en su totalidad o en parte utilizando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. En una forma de realización, el canal de medios también incluye elementos para operaciones de unidad de procesamiento gráfico de propósito general (GPGPU), donde el procesador gráfico se utiliza para llevar a cabo operaciones vectoriales SIMD utilizando programas de sombreado de cálculo que no están explícitamente relacionados con el renderizado de primitivas gráficas.

En algunas formas de realización, el canal de medios 2424 se configura de manera similar al canal 3D 2422. Un conjunto de órdenes para configurar el estado del canal de medios 2440 se envían o colocan en una cola de órdenes antes de las órdenes de objetos de medios 2442. En algunas formas de realización, las órdenes de estado del canal de medios 2440 incluyen datos para configurar los elementos del canal de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación y codificación de vídeo dentro del canal de medios, como el formato de codificación o decodificación. En algunas formas de realización, las órdenes de estado del canal de medios 2440 también soportan la utilización de uno o más punteros a elementos de estado "indirectos" que contienen un lote de configuraciones de estado.

En algunas formas de realización, las órdenes de objetos de medios 2442 suministran punteros a objetos de medios para ser procesados por el canal de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo para ser procesados. En algunas formas de realización, todos los estados del canal de medios deben ser válidos antes de emitir una orden de objeto de medios 2442. Una vez que el estado del canal se configura y las órdenes de objetos de medios 2442 están en cola, el canal de medios 2424 se activa por medio de una orden de ejecución 2444 o un evento de ejecución equivalente (por ejemplo, escritura de registro). Entonces, la salida del canal de medios 2424 puede ser post-procesada por operaciones proporcionadas por el canal 3D 2422 o el canal de medios 2424. En algunas formas de realización, las operaciones GPGPU se configuran y ejecutan de manera similar a las operaciones de medios.

Arquitectura de software de gráficos

La **Figura 25** ilustra una arquitectura de software de gráficos de ejemplo para un sistema de procesamiento de datos 2500 según algunas formas de realización. En algunas formas de realización, la arquitectura de software incluye una aplicación de gráficos 3D 2510, un sistema operativo 2520 y al menos un procesador 2530. En algunas formas de realización, el procesador 2530 incluye un procesador de gráficos 2532 y uno o más núcleos de procesador de propósito general 2534. La aplicación de gráficos 2510 y el sistema operativo 2520 se ejecutan cada uno en la memoria de sistema 2550 del sistema de procesamiento de datos.

En algunas formas de realización, la aplicación de gráficos 3D 2510 contiene uno o más programas de sombreado que incluyen instrucciones de sombreado 2512. Las instrucciones de lenguaje de sombreado pueden estar en un lenguaje de sombreado de alto nivel, como el lenguaje de sombreado de alto nivel (HLSL) o el lenguaje de sombreado OpenGL (GLSL). La aplicación también incluye instrucciones ejecutables 2514 en un lenguaje de máquina adecuado para su ejecución por el núcleo de procesador de propósito general 2534. La aplicación también incluye objetos gráficos 2516 definidos por datos de vértices.

En algunas formas de realización, el sistema operativo 2520 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo patentado de tipo UNIX, o un sistema operativo de tipo UNIX de código abierto que utiliza una variante del núcleo Linux. El sistema operativo 2520 puede soportar una API gráfica 2522 como la API Direct3D, la API OpenGL o la API Vulkan. Cuando la API Direct3D está en uso, el sistema operativo 2520 utiliza un compilador de sombreadores front-end 2524 para compilar cualesquiera instrucciones de sombreador 2512 en HLSL a un lenguaje de sombreadores de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede llevar a cabo una precompilación del sombreador. En algunas formas de realización, los sombreadores de alto nivel son compilados en sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 2510. En algunas formas de realización, las instrucciones de sombreado 2512 se proporcionan en una forma intermedia, como una versión de la Representación Intermedia Portátil Estándar (SPIR) utilizada por la API Vulkan.

En algunas formas de realización, el controlador de gráficos de modo de usuario 2526 contiene un compilador de sombreado back-end 2527 para convertir las instrucciones de sombreado 2512 en una representación específica de hardware. Cuando la API OpenGL está en uso, las instrucciones de sombreado 2512 en el lenguaje de alto nivel GLSL se pasan al controlador de gráficos de modo de usuario 2526 para su compilación. En algunas formas de realización, el controlador de gráficos de modo de usuario 2526 utiliza funciones de modo de núcleo del sistema operativo 2528 para comunicarse con un controlador de gráficos de modo de núcleo 2529. En algunas formas de realización, el controlador de gráficos en modo kernel 2529 se comunica con el procesador de gráficos 2532 para enviar órdenes e instrucciones.

Implementaciones núcleo IP

Uno o más aspectos de al menos una forma de realización se pueden implementar mediante código representativo almacenado en un medio legible por máquina que representa y/o define la lógica dentro de un circuito integrado como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan la lógica dentro del procesador. Cuando son leídas por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para llevar a cabo las técnicas descritas en la presente memoria. Tales representaciones, conocidas como "núcleos IP", son unidades reutilizables de lógica para un circuito integrado que se pueden almacenar en un medio tangible legible por máquina como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware puede suministrarse a varios 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 tal forma que el circuito lleve a cabo las operaciones descritas en asociación con cualquiera de las formas de realización descritas en la presente memoria.

La Figura 26 es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo IP 2600 que se puede utilizar para fabricar un circuito integrado para llevar a cabo operaciones de acuerdo con una forma de realización. El sistema de desarrollo de núcleo IP 2600 se puede utilizar para generar diseños modulares reutilizables que pueden incorporarse a un diseño mayor o utilizarse para construir un circuito integrado completo (por ejemplo, un circuito integrado SOC). Una instalación de diseño 2630 puede generar una simulación de software 2610 de un diseño de núcleo IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 2610 se puede utilizar para diseñar, probar y verificar el comportamiento del núcleo IP utilizando un modelo de simulación 2612. El modelo de simulación 2612 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Un diseño de nivel de transferencia de registro (RTL) 2615 se puede crear o sintetizar a continuación a partir del modelo de simulación 2612. El diseño RTL 2615 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, incluyendo la lógica asociada llevada a cabo utilizando las señales digitales modeladas. Además de un diseño RTL 2615, también pueden crearse, diseñarse o sintetizarse diseños de nivel inferior a nivel lógico o a nivel de transistor. Por lo tanto, los detalles particulares del diseño inicial y la simulación pueden variar.

El diseño RTL 2615 o equivalente puede ser sintetizado por la instalación de diseño en un modelo de hardware 2620, que puede estar en un lenguaje de descripción de hardware (HDL), o alguna otra representación de datos de diseño físico. El HDL puede ser simulado o probado para verificar el diseño del núcleo IP. El diseño del núcleo IP se puede almacenar para su implantación en una instalación de fabricación de terceros 2665 utilizando la memoria no volátil 2640 (por ejemplo, disco duro, memoria flash, o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño del núcleo IP puede ser transmitido (por ejemplo, a través de Internet) a través de una conexión por cable 2650 o una conexión inalámbrica 2660. A continuación, la instalación de fabricación 2665 puede fabricar un circuito integrado basándose, al menos en parte, en el diseño del núcleo IP. El circuito integrado fabricado se puede configurar para llevar a cabo operaciones de acuerdo con al menos una forma de realización descrita en la presente memoria.

Circuito integrado en un microprocesador de sistema de ejemplo

Las **Figuras 27-29** ilustran circuitos integrados de ejemplo y procesadores de gráficos asociados que pueden fabricarse utilizando uno o más núcleos IP, de acuerdo con varias formas de realización descritas en la presente memoria. Además de lo que se ilustra, se pueden incluir otros circuitos y lógica, incluidos procesadores/núcleos de gráficos adicionales, controladores de interfaz periférica o núcleos de procesador de uso general.

La **Figura 27** es un diagrama de bloques que ilustra un circuito integrado en un microprocesador de sistema 2700 de ejemplo que se puede fabricar utilizando uno o más núcleos IP, de acuerdo con una forma de realización. El circuito integrado 2700 de ejemplo incluye uno o más procesadores de aplicación 2705 (por ejemplo, CPU), al menos un procesador de gráficos 2710 y puede incluir adicionalmente un procesador de imágenes 2715 y/o un procesador de vídeo 2720, cualquiera de los cuales puede ser un núcleo IP modular de los mismos o múltiples instalaciones de diseño diferentes. El circuito integrado 2700 incluye una lógica de bus o de periféricos que incluye un controlador de USB 2725, un controlador de UART 2730, un controlador de SPI/SDIO 2735 y un controlador de I^2S/I^2C 2740. Además, el circuito integrado puede incluir un dispositivo de visualización 2745 acoplado a uno o más de un controlador de interfaz multimedia de alta definición (HDMI) 2750 y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 2755. El almacenamiento puede ser proporcionado por un subsistema de memoria flash 2760 que incluye

memoria flash y un controlador de memoria flash. La interfaz de memoria puede proporcionarse a través de un controlador de memoria 2765 para acceder a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 2770.

La **Figura 28** es un diagrama de bloques que ilustra un procesador gráfico 2810 de ejemplo de un circuito integrado en un microprocesador de sistema que se puede fabricar utilizando uno o más núcleos IP, de acuerdo con una forma de realización. El procesador de gráficos 2810 puede ser una variante del procesador de gráficos 2710 de la **Figura 27.** El procesador de gráficos 2810 incluye un procesador de vértices 2805 y uno o más procesadores de fragmentos 2815A-2815N (por ejemplo, 2815A, 2815B, 2815C, 2815D a 2815N-1 y 2815N). El procesador de gráficos 2810 puede ejecutar diferentes programas de sombreado a través de lógica separada, de tal forma que el procesador de vértices 2805 está optimizado para ejecutar operaciones para programas de sombreado de vértices, mientras que uno o más procesador(es) de fragmentos 2815A-2815N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreado de fragmentos o píxeles. El procesador de vértices 2805 lleva a cabo la fase de procesamiento de vértices del canal de gráficos 3D y genera primitivas y datos de vértices. Los procesadores de fragmentos 2815A-2815N utilizan las primitivas y los datos de vértices generados por el procesador de vértices 2805 para producir un framebuffer que se muestra en un dispositivo de visualización. En una forma de realización, el/los procesador/es de fragmentos 2815A-2815N están optimizados para ejecutar programas de sombreado de fragmentos como los previstos en la API OpenGL, que pueden utilizarse para llevar a cabo operaciones similares a las de un programa de sombreado de píxeles como los previstos en la API Direct 3D.

El procesador gráfico 2810 incluye además una o más unidades de gestión de memoria (MMU) 2820A-2820B, memorias caché 2825A-2825B, e interconexión(es) de circuitos 2830A-2830B. Las una o más MMU 2820A-2820B proporcionan una asignación entre direcciones virtuales y físicas para el circuito integrado 2810, de este modo como para el procesador de vértices 2805 y/o el/los procesador(es) de fragmentos 2815A-2815N, que pueden hacer referencia a datos de vértices o de imágenes/texturas almacenados en memoria, además de a datos de vértices o imágenes/texturas almacenados en las una o más memorias caché 2825A-2825B. En una forma de realización, la una o más MMU 2825A-2825B puede(n) estar sincronizada(s) con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con uno o más procesador(es) de aplicación 2705, procesador de imagen 2715, y/o procesador de vídeo 2720 de la **Figura 27**, de tal forma que cada procesador 2705-2720 puede participar en un sistema de memoria virtual compartido o unificado. Las una o más interconexiones de circuito 2830A-2830B permiten que el procesador de gráficos 2810 interactúe con otros núcleos IP dentro del SoC, ya sea a través de un bus interno del SoC o a través de una conexión directa, de acuerdo con las formas de realización.

La **Figura 29** es un diagrama de bloques que ilustra un procesador de gráficos 2910 adicional de ejemplo de un circuito integrado en un microprocesador de sistema que se puede fabricar utilizando uno o más núcleos IP, de acuerdo con una forma de realización. El procesador de gráficos 2910 puede ser una variante del procesador de gráficos 2710 de la **Figura 27.** El procesador de gráficos 2910 incluye las una o más MMU 2820A-2820B, memorias caché 2825A-2825B e interconexiones de circuito 2830A-2830B del circuito integrado 2800 de la **Figura 28**.

El procesador de gráficos 2910 incluye uno o más núcleo o núcleos de sombreador 2915A-2915N (por ejemplo, 2915A, 2915B, 2915C, 2915D, 2915E, 2915F, a 2915N-1 y 2915N), que proporciona una arquitectura de núcleo de sombreado unificada en la que un solo núcleo o tipo o núcleo puede ejecutar todo tipo de código de sombreado programable, incluyendo el código de programa de sombreador para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. El número exacto de núcleos de sombreador presentes puede variar entre formas de realización e implementaciones. Además, el procesador gráfico 2910 incluye un gestor de tareas entre núcleos 2905, que actúa como un distribuidor de subprocesos para enviar subprocesos de ejecución a uno o más núcleos de sombreado 2915A-2915N y una unidad de teselado 2918 para acelerar las operaciones de teselado para el renderizado basado en teselado, en el que las operaciones de renderizado para una escena se subdividen en el espacio de imagen, por ejemplo para explotar la coherencia espacial local dentro de una escena o para optimizar la utilización de memorias caché internas.

Algunas formas de realización hacen referencia al Ejemplo 1 que incluye un aparato para facilitar el procesamiento de una matriz dispersa, que comprende varias unidades de procesamiento cada una de las cuales comprende uno o más elementos de procesamiento, incluyendo lógica para leer operandos, una unidad de multiplicación para multiplicar dos o más operandos y un planificador para identificar operandos que tienen un valor cero y evitar la programación de los operandos que tienen el valor cero en la unidad de multiplicación.

El ejemplo 2 incluye la materia de estudio de estudio del ejemplo 1, en donde el planificador programa operandos distintos de cero en la unidad de multiplicación.

El Ejemplo 3 incluye la materia de estudio de los Ejemplos 1 y 2, en donde el planificador recupera los valores de operandos almacenados cuando los operandos se reciben en la lógica para leer operandos.

El Ejemplo 4 incluye la materia de estudio de los Ejemplos 1-3, comprendiendo además lógica de resultados de escritura para escribir un resultado de la multiplicación de los dos o más operandos en la unidad de multiplicación, en

donde el planificador escribe un valor cero en la lógica de resultados de escritura para los operandos que tienen un valor cero.

El Ejemplo 5 incluye la materia de estudio de los Ejemplos 1-4, comprendiendo además lógica de seguimiento de patrones para detectar uno o más segmentos de datos dispersos en un bloque de datos almacenado.

El Ejemplo 6 incluye la materia de estudio de estudio de los Ejemplos 1-5, en donde la lógica de seguimiento de patrones comprende además lógica de reconocimiento de patrones para llevar a cabo una operación de cuadro delimitador en el bloque de datos para determinar una similitud de los datos.

El Ejemplo 7 incluye la materia de estudio de los Ejemplos 1-6, en donde la lógica de reconocimiento de patrones rastrea datos almacenados en un dispositivo de memoria.

El Ejemplo 8 incluye la materia de estudio de los Ejemplos 1-7, en donde la lógica de reconocimiento de patrones rastrea datos almacenados en un dispositivo de memoria caché.

El Ejemplo 9 incluye la materia de estudio de los Ejemplos 1-8, en donde la lógica de reconocimiento de patrones rastrea datos almacenados en una tabla de páginas.

El Ejemplo 10 incluye la materia de estudio de los Ejemplos 1-9, en donde la lógica de seguimiento de patrones comprende además lógica de segmento para registrar una ubicación de dirección para cada segmento detectado de datos dispersos.

El Ejemplo 11 incluye la materia de estudio de los Ejemplos 1-10, comprendiendo además lógica para detectar la compresión de un matriz dispersa y una memoria intermedia comprimida dispersa para almacenar la matriz dispersa comprimida.

El Ejemplo 12 incluye la materia de estudio de los Ejemplos 1-11, en donde la matriz dispersa comprimida se genera dinámicamente basándose en un índice disperso.

El Ejemplo 13 incluye la materia de estudio de los Ejemplos 1-12, en donde la matriz dispersa comprimida comprende un matriz dispersa frecuentemente procesada en la una o más unidades de procesamiento.

Algunas formas de realización hacen referencia al Ejemplo 14 que incluye un aparato, que comprende un procesador gráfico que incluye varias unidades de ejecución (EU) y lógica para subdividir los varios EU y asignar cada subdivisión de EU para ejecutar subprocesos asociados con una capa de red neuronal.

El Ejemplo 15 incluye la materia de estudio del Ejemplo 14, en donde la subdivisión de UE comprende una primera subdivisión asignada para ejecutar subprocesos de capa de convolución, una segunda subdivisión asignada para ejecutar subprocesos de capa de desvío, una tercera subdivisión asignada para ejecutar subprocesos de capa de unidad lineal rectificada y una cuarta subdivisión asignada para ejecutar agrupación de subprocesos de capa

El Ejemplo 16 incluye la materia de estudio de los Ejemplos 14 y 15, en donde la lógica para subdividir los varios UE comparte resultados de ejecución entre cada capa de red neuronal.

Algunas formas de realización hacen referencia al Ejemplo 17 que incluye un método para facilitar el procesamiento de una matriz dispersa, que comprende recibir operandos en un elemento de procesamiento, determinar si uno o más de los operandos tienen un valor cero e impedir la programación de un operando en una unidad de multiplicación tras la determinación de que el operando tiene un valor cero.

El Ejemplo 18 incluye la materia de estudio del Ejemplo 17, comprendiendo además programar la multiplicación incluyendo un operando en la unidad de multiplicación sobre una determinación que el operando tiene un valor distinto de cero.

El Ejemplo 19 incluye la materia de estudio de los Ejemplos 17 y 18, comprendiendo además detectar uno o más segmentos de datos dispersos en un bloque de datos almacenado

El Ejemplo 20 incluye la materia de estudio de los Ejemplos 17-19, en donde la detección de uno o más segmentos de datos dispersos en el bloque de datos almacenado comprende llevar a cabo una operación de cuadro delimitador en el bloque de datos para determinar una similitud de los datos.

El Ejemplo 21 incluye la materia de estudio de los Ejemplos 17-20, comprendiendo además registrar una ubicación de dirección para cada segmento detectado de datos dispersos.

REIVINDICACIONES

1. Un aparato para procesar matrices dispersas, que comprende:
varias unidades de procesamiento gráfico (614) cada una de las cuales comprende un mecanismo de procesamiento de matrices dispersas (610) que incluye:

un planificador (613) incluido en un elemento de procesamiento (700) de la unidad de procesamiento gráfico (614), en donde el elemento de procesamiento (700) comprende además:

una lógica (701) configurada para leer operandos incluidos en instrucciones relacionadas con la multiplicación de las matrices dispersas;
una unidad de multiplicación (702) configurada para multiplicar dos o más operandos; y
una lógica de escritura del resultado (703) configurada para escribir un resultado de la multiplicación de los dos o más operandos en la unidad de multiplicación (702);
en donde el planificador (613) se configura para recuperar valores de operandos almacenados cuando los operandos se leen en la lógica (701),
en donde el planificador (613) se configura además para identificar operandos que tienen un valor cero en las matrices dispersas que se están procesando y evitar la programación de los operandos que tienen el valor cero en la unidad de multiplicación (702) y programar operandos distintos de cero en la unidad de multiplicación (702),
en donde el planificador (613) se configura además para escribir un valor cero en la lógica de resultados de escritura (703) para los operandos que tienen un valor cero,
caracterizado por que el mecanismo de procesamiento de matrices dispersas (610) comprende además:

un rastreador de patrones dispersos (615) configurado para determinar patrones de dispersidad de las matrices dispersas; y
una lógica de compresión (617) configurada para comprimir las matrices dispersas, en donde las matrices dispersas comprimidas se generan dinámicamente a partir de las matrices dispersas, basándose en índices dispersos, en los que los índices dispersos se definen como porcentaje de entradas distintas de cero de las matrices dispersas de acuerdo con los patrones de dispersidad determinados;
una memoria intermedia comprimida dispersa para almacenar las matrices dispersas comprimidas,
en donde la lógica de compresión (617) se configura además para descomprimir las matrices dispersas comprimidas en las matrices dispersas antes de su procesamiento por la una o más unidades de procesamiento.

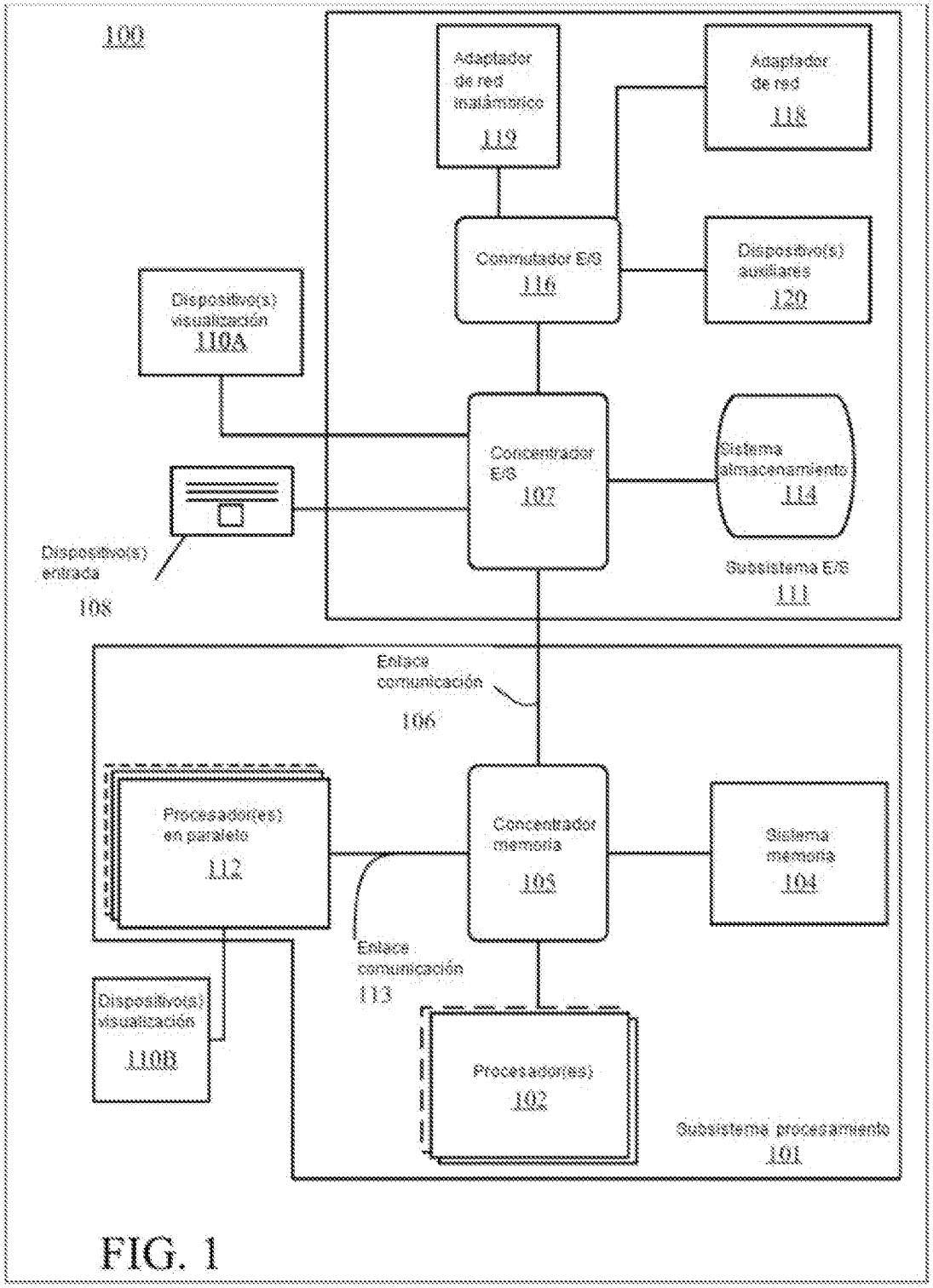

100
Adaptador de red inalámbrico 119
Adaptador de red 118
Conmutador E/S 116
Dispositivo(s) auxiliares 120
Dispositivo(s) visualización 110A
Concentrador E/S 107
Sistema almacenamiento 114
Subsistema E/S 111
Dispositivo(s) entrada 108
Enlace comunicación 106
Procesador(es) en paralelo 112
Concentrador memoria 105
Sistema memoria 104
Enlace comunicación 113
Dispositivo(s) visualización 110B
Procesador(es) 102
Subsistema procesamiento 101

FIG. 1

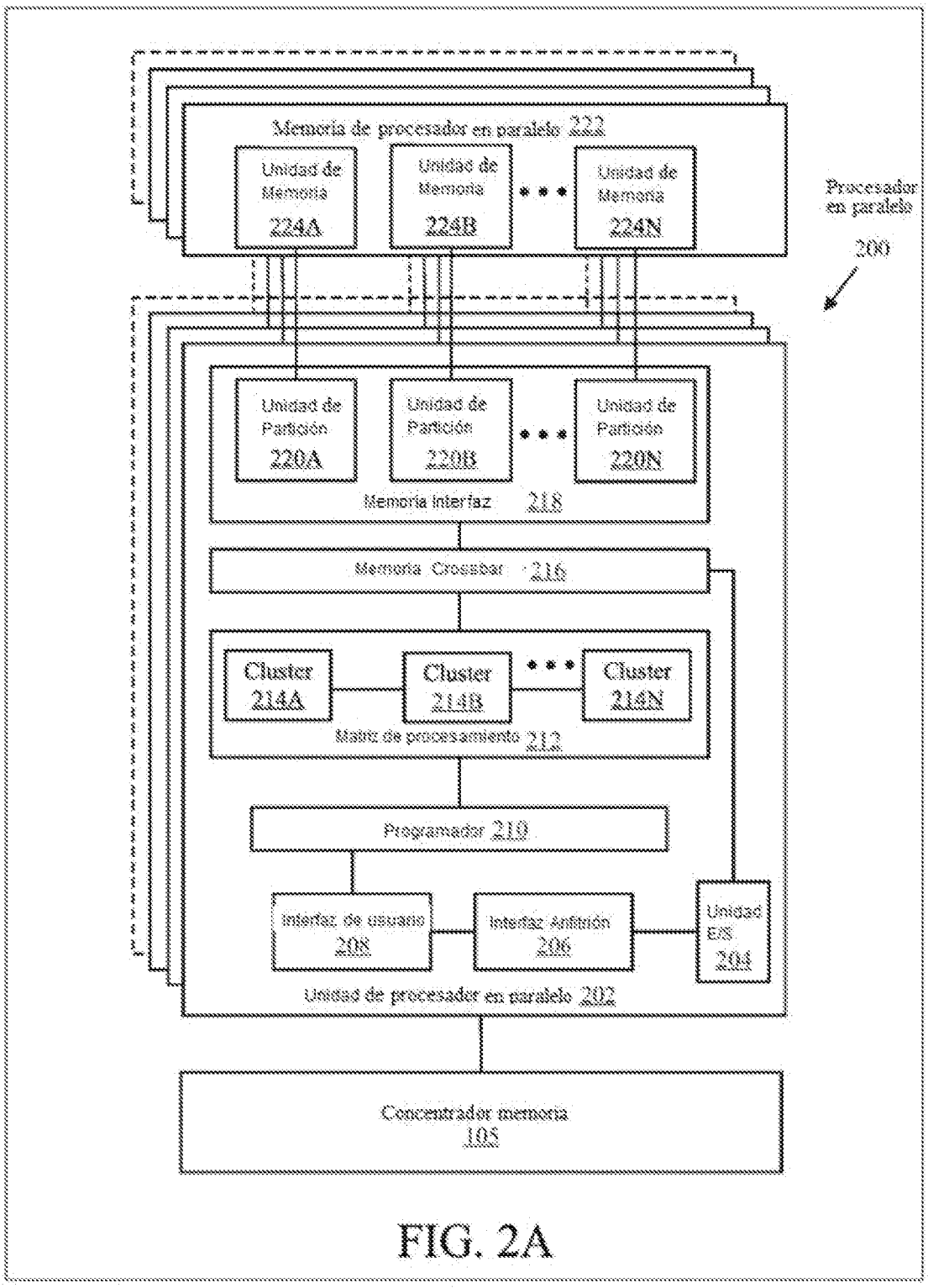

Memoria de procesador en paralelo 222
Unidad de Memoria 224A
Unidad de Memoria 224B
Unidad de Memoria 224N
Procesador en paralelo 200
Unidad de Partición 220A
Unidad de Partición 220B
Unidad de Partición 220N
Memoria interfaz 218
Memoria Crossbar 216
Cluster 214A
Cluster 214B
Cluster 214N
Matriz de procesamiento 212
Programador 210
Interfaz de usuario 208
Interfaz Anfitrión 206
Unidad E/S 204
Unidad de procesador en paralelo 202
Concentrador memoria 105

FIG. 2A

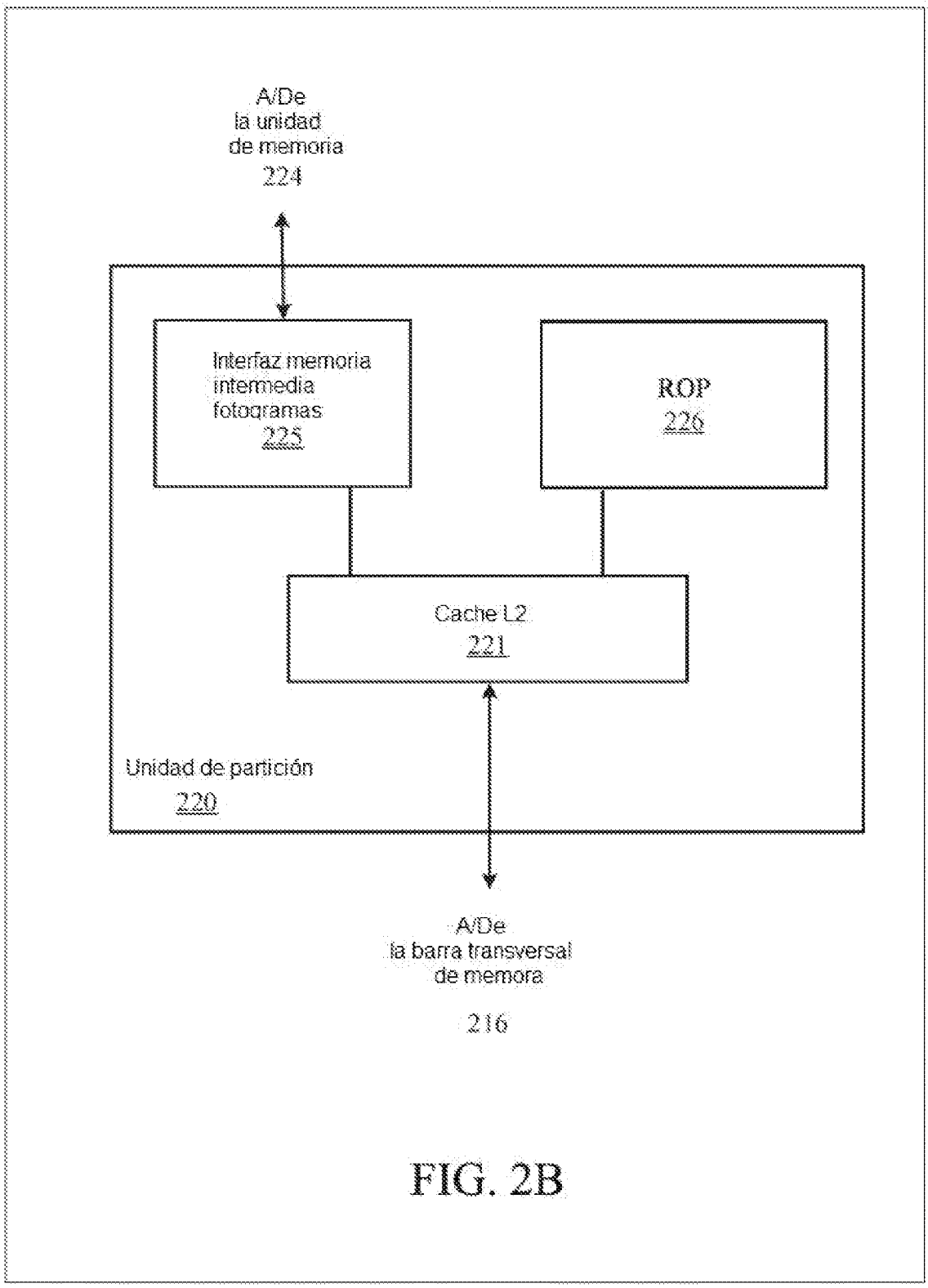

A/De
la unidad
de memoria
224
Interfaz memoria
intermedia
fotogramas
225
ROP
226
Cache L2
221
Unidad de partición
220
A/De
la barra transversal
de memora
216

FIG. 2B

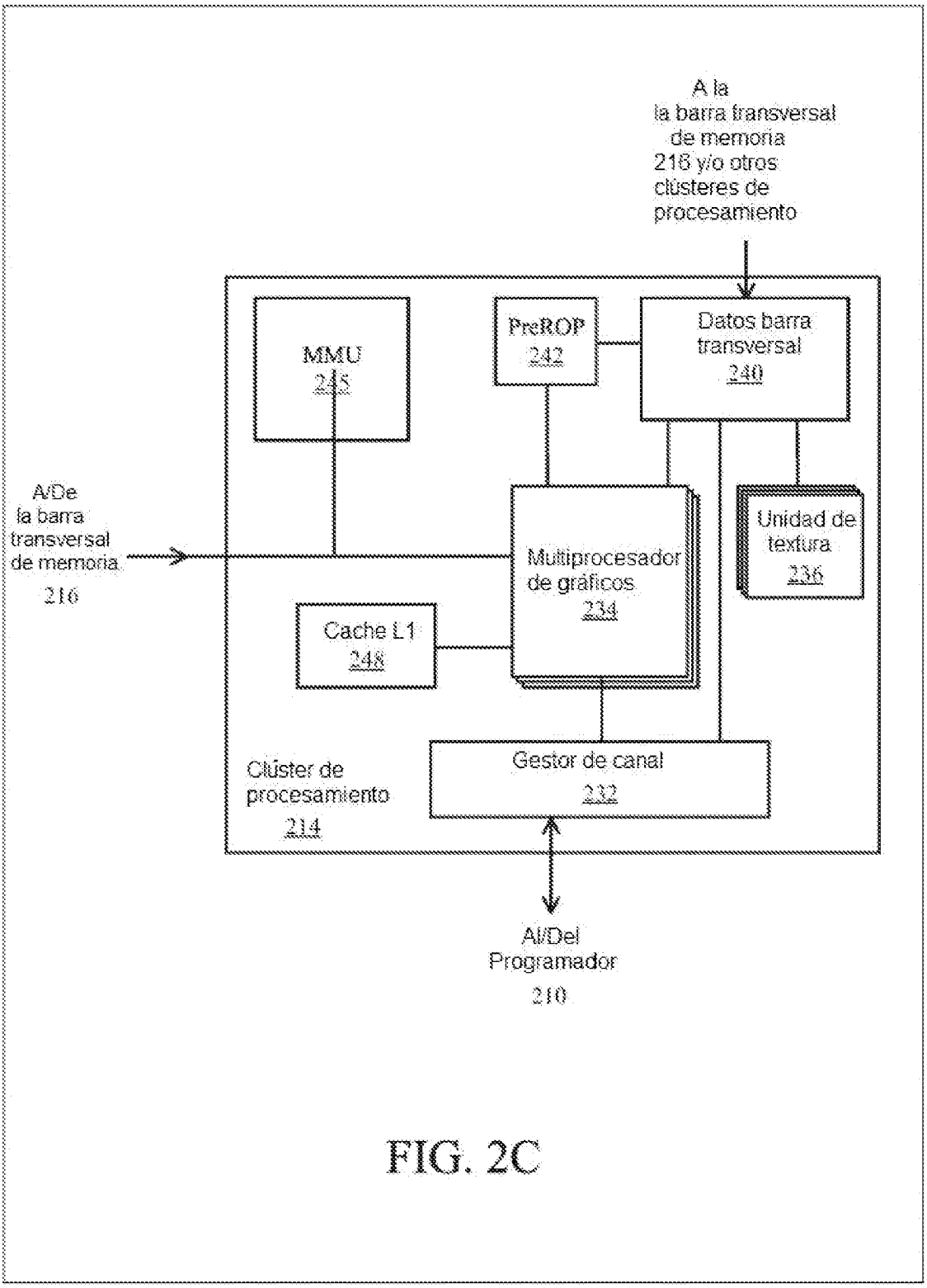

A la
la barra transversal
de memoria
216 y/o otros
clústeres de
procesamiento
MMU
245
PreROP
242
Datos barra
transversal
240
A/De
la barra
transversal
de memoria
216
Multiprocesador
de gráficos
234
Unidad de
textura
236
Cache L1
248
Clúster de
procesamiento
214
Gestor de canal
232
Al/Del
Programador
210

FIG. 2C

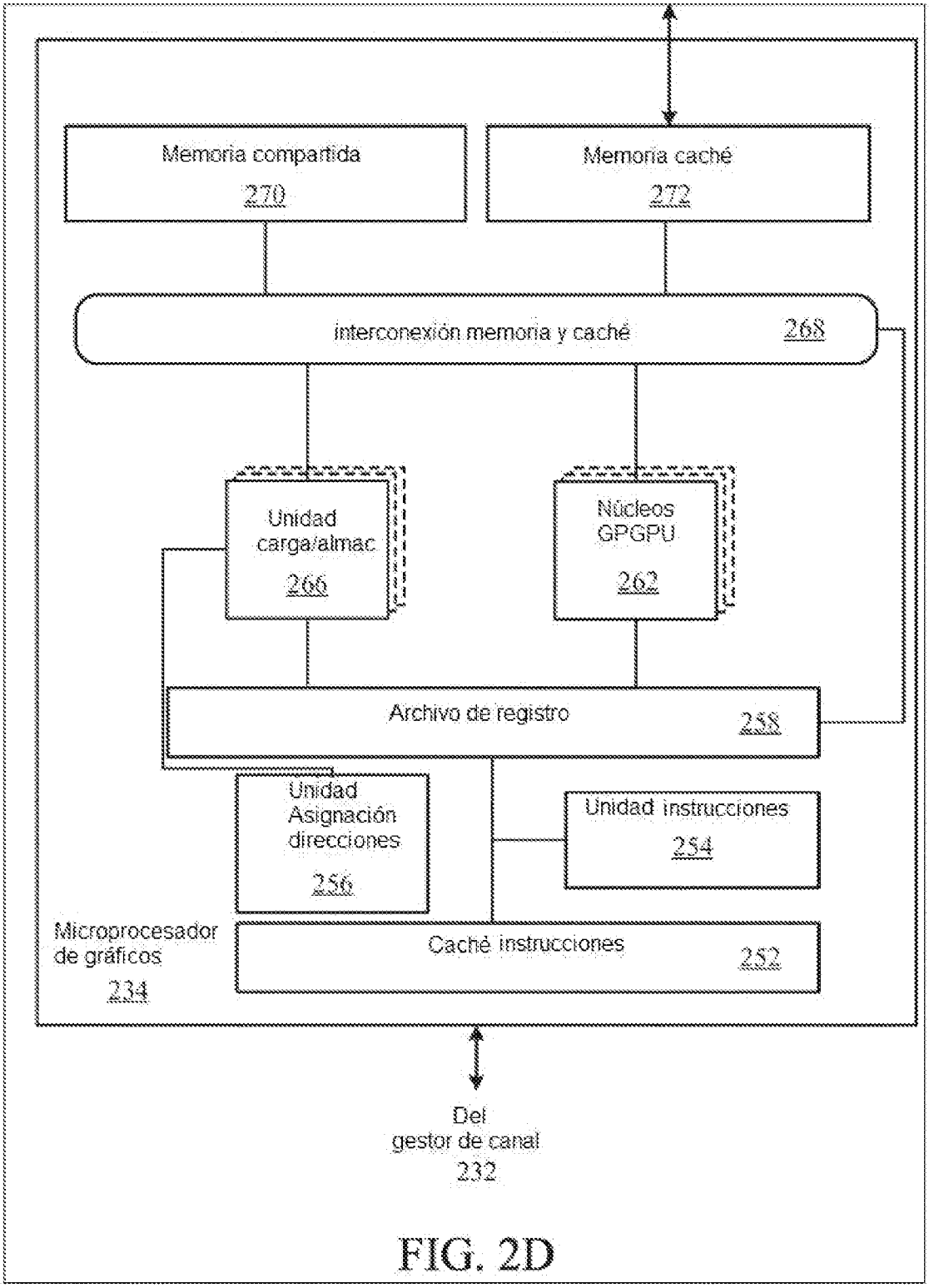

Memoria compartida
270
Memoria caché
272
interconexión memoria y caché
268
Unidad
carga/almac
266
Núcleos
GPGPU
262
Archivo de registro
258
Unidad
Asignación
direcciones
256
Unidad instrucciones
254
Microprocesador
de gráficos
234
Caché instrucciones
252
Del
gestor de canal
232

FIG. 2D

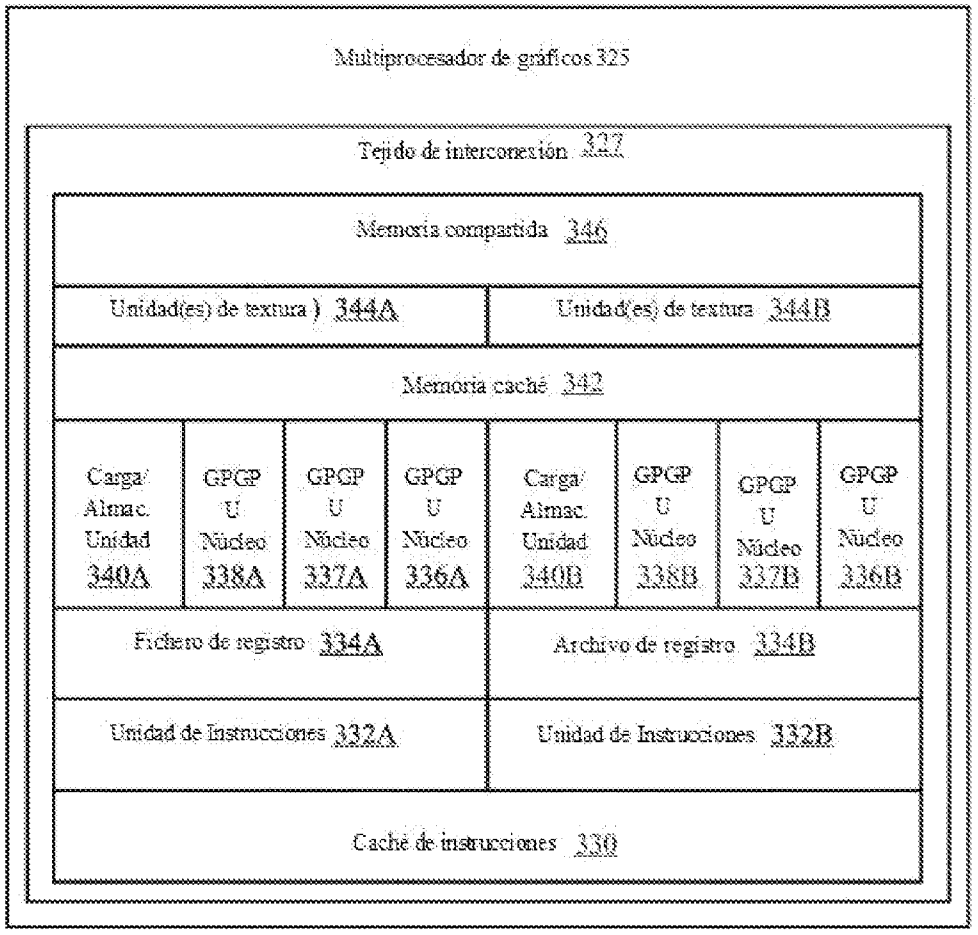

Multiprocesador de gráficos 325
Tejido de interconexión 327
Memoria compartida 346
Unidad(es) de textura) 344A
Unidad(es) de textura 344B
Memoria caché 342
Carga/ Almac. Unidad 340A
GPGP U Núcleo 338A
GPGP U Núcleo 337A
GPGP U Núcleo 336A
Carga/ Almac. Unidad 340B
GPGP U Núcleo 338B
GPGP U Núcleo 337B
GPGP U Núcleo 336B
Fichero de registro 334A
Archivo de registro 334B
Unidad de Instrucciones 332A
Unidad de Instrucciones 332B
Caché de instrucciones 330

FIG. 3A

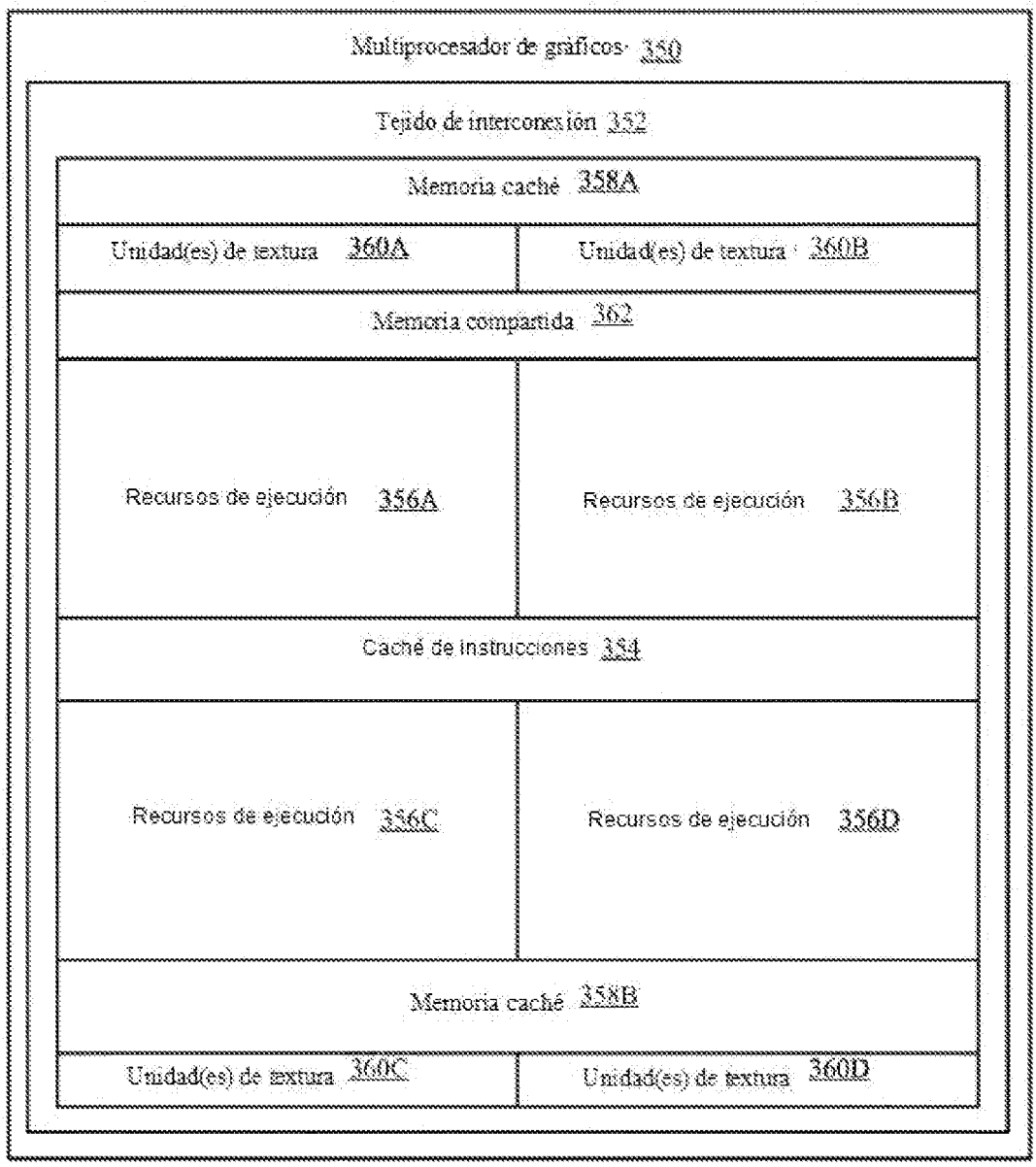

Multiprocesador de gráficos 350
Tejido de interconexión 352
Memoria caché 358A
Unidad(es) de textura 360A
Unidad(es) de textura 360B
Memoria compartida 362
Recursos de ejecución 356A
Recursos de ejecución 356B
Caché de instrucciones 354
Recursos de ejecución 356C
Recursos de ejecución 356D
Memoria caché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

FIG. 3B

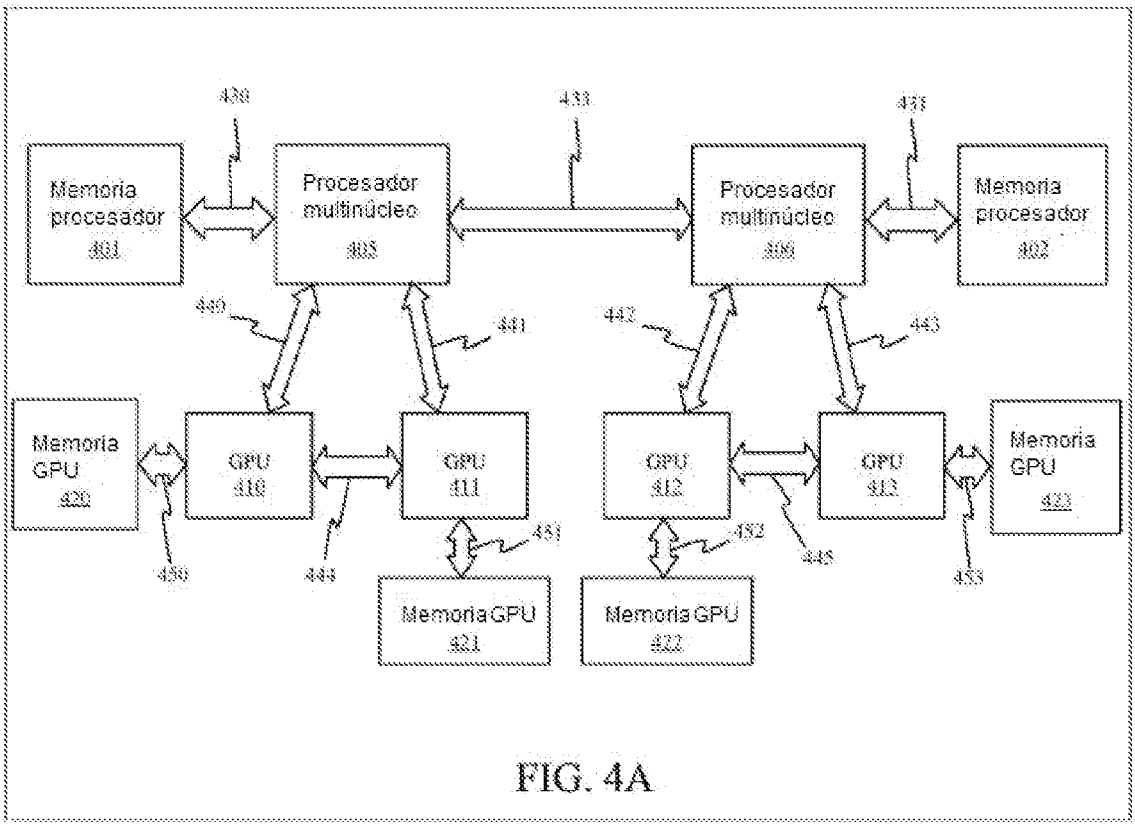

Memoria procesador 401
Procesador multinúcleo 403
Procesador multinúcleo 404
Memoria procesador 402
Memoria GPU 420
GPU 410
GPU 411
GPU 412
GPU 413
Memoria GPU 423
Memoria GPU 421
Memoria GPU 422

FIG. 4A

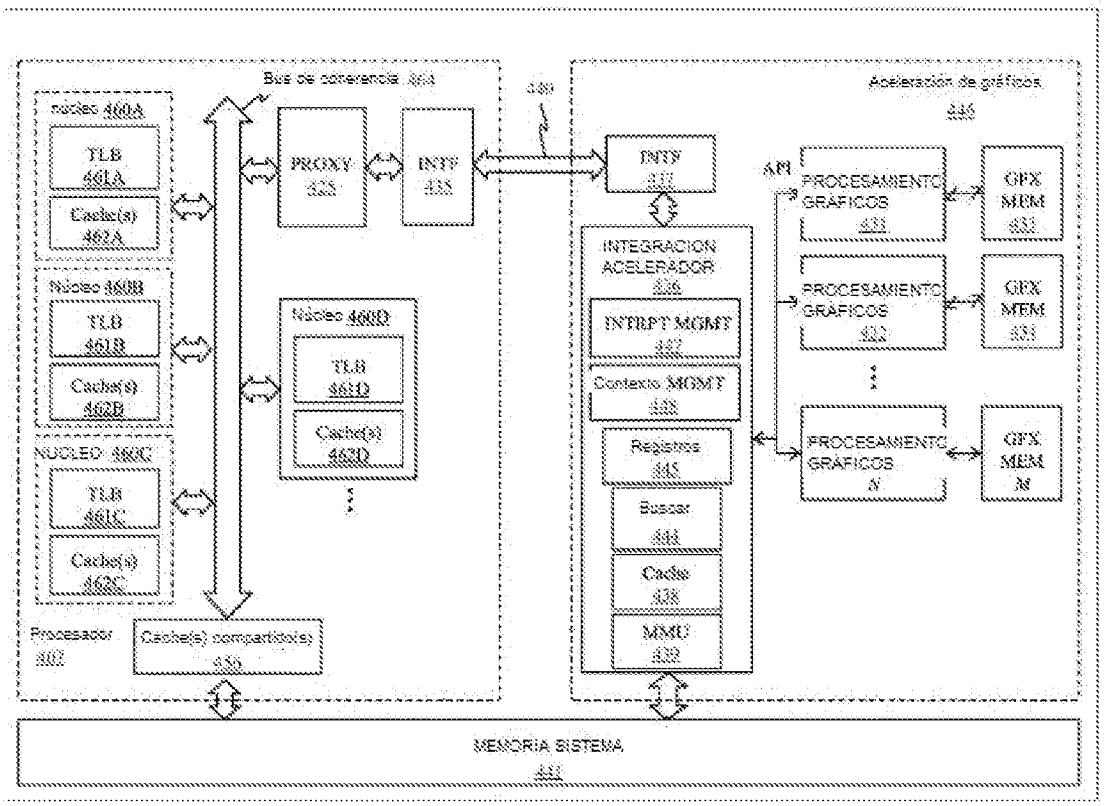

Bus de coherencia 464
Núcleo 460A
TLB 461A
Caché(s) 462A
Núcleo 460B
TLB 461B
Caché(s) 462B
NÚCLEO 460C
TLB 461C
Caché(s) 462C
Procesador 402
Caché(s) compartida(s) 456
PROXY 422
INTF 435
Núcleo 460D
TLB 461D
Caché(s) 462D
480
Aceleración de gráficos 446
INTF 437
INTEGRACIÓN ACELERADOR 436
INTRPT MGMT 442
Contexto MGMT 438
Registros 445
Buscar 444
Caché 438
MMU 439
API
PROCESAMIENTO GRÁFICOS 431
PROCESAMIENTO GRÁFICOS 432
PROCESAMIENTO GRÁFICOS N
GFX MEM 433
GFX MEM 434
GFX MEM N
MEMORIA SISTEMA 412

FIG. 4B

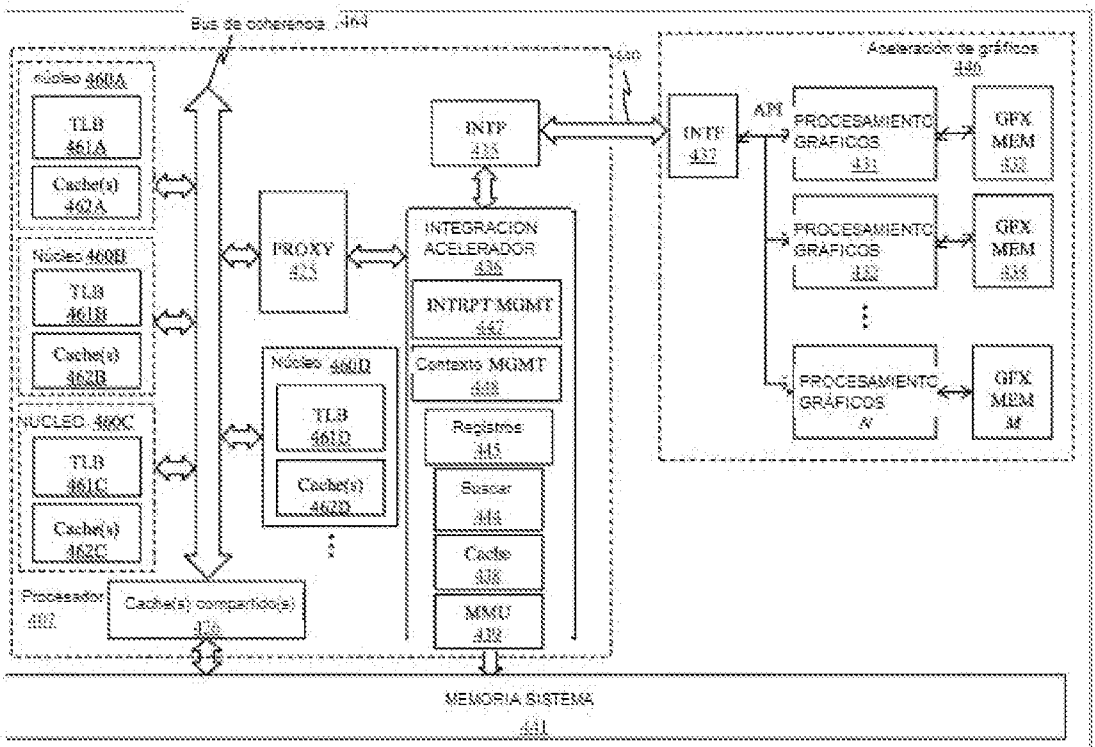

Bus de coherencia
Núcleo 460A
TLB 461A
Caché(s) 462A
Núcleo 460B
TLB 461B
Caché(s) 462B
NÚCLEO 460C
TLB 461C
Caché(s) 462C
Procesador 407
Caché(s) compartido(s)
PROXY 425
Núcleo 460D
TLB 461D
Caché(s) 462D
INTF 435
INTEGRACIÓN ACELERADOR 436
INTRPT MGMT 437
Contexto MGMT 438
Registros 445
Buscar 444
Caché 438
MMU 439
INTF 432
API
Aceleración de gráficos 440
PROCESAMIENTO GRÁFICOS 431
PROCESAMIENTO GRÁFICOS 433
PROCESAMIENTO GRÁFICOS N
GFX MEM 434
GFX MEM 435
GFX MEM M
MEMORIA SISTEMA 411

FIG. 4C

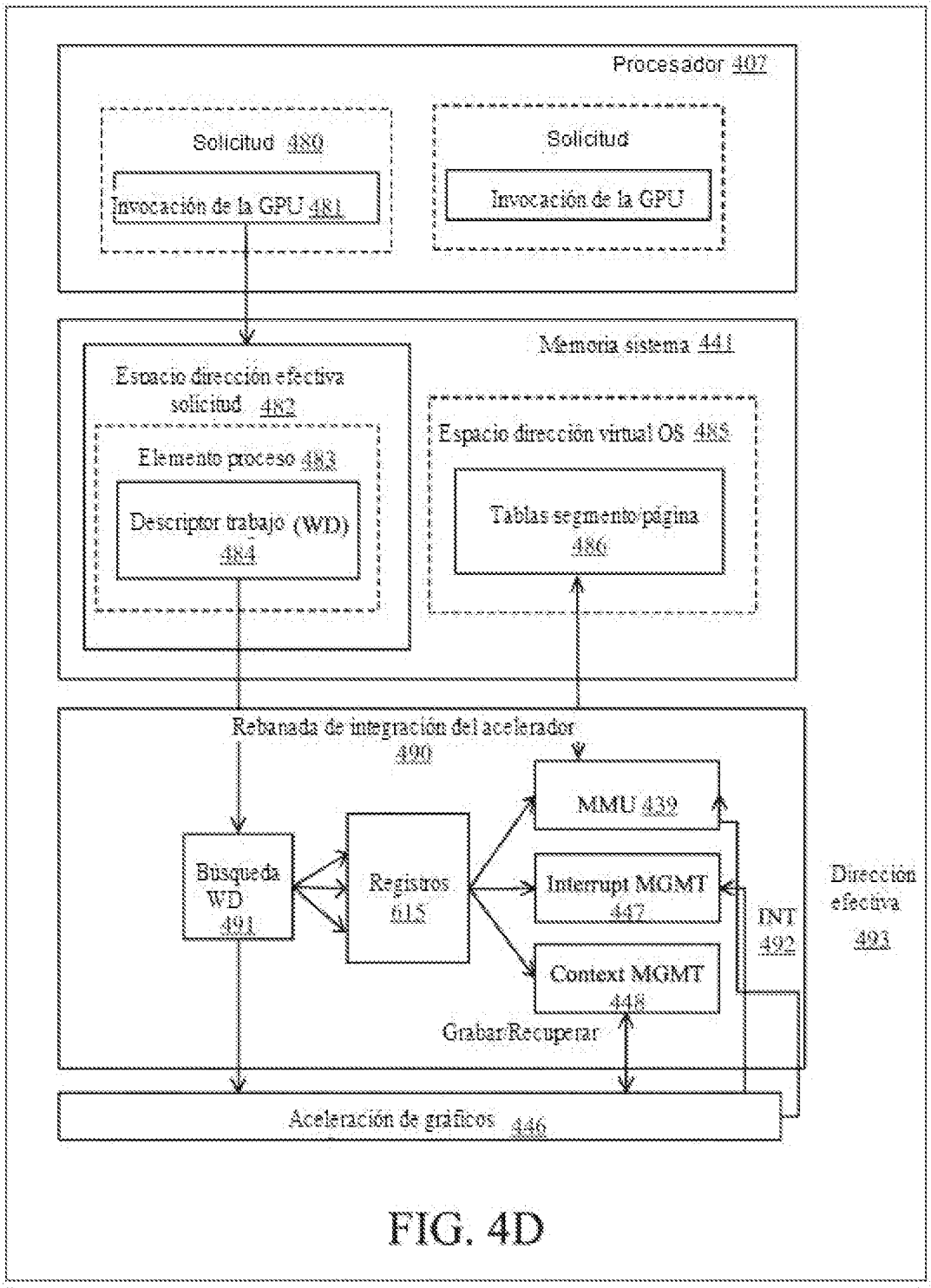

Procesador 407
Solicitud 480
Invocación de la GPU 481
Solicitud
Invocación de la GPU
Memoria sistema 441
Espacio dirección efectiva solicitud 482
Elemento proceso 483
Descriptor trabajo (WD) 484
Espacio dirección virtual OS 485
Tablas segmento/página 486
Rebanada de integración del acelerador 490
MMU 439
Búsqueda WD 491
Registros 415
Interrupt MGMT 447
Context MGMT 448
Grabar/Recuperar
INT 492
Dirección efectiva 423
Aceleración de gráficos 446

FIG. 4D

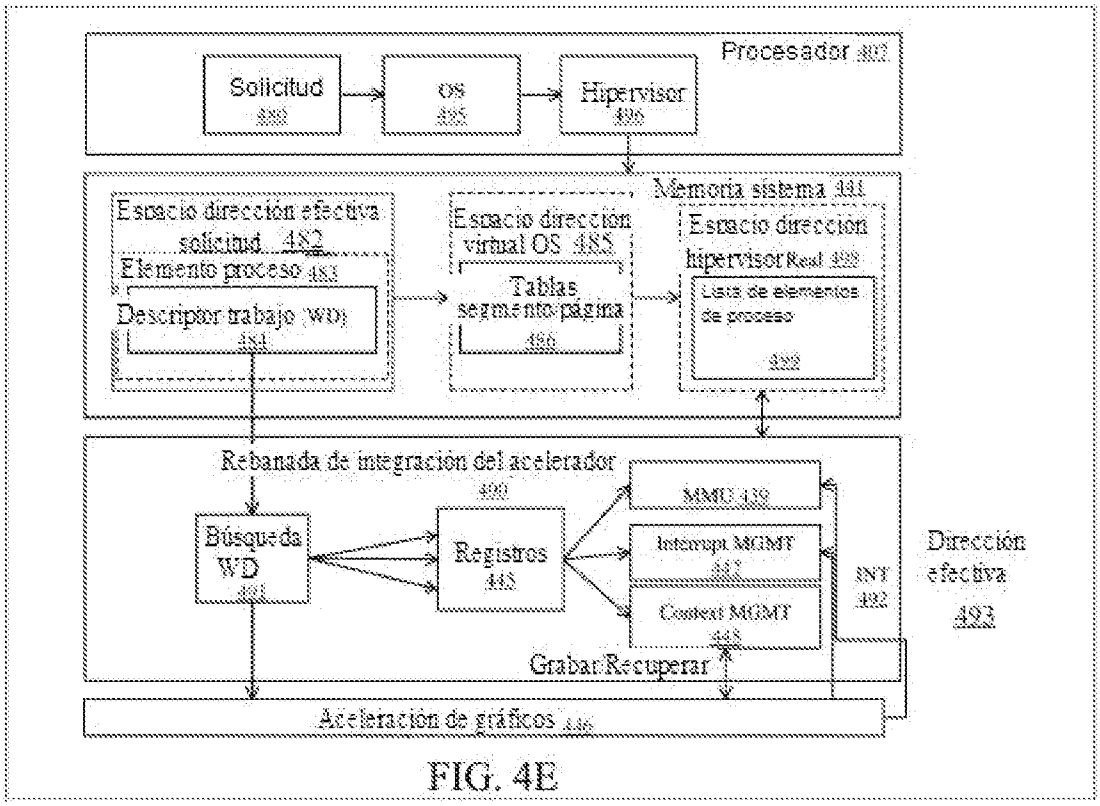

Procesador 402
Solicitud 480
OS 485
Hipervisor 406
Memoria sistema 431
Espacio dirección efectiva solicitud 482
Elemento proceso 483
Descriptor trabajo (WD) 484
Espacio dirección virtual OS 485
Tablas segmento página 486
Espacio dirección hipervisor Real 488
Lista de elementos de proceso 492
Rebanada de integración del acelerador 490
Búsqueda WD 491
Registros 443
MMU 439
Interrupt MGMT 447
Context MGMT 448
INT 492
Dirección efectiva 493
Grabar/Recuperar
Aceleración de gráficos 440

FIG. 4E

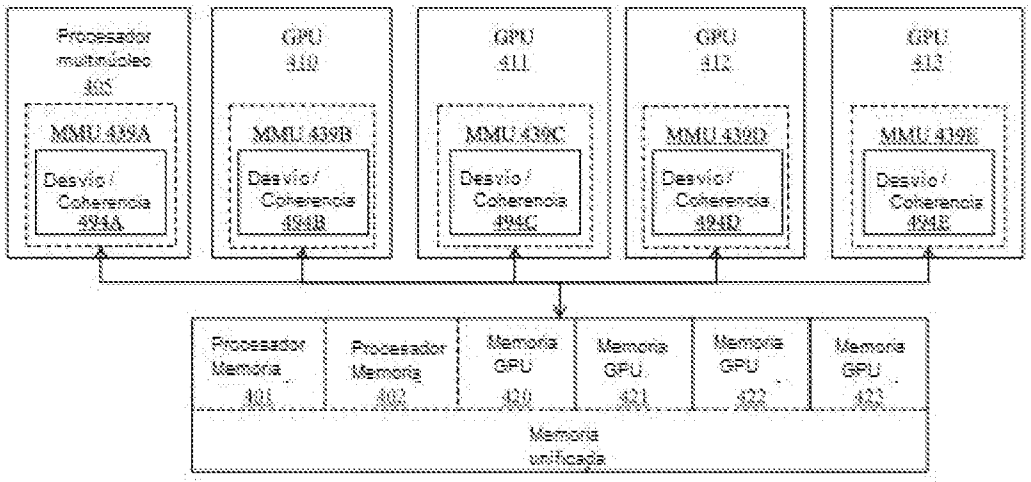

Procesador multinúcleo
405
MMU 439A
Desvío / Coherencia 494A
GPU
410
MMU 439B
Desvío / Coherencia 494B
GPU
411
MMU 439C
Desvío / Coherencia 494C
GPU
412
MMU 439D
Desvío / Coherencia 494D
GPU
413
MMU 439E
Desvío / Coherencia 494E
Procesador Memoria
401
Procesador Memoria
402
Memoria GPU
420
Memoria GPU
421
Memoria GPU
422
Memoria GPU
423
Memoria unificada

FIG. 4F

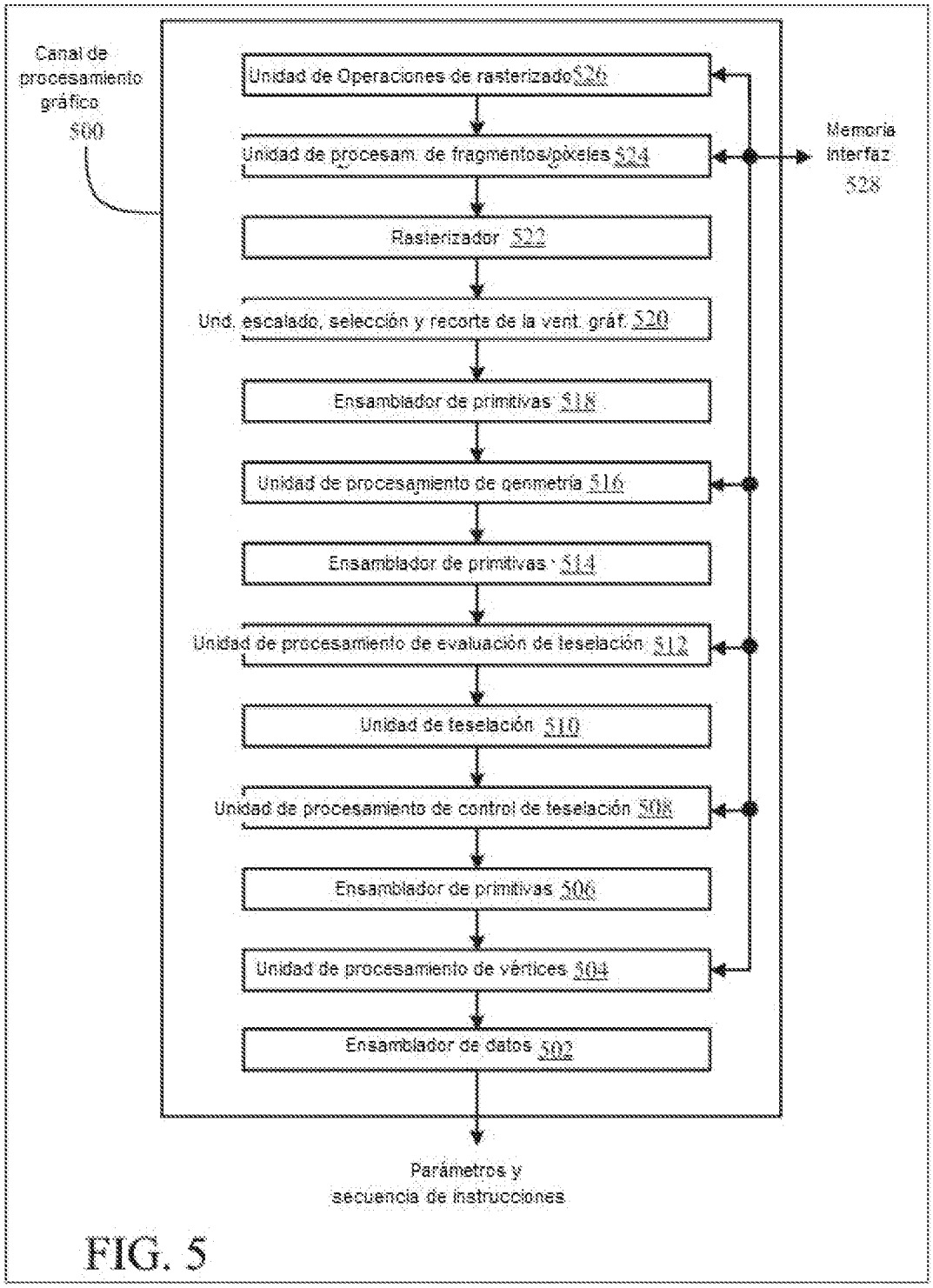

Canal de procesamiento gráfico 500
Unidad de Operaciones de rasterizado 526
Unidad de procesam. de fragmentos/píxeles 524
Memoria interfaz 528
Rasterizador 522
Und. escalado, selección y recorte de la vent. gráf. 520
Ensamblador de primitivas 518
Unidad de procesamiento de geometría 516
Ensamblador de primitivas 514
Unidad de procesamiento de evaluación de teselación 512
Unidad de teselación 510
Unidad de procesamiento de control de teselación 508
Ensamblador de primitivas 506
Unidad de procesamiento de vértices 504
Ensamblador de datos 502
Parámetros y secuencia de instrucciones

FIG. 5

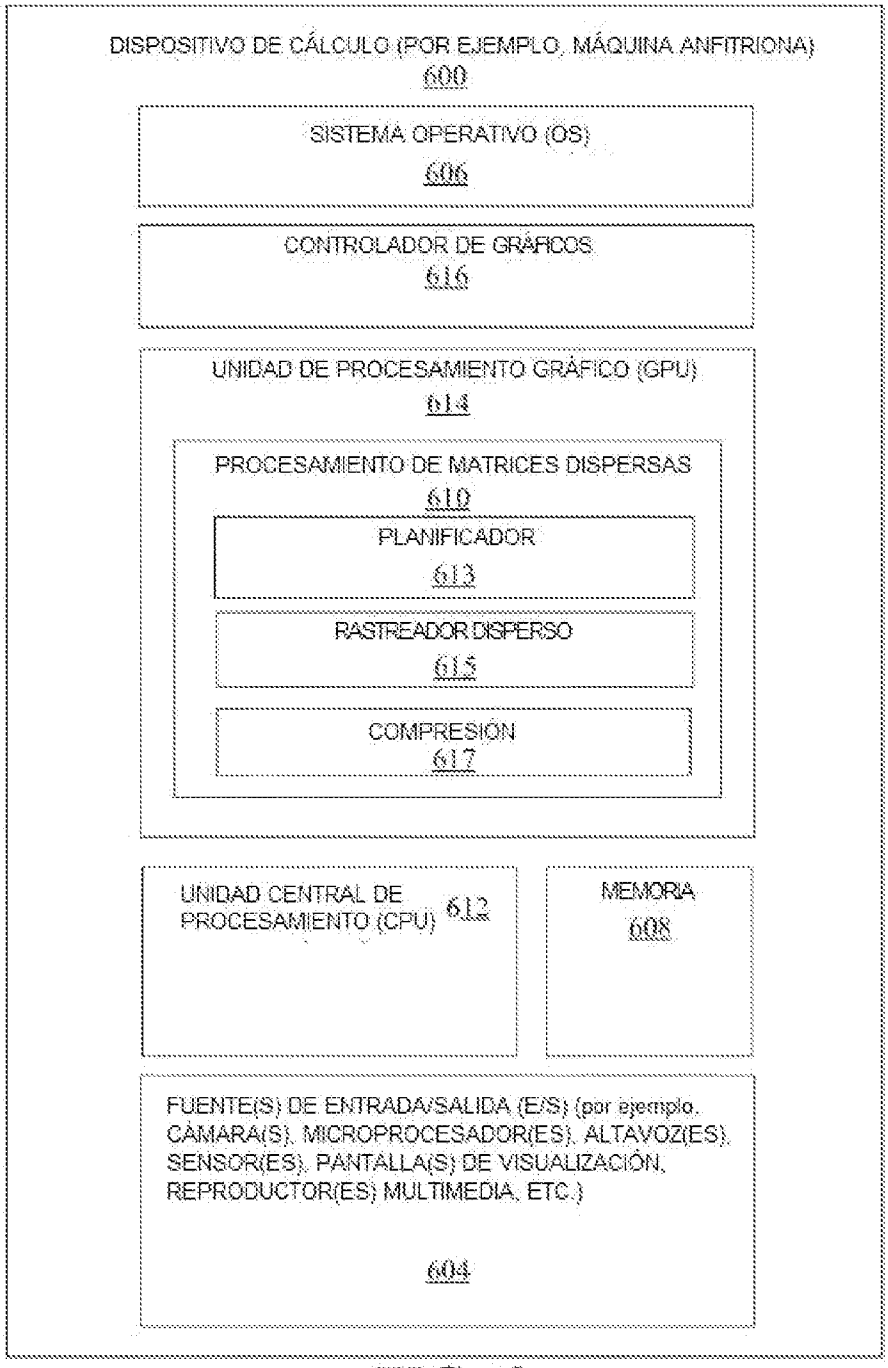

DISPOSITIVO DE CÁLCULO (POR EJEMPLO, MÁQUINA ANFITRIONA)
600
SISTEMA OPERATIVO (OS)
606
CONTROLADOR DE GRÁFICOS
616
UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)
614
PROCESAMIENTO DE MATRICES DISPERSAS
610
PLANIFICADOR
613
RASTREADOR DISPERSO
615
COMPRESIÓN
617
UNIDAD CENTRAL DE PROCESAMIENTO (CPU)
612
MEMORIA
608
FUENTE(S) DE ENTRADA/SALIDA (E/S) (por ejemplo, CÁMARA(S), MICROPROCESADOR(ES), ALTAVOZ(ES), SENSOR(ES), PANTALLA(S) DE VISUALIZACIÓN, REPRODUCTOR(ES) MULTIMEDIA, ETC.)
604

FIG. 6

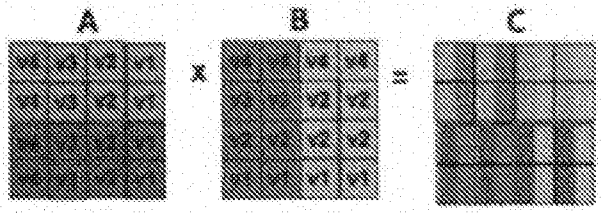


FIG. 7A

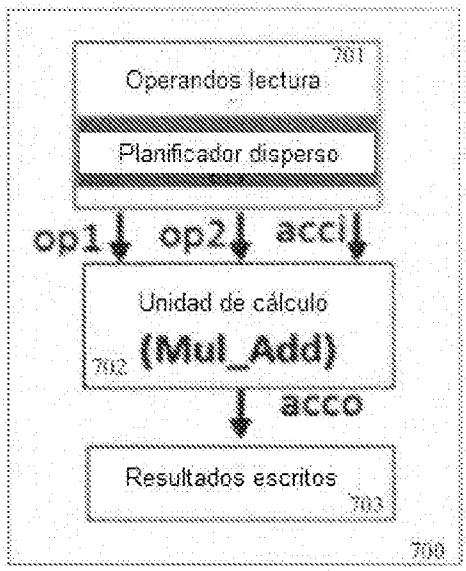


FIG. 7B

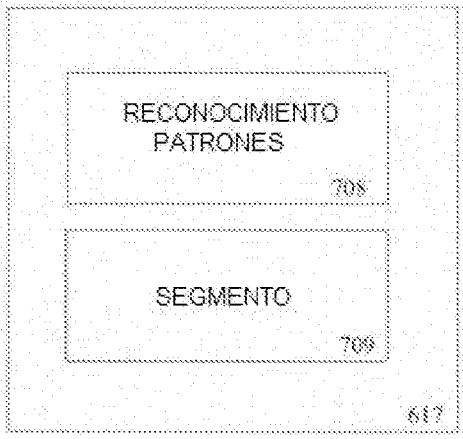


FIG. 7C

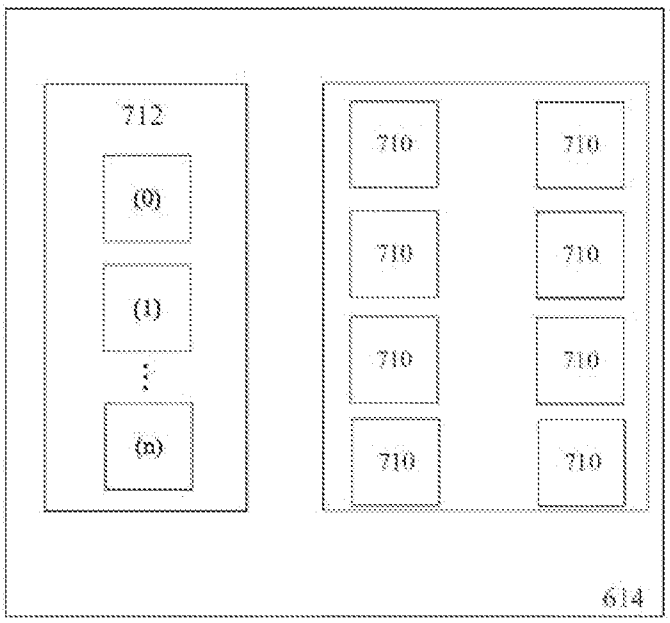

712
(0)
(1)
(n)
710
710
710
710
710
710
710
710
614

FIG. 7D

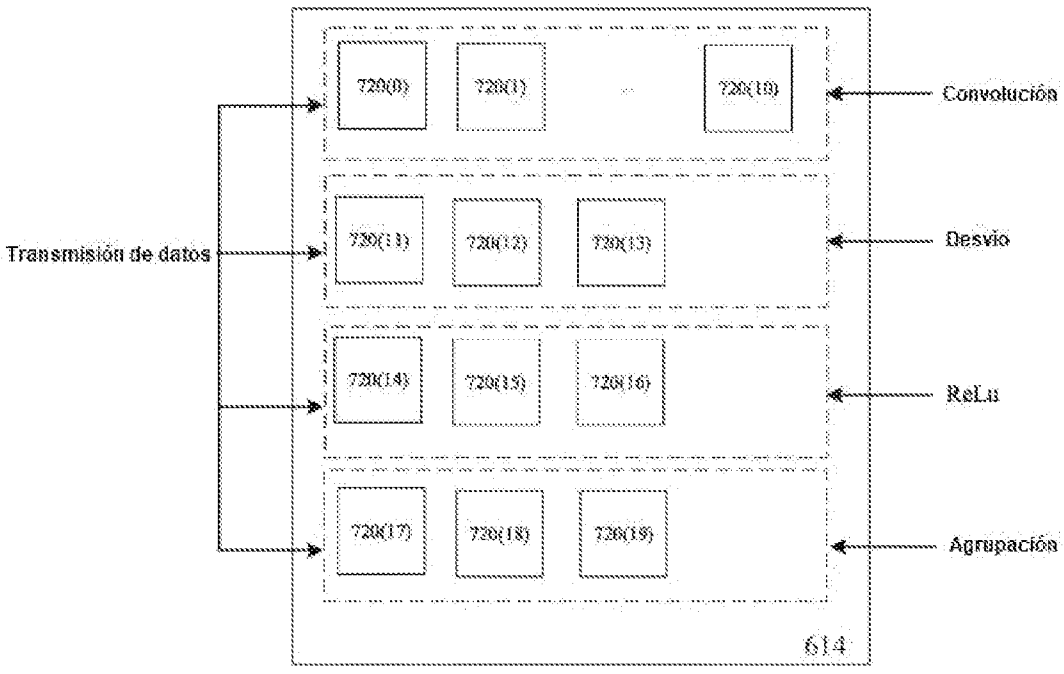

720(0)
720(1)
720(10)
720(11)
720(12)
720(13)
720(14)
720(15)
720(16)
720(17)
720(18)
720(19)
Transmisión de datos
Convolución
Desvío
ReLu
Agrupación
614

FIG. 7E

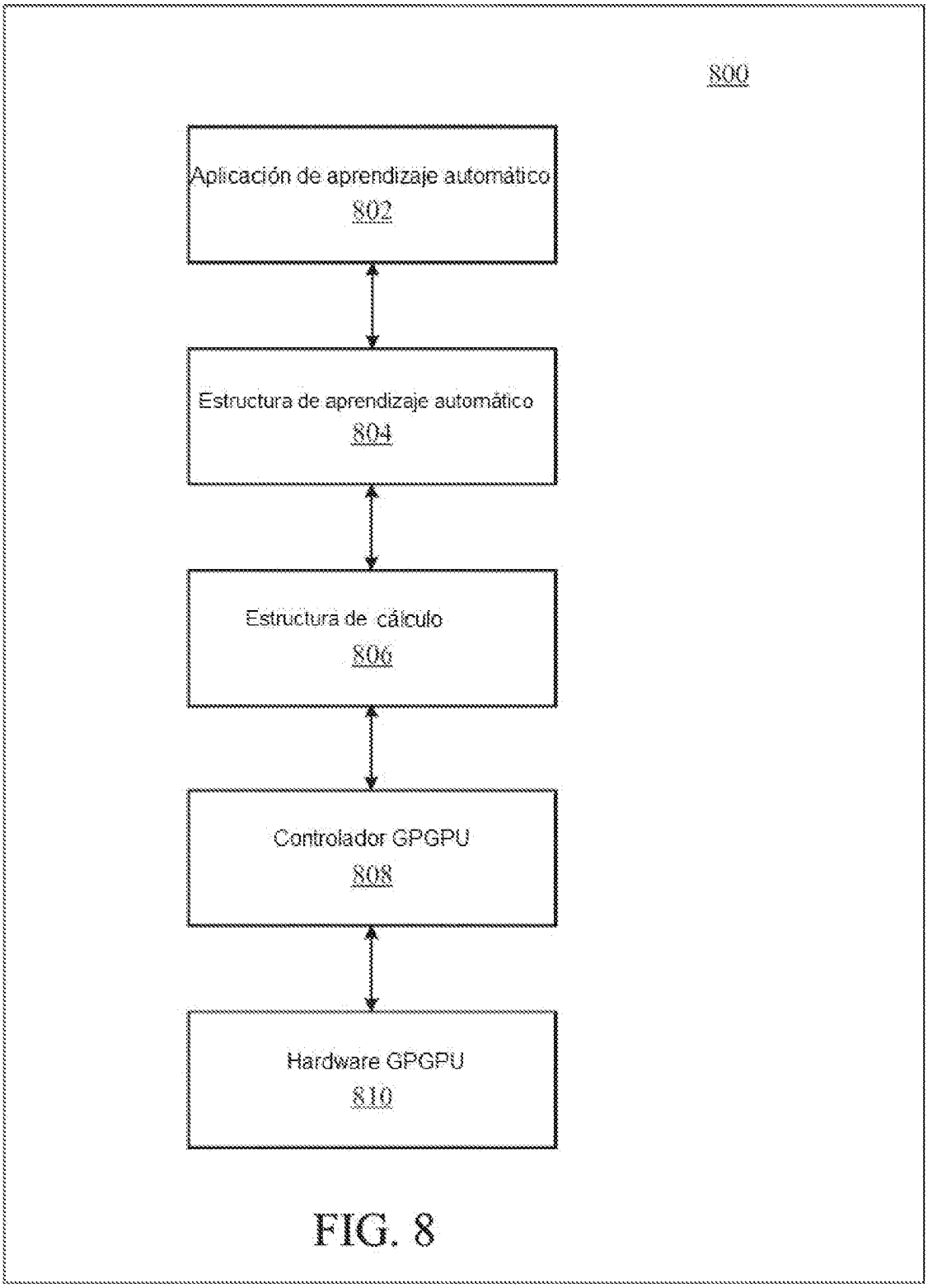

800
Aplicación de aprendizaje automático
802
Estructura de aprendizaje automático
804
Estructura de cálculo
806
Controlador GPGPU
808
Hardware GPGPU
810

FIG. 8

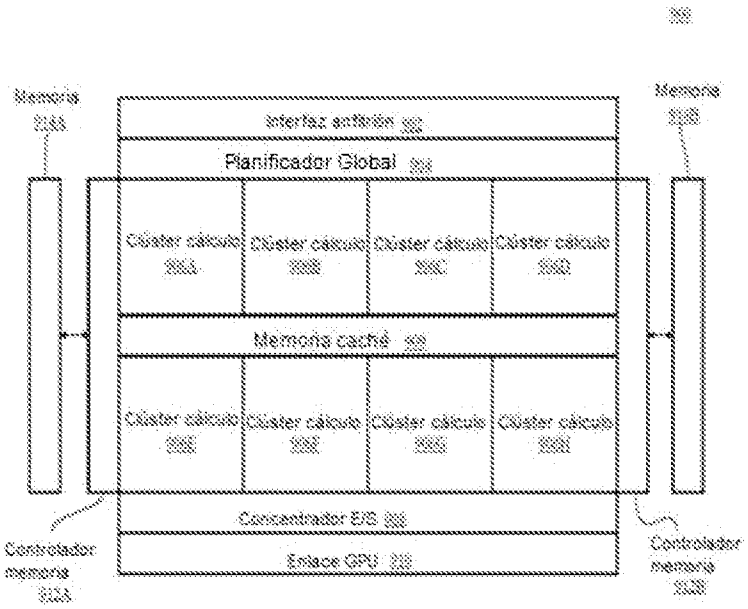

Memoria
Interfaz anfitrión
Memoria
Planificador Global
Clúster cálculo
Clúster cálculo
Clúster cálculo
Clúster cálculo
Memoria caché
Clúster cálculo
Clúster cálculo
Clúster cálculo
Clúster cálculo
Concentrador E/S
Enlace GPU
Controlador memoria
Controlador memoria

FIG. 9

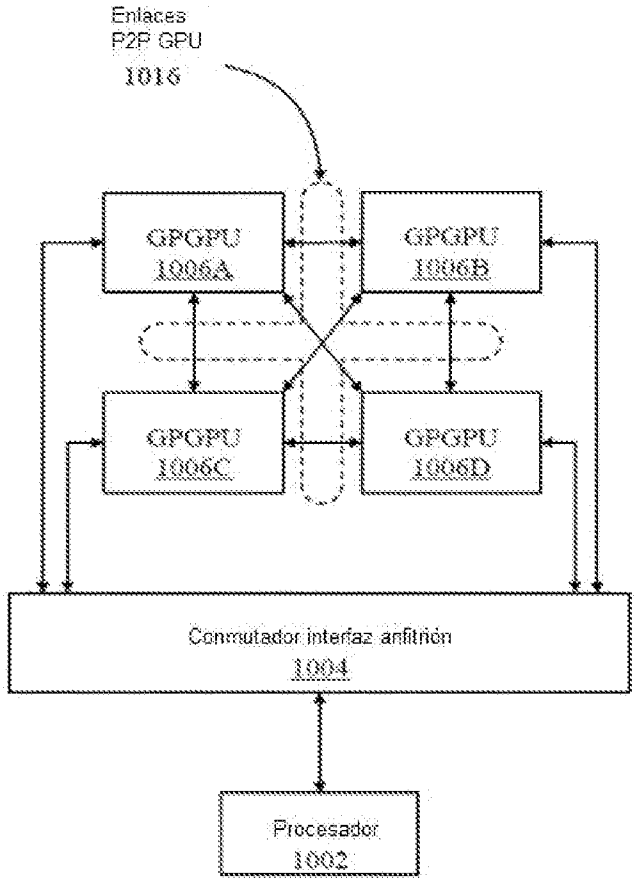

Enlaces
P2P GPU
1016
GPGPU
1006A
GPGPU
1006B
GPGPU
1006C
GPGPU
1006D
Conmutador interfaz anfitrión
1004
Procesador
1002

FIG. 10

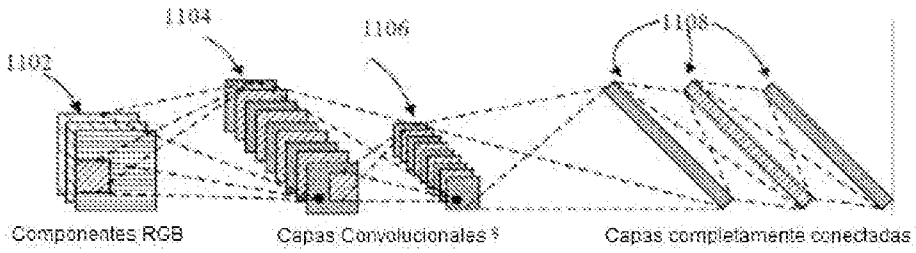


FIG. 11A

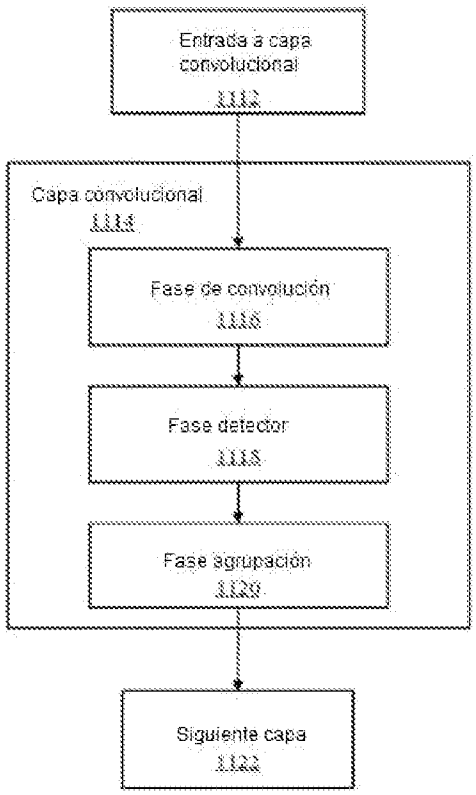


FIG. 11B

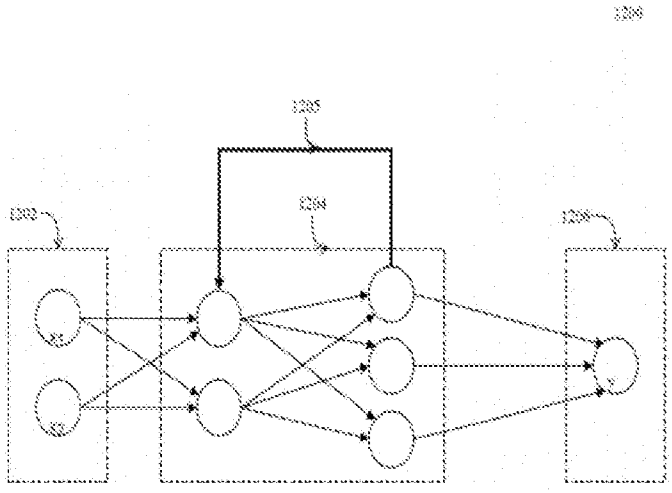


FIG. 12

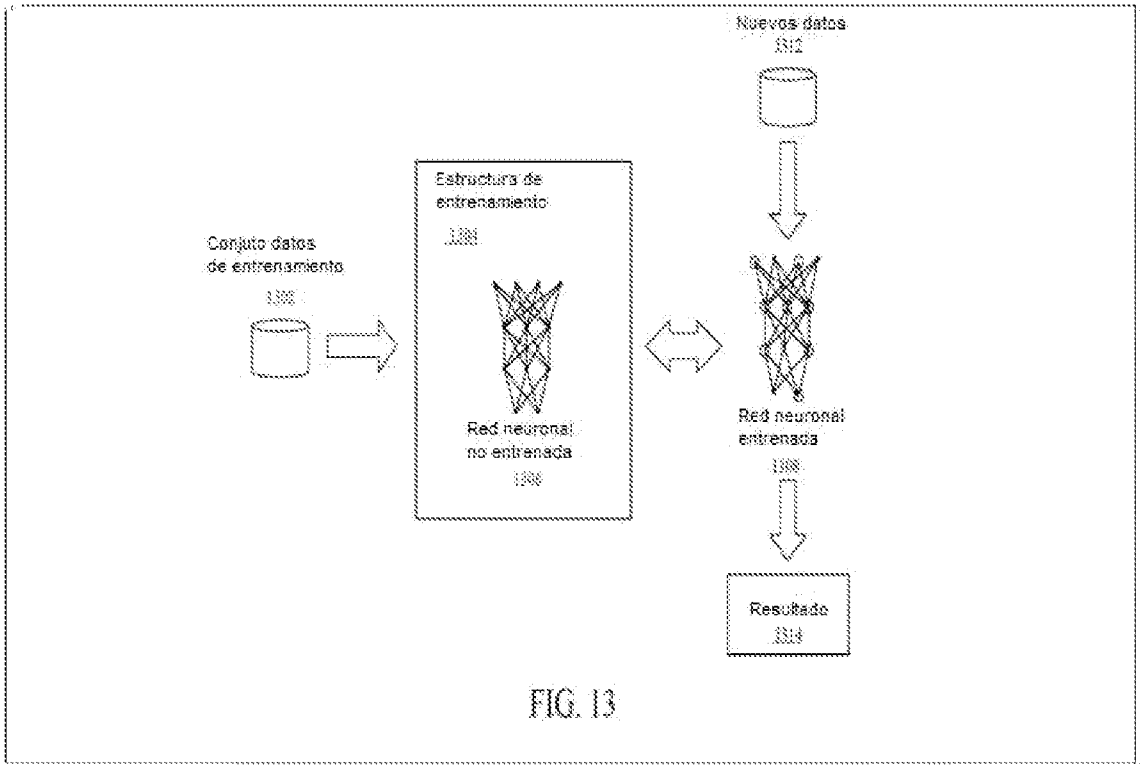

Nuevos datos
Estructura de entrenamiento
Conjunto datos de entrenamiento
Red neuronal no entrenada
Red neuronal entrenada
Resultado

FIG. 13

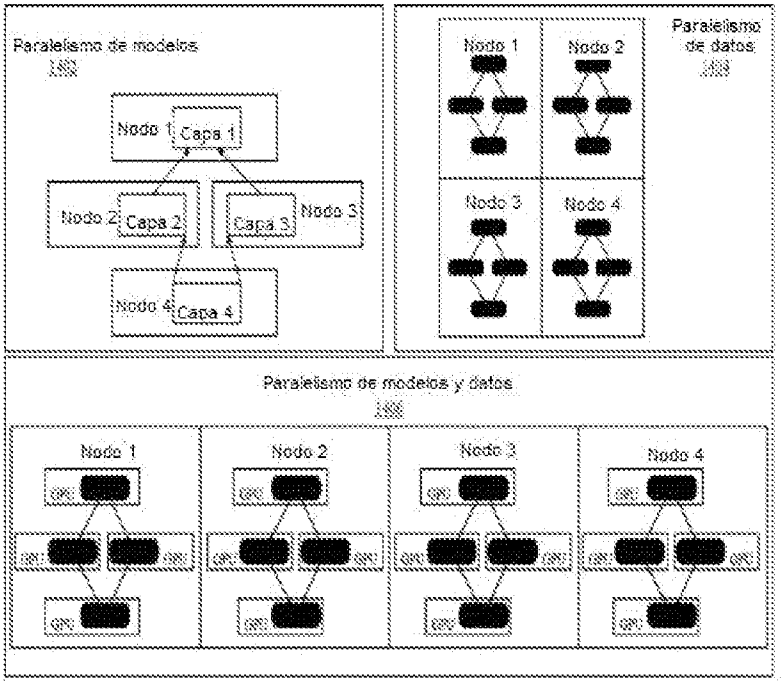

Paralelismo de modelos
Nodo 1
Capa 1
Nodo 2
Capa 2
Capa 3
Nodo 3
Nodo 4
Capa 4
Nodo 1
Nodo 2
Nodo 3
Nodo 4
Paralelismo de datos
Paralelismo de modelos y datos
Nodo 1
Nodo 2
Nodo 3
Nodo 4

FIG. 14

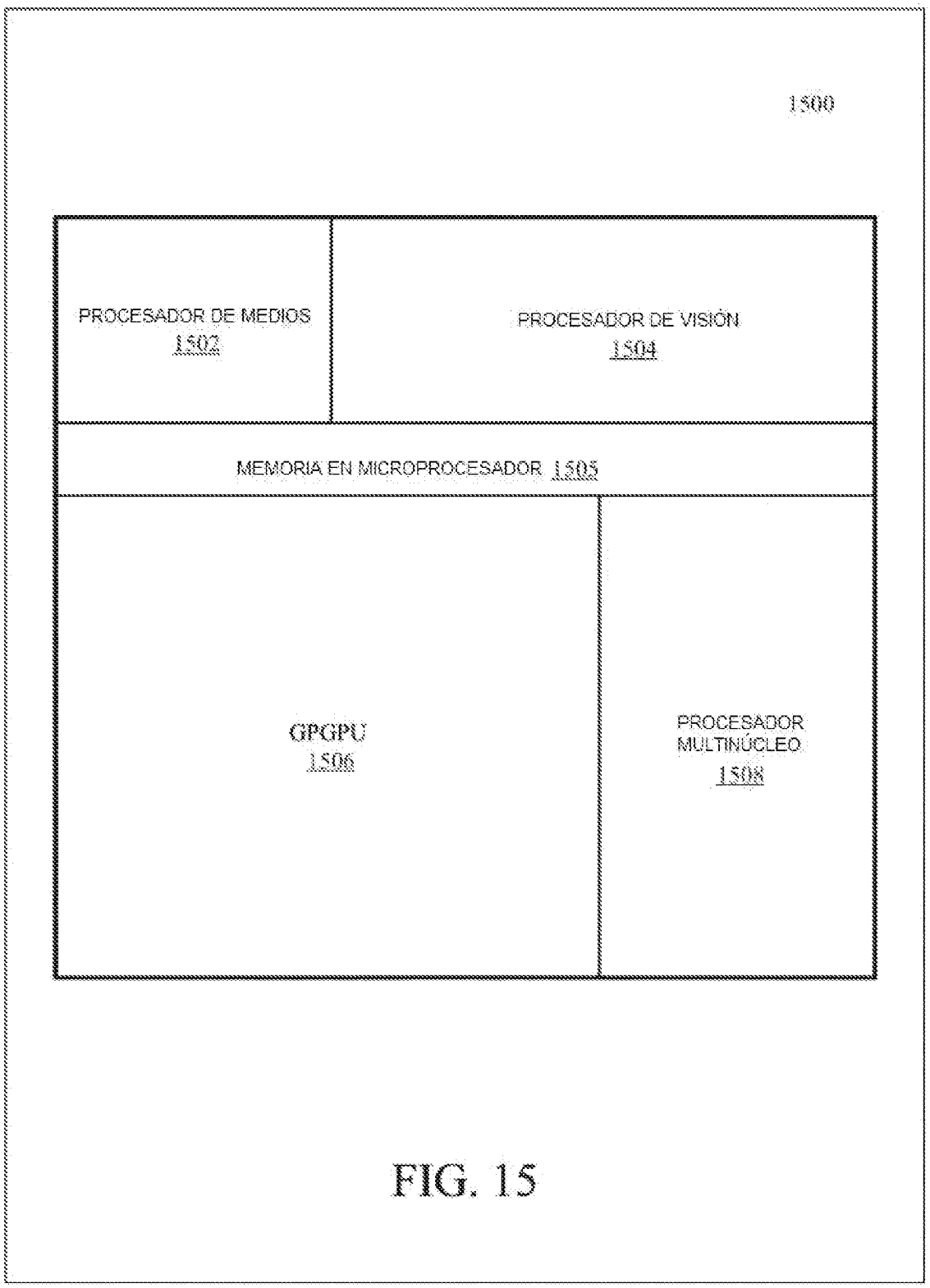

1500
PROCESADOR DE MEDIOS
1502
PROCESADOR DE VISIÓN
1504
MEMORIA EN MICROPROCESADOR 1505
GPGPU
1506
PROCESADOR
MULTINÚCLEO
1508

FIG. 15

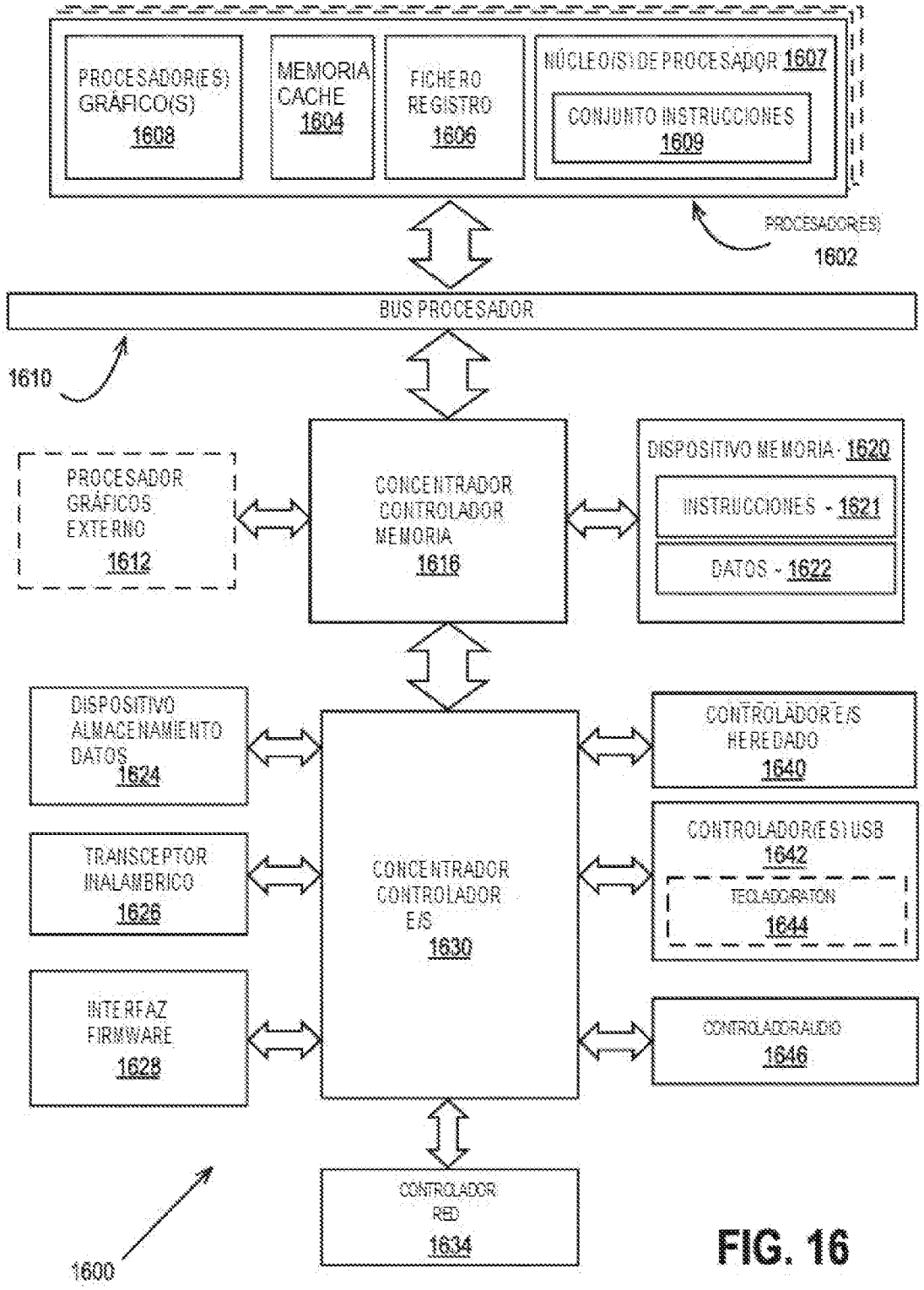

PROCESADOR(ES) GRÁFICO(S) 1608
MEMORIA CACHE 1604
FICHERO REGISTRO 1606
NÚCLEO(S) DE PROCESADOR 1607
CONJUNTO INSTRUCCIONES 1609
PROCESADOR(ES) 1602
BUS PROCESADOR
1610
PROCESADOR GRÁFICOS EXTERNO 1612
CONCENTRADOR CONTROLADOR MEMORIA 1616
DISPOSITIVO MEMORIA 1620
INSTRUCCIONES - 1621
DATOS - 1622
DISPOSITIVO ALMACENAMIENTO DATOS 1624
TRANSCEPTOR INALÁMBRICO 1626
INTERFAZ FIRMWARE 1628
CONCENTRADOR CONTROLADOR E/S 1630
CONTROLADOR E/S HEREDADO 1640
CONTROLADOR(ES) USB 1642
TECLADO/RATÓN 1644
CONTROLADOR AUDIO 1646
CONTROLADOR RED 1634
1600

FIG. 16

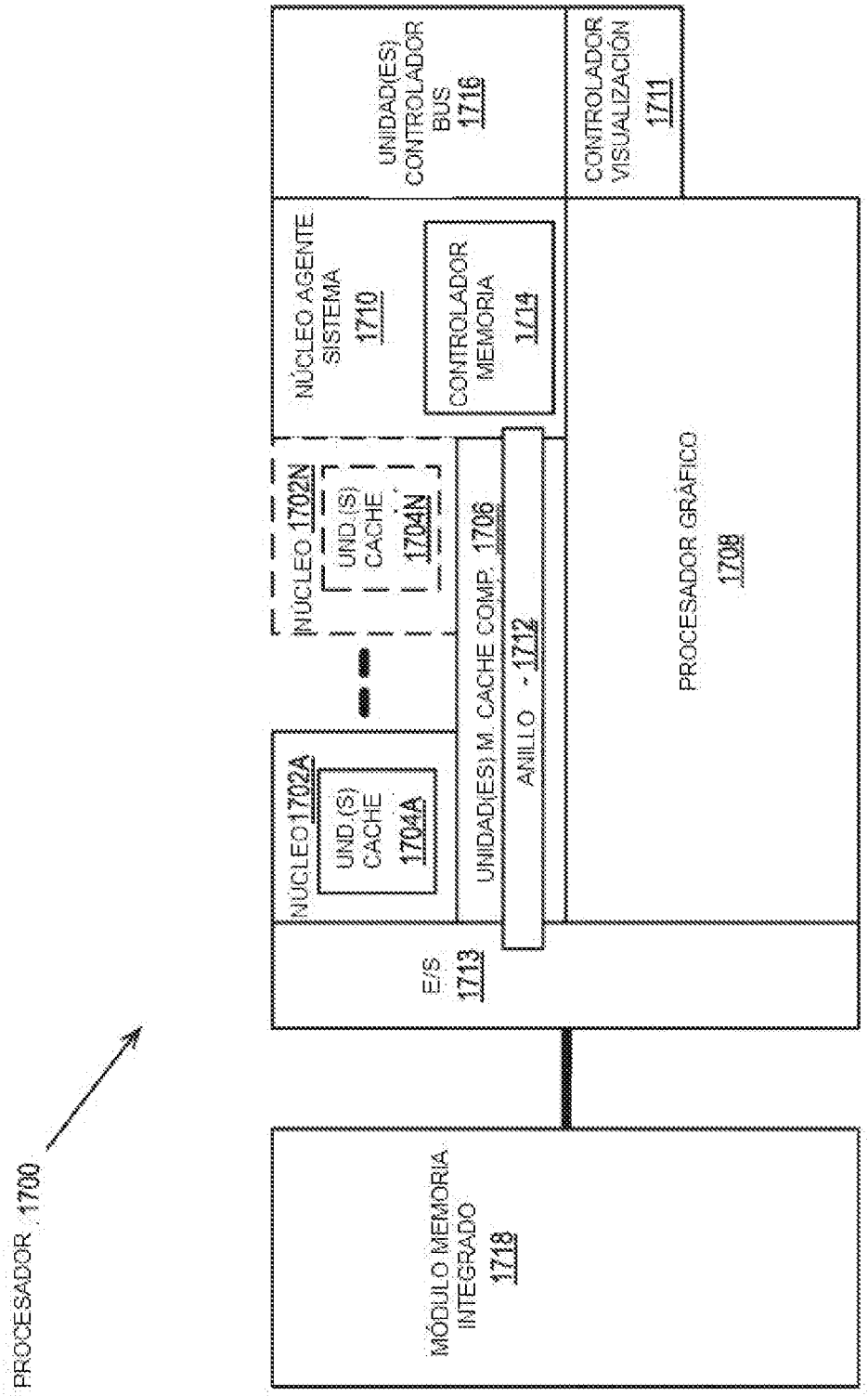

PROCESADOR 1700
MÓDULO MEMORIA INTEGRADO 1718
E/S 1713
NÚCLEO 1702A
UND.(S) CACHE 1704A
NÚCLEO 1702N
UND.(S) CACHE 1704N
UNIDAD(ES) M. CACHE COMP. 1706
ANILLO - 1712
NÚCLEO AGENTE SISTEMA 1710
CONTROLADOR MEMORIA 1714
UNIDAD(ES) CONTROLADOR BUS 1716
PROCESADOR GRÁFICO 1708
CONTROLADOR VISUALIZACIÓN 1711

FIG. 17

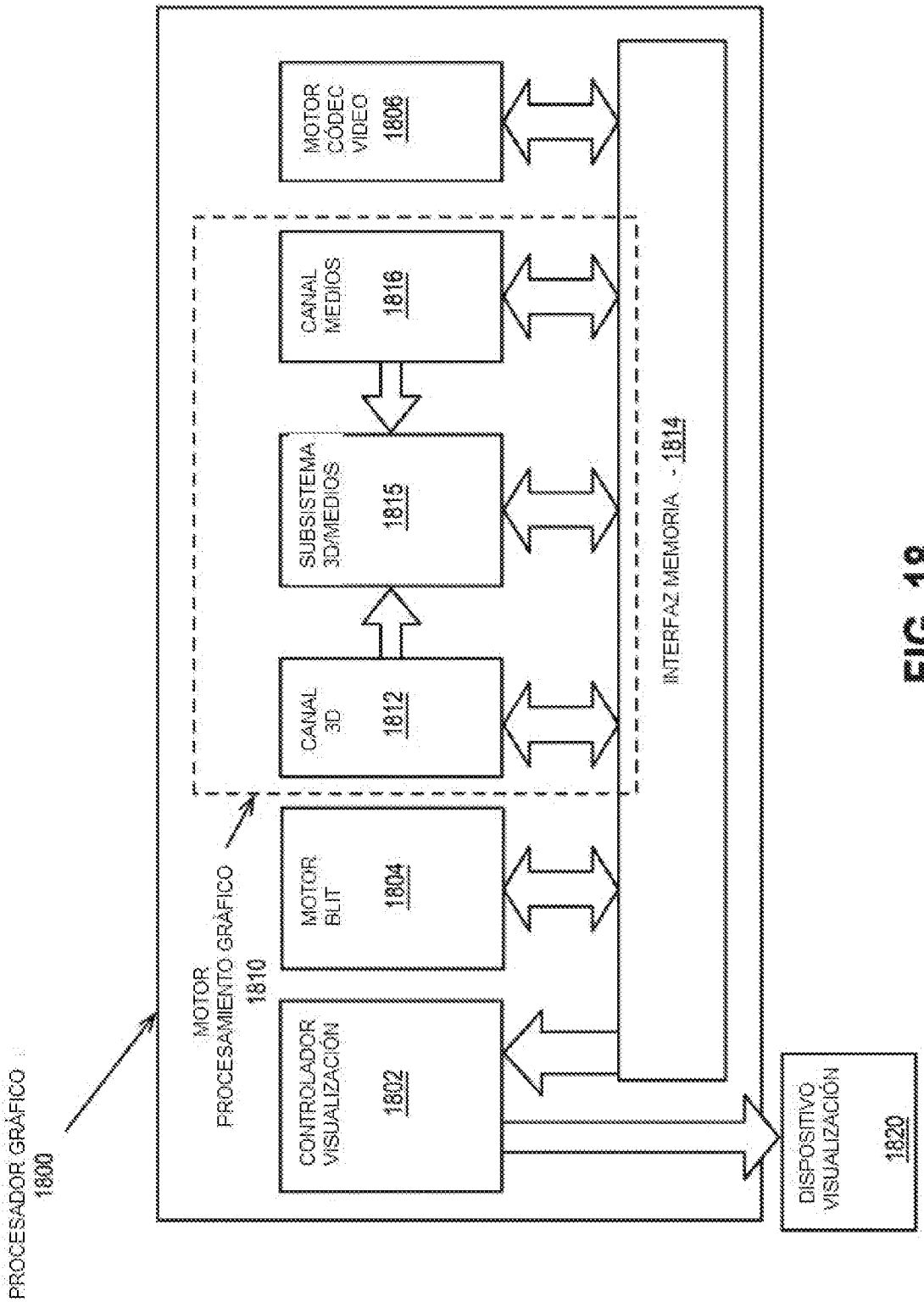

PROCESADOR GRÁFICO
1800
MOTOR
PROCESAMIENTO GRÁFICO
1810
CONTROLADOR
VISUALIZACIÓN
1812
MOTOR
BLIT
1814
CANAL
3D
1812
SUBSISTEMA
3D/MEDIOS
1815
CANAL
MEDIOS
1816
MOTOR
CÓDEC
VIDEO
1806
INTERFAZ MEMORIA - 1814
DISPOSITIVO
VISUALIZACIÓN
1820

FIG. 18

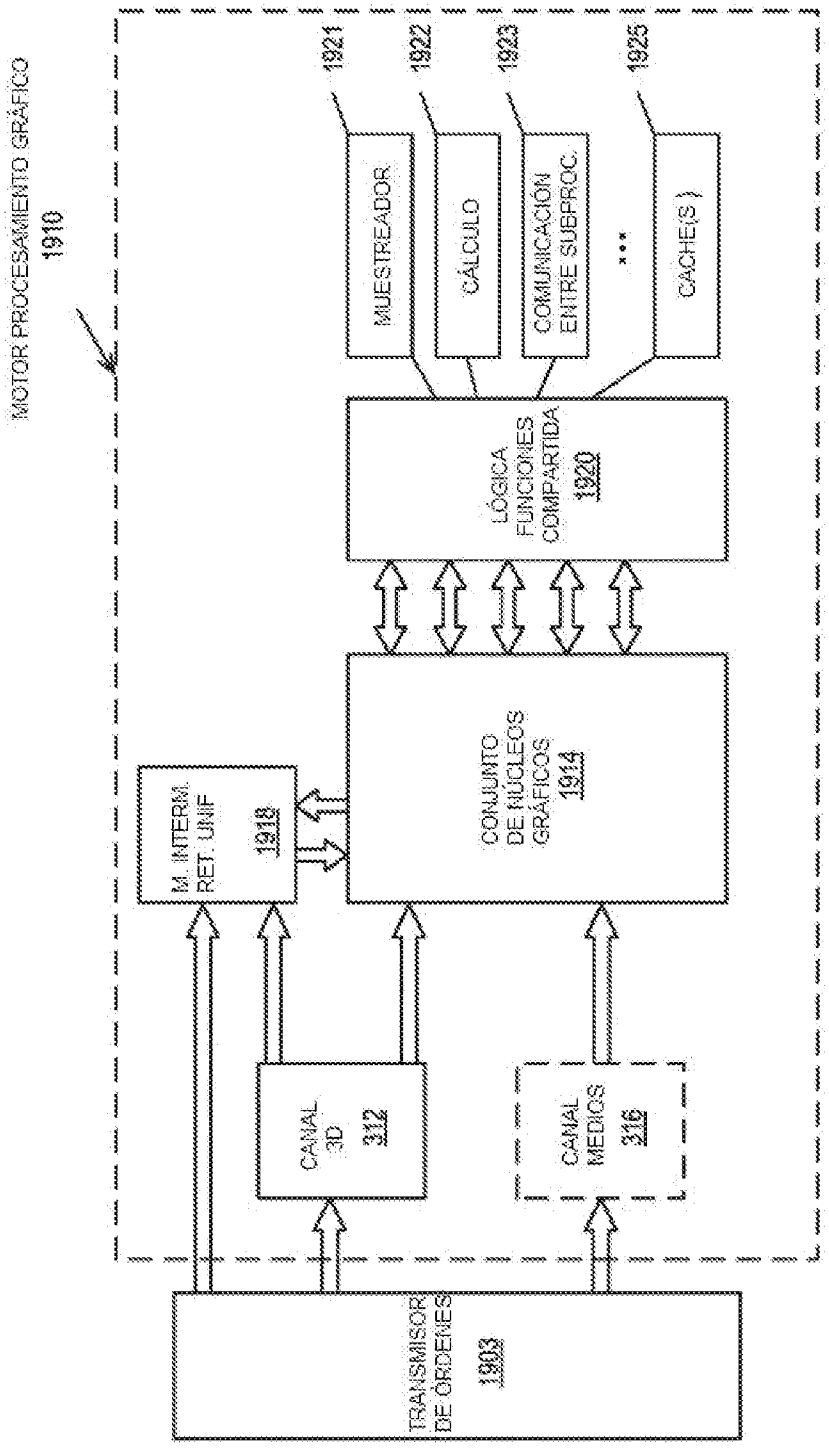

MOTOR PROCESAMIENTO GRÁFICO
1910
1921
1922
1923
1925
MUESTREADOR
CÁLCULO
COMUNICACIÓN ENTRE SUBPROC.
...
CACHE(S)
LÓGICA FUNCIONES COMPARTIDA
1920
CONJUNTO DE NÚCLEOS GRÁFICOS
1914
M. INTERM. RET. UNIF
1918
CANAL 3D
312
CANAL MEDIOS
316
TRANSMISOR DE ÓRDENES
1903

FIG. 19

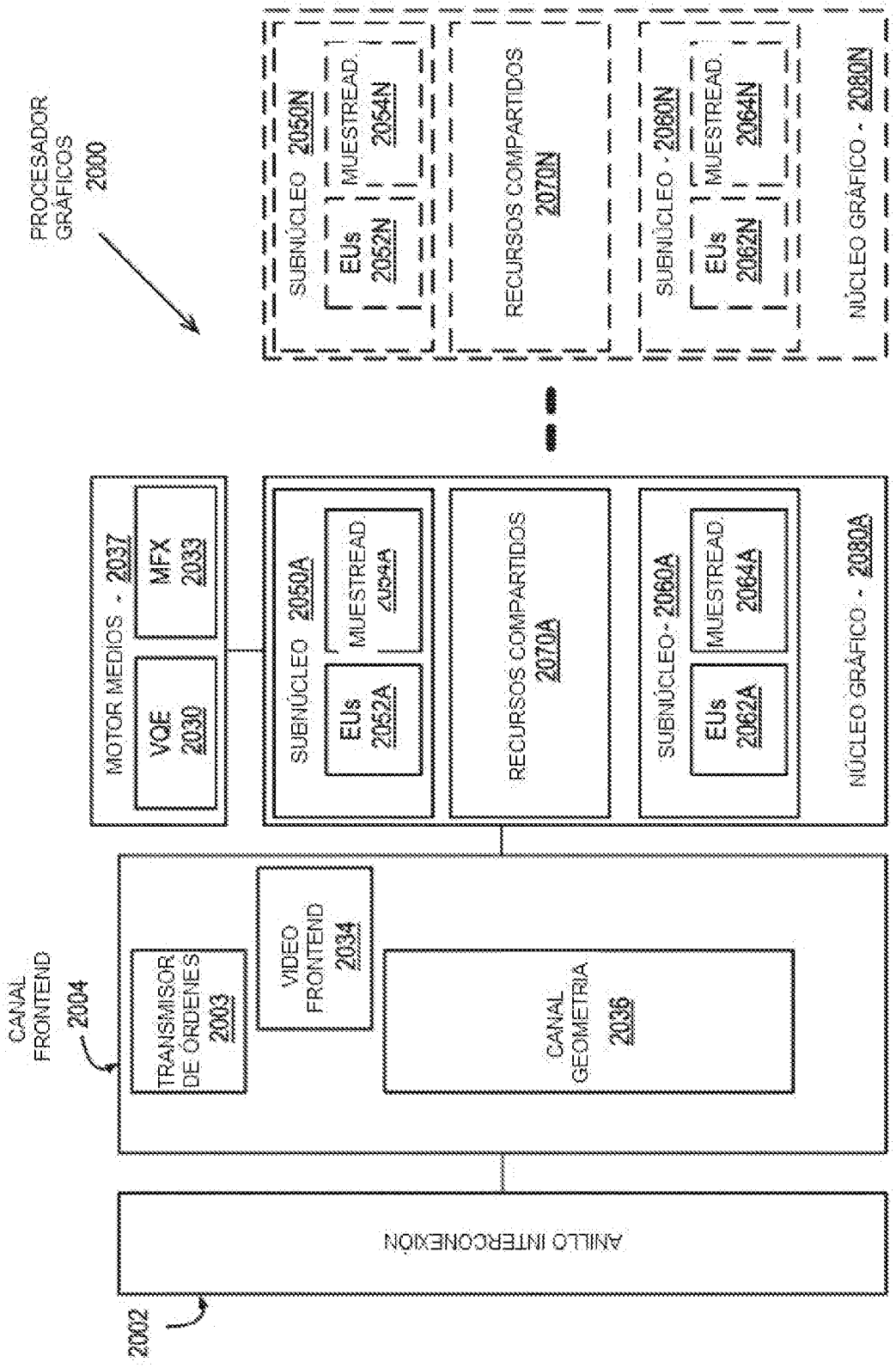

PROCESADOR GRÁFICOS 2000
ANILLO INTERCONEXIÓN 2002
CANAL FRONTEND 2004
TRANSMISOR DE ÓRDENES 2003
VIDEO FRONTEND 2034
CANAL GEOMETRÍA 2036
MOTOR MEDIOS - 2037
VQE 2030
MFX 2033
SUBNÚCLEO 2050A
EUs 2052A
MUESTREAD. 2054A
RECURSOS COMPARTIDOS 2070A
SUBNÚCLEO- 2060A
EUs 2062A
MUESTREAD. 2064A
NÚCLEO GRÁFICO - 2080A
SUBNÚCLEO 2050N
EUs 2052N
MUESTREAD. 2054N
RECURSOS COMPARTIDOS 2070N
SUBNÚCLEO - 2060N
EUs 2062N
MUESTREAD. 2064N
NÚCLEO GRÁFICO - 2080N

FIG. 20

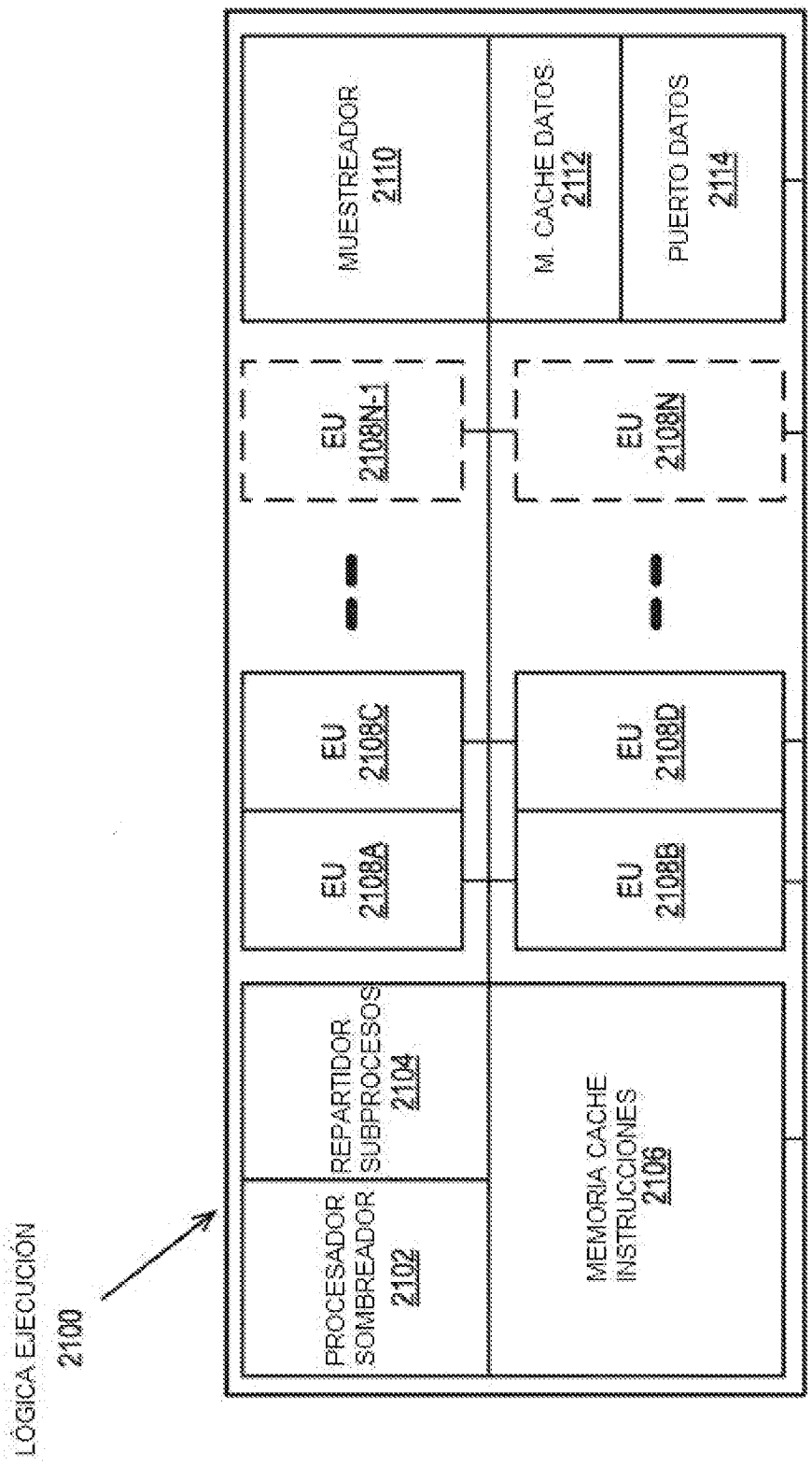

LÓGICA EJECUCIÓN
2100
PROCESADOR SOMBREADOR
2102
REPARTIDOR SUBPROCESOS
2104
MEMORIA CACHE INSTRUCCIONES
2106
EU 2108A
EU 2108B
EU 2108C
EU 2108D
EU 2108N-1
EU 2108N
MUESTREADOR
2110
M. CACHE DATOS
2112
PUERTO DATOS
2114

FIG. 21

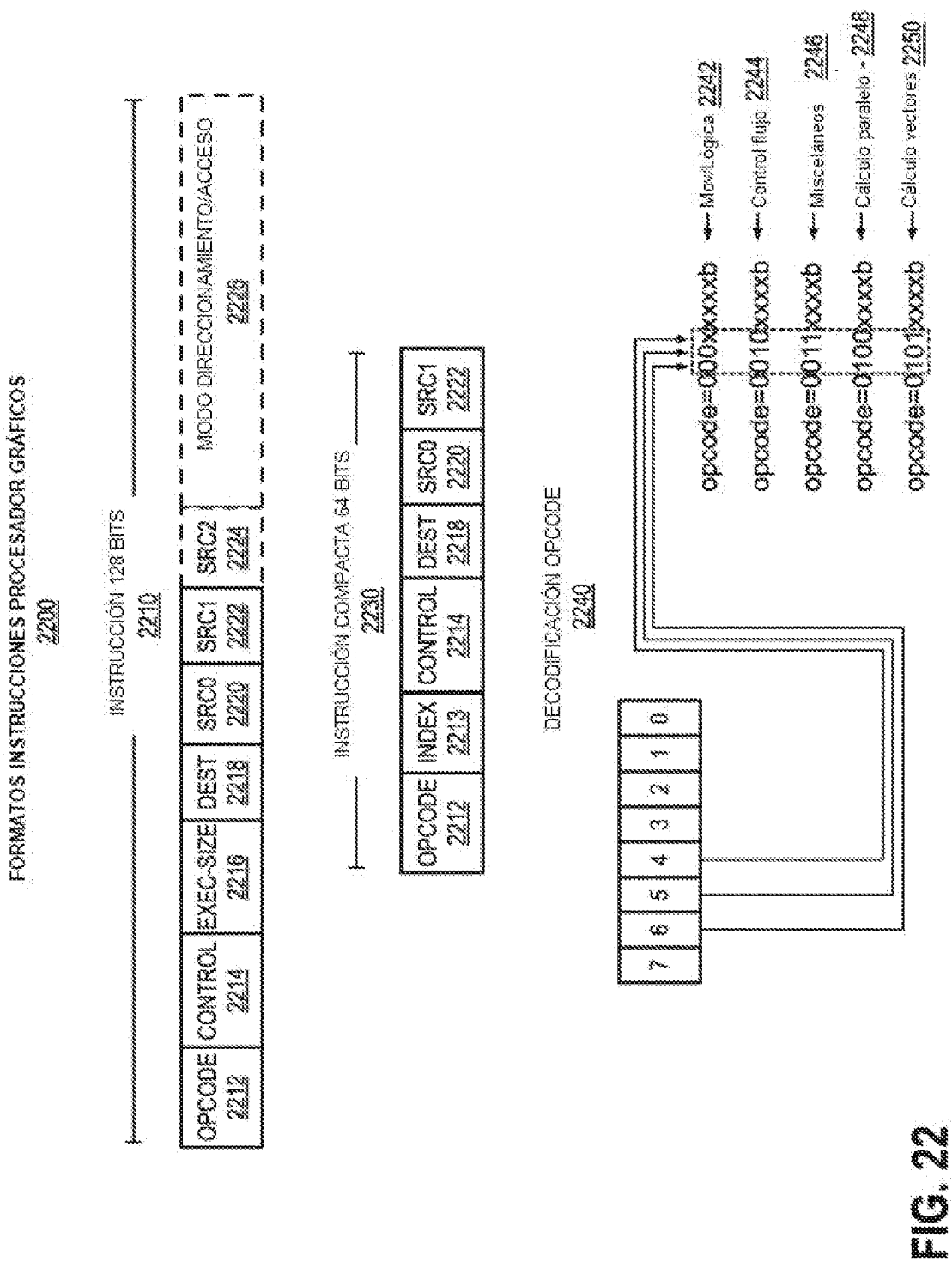

FORMATOS INSTRUCCIONES PROCESADOR GRÁFICOS
2200
INSTRUCCIÓN 128 BITS
2210
OPCODE 2212
CONTROL 2214
EXEC-SIZE 2216
DEST 2218
SRC0 2220
SRC1 2222
SRC2 2224
MODO DIRECCIONAMIENTO/ACCESO 2226
INSTRUCCIÓN COMPACTA 64 BITS
2230
OPCODE 2212
INDEX 2213
CONTROL 2214
DEST 2218
SRC0 2220
SRC1 2222
DECODIFICACIÓN OPCODE
2240
7 6 5 4 3 2 1 0
opcode=000xxxxxb
opcode=001xxxxxb
opcode=010xxxxxb
opcode=011xxxxxb
opcode=100xxxxxb
opcode=101xxxxxb
Mov/Lógica 2242
Control flujo 2244
Misceláneos 2246
Cálculo paralelo - 2248
Cálculo vectores 2250

FIG. 22

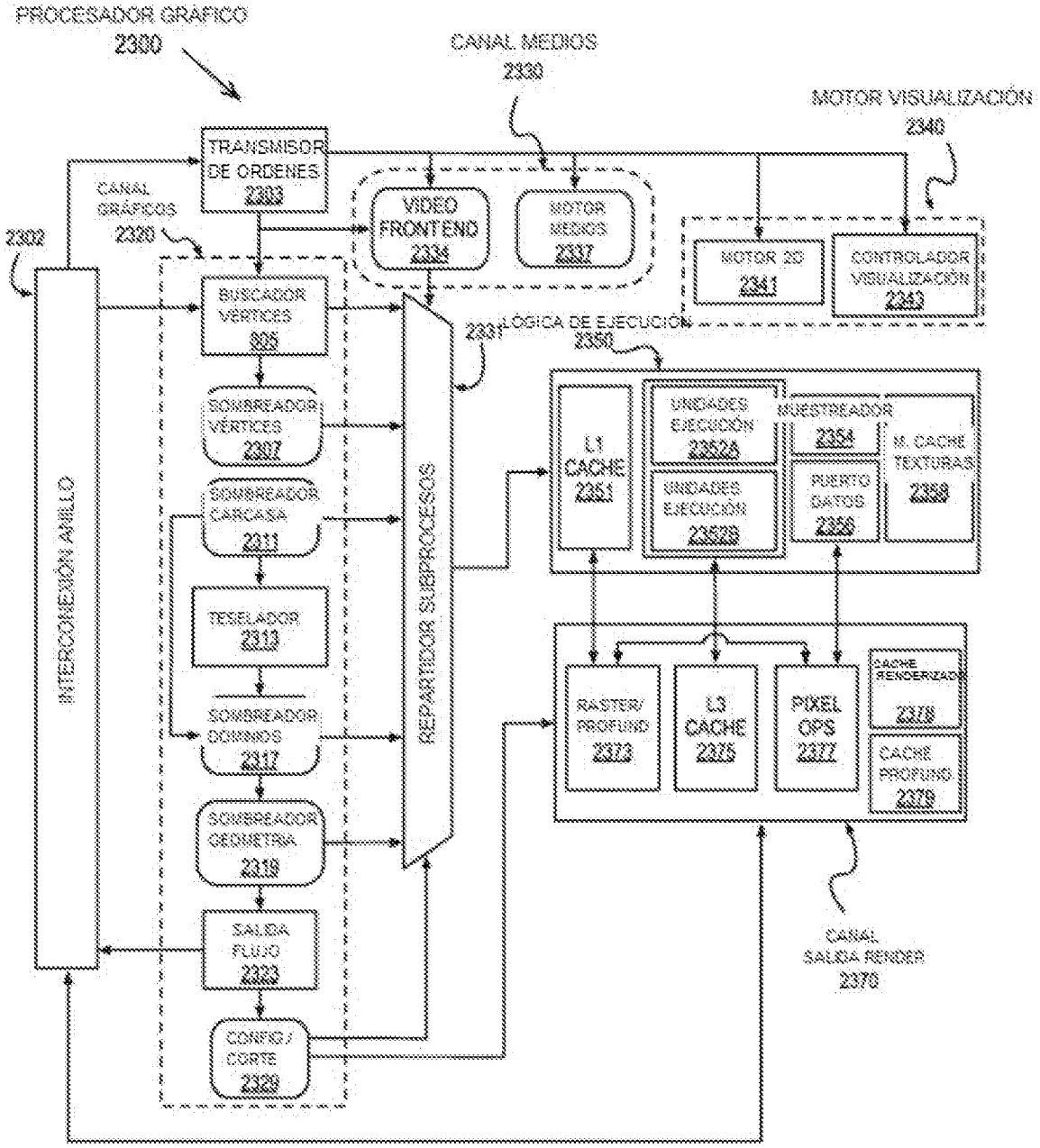

PROCESADOR GRÁFICO
2300
CANAL MEDIOS
2330
MOTOR VISUALIZACIÓN
2340
TRANSMISOR DE ÓRDENES
2303
CANAL GRÁFICOS
2320
2302
VIDEO FRONTEND
2334
MOTOR MEDIOS
2337
MOTOR 2D
2341
CONTROLADOR VISUALIZACIÓN
2343
BUSCADOR VÉRTICES
305
LÓGICA DE EJECUCIÓN
2331
2350
SOMBREADOR VÉRTICES
2307
SOMBREADOR CARCASA
2311
TESELADOR
2313
SOMBREADOR DOMINIO
2317
SOMBREADOR GEOMETRÍA
2319
SALIDA FLUJO
2323
CONFIG./ CORTE
2329
INTERCONEXIÓN ANILLO
REPARTIDOR SUBPROCESOS
L1 CACHE
2351
UNIDADES EJECUCIÓN
23528
UNIDADES EJECUCIÓN
23528
MUESTREADOR
2354
M. CACHE TEXTURAS
2358
PUERTO DATOS
2356
RASTER/ PROFUND
2373
L3 CACHE
2375
PIXEL OPS
2377
CACHE RENDERIZADO
2378
CACHE PROFUND
2379
CANAL SALIDA RENDER
2370

FIG. 23

FIG. 24A FORMATO ÓRDEN PROCESADOR DE GRÁFICOS 2400

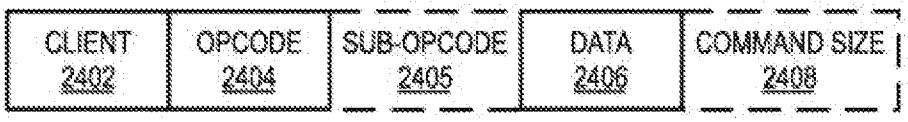
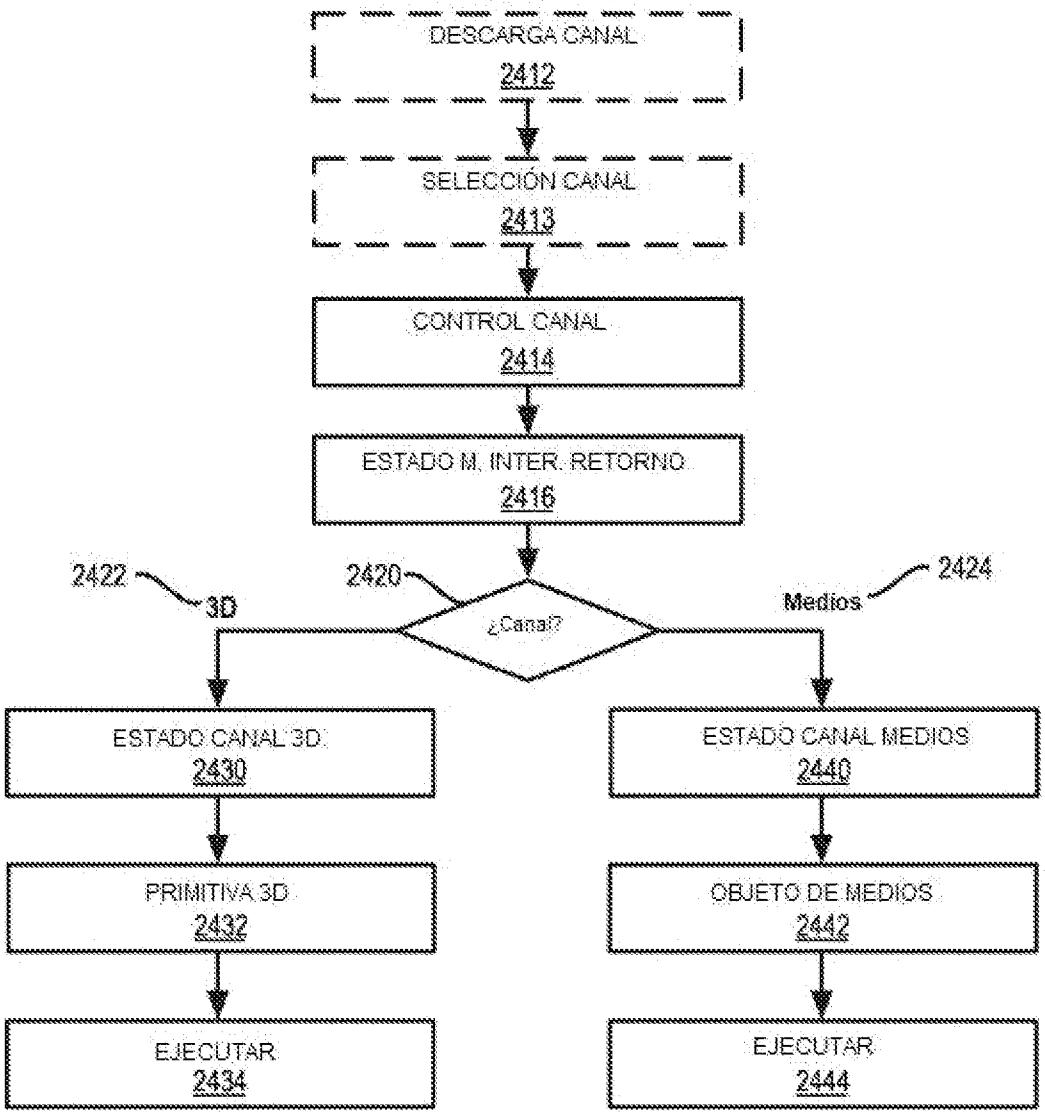


FIG. 24B SECUENCIA DE ÓRDENES DEL PROCESADOR DE GRÁFICOS 2410



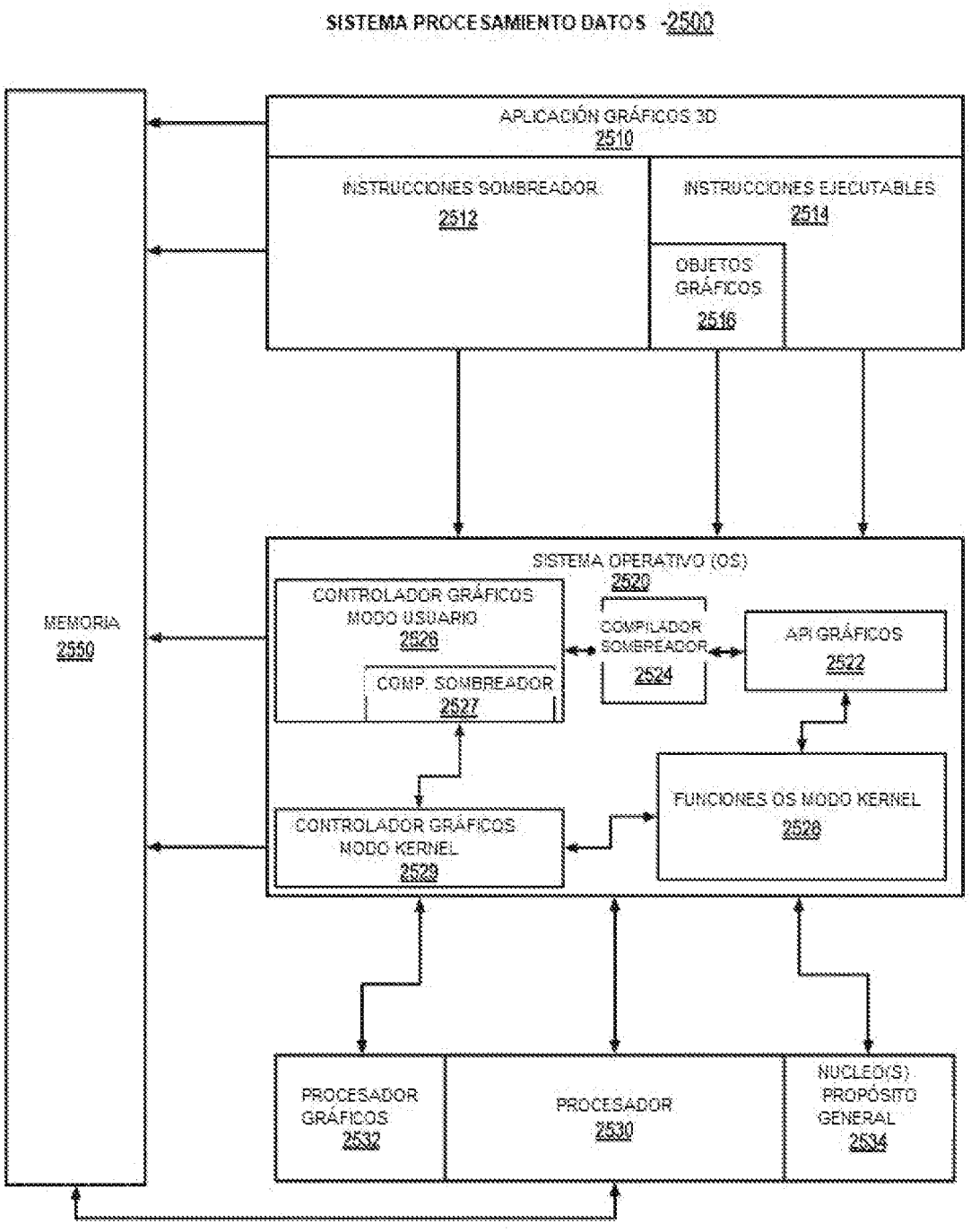

SISTEMA PROCESAMIENTO DATOS -2500
APLICACIÓN GRÁFICOS 3D
2510
INSTRUCCIONES SOMBREADOR
2512
INSTRUCCIONES EJECUTABLES
2514
OBJETOS GRÁFICOS
2516
MEMORIA
2550
SISTEMA OPERATIVO (OS)
2520
CONTROLADOR GRÁFICOS MODO USUARIO
2526
COMP. SOMBREADOR
2527
COMPILADOR SOMBREADOR
2524
API GRÁFICOS
2522
CONTROLADOR GRÁFICOS MODO KERNEL
2528
FUNCIONES OS MODO KERNEL
2529
PROCESADOR GRÁFICOS
2532
PROCESADOR
2530
NÚCLEO(S) PROPÓSITO GENERAL
2534

FIG. 25

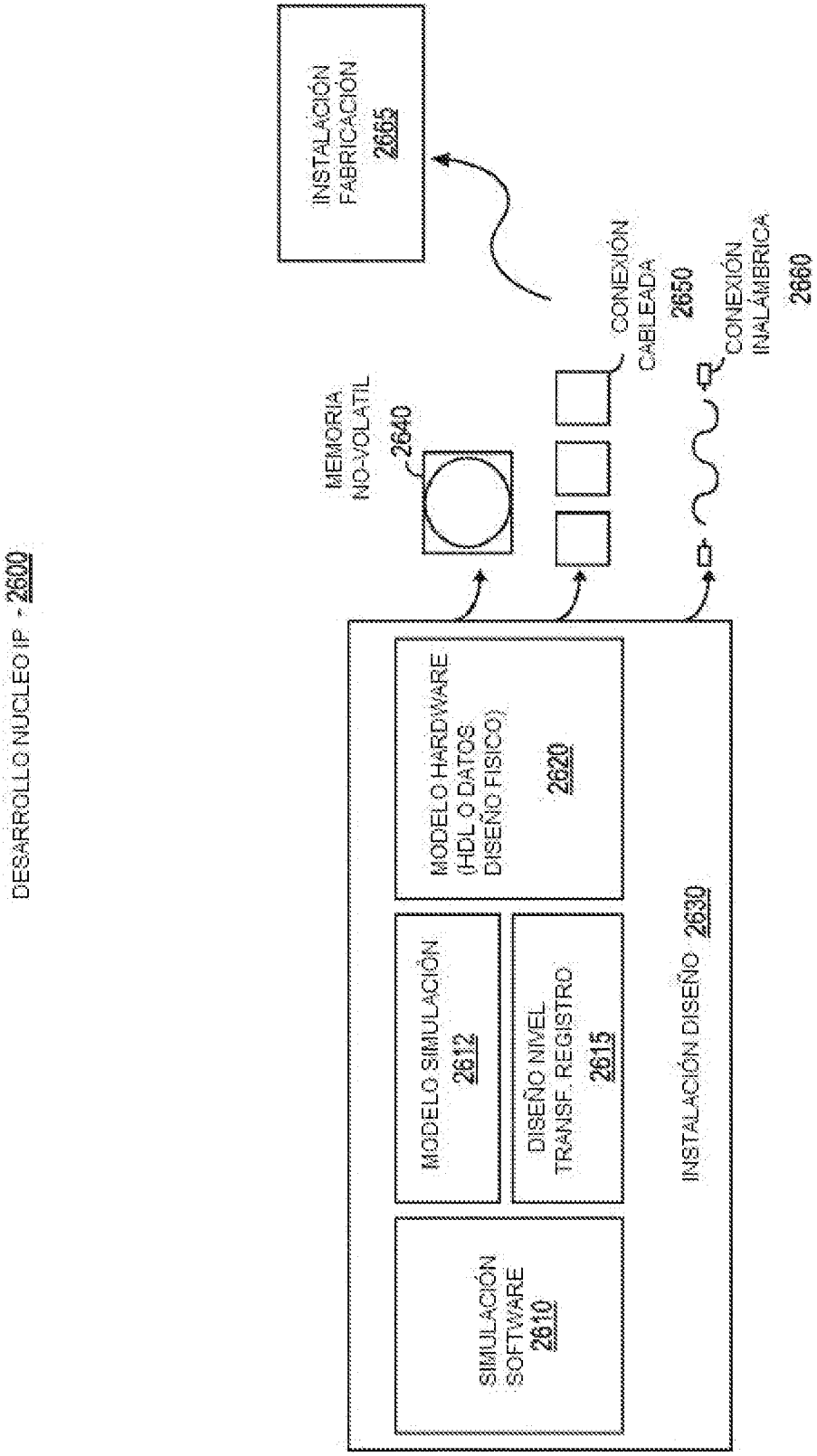

DESARROLLO NUCLEO IP - 2600
INSTALACIÓN DISEÑO 2630
SIMULACIÓN SOFTWARE 2610
MODELO SIMULACIÓN 2612
DISEÑO NIVEL TRANSF. REGISTRO 2615
MODELO HARDWARE (HDL O DATOS DISEÑO FISICO) 2620
MEMORIA NO-VOLATIL 2640
CONEXIÓN CABLEADA 2650
CONEXIÓN INALÁMBRICA 2660
INSTALACIÓN FABRICACIÓN 2665

FIG. 26

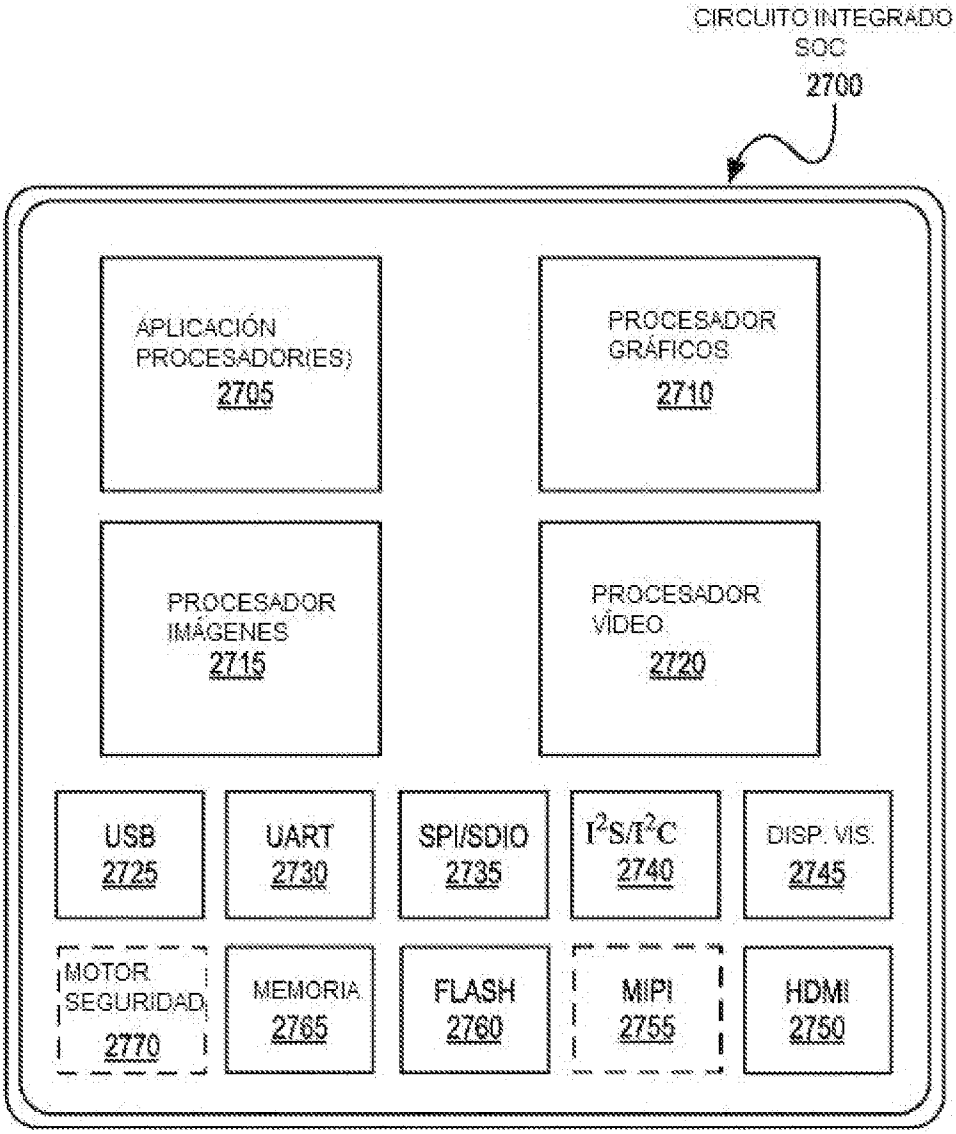

CIRCUITO INTEGRADO SOC
2700
APLICACIÓN PROCESADOR(ES)
2705
PROCESADOR GRÁFICOS
2710
PROCESADOR IMÁGENES
2715
PROCESADOR VÍDEO
2720
USB
2725
UART
2730
SPI/SDIO
2735
I2S/I2C
2740
DISP. VIS.
2745
MOTOR SEGURIDAD
2770
MEMORIA
2765
FLASH
2760
MIPI
2755
HDMI
2750

FIG. 27

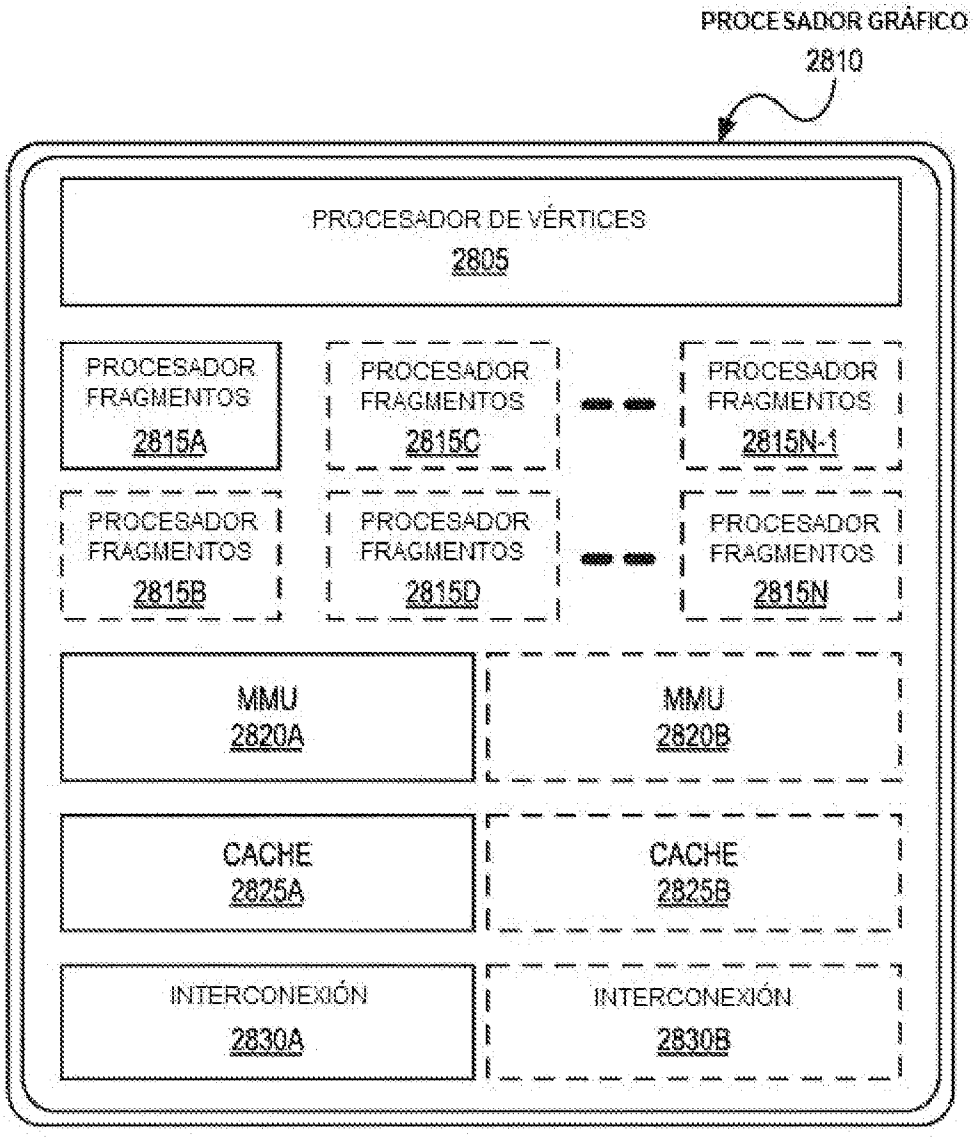

PROCESADOR GRÁFICO
2810
PROCESADOR DE VÉRTICES
2805
PROCESADOR FRAGMENTOS
2815A
PROCESADOR FRAGMENTOS
2815C
PROCESADOR FRAGMENTOS
2815N-1
PROCESADOR FRAGMENTOS
2815B
PROCESADOR FRAGMENTOS
2815D
PROCESADOR FRAGMENTOS
2815N
MMU
2820A
MMU
2820B
CACHE
2825A
CACHE
2825B
INTERCONEXIÓN
2830A
INTERCONEXIÓN
2830B

FIG. 28

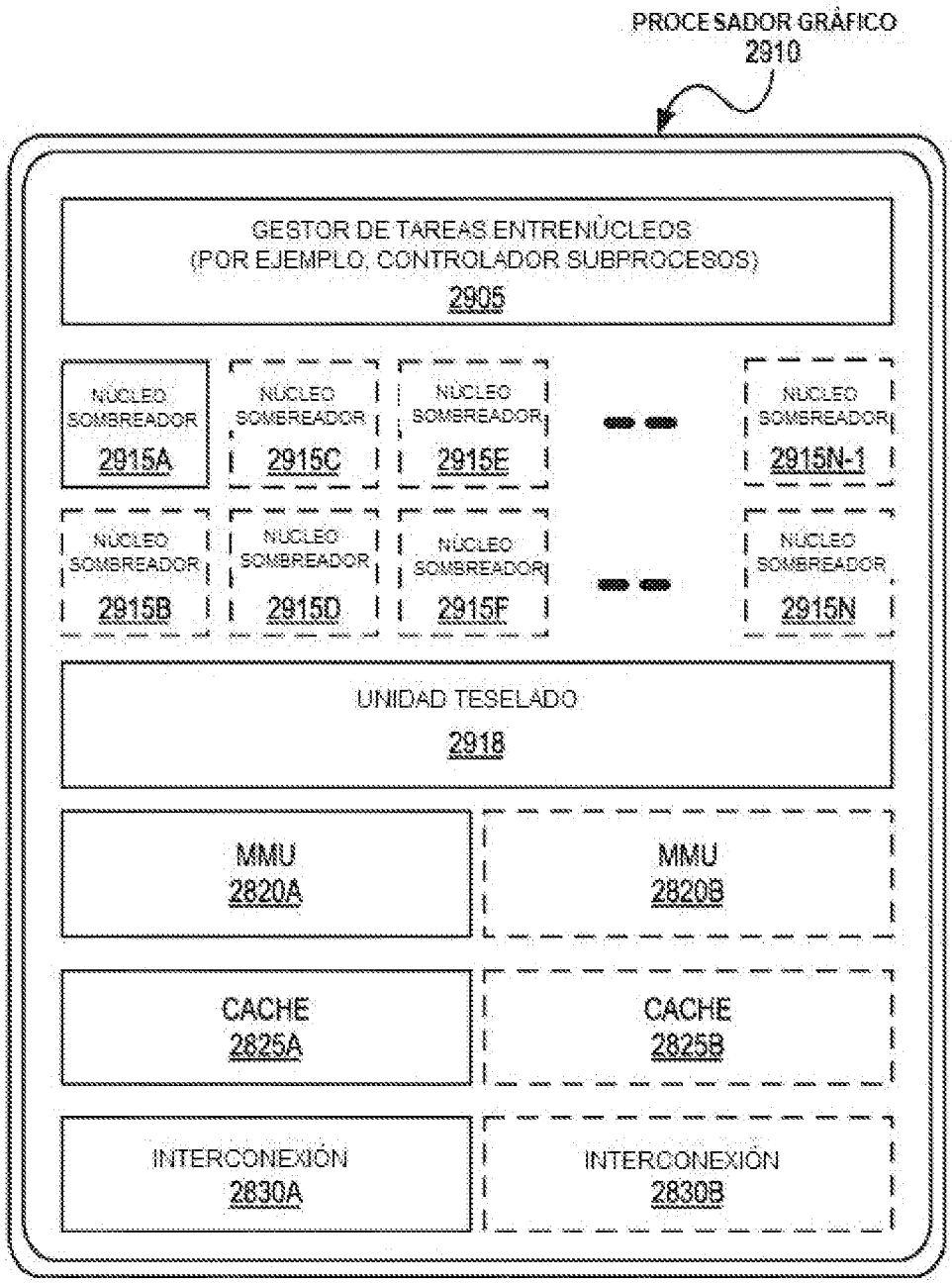

PROCESADOR GRÁFICO
2910
GESTOR DE TAREAS ENTRENÚCLEOS
(POR EJEMPLO, CONTROLADOR SUBPROCESOS)
2905
NÚCLEO SOMBREADOR
2915A
NÚCLEO SOMBREADOR
2915C
NÚCLEO SOMBREADOR
2915E
NÚCLEO SOMBREADOR
2915N-1
NÚCLEO SOMBREADOR
2915B
NÚCLEO SOMBREADOR
2915D
NÚCLEO SOMBREADOR
2915F
NÚCLEO SOMBREADOR
2915N
UNIDAD TESELADO
2918
MMU
2820A
MMU
2820B
CACHE
2825A
CACHE
2825B
INTERCONEXIÓN
2830A
INTERCONEXIÓN
2830B

FIG. 29