(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **3 005 227**

(51) Int. Cl.:

G06T 15/06 (2011.01)
G06T 15/50 (2011.01)
G06N 3/08 (2013.01)
G06N 3/045 (2013.01)
G06T 5/60 (2014.01)
G06T 5/70 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **26.08.2019 E 19193591 (5)**

(97) Fecha y número de publicación de la concesión europea: **09.10.2024 EP 3618007**

(54) Título: **Aparato y método para el entrenamiento en tiempo de ejecución de un motor de aprendizaje automático con eliminación de ruido**

(30) Prioridad:

28.08.2018 US 201816114537

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
14.03.2025

(73) Titular/es:

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

(72) Inventor/es:

WALD, INGO;
BENTHIN, CARSTEN y
AFRA, ATTILA TAMAS

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Aparato y método para el entrenamiento en tiempo de ejecución de un motor de aprendizaje automático con eliminación de ruido

ANTECEDENTES

Campo de la invención

Esta invención se refiere en general con el sector de los procesadores de gráficos. Más particularmente, la invención se refiere a un aparato y método para realizar operaciones de eliminación de ruido (por ejemplo, en un sistema de trazado de rayos) utilizando técnicas de aprendizaje automático.

Descripción de la técnica relacionada

El trazado de rayos es una técnica en la que se simula un transporte de luz a través de un renderizado basado en la física. Ampliamente utilizado en el renderizado cinemático, hasta hace unos pocos años se consideraba que consumía demasiados recursos para el rendimiento en tiempo real. Una de las operaciones clave en el trazado de rayos es procesar una consulta de visibilidad para las intersecciones de rayos y escenas conocida como "cruce de rayos", que calcula las intersecciones de rayos y escenas al atravesar e intersecar nodos en una jerarquía de volumen límite (BVH).

La publicación de Steve Bako et al. titulada "Kernel-predicting convolutional networks for denoising Monte Carlo renders", publicada en ACM transactions on graphics, ACM, vol. 36, n.º 4, 20 de julio de 2017, páginas 1-14, XP058372871, presenta un enfoque de aprendizaje profundo para eliminar el ruido de las imágenes renderizadas con Monte Carlo que produce resultados de alta calidad adecuados para la producción. El enfoque consiste en entrenar una red neuronal convolucional para aprender la compleja relación entre datos ruidosos y de referencia en un gran conjunto de fotogramas con diferentes efectos distribuidos de una película. La red entrenada se puede luego aplicar para eliminar el ruido de nuevas imágenes de otras películas con estilos y contenidos significativamente diferentes, con resultados de calidad de producción.

En la publicación de Christian Ledig et al. titulada "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", publicada en la Conferencia IEEE de 2017 sobre Visión por Computadora y Reconocimiento de Patrones (CVPR), 21-07-2017 (2017-07-21), XP055383327, se describe una red generativa adversaria (GAN) para superresolución de imágenes (SR) en la que se utiliza una asignación compleja entre información de imágenes de baja resolución (LR) y alta resolución (HR) para el entrenamiento para relacionar pares de ejemplos en parches de entrenamiento LR para los que se conocen las contrapartes HR correspondientes.

Sumario

La presente invención está definida por las reivindicaciones independientes. Las realizaciones ventajosas se describen mediante las reivindicaciones dependientes.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Una mejor comprensión de la presente invención se puede obtener a partir de la siguiente descripción detallada junto con los siguientes dibujos. Las realizaciones y ejemplos de las figuras 1-15 no corresponden a la invención reivindicada y se presentan únicamente con fines ilustrativos.

La **FIG. 1** es un diagrama de bloques de una realización de un sistema informático con un procesador que tiene uno o más núcleos de procesador y procesadores de gráficos;

la **FIG. 2** es un diagrama de bloques de una realización de un procesador que tiene uno o más núcleos de procesador, un controlador de memoria integrado y un procesador de gráficos integrado;

la **FIG. 3** es un diagrama de bloques de una realización de un procesador de gráficos que puede ser una unidad de procesamiento de gráficos discreta o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento;

la **FIG. 4** es un diagrama de bloques de una realización de un motor de procesamiento de gráficos para un procesador de gráficos;

la **FIG. 5** es un diagrama de bloques de otra realización de un procesador de gráficos;

las **FIGS. 6A-B** ilustran ejemplos de circuitos de ejecución y lógica;

la **FIG. 7** ilustra un formato de instrucción de unidad de ejecución de procesador de gráficos de acuerdo con un ejemplo;

la **FIG. 8** es un diagrama de bloques de otro ejemplo de un procesador de gráficos que incluye una tubería de gráficos, una tubería multimedia, un motor de visualización, una lógica de ejecución de subprocesos y una tubería de salida de renderizado;

la **FIG. 9A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos de acuerdo con un ejemplo;

la **FIG. 9B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos de acuerdo con un ejemplo;

la **FIG. 10** ilustra una arquitectura de software de gráficos ejemplar para un sistema de procesamiento de datos de acuerdo con un ejemplo;

las **FIGS. 11A-B** ilustran un sistema de desarrollo de núcleo IP ejemplar que puede usarse para fabricar un circuito integrado y un conjunto de paquete ejemplar;

la **FIG. 12** ilustra un sistema ejemplar en un circuito integrado de chip que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con un ejemplo;

las FIGS. 13A-B ilustran arquitecturas de procesadores de gráficos ejemplares en las que se pueden implementar ejemplos;

las **FIG. 14A-B** ilustran arquitecturas de procesadores de gráficos ejemplares;

la **FIG. 15** ilustra un ejemplo de una arquitectura para realizar el entrenamiento inicial de una arquitectura de aprendizaje automático;

la **FIG. 16** ilustra un ejemplo en el que un motor de aprendizaje automático se entrena y actualiza continuamente durante el tiempo de ejecución;

la **FIG. 17** ilustra otro ejemplo en el que un motor de aprendizaje automático se entrena y actualiza continuamente durante el tiempo de ejecución;

las **FIGS. 18A-B** ilustran ejemplos en los que se comparten datos de aprendizaje automático en una red; y

la **FIG. 19** ilustra una realización de un método para entrenar un motor de aprendizaje automático.

DESCRIPCIÓN DETALLADA

En la siguiente descripción, a efectos explicativos, se describen numerosos detalles específicos con el fin de proporcionar una comprensión completa de las realizaciones de la invención descritas a continuación. Sin embargo, será evidente para una persona experta en la materia que las realizaciones de la invención pueden practicarse sin algunos de estos detalles específicos. En otros casos, estructuras y dispositivos bien conocidos se muestran en forma de diagrama de bloques para evitar oscurecer los principios subyacentes de las realizaciones de la invención.

ARQUITECTURAS DE PROCESADORES DE GRÁFICOS Y TIPOS DE DATOS DE EJEMPLO

Descripción general del sistema

La **Figura 1** es un diagrama de bloques de un sistema de procesamiento 100, de acuerdo con una realización. En diversas realizaciones, el sistema 100 incluye uno o más procesadores 102 y uno o más procesadores de gráficos 108, y es un sistema de escritorio de un solo procesador, un sistema de estación de trabajo multiprocesador o un sistema de servidor que tiene una gran cantidad de procesadores 102 o núcleos de procesador 107. En una realización, el sistema 100 es una plataforma de procesamiento incorporada dentro de un circuito integrado de sistema en un chip (SoC) para su uso en dispositivos móviles, portátiles o integrados.

El sistema 100 puede incluir, o estar incorporado dentro de una plataforma de juego basada en servidor, una consola de juegos, incluyendo una consola de juegos y medios, una consola de juegos móvil, una consola de juegos portátil o una consola de juegos en línea. El sistema 100 es un teléfono móvil, un teléfono inteligente, un dispositivo informático de tableta o un dispositivo de Internet móvil. El sistema de procesamiento 100 también

puede incluir, acoplarse o estar integrado dentro de un dispositivo ponible, tal como un dispositivo ponible de tipo reloj inteligente, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. El sistema de procesamiento 100 es un dispositivo de televisión o decodificador que tiene uno o más procesadores 102 y una interfaz gráfica generada por uno o más procesadores de gráficos 108.

En algunas realizaciones, los uno o más procesadores 102 incluyen en cada caso uno o más núcleos de procesador 107 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para el software de sistema y de usuario. En algunas realizaciones, cada uno de los uno o más núcleos de procesador 107 está configurado para procesar un conjunto de instrucciones 109 específico. El conjunto de instrucciones 109 puede facilitar la computación del conjunto de instrucciones complejo (CISC), la computación del conjunto de instrucciones reducido (RISC) o la computación mediante una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 107 pueden procesar cada uno un conjunto de instrucciones 109 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo del procesador 107 también puede incluir otros dispositivos de procesamiento, tal como un procesador de señal digital (DSP).

El procesador 102 incluye memoria caché 104. Dependiendo de la arquitectura, el procesador 102 puede tener un solo caché interna o múltiples niveles de caché interna. La memoria caché se comparte entre varios componentes del procesador 102. El procesador 102 también utiliza una caché externa (por ejemplo, una caché de nivel 3 (L3) o una caché de último nivel (LLC)) (no mostrada), que puede compartirse entre los núcleos del procesador 107 utilizando técnicas de coherencia de caché conocidas. Además, en el procesador 102 se incluye un archivo de registro 106 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de punto flotante, registros de estado y un registro de puntero de instrucción). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos del diseño del procesador 102.

En algunas realizaciones, uno o más procesadores 102 están acoplados a uno o más buses de interfaz 110 para transmitir señales de comunicación tales como direcciones, datos o señales de control entre el procesador 102 y otros componentes en el sistema 100. El bus de interfaz 110 puede ser un bus de procesador, tal como una versión del bus de interfaz multimedia directa (DMI). Sin embargo, los buses de procesador no se limitan al bus DMI, y pueden incluir uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express), buses de memoria u otros tipos de buses de interfaz. El o los procesadores 102 incluyen un controlador de memoria integrado 116 y un concentrador controlador de plataforma 130. El controlador de memoria 116 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 100, mientras que el concentrador controlador de plataforma (PCH) 130 proporciona conexiones a dispositivos de E/S a través de un bus de E/S local.

El dispositivo de memoria 120 puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria de acceso aleatorio estática (SRAM), un dispositivo de memoria flash, un dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para servir como memoria de proceso. El dispositivo de memoria 120 puede funcionar como memoria de sistema para el sistema 100, para almacenar datos 122 e instrucciones 121 para su uso cuando uno o más procesadores 102 ejecutan una aplicación o proceso. El controlador de memoria 116 también se acopla con un procesador de gráficos externo 112 opcional, que puede comunicarse con uno o más procesadores de gráficos 108 en los procesadores 102 para realizar operaciones gráficas y multimedia. Un dispositivo de visualización 111 puede conectarse al o a los procesadores 102. El dispositivo de visualización 111 puede ser uno o más de un dispositivo de visualización interno, tal como un dispositivo electrónico móvil o un dispositivo portátil o un dispositivo de visualización externo conectado a través de una interfaz de visualización (por ejemplo, DisplayPort, etc.). El dispositivo de visualización 111 puede ser una pantalla montada en la cabeza (HMD), tal como un dispositivo de visualización estereoscópico para su uso en aplicaciones de realidad virtual (RV) o aplicaciones de realidad aumentada (RA).

El concentrador de controlador de plataforma 130 permite que los periféricos se conecten al dispositivo de memoria 120 y al procesador 102 a través de un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, entre otros, un controlador de audio 146, un controlador de red 134, una interfaz de firmware 128, un transceptor inalámbrico 126, sensores táctiles 125, un dispositivo de almacenamiento de datos 124 (por ejemplo, unidad de disco duro, memoria flash, etc.). El dispositivo de almacenamiento de datos 124 puede conectarse mediante una interfaz de almacenamiento (por ejemplo, SATA) o mediante un bus de periféricos, tal como un bus de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). Los sensores táctiles 125 pueden incluir sensores de pantalla táctil, sensores de presión o sensores de huellas digitales. El transceptor inalámbrico 126 puede ser un transceptor de Wi-Fi, un transceptor de Bluetooth o un transceptor de red móvil tal como un transceptor de 3G, 4G o evolución a largo plazo (LTE). La interfaz de firmware 128 posibilita la comunicación con firmware de sistema y puede ser, por ejemplo, una interfaz de firmware ampliable unificada (UEFI). El controlador de red 134 puede posibilitar una conexión de red a una red cableada. Un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de interfaz 110. El controlador de audio 146 es un controlador de audio de alta definición multicanal. El sistema 100 incluye un controlador de E/S heredado 140 opcional para acoplar dispositivos heredados (por ejemplo, Sistema Personal 2 (PS/2)) al sistema. El concentrador controlador de plataforma 130 también puede conectarse a uno o más controladores de bus serie universal (USB) 142 para conectar dispositivos

de entrada, tales como combinaciones de teclado y ratón 143, una cámara 144 u otros dispositivos de entrada USB.

Se apreciará que el sistema 100 mostrado es ejemplar y no limitativo, ya que también se pueden utilizar otros tipos de sistemas de procesamiento de datos que estén configurados de manera diferente. Por ejemplo, una instancia del controlador de memoria 116 y del concentrador de controlador de plataforma 130 puede integrarse en un procesador de gráficos externo discreto, tal como el procesador de gráficos externo 112. El concentrador de controlador de plataforma 130 y/o el controlador de memoria 1160 pueden ser externos al/a los uno o más procesador(es) 102. Por ejemplo, el sistema 100 puede incluir un controlador de memoria externo 116 y un concentrador de controlador de plataforma 130, que puede configurarse como un concentrador de controlador de memoria y un concentrador de controlador de periféricos dentro de un conjunto de chips de sistema que está en comunicación con el/los procesador(es) 102.

La **Figura 2** es un diagrama de bloques de una realización de un procesador 200 que tiene uno o más núcleos de procesador 202A-202N, un controlador de memoria integrado 214 y un procesador de gráficos integrado 208. Aquellos elementos de la Figura 2 que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otras partes en el presente documento, pero no se limitan a los mismas. El procesador 200 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 202N representado por los recuadros en línea discontinua. Cada uno de los núcleos de procesador 202A-202N incluye una o más unidades de memoria caché internas 204A-204N. Cada núcleo de procesador también tiene acceso a una o más unidades de caché 206 compartidas.

Las unidades de memoria caché internas 204A-204N y las unidades de memoria caché compartidas 206 representan una jerarquía de memoria caché dentro del procesador 200. 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. La lógica de coherencia de caché mantiene la coherencia entre las distintas unidades de caché 206 y 204A-204N.

El procesador 200 también puede incluir un conjunto de una o más unidades de controlador de bus 216 y un núcleo de agente de sistema 210. Las una o más unidades de controlador de bus 216 gestionan un conjunto de buses periféricos, tal como uno o más buses PCI o PCI express. El núcleo de agente de sistema 210 proporciona funcionalidad de gestión para los distintos componentes del procesador. El núcleo de agente de sistema 210 incluye uno o más controladores de memoria integrados 214 para administrar el acceso a varios dispositivos de memoria externa (no mostrados).

Uno o más de los núcleos de procesador 202A-202N incluyen soporte para multitarea con hilos simultánea. El núcleo de agente de sistema 210 incluye componentes para coordinar y operar los núcleos 202A-202N durante el procesamiento multitarea con hilos. El núcleo de agente de sistema 210 puede incluir adicionalmente una unidad de control de potencia (PCU), que incluye lógica y componentes para regular el estado de potencia de los núcleos de procesador 202A-202N y el procesador de gráficos 208.

En algunas realizaciones, el procesador 200 incluye además un procesador de gráficos 208 para ejecutar operaciones de procesamiento de gráficos. El procesador de gráficos 208 se acopla con el conjunto de unidades de caché 206 compartidas y el núcleo de agente de sistema 210, incluidos uno o más controladores de memoria integrados 214. El núcleo de agente de sistema 210 también incluye un controlador de visualización 211 para controlar la salida del procesador de gráficos a una o más pantallas acopladas. El controlador de visualización 211 también puede ser un módulo separado acoplado al procesador de gráficos a través de al menos una interconexión, o puede estar integrado dentro del procesador de gráficos 208.

Se utiliza una unidad de interconexión basada en anillo 212 para acoplar los componentes internos del procesador 200. Sin embargo, se puede utilizar una unidad de interconexión alternativa, tal como una interconexión punto a punto, una interconexión conmutada u otras técnicas, incluidas técnicas bien conocidas en la técnica. El procesador de gráficos 208 se acopla con la interconexión en anillo 212 a través de un enlace de E/S 213.

El enlace de E/S 213 ejemplar representa al menos una de múltiples variedades de interconexiones de E/S, incluida una interconexión de E/S en paquete que facilita la comunicación entre varios componentes del procesador y un módulo de memoria integrado de alto rendimiento 218, tal como un módulo eDRAM. Cada uno de los núcleos de procesador 202A-202N y el procesador de gráficos 208 utilizan módulos de memoria integrados 218 como una caché de último nivel compartida.

Los núcleos de procesador 202A-202N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. Los núcleos de procesador 202A-202N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 202A-202N 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. Los núcleos de procesador 202A-202N son heterogéneos en términos de microarquitectura, donde uno o más núcleos que tienen un consumo de energía relativamente mayor se acoplan con uno o más núcleos de energía que tienen un consumo de energía menor. Además, el procesador 200 se puede implementar en uno o más chips o como un circuito integrado de SoC que tenga los componentes ilustrados, además de otros componentes.

La **Figura 3** es un diagrama de bloques de un procesador de gráficos 300, que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento. El procesador de gráficos se comunica a través de una interfaz de E/S asignada en memoria con los registros del procesador de gráficos y con comandos colocados en la memoria de procesador. El procesador de gráficos 300 incluye una interfaz de memoria 314 para acceder a la memoria. La interfaz de memoria 314 puede ser una interfaz a la memoria local, a una o más cachés internas, a una o más cachés externas compartidas y/o a la memoria de sistema.

El procesador de gráficos 300 también incluye un controlador de visualización 302 para impulsar datos de salida de visualización a un dispositivo de visualización 320. El controlador de visualización 302 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. El dispositivo de visualización 320 puede ser un dispositivo de visualización interno o externo. El dispositivo de visualización 320 es un dispositivo de visualización montado en la cabeza, tal como un dispositivo de visualización de realidad virtual (VR) o un dispositivo de visualización de realidad aumentada (AR). El procesador de gráficos 300 incluye un motor de códec de vídeo 306 para codificar, decodificar o transcodificar medios hacia, desde o entre uno o más formatos de codificación multimedia, incluidos, entre otros, los formatos Moving Picture Experts Group (MPEG) como MPEG-2, los formatos Advanced Vídeo Coding (AVC) como H.264/MPEG-4 AVC, así como los formatos 421M/VC-1 de la Society of Motion Picture & Television Engineers (SMPTE) y del Joint Photographic Experts Group (JPEG) tal como JPEG y Motion JPEG (MJPEG).

El procesador de gráficos 300 incluye un motor de transferencia de imágenes en bloques (BLIT) 304 para realizar operaciones de rasterización bidimensionales (2D), incluidas, por ejemplo, transferencias de bloques de límite de bits. Sin embargo, las operaciones de gráficos 2D se realizan utilizando uno o más componentes del motor de procesamiento de gráficos (GPE) 310. El GPE 310 es un motor de cálculo para realizar operaciones gráficas, incluidas operaciones de gráficos tridimensionales (3D) y operaciones multimedia.

El GPE 310 incluye una tubería 3D 312 para realizar operaciones 3D, como renderizar imágenes y escenas tridimensionales utilizando funciones de procesamiento que actúan sobre formas primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La tubería 3D 312 incluye elementos de función fija y programables que realizan varias tareas dentro del elemento y/o generan hilos de ejecución hacia un subsistema 3D/multimedia 315. Aunque puede usarse la tubería 3D 312 para realizar operaciones multimedia, un ejemplo del GPE 310 también incluye una tubería multimedia 316 que se usa específicamente para realizar operaciones multimedia, tales como post procesamiento de vídeo y mejora de imagen.

La tubería multimedia 316 incluye funciones fijas o unidades lógicas programables para realizar una o más operaciones multimedia especializadas, tales como aceleración de decodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar de, o en nombre del motor de códec de vídeo 306. La tubería multimedia 316 incluye además una unidad de generación de hilos para generar hilos para su ejecución en el subsistema 3D/multimedia 315. Los hilos generados realizan cálculos para las operaciones multimedia en una o más unidades de ejecución de gráficos incluidas en el subsistema 3D/multimedia 315.

El subsistema 3D/multimedia 315 incluye lógica para ejecutar hilos generados por la tubería 3D 312 y la tubería multimedia 316. Las tuberías envían solicitudes de ejecución de hilos al subsistema 3D/multimedia 315, que incluye lógica de despacho de hilos para arbitrar y despachar las diversas solicitudes a los recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una serie de unidades de ejecución de gráficos para procesar los hilos multimedia y 3D. El subsistema 3D/multimedia 315 incluye una o más cachés internas para instrucciones y datos de hilos. El subsistema también incluye memoria compartida, incluidos registros y memoria direccionable, para compartir datos entre hilos y almacenar datos de salida.

Motor de procesamiento de gráficos

La **Figura 4** es un diagrama de bloques de un motor de procesamiento de gráficos 410 de un procesador de gráficos de acuerdo con algunas realizaciones. El motor de procesamiento de gráficos (GPE) 410 es una versión del GPE 310 que se muestra en la **Figura 3**. Los elementos de la **Figura 4** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento, pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella. Por ejemplo, se ilustran el conducto de 3D 312 y el conducto multimedia 316 de la Figura 3. La tubería multimedia 316 es opcional en algunos ejemplos del GPE 410 y puede no estar explícitamente incluida dentro del GPE 410. Por ejemplo, un procesador multimedia y/o de imágenes independiente está acoplado al GPE 410.

El GPE 410 se acopla con o incluye un transmisor de comandos 403, que proporciona un flujo de comandos a la tubería 3D 312 y/o a las tuberías multimedia 316. El transmisor de comandos 403 está acoplado a una memoria, que puede ser una memoria de sistema o una o más de las siguientes: memoria caché interna y memoria caché compartida. El transmisor de comandos 403 recibe comandos de la memoria y envía los comandos a la tubería 3D 312 y/o a la tubería multimedia 316. Los comandos son directivas extraídas de una memoria intermedia en anillo, que almacena comandos para la tubería 3D 312 y la tubería multimedia 316. La memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la tubería 3D 312 también pueden incluir referencias a datos almacenados en la memoria, tales como, entre otros, datos de vértice y geometría para la tubería 3D 312 y/o datos de imagen y objetos de memoria para la tubería multimedia 316. La tubería 3D 312 y la tubería multimedia 316 procesan los comandos y datos realizando operaciones a través de la lógica dentro de las tuberías respectivas o enviando uno o más hilos de ejecución a una matriz de núcleos de gráficos 414. La matriz de núcleos de gráficos 414 incluye uno o más bloques de núcleos de gráficos (por ejemplo, núcleos o núcleos DE gráficos 415A, núcleo o núcleos de gráficos 415B), incluyendo cada bloque uno o más núcleos de gráficos. Cada núcleo de gráficos incluye un conjunto de recursos de ejecución de gráficos que incluye lógica de ejecución de gráficos de propósito general y específica para realizar operaciones gráficas y computacionales, así como procesamiento de texturas de función fija y/o aprendizaje automático y lógica de aceleración de inteligencia artificial.

La tubería 3D 312 incluye una función fija y una lógica programable para procesar uno o más programas de sombreado, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cómputo u otros programas de sombreado, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleos de gráficos 414. La matriz de núcleos de gráficos 414 proporciona un bloque unificado de recursos de ejecución para su uso en el procesamiento de estos programas de sombreado. La lógica de ejecución de múltiples propósitos (por ejemplo, unidades de ejecución) dentro del núcleo o núcleos de gráficos 415A-414B de la matriz de núcleos de gráficos 414 incluye el soporte para diversos lenguajes de sombreador de API 3D y puede ejecutar múltiples subprocesos de ejecución simultáneos asociados con múltiples sombreadores.

La matriz de núcleos de gráficos 414 también incluye lógica de ejecución para realizar funciones multimedia, tal como procesamiento de vídeo y/o imágenes. Las unidades de ejecución incluyen además lógica de propósito general que se puede programar para realizar operaciones computacionales paralelas de propósito general, además de operaciones de procesamiento de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o en conjunto con la lógica de propósito general dentro del núcleo o núcleos del procesador 107 de la Figura 1 o el núcleo 202A-202N como en la Figura 2.

Los datos de salida generados por los subprocesos que se ejecutan en la matriz de núcleos de gráficos 414 pueden generar datos en la memoria en una memoria intermedia de retorno unificada (URB) 418. La URB 418 puede almacenar datos para múltiples subprocesos. La URB 418 puede usarse para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleos de gráficos 414. La URB 418 también se puede utilizar para la sincronización entre hilos en la matriz del núcleo de gráficos y la lógica de función fija dentro de la lógica de función compartida 420.

La matriz de núcleos de gráficos 414 es escalable, de modo que la matriz incluye una cantidad variable de núcleos de gráficos, cada uno con una cantidad variable de unidades de ejecución según el nivel de potencia y rendimiento objetivo del GPE 410. Los recursos de ejecución son escalables dinámicamente, de modo que pueden habilitarse o deshabilitarse según sea necesario.

La matriz de núcleos de gráficos 414 se acopla con la lógica de función compartida 420 que incluye múltiples recursos que se comparten entre los núcleos de gráficos en la matriz de núcleos de gráficos. Las funciones compartidas dentro de la lógica de función compartida 420 son unidades lógicas de hardware que proporcionan una funcionalidad complementaria especializada a la matriz de núcleos de gráficos 414. La lógica de función compartida 420 incluye, entre otras cosas, el muestreador 421, la matemática 422 y la lógica de comunicación entre hilos (ITC) 423. Adicionalmente, algunos ejemplos implementan una o más caché o cachés 425 dentro de la lógica de función compartida 420.

Se implementa una función compartida donde la demanda para una función especializada dada es insuficiente para la inclusión dentro de la matriz de núcleos de gráficos 414. En su lugar, se implementa una única instanciación de esa función especializada como una entidad autónoma en la lógica de función compartida 420 y se comparte entre los recursos de ejecución dentro de la matriz de núcleos de gráficos 414. El conjunto preciso de funciones que se comparten entre la matriz de núcleos de gráficos 414 y que se incluyen dentro de la matriz de núcleos de gráficos 414 varía. Funciones compartidas específicas dentro de la lógica de funciones compartidas 420 que son usadas ampliamente por la matriz de núcleos de gráficos 414 pueden incluirse dentro de la lógica de funciones compartidas 416 dentro de la matriz de núcleos de gráficos 414. La lógica de función compartida 416 dentro de la matriz de núcleos de gráficos 414 puede incluir parte o toda la lógica dentro de la lógica de función compartida 420. Todos los elementos lógicos dentro de la lógica de funciones compartidas 420 pueden duplicarse dentro de

la lógica de funciones compartidas 416 de la matriz de núcleos de gráficos 414. La lógica de función compartida 420 se excluye en favor de la lógica de función compartida 416 dentro de la matriz de núcleos de gráficos 414.

La **Figura 5** es un diagrama de bloques de la lógica de hardware de un núcleo de procesador de gráficos 500, de acuerdo con algunas realizaciones descritas en este documento. Aquellos elementos de la **Figura 5** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto. El núcleo del procesador de gráficos 500 ilustrado está incluido dentro de la matriz de núcleos de gráficos 414 de la **Figura 4**. El núcleo de procesador de gráficos 500, a veces denominado segmento de núcleo, puede ser uno o varios núcleos de gráficos dentro de un procesador de gráficos modular. El núcleo de procesador de gráficos 500 es un ejemplo de un segmento de núcleo de gráficos, y un procesador de gráficos como se describe en el presente documento puede incluir múltiples segmentos de núcleo de gráficos basadas en envolventes de potencia y rendimiento objetivo. Cada núcleo de procesador de gráficos 500 puede incluir un bloque de función fija 530 acoplado a múltiples subnúcleos 501A-501F, también denominados subsegmentos, que incluyen bloques modulares de lógica de función fija y de propósito general.

El bloque de función fija 530 incluye una tubería de función fija/geometría 536 que puede ser compartida por todos los subnúcleos en el núcleo de procesador de gráficos 500, por ejemplo, en implementaciones de procesadores de gráficos de menor rendimiento y/o menor consumo. La tubería de función fija/geométrica 536 incluye una tubería de función fija 3D (por ejemplo, la tubería 3D 312 como en la **Figura 3** y la **Figura 4**), una unidad de interfaz de vídeo, un generador de hilos y un despachador de hilos, y un gestor de memoria intermedia de retorno unificada, que gestiona memorias intermedias de retorno unificadas, tales como la memoria intermedia de retorno unificada 418 de la **Figura 4**.

El bloque de función fija 530 también incluye una interfaz de SoC de gráficos 537, un microcontrolador de gráficos 538 y una tubería multimedia 539. La interfaz de gráficos de SoC 537 proporciona una interfaz entre el núcleo de procesador de gráficos 500 y otros núcleos de procesador dentro de un circuito integrado de sistema en un chip. El microcontrolador de gráficos 538 es un subprocesador programable que se puede configurar para gestionar varias funciones del núcleo de procesador de gráficos 500, incluido el despacho de hilos, la programación y la preemisión. La tubería multimedia 539 (por ejemplo, la tubería multimedia 316 de la **Figura 3** y la **Figura 4**) incluye lógica para facilitar la decodificación, codificación, preprocesamiento y/o postprocesamiento de datos multimedia, incluidos datos de imagen y vídeo. La tubería multimedia 539 implementa operaciones multimedia a través de solicitudes para calcular o muestrear lógica dentro de los subnúcleos 501-501F.

La interfaz de SoC 537 permite que el núcleo de procesador de gráficos 500 se comunique con núcleos de procesador de aplicaciones de propósito general (por ejemplo, CPU) y/u otros componentes dentro de un SoC, incluidos elementos de la jerarquía de memoria tales como una memoria caché de último nivel compartida, la RAM del sistema y/o DRAM integrada en el chip o en el paquete. La interfaz de SoC 537 también puede permitir la comunicación con dispositivos de función fija dentro del SoC, tal como canales de imágenes de cámaras, y permite el uso y/o implementa átomos de memoria global que pueden compartirse entre el núcleo de procesador de gráficos 500 y las CPU dentro del SoC. La interfaz de SoC 537 también puede implementar controles de administración de energía para el núcleo de procesador de gráficos 500 y habilitar una interfaz entre un dominio de reloj del núcleo de gráficos 500 y otros dominios de reloj dentro del SoC. La interfaz de SoC 537 permite la recepción de memorias intermedias de comandos desde un transmisor de comandos y un despachador de hilos global que están configurados para proporcionar comandos e instrucciones a cada uno de uno o más núcleos de gráficos dentro de un procesador de gráficos. Los comandos e instrucciones se pueden enviar a la tubería multimedia 539, cuando se deben realizar operaciones multimedia, o a una tubería de geometría y función fija (por ejemplo, tubería de geometría y función fija 536, tubería de geometría y función fija 514) cuando se deben realizar operaciones de procesamiento de gráficos.

El microcontrolador gráfico 538 se puede configurar para realizar varias tareas de planificación y gestión para el núcleo de procesador de gráficos 500. El microcontrolador de gráficos 538 puede realizar la planificación de gráficos y/o carga de trabajo computacional en los diversos motores de gráficos paralelos dentro de las matrices de unidades de ejecución (EU) 502A-502F, 504A-504F dentro de los subnúcleos 501A-501F. En este modelo de planificación, el software anfitrión que se ejecuta en un núcleo de CPU de un SoC que incluye el núcleo de procesador de gráficos 500 puede enviar cargas de trabajo a uno de los múltiples timbres del procesador de gráficos, lo que invoca una operación de planificación en el motor gráfico apropiado. Las operaciones de planificación incluyen determinar qué carga de trabajo ejecutar a continuación, enviar una carga de trabajo a un transmisor de comandos, interrumpir las cargas de trabajo existentes que se ejecutan en un motor, monitorizar el progreso de una carga de trabajo y notificar al software anfitrión cuando se completa una carga de trabajo. El microcontrolador de gráficos 538 también puede facilitar estados de bajo consumo o inactivos para el núcleo de procesador de gráficos 500, lo que proporciona al núcleo de procesador de gráficos 500 la capacidad de guardar y restaurar registros dentro del núcleo de procesador de gráficos 500 en transiciones de estado de bajo consumo independientemente del sistema operativo y/o del software del controlador de gráficos en el sistema.

El núcleo de procesador de gráficos 500 puede tener más o menos de los subnúcleos ilustrados 501A-501F, hasta N subnúcleos modulares. Para cada conjunto de N subnúcleos, el núcleo de procesador de gráficos 500 también puede incluir lógica de función compartida 510, memoria compartida y/o caché 512, una tubería de función fija/geométrica 514, así como lógica de función fija adicional 516 para acelerar varias operaciones de procesamiento de gráficos y cálculo. La lógica de función compartida 510 puede incluir unidades lógicas asociadas con la lógica de función compartida 420 de la **Figura 4** (por ejemplo, lógica de comunicación entre subprocesos, matemáticas y/o de muestreo) que pueden ser compartidas por cada uno de los N subnúcleos dentro del núcleo de procesador de gráficos 500. La memoria compartida y/o caché 512 puede ser una memoria caché de último nivel para el conjunto de N subnúcleos 501A-501F dentro del núcleo de procesador de gráficos 500, y también puede servir como memoria compartida a la que pueden acceder múltiples subnúcleos. La tubería de geometría/función fija 514 se puede incluir en lugar de la tubería de geometría/función fija 536 dentro del bloque de función fija 530 y puede incluir las mismas unidades lógicas o similares.

El núcleo de procesador de gráficos 500 incluye lógica de función fija 516 adicional que puede incluir varias lógicas de aceleración de función fija para su uso por el núcleo de procesador de gráficos 500. La lógica de función fija 516 adicional incluye una tubería de geometría adicional para usar en sombreado de solo posición. En el sombreado de solo posición, existen dos tuberías de geometría: la tubería de geometría completa dentro de la tubería de función fija/geometría 516, 536, y una tubería de selección, que es una tubería de geometría adicional que puede incluirse dentro de la lógica de función fija 516 adicional. La tubería de selección es una versión reducida de la tubería de geometría completa. La tubería completa y la tubería de selección pueden ejecutar diferentes instancias de la misma aplicación, teniendo cada instancia un contexto separado. El sombreado de solo posición puede ocultar largas tiradas de triángulos descartados, lo que permite que el sombreado se complete antes en algunos casos. Por ejemplo, la lógica de tubería de selección dentro de la lógica de función fija 516 adicional puede ejecutar sombreadores de posición en paralelo con la aplicación principal y generalmente genera resultados críticos más rápido que la tubería completa, ya que la tubería de selección obtiene y sombrea solo el atributo de posición de los vértices, sin realizar la rasterización y la representación de los píxeles en la memoria intermedia de fotogramas. La tubería de selección puede usar los resultados críticos generados para calcular la información de visibilidad de todos los triángulos sin tener en cuenta si esos triángulos se eliminan o no. La tubería completa (que en este caso puede denominarse tubería de reproducción) puede consumir la información de visibilidad para omitir los triángulos descartados y sombrear solo los triángulos visibles que finalmente pasan a la fase de rasterización.

La lógica de función fija 516 adicional también puede incluir lógica de aceleración de aprendizaje automático, tal como lógica de multiplicación de matrices de función fija, para implementaciones que incluyen optimizaciones para el entrenamiento o la inferencia de aprendizaje automático.

Dentro de cada subnúcleo de gráficos 501A-501F se incluye un conjunto de recursos de ejecución que pueden usarse para realizar operaciones gráficas, multimedia y de cálculo en respuesta a solicitudes de la tubería de gráficos, la tubería multimedia o los programas de sombreado. Los subnúcleos de gráficos 501A-501F incluyen múltiples matrices de EU 502A-502F, 504A-504F, la lógica de despacho de subprocesos y comunicación entre subprocesos (TD/IC) 503A-503F, un muestreador 3D (por ejemplo, de textura) 505A-505F, un muestreador multimedia 506A-506F, un procesador de sombreador 507A-507F y una memoria local compartida (SLM) 508A-508F. Cada una de las matrices de EU 502A-502F, 504A-504F incluye múltiples unidades de ejecución, que son unidades de procesamiento de gráficos de propósito general que pueden realizar operaciones lógicas de coma flotante y de números enteros/coma fija al servicio de una operación de gráficos, multimedia o de cálculo, incluyendo programas de gráficos, multimedia o de sombreado de cálculo. La lógica de TD/IC 503A-503F realiza operaciones de control de subprocesos y de despacho de subprocesos locales para las unidades de ejecución dentro de un subnúcleo y facilita la comunicación entre subprocesos que se ejecutan en las unidades de ejecución del subnúcleo. El muestreador 3D 505A-505F puede leer texturas u otros datos relacionados con gráficos 3D en memoria. El muestreador 3D puede leer datos de textura de manera diferente basándose en el estado de muestra configurado y en el formato de textura asociado con una textura dada. El muestreador multimedia 506A-506F puede realizar operaciones de lectura similares basándose en el tipo y en el formato asociados con datos multimedia. Cada subnúcleo de gráficos 501A-501F puede incluir alternativamente un muestreador multimedia y 3D unificado. Los subprocesos que se ejecutan en las unidades de ejecución dentro de cada uno de los subnúcleos 501A-501F pueden hacer uso de la memoria local compartida 508A-508F dentro de cada subnúcleo, para posibilitar que los subprocesos que se ejecutan dentro de un grupo de subprocesos se ejecuten usando un agrupamiento común de memoria en chip.

Unidades de ejecución

Las Figuras 6A-6B ilustran la lógica de ejecución de subprocesos 600 que incluye una matriz de elementos de procesamiento empleados en un núcleo de procesador de gráficos según las realizaciones descritas en este documento. Los elementos de las Figuras 6A-6B que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura de este documento pueden operar o funcionar de cualquier manera similar a la descrita en otro lugar de este documento, pero no se limitan a ello. La **Figura 6A** ilustra una descripción general de la lógica de ejecución de hilos 600, que puede incluir una variante de la lógica de hardware ilustrada con cada subnúcleo 501A-501F de la FIG. 5. La **Figura** 6B ilustra detalles internos ejemplares de una unidad de ejecución.

Como se ilustra en la **Figura 6A**, la lógica de ejecución de hilos 600 incluye un procesador de sombreador 602, un despachador de hilos 604, una caché de instrucciones 606, una matriz de unidades de ejecución escalable que incluye una pluralidad de unidades de ejecución 608A-608N, un muestreador 610, una caché de datos 612 y un puerto de datos 614. La matriz de unidades de ejecución escalable puede escalar dinámicamente habilitando o deshabilitando una o más unidades de ejecución (por ejemplo, cualquiera de las unidades de ejecución 608A, 608B, 608C, 608D, hasta 608N-1 y 608N) según los requisitos computacionales de una carga de trabajo. Los componentes incluidos están interconectados a través de una red de interconexión que se enlaza cada uno de los componentes. La lógica de ejecución de hilos 600 incluye una o más conexiones a memoria, tal como la memoria de sistema o la memoria caché, a través de uno o más de los siguientes: caché de instrucciones 606, puerto de datos 614, muestreador 610 y unidades de ejecución 608A-608N. Cada unidad de ejecución (por ejemplo, 608A) es una unidad computacional de propósito general, programable e independiente, capaz de ejecutar múltiples hilos de hardware simultáneos mientras procesa múltiples elementos de datos en paralelo para cada hilo. La matriz de unidades de ejecución 608A-608N es escalable para incluir cualquier número de unidades de ejecución individuales.

Las unidades de ejecución 608A-608N se utilizan principalmente para ejecutar programas de sombreado. Un procesador de sombreador 602 puede procesar los diversos programas de sombreador y enviar hilos de ejecución asociados con los programas de sombreador a través de un despachador de hilos 604. El despachador de hilos incluye lógica para arbitrar solicitudes de inicio de hilos desde las tuberías de gráficos y multimedia e instanciar los hilos solicitados en una o más unidades de ejecución en las unidades de ejecución 608A-608N. Por ejemplo, una tubería de geometría puede enviar sombreadores de vértices, teselaciones o geometría a la lógica de ejecución de hilos para su procesamiento. El despachador de hilos 604 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas sombreadores en ejecución.

Las unidades de ejecución 608A-608N admiten un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreador de gráficos 3D estándar, de modo que los programas de sombreador de bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución admiten el procesamiento de vértices y geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y procesamiento de propósito general (por ejemplo, sombreadores multimedia y de cómputo). Cada una de las unidades de ejecución 608A-608N es capaz de múltiples emisiones de ejecución de múltiples datos, una instrucción (SIMD), y un funcionamiento de múltiples subprocesos habilita un entorno de ejecución efectivo frente a accesos de memoria de latencia superior. Cada subproceso de hardware dentro de cada unidad de ejecución tiene un archivo de registro de ancho de banda alto dedicado y un estado de subproceso independiente asociado. La ejecución es de múltiples emisiones por reloj a tuberías aptas para operaciones de números enteros, y de coma flotante de precisión sencilla y doble, capacidad de ramal de SIMD, operaciones lógicas, operaciones transcendentales y otras operaciones misceláneas. Mientras se esperan los datos de la memoria o una de las funciones compartidas, la lógica de dependencia dentro de las unidades de ejecución 608A-608N hace que un subproceso en espera pase a inactividad hasta que se devuelvan los datos solicitados. Mientras el subproceso en espera está inactivo, los recursos de hardware pueden dedicarse al procesamiento de otros subprocesos. Por ejemplo, durante un retraso asociado con una operación de sombreador de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, un sombreador de fragmentos u otro tipo de programa de sombreado, incluido un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 608A-608N opera sobre matrices de elementos de datos. El número de elementos de datos es el "tamaño de ejecución" o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso a elementos de datos, el enmascaramiento y el control de flujo dentro de las instrucciones. La cantidad de canales puede ser independiente de la cantidad de unidades lógicas aritméticas (ALU) o unidades de punto flotante (FPU) físicas para un procesador de gráficos en particular. Las unidades de ejecución 608A-608N admiten tipos de datos enteros y de punto flotante.

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

Una o más unidades de ejecución se pueden combinar en una unidad de ejecución fusionada 609A-609N que tiene una lógica de control de hilo (607A-607N) que es común a las UE fusionadas. Es posible fusionar varias UE en un grupo de UE. Cada UE del grupo de UE fusionadas se puede configurar para ejecutar un hilo de hardware SIMD independiente. El número de UE en un grupo de UE fusionadas puede variar. Además, se pueden realizar

varios anchos de SIMD por UE, incluidos, entre otros, SIMD8, SIMD16 y SIMD32. Cada unidad de ejecución de gráficos fusionada 609A-609N incluye al menos dos unidades de ejecución. Por ejemplo, la unidad de ejecución fusionada 609A incluye una primera EU 608A, una segunda EU 608B y una lógica de control de subprocesos 607A que es común a la primera EU 608A y a la segunda EU 608B. La lógica de control de subprocesos 607A controla los subprocesos ejecutados en la unidad de ejecución de gráficos fusionada 609A, permitiendo que cada EU dentro de las unidades de ejecución fusionadas 609A-609N se ejecute usando un registro de puntero de instrucción común.

Una o más memorias caché de instrucciones internas (por ejemplo, 606) se incluyen en la lógica de ejecución de subprocesos 600 para almacenar en memoria caché las instrucciones de subprocesos para las unidades de ejecución. Se incluyen una o más cachés de datos (por ejemplo, 612) para almacenar en caché los datos de hilo durante la ejecución del hilo. Se incluye un muestreador 610 para proporcionar muestreo de textura para operaciones 3D y muestreo multimedia para operaciones multimedia. El muestreador 610 incluye una funcionalidad especializada de muestreo de texturas o multimedia, para procesar datos de texturas o multimedia durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las tuberías de gráficos y multimedia envían solicitudes de inicio de hilos a la lógica de ejecución de hilos 600 a través de la lógica de generación y despacho de hilos. Una vez que un grupo de objetos geométricos ha sido procesado y rasterizados en datos de píxeles, se invoca la lógica de procesador de píxeles (por ejemplo, la lógica de sombreador de píxeles, la lógica de sombreador de fragmentos, etc.) dentro del procesador de sombreador 602 para calcular además la información de salida y hacer que los resultados se escriban en superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de esténcil, etc.). Un sombreador de píxeles o un sombreador de fragmentos calcula los valores de los distintos atributos de vértice que se interpolarán en el objeto rasterizado. Luego, la lógica de procesador de píxeles dentro del procesador de sombreador 602 ejecuta un programa de sombreado de píxeles o fragmentos suministrado por una interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreador, el procesador de sombreador 602 envía hilos a una unidad de ejecución (por ejemplo, 608A) a través del despachador de hilos 604. El procesador de sombreador 602 utiliza la lógica de muestreo de textura en el muestreador 610 para acceder a los datos de textura en los mapas de textura almacenados en la memoria. Las operaciones aritméticas sobre los datos de textura y los datos de geometría de entrada calculan datos de color de píxeles para cada fragmento geométrico o descartan uno o más píxeles del procesamiento posterior.

El puerto de datos 614 proporciona un mecanismo de acceso a memoria para que la lógica de ejecución de hilos 600 envíe datos procesados a la memoria para su posterior procesamiento en una tubería de salida de procesador de gráficos. El puerto de datos 614 incluye o se acopla a una o más memorias caché (por ejemplo, caché de datos 612) para almacenar en caché datos para el acceso a la memoria a través del puerto de datos.

Como se ilustra en la Figura 6B, una unidad de ejecución de gráficos 608 puede incluir una unidad de búsqueda de instrucciones 637, una matriz de archivos de registro general (GRF) 624, una matriz de archivos de registro arquitectónico (ARF) 626, un árbitro de hilos 622, una unidad de envío 630, una unidad de ramificación 632, un conjunto de unidades de punto flotante (FPU) SIMD 634 y un conjunto de ALU SIMD de enteros dedicados 635. Los GRF 624 y ARF 626 incluyen el conjunto de archivos de registro general y archivos de registro de arquitectura asociados con cada hilo de hardware simultáneo que puede estar activo en la unidad de ejecución de gráficos 608. El estado arquitectónico de cada hilo se mantiene en el ARF 626, mientras que los datos utilizados durante la ejecución del hilo se almacenan en el GRF 624. El estado de ejecución de cada hilo, incluidos los punteros de instrucciones para cada hilo, se pueden almacenar en registros específicos del hilo en el ARF 626.

La unidad de ejecución de gráficos 608 tiene una arquitectura que es una combinación de multitarea con hilos simultánea (SMT) y multitarea con hilos intercalada de grano fino (IMT). La arquitectura tiene una configuración modular que puede ajustarse con precisión en tiempo de diseño basándose en un número objetivo de subprocesos simultáneos y en un número de registros por unidad de ejecución, donde los recursos de unidad de ejecución se dividen a través de la lógica usada para ejecutar múltiples subprocesos simultáneos.

La unidad de ejecución de gráficos 608 puede emitir conjuntamente múltiples instrucciones, cada una de las cuales puede ser una instrucción diferente. El árbitro de hilos 622 del hilo de unidad de ejecución de gráficos 608 puede enviar las instrucciones a una de las unidades de envío 630, la unidad de ramificación 6342 o las FPU SIMD 634 para su ejecución. Cada hilo de ejecución puede acceder a 128 registros de propósito general dentro del GRF 624, donde cada registro puede almacenar 32 bytes, accesibles como un vector SIMD de 8 elementos de elementos de datos de 32 bits. Cada hilo de unidad de ejecución tiene acceso a 4 Kbytes dentro del GRF 624, y se pueden proporcionar mayores o menores recursos de registro. Se pueden ejecutar hasta siete hilos simultáneamente, aunque el número de hilos por unidad de ejecución también puede variar. En un ejemplo en el que siete hilos pueden acceder a 4 Kbytes, el GRF 624 puede almacenar un total de 28 Kbytes. Los modos de direccionamiento flexibles pueden permitir que los registros se direccionen juntos para construir registros más amplios de manera efectiva o para representar estructuras de datos de bloques rectangulares escalonados.

Las operaciones de memoria, las operaciones de muestreo y otras comunicaciones del sistema de latencia más larga se envían a través de instrucciones de "envío" que son ejecutadas por la unidad de envío de paso de mensajes 630. Las instrucciones de ramificación se envían a una unidad de ramificación 632 dedicada para facilitar la divergencia SIMD y la eventual convergencia.

La unidad de ejecución de gráficos 608 incluye una o más unidades de punto flotante SIMD (FPU) 634 para realizar operaciones de punto flotante. Las FPU 634 también admiten el cálculo de números enteros. Las FPU 634 pueden ejecutar mediante SIMD hasta M número de operaciones de punto flotante (o entero) de 32 bits, o ejecutar mediante SIMD hasta 2M operaciones de enteros de 16 bits o de punto flotante de 16 bits. Al menos una de las FPU proporciona capacidad matemática extendida para soportar funciones matemáticas trascendentales de alto rendimiento y punto flotante de 64 bits de doble precisión. También está presente un conjunto de ALU SIMD 635 de enteros de 8 bits, y puede optimizarse específicamente para realizar operaciones asociadas con cálculos de aprendizaje automático.

Se pueden crear instancias de matrices de múltiples instancias de la unidad de ejecución de gráficos 608 en una agrupación de subnúcleos de gráficos (por ejemplo, una subsección). Para lograr escalabilidad, los arquitectos de producto pueden elegir el número exacto de unidades de ejecución por agrupamiento de subnúcleos. La unidad de ejecución 608 puede ejecutar instrucciones a través de una pluralidad de canales de ejecución. En otro ejemplo, cada hilo ejecutado en la unidad de ejecución de gráficos 608 se ejecuta en un canal diferente.

La **Figura 7** es un diagrama de bloques que ilustra los formatos de instrucción 700 del procesador de gráficos de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución del procesador de gráficos admiten un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los recuadros de 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. El formato de instrucción 700 descrito e ilustrado son macroinstrucciones, en el sentido de que son instrucciones suministradas a la unidad de ejecución, a diferencia de las microoperaciones que resultan de la decodificación de la instrucción una vez que se procesa la instrucción.

Las unidades de ejecución de procesador de gráficos soportan de forma nativa instrucciones en un formato de instrucción de 128 bits 710. Un formato de instrucción compactado de 64 730 bits está disponible para algunas instrucciones según la instrucción seleccionada, las opciones de instrucción y la cantidad de operandos. El formato de instrucción de 128 bits nativo 710 proporciona acceso a todas las opciones de instrucción, mientras que algunas opciones y operaciones están restringidas en el formato de 64 bits 730. Las instrucciones nativas disponibles en el formato de 64 bits 730 varían. La instrucción se compacta en parte usando un conjunto de valores de índice en un campo de índice 713. El hardware de unidad de ejecución consulta un conjunto de tablas de compactación basándose en los valores de índice y usa las salidas de tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 710.

Para cada formato, el código de operación de instrucción 712 define la operación que ha de realizar 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 adición, la unidad de ejecución realiza una operación de adición simultánea en cada canal de color que representa un elemento de textura o un elemento de imagen. De forma predeterminada, la unidad de ejecución ejecuta cada instrucción en todos los canales de datos de los operandos. El campo de control de instrucciones 714 permite el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicción) y el orden de canales de datos (por ejemplo, swizzle). Para las instrucciones en el formato de instrucción de 128 bits 710, un campo de tamaño de ejecución 716 limita la cantidad de canales de datos que se ejecutarán en paralelo. El campo de tamaño de ejecución 716 no está disponible para su uso en el formato de instrucción compacta de 64 bits 730.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos, incluidos dos operandos de origen, src0 720, src1 722 y un destino 718. Las unidades de ejecución admiten instrucciones de destino dual, donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 724), donde el código de operación de instrucción 712 determina la cantidad de operandos de origen. El último operando de origen de una instrucción puede ser un valor inmediato (por ejemplo, codificado) que se pasa con la instrucción.

El formato de instrucción de 128 bits 710 incluye un campo de modo de acceso/dirección 726 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.

El formato de instrucción de 128 bits 710 incluye un campo de modo de acceso/dirección 726, que especifica un modo de dirección y/o un modo de acceso para la instrucción. El modo de acceso se utiliza para definir una alineación de acceso a datos para la instrucción. Algunos ejemplos soportan modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, donde la alineación de bytes del

modo de acceso determina la alineación de acceso de los operandos de instrucción. Por ejemplo, cuando está en un primer modo, la instrucción puede usar direccionamiento alineado a bytes para los operandos de origen y destino y cuando está en un segundo modo, la instrucción puede usar direccionamiento alineado a 16 bytes para todos los operandos de origen y destino.

La porción de modo de dirección del campo de modo de acceso/dirección 726 determina si la instrucción debe utilizar direccionamiento directo o indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, los bits de la instrucción proporcionan directamente la dirección de registro de uno o más operandos. Cuando se utiliza el modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos se puede calcular en base a un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

Las instrucciones se agrupan en función de los campos de bits del código de operación 712 para simplificar la decodificación de opcode 740. Para un código de operación de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de código de operación. La agrupación precisa de códigos de operación que se muestra es simplemente un ejemplo. Un grupo de códigos de operación de movimiento y lógica 742 incluye instrucciones de movimiento de datos y lógica (por ejemplo, mover (mov), comparar (cmp)). El grupo de movimiento y lógica 742 comparte los cinco bits más significativos (MSB), donde las instrucciones de movimiento (mov) tienen el formato 0000xxxxb y las instrucciones lógicas tienen el formato 0001xxxxb. Un grupo de instrucciones de control de flujo 744 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en la forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 746 incluye una combinación de instrucciones, incluidas instrucciones de sincronización (por ejemplo, esperar, enviar) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones matemáticas paralelas 748 incluye instrucciones aritméticas componente por componente (por ejemplo, sumar, multiplicar (mul)) en la forma de 0100xxxxb (por ejemplo, 0x40). El grupo de matemáticas paralelas 748 realiza las operaciones aritméticas en paralelo a través de canales de datos. El grupo de matemáticas vectoriales 750 incluye instrucciones aritméticas (por ejemplo, dp4) en la forma de 0101xxxxb (por ejemplo, 0x50). El grupo de matemáticas vectoriales realiza operaciones aritméticas, tales como cálculos de producto escalar en operandos vectoriales.

Tubería de gráficos

La **Figura 8** es un diagrama de bloques de otro ejemplo de un procesador de gráficos 800. Los elementos de la **Figura 8** que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento, pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a ella.

El procesador de gráficos 800 incluye una tubería de geometría 820, una tubería multimedia 830, un motor de visualización 840, una lógica de ejecución de hilos 850 y una tubería de salida de renderizado 870. El procesador de gráficos 800 es un procesador de gráficos dentro de un sistema de procesamiento multinúcleo que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos se controla mediante escrituras de registros en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 800 a través de una interconexión de anillo 802. La interconexión en anillo 802 acopla el procesador de gráficos 800 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de propósito general. Los comandos de la interconexión en anillo 802 son interpretados por un transmisor de comandos 803, que suministra instrucciones a componentes individuales de la tubería de geometría 820 o de la tubería multimedia 830.

El transmisor de comandos 803 dirige la operación de un buscador de vértices 805 que lee datos de vértice de la memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor de comandos 803. El buscador de vértices 805 proporciona datos de vértice a un sombreador de vértices 807, que realiza operaciones de transformación de espacio de coordenadas e iluminación en cada vértice. El buscador de vértices 805 y el sombreador de vértices 807 ejecutan instrucciones de procesamiento de vértices enviando hilos de ejecución a las unidades de ejecución 852A-852B a través de un despachador de hilos 831.

Las unidades de ejecución 852A-852B son un conjunto de procesadores vectoriales que tienen un conjunto de instrucciones para realizar operaciones de gráficos y multimedia. Las unidades de ejecución 852A-852B tienen una caché L1 851 adjunta que es específica para cada matriz o compartida entre las matrices. La caché se puede configurar como una caché de datos, una caché de instrucciones o una caché única que está particionada para contener datos e instrucciones en diferentes particiones.

La tubería de geometría 820 incluye componentes de teselación para realizar teselación acelerada por hardware de objetos 3D. Un sombreador de envoltura programable 811 configura las operaciones de teselación. Un sombreador de dominio programable 817 proporciona una evaluación de servidor (back-end) de la salida de teselación. Un teselador 813 opera bajo la dirección del sombreador de envoltura 811 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados en base a un modelo geométrico grueso que se proporciona como entrada a la tubería de geometría 820. Si no se utiliza teselación, se pueden

omitir los componentes de teselación (por ejemplo, el sombreador de envoltura 811, el teselador 813 y el sombreador de dominio 817).

Los objetos geométricos completos pueden ser procesados por un sombreador de geometría 819 a través de uno o más hilos enviados a las unidades de ejecución 852A-852B, o pueden proceder directamente al recortador 829. El sombreador de geometría opera sobre objetos geométricos completos, en lugar de sobre vértices o parches de vértices como en etapas anteriores de la tubería de gráficos. Si la teselación está deshabilitada, el sombreador de geometría 819 recibe entrada del sombreador de vértices 807. El sombreador de geometría 819 se puede programar mediante un programa de sombreador de geometría para realizar teselación de geometría si las unidades de teselación están deshabilitadas.

Antes de la rasterización, un recortador 829 procesa los datos de vértice. El recortador 829 puede ser un recortador de función fija o un recortador programable que tenga funciones de recorte y sombreador de geometría. Un componente rasterizador y de prueba de profundidad 873 en la tubería de salida de renderizado 870 envía sombreadores de píxeles para convertir los objetos geométricos en representaciones por píxel. La lógica de sombreador de píxeles está incluida en la lógica de ejecución de hilos 850. Una aplicación puede omitir el componente rasterizador y de prueba de profundidad 873 y acceder a datos de vértice no rasterizados a través de una unidad de salida de flujo 823.

El procesador de gráficos 800 tiene un bus de interconexión, una estructura de interconexión o algún otro mecanismo de interconexión que permite el paso de datos y mensajes entre los componentes principales del procesador. Las unidades de ejecución 852A-852B y las unidades lógicas asociadas (por ejemplo, caché L1 851, muestreador 854, caché de textura 858, etc.) se interconectan a través de un puerto de datos 856 para realizar acceso a memoria y comunicarse con los componentes de tubería de salida de renderizado del procesador. El muestreador 854, las cachés 851, 858 y las unidades de ejecución 852A-852B tienen cada uno rutas de acceso a memoria independientes. La caché de textura 858 también se puede configurar como una caché de muestreador.

La tubería de salida de renderizado 870 contiene un rasterizador y un componente de prueba de profundidad 873 que convierte objetos basados en vértices en una representación asociada basada en píxeles. La lógica de rasterizador incluye una unidad de enmascaramiento/ventana para realizar la rasterización de triángulos y líneas de función fija. También están disponibles una caché de renderizado 878 asociada y una caché de profundidad 879. Un componente de operaciones de píxeles 877 realiza operaciones basadas en píxeles sobre los datos, aunque en algunos casos, las operaciones de píxeles asociadas con operaciones 2D (por ejemplo, transferencias de imágenes de bloques de bits con combinación) son realizadas por el motor 2D 841, o sustituidas en el momento de la visualización por el controlador de visualización 843 utilizando planos de visualización superpuestos. Hay una caché L3 compartida 875 disponible para todos los componentes gráficos, lo que permite compartir datos sin utilizar la memoria principal del sistema.

La tubería multimedia de procesador de gráficos 830 incluye un motor multimedia 837 y un frontal (front-end) de vídeo 834. El frontal de vídeo 834 recibe comandos de tubería desde el transmisor de comandos 803. La tubería multimedia 830 incluye un transmisor de comandos independiente. El frontal de vídeo 834 procesa los comandos multimedia antes de enviar el comando al motor multimedia 837. El motor multimedia 837 incluye una funcionalidad de generación de hilos para generar hilos para su envío a la lógica de ejecución de hilos 850 a través del despachador de hilos 831.

El procesador de gráficos 800 incluye un motor de visualización 840. El motor de visualización 840 es externo al procesador 800 y se acopla con el procesador de gráficos a través de la interconexión en anillo 802 o algún otro bus o estructura de interconexión. El motor de visualización 840 incluye un motor 2D 841 y un controlador de visualización 843. El motor de visualización 840 contiene una lógica de propósito especial capaz de operar independientemente de la tubería 3D. El controlador de visualización 843 se acopla con un dispositivo de visualización (no mostrado), que puede ser un dispositivo de visualización integrado en el sistema, como en una computadora portátil, o un dispositivo de visualización externo conectado a través de un conector de dispositivo de visualización.

La tubería de geometría 820 y la tubería multimedia 830 se pueden configurar para realizar operaciones basadas en múltiples interfaces de programación de gráficos y multimedia y no son específicas de una interfaz de programación de aplicaciones (API). El software de controlador del procesador de gráficos traduce las llamadas API específicas de una biblioteca de gráficos o multimedia en particular en comandos que pueden ser procesados por el procesador de gráficos. Se proporciona soporte para Open Graphics Library (OpenGL), Open Computing Language (OpenCL) y/o la API de gráficos y cómputo Vulkan, todas del Grupo Khronos. También puede proporcionarse soporte para la biblioteca Direct3D de Microsoft Corporation. Puede soportarse una combinación de estas bibliotecas. También puede proporcionarse soporte para la Biblioteca de Visión Informática de Código Abierto (OpenCV). También se soportaría una API futura con una tubería 3D compatible si puede hacerse una asignación desde la tubería de la API futura a la tubería del procesador de gráficos.

Programación de tubería de gráficos

La **Figura 9A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos 900 de acuerdo con algunos ejemplos. La **Figura 9B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos 910 de acuerdo con un ejemplo. Los cuadros con líneas continuas en la **Figura 9A** ilustran los componentes que generalmente se incluyen en un comando de gráficos, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de los comandos de gráficos. El formato de comando de procesador de gráficos 900 ejemplar de la **Figura 9A** incluye campos de datos para identificar un cliente 902, un código de operación de comando (opcode) 904 y datos 906 para el comando. En algunos comandos también se incluyen un sub-opcode 905 y un tamaño de comando 908.

El cliente 902 especifica la unidad cliente del dispositivo de gráficos que procesa los datos del comando. Un analizador de comandos de procesador de gráficos examina el campo de cliente de cada comando para condicionar el procesamiento posterior del comando y enviar los datos de comando a la unidad cliente adecuada. Las unidades cliente de procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de renderizado, una unidad 2D, una unidad 3D y una unidad multimedia. Cada unidad cliente tiene una tubería de procesamiento correspondiente que procesa los comandos. Una vez que la unidad cliente recibe el comando, lee el opcode 904 y, si está presente, el sub-opcode 905 para determinar la operación a realizar. La unidad cliente ejecuta el comando utilizando la información del campo de datos 906. Para algunos comandos se espera un tamaño de comando 908 explícito para especificar el tamaño del comando. El analizador de comandos determina automáticamente el tamaño de al menos algunos de los comandos en función del opcode del comando. Los comandos se alinean mediante múltiplos de una palabra doble.

El diagrama de flujo de la **Figura 9B** ilustra una secuencia de comandos de procesador de gráficos 910 ejemplar. El software o firmware de un sistema de procesamiento de datos que incluye un ejemplo de procesador de gráficos, utiliza una versión de la secuencia de comandos mostrada para configurar, ejecutar y finalizar un conjunto de operaciones gráficas. Se muestra y describe una secuencia de comandos de muestra únicamente con fines ilustrativos. Además, los comandos pueden emitirse como un lote de comandos en una secuencia de comandos, de modo que el procesador de gráficos procesará la secuencia de comandos al menos parcialmente en concurrencia.

La secuencia de comandos de procesador de gráficos 910 puede comenzar con un comando de vaciado de tubería 912 para hacer que cualquier tubería de gráficos activa complete los comandos actualmente pendientes para la tubería. La tubería 3D 922 y la tubería multimedia 924 no funcionan simultáneamente. El vaciado de tubería se realiza para hacer que la tubería de gráficos activa complete todos los comandos pendientes. En respuesta a un vaciado de tubería, el analizador de comandos del procesador de gráficos pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. De manera opcional, cualquier dato en la memoria caché de renderizado que esté marcado como "sucio" se puede vaciar a la memoria. El comando de vaciado de tubería 912 puede usarse para la sincronización de tubería o antes de poner el procesador de gráficos en un estado de baja potencia.

Un comando de selección de tubería 913 se utiliza cuando una secuencia de comandos requiere que el procesador de gráficos cambie explícitamente entre tuberías. Un comando de selección de tubería 913 se requiere solo una vez dentro de un contexto de ejecución antes de emitir comandos de tubería a menos que el contexto sea emitir comandos para ambas tuberías. Se requiere un comando de vaciado de tubería 912 inmediatamente antes de un cambio de tubería a través del comando de selección de tubería 913.

Un comando de control de tubería 914 configura una tubería de gráficos para su funcionamiento y se utiliza para programar la tubería 3D 922 y la tubería multimedia 924. El comando de control de tubería 914 configura el estado de tubería para la tubería activa. El comando de control de tubería 914 se utiliza para la sincronización de tubería y para borrar datos de una o más memorias caché dentro de la tubería activa antes de procesar un lote de comandos.

Los comandos de estado de memoria intermedia de retorno 916 se utilizan para configurar un conjunto de memorias intermedias de retorno para que las respectivas tuberías escriban datos. Algunas operaciones de tubería requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. El procesador de gráficos también utiliza una o más memorias intermedias de retorno para almacenar datos de salida y realizar comunicación entre hilos. El estado de memoria intermedia de retorno 916 incluye la selección del tamaño y la cantidad de memorias intermedias de retorno a utilizar para un conjunto de operaciones de tubería.

Los comandos restantes en la secuencia de comandos difieren según la tubería activa para las operaciones. En base a una determinación de tubería 920, la secuencia de comandos se adapta a la tubería 3D 922 comenzando con el estado de tubería 3D 930 o a la tubería multimedia 924 comenzando en el estado de tubería multimedia 940.

Los comandos para configurar el estado de tubería 3D 930 incluyen comandos de configuración de estado 3D para el estado de la memoria intermedia de vértice, el estado de elemento de vértice, el estado de color constante, el estado de memoria intermedia de profundidad y otras variables de estado que se deben configurar antes de que se procesen los comandos primitivos 3D. Los valores de estos comandos se determinan, al menos en parte, en función de la API 3D particular en uso. Los comandos de estado de tubería 3D 930 también pueden deshabilitar u omitir selectivamente ciertos elementos de la tubería si esos elementos no se utilizarán.

El comando primitivo 3D 932 se utiliza para enviar primitivos 3D para que sean procesados por la tubería 3D. Los comandos y los parámetros asociados que se pasan al procesador de gráficos a través del comando primitivo 3D 932 se reenvían a la función de búsqueda de vértices en la tubería de gráficos. La función de obtención de vértices utiliza los datos del comando primitivo 3D 932 para generar estructuras de datos de vértice. Las estructuras de datos de vértice se almacenan en una o más memorias intermedias de retorno. El comando primitivo 3D 932 se utiliza para realizar operaciones de vértice en primitivos 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la tubería 3D 922 envía hilos de ejecución de sombreadores a unidades de ejecución de procesadores de gráficos.

La tubería 3D 922 se activa a través de un comando o evento de ejecución 934. Una escritura de registro desencadena la ejecución de comando. La ejecución se activa a través de un comando 'go' o 'kick' en la secuencia de comandos. La ejecución de comando se activa mediante un comando de sincronización de tubería para limpiar la secuencia de comandos a través de la tubería de gráficos. La tubería 3D realizará el procesamiento de geometría para los primitivos 3D. Una vez completadas las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles colorea los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones del servidor (back-end) de píxeles para esas operaciones.

La secuencia de comandos de procesador de gráficos 910 sigue la ruta de la tubería multimedia 924 cuando realiza operaciones multimedia. En general, el uso específico y la manera de programación de la tubería multimedia 924 dependen de los medios o las operaciones de cálculo que se realizarán. Es posible que se descarguen operaciones de decodificación multimedia específicas a la tubería multimedia durante la decodificación multimedia. La tubería multimedia también puede sortearse y la descodificación multimedia puede realizarse, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. La tubería multimedia también incluye elementos para operaciones de unidad de procesador de gráficos de propósito general (GPGPU), donde el procesador de gráficos se utiliza para realizar operaciones vectoriales SIMD utilizando programas de sombreador computacionales que no están relacionados explícitamente con la representación de primitivos gráficos.

La tubería multimedia 924 está configurada de manera similar a la tubería 3D 922. Un conjunto de comandos para configurar el estado de tubería multimedia 940 se despachan o se colocan en una cola de comandos antes de los comandos de objeto multimedia 942. Los comandos para el estado de tubería multimedia 940 incluyen datos para configurar los elementos de tubería multimedia que se utilizarán para procesar los objetos multimedia. Esto incluye datos para configurar la lógica de codificación y decodificación de vídeo dentro de la tubería multimedia, tal como el formato de codificación o decodificación. Los comandos para el estado de tubería multimedia 940 también admiten el uso de uno o más punteros a elementos de estado "indirectos" que contienen un lote de configuraciones de estado.

Los comandos de objetos multimedia 942 proporcionan punteros a objetos multimedia para su procesamiento por parte de la tubería multimedia. Los objetos multimedia incluyen memorias intermedias que contienen datos de vídeo para ser procesados. Todos los estados de tubería multimedia deben ser válidos antes de emitir un comando de objeto multimedia 942. Una vez que se configura el estado de tubería y se ponen en cola los comandos de objeto multimedia 942, la tubería multimedia 924 se activa a través de un comando de ejecución 944 o un evento de ejecución equivalente (por ejemplo, escritura de registro). La salida de la tubería multimedia 924 puede luego posprocesarse mediante operaciones proporcionadas por el canal 3D 922 o el canal multimedia 924. Las operaciones de GPGPU se configuran y ejecutan de manera similar a las operaciones multimedia.

Arquitectura de software de gráficos

La **Figura 10** ilustra una arquitectura de software de gráficos ejemplar para un sistema de procesamiento de datos 1000 de acuerdo con algunos ejemplos. La arquitectura de software incluye una aplicación de gráficos 3D 1010, un sistema operativo 1020 y al menos un procesador 1030. El procesador 1030 incluye un procesador de gráficos 1032 y uno o más núcleos de procesador de propósito general 1034. La aplicación de gráficos 1010 y el sistema operativo 1020 se ejecutan cada uno en la memoria de sistema 1050 del sistema de procesamiento de datos.

La aplicación de gráficos 3D 1010 contiene uno o más programas de sombreador que incluyen instrucciones de sombreador 1012. Las instrucciones del lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el lenguaje de sombreador de alto nivel (HLSL) o el lenguaje de sombreador OpenGL (GLSL). La aplicación también incluye instrucciones ejecutables 1014 en un lenguaje de máquina adecuado para ser

ejecutado por el núcleo de procesador de propósito general 1034. La aplicación también incluye objetos gráficos 1016 definidos por datos de vértice.

El sistema operativo 1020 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo propietario similar a UNIX o un sistema operativo de código abierto similar a UNIX que utiliza una variante del núcleo Linux. El sistema operativo 1020 puede soportar una API de gráficos 1022, tal como la API Direct3D, la API OpenGL o la API Vulkan. Cuando se utiliza la API Direct3D, el sistema operativo 1020 utiliza un compilador de sombreador frontal 1024 para compilar cualquier instrucción de sombreador 1012 en HLSL en un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede realizar una precompilación de sombreadores. Los sombreadores de alto nivel se compilan en sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 1010. Las instrucciones de sombreador 1012 se proporcionan en una forma intermedia, tal como una versión de la Representación Intermedia Portátil Estándar (SPIR) utilizada por la API de Vulkan.

El controlador de gráficos de modo usuario 1026 contiene un compilador de sombreador de servidor 1027 para convertir las instrucciones de sombreador 1012 en una representación específica del hardware. Cuando se utiliza la API OpenGL, las instrucciones de sombreador 1012 en el lenguaje de alto nivel GLSL se pasan a un controlador de gráficos de modo usuario 1026 para su compilación. El controlador de gráficos de modo usuario 1026 utiliza funciones de modo núcleo de sistema operativo 1028 para comunicarse con un controlador de gráficos de modo núcleo 1029. El controlador de gráficos en modo núcleo 1029 se comunica con el procesador de gráficos 1032 para enviar comandos e instrucciones.

Implementaciones de núcleo IP

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

La **Figura 11A** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleos IP 1100 que puede utilizarse para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo IP 1100 se puede utilizar para generar diseños modulares y reutilizables que se pueden incorporar en un diseño más grande o utilizar para construir un circuito integrado completo (por ejemplo, un circuito integrado de SOC). Una instalación de diseño 1130 puede generar una simulación de software 1110 de un diseño de núcleo IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 1110 se puede utilizar para diseñar, probar y verificar el comportamiento del núcleo IP utilizando un modelo de simulación 1112. El modelo de simulación 1112 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Luego se puede crear o sintetizar un diseño de nivel de transferencia de registro (RTL) 1115 a partir del modelo de simulación 1112. El diseño de RTL 1115 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, incluida la lógica asociada realizada utilizando las señales digitales modeladas. Además de un diseño de RTL 1115, también se pueden crear, diseñar o sintetizar diseños de nivel inferior a nivel lógico o a nivel de transistor. Por lo tanto, los detalles particulares del diseño inicial y la simulación pueden variar.

El diseño de RTL 1115 o equivalente puede ser sintetizado además por la instalación de diseño en un modelo de hardware 1120, que puede estar en un lenguaje de descripción de hardware (HDL) o alguna otra representación de datos de diseño físico. El HDL se puede simular o probar además para verificar el diseño de núcleo IP. El diseño de núcleo IP se puede almacenar para su entrega a una instalación de fabricación de terceros 1165 utilizando memoria no volátil 1140 (por ejemplo, disco duro, memoria flash, o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño de núcleo IP se puede transmitir (por ejemplo, a través de Internet) a través de una conexión por cable 1150 o una conexión inalámbrica 1160. La instalación de fabricación 1165 puede entonces fabricar un circuito integrado que se basa al menos en parte en el diseño de núcleo IP. El circuito integrado fabricado está configurado para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

La **Figura 11B** ilustra una vista lateral en sección transversal de un conjunto de paquete de circuitos integrados 1170, de acuerdo con algunas realizaciones descritas en este documento. El conjunto de paquete de circuitos integrados 1170 ilustra una implementación de uno o más dispositivos procesadores o aceleradores como se describe en el presente documento. El conjunto de paquete 1170 incluye múltiples unidades de lógica de hardware

1172, 1174 conectadas a un sustrato 1180. La lógica 1172, 1174 puede implementarse al menos parcialmente en hardware de lógica configurable o lógica de funcionalidad fija, y puede incluir una o más porciones de cualquiera del núcleo o núcleos de procesador, procesador(es) de gráficos u otros dispositivos aceleradores descritos en el presente documento. Cada unidad de lógica 1172, 1174 puede implementarse dentro de una pastilla de semiconductores y acoplarse con el sustrato 1180 mediante una estructura de interconexión 1173. La estructura de interconexión 1173 puede configurarse para encaminar señales eléctricas entre la lógica 1172, 1174 y el sustrato 1180, y puede incluir interconexiones tales como, pero sin limitación, protuberancias o pilares. La estructura de interconexión 1173 puede configurarse para encaminar señales eléctricas tales como, por ejemplo, señales de entrada/salida (E/S) y/o señales de alimentación o de masa asociadas con la operación de la lógica 1172, 1174. El sustrato 1180 es un sustrato laminado a base de epoxi. El sustrato de paquete 1180 puede incluir otros tipos de sustratos adecuados en otros ejemplos. El conjunto de paquete 1170 se puede conectar a otros dispositivos eléctricos a través de una interconexión de paquete 1183. La interconexión de paquete 1183 puede estar acoplada a una superficie del sustrato 1180 para enrutar señales eléctricas a otros dispositivos eléctricos, tal como una placa base, otro conjunto de chips o un módulo multichip.

Las unidades de lógica 1172, 1174 están acopladas eléctricamente con un puente 1182 que está configurado para enrutar señales eléctricas entre la lógica 1172, 1174. El puente 1182 puede ser una estructura de interconexión densa que proporciona una ruta para señales eléctricas. El puente 1182 puede incluir un sustrato de puente compuesto de vidrio o un material semiconductor adecuado. Se pueden formar características de enrutamiento eléctrico en el sustrato de puente para proporcionar una conexión de chip a chip entre la lógica 1172, 1174.

Aunque se ilustran dos unidades lógicas 1172, 1174 y un puente 1182, los ejemplos descritos en el presente documento pueden incluir más o menos unidades lógicas en una o más matrices. Las una o más matrices pueden estar conectadas por cero o más puentes, ya que el puente 1182 puede excluirse cuando la lógica está incluida en una sola matriz. Alternativamente, se pueden conectar múltiples matrices o unidades lógicas mediante uno o más puentes. Además, se pueden conectar varias unidades lógicas, matrices y puentes entre sí en otras configuraciones posibles, incluidas configuraciones tridimensionales.

Ejemplo de circuito integrado de sistema en chip

Las Figs. 12 a 14 ilustran circuitos integrados ejemplares y procesadores de gráficos asociados que pueden fabricarse utilizando uno o más núcleos IP, de acuerdo con varios ejemplos descritos en el presente documento. Además de lo ilustrado, se pueden incluir otros circuitos y lógica, incluidos procesadores/núcleos de gráficos adicionales, controladores de interfaz periférica o núcleos de procesador de propósito general.

La **Figura 12** es un diagrama de bloques que ilustra un circuito integrado de sistema en chip 1200 ejemplar que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con un ejemplo. El circuito integrado 1200 ejemplar incluye uno o más procesadores de aplicaciones 1205 (por ejemplo, CPU), al menos un procesador de gráficos 1210 y puede incluir adicionalmente un procesador de imágenes 1215 y/o un procesador de vídeo 1220, cualquiera de los cuales puede ser un núcleo IP modular de la misma o múltiples instalaciones de diseño diferentes. El circuito integrado 1200 incluye lógica de periférico o de bus que incluye un controlador de USB 1225, controlador de UART 1230, un controlador de SPI/SDIO 1235 y un controlador de I2S/I2C 1240. Además, el circuito integrado puede incluir un dispositivo de visualización 1245 acoplado a uno o más de un controlador de interfaz multimedia de alta definición (HDMI) 1250 y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 1255. El almacenamiento se puede proporcionar por un subsistema de memoria flash 1260 que incluye la memoria flash y un controlador de memoria flash. La interfaz de memoria se puede proporcionar por medio de un controlador de memoria 1265 para acceso a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 1270.

Las **Figuras 13A-13B** son diagramas de bloques que ilustran procesadores de gráficos ejemplares para su uso dentro de un SoC, de acuerdo con los ejemplos descritos en el presente documento. La **Figura 13A** ilustra un procesador de gráficos 1310 ejemplar de un circuito integrado de sistema en chip que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con un ejemplo. La **Figura 13B** ilustra un procesador de gráficos 1340 ejemplar adicional de un circuito integrado de sistema en chip que puede fabricarse utilizando uno o más núcleos IP, de acuerdo con un ejemplo. El procesador de gráficos 1310 de la **Figura 13A** es un ejemplo de un núcleo de procesador de gráficos de bajo consumo. El procesador de gráficos 1340 de la **Figura 13B** es un ejemplo de un núcleo de procesador de gráficos de mayor rendimiento. Cada uno de los procesadores de gráficos 1310, 1340 pueden ser variantes del procesador de gráficos 1210 de la **Figura 12**.

Como se muestra en la **Figura 13A**, el procesador de gráficos 1310 incluye un procesador de vértices 1305 y uno o más procesadores de fragmentos 1315A-1315N (por ejemplo, 1315A, 1315B, 1315C, 1315D, hasta 1315N-1 y 1315N). El procesador de gráficos 1310 puede ejecutar diferentes programas de sombreador a través de una lógica separada, de modo que el procesador de vértices 1305 está optimizado para ejecutar operaciones para programas de sombreador de vértices, mientras que uno o más procesadores de fragmentos 1315A-1315N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreador de fragmentos o píxeles. El procesador de vértices 1305 realiza la etapa de procesamiento de vértices de la tubería de gráficos 3D

y genera primitivas y datos de vértice. El procesador o procesadores de fragmentos 1315A-1315N usan los datos de primitiva y de vértice generados por el procesador de vértices 1305 para producir una memoria intermedia de fotogramas que se visualiza en un dispositivo de visualización. El procesador o procesadores de fragmentos 1315A-1315N están optimizados para ejecutar programas de sombreador de fragmentos según lo previsto en la API OpenGL, que pueden usarse para realizar operaciones similares a las de un programa de sombreador de píxeles según lo previsto en la API Direct 3D.

El procesador de gráficos 1310 incluye además una o más unidades de gestión de memoria (MMU) 1320A-1320B, cachés 1325A-1325B e interconexiones de circuitos 1330A-1330B. La una o más MMU 1320A-1320B proporcionan una asignación de direcciones virtuales a físicas para el procesador de gráficos 1310, incluido el procesador de vértices 1305 y/o el procesador o procesadores de fragmentos 1315A-1315N, que pueden hacer referencia a datos de vértice o de imagen/textura almacenados en la memoria, además de los datos de vértice o de imagen/textura almacenados en las cachés 1325A-1325B. La una o más MMU 1320A-1320B pueden estar sincronizadas con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con el uno o más procesadores de aplicación 1205, procesador de imagen 1215 y/o procesador de vídeo 1220 de la **Figura 12**, de modo que cada procesador 1205-1220 pueda participar en un sistema de memoria virtual compartido o unificado. Las una o más interconexiones de circuito 1330A-1330B permiten que el procesador de gráficos 1310 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.

Como se muestra en la **Figura 13B**, el procesador de gráficos 1340 incluye una o más MMU 1320A-1320B, cachés 1325A-1325B e interconexiones de circuitos 1330A-1330B del procesador de gráficos 1310 de la **Figura 13A**. El procesador de gráficos 1340 incluye uno o más núcleos de sombreador 1355A-1355N (por ejemplo, 1455A, 1355B, 1355C, 1355D, 1355E, 1355F, hasta 1355N-1 y 1355N), lo que proporciona una arquitectura de núcleo de sombreador unificada en la que un solo núcleo o tipo de núcleo puede ejecutar todos los tipos de código de sombreado programable, incluido el código de programa de sombreado para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. La cantidad exacta de núcleos de sombreador presentes puede variar entre implementaciones. Además, el procesador de gráficos 1340 incluye un administrador de tareas entre núcleos 1345, que actúa como un despachador de subprocesos para despachar subprocesos de ejecución a uno o más núcleos de sombreador 1355A-1355N y una unidad de tesela 1358 para acelerar las operaciones de tesela para el renderizado basado en teselas, en la que las operaciones de renderizado para una escena se subdividen en el espacio de la imagen, por ejemplo para explotar la coherencia espacial local dentro de una escena o para optimizar el uso de memoria caché interna.

Las **Figuras 14A-14B** ilustran lógica de procesador de gráficos ejemplar adicional de acuerdo con los ejemplos descritos en el presente documento. La **Figura 14A** ilustra un núcleo de gráficos 1400 que puede estar incluido dentro del procesador de gráficos 1210 de la **Figura 12**, y puede ser un núcleo de sombreador unificado 1355A-1355N como en la **Figura 13B**. La **Figura 14B** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela 1430 adicional, que es una unidad de procesamiento de gráficos de propósito general altamente paralela adecuada para su implementación en un módulo multichip.

Como se muestra en la **Figura 14A**, el núcleo de gráficos 1400 incluye una caché de instrucciones compartida 1402, una unidad de textura 1418 y una memoria compartida/caché 1420 que son comunes a los recursos de ejecución dentro del núcleo de gráficos 1400. El núcleo de gráficos 1400 puede incluir múltiples segmentos 1401A-1401N o particiones para cada núcleo, y un procesador de gráficos puede incluir múltiples instancias del núcleo de gráficos 1400. Los segmentos 1401A-1401N pueden incluir lógica de soporte que incluye una caché de instrucciones local 1404A-1404N, un planificador de subprocesos 1406A-1406N, un despachador de subprocesos 1408A-1408N y un conjunto de registros 1410A-1440N. Para realizar operaciones lógicas, los segmentos 1401A-1401N pueden incluir un conjunto de unidades de función (AFU 1412A-1412N) adicionales, unidades de coma flotante (FPU 1414A-1414N), unidades aritmético-lógicas de números enteros (ALU 1416-1416N), unidades de cálculo de direcciones (ACU 1413A-1413N), unidades de coma flotante de precisión doble (DPFPU 1415A-1415N) y unidades de procesamiento de matrices (MPU 1417A-1417N).

Algunas de las unidades computacionales operan con una precisión específica. Por ejemplo, las FPU 1414A-1414N pueden realizar operaciones de coma flotante de precisión sencilla (32 bits) y de media precisión (16 bits), mientras que las DPFPU 1415A-1415N realizan operaciones de coma flotante de precisión doble (64 bits). Las ALU 1416A-1416N pueden realizar operaciones con números enteros de precisión variable con una precisión de 8 bits, 16 bits y 32 bits, y pueden configurarse para operaciones de precisión mixta. Las MPU 1417A-1417N también pueden configurarse para operaciones con matrices de precisión mixta, incluyendo operaciones de coma flotante de media precisión y operaciones de enteros de 8 bits. Las MPU 1417-1417N pueden realizar una diversidad de operaciones con matrices para acelerar las estructuras de aplicaciones de aprendizaje automático, incluyendo la habilitación del soporte para la multiplicación de matriz con matriz general (GEMM) acelerada. Las AFU 1412A-1412N pueden realizar operaciones lógicas adicionales no soportadas por las unidades de coma flotante o de números enteros, incluyendo operaciones trigonométricas (por ejemplo, seno, coseno, etc.).

Como se muestra en la **Figura 14B**, una unidad de procesamiento de propósito general (GPGPU) 1430 se puede configurar para permitir que se realicen operaciones de cálculo altamente paralelas mediante una matriz de

unidades de procesamiento de gráficos. Además, la GPGPU 1430 se puede vincular directamente a otras instancias de la GPGPU para crear una agrupación de múltiples GPU para mejorar la velocidad de entrenamiento para redes neuronales particularmente profundas. La GPGPU 1430 incluye una interfaz de anfitrión 1432 para habilitar una conexión con un procesador anfitrión. La interfaz de anfitrión 1432 es una interfaz PCI Express. Sin embargo, la interfaz de anfitrión también puede ser una interfaz de comunicaciones o una estructura de comunicaciones específica del proveedor. La GPGPU 1430 recibe comandos del procesador anfitrión y utiliza un planificador global 1434 para distribuir los hilos de ejecución asociados con esos comandos a un conjunto de agrupaciones de cálculo 1436A-1436H. Las agrupaciones de cálculo 1436A-1436H comparten una memoria caché 1438. La memoria caché 1438 puede servir como memoria caché de nivel superior para memorias caché dentro de las agrupaciones de cálculo 1436A-1436H.

La GPGPU 1430 incluye la memoria 14434A-14434B acoplada con las agrupaciones de cálculo 1436A-1436H mediante un conjunto de controladores de memoria 1442A-1442B. La memoria 1434A-1434B puede incluir varios tipos de dispositivos de memoria, incluida la memoria de acceso aleatorio dinámico (DRAM) o la memoria de acceso aleatorio de gráficos, tal como la memoria de acceso aleatorio de gráficos sincrónica (SGRAM), incluida la memoria de doble velocidad de datos de gráficos (GDDR).

Los agrupaciones de cálculo 1436A-1436H incluyen cada una un conjunto de núcleos de gráficos, tal como el núcleo de gráficos 1400 de la FIG. 14A, que puede incluir múltiples tipos de unidades lógicas de punto flotante y de entero que pueden realizar operaciones computacionales en un rango de precisiones, incluidas las adecuadas para cálculos de aprendizaje automático. Por ejemplo, al menos un subconjunto de las unidades de punto flotante en cada una de los agrupaciones de cálculo 1436A-1436H se pueden configurar para realizar operaciones de punto flotante de 16 bits o 32 bits, mientras que un subconjunto diferente de las unidades de punto flotante se pueden configurar para realizar operaciones de punto flotante de 64 bits.

Se pueden configurar múltiples instancias de la GPGPU 1430 para que funcionen como una agrupación de cálculo. El mecanismo de comunicación utilizado por la agrupación de cálculo para la sincronización y el intercambio de datos varía. Las múltiples instancias de la GPGPU 1430 se comunican a través de la interfaz de anfitrión 1432. La GPGPU 1430 incluye un concentrador de E/S 1439 que acopla la GPGPU 1430 con un enlace de GPU 1440 que permite una conexión directa a otras instancias de la GPGPU. El enlace GPU 1440 está acoplado a un puente GPU a GPU dedicado que permite la comunicación y sincronización entre múltiples instancias de la GPGPU 1430. El enlace GPU 1440 se acopla a una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. Las múltiples instancias de la GPGPU 1430 están ubicadas en sistemas de procesamiento de datos separados y se comunican a través de un dispositivo de red al que se puede acceder mediante la interfaz de anfitrión 1432. El enlace GPU 1440 se puede configurar para permitir una conexión a un procesador anfitrión además de o como alternativa a la interfaz anfitrión 1432.

Aunque la configuración ilustrada de la GPGPU 1430 puede configurarse para entrenar redes neuronales, un ejemplo proporciona una configuración alternativa de la GPGPU 1430, que puede configurarse para el despliegue dentro de una plataforma de inferenciación de alto rendimiento o de baja potencia. En una configuración de inferencia, la GPGPU 1430 incluye menos de las agrupaciones de cálculo 1436A-1436H en relación con la configuración de entrenamiento. Adicionalmente, la tecnología de memoria asociada con la memoria 1434A-1434B puede diferir entre configuraciones de inferencia y de entrenamiento, con tecnologías de memoria de mayor ancho de banda dedicadas a las configuraciones de entrenamiento. La configuración de inferencia del GPGPU 1430 puede soportar la inferencia de instrucciones específica. Por ejemplo, una configuración de inferencia puede proporcionar soporte para una o más instrucciones de producto escalar de números enteros de 8 bits, que se utilizan comúnmente durante operaciones de inferencia para redes neuronales implementadas.

APARATO Y MÉTODO PARA ENTRENAR UN MOTOR DE APRENDIZAJE AUTOMÁTICO DURANTE EL TIEMPO DE EJECUCIÓN

Como se mencionó anteriormente, el trazado de rayos es una técnica de procesamiento de gráficos en la que se simula un transporte de luz a través de un renderizado basado en la física. Una de las operaciones clave en el trazado de rayos es procesar una consulta de visibilidad que requiere pruebas de cruce e intersección de nodos en una jerarquía de volumen delimitador (BVH).

Las técnicas basadas en el trazado de rayos y trayectorias calculan imágenes trazando rayos y trayectorias a través de cada píxel y utilizando un muestreo aleatorio para calcular efectos avanzados tales como sombras, brillo, iluminación indirecta, etc. El uso de solo unas pocas muestras es rápido pero produce imágenes ruidosas, mientras que el uso de muchas muestras produce imágenes de alta calidad, pero tiene un coste prohibitivo.

En los últimos años, una solución innovadora para el trazado de rayos y trayectorias para uso en tiempo real ha llegado en forma de "eliminación de ruido": el proceso de utilizar técnicas de procesamiento de imágenes para producir imágenes filtradas/sin ruido de alta calidad a partir de entradas ruidosas de bajo recuento de muestras. Las técnicas de eliminación de ruido más efectivas se basan en el aprendizaje profundo/automático, donde las redes neuronales convolucionales (RNC) aprenden cómo probablemente se vería una imagen ruidosa si se hubiera

calculado con más muestras. Esto funciona produciendo datos de entrenamiento con entradas de bajo recuento de muestras y verdad fundamental, una solución totalmente convergente para la misma escena y punto de vista, y entrenando la RNC para predecir el píxel convergente a partir de un vecindario de entradas de píxeles ruidosos alrededor del píxel en cuestión.

Aunque no es perfecta, esta técnica de eliminación de ruido basada en IA ha demostrado ser sorprendentemente eficaz. Sin embargo, la salvedad es que se requieren buenos datos de entrenamiento, ya que de lo contrario la red puede predecir los resultados incorrectos. Por ejemplo, si un estudio de películas animadas entrenó una RNC de eliminación de ruido en películas pasadas con escenas en tierra y luego intentó usar la RNC entrenada para eliminar el ruido de los fotogramas de una nueva película ambientada en el agua, la operación de eliminación de ruido funcionará de manera subóptima.

Para abordar este problema, un ejemplo recopila datos de aprendizaje de forma dinámica, durante la renderización, y entrena continuamente un motor de aprendizaje automático, tal como una RNC, en función de los datos en los que se está ejecutando en ese momento, mejorando así continuamente el motor de aprendizaje automático para la tarea en cuestión. Este ejemplo aún puede realizar una fase de entrenamiento antes del tiempo de ejecución, pero continúa ajustando los pesos de aprendizaje automático según sea necesario durante el tiempo de ejecución. Además, este ejemplo evita el alto costo de calcular los datos de referencia necesarios para el entrenamiento al restringir la generación de datos de aprendizaje a una subregión de la imagen en cada fotograma o cada N fotogramas. En particular, las entradas ruidosas de un fotograma se generan para eliminar el ruido del fotograma completo con la red actual. Además, se genera una pequeña región de píxeles de referencia y se utiliza para el entrenamiento continuo, como se describe a continuación.

Las implementaciones de eliminación de ruido existentes funcionan en una fase de entrenamiento y una fase de tiempo de ejecución. Durante la fase de entrenamiento, se define una topología de red que recibe una región de NxN píxeles con varios canales de datos por píxel, tales como color de píxel, profundidad, normal, desviación normal, identificadores de primitivas y albedo, y genera un color de píxel final. Se genera un conjunto de datos de entrenamiento "representativos" utilizando el valor de una fotograma de entradas de bajo recuento de muestras y haciendo referencia a los colores de píxel "deseados" calculados con un recuento de muestras muy alto. La red se entrena para estas entradas, lo que genera un conjunto de pesos "ideales" para la red. En estas implementaciones, los datos de referencia se utilizan para entrenar los pesos de la red para que la salida de la red coincida lo más posible con el resultado deseado.

En el tiempo de ejecución, se cargan los pesos de red ideales calculados previamente y se inicializa la red. Para cada fotograma, se genera una imagen de bajo recuento de muestras de entradas con eliminación de ruido (es decir, la misma que se utiliza para el entrenamiento). Para cada píxel, la vecindad dada de entradas de píxeles se ejecuta a través de la red para predecir el color del píxel "sin ruido", lo que genera un fotograma sin ruido.

La Figura 15 ilustra un ejemplo de implementación de entrenamiento inicial. Un motor de aprendizaje automático 1500 (por ejemplo, una RNC) recibe una región de N x N píxeles como datos de imagen de alto recuento de muestras 1702 con varios canales de datos por píxel, tal como color de píxel, profundidad, normal, desviación normal, identificadores de primitivas y albedo, y genera colores de píxel finales. Los datos de entrenamiento representativos se generan utilizando el valor de un fotograma de entradas de bajo recuento de muestras 1501. La red se entrena para estas entradas, generando un conjunto de pesos "ideales" 1505 que el motor de aprendizaje automático 1500 utiliza posteriormente para eliminar el ruido de las imágenes de bajo recuento de muestras en tiempo de ejecución.

Para mejorar las técnicas anteriores, un ejemplo de la invención aumenta la fase de eliminación de ruido para generar nuevos datos de entrenamiento en cada fotograma o en un subconjunto de fotogramas (por ejemplo, cada N fotogramas donde N = 2, 3, 4, 10, 25, etc.). En particular, como se ilustra en la Figura 16, este ejemplo elige una o más regiones en cada fotograma, denominadas aquí como "nuevas regiones de referencia" 1602, que se representan con un alto recuento de muestras en una memoria intermedia de alto recuento de muestras 1604 separada. Una memoria intermedia de bajo recuento de muestras 1603 almacena el fotograma de entrada de bajo recuento de muestras 1601 (incluida la región de muestras bajas 1604 correspondiente a la nueva región de referencia 1602).

La ubicación de la nueva región de referencia 1602 se selecciona aleatoriamente. Alternativamente, la ubicación de la nueva región de referencia 1602 se puede ajustar de una manera preestablecida para cada nuevo fotograma (por ejemplo, utilizando un movimiento predefinido de la región entre fotogramas, limitado a una región especificada en el centro del fotograma, etc.).

Independientemente de cómo se seleccione la nueva región de referencia, el motor de aprendizaje automático 1600 la utiliza para refinar y actualizar continuamente los pesos entrenados 1605 utilizados para la eliminación de ruido. En particular, en una realización, se representan los colores de píxeles de referencia de cada nueva región de referencia 1602 y las entradas de píxeles de referencia ruidosas de una región de bajo recuento de muestras correspondiente. Luego se realiza un entrenamiento complementario en el motor de aprendizaje automático 1600

utilizando la región de referencia de alto recuento de muestras 1602 y la región de bajo recuento de muestras correspondiente. A diferencia del entrenamiento inicial, este entrenamiento se realiza de manera continua durante el tiempo de ejecución para cada nueva región de referencia 1602, lo que garantiza que el motor de aprendizaje automático 1600 se entrene con precisión. Por ejemplo, se pueden evaluar los canales de datos por píxel (por ejemplo, color de píxel, profundidad, normal, desviación normal, etc.), que el motor de aprendizaje automático 1600 utiliza para realizar ajustes en los pesos entrenados 1605. Al igual que en el caso de entrenamiento (Figura 15), el motor de aprendizaje automático 1600 se entrena hacia un conjunto de pesos ideales 1605 para eliminar el ruido del fotograma de entrada de bajo recuento de muestras 1601 para generar el fotograma sin ruido 1620. Sin embargo, los pesos entrenados 1605 se actualizan continuamente, en función de nuevas características de imagen de nuevos tipos de fotogramas de entrada de bajo recuento de muestras 1601.

Las operaciones de reentrenamiento realizadas por el motor de aprendizaje automático 1600 se ejecutan simultáneamente en un proceso en segundo plano en la unidad de procesamiento de gráficos (GPU) o el procesador anfitrión. El bucle de renderizado, que puede implementarse como un componente de controlador y/o un componente de hardware de GPU, produce continuamente nuevos datos de entrenamiento (por ejemplo, en forma de nuevas regiones de referencia 1602) que coloca en una cola. El proceso de entrenamiento en segundo plano, ejecutado en la GPU o en el procesador anfitrión, lee continuamente los nuevos datos de entrenamiento de esta cola, vuelve a entrenar el motor de aprendizaje automático 1600 y lo actualiza con nuevos pesos 1605 a intervalos apropiados.

La Figura 17 ilustra un ejemplo de una implementación de este tipo en la que el proceso de entrenamiento en segundo plano 1700 es implementado por la CPU anfitrión 1710. En particular, el proceso de entrenamiento en segundo plano 1700 utiliza la nueva región de referencia de alto recuento de muestras 1602 y la región de muestras bajas 1604 correspondiente para actualizar continuamente los pesos entrenados 1605, actualizando así el motor de aprendizaje automático 1600.

Como se ilustra en la **Figura 18A**, en una implementación tal como en un juego en línea multijugador, diferentes máquinas anfitrionas 1820-1822 generan individualmente regiones de referencia que un proceso de entrenamiento en segundo plano 1700A-C transmite a un servidor 1800 (por ejemplo, tal como un servidor de juegos). El servidor 1800 luego realiza el entrenamiento en un motor de aprendizaje automático 1810 utilizando las nuevas regiones de referencia recibidas de cada uno de los anfitriones 1821-1822, actualizando los pesos 1805 como se describió anteriormente. Transmite estos pesos 1805 a las máquinas anfitrionas 1820 que almacenan los pesos 1605A-C, actualizando así cada motor de aprendizaje automático individual (no mostrado). Debido a que se le puede proporcionar al servidor 1800 una gran cantidad de regiones de referencia en un corto período de tiempo, puede actualizar de manera eficiente y precisa los pesos para cualquier aplicación dada (por ejemplo, un juego en línea) que esté siendo ejecutada por los usuarios.

Como se ilustra en la Figura 18B, las diferentes máquinas anfitrionas pueden generar nuevos pesos entrenados (por ejemplo, basados en regiones de entrenamiento/referencia 1602 como se describió anteriormente) y compartir los nuevos pesos entrenados con un servidor 1800 (por ejemplo, como un servidor de juegos) o, alternativamente, usar un protocolo de intercambio de igual a igual. Un componente de gestión de aprendizaje automático 1810 en el servidor genera un conjunto de pesos combinados 1805 usando los nuevos pesos recibidos de cada una de las máquinas anfitrionas. Los pesos combinados 1805, por ejemplo, pueden ser un promedio generado a partir de los nuevos pesos y actualizado continuamente como se describe en este documento. Una vez generados, las copias de los pesos combinados 1605A-C pueden transmitirse y almacenarse en cada una de las máquinas anfitrionas 1820-1821 que luego pueden usar los pesos combinados como se describe en este documento para realizar operaciones de eliminación de ruido.

Este mecanismo de actualización de circuito semicerrado puede ser utilizado por el fabricante de hardware. Por ejemplo, la red de referencia puede incluirse como parte del controlador distribuido por el fabricante de hardware. A medida que el controlador genera nuevos datos de entrenamiento utilizando las técnicas descritas en este documento y los envía continuamente al fabricante de hardware, el fabricante de hardware utiliza esta información para seguir mejorando sus implementaciones de aprendizaje automático para la próxima actualización del controlador.

En una implementación (por ejemplo, en el renderizado de películas por lotes en una granja de renderizado), el renderizador transmite las regiones de entrenamiento recién generadas a un servidor o base de datos dedicado (en la granja de renderizado de ese estudio) que agrega estos datos de múltiples nodos de renderizado a lo largo del tiempo. Un proceso independiente en una máquina independiente mejora continuamente la red de eliminación de ruido dedicada del estudio, y los nuevos trabajos de renderizado siempre utilizan la última red entrenada.

En la **Figura 19** se ilustra un procedimiento de acuerdo con una realización de la invención. El procedimiento puede implementarse en las arquitecturas descritas en este documento, pero no se limita a ningún sistema en particular o arquitectura de procesamiento de gráficos.

En 1901, como parte de la fase de entrenamiento inicial, se generan datos de imágenes de bajo recuento de muestras y datos de imágenes de alto recuento de muestras para una pluralidad de fotogramas de imagen. En 1902, se entrena un motor de eliminación de ruido de aprendizaje automático utilizando los datos de imagen de alto/bajo recuento de muestras. Por ejemplo, un conjunto de pesos de redes neuronales convolucionales asociados con características de píxeles se pueden actualizar de acuerdo con el entrenamiento. Sin embargo, se puede utilizar cualquier arquitectura de aprendizaje automático.

En 1903, en tiempo de ejecución, se generan fotogramas de imagen de bajo recuento de muestras junto con al menos una región de referencia que tiene un alto recuento de muestras. En 1904, el motor de aprendizaje automático y/o la lógica de entrenamiento independiente (por ejemplo, el módulo de entrenamiento en segundo plano 1700) utilizan la región de referencia de alto recuento de muestras para refinar continuamente el entrenamiento del motor de aprendizaje automático. Por ejemplo, la región de referencia de alto recuento de muestras se utiliza en combinación con una porción correspondiente de la imagen de bajo recuento de muestras para continuar enseñándole al motor de aprendizaje automático 1904 cómo realizar la eliminación de ruido de manera más efectiva. En una implementación de la RNC, por ejemplo, esto puede implicar actualizar los pesos asociados con la RNC.

Se pueden implementar múltiples variaciones de los ejemplos descritos anteriormente, tal como la manera en que se configura el bucle de retroalimentación al motor de aprendizaje automático, las entidades que generan los datos de entrenamiento, la manera en que los datos de entrenamiento se devuelven al motor de entrenamiento y cómo se proporciona la red mejorada a los motores de renderizado. Además, si bien los ejemplos descritos anteriormente realizan un entrenamiento continuo utilizando una única región de referencia, se puede utilizar cualquier cantidad de regiones de referencia. Además, como se mencionó anteriormente, las regiones de referencia pueden ser de diferentes tamaños, se pueden utilizar en diferentes números de fotogramas de imagen y se pueden colocar en diferentes ubicaciones dentro de los fotogramas de imagen utilizando diferentes técnicas (por ejemplo, aleatorias, de acuerdo con un patrón predeterminado, etc.).

Además, si bien se describe una red neuronal convolucional (RNC) como un ejemplo de un motor de aprendizaje automático 1600, los principios subyacentes de la invención se pueden implementar utilizando cualquier forma de motor de aprendizaje automático que sea capaz de refinar continuamente sus resultados utilizando nuevos datos de entrenamiento. A modo de ejemplo, y no de limitación, otras implementaciones de aprendizaje automático incluyen el procedimiento de grupo de manejo de datos (GMDH), memoria de corto plazo larga, computación de reservorio profundo, redes de creencias profundas, redes de apilamiento profundo tensorial y redes de codificación predictiva profunda, por nombrar algunas.

En los ejemplos, el término "motor" o "módulo" o "lógica" puede referirse a, ser parte de, o incluir un circuito integrado de aplicación específica (ASIC), un circuito electrónico, un procesador (compartido, dedicado o grupal) y/o una memoria (compartida, dedicada o grupal) que ejecutan uno o más programas de software o firmware, un circuito lógico combinacional y/u otros componentes adecuados que proporcionan la funcionalidad descrita. En los ejemplos, un motor, módulo o lógica se puede implementar en firmware, hardware, software o cualquier combinación de firmware, hardware y software.

Las realizaciones de la invención incluyen varios pasos, que se han descrito anteriormente. Los pasos pueden estar incorporados en instrucciones ejecutables por máquina que pueden usarse para hacer que un procesador de propósito general o de propósito especial realice los pasos. Alternativamente, estos pasos pueden ser realizados por componentes de hardware específicos que contienen lógica cableada para realizar los pasos, o por cualquier combinación de componentes de computadora programados y componentes de hardware personalizados.

Como se describe en el presente documento, las instrucciones pueden referirse a configuraciones específicas de hardware, tales como circuitos integrados de aplicación específica (ASIC) configurados para realizar ciertas operaciones o que tienen una funcionalidad predeterminada o instrucciones de software almacenadas en la memoria incorporadas en un medio legible por computadora no transitorio. Por lo tanto, las técnicas mostradas en las figuras se pueden implementar utilizando código y datos almacenados y ejecutados en uno o más dispositivos electrónicos (por ejemplo, una estación final, un elemento de red, etc.). Dichos dispositivos electrónicos almacenan y comunican (internamente y/o con otros dispositivos electrónicos a través de una red) código y datos utilizando medios legibles por máquinas informáticas, tales como medios de almacenamiento no transitorios legibles por máquinas informáticas (por ejemplo, discos magnéticos; discos ópticos de memoria de acceso aleatorio; memoria de solo lectura; dispositivos de memoria flash; memoria de cambio de fase) y medios de comunicación transitorios legibles por máquinas informáticas (por ejemplo, señales eléctricas, ópticas, acústicas u otras formas de señales propagadas, tales como ondas portadoras, señales infrarrojas, señales digitales, etc.).

Además, dichos dispositivos electrónicos normalmente incluyen un conjunto de uno o más procesadores acoplados a uno o más componentes adicionales, tales como uno o más dispositivos de almacenamiento (medios de almacenamiento legibles por máquina no transitorios), dispositivos de entrada/salida de usuario (por ejemplo, un teclado, una pantalla táctil y/o una pantalla) y conexiones de red. El acoplamiento del conjunto de procesadores y otros componentes se realiza normalmente a través de uno o más buses y puentes (también denominados

controladores de bus). El dispositivo de almacenamiento y las señales que transportan el tráfico de red representan respectivamente uno o más medios de almacenamiento legibles por máquina y medios de comunicación legibles por máquina. Así, el dispositivo de almacenamiento de un dispositivo electrónico dado, normalmente almacena código y/o datos para su ejecución en el conjunto de uno o más procesadores de ese dispositivo electrónico. Por supuesto, una o más partes de una realización de la invención se implementan utilizando diferentes combinaciones de software, firmware y/o hardware. A lo largo de esta descripción detallada, con fines explicativos, se expusieron numerosos detalles específicos con el fin de proporcionar una comprensión completa de la presente invención.

REIVINDICACIONES

1. Un aparato que comprende:

un motor de renderizado de gráficos de una unidad de procesamiento de gráficos, GPU (1430), configurado para renderizar una primera pluralidad de imágenes durante el tiempo de ejecución en un primer recuento de muestras; y
al menos un procesador (200, 1030) configurado para:
ejecutar el primer código de programa para realizar la eliminación de ruido de imágenes mediante un motor de aprendizaje automático (1500, 1600, 1810), en donde el motor de aprendizaje automático (1500, 1600, 1810) está configurado para eliminar el ruido de cada una de la primera pluralidad de imágenes durante el tiempo de ejecución utilizando pesos entrenados (1605);
generar, durante el tiempo de ejecución, una o más nuevas regiones de referencia (1602) en una o más de la primera pluralidad de imágenes en un segundo recuento de muestras mayor que el primer recuento de muestras;
representar colores de píxeles de referencia de cada nueva región de referencia (1602) y entradas de píxeles de referencia ruidosos de una región de bajo recuento de muestras correspondiente; y
realizar un entrenamiento en tiempo de ejecución del motor de aprendizaje automático (1500, 1600, 1810) utilizando una o más nuevas regiones de referencia (1602), en donde el entrenamiento en tiempo de ejecución comprende una evaluación de una o más nuevas regiones de referencia (1602) en combinación con la región de bajo recuento de muestras correspondiente en al menos una de la primera pluralidad de imágenes para actualizar los pesos entrenados (1605) en consecuencia.

2. El aparato de la reivindicación 1, en donde el al menos un procesador (200, 1030) comprende al menos un procesador (200, 1030) de la GPU (1430) o al menos un procesador anfitrión (200, 1030).

3. El aparato de la reivindicación 1 o 2, en donde el al menos un procesador (200, 1030) está configurado para ejecutar un segundo código de programa para entrenar inicialmente el motor de aprendizaje automático (1500, 1600, 1810) para realizar la eliminación de ruido de imagen.

4. El aparato de la reivindicación 1 o 2, en donde el motor de aprendizaje automático (1500, 1600, 1810) comprende una pluralidad de unidades de procesamiento interconectadas de la GPU (1430).

5. El aparato de la reivindicación 4, en donde el motor de aprendizaje automático (1500, 1600, 1810) comprende una red neuronal convolucional, RNC.

6. Un método que comprende:
renderizar, utilizando un motor de renderizado de gráficos de una unidad de procesamiento de gráficos, GPU (1430), una primera pluralidad de imágenes durante el tiempo de ejecución en un primer recuento de muestras;
eliminar el ruido de cada una de la primera pluralidad de imágenes durante el tiempo de ejecución mediante un motor de aprendizaje automático (1500, 1600, 1810) utilizando pesos entrenados (1605);
generar, durante el tiempo de ejecución, una o más nuevas regiones de referencia (1602) en una o más de la primera pluralidad de imágenes en un segundo recuento de muestras mayor que el primer recuento de muestras;
representar colores de píxeles de referencia de cada nueva región de referencia (1602) y entradas de píxeles de referencia ruidosos de una región de bajo recuento de muestras correspondiente; y
realizar un entrenamiento en tiempo de ejecución del motor de aprendizaje automático (1500, 1600, 1810) utilizando una o más nuevas regiones de referencia (1602), en donde el entrenamiento en tiempo de ejecución comprende una evaluación de una o más nuevas regiones de referencia (1602) en combinación con la región de bajo recuento de muestras correspondiente en al menos una de la primera pluralidad de imágenes para actualizar los pesos entrenados (1605) en consecuencia.

7. El método de la reivindicación 6, que comprende, además:
realizar el entrenamiento inicial del motor de aprendizaje automático (1500, 1600, 1810) para realizar la eliminación de ruido de imagen.

8. El método de la reivindicación 6, en donde el motor de aprendizaje automático (1500, 1600, 1810) comprende una red neuronal convolucional, RNC.

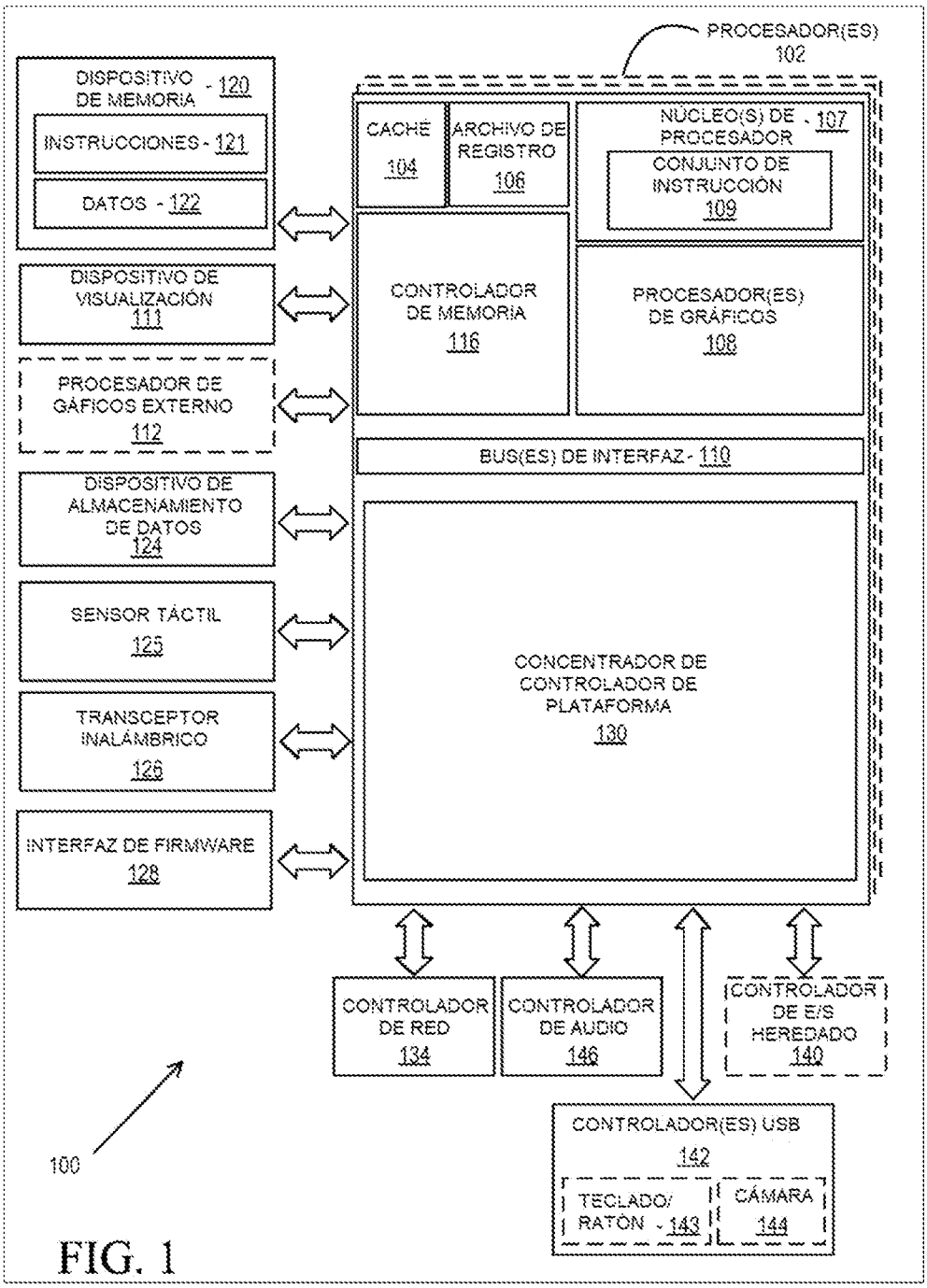

PROCESADOR(ES)
102
DISPOSITIVO DE MEMORIA - 120
INSTRUCCIONES - 121
DATOS - 122
CACHÉ
104
ARCHIVO DE REGISTRO
106
NÚCLEO(S) DE PROCESADOR - 107
CONJUNTO DE INSTRUCCIÓN
109
DISPOSITIVO DE VISUALIZACIÓN
111
CONTROLADOR DE MEMORIA
116
PROCESADOR(ES) DE GRÁFICOS
108
PROCESADOR DE GRÁFICOS EXTERNO
112
BUS(ES) DE INTERFAZ - 110
DISPOSITIVO DE ALMACENAMIENTO DE DATOS
124
SENSOR TÁCTIL
125
CONCENTRADOR DE CONTROLADOR DE PLATAFORMA
130
TRANSCEPTOR INALÁMBRICO
126
INTERFAZ DE FIRMWARE
128
CONTROLADOR DE RED
134
CONTROLADOR DE AUDIO
146
CONTROLADOR DE E/S HEREDADO
140
CONTROLADOR(ES) USB
142
TECLADO/ RATÓN - 143
CÁMARA
144
100

FIG. 1

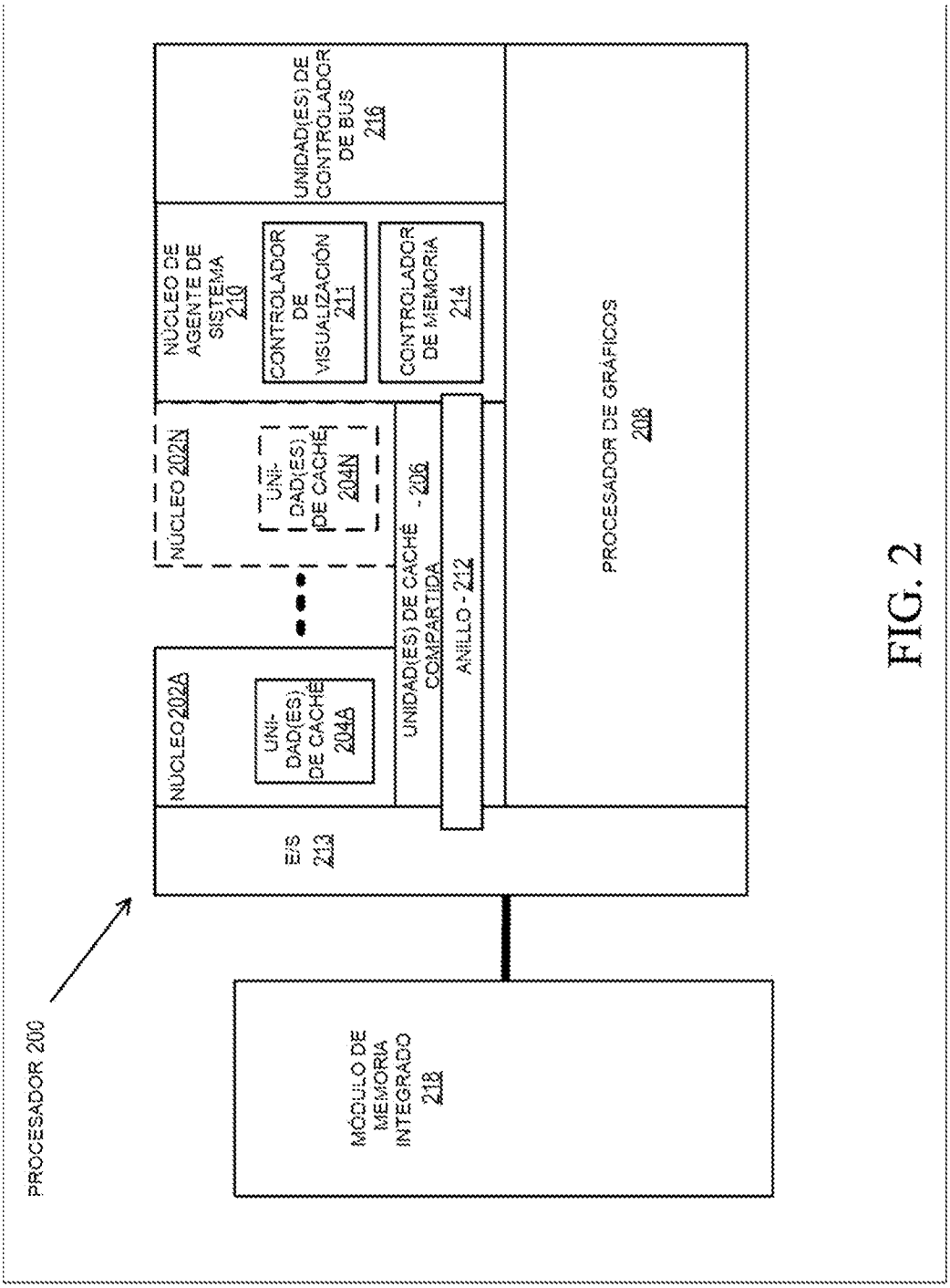

PROCESADOR 200
MÓDULO DE MEMORIA INTEGRADO 218
E/S 213
NÚCLEO 202A
UNI-DAD(ES) DE CACHÉ 204A
NÚCLEO 202N
UNI-DAD(ES) DE CACHÉ 204N
UNIDAD(ES) DE CACHÉ COMPARTIDA - 206
ANILLO - 212
NÚCLEO DE AGENTE DE SISTEMA 210
CONTROLADOR DE VISUALIZACIÓN 211
CONTROLADOR DE MEMORIA 214
UNIDAD(ES) DE CONTROLADOR DE BUS 216
PROCESADOR DE GRÁFICOS 208

FIG. 2

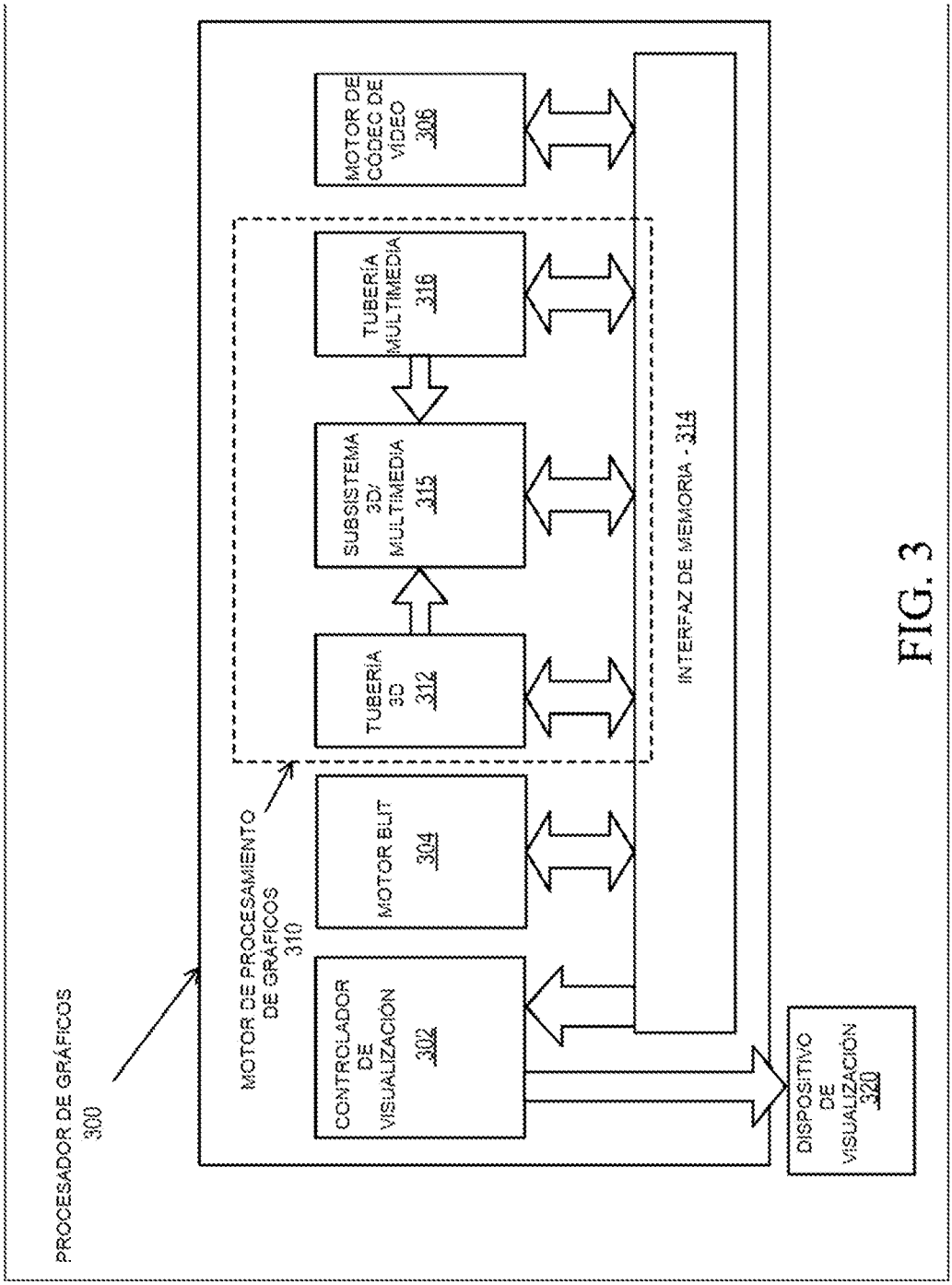

PROCESADOR DE GRÁFICOS
300
MOTOR DE PROCESAMIENTO DE GRÁFICOS
310
CONTROLADOR DE VISUALIZACIÓN
302
MOTOR BLIT
304
TUBERÍA 3D
312
SUBSISTEMA 3D/ MULTIMEDIA
315
TUBERÍA MULTIMEDIA
316
MOTOR DE CÓDEC DE VÍDEO
306
INTERFAZ DE MEMORIA - 314
DISPOSITIVO DE VISUALIZACIÓN
320

FIG. 3

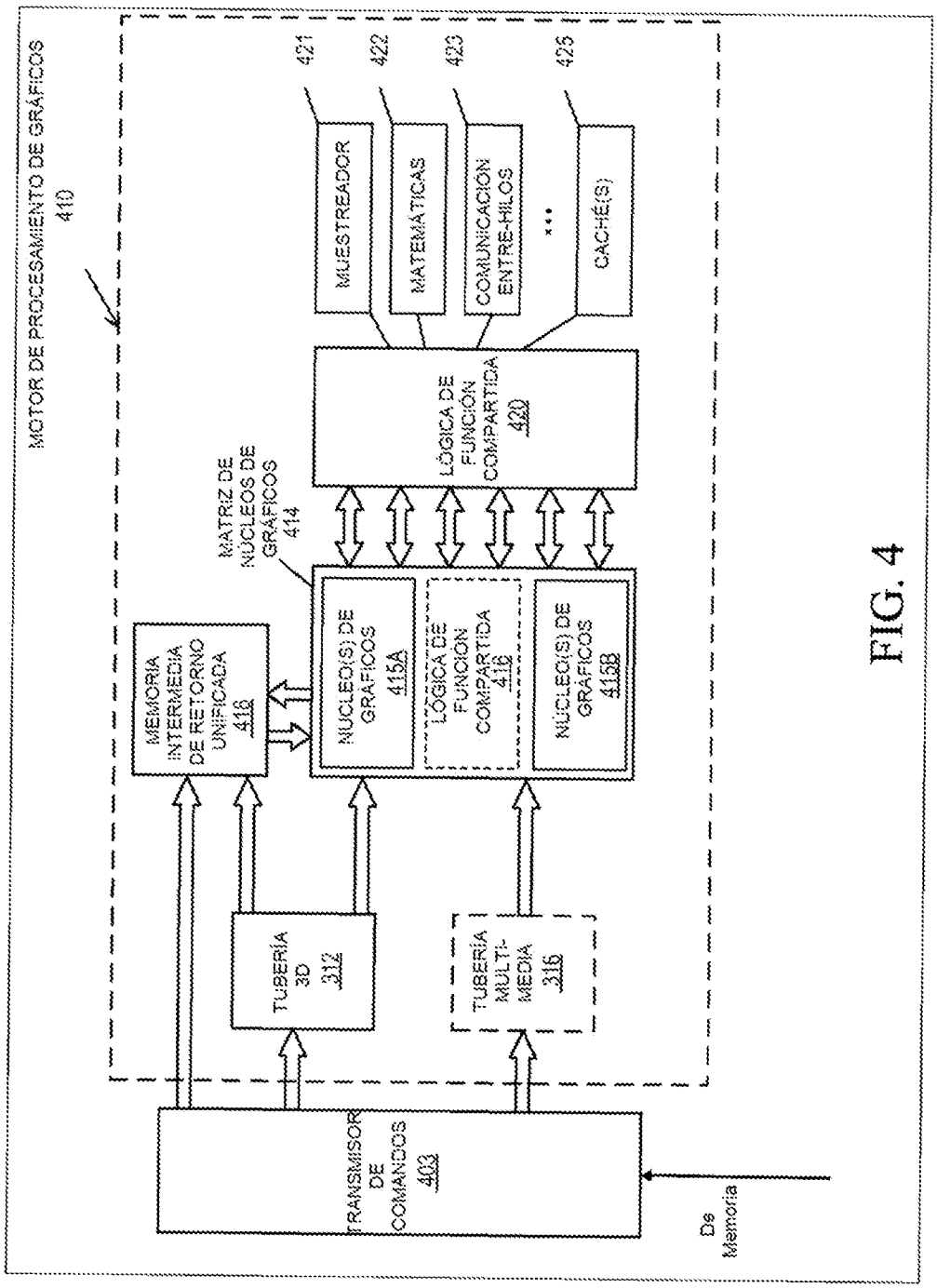

MOTOR DE PROCESAMIENTO DE GRÁFICOS
410
421
422
423
425
MUESTREADOR
MATEMÁTICAS
COMUNICACIÓN ENTRE-HILOS
...
CACHÉ(S)
LÓGICA DE FUNCIÓN COMPARTIDA
420
MATRIZ DE NÚCLEOS DE GRÁFICOS
414
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
418
NÚCLEO(S) DE GRÁFICOS
415A
LÓGICA DE FUNCIÓN COMPARTIDA
416
NÚCLEO(S) DE GRÁFICOS
415B
TUBERÍA 3D
312
TUBERÍA MULTI-MEDIA
316
TRANSMISOR DE COMANDOS
403
De Memoria

FIG. 4

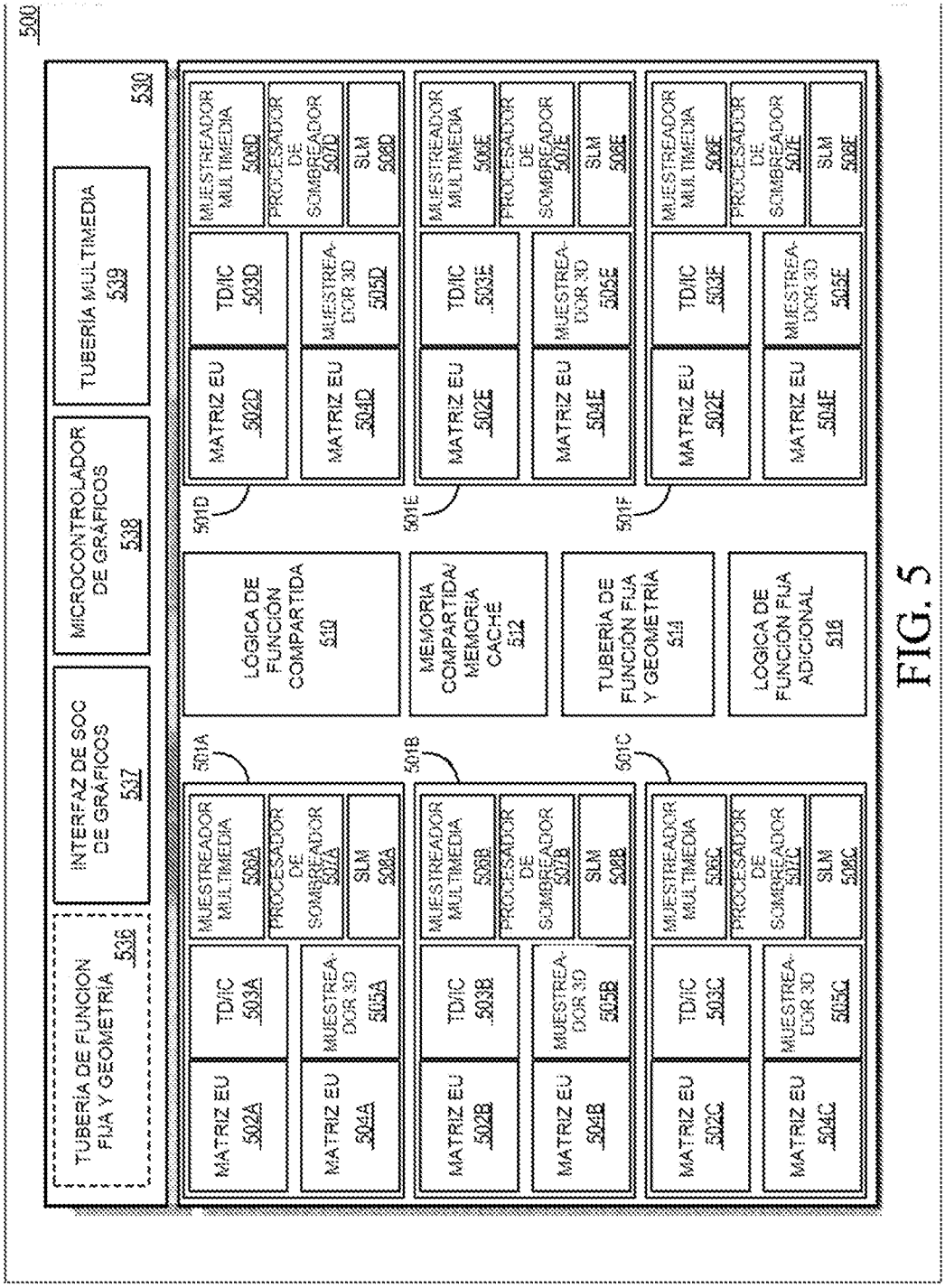

500
TUBERÍA DE FUNCIÓN FIJA Y GEOMETRÍA 536
INTERFAZ DE SOC DE GRÁFICOS 537
MICROCONTROLADOR DE GRÁFICOS 538
TUBERÍA MULTIMEDIA 539
530
501A
501B
501C
501D
501E
501F
MUESTREADOR MULTIMEDIA 508A
PROCESADOR DE SOMBREADOR 507A
SLM 508A
MATRIZ EU 502A
TD/IC 503A
MUESTREADOR 3D 505A
MATRIZ EU 504A
MUESTREADOR MULTIMEDIA 508B
PROCESADOR DE SOMBREADOR 507B
SLM 508B
MATRIZ EU 502B
TD/IC 503B
MUESTREADOR 3D 505B
MATRIZ EU 504B
MUESTREADOR MULTIMEDIA 508C
PROCESADOR DE SOMBREADOR 507C
SLM 508C
MATRIZ EU 502C
TD/IC 503C
MUESTREADOR 3D 505C
MATRIZ EU 504C
LÓGICA DE FUNCIÓN COMPARTIDA 510
MEMORIA COMPARTIDA/ MEMORIA CACHÉ 512
TUBERÍA DE FUNCIÓN FIJA Y GEOMETRÍA 514
LÓGICA DE FUNCIÓN FIJA ADICIONAL 516
MUESTREADOR MULTIMEDIA 508D
PROCESADOR DE SOMBREADOR 507D
SLM 508D
MATRIZ EU 502D
TD/IC 503D
MUESTREADOR 3D 505D
MATRIZ EU 504D
MUESTREADOR MULTIMEDIA 508E
PROCESADOR DE SOMBREADOR 507E
SLM 508E
MATRIZ EU 502E
TD/IC 503E
MUESTREADOR 3D 505E
MATRIZ EU 504E
MUESTREADOR MULTIMEDIA 508E
PROCESADOR DE SOMBREADOR 507E
SLM 508E
MATRIZ EU 502E
TD/IC 503E
MUESTREADOR 3D 505E
MATRIZ EU 504E

FIG. 5

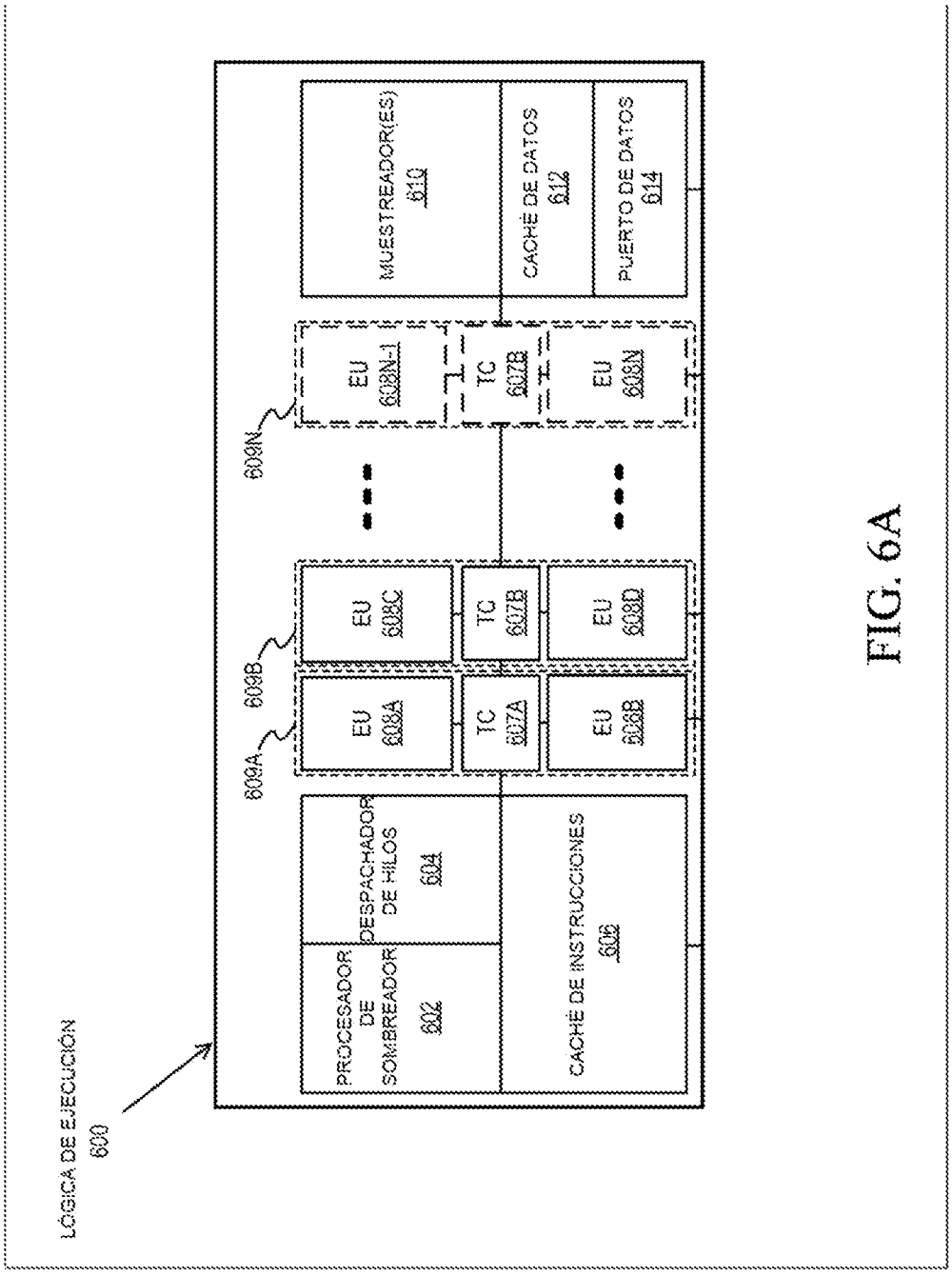

LÓGICA DE EJECUCIÓN
600
PROCESADOR DE SOMBREADOR
602
DESPACHADOR DE HILOS
604
CACHÉ DE INSTRUCCIONES
606
609A
609B
609N
EU 608A
EU 608B
EU 608C
EU 608D
EU 608N-1
EU 608N
TC 607A
TC 607B
TC 607B
MUESTREADOR(ES)
610
CACHÉ DE DATOS
612
PUERTO DE DATOS
614

FIG. 6A

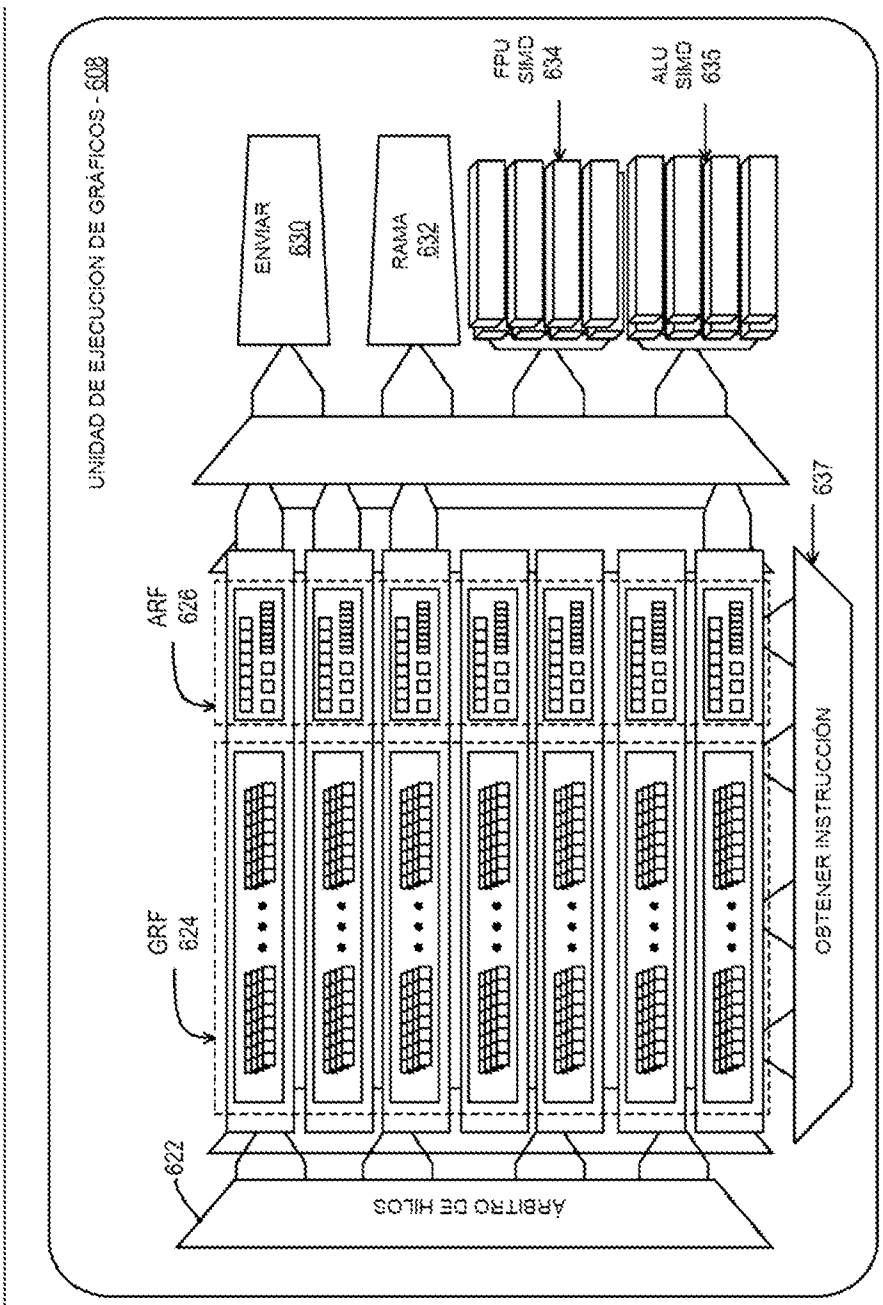

UNIDAD DE EJECUCIÓN DE GRÁFICOS - 608
ENVIAR 630
RAMA 632
FPU SIMD 634
ALU SIMD 635
637
ARF 626
GRF 624
622
ARBITRO DE HILOS
OBTENER INSTRUCCIÓN

FIG. 6B

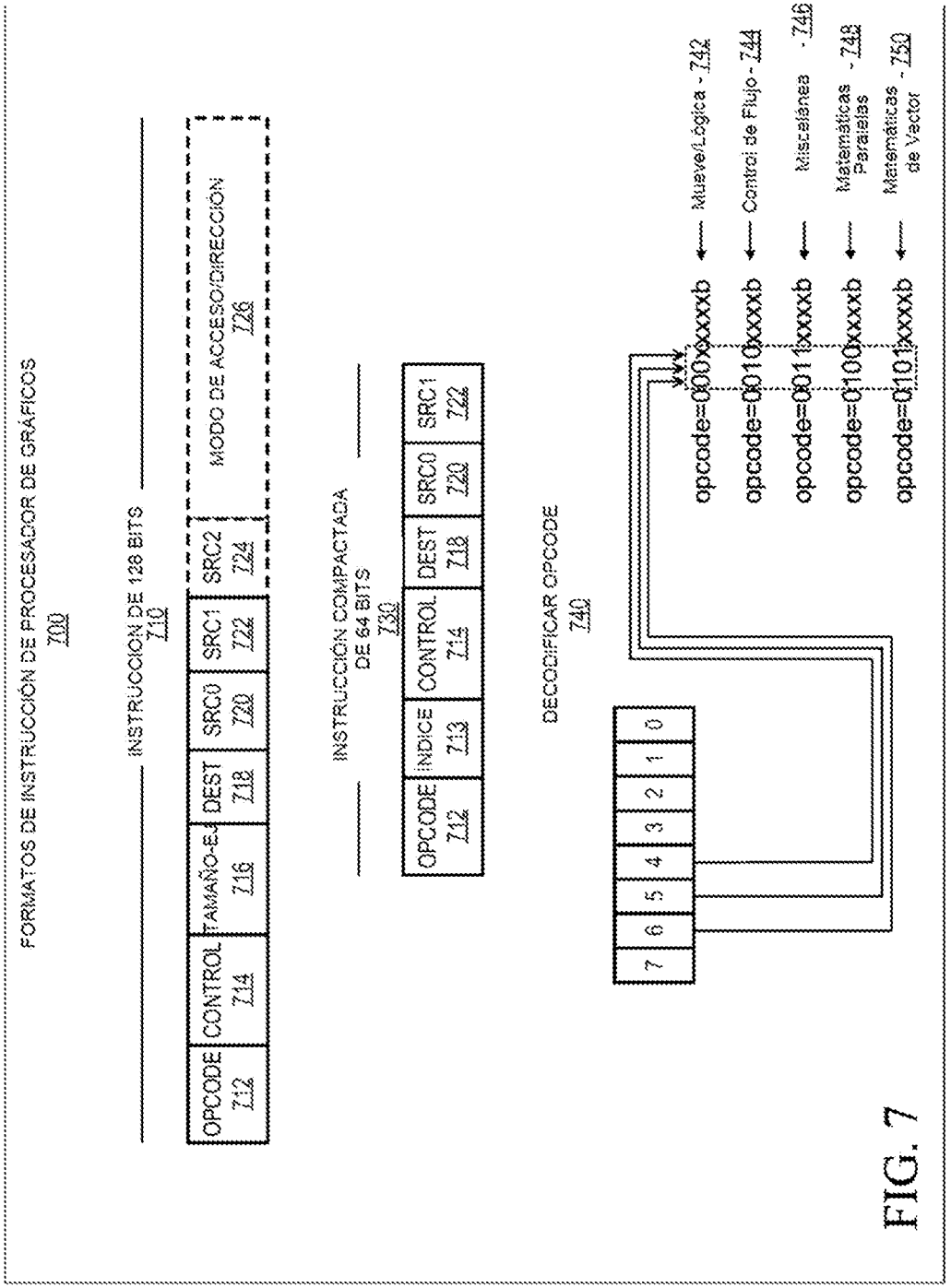

FORMATOS DE INSTRUCCIÓN DE PROCESADOR DE GRÁFICOS
700
INSTRUCCIÓN DE 128 BITS
710
OPCODE Z12
CONTROL Z14
TAMAÑO-E Z16
DEST Z18
SRC0 Z20
SRC1 Z22
SRC2 Z24
MODO DE ACCESO/DIRECCIÓN Z26
INSTRUCCIÓN COMPACTADA DE 64 BITS
730
OPCODE Z12
INDICE Z13
CONTROL Z14
DEST Z18
SRC0 Z20
SRC1 Z22
DECODIFICAR OPCODE
740
7 6 5 4 3 2 1 0
opcode=000xxxxb
opcode=001xxxxb
opcode=010xxxxb
opcode=011xxxxb
opcode=100xxxxb
opcode=101xxxxb
Mueve/Lógica - 742
Control de Flujo - 744
Miscelánea - 746
Matemáticas Paralelas - 748
Matemáticas de Vector - 750

FIG. 7

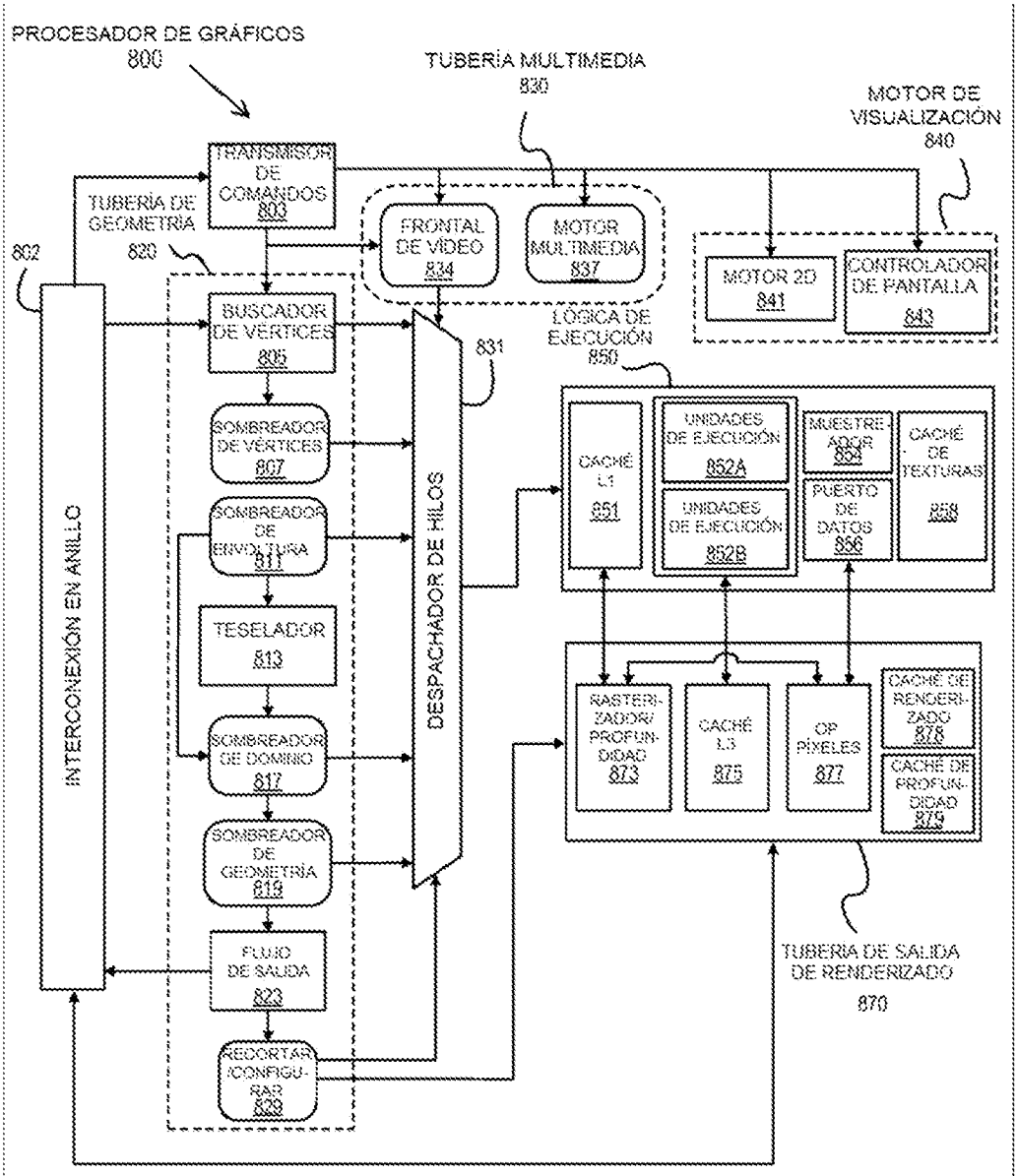
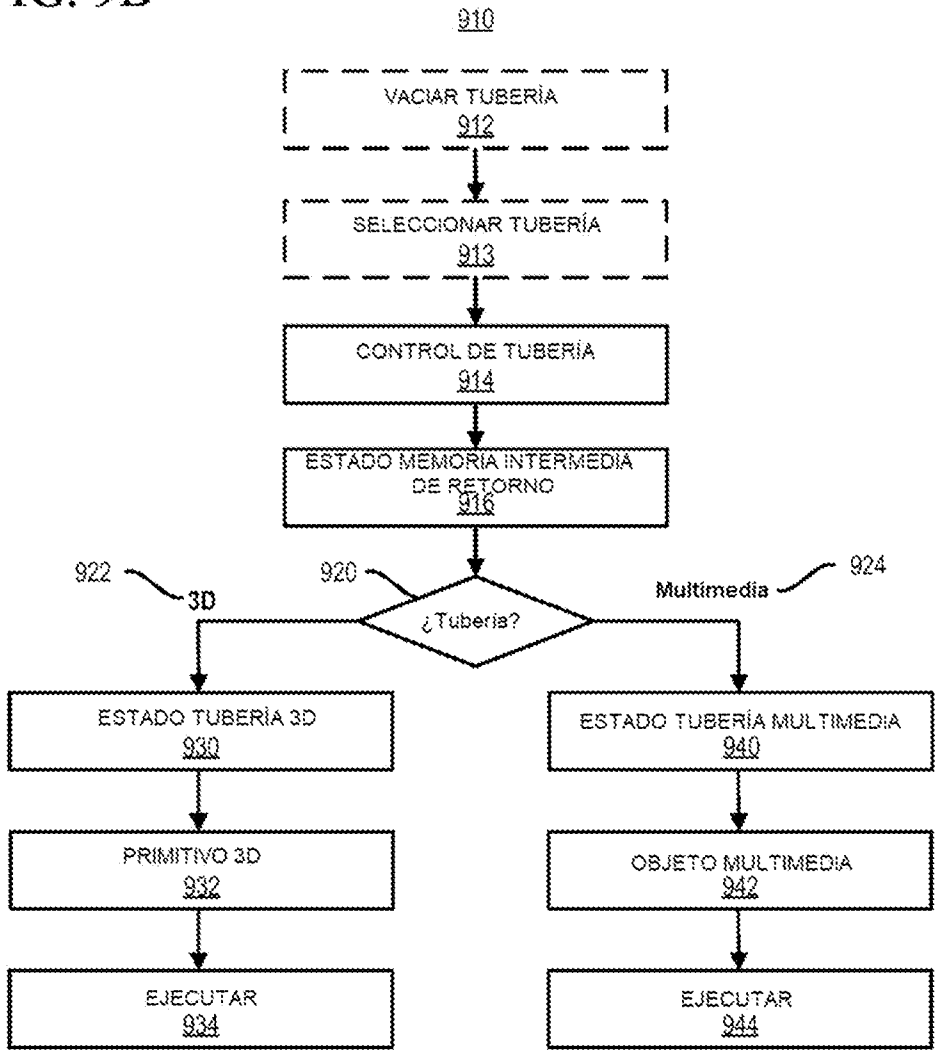

PROCESADOR DE GRÁFICOS
800
TUBERÍA MULTIMEDIA
830
MOTOR DE VISUALIZACIÓN
840
TRANSMISOR DE COMANDOS
803
TUBERÍA DE GEOMETRÍA
820
802
FRONTAL DE VÍDEO
834
MOTOR MULTIMEDIA
837
MOTOR 2D
841
CONTROLADOR DE PANTALLA
843
BUSCADOR DE VÉRTICES
805
LÓGICA DE EJECUCIÓN
850
831
SOMBREADOR DE VÉRTICES
807
SOMBREADOR DE ENVOLTURA
811
TESELADOR
813
SOMBREADOR DE DOMINIO
817
SOMBREADOR DE GEOMETRÍA
819
FLUJO DE SALIDA
823
RECORTAR /CONFIGU-RAR
829
INTERCONEXIÓN EN ANILLO
DESPACHADOR DE HILOS
CACHÉ L1
851
UNIDADES DE EJECUCIÓN
852A
UNIDADES DE EJECUCIÓN
852B
MUESTREA-DOR
854
PUERTO DE DATOS
856
CACHÉ DE TEXTURAS
858
RASTERI-ZADOR/ PROFUN-DIDAD
873
CACHÉ L3
875
OP PÍXELES
877
CACHÉ DE RENDERI-ZADO
878
CACHÉ DE PROFUN-DIDAD
879
TUBERÍA DE SALIDA DE RENDERIZADO
870

FIG. 8

FIG. 9A FORMATO DE COMANDO DE PROCESADOR DE GRÁFICOS

FIG. 9B SECUENCIA DE COMANDOS DE PROCESADOR DE GRÁFICOS



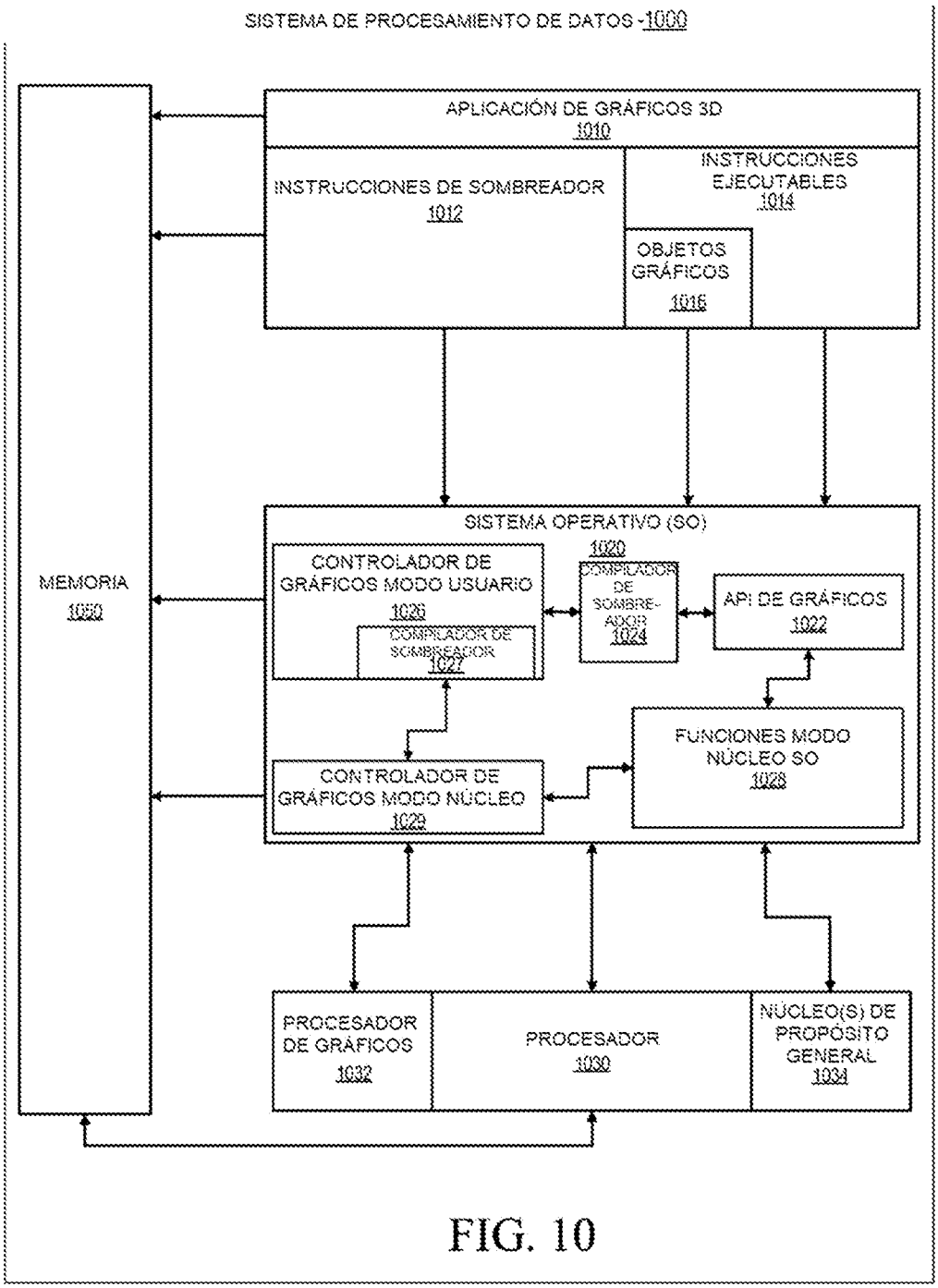

SISTEMA DE PROCESAMIENTO DE DATOS -1000
APLICACIÓN DE GRÁFICOS 3D
1010
INSTRUCCIONES DE SOMBREADOR
1012
INSTRUCCIONES EJECUTABLES
1014
OBJETOS GRÁFICOS
1016
SISTEMA OPERATIVO (SO)
1020
CONTROLADOR DE GRÁFICOS MODO USUARIO
1026
COMPILADOR DE SOMBREADOR
1027
COMPILADOR DE SOMBRE-ADOR
1024
API DE GRÁFICOS
1022
FUNCIONES MODO NÚCLEO SO
1028
CONTROLADOR DE GRÁFICOS MODO NÚCLEO
1029
MEMORIA
1050
PROCESADOR DE GRÁFICOS
1032
PROCESADOR
1030
NÚCLEO(S) DE PROPÓSITO GENERAL
1034

FIG. 10

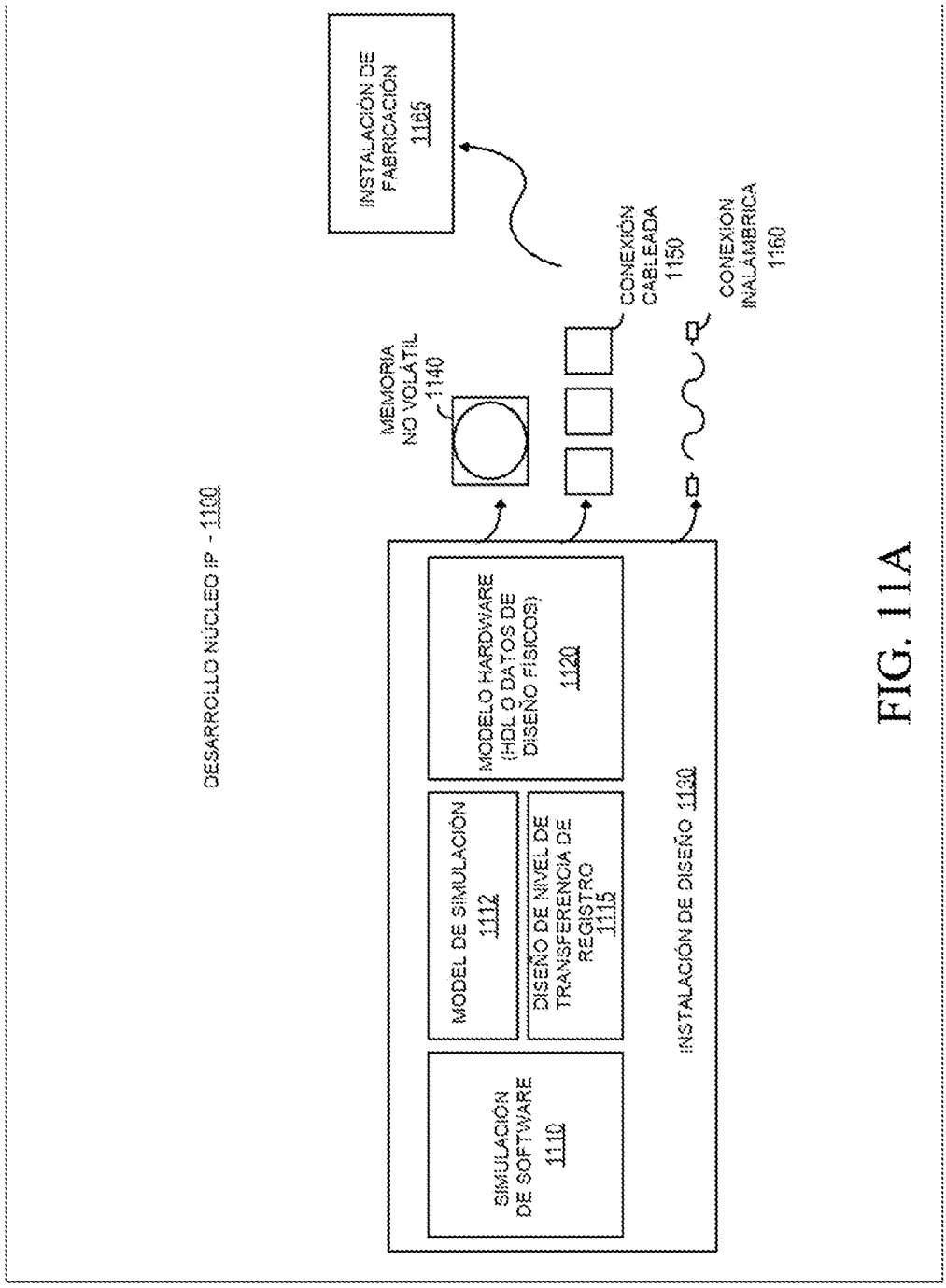

DESARROLLO NÚCLEO IP - 1100
SIMULACIÓN DE SOFTWARE 1110
MODEL DE SIMULACIÓN 1112
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 1115
MODELO HARDWARE (HDL O DATOS DE DISEÑO FÍSICOS) 1120
INSTALACIÓN DE DISEÑO 1130
MEMORIA NO VOLÁTIL 1140
CONEXIÓN CABLEADA 1150
CONEXIÓN INALÁMBRICA 1160
INSTALACIÓN DE FABRICACIÓN 1165

FIG. 11A

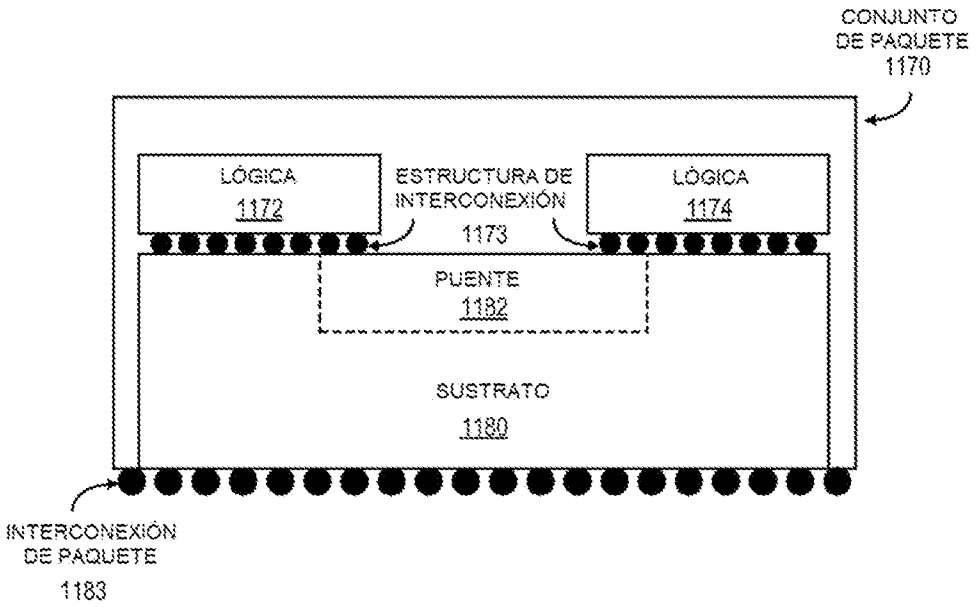

CONJUNTO
DE PAQUETE
1170
LÓGICA
1172
ESTRUCTURA DE
INTERCONEXIÓN
1173
LÓGICA
1174
PUENTE
1182
SUSTRATO
1180
INTERCONEXIÓN
DE PAQUETE
1183

FIG. 11B

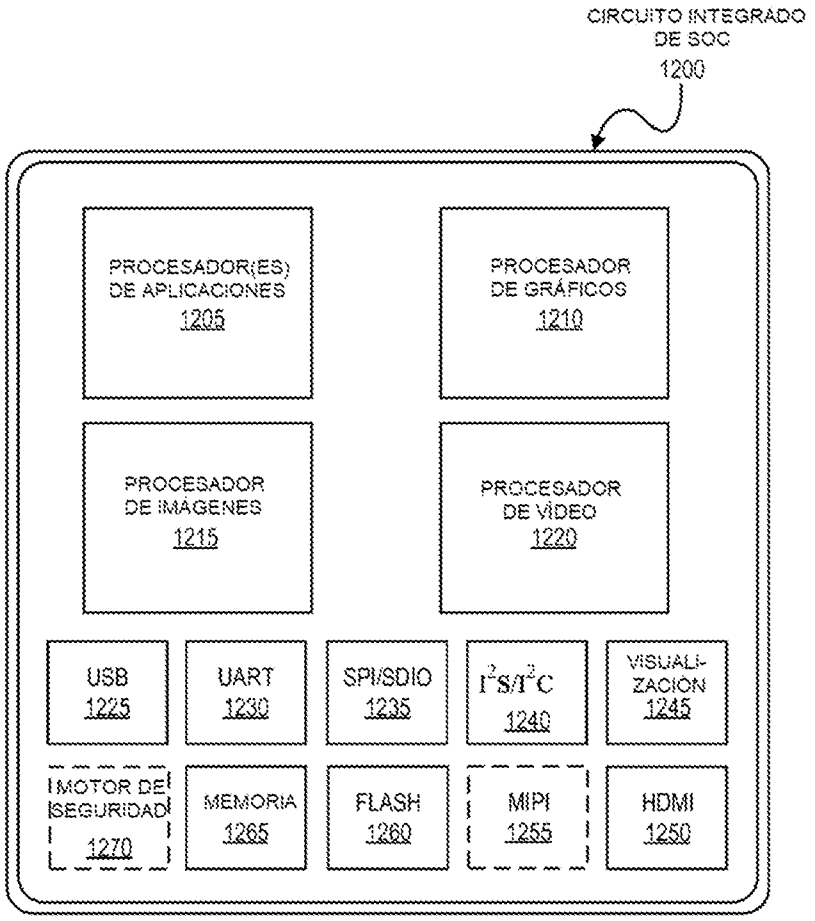

CIRCUITO INTEGRADO DE SOC
1200
PROCESADOR(ES) DE APLICACIONES
1205
PROCESADOR DE GRÁFICOS
1210
PROCESADOR DE IMÁGENES
1215
PROCESADOR DE VÍDEO
1220
USB
1225
UART
1230
SPI/SDIO
1235
I2S/I2C
1240
VISUALI-ZACION
1245
MOTOR DE SEGURIDAD
1270
MEMORIA
1265
FLASH
1260
MIPI
1255
HDMI
1250

FIG. 12

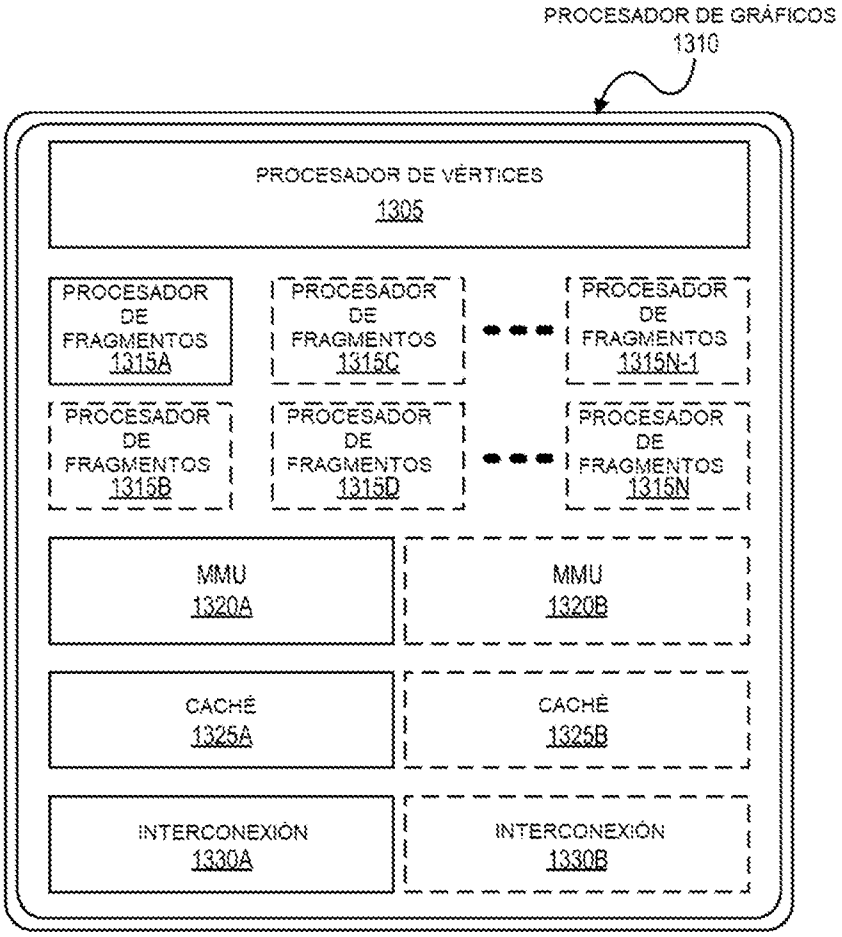

PROCESADOR DE GRÁFICOS
1310
PROCESADOR DE VÉRTICES
1305
PROCESADOR DE FRAGMENTOS
1315A
PROCESADOR DE FRAGMENTOS
1315C
PROCESADOR DE FRAGMENTOS
1315N-1
PROCESADOR DE FRAGMENTOS
1315B
PROCESADOR DE FRAGMENTOS
1315D
PROCESADOR DE FRAGMENTOS
1315N
MMU
1320A
MMU
1320B
CACHÉ
1325A
CACHÉ
1325B
INTERCONEXIÓN
1330A
INTERCONEXIÓN
1330B

FIG. 13A

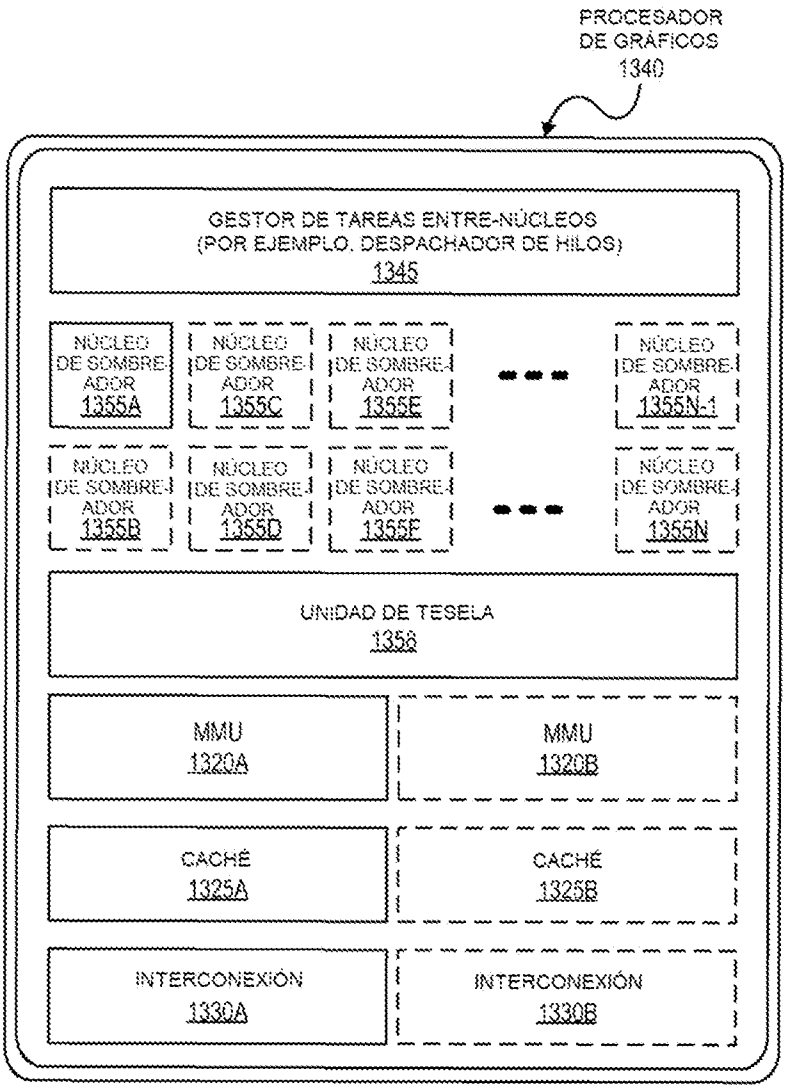

PROCESADOR DE GRÁFICOS
1340
GESTOR DE TAREAS ENTRE-NÚCLEOS (POR EJEMPLO, DESPACHADOR DE HILOS)
1345
NÚCLEO DE SOMBREADOR
1355A
NÚCLEO DE SOMBREADOR
1355C
NÚCLEO DE SOMBREADOR
1355E
NÚCLEO DE SOMBREADOR
1355N-1
NÚCLEO DE SOMBREADOR
1355B
NÚCLEO DE SOMBREADOR
1355D
NÚCLEO DE SOMBREADOR
1355F
NÚCLEO DE SOMBREADOR
1355N
UNIDAD DE TESELA
1358
MMU
1320A
MMU
1320B
CACHÉ
1325A
CACHÉ
1325B
INTERCONEXIÓN
1330A
INTERCONEXIÓN
1330B

FIG. 13B

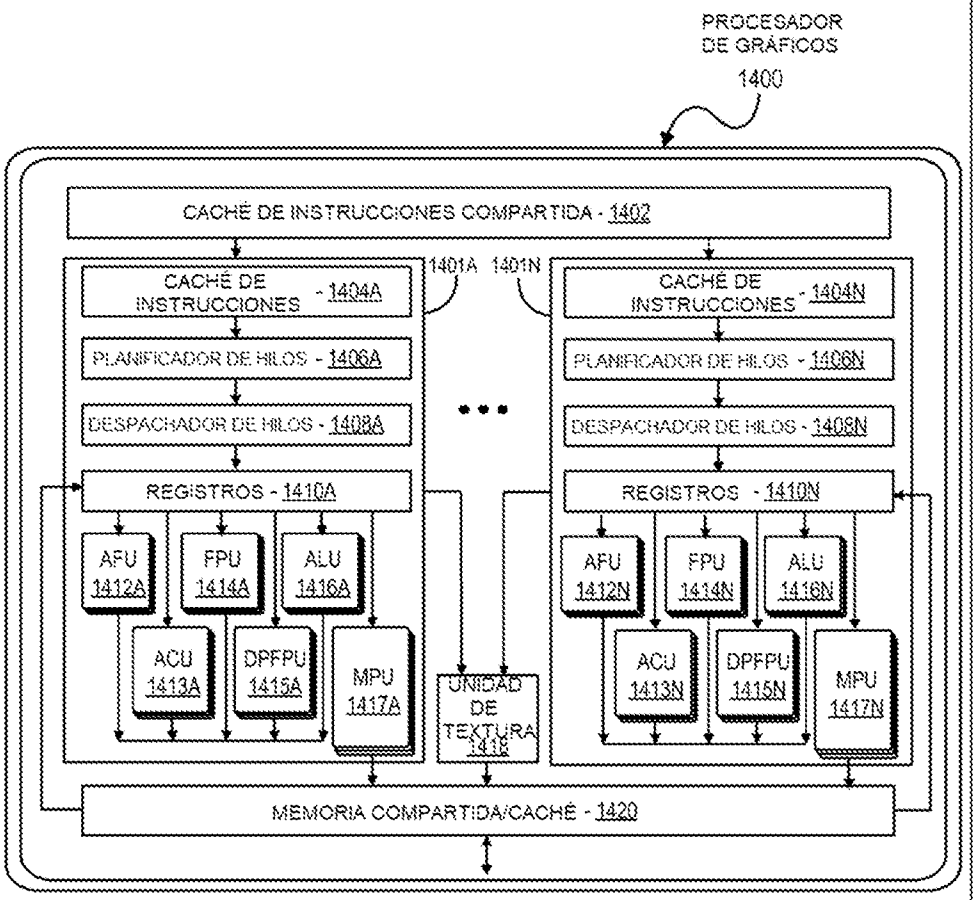

PROCESADOR DE GRÁFICOS
1400
CACHÉ DE INSTRUCCIONES COMPARTIDA - 1402
1401A
1401N
CACHÉ DE INSTRUCCIONES - 1404A
PLANIFICADOR DE HILOS - 1406A
DESPACHADOR DE HILOS - 1408A
REGISTROS - 1410A
AFU 1412A
FPU 1414A
ALU 1416A
ACU 1413A
DPFPU 1415A
MPU 1417A
UNIDAD DE TEXTURA 1418
CACHÉ DE INSTRUCCIONES - 1404N
PLANIFICADOR DE HILOS - 1406N
DESPACHADOR DE HILOS - 1408N
REGISTROS - 1410N
AFU 1412N
FPU 1414N
ALU 1416N
ACU 1413N
DPFPU 1415N
MPU 1417N
MEMORIA COMPARTIDA/CACHÉ - 1420

FIG. 14A

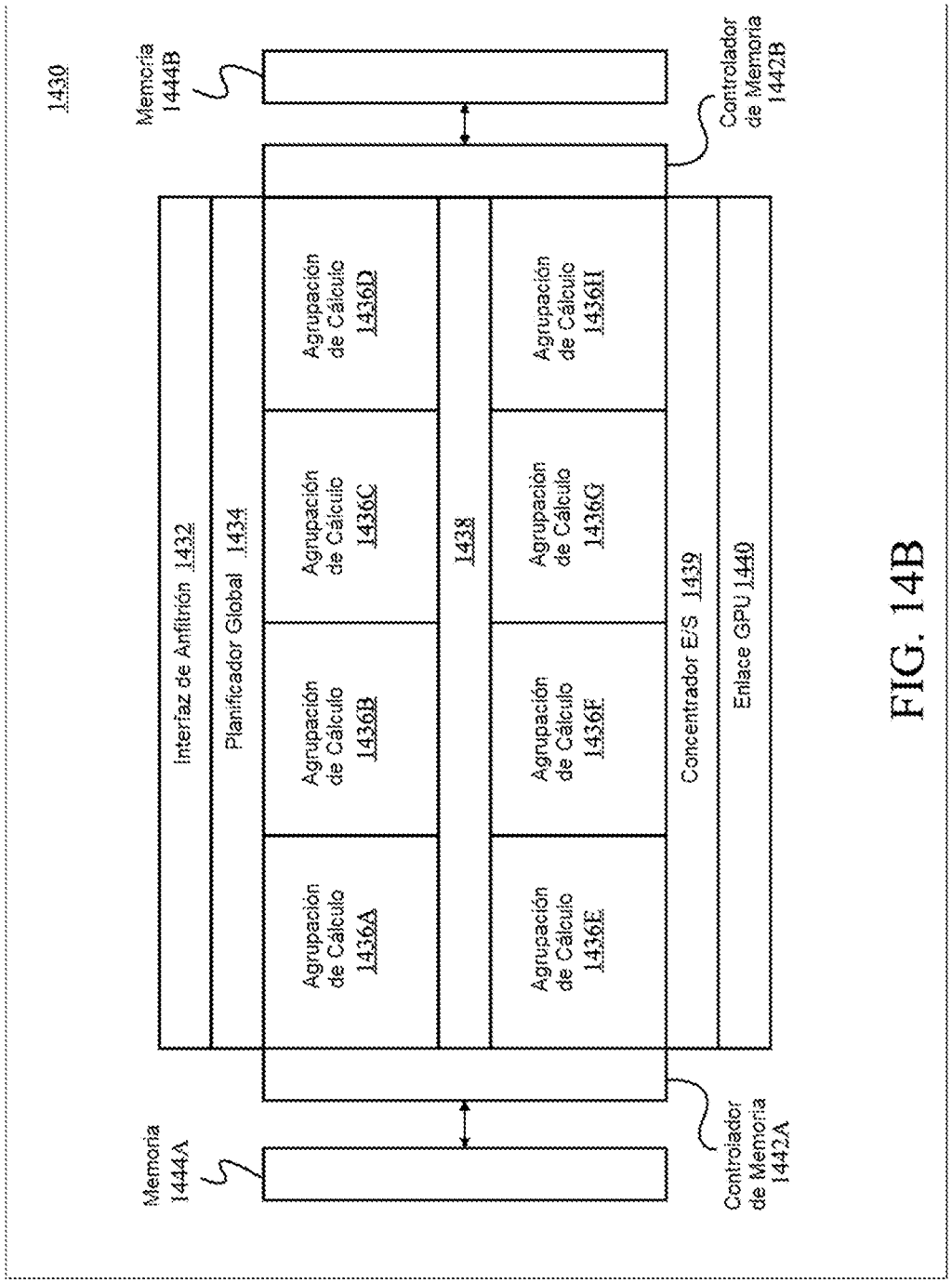

1430
Memoria 1444B
Controlador de Memoria 1442B
Interfaz de Anfitrión 1432
Planificador Global 1434
Agrupación de Cálculo 1436A
Agrupación de Cálculo 1436B
Agrupación de Cálculo 1436C
Agrupación de Cálculo 1436D
1438
Agrupación de Cálculo 1436E
Agrupación de Cálculo 1436F
Agrupación de Cálculo 1436G
Agrupación de Cálculo 1436H
Concentrador E/S 1439
Enlace GPU 1440
Memoria 1444A
Controlador de Memoria 1442A

FIG. 14B

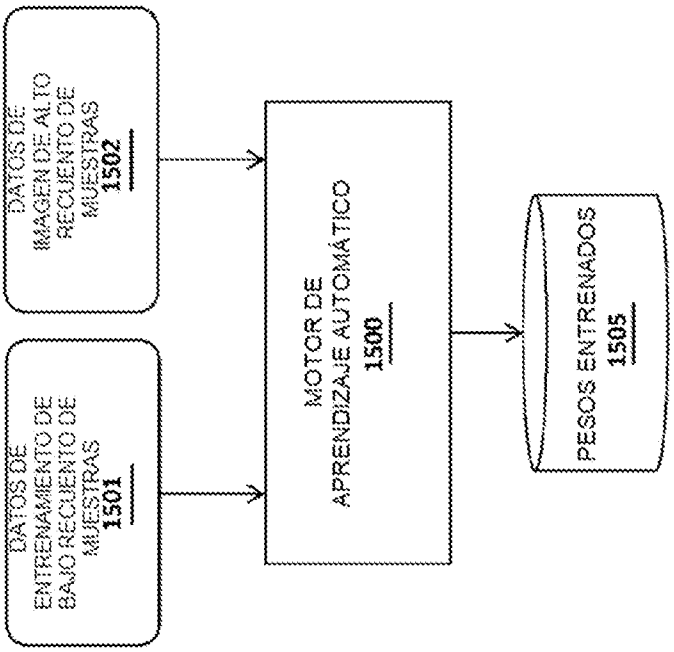

DATOS DE ENTRENAMIENTO DE BAJO RECUENTO DE MUESTRAS
1501
DATOS DE IMAGEN DE ALTO RECUENTO DE MUESTRAS
1502
MOTOR DE APRENDIZAJE AUTOMÁTICO
1500
PESOS ENTRENADOS
1505

FIG. 15

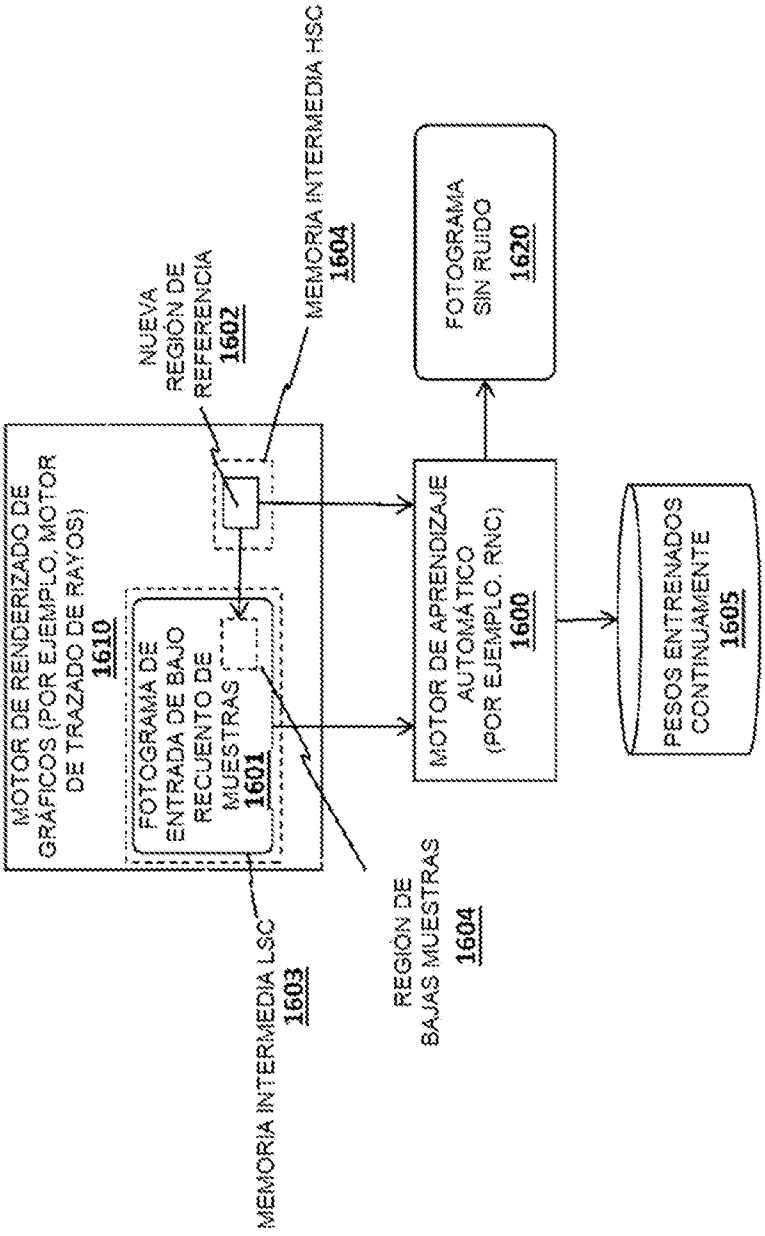

MOTOR DE RENDERIZADO DE GRÁFICOS (POR EJEMPLO, MOTOR DE TRAZADO DE RAYOS)
1610
FOTOGRAMA DE ENTRADA DE BAJO RECUENTO DE MUESTRAS
1601
NUEVA REGIÓN DE REFERENCIA
1602
MEMORIA INTERMEDIA HSC
1604
MEMORIA INTERMEDIA LSC
1603
REGIÓN DE BAJAS MUESTRAS
1604
MOTOR DE APRENDIZAJE AUTOMÁTICO (POR EJEMPLO, RNC)
1600
FOTOGRAMA SIN RUIDO
1620
PESOS ENTRENADOS CONTINUAMENTE
1605

FIG. 16

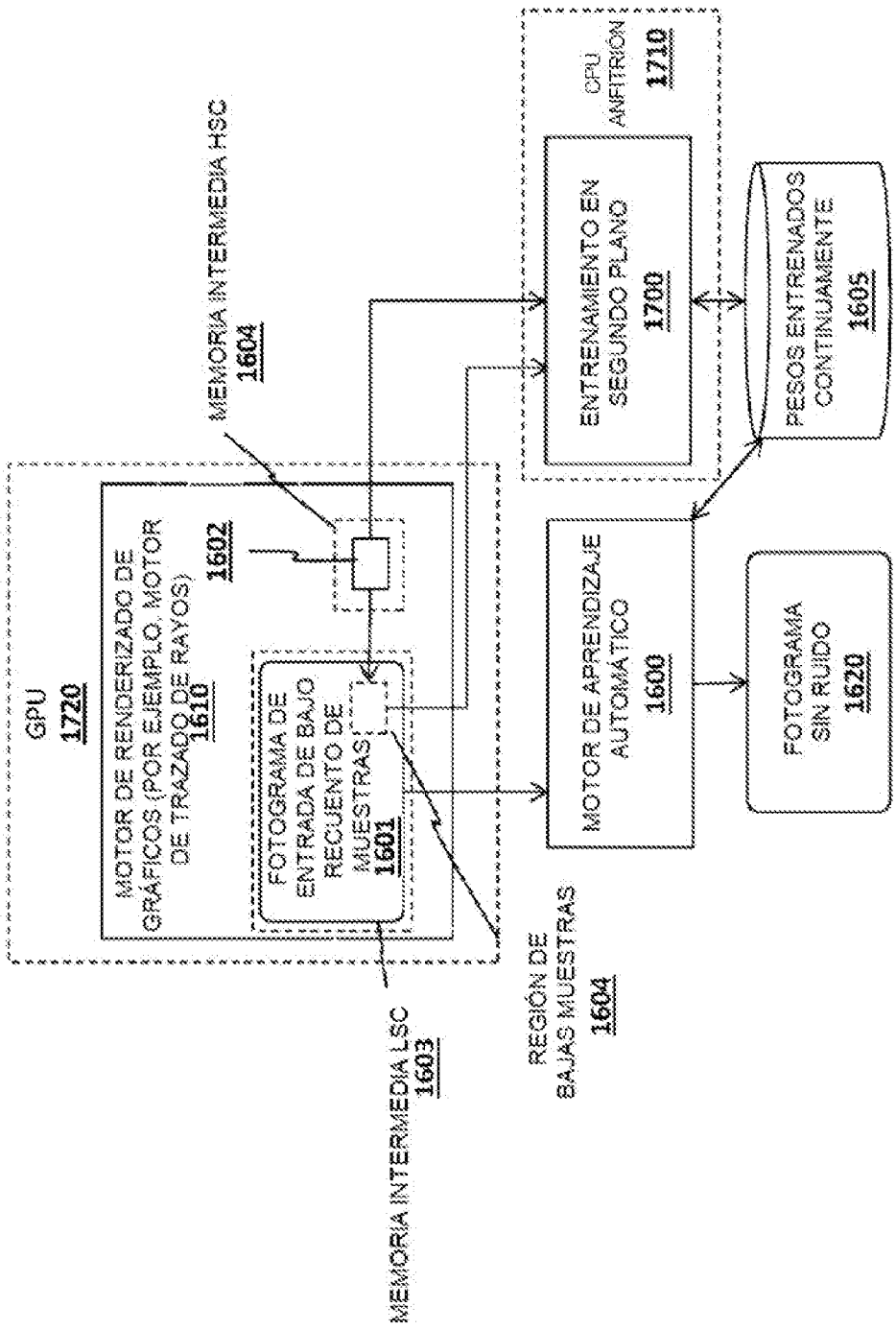

GPU
1720
MOTOR DE RENDERIZADO DE GRÁFICOS (POR EJEMPLO, MOTOR DE TRAZADO DE RAYOS)
1610
1602
FOTOGRAMA DE ENTRADA DE BAJO RECUENTO DE MUESTRAS
1601
MEMORIA INTERMEDIA HSC
1604
MEMORIA INTERMEDIA LSC
1603
REGIÓN DE BAJAS MUESTRAS
1604
CPU ANFITRIÓN
1710
ENTRENAMIENTO EN SEGUNDO PLANO
1700
MOTOR DE APRENDIZAJE AUTOMÁTICO
1600
PESOS ENTRENADOS CONTINUAMENTE
1605
FOTOGRAMA SIN RUIDO
1620

FIG. 17

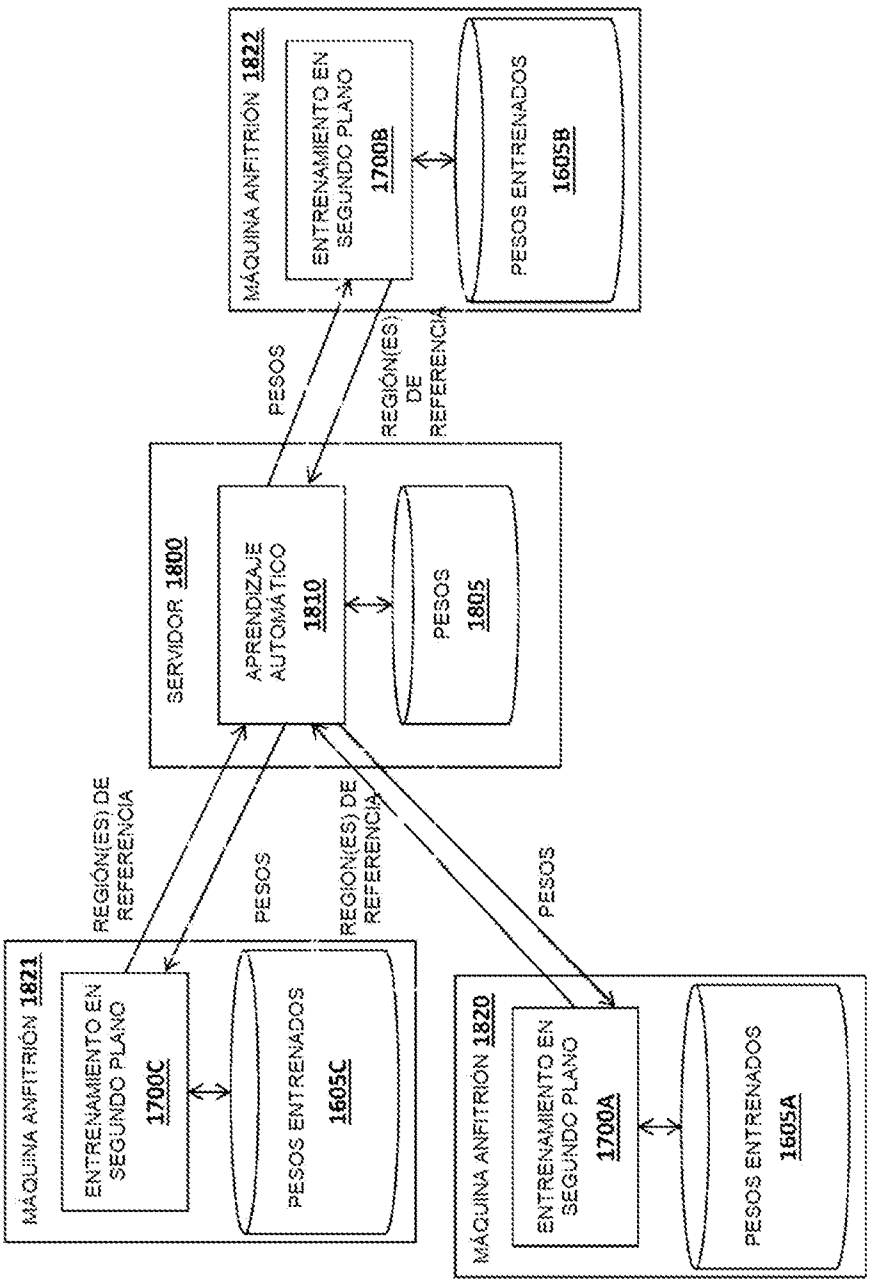

MÁQUINA ANFITRIÓN 1822
ENTRENAMIENTO EN SEGUNDO PLANO 1700B
PESOS ENTRENADOS 1605B
PESOS
REGIÓN(ES) DE REFERENCIA
SERVIDOR 1800
APRENDIZAJE AUTOMÁTICO 1810
PESOS 1805
REGIÓN(ES) DE REFERENCIA
PESOS
REGIÓN(ES) DE REFERENCIA
PESOS
MÁQUINA ANFITRIÓN 1821
ENTRENAMIENTO EN SEGUNDO PLANO 1700C
PESOS ENTRENADOS 1605C
MÁQUINA ANFITRIÓN 1820
ENTRENAMIENTO EN SEGUNDO PLANO 1700A
PESOS ENTRENADOS 1605A

FIG. 18A

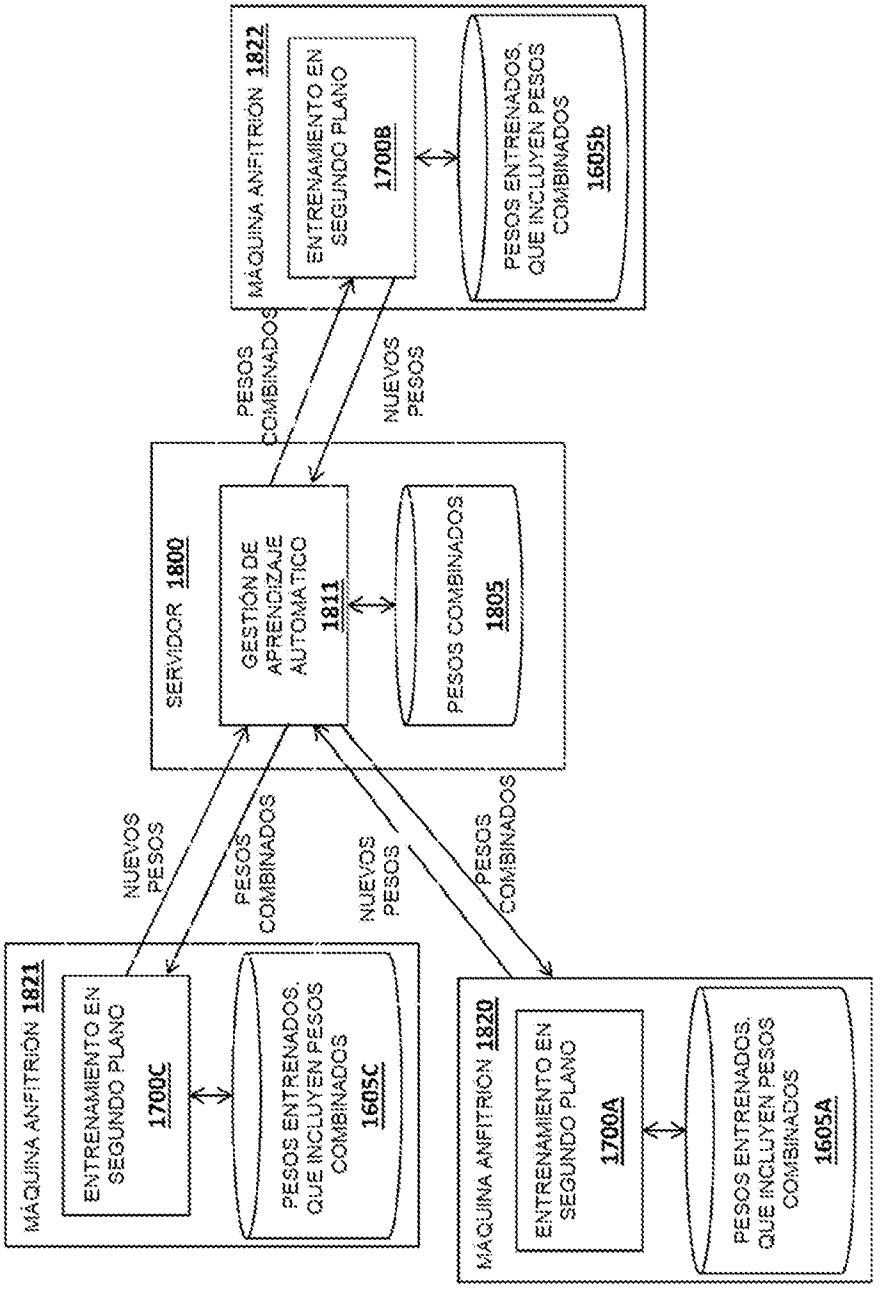

MÁQUINA ANFITRIÓN 1822
ENTRENAMIENTO EN SEGUNDO PLANO 1708B
PESOS ENTRENADOS, QUE INCLUYEN PESOS COMBINADOS 1605b
PESOS COMBINADOS
NUEVOS PESOS
SERVIDOR 1800
GESTIÓN DE APRENDIZAJE AUTOMÁTICO 1811
PESOS COMBINADOS 1805
NUEVOS PESOS
PESOS COMBINADOS
NUEVOS PESOS
PESOS COMBINADOS
MÁQUINA ANFITRIÓN 1821
ENTRENAMIENTO EN SEGUNDO PLANO 1708C
PESOS ENTRENADOS, QUE INCLUYEN PESOS COMBINADOS 1605C
MÁQUINA ANFITRIÓN 1820
ENTRENAMIENTO EN SEGUNDO PLANO 1708A
PESOS ENTRENADOS, QUE INCLUYEN PESOS COMBINADOS 1605A

FIG. 18B

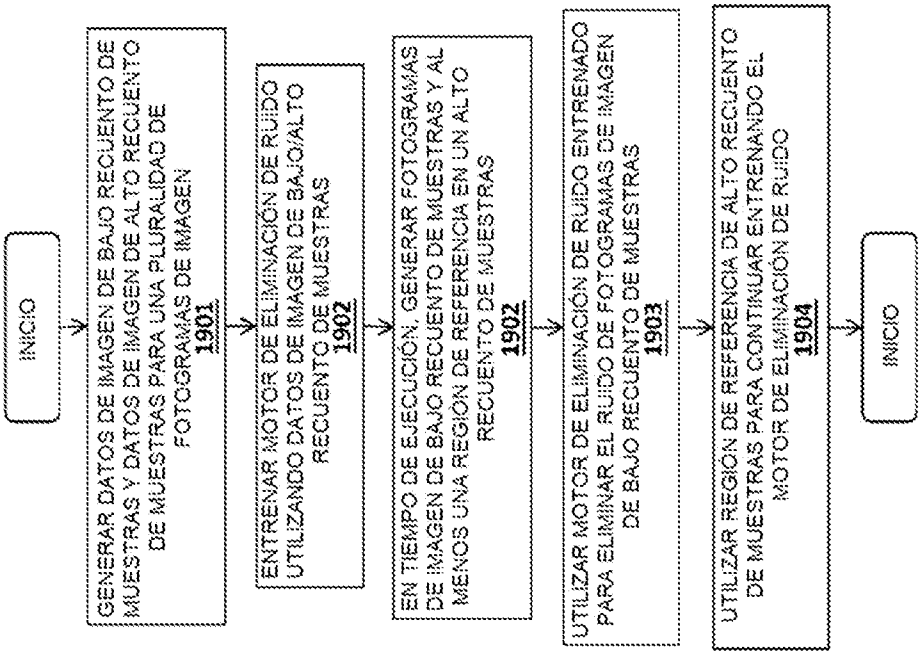

INICIO
GENERAR DATOS DE IMAGEN DE BAJO RECUENTO DE MUESTRAS Y DATOS DE IMAGEN DE ALTO RECUENTO DE MUESTRAS PARA UNA PLURALIDAD DE FOTOGRAMAS DE IMAGEN
1901
ENTRENAR MOTOR DE ELIMINACIÓN DE RUIDO UTILIZANDO DATOS DE IMAGEN DE BAJO/ALTO RECUENTO DE MUESTRAS
1902
EN TIEMPO DE EJECUCIÓN, GENERAR FOTOGRAMAS DE IMAGEN DE BAJO RECUENTO DE MUESTRAS Y AL MENOS UNA REGIÓN DE REFERENCIA EN UN ALTO RECUENTO DE MUESTRAS
1902
UTILIZAR MOTOR DE ELIMINACIÓN DE RUIDO ENTRENADO PARA ELIMINAR EL RUIDO DE FOTOGRAMAS DE IMAGEN DE BAJO RECUENTO DE MUESTRAS
1903
UTILIZAR REGIÓN DE REFERENCIA DE ALTO RECUENTO DE MUESTRAS PARA CONTINUAR ENTRENANDO EL MOTOR DE ELIMINACIÓN DE RUIDO
1904
INICIO

FIG. 19