(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 905 866**

(51) Int. CI.:

G06F 9/451 (2008.01)
G06F 9/455 (2008.01)
G09G 5/395 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **01.03.2018 E 18159474 (8)**

(97) Fecha y número de publicación de la concesión europea: **10.11.2021 EP 3385838**

(54) Título: **Aparato y método para visualización remota y protección de contenido en un entorno de procesamiento gráfico virtualizado**

(30) Prioridad:

07.04.2017 US 201715482535

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
12.04.2022

(73) Titular/es:

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

(72) Inventor/es:

VEMBU, BALAJI;
TANNER, JASON;
RAY, JOYDEEP;
KOKER, ALTUG;
APPU, ABHISHEK R. y
K, PATTABHIRAMAN

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Aparato y método para visualización remota y protección de contenido en un entorno de procesamiento gráfico virtualizado

ANTECEDENTES

Campo de la invención

Esta invención se refiere en general al campo de los procesadores informáticos. Más particularmente, la invención se refiere a un aparato y método para visualización remota y protección de contenido en un entorno de procesamiento gráfico virtualizado.

Descripción de la técnica relacionada

Se han llevado a cabo avances rápidos en la virtualización de unidades gráficas de procesador (GPU). Los entornos de procesamiento gráfico virtualizado se usan, por ejemplo, en la nube de medios, estaciones de trabajo/escritorios remotos, Instrumentación Virtual Intercambiable (IVI), virtualización de cliente rica, por nombrar algunos. Ciertas arquitecturas realizan una virtualización de GPU completa a través de captura y emulación para emular una GPU virtual con características completas mientras aún proporciona un rendimiento cercano a nativo pasando a través de recursos de memoria gráfica de rendimiento crítico.

Con la creciente importancia de las GPU en servidores para soportar 3D, medios y cargas de trabajo de GPGPU, la virtualización de GPU se está extendiendo más. Cómo virtualizar un acceso de memoria de GPU desde una máquina virtual (VM) es uno de los factores de diseño claves. La GPU tiene su propia memoria gráfica: o bien memoria de vídeo especializada o bien memoria de sistema compartida. Cuando la memoria de sistema se usa para gráficos, las direcciones físicas de invitado (GPA) necesitan traducirse a direcciones físicas de anfitrión (HPA) antes de accederse por el hardware.

Existen diversos enfoques para realizar la traducción para las GPU. Algunas implementaciones realizan traducción con soporte de hardware, pero la GPU puede pasarse a través a únicamente una VM. Otra solución es un enfoque de software que construye estructuras de sombra para la traducción. Por ejemplo, se implementan tablas de paginación de sombra en algunas arquitecturas tales como la solución de virtualización de GPU completa mencionada anteriormente, que puede soportar múltiples VM para compartir una GPU física.

En algunas implementaciones, las páginas de memoria de invitado/VM se respaldan por páginas de memoria de anfitrión. Un monitor de máquina virtual (VMM) (en ocasiones denominado un "Hipervisor") usa tablas de paginación extendidas (EPT), por ejemplo, para correlacionar desde una dirección física (PA) de invitado a una PA de anfitrión. Pueden usarse muchas tecnologías de uso compartido de memoria, tales como Fusión de Misma Página de Núcleo (KSM).

KSM combina páginas de múltiples VM con el mismo contenido, a una única página con protección de escritura. Es decir, si una página de memoria en VM1 (que correlaciona desde PA1 de invitado a PA1 de anfitrión), tiene los mismos contenidos que otra página de memoria en VM2 (que correlaciona desde PA2 de invitado a PA2 de anfitrión), puede usar únicamente una página de anfitrión (digamos HPA_SH) para respaldar la memoria de invitado. Es decir, tanto la PA1 de invitado de VM1 como PA2 de VM2 se correlacionan con HPA_SH con protección de escritura. Esto ahorra la memoria usada para el sistema, y es particularmente útil para páginas de memoria de solo lectura del invitado tal como páginas de código, y páginas de cero. Como KSM, se usa tecnología de copia en escritura (COW) para eliminar el uso compartido una vez que una VM modifica el contenido de página.

Se usa paso a través mediado en los sistemas de virtualización para rendimiento y uso compartido de dispositivo, en donde se presenta una única GPU física como múltiple GPU virtual a múltiples invitados con DMA directo, mientras los accesos de recursos de privilegios desde los invitados aún se capturan y emulan. En algunas implementaciones, cada invitado puede ejecutar el controlador de GPU nativo, y el DMA de dispositivo pasa directamente a la memoria sin intervención de hipervisor.

El documento "Engineering a fully GPU-accelerated H.264 encoder" de Bowei Li *et al.*, ACTAS DE SPIE, vol. 8878, 19 de julio de 2013 (19-07-2013), página 1-6, XP055503019, 1000 20th St. Bellingham WA 98225-6705 Estados Unidos; ISSN: 0277-786X, DOI: 10.1117/12.2030570, ISBN: 978-1-5106-1533-5, se refiere a un codificador de vídeo acelerado por GPU que usa la norma de codificación H.264 que incluye un algoritmo de estimación de movimiento con orientación de GPU.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Puede obtenerse un mejor entendimiento de la presente invención a partir de la siguiente descripción detallada en conjunto con los siguientes dibujos, en los que:

La Figura 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 gráficos;
La Figura 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 gráfico integrado;
La Figura 3 es un diagrama de bloques de una realización de un procesador gráfico que puede ser una unidad de procesamiento gráfico discreta, o puede ser un procesador gráfico integrado con una pluralidad de núcleos de procesamiento;
La Figura 4 es un diagrama de bloques de una realización de un motor de procesamiento gráfico para un procesador gráfico;
La Figura 5 es un diagrama de bloques de otra realización de un procesador gráfico;
La Figura 6 es un diagrama de bloques de lógica de ejecución de hilos que incluyen una matriz de elementos de procesamiento;
La Figura 7 ilustra un formato de instrucción de unidad de ejecución de procesador gráfico de acuerdo con una realización;
La Figura 8 es un diagrama de bloques de otra realización de un procesador gráfico que incluye una canalización de gráficos, una canalización de medios, un motor de visualización, lógica de ejecución de hilos y una canalización de salida de representación;
La Figura 9A es un diagrama de bloques que ilustra un formato de comando de procesador gráfico de acuerdo con una realización;
La Figura 9B es un diagrama de bloques que ilustra una secuencia de comandos de procesador gráfico de acuerdo con una realización;
La Figura 10 ilustra una arquitectura de software gráfica ilustrativa para un sistema de procesamiento de datos de acuerdo con una realización;
La Figura 11 ilustra un sistema de desarrollo de núcleo de IP ilustrativo que puede usarse para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización;
La Figura 12 ilustra un sistema ilustrativo en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización;
La Figura 13 ilustra un procesador gráfico ilustrativo de un sistema en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP;
La Figura 14 ilustra un procesador gráfico ilustrativo adicional de un sistema en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP
La Figura 15 ilustra un sistema de procesamiento gráfico ilustrativo;
La Figura 16 ilustra una arquitectura ilustrativa para virtualización gráfica completa;
La Figura 17 ilustra una arquitectura de procesamiento gráfico virtualizado ilustrativa que incluye unidades de procesamiento gráfico virtuales (vGPU);
La Figura 18 ilustra una realización de una arquitectura de virtualización con una IOMMU;
L Figura 19 ilustra una realización en la que se realiza procesamiento gráfico en un servidor;
Figura 20 ilustra una realización de una arquitectura gráfica virtualizada para una implementación automovilística;
La Figura 21 ilustra una asignación ilustrativa de direcciones físicas gráficas y apertura para cada máquina virtual;
La Figura 22 ilustra un sistema de visualización remoto de acuerdo con una realización de la invención;
La Figura 23 ilustra un método de acuerdo con una realización de la invención;
La Figura 24 es un diagrama de bloques que ilustra un sistema informático configurado para implementar uno o más aspectos de las realizaciones descritas en este documento;
Las Figuras 25A-25D ilustran componentes de procesador paralelo, de acuerdo con una realización;
Las Figuras 26A-26B son diagramas de bloques de multiprocesadores gráficos, de acuerdo con realizaciones;
Las Figuras 27A-27F ilustran una arquitectura ilustrativa en la que una pluralidad de GPU se acoplan comunicativamente a una pluralidad de procesadores de múltiples núcleos; y
La Figura 28 ilustra una canalización de procesamiento gráfico, de acuerdo con una realización.

DESCRIPCIÓN DETALLADA

En la siguiente descripción, para los propósitos de explicación, se exponen numerosos detalles específicos para proporcionar un entendimiento completo de las realizaciones de la invención descrita a continuación. Será evidente, sin embargo, para un experto en la materia que las realizaciones de la invención pueden ponerse en práctica sin algunos de estos detalles específicos. En otros casos, se muestran estructuras y dispositivos bien conocidos en forma de diagrama de bloques para evitar obstaculizar los principios subyacentes de las realizaciones de la invención.

ARQUITECTURAS ILUSTRATIVAS DE PROCESADOR GRÁFICO Y TIPOS DE DATOS

Visión 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 gráficos 108, y

puede ser un sistema de escritorio de un único procesador, un sistema de estación de trabajo de múltiples procesadores, o un sistema de servidor que tiene un gran número 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, de mano o embebidos.

Una realización del sistema 100 puede incluir, o incorporarse dentro de una plataforma de juego basada en servidor, una consola de juegos, incluyendo una consola de juegos y medios, una consola de juegos móvil, una consola de juegos de mano o una consola de juegos en línea. En algunas realizaciones, el sistema 100 es un teléfono móvil, teléfono inteligente, dispositivo informático de tableta o dispositivo de internet móvil. El sistema de procesamiento de datos 100 también puede incluir, acoplarse con, o integrarse dentro de un dispositivo ponible, tal como un dispositivo ponible de reloj inteligente, dispositivo de gafas inteligentes, dispositivo de realidad aumentada o dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 100 es una televisión o dispositivo de decodificador de salón que tiene uno o más procesadores 102 y una interfaz gráfica generada por uno o más procesadores gráficos 108.

En algunas realizaciones, cada uno del uno o más procesadores 102 incluye uno o más núcleos de procesador 107 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para sistema y software 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. En algunas realizaciones, el conjunto de instrucciones 109 puede facilitar Computación con Conjunto de Instrucciones Complejo (CISC), Computación con Conjunto de Instrucciones Reducido (RISC), o computación a través de una Palabra de Instrucción Muy Larga (VLIW). Cada uno de los múltiples núcleos de procesador 107 puede procesar un conjunto de instrucciones 109 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 107 también puede incluir otros dispositivos de procesamiento, tales como un Procesador de Señales Digitales (DSP).

En algunas realizaciones, el procesador 102 incluye una memoria caché 104. Dependiendo de la arquitectura, el procesador 102 puede tener una única memoria caché interna o múltiples niveles de memoria caché interna. En algunas realizaciones, la memoria caché se comparte entre diversos componentes del procesador 102. En algunas realizaciones, el procesador 102 también usa una memoria caché externa (por ejemplo, una memoria caché de Nivel 3 (L3) o Memoria Caché de Último Nivel (LLC)) (no mostrada), que puede compartirse entre núcleos de procesador 107 usando técnicas de coherencia de memoria caché conocidas. En el procesador 102 se incluye adicionalmente un archivo de registro 106 que puede incluir diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de enteros, registros de punto flotante, registros de estados y un registrador de puntero de instrucciones). Algunos registros pueden ser registros de fin general, mientras otros registros pueden ser específicos al diseño del procesador 102.

En algunas realizaciones, el procesador 102 se acopla con un bus de procesador 110 para transmitir señales de comunicación tal como dirección, datos o señales de control entre el procesador 102 y otros componentes en el sistema 100. En una realización, el sistema 100 usa una arquitectura de sistema de 'concentrador' ilustrativo, incluyendo un concentrador de controlador de memoria 116 y un concentrador de controlador de Entrada/Salida (I/O) 130. Un concentrador de controlador de memoria 116 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 100, mientras un concentrador de controlador de E/S (ICH) 130 proporciona conexiones a dispositivos de E/S a través de un bus de I/O local. En una realización, la lógica del concentrador de controlador de memoria 116 se integra dentro del procesador.

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), dispositivo de memoria flash, dispositivo de memoria de cambio de fase, o algún otro dispositivo de memoria que tiene un funcionamiento adecuado para servir como una memoria de proceso. En una realización, el dispositivo de memoria 120 puede operar como memoria de sistema para el sistema 100, para almacenar los datos 122 y las instrucciones 121 para su uso cuando el uno o más procesadores 102 ejecutan una aplicación o proceso. El concentrador de controlador de memoria 116 también se acopla con un procesador gráfico externo 112 opcional, que puede comunicarse con el uno o más procesadores gráficos 108 en los procesadores 102 para realizar operaciones gráficas y de medios.

En algunas realizaciones, el ICH 130 habilita que los periféricos se conecten al dispositivo de memoria 120 y al procesador 102 a través de un bus de I/O de alta velocidad. Los periféricos de I/O incluyen, pero sin limitación, un controlador de audio 146, una interfaz de firmware 128, un transceptor inalámbrico 126 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 124 (por ejemplo, unidad de disco duro, memoria flash, etc.) y un controlador de E/S heredado 140 para acoplar dispositivos (por ejemplo, Sistema Personal 2 (PS/2)) heredados al sistema. Uno o más controladores de Bus Serial Universal (USB) 142 conectan dispositivos de entrada, tales como combinaciones de teclado y ratón 144. Un controlador de red 134 también puede acoplarse con el ICH 130. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de procesador 110. Se apreciará que el sistema 100 mostrado es ilustrativo y no limitante, ya que también pueden usarse otros tipos de sistemas de procesamiento de datos que están configurados de forma diferente. Por ejemplo, el concentrador de controlador de E/S 130 puede integrarse dentro del uno o más procesadores 102, o el concentrador de controlador de

memoria 116 y el concentrador de controlador de E/S 130 pueden integrarse en un procesador gráfico externo discreto, tal como el procesador gráfico externo 112.

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 gráfico integrado 208. Esos elementos de la Figura 2 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en este documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otro sitio en este documento, pero sin limitación a tales. El procesador 200 puede incluir núcleos adicionales hasta e incluir núcleo adicional 202N representado por las cajas de líneas discontinuas. Cada uno de los núcleos de procesador 202A-202N incluye una o más unidades de memoria caché internas 204A-204N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades de memoria caché compartidas 206.

Las unidades de memoria caché internas 204A-204N y las unidades de memoria caché compartidas 206 representan una jerarquía de memorias caché dentro del procesador 200. La jerarquía de memorias caché puede incluir al menos un nivel de instrucción y memoria caché de 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, en donde el mayor nivel de memoria caché antes de la memoria externa se clasifica como la LLC. En algunas realizaciones, la lógica de coherencia de memoria caché mantiene la coherencia entre las diversas unidades de memoria caché 206 y 204A-204N.

En algunas realizaciones, 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. La una o más unidades de controlador de bus 216 gestionan un conjunto de buses periféricos, tales como uno o más buses de Interconexión de Componentes Periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 210 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunas realizaciones, el núcleo de agente de sistema 210 incluye uno o más controladores de memoria integrados 214 para gestionar el acceso a diversos dispositivos de memoria externa (no mostrados).

En algunas realizaciones, uno o más de los núcleos de procesador 202A-202N incluyen soporte para hilos múltiples simultáneos. En tal realización, el núcleo de agente de sistema 210 incluye componentes para coordinar y operar núcleos 202A-202N durante procesamiento de múltiples hilos. El núcleo de agente de sistema 210 puede incluir adicionalmente una unidad de control de potencia (PCU), que incluye lógica y componentes para regular el estado de potencia de los núcleos de procesador 202A-202N y el procesador gráfico 208.

En algunas realizaciones, el procesador 200 incluye adicionalmente el procesador gráfico 208 para ejecutar operaciones de procesamiento gráfico. En algunas realizaciones, el procesador gráfico 208 se acopla con el conjunto de unidades de memoria caché compartidas 206, y el núcleo de agente de sistema 210, incluyendo el uno o más controladores de memoria integrados 214. En algunas realizaciones, un controlador de visualización 211 se acopla con el procesador gráfico 208 para accionar el procesador gráfico emitido a uno o más visualizadores acoplados. En algunas realizaciones, el controlador de visualización 211 puede ser un módulo separado acoplado con el procesador gráfico a través de al menos una interconexión, o puede integrarse dentro del procesador gráfico 208 o del núcleo de agente de sistema 210.

En algunas realizaciones, se usa una unidad de interconexión basada en anillo 212 para acoplar los componentes internos del procesador 200. Sin embargo, puede usarse una unidad de interconexión alternativa, tal como una interconexión de punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador gráfico 208 se acopla con la interconexión de anillo 212 a través de un enlace de I/O 213.

El enlace de I/O 213 ilustrativo representa al menos una de múltiples variedades de interconexiones de I/O, incluyendo una interconexión de I/O en paquete que facilita comunicación entre diversos componentes de procesador y un módulo de memoria embebida de alto rendimiento 218, tal como un módulo de eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 202A-202N y el procesador gráfico 208 usan módulos de memoria embebida 218 como una Memoria Caché de Último Nivel compartida.

En algunas realizaciones, los núcleos de procesador 202A-202N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 202A-202N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), en donde uno o más de los núcleos de procesador 202A-202N ejecutan un primer conjunto de instrucciones, mientras al menos uno de los otros núcleos ejecuta un subconjunto del primer conjunto de instrucciones o un conjunto de instrucciones diferente. En una realización, los núcleos de procesador 202A-202N son heterogéneos en términos de microarquitectura, en donde uno o más núcleos que tienen un consumo de potencia relativamente mayor se acoplan con uno o más núcleos de potencia que tienen un menor consumo de potencia. Adicionalmente, el procesador 200 puede implementarse en uno o más chips o como un circuito integrado de SoC que tiene los componentes ilustrados, además de otros componentes.

La Figura 3 es un diagrama de bloques de un procesador gráfico 300, que puede ser una unidad de procesamiento gráfico discreta, o puede ser un procesador gráfico integrado con una pluralidad de núcleos de procesamiento. En algunas realizaciones, el procesador gráfico se comunica a través de una interfaz de E/S con correlación de memoria con registros en el procesador gráfico y con comandos situados en la memoria de procesador. En algunas realizaciones, el procesador gráfico 300 incluye una interfaz de memoria 314 para acceder a una memoria. La interfaz de memoria 314 puede ser una interfaz a memoria local, una o más memorias caché internas, una o más memorias caché externas compartidas y/o una memoria de sistema.

En algunas realizaciones, el procesador gráfico 300 también incluye un controlador de visualización 302 para accionar 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 el visualizador y composición de múltiples capas de vídeo o elementos de interfaz de usuario. En algunas realizaciones, el procesador gráfico 300 incluye un motor de códec de vídeo 306 para codificar, decodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluyendo, pero sin limitación a formatos de Grupo de Expertos en Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como la Sociedad de Ingenieros Cinematográficos y de Televisión (SMPTE) 421M/VC-1, y formatos de Grupo Mixto de Expertos en Fotografía (JPEG) tales como JPEG, y formatos de JPEG en movimiento (MJPEG).

En algunas realizaciones, el procesador gráfico 300 incluye un motor de transferencia de imagen de bloque (BLIT) 304 para realizar operaciones de rasterizador de dos dimensiones (2D) que incluyen, por ejemplo, transferencias de bloque de límite de bit. Sin embargo, en una realización, operaciones gráficas 2D se realizan usando uno o más componentes de motor de procesamiento gráfico (GPE) 310. En algunas realizaciones, el GPE 310 es un motor de cálculo para realizar operaciones gráficas, incluyendo operaciones gráficas de tres dimensiones (3D) y operaciones de medios.

En algunas realizaciones, el GPE 310 incluye una canalización 3D 312 para realizar operaciones 3D, tales como representar imágenes de tres dimensiones y escenas que usan funciones de procesamiento que actúan sobre formas de primitiva 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización 3D 312 incluye elementos de funciones programables y fijas que realizan diversas tareas dentro del elemento y/o hilos de ejecución de generación a un subsistema de medios/3D 315. Mientras la canalización 3D 312 puede usarse para realizar operaciones de medios, una realización de GPE 310 también incluye una canalización de medios 316 que se usa específicamente para realizar operaciones de medios, tal como posprocesamiento de vídeo y mejora de imagen.

En algunas realizaciones, la canalización de medios 316 incluye unidades de función fija o de lógica programable para realizar una o más operaciones de medios especializadas, tales como aceleración de decodificación de vídeo, desintercalado de vídeo y aceleración de codificación de vídeo en lugar de, o en nombre del motor de códec de vídeo 306. En algunas realizaciones, la canalización de medios 316 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema de medios/3D 315. Los hilos generados realizan cálculos para las operaciones de medios en una o más unidades de ejecución de gráficos incluidas en el subsistema de medios/3D 315.

En algunas realizaciones, el subsistema 3D/de medios 315 incluye lógica para ejecutar hilos generados por la canalización 3D 312 y la canalización de medios 316. En una realización, las canalizaciones enviaron peticiones de ejecución de hilos al subsistema 3D/de medios 315, que incluyen lógica de distribución de hilos para arbitrar y distribuir las diversas peticiones a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos 3D y de medios. En algunas realizaciones, el subsistema 3D/de medios 315 incluye una o más memorias caché internas para instrucciones y datos de hilos. En algunas realizaciones, el subsistema también incluye memoria compartida, incluyendo registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Motor de procesamiento gráfico

La Figura 4 es un diagrama de bloques de un motor de procesamiento gráfico 410 de un procesador gráfico de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento gráfico (GPE) 410 es una versión del GPE 310 mostrado 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 este documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otro sitio en este documento, pero sin limitación a tales. Por ejemplo, se ilustran la canalización 3D 312 y la canalización de medios 316 de la Figura 3. La canalización de medios 316 es opcional en algunas realizaciones del GPE 410 y puede no incluirse explícitamente dentro del GPE 410. Por ejemplo, y en al menos una realización, un procesador de medios y/o imágenes separado se acopla al GPE 410.

En algunas realizaciones, el GPE 410 se acopla con o incluye un transmisor de comandos 403, que proporciona un flujo de comandos a la canalización 3D 312 y/o canalizaciones de medios 316. En algunas realizaciones, el transmisor de comandos 403 se acopla con memoria, que puede ser memoria de sistema, o una o más de memoria caché interna y memoria caché compartida. En algunas realizaciones, el transmisor de comandos 403 recibe comandos desde la memoria y envía los comandos a la canalización 3D 312 y/o la canalización de medios 316. Los comandos son

directivas buscadas de una memoria intermedia de anillo, que almacena comandos para la canalización 3D 312 y la canalización de medios 316. En una realización, la memoria intermedia de anillo puede incluir adicionalmente memorias intermedias de comandos en lote que almacenan lotes de múltiples comandos. Los comandos para la canalización 3D 312 también pueden incluir referencias a datos almacenados en memoria, tales como pero sin limitación a datos de vértice y geometría para la canalización 3D 312 y/o datos de imagen y objetos de memoria para la canalización de medios 316. La canalización 3D 312 y la canalización de medios 316 procesan los comandos y datos realizando operaciones a través de lógica dentro de las respectivas canalizaciones o distribuyendo uno o más hilos de ejecución a una matriz de núcleo gráfico 414.

En diversas realizaciones, la canalización 3D 312 puede ejecutar uno o más programas de sombreador, tal como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculo u otros programas de sombreador, procesando las instrucciones y distribuyendo hilos de ejecución a la matriz de núcleo gráfico 414. La matriz de núcleo gráfico 414 proporciona un bloque unificado de recursos de ejecución. Lógica de ejecución multipropósito (por ejemplo, unidades de ejecución) dentro de la matriz de núcleo gráfico 414 incluye soporte para diversos lenguajes de sombreador de API 3D y puede ejecutar múltiples hilos de ejecución simultánea asociados con múltiples sombreadores.

En algunas realizaciones, la matriz de núcleo gráfico 414 también incluye lógica de ejecución para realizar funciones de medios, tales como procesamiento de vídeo y/o imagen. En una realización, las unidades de ejecución incluyen adicionalmente lógica de fin general que es programable para realizar operaciones de cálculo de fin general paralelas, además de operaciones de procesamiento gráfico. La lógica de fin general puede realizar operaciones de procesamiento en paralelo o en conjunto con lógica de fin general dentro del núcleo o núcleos de procesador 107 de la Figura 1 o el núcleo 202A-202N como en la Figura 2.

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleo gráfico 414 pueden emitir datos a memoria en una memoria intermedia de retorno unificada (URB) 418. La URB 418 puede almacenar datos para múltiples hilos. En algunas realizaciones, la URB 418 puede usarse para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleo gráfico 414. En algunas realizaciones, la URB 418 puede usarse adicionalmente para sincronización entre hilos en la matriz de núcleo gráfico y lógica de función fija dentro de la lógica de función compartida 420.

En algunas realizaciones, la matriz de núcleo gráfico 414 es escalable, de tal forma que la matriz incluye un número variable de núcleos gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y nivel de rendimiento del GPE 410. En una realización, los recursos de ejecución son escalables dinámicamente, de tal forma que los recursos de ejecución pueden habilitarse o deshabilitarse según se necesite.

La matriz de núcleo gráfico 414 se acopla con lógica de función compartida 420 que incluye múltiples recursos que se comparten entre los núcleos gráficos en la matriz de núcleo gráfico. Las funciones compartidas dentro de la lógica de función compartida 420 son unidades de lógica de hardware que proporcionan una funcionalidad suplementaria especializada a la matriz de núcleo gráfico 414. En diversas realizaciones, la lógica de función compartida 420 incluye, pero sin limitación a, muestreador 421, matemáticas 422 y lógica de comunicación entre hilos (ITC) 423. Adicionalmente, algunas realizaciones implementan una o más memoria o memorias caché 425 dentro de la lógica de función compartida 420. Una función compartida se implementa en donde la demanda para una función especializada dada es insuficiente para su inclusión dentro de la matriz de núcleo gráfico 414. En su lugar se implementa una única instanciación de esa función especializada como una entidad independiente en la lógica de función compartida 420 y compartida entre los recursos de ejecución dentro de la matriz de núcleo gráfico 414. El conjunto preciso de funciones que se comparten entre la matriz de núcleo gráfico 414 y se incluyen dentro de la matriz de núcleo gráfico 414 varía entre realizaciones.

La Figura 5 es un diagrama de bloques de otra realización de un procesador gráfico 500. Los elementos de la Figura 5 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en este documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otro sitio en este documento, pero sin limitación a tales.

En algunas realizaciones, el procesador gráfico 500 incluye una interconexión de anillo 502, un extremo frontal de canalización 504, un motor de medios 537 y núcleos gráficos 580A-580N. En algunas realizaciones, la interconexión de anillo 502 acopla el procesador gráfico a otras unidades de procesamiento, incluyendo otros procesadores gráficos o uno o más núcleos de procesador de fin general. En algunas realizaciones, el procesador gráfico es uno de muchos procesadores integrados dentro de un sistema de procesamiento de múltiples núcleos.

En algunas realizaciones, el procesador gráfico 500 recibe lotes de comandos a través de la interconexión de anillo 502. Los comandos entrantes se interpretan por un transmisor de comandos 503 en el extremo frontal de canalización 504. En algunas realizaciones, el procesador gráfico 500 incluye lógica de ejecución escalable para realizar procesamiento de geometría 3D y procesamiento de medios a través del núcleo o núcleos gráficos 580A-580N. Para comandos de procesamiento de geometría 3D, el transmisor de comandos 503 suministra comandos a la canalización de geometría 536. Para al menos algunos comandos de procesamiento de medios, el transmisor de comandos 503

suministra los comandos a un extremo frontal de vídeo 534, que se acopla con un motor de medios 537. En algunas realizaciones, el motor de medios 537 incluye un Motor de Calidad de Vídeo (VQE) 530 para vídeo y posprocesamiento de imagen y un motor de codificación/decodificación multiformato (MFX) 533 para proporcionar codificación y decodificación de datos de medios acelerado por hardware. En algunas realizaciones, cada uno de la canalización de geometría 536 y el motor de medios 537 genera hilos de ejecución para los recursos de ejecución de hilos proporcionados por al menos un núcleo gráfico 580A.

En algunas realizaciones, el procesador gráfico 500 incluye recursos de ejecución de hilos escalables que presentan núcleos modulares 580A-580N (en ocasiones denominados sectores de núcleo), teniendo cada uno múltiples subnúcleos 550A-550N, 560A-560N (en ocasiones denominados subsectores de núcleo). En algunas realizaciones, el procesador gráfico 500 puede tener cualquier número de núcleos gráficos 580A a 580N. En algunas realizaciones, el procesador gráfico 500 incluye un núcleo gráfico 580A que tiene al menos un primer subnúcleo 550A y un segundo subnúcleo 560A. En otras realizaciones, el procesador gráfico es un procesador de baja potencia con un único subnúcleo (por ejemplo, 550A). En algunas realizaciones, el procesador gráfico 500 incluye múltiples núcleos gráficos 580A-580N, incluyendo cada uno un conjunto de primeros subnúcleos 550A-550N y un conjunto de segundos subnúcleos 560A-560N. Cada subnúcleo en el conjunto de primeros subnúcleos 550A-550N incluye al menos un primer conjunto de unidades de ejecución 552A-552N y muestreadores de medios/textura 554A-554N. Cada subnúcleo en el conjunto de segundos subnúcleos 560A-560N incluye al menos un segundo conjunto de unidades de ejecución 562A-562N y muestreadores 564A-564N. En algunas realizaciones, cada subnúcleo 550A-550N, 560A-560N comparte un conjunto de recursos compartidos 570A-570N. En algunas realizaciones, los recursos compartidos incluyen memoria caché compartida y lógica de operación de píxel. En las diversas realizaciones del procesador gráfico también pueden incluirse otros recursos compartidos.

Unidades de ejecución

La Figura 6 ilustra lógica de ejecución de hilos 600 que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones de un GPE. Los elementos de la Figura 6 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en este documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otro sitio en este documento, pero sin limitación a tales.

En algunas realizaciones, la lógica de ejecución de hilos 600 incluye un procesador de sombreador 602, un distribuidor de hilos 604, una memoria caché de instrucciones 606, una matriz de unidad de ejecución escalable que incluye una pluralidad de unidades de ejecución 608A-608N, un muestreador 610, una memoria caché de datos 612 y un puerto de datos 614. En una realización, la matriz de unidad de ejecución escalable puede escalar dinámicamente habilitando o deshabilitando una o más unidades de ejecución (por ejemplo, cualquiera de la unidad de ejecución 608A, 608B, 608C, 608D a 608N-1 y 608N) basándose en los requisitos de cálculo de una carga de trabajo. En una realización, los componentes incluidos se interconectan a través de un tejido de interconexión que enlaza a cada uno de los componentes. En algunas realizaciones, la lógica de ejecución de hilos 600 incluye una o más conexiones a memoria, tal como memoria de sistema o memoria caché, a través de una o más de memoria caché de instrucciones 606, puerto de datos 614, muestreador 610 y unidades de ejecución 608A-608N. En algunas realizaciones, cada unidad de ejecución (por ejemplo, 608A) es una unidad de cálculo de fin general programable independiente que es capaz de ejecutar múltiples hilos de hardware simultáneos mientras procesa múltiples elementos de datos en paralelo para cada hilo. En diversas realizaciones, la matriz de unidades de ejecución 608A-608N es escalable para incluir cualquier número unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 608A-608N se usan esencialmente para ejecutar programas de sombreador. Un procesador de sombreador 602 puede procesar los diversos programas de sombreador y distribuir hilos de ejecución asociados con los programas de sombreador a través de un distribuidor de hilos 604. En una realización, el distribuidor de hilos incluye lógica para arbitrar peticiones de iniciación de hilos de las canalizaciones de gráficos y medios e instanciar los hilos solicitados en una o más unidad de ejecución en las unidades de ejecución 608A-608N. Por ejemplo, la canalización de geometría (por ejemplo, 536 de la Figura 5) puede distribuir vértices, teselación o sombreadores de geometría a la lógica de ejecución de hilos 600 (Figura 6) para su procesamiento. En algunas realizaciones, el distribuidor de hilos 604 también puede procesar peticiones de generación de hilos de tiempo de ejecución a partir de la ejecución de programas de sombreador.

En algunas realizaciones, las unidades de ejecución 608A-608N soportan un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreador de gráficos 3D estándar, de tal forma que programas de sombreador de bibliotecas gráficas (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución soportan procesamiento de vértices y geometría (por ejemplo, programas de vértice, programas de geometría, sombreadores de vértices), procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y procesamiento de fin general (por ejemplo, sombreadores de cálculo y de medios). Cada una de las unidades de ejecución 608A-608N es capaz de ejecución de instrucción única, múltiples datos (SIMD) multiemisión y operación multihilos habilita un entorno de ejecución eficiente frente a accesos de memoria de latencia mayor. Cada hilo de hardware dentro de cada unidad de ejecución tiene un archivo de registro de banda ancha alta especializado y estado de hilo independiente asociado. La ejecución es multiemisión por reloj a canalizaciones con capacidad de operaciones de punto flotante de precisión única o doble y de enteros, capacidad de rama de SIMD,

operaciones lógicas, operaciones transcendentales y otras operaciones misceláneas. Mientras espera datos de memoria o una de las funciones compartidas, la lógica de dependencia dentro de las unidades de ejecución 608A-608N provoca que un hilo de espera esté en suspensión hasta que se hayan devuelto los datos solicitados. Mientras el hilo de espera está en suspensión, recursos de hardware pueden dedicarse a procesar otros hilos. Por ejemplo, durante un retardo asociado con una operación de sombreador de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, sombreador de fragmentos u otro tipo de programa de sombreador, incluyendo un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 608A-608N opera en 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 acceso a elemento de datos, enmascaramiento y control de flujo dentro de instrucciones. El número de canales puede ser independiente del número de Unidades Aritméticas Lógica (ALU) físicas o Unidades de Punto Flotante (FPU) para un procesador gráfico particular. En algunas realizaciones, las unidades de ejecución 608A-608N soportan tipos de datos de enteros y de punto flotante.

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

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

Durante la ejecución, las canalizaciones de gráficos y medios enviaron peticiones de iniciación de hilos a lógica de ejecución de hilos 600 a través de lógica de generación y distribución de hilos. Una vez que un grupo de objetos geométricos se ha procesado y rasterizado en datos de píxel, lógica de procesador de píxel (por ejemplo, lógica de sombreador de píxel, sombreador de fragmentos lógica, etc.) dentro del procesador de sombreador 602 se invoca para calcular adicionalmente información emitida y provocar que los resultados se escriban en superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de patrones, etc.). En algunas realizaciones, un sombreador de píxeles o sombreador de fragmentos calcula los valores de los diversos atributos de vértice que tienen que interpolarse a través del objeto rasterizado. En algunas realizaciones, lógica de procesador de píxel dentro del procesador de sombreador 602 ejecuta, a continuación, un píxel suministrado por interfaz de programación de aplicación (API) o programa de sombreador de fragmentos. Para ejecutar el programa de sombreador, el procesador de sombreador 602 distribuye hilos a una unidad de ejecución (por ejemplo, 608A) a través de distribuidor de hilos 604. En algunas realizaciones, el sombreador de píxeles 602 usa lógica de muestreo de textura en el muestreador 610 para acceder a datos de textura en mapas de textura almacenados en memoria. Operaciones aritméticas en los datos de textura y los datos de geometría de entrada calculan datos de color de píxel para cada fragmento geométrico, o descarta uno o más píxeles de procesamiento adicional.

En algunas realizaciones, el puerto de datos 614 proporciona un mecanismo de acceso a memoria para que la lógica de ejecución de hilos 600 emita datos procesados a memoria para procesar en una canalización de salida de procesador gráfico. En algunas realizaciones, el puerto de datos 614 incluye o se acopla a una o más memorias caché (por ejemplo, memoria caché de datos 612) a datos de memoria caché para acceso de memoria a través del puerto de datos.

La Figura 7 es un diagrama de bloques que ilustra un formato de instrucción de procesador gráfico 700 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución de procesador gráfico soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Las cajas de línea continua ilustran los componentes que se incluyen generalmente en una instrucción de unidad de ejecución, mientras las líneas discontinuas incluyen componentes que son opcionales o que se incluyen únicamente en un subconjunto de las instrucciones. En algunas realizaciones, el formato de instrucción 700 descrito e ilustrado son macro instrucciones, en que son instrucciones suministradas a la unidad de ejecución, al contrario de micro operaciones que resultan de decodificación de instrucción una vez que se procesa la instrucción.

En algunas realizaciones, las unidades de ejecución de procesador gráfico soportan de forma nativa instrucciones en una instrucción de formato de 128 bits 710. Un formato de instrucción compactado de 64 bits 730 está disponible para algunas instrucciones basándose en la instrucción seleccionada, opciones de instrucción y número de operandos. El formato de instrucción de 128 bits nativo 710 proporciona acceso a todas las opciones de instrucción, mientras algunas opciones y operaciones se restringen en el formato de instrucción de 64 bits 730. Las instrucciones nativas disponibles en el formato de instrucción de 64 bits 730 varían por realización. En algunas realizaciones, 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 hace referencia a 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 opcode de instrucción 712 define la operación que la unidad de ejecución tiene que realizar. Las unidades de ejecución ejecutan cada instrucción en paralelo a través de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de suma, la unidad de ejecución realiza una operación de suma simultánea a través de cada canal de color que representa un elemento de textura o elemento de instantánea. Por defecto, la unidad de ejecución realiza cada instrucción a través de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 714 habilita control sobre ciertas opciones de ejecución, tales como selección de canales (por ejemplo, predicación) y orden de canal de datos (por ejemplo, mezcla). Para instrucciones en el formato de instrucción de 128 bits 710, un campo de tamaño de ejecución 716 limita el número de canales de datos que se ejecutarán en paralelo. En algunas realizaciones, el campo de tamaño de ejecución 716 no está disponible para su uso en el formato de instrucción compacto de 64 bits 730.

Algunas instrucciones de unidad de ejecución tienen hasta tres operandos que incluyen dos operandos de origen, src0 720, src1 722, y un destino 718. En algunas realizaciones, las unidades de ejecución soportan instrucciones de destino doble, en donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 724), en donde el opcode de instrucción 712 determina el número de operandos de origen. El último operando de origen de una instrucción puede ser un valor inmediato (por ejemplo, codificado de forma rígida) pasado con la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 710 incluye un campo de modo de acceso/dirección 726 que especifica, por ejemplo, si se usa modo de direccionamiento de registro directo o modo de direccionamiento de registro indirecto. Cuando se usa modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos se proporciona directamente mediante bits en la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 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. En una realización, el modo de acceso se usa para definir una alineación de acceso a datos para la instrucción. Algunas realizaciones soportan modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, en donde la alineación de byte 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 con alineación de byte para operandos de origen y destino y cuando está en un segundo modo, la instrucción puede usar direccionamiento con alineación de 16 bytes para todos los operandos de origen y destino.

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

En algunas realizaciones, las instrucciones se agrupan basándose en campos de bits de opcode 712 para simplificar la decodificación de opcode 740. Para un opcode de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de opcode. El agrupamiento de opcode preciso mostrado es solamente un ejemplo. En algunas realizaciones, un grupo de opcode de movimiento y lógica 742 incluye instrucciones de movimiento y lógica de datos (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y lógica 742 comparte los cinco bits más significativos (MSB), en donde las instrucciones de movimiento (mov) son en forma de 0000xxxxb y las instrucciones de lógica son en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 744 (por ejemplo, llamar, saltar (jmp)) incluye instrucciones en forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 746 incluye una mezcla de instrucciones, incluyendo instrucciones de sincronización (por ejemplo, esperar, enviar) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones matemáticas en paralelo 748 incluye instrucciones aritméticas de componente (por ejemplo, sumar, multiplicar (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de matemáticas en paralelo 748 realiza las operaciones aritméticas en paralelo a través de canales de datos. El grupo de matemáticas de vectores 750 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101 xxxxb (por ejemplo, 0x50). El grupo de matemáticas de vectores realiza aritmética tal como cálculos de producto escalar en operandos vectoriales.

Canalización de gráficos

La Figura 8 es un diagrama de bloques de otra realización de un procesador gráfico 800. Los elementos de la Figura 8 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en este documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otro sitio en este documento, pero sin limitación a tales.

En algunas realizaciones, el procesador gráfico 800 incluye una canalización de gráficos 820, una canalización de medios 830, un motor de visualización 840, una lógica de ejecución de hilos 850 y una canalización de salida de representación 870. En algunas realizaciones, el procesador gráfico 800 es un procesador gráfico dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de fin general. El procesador gráfico se controla mediante escrituras de registro escribe en uno o más registros de control (no mostrados) o a través de comandos emitidos al procesador gráfico 800 a través de una interconexión de anillo 802. En algunas realizaciones, la interconexión de anillo 802 acopla el procesador gráfico 800 a otros componentes de procesamiento, tales como otros procesadores gráficos o procesadores de fin general. Los comandos de la interconexión de anillo 802 se interpretan por un transmisor de comandos 803, que suministra instrucciones a componentes individuales de la canalización de gráficos 820 o la canalización de medios 830.

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

En algunas realizaciones, las unidades de ejecución 852A-852B están en una matriz de procesadores de vectores que tiene un conjunto de instrucciones para realizar operaciones gráficas y de medios. En algunas realizaciones, las unidades de ejecución 852A-852B tienen una memoria caché L1 851 conectada que es específica para cada matriz o se comparte entre las matrices. La memoria caché puede configurarse como una memoria caché de datos, una memoria caché de instrucciones, o una única memoria caché que se particiona para contener datos e instrucciones en diferentes particiones.

En algunas realizaciones, la canalización de gráficos 820 incluye componentes de teselación para realizar teselación acelerada por hardware de objetos 3D. En algunas realizaciones, un sombreador de casco programable 811 configura las operaciones de teselación. Un sombreador de dominio programable 817 proporciona evaluación de extremo final de salida de teselación. Un teselador 813 opera en la dirección de sombreador de casco 811 y contiene lógica de fin especial para generar un conjunto de objetos geométricos detallados basándose en un modelo geométrico grueso que se proporciona como una entrada a la canalización de gráficos 820. En algunas realizaciones, si no se usa teselación, los componentes de teselación (por ejemplo, sombreador de casco 811, teselador 813 y sombreador de dominio 817) pueden omitirse.

En algunas realizaciones, objetos geométricos completos pueden procesarse por un sombreador de geometría 819 a través de uno o más hilos distribuidos a las unidades de ejecución 852A-852B, o pueden proceder directamente al recortador 829. En algunas realizaciones, el sombreador de geometría opera en objetos geométricos enteros, en lugar de vértices o parches de vértices como en etapas anteriores de la canalización de gráficos. Si la teselación está deshabilitada, el sombreador de geometría 819 recibe una entrada desde el sombreador de vértices 807. En algunas realizaciones, el sombreador de geometría 819 es programable por un programa de sombreador de geometría para realizar teselación de geometría si las unidades de teselación están deshabilitadas.

Antes de rasterización, un recortador 829 procesa datos de vértice. El recortador 829 puede ser un recortador de función fija o un recortador programable que tiene funciones de recorte y sombreador de geometría. En algunas realizaciones, un componente de prueba de rasterizador y profundidad 873 en la canalización de salida de representación 870 distribuye sombreadores de píxeles para convertir los objetos geométricos a sus representaciones por píxel. En algunas realizaciones, se incluye una lógica de sombreador de píxel en la lógica de ejecución de hilos 850. En algunas realizaciones, una aplicación puede omitir el componente de prueba de rasterizador y profundidad 873 y acceder a datos de vértice no rasterizados a través de una unidad de flujo de salida 823.

El procesador gráfico 800 tiene un bus de interconexión, tejido de interconexión o algún otro mecanismo de interconexión que permite que datos y mensajes pasen entre los componentes principales del procesador. En algunas realizaciones, las unidades de ejecución 852A-852B y memoria o memorias caché 851 asociadas, muestreador de textura y medios 854, y memoria caché de textura/muestreador 858 se interconectan a través de un puerto de datos 856 para realizar acceso de memoria y comunicarse con componentes de canalización de salida de representación del procesador. En algunas realizaciones, cada uno del muestreador 854, memorias caché 851, 858 y unidades de ejecución 852A-852B tienen trayectorias de acceso a memoria separadas.

En algunas realizaciones, la canalización de salida de representación 870 contiene un componente de prueba de rasterizador y profundidad 873 que convierte objetos basados en vértices en una representación basada en píxel

asociada. En algunas realizaciones, la lógica de rasterizador incluye una unidad de ventana/enmascarador para realizar rasterización lineal y triangular de función fija. Una memoria caché de representación 878 asociada y memoria caché de profundidad 879 también están disponibles en algunas realizaciones. Un componente de operaciones de píxel 877 realiza operaciones basadas en píxel en los datos, aunque en algunos casos, operaciones de píxel asociadas con operaciones 2D (por ejemplo, transferencias de imagen de bloque de bit con combinación) se realizan por el motor 2D 841, o se sustituyen en el momento de visualización por el controlador de visualización 843 usando planos de visualización de superposición. En algunas realizaciones, una memoria caché compartida L3 875 está disponible a todos los componentes gráficos, permitiendo el uso compartido de datos sin el uso de memoria de sistema principal.

En algunas realizaciones, la canalización de medios de procesador gráfico 830 incluye un motor de medios 837 y un extremo frontal de vídeo 834. En algunas realizaciones, el extremo frontal de vídeo 834 recibe comandos de canalización desde el transmisor de comandos 803. En algunas realizaciones, la canalización de medios 830 incluye un transmisor de comandos separado. En algunas realizaciones, el extremo frontal de vídeo 834 procesa comandos de medios antes de enviar el comando al motor de medios 837. En algunas realizaciones, el motor de medios 837 incluye una funcionalidad de generación de hilos para generar hilos para distribuir a la lógica de ejecución de hilos 850 a través del distribuidor de hilos 831.

En algunas realizaciones, el procesador gráfico 800 incluye un motor de visualización 840. En algunas realizaciones, el motor de visualización 840 es externo al procesador 800 y se acopla con el procesador gráfico a través de la interconexión de anillo 802, o algún otro bus o tejido de interconexión. En algunas realizaciones, el motor de visualización 840 incluye un motor 2D 841 y un controlador de visualización 843. En algunas realizaciones, el motor de visualización 840 contiene lógica de fin especial con capacidad de operar independientemente de la canalización 3D. En algunas realizaciones, el controlador de visualización 843 se acopla con un dispositivo de visualización (no mostrado), que puede ser un dispositivo de visualización integrado en sistema, como en un ordenador portátil, o un dispositivo de visualización externo conectado a través de un conector de dispositivo de visualización.

En algunas realizaciones, la canalización de gráficos 820 y la canalización de medios 830 son configurables para realizar operaciones basándose en múltiples interfaces de programación de gráficos y medios y no son específicos a ninguna interfaz de programación de aplicación (API). En algunas realizaciones, software de controlador para el procesador gráfico traduce llamadas de API, son específicas para una biblioteca de gráficos o medios particular, a comandos que pueden procesarse por el procesador gráfico. En algunas realizaciones, se proporciona soporte para la Biblioteca de Gráficos Abierta (OpenGL), Lenguaje de Computación Abierto (OpenCL) y/o API de gráficos y cálculo, todos de Khronos Group. En algunas realizaciones, también puede proporcionarse soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, puede soportarse una combinación de estas bibliotecas. También puede proporcionarse soporte para la Biblioteca de Visión Artificial de Código Abierto (OpenCV). Una futura API con una canalización 3D compatible también se soportaría si puede hacerse una correlación desde la canalización de la futura API con la canalización del procesador gráfico.

Programación de canalización de gráficos

La Figura 9A es un diagrama de bloques que ilustra un formato de comando de procesador gráfico 900 de acuerdo con algunas realizaciones. **La Figura 9B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador gráfico 910 de acuerdo con una realización. Las cajas de línea continua en **la Figura 9A** ilustran los componentes que se incluyen generalmente en un comando de gráficos mientras las líneas discontinuas incluyen componentes que son opcionales o que se incluyen únicamente en un subconjunto de los comandos de gráficos. El formato de comando de procesador gráfico 900 ilustrativo de **la Figura 9A** incluye campos de datos para identificar un cliente objetivo 902 del comando, un código de operación (opcode) de comando 904 y los datos pertinentes 906 para el comando. En algunos comandos también se incluyen un subopcode 905 y un tamaño de comando 908.

En algunas realizaciones, el cliente 902 especifica la unidad de cliente del dispositivo gráfico que procesa los datos de comando. En algunas realizaciones, un analizador de comando de procesador gráfico examina el comando de cliente de cada comando para condicionar el procesamiento adicional del comando y encaminar los datos de comando a la unidad de cliente apropiada. En algunas realizaciones, las unidades de cliente de procesador gráfico incluyen una unidad de interfaz de memoria, una unidad de representación, una unidad 2D, una unidad 3D y una unidad de medios. Cada unidad de cliente tiene una correspondiente canalización de procesamiento que procesa los comandos. Una vez que la unidad de cliente recibe el comando, la unidad de cliente lee el opcode 904 y, si está presente, el subopcode 905 para determinar la operación a realizar. La unidad de cliente realiza el comando usando información en el campo de datos 906. Para algunos comandos se espera que un tamaño de comando explícito 908 especifique el tamaño del comando. En algunas realizaciones, el analizador de comandos determina automáticamente el tamaño de al menos algunos de los comandos basándose en el opcode de comando. En algunas realizaciones, los comandos se alinean a través de múltiples de una palabra doble.

El diagrama de flujo en **la Figura 9B** muestra una secuencia de comandos de procesador gráfico 910 ilustrativa. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que presenta una realización de un procesador gráfico usa una versión de la secuencia de comandos mostrada para configurar, ejecutar y terminar un conjunto de operaciones gráficas. Una secuencia de comandos de muestra se muestra y describe para propósitos

de ejemplo únicamente ya que las realizaciones no se limitan a estos comandos específicos o a esta secuencia de comandos. Además, los comandos pueden emitirse como un lote de comandos en una secuencia de comandos, de tal forma que el procesador gráfico procesará la secuencia de comandos en ocurrencia al menos parcialmente.

En algunas realizaciones, la secuencia de comandos de procesador gráfico 910 puede comenzar con un comando de vaciado de canalización 912 para provocar que cualquier canalización de gráficos activa complete los comandos pendientes en la actualidad para la canalización. En algunas realizaciones, la canalización 3D 922 y la canalización de medios 924 no operan simultáneamente. El vaciado de canalización se realiza para provocar que la canalización de gráficos activa complete cualquier comando pendiente. En respuesta a un vaciado de canalización, el analizador de comandos para el procesador gráfico pausará el procesamiento de comando hasta que los motores de extracción activos completen las operaciones pendientes y se invaliden las memorias caché de lectura pertinentes. Opcionalmente, cualquier dato en la memoria caché de representación que se marca como 'sucio' puede vaciarse a la memoria. En algunas realizaciones, el comando de vaciado de canalización 912 puede usarse para sincronización de canalización o antes de situar el procesador gráfico en un estado de baja potencia.

En algunas realizaciones, se usa un comando de selección de canalización 913 cuando una secuencia de comandos requiere que el procesador gráfico conmute explícitamente entre canalizaciones. En algunas realizaciones, se requiere un comando de selección de canalización 913 únicamente una vez está dentro de un contexto de ejecución antes de emitir comandos de canalización a no ser que el contexto es emitir comandos para ambas canalizaciones. En algunas realizaciones, se requiere un comando de vaciado de canalización 912 inmediatamente antes de una conmutación de canalización a través del comando de selección de canalización 913.

En algunas realizaciones, un comando de control de canalización 914 configura una canalización de gráficos para operación y se usa para programar la canalización 3D 922 y la canalización de medios 924. En algunas realizaciones, el comando de control de canalización 914 configura el estado de canalización para la canalización activa. En una realización, el comando de control de canalización 914 se usa para sincronización de canalización y para borrar datos de una o más memorias caché dentro de la canalización activa antes de procesar un lote de comandos.

En algunas realizaciones, se usan comandos para el estado de memoria intermedia de retorno 916 para configurar un conjunto de memorias intermedias de retorno para las respectivas canalizaciones para escribir datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas realizaciones, el procesador gráfico también usa una o más memorias intermedias de retorno para almacenar datos de salida y para realizar comunicación de hilos cruzada. En algunas realizaciones, configurar el estado de memoria intermedia de retorno 916 incluye seleccionar el tamaño y número de memorias intermedias de retorno a usar para un conjunto de operaciones de canalización.

Los restantes comandos en la secuencia de comandos difieren basándose en la canalización activa para operaciones. Basándose en una determinación de canalización 920, la secuencia de comandos se adapta a la canalización 3D 922 que comienza con el estado de canalización 3D 930 o la canalización de medios 924 que comienza en el estado de canalización de medios 940.

Los comandos para configurar el estado de canalización 3D 930 incluyen comandos de establecimiento de estado 3D para estado de memoria intermedia de vértices, estado de elemento de vértices, estado de color constante, estado de memoria intermedia de profundidad y otras variables de estado que tienen que configurarse antes de que se procesen los comandos de primitivas 3D. Los valores de estos comandos se determinan al menos en parte basándose en la API 3D particular en uso. En algunas realizaciones, los comandos del estado de canalización 3D 930 también son capaces de deshabilitar u omitir selectivamente ciertos elementos de canalización si no se usarán esos elementos.

En algunas realizaciones, el comando de la primitiva 3D 932 se usa para enviar primitivas 3D a procesar por la canalización 3D. Los comandos y parámetros asociados que se pasan al procesador gráfico a través del comando de la primitiva 3D 932 se reenvían a la función de búsqueda de vértices en la canalización de gráficos. La función de búsqueda de vértices usa los datos de comando de la primitiva 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. En algunas realizaciones, el comando de la primitiva 3D 932 se usa para realizar operaciones de vértices en primitivas 3D a través de sombreadores de vértices. Para procesar sombreadores de vértices, la canalización 3D 922 distribuye hilos de ejecución de sombreador a unidades de ejecución de procesador gráfico.

En algunas realizaciones, la canalización 3D 922 se desencadena a través de un comando o evento de ejecución 934. En algunas realizaciones, una escritura de registro desencadena la ejecución de comando. En algunas realizaciones, la ejecución se desencadena a través de un comando de 'adelante' o 'rechazar' en la secuencia de comandos. En una realización, la ejecución de comando se desencadena usando un comando de sincronización de canalización para vaciar la secuencia de comandos a través de la canalización de gráficos. La canalización 3D realizará procesamiento de geometría para las primitivas 3D. Una vez que las operaciones han finalizado, los objetos geométricos resultantes se rasterizan y el motor de píxeles colorea los píxeles resultantes. Para esas operaciones también pueden incluirse comandos adicionales para controlar operaciones de sombreado de píxel y de extremo final de píxel.

En algunas realizaciones, la secuencia de comandos de procesador gráfico 910 sigue la de la canalización de medios 924 cuando realiza operaciones de medios. En general, el uso específico y manera de programación para la canalización de medios 924 depende de las operaciones de medios o de cálculo a realizar. Operaciones de decodificación de medios específicas pueden descargarse a la canalización de medios durante la decodificación de medios. En algunas realizaciones, la canalización de medios también puede omitirse y la decodificación de medios puede realizarse en su totalidad o en parte usando recursos proporcionados por uno o más núcleos de procesamiento de fin general. En una realización, la canalización de medios también incluye elementos para operaciones de unidad de procesador gráfico de fin general (GPGPU), en donde el procesador gráfico se usa para realizar operaciones vectoriales de SIMD usando programas de sombreador de cálculo que no están explícitamente relacionados con la representación de primitivas gráficas.

En algunas realizaciones, la canalización de medios 924 está configurada de manera similar a la canalización 3D 922. Un conjunto de comandos para configurar el estado de canalización de medios 940 se distribuyen o sitúan en una cola de comandos antes de los comandos de objeto de medios 942. En algunas realizaciones, los comandos para el estado de canalización de medios 940 incluyen datos para configurar los elementos de canalización de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación de vídeo o codificación de vídeo dentro de la canalización de medios, tal como formato de codificación o decodificación. En algunas realizaciones, los comandos para el estado de canalización de medios 940 también soportan el uso de uno o más punteros a elementos de estado "indirectos" que contienen un lote de ajustes de estado.

En algunas realizaciones, los comandos de objeto de medios 942 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo a procesar. En algunas realizaciones, todos los estados de canalizaciones de medios deben ser válidos antes de emitir un comando de objeto de medios 942. Una vez que el estado de canalización está configurado y los comandos de objeto de medios 942 están en cola, la canalización de medios 924 se desencadena a través de un comando de ejecución 944 o un evento de ejecución equivalente (por ejemplo, escritura de registro). La salida de la canalización de medios 924 puede posprocesarse, a continuación, por operaciones proporcionadas por la canalización 3D 922 o la canalización de medios 924. En algunas realizaciones, las operaciones de GPGPU están configuradas y se ejecutan de manera similar a las operaciones de medios.

Arquitectura de software gráfica

La Figura 10 ilustra una arquitectura de software gráfica ilustrativa para un sistema de procesamiento de datos 1000 de acuerdo con algunas realizaciones. En algunas realizaciones, la arquitectura de software incluye una aplicación gráfica 3D 1010, un sistema operativo 1020 y al menos un procesador 1030. En algunas realizaciones, el procesador 1030 incluye un procesador gráfico 1032 y uno o más núcleo o núcleos de procesador de fin general 1034. Cada uno de la aplicación gráfica 1010 y el sistema operativo 1020 se ejecuta en la memoria de sistema 1050 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación gráfica 3D 1010 contiene uno o más programas de sombreador que incluyen instrucciones de sombreador 1012. Las instrucciones de lenguaje de sombreador pueden ser en un lenguaje de sombreador de nivel alto, tal como el Lenguaje de Sombreador de Nivel Alto (HLSL) o el Lenguaje de Sombreador de OpenGL (GLSL). La aplicación también incluye instrucciones ejecutables 1014 en un lenguaje de máquina adecuado para su ejecución por el núcleo de procesador de fin general 1034. La aplicación también incluye objetos gráficos 1016 definidos por datos de vértice.

En algunas realizaciones, el sistema operativo 1020 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo de tipo UNIX propietario o un sistema operativo de tipo UNIX de código abierto usando una variante del núcleo de Linux. El sistema operativo 1020 puede soportar una API gráfica 1022 tal como la API Direct3D, la API OpenGL o la API Vulkan. Cuando la API Direct3D está en uso, el sistema operativo 1020 usa un compilador de sombreador de extremo 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 precompilación de sombreador. En algunas realizaciones, los sombreadores de nivel alto se compilan en sombreadores de nivel bajo durante la compilación de la aplicación gráfica 3D 1010. En algunas realizaciones, las instrucciones de sombreador 1012 se proporcionan en una forma intermedia, tal como una versión de la Representación Intermedia Portable Estándar (SPIR) usada por la API Vulkan.

En algunas realizaciones, el controlador gráfico de modo de usuario 1026 contiene un compilador de sombreador de extremo final 1027 para convertir las instrucciones de sombreador 1012 a una representación específica de hardware. Cuando la API OpenGL está en uso, las instrucciones de sombreador 1012 en el lenguaje de alto nivel GLSL se pasan a un controlador gráfico de modo de usuario 1026 para su compilación. En algunas realizaciones, el controlador gráfico de modo de usuario 1026 usa funciones de modo de núcleo de sistema operativo 1028 para comunicarse con un controlador gráfico de modo de núcleo 1029. En algunas realizaciones, el controlador gráfico de modo de núcleo 1029 se comunica con el procesador gráfico 1032 para distribuir comandos e instrucciones.

Implementaciones de núcleo de IP

Uno o más aspectos de al menos una realización pueden implementarse mediante código representativo almacenado en un medio legible por máquina que representa y/o define 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 se leen por una máquina, las instrucciones pueden provocar que la máquina fabrique la lógica para realizar las técnicas descritas en este documento. Tales representaciones, conocidas como "núcleos de IP", son unidades reutilizables de lógica para un circuito integrado que puede almacenarse en un medio legible por máquina tangible como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware puede suministrarse a diversas instalaciones de cliente o de fabricación, que cargan el modelo de hardware en máquinas de fabricación que fabrican el circuito integrado. El circuito integrado puede fabricarse de tal forma que el circuito realiza operaciones descritas en asociación con cualquiera de las realizaciones descritas en este documento.

La Figura 11 es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 1100 que puede usarse para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo de IP 1100 puede usarse para generar diseños modulares y reutilizables que pueden incorporarse en un diseño mayor o usarse para construir todo un circuito integrado (por ejemplo, un circuito integrado de SOC). Una instalación de diseño 1130 puede generar una simulación de software 1110 de un diseño de núcleo de IP en un lenguaje de programación de nivel alto (por ejemplo, C/C++). La simulación de software 1110 puede usarse para diseñar, probar y verificar el comportamiento del núcleo de IP usando un modelo de simulación 1112. El modelo de simulación 1112 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Un diseño de nivel de transferencia de registro (RTL) 1115 puede crearse y sintetizarse, a continuación, 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, incluyendo la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño de RTL 1115, también pueden crearse, diseñarse o sintetizarse diseños de nivel inferior a nivel de lógica o nivel de transistor. Por lo tanto, los detalles particulares del diseño inicial y simulación pueden variar.

El diseño de RTL 1115 o equivalente puede sintetizarse adicionalmente mediante 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 puede simularse o probarse adicionalmente para verificar el diseño de núcleo de IP. El diseño de núcleo de IP puede almacenarse para su entrega a una instalación de fabricación de terceros 1165 usando memoria no volátil 1140 (por ejemplo, disco duro, memoria flash o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño de núcleo de IP puede transmitirse (por ejemplo, a través de la Internet) a través de una conexión por cable 1150 o una conexión inalámbrica 1160. La instalación de fabricación 1165 puede fabricar, a continuación, un circuito integrado que se basa al menos en parte en el diseño de núcleo de IP. El circuito integrado fabricado puede configurarse para realizar operaciones de acuerdo con al menos una realización descrita en este documento.

Sistema ilustrativo en un circuito integrado de chip

Las Figuras 12-14 ilustran circuitos integrados ilustrativos y procesadores gráficos asociados que pueden fabricarse usando uno o más núcleos de IP, de acuerdo con diversas realizaciones descritas en este documento. Además de lo que se ilustra, pueden incluirse otra lógica y circuitos, incluyendo procesadores/núcleos gráficos adicionales, controladores de interfaz periférica o núcleos de procesador de fin general.

La Figura 12 es un diagrama de bloques que ilustra un sistema en un circuito integrado de chip 1200 ilustrativo que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 1200 ilustrativo incluye uno o más procesador o procesadores de aplicación 1205 (por ejemplo, CPU), al menos un procesador gráfico 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 de IP modular de la misma o múltiples diferentes instalaciones de diseño. El circuito integrado 1200 incluye lógica periférica o de bus que incluye un controlador de USB 1225, un controlador de UART 1230, un controlador de SPI/SDIO 1235 y un controlador de I2S/I2C 1240. Adicionalmente, 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 la industria móvil (MIPI) 1255. Puede proporcionarse almacenamiento por un subsistema de memoria flash 1260 que incluye memoria flash y un controlador de memoria flash. La interfaz de memoria puede proporcionarse a través de un controlador de memoria 1265 para acceder a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad embebido 1270.

La Figura 13 es un diagrama de bloques que ilustra un procesador gráfico ilustrativo 1310 de un sistema en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El procesador gráfico 1310 puede ser una variante del procesador gráfico 1210 de la Figura 12. El procesador gráfico 1310 incluye un procesador de vértices 1305 y uno o más procesador o procesadores de fragmentos 1315A-1315N (por ejemplo, 1315A, 1315B, 1315C, 1315D a 1315N-1 y 1315N). El procesador gráfico 1310 puede ejecutar diferentes

programas de sombreador a través de lógica separada, de tal forma que el procesador de vértices 1305 se optimiza para ejecutar operaciones para programas de sombreador de vértices, mientras el uno o más procesador o procesadores de fragmentos 1315A-1315N ejecutan operaciones de sombreado de fragmento (por ejemplo, píxel) para programas de sombreador de fragmentos o píxeles. El procesador de vértices 1305 realiza la etapa de procesamiento de vértices de la canalización de gráficos 3D y genera primitivas y datos de vértice. El procesador o procesadores de fragmentos 1315A-1315N usan la primitiva y datos de vértice generados por el procesador de vértices 1305 para producir una memoria intermedia de fotogramas que se visualiza en un dispositivo de visualización. En una realización, el procesador o procesadores de fragmentos 1315A-1315N se optimizan para ejecutar programas de sombreador de fragmentos según se proporcionan en la API OpenGL, que pueden usarse para realizar operaciones similares que un programa de sombreador de píxeles según se proporciona para en la API Direct 3D.

El procesador gráfico 1310 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 1320A-1320B, memoria o memorias caché 1325A-1325B e interconexión o interconexiones de circuito 1330A-1330B. La una o más MMU 1320A-1320B proporcionan correlación de dirección virtual con física para el procesador gráfico 1310, incluyendo para el procesador de vértices 1305 y/o el procesador o procesadores de fragmentos 1315A-1315N, que pueden referenciar datos de vértice o de imagen/textura almacenados en memoria, además de datos de vértice o de imagen/textura almacenados en la una o más memoria o memorias caché 1325A-1325B. En una realización, la una o más MMU 1320A-1320B pueden sincronizarse con otras MMU dentro del sistema, incluyendo una o más MMU asociadas con el uno o más procesador o procesadores de aplicación 1205, procesador de imágenes 1215 y/o procesador de vídeo 1220 de la **Figura 12,** de tal forma que cada procesador 1205-1220 puede participar en un sistema de memoria virtual compartida o unificada. La una o más interconexión o interconexiones de circuito 1330A-1330B habilitan que procesador gráfico 1310 interactúe con otros núcleos de IP dentro del SoC, o bien a través de un bus interno del SoC o bien a través de una conexión directa, de acuerdo con realizaciones.

La Figura 14 es un diagrama de bloques que ilustra un procesador gráfico 1410 ilustrativo adicional de un sistema en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El procesador gráfico 1410 puede ser una variante del procesador gráfico 1210 de **la Figura 12**. El procesador gráfico 1410 incluye la una o más MMU 1320A-1320B, memoria o memorias caché 1325A-1325B, e interconexión o interconexiones de circuito 1330A-1330B del circuito integrado 1300 de **la Figura 13**.

El procesador gráfico 1410 incluye uno o más núcleo o núcleos de sombreador 1415A-1415N (por ejemplo, 1415A, 1415B, 1415C, 1415D, 1415E, 1415F a 1315N-1 y 1315N), que proporciona una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo de núcleo puede ejecutar todo tipo de código de sombreador programable, incluyendo código de programa de sombreador para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. El número exacto de núcleos de sombreador presentes puede variar entre realizaciones e implementaciones. Adicionalmente, el procesador gráfico 1410 incluye un gestor de tareas entre núcleos 1405, que actúa como un distribuidor de hilos para distribuir hilos de ejecución a uno o más núcleo o núcleos de sombreador 1415A-1415N y una unidad de mosaico 1418 para acelerar operaciones de mosaico para representación basada en losa, en las que las operaciones de representación para una escena se subdividen en espacio de imagen, por ejemplo, para explotar coherencia espacial local dentro de una escena o para optimizar el uso de memorias caché internas.

ARQUITECTURAS DE VIRTUALIZACIÓN GRÁFICA ILUSTRATIVAS

Algunas realizaciones de la invención se implementan en una plataforma que utiliza virtualización de unidad de procesador gráfico (GPU) completa. Como tal, a continuación se proporciona una vista general de las técnicas de virtualización de GPU empleadas en una realización de la invención, seguida por una descripción detallada de un aparato y método para sombreado de tabla de paginación accionada por patrón.

Una realización de la invención emplea un entorno de virtualización de GPU completo que ejecuta un controlador de gráficos nativo en el invitado, y paso a través mediado que consigue tanto buen rendimiento, escalabilidad como asilamiento seguro entre invitados. Esta realización presenta una GPU completa virtual a cada máquina virtual (VM) que puede acceder directamente a recursos de rendimiento crítico sin intervención del hipervisor en la mayoría de los casos, mientras las operaciones privilegiadas del invitado se capturan y emulan con coste mínimo. En una realización, una GPU virtual (vGPU), con características de GPU completas, se presenta a cada VM. Las VM pueden acceder directamente a recursos de rendimiento crítico, sin intervención del hipervisor en la mayoría de los casos, mientras las operaciones privilegiadas del invitado se capturan y emulan para proporcionar un aislamiento seguro entre las VM. El contexto de vGPU se conmuta por cuanto, para compartir la GPU física entre múltiples VM.

La Figura 15 ilustra una arquitectura de sistema de nivel alto en la que pueden implementarse realizaciones de la invención que incluye una unidad de procesamiento gráfico (GPU) 1500, una unidad de procesamiento central (CPU) 1520 y una memoria de sistema 1510 compartida entre la GPU 1500 y la CPU 1520. Un motor de representación 1502 busca comandos de GPU en una memoria intermedia de comandos 1512 en la memoria de sistema 1510, para acelerar la representación gráfica usando diversas características diferentes. El motor de visualización 1504 busca datos de píxel en la memoria intermedia de fotogramas 1514 y, a continuación, envía los datos de píxel a monitores externos para su visualización.

Ciertas arquitecturas usan memoria de sistema 1510 como memoria gráfica, mientras que otras GPU pueden usar memoria en chip. La memoria de sistema 1510 puede correlacionarse con múltiples espacios de direcciones virtuales mediante tablas de paginación de GPU 1506. Un espacio de direcciones virtuales globales de 2 GB, denominado memoria gráfica global, accesible tanto desde la GPU 1500 como la CPU 1520, se correlaciona a través de tablas de paginación globales. Los espacios de memoria gráfica locales se soportan en forma de múltiples espacios de direcciones virtuales locales de 2 GB, pero se limitan únicamente para acceder desde el motor de representación 1502, a través de tablas de paginación locales. La memoria gráfica global es principalmente la memoria intermedia de fotogramas 1514, pero también sirve como las memoria intermedia de comandos 1512. Se hacen gran cantidad de accesos a datos a memoria gráfica local cuando la aceleración de hardware está en progreso. Se emplean mecanismos de tabla de paginación similares por las GPU con memoria en chip.

En una realización, la CPU 1520 programa la GPU 1500 a través de comandos específicos de GPU, mostrados en la **Figura 15,** en un modelo de productor-consumidor. El controlador de gráficos programa comandos de GPU en la memoria intermedia de comandos 1512, incluyendo una memoria intermedia primaria y una memoria intermedia por lotes, de acuerdo con API de programación de alto nivel como OpenGL y DirectX. La GPU 1500, a continuación, busca y ejecuta los comandos. La memoria intermedia primaria, una memoria intermedia de anillo, puede encadenar otras memorias intermedias por lotes juntas. Los términos "memoria intermedia primaria" y "memoria intermedia de anillo" se usan indistintamente en lo sucesivo. La memoria intermedia por lotes se usa para transmitir la mayoría de los comandos (hasta ~98 %) por modelo de programación. Una tupla de registro (cabeza, cola) se usa para controlar la memoria intermedia de anillo. En una realización, la CPU 1520 envía los comandos a la GPU 1500 actualizando la cola, mientras la GPU 1500 busca comandos en la cabeza, y, a continuación, notifica a la CPU 1520 actualizando la cabeza, después de que los comandos han finalizado su ejecución.

Como se ha mencionado, una realización de la invención se implementa en una plataforma de virtualización de GPU completa con paso a través mediado. Como tal, cada VM se presenta con a GPU completa para ejecutar un controlador de gráficos nativo dentro de una VM. El desafío, sin embargo, es significante en tres formas: (1) complejidad en la virtualización de toda una GPU moderna sofisticada, (2) rendimiento debido a múltiples VM compartiendo la GPU, y (3) aislamiento seguro entre las VM sin ningún compromiso.

La Figura 16 ilustra una arquitectura de virtualización de GPU de acuerdo con una realización de la invención que incluye un hipervisor 1610 que se ejecuta en una GPU 1600, una máquina virtual (VM) privilegiada 1620 y una o más VM de usuario 1631 - 1632. Un módulo auxiliar de virtualización 1611 que se ejecuta en el hipervisor 1610 extiende la gestión de memoria para incluir tablas de paginación extendidas (EPT) 1614 para las VM de usuario 1631-1632 y una unidad de gestión de memoria virtual privilegiada (PVMMU) 1612 para la VM privilegiada 1620, para implementar las políticas de captura y paso a través. En una realización, cada VM 1620, 1631-1632 ejecuta el controlador de gráficos nativo 1628 que puede acceder directamente a los recursos de rendimiento crítico de la memoria intermedia de fotogramas y la memoria intermedia de comandos, con particionamiento de recursos como se describe a continuación. Para proteger los recursos privilegiados, es decir, los registros de E/S y las PTE, correspondientes accesos desde los controladores gráficos 1628 en las VM de usuario 1631-1632 y la VM privilegiada 1620, se capturan y reenvían al mediador de virtualización 1622 en la VM privilegiada 1620 para su emulación. En una realización, el mediador de virtualización 1622 usa hiperllamadas para acceder a la GPU física 1600 como se ilustra.

Además, en una realización, el mediador de virtualización 1622 implementa un planificador de GPU 1626, que se ejecuta simultáneamente con el planificador de CPU 1616 en el hipervisor 1610, para compartir la GPU física 1600 entre las VM 1631-1632. Una realización usa la GPU física 1600 para ejecutar directamente todos los comandos enviados desde una VM, de forma que evita la complejidad de emular el motor de representación, que es la parte más compleja dentro de la GPU. Mientras tanto, el paso a través de recurso tanto de la memoria intermedia de fotogramas como la memoria intermedia de comandos minimiza la intervención del hipervisor 1610 en los accesos de CPU, mientras que el planificador de GPU 1626 garantiza a cada VM un cuanto para ejecución de GPU directa. En consecuencia, la realización ilustrada consigue un buen rendimiento cuando se comparte la GPU entre múltiples VM.

En una realización, el auxiliar de virtualización 1611 captura o pasa a través de forma selectiva acceso de invitado de ciertos recursos de GPU. El auxiliar de virtualización 1611 manipula las entradas de la EPT 1614 para presentar u ocultar de forma selectiva un intervalo de direcciones específico a las VM de usuario 1631-1632, mientras usa un bit reservado de las PTE en la PVMMU 1612 para la VM privilegiada 1620, para capturar o pasar a través de forma selectiva accesos de invitado a un intervalo de direcciones específico. En ambos casos, se capturan los accesos de entrada/salida periféricos (PIO). Todos los accesos capturados se reenvían al mediador de virtualización 1622 para emulación mientras el mediador de virtualización 1611 usa hiperllamadas para acceder a la GPU física 1600.

Como se ha mencionado, en una realización, el mediador de virtualización 1622 emula GPU virtuales (vGPU) 1624 para accesos de recursos privilegiados, y lleva a cabo conmutaciones de contexto entre las vGPU 1624. Mientras tanto, el controlador de gráficos 1628 de la VM privilegiada 1620 se usa para inicializar el dispositivo físico y para gestionar la potencia. Una realización toma un modelo de liberación flexible, implementando el mediador de virtualización 1622 como un módulo de núcleo en la VM privilegiada 1620, para facilitar la unión entre el mediador de virtualización 1622 y el hipervisor 1610.

Un mecanismo de planificación de CPU/GPU dividida se implementa a través del planificador de CPU 1616 y el planificador de GPU 1626. Esto se hace debido a que el coste de una conmutación de contexto de GPU puede ser 1000 veces superior al coste de una conmutación de contexto de CPU (por ejemplo, ~700 us vs. ~300 ns). Además, el número de los núcleos de la CPU probablemente difiere del número de los núcleos de GPU en un sistema informático. En consecuencia, en una realización, un planificador de GPU 1626 se implementa de forma separada del planificador de CPU 1616 existente. El mecanismo de planificación de división conduce al requisito de accesos concurrentes a los recursos tanto desde la CPU como la GPU. Por ejemplo, mientras la CPU está accediendo a la memoria gráfica de VM1 1631, la GPU puede estar accediendo a la memoria gráfica de VM2 1632, simultáneamente.

Como se ha analizado anteriormente, en una realización, un controlador de gráficos nativo 1628 se ejecuta dentro de cada VM 1620, 1631-1632, que accede directamente a una porción de los recursos de rendimiento crítico, con operaciones privilegiadas emuladas por el mediador de virtualización 1622. El mecanismo de planificación de división conduce al diseño de particionamiento de recursos descrito a continuación. Para soportar mejor el particionamiento de recursos, una realización reserva una ventana de registro de E/S con correlación de memoria (MMIO) para transmitir la información de particionamiento de recursos a la VM.

En una realización, la ubicación y definición de virt_info se ha promovido a la especificación de hardware como una extensión de virtualización, de forma que el controlador de gráficos 1628 trata la extensión de forma nativa, y futuras generaciones de GPU siguen la especificación para compatibilidad hacia atrás.

Mientras se ilustra como un componente separado en la **Figura 16,** en una realización, la VM privilegiada 1620 que incluye el mediador de virtualización 1622 (y sus instancias de vGPU 1624 y planificador de GPU 1626) se implementa como un módulo dentro del hipervisor 1610.

En una realización, el mediador de virtualización 1622 gestiona las vGPU 1624 de todas las VM, capturando y emulando las operaciones privilegiadas. El mediador de virtualización 1622 trata las interrupciones de GPU física, y puede generar interrupciones virtuales a las VM 1631-1632 designadas. Por ejemplo, una interrupción de finalización física de la ejecución de comando puede desencadenar una interrupción de finalización física, entregada al propietario de la representación. La idea de emular una instancia de vGPU por semántica es simple; sin embargo, la implementación implica un gran esfuerzo de ingeniería y un entendimiento profundo de la GPU 1600. Por ejemplo, aproximadamente 700 registros de E/S pueden accederse por ciertos controladores gráficos.

En una realización, el planificador de GPU 1626 implementa una política de calidad de servicio (QoS) de grano grueso. Puede seleccionarse un cuanto de tiempo particular como un sector de tiempo para cada VM 1631-1632 para compartir los recursos de la GPU 1600. Por ejemplo, en una realización, se selecciona un cuanto de tiempo de 16 ms como el sector de tiempo de planificación, porque este valor resulta en una perceptibilidad humana baja a cambios de imagen. También se selecciona un cuanto de tiempo relativamente grande de este tipo porque el coste de la conmutación de contexto de GPU es 1000X mayor que el de la conmutación de contexto de CPU, de forma que no puede ser tan pequeño como el sector de tiempo en el planificador de CPU 1616. Los comandos desde una VM 1631-1632 se envían a la GPU 1600 de forma continua, hasta que el invitado/VM se queda sin su sector de tiempo. En una realización, el planificador de GPU 1626 espera que la memoria intermedia de anillo de invitado esté en reposo antes de conmutar, porque la mayoría de GPU hoy en día son no preventivas, lo que puede tener un impacto en la equidad. Para minimizar la tara de espera, puede implementarse un mecanismo de control de flujo de grano grueso, rastreando la notificación de comando para garantizar que los comandos apilados, en cualquier momento, están dentro de un cierto límite. Por lo tanto, el desplazamiento de tiempo entre el sector de tiempo asignado y el tiempo de ejecución es relativamente pequeño, en comparación con el cuanto grande, de forma que se consigue una política de QoS de grano grueso.

En una realización, en una conmutación de contexto de representación, el estado de canalización interno y estados de registro de E/S se guardan y restauran, y se realiza un vaciado de memoria caché/TLB, cuando se conmuta el motor de representación entre las vGPU 1624. El estado de canalización interno es invisible a la CPU, pero puede guardarse y restaurarse a través de comandos de GPU. Guardar/restaurar los estados de registro de E/S puede conseguirse a través de lecturas/escrituras en una lista del registros en el contexto de representación. Las memorias caché internas y Memorias Intermedias de Traducción Anticipadas (TLB), incluidas en las GPU modernas para acelerar accesos a datos y traducciones de direcciones, deben vaciarse usando comandos en la conmutación de contexto de representación, para garantizar el aislamiento y exactitud. Las etapas usadas para conmutar un contexto en una realización son: 1) guardar estados de E/S actuales, 2) vaciar el contexto actual, 3) usar los comandos adicionales para guardar el contexto actual, 4) usar los comandos adicionales para restaurar el nuevo contexto, y 5) restaurar el estado de E/S del nuevo contexto.

Como se ha mencionado, una realización usa una memoria intermedia de anillo especializada para transportar los comandos de GPU adicionales. La memoria intermedia de anillo de invitado (auditada) puede reutilizarse para rendimiento, pero no es seguro insertar directamente los comandos en la memoria intermedia de anillo de invitado, porque la CPU puede continuar poniendo en cola más comandos, conduciendo a contenido sobrescrito. Para evitar una condición de carrera, una realización conmuta desde la memoria intermedia de anillo de invitado a su propia

memoria intermedia de anillo especializada. Al final de la conmutación de contexto, esta realización conmuta desde la memoria intermedia de anillo especializada a la memoria intermedia de anillo de invitado de la nueva VM.

Una realización reutiliza el controlador de gráficos de la VM privilegiada 1620 para inicializar el motor de visualización y, a continuación, gestiona el motor de visualización para mostrar diferentes memorias intermedias de fotogramas de VM.

Cuando dos vGPU 1624 tienen la misma resolución, únicamente se conmutan las ubicaciones de memoria intermedia de fotogramas. Para diferentes resoluciones, la VM privilegiada puede usar un escalar de hardware, una característica común en las GPU modernas, para escalar la resolución ascendente y descendentemente automáticamente. Ambas técnicas tardan unos meros milisegundos. En muchos casos, puede no necesitarse la gestión de visualización tal como cuando la VM no se muestra en el visualizador físico (por ejemplo, cuando se aloja en los servidores remotos).

Como se ilustra en **La Figura 16,** una realización pasa a través los accesos a la memoria intermedia de fotogramas y memoria intermedia de comandos para acelerar operaciones de rendimiento crítico de una VM 1631-1632. Para el espacio de memoria gráfica global, con tamaño de 2 GB, pueden emplearse técnicas de particionamiento de recursos de memoria gráfica y aumento de espacio de direcciones. Para los espacios de memoria gráfica locales, cada uno también con un tamaño de 2GB, una memoria gráfica local por VM puede implementarse a través de la conmutación de contexto de representación, debido a que la memoria gráfica local es accesible únicamente por la GPU 1600.

Como se ha mencionado, una realización particiona la memoria gráfica global entre las VM 1631-1632. Como se ha explicado anteriormente, un mecanismo de planificación de CPU/GPU dividida requiere que la memoria gráfica global de diferentes VM pueda accederse simultáneamente por la CPU y la GPU, de forma que cada VM debe presentarse en cualquier momento con sus propios recursos, conduciendo al enfoque de particionamiento de recursos para la memoria gráfica global.

La Figura 17 ilustra detalles adicionales para una realización de una arquitectura de virtualización gráfica 1700 que incluye múltiples VM, por ejemplo, la VM 1730 y la VM 1740, gestionadas por el hipervisor 1710, incluyendo acceso a una matriz completa de características de GPU en una GPU 1720. En diversas realizaciones, el hipervisor 1710 puede habilitar que la VM 1730 o la VM 1740 utilice una memoria gráfica y otros recursos de GPU para la virtualización de GPU. Una o más GPU virtuales (vGPU), por ejemplo, las vGPU 1760A y 1760B, pueden acceder a toda la funcionalidad proporcionada por el hardware de la GPU 1720 basándose en la tecnología de virtualización de GPU. En diversas realizaciones, el hipervisor 1710 puede rastrear, gestionar recursos y ciclos de vida de las vGPU 1760A y 1760B como se describe en este documento.

En algunas realizaciones, las vGPU 1760A-B pueden incluir dispositivos de GPU virtual presentados a las VM 1730, 1740 y pueden usarse para interactuar con controladores de GPU nativos (por ejemplo, como se ha descrito anteriormente con respecto a la **Figura 16**). La VM 1730 o la VM 1740 puede acceder, a continuación, a toda la matriz de características de GPU y usar los dispositivos de GPU virtual en las vGPU 1760A-B para acceder a procesadores gráficos virtuales. Por ejemplo, una vez que la VM 1730 se captura en el hipervisor 1710, el hipervisor 1710 puede manipular una instancia de vGPU, por ejemplo, la vGPU 1760A, y determinar si la VM 1730 puede acceder a los dispositivos de GPU virtual en la vGPU 1760A. El contexto de vGPU puede conmutarse por cuanto o evento. En algunas realizaciones, la conmutación de contexto puede suceder por motor de representación de GPU tal como el motor de representación 3D 1722 o el motor de representación de blitter 1724. La conmutación periódica permite que múltiples VM compartan una GPU física de una manera que es transparente a las cargas de trabajo de las VM.

La virtualización de GPU puede tomar diversas formas. En algunas realizaciones, la VM 1730 puede habilitarse con paso a través de dispositivo, en donde toda la GPU 1720 se presenta a la VM 1730 como si estuvieran conectadas directamente. Al igual que un único núcleo de unidad de procesamiento central (CPU) puede asignarse para su uso exclusivo por la VM 1730, la GPU 1720 también puede asignarse para su uso exclusivo por la VM 1730, por ejemplo, incluso durante un tiempo limitado. Otro modelo de virtualización es uso compartido de tiempo, en donde la GPU 1720 o porciones de la misma pueden compartirse por múltiples VM, por ejemplo, la VM 1730 y la VM 1740, en una manera de multiplexación. En otras realizaciones el aparato 1700 también puede usar otros modelos de virtualización de GPU. En diversas realizaciones, la memoria gráfica asociada con la GPU 1720 puede particionarse, y asignarse a diversas vGPU 1760A-B en el hipervisor 1710.

En diversas realizaciones, las VM o la GPU 1720 pueden usar tablas de traducción de gráficos (GTT) para correlacionar memoria de procesador gráfico con memoria de sistema o traducir direcciones virtuales de GPU a direcciones físicas. En algunas realizaciones, el hipervisor 1710 puede gestionar la correlación de memoria gráfica a través de las GTT de sombra, y las GTT de sombra pueden mantenerse en una instancia de vGPU, por ejemplo, la vGPU 1760A. En diversas realizaciones, cada VM puede tener una correspondiente GTT de sombra para mantener la correlación entre direcciones de memoria gráfica y direcciones de memoria física, por ejemplo, direcciones de memoria de máquina en un entorno de virtualización. En algunas realizaciones, la GTT de sombra puede compartirse y mantener las correlaciones para múltiples VM. En algunas realizaciones, cada VM 1730 o VM 1740 puede incluir GTT tanto de por proceso como globales.

En algunas realizaciones, el aparato 1700 puede usar la memoria de sistema como la memoria gráfica. La memoria de sistema puede correlacionarse con múltiples espacios de direcciones virtuales mediante tablas de paginación de GPU. El aparato 1700 puede soportar espacio de memoria gráfica global y espacio de direcciones de memoria gráfica por proceso. El espacio de memoria gráfica global puede ser un espacio de direcciones virtuales, por ejemplo, 2 GB, correlacionado a través de una tabla de traducción de gráficos global (GGTT). La porción inferior de este espacio de direcciones en ocasiones se denomina la apertura, accesible tanto desde la GPU 1720 como la CPU (no mostrada). La porción superior de este espacio de direcciones se denomina espacio de memoria gráfica alta o espacio de memoria gráfica oculta, que puede usarse únicamente por la GPU 1720. En diversas realizaciones, la VM 1730, la VM 1740, el hipervisor 1710 o la GPU 1720 pueden usar tablas de traducción de gráficos global de sombra (SGGTT) para traducir direcciones de memoria gráfica a respectivas direcciones de memoria de sistema basándose en un espacio de direcciones de memoria global.

En virtualización completa de GPU, un esquema de particionamiento de espacio de memoria gráfica global estático puede hacer frente a un problema de escalabilidad. Por ejemplo, para un espacio de memoria gráfica global de 2 GB, los primeros 512 megabytes (MB) de espacio de direcciones virtuales pueden reservarse para apertura, y el resto del mismo, 1536 MB, pueden convertirse en el espacio de memoria gráfica alta (oculta). Con el esquema de particionamiento de espacio de memoria gráfica global estático, cada VM con virtualización completa de GPU habilitada puede asignarse con 128 MB de apertura y 384 MB de espacio de memoria gráfica alta. Por lo tanto, en el espacio de memoria gráfica global de 2 GB puede acomodarse únicamente un máximo de cuatro VM.

Además del problema de escalabilidad, las VM con espacio de memoria gráfica limitado también pueden sufrir degradación de rendimiento. En ocasiones, puede observarse una degradación de rendimiento grave en algunas cargas de trabajo de muchos medios de una aplicación de medios cuando usa aceleración de hardware de medios de GPU de forma extensiva. Como un ejemplo, para decodificar un flujo de bits de H.264/Codificación de Vídeo Avanzada (AVC) de 1080p de un canal, pueden necesitarse al menos 40 MB de memoria gráfica. Por lo tanto, para 10 canales de decodificación de flujos de bits de H264/AVC de 1080p, pueden necesitarse al menos 400 MB de espacio de memoria gráfica. Mientras tanto, puede tener que establecerse aparte algún espacio de memoria gráfica para conversión de composición/color de superficie, conmutando memoria intermedia de fotogramas de visualización durante el proceso de decodificación, etc. En este caso, 512 MB de espacio de memoria gráfica por VM puede ser insuficiente para que una VM ejecute múltiple codificación o decodificación de vídeo.

En diversas realizaciones, el aparato 100 puede conseguir sobreasignación de memoria gráfica de GPU con SGGTT bajo demanda. En algunas realizaciones, el hipervisor 1710 puede construir SGGTT bajo demanda, que puede incluir todas las traducciones que hay que usar para direcciones virtuales de memoria gráfica desde las VM propietarias de diferentes componentes de GPU.

En diversas realizaciones, al menos una VM gestionada por el hipervisor 1710 puede asignarse con más que direcciones de memoria gráfica globales particionada estáticas así como memoria. En algunas realizaciones, al menos una VM gestionada por el hipervisor 1710 puede asignarse con o ser capaz de acceder a todo el espacio de direcciones de memoria gráfica alta. En algunas realizaciones, al menos una VM gestionada por el hipervisor 1710 puede asignarse con o ser capaz de acceder a todo el espacio de direcciones de memoria gráfica.

El hipervisor/VMM 1710 puede usar el analizador de comandos 1718 para detectar el conjunto de trabajo de memoria potencial de un motor de representación de GPU para los comandos enviados por la VM 1730 o la VM 1740. En diversas realizaciones, la VM 1730 puede tener respectivas memorias intermedias de comandos (no mostradas) para mantener comandos de la carga de trabajo 3D 1732 o la carga de trabajo de medios 1734. De manera similar, la VM 1740 puede tener respectivas memorias intermedias de comandos (no mostradas) para mantener comandos de la carga de trabajo 3D 1742 o la carga de trabajo de medios 1744. En otras realizaciones, la VM 1730 o la VM 1740 puede tener otros tipos de cargas de trabajo de gráficos.

En diversas realizaciones, el analizador de comandos 1718 puede explorar un comando de una VM y determinar si el comando contiene operandos de memoria. Si es que sí, el analizador de comandos puede leer las correlaciones de espacio de memoria gráfica relacionadas, por ejemplo, desde una GTT para la VM, y, a continuación, escribir las mismas en una porción específica de carga de trabajo de la SGGTT. Después de que se explora toda una memoria intermedia de comandos de una carga de trabajo, puede generarse o actualizarse la SGGTT que mantiene correlaciones de espacio de direcciones de memoria asociadas con esta carga de trabajo. Adicionalmente, explorando los comandos que hay que ejecutar de la VM 1730 o la VM 1740, el analizador de comandos 1718 también puede mejorar la seguridad de operaciones de GPU, tal como mitigando operaciones maliciosas.

En algunas realizaciones, puede generarse una SGGTT para mantener traducciones para todas las cargas de trabajo de todas las VM. En algunas realizaciones, puede generarse una SGGTT para mantener traducciones para todas las cargas de trabajo, por ejemplo, de únicamente una VM. La porción de SGGTT específica de carga de trabajo puede construirse bajo demanda por el analizador de comandos 1718 para mantener las traducciones para una carga de trabajo específica, por ejemplo, la carga de trabajo 3D 1732 de la VM 1730 o la carga de trabajo de medios 1744 de la VM 1740. En algunas realizaciones, el analizador de comandos 1718 puede insertar la SGGTT en la cola de SGGTT 1714 e insertar la correspondiente carga de trabajo en la cola de carga de trabajo 1716.

En algunas realizaciones, el planificador de GPU 1712 puede construir tales SGGTT bajo demanda en el momento de ejecución. Un motor de hardware específico puede usar únicamente una pequeña porción del espacio de direcciones de memoria gráfica asignado a la VM 1730 en el momento de ejecución, y la conmutación de contexto de GPU sucede de forma infrecuente. Para aprovechar tales características de GPU, el hipervisor 1710 puede usar la SGGTT para que la VM 1730 únicamente mantenga las traducciones en ejecución y que hay que ejecutar para diversos componentes de GPU en lugar de toda la porción del espacio de direcciones de memoria gráfica global asignada a la VM 1730.

El planificador de GPU 1712 para la GPU 1720 puede separarse del planificador para CPU en el aparato 1700. Para aprovechar el paralelismo de hardware en algunas realizaciones, el planificador de GPU 1712 puede planificar las cargas de trabajo de forma separada para diferentes motores de GPU, por ejemplo, el motor de representación 3D 1722, el motor de representación de blitter 1724, el motor de representación de transmisor de comandos de vídeo (VCS) 1726 y el motor de representación de transmisor de comandos mejorado de vídeo (VECS) 1728. Por ejemplo, la VM 1730 puede ser intensiva en 3D, y puede necesitarse que la carga de trabajo 3D 1732 se planifique al motor de representación 3D 1722 en un momento. Mientras tanto, la VM 1740 puede ser intensiva en medios, y puede necesitarse que la carga de trabajo de medios 1744 se planifique al motor de representación de VCS 1726 y/o al motor de representación de VECS 1728. En este caso, el planificador de GPU 1712 puede planificar la carga de trabajo 3D 1732 de la VM 1730 y la carga de trabajo de medios 1744 de la VM 1740 de forma separada.

En diversas realizaciones, el planificador de GPU 1712 puede rastrear las SGGTT en ejecución usadas por respectivos motores de representación en la GPU 1720. En este caso, el hipervisor 1710 puede retener una SGGTT por motor de representación para rastrear todos los conjuntos de trabajo de memoria gráfica en ejecución en respectivos motores de representación. En algunas realizaciones, el hipervisor 1710 puede retener una única SGGTT para rastrear todos los conjuntos de trabajo de memoria gráfica en ejecución para todos los motores de representación. En algunas realizaciones, tal rastreo puede basarse en una cola de SGGTT en ejecución separada (no mostrada). En algunas realizaciones, tal rastreo puede basarse en marcas en la cola de SGGTT 1714, por ejemplo, usando un registro. En algunas realizaciones, tal rastreo puede basarse en marcas en la cola de carga de trabajo 1716, por ejemplo, usando un registro.

Durante el proceso de planificación, el planificador de GPU 1712 puede examinar la SGGTT de la cola de SGGTT 1714 para una carga de trabajo que hay que planificar de la cola de carga de trabajo 1716. En algunas realizaciones, para planificar la siguiente VM para un motor de representación particular, el planificador de GPU 1712 puede comprobar si los conjuntos de trabajo de memoria gráfica de la carga de trabajo particular usada por la VM para ese motor de representación entran en conflicto con los conjuntos de trabajo de memoria gráfica en ejecución o que hay que ejecutar por ese motor de representación. En otras realizaciones, tales comprobaciones de conflicto pueden extenderse para comprobar con los conjuntos de trabajo de memoria gráfica en ejecución o que hay que ejecutar por todos los demás motores de representación. En diversas realizaciones, tales comprobaciones de conflicto pueden basarse en las correspondientes SGGTT en cola de SGGTT 1714 o basándose en SGGTT retenidas por el hipervisor 1710 para rastrear todos los conjuntos de trabajo de memoria gráfica en ejecución en respectivos motores de representación como se analiza en este punto anteriormente.

Si no existe ningún conflicto, el planificador de GPU 1712 puede integrar juntos los conjuntos de trabajo de memoria gráfica en ejecución y que hay que ejecutar. En algunas realizaciones, también puede generarse una SGGTT resultante para los conjuntos de trabajo de memoria gráfica en ejecución y que hay que ejecutar para el motor de representación particular y almacenarse, por ejemplo, en la cola de SGGTT 1714 o en otros datos medios de almacenamiento. En algunas realizaciones, también puede generarse y almacenarse una SGGTT resultante para los conjuntos de trabajo de memoria gráfica en ejecución y que hay que ejecutar para todos los motores de representación asociados con una VM si las direcciones de memoria gráfica de todas estas cargas de trabajo no entran en conflicto entre sí.

Antes de enviar una carga de trabajo de VM seleccionada a la GPU 1720, el hipervisor 1710 puede escribir correspondientes páginas de SGGTT en la GPU 1720, por ejemplo, en las tablas de traducción de gráficos 1750. Por lo tanto, el hipervisor 1710 puede habilitar que esta carga de trabajo se ejecute con correlaciones correctas en el espacio de memoria gráfica global. En diversas realizaciones, todas de tales entradas de traducción pueden escribirse en las tablas de traducción de gráficos 1750, o bien en el espacio de memoria inferior 1754 o bien el espacio de memoria superior 1752. Las tablas de traducción de gráficos 1750 pueden contener tablas separadas por VM para mantener estas entradas de traducción en algunas realizaciones. Las tablas de traducción de gráficos 1750 también pueden contener tablas separadas por motor de representación para mantener estas entradas de traducción en otras realizaciones. En diversas realizaciones, las tablas de traducción de gráficos 1750 pueden contener, al menos, direcciones de memoria gráfica que hay que ejecutar.

Sin embargo, si existe un conflicto determinado por el planificador de GPU 1712, el planificador de GPU 1712 puede diferir, a continuación, la planificación de esa VM, e intenta planificar otra carga de trabajo de la misma o una diferente VM en su lugar. En algunas realizaciones, tal conflicto puede detectarse si dos o más VM pueden intentar usar una misma dirección de memoria gráfica, por ejemplo, para un mismo motor de representación o dos diferentes motores

de representación. En algunas realizaciones, el planificador de GPU 1712 puede cambiar la política de planificador para evitar seleccionar uno o más de los motores de representación, que tienen el potencial de entrar en conflicto entre sí. En algunas realizaciones, el planificador de GPU 1712 puede suspender el motor de hardware de ejecución para mitigar el conflicto.

En algunas realizaciones, un esquema de sobreasignación de memoria en virtualización de GPU como se analiza en este documento puede coexistir con esquemas de particionamiento de memoria gráfica global estático. Como un ejemplo, la apertura en el espacio de memoria inferior 1754 puede usarse aún para partición estática entre todas las VM. El espacio de memoria gráfica alta en el espacio de memoria superior 1752 puede usarse para el esquema de sobreasignación de memoria. En comparación con el esquema de particionamiento de espacio de memoria gráfica global estático, el esquema de sobreasignación de memoria en virtualización de GPU puede habilitar que cada VM use todo el espacio de memoria gráfica alta en el espacio de memoria superior 1752, que puede permitir que algunas aplicaciones dentro de cada VM usen un mayor espacio de memoria gráfica para un rendimiento mejorado.

Con esquemas de particionamiento de memoria gráfica global estático, una VM que inicialmente reclama una gran porción de memoria puede usar únicamente una pequeña porción en tiempo de ejecución, mientras otras VM pueden estar en el estado de escasez de memoria. Con sobreasignación de memoria, un hipervisor puede asignar memoria para VM bajo demanda, y la memoria ahorrada puede usarse para soportar más VM. Con sobreasignación de memoria basada en SGGTT, únicamente el espacio de memoria gráfica usado por las cargas de trabajo que hay que ejecutar puede asignarse en tiempo de ejecución, lo que ahorra espacio de memoria gráfica y soporta más VM para acceder a la GPU 1720.

Las arquitecturas actuales habilitan el alojamiento de cargas de trabajo de GPU en entornos en la nube y de centros de datos. La virtualización completa de GPU es una de las tecnologías de habilitación fundamentales usadas en la nube de GPU. En virtualización completa de GPU, el monitor de máquina virtual (VMM), particularmente el controlador de GPU virtual (vGPU), captura y emula los accesos de invitado a recursos de GPU privilegiados para seguridad y multiplexación, mientras pasa a través accesos de CPU a recursos de rendimiento crítico, tales como accesos de CPU a memoria gráfica. Los comandos de GPU, una vez enviados, se ejecutan directamente por la GPU sin intervención de VMM. Como resultado, se consigue un rendimiento cercano a nativo.

Los sistemas actuales usan la memoria de sistema para motores de GPU para acceder a una Tabla de Traducción Gráfica Global (GGTT) y/o una Tabla de Traducción Gráfica Por Proceso (PPGTT) para traducir desde direcciones de memoria gráfica de GPU a direcciones de memoria de sistema. Puede usarse un mecanismo de sombreado para la GGTT/PPGTT de la tabla de paginación de GPU de invitado.

La VMM puede usar una PPGTT de sombra que se sincroniza con la PPGTT de invitado. La PPGTT de invitado está protegida contra escritura de modo que la PPGTT de sombra puede sincronizarse de forma continua con la PPGTT de invitado capturando y emulando las modificaciones de invitado de su PPGTT. En la actualidad, la GGTT para cada vGPU se ensombrece y particiona entre cada VM y la PPGTT se ensombrece y es por VM (por ejemplo, sobre una base por proceso). El sombreado para la tabla de paginación de GGTT es sencillo ya que la tabla de PDE de GGTT permanece en el intervalo de MMIO de bar0 de PCI. Sin embargo, la sombra para la PPGTT depende de la protección contra estructura de la tabla de paginación de PPGTT de invitado y la tabla de paginación de sombra tradicional es muy complicada (y, por lo tanto, con errores) e ineficiente. Por ejemplo, la tabla de paginación de sombra de CPU tiene una tara de rendimiento de ~30 % en arquitecturas actuales. Por lo tanto, en algunos de estos sistemas se usa una tabla de paginación de sombra iluminada, que modifica el controlador de gráficos de invitado para cooperar en la identificación de una página usada para la página de tabla de paginación, y/o cuando se libera.

Las realizaciones de la invención incluyen una unidad de gestión de memoria (MMU) tal como una unidad de gestión de memoria de E/S (IOMMU) para recorrelacionar desde GPN (números de página de invitado) con correlación de PPGTT de invitado con HPN (número de página de anfitrión), sin depender de la PPGTT de sombra de baja eficiencia/complicada. Al mismo tiempo, una realización retiene la tabla de paginación de GGTT de sombra global para el aumento de dirección. Estas técnicas se denominan generalmente capa híbrida de correlación de direcciones (HLAM).

Una IOMMU, por defecto, no puede usarse en ciertas arquitecturas de paso a través mediado, dado que únicamente está disponible una única traducción de segundo nivel con múltiples VM. Una realización de la invención resuelve este problema, utilizando las siguientes técnicas:

1. Usar la IOMMU para llevar a cabo dos capas de traducción sin la PPGTT de sombra. En particular, en una realización, la GPU traduce desde dirección de memoria gráfica (GM_ADDR) a GPN, y la IOMMU traduce desde el GPN a HPN, en lugar de la PPGTT de sombra que traduce desde la GM_ADDR a HPN con protección contra escritura aplicada a la PPGTT de invitado.
2. En una realización, la tabla de paginación de IOMMU se gestiona por VM, y se conmuta (o quizás se conmuta parcialmente) cuando se conmuta la vGPU. Es decir, la tabla de paginación de IOMMU de la correspondiente VM se carga cuando se planifica la VM/vGPU.

3. Sin embargo, las direcciones correlacionadas con GGTT se comparten en una realización, y esta GGTT de sombra global debe permanecer válida porque la vCPU puede acceder a la dirección con correlación de GGTT (por ejemplo, tal como la apertura), incluso cuando la vGPU de esta VM no se planifica. Como tal, una realización de la invención usa una capa híbrida de traducción de direcciones que retiene la GGTT de sombra global, pero usa directamente la PPGTT de invitado.

4. En una realización, el espacio de direcciones de GPN se particiona para desplazar la dirección de GPN con correlación de GGTT (que se convierte a una entrada a la IOMMU, como la GPN) con un intervalo de direcciones especializado. Esto puede conseguirse capturando y emulando la tabla de paginación de GGTT. En una realización, el GPN se modifica desde la GGTT con un gran desplazamiento para evitar el solapamiento con la PPGTT en la correlación de IOMMU.

La Figura 18 ilustra una arquitectura empleada en una realización en la que se habilita una IOMMU 1830 para virtualización de dispositivo. La arquitectura ilustrada incluye dos VM 1801, 1811 ejecutadas en el hipervisor/VMM 1820 (aunque los principios subyacentes de la invención pueden implementarse con cualquier número de VM). Cada VM 1801, 1811 incluye un controlador 1802, 1812 (por ejemplo, un controlador de gráficos nativo) que gestiona una PPGTT de invitado y GGTT 1803, 1813, respectivamente. La IOMMU 1830 ilustrada incluye un módulo de HLAM 1831 para implementar la capa híbrida de técnicas de correlación de direcciones descritas en este documento. En particular, en esta realización, PPGTT de sombra no están presentes.

En una realización, toda la tabla de paginación de traducción de GPN a HPN 1833 de la VM de invitado (la VM de invitado 1811 en el ejemplo) se prepara en la correlación de IOMMU, y cada conmutación de vGPU desencadena un intercambio de tabla de paginación de IOMMU. Es decir, cuando se planifica cada VM 1801, 1811, su correspondiente tabla de traducción de GPN a HPN 1833 se intercambia. En una realización, la HLAM 1831 diferencia entre los GPN de GGTT y los GPN de PPGTT y modifica los GPN de GGTT de modo que no se solapan con los GPN de PPGTT cuando se realiza una consulta en la tabla de traducción 1833. En particular, en una realización, la lógica de generación de GPN virtual 1832 convierte la GGTT GPN a un GPN virtual que se usa, a continuación, para realizar una consulta en la tabla de traducción 1833 para identificar el correspondiente HPN.

En una realización, el GPN virtual se genera desplazando la GGTT por un desplazamiento especificado (potencialmente grande) para garantizar que las direcciones correlacionadas no se solapan/entran en conflicto con el GPN de PPGTT. Además, en una realización, dado que la CPU puede acceder a la dirección con correlación de GGTT (por ejemplo, la apertura) en cualquier momento, la GGTT de sombra global siempre será válida y permanecerá en la correlación de IOMMU 1833 por VM.

En una realización, la solución de correlación de direcciones de capa híbrida 1831 particiona el intervalo de direcciones de IOMMU en dos partes: una parte inferior reservada para traducción de GPN a HPN de PPGT y una parte superior reservada para traducción de GPN a HPN de GGTT virtual. Dado que el GPN se proporciona por la VM/Invitado 1811, el GPN debería estar en el intervalo del tamaño de memoria de invitado. En una realización, las tablas de paginación de PPGTT de invitado no se alteran y todos los GPN de la PPGTT se envían directamente al hardware de traducción gráfica/IOMMU mediante la ejecución de carga de trabajo. Sin embargo, en una realización, se captura la escritura/lectura de MMIO desde las VM de invitado y los cambios de tabla de paginación de GGTT se capturan y alteran como se describe en este documento (por ejemplo, añadiendo un desplazamiento grande al GPN para asegurar ningún solapamiento con la correlación de PPGTT en la IOMMU).

PROCESAMIENTO GRÁFICO VIRTUALIZADO REMOTO

En algunas realizaciones, de la invención, un servidor realiza virtualización gráfica, virtualizando GPU físicas y ejecutando aplicaciones gráficas en nombre de clientes. **La Figura 19** ilustra una de tal realización en la que dos clientes 1901 - 1902 se conectan a servidores 1930 a través de una red 1910 tal como la Internet y/o una red privada. Los servidores 1930 implementan un entorno gráfico virtualizado en el que un hipervisor 1960 asigna recursos desde una o más GPU físicas 1938, presentando los recursos como las GPU virtuales 1934-1935 a las VM/aplicaciones 1932-1933. Los recursos de procesamiento gráfico pueden asignar de acuerdo con políticas de asignación de recursos 1961 que pueden provocar que el hipervisor 1960 asigne recursos basándose en los requisitos de las aplicaciones 1932-1933 (por ejemplo, aplicaciones gráficas de mayor rendimiento que requieren más recursos), la cuenta de usuario asociada con las aplicaciones 1932-1933 (por ejemplo, con ciertos usuarios que pagan una prima para un mayor rendimiento), y/o la carga actual en el sistema. Los recursos de GPU que se asignan pueden incluir, por ejemplo, conjuntos de motores de procesamiento gráfico tales como motores 3D, motores de blit, unidades de ejecución y motores de medios, por nombrar algunos.

En una realización, un usuario de cada cliente 1901 - 1902 tiene una cuenta en el servicio que aloja el servidor o servidores 1930. Por ejemplo, el servicio puede ofrecer un servicio de suscripción para proporcionar a usuarios acceso remoto a aplicaciones en línea 1932-1933 tales como videojuegos, aplicaciones de productividad y aplicaciones de realidad virtual multijugador. En una realización, las aplicaciones se ejecutan remotamente en una máquina virtual en respuesta una entrada de usuario 1907-1908 desde los clientes 1901-1902. Aunque no se ilustra en la **Figura 19**, también pueden virtualizarse y usarse una o más CPU para ejecutar las aplicaciones 1932-1933, con operaciones de procesamiento gráfico descargado a las vGPU 1934-1935.

En una realización, se genera una secuencia de fotogramas de imagen por las vGPU 1934-1935 en respuesta a la ejecución de las operaciones gráficas. Por ejemplo, en un juego de disparos en primera persona, un usuario puede especificar la entrada 1907 para mover un personaje por un mundo de fantasía. En una realización, las imágenes resultantes se comprimen (por ejemplo, mediante circuitería/lógica de compresión, no mostrada) y se transmiten a través de la red 1910 a los clientes 1901-1902. En una implementación, puede usarse un algoritmo de compresión de vídeo tal como H.261; sin embargo, pueden usarse diversas técnicas de compresión diferentes. Los decodificadores 1905-1906 decodifican los flujos de vídeo entrantes, que se representan, a continuación, en respectivos visualizadores 1903-1904 de los clientes 1901-1902.

Usando el sistema ilustrado en la **Figura 19**, recursos de procesamiento gráfico de alto rendimiento tales como las GPU 1938 pueden asignarse a diferentes clientes que se suscriben al servicio. En una implementación de juego en línea, por ejemplo, los servidores 1930 pueden alojar nuevos videojuegos a medida que se lanzan. El código de programa de videojuego se ejecuta, a continuación, en el entorno virtualizado y los fotogramas de vídeo resultantes se comprimen y transmiten a cada cliente 1901-1902. Los clientes 1901-1902 en esta arquitectura no requieren recursos de procesamiento gráfico significativos. Por ejemplo, incluso un teléfono inteligente o tableta con potencia relativamente baja con un decodificador 1905-1906 será capaz de descomprimir un flujo de vídeo. Por lo tanto, los últimos videojuegos con gráficos intensivos pueden jugarse en cualquier tipo de cliente con capacidad de comprimir vídeo. Mientras los videojuegos se describen como una posible implementación, los principios subyacentes de la invención pueden usarse para cualquier forma de aplicación que requiere recursos de procesamiento gráfico (por ejemplo, aplicaciones de diseño gráfico, aplicaciones de trazado de rayos interactivas y no interactivas, software de productividad, software de edición de vídeo, etc.).

SOPORTE DE VISUALIZACIÓN VIRTUALIZADA

En una realización, el motor de visualización tiene diversas canalizaciones que pueden combinarse para representar cada fotograma de imagen. En un entorno virtualizado, por ejemplo, puede haber múltiples VM, por ejemplo, VM0 y VM1. Cada uno de los sistemas operativos (SO) en VM0 y VM1 puede crear múltiples ventanas en memoria. Por ejemplo, dos ventanas en cada una de las VM 0-1 pueden designarse VM0-W1 (ventana 1 en máquina virtual 0), VM0-W2 (ventana 2 en máquina virtual 0), VM1-W1 (ventana 1 en máquina virtual 1) y VM1-W2. Una vez generadas, estas ventanas pueden componerse juntas en un fotograma de imagen. Normalmente las VM están aisladas, por tanto es difícil conseguir que combinen datos para formar un fotograma de imagen integrado. La operación normal es tener un supervisor VM, VM0, para tener acceso a todas las ventanas de la VM1. Para visualizar cualquier cosa, VM1 necesita, a continuación, hacer una petición a VM0. El representador ejecutaría, a continuación, la carga de trabajo como si ejecutase la carga de trabajo de VM0 que incluye todas las memorias intermedias necesarias (que contienen todas las ventanas de VM1).

Este mismo entorno se está trasladando a modelos de uso de automoción en donde existe una pieza de hardware gráfico que ejecuta la agrupación de instrumentos (por ejemplo, el velocímetro) y el visualizador de información y entretenimiento en vehículo (IVI) (por ejemplo, navegación, radio, etc.). Habitualmente no es deseable tener el mismo SO implementando tanto la agrupación de instrumentos como el visualizador de IVI, por tanto las implementaciones actuales usan un SO para hacer la agrupación de instrumentos primaria y, a continuación, una VM separada se ejecuta para el visualizador de IVI. La razón para el aislamiento es porque (1) la agrupación de instrumentos necesita tener alta fiabilidad y operación en tiempo real y (2) las aplicaciones de IVI se actualizarán con mayor frecuencia (por ejemplo, instalando nuevos mapas para la navegación).

En una realización de la invención, el procesamiento de visualización se divide de modo que cada canalización y artículos asociados con la misma se particionan entre las VM usando límites de página de 4 kB. Una porción de la interfaz de visualización se pasa, a continuación, a VM0 (por ejemplo, la agrupación de instrumentos) y la otra porción de la interfaz de visualización se pasa a VM1. En una implementación, las tablas de paginación de CPU se usan para realizar la división para formar dos canalizaciones distintas: siendo una porción de las tablas de paginación asignada a IVI/VM1 y la otra porción a la agrupación de instrumentos/VM0. El resultado final es que tanto VM0 como VM1 ven sus propias porciones asignadas de la memoria de sistema. Configurar el sistema de esta manera permite que las VM0 operen completamente sin restricciones por la VM1 que está controlando su propia memoria intermedia de visualización. Sin embargo, existe un conjunto designado de artículos utilizados por las canalizaciones que aún se controlan por la VM0 tal como la urgencia y/o prioridad asociada con las peticiones de memoria. Estos se establecen globalmente por VM0 que puede designarse la VM confiable.

La Figura 20 ilustra una correlación ilustrativa que incluye una VM0 confiable 2011 usada para la canalización de visualización de agrupación de instrumentos 2031 y VM1 2012 usada para la canalización de visualización de IVI 2032. En el ejemplo ilustrado, se ejecuta un primer sistema operativo 2021 en la VM0 2011 con dos ventanas, W1 y W2. De manera similar, se ejecuta un segundo sistema operativo 2022 en la VM1 con dos ventanas, W1 y W2. Además de las tablas de paginación de CPU, como se ilustra en la **Figura 21**, en un entorno gráfico virtualizado, la tabla de traducción de invitado global (GTT) 2100 puede particionarse entre VM0 y VM1. El espacio de apertura 2101 se particiona también como se ilustra para permitir que tanto VM0 como VM1 se comuniquen con la CPU.

COMPOSICIÓN Y TRANSMISIÓN DE BAJA LATENCIA

Los flujos actuales para visualización remota/inalámbrica se accionan mediante software ejecutado en una CPU. Por ejemplo, la CPU debe individualmente (1) ordenar a la canalización de representación que represente el siguiente fotograma, (2) ordenar al hardware de visualización que visualice resultados en una memoria intermedia de fotogramas (potencialmente componiendo múltiples ventanas) (3) ordenar al codificador que codifique/comprima la memoria intermedia de fotogramas; y (4) ordenar a la interfaz de red que transmita la memoria intermedia de fotogramas a un destino (por ejemplo, un visualizador remoto).

Realizaciones de la invención incluyen circuitería adicional y lógica para reducir la necesidad de intervención de CPU continua durante la composición y transmisión de fotograma de vídeo. En particular, la CPU puede configurar cada uno de los bloques funcionales de extremo a extremo y, a continuación, separar durante un periodo de tiempo. Cada etapa en la canalización de procesamiento gráfico comunica, a continuación, con la etapa anterior y etapa posterior para realizar procesamiento de visualización remota sin intervención de CPU.

Para reducir la interacción de CPU requerida para representar fotogramas para visualización remota/inalámbrica, una realización de la invención proporciona mecanismos para coordinación entre las diversas etapas de procesamiento de visualización remota/inalámbrica. En particular, se realiza pase de mensajes entre etapas y registros actualizados en cada etapa para indicar un puntero de memoria intermedia de cabeza (H) y/o cola (T) actual para una memoria intermedia compartida (por ejemplo, indicando la ubicación de memoria intermedia que cada etapa está procesando en la actualidad).

Como se ilustra en la **Figura 22**, una realización de un sistema de visualización remoto incluye un motor de visualización 2211 incluye circuitería de representación de imagen 2D/3D para representar imágenes 2D o 3D para una o más ventanas (por ejemplo, dentro de un sistema operativo ejecutado en una VM). El motor de visualización 2211 también puede incluir circuitería de composición multiplano para combinar múltiples ventanas representadas en un fotograma de imagen final. Los fotogramas de imagen finales se almacenan dentro de una o más memorias intermedias de fotogramas 2221 que se introducen en un codificador 2212. En una realización, el codificador 2212 implementa una imagen/algoritmo de compresión de vídeo (por ejemplo, H.264 o similar) para comprimir secuencias de fotogramas y generar un flujo de vídeo comprimido, que almacena en una memoria intermedia de flujo comprimido 2221. Finalmente, un controlador de interfaz de red (NIC) 2213 transmite cada fotograma desde la memoria intermedia de flujo comprimido 2232 a un visualizador remoto 2214. En una realización, el NIC 2213 transmite cada fotograma a través de un enlace inalámbrico al visualizador remoto 2214, implementando una tecnología de visualización inalámbrica tal como WiDi.

En una realización, un conjunto de registros 2221-2223 se actualiza inicialmente por la CPU 2204 para almacenar punteros a cada memoria intermedia 2231-2232 así como cualquier otra información necesaria para cada una de las etapas 2211-2213 para comunicarse entre sí tal como identificando una región de memoria a través de la que intercambiar mensajes. Mientras los registros 2221-2223 se ilustran dentro de una etapa de procesamiento particular en la **Figura 22**, puede usarse un archivo de registro de fin general o fin especial que es accesible por todas las etapas de procesamiento 2211-2213.

En una realización, se actualiza un registro de motor de visualización 2221 para identificar el punto en la memoria intermedia de fotogramas 2231 en donde el motor de visualización 2211 comenzará a escribir el primer fotograma y los registros de codificador 2222 se actualizan para incluir un primer puntero a la ubicación en donde el codificador 2212 leerá de la memoria intermedia de fotogramas 2231 y un segundo puntero a la ubicación en donde el codificador 2212 escribirá en la memoria intermedia de flujo comprimido 2232. De manera similar, el registro de NIC 2223 se actualiza con un puntero a la ubicación en la memoria intermedia de flujo comprimido 2232 en donde el NIC 2213 comenzará a leer el siguiente fotograma comprimido.

En la operación, una vez que la CPU escribe los valores iniciales en los registros 2221-2223, ya no necesita interactuar con las diversas etapas de procesamiento de imagen 2211-2213 (como en sistemas con gestión de CPU actuales). En particular, a medida que el motor de visualización 2211 escribe en la memoria intermedia de fotogramas 2231, actualiza la ubicación de escritura actual, en ocasiones denominada como un puntero de cabeza (H), en el registro 2221. Una vez que ha alcanzado un punto especificado escribiendo en la memoria intermedia de fotogramas 2231 (por ejemplo, 16 píxeles o un macrobloque de ancho), notifica al codificador 2212 que comienza a leer de la ubicación almacenada en sus registros 2222, actualizando continuamente la ubicación leída más recientemente. También puede enviar notificaciones al motor de visualización 2211 para indicar su estado (es decir, la ubicación actual desde la que está leyendo).

De manera similar, a medida que el codificador 2212 escribe fotogramas comprimidos en la memoria intermedia de flujo comprimido 2232, puede actualizar su puntero de escritura (o el puntero de "cabeza") dentro de sus registros 2222 y notifica periódicamente al NIC 2213 su progreso (por ejemplo, cuando ha alcanzado un cierto umbral y periódicamente posteriormente). El NIC 2213, a su vez, actualiza su registro 2223 para incluir el valor de puntero de lectura actual a medida que lee de la memoria intermedia de flujo comprimido 2232 y transmite los datos a través de la red al visualizador remoto 2214 (que descomprime y visualiza los datos de imagen).

En una realización, la arquitectura mostrada en la **Figura 22** se usa para implementar un entorno de ejecución virtualizado con una VMM/hipervisor y una o más máquinas virtuales. En esta realización, hilos de cada máquina virtual pueden utilizar el motor de visualización 2211, el codificador 2212 y el NIC 2213 y las técnicas de gestión de memoria intermedia descritas en este documento.

De esta manera, a través de intercomunicación entre las diversas etapas de procesamiento 2211-2213, la CPU se libera de su responsabilidad para coordinar entre las diversas etapas 2211-2213 a medida que realizan sus respectivas operaciones.

En la **Figura 23** se ilustra un método de acuerdo con una realización. El método puede implementarse en las arquitecturas de GPU y procesador descritas anteriormente, pero sin limitación a ninguna arquitectura particular.

En 2301, para iniciar una secuencia de visualización remoto, la CPU escribe punteros de memoria intermedia para el motor de visualización, codificador y controlador de interfaz de red (NIC) y, a continuación, en 2302, se separa una vez que se escriben los punteros de memoria intermedia. Como se usa en este documento, separar significa que la CPU no necesita participar en el fotograma de imagen que representa, codifica y transmite el fotograma de imagen. La separación puede producirse para un único fotograma de imagen o múltiples fotogramas de imagen, dependiendo de la implementación.

En 2303, el motor de visualización comienza representando el fotograma de imagen y escribiendo los resultados en una primera memoria intermedia, actualizando el puntero de escritura de motor de visualización a medida que escribe. En una realización, el puntero de escritura de motor de visualización se almacena en un registro que es accesible tanto por el motor de visualización como el codificador. En 2304, el motor de visualización notifica al codificador cuando el puntero de escritura de motor de visualización ha alcanzado un umbral particular (por ejemplo, cuando se ha almacenado un ancho del fotograma de imagen con capacidad de comprimirse, tal como un macrobloque). Como alternativa, el codificador puede comprobar periódicamente el puntero de escritura para determinar la ubicación de memoria intermedia actual que se escribe. En 2305, el codificador comienza a leer de la primera memoria intermedia como se indica por el puntero de lectura de codificador, codificando los datos de imagen y actualizando el puntero de lectura a medida que avanza. En 2306, el codificador escribe porciones codificadas del fotograma de imagen en una segunda memoria intermedia, como se indica por el puntero de escritura de codificador, actualizando continuamente el puntero de escritura de codificador a medida que los datos se escriben.

En 2307, el codificador notifica al NIC cuando el puntero de escritura de codificador alcanza un umbral tal como un bloque de datos que puede empaquetarse y transmitirse por el NIC. Como alternativa, el NIC puede comprobar periódicamente el puntero de escritura del codificador para determinar cuándo comenzar a escribir. En 2308, el NIC comienza a leer los datos de imagen codificados de la segunda memoria intermedia como se indica por el puntero de lectura de NIC, actualizando el puntero de lectura a medida que los datos codificados se transmiten a través de una conexión de red al visualizador remoto, y actualizando el puntero de lectura a medida que se transmiten los datos.

El motor de visualización, codificador y NIC continúan procesando el fotograma de imagen, actualizando punteros de lectura/escritura y proporcionando notificaciones entre sí en 2309 hasta que el fotograma de imagen se ha representado, codificado y transmitido. El proceso se mueve al siguiente fotograma de imagen en 2310. Dependiendo de la implementación, el proceso puede volver, a continuación, a 2301, en donde la CPU inicia el proceso escribiendo punteros de memoria intermedia para el siguiente fotograma. Como alternativa, el proceso puede volver a 2303 y moverse en al siguiente fotograma de imagen sin intervención de CPU.

En algunas realizaciones, una unidad de procesamiento gráfico (GPU) se acopla comunicativamente a núcleos de anfitrión/procesador para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrón y diversas funciones de GPU de fin general (GPGPU). La GPU puede acoplarse comunicativamente al procesador de anfitrión/núcleos a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otras realizaciones, la GPU puede integrarse en el mismo paquete o chip que los núcleos y acoplarse comunicativamente a los núcleos a través de un bus/interconexión de procesador interno (es decir, interno al paquete o chip). Independientemente de la manera en la que se conecta la GPU, los núcleos de procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU usa, a continuación, circuitería/lógica especializada para procesar de forma eficiente estos comandos/instrucciones.

En la siguiente descripción, se exponen numerosos detalles específicos para proporcionar un entendimiento más completo. Sin embargo, será evidente para un experto en la materia que las realizaciones descritas en este documento pueden ponerse en práctica sin uno o más de estos detalles específicos. En otros casos, no se han descrito características bien conocidas para evitar obstaculizar los detalles de las presentes realizaciones.

Visión general del sistema

La Figura 24 es un diagrama de bloques que ilustra un sistema informático 2400 configurado para implementar uno o más aspectos de las realizaciones descritas en este documento. El sistema informático 2400 incluye un subsistema de procesamiento 2401 que tiene uno o más procesador o procesadores 2402 y una memoria de sistema 2404 que se comunica a través de una trayectoria de interconexión que puede incluir un concentrador de memoria 2405. El concentrador de memoria 2405 puede ser un componente separado dentro de un componente de conjunto de chips o puede integrarse dentro del uno o más procesador o procesadores 2402. El concentrador de memoria 2405 se acopla con un subsistema de E/S 2411 a través de un enlace de comunicación 2406. El subsistema de E/S 2411 incluye un concentrador de E/S 2407 que puede habilitar que el sistema informático 2400 reciba una entrada de uno o más dispositivo o dispositivos de entrada 2408. Adicionalmente, el concentrador de E/S 2407 puede habilitar que un controlador de visualización, que puede incluirse en el uno o más procesador o procesadores 2402, proporcione salidas a uno o más dispositivo o dispositivos de visualización 2410A. En una realización, el uno o más dispositivo o dispositivos de visualización 2410A acoplados con el concentrador de E/S 2407 pueden incluir un dispositivo de visualización local, interno o embebido.

En una realización, el subsistema de procesamiento 2401 incluye uno o más procesador o procesadores paralelos 2412 acoplados al concentrador de memoria 2405 a través de un bus u otro enlace de comunicación 2413. El enlace de comunicación 2413 puede ser una de cualquier número de tecnologías o protocolos de comunicación basados en normas, tal como, pero sin limitación a PCI Express, o puede ser una interfaz de comunicaciones o tejido de comunicaciones específico de un proveedor. En una realización, el uno o más procesador o procesadores paralelos 2412 forman un sistema de procesamiento paralelo o vectorial computacionalmente centrado que incluyen un gran número de núcleos de procesamiento y/o agrupaciones de procesamiento, tales como muchos procesador de núcleo integrado (MIC). En una realización, el uno o más procesador o procesadores paralelos 2412 forman un subsistema de procesamiento gráfico que puede emitir píxeles a uno del uno o más dispositivo o dispositivos de visualización 2410A acoplados a través del concentrador de E/S 2407. El uno o más procesador o procesadores paralelos 2412 también pueden incluir un controlador de visualización e interfaz de visualización (no mostrados) para habilitar una conexión directa a uno o más dispositivo o dispositivos de visualización 2410B.

Dentro del subsistema de E/S 2411, un sistema unidad de almacenamiento 2414 puede conectarse al concentrador de E/S 2407 para proporcionar un mecanismo de almacenamiento para el sistema informático 2400. Un conmutador de E/S 2416 puede usarse para proporcionar un mecanismo de interfaz para habilitar conexiones entre el concentrador de E/S 2407 y otros componentes, tal como a adaptador de red 2418 y/o adaptador de red inalámbrica 2419 que puede integrarse en la plataforma, y diversos otros dispositivos que pueden añadirse a través de uno o más dispositivo o dispositivos de complemento 2420. El adaptador de red 2418 puede ser un adaptador de Ethernet u otro adaptador de red por cable. El adaptador de red inalámbrica 2419 puede incluir uno o más de Wi-Fi, Bluetooth, comunicación de campo cercano (NFC) u otro dispositivo de red que incluye una o más radios inalámbricas.

El sistema informático 2400 puede incluir otros componentes no mostrados explícitamente, incluyendo USB u otras conexiones de puerto, unidades de almacenamiento óptico, dispositivos de captura de vídeo y similares, también puede conectarse al concentrador de E/S 2407. Las trayectorias de comunicación que interconectan los diversos componentes en la Figura 24 pueden implementarse usando cualquier protocolo adecuado, tales como protocolos basados en PCI (Interconexión de Componentes Periféricos) (por ejemplo, PCI-Express), o cualquier otro bus o interfaz de comunicación de punto a punto y/o protocolo o protocolos, tales como la interconexión de alta velocidad NV-Link, o protocolos de interconexión conocidos en la técnica.

En una realización, el uno o más procesador o procesadores paralelos 2412 incorporan circuitería optimizada para procesamiento de gráficos y vídeo, incluyendo, por ejemplo, circuitería de salida de vídeo, y constituyen una unidad de procesamiento gráfico (GPU). En otra realización, el uno o más procesador o procesadores paralelos 2412 incorporan circuitería optimizada para procesamiento de fin general, mientras preserva la arquitectura de cálculo subyacente, descrita en mayor detalle en este documento. En otra realización más, componentes del sistema informático 2400 pueden integrarse con uno o más otros elementos de sistema en un único circuito integrado. Por ejemplo, el uno o más procesador o procesadores paralelos 2412, concentrador de memoria 2405, procesador o procesadores 2402 y concentrador de E/S 2407 pueden integrarse en un circuito integrado de sistema en chip (SoC). Como alternativa, los componentes del sistema informático 2400 pueden integrarse en un único paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una porción de los componentes del sistema informático 2400 puede integrarse en un módulo multichip (MCM), que puede interconectarse con otros módulos multichip en un sistema informático modular.

Se apreciará que el sistema informático 2400 mostrado en este documento es ilustrativo y que son posibles variaciones y modificaciones. La topología de conexión, incluyendo el número y disposición de puentes, el número de procesador o procesadores 2402 y el número de procesador o procesadores paralelos 2412 pueden modificarse según se desee. Por ejemplo, en algunas realizaciones, la memoria de sistema 2404 se conecta al procesador o procesadores 2402 directamente en lugar de a través de un puente, mientras otros dispositivos se comunican con la memoria de sistema 2404 a través del concentrador de memoria 2405 y el procesador o procesadores 2402. En otras topologías alternativas, el procesador o procesadores paralelos 2412 se conectan al concentrador de E/S 2407 o directamente a uno del uno o más procesador o procesadores 2402, en lugar de al concentrador de memoria 2405. En otras realizaciones, el concentrador de E/S 2407 y el concentrador de memoria 2405 pueden integrarse en un único chip.

Algunas realizaciones pueden incluir dos o más conjuntos de procesador o procesadores 2402 conectados a través de múltiples zócalos, que pueden acoplarse con dos o más instancias del procesador o procesadores paralelos 2412.

Algunos de los componentes particulares mostrados en este documento son opcionales y pueden no incluirse en todas las implementaciones del sistema informático 2400. Por ejemplo, puede soportarse cualquier número de tarjetas o periféricos de complemento, o algunos componentes pueden eliminarse. Adicionalmente, algunas arquitecturas pueden usar diferente terminología para componentes similares a los ilustrados en la Figura 24. Por ejemplo, el concentrador de memoria 2405 puede denominarse como un puente norte en algunas arquitecturas, mientras el concentrador de E/S 2407 puede denominarse como un puente sur.

La Figura 25A ilustra un procesador paralelo 2500, de acuerdo con una realización. Los diversos componentes del procesador paralelo 2500 pueden implementarse usando uno o más dispositivos de circuito integrado, tales como procesadores programables, circuitos integrados específicos de la aplicación (ASIC) o matrices de puertas programables en campo (FPGA). El procesador paralelo 2500 ilustrado es una variante del uno o más procesador o procesadores paralelos 2412 mostrados en la Figura 24, de acuerdo con una realización.

En una realización, el procesador paralelo 2500 incluye una unidad de procesamiento paralelo 2502. La unidad de procesamiento paralelo incluye una unidad de E/S 2504 que habilita comunicación con otros dispositivos, incluyendo otras instancias de la unidad de procesamiento paralelo 2502. La unidad de E/S 2504 puede conectarse directamente a otros dispositivos. En una realización, la unidad de E/S 2504 se conecta con otros dispositivos a través del uso de un concentrador o interfaz de conmutación, tal como el concentrador de memoria 25405. Las conexiones entre el concentrador de memoria 25405 y la unidad de E/S 2504 forman un enlace de comunicación 25413. Dentro de la unidad de procesamiento paralelo 2502, la unidad de E/S 2504 se conecta con una interfaz de anfitrión 2506 y una barra cruzada de memoria 2516, en donde la interfaz de anfitrión 2506 recibe comandos dirigidos a realizar operaciones de procesamiento y la barra cruzada de memoria 2516 recibe comandos dirigidos a realizar operaciones de memoria.

Cuando la interfaz de anfitrión 2506 recibe una memoria intermedia de comandos a través de la unidad de E/S 2504, la interfaz de anfitrión 2506 puede dirigir operaciones de trabajo para realizar esos comandos a un extremo frontal 2508. En una realización, el extremo frontal 2508 se acopla con un planificador 2510, que está configurado para distribuir comandos u otros artículos de trabajo a una matriz de agrupación de procesamiento 2512. En una realización, el planificador 2510 garantiza que la matriz de agrupación de procesamiento 2512 se configura correctamente y en un estado válido antes de que las tareas se distribuyan a las agrupaciones de procesamiento de la matriz de agrupación de procesamiento 2512. En una realización, el planificador 2510 se implementa a través de lógica de firmware que se ejecuta en un microcontrolador. El planificador implementado en microcontrolador 2510 es configurable para realizar operaciones de planificación compleja y distribución de trabajo en granularidad gruesa y fina, habilitando una rápida conmutación de prioridad y contexto de hilos que se ejecutan en la matriz de procesamiento 2512. En una realización, el software de anfitrión puede verificar cargas de trabajo para planificación en la matriz de procesamiento 2512 a través de una de múltiples timbres de procesamiento gráfico. Las cargas de trabajo pueden distribuirse, a continuación, automáticamente a través de la matriz de procesamiento 2512 por el planificador 2510 lógica dentro del microcontrolador de planificador.

La matriz de agrupación de procesamiento 2512 puede incluir hasta "N" agrupaciones de procesamiento (por ejemplo, agrupación 2514A, agrupación 2514B a agrupación 2514N). Cada agrupación 2514A-2514N de la matriz de agrupación de procesamiento 2512 puede ejecutar un gran número de hilos concurrentes. El planificador 2510 puede asignar trabajo a las agrupaciones 2514A-2514N de la matriz de agrupación de procesamiento 2512 usando diversos algoritmos de planificación y/o distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surge para cada tipo de programa o cálculo. La planificación puede tratarse dinámicamente por el planificador 2510, o puede ser asistida en parte mediante lógica de compilación durante la compilación de lógica de programa configurada para su ejecución por la matriz de agrupación de procesamiento 2512. En una realización, pueden asignarse diferentes agrupaciones 2514A-2514N de la matriz de agrupación de procesamiento 2512 para procesar diferentes tipos de programas o para realizar diferentes tipos de cálculos.

La matriz de agrupación de procesamiento 2512 puede configurarse para realizar diversos tipos de operaciones de procesamiento paralelo. En una realización, la matriz de agrupación de procesamiento 2512 está configurada para realizar operaciones de cálculo en paralelo de fin general. Por ejemplo, la matriz de agrupación de procesamiento 2512 puede incluir lógica para ejecutar tareas de procesamiento que incluyen filtrado de vídeo y/o datos de audio, realización de operaciones de modelado, incluyendo operaciones físicas, y realización de transformaciones de datos.

En una realización, la matriz de agrupación de procesamiento 2512 está configurada para realizar operaciones de procesamiento gráfico paralelo. En realizaciones en las que el procesador paralelo 2500 está configurado para realizar operaciones de procesamiento gráfico, la matriz de agrupación de procesamiento 2512 puede incluir lógica adicional para soportar la ejecución de tales operaciones de procesamiento gráfico, incluyendo, pero sin limitación a lógica de muestreo de textura para realizar operaciones de textura, así como lógica de teselación y otra lógica de procesamiento de vértices. Adicionalmente, la matriz de agrupación de procesamiento 2512 puede configurarse para ejecutar programas de sombreador relacionados con procesamiento gráfico tales como, pero sin limitación a, sombreadores

de vértices, sombreadores de teselación, sombreadores de geometría y sombreadores de píxeles. La unidad de procesamiento paralelo 2502 puede transferir datos desde la memoria de sistema a través de la unidad de E/S 2504 para su procesamiento. Durante el procesamiento, los datos transferidos pueden almacenarse en memoria en chip (por ejemplo, memoria de procesador paralelo 2522) durante el procesamiento, a continuación, escribirse de vuelta en la memoria de sistema.

En una realización, cuando la unidad de procesamiento paralelo 2502 se usa para realizar procesamiento gráfico, el planificador 2510 puede configurarse para dividir la carga de trabajo de procesamiento en tareas de tamaño aproximadamente igual, para habilitar una mejor distribución de las operaciones de procesamiento gráfico a múltiples agrupaciones 2514A-2514N de la matriz de agrupación de procesamiento 2512. En algunas realizaciones, porciones de la matriz de agrupación de procesamiento 2512 pueden configurarse para realizar diferentes tipos de procesamiento. Por ejemplo, una primera porción puede configurarse para realizar sombreado de vértices y generación de tipología, una segunda porción puede configurarse para realizar teselación y sombreado de geometría, y una tercera porción puede configurarse para realizar sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen representada para su visualización. Los datos intermedios producidos por una o más de las agrupaciones 2514A-2514N pueden almacenarse en memorias intermedias para permitir que los datos intermedios se transmitan entre las agrupaciones 2514A-2514N para su procesamiento adicional.

Durante la operación, la matriz de agrupación de procesamiento 2512 puede recibir tareas de procesamiento que hay que ejecutar a través del planificador 2510, que recibe comandos que definen tareas de procesamiento desde el extremo frontal 2508. Para operaciones de procesamiento gráfico, las tareas de procesamiento pueden incluir índices de datos que hay que procesar, por ejemplo, datos de superficie (parche), datos de primitiva, datos de vértice y/o datos píxel, así como parámetros de estado y comandos que definen cómo tienen que procesarse los datos (por ejemplo, qué programa tiene que ejecutarse). El planificador 2510 puede configurarse para buscar los índices que corresponden a las tareas o puede recibir los índices desde el extremo frontal 2508. El extremo frontal 2508 puede configurarse para garantizar que la matriz de agrupación de procesamiento 2512 está configurada a un estado válido antes de que se inicie la carga de trabajo especificada por memorias intermedias de comandos entrantes (por ejemplo, memorias intermedias por lotes, memorias intermedias de inserción, etc.).

Cada una de las una o más instancias de la unidad de procesamiento paralelo 2502 puede acoplarse con la memoria de procesador paralelo 2522. La memoria de procesador paralelo 2522 puede accederse a través de la barra cruzada de memoria 2516, que puede recibir peticiones de memoria desde la matriz de agrupación de procesamiento 2512 así como la unidad de E/S 2504. La barra cruzada de memoria 2516 puede acceder a la memoria de procesador paralelo 2522 a través de una interfaz de memoria 2518. La interfaz de memoria 2518 puede incluir múltiples unidades de partición (por ejemplo, unidad de partición 2520A, unidad de partición 2520B a unidad de partición 2520N) cada una de las cuales puede acoplarse a una porción (por ejemplo, unidad de memoria) de memoria de procesador paralelo 2522. En una implementación el número de unidades de partición 2520A-2520N está configurado para ser igual al número de unidades de memoria, de tal forma que una primera unidad de división 2520A tiene una correspondiente primera unidad de memoria 2524A, una segunda unidad de división 2520B tiene una correspondiente unidad de memoria 2524B, y una N^{ésima} unidad de partición 2520N tiene una correspondiente N^{ésima} unidad de memoria 2524N. En otras realizaciones, el número de unidades de partición 2520A-2520N puede no ser igual al número de dispositivos de memoria.

En diversas realizaciones, las unidades de memoria 2524A-2524N pueden incluir diversos tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámica (DRAM) o memoria gráfica de acceso aleatorio, tal como memoria gráfica de acceso aleatorio síncrona (SGRAM), incluyendo memoria de tasa de datos doble gráfica (GDDR). En una realización, las unidades de memoria 2524A-2524N también pueden incluir memoria apilada 3D, incluyendo pero sin limitación a memoria de ancho de banda alto (HBM). Expertos en la materia apreciarán que la implementación específica de las unidades de memoria 2524A-2524N puede variar, y puede seleccionarse de uno de diversos diseños convencionales. Objetivos de representación, tales como memorias intermedias de fotogramas o mapas de textura, pueden almacenarse a través de las unidades de memoria 2524A-2524N, permitiendo que las unidades de partición 2520A-2520N escriban porciones de cada objetivo de representación en paralelo a uso de forma eficiente del ancho de banda disponible de memoria de procesador paralelo 2522. En algunas realizaciones, una instancia local de la memoria de procesador paralelo 2522 puede excluirse en favor de un diseño de memoria unificada que utiliza memoria de sistema en conjunto con memoria caché local.

En una realización, una cualquiera de las agrupaciones 2514A-2514N de la matriz de agrupación de procesamiento 2512 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 2524A-2524N dentro de la memoria de procesador paralelo 2522. La barra cruzada de memoria 2516 puede configurarse para transferir la salida de cada agrupación 2514A-2514N a cualquier unidad de partición 2520A-2520N o a otra agrupación 2514A-2514N, que puede realizar operaciones de procesamiento adicionales en la salida. Cada agrupación 2514A-2514N puede comunicarse con la interfaz de memoria 2518 a través de la barra cruzada de memoria 2516 para leer de o escribir en diversos dispositivos de memoria externa. En una realización, la barra cruzada de memoria 2516 tiene una conexión a la interfaz de memoria 2518 para comunicarse con la unidad de E/S 2504, así como una conexión a una instancia local de la memoria de procesador paralelo 2522, habilitando que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 2514A-2514N se comuniquen con memoria de sistema u otra memoria que

no es local a la unidad de procesamiento paralelo 2502. En una realización, la barra cruzada de memoria 2516 puede usar canales virtuales para separar flujos de tráfico entre las agrupaciones 2514A-2514N y las unidades de partición 2520A-2520N.

Mientras una única instancia de la unidad de procesamiento paralelo 2502 se ilustra dentro del procesador paralelo 2500, puede incluirse cualquier número de instancias de la unidad de procesamiento paralelo 2502. Por ejemplo, múltiples instancias de la unidad de procesamiento paralelo 2502 pueden proporcionarse en una única tarjeta de complemento, o pueden interconectarse múltiples tarjetas de complemento. Las diferentes instancias de la unidad de procesamiento paralelo 2502 pueden configurarse para interoperar incluso si las diferentes instancias tienen diferentes números de núcleos de procesamiento, diferentes cantidades de memoria local de procesador paralelo, y/u otras diferencias de configuración. Por ejemplo y en una realización, algunas instancias de la unidad de procesamiento paralelo 2502 pueden incluir unidades de punto flotante de mayor precisión en relación con otras instancias. Sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 2502 o el procesador paralelo 2500 pueden implementarse en una diversidad de configuraciones y formar factores, incluyendo pero sin limitación ordenadores personales de escritorio, portátiles o de mano, servidores, estaciones de trabajo, consolas de juegos y/o sistemas embebidos.

La Figura 25B es un diagrama de bloques de una unidad de partición 2520, de acuerdo con una realización. En una realización, la unidad de partición 2520 es una instancia de una de las unidades de partición 2520A-2520N de **la Figura 25A**. Como se ilustra, la unidad de partición 2520 incluye una memoria caché L2 2521, una interfaz de memoria intermedia de fotogramas 2525 y una ROP (unidad de operaciones de trama) 2526. La memoria caché L2 2521 es una memoria caché de lectura/escritura que está configurada para realizar operaciones de carga y almacenamiento recibidas desde la barra cruzada de memoria 2516 y la ROP 2526. Fallos de lecturas y peticiones de reescritura urgentes se emiten por la memoria caché L2 2521 a la interfaz de memoria intermedia de fotogramas 2525 para su procesamiento. También pueden enviarse actualizaciones a la memoria intermedia de fotogramas a través de la interfaz de memoria intermedia de fotogramas 2525 para su procesamiento. En una realización, la interfaz de memoria intermedia de fotogramas 2525 interactúa con una de las unidades de memoria en memoria de procesador paralelo, tales como las unidades de memoria 2524A-2524N de la Figura 25 (por ejemplo, dentro de la memoria de procesador paralelo 2522).

En aplicaciones gráficas, la ROP 2526 es una unidad de procesamiento que realiza operaciones de trama tales como patrón, prueba z, combinación y similares. La ROP 2526 emite, a continuación, datos gráficos procesados que se almacenan en memoria gráfica. En algunas realizaciones, la ROP 2526 incluye lógica de compresión para comprimir datos de profundidad o color que se escriben en memoria y descomprime datos de profundidad o color que se leen de memoria. La lógica de compresión puede ser lógica de compresión sin pérdida que hace uso de uno o más de múltiples algoritmos de compresión. El tipo de compresión que se realiza por la ROP 2526 puede variar basándose en las características estadísticas de los datos que hay que comprimir. Por ejemplo, en una realización, se realiza compresión de color delta en datos de profundidad y color sobre una base de por losa.

En algunas realizaciones, la ROP 2526 se incluye dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 2514A-2514N de **la Figura 25**) en lugar de dentro de la unidad de partición 2520. En tal realización, las peticiones de lectura y escritura para datos de píxel se transmiten a través de la barra cruzada de memoria 2516 en lugar de datos de fragmentos de píxel. Los datos gráficos procesados pueden visualizarse en un dispositivo de visualización, tal como uno del uno o más dispositivo o dispositivos de visualización 2410 de **la Figura 24**, encaminarse para su procesamiento adicional por el procesador o procesadores 2402, o encaminarse para su procesamiento adicional por una de las entidades de procesamiento dentro del procesador paralelo 2500 de la Figura 25A.

La Figura 25C es un diagrama de bloques de una agrupación de procesamiento 2514 dentro de una unidad de procesamiento paralelo, de acuerdo con una realización. En una realización, la agrupación de procesamiento es una instancia de una de las agrupaciones de procesamiento 2514A-2514N de la Figura 25. La agrupación de procesamiento 2514 puede configurarse para ejecutar muchos hilos en paralelo, en donde el término "hilo" se refiere a una instancia de un programa particular que se ejecuta en un conjunto particular de datos de entrada. En algunas realizaciones, se usan técnicas de emisión de instrucciones de única instrucción, múltiples datos (SIMD) para soportar la ejecución paralela de un gran número de hilos sin proporcionar múltiples unidades de instrucción independientes. En otras realizaciones, se usan técnicas de única instrucción, múltiples hilos (SIMT) para soportar la ejecución paralela de un gran número de hilos generalmente sincronizados, usando una unidad de instrucción común configurada para emitir instrucciones a un conjunto de motores de procesamiento dentro de cada una de las agrupaciones de procesamiento. A diferencia de un régimen de ejecución de SIMD, en donde todos los motores de procesamiento ejecutan habitualmente instrucciones idénticas, la ejecución de SIMT permite que diferentes hilos sigan más fácilmente trayectorias de ejecución divergentes a través de un programa de hilos dado. Expertos en la materia entenderán que un régimen de procesamiento de SIMD representa un subconjunto funcional de un régimen de procesamiento de SIMT.

La operación de la agrupación de procesamiento 2514 puede controlarse a través de un gestor de canalización 2532 que distribuye tareas de procesamiento a procesadores paralelos de SIMT. El gestor de canalización 2532 recibe instrucciones desde el planificador 2510 de la Figura 25 y gestiona la ejecución de esas instrucciones a través de un

multiprocesador gráfico 2534 y/o una unidad de textura 2536. El multiprocesador gráfico 2534 ilustrado es una instancia ilustrativa de un procesador paralelo de SIMT. Sin embargo, diversos tipos de procesadores paralelos de SIMT de diferentes arquitecturas pueden incluirse dentro de la agrupación de procesamiento 2514. Una o más instancias del multiprocesador gráfico 2534 pueden incluirse dentro de una agrupación de procesamiento 2514. El multiprocesador gráfico 2534 puede procesar datos y puede usarse una barra cruzada de datos 2540 para distribuir los datos procesados a uno de múltiples posibles destinos, incluyendo otras unidades de sombreador. El gestor de canalización 2532 puede facilitar la distribución de datos procesados especificando destinos para datos procesados que hay que distribuir a través de la barra cruzada de datos 2540.

Cada multiprocesador gráfico 2534 dentro de la agrupación de procesamiento 2514 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades de lógica aritmética, unidades de carga-almacenamiento, etc.). La lógica de ejecución funcional puede configurarse en una manera de canalización en la que pueden emitirse nuevas instrucciones antes de que se completen las instrucciones anteriores. La lógica de ejecución funcional soporta una diversidad de operaciones que incluyen aritmética de enteros y de punto flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y cálculo de diversas funciones de álgebra. En una realización, puede aprovecharse el mismo hardware de unidad funcional para realizar diferentes operaciones y puede estar presente cualquier combinación de unidades funcionales.

Las instrucciones transmitidas a la agrupación de procesamiento 2514 constituyen un hilo. Un conjunto de hilos que se ejecuta a través del conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa en diferentes datos de entrada. Cada hilo dentro de un grupo de hilos puede asignarse a un motor de procesamiento diferente dentro de un multiprocesador gráfico 2534. Un grupo de hilos puede incluir menos hilos que el número de motores de procesamiento dentro del multiprocesador gráfico 2534. Cuando un grupo de hilos incluye menos hilos que el número de motores de procesamiento, uno o más de los motores de procesamiento puede estar en reposo durante ciclos en los que ese grupo de hilos se está procesando. Un grupo de hilos también puede incluir más hilos que el número de motores de procesamiento dentro del multiprocesador gráfico 2534. Cuando el grupo de hilos incluye más hilos que el número de motores de procesamiento dentro del multiprocesador gráfico 2534, el procesamiento puede realizarse en ciclos de reloj consecutivos. En una realización, múltiples grupos de hilos pueden ejecutarse simultáneamente en un multiprocesador gráfico 2534.

En una realización, el multiprocesador gráfico 2534 incluye una memoria caché interna para realizar operaciones de carga y almacenamiento. En una realización, el multiprocesador gráfico 2534 puede prescindir de una memoria caché interna y usar una memoria caché (por ejemplo, memoria caché L1 308) dentro de la agrupación de procesamiento 2514. Cada multiprocesador gráfico 2534 también tiene acceso a memorias caché L2 dentro de las unidades de partición (por ejemplo, las unidades de partición 2520A-2520N de **la Figura 25**) que se comparten entre todas las agrupaciones de procesamiento 2514 y pueden usarse para transferir datos entre hilos. El multiprocesador gráfico 2534 también puede acceder a memoria global fuera de chip, que puede incluir una o más de memoria local de procesador paralelo y/o memoria de sistema. Cualquier memoria externa a la unidad de procesamiento paralelo 2502 puede usarse como memoria global. Realizaciones en las que la agrupación de procesamiento 2514 incluye múltiples instancias del multiprocesador gráfico 2534 pueden compartir instrucciones y datos comunes, que pueden almacenarse en la memoria caché L1 308.

Cada agrupación de procesamiento 2514 puede incluir una MMU (unidad de gestión de memoria) 2545 que está configurada para correlacionar direcciones virtuales con direcciones físicas. En otras realizaciones, una o más instancias de la MMU 2545 pueden residir dentro de la interfaz de memoria 2518 de la Figura 25. La MMU 2545 incluye un conjunto de entradas de tabla de paginación (PTE) usadas para correlacionar una dirección virtual con una dirección física de una losa (háblese más de mosaico) y opcionalmente un índice de línea de memoria caché. La MMU 2545 puede incluir memorias intermedias de traducción anticipada (TLB) de direcciones o memorias caché que pueden residir dentro del multiprocesador gráfico 2534 o la memoria caché L1 o la agrupación de procesamiento 2514. La dirección física se procesa para distribuir la localización de acceso a datos de superficie para permitir una petición eficiente que se intercala entre unidades de partición. El índice de línea de memoria caché puede usarse para determinar si una petición para una línea de caché es un acierto o fallo.

En aplicaciones gráficas e informáticas, una agrupación de procesamiento 2514 puede configurarse de tal forma que cada multiprocesador gráfico 2534 se acopla a una unidad de textura 2536 para realizar operaciones de correlación de texturas, por ejemplo, determinando posiciones de muestra de texturas, leyendo datos de textura, y filtrando los datos de textura. Los datos de textura se leen de una memoria caché interna de textura L1 (no mostrada) o en algunas realizaciones de la memoria caché L1 dentro del multiprocesador gráfico 2534 y se busca en una memoria caché L2, memoria local de procesador paralelo, o memoria de sistema, según se necesite. Cada multiprocesador gráfico 2534 emite tareas procesadas a la barra cruzada de datos 2540 para proporcionar la tarea procesada a otra agrupación de procesamiento 2514 para su procesamiento adicional o para almacenar la tarea procesada en una memoria caché L2, memoria local de procesador paralelo o memoria de sistema a través de la barra cruzada de memoria 2516. Una preROP 2542 (unidad de operaciones de pretrama) está configurada para recibir datos desde el multiprocesador gráfico 2534, dirigir datos a unidades de ROP, que pueden ubicarse con unidades de partición como se describe en este documento (por ejemplo, las unidades de partición 2520A-2520N de **la Figura 25**). La unidad de preROP 2542

puede realizar optimizaciones para combinación de colores, organizar datos de color de píxeles y realizar traducciones de direcciones.

Se apreciará que el núcleo arquitectura descrito en este documento es ilustrativo y que son posibles variaciones y modificaciones. Cualquier número de unidades de procesamiento, por ejemplo, el multiprocesador gráfico 2534, la unidad de texturas 2536, las preROP 2542, etc., pueden incluirse dentro de una agrupación de procesamiento 2514. Además, mientras se muestra únicamente una agrupación de procesamiento 2514, una unidad de procesamiento paralelo como se describe en este documento puede incluir cualquier número de instancias de la agrupación de procesamiento 2514. En una realización, cada agrupación de procesamiento 2514 puede configurarse para operar independientemente de otras agrupaciones de procesamiento 2514 usando unidades de procesamiento, memorias caché L1, etc. separadas y distintas.

La Figura 25D muestra un multiprocesador gráfico 2534, de acuerdo con una realización. En tal realización, el multiprocesador gráfico 2534 se acopla con el gestor de canalización 2532 de la agrupación de procesamiento 2514. El multiprocesador gráfico 2534 tiene una canalización de ejecución que incluye, pero sin limitación a, una memoria caché de instrucciones 2552, una unidad de instrucción 2554, una unidad de correlación de direcciones 2556, un archivo de registro 2558, uno o más núcleos de unidad de procesamiento gráfico de fin general (GPGPU) 2562 y una o más unidades de carga/almacenamiento 2566. Los núcleos de GPGPU 2562 y las unidades de carga/almacenamiento 2566 se acoplan con la memoria caché 2572 y la memoria compartida 2570 a través de una interconexión de memoria y memoria caché 2568.

En una realización, la memoria caché de instrucciones 2552 recibe un flujo de instrucciones para ejecutar desde el gestor de canalización 2532. Las instrucciones se almacenan en memoria caché en la memoria caché de instrucciones 2552 y se distribuyen para su ejecución por la unidad de instrucción 2554. La unidad de instrucción 2554 puede distribuir instrucciones como grupos de hilos (por ejemplo, saltos), con cada hilo del grupo de hilos asignado a una diferente unidad de ejecución dentro del núcleo de GPGPU 2562. Una instrucción puede acceder a cualquiera de un espacio de direcciones locales, compartidas o globales especificando una dirección dentro de un espacio de direcciones unificadas. La unidad de correlación de direcciones 2556 puede usarse para traducir direcciones en el espacio de direcciones unificadas a una dirección de memoria distinta que puede accederse por las unidades de carga/almacenamiento 2566.

El archivo de registro 2558 proporciona un conjunto de registros para las unidades funcionales del multiprocesador gráfico 324. El archivo de registro 2558 proporciona almacenamiento temporal para operandos conectados a las trayectorias de datos de las unidades funcionales (por ejemplo, núcleos de GPGPU 2562, unidades de carga/almacenamiento 2566) del multiprocesador gráfico 324. En una realización, el archivo de registro 2558 se divide entre cada una de las unidades funcionales de tal forma que a cada unidad funcional se asigna una porción especializada del archivo de registro 2558. En una realización, el archivo de registro 2558 se divide entre los diferentes saltos que se ejecutan por el multiprocesador gráfico 324.

Cada uno de los núcleos de GPGPU 2562 puede incluir unidades de punto flotante (FPU) y/o unidades de lógica de aritmética de enteros (ALU) que se usan para ejecutar instrucciones del multiprocesador gráfico 324. Los núcleos de GPGPU 2562 pueden ser similares en arquitectura o pueden diferir en arquitectura, de acuerdo con realizaciones. Por ejemplo y en una realización, una primera porción de los núcleos de GPGPU 2562 incluyen una FPU de una precisión simple y una ALU de enteros mientras una segunda porción de los núcleos de GPGPU incluyen una FPU de precisión doble. En una realización, las FPU pueden implementar la norma de IEEE 754-2008 para aritmética de punto flotante o habilitar aritmética de punto flotante de precisión variable. El multiprocesador gráfico 324 puede incluir adicionalmente una o más unidades de función fija o función especial para realizar funciones específicas tales como operaciones de copiar rectángulos o combinación de píxeles. En una realización, uno o más de los núcleos de GPGPU también pueden incluir lógica de función fija o especial.

En una realización, los núcleos de GPGPU 2562 incluyen SIMD lógica con capacidad de realizar una única instrucción en múltiples conjuntos de datos. En una realización, los núcleos de GPGPU 2562 pueden ejecutar físicamente instrucciones de SIMD4, SIMD8 y SIMD16 y ejecutar lógicamente instrucciones de SIMD1, SIMD2 y SIMD32. Las instrucciones de SIMD para los núcleos de GPGPU pueden generarse en tiempo de compilación por un compilador de sombreador o generase automáticamente cuando se ejecutan programas escritos y compilados para arquitecturas de único programa, múltiples datos (SPMD) o SIMT. Múltiples hilos de un programa configurado para el modelo de ejecución de SIMT pueden ejecutarse a través de una única instrucción de SIMD. Por ejemplo y en una realización, ocho hilos de SIMT que realizan las mismas o similares operaciones pueden ejecutarse en paralelo a través de una única unidad lógica de SIMD8.

La interconexión de memoria y memoria caché 2568 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador gráfico 324 al archivo de registro 2558 y a la memoria compartida 2570. En una realización, la interconexión de memoria y memoria caché 2568 es una interconexión de barra cruzada que permite que la unidad de carga/almacenamiento 2566 implemente operaciones de carga y almacenamiento entre la memoria compartida 2570 y el archivo de registro 2558. El archivo de registro 2558 puede operar en la misma frecuencia que los núcleos de GPGPU 2562, por lo tanto la transferencia de datos entre los núcleos de GPGPU 2562

y el archivo de registro 2558 es de latencia muy baja. La memoria compartida 2570 puede usarse para habilitar comunicación entre hilos que se ejecutan en las unidades funcionales dentro del multiprocesador gráfico 2534. La memoria caché 2572 puede usarse como una memoria caché de datos por ejemplo, para almacenar en memoria caché datos de textura comunicados entre las unidades funcionales y la unidad de textura 2536. La memoria compartida 2570 también puede usarse como una memoria caché gestionada por programa. Los hilos que se ejecutan en los núcleos de GPGPU 2562 pueden almacenar programáticamente datos dentro de la memoria compartida además de los datos almacenados en memoria caché automáticamente que se almacenan dentro de la memoria caché 2572.

Las Figuras 26A-26B ilustran multiprocesadores gráficos adicionales, de acuerdo con realizaciones. Los multiprocesadores gráficos 2625, 2650 ilustrados son variantes del multiprocesador gráfico 2534 de **la Figura 25C**. Los multiprocesadores gráficos 2625, 2650 ilustrados pueden configurarse como un multiprocesador de transmisión (SM) con capacidad de ejecución simultánea de un gran número de hilos de ejecución.

La Figura 26A muestra un multiprocesador gráfico 2625 de acuerdo con una realización adicional. El multiprocesador gráfico 2625 incluye múltiples instancias adicionales de unidades de recursos de ejecución en relación con el multiprocesador gráfico 234 de **la Figura 25D**. Por ejemplo, el multiprocesador gráfico 525 puede incluir múltiples instancias de la unidad de instrucción 2632A-2632B, el archivo de registro 2634A-2634B y la unidad o unidades de textura 2644A-2644B. El multiprocesador gráfico 2625 también incluye múltiples conjuntos de unidades de ejecución gráficas o de cálculo (por ejemplo, el núcleo de GPGPU 2636A-2636B, el núcleo de GPGPU 2637A-2637B, el núcleo de GPGPU 2638A-2638B) y múltiples conjuntos de unidades de carga/almacenamiento 2640A-2640B. En una realización, las unidades de recursos de ejecución tienen una memoria caché de instrucciones 2630, una memoria caché de textura y/o datos 2642 y una memoria compartida 2646 común.

Los diversos componentes pueden comunicarse a través de un tejido de interconexión 2627. En una realización, el tejido de interconexión 2627 incluye una o más conmutaciones de barra cruzada para habilitar comunicación entre los diversos componentes del multiprocesador gráfico 2625. En una realización, el tejido de interconexión 2627 es una capa de tejido de red de alta velocidad separada sobre la que se apila cada componente del multiprocesador gráfico 2625. Los componentes del multiprocesador gráfico 2625 se comunican con componentes remotos a través del tejido de interconexión 2627. Por ejemplo, cada uno de los núcleos de GPGPU 2636A-2636B, 2637A-2637B y 2678A-2638B puede comunicarse con la memoria compartida 2646 a través del tejido de interconexión 2627. El tejido de interconexión 2627 puede arbitrar la comunicación dentro del multiprocesador gráfico 2625 para garantizar una asignación de ancho de banda equitativa entre componentes.

La Figura 26B muestra un multiprocesador gráfico 2650 de acuerdo con una realización adicional. El procesador gráfico incluye múltiples conjuntos de recursos de ejecución 2656A-2656D, en donde cada conjunto de recurso de ejecución incluye múltiples unidades de instrucción, archivo de registros, núcleos de GPGPU, y unidades de carga y almacenamiento, como se ilustra en **la Figura 25D** y **la Figura 26A**. Los recursos de ejecución 2656A-2656D pueden trabajar conjuntamente con la unidad o unidades de textura 2660A-2660D para operaciones de textura, mientras comparten una memoria caché de instrucciones 2654, y la memoria compartida 2662. En una realización, los recursos de ejecución 2656A-2656D pueden compartir una memoria caché de instrucciones 2654 y memoria compartida 2662, así como múltiples instancias de una memoria caché de textura y/o datos 2658A-2658B. Los diversos componentes pueden comunicarse a través de un tejido de interconexión 2652 similar al tejido de interconexión 2627 de **la Figura 26A**.

Expertos en la materia entenderán que la arquitectura descrita en **las Figuras 24**, **25A-25D** y **26A-26B** son descriptivas y no limitantes del alcance de las presentes realizaciones. Por lo tanto, las técnicas descritas en este documento pueden implementarse en cualquier unidad de procesamiento configurada apropiadamente, incluyendo, sin limitación, uno o más procesadores de aplicación móvil, una o más unidades de procesamiento central (CPU) de escritorio o servidor que incluyen CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tal como la unidad de procesamiento paralelo 2502 de **la Figura 25**, así como uno o más procesadores gráficos o unidades de procesamiento de fin especial, sin alejarse del alcance de las realizaciones descritas en este documento.

En algunas realizaciones, un procesador paralelo o GPGPU como se describe en este documento se acopla comunicativamente a núcleos de anfitrión/procesador para acelerar operaciones gráficas, operaciones de aprendizaje automático, operaciones de análisis de patrón y diversas funciones de GPU de fin general (GPGPU). La GPU puede acoplarse comunicativamente al procesador de anfitrión/núcleos a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otras realizaciones, la GPU puede integrarse en el mismo paquete o chip que los núcleos y acoplarse comunicativamente a los núcleos a través de un bus/interconexión de procesador interno (es decir, interno al paquete o chip). Independientemente de la manera en la que se conecta la GPU, los núcleos de procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU usa, a continuación, circuitería/lógica especializada para procesar de forma eficiente estos comandos/instrucciones.

Técnicas para interconexión de GPU a procesador de anfitrión

La Figura 27A ilustra una arquitectura ilustrativa en la que una pluralidad de GPU 2710-2713 se acoplan comunicativamente a una pluralidad de procesadores de múltiples núcleos 2705-2706 a través de enlaces de alta velocidad 2740-2743 (por ejemplo, buses, interconexiones de punto a punto, etc.). En una realización, los enlaces de alta velocidad 2740-2743 soportan un caudal de comunicación de 4 GB/s, 30 GB/s, 80 GB/s o mayor, dependiendo de la implementación. Pueden usarse diversos protocolos de interconexión incluyendo, pero sin limitación a, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no se limitan a ningún protocolo o caudal de comunicación particular.

Además, en una realización, dos o más de las GPU 2710-2713 se interconectan a través de enlaces de alta velocidad 2744-2745, que pueden implementarse usando los mismos o diferentes protocolos/enlaces que los usados para los enlaces de alta velocidad 2740-2743. De manera similar, dos o más de los procesadores de múltiples núcleos 2705-2706 pueden conectarse a través del enlace de alta velocidad 2733 que puede ser buses de múltiples procesadores simétricos (SMP) que operan a 20 GB/s, 30 GB/s, 120 GB/s o más. Como alternativa, toda la comunicación entre los diversos componentes de sistema mostrados en la **Figura 27A** puede lograrse usando los mismos protocolos/enlaces (por ejemplo, a través de un tejido de interconexión común). Como se ha mencionado, sin embargo, los principios subyacentes de la invención no se limitan a cualquier tipo particular de tecnología de interconexión.

En una realización, cada procesador de múltiples núcleos 2705-2706 se acopla comunicativamente a una memoria de procesador 2701-2702, a través de las interconexiones de memoria 2730-2731, respectivamente, y cada GPU 2710-2713 se acopla comunicativamente a la memoria de GPU 2720-2723 a través de las interconexiones de memoria de GPU 2750-2753, respectivamente. Las interconexiones de memoria 2730-2731 y 2750-2753 pueden utilizar las mismas o diferentes tecnologías de acceso a memoria. A modo de ejemplo, y no como limitación, las memorias de procesador 2701-2702 y memorias de GPU 2720-2723 pueden ser memorias volátiles tales como memorias de acceso aleatorio dinámicas (DRAM) (incluyendo DRAM apiladas), SDRAM de DDR Gráfica (GDDR) (por ejemplo, GDDR5, GDDR6), o Memoria de Ancho de Banda Alto (HBM) y/o pueden ser memorias no volátiles tales como 3D XPoint o Nano-Ram. En una realización, alguna porción de las memorias puede ser memoria volátil y otra porción puede ser memoria no volátil (por ejemplo, usando una jerarquía de memorias de dos niveles (2LM)).

Como se describe a continuación, aunque los diversos procesadores 2705-2706 y las GPU 2710-2713 pueden acoplarse físicamente a una memoria 2701-2702, 2720-2723 particular, respectivamente, puede implementarse una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtuales (también denominado espacio de "direcciones efectivas") se distribuye entre todas las diversas memorias físicas. Por ejemplo, cada una de las memorias de procesador 2701-2702 puede comprender 64 GB del espacio de direcciones de memoria de sistema y cada una de las memorias de GPU 2720-2723 puede comprender 32 GB del espacio de direcciones de memoria de sistema (resultando en un total de 256 GB de memoria direccionable en este ejemplo).

La Figura 27B ilustra detalles adicionales para una interconexión entre un procesador de múltiples núcleos 2707 y un módulo de aceleración gráfica 2746 de acuerdo con una realización. El módulo de aceleración gráfica 2746 puede incluir uno o más chips de GPU integrados en una tarjeta de línea que se acopla al procesador 2707 a través del enlace de alta velocidad 2740. Como alternativa, el módulo de aceleración gráfica 2746 puede integrarse en el mismo paquete o chip que el procesador 2707.

El procesador 2707 ilustrado incluye una pluralidad de núcleos 2760A-2760D, cada uno con una memoria intermedia de traducción anticipada 2761A-2761D y una o más memorias caché 2762A-2762D. Los núcleos pueden incluir diversos otros componentes para ejecutar instrucciones y procesar datos que no se ilustran para evitar obstaculizar los principios subyacentes de la invención (por ejemplo, unidades de búsqueda de instrucción, unidades de predicción de rama, decodificadores, unidades de ejecución, memorias intermedias de reordenación, etc.). Las memorias caché 2762A-2762D pueden comprender memorias caché de nivel 1 (L1) y nivel 2 (L2). Además, una o más memorias caché compartidas 2726 pueden incluirse en la jerarquía de almacenamiento en memoria caché y compartirse por conjuntos de los núcleos 2760A-2760D. Por ejemplo, una realización del procesador 2707 incluye 24 núcleos, cada uno con su propia memoria caché L1, doce memorias caché L2 compartidas, y doce memorias caché L3 compartidas. En esta realización, una de las memorias caché L2 y L3 se comparten por dos núcleos adyacentes. El procesador 2707 y el módulo de integración de acelerador gráfico 2746 se conectan con la memoria de sistema 2741, que puede incluir las memorias de procesador 2701-2702

La coherencia se mantiene para datos e instrucciones almacenadas en las diversas memorias caché 2762A-2762D, 2756 y la memoria de sistema 2741 a través de comunicación entre núcleos a través de un bus de coherencia 2764. Por ejemplo, cada memoria caché puede tener lógica/circuitería de coherencia de memoria caché asociada con la misma para comunicarse a través del bus de coherencia 2764 en respuesta a lecturas o escrituras detectadas en líneas de memoria caché particulares. En una implementación, se implementa un protocolo de inspección de memoria caché a través del bus de coherencia 2764 para inspeccionar accesos a memoria caché. Los expertos en la materia entienden bien las técnicas de investigación/coherencia de memoria caché y no se describirán en detalle en este punto para evitar obstaculizar los principios subyacentes de la invención.

En una realización, un circuito intermediario 2725 acopla comunicativamente el módulo de aceleración gráfica 2746 al bus de coherencia 2764, permitiendo que el módulo de aceleración gráfica 2746 participe en el protocolo de coherencia

de memoria caché como un par de los núcleos. En particular, una interfaz 2735 proporciona conectividad al circuito intermediario 2725 a través del enlace de alta velocidad 2740 (por ejemplo, un bus de PCIe, NVLink, etc.) y una interfaz 2737 conecta el módulo de aceleración gráfica 2746 al enlace 2740.

En una implementación, un circuito de integración de acelerador 2736 proporciona gestión de memoria caché, acceso a memoria, gestión de contexto y servicios de gestión de interrupción en nombre de una pluralidad de motores de procesamiento gráfico 2731, 2732, N del módulo de aceleración gráfica 2746. Cada uno de los motores de procesamiento gráfico 2731, 2732, N puede comprender una unidad de procesamiento gráfico (GPU) separada. Como alternativa, los motores de procesamiento gráfico 2731, 2732, N pueden comprender diferentes tipos de motores de procesamiento gráfico dentro de una GPU tales como unidades de ejecución de gráficos, motores de procesamiento de medios (por ejemplo, codificadores/decodificadores de vídeo), muestreadores y motores de blit. En otras palabras, el módulo de aceleración gráfica puede ser una GPU con una pluralidad de motores de procesamiento gráfico 2731-2732, N o los motores de procesamiento gráfico 2731-2732, N pueden ser GPU individuales integrados en un paquete, tarjeta de línea o chip común.

En una realización, el circuito de integración de acelerador 2736 incluye una unidad de gestión de memoria (MMU) 2739 para realizar diversas funciones de gestión de memoria tales como traducciones de memoria de virtual a física (también denominadas traducciones de memoria efectiva a real) y protocolos de acceso a memoria para acceder a la memoria de sistema 2741. La MMU 2739 también puede incluir una memoria intermedia de traducción anticipada (TLB) (no mostrada) para almacenar en memoria caché las traducciones de direcciones virtuales/efectivas a físicas/reales. En una implementación, una memoria caché 2738 almacena comandos y datos para acceso eficiente por los motores de procesamiento gráfico 2731-2732, N. En una realización, los datos almacenados en la memoria caché 2738 y las memorias gráficas 2733-2734, N se mantienen coherentes con las memorias caché de núcleo 2762A-2762D, 2756 y la memoria de sistema 2711. Como se ha mencionado, esto puede conseguirse a través del circuito intermediario 2725 que toma parte en el mecanismo de coherencia de memoria caché en nombre de la memoria caché 2738 y las memorias 2733-2734, N (por ejemplo, enviando actualizaciones a la memoria caché 2738 relacionadas con modificaciones/accesos de líneas de memoria caché en las memorias caché de procesador 2762A-2762D, 2756 y recibiendo actualizaciones desde la memoria caché 2738).

Un conjunto de registros 2745 almacenan datos de contexto para hilos ejecutados por los motores de procesamiento gráfico 2731-2732, N y un circuito de gestión de contexto 2748 gestiona los contextos de hilo. Por ejemplo, el circuito de gestión de contexto 2748 puede realizar operaciones de guardado y restauración para guardar y restaurar contextos de los diversos hilos durante conmutaciones de contexto (por ejemplo, en donde se guarda un primer hilo y se almacena un segundo hilo de modo que el segundo hilo puede ejecutarse por un motor de procesamiento gráfico). Por ejemplo, en una conmutación de contexto, el circuito de gestión de contexto 2748 puede almacenar valores de registro actuales en una región designada en memoria (por ejemplo, identificada por un puntero de contexto). A continuación, puede restaurar los valores de registro cuando vuelve al contexto. En una realización, un circuito de gestión de interrupción 2747 recibe y procesa interrupciones recibidas desde dispositivos de sistema.

En una implementación, la MMU 2739 traduce direcciones virtuales/efectivas de un motor de procesamiento gráfico 2731 a direcciones reales/físicas en la memoria de sistema 2711. Una realización del circuito de integración de acelerador 2736 soporta múltiples (por ejemplo, 4, 8, 16) módulos de acelerador gráfico 2746 y/u otros dispositivos aceleradores. El módulo de acelerador gráfico 2746 puede dedicarse a una única aplicación ejecutada en el procesador 2707 o puede compartirse entre múltiples aplicaciones. En una realización, se presenta un entorno de ejecución de gráficos virtualizado en el que los recursos de los motores de procesamiento gráfico 2731-2732, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos pueden subdividirse en "sectores" que se asignan a diferentes VM y/o aplicaciones basándose en los requisitos y prioridades de procesamiento asociados con las VM y/o aplicaciones.

Por lo tanto, el circuito de integración de acelerador actúa como un puente al sistema para el módulo de aceleración gráfica 2746 y proporciona servicios de traducción de direcciones y memoria caché de sistema. Además, el circuito de integración de acelerador 2736 puede proporcionar instalaciones de virtualización para que el procesador de anfitrión gestione la virtualización de los motores de procesamiento gráfico, interrupciones y gestión de memoria.

Debido a que los recursos de hardware de los motores de procesamiento gráfico 2731-2732, N se correlacionan explícitamente con el espacio de direcciones reales vistas en el procesador de anfitrión 2707, cualquier procesador de anfitrión puede direccionar estos recursos directamente usando un valor de dirección efectiva. Una función del circuito de integración de acelerador 2736, en una realización, es la separación física de los motores de procesamiento gráfico 2731-2732, N de modo que aparecen al sistema como unidades independientes.

Como se ha mencionado, en la realización ilustrada, una o más memorias gráficas 2733-2734, M se acoplan a cada uno de los motores de procesamiento gráfico 2731-2732, N, respectivamente. Las memorias gráficas 2733-2734, M almacenan instrucciones y datos que se procesan por cada uno de los motores de procesamiento gráfico 2731-2732, N. Las memorias gráficas 2733-2734, M pueden ser memorias volátiles tales como DRAM (incluyendo DRAM apiladas), memoria de GDDR (por ejemplo, GDDR5, GDDR6) o HBM, y/o pueden ser memorias no volátiles tales como 3D XPoint o Nano-Ram.

En una realización, para reducir tráfico de datos a través del enlace 2740, se usan técnicas de polarización para garantizar que los datos almacenados en memorias gráficas 2733-2734, M son datos que se usarán más frecuentemente por los motores de procesamiento gráfico 2731-2732, N y no se usarán preferentemente por los núcleos 2760A-2760D (al menos no frecuentemente). De manera similar, el mecanismo de polarización intenta mantener los datos necesarios por los núcleos (y preferentemente no los motores de procesamiento gráfico 2731-2732, N) dentro de las memorias caché 2762A-2762D, 2756 de los núcleos y memoria de sistema 2711.

La Figura 27C ilustra otra realización en la que el circuito de integración de acelerador 2736 se integra dentro del procesador 2707. En esta realización, los motores de procesamiento gráfico 2731-2732, N se comunican directamente a través del enlace de alta velocidad 2740 al circuito de integración de acelerador 2736 a través de la interfaz 2737 y la interfaz 2735 (que, de nuevo, puede utilizarse cualquier forma de bus o protocolo de interfaz). El circuito de integración de acelerador 2736 puede realizar las mismas operaciones que las descritas con respecto a la **Figura 27B**, pero potencialmente en un caudal mayor dada su proximidad cercana al bus de coherencia 2762 y las memorias caché 2762A-2762D, 2726.

Una realización soporta diferentes modelos de programación que incluyen un modelo de programación de proceso especializado (sin virtualización de módulo de aceleración gráfica) y modelos de programación compartidos (con virtualización). Los últimos pueden incluir modelos de programación que se controlan por el circuito de integración de acelerador 2736 y modelos de programación que se controlan por el módulo de aceleración gráfica 2746.

En una realización del modelo de proceso especializado, los motores de procesamiento gráfico 2731-2732, N se dedican a una única aplicación o proceso en un único sistema operativo. La única aplicación puede canalizar otras peticiones de aplicación a los motores gráficos 2731-2732, N, proporcionando virtualización dentro de una VM/partición.

En los modelos de programación de proceso especializado, los motores de procesamiento gráfico 2731-2732, N, pueden compartirse por múltiples VM/particiones de aplicación. Los modelos compartidos requieren un hipervisor de sistema para virtualizar los motores de procesamiento gráfico 2731-2732, N para permitir acceso por cada sistema operativo. Para sistemas de una única partición sin un hipervisor, los motores de procesamiento gráfico 2731-2732, N son propiedad del sistema operativo. En ambos casos, el sistema operativo puede virtualizar los motores de procesamiento gráfico 2731-2732, N para proporcionar acceso a cada proceso o aplicación.

Para el modelo de programación compartido, el módulo de aceleración gráfica 2746 o un motor de procesamiento gráfico individual 2731-2732, N selecciona un elemento de proceso usando un tratamiento de proceso. En una realización, los elementos de proceso se almacenan en la memoria de sistema 2711 y son direccionables usando las técnicas de traducción de direcciones efectivas a direcciones reales descritas en este documento. El tratamiento de proceso puede ser un valor específico de la implementación proporcionado al proceso de anfitrión cuando registra su contexto con el motor de procesamiento gráfico 2731-2732, N (es decir, llamando al software de sistema para añadir el elemento de proceso a la lista enlazada de elementos de proceso). Los 16 bits inferiores del tratamiento de proceso pueden ser el desplazamiento del elemento de proceso dentro de la lista enlazada de elementos de proceso.

La Figura 27D ilustra un sector de integración de acelerador 2790 ilustrativo. Como se usa en este documento, un "sector" comprende una porción especificada de los recursos de procesamiento del circuito de integración de acelerador 2736. El espacio de direcciones efectivas de aplicación 2782 dentro de la memoria de sistema 2711 almacena los elementos de proceso 2783. En una realización, los elementos de proceso 2783 se almacenan en respuesta a las invocaciones de GPU 2781 desde las aplicaciones 2780 ejecutadas en el procesador 2707. Un elemento de proceso 2783 contiene el estado de proceso para la correspondiente aplicación 2780. Un descriptor de trabajo (WD) 2784 contenido en el elemento de proceso 2783 puede ser una única petición de trabajo por una aplicación o puede contener un puntero a una cola de trabajos. En el último caso, el WD 2784 es un puntero a la cola de peticiones de trabajo en el espacio de direcciones 2782 de la aplicación.

El módulo de aceleración gráfica 2746 y/o los motores de procesamiento gráfico individuales 2731-2732, N pueden compartirse por todos o un subconjunto de los procesos en el sistema. Realizaciones de la invención incluyen una infraestructura para establecer el estado de proceso y enviar un WD 2784 a un módulo de aceleración gráfica 2746 para iniciar un trabajo en un entorno virtualizado.

En una implementación, el modelo de programación de proceso especializado es específico de la implementación. En este modelo, un único proceso es propietario del módulo de aceleración gráfica 2746 o un motor de procesamiento gráfico individual 2731. Debido a que el módulo de aceleración gráfica 2746 es propiedad de un único proceso, el hipervisor inicializa el circuito de integración de acelerador 2736 para la partición propietaria y el sistema operativo inicializa el circuito de integración de acelerador 2736 para proceso propietario en el momento en el que se asigna el módulo de aceleración gráfica 2746.

En la operación, una unidad de búsqueda de WD 2791 en el sector de integración de acelerador 2790 busca el siguiente WD 2784 que incluye una indicación del trabajo que hay que hacer por uno de los motores de procesamiento

gráfico del módulo de aceleración gráfica 2746. Los datos del WD 2784 pueden almacenarse en registros 2745 y usarse por la MMU 2739, circuito de gestión de interrupción 2747 y/o circuito de gestión de contexto 2746, como se ilustra. Por ejemplo, una realización de la MMU 2739 incluye circuitería de recorrido de segmento/página para acceder a tablas de segmento/paginación 2786 dentro del espacio de direcciones virtuales de SO 2785. El circuito de gestión de interrupción 2747 puede procesar eventos de interrupción 2792 recibidos desde el módulo de aceleración gráfica 2746. Cuando se realizan operaciones gráficas, una dirección efectiva 2793 generada por un motor de procesamiento gráfico 2731-2732, N se traduce a una dirección real por la MMU 2739.

En una realización, el mismo conjunto de registros 2745 se duplica por cada motor de procesamiento gráfico 2731-2732, N y/o módulo de aceleración gráfica 2746 y puede inicializarse por el hipervisor o sistema operativo. Cada uno de estos registros duplicados puede incluirse en un sector de integración de acelerador 2790. En la **Tabla 1** se muestran registros ilustrativos que pueden inicializarse por el hipervisor.

Tabla 1 - Registros inicializados por hipervisor

1	Registro de control de sector
2	Puntero de área de procesos planificados de dirección real (RA)
3	Registro de anulación de máscara de autoridad
4	Desplazamiento de entrada de tabla de vector de interrupción
5	Límite de entrada de tabla de vector de interrupción
6	Registro de estados
7	ID de partición lógica
8	Puntero de grabación de utilización de acelerador de hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

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

Tabla 2 - Registros inicializados por sistema operativo

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

En una realización, cada WD 2784 es específico a un módulo de aceleración gráfica 2746 y/o motor de procesamiento gráfico 2731-2732, N particular. Contiene toda la información que requiere un motor de procesamiento gráfico 2731-2732, N para hacer su trabajo o puede ser un puntero a una ubicación de memoria en donde la aplicación ha establecido una cola de comandos de trabajo que hay que completar.

La Figura 27E ilustra detalles adicionales para una realización de un modelo compartido. Esta realización incluye un espacio de direcciones reales de hipervisor 2798 en el que se almacena una lista de elementos de proceso 2799. El espacio de direcciones reales de hipervisor 2798 es accesible a través de un hipervisor 2796 que virtualiza los motores de módulo de aceleración gráfica para el sistema operativo 2795.

Los modelos de programación compartidos permiten que todos o un subconjunto de procesos de entre todas o un subconjunto de particiones en el sistema usen un módulo de aceleración gráfica 2746. Existen dos modelos de programación en donde el módulo de aceleración gráfica 2746 se comparte por múltiples procesos y particiones: uso compartido por sector de tiempo y uso compartido directo de gráficos.

En este modelo, el sistema hipervisor 2796 es propietario del módulo de aceleración gráfica 2746 y hace su función disponible para topos los sistemas operativos 2795. Para que un módulo de aceleración gráfica 2746 soporte virtualización por el sistema hipervisor 2796, el módulo de aceleración gráfica 2746 puede cumplir con los siguientes requisitos: 1) La petición de trabajo de una aplicación debe ser autónoma (es decir, el estado no necesita mantenerse entre trabajos), o el módulo de aceleración gráfica 2746 debe proporcionar un mecanismo de guardado y restauración de contexto. 2) La petición de trabajo de una aplicación se garantiza por el módulo de aceleración gráfica 2746 para completar en una cantidad especificada de tiempo, incluyendo cualquier fallo de traducción, o el módulo de aceleración gráfica 2746 proporciona la capacidad de adelantar el procesamiento del trabajo. 3) El módulo de aceleración gráfica 2746 debe garantizarse la equidad entre procesos cuando opera en el modelo de programación de uso compartido directo.

En una realización, para el modelo de uso compartido, se requiere que la aplicación 2780 haga una llamada de sistema del sistema operativo 2795 con un tipo de módulo de aceleración gráfica 2746, un descriptor de trabajo (WD), un valor

de registro de máscara de autoridad (AMR) y un puntero de área de guardado/restauración de contexto (CSRP). El tipo de módulo de aceleración gráfica 2746 describe la función de aceleración dirigida para la llamada de sistema. El tipo de módulo de aceleración gráfica 4276 puede ser un valor específico del sistema. El WD se formatea específicamente para el módulo de aceleración gráfica 2746 y puede ser en forma de un comando de módulo de aceleración gráfica 2746, un puntero de dirección efectiva a una estructura definida por usuario, un puntero de dirección efectiva a una cola de comandos, o cualquier otra estructura de datos para describir el trabajo a hacer por el módulo de aceleración gráfica 2746. En una realización, el valor de AMR es el estado de AMR a usar para el proceso actual. El valor pasado al sistema operativo es similar a una aplicación que establece el AMR. Si las implementaciones de circuito de integración de acelerador 2736 y de módulo de aceleración gráfica 2746 no soportan un Registro de Anulación de Máscara de Autoridad de Usuario (UAMOR), el sistema operativo puede aplicar el valor de UAMOR actual al valor de AMR antes de pasar el AMR en la llamada de hipervisor. El hipervisor 2796 puede aplicar opcionalmente el valor de registro de anulación de máscara de autoridad (AMOR) actual antes de situar el AMR en el elemento de proceso 2783. En una realización, el CSRP es uno de los registros 2745 que contienen la dirección efectiva de un área en el espacio de direcciones 2782 de la aplicación para que el módulo de aceleración gráfica 2746 guarde y restaure el estado de contexto. Este puntero es opcional si no se requiere ningún estado a guardar entre trabajos o cuando se adelanta un trabajo. El área de guardado/restauración de contexto puede anclarse en memoria de sistema.

Tras recibir la llamada de sistema, el sistema operativo 2795 puede verificar que la aplicación 2780 ha registrado y se le ha dado la autoridad para usar el módulo de aceleración gráfica 2746. El sistema operativo 2795 llama, a continuación, al hipervisor 2796 con la información mostrada en la **Tabla 3.**

Tabla 3 - Parámetros de llamada de SO a hipervisor

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

Tras recibir la llamada de hipervisor, el hipervisor 2796 verifica que el sistema operativo 2795 ha registrado y se le ha dado la autoridad de usar el módulo de aceleración gráfica 2746. El hipervisor 2796 pone, a continuación, el elemento de proceso 2783 en la lista enlazada de elementos de proceso para el correspondiente tipo de módulo de aceleración gráfica 2746. El elemento de proceso puede incluir la información mostrada en la **Tabla 4.**

Tabla 4 - Información de elemento de proceso

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

En una realización, el hipervisor inicializa una pluralidad de registros 2745 de sector de integración de acelerador 2790.

Como se ilustra en la **Figura 27F**, una realización de la invención emplea una memoria unificada direccionable a través de un espacio de direcciones virtuales de memoria común usado para acceder a las memorias de procesador físicas 2701-2702 y las memorias de GPU 2720-2723. En esta implementación, las operaciones ejecutadas en las GPU 2710-2713 utilizan el mismo espacio de direcciones de memoria virtuales/efectivas para acceder a las memorias de procesadores 2701-2702 y viceversa, simplificando de este modo la programabilidad. En una realización, una primera porción del espacio de direcciones virtuales/efectivas se asigna a la memoria de procesador 2701, una segunda porción a la segunda memoria de procesador 2702, una tercera porción a la memoria de GPU 2720, y así sucesivamente. Todo el espacio de memoria virtual/efectiva (en ocasiones denominada como el espacio de direcciones efectivas) se distribuye de este modo a través de cada una de las memorias de procesador 2701-2702 y

memorias de GPU 2720-2723, permitiendo que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual correlacionada con esa memoria.

En una realización, la circuitería de gestión de polarización/coherencia 2794A-2794E dentro de una o más de las MMU 2739A-2739E garantiza la coherencia de memoria caché entre las memorias caché de los procesadores de anfitrión (por ejemplo, 2705) y las GPU 2710-2713 e implementa técnicas de polarización que indican las memorias físicas en las que deberían almacenarse ciertos tipos de datos. Mientras múltiples instancias de circuitería de gestión de polarización/coherencia 2794A-2794E se ilustran en la **Figura 27F**, la circuitería de polarización/coherencia puede implementarse dentro de la MMU de uno o más procesadores de anfitrión 2705 y/o dentro del circuito de integración de acelerador 2736.

Una realización permite que la memoria conectada a GPU 2720-2723 se correlacione como parte de memoria de sistema, y se acceda usando tecnología de memoria virtual compartida (SVM), pero sin sufrir los inconvenientes de rendimiento típicos asociados con coherencia de memoria caché de sistema completa. La capacidad de que la memoria conectada a GPU 2720-2723 sea accedida como memoria de sistema sin tara de coherencia de memoria caché onerosa proporciona un entorno de operación beneficioso para descarga de GPU. Esta disposición permite que el software del procesador de anfitrión 2705 configure operandos y acceda a resultados de cálculo, sin la tara de copias de datos de DMA de E/S tradicionales. Tales copias tradicionales implican llamadas de controlador, interrupciones y accesos de E/S con correlación de memoria (MMIO) que son todas ineficientes en relación con accesos de memoria simples. Al mismo tiempo, la capacidad de acceder a la memoria conectada a GPU 2720-2723 sin taras de coherencia de memoria caché puede ser crítica al tiempo de ejecución de un cálculo descargado. En casos con tráfico de memoria de escritura de transmisión sustancial, por ejemplo, la tara de coherencia de memoria caché puede reducir significativamente el ancho de banda de escritura efectivo visto por una GPU 2710-2713. La eficiencia de configuración de operando, la eficiencia de acceso a resultados y la eficiencia de cálculo de GPU tienen todas una función en la determinación de la efectividad de la descarga de GPU.

En una implementación, la selección de entre la polarización de GPU y la polarización de procesador de anfitrión se acciona por una estructura de datos de seguimiento de polarización. Puede usarse una tabla de polarización, por ejemplo, que puede ser una estructura granular de página (es decir, controlada en la granularidad de una página de memoria) que incluye 1 o 2 bits por página de memoria conectada a GPU. La tabla de polarización puede implementarse en un intervalo de memoria robado de una o más memorias conectadas a GPU 2720-2723, con o sin una memoria caché de polarización en la GPU 2710-2713 (por ejemplo, a entradas de memoria caché usadas frecuentemente/recientemente de la tabla de polarización). Como alternativa, toda la tabla de polarización puede mantenerse dentro de la GPU.

En una implementación, se accede a la entrada de tabla de polarización asociada con cada acceso a la memoria conectada a GPU 2720-2723 antes del acceso real a la memoria de GPU, provocando las siguientes operaciones. Primero, las peticiones locales desde la GPU 2710-2713 que encuentran su página en la polarización de GPU se reenvían directamente a una correspondiente memoria de GPU 2720-2723. Las peticiones locales desde la GPU que encuentran su página en la polarización de anfitrión se reenvían al procesador 2705 (por ejemplo, a través de un enlace de alta velocidad como se ha analizado anteriormente). En una realización, las peticiones desde el procesador 2705 que encuentran la página solicitada en la polarización de procesador de anfitrión completan la petición como una lectura de memoria normal. Como alternativa, las peticiones dirigidas a una página con polarización de GPU pueden reenviarse a la GPU 2710-2713. La GPU puede pasar, a continuación, la página a una polarización de procesador de anfitrión si no está en la actualidad usando la página.

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

Un mecanismo para cambiar el estado de polarización emplea una llamada de API (por ejemplo, OpenCL), que, a su vez, llama al controlador de visualización de la GPU que, a su vez, envía un mensaje (o pone en cola un descriptor de comando) a la GPU que dirige el mismo para cambiar el estado de polarización y, para algunas transiciones, realizan una operación de vaciado de memoria caché en el anfitrión. La operación de vaciado de memoria caché se requiere para una transición desde la polarización del procesador de anfitrión 2705 a polarización de GPU, pero no se requiere para la transición opuesta.

En una realización, la coherencia de memoria caché se mantiene representando temporalmente páginas con polarización de GPU que no se pueden almacenar en memoria caché por el procesador de anfitrión 2705. Para acceder a estas páginas, el procesador 2705 puede solicitar accesos desde la GPU 2710 que puede o puede no conceder acceso inmediatamente, dependiendo de la implementación. Por lo tanto, para reducir la comunicación entre el procesador 2705 y la GPU 2710 es beneficioso garantizar que páginas con polarización de GPU son las que se requieren por la GPU, pero no el procesador de anfitrión 2705 y viceversa.

Canalización de procesamiento gráfico

La Figura 28 ilustra una canalización de procesamiento gráfico 2800, de acuerdo con una realización. En una realización, un procesador gráfico puede implementar la canalización de procesamiento gráfico 2800 ilustrada. El procesador gráfico puede incluirse dentro de los subsistemas de procesamiento paralelo como se describe en este documento, tal como el procesador paralelo 2500 de **la Figura 25**, que, en una realización, es una variante del procesador o procesadores paralelos 2412 de **la Figura 24**. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento gráfico 2800 a través de una o más instancias de la unidad de procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 2502 de la Figura 25) como se describe en este documento. Por ejemplo, una unidad de sombreador (por ejemplo, el multiprocesador gráfico 2534 de la Figura 26) puede configurarse para realizar las funciones de uno o más de una unidad de procesamiento de vértices 2804, una unidad de procesamiento de control de teselación 2808, una unidad de procesamiento de evaluación de teselación 2812, una unidad de procesamiento de geometría 2816 y una unidad de procesamiento de fragmento/píxel 2824. Las funciones de ensamblador de datos 2802, ensambladores de primitivas 2806, 2814, 2818, unidad de teselación 2810, rasterizador 2822 y unidad de operaciones de trama 2826 también pueden realizarse por otros motores de procesamiento dentro de una agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 214 de **la Figura 3**) y una correspondiente unidad de partición (por ejemplo, la unidad de partición 220A-220N de **la Figura 2**). La canalización de procesamiento gráfico 2800 también puede implementarse usando unidades de procesamiento especializadas para una o más funciones. En una realización, una o más porciones de la canalización de procesamiento gráfico 2800 pueden realizarse mediante lógica de procesamiento paralelo dentro de un procesador de fin general (por ejemplo, CPU). En una realización, una o más porciones de la canalización de procesamiento gráfico 2800 pueden acceder a memoria en chip (por ejemplo, la memoria de procesador paralelo 2522 como en **la Figura 25**) a través de una interfaz de memoria 2828, que puede ser una instancia de la interfaz de memoria 2518 de **la Figura 25**.

En una realización, el ensamblador de datos 2802 es una unidad de procesamiento que recopila datos de vértice para superficies y primitivas. El ensamblador de datos 2802 emite, a continuación, los datos de vértice, incluyendo los atributos de vértice, a la unidad de procesamiento de vértices 2804. La unidad de procesamiento de vértices 2804 es una unidad de ejecución programable que ejecuta programas de sombreador de vértices, iluminando y transformando datos de vértice como se especifica por los programas de sombreador de vértices. La unidad de procesamiento de vértices 2804 lee datos que se almacenan en memoria caché, memoria local o de sistema para su uso en el procesamiento de los datos de vértice y puede programarse para transformar los datos de vértice desde una representación de coordinadas basada en objeto a un espacio de coordenadas de espacio global o un espacio de coordinadas de dispositivo normalizado.

Una primera instancia de un ensamblador de primitivas 2806 recibe atributos de vértice desde la unidad de procesamiento de vértices 280. Las lecturas del ensamblador de primitivas 2806 almacenaron atributos de vértice según se necesite y construye primitivas gráficas para su procesamiento por la unidad de procesamiento de control de teselación 2808. Las primitivas gráficas incluyen triángulos, segmentos de línea, puntos, parches y así sucesivamente, según se soportan por diversas interfaces de programación de aplicación (API) de procesamiento gráfico.

La unidad de procesamiento de control de teselación 2808 trata los vértices introducidos como puntos de control para un parche geométrico. Los puntos de control se transforman desde una representación introducida desde el parche (por ejemplo, las bases del parche) a una representación que es adecuada para su uso en evaluación de superficie por la unidad de procesamiento de evaluación de teselación 2812. La unidad de procesamiento de control de teselación 2808 también puede calcular factores de teselación para bordes de parches geométricos. Un factor de teselación se aplica a un único borde y cuantifica un nivel dependiente de vista de detalle asociado con el borde. Una unidad de teselación 2810 está configurada para recibir los factores de teselación para bordes de un parche y para teselar el parche en múltiples primitivas geométricas tales como primitivas de línea, de triángulo o cuadrilaterales, que se transmiten a una unidad de procesamiento de evaluación de teselación 2812. La unidad de procesamiento de evaluación de teselación 2812 opera en coordenadas parametrizadas de el parche subdividido para generar una representación de superficie y atributos de vértice para cada vértice asociado con las primitivas geométricas.

Una segunda instancia de un ensamblador de primitivas 2814 recibe atributos de vértice desde la unidad de procesamiento de evaluación de teselación 2812, leyendo atributos de vértice almacenados según se necesite, y construye primitivas geométricas para su procesamiento por la unidad de procesamiento de geometría 2816. La unidad de procesamiento de geometría 2816 es una unidad de ejecución programable que ejecuta programas de sombreador de geometría para transformar primitivas geométricas recibidas desde el ensamblador de primitivas 2814 como se especifica por los programas de sombreador de geometría. En una realización, la unidad de procesamiento de geometría 2816 se programa para subdividir las primitivas geométricas en una o más nuevas primitivas geométricas y calcular parámetros usados para rasterizar las nuevas primitivas geométricas.

En algunas realizaciones, la unidad de procesamiento de geometría 2816 puede añadir o eliminar elementos en el flujo de geometría. La unidad de procesamiento de geometría 2816 emite los parámetros y vértices que especifican nuevas primitivas geométricas al ensamblador de primitivas 2818. El ensamblador de primitivas 2818 recibe los parámetros y vértices desde la unidad de procesamiento de geometría 2816 y construye primitivas geométricas para su procesamiento por una unidad de escalado de ventanilla, selección y recorte 2820. La unidad de procesamiento de

geometría 2816 lee datos que se almacenan en memoria de procesador paralelo o memoria de sistema para su uso en el procesamiento de datos de geometría. La unidad de escalado de ventanilla, selección y recorte 2820 realiza recorte, selección y escalado de ventanilla y emite primitivas geométricas procesadas a un rasterizador 2822.

El rasterizador 2822 puede realizar selección de profundidad y otras optimizaciones basadas en profundidad. El rasterizador 2822 también realiza conversión de exploración en las nuevas primitivas geométricas para generar fragmentos y emitir esos fragmentos y datos de cobertura asociados a la unidad de procesamiento de fragmento/píxel 2824. La unidad de procesamiento de fragmento/píxel 2824 es una unidad de ejecución programable que está configurada para ejecutar programas de sombreador de fragmentos o programas de sombreador de píxeles. La unidad de procesamiento de fragmento/píxel 2824 que transforma fragmentos o píxeles recibidos desde el rasterizador 2822, como se especifica por el fragmento o programas de sombreador de píxeles. Por ejemplo, la unidad de procesamiento de fragmento/píxel 2824 puede programarse para realizar operaciones incluidas, pero sin limitación a, correlación de texturas, sombreado, combinación, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se emiten a una unidad de operaciones de trama 2826. La unidad de procesamiento de fragmento/píxel 2824 puede leer datos que se almacena en o bien la memoria de procesador paralelo o bien la memoria de sistema para su uso cuando procesa los datos de fragmento. Los programas de sombreador de fragmento o píxeles pueden configurarse para sombrear en muestra, píxel, losa u otras granularidades dependiendo de la tasa de muestreo configurada para las unidades de procesamiento.

La unidad de operaciones de trama 2826 es una unidad de procesamiento que realiza operaciones de trama que incluye, pero sin limitación a, patrón, prueba z, combinación y similares, y emite datos de píxel como datos de gráficos procesados que hay que almacenar en memoria gráfica (por ejemplo, memoria de procesador paralelo 2522 como en **la Figura 25**, y/o memoria de sistema 2404 como en **la Figura 24**, que hay que visualizar en el uno o más dispositivo o dispositivos de visualización 2410 o para su procesamiento adicional por uno del uno o más procesador o procesadores 2402 o procesador o procesadores paralelos 2412. En algunas realizaciones, la unidad de operaciones de trama 2826 está configurada para comprimir datos z o color que se escriben en la memoria y descomprimir datos z o de color que se leen de la memoria.

Las realizaciones de la invención proporcionan significantes beneficios sobre los flujos actuales para visualización remota/inalámbrica, que se accionan por software ejecutado en una CPU. Las técnicas descritas en este documento reducen significativamente la participación de la CPU durante la representación de fotograma, proceso de codificación y transmisión, mejorando de este modo el rendimiento.

En las realizaciones, el término "motor" o "módulo" o "lógica" puede referirse a, ser parte de, o incluir un circuito integrado específico de la aplicación (ASIC), un circuito electrónico, un procesador (compartido, especializado o grupo), y/o memoria (compartida, especializada o grupo) 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 las realizaciones, un motor o un módulo puede implementarse en firmware, hardware, software o cualquier combinación de firmware, hardware y software.

Realizaciones de la invención pueden incluir diversas etapas, que se han descrito anteriormente. Las etapas pueden incorporarse en instrucciones ejecutables en máquinas que puede usarse para provocar que un procesador de fin general o fin especial realice las etapas. Como alternativa, estas etapas pueden realizarse por componentes de hardware específicos que contienen lógica por cable para realizar las etapas, o mediante cualquier combinación de componentes informáticos programados y componentes de hardware personalizados.

Como se describe en este documento, las instrucciones pueden referirse a configuraciones de hardware específicas tales como circuitos integrados específicos de la aplicación (ASIC) configurados para realizar ciertas operaciones o que tienen una funcionalidad predeterminada o instrucciones de software almacenadas en memoria incorporada en un medio legible por ordenador no transitorio. Por lo tanto, las técnicas mostradas en las figuras pueden implementarse usando código y datos almacenados y ejecutarse en uno o más dispositivos electrónicos (por ejemplo, una estación de extremo, un elemento de red, etc.). Tales dispositivos electrónicos almacenan y comunican (internamente y/o con otros dispositivos electrónicos a través de una red) código y datos usando medios informáticos legibles por máquina, tales como medios informáticos de almacenamiento legible por máquina no transitorios (por ejemplo, discos magnéticos; discos ópticos; memoria de acceso aleatorio; memoria de solo lectura; dispositivos de memoria flash; memoria de cambio de fase) y medios informáticos de comunicación legibles por máquina transitorios (por ejemplo, eléctrica, óptica, acústica u otra forma de señales propagadas, tales como ondas portadoras, señales infrarrojas, señales digitales, etc.).

Además, tales dispositivos electrónicos habitualmente incluyen un conjunto de uno o más procesadores acoplados a uno o más otros componentes, tales como uno o más dispositivos de almacenamiento (medios de almacenamiento legible por máquina no transitorio), dispositivos de entrada/salida de usuario (por ejemplo, un teclado, una pantalla táctil y/o un visualizador) y conexiones de red. El acoplamiento del conjunto de procesadores y otros componentes habitualmente es a través de uno o más buses y puentes (también denominados controladores de bus). El dispositivo de almacenamiento y 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. Por lo tanto, el dispositivo de

almacenamiento de un dispositivo electrónico dado habitualmente 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 pueden implementarse usando diferentes combinaciones de software, firmware y/o hardware. A lo largo de toda esta descripción detallada, para los propósitos de explicación, se expusieron numerosos detalles específicos para proporcionar un entendimiento completo de la presente invención. Será evidente, sin embargo, para un experto en la materia que la invención puede ponerse en práctica sin algunos de estos detalles específicos. En ciertos casos, estructuras y funciones bien conocidas no se describieron con detalles elaborados para evitar obstaculizar la materia objeto de la presente invención. Por consiguiente, el alcance de la invención se define por las reivindicaciones independientes adjuntas. Las reivindicaciones dependientes definen realizaciones preferidas.

Page Break

REIVINDICACIONES

1. Un aparato que comprende:
una unidad de procesamiento gráfico (800) que incluye un motor de visualización (2211, 840) para representar una secuencia de imágenes de vídeo a una memoria intermedia de fotogramas (2231);
un codificador (2212) para comprimir la secuencia de imágenes de vídeo para generar una secuencia de imágenes de vídeo comprimidas a una memoria intermedia de flujo comprimido (2232);
un controlador de interfaz de red (2213) para transmitir las imágenes de vídeo comprimidas a través de un enlace de red a un visualizador remoto (2214);
una pluralidad de registros de puntero de memoria intermedia (2221, 2222, 2223) para almacenar punteros de lectura y punteros de escritura que identifican ubicaciones de lectura y ubicaciones de escritura, respectivamente, en una memoria intermedia de fotogramas (2231) y la memoria intermedia de flujo comprimido (2232);
una unidad de procesamiento central, CPU (2204), para inicializar los punteros de lectura y punteros de escritura para procesar una o más de las imágenes de vídeo; y
el motor de visualización (2211) para acceder a un primer puntero de escritura para escribir en una ubicación especificada en la memoria intermedia de fotogramas (2231), el codificador (2212) para comenzar a leer de la memoria intermedia de fotogramas (2231) basándose en un primer valor de puntero de lectura, el codificador (2212) para escribir en la memoria intermedia de flujo comprimido (2232) basándose en un segundo valor de puntero de escritura, y el controlador de interfaz de red (2213) para leer de la memoria intermedia de flujo comprimido (2232) basándose en un segundo valor de puntero de lectura, el primer y segundo valores de puntero de escritura y lectura que hay que actualizar sin intervención de la CPU (2204) a medida que el motor de visualización (2211) escribe en la memoria intermedia de fotogramas (2231), el codificador (2212) lee de la memoria intermedia de fotogramas (2231) y escribe en la memoria intermedia de flujo comprimido (2232), y el controlador de interfaz de red (2213) lee de la memoria intermedia de flujo comprimido (2232);
caracterizado por que el motor de visualización (2211) es para transmitir una señal de notificación al codificador (2212) cuando el primer puntero de escritura alcanza un macrobloque, el codificador (2212) para, en respuesta, comenzar a leer de la memoria intermedia de fotogramas (2231) en el primer valor de puntero de lectura.

2. El aparato de visualización remoto de acuerdo con la reivindicación 1, en donde el codificador (2212) es para transmitir una señal de notificación al controlador de interfaz de red (2213) cuando el segundo puntero de escritura alcanza un umbral especificado, el controlador de interfaz de red (2213) para, en respuesta, comenzar a leer de la memoria intermedia de fotogramas (2231) en el segundo valor de puntero de lectura.

3. El aparato de visualización remoto de acuerdo con la reivindicación 1 o 2, en donde el controlador de interfaz de red (2213) es para transmitir las imágenes de vídeo comprimidas a través de un enlace inalámbrico al visualizador remoto (2214).

4. El aparato de visualización remoto de acuerdo con la reivindicación 3, en donde el enlace inalámbrico comprende un enlace WiDi.

5. El aparato de visualización remoto de acuerdo con la reivindicación 1 o 2, que comprende adicionalmente:
circuitería para ejecutar hilos de una o más máquinas virtuales, los hilos para provocar que se ejecuten comandos gráficos, provocando de esta manera que el motor de visualización (2211) represente la secuencia de imágenes de vídeo.

6. El aparato de visualización remoto de acuerdo con la reivindicación 1 o 5 en donde los registros de puntero de memoria intermedia comprenden registros de fin general.

7. Un método que comprende:
representar, por un motor de visualización (2211, 849) incluido en una unidad de procesamiento gráfico (800), una secuencia de imágenes de vídeo a una memoria intermedia de fotogramas (2231);
comprimir la secuencia de imágenes de vídeo a una memoria intermedia de flujo comprimido (2232) para generar una secuencia de imágenes de vídeo comprimidas;
transmitir las imágenes de vídeo comprimidas a través de un enlace de red a un visualizador remoto (2214);
almacenar punteros de lectura y punteros de escritura en una pluralidad de registros de puntero de memoria intermedia (2221, 2222, 2223), identificando los punteros de lectura y punteros de escritura ubicaciones de lectura y ubicaciones de escritura, respectivamente, en una memoria intermedia de fotogramas (2231) y una memoria intermedia de flujo comprimido (2232);
inicializar los punteros de lectura y punteros de escritura para procesar una o más de las imágenes de vídeo por una CPU (2204);

acceder a un primer puntero de escritura por un motor de visualización (840, 2211) de una unidad de procesamiento gráfico (800) para escribir en una ubicación especificada en la memoria intermedia de fotogramas (2231);
leer de la memoria intermedia de fotogramas (2231) por un codificador (2212) basándose en un primer valor de puntero de lectura;
escribir por el codificador (2212) en la memoria intermedia de flujo comprimido (2232) basándose en un segundo valor de puntero de escritura;
leer de la memoria intermedia de flujo comprimido (2232) por un controlador de interfaz de red (2213) basándose en un segundo valor de puntero de lectura, el primer y segundo valores de puntero de escritura y lectura que hay que actualizar sin intervención de la CPU (2204) cuando el motor de visualización (2211) escribe en la memoria intermedia de fotogramas (2231), el codificador (2212) lee de la memoria intermedia de fotogramas (2231) y escribe en la memoria intermedia de flujo comprimido (2232), y el controlador de interfaz de red (2213) lee de la memoria intermedia de flujo comprimido (2232);
caracterizado por
transmitir una señal de notificación desde el motor de visualización (2211) al codificador cuando el primer puntero de escritura alcanza un macrobloque, el codificador (2212) para, en respuesta, comenzar a leer de la memoria intermedia de fotogramas (2231) en el primer valor de puntero de lectura.

8. El método de acuerdo con la reivindicación 7, que comprende adicionalmente:

transmitir una señal de notificación desde el codificador (2212) al controlador de interfaz de red (2213) cuando el segundo puntero de escritura alcanza un umbral especificado, el controlador de interfaz de red (2213) para, en respuesta, comenzar a leer de la memoria intermedia de fotogramas (2231) en el segundo valor de puntero de lectura.

9. El método de acuerdo con la reivindicación 7 u 8, en donde el controlador de interfaz de red (2213) es para transmitir las imágenes de vídeo comprimidas a través de un enlace inalámbrico al visualizador remoto (2214).

10. El método de acuerdo con la reivindicación 9, en donde el enlace inalámbrico comprende un enlace WiDi.

11. El método de acuerdo con la reivindicación 7, que comprende adicionalmente:
ejecutar hilos de una o más máquinas virtuales, los hilos para provocar que se ejecuten comandos gráficos, provocando de esta manera que el motor de visualización (2211) represente la secuencia de imágenes de vídeo.

12. El método de acuerdo con la reivindicación 7 u 8, en donde los registros de puntero de memoria intermedia (2221, 2222, 2223) comprenden registros de fin general.

Page Break

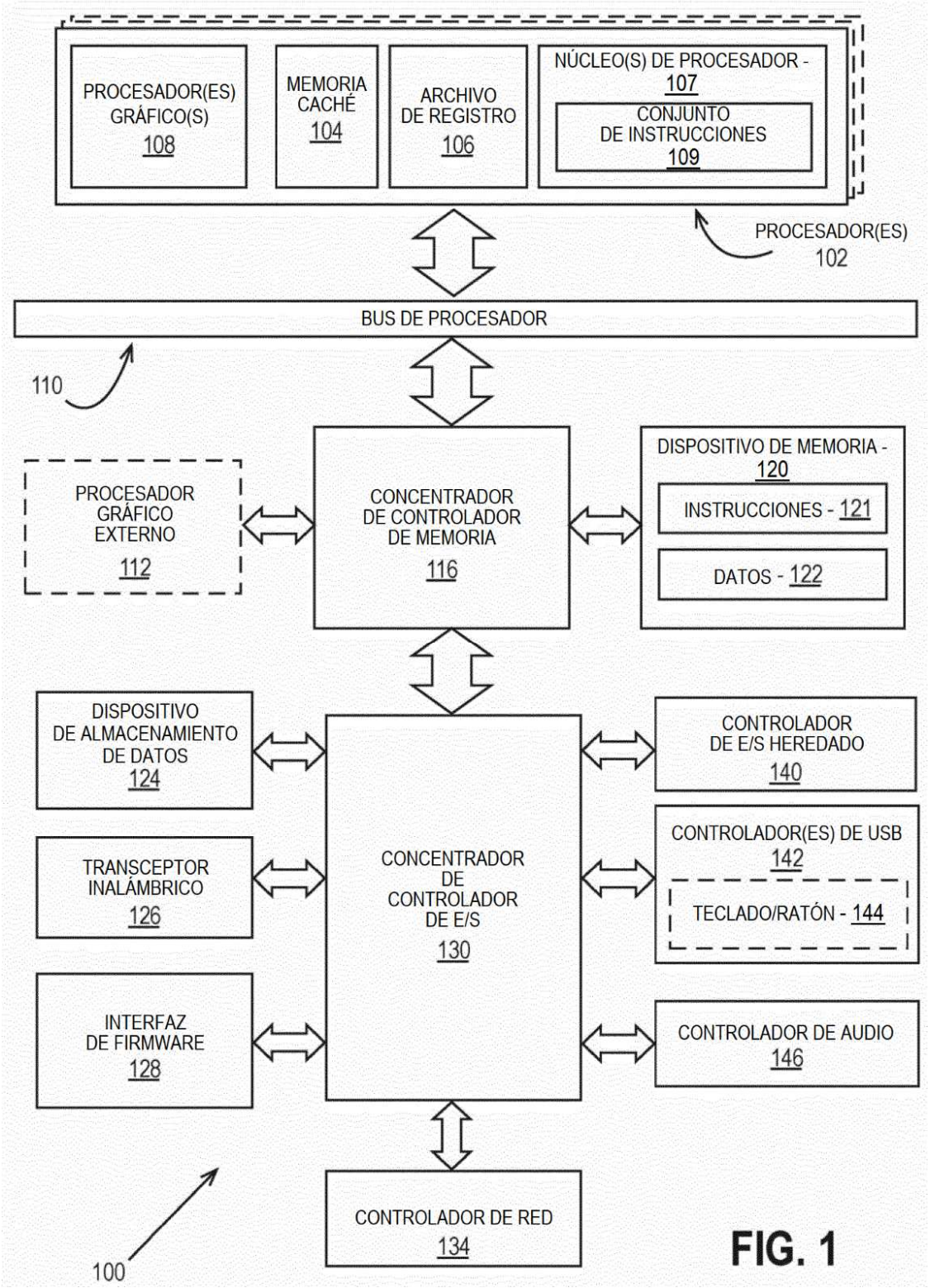

PROCESADOR(ES) GRÁFICO(S) 108
MEMORIA CACHÉ 104
ARCHIVO DE REGISTRO 106
NÚCLEO(S) DE PROCESADOR - 107
CONJUNTO DE INSTRUCCIONES 109
PROCESADOR(ES) 102
BUS DE PROCESADOR
110
PROCESADOR GRÁFICO EXTERNO 112
CONCENTRADOR DE CONTROLADOR DE MEMORIA 116
DISPOSITIVO DE MEMORIA - 120
INSTRUCCIONES - 121
DATOS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

FIG. 1

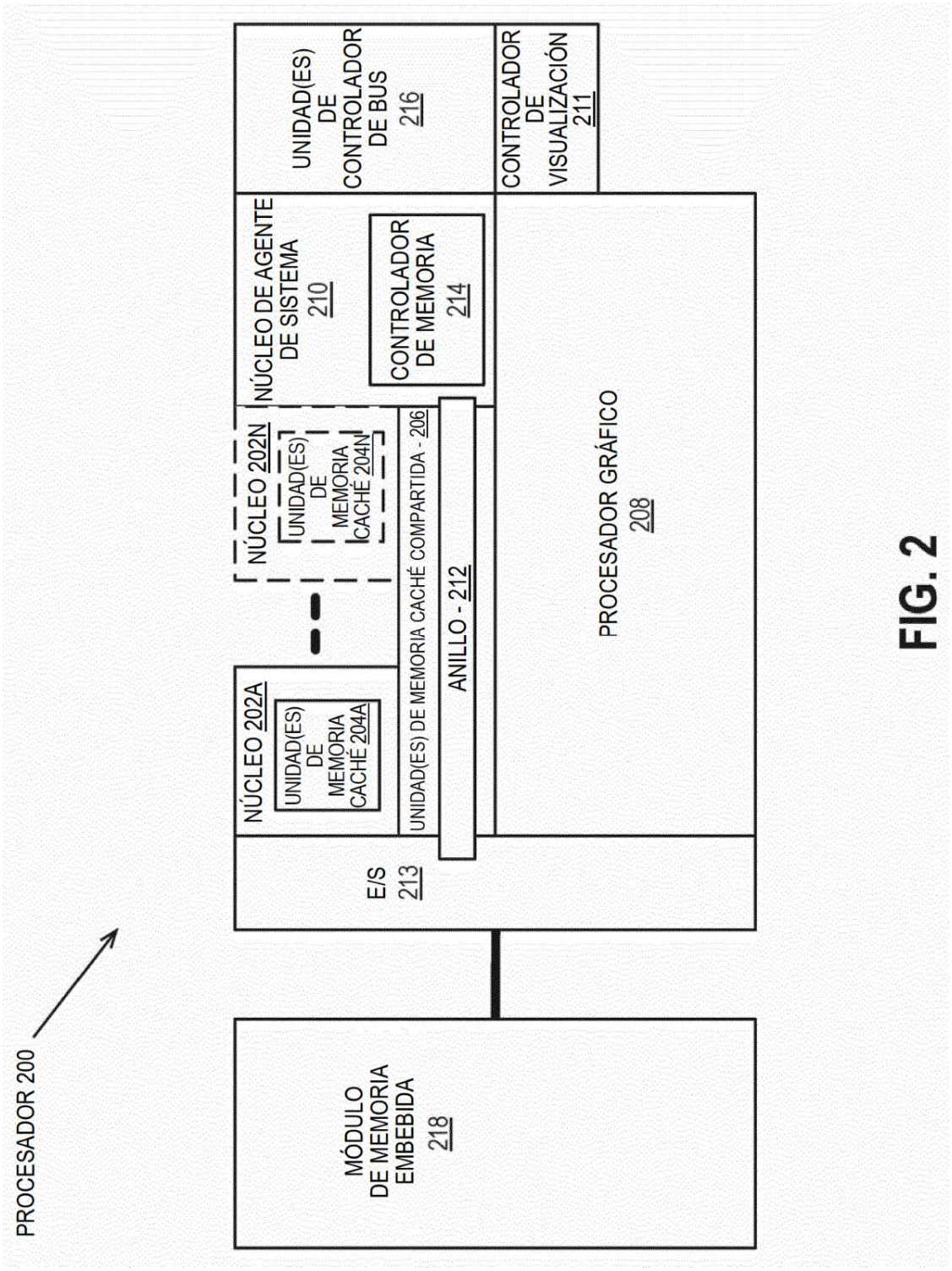

PROCESADOR 200
MÓDULO DE MEMORIA EMBEBIDA 218
E/S 213
NÚCLEO 202A
UNIDAD(ES) DE MEMORIA CACHÉ 204A
NÚCLEO 202N
UNIDAD(ES) DE MEMORIA CACHÉ 204N
UNIDAD(ES) DE MEMORIA CACHÉ COMPARTIDA - 206
ANILLO - 212
NÚCLEO DE AGENTE DE SISTEMA 210
CONTROLADOR DE MEMORIA 214
UNIDAD(ES) DE CONTROLADOR DE BUS 216
PROCESADOR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

FIG. 2

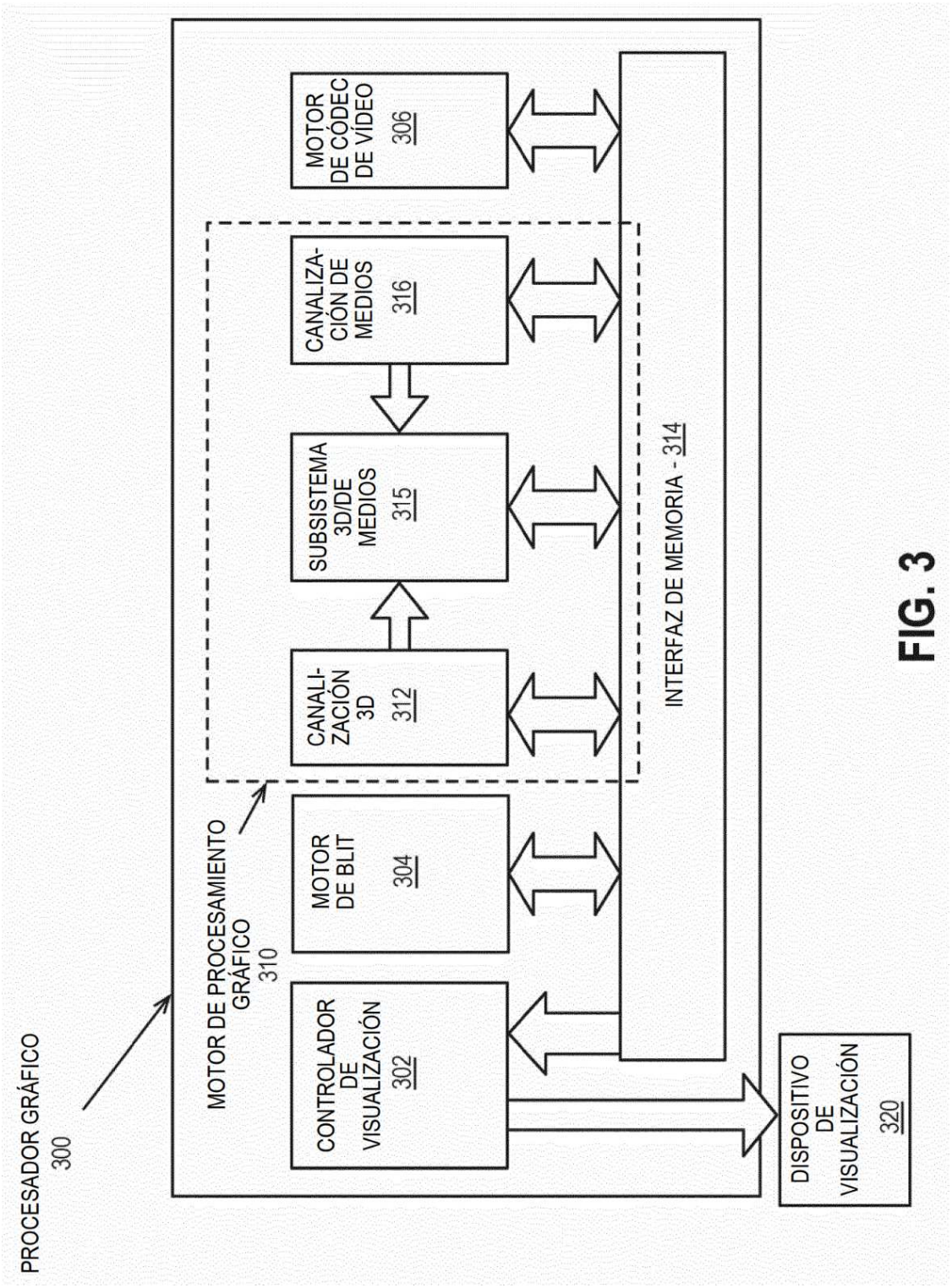

PROCESADOR GRÁFICO
300
MOTOR DE PROCESAMIENTO GRÁFICO
310
CONTROLADOR DE VISUALIZACIÓN
302
MOTOR DE BLIT
304
CANALI-ZACIÓN 3D
312
SUBSISTEMA 3D/DE MEDIOS
315
CANALIZA-CIÓN DE MEDIOS
316
MOTOR DE CÓDEC DE VÍDEO
306
INTERFAZ DE MEMORIA - 314
DISPOSITIVO DE VISUALIZACIÓN
320

FIG. 3

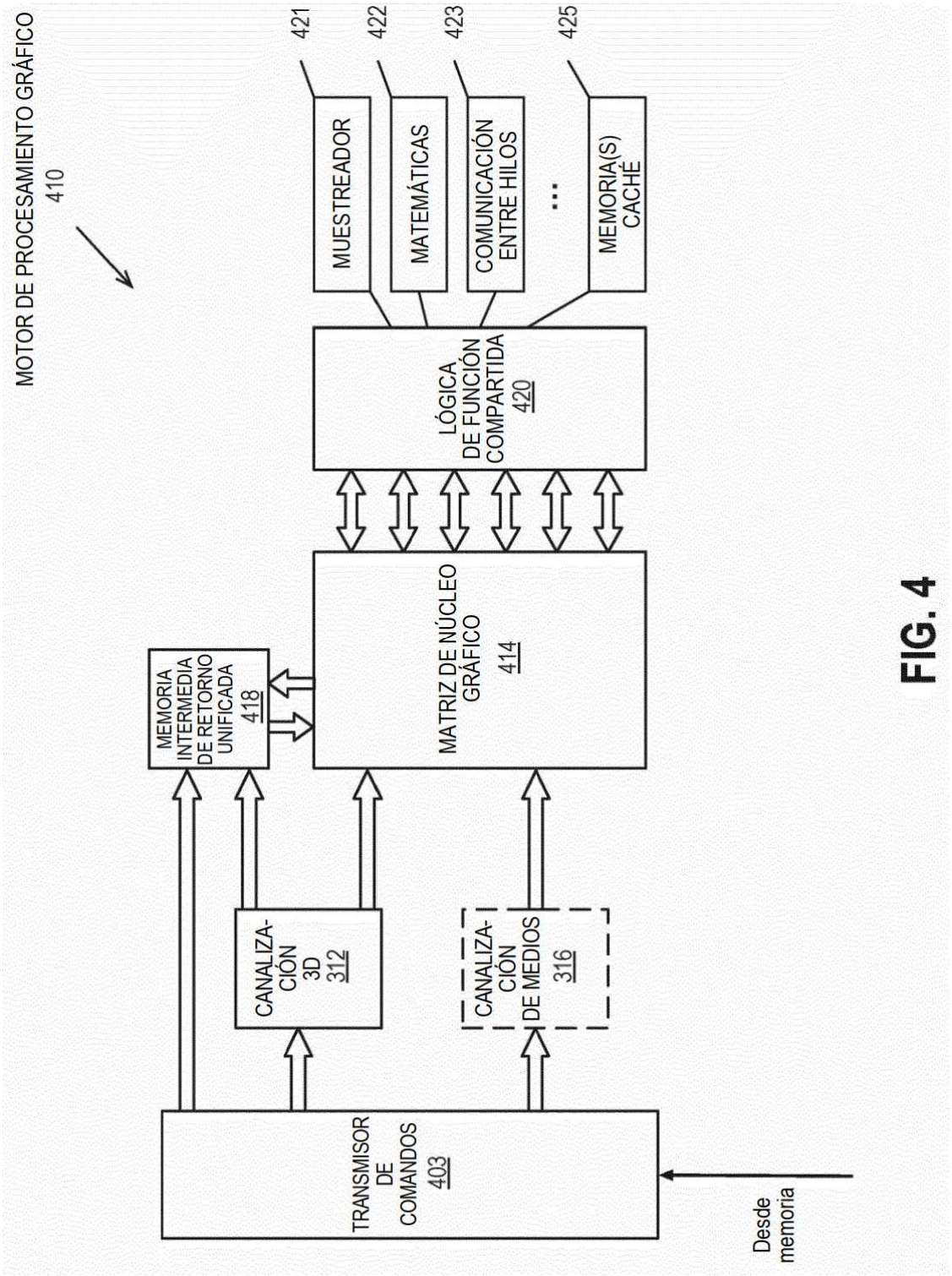

MOTOR DE PROCESAMIENTO GRÁFICO
410
421
422
423
425
MUESTREADOR
MATEMÁTICAS
COMUNICACIÓN ENTRE HILOS
...
MEMORIA(S) CACHÉ
LÓGICA DE FUNCIÓN COMPARTIDA
420
MATRIZ DE NÚCLEO GRÁFICO
414
MEMORIA INTERMEDIA DE RETORNO UNIFICADA
418
CANALIZA-CIÓN 3D
312
CANALIZA-CIÓN DE MEDIOS
316
TRANSMISOR DE COMANDOS
403
Desde memoria

FIG. 4

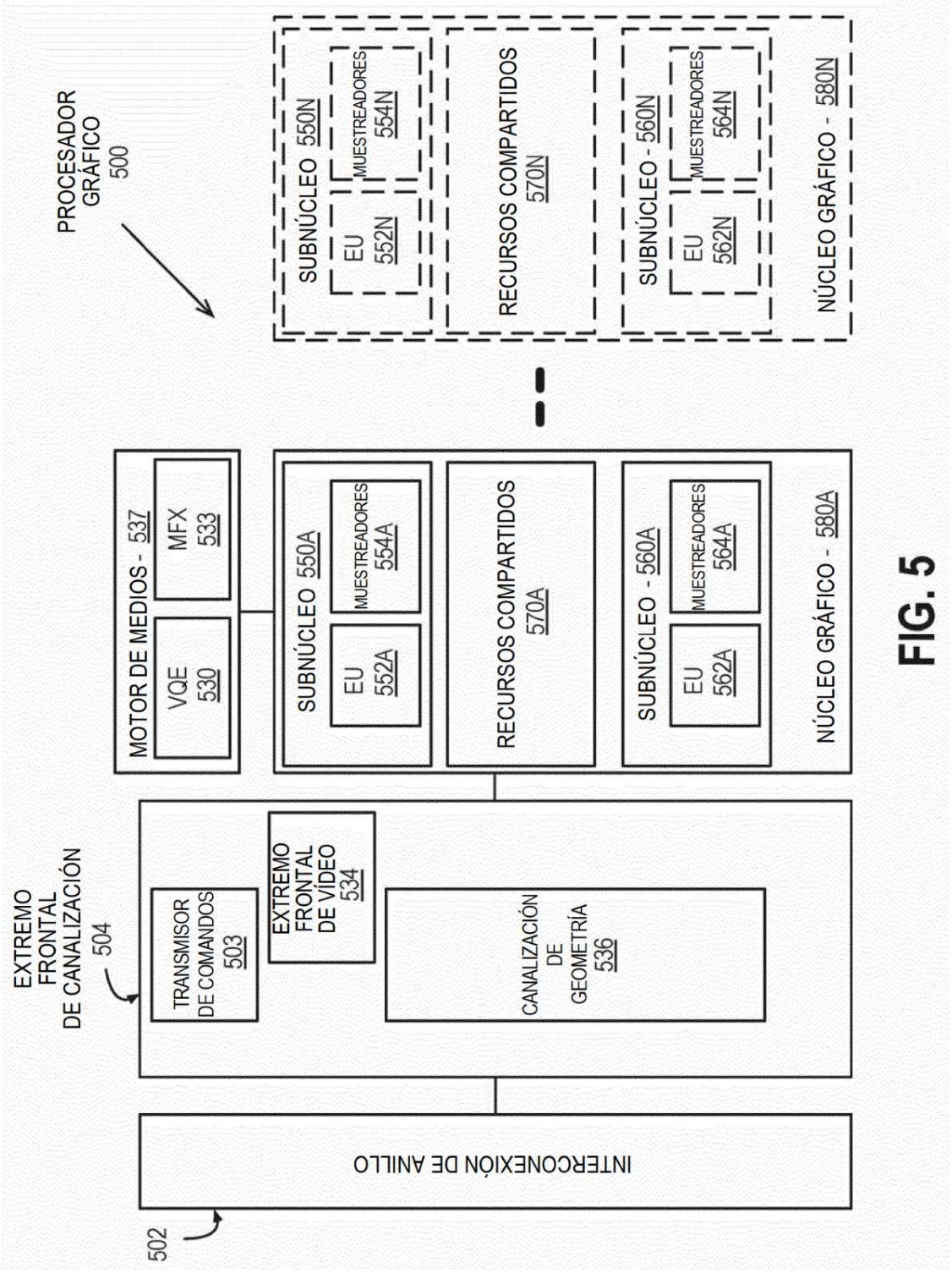

PROCESADOR GRÁFICO 500
EXTREMO FRONTAL DE CANALIZACIÓN 504
TRANSMISOR DE COMANDOS 503
EXTREMO FRONTAL DE VÍDEO 534
CANALIZACIÓN DE GEOMETRÍA 536
MOTOR DE MEDIOS - 537
VQE 530
MFX 533
SUBNÚCLEO 550A
EU 552A
MUESTREADORES 554A
RECURSOS COMPARTIDOS 570A
SUBNÚCLEO - 560A
EU 562A
MUESTREADORES 564A
NÚCLEO GRÁFICO - 580A
SUBNÚCLEO 550N
EU 552N
MUESTREADORES 554N
RECURSOS COMPARTIDOS 570N
SUBNÚCLEO - 560N
EU 562N
MUESTREADORES 564N
NÚCLEO GRÁFICO - 580N
INTERCONEXIÓN DE ANILLO
502

FIG. 5

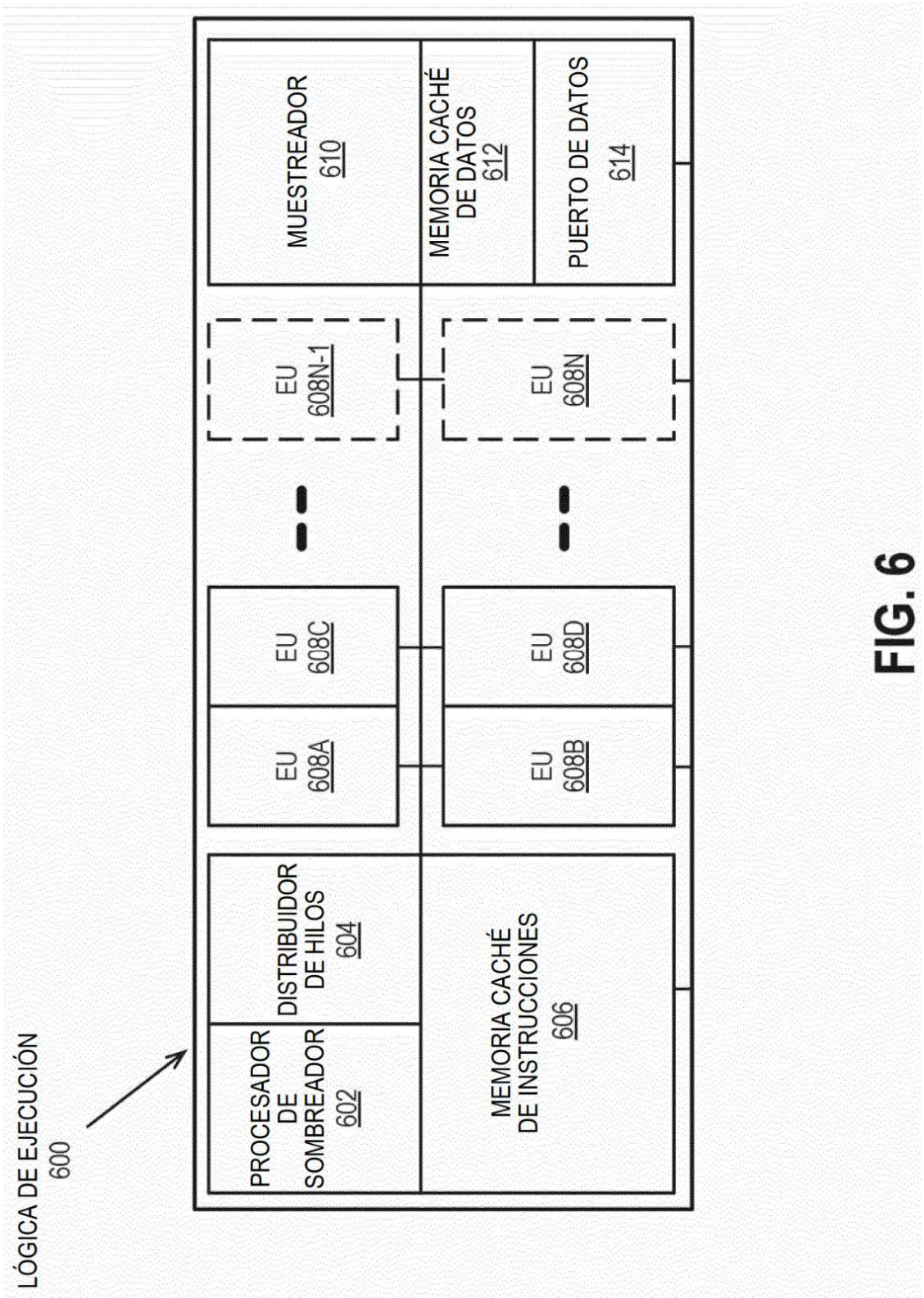

LÓGICA DE EJECUCIÓN
600
PROCESADOR DE SOMBREADOR
602
DISTRIBUIDOR DE HILOS
604
MEMORIA CACHÉ DE INSTRUCCIONES
606
EU 608A
EU 608B
EU 608C
EU 608D
EU 608N-1
EU 608N
MUESTREADOR
610
MEMORIA CACHÉ DE DATOS
612
PUERTO DE DATOS
614

FIG. 6

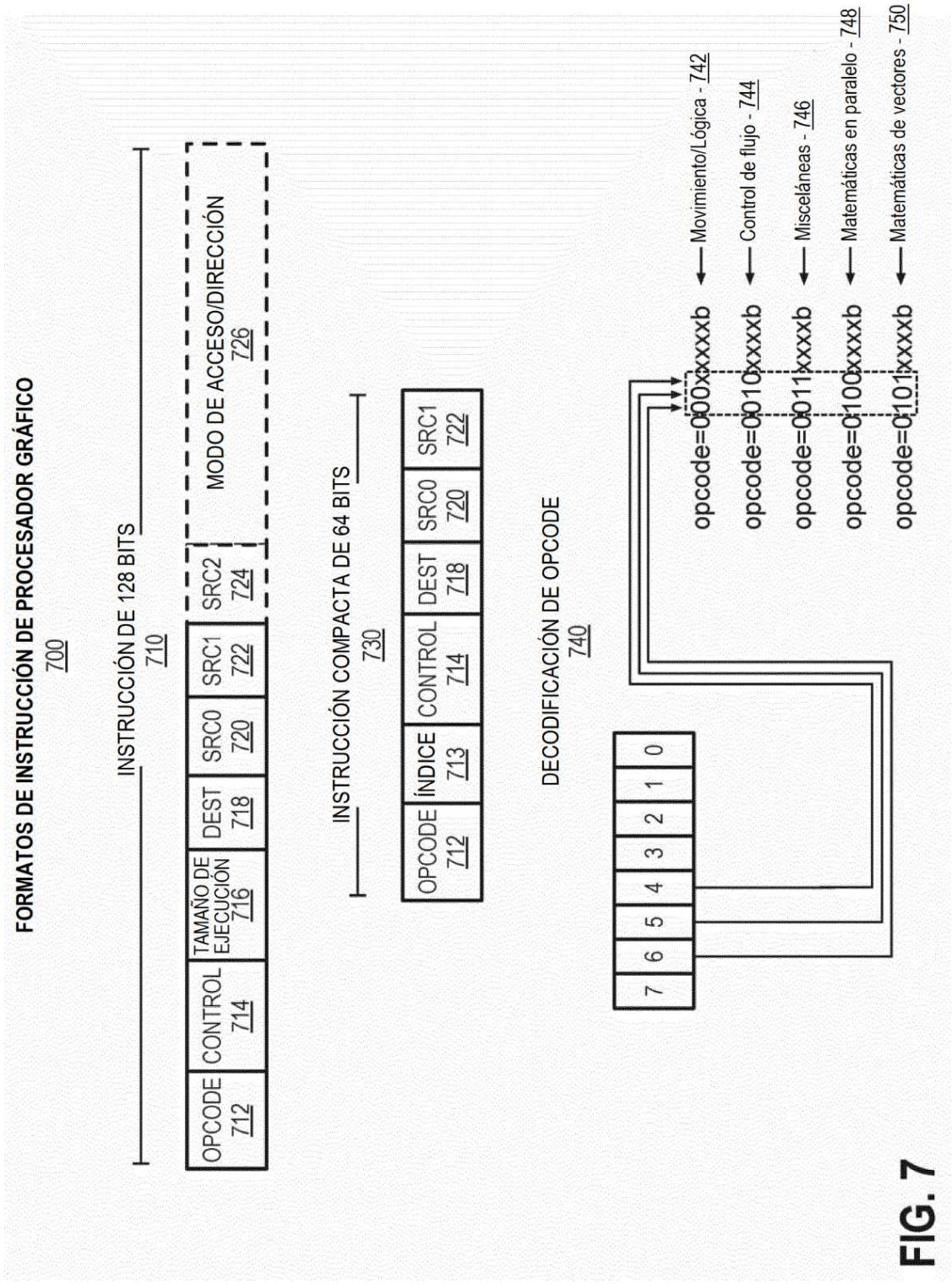

FORMATOS DE INSTRUCCIÓN DE PROCESADOR GRÁFICO
700
INSTRUCCIÓN DE 128 BITS
710
OPCODE 712
CONTROL 714
TAMAÑO DE EJECUCIÓN 716
DEST 718
SRC0 720
SRC1 722
SRC2 724
MODO DE ACCESO/DIRECCIÓN 726
INSTRUCCIÓN COMPACTA DE 64 BITS
730
OPCODE 712
ÍNDICE 713
CONTROL 714
DEST 718
SRC0 720
SRC1 722
DECODIFICACIÓN DE OPCODE
740
7 6 5 4 3 2 1 0
opcode=000xxxxb
opcode=001xxxxb
opcode=010xxxxb
opcode=011xxxxb
opcode=100xxxxb
opcode=101xxxxb
Movimiento/Lógica - 742
Control de flujo - 744
Misceláneas - 746
Matemáticas en paralelo - 748
Matemáticas de vectores - 750

FIG. 7

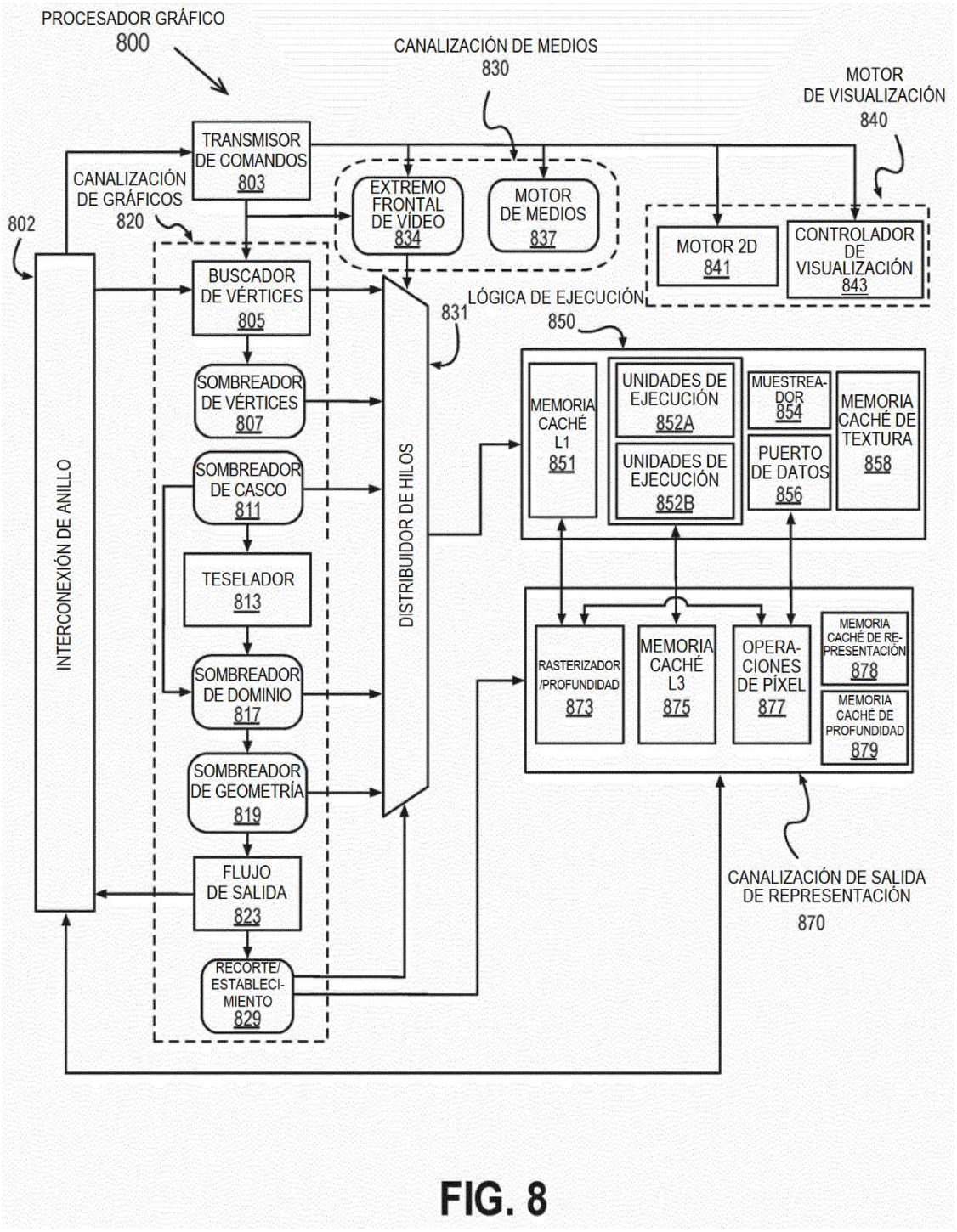

PROCESADOR GRÁFICO
800
CANALIZACIÓN DE MEDIOS
830
MOTOR DE VISUALIZACIÓN
840
TRANSMISOR DE COMANDOS
803
CANALIZACIÓN DE GRÁFICOS
820
802
EXTREMO FRONTAL DE VÍDEO
834
MOTOR DE MEDIOS
837
MOTOR 2D
841
CONTROLADOR DE VISUALIZACIÓN
843
BUSCADOR DE VÉRTICES
805
LÓGICA DE EJECUCIÓN
850
831
SOMBREADOR DE VÉRTICES
807
SOMBREADOR DE CASCO
811
TESELADOR
813
SOMBREADOR DE DOMINIO
817
SOMBREADOR DE GEOMETRÍA
819
FLUJO DE SALIDA
823
RECORTE/ ESTABLECI-MIENTO
829
INTERCONEXIÓN DE ANILLO
DISTRIBUIDOR DE HILOS
MEMORIA CACHÉ L1
851
UNIDADES DE EJECUCIÓN
852A
UNIDADES DE EJECUCIÓN
852B
MUESTREA-DOR
854
PUERTO DE DATOS
856
MEMORIA CACHÉ DE TEXTURA
858
RASTERIZADOR /PROFUNDIDAD
873
MEMORIA CACHÉ L3
875
OPERA-CIONES DE PÍXEL
877
MEMORIA CACHÉ DE RE-PRESENTACIÓN
878
MEMORIA CACHÉ DE PROFUNDIDAD
879
CANALIZACIÓN DE SALIDA DE REPRESENTACIÓN
870

FIG. 8

FIG. 9A

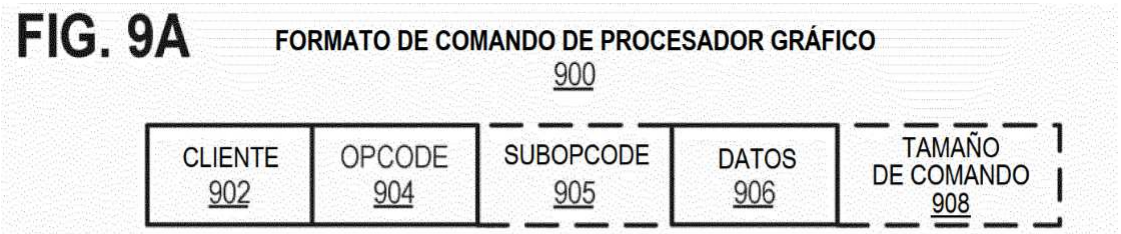
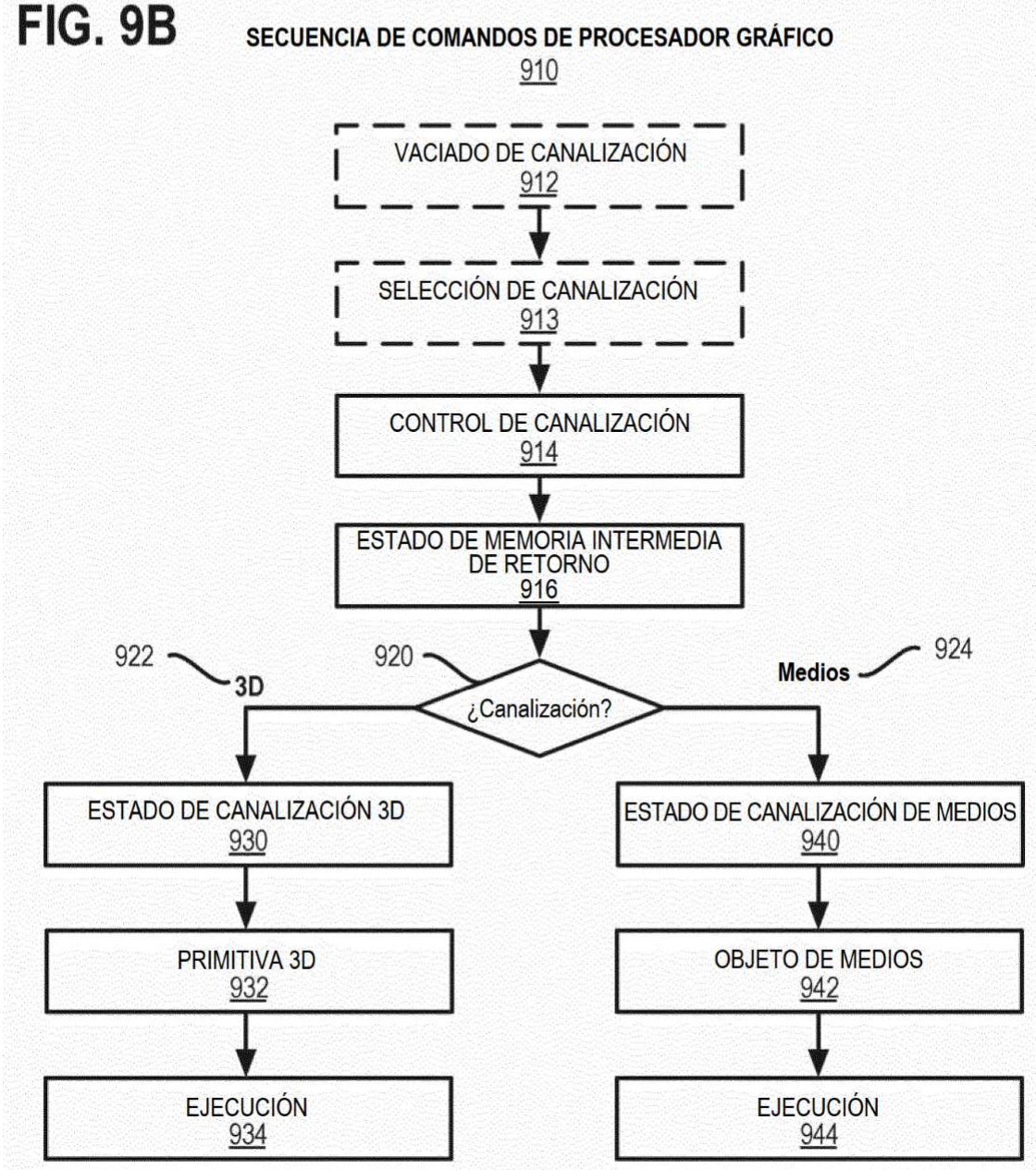

FORMATO DE COMANDO DE PROCESADOR GRÁFICO
900
CLIENTE
902
OPCODE
904
SUBOPCODE
905
DATOS
906
TAMAÑO DE COMANDO
908

FIG. 9B


SECUENCIA DE COMANDOS DE PROCESADOR GRÁFICO
910
VACIADO DE CANALIZACIÓN
912
SELECCIÓN DE CANALIZACIÓN
913
CONTROL DE CANALIZACIÓN
914
ESTADO DE MEMORIA INTERMEDIA DE RETORNO
916
922
3D
920
¿Canalización?
Medios
924
ESTADO DE CANALIZACIÓN 3D
930
PRIMITIVA 3D
932
EJECUCIÓN
934
ESTADO DE CANALIZACIÓN DE MEDIOS
940
OBJETO DE MEDIOS
942
EJECUCIÓN
944

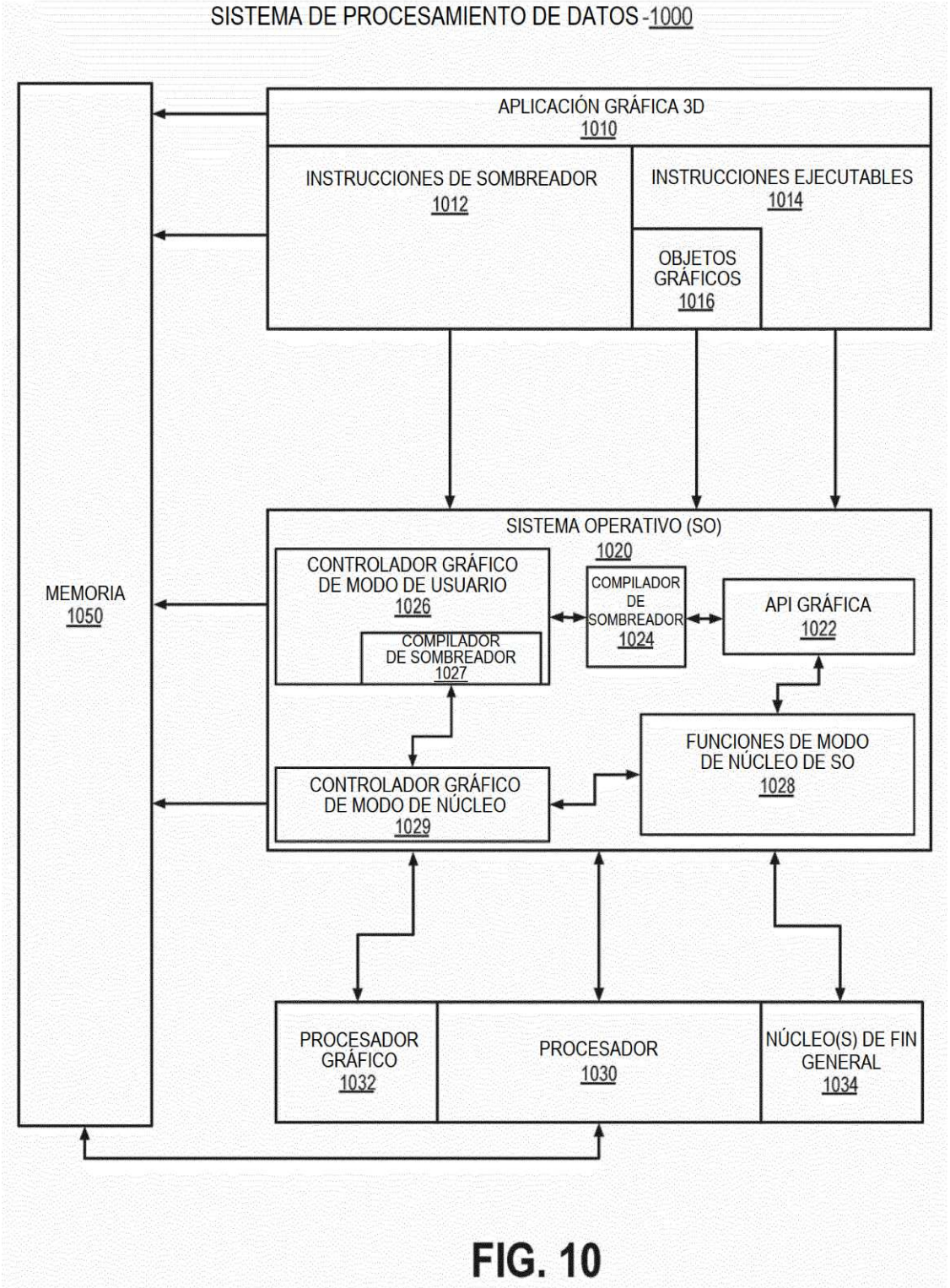

SISTEMA DE PROCESAMIENTO DE DATOS -1000
APLICACIÓN GRÁFICA 3D
1010
INSTRUCCIONES DE SOMBREADOR
1012
INSTRUCCIONES EJECUTABLES
1014
OBJETOS GRÁFICOS
1016
MEMORIA
1050
SISTEMA OPERATIVO (SO)
1020
CONTROLADOR GRÁFICO DE MODO DE USUARIO
1026
COMPILADOR DE SOMBREADOR
1027
COMPILADOR DE SOMBREADOR
1024
API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

FIG. 10

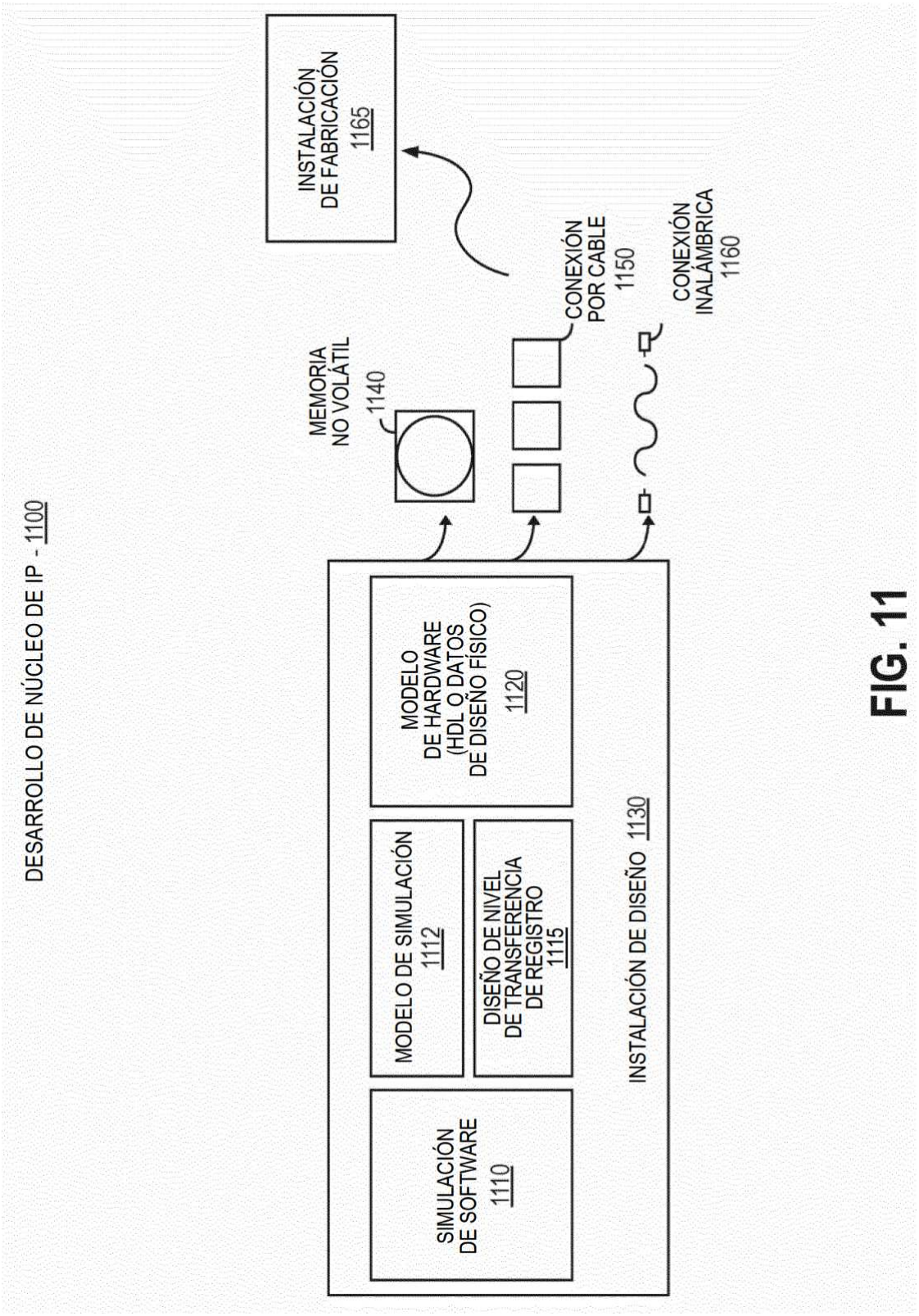

DESARROLLO DE NÚCLEO DE IP - 1100
INSTALACIÓN DE DISEÑO 1130
SIMULACIÓN DE SOFTWARE 1110
MODELO DE SIMULACIÓN 1112
DISEÑO DE NIVEL DE TRANSFERENCIA DE REGISTRO 1115
MODELO DE HARDWARE (HDL O DATOS DE DISEÑO FÍSICO) 1120
MEMORIA NO VOLÁTIL 1140
CONEXIÓN POR CABLE 1150
CONEXIÓN INALÁMBRICA 1160
INSTALACIÓN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

FIG. 11

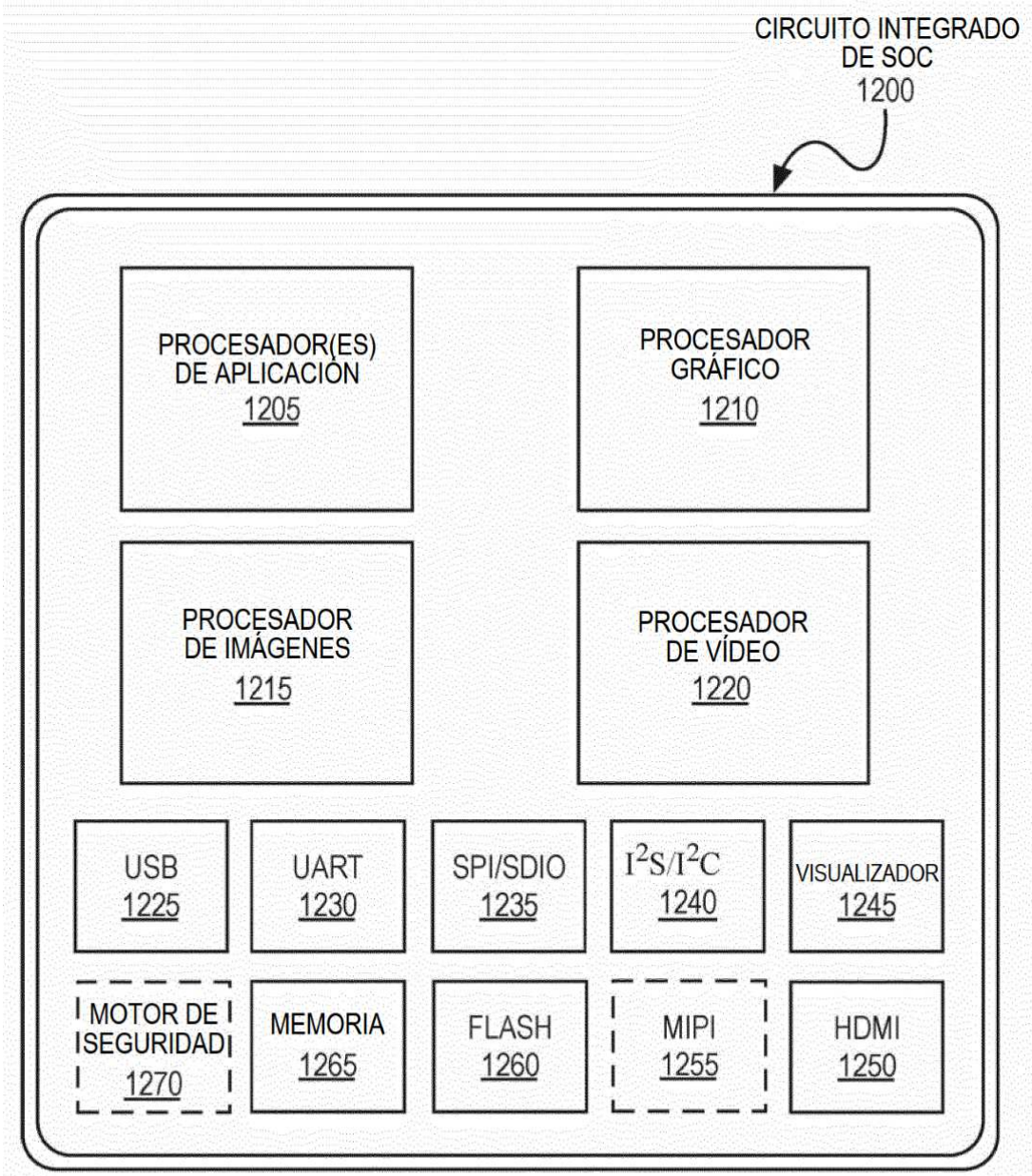

CIRCUITO INTEGRADO DE SOC
1200
PROCESADOR(ES) DE APLICACIÓN
1205
PROCESADOR GRÁFICO
1210
PROCESADOR DE IMÁGENES
1215
PROCESADOR DE VÍDEO
1220
USB
1225
UART
1230
SPI/SDIO
1235
I2S/I2C
1240
VISUALIZADOR
1245
MOTOR DE SEGURIDAD
1270
MEMORIA
1265
FLASH
1260
MIPI
1255
HDMI
1250

FIG. 12

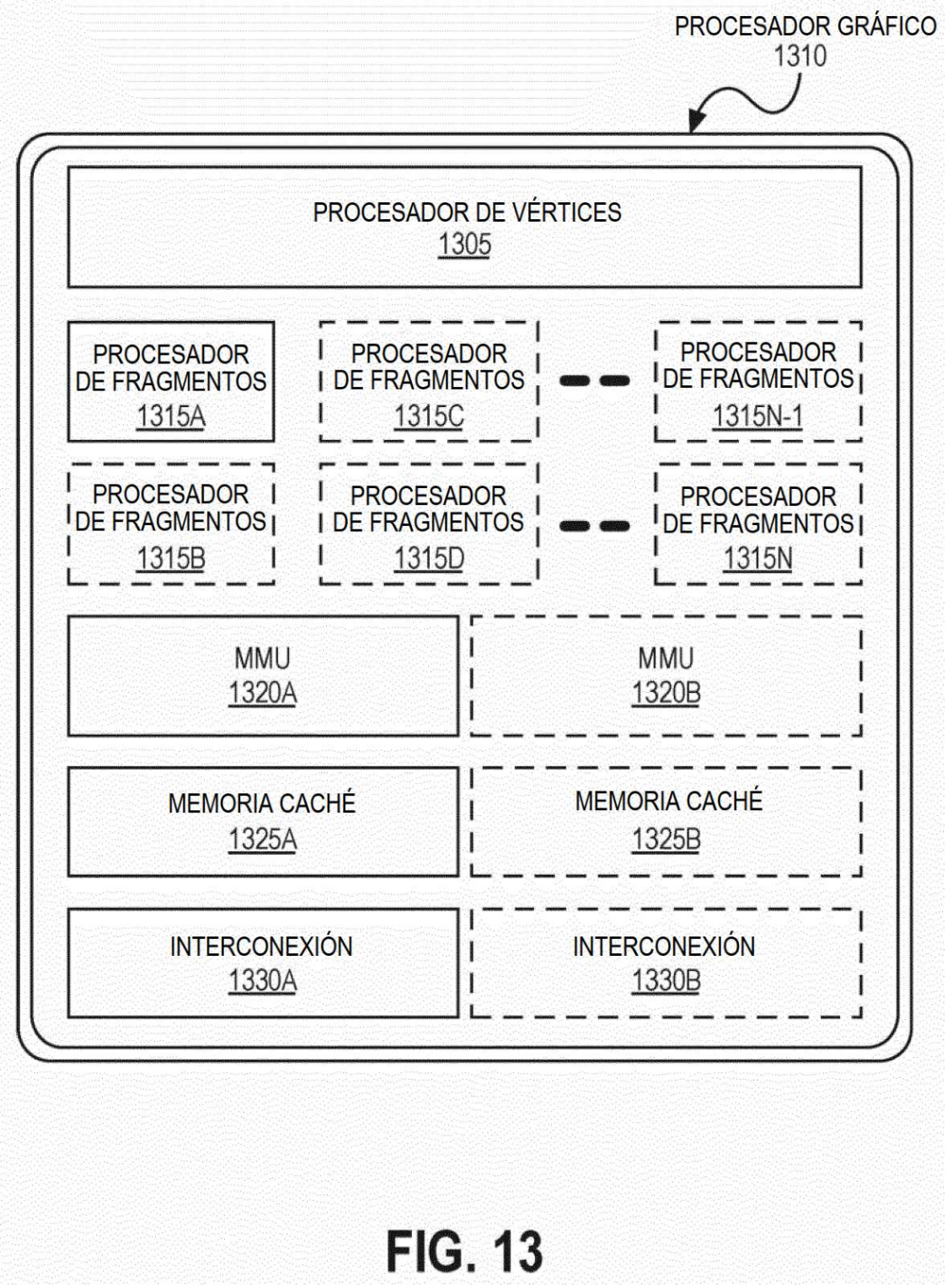

PROCESADOR GRÁFICO
1310
PROCESADOR DE VÉRTICES
1305
PROCESADOR DE FRAGMENTOS
1315A
PROCESADOR DE FRAGMENTOS
1315C
PROCESADOR DE FRAGMENTOS
1315N-1
PROCESADOR DE FRAGMENTOS
1315B
PROCESADOR DE FRAGMENTOS
1315D
PROCESADOR DE FRAGMENTOS
1315N
MMU
1320A
MMU
1320B
MEMORIA CACHÉ
1325A
MEMORIA CACHÉ
1325B
INTERCONEXIÓN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

FIG. 13

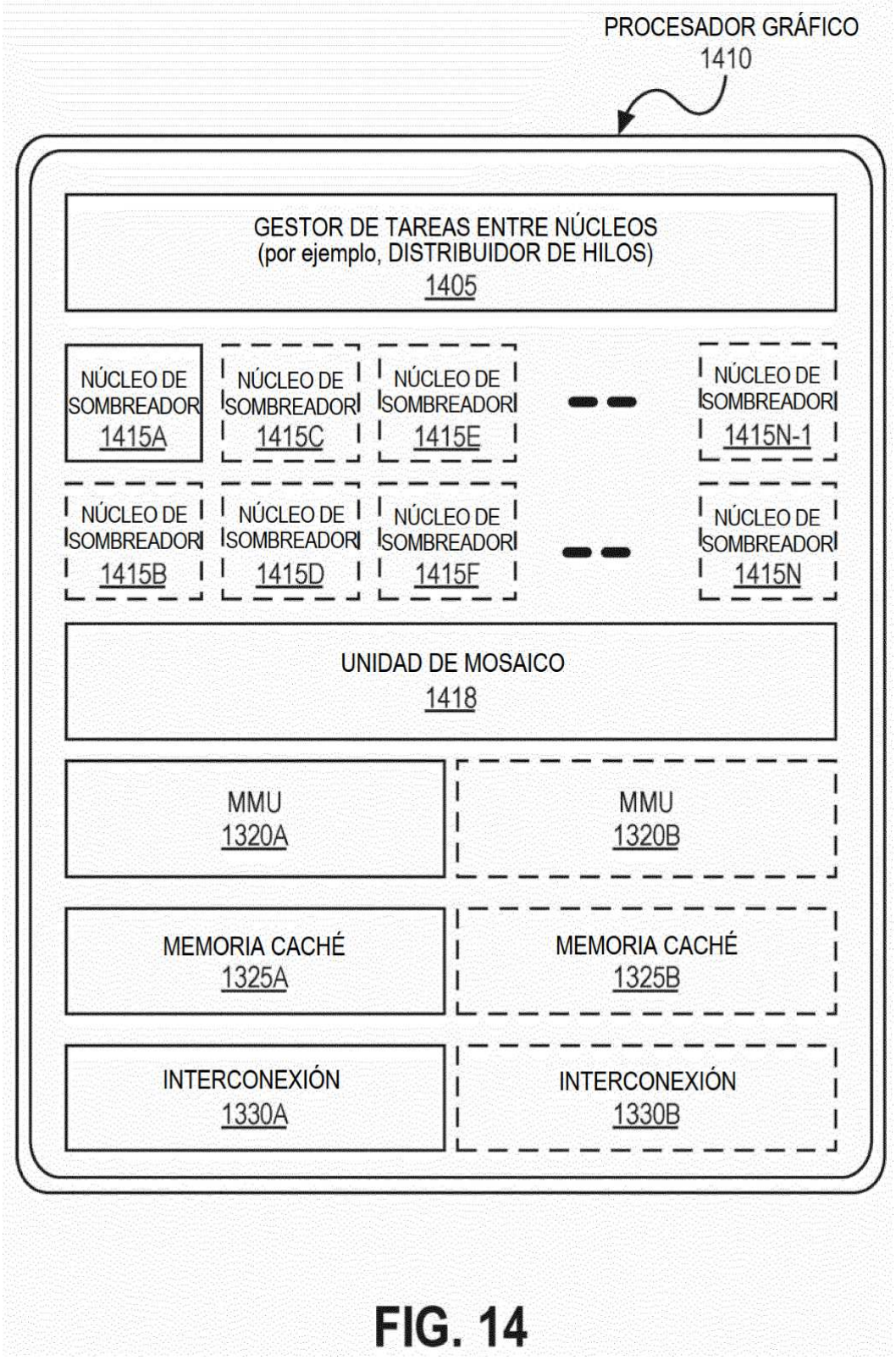

PROCESADOR GRÁFICO
1410
GESTOR DE TAREAS ENTRE NÚCLEOS
(por ejemplo, DISTRIBUIDOR DE HILOS)
1405
NÚCLEO DE SOMBREADOR
1415A
NÚCLEO DE SOMBREADOR
1415C
NÚCLEO DE SOMBREADOR
1415E
NÚCLEO DE SOMBREADOR
1415N-1
NÚCLEO DE SOMBREADOR
1415B
NÚCLEO DE SOMBREADOR
1415D
NÚCLEO DE SOMBREADOR
1415F
NÚCLEO DE SOMBREADOR
1415N
UNIDAD DE MOSAICO
1418
MMU
1320A
MMU
1320B
MEMORIA CACHÉ
1325A
MEMORIA CACHÉ
1325B
INTERCONEXIÓN
1330A
INTERCONEXIÓN
1330B

FIG. 14

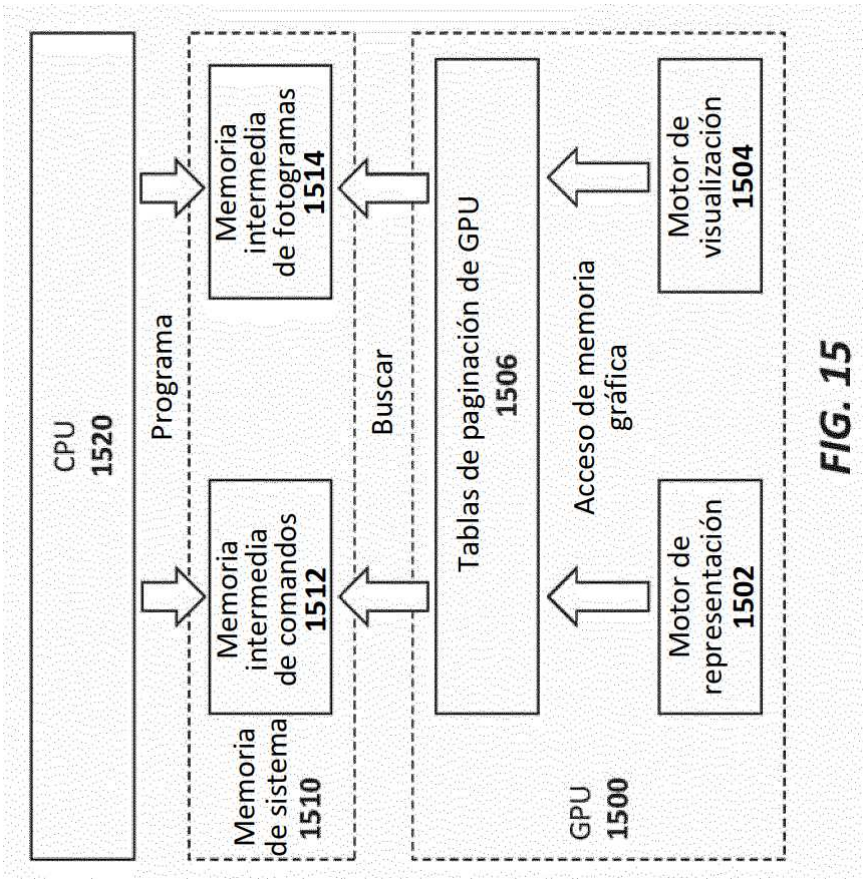

CPU
1520
Programa
Memoria
de sistema
1510
Memoria
intermedia
de comandos
1512
Memoria
intermedia
de fotogramas
1514
Buscar
GPU
1500
Tablas de paginación de GPU
1506
Acceso de memoria
gráfica
Motor de
representación
1502
Motor de
visualización
1504

FIG. 15

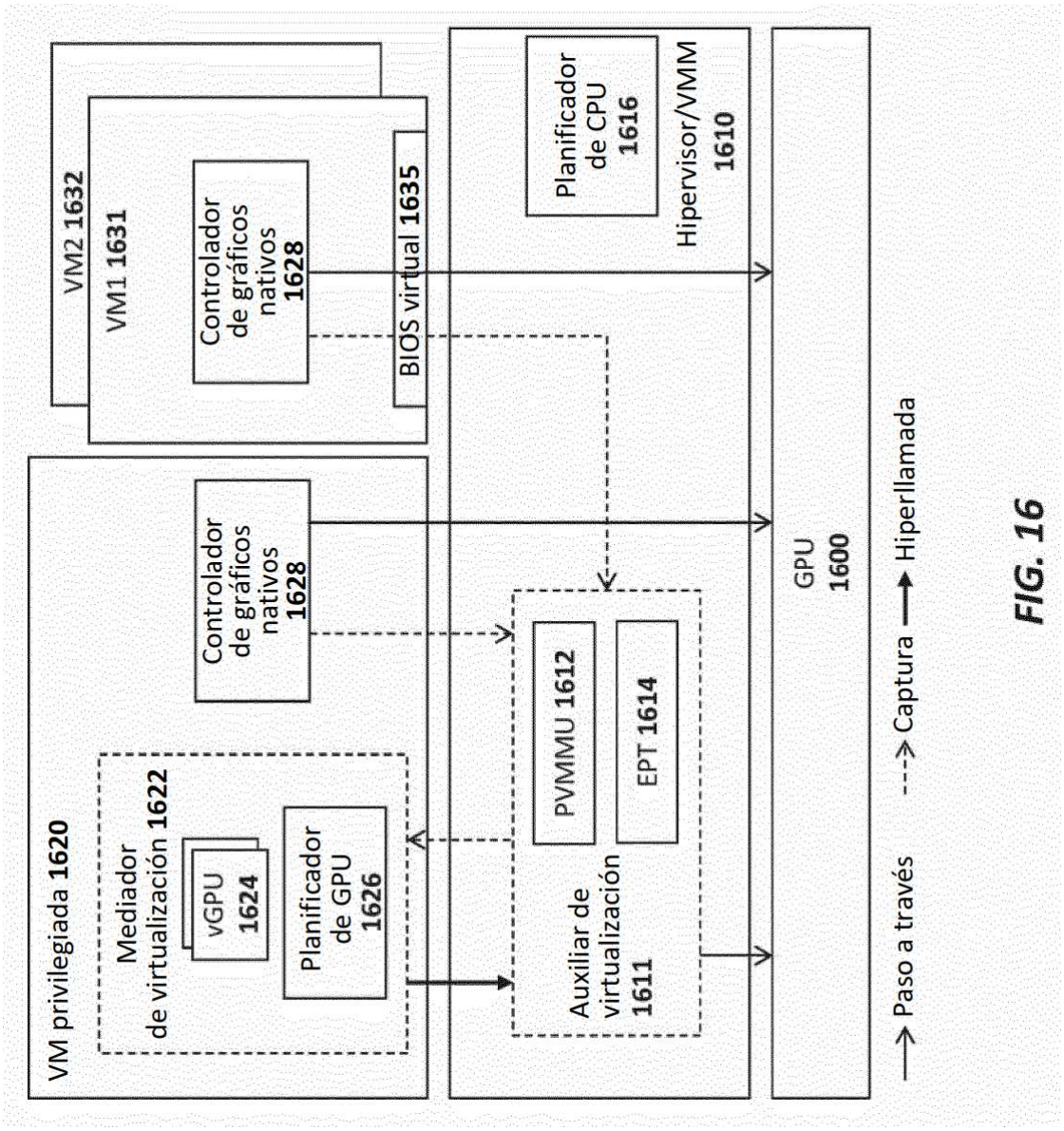

VM2 1632
VM1 1631
Controlador de gráficos nativos 1628
BIOS virtual 1635
Planificador de CPU 1616
Hipervisor/VMM 1610
VM privilegiada 1620
Mediador de virtualización 1622
vGPU 1624
Planificador de GPU 1626
Controlador de gráficos nativos 1628
PVMMU 1612
EPT 1614
Auxiliar de virtualización 1611
GPU 1600
Paso a través
Captura
Hiperllamada

FIG. 16

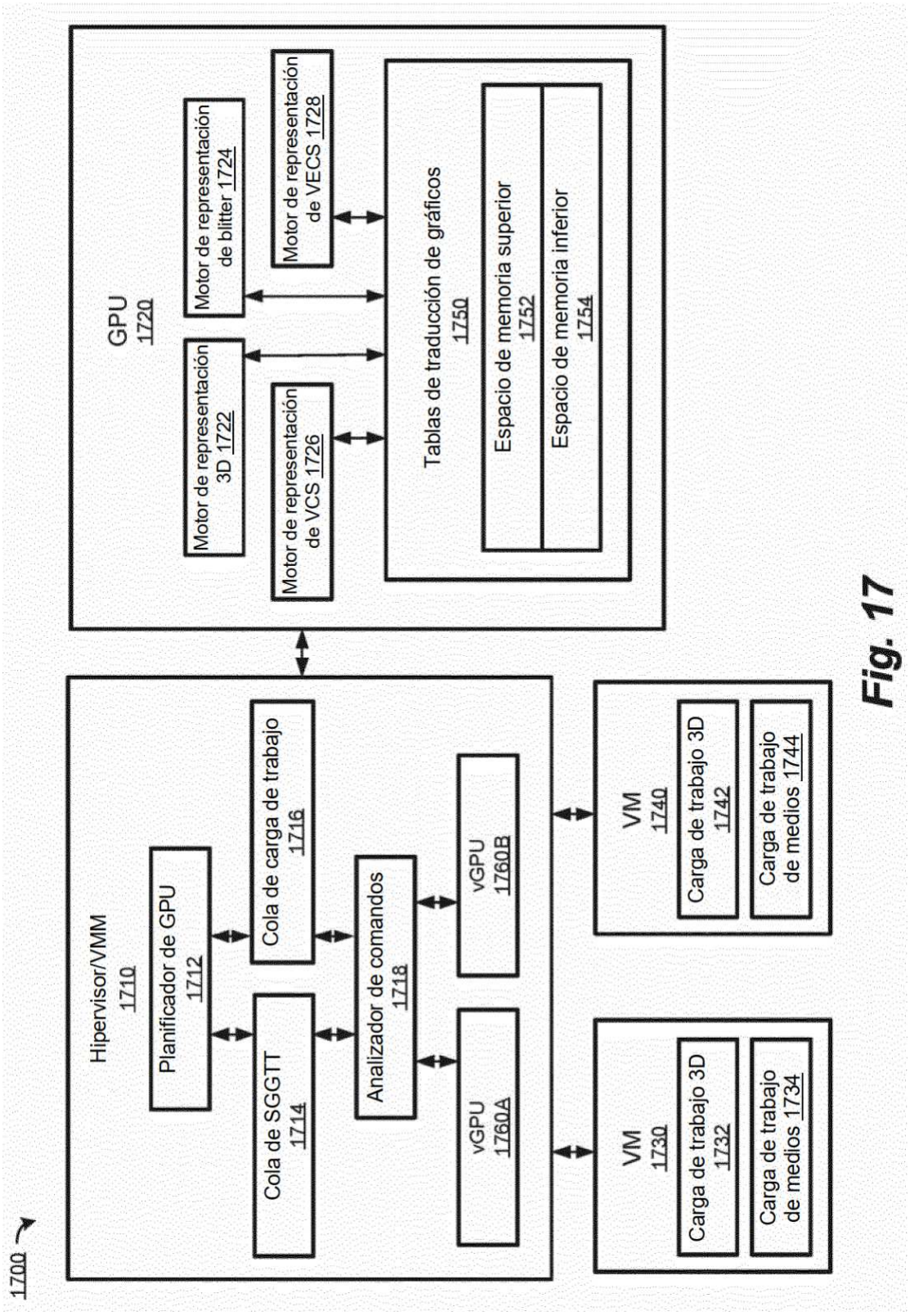

1700
Hipervisor/VMM
1710
Planificador de GPU
1712
Cola de SGGTT
1714
Cola de carga de trabajo
1716
Analizador de comandos
1718
vGPU
1760A
vGPU
1760B
VM
1730
Carga de trabajo 3D
1732
Carga de trabajo de medios 1734
VM
1740
Carga de trabajo 3D
1742
Carga de trabajo de medios 1744
GPU
1720
Motor de representación 3D 1722
Motor de representación de blitter 1724
Motor de representación de VCS 1726
Motor de representación de VECS 1728
Tablas de traducción de gráficos
1750
Espacio de memoria superior
1752
Espacio de memoria inferior
1754

Fig. 17

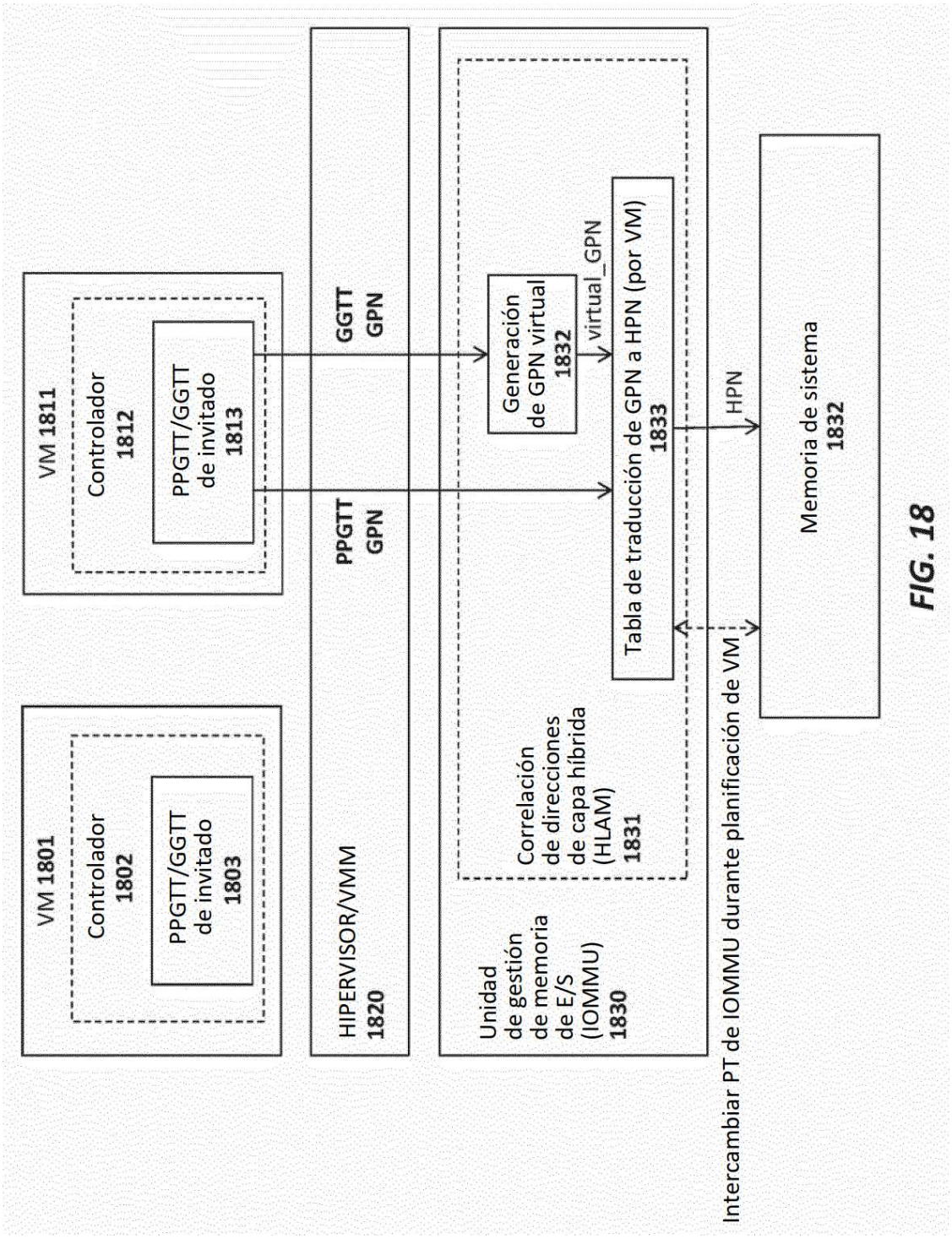

VM 1801
Controlador
1802
PPGTT/GGTT
de invitado
1803
VM 1811
Controlador
1812
PPGTT/GGTT
de invitado
1813
HIPERVISOR/VMM
1820
PPGTT
GPN
GGTT
GPN
Unidad
de gestión
de memoria
de E/S
(IOMMU)
1830
Correlación
de direcciones
de capa híbrida
(HLAM)
1831
Generación
de GPN virtual
1832
virtual_GPN
Tabla de traducción de GPN a HPN (por VM)
1833
HPN
Memoria de sistema
1832
Intercambiar PT de IOMMU durante planificación de VM

FIG. 18

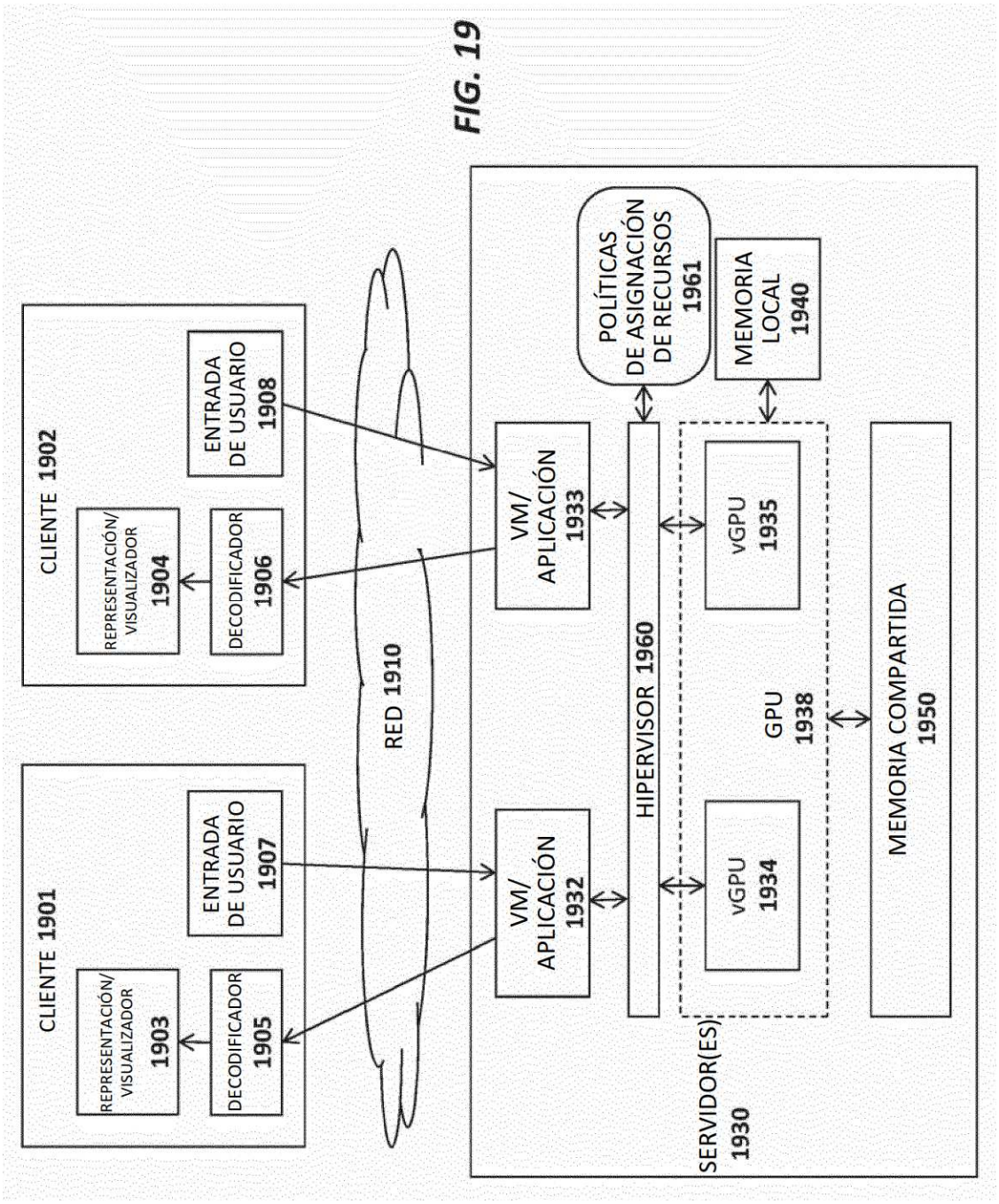

CLIENTE 1901
REPRESENTACIÓN/ VISUALIZADOR 1903
DECODIFICADOR 1905
ENTRADA DE USUARIO 1907
CLIENTE 1902
REPRESENTACIÓN/ VISUALIZADOR 1904
DECODIFICADOR 1906
ENTRADA DE USUARIO 1908
RED 1910
SERVIDOR(ES) 1930
VM/ APLICACIÓN 1932
VM/ APLICACIÓN 1933
HIPERVISOR 1960
GPU 1938
vGPU 1934
vGPU 1935
POLÍTICAS DE ASIGNACIÓN DE RECURSOS 1961
MEMORIA LOCAL 1940
MEMORIA COMPARTIDA 1950

FIG. 19

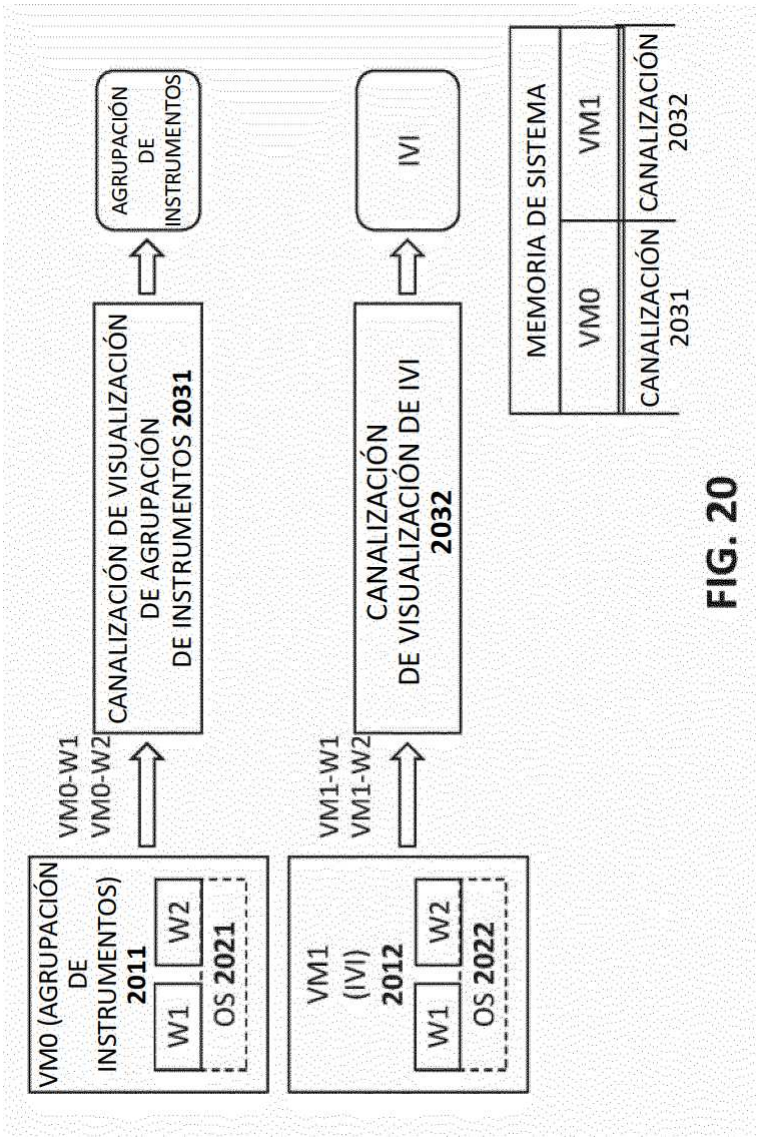

VM0 (AGRUPACIÓN DE INSTRUMENTOS) 2011
W1
W2
OS 2021
VM0-W1
VM0-W2
CANALIZACIÓN DE VISUALIZACIÓN DE AGRUPACIÓN DE INSTRUMENTOS 2031
AGRUPACIÓN DE INSTRUMENTOS
VM1 (IVI) 2012
W1
W2
OS 2022
VM1-W1
VM1-W2
CANALIZACIÓN DE VISUALIZACIÓN DE IVI 2032
IVI
MEMORIA DE SISTEMA
VM0
VM1
CANALIZACIÓN 2031
CANALIZACIÓN 2032

FIG. 20

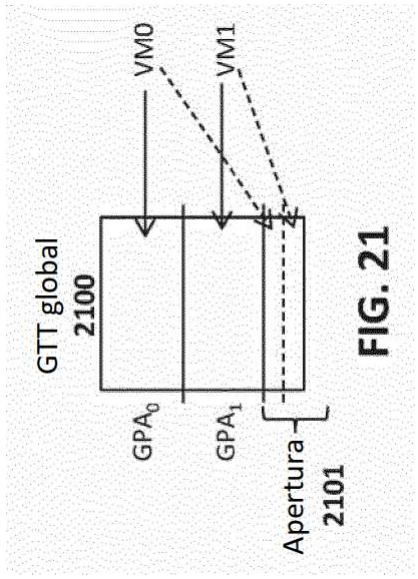

GTT global 2100
VM0
VM1
GPA0
GPA1
Apertura 2101

FIG. 21

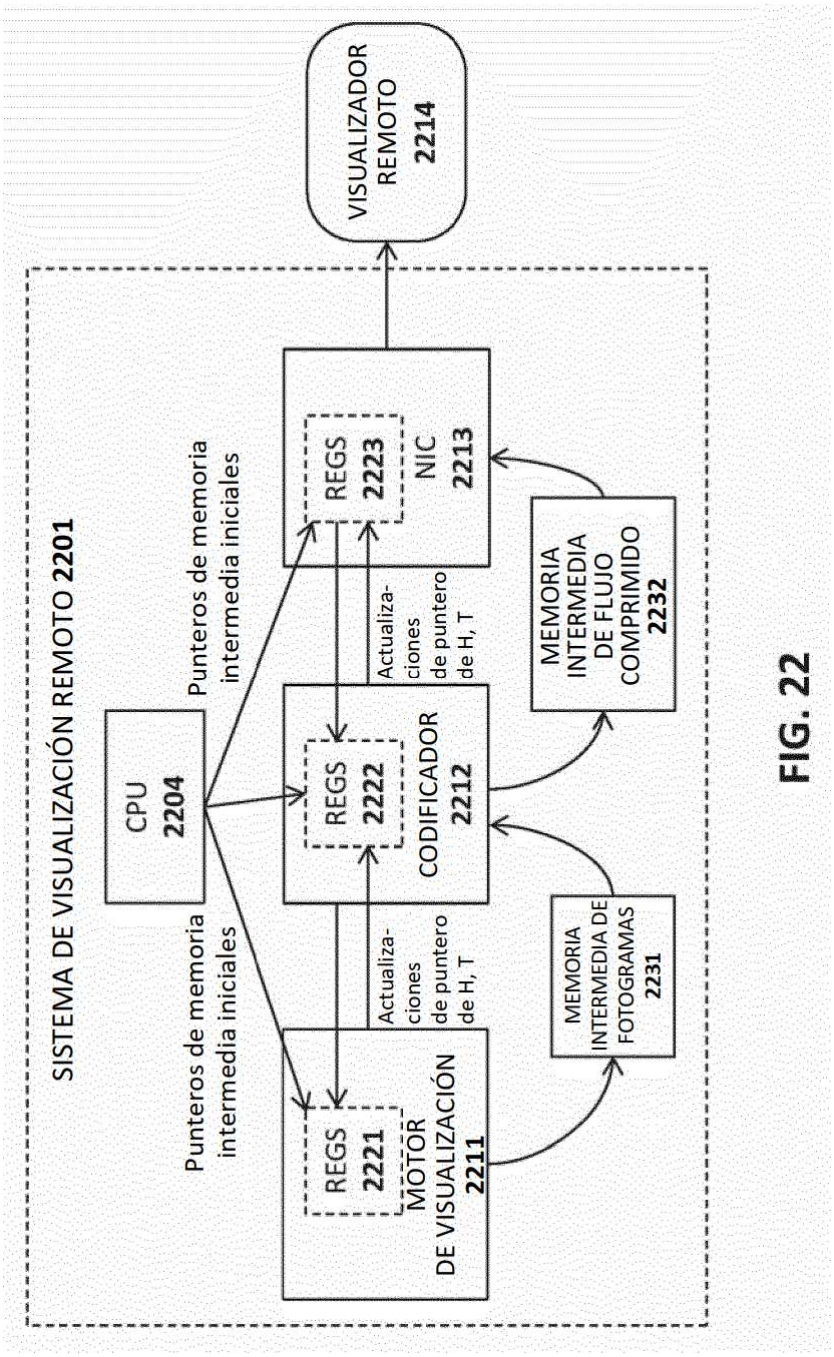

SISTEMA DE VISUALIZACIÓN REMOTO 2201
VISUALIZADOR REMOTO 2214
CPU 2204
Punteros de memoria intermedia iniciales
Punteros de memoria intermedia iniciales
REGS 2223
NIC 2213
REGS 2222
CODIFICADOR 2212
Actualiza-ciones de puntero de H, T
Actualiza-ciones de puntero de H, T
MEMORIA INTERMEDIA DE FLUJO COMPRIMIDO 2232
REGS 2221
MOTOR DE VISUALIZACIÓN 2211
MEMORIA INTERMEDIA DE FOTOGRAMAS 2231

FIG. 22

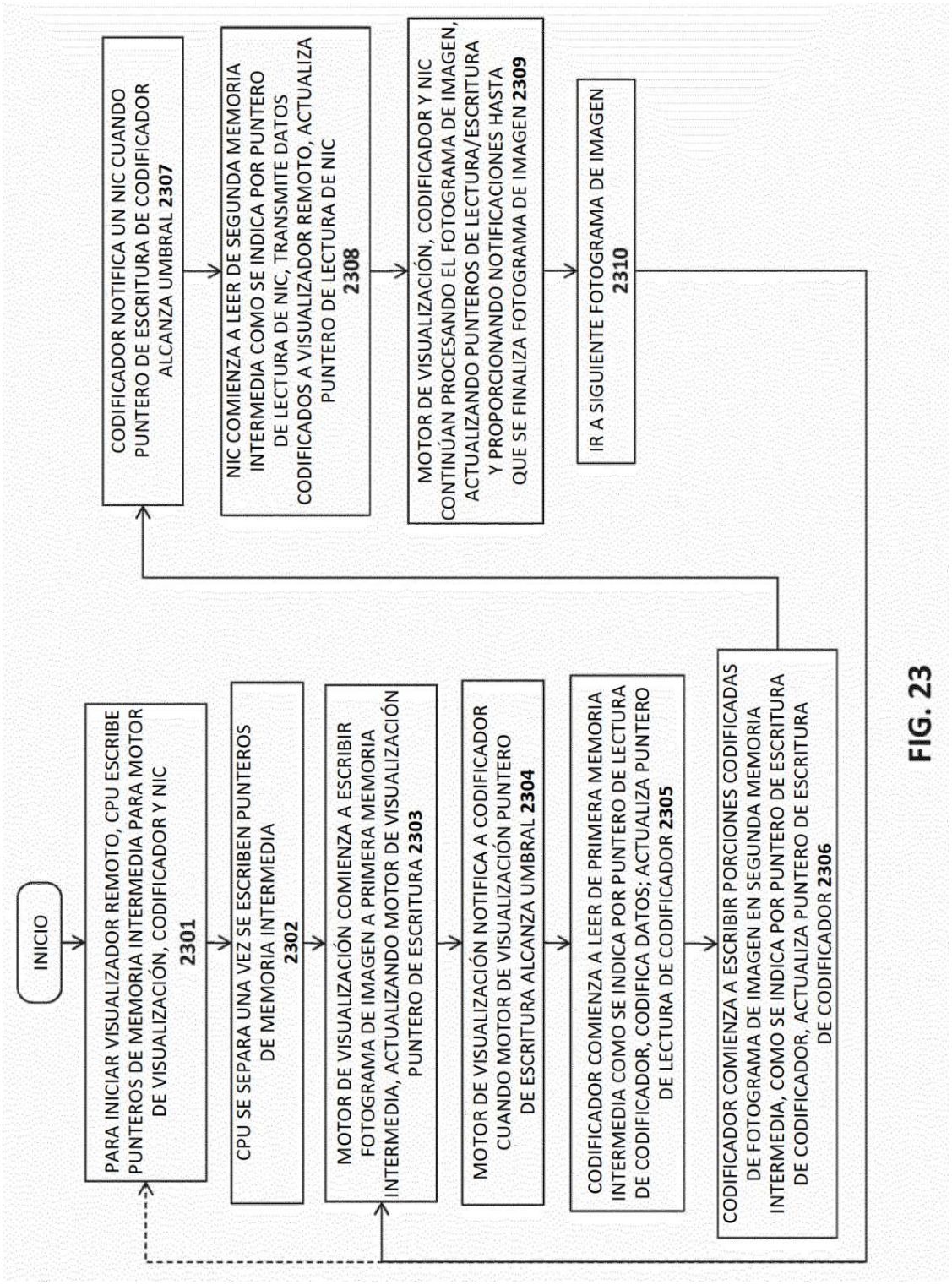

INICIO
PARA INICIAR VISUALIZADOR REMOTO, CPU ESCRIBE PUNTEROS DE MEMORIA INTERMEDIA PARA MOTOR DE VISUALIZACIÓN, CODIFICADOR Y NIC 2301
CPU SE SEPARA UNA VEZ SE ESCRIBEN PUNTEROS DE MEMORIA INTERMEDIA 2302
MOTOR DE VISUALIZACIÓN COMIENZA A ESCRIBIR FOTOGRAMA DE IMAGEN A PRIMERA MEMORIA INTERMEDIA, ACTUALIZANDO MOTOR DE VISUALIZACIÓN PUNTERO DE ESCRITURA 2303
MOTOR DE VISUALIZACIÓN NOTIFICA A CODIFICADOR CUANDO MOTOR DE VISUALIZACIÓN PUNTERO DE ESCRITURA ALCANZA UMBRAL 2304
CODIFICADOR COMIENZA A LEER DE PRIMERA MEMORIA INTERMEDIA COMO SE INDICA POR PUNTERO DE LECTURA DE CODIFICADOR, CODIFICA DATOS; ACTUALIZA PUNTERO DE LECTURA DE CODIFICADOR 2305
CODIFICADOR COMIENZA A ESCRIBIR PORCIONES CODIFICADAS DE FOTOGRAMA DE IMAGEN EN SEGUNDA MEMORIA INTERMEDIA, COMO SE INDICA POR PUNTERO DE ESCRITURA DE CODIFICADOR, ACTUALIZA PUNTERO DE ESCRITURA DE CODIFICADOR 2306
CODIFICADOR NOTIFICA UN NIC CUANDO PUNTERO DE ESCRITURA DE CODIFICADOR ALCANZA UMBRAL 2307
NIC COMIENZA A LEER DE SEGUNDA MEMORIA INTERMEDIA COMO SE INDICA POR PUNTERO DE LECTURA DE NIC, TRANSMITE DATOS CODIFICADOS A VISUALIZADOR REMOTO, ACTUALIZA PUNTERO DE LECTURA DE NIC 2308
MOTOR DE VISUALIZACIÓN, CODIFICADOR Y NIC CONTINÚAN PROCESANDO EL FOTOGRAMA DE IMAGEN, ACTUALIZANDO PUNTEROS DE LECTURA/ESCRITURA Y PROPORCIONANDO NOTIFICACIONES HASTA QUE SE FINALIZA FOTOGRAMA DE IMAGEN 2309
IR A SIGUIENTE FOTOGRAMA DE IMAGEN 2310

FIG. 23

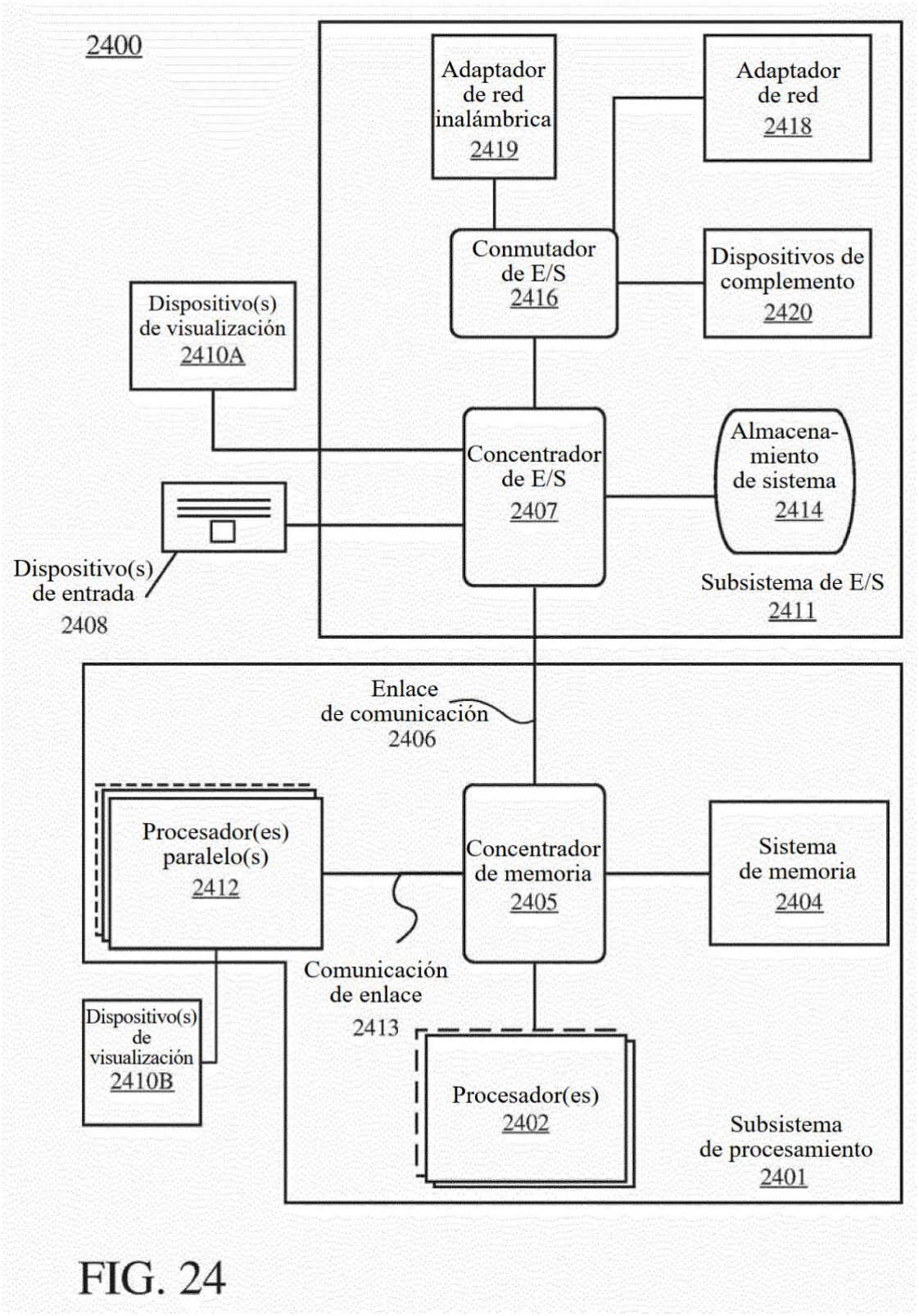

2400
Adaptador de red inalámbrica 2419
Adaptador de red 2418
Conmutador de E/S 2416
Dispositivos de complemento 2420
Dispositivo(s) de visualización 2410A
Concentrador de E/S 2407
Almacena-miento de sistema 2414
Dispositivo(s) de entrada 2408
Subsistema de E/S 2411
Enlace de comunicación 2406
Procesador(es) paralelo(s) 2412
Concentrador de memoria 2405
Sistema de memoria 2404
Comunicación de enlace 2413
Dispositivo(s) de visualización 2410B
Procesador(es) 2402
Subsistema de procesamiento 2401

FIG. 24

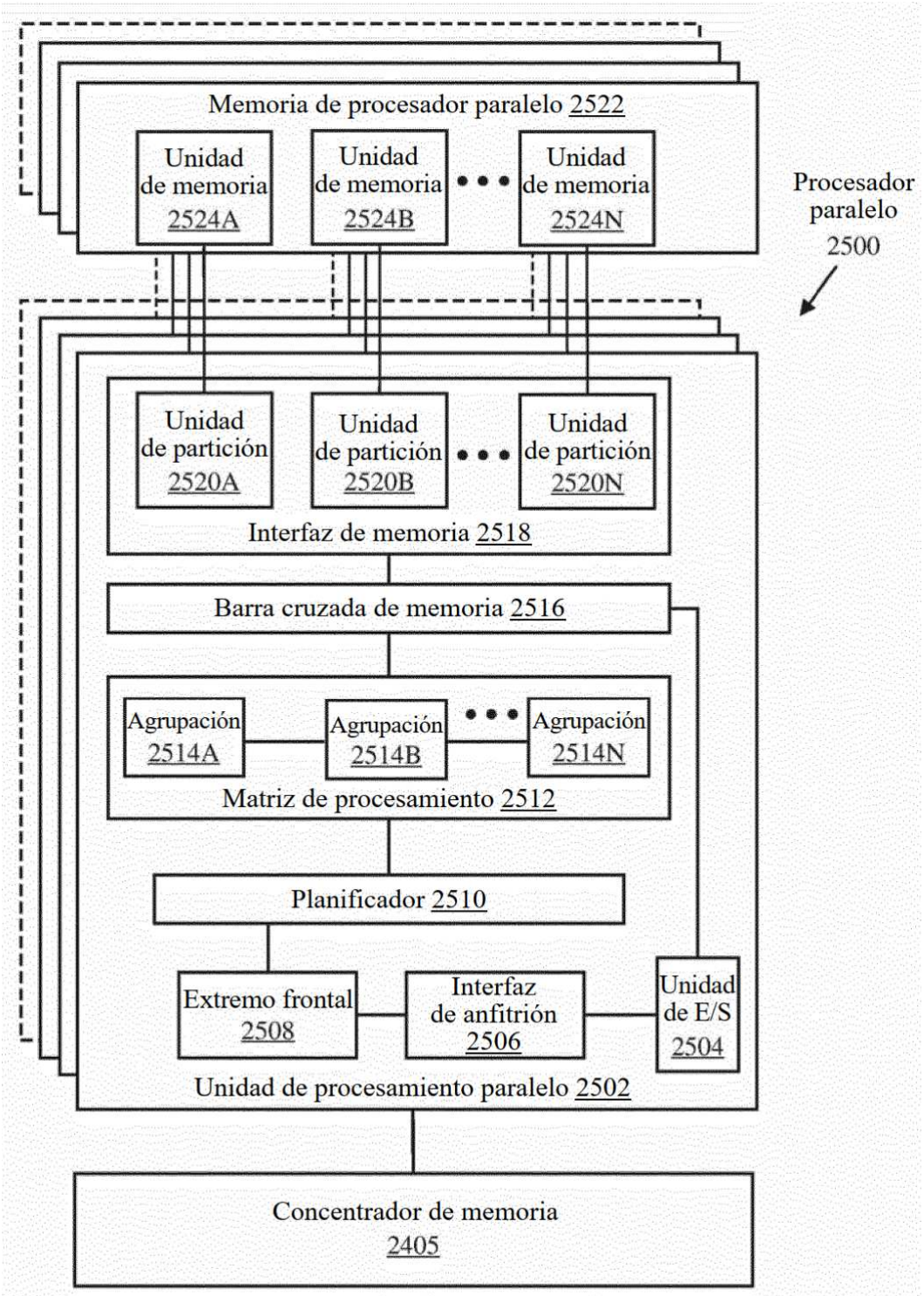

Memoria de procesador paralelo 2522
Unidad de memoria 2524A
Unidad de memoria 2524B
Unidad de memoria 2524N
Procesador paralelo 2500
Unidad de partición 2520A
Unidad de partición 2520B
Unidad de partición 2520N
Interfaz de memoria 2518
Barra cruzada de memoria 2516
Agrupación 2514A
Agrupación 2514B
Agrupación 2514N
Matriz de procesamiento 2512
Planificador 2510
Extremo frontal 2508
Interfaz de anfitrión 2506
Unidad de E/S 2504
Unidad de procesamiento paralelo 2502
Concentrador de memoria 2405

FIG. 25A

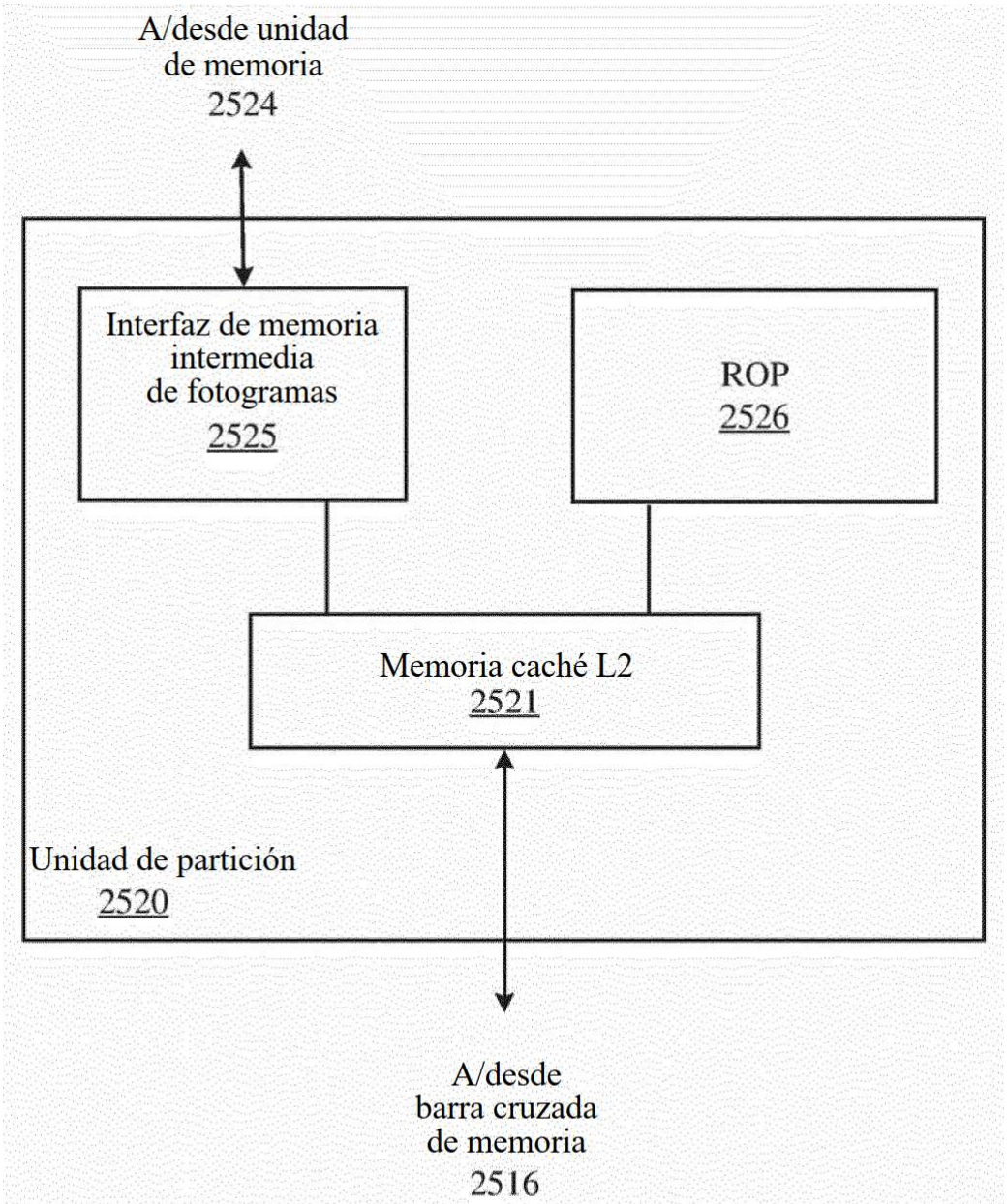

A/desde unidad
de memoria
2524
Interfaz de memoria
intermedia
de fotogramas
2525
ROP
2526
Memoria caché L2
2521
Unidad de partición
2520
A/desde
barra cruzada
de memoria
2516

FIG. 25B

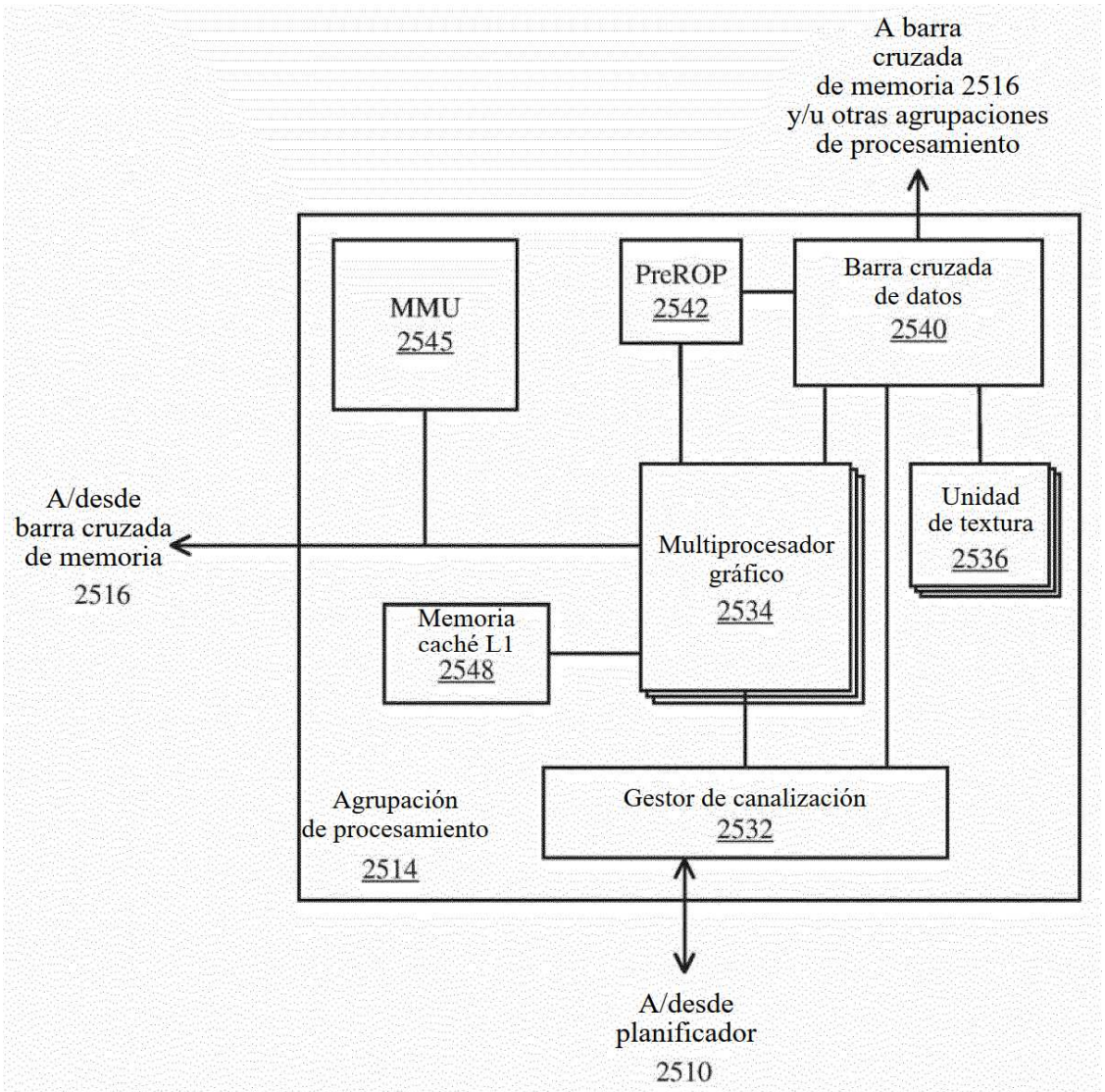

A barra cruzada de memoria 2516 y/u otras agrupaciones de procesamiento
MMU 2545
PreROP 2542
Barra cruzada de datos 2540
A/desde barra cruzada de memoria 2516
Multiprocesador gráfico 2534
Unidad de textura 2536
Memoria caché L1 2548
Agrupación de procesamiento 2514
Gestor de canalización 2532
A/desde planificador 2510

FIG. 25C

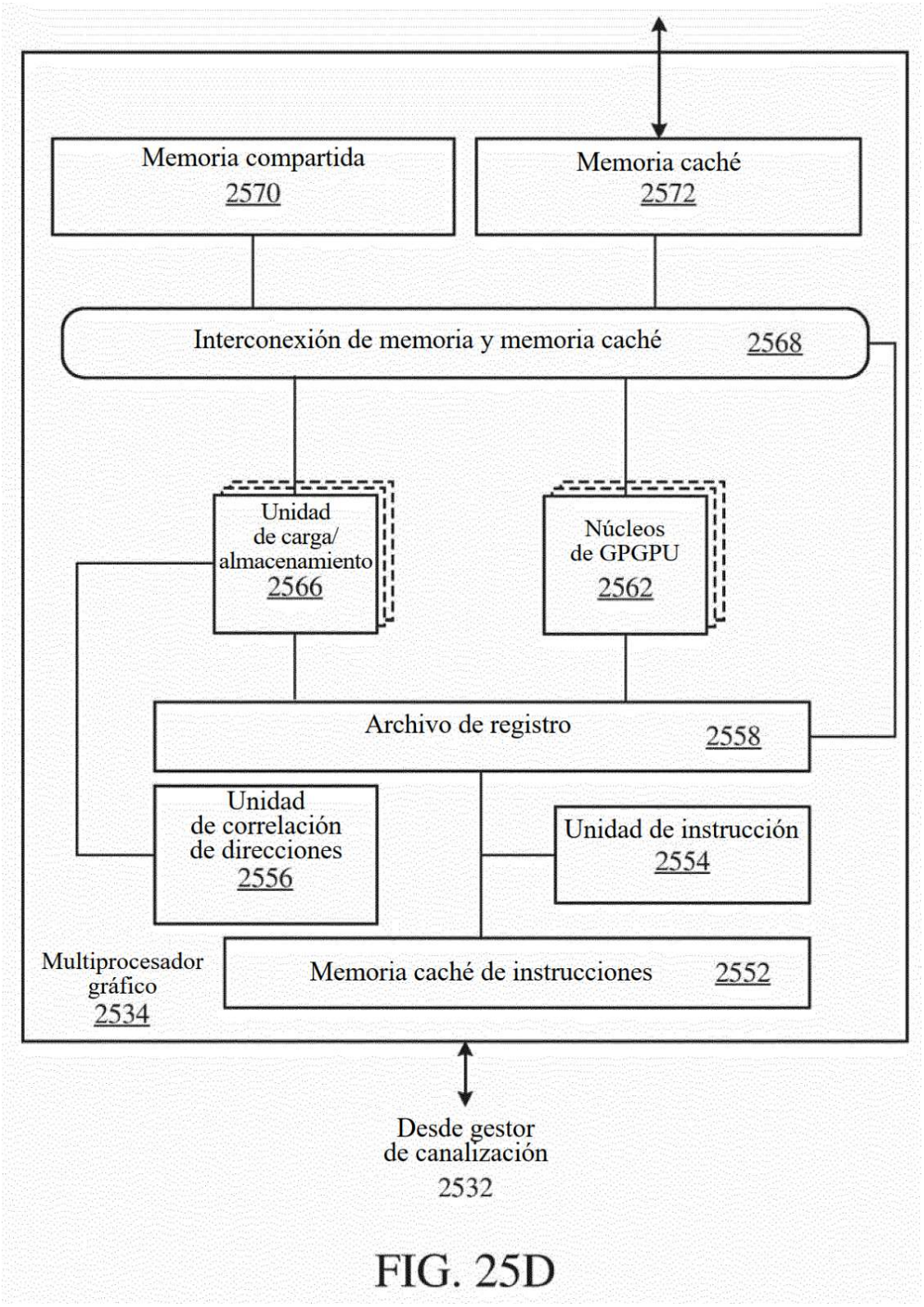

Memoria compartida
2570
Memoria caché
2572
Interconexión de memoria y memoria caché
2568
Unidad
de carga/
almacenamiento
2566
Núcleos
de GPGPU
2562
Archivo de registro
2558
Unidad
de correlación
de direcciones
2556
Unidad de instrucción
2554
Multiprocesador
gráfico
2534
Memoria caché de instrucciones
2552
Desde gestor
de canalización
2532

FIG. 25D

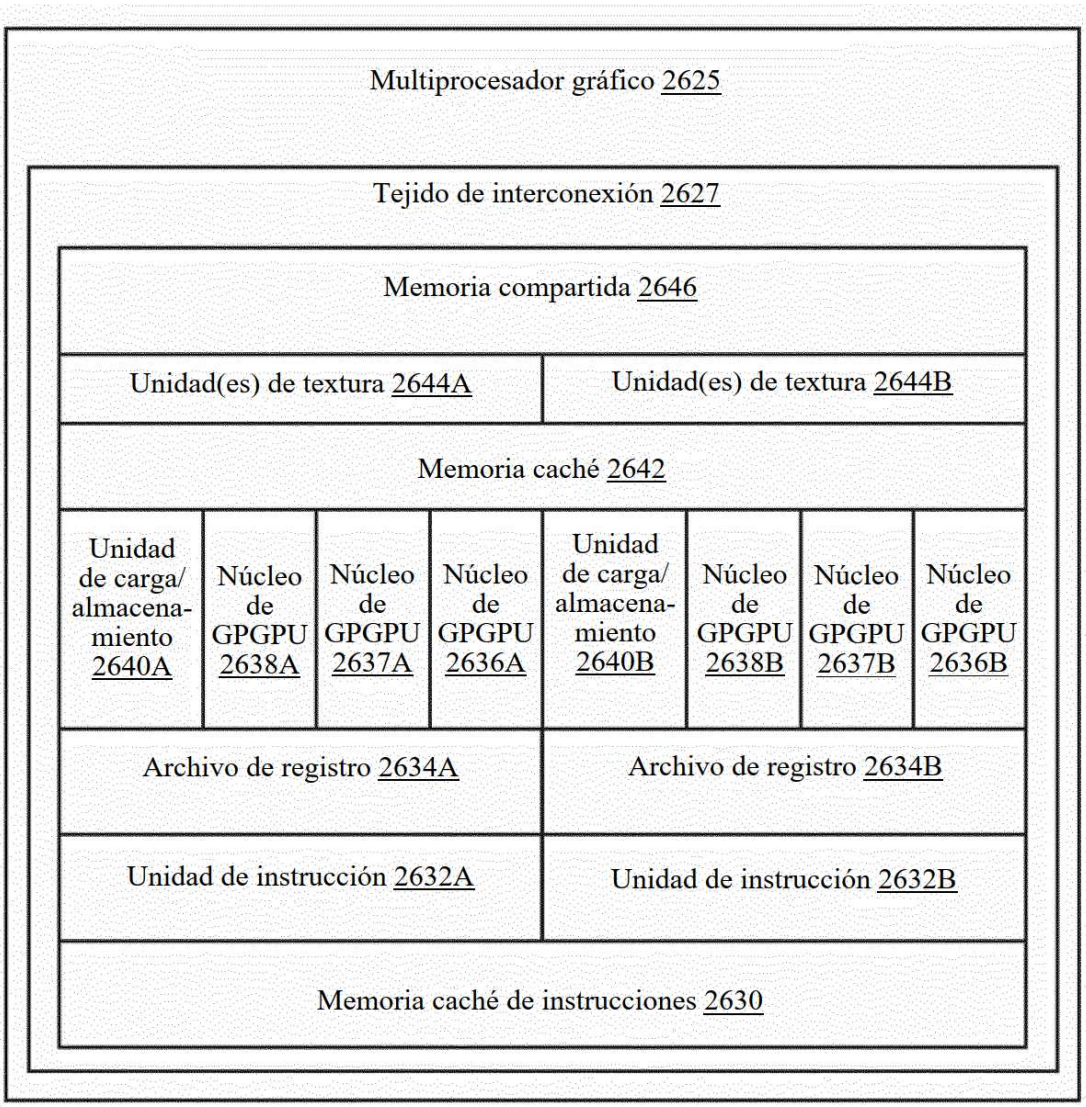

Multiprocesador gráfico 2625
Tejido de interconexión 2627
Memoria compartida 2646
Unidad(es) de textura 2644A
Unidad(es) de textura 2644B
Memoria caché 2642
Unidad de carga/ almacena-miento 2640A
Núcleo de GPGPU 2638A
Núcleo de GPGPU 2637A
Núcleo de GPGPU 2636A
Unidad de carga/ almacena-miento 2640B
Núcleo de GPGPU 2638B
Núcleo de GPGPU 2637B
Núcleo de GPGPU 2636B
Archivo de registro 2634A
Archivo de registro 2634B
Unidad de instrucción 2632A
Unidad de instrucción 2632B
Memoria caché de instrucciones 2630

FIG. 26A

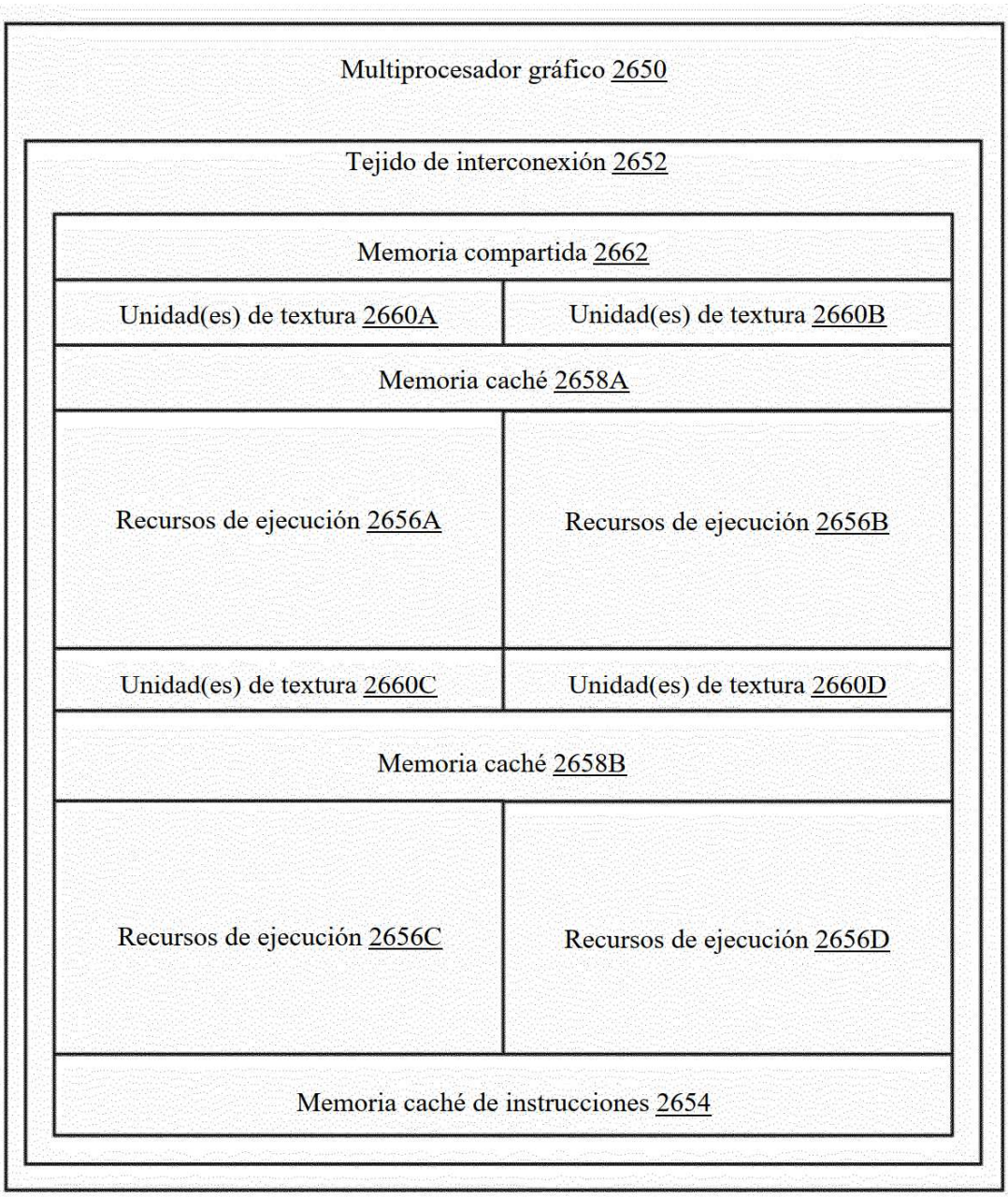

Multiprocesador gráfico 2650
Tejido de interconexión 2652
Memoria compartida 2662
Unidad(es) de textura 2660A
Unidad(es) de textura 2660B
Memoria caché 2658A
Recursos de ejecución 2656A
Recursos de ejecución 2656B
Unidad(es) de textura 2660C
Unidad(es) de textura 2660D
Memoria caché 2658B
Recursos de ejecución 2656C
Recursos de ejecución 2656D
Memoria caché de instrucciones 2654

FIG. 26B

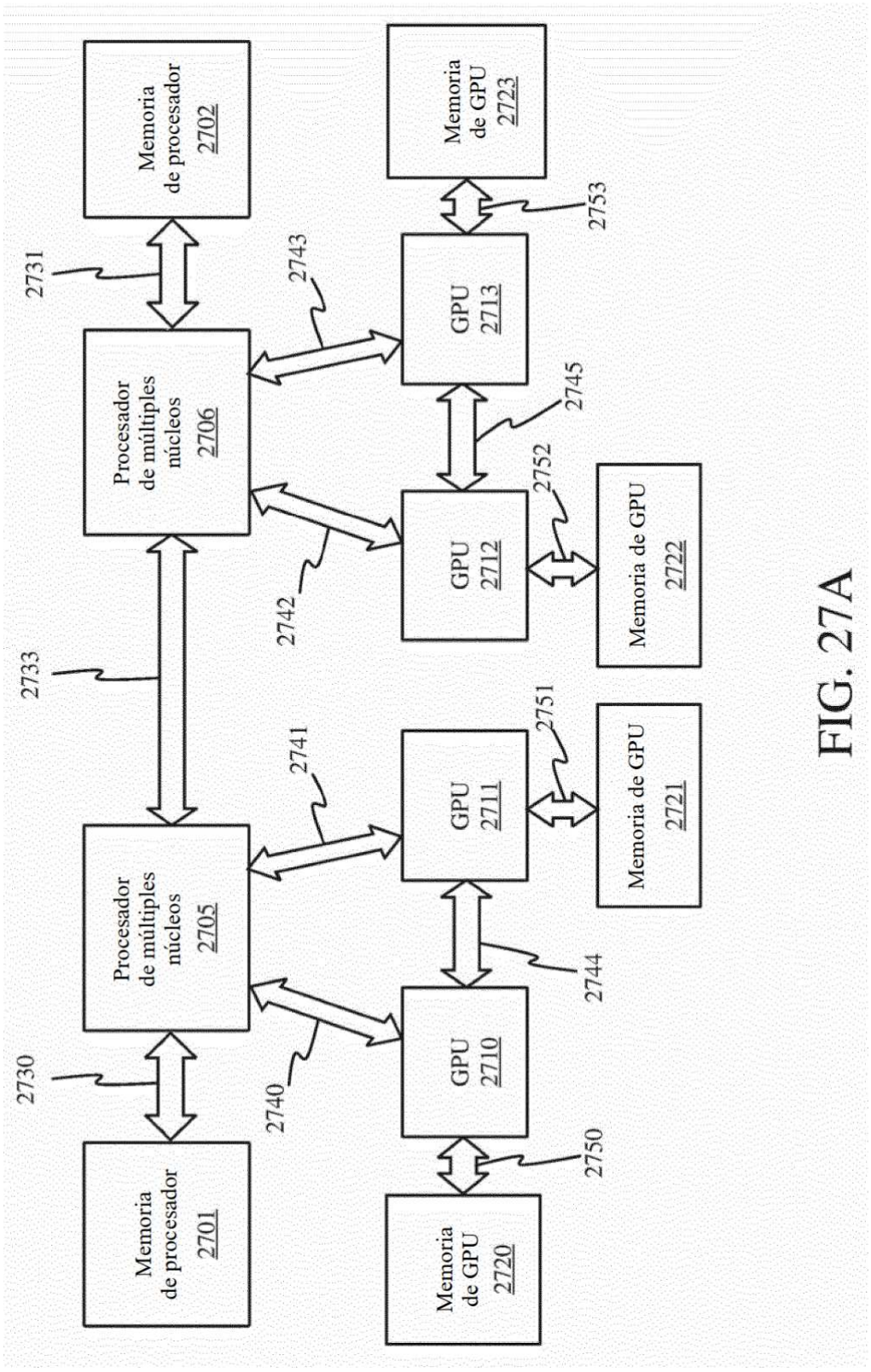

Memoria de procesador 2701
Memoria de procesador 2702
Procesador de múltiples núcleos 2705
Procesador de múltiples núcleos 2706
GPU 2710
GPU 2711
GPU 2712
GPU 2713
Memoria de GPU 2720
Memoria de GPU 2721
Memoria de GPU 2722
Memoria de GPU 2723
2730
2731
2733
2740
2741
2742
2743
2744
2745
2750
2751
2752
2753

FIG. 27A

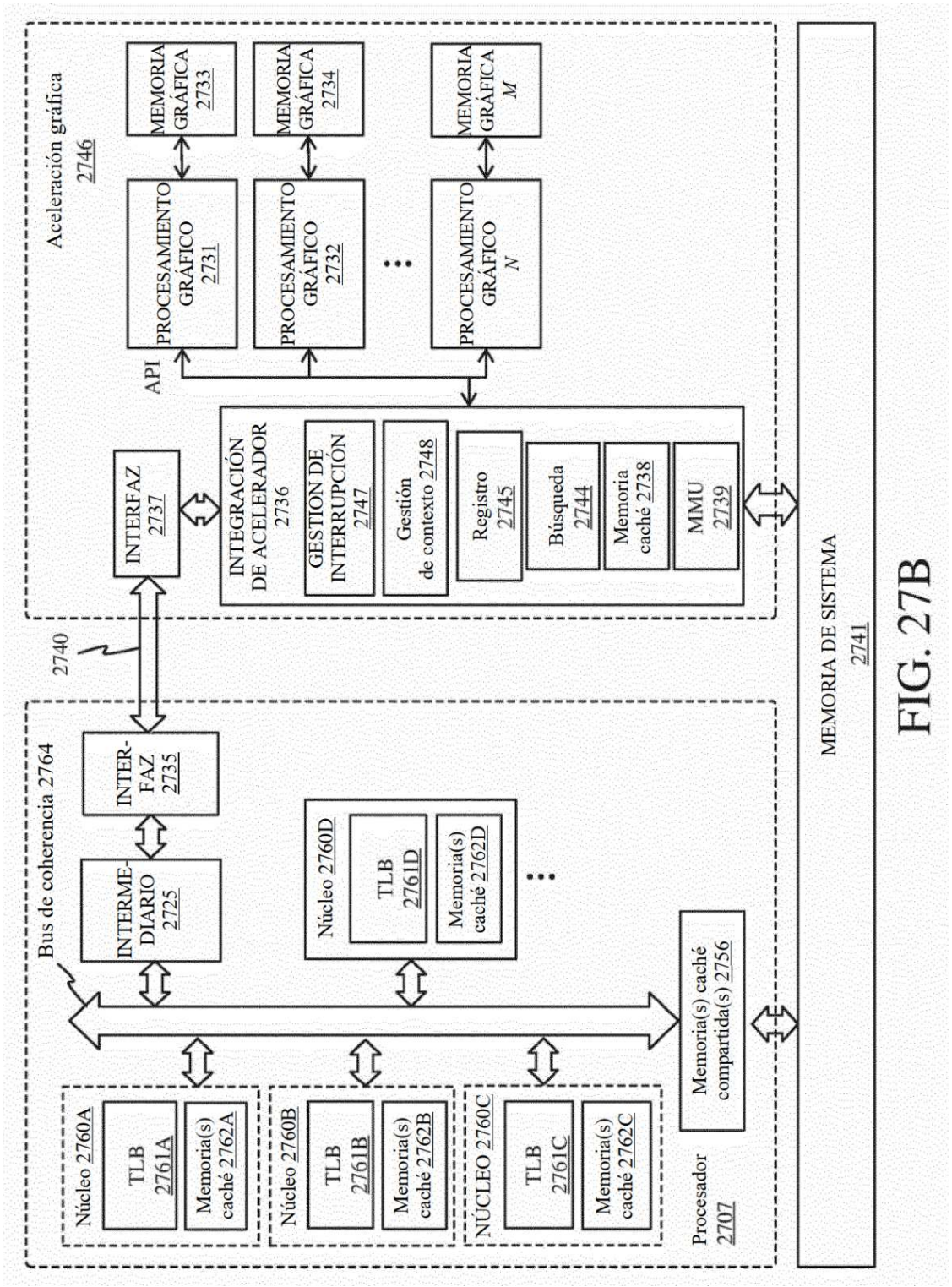

Aceleración gráfica 2746
MEMORIA GRÁFICA 2733
MEMORIA GRÁFICA 2734
MEMORIA GRÁFICA M
PROCESAMIENTO GRÁFICO 2731
PROCESAMIENTO GRÁFICO 2732
PROCESAMIENTO GRÁFICO N
API
INTERFAZ 2737
INTEGRACIÓN DE ACELERADOR 2736
GESTION DE INTERRUPCIÓN 2747
Gestión de contexto 2748
Registro 2745
Búsqueda 2744
Memoria caché 2738
MMU 2739
2740
Bus de coherencia 2764
INTERME-DIARIO 2725
INTER-FAZ 2735
Núcleo 2760A
TLB 2761A
Memoria(s) caché 2762A
Núcleo 2760B
TLB 2761B
Memoria(s) caché 2762B
NÚCLEO 2760C
TLB 2761C
Memoria(s) caché 2762C
Núcleo 2760D
TLB 2761D
Memoria(s) caché 2762D
Memoria(s) caché compartida(s) 2756
Procesador 2707
MEMORIA DE SISTEMA 2741

FIG. 27B

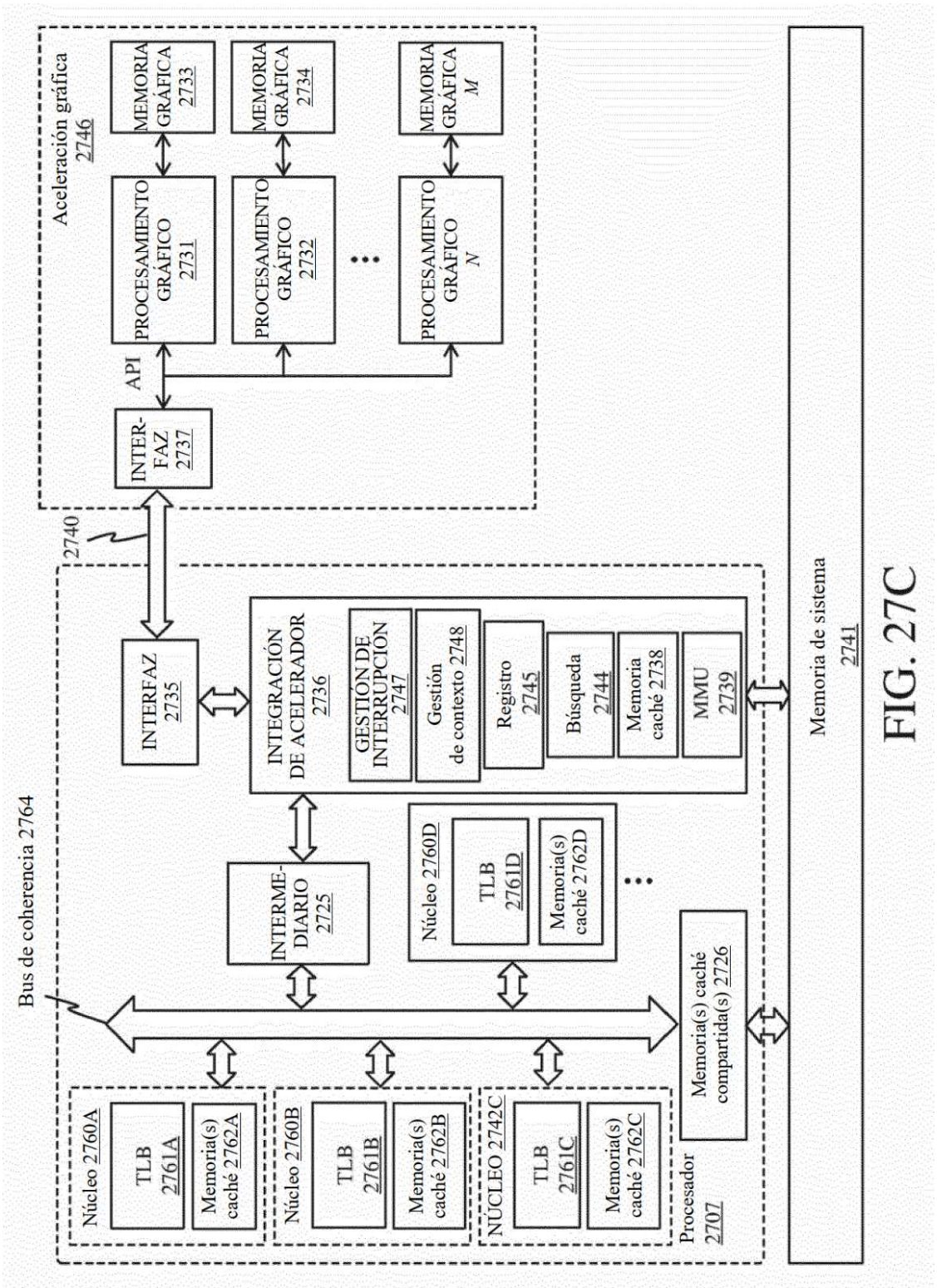

Aceleración gráfica 2746
MEMORIA GRÁFICA 2733
MEMORIA GRÁFICA 2734
MEMORIA GRÁFICA M
PROCESAMIENTO GRÁFICO 2731
PROCESAMIENTO GRÁFICO 2732
PROCESAMIENTO GRÁFICO N
API
INTER-FAZ 2737
2740
INTERFAZ 2735
INTEGRACIÓN DE ACELERADOR 2736
GESTIÓN DE INTERRUPCIÓN 2747
Gestión de contexto 2748
Registro 2745
Búsqueda 2744
Memoria caché 2738
MMU 2739
Bus de coherencia 2764
INTERME-DIARIO 2725
Núcleo 2760D
TLB 2761D
Memoria(s) caché 2762D
Memoria de sistema 2741
Núcleo 2760A
TLB 2761A
Memoria(s) caché 2762A
Núcleo 2760B
TLB 2761B
Memoria(s) caché 2762B
NÚCLEO 2742C
TLB 2761C
Memoria(s) caché 2762C
Memoria(s) caché compartida(s) 2726
Procesador 2707

FIG. 27C

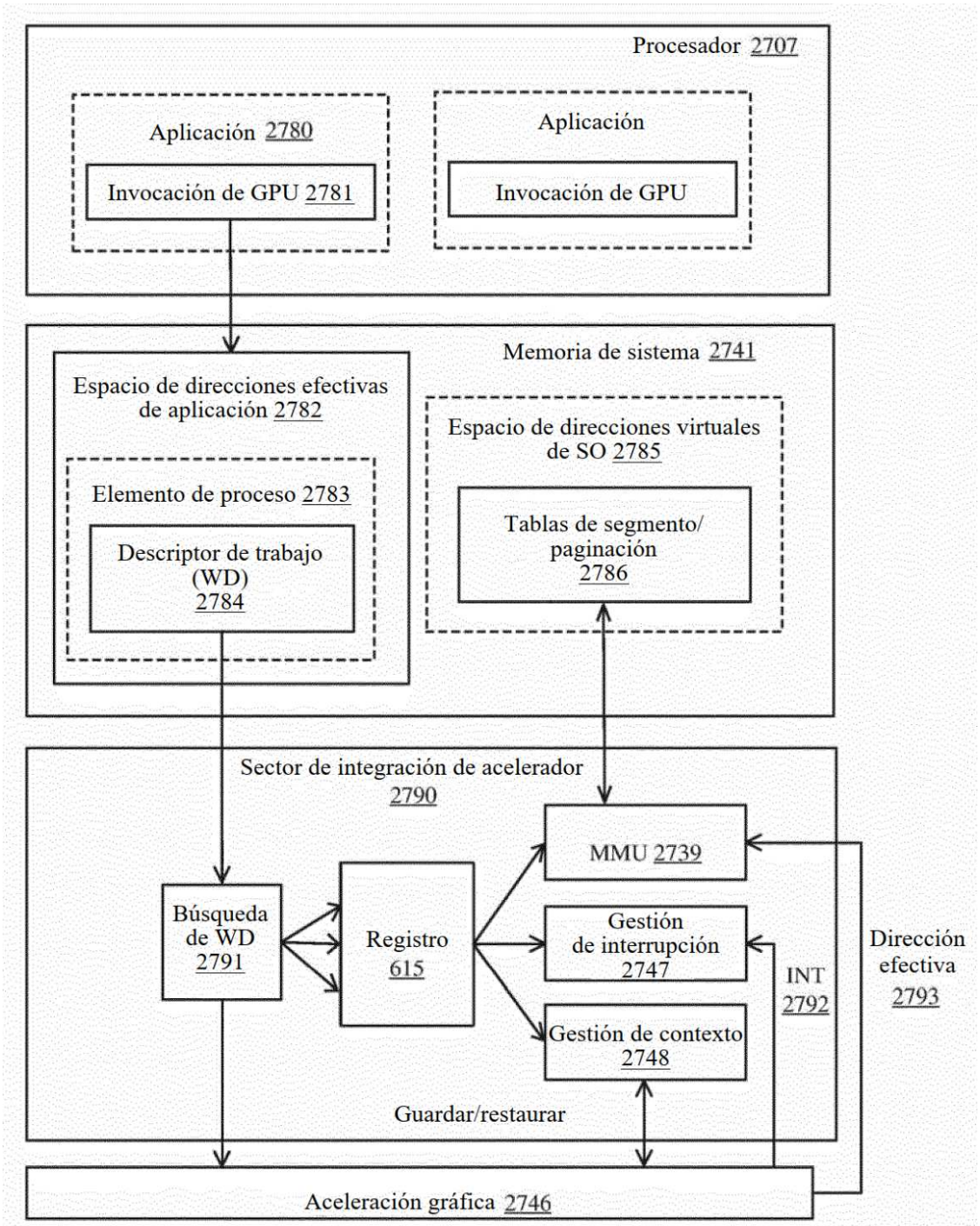

Procesador 2707
Aplicación 2780
Invocación de GPU 2781
Aplicación
Invocación de GPU
Memoria de sistema 2741
Espacio de direcciones efectivas de aplicación 2782
Elemento de proceso 2783
Descriptor de trabajo (WD) 2784
Espacio de direcciones virtuales de SO 2785
Tablas de segmento/ paginación 2786
Sector de integración de acelerador 2790
MMU 2739
Búsqueda de WD 2791
Registro 615
Gestión de interrupción 2747
Gestión de contexto 2748
INT 2792
Dirección efectiva 2793
Guardar/restaurar
Aceleración gráfica 2746

FIG. 27D

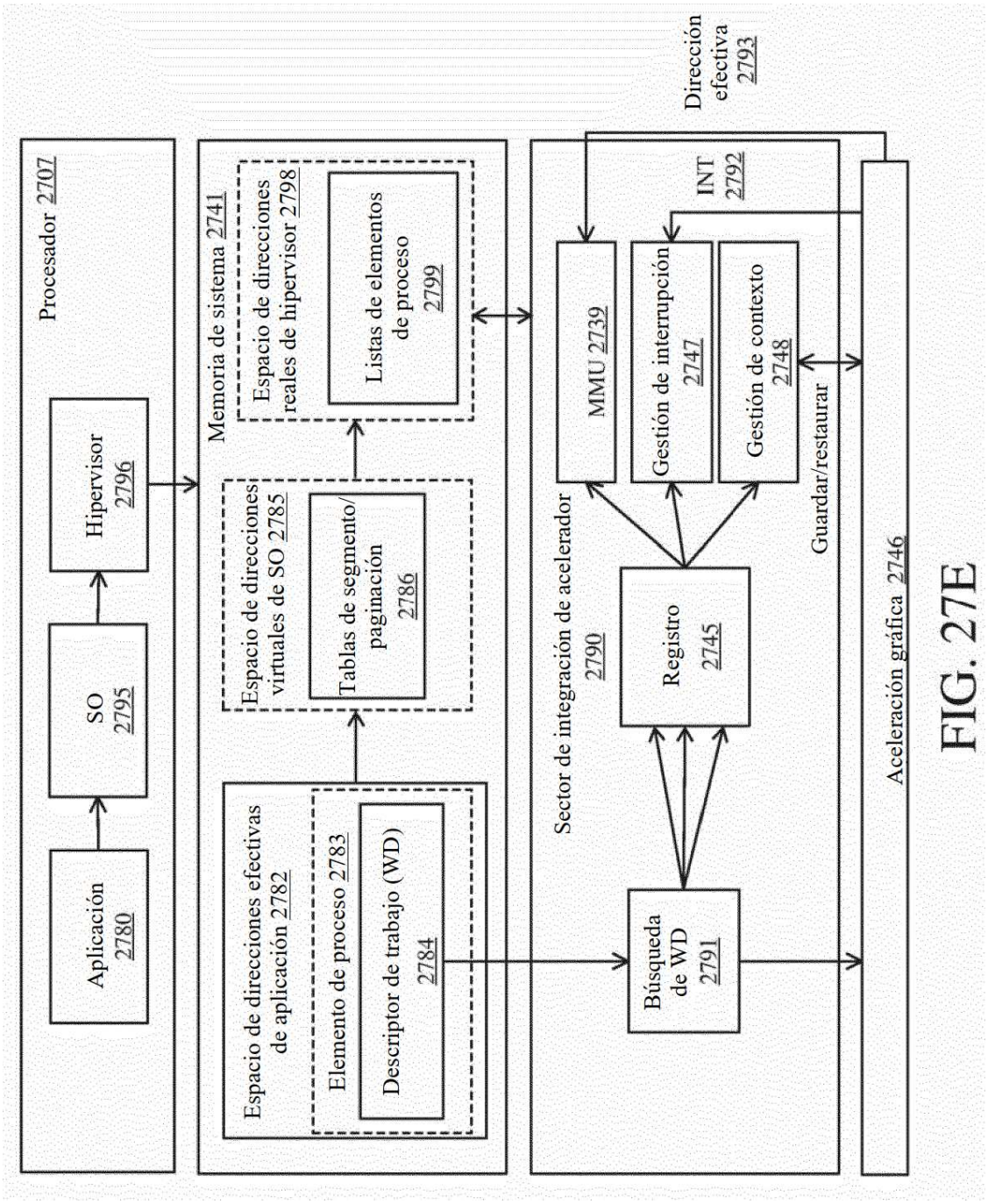

Procesador 2707
Aplicación 2780
SO 2795
Hipervisor 2796
Memoria de sistema 2741
Espacio de direcciones efectivas de aplicación 2782
Elemento de proceso 2783
Descriptor de trabajo (WD) 2784
Espacio de direcciones virtuales de SO 2785
Tablas de segmento/ paginación 2786
Espacio de direcciones reales de hipervisor 2798
Listas de elementos de proceso 2799
Sector de integración de acelerador 2790
Búsqueda de WD 2791
Registro 2745
MMU 2739
Gestión de interrupción 2747
Gestión de contexto 2748
Dirección efectiva 2793
INT 2792
Guardar/restaurar
Aceleración gráfica 2746

FIG. 27E

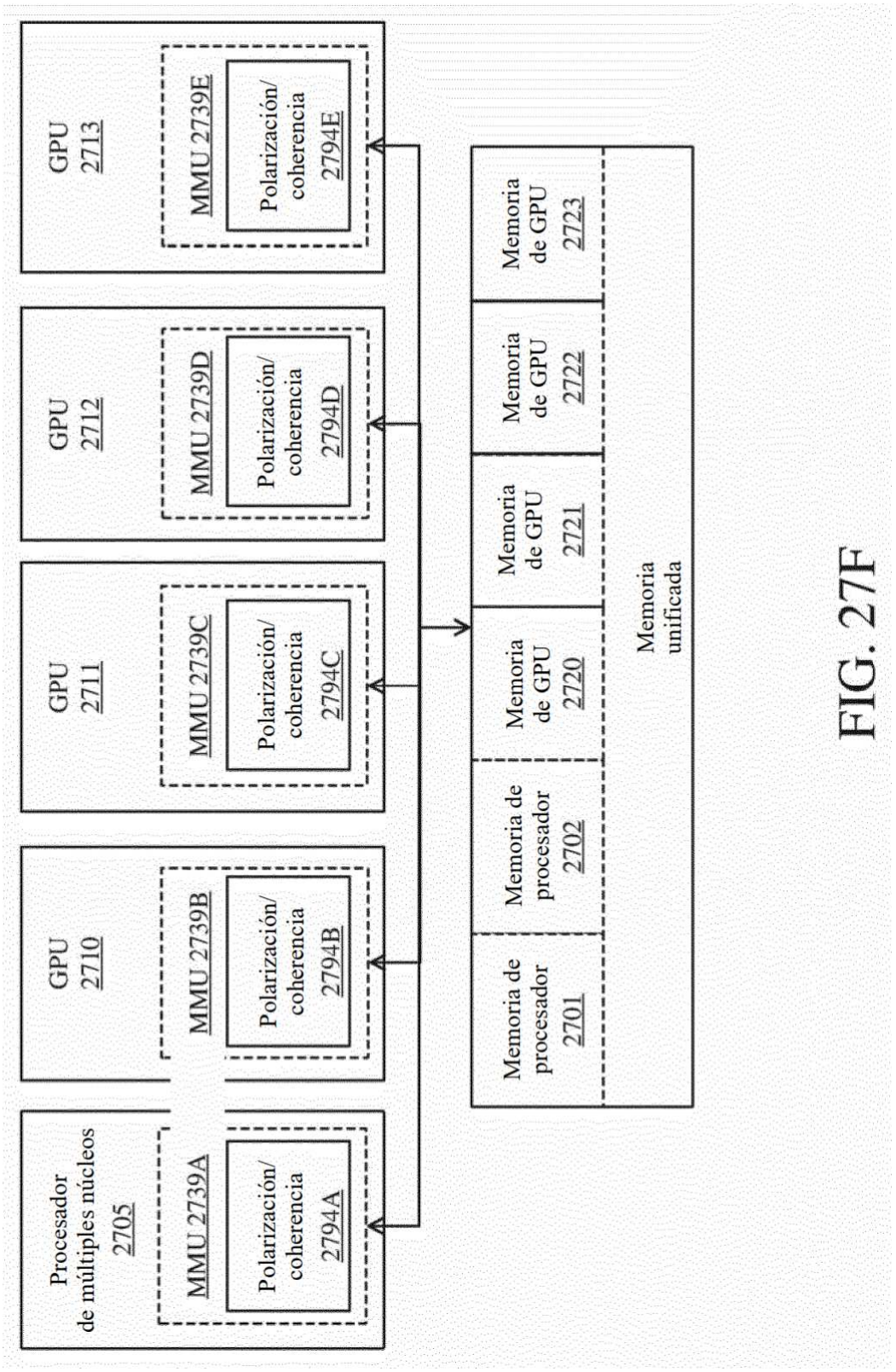

Procesador de múltiples núcleos 2705
GPU 2710
GPU 2711
GPU 2712
GPU 2713
MMU 2739A
MMU 2739B
MMU 2739C
MMU 2739D
MMU 2739E
Polarización/ coherencia 2794A
Polarización/ coherencia 2794B
Polarización/ coherencia 2794C
Polarización/ coherencia 2794D
Polarización/ coherencia 2794E
Memoria de procesador 2701
Memoria de procesador 2702
Memoria de GPU 2720
Memoria de GPU 2721
Memoria de GPU 2722
Memoria de GPU 2723
Memoria unificada

FIG. 27F

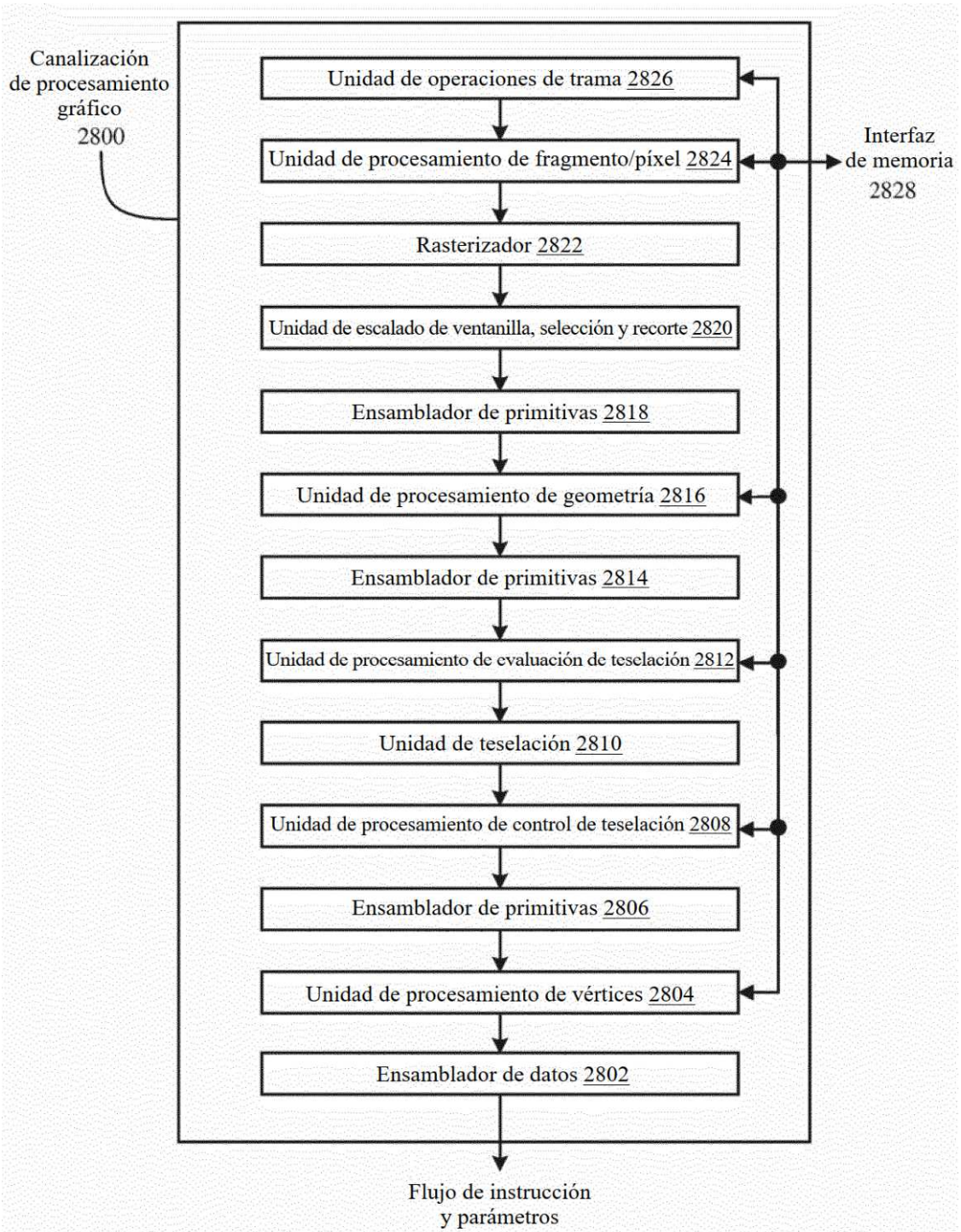

Canalización de procesamiento gráfico 2800
Unidad de operaciones de trama 2826
Unidad de procesamiento de fragmento/píxel 2824
Interfaz de memoria 2828
Rasterizador 2822
Unidad de escalado de ventanilla, selección y recorte 2820
Ensamblador de primitivas 2818
Unidad de procesamiento de geometría 2816
Ensamblador de primitivas 2814
Unidad de procesamiento de evaluación de teselación 2812
Unidad de teselación 2810
Unidad de procesamiento de control de teselación 2808
Ensamblador de primitivas 2806
Unidad de procesamiento de vértices 2804
Ensamblador de datos 2802
Flujo de instrucción y parámetros

FIG. 28