

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 975 950**

51 Int. Cl.:

G06F 12/1027 (2006.01)

G06F 12/1036 (2006.01)

G06F 12/1009 (2006.01)

G06F 12/1081 (2006.01)

G06F 13/16 (2006.01)

G06F 13/40 (2006.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **29.03.2018** **E 22189858 (8)**

97 Fecha y número de publicación de la concesión europea: **03.01.2024** **EP 4109283**

54 Título: **Aparato y método de gestión de memoria en un entorno de procesamiento de gráficos**

30 Prioridad:

07.04.2017 US 201715482690

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

18.07.2024

73 Titular/es:

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

72 Inventor/es:

**COORAY, NIRANJAN L;
APPU, ABHISHEK R;
KOKER, ALTUG;
RAY, JOYDEEP;
VEMBU, BALAJI;
K, PATTABHIRAMAN;
PUFFER, DAVID;
COWPERTHWAITHE, DAVID J;
SANKARAN, RAJESH M.;
SINGH, SATYESHWAR;
KP, SAMEER;
SHAH, ANKUR N. y
TIAN, KUN**

74 Agente/Representante:

LEHMANN NOVO, María Isabel

ES 2 975 950 T3

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

DESCRIPCIÓN

Aparato y método de gestión de memoria en un entorno de procesamiento de gráficos

5 ANTECEDENTES

Campo de la invención

10 Esta invención se refiere en general al campo de los procesadores informáticos. Más particularmente, la invención se refiere a un aparato y a un método de gestión de memoria en un entorno de procesamiento de gráficos.

Descripción de la técnica relacionada

15 Se han producido recientemente avances rápidos en la virtualización de unidades de procesador de gráficos (GPU). Entornos de procesamiento de gráficos 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 unos pocos. Ciertas arquitecturas realizan virtualización de GPU completa a través de captura y emulación para emular una GPU virtual con características completas (vGPU) mientras aún proporcionan un rendimiento cercano a nativo pasando a través de recursos de memoria de gráficos de rendimiento crítico.

20 Con la importancia creciente de las GPU en servidores para soportar 3D, medios y cargas de trabajo de GPGPU, la virtualización de GPU está cada vez más extendida. Cómo virtualizar el acceso a memoria de GPU desde una máquina virtual (VM) es uno de los factores de diseño claves. La GPU tiene su propia memoria de gráficos: o bien memoria de vídeo especializada o bien memoria de sistema compartida. Cuando la memoria de sistema se usa para 25 gráficos, las direcciones físicas de invitado (GPA) necesitan traducirse a direcciones físicas de anfitrión (HPA) antes de que se accedan mediante hardware.

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

35 En algunas implementaciones, las páginas de memoria de invitado/VM se respaldan mediante 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 compartición de memoria, tales como Fusión de Misma Página de Núcleo (KSM).

40 KSM combina páginas de múltiples VM con el mismo contenido, en una única página con protección contra escritura. Es decir, si una página de memoria en VM1 (que se correlaciona desde PA1 de invitado a PA1 de anfitrión), tiene los mismos contenidos que otra página de memoria en VM2 (que se 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 la PA2 de VM2 se correlacionan con HPA_SH 45 con protección contra escritura. Esto ahorra la memoria usada para el sistema, y es particularmente útil para páginas de memoria de solo lectura del invitado, tales como páginas de código y páginas de cero. Con KSM, se usa tecnología de copia en escritura (COW) para eliminar la compartición una vez que una VM modifica el contenido de página.

50 Se usa paso a través mediado en sistemas de virtualización para el rendimiento y compartición de dispositivo, donde una única GPU física se presenta como múltiples GPU virtuales a múltiples invitados con DMA directo, mientras que 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 dispositivo DMA va directamente a la memoria sin intervención del hipervisor.

55 El documento US 2013/276096 A1 divulga un aparato de procesamiento de datos que comprende una CPU, una memoria y una unidad de procesamiento de gráficos (GPU), dentro de la cual se proporcionan núcleos de procesamiento de gráficos que efectúan tareas de procesamiento de gráficos delegadas por la CPU. Una unidad de control de sistema de GPU comprende una MMU maestra para proporcionar una administración central de las traducciones de dirección virtual a física y para administrar el control sobre qué áreas de memoria son accesibles a 60 las tareas que se están efectuando mediante los núcleos de procesamiento de gráficos sobre una base de hilos.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

65 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;

5 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 de gráficos integrado;

10 La **Figura 3** es un diagrama de bloques de una realización de un procesador de gráficos que puede ser una unidad de procesamiento de gráficos discreta, o puede ser un procesador de gráficos integrado con una pluralidad de núcleos de procesamiento;

La **Figura 4** es un diagrama de bloques de una realización de un motor de procesamiento de gráficos para un procesador de gráficos;

15 La **Figura 5** es un diagrama de bloques de otra realización de un procesador de gráficos;

La **Figura 6** es un diagrama de bloques de lógica de ejecución de hilos que incluye una matriz de elementos de procesamiento;

20 La **Figura 7** ilustra un formato de instrucción de unidad de ejecución de procesador de gráficos de acuerdo con una realización;

25 La **Figura 8** es un diagrama de bloques de otra realización de un procesador de gráficos que incluye una tubería de gráficos, una tubería de medios, un motor de visualización, lógica de ejecución de hilos y una tubería de salida de representación;

La **Figura 9A** es un diagrama de bloques que ilustra un formato de comando de procesador de gráficos de acuerdo con una realización;

30 La **Figura 9B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos de acuerdo con una realización;

35 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;

40 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 de gráficos ilustrativo de un sistema en un circuito integrado de chip que puede fabricarse usando uno o más núcleos de IP;

45 La **Figura 14** ilustra un procesador de gráficos 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 de gráficos ilustrativo;

50 La **Figura 16** ilustra una arquitectura ilustrativa para virtualización gráfica completa;

La **Figura 17** ilustra una arquitectura de procesamiento de gráficos virtualizada ilustrativa que incluye unidades de procesamiento de gráficos virtuales (vGPU);

55 La **Figura 18** ilustra una realización de una arquitectura de virtualización con una IOMMU;

La **Figura 19** ilustra una realización en la que se realiza procesamiento de gráficos en un servidor;

60 La **Figura 20** ilustra una realización en la que múltiples cortes de gráficos incluyen circuitería de almacenamiento en memoria intermedia y arbitraje;

La **Figura 21** ilustra múltiples conjuntos de memorias intermedias de acuerdo con una realización;

La **Figura 22** ilustra un método de acuerdo con una realización de la invención;

65 La **Figura 23** ilustra unidades de gestión de memoria maestra y esclava que dan servicio a diferentes conjuntos de

cortes;

La **Figura 24** ilustra un método de acuerdo con una realización de la invención;

5 La **Figura 25** ilustra una realización que usa valores de identificador de espacio de direcciones de proceso (PASID) para abordar un gran número de unidades de procesamiento de gráficos (GPU);

La **Figura 26** ilustra un método de acuerdo con una realización de la invención;

10 La **Figura 27** ilustra una disposición ilustrativa de registros de dirección de base (BAR) de invitado y BAR de anfitrión;

La **Figura 28** ilustra correlaciones ilustrativas desde una entrada de tabla de paginación a un espacio de dirección física de anfitrión;

15 La **Figura 29** compara tablas de traducción de gráficos de un solo nivel con tablas de traducción multinivel; y

La **Figura 30** ilustra una realización en la que a ciertas máquinas virtuales se asignan tablas de traducción de gráficos de un solo nivel y a otras VM se asignan tablas de traducción multinivel.

20 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.

30 ARQUITECTURAS ILUSTRATIVAS DE PROCESADOR DE GRÁFICOS 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 de gráficos 108, y puede ser un sistema de sobremesa de procesador único, un sistema de estación de trabajo de multiprocesador 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 integrados.

Una realización del sistema 100 puede incluir, o incorporarse dentro de, una plataforma de juegos basada en servidor, una consola de juegos, incluyendo una consola de juegos y de 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, un teléfono inteligente, un dispositivo informático de tipo tableta o un 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, un dispositivo de gafas inteligentes, un dispositivo de realidad aumentada o un dispositivo de realidad virtual. En algunas realizaciones, el sistema de procesamiento de datos 100 es un dispositivo de televisión o de decodificador de salón que tiene uno o más procesadores 102 y una interfaz gráfica generada por uno o más procesadores de gráficos 108.

En algunas realizaciones, cada uno de los uno o más procesadores 102 incluye uno o más núcleos de procesador 107 para procesar instrucciones que, cuando se ejecutan, realizan operaciones para software de usuario y sistema. En algunas realizaciones, cada uno de los uno o más núcleos de procesador 107 está configurado para procesar un conjunto de instrucciones 109 específico. En algunas realizaciones, el conjunto de instrucciones 109 puede facilitar el cómputo de conjunto de instrucciones complejo (CISC), el cómputo de conjunto de instrucciones reducido (RISC) o el cómputo a través de una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 107 pueden procesar, cada uno, un conjunto de instrucciones 109 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo 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 la memoria caché 104. Dependiendo de la arquitectura, el procesador 102 puede tener una única caché interna o múltiples niveles de caché interna. En algunas realizaciones, la memoria caché se comparte entre diversos componentes del procesador 102. En algunas realizaciones, el procesador 102 también usa una caché externa (por ejemplo, una caché de nivel 3 (L3) o una caché de último nivel (LLC)) (no mostrada), que se puede compartir entre los núcleos de procesador 107 usando técnicas de coherencia de caché conocidas. Se incluye adicionalmente, en el procesador 102, un archivo de registro 106 que puede incluir

diferentes tipos de registros para almacenar diferentes tipos de datos (por ejemplo, registros de números enteros, registros de coma flotante, registros de estado y un registro de puntero de instrucción). Algunos registros pueden ser registros de propósito general, mientras que otros registros pueden ser específicos del diseño del procesador 102.

En algunas realizaciones, el procesador 102 está acoplado con un bus de procesador 110 para transmitir señales de comunicación tales como señales de dirección, de datos o 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' ilustrativa, incluyendo un concentrador de controlador de memoria 116 y un concentrador de controlador de entrada-salida (E/S) 130. Un concentrador de controlador de memoria 116 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 100, mientras que un concentrador del controlador de E/S (ICH) 130 proporciona conexiones a dispositivos de E/S mediante un bus de E/S local. En una realización, la lógica del concentrador de controlador de memoria 116 está integrada 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), un dispositivo de memoria flash, dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para servir como una memoria de proceso. En una realización, el dispositivo de memoria 120 puede operar como memoria de sistema para el sistema 100, para almacenar datos 122 e 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 de gráficos externo opcional 112, que puede comunicarse con el uno o más procesadores de gráficos 108 en los procesadores 102 para realizar operaciones de gráficos y medios.

En algunas realizaciones, el ICH 130 posibilita que los periféricos se conecten al dispositivo de memoria 120 y al procesador 102 mediante un bus de E/S de alta velocidad. Los periféricos de E/S 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, una unidad de disco duro, memoria flash, etc.), y un controlador de E/S heredado 140 para acoplar dispositivos heredados (por ejemplo, de tipo sistema personal 2 (PS/2)) al sistema. Uno o más controladores de Bus Serie Universal (USB) 142 conectan los dispositivos de entrada, tales como las combinaciones de teclado y ratón 144. Un controlador de red 134 puede acoplarse también con el ICH 130. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus del procesador 110. Se apreciará que el sistema 100 mostrado es ilustrativo y no limitante, ya que pueden usarse también otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por ejemplo, el concentrador del controlador de E/S 130 puede estar integrado 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 estar integrados en un procesador de gráficos externo discreto, tal como el procesador de gráficos 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 de gráficos integrado 208. Esos elementos de la Figura 2 que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en otra parte en el presente documento, pero no están limitados a este tipo. El procesador 200 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 202N representado por los recuadros con línea discontinua. Cada uno de los núcleos de procesador 202A-202N incluye una o más unidades de caché internas 204A-204N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades almacenadas en caché compartidas 206.

Las unidades de caché internas 204A-204N y las unidades de caché compartidas 206 representan una jerarquía de memoria caché dentro del procesador 200. La jerarquía de memoria caché puede incluir al menos un nivel de caché de instrucciones y de datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel medio compartida, tal como una caché de Nivel 2 (L2), de Nivel 3 (L3), de Nivel 4 (L4) o de otros niveles, en donde el nivel más alto de caché antes de la memoria externa se clasifica como LLC. En algunas realizaciones, la lógica de coherencia de caché mantiene la coherencia entre las diversas unidades de caché 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. Las una o más unidades controladoras de bus 216 gestionan un conjunto de buses de periféricos, tales como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 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 externos (no mostrados).

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

regular el estado de potencia de los núcleos de procesador 202A-202N y el procesador de gráficos 208.

En algunas realizaciones, el procesador 200 incluye adicionalmente un procesador de gráficos 208 para ejecutar operaciones de procesamiento de gráficos. En algunas realizaciones, el procesador de gráficos 208 se acopla con el conjunto de unidades de caché compartidas 206 y el núcleo de agente de sistema 210, incluyendo los uno o más controladores de memoria integrados 214. En algunas realizaciones, un controlador de visualización 211 está acoplado con el procesador de gráficos 208 para controlar una salida de procesador de gráficos a una o más pantallas acopladas. En algunas realizaciones, el controlador de visualización 211 puede ser un módulo separado acoplado con el procesador de gráficos a través de al menos una interconexión, o se puede integrar dentro del procesador de gráficos 208 o el 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, se puede usar una unidad de interconexión alternativa, tal como una interconexión de punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador de gráficos 208 se acopla con la interconexión en anillo 212 mediante un enlace de E/S 213.

El enlace de E/S 213 ilustrativo representa al menos una de múltiples variedades de interconexiones de E/S, que incluyen una interconexión de E/S en paquete que facilita la comunicación entre diversos componentes de procesador y un módulo de memoria integrado de alto rendimiento 218, tal como un módulo eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 202A-202N y el procesador de gráficos 208 usan módulos de memoria integrados 218 como una 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 que al menos uno de los otros núcleos ejecuta un subconjunto del primer conjunto de instrucciones o un conjunto de instrucciones diferente. En una realización, los núcleos de procesador 202A-202N son heterogéneos en términos de microarquitectura, en donde uno o más núcleos que tienen un consumo de energía relativamente superior se acoplan con uno o más núcleos de potencia que tienen un consumo de energía inferior. Adicionalmente, el procesador 200 se puede implementar en uno o más chips o como un circuito integrado de SoC que tiene los componentes ilustrados, además de otros componentes.

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

En algunas realizaciones, el procesador de gráficos 300 también incluye un controlador de visualización 302 para controlar unos datos de salida de visualización a un dispositivo de visualización 320. El controlador de visualización 302 incluye hardware para uno o más planos de superposición para la visualización y composición de múltiples capas de elementos de interfaz de usuario o de vídeo. En algunas realizaciones, el procesador de gráficos 300 incluye un motor de códec de vídeo 306 para codificar, descodificar o transcodificar medios a, desde o entre uno o más formatos de codificación de medios, incluyendo, pero sin limitación, formatos del Grupo de Expertos en Imágenes en Movimiento (MPEG) tales como MPEG-2, formatos de Codificación de Vídeo Avanzada (AVC) tales como H.264/MPEG-4 AVC, así como de la Sociedad de Ingenieros de Imágenes en Movimiento y de Televisión (SMPTE) 421M/VC-1 y formatos del Grupo Conjunto de Expertos en Fotografía (JPEG) tales como los formatos JPEG y Motion JPEG (MJPEG).

En algunas realizaciones, el procesador de gráficos 300 incluye un motor de transferencia de imágenes en bloque (BLIT) 304 para realizar operaciones de rasterizador bidimensionales (2D), incluyendo, por ejemplo, transferencias de bloque de frontera de bits. Sin embargo, en una realización, se realizan operaciones de gráficos 2D usando uno o más componentes del motor de procesamiento de gráficos (GPE) 310. En algunas realizaciones, el GPE 310 es un motor de cómputo para realizar operaciones de gráficos, incluyendo operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 310 incluye una tubería de 3D 312 para realizar operaciones 3D, tales como representar imágenes y escenas tridimensionales usando funciones de procesamiento que actúan sobre formas de primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La tubería 3D 312 incluye elementos de función programables y fijos que realizan diversas tareas dentro del elemento y/o abarcan hilos de ejecución a un subsistema 3D/de medios 315. Aunque la tubería de 3D 312 puede usarse para realizar operaciones de medios, una realización del GPE 310 también incluye una tubería de medios 316 que se usa específicamente para realizar operaciones de

medios, tales como posprocesamiento de vídeo y potenciación de imagen.

En algunas realizaciones, la tubería de medios 316 incluye unidades de lógica programable o de función fija para realizar una o más operaciones de medios especializadas, tales como aceleración de descodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar o en nombre del motor de códec de vídeo 306. En algunas realizaciones, la tubería de medios 316 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema de 3D/de medios 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 3D/de medios 315.

En algunas realizaciones, el subsistema de 3D/de medios 315 incluye lógica para ejecutar hilos generados por la tubería de 3D 312 y la tubería de medios 316. En una realización, las canalizaciones envían solicitudes de ejecución de hilos al subsistema de 3D/de medios 315, incluyendo lógica de despacho de hilos para arbitrar y despachar las diversas solicitudes a recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos de 3D y de medios. En algunas realizaciones, el subsistema de 3D/de medios 315 incluye una o más cachés internas para datos e instrucciones de hilo. En algunas realizaciones, el subsistema también incluye memoria compartida, incluyendo registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Motor de procesamiento de gráficos

La **Figura 4** es un diagrama de bloques de un motor de procesamiento de gráficos 410 de un procesador de gráficos de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento de gráficos (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 el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa. Por ejemplo, se ilustran la tubería 3D 312 y la tubería de medios 316 de la Figura 3. La tubería 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 de imágenes separado se acopla al GPE 410.

En algunas realizaciones, el GPE 410 se acopla con o incluye un emisor por flujo continuo de comando 403, que proporciona un flujo de comandos a la tubería 3D 312 y/o a las tuberías de medios 316. En algunas realizaciones, el emisor por flujo continuo de comando 403 está acoplado con memoria, que puede ser memoria de sistema, o una o más de memoria caché interna y memoria caché compartida. En algunas realizaciones, el emisor por flujo continuo de comando 403 recibe comandos desde la memoria y envía los comandos a la tubería 3D 312 y/o a la tubería de medios 316. Los comandos son directivas extraídas desde una memoria intermedia en anillo, que almacena comandos para la tubería 3D 312 y la tubería de medios 316. En una realización, la memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la tubería de 3D 312 también pueden incluir referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértice y de geometría para la tubería de 3D 312 y/o datos de imagen y objetos de memoria para la tubería de medios 316. La tubería de 3D 312 y la tubería de medios 316 procesan los comandos y datos realizando operaciones a través de lógica dentro de las tuberías respectivas o despachando uno o más hilos de ejecución a una matriz de núcleo de gráficos 414.

En diversas realizaciones, la tubería de 3D 312 puede ejecutar uno o más programas de sombreado, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculo u otros programas de sombreado, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleo de gráficos 414. La matriz de núcleo de gráficos 414 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución de múltiples propósitos (por ejemplo, unidades de ejecución) dentro de la matriz de núcleo de gráficos 414 incluye soporte para diversos lenguajes de sombreador de API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

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

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleo de gráficos 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 se puede usar para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleo de gráficos 414. En algunas realizaciones, la URB 418 se puede usar adicionalmente para la sincronización entre hilos en la matriz de núcleo de gráficos y la lógica de función fija dentro de la lógica de funciones compartidas 420.

En algunas realizaciones, la matriz de núcleo de gráficos 414 es escalable, de manera que la matriz incluye un número variable de núcleos de gráficos, teniendo cada uno un número variable de unidades de ejecución basándose en la potencia objetivo y en el nivel de rendimiento del GPE 410. En una realización, los recursos de ejecución son dinámicamente escalables, de manera que los recursos de ejecución se pueden habilitar o deshabilitar según sea necesario.

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

La **Figura 5** es un diagrama de bloques de otra realización de un procesador de gráficos 500. Los elementos de la Figura 5 que tienen los mismos números de referencia (o nombres) que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

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

En algunas realizaciones, el procesador de gráficos 500 recibe lotes de comandos a través de la interconexión en anillo 502. Los comandos entrantes son interpretados por un emisor por flujo continuo de comando 503 en el extremo frontal de tubería 504. En algunas realizaciones, el procesador de gráficos 500 incluye lógica de ejecución ajustable a escala para realizar un procesamiento de geometría 3D y un procesamiento de medios a través del núcleo o núcleos de gráficos 580A-580N. Para los comandos de procesamiento de geometría 3D, el emisor por flujo continuo de comando 503 suministra comandos a la tubería de geometría 536. Para al menos algunos comandos de procesamiento de medios, el emisor por flujo continuo de comando 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 posprocesamiento de vídeo y de imagen y un motor de codificación/decodificación multiformato (MFX) 533 para proporcionar la codificación y decodificación de datos de medios acelerados por hardware. En algunas realizaciones, la tubería de geometría 536 y el motor de medios 537 cada uno genera hilos de ejecución para los recursos de ejecución de hilo proporcionados por al menos un núcleo de gráficos 580A.

En algunas realizaciones, el procesador de gráficos 500 incluye recursos de ejecución de hilo escalables que presentan núcleos modulares 580A-580N (en ocasiones denominados cortes de núcleo), teniendo cada uno múltiples subnúcleos 550A-550N, 560A-560N (en ocasiones denominados subcortes de núcleo). En algunas realizaciones, el procesador de gráficos 500 puede tener cualquier número de núcleos de gráficos 580A a 580N. En algunas realizaciones, el procesador de gráficos 500 incluye un núcleo de gráficos 580A que tiene al menos un primer subnúcleo 550A y un segundo subnúcleo 560A. En otras realizaciones, el procesador de gráficos es un procesador de baja potencia con un único subnúcleo (por ejemplo, 550A). En algunas realizaciones, el procesador de gráficos 500 incluye múltiples núcleos de 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/texturas 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. También pueden incluirse otros recursos compartidos en las diversas realizaciones del procesador de gráficos.

Unidades de ejecución

La **Figura 6** ilustra la 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 el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero no se limitan a tal cosa.

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

En algunas realizaciones, las unidades de ejecución 608A-608N se usan principalmente para ejecutar programas sombreadores. Un procesador del sombreador 602 puede procesar los diversos programas sombreadores y despachar hilos de ejecución asociados con los programas sombreadores mediante un despachador de hilo 604. En una realización, el despachador de hilos incluye lógica para arbitrar solicitudes de iniciación de hilo desde las tuberías de gráficos y de medios e instanciar los hilos solicitados en una o más unidades de ejecución en las unidades de ejecución 608A-608N. Por ejemplo, la tubería de geometría (por ejemplo, 536 de la Figura 5) puede despachar sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilos 600 (la Figura 6) para su procesamiento. En algunas realizaciones, el despachador de hilos 604 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas de sombreado en ejecución.

En algunas realizaciones, las unidades de ejecución 608A-608N soportan un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreador de gráficos 3D convencionales, de manera que programas de sombreado desde bibliotecas de gráficos (por ejemplo, Direct 3D y OpenGL) se ejecutan con una traducción mínima. Las unidades de ejecución soportan un procesamiento de vértices y de geometría (por ejemplo, programas de vértices, programas de geometría, sombreadores de vértices), un procesamiento de píxeles (por ejemplo, sombreadores de píxeles, sombreadores de fragmentos) y un procesamiento de propósito general (por ejemplo, sombreadores de cómputo y de medios). Cada una de las unidades de ejecución 608A-608N es capaz de múltiples emisiones de una ejecución de una única instrucción - múltiples datos (SIMD), y un funcionamiento de múltiples hilos habilita un entorno de ejecución eficiente frente a accesos de memoria de latencia superior. Cada hilo de hardware dentro de cada unidad de ejecución tiene un archivo de registro de ancho de banda alto dedicado y un estado de hilo independiente asociado. La ejecución es de múltiples emisiones por reloj a tuberías capaces de realizar operaciones de números enteros, de coma flotante de precisión sencilla y doble, capacidad de bifurcación de SIMD, operaciones lógicas, operaciones trascendentales y otras operaciones misceláneas. Mientras se esperan datos desde memoria o una de las funciones compartidas, una lógica de dependencia dentro de las unidades de ejecución 608A-608N hace que un hilo en espera pase a estar inactivo hasta que se hayan devuelto los datos solicitados. Mientras el hilo en espera está inactivo, se pueden dedicar recursos de hardware a procesar otros hilos. Por ejemplo, durante un retardo asociado con una operación de sombreador de vértices, una unidad de ejecución puede realizar operaciones para un sombreador de píxeles, un sombreador de fragmentos u otro tipo de programa de sombreado, incluyendo un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 608A-608N opera sobre matrices de elementos de datos. El número de elementos de datos es el "tamaño de ejecución" o el número de canales para la instrucción. Un canal de ejecución es una unidad lógica de ejecución para el acceso, enmascaramiento y control de flujo de elementos de datos dentro de las instrucciones. El número de canales puede ser independiente del número de Unidades Aritméticas Lógicas (ALU) o Unidades de Coma Flotante (FPU) físicas para un procesador de gráficos particular. En algunas realizaciones, las unidades de ejecución 608A-608N soportan tipos de datos de números enteros y de coma flotante.

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

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

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

Durante la ejecución, los gráficos y las tuberías de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilo 600 mediante lógica de generación y despacho de hilo. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos en datos de píxeles, se invoca la lógica de procesador de píxel (por ejemplo, lógica de sombreador de píxel, lógica de sombreador de fragmento, etc.) dentro del procesador del sombreador 602 para que calcule adicionalmente información de salida y haga que se escriban los resultados en las superficies de salida (por ejemplo, memorias intermedias de color, memorias intermedias de profundidad, memorias intermedias de estarcido, etc.). En algunas realizaciones, un sombreador de píxeles o un sombreador de fragmentos calcula los valores de los diversos atributos de vértice que se van a interpolar a lo largo del objeto rasterizado. En algunas realizaciones, una lógica de procesador de píxeles dentro del procesador de sombreado 602 ejecuta entonces un programa de sombreado de píxeles o de fragmentos suministrado por interfaz de programación de aplicaciones (API). Para ejecutar el programa de sombreado, el procesador de sombreado 602 despacha hilos a una unidad de ejecución (por ejemplo, 608A) a través del despachador de hilos 604. En algunas realizaciones, el sombreador de píxeles 602 usa una lógica de muestreo de textura en el muestreador 610 para acceder a datos de textura en correlaciones de textura almacenadas en memoria. Operaciones aritméticas sobre los datos de textura y los datos de geometría de entrada computan datos de color de píxel para cada fragmento geométrico, o descartan el procesamiento adicional de uno o más píxeles.

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

La Figura 7 es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos 700 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución de procesador de gráficos soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los cuadros con línea continua ilustran los componentes que se incluyen, en general, en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de las instrucciones. En algunas realizaciones, el formato de instrucción 700 descrito e ilustrado son macro-instrucciones, en el sentido de que las mismas son instrucciones suministradas a la unidad de ejecución, en contraposición a micro-operaciones resultantes de la decodificación de instrucciones una vez que se ha procesado la instrucción.

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

Para cada formato, el código de operación de instrucción 712 define la operación que ha de realizar la unidad de ejecución. Las unidades de ejecución ejecutan cada instrucción en paralelo a lo largo de los múltiples elementos de datos de cada operando. Por ejemplo, en respuesta a una instrucción de suma, la unidad de ejecución realiza una operación de suma simultánea a lo largo de cada canal de color que representa un elemento de textura o un elemento de imagen. Por defecto, la unidad de ejecución ejecuta cada instrucción a lo largo de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucción 714 habilita el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicación) y el orden de canal de datos (por ejemplo, referenciación). Para instrucciones en el formato de instrucción de 128 bits 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, incluyendo dos operandos de origen, src0 720, src1 722 y un destino 718. En algunas realizaciones, las unidades de ejecución soportan instrucciones de destino dual, en donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 724), en donde el código de operación 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 manera 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 el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se usa el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos es proporcionada directamente por bits en la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 710 incluye un campo de modo de dirección/acceso 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 de datos para la instrucción. Algunas realizaciones soportan modos de acceso que incluyen un modo de acceso alineado de 16 bytes y un modo de acceso alineado de 1 byte, en donde la alineación de bytes del modo de acceso determina la alineación de acceso de los operandos de instrucción. Por ejemplo, cuando está en un primer modo, la instrucción puede usar un direccionamiento alineado por byte para los operandos de origen y de destino y, cuando está en un segundo modo, la instrucción puede usar un direccionamiento alineado por 16 bytes para todos los operandos de origen y de destino.

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

En algunas realizaciones, las instrucciones se agrupan basándose en los campos de bits del código de operación 712 para simplificar la decodificación de código de operación 740. Para un código de operación de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de código de operación. La agrupación de código de operación precisa mostrada es simplemente un ejemplo. En algunas realizaciones, un grupo de código de operación de movimiento y de lógica 742 incluye instrucciones de movimiento y de lógica de datos (por ejemplo, mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y de lógica 742 comparte los cinco bits más significativos (MSB), en donde las instrucciones de movimiento (mov) están en forma de 0000xxxxb y las instrucciones de lógica están en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 744 (por ejemplo, llamada, salto (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, espera, envío) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cálculo matemático paralelo 748 incluye instrucciones aritméticas a nivel de componente (por ejemplo, suma, multiplicación (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cálculo matemático paralelo 748 realiza las operaciones aritméticas en paralelo a lo largo de canales de datos. El grupo de cálculo matemático vectorial 750 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cálculo matemático vectorial realiza aritmética tal como cálculos de producto escalar sobre operandos de vectores.

Tubería de gráficos

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

En algunas realizaciones, el procesador de gráficos 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 de gráficos 800 es un procesador de gráficos dentro de un sistema de procesamiento de múltiples núcleos que incluye uno o más núcleos de procesamiento de propósito general. El procesador de gráficos se controla por escrituras de registro en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 800 mediante una interconexión en anillo 802. En algunas realizaciones, la interconexión en anillo 802 acopla el procesador de gráficos 800 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de fin general. Los comandos desde la interconexión en anillo 802 son interpretados por un emisor por flujo continuo de comando 803, que suministra instrucciones a componentes individuales de la canalización de gráficos 820 o la canalización de medios 830.

En algunas realizaciones, el emisor por flujo continuo de comando 803 dirige el funcionamiento de un extractor de vértices 805 que lee datos de vértice desde memoria y ejecuta comandos de procesamiento de vértices proporcionados por el emisor por flujo continuo de comando 803. En algunas realizaciones, el extractor de vértices 805 proporciona datos de vértice a un sombreador de vértices 807, que realiza operaciones de transformación y de

iluminación de espacio de coordenadas en cada vértice. En algunas realizaciones, el extractor de vértices 805 y el sombreador de vértices 807 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a las unidades de ejecución 852A-852B a través de un despachador de hilos 831.

5 En algunas realizaciones, las unidades de ejecución 852A-852B son una matriz de procesadores de vectores que tienen un conjunto de instrucciones para realizar operaciones de gráficos y de medios. En algunas realizaciones, las unidades de ejecución 852A-852B tienen una caché de L1 851 anexada que es específica para cada matriz o que se comparte entre las matrices. La caché se puede configurar como una caché de datos, una caché de instrucciones o una única caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.

10 En algunas realizaciones, la canalización de gráficos 820 incluye componentes de teselación para realizar una 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 una evaluación de extremo trasero de la salida de teselación. Un teselador 813 opera en la dirección del sombreador de casco 811 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados basándose en un modelo geométrico aproximado que se proporciona como entrada a la canalización de gráficos 820. En algunas realizaciones, si no se usa la teselación, se pueden sortear los componentes de teselación (por ejemplo, el sombreador de casco 811, el teselador 813 y el sombreador de dominio 817).

20 En algunas realizaciones, pueden procesarse los objetos geométricos completos por un sombreador de geometría 819 mediante uno o más hilos despachados a unidades de ejecución 852A-852B o puede continuarse directamente al recortador 829. En algunas realizaciones, el sombreador de geometría opera sobre objetos geométricos enteros, en lugar de vértices o parches de vértices como en fases previas de la tubería de gráficos. Si la teselación está inhabilitada, el sombreador de geometría 819 recibe una entrada desde el sombreador de vértices 807. En algunas realizaciones, el sombreador de geometría 819 se puede programar mediante un programa de sombreado de geometría para realizar una teselación de geometría si las unidades de teselación están inhabilitadas.

30 Antes de la 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 de sombreador de geometría. En algunas realizaciones, un componente de prueba de rasterizador y de profundidad 873 en la canalización de salida de representación 870 despacha sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunas realizaciones, la lógica de sombreado de píxeles se incluye en la lógica de ejecución de hilos 850. En algunas realizaciones, una aplicación puede sortear el componente de prueba de rasterizador y de profundidad 873 y acceder a datos de vértice sin rasterizar a través de una unidad de salida de flujo 823.

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

45 En algunas realizaciones, la tubería de salida de representación 870 contiene un componente de prueba de rasterizador y de profundidad 873 que convierte objetos basados en vértices en una representación basada en píxeles asociada. En algunas realizaciones, la lógica de rasterizador incluye una unidad generadora de ventanas/enmascaradora para realizar una rasterización de líneas y de triángulos de función fija. Una caché de representación 878 y una caché de profundidad 879 asociadas también están disponibles en algunas realizaciones. Un componente de operaciones de píxel 877 realiza operaciones basadas en píxeles sobre los datos, aunque, en algunas instancias, las operaciones de píxel asociadas con operaciones 2D (por ejemplo, transferencias de imagen de bloque de bits con mezcla) son realizadas por el motor 2D 841, o son sustituidas en el momento de la visualización por el controlador de visualización 843 usando planos de visualización de superposición. En algunas realizaciones, está disponible una caché de L3 compartida 875 para todos los componentes de gráficos, permitiendo la compartición de datos sin el uso de memoria de sistema principal.

60 En algunas realizaciones, la canalización de medios del procesador de gráficos 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 emisor por flujo continuo de comando 803. En algunas realizaciones, la canalización de medios 830 incluye un emisor por flujo continuo de comando 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 funcionalidad de generación de hilos para abarcar hilos para despachar a la lógica de ejecución de hilo 850 mediante el despachador de hilo 831.

65 En algunas realizaciones, el procesador de gráficos 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 de gráficos

a través de la interconexión en 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 una lógica de propósito especial capaz de funcionar independientemente de la canalización de 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 anexo a través de un conector de dispositivo de visualización.

En algunas realizaciones, la canalización de gráficos 820 y la canalización de medios 830 se pueden configurar para realizar operaciones basándose en múltiples interfaces de programación de gráficos y de medios y no son específicas de ninguna interfaz de programación de aplicaciones (API) concreta. En algunas realizaciones, software de controlador para el procesador de gráficos traduce llamadas de API que son específicas de una biblioteca de medios o de gráficos particular a comandos que pueden ser procesados por el procesador de gráficos. En algunas realizaciones, se proporciona soporte para la Biblioteca de Gráficos Abierta (OpenGL), Lenguaje Informático Abierto (OpenCL) y/o API de gráficos y de cómputo Vulkan, todas ellas del grupo Khronos. En algunas realizaciones, también se puede proporcionar soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, se puede soportar una combinación de estas bibliotecas. También se puede proporcionar soporte para la Biblioteca de Visión por Ordenador de Código Abierto (OpenCV). También se soportaría una API futura con una tubería de 3D compatible si se puede hacer una correlación desde la tubería de la API futura a la tubería del procesador de gráficos.

Programación de tubería de gráficos

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

En algunas realizaciones, el cliente 902 especifica la unidad de cliente del dispositivo de gráficos que procesa los datos de comando. En algunas realizaciones, un analizador de comandos de procesador de gráficos examina el campo de cliente de cada comando para acondicionar el procesamiento adicional del comando y encaminar los datos de comando a la unidad de cliente apropiada. En algunas realizaciones, las unidades de cliente de procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de representación, una unidad de 2D, una unidad de 3D y una unidad de medios. Cada unidad de cliente tiene una tubería de procesamiento correspondiente que procesa los comandos. Una vez que el comando ha sido recibido por la unidad de cliente, la unidad de cliente lee el código de operación 904 y, si está presente, el subcódigo de operación 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 código de operación de comando. En algunas realizaciones, los comandos se alinean a través de múltiplos de una palabra doble.

El diagrama de flujo en la Figura 9B muestra una secuencia de comandos de procesador de gráficos 910 ilustrativo. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que cuenta con una realización de un procesador de gráficos usa una versión de la secuencia de comandos mostrada para establecer, ejecutar y terminar un conjunto de operaciones de gráficos. Se muestra y se describe una secuencia de comandos de muestra solo con fines de ejemplo, debido a que las realizaciones no se limitan a estos comandos específicos o a esta secuencia de comandos. Además, los comandos se pueden emitir como un lote de comandos en una secuencia de comandos, de manera que el procesador de gráficos procesará la secuencia de comandos de manera al menos parcialmente concurrente.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 910 puede comenzar con un comando de vaciado de canalización 912 para hacer que cualquier canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunas realizaciones, la canalización de 3D 922 y la canalización de medios 924 no funcionan de manera concurrente. El vaciado de tubería se realiza para hacer que la tubería de gráficos activa complete cualquier comando pendiente. En respuesta a un vaciado de tubería, el analizador de comandos para el procesador de gráficos pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. Opcionalmente, cualquier dato en la caché de representación que se marque como 'sucio' se puede vaciar a memoria. En algunas realizaciones, el comando de vaciado de canalización 912 se puede usar para la sincronización de canalización o antes de poner el procesador de gráficos en un estado de baja potencia.

En algunas realizaciones, se usa un comando de selección de canalización 913 cuando una secuencia de comandos requiere que el procesador de gráficos conmute explícitamente entre canalizaciones. En algunas realizaciones, se requiere un comando de selección de canalización 913 solo una vez dentro de un contexto de ejecución antes de emitir comandos de canalización, a menos que el contexto sea emitir comandos para ambas canalizaciones. En algunas realizaciones, se requiere un comando de vaciado de tubería 912 inmediatamente antes de una conmutación de tubería a través del comando de selección de tubería 913.

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

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

Los comandos restantes en la secuencia de comandos difieren basándose en la tubería activa para las operaciones. Basándose en una determinación de tubería 920 la secuencia de comandos está adaptada a la tubería 3D 922 que comienza con el estado de tubería 3D 930 o la tubería de medios 924 que comienza en el estado de la tubería de medios 940.

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

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

En algunas realizaciones, la canalización de 3D 922 se desencadena a través de un comando o evento de la ejecución 934. En algunas realizaciones, una escritura de registro desencadena una ejecución de comando. En algunas realizaciones, la ejecución se desencadena a través de un comando 'ir' o 'poner en marcha' en la secuencia de comandos. En una realización, la ejecución de comando se desencadena usando un comando de sincronización de tubería para vaciar la secuencia de comandos a través de la tubería de gráficos. La tubería de 3D realizará un procesamiento de geometría para las primitivas 3D. Una vez que se han completado las operaciones, los objetos geométricos resultantes se rasterizan y el motor de píxeles da color a los píxeles resultantes. También se pueden incluir comandos adicionales para controlar el sombreado de píxeles y las operaciones de extremo trasero de píxeles para esas operaciones.

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

En algunas realizaciones, la canalización de medios 924 se configura de una manera similar a la de la canalización de 3D 922. Un conjunto de comandos para configurar el estado de tubería de medios 940 se despachan o se colocan en una cola de comandos antes de los comandos de objeto de medios 942. En algunas realizaciones, los comandos para el estado de tubería de medios 940 incluyen datos para configurar los elementos de tubería de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de descodificación de vídeo y de codificación de vídeo dentro de la tubería de medios, tal como el formato de codificación o de descodificación. En algunas realizaciones, los comandos para el estado de tubería 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 tubería de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo a procesar. En algunas realizaciones, todos los estados de canalización de medios han de ser válidos antes de emitir un comando de objeto de medios 942. Una vez que se ha configurado el estado de canalización y los comandos de objeto de medios 942 se han puesto 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, una escritura de registro). La salida desde la canalización de medios 924 se puede posprocesar a continuación mediante operaciones proporcionadas por la canalización de 3D 922 o la canalización de medios 924. En algunas realizaciones, las operaciones de GPGPU se configuran y se ejecutan de una manera similar a la de las operaciones de medios.

Arquitectura de software de gráficos

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

En algunas realizaciones, la aplicación de gráficos 3D 1010 contiene uno o más programas de sombreado que incluyen las instrucciones de sombreador 1012. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, tal como el lenguaje de sombreador de alto nivel (HLSL) o el lenguaje de sombreador de OpenGL (GLSL). La aplicación también incluye las instrucciones ejecutables 1014 en un lenguaje máquina adecuado para su ejecución por el núcleo de procesador de propósito general 1034. La aplicación también incluye los objetos de 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 de propiedad exclusiva o un sistema operativo de tipo UNIX de código abierto que usa una variante del núcleo de Linux. El sistema operativo 1020 puede soportar una API de gráficos 1022 tal como la API de Direct3D, la API de OpenGL o la API de Vulkan. Cuando está en uso la API de Direct3D, el sistema operativo 1020 usa un compilador de sombreador de extremo frontal 1024 para compilar cualquier instrucción de sombreador 1012 en HLSL a un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede realizar una precompilación de sombreador. En algunas realizaciones, sombreadores de alto nivel se compilan a sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 1010. En algunas realizaciones, las instrucciones de sombreador 1012 se proporcionan en una forma intermedia, tal como una versión de la representación intermedia portátil convencional (SPIR) usada por la API de Vulkan.

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

Implementaciones de núcleo de IP

Uno o más aspectos de al menos una realización se pueden implementar mediante un código representativo almacenado en un medio legible por máquina que representa y/o define una lógica dentro de un circuito integrado tal como un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan una lógica diversa dentro del procesador. Cuando son leídas por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para realizar las técnicas descritas en el presente documento. Tales representaciones,

conocidas como "núcleos de IP", son unidades reutilizables de lógica para un circuito integrado que se pueden almacenar en un medio legible por máquina tangible como un modelo de hardware que describe la estructura del circuito integrado. El modelo de hardware se puede suministrar a diversos clientes o instalaciones de fabricación, que cargan el modelo de hardware en máquinas de fabricación que fabrican el circuito integrado. El circuito integrado se puede fabricar de manera que el circuito realiza operaciones descritas en asociación con cualquiera de las realizaciones descritas en el presente documento.

La **Figura 11** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 1100 que se puede usar para fabricar un circuito integrado para realizar operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo de IP 1100 se puede usar para generar diseños reutilizables modulares que se pueden incorporar en un diseño más grande o usarse para construir todo un circuito integrado (por ejemplo, un circuito integrado de SoC). Una instalación de diseño 1130 puede generar una simulación por software 1110 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). El software de simulación 1110 se puede usar para diseñar, someter a prueba 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. A continuación, puede crearse un diseño de nivel de transferencia de registro (RTL) 1115 o sintetizarse a partir del modelo de simulación 1112. El diseño RTL 1115 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registro de hardware, que incluye la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño RTL 1115, pueden crearse, diseñarse, o sintetizarse también diseños de nivel inferior al nivel de lógica o al nivel de transistores. Por lo tanto, los detalles particulares del diseño y simulación inicial pueden variar.

El diseño de RTL 1115 o equivalente puede sintetizarse adicionalmente por la instalación de diseño en un modelo de hardware 1120, que puede ser en un lenguaje de descripción de hardware (HDL) o alguna otra representación de datos de diseño físico. El HDL se puede simular o someter a prueba adicionalmente para verificar el diseño de núcleo de IP. El diseño de núcleo de IP se puede almacenar para su entrega a una instalación de fabricación de terceros 1165 usando la 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 se puede transmitir (por ejemplo, a través de Internet) a través de una conexión cableada 1150 o una conexión inalámbrica 1160. La instalación de fabricación 1165 puede fabricar a continuación un circuito integrado que está basado al menos en parte en el diseño de núcleo de IP. El circuito integrado fabricado se puede configurar para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Circuito integrado de sistema en un chip ilustrativo

Las **Figuras 12-14** ilustran circuitos integrados ilustrativos y procesadores de gráficos asociados que se pueden fabricar usando uno o más núcleos de IP, de acuerdo con diversas realizaciones descritas en el presente documento. Además de lo que se ilustra, se pueden incluir otros circuitos y lógica, incluyendo procesadores/núcleos de gráficos adicionales, controladores de interfaz de periféricos o núcleos de procesador de propósito general.

La **Figura 12** es un diagrama de bloques que ilustra un circuito integrado de sistema en un chip ilustrativo 1200 que se puede fabricar usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 1200 ilustrativo incluye uno o más procesadores de aplicaciones 1205 (por ejemplo, unas CPU), al menos un procesador de gráficos 1210, y puede incluir adicionalmente un procesador de imágenes 1215 y/o un procesador de vídeo 1220, cualquiera de los cuales puede ser un núcleo de IP modular desde las mismas o múltiples instalaciones de diseño diferentes. El circuito integrado 1200 incluye lógica de bus o de periféricos 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 industria móvil (MIPI) 1255. El almacenamiento puede proporcionarse 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 el acceso a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 1270.

La **Figura 13** es un diagrama de bloques que ilustra un procesador de gráficos 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 de gráficos 1310 puede ser una variante del procesador de gráficos 1210 de la Figura 12. El procesador de gráficos 1310 incluye un procesador de vértices 1305 y uno o más procesadores de fragmentos 1315A-1315N (por ejemplo, 1315A, 1315B, 1315C, 1315D a 1315N-1 y 1315N). El procesador de gráficos 1310 puede ejecutar diferentes programas de sombreado a través de lógica separada, de manera que el procesador de vértices 1305 se optimiza para ejecutar operaciones para programas de sombreado de vértices, mientras que los uno o más procesadores de fragmentos 1315A-1315N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas de sombreado de fragmentos o de píxeles. El procesador de vértices 1305 realiza la fase de procesamiento de vértices de la tubería de gráficos 3D y genera primitivas y datos de vértice. El procesador o procesadores de fragmentos 1315A-1315N usan los datos de primitiva y de vértice generados por el procesador de vértices 1305 para producir una memoria intermedia de tramas 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 sombreado de fragmentos de acuerdo con lo previsto en la API de OpenGL, que se pueden usar para realizar operaciones similares como un programa de sombreado de píxeles de acuerdo con lo previsto en la API de Direct 3D.

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

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

El procesador de gráficos 1410 incluye uno o más núcleos de sombreador 1415A-1415N (por ejemplo, 1415A, 1415B, 1415C, 1415D, 1415E, 1415F a 1315N-1 y 1315N), lo que proporciona una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo o núcleo puede ejecutar todos los tipos de código de sombreado programable, incluyendo código de programa de sombreado para implementar sombreadores de vértices, sombreadores de fragmentos y/o sombreadores de cálculo. El número exacto de núcleos de sombreador presentes puede variar entre realizaciones e implementaciones. Adicionalmente, el procesador de gráficos 1410 incluye un gestor de tareas entre núcleos 1405, que actúa como un despachador de hilos para despachar hilos de ejecución a uno o más núcleo o núcleos de sombreador 1415A-1415N y una unidad de teselación 1418 para acelerar operaciones de teselación para una representación basada en teselas, en la que las operaciones de representación para una escena se subdividen en el espacio de imágenes, por ejemplo, para aprovechar la coherencia espacial local dentro de una escena o para optimizar el uso de cachés internas.

ARQUITECTURAS DE VIRTUALIZACIÓN DE GRÁFICOS ILUSTRATIVAS

Algunas realizaciones de la invención se implementan en una plataforma que utiliza una virtualización de unidad de procesador de gráficos (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 basada en patrones.

Una realización de la invención emplea un entorno de virtualización de GPU completa que ejecuta un controlador de gráficos nativo en el invitado, y paso a través mediado que consigue tanto buen rendimiento, escalabilidad como aislamiento seguro entre invitados. Esta realización presenta una GPU completa virtual a cada máquina virtual (VM) que puede acceder directamente a recursos críticos para el rendimiento sin intervención del hipervisor en la mayoría de los casos, mientras que las operaciones privilegiadas desde el invitado se capturan y emulan con coste mínimo. En una realización, una GPU virtual (vGPU), con características de GPU completa, se presenta a cada VM. Las VM pueden acceder directamente a recursos críticos para el rendimiento, sin intervención del hipervisor en la mayoría de los casos, mientras que 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 de gráficos (GPU) 1500, una unidad central de procesamiento (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 de una memoria intermedia de comandos 1512 en memoria de sistema 1510, para acelerar representación de gráficos usando diversas características diferentes. El motor de visualización 1504 busca datos de píxeles de la memoria intermedia de fotogramas 1514 y, a continuación, envía los datos de píxeles a monitores externos para su visualización.

Ciertas arquitecturas usan una memoria de sistema 1510 como memoria de gráficos, mientras que otras GPU pueden usar memoria en chip. La memoria de sistema 1510 puede correlacionarse con múltiples espacios de dirección virtual mediante tablas de paginación de GPU 1506. Un espacio de direcciones virtual global de 2 GB, denominado memoria de gráficos global, accesible tanto desde la GPU 1500 como la CPU 1520, se correlaciona a

través de tablas de paginación globales. Los espacios de memoria de gráficos locales se soportan en forma de múltiples espacios de dirección virtual locales de 2 GB, aunque están limitados únicamente para acceder desde el motor de representación 1502, a través de tablas de paginación locales. La memoria de gráficos global es principalmente la memoria intermedia de fotogramas 1514, pero también sirve como la memoria intermedia de comandos 1512. Se hacen muchos accesos de datos a memoria de gráficos local cuando la aceleración de hardware está en progreso. Las GPU con memoria en chip emplean mecanismos de tabla de paginación similares.

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 nivel alto como OpenGL y DirectX. La GPU 1500, a continuación, busca y ejecuta los comandos. La memoria intermedia primaria, una memoria intermedia en anillo, puede encadenar juntas otras memorias intermedias por lotes. Los términos "memoria intermedia primaria" y "memoria intermedia en 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. Se usa una tupla de registro (cabeza, cola) para controlar la memoria intermedia en 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 una GPU completa para ejecutar un controlador de gráficos nativo dentro de una VM. El desafío, sin embargo, es significativo 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 frente a ~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 de gráficos de VM1 1631, la GPU puede estar accediendo a la memoria de gráficos 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 en 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 adelantada (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 en anillo especializada para transportar los comandos de GPU adicionales. La memoria intermedia en anillo de invitado (auditada) puede reutilizarse para rendimiento, pero no es seguro insertar directamente los comandos en la memoria intermedia en 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 en anillo de invitado a su propia memoria intermedia en anillo especializada. Al final de la conmutación de contexto, esta realización conmuta desde la memoria intermedia en anillo especializada a la memoria intermedia en 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 descendientemente 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 de gráficos global, con tamaño de 2 GB, pueden emplearse técnicas de particionamiento de recursos de memoria de gráficos y aumento de espacio de direcciones. Para los espacios de memoria de gráficos locales, cada uno también con un tamaño de 2 GB, una memoria de gráficos local por VM puede implementarse a través de la conmutación de contexto de representación, debido a que la memoria de gráficos local es accesible únicamente por la GPU 1600.

Como se ha mencionado, una realización particiona la memoria de gráficos global entre las VM 1631-1632. Como se ha explicado anteriormente, un mecanismo de planificación de CPU/GPU dividida requiere que la memoria de gráficos 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos. 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 de gráficos global y espacio de direcciones de memoria de gráficos por proceso. El espacio de memoria de gráficos 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 de gráficos alta o espacio de memoria de gráficos 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 de gráficos 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 de gráficos global estático puede hacer frente a un problema de escalabilidad. Por ejemplo, para un espacio de memoria de gráficos 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 de gráficos alta (oculta). Con el esquema de particionamiento de espacio de memoria de gráficos global estático, cada VM con virtualización completa de GPU habilitada puede asignarse con 128 MB de apertura y 384 MB de espacio de memoria de gráficos alta. Por lo tanto, en el espacio de memoria de gráficos global de 2 GB puede acomodarse únicamente un máximo de cuatro VM.

Además del problema de escalabilidad, las VM con espacio de memoria de gráficos 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 de gráficos. 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 de gráficos. Mientras tanto, puede tener que establecerse aparte algún espacio de memoria de gráficos 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos global 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 de gráficos 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 de gráficos.

El hipervisor/VMM 1710 puede usar el analizador de comandos 1718 para detectar el conjunto potencial de trabajo de memoria 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 de gráficos 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 de gráficos 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 de gráficos 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 emisor por flujo continuo de comando de vídeo (VCS) 1726 y el motor de representación de emisor por flujo continuo de comando 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 de gráficos en ejecución y que hay que ejecutar para todos los motores de representación asociados con una VM si las direcciones de memoria de gráficos 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 de gráficos 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 de gráficos 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 de gráficos, 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 de gráficos 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 de gráficos 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 de gráficos global estático, el esquema de sobreasignación de memoria en virtualización de GPU puede habilitar que cada VM use todo el espacio de memoria de gráficos alta en el espacio de memoria superior 1752, que puede permitir que algunas aplicaciones dentro de cada VM usen un mayor espacio de memoria de gráficos para un rendimiento mejorado.

Con esquemas de particionamiento de memoria de gráficos 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 de gráficos usado por las cargas de trabajo que hay que ejecutar puede asignarse en tiempo de ejecución, lo que ahorra espacio de memoria de gráficos 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 de gráficos. 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 de gráficos 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 correlacionar 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 dirección (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 de gráficos (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 PPGTT 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 DE GRÁFICOS 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** 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos 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 de gráficos (por ejemplo, aplicaciones de diseño gráfico, aplicaciones de trazado de rayos interactivas y no interactivas, software de productividad, software de edición de vídeo, etc.).

PROGRESO DIRECTO GARANTIZADO

El tejido de memoria en una implementación de GPU virtualizada se comparte mediante diversos recursos de procesamiento de gráficos en una GPU (por ejemplo, EU, muestreadores, sombreadores, puertos de datos, etc.). La lógica de aprovisionamiento de tejido de memoria dinámica asigna una porción del ancho de banda de tejido de memoria a cada uno de estos recursos usando una política de arbitraje que puede ser un factor en la VM o aplicación particular para la que el recurso está realizando su función (por ejemplo, de acuerdo con una prioridad asociada con la VM/aplicación).

Una realización de la invención implementa un mecanismo de cola inteligente en el que se asignan colas en cada nivel del tejido de memoria a VM particulares. Si una cola aguas abajo se llena para una VM particular, un arbitrador bloqueará datos en la cola aguas arriba para esa VM, evitando de este modo que el tráfico de una VM bloquee el tráfico de otra VM.

La **Figura 20** ilustra una realización de una GPU 2060 con recursos de procesamiento de gráficos subdivididos en una pluralidad de cortes 2010-2014. En una realización, los cortes 2010-2014 y motores de medios 2020-2021 se comparten mediante múltiples VM a través de la interfaz 2022, que también acopla los cortes 2010-2014 a un subsistema de caché y memoria 2070. Los recursos de cada corte 2010-2014 se acoplan al tejido de memoria 2051 en puntos de conexión accesibles a través de lógica de almacenamiento en memoria intermedia y arbitraje 2050. En una realización, cada corte puede incluir un conjunto designado de motores de procesamiento de gráficos, tales como motores de procesamiento 3D, motores de BLIT y unidades de ejecución, por nombrar unos pocos. Dependiendo de la implementación, cada corte 2010-2014 puede incluir el mismo número y tipo de motores de procesamiento de gráficos o a cada corte 2010-2014 puede asignarse un número y tipo diferentes de motores de procesamiento de gráficos.

Una realización de la invención usa técnicas de puesta en cola multinivel para garantizar que el tráfico bloqueado de una VM no bloqueará el tráfico de otra VM. En particular, como se ilustra en la **Figura 21**, se implementa una arquitectura de puesta en cola multinivel en la que los arbitradores 2150-2151 conectan los cortes 2170-2171 al tejido de memoria a través de series de colas 2101-2103, 2111-2113, 2121-2123. La interfaz 2022 incluye circuitería y lógica de gestión de memoria (por ejemplo, una TLB, lógica de recorrido de página, etc.) para implementar transacciones de memoria en nombre de los cortes 2010-2014 y motores de medios 2020-2021 dentro del subsistema de caché y memoria 2070 (por ejemplo, realizando traducciones de memoria virtual a física, etc.). En la realización ilustrada, cada arbitrador 2150-2151 se coloca entre un conjunto de colas aguas arriba y un segundo de colas aguas abajo. Una cola particular en cada nivel (por ejemplo, las colas 2101, 2111, 2121) puede asignarse a una VM o aplicación particular. En una implementación, un arbitrador 2150-2151 no pondrá en cola tráfico en la cola aguas arriba de una VM (por ejemplo, la cola 2111) a no ser que haya disponible espacio en su cola aguas abajo (por ejemplo, la cola 2121).

A modo de ejemplo, y no como limitación, si las colas 2101, 2111, 2121 están almacenando tráfico para la VM0, el arbitrador 2150 no añadirá nuevo tráfico a la cola aguas arriba 2111 si la cola aguas abajo 2121 para VM0 está llena

(por ejemplo, debido a un fallo de página dentro de la interfaz 1322). En su lugar, el arbitrador 450 añadirá tráfico a las colas 2112-2113 que pueden asignarse a otras VM. Una vez que la cola aguas abajo 2121 tiene espacio disponible, el arbitrador 2150 almacenará de nuevo tráfico en la cola aguas arriba 2111. En una implementación, el arbitrador 2151 puede enviar una señal al arbitrador 2150 para notificar al arbitrador 2150 cuándo está llena la cola 2121, provocando de esta manera que el arbitrador 2150 se abstenga de poner en cola más tráfico en la cola 2111 y en su lugar se centre en poner en cola tráfico en las otras colas 2112-2113. De manera similar, cuando haya espacio disponible en la cola 2121, el arbitrador 2151 notificará al arbitrador 2150 que puede comenzar a poner en cola tráfico en la cola 2111.

Una realización de la invención soporta canales virtuales en los que a cada VM se asigna un canal virtual diferente para acceder al tejido de memoria. Cada canal virtual, a su vez, se asocia con una secuencia de colas, tales como las colas 2101, 2111, 2121 en el ejemplo anterior. Los recursos de corte 2170-2171 se disponen entre las colas y se comparten por las diversas VM/aplicaciones ejecutadas en el sistema. El resultado final es que si el tráfico se bloquea en un conjunto de colas para una VM (por ejemplo, como resultado de un fallo de página dentro de la interfaz 1322), aún puede procesarse tráfico de datos para otras VM a través del tejido de memoria 1351.

En la **Figura 22** se ilustra un método de acuerdo con una realización de la invención. El método puede implementarse dentro del contexto de las arquitecturas de procesamiento de gráficos descritas en el presente documento, pero sin limitación a ninguna arquitectura particular.

En 2201, se asignan una secuencia de colas aguas arriba y aguas abajo a cada VM/aplicación. Obsérvese que los términos "aguas arriba" y "aguas abajo" se usan en un sentido relativo, es decir, una cola puede ser aguas arriba en relación con una primera cola y aguas abajo en relación con una segunda cola. En una realización, una única cola se asigna a la VM/aplicación en cada nivel en el flujo. En 2202, se proporciona una realimentación desde cada cola aguas abajo a cada cola aguas arriba (o a lógica de arbitraje que controla las entradas a las colas) para indicar el uso de colas. Por ejemplo, una señal puede enviarse para indicar que una cola aguas arriba está llena o cercana a estar llena.

Si se determina que una cola particular está llena (por ejemplo, la cola N) en 2203, entonces cualquier dato nuevo se bloquea de entrar en la cola aguas arriba (por ejemplo, la cola N-1) en 2204. El proceso se repite, a continuación, desde 2202 o, como alternativa, 2201 si hay un conjunto nuevo o diferente de VM/aplicaciones. Una vez que la cola N ya no está llena, puede almacenarse tráfico nuevo dentro de la cola N-1.

IMPLEMENTACIÓN DE IOMMU CON MÚLTIPLES CORTES

En implementaciones que incluyen múltiples pilas de recursos de procesamiento de gráficos (por ejemplo, múltiples conjuntos de cortes), cada pila tiene su propia interfaz al tejido de memoria y su propia unidad de gestión de memoria (MMU). Cada MMU puede realizar traducciones de direcciones en nombre de los cortes en su pila y almacena en caché traducciones accedidas recientemente en una TLB local. En algunas implementaciones actuales, cada una de las MMU se comunican con una Unidad de Gestión de Memoria de Entrada/Salida (IOMMU) central para realizar las traducciones de direcciones y garantizar la coherencia de las traducciones de direcciones.

Una realización de la invención asegura la coordinación entre las MMU estableciendo una MMU como una "maestra" y las restantes MMU como "esclavas". Toda comunicación con la IOMMU se produce a través de la maestra.

La **Figura 23** ilustra una realización en la que tres conjuntos de recursos de procesamiento de gráficos 2370-2372 realizan transacciones de memoria a través de un único IOMMU 2380. Cada conjunto de recursos de procesamiento de gráficos 2370-2372 incluye una pluralidad de cortes 2310-2314, 2330-2334, 2350-2354, cada uno de los cuales puede incluir conjuntos de unidades de ejecución (EU), muestreadores, motores 3D, rasterizadores, sombreadores de píxeles, unidades transversales o cualquier otra forma de recursos de procesamiento de gráficos. Cada conjunto de recursos de procesamiento de gráficos también incluye unidades de procesamiento de medios 2320-2321, 2340-2341, 2360-2361 y tiene una unidad de gestión de memoria (MMU) especializada 2322, 2342, 2362 para realizar operaciones de acceso a memoria, tales como traducciones de direcciones, operaciones de fallo de página y operaciones de recorrido de página. Cada MMU 2322, 2342, 2362 puede incluir una TLB local 2323, 2343, 2363 para almacenar en caché traducciones de direcciones virtuales a físicas y una o más memorias caché 2325, 2345, 2365 para almacenar en caché datos e instrucciones.

En una implementación, una de las MMU 2322 se designa la "maestra" que se comunica directamente con la IOMMU 2380 en nombre de las otras MMU 2342, 2362 que se designan como "esclavas". Las transacciones de memoria de las MMU esclavas 2342, 2362 se envían inicialmente a la MMU maestra 2322, que reenvía las mismas a la IOMMU para su procesamiento. Por lo tanto, existe un único punto de contacto con la IOMMU, simplificando la coordinación y reduciendo el tráfico.

En una realización, ciertas operaciones de gestión de memoria, tales como traducciones de direcciones, pueden tratarse sin acceder a la IOMMU 2380. Por ejemplo, si la MMU 2362 requiere una traducción que se almacena en la TLB 2343 de la MMU 2342 o la TLB 2323 de la MMU 2322, entonces la traducción puede proporcionarse a una

MMU sin interacción con la IOMMU. De manera similar, si un corte particular requiere datos almacenados en una caché local 2325, 2345, 2365, entonces los datos pueden recuperarse sin cargar la IOMMU 2380. Por lo tanto, en una realización, las operaciones de gestión de memoria se intentan inicialmente internamente (es decir, dentro de una MMU o a través de comunicación con otras MMU) antes de transmitir una petición a la IOMMU 2380.

En una realización, un código de ID se incorpora en un campo privado de cada transacción que identifica inequívocamente la MMU desde la que se originó. Las transacciones enviadas a la IOMMU 2380 y respuestas desde la IOMMU 2380 incluirán el código de ID. La circuitería de encaminamiento de transacción 2324, 2344, 2364 dentro de las MMU usan el código de ID para encaminar la respuesta desde la IOMMU 2380 a la MMU solicitante. En una realización, la circuitería de encaminamiento de transacción 2324, 2344, 2364 mantiene una tabla de encaminamiento u otra estructura de datos que asocia cada una de las MMU 2322, 2342, 2362 con su código de ID.

Como se ha mencionado, cada una de las MMU 2322, 2342, 2362 incluye su propia TLB 2323, 2343, 2363, respectivamente, que almacena en caché traducciones de direcciones utilizadas recientemente. En un entorno virtualizado, cada entrada de TLB puede incluir una correlación completa desde una dirección de invitado virtual (GVA) a una dirección física de anfitrión (HPA). Por lo tanto, si una traducción se almacena en una TLB, no se requiere una traducción de dos niveles estándar (es decir, desde GVA a dirección física de invitado (GPA) y de la GPA a la HPA). Las TLB 2323, 2342, 2363 se mantienen coherentes a través de comunicación entre cada una de las MMU 2322, 2342, 2362 y comunicación entre la MMU maestra 2322 y la IOMMU 2380.

En la **Figura 24** se ilustra un método de acuerdo con una realización de la invención. El método puede implementarse dentro del contexto de las arquitecturas de procesamiento de gráficos descritas en el presente documento, pero sin limitación a ninguna arquitectura particular.

En 2401, una primera MMU se designa como una maestra y una o más otras MMU se designan como esclavas. Como se ha analizado anteriormente, cada una de las MMU puede configurarse para dar servicio a peticiones de una pluralidad de cortes de procesamiento de gráficos. En 2402, una MMU esclava recibe una petición de transacción de memoria desde uno de sus cortes. Si puede dar servicio a la petición localmente, determinarse en 2203, entonces la MMU esclava genera una respuesta por sí misma en 2404. Por ejemplo, la MMU esclava puede acceder a la traducción de dirección desde su TLB o a datos locales desde su caché local.

Si la esclava no puede dar servicio a la petición localmente, entonces en 2405, reenvía la petición a la MMU maestra, incluyendo la MMU esclava código de ID en el paquete de transacción. La MMU maestra puede ser capaz de dar servicio a la petición por sí misma o puede enviar la petición a la IOMMU. En 2406, la MMU maestra envía la petición a la IOMMU que puede acceder a memoria de sistema en nombre de la MMU esclava y genera una respuesta (por ejemplo, que contiene los datos solicitados). En una realización, la petición y la respuesta incluyen el código de ID de MMU esclava. La MMU maestra recibe la respuesta desde la IOMMU en 2407 y encamina la respuesta a la MMU esclava usando el código de ID de la MMU esclava. En 2408, la MMU esclava reenvía la respuesta al corte solicitante.

Las técnicas anteriormente descritas reducen el tráfico entre la MMU y IOMMU porque muchas peticiones de memoria pueden recibir servicio localmente. Además, las realizaciones de la invención reducen las líneas de comunicación físicas requeridas para que la IOMMU de servicio a una pluralidad de MMU.

GESTIÓN DE UNIDADES DE PROCESAMIENTO DE GRÁFICOS VIRTUALES USANDO UN ID DE ESPACIO DE DIRECCIONES DE PROCESO

Los dispositivos de PCI Express se enumeran usando valores de Bus:Dispositivo:Función en lo que los valores de función están limitados a 0-7. Como resultado, las implementaciones actuales que distinguen GPU virtuales (vGPU) usando diferentes valores de función están limitadas a 8 vGPU. Mientras es posible modificar el valor de dispositivo dentro del esquema de enumeración de PCIe, esto se complicaría ya que el nuevo valor de dispositivo para un dispositivo de gráficos puede entrar en conflicto con otros dispositivos (por ejemplo, dispositivos de gráficos se enumeran generalmente con un valor de dispositivo de 2).

Una realización de la invención proporciona soporte para vGPU adicionales usando el ID de espacio de direcciones de proceso (PASID) para identificar diferentes vGPU. Dado el valor de PASID de 20 bits, por ejemplo, puede abordarse un número prácticamente ilimitado de vGPU.

La **Figura 25** ilustra consultas de tabla de paginación multinivel empleadas en una realización que utilizan el PASID para distinguir entre las vGPU. Un puntero raíz 2501 apunta a la base de una tabla raíz 2512. Como se ilustra, el valor B (Bus) de la enumeración B:D:F se usa como un desplazamiento para identificar la entrada 2502 que apunta a la base de una tabla de contexto 2513. Los valores D (Dispositivo) y F (Función) se usan a continuación como un desplazamiento a la entrada 2503 en la tabla de contexto 2513.

En implementaciones anteriores, la entrada en la tabla de contexto 2513 apuntaría tanto a la tabla de PASID 2514 como a la tabla de nivel 4 de mapa de página de segundo nivel (SL PML-4) (como se indica por la línea discontinua

con una X). En una realización de la invención, la tabla de contexto 2513 no apunta a la SL PML-4. En su lugar, el valor de PASID 2504 se usa para identificar tanto la tabla de PML-4 de primer nivel cuando realiza correlación de direcciones virtuales de gráficos (GVA) a direcciones físicas de gráficos (GPA) como la tabla de PML-4 de segundo nivel cuando realiza correlación de GPA a direcciones físicas de anfitrión (HPA). Por lo tanto, en la realización ilustrada, la entrada 2503 en la tabla de contexto 2513 identifica una tabla de PASID 2514 y el valor de PASID se usa como un desplazamiento para identificar una entrada 2504 que apunta tanto a la PML-4 como a la SL PML-4. Un beneficio de esta disposición es que el PASID es un valor de 20 bits, proporcionando 2^{20} de espacio de direccionamiento para identificar un gran número de diferentes GPU virtuales (en contraposición a sistemas anteriores que estaban sujetos a 8 vGPU usando la enumeración B:D:F).

En esta realización, puede usarse una porción especificada del PASID (por ejemplo, N bits contiguos o bits distribuidos dentro del PASID) para identificar el conjunto de tablas de primer nivel y segundo nivel para una vGPU particular. Por supuesto, el identificador de vGPU puede almacenarse en el PASID en una diversidad de formas mientras aún cumple con los principios subyacentes de la invención.

En la **Figura 26** se ilustra un método de acuerdo con una realización de la invención. El método puede implementarse dentro del contexto de las arquitecturas de procesamiento de gráficos descritas en el presente documento, pero sin limitación a ninguna arquitectura particular.

En 2601, cada vGPU se asocia con un conjunto particular de bits o intervalo de bits dentro del PASID. En 2603, en respuesta a una petición de traducción de dirección para una traducción de dirección virtual de gráficos (GVA) a dirección física de anfitrión (HPA), se usa una primera porción del PASID para identificar tablas que contienen la GVA a dirección física de gráficos (GPA) (es decir, tablas de "primer nivel"). Se usa una segunda porción del PASID para identificar tablas que contienen traducciones de GPA a HPA (es decir, tablas de "segundo nivel").

Las realizaciones de la invención simplifican drásticamente la configuración de sistemas de gráficos virtualizados con más de 8 vGPU (el límite actual usando enumeración de PCIe). Como se ha descrito anteriormente, esto puede conseguirse usando el PASID para direccionar diferentes vGPU, aprovechándose del intervalo de direcciones extremadamente grande soportado por el PASID de 20 bits.

IMPLEMENTACIONES DE REGISTRO DE DIRECCIONES BASE VIRTUALES DE INVITADO

Una realización usa un Registro de Direcciones Base (BAR) de PCI Express virtual para exponer memoria de dispositivo privada al sistema operativo (SO) e Hipervisor/VMM, asignando esa memoria privada entre máquinas virtuales (VM) de invitado, haciendo cumplir las asignaciones cuando el dispositivo virtualizado accede a la memoria privada en nombre de un invitado. El dispositivo puede crear un Registro de Direcciones Base de PCI (BAR_H) y el VMM/Hipervisor correlaciona el BAR con el dispositivo (o función) con una VM (es decir, BAR_{Anfitrión} → BAR_{Invitado}). Si el hardware tiene la razón para tratar este intervalo de BAR separadamente (por ejemplo, omitiendo la IOMMU, diferentes configuraciones de control, etc.) entonces se determina la página de invitado del BAR_G. Puede usarse un registro real o emulado para permitir que el software autnotifique el valor de invitado de BAR_G al hardware. El hardware, a continuación, puede detectar y traducir direcciones físicas de invitado (GPA) dentro de este intervalo como se describe a continuación.

Una realización de la invención se usa para asignar memoria de ancho de banda alto (HBM). Cuando una única unidad de procesamiento de gráficos (GPU) que utiliza HBM se comparte con múltiples VM, el hardware y software de gráficos deben distribuir la HBM entre las VM. Se implementan técnicas para garantizar que los contenidos de estas asignaciones por VM estén protegidos contra accesos por otras VM, ya se conceda o no una porción de una GPU dada.

En una realización de la invención, debido a que HBM representa memoria física real, y no solo espacio de direcciones (es decir, la Tabla de Traducción de Gráficos Global (GGTT)), el esquema de asignación trata menos del número máximo de funciones virtuales (VF) activas sin desperdiciar HBM. En particular, esta realización incluye hardware que soporta una asignación de tamaño de HBM programable para cada VF (en lugar de simplemente dividir la HBM uniformemente en TotalVF+1 particiones).

Una realización asigna HBM a las VF en incrementos de 2M. Esto es compatible con diseños de IOMMU actuales con tablas de paginación que soportan tamaños de página de 4K y 2M. Sin embargo, los principios subyacentes de la invención no se limitan a ningún tamaño de incremento particular para asignar HBM.

La **Figura 27** ilustra una asignación flexible de la HBM a las VM que comparten un único dispositivo de gráficos a través de valores base y de límite sobre una base de VF a VF. En particular, en una realización, la asignación de tamaño de HBM programable 2700 se realiza dimensionando la VF HBMBAR 2704 de las funciones virtuales para ser iguales al tamaño de la HBMBAR de anfitrión 2702 (por ejemplo, 2G). Cada VF individual se aprovisiona, a continuación, con un intervalo, VF_n_HBMSIZE (por ejemplo, donde n está entre 1 y 7 para siete funciones virtuales diferentes), dentro de la HBM privada 2701 que será accesible desde su propio VF_n_HBMBAR_H. Los VBAR de anfitrión 2703 se ilustran para VF1 (VF1_HBMBAR_H), VF2 (VF2_HBMBAR_H) y VF3 (VF3_HBMBAR_H). Este

intervalo se correlaciona con cada `VFn_HBMBAR_H` que se inicia en la base (0) y se extiende desde ahí para `VFn_HBMSIZE` MB, como se indica por los indicadores de Límite 1, 2 y 3. Esto puede o no cubrir todo el `VFn_HBMBAR_H`. Si no, entonces la región por encima de `VFn_HBMSIZE` se correlaciona con un acceso no válido (es decir, una página ficticia, única por VF), como se ilustra en la **Figura 27**, en lugar de a cualquier otra parte de la memoria de HBM privada u otras partes de DRAM.

En una realización, el mecanismo de detección y omisión de HBM descrito anteriormente para paso a través se extiende para cada VF individual. Para hacer esto, una realización incluye una `VFn_HBMBASE_G`, `VFn_HBMBASE_H` y `VFn_HBMSIZE` separadas por VF (G=invitado y H= anfitrión). Únicamente la `VFn_HBMBASE_G` puede ser escrita por la VM de invitado. Los otros valores son programables únicamente a partir de la función física (por ejemplo, función 0 como se indica en la tabla a continuación).

Usando estas técnicas, el software puede configurar entre 0 y `HBMSIZE` de HBM para cada VF. El software de función física (PF) en el dominio de control gobierna la disposición de los segmentos en la HBM privada. Este SW de PF asegura que las regiones relacionadas con VF no son solapantes. Pueden ser adyacentes aunque pequeños huecos pueden requerir posteriormente la compactación para recuperarse en tamaños útiles.

En una realización, el controlador de VMM/PF puede suspender la operación de una VM en una VF, modificar esta correlación de memoria y, a continuación, reanudar la VM. Esta técnica se usa para reequilibrar/redimensionar la asignación de HBM a una VM para compactar y consolidar pequeños intervalos libres de HBM que resultan del apagado de VM, etc. Por ejemplo, en la **Figura 27**, si se destruyera VM3, entonces el intervalo de las VM2 podría desplazarse hacia abajo adyacente a las VM1.

En una implementación, el software de aprovisionamiento de VMM/Hipervisor comunica la asignación de HBM para una interfaz de VF particular a través del controlador de PF antes de lanzar una VM con esa VF asignada. El software PF determina el valor de `VFn_HBMBASE_H` y `VFn_HBMSIZE` para la VF. `VFn_HBMSIZE` puede hacerse disponible a la VM durante la inicialización de tal forma que el controlador de modo de núcleo (KMD) de VF de la VM puede descubrir la cantidad de memoria HBM disponible a gestionar dentro de su `HBMBAR`. Las VF pueden todas tener el mismo tamaño `HBMBAR`. Una vez que la VMM/Hipervisor ha correlacionado ese `BAR` con el espacio de direcciones de la VM, el software de VM notificará de vuelta la base observada de la `HBMBAR` (`VFn_HBMBASE_G`) como una dirección física de gráficos (GPA). Esto puede soportarse a través de un canal de software de VF->PF y/o un registro de E/S correlacionado con memoria (MMIO) emulado de acuerdo con el marco de virtualización de entrada-salida (IOV) escalable implementado en arquitecturas actuales (por ejemplo, implementaciones de PCI Express).

En una realización, estos tres registros (`VFn_HBMBASE_H`, `VFn_HBMBAR_G`, `VFn_HBMSIZE`) se implementan para cada motor dentro de la GPU (por ejemplo, cada motor de representación, motor de medios, etc.) y el emisor por flujo continuo de comando (CS) es responsable de programar los mismos cuando carga un nuevo contexto en un motor. El CS puede cargar esa información desde un descriptor de contexto en memoria de anfitrión (pero estos datos no deben ser accesibles por los KMD de anfitrión u otro software de VM). Como alternativa, si el número total de VF es relativamente pequeño (por ejemplo, 7), el árbitro de gráficos (GAM) puede implementar los 7 conjuntos de registros de VF y elegir entre los mismos por consiguiente, sin ninguna dependencia adicional del CS (aunque el CS aún proporciona VF de motor a GAM, por requisitos de SR-IOV existentes).

La **Figura 28** ilustra un ejemplo específico que realiza una omisión de tecnología de virtualización para E/S (Vt-d) directa o redirección a una trayectoria de acceso no válida dependiendo del desplazamiento del acceso con las VF `HBMBAR`. En esta realización, cada controlador de VF continúa usando `PTE_G` en las tablas 2802 de tabla de traducción de gráficos global (GGTT) y tabla de traducción de gráficos por proceso (PPGTT) (incluyendo la Abertura). En el momento de uso, el hardware realiza una comparación de intervalo para `VFn_HBMBAR_G` y `VFn_HBMSIZE` para la VF que se ejecuta en ese motor o visualizador como se indica en (1). En un acceso dentro de ese intervalo, el hardware se correlaciona de nuevo con un desplazamiento para `VFn_HBMBASE_H` para esa VF en (2). Si el desplazamiento calculado es mayor que `VFn_HBMSIZE` (determinado en (4), después de omitir VT-d en (3)), entonces el acceso no es válido (6) y se trata de tal manera para evitar cualquier fuga de datos dentro, fuera o a través de las VM a través de accesos no válidos. Si el desplazamiento está dentro del límite, entonces el desplazamiento se añade a la base en (5). Si la `PTE` de GPA no está en intervalo de la `HBMBAR_G`, como se determina en (1), entonces se realiza un recorrido de página en (7) y HPA se determinan y almacenan en (8).

`PTE_H` debe llegar/llegará únicamente como un acceso no válido (6) o dentro del intervalo asignado de HBM para esa VF (5) y en ningún otro sitio cuando se desencadene la omisión (3). Cuando la `PTE_G` está fuera del intervalo de `VFn_HBMBAR_G` dentro de la VM entonces no está omitiendo el recorrido de IOMMU de 2º nivel habitual y por tanto VT-d gobernará la determinación de `PTE_H`.

VIRTUALIZACIÓN DE TABLAS DE TRADUCCIÓN DE GRÁFICOS A TRAVÉS DE UN GRAN NÚMERO DE VMS

A medida que aumentan las resoluciones y el número de VM usadas para procesamiento de gráficos, las tablas de traducción de gráficos (GTT) se sobrecargarán. Las GTT actuales tienen limitaciones de tamaño (por ejemplo, 4 GB)

que no será adecuado para soportar futuras implementaciones.

En una realización, en lugar de una tabla de paginación de GTT de 1 nivel, se usa una tabla de paginación multinivel en la que a cada VM se asigna su propia tabla de paginación. Puede realizarse, entonces, un recorrido de página multinivel por VM. No se hace hoy un recorrido de página multinivel debido a problemas de rendimiento (por ejemplo, efecto bandera). Para abordar estas limitaciones, las realizaciones de la invención realizan almacenamiento en caché inteligente y pretraducción para mejorar el rendimiento.

La **Figura 29** ilustra una comparación entre una GTT global de un solo nivel 2900 y una GTT global multinivel 2901. En la GTT global de un solo nivel 2900, a cada VM se asigna una porción designada del espacio de dirección física de gráficos dentro de la tabla que correlaciona cada GPA con una HPA. En contraposición, en la GTT global multinivel 2901, a cada VM se asigna su propia tabla de paginación de nivel 1 en la que cada entrada apunta a una tabla de paginación de nivel 2 como se ilustra. En una realización, las TLB pueden rellenarse inteligentemente para garantizar que las páginas estén disponibles fácilmente.

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

En la descripción siguiente, se exponen numerosos detalles específicos para proporcionar un entendimiento más minucioso. Sin embargo, será evidente para un experto en la materia que las realizaciones descritas en el presente documento se pueden poner en práctica sin uno o más de estos detalles específicos. En otras instancias, no se han descrito características bien conocidas para evitar complicar los detalles de las presentes realizaciones.

Visión general del sistema

La **Figura 1** es un diagrama de bloques que ilustra un sistema informático 100 configurado para implementar uno o más aspectos de las realizaciones descritas en el presente documento. El sistema informático 100 incluye un subsistema de procesamiento 101 que tiene uno o más procesadores 102 y una memoria de sistema 104 que se comunica a través de una ruta de interconexión que puede incluir un concentrador de memoria 105. El concentrador de memoria 105 puede ser un componente separado dentro de un componente de conjunto de chips o se puede integrar dentro de los uno o más procesadores 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 a través de un enlace de comunicación 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede habilitar que el sistema informático 100 reciba una entrada desde uno o más dispositivos de entrada 108. Adicionalmente, el concentrador de E/S 107 puede habilitar que un controlador de visualización, que se puede incluir en los uno o más procesadores 102, proporcione salidas a uno o más dispositivos de visualización 110A. En una realización, los uno o más dispositivos de visualización 110A acoplados con el concentrador de E/S 107 pueden incluir un dispositivo de visualización local, interno o embebido.

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

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

Wi-Fi, de Bluetooth, de comunicación de campo cercano (NFC) o de otro tipo que incluye una o más radios inalámbricas.

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

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

Se apreciará que el sistema informático 100 mostrado en el presente documento es ilustrativo y que son posibles variaciones y modificaciones. La topología de conexión, incluyendo el número y disposición de puentes, el número del procesador o procesadores 102, y el número del procesador o procesadores paralelos 112, se puede modificar como se desee. Por ejemplo, en algunas realizaciones, la memoria de sistema 104 está conectada al procesador o procesadores 102 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria de sistema 104 mediante el concentrador de memoria 105 y el procesador o procesadores 102. En otras topologías alternativas, el procesador o procesadores paralelos 112 están conectados al concentrador de E/S 107 o directamente a uno de los uno o más procesadores 102, en lugar de al concentrador de memoria 105. En otras realizaciones, el concentrador de E/S 107 y el concentrador de memoria 105 se pueden integrar en un único chip. Algunas realizaciones pueden incluir dos o más conjuntos del procesador o procesadores 102 anexados a través de múltiples zócalos, que se pueden acoplar con dos o más instancias del procesador o procesadores paralelos 112.

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

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

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

Cuando la interfaz de anfitrión 206 recibe una memoria intermedia de comandos a través de la unidad de E/S 204, la interfaz de anfitrión 206 puede dirigir operaciones de trabajo para realizar esos comandos a un extremo frontal 208. En una realización, el extremo frontal 208 se acopla con un planificador 210, que está configurado para distribuir comandos u otros elementos de trabajo a una matriz de agrupaciones de procesamiento 212. En una realización, el planificador 210 garantiza que la matriz de agrupaciones de procesamiento 212 está configurada apropiadamente y en un estado válido antes de que las tareas se distribuyan a las agrupaciones de procesamiento de la matriz de

agrupaciones de procesamiento 212. En una realización, el planificador 210 se implementa a través de lógica de firmware que se ejecuta en un microcontrolador. El planificador implementado por microcontrolador 210 se puede configurar para realizar operaciones de planificación y de distribución de trabajo complejas con granularidad gruesa y fina, lo que habilita un rápido otorgamiento de prioridad y conmutación de contexto de hilos que se ejecutan en la matriz de procesamiento 212. En una realización, el software de anfitrión puede probar cargas de trabajo para su planificación en la matriz de procesamiento 212 a través de uno de múltiples llamadores de procesamiento de gráficos. Las cargas de trabajo se pueden distribuir entonces automáticamente a lo largo de la matriz de procesamiento 212 por la lógica del planificador 210 dentro del microcontrolador de planificador.

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

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

En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de procesamiento de gráficos paralelo. En realizaciones en las que el procesador paralelo 200 está configurado para realizar operaciones de procesamiento de gráficos, la matriz de agrupaciones de procesamiento 212 puede incluir lógica adicional para soportar la ejecución de tales operaciones de procesamiento de gráficos, incluyendo, pero sin limitación, lógica de muestreo de textura para realizar operaciones de textura, así como lógica de teselación y otra lógica de procesamiento de vértices. Adicionalmente, la matriz de agrupaciones de procesamiento 212 se puede configurar para ejecutar programas de sombreado relacionados con el procesamiento de gráficos tales como, pero sin limitación, sombreadores de vértices, sombreadores de teselación, sombreadores de geometría y sombreadores de píxeles. La unidad de procesamiento paralelo 202 puede transferir datos desde memoria de sistema a través de la unidad de E/S 204 para su procesamiento. Durante el procesamiento, los datos transferidos se pueden almacenar en memoria en chip (por ejemplo, la memoria de procesador paralelo 222) durante el procesamiento y, entonces, escribirse en diferido en memoria de sistema.

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

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

Cada una de las una o más instancias de la unidad de procesamiento paralelo 202 se puede acoplar con la memoria de procesador paralelo 222. La memoria de procesador paralelo 222 puede accederse mediante la barra transversal de memoria 216, que puede recibir solicitudes de memoria de la matriz de agrupación de procesamiento 212, así

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

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

En una realización, una cualquiera de las agrupaciones 214A-214N de la matriz de agrupaciones de procesamiento 212 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 224A-224N dentro de la memoria de procesador paralelo 222. La barra transversal de memoria 216 se puede configurar para transferir la salida de cada agrupación 214A-214N a cualquier unidad de subdivisión 220A-220N o a otra agrupación 214A-214N, que puede realizar operaciones de procesamiento adicionales sobre la salida. Cada agrupación 214A-214N se puede comunicar con la interfaz de memoria 218 a través de la barra transversal de memoria 216 para leer desde o escribir en diversos dispositivos de memoria externos. En una realización, la barra transversal de memoria 216 tiene una conexión a la interfaz de memoria 218 para comunicarse con la unidad de E/S 204, así como una conexión a una instancia local de la memoria de procesador paralelo 222, habilitando que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 214A-214N se comuniquen con memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 202. En una realización, la barra transversal de memoria 216 puede usar canales virtuales para separar flujos de tráfico entre las agrupaciones 214A-214N y las unidades de subdivisión 220A-220N.

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

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

En aplicaciones de gráficos, la ROP 226 es una unidad de procesamiento que realiza operaciones de rasterización tales como estarcido, prueba z, mezcla y similares. La ROP 226 emite entonces datos de gráficos procesados que

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

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

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

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

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

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

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

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

En aplicaciones de gráficos e informáticas, una agrupación de procesamiento 214 se puede configurar de manera que cada multiprocesador de gráficos 234 está acoplado a una unidad de textura 236 para realizar operaciones de correlación de textura, por ejemplo, determinar posiciones de muestra de textura, leer datos de textura y filtrar los datos de textura. Se leen datos de textura desde una caché de L1 de textura interna (no mostrada) o, en algunas realizaciones, desde la caché de L1 dentro del multiprocesador de gráficos 234 y se extraen desde una caché de L2, memoria de procesador paralelo local o memoria de sistema, según sea necesario. Cada multiprocesador de gráficos 234 emite tareas procesadas a la barra transversal de datos 240 para proporcionar la tarea procesada a otra agrupación de procesamiento 214 para su procesamiento adicional o para almacenar la tarea procesada en una caché de L2, memoria de procesador paralelo local o memoria de sistema a través de la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones prerrasterización) está configurada para recibir datos desde el multiprocesador de gráficos 234, dirigir datos a unidades de ROP, que se pueden ubicar con unidades de subdivisión como se describe en el presente documento (por ejemplo, las unidades de subdivisión 220A-220N de la Figura 2). La unidad de preROP 242 puede realizar optimizaciones para la mezcla de color, organizar datos de color de píxel y realizar traducciones de dirección.

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

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

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

El fichero de registro 258 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de

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

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

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

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

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

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

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

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

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

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

Técnicas para interconexión de GPU a procesador de anfitrión

La **Figura 4A** ilustra una arquitectura ilustrativa en la que una pluralidad de GPU 410-413 están acopladas de manera comunicativa a una pluralidad de procesadores de múltiples núcleos 405-406 a través de los enlaces de alta velocidad 440-443 (por ejemplo, buses, interconexiones de punto a punto, etc.). En una realización, los enlaces de alta velocidad 440-443 soportan un caudal de comunicación de 4 GB/s, 30 GB/s, 80 GB/s o superior, dependiendo de la implementación. Se pueden usar diversos protocolos de interconexión, incluyendo, pero sin limitación, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no están limitados a ningún protocolo o caudal de comunicación particular.

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

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

Como se describe a continuación, aunque los diversos procesadores 405-406 y las diversas GPU 410-413 se pueden acoplar físicamente a una memoria 401-402, 420-423 particular, respectivamente, se puede implementar una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtual (también

denominado espacio "de direcciones eficaces") está distribuido entre todas las diversas memorias físicas. Por ejemplo, cada una de las memorias de procesador 401-402 puede comprender 64 GB del espacio de direcciones de memoria de sistema y cada una de las memorias de GPU 420-423 puede comprender 32 GB del espacio de direcciones de memoria de sistema (dando como resultado un total de memoria direccionable de 256 GB en este ejemplo).

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

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

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

En una realización, un circuito intermediario 425 acopla de manera comunicativa el módulo de aceleración de gráficos 446 al bus de coherencia 464, permitiendo que el módulo de aceleración de gráficos 446 participe en el protocolo de coherencia de caché como un homólogo de los núcleos. En particular, una interfaz 435 proporciona conectividad al circuito de intermediario 425 a través del enlace de alta velocidad 440 (por ejemplo, un bus PCIe, NVLink, etc.) y una interfaz 437 conecta el módulo de aceleración de gráficos 446 al enlace 440.

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

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

Un conjunto de registros 445 almacena datos de contexto para hilos ejecutados por los motores de procesamiento de gráficos 431-432, N y un circuito de gestión de contexto 448 gestiona los contextos de hilo. Por ejemplo, el

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

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

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

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

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

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

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

Una realización soporta diferentes modelos de programación que incluyen un modelo de programación de proceso especializado (sin virtualización de módulo de aceleración de gráficos) y modelos de programación compartidos (con virtualización). El último puede incluir modelos de programación que se controlan por el circuito de integración del acelerador 436 y modelos de programación que se controlan por el módulo de aceleración de gráficos 446.

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

En los modelos de programación de proceso especializado, los motores de procesamiento de gráficos 431-432, N, pueden compartirse por múltiples particiones de VM/aplicación. Los modelos compartidos requieren un sistema hipervisor para virtualizar los motores de procesamiento de gráficos 431-432, N para permitir el acceso por cada

sistema operativo. Para sistemas de única partición sin un hipervisor, los motores de procesamiento de gráficos 431-432, N son propiedad del sistema operativo. En ambos casos, el sistema operativo puede virtualizar los motores de procesamiento de gráficos 431-432, N para proporcionar acceso a cada proceso o aplicación.

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

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

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

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

35 Durante el funcionamiento, una unidad de extracción de WD 491 en el corte de integración de acelerador 490 extrae el WD 484 siguiente que incluye una indicación del trabajo a hacer por uno de los motores de procesamiento de gráficos del módulo de aceleración de gráficos 446. Los datos del WD 484 pueden almacenarse en registros 445 y usarse por la MMU 439, el circuito de gestión de interrupción 447 y/o el circuito de gestión de contexto 446 como se ilustra. Por ejemplo, una realización de la MMU 439 incluye circuitería de recorrido de segmentos/páginas para acceder a las tablas de segmentos/páginas 486 dentro del espacio de direcciones virtuales de SO 485. El circuito de gestión de interrupciones 447 puede procesar los eventos de interrupción 492 recibidos del módulo de aceleración de gráficos 446. Cuando se realizan operaciones de gráficos, una dirección eficaz 493 generada por un motor de procesamiento de gráficos 431-432, N es traducida a una dirección real por la MMU 439.

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

Tabla 1 - Registros de hipervisor inicializados

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

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

Tabla 2 - Registros inicializados por sistema operativo

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

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

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

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

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

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

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

Tabla 3 - Parámetros de llamada de SO a hipervisor

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de guardado/restablecimiento de contexto (CSRP) de dirección eficaz (EA)

4	Un ID de proceso (PID) e ID de hilo (TID) opcional
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de segmentos de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)

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

Tabla 4 - Información de elemento de proceso

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

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

Como se ilustra en la **Figura 4F**, una realización de la invención emplea una memoria unificada direccionable a través de un espacio de direcciones de memoria virtual común usado para acceder a las memorias de procesador físico 401-402 y a las memorias de GPU 420-423. En esta implementación, operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de direcciones de memoria virtual/eficaz para acceder a las memorias de procesadores 401-402, y viceversa, simplificando de ese modo la programabilidad. En una realización, una primera porción del espacio de direcciones virtual/eficaz está asignada a la memoria de procesador 401, una segunda porción a la segunda memoria de procesador 402, una tercera porción a la memoria de GPU 420, y así sucesivamente. El espacio de memoria virtual/eficaz total (denominado, en ocasiones, el espacio de direcciones eficaces) está distribuido, por lo tanto, a lo largo de cada una de las memorias de procesador 401-402 y las memorias de GPU 420-423, permitiendo que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual correlacionada con esa memoria.

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

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

En una implementación, la selección entre el desvío de GPU y el desvío de procesador de anfitrión es controlada por una estructura de datos de rastreador de desvío. Se puede usar una tabla de desvíos, por ejemplo, que puede ser

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

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

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

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

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

Tubería de procesamiento de gráficos

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

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

los datos de vértice y se puede programar para transformar los datos de vértice desde una representación de coordenadas basada en objetos a un espacio de coordenadas de espacio mundial o un espacio de coordenadas de dispositivo normalizado.

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

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

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

35 En algunas realizaciones, la unidad de procesamiento de geometría 516 puede añadir o borrar elementos en el flujo de geometría. La unidad de procesamiento de geometría 516 emite los parámetros y vértices que especifican primitivas de gráficos nuevas al ensamblador de primitivas 518. El ensamblador de primitivas 518 recibe los parámetros y vértices desde la unidad de procesamiento de geometría 516 y construye primitivas de gráficos para su procesamiento por una unidad de escala, selección y recorte de ventana gráfica 520. La unidad de procesamiento de geometría 516 lee datos que están almacenados en memoria de procesador paralelo o memoria de sistema para su uso en el procesamiento de los datos de geometría. La unidad de escala, selección y recorte de ventana gráfica 520 realiza el recorte, la selección y el ajuste a escala de ventana gráfica y emite primitivas de gráficos procesadas a un rasterizador 522.

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

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

datos z o de color que se leen desde memoria.

Una realización de la invención implementa un enfoque híbrido en el que a ciertas VM se asignan tablas de paginación de un solo nivel (latencia más baja) mientras a otras VM se asignan tablas de paginación multinivel (latencia más alta). En la **Figura 30**, por ejemplo, a la VM0 se ha asignado una GTT de un solo nivel 3000 mientras que a la VM1 y potencialmente otras VM se han asignado GTT multinivel 3001. Esta implementación puede usarse cuando se está usando una VM (por ejemplo, VM0) para aplicaciones en tiempo real y baja latencia y las otras VM se usan para aplicaciones tolerantes a la latencia (por ejemplo, donde se usa una VM para visualizar datos de automoción en tiempo real, tales como en el panel de instrumentos, mientras que una latencia más alta es aceptable para las otras VM, tales como para el sistema de entretenimiento o sistema NAV).

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.

Las realizaciones de la invención pueden incluir diversas etapas, que se han descrito anteriormente. Las etapas pueden materializarse en instrucciones ejecutables por máquina que pueden usarse para hacer que un procesador de propósito general o de propósito especial realice las etapas. Como alternativa, estas etapas pueden ser realizadas por componentes de hardware específicos que contienen lógica cableada para realizar las etapas, o por cualquier combinación de componentes informáticos programados y componentes de hardware personalizados.

Como se describe en el presente documento, las instrucciones pueden referirse a configuraciones específicas de hardware, tales como circuitos integrados específicos de la aplicación (ASIC) configurados para realizar ciertas operaciones o tener una funcionalidad predeterminada o instrucciones de software almacenadas en la memoria materializadas en un medio legible por ordenador no transitorio. Por lo tanto, las técnicas mostradas en las figuras se pueden implementar usando código y datos almacenados y ejecutados en uno o más dispositivos electrónicos (por ejemplo, un puesto terminal, un elemento de red, etc.). Tales dispositivos electrónicos almacenan y comunican (internamente y/o con otros dispositivos electrónicos a través de una red) códigos y datos usando medios legibles por máquina informáticos, tales como medios de almacenamiento legibles por máquina informáticos 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 de comunicación legibles por máquina informáticos transitorios (por ejemplo, señales eléctricas, ópticas, acústicas u otras formas señales propagadas - tales como ondas portadoras, señales infrarrojas, señales digitales, etc.).

Además, tales dispositivos electrónicos incluyen habitualmente un conjunto de uno o más procesadores acoplados a otros uno o más componentes, tales como uno o más dispositivos de almacenamiento (medios de almacenamiento legibles por máquina no transitorios), dispositivos de entrada/salida de usuario (por ejemplo, un teclado, una pantalla táctil y/o una pantalla) y conexiones de red. El acoplamiento del conjunto de procesadores y otros componentes se produce habitualmente a través de uno o más buses y puentes (también denominados controladores de bus). El dispositivo de almacenamiento y las señales que portan 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 almacena habitualmente código y/o datos para su ejecución en el conjunto de uno o más procesadores de ese dispositivo electrónico. Por supuesto, una o más partes de una realización de la invención se pueden implementar usando diferentes combinaciones de software, firmware y/o hardware. A lo largo de toda esta descripción detallada, con fines de explicación, se expusieron numerosos detalles específicos para proporcionar un entendimiento completo de la presente invención. Sin embargo, será evidente para un experto en la materia que la invención se puede poner en práctica sin algunos de estos detalles específicos. En ciertas instancias, estructuras y funciones bien conocidas no se describieron con todo lujo de detalles para evitar complicar la materia objeto de la presente invención. (PLEASE TRANSLATE FROM PAGE 96, LINE 30 TO CLAIM 7, LINE 3) La invención se define en la reivindicación independiente adjunta. Las realizaciones ventajosas se definen mediante las reivindicaciones dependientes adjuntas.

REIVINDICACIONES

1. Un procesador gráfico (2060) que comprende:

una pluralidad de recursos de procesamiento de gráficos para ejecutar instrucciones y procesar datos, estando la pluralidad de recursos de procesamiento de gráficos particionados en cortes (2010-2014), en donde cada corte de

debe asignar a una máquina virtual, VM;

una interfaz de memoria (2022) para acoplar los recursos de procesamiento de gráficos a una memoria del sistema (2381);

una pluralidad de unidades de gestión de memoria, MMU, (2322, 2342, 2362), comprendiendo cada MMU (2322, 2342, 2362):

una pluralidad de memorias intermedias de traducción adelantada, TLB (2323, 2343, 2363), cada TLB (2323, 2343, 2363) almacena las correlaciones de las direcciones de invitado virtual, GVA, con las direcciones físicas de anfitrión, HPA, recuperadas de las tablas de paginación en la memoria del sistema (2381), donde cada GVA está asociada con una VM particular; y

una interfaz para acoplar las MMU (2322, 2342, 2362) a una MMU a nivel de sistema (2380), la interfaz proporciona una comunicación entre la MMU a nivel de sistema (2380) y las MMU (2322, 2342, 2362) para garantizar la coherencia entre las correlaciones de las GVA con las HPA almacenadas en la pluralidad de TLB (2323, 2343, 2363) y las correlaciones de las GVA con las HPA almacenadas en las TLB de la MMU a nivel de sistema (2380).

2. El procesador gráfico de la reivindicación 1, en donde se deben incluir uno o más cortes (2010-2014) en una primera unidad de procesamiento de gráficos virtual, vGPU, que se debe proporcionar a una primera VM, donde la primera VM ejecuta uno o más hilos de la pluralidad de hilos en la primera vGPU.

3. El procesador de gráficos de la reivindicación 2, en donde la primera vGPU se identifica de manera única con un primer valor de identificador de espacio de direcciones de proceso, PASID, y se identifica de manera única al menos una segunda vGPU con un segundo valor de PASID.

4. El procesador de gráficos de la reivindicación 3, en donde la lógica de hardware de traducción de direcciones debe utilizar el primer valor de PASID para identificar las primeras entradas de tabla de paginación y/o entradas TLB asociadas con la primera vGPU y la lógica de hardware de traducción de direcciones debe utilizar el segundo valor de PASID para identificar las segundas entradas de tabla de paginación y/o entradas TLB asociadas con la segunda vGPU.

5. El procesador de gráficos de cualquiera de las reivindicaciones 1 a 4, en donde la memoria de sistema (2381) comprende una memoria de acceso aleatorio dinámica, DRAM, apilada.

6. El procesador de gráficos de la reivindicación 5, en donde la DRAM apilada comprende una memoria de ancho de banda alto, HBM.

7. El procesador de gráficos de cualquiera de las reivindicaciones 1 a 6, en donde la pluralidad de TLB (2323, 2343, 2363) incluye una primera TLB para proporcionar una correlación de una primera GVA con una HPA a una segunda TLB en respuesta a una petición de traducción de la primera GVA.

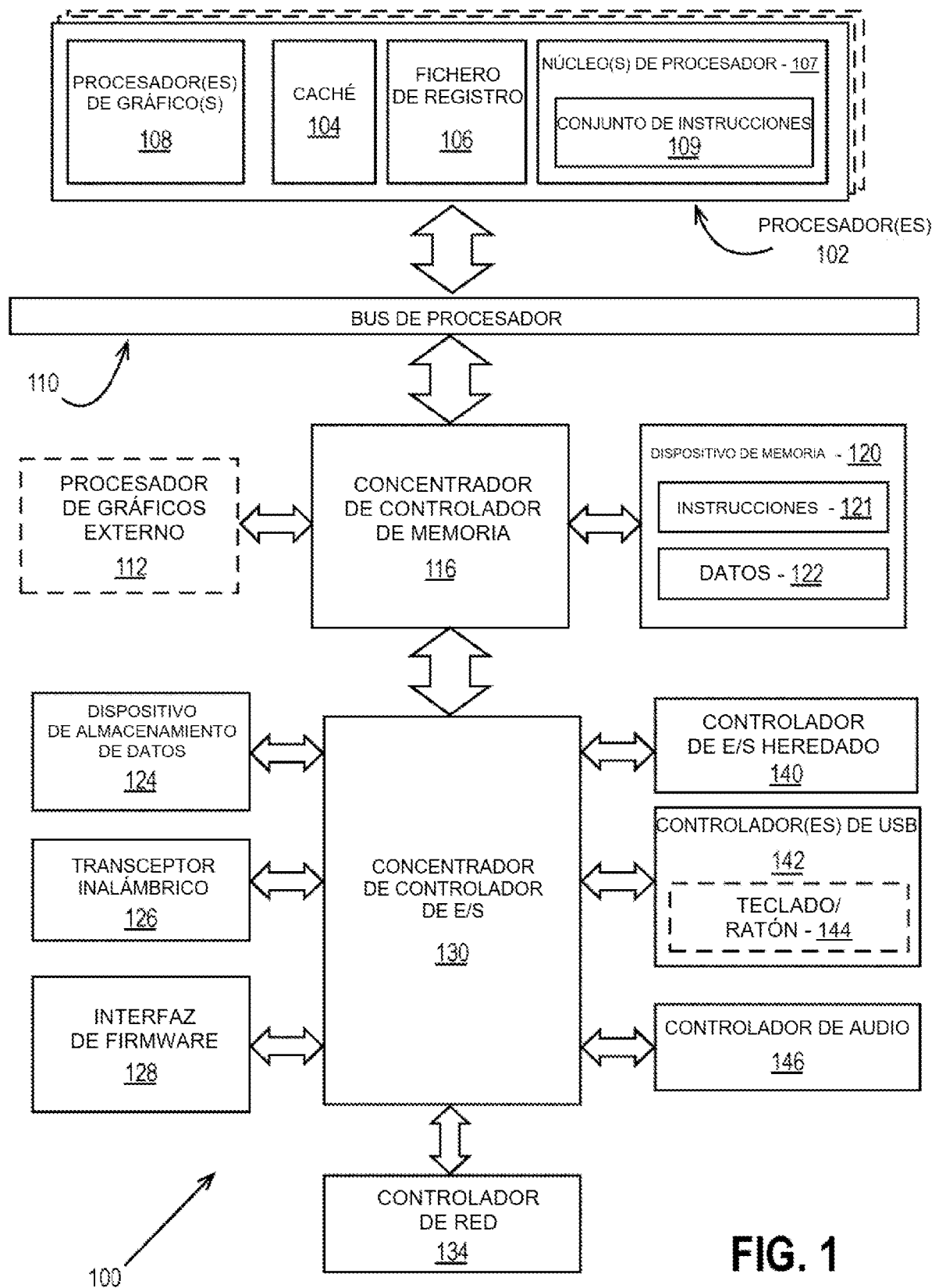


FIG. 1

PROCESADOR 200

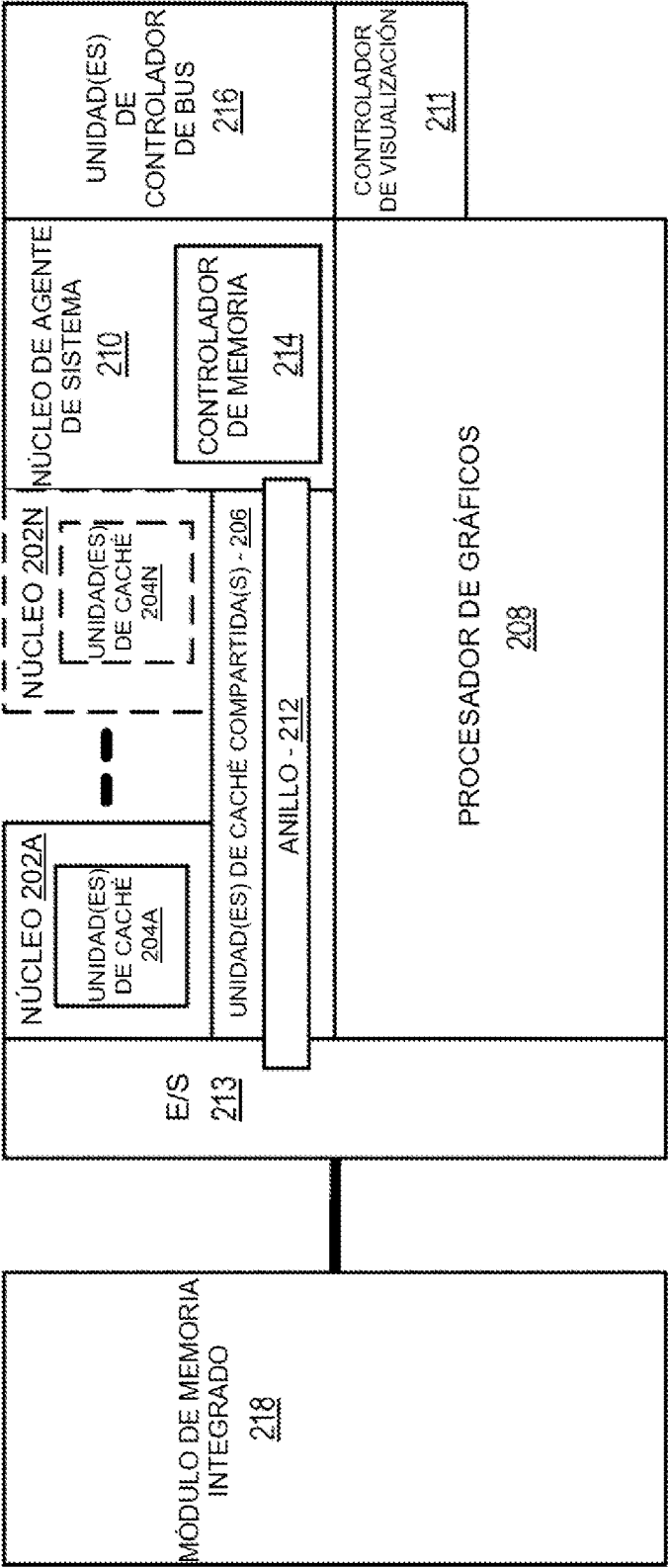


FIG. 2

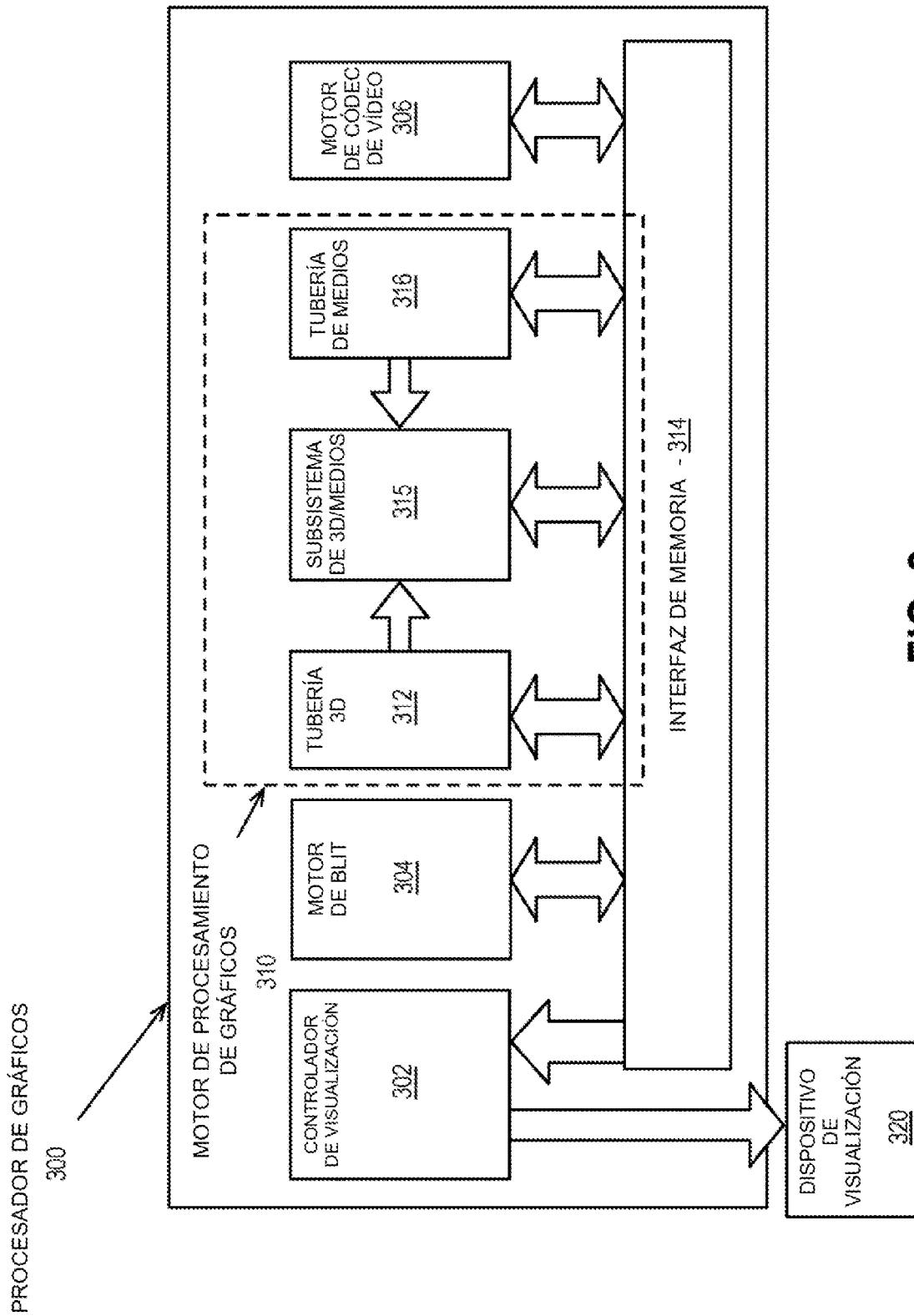


FIG. 3

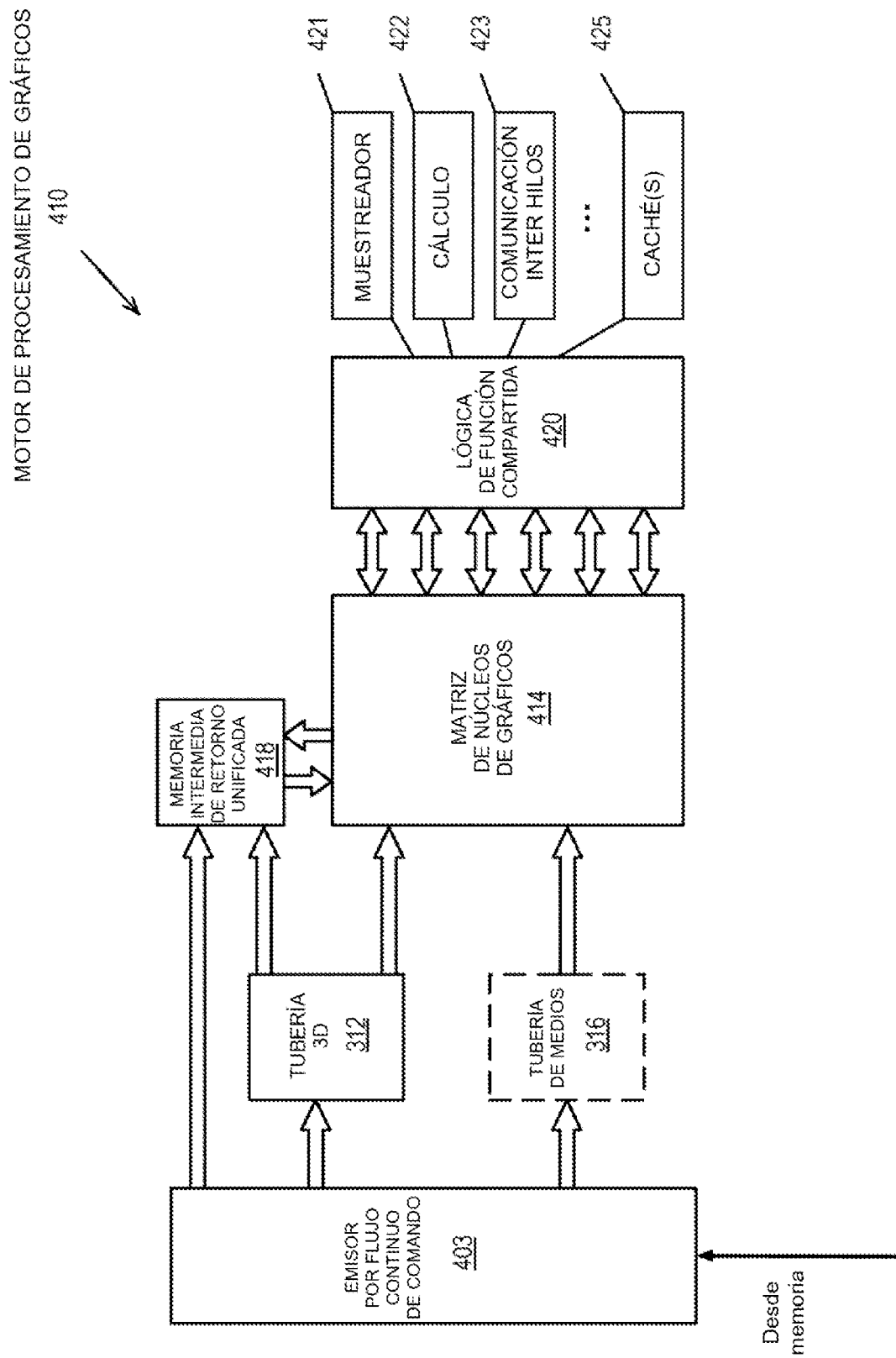


FIG. 4

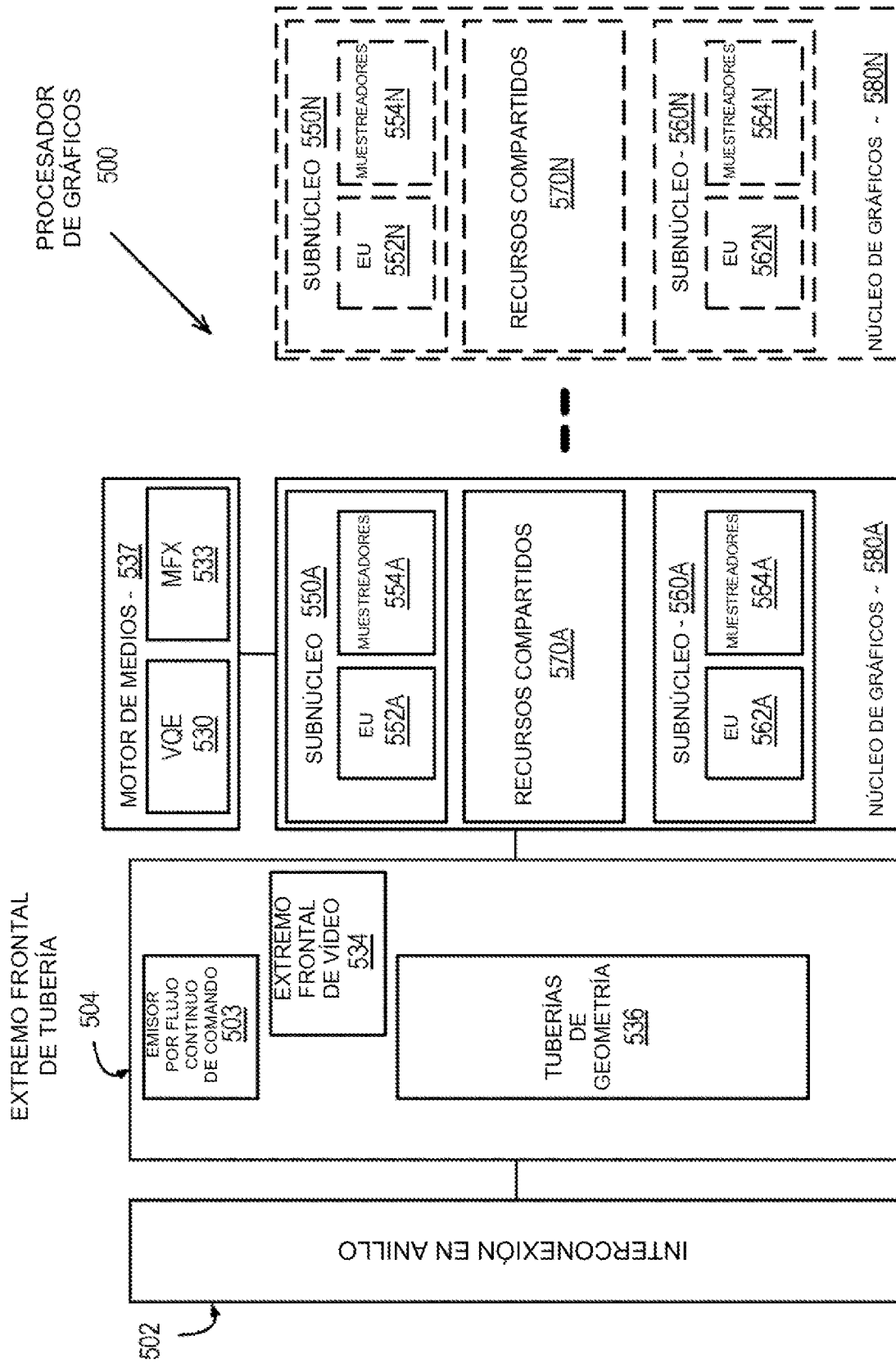


FIG. 5

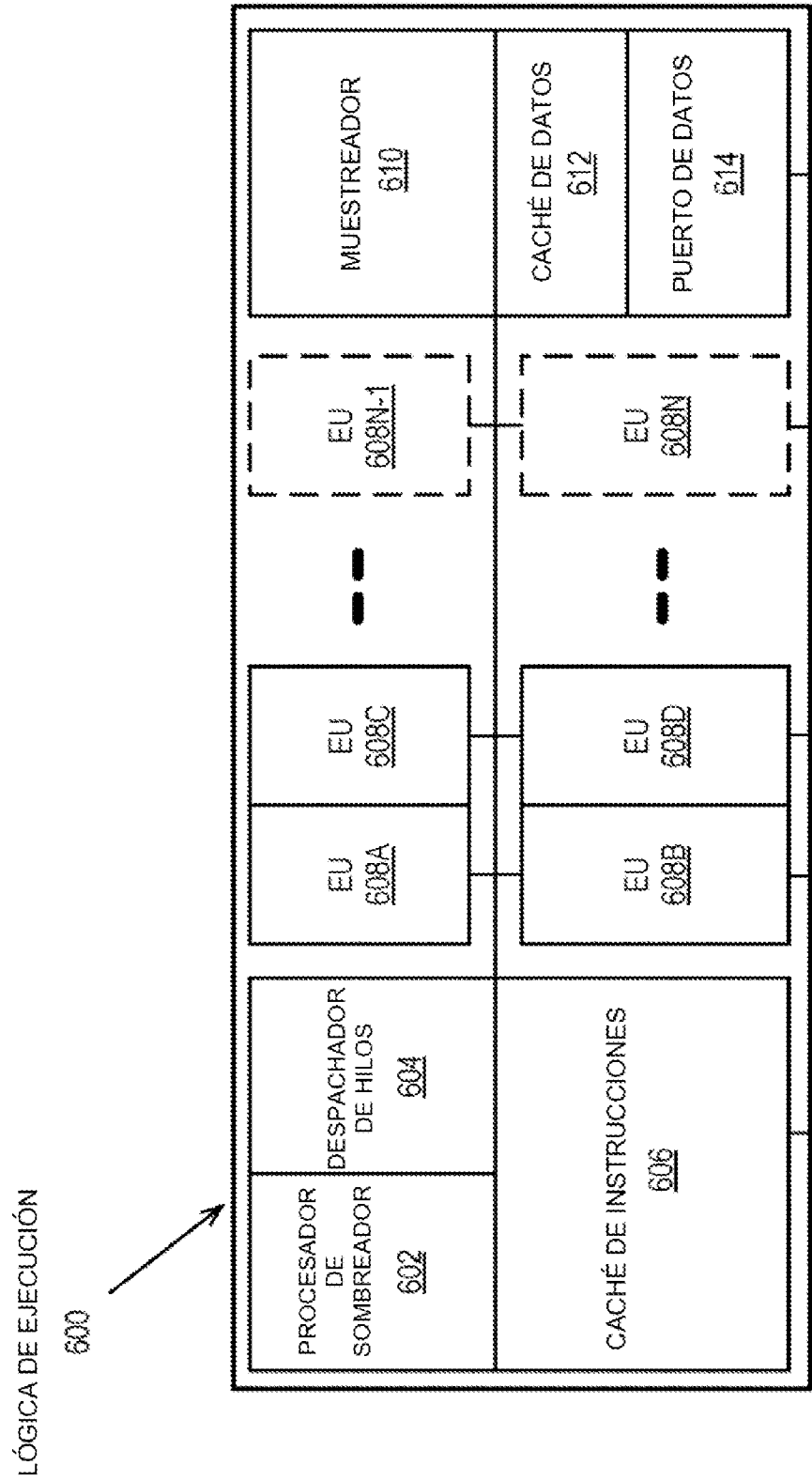


FIG. 6

FORMATOS DE INSTRUCCIÓN DE PROCESADOR DE GRÁFICOS

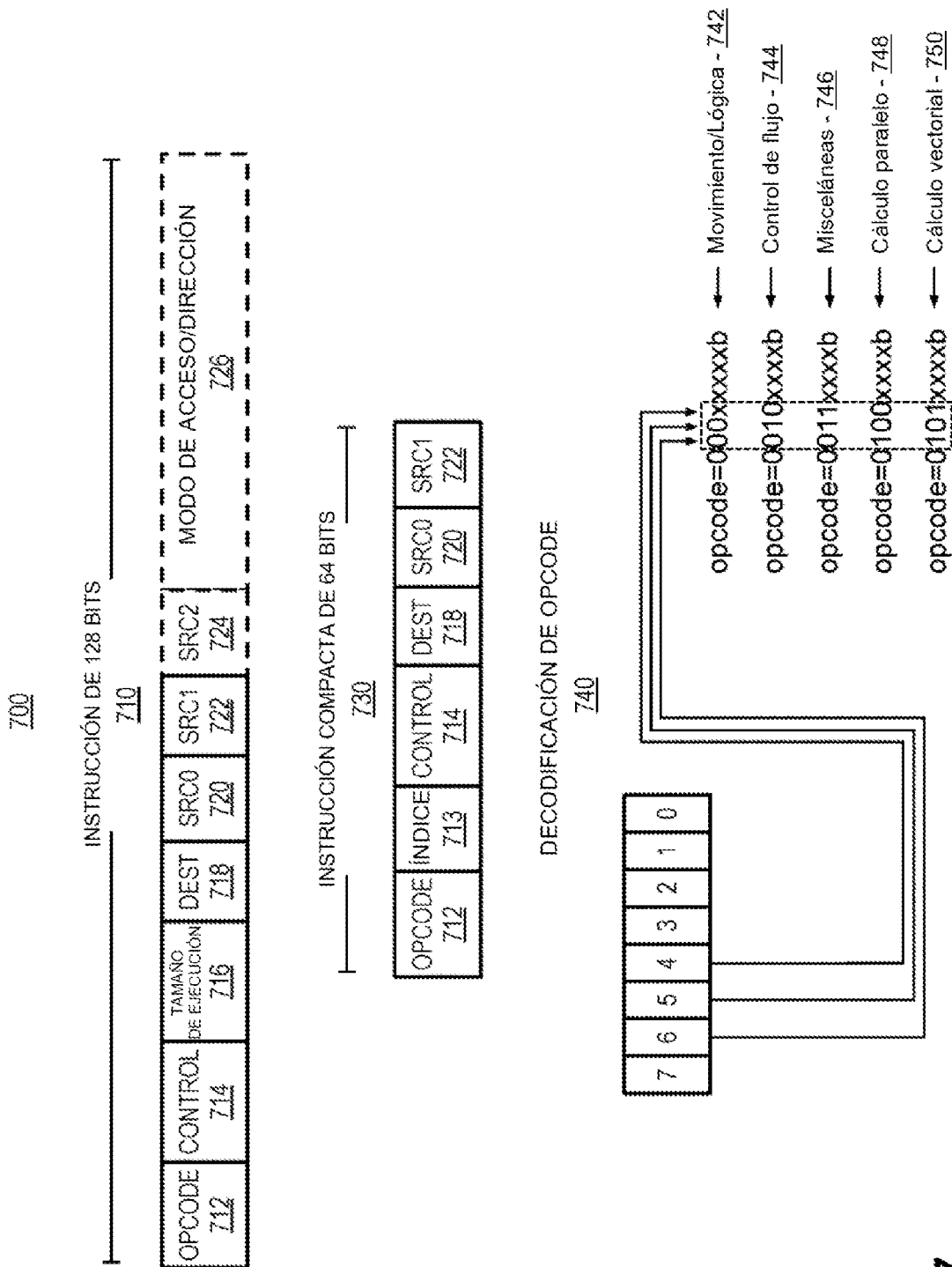


FIG. 7

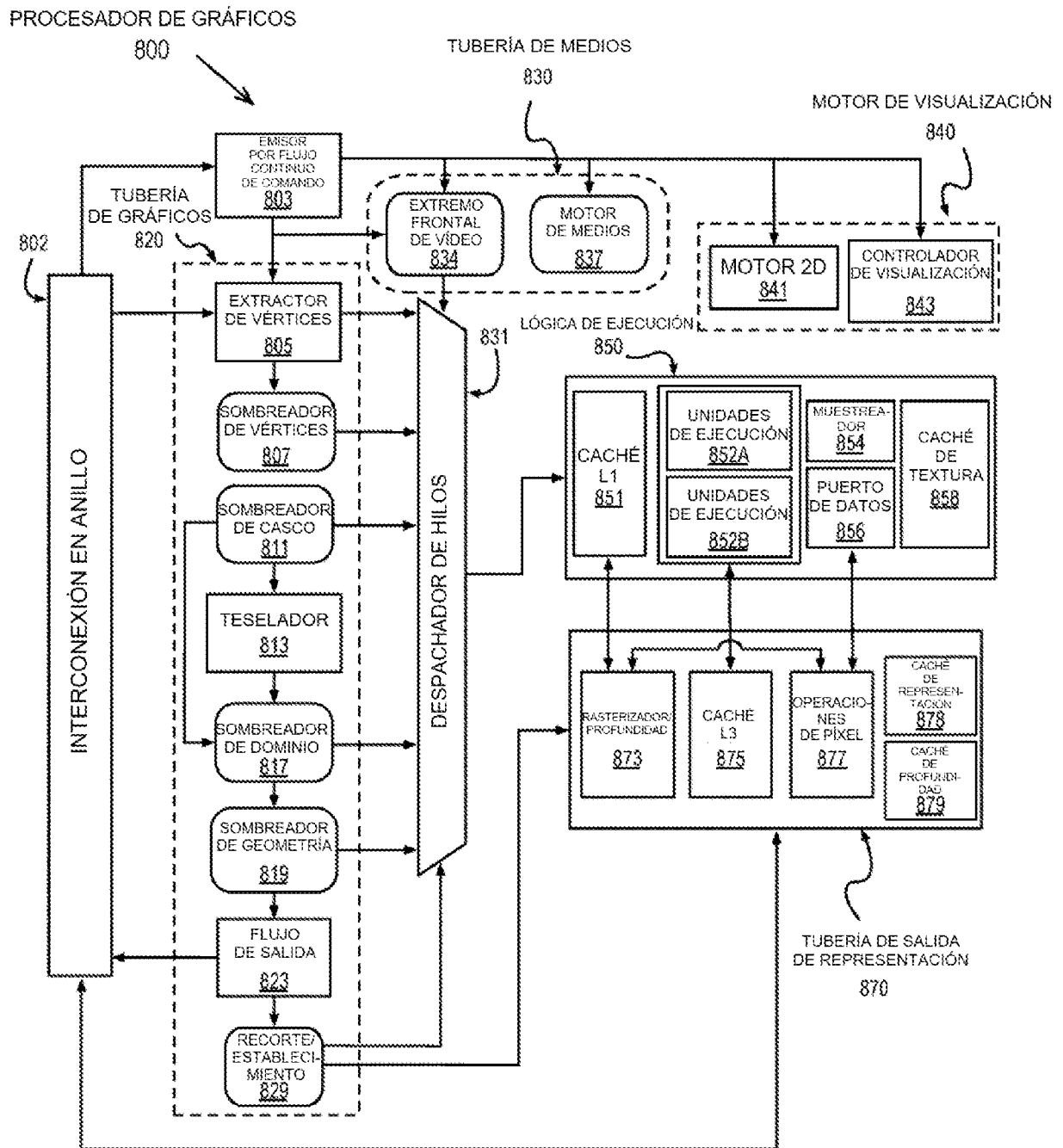


FIG. 8

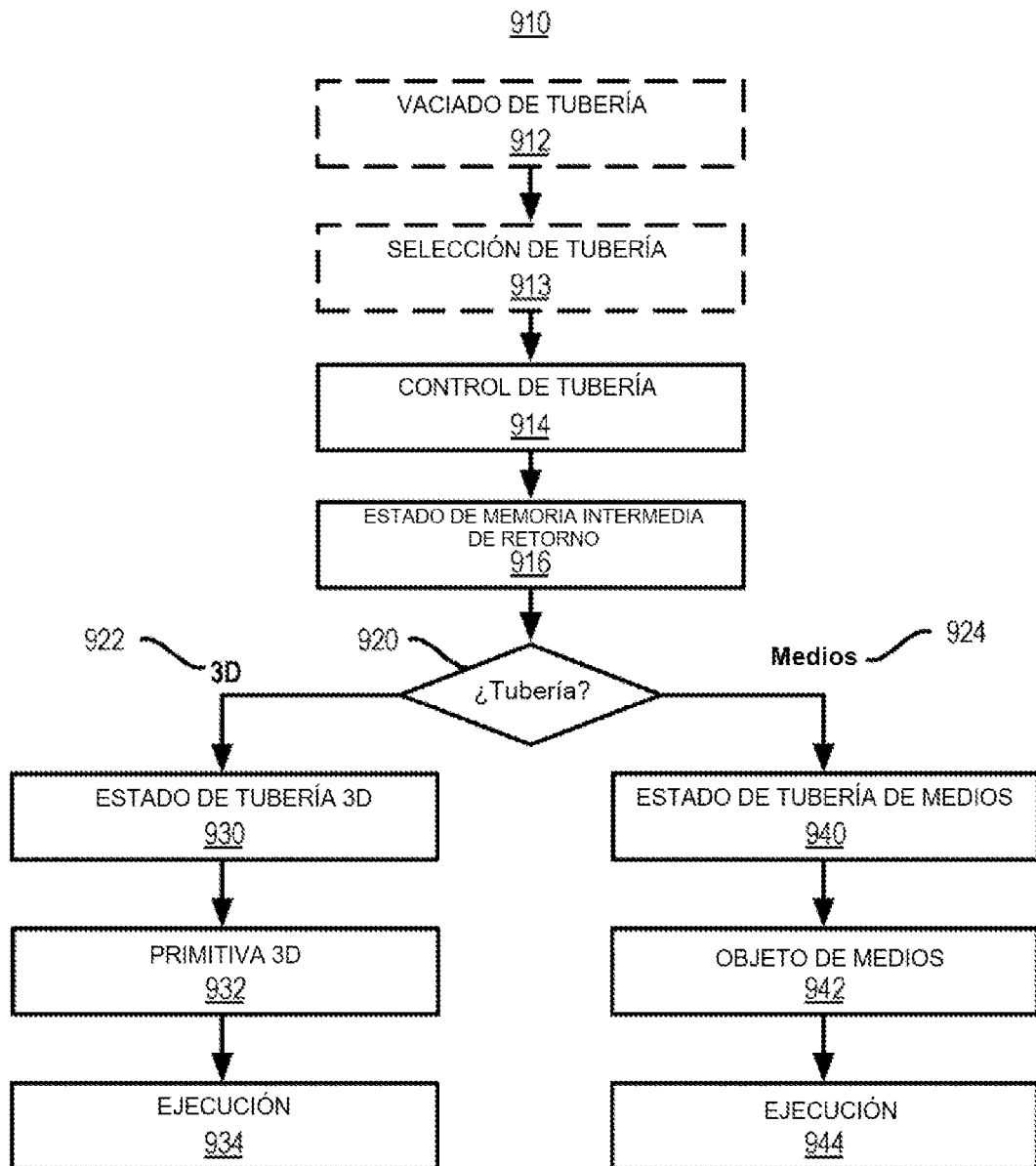
FIG. 9A

FORMATO DE COMANDO DE PROCESADOR DE GRÁFICOS



FIG. 9B

SECUENCIA DE COMANDOS DE PROCESADOR DE GRÁFICOS



SISTEMA DE PROCESAMIENTO DE DATOS - 1000

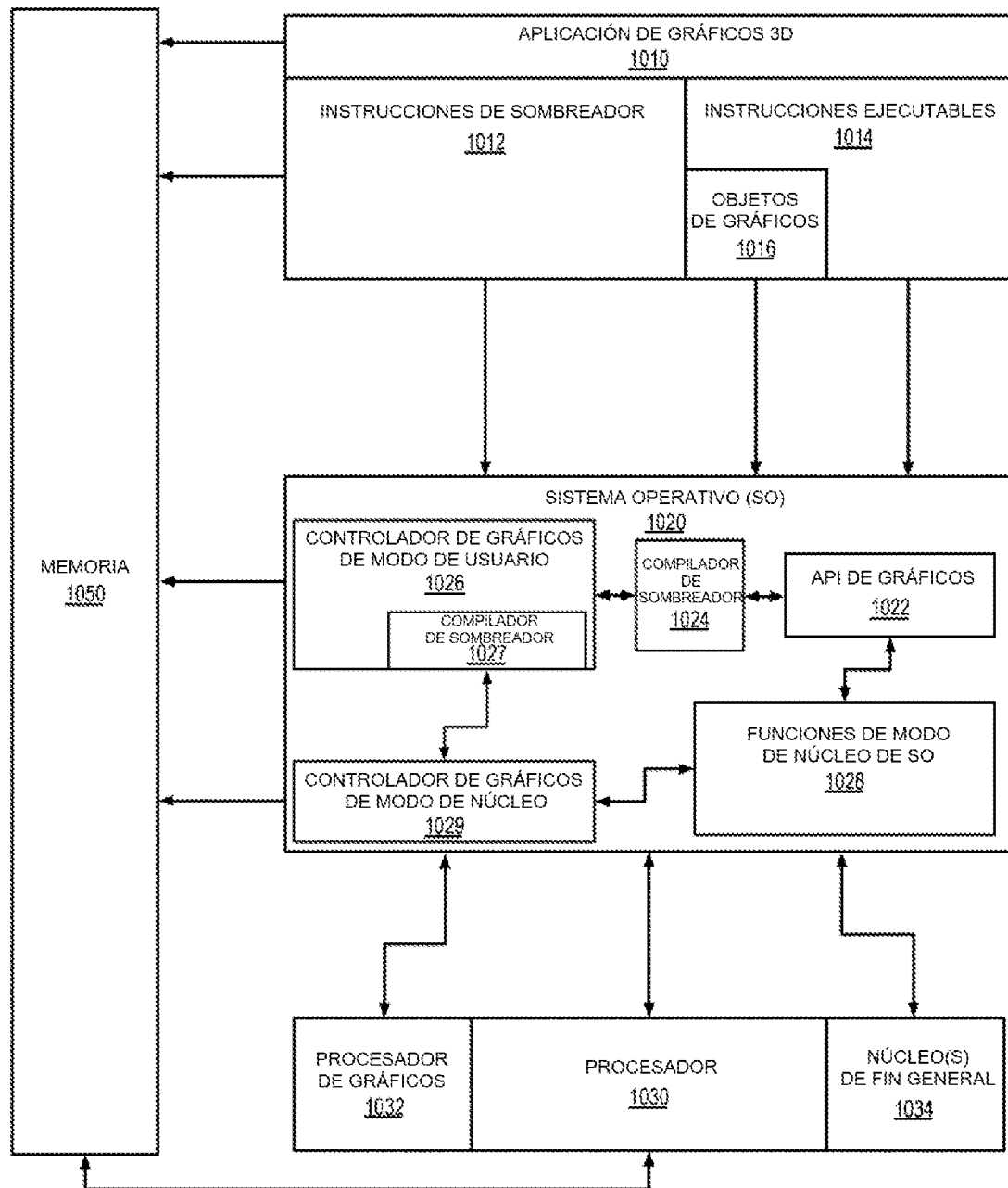


FIG. 10

DESARROLLO DE NÚCLEO DE IP - 1100

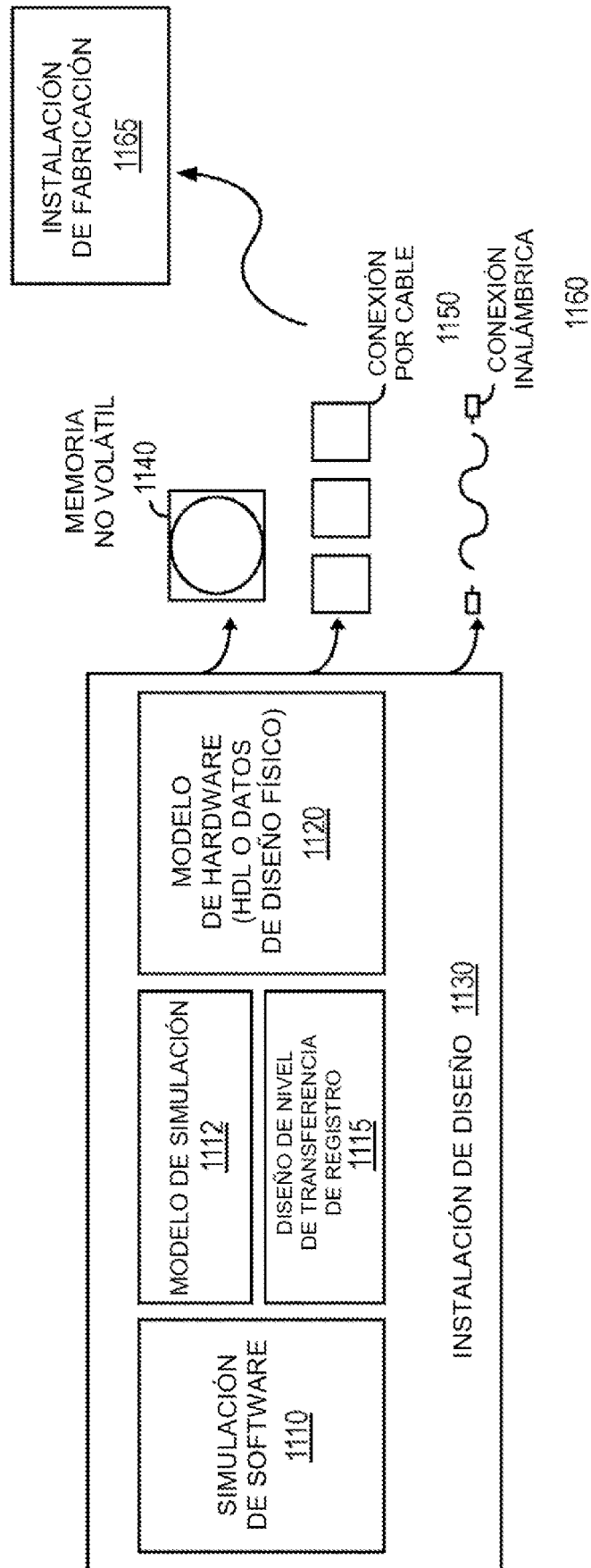


FIG. 11

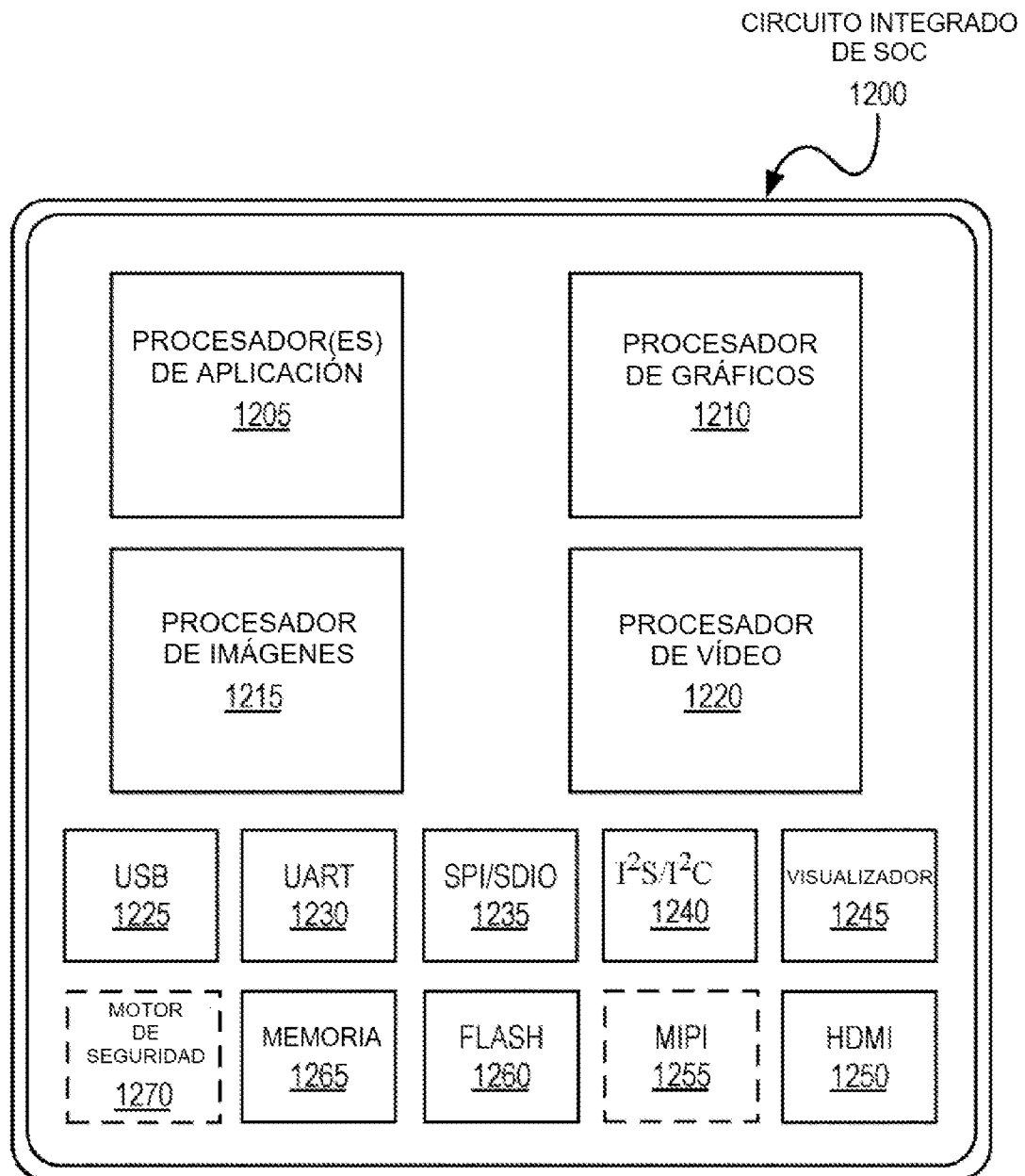


FIG. 12

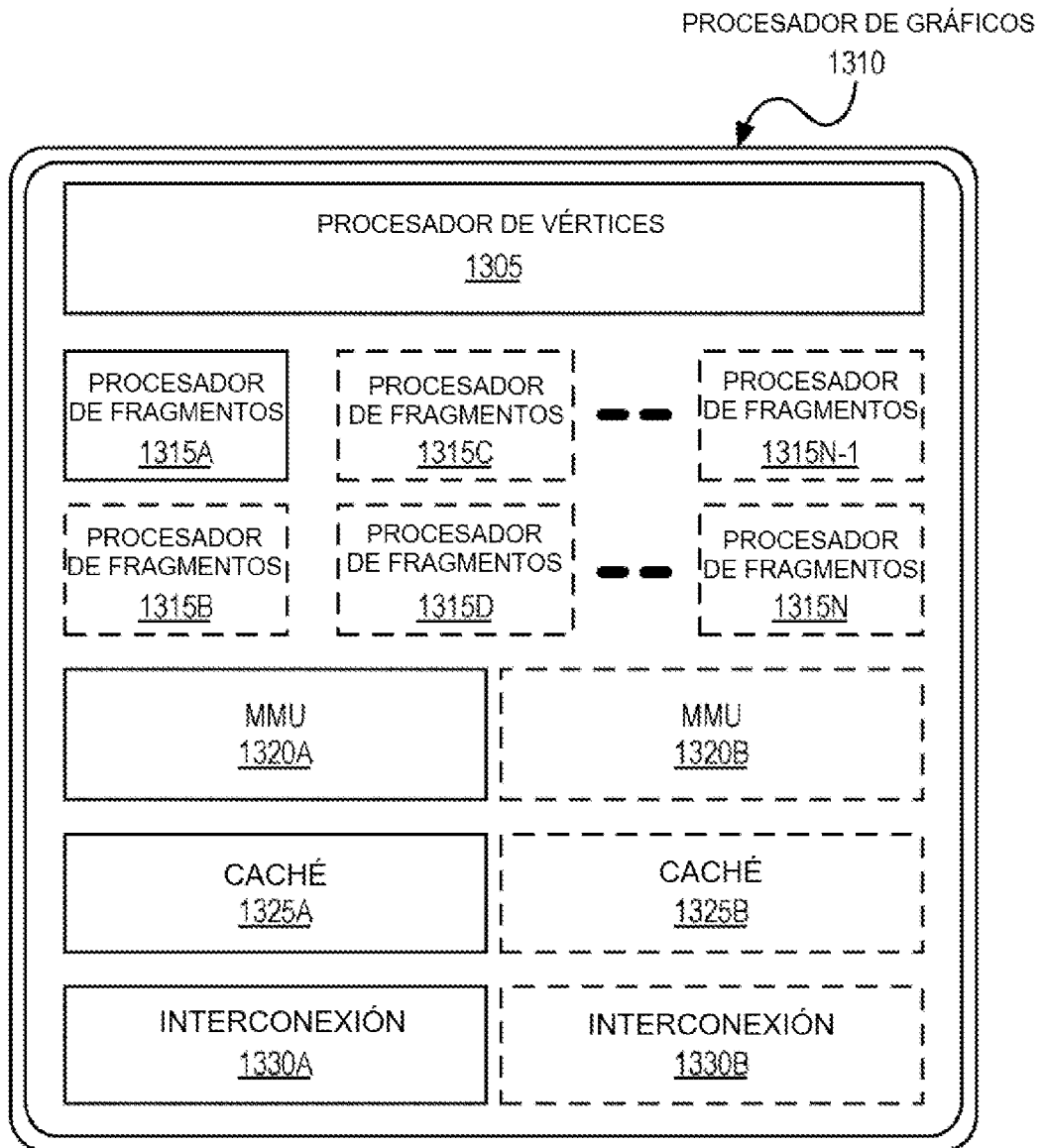


FIG. 13

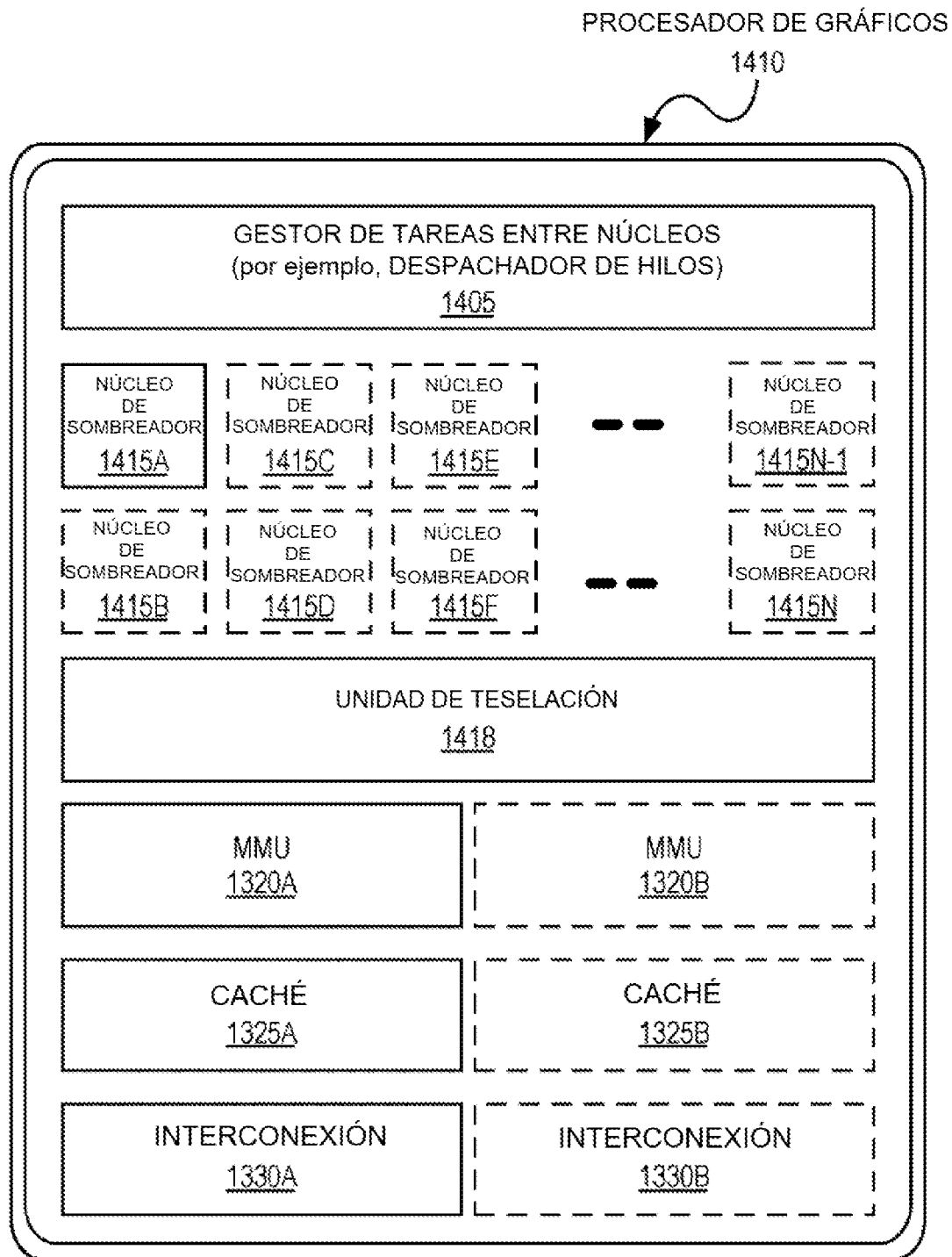


FIG. 14

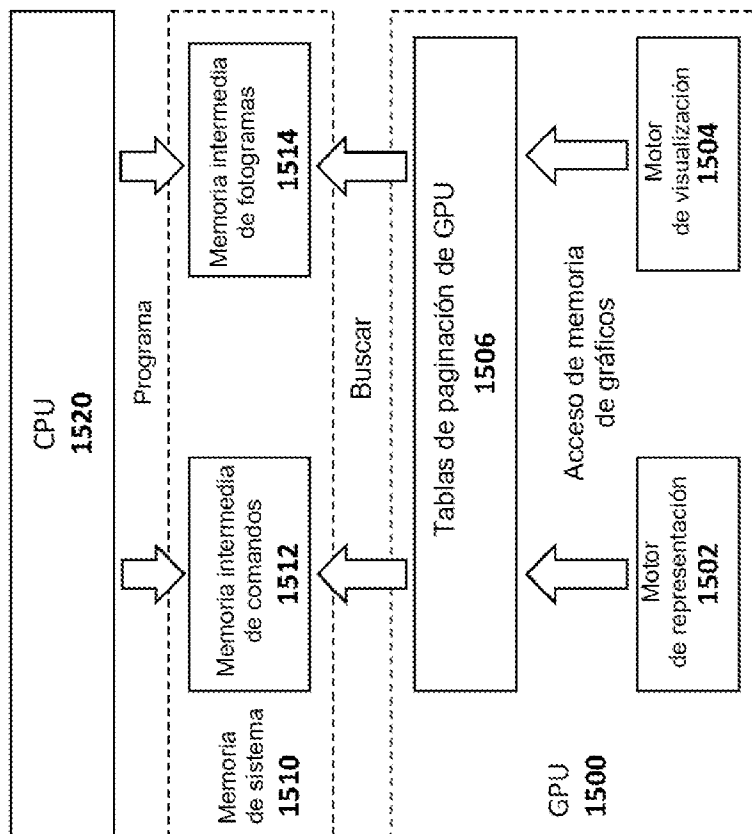


FIG. 15

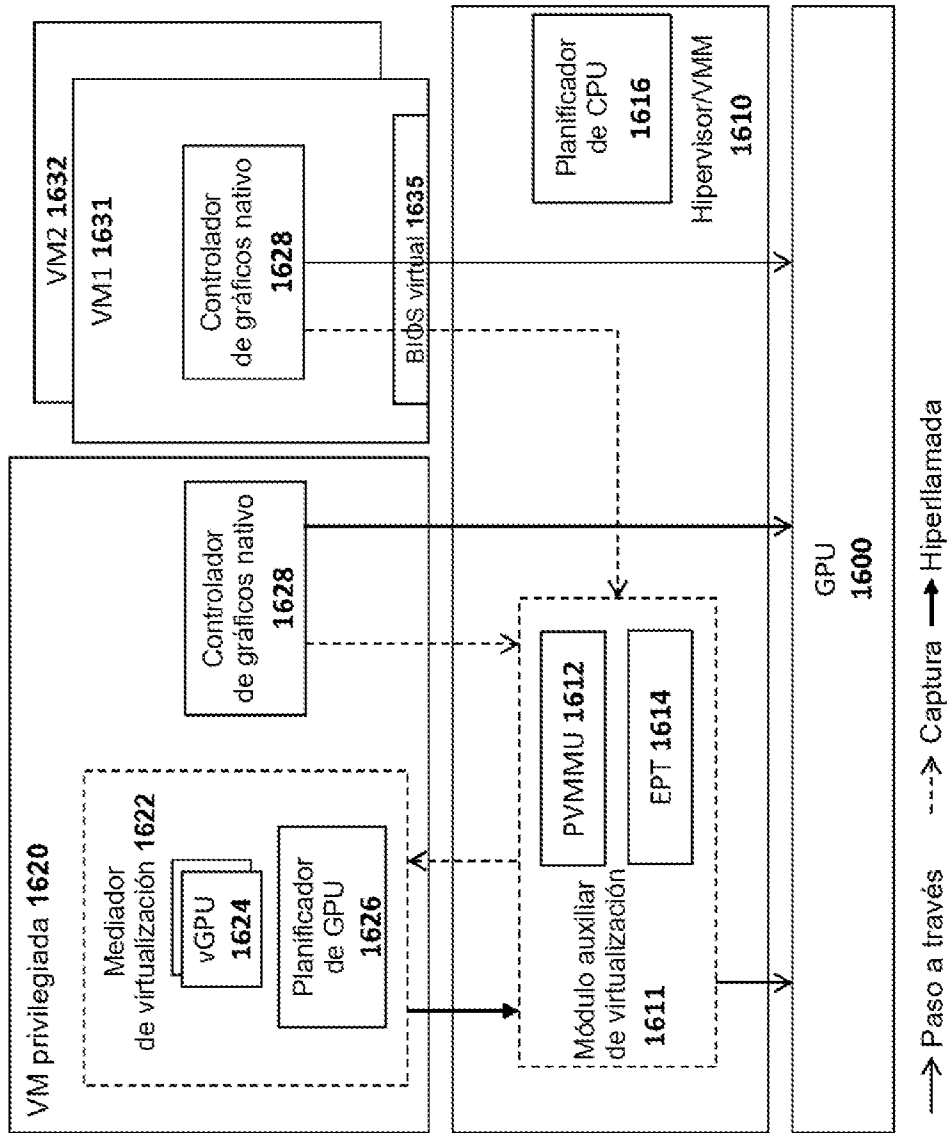


FIG. 16

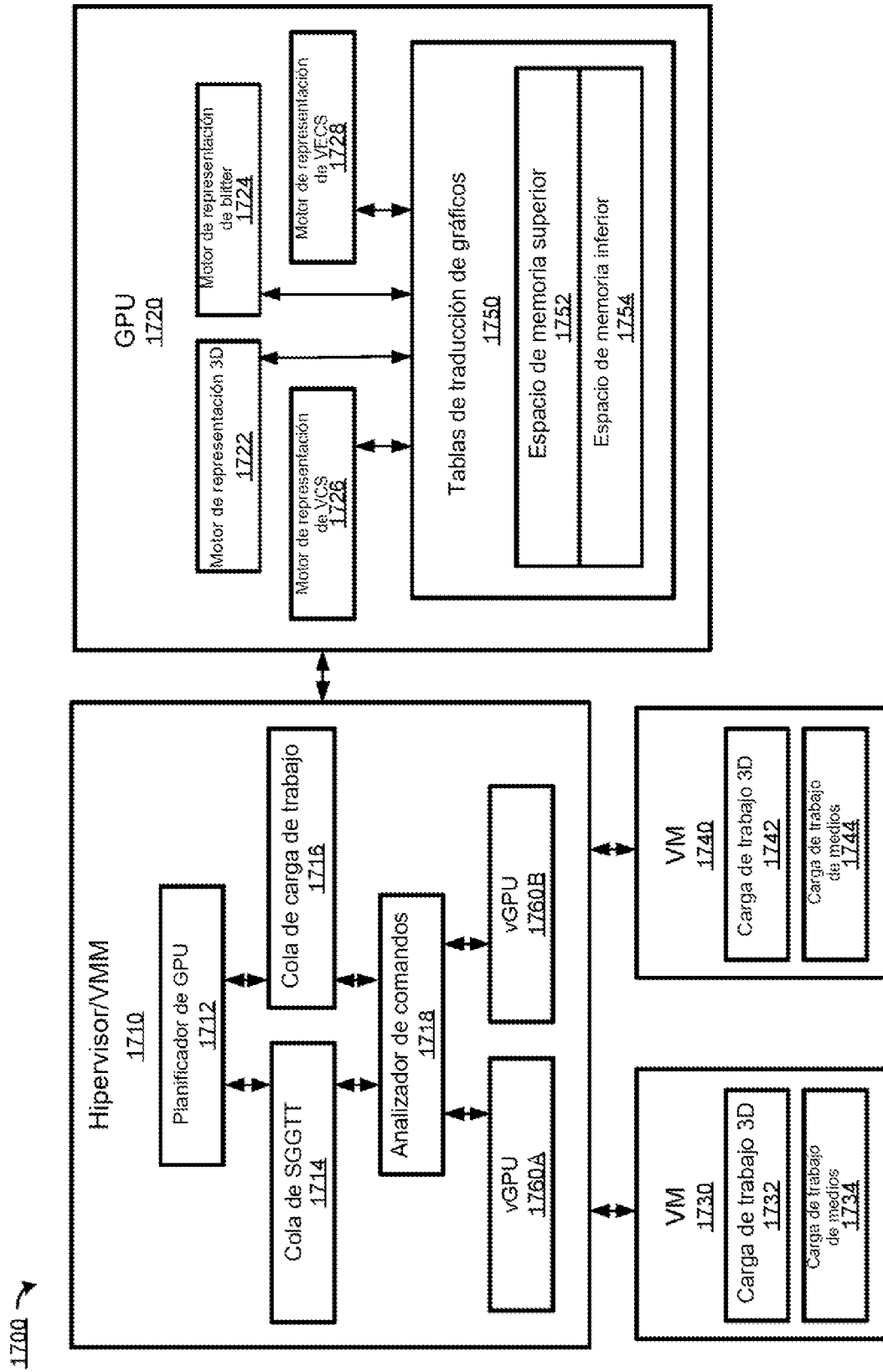


Fig. 17

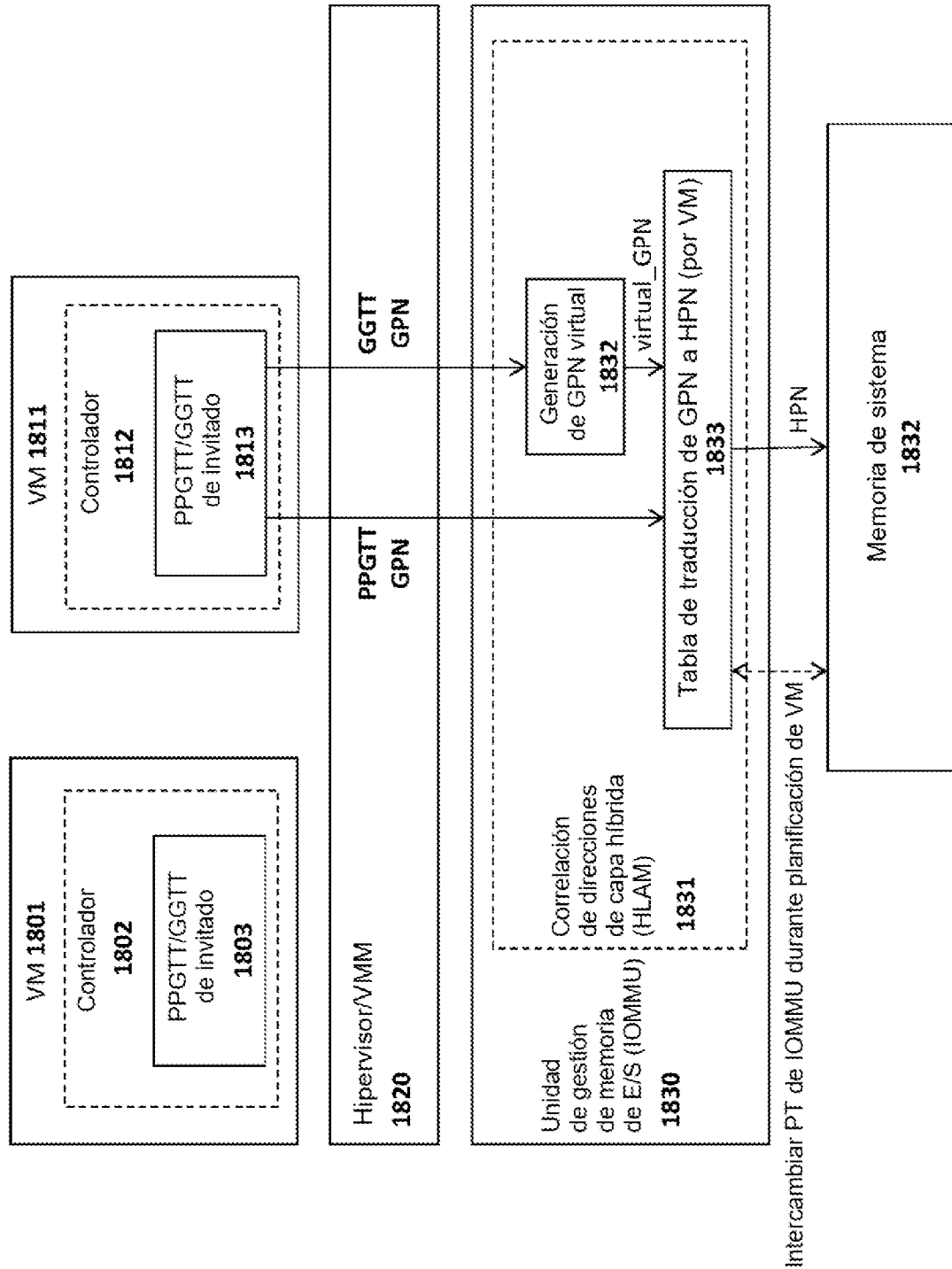


FIG. 18

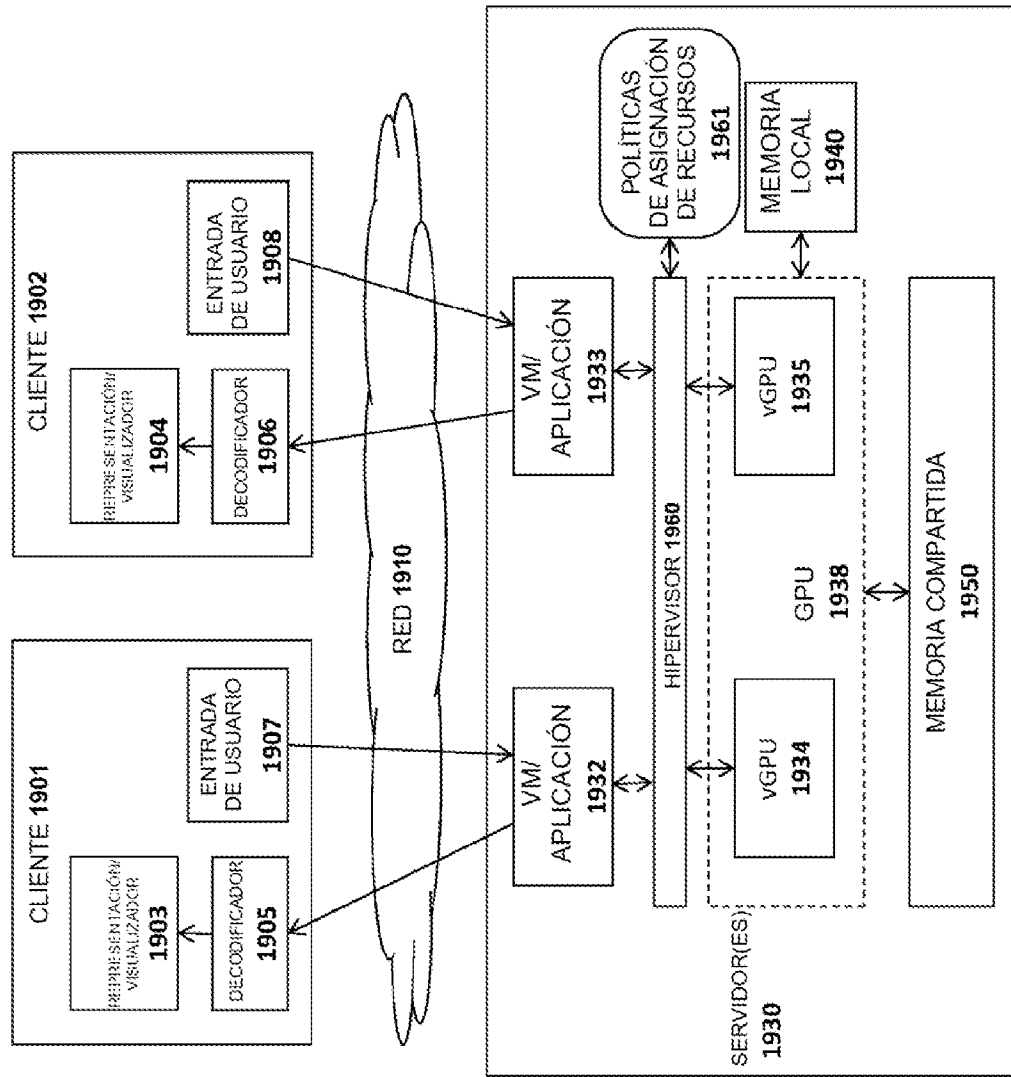


FIG. 19

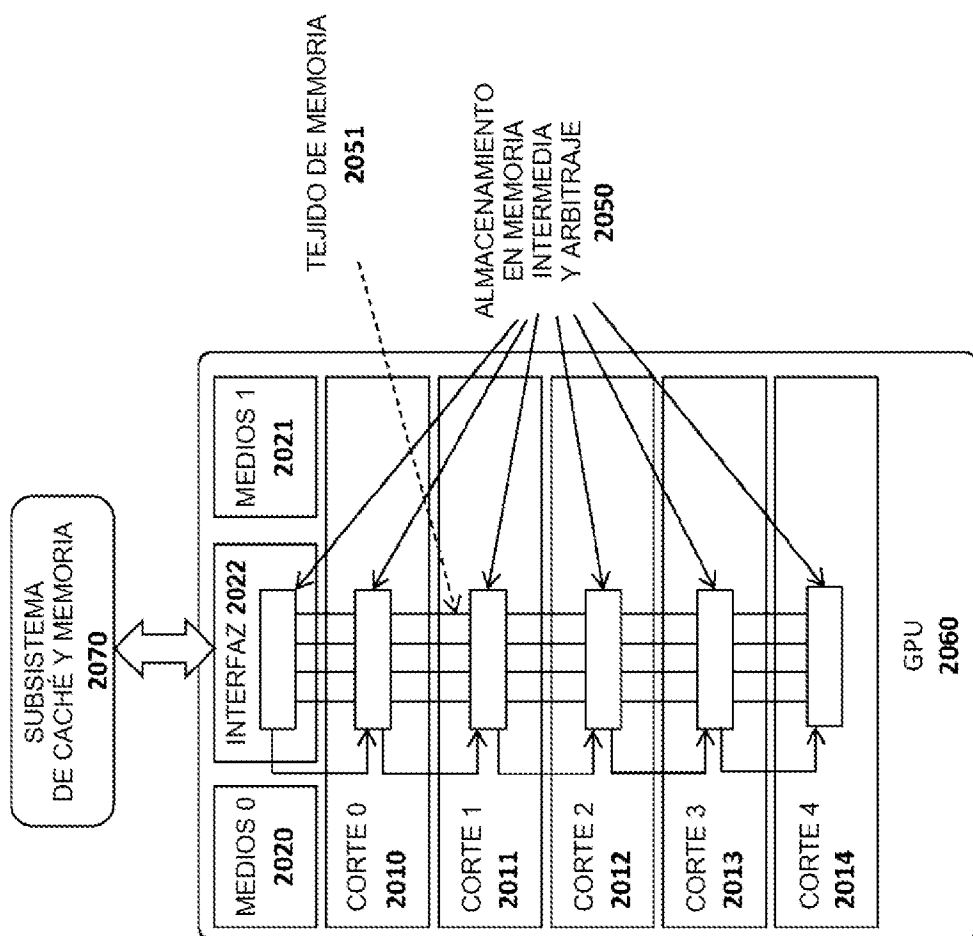


FIG. 20

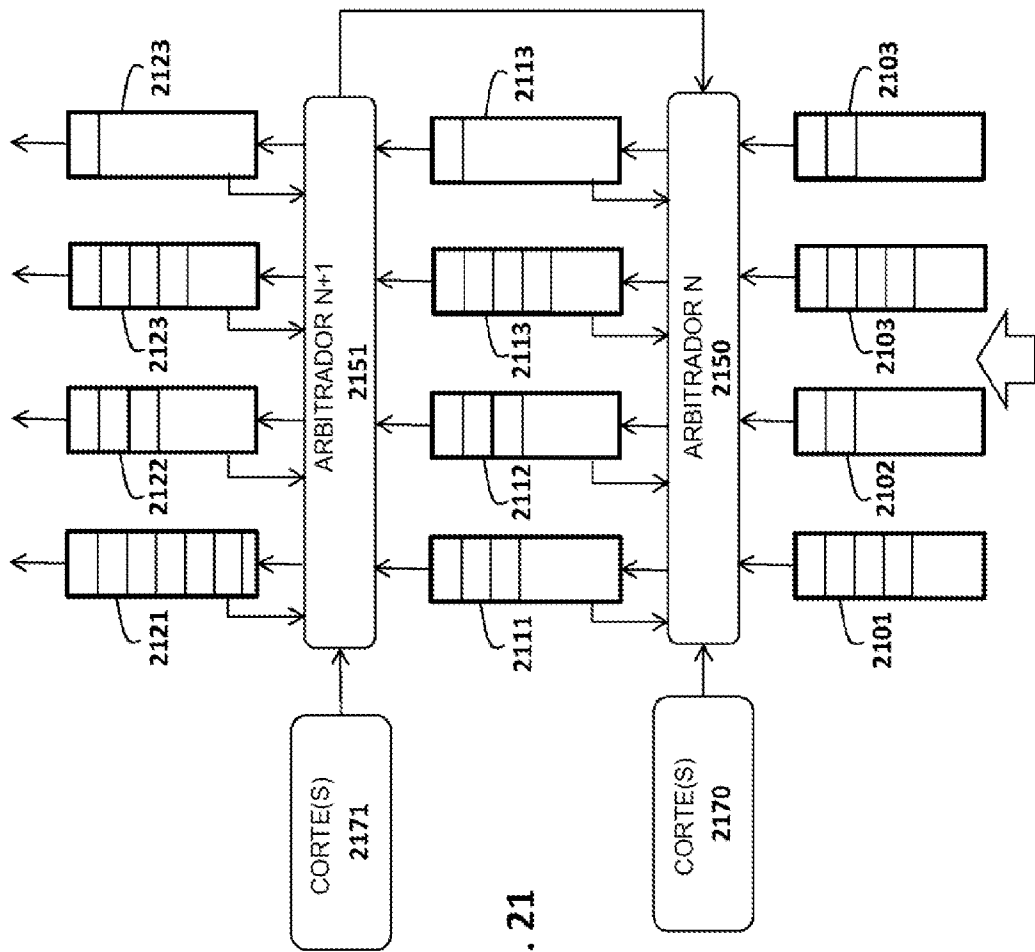


FIG. 21

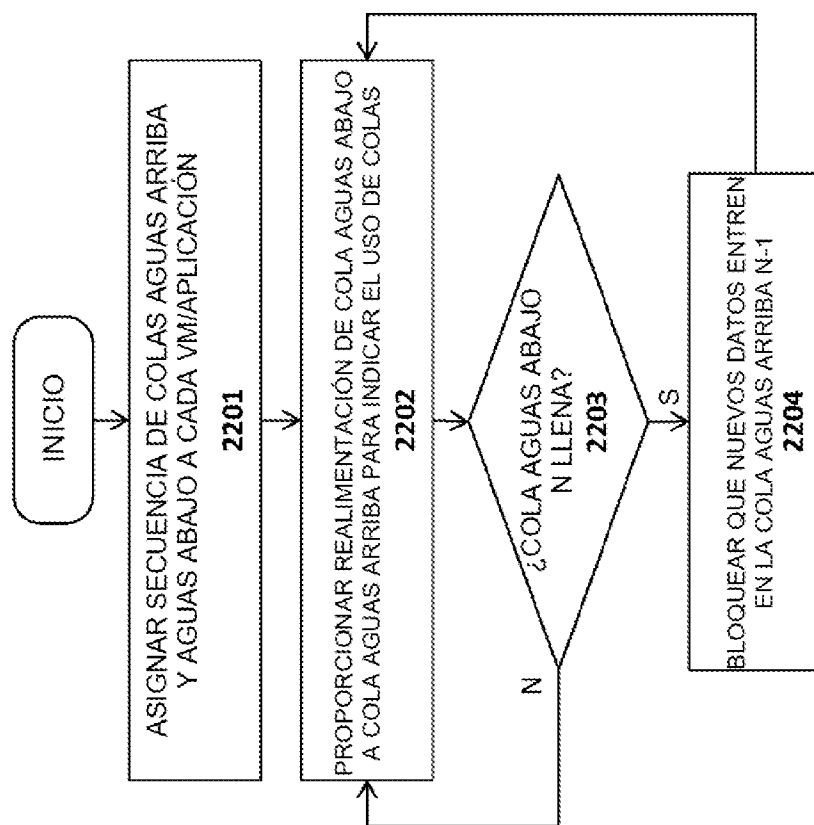
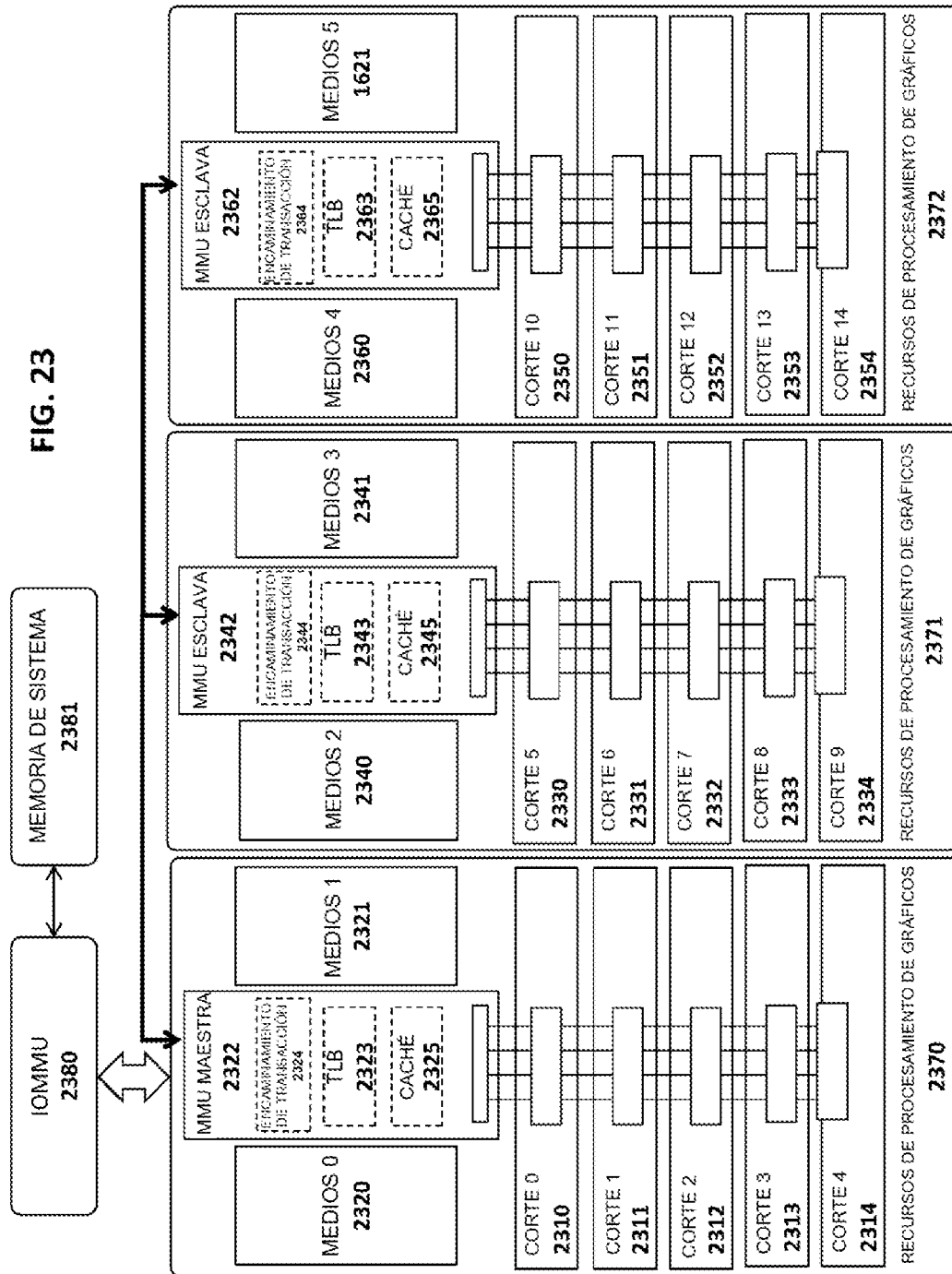


FIG. 22



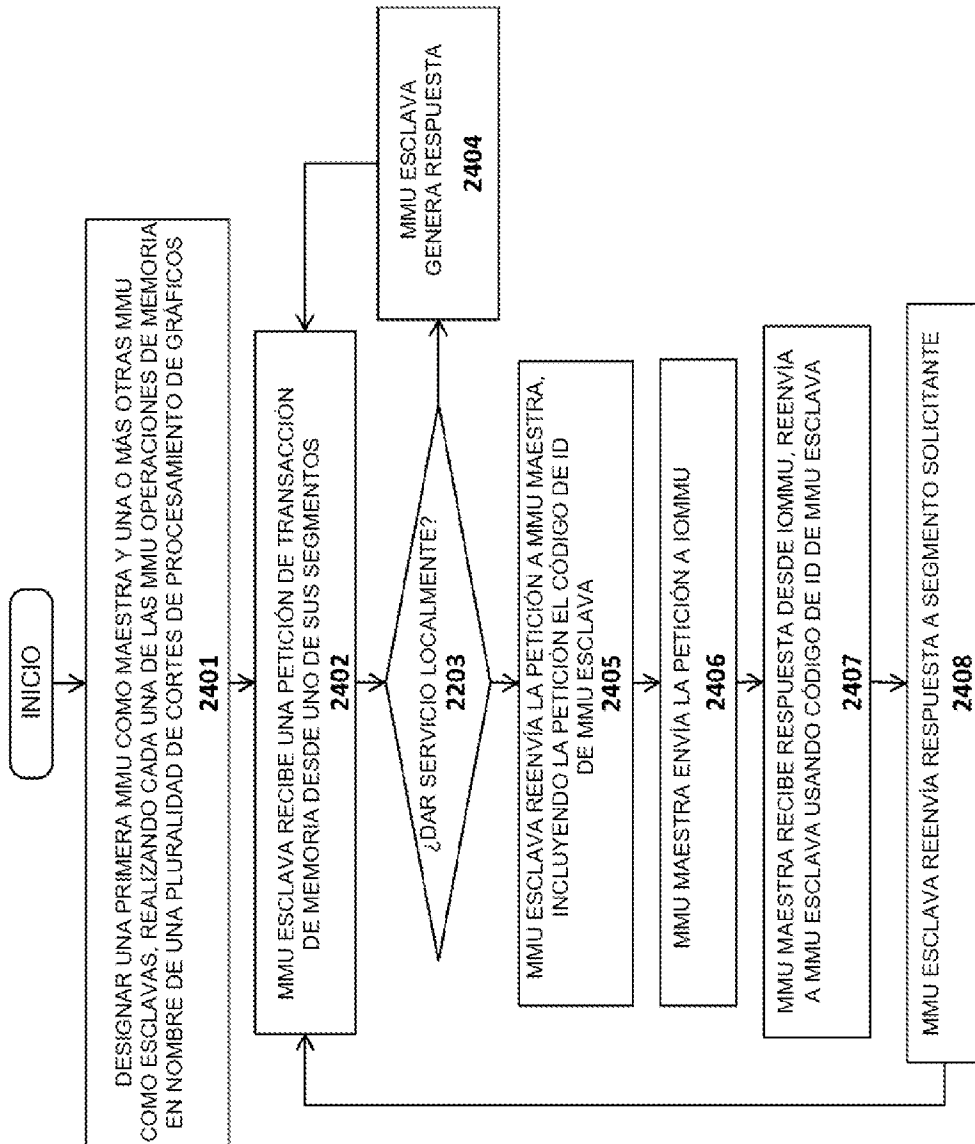


FIG. 24

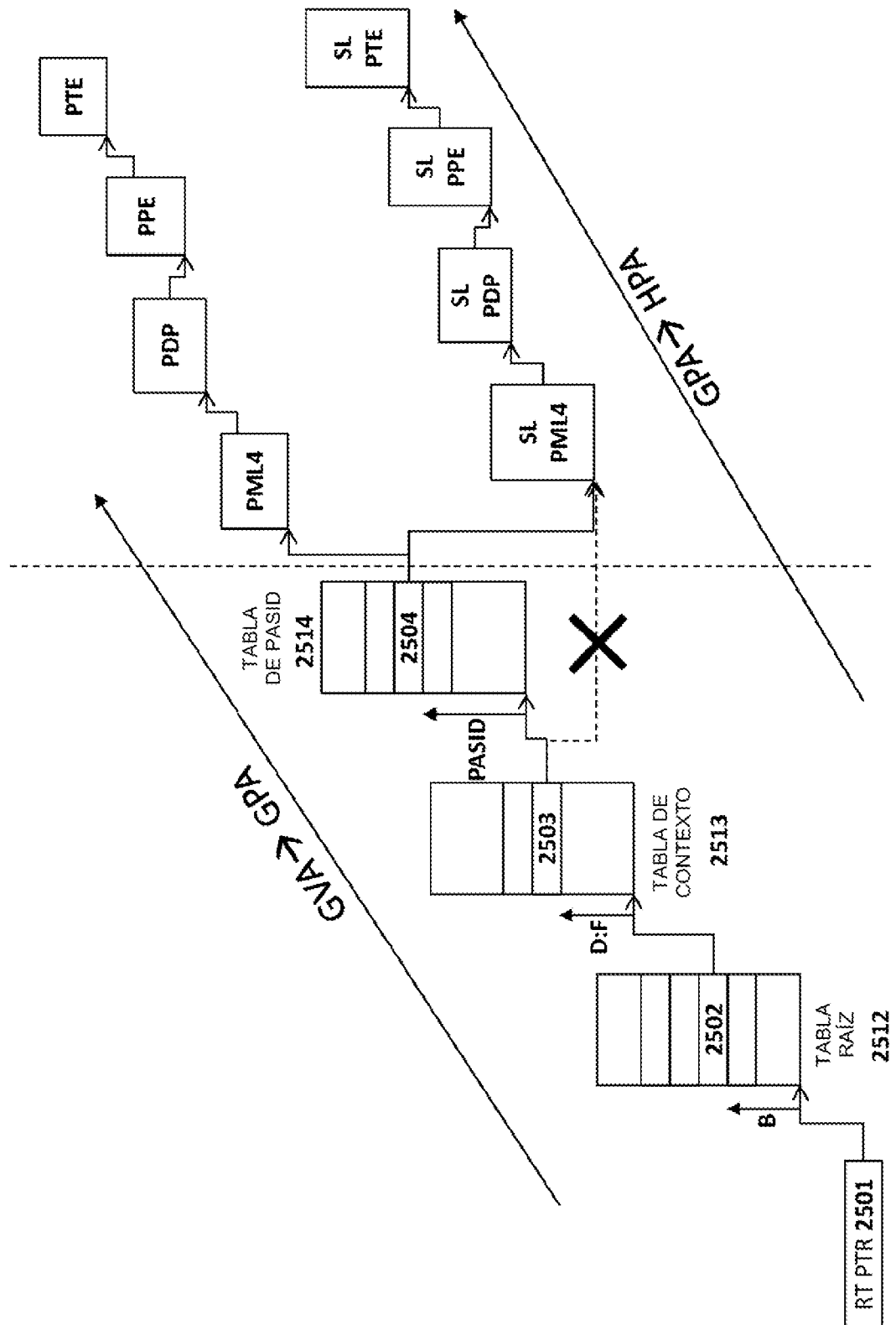


FIG. 25

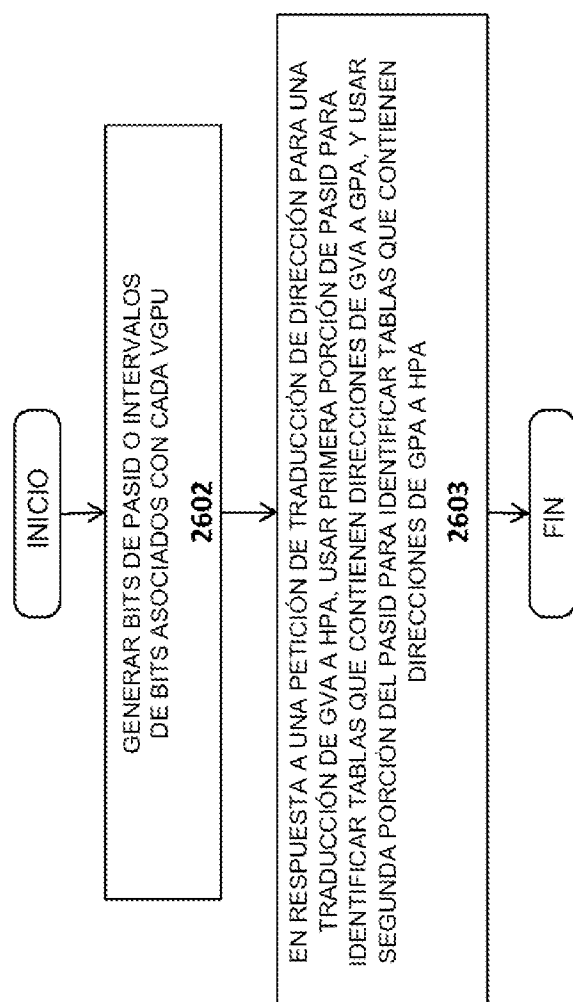


FIG. 26

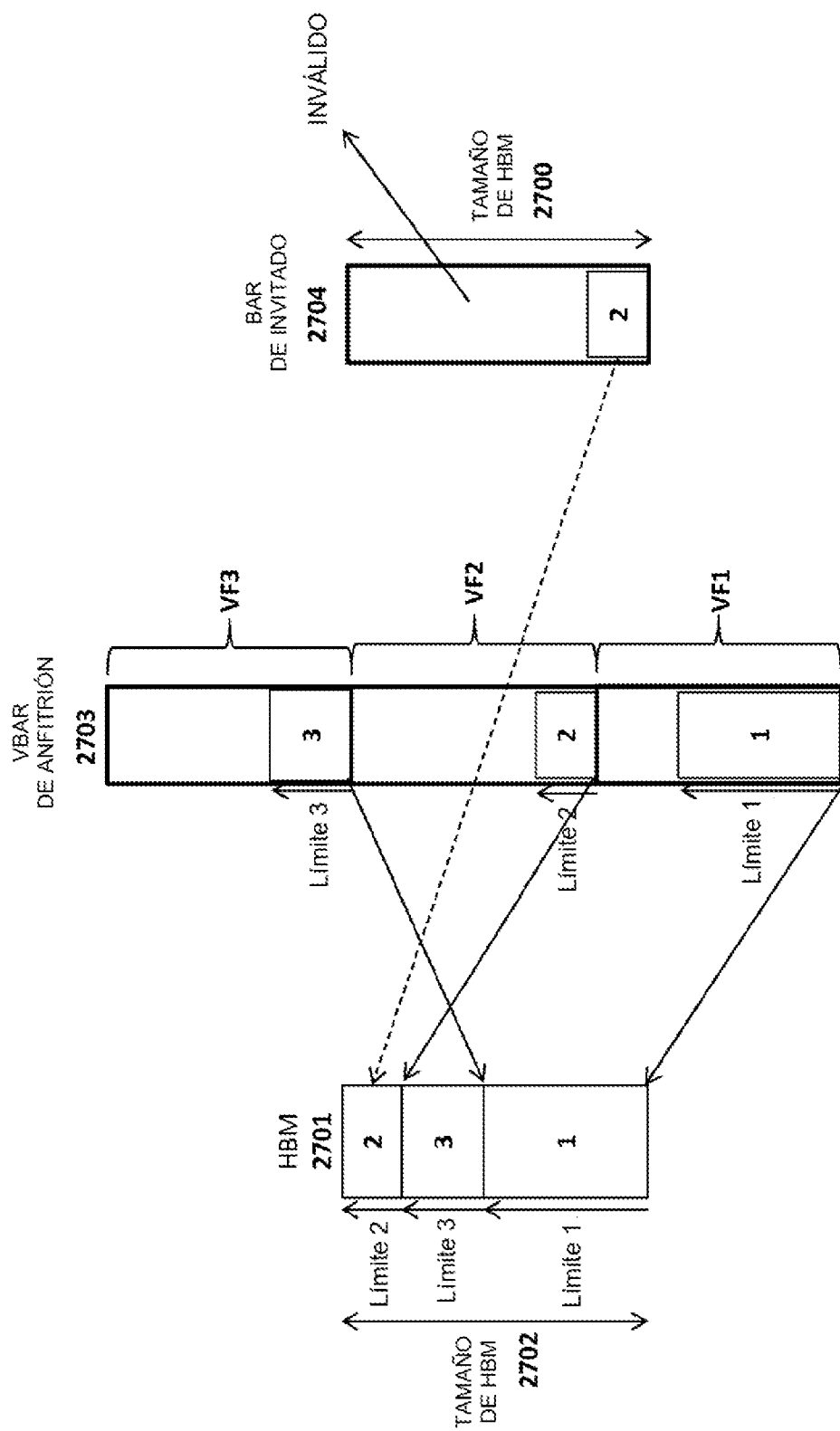


FIG. 27

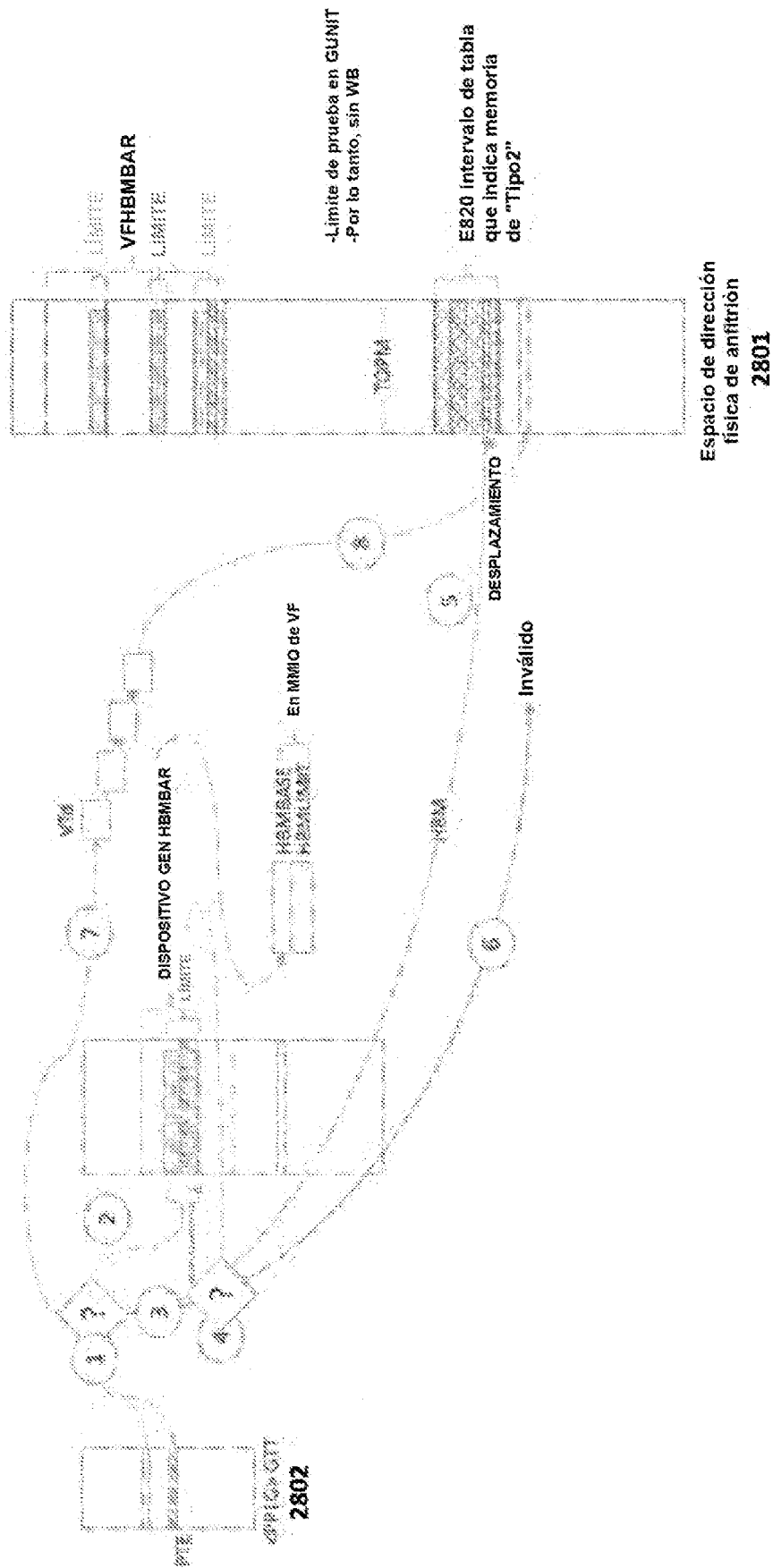


FIG. 28

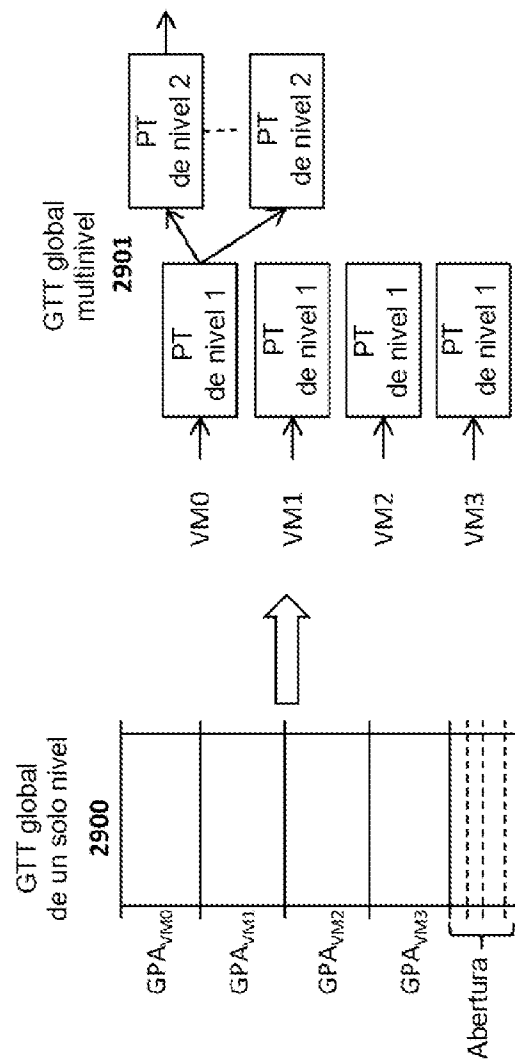


FIG. 29

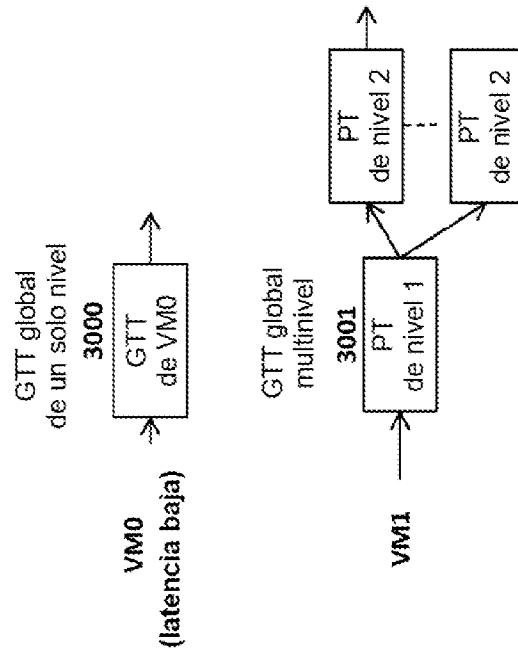


FIG. 30