

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 975 378**

51 Int. Cl.:

G06F 9/50 (2006.01)

H04W 4/40 (2008.01)

H04L 67/12 (2012.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **19.03.2018** **E 18162597 (1)**

97 Fecha y número de publicación de la concesión europea: **03.01.2024** **EP 3401786**

54 Título: **Detección y respuesta avanzadas de vehículos autónomos**

30 Prioridad:

17.04.2017 US 201715488914

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

04.07.2024

73 Titular/es:

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

72 Inventor/es:

**LAKSHMANAN, BARATH;
HURD, LINDA L.;
ASHBAUGH, BEN J.;
OULD-AHMED-VALL, ELMOUSTAPHA;
MA, LIWEI;
JIN, JINGYI;
GOTTSCHLICH, JUSTIN E.;
SAKTHIVEL, CHANDRASEKARAN;
STRICKLAND, MICHAEL S.;
LEWIS, BRIAN T.;
KUPER, LINDSEY;
KOKER, ALTUG;
APPU, ABHISHEK R.;
SURTI, PRASOONKUMAR;
RAY, JOYDEEP;
VEMBU, BALAJI;
TUREK, JAVIER S. y
FAROOQUI, NAILA**

74 Agente/Representante:

LEHMANN NOVO, María Isabel

ES 2 975 378 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

Detección y respuesta avanzadas de vehículos autónomos

5 **CAMPO**

Las realizaciones se refieren en general al procesamiento de datos y, más particularmente, al procesamiento de datos mediante una unidad de procesamiento de gráficos de propósito general.

10 **ANTECEDENTES DE LA DESCRIPCIÓN**

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

Para aumentar adicionalmente el rendimiento, los procesadores de gráficos típicamente implementan técnicas de procesamiento, tales como canalizaciones, que intentan procesar, en paralelo, tantos datos de gráficos como sea posible a lo largo de todas las diferentes partes de la canalización de gráficos. Los procesadores de gráficos paralelos con arquitecturas de múltiples hilos y única instrucción (SIMT) se diseñan para maximizar la cantidad de procesamiento paralelo en la canalización de gráficos. En una arquitectura de SIMT, grupos de hilos paralelos intentan ejecutar instrucciones de programa de manera síncrona conjuntamente tan a menudo como sea posible para aumentar la eficiencia de procesamiento. Puede encontrarse una vista global general del software y hardware para arquitecturas SIMT en Shane Cook, CUDA Programming, capítulo 3, páginas 37-51 (2013) y/o Nicholas Wilt, CUDA Handbook, A Comprehensive Guide to GPU Programming, secciones 2.6.2 a 3.1.2 (junio de 2013).

El documento US 2015/331422A1 divulga un sistema de interfaz de vehículo autónomo, en donde cada nodo del sistema se implementa como un nodo distribuido para el procesamiento de datos independiente de datos de sensor de bajo nivel y/o datos de sistema de alto nivel. Los datos de sistema de alto nivel se abstraen de los datos de sensor de bajo nivel, lo que proporciona invariancia a la configuración del sistema en algoritmos de procesamiento de nivel superior. El sistema de interfaz de vehículo autónomo incluye al menos un bus en tiempo real para comunicaciones de datos de los datos de sensor de bajo nivel y los datos de sistema de alto nivel entre los nodos del sistema. El sistema también incluye una interfaz de programación de aplicaciones (API) configurada para que los nodos del sistema accedan a los datos de sensor de bajo nivel y a los datos de sistema de alto nivel que se publican en el bus en tiempo real y son accesibles a través de la API.

El documento US 2017/068550A1 divulga un asistente personal distribuido. Un dispositivo electrónico está configurado para transmitir y recibir datos, incluida una solicitud de usuario para un servicio del asistente personal. Se determina una tarea a realizar en respuesta a la solicitud del usuario y se estiman las características de rendimiento para completar la tarea con el dispositivo electrónico basándose en una heurística. Las características de rendimiento se utilizan para determinar si se debe ejecutar la tarea en el dispositivo electrónico o fuera del dispositivo electrónico. Si la tarea se va a ejecutar en el exterior, se genera y transmite un código ejecutable para llevar a cabo la tarea.

Gil A. E. et al.: "Cooperative scheduling of tasks for networked uninhabited autonomous vehicles", Conferencia IEEE sobre Decisión y Control, 2003, divulga un modelo de control cooperativo para un conjunto de vehículos autónomos deshabitados (UAV). Antes de que comience una misión, se asigna una serie de tareas al conjunto de UAV. Después del despliegue, los UAV cooperan para decidir qué UAV del conjunto debe realizar la tarea individual.

La invención se define por un dispositivo informático dentro de un vehículo autónomo y un vehículo autónomo, un método y uno o más medios de almacenamiento legibles por máquina según las reivindicaciones independientes. Se definen realizaciones preferidas en las reivindicaciones dependientes.

55 **BREVE DESCRIPCIÓN DE LOS DIBUJOS**

De modo que las características de la presente invención puedan entenderse en detalle, puede obtenerse una descripción de la invención más particular haciendo referencia a las realizaciones, algunas de las cuales se ilustran en los dibujos adjuntos. Sin embargo, se ha de observar que los dibujos adjuntos ilustran únicamente realizaciones habituales y, por lo tanto, no han de considerarse limitantes del alcance de todas las realizaciones.

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

la **Figura 2A-2D** ilustra unos componentes de procesador paralelo, de acuerdo con una realización;

las **Figuras 3A-3B** son diagramas de bloques de multiprocesadores de gráficos, de acuerdo con las realizaciones;

las **Figuras 4A-4F** ilustran una arquitectura ilustrativa en la que una pluralidad de GPU están comunicativamente acopladas a una pluralidad de procesadores de múltiples núcleos;

la **Figura 5** ilustra una canalización de procesamiento de gráficos, de acuerdo con una realización;

la **Figura 6** ilustra una pila de software de aprendizaje automático, de acuerdo con una realización;

la **Figura 7** ilustra una unidad de procesamiento de gráficos de propósito general altamente paralela, de acuerdo con una realización;

la **Figura 8** ilustra un sistema informático de múltiples GPU, de acuerdo con una realización;

las **Figuras 9A-9B** ilustran capas de redes neuronales profundas ilustrativas;

la **Figura 10** ilustra una red neuronal recurrente ilustrativa;

la **Figura 11** ilustra el entrenamiento y despliegue de una red neuronal profunda;

la **Figura 12** es un diagrama de bloques que ilustra un aprendizaje distribuido;

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

la **Figura 14** ilustra una red de comunicaciones de vehículos utilizable mediante realizaciones descritas en el presente documento;

la **Figura 15** ilustra un sistema de gestión de cálculo distribuido, de acuerdo con una realización;

la **Figura 16** ilustra una relación entre la potencia computacional y la latencia de comunicación en un vehículo autónomo local como se describe en el presente documento;

la **Figura 17** es un diagrama de flujo que ilustra la lógica de descarga de cálculo de un vehículo autónomo, de acuerdo con una realización;

la **Figura 18** ilustra el uso de sensores de vehículos autónomos para ayudar en una búsqueda distribuida;

la **Figura 19** ilustra la lógica de búsqueda de vehículos autónomos distribuida, de acuerdo con una realización;

la **Figura 20** ilustra el intercambio de datos de inferencia entre vehículos autónomos, de acuerdo con una realización;

la **Figura 21** es un diagrama de flujo que ilustra la lógica de intercambio de inferencias, de acuerdo con una realización;

la **Figura 22** ilustra un vehículo autónomo configurado para minimizar los daños por accidentes de vehículos autónomos a través de sensores internos, de acuerdo con una realización;

la **Figura 23** es un diagrama de flujo que ilustra la lógica de mitigación y evitación de colisiones, de acuerdo con una realización;

la **Figura 24** ilustra un sistema autónomo postcolisión, de acuerdo con una realización;

la **Figura 25** ilustra detalles de la lógica autónoma postcolisión, de acuerdo con una realización;

la **Figura 26** es un diagrama de bloques de un sistema de procesamiento, de acuerdo con una realización;

la **Figura 27** es un diagrama de bloques de una realización de un procesador que tiene uno o más núcleos de procesador, un controlador de memoria integrado y un procesador de gráficos integrado;

la **Figura 28** es un diagrama de bloques 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 29** es un diagrama de bloques de un motor de procesamiento de gráficos de un procesador de gráficos de acuerdo con algunas realizaciones;

la **Figura 30** es un diagrama de bloques de un procesador de gráficos proporcionado por una realización adicional;

la **Figura 31** ilustra la lógica de ejecución de hilo que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones;

la **Figura 32** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos de acuerdo con algunas realizaciones;

la **Figura 33** es un diagrama de bloques de un procesador de gráficos de acuerdo con otra realización.

La **Figura 34A-34B** ilustra un formato de comando de procesador de gráficos y secuencia de comandos, de acuerdo con algunas realizaciones;

la **Figura 35** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos de acuerdo con algunas realizaciones;

la **Figura 36** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP, de acuerdo con una realización;

la **Figura 37** es un diagrama de bloques que ilustra un sistema ilustrativo en un circuito de chip integrado, de acuerdo con una realización;

la **Figura 38** es un diagrama de bloques que ilustra un procesador de gráficos adicional, de acuerdo con una realización; y

la **Figura 39** es un diagrama de bloques que ilustra un procesador de gráficos ilustrativo adicional de un sistema en un circuito de chip integrado, de acuerdo con una realización.

DESCRIPCIÓN DETALLADA

En algunas realizaciones, una unidad de procesamiento de gráficos (GPU) está acoplada de manera comunicativa a núcleos de anfitrión/de procesador para acelerar las operaciones de gráficos, las operaciones de aprendizaje automático, las operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede estar acoplada de manera comunicativa al procesador/núcleos anfitriones 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 se puede acoplar con capacidad de comunicación con los núcleos a través de un bus/interconexión interna del procesador (es decir, interna al paquete o chip). Independientemente de la manera en la que esté conectada la GPU, los núcleos del procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU utiliza entonces circuitería/lógica dedicada para procesar de forma efectiva estos comandos/instrucciones.

En la siguiente descripción, se exponen numerosos detalles específicos para proporcionar una comprensión más minuciosa. Sin embargo, será evidente para un experto en la técnica que las realizaciones descritas en el presente documento pueden ponerse en práctica sin uno o más de estos detalles específicos. En otros casos, no se han descrito características bien conocidas para evitar oscurecer los detalles de las presentes realizaciones.

Vista 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 procesador(es) 102 y una memoria de sistema 104 que se comunica mediante 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 puede integrarse dentro del/de los uno o más procesador(es) 102. El concentrador de memoria 105 se acopla con un subsistema de E/S 111 mediante un enlace de comunicaciones 106. El subsistema de E/S 111 incluye un concentrador de E/S 107 que puede permitir al sistema informático 100 que reciba entrada desde uno o más dispositivos de entrada 108. Adicionalmente, el concentrador de E/S 107 puede permitir un controlador de visualización, que puede estar incluido en el uno o más procesadores 102, para que proporcione salidas a uno o más dispositivos de visualización 110A. En una realización, el uno o más dispositivos de visualización 110A acoplados con el concentrador de E/S 107 pueden incluir un dispositivo de visualización local, interno o integrado.

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

Dentro del subsistema de E/S 111, una unidad de almacenamiento de sistema 114 puede conectarse al concentrador de E/S 107 para proporcionar un mecanismo de almacenamiento para el sistema informático 100. Puede usarse un conmutador de E/S 116 para proporcionar un mecanismo de interfaz para posibilitar conexiones entre el concentrador de E/S 107 y otros componentes, tal como un adaptador de red 118 y/o adaptador de red inalámbrica 119 que pueden estar integrados en la plataforma, y diversos otros dispositivos que pueden añadirse mediante uno o más dispositivos de adición 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, y puede conectarse también al concentrador de E/S 107. Las rutas de comunicación que interconectan los diversos componentes en la Figura 1 pueden implementarse 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(s) 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, el 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, el uno o más procesadores paralelos 112 incorporan circuitería optimizada para procesamiento de propósito general, mientras conservan la arquitectura computacional subyacente, descrita en mayor detalle en el presente documento. En otra realización más, los componentes del sistema informático 100 pueden estar integrados con uno o más otros elementos de sistema en un único circuito integrado. Por ejemplo, los uno o más procesadores paralelos 112, el concentrador de memoria 105, el/los procesador(es) 102 y el concentrador de E/S 107 pueden integrarse en un circuito integrado de sistema en chip (SoC). Como alternativa, los componentes del sistema informático 100 pueden integrarse en un único paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una parte de los componentes del sistema informático 100 se puede integrar en un módulo multichip (MCM), que se puede interconectar con otros módulos multichip en un sistema informático modular.

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

Algunos de los componentes particulares que se muestran en el presente documento son opcionales y es posible que no se incluyan en todas las implementaciones del sistema informático 100. Por ejemplo, puede soportarse cualquier número de tarjetas o periféricos de adición, o pueden eliminarse algunos componentes. Adicionalmente, algunas arquitecturas pueden usar diferente terminología para componentes similares a aquellos ilustrados en la Figura 1. Por ejemplo, el concentrador de memoria 105 puede denominarse puente norte en algunas arquitecturas, mientras que el concentrador de E/S 107 puede denominarse puente sur.

La **Figura 2A** ilustra un procesador paralelo 200, de acuerdo con una realización. Los diversos componentes del procesador paralelo 200 pueden implementarse usando uno o más dispositivos de circuito integrado, tal 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 del 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 permite la comunicación con otros dispositivos, incluidas otras instancias de la unidad de procesamiento paralelo 202. La unidad de E/S 204 puede conectarse directamente a otros dispositivos. En una realización, la unidad de E/S 204 se conecta con otros dispositivos mediante el uso de una interfaz de concentrador o 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 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, 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 comando mediante la unidad de E/S 204, la interfaz de anfitrión 206 puede dirigir operaciones de trabajo para realizar aquellos 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 mediante una lógica de firmware que se ejecuta en un microcontrolador. El planificador implementado por el microcontrolador 210 puede configurarse 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 mediante uno de múltiples llamadores de procesamiento de gráficos. Las cargas de trabajo pueden entonces ser automáticamente distribuidas 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 cálculo. 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, pueden asignarse 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 puede configurarse para realizar diversos tipos de operaciones de procesamiento paralelo. En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones informáticas paralelas de propósito general. Por ejemplo, la matriz de agrupaciones de procesamiento 212 puede incluir lógica para ejecutar tareas de procesamiento que incluye filtración de datos de video y/o de audio, realización de operaciones de modelado, que incluye operaciones físicas y realización de transformaciones de datos.

En una realización, la matriz de agrupaciones de procesamiento 212 está configurada para realizar operaciones de procesamiento de gráficos en paralelo. En las 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, que incluyen, 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 puede configurarse para ejecutar programas sombreadores 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 la memoria de sistema mediante la unidad de E/S 204 para su procesamiento. Durante el procesamiento, los datos transferidos pueden almacenarse en memoria en chip (por ejemplo, la memoria de procesador paralelo 222) durante el procesamiento y, a continuación, escribirse en diferido en memoria de sistema.

En una realización, cuando se usa la unidad de procesamiento paralelo 202 para realizar el procesamiento de gráficos, el planificador 210 puede estar configurado para dividir la carga de trabajo de procesamiento en tareas de tamaño aproximadamente igual, para permitir 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, partes de la matriz de agrupaciones de procesamiento 212 se pueden configurar para realizar diferentes tipos de procesamiento. Por ejemplo, una primera porción puede estar configurada para realizar sombreado de vértices y generación de topología, una segunda porción puede estar configurada para realizar teselación y sombreado de geometría, y una tercera porción puede estar configurada para realizar sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen renderizada para su visualización. Los datos intermedios producidos por una o

más de las agrupaciones 214A-214N pueden almacenarse en memorias intermedias para permitir que se transmitan los datos intermedios entre las agrupaciones 214A-214N para su procesamiento adicional.

Durante la operación, 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 que hay que 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 han de procesarse los datos (por ejemplo, qué programa ha de ejecutarse). El planificador 210 puede configurarse para extraer los índices que corresponden a las tareas o puede recibir los índices desde el extremo frontal 208. El extremo frontal 208 puede configurarse para garantizar que la matriz de agrupaciones de procesamiento 212 esté configurada en un estado válido antes de que se inicie la carga de trabajo especificada por memorias intermedias de comandos entrantes (por ejemplo, memorias intermedias por lotes, memorias intermedias push, etc.).

Cada una de las una o más instancias de la unidad de procesamiento paralelo 202 puede acoplarse con la memoria de procesador paralelo 222. Se puede acceder a la memoria de procesador paralelo 222 a través de la barra transversal de memoria 216, que puede recibir solicitudes de memoria desde la matriz de agrupaciones de procesamiento 212 así como la unidad de E/S 204. La barra transversal de memoria 216 puede acceder a la memoria de procesador paralelo 222 mediante una interfaz de memoria 218. La interfaz de memoria 218 puede incluir múltiples unidades de subdivisión (por ejemplo, unidad de subdivisión 220A, unidad de subdivisión 220B, hasta la unidad de subdivisión 220N) pudiendo cada una emparejarse a una parte (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 correspondiente primera unidad de memoria 224A, una segunda unidad de subdivisión 220B tiene una correspondiente unidad de memoria 224B y una unidad de subdivisión Nésima 220N tiene una correspondiente unidad de memoria Nésima 224N. 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, que incluyen memoria de acceso aleatorio dinámica (DRAM) o memoria de acceso aleatorio de gráficos, tal como la memoria de acceso aleatorio de gráficos síncrona (SGRAM), que incluye la memoria de tasa de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 224A-224N pueden incluir también memoria 3D apilada, que incluye, 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 puede seleccionarse entre uno de varios diseños convencionales. Los objetivos de renderización, tales como memorias intermedias de tramas o mapas de textura, pueden almacenarse a través de las unidades de memoria 224A-224N, permitiendo que las unidades de subdivisión 220A-220N escriban partes de cada objetivo de renderización en paralelo para usar eficientemente el ancho de banda disponible de la memoria de procesador paralelo 222. En algunas realizaciones, puede excluirse una instancia local de la memoria de procesador paralelo 222 en favor de un diseño de memoria unificado que utiliza memoria de sistema en conjunto 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 puede estar configurada para transferir la salida de cada agrupación 214A-214N en cualquier unidad de subdivisión 220A-220N o en otra agrupación 214A-214N, que puede realizar operaciones de procesamiento adicionales en la salida. Cada agrupación 214A-214N puede comunicarse con la interfaz de memoria 218 a través de la barra transversal de memoria 216 para leer desde o escribir en diversos dispositivos de memoria externos. En una realización, la barra transversal de memoria 216 tiene una conexión a la interfaz de memoria 218 para comunicarse con la unidad de E/S 204, así como una conexión a una instancia local de la memoria de procesador paralelo 222, lo que posibilita que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 214A-214N se comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 202. En una realización, la barra transversal de memoria 216 puede usar canales virtuales para separar flujos de tráfico entre las agrupaciones 214A-214N y las unidades de subdivisión 220A-220N.

Aunque se ilustra una única instancia de la unidad de procesamiento paralelo 202 dentro del procesador paralelo 200, puede incluirse 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 adición, o se pueden interconectar múltiples tarjetas de adición. Las diferentes instancias de la unidad de procesamiento paralelo 202 pueden configurarse 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 más alta con relación a otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 202 o del procesador paralelo 200 se pueden implementar en una variedad de configuraciones y factores de forma, que incluyen, entre otros, ordenadores personales de escritorio, portátiles o de mano, servidores, estaciones de trabajo, consolas de videojuegos 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 L2 221, una interfaz de memoria intermedia de trama 225 y una ROP 226 (unidad de operaciones de rasterización). La memoria caché L2 221 es una caché de lectura/escritura que está configurada para realizar operaciones de carga y almacenamiento recibidas desde la barra transversal de memoria 216 y la ROP 226. Los fallos de lectura y las solicitudes de escritura urgentes se emiten por la memoria caché L2 221 a la interfaz de memoria intermedia de trama 225 para su procesamiento. Las actualizaciones también se pueden enviar a la memoria intermedia de trama a través de la interfaz de memoria intermedia de trama 225 para su procesamiento. En una realización, la interfaz de memoria intermedia de trama 225 interactúa con una de las unidades de memoria en la memoria de procesador paralelo, tal como las unidades de memoria 224A-224N de la Figura 2 (por ejemplo, dentro de la memoria de procesador paralelo 222).

En las 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 a continuación emite los datos de gráficos procesados que se almacenan en la 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 realiza la ROP 226 puede variar de acuerdo con las características estadísticas de los datos a comprimir. Por ejemplo, en una realización, se lleva a cabo compresión de color delta en datos de profundidad y de color en una base por teselas.

En algunas realizaciones, la ROP 226 está incluida 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, las solicitudes de lectura y escritura para datos de píxeles se transmiten a través de la barra transversal de memoria 216 en lugar de los datos de fragmento de píxel. Los datos de gráficos procesados pueden visualizarse 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/los procesador(es) 102, o encaminarse para su procesamiento adicional por una de las entidades de procesamiento dentro del procesador paralelo 200 de la Figura 2A.

La **Figura 2C** es un diagrama de bloques de 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 puede estar configurada para ejecutar muchos hilos en paralelo, 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 instrucción de única instrucción de 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 de múltiples hilos (SIMT) para soportar la ejecución paralela de un gran número de hilos generalmente sincronizados, usando una unidad de instrucciones 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, 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 puede controlarse mediante un gestor de canalizaciones 232 que distribuye las tareas de procesamiento a procesadores paralelos de SIMT. El gestor de canalizaciones 232 recibe instrucciones desde el planificador 210 de la Figura 2 y gestiona la ejecución de 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, pueden incluirse diversos tipos de procesadores paralelos de SIMT de arquitecturas diferentes dentro de la agrupación de procesamiento 214. Una o más instancias del multiprocesador de gráficos 234 pueden incluirse dentro de una agrupación de procesamiento 214. El multiprocesador de gráficos 234 puede procesar datos y puede usarse una barra transversal de datos 240 para distribuir los datos procesados a uno de múltiples posibles destinos, que incluyen otras unidades sombreadoras. El gestor de canalizaciones 232 puede facilitar la distribución de datos procesados especificando destinos para que se distribuyan datos procesados mediante la barra transversal de datos 240.

Cada multiprocesador de gráficos 234 dentro de la agrupación de procesamiento 214 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades aritmético-lógicas, unidades de carga-almacenamiento, etc.). La lógica de ejecución funcional se puede configurar como una canalización, en donde se pueden emitir nuevas instrucciones antes de que se completen las instrucciones anteriores. La lógica de ejecución funcional soporta una diversidad de operaciones que incluyen aritmética de números enteros y de coma flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y cálculo de diversas funciones algebraicas. En una realización, se puede hacer uso del mismo hardware funcional-unitario para llevar a cabo 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 través del conjunto de motores de procesamiento paralelo es un grupo de hilos. Un grupo de hilos ejecuta el mismo programa en diferentes datos de entrada. Cada hilo dentro de un grupo de hilos puede asignarse a un motor de procesamiento diferente dentro de un multiprocesador 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 pueden encontrarse en reposo 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, el procesamiento se puede realizar durante ciclos de reloj consecutivos. En una realización, pueden ejecutarse múltiples grupos de hilos 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 almacenamiento. En una realización, el multiprocesador de gráficos 234 puede prescindir de una caché interna y usar una memoria caché (por ejemplo, la caché L1 308) dentro de la agrupación de procesamiento 214. Cada multiprocesador de gráficos 234 también tiene acceso a cachés L2 dentro de las unidades de subdivisión (p. ej., las unidades de subdivisión 220A-220N de la Figura 2) que se comparten entre todas las agrupaciones de procesamiento 214 y pueden usarse 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 puede usarse 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 pueden almacenarse en la caché L1 308.

Cada agrupación de procesamiento 214 puede incluir una MMU 245 (unidad de gestión de memoria) que está configurada para mapear 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 páginas (PTE) usadas para mapear una dirección virtual con una dirección física de una tesela y, opcionalmente, un índice de líneas 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 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 puede usarse 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 puede configurarse de manera que cada multiprocesador de gráficos 234 se acopla a una unidad de textura 236 para realizar operaciones de mapeo 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 L1 de textura interna (no mostrada) o, en algunas realizaciones, desde la caché L1 dentro del multiprocesador de gráficos 234 y se extraen desde una caché 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 L2, memoria de procesador paralelo local o memoria de sistema mediante la barra transversal de memoria 216. Una preROP 242 (unidad de operaciones previas a la rasterización) está configurada para recibir datos desde el multiprocesador de gráficos 234, dirigir datos a las unidades de ROP, que pueden estar ubicadas con unidades de subdivisión como se describe en el presente documento (p. ej., las unidades de subdivisión 220A-220N de la Figura 2). La unidad 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. Puede incluirse 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 puede configurarse para funcionar independientemente de otras agrupaciones de procesamiento 214 usando unidades de procesamiento separadas y distintas, cachés L1, etc.

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, entre otros, una caché de instrucciones 252, una unidad de instrucciones 254, una unidad de mapeo de direcciones 256, un archivo de registro 258, uno o más núcleos de unidad de procesamiento de gráficos de propósito general (GPGPU) 262, y una o más unidades de carga/almacenamiento 266. Los núcleos de GPGPU 262 y las unidades de carga/almacenamiento 266 están acoplados con la memoria caché 272 y a 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 canalizaciones 232. Las instrucciones se almacenan en la caché de instrucciones 252 y se envían para su ejecución por la unidad de instrucciones 254. La unidad de instrucciones 254 puede enviar instrucciones como grupos de hilos (por ejemplo, envoltentes), con cada hilo del grupo de hilos asignado a una unidad de ejecución diferente dentro del núcleo GPGPU 262. Una instrucción puede acceder a cualquiera del espacio de direcciones local, compartido o global, especificando una dirección dentro de un espacio de direcciones unificado. La unidad de mapeo de direcciones 256 se puede usar para traducir direcciones en el espacio de direcciones unificado en una dirección de memoria distinta a la que pueden acceder las unidades de carga/almacenamiento 266.

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

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

En una realización, los núcleos GPGPU 262 incluyen lógica SIMD capaz de realizar una única instrucción en múltiples conjuntos de datos. En una realización, los núcleos GPGPU 262 pueden ejecutar físicamente instrucciones SIMD4, SIMD8 y SIMD16 y ejecutar lógicamente instrucciones SIMD1, SIMD2 y SIMD32. Las instrucciones SIMD para los núcleos GPGPU pueden ser generadas en tiempo de compilación por un compilador sombreador o pueden generarse 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 pueden ejecutarse mediante una única instrucción de SIMD. Por ejemplo, y en una realización, se pueden ejecutar en paralelo ocho hilos SIMT que realizan operaciones iguales o similares a través de una única unidad lógica SIMD8.

La interconexión de memoria y 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 caché 268 es una interconexión de barra transversal que permite que la unidad de carga/almacenamiento 266 para implementar operaciones de carga y almacenamiento entre la memoria compartida 270 y el archivo de registro 258. El archivo de registro 258 puede operar a la misma frecuencia que los núcleos GPGPU 262, por lo tanto, la transferencia de datos entre los núcleos GPGPU 262 y el archivo de registro 258 tiene una latencia muy baja. La memoria compartida 270 se puede usar para permitir la comunicación entre hilos que se ejecutan en las unidades funcionales dentro del multiprocesador de gráficos 234. La memoria caché 272 puede usarse 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 también se puede utilizar como un programa gestionado en caché. Los hilos que se ejecutan en los núcleos GPGPU 262 pueden almacenar datos programáticamente dentro de la memoria compartida además de los datos almacenados en caché automáticamente que se almacenan dentro de la memoria caché 272.

Las **Figuras 3A-3B** ilustran multiprocesadores de gráficos adicionales, de acuerdo con las 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 pueden estar configurados como un multiprocesador de envío 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 instrucciones 332A-332B, del archivo de registro 334A-334B y de la o las unidades de textura 344A-344B. El multiprocesador de gráficos 325 también incluye múltiples conjuntos de gráficos o unidades de ejecución de cálculo (por ejemplo, núcleo GPGPU 336A-336B, núcleo GPGPU 337A-337B, núcleo 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, memoria caché de textura y/o de datos 342 y una memoria compartida 346.

Los diversos componentes pueden comunicarse mediante una estructura de interconexión 327. En una realización, la estructura de interconexión 327 incluye uno o más conmutadores de barra transversal para permitir la comunicación entre los diversos componentes del multiprocesador de gráficos 325. En una realización, la estructura de interconexión 327 es una capa de estructura de red de alta velocidad separada en la que se apila cada componente del multiprocesador de gráficos 325. Los componentes del multiprocesador de gráficos 325 se comunican con componentes remotos mediante la estructura de interconexión 327. Por ejemplo, cada uno de los núcleos GPGPU 336A-336B, 337A-337B y 3378A-338B pueden comunicarse con la memoria compartida 346 a través de la estructura de interconexión 327. La estructura 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, donde cada conjunto de recursos de ejecución incluye múltiples unidades de instrucción, archivos de registro, núcleos GPGPU y unidades de almacenamiento de carga, como se ilustra en la Figura 2D y la Figura 3A. Los recursos de ejecución 356A-356D pueden funcionar en conjunto con la(s) unidad(es) 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 por medio de una estructura de interconexión 352 similar a la estructura 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 pueden implementarse en cualquier unidad de procesamiento configurada adecuadamente, incluyendo, entre otros, uno o más procesadores de aplicaciones móviles, una o más unidades centrales de procesamiento (CPU) de escritorio o de servidor, incluidas 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 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 puede estar acoplada comunicativamente 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 se puede acoplar con capacidad de comunicación con los núcleos a través de un bus/interconexión interna del procesador (es decir, interna al paquete o chip). Independientemente de la manera en la que esté conectada la GPU, los núcleos del procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU utiliza entonces circuitería/lógica dedicada para procesar de forma efectiva estos comandos/instrucciones.

Técnicas para la interconexión de GPU a procesador 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 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 mayor, dependiendo de la implementación. Se pueden utilizar varios protocolos de interconexión, incluidos, entre otros, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no 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 enlaces de alta velocidad 444-445, que pueden implementarse usando protocolos/enlaces iguales o diferentes a los utilizados para los enlaces de alta velocidad 440-443. De manera similar, dos o más de los procesadores de múltiples núcleos 405-406 pueden conectarse a través del enlace de alta velocidad 433 que pueden ser buses de multiprocesador simétrico (SMP) que funcionan a 20 GB/s, 30 GB/s, 120 GB/s o más. Como alternativa, toda la comunicación entre los diversos componentes de sistema mostrados en la Figura 4A puede conseguirse usando los mismos protocolos/enlaces (p. ej., a través de una estructura de interconexión común). Sin embargo, como se ha mencionado, 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, mediante 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, tal 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, una 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 GPU 410-413 pueden estar acoplados físicamente a una memoria particular 401-402, 420-423, respectivamente, se puede implementar una arquitectura de memoria unificada en la que el mismo espacio de direcciones del sistema virtual (también conocido como espacio de "direcciones eficaces") se distribuye entre todas las diversas memorias físicas. Por ejemplo, cada una de las memorias de procesador 401-402 puede comprender 64 GB el espacio de direcciones de la memoria del sistema y cada una de las memorias de GPU 420-423 puede comprender 32 GB el espacio de direcciones de la memoria del sistema (dando como resultado un total de 256 GB de memoria direccionable 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 mediante el enlace de alta velocidad 440. Como alternativa, el módulo de aceleración de gráficos 446 puede estar integrado 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 varios otros componentes para ejecutar instrucciones y procesar datos que no se ilustran para evitar complicar los principios subyacentes de la invención (por ejemplo, unidades de búsqueda de instrucciones, unidades de predicción de ramificación, decodificadores, unidades de ejecución, memorias intermedias de reordenamiento, etc.). Las cachés 462A-462D pueden comprender cachés de nivel 1 (L1) y de nivel 2 (L2). Además, una o más cachés compartidas 426 pueden incluirse en la jerarquía de almacenamiento en caché y ser compartidas por conjuntos de núcleos 460A-460D. Por ejemplo, una realización del procesador 407 incluye 24 núcleos, cada uno con su propia caché L1, doce cachés L2 compartidas y doce cachés L3 compartidas. En esta realización, una de las cachés L2 y 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

La coherencia se mantiene para los datos e instrucciones almacenados en las diversas cachés 462A-462D, 456 y la memoria de sistema 441 a través de comunicación entre núcleos a través de un bus de coherencia 464. Por ejemplo, cada caché puede tener una lógica/circuitería de coherencia de caché asociada con la misma 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 de caché a través del bus de coherencia 464 para monitorizar los accesos de caché. Las técnicas de coherencia/monitorización de caché son bien entendidas por los expertos en la técnica y no se describirán aquí en detalle para evitar complicar los principios subyacentes de la invención.

En una realización, un circuito proxy 425 acopla con capacidad de comunicación el módulo de aceleración de gráficos 446 al bus de coherencia 464, permitiendo al módulo de aceleración de gráficos 446 participar en el protocolo de coherencia de caché como un par de los núcleos. En particular, una interfaz 435 proporciona conectividad al circuito proxy 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 beneficio 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 motor de procesamiento de gráficos dentro de una GPU, tal como las unidades de ejecución de gráficos, los motores de procesamiento de medios (por ejemplo, codificadores/decodificadores de vídeo), muestreadores y motores blit. En otras palabras, el módulo de aceleración de gráficos puede ser una GPU con una pluralidad de motores de procesamiento de gráficos 431-432, N o los motores de procesamiento de gráficos 431-432, N pueden ser 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 efectiva 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 un 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 441. Como se ha mencionado, esto puede conseguirse mediante el circuito proxy 425 que toma parte en el mecanismo de coherencia

de caché en beneficio 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 hilos. Por ejemplo, el circuito de gestión de contexto 448 puede realizar operaciones de guardado y restauración para guardar y restaurar contextos de los diversos hilos durante cambios de contexto (por ejemplo, cuando se guarda un primer hilo y se almacena un segundo hilo de modo que el segundo hilo puede ejecutarse por un motor de procesamiento de gráficos). Por ejemplo, en un cambio de contexto, el circuito de gestión de contexto 448 puede almacenar valores de registro actuales en una región designada en memoria (por ejemplo, identificada por un puntero de contexto). Luego puede restaurar los valores del registro al regresar al contexto. En una realización, un circuito de gestión de interrupciones 447 recibe y procesa interrupciones recibidas desde los dispositivos de sistema.

En una implementación, las direcciones virtuales/eficaces de un motor de procesamiento de gráficos 431 se traducen a direcciones reales/físicas en la memoria de sistema 411 por la MMU 439. 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 de acelerador de gráficos 446 puede estar especializado en una única aplicación ejecutada en el procesador 407 o puede compartirse entre múltiples aplicaciones. En una realización, se presenta un entorno de ejecución de gráficos virtualizado en el que los recursos de los motores de procesamiento de gráficos 431-432, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos pueden subdividirse en "segmentos" que se asignan a diferentes VM y/o aplicaciones en función de los requisitos de procesamiento y las prioridades asociadas con las VM y/o aplicaciones.

Por lo tanto, el circuito de integración de acelerador actúa como un puente al sistema para el módulo de aceleración 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 asignan explícitamente al espacio de direcciones real visto por el procesador de anfitrión 407, cualquier procesador de anfitrión puede direccionar 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 al sistema como unidades independientes.

Como se ha mencionado, 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 están siendo procesados 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 (incluidas DRAM apiladas), memoria GDDR (por ejemplo, GDDR5, GDDR6) o HBM, y/o pueden ser memorias no volátiles 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 sean datos que serán utilizados con la mayor frecuencia por los motores de procesamiento de gráficos 431-432, N y preferentemente no utilizados por los núcleos 460A-460D (al menos no con frecuencia). De manera similar, el mecanismo de desvío intenta mantener datos que necesitan los núcleos (y, preferentemente, no 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 mediante la interfaz 437 y la interfaz 435 (que, de nuevo, puede utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración de acelerador 436 puede realizar las mismas operaciones que aquellas descritas con respecto a la **Figura 4B**, pero potencialmente a un caudal superior dada 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 dedicado (sin virtualización de módulo de aceleración de gráficos) y modelos de programación compartida (con virtualización). El último puede incluir modelos de programación que se controlan por el circuito de integración de 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 en 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, lo que proporciona virtualización dentro de una VM/subdivisión.

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

Para el modelo de programación compartida, el módulo de aceleración de gráficos 446 o un motor de procesamiento de gráficos 431-432, N individual selecciona un elemento de proceso usando un manejador de proceso. En una realización, los elementos de proceso se almacenan en la memoria de sistema 411 y se pueden direccionar utilizando 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 al software de sistema para añadir el elemento de proceso a la lista vinculada de elementos de proceso). Los 16 bits inferiores del manejador de proceso pueden ser el desplazamiento del elemento de proceso dentro de la lista vinculada de elementos de proceso.

La **Figura 4D** ilustra un segmento de integración de acelerador 490 ilustrativo. Como se usa en el presente documento, un "segmento" 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 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 correspondiente aplicación 480. 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 el último caso, el WD 484 es un puntero a la cola de solicitud de trabajo en el espacio de direcciones 482 de la aplicación.

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

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

En la operación, una unidad de extracción de WD 491 en el segmento de integración de acelerador 490 extrae el siguiente WD 484 que incluye una indicación del trabajo que va a hacerse 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 ser utilizados por la MMU 439, el circuito de gestión de interrupciones 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 páginas/segmentos para acceder a las tablas de segmentos/páginas 486 dentro del espacio de direcciones virtuales del SO 485. El circuito de gestión de interrupciones 447 puede procesar eventos de interrupción 492 recibidos del módulo de aceleración de gráficos 446. Cuando se realizan operaciones de gráficos, una dirección eficaz 493 generada por un motor de procesamiento de gráficos 431-432, N es traducida a una dirección real por la MMU 439.

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

Tabla 1 - Registros inicializados por hipervisor

1	Registro de control de segmentos
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

1	Registro de control de segmentos
6	Registro de estado
7	ID de subdivisión lógica
8	Puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

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

5 **Tabla 2 - Registros inicializados por el sistema operativo**

1	Identificación de procesos e hilos
2	Puntero de guardado/restauración de contexto de dirección eficaz (EA)
3	Puntero de registro de utilización de acelerador de dirección virtual (VA)
4	Puntero de tabla de segmentos de almacenamiento de dirección virtual (VA)
5	Máscara de autoridad
6	Descriptor de trabajo

En una realización, cada WD 484 es específico de un módulo de aceleración de gráficos 446 y/o de un motor de procesamiento de gráficos 431-432, N particular. Este 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 en la que la aplicación ha establecido una cola de comandos de trabajo que hay que completar.

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

Los modelos de programación compartidos permiten que todos o un subconjunto de procesos de todas o un subconjunto de subdivisiones del sistema utilicen un módulo de aceleración de gráficos 446. Hay dos modelos de programación donde el módulo de aceleración de gráficos 446 se comparte por múltiples procesos y subdivisiones: compartido en segmentos de tiempo y compartido dirigido a gráficos.

En este modelo, el hipervisor de sistema 496 posee el módulo de aceleración de gráficos 446 y pone su función a disposición de todos los sistemas operativos 495. Para que un módulo de aceleración de gráficos 446 soporte virtualización por el hipervisor de sistema 496, el módulo de aceleración de gráficos 446 puede adherirse a los siguientes requisitos: 1) La solicitud de trabajo de una aplicación debe ser autónoma (es decir, no es necesario mantener el estado entre trabajos), o el módulo de aceleración de gráficos 446 debe proporcionar un mecanismo de guardado y restauración de contexto. 2) El módulo de aceleración de gráficos 446 garantiza que la solicitud de trabajo de una aplicación se complete en una cantidad de tiempo especificada, incluido cualquier fallo de conversión, o el módulo de aceleración de gráficos 446 proporciona la capacidad de anular el procesamiento del trabajo. 3) Se ha de garantizar al módulo de aceleración de gráficos 446 la equidad entre procesos cuando se opera en el modelo de programación compartido dirigido.

En una realización, para el modelo compartido, se requiere que la aplicación 480 haga una llamada de sistema al sistema operativo 495 con un tipo 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 guardado/restauración de contexto (CSRP). El tipo del módulo de aceleración de gráficos 446 describe la función de aceleración dirigida como objetivo para la llamada de sistema. El tipo del módulo de aceleración de gráficos 446 puede ser un valor específico del sistema. Al WD se le da formato específicamente para el módulo de aceleración de gráficos 446, y puede estar en forma de un comando del módulo de aceleración de gráficos 446, un puntero de dirección eficaz a una estructura definida por el usuario, un puntero de dirección eficaz a una cola de comandos, o cualquier otra estructura de datos para describir el trabajo a hacer por el módulo de aceleración de gráficos 446. En una realización, el valor de AMR es el estado de AMR que usar para el proceso actual. El valor pasado al sistema operativo es similar a que una aplicación establezca el AMR. Si las implementaciones del circuito de integración de acelerador 436 y del módulo de aceleración de gráficos 446 no soportan un Registro de anulación de máscara de autoridad de usuario (UAMOR), el sistema operativo puede aplicar el valor de UAMOR actual al valor de AMR antes de pasar el AMR en la llamada de hipervisor. Opcionalmente, el hipervisor 496 puede aplicar 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 482 de la aplicación para que el módulo de aceleración de gráficos 446 guarde y restaure el estado de contexto. Este puntero es opcional si no es necesario guardar un estado entre trabajos o cuando se adelanta un trabajo. El área de guardado/restauración de contexto puede estar fijada en la memoria de sistema.

Después de recibir la llamada de sistema, el sistema operativo 495 puede verificar que se ha registrado la aplicación 480 y que se le ha dado la autoridad para usar el módulo de aceleración de gráficos 446. El sistema operativo 495 llama, a continuación, 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/restauración de contexto (CSRP) de dirección eficaz (EA)
4	Un ID de proceso (PID) y un 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)

Al recibir la llamada de hipervisor, el hipervisor 496 verifica que el sistema operativo 495 se ha registrado y se le ha otorgado la autoridad para usar el módulo de aceleración de gráficos 446. El hipervisor 496, a continuación, pone el elemento de proceso 483 en la lista de elementos de proceso vinculados para el correspondiente tipo de módulo de aceleración de gráficos 446. 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/restauración de contexto (CSRP) de dirección eficaz (EA)
4	Un ID de proceso (PID) y un 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 del segmento de integración de acelerador 490.

Como se ilustra en la **Figura 4F**, una realización de la invención emplea una memoria unificada direccionable mediante 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, las operaciones ejecutadas en las GPU 410-413 utilizan el mismo espacio de direcciones de memoria virtual/eficaz para acceder a las memorias de procesador 401-402 y viceversa, simplificando de esta manera la programabilidad. En una realización, una primera porción del espacio de direcciones virtuales/eficaces 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 (en ocasiones denominado el espacio de direcciones eficaces) está distribuido, de esta manera, a través de cada una de las memorias de procesador 401-402 y de las memorias de GPU 420-423, permitiendo que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual mapeada a 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 desvío que indican las memorias físicas en las que deben almacenarse ciertos tipos de datos. Aunque se ilustran múltiples casos de 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 de acelerador 436.

Una realización permite que la memoria adjunta a la GPU 420-423 se asigne como parte de memoria de sistema, y que se acceda a ella usando tecnología de memoria virtual compartida (SVM), pero sin sufrir las desventajas de rendimiento típicas asociadas con la coherencia de caché de sistema completa. La capacidad de que se acceda a la memoria adjunta a la GPU 420-423 como memoria de sistema sin sobrecarga de coherencia de caché onerosa proporciona un entorno de operación beneficioso para la descarga de la GPU. Esta disposición permite que el software del procesador de anfitrión 405 establezca operandos y acceda a resultados de cálculo, sin la sobrecarga de copias de datos de DMA de E/S tradicionales. Tales copias tradicionales implican llamadas de controlador, interrupciones y accesos de E/S asignados con memoria (MMIO) que son, todos ellos, ineficientes en relación con los accesos de memoria sencillos. Al mismo tiempo, la capacidad de acceder a la memoria adjunta a la GPU 420-423 sin sobrecargas de coherencia de caché puede ser fundamental para el tiempo de ejecución de un cálculo descargado. En casos con tráfico de memoria de escritura de transmisión por flujo continuo sustancial, por ejemplo, la sobrecarga 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 cálculo 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 desvío, por ejemplo, que puede ser una estructura granular de página (es decir, controlada en la granularidad de una página de memoria) que incluye 1 o 2 bits por página de memoria adjunta a la GPU. La tabla de desvío se puede implementar en un rango de memoria robada de una o más memorias adjuntas a la GPU 420-423, con o sin una caché de desvío en la GPU 410-413 (por ejemplo, para almacenar en caché las entradas utilizadas con frecuencia/recientemente de la tabla de desvío). Como alternativa, toda la tabla de desvío puede mantenerse dentro de la GPU.

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

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

Un mecanismo para cambiar el estado de desvío emplea una llamada API (por ejemplo, OpenCL), que, a su vez, llama al controlador del dispositivo de la GPU que, a su vez, envía un mensaje (o pone en cola un descriptor de comandos) a la GPU que la dirige para cambiar el estado de desvío y, para algunas transiciones, realiza una operación de vaciado de caché en el anfitrión. Se requiere la operación de vaciado de caché para una transición desde el procesador de anfitrión 405 a un desvío de GPU, pero no se requiere para la transición 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 otorgar o no acceso inmediato, dependiendo de la implementación. Por tanto, para reducir la comunicación entre el procesador 405 y la GPU 410 es beneficioso garantizar que las páginas con desvío de GPU sean aquellas que requiere la GPU pero no el procesador de anfitrión 405 y viceversa.

Canalización de procesamiento de gráficos

La **Figura 5** ilustra una canalización de procesamiento de gráficos 500, de acuerdo con una realización. En una realización, un procesador de gráficos puede implementar la canalización de procesamiento de gráficos 500 ilustrada. El procesador de gráficos puede incluirse dentro de los subsistemas de procesamiento paralelo como se describe en el presente documento, tal como el procesador paralelo 200 de la Figura 2, que, en una realización, es una variante del procesador o procesadores paralelos 112 de la Figura 1. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento de gráficos 500 mediante una o más instancias de la unidad de procesamiento paralelo (p. ej., la unidad de procesamiento paralelo 202 de la Figura 2) como se describe en el presente

documento. Por ejemplo, una unidad sombreadora (p. ej., 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 vértices 504, una unidad de procesamiento de control de teselación 508, una unidad de procesamiento de evaluación de teselación 512, una unidad de procesamiento de geometría 516 y una unidad de procesamiento de fragmentos/píxeles 524. Las funciones del ensamblador de datos 502, los ensambladores de primitivas 506, 514, 518, la unidad de teselación 510, el rasterizador 522, y la unidad de operaciones del rasterizador 526 pueden realizarse también por otros motores de procesamiento dentro de una agrupación de procesamiento (p. ej., la agrupación de procesamiento 214 de la Figura 3) y una correspondiente unidad de subdivisión (p. ej., la unidad de subdivisión 220A-220N de la Figura 2). La canalización de procesamiento de gráficos 500 puede implementarse también 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, CPU). En una realización, una o más porciones de la canalización de procesamiento de gráficos 500 pueden acceder a una memoria en chip (p. ej., la memoria de procesador paralelo 222 como en la Figura 2) mediante 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 vértices 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 sombreadores de vértice, iluminando y transformando datos de vértice de acuerdo con lo especificado por los programas sombreadores de vértice. 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 vértices y puede programarse para transformar los datos de vértices desde una representación de coordenadas basada en objetos hasta un espacio de coordenadas de espacio mundial o un espacio de coordenadas de dispositivo normalizado.

Una primera instancia de un ensamblador de primitivas 506 recibe atributos de vértices desde la unidad de procesamiento de vértices 504. El ensamblador de primitivas 506 lee atributos de vértices almacenados según sea necesario y construye primitivas de gráficos para su procesamiento por la unidad de procesamiento de control de teselación 508. Los primitivas de gráficos incluyen triángulos, segmentos de línea, puntos, parches, etc., de acuerdo con lo soportado por varias interfaces de programación de aplicaciones (API) de procesamiento de gráficos.

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

Una segunda instancia de un ensamblador de primitivas 514 recibe atributos de vértices desde la unidad de procesamiento de evaluación de teselación 512, que lee los atributos de vértices 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 sombreadores de geometría para transformar primitivas de gráficos recibidas desde el ensamblador de primitivas 514 como se especifica por los programas sombreadores de geometría. En una realización, la unidad de procesamiento de geometría 516 está programada para subdividir las primitivas de gráficos en una o más primitivas de gráficos nuevas y calcular parámetros usados para rasterizar las nuevas primitivas de gráficos.

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

El rasterizador 522 puede realizar optimizaciones de selección de profundidad y otras basadas en profundidad. El rasterizador 522 también realiza conversión de escaneo en las nuevas primitivas de gráficos para generar fragmentos y enviar 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 sombreadores de fragmentos o programas sombreadores 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 sombreadores de fragmentos o de píxeles. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 524 puede programarse para realizar operaciones que incluyen, pero sin limitación, mapeo 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 están almacenados en la memoria de procesador paralelo o en la memoria de sistema para su uso cuando se procesan los datos de fragmentos. Los programas sombreadores de fragmentos o de píxeles pueden estar configurados para sombrear a granularidad de muestra, de píxel, de tesela u otras dependiendo de la tasa de muestreo configurada para las unidades de procesamiento.

La unidad de operaciones de rasterización 526 es una unidad de procesamiento que realiza operaciones de rasterización que incluyen, pero sin limitación, estarcido, prueba z, mezcla y similares, y emite datos de píxeles como datos de gráficos procesados para que se almacenen en la memoria de gráficos (p. ej., la memoria de procesador paralelo 222 como en la Figura 2, y/o la memoria de sistema 104 como en la Figura 1, para que se visualicen en el uno o más dispositivos de visualización 110 o para su procesamiento adicional por uno del uno o más procesadores 102 o procesadores paralelos 112. En algunas 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 la memoria.

Vista general de aprendizaje automático

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

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

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

La precisión de un algoritmo de aprendizaje automático puede verse afectada significativamente por la calidad del conjunto de datos usado para entrenar el algoritmo. El proceso de entrenamiento puede ser computacionalmente intensivo y puede requerir una cantidad de tiempo significativa en un procesador de propósito general convencional. Por consiguiente, se usa hardware de procesamiento paralelo para entrenar muchos tipos de algoritmos de aprendizaje automático. Esto es particularmente útil para optimizar el entrenamiento de redes neuronales, ya que los cálculos realizados al ajustar los coeficientes en las redes neuronales se prestan de manera natural a implementaciones paralelas. Específicamente, muchos algoritmos de aprendizaje automático y aplicaciones de software se han adaptado a hacer uso del hardware de procesamiento paralelo dentro de dispositivos de procesamiento de gráficos de propósito general.

La **Figura 6** es un diagrama generalizado de una pila de software de aprendizaje automático 600. Una aplicación de aprendizaje automático 602 puede estar configurada para entrenar una red neuronal usando un conjunto de datos de

entrenamiento o para usar una red neuronal profunda entrenada para implementar la inteligencia automática. La aplicación de aprendizaje automático 602 puede incluir funcionalidad de entrenamiento e inferencia para una red neuronal y/o software especializado que puede usarse para entrenar una red neuronal antes del despliegue. La aplicación de aprendizaje automático 602 puede implementar cualquier tipo de inteligencia automática incluyendo, pero sin limitación, reconocimiento de imágenes, mapeo y localización, navegación autónoma, síntesis de habla, formación de imágenes médicas o traducción de idioma.

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

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

Aceleración de aprendizaje automático de GPGPU

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

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

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

En una realización, cada agrupación de cálculo 706A-706H incluye un conjunto de multiprocesadores de gráficos, tal como el multiprocesador de gráficos 400 de la Figura 4A. Los multiprocesadores de gráficos de la agrupación de cálculo tienen múltiples tipos de unidades de lógica de enteros y de coma flotante que pueden realizar operaciones computacionales con un intervalo de precisiones que incluyen unas adecuadas para cálculos de aprendizaje automático. Por ejemplo, y en una realización, al menos un subconjunto de las unidades de coma flotante en cada una de las agrupaciones de cálculo 706A-H puede estar configurado para realizar operaciones de coma flotante de 16 bits o de 32 bits, mientras que un subconjunto diferente de las unidades de coma flotante puede estar configurado para realizar operaciones de coma flotante de 64 bits.

Pueden configurarse múltiples instancias de la GPGPU 700 para operar como una agrupación de cálculo. El mecanismo de comunicación usado por la agrupación de cálculo para la sincronización y el intercambio de datos varía a través de las realizaciones. En una realización, las múltiples instancias de la GPGPU 700 se comunican a través de

la interfaz de anfitrión 702. En una realización, la GPGPU 700 incluye un concentrador de E/S 708 que acopla la GPGPU 700 con un enlace de GPU 710 que posibilita una conexión directa a otras instancias de la GPGPU. En una realización, el enlace de GPU 710 está acoplado a un puente de GPU a GPU dedicado que posibilita la comunicación y sincronización entre múltiples instancias de la GPGPU 700. En una realización, el enlace de GPU 710 se acopla con una interconexión de alta velocidad para transmitir y recibir datos a otras GPGPU o procesadores paralelos. En una realización, las múltiples instancias de la GPGPU 700 están ubicadas en sistemas de procesamiento de datos separados y se comunican mediante un dispositivo de red que es accesible mediante la interfaz de anfitrión 702. En una realización, el enlace de GPU 710 puede estar configurado para posibilitar una conexión a un procesador de anfitrión además de o como una alternativa a la interfaz de anfitrión 702.

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

La **Figura 8** ilustra un sistema informático de múltiples GPU 800, de acuerdo con una realización. El sistema informático de múltiples GPU 800 puede incluir un procesador 802 acoplado a múltiples GPGPU 806A-D mediante un conmutador de interfaz de anfitrión 804. El conmutador de interfaz de anfitrión 804, en una realización, es un dispositivo de conmutador de PCI express que acopla el procesador 802 a un bus de PCI express a través del que el procesador 802 puede comunicarse con el conjunto de GPGPU 806A-D. Cada una de las múltiples GPGPU 806A-806D puede ser una instancia de la GPGPU 700 de la Figura 7. Las GPGPU 806A-D pueden interconectarse mediante un conjunto de enlaces de GPU a GPU de punto a punto de alta velocidad 816. Los enlaces de GPU a GPU de alta velocidad pueden conectarse a cada una de las GPGPU 806A-806D mediante un enlace de GPU dedicado, tal como el enlace de GPU 710 como en la Figura 7. Los enlaces de GPU de P2P 816 posibilitan la comunicación directa entre cada una de las GPGPU 806A-D sin requerir la comunicación a través del bus de interfaz de anfitrión a la que está conectado el procesador 802. Con el tráfico de GPU a GPU dirigido a los enlaces de GPU de P2P, el bus de interfaz de anfitrión permanece disponible para el acceso a memoria de sistema o para comunicarse con otras instancias del sistema informático de múltiples GPU 800, por ejemplo, mediante uno o más dispositivos de red. Aunque en la realización ilustrada las GPGPU 806A-D se conectan al procesador 802 mediante el conmutador de interfaz de anfitrión 804, en una realización, el procesador 802 incluye el soporte directo para los enlaces de GPU de P2P 816 y puede conectarse directamente a las GPGPU 806A-806D.

Implementaciones de red neuronal de aprendizaje automático

La arquitectura informática proporcionada por las realizaciones descritas en el presente documento puede estar configurada para realizar los tipos de procesamiento paralelo que son particularmente adecuados para entrenar y desplegar redes neuronales para aprendizaje automático. Una red neuronal puede generalizarse como una red de funciones que tienen una relación de gráfico. Como es bien conocido en la técnica, existe una variedad de tipos de implementaciones de redes neuronales utilizadas en el aprendizaje automático. Un tipo ilustrativo de red neuronal es la red de realimentación prospectiva, como se ha descrito anteriormente.

Un segundo tipo ilustrativo de red neuronal es la red neuronal convolucional (CNN). Una CNN es una red neuronal de realimentación prospectiva especializada para procesar datos que tienen una topología similar a cuadrícula conocida, tal como datos de imagen. En consecuencia, las CNN se usan comúnmente para aplicaciones de reconocimiento de imágenes y de visión de cálculo, pero pueden usarse también para otros tipos de reconocimiento de patrones, tales como procesamiento de habla y de idioma. Los nodos en la capa de entrada de CNN están organizados en un conjunto de "filtros" (detectores de características inspirados por los campos receptivos encontrados en la retina), y la salida de cada conjunto de filtros se propaga a nodos en capas sucesivas de la red. Los cálculos para una CNN incluyen aplicar la operación matemática de convolución a cada filtro para producir la salida de ese filtro. La convolución es una clase especializada de operación matemática realizada por dos funciones para producir una tercera función que es una versión modificada de una de las dos funciones originales. En terminología de redes convolucionales, la primera función de la convolución puede denominarse entrada, mientras que la segunda función puede denominarse núcleo de convolución. La salida puede denominarse mapa de características. Por ejemplo, la entrada a una capa de convolución puede ser una matriz multidimensional de datos que define los diversos componentes de color de una imagen de entrada. El núcleo de convolución puede ser una matriz multidimensional de parámetros, donde los parámetros están adaptados por el proceso de entrenamiento para la red neuronal.

Las redes neuronales recurrentes (RNN) son una familia de las redes neuronales de realimentación prospectiva que incluyen conexiones de realimentación entre capas. Las RNN posibilitan el modelado de datos secuenciales compartiendo datos de parámetros a través de diferentes partes de la red neuronal. La arquitectura para una RNN incluye ciclos. Los ciclos representan la influencia de un valor presente de una variable sobre su propio valor en el

futuro, ya que al menos una porción de los datos de salida de la RNN se utiliza como realimentación para procesar entradas posteriores en una secuencia. Esta característica hace a las RNN particularmente útiles para procesamiento de idioma debido a la naturaleza variable en la que pueden estar compuestos los datos de idioma.

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

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

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

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

Las Figuras 9A-B ilustran una red neuronal convolucional ilustrativa. La Figura 9A ilustra diversas capas dentro de una CNN. Como se muestra en la Figura 9A, una CNN ilustrativa usada para modelar el procesamiento de imagen puede recibir la entrada 902 que describe las componentes de rojo, verde y azul (RGB) de una imagen de entrada. La entrada 902 puede ser procesada por múltiples capas convolucionales (por ejemplo, la capa convolucional 904, la capa convolucional 906). La salida de las múltiples capas convolucionales puede procesarse opcionalmente por un conjunto de capas completamente conectadas 908. Las neuronas en una capa completamente conectada tienen conexiones completas a todas las activaciones en la capa anterior, como se ha descrito anteriormente para una red de realimentación prospectiva. La salida de las capas completamente conectadas 908 puede usarse para generar un resultado de salida a partir de la red. Las activaciones dentro de las capas completamente conectadas 908 pueden calcularse usando una multiplicación matricial en lugar de la convolución. No todas las implementaciones de CNN hacen uso de las capas completamente conectadas 1108. Por ejemplo, en algunas implementaciones, la capa convolucional 906 puede generar la salida de la CNN.

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

La Figura 9B ilustra fases de cálculo ilustrativas dentro de una capa convolucional de una CNN. La entrada a una capa convolucional 912 de una CNN puede procesarse en tres fases de una capa convolucional 914. Las tres etapas pueden incluir una etapa de convolución 916, una etapa de detector 918 y una etapa de agrupamiento 920. La capa convolucional 914 puede emitir a continuación datos a una capa convolucional sucesiva. La capa convolucional final

de la red puede generar datos de mapeo de características de salida o proporcionar entrada a una capa completamente conectada, por ejemplo, para generar un valor de clasificación para la entrada a la CNN.

En la etapa de convolución 916 se realizan varias convoluciones en paralelo para producir un conjunto de activaciones lineales. La etapa de convolución 916 puede incluir una transformación afín, que es cualquier transformación que puede especificarse como una transformación lineal más una traducción. Las transformaciones afines incluyen rotaciones, traslaciones, escalamiento y combinaciones de estas transformaciones. La etapa de convolución calcula la salida de funciones (por ejemplo, neuronas) que están conectadas a regiones específicas en la entrada, que puede determinarse como la región local asociada con la neurona. Las neuronas calculan un producto escalar entre los pesos de las neuronas y la región en la entrada local a la que están conectadas las neuronas. La salida de la etapa de convolución 916 define un conjunto de activaciones lineales que se procesan por etapas sucesivas de la capa convolucional 914.

Las activaciones lineales pueden procesarse por una etapa de detector 918. En la etapa de detector 918, cada activación lineal se procesa por una función de activación no lineal. La función de activación no lineal aumenta las propiedades no lineales de la red global sin afectar a los campos receptivos de la capa de convolución. Pueden usarse varios tipos de funciones de activación no lineal. Un tipo particular es la unidad lineal rectificadora (ReLU), que usa una función de activación definida como $f(x) = \max(0, x)$, de manera que la activación tiene un umbral de cero.

La etapa de agrupamiento 920 usa una función de agrupación que sustituye la salida de la capa convolucional 906 con un sumario estadístico de las salidas cercanas. La función de agrupamiento puede usarse para introducir la invarianza de traslación en la red neuronal, de manera que traslaciones pequeñas en la entrada no cambian las salidas agrupadas. La invarianza a la traslación local puede ser útil en situaciones donde la presencia de una característica en los datos de entrada es más importante que la ubicación precisa de la característica. Pueden usarse diversos tipos de funciones de agrupamiento durante la etapa de agrupamiento 920, que incluye agrupamiento máximo, agrupamiento promedio y agrupamiento de norma l2. Adicionalmente, algunas implementaciones de CNN no incluyen una fase de agrupamiento. En su lugar, tales implementaciones sustituyen una etapa de convolución adicional que tiene un paso mayor en relación con las etapas de convolución anteriores.

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

La **Figura 10** ilustra una red neuronal recurrente 1000 ilustrativa. En una red neuronal recurrente (RNN), el estado anterior de la red influye en la salida del estado actual de la red. Las RNN pueden crearse en una diversidad de maneras usando una diversidad de funciones. El uso de RNN generalmente gira en torno al uso de modelos matemáticos para predecir el futuro en función de una secuencia previa de entradas. Por ejemplo, puede usarse una RNN para realizar modelado de idioma estadístico para predecir una palabra próxima dada en una secuencia de palabras anterior. La RNN 1000 ilustrada puede describirse como que tiene una capa de entrada 1002 que recibe un vector de entrada, capas ocultas 1004 para implementar una función recurrente, un mecanismo de realimentación 1005 para posibilitar una 'memoria' de estados anteriores y una capa de salida 1006 para emitir un resultado. La RNN 1000 opera basándose en pasos de tiempo. El estado de la RNN en un paso de tiempo dado se ve influenciado basándose en el paso de tiempo anterior mediante el mecanismo de realimentación 1005. Para un paso de tiempo dado, se define el estado de las capas ocultas 1004 por el estado anterior y la entrada en el paso de tiempo actual. Puede procesarse una entrada inicial (x_1) en un primer paso de tiempo por la capa oculta 1004. Puede procesarse una segunda entrada (x_2) por la capa oculta 1004 usando información de estado que se determina durante el procesamiento de la entrada inicial (x_1). Un estado dado puede calcularse como $s_t = f(Ux_t + Ws_{t-1})$, donde U y W son matrices de parámetros. La función f es, en general, una no linealidad, tal como la función tangente hiperbólica (Tanh) o una variante de la función rectificadora $f(x) = \max(0, x)$. Sin embargo, la función matemática específica usada en las capas ocultas 1004 puede variar dependiendo de los detalles de la implementación específica de la RNN 1000.

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

La **Figura 11** ilustra el entrenamiento y despliegue de una red neuronal profunda. Una vez que se ha estructurado una red dada para una tarea, se entrena la red neuronal usando un conjunto de datos de entrenamiento 1102. Se han desarrollado diversas estructuras de entrenamiento 1104 para posibilitar la aceleración de hardware del proceso de entrenamiento. Por ejemplo, la estructura de aprendizaje automático 604 de la Figura 6 puede estar configurada como

una estructura de entrenamiento 604. La estructura de entrenamiento 604 puede engancharse a una red neuronal no entrenada 1106 y posibilita que se entrene la red neuronal no entrenada usando los recursos de procesamiento paralelo descritos en el presente documento para generar una red neuronal entrenada 1108.

- 5 Para iniciar el proceso de entrenamiento, pueden elegirse los pesos iniciales aleatoriamente o mediante entrenamiento previo usando una red de creencia profunda. El ciclo de entrenamiento puede realizarse a continuación de una manera supervisada o no supervisada.

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

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

35 Pueden emplearse también variaciones en el entrenamiento supervisado y no supervisado. El aprendizaje semisupervisado es una técnica en la que el conjunto de datos de entrenamiento 1102 incluye una mezcla de datos etiquetados y no etiquetados de la misma distribución. El aprendizaje incremental es una variante del aprendizaje supervisado en el que los datos de entrada se utilizan continuamente para entrenar aún más el modelo. El aprendizaje incremental permite que la red neuronal entrenada 1108 se adapte a los nuevos datos 1112 sin olvidar el conocimiento inculcado dentro de la red durante el entrenamiento inicial.

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

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

55 En el paralelismo de modelo 1202, diferentes nodos computacionales en un sistema distribuido pueden realizar cálculos de entrenamiento para diferentes partes de una única red. Por ejemplo, cada capa de una red neuronal puede entrenarse por un nodo de procesamiento diferente del sistema distribuido. Los beneficios del paralelismo de modelo incluyen la capacidad de escalar a modelos particularmente grandes. La división de los cálculos asociados con diferentes capas de la red neuronal posibilita el entrenamiento de redes neuronales muy grandes en las que los pesos para todas las capas no se ajustarían en la memoria de un único nodo computacional. En algunos casos, el paralelismo de modelo puede ser particularmente útil al realizar entrenamiento no supervisado de redes neuronales grandes.

60 En el paralelismo de datos 1204, los diferentes nodos de la red distribuida tienen una instancia completa del modelo y cada nodo recibe una porción diferente de los datos. Los resultados desde los diferentes nodos se combinan a continuación. Aunque son posibles diferentes enfoques para el paralelismo de datos, los enfoques de entrenamiento de datos paralelos requieren, todos ellos, una técnica de combinación de resultados y de sincronización de los parámetros de modelo entre cada nodo. Los enfoques ilustrativos para la combinación de datos incluyen promediado de parámetros y paralelismo de datos basado en actualizaciones. El promediado de parámetros entrena cada nodo

en un subconjunto de los datos de entrenamiento y establece los parámetros globales (por ejemplo, pesos, desviaciones) al promedio de los parámetros desde cada nodo. El promediado de parámetros usa un servidor de parámetros central que mantiene los datos de parámetro. El paralelismo de datos basado en actualizaciones es similar al promediado de parámetros excepto en que, en lugar de transferir parámetros desde los nodos al servidor de parámetros, se transfieren las actualizaciones al modelo. Adicionalmente, el paralelismo de datos basado en actualizaciones puede realizarse de una manera descentralizada, donde las actualizaciones se comprimen y se transfieren entre nodos.

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

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

Aplicaciones ilustrativas del aprendizaje automático

El aprendizaje automático puede aplicarse para resolver una diversidad de problemas tecnológicos, incluyendo, pero sin limitación, visión informática, conducción y navegación autónoma, reconocimiento de habla y procesamiento de idioma. La visión informática ha sido tradicionalmente una de las áreas de investigación más activas para aplicaciones de aprendizaje automático. Las aplicaciones de visión informática varían de reproducir capacidades visuales humanas, tales como reconocer caras, a crear nuevas categorías de capacidades visuales. Por ejemplo, las aplicaciones de visión informática pueden configurarse para reconocer ondas de sonido de las vibraciones inducidas en los objetos visibles en un vídeo. El aprendizaje automático acelerado por procesadores paralelos permite entrenar aplicaciones de visión informática utilizando un conjunto de datos de entrenamiento significativamente más grande de lo que antes era factible y permite implementar sistemas de inferencia utilizando procesadores paralelos de baja potencia.

El aprendizaje automático acelerado por procesador paralelo tiene aplicaciones de conducción autónoma que incluyen el reconocimiento de señales de carril y carretera, evitación de obstáculos, navegación y control de conducción. Las técnicas de aprendizaje automático aceleradas pueden usarse para entrenar modelos de conducción basándose en conjuntos de datos que definen las respuestas apropiadas a entrada de entrenamiento específica. El/los procesador(es) paralelos descritos en el presente documento pueden permitir un entrenamiento rápido de las redes neuronales cada vez más complejas utilizadas para soluciones de conducción autónoma y permiten la implementación de procesadores de inferencia de baja potencia en una plataforma móvil adecuada para la integración en vehículos autónomos.

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

El aprendizaje automático acelerado por procesador paralelo puede usarse también para acelerar el procesamiento de idioma natural. Los procedimientos de aprendizaje automático pueden hacer uso de algoritmos de inferencia estadística para producir modelos que son robustos ante una entrada errónea o no familiar. Las aplicaciones de procesador de idioma natural ilustrativas incluyen traducción de máquina automática entre idiomas humanos.

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

La **Figura 13** ilustra un sistema en chip (SOC) de inferencia 1300 ilustrativo adecuado para realizar una inferencia usando un modelo entrenado. El SOC 1300 puede integrar componentes de procesamiento que incluyen un procesador de medios 1302, un procesador de visión 1304, una GPGPU 1306 y un procesador de múltiples núcleos 1308. El SOC 1300 puede incluir adicionalmente la memoria en chip 1305 que puede posibilitar un agrupamiento de datos en chip compartido al que puede acceder cada uno de los componentes de procesamiento. Los componentes de procesamiento pueden optimizarse para una operación de baja potencia para posibilitar el despliegue en una diversidad de plataformas de aprendizaje automático, incluyendo vehículos autónomos y robots autónomos. Por

ejemplo, una implementación del SOC 1300 puede usarse como una porción del sistema de control principal para un vehículo autónomo. Donde el SOC 1300 está configurado para su uso en vehículos autónomos, el SOC se diseña y está configurado para cumplir con las normas de seguridad funcional relevantes de la jurisdicción de despliegue.

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

15 El procesador de múltiples núcleos 1308 puede incluir lógica de control para asistir con la secuenciación y la sincronización de transferencias de datos y operaciones de memoria compartida realizadas por el procesador de medios 1302 y el procesador de visión 1304. El procesador de múltiples núcleos 1308 también puede funcionar como un procesador de aplicaciones para ejecutar aplicaciones de software que pueden hacer uso de la capacidad de cálculo de inferencia de la GPGPU 1306. Por ejemplo, al menos una porción de la lógica de navegación y de conducción puede implementarse en software que se ejecuta en el procesador de múltiples núcleos 1308. Tal software puede emitir directamente cargas de trabajo computacionales a la GPGPU 1306 o las cargas de trabajo computacionales pueden emitirse al procesador de múltiples núcleos 1308, que puede descargar al menos una porción de esas operaciones a la GPGPU 1306.
20

25 La GPGPU 1306 puede incluir agrupaciones de cálculo, tal como una configuración de baja potencia de las agrupaciones de cálculo 706A-706H dentro de la unidad de procesamiento de gráficos de propósito general altamente paralela 700. Las agrupaciones de cálculo dentro de la GPGPU 1306 pueden soportar instrucciones que están optimizadas específicamente para realizar cálculos de inferencia en una red neuronal entrenada. Por ejemplo, la GPGPU 1306 puede soportar instrucciones para realizar cálculos de precisión baja, tales como operaciones vectoriales de números enteros de 8 bits y de 4 bits.
30

Detección y respuesta avanzadas de vehículos autónomos

35 Los sistemas de vehículos avanzados se pueden combinar con técnicas de aprendizaje automático para permitir la conducción semiautónoma y autónoma en los vehículos convencionales. Para mejorar las capacidades de los vehículos autónomos, las realizaciones descritas en el presente documento proporcionan múltiples funciones avanzadas de detección y cálculo. Una realización prevé una respuesta autónoma tras la colisión de vehículos. Una realización permite capacidades de búsqueda asistida utilizando capacidades de detección de automóviles. Una
40 realización proporciona técnicas para minimizar los daños por accidentes para vehículos autónomos a través de sensores internos del vehículo. Una realización permite compartir datos de inferencia entre vehículos autónomos. Cada una de las realizaciones descritas en el presente documento puede hacer uso de uno o más vehículos cercanos para cálculos antes, durante y después de las sesiones de conducción autónoma.

45 Los sistemas de conducción autónoma deben realizar una cantidad considerable de cálculos para, por ejemplo, planificar rutas, reconocer y evitar peatones y obstáculos, detectar y comprender semáforos, cruces de peatones y señales, y determinar el movimiento de objetos. Estas tareas no solo requieren un uso intensivo de cálculo, sino que a menudo deben realizarse bajo limitaciones de tiempo real. Las realizaciones descritas en el presente documento proporcionan técnicas para permitir que un vehículo autónomo haga uso de vehículos cercanos dentro de una red de
50 vehículos autónomos. Estas técnicas se pueden realizar como una alternativa de menor latencia o mayor confiabilidad a las técnicas de cálculo basadas en la nube.

La tecnología de comunicación de vehículo a vehículo está disponible para permitir que los vehículos reporten información como velocidad, posición e incidentes en la carretera a vehículos cercanos a través de una red inalámbrica ad-hoc. Las realizaciones descritas en el presente documento amplían esta capacidad de comunicación para permitir
55 que los vehículos cercanos realicen operaciones informáticas distribuidas para obtener resultados más rápidos o más confiables de lo que sería posible con los recursos de cálculo dentro de un solo vehículo. En una realización, para realizar operaciones de cálculo particularmente intensas en presencia de una red inalámbrica suficientemente rápida, un vehículo autónomo puede descargar un subconjunto de las operaciones de cálculo a un sistema de cálculo basado en la nube. Sin embargo, si no hay disponible una red inalámbrica de área amplia lo suficientemente rápida o si se requiere una respuesta de cálculo urgente, el vehículo puede intentar conectarse de forma inalámbrica con los
60 vehículos circundantes para establecer temporalmente una red de cálculo distribuida.

El consorcio de fabricantes de automóviles Car-to-X ha desarrollado un estándar de comunicaciones para vehículos que permite que los vehículos y los dispositivos fijos en la carretera intercambien información y advertencias a través de redes inalámbricas ad-hoc para mejorar la seguridad en la conducción. Las realizaciones descritas en el presente
65

documento amplían los estándares de comunicación bajo Car-to-X para permitir la funcionalidad distribuida para vehículos autónomos que tienen una relación de confianza establecida.

En una realización, los vehículos autónomos que tienen una relación de confianza se pueden configurar para compartir recursos de cálculo cuando sea necesario. Cada vehículo autónomo tiene un nivel básico de capacidad de cálculo. Sin embargo, pueden surgir circunstancias en las que el sistema de conducción autónoma de un vehículo determine que se pueden requerir recursos computacionales que excedan los recursos computacionales actuales del vehículo o que los recursos computacionales del vehículo no puedan producir los resultados deseados dentro de un período razonable. Estas circunstancias pueden surgir, por ejemplo, cuando un vehículo autónomo generalmente puede depender de recursos de cálculo basados en la nube, pero esos recursos no están disponibles temporalmente debido a una mala recepción de la red inalámbrica móvil. Las técnicas para la descarga de cálculo descritas en el presente documento se basan en la tecnología existente para la comunicación entre automóviles. La seguridad y privacidad de esas técnicas de comunicación se abordan en la capa de comunicación subyacente. Por lo tanto, se supone en todo momento la existencia de un protocolo de comunicación inalámbrica seguro y un método para determinar un estado de confianza para un vecino.

En una realización, los vehículos autónomos que están conectados a una red local o basada en la nube se pueden configurar para usar sensores externos para ayudar con funciones de búsqueda distribuida. Por ejemplo, los vehículos conectados a la red se pueden configurar para recibir datos de un sistema de alerta de sustracción de niños o un sistema de transmisión de emergencia similar. Los vehículos conectados a la red pueden utilizar cámaras y sensores a bordo utilizados para la conducción autónoma para buscar pasivamente vehículos identificados.

En una realización, los datos de inferencia generados por un vehículo conectado a la red se pueden compartir con otros vehículos conectados cercanos. Por ejemplo, un vehículo delantero en un grupo de múltiples vehículos autónomos puede compartir determinaciones de inferencia con otros vehículos que pueden no tener acceso al campo de visión del sensor del vehículo delantero. Si el vehículo delantero determina que hay un obstáculo presente o que ha ocurrido un evento peligroso, el vehículo puede compartir dichas determinaciones con los vehículos conectados cercanos.

Si bien los vehículos autónomos pueden contener una variedad de sensores y lógica de navegación para evitar colisiones, en algunas circunstancias las colisiones entre vehículos autónomos pueden ser inevitables. Una realización descrita en el presente documento puede aprovechar los sensores externos e internos de un vehículo autónomo para intentar minimizar el daño a los pasajeros dentro del vehículo. En una realización, los vehículos conectados pueden compartir datos de sensores internos e intentar realizar una minimización de daños multilateral una vez que uno o más de los vehículos determine que una colisión inevitable es inminente.

En una realización, los vehículos autónomos que han estado involucrados en una colisión pueden configurarse para realizar una respuesta autónoma a uno o más escenarios postcolisión. La respuesta postcolisión se puede configurar en función de los datos de sensores internos y externos que se recopilan antes, durante y después de la colisión. Se puede realizar una evaluación de daños y lesiones en función de los datos recopilados de los sensores y se puede entrenar un modelo de aprendizaje automático para seleccionar entre una variedad de posibles respuestas postcolisión.

Uso de múltiples vehículos para cálculo y detección para la conducción autónoma

La **Figura 14** ilustra una red de comunicaciones de vehículos 1400 utilizable mediante realizaciones descritas en el presente documento. Múltiples vehículos autónomos, incluidos vehículos de transporte público 1416 y vehículos de pasajeros 1418, pueden conectarse a una red (por ejemplo, Internet 1402) a través de múltiples puntos de acceso, incluido un acceso basado en satélite 1404 y/o una estación base de radio inalámbrica móvil 1406. Cada uno de los vehículos de transporte público 1416 y los vehículos de pasajeros 1418 incluye dispositivos de red inalámbrica que son configurables para conectarse a redes de área amplia, así como a redes de área local dentro del alcance de los vehículos. El acceso también puede estar disponible a través de uno o más puntos de acceso inalámbrico en la carretera 1412A-1412B. Los puntos de acceso inalámbrico en carretera 1412A-1412B y la estación base de radio inalámbrica móvil 1406 pueden conectarse con enrutadores de acceso 1410A-1410B que permiten el acceso a Internet 1402 u otra red de área amplia. Además, puede estar disponible la comunicación de vehículo a infraestructura (V2I). El sistema V2I incluye múltiples nodos de cálculo estacionarios en la carretera 1414A-1414B conectados a la red y a la infraestructura de la red eléctrica. El sistema V2I puede participar en la descarga de cálculo de manera similar a los vehículos autónomos cercanos.

Las realizaciones descritas en el presente documento proporcionan beneficios más allá de los sistemas convencionales de descarga de cálculo y de vehículo a vehículo al aprovechar la capacidad de los vehículos autónomos para coordinar la conducción autónoma y la funcionalidad de navegación. Por ejemplo, cuando se desea la descarga de cálculo de vehículo a vehículo, parte del proceso de selección entre posibles clientes de cálculo remoto es la capacidad de los vehículos autónomos para coordinar los datos de navegación y destino. Cuando varios vehículos autónomos comparten una ruta común, la lógica de gestión de cálculo asociada con los vehículos puede coordinar y determinar la cantidad de tiempo que los vehículos estarán dentro del alcance de la comunicación

inalámbrica. Además, los vehículos que tienen una disposición de cálculo confiable pueden maniobrar de forma autónoma en una formación que permita mejor el intercambio inalámbrico de datos, siempre que dichas maniobras no causen una desviación de las rutas de navegación y trayectoria previamente determinadas.

La **FIG. 15** ilustra un sistema de gestión de cálculo distribuido 1500, de acuerdo con una realización. En una realización, un conjunto de cargas de trabajo de cálculo 1510 para un vehículo autónomo puede ser procesado en parte por un gestor de cálculo local 1520 y en parte por un gestor de cálculo remoto 1530. El gestor de cálculo local 1520 es un gestor de cálculo que se ejecuta en una infraestructura de cálculo de un vehículo autónomo, por ejemplo, a través del sistema de cálculo que realiza operaciones de inferencia para el modelo de aprendizaje automático que permite la conducción y navegación autónomas. Las unidades de cálculo que ejecutan la carga de trabajo de cálculo descrita en el presente documento pueden ser cualquiera de las agrupaciones de cálculo, unidades de cálculo, bloques de cálculo, multiprocesadores de transmisión, multiprocesadores de gráficos o lógica de procesamiento de propósito general como se describe en el presente documento. El gestor de cálculo local 1520 puede facilitar las operaciones de cálculo local 1525 para una porción de las cargas de trabajo de cálculo 1510, mientras distribuye una porción de las cargas de trabajo de cálculo 1510 a un sistema de cálculo remoto 1535 gestionado por un gestor de cálculo remoto 1530 de un vehículo autónomo de confianza cercano.

El gestor de cálculo local 1520 y el gestor de cálculo remoto 1530 pueden incluir infraestructuras de cálculo comunes o compatibles. El gestor de cálculo local 1520 incluye una unidad de gestión de tiempo de ejecución 1526, un módulo de confianza 1527 y una unidad de descubrimiento remoto 1528. El gestor de cálculo remoto 1530 incluye módulos que son similares o compatibles con los módulos del gestor de cálculo local, incluyendo una unidad de gestión de tiempo de ejecución 1526, un módulo de confianza 1537, una unidad de descubrimiento remoto 1538 o unidades equivalentes. Las unidades de gestión de tiempo de ejecución 1526, 1536 rastrean la ejecución de cargas de trabajo de cálculo 1510 y son responsables de cargar y descargar cargas de trabajo de cálculo en unidades de cálculo locales o remotas asociadas. Los módulos de confianza 1527, 1537 mantienen testigos y certificados de identidad que se utilizan para verificar la autenticidad de los entornos de cálculo de los vehículos autónomos locales y remotos. El módulo de confianza 1527 también puede incluir claves o certificados que se usan para firmar o cifrar datos seguros transmitidos entre el gestor de cálculo local 1520 y el gestor de cálculo remoto 1530. Las unidades de descubrimiento remoto 1528, 1538 se utilizan para descubrir y localizar vehículos autónomos dentro del alcance inalámbrico que pueden participar en el intercambio de cálculo confiable. Las tareas de descubrimiento incluyen determinar qué vehículos autónomos pueden aceptar la descarga de cargas de trabajo de cálculo y determinar el conjunto de vehículos autónomos desde los cuales se aceptarán cargas de trabajo de cálculo.

El gestor de cálculo local 1520 puede ejecutar la lógica de descarga 1522 que incluye una unidad de subdivisión de carga de trabajo 1523 y una unidad de selección de vehículo remoto 1524. La lógica de descarga de ejecución 1522 está configurada para descargar cualquiera de las cargas de trabajo de cálculo 1510 que están disponibles para descargar a recursos de cálculo remoto en respuesta a una indicación de que el número de cargas de trabajo de cálculo excede la capacidad de cálculo local. La unidad de subdivisión de carga de trabajo 1523 puede examinar las cargas de trabajo de cálculo 1510 para determinar si alguna de las cargas de trabajo depende de la salida de cualquiera de las otras cargas de trabajo dentro de las cargas de trabajo de cálculo 1510. Las cargas de trabajo que tienen dependencias se pueden programar como una unidad para ejecución local o remota. La unidad de selección de vehículo remoto 1524 se utiliza para determinar cuáles de uno o más vehículos autónomos confiables que están dentro del alcance de la red inalámbrica tienen recursos de cálculo disponibles y si la latencia de esos vehículos es aceptable para las cargas de trabajo de cálculo que se programarán. La unidad de selección de vehículo remoto 1524 puede recibir actualizaciones periódicas desde la unidad de descubrimiento remoto 1528 y, basándose en diversos criterios, puede seleccionar uno o más vehículos remotos disponibles para descargar cargas de trabajo de cálculo. En una realización, se puede realizar una determinación de la latencia aceptable midiendo la latencia de comunicación a vehículos autónomos locales determinando si la latencia está por debajo de un umbral objetivo que se selecciona en función del tipo de descarga de cálculo que se va a realizar. Por ejemplo, las operaciones de cálculo sensibles a la latencia, tal como los cálculos basados en datos sensoriales en tiempo real, tendrán un umbral de latencia más bajo en relación con las operaciones de cálculo que son menos sensibles a la latencia, tal como la realización especulativa de cálculos de rutas alternativas basados en información de tráfico actualizada.

La unidad de selección de vehículo remoto 1524 puede usar una o más de múltiples técnicas de selección de vehículos para seleccionar un vehículo remoto al que se van a descargar las cargas de trabajo de cálculo. En diversas realizaciones, los vehículos remotos se pueden seleccionar basándose en una estrategia de selección aleatoria, una estrategia de selección basada en la capacidad de cálculo, una estrategia de selección basada en la distancia o una estrategia de selección de múltiples elementos que utiliza múltiples estrategias de selección. La estrategia de selección aleatoria selecciona un nodo de vehículo al azar y luego calcula la cantidad de cargas de trabajo de cálculo que se pueden realizar. Si el nodo seleccionado puede completar una o más cargas de trabajo, entonces se descargarán una o más cargas de trabajo al nodo. Si el nodo seleccionado aleatoriamente no puede realizar al menos una carga de trabajo, se evaluará un nodo aleatorio diferente. Se puede realizar una estrategia similar para la capacidad de cálculo o la estrategia de selección basada en la distancia, excepto que la selección inicial y posterior de vehículos vecinos se realiza en función de los vehículos que tienen la mayor capacidad de cálculo disponible o que están más cerca en distancia del vehículo autónomo local. La estrategia de selección de múltiples elementos selecciona un nodo de vehículo para descargar considerando múltiples parámetros de los nodos de vehículo sustitutos.

Además de las técnicas de selección descritas anteriormente, una realización descrita en el presente documento permite una estrategia de selección basada en navegación. La estrategia de selección basada en navegación considera el conjunto de vehículos autónomos de confianza vecinos y clasifica el conjunto de vehículos en función de un destino de navegación común o una ruta de navegación común. Los vehículos autónomos vecinos que tienen una relación de confianza pueden intercambiar de forma segura datos de destino y ruta de navegación. Si uno o más vehículos autónomos vecinos comparten un destino común o viajan a lo largo de una ruta común durante un período, los vehículos autónomos pueden establecer una relación de cálculo distribuido y descargarse mutuamente cargas de trabajo de cálculo durante la duración de la ruta común. La estrategia de selección basada en navegación agrega un elemento adicional de previsibilidad a los movimientos relativos de los vehículos autónomos dentro de la red confiable, de modo que cada vehículo local y remoto puede estimar cuándo otros vehículos entrarán y saldrán del rango de comunicación inalámbrica local.

Los vehículos autónomos conectados capaces de conectarse a centros de datos basados en la nube tienen acceso a una amplia gama de capacidades de cálculo a expensas de cierto grado de latencia debido a la red de comunicación inalámbrica o satelital de banda ancha a través de la cual el vehículo autónomo conectado se comunica con el centro de datos. Para cargas de trabajo que tienen una alta demanda computacional, los vehículos autónomos conectados pueden descargar una porción de esas cargas de trabajo al centro de datos a través de la red de comunicaciones. Sin embargo, pueden surgir circunstancias en las que un vehículo autónomo no pueda acceder a un centro de datos basado en la nube. En tales circunstancias, la capacidad de establecer una conexión de cálculo distribuido confiable con vehículos autónomos vecinos puede ser particularmente óptima.

La **Figura 16** ilustra una relación 1600 entre la potencia computacional y la latencia de comunicación en un vehículo autónomo local 1602 como se describe en el presente documento. En una realización, el vehículo autónomo local 1602 puede conectarse a una nube de vehículo autónomo y a un centro de datos 1606 a través de una red de comunicaciones de área amplia, tal como una red de comunicación móvil inalámbrica o basada en satélite. Tales conexiones permiten que el vehículo autónomo local 1602 acceda a una mayor potencia computacional a expensas de una mayor latencia de comunicación. Alternativamente, el vehículo autónomo local 1602 puede acceder a recursos de cálculo de vehículos autónomos vecinos y nodos de cálculo V2I en la carretera con latencia reducida, aunque esos recursos de cálculo tendrán una potencia computacional reducida en relación con la nube del vehículo autónomo y el centro de datos 1606.

En una realización, las técnicas de descarga de cálculo se optimizan principalmente para su uso en situaciones en las que los recursos de cálculo en el vehículo son insuficientes y la latencia de comunicación en la nube es demasiado grande. La decisión de descargar el cálculo a vehículos de confianza vecinos o a la nube considera numerosos factores que incluyen la cantidad de cálculo a realizar y las latencias de comunicación desde el vehículo autónomo local 1602 a los nodos V2I y AV vecinos 1604 o a una nube de vehículo autónomo y centro de datos 1606.

La comunicación con los AV vecinos y los nodos V2I 1604 puede utilizar tecnologías emergentes tales como Comunicaciones Dedicadas de Corto Alcance (DSRC), que permiten una transmisión de datos muy alta en un rango de corto a medio. DSRC se puede utilizar para la comunicación de datos en la banda de radio industrial, científica y médica (ISM) de 5,725 GHz a 5,875 GHz. Los sistemas DSRC constan de unidades en la carretera y unidades a bordo con transceptores y transpondedores. Las unidades a bordo están ubicadas dentro de vehículos autónomos, robots y/u otras máquinas que pueden comunicarse a través de una red DSRC.

El uso de tecnologías tal como DSRC, latencia de comunicación de automóvil a automóvil y V2I a menudo puede proporcionar un mayor ancho de banda y/o comunicaciones de menor latencia en relación con el enlace de comunicación con un centro de datos en la nube, particularmente en condiciones donde la conectividad de la nube es deficiente. Por ejemplo, si bien las fuertes lluvias pueden detener o ralentizar una conexión a Internet por satélite, DSRC no se ve afectado por condiciones climáticas extremas. Además, tales condiciones pueden ser aquellas en las que la capacidad de descarga de cálculo sea más útil, ya que las condiciones climáticas extremas pueden hacer que las tareas de conducción autónoma sean más desafiantes.

La **Figura 17** es un diagrama de flujo que ilustra la lógica 1700 de descarga de cálculo de un vehículo autónomo, de acuerdo con una realización. La lógica de descarga de cálculo de vehículo autónomo 1700 se habilita, en una realización, mediante la lógica de descarga 1522 como en la Figura 15. En una realización, la lógica de descarga de cálculo de vehículo autónomo 1700 puede determinar que se va a procesar una carga de trabajo computacional como se muestra en el bloque 1702. La lógica de descarga de cálculo de vehículo autónomo 1700 puede entonces determinar si la latencia de comunicación en la nube actual es aceptable o está dentro de un intervalo aceptable, como se muestra en el bloque 1703. Cuando la latencia de comunicación en la nube actual es aceptable, la lógica de descarga de vehículo autónomo puede despachar al menos una porción de la carga de trabajo para su procesamiento a través de un centro de datos remoto, como se muestra en el bloque 1704. Sin embargo, si la latencia de comunicación en la nube no es aceptable en el bloque 1703, la lógica de descarga de cálculo de vehículo autónomo 1700 puede determinar si los recursos de cálculo de los vehículos vecinos están disponibles y son suficientes en el bloque 1706. Si los recursos de cálculo de los vehículos vecinos no son suficientes, la lógica de descarga de cálculo de vehículo autónomo 1700 puede usar recursos locales para realizar las operaciones de cálculo, como se muestra en el bloque

1705. Si los recursos de cálculo de los vehículos vecinos están disponibles y son suficientes en el bloque 1706, pero la latencia de comunicación con esos vehículos es inaceptable en el bloque 1708, la lógica de descarga de cálculo de vehículo autónomo 1700 puede usar recursos locales para el cálculo en el bloque 1705. Si la latencia de comunicación es aceptable en el bloque 1708, la lógica de descarga de cálculo de vehículo autónomo 1700 puede enviar al menos una porción de la carga de trabajo para su procesamiento en vehículos autónomos vecinos o nodos V2I, como se muestra en el bloque 1710. Si solo se usan recursos locales para el cálculo en el bloque 1705, parte de la capacidad del vehículo autónomo puede limitarse o reducirse, como se muestra en el bloque 1707. Por ejemplo, se puede reducir la velocidad máxima a la que el vehículo autónomo puede viajar en modo totalmente autónomo. Alternativamente, se puede reducir la velocidad a la que un vehículo autónomo puede seleccionar rutas alternativas en función de las condiciones dinámicas del tráfico.

En una realización, los vehículos autónomos que están conectados a una red local o basada en la nube se pueden configurar para usar sensores externos para ayudar con funciones de búsqueda distribuida. Por ejemplo, los vehículos conectados a la red se pueden configurar para recibir datos de un sistema de alerta de sustracción de niños o un sistema de transmisión de emergencia similar. Los vehículos conectados a la red pueden utilizar cámaras y sensores a bordo utilizados para la conducción autónoma para buscar pasivamente vehículos identificados.

La **Figura 18** ilustra el uso de sensores de vehículos autónomos para ayudar en una búsqueda distribuida. En una realización, uno o más vehículos autónomos pueden recibir un descriptor o a través de una red de vehículos autónomos basada en la nube, V2I y/o vecina. El descriptor identifica un objetivo de búsqueda 1802. Los vehículos autónomos que reciben el descriptor pueden habilitar un elemento de trabajo de búsqueda en segundo plano que analiza pasivamente los datos de los sensores recibidos a través de sensores externos utilizados para permitir tareas de conducción autónoma (por ejemplo, cámaras, radar, Lidar, etc.). Cualquier objeto que se ocupe del área de detección de sensor 1820, 1830 de vehículos autónomos que estén ejecutando la tarea de búsqueda se puede clasificar para determinar si el objeto coincide con los detalles proporcionados para el objetivo de búsqueda 1802. Por ejemplo, un descriptor de búsqueda puede indicar que el objetivo de búsqueda tiene un tipo de matrícula o número de matrícula específico. Alternativamente, el descriptor para el objetivo de búsqueda 1802 puede ser una descripción más general de un vehículo, tal como marca, modelo y color del vehículo. Si un vehículo identificado por el objetivo del descriptor de búsqueda pasa al área de detección de sensor 1820, 1830 de un vehículo autónomo conectado, el vehículo puede enviar información tal como datos de cámaras y sensores al iniciador de la búsqueda. En una realización, se puede habilitar una búsqueda más general para el objetivo de búsqueda 1802. Por ejemplo, los vehículos autónomos se pueden configurar para buscar vehículos que exhiban un comportamiento de conducción aberrante, tal como cambios repetidos de carril inseguros o cambios rápidos de velocidad. Estos datos pueden transmitirse a las autoridades de seguridad pertinentes de la región por la que viajan los vehículos autónomos. Estos datos también se pueden transmitir a vehículos cercanos que no tengan el objetivo de búsqueda 1802 dentro del área de detección de sensor relevante para el vehículo.

La **Figura 19** ilustra la lógica de búsqueda de vehículos autónomos distribuidos 1900, de acuerdo con una realización. En una realización, la lógica de búsqueda de vehículos autónomos 1900 puede ejecutarse como una de las cargas de trabajo de cálculo 1510 en los recursos de cálculo de un vehículo autónomo local, como se muestra en la Figura 15. Los datos de búsqueda y/o la carga de trabajo de búsqueda también se pueden distribuir a vehículos autónomos vecinos a través de una red inalámbrica de corto alcance.

En una realización, un nodo de cálculo en un vehículo autónomo puede recibir un descriptor que identifica un objetivo de búsqueda, como se muestra en el bloque 1902. El descriptor puede identificar una o más marcas, modelos, colores o un identificador numérico específico asociado con el vehículo. El identificador numérico específico del vehículo puede incluir, entre otros, un número de placa o un número de identificador diferente, tal como un número de identificación de vehículo de servicio público, un número de vehículo de transporte público o un número de identificación de vehículo de transporte privado. En una realización, el descriptor puede identificar un patrón de comportamiento en lugar de un vehículo específico. Por ejemplo, el descriptor de búsqueda puede especificar una búsqueda general de comportamiento de conducción errático. En una realización, el descriptor se recibe desde una red de vehículos conectados. Una realización también está configurada para recibir descriptores de búsqueda desde un sistema de transmisión de emergencia tal como el sistema inalámbrico de alerta de emergencia.

En una realización, el descriptor puede identificar un área de búsqueda en la que se espera que esté ubicado el objetivo de búsqueda. En tal realización, el nodo de cálculo puede hacer referencia a datos de ubicación para determinar si el vehículo autónomo está dentro del área de búsqueda definida en el bloque 1903. Si el vehículo autónomo no está en el área de búsqueda, el nodo de cálculo puede simplemente salir en el bloque 1905. Alternativamente, la lógica relacionada con la búsqueda del nodo de cálculo puede entrar en un estado de suspensión hasta que se reciba un nuevo descriptor en el bloque 1902.

En una realización, si el vehículo autónomo está en el área de búsqueda identificada, el nodo de cálculo puede generar una instancia de la lógica de búsqueda de vehículo autónomo 1900 en respuesta a la recepción del descriptor, como se muestra en el bloque 1904. Sin embargo, en algunas realizaciones, la lógica de búsqueda de vehículos autónomos 1900 se genera e inicializa durante el inicio del nodo de cálculo y entra en un estado de suspensión hasta que se recibe un descriptor. La lógica de búsqueda de vehículos autónomos 1900 puede luego configurar un elemento de

trabajo para su ejecución en el nodo de cálculo para buscar pasivamente un vehículo o patrón de conducción que corresponda con el descriptor de búsqueda, como se muestra en 1906. El elemento de trabajo configurado puede señalar la lógica de búsqueda de vehículos autónomos 1900 si se localiza un candidato de búsqueda que coincida con el descriptor de búsqueda.

En una realización, la lógica de búsqueda de vehículos autónomos 1900 puede propagar los datos de búsqueda a cualquier vehículo autónomo vecino confiable o nodos de vehículo a infraestructura que no tengan el descriptor de búsqueda. Si un vehículo autónomo confiable o un nodo V2I ingresa al alcance inalámbrico local, la lógica de búsqueda de vehículos autónomos 1900 puede determinar si un vehículo autónomo vecino o un nodo V2I no tiene los datos del descriptor de búsqueda en el bloque 1907. La lógica de búsqueda de vehículos autónomos 1900 puede entonces transmitir un descriptor de búsqueda al vehículo autónomo vecino confiable o al nodo V2I en el bloque 1908 del vehículo o nodo que no tiene los datos del descriptor de búsqueda.

En una realización, los vehículos conectados descritos en el presente documento generalmente pueden compartir datos de inferencia con vehículos conectados vecinos dentro de cierta proximidad física. El intercambio de inferencias se puede realizar a través de una conexión de radio inalámbrica confiable de corto alcance como se describe en el presente documento, incluida, entre otras, una conexión DSRC. El intercambio de inferencias también se puede extender más allá de los vehículos autónomos a drones autónomos, robots de transporte autónomos o cualquier otra máquina capaz de realizar navegación autónoma a través de un modelo de aprendizaje automático. En una realización, las determinaciones de inferencia realizadas basándose en entradas en tiempo real se transmiten de forma segura a vehículos autónomos vecinos como avisos de información. Los vehículos y/o máquinas autónomos dentro del alcance inalámbrico local pueden utilizar la información de inferencia transmitida para aumentar las determinaciones de inferencia local basadas en sensores locales. Dicho intercambio de información puede permitir que los vehículos o máquinas observen o reciban aviso de peligros o condiciones circundantes que no pueden ser detectados directamente por los sensores locales en esos vehículos o máquinas.

La **Figura 20** ilustra el intercambio de datos de inferencia entre vehículos autónomos, de acuerdo con una realización. En una realización, un conjunto de múltiples vehículos autónomos puede incluir un vehículo delantero 2010 y un vehículo trasero 2020, aunque el conjunto de vehículos puede incluir más de dos vehículos. Una realización permite que el conjunto de múltiples vehículos autónomos se conecte a través de un sistema de comunicación inalámbrico seguro, tal como una conexión Wi-Fi o DSRC. El vehículo delantero 2010 tiene un área de detección de sensor 2004 que incluye áreas que pueden no ser detectables por el vehículo trasero 2020. Cuando el módulo de inferencia del vehículo delantero 2010 detecta que un obstáculo o evento peligroso 2002 está presente a lo largo de una ruta común del conjunto de múltiples vehículos autónomos, la determinación de inferencia se puede compartir con todos los vehículos en el conjunto de múltiples vehículos autónomos, incluido el vehículo trasero 2020.

La **Figura 21** es un diagrama de flujo que ilustra la lógica de intercambio de inferencias 2100, de acuerdo con una realización. En una realización, la lógica de intercambio de inferencia 2100 puede ser una de las cargas de trabajo de cálculo 1510 ejecutadas en un gestor de cálculo local 1520, como en la Figura 15, y puede transmitir información de inferencia para calcular cargas de trabajo gestionadas por un gestor de cálculo remoto 1530. Por ejemplo, y en una realización, el intercambio de inferencias se realiza entre vehículos autónomos mediante una descarga mutua de cargas de trabajo de cálculo que están configuradas para intercambiar datos de inferencia que están determinados por los respectivos modelos de cálculo locales utilizados para la conducción autónoma.

En una realización, la lógica de intercambio de inferencias 2100 puede establecer una conexión inalámbrica segura con un vehículo autónomo en proximidad física, como se muestra en el bloque 2102. La lógica de intercambio de inferencias 2100 puede entonces intercambiar información de confianza a través de una conexión inalámbrica segura para establecer un enlace de cálculo confiable con el vehículo autónomo, como se muestra en el bloque 2104. La información de confianza pueden ser testigos de identidad y certificados que se usan para verificar la autenticidad de los entornos de cálculo del vehículo autónomo local y remoto y pueden mantenerse mediante un módulo de confianza dentro de cada vehículo (por ejemplo, el módulo de confianza 1527 y el módulo de confianza 1537 como en la Figura 15). La información de confianza puede permitir que los vehículos establezcan un entorno de cálculo distribuido confiable para permitir que los datos y las cargas de trabajo se transfieran entre los entornos de cálculo de los vehículos. En una realización, después de establecer el enlace de cálculo confiable dentro de los vehículos autónomos, la lógica de intercambio de inferencias 2100 puede intercambiar manejadores de datos de inferencia entre vehículos autónomos, como se muestra en el bloque 2106. El intercambio de manejadores de datos de inferencia puede facilitar cualquier precisión de datos o traducción de formato que pueda ser necesaria durante el intercambio de información de inferencia, por ejemplo, si los modelos de datos utilizados por los distintos vehículos no son estrictamente compatibles. El intercambio de manejadores de datos de inferencia entre vehículos autónomos se puede realizar de manera similar a descargar una tarea de cálculo entre los distintos vehículos. Sin embargo, en una realización el intercambio de datos de inferencia se puede realizar sin requerir un intercambio de manejadores de datos de inferencia.

La lógica de intercambio de inferencias 2100 puede luego continuar realizando operaciones de inferencia generales usando un modelo de aprendizaje automático durante operaciones autónomas basadas en datos de sensores locales, como se muestra en el bloque 2108. En paralelo, la lógica de intercambio de inferencias 2100 puede luego transmitir

determinaciones de inferencia para un obstáculo o evento peligroso detectado por datos de sensores locales en el bloque 2110 y/o recibir determinaciones de inferencia para un obstáculo o evento peligroso determinado a través de datos de sensores remotos en el bloque 2112.

5 **Minimizar los daños por accidentes de vehículos autónomos mediante sensores internos**

Los vehículos autónomos, al menos durante el funcionamiento autónomo, tienen plena conciencia situacional de los vehículos que los rodean en la medida en que los sensores locales o remotos pueden ver o percibir el entorno del vehículo autónomo y la posición, velocidad, aceleración y orientación de otros vehículos cercanos. En los vehículos autónomos convencionales esa capacidad se utiliza para navegar de forma autónoma entre ubicaciones y evitar colisiones durante la navegación. Sin embargo, si surge un escenario en el que un modelo de aprendizaje automático dentro de un vehículo autónomo determina que no se puede evitar una colisión, el vehículo puede maniobrar de forma autónoma para minimizar la cantidad de daño humano (por ejemplo, lesiones) resultante de la colisión. Si el vehículo autónomo determina que una colisión es inevitable, el vehículo puede, dentro del plazo permitido previo a la colisión, ajustar la posición y orientación del vehículo para reducir la cantidad de fuerza que se aplicará a los pasajeros. Por ejemplo, y en una realización, si el vehículo autónomo determina que una colisión frontal es inevitable, el vehículo puede realizar maniobras de último momento para minimizar los daños ajustando el ángulo en el que se producirá la colisión. En una realización, el vehículo puede realizar de forma autónoma una o más actividades previas a la colisión, tales como preinflar uno o más airbags dentro de la cabina de pasajeros del vehículo, preparar uno o más mecanismos de control de daños dentro del bastidor del vehículo, y/o preinflar uno o más airbags externos del vehículo autónomo. En una realización, el vehículo puede aprovechar los sensores internos para determinar la posición y orientación de los ocupantes del vehículo. Con conocimiento de la posición y orientación de los ocupantes del vehículo, el vehículo autónomo puede determinar un área de impacto ideal para una colisión que minimice el daño potencial a los ocupantes del vehículo.

La **Figura 22** ilustra un vehículo autónomo 2200 configurado para minimizar los daños por accidentes de vehículos autónomos a través de sensores internos, de acuerdo con una realización. En una realización, el vehículo autónomo 2200 incluye áreas de detección de sensores externos frontales y de lectura 2204A-2204B para detectar otros vehículos, obstáculos, señales de tráfico, marcas viales y otra información que sea útil para la navegación autónoma. El vehículo autónomo 2200 también puede incluir áreas de detección de sensores laterales (no mostradas) para detectar la presencia de otros vehículos u obstáculos (por ejemplo, barandillas, etc.) que están ubicados a cada lado del vehículo. El vehículo autónomo 2200 incluye un conjunto de sensores internos 2206 que se utilizan para determinar la posición de cada ocupante del vehículo. Los sensores internos 2206 pueden incluir una variedad de sensores, incluidos sensores de proximidad y/o sensores de asiento que pueden usarse para determinar qué asientos dentro del vehículo están ocupados. Basándose en estos datos del sensor, el vehículo autónomo 2200 puede mantener un mapa de posiciones ocupadas dentro del vehículo. El vehículo puede actualizar periódicamente este mapa si los pasajeros se mueven dentro del vehículo durante el modo de conducción autónoma. Usando los sensores internos 2206, el vehículo autónomo 2200 puede determinar un área de impacto ideal 2202 para el vehículo de manera que, si ocurre una colisión, se produzca la menor cantidad de lesiones para los ocupantes. La zona de impacto ideal 2202, por ejemplo, puede ser una zona desocupada del vehículo. Si bien el vehículo autónomo 2200 intentará evitar colisiones con otros vehículos u objetos, si el vehículo determina que una colisión es inevitable, por ejemplo, debido a acciones de otros vehículos o una pérdida incontrolable de tracción, el vehículo puede maniobrar de forma autónoma hasta una posición para aceptar la colisión en el área de impacto ideal 2202.

Además de maniobrar o intentar maniobrar para recibir una colisión inevitable en el área de impacto ideal 2202, el vehículo autónomo 2200 puede iniciar uno o más mecanismos de mitigación de colisiones. El mecanismo de mitigación de colisiones puede incluir airbags internos y/o externos que se preinflan antes de la colisión, la activación de los pretensores de los cinturones de seguridad y la habilitación de cualquier sistema de mitigación de daños estructurales que pueda estar presente dentro del vehículo.

La **Figura 23** es un diagrama de flujo que ilustra la lógica de mitigación y evitación de colisiones 2300, de acuerdo con una realización. En una realización, la lógica de mitigación y evitación de colisiones 2300 puede determinar, mediante un conjunto de sensores internos, un conjunto de ubicaciones ocupadas dentro de un vehículo autónomo, como se muestra en el bloque 2302. En función del conjunto de ubicaciones ocupadas, la lógica de mitigación y evitación de colisiones 2300 puede determinar un área de impacto ideal para el vehículo, como se muestra en el bloque 2304. En una realización, el lugar de impacto ideal es un lugar dentro del vehículo que no esté ocupado por un pasajero. La lógica de mitigación y evitación de colisiones 2300 puede entonces habilitar la navegación autónoma con evitación de colisiones, como se muestra en el bloque 2306. En una realización, la lógica de mitigación y evitación de colisiones 2300 puede actualizar periódicamente una evaluación de ubicaciones ocupadas dentro del vehículo autónomo. La lógica de mitigación y evitación de colisiones 2300 también puede actualizar la evaluación de ubicaciones ocupadas en respuesta a la detección de movimiento dentro del vehículo. Si la lógica de mitigación y evitación de colisiones 2300 detecta una colisión potencial a través de sensores externos en el bloque 2308, la lógica de mitigación y evitación de colisiones 2300 puede determinar si la colisión es una colisión evitable en el bloque 2903. Si la colisión es una colisión evitable, la lógica de mitigación y evitación de colisiones 2300 puede hacer que el vehículo autónomo maniobre para evitar la colisión en el bloque 2310. Si la colisión no se puede evitar, la lógica de mitigación y evitación de colisiones 2300 puede iniciar la lógica de mitigación de colisiones en el bloque 2312. La lógica de mitigación de

colisiones puede hacer que el vehículo realice operaciones que incluyen ajustar la orientación del vehículo de modo que el área de colisión esté dentro del área de impacto ideal y/o permitir cualquiera o todas las técnicas de mitigación de daños descritas en el presente documento.

5 **Respuesta autónoma al escenario postcolisión**

En una realización, los vehículos autónomos que han estado involucrados en una colisión pueden configurarse para realizar una respuesta autónoma a uno o más escenarios postcolisión. La respuesta postcolisión se puede configurar en función de los datos de sensores internos y externos que se recopilan antes, durante y después de la colisión. Se puede realizar una evaluación de daños y lesiones en función de los datos recopilados de los sensores y se puede entrenar un modelo de aprendizaje automático para seleccionar entre una variedad de posibles respuestas postcolisión.

La **Figura 24** ilustra un sistema autónomo postcolisión 2400, de acuerdo con una realización. El sistema autónomo postcolisión 2400 incluye un conjunto de sensores de vehículo 2420 para evaluar el estado del vehículo después de una colisión. En una realización, los sensores de vehículo 2420 incluyen, entre otros, una o más cámaras 2422, giroscopios 2424 y Lidar 2426 que pueden usarse para determinar la ubicación, posición, orientación y entorno local del vehículo. Los sensores de vehículo 2420 también pueden incluir sensores de daño 2428 y sensores de detección de colisión 2430 y/o lógica de detección de colisiones que se pueden activar en respuesta a la detección de una desaceleración rápida por parte del vehículo junto con un choque físico detectado por los giroscopios 2424. El sistema autónomo postcolisión 2400 también puede incluir un conjunto de sensores de supervisión de salud de pasajero 2432, incluidas cámaras o sensores internos que pueden recopilar datos que se utilizan para evaluar la salud de los pasajeros en un escenario postcolisión. Por ejemplo, y en una realización, el sistema autónomo postcolisión 2400 incluye una o más cámaras infrarrojas internas que pueden usarse para detectar marcadores de peligro fisiológico entre los pasajeros de un vehículo autónomo después de una colisión de vehículo.

Los datos de los sensores de vehículo 2420 y los sensores de supervisión de salud de pasajero 2432 se pueden proporcionar a una estructura de aprendizaje automático 2440 que se ejecuta en recursos de cálculo dentro del vehículo autónomo. La estructura de aprendizaje automático puede proporcionar los datos de entrada a un modelo de aprendizaje automático 2442 que puede generar una respuesta de colisión inferida 2450. La respuesta de colisión inferida 2450 puede incluir un conjunto de acciones que, si es posible, el vehículo autónomo realizará automáticamente en respuesta al escenario actual postcolisión.

En una realización, basándose en la salud de los pasajeros y las capacidades del vehículo, la respuesta de colisión inferida 2450 incluye transmitir un informe de incidente a las autoridades pertinentes, que pueden incluir organizaciones policiales estatales y/o locales, compañías de seguros y/o fabricantes de vehículos. El informe de incidente se puede transmitir a través de una red de comunicaciones de un vehículo 1400 conectada como en la Figura 14. El informe de incidente puede incluir telemetría del vehículo inmediatamente antes, durante y después de la colisión, así como cualquier información que pueda determinarse para otros vehículos involucrados en la colisión. Por ejemplo, la información sobre cualquier otro vehículo involucrado en la colisión se puede determinar basándose en los datos recopilados por los sensores de vehículo 2420. Además, el informe de incidente puede incluir las coordenadas GPS o la ubicación del mapa en la que ocurrió el incidente. Si los sensores de supervisión de salud de pasajero detectan una lesión grave en un pasajero, se puede enviar un mensaje automático a los servicios de emergencia con el estado de la lesión del pasajero y la ubicación del incidente. En una realización, el sistema autónomo postcolisión 2400 puede intentar solicitar a los pasajeros un informe verbal de lesiones después de una colisión. El informe verbal de lesiones o una indicación de falta de respuesta por parte de un pasajero también puede transmitirse a los servicios de emergencia. En una realización, los componentes adicionales de una respuesta de colisión inferida 2450 pueden incluir recopilar y transmitir datos de sensores con respecto al entorno inmediato del vehículo, incluidas imágenes de sensores de peligros ambientales tales como una obstrucción de la carretera que puede haber contribuido a la colisión. En una realización, la respuesta de colisión inferida 2450 también puede incluir contactar automáticamente con cualquier contacto de emergencia que pueda configurarse para los ocupantes del vehículo.

La **Figura 25** ilustra detalles de la lógica autónoma postcolisión 2500, de acuerdo con una realización. La lógica autónoma postcolisión 2500 se puede ejecutar en los recursos de cálculo descritos en el presente documento y se puede iniciar en respuesta a una colisión detectada. En respuesta a una colisión detectada, la lógica autónoma postcolisión 2500 puede compilar un informe de incidente basado en datos de sensor, como se muestra en el bloque 2502. En una realización, el grado de daño detectado por la lógica autónoma postcolisión 2500 determina el grado de respuesta. La lógica autónoma postcolisión 2500 puede determinar la gravedad del daño o lesión basándose en los detalles del informe de lesiones en el bloque 2504. En una realización, el informe de incidente también incluye información recopilada mediante una solicitud de estado verbal de cualquier pasajero que pueda haber estado dentro del vehículo durante la colisión. Sin embargo, en algunas realizaciones la lógica autónoma postcolisión 2500 se puede implementar después de una colisión de un vehículo autónomo de transporte de carga desocupado. En dicho sistema, no se anticiparán lesiones a los pasajeros.

Si la lógica autónoma postcolisión 2500, en el bloque 2505, determina que se han producido daños graves al vehículo o cualquier lesión del pasajero, la lógica autónoma postcolisión 2500 puede transmitir el informe de incidente a los

servicios de emergencia apropiados para la ubicación, como se muestra en el bloque 2506. El informe de incidente se puede transmitir a través de la red del vehículo descrita en el presente documento o mediante técnicas de comunicación inalámbrica móvil convencionales. Una vez que se ha transmitido el informe de incidente, la lógica autónoma postcolisión 2500 puede activar cualquier característica de seguridad postcolisión disponible en el bloque 2508. Las características de seguridad postcolisión pueden incluir luces de peligro o de advertencia. En una realización, la lógica autónoma postcolisión 2500 también puede desconectar la energía principal del vehículo mientras se mantiene la energía auxiliar para los recursos de cálculo. Para los vehículos eléctricos, esto puede incluir desconectar la energía de los motores eléctricos primarios. Para los vehículos autónomos de combustión interna, el motor de combustión interna primario se puede desactivar automáticamente. En una realización, una o más puertas del vehículo se pueden desbloquear automáticamente.

En una realización, la lógica autónoma postcolisión 2500 puede evitar la transmisión del informe de incidente a los servicios de emergencia si la lógica determina en el bloque 2505 que no se han producido daños graves al vehículo ni lesiones a los pasajeros. En cambio, la lógica autónoma postcolisión 2500 puede determinar en el bloque 2507 si el vehículo es móvil después de la colisión. La determinación de la movilidad del vehículo se puede realizar basándose en sensores de daños dentro del vehículo que pueden determinar un estado operativo del vehículo. Si la lógica autónoma postcolisión 2500 determina que el vehículo es móvil en el bloque 2507, la lógica puede maniobrar de forma autónoma el vehículo hasta una ubicación segura cercana en el bloque 2509. Por ejemplo, la lógica autónoma postcolisión 2500 puede maniobrar de forma autónoma el vehículo hacia el costado de la carretera y fuera del flujo de tráfico. En una realización, el vehículo se maniobra de forma autónoma solo si la maniobra se puede realizar sin causar lesiones al pasajero o lesiones a los peatones o transeúntes cercanos. Si la lógica autónoma postcolisión 2500 determina que el vehículo no es móvil en el bloque 2507, la lógica puede activar las características de seguridad postcolisión descritas en el presente documento en el bloque 2508. En cada escenario, en una realización la lógica autónoma postcolisión 2500 puede transmitir el informe de incidente al fabricante del vehículo, como se muestra en el bloque 2510. En una realización, el informe de incidente que se transmite al fabricante del vehículo se centra principalmente en el vehículo y se centra en la telemetría pre y postcolisión. Es posible que se elimine información personal específica de dicho informe antes de su transmisión.

Sistema de procesamiento de gráficos ilustrativo adicional

Los detalles de las realizaciones descritas anteriormente pueden incorporarse dentro de los sistemas y dispositivos de procesamiento de gráficos descritos a continuación. Los dispositivos y el sistema de procesamiento de gráficos de las Figuras 26-39 ilustran hardware de procesamiento de gráficos y sistemas alternativos que pueden implementar todas y cada una de las técnicas descritas anteriormente.

Vista global del sistema de procesamiento de gráficos ilustrativo adicional

La **Figura 26** es un diagrama de bloques de un sistema de procesamiento 2600, de acuerdo con una realización. En diversas realizaciones, el sistema 2600 incluye uno o más procesadores 2602 y uno o más procesadores de gráficos 2608, 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 2602 o núcleos de procesador 2607. En una realización, el sistema 2600 es una plataforma de procesamiento incorporada dentro de un circuito integrado de sistema en chip (SoC) para su uso en dispositivos móviles, de mano o integrados.

Una realización del sistema 2600 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 2600 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 2600 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 2600 es un dispositivo de televisión o de descodificador de salón que tiene uno o más procesadores 2602 y una interfaz gráfica generada por uno o más procesadores de gráficos 2608.

En algunas realizaciones, cada uno de los uno o más procesadores 2602 incluye uno o más núcleos de procesador 2607 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 2607 está configurado para procesar un conjunto de instrucciones 2609 específico. En algunas realizaciones, el conjunto de instrucciones 2609 puede facilitar la computación de conjunto de instrucciones complejas (CISC), la computación de conjunto de instrucciones reducidas (RISC) o la computación mediante una palabra de instrucción muy larga (VLIW). Múltiples núcleos de procesador 2607 pueden procesar, cada uno, un conjunto de instrucciones 2609 diferente, que puede incluir instrucciones para facilitar la emulación de otros conjuntos de instrucciones. El núcleo de procesador 2607 también puede incluir otros dispositivos de procesamiento, tales como un procesador de señal digital (DSP).

En algunas realizaciones, el procesador 2602 incluye la memoria caché 2604. Dependiendo de la arquitectura, el procesador 2602 puede tener una única memoria caché interna o múltiples niveles de caché interna. En algunas

realizaciones, la memoria caché se comparte entre varios componentes del procesador 2602. En algunas realizaciones, el procesador 2602 también usa una caché externa (por ejemplo, una caché de nivel 3 (L3) o una caché de último nivel (LLC)) (no mostrada), que puede compartirse entre los núcleos de procesador 2607 usando técnicas de coherencia de caché conocidas. Se incluye adicionalmente un archivo de registro 2606 en el procesador 2602 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 2602.

En algunas realizaciones, el procesador 2602 está acoplado con un bus de procesador 2610 para transmitir señales de comunicación tales como dirección, datos o señales de control entre el procesador 2602 y otros componentes en el sistema 2600. En una realización, el sistema 2600 usa una arquitectura de sistema de 'concentrador' ilustrativa, incluyendo un concentrador de controlador de memoria 2616 y un concentrador de controlador de entrada-salida (E/S) 2630. Un concentrador de controlador de memoria 2616 facilita la comunicación entre un dispositivo de memoria y otros componentes del sistema 2600, mientras que un concentrador de controlador de E/S (ICH) 2630 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 2616 está integrada dentro del procesador.

El dispositivo de memoria 2620 puede ser un dispositivo de memoria de acceso aleatorio dinámica (DRAM), un dispositivo de memoria de acceso aleatorio estática (SRAM), dispositivo de memoria flash, dispositivo de memoria de cambio de fase o algún otro dispositivo de memoria que tiene un rendimiento adecuado para servir como una memoria de proceso. En una realización, el dispositivo de memoria 2620 puede funcionar como memoria de sistema para el sistema 2600, para almacenar datos 2622 e instrucciones 2621 para su uso cuando uno o más procesadores 2602 ejecutan una aplicación o proceso. El concentrador de controlador de memoria 2616 también se acopla con un procesador de gráficos externo opcional 2612, que puede comunicarse con el uno o más procesadores de gráficos 2608 en los procesadores 2602 para realizar operaciones de gráficos y de medios.

En algunas realizaciones, el ICH 2630 permite que los periféricos se conecten al dispositivo de memoria 2620 y al procesador 2602 a través de un bus de E/S de alta velocidad. Los periféricos de E/S incluyen, entre otros, un controlador de audio 2646, una interfaz de firmware 2628, un transceptor inalámbrico 2626 (por ejemplo, Wi-Fi, Bluetooth), un dispositivo de almacenamiento de datos 2624 (por ejemplo, unidad de disco duro, memoria flash, etc.), y un controlador de E/S 2640 heredado para acoplar dispositivos heredados (por ejemplo, Personal System 2 (PS/2)) al sistema. Uno o más controladores de bus serie universal (USB) 2642 conectan dispositivos de entrada, tales como las combinaciones de teclado y ratón 2644. Un controlador de red 2634 también puede acoplarse con el ICH 2630. En algunas realizaciones, un controlador de red de alto rendimiento (no mostrado) se acopla con el bus de procesador 2610. Se apreciará que el sistema 2600 mostrado es ilustrativo y no limitante, debido a que también pueden usarse otros tipos de sistemas de procesamiento de datos que están configurados de manera diferente. Por ejemplo, el concentrador de controlador de E/S 2630 puede integrarse dentro de los uno o más procesadores 2602, o el concentrador de controlador de memoria 2616 y el concentrador de controlador de E/S 2630 pueden integrarse en un procesador de gráficos externo discreto, tal como el procesador de gráficos externo 2612.

La **Figura 27** es un diagrama de bloques de una realización de un procesador 2700 que tiene uno o más núcleos de procesador 2702A-2702N, un controlador de memoria integrado 2714 y un procesador de gráficos integrado 2708. Aquellos elementos de la **Figura 27** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto. El procesador 2700 puede incluir núcleos adicionales hasta e incluyendo el núcleo adicional 2702N representado por los recuadros con línea discontinua. Cada uno de los núcleos de procesador 2702A-2702N incluye una o más unidades de caché internas 2704A-2704N. En algunas realizaciones, cada núcleo de procesador también tiene acceso a una o más unidades de caché compartidas 2706.

Las unidades de caché internas 2704A-2704N y las unidades de caché compartidas 2706 representan una jerarquía de memoria caché dentro del procesador 2700. La jerarquía de memoria caché puede incluir al menos un nivel de caché de instrucciones y datos dentro de cada núcleo de procesador y uno o más niveles de caché de nivel intermedio compartida, tal como nivel 2 (L2), nivel 3 (L3), nivel 4 (L4) u otros niveles de caché, 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 2706 y 2704A-2704N.

En algunas realizaciones, el procesador 2700 también puede incluir un conjunto de una o más unidades de controlador de bus 2716 y un núcleo de agente de sistema 2710. La una o más unidades de controlador de bus 2716 gestionan un conjunto de buses periféricos, tales como uno o más buses de interconexión de componentes periféricos (por ejemplo, PCI, PCI Express). El núcleo de agente de sistema 2710 proporciona funcionalidad de gestión para los diversos componentes de procesador. En algunas realizaciones, el núcleo de agente de sistema 2710 incluye uno o más controladores de memoria integrados 2714 para gestionar el acceso a varios dispositivos de memoria externos (no mostrados).

En algunas realizaciones, uno o más de los núcleos de procesador 2702A-2702N incluyen soporte para múltiples hilos simultáneos. En una realización de este tipo, el núcleo de agente de sistema 2710 incluye componentes para coordinar y operar los núcleos 2702A-2702N durante el procesamiento de múltiples hilos. El núcleo de agente de sistema 2710 puede incluir adicionalmente una unidad de control de energía (PCU), que incluye lógica y componentes para regular el estado de energía de los núcleos de procesador 2702A-2702N y el procesador de gráficos 2708.

En algunas realizaciones, el procesador 2700 incluye adicionalmente el procesador de gráficos 2708 para ejecutar las operaciones de procesamiento de gráficos. En algunas realizaciones, el procesador de gráficos 2708 se acopla con el conjunto de unidades de caché compartida 2706 y el núcleo de agente de sistema 2710, incluidos el uno o más controladores de memoria integrados 2714. En algunas realizaciones, un controlador de visualización 2711 está acoplado con el procesador de gráficos 2708 para controlar la salida del procesador de gráficos a una o más pantallas acopladas. En algunas realizaciones, el controlador de visualización 2711 puede ser un módulo separado acoplado con el procesador de gráficos a través de al menos una interconexión, o puede estar integrado dentro del procesador de gráficos 2708 o el núcleo de agente de sistema 2710.

En algunas realizaciones, se usa una unidad de interconexión basada en anillo 2712 para acoplar los componentes internos del procesador 2700. Sin embargo, se puede usar una unidad de interconexión alternativa, tal como una interconexión punto a punto, una interconexión conmutada u otras técnicas, incluyendo técnicas bien conocidas en la técnica. En algunas realizaciones, el procesador de gráficos 2708 se acopla con la interconexión en anillo 2712 mediante un enlace de E/S 2713.

El enlace de E/S 2713 ilustrativo representa al menos una de múltiples variedades de interconexiones de E/S, incluyendo 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 2718, tal como un módulo de eDRAM. En algunas realizaciones, cada uno de los núcleos de procesador 2702A-2702N y el procesador de gráficos 2708 usan módulos de memoria integrados 2718 como una caché de último nivel compartida.

En algunas realizaciones, los núcleos de procesador 2702A-2702N son núcleos homogéneos que ejecutan la misma arquitectura de conjunto de instrucciones. En otra realización, los núcleos de procesador 2702A-2702N son heterogéneos en términos de arquitectura de conjunto de instrucciones (ISA), donde uno o más de los núcleos de procesador 2702A-2702N 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 2702A-2702N son heterogéneos en términos de microarquitectura, 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 2700 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 28** es un diagrama de bloques de un procesador de gráficos 2800, 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 mediante una interfaz de E/S de memoria asignada a registros en el procesador de gráficos y con comandos colocados en la memoria de procesador. En algunas realizaciones, el procesador de gráficos 2800 incluye una interfaz de memoria 2814 para acceder a memoria. La interfaz de memoria 2814 puede ser una interfaz a una memoria local, una o más cachés internas, una o más cachés externas compartidas y/o a una memoria de sistema.

En algunas realizaciones, el procesador de gráficos 2800 también incluye un controlador de visualización 2802 para enviar datos de salida de visualización a un dispositivo de visualización 2820. El controlador de visualización 2802 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 2800 incluye un motor de códec de vídeo 2806 para codificar, decodificar 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 2800 incluye un motor de transferencia de imágenes en bloque (BLIT) 2804 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) 2810. En algunas realizaciones, el GPE 2810 es un motor de cálculo para realizar operaciones de gráficos, incluyendo operaciones de gráficos tridimensionales (3D) y operaciones de medios.

En algunas realizaciones, el GPE 310 incluye una canalización 3D 2812 para llevar a cabo operaciones 3D, como por ejemplo representar imágenes y escenas tridimensionales utilizando funciones de procesamiento que actúan sobre formas de primitivas 3D (por ejemplo, rectángulo, triángulo, etc.). La canalización 3D 2812 incluye elementos de

función programable y fija que llevan a cabo diversas tareas dentro del elemento y/o generan hilos de ejecución en un subsistema de 3D/medios 2815. Aunque la canalización 3D 2812 se puede utilizar para llevar a cabo operaciones de medios, una realización del GPE 2810 también incluye una canalización de medios 2816 que se utiliza específicamente para llevar a cabo operaciones de medios, como por ejemplo postprocesamiento de vídeo y mejora de imagen.

En algunas realizaciones, la canalización de medios 2816 incluye unidades lógicas programables o de función fija para llevar a cabo una o más operaciones de medios especializadas, como por ejemplo aceleración de descodificación de vídeo, desentrelazado de vídeo y aceleración de codificación de vídeo en lugar de, o en nombre del motor de códec de vídeo 2806. En algunas realizaciones, la canalización de medios 2816 incluye adicionalmente una unidad de generación de hilos para generar hilos para su ejecución en el subsistema de 3D/medios 2815. 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/medios 2815.

En algunas realizaciones, el subsistema de 3D/medios 2815 incluye una lógica para ejecutar hilos generados por la canalización 3D 2812 y la canalización de medios 2816. En una realización, las canalizaciones envían solicitudes de ejecución de hilos al subsistema de 3D/medios 2815, que incluye lógica de envío de hilos para arbitrar y enviar las diversas solicitudes a los recursos de ejecución de hilos disponibles. Los recursos de ejecución incluyen una matriz de unidades de ejecución de gráficos para procesar los hilos de medios y 3D. En algunas realizaciones, el subsistema de 3D/medios 2815 incluye una o más cachés internas para instrucciones y datos de hilos. En algunas realizaciones, el subsistema también incluye memoria compartida, incluyendo registros y memoria direccionable, para compartir datos entre hilos y para almacenar datos de salida.

Motor de procesamiento de gráficos adicional ilustrativo

La **Figura 29** es un diagrama de bloques de un motor de procesamiento de gráficos 2910 de un procesador de gráficos de acuerdo con algunas realizaciones. En una realización, el motor de procesamiento de gráficos (GPE) 2910 es una versión del GPE 2810 mostrado en la **Figura 28**. Los elementos de la **Figura 29** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto. Por ejemplo, se ilustra la canalización 3D 2812 y la canalización de medios 2816 de la **Figura 28**. La canalización de medios 2816 es opcional en algunas realizaciones del GPE 2910 y puede no incluirse explícitamente dentro del GPE 2910. Por ejemplo, y en al menos una realización, un procesador de medios y/o de imágenes separado se acopla al GPE 2910.

En algunas realizaciones, el GPE 2910 se acopla con o incluye un transmisor por flujo continuo de comandos 2903, que proporciona un flujo de comandos a la canalización 3D 2812 y/o a las canalizaciones de medios 2816. En algunas realizaciones, el transmisor por flujo continuo de comandos 2903 está acoplado con memoria, que puede ser memoria de sistema, o una o más de memoria de caché interna y memoria de caché compartida. En algunas realizaciones, el transmisor por flujo continuo de comandos 2903 recibe comandos desde la memoria y envía los comandos a la canalización 3D 2812 y/o a la canalización de medios 2816. Los comandos son directivas extraídas de una memoria intermedia en anillo, que almacena comandos para la canalización 3D 2812 y la canalización de medios 2816. En una realización, la memoria intermedia en anillo puede incluir adicionalmente memorias intermedias de comandos por lotes que almacenan lotes de múltiples comandos. Los comandos para la canalización 3D 2812 también pueden incluir referencias a datos almacenados en memoria, tales como, pero sin limitación, datos de vértice y de geometría para la canalización 3D 2812 y/o datos de imagen y objetos de memoria para la canalización de medios 2816. La canalización 3D 2812 y la canalización de medios 2816 procesan los comandos y datos realizando operaciones mediante una lógica dentro de las canalizaciones respectivas o despachando uno o más hilos de ejecución a una matriz de núcleo de gráficos 2914.

En diversas realizaciones, la canalización 3D 2812 puede ejecutar uno o más programas sombreadores, tales como sombreadores de vértices, sombreadores de geometría, sombreadores de píxeles, sombreadores de fragmentos, sombreadores de cálculo u otros programas sombreadores, procesando las instrucciones y despachando hilos de ejecución a la matriz de núcleo de gráficos 2914. La matriz de núcleo de gráficos 2914 proporciona un bloque unificado de recursos de ejecución. La lógica de ejecución multipropósito (por ejemplo, unidades de ejecución) dentro de la matriz de núcleo de gráficos 2914 incluye soporte para varios lenguajes de sombreador API 3D y puede ejecutar múltiples hilos de ejecución simultáneos asociados con múltiples sombreadores.

En algunas realizaciones, la matriz de núcleo de gráficos 2914 también incluye una 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 una lógica de propósito general que es programable para realizar operaciones computacionales de propósito general paralelas, además de operaciones de procesamiento de gráficos. La lógica de propósito general puede realizar operaciones de procesamiento en paralelo o en conjunto con la lógica de propósito general dentro del núcleo o núcleos de procesador 2607 de la **Figura 26** o el núcleo 2702A-2702N como en la **Figura 27**.

Los datos de salida generados por hilos que se ejecutan en la matriz de núcleo de gráficos 2914 pueden emitir datos a memoria en una memoria intermedia de retorno unificada (URB) 2918. La URB 2918 puede almacenar datos para múltiples hilos. En algunas realizaciones, la URB 2918 puede usarse para enviar datos entre diferentes hilos que se ejecutan en la matriz de núcleo de gráficos 2914. En algunas realizaciones, la URB 2918 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 2920.

En algunas realizaciones, la matriz de núcleo de gráficos 2914 es ajustable a escala, de modo 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 2910. En una realización, los recursos de ejecución son dinámicamente ajustables a escala, de manera que los recursos de ejecución pueden habilitarse o deshabilitarse según sea necesario.

La matriz de núcleo de gráficos 2914 se acopla con la lógica de funciones compartidas 2920 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 funciones compartidas 2920 son unidades de lógica de hardware que proporcionan una funcionalidad complementaria especializada a la matriz de núcleo de gráficos 2914. En diversas realizaciones, la lógica de funciones compartidas 2920 incluye, pero sin limitación, la lógica del muestreador 2921, del cómputo matemático 2922 y de la comunicación entre hilos (ITC) 2923. Adicionalmente, algunas realizaciones implementan una o más cachés 2925 dentro de la lógica de funciones compartidas 2920. 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 2914. En su lugar, una única instanciación de esa función especializada se implementa como una entidad autónoma en la lógica de funciones compartidas 2920 y se comparte entre los recursos de ejecución dentro de la matriz de núcleo de gráficos 2914. El conjunto preciso de funciones que se comparten entre la matriz de núcleo de gráficos 2914 y se incluyen dentro de la matriz de núcleo de gráficos 2914 varía entre realizaciones.

La **Figura 30** es un diagrama de bloques de otra realización de un procesador de gráficos 3000. Los elementos de la **Figura 30** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto.

En algunas realizaciones, el procesador de gráficos 3000 incluye una interconexión en anillo 3002, un extremo frontal de canalización 3004, un motor de medios 3037 y núcleos de gráficos 3080A-3080N. En algunas realizaciones, la interconexión en anillo 3002 acopla el procesador de gráficos a otras unidades de procesamiento, que incluyen otros procesadores de gráficos o uno o más núcleos de procesadores 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 3000 recibe lotes de comandos mediante la interconexión en anillo 3002. Los comandos entrantes son interpretados por un transmisor por flujo continuo de comandos 3003 en el extremo frontal de canalización 3004. En algunas realizaciones, el procesador de gráficos 3000 incluye una lógica de ejecución escalable para realizar procesamiento de geometría 3D y procesamiento de medios mediante el núcleo o núcleos de gráficos 3080A-3080N. Para los comandos de procesamiento de geometría 3D, el transmisor por flujo continuo de comandos 3003 suministra comandos a la canalización de geometría 3036. Para al menos algunos comandos de procesamiento de medios, el transmisor por flujo continuo de comandos 3003 suministra los comandos a un extremo frontal de video 3034, que se acopla con un motor de medios 3037. En algunas realizaciones, el motor de medios 3037 incluye un motor de calidad de video (VQE) 3030 para postprocesamiento de video y de imagen y un motor de codificación/decodificación de múltiples formatos (MFX) 3033 para proporcionar codificación y decodificación de datos de medios acelerados por hardware. En algunas realizaciones, la canalización de geometría 3036 y el motor de medios 3037 generan, cada uno, hilos de ejecución para los recursos de ejecución de hilos proporcionados por al menos un núcleo de gráficos 3080A.

En algunas realizaciones, el procesador de gráficos 3000 incluye recursos de ejecución de hilos ajustables a escala que cuentan con los núcleos modulares 3080A-3080N (denominados, en ocasiones, segmentos de núcleo), teniendo cada uno múltiples subnúcleos 3050A-550N, 3060A-3060N (denominados, en ocasiones, subsegmentos de núcleo). En algunas realizaciones, el procesador de gráficos 3000 puede tener cualquier número de núcleos de gráficos 3080A a 3080N. En algunas realizaciones, el procesador de gráficos 3000 incluye un núcleo de gráficos 3080A que tiene al menos un primer subnúcleo 3050A y un segundo subnúcleo 3060A. En otras realizaciones, el procesador de gráficos es un procesador de baja potencia con un único subnúcleo (por ejemplo, 3050A). En algunas realizaciones, el procesador de gráficos 3000 incluye múltiples núcleos de gráficos 3080A-3080N, incluyendo cada uno un conjunto de primeros subnúcleos 3050A-3050N y un conjunto de segundos subnúcleos 3060A-3060N. Cada subnúcleo del conjunto de primeros subnúcleos 3050A-3050N incluye al menos un primer conjunto de unidades de ejecución 3052A-3052N y muestreadores de medios/texturas 3054A-3054N. Cada subnúcleo del conjunto de segundos subnúcleos 3060A-3060N incluye al menos un segundo conjunto de unidades de ejecución 3062A-3062N y muestreadores 3064A-3064N. En algunas realizaciones, cada subnúcleo 3050A-3050N, 3060A-3060N comparte un conjunto de recursos compartidos 3070A-3070N. En algunas realizaciones, los recursos compartidos incluyen memoria caché compartida

y lógica de operación de píxeles. Se pueden incluir también otros recursos compartidos en las diversas realizaciones del procesador de gráficos.

Unidades de ejecución adicionales ilustrativas

La **Figura 31** ilustra lógica de ejecución de hilos 3100 que incluye una matriz de elementos de procesamiento empleados en algunas realizaciones de un GPE. Los elementos de la **Figura 31** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto.

En algunas realizaciones, la lógica de ejecución de hilos 3100 incluye un procesador sombreador 3102, un despachador de hilos 3104, una caché de instrucciones 3106, una matriz de unidad de ejecución escalable que incluye una pluralidad de unidades de ejecución 3108A-3108N, un muestreador 3110, una caché de datos 3112 y un puerto de datos 3114. En una realización, la unidad de ejecución escalable puede escalar dinámicamente activando o desactivando una o más unidades de ejecución (por ejemplo, cualquiera de la unidad de ejecución 3108A, 3108B, 3108C, 3108D, a 3108N-1 y 3108N) basándose en los requisitos de cálculo de una carga de trabajo. En una realización, los componentes incluidos están interconectados mediante una estructura de interconexión que se enlaza a cada uno de los componentes. En algunas realizaciones, la lógica de ejecución de hilos 3100 incluye una o más conexiones a memoria, tal como la memoria de sistema o memoria caché, a través de una o más de la caché de instrucciones 3106, el puerto de datos 3114, el muestreador 3110 y las unidades de ejecución 3108A-3108N. En algunas realizaciones, cada unidad de ejecución (por ejemplo, 3108A) es una unidad computacional de propósito general programable autónoma que es capaz de ejecutar múltiples hilos de hardware simultáneos mientras se procesan múltiples elementos de datos en paralelo para cada hilo. En diversas realizaciones, la matriz de unidades de ejecución 3108A-3108N es ajustable a escala para incluir cualquier número de unidades de ejecución individuales.

En algunas realizaciones, las unidades de ejecución 3108A-3108N se usan principalmente para ejecutar programas sombreadores. Un procesador sombreador 3102 puede procesar los diversos programas sombreadores y enviar hilos de ejecución asociados con los programas sombreadores por medio de un despachador de hilos 3104. En una realización, el despachador de hilos incluye lógica para arbitrar las solicitudes de iniciación de hilo desde las canalizaciones de gráficos y de medios e instanciar los hilos solicitados en una o más unidades de ejecución en las unidades de ejecución 3108A-3108N. Por ejemplo, la canalización de geometría (por ejemplo, 3036 de la **Figura 30**) puede despachar los sombreadores de vértices, de teselación o de geometría a la lógica de ejecución de hilos 3100 (**Figura 31**) para su procesamiento. En algunas realizaciones, el despachador de hilos 3104 también puede procesar solicitudes de generación de hilos en tiempo de ejecución desde los programas sombreadores en ejecución.

En algunas realizaciones, las unidades de ejecución 3108A-3108N soportan un conjunto de instrucciones que incluye soporte nativo para muchas instrucciones de sombreador de gráficos 3D estándar, de tal forma que los programas sombreadores de las 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 cálculo y de medios). Cada una de las unidades de ejecución 3108A-3108N es capaz de múltiples emisiones de ejecución de múltiples datos de instrucción única (SIMD), y una operación de múltiples hilos posibilita 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 alto ancho de banda dedicado y un estado de hilo independiente asociado. La ejecución es de múltiples emisiones por reloj a canalizaciones que pueden realizar operaciones de números enteros, de coma flotante de precisión sencilla y doble, de capacidad de ramificación de SIMD, operaciones lógicas, operaciones trascendentales y otras operaciones misceláneas. Mientras se esperan datos desde la memoria o una de las funciones compartidas, una lógica de dependencia dentro de las unidades de ejecución 3108A-3108N 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, pueden dedicarse 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 sombreador, incluyendo un sombreador de vértices diferente.

Cada unidad de ejecución en las unidades de ejecución 3108A-3108N 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 3108A-3108N soportan tipos de datos de números enteros y de coma flotante.

El conjunto de instrucciones de la unidad de ejecución incluye instrucciones de SIMD. Los diversos elementos de datos se pueden almacenar como un tipo de datos empaquetados en un registro y la unidad de ejecución procesará los diversos elementos basándose en el tamaño de datos de los elementos. Por ejemplo, cuando se opera sobre un vector de 256 bits de ancho, los 256 bits del vector se almacenan en un registro y la unidad de ejecución opera sobre

el vector como cuatro elementos de datos empaquetados de 64 bits separados (elementos de datos de tamaño de palabra cuádruple (QW)), ocho elementos de datos empaquetados de 32 bits separados (elementos de datos de tamaño de palabra doble (DW)), dieciséis elementos de datos empaquetados de 16 bits separados (elementos de datos de tamaño de palabra (W)) o treinta y dos elementos de datos de 8 bits separados (elementos de datos de tamaño de byte (B)). Sin embargo, son posibles diferentes anchuras de vector y tamaños de registro.

Una o más cachés de instrucciones internas (por ejemplo, 3106) están incluidas en la lógica de ejecución de hilos 3100 a las instrucciones de hilo de caché para las unidades de ejecución. En algunas realizaciones, se incluyen una o más cachés de datos (por ejemplo, 3112) para almacenar en caché los datos del hilo durante la ejecución del hilo. En algunas realizaciones, se incluye un muestreador 3110 para proporcionar un muestreo de textura para operaciones 3D y muestreo de medios para operaciones de medios. En algunas realizaciones, el muestreador 3110 incluye funcionalidad especializada de muestreo de textura o de medios para procesar datos de textura o de medios durante el proceso de muestreo antes de proporcionar los datos muestreados a una unidad de ejecución.

Durante la ejecución, las canalizaciones de gráficos y de medios envían solicitudes de iniciación de hilo a la lógica de ejecución de hilos 3100 por medio de una lógica de generación y de despacho de hilos. Una vez que se ha procesado y rasterizado un grupo de objetos geométricos para obtener datos de píxel, se invoca una lógica de procesador de píxeles (por ejemplo, lógica de sombreador de píxeles, lógica de sombreador de fragmentos, etc.) dentro del procesador sombreador 3102 para calcular adicionalmente información de salida y hacer que se escriban resultados para emitir superficies (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 sombreador 3102 ejecuta a continuación un programa sombreador de píxeles o de fragmentos suministrado por una interfaz de programación de aplicaciones (API). Para ejecutar el programa sombreador, el procesador sombreador 3102 despacha hilos a una unidad de ejecución (por ejemplo, 3108A) por medio del despachador de hilos 3104. En algunas realizaciones, el sombreador de píxeles 3102 utiliza una lógica de muestreo de textura en el muestreador 3110 para acceder a datos de textura en mapas de textura almacenadas en memoria. Unas operaciones aritméticas sobre los datos de textura y los datos de geometría de entrada calculan 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 3114 proporciona un mecanismo de acceso de memoria para que la lógica de ejecución de hilos 3100 emita datos procesados a la memoria para su procesamiento en una canalización de salida del procesador de gráficos. En algunas realizaciones, el puerto de datos 3114 incluye o se acopla a una o más memorias caché (por ejemplo, la caché de datos 3112) para almacenar en caché datos para un acceso de memoria por medio del puerto de datos.

La **Figura 32** es un diagrama de bloques que ilustra unos formatos de instrucción de procesador de gráficos 3200 de acuerdo con algunas realizaciones. En una o más realizaciones, las unidades de ejecución del procesador de gráficos soportan un conjunto de instrucciones que tiene instrucciones en múltiples formatos. Los recuadros de líneas continuas ilustran los componentes que se incluyen generalmente en una instrucción de unidad de ejecución, mientras que las líneas discontinuas incluyen componentes que son opcionales o que solo se incluyen en un subconjunto de las instrucciones. En algunas realizaciones, el formato de instrucción 3200 descrito e ilustrado son macroinstrucciones, en el sentido de que las mismas son instrucciones suministradas a la unidad de ejecución, en contraposición a microoperaciones resultantes de la decodificación de instrucciones una vez que se ha procesado la instrucción.

En algunas realizaciones, las unidades de ejecución del procesador de gráficos soportan de manera nativa instrucciones en un formato de instrucción de 128 bits 3210. Un formato de instrucción compacta de 64 bits 3230 está disponible para algunas instrucciones en función de 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 64 bits 3230. Las instrucciones nativas disponibles en el formato de 64 bits 3230 varían según la realización. En algunas realizaciones, la instrucción se compacta en parte usando un conjunto de valores de índice en un campo de índice 3213. El hardware de la unidad de ejecución hace referencia a un conjunto de tablas de compactación basándose en los valores de índice y usa las salidas de tabla de compactación para reconstruir una instrucción nativa en el formato de instrucción de 128 bits 3210.

Para cada formato, el codope de instrucción 3212 define la operación que ha de llevar a cabo 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 adición, la unidad de ejecución realiza una operación de adición simultánea en cada canal de color que representa un elemento de textura o elemento de imagen. Por defecto, la unidad de ejecución realiza cada instrucción a través de todos los canales de datos de los operandos. En algunas realizaciones, el campo de control de instrucciones 3214 permite el control sobre ciertas opciones de ejecución, tales como la selección de canales (por ejemplo, predicación) y el orden de los canales de datos (por ejemplo, swizzle). Para instrucciones en el formato de instrucción de 128 bits 3210, un campo de tamaño de ejecución

3216 limita el número de canales de datos que se ejecutarán en paralelo. En algunas realizaciones, el campo de tamaño de ejecución 3216 no está disponible para su uso en el formato de instrucción compacta de 64 bits 3230.

Algunas instrucciones de la unidad de ejecución tienen hasta tres operandos, incluyendo dos operandos de origen, src0 3220, src1 3222 y un destino 3218. En algunas realizaciones, las unidades de ejecución soportan instrucciones de destino dual, donde uno de los destinos está implícito. Las instrucciones de manipulación de datos pueden tener un tercer operando de origen (por ejemplo, SRC2 3224), donde el código de instrucción 3212 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 fija) pasado con la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 3210 incluye un campo de modo de acceso/dirección 3226 que especifica, por ejemplo, si se utiliza el modo de direccionamiento de registro directo o el modo de direccionamiento de registro indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, la dirección de registro de uno o más operandos es proporcionada directamente por bits en la instrucción.

En algunas realizaciones, el formato de instrucción de 128 bits 3210 incluye un campo de modo de dirección/acceso 3226, que especifica un modo de dirección y/o un modo de acceso para la instrucción. En una realización, el modo de acceso se utiliza 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, 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 utilizar un direccionamiento alineado por byte para los operandos de origen y de destino y, cuando está en un segundo modo, la instrucción puede utilizar un direccionamiento alineado de 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 3226 determina si la instrucción va a utilizar direccionamiento directo o indirecto. Cuando se utiliza el modo de direccionamiento de registro directo, unos bits de la instrucción proporcionan directamente la dirección de registro de uno o más operandos. Cuando se utiliza un modo de direccionamiento de registro indirecto, la dirección de registro de uno o más operandos se puede calcular en función de un valor de registro de dirección y un campo inmediato de dirección en la instrucción.

En algunas realizaciones, las instrucciones se agrupan en función de los campos de bits del código 3212 para simplificar la decodificación de código 3240. Para un código de 8 bits, los bits 4, 5 y 6 permiten que la unidad de ejecución determine el tipo de código. La agrupación de código precisa mostrada es simplemente un ejemplo. En algunas realizaciones, un grupo de código de movimiento y lógica 3242 incluye instrucciones de movimiento y de lógica de datos (p. ej., mover (mov), comparar (cmp)). En algunas realizaciones, el grupo de movimiento y lógica 3242 comparte los cinco bits más significativos (MSB), donde las instrucciones mover (mov) están en forma de 0000xxxxb y las instrucciones de lógica están en forma de 0001xxxxb. Un grupo de instrucciones de control de flujo 3244 (por ejemplo, llamada, salto (jmp)) incluye instrucciones en forma de 0010xxxxb (por ejemplo, 0x20). Un grupo de instrucciones misceláneas 3246 incluye una mezcla de instrucciones, incluidas instrucciones de sincronización (por ejemplo, esperar, enviar) en forma de 0011xxxxb (por ejemplo, 0x30). Un grupo de instrucciones de cómputo matemático paralelo 3248 incluye instrucciones aritméticas a nivel de componente (por ejemplo, suma, multiplicación (mul)) en forma de 0100xxxxb (por ejemplo, 0x40). El grupo de cómputo matemático paralelo 3248 realiza las operaciones aritméticas en paralelo a través de canales de datos. El grupo de cómputo matemático vectorial 3250 incluye instrucciones aritméticas (por ejemplo, dp4) en forma de 0101xxxxb (por ejemplo, 0x50). El grupo de cómputo matemático vectorial realiza aritmética tal como cálculos de producto escalar sobre operandos de vectores.

Canalización de gráficos adicionales ilustrativa

La **Figura 33** es un diagrama de bloques de otra realización de un procesador de gráficos 3300. Los elementos de la **Figura 33** que tienen los mismos números (o nombres) de referencia que los elementos de cualquier otra figura en el presente documento pueden operar o funcionar de cualquier manera similar a la descrita en cualquier otra parte en el presente documento, pero sin limitación a esto.

En algunas realizaciones, el procesador de gráficos 3300 incluye una canalización de gráficos 3320, una canalización de medios 3330, un motor de visualización 3340, una lógica de ejecución de hilos 3350 y una canalización de salida de representación 3370. En algunas realizaciones, el procesador de gráficos 3300 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 es controlado por escrituras de registro en uno o más registros de control (no mostrados) o mediante comandos emitidos al procesador de gráficos 3300 mediante una interconexión en anillo 3302. En algunas realizaciones, la interconexión en anillo 3302 acopla el procesador de gráficos 3300 a otros componentes de procesamiento, tales como otros procesadores de gráficos o procesadores de propósito general. Los comandos desde la interconexión en anillo 3302 son interpretados por un transmisor por flujo continuo de comandos 3303, que suministra instrucciones a componentes individuales de la canalización de gráficos 3320 o la canalización de medios 3330.

- En algunas realizaciones, el transmisor por flujo continuo de comandos 3303 dirige la operación de un extractor de vértices 3305 que lee datos de vértice desde memoria y ejecuta comandos de procesamiento de vértices proporcionados por el transmisor por flujo continuo de comandos 3303. En algunas realizaciones, el extractor de vértices 3305 proporciona datos de vértice a un sombreador de vértices 3307, que realiza operaciones de transformación y de iluminación de espacio de coordenadas en cada vértice. En algunas realizaciones, el extractor de vértices 3305 y el sombreador de vértices 3307 ejecutan instrucciones de procesamiento de vértices despachando hilos de ejecución a las unidades de ejecución 3352A-3352B mediante un despachador de hilos 3331.
- En algunas realizaciones, las unidades de ejecución 3352A-3352B 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 3352A-3352B tienen una caché L1 3351 adjunta que es específica para cada matriz o que se comparte entre las matrices. La caché se puede configurar como una caché de datos, una caché de instrucciones o una única caché que se subdivide para contener datos e instrucciones en diferentes subdivisiones.
- En algunas realizaciones, la canalización de gráficos 3320 incluye componentes de teselación para realizar teselación acelerada por hardware de objetos 3D. En algunas realizaciones, un sombreador de casco programable 811 configura las operaciones de teselación. Un sombreador de dominio programable 817 proporciona una evaluación de extremo posterior del resultado de la teselación. Un teselador 3313 opera en la dirección del sombreador de casco 3311 y contiene una lógica de propósito especial para generar un conjunto de objetos geométricos detallados en función de un modelo geométrico en bruto que se proporciona como entrada a la canalización de gráficos 3320. En algunas realizaciones, si no se usa la teselación, pueden eludirse los componentes de teselación (por ejemplo, el sombreador de casco 3311, el teselador 3313 y el sombreador de dominio 3317).
- En algunas realizaciones, unos objetos geométricos completos pueden ser procesados por un sombreador de geometría 3319 mediante uno o más hilos despachados a las unidades de ejecución 3352A-3352B, o puede avanzar directamente al recortador 3329. En algunas realizaciones, el sombreador de geometría opera sobre objetos geométricos enteros, en lugar de vértices o parches de vértices como en fases previas de la canalización de gráficos. Si la teselación está deshabilitada, el sombreador de geometría 3319 recibe una entrada desde el sombreador de vértices 3307. En algunas realizaciones, el sombreador de geometría 3319 se puede programar mediante un programa sombreador de geometría para realizar una teselación de geometría si las unidades de teselación están deshabilitadas.
- Antes de la rasterización, un recortador 3329 procesa datos de vértice. El recortador 3329 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 rasterización y prueba de profundidad 3373 en el canal de salida de representación 3370 envía sombreadores de píxeles para convertir los objetos geométricos en sus representaciones por píxel. En algunas realizaciones, la lógica de sombreador de píxeles se incluye en la lógica de ejecución de hilos 3350. En algunas realizaciones, una aplicación puede omitir el componente de rasterización y prueba de profundidad 3373 y acceder a datos de vértice sin rasterizar mediante una unidad de salida de flujo 3323.
- El procesador de gráficos 3300 tiene un bus de interconexión, una estructura 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 3352A-3352B y la caché o cachés 3351 asociadas, el muestreador de textura y de medios 3354 y la caché de textura/muestreador 3358 se interconectan mediante un puerto de datos 3356 para realizar un acceso de memoria y comunicarse con componentes de canalización de salida de representación del procesador. En algunas realizaciones, cada uno del muestreador 3354, las cachés 3351, 3358 y las unidades de ejecución 3352A-3352B tiene rutas de acceso de memoria separadas.
- En algunas realizaciones, la canalización de salida de representación 3370 contiene un componente de rasterización y prueba de profundidad 3373 que convierte objetos basados en vértices en una representación asociada basada en píxeles. 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 3378 y una caché de profundidad 3379 asociadas también están disponibles en algunas realizaciones. Un componente de operaciones de píxel 3377 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 3341, o son sustituidas en el momento de la visualización por el controlador de visualización 3343 usando planos de visualización de superposición. En algunas realizaciones, está disponible una caché L3 compartida 3375 para todos los componentes de gráficos, permitiendo compartir datos sin el uso de memoria de sistema principal.
- En algunas realizaciones, la canalización de medios del procesador de gráficos 3330 incluye un motor de medios 3337 y un extremo frontal de vídeo 3334. En algunas realizaciones, el extremo frontal de vídeo 3334 recibe comandos de canalización desde el transmisor por flujo continuo de comandos 3303. En algunas realizaciones, la canalización de medios 3330 incluye un transmisor por flujo continuo de comandos separado. En algunas realizaciones, el extremo frontal de vídeo 3334 procesa comandos de medios antes de enviar el comando al motor de medios 3337. En algunas realizaciones, el motor de medios 3337 incluye una funcionalidad de generación de hilos para generar hilos para su envío a la lógica de ejecución de hilos 3350 a través del despachador de hilos 3331.

En algunas realizaciones, el procesador de gráficos 3300 incluye un motor de visualización 3340. En algunas realizaciones, el motor de visualización 3340 es externo al procesador 3300 y se acopla con el procesador de gráficos a través de la interconexión en anillo 3302, o algún otro bus o estructura de interconexión. En algunas realizaciones, el motor de visualización 3340 incluye un motor 2D 3341 y un controlador de visualización 3343. En algunas realizaciones, el motor de visualización 3340 contiene una lógica de propósito especial que puede operar independientemente de la canalización 3D. En algunas realizaciones, el controlador de visualización 3343 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 mediante un conector de dispositivo de visualización.

En algunas realizaciones, la canalización de gráficos 3320 y la canalización de medios 3330 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, el 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 cálculo Vulkan, todas ellas del grupo Khronos. En algunas realizaciones, también se puede proporcionar soporte para la biblioteca Direct3D de Microsoft Corporation. En algunas realizaciones, una combinación de estas bibliotecas puede ser compatible. También puede ser compatible con la biblioteca Open Source Computer Vision Library (OpenCV). Una futura API con una canalización 3D compatible también puede ser compatible si se puede hacer un mapeo desde la canalización de la futura API a la canalización del procesador de gráficos.

Programación de la canalización de gráficos

La **Figura 34A** es un diagrama de bloques que ilustra un formato de comandos de procesador de gráficos 3400 de acuerdo con algunas realizaciones. La **Figura 34B** es un diagrama de bloques que ilustra una secuencia de comandos de procesador de gráficos 3410 de acuerdo con una realización. Los recuadros de línea continua en la **Figura 34A** 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 están incluidos únicamente en un subconjunto de comandos de gráficos. El formato de comandos de procesador de gráficos 3400 ilustrativo de la **Figura 34A** incluye campos de datos para identificar un cliente objetivo 3402 del comando, un código de operación del comando (código) 3404 y los datos relevantes 3406 para el comando. También se incluye un sub-código 3405 y un tamaño de comando 3408 en algunos comandos.

En algunas realizaciones, el cliente 3402 especifica la unidad cliente del dispositivo gráfico que procesa los datos de comando. En algunas realizaciones, un analizador de comandos del procesador de gráficos examina el campo de cliente de cada comando para condicionar el procesamiento posterior del comando y dirigir los datos de comando a la unidad de cliente apropiada. En algunas realizaciones, las unidades de cliente del procesador de gráficos incluyen una unidad de interfaz de memoria, una unidad de representación, una unidad 2D, una unidad 3D y una unidad multimedia. Cada unidad de cliente tiene una canalización de procesamiento correspondiente que procesa los comandos. Una vez que el comando es recibido por la unidad de cliente, la unidad de cliente lee el código 3404 y, si está presente, el sub-código 3405 para determinar la operación a llevar a cabo. La unidad de cliente realiza el comando usando información en el campo de datos 3406. Para algunos comandos se espera que un tamaño de comando 3408 explícito 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 comandos. En algunas realizaciones, los comandos se alinean mediante múltiplos de una palabra doble.

El diagrama de flujo en la **Figura 34B** muestra una secuencia de comandos de procesador de gráficos 3410 ilustrativo. En algunas realizaciones, el software o firmware de un sistema de procesamiento de datos que presenta 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 una secuencia de comandos de muestra y se describe únicamente con fines ilustrativos, ya que las realizaciones no están limitadas a estos comandos específicos o para esta secuencia de comandos. Además, los comandos pueden emitirse como un lote de comandos en una secuencia de comandos, de modo que el procesador de gráficos procese la secuencia de comandos al menos parcialmente de forma simultánea.

En algunas realizaciones, la secuencia de comandos de procesador de gráficos 3410 puede comenzar con un comando de descarga de canalización 3412 para hacer que cualquier canalización de gráficos activa complete los comandos actualmente pendientes para la canalización. En algunas realizaciones, la canalización 3D 3422 y la canalización de medios 3424 no funcionan simultáneamente. Se realiza la descarga de la canalización para hacer que la canalización de gráficos activa complete cualquier comando pendiente. En respuesta a una descarga de canalización, el analizador de comandos para el procesador de gráficos pausará el procesamiento de comandos hasta que los motores de dibujo activos completen las operaciones pendientes y se invaliden las cachés de lectura relevantes. Opcionalmente, cualquier dato en la caché de representación que esté marcado como "sucio" se puede

descargar de la memoria. En algunas realizaciones, el comando de descarga de canalización 3412 se puede utilizar para la sincronización de la canalización o antes de colocar el procesador de gráficos en un estado de bajo consumo.

En algunas realizaciones, se usa un comando de selección de canalización 3413 cuando una secuencia de comandos requiere que el procesador de gráficos cambie explícitamente entre canalizaciones. En algunas realizaciones, se requiere un comando de selección de canalización 3413 solo una vez dentro de un contexto de ejecución antes de emitir comandos de canalización, a menos que el contexto sea para emitir comandos para ambas canalizaciones. En algunas realizaciones, se requiere un comando de descarga de canalización 3412 inmediatamente antes de una conmutación de canalización mediante el comando de selección de canalización 3413.

En algunas realizaciones, un comando de control de canalización 3414 configura una canalización de gráficos para su operación y se utiliza para programar la canalización 3D 3422 y la canalización de medios 3424. En algunas realizaciones, el comando de control de canalización 3414 configura el estado de canalización para la canalización activa. En una realización, se usa el comando de control de canalización 3414 para la sincronización de canalización y para limpiar 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 de estado de memoria intermedia de retorno 3416 para configurar un conjunto de memorias intermedias de retorno para que las respectivas canalizaciones escriban datos. Algunas operaciones de canalización requieren la asignación, selección o configuración de una o más memorias intermedias de retorno en las que las operaciones escriben datos intermedios durante el procesamiento. En algunas realizaciones, el procesador de gráficos también usa una o más memorias intermedias de retorno para almacenar datos de salida y realizar comunicación de hilos cruzada. En algunas realizaciones, el estado de memoria intermedia de retorno 3416 incluye seleccionar el tamaño y número de memorias intermedias de retorno para su uso para un conjunto de operaciones de canalización.

Los comandos restantes en la secuencia de comandos difieren basándose en la canalización activa para las operaciones. Basándose en una determinación de canalización 3420, la secuencia de comandos se adapta a la canalización 3D 3422 comenzando con el estado de canalización 3D 3430, o a la canalización de medios 3424 comenzando en el estado de canalización de medios 3440.

Los comandos para configurar el estado de canalización 3D 3430 incluyen comandos de configuración de estado 3D para el estado de memoria intermedia de vértices, el estado de elemento de vértice, el estado de color constante, el estado de memoria intermedia de profundidad y otras variables de estado que deben configurarse antes de que se procesen comandos de primitivas 3D. Los valores de estos comandos se determinan, al menos en parte, en función del API 3D particular en uso. En algunas realizaciones, los comandos de estado de canalización 3D 3430 también pueden desactivar o eludir selectivamente ciertos elementos de canalización si esos elementos no se usan.

En algunas realizaciones, el comando de primitivas 3D 3432 se usa para enviar primitivas 3D que serán procesadas mediante la canalización 3D. Los comandos y parámetros asociados que se pasan al procesador de gráficos a través del comando de primitiva 3D 3432 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 primitiva 3D 3432 para generar estructuras de datos de vértices. Las estructuras de datos de vértices se almacenan en una o más memorias intermedias de retorno. En algunas realizaciones, se usa el comando de primitiva 3D 3432 para realizar operaciones de vértice en primitivas 3D mediante sombreadores de vértice. Para procesar sombreadores de vértice, la canalización 3D 3422 envía hilos de ejecución de sombreadores a unidades de ejecución del procesador de gráficos.

En algunas realizaciones, la canalización 3D 3422 se desencadena a través de un comando o evento de ejecución 3434. En algunas realizaciones, una escritura de registro desencadena una ejecución de comando. En algunas realizaciones, la ejecución se desencadena por medio de un comando 'ir' o 'poner en marcha' en la secuencia de comandos. En una realización, la ejecución de comando se desencadena usando un comando de sincronización de canalización para descargar la secuencia de comandos a través de la canalización de gráficos. La canalización 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 pueden incluirse 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 3410 sigue la ruta de canalización de medios 3424 cuando se llevan a cabo operaciones de medios. En general, el uso específico y la forma de programación para la canalización de medios 3424 depende de los medios o de las operaciones de cálculo que se van a realizar. Las operaciones de decodificación de medios específicas pueden descargarse en la canalización de medios durante la decodificación de medios. En algunas realizaciones, puede eludirse también la canalización de medios y puede realizarse la decodificación de medios, en su totalidad o en parte, usando recursos proporcionados por uno o más núcleos de procesamiento de propósito general. En una realización, la canalización de medios también incluye elementos para operaciones de unidad de procesador de gráficos de propósito general (GPGPU), donde el procesador

de gráficos se usa para realizar operaciones de vector SIMD usando programas sombreadores computacionales que no están relacionados explícitamente con la representación de primitivas de gráficos.

En algunas realizaciones, la canalización de medios 3424 se configura de una manera similar a la canalización 3D 3422. Un conjunto de comandos para configurar el estado de canalización de medios 3440 se despacha o coloca en una cola de comandos antes de los comandos de objeto de medios 3442. En algunas realizaciones, los comandos de estado de canalización de medios 3440 incluyen datos para configurar los elementos de canalización de medios que se usarán para procesar los objetos de medios. Esto incluye datos para configurar la lógica de decodificación de vídeo y codificación de vídeo dentro de la canalización de medios, tal como el formato de codificación o decodificación. En algunas realizaciones, los comandos de estado de canalización de medios 3440 también soportan el uso de uno o más punteros a elementos de estado "indirecto" que contienen un lote de ajustes de estado.

En algunas realizaciones, los comandos de objeto de medios 3442 suministran punteros a objetos de medios para su procesamiento por la canalización de medios. Los objetos de medios incluyen memorias intermedias de memoria que contienen datos de vídeo a procesar. En algunas realizaciones, todos los estados de canalización de medios deben ser válidos antes de que se emita un comando de objeto de medios 3442. Una vez que está configurado el estado de la canalización y se ponen en cola los comandos de objeto de medios 3442, se activa la canalización de medios 3424 mediante un comando de ejecución 3444 o un evento de ejecución equivalente (por ejemplo, escritura de registro). La salida de la canalización de medios 3424 puede luego ser procesada posteriormente mediante operaciones proporcionadas por la canalización 3D 3422 o la canalización de medios 3424. 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 35** ilustra una arquitectura de software de gráficos ilustrativa para un sistema de procesamiento de datos 3500 de acuerdo con algunas realizaciones. En algunas realizaciones, la arquitectura de software incluye una aplicación de gráficos 3D 3510, un sistema operativo 3520 y al menos un procesador 3530. En algunas realizaciones, el procesador 3530 incluye un procesador de gráficos 3532 y uno o más núcleos de procesador de propósito general 3534. Cada uno de la aplicación de gráficos 3510 y el sistema operativo 3520 se ejecutan en la memoria de sistema 3550 del sistema de procesamiento de datos.

En algunas realizaciones, la aplicación de gráficos 3D 3510 contiene uno o más programas sombreadores que incluyen instrucciones de sombreador 3512. Las instrucciones de lenguaje de sombreador pueden estar en un lenguaje de sombreador de alto nivel, como por ejemplo el lenguaje de sombreador de alto nivel (HLSL) o el lenguaje de sombreador de OpenGL (GLSL). La aplicación incluye también instrucciones ejecutables 3514 en un lenguaje máquina adecuado para su ejecución por el núcleo de procesador de propósito general 3534. La aplicación también incluye los objetos de gráficos 3516 definidos por los datos de vértices.

En algunas realizaciones, el sistema operativo 3520 es un sistema operativo Microsoft® Windows® de Microsoft Corporation, un sistema operativo propietario tipo UNIX o un sistema operativo tipo UNIX de código abierto que utiliza una variante del kernel de Linux. El sistema operativo 3520 puede soportar una API de gráficos 3522 como por ejemplo la API Direct3D, la API OpenGL o la API Vulkan. Cuando está en uso la API Direct3D, el sistema operativo 3520 utiliza un compilador de sombreador de extremo frontal 3524 para compilar cualesquiera instrucciones de sombreador 3512 en HLSL en un lenguaje de sombreador de nivel inferior. La compilación puede ser una compilación justo a tiempo (JIT) o la aplicación puede llevar a cabo una precompilación de sombreador. En algunas realizaciones, los sombreadores de alto nivel se compilan en sombreadores de bajo nivel durante la compilación de la aplicación de gráficos 3D 3510. En algunas realizaciones, las instrucciones de sombreador 3512 se proporcionan en una forma intermedia, como por ejemplo una versión de la Representación Intermedia Portátil Estándar (SPIR) utilizada por la API Vulkan.

En algunas realizaciones, el controlador de gráficos de modo de usuario 3526 contiene un compilador de sombreador de extremo trasero 3527 para convertir las instrucciones de sombreador 3512 en una representación específica de hardware. Cuando la API OpenGL está en uso, las instrucciones de sombreador 3512 en el lenguaje de alto nivel GLSL se pasan al controlador de gráficos de modo de usuario 3526 para su compilación. En algunas realizaciones, el controlador de gráficos de modo de usuario 3526 utiliza las funciones de modo de núcleo de sistema operativo 3528 para comunicarse con un controlador de gráficos de modo de núcleo 3529. En algunas realizaciones, el controlador de gráficos de modo de núcleo 3529 se comunica con el procesador de gráficos 3532 para despachar comandos e instrucciones.

Implementaciones de núcleos de IP

Uno o más aspectos de al menos una realización se pueden implementar mediante código representativo almacenado en un medio legible por máquina que representa y/o define la lógica dentro de un circuito integrado como por ejemplo un procesador. Por ejemplo, el medio legible por máquina puede incluir instrucciones que representan diversas lógicas dentro del procesador. Cuando son leídas por una máquina, las instrucciones pueden hacer que la máquina fabrique la lógica para llevar a cabo las técnicas descritas en el presente documento. Dichas 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 tal forma que el circuito lleve a cabo las operaciones descritas en asociación con cualquiera de las realizaciones descritas en el presente documento.

La **Figura 36** es un diagrama de bloques que ilustra un sistema de desarrollo de núcleo de IP 3600 que puede usarse para fabricar un circuito integrado para realizar las operaciones de acuerdo con una realización. El sistema de desarrollo de núcleo de IP 3600 puede usarse para generar diseños reutilizables modulares que pueden incorporarse en un diseño más grande o usarse para construir un circuito integrado entero (por ejemplo, un circuito de SOC integrado). Una instalación de diseño 3630 puede generar una simulación de software 3610 de un diseño de núcleo de IP en un lenguaje de programación de alto nivel (por ejemplo, C/C++). La simulación de software 3610 se puede usar para diseñar, probar y verificar el comportamiento del núcleo IP usando un modelo de simulación 3612. El modelo de simulación 3612 puede incluir simulaciones funcionales, de comportamiento y/o de temporización. Puede crearse o sintetizarse, a continuación, un diseño de nivel de transferencia de registro (RTL) 3615 a partir del modelo de simulación 3612. El diseño de RTL 3615 es una abstracción del comportamiento del circuito integrado que modela el flujo de señales digitales entre registros de hardware, incluyendo la lógica asociada realizada usando las señales digitales modeladas. Además de un diseño de RTL 3615, también pueden crearse, diseñarse o sintetizarse diseños de nivel inferior a nivel de lógica o a nivel de transistores. Por lo tanto, los detalles particulares del diseño y simulación inicial pueden variar.

El diseño de RTL 3615, o un equivalente, puede ser sintetizado adicionalmente por la instalación de diseño para dar un modelo de hardware 3620, que puede estar en un lenguaje de descripción de hardware (HDL) o alguna otra representación de datos de diseño físico. El HDL puede simularse o someterse a prueba adicionalmente para verificar el diseño de núcleo de IP. El diseño de núcleo de IP puede almacenarse para su entrega a una instalación de fabricación de terceros 3665 usando la memoria no volátil 3640 (por ejemplo, disco duro, memoria flash o cualquier medio de almacenamiento no volátil). Como alternativa, el diseño de núcleo de IP puede transmitirse (por ejemplo, mediante Internet) a través de una conexión cableada 3650 o una conexión inalámbrica 3660. La instalación de fabricación 3665 puede luego fabricar un circuito integrado que se base al menos en parte en el diseño de núcleo de IP. El circuito integrado fabricado puede estar configurado para realizar operaciones de acuerdo con al menos una realización descrita en el presente documento.

Circuito integrado de sistema en chip ilustrativo

Las **Figuras 37-39** ilustran circuitos integrados ilustrativos y procesadores de gráficos asociados que pueden fabricarse usando uno o más núcleos de IP, de acuerdo con diversas realizaciones descritas en el presente documento. Además de lo que se ilustra, puede incluirse otra lógica y circuitos, que incluyen procesadores de gráficos/núcleos adicionales, controladores de interfaz periférica o núcleos de procesador de propósito general.

La **Figura 37** es un diagrama de bloques que ilustra un circuito integrado de sistema en chip 3700 ilustrativo que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El circuito integrado 3700 ilustrativo incluye uno o más procesadores de aplicaciones 3705 (por ejemplo, unas CPU), al menos un procesador de gráficos 3710, y puede incluir adicionalmente un procesador de imágenes 3715 y/o un procesador de vídeo 3720, cualquiera de los cuales puede ser un núcleo de IP modular desde las mismas o múltiples instalaciones de diseño diferentes. El circuito integrado 3700 incluye una lógica de bus o de periféricos que incluye un controlador de USB 3725, un controlador de UART 3730, un controlador de SPI/SDIO 3735 y un controlador de I²S/I²C 3740. Además, el circuito integrado puede incluir un dispositivo de pantalla 3745 acoplado a uno o más de un controlador 3750 de interfaz multimedia de alta definición (HDMI) y una interfaz de visualización de interfaz de procesador de industria móvil (MIPI) 3755. El almacenamiento puede proporcionarse por un subsistema de memoria flash 3760 que incluye la memoria flash y un controlador de memoria flash. La interfaz de memoria se puede proporcionar mediante un controlador de memoria 3765 para el acceso a dispositivos de memoria SDRAM o SRAM. Algunos circuitos integrados incluyen adicionalmente un motor de seguridad integrado 3770.

La **Figura 38** es un diagrama de bloques que ilustra un procesador de gráficos 3810 ilustrativo de un circuito integrado de sistema en chip que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El procesador de gráficos 3810 puede ser una variante del procesador de gráficos 3710 de la **Figura 37**. El procesador de gráficos 3810 incluye un procesador de vértices 3805 y uno o más procesadores de fragmentos 3815A-3815N (por ejemplo, 3815A, 3815B, 3815C, 3815D a 3815N-1 y 3815N). El procesador de gráficos 3810 puede ejecutar diferentes programas sombreadores por medio de una lógica diferente, de tal forma que el procesador de vértices 3805 se optimiza para ejecutar operaciones para programas sombreadores de vértices, mientras que los uno o más procesadores de fragmentos 3815A-3815N ejecutan operaciones de sombreado de fragmentos (por ejemplo, píxeles) para programas sombreadores de fragmentos o de píxeles. El procesador de vértices 3805 lleva a cabo la fase de procesamiento de vértices de la canalización de gráficos 3D y genera datos de primitivas y de vértices. El/los procesador(es) de fragmentos 3815A-3815N utilizan los datos de primitivas y de vértices generados por el procesador de vértices 3805 para producir una memoria intermedia de tramas que se muestra en un dispositivo de visualización.

En una realización, el/los procesador(es) de fragmentos 3815A-3815N se optimizan para ejecutar programas sombreadores de fragmentos según lo previsto en la API de OpenGL, que se pueden utilizar para llevar a cabo operaciones similares como un programa sombreador de píxeles según lo previsto en la API de Direct 3D.

El procesador de gráficos 3810 incluye adicionalmente una o más unidades de gestión de memoria (MMU) 3820A-3820B, caché o cachés 3825A-3825B e interconexión o interconexiones de circuito 3830A-3830B. La una o más MMU 3820A-3820B proporcionan mapeo de dirección virtual a física para el circuito integrado 3810, incluyendo para el procesador de vértices 3805 y/o el/los procesador(es) de fragmentos 3815A-3815N, que pueden hacer referencia a los datos de vértices o de imagen/textura almacenados en memoria, además de los datos de vértices o imagen/textura almacenados en la una o más caché o cachés 3825A-3825B. En una realización, la una o más MMU 3825A-3825B pueden estar sincronizadas con otras MMU dentro del sistema, que incluyen una o más MMU asociadas con el uno o más procesadores de aplicación 3705, el procesador de imágenes 3715 y/o el procesador de vídeo 3720 de la **Figura 37**, de manera que cada procesador 3705-3720 puede participar en un sistema de memoria virtual compartida o unificada. La una o más interconexiones de circuito 3830A-3830B posibilitan que el procesador de gráficos 3810 interconecte con otros núcleos de IP dentro del SoC, mediante un bus interno del SoC o mediante una conexión directa, de acuerdo con las realizaciones.

La **Figura 39** es un diagrama de bloques que ilustra un procesador de gráficos 3910 ilustrativo adicional de un circuito integrado de sistema en chip que puede fabricarse usando uno o más núcleos de IP, de acuerdo con una realización. El procesador de gráficos 3910 puede ser una variante del procesador de gráficos 3710 de la **Figura 37**. El procesador de gráficos 3910 incluye la una o más MMU 3820A-3820B, las cachés 3825A-3825B y las interconexiones de circuito 3830A-3830B del circuito integrado 3800 de la **Figura 38**.

El procesador de gráficos 3910 incluye uno o más núcleos de sombreador 3915A-3915N (por ejemplo, 3915A, 3915B, 3915C, 3915D, 3915E, 3915F a 3915N-1, y 3915N), que proporcionan una arquitectura de núcleo de sombreador unificada en la que un único núcleo o tipo o núcleo puede ejecutar todos los tipos de código sombreador programable, que incluyen código de programa sombreador para implementar sombreadores de vértice, sombreadores de fragmento y/o sombreadores de cálculo. El número exacto de núcleos de sombreador presentes puede variar entre realizaciones e implementaciones. Adicionalmente, el procesador de gráficos 3910 incluye un gestor de tareas inter-núcleo 3905, que actúa como un despachador de hilos para despachar hilos de ejecución a uno o más núcleos de sombreador 3915A-3915N y una unidad de tesela 3918 para acelerar las operaciones de tesela para la representación basada en teselas, en las que las operaciones de representación para una escena se subdividen en el espacio de imágenes, por ejemplo, para aprovechar la coherencia de espacio local dentro de una escena o para optimizar el uso de cachés internas.

Las siguientes cláusulas y/o ejemplos pertenecen a realizaciones específicas o ejemplos de las mismas. Los detalles específicos en los ejemplos pueden usarse en cualquier parte en una o más realizaciones. Las diversas características de las diferentes realizaciones o ejemplos pueden combinarse de manera variable con algunas características incluidas y otras excluidas para adecuarse a una diversidad de diferentes aplicaciones. Los ejemplos pueden incluir la materia objeto tal como un método, medios para realizar actos del método, al menos un medio legible por máquina que incluye instrucciones que, cuando se realizan por una máquina, hacen que la máquina realice actos del método, o de un aparato o sistema de acuerdo con las realizaciones y ejemplos descritos en el presente documento. Diversos componentes pueden ser un medio para realizar las operaciones o funciones descritas.

Una realización proporciona un dispositivo informático dentro de un vehículo autónomo, comprendiendo el dispositivo informático un dispositivo de red inalámbrica para permitir una conexión de datos inalámbrica con una red de vehículo autónomo, un conjunto de múltiples procesadores que incluyen un procesador de propósito general y un procesador de gráficos de propósito general, el conjunto de múltiples procesadores para ejecutar un gestor de cálculo para gestionar la ejecución de cargas de trabajo de cálculo asociadas con el vehículo autónomo, la carga de trabajo de cálculo asociada con las operaciones autónomas del vehículo autónomo y la lógica de descarga configurada para ejecutarse en el conjunto de múltiples procesadores, la lógica de descarga para determinar descargar una o más de las cargas de trabajo de cálculo a uno o más vehículos autónomos dentro del alcance del dispositivo de red inalámbrica.

Una realización proporciona un método para realizar operaciones de aprendizaje automático para un vehículo autónomo, comprendiendo el método determinar que una carga de trabajo computacional debe ser procesada por un dispositivo informático del vehículo autónomo; determinar una primera latencia hacia un centro de datos remoto a través de una red de vehículos autónomos; enviar al menos una primera porción de la carga de trabajo computacional para su procesamiento a través del centro de datos remoto cuando la primera latencia está por debajo de un umbral asociado con la carga de trabajo computacional; determinar una segunda latencia para un vehículo autónomo dentro del alcance de un dispositivo de red inalámbrica en respuesta a una determinación de que la primera latencia está por encima del umbral asociado con la carga de trabajo computacional; y enviar al menos una segunda porción de la carga de trabajo computacional en respuesta a determinar que la segunda latencia está por debajo del umbral asociado con la carga de trabajo computacional.

5 Una realización proporciona un vehículo autónomo que comprende un dispositivo de red inalámbrica para permitir una conexión de datos inalámbrica con una red de vehículo autónomo; un conjunto de múltiples procesadores que incluye un procesador de propósito general y un procesador de gráficos de propósito general, el conjunto de múltiples procesadores para ejecutar un gestor de cálculo para gestionar la ejecución de cargas de trabajo de cálculo asociadas con el vehículo autónomo; y lógica de descarga configurada para ejecutarse en el conjunto de múltiples procesadores, la lógica de descarga para determinar descargar una o más de las cargas de trabajo de cálculo a uno o más vehículos autónomos dentro del alcance del dispositivo de red inalámbrica.

10 Las realizaciones descritas en el presente documento se refieren a configuraciones específicas de hardware, tales como circuitos integrados específicos de la aplicación (ASIC), configuradas para realizar ciertas operaciones o que tienen una funcionalidad predeterminada. Tales dispositivos electrónicos habitualmente incluyen un conjunto de uno o más procesadores acoplados a uno o más de otros 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 habitualmente se produce 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 tanto, los dispositivos de almacenamiento de un dispositivo electrónico dado habitualmente almacenan un código y/o datos para su ejecución en el conjunto de uno o más procesadores de ese dispositivo electrónico.

20 Por supuesto, una o más partes de una realización pueden implementarse usando diferentes combinaciones de software, firmware y/o hardware. A lo largo de toda esta descripción detallada, con fines explicativos, 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 las realizaciones pueden ponerse en práctica sin algunos de estos detalles específicos. En ciertas instancias, no se han descrito estructuras y funciones bien conocidas con todo lujo de detalles para evitar complicar la materia objeto inventiva de las realizaciones. En consecuencia, el alcance de la invención debería evaluarse en términos de las reivindicaciones que se aportan a continuación.

REIVINDICACIONES

1. Un dispositivo informático dentro de un vehículo autónomo, comprendiendo el dispositivo informático:
un dispositivo de red inalámbrica para permitir una conexión de datos inalámbrica con una red de vehículo autónomo;
5 un conjunto de múltiples procesadores que incluye un procesador de propósito general y un procesador de gráficos de propósito general, siendo el conjunto de múltiples procesadores para ejecutar un gestor de cálculo (1520) para gestionar la ejecución de cargas de trabajo de cálculo (1510) asociadas con el vehículo autónomo, permitiendo las cargas de trabajo de cálculo (1510) la conducción y navegación autónomas; y
10 una lógica de descarga (1522) configurada para ejecutarse en el conjunto de múltiples procesadores, siendo la lógica de descarga (1522) para determinar descargar una o más de las cargas de trabajo de cálculo (1510) a uno o más vehículos autónomos dentro del alcance del dispositivo de red inalámbrica o a un centro de datos remoto a través de la red de vehículos autónomos,
en donde la lógica de descarga (1522) debe determinar descargar una o más de las cargas de trabajo de cálculo (1510) a uno del uno o más vehículos autónomos dentro del alcance del dispositivo de red inalámbrica y el centro de
15 datos remoto a través de la red de vehículos autónomos en función de factores que incluyen la demanda computacional y la latencia de comunicación.
2. El dispositivo informático según la reivindicación 1, en donde la lógica de descarga (1522), en respuesta a un aviso de que se va a descargar una carga de trabajo de cálculo, debe determinar si una primera latencia al centro de datos
20 remoto es aceptable para la carga de trabajo de cálculo que se va a descargar y descargar la carga de trabajo de cálculo al centro de datos remoto en respuesta a una determinación de que la primera latencia está dentro de un intervalo aceptable.
3. El dispositivo informático según la reivindicación 1 o 2, en donde la lógica de descarga (1522), en respuesta a un
25 aviso de que se va a descargar una carga de trabajo de cálculo, debe determinar si una segunda latencia a uno de los uno o más vehículos autónomos dentro del alcance del dispositivo de red inalámbrica es aceptable y descargar la carga de trabajo de cálculo al vehículo autónomo asociado con la segunda latencia.
4. El dispositivo informático según la reivindicación 3, en donde la lógica de descarga (1522) debe determinar si los
30 recursos de cálculo disponibles en el vehículo autónomo asociado con la segunda latencia son suficientes para procesar la carga de trabajo de cálculo que se va a descargar antes de que la carga de trabajo de cálculo se descargue al vehículo autónomo asociado con la segunda latencia.
5. El dispositivo informático según la reivindicación 1, en donde la lógica de descarga (1522) debe determinar que
35 existe una relación segura y confiable con uno o más vehículos autónomos dentro del alcance del dispositivo de red inalámbrica antes de que la lógica de descarga (1522) determine descargar una o más de las cargas de trabajo de cálculo (1510) al uno o más vehículos autónomos.
6. El dispositivo informático según la reivindicación 1, en donde el conjunto de múltiples procesadores debe recibir una
40 carga de trabajo de cálculo descargada de un vehículo autónomo vecino.
7. El dispositivo informático según la reivindicación 6, en donde el conjunto de múltiples procesadores recibe datos de
sensores de un conjunto de sensores conectados y la carga de trabajo de cálculo descargada del vehículo autónomo
45 vecino incluye una operación de búsqueda que se realizará en función de los datos de sensores recibidos.
8. El dispositivo informático según la reivindicación 7, en donde la operación de búsqueda incluye localizar un objetivo
de una búsqueda dentro de un área de detección de sensor de un sensor de vehículo autónomo.
9. Un vehículo autónomo que comprende un dispositivo informático según una cualquiera de las reivindicaciones
50 anteriores.
10. Un método realizado por un dispositivo informático dentro de un vehículo autónomo, que comprende:
permitir, mediante un dispositivo de red inalámbrica, una conexión de datos inalámbrica con una red de vehículos
55 autónomos;
ejecutar, mediante un conjunto de múltiples procesadores que incluyen un procesador de propósito general y un
procesador de gráficos de propósito general, un gestor de cálculo para gestionar la ejecución de cargas de trabajo de
cálculo asociadas con el vehículo autónomo, permitiendo las cargas de trabajo de cálculo la conducción y navegación
autónomas; y
60 determinar, mediante una lógica de descarga que se ejecuta en el conjunto de múltiples procesadores, descargar una
o más de las cargas de trabajo de cálculo a uno o más vehículos autónomos dentro del alcance del dispositivo de red
inalámbrica o a un centro de datos remoto a través de la red de vehículos autónomos, en donde la lógica de descarga
debe determinar descargar una o más de las cargas de trabajo de cálculo a uno de los uno o más vehículos autónomos
dentro del alcance del dispositivo de red inalámbrica y el centro de datos remoto a través de la red de vehículos
65 autónomos en función de factores que incluyen la demanda computacional y la latencia de comunicación.

11. Uno o más medios de almacenamiento legibles por máquina que almacenan instrucciones en los mismos que, cuando se ejecutan mediante un dispositivo informático, configuran el dispositivo informático para realizar un método según la reivindicación 10.

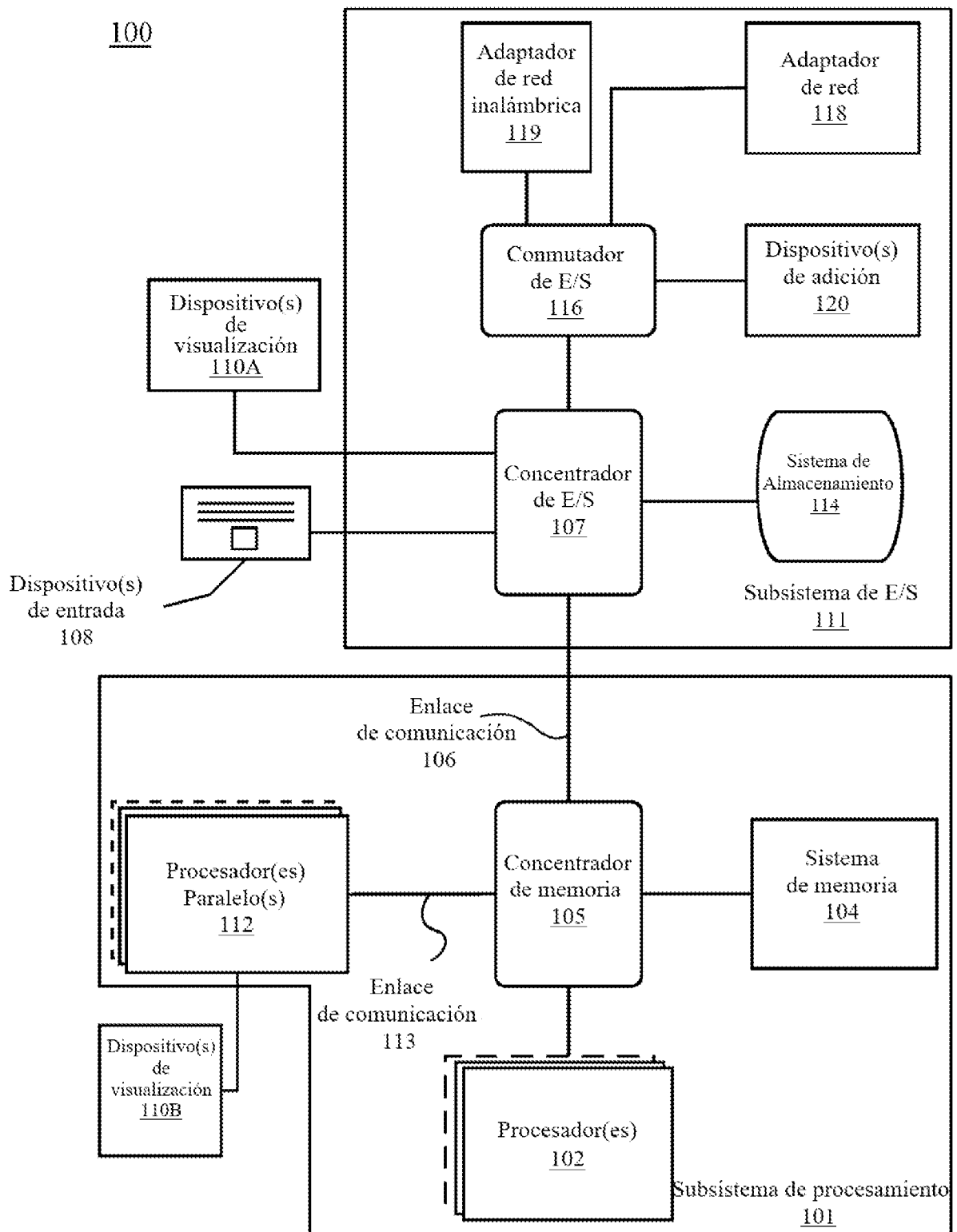


FIG. 1

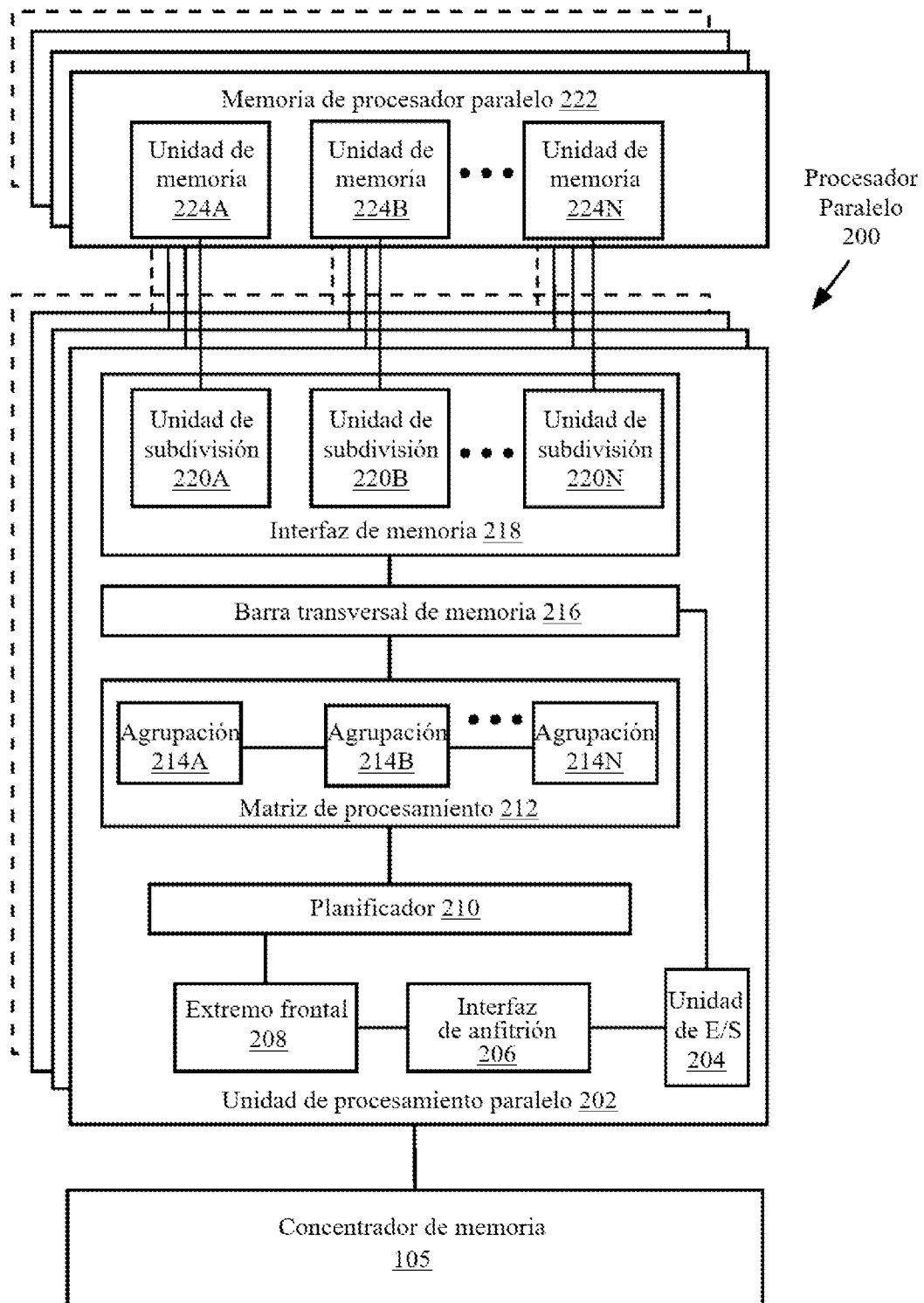


FIG. 2A

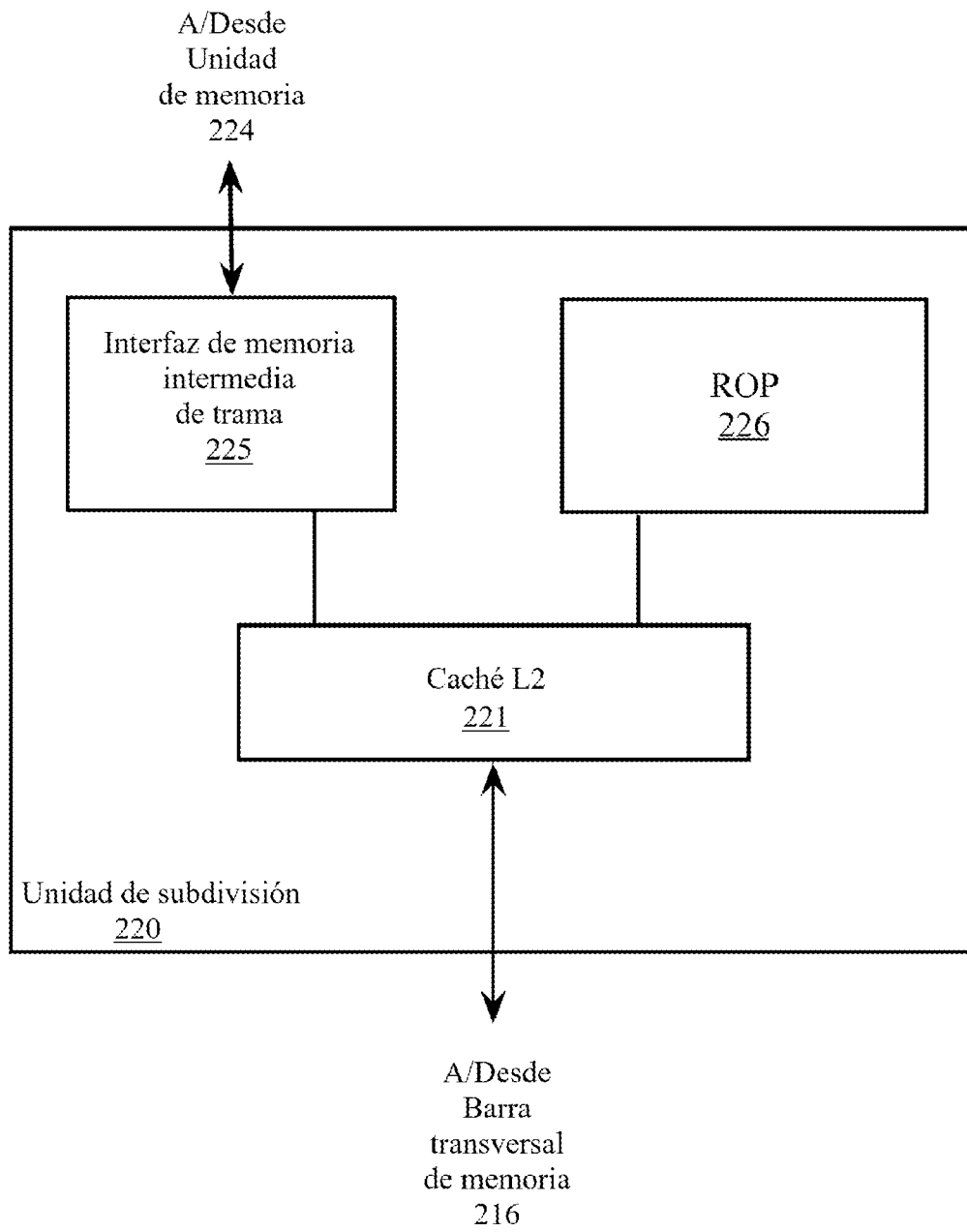


FIG. 2B

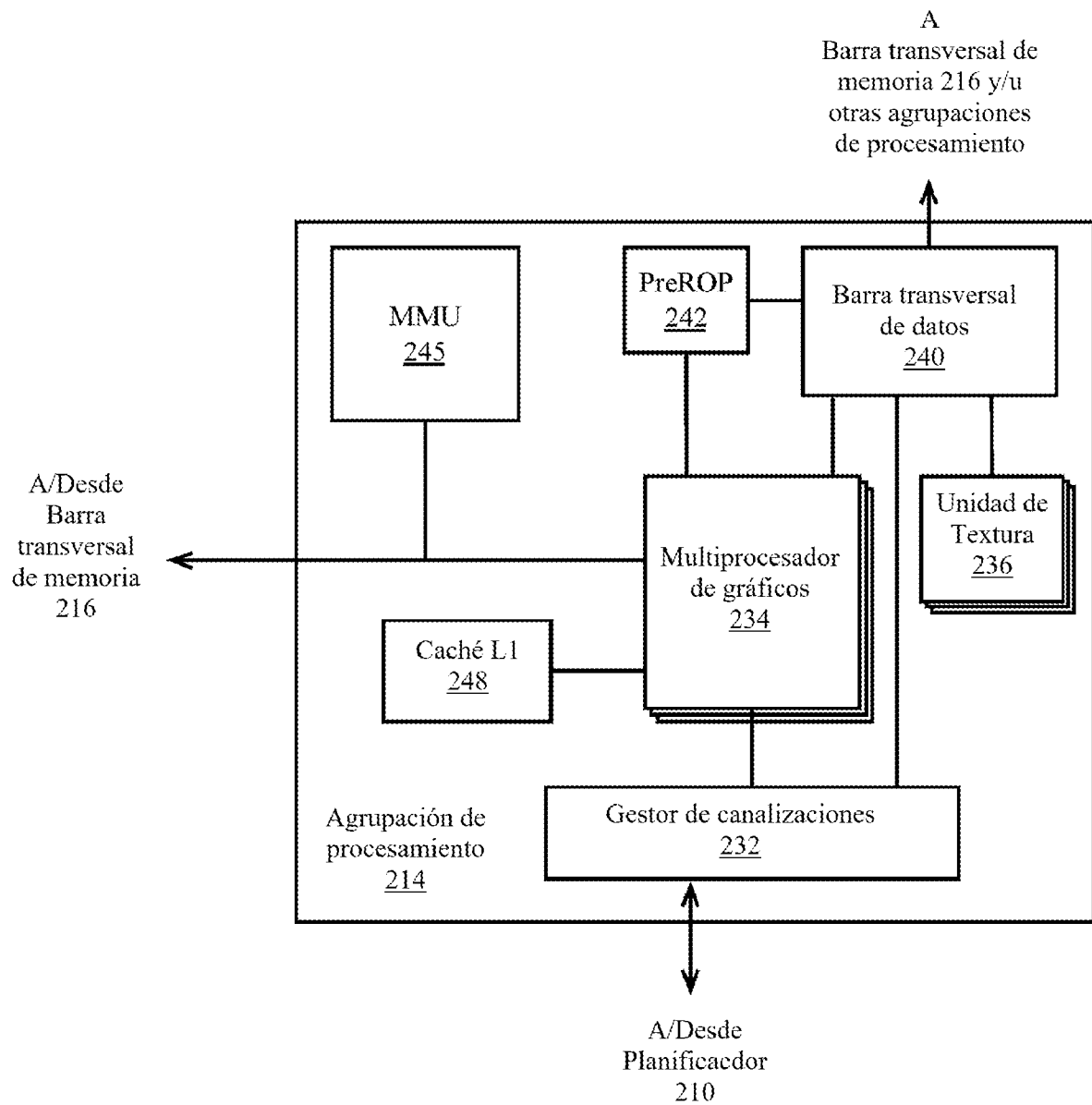


FIG. 2C

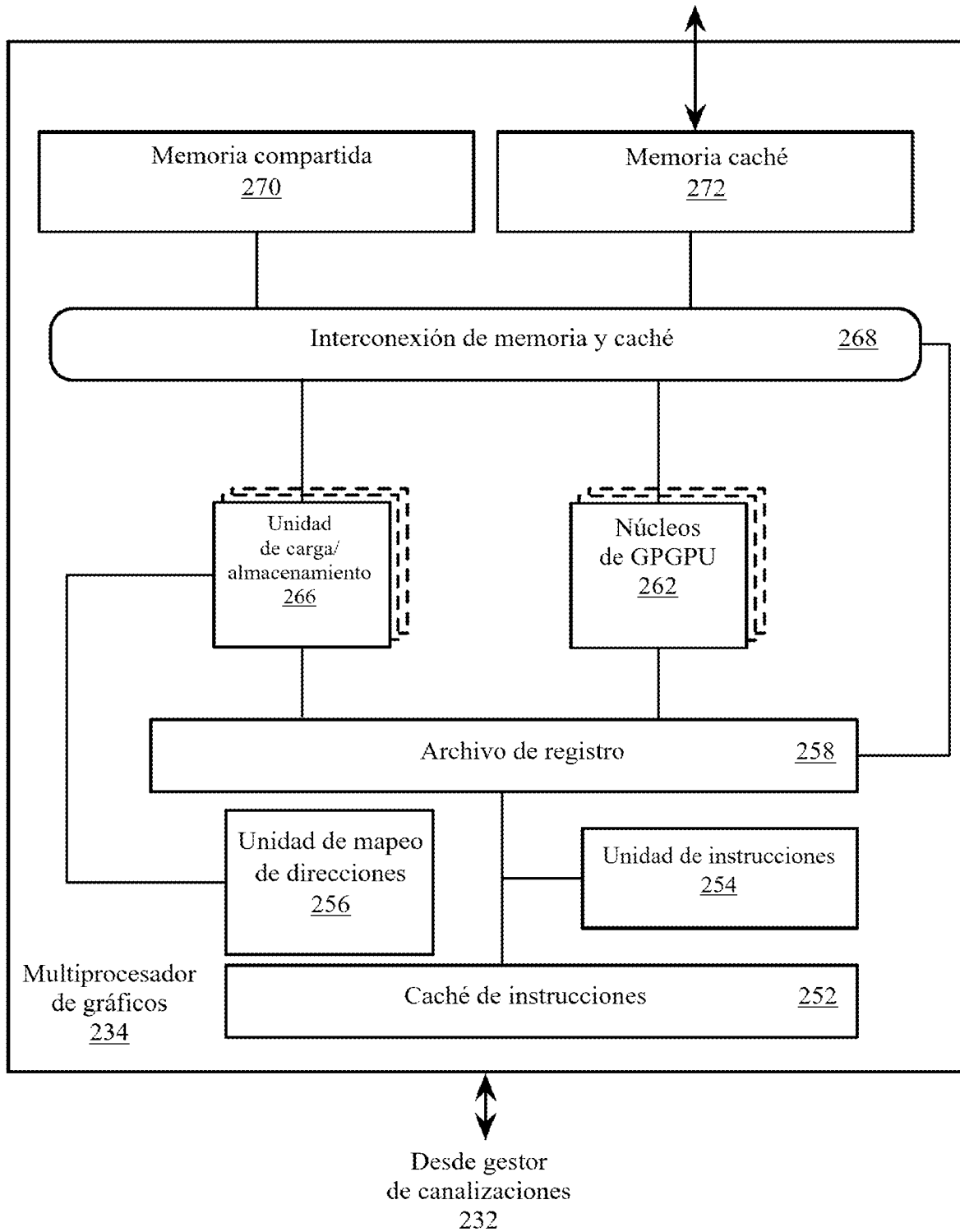


FIG. 2D

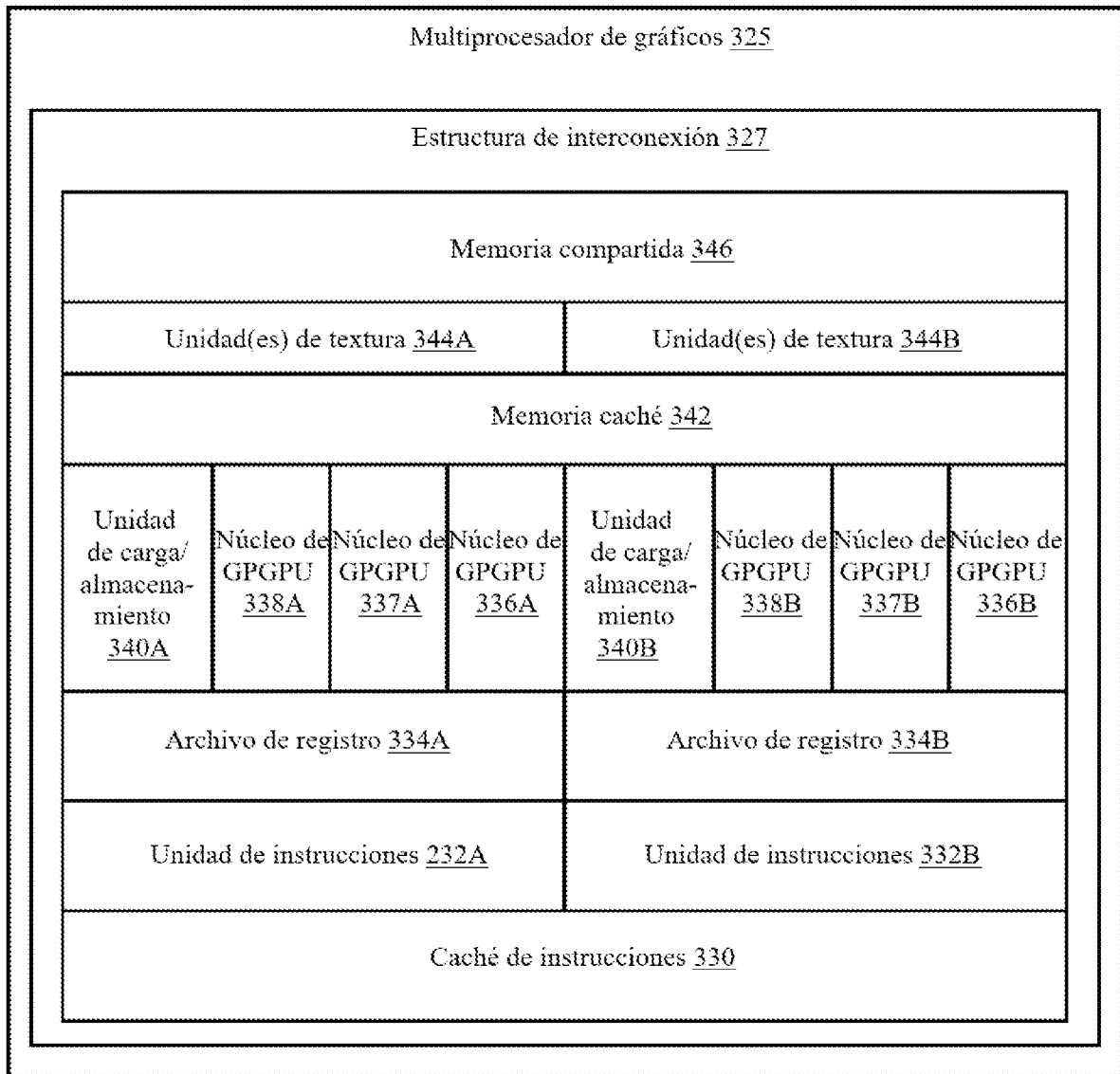


FIG. 3A

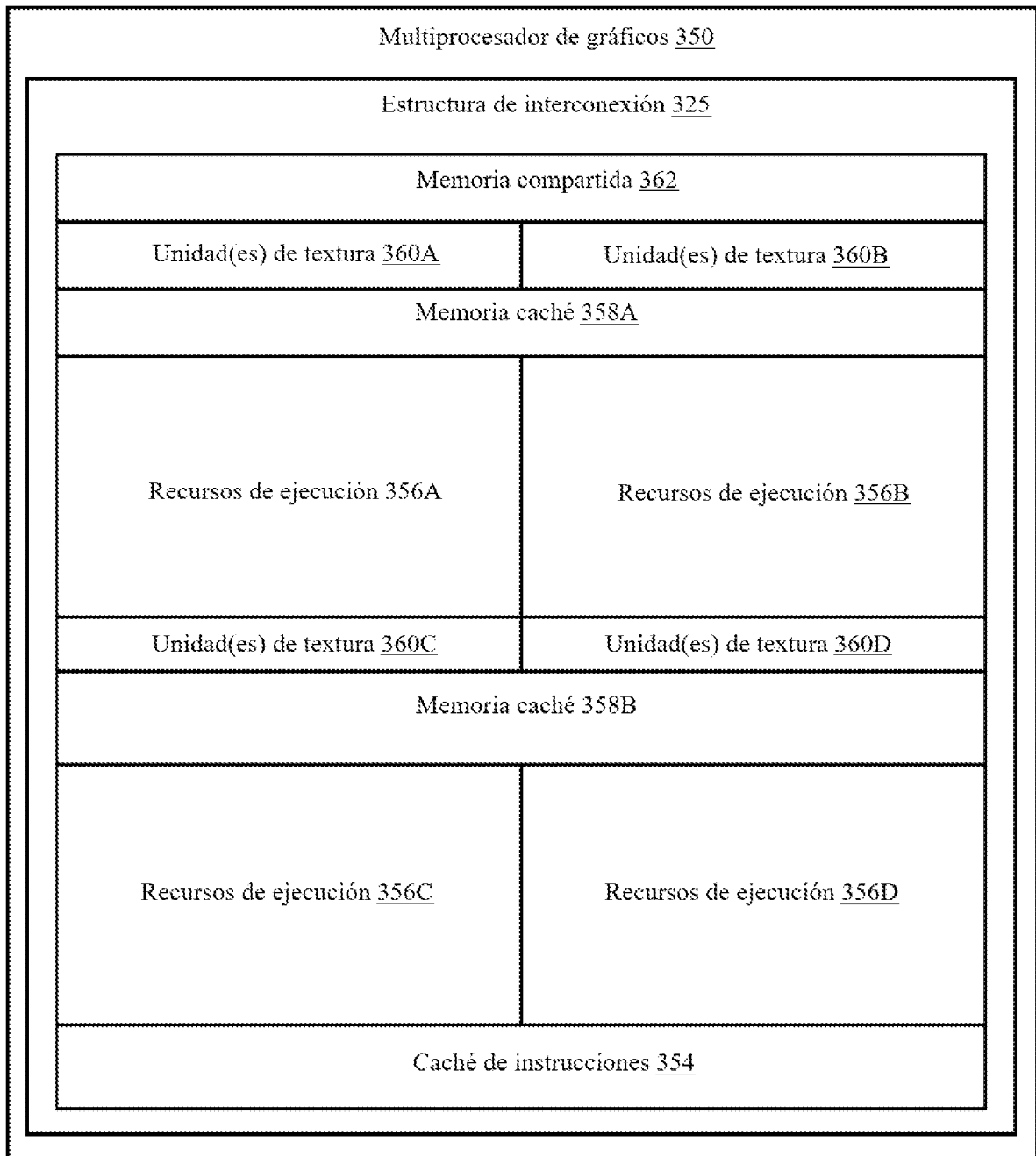


FIG. 3B

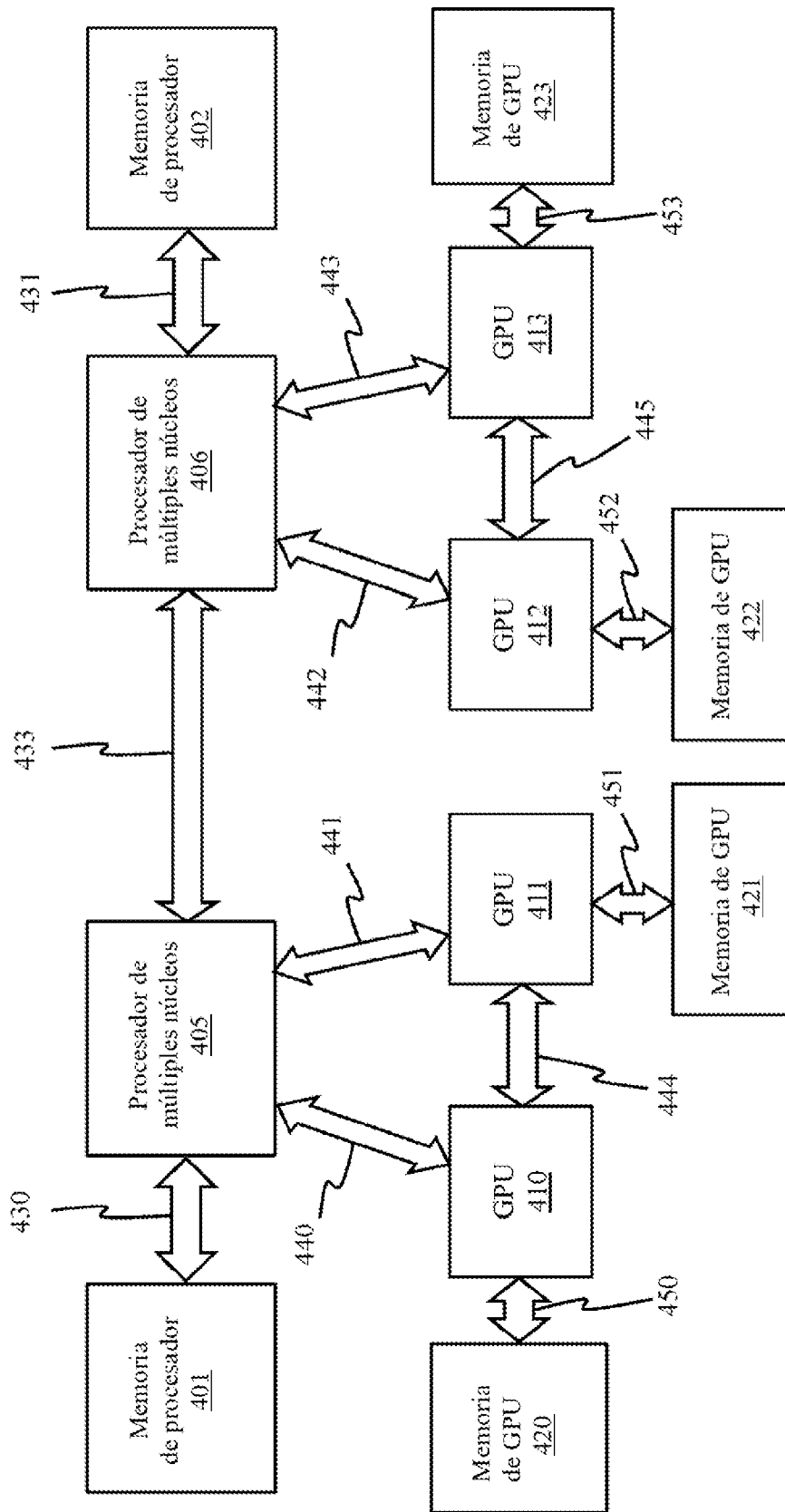


FIG. 4A

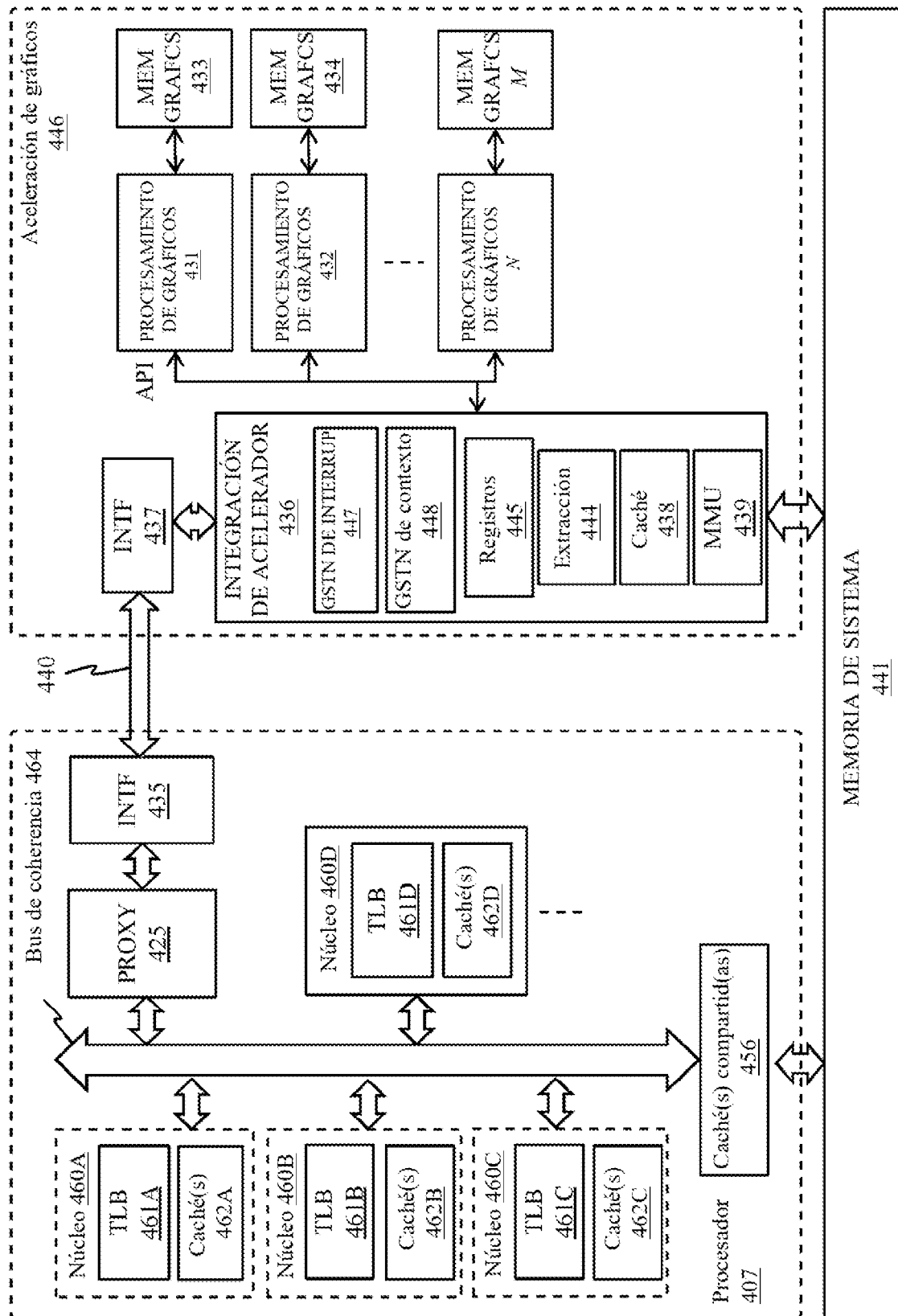


FIG. 4B

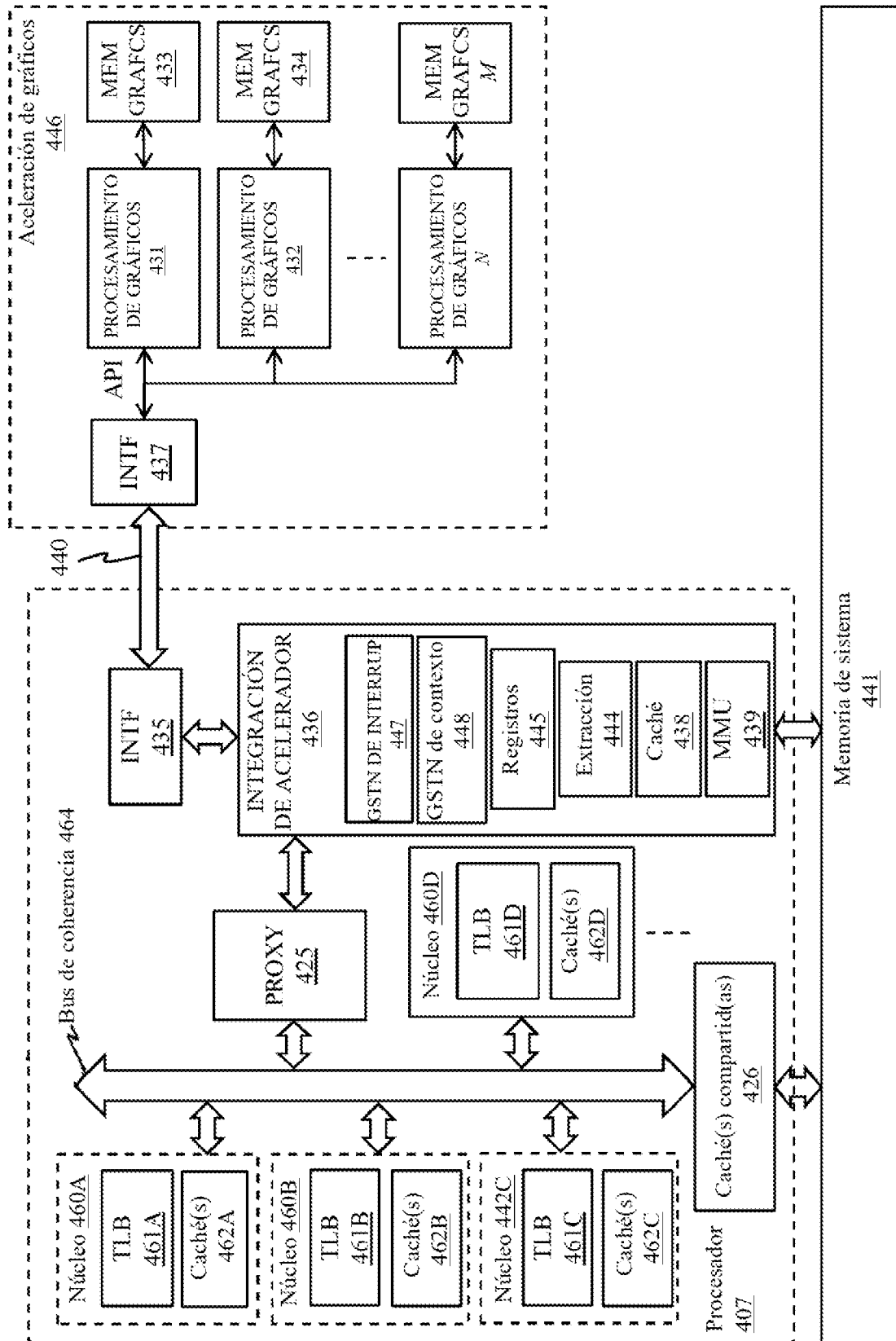


FIG. 4C

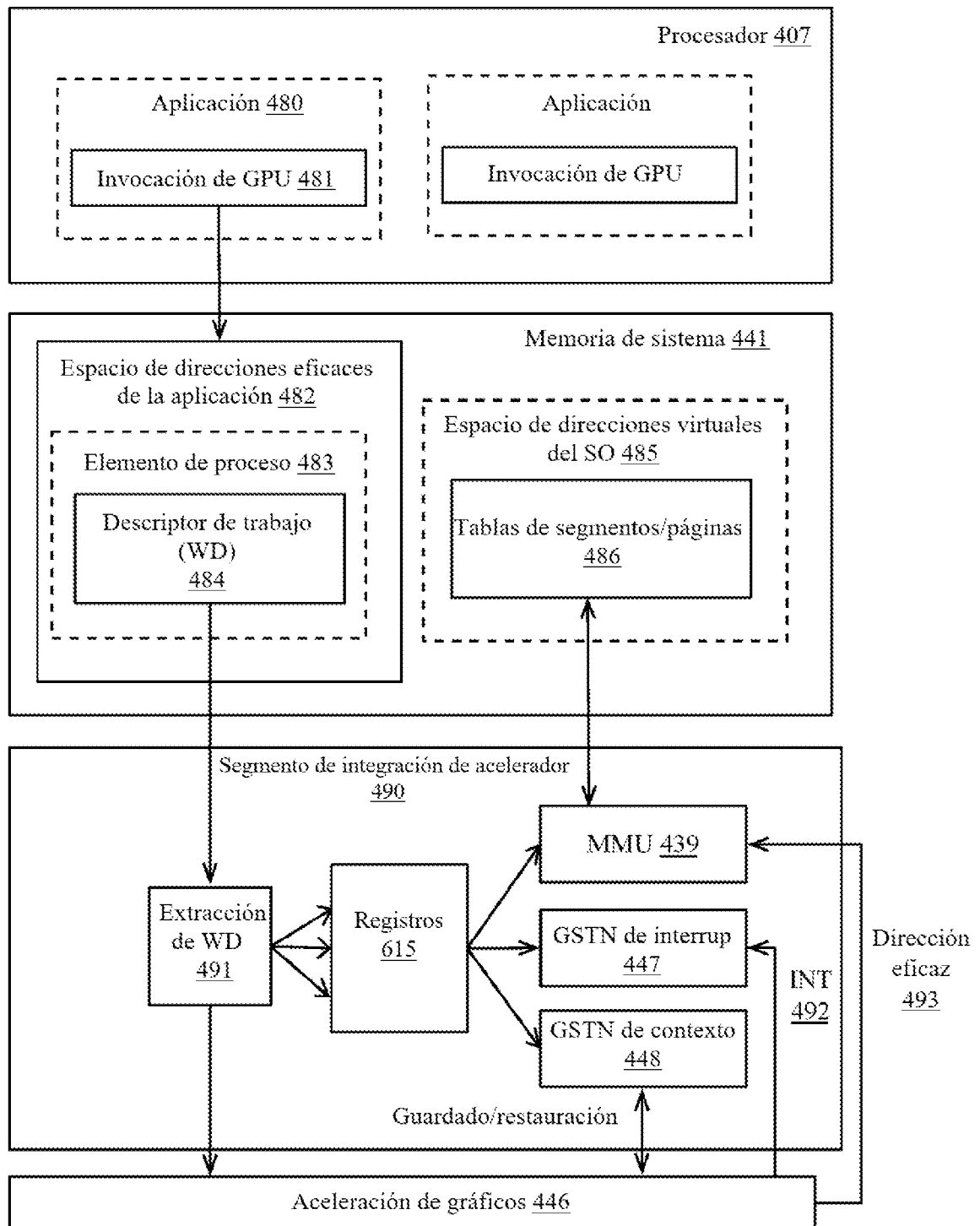


FIG. 4D

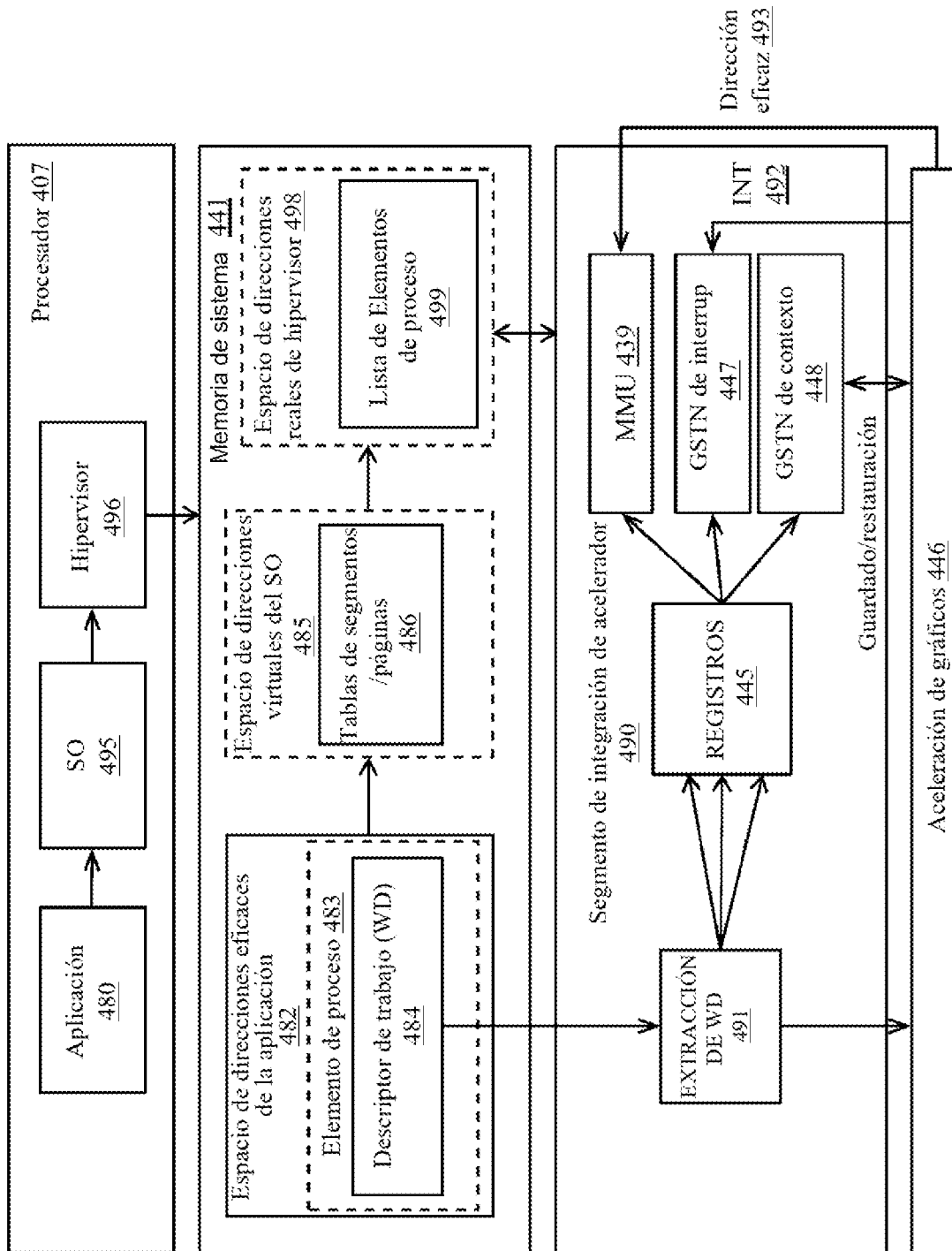


FIG. 4E

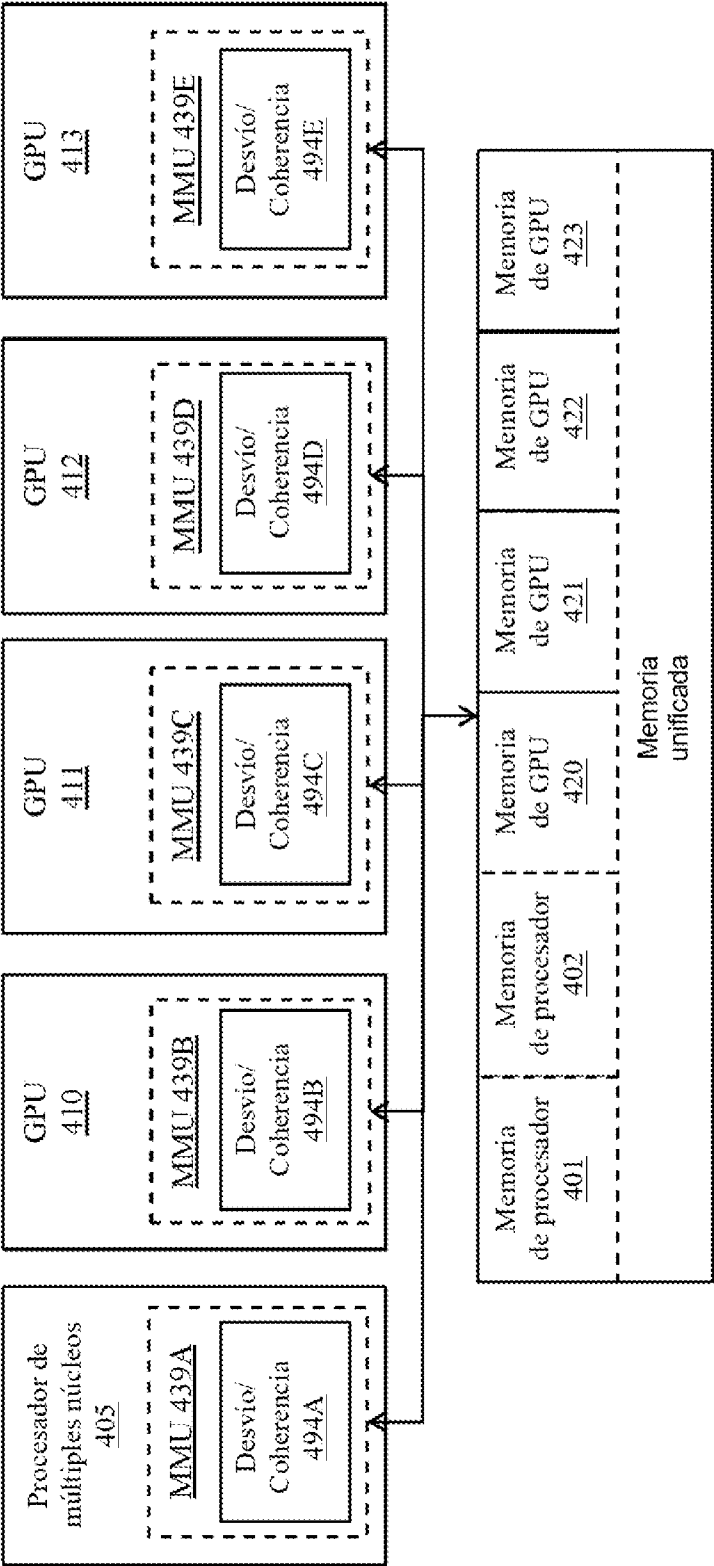


FIG. 4F

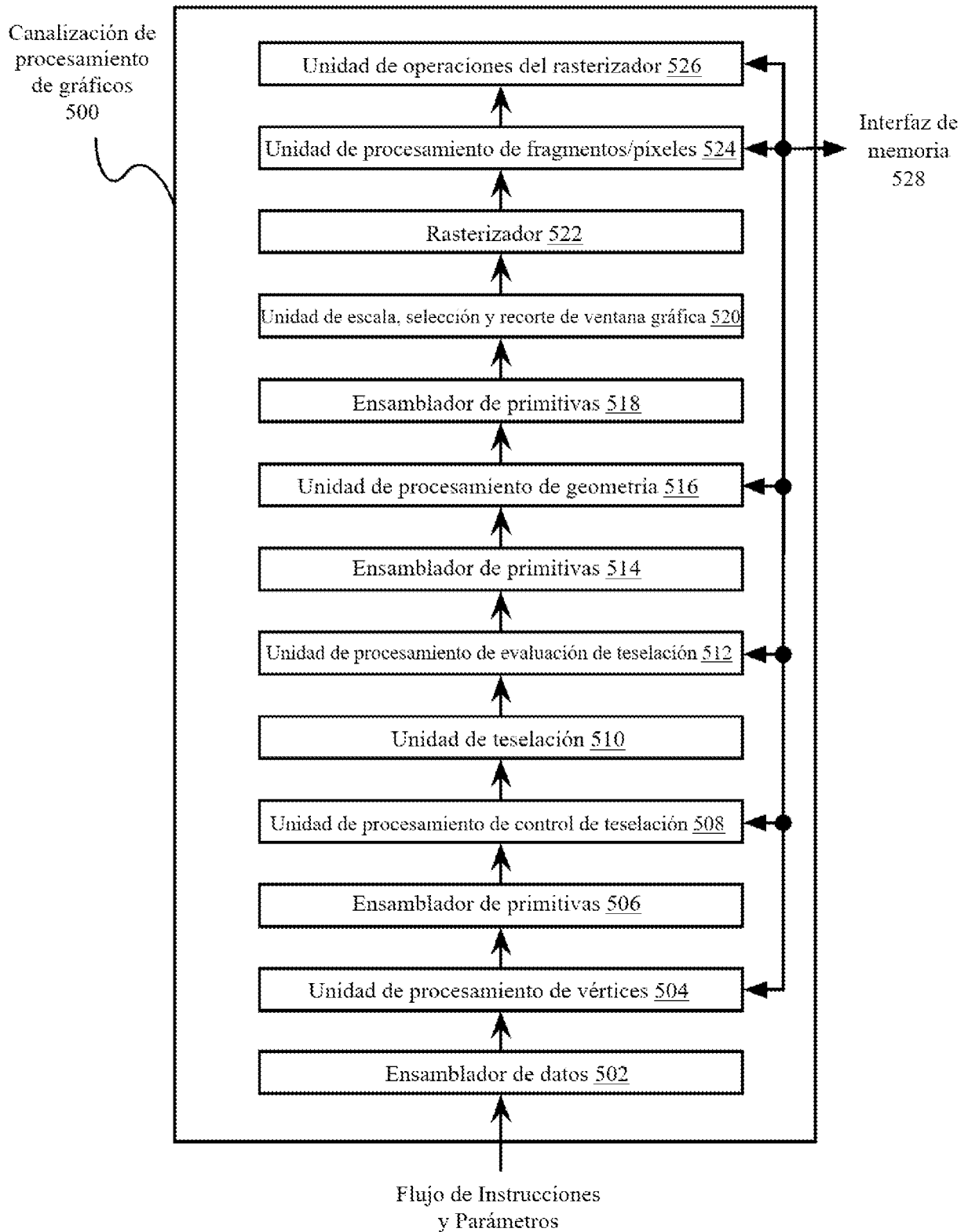


FIG. 5

600

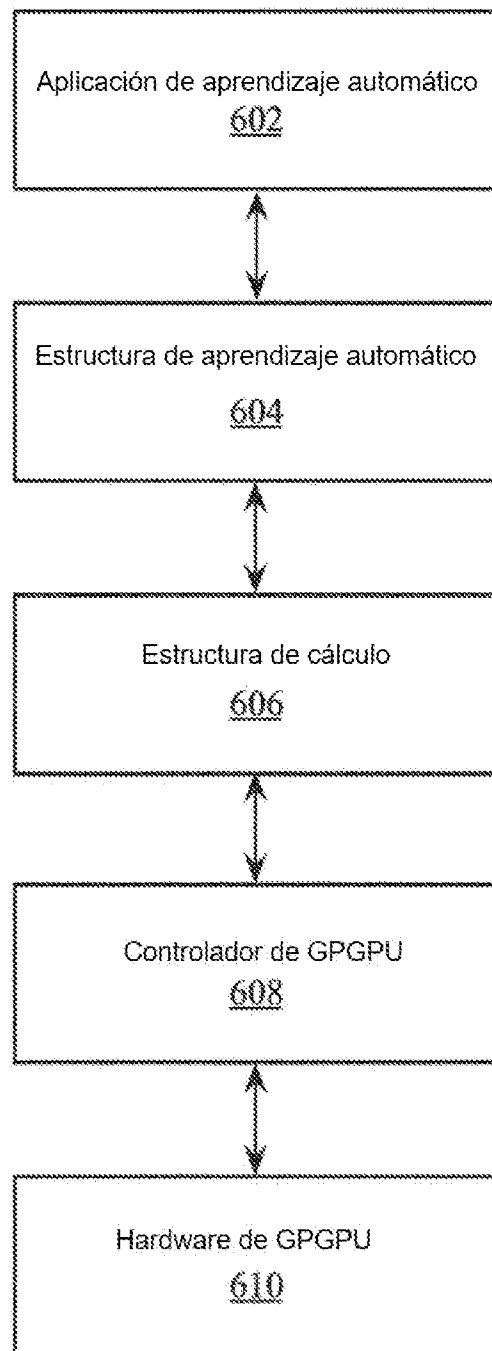


FIG. 6

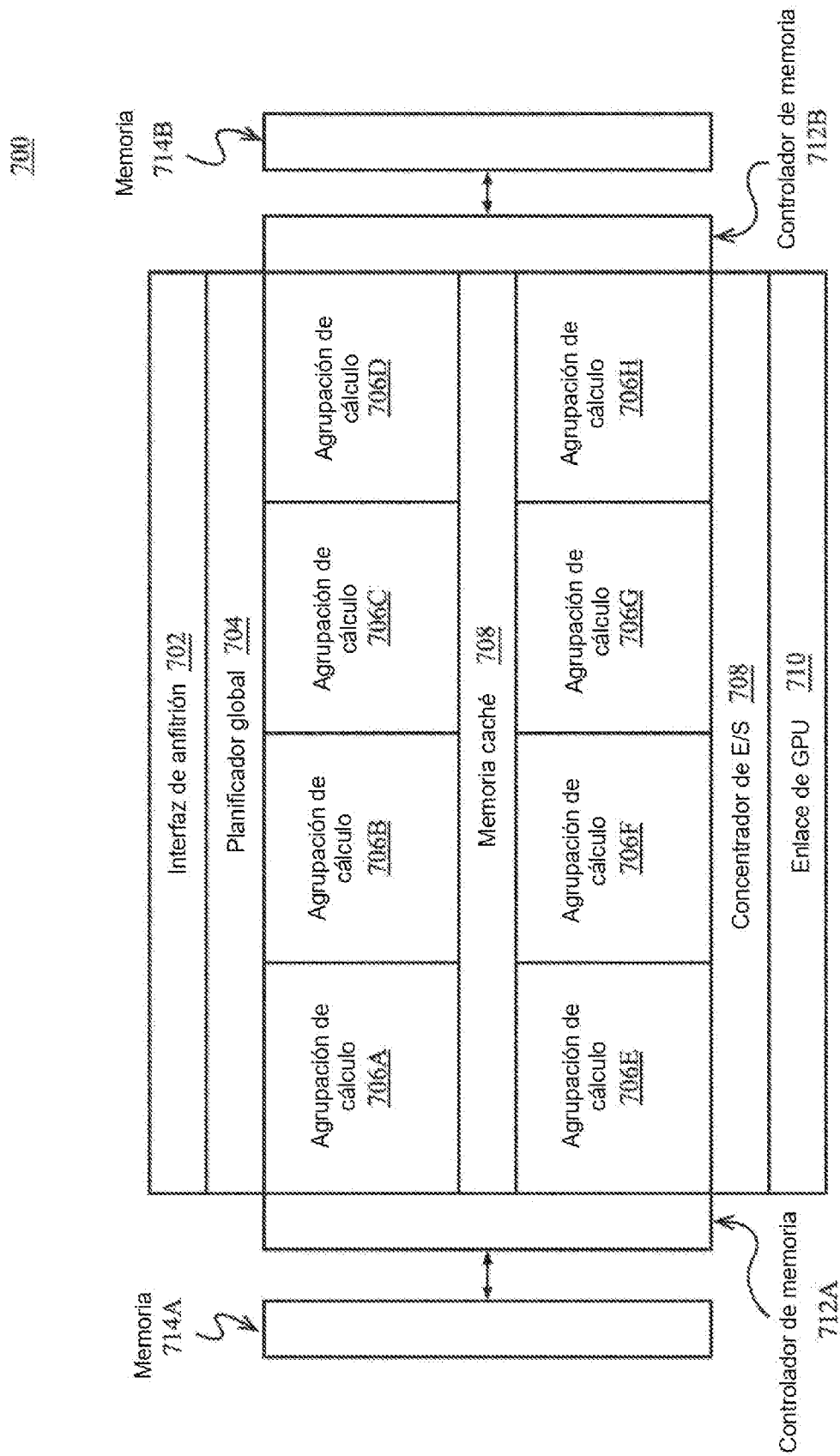


FIG. 7

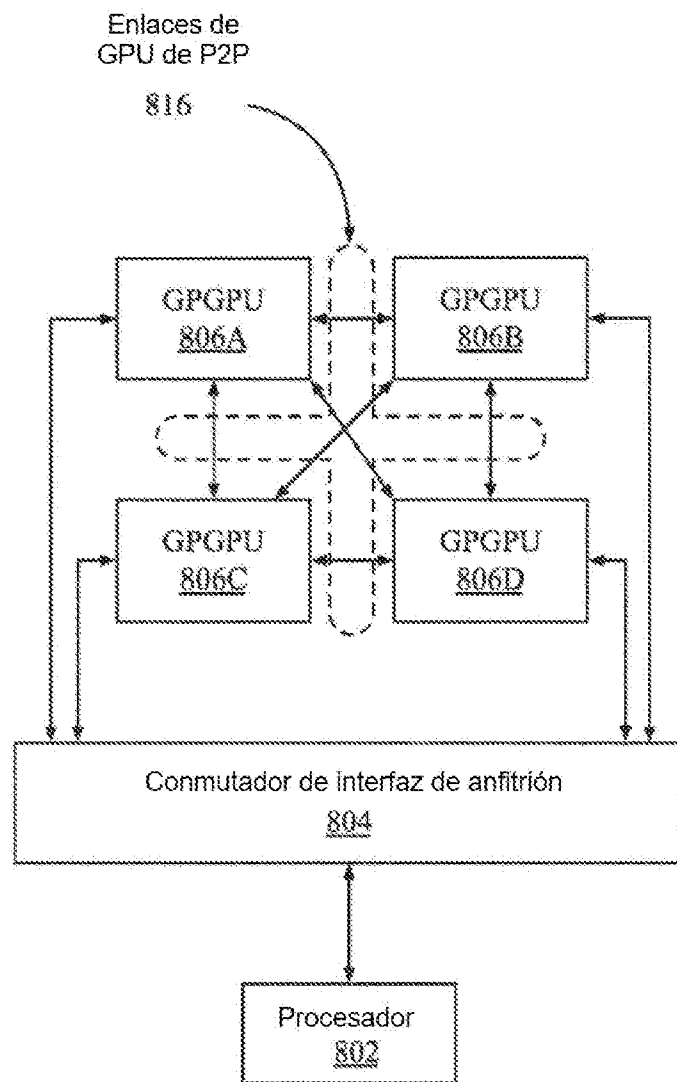


FIG. 8

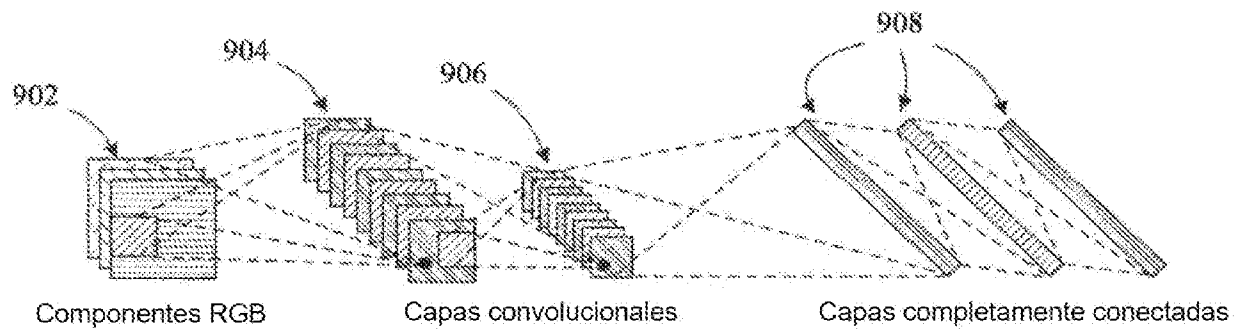


FIG. 9A

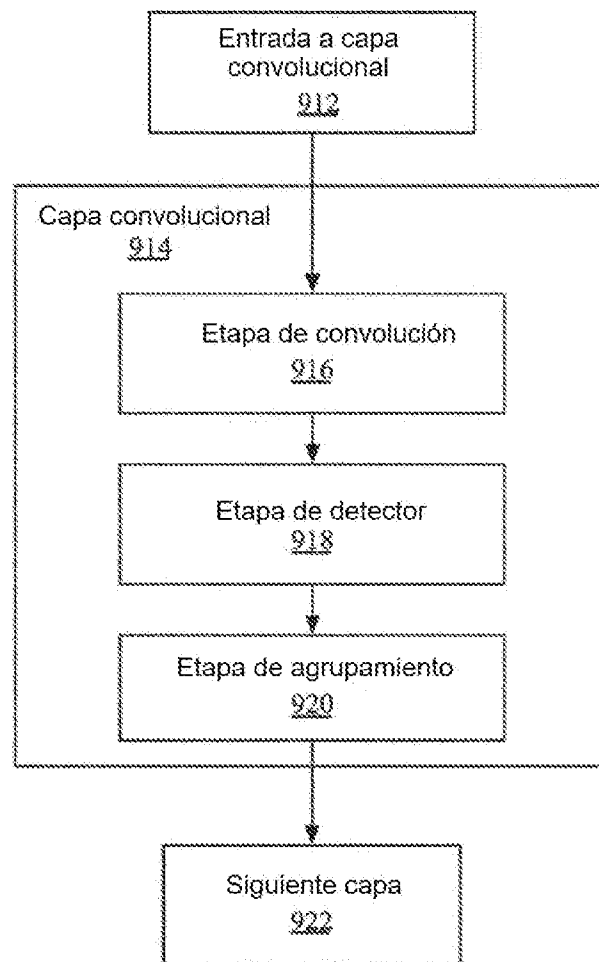


FIG. 9B

1000

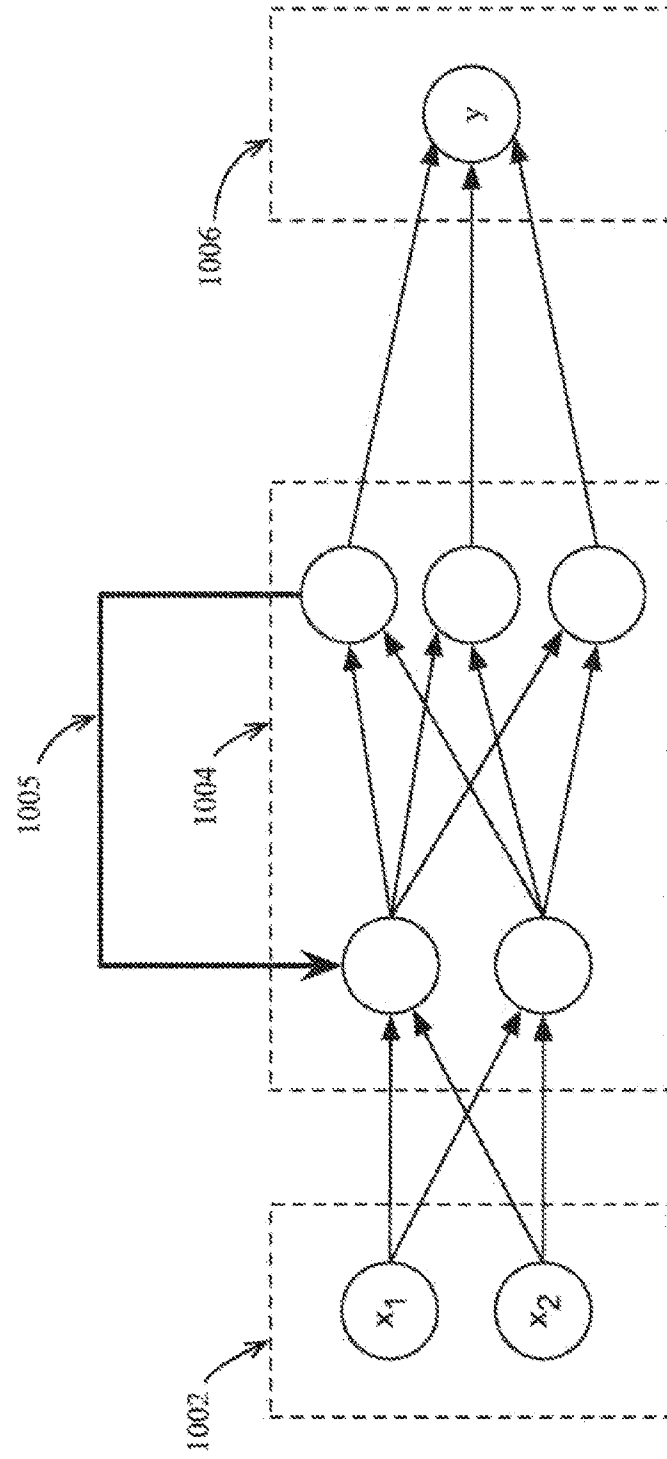


FIG. 10

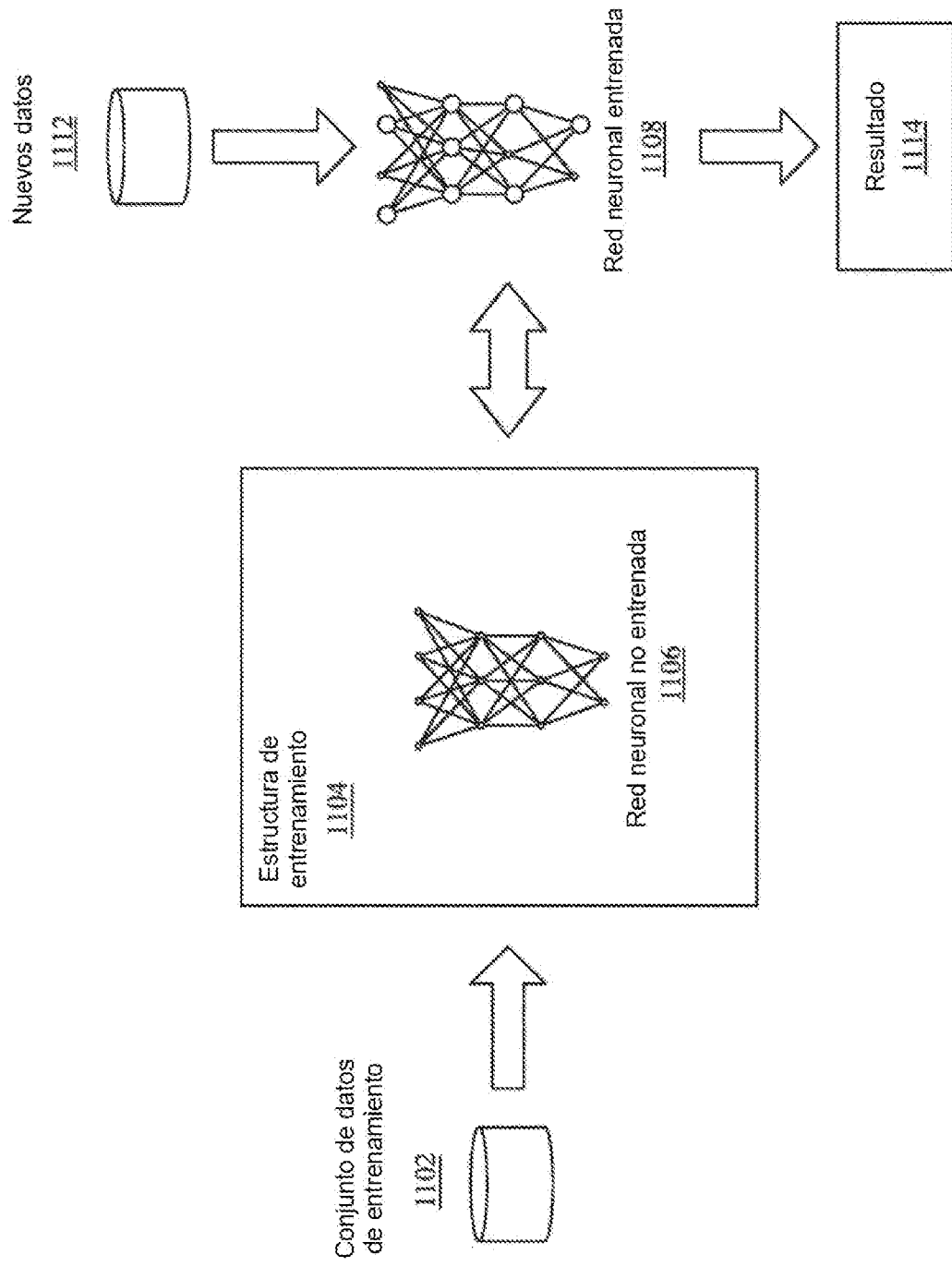


FIG. 11

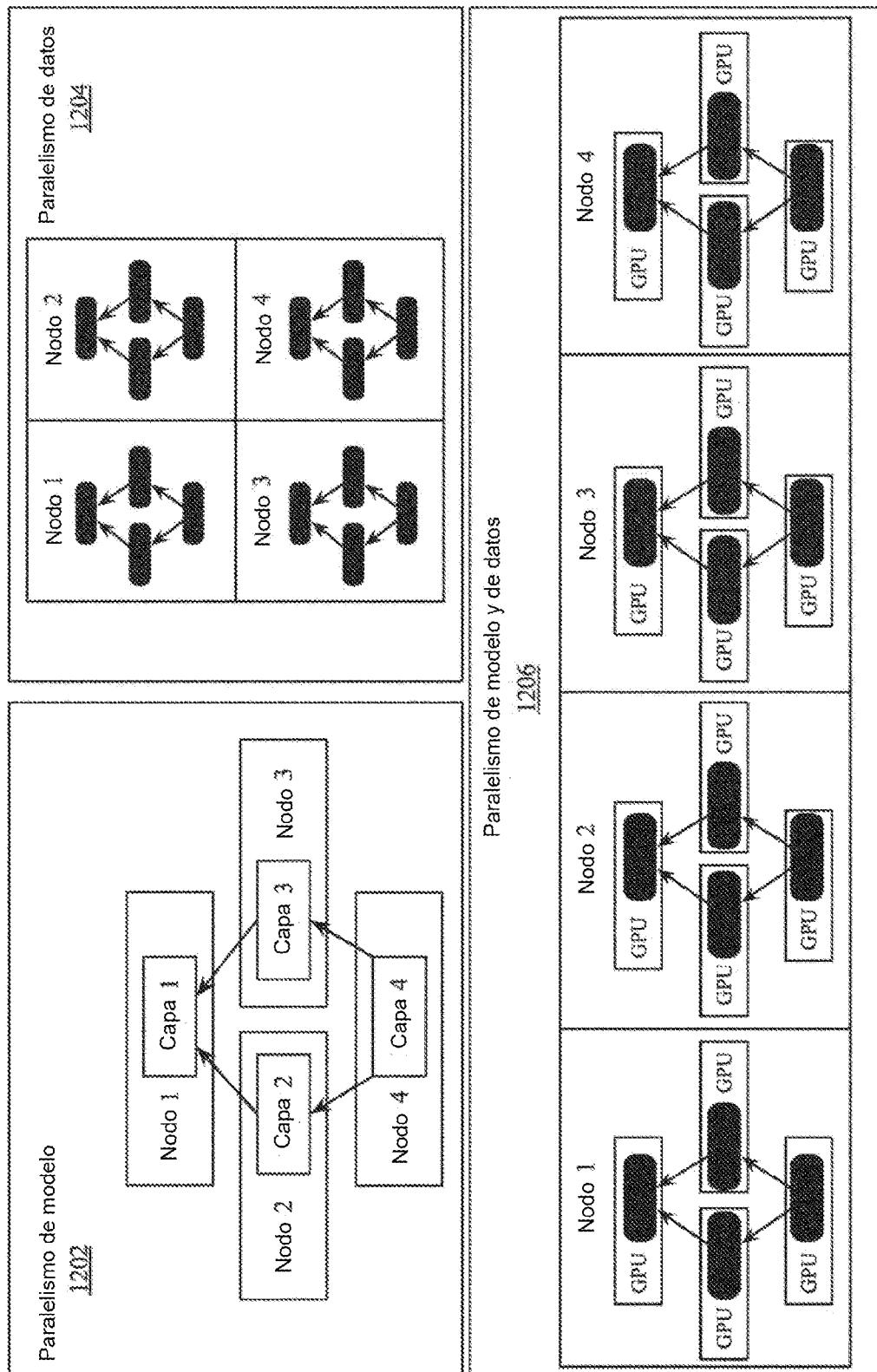


FIG. 12

1300

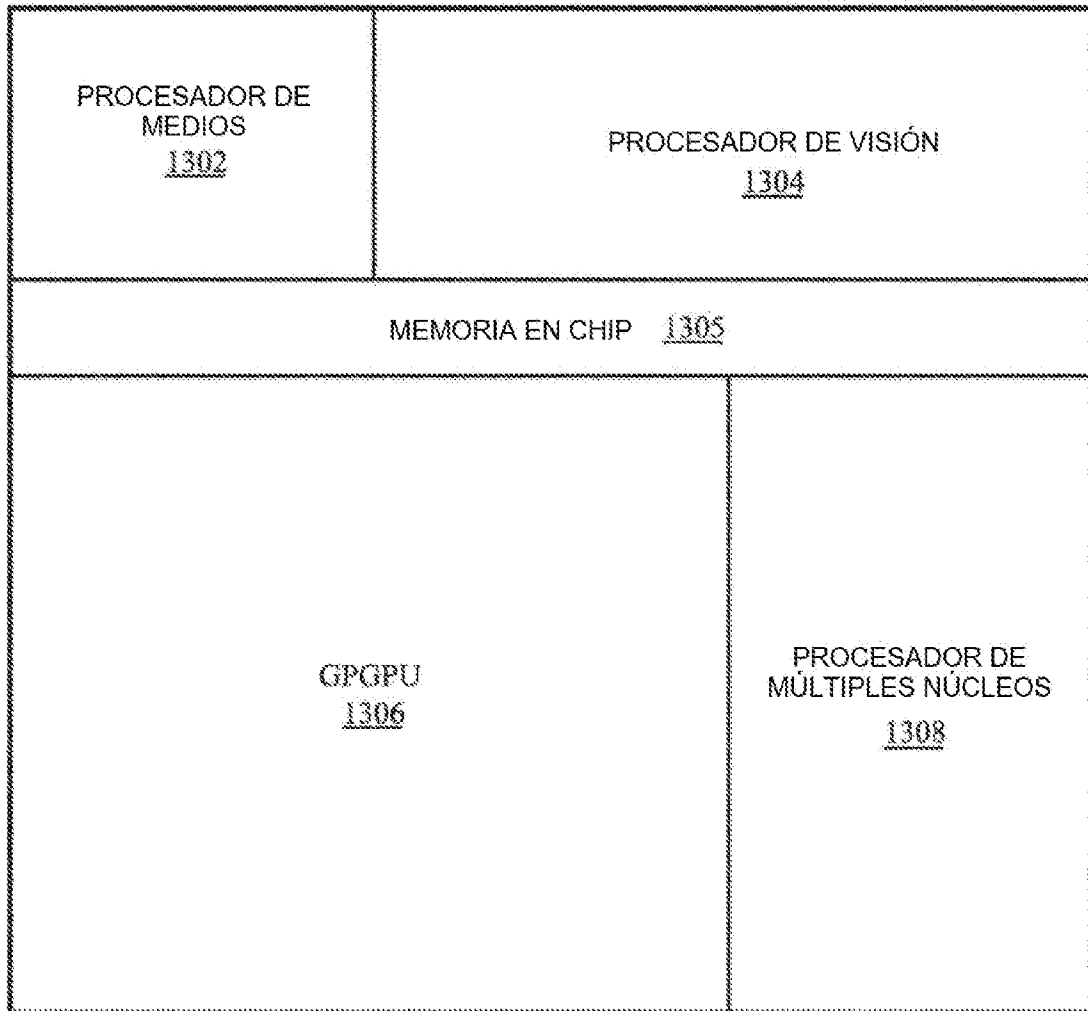


FIG. 13

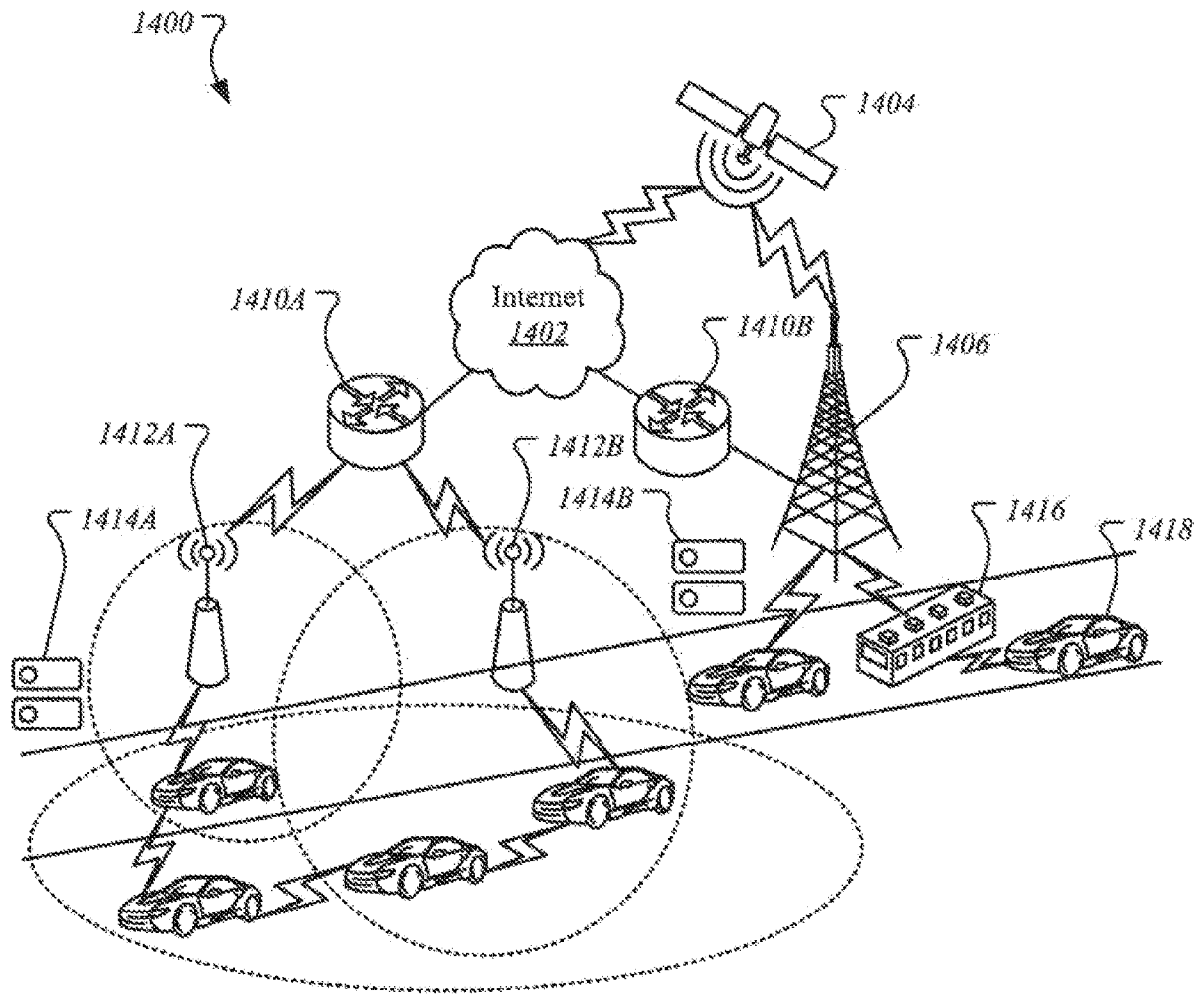


FIG. 14

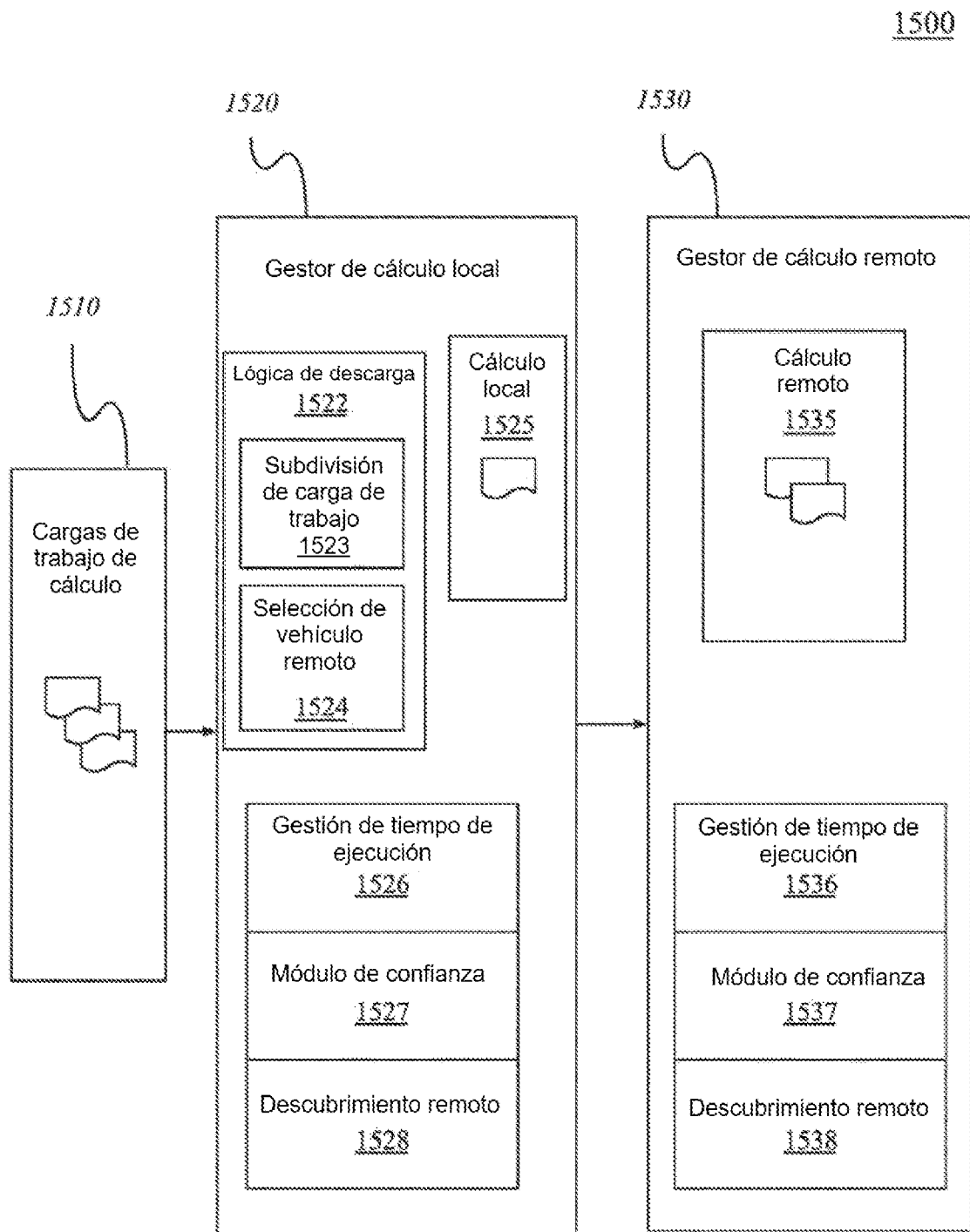


FIG. 15

1600

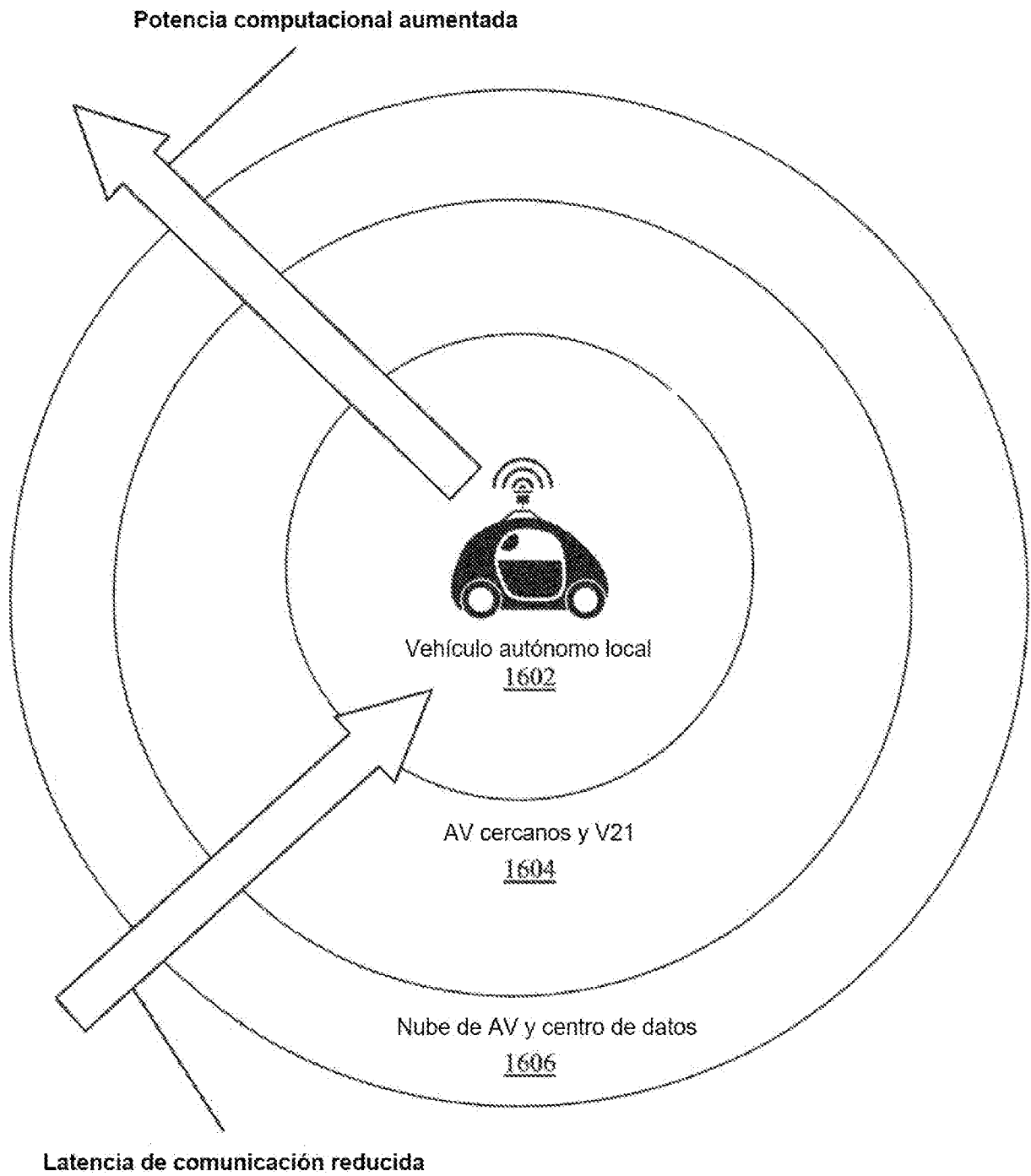
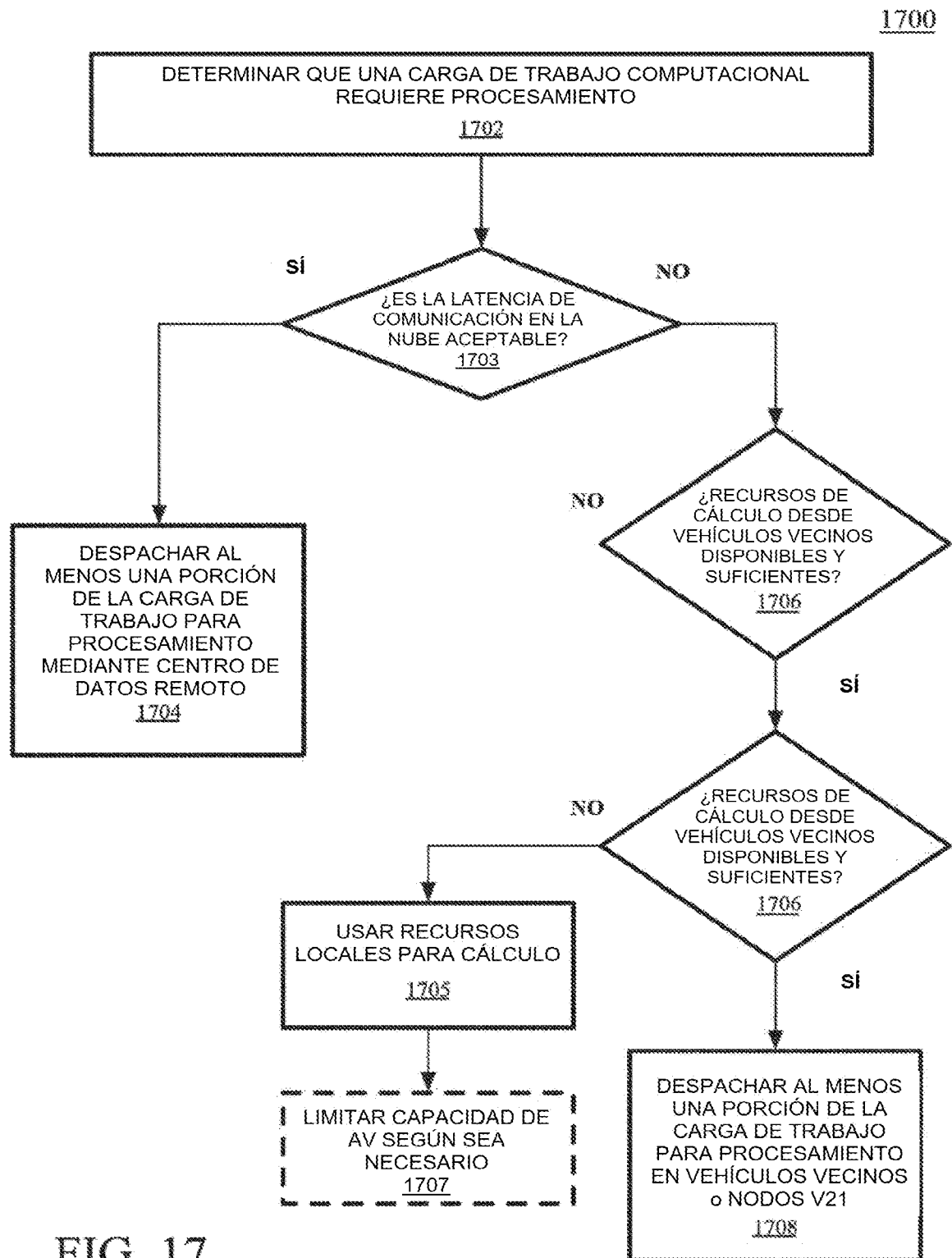


FIG. 16



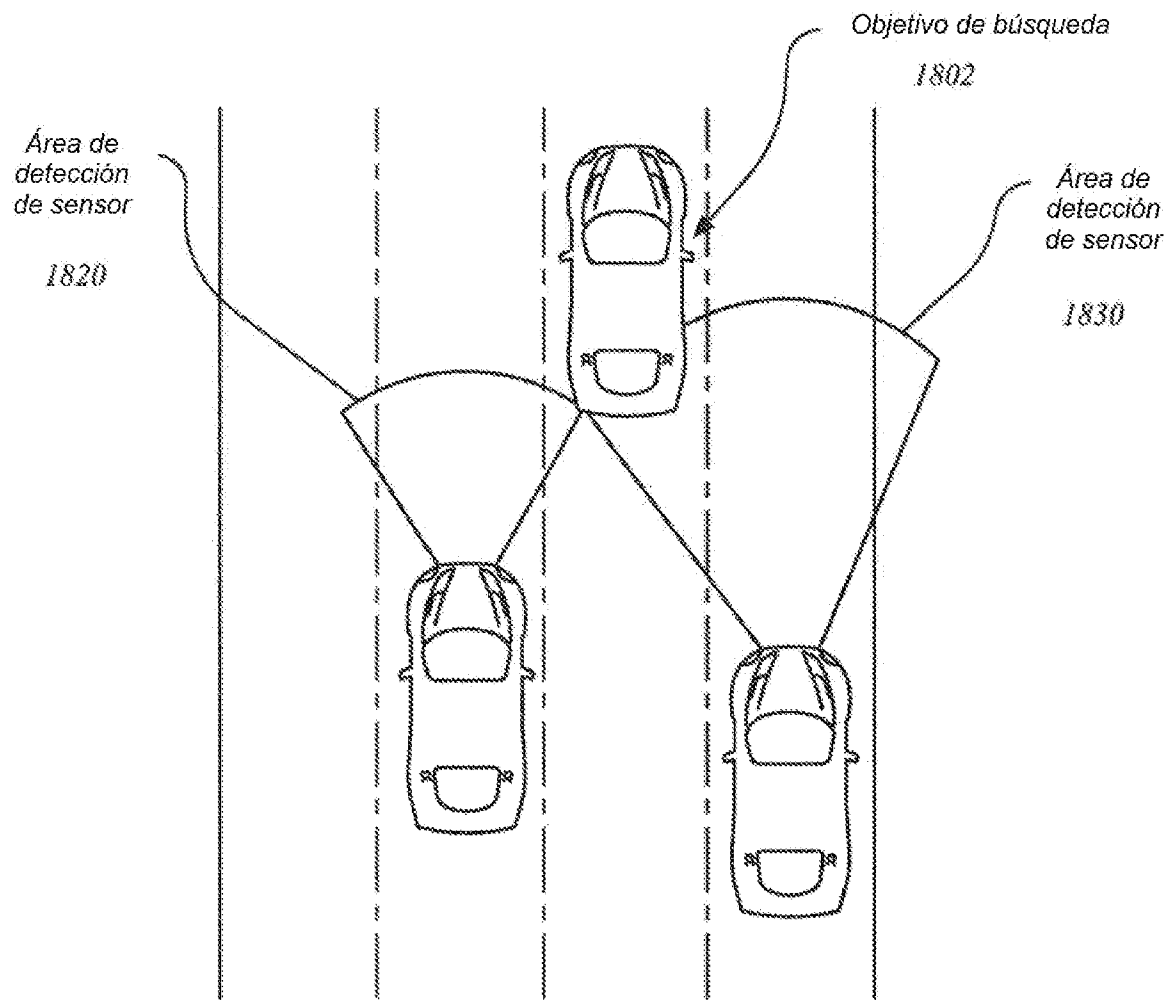
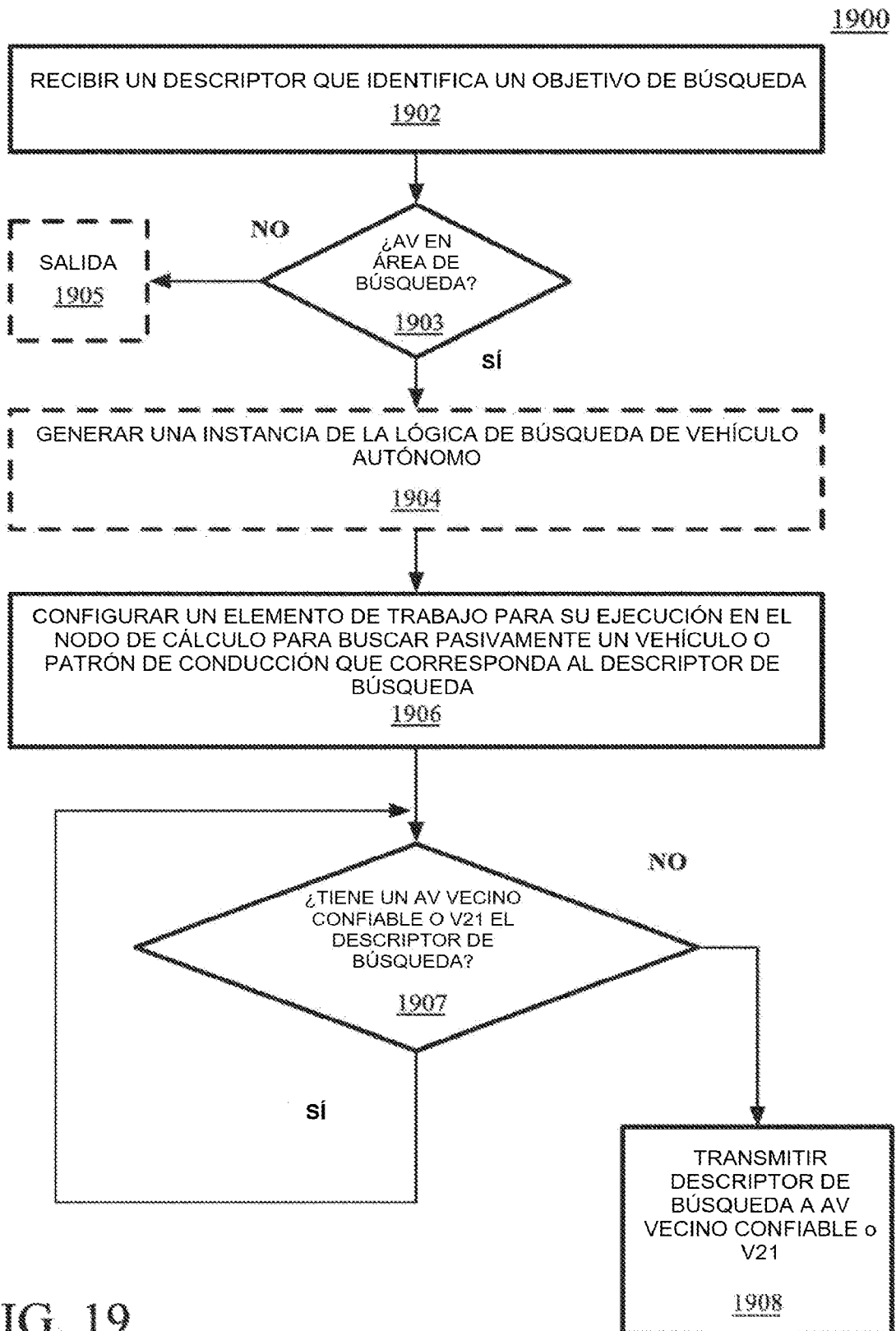


FIG. 18



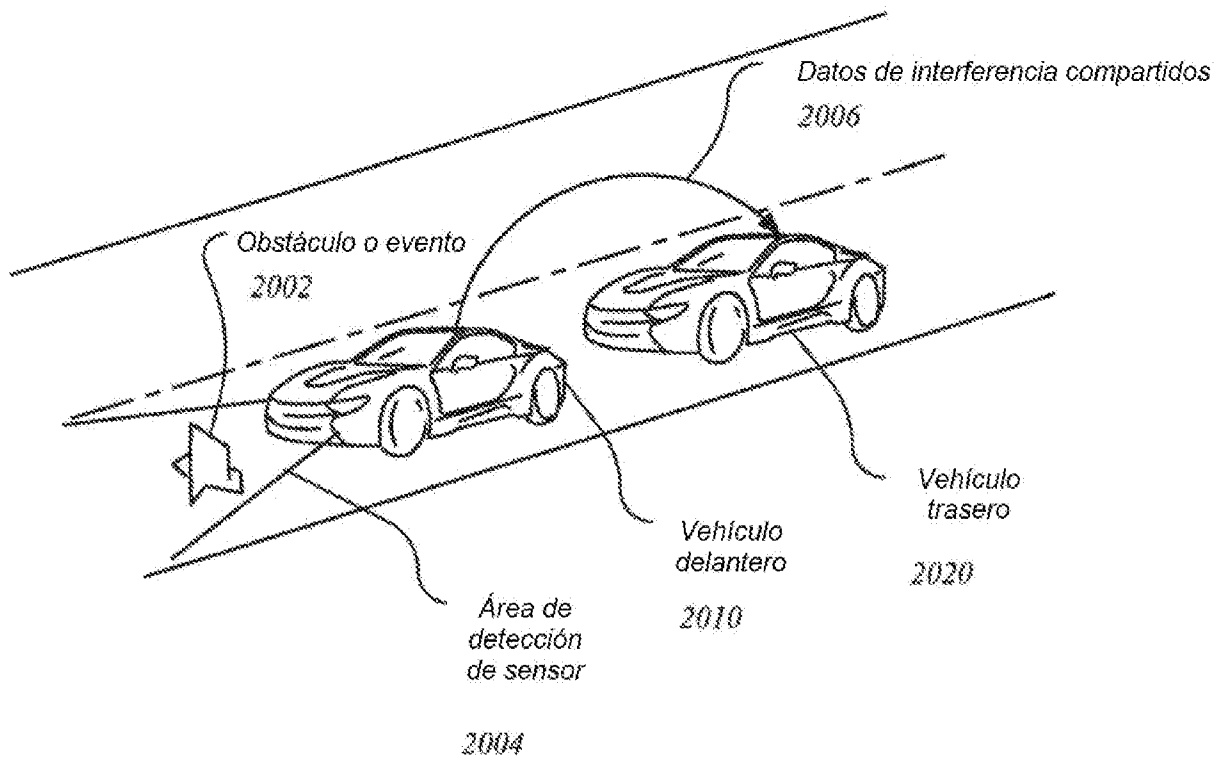


FIG. 20

2100

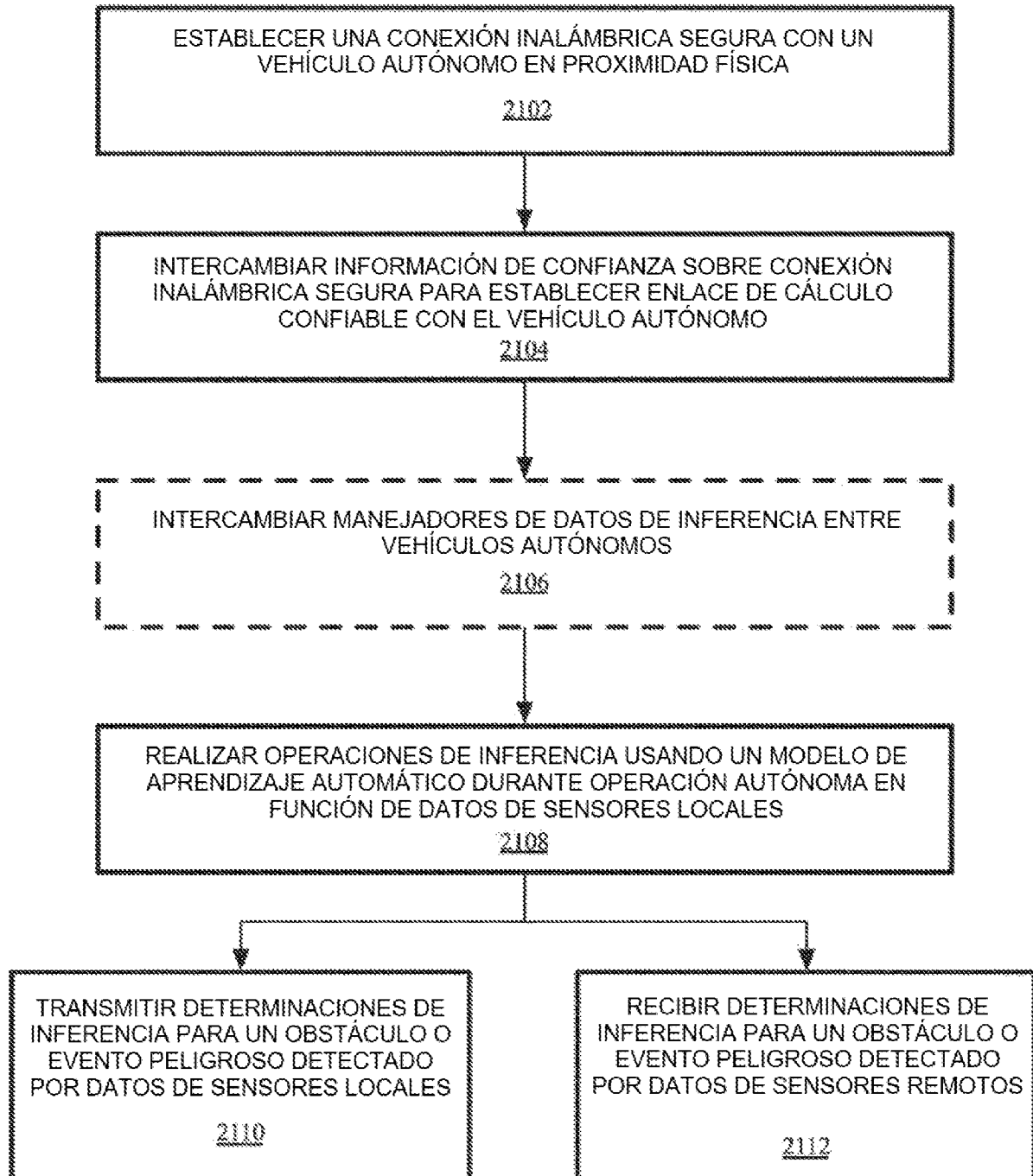


FIG. 21

2200

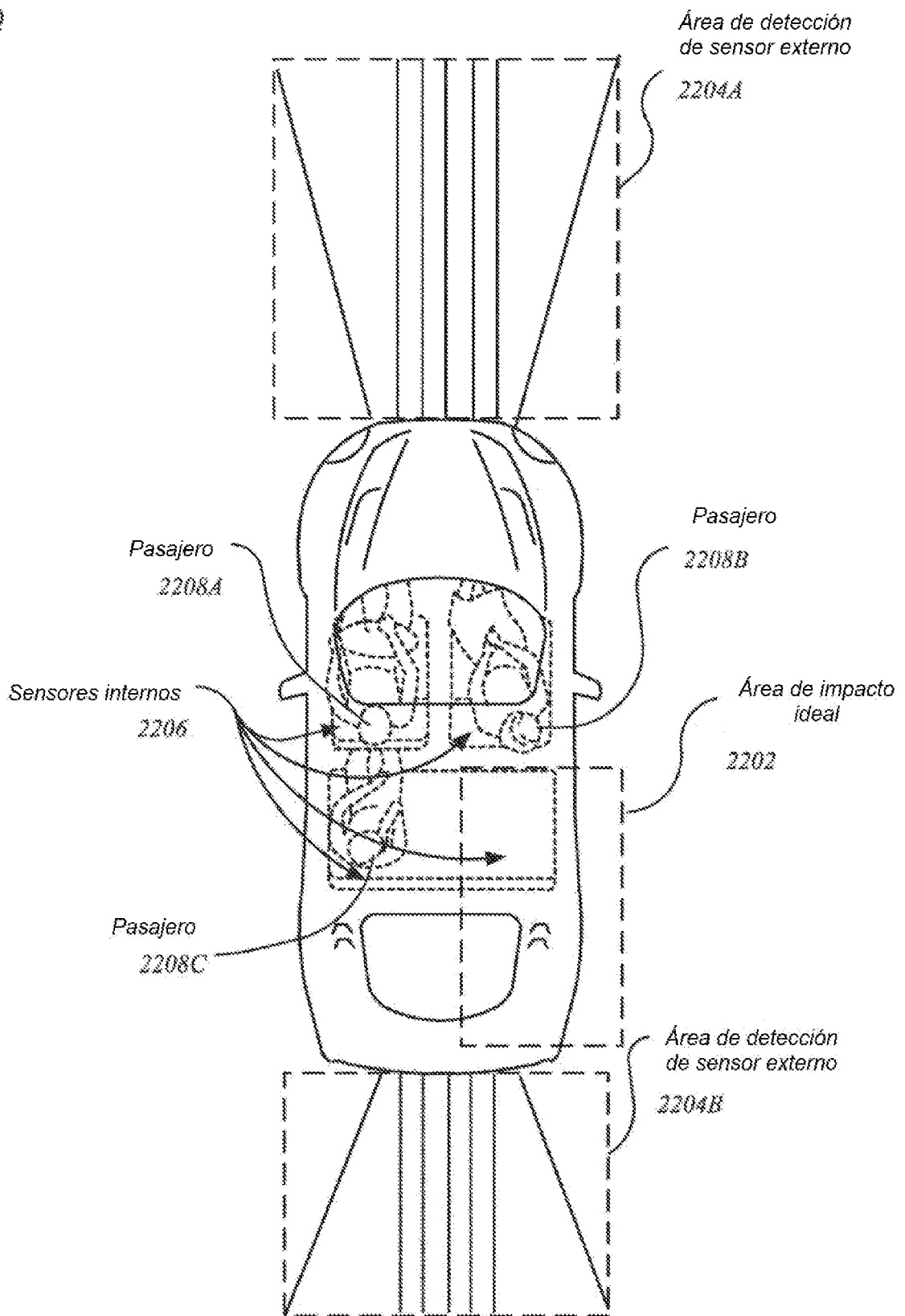


FIG. 22

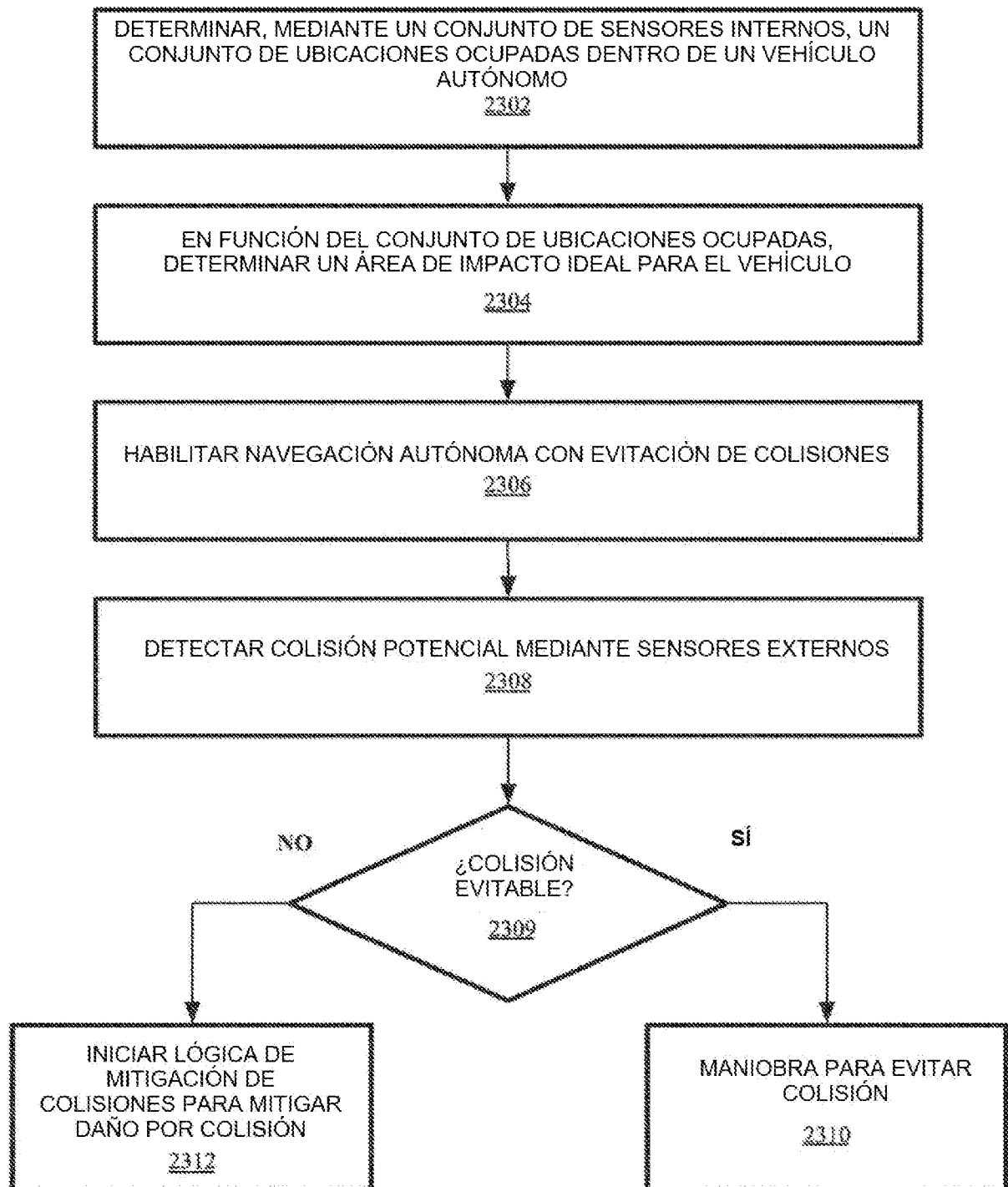
2300

FIG. 23

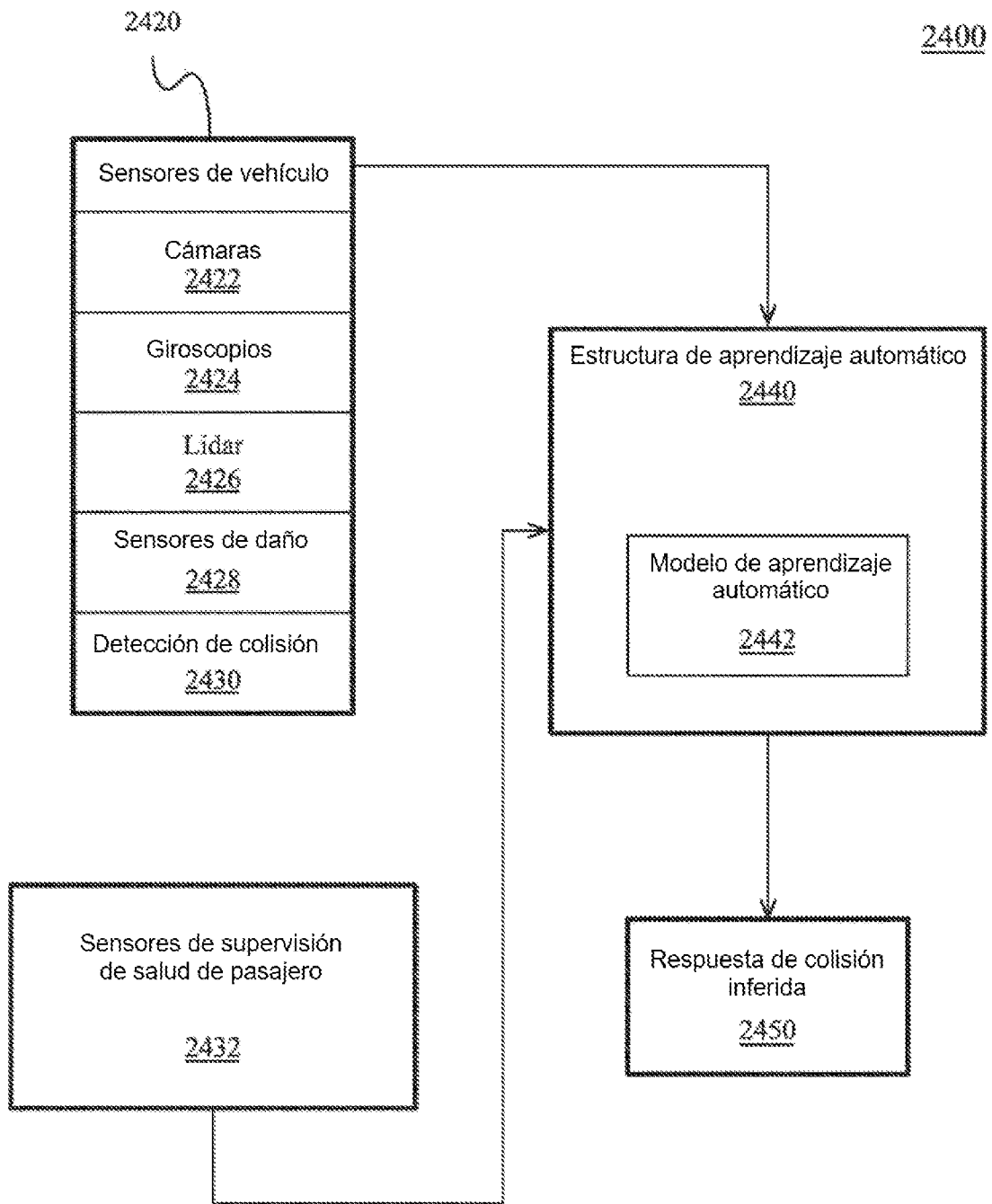


FIG. 24

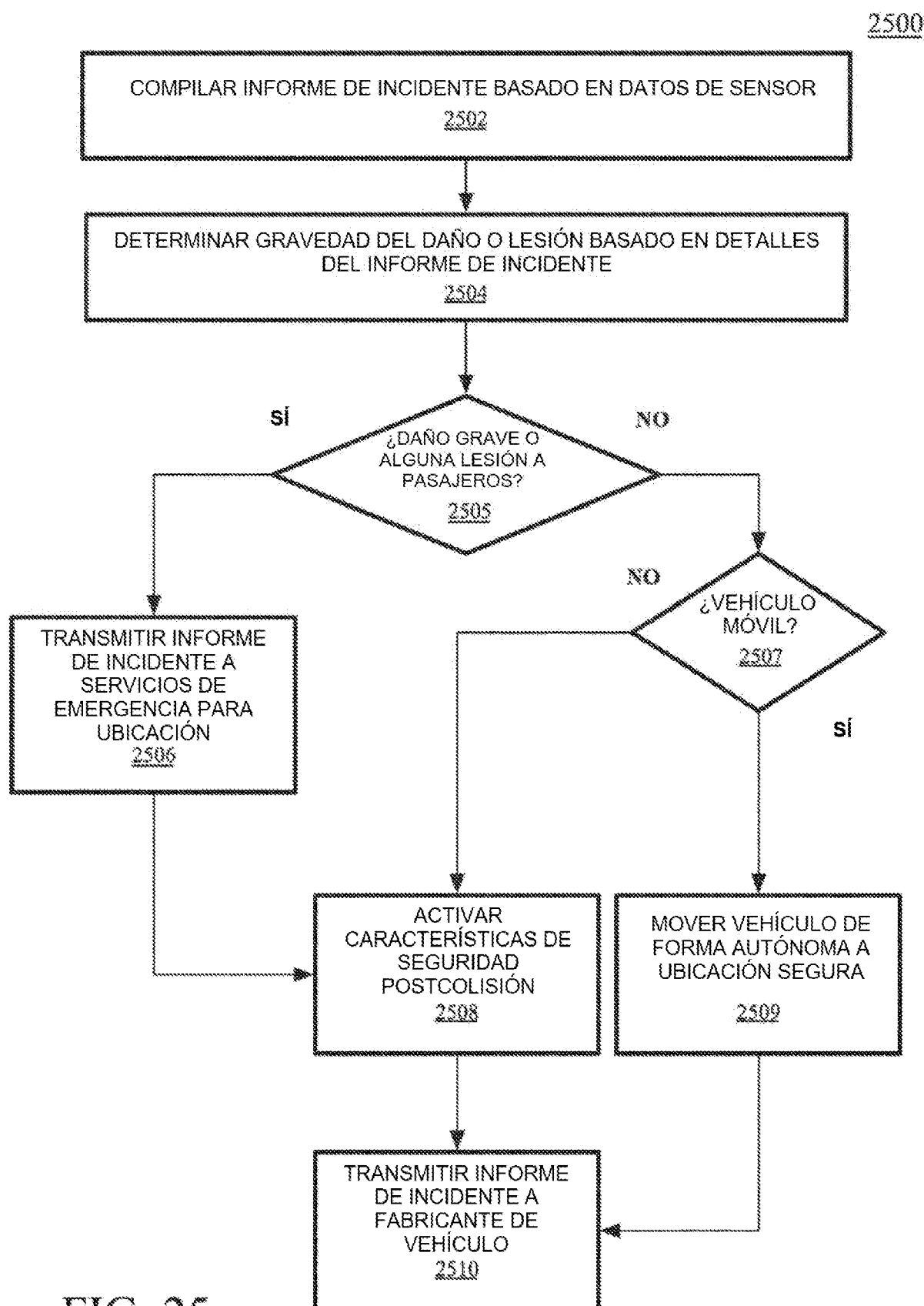
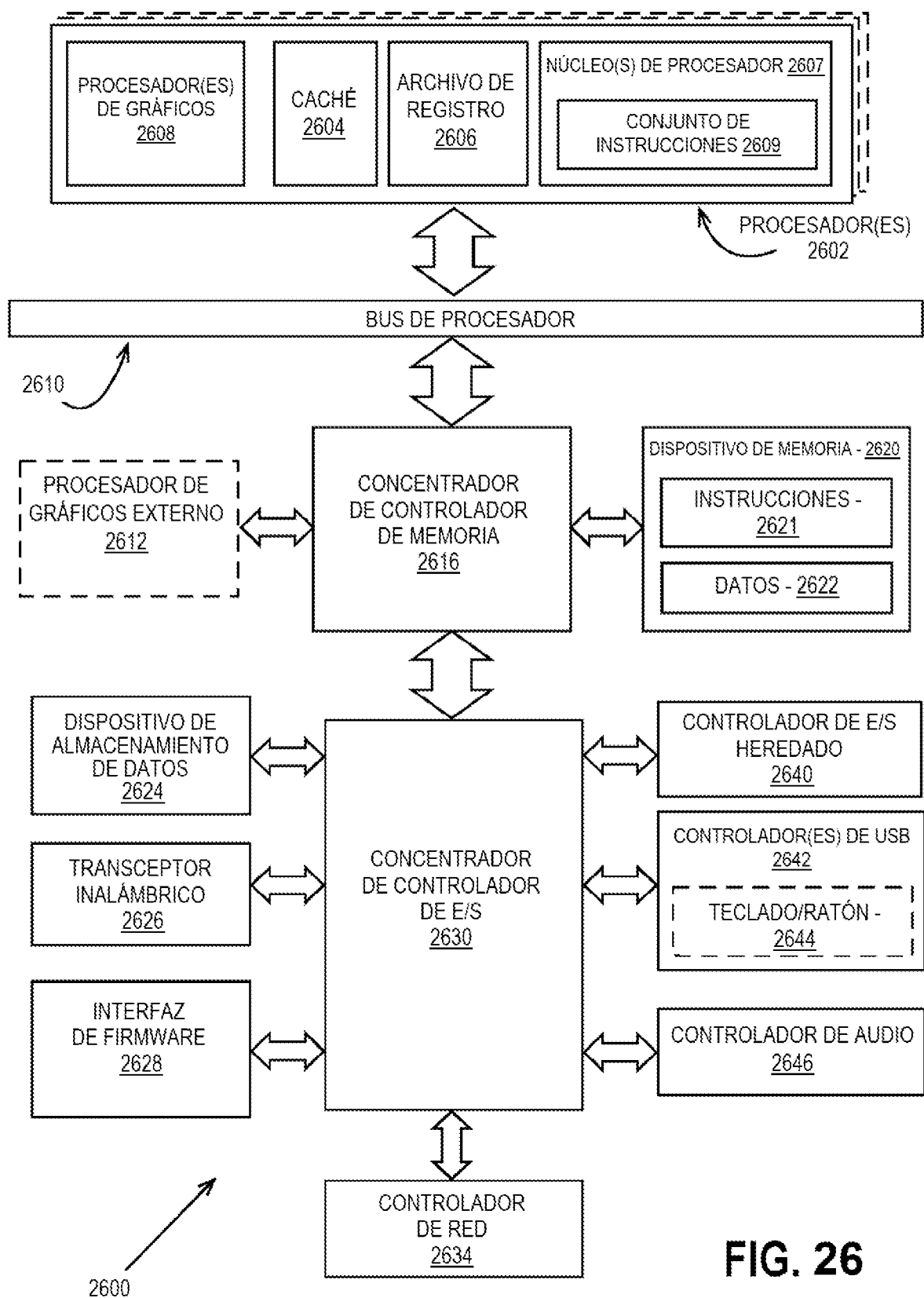


FIG. 25



PROCESADOR 2700

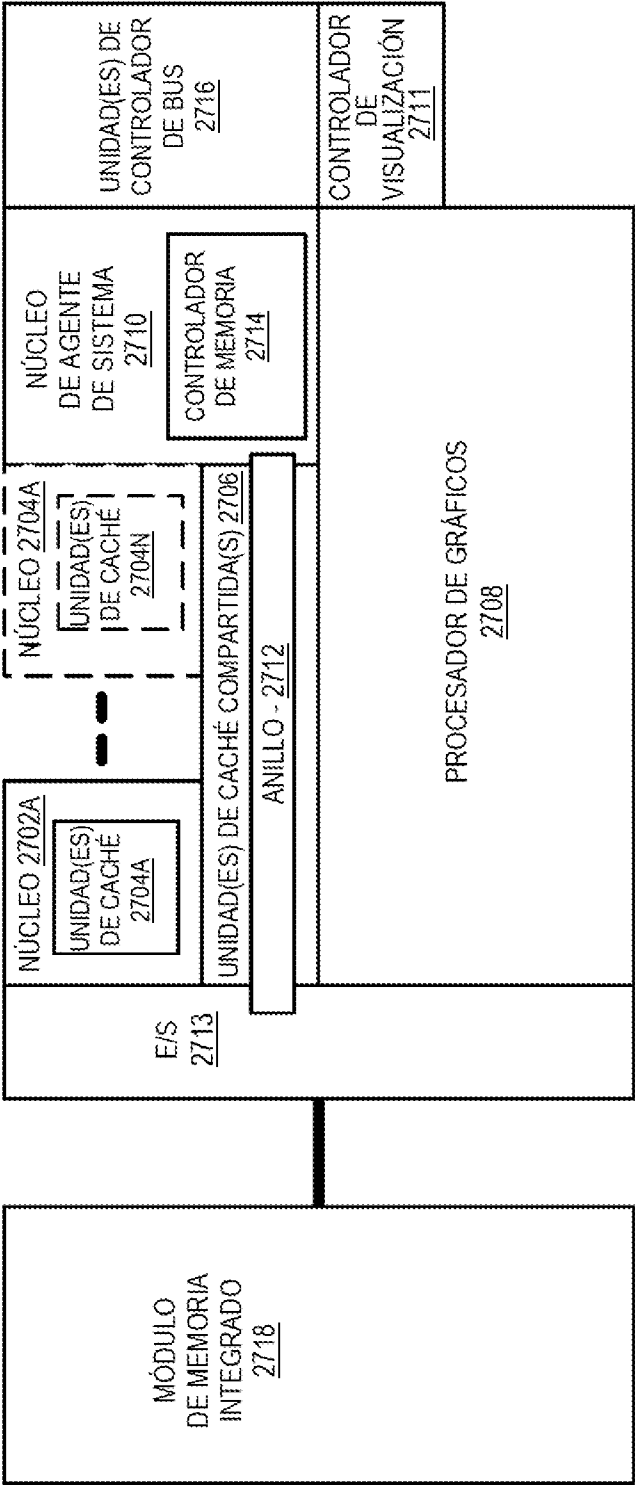
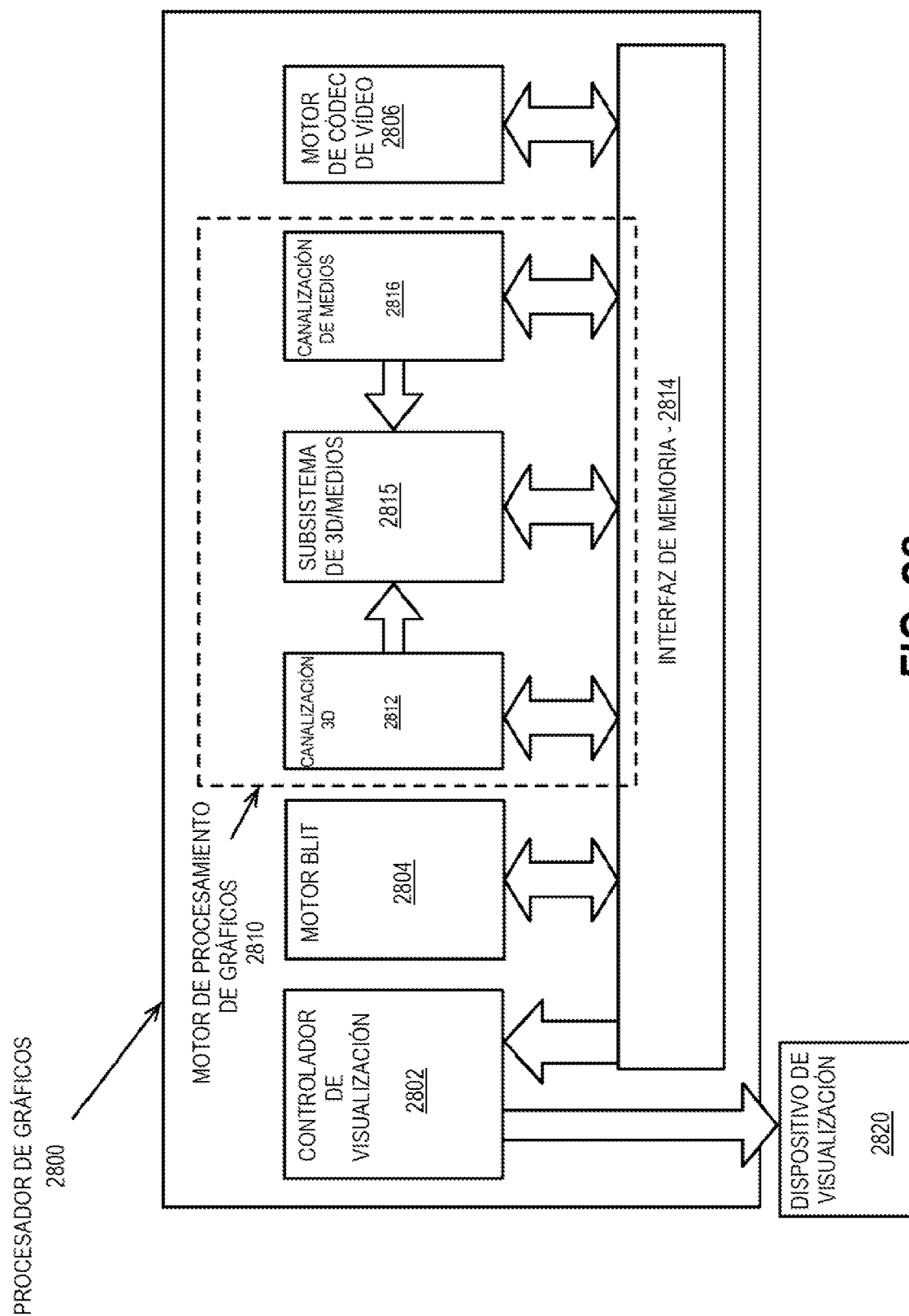


FIG. 27



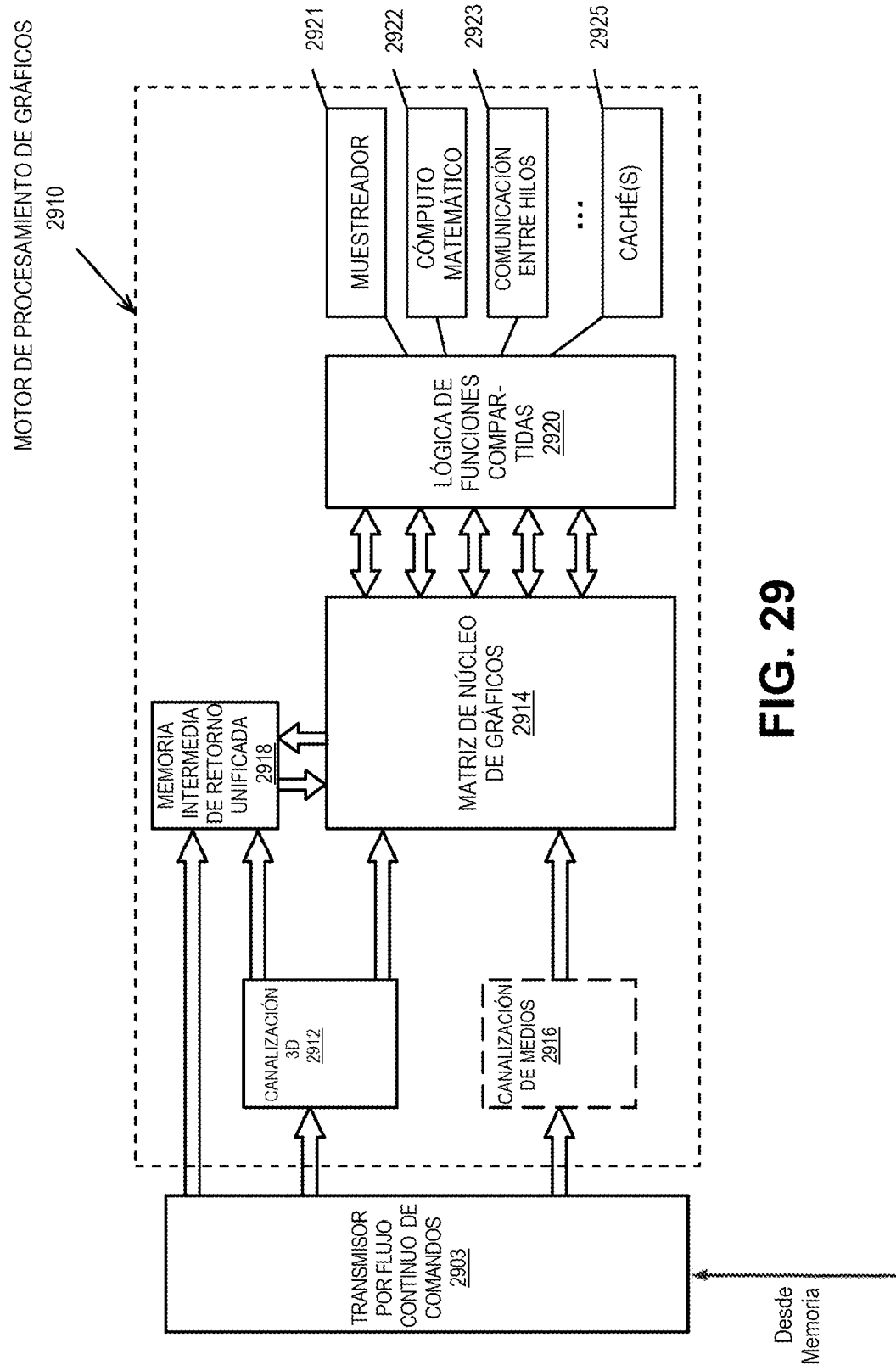


FIG. 29

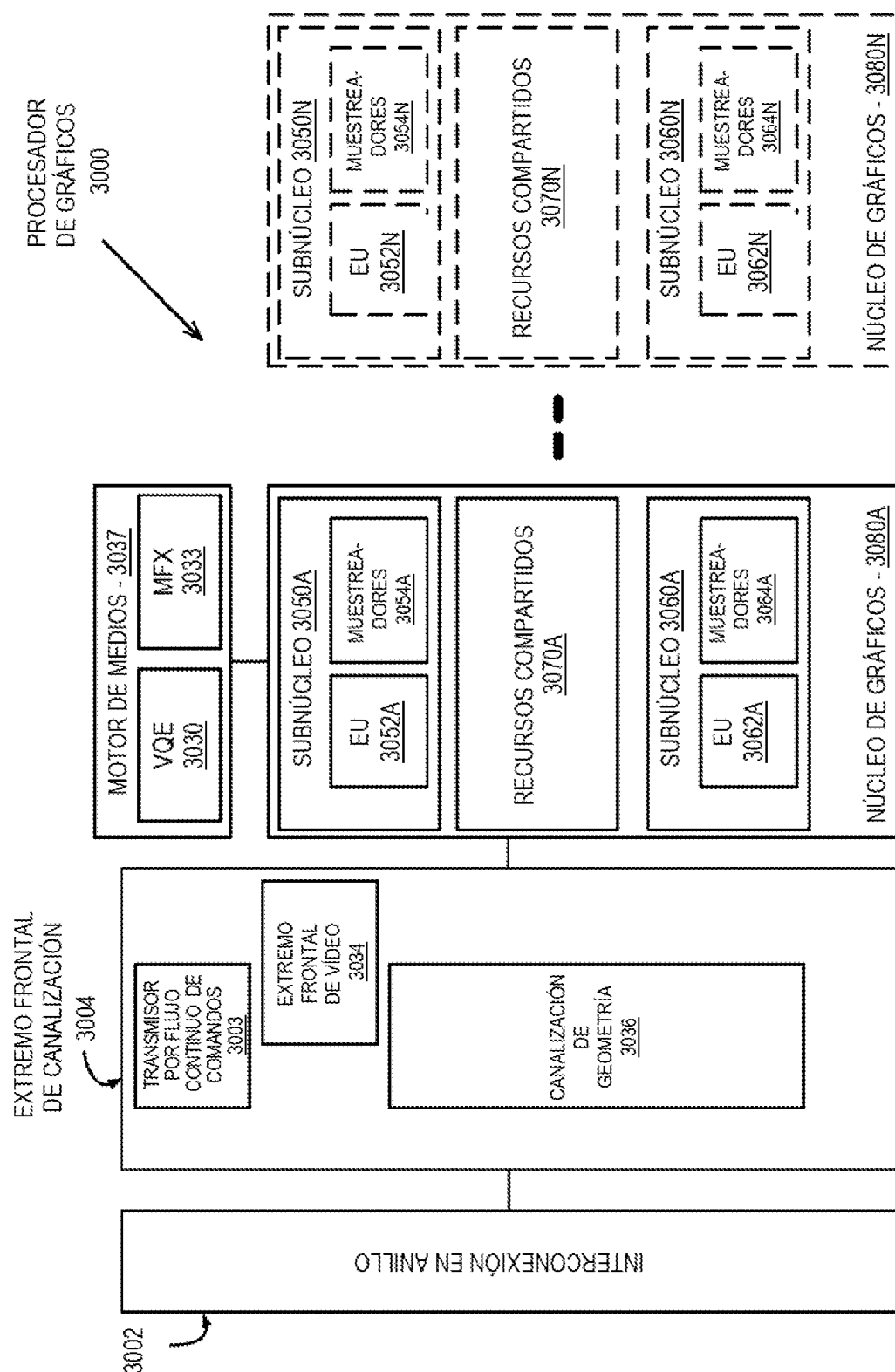


FIG. 30

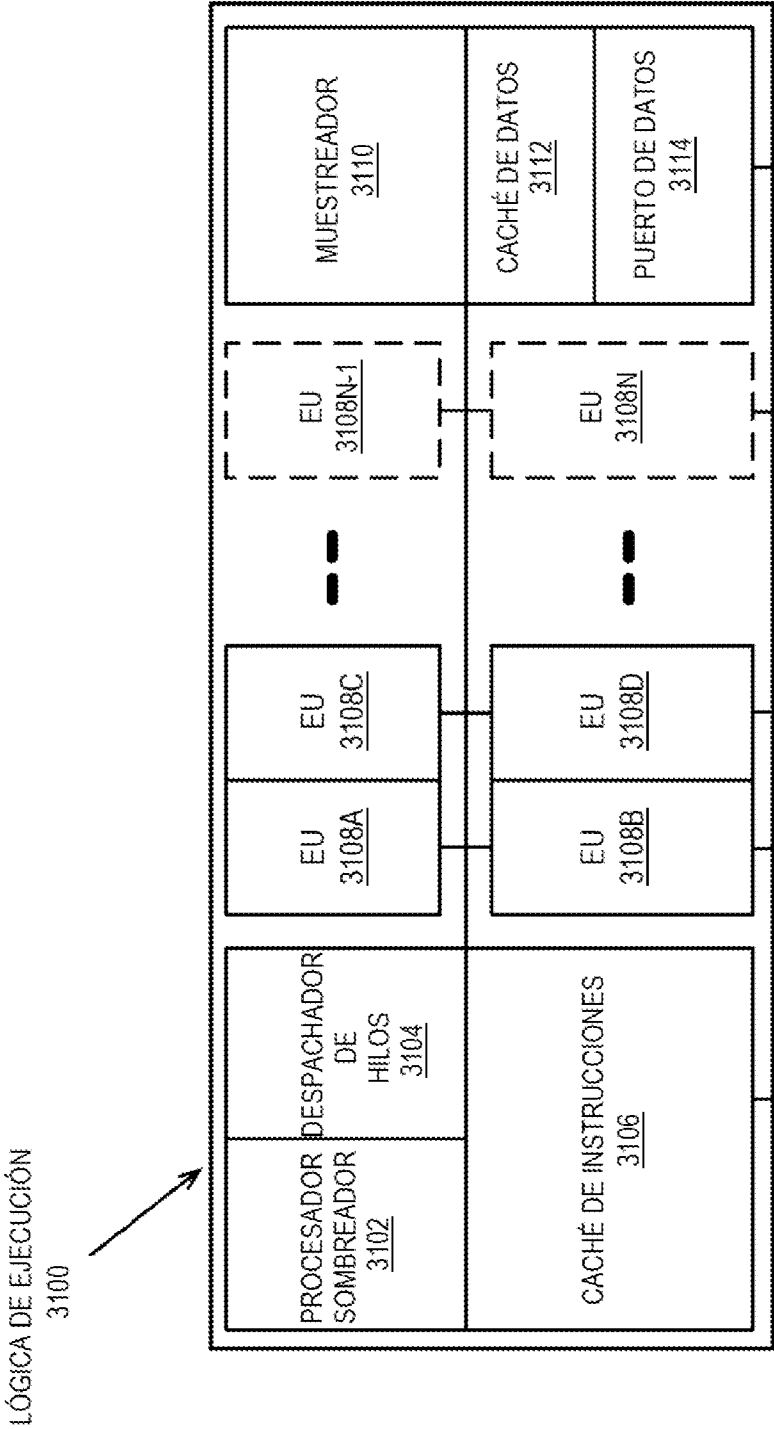


FIG. 31

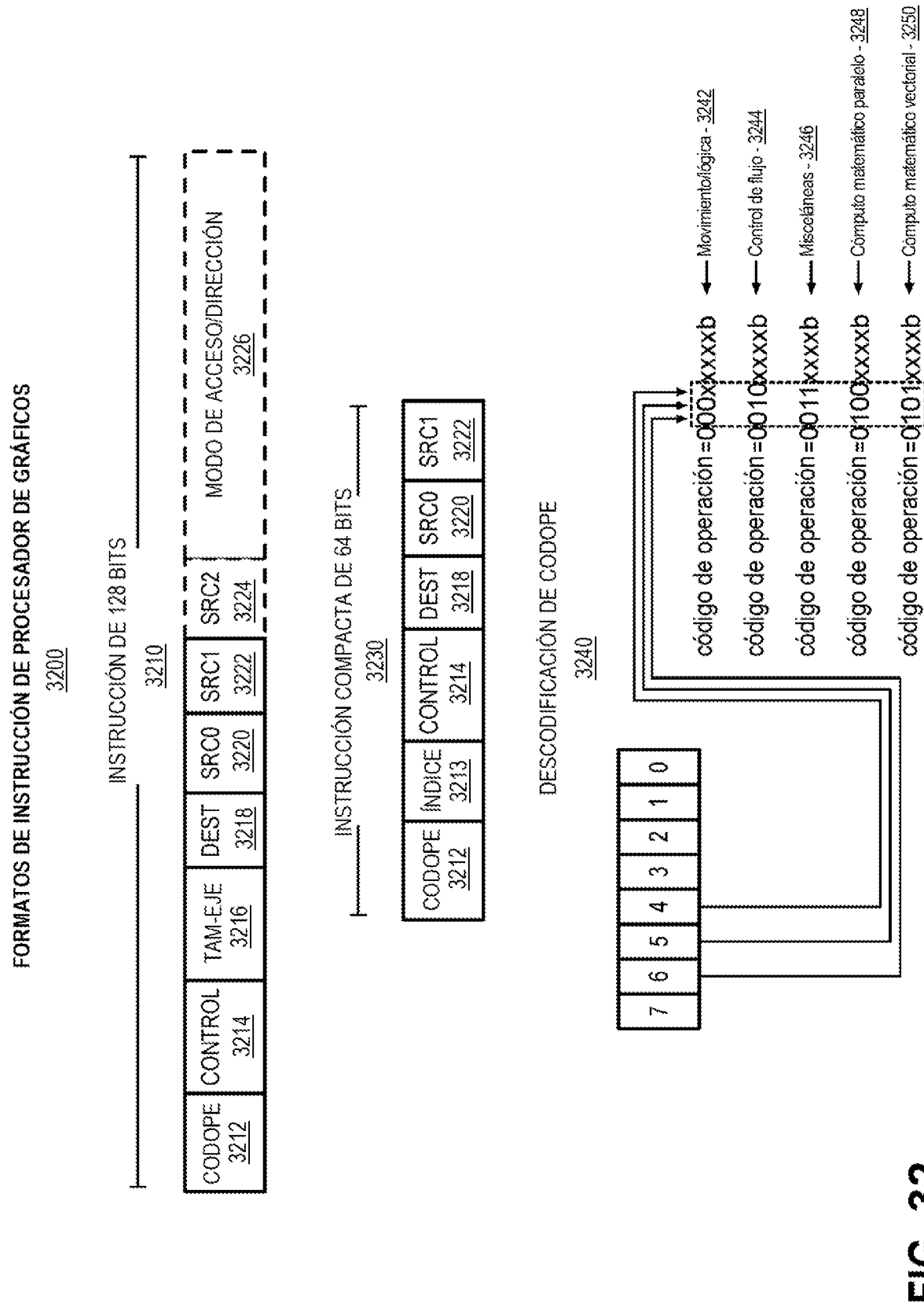


FIG. 32

PROCESADOR DE GRÁFICOS

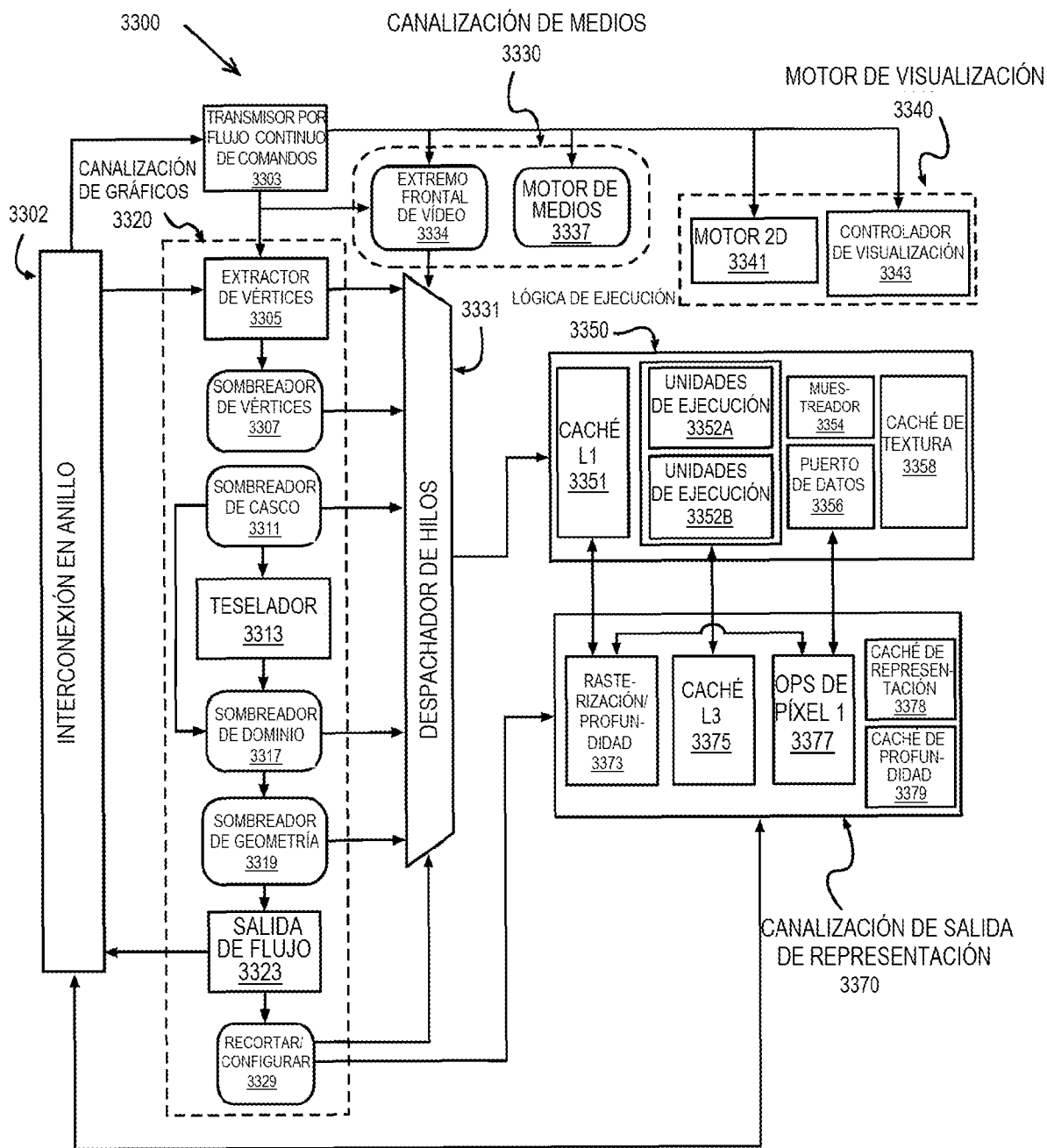


FIG. 33

FIG. 34A FORMATO DE COMANDOS DE PROCESADOR DE GRÁFICOS

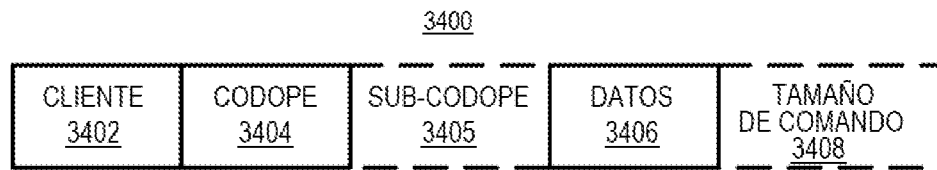
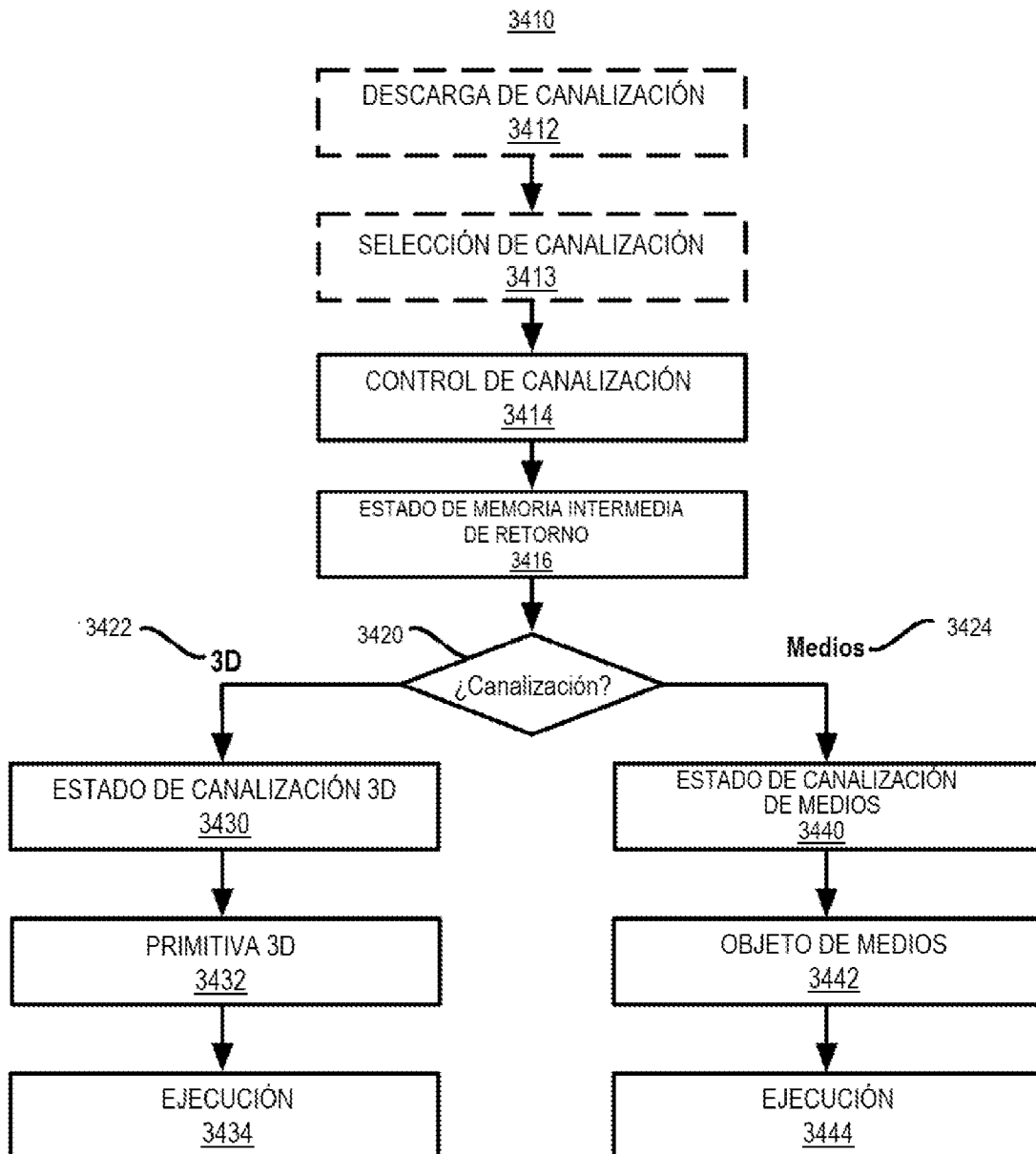


FIG. 34B SECUENCIA DE COMANDOS DE PROCESADOR DE GRÁFICOS



SISTEMA DE PROCESAMIENTO DE DATOS - 3500

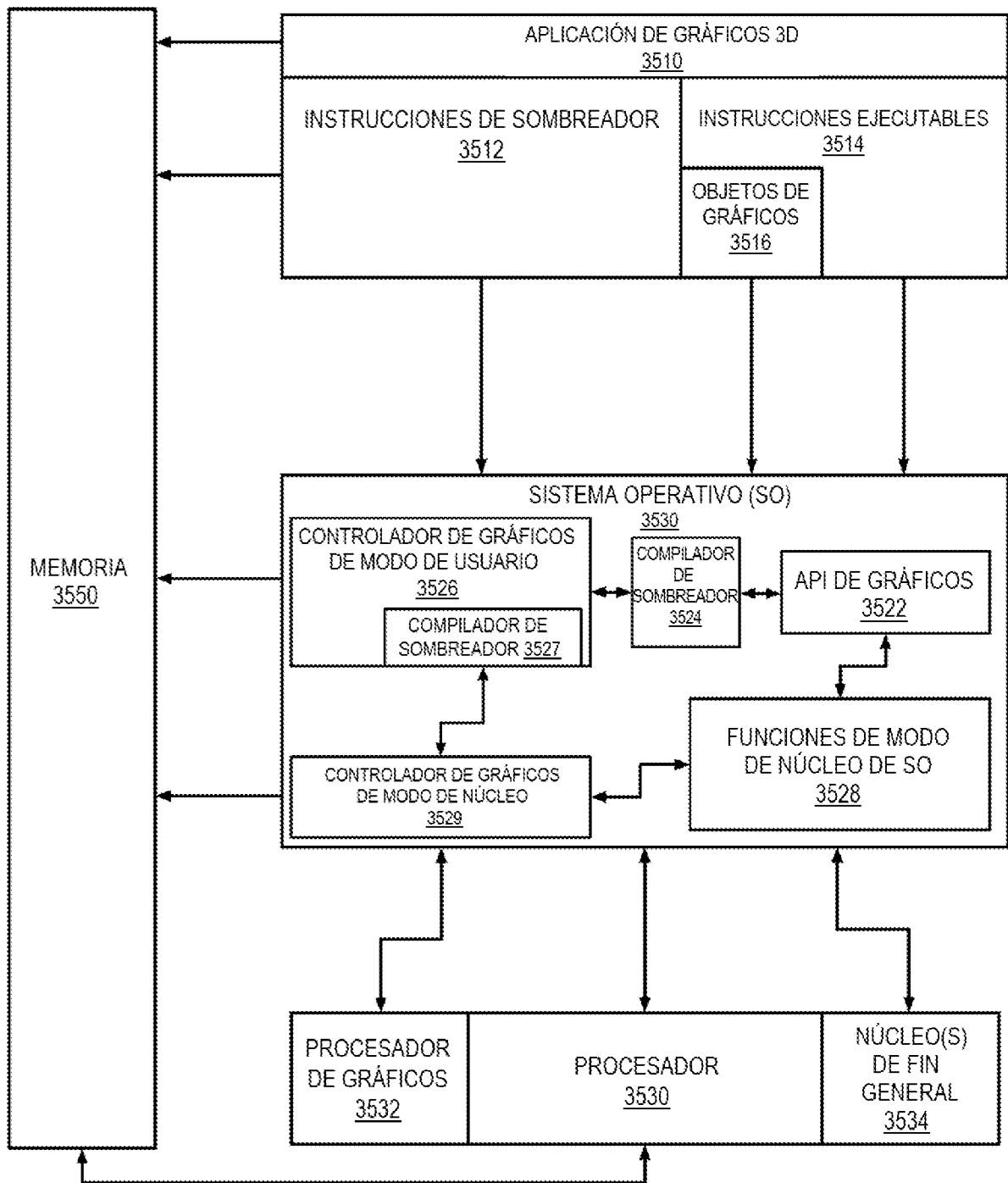


FIG. 35

DESARROLLO DE NÚCLEO DE IP - 3600

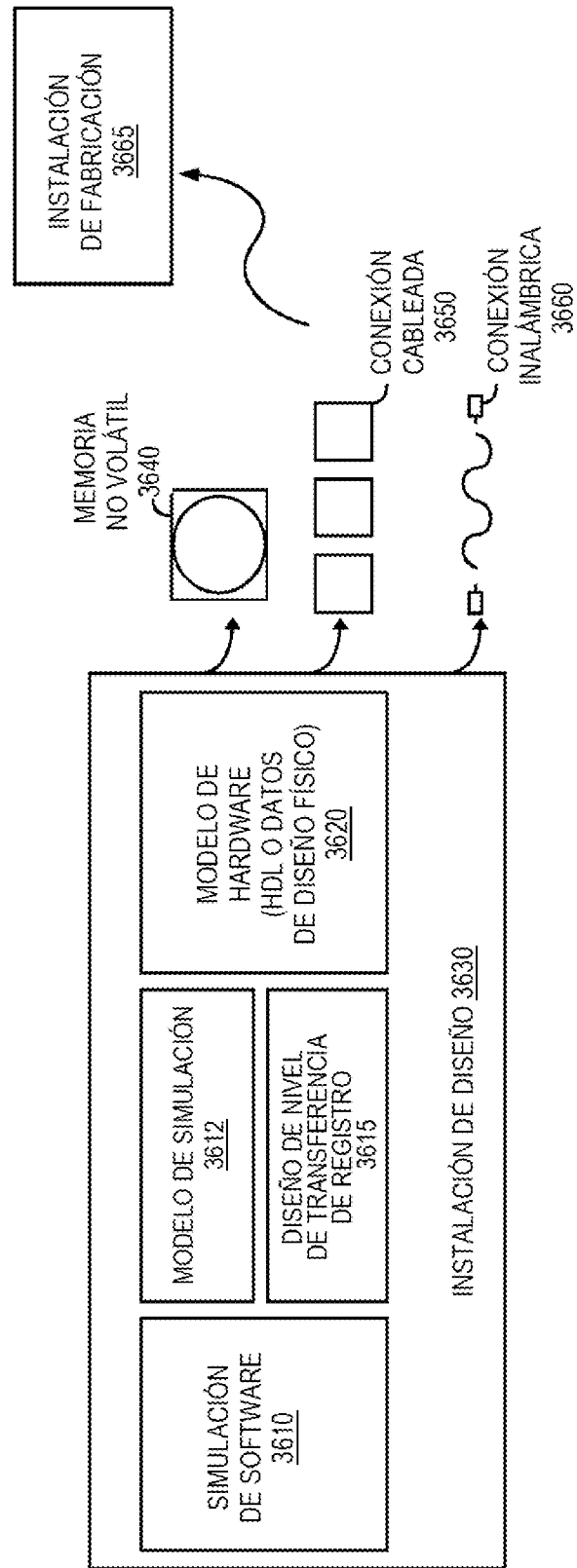


FIG. 36

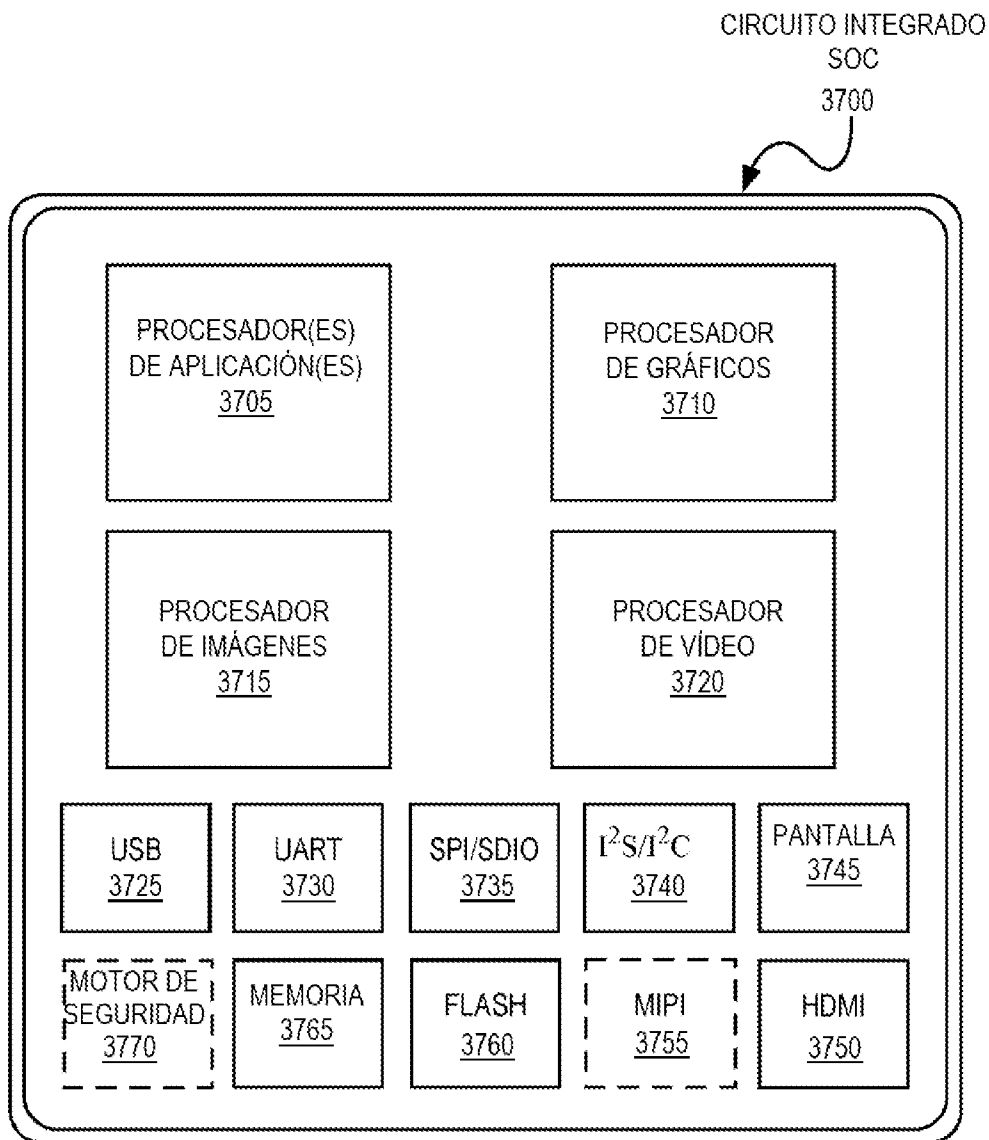


FIG. 37

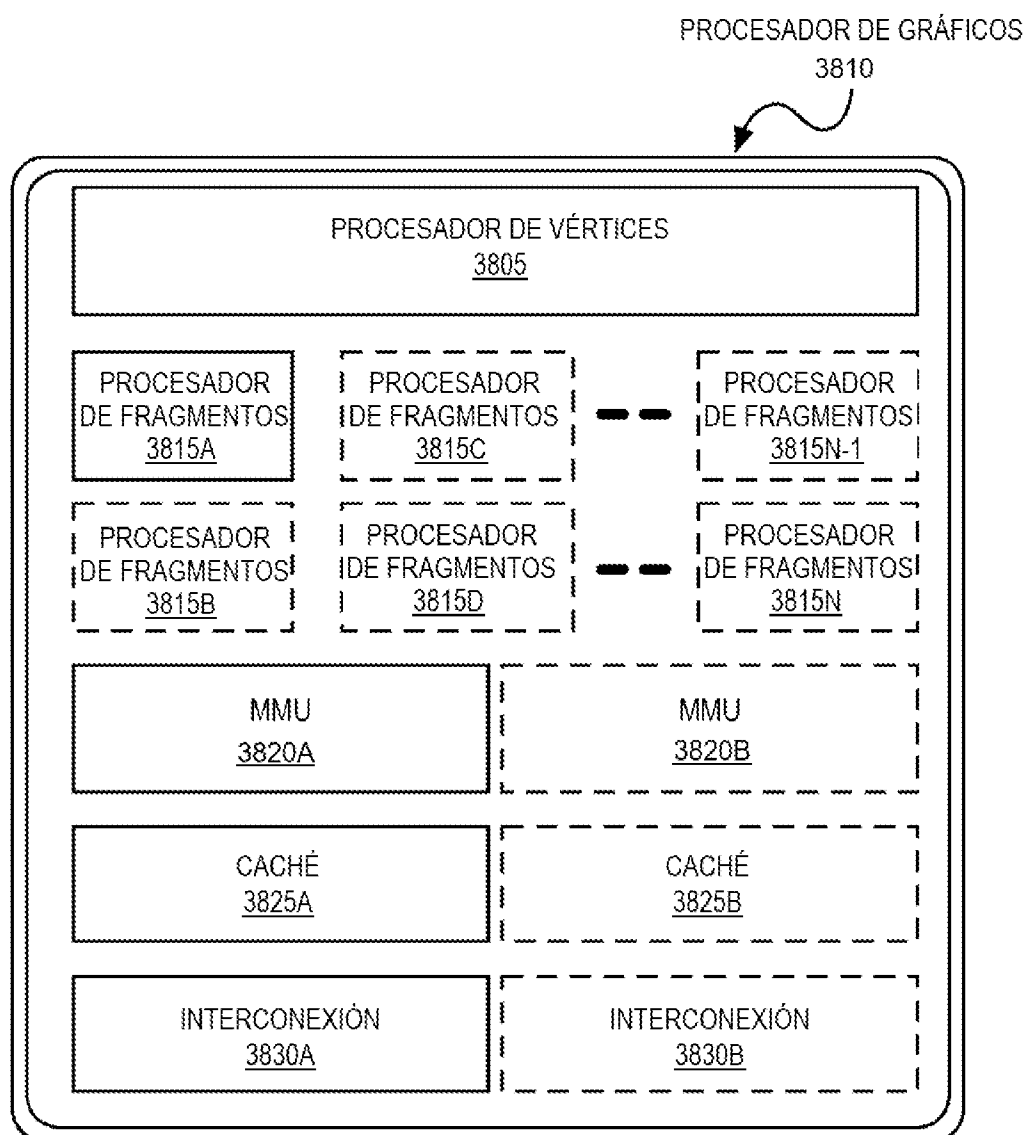


FIG. 38

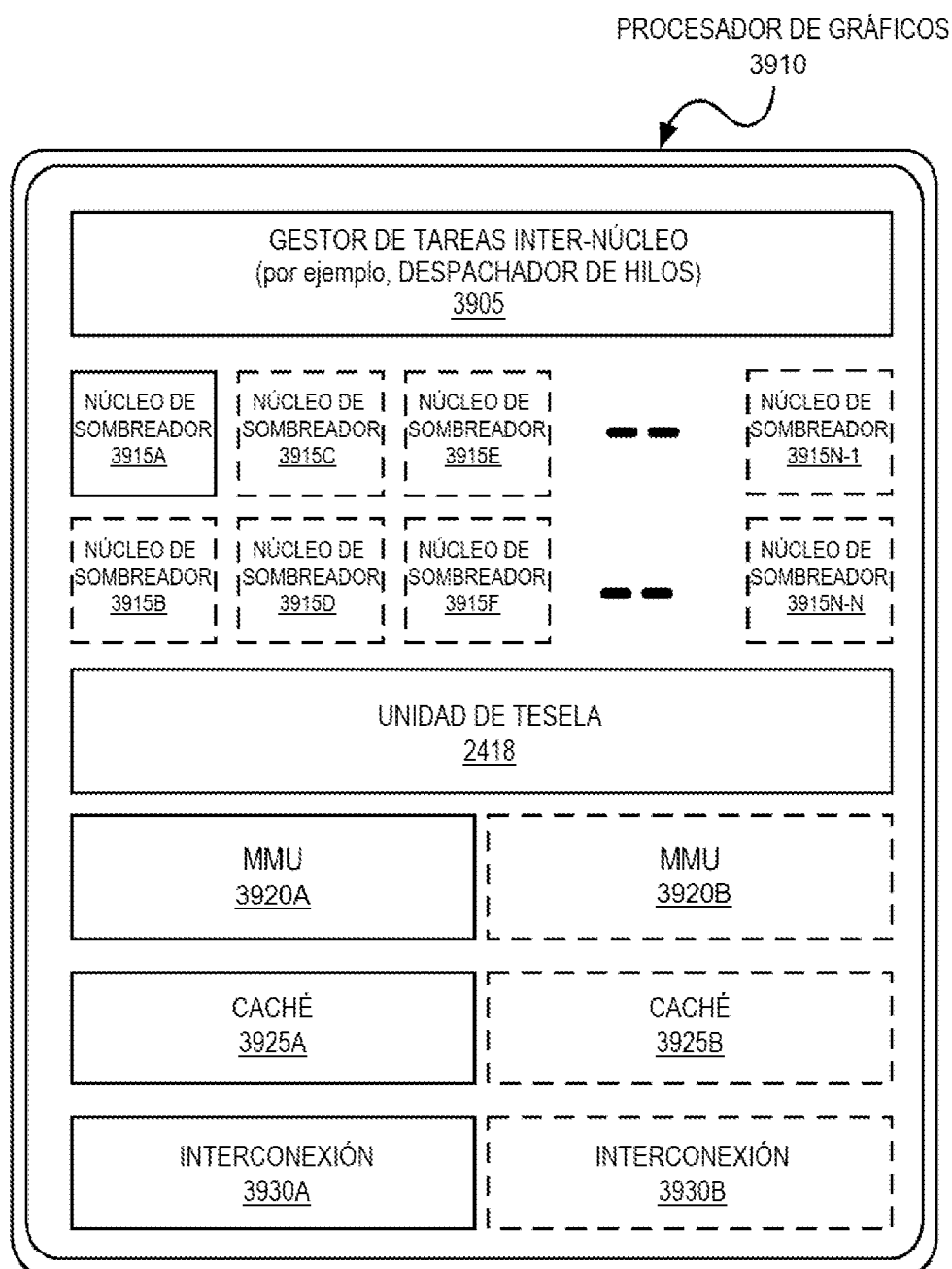


FIG. 39