(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 617 303**

(51) Int. CI.:

G06F 9/54 (2006.01)
G06F 9/38 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **19.09.2011 PCT/US2011/052196**

(87) Fecha y número de publicación internacional: **29.03.2012 WO2012040121**

(96) Fecha de presentación y número de la solicitud europea: **19.09.2011 E 11764399 (9)**

(97) Fecha y número de publicación de la concesión europea: **23.11.2016 EP 2619666**

(54) Título: **Técnicas de comunicación entre procesadores en una plataforma informática de múltiples procesadores**

(30) Prioridad:

16.09.2011 US 201113235266
20.09.2010 US 384571 P
04.08.2011 US 201161515182 P
16.09.2011 US 201113235236

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
16.06.2017

(73) Titular/es:

QUALCOMM INCORPORATED (100.0%)
5775 Morehouse Drive
San Diego, CA 92121-1714, US

(72) Inventor/es:

BOURD, ALEXEI, V.;
SHARP, COLIN, CHRISTOPHER;
GARCIA GARCIA, DAVID, RIGEL y
ZHANG, CHIHONG

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Técnicas de comunicación entre procesadores en una plataforma informática de múltiples procesadores.

Campo técnico

La divulgación se refiere a plataformas informáticas y, más particularmente, a plataformas informáticas que incluyen múltiples procesadores

Antecedentes

Las plataformas informáticas que incluyen múltiples procesadores se utilizan para mejorar el rendimiento de las aplicaciones que tienen requisitos de alta intensidad de cálculo y/o requisitos de alto caudal de datos. Una plataforma informática de múltiples procesadores puede incluir una unidad de procesamiento central (CPU) de propósito general que puede actuar como un dispositivo anfitrión y uno o más dispositivos informáticos que la CPU anfitriona puede utilizar para descargar las prestaciones de tareas intensivas en cálculo, mejorando así el rendimiento de todo el sistema. En algunos casos, uno o más dispositivos informáticos pueden ser diseñados específicamente para procesar ciertos tipos de tareas más eficientemente que la CPU anfitriona, lo cual puede proporcionar mejoras adicionales de rendimiento para el sistema en general. Por ejemplo, los uno o más dispositivos informáticos pueden ser diseñados específicamente para ejecutar algoritmos paralelos más eficazmente que la CPU anfitriona.

Un tipo de dispositivo informático que se puede utilizar en un sistema informático de múltiples procesadores es una unidad de procesamiento de gráficos (GPU). Tradicionalmente, las GPU incluían hardware de función fija que estaba diseñado específicamente para la representación en tiempo real de gráficos en 3 dimensiones (3D) para un dispositivo de visualización, pero no era habitualmente programable, es decir, un programa compilado no se podía descargar a la GPU y ejecutar en la GPU. Más recientemente, sin embargo, con el desarrollo de unidades de sombreado programables, gran parte de la arquitectura de la GPU se ha desplazado a una arquitectura programable que incluye muchos elementos de procesamiento en paralelo. La arquitectura programable permite que la GPU facilite la ejecución, no solo de las operaciones de gráficos, sino también de tareas de cálculo de propósito general de una manera sumamente paralela.

El uso de una GPU para ejecutar tareas de cálculo específicas sin gráficos, de propósito general, puede denominarse en el presente documento cálculo de propósito general en unidades de procesamiento de gráficos (GPGPU) o, de forma alternativa, cálculo de GPU. En algunos casos, las GPU puede dejar disponibles interfaces de programación de aplicaciones (API), que no son específicas para gráficos, facilitando de esta manera la programación de la GPU para la ejecución de tareas de cálculo de propósito general. Las tareas de cálculo de GPU pueden incluir tareas que son intensivas en cálculos y/o que incluyen un alto grado de paralelismo, por ejemplo, cálculos matriciales, cálculos de procesamiento de señales, algoritmos estadísticos, aplicaciones de modelado molecular, aplicaciones financieras, formación de imágenes médicas, aplicaciones de criptoanálisis, etc.

Una GPU es solo un tipo de dispositivo informático que se puede utilizar en una plataforma informática de múltiples procesadores, y también pueden utilizarse otros tipos de dispositivos informáticos además o en lugar de una GPU. Por ejemplo, entre otros tipos de dispositivos informáticos que pueden utilizarse en una plataforma informática de múltiples procesadores se incluyen, por ejemplo, una CPU adicional, un procesador de señales digitales (DSP), un procesador Motor de Banda Ancha Celular (Cell / BE) o cualquier otro tipo de unidad de procesamiento.

Una plataforma informática de múltiples procesadores con múltiples dispositivos informáticos puede ser una plataforma homogénea o una plataforma heterogénea. En una plataforma homogénea, todos los dispositivos informáticos comparten una arquitectura común del conjunto de instrucciones (ISA). Por el contrario, una plataforma heterogénea puede incluir dos o más dispositivos informáticos con diferentes ISA. En general, los diferentes tipos de dispositivos informáticos pueden tener diferentes ISA, y diferentes marcas de dispositivos informáticos del mismo tipo también puede tener diferentes ISA.

El rendimiento de una plataforma informática de múltiples procesadores puede mejorarse aún más mediante la utilización de dispositivos informáticos de múltiples núcleos y/o dispositivos informáticos de muchos núcleos. Un ejemplo de un dispositivo informático de múltiples núcleos es la GPU que se ha descrito anteriormente, que contiene una unidad de sombreado programable que tiene una pluralidad de núcleos de procesamiento. Las CPU, sin embargo, también pueden diseñarse para incluir múltiples núcleos de procesamiento. En general, cualquier chip o troquel que incluye múltiples núcleos de procesamiento puede considerarse como un procesador de múltiples núcleos. Un núcleo de procesamiento puede referirse a una unidad de procesamiento que es capaz de ejecutar una instrucción en un determinado conjunto de datos. Por ejemplo, una sola unidad lógica y aritmética (ALU) o un procesador vectorial dentro de una GPU puede considerarse como un núcleo de procesamiento. Los procesadores de muchos núcleos se refieren en general a los procesadores de múltiples núcleos que tienen un número relativamente alto de núcleos, por ejemplo, superior a diez núcleos y, por lo general, están diseñados utilizando técnicas diferentes a las que se utilizan para diseñar procesadores de múltiples núcleos con un número menor de

núcleos. Los procesadores de múltiples núcleos proporcionan una mejora del rendimiento al permitir que un programa de software se ejecute en paralelo, por ejemplo, al mismo tiempo, en varios núcleos en un único chip.

Un modelo de programación en paralelo se refiere a un modelo de programación que está diseñado para permitir que un programa se ejecute al mismo tiempo en múltiples núcleos de procesamiento. El programa puede ser un programa de múltiples hilos, en cuyo caso, un solo hilo puede funcionar en cada núcleo de procesamiento. En algunos ejemplos, un único dispositivo informático puede incluir la totalidad de los núcleos de procesamiento utilizados para ejecutar el programa. En otros ejemplos, algunos de los núcleos de procesamiento utilizados para ejecutar el programa pueden estar situados en diferentes dispositivos informáticos del mismo tipo o de un tipo diferente.

Se puede utilizar una interfaz de programación de aplicaciones (API) del modelo de programación en paralelo, de plataforma informática heterogénea, para múltiples plataformas y múltiples proveedores, para proporcionar una especificación de lenguaje común para la programación en paralelo de una plataforma informática heterogénea, de múltiples núcleos, que incluye diferentes tipos de dispositivos informáticos potencialmente fabricados por diferentes proveedores que implementan diferentes ISA. El Lenguaje Informático Abierto (OpenCL™) es un ejemplo de una API de programación en paralelo, de plataforma informática heterogénea, de múltiples plataformas y múltiples proveedores. Tales API pueden estar diseñadas para permitir un procesamiento de datos más generalizado en una GPU. Por ejemplo, más allá de la exposición de las capacidades del subsistema de sombreado ampliado mediante un lenguaje informático, estas API pueden generalizar el flujo de datos y los trayectos de control hacia la GPU de una manera específica sin gráficos. Actualmente, sin embargo, los conjuntos de instrucciones proporcionados por tales API se basan en la arquitectura de hardware de una GPU y, por lo tanto, están limitadas a la funcionalidad que es compatible con arquitecturas de GPU existentes.

El documento US 6618759 se refiere a las operaciones de gráficos tridimensionales en un entorno en red que tiene un anfitrión cliente y un anfitrión principal. El anfitrión cliente es capaz de emitir comandos de OpenGL para ser ejecutados por un anfitrión principal de forma inmediata. La patente divulga el almacenamiento en memoria caché de comandos de gráficos en modalidad inmediata, en particular, proporcionando dos tipos de comandos de gráficos en modalidad inmediata, un tipo de paquete de tamaño completo que incluye un campo de código de memoria caché, un campo de índice de tabla y un campo de datos correspondiente al comando, y un tipo de paquete truncado que incluye solo el campo de código de memoria caché y el campo de índice de tabla. En el caso de que el comando ya haya sido almacenado dentro del servidor principal, solo el paquete truncado se envía al anfitrión servidor y el anfitrión servidor es capaz de ejecutar el comando recuperando el campo de datos previamente almacenado en respuesta al índice de tabla suministrado en el paquete truncado.

Sumario

Esta divulgación describe las técnicas de comunicación que pueden utilizarse dentro de una plataforma informática de múltiples procesadores. Las técnicas pueden, en algunos ejemplos, proporcionar interfaces de software que pueden utilizarse para prestar soporte al paso de mensajes dentro de una plataforma informática de múltiples procesadores que inicia las tareas utilizando las colas de comandos. Las técnicas pueden, en ejemplos adicionales, proporcionar interfaces de software que pueden utilizarse para la comunicación entre procesadores de memoria compartida dentro de una plataforma informática de múltiples procesadores. En ejemplos adicionales, las técnicas pueden proporcionar una unidad de procesamiento de gráficos (GPU), que incluye hardware para prestar soporte al paso de mensajes y/o a la comunicación de memoria compartida entre la GPU y una CPU anfitriona.

De acuerdo con un primer aspecto de la presente invención, se proporciona un aparato que comprende:

medios para colocar una pluralidad de comandos en una cola de comandos en respuesta a la recepción de una o más instrucciones de puesta en cola desde un proceso que se está ejecutando en un dispositivo anfitrión, incluyendo la pluralidad de comandos un primer comando que instruye al dispositivo anfitrión para transferir datos entre un primer espacio de memoria asociado con el dispositivo anfitrión y un segundo espacio de memoria asociado con una unidad de procesamiento de gráficos (GPU), incluyendo la pluralidad de comandos, además, un segundo comando que instruye al dispositivo anfitrión para iniciar la ejecución de una tarea en la GPU; y

medios para hacer pasar uno o más mensajes entre el proceso que se está ejecutando en el dispositivo anfitrión y una tarea que se está ejecutando en la GPU, mientras la tarea se está ejecutando en la GPU, y en respuesta a la recepción de una o más instrucciones de paso de mensajes desde el proceso que se está ejecutando en el dispositivo anfitrión.

De acuerdo con un segundo aspecto de la presente invención, se proporciona un procedimiento que comprende:

la colocación, con una interfaz de cola de comandos que se está ejecutando en uno o más procesadores de un dispositivo anfitrión, de una pluralidad de comandos en una cola de comandos, en respuesta a la recepción de una o más instrucciones de puesta en cola desde un proceso que se está ejecutando en el dispositivo anfitrión, incluyendo la pluralidad de comandos un primer comando que instruye al dispositivo anfitrión para transferir datos entre un

primer espacio de memoria asociado con el dispositivo anfitrión y un segundo espacio de memoria asociado con una unidad de procesamiento de gráficos (GPU), incluyendo la pluralidad de comandos, además, un segundo comando que instruye al dispositivo anfitrión para iniciar la ejecución de una tarea en la GPU; y

el paso, con una interfaz de paso de mensajes que se está ejecutando en los uno o más procesadores del dispositivo anfitrión, de uno o más mensajes entre el proceso que se está ejecutando en el dispositivo anfitrión y una tarea que se está ejecutando en la GPU, mientras la tarea se está ejecutando en la GPU, y en respuesta a la recepción de una o más instrucciones de paso de mensajes desde el proceso en ejecución en el dispositivo anfitrión.

De acuerdo con un tercer aspecto de la presente invención, se proporciona un medio legible por ordenador que comprende instrucciones para hacer que uno o más procesadores lleven a cabo el procedimiento de acuerdo con el segundo aspecto de la presente invención.

Breve descripción de los dibujos

La FIG. 1 es un diagrama de bloques que ilustra un sistema informático ejemplar que puede utilizarse para llevar a cabo técnicas de paso de mensajes de acuerdo con esta divulgación.

La FIG. 2 es un diagrama de bloques que ilustra una GPU ejemplar que puede utilizarse en el sistema informático de la FIG. 1 de acuerdo con esta divulgación.

La FIG. 3 es un diagrama de flujo que ilustra una técnica ejemplar para el paso de mensajes en un entorno de plataforma de múltiples procesadores de acuerdo con esta divulgación.

La FIG. 4 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de envío emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación.

Las FIGs. 5 y 6 son diagramas de flujo que ilustran técnicas ejemplares que pueden utilizarse para implementar partes de la técnica ilustrada en la FIG. 4 de acuerdo con esta divulgación.

La FIG. 7 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de un mensaje recibido en un dispositivo informático, por ejemplo, una GPU, de acuerdo con esta divulgación.

La FIG. 8 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de recepción emitida por una tarea que se está ejecutando en un dispositivo informático, por ejemplo, una GPU, de acuerdo con esta divulgación.

Las FIGs. 9 y 10 son diagramas de flujo que ilustran técnicas ejemplares que pueden utilizarse para implementar partes de la técnica ilustrada en la FIG. 8 de acuerdo con esta divulgación.

La FIG. 11 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de envío emitida por un proceso que se está ejecutando en un dispositivo informático, por ejemplo, una GPU, de acuerdo con esta divulgación.

Las FIGs. 12 y 13 son diagramas de flujo que ilustran técnicas ejemplares que pueden utilizarse para implementar partes de la técnica ilustrada en la FIG. 11 de acuerdo con esta divulgación.

La FIG. 14 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de registro de rutina de devolución de llamada, emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación.

La FIG. 15 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de una interrupción recibida desde un dispositivo informático de acuerdo con esta divulgación.

Las FIGs. 16 y 17 son diagramas de flujo que ilustran técnicas ejemplares que pueden utilizarse para implementar partes de la técnica ilustrada en la FIG. 15 de acuerdo con esta divulgación.

La FIG. 18 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de lectura emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación.

La FIG. 19 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar partes de la técnica ilustrada en la FIG. 18 de acuerdo con esta divulgación.

La FIG. 20 es un diagrama de bloques que ilustra un sistema informático ejemplar que puede facilitar el uso de objetos de memoria inmediata de acuerdo con esta divulgación.

La FIG. 21 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de creación de objetos de memoria emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación.

La FIG. 22 es un diagrama de flujo que ilustra otra técnica ejemplar para la ejecución de una instrucción de creación de objetos de memoria emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación.

Las FIGs. 23 a 26 son diagramas de flujo que ilustran técnicas ejemplares para el procesamiento de instrucciones en modalidad de memoria caché y en modalidad inmediata, de acuerdo con esta divulgación.

La FIG. 27 es un diagrama de bloques que ilustra una GPU ejemplar que puede utilizarse en el sistema informático de la FIG. 20 de acuerdo con esta divulgación.

La FIG. 28 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de instrucciones en modalidad de memoria caché y en modalidad inmediata de acuerdo con esta divulgación.

La FIG. 29 es un diagrama de flujo que ilustra otra técnica ejemplar para la ejecución de una instrucción de creación de objetos de memoria emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación.

La FIG. 30 es un diagrama de flujo que ilustra cómo una GPU puede procesar una secuencia de instrucciones compiladas de acuerdo con una primera técnica de compilación de acuerdo con esta divulgación.

La FIG. 31 es un diagrama de flujo que ilustra una técnica ejemplar para la compilación de código fuente para una tarea de acuerdo con esta divulgación.

La FIG. 32 es un diagrama de flujo que ilustra una técnica ejemplar que puede ser utilizada por una GPU para utilizar selectivamente los servicios de almacenamiento en memoria caché de acuerdo con esta divulgación.

Descripción detallada

Esta divulgación describe las técnicas de comunicación que pueden utilizarse dentro de una plataforma informática de múltiples procesadores. Las técnicas pueden, en algunos ejemplos, proporcionar interfaces de software que pueden utilizarse para prestar soporte al paso de mensajes dentro de una plataforma informática de múltiples procesadores que inicia las tareas utilizando las colas de comandos. Las técnicas pueden, en ejemplos adicionales, proporcionar interfaces de software que pueden utilizarse para la comunicación entre procesadores de memoria compartida dentro de una plataforma informática de múltiples procesadores. En ejemplos adicionales, las técnicas pueden proporcionar una unidad de procesamiento de gráficos (GPU) que incluye hardware para prestar soporte al paso de mensajes y/o a la comunicación de memoria compartida entre la GPU y una CPU anfitriona.

En los últimos años, los procesadores que fueron diseñados originalmente para el procesamiento de gráficos tridimensionales en tiempo real, por ejemplo, una unidad de procesamiento de gráficos (GPU), se generalizaron para realizar tareas de cálculo de propósito general (GPGPU). El valor de las GPGPU se ha demostrado en parte por la adopción de estándares para toda la industria, tales como, por ejemplo, el estándar Lenguaje Informático Abierto (OpenCL™). OpenCL es un ejemplo de una API de programación en paralelo de plataforma informática heterogénea, de múltiples plataformas y múltiples proveedores, que puede utilizarse para ejecutar programas que tienen paralelismo a nivel de tareas y/o paralelismo a nivel de datos sobre una plataforma informática de múltiples procesadores. La API está diseñada específicamente para permitir el procesamiento de datos más generalizado en una GPU, generalizando las vías de flujo de datos y de control de la GPU de manera específica sin gráficos. Una limitación de este enfoque es la granularidad gruesa de la comunicación de datos entre la CPU anfitriona y los dispositivos informáticos, por ejemplo, una GPU.

Por ejemplo, la API de OpenCL proporciona una interfaz de cola de comandos que presta soporte a la granularidad a nivel de tareas de la comunicación entre un dispositivo anfitrión y uno o más dispositivos informáticos. Cada cola de comandos en general mantiene los comandos que serán ejecutados por un dispositivo informático específico. Un proceso anfitrión que se está ejecutando en un dispositivo anfitrión puede transferir datos entre el espacio de memoria del anfitrión y el espacio de memoria del dispositivo, mediante la colocación de un comando en la cola de comandos que instruye al dispositivo anfitrión para realizar la transferencia de memoria. De forma similar, el proceso anfitrión puede hacer que una tarea empiece a ejecutarse en un dispositivo informático mediante la colocación de un comando en la cola de comandos, instruyendo al dispositivo anfitrión para ejecutar una tarea en el dispositivo informático.

La interfaz de cola de comandos puede configurarse para proporcionar la ejecución de comandos en orden o la ejecución de comandos de forma desordenada. Cuando la interfaz de cola de comandos está configurada para proporcionar la ejecución de comandos en orden, la interfaz de cola de comandos garantiza que los comandos se

ejecutarán en el orden en que los comandos se colocaron en la cola de comandos y que la ejecución de un comando posterior no comenzará hasta que el comando anterior haya finalizado su ejecución. Por lo tanto, cuando el proceso anfitrión coloca un comando en la cola de comandos para ejecutar una tarea, la cola de comandos espera a que la tarea complete la ejecución antes de ejecutar comandos adicionales cualesquiera que pueden ser colocados posteriormente en la cola de comandos.

En una configuración sencilla que implica una CPU anfitriona y una GPU y una cola de comandos en orden, un sistema de comunicación entre la CPU anfitriona y la GPU puede implicar las siguientes operaciones: (1) la CPU anfitriona prepara los datos y los coloca en la memoria accesible para la GPU; (2) la CPU anfitriona ordena a la GPU que ejecute una tarea; (3) la CPU anfitriona espera a que la GPU termine la ejecución de la tarea; y (4) la CPU anfitriona copia los datos desde la memoria accesible para la GPU a la memoria del anfitrión. En una configuración de este tipo, todos los datos necesarios para la ejecución de una tarea en la GPU se transfieren a la memoria accesible para la GPU antes del inicio de la ejecución de la tarea, y los datos producidos por la tarea que se está ejecutando en la GPU no están disponibles para la CPU anfitriona hasta después de que la tarea que se está ejecutando en la GPU complete su ejecución. Esta tosquedad en el intercambio de datos entre una CPU anfitriona y una GPU puede impedir las implementaciones efectivas de muchas operaciones útiles para aplicaciones basadas en paralelo, tales como, por ejemplo, la transmisión de mensajes entre procesos entre un proceso que se está ejecutando en el dispositivo anfitrión y una tarea que se está ejecutando en la GPU. Tales mensajes pueden ser útiles, por ejemplo, para permitir que una tarea que se está ejecutando en la GPU tenga la capacidad de ejecutar una llamada a procedimiento remoto (RPC) en la CPU anfitriona.

Cuando la interfaz de cola de comandos está configurada para proporcionar la ejecución desordenada de los comandos, el proceso anfitrión no es capaz de controlar cuándo, durante la ejecución de una tarea en particular, se llevará a cabo la ejecución de un comando en particular. Como tal, el modo de ejecución desordenada para la cola de comandos tampoco permite eficazmente la implementación del paso de mensajes entre procesos, entre un proceso que se está ejecutando en el dispositivo anfitrión y una tarea que se está ejecutando en la GPU.

Con respecto al modelo de memoria utilizada en OpenCL, la API define las denominadas memorias intermedias globales del CL y las imágenes globales del CL que pueden utilizarse para la compartición de datos entre una CPU anfitriona y una GPU o para compartir datos entre múltiples dispositivos informáticos de OpenCL. Sin embargo, la CPU y la GPU no pueden leer o escribir en una memoria intermedia al mismo tiempo. Habitualmente, una CPU prepara una o más memorias intermedias que contienen datos de origen y pasa las memorias intermedias a la GPU para su procesamiento. La GPU o bien modifica estas memorias intermedias o pone resultados en otras memorias intermedias que también fueron asignadas por software ejecutándose en la CPU *a priori* con el propósito de recibir las modificaciones de datos de GPU.

Aunque los objetos de memoria en OpenCL actualmente permiten utilizar una región del espacio de memoria del anfitrión para almacenar los datos de memoria intermedia utilizados por un dispositivo informático, la especificación permite que los dispositivos informáticos almacenen en memoria caché estos datos para la ejecución más eficaz de una tarea. El dispositivo anfitrión, en general, no es capaz de invalidar inmediatamente la memoria caché del dispositivo informático que se utiliza para almacenar en memoria caché los datos de memoria intermedia. Por lo tanto, incluso si el dispositivo anfitrión fuera a sobrescribir ciertos datos de memoria intermedia almacenados en el espacio de memoria del anfitrión, no hay garantía de que la memoria caché en el dispositivo informático se actualice con el fin de proporcionar al dispositivo informático acceso inmediato a los datos modificados. Además, dado que los resultados de los cálculos realizados por el dispositivo informático se pueden almacenar en la memoria caché del dispositivo informático, el proceso anfitrión que se está ejecutando en el dispositivo anfitrión no es capaz de leer ningún resultado parcial de la memoria intermedia, debido a que tales datos pueden ser inválidos debido a nuevos datos almacenados en la memoria caché del dispositivo informático. Por lo tanto, el modelo de gestión de memoria en OpenCL no permite inmediatamente compartir datos en proceso mediante una memoria compartida.

Las técnicas descritas en esta divulgación pueden utilizarse, en algunos ejemplos, para superar una o más de las limitaciones anteriormente mencionadas de la API de OpenCL. Por ejemplo, las técnicas de esta divulgación pueden proporcionar interfaces de software que se pueden utilizar para prestar soporte al paso de mensajes entre procesos dentro de una plataforma informática de múltiples procesadores que inicia las tareas utilizando colas de comandos de granularidad a nivel de tareas. Como otro ejemplo, las técnicas de esta divulgación pueden proporcionar interfaces de software que pueden utilizarse para prestar soporte al intercambio de datos en proceso mediante una memoria compartida dentro de una plataforma informática de múltiples procesadores.

En algunos ejemplos, las técnicas de esta divulgación pueden proporcionar una arquitectura de hardware de GPU que facilite el paso de mensajes a nivel de software. Por ejemplo, las técnicas de esta divulgación pueden proporcionar una arquitectura de hardware de GPU que está configurada para prestar soporte a la ejecución de instrucciones de paso de mensajes a nivel de software. En otros ejemplos, las técnicas de la presente divulgación pueden proporcionar una arquitectura de hardware de GPU que facilita la comunicación de memoria compartida entre la GPU y una CPU anfitriona. Por ejemplo, las técnicas de esta divulgación pueden proporcionar una arquitectura de hardware de GPU que está configurada para habilitar e inhabilitar de forma selectiva servicios de almacenamiento en memoria caché para un espacio de memoria compartida y/o para habilitar e inhabilitar de forma

selectiva un mecanismo de coherencia de memoria caché para un espacio de memoria compartida.

De acuerdo con un primer aspecto de esta divulgación, se proporciona una interfaz de paso de mensajes que facilita la ejecución de instrucciones de paso de mensajes entre un dispositivo anfitrión y uno o más dispositivos informáticos durante la ejecución de una tarea por el dispositivo informático. El paso de mensajes puede referirse a una forma de comunicación entre procesos, y potencialmente entre dispositivos, en la que cada uno de los procesos que se están comunicando realiza conjuntos complementarios de operaciones para pasar con éxito un mensaje. Por ejemplo, cada uno de los procesos que se comunica de acuerdo con un protocolo de paso de mensajes puede poner en práctica una operación de envío y una operación de recepción. Las técnicas de paso de mensajes en la presente divulgación pueden permitir que una CPU y un dispositivo informático, por ejemplo, una GPU, se pasen mensajes entre sí durante la ejecución de una tarea en el dispositivo informático. De esta manera, una plataforma informática de múltiples procesadores que implementa un esquema de comunicación de cola de comandos de granularidad a nivel de tareas puede ser capaz de facilitar la comunicación entre procesos y/o entre dispositivos.

En algunos ejemplos, las técnicas de paso de mensajes descritas en esta divulgación pueden denominarse técnicas de "señalización de fuera de banda" debido a que las técnicas pueden utilizar una interfaz distinta a la interfaz de cola de comandos, que se utiliza habitualmente en OpenCL para la comunicación entre un dispositivo anfitrión y un dispositivo informático, tal como, por ejemplo, una GPU. En otras palabras, las técnicas de esta divulgación pueden incluir una nueva interfaz de comunicación fuera de banda que está lógicamente separada de la interfaz de cola de comandos dentro de banda, incluida dentro de OpenCL. La interfaz de comunicación fuera de banda puede no estar sujeta a la misma granularidad a nivel de tareas a la que está sujeta la interfaz de cola de comandos, proporcionando de ese modo una solución a una o más de las limitaciones descritas anteriormente con respecto a la granularidad a nivel de tareas de la cola de comandos.

Los mensajes transferidos entre la CPU y la GPU de acuerdo con las técnicas de esta divulgación pueden ser de cualquier tipo de mensaje. Ejemplos de diferentes tipos de mensajes incluyen señales, solicitudes de asignación de memoria, solicitudes de des-asignación de memoria, mensajes de notificación, mensajes de sincronización, mensajes de invocación a procedimiento remoto (por ejemplo, mensajes que son parte de una llamada a procedimiento remoto (RPC)), paquetes de datos, mensajes de informes, mensajes de mecanismo de afirmación y mensajes de registro.

En el actual paradigma de OpenCL, todas las peticiones de una CPU anfitriona a una GPU se ponen en colas de comandos de OpenCL y se envían luego a la GPU. En particular, una aplicación podría poner en la cola de comandos un gran número de ejecuciones del núcleo y operaciones de memoria intermedia. Mientras tanto, si la primera tarea puesta en cola, por ejemplo, una ejecución del núcleo, necesita, por ejemplo, solicitar la asignación de memoria adicional de una CPU, surgen varios problemas. En primer lugar, ¿cómo notifica la GPU a la CPU desde el interior de un núcleo en ejecución que necesita que se produzca la asignación de memoria? En segundo lugar, ¿cómo notificar la CPU a la GPU la finalización de la asignación de memoria y de la dirección del bloque de memoria recién asignado? Las técnicas de interfaz de paso de mensajes de esta divulgación, sin embargo, pueden ser capaces de resolver estos problemas al permitir que uno o más mensajes que contienen las notificaciones y la información descritos anteriormente se pasen entre la CPU y la GPU.

Las técnicas de señalización fuera de banda de esta divulgación pueden utilizarse, en algunos ejemplos, para implementar la señalización entre una CPU anfitriona y uno o más dispositivos informáticos, por ejemplo, dispositivos informáticos de OpenCL. La señalización fuera de banda puede proporcionar rápidas notificaciones de fuera de banda, por ejemplo, utilizando un mecanismo de entrega solicitada o no solicitada. En algunos ejemplos, las técnicas de señalización fuera de banda pueden llevar cantidades relativamente pequeñas de datos.

De acuerdo con un segundo aspecto de esta divulgación, se proporciona una GPU que es capaz de enviar mensajes a, y recibir mensajes desde, los procesos que se están ejecutando en procesadores distintos a la GPU. Por ejemplo, una GPU puede incluir hardware que está configurado para implementar una o más operaciones para enviar y recibir mensajes. En algunos ejemplos, una GPU diseñada de acuerdo con esta divulgación puede incluir uno o más registros accesibles por anfitrión, configurados para el almacenamiento de información de estado y de datos, asociada con un protocolo de paso de mensajes. Los uno o más registros pueden estar configurados para facilitar el paso de mensajes entre una tarea que se está ejecutando en la GPU y un proceso que se está ejecutando en un dispositivo distinto a la GPU. En ejemplos adicionales, el bloque de procesamiento de la ALU de la GPU (por ejemplo, una unidad de sombreado programable) puede estar acoplado comunicativamente a los registros accesibles por anfitrión para enviar y recibir mensajes mediante los registros accesibles por anfitrión. La GPU también puede diseñarse para incluir varios mecanismos de sondeo y/o interrupción para implementar técnicas de paso de mensajes, síncronas y/o asíncronas.

De acuerdo con un tercer aspecto de esta divulgación, se proporciona una interfaz de memoria intermedia que permite crear objetos de memoria inmediata. Los objetos de memoria inmediata se pueden usar para implementar un espacio de memoria compartida no almacenable en memoria caché y/o un espacio de memoria compartida coherente con memoria caché, con el fin de compartir datos entre un proceso que se está ejecutando en un dispositivo anfitrión y una tarea que se está ejecutando en un dispositivo informático, mientras la tarea se está

ejecutando en el dispositivo informático. El espacio de memoria compartida puede ser un espacio de memoria que sea accesible tanto por el dispositivo anfitrión como por un dispositivo informático, por ejemplo, una GPU, durante la ejecución de una tarea por el dispositivo informático. Un espacio de memoria compartida no almacenable en memoria caché, como se usa en el presente documento, puede referirse a un espacio de memoria compartida para el cual una o más memorias caché correspondientes en el dispositivo anfitrión y/o el dispositivo informático, o ambos, están inhabilitadas para el espacio de memoria. Un espacio de memoria compartida coherente con memoria caché, como se usa en el presente documento, puede referirse a un espacio de memoria compartida donde se utilizan técnicas de coherencia de caché de memoria compartida para mantener la coherencia de memoria caché dentro de una o más memorias caché correspondientes en uno entre el dispositivo anfitrión y/o el dispositivo informático, o en ambos. El espacio de memoria compartida no almacenable en memoria caché y el espacio de memoria compartida coherente con memoria caché pueden permitir la compartición de datos en cualquier momento. Los objetos de memoria inmediata pueden, en algunos ejemplos, implementarse como una memoria compartida volátil no almacenable en memoria caché y/o como una memoria compartida volátil coherente con memoria caché para el dispositivo anfitrión y el dispositivo informático.

En algunos ejemplos, los objetos de memoria inmediata de esta divulgación pueden estar integrados dentro de una API de programación en paralelo, de plataforma informática heterogénea, de múltiples plataformas y múltiples proveedores, que incluye un esquema de gestión de memoria de objetos de memoria. Por ejemplo, los objetos de memoria inmediatos pueden estar integrados en OpenCL como un atributo adicional de los objetos de memoria de OpenCL, por ejemplo, objetos de memoria intermedia de OpenCL o los objetos de imagen de OpenCL. En tales ejemplos, los objetos de memoria inmediatos se pueden crear mediante la modificación de las funciones de creación de objetos de memoria para incluir un parámetro o indicador que especifica si el objeto de memoria resultante creado por la llamada a la función debería ser un objeto de memoria de modalidad estándar o un objeto de memoria de modalidad inmediata. De esta manera, las técnicas de esta divulgación pueden permitir a los sistemas informáticos de múltiples procesadores que implementan las API que incluyen esquemas de gestión de memoria de objetos de memoria, tales como OpenCL, poner en práctica el intercambio de datos en proceso mediante un espacio de memoria compartida que no esté sujeto a problemas de coherencia de memoria caché.

En ejemplos adicionales, los objetos de memoria inmediata de esta divulgación pueden utilizarse para el intercambio de datos en desarrollo entre una CPU anfitriona y un dispositivo informático de OpenCL o entre diferentes dispositivos informáticos de OpenCL. En ejemplos adicionales, los objetos de memoria inmediatos pueden contener marcadores de sincronización interna. En ejemplos adicionales, los objetos de memoria inmediatos se pueden usar junto con las señales fuera de banda para la sincronización.

De acuerdo con un cuarto aspecto de esta divulgación, se proporciona una GPU que incluye una memoria caché correspondiente a un espacio de memoria compartida, que puede inhabilitarse de forma selectiva para determinados espacios de direcciones de memoria con el fin de proporcionar un espacio de memoria compartida no almacenable en memoria caché. Por ejemplo, la GPU puede habilitar e inhabilitar de forma selectiva los servicios de memoria caché proporcionados por una caché asociada con un espacio de memoria compartida, en respuesta a la recepción de información que especifica si se deberían utilizar los servicios de almacenamiento en memoria caché para la ejecución de operaciones de lectura y/o escritura con respecto al espacio de memoria compartida. En algunos ejemplos, la información que especifica si se deberían utilizar los servicios de almacenamiento en memoria caché para la ejecución de operaciones de lectura y/o escritura con respecto al espacio de memoria compartida puede ser una instrucción de modalidad de almacenamiento en memoria caché o una instrucción de modalidad inmediata que especifica si debería utilizarse una modalidad de memoria caché o una modalidad inmediata para ejecutar la instrucción particular. En ejemplos adicionales, la información que especifica si se deberían utilizar los servicios de almacenamiento en memoria caché para la ejecución de las operaciones de lectura y/o escritura con respecto al espacio de memoria compartida puede ser un atributo de objeto de memoria en modalidad inmediata que especifica si una modalidad inmediata está habilitada para el objeto de memoria.

En ejemplos adicionales, las técnicas de esta divulgación pueden proporcionar una GPU que incluye una modalidad de coherencia con memoria caché que puede habilitarse selectivamente para proporcionar un espacio de memoria compartida coherente con memoria caché. En algunos ejemplos, la GPU puede habilitar selectivamente una modalidad de coherencia de memoria caché para una parte de la memoria caché correspondiente al espacio de memoria compartida, en base a una o más instrucciones recibidas desde un dispositivo anfitrión. El dispositivo anfitrión puede emitir una o más instrucciones a la GPU para habilitar selectivamente la modalidad de coherencia de memoria caché del espacio de memoria compartida para la parte de la memoria caché correspondiente al espacio de memoria compartida tras la asignación del espacio de memoria compartida por el dispositivo anfitrión, en base a un parámetro de modalidad inmediata especificado por un proceso anfitrión.

Las técnicas de señalización fuera de banda y de almacenamiento inmediato en memoria intermedia de esta divulgación pueden proporcionar un acoplamiento de tareas más minucioso entre una CPU anfitriona y la GPU, o entre dos dispositivos informáticos de OpenCL, en comparación con lo que de otra manera se podría obtener utilizando solamente la interfaz de cola de comandos de OpenCL. Las técnicas de esta divulgación pueden permitir que una plataforma informática de múltiples procesadores lleve a cabo una diversidad de operaciones con el fin de asistir en la ejecución eficaz de programas en paralelo y/o de múltiples hilos. Por ejemplo, las técnicas de esta

divulgación pueden permitir que una tarea que está ejecutándose en una GPU lance una RPC. Como otro ejemplo, las técnicas de esta divulgación pueden permitir que una tarea que se está ejecutando en la GPU lance, mediante la CPU, otra tarea de GPU. Como un ejemplo adicional, las técnicas de esta divulgación pueden permitir que una tarea que se está ejecutando en la GPU emita solicitudes de gestión de recursos, tales como, por ejemplo, las solicitudes de asignación de memoria y/o de des-asignación de memoria, a la CPU y/o al controlador que se está ejecutando en la CPU. Como otro ejemplo más, las técnicas de esta divulgación pueden permitir que una tarea de que se está ejecutando en la GPU lleve a cabo comprobaciones de estado y el paso general de mensajes a una CPU, tal como, por ejemplo, la implementación de un mecanismo de afirmación, informes de progreso y/o registro de diagnóstico.

La FIG. 1 es un diagrama de bloques que ilustra un sistema informático ejemplar 10 de acuerdo con esta divulgación. El sistema informático 10 está configurado para procesar una o más aplicaciones de software en múltiples dispositivos de procesamiento. En algunos ejemplos, las una o más aplicaciones de software pueden incluir un proceso anfitrión, y el sistema informático 10 puede estar configurado para ejecutar el proceso anfitrión y para distribuir la ejecución de una o más tareas iniciadas por el proceso anfitrión en otros dispositivos informáticos dentro del sistema informático 10. En ejemplos adicionales, el proceso anfitrión y/o las tareas ejecutadas por el sistema informático 10 pueden ser programados de acuerdo con un modelo de programación en paralelo. Por ejemplo, las aplicaciones pueden incluir instrucciones que están diseñadas para aprovechar el paralelismo a nivel de tareas y/o el paralelismo a nivel de datos de los sistemas de hardware subyacentes.

El sistema informático 10 puede ser un ordenador personal, un ordenador de sobremesa, un ordenador portátil, una estación de trabajo de ordenador, una consola o plataforma de videojuegos, un teléfono móvil, tal como, por ejemplo, un teléfono celular o por satélite, un teléfono móvil, un teléfono fijo, un teléfono de Internet, un dispositivo portátil como un dispositivo de videojuegos portátil o un asistente digital personal (PDA), un reproductor de medios digitales, tal como un reproductor personal de música, un reproductor de vídeo, un dispositivo de visualización, o un televisor, un decodificador de televisión, un servidor, un dispositivo de red intermedio, un ordenador central o cualquier otro tipo de dispositivo que procese información.

El sistema informático 10 incluye un dispositivo anfitrión 12, una unidad de procesamiento de gráficos (GPU) 14, una memoria 16 y una red 18 de interconexión. El dispositivo anfitrión 12 está configurado para proporcionar una plataforma para la ejecución de un proceso anfitrión y un módulo de tiempo de ejecución para una API de plataforma informática de múltiples procesadores. Habitualmente, el dispositivo anfitrión 12 es una CPU de propósito general, aunque el dispositivo anfitrión 12 puede ser cualquier tipo de dispositivo capaz de ejecutar programas. El dispositivo anfitrión 12 está acoplado comunicativamente a la GPU 14 y a la memoria 16 a través de la red de interconexión 18. El dispositivo anfitrión 12 incluye un proceso anfitrión 20 y un módulo de tiempo de ejecución 22, cada uno de los cuales se puede ejecutar en cualquier combinación de uno o más procesadores programables.

El proceso anfitrión 20 incluye un conjunto de instrucciones que forman un programa de software para su ejecución en la plataforma de sistema informático del sistema informático 10. El programa de software puede estar diseñado para realizar una o más tareas específicas para un usuario final. Tales tareas pueden, en algunos ejemplos, implicar algoritmos intensivos en cálculo que pueden explotar los múltiples dispositivos de procesamiento y las arquitecturas paralelas proporcionadas por el sistema informático 10.

El módulo de tiempo de ejecución 22 puede ser un módulo de software que se está ejecutando en el dispositivo anfitrión 12 que implementa una o más interfaces configuradas para dar servicio a una o más de las instrucciones contenidas en el proceso anfitrión 20. Las interfaces implementadas por el módulo de tiempo de ejecución 22 incluyen una interfaz de cola de comandos 24 y una interfaz de paso de mensajes de anfitrión 26. En algunos ejemplos, el módulo de tiempo de ejecución 22 puede implementar una o más interfaces contenidas dentro de una API estándar del sistema de múltiples procesadores, además de las interfaces descritas en esta divulgación. En algunos ejemplos, la API estándar puede ser una API de plataforma informática heterogénea, una API de múltiples plataformas, una API de múltiples proveedores, una API de programación en paralelo, una API de programación en paralelo a nivel de tareas y/o una API de programación en paralelo a nivel de datos. En otros ejemplos, la API estándar puede ser la API de OpenCL. En dichos ejemplos, el módulo de tiempo de ejecución 22 puede estar diseñado para estar en conformidad con una o más de las especificaciones de OpenCL. En ejemplos adicionales, el módulo de tiempo de ejecución 22 puede implementarse como parte de, o ser, un programa controlador, por ejemplo, un controlador de GPU.

La interfaz de cola de comandos 24 está configurada para recibir una o más instrucciones de puesta en cola desde el proceso anfitrión 20 y para ejecutar las funciones especificadas por las instrucciones recibidas. En algunos ejemplos, la interfaz de cola de comandos 24 puede estar diseñada de acuerdo con la especificación de OpenCL. Por ejemplo, la interfaz de cola de comandos 24 puede implementar una o más de las instrucciones de puesta en cola especificadas en la especificación de OpenCL para interactuar con las colas de comandos.

De acuerdo con esta divulgación, la interfaz de paso de mensajes de anfitrión 26 está configurada para recibir una o más instrucciones de paso de mensajes desde el proceso anfitrión 20 y para ejecutar las funciones especificadas por las instrucciones recibidas. En algunos ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede implementarse como una extensión para una API estándar existente, tal como, por ejemplo, la API de OpenCL. En

ejemplos adicionales, la interfaz de paso de mensajes de anfitrión 26 se puede integrar en una API estándar existente, tal como, por ejemplo, la API de OpenCL.

La GPU 14 está configurada para ejecutar una o más tareas en respuesta a las instrucciones recibidas desde el dispositivo anfitrión 12. La GPU 14 puede ser cualquier tipo de GPU que incluya uno o más elementos de procesamiento programables. Por ejemplo, la GPU 14 puede incluir una o más unidades de sombreado programables que están configuradas para ejecutar una pluralidad de instancias de ejecución para una tarea en paralelo. Las unidades de sombreado programables pueden incluir una unidad de sombreado de vértices, una unidad de sombreado de fragmentos, una unidad de sombreado de geometría y/o una unidad de sombreado unificada. La GPU 14 está acoplada comunicativamente al dispositivo anfitrión 12 y a la memoria 16 a través de la red de interconexión 18. La GPU 14 incluye una tarea 28 y una interfaz de paso de mensajes de dispositivo 30. La tarea 28 y la interfaz de paso de mensajes de dispositivo 30 se puede ejecutar en cualquier combinación de uno o más elementos de procesamiento programables.

La tarea 28 comprende un conjunto de instrucciones que forman una tarea para la ejecución en un dispositivo informático en el sistema informático 10. En algunos ejemplos, el conjunto de instrucciones para la tarea 28 puede definirse en el proceso anfitrión 20 y, en algunos casos, compilarse mediante instrucciones incluidas en el proceso anfitrión 20 que se están ejecutando en el dispositivo anfitrión 12. En otros ejemplos, la tarea 28 puede ser un programa de núcleo que tiene múltiples instancias de ejecución que se están ejecutando en la GPU 14 en paralelo. En dichos ejemplos, el proceso anfitrión 20 puede definir un espacio de índices para el núcleo que correlaciona instancias de ejecución del núcleo con respectivos elementos de procesamiento para la ejecución de las instancias de ejecución del núcleo, y la GPU 14 puede ejecutar las múltiples instancias de ejecución de núcleo para la tarea 28 de acuerdo con el espacio de índices definido para el núcleo.

De acuerdo con esta divulgación, la interfaz de paso de mensajes de dispositivo 30 está configurada para recibir una o más instrucciones de paso de mensajes desde el proceso anfitrión 20 y para ejecutar las funciones especificadas por las instrucciones recibidas. En algunos ejemplos, la interfaz de paso de mensajes de dispositivo 30 puede implementarse como una extensión de una API estándar existente. Por ejemplo, la API estándar puede ser una API de dispositivo informático estándar, tal como, por ejemplo, la API de C de OpenCL. En ejemplos adicionales, la interfaz de paso de mensajes de dispositivo 30 puede estar integrada en una API estándar existente, tal como, por ejemplo, la API de C de OpenCL.

La memoria 16 está configurada para almacenar datos para su uso por uno entre el dispositivo anfitrión 12 y la GPU 14, o ambos. La memoria 16 puede incluir cualquier combinación de una o más memorias o dispositivos de almacenamiento, volátiles o no volátiles, tales como, por ejemplo, memoria de acceso aleatorio (RAM), RAM estática (SRAM), RAM dinámica (DRAM), memoria de solo lectura (ROM), ROM borrable y programable (EPROM), ROM borrable y programable eléctricamente (EEPROM), memoria flash, un medio magnético de almacenamiento de datos o un medio de almacenamiento óptico. La memoria 16 está acoplada comunicativamente a un dispositivo anfitrión 12 y a la GPU 14 a través de la red de interconexión 18. La memoria 16 incluye la cola de comandos 32.

La cola de comandos 32 puede ser una estructura de datos implementada en la memoria 16 que almacena y recupera los comandos recibidos desde la interfaz de cola de comandos 24. En algunos ejemplos, la cola de comandos 32 puede ser una memoria intermedia que almacena los comandos en un orden particular para su ejecución.

La red de interconexión 18 está configurada para facilitar la comunicación entre el dispositivo anfitrión 12, la GPU 14 y la memoria 16. La red de interconexión 18 puede ser cualquier tipo de red de interconexión conocida en la técnica. En el sistema informático ejemplar 10 de la FIG. 1, la red de interconexión 18 es un bus. El bus puede incluir una o más de cualquiera entre una variedad de estructuras de bus, tales como, por ejemplo, un bus de tercera generación (por ejemplo, un bus HyperTransport o un bus InfiniBand), un bus de segunda generación (por ejemplo, un bus de Puerto de Gráficos Avanzado, un bus Expreso de Interconexión de Componentes Periféricos (PCIe) o un bus de Interfaz Extensible Avanzada (AXI)) o cualquier otro tipo de bus. La red de interconexión 18 está acoplada al dispositivo anfitrión 12, la GPU 14 y la memoria 16.

Las estructuras y funcionalidades de los componentes en el sistema informático 10 se describirán ahora con más detalle. Como se ha expuesto anteriormente, el proceso anfitrión 20 incluye un conjunto de instrucciones. El conjunto de instrucciones puede incluir, por ejemplo, una o más instrucciones de puesta en cola, y una o más instrucciones de paso de mensajes de anfitrión. En ejemplos adicionales, el conjunto de instrucciones puede incluir instrucciones que especifican las tareas o núcleos que se tienen que ejecutar en la GPU 14, instrucciones que crean las colas de comandos y asocian las colas de comando con determinados dispositivos, instrucciones que compilan y vinculan programas, instrucciones que configuran parámetros de núcleo , instrucciones que definen espacios de índices, instrucciones que definen un contexto de dispositivo y otras instrucciones que prestan soporte a la funcionalidad proporcionada por el proceso anfitrión 20.

El proceso anfitrión 20 puede interactuar con la interfaz de cola de comandos 24 mediante la emisión de una o más instrucciones de puesta en cola para la interfaz de cola de comandos 24, que instruyen a la interfaz de cola de

comandos 24 para colocar uno o más comandos en la cola de comandos 32. Las una o más instrucciones de puesta en cola pueden incluir instrucciones de puesta en cola de transferencias de memoria, que instruyen a la interfaz de cola de comandos 24 para poner en cola un comando de transferencia de memoria en la cola de comandos 32. Por ejemplo, las una o más instrucciones de puesta en cola pueden incluir una instrucción para poner en cola un comando que instruye al dispositivo anfitrión 12, por ejemplo, el módulo de tiempo de ejecución 22 que se ejecuta en el dispositivo anfitrión 12, para transferir datos entre un espacio de memoria asociado con el dispositivo anfitrión 12 y un espacio de memoria asociado con la GPU 14.

Un espacio de memoria puede estar asociado con el dispositivo anfitrión 12, si el espacio de memoria es accesible por el dispositivo anfitrión 12 durante la ejecución del proceso anfitrión 20 por el dispositivo anfitrión 12. De manera similar, un espacio de memoria puede estar asociado con la GPU 14 si el espacio de memoria es accesible por la GPU 14 durante la ejecución de la tarea 28 por la GPU 14. El espacio de memoria asociado con el dispositivo anfitrión 12 puede denominarse en la presente memoria espacio de memoria del anfitrión, y el espacio de memoria asociado con la GPU 14 puede denominarse en la presente memoria espacio de memoria del dispositivo. En algunos ejemplos, la memoria 16 puede incluir partes tanto del espacio de memoria del anfitrión como del espacio de memoria del dispositivo. En otros ejemplos, las partes de uno entre el espacio de memoria del anfitrión y el espacio de memoria del dispositivo, o ambos, pueden estar situadas en otros uno o más dispositivos de memoria que no se muestran en el sistema informático 10 de la FIG. 1.

En algunos ejemplos, el comando que instruye al dispositivo anfitrión 12 para transferir datos entre un espacio de memoria asociado con el dispositivo anfitrión 12 y un espacio de memoria asociado con la GPU 14 puede ser un comando que instruya al módulo de tiempo de ejecución 22 para transferir los datos almacenados en una parte del espacio de memoria del anfitrión a un objeto de memoria intermedia asignado en el espacio de memoria del dispositivo. La instrucción emitida por el proceso anfitrión 20 para poner en cola un comando de este tipo puede denominarse en el presente documento una instrucción de puesta en cola de escritura de memoria intermedia. En algunos casos, la instrucción de puesta en cola de memoria intermedia de escritura puede adoptar la forma de la función *clEnqueueWriteBujfer ()* especificada por la especificación de API de OpenCL.

En ejemplos adicionales, el comando que instruye al dispositivo anfitrión 12 para transferir datos entre un espacio de memoria asociado con el dispositivo anfitrión 12 y un espacio de memoria asociado con la GPU 14 puede ser un comando que instruye al módulo de tiempo de ejecución 22 para transferir los datos almacenados en un objeto de memoria intermedia, asignado en el espacio de memoria del dispositivo, a una parte del espacio de memoria del anfitrión. La instrucción emitida por el proceso anfitrión 20 para poner en cola un comando de este tipo puede denominarse en el presente documento una instrucción de puesta en cola de lectura de memoria intermedia. En algunos casos, la instrucción de puesta en cola de lectura de memoria intermedia puede tomar la forma de la función *clEnqueueReadBujfer()* especificada por la especificación de API de OpenCL.

Las una o más instrucciones de puesta en cola también pueden incluir instrucciones de puesta en cola de ejecución de tareas que instruyen a la interfaz de cola de comandos 24 para poner en cola un comando de ejecución de tareas en la cola de comandos 32. Por ejemplo, las una o más instrucciones de puesta en cola pueden incluir una instrucción para poner en cola un comando que instruye a un dispositivo anfitrión 12, por ejemplo, el módulo de tiempo de ejecución 22 que se ejecuta en el dispositivo anfitrión 12, para ejecutar una tarea en la GPU 14. En algunos ejemplos, el comando para ejecutar la tarea puede ser un comando para ejecutar múltiples instancias de ejecución de la tarea en una pluralidad de elementos de procesamiento en la GPU 14 en paralelo. Por ejemplo, la tarea puede ser el núcleo, y el proceso anfitrión 20 puede definir un espacio de índices para el núcleo que correlaciona instancias de ejecución del núcleo con los respectivos elementos de procesamiento en la GPU 14 para la ejecución de las instancias de ejecución del núcleo. En un ejemplo de ese tipo, el comando para ejecutar una tarea puede ser un comando para ejecutar un núcleo en la GPU 14 de acuerdo con un espacio de índices definido para la GPU 14. En algunos casos, una instrucción de puesta en cola de ejecución de tareas puede tomar la forma de la función *clEnqueueNDRangeKernel ()* especificada por la API de OpenCL.

De acuerdo con esta divulgación, el proceso anfitrión 20 también puede interactuar con la interfaz de paso de mensajes del anfitrión 26, emitiendo una o más instrucciones de paso de mensajes del anfitrión a la interfaz de paso de mensajes del anfitrión 26, instruyendo a la interfaz de paso de mensajes del anfitrión 26 para pasar uno o más mensajes entre el proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12 y la tarea 28 que se está ejecutando en la GPU 14. Las instrucciones de paso de mensajes del anfitrión pueden ser ejecutadas por el dispositivo anfitrión 12.

Las instrucciones de paso de mensajes del anfitrión pueden, en algunos ejemplos, incluir una instrucción de envío que instruye al dispositivo anfitrión 12 para enviar los datos especificados a un dispositivo especificado. Por ejemplo, la instrucción de envío puede instruir a la interfaz de paso de mensajes del anfitrión 26 para enviar un mensaje desde el proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12 a la tarea 28 que se está ejecutando en la GPU 14. En algunos ejemplos, la instrucción de envío puede incluir un primer parámetro de entrada que especifica un dispositivo particular al cual debería enviarse el mensaje y un segundo parámetro de entrada que especifica el contenido del mensaje a enviar.

La instrucción de envío puede ser o bien una instrucción de envío bloqueante o una instrucción de envío no bloqueante. La instrucción de envío puede incluir, en algunos ejemplos, un tercer parámetro de entrada que especifica si la instrucción de envío es una instrucción de envío bloqueante o una instrucción de envío no bloqueante. Una instrucción de envío bloqueante puede esperar hasta que se haya completado la operación de envío antes de volver al proceso de llamada, por ejemplo, el proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12. Una instrucción de envío no bloqueante puede volver al proceso de llamada sin tener que esperar hasta que se complete la operación de envío. Por ejemplo, la instrucción de envío no bloqueante puede devolver un asidero a la operación de envío particular, que posteriormente puede ser consultado por el proceso de llamada para determinar si la operación de envío se ha realizado con éxito. Una operación de envío no bloqueante puede fallar y, en el caso de fallo, el proceso de llamada puede tener que emitir de nuevo la instrucción de envío para volver a intentar la operación de envío.

En algunos ejemplos, la interfaz para la instrucción de envío puede adoptar la forma siguiente:

```
clEnviarDatosFueraDeBanda(
        cl_dispositivo *identificador_dispositivo,
        int datos_OOB,
        bool bloqueante)
```

donde clEnviarDatosFueraDeBanda es el identificador de instrucción, cl_dispositivo *identificador_dispositivo es un parámetro de entrada que especifica un dispositivo de OpenCL particular al cual debería enviarse el mensaje, int datos_OOB es un parámetro de entrada que especifica el contenido del mensaje a enviar, y bool bloqueante es un parámetro de entrada que especifica si la instrucción es una instrucción de envío bloqueante o una instrucción de envío no bloqueante. En el caso de una instrucción bloqueante, la instrucción puede devolver un parámetro indicativo de si se completó con éxito la operación de envío. En el caso de una instrucción no bloqueante, la instrucción puede devolver un parámetro de asidero para la posterior consulta de estado por el proceso de llamada.

Las instrucciones de paso de mensajes del anfitrión pueden, en algunos ejemplos, incluir una instrucción de registro de rutina de devolución de llamada que instruye al dispositivo anfitrión 12 para registrar una devolución de llamada para recibir datos desde un dispositivo especificado de forma asincrónica. Por ejemplo, la instrucción de registro de rutina de devolución de llamada puede instruir a la interfaz de paso de mensajes del anfitrión 26 para invocar una rutina de devolución de llamada en respuesta a la recepción de una señal desde la GPU 14 que indica que la tarea que se está ejecutando en la GPU 14 ha enviado un mensaje al proceso anfitrión 20. La instrucción de registro de rutina de devolución de llamada puede incluir un primer parámetro de entrada que especifica un dispositivo concreto para el que se debería registrar una rutina de devolución de llamada y un segundo parámetro de entrada que especifica la ubicación de la memoria de la rutina de devolución de llamada.

En algunos ejemplos, la interfaz para la instrucción de registro de rutina de devolución de llamada puede adoptar la forma siguiente:

```
clRegistrarDevoluciónDeLlamadaDeDatosFueraDeBanda(
        cl_dispositivo *identificador_dispositivo,
        void(*) (int) punteroDevoluciónDeLlamada)
```

donde clRegistrarDevoluciónDeLlamadaDeDatosFueraDeBanda es el identificador de instrucción, cl_dispositivo *identificador_dispositivo es un parámetro de entrada que especifica un dispositivo de OpenCL particular al cual debería enviarse el mensaje, y void(*) (int) punteroDevoluciónDeLlamada es un parámetro de entrada que especifica la ubicación de la memoria de la rutina de devolución de llamada. La instrucción de registro de rutina de devolución de llamada puede devolver un parámetro indicativo de si se completó con éxito la operación de registro de rutina de devolución de llamada.

Las instrucciones de paso de mensajes del anfitrión pueden, en algunos ejemplos, incluir una instrucción de sondeo que instruye al dispositivo anfitrión 12 para intentar leer datos de un dispositivo especificado. Por ejemplo, la instrucción de sondeo puede instruir a la interfaz de paso de mensajes del anfitrión 26 para sondear la GPU 14 para obtener información de estado de mensaje, indicativa de si la tarea 28 que se está ejecutando en la GPU 14 ha enviado un mensaje. La instrucción de sondeo puede incluir un parámetro de entrada que especifica un dispositivo en particular a sondear y un parámetro de salida que especifica los datos obtenidos, si los hubiera, como resultado del sondeo.

En algunos ejemplos, la interfaz para la instrucción de sondeo puede adoptar la forma siguiente:

```
clIntentarLecturaDeDatosFueraDeBanda(
        cl_dispositivo *identificador_dispositivo,
        int *datos_OOB)
```

donde ClIntentarLecturaDeDatosFueraDeBanda es el identificador de instrucción, cl_dispositivo

*identificador_dispositivo es un parámetro de entrada que especifica un dispositivo de OpenCL particular a ser sondeado, e int *datos_OOB es un parámetro de salida que especifica los datos obtenidos, si los hubiera, como resultado del sondeo. La instrucción de sondeo puede devolver un parámetro indicativo de si se obtuvieron datos con éxito en la operación de sondeo.

De forma similar al proceso anfitrión 20, la tarea 28 puede incluir una o más instrucciones de paso de mensajes de dispositivo que son ejecutadas por un dispositivo informático. Las instrucciones de paso de mensajes de dispositivo pueden incluir una instrucción de envío que instruye a un dispositivo informático para enviar los datos especificados al dispositivo anfitrión 12. Por ejemplo, la instrucción de envío puede instruir a la GPU 14 para enviar un mensaje desde la tarea 28 que se está ejecutando en la GPU 14 al proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12.

La instrucción de envío puede ser o bien una instrucción de envío bloqueante o una instrucción de envío no bloqueante. La instrucción de envío puede incluir, en algunos ejemplos, un primer parámetro de entrada que especifica si la instrucción de envío es una instrucción de envío bloqueante o una instrucción de envío no bloqueante. Una instrucción de envío bloqueante puede detener el proceso de llamada, por ejemplo, la tarea 28 que se está ejecutando en la GPU 14, y esperar a que la operación de envío se complete antes de volver al proceso de llamada. Una instrucción de envío no bloqueante puede volver al proceso de llamada sin tener que esperar hasta que se complete la operación de envío. Por ejemplo, la instrucción de envío no bloqueante puede devolver un asidero a la operación de envío particular, que posteriormente puede ser consultado por el proceso de llamada para determinar si la operación de envío se ha realizado con éxito. Una operación de envío no bloqueante puede fallar y, en el caso de fallo, el proceso de llamada puede tener que emitir de nuevo la instrucción de envío para volver a intentar la operación de envío. La instrucción de envío puede incluir un segundo parámetro de entrada que especifica el contenido del mensaje a ser enviado al dispositivo anfitrión.

En algunos ejemplos, la interfaz para la instrucción de envío puede adoptar la forma siguiente:

```
enviar_datosoob (
        bool bloqueante,
        int datos)
```

donde enviar_datosoob es el identificador de instrucción, bool bloqueante es un parámetro de entrada que especifica si la instrucción es una instrucción de envío bloqueante o una instrucción de envío no bloqueante, e int datos es un parámetro de entrada que especifica el contenido del mensaje a enviar. En el caso de una instrucción bloqueante, la instrucción puede devolver un parámetro indicativo de si se completó con éxito la operación de envío. En el caso de una instrucción no bloqueante, la instrucción puede devolver un parámetro de asidero para la posterior consulta de estado por el proceso de llamada.

Las instrucciones de paso de mensajes de dispositivo pueden, en algunos ejemplos, incluir una instrucción de recepción que instruye a un dispositivo informático para recibir datos desde el dispositivo anfitrión 12. Por ejemplo, la instrucción de recepción puede instruir a la GPU 14, por ejemplo, a la interfaz de paso de mensajes de dispositivo 30, para proporcionar a la tarea 28 que se está ejecutando en la GPU 14 un mensaje enviado a la tarea 28 desde el proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12, si está disponible. Una instrucción de este tipo puede utilizarse para prestar soporte a un mecanismo de sondeo.

La instrucción de recepción puede ser o una instrucción de recepción bloqueante o una instrucción de recepción no bloqueante. La instrucción de recepción puede incluir, en algunos ejemplos, un parámetro de entrada que especifica si la instrucción de recepción es una instrucción de recepción bloqueante o una instrucción de recepción no bloqueante. Una instrucción de recepción bloqueante puede detener el proceso de llamada, por ejemplo, la tarea 28 que se está ejecutando en la GPU 14, y esperar hasta que un mensaje esté disponible antes de volver al proceso de llamada. Una instrucción de recepción no bloqueante puede volver al proceso de llamada sin esperar hasta que un mensaje esté disponible. Por ejemplo, si un mensaje está disponible, la instrucción de recepción no bloqueante puede devolver el mensaje. Sin embargo, si un mensaje no está disponible, la instrucción de recepción no bloqueante puede fallar. En caso de fallo, el proceso de llamada puede tener que emitir de nuevo la instrucción de recepción para volver a intentar la operación de recepción. La instrucción puede incluir recibir un parámetro de salida que especifica los datos obtenidos, si los hubiera, como resultado de la operación de recepción.

En algunos ejemplos, la interfaz para la instrucción de recepción puede adoptar la forma siguiente:

```
recibir_datosoob (
        bool bloqueante,
        int datos)
```

donde recibir_datosoob es el identificador de instrucción, bool bloqueante es un parámetro de entrada que especifica si la instrucción es una instrucción de recepción bloqueante o una instrucción de recepción no bloqueante, e int datos es un parámetro de salida que especifica los datos obtenidos, si los hubiera, como resultado de la operación

de recepción. La instrucción puede devolver un parámetro indicativo de si se completó con éxito la operación de recepción.

La interfaz de cola de comandos 24 está configurada para poner los comandos en la cola de comandos 32. Por ejemplo, la interfaz de cola de comandos 24 puede recibir una o más instrucciones de puesta en cola desde el proceso anfitrión 20, y colocar uno o más comandos en la cola de comandos 32 en respuesta a la recepción de una o más instrucciones de puesta en cola del proceso anfitrión 20. Las una o más instrucciones de puesta en cola pueden incluir instrucciones de puesta en cola de ejecuciones de tareas e instrucciones de puesta en cola de transferencias de datos que instruyen a la interfaz de cola de comandos 24 para poner en cola, respectivamente, los comandos de ejecución de tareas y los comandos de transferencia de datos.

La interfaz de cola de comandos 24 también está configurada para ejecutar los comandos almacenados en la cola de comandos 32. Para los comandos de transferencia de datos, la interfaz de cola de comandos 24 puede transferir datos entre el espacio de memoria del anfitrión y el espacio de memoria del dispositivo. Por ejemplo, para un comando de memoria intermedia de escritura, la interfaz de cola de comandos 24 puede transferir los datos almacenados en una parte del espacio de memoria del anfitrión a un objeto de memoria intermedia asignado en el espacio de memoria del dispositivo. Como otro ejemplo, para un comando de memoria intermedia de lectura, la interfaz de cola de comandos 24 puede transferir los datos almacenados en un objeto de memoria intermedia asignado en el espacio de memoria del dispositivo a una parte del espacio de memoria del anfitrión. El espacio de memoria del dispositivo puede corresponder a un dispositivo al cual está asociada la cola de comandos 32.

Para los comandos de ejecución de tareas, la interfaz de cola de comandos 24 puede provocar el inicio de la ejecución de la tarea en un dispositivo asociado con una cola de comandos. Por ejemplo, en el ejemplo de la FIG. 1, la cola de comandos 32 está asociada con la GPU 14 dentro del contexto de módulo de tiempo de ejecución 22. Por lo tanto, cuando se está ejecutando un comando de ejecución de tareas, la interfaz de cola de comandos 24 puede hacer que una tarea empiece a ejecutarse en la GPU 14. En algunos ejemplos, la interfaz de cola de comandos 24 puede hacer que la tarea empiece a ejecutarse en la GPU 14 mediante la colocación de uno o más comandos en una cola de comandos local contenida dentro de la GPU 14. En otros ejemplos, la interfaz de cola de comandos 24 puede hacer que la tarea empiece a ejecutarse en la GPU 14 mediante el envío de una o más instrucciones a la GPU 14, instruyendo a la GPU 14 para comenzar la ejecución de la tarea. La interfaz de cola de comandos 24 puede utilizar la red de interconexión 18 para comunicarse con la GPU 14, la memoria 16 y los espacios de memoria del anfitrión y del dispositivo.

En algunos ejemplos, la interfaz de cola de comandos 24 puede ejecutar los comandos en orden. En dichos ejemplos, si un primer comando se pone en cola antes de un segundo comando, la ejecución del segundo comando comienza después de que el primer comando haya finalizado su ejecución. En ejemplos adicionales, la interfaz de cola de comandos 24 puede ejecutar los comandos de forma desordenada. En dichos ejemplos, si un primer comando se pone en cola antes de un segundo comando, la ejecución del segundo comando no necesariamente empieza después de que el primer comando haya finalizado su ejecución.

La interfaz de paso de mensajes del anfitrión 26 está configurada para ejecutar una o más instrucciones de paso de mensajes recibidas desde el proceso anfitrión 20. Por ejemplo, en respuesta a la recepción de una o más instrucciones de paso de mensajes desde el proceso anfitrión 20, la interfaz de paso de mensajes del anfitrión 26 puede pasar uno o más mensajes entre el proceso anfitrión 20 que se están ejecutando en el dispositivo anfitrión 12 y la tarea 28 que se está ejecutando en la GPU 14 mientras la tarea 28 se está ejecutando en la GPU 14. En algunos ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede ejecutar una o más instrucciones de paso de mensajes sin colocar ningún comando en cola de comandos 32.

De acuerdo con un primer ejemplo, en respuesta a la recepción de una instrucción de envío desde el proceso anfitrión 20, la interfaz de paso de mensajes del anfitrión 26 puede enviar un mensaje desde el proceso anfitrión 20 a la tarea 28 mientras la tarea 28 se está ejecutando en la GPU 14. Por ejemplo, la interfaz de paso de mensajes del anfitrión 26 puede componer un mensaje saliente en base a los datos de mensajes contenidos dentro de la instrucción de envío, y transferir el mensaje saliente, a través de la red de interconexión 18, a un dispositivo especificado en la instrucción de envío, por ejemplo, la GPU 14, para su entrega a una tarea que se está ejecutando en el dispositivo especificado, por ejemplo, la tarea 28.

De acuerdo con un segundo ejemplo, en respuesta a la recepción de una instrucción de registro de rutina de devolución de llamada del proceso anfitrión 20, la interfaz de paso de mensajes del anfitrión 26 puede asociar la rutina de devolución de llamada especificada en la instrucción con una señal desde el dispositivo especificado en la instrucción, por ejemplo, la GPU 14, lo cual indica que la tarea, por ejemplo, la tarea 28, que se está ejecutando en el dispositivo especificado ha enviado un mensaje. En algunos ejemplos, la señal desde el dispositivo puede ser una señal de interrupción. La señal de interrupción, en algunos ejemplos, puede administrarse mediante una línea de señal de interrupción dedicada. En respuesta a la recepción de la señal desde el dispositivo especificado que indica que la tarea que se está ejecutando en el dispositivo ha enviado un mensaje, la interfaz de paso de mensajes del anfitrión 26 puede iniciar la ejecución de la rutina de devolución de llamada especificada en la instrucción de registro de rutina de devolución de llamada. La rutina de devolución de llamada puede obtener el mensaje enviado por la

tarea, por ejemplo, la tarea 28, desde el dispositivo especificado, por ejemplo, la GPU 14, y devolver el mensaje al proceso anfitrión 20 para su posterior procesamiento.

De acuerdo con un tercer ejemplo, en respuesta a la recepción de una instrucción de sondeo, la interfaz de paso de mensajes del anfitrión 26 puede sondear el dispositivo especificado en la instrucción, por ejemplo, la GPU 14, para obtener información sobre el estado del mensaje. La interfaz de paso de mensajes del anfitrión 26 puede utilizar la red de interconexión 18 u otra vía de comunicación basada en hardware para sondear el dispositivo. Si la información de estado de mensaje indica que una tarea, por ejemplo, la tarea 28, que se está ejecutando en el dispositivo especificado, por ejemplo, la GPU 14, ha enviado el mensaje, la interfaz de paso de mensajes del anfitrión 26 puede obtener el mensaje desde el dispositivo especificado y devolver el mensaje al proceso anfitrión 20 para su posterior procesamiento.

La interfaz de paso de mensajes de dispositivo 30 está configurada para ejecutar una o más instrucciones de paso de mensajes de dispositivo recibidas desde la tarea 28. Por ejemplo, en respuesta a la recepción de una o más instrucciones de paso de mensajes de dispositivo desde la tarea 28, la interfaz de paso de mensajes de dispositivo 30 puede pasar uno o más mensajes entre la tarea 28 que se está ejecutando en la GPU 14 y el proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12 mientras la tarea 28 se está ejecutando en la GPU 14.

De acuerdo con un primer ejemplo, en respuesta a la recepción de una instrucción de envío, la interfaz de paso de mensajes de dispositivo 30 puede enviar un mensaje desde la tarea 28 que se está ejecutando en la GPU 14 al proceso anfitrión 20 que se está ejecutando en el dispositivo anfitrión 12. Por ejemplo, la interfaz de paso de mensajes de dispositivo 30 puede componer un mensaje saliente sobre la base de datos de mensajes contenidos dentro de la instrucción de envío, y transferir el mensaje saliente, a través de la red de interconexión 18, al dispositivo anfitrión 12 para su entrega al proceso anfitrión 20.

De acuerdo con un segundo ejemplo, en respuesta a la recepción de una instrucción de recepción desde la tarea 28, la interfaz de paso de mensajes de dispositivo 30 puede determinar si un mensaje del proceso anfitrión 20 está disponible. En algunos ejemplos, la interfaz de paso de mensajes de dispositivo 30 puede comprobar uno o más registros accesibles por el anfitrión para determinar si un mensaje está disponible. Si un mensaje del proceso anfitrión 20 está disponible, la interfaz de paso de mensajes de dispositivo 30 puede proporcionar el mensaje a la tarea 28.

Aunque la interfaz de cola de comandos 24 y la interfaz de paso de mensajes del anfitrión 26 se ilustran como componentes que son independientes del proceso anfitrión en la FIG. 1, en algunos ejemplos, la funcionalidad de una entre la interfaz de cola de comandos 24 y la interfaz de paso de mensajes del anfitrión 26, o ambas, pueden compilarse parcialmente y/o completamente en el proceso anfitrión 20. De manera similar, en algunos ejemplos, la funcionalidad de la interfaz de paso de mensajes de dispositivo 30 puede compilarse parcialmente y/o completamente en la tarea 28.

Para facilitar la ilustración, el sistema informático ejemplar 10 ilustrado en la FIG. 1 describe las técnicas de paso de mensajes de esta divulgación usando la GPU 14 como un dispositivo informático. Debería reconocerse, sin embargo, que las técnicas de esta divulgación pueden ser aplicadas a sistemas informáticos de múltiples procesadores que tienen dispositivos informáticos distintos a una GPU, además o en lugar de la GPU 14. En algunos ejemplos, los dispositivos informáticos pueden ser dispositivos informáticos de OpenCL. Un dispositivo informático de OpenCL incluye una o más unidades informáticas. Cada una de las unidades informáticas incluye uno o más elementos de procesamiento. Por ejemplo, una unidad informática puede ser un grupo de elementos de procesamiento, por ejemplo, ALU, que tiene una memoria compartida en chip que puede ser utilizada por todos los elementos de procesamiento en la unidad informática. Un elemento de trabajo puede ser una entre una pluralidad de ejecuciones paralelas de un núcleo o tarea invocada en un dispositivo informático de OpenCL por un comando colocado en una cola de comandos. Cada elemento de trabajo puede ejecutarse en un elemento de procesamiento individual en una unidad informática en paralelo con otros elementos de trabajo que se están ejecutando en otros elementos de procesamiento. Un grupo de trabajo puede ser una colección de uno o más elementos de trabajo que se procesan en una única unidad informática dentro del dispositivo informático como parte de un único comando de ejecución de núcleo. Un servidor de OpenCL puede ser una CPU central de la plataforma que se utiliza para ejecutar la capa de tiempo de ejecución de OpenCL.

La API de OpenCL puede proporcionar un conjunto común de interfaces para la interacción entre el dispositivo anfitrión y los diferentes tipos de dispositivos informáticos. Por ejemplo, la API de OpenCL puede proporcionar una interfaz común para la interacción entre un dispositivo anfitrión y un dispositivo informático de GPU, y el dispositivo anfitrión y un dispositivo informático que no sea una GPU. La API de OpenCL permite que el servidor utilice una interfaz común para ejecutar tareas (por ejemplo, núcleos de OpenCL) en los distintos dispositivos informáticos. En algunos ejemplos, las tareas pueden ser tareas de cálculo de propósito general, y la API de OpenCL puede permitir que el anfitrión haga que la tarea informática de propósito general se ejecute en un dispositivo informático de GPU.

El sistema informático ejemplar 10 mostrado en la FIG. 1 ilustra una infraestructura y técnicas para facilitar el paso de mensajes y/o la señalización fuera de banda entre un dispositivo anfitrión y un dispositivo informático. En otros

sistemas informáticos ejemplares, sin embargo, las técnicas pueden extenderse inmediatamente para proporcionar el paso de mensajes en proceso entre diferentes dispositivos informáticos (por ejemplo, dispositivos informáticos de OpenCL) en un sistema informático que tiene más de un dispositivo informático. En dichos ejemplos, pueden conectarse una o más líneas de interrupción entre diferentes dispositivos informáticos.

La FIG. 2 es un diagrama de bloques que ilustra una GPU ejemplar 40 que puede utilizarse en el sistema informático 10 de la FIG. 1 de acuerdo con esta divulgación. En algunos ejemplos, la GPU 40 puede utilizarse para implementar la GPU 14 ilustrada en la FIG. 1. La GPU 40 incluye un bloque de procesamiento de GPU 42, registros de GPU accesibles por el anfitrión 44 y un controlador de bus 46. La GPU 40 puede estar acoplada comunicativamente a otros uno o más dispositivos anfitriones o dispositivos informáticos a través de la red de interconexión 18.

El bloque de procesamiento de la GPU 42 está configurado para ejecutar las tareas y para facilitar el paso de mensajes entre las tareas que se están ejecutando en el bloque de procesamiento de la GPU 42 y los procesos que se están ejecutando en otros dispositivos anfitriones o informáticos. El bloque de procesamiento de la GPU 42 está acoplado comunicativamente a los registros de GPU accesibles por el anfitrión 44, por ejemplo, mediante una o más líneas de control y/o datos. En algunos ejemplos, el bloque de procesamiento de la GPU 42 puede denominarse un bloque de unidad lógica y aritmética (ALU). El bloque de procesamiento de la GPU 42 incluye una tarea 48, un módulo de paso de mensajes 50, un registro de datos entrantes 52 y un registro de datos salientes 54.

Los registros de GPU accesibles por el anfitrión 44 están configurados para almacenar datos que pueden comunicarse hacia o desde un dispositivo anfitrión. Los registros de GPU accesibles por el anfitrión 44 incluyen un registro de estado de mensajes 56, un registro de recuento de mensajes 58, un registro de mensajes entrantes 60, un registro de mensajes salientes 62, un registro de estado de interrupción 64 y un registro de confirmación de interrupción 66. Cada uno de los registros de GPU accesibles por el anfitrión 44 puede ser accesible mediante un dispositivo anfitrión, por ejemplo, el dispositivo anfitrión 12 en la FIG. 1. En algunos ejemplos, los registros de GPU accesibles por el anfitrión 44 pueden ser registros correlacionados con memoria, es decir, registros que están correlacionados con, y direccionables en, el espacio de memoria de un dispositivo anfitrión. En otros ejemplos, los registros de GPU accesibles por el anfitrión 44 pueden ser registros correlacionados con entrada / salida (correlacionados con E/S), es decir, registros correlacionados con el espacio de E/S de un dispositivo anfitrión. Los registros de GPU accesibles por el anfitrión 44 están acoplados comunicativamente al bloque de procesamiento de la GPU 42 mediante una o más líneas de control y/o datos. Los registros de GPU accesibles por el anfitrión 44 también están acoplados comunicativamente al controlador de bus 46 a través de la red de interconexión 18.

La tarea 48 puede ejecutarse en uno o más procesadores programables. En algunos ejemplos, el bloque de procesamiento de la GPU 42 puede incluir múltiples procesadores o elementos de procesamiento configurados para ejecutar múltiples instancias de ejecución de la tarea 48. La tarea 48 puede ser esencialmente similar a la tarea 28 descrita anteriormente con respecto a la FIG. 1, y por lo tanto no se describirá con más detalle.

El módulo de paso de mensajes 50 está configurado para controlar las operaciones de paso de mensajes realizadas por la GPU 40. El módulo de paso de mensajes 50 puede implementarse en hardware, software, firmware o cualquier combinación de los mismos. En algunos ejemplos, si una parte de, o toda, la funcionalidad del módulo de paso de mensajes 50 se implementa en software, las instrucciones de software para dicha implementación se pueden incluir dentro del mismo archivo ejecutable que el archivo ejecutable que contiene instrucciones de software para la tarea 48. El módulo de paso de mensajes 50 está acoplado comunicativamente a la tarea 48, el módulo de paso de mensajes 50, el registro de datos entrantes 52 y el registro de datos salientes 54.

El módulo de paso de mensajes 50 puede pasar uno o más mensajes, mediante los registros de GPU accesibles por el anfitrión 44, entre la tarea 48 que se está ejecutando en uno o más procesadores y un proceso que se está ejecutando en un dispositivo anfitrión, mientras la tarea 48 se está ejecutando en uno o más procesadores, y en respuesta a la recepción de una o más instrucciones de paso de mensajes desde la tarea 48. En algunos ejemplos, las una o más instrucciones de paso de mensajes pueden incluir una instrucción de envío que instruye al módulo de paso de mensajes 50 para enviar un mensaje desde la tarea 48 a un proceso que se está ejecutando en un dispositivo anfitrión. En tales ejemplos, el módulo de paso de mensajes 50 puede almacenar datos de mensajes asociados con el mensaje en uno de los registros de GPU accesibles por el anfitrión 44. En ejemplos adicionales, las una o más instrucciones de paso de mensajes pueden incluir una instrucción de recepción que instruye al módulo de paso de mensajes 50 para proporcionar a la tarea 48 un mensaje enviado a la tarea 48 desde un proceso que se está ejecutando en un dispositivo anfitrión, si está disponible. En tales ejemplos, el módulo de paso de mensajes 50 puede obtener datos de mensajes asociados con el mensaje desde uno o más de los registros de GPU accesibles por el anfitrión 44.

El registro de datos entrantes 52, en el ejemplo de la FIG. 2, es un registro de hardware que almacena los datos entrantes recibidos desde un dispositivo externo mediante un registro de mensajes entrantes 60. El registro de datos entrantes 52 también puede almacenar un bit de estado que indica si los datos en un registro de datos entrantes 52 se han consumido y/o si los datos en el registro de datos entrantes 52 están disponibles para su lectura. El registro de datos entrantes 52 está acoplado comunicativamente al registro de mensajes entrantes 60 mediante una o más líneas de datos. En algunos ejemplos, el número de líneas de datos puede ser igual al número de bits en el registro

de datos entrantes 52, los cuales pueden ser ambos iguales al número de bits en el mensaje. En otros ejemplos, el número de bits puede ser de 32 bits. En algunos ejemplos, el bloque de procesamiento de la GPU 42 puede implementar una memoria intermedia interna del tipo 'primero en entrar, primero en salir' (FIFO) para almacenar una pluralidad de mensajes entrantes recibidos desde el registro de datos entrantes 52.

El registro de datos salientes 54, en el ejemplo de la FIG. 2, es un registro de hardware que almacena los datos salientes recibidos desde una o más instrucciones de paso de mensajes emitidas por la tarea 48. El registro de datos salientes 54 está acoplado comunicativamente al registro de mensajes salientes 62 mediante una o más líneas de datos. En algunos ejemplos, el número de líneas de datos puede ser igual al número de bits en el registro de datos salientes 54, los cuales pueden ser ambos iguales al número de bits en el mensaje. En algunos ejemplos, el registro de datos salientes 54 y el registro de mensajes salientes 62 pueden estar configurados de tal manera que, cuando el módulo de paso de mensajes 50 escribe datos en el registro de datos salientes 54, el registro de mensajes salientes 62 se actualiza automáticamente con los datos escritos en el registro de datos salientes 54. En algunos ejemplos, el bloque de procesamiento de la GPU 42 puede implementar una memoria intermedia del tipo 'primero en entrar, primero en salir' (FIFO) para almacenar una pluralidad de mensajes salientes a escribir en el registro de datos salientes 54.

El registro de estado de mensajes 56, en el ejemplo de la FIG. 2, está configurado para almacenar datos indicativos de si un mensaje entrante fue aceptado por la GPU 40. El registro de estado de mensajes 56 puede ser utilizado por un dispositivo anfitrión para determinar si un mensaje fue transmitido con éxito y, en algunos ejemplos, para implementar un mecanismo de retroceso y/o desborde. Después de aceptar un mensaje entrante, el módulo de paso de mensajes 50 puede fijar el registro de estado de mensajes 56 en un valor particular, lo cual indica que el mensaje entrante fue aceptado.

El registro de recuento de mensajes 58, en el ejemplo de la FIG. 2, está configurado para almacenar datos indicativos de si el registro de mensajes entrantes 60 contiene un mensaje entrante. En algunos ejemplos, el registro de recuento de mensajes 58 puede enviar una señal al módulo de paso de mensajes 50 para indicar la llegada de mensajes cuando el registro de recuento de mensajes 58 ha sido incrementado por un dispositivo anfitrión. En algunos casos, la señal puede ser una línea de pulso de 1 bit. En ejemplos adicionales, el módulo de paso de mensajes 50 puede disminuir el registro de recuento de mensajes 58 después de leer el mensaje procedente del registro de datos entrantes 52.

El registro de mensajes entrantes 60, en el ejemplo de la FIG. 2, está configurado para almacenar datos de mensajes entrantes. Por ejemplo, un dispositivo anfitrión puede colocar datos de mensajes entrantes en el registro de mensajes entrantes 60 con el fin de enviar un mensaje a la tarea 48. El registro de mensajes entrantes 60 está acoplado comunicativamente al registro de datos entrantes 52.

El registro de mensajes salientes 62, en el ejemplo de la FIG. 2, está configurado para almacenar datos de mensajes salientes recibidos desde el registro de datos salientes 54. El registro de mensajes salientes 62 puede actualizar automáticamente los datos del registro de mensajes salientes 62 para corresponderse con el registro de datos salientes 54 cuando se escriban los nuevos datos en el registro de datos salientes 54. En algunos ejemplos, el módulo de paso de mensajes 50 puede generar una señal de interrupción en respuesta a un mensaje saliente que se escribe en el registro de mensajes salientes 62. La señal de interrupción puede enviarse a un dispositivo anfitrión e indicar que el módulo de paso de mensajes 50 ha enviado un mensaje.

El registro de estado de interrupción 64, en el ejemplo de la FIG. 2, está configurado para almacenar un bit de estado indicativo de si un mensaje saliente se ha escrito en el registro de mensajes salientes 62. Por ejemplo, el registro de estado de interrupción 64 y el registro de mensajes salientes 62 pueden estar configurados de tal manera que un bit de estado en el registro de estado de interrupción 64 se active cuando un mensaje saliente se escribe en el registro de mensajes salientes 62. El bit de estado puede permitir que un proceso que se está ejecutando en un dispositivo anfitrión sondee la GPU 40 para ver si hay un mensaje disponible.

El registro de confirmación de interrupción 66, en el ejemplo de la FIG. 2, está configurado para almacenar un bit de confirmación indicativo de si el dispositivo anfitrión ha leído el mensaje saliente almacenado en el registro de mensajes salientes 62. Por ejemplo, el registro de mensajes salientes 62 y el registro de confirmación de interrupción 66 pueden estar configurados de tal manera que, cuando un mensaje saliente se escribe en el registro de mensajes salientes 62, se activa el bit de confirmación en el registro de confirmación de interrupción 66. En tal ejemplo, después de que un dispositivo anfitrión lea el registro de mensajes salientes 62, el dispositivo anfitrión puede borrar el bit de confirmación indicando de este modo que el dispositivo anfitrión ha leído el mensaje saliente y que un nuevo mensaje saliente puede escribirse en el registro de mensajes salientes 62. El bit de confirmación puede utilizarse para implementar un esquema de control de flujo para datos de mensajes salientes.

El controlador de bus 46, en el ejemplo de la FIG. 2, está configurado para permitir a dispositivos externos acceder a los registros de GPU accesibles por el anfitrión 44 a través de la red de interconexión 18. Por ejemplo, el controlador del bus 46 puede multiplexar y demultiplexar señales de bus y realizar diversas operaciones de recepción y transmisión especificadas por las señales del bus. El controlador de bus 46 puede funcionar de acuerdo con uno o

más estándares de buses públicos o de propiedad industrial.

A continuación se describirán diversas técnicas para el paso de mensajes en los sistemas informáticos de múltiples procesadores, de acuerdo con ciertos aspectos de esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica las técnicas ejemplares mostradas en las FIGs. 3 a 19 Para facilitar la explicación, las técnicas pueden describirse con respecto a los componentes del sistema informático ejemplar 10 que se muestra en la FIG. 1, pero se debería entender que las técnicas pueden realizarse en otros sistemas con los mismos, o diferentes, componentes en la misma configuración o en una diferente. En ejemplos adicionales, pueden describirse algunas de las técnicas mostradas en las FIGs. 3 a 19 con respecto a componentes específicos de la GPU 40 de la FIG. 2. Una vez más, se debería entender que la FIG. 2 es solo un ejemplo de una GPU que puede ser capaz de implementar las técnicas de esta divulgación, y que tales técnicas pueden ser realizadas por otras GPU con los mismos, o diferentes, componentes en la misma configuración o una diferente.

La FIG. 3 ilustra una técnica ejemplar para el paso de mensajes en un entorno de plataforma de múltiples procesadores de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 3. La interfaz de cola de comandos 24 coloca un comando de transferencia de memoria en la cola de comandos 32 (70). La interfaz de cola de comandos 24 coloca un comando de ejecución de tareas en la cola de comandos 32 (72). La interfaz de cola de comandos 24 ejecuta el comando de ejecución de tareas para iniciar la ejecución de la tarea en la GPU 14 (74). La interfaz de paso de mensajes del anfitrión 26 pasa uno o más mensajes entre el dispositivo anfitrión 12 y la GPU 14 mientras la tarea 28 se está ejecutando en la GPU 14 (76). Por ejemplo, la interfaz de paso de mensajes del anfitrión 26 puede pasar un mensaje a la GPU 14 que se origina a partir de una o más instrucciones de envío emitidas por el proceso anfitrión 20. Las una o más instrucciones de envío pueden especificar que la GPU 14, o una tarea que se está ejecutando en la GPU 14, es el destino para el mensaje.

La FIG. 4 muestra una técnica ejemplar para la ejecución de una instrucción de envío emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 4. La interfaz de paso de mensajes del anfitrión 26 recibe una instrucción de envío desde el proceso anfitrión 20 (78). La interfaz de paso de mensajes del anfitrión 26 genera un mensaje saliente en base a los datos de mensajes contenidos en la instrucción de envío (80). En algunos ejemplos, el mensaje saliente puede ser idéntico a los datos de mensajes contenidos en la instrucción de envío. En ejemplos adicionales, la interfaz de paso de mensajes del anfitrión 26 puede anexar uno o más elementos de información de cabecera y/o información de encaminamiento a los datos de mensajes contenidos en la instrucción de envío para generar el mensaje saliente. En otros ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede realizar una o más operaciones de codificación o de transformación en los datos de mensajes contenidos en la instrucción de envío para generar el mensaje saliente. La interfaz de paso de mensajes del anfitrión 26 puede enviar el mensaje saliente a la GPU 14 (82).

La interfaz de paso de mensajes del anfitrión 26 puede determinar si la instrucción de envío es una instrucción bloqueante o una instrucción no bloqueante (84). En algunos ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede tomar la determinación de si la instrucción de envío es una instrucción bloqueante o no bloqueante, en base a un parámetro de entrada especificado en la instrucción de envío. En otros ejemplos, pueden utilizarse dos tipos diferentes de instrucciones de envío, y la interfaz de paso de mensajes del anfitrión 26 puede tomar la determinación de si la instrucción de envío es una instrucción bloqueante o no bloqueante, en base al tipo de instrucción, por ejemplo, el código de operación (codeop) de la instrucción. Si la interfaz de paso de mensajes del anfitrión 26 determina que la instrucción de envío es una instrucción no bloqueante, la interfaz de paso de mensajes del anfitrión 26 puede devolver un asidero al proceso de llamada (86). El asidero puede permitir que el proceso de llamada consulte el asidero para determinar si el mensaje ha sido enviado con éxito en un momento posterior. Si la consulta posterior indica que el envío ha fallado, el proceso de llamada puede tener que emitir una instrucción de envío posterior para volver a intentar la operación de envío. En algunos ejemplos, el proceso de llamada puede implementar una rutina de retroceso o un mecanismo de desborde en respuesta a una operación de envío fallida.

Si la interfaz de paso de mensajes del anfitrión 26 determina que la instrucción de envío es una instrucción bloqueante, la interfaz de paso de mensajes del anfitrión 26 puede determinar si el mensaje saliente fue recibido con éxito por la GPU 14 (88). Si la interfaz de paso de mensajes del anfitrión 26 determina que el mensaje saliente se ha recibido con éxito, la interfaz de paso de mensajes del anfitrión 26 puede devolver un valor al proceso de llamada indicando que el mensaje contenido en la instrucción de envío se ha enviado con éxito (90). De lo contrario, si la interfaz de paso de mensajes del anfitrión 26 determina que el mensaje saliente no se ha recibido con éxito, la interfaz de paso de mensajes del anfitrión 26 puede proceder a procesar el bloque 82 y volver a enviar el mensaje saliente a la GPU 14. La instrucción de bloqueo puede completarse, en algunos ejemplos, cuando la interfaz de paso de mensajes del anfitrión 26 determina que el mensaje fue recibido con éxito o que se ha alcanzado un número de umbral de intentos de entrega sin éxito.

La FIG. 5 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de proceso 82 en la FIG. 4 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en

la FIG. 5. La interfaz de paso de mensajes del anfitrión 26 puede colocar o almacenar el mensaje saliente en el registro de mensajes entrantes 60 de la GPU 40 (92). La interfaz de paso de mensajes del anfitrión 26 puede incrementar el registro de recuento de mensajes 58 de la GPU 40 para indicar al módulo de paso de mensajes 50 en la GPU 14 que un nuevo mensaje ha llegado (94). En algunos ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede utilizar hardware de registro correlacionado con memoria y/o hardware de registro correlacionado con E/S, conocido en la técnica, para llevar a cabo uno o más de los bloques de procesos 92 y 94.

La FIG. 6 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de decisión 88 en la FIG. 4 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 6. La interfaz de paso de mensajes del anfitrión 26 puede comprobar un bit de estado en el registro de estado de mensajes 56 de la GPU 40 (96). La interfaz de paso de mensajes del anfitrión 26 puede determinar si el mensaje enviado fue aceptado por la GPU 14, en base al bit de estado en el registro de estado de mensajes 56 (98). Si el bit de estado indica que el mensaje enviado fue aceptado por la GPU 14, la interfaz de paso de mensajes del anfitrión 26 puede determinar que el mensaje saliente se ha recibido con éxito (100). Por otro lado, si el bit de estado indica que el mensaje enviado no fue aceptado por la GPU 14, la interfaz de paso de mensajes del anfitrión 26 puede determinar que el mensaje saliente no se ha recibido con éxito (102).

La FIG. 7 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de un mensaje recibido en un dispositivo informático, tal como, por ejemplo, una GPU. En algunos ejemplos, la GPU 40 de la FIG. 2 puede utilizarse para poner en práctica la técnica ejemplar que se muestra en la FIG. 7. El módulo de paso de mensajes 50 en la GPU 40 recibe una señal de llegada de mensaje (104). Por ejemplo, el registro de recuento de mensajes 58 puede estar configurado de tal manera que cada vez que un dispositivo anfitrión incrementa un registro de recuento de mensajes 58, se envía un pulso de llegada de mensaje al módulo de paso de mensajes 50. El módulo de paso de mensajes 50 puede hacer que los datos almacenados en el registro de mensajes entrantes 60 sean transferidos al registro de datos entrantes 52 (106). Por ejemplo, el módulo de paso de mensajes 50 puede emitir una señal de control al registro de datos entrantes 52, haciendo que el registro de datos entrantes 52 sobrescriba los datos actuales almacenados en el registro de datos entrantes 52 con los datos almacenados en el registro de mensajes entrantes 60. El módulo de paso de mensajes 50 puede activar el bit de estado en el registro de datos entrantes 52 para indicar que los datos están disponibles en el registro de datos entrantes 52, por ejemplo, no están consumidos (108). El módulo de paso de mensajes 50 puede activar un bit de estado en el registro de estado de mensajes 56 para indicar que el mensaje entrante ha sido aceptado por la GPU 40 (110).

La FIG. 8 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de recepción emitida por una tarea que se está ejecutando en un dispositivo informático de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 8. La interfaz de paso de mensajes de dispositivo 30 recibe una instrucción de recepción desde la tarea 28 (112). La interfaz de paso de mensajes de dispositivo 30 determina si un mensaje está disponible a partir de un dispositivo anfitrión (114).

Si el módulo de paso de mensajes 50 determina que el mensaje no está disponible, el módulo de paso de mensajes 50 puede determinar si la instrucción de recepción es una instrucción de recepción bloqueante o una instrucción de recepción no bloqueante (116). En algunos ejemplos, el módulo de paso de mensajes 50 puede tomar la determinación de si la instrucción de recepción es una instrucción bloqueante o no bloqueante, en base a un parámetro de entrada especificado en la instrucción de recepción. En otros ejemplos, pueden utilizarse dos tipos diferentes de instrucciones de recepción, y el módulo de paso de mensajes 50 pueden tomar la determinación de si la instrucción de recepción es una instrucción bloqueante o no bloqueante, en base al tipo de instrucción, por ejemplo, el código de operación (codeop) de la instrucción. Si el módulo de paso de mensajes 50 determina que la instrucción de recepción es una instrucción bloqueante, el módulo de paso de mensajes 50 puede volver al bloque de decisión 114 para determinar si un mensaje entrante está disponible. De lo contrario, si el módulo de paso de mensajes 50 determina que la instrucción es una instrucción no bloqueante, el módulo de paso de mensajes 50 puede devolver un valor al proceso de llamada indicando que la instrucción de recepción ha fallado (118).

Si el módulo de paso de mensajes 50 determina que un mensaje está disponible en el dispositivo anfitrión, el módulo de paso de mensajes 50 puede devolver los datos de mensajes al proceso de llamada (120). El módulo de paso de mensajes 50 determina si los datos de mensajes deberían ser marcados como consumidos (122). El módulo de paso de mensajes 50 puede determinar si los datos deberían marcarse como consumidos en base a una o más modalidades de consumo. En algunos ejemplos, la modalidad de consumo puede estar señalada por cable a la GPU 14. En otros ejemplos, la modalidad de consumo puede ser programable mediante la tarea 28 y/o el proceso anfitrión 20. Por ejemplo, una instrucción de envío y/o recepción en la tarea 28 o el proceso anfitrión 20 puede contener un parámetro que especifica una modalidad de consumo particular. Por ejemplo, una modalidad de consumo puede especificar que los datos de mensajes deberían marcarse como consumidos cuando al menos una instancia de ejecución de la tarea ha leído los datos. Como otro ejemplo, una modalidad de consumo puede especificar que los datos de mensajes deberían marcarse como consumidos cuando al menos un número de umbral de instancias de ejecución de la tarea han leído los datos.

Si el módulo de paso de mensajes 50 determina que los datos de mensajes deberían marcarse como consumidos, el módulo de paso de mensajes 50 puede borrar los datos de mensajes (124). Por ejemplo, el módulo de paso de mensajes 50 puede borrar un bit de estado en el registro de datos entrantes 52. Por otro lado, si el módulo de paso de mensajes 50 determina que los datos de mensajes no deberían marcarse como consumidos, el módulo de paso de mensajes 50 puede mantener los datos de mensajes (126). Por ejemplo, el módulo de paso de mensajes 50 puede no borrar el bit de estado en el registro de datos entrantes 52.

La FIG. 9 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de decisión 114 en la FIG. 8 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 9. El módulo de paso de mensajes 50 puede leer un bit de estado en el registro de datos entrantes 52 de la GPU 40 (128). El módulo de paso de mensajes 50 puede determinar si se ha activado el bit de estado (130). Si se ha activado el bit de estado en el registro de datos entrantes 52, el módulo de paso de mensajes 50 puede determinar que el mensaje entrante está disponible (132). Por otro lado, si el bit de estado en el registro de datos entrantes 52 no está activado, el módulo de paso de mensajes 50 puede determinar que el mensaje entrante no está disponible (134).

La FIG. 10 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de proceso 120 en la FIG. 8 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 10. El módulo de paso de mensajes 50 puede recuperar los datos de mensajes entrantes a partir del registro de datos entrantes 52 en la GPU 40 (136). El módulo de paso de mensajes 50 puede generar datos de mensajes de retorno para la tarea 48 en base a los datos de mensajes recuperados desde el registro de datos entrantes 52 (138). En algunos ejemplos, los datos de mensajes devueltos pueden ser idénticos a los datos de mensajes contenidos en el registro de datos entrantes 52. En ejemplos adicionales, el módulo de paso de mensajes 50 puede eliminar uno o más elementos de información de cabecera y/o información de encaminamiento, a partir de los datos de mensajes contenidos en el registro de datos entrantes 52, para generar los datos de mensajes de retorno. En ejemplos adicionales, el módulo de paso de mensajes 50 puede realizar una o más operaciones de decodificación o de transformación de los datos de mensajes contenidos en el registro de datos entrantes 52 para generar los datos de mensajes de retorno. El módulo de paso de mensajes 50 proporciona los datos de mensajes a la tarea 48 (140).

La FIG. 11 muestra una técnica ejemplar para la ejecución de una instrucción de envío emitida por un proceso que se está ejecutando en un dispositivo informático, por ejemplo, la GPU 14, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 11. El módulo de paso de mensajes 50 recibe una instrucción de envío desde la tarea 28 (142). El módulo de paso de mensajes 50 genera un mensaje saliente en base a datos de mensajes contenidos en la instrucción de envío (144). En algunos ejemplos, el mensaje saliente puede ser idéntico a los datos de mensajes contenidos en la instrucción de envío. En ejemplos adicionales, el módulo de paso de mensajes 50 puede anexar uno o más elementos de información de cabecera y/o información de encaminamiento a los datos de mensajes contenidos en la instrucción de envío para generar el mensaje saliente. En otros ejemplos, el módulo de paso de mensajes 50 puede realizar una o más operaciones de codificación o de transformación en los datos de mensajes contenidos en la instrucción de envío para generar el mensaje saliente. El módulo de paso de mensajes 50 puede enviar el mensaje saliente al dispositivo anfitrión 12 (146).

El módulo de paso de mensajes 50 puede determinar si la instrucción de envío es una instrucción bloqueante o una instrucción no bloqueante (148). En algunos ejemplos, el módulo de paso de mensajes 50 puede tomar la determinación de si la instrucción de envío es una instrucción bloqueante o no bloqueante, en base a un parámetro de entrada especificado en la instrucción de envío. En otros ejemplos, pueden utilizarse dos tipos diferentes de instrucciones de envío, y el módulo de paso de mensajes 50 pueden tomar la determinación de si la instrucción de envío es una instrucción bloqueante o no bloqueante, en base al tipo de instrucción, por ejemplo, el código de operación (codeop) de la instrucción. Si el módulo de paso de mensajes 50 determina que la instrucción de envío es una instrucción no bloqueante, el módulo de paso de mensajes 50 puede devolver un asidero al proceso de llamada, por ejemplo, la tarea 28 (150). El asidero puede permitir que el proceso de llamada consulte el asidero para determinar si el mensaje ha sido enviado con éxito en un momento posterior. Si la consulta posterior indica que la operación de envío falló, el proceso de llamada puede tener que emitir una instrucción de envío posterior para volver a intentar la operación de envío.

Si el módulo de paso de mensajes 50 determina que la instrucción de envío es una instrucción bloqueante, el módulo de paso de mensajes 50 puede determinar si el mensaje saliente ha sido recibido con éxito por el dispositivo anfitrión 12 (152). Por ejemplo, el módulo de paso de mensajes 50 puede sondear un registro de estado contenido dentro del dispositivo anfitrión 12, que indica si el mensaje fue aceptado. Si el módulo de paso de mensajes 50 determina que el mensaje saliente se ha recibido con éxito, el módulo de paso de mensajes 50 puede devolver un valor al proceso de llamada que indica que el mensaje contenido en la instrucción de envío ha sido enviado con éxito (154). De lo contrario, si el módulo de paso de mensajes 50 determina que el mensaje saliente no se ha recibido con éxito, el módulo de paso de mensajes 50 puede proceder a procesar el bloque 146 y re-enviar el mensaje saliente al dispositivo anfitrión 12. La instrucción bloqueante se puede completar, en algunos ejemplos, cuando el módulo de

paso de mensajes 50 determina que el mensaje fue recibido con éxito o se ha alcanzado un número de umbral de intentos de entrega sin éxito.

La FIG. 12 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de proceso 146 en la FIG. 11 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 12. El módulo de paso de mensajes 50 puede colocar o almacenar el mensaje saliente en el registro de datos salientes 54 (156). El registro de mensajes salientes 62 puede actualizar los datos en el registro de mensajes salientes 62 para corresponderse con el registro de datos salientes 54, en respuesta a los nuevos datos que se colocan en el registro de datos salientes 54 (158). El módulo de paso de mensajes 50 puede generar y enviar una señal de interrupción al dispositivo anfitrión 12 indicando que hay un mensaje disponible proveniente de la tarea 28 de la GPU 40 (160).

La FIG. 13 es otra técnica ejemplar que puede utilizarse para implementar el bloque de proceso 146 de la FIG. 11 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 13. El módulo de paso de mensajes 50 puede colocar o almacenar el mensaje saliente en el registro de datos salientes 54 (162). El registro de mensajes salientes 62 puede actualizar los datos en el registro de mensajes salientes 62 para corresponderse con el registro de datos salientes 54, en respuesta a los nuevos datos que se colocan en el registro de datos salientes 54 (164). El módulo de paso de mensajes 50 puede activar un bit de estado en el registro de estado de interrupción 64 para indicar que hay un mensaje disponible de la tarea 28 de la GPU 40. Puede activarse el bit de estado para permitir que el dispositivo anfitrión 12 sondee la GPU 40 para determinar si un mensaje está disponible (166).

La FIG. 14 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de registro de rutina de devolución de llamada, emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 14. La interfaz de paso de mensajes del anfitrión 26 recibe una instrucción de registro de rutina de devolución de llamada desde el proceso anfitrión 20 (168). La interfaz de paso de mensajes del anfitrión 26 asocia la rutina de devolución de llamada, especificada en la instrucción de registro de rutina de devolución de llamada, con una señal de interrupción desde un dispositivo especificado en la instrucción, por ejemplo, la GPU 14 (170). En algunos ejemplos, la señal de interrupción puede ser indicativa de que una tarea que se está ejecutando en el dispositivo especificado, por ejemplo, la tarea 28 que se está ejecutando en la GPU 14, ha enviado un mensaje. La señal de interrupción, en algunos ejemplos, puede entregarse mediante una línea de señal de interrupción dedicada que está acoplada entre el dispositivo anfitrión 12 y la GPU 14. En otros ejemplos, la señal de interrupción puede ser indicativa de otros sucesos, además de la tarea 28 que envía un mensaje. En dichos ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede realizar un procesamiento adicional después de recibir la señal de interrupción, para determinar cuál de los múltiples sucesos está representado por la señal.

La interfaz de paso de mensajes del anfitrión 26 determina si la rutina de devolución de llamada se asoció con éxito con la señal de interrupción (172). Si la rutina de devolución de llamada se asoció con éxito con la señal de interrupción, la interfaz de paso de mensajes del anfitrión 26 puede devolver un valor al proceso de llamada, que indica que la operación de registro de rutina de devolución de llamada se completó con éxito (174). De lo contrario, si la rutina de devolución de llamada no se asoció con éxito con la señal de interrupción, por ejemplo, se produjo un error, la interfaz de paso de mensajes del anfitrión 26 puede devolver un valor al proceso de llamada que indica que la operación de registro de rutina de devolución de llamada ha fallado (176).

La FIG. 15 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de una interrupción recibida desde un dispositivo informático de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 15. La interfaz de paso de mensajes del anfitrión 26 recibe una señal de interrupción desde un dispositivo informático, por ejemplo, la GPU 14. (178). La interfaz de paso de mensajes del anfitrión 26 determina si se envió la señal de interrupción en respuesta a un suceso de recepción de mensajes (180). En otras palabras, la interfaz de paso de mensajes del anfitrión 26 puede determinar si la señal de interrupción indica que una tarea que se está ejecutando en el dispositivo, por ejemplo, la tarea 28 que se está ejecutando en la GPU 14, ha enviado un mensaje.

En algunos ejemplos, la señal de interrupción puede ser una señal de interrupción dedicada que señaliza sucesos de recepción de mensajes y ningún otro suceso. En tales ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede determinar que se envió la señal de interrupción en respuesta a un suceso de recepción de mensaje, en virtud de la recepción de la propia señal de interrupción, y que no es necesario realizar ninguna otra operación. En ejemplos en los que la señal de interrupción señaliza una pluralidad de posibles sucesos, la interfaz de paso de mensajes del anfitrión 26 tal vez necesite consultar el dispositivo informático para determinar qué suceso se ha señalizado.

Si la interfaz de paso de mensajes del anfitrión 26 determina que no se envió la señal de interrupción en respuesta a un suceso de recepción de mensaje, la interfaz de paso de mensajes del anfitrión 26 puede comprobar si hay otros

tipos de sucesos (182). De lo contrario, si la interfaz de paso de mensajes del anfitrión 26 determina que se envió la señal de interrupción en respuesta a un suceso de recepción de mensajes, la interfaz de paso de mensajes del anfitrión 26 puede ejecutar la rutina de devolución de llamada asociada con el dispositivo desde el que se recibió el mensaje (184).

La FIG. 16 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de decisión 180 en la FIG. 15 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 16. La interfaz de paso de mensajes del anfitrión 26 puede leer el registro de estado de interrupción 64 en la GPU 40 (186). La interfaz de paso de mensajes del anfitrión 26 puede determinar si un bit de estado en el registro de estado de interrupción 64 indica que un nuevo mensaje está disponible para el dispositivo anfitrión (188). Por ejemplo, el módulo de paso de mensajes 50 puede activar el bit de estado en el registro de estado de interrupción 64 cuando un mensaje está disponible, y la interfaz de paso de mensajes del anfitrión 26 puede sondear el registro de estado de interrupción 64 para determinar si se activa el bit de estado con el fin de determinar si un nuevo mensaje está disponible para el dispositivo anfitrión. Si el bit de estado indica que un nuevo mensaje está disponible para el dispositivo anfitrión, la interfaz de paso de mensajes del anfitrión 26 puede determinar que se ha enviado la señal de interrupción en respuesta a un suceso de recepción de mensaje (190). Por otro lado, si el bit de estado indica que un nuevo mensaje no está disponible para el dispositivo anfitrión, la interfaz de paso de mensajes del anfitrión 26 puede determinar que no se ha enviado la señal de interrupción en respuesta a un suceso de recepción de mensaje (192).

La FIG. 17 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de proceso 184 en la FIG. 15 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 17. La interfaz de paso de mensajes del anfitrión 26 puede recuperar el mensaje desde el registro de mensajes salientes 62 en la GPU 40 (194). La interfaz de paso de mensajes del anfitrión 26 puede borrar un bit de confirmación en el registro de confirmación de interrupción 66 (196). El borrado del bit de confirmación puede facilitar el control del flujo de la GPU 40. Por ejemplo, la GPU 40 puede activar el bit de confirmación en el registro de confirmación de interrupción 66 cuando se escribe un mensaje saliente en el registro de mensajes salientes 62, y esperar hasta que el bit de confirmación se haya borrado antes de escribir datos adicionales en el registro de mensajes salientes 62.

La FIG. 18 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de lectura emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 para poner en práctica la técnica ejemplar que se muestra en la FIG. 18. La interfaz de paso de mensajes del anfitrión 26 recibe una instrucción de lectura que especifica un dispositivo en particular, del cual leer los datos (198). La interfaz de paso de mensajes principal 26 sondea el dispositivo especificado en la instrucción de lectura (200). La interfaz de paso de mensajes del anfitrión 26 determina si un mensaje está disponible en el dispositivo especificado en la instrucción de recepción, en base a los datos de sondeo recibidos desde la operación de sondeo (202). Si la interfaz de paso de mensajes del anfitrión 26 determina que un mensaje está disponible en el dispositivo especificado en la instrucción de recepción, la interfaz de paso de mensajes del anfitrión 26 puede recuperar el mensaje desde el dispositivo especificado en la instrucción de lectura (204). En algunos ejemplos, la interfaz de paso de mensajes del anfitrión 26 puede recuperar el mensaje desde un registro en el dispositivo que sea accesible para el dispositivo anfitrión 12, por ejemplo, el registro de mensajes salientes en la GPU 62 40. La interfaz de paso de mensajes del anfitrión 26 puede devolver los datos de mensajes al proceso llamante, por ejemplo, el proceso anfitrión 20 (206). Si la interfaz de paso de mensajes del anfitrión 26 determina que un mensaje no está disponible en el dispositivo especificado en la instrucción de recepción, la interfaz de paso de mensajes del anfitrión 26 puede devolver un valor que indica que la instrucción de lectura falló (208). El proceso llamante tal vez necesite emitir de nuevo la instrucción de lectura para volver a intentar la operación de lectura.

La FIG. 19 es un diagrama de flujo que ilustra una técnica ejemplar que puede utilizarse para implementar el bloque de decisión 202 en la FIG. 18 de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 10 de la FIG. 1 y/o la GPU 40 de la FIG. 2 para poner en práctica la técnica ejemplar que se muestra en la FIG. 19. La interfaz de paso de mensajes del anfitrión 26 puede leer el registro de estado de interrupción 64 en la GPU 40 (210). La interfaz de paso de mensajes del anfitrión 26 puede determinar si un bit de estado en el registro de estado de interrupción 64 indica que un nuevo mensaje está disponible para el dispositivo anfitrión (212). Por ejemplo, el módulo de paso de mensajes 50 puede activar el bit de estado en el registro de estado de interrupción 64 cuando un mensaje está disponible, y la interfaz de paso de mensajes del anfitrión 26 puede sondear el registro de estado de interrupción 64 para determinar si está activado el bit de estado, con el fin de determinar si un nuevo mensaje está disponible para el dispositivo anfitrión. Si está activado el bit de estado, la interfaz de paso de mensajes del anfitrión 26 puede determinar que un mensaje está disponible (214). Por otro lado, si el bit de estado no está activado, la interfaz de paso de mensajes del anfitrión 26 puede determinar que el mensaje no está disponible (216).

Aunque las técnicas de paso de mensajes implementadas por la interfaz de paso de mensajes del anfitrión 26 y la interfaz de paso de mensajes de dispositivo 30 se han descrito anteriormente como proporcionando señalización

fuera de banda entre el dispositivo anfitrión 12 y la GPU 14, en otros sistemas ejemplares pueden utilizarse otras técnicas para proporcionar señalización fuera de banda. Por ejemplo, se puede definir una cola especial de alta prioridad, en algunos ejemplos, que puede utilizarse para el envío de mensajes fuera de banda.

La FIG. 20 es un diagrama de bloques que ilustra un sistema informático ejemplar 310 que puede facilitar el uso de objetos de memoria inmediata de acuerdo con esta divulgación. El sistema informático 310 está configurado para procesar una o más aplicaciones de software en múltiples dispositivos de procesamiento. En algunos ejemplos, las una o más aplicaciones pueden incluir un proceso anfitrión, y el sistema informático 310 puede estar configurado para ejecutar el proceso anfitrión y para distribuir la ejecución de una o más tareas iniciadas por el proceso anfitrión en otros dispositivos informáticos dentro del sistema informático 310. En otros ejemplos, el proceso anfitrión y/o las tareas ejecutadas por el sistema informático 310 pueden ser programados de acuerdo con un modelo de programación en paralelo. Por ejemplo, las aplicaciones pueden incluir instrucciones que están diseñadas para aprovechar el paralelismo a nivel de tareas y/o el paralelismo a nivel de datos de los sistemas de hardware subyacentes.

El sistema informático 310 puede comprender un ordenador personal, un ordenador de sobremesa, un ordenador portátil, una estación de trabajo de ordenador, una consola o una plataforma de videojuegos, un teléfono móvil, tal como, por ejemplo, un teléfono celular o por satélite, un teléfono móvil, un teléfono fijo, un teléfono de Internet, un dispositivo de mano como un dispositivo de videojuegos portátil o un asistente digital personal (PDA), un reproductor de medios digitales, tal como un reproductor personal de música, un reproductor de vídeo, un dispositivo de visualización, o un televisor, un decodificador de televisión, un servidor, un dispositivo de red intermedia, un ordenador central o cualquier otro tipo de dispositivo que procesa información.

El sistema informático 310 incluye un dispositivo anfitrión 312, una GPU 314, una memoria 316 y una red de interconexión 318. El dispositivo anfitrión 312 está configurado para proporcionar una plataforma para la ejecución de un proceso anfitrión y un módulo de tiempo de ejecución para una API de la plataforma informática de múltiples procesadores. Habitualmente, el dispositivo anfitrión 312 es una CPU de propósito general, aunque el dispositivo anfitrión 12 puede ser cualquier tipo de dispositivo capaz de ejecutar programas. El dispositivo anfitrión 12 está acoplado comunicativamente a la GPU 314 y a la memoria 316 a través de la red de interconexión 318. El dispositivo anfitrión 312 incluye un proceso anfitrión 320, un módulo de tiempo de ejecución 322, una memoria caché del anfitrión 324 y un módulo de control de memoria caché del anfitrión. El proceso anfitrión 320 y el módulo de tiempo de ejecución 322 se pueden ejecutar en cualquier combinación de uno o más procesadores programables.

El proceso anfitrión 320 incluye un conjunto de instrucciones que forman un programa de software para su ejecución en la plataforma del sistema informático 310. El programa de software puede estar diseñado para realizar una o más tareas específicas para un usuario final. Tales tareas pueden, en algunos ejemplos, implicar algoritmos intensivos en cálculo, que pueden explotar los múltiples dispositivos de procesamiento y arquitecturas paralelas proporcionados por el sistema informático 310.

El módulo de tiempo de ejecución 322 puede ser un módulo de software que implementa una o más interfaces configuradas para dar servicio a una o más de las instrucciones contenidas en el proceso anfitrión 320. Las interfaces implementadas por el módulo de tiempo de ejecución 322 incluyen la interfaz de memoria intermedia 328. En algunos ejemplos, el módulo de tiempo de ejecución 322 puede implementar una o más entre la interfaz de cola de comandos 24 mostrada en la FIG. 1 y/o la interfaz de paso de mensajes del anfitrión 26 mostrada en la FIG. 1, además de la interfaz de memoria intermedia 328. En ejemplos adicionales, el módulo de tiempo de ejecución 322 puede implementar una o más interfaces contenidas dentro de una API estándar dentro del sistema de múltiples procesadores, además de las interfaces descritas en esta divulgación. En algunos ejemplos, la API estándar puede ser una API de plataforma informática heterogénea, una API de múltiples plataformas, una API de múltiples proveedores, una API de programación en paralelo, una API de programación en paralelo a nivel de tareas y/o una API de programación en paralelo a nivel de datos. En otros ejemplos, la API estándar puede ser la API de OpenCL. En tales ejemplos, el módulo de tiempo de ejecución 322 puede estar diseñado para estar en conformidad con una o más de las especificaciones de OpenCL. En ejemplos adicionales, el módulo de tiempo de ejecución 322 se puede implementar como parte de, o ser, un programa controlador, por ejemplo, un controlador de GPU.

La interfaz de memoria intermedia 328 está configurada para recibir una o más instrucciones de creación de objetos de memoria desde el proceso anfitrión 20 y para ejecutar las funciones especificadas por las instrucciones recibidas. En algunos ejemplos, la interfaz de memoria intermedia 328 puede implementarse como una extensión para una API estándar existente, tal como, por ejemplo, la API de OpenCL. En ejemplos adicionales, la interfaz de cola de comandos 24 puede estar integrada en una API estándar existente, tal como, por ejemplo, la API de OpenCL.

La memoria caché del anfitrión 324 está configurada para almacenar datos para su uso por parte de los procesos que se están ejecutando dentro del dispositivo anfitrión 312. En algunos ejemplos, el espacio de memoria asociado a los datos almacenados en la memoria caché del anfitrión 324 puede superponerse con una parte del espacio de memoria en la memoria 316. La memoria caché del anfitrión 324 puede ser cualquier tipo de memoria caché conocida en la técnica. Por ejemplo, la memoria caché del anfitrión 324 puede incluir cualquier combinación de niveles de memoria caché (por ejemplo, LI, L2, etc.) y/o esquemas de correlación (por ejemplo, correlación directa,

totalmente asociativa, asociativa por conjuntos, etc.). El módulo de control de la memoria caché del anfitrión 326 está configurado para controlar el funcionamiento de la memoria caché del anfitrión 324.

La GPU 314 está configurada para ejecutar una o más tareas en respuesta a las instrucciones recibidas desde el dispositivo anfitrión 312. La GPU 314 puede ser cualquier tipo de GPU que incluya uno o más procesadores o elementos de procesamiento programables. Por ejemplo, la GPU 314 puede incluir una o más unidades de sombreado programables que están configuradas para ejecutar una pluralidad de instancias de ejecución para una tarea en paralelo. Las unidades de sombreado programables pueden incluir una unidad de sombreado de vértices, una unidad de sombreado de fragmentos, una unidad de sombreado de geometría y/o una unidad de sombreado unificada. La GPU 314 está acoplada comunicativamente al dispositivo anfitrión 312 y a la memoria 316 a través de la red de interconexión 318. La GPU 314 incluye una tarea 330, una memoria caché de GPU 332 y un módulo de control de memoria caché de GPU 334. La tarea 330 puede ejecutarse en cualquier combinación de uno o más elementos de procesamiento programables.

La tarea 330 comprende un conjunto de instrucciones que forman una tarea para la ejecución en un dispositivo informático en un sistema informático 310. En algunos ejemplos, el conjunto de instrucciones para la tarea 330 puede definirse en el proceso anfitrión 320 y, en algunos casos, compilarse mediante las instrucciones incluidas en el proceso anfitrión 320. En ejemplos adicionales, la tarea 330 puede ser un programa de núcleo que tiene múltiples instancias de ejecución que se están ejecutando en la GPU 314 en paralelo. En dichos ejemplos, el proceso anfitrión 320 puede definir un espacio de índices para el núcleo que correlaciona instancias de ejecución del núcleo con los respectivos elementos de procesamiento para la ejecución de las instancias de ejecución de núcleo, y la GPU 314 puede ejecutar las múltiples instancias de ejecución de núcleo para la tarea 330 de acuerdo con el espacio de índices definido para el núcleo.

La memoria caché de GPU 332 está configurada para almacenar datos para su uso por parte de las tareas que se ejecutan dentro de la GPU 314. En algunos ejemplos, el espacio de memoria asociado a los datos almacenados en la memoria caché de GPU 332 puede superponerse con una parte del espacio de memoria en la memoria 316. La memoria caché de GPU 332 puede ser cualquier tipo de memoria caché conocida en la técnica. Por ejemplo, la memoria caché de GPU 332 puede incluir cualquier combinación de niveles de memoria caché (por ejemplo, LI, L2, etc.) y/o esquemas de correlación (por ejemplo, correlación directa, totalmente asociativa, asociativa por conjuntos, etc.). El módulo de control de memoria caché de GPU 334 está configurado para controlar el funcionamiento de la memoria caché de GPU 332.

La memoria 316 está configurada para almacenar datos para su uso por uno entre el dispositivo anfitrión 312 y la GPU 314, o ambos. La memoria 316 puede incluir cualquier combinación de una o más memorias o dispositivos de almacenamiento volátiles o no volátiles, tales como, por ejemplo, memoria de acceso aleatorio (RAM), RAM estática (SRAM), RAM dinámica (DRAM), memoria de solo lectura (ROM), ROM borrable y programable (EPROM), ROM borrable y programable eléctricamente (EEPROM), memoria flash, un medio magnético de almacenamiento de datos o un medio de almacenamiento óptico. La memoria 316 está acoplada comunicativamente al dispositivo anfitrión 312 y a la GPU 314 a través de la red de interconexión 318. La memoria 316 incluye el espacio de memoria compartida 336. El espacio de memoria compartida 336 puede ser un espacio de memoria que es accesible tanto por el dispositivo anfitrión 312 como por la GPU 314.

La red de interconexión 318 está configurada para facilitar la comunicación entre el dispositivo anfitrión 312, la GPU 314 y la memoria 316. La red de interconexión 318 puede ser cualquier tipo de red de interconexión conocida en la técnica. En el sistema informático ejemplar 310 de la FIG. 20, la red de interconexión 318 es un bus. El bus puede incluir uno o más de cualquiera entre una diversidad de estructuras de bus, tales como, por ejemplo, un bus de tercera generación (por ejemplo, un bus HyperTransport o un bus InfiniBand), un bus de segunda generación (por ejemplo, un bus de Puerto de Gráficos Avanzado, un bus Expreso de Interconexión de Componentes Periféricos (PCIe) o un bus de Interfaz Extensible Avanzada (AXI)) o cualquier otro tipo de bus. La red de interconexión 318 está acoplada al dispositivo anfitrión 312, la GPU 314 y la memoria 316.

Las estructuras y funcionalidades de los componentes en el sistema informático 310 se describirán ahora con más detalle. Como se ha expuesto anteriormente, el proceso anfitrión 320 incluye un conjunto de instrucciones. El conjunto de instrucciones puede incluir, por ejemplo, una o más instrucciones de creación de objetos de memoria. En ejemplos adicionales, el conjunto de instrucciones puede incluir instrucciones que especifican las tareas o núcleos que se tienen que ejecutar en la GPU 314, instrucciones que crean colas de comandos y que asocian las colas de comandos con determinados dispositivos, instrucciones que compilan y vinculan programas, instrucciones que configuran parámetros de núcleo , instrucciones que definen espacios de índices, instrucciones que definen un contexto de dispositivo, instrucciones de puesta en cola, instrucciones de paso de mensajes y otras instrucciones que prestan soporte a la funcionalidad proporcionada por el proceso anfitrión 320.

De acuerdo con esta divulgación, el proceso anfitrión 320 puede interactuar con la interfaz de memoria intermedia 328 mediante la emisión de una o más instrucciones de creación de objetos de memoria para la interfaz de memoria intermedia 328, instruyendo a la interfaz de memoria intermedia 328 para crear un objeto de memoria basado en la información contenida en la instrucción que especifica si una modalidad inmediata está habilitada para el objeto de

memoria. Como se usa en el presente documento, un objeto de memoria puede referirse a un objeto de software que representa una región del espacio de memoria accesible por la GPU 314. En algunos ejemplos, la región del espacio de memoria también puede ser accesible por el dispositivo anfitrión 312. Un objeto de memoria puede incluir los datos contenidos en el espacio de memoria asociado con el objeto de memoria. El objeto de memoria puede incluir además una o más características asociadas con el espacio de memoria. En algunos ejemplos, un objeto de memoria puede incluir un asidero para una región de recuento de referencia de la memoria global, por ejemplo, la memoria 316.

Los objetos de memoria pueden incluir objetos de memoria intermedia y objetos de imagen. Un objeto de memoria intermedia puede ser un objeto de memoria que almacena una colección unidimensional de octetos. La colección unidimensional de octetos pueden ser los datos asociados con el objeto de memoria. Un objeto de memoria intermedia también puede incluir información tal como, por ejemplo, el tamaño del espacio de memoria asociado con el objeto de memoria intermedia, en octetos, la información de uso para el objeto de memoria intermedia y la región del espacio de memoria asignado para el objeto de memoria intermedia. Un objeto de imagen almacena una matriz bidimensional o tridimensional de datos, tales como, por ejemplo, una textura, una memoria intermedia de trama o una imagen. Un objeto de imagen también puede incluir información tal como, por ejemplo, las dimensiones de la imagen, una descripción de cada elemento en la imagen, la información de uso para el objeto de la imagen y la región del espacio de memoria asignado para el objeto de la imagen.

De acuerdo con algunos aspectos de esta divulgación, las instrucciones de creación de objetos de memoria pueden incluir un parámetro de entrada que especifica si una modalidad inmediata debería ser habilitada para el objeto de memoria a crear. Como se ha expuesto en más detalle en el presente documento, cuando se habilita la modalidad inmediata, el objeto de memoria puede implementarse como una memoria compartida no almacenable en memoria caché y/o como una memoria compartida coherente con memoria caché. Cuando la modalidad inmediata está inhabilitada, el objeto de memoria puede no implementarse necesariamente como una memoria compartida no almacenable en memoria caché, o como una memoria compartida coherente con memoria caché.

En algunos ejemplos, el objeto de memoria puede incluir un atributo de modalidad inmediata que sea indicativo de si el objeto de memoria es un objeto de memoria en modalidad inmediata. En tales ejemplos, la interfaz de memoria intermedia 328 puede estar configurada para fijar el atributo de modalidad inmediata, para el objeto de memoria que tiene que crearse, en un valor indicativo de si la modalidad inmediata está habilitada para el objeto de memoria, en base a la información que especifica si la modalidad inmediata debería ser habilitada para el objeto de memoria. El atributo de modalidad inmediata del objeto de memoria puede ser utilizado por el sistema informático 310 para determinar si implementa o no el objeto de memoria como una memoria compartida no almacenable en memoria caché y/o como una memoria compartida coherente con memoria caché.

Las instrucciones de creación de objetos de memoria pueden, en algunos ejemplos, incluir una instrucción de creación de objetos de memoria intermedia que instruye a la interfaz de memoria intermedia 328 para crear un objeto de memoria intermedia en base a la información contenida en la instrucción que especifica si una modalidad inmediata está habilitada para el objeto de memoria intermedia. Las instrucciones de creación de objetos de memoria pueden, en ejemplos adicionales, incluir una instrucción de creación de objetos de imagen que instruye a la interfaz de memoria intermedia 328 para crear un objeto de imagen basándose en la información contenida en la instrucción que especifica si una modalidad inmediata está habilitada para el objeto de la imagen.

En algunos ejemplos, la interfaz para la instrucción de creación de objetos de memoria intermedia puede adoptar la forma siguiente:

```
cl_mem clCrearMemoriaIntermedia (
        cl_context contexto,
        cl_mem_flags indicadores,
        size_t tamaño,
        void *puntero_anfitrión,
        cl_int *dev_coderror)
```

donde clCrearMemoriaIntermedia es el identificador de la instrucción, cl_context contexto es un contexto válido, por ejemplo, un contexto de OpenCL, que se utiliza para crear el objeto de memoria intermedia, cl_mem_flags indicadores es un campo de bits que se utiliza para especificar la información de asignación y uso para el objeto de memoria intermedia, size_t tamaño es un parámetro que especifica el tamaño en octetos del objeto de memoria intermedia a ser asignado, void *puntero_anfitrión es un puntero para los datos de la memoria intermedia que pueden ser ya asignados por la aplicación, y cl_int *dev_coderror devuelve uno o más códigos de error. La instrucción puede devolver el objeto de memoria intermedia creado como un objeto de memoria cl_mem. En este ejemplo, el parámetro de entrada que especifica si una modalidad inmediata debería ser habilitada para el objeto de la imagen puede ser, por ejemplo, un indicador CL_INMEDIATO que se especifica en el campo cl_mem_flags indicadores.

En otros ejemplos, la interfaz para la instrucción de creación de objetos de imagen puede adoptar la forma siguiente:

```
cl_mem clCrearImagen2D (
        cl_context contexto,
        cl_mem_flags indicadores,
        const cl_image_format *formato_imagen,
        size_t ancho_imagen,
        size_t altura_imagen,
        size_t paso_línea_imagen,
        void *puntero_anfitrión,
        cl_int *dev_coderror)
```

donde clCrearImagen2D es el identificador de la instrucción, cl_context contexto es un contexto válido, por ejemplo, un contexto de OpenCL, que se utiliza para crear el objeto de memoria intermedia, cl_mem_flags indicadores es un campo de bits que se utiliza para especificar la información de la asignación y la utilización de la imagen del objeto, const cl_image_format *formato_imagen apunta a una estructura que describe las propiedades de formato de la imagen que debe asignarse, size_t ancho_imagen es el ancho de la imagen en píxeles, size_t altura_imagen es la altura de la imagen en píxeles, size_t paso_línea_imagen es el paso de la línea de escaneo en octetos, void *puntero_anfitrión es un puntero a los datos de imagen que ya pueden ser asignados por la aplicación y cl_int *dev_coderror devuelve uno o más códigos de error. La instrucción puede devolver el objeto de imagen creado como un objeto de memoria cl_mem. En este ejemplo, el parámetro de entrada que especifica si una modalidad inmediata debería ser habilitada para el objeto de la imagen puede ser, por ejemplo, un indicador CL_INMEDIATO que se especifica en el campo cl_mem_flags indicadores.

En algunos ejemplos, las interfaces de creación de objetos de memoria pueden estar configuradas para permitir solo un atributo de SOLO_ESCRITURA o un atributo de SOLO_LECTURA en términos de atributos de lectura / escritura. En otras palabras, en tales ejemplos, la interfaz de memoria intermedia 328 puede no admitir un atributo de LECTURA_ESCRITURA. Las imágenes de CL no inmediatas pueden tener ya una característica tal, proporcionada por la especificación de OpenCL. No permitir el atributo LECTURA_ESCRITURA puede reducir la complejidad en el mantenimiento de la coherencia de la memoria caché.

De acuerdo con esta divulgación, la interfaz de memoria intermedia 328 está configurada para recibir una instrucción que especifica si una modalidad inmediata debe ser habilitada para el espacio de memoria compartida 336, que es accesible tanto por el dispositivo anfitrión 312 como por la GPU 314, y para habilitar selectivamente la modalidad inmediata para el espacio de memoria compartida 336, en base a la instrucción recibida que especifica si la modalidad inmediata debería ser habilitada para el espacio de memoria compartida 336. Por ejemplo, la interfaz de memoria intermedia 328 puede habilitar la modalidad inmediata para el espacio de memoria compartida 336 si la instrucción especifica que la modalidad inmediata debería ser habilitada para el espacio de memoria compartida 336, e inhabilitar la modalidad inmediata para el espacio de memoria compartida 336 si la instrucción especifica que la modalidad inmediata debería ser inhabilitada para el espacio de memoria compartida 336. La instrucción puede ser, por ejemplo, una entre una instrucción de creación de objetos de memoria, una instrucción de creaciones de objetos de memoria intermedia o una instrucción de creación de objetos de imagen. El espacio de memoria compartida 336 puede corresponder, por ejemplo, a un objeto de memoria, un objeto de memoria intermedia o un objeto de imagen.

En algunos ejemplos, cuando la interfaz de memoria intermedia 328 habilita la modalidad inmediata para el espacio de memoria compartida 336, la interfaz de memoria intermedia 328 puede hacer que los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 se inhabiliten. De manera similar, cuando la interfaz de memoria intermedia 328 inhabilita la modalidad inmediata para el espacio de memoria compartida 336, la interfaz de memoria intermedia 328 puede hacer que los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 sean habilitados para el espacio de memoria compartida 336. Los servicios de almacenamiento en memoria caché pueden ser realizados por una entre la memoria caché del anfitrión 324 y la memoria caché de la GPU 332, o ambas. Los servicios de almacenamiento en memoria caché, como se usan en el presente documento, pueden referirse a los servicios que habitualmente son llevados a cabo por una memoria caché, como se conoce en la técnica.

En ejemplos adicionales, la interfaz de memoria intermedia 328 puede habilitar e inhabilitar la modalidad inmediata para el espacio de la memoria compartida 336, mediante la fijación de un atributo de modalidad inmediata, asociado con el espacio de memoria compartida 336, en un valor indicativo de si la modalidad inmediata está habilitada para el espacio de memoria compartida. Por ejemplo, la interfaz de memoria intermedia 328 puede habilitar la modalidad inmediata para el espacio de memoria compartida 336 mediante la fijación de un atributo de modalidad inmediata, asociado con el espacio de memoria compartida 336, en un valor que indica que la modalidad inmediata está habilitada para el espacio de memoria compartida 336, por ejemplo, atributo de modalidad inmediata = verdadero. De manera similar, la interfaz de memoria intermedia 328 puede inhabilitar la modalidad inmediata para el espacio de memoria compartida 336 fijando el atributo de modalidad inmediata, asociado con el espacio de la memoria compartida 336, en un valor que indica que la modalidad inmediata está inhabilitada para el espacio de memoria compartida 336, por ejemplo, atributo de modalidad inmediata = falso. El atributo de modalidad inmediata puede, en

algunos casos, ser una variable global, por ejemplo, una variable booleana, accesible por la tarea 330 que se está ejecutando en la GPU 314. En algunos ejemplos, el atributo de modalidad inmediata puede ser almacenado dentro del espacio de memoria compartida 336. En otros ejemplos, el atributo de modalidad inmediata se puede almacenar en una ubicación accesible por la tarea 330 que se está ejecutando en la GPU 314, en lugar del espacio de memoria compartida 336. En los casos en que el espacio de memoria compartida 336 es parte de un objeto de memoria, el atributo de modalidad inmediata puede ser almacenado en una ubicación del espacio de memoria donde se almacenan los otros atributos del objeto de memoria.

En ejemplos en los que la interfaz de memoria intermedia 328 habilita e inhabilita la modalidad inmediata para el espacio de memoria compartida 336 mediante la activación de un atributo de modalidad inmediata asociado con el espacio de memoria compartida 336, el código fuente de la tarea 330 puede, en algunos casos, ser compilado de tal manera que, antes de la realización de una lectura o escritura en memoria con respecto al espacio de memoria compartida 336, la tarea 330 acceda al atributo de modalidad inmediata asociado con el espacio de memoria compartida 336, y determine si una modalidad inmediata está habilitada para el espacio de memoria compartida 336 en base al atributo de modalidad inmediata para espacio de memoria compartida 336. Si la modalidad inmediata está habilitada para el espacio de memoria compartida 336, a continuación la tarea 330 puede programarse para ejecutar una instrucción de lectura o escritura en modalidad inmediata, para leer datos de, o escribir datos en, el espacio de memoria compartida 336. Por otro lado, si la modalidad inmediata no está habilitada para el espacio de memoria compartida, a continuación la tarea 330 puede programarse para ejecutar una instrucción de lectura o escritura en modalidad de almacenamiento en memoria caché, por ejemplo, una instrucción de lectura o escritura en memoria caché, para leer datos de, o escribir datos en, el espacio de memoria compartida 336.

Las instrucciones de escritura y lectura en modalidad inmediata pueden, por ejemplo, llevar a cabo las operaciones de lectura y escritura, respectivamente, sin necesidad de utilizar los servicios de almacenamiento en memoria caché. Por ejemplo, la instrucción de lectura en modalidad inmediata puede hacer que la memoria caché sea invalidada antes de realizar la operación de lectura y/o puede pasar por alto la memoria caché cuando se realiza la operación de lectura. La instrucción de escritura en modalidad inmediata, por ejemplo, puede hacer que la memoria caché realice una re-escritura inmediata cuando se realiza la operación de escritura y/o puede pasar por alto la memoria caché cuando se realiza la operación de escritura. Las instrucciones de lectura y escritura en memoria caché pueden, por ejemplo, ejecutar operaciones de lectura y escritura, respectivamente, utilizando los servicios de almacenamiento de una de ellas, o ambas, de la caché de GPU 332.

En casos adicionales, el compilador para la tarea 330 puede tener acceso a información, al compilar el código fuente de la tarea 330, indicativa de si la modalidad inmediata está habilitada para el espacio de memoria compartida 336. Por ejemplo, el código fuente de la tarea 330, por ejemplo, el código fuente del núcleo, puede incluir un indicador indicativo de si la modalidad inmediata está habilitada para un objeto de memoria utilizado por la tarea 330 y asociado con el espacio de memoria compartido 336. En algunos ejemplos, el indicador puede tomar la forma de un calificador de atributo de OpenCL, tal como, por ejemplo, un calificador de atributo _cl_immediate. Si la modalidad inmediata está habilitada para el objeto de memoria asociado con el espacio de memoria compartida 336, entonces el compilador puede compilar la tarea 330 de tal manera que el código compilado para la tarea 330 incluya instrucciones de lectura y/o escritura en modalidad inmediata para las operaciones de lectura o escritura que tienen lugar con respecto al espacio de memoria compartida 336. De lo contrario, si la modalidad inmediata no está habilitada para el objeto de memoria asociado con el espacio de memoria compartida 336, a continuación, el compilador puede compilar la tarea 330 de tal manera que el código compilado para la tarea 330 no incluya instrucciones de lectura y/o escritura en modalidad inmediata para las operaciones de lectura o escritura que se realizan con respecto al espacio de memoria compartida 336. Por ejemplo, el compilador puede compilar la tarea 330 de tal manera que el código compilado para la tarea 330 incluya instrucciones de lectura y/o escritura en memoria caché para las operaciones de lectura o escritura que se realizan con respecto al espacio de memoria compartida 336.

En otros ejemplos, la interfaz de memoria intermedia 328 puede habilitar e inhabilitar la modalidad inmediata para el espacio de memoria compartida 336 habilitando e inhabilitando las prestaciones de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 mediante al menos una entre la memoria caché del anfitrión 324 en el dispositivo anfitrión 312 y la memoria caché de GPU 332 en la GPU 314. Por ejemplo, la interfaz de memoria intermedia 328 puede habilitar la modalidad inmediata para el espacio de memoria compartida 336 inhabilitando las prestaciones de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 mediante al menos una entre la memoria caché del anfitrión 324 en el dispositivo anfitrión 312 y la memoria caché de GPU 332 en la GPU 314. De forma similar, la interfaz de memoria intermedia 328 puede inhabilitar la modalidad inmediata para el espacio de memoria compartida 336 al permitir la prestación de servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 por al menos una entre la memoria caché del anfitrión 324 en el dispositivo anfitrión 312 y la memoria caché de GPU 332 en la GPU 314.

En tales ejemplos, la interfaz de memoria intermedia 328 puede habilitar e inhabilitar las prestaciones de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 mediante la configuración de un módulo de control de memoria caché basado en hardware y/o una unidad de gestión de memoria basada en hardware, asociada con la memoria caché que realiza los servicios de almacenamiento en

memoria caché para el espacio de memoria compartida 336. Por ejemplo, para hacer posible la prestación de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 por parte de la memoria caché del anfitrión 324, la interfaz de memoria intermedia 328 puede configurar el módulo de control de la memoria caché del anfitrión 326 de tal manera que los servicios de almacenamiento en memoria caché sean proporcionados por la memoria caché del anfitrión 324 para el espacio de memoria compartida 336. Para inhabilitar la prestación de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 por parte de la memoria caché del anfitrión 324, la interfaz de memoria intermedia 328 puede, por ejemplo, configurar el módulo de control de memoria caché del anfitrión 326 de tal manera que los servicios de almacenamiento en memoria caché no sean proporcionados por la memoria caché del anfitrión 324 para el espacio de memoria compartida 336. De forma similar, para habilitar la prestación de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 mediante la memoria caché de GPU 332, la interfaz de memoria intermedia 328 puede, por ejemplo, configurar el módulo de control de memoria caché de GPU 334 de tal manera que los servicios de almacenamiento en memoria caché sean proporcionados por la memoria caché del anfitrión 324 para el espacio de memoria compartida 336. Para inhabilitar la prestación de los servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 mediante la memoria caché de GPU 332, la interfaz de memoria intermedia 328 puede, por ejemplo, configurar el módulo de control de memoria caché de GPU 334 de tal manera que los servicios de almacenamiento en memoria caché no sean proporcionados por la memoria caché de GPU 332 para el espacio de memoria compartida 336.

En algunos ejemplos, la interfaz de memoria intermedia 328 puede configurar uno entre el módulo de control de memoria caché del anfitrión 326 y el módulo de control de memoria caché de GPU 334, o ambos, mediante la fijación de uno o más indicadores inmediatos basados en hardware, asociados con el espacio de memoria compartida 336, en un valor indicativo de si deberían proporcionarse servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336. Los uno o más indicadores inmediatos basados en hardware pueden ser, en algunos ejemplos, uno o más registros. En ejemplos adicionales, los indicadores inmediatos basados en hardware pueden ser parte de una tabla de indicadores inmediatos donde cada indicador inmediato en la tabla de indicadores inmediatos corresponde a un espacio de direcciones en particular dentro de la memoria 316. En cualquier caso, cuando los uno o más indicadores inmediatos asociados con el espacio de memoria compartida 336 se fijan en valores que indican que deberían proporcionarse servicios de almacenamiento en memoria caché, el módulo de control de memoria caché del anfitrión 326 y/o el módulo de control de memoria caché de GPU 334 pueden proporcionar servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336 utilizando la caché del anfitrión 324 y/o la caché de GPU 332. De forma similar, cuando los uno o más indicadores asociados con el espacio de memoria compartida 336 se fijan en valores que indican que no deberían proporcionarse servicios de almacenamiento en memoria caché, el módulo de control de memoria caché del anfitrión 326 y/o el módulo de control de memoria caché de GPU 334 no pueden proporcionar servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336.

En tales ejemplos, el módulo de control de memoria caché de GPU 334 puede estar configurado para procesar instrucciones de lectura y/o instrucciones de escritura para direcciones de memoria dentro del espacio de direcciones de la memoria 316. Las instrucciones de lectura y escritura pueden ser emitidas al módulo de control de memoria caché de GPU 334, por ejemplo, mediante la tarea 330 que se está ejecutando en la GPU 314. En respuesta a la recepción de una instrucción de lectura o escritura para leer datos de, o escribir datos en, una ubicación de memoria dentro de un espacio de direcciones concreto de la memoria 316, el módulo de control de memoria caché de GPU 334 puede identificar un indicador basado en hardware, asociado con el espacio de direcciones, y determinar si se utilizan los servicios de almacenamiento en memoria caché de la memoria caché de GPU 332 al procesar la instrucción de lectura o escritura en base al valor del indicador basado en hardware. Si el módulo de control de memoria caché de GPU 334 determina utilizar los servicios de almacenamiento en memoria caché de la memoria caché de GPU 332, entonces el módulo de control de memoria caché de GPU 334 puede, por ejemplo, tratar de leer los datos de la memoria caché de GPU 332 si los datos son válidos y/o escribir datos en la memoria caché de GPU 332. Si el módulo de control de memoria caché de GPU 334 determina no utilizar los servicios de almacenamiento en memoria caché de la memoria caché de GPU 332, a continuación, el módulo de control de memoria caché de GPU 334 puede, en algunos ejemplos, pasar por alto la memoria caché de GPU 332 y leer datos de, o escribir datos directamente en, la memoria 316. En ejemplos adicionales, si el módulo de control de memoria caché de GPU 334 determina no utilizar los servicios de almacenamiento en memoria caché de la memoria caché de GPU 332, entonces el módulo de control de memoria caché de GPU 334 puede invalidar una parte de la memoria caché 332 asociada con el espacio de direcciones antes de ejecutar una instrucción de lectura y/o llevar a cabo una técnica de re-escritura en memoria caché o escritura simultánea en memoria caché cuando se ejecuta una instrucción de escritura. El módulo de control de memoria caché del anfitrión 334 puede funcionar de una manera similar con respecto a la memoria caché del anfitrión 324 en respuesta a las instrucciones de lectura y escritura recibidas desde el proceso anfitrión 320 que se está ejecutando en el dispositivo anfitrión 312.

En ejemplos adicionales, la interfaz de memoria intermedia 328 puede habilitar e inhabilitar la modalidad inmediata para el espacio de memoria compartida 336, habilitando e inhabilitando una modalidad de coherencia de memoria caché de la memoria compartida para al menos una entre la memoria caché del anfitrión 324 en el dispositivo anfitrión 312 y la memoria caché de GPU 332 en la GPU 314. Por ejemplo, para habilitar la modalidad inmediata para el espacio de memoria compartida 336, la interfaz de memoria intermedia 328 puede habilitar la modalidad de

coherencia de memoria caché de la memoria compartida para al menos una entre la memoria caché del anfitrión 324 en el dispositivo anfitrión 312 y la memoria caché de GPU 332 en la GPU 314. De forma similar, para inhabilitar la modalidad inmediata para el espacio de memoria compartida 336, la interfaz de memoria intermedia 328 puede inhabilitar la modalidad de coherencia de memoria caché de la memoria compartida para al menos una entre la memoria caché del anfitrión 324 en el dispositivo anfitrión 312 y la memoria caché de GPU 332 en la GPU 314. En tales ejemplos, la interfaz de memoria intermedia 328 puede, en algunos casos, habilitar una modalidad de coherencia de memoria caché de la memoria compartida para la memoria caché del anfitrión 324, mediante la configuración de uno entre el módulo de control de memoria caché del anfitrión 326 y el módulo de control de memoria caché de GPU 334, o ambos, para habilitar la modalidad de coherencia de memoria caché de la memoria compartida e inhabilitar la modalidad de coherencia de memoria caché de la memoria compartida para la memoria caché del anfitrión 324, mediante la configuración de uno entre el módulo de control de memoria caché del anfitrión 326 y el módulo de control de memoria caché de GPU 334, o ambos, para inhabilitar la modalidad de coherencia de memoria caché de la memoria compartida.

Cuando se habilita una modalidad de coherencia de memoria caché de la memoria compartida para la memoria caché del anfitrión 324, el módulo de control de memoria caché principal 326 puede llevar a cabo técnicas de coherencia de memoria caché de la memoria compartida con respecto al espacio de memoria compartida 336, de acuerdo con procedimientos conocidos. Cuando la modalidad de coherencia de memoria caché de la memoria compartida para la memoria caché del anfitrión 324 está inhabilitada, la memoria caché del anfitrión 324 no puede realizar las técnicas de coherencia de memoria caché de la memoria compartida con respecto al espacio de memoria compartida 336. De forma similar, cuando se habilita una modalidad de coherencia de memoria caché de la memoria compartida para la memoria caché de GPU 332, el módulo de control de memoria caché de GPU 334 puede llevar a cabo técnicas de coherencia de memoria caché de la memoria compartida con respecto al espacio de memoria compartida 336, de acuerdo con procedimientos conocidos. Cuando la modalidad de coherencia de memoria caché de la memoria compartida para la memoria caché de GPU 332 está inhabilitada, el módulo de control de memoria caché de GPU 334 no puede realizar las técnicas de coherencia de memoria caché de la memoria compartida con respecto al espacio de memoria compartida 336.

Para facilitar la ilustración, el sistema informático ejemplar 310 ilustrado en la FIG. 20 describe las técnicas de almacenamiento inmediato en memoria intermedia de esta divulgación usando la GPU 314 como un dispositivo informático. Debería reconocerse que las técnicas de esta divulgación pueden ser aplicadas a sistemas informáticos de múltiples procesadores que tienen dispositivos informáticos distintos a una GPU, además o en lugar de la GPU 314. En algunos ejemplos, los dispositivos informáticos en el sistema informático 310 pueden ser dispositivos informáticos de Open CL. Además, el sistema informático ejemplar 310 mostrado en la FIG. 20 ilustra una infraestructura y técnicas para la implementación de objetos de memoria inmediata que facilitan la compartición de datos en proceso entre un dispositivo anfitrión y un dispositivo informático. En otros sistemas informáticos ejemplares, sin embargo, las técnicas pueden extenderse inmediatamente para proporcionar la compartición de datos en proceso entre diferentes dispositivos informáticos (por ejemplo, dispositivos informáticos de OpenCL) en un sistema informático que tiene más de un dispositivo informático. En dichos ejemplos, pueden conectarse una o más líneas de interrupción entre diferentes dispositivos informáticos.

La FIG. 21 es un diagrama de flujo que ilustra una técnica ejemplar para la ejecución de una instrucción de creación de objetos de memoria, emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 21. La instrucción de creación de objetos de memoria puede ser una instrucción de creación de objetos de memoria intermedia o una instrucción de creación de objetos de imagen. La interfaz de memoria intermedia 328 recibe una instrucción de creación de objetos de memoria (340). La interfaz de memoria intermedia 328 determina si las instrucciones de creación de objetos de memoria especifican que la modalidad inmediata debería ser habilitada para el objeto de memoria (342). Por ejemplo, la interfaz de memoria intermedia 328 puede determinar si un parámetro de indicador inmediato está incluido en una lista de parámetros para la instrucción de creación de objetos de memoria.

Si la interfaz de memoria intermedia 328 determina que la instrucción de creación de objetos de memoria no especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria, entonces la interfaz de memoria intermedia 328 puede asignar un espacio de memoria compartida 336 al objeto de memoria que se creará (344), hacer que la prestación de los servicios de almacenamiento en memoria caché, mediante una entre la memoria caché del anfitrión 324 y la memoria caché de GPU 332, o ambas, esté habilitada para el espacio de memoria compartida 336 (346), y devolver una referencia al objeto de memoria creado (348). La instrucción de creación de objetos de memoria puede especificar que la modalidad inmediata no se debería habilitar, por ejemplo, no incluyendo un parámetro de indicador inmediato, o especificando con otro valor de parámetro que la modalidad inmediata no debería ser habilitada. Por el contrario, si la interfaz de memoria intermedia 328 determina que la instrucción de creación del objetos de memoria especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria, entonces la interfaz de memoria intermedia 328 puede asignar un espacio de memoria compartida 336 para el objeto de memoria que debe crearse (350), hacer que la prestación de los servicios de almacenamiento en memoria caché, mediante una entre la memoria caché del anfitrión 324 y la memoria caché de GPU 332, o ambas, sea inhabilitada para el espacio de memoria compartida 336 (352), y devolver una referencia al

objeto de memoria creado (354). La instrucción de creación de objetos de memoria puede especificar que la modalidad inmediata debería ser habilitada, por ejemplo, incluyendo un parámetro de indicador inmediato, o especificando con otro valor de parámetro que la modalidad inmediata debería estar habilitada.

En algunos ejemplos, la interfaz de memoria intermedia 328 puede hacer que la prestación de los servicios de almacenamiento en memoria caché sea habilitada para el espacio de memoria compartida 336, mediante la fijación de un atributo de modalidad inmediata del objeto de memoria asociado con el espacio de memoria compartida 336, en un valor que indica que deberían proporcionarse servicios de almacenamiento en memoria caché para el objeto de memoria asociado con el espacio de memoria compartida 336. Análogamente, la interfaz de memoria intermedia 328 puede hacer que la prestación de los servicios de almacenamiento en memoria caché sea inhabilitada para el espacio de memoria compartida 336, fijando el atributo de modalidad inmediata del objeto de memoria asociado con el espacio de memoria compartida 336, en un valor que indica que no deberían proporcionarse servicios de almacenamiento en memoria caché para el objeto de memoria asociado con el espacio de memoria compartida 336. El objeto de memoria devuelto puede incluir el atributo de modalidad inmediata. En tales ejemplos, el atributo de modalidad inmediata para el objeto de memoria puede ser accesible por parte de uno entre el proceso anfitrión 320 que se está ejecutando en el dispositivo anfitrión 312 y la tarea 330 que se está ejecutando en la GPU 314, o ambos. El proceso anfitrión 320 y/o la tarea 330 pueden determinar si se utilizan los servicios de almacenamiento en memoria caché al ejecutar instrucciones de lectura y escritura particulares con respecto al espacio de memoria compartida 336, en base al atributo de modalidad inmediata del objeto de memoria asociado con el espacio de memoria compartida 336.

En otros ejemplos, la interfaz de memoria intermedia 328 puede hacer que la prestación de los servicios de almacenamiento en memoria caché sea habilitada para el espacio de memoria compartida 336, mediante la configuración de uno o más indicadores inmediatos, basados en hardware, asociados al espacio de memoria compartida 336, en un valor que indica que deberían proporcionarse servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336. Análogamente, la interfaz de memoria intermedia 328 puede hacer que la prestación de los servicios de almacenamiento en memoria caché sea inhabilitada para el espacio de memoria compartida 336, mediante la configuración de uno o más indicadores inmediatos, basados en hardware, asociados al espacio de memoria compartida 336, en un valor que indica que no deberían proporcionarse servicios de almacenamiento en memoria caché para el espacio de memoria compartida 336. Uno o más indicadores inmediatos basados en hardware pueden estar situados en uno o más entre el módulo de control de memoria caché del anfitrión 326 y un módulo control de memoria caché de GPU 334 u otra unidad de gestión de memoria local o global (no mostrada).

En ejemplos adicionales, la interfaz de memoria intermedia 328 puede devolver el objeto de memoria al proceso llamante, por ejemplo, el proceso anfitrión 320, antes de la asignación del espacio de memoria física en la memoria 316 para almacenar los datos. En tales ejemplos, la interfaz de memoria intermedia 328 puede incluir el atributo de modalidad inmediata en el objeto de memoria devuelto.

Entonces, cuando la memoria 316 se asigna en un momento posterior para el objeto de memoria, la interfaz de memoria intermedia 328 u otro módulo puede configurar los uno o más indicadores inmediatos basados en hardware, en base al atributo de modalidad inmediata del objeto de memoria.

La FIG. 22 es un diagrama de flujo que ilustra otra técnica ejemplar para la ejecución de una instrucción de creación de objetos de memoria, emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 22. La instrucción de creación de objetos de memoria puede ser una instrucción de creación de objetos de memoria intermedia o una instrucción de creación de objetos de imagen. La interfaz de memoria intermedia 328 recibe una instrucción de creación de objetos de memoria (356). La interfaz de memoria intermedia 328 determina si la instrucción de creación de objetos de memoria especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria (358). Por ejemplo, la interfaz de memoria intermedia 328 puede determinar si un parámetro de indicador inmediato está incluido en una lista de parámetros para la instrucción de creación de objetos de memoria.

Si la interfaz de memoria intermedia 328 determina que la instrucción de creación de objetos de memoria no especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria, entonces la interfaz de memoria intermedia 328 puede asignar un espacio de memoria compartida 336 para el objeto de memoria que va a ser creado (360), inhabilitar una modalidad de coherencia de memoria caché de la memoria compartida para el espacio de memoria compartida 336 (362), y devolver una referencia al objeto de memoria creado (364). Por el contrario, si la interfaz de memoria intermedia 328 determina que la instrucción de creación de objetos de memoria especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria, entonces la interfaz de memoria intermedia 328 puede asignar un espacio de memoria compartida 336 para el objeto de memoria que va a ser creado (366), habilitar una modalidad de coherencia de memoria caché de la memoria compartida para el espacio de memoria compartida 336 (368), y devolver una referencia al objeto de memoria creado (370).

En algunos ejemplos, la interfaz de memoria intermedia 328 puede devolver el objeto de memoria al proceso

llamante, por ejemplo, el proceso anfitrión 320, antes de la asignación del espacio de memoria física en la memoria 316 para almacenar los datos. En tales ejemplos, la interfaz de memoria intermedia 328 puede incluir el atributo de modalidad inmediata en el objeto de memoria devuelto. Entonces, cuando la memoria 316 se asigna en un momento posterior para el objeto de memoria, la interfaz de memoria intermedia 328 u otro módulo puede habilitar o inhabilitar la modalidad de coherencia de memoria caché del espacio de memoria compartida, en base al atributo de modalidad inmediata del objeto de memoria.

Las FIGs. 23 a 28 ilustran técnicas ejemplares que una GPU puede utilizar para procesar las instrucciones de carga y almacenamiento en la modalidad inmediata y en la modalidad de memoria caché, de acuerdo con esta divulgación. Como se ha expuesto anteriormente, el código fuente para la tarea 330 puede, en algunos ejemplos, compilarse de forma que el código compilado pueda incluir tanto las instrucciones en la modalidad de memoria caché como las instrucciones en la modalidad inmediata, con el fin de prestar soporte tanto a los objetos de memoria inmediata como a los objetos en memoria caché. Las instrucciones en modalidad de memoria caché pueden ejecutar operaciones de lectura y escritura con respecto a una memoria, utilizando los servicios de almacenamiento en memoria caché de una memoria caché asociada con la memoria subyacente, y las instrucciones de modalidad inmediata pueden ejecutar operaciones de lectura y escritura con respecto a una memoria sin necesidad de utilizar servicios de almacenamiento en memoria caché de una memoria caché asociada con la memoria subyacente. Las instrucciones en modalidad de memoria caché pueden denominarse, de forma alternativa en el presente documento, instrucciones de modalidad no inmediata. Las instrucciones de carga y almacenamiento pueden, de forma alternativa, denominarse en el presente documento instrucciones de lectura y escritura, respectivamente.

En algunos ejemplos, la versión en modalidad de memoria caché de una instrucción de carga o almacenamiento y la versión en modalidad inmediata de la instrucción de carga o almacenamiento pueden ser instrucciones diferentes, por ejemplo, teniendo cada una un código de operación, es decir, codeop, diferente. En ejemplos adicionales, la versión en modalidad de memoria caché de una instrucción de carga o almacenamiento y la versión en modalidad inmediata de la instrucción de carga o almacenamiento pueden ser la misma instrucción, por ejemplo, teniendo las dos el mismo código de operación. En tales ejemplos, un parámetro proporcionado con la instrucción puede especificar si la instrucción es en modalidad de memoria caché o en modalidad inmediata.

La FIG. 23 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de instrucciones en modalidad de memoria caché y en modalidad inmediata, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 23. En el ejemplo de la FIG. 23, la modalidad inmediata se denomina una modalidad de omisión de memoria caché, y las instrucciones en la modalidad inmediata se corresponden con las instrucciones en la modalidad de omisión de memoria caché. El módulo de control de memoria caché de GPU 334 recibe una instrucción de carga que especifica una ubicación de memoria y si está habilitada una modalidad de omisión de memoria caché (372). El módulo de control de memoria caché de GPU 334 determina si la instrucción de carga especifica que la modalidad de omisión de memoria caché está habilitada (374). En algunos casos, el módulo de control de memoria caché de GPU 334 puede determinar si la instrucción de carga especifica que la modalidad de omisión de la memoria caché está habilitada, en base al tipo de instrucción, por ejemplo, el código de operación de la instrucción. En casos adicionales, el módulo de control de memoria caché de GPU 334 puede determinar si la instrucción de carga especifica que la modalidad de omisión de memoria caché está habilitada, en base a un parámetro incluido con la instrucción de carga, que es indicativo de si la modalidad de omisión de memoria caché está habilitada. Si el módulo de control de memoria caché de GPU 334 determina que la modalidad de omisión de memoria caché no está habilitada, entonces el módulo de control de memoria caché de GPU 334 recupera los datos desde una memoria caché, por ejemplo, la memoria caché de GPU 332, en una ubicación de memoria caché asociada a la ubicación de memoria especificada en la instrucción de carga (376). Por otra parte, si el módulo de control de memoria caché de GPU 334 determina que la modalidad de omisión de memoria caché está habilitada, entonces el módulo de control de memoria caché de GPU 334 recupera los datos desde una memoria, por ejemplo, el espacio de memoria compartida 336, en la ubicación de memoria especificada en la instrucción de carga (378).

La FIG. 24 es un diagrama de flujo que ilustra otra técnica ejemplar para el procesamiento de instrucciones en la modalidad de memoria caché y en la modalidad inmediata, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 24. En el ejemplo de la FIG. 24, la modalidad inmediata se denomina modalidad de omisión de memoria caché, y las instrucciones en la modalidad inmediata se corresponden con las instrucciones en la modalidad de omisión de memoria caché. El módulo de control de memoria caché de GPU 334 recibe una instrucción de almacenamiento que especifica una ubicación de memoria, datos para almacenar y si una modalidad de omisión de memoria caché está habilitada (380). El módulo de control de memoria caché de GPU 334 determina si la instrucción de almacenamiento especifica que la modalidad de omisión de memoria caché está habilitada (382). En algunos casos, el módulo de control de memoria caché de GPU 334 puede determinar si la instrucción de almacenamiento especifica que la modalidad de omisión de memoria caché se habilita, en función del tipo de instrucción, por ejemplo, el código de operación de la instrucción. En casos adicionales, el módulo de control de memoria caché de GPU 334 puede determinar si la instrucción de almacenamiento especifica que la modalidad de omisión de memoria caché está habilitada, en base a un parámetro, incluido con la instrucción de carga, que es indicativo de si la modalidad de omisión de memoria caché está habilitada. Si el módulo de control de memoria caché de GPU 334 determina que la

modalidad de omisión de memoria caché no está habilitada, entonces el módulo de control de memoria caché de GPU 334 almacena los datos especificados en la instrucción de almacenamiento en una memoria caché, por ejemplo, la memoria caché de GPU 332, en una ubicación de memoria caché asociada a la ubicación de memoria especificada en la instrucción de almacenamiento (384). Por otra parte, si el módulo de control de memoria caché de GPU 334 determina que la modalidad de omisión de memoria caché está habilitada, entonces el módulo de control de memoria caché de GPU 334 almacena los datos especificados en la instrucción de almacenamiento en una memoria, por ejemplo, el espacio de memoria compartida 336, en la ubicación de memoria especificada en la instrucción de almacenamiento (386).

La FIG. 25 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de instrucciones en modalidad de memoria caché y en modalidad inmediata, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 25. El módulo de control de memoria caché de GPU 334 recibe una instrucción de almacenamiento que especifica una ubicación de memoria, datos para almacenar y si una modalidad inmediata está habilitada (388). El módulo de control de memoria caché de GPU 334 almacena los datos especificados en la instrucción de almacenamiento en una memoria caché, por ejemplo, la memoria caché de GPU 332, en una ubicación de memoria caché asociada a la ubicación de memoria especificada en la instrucción de almacenamiento (390). El módulo de control de memoria caché de GPU 334 determina si la modalidad inmediata está habilitada en base a la información de la instrucción de almacenamiento que especifica si la modalidad inmediata está habilitada (392). La información que especifica si la modalidad inmediata está habilitada puede ser, en algunos ejemplos, el tipo de instrucción, por ejemplo, el código de operación para la instrucción y/o un parámetro incluido en la instrucción que especifica si la modalidad inmediata está habilitada para la instrucción. Si la modalidad inmediata no está habilitada, el módulo de control de memoria caché de GPU 334 no lleva a cabo una operación inmediata de re-escritura en memoria caché (394). Por otro lado, si la modalidad inmediata está habilitada, el módulo de control de memoria caché de GPU 334 realiza una operación inmediata de re-escritura en memoria caché (396).

La FIG. 26 es un diagrama de flujo que ilustra otra técnica ejemplar para el procesamiento de instrucciones en modalidad de memoria caché y en modalidad inmediata, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 26. El módulo de control de memoria caché de GPU 334 recibe una instrucción de carga que especifica una ubicación de memoria y si una modalidad inmediata está habilitada (398). El módulo de control de memoria caché de GPU 334 determina si la modalidad inmediata está habilitada en base a la información de la instrucción de carga que especifica si la modalidad inmediata está habilitada (400). La información que especifica si la modalidad inmediata está habilitada puede ser, en algunos ejemplos, el tipo de instrucción, por ejemplo, el código de operación para la instrucción y/o un parámetro incluido en la instrucción que especifica si la modalidad inmediata está habilitada para la instrucción. Si la modalidad inmediata no está habilitada, el módulo de control de memoria caché de GPU 334 no vacía e invalida la memoria caché (402). El módulo de control de memoria caché de GPU 334 recupera los datos especificados en la instrucción de carga, ya sea desde una memoria caché, por ejemplo, la memoria caché de GPU 332, si los datos están disponibles en la memoria caché o bien, si los datos no están disponibles en la memoria caché, desde la memoria subyacente (404). Si la modalidad inmediata está habilitada, el módulo de control de memoria caché de GPU 334 vacía e invalida la memoria caché (406). El módulo de control de memoria caché de GPU 334 recupera los datos especificados en la instrucción de carga desde la memoria subyacente (408). La memoria caché no devuelve los datos, ya que la memoria caché se ha vaciado e invalidado.

La FIG. 27 es un diagrama de bloques que ilustra una GPU ejemplar 420 que puede utilizarse en el sistema informático 310 de la FIG. 20 de acuerdo con esta divulgación. En algunos ejemplos, la GPU 420 puede utilizarse para implementar la GPU 314 ilustrada en la FIG. 20. La GPU 420 incluye un módulo de procesamiento de GPU 422, un módulo de control de memoria caché de GPU 424, una memoria caché de GPU 426, un bus de memoria caché 428 y un bus de omisión 430. El módulo de procesamiento de GPU 422 está acoplado comunicativamente al módulo de control de memoria caché de GPU 424 mediante el bus de memoria caché 428. El módulo de procesamiento de GPU 422 también está acoplado comunicativamente a la memoria 316 mediante el bus de omisión 430. El módulo de control de memoria caché de GPU 424 y la memoria caché de GPU 426 son esencialmente similares al módulo de control de memoria caché de GPU 334 y a la memoria caché de GPU 332 en la FIG. 20, y no se describirán con más detalle. El módulo de procesamiento de GPU 422 incluye un elemento de procesamiento 432 y un controlador de bus 434. El elemento de procesamiento 432 está configurado para emitir instrucciones de carga y almacenamiento al controlador de bus 434.

El controlador de bus 434 puede estar configurado para remitir instrucciones de carga y almacenamiento a ubicaciones adecuadas mediante el bus de memoria caché 428 y el bus de omisión 430. El controlador de bus 434 puede estar configurado para funcionar en una modalidad inmediata o una modalidad no inmediata, en base a información de la instrucción de carga o almacenamiento que es indicativa de si la instrucción es una instrucción de modalidad inmediata o de modalidad de memoria caché. Cuando el controlador de bus 434 está configurado para funcionar en una modalidad no inmediata, es decir, una modalidad de memoria caché, el controlador de bus 434 puede usar el bus de memoria caché 428 para enviar instrucciones de carga y almacenamiento al módulo de control de memoria caché de GPU 424, para su ejecución. Por otro lado, cuando el controlador de bus 434 está configurado para funcionar en una modalidad inmediata, el controlador de bus 434 puede usar el bus de omisión 430 para remitir

instrucciones de carga y almacenamiento a la memoria 316, para su ejecución.

La FIG. 28 es un diagrama de flujo que ilustra una técnica ejemplar para el procesamiento de instrucciones en modalidad de memoria caché y en modalidad inmediata, de acuerdo con esta divulgación. En algunos ejemplos, se puede utilizar la GPU 420 de la FIG. 27 para implementar la técnica ejemplar que se muestra en la FIG. 28. El controlador de bus 434 recibe una instrucción de carga o almacenamiento (440). El controlador de bus 434 determina si la modalidad inmediata está habilitada, en base a la información de la instrucción de carga o almacenamiento que especifica si la modalidad inmediata está habilitada (442). La información que especifica si la modalidad inmediata está habilitada puede ser, en algunos ejemplos, el tipo de instrucción, por ejemplo, el código de operación para la instrucción y/o un parámetro incluido en la instrucción que especifica si la modalidad inmediata está habilitada para la instrucción. Si el controlador de bus 434 determina que la modalidad inmediata no está habilitada, entonces el controlador de bus 434 utiliza el bus de memoria caché 428 para enviar la instrucción recibida al módulo de control de memoria caché de GPU 424 (444). De lo contrario, si el controlador de bus 434 determina que la modalidad inmediata está habilitada, entonces el controlador de bus 434 utiliza el bus de omisión 430 para remitir la instrucción recibida a la memoria 316 (446).

La FIG. 29 es un diagrama de flujo que ilustra otra técnica ejemplar para la ejecución de una instrucción de creación de objetos de memoria, emitida por un proceso que se está ejecutando en un dispositivo anfitrión, de acuerdo con esta divulgación. En algunos ejemplos, puede utilizarse el sistema informático 310 de la FIG. 20 para implementar la técnica ejemplar que se muestra en la FIG. 29. La instrucción de creación de objetos de memoria puede ser una instrucción de creación de objetos de memoria intermedia o una instrucción de creación de objetos de imagen. La interfaz de memoria intermedia 328 recibe una instrucción de creación de objetos de memoria (448). La interfaz de memoria intermedia 328 determina si la instrucción de creación de objetos de memoria especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria (450). Por ejemplo, la interfaz de memoria intermedia 328 puede determinar si un parámetro de indicador inmediato está incluido en una lista de parámetros para la instrucción de creación de objetos de memoria.

Si la interfaz de memoria intermedia 328 determina que la instrucción de creación de objetos de memoria no especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria, entonces la interfaz de memoria intermedia 328 fija un atributo de modalidad inmediata para el objeto de memoria creado en un valor indicativo de que la modalidad inmediata no está permitida, por ejemplo, "falso" (452). Por otra parte, si la interfaz de memoria intermedia 328 determina que la instrucción de creación de objetos de memoria especifica que la modalidad inmediata debería ser habilitada para el objeto de memoria, entonces la interfaz de memoria intermedia 328 fija un atributo de modalidad inmediata para el objeto de memoria creado en un valor indicativo de que la modalidad inmediata está habilitada, por ejemplo, "verdadero" (454). El atributo de modalidad inmediata del objeto de memoria puede ser utilizado por el dispositivo anfitrión 312 y/o la GPU 314, en algunos ejemplos, para determinar si se ejecutan las operaciones de lectura y escritura en la modalidad inmediata o la modalidad de memoria caché al acceder a datos almacenados en un objeto de memoria particular.

En algunos ejemplos, el proceso anfitrión 320 y/o la tarea 330 pueden desear programar algunos objetos de memoria para que sean objetos de memoria inmediata, y otros objetos para que sean objetos de memoria en memoria caché, es decir, objetos de memoria no inmediata. Las técnicas de esta divulgación pueden, en algunos ejemplos, incluir técnicas de compilación especializadas que permiten que una tarea compilada 330 realice operaciones de lectura y escritura con respecto tanto a los objetos de memoria caché como a los objetos de memoria inmediata. Una primera técnica ejemplar de compilación puede compilar una operación de lectura o una operación de escritura dada en una secuencia de instrucciones. La secuencia de instrucciones puede comprobar el valor de un atributo de modalidad inmediata para un objeto de memoria a leer o escribir, y determinar si se ejecuta una instrucción en modalidad de memoria caché o una instrucción en modalidad inmediata, en base al valor del atributo de modalidad inmediata. Una segunda técnica ejemplar de compilación puede utilizar información en el código fuente que es indicativa de si el objeto de memoria es un objeto de modalidad inmediata, para seleccionar instrucciones en modalidad de memoria caché o instrucciones en modalidad inmediata, para su uso en el código compilado para acceder al objeto de memoria.

De acuerdo con la primera técnica ejemplar de compilación, un compilador puede compilar código fuente para la tarea 330 de tal manera que el código compilado para la tarea 330 incluya una secuencia de lectura de acuerdo con el siguiente pseudo-código ejemplar:

```
si ( esInmediata ) {
    lectura_inmediata (…)
}
si no {
lectura_en_memoria_caché (…)
}
```

donde "esInmediata" representa un atributo booleano de modalidad inmediata para un objeto de memoria desde el cual se leerán datos, "lectura_inmediata (...)" representa una instrucción de lectura en modalidad inmediata y

"lectura_en_memoria_caché (...)" representa una instrucción de lectura en modalidad de memoria caché.

El módulo de control de memoria caché de GPU 334 puede procesar la instrucción lectura_inmediata (...) mediante, por ejemplo, la invalidación de la memoria caché de GPU 332, si se utiliza, antes de leer los datos de la memoria caché de GPU 332. El módulo de control de memoria caché de GPU 334 puede procesar la instrucción lectura_en_memoria_caché (...) mediante, por ejemplo, la lectura de datos de la memoria caché de GPU 332 de una forma normal, por ejemplo, sin invalidar la memoria caché antes de realizar la lectura.

De acuerdo con la primera técnica ejemplar de compilación, un compilador puede compilar el código fuente para la tarea 330 de tal manera que el código compilado para la tarea 330 incluya una secuencia de escritura de acuerdo con el siguiente pseudo-código ejemplar:

```
si ( esInmediata ) {
    escritura_inmediata (…)
}
si no {
escritura_en_memoria_caché (…)
}
```

donde "esInmediata" representa un atributo booleano de modalidad inmediata para un objeto de memoria en el que se escribirán datos, "escritura_inmediata (...) " representa una instrucción de escritura en modalidad inmediata, y "escritura_en_memoria_caché (...) " representa una instrucción de escritura en modalidad de memoria caché.

El módulo de control de memoria caché de GPU 334 puede procesar la instrucción escritura_inmediata (...), en algunos ejemplos, mediante el uso de una modalidad de escritura simultánea para la memoria caché de GPU 332, si se utiliza una memoria caché. En ejemplos adicionales, el módulo de control de memoria caché de GPU 334 puede procesar la instrucción escritura_inmediata (...) escribiendo datos en la memoria caché de GPU 332 si se utiliza una memoria caché, y realizando un vaciado de memoria caché para la memoria caché de GPU 332 en respuesta a la escritura de los datos en la memoria caché de GPU 332. El módulo de control de memoria caché de GPU 334 puede procesar la instrucción escritura_en_memoria_caché (...) escribiendo datos en la memoria caché de GPU 332 de una forma normal, por ejemplo, sin utilizar una modalidad de escritura simultánea y/o sin vaciar la memoria caché en respuesta a la operación de escritura.

La FIG. 30 es un diagrama de flujo que ilustra cómo una GPU puede procesar una secuencia de instrucciones compiladas de acuerdo con la primera técnica de compilación descrita anteriormente. En algunos ejemplos, se puede usar la técnica ilustrada en la FIG. 30 para implementar el pseudo-código ejemplar proporcionado anteriormente para las secuencias de lectura y escritura. La tarea 330 comienza una secuencia de lectura o una secuencia de escritura (456). Por ejemplo, la tarea 330 puede comenzar la secuencia de lectura o la secuencia de escritura cuando la tarea 330 llega a un punto en la ejecución de la tarea 330 donde debería producirse una instrucción de lectura o escritura para un objeto de memoria particular. La tarea 330 accede al atributo de modalidad inmediata, asociado con el objeto de memoria del cual se leerán los datos o en el cual se escribirán los datos (458). La tarea 330 determina si el atributo para el objeto de memoria se fija en un valor que indica que la modalidad inmediata está habilitada, por ejemplo, "verdadero" (460). Si la tarea 330 determina que el atributo para el objeto de memoria se fija en un valor indicativo de que la modalidad inmediata no está habilitada, entonces la tarea 330 realiza una operación de lectura o escritura en memoria caché para el objeto de memoria utilizando una instrucción de lectura o escritura en memoria caché (462). De lo contrario, si la tarea 330 determina que el atributo para el objeto de memoria se fija en un valor que indica que la modalidad inmediata está habilitada, entonces la tarea 330 realiza una operación de lectura o escritura inmediata para el objeto de memoria utilizando una instrucción de lectura o escritura inmediata (464).

De acuerdo con la segunda técnica ejemplar de compilación, al compilar el código fuente, el compilador puede tener acceso a la información indicativa de si la modalidad inmediata está habilitada para un objeto de memoria particular del cual lea o en el cual escriba la tarea 330. El compilador puede utilizar esta información para compilar el código fuente para la tarea 330, para seleccionar entre las instrucciones de lectura y escritura en modalidad de memoria caché y las instrucciones de lectura y escritura en modalidad inmediata cuando la tarea 330 lee de, o escribe en, el objeto de memoria particular.

En algunos ejemplos, la información indicativa de si la modalidad inmediata está habilitada para un objeto de memoria particular puede ser un atributo de tiempo de compilación, indicativo de si la modalidad inmediata está habilitada para uno o más objetos de memoria a los que accede el código fuente para la tarea 330. Por ejemplo, el código fuente para la tarea 330, por ejemplo, el código fuente del núcleo, puede incluir un atributo de tiempo de compilación indicativo de si la modalidad inmediata está habilitada para uno o más objetos de memoria utilizados por la tarea 330. El atributo de tiempo de compilación puede, en algunos casos, tomar la forma de un calificador de atributo de OpenCL, tal como, por ejemplo, _cl_immediate. El calificador de atributo puede estar asociado con uno o más objetos de memoria particulares y/o una o más variables que se almacenan dentro de los uno o más objetos de memoria. Cuando el calificador de atributo está asociado con un objeto de memoria particular, el compilador puede

determinar que la modalidad inmediata está habilitada para el objeto de memoria. De forma similar, cuando el calificador de atributo no está asociado con un objeto de memoria particular, el compilador puede determinar que la modalidad inmediata no está habilitada para el objeto de memoria. Usar un atributo de este tipo puede reducir el trabajo para el compilador y reducir potencialmente el tamaño del núcleo. En algunos ejemplos, las aplicaciones de software pueden limitar el uso de memorias intermedias inmediatas para los casos en que se necesitan tales memorias intermedias. En tales ejemplos, la decisión de utilizar o no memorias intermedias inmediatas puede ser una decisión en tiempo de compilación.

Si el atributo de tiempo de compilación indica que la modalidad inmediata está habilitada para el objeto de memoria asociado con el espacio de memoria compartida 336, entonces el compilador puede compilar la tarea 330 de tal manera que el código compilado para la tarea 330 incluya instrucciones de lectura y/o escritura en modalidad inmediata para operaciones de lectura o escritura que se realizan con respecto al espacio de memoria compartida 336. De lo contrario, si la modalidad inmediata no está habilitada para el objeto de memoria asociado con el espacio de memoria compartida 336, a continuación, el compilador puede compilar la tarea 330 de tal manera que el código compilado para la tarea 330 no incluya instrucciones de lectura y/o escritura en modalidad inmediata para las operaciones de lectura o escritura que se realizan con respecto al espacio de memoria compartida 336. Por ejemplo, el compilador puede compilar la tarea 330 de tal manera que el código compilado para la tarea 330 incluya instrucciones de lectura y/o escritura en memoria caché para las operaciones de lectura o escritura que se realizan con respecto al espacio de memoria compartida 336.

La FIG. 31 es un diagrama de flujo que ilustra una técnica ejemplar para la compilación de código fuente para una tarea de acuerdo con esta divulgación. El código resultante, compilado utilizando las técnicas en la FIG. 31 puede, en algunos ejemplos, corresponder a la tarea 330 en la FIG. 20. En la técnica ejemplar de la FIG. 31, la tarea 330 se denomina un núcleo. El compilador procesa un argumento del núcleo que es implementado por un objeto de memoria (466).

El compilador determina si el objeto de memoria es un objeto de memoria en modalidad inmediata (468). En algunos ejemplos, el compilador puede determinar si el objeto de memoria es un objeto de memoria en modalidad inmediata en base a la información incluida en el código fuente del núcleo, por ejemplo, un atributo de tiempo de compilación asociado con el argumento de núcleo. Si el compilador determina que el objeto de memoria no es un objeto de memoria en modalidad inmediata, entonces el compilador compila las operaciones de lectura y las operaciones de escritura asociadas con el argumento de núcleo particular usando instrucciones de lectura y escritura en memoria caché (470). Por otro lado, si el compilador determina que el objeto de memoria es un objeto de memoria en la modalidad inmediata, entonces el compilador compila las operaciones de lectura y las operaciones de escritura asociadas con el argumento de núcleo particular usando las instrucciones de lectura y escritura en modalidad inmediata (472).

La FIG. 32 es un diagrama de flujo que ilustra una técnica ejemplar que puede ser utilizada por una GPU para utilizar selectivamente los servicios de almacenamiento en memoria caché de acuerdo con esta divulgación. Por ejemplo, las técnicas pueden permitir que una GPU utilice selectivamente una memoria caché de GPU asociada con una memoria para ejecutar al menos una entre una operación de lectura y una operación de escritura, con respecto a un espacio de memoria de la memoria, en respuesta a la recepción de información que especifica si se deberían utilizar los servicios de almacenamiento en memoria caché para la ejecución de al menos una de las operaciones de lectura y las operaciones de escritura con respecto al espacio de memoria. En algunos ejemplos, la GPU 314 ilustrada en la FIG. 20 y/o la GPU 420 ilustrada en la FIG. 27 pueden utilizarse para implementar las técnicas ilustradas en la FIG. 32.

La GPU 314 recibe al menos una entre una instrucción de lectura y una instrucción de escritura para procesar (474). La instrucción recibida puede instruir a la GPU 314 para ejecutar al menos una entre una operación de lectura y una operación de escritura, con respecto a un espacio de memoria de una memoria. La GPU 314 recibe información de la modalidad de memoria caché que especifica si los servicios de almacenamiento en memoria caché se deberían utilizar para la ejecución de al menos una entre las operaciones de lectura y las operaciones de escritura, con respecto al espacio de memoria (476). En algunos ejemplos, la información de modalidad de memoria caché puede estar incluida dentro de la instrucción recibida. En ejemplos adicionales, la información de modalidad de memoria caché puede ser un atributo de modalidad inmediata de un objeto de memoria asociado con el espacio de memoria. La GPU 314 determina si se utilizan los servicios de almacenamiento en memoria caché en base a la información de modalidad de memoria caché (478). En respuesta a la recepción de información que especifica que se deberían utilizar los servicios de almacenamiento en memoria caché para la ejecución de la instrucción recibida, la GPU 314 puede utilizar los servicios de almacenamiento en memoria caché para ejecutar la instrucción recibida (480). En respuesta a la recepción de información que especifica que no se deberían utilizar los servicios de almacenamiento en memoria caché para la ejecución de la instrucción recibida, la GPU 314 puede no utilizar los servicios de almacenamiento en memoria caché para ejecutar la instrucción recibida (482). En algunos ejemplos, la GPU 314 puede utilizar una o más de las técnicas ilustradas en las FIGs. 23 a 28 y 30 para implementar uno o más entre el cuadro de decisión 478 y los cuadros de proceso 480 y 482. En algunos casos, un módulo de control de memoria caché de GPU o una unidad de gestión de memoria, por ejemplo, el módulo de control de memoria caché de GPU 334 que se ilustra en la FIG. 20, puede utilizarse para implementar las técnicas mostradas en la FIG. 32. En casos

adicionales, un controlador de bus, por ejemplo, el controlador de bus 434 que se ilustra en la FIG. 27, puede utilizarse para implementar las técnicas mostradas en la FIG. 32.

En algunos ejemplos, con el fin de implementar los objetos de memoria inmediata, puede diseñarse una ALU de GPU para ejecutar una instrucción de ALU que invalida la memoria caché de la memoria global y/o una parte específica de la memoria caché de la memoria global, especificada en la instrucción. En general, el dispositivo anfitrión 312 puede utilizar las capacidades de CPU existentes para implementar los objetos de memoria inmediata.

Varios casos de uso para las técnicas de señalización fuera de banda, descritas en esta divulgación, por ejemplo, las técnicas de paso de mensajes descritas en el presente documento, y los objetos de memoria inmediata descritos en esta divulgación, ahora serán expuestos en más detalle. De acuerdo con un primer caso de uso, la señalización fuera de banda se puede utilizar como una característica independiente sin usar necesariamente objetos de memoria inmediata, además de las técnicas de señalización fuera de banda. La señalización fuera de banda se puede usar para la sincronización y para pasar una cantidad relativamente pequeña de datos de forma rápida. En algunos ejemplos, la señalización fuera de banda puede tener una latencia más baja que los objetos de memoria inmediata, pero tener un ancho de banda más bajo que los objetos de memoria inmediata.

La señalización fuera de banda también se puede usar de acuerdo con el primer caso de uso para las operaciones de asignación de memoria. Por ejemplo, una GPU puede utilizar la señalización fuera de banda para solicitar que la CPU anfitriona asigne una nueva memoria intermedia. La GPU puede también utilizar la señalización fuera de banda para especificar a la CPU anfitriona una longitud de memoria intermedia solicitada. Como otro ejemplo, una CPU puede utilizar la señalización fuera de banda después de la asignación de una memoria intermedia para enviar un puntero a la GPU que especifica una ubicación de memoria para la memoria intermedia.

La señalización fuera de banda también se puede usar de acuerdo con el primer caso de uso para llamadas a procedimientos remotos donde se va a intercambiar una pequeña cantidad de datos. Por ejemplo, en un caso en el que un núcleo que se está ejecutando en una unidad de cálculo dentro de un dispositivo informático utiliza una RPC para lanzar otro núcleo en otra unidad de cálculo, ya sea en el mismo dispositivo informático o en otro dispositivo informático, los datos de la RPC podrían almacenarse en la memoria local de la unidad de cálculo de lanzamiento. Las técnicas de señalización fuera de banda de esta divulgación pueden utilizarse para transferir los datos desde la memoria local de la unidad de cálculo de lanzamiento a la memoria local de la unidad de cálculo que ejecuta el núcleo recién lanzado.

La señalización fuera de banda también se puede usar de acuerdo con el primer caso de uso para la notificación de avances. Por ejemplo, una GPU puede utilizar la señalización fuera de banda para informar del porcentaje de avance de la tarea actual a la CPU anfitriona.

La señalización fuera de banda también se puede usar de acuerdo con el primer caso de uso para la notificación de errores. Por ejemplo, una GPU puede utilizar la señalización fuera de banda para notificar un código de error a la CPU anfitriona.

La señalización fuera de banda también se puede usar de acuerdo con el primer caso de uso para facilitar los cambios de contexto. Por ejemplo, una CPU anfitriona puede utilizar la señalización fuera de banda para solicitar que una GPU guarde un estado para prepararse para un cambio de contexto

De acuerdo con un segundo caso de uso, los objetos de memoria inmediata se pueden usar como una característica independiente sin usar necesariamente la señalización fuera de banda, además de los objetos de memoria inmediata. Por ejemplo, las memorias intermedias inmediatas se pueden utilizar para llevar a cabo el intercambio de una cantidad relativamente grande de datos. Las memorias intermedias inmediatas pueden contener no solo datos, sino también marcadores de sincronización. En este caso, un productor de datos puede primero escribir datos en la memoria intermedia, y a continuación escribir un marcador de sincronización que indica la disposición y/o la ubicación de los datos a un consumidor. El consumidor busca unos datos de sincronización en una ubicación decidida a priori, tal como, por ejemplo, en la sección de cabecera de la memoria intermedia mediante el sondeo de esta ubicación de memoria. Una vez obtenido el marcador de sincronización, el consumidor lee los datos. Pueden aplicarse técnicas similares a los objetos de imágenes inmediatos.

Pueden emplearse múltiples protocolos de sincronización para estas técnicas. Por ejemplo, los marcadores de sincronización pueden estar integrados dentro de la memoria intermedia de datos, o pueden estar situados en memorias intermedias separadas. Tales técnicas se pueden aplicar a la transmisión de datos comprimidos que se comprimen utilizando esquemas de codificación de longitud variable o de codificación de longitud de racha.

De acuerdo con un tercer caso de uso, se pueden usar objetos de memoria inmediata conjuntamente con la señalización fuera de banda, por ejemplo, para llevar a cabo el intercambio de una cantidad relativamente grande de datos. En este caso, se puede usar la señalización fuera de banda para la sincronización mientras los objetos de memoria inmediata almacenan los datos. Por ejemplo, un productor de datos puede colocar datos en una memoria intermedia inmediata y notificar a un consumidor sobre la disposición y la ubicación y/o el tamaño de los datos

utilizando la señalización fuera de banda. En un escenario controlado por flujo, el consumidor lee los datos y notifica al productor que la memoria intermedia puede reutilizarse. La notificación también puede realizarse utilizando la señalización fuera de banda. Tales técnicas se pueden utilizar en los algoritmos que requieren transporte canalizado de datos controlado por flujo. Para una CPU anfitriona y una GPU, tales técnicas se pueden utilizar, por ejemplo, para el registro de diagnósticos. Para múltiples dispositivos informáticos de OpenCL, estas técnicas se pueden utilizar para conectar varios dispositivos en un canal de datos controlado por flujo asíncrono. Esto puede permitir a la aplicación dividirse en bloques que son más adecuados para cada CPU o GPU, poner en marcha diversas etapas de procesamiento canalizado en varios dispositivos y/o descargar la mayoría, o incluso la totalidad, de la sincronización de datos desde una CPU anfitriona.

En algunos ejemplos, las técnicas de esta divulgación pueden proporcionar una interfaz de paso de mensajes que facilita el envío y recepción de mensajes entre un proceso que se está ejecutando en un dispositivo anfitrión y una tarea que se está ejecutando en un dispositivo informático para una plataforma informática de múltiples procesadores que inicia tareas usando las colas de comandos. El dispositivo informático puede, en algunos casos, ser una GPU. En casos adicionales, el dispositivo informático puede ser cualquier tipo de dispositivo informático definido por una API de plataforma informática heterogénea, de múltiples plataformas y múltiples proveedores.

En ejemplos adicionales, las técnicas de esta divulgación pueden proporcionar una GPU que incluye uno o más registros que son accesibles por un dispositivo anfitrión. Los uno o más registros pueden estar configurados para facilitar el paso de mensajes entre una tarea que se está ejecutando en la GPU y un proceso que se está ejecutando en un dispositivo distinto a la GPU.

En ejemplos adicionales, las técnicas de esta divulgación pueden proporcionar una interfaz de memoria intermedia que permite la creación de los objetos de memoria inmediata. Los objetos de memoria inmediata se pueden usar para implementar un espacio de memoria compartida no almacenable en memoria caché y/o un espacio de memoria compartida coherente con memoria caché, con el fin de compartir datos entre un proceso que se está ejecutando en un dispositivo anfitrión y una tarea que se está ejecutando en un dispositivo informático, mientras la tarea se está ejecutando en el dispositivo informático. El dispositivo informático puede, en algunos casos, ser una unidad de procesamiento de gráficos (GPU). En otros casos, el dispositivo informático puede ser cualquier tipo de dispositivo informático definido por una API de plataforma informática heterogénea, de múltiples plataformas y múltiples proveedores.

En otros ejemplos, las técnicas de la presente divulgación pueden proporcionar una GPU que incluya una memoria caché para un espacio de memoria compartida que pueda inhabilitarse selectivamente para proporcionar un espacio de memoria compartida no almacenable en memoria caché. En ejemplos adicionales, las técnicas de la presente divulgación pueden proporcionar una GPU que incluya una modalidad de coherencia de memoria caché que pueda habilitarse selectivamente para proporcionar un espacio de memoria compartida coherente con memoria caché.

Las técnicas descritas en esta divulgación pueden implementarse, al menos en parte, en hardware, software, firmware o cualquier combinación de los mismos. Por ejemplo, varios aspectos de las técnicas descritas pueden implementarse dentro de uno o más procesadores, incluyendo uno o más microprocesadores, procesadores de señales digitales (DSP), circuitos integrados específicos de la aplicación (ASIC), formaciones de compuertas lógicas programables en el terreno (FPGA), o cualquier otro sistema de circuitos de lógica integrada o discreta equivalente, así como en cualquier combinación de tales componentes. El término "procesador" o "sistema de circuitos de procesamiento" puede referirse en general a cualquiera de los sistemas de circuitos lógicos anteriores, en solitario o en combinación con otros sistemas de circuitos lógicos, o cualquier otro sistema de circuitos equivalente, tal como hardware discreto que lleve a cabo el procesamiento.

Tales hardware, software y firmware puede implementarse dentro del mismo dispositivo o dentro de dispositivos independientes para prestar soporte a las diversas operaciones y funciones descritas en esta divulgación. Además, cualquiera de las unidades, módulos o componentes descritos se puede implementar en conjunto o por separado como dispositivos lógicos discretos pero interoperables. La representación de diferentes funciones como módulos o unidades está concebida para destacar diferentes aspectos funcionales y no implica necesariamente que tales módulos o unidades deban ser realizados por componentes de hardware o software independientes. Por el contrario, la funcionalidad asociada con uno o más módulos o unidades puede ser realizada por componentes de hardware, firmware y/o software independientes o integrados dentro de los componentes comunes o independientes de hardware o software.

Las técnicas descritas en la presente divulgación también se pueden almacenar, realizar o codificar en un medio legible por ordenador, tal como un medio de almacenamiento legible por ordenador que almacena instrucciones. Las instrucciones integradas o codificadas en un medio legible por ordenador pueden hacer que uno o más procesadores lleven a cabo las técnicas descritas en el presente documento, por ejemplo, cuando las instrucciones son ejecutadas mediante los uno o más procesadores. Los medios de almacenamiento legibles por ordenador pueden incluir memoria de acceso aleatorio (RAM), memoria de solo lectura (ROM), memoria de solo lectura programable (PROM), memoria de solo lectura programable y borrable (EPROM), memoria de solo lectura programable y borrable electrónicamente (EEPROM), memoria flash, un disco duro, un CD-ROM, un disquete, un

casete, medios magnéticos, medios ópticos u otros medios de almacenamiento legibles por ordenador que sean tangibles.

Entre los medios legibles por ordenador pueden incluirse medios de almacenamiento legibles por ordenador, los cuales corresponden a un medio de almacenamiento tangible, tal como los enumerados anteriormente. Los medios legibles por ordenador también pueden comprender medios de comunicación que incluyen cualquier medio que facilite la transferencia de un programa informático desde un lugar a otro, por ejemplo, de acuerdo con un protocolo de comunicación. De esta manera, la frase "medios legibles por ordenador" puede corresponder, en general, a (1) medios de almacenamiento legibles por ordenador tangibles que sean no transitorios, y (2) un medio de comunicación legible por ordenador no tangible tal como una señal transitoria u onda portadora.

REIVINDICACIONES

1. Un aparato que comprende:

medios para colocar (24) una pluralidad de comandos en una cola de comandos (32) en respuesta a la recepción de una o más instrucciones de puesta en cola desde un proceso (20) que se está ejecutando en un dispositivo anfitrión (12),
caracterizado porque la pluralidad de comandos incluye un primer comando que instruye al dispositivo anfitrión (12) para transferir datos entre un primer espacio de memoria asociado con el dispositivo anfitrión (12) y un segundo espacio de memoria asociado con una unidad de procesamiento de gráficos, GPU (14), incluyendo la pluralidad de comandos además un segundo comando que instruye al dispositivo anfitrión (12) para iniciar la ejecución de una tarea (28) en la GPU (14); y
porque el aparato comprende medios para pasar (30) uno o más mensajes entre el proceso que se está ejecutando en el dispositivo anfitrión (12) y una tarea (28) que se está ejecutando en la GPU (14) mientras la tarea (28) se está ejecutando en la GPU y en respuesta a la recepción de una o más instrucciones de paso de mensajes desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12).

2. El aparato de la reivindicación 1, que comprende además uno o más procesadores,

en el que los medios para colocar (24) la pluralidad de comandos en la cola de comandos (32) comprenden una interfaz de cola de comandos (24) que se está ejecutando en los uno o más procesadores, y configurada para colocar la pluralidad de comandos en la cola de comandos (32) en respuesta a la recepción de las una o más instrucciones de puesta en cola desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12), y
en el que el medio para pasar (30) uno o más mensajes comprende una interfaz de paso de mensajes (30) que se está ejecutando en los uno o más procesadores, y configurada para pasar los uno o más mensajes entre el proceso que se está ejecutando en el dispositivo anfitrión (12) y la tarea (28) que se está ejecutando en la GPU (14) mientras la tarea (28) se está ejecutando en la GPU (14) y en respuesta a la recepción de las una o más instrucciones de paso de mensajes desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12).

3. El aparato de la reivindicación 2,

en el que las una o más instrucciones de paso de mensajes comprenden una instrucción de envío que instruye a la interfaz de paso de mensajes para enviar un mensaje desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12) a la tarea (28) que se está ejecutando en la GPU (14), y
en el que la interfaz de paso de mensajes (30) está configurada además para enviar, en respuesta a la recepción de la instrucción de envío, el mensaje desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12) a la tarea (28) que se está ejecutando en la GPU (14), mientras la tarea (28) se está ejecutando en la GPU (14).

4. El aparato de la reivindicación 2,

en el que las una o más instrucciones de paso de mensajes comprenden una instrucción de registro de rutina de devolución de llamada, que instruye a la interfaz de paso de mensajes (30) para invocar una rutina de devolución de llamada en respuesta a la recepción de una señal desde la GPU (14) que indica que la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje, y
en el que la interfaz de paso de mensajes (30) está configurada además para iniciar la ejecución de la rutina de devolución de llamada especificada en la instrucción de registro de rutina de devolución de llamada, en respuesta a la recepción de la señal desde la GPU (14) que indica que la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje.

5. El aparato de la reivindicación 2,

en el que las una o más instrucciones de paso de mensajes comprenden una instrucción de sondeo que instruye a la interfaz de paso de mensajes para sondear la GPU (14), para obtener información de estado de mensaje, indicativa de si una tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje, y
en el que la interfaz de paso de mensajes (30) está además configurada para sondear la GPU (14), para obtener la información de estado de mensaje en respuesta a la recepción de la instrucción de sondeo y, si la información de estado de mensaje indica que la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje, obtener el mensaje desde la GPU (14).

6. El aparato de la reivindicación 2, en el que la interfaz de paso de mensajes (30) está configurada además para ejecutar las una o más instrucciones de paso de mensajes sin colocar ningún comando en la cola de comandos (32).

7. Un procedimiento que comprende:

la colocación (70, 72), con una interfaz de cola de comandos (24) que se está ejecutando en uno o más procesadores de un dispositivo anfitrión (12), de una pluralidad de comandos en una cola de comandos (32), en respuesta a la recepción de una o más instrucciones de puesta en cola desde un proceso (20) que se está ejecutando en el dispositivo anfitrión (12),
caracterizado porque la pluralidad de comandos incluye un primer comando que instruye al dispositivo principal (12) para transferir datos entre un primer espacio de memoria asociado con el dispositivo anfitrión (12) y un segundo espacio de memoria asociado con una unidad de procesamiento de gráficos, GPU (14), incluyendo la pluralidad de comandos además un segundo comando que instruye al dispositivo anfitrión (12) para iniciar la ejecución de una tarea (28) en la GPU (14); y
porque el procedimiento comprende el paso (76), con una interfaz de paso de mensajes (30) que se está ejecutando en los uno o más procesadores del dispositivo anfitrión (12), de uno o más mensajes entre el proceso (20) que se está ejecutando en el dispositivo anfitrión (12) y una tarea (28) que se está ejecutando en la GPU (14), mientras la tarea (28) se está ejecutando en la GPU (14) y en respuesta a la recepción de una o más instrucciones de paso de mensajes desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12).

8. El procedimiento de la reivindicación 7,

en el que las una o más instrucciones de paso de mensajes comprenden una instrucción de envío que instruye a la interfaz de paso de mensajes (30) para enviar un mensaje desde el proceso (20) que se está ejecutando en el dispositivo anfitrión (12) a la tarea (28) que se está ejecutando en la GPU (14), y
en el que el procedimiento comprende además el envío, con la interfaz de paso de mensajes, del mensaje desde el proceso que se está ejecutando en el dispositivo anfitrión (12) a la tarea (28) que se está ejecutando en la GPU (14), mientras la tarea (28) se está ejecutando en la GPU (14) y en respuesta a la recepción de la instrucción de envío.

9. El procedimiento de la reivindicación 7,

en el que las una o más instrucciones de paso de mensajes comprenden una instrucción de registro de rutina de devolución de llamada que instruye a la interfaz de paso de mensajes (30) para invocar una rutina de devolución de llamada en respuesta a la recepción de una señal desde la GPU (14) que indica que la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje, y
en el que el procedimiento comprende además iniciar la ejecución de la rutina de devolución de llamada especificada en la instrucción de registro de rutina de devolución de llamada, en respuesta a la recepción de la señal desde la GPU (14), indicando que la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje.

10. El procedimiento de la reivindicación 7,

en el que las una o más instrucciones de paso de mensajes comprenden una instrucción de sondeo que instruye a la interfaz de paso de mensajes (30) para sondear la GPU (14) para obtener información de estado de mensaje, indicativa de si la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje, y
en el que el procedimiento comprende además:
el sondeo, con la interfaz de paso de mensajes (30), de la GPU (14) para obtener la información de estado de mensaje en respuesta a la recepción de la instrucción de sondeo; y
si la información de estado de mensaje indica que la tarea (28) que se está ejecutando en la GPU (14) ha enviado un mensaje, la obtención del mensaje desde la GPU (14).

11. El procedimiento de la reivindicación 7, que comprende además:

la ejecución, con la interfaz de paso de mensajes (30), de las una o más instrucciones de paso de mensajes sin colocar ningún comando en cola de comandos (32).

12. Un medio legible por ordenador que comprende instrucciones que hacen que uno o más procesadores realicen el procedimiento de cualquiera de las reivindicaciones 7 a 11.

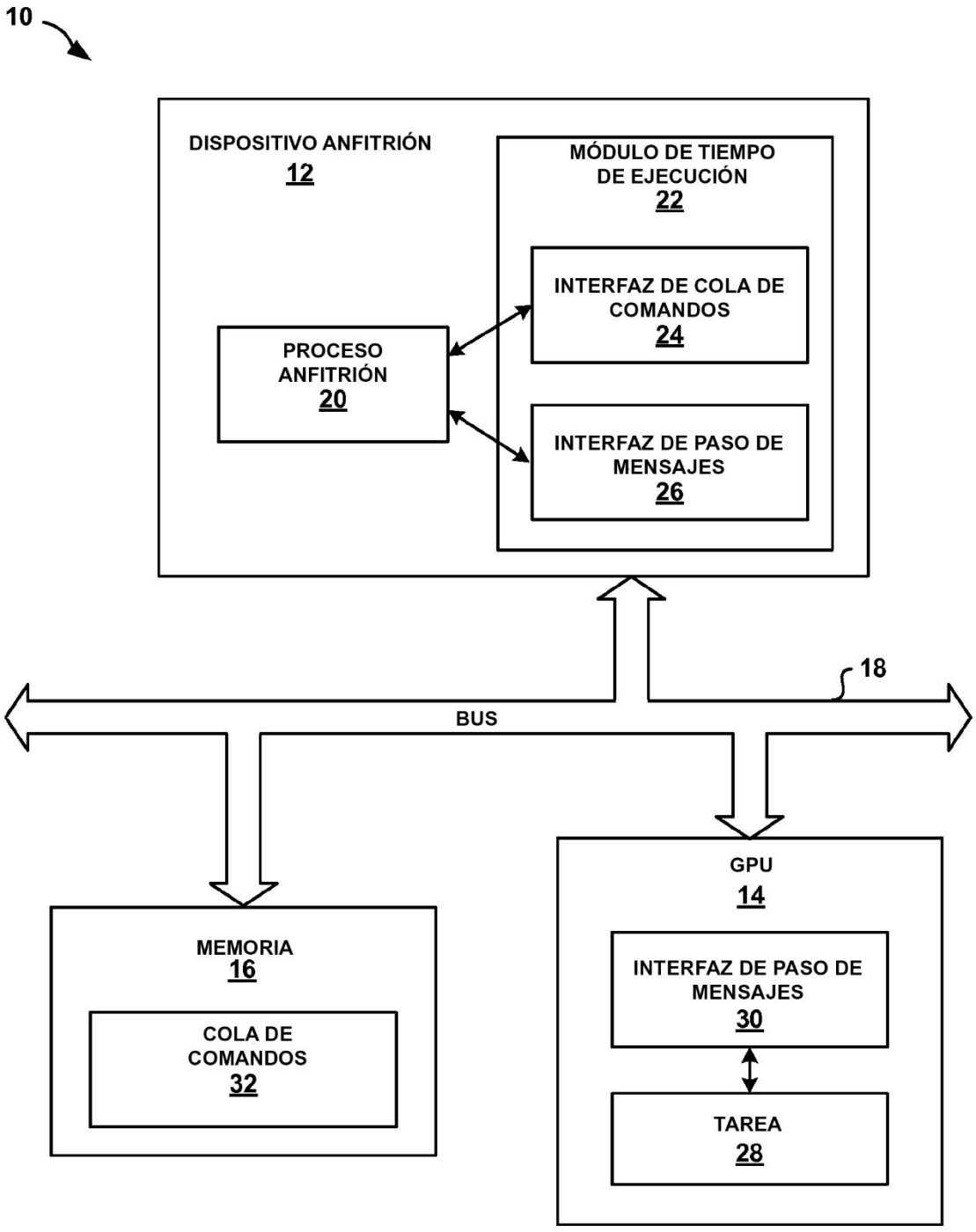
10
DISPOSITIVO ANFITRIÓN
12
MÓDULO DE TIEMPO DE EJECUCIÓN
22
INTERFAZ DE COLA DE COMANDOS
24
PROCESO ANFITRIÓN
20
INTERFAZ DE PASO DE MENSAJES
26
18
BUS
GPU
14
MEMORIA
16
INTERFAZ DE PASO DE MENSAJES
30
COLA DE COMANDOS
32
TAREA
28

FIG. 1

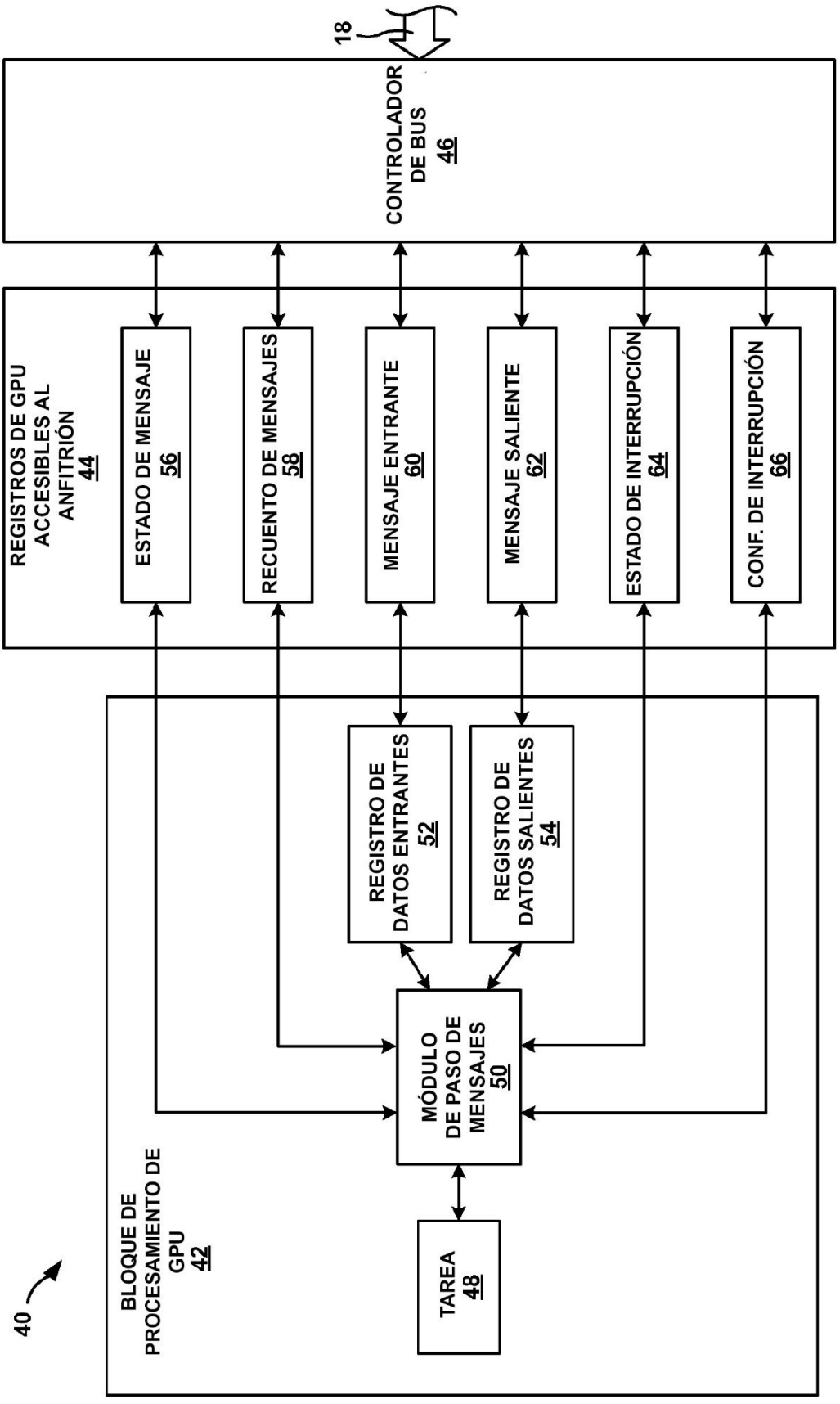
18
CONTROLADOR DE BUS 46
REGISTROS DE GPU ACCESIBLES AL ANFITRIÓN 44
ESTADO DE MENSAJE 56
RECUENTO DE MENSAJES 58
MENSAJE ENTRANTE 60
MENSAJE SALIENTE 62
ESTADO DE INTERRUPCIÓN 64
CONF. DE INTERRUPCIÓN 66
REGISTRO DE DATOS ENTRANTES 52
REGISTRO DE DATOS SALIENTES 54
MÓDULO DE PASO DE MENSAJES 50
TAREA 48
BLOQUE DE PROCESAMIENTO DE GPU 42
40

FIG. 2

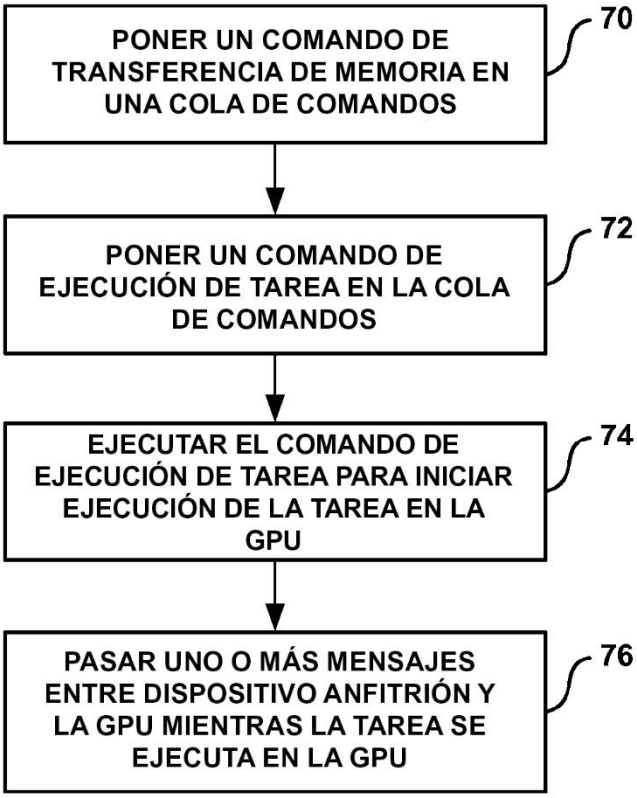
PONER UN COMANDO DE TRANSFERENCIA DE MEMORIA EN UNA COLA DE COMANDOS
70
PONER UN COMANDO DE EJECUCIÓN DE TAREA EN LA COLA DE COMANDOS
72
EJECUTAR EL COMANDO DE EJECUCIÓN DE TAREA PARA INICIAR EJECUCIÓN DE LA TAREA EN LA GPU
74
PASAR UNO O MÁS MENSAJES ENTRE DISPOSITIVO ANFITRIÓN Y LA GPU MIENTRAS LA TAREA SE EJECUTA EN LA GPU
76

FIG. 3

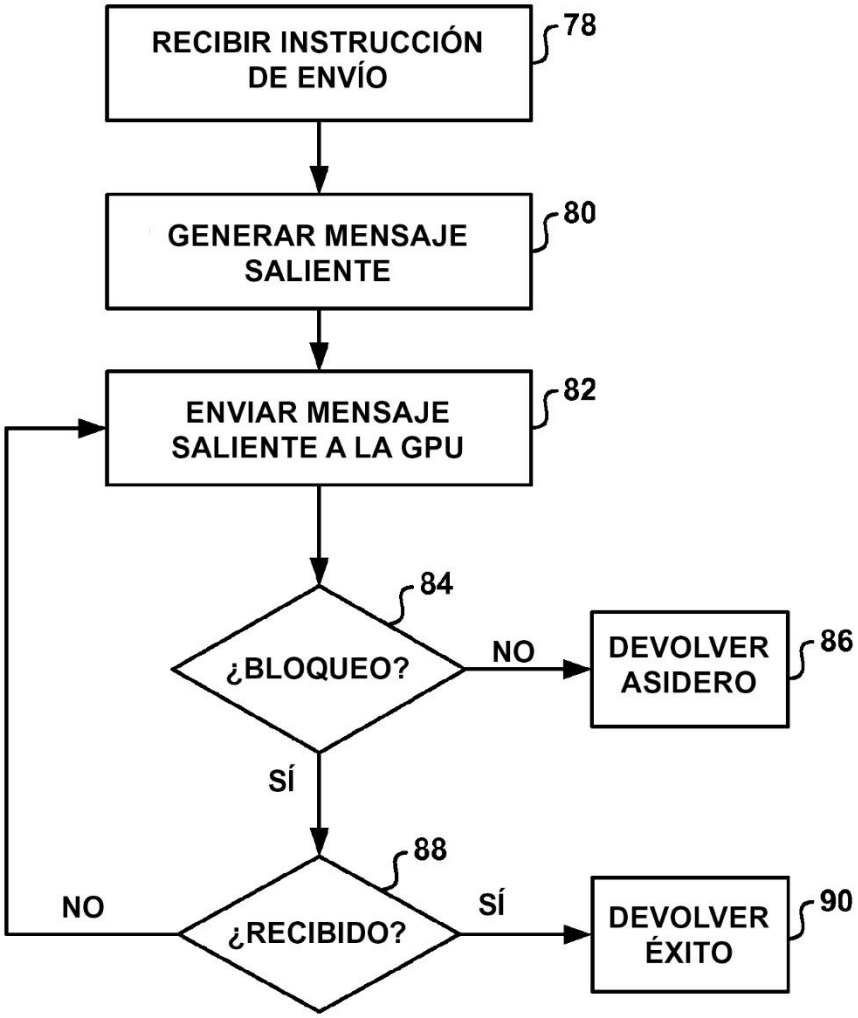
RECIBIR INSTRUCCIÓN DE ENVÍO
78
GENERAR MENSAJE SALIENTE
80
ENVIAR MENSAJE SALIENTE A LA GPU
82
84
¿BLOQUEO?
NO
DEVOLVER ASIDERO
86
SÍ
88
NO
¿RECIBIDO?
SÍ
DEVOLVER ÉXITO
90

FIG. 4

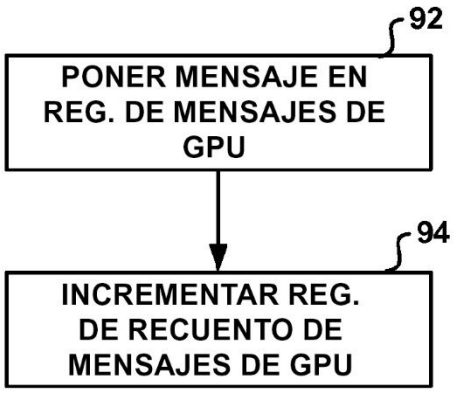
92
PONER MENSAJE EN REG. DE MENSAJES DE GPU
94
INCREMENTAR REG. DE RECUENTO DE MENSAJES DE GPU

FIG. 5

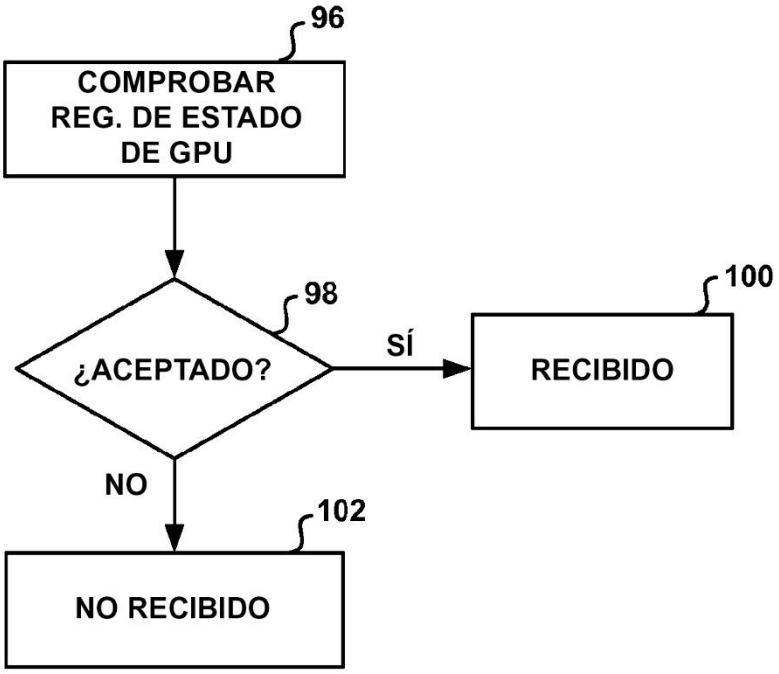
96
COMPROBAR REG. DE ESTADO DE GPU
98
¿ACEPTADO?
SÍ
100
RECIBIDO
NO
102
NO RECIBIDO

FIG. 6

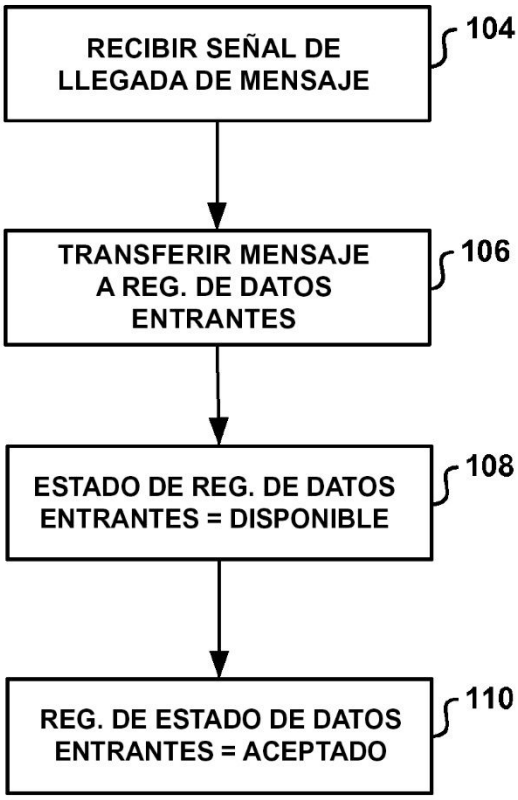
RECIBIR SEÑAL DE LLEGADA DE MENSAJE
104
TRANSFERIR MENSAJE A REG. DE DATOS ENTRANTES
106
ESTADO DE REG. DE DATOS ENTRANTES = DISPONIBLE
108
REG. DE ESTADO DE DATOS ENTRANTES = ACEPTADO
110

FIG. 7

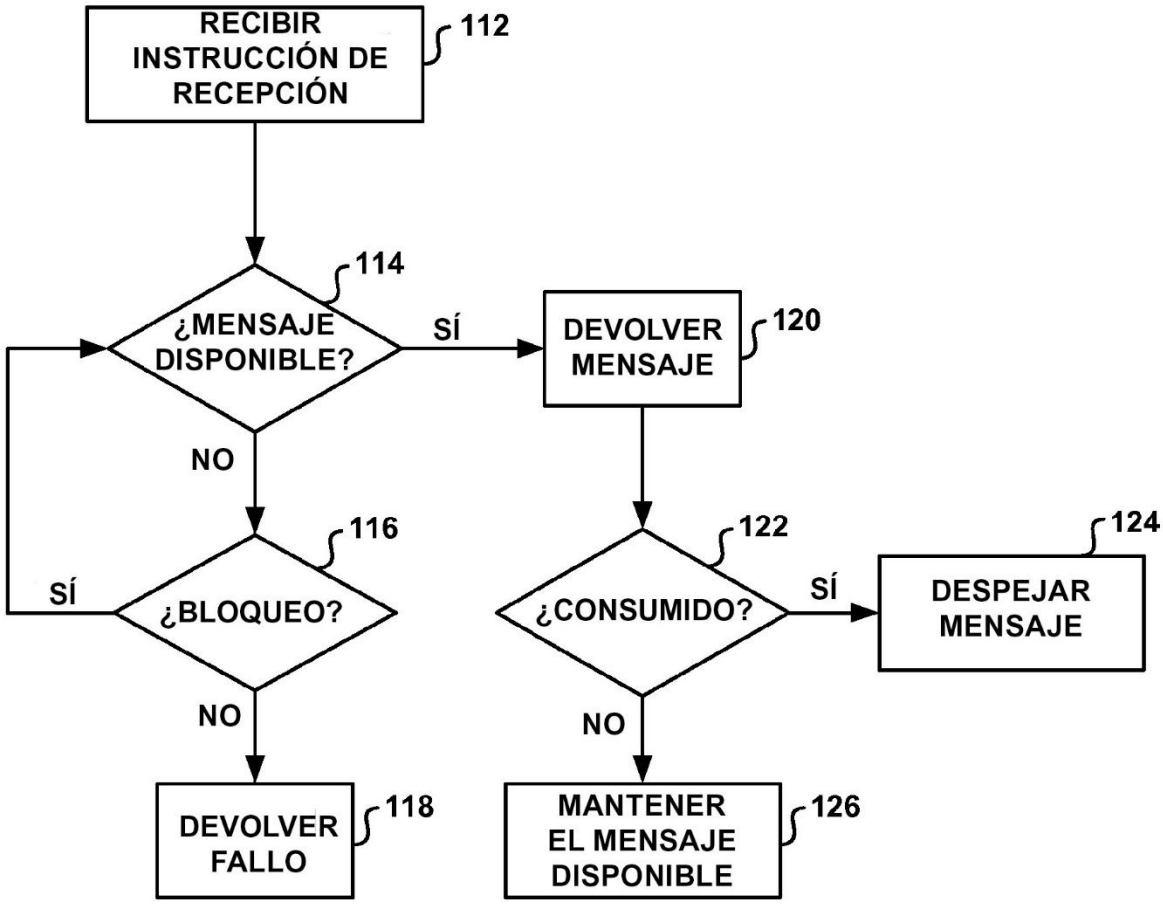
RECIBIR INSTRUCCIÓN DE RECEPCIÓN
112
¿MENSAJE DISPONIBLE?
114
SÍ
DEVOLVER MENSAJE
120
NO
¿BLOQUEO?
116
SÍ
NO
DEVOLVER FALLO
118
¿CONSUMIDO?
122
SÍ
DESPEJAR MENSAJE
124
NO
MANTENER EL MENSAJE DISPONIBLE
126

FIG. 8

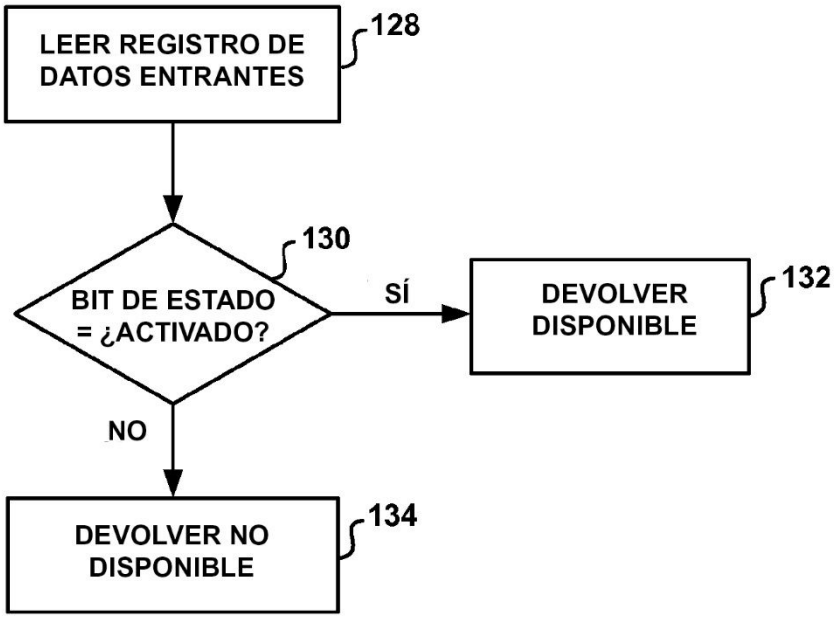
LEER REGISTRO DE DATOS ENTRANTES
128
BIT DE ESTADO = ¿ACTIVADO?
130
SÍ
DEVOLVER DISPONIBLE
132
NO
DEVOLVER NO DISPONIBLE
134

FIG. 9

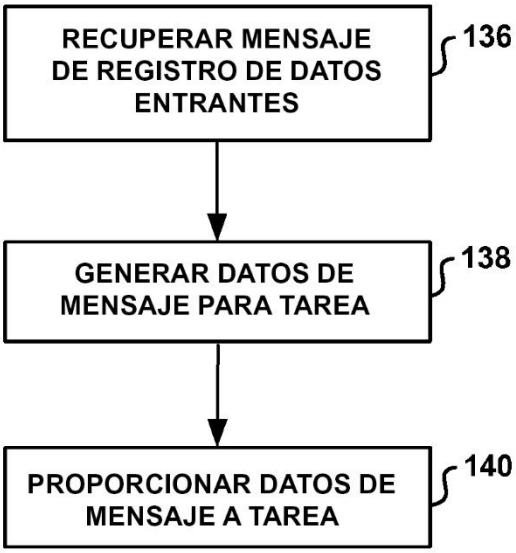
RECUPERAR MENSAJE DE REGISTRO DE DATOS ENTRANTES
136
GENERAR DATOS DE MENSAJE PARA TAREA
138
PROPORCIONAR DATOS DE MENSAJE A TAREA
140

FIG. 10

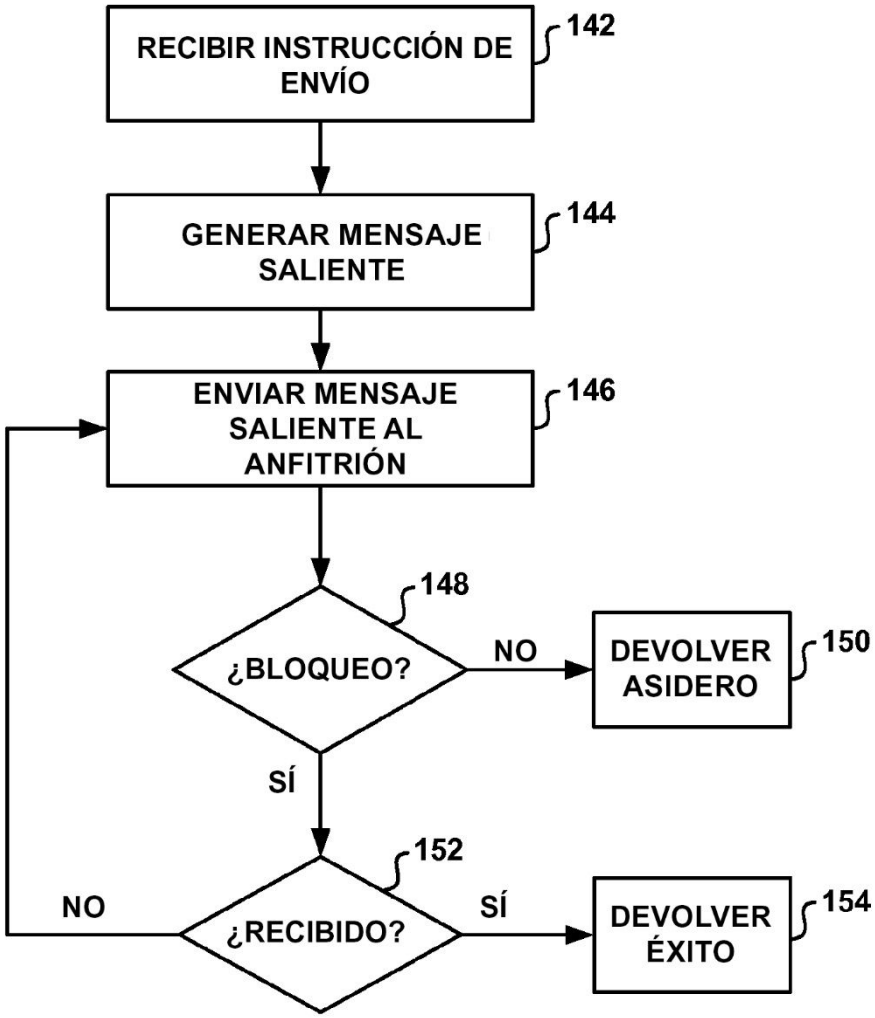
RECIBIR INSTRUCCIÓN DE ENVÍO
142
GENERAR MENSAJE SALIENTE
144
ENVIAR MENSAJE SALIENTE AL ANFITRIÓN
146
148
¿BLOQUEO?
NO
DEVOLVER ASIDERO
150
SÍ
152
NO
¿RECIBIDO?
SÍ
DEVOLVER ÉXITO
154

FIG. 11

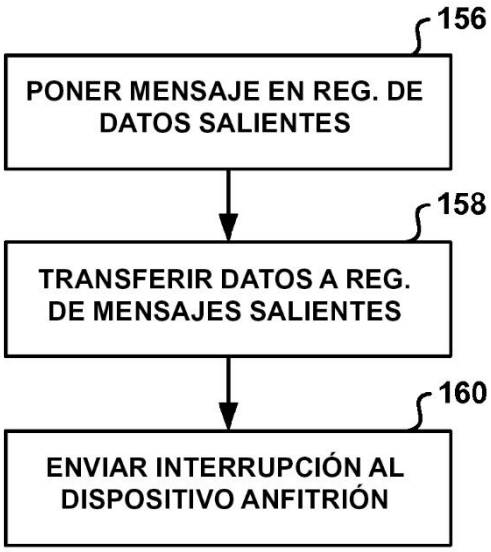
156
PONER MENSAJE EN REG. DE DATOS SALIENTES
158
TRANSFERIR DATOS A REG. DE MENSAJES SALIENTES
160
ENVIAR INTERRUPCIÓN AL DISPOSITIVO ANFITRIÓN

FIG. 12

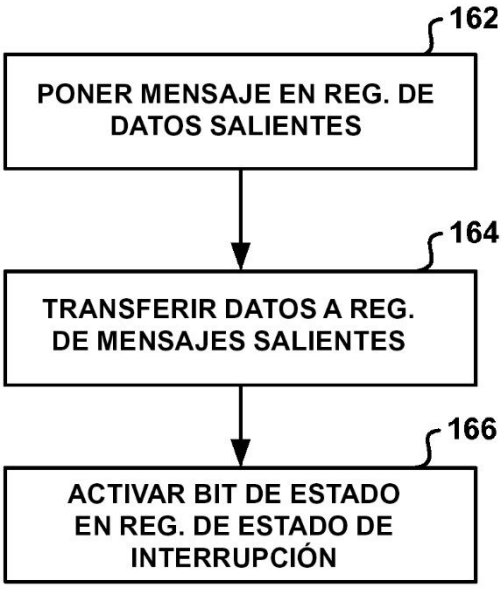
162
PONER MENSAJE EN REG. DE DATOS SALIENTES
164
TRANSFERIR DATOS A REG. DE MENSAJES SALIENTES
166
ACTIVAR BIT DE ESTADO EN REG. DE ESTADO DE INTERRUPCIÓN

FIG. 13

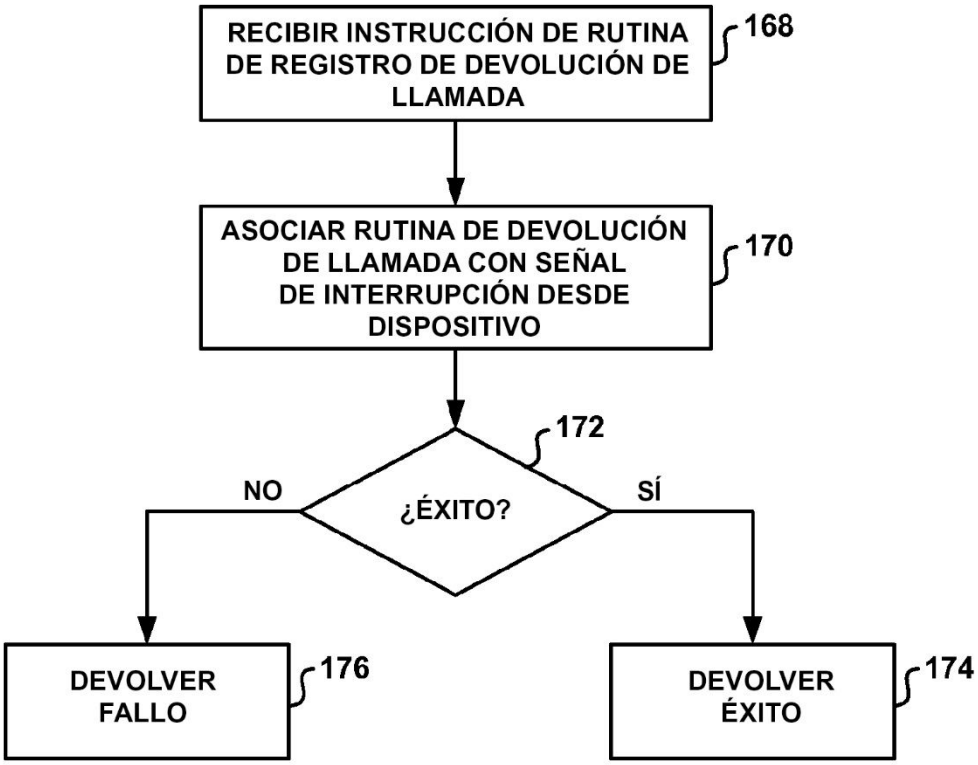
RECIBIR INSTRUCCIÓN DE RUTINA DE REGISTRO DE DEVOLUCIÓN DE LLAMADA
168
ASOCIAR RUTINA DE DEVOLUCIÓN DE LLAMADA CON SEÑAL DE INTERRUPCIÓN DESDE DISPOSITIVO
170
172
¿ÉXITO?
NO
SÍ
DEVOLVER FALLO
176
DEVOLVER ÉXITO
174

FIG. 14

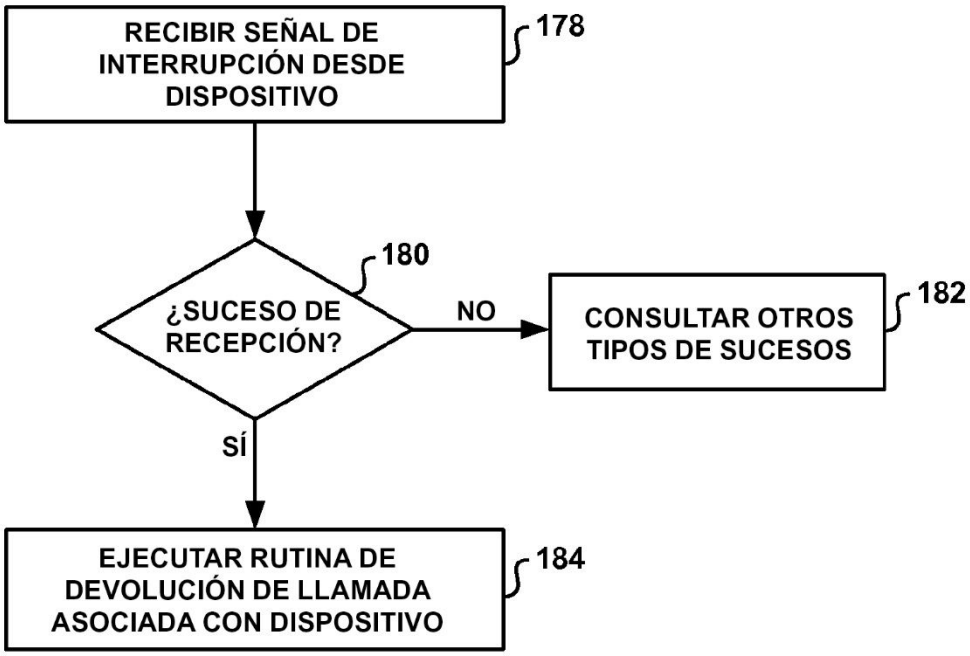
RECIBIR SEÑAL DE INTERRUPCIÓN DESDE DISPOSITIVO
178
¿SUCESO DE RECEPCIÓN?
180
NO
CONSULTAR OTROS TIPOS DE SUCESOS
182
SÍ
EJECUTAR RUTINA DE DEVOLUCIÓN DE LLAMADA ASOCIADA CON DISPOSITIVO
184

FIG. 15

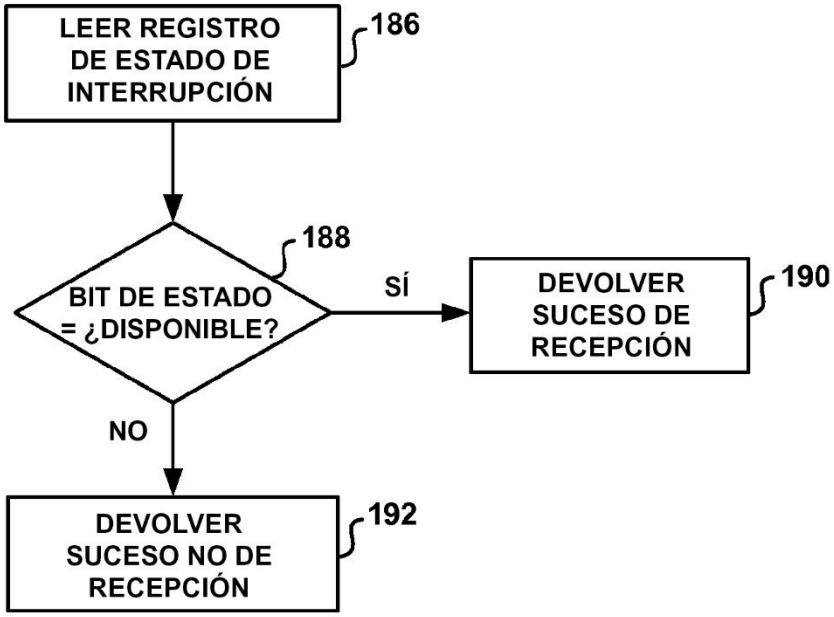
LEER REGISTRO DE ESTADO DE INTERRUPCIÓN
186
BIT DE ESTADO = ¿DISPONIBLE?
188
SÍ
DEVOLVER SUCESO DE RECEPCIÓN
190
NO
DEVOLVER SUCESO NO DE RECEPCIÓN
192

FIG. 16

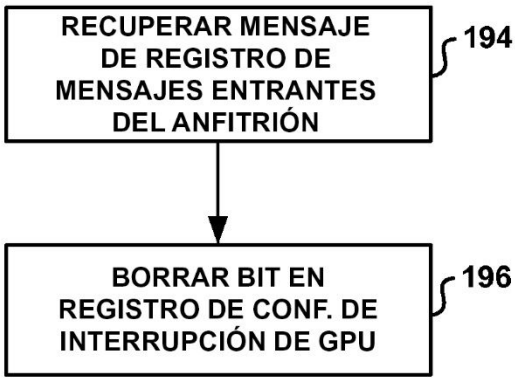
RECUPERAR MENSAJE DE REGISTRO DE MENSAJES ENTRANTES DEL ANFITRIÓN
194
BORRAR BIT EN REGISTRO DE CONF. DE INTERRUPCIÓN DE GPU
196

FIG. 17

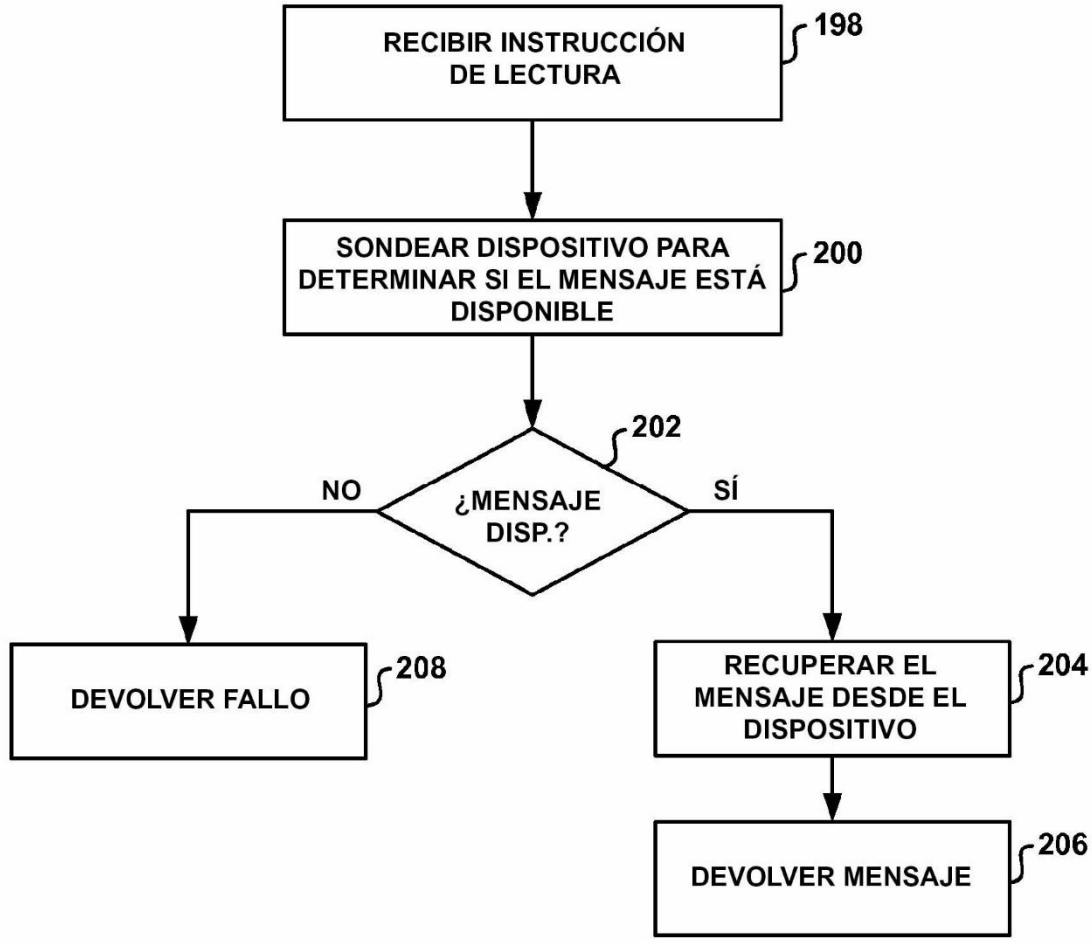
RECIBIR INSTRUCCIÓN DE LECTURA
198
SONDEAR DISPOSITIVO PARA DETERMINAR SI EL MENSAJE ESTÁ DISPONIBLE
200
202
¿MENSAJE DISP.?
NO
SÍ
DEVOLVER FALLO
208
RECUPERAR EL MENSAJE DESDE EL DISPOSITIVO
204
DEVOLVER MENSAJE
206

FIG. 18

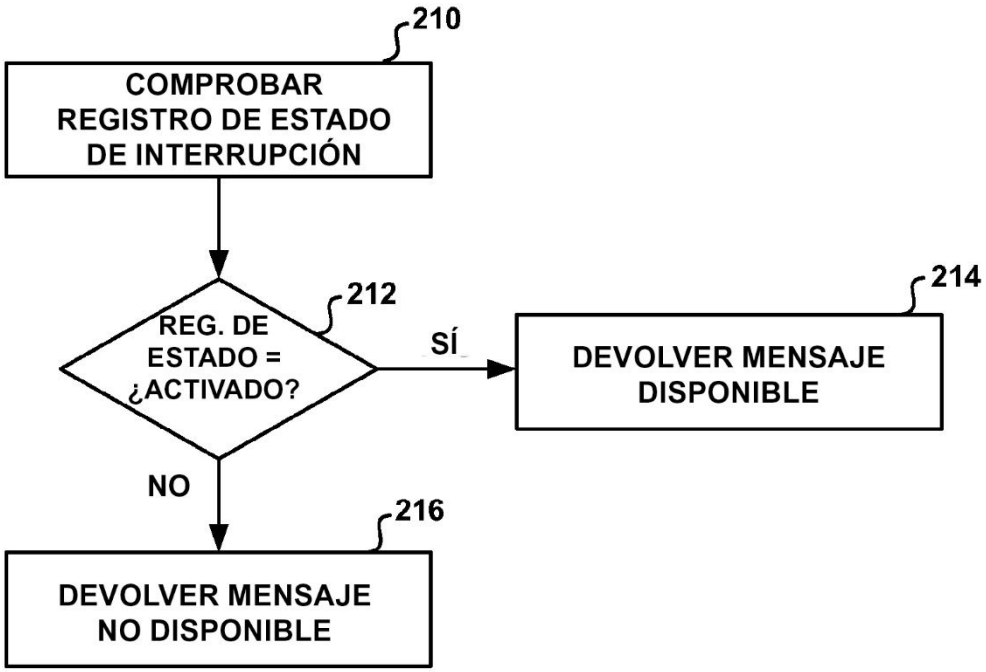
210
COMPROBAR REGISTRO DE ESTADO DE INTERRUPCIÓN
212
REG. DE ESTADO = ¿ACTIVADO?
SÍ
214
DEVOLVER MENSAJE DISPONIBLE
NO
216
DEVOLVER MENSAJE NO DISPONIBLE

FIG. 19

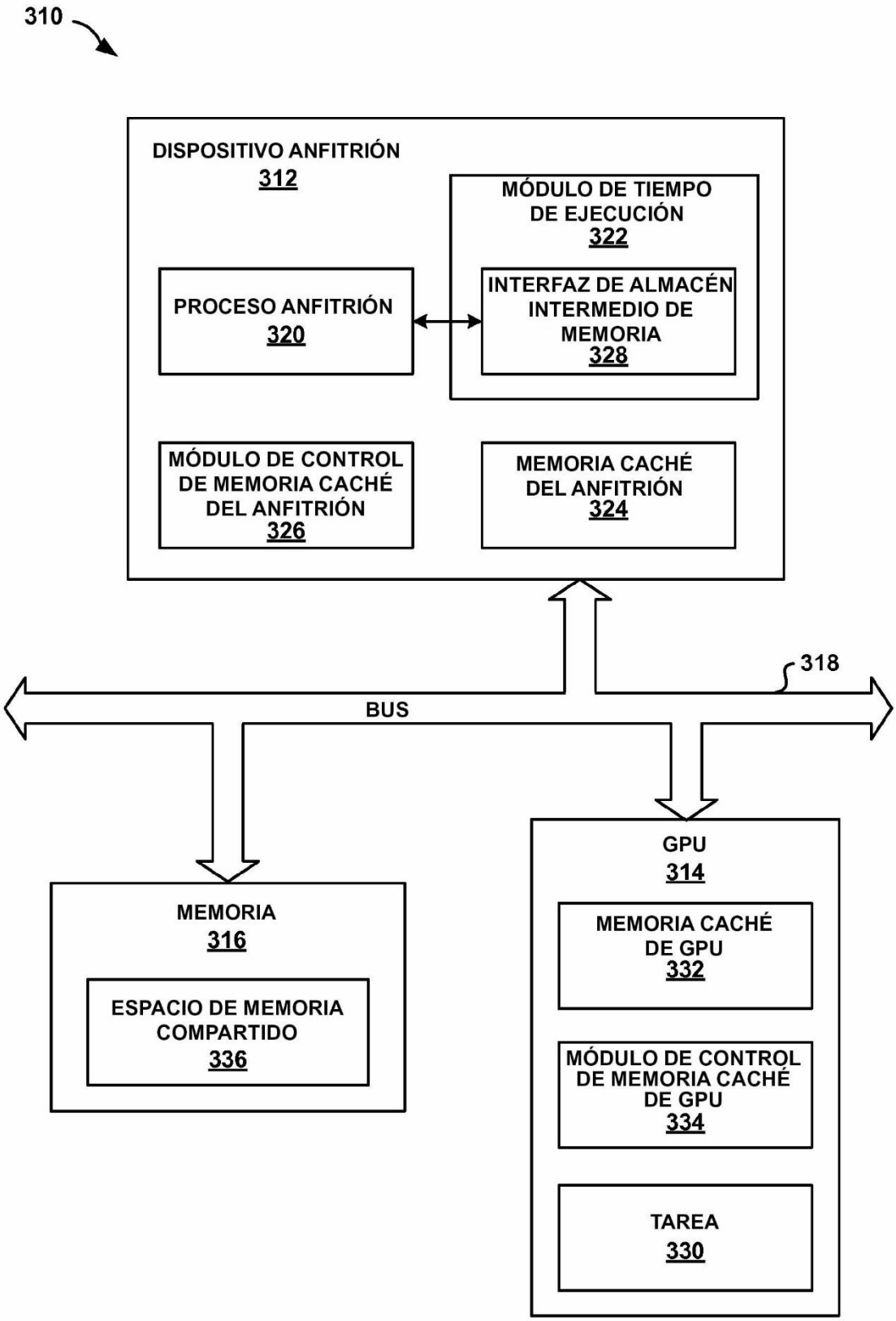
310
DISPOSITIVO ANFITRIÓN
312
MÓDULO DE TIEMPO DE EJECUCIÓN
322
PROCESO ANFITRIÓN
320
INTERFAZ DE ALMACÉN INTERMEDIO DE MEMORIA
328
MÓDULO DE CONTROL DE MEMORIA CACHÉ DEL ANFITRIÓN
326
MEMORIA CACHÉ DEL ANFITRIÓN
324
318
BUS
MEMORIA
316
ESPACIO DE MEMORIA COMPARTIDO
336
GPU
314
MEMORIA CACHÉ DE GPU
332
MÓDULO DE CONTROL DE MEMORIA CACHÉ DE GPU
334
TAREA
330

FIG. 20

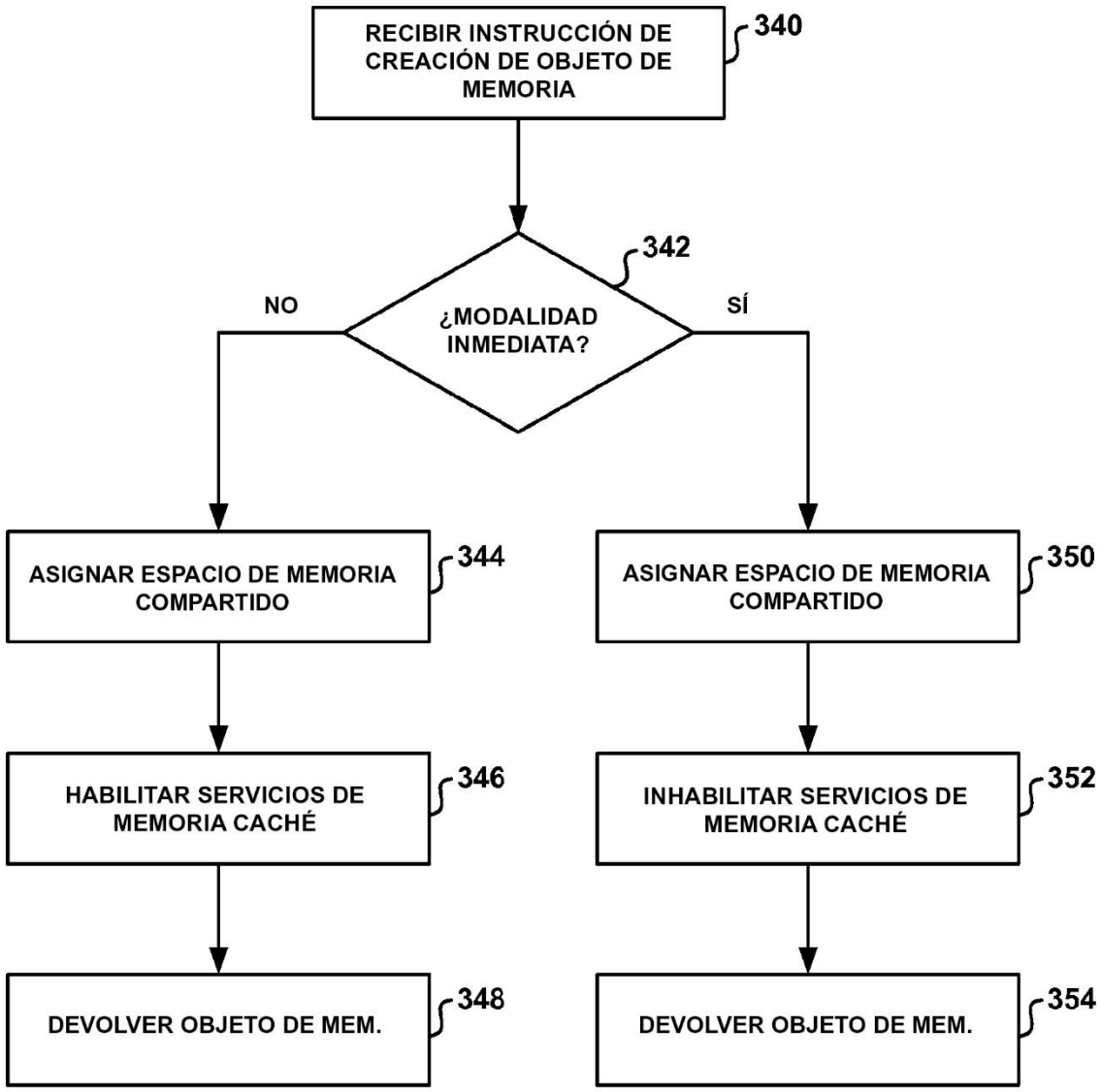
RECIBIR INSTRUCCIÓN DE CREACIÓN DE OBJETO DE MEMORIA
340
342
¿MODALIDAD INMEDIATA?
NO
SÍ
ASIGNAR ESPACIO DE MEMORIA COMPARTIDO
344
HABILITAR SERVICIOS DE MEMORIA CACHÉ
346
DEVOLVER OBJETO DE MEM.
348
ASIGNAR ESPACIO DE MEMORIA COMPARTIDO
350
INHABILITAR SERVICIOS DE MEMORIA CACHÉ
352
DEVOLVER OBJETO DE MEM.
354

FIG. 21

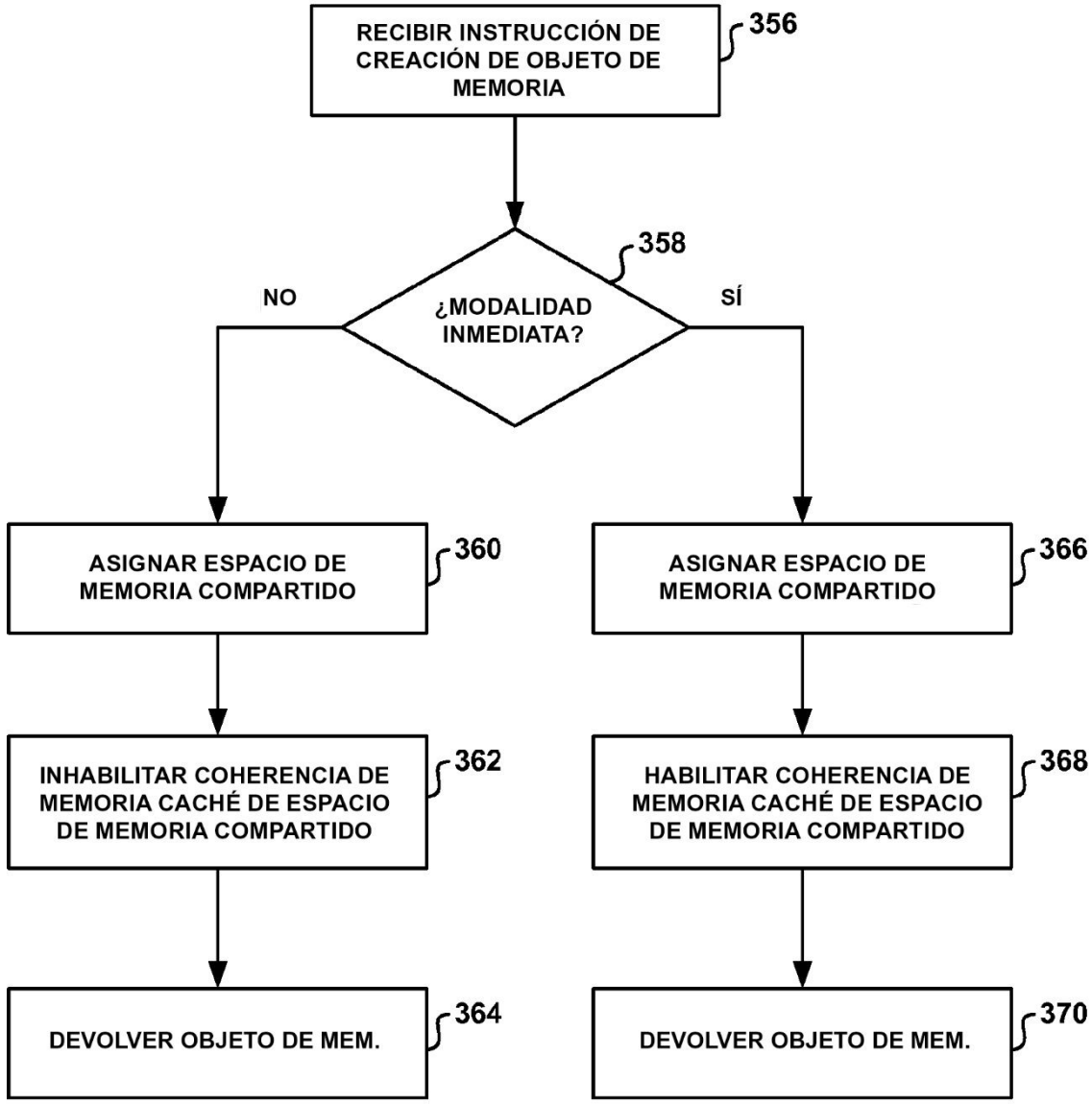
RECIBIR INSTRUCCIÓN DE CREACIÓN DE OBJETO DE MEMORIA
356
¿MODALIDAD INMEDIATA?
358
NO
SÍ
ASIGNAR ESPACIO DE MEMORIA COMPARTIDO
360
INHABILITAR COHERENCIA DE MEMORIA CACHÉ DE ESPACIO DE MEMORIA COMPARTIDO
362
DEVOLVER OBJETO DE MEM.
364
ASIGNAR ESPACIO DE MEMORIA COMPARTIDO
366
HABILITAR COHERENCIA DE MEMORIA CACHÉ DE ESPACIO DE MEMORIA COMPARTIDO
368
DEVOLVER OBJETO DE MEM.
370

FIG. 22

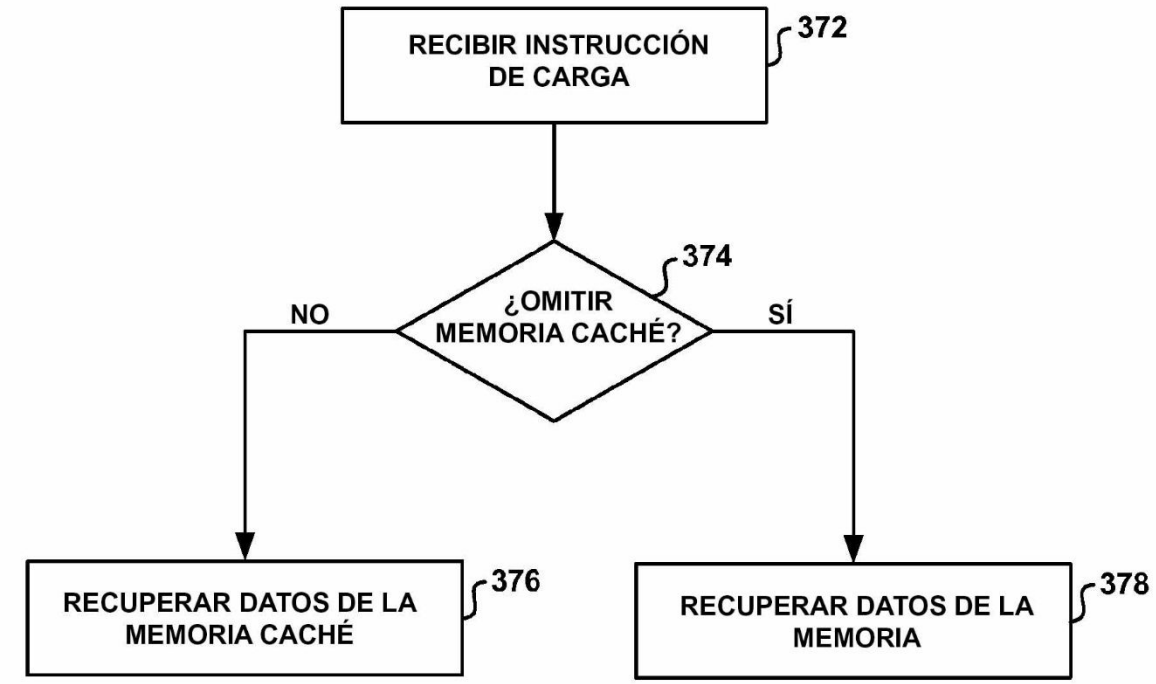
RECIBIR INSTRUCCIÓN DE CARGA
372
¿OMITIR MEMORIA CACHÉ?
374
NO
SÍ
RECUPERAR DATOS DE LA MEMORIA CACHÉ
376
RECUPERAR DATOS DE LA MEMORIA
378

FIG. 23

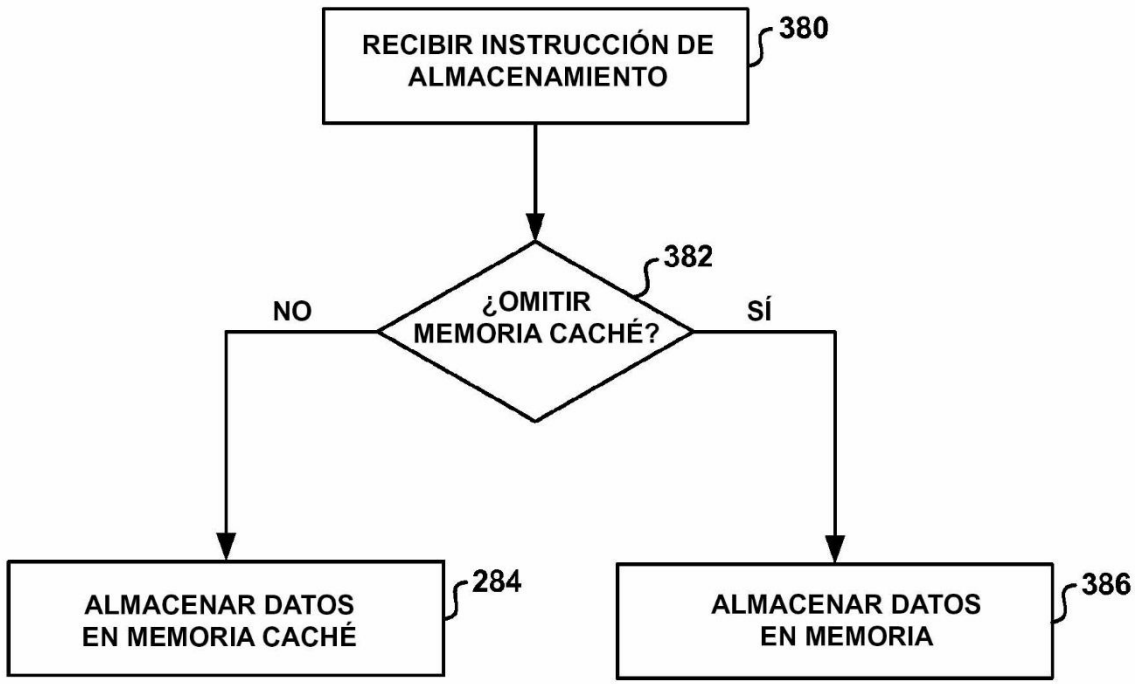
RECIBIR INSTRUCCIÓN DE ALMACENAMIENTO
380
¿OMITIR MEMORIA CACHÉ?
382
NO
SÍ
ALMACENAR DATOS EN MEMORIA CACHÉ
284
ALMACENAR DATOS EN MEMORIA
386

FIG. 24

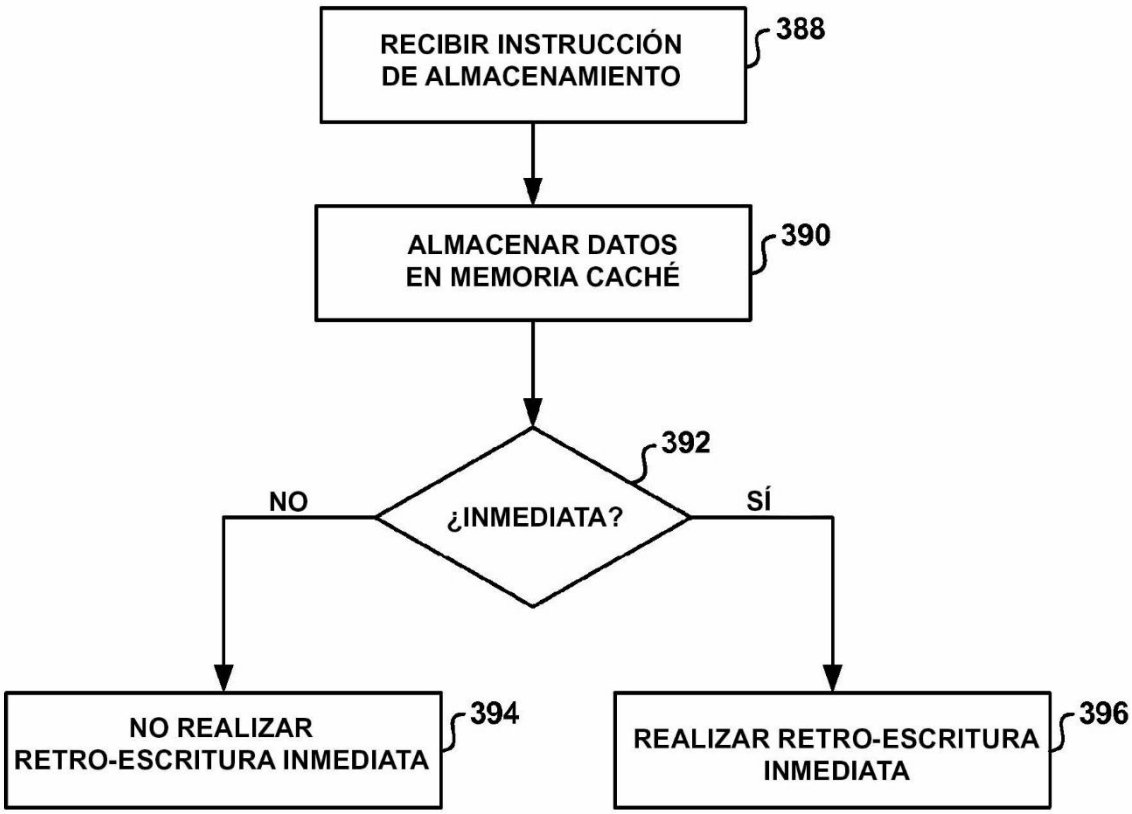
RECIBIR INSTRUCCIÓN DE ALMACENAMIENTO
388
ALMACENAR DATOS EN MEMORIA CACHÉ
390
¿INMEDIATA?
392
NO
SÍ
NO REALIZAR RETRO-ESCRITURA INMEDIATA
394
REALIZAR RETRO-ESCRITURA INMEDIATA
396

FIG. 25

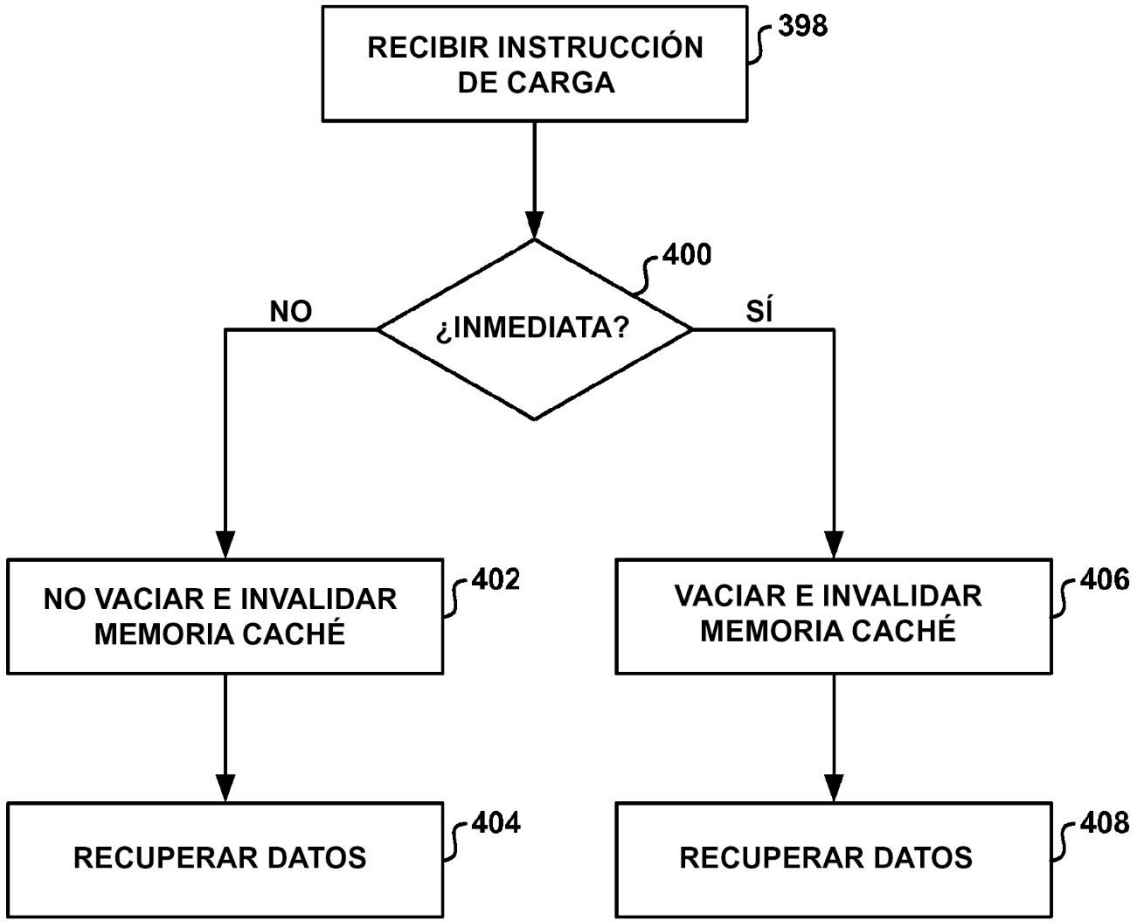
RECIBIR INSTRUCCIÓN DE CARGA
398
400
¿INMEDIATA?
NO
SÍ
NO VACIAR E INVALIDAR MEMORIA CACHÉ
402
VACIAR E INVALIDAR MEMORIA CACHÉ
406
RECUPERAR DATOS
404
RECUPERAR DATOS
408

FIG. 26

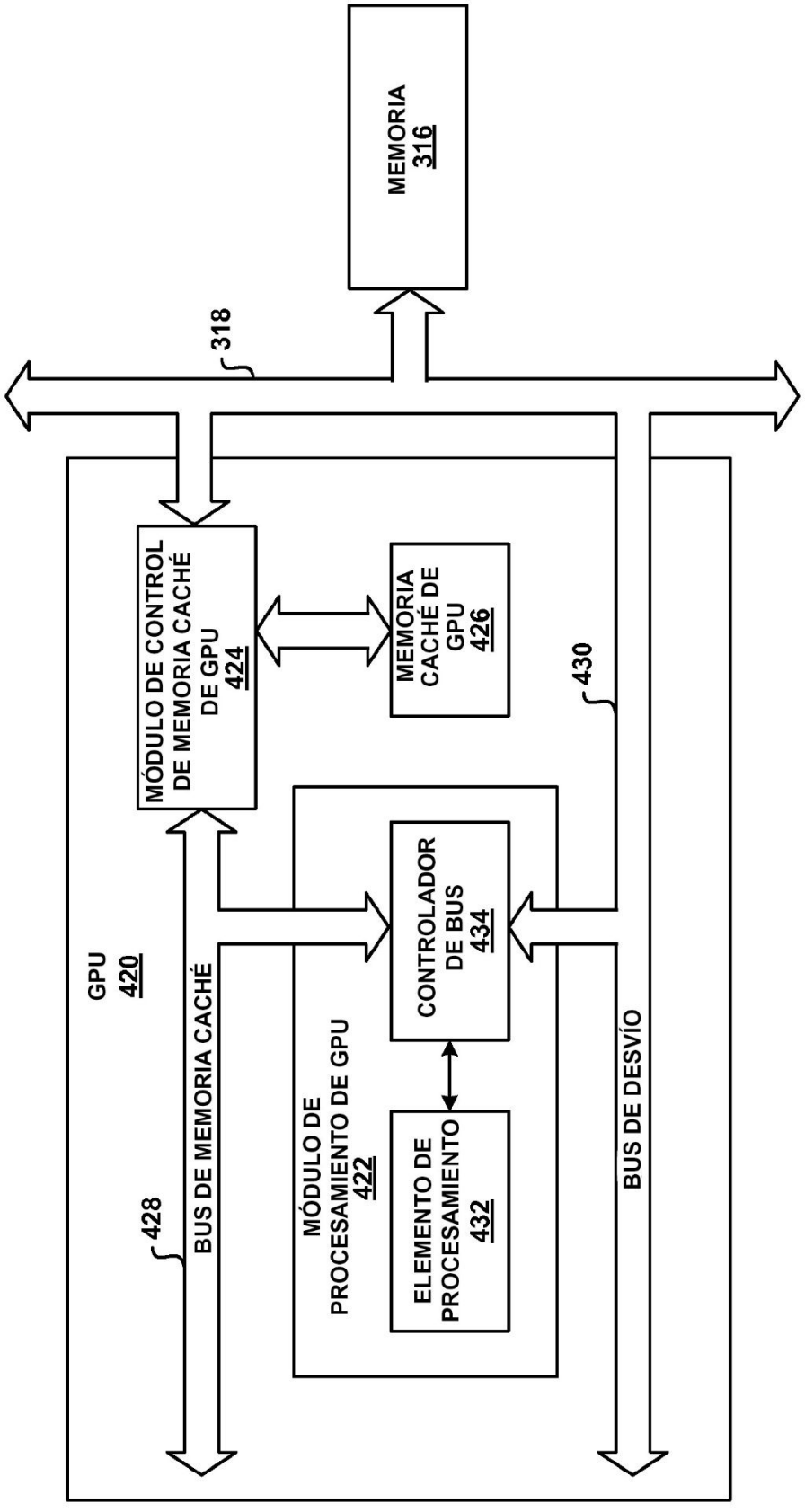
MEMORIA
316
318
MÓDULO DE CONTROL DE MEMORIA CACHÉ DE GPU
424
MEMORIA CACHÉ DE GPU
426
430
GPU
420
CONTROLADOR DE BUS
434
MÓDULO DE PROCESAMIENTO DE GPU
422
ELEMENTO DE PROCESAMIENTO
432
428
BUS DE MEMORIA CACHÉ
BUS DE DESVÍO

FIG. 27

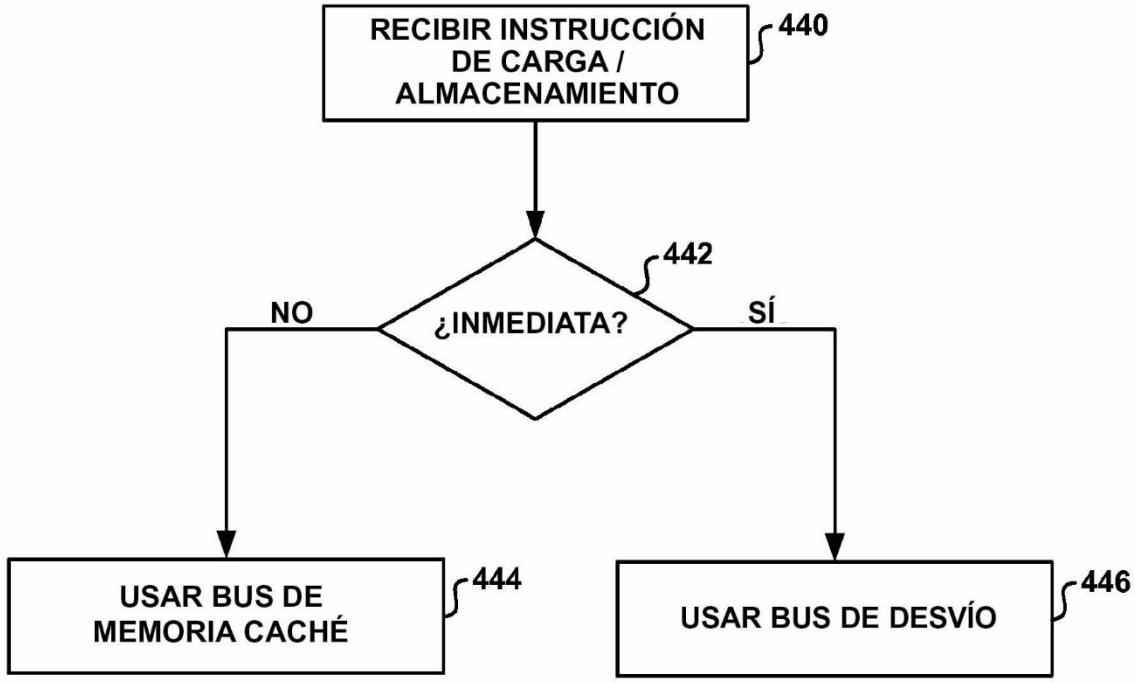
RECIBIR INSTRUCCIÓN DE CARGA / ALMACENAMIENTO
440
442
¿INMEDIATA?
NO
SÍ
USAR BUS DE MEMORIA CACHÉ
444
USAR BUS DE DESVÍO
446

FIG. 28

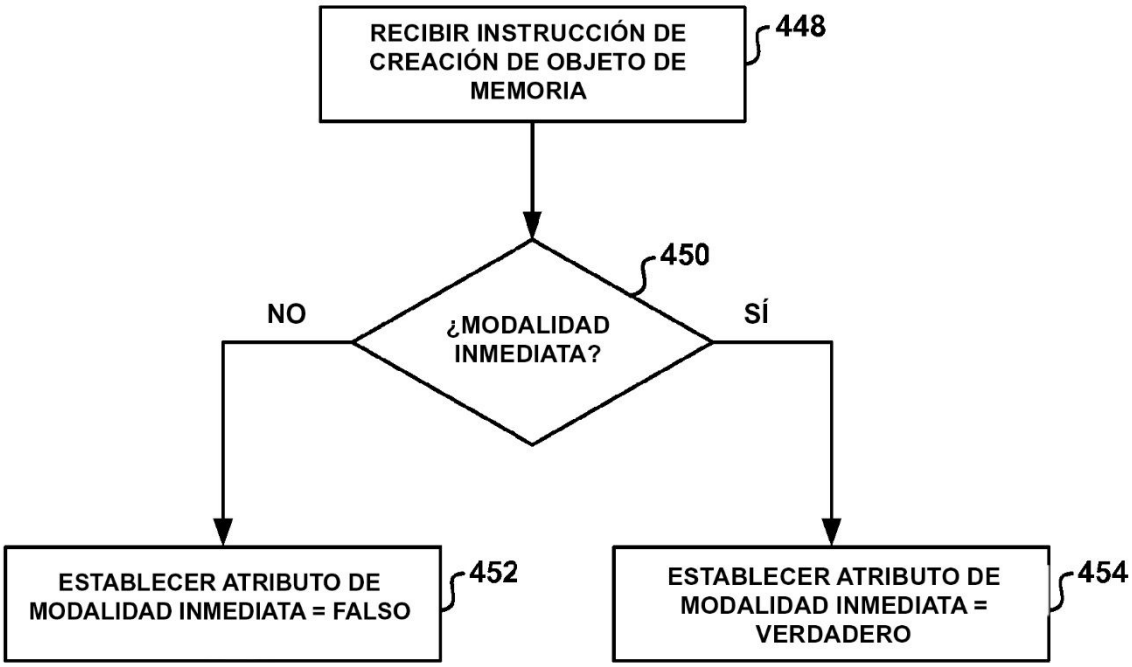
RECIBIR INSTRUCCIÓN DE CREACIÓN DE OBJETO DE MEMORIA
448
¿MODALIDAD INMEDIATA?
450
NO
SÍ
ESTABLECER ATRIBUTO DE MODALIDAD INMEDIATA = FALSO
452
ESTABLECER ATRIBUTO DE MODALIDAD INMEDIATA = VERDADERO
454

FIG. 29

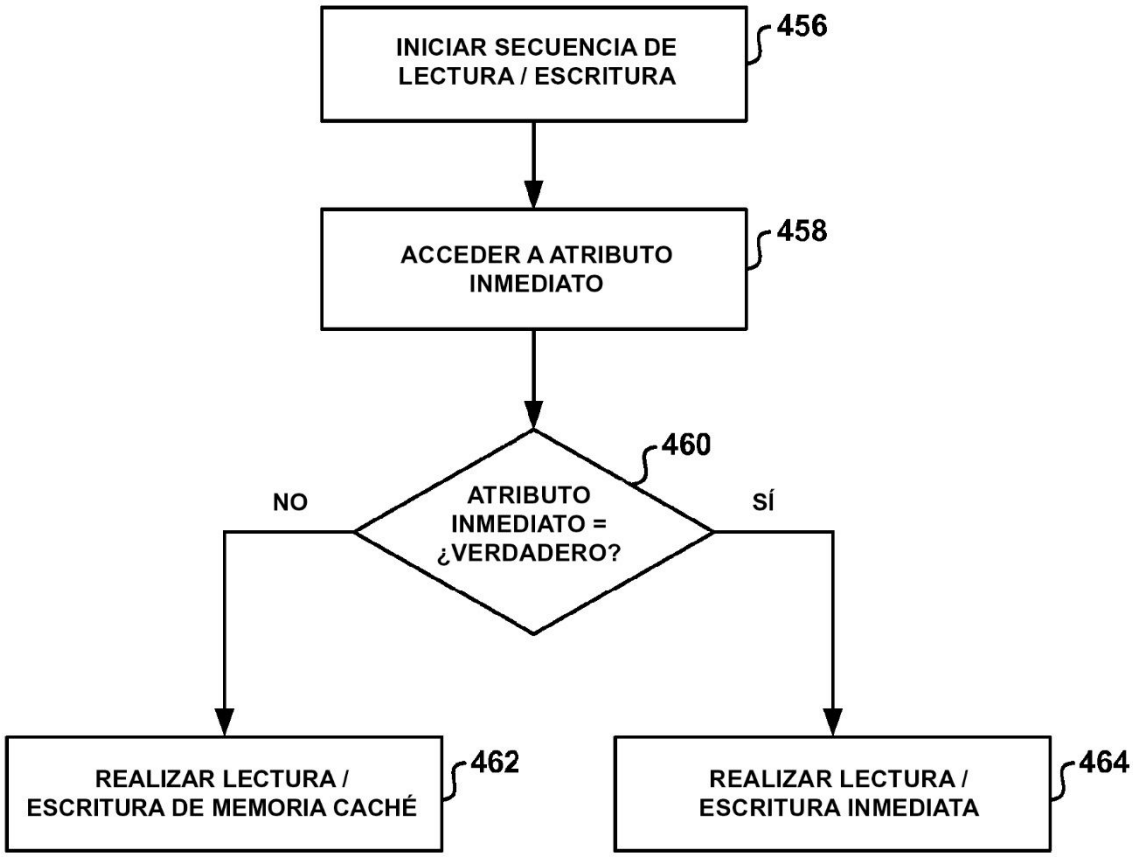
INICIAR SECUENCIA DE LECTURA / ESCRITURA
456
ACCEDER A ATRIBUTO INMEDIATO
458
ATRIBUTO INMEDIATO = ¿VERDADERO?
460
NO
SÍ
REALIZAR LECTURA / ESCRITURA DE MEMORIA CACHÉ
462
REALIZAR LECTURA / ESCRITURA INMEDIATA
464

FIG. 30

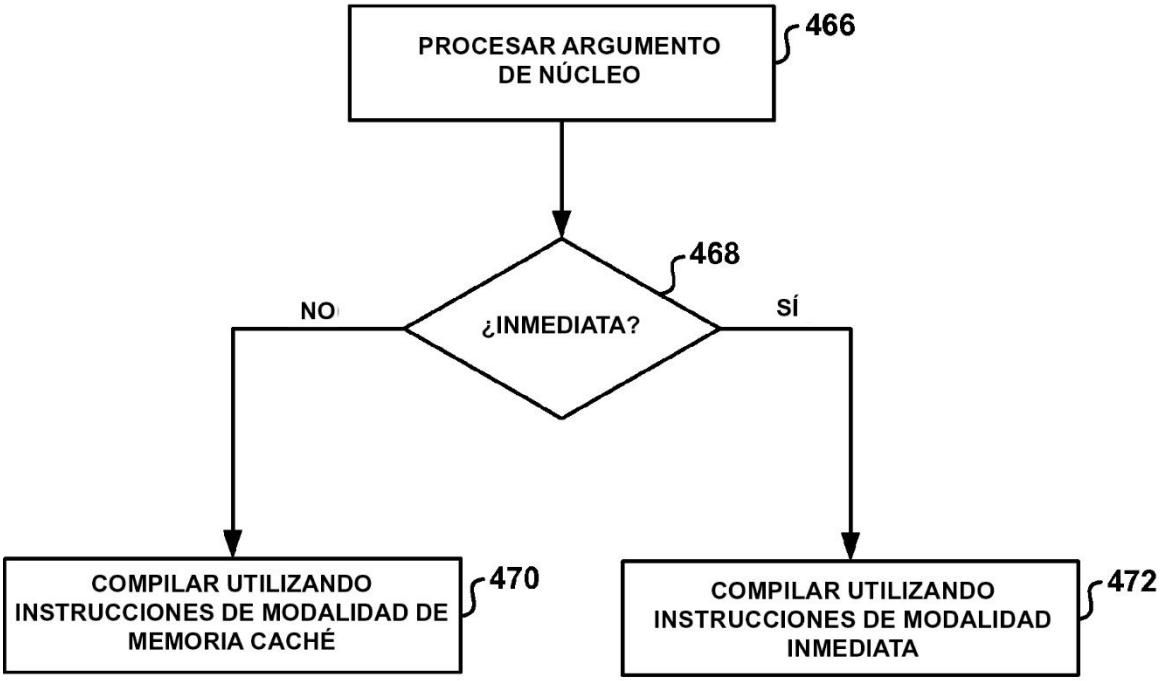
PROCESAR ARGUMENTO DE NÚCLEO
466
¿INMEDIATA?
468
NO
SÍ
COMPILAR UTILIZANDO INSTRUCCIONES DE MODALIDAD DE MEMORIA CACHÉ
470
COMPILAR UTILIZANDO INSTRUCCIONES DE MODALIDAD INMEDIATA
472

FIG. 31

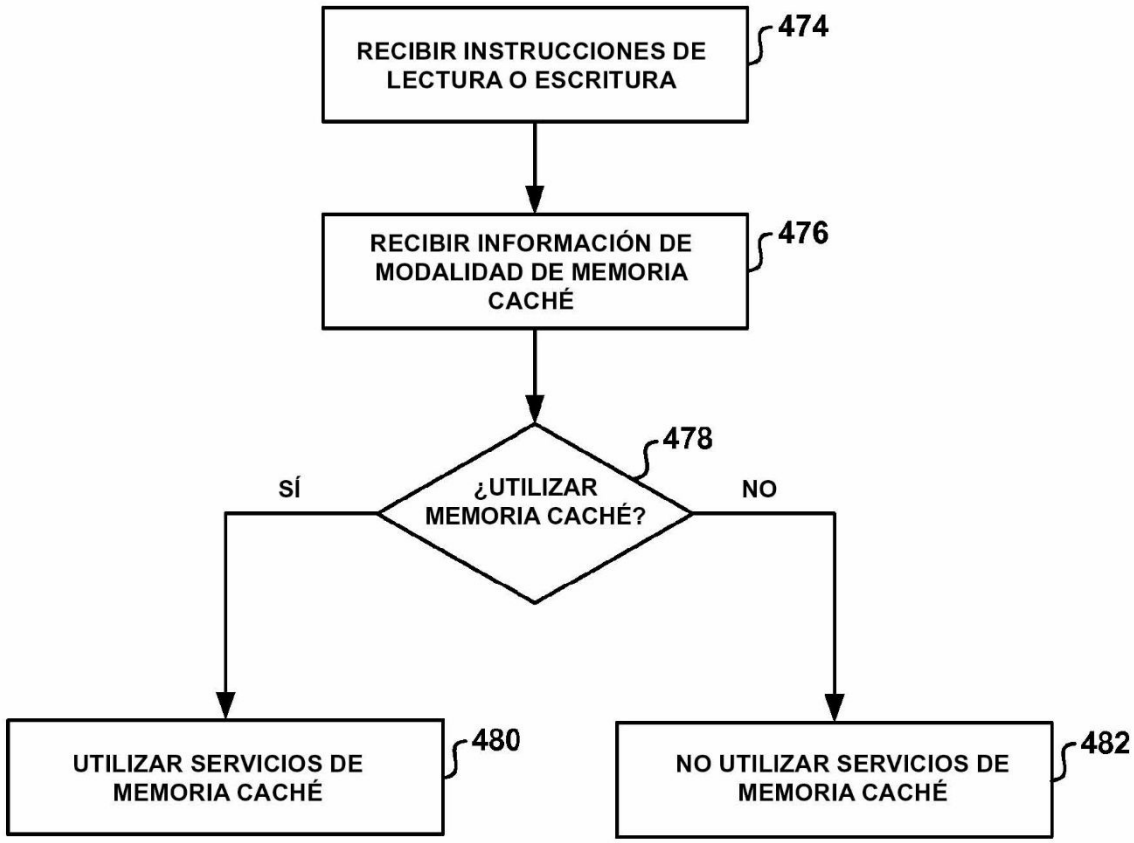
RECIBIR INSTRUCCIONES DE LECTURA O ESCRITURA
474
RECIBIR INFORMACIÓN DE MODALIDAD DE MEMORIA CACHÉ
476
¿UTILIZAR MEMORIA CACHÉ?
478
SÍ
NO
UTILIZAR SERVICIOS DE MEMORIA CACHÉ
480
NO UTILIZAR SERVICIOS DE MEMORIA CACHÉ
482

FIG. 32