(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 988 859**

(51) Int. Cl.:

G06F 17/16 (2006.01)
G06F 9/30 (2008.01)
G06N 3/063 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **03.05.2019 E 19172622 (3)**

(97) Fecha y número de publicación de la concesión europea: **29.05.2024 EP 3579117**

(54) Título: **Instrucción de multiplicación de matrices de dispersión variable de formato variable**

(30) Prioridad:

08.06.2018 US 201816003545

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
21.11.2024

(73) Titular/es:

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

(72) Inventor/es:

ANDERS, MARK A.;
KAUL, HIMANSHU y
MATHEW, SANU

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Instrucción de multiplicación de matrices de dispersión variable de formato variable

CAMPO DE LA INVENCIÓN

El campo de la invención se refiere en general a arquitectura de procesador de ordenador y, más específicamente, a un procesador, un método realizado por el procesador y aparato adaptado para procesar instrucción de multiplicación de matrices de dispersión variable de formato variable.

ESTADO DE LA TÉCNICA ANTERIOR

Las arquitecturas de aprendizaje automático, tales como las redes neuronales profundas, se han aplicado a campos que incluyen la visión informática, el reconocimiento de voz, el procesamiento de lenguaje natural, el reconocimiento de audio, el filtrado de redes sociales, la traducción automática, la bioinformática y el diseño de fármacos. El aprendizaje profundo es una clase de algoritmos de aprendizaje automático. Maximizar la flexibilidad y la rentabilidad de los algoritmos y cálculos de aprendizaje profundo puede ayudar a satisfacer las necesidades de los procesadores de aprendizaje profundo, por ejemplo, aquellos que realizan aprendizaje profundo en un centro de datos.

La multiplicación de matrices es un limitador clave de rendimiento/potencia para muchos algoritmos, incluyendo el aprendizaje automático. Algunos enfoques de multiplicación de matrices convencionales están especializados, por ejemplo, carecen de la flexibilidad para soportar una diversidad de formatos de datos (número entero 8b/16b con signo y sin signo, coma flotante 16b) con acumuladores anchos, y la flexibilidad para soportar matrices tanto densas como dispersas.

El documento US 2016/283240 A1 se refiere a métodos y aparatos relacionados con la aceleración de la multiplicación de vectores. Por ejemplo, un aparato incluye una primera memoria intermedia para almacenar una primera línea de caché de índices para elementos de un primer vector, una segunda memoria intermedia para almacenar una segunda línea de caché de índices para elementos de un segundo vector, una unidad de comparación para comparar cada índice de la primera línea de caché de índices con cada índice de la segunda línea de caché de índices, una pluralidad de multiplicadores para multiplicar cada uno un elemento del primer vector y un elemento del segundo vector para una coincidencia de índice de la unidad de comparación para producir un producto, y un sumador para sumar el producto de cada uno de la pluralidad de multiplicadores.

SAU-GEE CHEN SAU-GEE CHEN ET AL, "New Systolic Arrays for Matrix Multiplication", PARALLEL PROCESSING, 1994. ICPP 1994. INTERNATIONAL CONFERENCE ON, IEEE, PISCATAWAY, NJ, Estados Unidos, páginas 211 - 215, describe matrices sistólicas para la multiplicación de matrices.

El documento US 4 686 645 A se refiere a un procesador de datos digital para matriz/multiplicación de matriz que incluye una matriz sistólica de sumadores completos activados sincronizados al vecino más cercano. Los sumadores están dispuestos para multiplicar dos bits de datos de entrada y para sumar su producto a un bit de suma acumulativa de entrada y un bit de acarreo de un cálculo de bits de orden inferior. Los bits de datos de resultado y de entrada se emiten a celdas vecinas respectivas, recirculándose un nuevo bit de acarreo para su posterior adición a un cálculo de bits de orden superior. Se introducen elementos de columna de una matriz y elementos de fila de la otra a cualquier lado de la matriz en serie de bits, los bits menos significativos al principio, para contrapropagación mutua a través de la misma con un retardo de tiempo acumulativo entre la entrada de columnas o filas adyacentes. Las interacciones de matriz a nivel de bits para el cálculo de matriz de producto se producen en células individuales. Los pares de árboles de sumadores intercalados se conectan de manera conmutable a la matriz para acumular contribuciones de nivel de bits a elementos de matriz de producto.

SUMARIO

La presente invención está definida en las reivindicaciones independientes. Las reivindicaciones dependientes definen realizaciones de las mismas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La presente invención se ilustra a modo de ejemplo y sin limitación en las figuras de los dibujos adjuntos, en los que referencias similares indican elementos similares y en los que:

La **Figura 1** es un diagrama de bloques que ilustra los componentes de procesamiento para ejecutar una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con una realización;

La **Figura 2** es un diagrama de bloques de una matriz de procesamiento para ejecutar una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones;

La **Figura 3** es un diagrama de flujo de bloques que ilustra la ejecución parcial de una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones;

La **Figura 4** es un diagrama de flujo de bloques que ilustra una canalización de ejecución para ejecutar una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones;

La **Figura 5** es un diagrama de bloques que ilustra señales de control de enrutamiento compartidas entre unidades de procesamiento y circuitería de enrutamiento mientras se ejecuta una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones;

La **Figura 6** es un diagrama de flujo de bloques que ilustra un procesador que ejecuta una multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones;

La **Figura 7** es un diagrama de bloques que ilustra un circuito de acumulación de multiplicación de números enteros/coma flotante de precisión variable, de acuerdo con algunas realizaciones;

Las **Figuras 8A-8C** son diagramas de bloques que ilustran un formato de instrucción compatible con vectores genérico y plantillas de instrucciones del mismo de acuerdo con realizaciones de la invención;

La **Figura 8A** es un diagrama de bloques que ilustra un formato para una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones;

La **Figura 8B** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores genérico y plantillas de instrucciones de clase A del mismo de acuerdo con realizaciones de la invención;

La **Figura 8C** es un diagrama de bloques que ilustra el formato de instrucción compatible con vectores genérico y plantillas de instrucciones de clase B del mismo acuerdo con realizaciones de la invención;

La **Figura 9A** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores específico ilustrativo de acuerdo con realizaciones de la invención;

La **Figura 9B** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico que componen el campo de código de operación completo de acuerdo con una realización de la invención;

La **Figura 9C** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico que componen el campo de índice de registro de acuerdo con una realización de la invención;

La **Figura 9D** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico que componen el campo de operación de aumento de acuerdo con una realización de la invención;

La **Figura 10** es un diagrama de bloques de una arquitectura de registro de acuerdo con una realización de la invención;

La **Figura 11A** es un diagrama de bloques que ilustra tanto una canalización en orden como una canalización de emisión/ejecución en desorden y cambio de nombre de registro ilustrativas de acuerdo con realizaciones de la invención;

La **Figura 11B** es un diagrama de bloques que ilustra tanto una realización ilustrativa de un núcleo de arquitectura en orden como un núcleo de arquitectura de emisión/ejecución en desorden y cambio de nombre de registro ilustrativos que se van a incluir en un procesador de acuerdo con realizaciones de la invención;

Las Figuras 12A-B ilustran un ejemplo más específico de un diagrama de bloques de una arquitectura de núcleo en orden, cuyo núcleo sería uno de varios bloques lógicos (que incluyen otros núcleos del mismo tipo y/o tipos diferentes) en un chip;

La **Figura 12A** es un diagrama de bloques de un único núcleo de procesador, junto con su conexión a la red de interconexión en el encapsulado y con su subconjunto local de la caché de nivel 2 (L2), de acuerdo con realizaciones de la invención;

La **Figura 12B** es una vista ampliada de parte del núcleo del procesador en la **Figura 12A** de acuerdo con realizaciones de la invención;

La **Figura 13** es un diagrama de bloques de un procesador que puede tener más de un núcleo, puede tener un controlador de memoria integrado y puede tener gráficos integrados de acuerdo con realizaciones de la invención;

Las **Figuras 14-17** son diagramas de bloques de arquitecturas informáticas ilustrativas;

La **Figura 14** muestra un diagrama de bloques de un sistema de acuerdo con una realización de la presente invención;

La **Figura 15** es un diagrama de bloques de un primer sistema ilustrativo más específico de acuerdo con una realización de la presente invención;

La **Figura 16** es un diagrama de bloques de un segundo sistema ilustrativo más específico de acuerdo con una realización de la presente invención;

La **Figura 17** es un diagrama de bloques de un sistema en un chip (SoC) de acuerdo con una realización de la presente invención; y

La **Figura 18** es un diagrama de bloques que contrasta el uso de un convertidor de instrucciones de software para convertir instrucciones binarias en un conjunto de instrucciones de origen en instrucciones binarias en un conjunto de instrucciones objetivo de acuerdo con realizaciones de la invención.

DESCRIPCIÓN DETALLADA DE LAS REALIZACIONES

En la siguiente descripción, se exponen numerosos detalles específicos. Sin embargo, se entiende que algunas realizaciones pueden ponerse en práctica sin estos detalles específicos. En otros casos, no se han mostrado en detalle circuitos, estructuras y técnicas bien conocidos para no complicar la comprensión de esta descripción.

Las referencias en la memoria descriptiva a "una realización", "una realización ilustrativa", etc., indican que la realización descrita puede incluir un rasgo, estructura o característica, pero cada realización no necesariamente puede incluir el rasgo, estructura, o característica. Además, tales expresiones no se refieren necesariamente a la misma realización. Además, cuando se describe un rasgo, estructura o característica en relación con una realización, se afirma que está dentro del conocimiento de un experto en la materia modificar tal rasgo, estructura o característica acerca de otras realizaciones si se describe explícitamente.

Las realizaciones divulgadas proporcionan ejecución mejorada de una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM). Las realizaciones divulgadas realizan multiplicación de matriz y multiplicación-acumulación para una diversidad de diferentes formatos de datos que incluyen formatos de números enteros de 8 bits/16 bits con signo/sin signo y de coma flotante de 16 bits/32 bits. Además, las realizaciones divulgadas usan enrutamiento basado en toma de contacto de bloqueo con difusión de elementos de datos de matriz entre nodos en una matriz de procesamiento para soportar operandos de matriz densa o dispersa y para evitar la multiplicación por ceros de matrices dispersas. Además, un modo de 8 bits divulgado se optimiza para lograr un caudal de 4x reconfigurando cada unidad de procesamiento en la matriz de procesamiento en una matriz de 2x2.

Como se usa en el presente documento, la dispersión de una matriz se define como la proporción de elementos distintos de cero, siendo los elementos restantes cero o nulos. Por ejemplo, en algunas realizaciones, una matriz dispersa que tiene una dispersión de 0,125, tiene únicamente 1/8, o 12,5 % de sus elementos con valores distintos de cero. También es posible que la dispersión se use para referirse a la proporción de elementos que tienen un valor cero. De cualquier manera, las realizaciones divulgadas aprovechan la dispersión de una o ambas matrices en una multiplicación de matrices para mejorar la potencia, el rendimiento, la flexibilidad y/o la funcionalidad.

Se espera que la flexibilidad, la funcionalidad y el coste se mejoren mediante las realizaciones divulgadas proporcionando circuitería para multiplicar matrices que tienen diversos formatos y diversos grados de dispersión. A diferencia de algunos enfoques que usarían hardware diferente especializado dedicado a cada uno de los diversos formatos y dispersión, las realizaciones divulgadas proporcionan hardware que puede configurarse para acomodar las variaciones. A diferencia de algunos enfoques que desperdiciarían potencia y rendimiento multiplicando elementos cero de las matrices, las realizaciones divulgadas evitan al menos algunas multiplicaciones de cero cuando se opera en un modo denso-disperso o disperso-disperso, como se describe a continuación.

Se espera que las realizaciones divulgadas mejoren el coste y el área proporcionando un único circuito de ejecución reconfigurable para soportar una diversidad de formatos de datos, incluyendo tanto números enteros como de coma flotante, en comparación con algunos enfoques que se basan en diferentes circuitos que se especializan en diferentes formatos de datos. Las realizaciones divulgadas proporcionan un acelerador de multiplicación de matrices que soporta tanto formatos de datos de coma flotante como de números enteros, con acumulación. El acelerador divulgado también puede optimizarse para operar en matrices dispersas, evitando multiplicar los elementos cero. Combinando estas características en un circuito reconfigurable, las realizaciones divulgadas permiten, por lo tanto, que un circuito acelerador de multiplicación de matriz única soporte múltiples formatos de precisión con acumuladores anchos, mientras se reconfigura eficientemente para matrices densas o dispersas. Las realizaciones de acelerador divulgadas mejoran la eficiencia de área y energía mientras que proporcionan flexibilidad para soportar muchas cargas de trabajo de multiplicación de matrices típicas, tales como aprendizaje automático.

La **Figura 1** es un diagrama de bloques que ilustra componentes de procesamiento para ejecutar instrucción o instrucciones de multiplicación de matrices de dispersión variable de formato variable VFVSMM 103, de acuerdo con algunas realizaciones. Como se ilustra, el almacenamiento 101 almacena la instrucción o instrucciones de VFVSMM 103 para ser ejecutadas. Como se describe además a continuación, en algunas realizaciones, el sistema informático 100 es un procesador de una única instrucción, múltiples datos (SIMD) para procesar concurrentemente múltiples elementos de datos basándose en una única instrucción.

En operación, la instrucción o instrucciones de VFVSMM 103 deben extraerse del almacenamiento 101 mediante el circuito de extracción 105. La instrucción de VFVSMM 107 extraída se va a decodificar mediante el circuito de decodificación 109. El formato de instrucción de VFVSMM, que se ilustra y describe además con respecto a las **Figuras 7**, 8A-B y 9A-D, tiene campos (no mostrados en este punto) para especificar un código de operación y vectores complejos de destino, multiplicador, multiplicando y sumando. La circuitería de decodificación 109 decodifica la instrucción de VFVSMM 107 extraída en una o más operaciones. En algunas realizaciones, esta decodificación incluye generar una pluralidad de microoperaciones para ser realizadas por la circuitería de ejecución (tal como la circuitería de ejecución 119) junto con la circuitería de enrutamiento 118. La circuitería de decodificación 109 también decodifica sufijos y prefijos de instrucciones (si se usan). La circuitería de ejecución 119, que opera junto con la circuitería de enrutamiento 117, se describe e ilustra con más detalle a continuación, al menos con respecto a las **Figuras 2-6**, 11A-B y **12A-B.**

En algunas realizaciones, el circuito 113 de cambio de nombre de registro, asignación de registro y/o planificación proporciona funcionalidad para uno o más de: 1) cambiar el nombre de los valores de operandos lógicos a valores de operandos físicos (por ejemplo, una tabla de alias de registro en algunas realizaciones), 2) asignar bits y banderas de estado a la instrucción decodificada, y 3) planificar la instrucción de VFVSMM decodificada 111 para su ejecución en la circuitería de ejecución 119 de un agrupamiento de instrucciones (por ejemplo, usando una estación de reserva en algunas realizaciones).

Los registros (archivo de registro) y/o la memoria 115 almacenan datos como operandos de la instrucción de VFVSMM decodificada 111 para ser operados por la circuitería de ejecución 119. Los tipos de registro ilustrativos incluyen registros de máscara de escritura, registros de datos empaquetados, registros de propósito general y registros de coma flotante, como se describe e ilustra con mayor detalle a continuación, al menos con respecto a la **Figura 10.**

En algunas realizaciones, el circuito de reescritura 120 confirma el resultado de la ejecución de la instrucción de VFVSMM decodificada 111. La circuitería de ejecución 119 y el sistema 100 se ilustran y describen además con respecto a las **Figuras 2-6**, 11A-B y **12A-B.**

La **Figura 2** es un diagrama de bloques de una matriz de procesamiento para ejecutar una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones. Como se muestra, el sistema 200 incluye la matriz de entrada A 202, la matriz de entrada B 204, la matriz de salida C 206 y la circuitería de enrutamiento 212. También se muestra la circuitería de ejecución 208, que incluye una matriz de procesamiento de unidades de procesamiento (M x N) 210. En algunas realizaciones, cada una de las unidades de procesamiento en la matriz de procesamiento es un circuito de multiplicación-acumulación, una vista ampliada del cual se muestra como MAC 214.

Una ventaja de las unidades de procesamiento divulgadas 210 es que pueden reutilizarse para realizar multiplicaciones y acumulaciones de multiplicación en matrices que tienen una diversidad de formatos diferentes. Por ejemplo, como se describe e ilustra con respecto a la **Figura 8A,** las realizaciones divulgadas pueden realizar la instrucción de VFVSMM en cualquiera de diversos formatos de datos, tal como número entero de 8 bits, número entero de 16 bits, número entero de 32 bits, coma flotante de precisión media, coma flotante de precisión sencilla o coma flotante de precisión doble, y tamaño de elemento 712, en términos de un número de bits por elemento de matriz. Evitando la necesidad de implementar hardware diferente para procesar diferentes tipos de datos, las realizaciones divulgadas proporcionan un beneficio de potencia y coste reutilizando la misma circuitería para diferentes tipos de datos.

En algunas realizaciones, por ejemplo, cuando se procesan datos de números enteros de 8 bits, el caudal de circuitería de ejecución se cuadriplica configurando cada unidad de procesamiento para realizar una multiplicación de matriz de 2x2.

Como se describe en el presente documento, una unidad de procesamiento en ocasiones se denomina un elemento de procesamiento y en ocasiones se denomina un circuito de procesamiento y en ocasiones se puede denominar como un nodo de procesamiento. Independientemente de la redacción, se pretende que la unidad de procesamiento comprenda circuitería para realizar cálculos de ruta de datos y proporcionar lógica de control.

En operación, la circuitería de enrutamiento 212 y la circuitería de ejecución 208 operan en modo denso-denso, modo denso-disperso o modo disperso-disperso, como sigue.

MODO DENSO-DENSO

En algunas realizaciones, la circuitería de enrutamiento y de ejecución se colocan en un modo denso-denso por software, por ejemplo, estableciendo un registro de control que controla la circuitería de enrutamiento y ejecución. Las realizaciones divulgadas mejoran la potencia y el rendimiento evitando multiplicaciones que implican cero elementos de la matriz dispersa. Las realizaciones divulgadas proporcionan una ventaja de coste permitiendo que la misma circuitería se reutilice en diversos modos, incluyendo en diversas condiciones de dispersión y con diversos formatos de datos.

En algunas realizaciones, la circuitería de enrutamiento y ejecución se coloca en un modo denso-denso según se da instrucción por la instrucción de VFVSMM, por ejemplo, usando un sufijo del código de operación. El formato de la instrucción de VFVSMM se describe e ilustra adicionalmente con respecto a las Figuras 8A-C y 9A-D. En algunas realizaciones, la circuitería de enrutamiento y ejecución entra en un modo denso-denso en respuesta a que una o ambas de las matrices A y B se almacenen en formato empaquetado, con un especificador que acompaña a cada elemento de matriz y que especifica la posición lógica del elemento dentro del matriz lógica A o B.

En operación, la circuitería de ejecución, usando la circuitería de enrutamiento 212, que opera en un modo denso-denso, en respuesta a la instrucción de VFVSMM decodificada, ha de enrutar cada fila de la matriz A especificada, escalonando las filas posteriores, a una fila correspondiente de una matriz de procesamiento que tiene unidades de procesamiento (M x N), y enrutar cada columna de la matriz B especificada, escalonando columnas posteriores, a una columna correspondiente de la matriz de procesamiento. En algunas realizaciones, cada fila y columna se escalona por un ciclo de reloj, permitiendo que cada elemento de procesamiento infiera las direcciones de fila y columna de cada elemento de matriz A y matriz B recibido basándose en un ciclo de reloj y en una posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento.

Continuando la operación, cada una de las unidades de procesamiento (MxN) ha de generar un producto de cada uno de los K elementos de datos recibidos horizontalmente y K elementos de datos recibidos verticalmente, y acumular cada producto generado con un valor previo de un elemento correspondiente de la matriz C que tiene una misma posición de matriz relativa que una posición de la unidad de procesamiento en la matriz.

MODO DENSO-DISPERSO

En algunas realizaciones, la circuitería de enrutamiento y la circuitería de ejecución se colocan en un modo denso-disperso por software, por ejemplo, estableciendo un registro de control que controla la circuitería de enrutamiento y ejecución. Las realizaciones divulgadas mejoran la potencia y el rendimiento evitando multiplicaciones que implican cero elementos de la matriz dispersa. Las realizaciones divulgadas proporcionan una ventaja de coste permitiendo que la misma circuitería se reutilice en diversos modos, incluyendo en diversas condiciones de dispersión y con diversos formatos de datos.

En algunas realizaciones, la circuitería de enrutamiento y ejecución se coloca en un modo denso-disperso según se da instrucción por la instrucción de VFVSMM, por ejemplo, usando un sufijo del código de operación. El formato de la instrucción de VFVSMM se describe e ilustra adicionalmente con respecto a las Figuras 8A-C y 9A-D. En algunas realizaciones, la circuitería de enrutamiento y ejecución entra en un modo denso-disperso en respuesta a que una o ambas de las matrices A y B se almacenen en formato empaquetado, con un especificador que acompaña a cada elemento de matriz y que especifica la posición lógica del elemento dentro de la matriz lógica A o B.

Debería observarse que se puede hacer que el procesador ejecute una instrucción de VFVSMM en modo denso-disperso incluso si ambas matrices A y B son matrices densas. Siempre que la matriz dispersa en una situación de este tipo esté formateada para incluir información de dirección para cada elemento de datos, el procesador podría construirse para realizar una o más comprobaciones de dirección que, cada una, determinaría una coincidencia de dirección porque las matrices A y B son matrices densas con direcciones adyacentes separadas por uno. Operar en modo denso-disperso en una situación de este tipo incurriría, por lo tanto, en algún coste de ejecución adicional, pero puede simplificar la tarea de ejecutar la instrucción de VFVSMM.

En algunas realizaciones, en las que el procesador va a ejecutar la instrucción de VFVSMM en modo denso-disperso, la matriz B especificada es una matriz dispersa (que tiene una dispersión menor que uno, definiéndose la dispersión como la proporción de elementos distintos de cero, siendo los elementos restantes cero o nulos), y debe comprender únicamente elementos distintos de cero de una matriz lógica que comprende (K x N) elementos, incluyendo cada elemento un campo para especificar sus direcciones de fila y columna lógicas.

En operación, la circuitería de ejecución, usando la circuitería de enrutamiento 212 y operando en el modo denso-disperso en respuesta a la instrucción de VFVSMM decodificada, ha de enrutar cada fila de la matriz A especificada, escalonando las filas posteriores, a la fila correspondiente de una matriz de procesamiento (M x N), y enrutar cada columna de la matriz B especificada a la columna correspondiente de la matriz de procesamiento.

Siendo una matriz dispersa, la matriz B especificada ha de almacenarse y cargarse desde la memoria en un formato disperso empaquetado, almacenando únicamente los elementos distintos de cero. Por lo tanto, los elementos posteriores de las matrices A y B pueden tener huecos en sus direcciones de fila y columna. En algunas realizaciones, cada fila está escalonada por un ciclo de reloj, permitiendo que cada elemento de procesamiento infiera la dirección de columna de cada elemento de matriz A recibido basándose en un ciclo de reloj.

Continuando la operación, cada una de las unidades de procesamiento (MxN) en la matriz de procesamiento 210, que opera en un modo denso-disperso, ha de determinar, basándose en el reloj y en una posición de la unidad de procesamiento dentro de la matriz de procesamiento 210, una columna y una dirección de fila de cada elemento recibido horizontalmente. Cada una de las unidades de procesamiento (MxN) en la matriz de procesamiento 210 ha de determinar a continuación si existe una coincidencia de dirección entre la dirección de fila lógica del elemento recibido verticalmente y la dirección de columna del elemento recibido horizontalmente. Cuando existe la coincidencia, la unidad de procesamiento ha de generar el producto. Y, cuando no existe coincidencia, la unidad de procesamiento ha de mantener el elemento recibido horizontalmente y pasar el elemento recibido verticalmente si la dirección de columna es mayor que la dirección de fila lógica, de lo contrario, ha de mantener el elemento recibido verticalmente y pasar el elemento recibido horizontalmente.

MODO DISPERSO-DISPERSO

En algunas realizaciones, la circuitería de enrutamiento y la circuitería de ejecución se colocan en un modo disperso-disperso por software, por ejemplo. estableciendo un registro de control que controla la circuitería de enrutamiento y ejecución. Las realizaciones divulgadas mejoran la potencia y el rendimiento evitando multiplicaciones que implican cero elementos de la matriz dispersa. Las realizaciones divulgadas proporcionan una ventaja de coste permitiendo que la misma circuitería se reutilice en diversos modos, incluyendo en diversas condiciones de dispersión y con diversos formatos de datos.

En algunas realizaciones, la circuitería de enrutamiento y ejecución se coloca en el modo disperso-disperso como se indica por la instrucción de VFVSMM, por ejemplo, usando un sufijo del código de operación. El formato de la instrucción de VFVSMM se describe e ilustra además con respecto a las Figuras 8A-C y 9A-D. En algunas realizaciones, la circuitería de enrutamiento y ejecución entra en el modo disperso-disperso en respuesta a que las matrices A y B se almacenan en formato empaquetado, con un especificador que acompaña a cada elemento y que especifica la posición lógica del elemento dentro de la matriz lógica A o B.

En algunas realizaciones, en las que el procesador va a ejecutar la instrucción de VFVSMM en modo disperso-disperso, las matrices A y B especificadas son ambas matrices dispersas (que tienen una dispersión menor que uno, definiéndose la dispersión como la proporción de elementos distintos de cero, siendo los elementos restantes cero o nulos). En tales realizaciones, las matrices A y B especificadas ha de almacenarse cada una en la memoria como una matriz dispersa empaquetada que comprende únicamente elementos distintos de cero de matrices lógicas (M x K) y (K x N), respectivamente, incluyendo cada elemento un campo para especificar sus direcciones de fila y columna lógicas.

En operación, la circuitería de ejecución, usando la circuitería de enrutamiento 212 que opera en el modo disperso-disperso, ha de enrutar cada fila de la matriz A especificada a la fila correspondiente de una matriz de procesamiento (M x N), y enrutar cada columna de la matriz B especificada en la columna correspondiente de la matriz de procesamiento.

Al ser matrices dispersas, las matrices A y B especificadas han de almacenarse en y cargarse desde la memoria en formato disperso empaquetado, almacenando únicamente los elementos distintos de cero e incluyendo una dirección del elemento en la matriz lógica. Por lo tanto, los elementos posteriores de las matrices A y B pueden tener huecos en sus direcciones de fila y columna.

Continuando la operación, cada una de las unidades de procesamiento (MxN), que opera en un modo disperso-disperso, ha de comparar la dirección de fila lógica del elemento recibido verticalmente y la dirección de columna lógica del elemento recibido horizontalmente. Cuando existe una coincidencia de dirección, la unidad de procesamiento ha de generar el producto. Y, cuando no existe coincidencia, la unidad de procesamiento ha de

mantener el elemento recibido horizontalmente y pasar el elemento recibido verticalmente si la dirección de columna lógica es mayor que la dirección de fila lógica y, de lo contrario, mantener el elemento recibido verticalmente y pasar el elemento recibido horizontalmente.

La **Figura 3** es un diagrama de flujo de bloques que ilustra la ejecución parcial de una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones. Como se muestra, la circuitería de ejecución 300 incluye cuadrícula de MAC 308, cuyas las filas reciben elementos de matriz A de filas correspondientes de la matriz de entrada A 302, y cuyas columnas reciben elementos de matriz B de columnas correspondientes de la matriz de entrada B 304. La cuadrícula de MAC 308 produce la matriz de salida C 306. En algunas realizaciones, cuando se opera en modo denso-denso, las filas y columnas se "escalonan" por un ciclo de reloj, permitiendo que cada elemento de procesamiento infiera las direcciones de fila y columna de cada elemento de matriz A y matriz B recibido basándose en un ciclo de reloj y en una posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento. Como se muestra, la fila 1 de la matriz de entrada A 302 se enruta un ciclo antes de la fila 2, y dos ciclos antes de la carga 3.

En operación, cuando se opera en un modo denso-denso, cada una de las unidades de procesamiento (MxN) ha de generar K productos de elementos de matriz A y matriz B coincidentes recibidos de las matrices A y B especificadas, respectivamente, para que exista una coincidencia cuando el elemento de matriz B tiene la misma dirección de fila que una dirección de columna del elemento de matriz A, y para acumular cada producto generado con un elemento correspondiente de la matriz C especificada que tiene una misma posición relativa que una posición de la unidad de procesamiento en la matriz de procesamiento.

Cuando se opera en un modo denso-disperso, cada una de las MAC (MxN) en la cuadrícula de las MAC 308 ha de determinar si existe una coincidencia de dirección entre la dirección de fila lógica especificada del elemento de matriz B y la dirección de columna del elemento de matriz A, y, cuando existe la coincidencia, generar el producto, y, cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B cuando la dirección de columna del elemento de matriz A es mayor que la dirección de fila lógica especificada del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A.

Cuando se opera en modo disperso-disperso, cada una de las MAC (MxN) en la cuadrícula de las MAC 308 ha determinar si existe una coincidencia entre la dirección de columna lógica especificada del elemento de matriz A y la dirección de fila lógica especificada del elemento de matriz B, y, cuando existe la coincidencia, generar el producto, y, cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B cuando la dirección de columna lógica especificada del elemento de matriz A es mayor que la dirección de fila lógica especificada del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A.

El circuito de ejecución 300 se describe e ilustra con más detalle a continuación, al menos con respecto a las **Figuras 2**, 4-6, 11A-B y **12A-B**

La **Figura 4** es un diagrama de flujo de bloques que ilustra una canalización de ejecución para ejecutar una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones. Como se muestra, la matriz A 402 y la matriz B 404 son ambas matrices densas. En operación, la circuitería de ejecución, que opera en un modo denso-denso, en respuesta a la instrucción de VFVSMM decodificada, ha de enrutar cada fila de la matriz A especificada, escalonando las filas posteriores, a una fila correspondiente de una matriz de procesamiento que tiene unidades de procesamiento (M x N), y enrutar cada columna de la matriz B especificada, escalonando columnas posteriores, a una columna correspondiente de la matriz de procesamiento. También se muestran siete instantáneas consecutivas 408A-G, tomadas en siete puntos a lo largo de la línea de tiempo 400: 418, 420, 422, 424, 426, 426 y 428.

En algunas realizaciones, escalonar filas y columnas posteriores se refiere a retardar el enrutamiento de cada fila y columna posteriores en una fila y columna correspondientes de la matriz de procesamiento en un ciclo. Escalonar las filas y columnas posteriores ofrece la ventaja de crear una canalización para alinear horizontal y verticalmente para realizar 27 multiplicaciones-acumulaciones durante 7 ciclos, como se muestra en las instantáneas 408A-G. Escalonar filas y columnas también permite que cada elemento de procesamiento infiera las direcciones de fila y columna de cada elemento de matriz A y matriz B recibido basándose en un ciclo de reloj y en una posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento

Como se muestra en la instantánea 408A, la fila 0 de la matriz A 402 y la columna 0 de la matriz B 404 se enrutan a una fila correspondiente y una columna correspondiente de la matriz de procesamiento. También en 408A, los elementos correspondientes de la matriz A 402 y la matriz B 404 se multiplican en y se acumulan con datos anteriores en el elemento C(0,0) de la matriz C 406.

En la instantánea 408B, habiendo transcurrido un ciclo, la fila 0 de la matriz A 402 y la columna 0 se han desplazado en 1 punto, y la fila 1 de la matriz A 402 y la columna 1 de la matriz B 404 se enrutan a una fila correspondiente y una columna correspondiente de la matriz de procesamiento. También en 408B, los elementos correspondientes de la

matriz A 402 y la matriz B 404 se multiplican en C(0,0), C(0,1) y C(1, 0). El producto generado en C(0,0) se acumula con el valor previo generado en 408A.

En la instantánea 408C, habiendo transcurrido otro ciclo, las filas 0 y 1 de la matriz A 402 y las columnas 0 y 1 de la matriz B 404 se han desplazado en 1 punto, y la fila 2 de la matriz A 402 y la columna 2 de la matriz B 404 se enrutan a una fila correspondiente y una columna correspondiente de la matriz de procesamiento. También en 408C, los elementos correspondientes de la matriz A 402 y la matriz B 404 se multiplican y acumulan con valores previos, si los hubiera, en C(0,0), C(0,1), C(0,2), C (1, 0), C(1,1) y C(2,0). Como se indica por su contorno en negrita, el producto acumulado generado en C(0,0) es el valor final de C(0,0).

En la instantánea 408D, habiendo transcurrido otro ciclo, las filas 0-2 de la matriz A 402 y las columnas 0-2 de la matriz B 404 se han desplazado en 1 punto. También en 408D, los elementos correspondientes de la matriz A 402 y la matriz B 404 se multiplican y acumulan con valores anteriores, si los hubiera, en C(0,1), C(0,2), C(1, 0), C (1,1), C(1,2), C(2,0) y C(2,1). Como se indica por sus contornos en negrita, los productos acumulados generados en C(0,1) y C(1,0) son los valores finales de C(0,1) y C(1,0).

En la instantánea 408E, habiendo transcurrido otro ciclo, las filas 0-2 de la matriz A 402 y las columnas 0-2 de la matriz B 404 se han desplazado en 1 punto. También en 408E, los elementos correspondientes de la matriz A 402 y la matriz B 404 se multiplican y acumulan con valores anteriores, si los hubiera, en C(0,2), C(1,1), C(1,2), C (2,0), C(2,1) y C(2,2). Como se indica por sus contornos en negrita, los productos acumulados generados en C(0,2), C(1,1) y C(2,0) son los valores finales de C(0,2), C(1,1) y C(2,0).

En la instantánea 408F, habiendo transcurrido otro ciclo, las filas 1-2 de la matriz A 402 y las columnas 1-2 de la matriz B 404 se han desplazado en 1 punto. También en 408F, los elementos correspondientes de la matriz A 402 y la matriz B 404 se multiplican y acumulan con valores previos, si los hubiera, en C(1,2), C(2,1) y C(2,2). Como se indica por sus contornos en negrita, los productos acumulados generados en C(2,1) y C(1,2) son los valores finales de C(1,2) y C(2,1).

En la instantánea 408G, habiendo transcurrido otro ciclo, la fila 2 de la matriz A 402 y la columna 2 de la matriz B 404 se han desplazado en 1 punto. También en 408G, los elementos correspondientes de la matriz A 402 y la matriz B 404 se multiplican y acumulan con valores previos, si los hubiera, en C(2,2). Como se indica por su contorno en negrita, el producto acumulado generado en C(2,2) son los valores finales de C(2,2).

La **Figura 5** es un diagrama de bloques que ilustra señales de control de enrutamiento compartidas entre unidades de procesamiento y circuitería de enrutamiento mientras se ejecuta una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones. Se ilustran cuatro instantáneas 550A-550D a lo largo de cuatro ciclos de una porción de cuatro nodos de una única fila de unidades de procesamiento que operan en modo disperso-disperso. Cada una de las cuatro instantáneas 550A-550D ilustra cuatro unidades de procesamiento 554A-D - 560A-D, cada una de las cuales recibe una entrada de elemento de datos vertical con una dirección de fila, y una entrada de elemento de datos horizontal con una dirección de columna. En algunas realizaciones, las direcciones de fila de los elementos horizontales y las direcciones de columna de los elementos verticales se definen implícitamente por la posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento.

Debido a los requisitos de formato de las matrices de entrada A y B, las direcciones de fila y columna nunca disminuirán. En su lugar, las direcciones de fila y columna de elementos de datos consecutivos aumentarán cada una en uno cuando esté en modo denso y en cero o más cuando esté en modo disperso hasta que se alcance el final de fila o final de columna (la dirección no cambiará en modo disperso si se declaró una solicitud de mantenimiento en el ciclo anterior, como se ilustra y se describe con respecto a la instantánea 550A).

Para ilustrar señales de control de toma de contacto compartidas entre nodos de acuerdo con algunas realizaciones, la matriz de procesamiento de la **Figura 5** ha de operar en modo disperso-disperso.

En operación, cada unidad de procesamiento que opera en un modo disperso ha de comparar la dirección de fila de elementos recibidos verticalmente con la dirección de columna de elementos recibidos horizontalmente. (Observe que tal comprobación de dirección puede realizarse, pero no es necesaria cuando se opera en modo denso, durante la que la dirección de cada elemento ha de aumentar exactamente en uno. Cuando se opera en modo denso-disperso, únicamente es necesario comprobar las direcciones de entradas recibidas de la matriz dispersa).

Si las direcciones coinciden, y si no se solicita que el elemento vertical ni el horizontal se mantenga por una unidad de procesamiento aguas abajo, la unidad de procesamiento multiplica los elementos recibidos y acumula el producto con contenidos previos del elemento de matriz de destino correspondiente. Como se usa en el presente documento, el término "correspondiente" indica un elemento de matriz de destino que tiene una misma posición relativa dentro de la matriz de destino (M x N) que la posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento (M x N).

Pero si las direcciones no coinciden, la unidad de procesamiento ha de mantener el elemento que tiene una dirección más alta y pasar el otro elemento. Dado que las direcciones de fila y columna nunca disminuyen, no tiene sentido mantener el elemento de dirección inferior; nunca habrá una coincidencia de dirección. La unidad de procesamiento, sin embargo, mantiene el elemento con dirección más grande para usarse en caso de que el otro elemento con una dirección que coincide con la dirección llegue en el futuro. En algunas realizaciones, cada unidad de procesamiento tiene algún elemento de almacenamiento, tal como un registro o un biestable, para mantener el elemento de datos.

Además, cuando las direcciones no coinciden, la unidad de procesamiento ha de enviar una solicitud de mantenimiento aguas arriba en la dirección del elemento de datos que se está manteniendo para que el elemento de datos pueda continuar manteniéndose, y envía una señal de notificación de mantenimiento aguas abajo en la dirección del elemento de datos que se está manteniendo.

En algunas realizaciones, una unidad de procesamiento que recibe una solicitud de mantenimiento desde un nodo aguas abajo ha de generar y enviar una correspondiente solicitud de mantenimiento aguas arriba.

En algunas realizaciones, por ejemplo, como se ilustra y describe con respecto a la instantánea 550A, una unidad de procesamiento, junto con circuitería de enrutamiento, ha de difundir un elemento de datos aguas abajo a dos o más unidades de procesamiento que pueden usar el elemento.

La Tabla 1 enumera diversos controles de toma de contacto usados entre unidades de procesamiento.

Tabla 1 - Controles de toma de contacto

Señal	Descripción
Solicitud de mantenimiento	Solicitar a nodo aguas arriba que mantenga
Notificación de mantenimiento	Notificar aguas abajo del plan para mantener

Para describir las instantáneas de ejecución de la **Figura 5,** antes del ciclo 1 550A, las cuatro unidades de procesamiento 554A-560A tenían elementos verticales con direcciones de fila iguales a 3, 2, 1 y 0, respectivamente. Como se muestra, los elementos de datos verticales que llegan en el ciclo 1 550A a las unidades de procesamiento 556A y 560A tienen ambos una dirección de fila igual a "4", pero pueden tener datos diferentes porque están en columnas diferentes. También en el ciclo 1 550A, un elemento horizontal que tiene una dirección de columna igual a 4 se difunde a cada unidad de procesamiento que puede usarlo (debido a la solicitud de mantenimiento horizontal generada por la unidad de procesamiento 558A durante el ciclo 1 550A, el elemento de datos que tiene una dirección de columna igual a "4" se mantendrá durante el ciclo 2 550B en un biestable en el nodo de procesamiento 556B. También en el ciclo 1, la solicitud de mantenimiento horizontal de la unidad de procesamiento 558A hace que la unidad de procesamiento 556A mantenga el elemento horizontal, direccionado como "4" durante el siguiente ciclo, repitiendo el elemento de datos en el ciclo 2 550B.

Como se muestra, el elemento de datos vertical recibido por la unidad de procesamiento 558A en el ciclo 1, tiene una dirección de fila igual a "2" y fue mantenido por un biestable presumiblemente debido a una solicitud de mantenimiento vertical durante el ciclo anterior. No hay solicitud de mantenimiento vertical para el ciclo 1 por la unidad de procesamiento 558A. Debido a que la dirección de entrada horizontal "4" es mayor que la dirección de entrada vertical, "2", la unidad de procesamiento 558A genera una solicitud de mantenimiento horizontal y la envía de vuelta a 556A, provocando que el biestable en 556B se cierre en el ciclo 2.

Como se muestra, la unidad de procesamiento 558A durante el ciclo 1 550A también genera y envía una notificación de mantenimiento horizontal aguas abajo. En operación, las unidades de procesamiento aguas abajo que reciben la notificación de mantenimiento horizontal la destinan a uno o más de los usos (no mostrados): 1) puede haber múltiples solicitudes de mantenimiento que se propagan aguas arriba y se envía notificación de mantenimiento desde el nodo que mantendrá los datos. En este caso, si 560A hubiera recibido una solicitud de mantenimiento durante el ciclo 1, la notificación de mantenimiento desde 558A señalizaría al nodo 560A que algún nodo aguas arriba estaba manteniendo esos datos, por lo que 560A no necesitaría hacerlo. 2) la notificación de mantenimiento desde un nodo también afecta si se realiza una multiplicación.

Continuando la ejecución en el ciclo 2 550B, se producen multiplicaciones-acumulaciones en las unidades de procesamiento 554B, 556B y 560B debido a direcciones coincidentes entre elementos entrantes con dirección de 5, entre elementos registrados con dirección de 4, y entre un elemento entrante y registrado con dirección de 4, respectivamente.

Continuando la ejecución en el ciclo 3 550C, se producen multiplicaciones-acumulaciones en los cuatro nodos porque existe una coincidencia de dirección en cada caso entre elementos entrantes (como en 554C, 558C y 560C) o entre elementos registrados (como en 556C). Obsérvese que la unidad de procesamiento 554C realiza una multiplicación-

acumulación en elementos de datos entrantes, y la unidad de procesamiento 556C realiza la multiplicación-acumulación en elementos de datos registrados. En algunas realizaciones, como se muestra, el circuito de ejecución, en respuesta a la determinación de una coincidencia de dirección entre elementos horizontales y verticales, realiza la multiplicación-acumulación en elementos registrados que tienen una dirección de 5 en 556C, y, a continuación, realiza la multiplicación-acumulación de los elementos entrantes que tienen una dirección de 6 durante el siguiente ciclo.

Continuando la ejecución en el ciclo 4 550D, se producen multiplicaciones-acumulaciones en 550D porque existe una coincidencia de dirección entre elementos entrantes (como en 554D) o registrados (como en 556D).

La **Figura 6** es un diagrama de flujo de bloques que ilustra un procesador que ejecuta una multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones. Como se muestra, un procesador que ejecuta el flujo 600 en respuesta a una instrucción de VFVSMM en 602 es para extraer, usando circuitería de extracción, una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM) que tiene campos para especificar ubicaciones donde se almacenan cada una de A, B y C matrices que tienen elementos (M x K), (K x N) y (M x N), respectivamente. En 604, el procesador ha de decodificar, usando circuitos de decodificación, la instrucción de multiplicación de matrices extraída. En algunas realizaciones, en 606, el procesador ha de recuperar elementos de datos asociados con las matrices A y B especificadas. La operación 606 es opcional, como se indica por su borde de línea discontinua, en la medida en que los elementos de datos pueden recuperarse en un tiempo diferente o no recuperarse en absoluto. En 608, el procesador ha de responder a la instrucción de VFVSMM decodificada, usando circuitería de ejecución que opera en un modo denso-denso, enrutando cada fila de la matriz A especificada, escalonando filas posteriores, a una fila correspondiente de una matriz de procesamiento que tiene unidades de procesamiento (M x N), y enrutando cada columna de la matriz B especificada, escalonando columnas posteriores, a una columna correspondiente de la matriz de procesamiento. En algunas realizaciones, no mostradas, el procesador ha de enrutar cada fila de la matriz A especificada y cada columna de la matriz B especificada a una tasa de un elemento por ciclo de reloj y escalonar cada fila posterior y columna posterior en un ciclo de reloj, y cada una de las unidades de procesamiento (M x N) es para inferir direcciones de columna y fila de cada elemento de matriz A y matriz B recibido basándose en un ciclo de reloj y en una posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento. En algunas realizaciones, no mostradas, el procesador mantiene un recuento de elementos o índice de elementos, en donde cada una de las unidades de procesamiento se refiere al recuento de elementos, en lugar de inferir nada. En algunas realizaciones, no mostradas, cada uno de los elementos de matriz A y elementos de matriz B incluye un campo para especificar su posición lógica dentro de las matrices A o B. En 610, el procesador ha de generar, por cada una de las unidades de procesamiento (M x N), K productos de elementos de matriz A y matriz B coincidentes recibidos de las matrices A y B especificadas, respectivamente, para que exista una coincidencia cuando el elemento de matriz B tiene una misma dirección de fila que una dirección de columna del elemento de matriz A, y para acumular cada producto generado con un elemento correspondiente de la matriz C especificada que tiene una misma posición relativa que una posición de la unidad de procesamiento en la matriz de procesamiento.

La **Figura 7** es un diagrama de bloques que ilustra un circuito de multiplicación acumulación de números enteros/coma flotante de precisión variable, de acuerdo con algunas realizaciones. Como se muestra, el circuito de multiplicación-acumulación 700 incluye el multiplicador de FP16/INT16/INT8 701 y el acumulador de FP32/INT48/INT24 702.

En operación, el multiplicador de FP16/INT16/INT8 701 ha de reconfigurarse entre números enteros 8b/16b y entradas de coma flotante 16b para soportar diferentes requisitos de rendimiento, rango numérico y precisión. Además, los acumuladores anchos permiten resultados más precisos para matrices de alta dimensión. El MAC se organiza como una canalización de dos ciclos de multiplicación seguida de acumulación. Cuatro multiplicadores 8b 703A-703D, cada uno reconfigurable independientemente con signo/sin signo por entrada, entregan resultados 16b de 4 vías y 32b de 1 vía en modos INT8 e INT16, respectivamente. El resultado 32b usa cada uno de estos como un cuadrante de multiplicación 16b, con los resultados sumados usando la significación correcta. Las operaciones de coma flotante mapean la multiplicación de mantisa 11b con el multiplicador 16b, con toda la mantisa 22b no normalizada enviada a la etapa de sumador posterior junto con un exponente de producto de precisión sencilla. El producto no normalizado requiere únicamente un bit de guarda adicional para la adición de coma flotante, mientras se elimina la normalización de la ruta crítica de multiplicación. El resultado de multiplicación de FP16 es completamente representable por el rango de producto de FP32 y la precisión, lo que permite la eliminación de la detección de sub/desbordamiento y la lógica de saturación, así como cualquier lógica de redondeo en el multiplicador.

La latencia del sumador de FP32 afecta directamente al caudal de MAC ya que no puede canalizarse para acumulación consecutiva. En lugar de un detector de cero inicial posterior a la adición (LZD), se usa un anticipador de cero inicial (LZA 704) en paralelo con el sumador 32b para calcular los desplazamientos a la izquierda de la normalización. Puede ser necesario negar la salida del sumador para que las operaciones de resta produzcan una mantisa sin signo. La criticidad del MSB sumador de llegada tardía se oculta retardando esta etapa de negación hasta después de la normalización. La negación de complemento a dos también requiere un incremento después de la operación de inversión. Esto se fusiona con el incrementador de redondeo final para eliminarlo de la ruta crítica. El producto de número entero 32b se acumula con el sumador de mantisa para reconfiguración eficiente de área, mientras que los productos 16b requieren dos sumadores 16b adicionales para un caudal de 4 vías. La acumulación de números enteros amplia usa cuatro incrementadores/decrementadores 8b para los bits superiores, dos de los cuales pueden

reconfigurarse para operar como una unidad 16b para la acumulación 48b. Los multiplexores de derivación reducen la ruta crítica cuando se reconfigura la ruta de datos de mantisa para el modo de número entero. Las puertas de aislamiento de ruta de datos colocadas de manera óptima y la sincronización de reloj basada en modo aseguran que no haya actividad de conmutación en nodos lógicos y de reloj no usados. Añadir soporte de INT48 al acumulador de FP32 aumenta el área en un 20 %, mientras que el soporte de INT24 de 4 vías aumenta esto en un 8 % incremental.

La **Figura 8A** es un diagrama de bloques que ilustra un formato para una instrucción de multiplicación de matrices de dispersión variable de formato variable (VFVSMM), de acuerdo con algunas realizaciones. Como se muestra, la instrucción de VFVSMM 800 incluye el código de operación 801 (VFVSMM*) y campos para especificar las matrices de destino 802, de origen 1803 y de origen 2804. Como se usa en el presente documento, las matrices de origen 1, de origen 2 y destino se denominan en ocasiones como las matrices A, B y C, respectivamente. La instrucción de VFVSMM 800 incluye además campos opcionales para especificar el formato de datos 805, tal como número entero, coma flotante de precisión media, coma flotante de precisión sencilla o coma flotante de precisión doble, y tamaño de elemento 806, en términos de un número de bits por elemento de matriz. El formato de datos 805 puede incluso especificar un formato personalizado dependiente de la implementación. La instrucción de VFVSMM 800 en ocasiones incluye campos para especificar M 807, N 808 y K 809, donde las matrices A, B y C especificadas tienen elementos de datos (M x K), (K x N) y (M x N), respectivamente. Como se indica por sus bordes de línea discontinua, el formato de datos 805, el tamaño de elemento 806, M 807, N 808 y K 809 son opcionales, en la medida en que pueden omitirse, asumiendo de este modo valores por defecto predeterminados. En algunas realizaciones, uno o más del formato de datos 805, el tamaño de elemento 806, M 807, N 808 y K 809 se especifican como parte del código de operación 801, por ejemplo, como un código seleccionado para el código de operación, un sufijo o un prefijo. Por ejemplo, el código de operación 801 puede incluir un sufijo, tal como "B", "W", "D" o "Q", para especificar un tamaño de elemento de ocho, dieciséis, treinta y dos o sesenta y cuatro bits, respectivamente. Se muestra que el código de operación 801 incluye un asterisco para indicar que, opcionalmente, puede incluir prefijos o sufijos adicionales para especificar comportamientos de instrucción. Si la instrucción de VFVSMM 800 no especifica ninguno de los parámetros opcionales, se aplican valores por defecto predeterminados según sea necesario. El formato de la instrucción de VFVSMM 800 se ilustra y describe con más detalle con respecto a las Figuras 8B-C y

las Figuras 9A-D.

CONJUNTOS DE INSTRUCCIONES

Un conjunto de instrucciones puede incluir uno o más formatos de instrucciones. Un formato de instrucciones dado puede definir diversos campos (por ejemplo, número de bits, ubicación de los bits) para especificar, entre otras cosas, la operación a realizar (por ejemplo, código de operación) y los operandos en los que se realizará esa operación y/u otro campo o campos de datos (por ejemplo, máscara). Algunos formatos de instrucción se desglosan aún más mediante la definición de plantillas de instrucciones (o subformatos). Por ejemplo, las plantillas de instrucciones de un formato de instrucciones dado pueden definirse para tener diferentes subconjuntos de los campos del formato de instrucciones (los campos incluidos típicamente están en el mismo orden, pero al menos algunos tienen posiciones de bits diferentes porque hay menos campos incluidos) y/o definirse para tener un campo dado interpretado de manera diferente. Por lo tanto, cada instrucción de una ISA se expresa usando un formato de instrucción dado (y, si se define, en una de las plantillas de instrucción de ese formato de instrucción) e incluye campos para especificar la operación y los operandos. Por ejemplo, una instrucción ADD (AÑADIR) de ejemplo tiene un código de operación específico y un formato de instrucción que incluye un campo de código de operación para especificar ese código de operación y campos de operando para seleccionar operandos (fuente1/destino y fuente2); y una aparición de esta instrucción ADD en un flujo de instrucciones tendrá contenidos específicos en los campos de operando que seleccionan operandos específicos. Se ha lanzado y/o publicado un conjunto de extensiones de SIMD denominadas Extensiones Vectoriales Avanzadas (AVX) (AVX1 y AVX2) y que usan el esquema de codificación de Extensiones Vectoriales (VEX) (por ejemplo, véase Manual del desarrollador de Software de Arquitecturas Intel® 64 e IA-32, septiembre de 2014; y véase la referencia de programación de extensiones vectoriales avanzadas Intel®, octubre de 2014).

FORMATOS DE INSTRUCCIONES ILUSTRATIVOS

Las realizaciones de las instrucciones descritas en el presente documento pueden realizarse en diferentes formatos. Además, a continuación, se detallan sistemas, arquitecturas y canales ilustrativos. Se pueden ejecutar realizaciones de la instrucción o instrucciones en tales sistemas, arquitecturas y canalizaciones, pero no se limitan a las detalladas.

FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTORES GENÉRICO

Un formato de instrucción compatible con vectores es un formato de instrucción adecuado para instrucciones vectoriales (por ejemplo, hay ciertos campos específicos para operaciones vectoriales). Si bien se describen realizaciones en las que tanto las operaciones vectoriales como las escalares están soportadas a través del formato de instrucción compatible con vectores, realizaciones alternativas usan únicamente operaciones vectoriales en el formato de instrucción compatible con vectores.

Las **Figuras 8B-8C** son diagramas de bloques que ilustran un formato de instrucción compatible con vectores genérico y plantillas de instrucciones del mismo de acuerdo con realizaciones de la invención. La **Figura 8B** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores genérico y plantillas de instrucción de clase A del mismo de acuerdo con realizaciones de la invención; mientras que la **Figura 8C** es un diagrama de bloques que ilustra el formato de instrucción compatible con vectores genérico y sus plantillas de instrucción de clase B de acuerdo con realizaciones de la invención. Específicamente, un formato de instrucción compatible con vectores genérico 811 para el que se definen plantillas de instrucciones de clase A y clase B, ambas de las cuales incluyen plantillas de instrucciones sin acceso a memoria 812 y plantillas de instrucciones de acceso a memoria 820. El término genérico en el contexto del formato de instrucción compatible con vectores se refiere al formato de instrucciones que no está vinculado a ningún conjunto de instrucciones específico.

Si bien se describirán realizaciones de la invención en las que el formato de instrucción compatible con vectores soporta lo siguiente: una longitud (o tamaño) de operando de vector de 64 bytes con anchuras (o tamaños) de elementos de datos de 32 bits (4 bytes) o 64 bits (8 bytes) (y, por lo tanto, un vector de 64 bytes consiste en 16 elementos de tamaño de palabra doble o, como alternativa, 8 elementos de tamaño de palabra cuádruple); una longitud (o tamaño) de operando de vector de 64 bytes con anchuras (o tamaños) de elementos de datos de 16 bits (2 bytes) u 8 bits (1 byte); una longitud (o tamaño) de operando de vector de 32 bytes con anchuras (o tamaños) de elementos de datos de 32 bits (4 bytes), 64 bits (8 bytes), 16 bits (2 bytes) u 8 bits (1 byte); y una longitud (o tamaño) de operando de vector de 16 bytes con anchuras (o tamaños) de elementos de datos de 32 bits (4 bytes), 64 bits (8 bytes), 16 bits (2 bytes) u 8 bits (1 byte); realizaciones alternativas pueden soportar más, menos y/o diferentes tamaños de operandos vectoriales (por ejemplo, operandos vectoriales de 256 bytes) con más, menos o diferentes anchuras de elementos de datos (por ejemplo, anchuras de elementos de datos de 128 bits (16 bytes)).

Las plantillas de instrucciones de clase A en la **Figura 8B** incluyen: 1) dentro de las plantillas de instrucciones sin acceso a memoria 812 se muestra una plantilla de instrucciones de operación de tipo control de redondeo completo sin acceso a memoria 813 y una plantilla de instrucciones de operación de tipo transformación de datos sin acceso a memoria 815; y 2) dentro de las plantillas de instrucciones de acceso a memoria 820 se muestra una plantilla de instrucciones de acceso a memoria temporal 825 y una plantilla de instrucciones de acceso a memoria no temporal 830. Las plantillas de instrucción de clase B en la **Figura 8B** incluyen: 1) dentro de las plantillas de instrucciones sin acceso a memoria 812 se muestra una plantilla de instrucciones de operación de tipo de control de redondeo parcial de control de máscara de escritura sin acceso a memoria 814 y una plantilla de instrucciones de operación de tipo vsize de control de máscara de escritura sin acceso a memoria 817; y 2) dentro de las plantillas de instrucciones de acceso a memoria 820 se muestra una plantilla de instrucciones de control de máscara de escritura de acceso a memoria 827.

El formato de instrucción compatible con vectores genérico 811 incluye los siguientes campos enumerados a continuación en el orden ilustrado en las **Figuras 8B-8C.**

Campo de formato 840 - un valor específico (un valor de identificador de formato de instrucción) en este campo identifica de forma única el formato de instrucción compatible con vectores y, por lo tanto, las ocurrencias de instrucciones en el formato de instrucción compatible con vectores en los flujos de instrucciones. Como tal, este campo es opcional en el sentido de que no es necesario para un conjunto de instrucciones que únicamente tiene el formato de instrucción compatible con vectores genérico.

Campo de operación de base 842 - su contenido distingue diferentes operaciones de base.

Campo de índice de registro 844 - su contenido, directamente o a través de la generación de direcciones, especifica las ubicaciones de los operandos de origen y destino, ya sea en los registros o en la memoria. Estos incluyen un número suficiente de bits para seleccionar N registros de un archivo de registros PxQ (por ejemplo, 32x512, 16x128, 32x1024, 64x1024). Mientras que en una realización N puede ser hasta tres registros de origen y uno de destino, realizaciones alternativas pueden soportar más o menos registros de origen y de destino (por ejemplo, pueden soportar hasta dos orígenes donde uno de estos orígenes también actúa como el destino, pueden soportar hasta tres orígenes donde uno de estos orígenes también actúa como el destino, puede soportar hasta dos orígenes y un destino).

Campo modificador 846: su contenido distingue apariciones de instrucciones en el formato de instrucción de vectores genérico que especifican acceso a memoria de aquellas que no lo hacen; es decir, entre plantillas de instrucciones sin acceso a memoria 812 y plantillas de instrucciones con acceso a memoria 820. Las operaciones de acceso a memoria leen y/o escriben en la jerarquía de memoria (especificando, en algunos casos, las direcciones de origen y/o de destino usando valores en registros), mientras que las operaciones sin acceso a memoria no lo hacen (por ejemplo, el origen y los destinos son registros). Si bien en una realización este campo también selecciona entre tres formas diferentes de realizar cálculos de direcciones de memoria, realizaciones alternativas pueden soportar más, menos o diferentes formas de realizar cálculos de direcciones de memoria.

Campo de operación de aumento 850 - su contenido distingue cuál de una diversidad de operaciones diferentes se realizará además de la operación de base. Este campo es específico del contexto. En una realización de la invención, este campo se divide en un campo de clase 868, un campo alfa 852 y un campo beta 854. El campo de operación de

aumento 850 permite que se realicen grupos comunes de operaciones en una única instrucción en lugar de 2, 3 o 4 instrucciones.

Campo de escala 860: su contenido permite el escalado del contenido del campo de índice para la generación de direcciones de memoria (por ejemplo, para la generación de direcciones que usa 2^{escala} * índice + base).

Campo de desplazamiento 862A - su contenido se usa como parte de la generación de direcciones de memoria (por ejemplo, para la generación de direcciones que usa 2^{escala} * índice + base + desplazamiento).

Campo de factor de desplazamiento 862B (obsérvese que la yuxtaposición del campo de desplazamiento 862A directamente sobre el campo de factor de desplazamiento 862B indica que se usa uno u otro) - su contenido se usa como parte de la generación de direcciones; especifica un factor de desplazamiento a escalar de acuerdo con el tamaño de un acceso a memoria (N), donde N es el número de bytes en el acceso a memoria (por ejemplo, para la generación de direcciones que utiliza 2^{escala} * índice + base + desplazamiento escalado). Los bits redundantes de bajo orden se ignoran y, por lo tanto, el contenido del campo del factor de desplazamiento se multiplica por el tamaño total de los operandos de memoria (N) para generar el desplazamiento final que se usará para calcular una dirección efectiva. El valor de N está determinado por el hardware del procesador en tiempo de ejecución basándose en el campo de código de operación completo 874 (descrito más adelante en el presente documento) y el campo de manipulación de datos 854C. El campo de desplazamiento 862A y el campo de factor de desplazamiento 862B son opcionales en el sentido de que no se usan para las plantillas de instrucciones sin acceso a memoria 812 y/o diferentes realizaciones pueden implementar únicamente uno o ninguno de los dos.

Campo de anchura de elemento de datos 864 - su contenido distingue cuál de un número de anchuras de elementos de datos va a usarse (en algunas realizaciones, para todas las instrucciones; en otras realizaciones, solo para algunas de las instrucciones). Este campo es opcional en el sentido de que no es necesario si únicamente se soporta una anchura de elemento de datos y/o se soportan anchuras de elementos de datos usando algún aspecto de los códigos de operación.

Campo de máscara de escritura 870 - su contenido controla, en una base de posición de elemento de datos, si esa posición de elemento de datos en el operando del vector de destino refleja el resultado de la operación de base y la operación de aumento. Las plantillas de instrucciones de clase A soportan la fusión de máscaras de escritura, mientras que las plantillas de instrucciones de clase B soportan tanto la fusión como la puesta a cero de máscaras de escritura. Cuando se fusionan, las máscaras vectoriales permiten proteger de actualizaciones cualquier conjunto de elementos en el destino durante la ejecución de cualquier operación (especificada por la operación de base y la operación de aumento); en otra realización, conservando el valor antiguo de cada elemento del destino donde el bit de máscara correspondiente tiene un 0. Por el contrario, cuando las máscaras vectoriales de puesta a cero permiten poner a cero cualquier conjunto de elementos en el destino durante la ejecución de cualquier operación (especificada por la operación de base y la operación de aumento); en una realización, un elemento del destino se establece a 0 cuando el bit de máscara correspondiente tiene un valor 0. Un subconjunto de esta funcionalidad es la capacidad de controlar la longitud del vector de la operación que se realiza (es decir, el intervalo de elementos que se modifican, desde el primero hasta el último); sin embargo, no es necesario que los elementos que se modifican sean consecutivos. Por lo tanto, el campo de máscara de escritura 870 permite operaciones vectoriales parciales, que incluyen cargas, almacenamientos, aritméticas, lógicas, etc. Si bien se describen realizaciones de la invención en las que el contenido del campo de máscara de escritura 870 selecciona uno de un número de registros de máscara de escritura que contiene la máscara de escritura que se va a usar (y, por tanto, el contenido del campo de máscara de escritura 870 identifica indirectamente ese enmascaramiento que se va a realizar), realizaciones alternativas, en su lugar, o adicionales, permiten que el contenido del campo de escritura de máscara 870 especifique directamente el enmascaramiento que se va a realizar.

Campo inmediato 872 - su contenido permite la especificación de un inmediato. Este campo es opcional en el sentido de que no está presente en una implementación del formato compatible con vectores genéricos que no soporta inmediato y no está presente en instrucciones que no usan un inmediato.

Campo de clase 868 - su contenido distingue entre diferentes clases de instrucciones. Con referencia a las Figuras 8A-B, el contenido de este campo selecciona entre instrucciones de clase A y clase B. En las Figuras 8A-B, se usan cuadrados de esquinas redondeadas para indicar que un valor específico está presente en un campo (por ejemplo, clase A 868A y clase B 868B para el campo de clase 868 respectivamente en las Figuras 8A-B).

Plantillas de instrucción de clase A

En el caso de las plantillas de instrucción sin acceso a memoria de clase A 812, el campo alfa 852 se interpreta como un campo de RS 852A, cuyo contenido distingue cuál de los diferentes tipos de operación de aumento va a realizarse (por ejemplo, la redondeo 852A.1 y la transformada de datos 852A.2 se especifican respectivamente para las plantillas de instrucción de operación de tipo de redondeo sin acceso a memoria 810 y operación de tipo transformada de datos sin acceso a memoria 815), mientras que el campo beta 854 distingue cuál de las operaciones del tipo especificado

va a realizarse. En las plantillas de instrucciones sin acceso a memoria 812, el campo de escala 860, el campo de desplazamiento 862A y el campo de escala de desplazamiento 862B no están presentes.

PLANTILLAS DE INSTRUCCIONES SIN ACCESO A MEMORIA - OPERACIÓN DE TIPO DE CONTROL DE REDONDEO COMPLETO

En la plantilla de instrucción de operación de tipo de control de redondeo completo sin acceso a memoria 813, el campo beta 854 se interpreta como un campo de control de redondeo 854A, cuyo contenido o contenidos proporcionan redondeo estático. Mientras que, en las realizaciones descritas de la invención, el campo de control de redondeo 854A incluye un campo de supresión de todas las excepciones de coma flotante (SAE) 856 y un campo de control de operación de redondeo 858, las realizaciones alternativas pueden soportar la codificación de ambos conceptos en el mismo campo o solo tener uno o el otro de estos conceptos/campos (por ejemplo, pueden tener solo el campo de control de operación de redondeo 858).

Campo de SAE 856 - su contenido distingue si se debe o no deshabilitar el informe de eventos de excepción; cuando el contenido del campo de SAE 856 indica que la supresión está habilitada, una instrucción dada no informa ningún tipo de bandera de excepción de coma flotante y no genera ningún manejador de excepción de coma flotante.

Campo de control de operación de redondeo 858 - su contenido distingue cuál de un grupo de operaciones de redondeo se realizará (por ejemplo, redondeo hacia arriba, redondeo hacia abajo, redondeo hacia cero y redondeo hacia el valor más cercano). Por tanto, el campo de control de operación de redondeo 858 permite el cambio del modo de redondeo por instrucción. En una realización de la invención donde un procesador incluye un registro de control para especificar modos de redondeo, el contenido del campo de control de operación de redondeo 850 anula ese valor de registro.

PLANTILLAS DE INSTRUCCIÓN SIN ACCESO A MEMORIA - OPERACIÓN DE TIPO DE TRANSFORMACIÓN DE DATOS

En la plantilla de instrucciones de operación de tipo de transformación de datos sin acceso a memoria 815, el campo beta 854 se interpreta como un campo de transformación de datos 854B, cuyo contenido distingue cuál de un número de transformaciones de datos se va a realizar (por ejemplo, sin transformación de datos, mezcla, difusión).

En el caso de una plantilla de instrucción de acceso a memoria 820 de clase A, el campo alfa 852 se interpreta como un campo de sugerencia de desalojo 852B, cuyo contenido distingue cuál de las sugerencias de desalojo se va a usar (en la **Figura 8A**, temporal 852B.1 y no temporal 852B.2 se especifican respectivamente para la plantilla de instrucción de acceso a memoria, temporal 825 y la plantilla de instrucción de acceso a memoria, no temporal 830), mientras que el campo beta 854 se interpreta como un campo de manipulación de datos 854C, cuyo contenido distingue cuál de un número de operaciones de manipulación de datos (también conocidas como primitivas) se va a realizar (por ejemplo, sin manipulación; difusión; conversión ascendente de un origen y conversión descendente de un destino). Las plantillas de instrucción con acceso a memoria 820 incluyen el campo de escala 860 y, opcionalmente, el campo de desplazamiento 862A o el campo de escala de desplazamiento 862B.

Las instrucciones de memoria vectoriales realizan cargas y almacenes de vectores en la memoria, con soporte de conversión. Al igual que con las instrucciones vectoriales normales, las instrucciones de memoria vectorial transfieren datos desde/hacia la memoria en forma de elementos de datos, y los elementos que realmente se transfieren están dictados por el contenido de la máscara vectorial que se selecciona como máscara de escritura.

PLANTILLAS DE INSTRUCCIONES DE ACCESO A LA MEMORIA-TEMPORAL

Los datos temporales son datos que probablemente se reutilizarán lo suficientemente pronto como para beneficiarse del almacenamiento en caché. Sin embargo, esto es una sugerencia, y diferentes procesadores pueden implementarla de diferentes maneras, incluso ignorando la sugerencia por completo.

PLANTILLAS DE INSTRUCCIONES DE ACCESO A MEMORIA - NO TEMPORAL

Los datos no temporales son datos que es poco probable que se reutilicen lo suficientemente pronto como para beneficiarse del almacenamiento en la memoria caché en la caché de 1er nivel y se les debe dar prioridad para el desalojo. Sin embargo, esto es una sugerencia, y diferentes procesadores pueden implementarla de diferentes maneras, incluso ignorando la sugerencia por completo.

PLANTILLAS DE INSTRUCCIONES DE CLASE B

En el caso de las plantillas de instrucciones de clase B, el campo alfa 852 se interpreta como un campo de control de máscara de escritura (Z) 852C, cuyo contenido distingue si el enmascaramiento de escritura controlado por el campo de máscara de escritura 870 debe ser una fusión o una puesta a cero.

En el caso de las plantillas de instrucciones sin acceso a memoria 812 de clase B, parte del campo beta 854 se interpreta como un campo de RL 857A, cuyo contenido distingue cuál de los diferentes tipos de operación de aumento se va a realizar (por ejemplo, redondeo 857A.1 y la longitud de vector (VSIZE) 857A.2 se especifican respectivamente para la plantilla de instrucciones de operación de tipo de control de redondeo parcial de control de máscara de escritura sin acceso a memoria 814 y la plantilla de instrucciones sin de operación de tipo VSIZE de control de máscara de escritura sin acceso a memoria 817), mientras que el resto del campo beta 854 distingue cuál de las operaciones del tipo especificado se va a realizar. En las plantillas de instrucciones sin acceso a memoria 812, el campo de escala 860, el campo de desplazamiento 862A y el campo de escala de desplazamiento 862B no están presentes.

En la plantilla de instrucción de operación sin acceso a memoria, de control de máscara de escritura, tipo de control de redondeo parcial 810, el resto del campo beta 854 se interpreta como un campo de operación de redondeo 859A y el informe de eventos de excepción está inhabilitado (una instrucción dada no informa ninguna clase de bandera de excepción de coma flotante y no genera ningún manejador de excepción de coma flotante).

Campo de control de operación de redondeo 859A - al igual que el campo de control de operación de redondeo 858, su contenido distingue cuál de un grupo de operaciones de redondeo realizar (por ejemplo, redondeo hacia arriba, redondeo hacia abajo, redondeo hacia cero y redondeo hacia el valor más cercano). Por tanto, el campo de control de operación de redondeo 859A permite el cambio del modo de redondeo por instrucción. En una realización de la invención donde un procesador incluye un registro de control para especificar modos de redondeo, el contenido del campo de control de operación de redondeo 850 anula ese valor de registro.

En la plantilla de instrucciones de operación de tipo VSIZE de control de máscara de escritura sin acceso a memoria 817, el resto del campo beta 854 se interpreta como un campo de longitud de vector 859B, cuyo contenido distingue cuál de un número de longitudes de vector de datos se va a realizar en (por ejemplo, 128, 256 o 512 bytes).

En el caso de una plantilla de instrucción con acceso a memoria 820 de clase B, parte del campo beta 854 se interpreta como un campo de difusión 857B, cuyo contenido distingue si se va a realizar o no la operación de manipulación de datos de tipo difusión, mientras que el resto del campo beta 854 se interpreta como el campo de longitud vectorial 859B. Las plantillas de instrucción con acceso a memoria 820 incluyen el campo de escala 860 y, opcionalmente, el campo de desplazamiento 862A o el campo de escala de desplazamiento 862B.

Con respecto al formato de instrucción compatible con vectores genéricos 811, se muestra un campo de código de operación completo 874 que incluye el campo de formato 840, el campo de operación de base 842 y el campo de anchura de elemento de datos 864. Aunque se muestra una realización donde el campo de código de operación completo 874 incluye todos estos campos, el campo de código de operación completo 874 incluye menos de todos estos campos en realizaciones que no los soportan todos. El campo de código de operación completo 874 proporciona el código de operación (código de operación).

El campo de operación de aumento 850, el campo de anchura de elemento de datos 864 y el campo de máscara de escritura 870 permiten que estas características se especifiquen en una base por instrucción en el formato de instrucción compatible con vectores genérico.

La combinación de campo de máscara de escritura y campo de anchura de elemento de datos crea instrucciones escritas que permiten que la máscara se aplique basándose en diferentes anchuras de elementos de datos.

Las diversas plantillas de instrucciones que se encuentran dentro de la clase A y la clase B son beneficiosas en diferentes situaciones. En algunas realizaciones de la invención, diferentes procesadores o diferentes núcleos dentro de un procesador pueden soportar solo la clase A, solo la clase B o ambas clases. Por ejemplo, un núcleo en desorden de propósito general de alto rendimiento destinado a computación de propósito general puede soportar únicamente clase B, un núcleo destinado principalmente a gráficos y/o computación científica (rendimiento) puede soportar únicamente clase A, y un núcleo destinado a ambas puede soportar ambas (por supuesto, un núcleo que tiene alguna mezcla de plantillas e instrucciones de ambas clases, pero no todas las plantillas ni instrucciones de ambas clases están dentro del alcance de la invención). Además, un solo procesador puede incluir múltiples núcleos, todos los cuales soportan la misma clase o en los que diferentes núcleos soportan diferentes clases. Por ejemplo, en un procesador con gráficos y núcleos de propósito general separados, uno de los núcleos de gráficos destinado principalmente a gráficos y/o computación científica puede soportar únicamente la clase A, mientras que uno o más de los núcleos de propósito general pueden ser núcleos de propósito general de alto rendimiento con ejecución en desorden y cambio de nombre de registro destinado a computación de propósito general que soportan únicamente clase B. Otro procesador que no tiene un núcleo de gráficos separado, puede incluir uno más núcleos en orden o en desorden de propósito general que soportan tanto clase A como clase B. Por supuesto, las características de una clase también pueden implementarse en la otra clase en diferentes realizaciones de la invención. Los programas escritos en un lenguaje de alto nivel se pondrían (por ejemplo, compilados justo a tiempo o compilados estáticamente) en una diversidad de formas ejecutables diferentes, que incluyen: 1) una forma que tiene únicamente instrucciones de la clase o clases soportadas por el procesador objetivo para su ejecución; o 2) una forma que tiene rutinas alternativas escritas usando diferentes combinaciones de las instrucciones de todas las clases y que tiene un código de flujo de control

que selecciona las rutinas a ejecutar basándose en las instrucciones soportadas por el procesador que actualmente está ejecutando el código.

FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTORES ESPECÍFICO ILUSTRATIVO

La **Figura 9A** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores específico ilustrativo de acuerdo con realizaciones de la invención. La **Figura 9A** muestra un formato de instrucción compatible con vectores específico 900 que es específico en el sentido de que especifica la ubicación, el tamaño, la interpretación y el orden de los campos, así como los valores para algunos de esos campos. El formato de instrucciones específico compatible con vectores 900 se puede usar para ampliar el conjunto de instrucciones x86 y, por tanto, algunos de los campos son similares o iguales a aquellos usados en el conjunto de instrucciones x86 existente y la extensión del mismo (por ejemplo, AVX). Este formato sigue siendo consistente con el campo de codificación de prefijo, el campo de bytes de código de operación real, el campo MOD R/M, el campo SIB, el campo de desplazamiento y los campos inmediatos del conjunto de instrucciones x86 existente con extensiones. Se ilustran los campos de la **Figura 8** en el que se mapean los campos de la **Figura 9A**.

Debe entenderse que, aunque las realizaciones de la invención se describen con referencia al formato de instrucción compatible con vectores específico 900 en el contexto del formato de instrucción compatible con vectores genérico 811 con propósitos ilustrativos, la invención no se limita al formato de instrucción compatible con vectores específico 900 excepto cuando se reivindica. Por ejemplo, el formato de instrucción compatible con vectores genérico 811 contempla una diversidad de tamaños posibles para los diversos campos, mientras que el formato de instrucción compatible con vectores específico 900 se muestra teniendo campos de tamaños específicos. A modo de ejemplo específico, mientras que el campo de anchura de elemento de datos 864 se ilustra como un campo de un bit en el formato de instrucción compatible con vectores específico 900, la invención no está así limitada (es decir, el formato de instrucción compatible con vectores genérico 811 contempla otros tamaños del campo de anchura de elemento de datos 864).

El formato de instrucción compatible con vectores genérico 811 incluye los siguientes campos enumerados a continuación en el orden ilustrado en la **Figura 9A.**

Prefijo EVEX (Bytes 0-3) 902: está codificado en formato de cuatro bytes.

Campo de formato 840 (Byte 0 de EVEX, bits [7:0]) - el primer byte (Byte 0 de EVEX) es el campo de formato 840 y contiene 0x62 (el valor único usado para distinguir el formato de instrucción compatible con vectores en una realización del invención).

El segundo-cuarto de bytes (Bytes de EVEX 1-3) incluyen un número de campos de bits que proporcionan una capacidad específica.

Campo REX 905 (Byte de EVEX 1, bits [7-5]): consta de un campo de bits EVEX.R (Byte de EVEX 1, bit [7] - R), campo de bits EVEX.X (byte de EVEX 1, bit [6] - X), y 857BEX byte 1, bit[5] - B). Los campos de bits EVEX.R, EVEX.X y EVEX.B proporcionan la misma funcionalidad que los campos de bits VEX correspondientes y se codifican en forma de complemento a 1, es decir, ZMM0 se codifica como 1111B, ZMM15 se codifica como 0000B. Otros campos de las instrucciones codifican los tres bits inferiores de los índices de registro como es conocido en la técnica (rrr, xxx y bbb), de modo que Rrrr, Xxxx y Bbbb pueden formarse sumando EVEX.R, EVEX.X, y EVEX.B.

REX' 910A - esta es la primera parte del campo REX' 910 y es el campo de bits EVEX.R' (byte de EVEX 1, bit [4] - R') que se usa para codificar los 16 superiores o los 16 inferiores del conjunto de 32 registros extendido. En una realización de la invención, este bit, junto con otros como se indica a continuación, se almacena en formato de bits invertidos para distinguir (en el bien conocido modo x86 de 32 bits) de la instrucción BOUND, cuyo byte de código de operación real es 62, pero no acepta en el campo MOD R/M (descrito a continuación) el valor de 11 en el campo MOD; realizaciones alternativas de la invención no almacenan este y los otros bits indicados a continuación en el formato invertido. Se usa un valor de 1 para codificar los 16 registros inferiores. En otras palabras, R'Rrrr se forma combinando EVEX.R', EVEX.R y el otro RRR de otros campos.

Campo de mapa de código de operación 915 (byte 1 de EVEX, bits [3:0] - mmmm) - su contenido codifica un byte de código de operación inicial implícito (0F, 0F 38 o 0F 3).

El campo de anchura de elemento de datos 864 (byte de EVEX 2, bit [7] - W) - se representa mediante la notación EVEX.W. EVEX.W se usa para definir la granularidad (tamaño) del tipo de datos (ya sean elementos de datos de 32 bits o elementos de datos de 64 bits).

EVEX.vvvv 920 (Byte de EVEX 2, bits [6:3]-vvvv)- la función de EVEX.vvvv puede incluir lo siguiente: 1) EVEX.vvvv codifica el primer operando de registro de origen, especificado en forma invertida (complemento a 1) y es válido para instrucciones con 2 o más operandos de origen; 2) EVEX.vvvv codifica el operando de registro de destino, especificado en forma de complemento a 1 para ciertos desplazamientos de vector; o 3) EVEX.vvvv no codifica ningún operando,

el campo está reservado y debe contener 1111b. Por tanto, el campo 920 de EVEX.vvvv codifica los 4 bits de orden inferior del primer especificador de registro de origen almacenado en forma invertida (complemento a 1s). Dependiendo de la instrucción, se usa un campo de bits EVEX adicional diferente para extender el tamaño del especificador a 32 registros.

EVEX.U Campo de clase 868 (Byte de EVEX 2, bit [2]-U) - Si EVEX.U = 0, indica clase A o EVEX.U0; si EVEX.U = 1, indica clase B o EVEX.U1.

Campo de codificación de prefijo 925 (byte de EVEX 2, bits [1:0] - pp) - proporciona bits adicionales para el campo de operación de base. Además de proporcionar soporte para las instrucciones SSE heredadas en el formato de prefijo EVEX, esto también tiene la ventaja de compactar el prefijo de SIMD (en lugar de requerir un byte para expresar el prefijo de SIMD, el prefijo de EVEX requiere solo 2 bits). En una realización, para soportar instrucciones SSE heredadas que usan un prefijo SIMD (66H, F2H, F3H) tanto en el formato heredado como en el formato de prefijo EVEX, estos prefijos SIMD heredados se codifican en el campo de codificación de prefijo SIMD; y en tiempo de ejecución se expanden al prefijo SIMD heredado antes de proporcionarse al PLA del decodificador (para que el PLA pueda ejecutar tanto el formato heredado como EVEX de estas instrucciones heredadas sin modificación). Aunque las instrucciones más nuevas podrían usar el contenido del campo de codificación del prefijo EVEX directamente como una extensión del código de operación, ciertas realizaciones se expanden de manera similar para mantener la consistencia, pero permiten que estos prefijos de SIMD heredados especifiquen diferentes significados. Una realización alternativa puede rediseñar el PLA para soportar las codificaciones de prefijo de SIMD de 2 bits y, por lo tanto, no requerir la expansión.

Campo alfa 852 (byte de EVEX 3, bit [7] - EH; también conocido como EVEX.EH, EVEX.rs, EVEX.RL, EVEX.control de máscara de escritura y EVEX.N; también ilustrado con α) - como se describió anteriormente, este campo es específico del contexto.

Campo beta 854 (byte de EVEX 3, bits [6:4]-SSS, también conocido como EVEX.s_{2-0}, EVEX.r_{2-0}, EVEX.rr1, EVEX.LL0, EVEX.LLB; también ilustrado con βββ) - como se describió anteriormente, este campo es específico del contexto.

REX' 910B - este es el resto del campo REX' 910 y es el campo de bits EVEX.V' (Byte de EVEX 3, bit [3] - V') que se puede usar para codificar los 16 superiores o los 16 inferiores del conjunto de 32 registros extendido. Este bit se almacena en formato de bits invertidos. Se usa un valor de 1 para codificar los 16 registros inferiores. En otras palabras, V'VVVV se forma combinando EVEX.V', EVEX.vvvv.

Campo de máscara de escritura 870 (byte de EVEX 3, bits [2:0]-kkk) - su contenido especifica el índice de un registro en los registros de máscara de escritura como se describió anteriormente. En una realización de la invención, el valor específico EVEX.kkk=000 tiene un comportamiento especial que implica que no se usa una máscara de escritura para la instrucción particular (esto se puede implementar de una diversidad de formas, incluyendo el uso de una máscara de escritura programada a todos o al hardware que sortea el hardware de enmascaramiento).

El campo de código de operación real 930 (byte 4) también se conoce como byte de código de operación. Parte del código de operación se especifica en este campo.

El campo MOD R/M 940 (byte 5) incluye el campo MOD 942, el campo Reg 944 y el campo R/M 946. Como se describió anteriormente, el contenido del campo MOD 942 distingue entre operaciones de acceso a memoria y operaciones sin acceso a memoria. La función del campo Reg 944 se puede resumir en dos situaciones: codificar el operando del registro de destino o un operando del registro de origen, o tratarse como una extensión de código de operación y no usarse para codificar un operando de instrucción. La función del campo R/M 946 puede incluir lo siguiente: codificar el operando de instrucción que hace referencia a una dirección de memoria, o codificar el operando del registro de destino o un operando del registro de origen.

Byte de escala, índice, base (SIB) (Byte 6): como se describió anteriormente, el contenido el campo de escala 850 se utiliza para la generación de direcciones de memoria. SIB.xxx 954 y SIB.bbb 956 - el contenido de estos campos ha sido mencionado anteriormente con respecto a los índices de registro Xxxx y Bbbb.

Campo de desplazamiento 862A (Bytes 7-10) - cuando el campo MOD 942 contiene 10, los bytes 7-10 son el campo de desplazamiento 862A, y funciona igual que el desplazamiento de 32 bits heredado (disp32) y funciona con granularidad de byte.

Campo de factor de desplazamiento 862B (Byte 7) - cuando el campo MOD 942 contiene 01, el byte 7 es el campo de factor de desplazamiento 862B. La ubicación de este campo es la misma que la del desplazamiento de 8 bits (disp8) del conjunto de instrucciones x86 heredado, que funciona con granularidad de bytes. Dado que disp8 es un signo extendido, solo puede direccionar entre -128 y 127 compensaciones de bytes; en términos de líneas de caché de 64 bytes, disp8 usa 8 bits que se pueden establecer en solo cuatro valores realmente útiles: -128, -64, 0 y 64; dado que a menudo se necesita un rango mayor, se usa disp32; sin embargo, disp32 requiere 4 bytes. A diferencia de disp8 y disp32, el campo de factor de desplazamiento 862B es una reinterpretación de disp8; cuando se usa el campo de

factor de desplazamiento 862B, el desplazamiento real se determina por el contenido del campo de factor de desplazamiento multiplicado por el tamaño del acceso de operando de memoria (N). Este tipo de desplazamiento se denomina disp8*N. Esto reduce la longitud de instrucción promedio (un solo byte de lo usado para el desplazamiento, pero con un rango mucho mayor). Tal desplazamiento comprimido se basa en la suposición de que el desplazamiento efectivo es un múltiplo de la granularidad del acceso a memoria y, por lo tanto, no es necesario codificar los bits de bajo orden redundantes de la compensación de dirección. En otras palabras, el campo de factor de desplazamiento 862B sustituye el desplazamiento de 8 bits del conjunto de instrucciones x86 heredado. Por tanto, el campo de factor de desplazamiento 862B se codifica de la misma manera que un desplazamiento de 8 bits del conjunto de instrucciones x86 (por lo que no hay cambios en las reglas de codificación ModRM/SIB) con la única excepción de que disp8 se sobrecarga a disp8*N. En otras palabras, no hay cambios en las reglas de codificación o longitudes de codificación, sino solo en la interpretación del valor de desplazamiento por el hardware (que necesita escalar el desplazamiento por el tamaño del operando de memoria para obtener una compensación de dirección por bytes). El campo inmediato 872 funciona como se ha descrito anteriormente.

CAMPO DE CÓDIGO DE OPCIÓN COMPLETO

La **Figura 9B** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico 900 que componen el campo de código de operación completo 874 de acuerdo con una realización de la invención. Específicamente, el campo de código de operación completo 874 incluye el campo de formato 840, el campo de operación de base 842 y el campo de anchura de elemento de datos (W) 864. El campo de operación de base 842 incluye el campo de codificación de prefijo 925, el campo de mapa de código de operación 915 y el campo de código de operación real 930.

CAMPO DE ÍNDICE DE REGISTRO

La **Figura 9C** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico 900 que componen el campo de índice de registro 844 de acuerdo con una realización de la invención. Específicamente, el campo de índice de registro 844 incluye el campo REX 905, el campo REX' 910, el campo MODR/M.reg 944, el campo MODR/M.r/m 946, el campo VVVV 920, el campo xxx 954 y el campo bbb 956.

CAMPO DE OPERACIÓN DE AUMENTO

La **Figura 9D** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico 900 que componen el campo de operación de aumento 850 de acuerdo con una realización de la invención. Cuando el campo de clase (U) 868 contiene 0, significa EVEX.U0 (clase A 868A); cuando contiene 1, significa EVEX.U1 (clase B 868B). Cuando U=0 y el campo MOD 942 contiene 11 (lo que significa una operación sin acceso a memoria), el campo alfa 852 (byte de EVEX 3, bit [7] - EH) se interpreta como el campo de rs 852A. Cuando el campo de RS 852A contiene un 1 (redondeo 852A.1), el campo beta 854 (byte de EVEX 3, bits [6:4]-SSS) se interpreta como el campo de control de redondeo 854A. El campo de control de redondeo 854A incluye un campo de SAE de un bit 856 y un campo de operación de redondeo de dos bits 858. Cuando el campo rs 852A contiene un 0 (transformación de datos 852A.2), el campo beta 854 (byte de EVEX 3, bits [6:4]-SSS) se interpreta como un campo de transformación de datos de tres bits 854B. Cuando U=0 y el campo MOD 942 contiene 00, 01 o 10 (lo que significa una operación de acceso a memoria), el campo alfa 852 (byte de EVEX 3, bit [7] - EH) se interpreta como el campo de sugerencia de desalojo (EH) 852B y el campo beta 854 (byte de EVEX 3, bits [6:4]-SSS) se interpreta como un campo de manipulación de datos de tres bits 854C.

Cuando U=1, el campo alfa 852 (byte de EVEX 3, bit [7] - EH) se interpreta como el campo de control de máscara de escritura (Z) 852C. Cuando U=1 y el campo MOD 942 contiene 11 (lo que significa una operación sin acceso a memoria), parte del campo beta 854 (Byte de EVEX 3, bit [4]- S_0) se interpreta como el campo de RL 857A; cuando contiene un 1 (redondeo 857A.1) el resto del campo beta 854 (byte de EVEX 3, bit [6-5]- S_{2-1}) se interpreta como el campo de operación de redondeo 859A, mientras que cuando el campo de RL 857A contiene un 0 (VSIZE 857.A2) el resto del campo beta 854 (byte de EVEX 3, bit [6-5]- S_{2-1}) se interpreta como el campo de longitud de vector 859B (byte de EVEX 3, bit [6-5]- L_{1-0}). Cuando U=1 y el campo MOD 942 contiene 00, 01 o 10 (lo que significa una operación con acceso a memoria), el campo beta 854 (byte de EVEX 3, bits [6:4]- SSS) se interpreta como el campo de longitud de vector 859B (byte de EVEX 3, bit [6-5]- L_{1-0}) y el campo de difusión 857B (byte de EVEX 3, bit [4]- B).

ARQUITECTURA DE REGISTRO ILUSTRATIVA

La **Figura 10** es un diagrama de bloques de una arquitectura de registro 1000 de acuerdo con una realización de la invención. En la realización ilustrada, hay 32 registros vectoriales 1010 que tienen 512 bits de anchura; estos registros se referencian como zmm0 a zmm31. Los 256 bits de orden inferior de los 16 registros zmm inferiores se superponen en los registros ymm0-16. Los 128 bits de orden inferior de los 16 registros zmm inferiores (los 128 bits de orden inferior de los registros ymm) se superponen en los registros xmm0-15. El formato de instrucciones específico compatible con vectores 900 opera en este archivo de registro superpuesto, como se ilustra en las siguientes tablas.

Longitud de vector ajustable	Clase	Operaciones	Registros
Plantillas de instrucciones que no incluyen el campo de longitud de vector 859B	A (Figura 8A; U=0)	810, 815, 825, 830	registros zmm (la longitud del vector es de 64 bytes)
	B (Figura 8B; U=1)	812	registros zmm (la longitud del vector es de 64 bytes)
Plantillas de instrucciones que incluyen el campo de longitud de vector 859B	B (Figura 8B; U=1)	817, 827	registros zmm, ymm o xmm (la longitud del vector es de 64 bytes, 32 bytes o 16 bytes) dependiendo del campo de longitud de vector 859B

En otras palabras, el campo de longitud de vector 859B selecciona entre una longitud máxima y una o más longitudes más cortas, donde cada una de tales longitudes más cortas es la mitad de la longitud de la longitud precedente; y las plantillas de instrucciones sin el campo de longitud de vector 859B operan en la longitud de vector máxima. Además, en una realización, las plantillas de instrucciones de clase B del formato de instrucciones específico compatible con vectores 900 operan en datos de coma flotante de precisión sencilla/doble escalar o empaquetados y datos de número entero escalar o empaquetados. Las operaciones escalares son operaciones realizadas en la posición del elemento de datos de orden más bajo en un registro zmm/ymm/xmm; las posiciones de los elementos de datos de orden superior se dejan igual que antes de la instrucción o se ponen a cero dependiendo de la realización.

Registros de máscara de escritura 1015 - en la realización ilustrada, hay 8 registros de máscara de escritura (k0 a k7), cada uno de 64 bits de tamaño. En una realización alternativa, los registros de máscara de escritura 1015 tienen un tamaño de 16 bits. Como se ha descrito anteriormente, en una realización de la invención, el registro de máscara vectorial k0 no se puede usar como máscara de escritura; cuando se usa la codificación que normalmente indicaría k0 para una máscara de escritura, selecciona una máscara de escritura programada de 0x6F, deshabilitando efectivamente la máscara de escritura para esa instrucción.

Registros de propósito general 1025 - en la realización ilustrada, hay dieciséis registros de propósito general de 64 bits que se usan junto con los modos de direccionamiento x86 existentes para direccionar operandos de memoria. A estos registros se hace referencia con los nombres RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP y R8 a R15.

Archivo de registro de pila de coma flotante escalar (pila x87) 1045, en el que se asigna un alias al archivo de registro plano de número entero empaquetado MMX 1050, en la realización ilustrada, la pila x87 es una pila de ocho elementos usada para realizar operaciones escalares de coma flotante en datos de coma flotante de 32/64/80 bits usando la extensión del conjunto de instrucciones x87; mientras que los registros MMX se usan para realizar operaciones con datos de números enteros empaquetados de 64 bits, así como para contener operandos para algunas operaciones realizadas entre los registros MMX y XMM.

Realizaciones alternativas de la invención pueden usar registros más anchos o más estrechos. Además, las realizaciones alternativas de la invención pueden usar más, menos o diferentes registros y archivos de registro.

ARQUITECTURAS DE NÚCLEO, PROCESADORES Y ARQUITECTURAS INFORMÁTICAS ILUSTRATIVAS

Los núcleos de procesador se pueden implementar de diferentes maneras, para diferentes propósitos y en diferentes procesadores. Por ejemplo, las implementaciones de tales núcleos pueden incluir: 1) un núcleo en orden de propósito general destinado a computación de propósito general; 2) un núcleo en desorden de propósito general de alto rendimiento destinado a computación de propósito general; 3) un núcleo de propósito especial destinado principalmente a gráficos y/o computación científica (rendimiento). Las implementaciones de diferentes procesadores pueden incluir: 1) una CPU que incluye uno o más núcleos en orden de propósito general destinados a la computación de propósito general y/o uno o más núcleos en desorden de propósito general destinados a la computación de propósito general; y 2) un coprocesador que incluye uno o más núcleos de propósito especial destinados principalmente a gráficos y/o temas científicos (rendimiento). Tales procesadores diferentes conducen a diferentes arquitecturas de sistemas informáticos, que pueden incluir: 1) el coprocesador en un chip separado de la CPU; 2) el coprocesador en un encapsulado separado en el mismo paquete que una CPU; 3) el coprocesador en el mismo encapsulado que una CPU (en cuyo caso, un coprocesador de este tipo en ocasiones se denomina lógica de propósito especial, tal como gráficos integrados y/o lógica científica (rendimiento), o núcleos de propósito especial); y 4) un sistema en un chip que puede incluir en el mismo encapsulado la CPU descrita (en ocasiones denominada como el núcleo o núcleos de aplicación o el procesador o procesadores de aplicación), el coprocesador descrito anteriormente

y funcionalidad adicional. A continuación, se describen arquitecturas de núcleo ilustrativas, seguidas de descripciones de procesadores y arquitecturas informáticas ilustrativas.

ARQUITECTURAS DE NÚCLEO ILUSTRATIVAS

DIAGRAMA DE BLOQUES DEL NÚCLEO EN ORDEN Y EN DESORDEN

La **Figura 11A** es un diagrama de bloques que ilustra tanto una canalización en orden ilustrativa como una canalización de emisión/ejecución en desorden y cambio de nombre de registro ilustrativa de acuerdo con realizaciones de la invención. La **Figura 11B** es un diagrama de bloques que ilustra tanto una realización ilustrativa de un núcleo de arquitectura en orden como un núcleo de arquitectura de emisión/ejecución en desorden de cambio de nombre de registro ilustrativo que se va a incluir en un procesador de acuerdo con realizaciones de la invención. Los cuadros con líneas continuas de las Figuras 11A-B ilustran la canalización en orden y el núcleo en orden, mientras que la adición opcional de los cuadros con líneas discontinuas ilustra la canalización y núcleo de emisión/ejecución en desorden de cambio de nombre de registro. Dado que el aspecto en orden es un subconjunto del aspecto en desorden, se describirá el aspecto en desorden.

En la **Figura 11A,** una canalización de procesador 1100 incluye una etapa de extracción 1102, una etapa de decodificación de longitud 1104, una etapa de decodificación 1106, una etapa de asignación 1108, una etapa de cambio de nombre 1110, una etapa de planificación (también conocida como despacho o emisión) 1112, una etapa de lectura de registro/lectura de memoria 1114, una etapa de ejecución 1116, una etapa de reescritura/escritura de memoria 1118, una etapa de manejo de excepciones 1122 y una etapa de confirmación 1124.

La **Figura 11B** muestra el núcleo de procesador 1190 que incluye una unidad de extremo frontal 1130 acoplada a una unidad de motor de ejecución 1150, y ambas están acopladas a una unidad de memoria 1170. El núcleo 1190 puede ser un núcleo informático de conjunto de instrucciones reducido (RISC), un núcleo informático de conjunto de instrucciones complejo (CISC), un núcleo de palabra de instrucción muy larga (VLIW) o un tipo de núcleo híbrido o alternativo. Como otra opción más, el núcleo 1190 puede ser un núcleo de propósito especial, tal como, por ejemplo, un núcleo de red o comunicación, un motor de compresión, un núcleo de coprocesador, un núcleo de unidad de procesamiento de gráficos informáticos de propósito general (GPGPU), un núcleo de gráficos o similar.

La unidad de extremo frontal 1130 incluye una unidad de predicción de bifurcación 1132 acoplada a una unidad de caché de instrucciones 1134, que está acoplada a una memoria intermedia de traducción adelantada de instrucciones (TLB) 1136, que está acoplada a una unidad de extracción de instrucciones 1138, que está acoplada a una unidad de decodificación 1140. La unidad de decodificación 1140 (o decodificador) puede decodificar instrucciones y generar como salida una o más microoperaciones, puntos de entrada de microcódigo, microinstrucciones, otras instrucciones u otras señales de control, que se decodifican a partir de, o que de otro modo reflejan o se derivan de las instrucciones originales. La unidad de decodificación 1140 puede implementarse usando diversos mecanismos diferentes. Ejemplos de mecanismos adecuados incluyen, pero no se limitan a tablas de consulta, implementaciones de hardware, matrices lógicas programables (PLA), memorias de solo lectura (ROM) de microcódigo, etc. En una realización, el núcleo 1190 incluye una ROM de microcódigo u otro medio que almacena microcódigo para determinadas macroinstrucciones (por ejemplo, en la unidad de decodificación 1140 o de otro modo dentro de la unidad de extremo frontal 1130). La unidad de decodificación 1140 está acoplada a una unidad de cambio de nombre/asignación 1152 en la unidad de motor de ejecución 1150.

La unidad de motor de ejecución 1150 incluye la unidad de cambio de nombre/asignación 1152 acoplada a una unidad de retiro 1154 y un conjunto de una o más unidades de programación 1156. La unidad o unidades de planificador 1156 representan cualquier número de planificadores diferentes, incluyendo estaciones de reserva, ventana de instrucciones central, etc. La unidad o unidades de planificación 1156 están acopladas a la unidad o unidades de archivo de registro físico 1158. Cada una de las unidades de archivos de registro físico 1158 representa uno o más archivos de registro físico, diferentes de los cuales almacenan uno o más tipos de datos diferentes, como número entero escalar, coma flotante escalar, número entero empaquetado, coma flotante empaquetado, número entero vectorial, coma flotante vectorial, estado (por ejemplo, un puntero de instrucción que es la dirección de la siguiente instrucción que se va a ejecutar), etc. En una realización, la unidad de archivos de registro físico 1158 comprende una unidad de registros vectoriales, una unidad de registros de máscara de escritura y una unidad de registros escalares. Estas unidades de registro pueden proporcionar registros vectoriales arquitectónicos, registros de máscara vectorial y registros de propósito general. La unidad o unidades de archivo o archivos de registro físico 1158 se superponen con la unidad de retiro 1154 para ilustrar diversas formas en las que se puede implementar el cambio de nombre de registro y la ejecución en desorden (por ejemplo, usando una memoria o memorias intermedias de reordenación y un archivo o archivos de registro de retiro; usando un archivo o archivos futuros, una memoria o memorias intermedias de historial y un archivo o archivos de registro de retiro; usando mapas de registros y una agrupación de registros; etc.). La unidad de retiro 1154 y la o las unidades de archivos de registro físico 1158 están acopladas al o a los grupos de ejecución 1160. El o los grupos de ejecución 1160 incluye un conjunto de una o más unidades de ejecución 1162 y un conjunto de una o más unidades con acceso a memoria 1164. Las unidades de ejecución 1162 pueden realizar varias operaciones (por ejemplo, desplazamientos, suma, resta, multiplicación) y en varios tipos de datos (por ejemplo, coma flotante escalar, número entero empaquetado, coma flotante empaquetado, número entero vectorial, vector coma

flotante). Si bien algunas realizaciones pueden incluir varias unidades de ejecución dedicadas a funciones o conjuntos de funciones específicas, otras realizaciones pueden incluir solo una unidad de ejecución o múltiples unidades de ejecución que realizan todas las funciones. La unidad o unidades de planificador 1156, la unidad o unidades de archivo o archivos de registro físico 1158 y la agrupación o agrupación de ejecución 1160 se muestran como posiblemente varios porque ciertas realizaciones crean canalizaciones separadas para ciertos tipos de datos/operaciones (por ejemplo, una canalización de número entero escalar, una canalización de coma flotante escalar/número entero empaquetado/coma flotante empaquetado/número entero vectorial/coma flotante vectorial y/o una canalización de acceso a memoria, cada una de las cuales tiene su propia unidad de planificador, unidad de archivo o archivos de registro físico y/o agrupación de ejecución - y en el caso de una canalización de acceso a memoria separada, se implementan ciertas realizaciones en las que únicamente la agrupación de ejecución de esta canalización tiene la unidad o unidades de acceso a memoria 1164). También se debe entender que, cuando se usan canalizaciones separadas, una o más de estas canalizaciones pueden ser de emisión/ejecución en desorden y el resto en orden.

El conjunto de unidades con acceso a memoria 1164 está acoplado a la unidad de memoria 1170, que incluye una unidad TLB de datos 1172 acoplada a una unidad de caché de datos 1174 acoplada a una unidad de caché de nivel 2 (L2) 1176. En una realización ilustrativa, las unidades de acceso a memoria 1164 pueden incluir una unidad de carga, una unidad de dirección de almacenamiento y una unidad de datos de almacenamiento, cada una de las cuales está acoplada a la unidad de TLB de datos 1172 en la unidad de memoria 1170. La unidad de caché de instrucciones 1134 se acopla adicionalmente a una unidad de caché de nivel 2 (L2) 1176 en la unidad de memoria 1170. La unidad de caché de L2 1176 se acopla a otros uno o más niveles de caché y, finalmente, a una memoria principal.

A modo de ejemplo, la arquitectura de núcleo de emisión/ejecución en desorden de cambio de nombre de registro ilustrativa puede implementar la canalización 1100 como sigue: 1) la instrucción extraer 1138 realiza las etapas de extracción y decodificación de longitud 1102 y 1104; 2) la unidad de decodificación 1140 realiza la etapa de decodificación 1106; 3) la unidad de cambio de nombre/asignador 1152 realiza la etapa de asignación 1108 y la etapa de cambio de nombre 1110; 4) la unidad o unidades de planificador 1156 realizan la etapa de planificación 1112; 5) la unidad o unidades de archivo o archivos de registro físico 1158 y la unidad de memoria 1170 realizan la etapa de lectura de registro/lectura de memoria 1114; el grupo de ejecución 1160 realiza la etapa de ejecución 1116; 6) la unidad de memoria 1170 y la unidad o unidades de archivo o archivos de registro físico 1158 realizan la etapa de reescritura/escritura en memoria 1118; 7) diversas unidades pueden estar implicadas en la etapa de manejo de excepciones 1122; y 8) la unidad de retiro 1154 y la unidad o unidades de archivo o archivos de registro físico 1158 realizan la etapa de confirmación 1124.

El núcleo 1190 puede soportar uno o más conjuntos de instrucciones (por ejemplo, el conjunto de instrucciones x86 (con algunas extensiones que se han añadido con versiones más nuevas); el conjunto de instrucciones MIPS de MIPS Technologies de Sunnyvale, CA; el conjunto de instrucciones ARM (con extensiones adicionales opcionales tales como NEON) de ARM Holdings de Sunnyvale, CA), que incluyen la instrucción o instrucciones descritas en el presente documento. En una realización, el núcleo 1190 incluye una lógica para soportar una extensión del conjunto de instrucciones de datos empaquetados (por ejemplo, AVX1, AVX2), permitiendo de este modo que las operaciones usadas por muchas aplicaciones multimedia se realicen usando datos empaquetados.

Debe entenderse que el núcleo puede soportar múltiples hilos (ejecutar dos o más conjuntos paralelos de operaciones o hilos), y puede hacerlo de diversas maneras, incluyendo múltiples hilos en intervalos de tiempo, múltiples hilos simultáneos (donde un único núcleo físico proporciona un núcleo lógico para cada uno de los hilos de ese núcleo físico que está realizando múltiples hilos simultáneamente), o una combinación de los mismos (por ejemplo, extracción y decodificación en intervalos de tiempo y múltiples hilos simultáneos a partir de entonces, tal como en la tecnología Intel® Hyperthreading).

Aunque el cambio de nombre de registro se describe en el contexto de una ejecución en desorden, se debería entender que el cambio de nombre de registro se puede usar en una arquitectura en orden. Aunque la realización ilustrada del procesador también incluye unidades de caché de instrucciones y de datos 1134/1174 separadas y una unidad de caché de L2 1176 compartida, realizaciones alternativas pueden tener una única caché interna tanto para instrucciones como para datos, tal como, por ejemplo, una caché interna de nivel 1 (L1) o múltiples niveles de caché interna. En algunas realizaciones, el sistema puede incluir una combinación de una memoria caché interna y una memoria caché externa que es externa al núcleo y/o al procesador. Como alternativa, toda la caché puede ser externa al núcleo y/o al procesador.

ARQUITECTURA DE NÚCLEO EN ORDEN ILUSTRATIVA ESPECÍFICA

Las Figuras 12A-B ilustran un diagrama de bloques de una arquitectura de núcleo en orden, ilustrativa más específica, cuyo núcleo sería uno de varios bloques lógicos (que incluyen otros núcleos del mismo tipo y/o tipos diferentes) en un chip. Los bloques lógicos se comunican a través de una red de interconexión de gran ancho de banda (por ejemplo, una red en anillo) con alguna lógica de función fija, interfaces de E/S de memoria y otra lógica de E/S necesaria, de acuerdo con la aplicación.

La **Figura 12A** es un diagrama de bloques de un único núcleo de procesador, junto con su conexión a la red de interconexión integrada 1202 y con su subconjunto local de la memoria caché de nivel 2 (L2) 1204, de acuerdo con realizaciones de la invención. En una realización, un decodificador de instrucciones 1200 soporta el conjunto de instrucciones x86 con una extensión del conjunto de instrucciones de datos empaquetados. Una caché L1 1206 permite accesos de baja latencia a la memoria caché en las unidades escalares y vectoriales. Mientras que en una realización (para simplificar el diseño), una unidad escalar 1208 y una unidad vectorial 1210 usan conjuntos de registros separados (respectivamente, registros escalares 1212 y registros vectoriales 1214) y los datos transferidos entre ellas se escriben en la memoria y, a continuación, se vuelven a leer desde una caché de nivel 1 (L1) 1206, realizaciones alternativas de la invención pueden usar un enfoque diferente (por ejemplo, usar un único conjunto de registros o incluir una ruta de comunicación que permite que los datos se transfieran entre los dos archivos de registro sin tener que escribirse ni volverse a leer).

El subconjunto local de la caché de L2 1204 es parte de una caché de L2 global que se divide en subconjuntos locales separados, uno por núcleo de procesador. Cada núcleo de procesador tiene una ruta de acceso directo a su propio subconjunto local de la caché de L2 1204. Los datos leídos por un núcleo de procesador se almacenan en su subconjunto de caché de L2 1204 y se puede acceder a los mismos rápidamente, en paralelo con que otros núcleos de procesador accedan a sus propios subconjuntos de caché de L2 locales. Los datos escritos por un núcleo de procesador se almacenan en su propio subconjunto de caché L2 1204 y se eliminan de otros subconjuntos, si es necesario. La red en anillo garantiza la coherencia de los datos compartidos. La red en anillo es bidireccional para permitir que agentes tales como núcleos de procesador, cachés L2 y otros bloques lógicos se comuniquen entre sí dentro del chip. Cada ruta de datos del anillo tiene 1012 bits de anchura por dirección.

La **Figura 12B** es una vista ampliada de parte del núcleo del procesador en la **Figura 12A** de acuerdo con realizaciones de la invención. La **Figura 12B** incluye una caché de datos L1 1206A parte de la caché L1 1204, así como más detalles con respecto a la unidad de vector 1210 y los registros vectoriales 1214. Específicamente, la unidad vectorial 1210 es una unidad de procesamiento vectorial (VPU) de 16 de ancho (véase la ALU 1228 de 16 de ancho), que ejecuta una o más instrucciones flotantes de precisión de números enteros, sencilla y doble. La VPU soporta el mezclado de las entradas de registro con la unidad de mezcla 1220, la conversión numérica con las unidades de conversión numérica 1222A-B y la replicación con la unidad de replicación 1224 en la entrada de memoria. Los registros de máscara de escritura 1226 permiten predecir escrituras vectoriales resultantes.

La **Figura 13** es un diagrama de bloques de un procesador 1300 que puede tener más de un núcleo, puede tener un controlador de memoria integrado y puede tener gráficos integrados de acuerdo con realizaciones de la invención. Los recuadros con líneas continuas en la **Figura 13** ilustran un procesador 1300 con un único núcleo 1302A, un agente de sistema 1310, un conjunto de una o más unidades de controlador de bus 1316, mientras que, la adición opcional de los recuadros con líneas discontinuas ilustra un procesador alternativo 1300 con múltiples núcleos 1302A-N, un conjunto de una o más unidad o unidades de controlador de memoria integrado 1314 en la unidad de agente de sistema 1310 y lógica de propósito especial 1308.

Por tanto, diferentes implementaciones del procesador 1300 pueden incluir: 1) una CPU con la lógica de propósito especial 1308 que es una lógica (que puede incluir uno o más núcleos) de gráficos y/o temas científicos integrados (rendimiento), y siendo los núcleos 1302A-N uno o más núcleos de propósito general (por ejemplo, núcleos en orden de propósito general, núcleos en desorden de propósito general, una combinación de los dos); 2) un coprocesador con núcleos 1302A-N que son un gran número de núcleos de propósito especial destinados principalmente a gráficos y/o temas científicos (rendimiento); y 3) un coprocesador con los núcleos 1302A-N que son un gran número de núcleos en orden de propósito general. Por tanto, el procesador 1300 puede ser un procesador de propósito general, coprocesador o procesador de propósito especial, tal como, por ejemplo, un procesador de red o comunicación, motor de compresión, procesador de gráficos, GPGPU (unidad de procesamiento de gráficos de propósito general), un coprocesador de alto rendimiento de muchos núcleos integrados (MIC) (que incluye 30 o más núcleos), procesador integrado o similar. El procesador puede implementarse en uno o más chips. El procesador 1300 puede ser parte y/o puede implementarse en uno o más sustratos usando cualquiera de un número de tecnologías de proceso, tales como, por ejemplo, BiCMOS, CMOS o NMOS.

La jerarquía de memoria incluye uno o más niveles de caché dentro de los núcleos, un conjunto o una o más unidades de caché compartidas 1306 y memoria externa (no mostrada) acoplada al conjunto de unidades de controlador de memoria integrado 1314. El conjunto de unidades de caché compartida 1306 puede incluir una o más cachés de nivel medio, tales como nivel 2 (L2), nivel 3 (L3), nivel 4 (L4), u otros niveles de caché, una caché de último nivel (LLC). y/o combinaciones de las mismas. Si bien en una realización una unidad de interconexión basada en anillo 1312 interconecta la lógica de gráficos integrada 1308 (la lógica de gráficos integrada 1308 es un ejemplo y también se denomina en el presente documento lógica de propósito especial), el conjunto de unidades de caché compartida 1306 y la unidad de agente del sistema 1310/unidad o unidades de controlador de memoria integrada 1314, realizaciones alternativas pueden usar cualquier número de técnicas bien conocidas para interconectar dichas unidades. En una realización, se mantiene la coherencia entre una o más unidades de caché 1306 y núcleos 1302-A-N.

En algunas realizaciones, uno o más de los núcleos 1302A-N son capaces de realizar subprocesos múltiples. El agente de sistema 1310 incluye aquellos componentes que coordinan y operan los núcleos 1302A-N. La unidad de agente de

sistema 1310 puede incluir, por ejemplo, una unidad de control de energía (PCU) y una unidad de visualización. La PCU puede ser o incluir la lógica y los componentes necesarios para regular el estado de energía de los núcleos 1302A-N y la lógica de gráficos integrados 1308. La unidad de visualización es para controlar una o más pantallas conectadas externamente.

Los núcleos 1302A-N pueden ser homogéneos o heterogéneos en términos de conjunto de instrucciones de arquitectura; es decir, dos o más de los núcleos 1302A-N pueden ser capaces de ejecutar el mismo conjunto de instrucciones, mientras que otros pueden ser capaces de ejecutar solo un subconjunto de ese conjunto de instrucciones o un conjunto de instrucciones diferente.

ARQUITECTURAS INFORMÁTICAS ILUSTRATIVAS

Las **Figuras 14-17** son diagramas de bloques de arquitecturas informáticas ilustrativas. Otros diseños y configuraciones de sistema conocidos en la técnica para portátiles, equipos de sobremesa, PC portátiles, asistentes digitales personales, estaciones de trabajo de ingeniería, servidores, dispositivos de red, concentradores de red, conmutadores, procesadores integrados, procesadores de señales digitales (DSP), dispositivos de gráficos, dispositivos de videojuegos, decodificadores de salón, microcontroladores, teléfonos móviles, reproductores multimedia portátiles, dispositivos de mano y diversos otros dispositivos electrónicos, también son adecuados. En general, son generalmente adecuados una gran diversidad de sistemas o dispositivos electrónicos que pueden incorporar un procesador y/u otra lógica de ejecución como se divulga en el presente documento.

Haciendo referencia ahora a la **Figura 14,** se muestra un diagrama de bloques de un sistema 1400 de acuerdo con una realización de la presente invención. El sistema 1400 puede incluir uno o más procesadores 1410, 1415, que están acoplados a un concentrador de controlador 1420. En una realización, el concentrador DE controlador 1420 incluye un concentrador DE controlador de memoria gráfica (GMCH) 1490 y un concentrador de entrada/salida (IOH) 1450 (que puede estar en chips separados); el GMCH 1490 incluye controladores de memoria y gráficos a los que están acoplados la memoria 1440 y un coprocesador 1445; el IOH 1450 acopla los dispositivos de entrada/salida (E/S) 1460 al GMCH 1490. Alternativamente, uno o ambos controladores de memoria y gráficos están integrados dentro del procesador (como se describe en el presente documento), la memoria 1440 y el coprocesador 1445 están acoplado directamente al procesador 1410, y al concentrador de controlador 1420 en un único chip con el IOH 1450.

La naturaleza opcional de los procesadores adicionales 1415 se indica en la **Figura 14** con líneas discontinuas. Cada procesador 1410, 1415 puede incluir uno o más de los núcleos de procesamiento descritos en el presente documento y puede ser alguna versión del procesador 1300.

La memoria 1440 puede ser, por ejemplo, memoria de acceso aleatorio dinámica (DRAM), memoria de cambio de fase (PCM), o una combinación de las dos. Para al menos una realización, el concentrador de controlador 1420 se comunica con el o los procesadores 1410, 1415 a través de un bus multipunto, tal como un bus frontal (FSB), una interfaz punto a punto tal como QuickPath Interconnect (QPI) o una conexión similar 1495.

En una realización, el coprocesador 1445 es un procesador de propósito especial, tal como, por ejemplo, un procesador MIC de alto rendimiento, un procesador de red o comunicación, motor de compresión, procesador de gráficos, GPGPU, procesador integrado o similar. En una realización, el concentrador de controlador 1420 puede incluir un acelerador de gráficos integrado.

Puede haber una diversidad de diferencias entre los recursos físicos 1410, 1415 en términos de un espectro de métricas de valor que incluyen características arquitectónicas, microarquitectónicas, térmicas, de consumo de energía y similares.

En una realización, el procesador 1410 ejecuta instrucciones que controlan operaciones de procesamiento de datos de un tipo general. Incrustadas dentro de las instrucciones puede haber instrucciones de coprocesador. El procesador 1410 reconoce estas instrucciones de coprocesador como de un tipo que debería ser ejecutado por el coprocesador adjunto 1445. En consecuencia, el procesador 1410 emite estas instrucciones de coprocesador (o señales de control que representan instrucciones de coprocesador) en un bus del coprocesador u otra interconexión, al coprocesador 1445. Los coprocesadores 1445 aceptan y ejecutan las instrucciones de coprocesador recibidas.

Con referencia ahora a la **Figura 15**, se muestra un diagrama de bloques de un primer sistema 1500 ilustrativo más específico de acuerdo con una realización de la presente invención. Como se muestra en la **Figura 15**, el sistema multiprocesador 1500 es un sistema de interconexión punto a punto e incluye un primer procesador 1570 y un segundo procesador 1580 acoplados a través de una interconexión punto a punto 1550. Cada uno de los procesadores 1570 y 1580 puede ser alguna versión del procesador 1300. En una realización de la invención, los procesadores 1570 y 1580 son respectivamente los procesadores 1410 y 1415, mientras que el coprocesador 1538 es el coprocesador 1445. En otra realización, los procesadores 1570 y 1580 son respectivamente el procesador 1410 y el coprocesador 1445.

Los procesadores 1570 y 1580 se muestran incluyendo las unidades de controlador de memoria integrada (IMC) 1572 y 1582, respectivamente. El procesador 1570 también incluye como parte de sus unidades de controlador de bus,

interfaces punto a punto (P-P) 1576 y 1578; de manera similar, el segundo procesador 1580 incluye interfaces P-P 1586 y 1588. Los procesadores 1570, 1580 pueden intercambiar información a través de una interfaz punto a punto (P-P) 1550 usando circuitos de interfaz P-P 1578, 1588. Como se muestra en la **Figura 15**, los IMC 1572 y 1582 acoplan los procesadores a las respectivas memorias, en concreto, una memoria 1532 y una memoria 1534, que pueden ser porciones de la memoria principal conectadas localmente a los respectivos procesadores.

Cada uno de los procesadores 1570, 1580 puede intercambiar información con un conjunto de chips 1590 a través de interfaces P-P individuales 1552, 1554 usando circuitos de interfaz de punto a punto 1576, 1594, 1586, 1598. El conjunto de chips 1590 puede opcionalmente intercambiar información con el coprocesador 1538 a través de una interfaz de alto rendimiento 1592. En una realización, el coprocesador 1538 es un procesador de propósito especial, tal como, por ejemplo, un procesador MIC de alto rendimiento, un procesador de red o comunicación, motor de compresión, procesador de gráficos, GPGPU, procesador integrado o similar.

Una memoria caché compartida (no mostrada) puede incluirse en cualquiera de los procesadores o fuera de ambos procesadores aún conectada con los procesadores a través de la interconexión PP, de manera que la información de caché local de uno o ambos procesadores puede almacenarse en la memoria caché compartida si un procesador se coloca en un modo de bajo consumo.

El conjunto de chips 1590 se puede acoplar a un primer bus 1516 a través de una interfaz 1596. En una realización, el primer bus 1516 puede ser un bus de interconexión de componentes periféricos (PCI), o un bus tal como un bus PCI Express u otro bus de interconexión de E/S de tercera generación, aunque el alcance de la presente invención no está así limitado.

Como se muestra en la **Figura 15**, diversos dispositivos de E/S 1514 pueden acoplarse al primer bus 1516, junto con un puente de bus 1518 que acopla el primer bus 1516 a un segundo bus 1520. En una realización, uno o más procesadores adicionales 1515, tales como coprocesadores, procesadores MIC de alto rendimiento, GPGPU, aceleradores (tales como, por ejemplo, aceleradores de gráficos o unidades de procesamiento de señales digitales (DSP)), conjuntos de puertas programables en campo o cualquier otro procesador, están acoplados al primer bus 1516. En una realización, el segundo bus 1520 puede ser un bus de recuento bajo de pines (LPC). Se pueden acoplar diversos dispositivos a un segundo bus 1520 que incluye, por ejemplo, un teclado y/o ratón 1522, dispositivos de comunicaciones 1527 y una unidad de almacenamiento 1528 tal como una unidad de disco u otro dispositivo de almacenamiento masivo que puede incluir instrucciones/códigos y datos 1530, en una realización. Además, se puede acoplar una E/S de audio 1524 al segundo bus 1520. Obsérvese que son posibles otras arquitecturas. Por ejemplo, en lugar de la arquitectura punto a punto de la **Figura 15,** un sistema puede implementar un bus multipunto u otra arquitectura similar.

Con referencia ahora a la **Figura 16**, se muestra un diagrama de bloques de un segundo sistema 1600 ilustrativo más específico de acuerdo con una realización de la presente invención. Elementos similares en las **Figuras 15 y 16** llevan números de referencia similares, y ciertos aspectos de la **Figura 15** se han omitido de la **Figura 16** para evitar complicar otros aspectos de la **Figura 16**.

La **Figura 16** ilustra que los procesadores 1570, 1580 pueden incluir memoria integrada y lógica de control de E/S ("CL") 1572 y 1582, respectivamente. Por tanto, la CL 1572, 1582 incluyen unidades controladoras de memoria integradas e incluyen lógica de control de E/S. La **Figura 16** ilustra que no solo las memorias 1532, 1534 están acopladas a la CL 1572, 1582, sino que también los dispositivos de E/S 1614 también están acoplados a la lógica de control 1572, 1582. Los dispositivos de E/S heredados 1615 están acoplados al conjunto de chips 1590.

Con referencia ahora a la **Figura 17**, se muestra un diagrama de bloques de un SoC 1700 de acuerdo con una realización de la presente invención. Elementos similares en la **Figura 13** llevan los mismos números de referencia. Además, los recuadros con línea discontinua son características opcionales en los SoC más avanzados. En la **Figura 17**, una unidad o unidades de interconexión 1702 están acopladas a: un procesador de aplicaciones 1710 que incluye un conjunto de uno o más núcleos 1302A-N, que incluyen una o más unidades de caché 1304A-N y una unidad o unidades de caché compartida 1306; una unidad de agente de sistema 1310; una unidad o unidades de controlador de bus 1316; una unidad o unidades de controlador de memoria integrado 1314; un conjunto o uno o más coprocesadores 1720 que pueden incluir lógica de gráficos integrada, un procesador de imágenes, un procesador de audio y un procesador de vídeo; una unidad de memoria de acceso aleatorio estática (SRAM) 1730; una unidad de acceso directo a memoria (DMA) 1732; y una unidad de visualización 1740 para acoplarse a una o más pantallas externas. En una realización, el coprocesador o coprocesadores 1720 incluyen un procesador de propósito especial, tal como, por ejemplo, un procesador de red o comunicación, motor de compresión, GPGPU, un procesador MIC de alto rendimiento, procesador integrado o similar.

Las realizaciones de los mecanismos divulgados en el presente documento pueden implementarse en hardware, software, firmware o una combinación de tales enfoques de implementación. Las realizaciones de la invención pueden implementarse como programas informáticos o código de programa que se ejecuta en sistemas programables que comprenden al menos un procesador, un sistema de almacenamiento (que incluye memoria y/o elementos de almacenamiento volátiles y no volátiles), al menos un dispositivo de entrada y al menos un dispositivo de salida.

El código de programa, tal como el código 1530 ilustrado en la **Figura 15**, se puede aplicar a las instrucciones de entrada para realizar las funciones descritas en el presente documento y generar información de salida. La información de salida puede aplicarse a uno o más dispositivos de salida, de forma conocida. Para los propósitos de esta solicitud, un sistema de procesamiento incluye cualquier sistema que tenga un procesador, tal como, por ejemplo; un procesador de señales digitales (DSP), un microcontrolador, un circuito integrado de específico de la aplicación (ASIC) o un microprocesador.

El código de programa puede implementarse en un lenguaje de programación orientado a objetos o procedural de alto nivel para comunicarse con un sistema de procesamiento. El código del programa también puede implementarse en lenguaje ensamblador o máquina, si se desea. De hecho, los mecanismos descritos en el presente documento no están limitados en su alcance a ningún lenguaje de programación particular. En cualquier caso, el lenguaje puede ser un lenguaje compilado o interpretado.

Uno o más aspectos de al menos una realización pueden implementarse mediante instrucciones representativas almacenadas en un medio legible por máquina que representa diversa lógica dentro del procesador que, cuando las lee una máquina, hace que la máquina fabrique lógica para realizar las técnicas descritas en el presente documento. Tales representaciones, conocidas como "núcleos de IP", pueden almacenarse en un medio legible por máquina tangible y suministrarse a diversos clientes o instalaciones de fabricación para cargarlas en las máquinas de fabricación que realmente hacen la lógica o el procesador.

Tales medios de almacenamiento legibles por máquina pueden incluir, sin limitación, disposiciones tangibles no transitorias de artículos fabricados o formados por una máquina o dispositivo, que incluyen medios de almacenamiento tales como discos duros, cualquier otro tipo de disco, incluyendo disquetes, discos ópticos memorias de solo lectura de disco compacto (CD-ROM), discos compactos reescribibles (CD-RW) y discos magneto-ópticos, dispositivos de semiconductores tales como memorias de solo lectura (ROM), memorias de acceso aleatorio (RAM) tales como memorias de acceso aleatorio dinámicas (DRAM), memorias de acceso aleatorio estáticas (SRAM), memorias de solo lectura programables y borrables (EPROM), memorias flash, memorias de solo lectura programables y borrables eléctricamente (EEPROM), memorias de cambio de fase (PCM), tarjetas magnéticas u ópticas, o cualquier otro tipo de medio adecuado para almacenamiento de instrucciones electrónicas.

En consecuencia, las realizaciones de la invención también incluyen medios legibles por máquina, tangibles, no transitorios que contienen instrucciones o datos de diseño, tales como el lenguaje de descripción de hardware (HDL), que define estructuras, circuitos, aparatos, procesadores y/o características del sistema descritos en el presente documento. Tales realizaciones también pueden denominarse productos de programa.

EMULACIÓN (INCLUYENDO TRADUCCIÓN BINARIA, TRANSFORMACIÓN DE CÓDIGOS, ETC.)

En algunos casos, se puede usar un convertidor de instrucciones para convertir una instrucción de un conjunto de instrucciones de origen a un conjunto de instrucciones de destino. Por ejemplo, el convertidor de instrucciones puede traducir (por ejemplo, usando traducción binaria estática, traducción binaria dinámica que incluye compilación dinámica), transformar, emular o convertir de otro modo una instrucción en una o más otras instrucciones a procesar por el núcleo. El convertidor de instrucciones puede implementarse en software, hardware, firmware o una combinación de los mismos. El convertidor de instrucciones puede estar en el procesador, fuera del procesador o en parte dentro y fuera del procesador.

La **Figura 18** es un diagrama de bloques que contrasta el uso de un convertidor de instrucciones de software para convertir instrucciones binarias en un conjunto de instrucciones de origen en instrucciones binarias en un conjunto de instrucciones objetivo de acuerdo con realizaciones de la invención. En la realización ilustrada, el convertidor de instrucciones es un convertidor de instrucciones de software, aunque, como alternativa, el convertidor de instrucciones puede implementarse en software, firmware, hardware o diversas combinaciones de los mismos. La **Figura 18** muestra que un programa en un lenguaje de alto nivel 1802 puede compilarse usando un compilador x86 1804 para generar código binario x86 1806 que puede ejecutarse de forma nativa mediante un procesador con al menos un núcleo de conjunto de instrucciones x86 1816. El procesador con al menos un núcleo de conjunto de instrucciones x86 1816 representa cualquier procesador que puede realizar sustancialmente las mismas funciones que un procesador Intel con al menos un núcleo de conjunto de instrucciones x86 ejecutando o procesando de otro modo (1) una porción sustancial del conjunto de instrucciones del núcleo de conjunto de instrucciones Intel x86 o (2) versiones de código objeto de aplicaciones u otro software que tiene como objetivo ejecutarse en un procesador Intel con al menos un núcleo del conjunto de instrucciones x86, para lograr sustancialmente el mismo resultado que un procesador Intel con al menos un núcleo del conjunto de instrucciones x86. El compilador x86 1804 representa un compilador que puede funcionar para generar código binario x86 1806 (por ejemplo, código objeto) que puede, con o sin procesamiento de vinculación adicional, ejecutarse en el procesador con al menos un núcleo de conjunto de instrucciones x86 1816. De manera similar, la **Figura 18** muestra que el programa en el lenguaje de alto nivel 1802 puede compilarse usando un compilador de conjunto de instrucciones alternativo 1808 para generar un código binario de conjunto de instrucciones alternativo 1810 que puede ejecutarse de forma nativa por un procesador sin al menos un núcleo de conjunto de instrucciones x86 1814 (por ejemplo, un procesador con núcleos que ejecutan el conjunto de instrucciones MIPS de

MIPS Technologies de Sunnyvale, CA y/o que ejecutan el conjunto de instrucciones ARM de ARM Holdings de Sunnyvale, CA). El convertidor de instrucciones 1812 se usa para convertir el código binario x86 1806 en un código que el procesador puede ejecutar de forma nativa sin un núcleo de conjunto de instrucciones x86 1814. No es probable que este código convertido sea el mismo que el código binario del conjunto de instrucciones alternativo 1810 porque es difícil hacer un convertidor de instrucciones apto para esto; sin embargo, el código convertido conseguirá la operación general y estará compuesto por instrucciones del conjunto de instrucciones alternativo. Por tanto, el convertidor de instrucciones 1812 representa software, firmware, hardware o una combinación de los mismos que, mediante emulación, simulación o cualquier otro proceso, permite que un procesador u otro dispositivo electrónico que no tiene un procesador o núcleo de conjunto de instrucciones x86 ejecute el código binario x86 1806.

REIVINDICACIONES

1. Un procesador (100) que comprende:
circuitería de extracción y decodificación (105, 109) para extraer y decodificar una instrucción de multiplicación de matrices de dispersión variable de formato variable, VFVSMM, (107, 111), que tiene campos para especificar ubicaciones de matrices A, B y C que tienen elementos MxK, KxN y MxN, respectivamente; y
circuitería de ejecución (119, 208) que incluye una matriz de procesamiento de unidades de procesamiento (M x N) (210), reconfigurables para operar en uno de un modo denso-denso, modo denso-disperso y modo disperso-disperso, en donde la circuitería de ejecución (119, 208), que opera en el modo denso-denso, en respuesta a la instrucción de VFVSMM decodificada, está configurada para enrutar cada fila de la matriz A, escalonando las filas posteriores, a una fila correspondiente de la matriz de procesamiento y enrutar cada columna de la matriz B, escalonando las columnas posteriores, a una columna correspondiente de la matriz de procesamiento, y
en donde cada una de las unidades de procesamiento MxN (210), que opera en el modo denso-denso, está configurada para:
generar K productos de elementos de matriz A y matriz B coincidentes recibidos de las matrices A y B, respectivamente, para que exista una coincidencia cuando el elemento de matriz B tiene la misma dirección de fila que una dirección de columna del elemento de matriz A; y
acumular cada producto generado con un elemento correspondiente de la matriz C que tiene una misma posición relativa que la de la unidad de procesamiento (210) en la matriz de procesamiento,
en donde la circuitería de ejecución (119, 208), que opera en el modo denso-disperso, en respuesta a la instrucción de VFVSMM decodificada, está configurada para enrutar cada fila de la matriz A, escalonando filas posteriores, a la fila correspondiente de la matriz de procesamiento, y enrutar cada columna de la matriz B a la columna correspondiente de la matriz de procesamiento, siendo la matriz B para comprender únicamente elementos distintos de cero de una matriz dispersa que comprende lógicamente KxN elementos, con cada elemento para incluir un campo para especificar sus direcciones de fila y columna lógicas, y
en donde cada una de las unidades de procesamiento MxN (210), que opera en el modo denso-disperso, está configurada para:
determinar si existe una coincidencia de dirección entre la dirección de fila lógica del elemento de matriz B y la dirección de columna del elemento de matriz A; y
cuando existe la coincidencia, generar el producto, y cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B cuando la dirección de columna del elemento de matriz A es mayor que la dirección de fila lógica del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A,
en donde la circuitería de ejecución (119, 208), que opera en el modo disperso-disperso, en respuesta a la instrucción de VFVSMM decodificada, está configurada para enrutar cada fila de la matriz A a la fila correspondiente de la matriz de procesamiento, y enrutar cada columna de la matriz B a la columna correspondiente de la matriz de procesamiento, siendo las matrices A y B matrices dispersas que comprenden únicamente elementos distintos de cero de matrices lógicas MxK y KxN, respectivamente, cada elemento que incluye un campo para especificar sus direcciones de fila y columna lógicas, y
en donde cada una de las unidades de procesamiento MxN (210), que opera en el modo disperso-disperso, está configurada para:
determinar si existe una coincidencia entre la dirección de columna lógica del elemento de matriz A y la dirección de fila lógica del elemento de matriz B; y
cuando existe la coincidencia, generar el producto, y cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B cuando la dirección de columna lógica del elemento de matriz A es mayor que la dirección de fila lógica del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A,
en donde la instrucción de VFVSMM está adaptada además para especificar un formato de datos de cada uno de los elementos de datos de las matrices A, B y C especificadas, siendo el formato de datos uno de número entero, coma flotante de precisión media, coma flotante de precisión sencilla, coma flotante de precisión doble y un formato personalizado,
en donde cuando la instrucción de VFVSMM especifica un número entero de 8 bits como un formato de datos de cada uno de los elementos de datos de las matrices A, B y C especificadas, cada una de las unidades de procesamiento (210) en la matriz de procesamiento está configurada para realizar una multiplicación de matriz 2x2 cuando se procesan los elementos de datos de números enteros de 8 bits.

2. El procesador de la reivindicación 1, en donde la circuitería de ejecución (119, 208) está configurada para enrutar cada fila de la matriz A y cada columna de la matriz B a una tasa de un elemento por ciclo de reloj y para escalonar cada fila posterior y columna posterior por un ciclo de reloj, y en donde cada una de las unidades de procesamiento MxN (210) está configurada para inferir direcciones de columna y fila de cada elemento de matriz A y matriz B recibido basándose en un ciclo de reloj y en una posición relativa de la unidad de procesamiento 210) dentro de la matriz de procesamiento.

3. El procesador de la reivindicación 1 o 2, en donde cada una de las unidades de procesamiento MxN (210) está configurada además para, cuando no existe coincidencia, generar y enviar una solicitud de mantenimiento aguas arriba

en una dirección del elemento de datos que se está manteniendo, y generar y enviar una notificación de mantenimiento aguas abajo en la dirección del elemento de datos que se está manteniendo.

4. El procesador de las reivindicaciones 1-3, en donde la circuitería de ejecución (119, 208), cuando pasa un elemento de datos, está configurada para difundir el elemento de datos aguas abajo a dos o más unidades de procesamiento (210).

5. Un método (600), que comprende:
extraer y decodificar (602, 604), usando circuitería de extracción y decodificación (105, 109) de un procesador (100), una instrucción de multiplicación de matriz de dispersión variable de formato variable, VFVSMM, (107, 111) que tiene campos para especificar ubicaciones de matrices A, B y C que tienen elementos MxK, KxN y MxN, respectivamente, y
responder (608) a la instrucción de VFVSMM decodificada, usando circuitería de ejecución (119, 208) que incluye una matriz de procesamiento de unidades de procesamiento (M x N) (210), del procesador reconfigurable para operar en uno de un modo denso-denso, modo denso-disperso y modo disperso-disperso, enrutando por la circuitería de ejecución, que opera en el modo denso-denso, en respuesta a la instrucción de VFVSMM decodificada cada fila de la matriz A, escalonando filas posteriores, a una fila correspondiente de la matriz de procesamiento y
escalonar columnas posteriores cada columna de la matriz B a una columna correspondiente de la matriz de procesamiento, y
generar por cada una de las unidades de procesamiento MxN que operan en el modo denso-denso, K productos de elementos de matriz A y matriz B coincidentes recibidos de las matrices A y B, respectivamente, para que exista una coincidencia cuando el elemento de matriz B tiene una misma dirección de fila que una dirección de columna del elemento de matriz A; y
acumular por cada una de las unidades de procesamiento MxN cada producto generado con un elemento correspondiente de la matriz C que tiene una misma posición relativa que una posición de la unidad de procesamiento en la matriz de procesamiento (610), o enrutar por la circuitería de ejecución, que opera en el modo denso-disperso, en respuesta a la instrucción de VFVSMM decodificada, cada fila de la matriz A, escalonar las filas posteriores, a la fila correspondiente de la matriz de procesamiento, y cada columna de la matriz B en la columna correspondiente de la matriz de procesamiento, siendo la matriz B para comprender únicamente elementos distintos de cero de una matriz dispersa que comprende lógicamente KxN elementos, con cada elemento para incluir un campo para especificar sus direcciones de fila y columna lógicas, y
determinar por cada una de las unidades de procesamiento MxN, que operan en denso-disperso, si existe una coincidencia de dirección entre la dirección de fila lógica del elemento de matriz B y la dirección de columna del elemento de matriz A; y
cuando existe la coincidencia, generar por la unidad de procesamiento respectiva el producto, y cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B por la unidad de procesamiento respectiva cuando la dirección de columna inferida del elemento de matriz A es mayor que la dirección de fila lógica del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento Z de matriz A por la unidad de procesamiento respectiva, o
enrutar, por la circuitería de ejecución, que opera en el modo disperso-disperso, en respuesta a la instrucción de VFVSMM decodificada, cada fila de la matriz A en la fila correspondiente de la matriz de procesamiento, y cada columna de la matriz B en la columna correspondiente de la matriz de procesamiento, siendo las matrices A y B dispersas que comprenden únicamente elementos distintos de cero de matrices MxK y KxN lógicas, respectivamente, incluyendo cada elemento un campo para especificar sus direcciones de fila y columna lógicas, y
determinar, por cada una de las unidades de procesamiento MxN, que operan en el modo disperso-disperso, si existe una coincidencia entre la dirección de columna lógica del elemento de matriz A y la dirección de fila lógica del elemento de matriz B; y
cuando existe la coincidencia, generar por la unidad de procesamiento respectiva el producto, y cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B por la unidad de procesamiento respectiva cuando la dirección de columna lógica del elemento de matriz A es mayor que la dirección de fila lógica del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A por la unidad de procesamiento respectiva,
en donde cuando la instrucción de VFVSMM especifica un número entero de 8 bits como un formato de datos de cada uno de los elementos de datos de las matrices A, B y C especificadas,
realizar por cada una de las unidades de procesamiento (210) en la multiplicación de matriz de 2x2 de matriz de procesamiento cuando se procesan los elementos de datos de números enteros de 8 bits

6. El método de la reivindicación 5, en donde el enrutamiento por la circuitería de ejecución de cada fila de la matriz A y cada columna de la matriz B es a una tasa de un elemento por ciclo de reloj, comprendiendo el método además escalonar por la circuitería de ejecución cada fila posterior y columna posterior por un ciclo de reloj, e inferir por cada una de las unidades de procesamiento MxN direcciones de columna y fila de cada elemento de matriz A y matriz B recibido basándose en un ciclo de reloj y en una posición relativa de la unidad de procesamiento dentro de la matriz de procesamiento.

7. El método de la reivindicación 5 o 6, que comprende además generar y enviar por cada una de las unidades de procesamiento MxN, cuando no existe coincidencia, una solicitud de mantenimiento aguas arriba en una dirección del

elemento de datos que se está manteniendo, y generar y enviar por cada una de las unidades de procesamiento MxN una notificación de mantenimiento aguas abajo en la dirección del elemento de datos que se está manteniendo.

8. Un aparato que comprende:
medios: (105) para extraer (602) una instrucción de multiplicación de matrices de dispersión variable de formato variable, VFVSMM (107);
medios (109) para decodificar (604) la instrucción de VFVSMM (107) que tiene campos para especificar ubicaciones de matrices A, B y C que tienen elementos MxK, KxN y MxN, respectivamente; y
medios de ejecución (119, 208) que incluyen una matriz de procesamiento de unidades de procesamiento (M x N) (210), para ejecutar la instrucción de VFVSMM decodificada (111), reconfigurable para operar en uno de un modo denso-denso, modo denso-disperso y modo disperso-disperso,
en donde los medios de ejecución, que operan en el modo denso-denso, en respuesta a la instrucción de VFVSMM decodificada, están configurados para enrutar cada fila de la matriz A, escalonando filas posteriores, a una fila correspondiente de una matriz de procesamiento que tiene MxN unidades de procesamiento, y enrutar cada columna de la matriz B, escalonando columnas posteriores, a una columna correspondiente de la matriz de procesamiento, y
en donde cada una de las unidades de procesamiento MxN, que opera en el modo denso-denso, está configurada para:
generar K productos de elementos de matriz A y matriz B coincidentes recibidos de las matrices A y B, respectivamente, para que exista una coincidencia cuando el elemento de matriz B tiene la misma dirección de fila que una dirección de columna del elemento de matriz A; y
acumular cada producto generado con un elemento correspondiente de la matriz C que tiene una misma posición relativa que la de la unidad de procesamiento en la matriz de procesamiento,
en donde los medios para ejecutar, que operan en el modo denso-disperso, en respuesta a la instrucción de VFVSMM decodificada, están configurados para enrutar cada fila de la matriz A, escalonar filas posteriores, a la fila correspondiente de la matriz de procesamiento, y para enrutar cada columna de la matriz B en la columna correspondiente de la matriz de procesamiento, y
en donde cada una de las unidades de procesamiento MxN, que opera en el modo denso-disperso, está configurada para:
determinar si existe una coincidencia de dirección entre la dirección de fila lógica del elemento de matriz B y la dirección de columna del elemento de matriz A; y
cuando existe la coincidencia, generar el producto, y cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B cuando la dirección de columna del elemento de matriz A es mayor que la dirección de fila lógica del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A,
en donde los medios para ejecutar, que operan en el modo disperso-disperso, en respuesta a la instrucción de VFVSMM decodificada, están configurados para enrutar cada fila de la matriz A a la fila correspondiente de la matriz de procesamiento, y para enrutar cada columna de la matriz B a la columna correspondiente de la matriz de procesamiento, y
en donde cada una de las unidades de procesamiento MxN, que opera en el modo disperso-disperso, está configurada para:
determinar si existe una coincidencia entre la dirección de columna lógica del elemento de matriz A y la dirección de fila lógica del elemento de matriz B; y
cuando existe la coincidencia, generar el producto, y cuando no existe coincidencia, mantener el elemento de matriz A y pasar el elemento de matriz B cuando la dirección de columna lógica del elemento de matriz A es mayor que la dirección de fila lógica del elemento de matriz B y, de lo contrario, mantener el elemento de matriz B y pasar el elemento de matriz A,
en donde la instrucción de VFVSMM está adaptada además para especificar un formato de datos de cada uno de los elementos de datos de las matrices A, B y C especificadas, siendo el formato de datos uno de número entero, coma flotante de precisión media, coma flotante de precisión sencilla, coma flotante de precisión doble y un formato personalizado,
en donde cuando la instrucción de VFVSMM especifica un número entero de 8 bits como un formato de datos de cada uno de los elementos de datos de las matrices A, B y C especificadas, cada una de las unidades de procesamiento (210) en la matriz de procesamiento está configurada para realizar una multiplicación de matriz 2x2 cuando se procesan los elementos de datos de números enteros de 8 bits.

9. El aparato de la reivindicación 8, en donde la matriz B comprende únicamente elementos distintos de cero de una matriz dispersa que comprende lógicamente KxN elementos, incluyendo cada elemento un campo para especificar sus direcciones de fila y columna lógicas.

10. El aparato de la reivindicación 8;
en donde las matrices A y B son matrices dispersas que comprenden únicamente elementos distintos de cero de matrices MxK y KxN lógicas, respectivamente, con cada elemento que incluye un campo para especificar sus direcciones de fila y columna lógicas.

11. Almacenamiento legible por máquina que incluye instrucciones legibles por máquina, cuando se ejecutan, para implementar un método o realizar un aparato de acuerdo con cualquier reivindicación anterior.

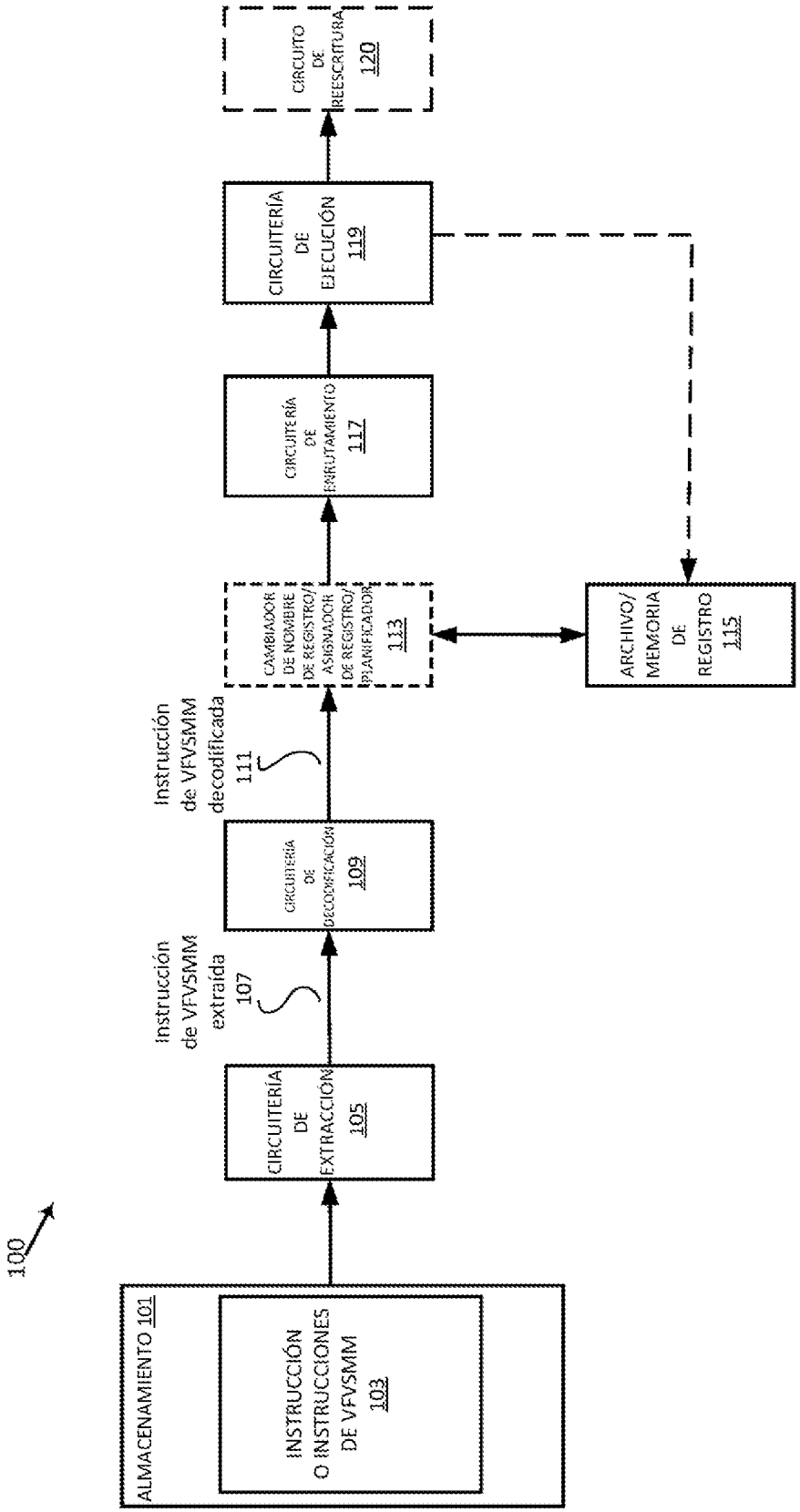

100
ALMACENAMIENTO 101
INSTRUCCIÓN O INSTRUCCIONES DE VFVSMM 103
CIRCUITERÍA DE EXTRACCIÓN 105
Instrucción de VFVSMM extraída 107
CIRCUITERÍA DE DECODIFICACIÓN 109
Instrucción de VFVSMM decodificada 111
CAMBIADOR DE NOMBRE DE REGISTRO/ ASIGNADOR DE REGISTRO/ DESAMBIGUADOR 113
ARCHIVO/ MEMORIA DE REGISTRO 115
CIRCUITERÍA DE ENRUTAMIENTO 117
CIRCUITERÍA DE EJECUCIÓN 119
CIRCUITO DE REESCRITURA 120

FIG. 1

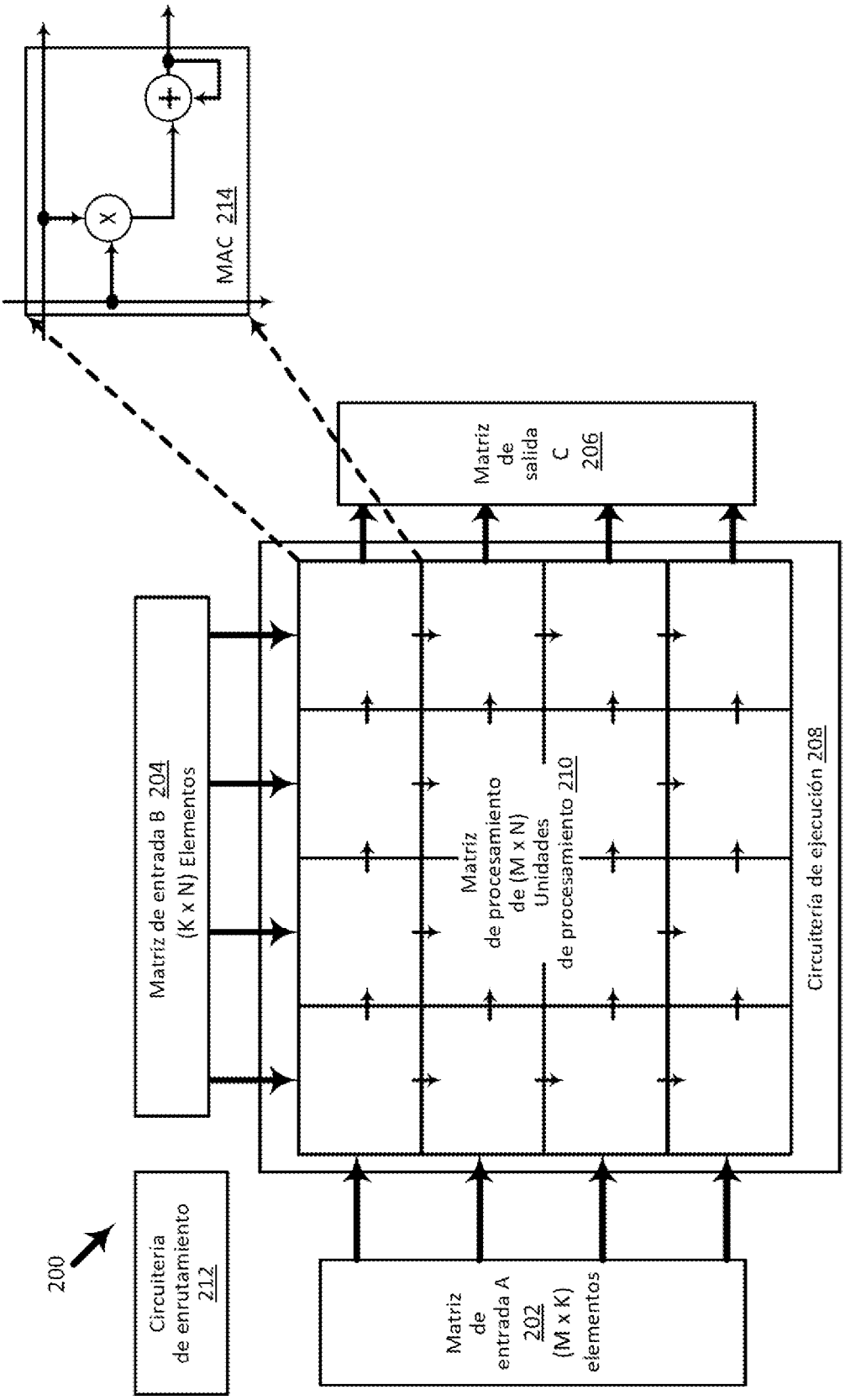

200
Circuitería de enrutamiento 212
Matriz de entrada A 202 (M x K) elementos
Matriz de entrada B 204 (K x N) Elementos
Matriz de procesamiento de (M x N) Unidades de procesamiento 210
Circuitería de ejecución 208
Matriz de salida C 206
MAC 214

FIG. 2

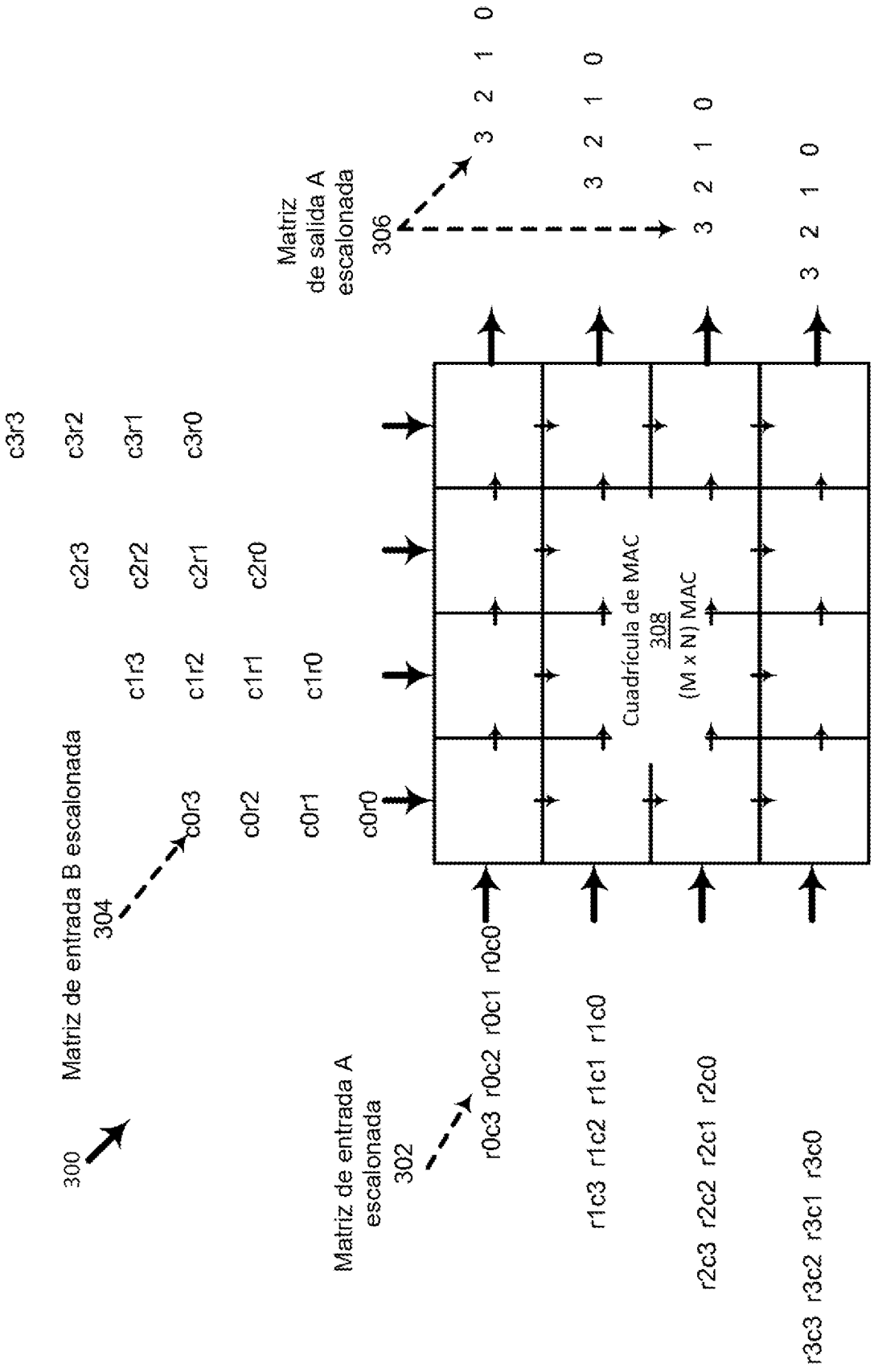

300
Matriz de entrada B escalonada
304
c3r3
c3r2
c3r1
c3r0
c2r3
c2r2
c2r1
c2r0
c1r3
c1r2
c1r1
c1r0
c0r3
c0r2
c0r1
c0r0
Matriz de entrada A escalonada
302
r0c3 r0c2 r0c1 r0c0
r1c3 r1c2 r1c1 r1c0
r2c3 r2c2 r2c1 r2c0
r3c3 r3c2 r3c1 r3c0
Cuadrícula de MAC
308
(M x N) MAC
Matriz de salida A escalonada
306
3 2 1 0
3 2 1 0
3 2 1 0
3 2 1 0

FIG. 3

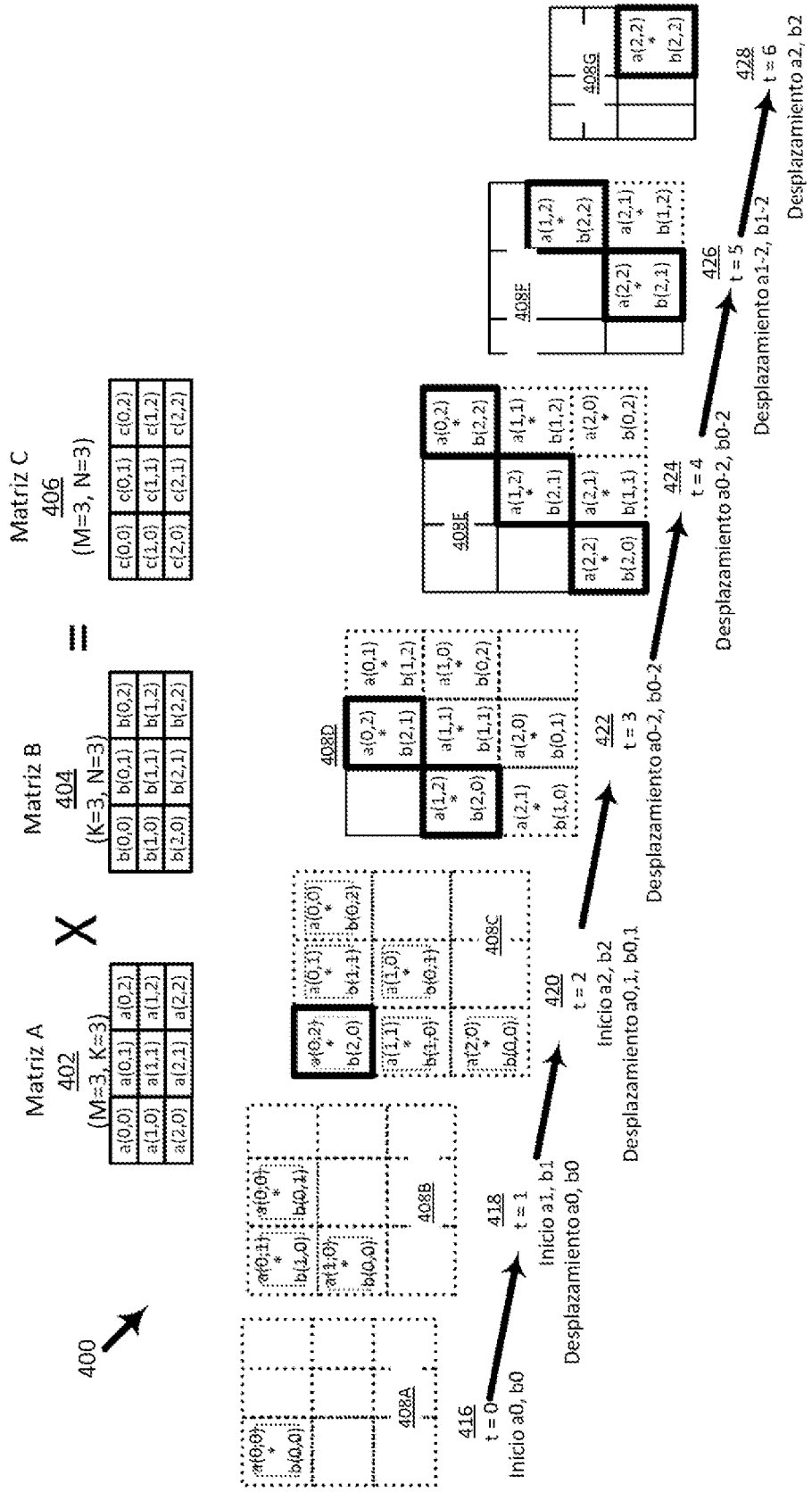

400
Matriz A
402
(M=3, K=3)
Matriz B
404
(K=3, N=3)
Matriz C
406
(M=3, N=3)
408A
408B
408C
408D
408E
408G
416
t = 0
Inicio a0, b0
418
t = 1
Inicio a1, b1
Desplazamiento a0, b0
420
t = 2
Inicio a2, b2
Desplazamiento a0,1, b0,1
422
t = 3
Desplazamiento a0-2, b0-2
424
t = 4
Desplazamiento a0-2, b0-2
426
t = 5
Desplazamiento a1-2, b1-2
428
t = 6
Desplazamiento a2, b2

FIG. 4

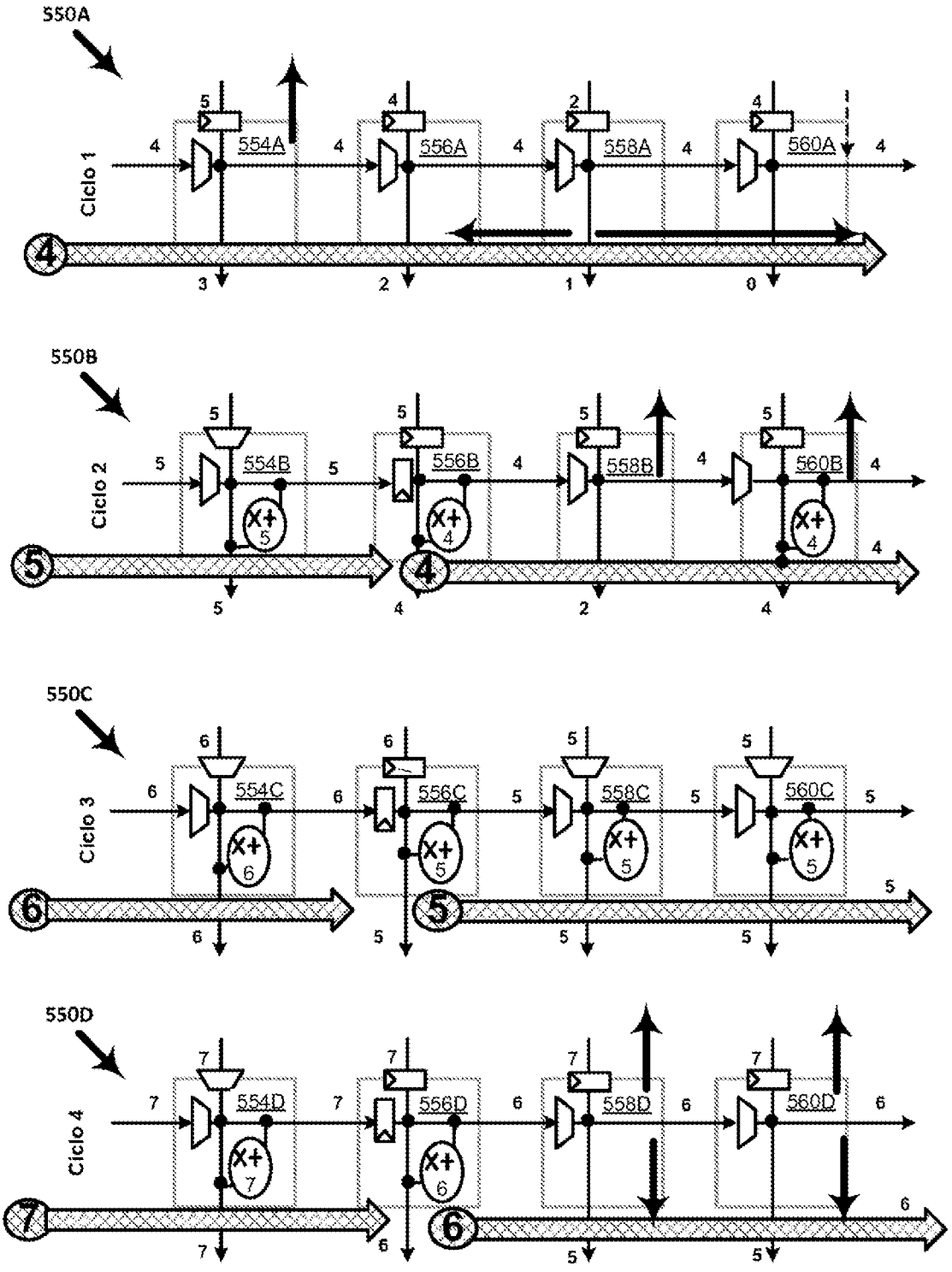

550A
Ciclo 1
554A
556A
558A
560A
550B
Ciclo 2
554B
556B
558B
560B
550C
Ciclo 3
554C
556C
558C
560C
550D
Ciclo 4
554D
556D
558D
560D

FIG. 5

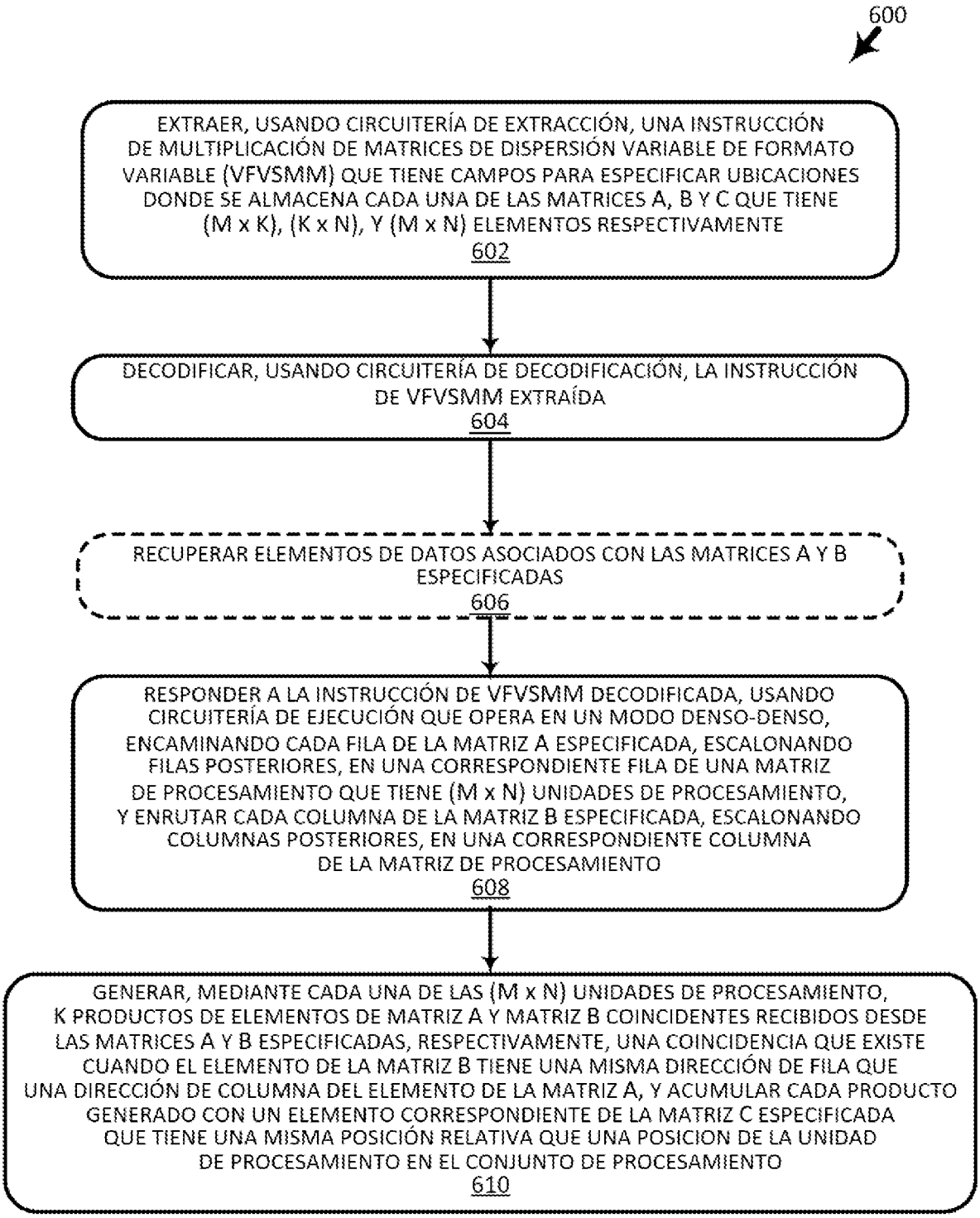

600
EXTRAER, USANDO CIRCUITERÍA DE EXTRACCIÓN, UNA INSTRUCCIÓN DE MULTIPLICACIÓN DE MATRICES DE DISPERSIÓN VARIABLE DE FORMATO VARIABLE (VFVSMM) QUE TIENE CAMPOS PARA ESPECIFICAR UBICACIONES DONDE SE ALMACENA CADA UNA DE LAS MATRICES A, B Y C QUE TIENE (M x K), (K x N), Y (M x N) ELEMENTOS RESPECTIVAMENTE
602
DECODIFICAR, USANDO CIRCUITERÍA DE DECODIFICACIÓN, LA INSTRUCCIÓN DE VFVSMM EXTRAÍDA
604
RECUPERAR ELEMENTOS DE DATOS ASOCIADOS CON LAS MATRICES A Y B ESPECIFICADAS
606
RESPONDER A LA INSTRUCCIÓN DE VFVSMM DECODIFICADA, USANDO CIRCUITERÍA DE EJECUCIÓN QUE OPERA EN UN MODO DENSO-DENSO, ENCAMINANDO CADA FILA DE LA MATRIZ A ESPECIFICADA, ESCALONANDO FILAS POSTERIORES, EN UNA CORRESPONDIENTE FILA DE UNA MATRIZ DE PROCESAMIENTO QUE TIENE (M x N) UNIDADES DE PROCESAMIENTO, Y ENRUTAR CADA COLUMNA DE LA MATRIZ B ESPECIFICADA, ESCALONANDO COLUMNAS POSTERIORES, EN UNA CORRESPONDIENTE COLUMNA DE LA MATRIZ DE PROCESAMIENTO
608
GENERAR, MEDIANTE CADA UNA DE LAS (M x N) UNIDADES DE PROCESAMIENTO, K PRODUCTOS DE ELEMENTOS DE MATRIZ A Y MATRIZ B COINCIDENTES RECIBIDOS DESDE LAS MATRICES A Y B ESPECIFICADAS, RESPECTIVAMENTE, UNA COINCIDENCIA QUE EXISTE CUANDO EL ELEMENTO DE LA MATRIZ B TIENE UNA MISMA DIRECCIÓN DE FILA QUE UNA DIRECCIÓN DE COLUMNA DEL ELEMENTO DE LA MATRIZ A, Y ACUMULAR CADA PRODUCTO GENERADO CON UN ELEMENTO CORRESPONDIENTE DE LA MATRIZ C ESPECIFICADA QUE TIENE UNA MISMA POSICIÓN RELATIVA QUE UNA POSICION DE LA UNIDAD DE PROCESAMIENTO EN EL CONJUNTO DE PROCESAMIENTO
610

FIG. 6

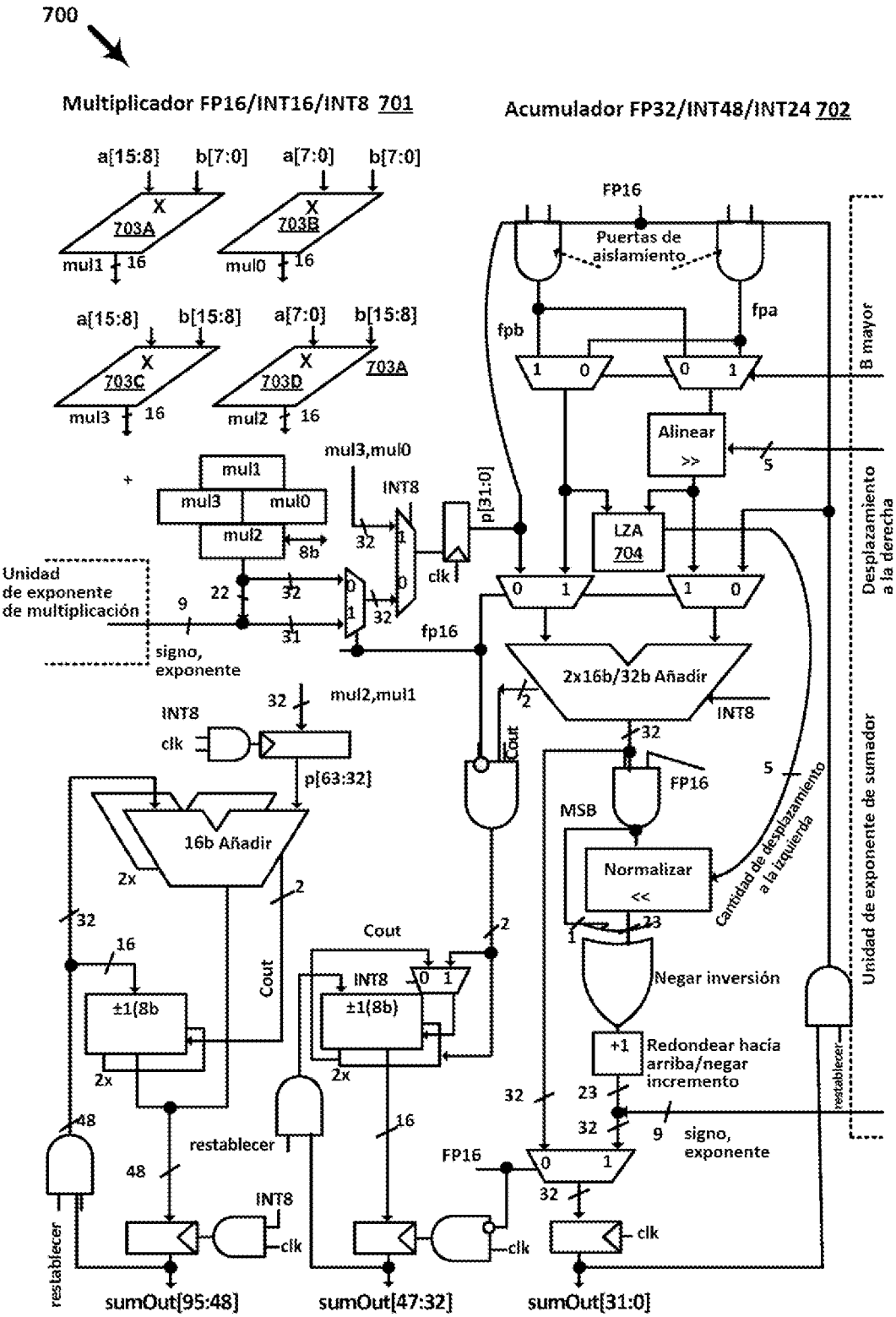

700
Multiplicador FP16/INT16/INT8 701
Acumulador FP32/INT48/INT24 702
a[15:8]
b[7:0]
a[7:0]
b[7:0]
703A
703B
mul1
mul0
16
a[15:8]
b[15:8]
a[7:0]
b[15:8]
703C
703D
703A
mul3
mul2
16
mul3,mul0
mul1
mul3
mul0
mul2
8b
INT8
32
p[31:0]
clk
Unidad de exponente de multiplicación
22
32
9
31
signo, exponente
fp16
mul2,mul1
INT8
clk
p[63:32]
16b Añadir
2x
Cout
±1(8b)
restablecer
48
sumOut[95:48]
sumOut[47:32]
sumOut[31:0]
FP16
Puertas de aislamiento
fpb
fpa
B mayor
Alinear
>>
5
LZA
704
Desplazamiento a la derecha
2x16b/32b Añadir
MSB
Normalizar
<<
Cantidad de desplazamiento a la izquierda
Negar inversión
+1
Redondear hacia arriba/negar incremento
23
signo, exponente
Unidad de exponente de sumador

FIG. 7

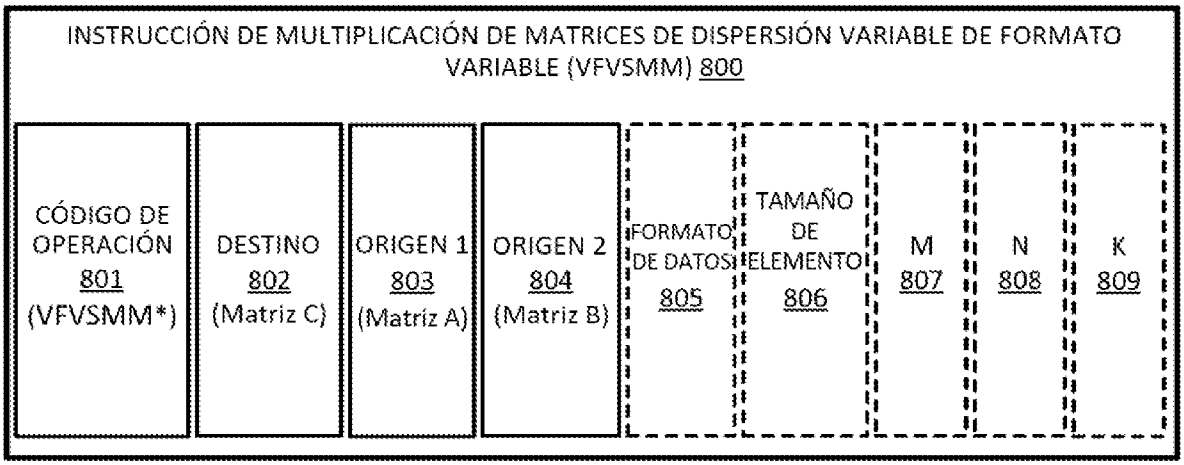

INSTRUCCIÓN DE MULTIPLICACIÓN DE MATRICES DE DISPERSIÓN VARIABLE DE FORMATO VARIABLE (VFVSMM) 800
CÓDIGO DE OPERACIÓN 801 (VFVSMM*)
DESTINO 802 (Matriz C)
ORIGEN 1 803 (Matriz A)
ORIGEN 2 804 (Matriz B)
FORMATO DE DATOS 805
TAMAÑO DE ELEMENTO 806
M 807
N 808
K 809

FIG. 8A

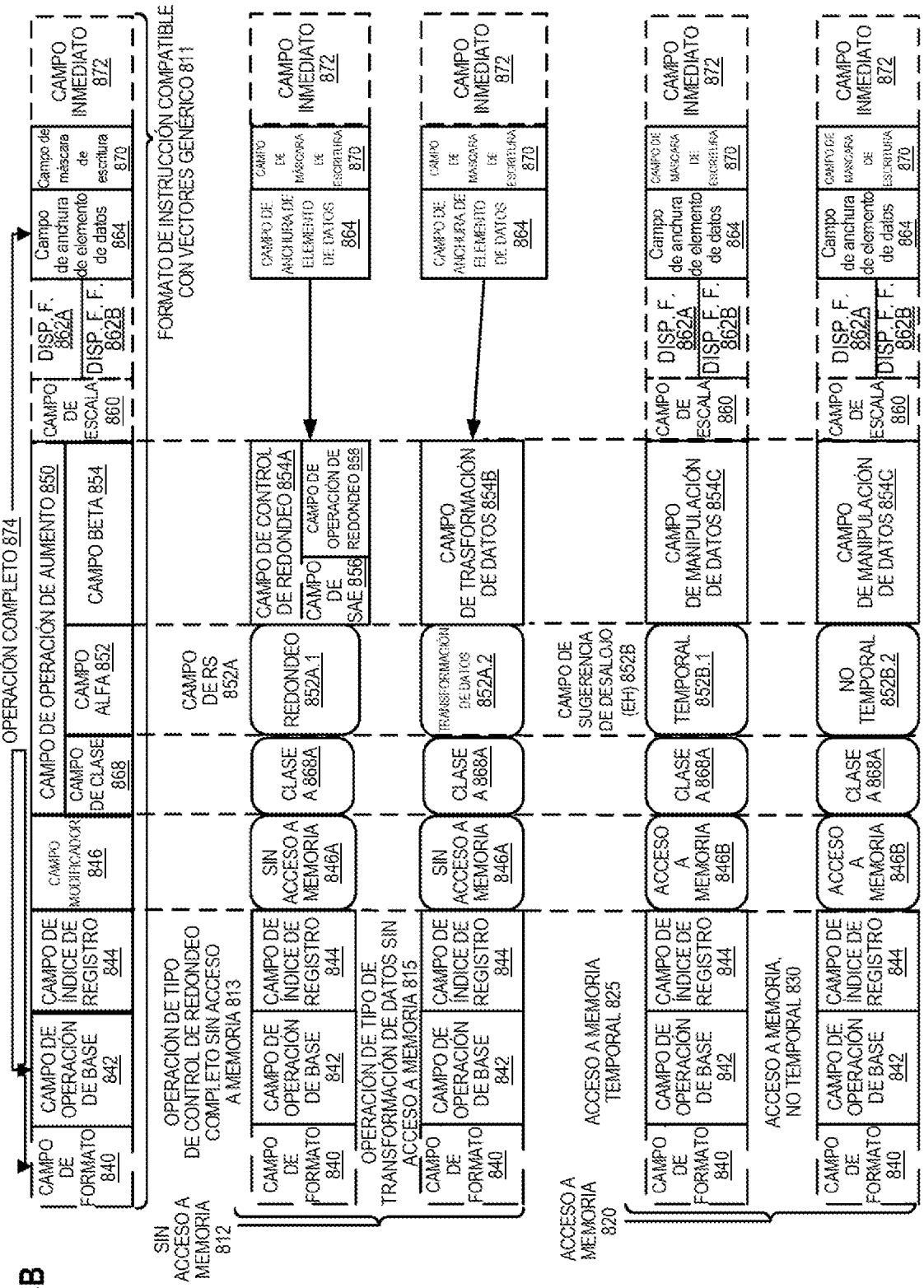
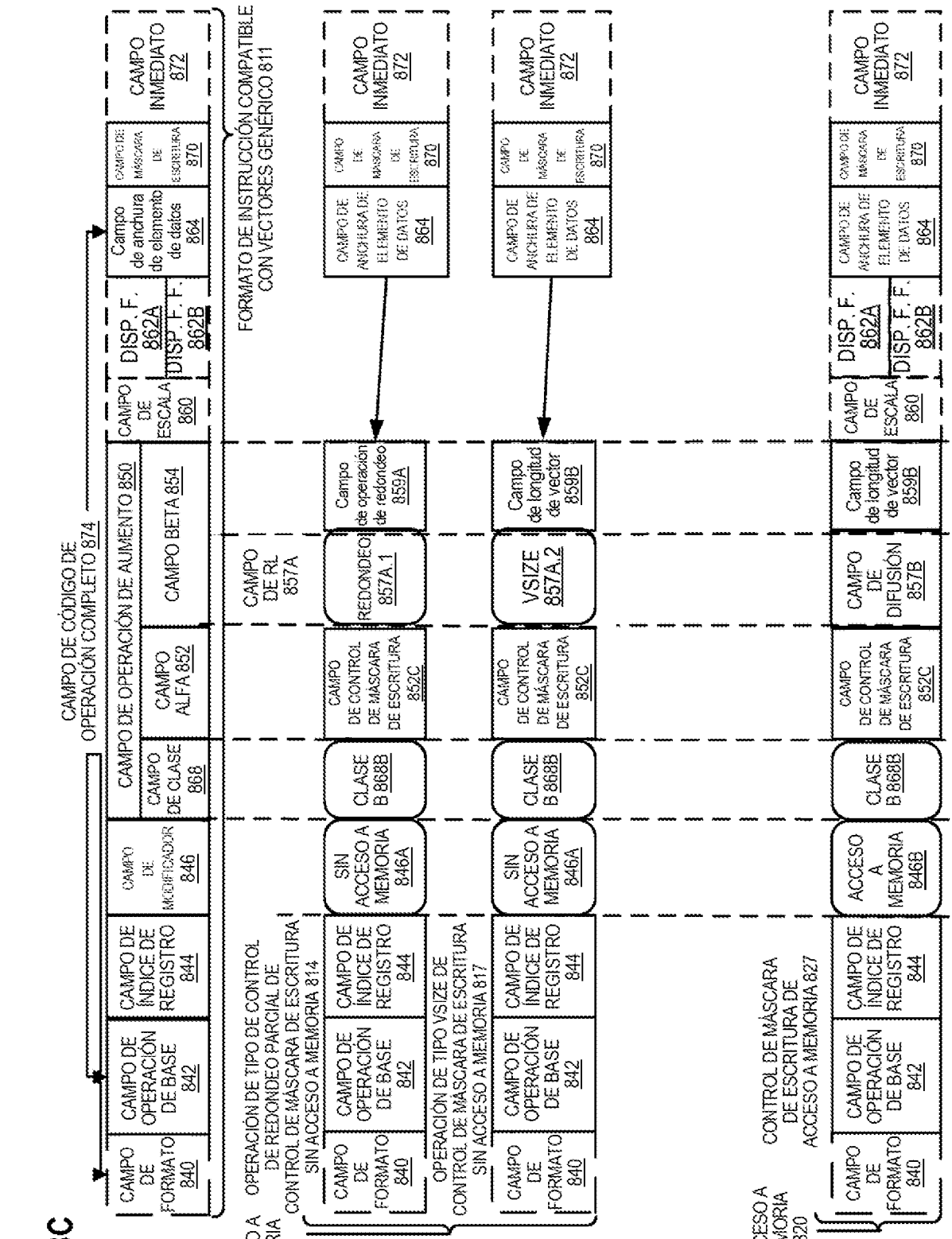

FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTORES GENÉRICO 811
CAMPO DE CÓDIGO DE OPERACIÓN COMPLETO 874
CAMPO DE FORMATO 840
CAMPO DE OPERACIÓN DE BASE 842
CAMPO DE ÍNDICE DE REGISTRO 844
CAMPO MODIFICADOR 846
CAMPO DE CLASE 868
CAMPO DE OPERACIÓN DE AUMENTO 850
CAMPO ALFA 852
CAMPO BETA 854
CAMPO DE ESCALA 860
DISP. F. 862A
DISP. F. F. 862B
Campo de anchura de elemento de datos 864
Campo de máscara de escritura 870
CAMPO INMEDIATO 872
ACCESO A MEMORIA 820
SIN ACCESO A MEMORIA 812
OPERACIÓN DE TIPO DE CONTROL DE REDONDEO COMPLETO SIN ACCESO A MEMORIA 813
SIN ACCESO A MEMORIA 846A
CLASE A 868A
CAMPO DE RS 852A
REDONDEO 852A.1
CAMPO DE CONTROL DE REDONDEO 854A
CAMPO DE SAE 856
CAMPO DE OPERACIÓN DE REDONDEO 858
CAMPO DE ANCHURA DE ELEMENTO DE DATOS 864
CAMPO DE MÁSCARA DE ESCRITURA 870
CAMPO INMEDIATO 872
OPERACIÓN DE TIPO DE TRANSFORMACIÓN DE DATOS SIN ACCESO A MEMORIA 815
TRANSFORMACIÓN DE DATOS 852A.2
CAMPO DE TRASFORMACIÓN DE DATOS 854B
ACCESO A MEMORIA TEMPORAL 825
CAMPO DE SUGERENCIA DE DESALOJO (EH) 852B
ACCESO A MEMORIA 846B
TEMPORAL 852B.1
CAMPO DE MANIPULACIÓN DE DATOS 854C
ACCESO A MEMORIA, NO TEMPORAL 830
NO TEMPORAL 852B.2

FIG.8B

FIG. 8C


CAMPO DE CÓDIGO DE OPERACIÓN COMPLETO 874
FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTORES GENÉRICO 811
CAMPO DE FORMATO 840
CAMPO DE OPERACIÓN DE BASE 842
CAMPO DE ÍNDICE DE REGISTRO 844
CAMPO MODIFICADOR 846
CAMPO DE OPERACIÓN DE AUMENTO 850
CAMPO DE CLASE 868
CAMPO ALFA 852
CAMPO BETA 854
CAMPO DE ESCALA 860
DISP. F. 862A
DISP. F. F. 862B
Campo de anchura de elemento de datos 864
CAMPO DE MÁSCARA DE ESCRITURA 870
CAMPO INMEDIATO 872
ACCESO A MEMORIA 820
SIN ACCESO A MEMORIA 812
OPERACIÓN DE TIPO DE CONTROL DE REDONDEO PARCIAL DE CONTROL DE MÁSCARA DE ESCRITURA SIN ACCESO A MEMORIA 814
OPERACIÓN DE TIPO VSIZE DE CONTROL DE MÁSCARA DE ESCRITURA SIN ACCESO A MEMORIA 817
CONTROL DE MÁSCARA DE ESCRITURA DE ACCESO A MEMORIA 827
SIN ACCESO A MEMORIA 846A
ACCESO A MEMORIA 846B
CLASE B 868B
CAMPO DE CONTROL DE MÁSCARA DE ESCRITURA 852C
CAMPO DE RL 857A
REDONDEO 857A.1
VSIZE 857A.2
Campo de operación de redondeo 859A
Campo de longitud de vector 859B
CAMPO DE DIFUSIÓN 857B

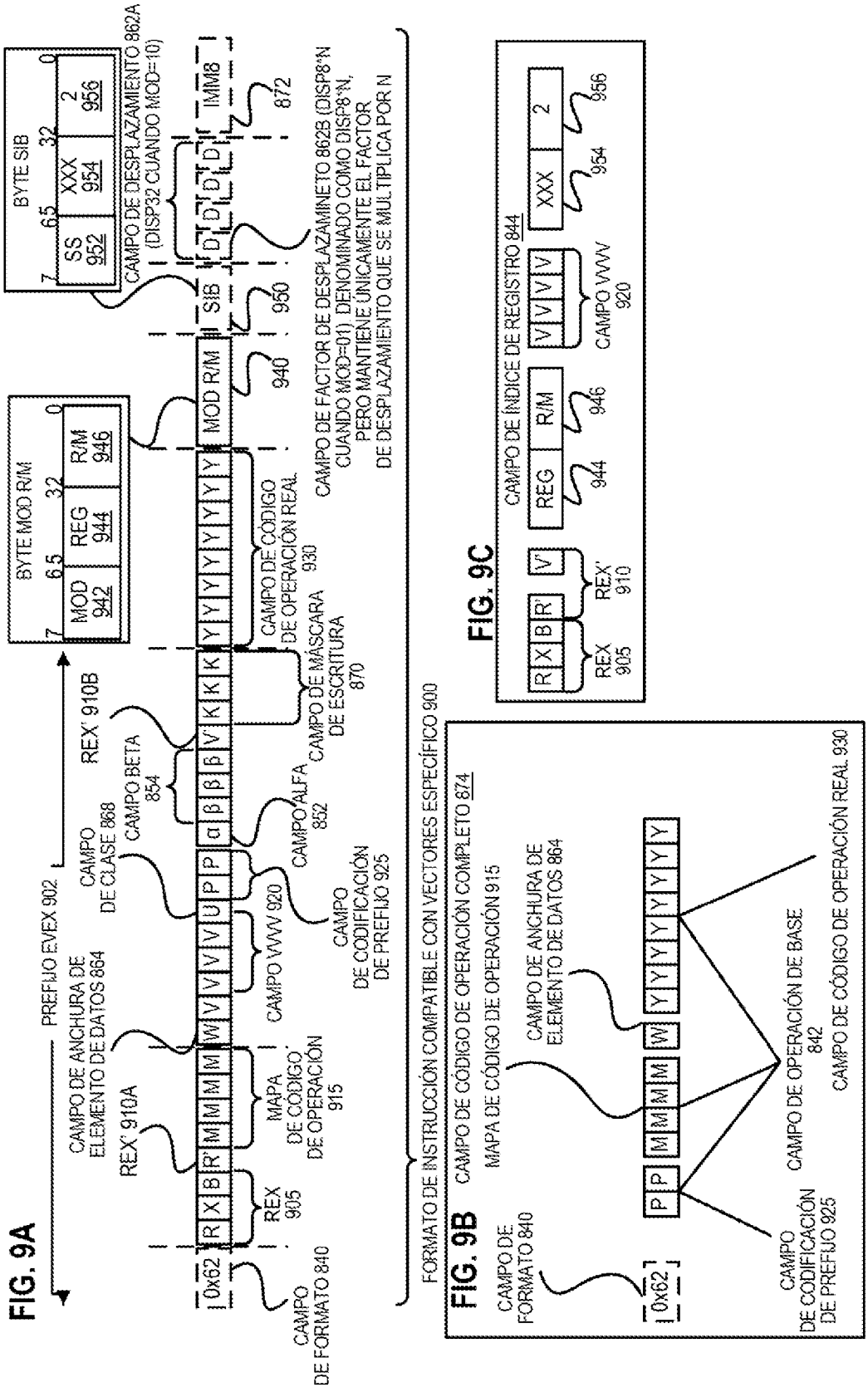


FIG. 9A
PREFIJO EVEX 902
CAMPO DE ANCHURA DE ELEMENTO DE DATOS 864
REX' 910A
CAMPO DE CLASE 868
CAMPO BETA 854
REX' 910B
BYTE MOD R/M
MOD 942
REG 944
R/M 946
BYTE SIB
SS 952
XXX 954
2 956
CAMPO DE DESPLAZAMIENTO 862A (DISP32 CUANDO MOD=10)
0x62
REX 905
MAPA DE CÓDIGO DE OPERACIÓN 915
CAMPO VVVV 920
CAMPO ALFA 852
CAMPO DE CODIFICACIÓN DE PREFIJO 925
CAMPO DE MÁSCARA DE ESCRITURA 870
CAMPO DE CÓDIGO DE OPERACIÓN REAL 930
CAMPO DE FACTOR DE DESPLAZAMIENTO 862B (DISP8*N CUANDO MOD=01) DENOMINADO COMO DISP8*N, PERO MANTIENE ÚNICAMENTE EL FACTOR DE DESPLAZAMIENTO QUE SE MULTIPLICA POR N
CAMPO DE FORMATO 840
SIB
IMM8
940
950
872
FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTORES ESPECÍFICO 900
FIG. 9B
CAMPO DE CÓDIGO DE OPERACIÓN COMPLETO 874
MAPA DE CÓDIGO DE OPERACIÓN 915
CAMPO DE ANCHURA DE ELEMENTO DE DATOS 864
CAMPO DE FORMATO 840
CAMPO DE CODIFICACIÓN DE PREFIJO 925
CAMPO DE OPERACIÓN DE BASE 842
CAMPO DE CÓDIGO DE OPERACIÓN REAL 930
FIG. 9C
CAMPO DE ÍNDICE DE REGISTRO 844
REX 905
REX' 910
REG 944
R/M 946
CAMPO VVVV 920
XXX 954
2 956

FIG. 9D

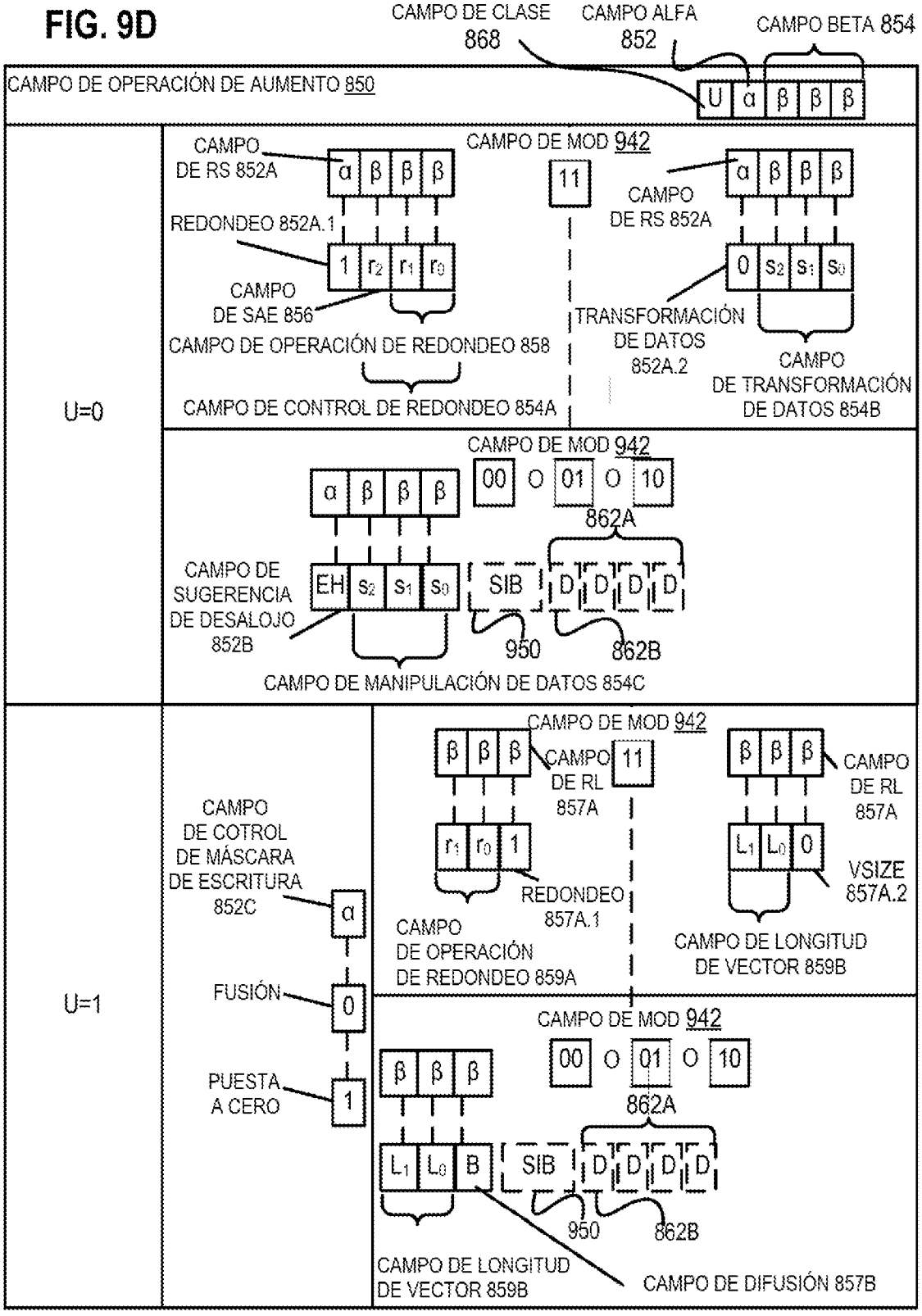
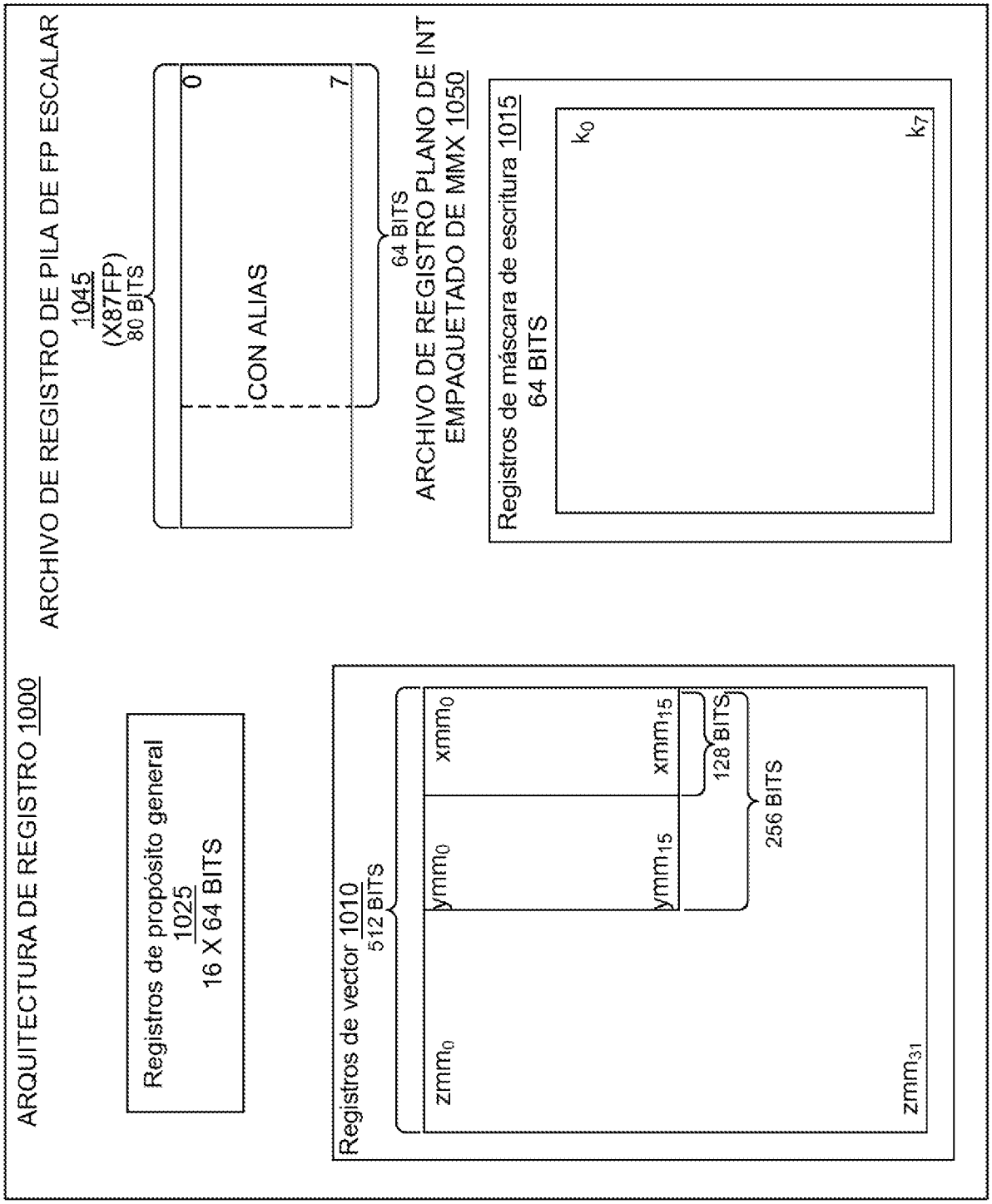

CAMPO DE CLASE 868
CAMPO ALFA 852
CAMPO BETA 854
CAMPO DE OPERACIÓN DE AUMENTO 850
U α β β β
U=0
CAMPO DE RS 852A
CAMPO DE MOD 942
11
REDONDEO 852A.1
1 r2 r1 r0
CAMPO DE SAE 856
CAMPO DE OPERACIÓN DE REDONDEO 858
CAMPO DE CONTROL DE REDONDEO 854A
0 s2 s1 s0
TRANSFORMACIÓN DE DATOS 852A.2
CAMPO DE TRANSFORMACIÓN DE DATOS 854B
00 O 01 O 10
862A
EH s2 s1 s0
SIB
D D D D
CAMPO DE SUGERENCIA DE DESALOJO 852B
950
862B
CAMPO DE MANIPULACIÓN DE DATOS 854C
U=1
CAMPO DE COTROL DE MÁSCARA DE ESCRITURA 852C
FUSIÓN
PUESTA A CERO
CAMPO DE RL 857A
r1 r0 1
REDONDEO 857A.1
CAMPO DE OPERACIÓN DE REDONDEO 859A
L1 L0 0
VSIZE 857A.2
CAMPO DE LONGITUD DE VECTOR 859B
L1 L0 B
CAMPO DE DIFUSIÓN 857B

FIG. 10


ARQUITECTURA DE REGISTRO 1000
Registros de propósito general 1025
16 X 64 BITS
ARCHIVO DE REGISTRO DE PILA DE FP ESCALAR 1045
(X87FP)
80 BITS
0
7
CON ALIAS
64 BITS
ARCHIVO DE REGISTRO PLANO DE INT EMPAQUETADO DE MMX 1050
Registros de máscara de escritura 1015
64 BITS
k0
k7
Registros de vector 1010
512 BITS
zmm0
ymm0
xmm0
ymm15
xmm15
128 BITS
256 BITS
zmm31

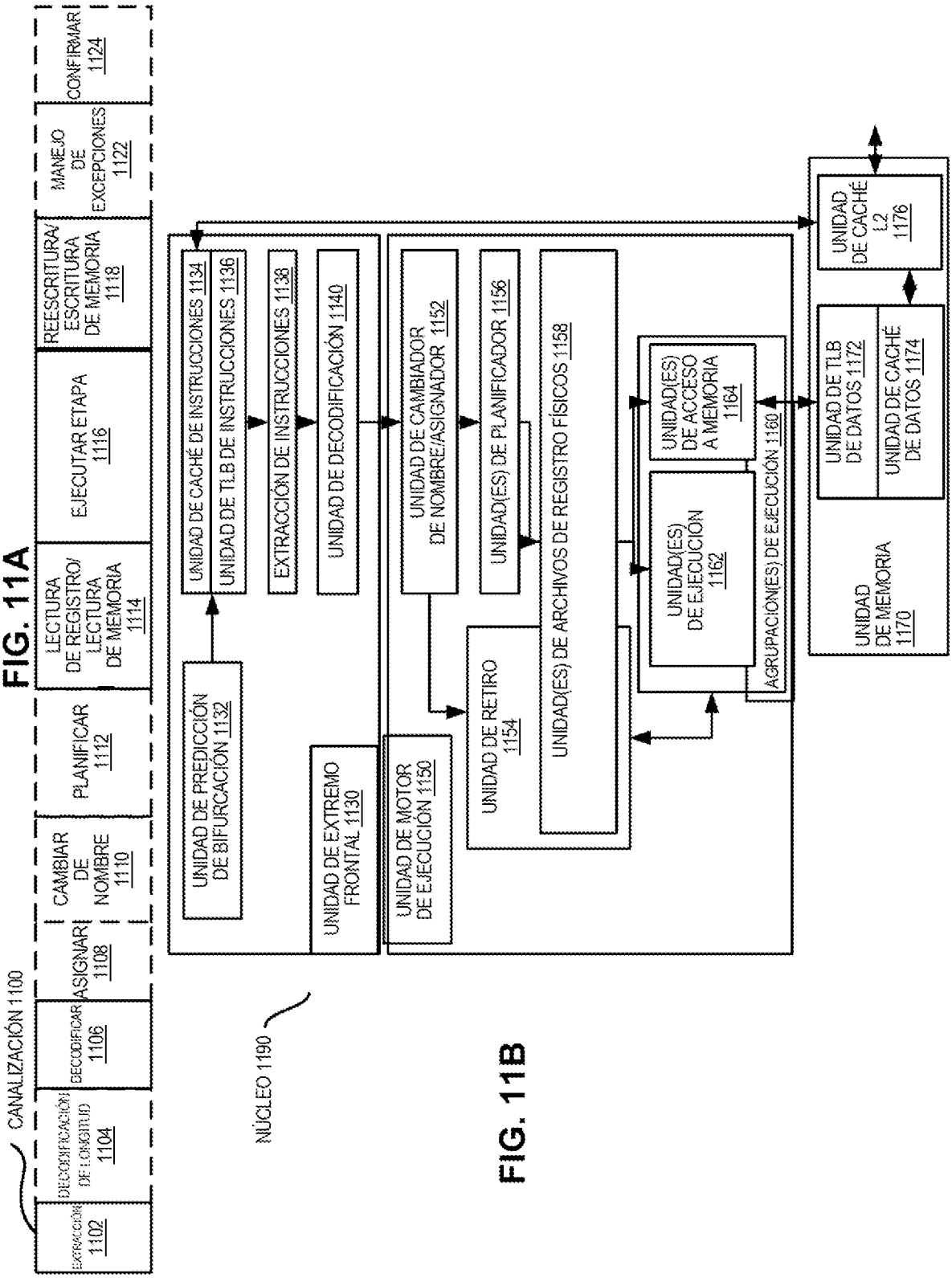


FIG. 11A

FIG. 11B

FIG. 12B

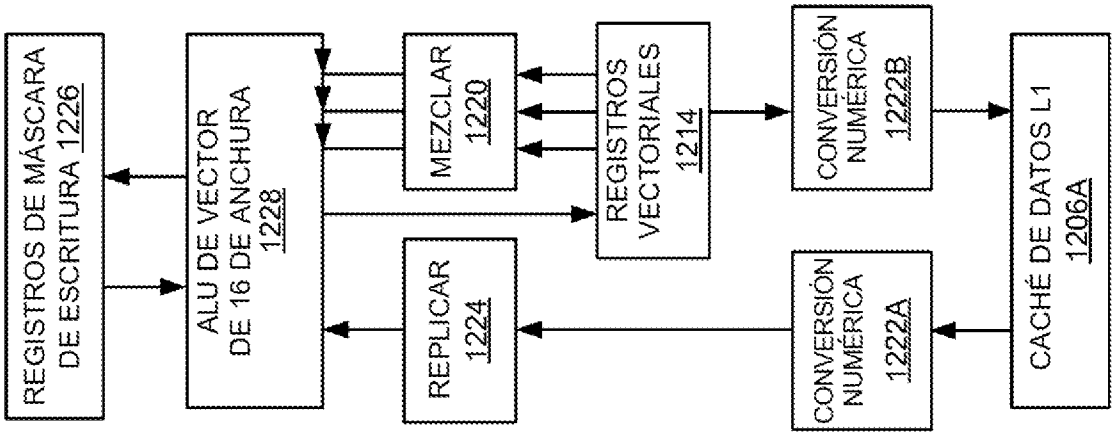
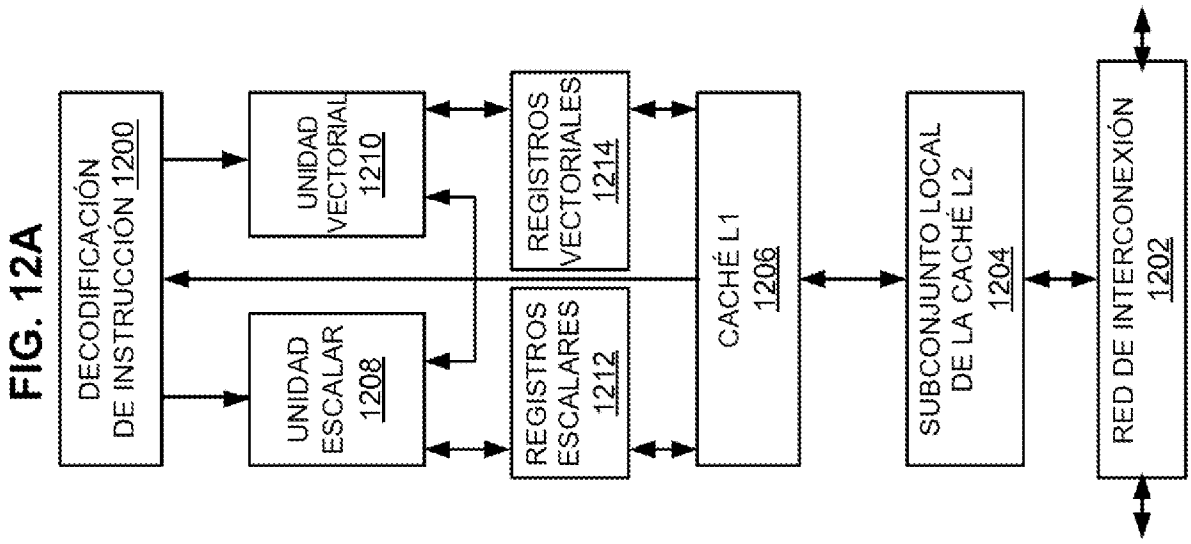


FIG. 12A



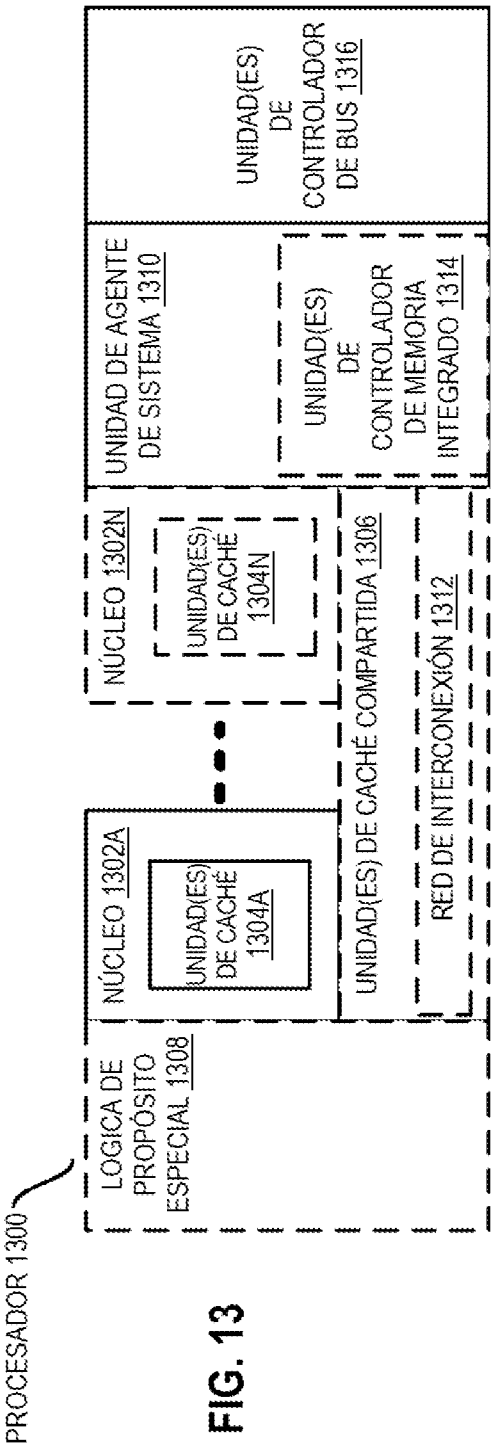

PROCESADOR 1300
LOGICA DE PROPÓSITO ESPECIAL 1308
NÚCLEO 1302A
UNIDAD(ES) DE CACHÉ 1304A
NÚCLEO 1302N
UNIDAD(ES) DE CACHÉ 1304N
UNIDAD(ES) DE CACHÉ COMPARTIDA 1306
RED DE INTERCONEXIÓN 1312
UNIDAD DE AGENTE DE SISTEMA 1310
UNIDAD(ES) DE CONTROLADOR DE MEMORIA INTEGRADO 1314
UNIDAD(ES) DE CONTROLADOR DE BUS 1316

FIG. 13

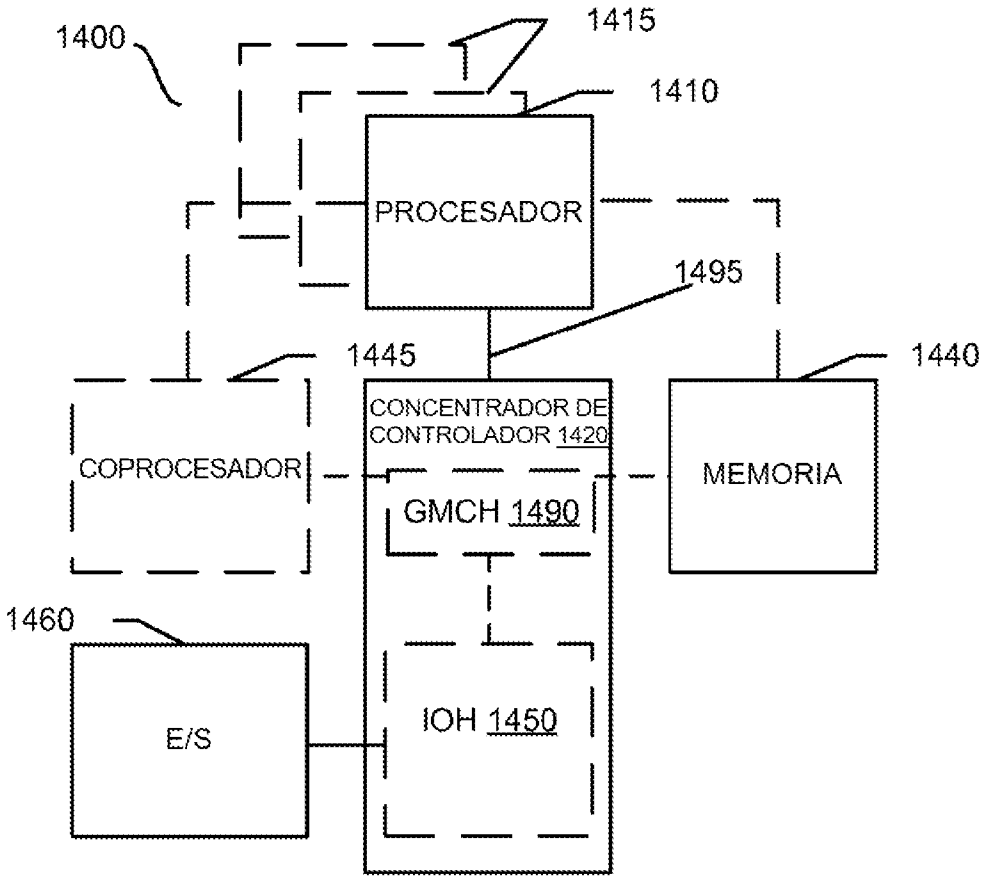

1400
1415
1410
PROCESADOR
1495
1445
1440
CONCENTRADOR DE CONTROLADOR 1420
COPROCESADOR
MEMORIA
GMCH 1490
1460
E/S
IOH 1450

FIG. 14

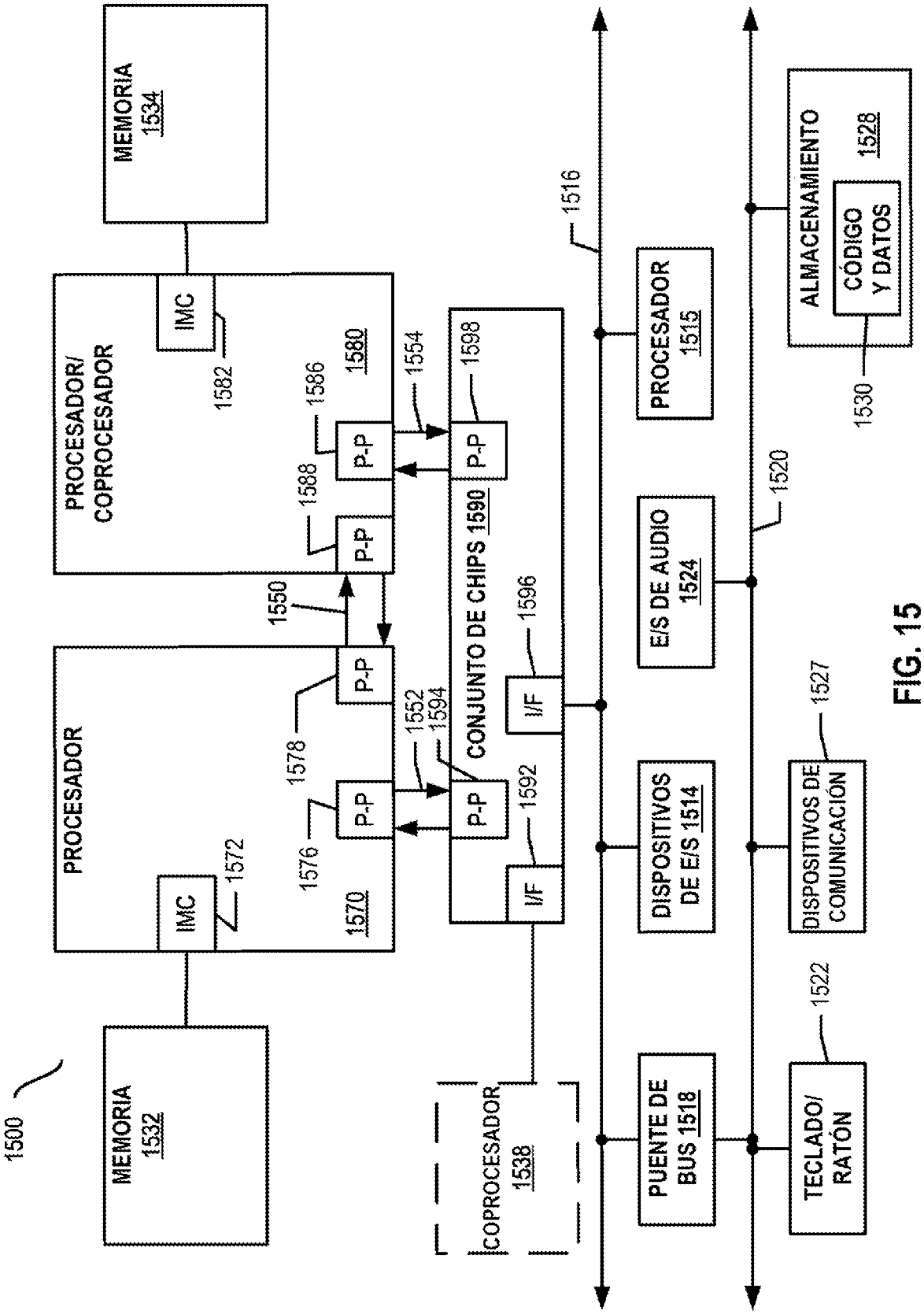

1500
MEMORIA 1532
MEMORIA 1534
PROCESADOR
IMC
1572
1570
1576
1578
P-P
P-P
1552
1594
1550
PROCESADOR/ COPROCESADOR
IMC
1582
1580
1586
1588
P-P
P-P
1554
CONJUNTO DE CHIPS 1590
1598
P-P
P-P
1592
I/F
1596
I/F
COPROCESADOR 1538
1516
PUENTE DE BUS 1518
DISPOSITIVOS DE E/S 1514
E/S DE AUDIO 1524
PROCESADOR 1515
1520
TECLADO/ RATÓN
1522
DISPOSITIVOS DE COMUNICACIÓN
1527
ALMACENAMIENTO
CÓDIGO Y DATOS
1528
1530

FIG. 15

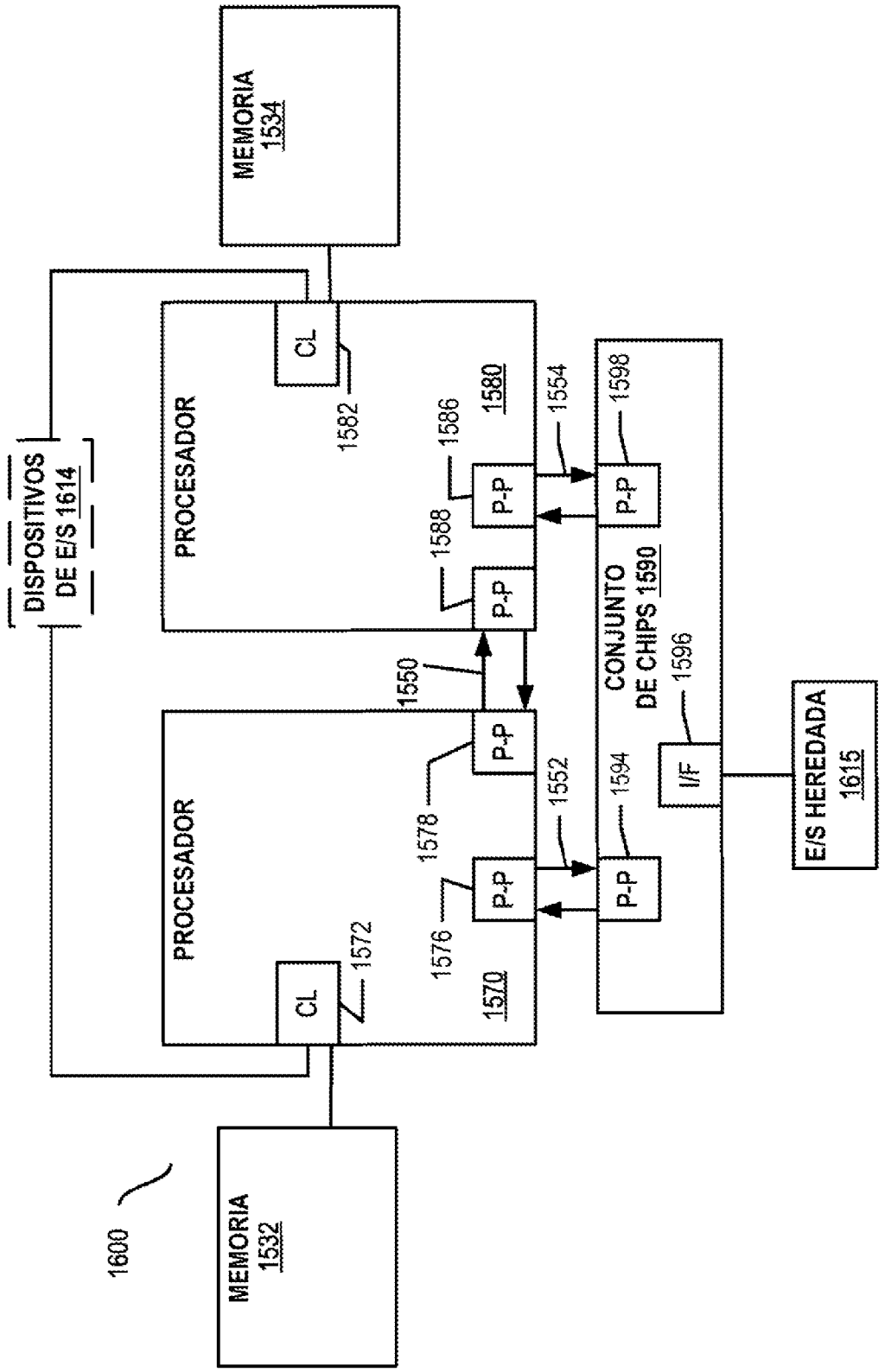

1600
MEMORIA 1532
MEMORIA 1534
DISPOSITIVOS DE E/S 1614
PROCESADOR
CL
1572
1582
1570
1580
1586
1588
1576
1578
1550
1552
1554
P-P
CONJUNTO DE CHIPS 1590
1594
1596
1598
I/F
E/S HEREDADA 1615

FIG. 16

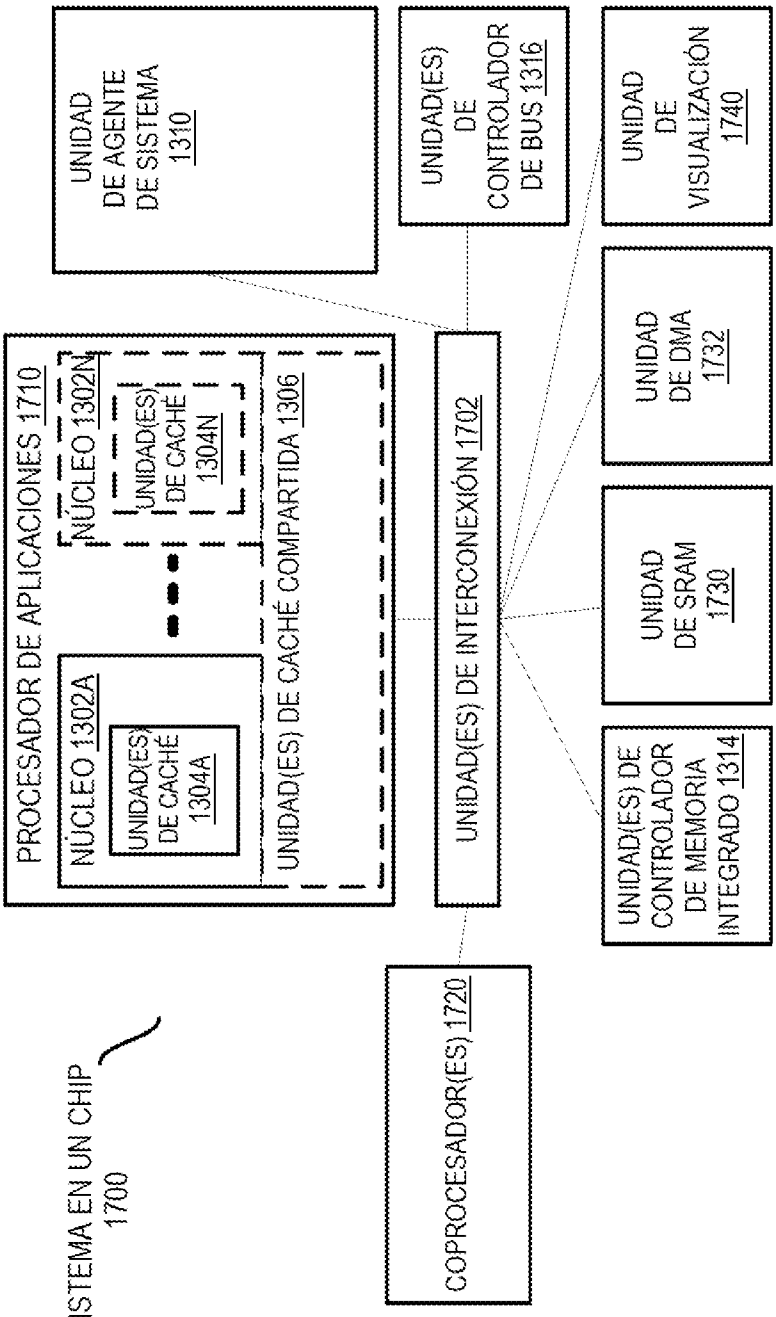

SISTEMA EN UN CHIP
1700
PROCESADOR DE APLICACIONES 1710
NÚCLEO 1302A
UNIDAD(ES) DE CACHÉ 1304A
NÚCLEO 1302N
UNIDAD(ES) DE CACHÉ 1304N
UNIDAD(ES) DE CACHÉ COMPARTIDA 1306
UNIDAD DE AGENTE DE SISTEMA 1310
UNIDAD(ES) DE CONTROLADOR DE BUS 1316
UNIDAD DE VISUALIZACIÓN 1740
UNIDAD DE DMA 1732
UNIDAD DE SRAM 1730
UNIDAD(ES) DE CONTROLADOR DE MEMORIA INTEGRADO 1314
UNIDAD(ES) DE INTERCONEXIÓN 1702
COPROCESADOR(ES) 1720

FIG. 17

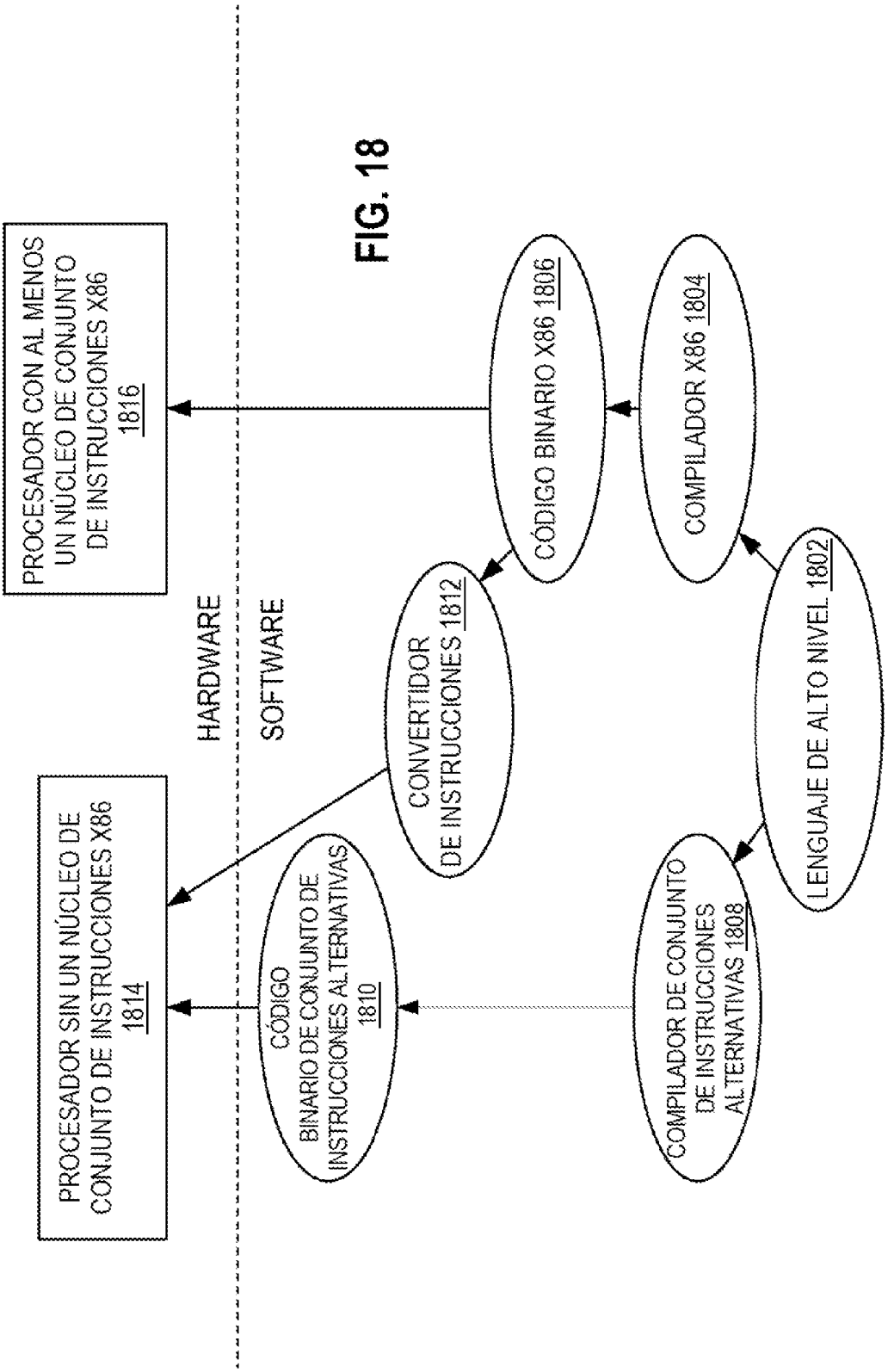

PROCESADOR CON AL MENOS UN NÚCLEO DE CONJUNTO DE INSTRUCCIONES X86 1816
HARDWARE
SOFTWARE
PROCESADOR SIN UN NÚCLEO DE CONJUNTO DE INSTRUCCIONES X86 1814
CÓDIGO BINARIO DE CONJUNTO DE INSTRUCCIONES ALTERNATIVAS 1810
CONVERTIDOR DE INSTRUCCIONES 1812
CÓDIGO BINARIO X86 1806
COMPILADOR X86 1804
COMPILADOR DE CONJUNTO DE INSTRUCCIONES ALTERNATIVAS 1808
LENGUAJE DE ALTO NIVEL 1802

FIG. 18