(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 688 603**

(51) Int. CI.:

G06F 9/38 (2008.01)
G06F 9/30 (2008.01)
G06F 15/80 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **17.09.2012 PCT/SE2012/050979**

(87) Fecha y número de publicación internacional: **25.04.2013 WO13058695**

(96) Fecha de presentación y número de la solicitud europea: **17.09.2012 E 12784087 (4)**

(97) Fecha y número de publicación de la concesión europea: **01.08.2018 EP 2751668**

(54) Título: **Procesador digital de señales y dispositivo de comunicación de banda base**

(30) Prioridad:

18.10.2011 SE 1150966

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
05.11.2018

(73) Titular/es:

MEDIATEK SWEDEN AB (100.0%)
Teknikringen 10
583 30 Linköping, SE

(72) Inventor/es:

NILSSON, ANDERS

(74) Agente/Representante:

IZQUIERDO BLANCO, María Alicia

DESCRIPCION

Procesador digital de señales y dispositivo de comunicación de banda base

Campo Técnico

La presente invención se refiere a un Procesador digital de señales basado en SIMT.

Antecedentes y Técnica Relacionada

Muchos dispositivos de comunicación móvil usan un transceptor de radio que incluye uno o más procesadores digitales de señales (DSP).

Para un rendimiento y fiabilidad aumentados, muchos terminales móviles usan actualmente un tipo de DSP conocido como procesador de banda base (BBP), para manejar muchas de las funciones de procesamiento de señales asociadas con el procesamiento de las señales de radio recibidas y la preparación de señales para su transmisión. Es ventajoso separar tales funciones del procesador principal, ya que son altamente dependientes del tiempo y pueden requerir un sistema operativo en tiempo real. Existe el deseo de que tales procesadores de banda base sean lo más flexibles posibles para adaptarse a estándares en desarrollo y permitir la reutilización de hardware. Por lo tanto, se han desarrollado procesadores de banda base programables, PBBP.

Muchas de las funciones realizadas frecuentemente en tales procesadores se realizan en grandes cantidades de muestras de datos. Por lo tanto, un tipo de procesador conocido como procesador De datos Múltiples de Instrucción Única (SIMD) es útil ya que permite que una única instrucción funcione en múltiples elementos de datos, en lugar de en un elemento de datos a la vez.

Como un desarrollo adicional de la arquitectura SIMD, se ha desarrollado la arquitectura de Tareas Múltiples de flujo de Instrucción Única (SIMT). Tradicionalmente, en la arquitectura SIMT se han proporcionado una o dos unidades de ejecución de vectores de tipo SIMD en asociación con una unidad de ejecución de enteros que puede ser parte de un procesador de núcleo.

La solicitud de patente internacional WO 2007/018467 divulga un DSP de acuerdo con la arquitectura SIMT, que tiene un procesador de núcleo que incluye un procesador de enteros y una memoria de programa, y dos unidades de ejecución de vectores que están conectadas, pero no integradas en el núcleo. Las unidades de ejecución de vectores pueden ser Unidades Aritméticas Lógicas Complejas (CALU) o Unidades de Multiplicación-Acumulación Complejas (CMAC). El núcleo tiene una memoria de programa para distribuir instrucciones a las unidades de ejecución. En la WO2007/018467cada una de las unidades de ejecución de vectores tiene un decodificador de instrucciones separado. Esto permite el uso de las unidades de ejecución de vectores de manera independiente entre sí, y de otras partes del procesador, de una manera eficiente.

Típicamente, la arquitectura del conjunto de instrucciones para el procesador SIMT puede incluir tres clases de instrucciones compuestas.

- Instrucciones RISC, que operan en operandos enteros de 16 bits. La clase de instrucción RISC incluye la mayoría de las instrucciones orientadas al control y se puede ejecutar dentro de la unidad de ejecución de enteros del núcleo del procesador.
- Instrucciones DSP, que operan en datos de valor complejo que tienen una parte real y una parte imaginaria. Las instrucciones DSP se pueden ejecutar en uno o más de los clústeres de SIMD.
- Instrucciones vectoriales. Las instrucciones vectoriales pueden considerarse extensiones de las instrucciones DSP ya que operan en conjuntos de datos grandes y pueden utilizar modos de direccionamiento avanzados y soporte vectorial.

Por lo tanto, la arquitectura SIMT ofrece tanto el rendimiento de computación a nivel de tarea y vectorial de SIMD tarea como suficiente flexibilidad de control RISC al mismo tiempo.

En una arquitectura SIMT, por lo tanto, hay varias unidades de ejecución. Normalmente, una instrucción puede ser emitida desde la memoria del programa a una de las unidades de ejecución en cada ciclo de reloj. Como las operaciones de vector operan típicamente en vectores grandes, una instrucción recibida en una unidad de ejecución de vectores durante un ciclo de reloj tardará un número de ciclos de reloj en ser procesada. En los ciclos de reloj siguientes, por lo tanto, pueden emitirse instrucciones a otras unidades de computación del procesador. Como las instrucciones de vectores se ejecutan en vectores largos, muchas instrucciones RISC se pueden ejecutar concurrentemente con la operación de vector.

Muchos algoritmos de banda base pueden descomponerse en cadenas de tareas de banda base más pequeñas con pocas dependencias de retroceso entre tareas. Esta propiedad no solo permite que se realicen

diferentes tareas en paralelo en unidades de ejecución de vectores, sino que también puede explotarse usando la arquitectura del conjunto de instrucciones anterior.

A menudo, para proporcionar sincronización de flujo de control y controlar el flujo de datos, pueden usarse instrucciones "inactivas" para detener el flujo de control hasta que se complete una operación de vector dada. La instrucción "inactiva" detendrá la carga de instrucciones adicionales hasta que se cumpla una condición particular. Tal condición puede ser la finalización de una instrucción de vector en una unidad de ejecución de vectores.

Típicamente, una tarea DSP comprenderá una secuencia de una a diez instrucciones, como se tratará con más detalle más adelante. Esto significa que la unidad de ejecución de vectores recibirá una instrucción vectorial, por ejemplo, para realizar un cálculo, y ejecutarlo en el vector de datos proporcionado hasta que se haga con el vector completo. La siguiente instrucción será procesar el resultado y almacenarlo en la memoria, lo que teóricamente puede ocurrir inmediatamente después de que se haya realizado el cálculo en el vector completo. A menudo, sin embargo, una unidad de ejecución de vectores tiene que esperar varios ciclos de reloj para su siguiente instrucción desde la memoria del programa ya que el núcleo del procesador está ocupado esperando a que se completen otras unidades de vectores, lo que lleva a una utilización ineficaz de la unidad de ejecución de vectores. La EP 0042442 se refiere a un sistema de procesamiento de información que comprende una unidad aritmética escalar y una unidad aritmética vectorial para la ejecución de instrucciones vectoriales.

Sumario de la invención

Es un objetivo de la presente invención hacer el procesamiento de instrucciones vectoriales en una arquitectura SIMT más eficiente.

Este objetivo se logra de acuerdo con la presente invención mediante una unidad de ejecución de vectores para su uso en un procesador digital de señales, dicha unidad de ejecución de vectores estando dispuesta para ejecutar instrucciones, incluyendo instrucciones vectoriales que se deben realizar en datos múltiples en forma de un vector, comprendiendo un controlador de vectores dispuesto para determinar si una instrucción es una instrucción vectorial y, si lo es, informar un registro de recuento dispuesto para mantener la longitud del vector, dicho controlador de vectores estando dispuesto además para controlar la ejecución de instrucciones, dicha unidad de ejecución de vectores estando caracterizada porque comprende una cola local dispuesta para recibir por lo menos una primera y una segunda instrucción de una memoria de programa y para mantener la segunda instrucción en la cola local hasta que se cumpla una condición predefinida, y porque el controlador de vectores comprende medios de control de cola dispuestos para controlar la cola local.

Preferiblemente, el controlador de vectores controla la ejecución de las instrucciones sobre la base de una señal de emisión recibida desde el núcleo. Alternativamente, la señal de emisión puede manejarse localmente por la misma unidad de ejecución de vectores.

Como la cola local provista para cada unidad de ejecución de vectores, puede proporcionarse un paquete de instrucciones que comprende varias instrucciones para una unidad de vectores a la unidad de vectores de una vez. Para habilitar la sincronización de instrucciones en la cola local para la ejecución de instrucciones vectoriales, se proporciona una instrucción, denominada instrucción SYNC, que pausará la lectura de instrucciones de la cola local, hasta que se cumpla una condición, típicamente que la ruta de datos esté lista para recibir y ejecutar otra instrucción. Estas dos características juntas permiten que se envíe una secuencia de instrucciones a la unidad de ejecución de vectores de una vez, para ser almacenada en la cola local y procesarse en secuencia en la unidad de ejecución de vectores de manera que tan pronto como la unidad de ejecución de vectores termine con una instrucción pueda comenzar en la siguiente. De esta manera cada unidad de ejecución de vectores puede trabajar con un mínimo de tiempo inactivo.

Por tanto, el procesamiento de acuerdo con la invención se hace más eficiente aumentando el paralelismo en el procesador, ya que las unidades de ejecución de vectores pueden trabajar más independientemente unas de las otras. La invención se basa en la idea de que en la técnica anterior una unidad de ejecución de vectores que ha terminado una instrucción vectorial a menudo no puede recibir inmediatamente la siguiente instrucción, ya que todas las unidades de ejecución de vectores reciben sus comandos de la misma cola, es decir, la memoria del programa en el núcleo del procesador. Esto sucederá cuando una unidad de ejecución de vectores esté lista para recibir un nuevo comando, mientras que el primer comando en la memoria del programa está destinado a otra unidad de ejecución de vectores que está ocupada. En este caso, ninguna unidad de ejecución de vectores puede recibir un nuevo comando hasta que la otra unidad de ejecución de vectores esté lista para recibir su próximo comando.

En una realización preferida, la unidad de ejecución de vectores comprende además

- Un registro de instrucciones dispuesto para recibir y almacenar instrucciones
- Un decodificador de instrucciones dispuesto para decodificar instrucciones almacenadas en el registro de instrucciones

- Una pluralidad de rutas de datos controladas por el decodificador de instrucciones.

Preferiblemente, la cola local está dispuesta para pausar la lectura de instrucciones hasta que la ruta de datos esté lista para recibir y ejecutar otra instrucción. Esto optimizará el manejo de la cola en la instrucción de vectores y el manejo general de las instrucciones en el procesador al que pertenece la unidad de ejecución de vectores.

Preferiblemente, el medio de control de cola comprende un controlador de cola dispuesto para mantener información de estado relacionada con la cola, como cuán completa está la cola, y para controlar el envío de instrucciones desde la cola local a la unidad de ejecución de vectores para su ejecución. El controlador de cola también puede estar dispuesto para, si se envía una nueva instrucción a la cola y la cola está llena, generar un mensaje de error.

El medio de control de cola puede estar dispuesto para emitir una señal específica que instruya a la cola local para pausar la lectura de instrucciones de la cola local hasta que se cumpla una condición específica, por ejemplo, que la ruta de datos esté lista para aceptar una nueva instrucción.

Preferiblemente, el controlador de vectores está dispuesto para hacer que se envíe una señal a una unidad de control de flujo de programa del procesador digital de señales para indicar que la unidad está lista para aceptar una nueva instrucción. El envío de esta señal puede basarse en la información enviada desde el decodificador de instrucciones al controlador de vectores acerca de la instrucción que se está ejecutando en un momento dado. La señal también puede basarse en el número de instrucciones actualmente en la cola, por ejemplo, si hay espacio para más instrucciones en la cola.

La invención también se refiere a un procesador digital de señales que comprende:

- un núcleo de procesador que incluye una unidad de ejecución de enteros configurada para ejecutar instrucciones de enteros; y
- por lo menos una primera y una segunda unidades de ejecución de vectores separadas de y acopladas al núcleo del procesador, en donde cada unidad de ejecución de vectores es una unidad de ejecución de vectores de acuerdo con lo anterior;
- dicho procesador digital de señales comprendiendo una memoria de programa dispuesta para contener instrucciones para la primera y la segunda unidades de ejecución de vector y una lógica de emisión para emitir instrucciones, incluyendo instrucciones de vectores, a la primera y segunda unidades de ejecución de vectores.

Tal procesador digital permitirá un uso más concurrente de sus unidades de ejecución de vectores, como se ha tratado anteriormente.

Típicamente, la memoria de programa está dispuesta en el núcleo del procesador y también está dispuesta para contener instrucciones para la unidad de ejecución de enteros.

La invención también se refiere a un dispositivo de comunicación de banda base adecuado para comunicación por cable e inalámbrica multimodo, que comprende:

- Una unidad de front-end configurada para transmitir y/o recibir señales de comunicación;
- Un procesador digital de señales programable acoplado a la unidad front-end analógica, en donde el procesador digital de señales programable es un procesador digital de señales de acuerdo con lo anterior.

En una realización preferida, las unidades de ejecución de vectores a las que se hace referencia a lo largo de este documento son unidades de ejecución de vectores de tipo SIMD o coprocesadores programables dispuestos para operar sobre vectores de datos.

La cola local puede ser una cola Primero en entrar Primero en salir (FIFO) con la longitud deseada, por ejemplo, de 4 a 8 instrucciones. También puede ser cualquier otro tipo de cola adecuada.

El procesador de acuerdo con las realizaciones de esta invención es particularmente útil para procesadores Digitales de Señales, especialmente procesadores de banda base. La unidad de front-end puede ser una unidad de front-end analógica dispuesta para transmitir y/o recibir señales de radiofrecuencia o banda de base.

Tales procesadores son ampliamente utilizados en diferentes tipos de dispositivos de comunicación, como teléfonos móviles, receptores de televisión y cable módem. En consecuencia, el dispositivo de comunicación de banda base puede estar dispuesto para la comunicación en una red de comunicaciones de móviles, por ejemplo, como un teléfono móvil o un dispositivo de comunicaciones de datos móviles. El dispositivo de comunicación de banda base también puede estar dispuesto para comunicación de acuerdo con otros estándares inalámbricos, como

Bluetooth o WiFi. También puede ser un receptor de televisión, un cable módem, un módem WiFI o cualquier otro tipo de dispositivo de comunicación que pueda enviar una señal de banda base a su procesador. Debe entenderse que el término "banda de base" solo se refiere a la señal manejada internamente en el procesador. Las señales de comunicación realmente recibidas y/o transmitidas pueden ser cualquier tipo adecuado de señales de comunicación, recibidas en conexiones por cable o inalámbricas. Las señales de comunicación se convierten por una unidad de front-end del dispositivo a una señal de banda base, de una manera adecuada.

Breve Descripción de los Dibujos

A continuación, se describirá con más detalle la invención, a modo de ejemplo, y con referencia a los dibujos adjuntos.

La Fig. 1 es una descripción general del sistema de un terminal móvil típico que incluye un procesador de banda base.
La Fig. 2 ilustra un ejemplo de la arquitectura SIMT.
La Fig. 3 es un diagrama de bloques del procesador de banda base de acuerdo con una realización de la invención.
La Fig. 4 es un diagrama que ilustra los canales de emisión de instrucciones de una realización del núcleo del procesador de la FIG. 2.
La Fig. 5 ilustra la lógica de emisión de instrucciones en procesadores SIMT
La Fig. 6 ilustra una unidad SIMT de acuerdo con la técnica anterior
La Fig. 7 ilustra una unidad SIMT que tiene las características adicionales de una realización general de la invención
La Fig. 8 ilustra una unidad SIMT de acuerdo con una realización preferida de la invención.
La Fig. 9 ilustra el principio de funcionamiento de la cola local de acuerdo con una realización de la invención.

Descripción detallada de las Realizaciones

La Figura 1 ilustra un ejemplo de terminal móvil 1 que comprende un procesador de banda base 3 que será el tema principal de esta solicitud. Como es común en la técnica, el terminal 1 comprende medios para recibir y transmitir señales de comunicación. En este ejemplo, esto incluye antenas 5 conectadas a una unidad front-end analógica 7, que comprende un convertidor analógico a digital ADC para la dirección de recepción y un convertidor digital a analógico DAC para la dirección de transmisión. La unidad front-end analógica 7 está conectada al procesador de banda base 3. El procesador de banda base 3 normalmente, pero no necesariamente, comprende un procesador de Corrección de Errores Directo (FEC) 9, para funciones de corrección de errores como entrelazado, decodificación de Viterbi, etc., como es común en la técnica. Típicamente, el procesador de banda base 3 está a su vez conectado a una unidad MAC 11, que a su vez está conectada a un procesador de aplicaciones 13.

Típicamente, pero no necesariamente, el terminal 1 tiene un bus y un subsistema de memoria 15 que interconectan el procesador de banda base, la unidad MAC 11 y el procesador de aplicaciones 13. El terminal también comprende interfaces periféricas 17 para la entrada/salida del usuario, incluyendo típicamente un teclado, un Interfaz de cámara e interfaces para conexiones a otras unidades, por ejemplo, una interfaz USB.

Como entenderá el experto en la técnica, la unidad front-end analógica puede disponerse para manejar cualquier tipo de señales entrantes y salientes incluyendo señales de radiofrecuencia, señales de banda base y otras y proporcionar una señal de banda base al procesador de banda base 3.

La Figura 2 ilustra un ejemplo de un procesador de banda base 200 de acuerdo con la arquitectura SIMT. El procesador 200 incluye un núcleo controlador 201 y una primera 203 y una segunda 205 unidad de ejecución de vectores, que se tratarán con más detalle a continuación. Una unidad FEC 206 como se ha tratado en la Figura 1 está conectada a la red en chip. En una implementación concreta, por supuesto, la unidad FEC 206 puede comprender varias unidades diferentes.

Una unidad de interfaz de host 207 proporciona conexión al procesador de host que se muestra en la Fig. 1 (no mostrado en la Fig. 2) Si hay un procesador MAC presente, como en la Fig. 1, el procesador MAC está conectado entre la unidad de interfaz de host 207 y el procesador de host. Una unidad front-end digital 209 proporciona conexión a la unidad ADC/DAC mostrada en la Fig. 1 de una manera bien conocida en la técnica.

Como es común en la técnica, el núcleo controlador 201 comprende una memoria de programa 211 así como lógica de emisión de instrucciones y funciones para soporte de múltiples contextos. Para cada contexto de ejecución, o hilo, soportado esto incluye un contador de programa, un puntero de pila y un archivo de registro (no mostrado explícitamente en la Figura 2) Por lo general, se soportan 2-3 hilos.

El núcleo de controlador 201 también comprende una unidad de ejecución de enteros 212 que comprende un archivo de registro RF, una memoria de núcleo de enteros ICM, una unidad multiplicadora MUL y una Unidad de

Aritmética y Lógica/Cambio (ALSU). La ALSU también se puede implementar como dos unidades, Unidad Aritmética Lógica y Unidad de Cambio. Estas unidades son conocidas en la técnica y no se muestran en la Figura 2.

La primera unidad de ejecución de vectores 203 en este ejemplo es una unidad de ejecución de vectores CMAC, que comprende un controlador de vectores 213, una unidad de carga/almacenamiento de vectores 215 y un número de rutas de datos 217. El controlador de vectores de esta primera unidad de ejecución de vectores está conectado a la memoria de programa 211 del núcleo de controlador 201 a través de la lógica de emisión, para recibir señales de emisión relacionadas con instrucciones de la memoria de programa. En la descripción anterior, la lógica de emisión decodifica la palabra de instrucción para obtener la señal de emisión y envía esta señal de emisión a la unidad de ejecución de vectores como una señal separada. También sería posible permitir que el controlador de vectores de la unidad de ejecución de vectores genere la señal de emisión localmente. En este caso, las señales de emisión se crean por el controlador de vectores en base a la palabra de instrucción de la misma manera que lo sería en la lógica de emisión.

Una segunda unidad de ejecución de vectores 205 es una unidad de ejecución de vectores CALU que comprende un controlador de vectores 223, una unidad de carga/almacenamiento de vectores 225 y una serie de rutas de datos 227. El controlador de vectores 223 de esta segunda unidad de ejecución de vectores también está conectado a la memoria de programa 211 del núcleo de controlador 201, a través de la lógica de emisión, para recibir señales de emisión relacionadas con instrucciones de la memoria de programa.

La función de las rutas de datos 217, 227 y las unidades de carga/almacenamiento de vectores 215, 225 se tratarán a continuación.

Podría haber un número arbitrario de unidades de ejecución de vectores, incluyendo solo unidades CMAC, solo unidades CALU o un número adecuado de cada tipo. También puede haber otros tipos de unidades de ejecución de vectores distintas a CMAC y CALU. Como se ha explicado anteriormente, una unidad de ejecución de vectores es un procesador que puede procesar instrucciones vectoriales, lo que significa que una única instrucción realiza la misma función para una series unidades de datos. Los datos pueden ser complejos o reales, y se agrupan en bytes o palabras y se empaquetan en un vector para ser operados por una unidad de ejecución de vectores. En este documento, las unidades CALU y CMAC se usan como ejemplos, pero debe tenerse en cuenta que las unidades de ejecución de vectores pueden usarse para realizar cualquier función adecuada en vectores de datos.

Para habilitar varias operaciones vectoriales concurrentes, el procesador preferiblemente tiene un sistema de memoria distribuida donde la memoria se divide en varios bancos de memoria, representados en la Figura 2 por el banco de memoria 0 230 al banco de memoria N 231. Cada banco de memoria 230, 231 tiene su propia memoria compleja 232, 233 y, unidad de generación de direcciones AGU 234, 235 respectivamente. Esta disposición junto con la red en chip mejora la eficiencia energética del sistema de memoria y el rendimiento del procesador ya que se pueden realizar cálculos de direcciones múltiples en paralelo. El PBBP de la Figura 2 también incluye uno o más bancos de memoria de enteros 238, que incluyen una memoria 239 y una unidad de generación de direcciones 240.

Como se conoce en la técnica, típicamente se conectan una serie de aceleradores 242, ya que permiten la implementación eficiente de ciertas funciones de banda base como codificación e intercalado de canales. Dichos aceleradores son bien conocidos en la técnica y no se tratarán aquí con detalle. Los aceleradores pueden ser configurables para ser reutilizados por muchos estándares diferentes.

Una red en chip 244 conecta el núcleo de controlador 201, la unidad front-end digital 209, la unidad de interfaz de host 207, las unidades de ejecución de vectores 203, 205, los bancos de memoria 230, 232, el banco de enteros 238 y los aceleradores 242.

Cada una de las unidades de ejecución de vectores 203, 205 comprende una unidad de carga/almacenamiento de vectores 215, 225 dispuesta para funcionar como una interfaz entre el puerto de red y la ruta de datos en la unidad de ejecución de vectores. Típicamente, las unidades de ejecución 203, 205 están conectadas a los bancos de memoria 230, 231 a través de la red 244, pero también se pueden soportar conexiones a otras unidades como los aceleradores 242 y otras unidades de ejecución de vectores. La función de carga se usa para obtener datos de las otras unidades conectadas a la red 244 (por ejemplo de un banco de memoria) y la función de almacenamiento se usa para almacenar datos de las unidades de ejecución 203, 205 , por ejemplo, a una unidad de memoria 230, 231 a través de la red 244. Los datos también pueden obtenerse de otras unidades de ejecución de vectores y/o los resultados de computación pueden enviarse a otras unidades de ejecución de vectores para su procesamiento adicional. Cada unidad de ejecución de vectores también comprende un controlador de vectores 213, 223 dispuesto para recibir instrucciones de la memoria de programa PM 211. Las unidades de carga de vectores 215, 225 pueden cargar datos usando dos modos diferentes. En el primer modo, pueden cargarse múltiples elementos de datos desde un banco de memorias 230, 232 u otras fuentes, como se ha tratado anteriormente. En el otro modo, los datos pueden cargarse un elemento de datos a la vez y luego distribuirse a las rutas de datos SIMD en una unidad de ejecución dada. El último modo puede usarse para reducir el número de accesos de memoria cuando se procesan datos consecutivos por la unidad de ejecución.

En la realización ilustrada, la segunda unidad de ejecución de vectores 205 se muestra como una ALU compleja de cuatro vías que puede incluir cuatro rutas de datos independientes 227 que tienen cada una un multiplicador-acumulador corto complejo (CSMAC) como es común en la técnica. Como se describirá con mayor detalle a continuación, la CALU 205 puede ejecutar instrucciones vectoriales. En una realización, la CALU 205 puede ser particularmente adecuada para ejecutar instrucciones vectoriales complejas. Además, cada uno de las rutas de datos independientes 227 de la CALU 205 puede ejecutar concurrentemente las instrucciones vectoriales complejas.

La primera unidad de ejecución de vectores 203 se muestra como una CMAC de cuatro vías con cuatro rutas de datos complejas que pueden ejecutarse concurrentemente o por separado. Las cuatro rutas de datos complejas incluyen multiplicadores, sumadores y registros de acumuladores (no todos se muestran en la FIG. 2) Por tanto, en esta realización, la CMAC 203 puede denominarse ruta de datos CMAC de cuatro vías. Además de multiplicar y sumar, la CMAC 203 también puede realizar operaciones de redondeo y escalado y soportar la saturación como se conoce en la técnica.

En una realización, las operaciones de CMAC 203 pueden dividirse en múltiples pasos de canalización. Además, cada una de las cuatro rutas de datos complejas 217 puede calcular una multiplicación y acumulación complejas en un ciclo de reloj. La CMAC 203 (es decir, las cuatro rutas de datos juntas) puede ejecutar una operación en un vector de N elementos en N/4 ciclos de reloj, para soportar computación vectorial compleja (por ejemplo, convolución compleja, convolución compleja conjugada y producto de punto vectorial complejo). Además, la CMAC 203 también puede soportar operaciones sobre valores complejos almacenados en los registros de acumuladores (por ejemplo, suma, resta, conjugación, etc. complejas). Por ejemplo, la CMAC 203, puede computar una multiplicación compleja como (AR + JAI)* (BR + JBI) en un ciclo de reloj y la acumulación compleja en un ciclo de reloj y soportar computación vectorial compleja (por ejemplo, convolución compleja, convolución compleja conjugada, y producto de punto vectorial complejo).

En una realización, la arquitectura del conjunto de instrucciones para el núcleo del procesador 201 puede incluir tres clases de instrucciones compuestas. La primera clase de instrucciones son instrucciones RISC, que operan en operandos de enteros de 16 bits. La clase de instrucción RISC incluye la mayoría de las instrucciones orientadas al control y puede ejecutarse dentro de la unidad de ejecución de enteros 212 del núcleo del procesador 201. La siguiente clase de instrucciones son las instrucciones DSP, que operan en datos de valor complejo que tienen una parte real y una parte imaginaria. Las instrucciones DSP pueden ejecutarse en una o más de las unidades de ejecución de vectores 203, 205. La tercera clase de instrucciones son las instrucciones vectoriales. Las instrucciones vectoriales pueden considerarse extensiones de las instrucciones DSP ya que operan en conjuntos de datos grandes y pueden utilizar modos de direccionamiento avanzados y soporte vectorial. Las instrucciones vectoriales pueden operar en tipos de datos complejos o reales.

La Fig. 3 es un diagrama de bloques del procesador de banda de base, PBBP 200, de acuerdo con una realización de la invención. El PBBP 200 incluye un núcleo de procesador que incluye una unidad de ejecución tipo RISC, y que está representado por la ruta 510 de datos RISC. El PBBP tiene además varias unidades de ejecución de vectores 520, 530 cada una incluyendo una unidad de control de vectores 275 respectivamente y una ruta de datos SIMD 525, 535, respectivamente. Como es común en la técnica, cada ruta de datos 525, 535 puede comprender varias rutas de datos. Típicamente, por ejemplo, la ruta de datos 525 tiene cuatro rutas de datos CMAC paralelas que juntas constituyen la ruta de datos 525.

Para proporcionar control sobre las múltiples unidades de ejecución de vectores, el hardware de núcleo 500 incluye una unidad de control de flujo de programa 501 acoplada a un contador de programa 502 que a su vez está acoplado a la memoria de programa (PM) 503. La PM 503 está acoplada al multiplexor 504, la unidad extracción de campo 508. El multiplexor 504 está acoplado al registro de instrucciones 505, que está acoplado al decodificador de instrucciones 506. El decodificador de instrucciones 506 está acoplado además al registro de señal de control (CSR) 507, que a su vez está acoplado al resto de la ruta de datos RISC 510.

De manera similar, cada una de las unidades de ejecución de vectores 520 y 530 también está dispuesta para recibir instrucciones de la memoria de programa 503 ubicada en el núcleo. Las unidades de ejecución de vectores incluyen registros de longitud de vectores 521, 531 respectivos, registros de instrucciones 522, 532, decodificadores de instrucciones 523, 533 y CSR 524, 534, que están acoplados a sus respectivas rutas de datos 525 y 535. Estas unidades y sus funciones se tratarán con más detalle, en la medida en que son relevantes para la invención, en relación con la Fig. 5.

La FIG. 4 es un ejemplo del manejo de las instrucciones de la técnica anterior de la memoria del programa a las varias unidades de ejecución, previsto como una ilustración del problema subyacente de la invención. La columna izquierda de la FIG. 4 representa el tiempo (en ciclos de reloj de ejecución). Las columnas restantes representan, de izquierda a derecha, los canales de ejecución de una primera y una segunda unidad de ejecución de vectores (más específicamente, las rutas de datos de CMAC 203 y CALU 205) y la unidad de ejecución de enteros y

la emisión de instrucciones de la misma. Más particularmente, en el primer ciclo de reloj, se emite una instrucción vectorial compleja (por ejemplo, CMAC.256) a CMAC 203. Como se muestra, la instrucción vectorial tarda muchos ciclos en completarse. En el siguiente ciclo de reloj, se emite una instrucción vectorial a CALU 205. En el siguiente ciclo de reloj, se emite una instrucción de enteros a la unidad de ejecución de enteros 510. En los siguientes ciclos, mientras se ejecutan las instrucciones vectoriales, pueden emitirse cualquier número de instrucciones de enteros a la unidad de ejecución de enteros 510. Se observa que aunque no se muestra, las unidades de ejecución de vectores restantes pueden también ejecutar concurrentemente instrucciones de manera similar.

En algunos casos, puede incluirse una instrucción "inactiva" en la secuencia de instrucciones, para evitar que el controlador de flujo del programa de núcleo obtenga instrucciones de la memoria del programa. Por ejemplo, para sincronizar el flujo del programa con la finalización de una instrucción vectorial, la instrucción "inactiva" puede usarse para suspender la obtención de instrucciones hasta que se haya cumplido una cierta condición. Típicamente, esta condición será que la unidad de ejecución de vectores en cuestión se haga con una instrucción vectorial anterior y pueda recibir una nueva instrucción. En este caso, el controlador de vectores 275 de la unidad de ejecución de vectores 520, 530 involucrada enviará una indicación, como un indicador, al controlador de flujo de programa 501 indicando que la unidad de ejecución de vectores está lista para recibir otra instrucción.

Las instrucciones inactivas pueden usarse para más de una unidad de ejecución de vectores al mismo tiempo. En este caso, no pueden enviarse más instrucciones desde la memoria de programa 503 hasta que cada una de las unidades de ejecución de vectores 520, 530 en cuestión haya enviado un indicador que indique que está listo para recibir una nueva instrucción.

En el ejemplo en la Fig. 4, la instrucción "inactiva" se emite después de las instrucciones de enteros mencionadas anteriormente. La instrucción inactiva se utiliza en este ejemplo para detener el flujo de control hasta que se complete la operación de vectores realizada por la CMAC 203.

El siguiente ejemplo se tratará sobre la base de un SIMT DSP con un número arbitrario de unidades de ejecución. Por simplicidad, en este ejemplo se supone que todas las unidades son unidades de ejecución de vectores CMAC, pero en la práctica se mezclarán y usarán juntas unidades de diferentes tipos.

En muchos algoritmos y programas de procesamiento de banda base, el algoritmo se puede descomponer en una serie de tareas DSP, cada una de las cuales consiste de un "prólogo", una operación vectorial y un "epílogo". El prólogo se usa principalmente para despejar acumuladores, configurar modos de direccionamiento y punteros y similares, antes de que se pueda realizar la operación vectorial. Cuando la operación vectorial se ha completado, el resultado de la operación vectorial puede procesarse adicionalmente por código en la parte "epílogo" de la tarea. En procesadores SIMT, típicamente solo se necesita una instrucción vectorial para realizar la operación vectorial.

El diseño típico de una tarea DSP se ejemplifica por la siguiente tarea de ejemplo de acuerdo con el estado de la técnica:
El fragmento de código en el ejemplo realiza un cálculo de producto de punto complejo sobre 512 valores complejos y luego almacena el resultado en la memoria nuevamente. La rutina requiere que las siguientes instrucciones sean obtenidas por el núcleo del procesador.

```
cmac0       ;Suponer que cmac0 está seleccionado
prólogo:    ;Configuración de la dirección
            ldi # 0, r0
            out r0, cdm0_addr
            out r0, cdm1_addr
            out  r0, cdm2_addr
            setcmvl. 512                ; Establecer la longitud del vector en 512
vectorop:   cmac [0],[1],[2]            ; Realizar la operación cmac sobre <longitud del vector>
                                        ; muestras
            idle # cmac0                ; Detener la obtención del programa hasta que cmac0
                                        esté listo
epílogo:    star [3]                    ; Almacenar acumulador
```

En el ejemplo anterior, las *instrucciones setcmvl, cmac* y *star se envían* y se ejecutan en la unidad de ejecución de vectores CMAC, mientras que las *instrucciones ldi, out* e *idle* se ejecutan en el núcleo de enteros ("núcleo").

La longitud del vector de las instrucciones del vector indica en cuántas palabras de datos (muestras) debe operar la unidad de ejecución de vectores. La longitud del vector puede establecerse de cualquier manera adecuada,

por ejemplo, una de las siguientes:

1) Por instrucciones dedicadas, como *setcmvl.123* en el ejemplo anterior
2) Llevarse en la misma instrucción, por ejemplo de acuerdo con el formato: *cmac.123,* como se muestra en la Fig. 4.
3) Establecerse por un registro de control, por ejemplo, de acuerdo con el formato *out r0, cmac_vector_length*

La instrucción *idle # cmac0* instruye al controlador de flujo del programa principal para dejar de obtener nuevas instrucciones hasta que la unidad CMAC0 haya terminado su operación vectorial. Después de que la función inactiva se libera y permite que se obtengan nuevas instrucciones, se obtiene la instrucción "star" y se envía a la unidad de ejecución de vectores CMACO. La instrucción star instruye a la unidad de ejecución de vectores CMAC que almacene el acumulador en la memoria.

En el siguiente ejemplo, que también ilustra la técnica anterior, se usan dos unidades de ejecución de vectores. La secuencia de instrucciones relacionada con la primera unidad de ejecución de vectores es la misma que la anterior:

```
.cmac0                  ; Suponer que cmac0 está seleccionado
prólogo:                ; Configuración de la dirección
    ldi # 0, r0
    out r0, cdm0_addr
    out r0, cdm1_addr
    out r0, cdm2_addr
    setcmvl.512                                  ; Establecer la longitud del vector en 512
vectorop:               cmac [0],[1],[2];        Realizar la operación cmac sobre <longitud del vector>
    idle #cmac0                                  ; muestras
                                                 ; Detener la obtención del programa hasta que cmac0 esté listo
Epílogo:                star [3];                Almacenar acumulador
```

La secuencia de instrucciones relacionada con la segunda unidad de ejecución de vectores es:

```
.cmac1   ; Suponer que cmac1 está seleccionado
prólogo: ; Configuración de la dirección
         ldi # 0, r0
         out r0, cdm3_addr
         out r0, cdm4_addr
         out r0, cdm5_addr
         setcmvl.2048                  ; Establecer la longitud del vector en 2048
vectorop: cmac [0],[1],[2]             ; Realizare la operación cmac sobre <longitud del vector>
                                       ; muestras
         idle #cmac1                   ; Detener la obtención del programa hasta que cmac0 esté listo
Epílogo:    star [3]                       ; Almacenar acumulador
```

En este caso, la segunda unidad de ejecución de vectores es instruida para realizar una operación vectorial de longitud 2048, que tomará 4 veces más tiempo que la operación de longitud 512 en la primera unidad de ejecución de vectores. La primera unidad de ejecución de vectores terminará por lo tanto antes que la segunda unidad de ejecución de vectores. Como la memoria del programa es instruida, mediante la instrucción *Idle #cmac1* para mantener la siguiente instrucción hasta que la segunda unidad de ejecución de vectores haya acabado, tampoco podrá enviar una nueva instrucción a la primera unidad de ejecución de vectores hasta que la segunda unidad de ejecución de vectores haya acabado. Por lo tanto, la primera unidad de ejecución de vectores estará inactiva durante más de 1000 ciclos de reloj debido a la instrucción *idle* relacionada con la segunda unidad de ejecución de vectores.

El ejemplo anterior usa dos unidades de ejecución de vectores. Como se comprenderá, este será un problema mayor cuanto mayor sea el número de unidades de ejecución de vectores, ya que una instrucción inactiva

relacionada con una unidad de ejecución de vectores particular afectará potencialmente a un número mayor de otras unidades de ejecución de vectores. De acuerdo con la invención, este problema se reduce proporcionando una cola local para cada unidad de ejecución de vectores. La cola local está dispuesta para recibir de la memoria del programa en el núcleo del procesador una o más instrucciones para que su unidad de ejecución de vectores se ejecute consecutivamente, y para enviar una instrucción a la vez a la ejecución de vectores.

Al mismo tiempo, se introduce un comando, que instruye a la cola local para mantener la siguiente instrucción hasta que se cumpla una condición particular. La condición puede ser, por ejemplo, que la unidad de ejecución de vectores finalice con el comando anterior o que la ruta de datos esté lista para recibir una nueva instrucción. Por el bien de la simplicidad, en este documento, este nuevo comando es referido SYNC. La condición puede indicarse en la palabra de instrucción a la instrucción SYNC, o puede leerse desde el archivo de registro de control o desde otra fuente.

A continuación se proporciona un ejemplo de una secuencia de instrucciones usando el nuevo comando SYNC:

```
.cmac0 ;Seleccionar cmac0 como destino de instrucciones relacionadas con cmac
       ;Configuración de la dirección
       ldi # 0, r0
       out r0, cdm0_addr
       out r0, cdm1_addr
       out r0, cdm2_addr
       setcmvl.512              ; Establecer la longitud del vector en 512
       cmac [0], [1], [2]       ; Realizar la operación cmac sobre 512 muestras
       sync                     ; Detener cola de programa hasta que cmac esté listo
       star [3]                 ; Almacenar acumulador
cmac1  ; Seleccionar cmac1 como destino de instrucciones relacionadas con cmac
       ;Configuración de la dirección
       ldi # 0, r0
       out r0, cdm3_addr
       out r0, cdm4_addr
       out r0, cdm5_addr
       setcmvl.2048             ; Establecer la longitud del vector en 2048
       cmac [0], [1], [2]       ; Realzar la operación cmac sobre 2048 muestras
       sync                     ; Detener cola de programa hasta que cmac esté listo
       star [3]                 : Almacenar acumulador
```

Al contrario que la técnica anterior, cada una de estas dos secuencias de comandos puede enviarse de una vez a la cola local de la unidad de ejecución de vectores afectada y almacenarse ahí mientras se espera que se envíe un comando en ese momento al decodificador de instrucciones dentro de la unidad de ejecución de vectores. Como se ha explicado anteriormente, el comando *sync* se proporciona para detener la cola local hasta que la unidad de ejecución de vectores haya acabado con el comando *cmac,* que es una instrucción vectorial y por lo tanto requiere varios ciclos de reloj para realizarse.

La Figura 5 ilustra la lógica de emisión de instrucciones en un procesador 700 de banda base de la técnica anterior que puede usarse como un punto de partida para la presente invención. El procesador de banda base comprende un núcleo RISC 701 que tiene una memoria de programa PM 702 que contiene instrucciones para las varias unidades de ejecución del procesador y una unidad de control de flujo de programa RISC 703. De la memoria de programa 702, las instrucciones se obtienen a una unidad lógica de emisión 705, que es común para todas las unidades de ejecución y está dispuesta para controlar dónde enviar cada instrucción específica. La lógica de emisión 705 corresponde a las unidades de extracción de Unidad de campo 508 y al control de emisión 509 de la Fig. 3. La lógica de emisión está conectada en este caso a una serie de unidades de ejecución de vectores 710, 712, 714 y a través de un multiplexor 715 a una unidad de núcleo RISC + ruta de datos 716, siendo esta última parte del núcleo RISC y correspondiente a las unidades 505, 506 , 507 y 510 de la Fig. 3. Como se ha explicado anteriormente, en una realización, las palabras de instrucción, que comprenden las instrucciones reales, se envían a todas las unidades de ejecución, mientras que la señal de emisión correspondiente a una instrucción particular se envía solo a la unidad de ejecución que ejecutará esta instrucción. En una realización alternativa, la señal de emisión es manejada localmente por cada unidad de ejecución de vectores.

La Figura 6 ilustra una unidad de ejecución de vectores 710, que puede ser una de las unidades de ejecución de vectores 710, 712, 714 de la Fig. 5, de acuerdo con la técnica anterior. La unidad de ejecución de vectores 710 tiene un controlador de vectores 720, un contador de longitud de vectores 721, un registro de instrucciones 722 y una unidad de decodificación de instrucciones 723. Como en la Figura 5 la unidad de ejecución de vectores 710 de la Figura 6 recibe instrucciones de la memoria del programa 702, aunque la Fig. 6 se ha simplificado. La palabra de instrucción es la instrucción real y se recibe en el registro de instrucciones 722 y se envía al decodificador de instrucciones 723. La señal de emisión se recibe en el controlador de vectores a través de la unidad lógica de emisión 705 y se usa para controlar la ejecución de la palabra de instrucción. Si la señal de emisión está activa, la instrucción se carga en el registro de instrucciones, se decodifica y ejecuta, de lo contrario se descarta. El controlador de vectores 720 también gestiona el contador de longitud de vectores 721 y otras señales de control usadas en el sistema, como se tratará a continuación.

Tradicionalmente, durante cada ciclo de reloj, una instrucción destinada a una de las unidades de ejecución, puede obtenerse de la memoria de programa 702. El campo de unidad en la palabra de instrucción puede extraerse de la palabra de instrucción y usarse para controlar a qué unidad de control se envía la instrucción. Por ejemplo, si el campo de la unidad es "000", la instrucción puede enviarse a la ruta de datos RISC. Esto puede hacer que la lógica de emisión 705 permita que la palabra de instrucción pase a través del multiplexor 715 en el núcleo de RISC 716 (no mostrado en la Fig. 6), mientras que no se carguen nuevas instrucciones en las unidades de ejecución del vectores en este ciclo. Sin embargo, si el campo de la unidad tiene cualquier otro valor, la lógica de emisión 705 puede habilitar la señal de emisión de instrucción correspondiente a la unidad de ejecución de vectores para la que está destinada. Luego el controlador de vectores 720 en la unidad de ejecución de vectores seleccionado permite que la palabra de instrucción pase a través del registro de instrucciones 722 de dicha unidad de ejecución de vectores. En ese caso, se enviará una instrucción NOP al registro de instrucciones de la ruta de datos RISC en el núcleo RISC 716.

Para manejar instrucciones vectoriales, cuando se envía una instrucción a las unidades de ejecución de vectores, el campo de longitud del vector de la palabra de instrucción puede extraerse y almacenarse en el registro de recuento 721. Este registro de recuento puede usarse para realizar un seguimiento de la longitud del vector en la instrucción vectorial correspondiente, y cuándo enviar el indicador que indica que la unidad de ejecución de vectores está lista para recibir otra instrucción. Cuando una unidad de ejecución de vectores correspondiente ha terminado la operación vectorial, el controlador de vectores 720 puede hacer que se envíe una señal (indicador) al control de flujo de programa 703 (no mostrado en la Fig. 6) para indicar que la unidad está lista para aceptar una nueva instrucción. El controlador de vectores 720 de cada unidad de ejecución de vectores 520, 530 (ver Fig. 3) puede crear adicionalmente señales de control para estados de prólogo y epílogo dentro de la unidad de ejecución. Tales señales de control pueden controlar VLU y VSU para operaciones vectoriales y también gestionar longitudes de vector impares, por ejemplo.

Cuando la lógica de emisión 705 determina, decodificando el campo de la unidad, que una instrucción particular debería enviarse a una unidad de ejecución de vectores particular, la palabra de instrucción se carga desde la memoria de programa 702 en el registro de instrucciones 722. Además, si la instrucción se determina (por el controlador de vectores) para llevar un campo de longitud de vector, el registro de recuento 721 se carga con este valor el valor de longitud de vector. El controlador de vector 720 decodifica partes de la palabra de instrucción para determinar si la instrucción es una instrucción de vector y lleva información de la longitud del vector. Si es así, el controlador de vectores 720 activa una señal para que el registro de recuento 721 cargue un valor que indica la longitud del vector en el registro de recuento 721. El controlador de vectores 720 también instruye a la unidad decodificadora de instrucciones 723 para que comience a descodificar la instrucción y comience a enviar señales de control a la ruta de datos 724. La instrucción en el registro de instrucciones 722 se decodifica luego por el decodificador de instrucciones 723, cuyas señales de control se mantienen en el registro de control de señales 724 antes de que se envíen a la ruta de datos. El registro de recuento 721 realiza un seguimiento del número de veces que se debe repetir la instrucción, que es la longitud del vector, de una manera convencional.

La Figura 7 ilustra una unidad de ejecución de vectores 810 de acuerdo con la invención. La unidad de ejecución de vectores comprende todos los elementos de la unidad de ejecución de vectores de la técnica anterior mostrada en la Figura 6 denotados por los mismos números de referencia. Además, la unidad de ejecución de vectores de acuerdo con la invención tiene una cola local 730 dispuesta para contener una serie de instrucciones recibidas de la memoria del programa. Un controlador de cola 732 dispuesto para controlar la cola local 730 está dispuesto en la unidad de control de vectores 720. La cola 730 y el controlador de cola 732 están conectados entre sí para intercambiar información y comandos. Por ejemplo, el controlador de cola 732 puede comprender un contador dispuesto para hacer un seguimiento del número de instrucciones en la cola 730. Alternativamente, la misma cola puede realizar un seguimiento de su estado y enviar información que indique que está llena, o vacía, o casi llena o vacía, al controlador de cola 732. Por tanto, el controlador de cola 732 contiene información de estado sobre la cola local 730 y puede enviar señales de control para comenzar, detener o vaciar la cola local 730. El decodificador de instrucciones 723 está dispuesto para informar al controlador de vectores 730 sobre que instrucción se está ejecutando actualmente.

Como se ha explicado anteriormente, muchas tareas de DSP se implementan como una secuencia de instrucciones, por ejemplo, un prólogo, una instrucción vectorial y un epílogo. Las instrucciones vectoriales se ejecutarán durante varios ciclos de reloj durante los cuales no se puede obtener ningún comando nuevo. En este caso, como se ha explicado anteriormente, la nueva instrucción SYNC se usa para hacer que la cola local mantenga la siguiente instrucción hasta que se cumpla una condición particular. Cuando se informa al controlador de cola 732 de que el decodificador de instrucciones 723 ha decodificado una instrucción de "sincronización", establecerá un modo en el controlador de cola 732 deteniendo la cola local 730 hasta que se cumpla la condición. Esto se implementa normalmente usando la información restante de longitud del vector restante y la información sobre la instrucción actual del decodificador de instrucciones. También pueden usarse indicadores que se envían desde la ruta de datos 724 al controlador de cola 732. Típicamente, la condición será que el procesamiento de la instrucción de vector se finalice de tal manera que el decodificador de instrucciones 723 en la unidad de ejecución de vectores esté listo para procesar la siguiente instrucción.

La cola local 730 podría ser cualquier clase de cola adecuada para contener el número deseado de instrucciones. En uno, es una cola FIFO capaz de contener un número apropiado, por ejemplo, 8 instrucciones.

La Figura 8 ilustra una unidad de ejecución de vectores 910 de acuerdo con una realización preferida de la invención. La unidad de ejecución de vectores que se muestra en la Figura 8 comprende las mismas unidades que en la Figura 7, interconectados de la misma manera. En esta realización, sin embargo, la cola local 740 es una cola cíclica adecuada para repetir un número especificado de instrucciones. Esto será particularmente ventajoso en implementaciones en las que la misma secuencia de instrucciones se va a ejecutar una gran cantidad de veces. El número de veces puede exceder algunas veces de 1000. En este caso, se puede guardar una cantidad significativa de ancho de banda en la ruta de control al no tener que enviar las mismas instrucciones desde la unidad de núcleo a la unidad de ejecución de vectores cada vez que se van a ejecutar.

Como en la Figura 7 hay un controlador de cola 732 dispuesto en el controlador de vectores 720'. En la realización de la Figura 8 también hay un administrador de búfer 744 dispuesto para hacer un seguimiento de las instrucciones que se van a repetir, y el número de veces que se debe repetir una instrucción. Con este propósito hay dos registros, que también están controlados por el controlador de vectores 720: un registro de repetición 746 para almacenar el número de repeticiones de la instrucción y un registro de recuento de instrucciones 748 dispuesto para contener el número de instrucciones que se van a repetir.

Como todas las instrucciones emitidas a la unidad de ejecución de vectores pasan la cola 740, es decir, el búfer cíclico, el búfer recordará las últimas N instrucciones (típicamente 8-16).
El registro de repetición 746 está configurado para contener el número de repeticiones a ser ejecutadas. El registro de repetición 746 puede cargarse por el archivo de registro de control o puede leerse de la palabra de instrucción emitida a la unidad de ejecución de vectores o mediante cualquier otro método.

El registro de recuento de instrucciones 748 está configurado para contener el número que indica cuántas instrucciones en el búfer cíclico 740 deben incluirse en el bucle de repetición. El registro de recuento de instrucciones puede cargarse por el archivo de registro de control o puede leerse de la palabra de instrucción emitida a la unidad de ejecución de vectores o por cualquier otro método.

Cuando se emite una instrucción de "repetición" o instrucción con un conjunto de "indicador de repetición" a la unidad de ejecución de vectores, el decodificador de instrucciones 723 junto con el controlador de vectores 720 instruye al controlador de cola 732 que envíe instrucciones desde el búfer cíclico 740 al registro de instrucciones 722.

Como en la Fig. 7, cuando el decodificador de instrucciones 723 encuentra una instrucción de "sync", el decodificador de instrucciones instruye al controlador de cola 732 que deje de obtener instrucciones de la cola cíclica local hasta que se haya tenido lugar una condición predefinida. Esta condición es típicamente que la instrucción anterior que se obtuvo de la cola se haya completado de tal manera que el decodificador esté listo para recibir una nueva instrucción.

Aunque la cola local 730, 740 y el registro de instrucciones 722 se muestran en este documento como entidades separadas, sería posible combinarlos en una unidad. Por ejemplo, el registro de instrucciones 722 podría integrarse como el último elemento de la cola local.

El administrador de búfer 744 supervisa el funcionamiento del búfer local 740 y gestiona la repetición de las instrucciones almacenadas actualmente en el búfer circular, mientras que el controlador de cola 732 gestiona el inicio/parada del envío de instrucciones desde el búfer/cola circular 740.

El administrador de búfer 744 gestiona además el registro de repetición 746 y realiza un seguimiento de cuántas repeticiones se han realizado. Cuando se ha realizado el número de repeticiones especificadas en el

registro de repetición 746, se envía una señal al controlador de vector 720' que luego puede enviarse al control de flujo de programa enviado 703 (no mostrado en la Fig. 8) para indicar que la operación está completa.

Cuando se ha llevado a cabo el número de repeticiones solicitadas, el comportamiento del búfer circular 740 vuelve al valor predeterminado de funcionalidad de cola, almacenando las últimas instrucciones emitidas de tal manera que pueda iniciarse una nueva instrucción de repetición.

La Fig. 9 ilustra el principio de funcionamiento de la cola local de acuerdo con una realización de la invención. La cola misma está representada por una línea horizontal 901. Una primera flecha vertical simboliza el puntero de escritura 903, que indica la posición de la cola en la que se está escribiendo actualmente una nueva instrucción. Una flecha horizontal correspondiente 905 indica la dirección en la que se mueve el puntero de escritura, hacia la derecha en el dibujo.

Una segunda flecha vertical simboliza el puntero de lectura 907, que indica la posición de la cola desde la cual se está leyendo actualmente una instrucción que se va a ejecutar. Una flecha horizontal correspondiente 909 indica la dirección en la que se mueve el puntero de lectura, en la misma dirección que el puntero de escritura 903. La distancia entre el puntero de escritura 903 y el puntero de lectura 907 es la longitud actual de la cola, es decir, el número de instrucciones actualmente en la cola.

En el ejemplo de la Fig. 9 una secuencia de instrucciones que deben repetirse varias veces se ha escrito en la cola. El inicio de la secuencia y el final de la secuencia se indican mediante una primera línea vertical 911 y una segunda línea vertical 913 a través de la línea horizontal 901. Una flecha hacia atrás 915 indica que cuando el puntero de lectura 907 alcanza el final de la secuencia de comandos indicada por la segunda línea vertical 913, el puntero de lectura volverá al inicio de la secuencia de comandos indicada por la primera línea vertical 911. Esto se repetirá hasta que la secuencia de instrucciones se haya ejecutado el número especificado de veces.

La lógica de control (no mostrada) está dispuesta para realizar un seguimiento del número de instrucciones en la secuencia que se va a iterar, y sus posiciones en la cola. Esto incluye, por ejemplo:

- La posición 911 del comienzo de la secuencia de instrucciones que se van a repetir
- La posición 913 del final de la secuencia de instrucciones que se van a repetir
- El número de veces que la secuencia de instrucciones se va a repetir

En lugar del inicio y el final de la secuencia, la posición de comienzo o final de la secuencia puede almacenarse junto con la longitud de la secuencia, es decir, el número de instrucciones incluidas en la secuencia.

REIVINDICACIONES

1. Una unidad de ejecución de vectores (203, 205, 520, 530) para su uso en un procesador digital de señales (200) que tiene un núcleo de procesador, dicho núcleo comprendiendo una memoria de programa dispuesta para contener instrucciones para una pluralidad de unidades de ejecución, el procesador digital de señales comprendiendo adicionalmente una pluralidad de unidades de memoria de datos dispuestas para contener datos para ser usados por la unidad de ejecución de vectores, el núcleo, la unidad de ejecución de vectores y las unidades de memoria de datos estando interconectadas por una red (244), dicha unidad de ejecución de vectores estando dispuesta para ejecutar instrucciones, incluyendo instrucciones vectoriales que deben realizarse en datos múltiples en forma de un vector, comprendiendo un registro de instrucciones (722) dispuesto para recibir y almacenar instrucciones, un decodificador de instrucciones (723) dispuesto para decodificar instrucciones almacenadas en el registro de instrucciones, y por lo menos una ruta de datos controlada por el decodificador de instrucciones, dicha unidad de ejecución de vectores comprendiendo además un controlador de vectores (720, 720') y un registro de recuento (721), dicho controlador de vectores estando dispuesto para determinar si una instrucción es una instrucción vectorial y , si lo es, informar al registro de recuento (721), que está dispuesto para contener la longitud del vector, dicho controlador de vectores (720, 720') estando además dispuesto para controlar la ejecución de instrucciones, dicha unidad de ejecución de vectores estando además **caracterizado porque**

- comprende una cola local (730) dispuesta para recibir por lo menos una primera y una segunda instrucciones de la memoria del programa, para proporcionar la primera instrucción al registro de instrucciones (722) y para contener la segunda instrucción en la cola local (730) hasta que se cumple una condición predefinida, y porque
- el controlador de vectores (720, 720') comprende un medio de control de cola (732, 721) dispuesto para controlar la cola local.

2. Una unidad de ejecución de vectores de acuerdo con la reivindicación 1, dispuesta además para recibir una señal de emisión y para controlar la ejecución de instrucciones basadas en esta señal de emisión.

3. Una unidad de ejecución de vectores de acuerdo con la reivindicación 1 ó 2, en la que el por lo menos una ruta de datos comprende además una pluralidad de rutas de datos controladas por el decodificador de instrucciones.

4. Una unidad de ejecución de vectores de acuerdo con cualquiera de las reivindicaciones anteriores, en la que la cola local (730) está dispuesta para pausar la lectura de instrucciones hasta que la ruta de datos esté lista para recibir y ejecutar otra instrucción.

5. Unidad de ejecución de vectores de acuerdo con cualquiera de las reivindicaciones anteriores, en la que el medio de control de cola (732) comprende un controlador de cola dispuesto para contener información de estado relacionada con la cola, como de llena está la cola local (730) y para controlar el envío de instrucciones desde la cola local (730) a la unidad de ejecución de vectores (203, 205, 520, 530) para su ejecución.

6. Unidad de ejecución de vectores de acuerdo con la reivindicación 5, en la que el controlador de cola está dispuesto para, si se envía una nueva instrucción a la cola y la cola está llena, generar un mensaje de error.

7. Una unidad de ejecución de vectores de acuerdo con la reivindicación 6, en la que el medio de control de cola (732) está dispuesto para emitir una señal específica que instruye a la cola local para pausar la lectura de instrucciones de la cola local hasta que se cumpla la condición.

8. Una unidad de ejecución de vectores de acuerdo con cualquiera de las reivindicaciones anteriores, en la que el controlador de vectores (720, 720') está dispuesto para hacer que se envíe una señal a un control de flujo de programa (703) del Procesador digital de señales para indicar que la unidad está listo para aceptar una nueva instrucción.

9. Una unidad de ejecución de vectores de acuerdo con cualquiera de las reivindicaciones anteriores, en la que el decodificador de instrucciones (723) está dispuesto para informar al controlador de vectores (720, 720') sobre la instrucción que se está ejecutando en cualquier momento dado.

10. Una unidad de ejecución de vectores de acuerdo con cualquiera de las reivindicaciones anteriores, en la que la cola local (730) es una cola de primero en entrar, primero en salir.

11. Un procesador digital de señales (200) que comprende:

- un núcleo de procesador (201) que incluye una unidad de ejecución de enteros (212) configurada para ejecutar instrucciones de enteros; y
- por lo menos una primera y una segunda unidades de ejecución de vectores (203, 205, 520, 530) separadas de y acopladas al núcleo de procesador (201), en donde cada unidad de ejecución de vectores

(203, 205) es una unidad de ejecución de vectores de acuerdo con cualquiera de las reivindicaciones anteriores;

- una red en chip (244) dispuesta para proporcionar conexiones entre el núcleo del procesador y la primera y segunda unidades de ejecución del vectores (203, 205, 520, 530)
- dicho procesador digital de señales comprende una memoria de programa (211) dispuesta para contener instrucciones para la primera y segunda unidades de ejecución de vectores (203, 205) y lógica de emisión para emitir instrucciones, incluyendo instrucciones vectoriales, a la primera y la segunda unidades de ejecución de vectores.

12. Un procesador digital de señales de acuerdo con la reivindicación 11, en el que la memoria de programa (211) también está dispuesta para contener instrucciones para la unidad de ejecución de enteros (212).

13. Un procesador digital de señales de acuerdo con cualquiera de las reivindicaciones 11-12, en el que la memoria de programa (211) está dispuesta en el núcleo del procesador (201).

14. Un dispositivo de comunicación de banda base adecuado para la comunicación con cable e inalámbrica multimodo, que comprende:

- una unidad frontal (7) configurada para transmitir y/o recibir señales de comunicación;
- un procesador digital de señales programable (3) acoplado con la unidad front-end analógica, en el que el procesador digital de señales programable es un procesador digital de señales de acuerdo con cualquiera de las reivindicaciones 11-13.

15.. Un dispositivo de comunicación de banda base de acuerdo con la reivindicación 14, en el que la unidad front-end (7) es una unidad front-end analógica dispuesta para transmitir y/o recibir señales de radiofrecuencia o banda de base.

16.. Un dispositivo de comunicación de banda base de acuerdo con la reivindicación 14 ó 15, dicho dispositivo de comunicación de banda base para comunicación en una red de comunicaciones inalámbricas, como una red de comunicaciones móviles.

17. Un dispositivo de comunicación de banda base de acuerdo con la reivindicación 14, dicho dispositivo de comunicación de banda base siendo un receptor de televisión.

18. Un dispositivo de comunicación de banda base de acuerdo con la reivindicación 14, dicho dispositivo de comunicación de banda base siendo un cable módem.

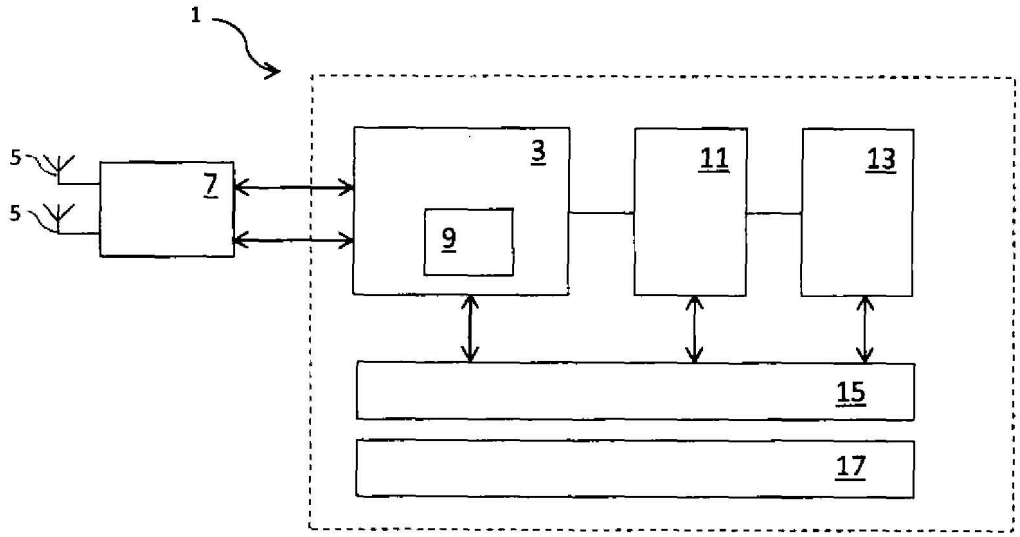
1
5
5
7
3
9
11
13
15
17

Fig. 1

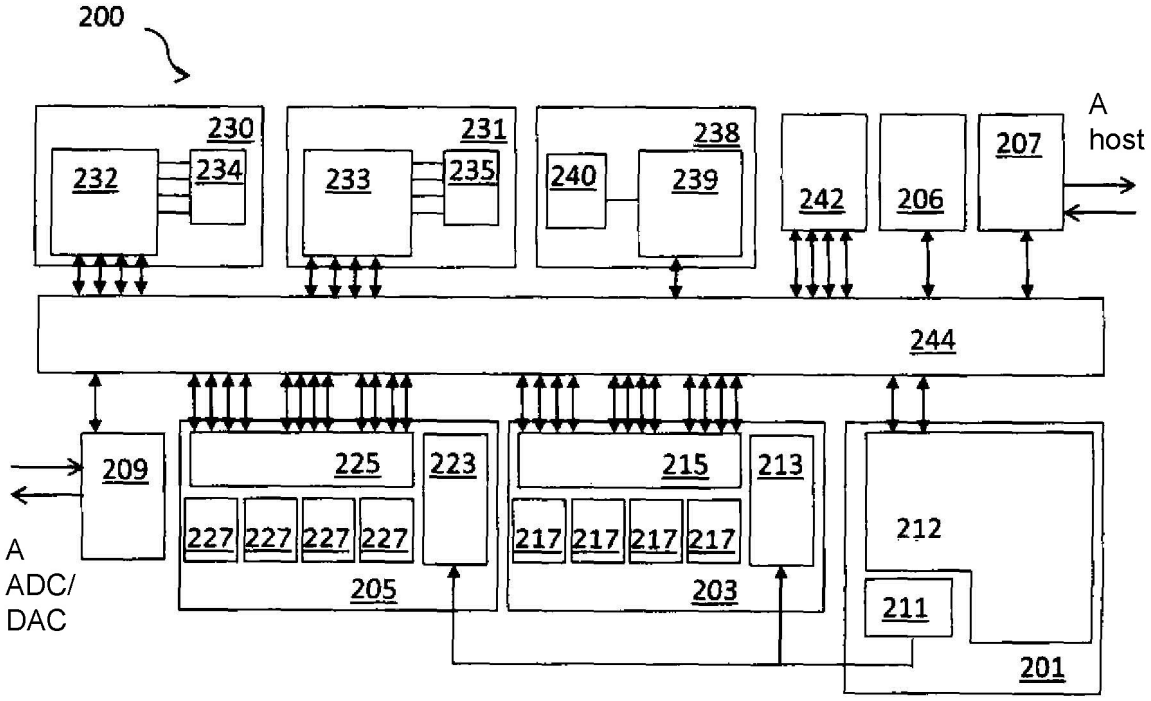
200
230
232
234
231
233
235
238
240
239
242
206
207
A
host
244
209
225
223
215
213
227
227
227
227
217
217
217
217
212
211
205
203
201
A
ADC/
DAC

Fig. 2

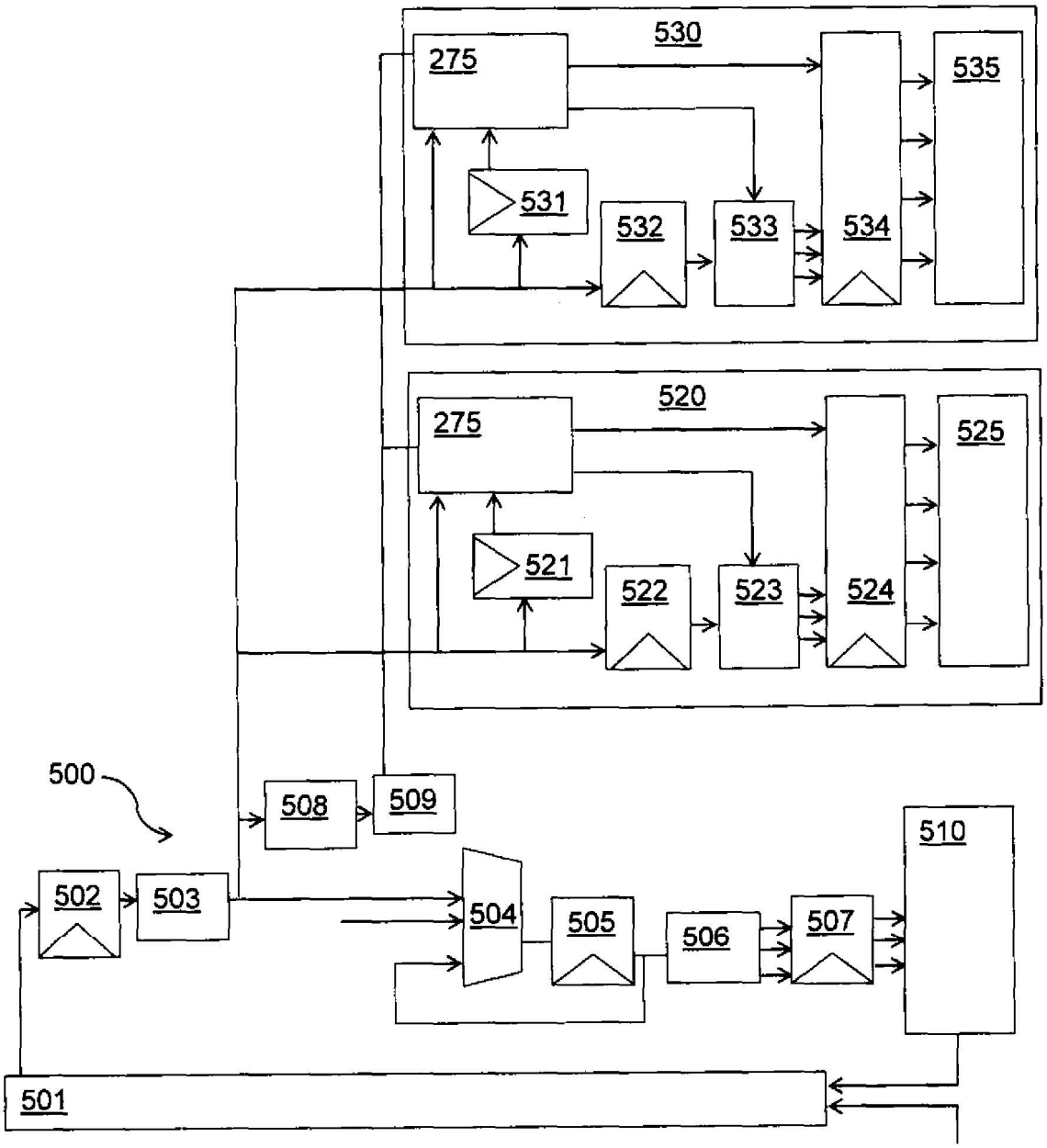
530
275
531
532
533
534
535
520
275
521
522
523
524
525
500
508
509
510
502
503
504
505
506
507
501

Fig. 3

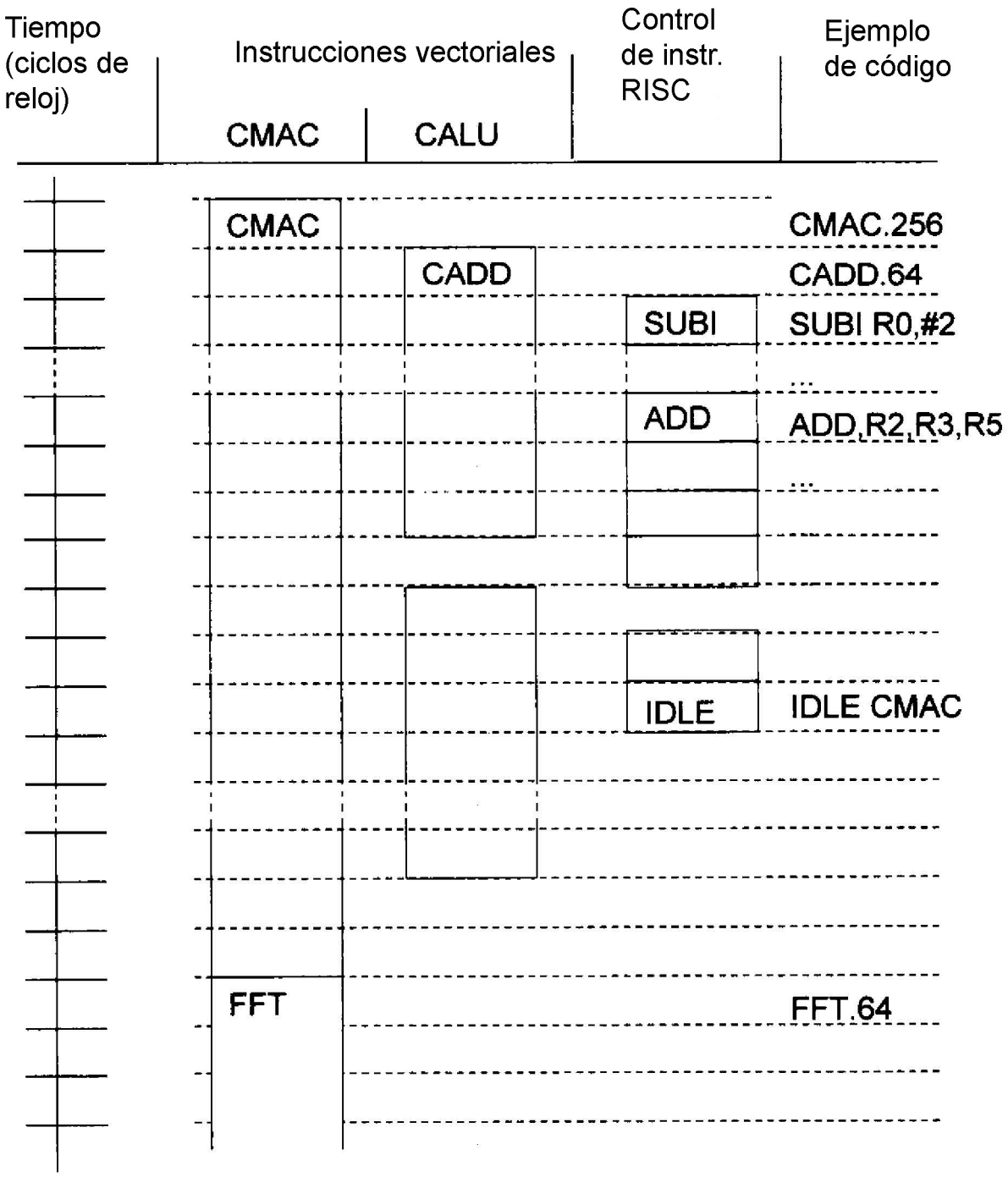

Fig. 4 ESTADO DE LA TECNICA

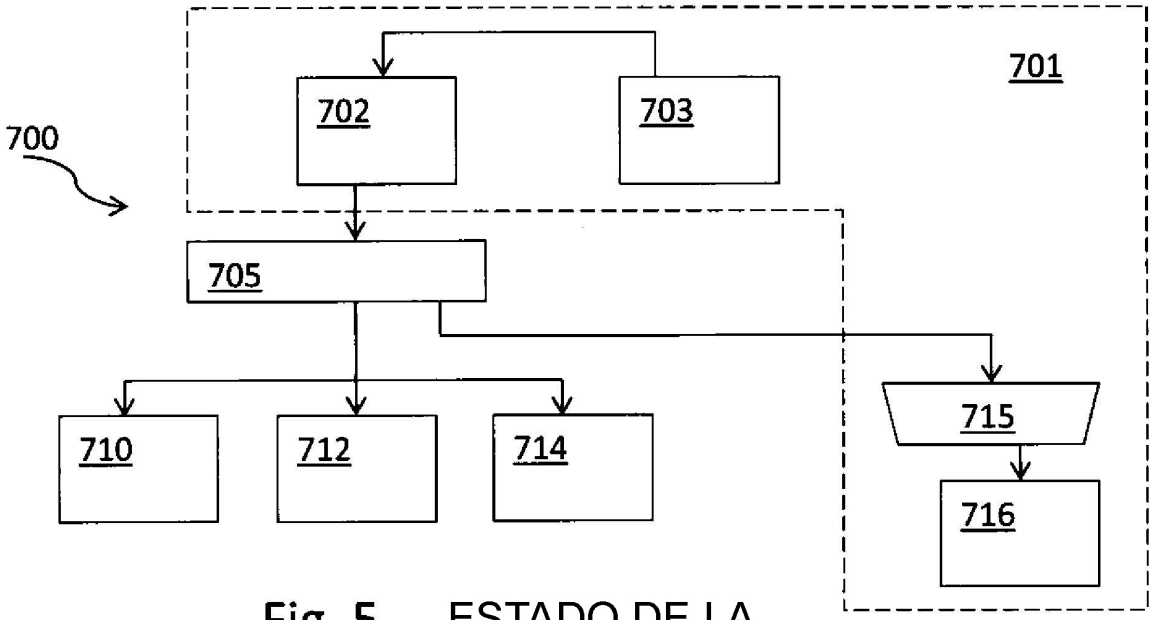

Fig. 5 ESTADO DE LA TECNICA

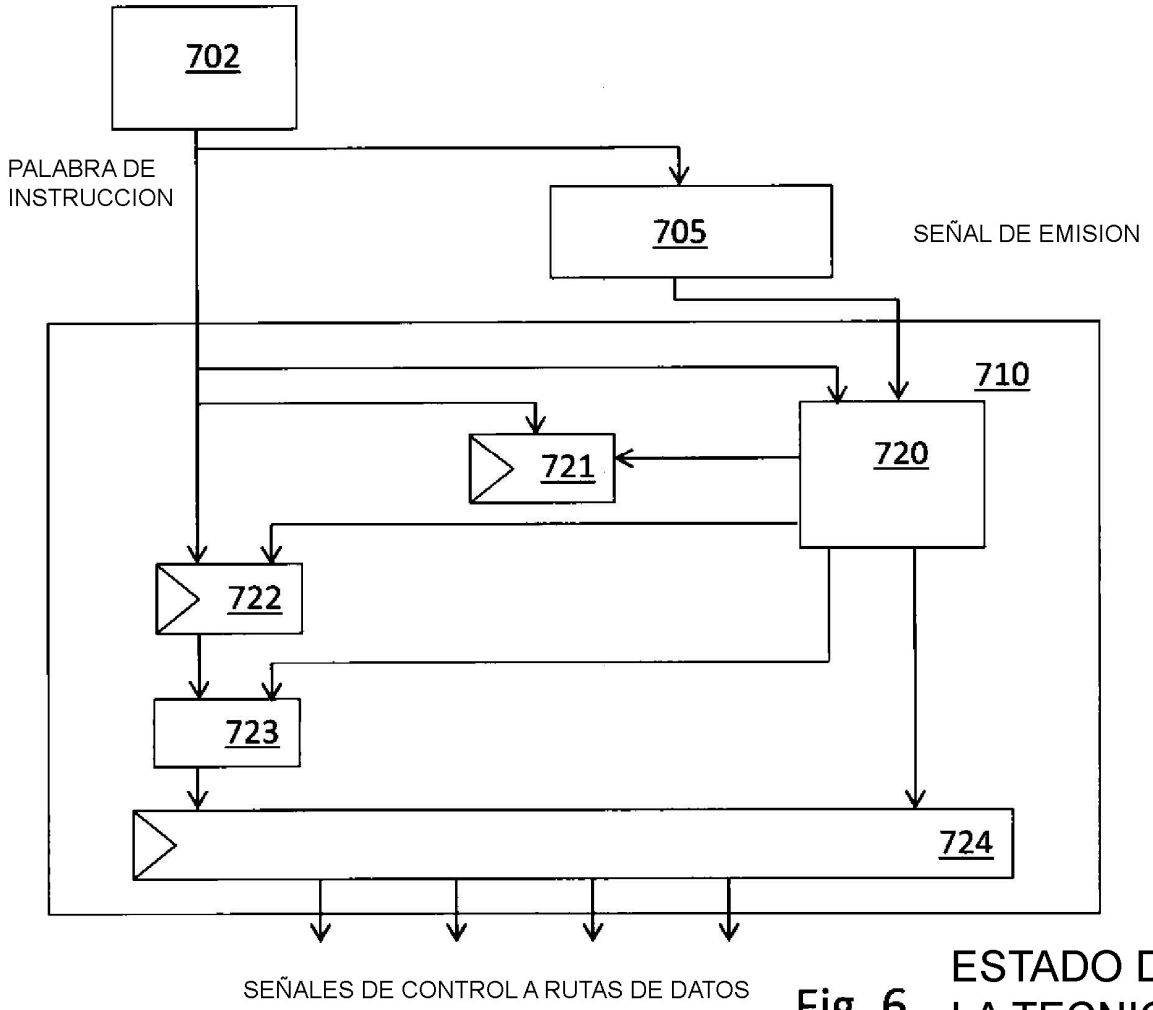

Fig. 6 ESTADO DE LA TECNICA

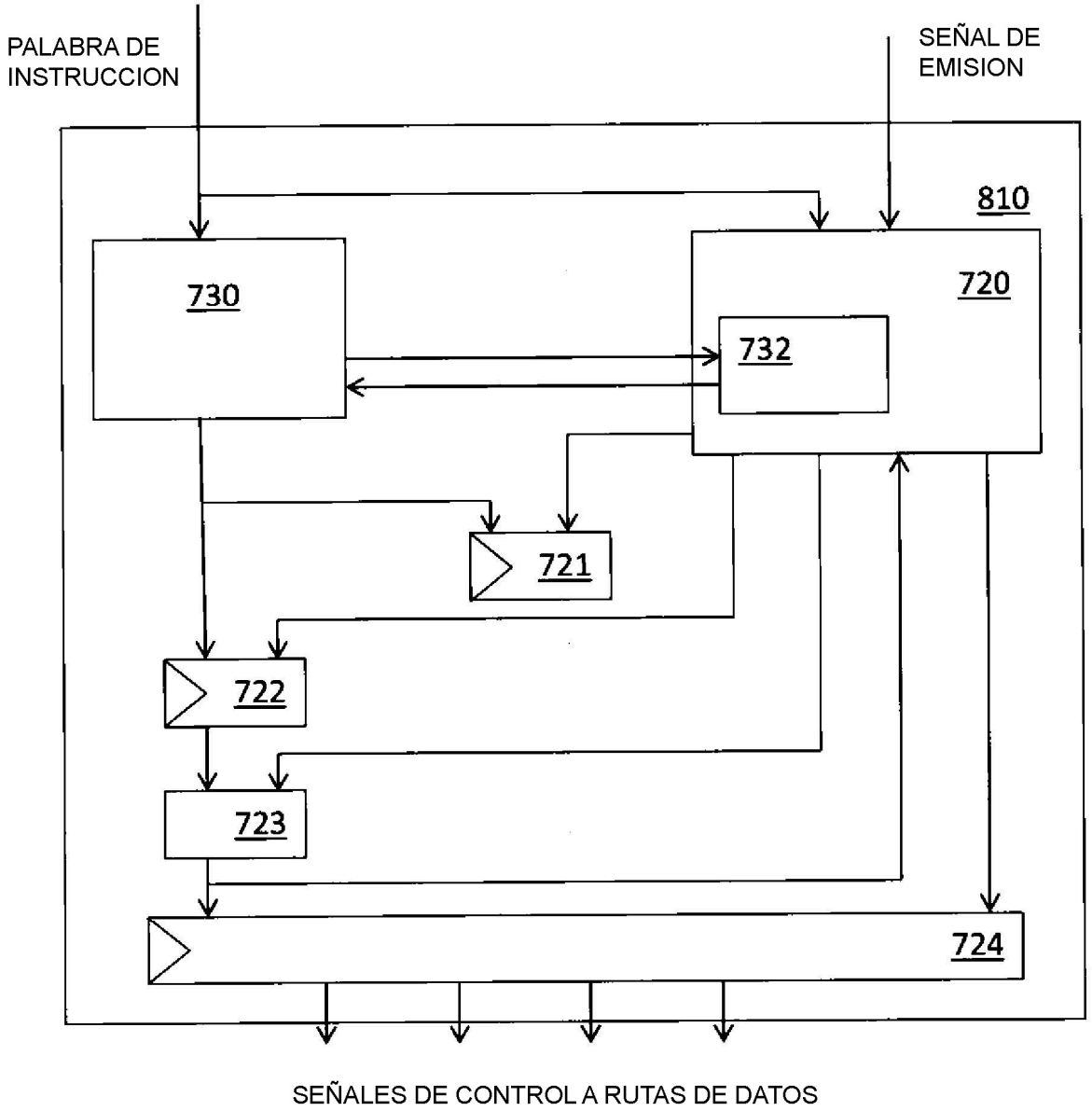
PALABRA DE INSTRUCCION
SEÑAL DE EMISION
810
730
720
732
721
722
723
724
SEÑALES DE CONTROL A RUTAS DE DATOS

Fig. 7

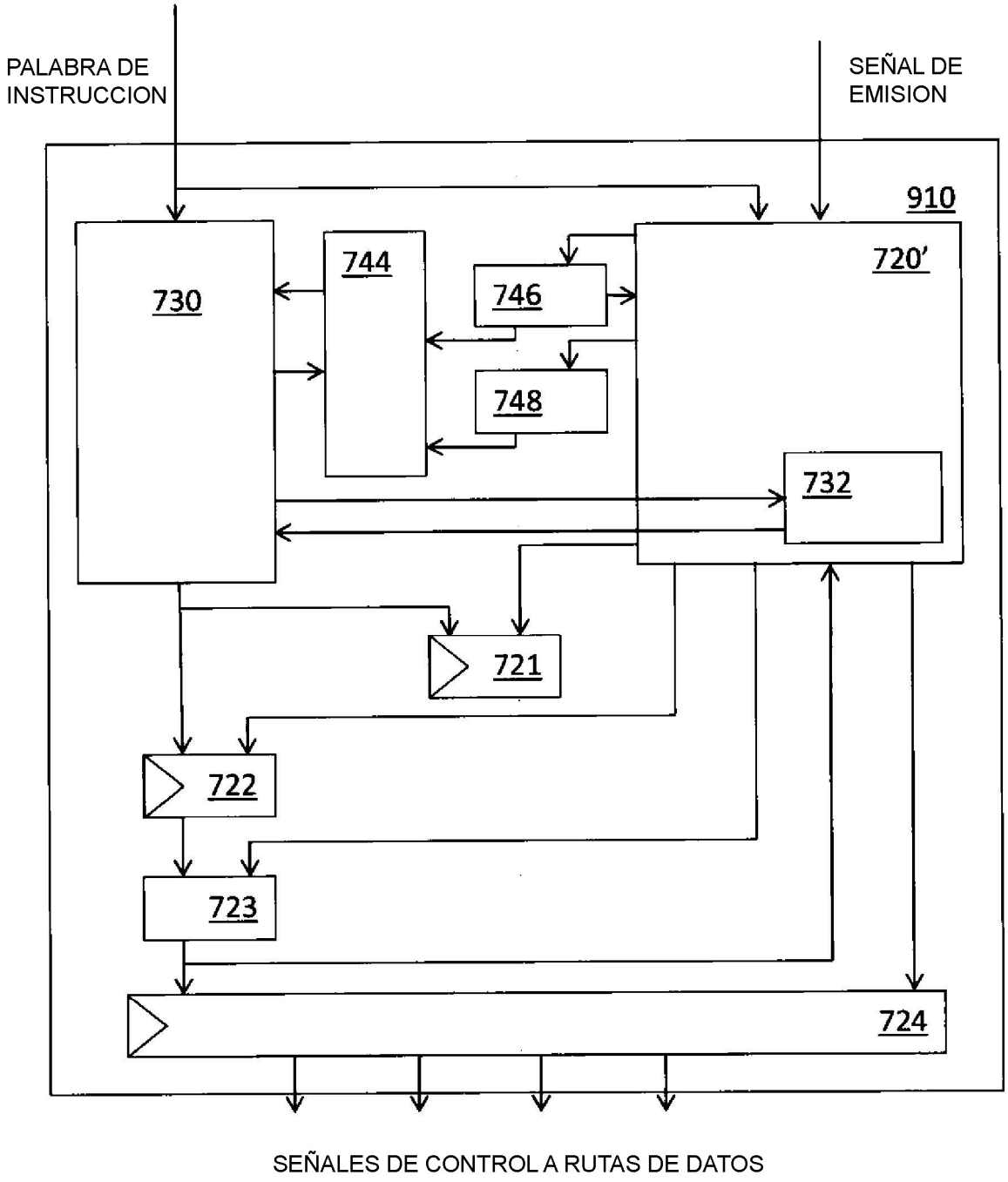
PALABRA DE INSTRUCCION
SEÑAL DE EMISION
910
730
744
746
720'
748
732
721
722
723
724
SEÑALES DE CONTROL A RUTAS DE DATOS

Fig. 8

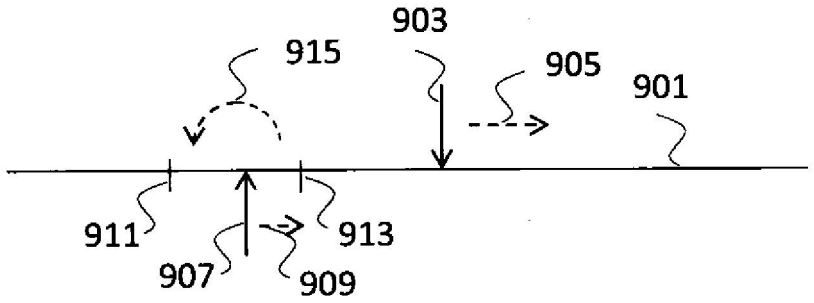

Fig. 9