(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 647 099**

(51) Int. CI.:

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

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **28.11.2012 PCT/SE2012/051321**

(87) Fecha y número de publicación internacional: **27.06.2013 WO13095258**

(96) Fecha de presentación y número de la solicitud europea: **28.11.2012 E 12816376 (3)**

(97) Fecha y número de publicación de la concesión europea: **16.08.2017 EP 2751671**

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

(30) Prioridad:

20.12.2011 SE 1151231

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
19.12.2017

(73) Titular/es:

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

(72) Inventor/es:

NILSSON, ANDERS y
TELL, ERIC

(74) Agente/Representante:

IZQUIERDO BLANCO, María Alicia

Procesador de señal digital y dispositivo de comunicación de banda base

DESCRIPCIÓN

Campo técnico

La presente invención se refiere a un procesador de señal digital (DSP), por ejemplo, un DSP basado en SIMT.

Antecedentes y técnica relacionada

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

Para un mayor rendimiento y fiabilidad, muchos terminales móviles utilizan actualmente un tipo de DSP conocido como procesador de banda base (BBP), para manipular muchas de las funciones de procesamiento de señales asociadas con el procesamiento de la señal de radio recibida y la preparación de señales para la transmisión. Es ventajoso separar tales funciones del procesador principal, ya que dependen en gran medida 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 posible para adaptarse a las normas en desarrollo y permitir la reutilización de hardware. Por lo tanto, se han desarrollado procesadores de banda base programables, PBBP.

Muchas de las funciones que se realizan con frecuencia 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 porque permite que una sola instrucción funcione en múltiples elementos de datos, en lugar de en un elemento de datos a la vez. Se pueden disponer múltiples elementos de datos en un vector, y en este documento se hará referencia a una unidad de procesamiento adecuada para operar sobre un vector de datos como una unidad de ejecución de vectores.

Como un desarrollo adicional de la arquitectura SIMD, se ha desarrollado la arquitectura de secuencia de múltiples tareas de una sola instrucción (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 central.

La solicitud de patente internacional WO 2007/018467 divulga un DSP según la arquitectura SIMT, que tiene un núcleo de procesador que incluye una unidad de ejecución de enteros y una memoria de programa, y dos unidades de ejecución de vector que están conectadas, pero no integradas en el núcleo. Las unidades de ejecución de vectores pueden ser Unidades Aritméticas Lógicas Complejas (CALU) o Unidades Acumuladas múltiples complejas (CMAC). El núcleo tiene una memoria de programa para distribuir instrucciones a las unidades de ejecución. En el documento WO2007/018467 cada una de las unidades de ejecución de vectores tiene un descodificador 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.

Sumario de la invención

Es un objetivo de la presente invención fabricar un procesador SIMT más flexible y que permita un uso más eficiente de la memoria de programa, emitir ancho de banda y unidades de ejecución.

Este objetivo se logra según la presente invención mediante un procesador digital que comprende:

- un núcleo del procesador, que incluye una unidad de ejecución de enteros configurada para ejecutar instrucciones enteras; y
- al menos una primera y una segunda unidades de ejecución de vector separadas y acopladas al núcleo del procesador, teniendo dichas unidades de ejecución de vector un primer y un segundo número de trayectorias de datos, respectivamente, estando dichas unidades de ejecución de vector dispuestas para ejecutar instrucciones, incluyendo instrucciones vectoriales que se deben realizar con datos múltiples en forma de un vector;
- dicho procesador de señal digital comprende una memoria de programa dispuesta para contener instrucciones para la primera y segunda unidades de ejecución de vector y una lógica de emisión para emitir instrucciones, que incluyen instrucciones de vector, a la primera y segunda unidades de ejecución de vector.

El procesador de señal digital se caracteriza porque el procesador comprende una unidad de control de emisión para seleccionar al menos dos unidades de ejecución que deben recibir y ejecutar la misma instrucción al mismo tiempo, y lógica para enviar la instrucción a dichas al menos dos unidades de ejecución.

En el procesador definido anteriormente, la misma instrucción se puede usar para controlar varias unidades de ejecución. Esto reduce significativamente la sobrecarga de control al enviar la misma instrucción a varias unidades de ejecución. También permite la ejecución paralela de la misma instrucción en varias unidades de ejecución. La posibilidad de iniciar varias unidades de ejecución a la vez hace que el manejo de las instrucciones sea muy eficiente. Una unidad de ejecución puede ser una unidad de ejecución vectorial, una unidad de ejecución escalar o una unidad de ejecución de enteros. Una unidad de ejecución escalar está dispuesta para procesar un elemento de datos a la vez, pero el elemento de datos puede ser un número entero o un valor complejo. Por ejemplo, la misma instrucción vectorial puede enviarse a dos o más unidades de ejecución de vectores para que se realicen en diferentes conjuntos de datos. Ejemplos de instrucciones no vectoriales que a menudo se envían a más de una unidad de ejecución de vectores son *clear* y *star.* Es posible, por ejemplo, tener un grupo receptor que incluya todas las unidades de ejecución de vectores.

En una realización preferida, cada unidad de ejecución de vector que comprende un controlador vectorial dispuesto para determinar si una instrucción es una instrucción vectorial y, si lo es, informar un registro de conteo dispuesto para mantener la longitud del vector, estando dichos controladores vectoriales dispuestos para controlar la ejecución de instrucciones

El procesador también puede comprender uno o más aceleradores, conocidos en la técnica. El término unidad funcional, cuando se usa en este documento, indica una unidad de ejecución o un acelerador.

Preferentemente, se define un número de grupos de problemas, comprendiendo cada grupo receptor al menos una de las unidades de ejecución, y al menos un grupo receptor que comprende más de uno de la unidad de ejecución, y la unidad de control de expedición está dispuesta para seleccionar al menos dos unidades de ejecución seleccionando un grupo receptor. Esto puede estar codificado en el núcleo.

Alternativamente, en una realización preferida, la unidad de control de expedición comprende además al menos una máscara asociada con al menos un grupo de emisión, indicando dicha máscara qué unidad o unidades de ejecución en el grupo receptor deberían recibir y ejecutar la instrucción. Esto hace posible cambiar la definición de grupos de problemas y la selección de unidades de ejecución para cada grupo receptor, lo que hace que el procesador sea más flexible.

Un grupo receptor puede comprender al menos una unidad de ejecución de enteros y / o al menos una unidad de ejecución vectorial. Se puede definir un grupo receptor para que comprenda solo unidades de ejecución del mismo tipo, o una combinación de unidades de ejecución de diferentes tipos, según se desee. Puede ser adecuado definir un grupo receptor que incluya todas las unidades de ejecución, por ejemplo, para emitir el comando *clear*.

Una instrucción puede implicar la lectura de datos y la escritura de datos en otras unidades del procesador. Cuando se envía la misma instrucción a varias unidades de ejecución en un grupo receptor, normalmente cada unidad de ejecución debe trabajar con su propio conjunto de otras unidades para evitar que varias unidades de ejecución intenten leer o escribir en la misma unidad. Por lo tanto, en una realización preferida, al menos una unidad de ejecución comprende una tabla de asignación para traducir información contenida en una instrucción que indica al menos otra unidad con la que la ejecución debería interactuar, por ejemplo, desde qué memoria debería leer datos. Aún así, se pueden disponer dos o más unidades de ejecución para recibir datos de la misma unidad de memoria o unidad funcional en el procesador, por ejemplo cuando una unidad de ejecución en el grupo receptor debe realizar la función A = suma (X * Y), y otra es realizar la función B = suma (X * Z), siendo X, Y y Z vectores de datos obtenidos de las otras unidades en el procesador.

Una forma de manejar el resultado de un grupo receptor implica escribir el resultado de cada unidad de ejecución en el grupo receptor en la misma unidad de registro vectorial y dejar que la unidad de registro vectorial realice las instrucciones involucradas en el procesamiento del resultado.

Preferentemente, el decodificador de instrucciones está dispuesto para informar a la unidad de registro vectorial sobre la instrucción que se está ejecutando en cualquier momento dado.

La selección de qué grupo receptor debe realizar una instrucción en particular se puede manejar de diferentes maneras. Normalmente, una señal de problema se extraerá en el núcleo y se enviará a la unidad de ejecución correspondiente. En este caso, la al menos una unidad de ejecución en un grupo receptor está dispuesta además para recibir una señal de problema y para controlar la ejecución de instrucciones basadas en esta señal de problema. Alternativamente, cada unidad de ejecución de vector puede disponerse para extraer una señal de emisión de una palabra de instrucción recibida y determinar si debería participar en la ejecución de la palabra de instrucción en función de la señal de emisión.

Preferentemente, el controlador vectorial controla la ejecución de las instrucciones sobre la base de una señal de emisión recibida desde el núcleo. Alternativamente, la señal de emisión puede ser manejada localmente por la propia unidad de ejecución. Cómo implementar esto es conocido en la técnica.

El procesamiento de acuerdo con la invención se hace más eficiente al permitir el procesamiento simultáneo de una instrucción en dos conjuntos de datos diferentes por dos unidades de ejecución. También sería posible permitir que dos unidades de ejecución procesaran diferentes partes del mismo conjunto de datos, siempre que las diferentes partes se almacenaran en diferentes memorias. Esto permite un procesamiento más eficiente de grandes conjuntos de datos que lo que está habilitado en la técnica anterior, sin tener que implementar unidades de ejecución de vector más grandes. Como solución alternativa, la capacidad de una unidad de ejecución de vectores podría incrementarse aumentando el número de rutas de datos incluidas en la unidad de ejecución de vectores, pero una unidad de ejecución de vectores de alta capacidad sería innecesariamente grande para la mayoría de los comandos y, por lo tanto, ineficaz. Por lo tanto, la invención proporciona una solución más flexible y rentable que la de proporcionar una unidad de ejecución de vectores única con mayor capacidad

La distribución de instrucciones y datos hacia y desde varias unidades de una vez permite un manejo extremadamente eficiente de las instrucciones, ya que puede lograrse el envío de la misma señal entre varias unidades a prácticamente el mismo costo que la señalización entre dos unidades.

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 frontal configurada para transmitir y / o recibir señales de comunicación;
- Un procesador de señal digital programable acoplado a la unidad frontal, en el que el procesador de señal digital programable es un procesador de señal digital 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.

El procesador según las realizaciones de esta invención es particularmente útil para procesadores de señal digital, especialmente procesadores de banda base. La unidad frontal puede ser una unidad frontal analógica dispuesta para transmitir y / o recibir señales de radiofrecuencia o banda de base.

Dichos procesadores son ampliamente utilizados en diferentes tipos de dispositivos de comunicación, como teléfonos móviles, receptores de televisión y módems de cable. En consecuencia, el dispositivo de comunicación de banda base puede estar dispuesto para la comunicación en una red de comunicaciones celulares, 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, tales 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 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 con cable o inalámbricas. Las señales de comunicación son convertidas por una unidad frontal del dispositivo a una señal de banda base, de una manera adecuada.

Breve descripción de los dibujos

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

La figura 1 es un diagrama de bloques del procesador de banda base de acuerdo con una realización de la invención.

La figura 2 ilustra un formato de instrucción que puede usarse para seleccionar un grupo receptor particular.

La figura 3 ilustra la lógica de problemas de instrucción en un procesador SIMT

La figura 4A ilustra las funciones lógicas de problemas

La figura 4B ilustra una máscara que puede usarse para especificar grupos receptores

La figura 5 es un diagrama que ilustra los tubos de emisión de instrucciones de una realización del núcleo del procesador de la figura 2.

La figura 6 ilustra una forma de manejar la señal inactiva en un grupo receptor

Descripción detallada de las realizaciones

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

Una unidad de interfaz de ordenador principal 207 proporciona conexión al procesador de ordenador principal (no mostrado). Si está presente un procesador de MAC, está conectado entre la unidad de interfaz de ordenador principal 207 y el procesador de ordenador principal. Una unidad de terminal frontal digital 209 proporciona conexión a una unidad ADC / DAC 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 problemas de instrucción y funciones para soporte de múltiples contextos. Para cada contexto de ejecución, o subproceso, admitido esto incluye un contador de programa, un puntero de pila y un archivo de registro (no se muestra explícitamente en la figura 2). Por lo general, se admiten 2-3 hilos. Esto permite el uso de una función llamada *fork,* que permite al núcleo realizar ciertas instrucciones mientras, por ejemplo, una unidad de ejecución de vectores está ejecutando una instrucción vectorial. Por lo tanto, no se desea tener grupos de problemas superpuestos entre los diferentes subprocesos. Por lo tanto, cada subproceso preferentemente tiene su propio conjunto de unidades de ejecución de vectores, para evitar una situación en la que dos subprocesos intenten usar la misma unidad de ejecución de vectores al mismo tiempo. Típicamente, es posible en el sistema utilizar la misma unidad de ejecución de vectores en más de un hilo, pero si un hilo intenta enviar una señal de problema a una unidad de ejecución de vectores que ya está siendo utilizada por otro hilo, se emitirá un mensaje de error.

El núcleo de controlador 201 también comprende una unidad de ejecución de enteros 212 que comprende un archivo de registro RF, un ICM de memoria de núcleo entero, una unidad de multiplicador MUL y una Unidad de Aritmética y Lógica / Cambio (ALSU). Estas unidades son conocidas en la técnica y no se muestran en la Figura 1.

Una red en chip 244 interconecta todas las unidades del procesador, incluyendo el núcleo controlador 201, la unidad de interfaz frontal 209, la unidad de interfaz de ordenador principal 207, las unidades de ejecución de vector 203, 205, los bancos de memoria 230, 232, la memoria de enteros banco 238 y los aceleradores 242.

En este ejemplo, cada una de la primera unidad de ejecución de vector 203 y la segunda unidad de ejecución de vector 205 son unidades de ejecución de vector CMAC, comprendiendo cada una un controlador de vector 213, una unidad de carga / almacenamiento de vector 215 y un número de rutas de datos 217. La función de carga se utiliza para obtener datos de las otras unidades conectadas a la red 244 incorporada en chip (por ejemplo desde un banco de memoria) y la función de almacenamiento se usa para almacenar datos de las unidades de ejecución 203, 205 a, por ejemplo, una unidad de memoria 230, 231 a través de la red en chip 244. Los datos también pueden obtenerse de otras unidades de ejecución de vectores y / o los resultados de cálculo pueden enviarse a otras unidades de ejecución de vectores para su posterior procesamiento. Cada unidad de ejecución de vector también comprende un controlador de vector 213, 223 dispuesto para recibir instrucciones desde la memoria de programa 211.

El controlador de vector de esta primera unidad de ejecución de vector está conectado a la memoria de programa 211 del núcleo de controlador 201 a través de la lógica de emisión, para recibir señales de problema relacionadas con las instrucciones de la memoria de programa. En la descripción anterior, la lógica de problema decodifica la palabra de instrucción para obtener la señal de problema y envía esta señal de problema a la unidad de ejecución de vector como una señal separada. También sería posible permitir que el controlador vectorial de la unidad de ejecución de vectores genere la señal de problema localmente. En este caso, las señales de problema son creadas por el controlador de vector en base a la palabra de instrucción de la misma manera que lo sería en la lógica de problema.

Alternativamente, las unidades de ejecución de vector 203, 205 son una unidad de ejecución de vector CALU de un tipo conocido en la técnica, que comprende un controlador de vector 223, una unidad de almacenamiento / carga de vector 225 y un número de rutas de datos 227. El controlador de vector 223 de esta segunda unidad de ejecución de vector también está conectado a la memoria de programa 211 del núcleo de controlador 201, a través de la lógica de problema, para recibir señales de problema relacionadas con las instrucciones de la memoria de programa.

Las unidades de ejecución de vector 203, 205 también podrían ser cualquier tipo de unidades de ejecución de vector. Aunque se muestran y discuten dos unidades de ejecución de vectores, el método de la invención puede extenderse para enviar la misma instrucción a tres o más unidades de ejecución de vectores.

Podría haber un número arbitrario de unidades de ejecución de vectores, además de las dos que se muestran en la Figura 1. Puede haber solo unidades CMAC, solo unidades CALU o un número adecuado de cada tipo. También puede haber otros tipos de unidades de ejecución de vectores que CMAC y CALU. Como se explicó anteriormente, una unidad de ejecución de vectores es un procesador que puede procesar instrucciones vectoriales, lo que significa que una sola instrucción realiza la misma función para varias 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 utilizan como ejemplos, pero debe tenerse en cuenta que las unidades de ejecución de vectores se pueden usar para realizar cualquier función adecuada en vectores de datos.

Para habilitar varias operaciones vectoriales concurrentes, el procesador tiene preferentemente un sistema de memoria distribuida donde la memoria está dividida en varios bancos de memoria, representados en la Figura 1 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, la unidad de generación de direcciones AGU 234, 235 respectivamente. El PBBP de la figura 1 también incluye uno o más bancos de memoria 238 enteros, que incluyen una memoria 239 y una unidad de generación de dirección 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 tales como codificación de canales e intercalado. Dichos aceleradores son bien conocidos en la técnica y no se discutirán aquí en detalle. Los aceleradores pueden ser configurables para ser reutilizados por muchos estándares diferentes.

La primera y la segunda unidades de ejecución de vector 203, 205 se muestran como unidades CMAC de cuatro vías con cuatro rutas de datos complejas que pueden ejecutarse simultáneamente o por separado. Las cuatro rutas de datos complejas incluyen multiplicadores, sumadores y registros de acumuladores (no todos mostrados en la Figura 1). Por lo tanto, en esta realización, CMAC 203 se puede denominar ruta de datos CMAC de cuatro vías. Además de multiplicar y agregar, el CMAC 203 también puede realizar operaciones de redondeo y escalado y soportar la saturación como se conoce en la técnica.

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 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 porción real y una parte imaginaria. Las instrucciones de DSP pueden ejecutarse en una o más de las unidades de ejecución de vector 203, 205. La tercera clase de instrucciones son las instrucciones Vector. Las instrucciones vectoriales pueden considerarse extensiones de las instrucciones DSP ya que operan en grandes conjuntos de datos y pueden utilizar modos de direccionamiento avanzados y soporte vectorial. Las instrucciones vectoriales pueden operar en tipos de datos complejos o reales.

En la técnica anterior, las unidades CMAC 203, 205 están dispuestas para funcionar por separado, procesando cada una instrucción, en un conjunto de datos, a la vez. De acuerdo con la invención, se incluyen medios de control que permitirán que las unidades CMAC 203, 205 trabajen simultáneamente en el mismo conjunto de datos con el fin de acelerar el procesamiento.

Para ilustración, en la técnica anterior, cada unidad de ejecución de vector tiene un nombre. El comando

.cmac 0

<instr>

significa que todas las siguientes instrucciones de CMAC deben enviarse al número de unidad 0 de CMAC. Esta información se encuentra en las instrucciones mismas y se decodifica en la lógica de problemas en el núcleo 201, o por las propias unidades de ejecución de vectores.

De acuerdo con la invención, se especifican grupos de unidades de ejecución, llamados grupos de problemas, cada grupo receptor comprende una o más unidades de ejecución del mismo tipo o de diferentes tipos. Cuando se emite una instrucción, el campo de unidad en la palabra de instrucción no codificará directamente una de las unidades de ejecución, sino que indicará uno de los grupos de problemas, como se analizará en relación con las Figuras 4A y 4B. La información acerca de qué unidades de ejecución están incluidas en cada grupo receptor se puede mantener en cualquier unidad adecuada, por ejemplo en una memoria dedicada en el núcleo de procesador 201, tal como la unidad lógica de emisión 705 de la figura 3. Esto se discutirá con más detalle en conexión con las Figuras 4A y 4B. Un grupo receptor se puede indicar en una instrucción de la misma manera que una unidad de ejecución de vectores única en la técnica anterior.

De acuerdo con la invención, se define un nuevo comando para decir que todas las instrucciones de un tipo particular deben enviarse a un grupo receptor particular, y no a una unidad de ejecución de vector individual. Si se han emitido los siguientes comandos:

.issuegroup<cmac> 0

.issuegroup<calu>5

esto significa que todas las instrucciones de cmac se deben enviar al número de grupo de expedición 0 y todas las instrucciones de calu se deben enviar al número de grupo de expedición 5. Si una instrucción cmac como *cacc x, y* se emite se enviará a emitir el número de grupo 0. Si una instrucción de calu como *vadd z,* b se emite, se enviará para emitir el número de grupo 5. Las unidades de ejecución de vectores en un grupo receptor pueden tener el mismo número de rutas de datos, o diferentes números de rutas de datos.

La Figura 2 muestra un ejemplo de un formato de instrucción. En este ejemplo, un grupo receptor llamado grupo receptor 0 se indica mediante la codificación del grupo receptor 0 0 1. En el ejemplo que se muestra en la Figura 2, la unidad de ejecución de enteros tiene su propia entrada y no está incluida en ningún grupo receptor. También sería posible definir un grupo receptor, por ejemplo, número de grupo de expedición 0 para incluir la unidad de ejecución de enteros. En este ejemplo alternativo, un grupo receptor se usaría para procesar instrucciones enteras. En el ejemplo de la Figura 2, utilizando tres bits para el número de grupo de emisión, se pueden especificar ocho grupos de problemas diferentes. Si se desea una mayor cantidad de grupos de problemas, la cantidad de bits utilizados para indicar los grupos de problemas debe aumentarse en consecuencia. La letra x en la figura indica un elemento de datos.

Como se explicó en relación con la Figura 1 anterior, el núcleo normalmente admite dos o más subprocesos o contextos. Como en el caso en que se usan unidades de ejecución de vectores individuales, no es deseable involucrar a la misma unidad funcional en dos o más hilos porque existe el riesgo de conflicto. Preferentemente, por lo tanto, se agrega un bit adicional al campo de problema en la figura 2, para indicar con qué hilo, o contexto, se puede usar el grupo de emisión.

La Figura 3 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 701 que tiene una memoria de programa PM 702 que contiene instrucciones para las diversas unidades de ejecución del procesador, y una unidad de control de flujo de programa 703. La unidad de control de flujo de programa 703 está dispuesta para señalar la siguiente dirección desde la que debe leerse una instrucción en la memoria de programa 702. A partir de la memoria de programa 702, las instrucciones se obtienen a una unidad lógica de expedición 705, que es común a todas las unidades de ejecución y está dispuesta para controlar a dónde enviar cada instrucción específica. La unidad lógica de emisión 705 está conectada en este caso a un número de unidades de ejecución de vector 710, 712, 714 y a través de un multiplexor 715 a una unidad de ejecución de enteros 716. Como se explicó 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 vector.

La figura 4A ilustra un ejemplo de una unidad de control de problemas, correspondiente a la unidad 705 de la figura 3, de acuerdo con la invención. Como antes, el núcleo comprende una memoria de programa 211 que contiene instrucciones para unidades de ejecución de vector. Una unidad 321 de precodificación está dispuesta para determinar qué unidad de ejecución debería recibir cada instrucción que se lee desde la memoria del programa. La palabra de instrucción se envía directamente desde la memoria de programa 211 a todas las unidades de ejecución. Esto no se muestra en la Fig. 4A, que solo muestra las señales de control. La señal de emisión, que transporta la información sobre qué unidad o unidades funcionales deben realizar la instrucción, se envía a través de un demultiplexor 324. La señal de problema puede enviarse a la unidad de ejecución de enteros en el núcleo, como se muestra mediante la flecha marcada CORE del demultiplexor. Alternativamente, la señal de problema puede estar destinada a un grupo receptor. En este caso, la señal de problema puede enviarse tal como está a todas las unidades funcionales en este grupo receptor.

En una realización preferida, sin embargo, para proporcionar más flexibilidad, se puede usar una máscara en conexión con la señal de emisión, como se muestra en la figura 4A. En este caso, se disponen varias unidades de máscara 326, 328, 330, una para cada grupo receptor. Una unidad de operador lógico 332, 334 recibe la señal de emisión prevista para un grupo receptor desde el demultiplexor 324. Esta unidad de operador lógico 332, 334 también recibe información de la unidad de máscara 326, 328, 330 correspondiente a este grupo receptor y determina qué unidades funcionales en el grupo receptor deberían recibir la instrucción. La función de la unidad de máscara se analizará con más detalle a continuación. Cuando la unidad del operador lógico ha determinado, en función de la señal de emisión y la información de la máscara, qué unidad o unidades funcionales deben realizar la instrucción, la señal de emisión se envía a estas unidades de ejecución del vector. De esta forma, las unidades

funcionales incluidas en un grupo receptor pueden variar dinámicamente en lugar de estar codificadas en el sistema durante la configuración.

La figura 4B muestra un ejemplo de unidad de máscara 325 de acuerdo con la realización anterior. La unidad de máscara comprende una máscara que identifica las unidades de ejecución de vector en un grupo de unidades de ejecución de vector que realmente deberían recibir la instrucción. En la práctica, la máscara tiene un bit para cada unidad de ejecución del vector, que se puede establecer en 0 o 1, para indicar si la unidad de ejecución del vector debe incluirse en el grupo receptor o no. Esta información se combina con la información contenida en la señal de emisión para determinar qué unidades de ejecución de vector deben recibir la instrucción.

En este ejemplo, las unidades de máscara 326, 328, 330 se usan todas para el mismo grupo receptor. Como se indica por otra unidad de máscara 340, también puede haber unidades de máscara para uno o más grupos de emisión adicionales. El objetivo principal de tener múltiples registros de máscara para un grupo receptor es permitir que cada contexto tenga su propio registro de máscara separado.

En el ejemplo de la Figura 4B, nueve unidades de ejecución de vectores están potencialmente incluidas en el grupo receptor. La información almacenada en la unidad de filtro indica que la primera y la última de estas unidades de ejecución deberían participar en la ejecución de la instrucción. Como se entenderá a partir de lo anterior, los grupos de problemas se pueden definir sin la unidad de máscara, pero la unidad de máscara permite la definición dinámica de grupos de problemas dentro de grupos de unidades de ejecución predefinidos.

La Figura 5 ilustra cómo se puede acceder simultáneamente a una unidad de memoria 230 desde ambas unidades CMAC 203, 205 en un grupo receptor particular. Como se muestra mediante la flecha bifurcada que apunta desde la memoria 230 a ambas unidades CMAC 203, 205, los datos pueden leerse desde la memoria 230 a ambas unidades CMAC 203, 205 o escribirse en la memoria desde ambas unidades CMAC 203, 205. La flecha conjunta de las unidades CMAC 203, 205 a la unidad de memoria 230 ilustra que las señales de control de las unidades CMAC pueden enviarse a la misma entrada de control de la unidad 230 de memoria. Ambas unidades CMAC 203, 205 pueden recibir los mismos datos de la unidad de memoria al mismo tiempo. Para escribir en la unidad de memoria, naturalmente, deben tomar turnos. Esto se puede organizar de varias maneras, conocidas por la persona experta. Por supuesto, las unidades CMAC 203, 205 son solo un ejemplo; podrían ser cualquier unidad de ejecución. Y las conexiones divididas y conjuntas se implementan realmente en la red en chip 244, que permite las conexiones entre todas las unidades en el procesador.

La figura 5 también incluye una unidad de registro vectorial 902 que puede estar dispuesta para recibir y combinar los resultados de ambas o todas las unidades de ejecución en un grupo receptor. La unidad de registro vectorial 902 también está conectada directamente a la red en chip 244 para permitir el intercambio de datos con todas las otras unidades en el procesador. Si se organiza una unidad de registro vectorial, realizará el epílogo. El epílogo implicaría combinar los resultados de la manera deseada, por ejemplo, sumarlos.

Las funciones del grupo receptor son particularmente útiles en situaciones en las que es importante que ambas unidades CMAC comiencen exactamente al mismo tiempo y funcionen de forma sincronizada. Por lo general, las funciones de múltiples expediciones se utilizan para permitir que varias unidades de ejecución de vectores ejecuten la misma instrucción, es decir, cuando se desea transmitir la misma instrucción a varias unidades de ejecución de vectores. Esto se aplica tanto a situaciones donde la sincronización de la ejecución es importante y donde varias unidades de ejecución de vectores deben recibir las mismas instrucciones, pero no es esencial que estén sincronizadas. Un ejemplo de esto último es el *clear* instrucción que se utiliza para borrar una unidad de ejecución de vectores. Para borrar todas las unidades de ejecución de vectores, un grupo receptor podría definirse como que comprende todas las unidades de ejecución de vectores y la instrucción podría enviarse a este grupo receptor.

El siguiente ejemplo se analizará sobre la base de un SIMT DSP con un número arbitrario de unidades de ejecución. Para simplificar, se supone que todas las unidades en este ejemplo son unidades de ejecución vectorial CMAC, pero en la práctica un procesador de señal digital tendrá unidades de diferentes tipos.

En muchos algoritmos y programas de procesamiento de banda base, el algoritmo se puede descomponer en varias tareas DSP, cada una de las cuales consiste en un "prólogo", una operación vectorial y un "epilogo". 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 del vector se ha completado, el resultado de la operación vectorial puede procesarse adicionalmente por código en la parte "epilogo" de la tarea. En los procesadores SIMT, normalmente solo se necesita una instrucción vectorial para realizar la operación del vector.

El diseño típico de una tarea DSP según la invención se ilustra mediante la siguiente tarea de ejemplo:

El fragmento de código en el ejemplo realiza un cálculo complejo de producto de puntos sobre 512 valores complejos y luego almacena el resultado nuevamente en la memoria. La rutina requiere que el núcleo del procesador obtenga las siguientes instrucciones.

```
.issuegroup cmac 1        ;Assume issue group 1 is selected for cmac operations

prolog:   ;Address setup
          ldi #0, r0
          out r0, cdm0_addr
          out r0, cdm1_addr
          out r0, cdm2_addr
          setcmvl.512          ; Set vector length to 512

vectorop: cmac [0],[1],[2]     ; Perform cmac operation over <vector length>
                               ; samples
          idle #cmac0          ; Stop program fetching until cmac0 is ready

epilog:   star [3]             ; Store accumulator
```

En el ejemplo anterior, las instrucciones *setcmvl, cmac* y *star* se emiten 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"). El parámetro [3] a la instrucción de star indica la dirección del puerto de red indirecto de la unidad a la que se enviarán los datos resultantes.

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, tal como *setcmvl.123* en el ejemplo anterior

2) Llevado en la propia instrucción, por ejemplo según el formato: *cmac.123,* como se muestra en la Figura 4.

3) Establecido por un registro de control, por ejemplo de acuerdo con el formato *fuera r0, cmac_vector_length*

La instrucción *idle #cmac0* instruye al controlador de flujo del programa principal para que deje de buscar 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, la instrucción "star" se busca y se envía a la unidad de ejecución del vector CMAC0. La instrucción de star ordena a la unidad de ejecución del vector CMAC que almacene el acumulador en la memoria.

Hay tres formas posibles de manejar el resultado de las unidades de ejecución de un grupo receptor. Lo más simple y más común es que las unidades de ejecución han trabajado por separado en conjuntos de datos, y que cada instrucción o secuencia de instrucciones termina individualmente. En este caso, el resultado puede manejarse de una manera común en la técnica.

Una segunda alternativa es que los resultados de dos o más unidades de ejecución que constituyen un grupo receptor se deben manejar juntas. Una forma de lograr esto sería proporcionar un archivo de registro vectorial 902 como se muestra en la figura 5, dispuesto para recibir el resultado de todo el grupo receptor y para realizar el epílogo. El epílogo implicaría combinar los resultados de la manera deseada, por ejemplo, sumarlos.

Una tercera opción sería permitir que solo una de las unidades de ejecución realice el epílogo. En este caso, para todas las unidades de ejecución excepto una en un grupo receptor, la última instrucción sería que la unidad de ejecución enviara sus datos a la unidad de ejecución del grupo receptor que debía realizar la combinación final de los resultados.

En el ejemplo anterior, los parámetros [0], [1], [2] en las instrucciones

vectorop: cmac [0],[1],[2]

indican las direcciones del puerto de red indirectas de las memorias desde donde leer y hacia donde escribir, respectivamente, para la operación, suponiendo en este caso que los datos se leen desde dos memorias y el resultado se escribe a una memoria. Por lo tanto, se da la misma información de memoria a todas las unidades de ejecución de vectores involucradas. Obviamente, normalmente no es deseable que todas las unidades de ejecución de vectores en el grupo receptor involucradas trabajen en los mismos datos. Para resolver este problema, cada unidad de ejecución de vectores tiene una tabla de asignación de puertos de red para traducir los parámetros [0], [1], [2] exactamente al puerto de red desde donde debe leerse o escribirse esta unidad de ejecución de vectores. Normalmente, cada unidad de ejecución de vectores de un grupo receptor tendrá una tabla de asignación única. Como se comprenderá en la Figura 5, las unidades de ejecución de vectores pueden trabajar en datos de las mismas unidades de memoria o de diferentes unidades de memoria. Por ejemplo, las dos unidades de ejecución de vector 203, 205 podrían realizar las funciones Σx y Σxz, respectivamente, siendo x, y y z vectores de datos obtenidos de una primera, una segunda y una tercera memoria, respectivamente.

La instrucción inactiva se usa en la arquitectura SIMT para detener las instrucciones de búsqueda de la memoria del programa hasta que una unidad de ejecución de vectores particular finalice con sus instrucciones. Cuando finaliza una unidad de ejecución de vectores, devuelve una señal para indicar al núcleo que está lista. Esta señal podría iniciar una señal de interrupción. Cuando se usan grupos receptores, preferentemente, la instrucción inactiva debe detener la obtención de instrucciones hasta que todas las unidades de ejecución de vectores en el grupo receptor hayan finalizado. Por lo tanto, el núcleo debe manejar señales de listo de todas las unidades de ejecución de vectores en el grupo receptor de forma coordinada. Normalmente, cuando las unidades de ejecución en un grupo receptor ejecutan la misma instrucción y no se producen bloqueos en las unidades de ejecución, todas las unidades de ejecución dentro del mismo grupo receptor deben liberar su señal de interrupción al mismo tiempo. Para permitir flexibilidad, es posible especificar si se debe usar lógica "y" u "o" para formar la señal de salida correspondiente. Por ejemplo, el criterio puede ser que la señal de listo haya sido recibida de todas las unidades de vectores, es decir, todas las unidades de ejecución de vectores en el grupo receptor deberían estar terminadas. Alternativamente, el criterio puede ser que una de las unidades de vectores haya emitido la señal de listo. Una forma práctica de manejar esto se muestra en la Figura 6. Una unidad lógica 904 está dispuesta para recibir la señal de listo de cada una de las unidades de ejecución de vectores 0, 1, 2 en un grupo receptor. La unidad lógica 904 también tiene información de la máscara del grupo receptor 900 discutida en conexión con la figura 3B y está dispuesta para realizar una función lógica adecuada, por ejemplo, O, Y o XO para lograr el resultado deseado.

Reivindicaciones

1. Un procesador de señal digital (200) que comprende:

- un núcleo de procesador (201) que incluye una unidad de ejecución de enteros (212) configurada para ejecutar instrucciones de enteros; y

- al menos una primera y una segunda unidades de ejecución de vectores (203, 205, 520, 530) separadas y acopladas al núcleo de procesador (201) teniendo dichas unidades de ejecución de vectores un primer y un segundo número de rutas de datos, respectivamente, estando cada una de dichas unidades de ejecución de vectores dispuesta para ejecutar instrucciones, incluyendo instrucciones de vectores que se deben realizar en múltiples palabras de datos de valor y complejos en forma de un vector y para devolver una señal cuando ha acabado indicando al núcleo que está lista, estando cada instrucción comprendida en una palabra de instrucción;

- al menos una primera unidad de memoria (230, 231) que comprende datos con los que trabajar mediante la primera y la segunda unidades de ejecución de vectores (203, 205, 520, 530)

- una red en chip que interconecta el núcleo del procesador (201), las unidades de ejecución de vectores (203, 205, 520, 530) y la al menos una unidad de memoria (230, 231),

- comprendiendo dicho procesador de señal digital una memoria de programa (211) dispuesta para contener instrucciones para la primera y segunda unidades de ejecución de vectores (203, 205, 520, 530) y lógica de emisión para emitir instrucciones, incluyendo instrucciones de vectores, a la primera y segunda unidades de ejecución de vectores, estando dicho procesador de señal digital **caracterizado por que** el procesador comprende al menos una unidad de control (705; 213, 223) para controlar, basándose en una señal receptora extraída de una palabra de instrucción al menos dos unidades de ejecución que van a recibir y ejecutar la instrucción comprendida en la palabra de instrucción al mismo tiempo y lógica para enviar la instrucción a dichas al menos dos unidades de ejecución.

2. Un procesador según la reivindicación 1, en el que se define un número de grupos receptores, comprendiendo cada grupo receptor al menos una de las unidades de ejecución (212, 203, 205) y al menos un grupo receptor que comprende más de uno de la unidad de ejecución. y la unidad de control de la emisión (705) dispuesta para seleccionar las al menos dos unidades de ejecución seleccionando un grupo receptor.

3. Un procesador según la reivindicación 1 o 2, en el que la unidad de control de emisión (705) comprende además al menos una máscara (900) asociada con al menos un grupo receptor, indicando qué unidad o unidades de ejecución en el grupo receptor deberían recibir y ejecutar la instrucción.

4. Un procesador según una cualquiera de las reivindicaciones precedentes, en el que un grupo receptor puede comprender al menos una unidad de ejecución de enteros (212) y / o al menos una unidad de ejecución de vectores (203, 205, 520, 530).

5. Un procesador según una cualquiera de las reivindicaciones precedentes, en el que al menos una unidad de ejecución comprende una tabla de mapeo para traducir la información contenida en una instrucción que indica al menos otra unidad con la que la ejecución debería interaccionar, por ejemplo, desde cuya memoria debería leer datos.

6. Un procesador según una cualquiera de las reivindicaciones precedentes, en el que cada unidad de ejecución de vectores comprende 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 contener la longitud del vector, estando dichos controladores de vectores además dispuestos para controlar la ejecución de las instrucciones.

7. Un procesador según una cualquiera de las reivindicaciones precedentes, que comprende además una unidad de archivo de registro de vectores (902), en el que las unidades de ejecución de un grupo receptor pueden estar instruidas para escribir el resultado de una ejecución de una instrucción en la unidad de archivo de registro de vectores.

8. Un procesador según una cualquiera de las reivindicaciones precedentes, en el que el descodificador de instrucciones (723) está dispuesto para informar al controlador de vectores (720, 720') sobre la instrucción que se está ejecutando en cualquier momento dado.

9. Un procesador según la reivindicación 1, en el que la al menos una unidad de ejecución de vectores (203, 205, 212) en un grupo receptor está dispuesta además para recibir una señal del receptor y para controlar la ejecución de instrucciones basadas en esta señal del receptor.

10. Un procesador según la reivindicación 1, en el que cada unidad de ejecución de vectores (203, 205, 520, 530) está dispuesta para extraer una señal de receptor de una palabra de instrucción recibida y determinar si debería participar en la ejecución de la palabra de instrucción en función de la señal del receptor.

11. 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 de señal digital programable (3) acoplado a la unidad de frontal analógica, en el que el procesador de señal digital programable es un procesador de señal digital de acuerdo con una cualquiera de las reivindicaciones 1 - 10.

12. Un dispositivo de comunicación de banda base de acuerdo con la reivindicación 11, en el que la unidad frontal (7) es una unidad frontal analógica dispuesta para transmitir y / o recibir señales de radiofrecuencia o de banda base.

13. Un dispositivo de comunicación de banda base según la reivindicación 11 o 12, siendo dicho dispositivo de comunicación de banda base para la comunicación en una red de comunicaciones inalámbricas, tal como una red de comunicaciones celulares.

14. Un dispositivo de comunicación de banda base según la reivindicación 11, siendo dicho dispositivo de comunicación de banda base un receptor de televisión.

15. Un dispositivo de comunicación de banda base según la reivindicación 11, siendo dicho dispositivo de comunicación de banda base un módem por cable.

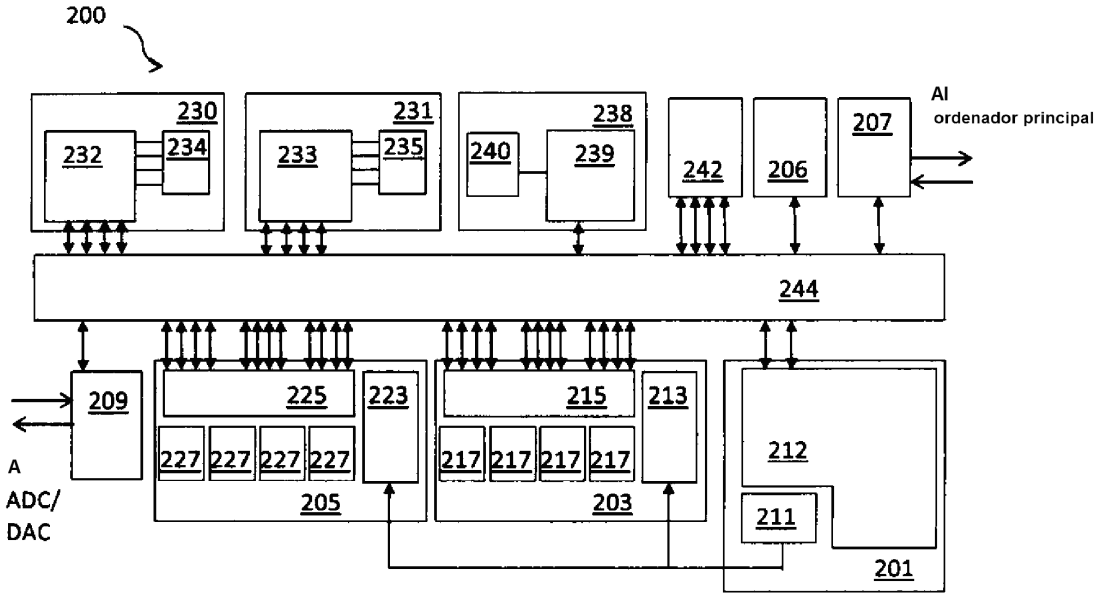

Fig. 1

	23	22	21	20	19	·	·	·	0
RUTA DE DATOS INT.	0	0	0	x	x	·	·	·	x
GRUPO RECEPTOR	0	0	0	1	x	x	·	·	·	x

Fig. 2

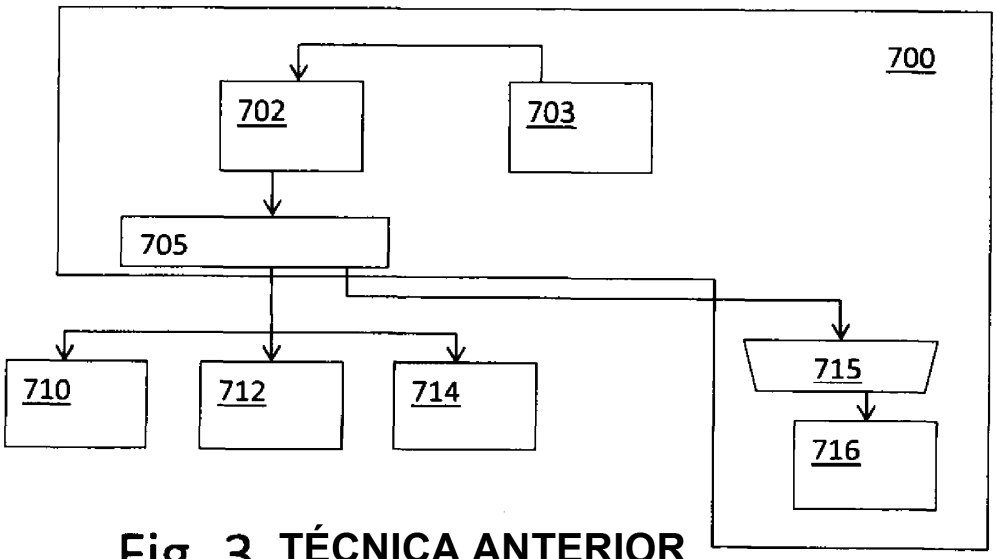

Fig. 3 TÉCNICA ANTERIOR

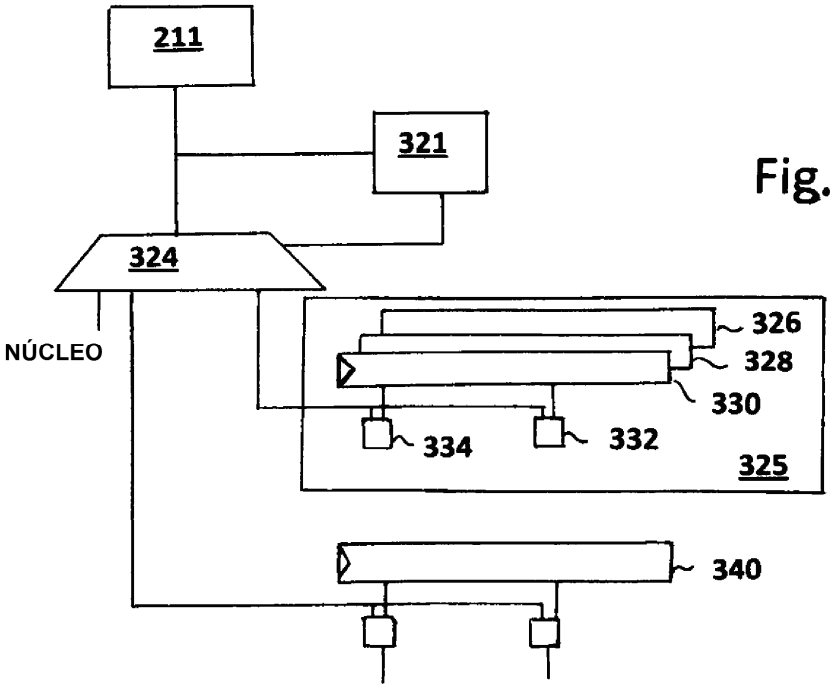

Fig. 4A

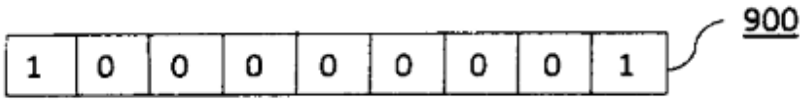

Fig. 4B

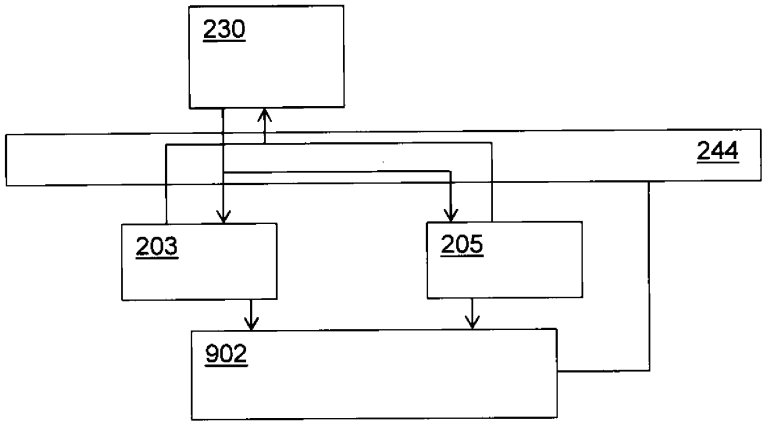

Fig. 5

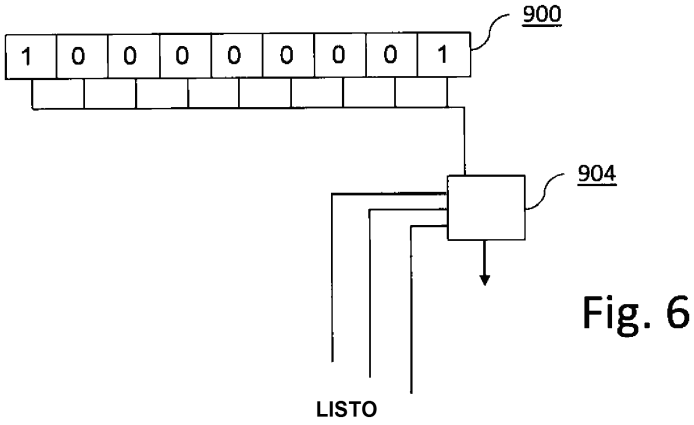

Fig. 6