(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 365 131**

(51) Int. Cl.:
H03M 13/11 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **02739608 .4**
(96) Fecha de presentación : **31.05.2002**
(97) Número de publicación de la solicitud: **1407420**
(97) Fecha de publicación de la solicitud: **14.04.2004**

(54) Título: **Procedimiento y aparato para decodificar códigos LDPC.**

(30) Prioridad: **15.06.2001 US 298480 P**
10.10.2001 US 975331

(45) Fecha de publicación de la mención BOPI:
22.09.2011

(45) Fecha de la publicación del folleto de la patente:
22.09.2011

(73) Titular/es: **QUALCOMM INCORPORATED**
5775 Morehouse Drive
San Diego, California 92121, US

(72) Inventor/es: **Richardson, Tom y**
Novichkov, Vladimir

(74) Agente: **Carpintero López, Mario**

DESCRIPCIÓN

Procedimientos y Aparato para Decodificar Códigos LDPC.

Campo de la Invención

La presente invención se dirige a procedimientos y aparatos para la detección y/o corrección de errores en datos binarios, por ejemplo, a través del uso de códigos de comprobación de la paridad, tales como los códigos de comprobación de paridad de baja densidad (LDPC).

Antecedentes

En la era moderna de la información, se usan los valores binarios, es decir, unos y ceros, para representar y comunicar diversos tipos de información, por ejemplo, video, audio, información estadística, etc. Desafortunadamente, durante el almacenamiento, la transmisión y/o el procesamiento de los datos binarios, pueden introducirse errores no intencionados, por ejemplo un uno puede cambiar a cero o viceversa.

Generalmente, en el caso de la transmisión de datos, un receptor observa cada uno de los bits recibidos en presencia de ruido o distorsión y sólo se obtiene una indicación del valor de bit. Bajo estas circunstancias se interpretan los valores observados como una fuente de bits "software". Un bit software indica una estimación preferida del valor de bit, es decir, un uno o un cero, junto con alguna indicación de la fiabilidad de la estimación. Aunque el número de errores puede ser relativamente bajo, incluso un pequeño número de errores o un nivel de distorsión puede dar como resultado que los datos resulten inutilizables, en el caso de errores de transmisión, pueden necesitar la retransmisión de los datos.

Para proporcionar un mecanismo para la comprobación de errores y, en algunos casos, para corregir los errores, los datos binarios pueden codificarse para introducir cuidadosamente una redundancia diseñada. La codificación de una unidad de datos produce lo que comúnmente se denomina como una palabra de código. Debido a su redundancia, una palabra de código a menudo incluirá más bits que la unidad de entrada de datos a partir de la cual se produjo la palabra de código.

Cuando las señales que surgen de palabras de código transmitidas se reciben o se procesan, la información redundante incluida en la palabra de código como se observa en la señal puede usarse para identificar y/o corregir los errores en las mismas o eliminar distorsión de la señal recibida para recuperar la unidad de datos original. Tal comprobación y/o la corrección de errores pueden implementarse como parte del proceso de decodificación. En ausencia de errores, o en el caso de errores o distorsión corregibles, puede usarse la decodificación para recuperar a partir de los datos fuentes que se están procesando, la unidad de datos original que se codificó. En el caso de errores no recuperables, el proceso de decodificación puede producir una indicación de que los datos originales no pueden recuperarse totalmente. Tales indicaciones de fallo de la decodificación pueden usarse para iniciar la retransmisión de los datos.

Aunque la redundancia de los datos puede aumentar la fiabilidad de los datos a almacenar o transmitidos, se hace a coste de espacio de almacenamiento y/o el uso de un valioso ancho de banda de comunicaciones. Por consiguiente, es deseable añadir redundancia de un modo eficaz, maximizando la magnitud de la capacidad de corrección/detección de errores obtenida para una cantidad determinada de redundancia introducida en los datos.

Con el uso incrementado de las líneas de fibra óptica para las comunicaciones de datos y el aumento de la tasa a la cual pueden leerse los datos y almacenarse en dispositivos de almacenamiento de datos, por ejemplo en dispositivos de disco, cintas, etc., hay una necesidad en aumento no sólo de un uso eficaz de la capacidad de almacenamiento y de transmisión de datos sino también de la capacidad de codificar y decodificar datos a altas tasas de velocidad.

Aunque la eficacia de la codificación y las altas tasas de datos son importantes, para que un sistema de codificación y/o decodificación sea práctico de usar en un amplio intervalo de dispositivos, por ejemplo, dispositivos de consumo, es importante que los codificadores y/o decodificadores se puedan implementar a un coste razonable. Por consiguiente, la capacidad de implementar de forma eficaz los esquemas de codificación/decodificación utilizados para el propósito de la corrección y/o detección de errores en términos, por ejemplo, de coste del hardware, puede ser importante.

A lo largo de años se han usado diversos tipos de esquemas de codificación para los propósitos de corrección de errores. Una clase de códigos, generalmente denominados como "códigos turbo" se inventaron recientemente (1993). Los códigos turbo ofrecen beneficios significativos sobre las técnicas de codificación más antiguas tales como los códigos convolucionales y han encontrado numerosas aplicaciones.

En conjunción con el advenimiento de los códigos turbo, ha habido un interés en aumento en otra clase de códigos relacionados, aparentemente más simples, comúnmente denominados como códigos de comprobación de paridad

de baja densidad (LDPC). Los códigos LDPC se inventaron realmente por Gallager hace unos 40 años (1961) pero sólo han pasado a primer plano recientemente. Los códigos turbo y los códigos LDPC son esquemas de codificación que se usan en el contexto de los llamados sistemas de codificación iterativos, esto es, se decodifican usando decodificadores iterativos. Recientemente, se ha mostrado que los códigos LDPC pueden proporcionar muy buen funcionamiento de detección y corrección de errores, superando o igualando el de los códigos turbo para largas palabras de código, por ejemplo tamaños de palabras de código que exceden aproximadamente 1000 bits, dada la selección adecuada de parámetros de codificación LDPC. Además, los códigos LDPC pueden decodificarse potencialmente a velocidades mucho más altas que los códigos turbo.

En muchos esquemas de codificación, las palabras de código más largas son a menudo más resistentes para los propósitos de detección y corrección de errores debido a la interacción de la codificación sobre un mayor número de bits. De este modo el uso de palabras de código largas puede ser beneficioso en términos de aumentar la capacidad de detectar y corregir errores. Esto es particularmente cierto para los códigos turbo y los códigos LDPC. De este modo, en muchas aplicaciones es deseable el uso de palabras de código largas, por ejemplo, palabras de código que exceden al millar de bits de longitud.

La principal dificultad encontrada en la adopción de la codificación de LDPC y la codificación Turbo en el contexto de largas palabras de código, donde el uso de tales códigos ofrece ser el más prometedor, es la complejidad de implementación de estos sistemas de codificación. En un sentido práctico, la complejidad se traduce directamente en coste de la implementación. Ambos sistemas de codificación son significativamente más complejos que los sistemas de codificación utilizados tradicionalmente tales como los códigos convolucionales y los códigos de Reed Solomon.

El análisis de la complejidad de los algoritmos de procesamiento de la señal usualmente se centra en las cuentas de las operaciones. Cuando se intenta explotar el paralelismo hardware en los sistemas de codificación iterativa, especialmente en el caso de los códigos LDPC, se presenta una complejidad significativa no desde el punto de vista de los requisitos de cálculo sino más bien de los requisitos de encaminamiento. La raíz de los problemas descansa en la construcción de los propios códigos.

Los códigos LDPC y los códigos turbo descansan en el intercalado de mensajes dentro de un proceso iterativo. Para realizar bien el código, el intercalado debe tener buenas propiedades de mezclado. Esto necesita la implementación de un proceso de intercalado complejo.

Los códigos LDPC se representan bien por gráficos bipartitos, a menudo llamados gráficos de Tanner, en los cuales un conjunto de nodos, los nodos de *variables*, corresponden a los bits de la palabra de código y el otro conjunto de nodos, los nodos de *restricciones*, a menudo llamados nodos de *comprobación*, corresponden al conjunto de restricciones de comprobación de paridad que define el código. Las aristas en el gráfico conectan los nodos de variables a los nodos de restricciones. Un nodo de variable y un nodo de restricción se dice que son *vecinos* si están conectados por una arista en el gráfico. Por simplicidad, generalmente asumimos que un par de nodos se conectan a lo sumo por una arista. A cada uno de los nodos de variable está asociado un bit de la palabra de código. En algunos casos, algunos de estos bits pueden ser *perforados o conocidos*, como se tratará adicionalmente, más adelante.

Una secuencia de bits asociada uno a uno con la secuencia de nodos de variables es una palabra de código si y sólo si, para cada uno de los nodos de restricción, los bits vecinos del nodo de restricción (a través de su asociación con los nodos de variables) suman cero en módulo dos, es decir comprende un número par de unos.

Los decodificadores y los algoritmos de decodificación utilizados para decodificar las palabras de código de LDPC operan intercambiando mensajes dentro del gráfico a lo largo de las aristas y actualizando estos mensajes realizando cálculos en los nodos en base a los mensajes entrantes. Tales algoritmos generalmente se denominarán como algoritmos de paso de mensajes. Cada uno de los nodos de variables en el gráfico está inicialmente provisto con un bit software, denominado *valor recibido*, que indica una estimación del valor asociado del bit como se determina por las observaciones a partir, por ejemplo del canal de comunicaciones. Idealmente, las estimaciones para bits separados son estadísticamente independientes. Este ideal puede violarse en la práctica, y a menudo se viola. Una colección de valores recibidos constituye una *palabra recibida*. Para los fines de esta aplicación podemos identificar la señal observada por el receptor, por ejemplo, en un sistema de comunicaciones con la palabra recibida.

El número de aristas conectadas a un nodo, es decir un nodo de variable o un nodo de restricción, se denomina como el grado del nodo. Un gráfico o código *regular* es aquel para el cual todos los nodos de variables tienen el mismo grado, digamos j, y todos los nodos de restricción tienen el mismo grado, digamos k. En este caso decimos que el código es un código regular (j, k). Estos fueron los códigos considerados originalmente por Gallager (1961). En contraste con un código "regular", un código irregular tiene los nodos de restricción y/o los nodos variables de diferentes grados. Por ejemplo, algunos nodos variable pueden ser de grado 4, otros de grado 3 y otros más de grado 2.

Aunque los códigos irregulares pueden ser más complicados de representar y/o de implementar, se ha mostrado

que los códigos LDPC irregulares pueden proporcionar un funcionamiento de corrección/detección de errores superior cuando se comparan con los códigos LDPC regulares.

Para describir con más precisión el proceso de decodificación introducimos la noción de un e*ncaje* en la descripción de los gráficos LDPC. Un encaje puede verse como una asociación de una arista en el gráfico con un nodo en el gráfico. Cada uno de los nodos tiene un encaje para cada una de las aristas conectadas al mismo y las aristas se "conectan" dentro de los encajes. De este modo, un nodo de grado d tiene d encajes fijados al mismo. Si el gráfico tiene L aristas entones hay L encajes sobre el lado de los nodos de variables del gráfico, llamados encajes de variables, y L encajes sobre el lado de los nodos de restricciones del gráfico, llamados encajes de restricciones. Para el propósito de identificación y ordenación, los encajes de variables pueden numerarse de 1, …, L de modo que todos los encajes de variables conectados a un nodo de variable aparecen de forma contigua. En tal caso, si los primeros tres nodos de variable tienen grados d_1, d_2, y d_3 respectivamente, entonces los encajes de variables 1, …, d_1 se conectan al primer nodo de variable, los encajes de variables $d_1 + 1$, …, $d_1 + d_2$ se conectan al segundo nodo de variable, y los encajes de variables $d_1 + d_2 + 1$, …, $d_1 + d_2 + d_3$ se conectan al tercer nodo de variable. Los encajes de los nodos de restricción pueden numerarse de forma similar de 1, …,L con todos los encajes de restricciones conectados a un nodo restricción apareciendo de forma contigua. Una arista puede verse como un emparejamiento de encajes, procediendo cada uno del par de cada lado del gráfico. De este modo, las aristas del gráfico representan un intercalado o permutación sobre los encajes desde un lado del gráfico, por ejemplo, el lado de los nodos de variables, al otro, por ejemplo el lado de los nodos de restricciones. Las permutaciones asociadas con estos sistemas son a menudo complejos, reflejando la complejidad del intercalador como se ha indicado anteriormente, requiriendo un encaminamiento complejo del paso del mensaje para su implementación.

La noción de algoritmos de paso de mensajes implementados sobre gráficos es más general que la decodificación de LDPC. La vista general es un gráfico con nodos intercambiando mensajes a lo largo de las aristas en el gráfico y realizando cálculos basados en los mensajes entrantes para producir mensajes salientes.

Un gráfico bipartito de ejemplo 100 que determina un código LDPC regular (3, 6) de longitud 10 y tasa de un medio se muestra en la Fig. 1. La longitud 10 indica que hay 10 nodos de variables $V_1 – V_{10}$, identificado cada uno con un bit de la palabra de código $X_1 – X_{10}$ (y sin perforación en este caso), identificados de modo general por el número de referencia 102. Una tasa de un medio indica que hay la mitad de nodos de comprobación que de nodos de variables, es decir, hay cinco nodos de comprobación $C_1 – C_5$ identificados por el número de referencia 106. Una tasa de un medio indica además que las cinco restricciones son linealmente independientes, como se trata más adelante. Cada una de las líneas 104 representa una arista, por ejemplo un camino de comunicación o conexión, entre los nodos de comprobación y los nodos de variable a los cuales está conectada la línea. Cada arista identifica dos encajes, un encaje de variable y un encaje de restricción. Las aristas pueden numerarse de acuerdo con sus encajes de variables y sus encajes de restricciones. La numeración de los encajes de variables corresponde con el ordenamiento de las aristas (de arriba abajo) como aparece sobre el lado de los nodos de variables al punto donde están conectadas a los nodos variables. La numeración de los encajes de restricciones corresponden con el ordenamiento de aristas (de arriba abajo) como aparece sobre el lado de los nodos de restricciones en el punto donde están conectados a los nodos de restricciones. Durante la decodificación, los mensajes se pasan en ambas direcciones a lo largo de las aristas. De este modo, como parte del proceso de decodificación los mensajes se pasan a lo largo de una arista desde un nodo de restricción a un nodo de variable y viceversa.

Aunque la Fig. 1 ilustra el gráfico asociado con un código de longitud 10, puede apreciarse que la representación del gráfico para una palabra de código de longitud 1000 sería 100 veces más complicada.

Una alternativa al uso de un gráfico para representar los códigos es usar una representación de matriz tal como la mostrada en la Fig. 2. En la representación de matriz de un código, la matriz 202 H, comúnmente denominada como la *matriz de comprobación de paridad*, incluye la conexión de aristas relevantes, la información de los nodos de variables y de los nodos de restricciones. En la matriz H, cada una de las columnas corresponde a uno de los nodos de variables mientras que cada una de las filas corresponde a uno de los nodos de restricción. Como hay 10 nodos de variables y 5 nodos de restricciones en el código de ejemplo, la matriz H incluye 10 columnas y 5 filas. La entrada de la matriz que corresponde a un nodo de variable particular y un nodo de restricción particular se pone a 1 si está presente la arista en el gráfico, es decir, si los dos nodos son vecinos, de lo contrario se fija a 0. Por ejemplo, como el nodo de variable V_1 está conectado al nodo de restricción C_1 por una arista, se coloca un uno en la esquina superior izquierda de la matriz 202. Sin embargo, el nodo de variable V_4 no está conectado al nodo de restricción C_1 de modo que se coloca un 0 en la cuarta posición de la primera fila de la matriz 202 indicando que los nodos de variable y de restricción correspondientes no están conectados. Decimos que las restricciones son linealmente independientes si las filas de H son vectores linealmente independientes sobre GF[2] (un campo de Galois de orden 2). La numeración de aristas por encajes, variables o restricciones, corresponde a la numeración de los 1 en H. La numeración de encajes de variables corresponde con la numeración de arriba abajo dentro de las columnas y procediendo de izquierda a derecha de columna a columna, como se muestra en la matriz 208. La numeración de encajes de restricciones corresponde con la numeración de izquierda a derecha a través de las columnas y procediendo de arriba a abajo de fila en fila, como se muestra en la matriz 210.

En el caso de una representación de matriz, la palabra de código X que se transmite puede representarse como un vector 206 que incluye los bits $X_1 – X_n$ de la palabra de código a procesar. Una secuencia de bits $X_1 – X_n$ es una palabra de código si y sólo si el producto de la matriz 206 y 202 es igual a cero, esto es Hx = 0.

En el contexto de la discusión de las palabras de código asociadas a los gráficos LDPC, debería apreciarse que en algunos casos la palabra de código puede *perforarse*. La perforación es el acto de eliminación de bits de una palabra de código para obtener, en efecto, una palabra de código más corta. En el caso de gráficos de LDPC esto significa que algunos de los nodos de variables en el gráfico corresponden a bits que no se transmiten realmente. Estos nodos de variables y los bits asociados con los mismos se denominan a menudo como variables de estado. Cuando se usa la perforación, el decodificador puede usarse para reconstruir la porción de la palabra de código que no está físicamente comunicada sobre el canal de comunicaciones. Cuando se transmite una palabra de código perforada el dispositivo receptor puede poblar inicialmente los valores perdidos de las palabras recibidas (bits) con unos y ceros asignados, por ejemplo de un modo arbitrario, junto con una indicación (bit software) de que estos valores son completamente no fiables, es decir, que estos valores están *borrados*. Con el fin de explicar la invención, asumimos que, cuando se usan, estos valores poblados por el receptor son parte de la palabra recibida que se va a procesar.

Consideremos por ejemplo el sistema 350 mostrado en la Fig. 3. El sistema 350 incluye un codificador 352, un decodificador 357, y un canal de comunicaciones 356. El codificador 352 incluye un circuito de codificación 353 que procesa los datos de entrada a para producir una palabra de código X. La palabra de código X incluye, para los fines de la detección y/o la corrección de errores, alguna redundancia. La palabra de código X puede transmitirse sobre el canal de comunicaciones. Alternativamente, la palabra de código X puede dividirse a través del dispositivo de selección de datos 354 en las porciones primera y segunda X', X'' respectivamente por alguna técnica de selección de datos. Una de las porciones de la palabra de código, por ejemplo, la primera porción X', puede transmitirse a continuación sobre el canal de comunicaciones a un receptor incluyendo el decodificador 357 mientras que la segunda porción X'' se perfora. Como resultado de las distorsiones producidas por el canal de comunicaciones 356, las porciones de la palabra de código transmitida pueden perderse o corromperse. Desde la perspectiva del decodificador, los datos perforados pueden interpretarse como perdidos.

En el receptor se insertan los bits software dentro de la palabra recibida para tomar el lugar de los bits perdidos o perforados. Los bits software insertados indican los bits software X'' borrados y/o los bits perdidos en la transmisión.

El decodificador 357 intentará la reconstrucción de la palabra de código completa X a partir de la palabra recibida Y, y de cualesquiera bits software insertados, y a continuación realiza una operación de decodificación de datos para producir A a partir de la palabra de código reconstruida X.

El decodificador 357 incluye un decodificador de canal 358 para reconstruir la palabra de código completa X a partir de la palabra recibida Y. Además incluye un decodificador de datos 359 para eliminar la información redundante incluida en la palabra de código para producir los datos de entrada original A a partir de la palabra de código reconstruida X.

Se apreciará que las palabras recibidas generadas conjuntamente con la codificación LDPC, pueden procesarse realizando las operaciones de decodificación de LDPC sobre las mismas, por ejemplo, las operaciones de corrección y detección de errores, para generar una versión reconstruida de la palabra de código original. La palabra de código reconstruida puede someterse a continuación a la decodificación de datos para recuperar los datos originales que se codificaron. El proceso de decodificación de datos puede ser, por ejemplo, simplemente seleccionar un subconjunto específico de bits a partir de la palabra de código reconstruida.

Las operaciones de decodificación de LDPC generalmente comprenden algoritmos de paso de mensajes. Hay muchos algoritmos de paso de mensajes potencialmente útiles y el uso de tales algoritmos no está limitado a la decodificación de LDPC. La invención actual puede aplicarse en el contexto de virtualmente cualquiera de tales algoritmos de paso de mensajes y por lo tanto puede usarse en diversos sistemas de paso de mensajes de los cuales los decodificadores LDPC son sólo un ejemplo.

Para completar daremos una breve descripción matemática de una realización de uno de los algoritmos de paso de mensajes mejor conocidos, conocido como una propagación de creencia.

La propagación de creencia para códigos LDPC (binarios) puede expresarse como sigue. Los mensajes transmitidos a lo largo de las aristas del gráfico se interpretan como un log de probabilidades log (p_0 / p_1) siendo pi el bit asociado con el nodo variable. En este punto (p0, p1) representan una distribución de probabilidad condicional sobre el bit asociado. Los bits software proporcionados al decodificador por el receptor también se dan en la forma de un registro de probabilidad. De este modo, los valores recibidos, es decir, los elementos de la palabra recibida son log de probabilidades de los bits asociados condicionadas por la observación de los bits proporcionados por el canal de comunicaciones. En general, un mensaje m representa el log de probabilidad m y un valor recibido y representa un log de probabilidad y. Para los bits perforados el valor recibido y se pone a 0, indicando $p_0 = p_1 = ½$.

Consideremos las normas de paso de mensajes de la propagación de creencia. Los mensajes se denominan por

m^{C2V} para mensajes desde los nodos de comprobación a los nodos de variable y por m^{V2C} para los mensajes desde los nodos de variables a los nodos de comprobación. Consideremos un nodo de variable con d aristas. Para cada una de las aristas j = 1, …d, sea $m^{C2V}(i)$ el mensaje entrante sobre la arista i. Muy al comienzo del proceso de decodificación fijamos $m^{C2V} = 0$ para cada arista. Entonces los mensajes salientes vendrán dados por

$$m^{V2C}(j) = y + \Sigma^{d}_{i=1} m^{C2V}(i) - m^{C2V}(j)$$

En los nodos de comprobación es más conveniente representar los mensajes usando su 'signo' y magnitudes. De este modo para el mensaje m, sea $m_p \in GF[2]$ que denota la 'paridad' del mensaje, es decir, $m_p = 0$ si $m \geq 0$ y $m_p = 1$ si $m < 0$. Adicionalmente, sea $m_r \in [0, \infty]$ que denota la magnitud de m. De este modo, tenemos $m = -1^{mp} m_r$. En el nodo de comprobación las actualizaciones para m_p y m_r están separadas. Tenemos, para un nodo de comprobación del grado d

$$mp^{C2V}(j) = (\Sigma^{d}_{i=1} m_p^{V2C}(I)) - m_p^{V2C}(j),$$

donde toda la adición es sobre GF[2], y

$$m_r^{C2V}(j) = F^{-1}((\Sigma^{d}_{i=1} F(m_r^{V2C}(i))) - F(m_r^{V2C}(j))),$$

donde definimos F(x)=log coth (x/2). (En ambas de las ecuaciones anteriores el superíndice V2C indica mensajes entrantes en el nodo de comprobación). Observamos que F es igual a su propia inversa $F^{-1}(x) = F(x)$.

La mayor parte de los algoritmos de paso de mensajes pueden verse como aproximaciones a la propagación de creencia. Se apreciará que en cualquier implementación digital práctica los mensajes comprenderán un número finito de bits y normas de adaptación de mensajes debidamente adaptadas.

Será evidente que la complejidad asociada con la representación de los códigos de LDPC para palabras de código largas es de enormes proporciones, al menos para las implementaciones por hardware que intentan explotar el paralelismo. Además, puede ser difícil implementar el paso de mensajes de modo que soporte el procesamiento a altas velocidades.

Para hacer un uso más práctico de los códigos LDPC, se necesitan procedimientos de representación de los códigos LDPC correspondientes a palabras de código largas en una forma eficaz y compacta reduciendo por lo tanto la cantidad de información requerida para representar el código, es decir, describir el gráfico asociado. Además, hay necesidad de técnicas que permitan el paso de los mensajes asociado con múltiples nodos y múltiples aristas, por ejemplo, cuatro o más nodos o aristas, a realizar en paralelo en un modo controlado de forma fácil, permitiendo por lo tanto incluso que largas palabras de código se decodifiquen de forma eficaz en una cantidad de tiempo razonable. Existe además la necesidad de una arquitectura de decodificador que es suficientemente flexible para decodificar varios códigos LDPC diferentes. Esto es porque muchas aplicaciones requieren códigos de diferentes longitudes y tasas. Incluso es más deseable una arquitectura que permita que la especificación de un código LDPC particular sea programable.

Breve Descripción de las Figuras

La Figura 1 ilustra una representación de un gráfico bipartito de un código LDPC regular de ejemplo de longitud diez.

La Figura 2 es una representación de matriz del código ilustrado gráficamente en la Fig. 1.

La Figura 3 ilustra la codificación, transmisión y decodificación de datos.

La Figura 4 ilustra una representación de un gráfico bipartito de un código LDPC irregular de ejemplo

La Figura 5, que comprende la combinación de las Fig 5a hasta la 5d, ilustra etapas realizadas como parte de la operación de decodificación LDPC de acuerdo con el código LDPC ilustrado en la Fig. 4.

La Figura 6 es una representación gráfica de un código LDPC pequeño que se usa como base de un código LDPC mucho más largo para presentar un ejemplo de acuerdo con la presente invención.

La Figura 7 ilustra la representación de la matriz de comprobación de paridad del código LDPC pequeño ilustrado gráficamente en la Fig. 6.

La Figura 8 ilustra cómo pueden estar dispuestas las aristas en el código mostrados en la Fig. 6, por ejemplo, numeradas en orden desde el lado de los nodos de variables y cómo aparecerían las mismas aristas desde el lado de los nodos de restricciones.

La Figura 9 ilustra un sistema para realizar una operación de decodificación de LDPC en serie.

La Figura 10 ilustra gráficamente el efecto de hacer tres copias del gráfico LDPC pequeño mostrado en la Fig. 6.

La Figura 11 ilustra la representación de la matriz de comprobación de paridad del gráfico de LDPC mostrado en la Fig. 10.

La Figura 12 ilustra cómo pueden estar dispuestas las aristas en el código mostrado en la Fig. 11, por ejemplo numeradas en orden desde el lado de los nodos de variables y cómo aparecerán las mismas aristas desde el lado de los nodos de restricciones.

La Figura 13 ilustra el efecto de reemplazar las matrices identidad de 3x3 mostradas en la Fig. 11 con matrices de permutaciones cíclicas de acuerdo con una realización de ejemplo de la presente invención.

La Figura 14 ilustra cómo se pueden numerar las aristas en el código mostrado en la Fig. 13 desde el lado de los nodos de variable, y como aparecerán las mismas aristas desde el lado de los nodos de restricciones después de someterse a una permutación cíclica de acuerdo con la invención.

La Figura 15 ilustra un decodificador LDPC implementado de acuerdo con la presente invención que convierte a vectores el decodificador de la Fig. 9.

Las Figuras 16 y 17 ilustran los otros decodificadores LDPC implementados de acuerdo con la presente invención.

Se llama la atención de un documento de Boutillion E y otros: "Diseño del primer-decodificador de código", PROC. DEL SIMPOSIO INTERNACIONAL SOBRE CÓDIGOS TURBO Y TEMAS RELACIONADOS, BREST, FRANCIA, 4 de Septiembre de 2000, páginas 459 – 462, XP08011934 y otros. El documento describe una arquitectura de decodificador para un código de Comprobación de Paridad de Baja Densidad. En particular, se describe una arquitectura que usa N memorias distintas que se dirigen separadamente. En el sistema se seleccionan N elementos aleatoriamente uno cada vez de cada una de las N memorias dirigidas separadamente y de forma aleatoria.

Sumario de la invención

De acuerdo con la presente invención, como se muestra en la reivindicación 1 se proporcionan un aparato para la realización de operaciones de decodificación de paso de mensajes, y procedimientos para la realización del procesamiento de decodificación de paso de mensajes, como se muestra en las reivindicaciones 17 y 26. Las realizaciones de la invención se reivindican en las reivindicaciones dependientes.

La presente invención se dirige a procedimientos y un aparato para realizar las operaciones de decodificación sobre palabras que usan las técnicas de decodificación de paso de mensajes. Las técnicas de la presente invención están particularmente bien adaptadas para su uso con códigos LDPC largos, por ejemplo, palabras de código de más de 750 bits de longitud, pero pueden usarse también para longitudes más cortas. Las técnicas y el aparato de la presente invención pueden también usarse para diseño y decodificación de gráficos donde se usan otros tipos de algoritmos de paso de mensajes. Para fines de explicación de la invención, sin embargo, se describirán los decodificadores LDPC y las técnicas de decodificación de ejemplo.

Las técnicas de la presente invención permiten la decodificación de gráficos de LDPC que poseen una cierta estructura jerárquica en la cual un gráfico LDPC completo parece estar, en gran parte, constituido de múltiples copias, digamos Z, de un gráfico Z veces más pequeño. Las Z copias del gráfico pueden ser idénticas. Para ser precisos, nos referiremos a un gráfico más pequeño como el gráfico *proyectado*. La técnica puede apreciarse mejor considerando en primer lugar un decodificador que decodifica Z pequeños gráficos LDPC idénticos de forma síncrona y en paralelo. Consideremos un decodificador de paso de mensajes para un gráfico único pequeño LDPC. El decodificador implementa una secuencia de operaciones que corresponden a un algoritmo de paso de mensajes. Consideremos ahora aumentar el mismo decodificador de modo que decodifica tales Z gráficos idénticos LDPC de forma síncrona y en paralelo. Cada una de las operaciones en el algoritmo de paso de mensajes se repite Z veces. Obsérvese que la eficacia del proceso de decodificación mejorará porque, en total, la decodificación procede Z veces más rápida y porque los mecanismos de control requeridos para controlar el proceso de paso de mensajes no necesitan replicarse para las Z copias sino que pueden en cambio compartirse por las Z copias. También podemos ver que el decodificador anterior de Z copias en paralelo como un decodificador de *vectores*. Podemos ver el proceso de realización de Z copias del gráfico más pequeño como una conversión a vectores del gráfico más pequeño (proyectado): Cada uno de los nodos del gráfico más pequeño se convierte en un *nodo de vector*, que comprende Z nodos, cada una de las aristas del gráfico más pequeño se convierte en una *arista de vector*, consistente de Z aristas, cada uno de los mensajes intercambiados en la decodificación del gráfico más pequeño se convierte en un *mensaje de vectores*, que comprende Z mensajes.

La presente invención obtiene la eficacia de la conversión a vectores descrita anteriormente mientras que se modifica el mismo de modo que el decodificador de vectores es de hecho la decodificación de un gran gráfico, Z veces mayor que el gráfico proyectado. Esto se realiza interconectando las Z copias del gráfico proyectado de un modo controlado. Específicamente, permitimos que las Z aristas dentro de una arista de vectores sufra una

permutación, o intercambio, entre las copias del gráfico proyectado a medida que avanzan, por ejemplo, desde el lado de los nodos de variables al lado de los nodos de restricciones. En el proceso de paso del mensaje de vectores correspondiente a los Z gráficos proyectados en paralelo este intercambio se implementa permutando los mensajes dentro de un mensaje de vectores a medida que pasan desde un lado del gráfico de vectores al otro.

Consideramos la indexación de los gráficos de LDPC proyectados por 1, j, …, Z. En el decodificador estrictamente en paralelo los nodos de variables en el gráfico j se conectan sólo con nodos de restricciones en el gráfico j. De acuerdo con la presente invención, tomamos una arista de vectores, incluyendo la arista correspondiente a cada una de las copias del gráfico, y permitimos una permutación dentro de las Z aristas, por ejemplo permitimos que los encajes de restricciones correspondientes a las aristas dentro de las aristas de vectores permuten, es decir, que se reordenen. A partir de ahora a menudo nos referiremos a las permutaciones, es decir, las reordenaciones, dentro de las aristas de vectores como *rotaciones*.

De este modo, de acuerdo con la presente invención, puede representarse un gráfico relativamente grande, es decir, describirse, usando relativamente poca memoria. Por ejemplo, un gráfico puede representarse almacenando la información que describe el gráfico proyectado y la información que describe las rotaciones. Como alternativa, la descripción del gráfico puede realizarse como un circuito que implementa una función que describe la conectividad del gráfico.

Por consiguiente, la técnica de representación de gráficos de la presente invención facilita las implementaciones de gráficos, por ejemplo, convertidos a vectores. Además, las técnicas de representación de gráficos de la presente invención pueden usarse para soportar la decodificación de gráficos regulares o irregulares, con o sin variables de estado. La información que describe los grados de los nodos en el gráfico proyectado puede almacenarse y proporcionarse a un elemento de procesamiento de nodos de vectores. Obsérvese que todos los nodos que pertenecen a un nodo de vector tendrán el mismo grado de modo que la información de grado se requiere sólo para un gráfico proyectado.

En diversas realizaciones, el decodificador se hace programable permitiendo por lo tanto programarse con múltiples descripciones de gráficos, por ejemplo, como se expresa en términos de gráficos proyectados almacenados y la información de rotación almacenada o en términos de una función implementada. Por consiguiente, los decodificadores de la presente invención pueden programarse para decodificar una gran cantidad de diferentes códigos, por ejemplo, tanto regular como irregular. En algunas realizaciones particulares el decodificador se usa para un gráfico fijo o para grados fijos y esta información. En tales realizaciones la información de descripción del gráfico puede ser programada de antemano o implícita. En tales casos el decodificador puede ser menos flexible que las realizaciones programables pero se ahorran los recursos requeridos para soportar la funcionalidad de la programación.

De acuerdo con una realización de la presente invención, se proporciona una memoria de mensajes que incluye filas de localizaciones de memoria, correspondiendo cada fila a los mensajes asociados con una copia del gráfico proyectado. Los mensajes correspondientes a los Z gráficos proyectados múltiples se almacenan para formar columnas de Z mensajes por columna, de modo que una columna corresponde con un mensaje de vectores. Esta disposición de la memoria permite que los mensajes de vectores, por ejemplo, un conjunto de Z mensajes, correspondientes a la arista de vectores se lean o se escriban en memoria como una unidad, usando por ejemplo una instrucción SIMD para acceder a todos los Z mensajes en una columna en una operación. De este modo la memoria soporta la lectura y escritura de mensajes de vectores como unidades. Por consiguiente, la presente invención evita la necesidad de proporcionar una dirección de lectura/escritura diferentes para cada uno de los mensajes individuales en un conjunto de Z mensajes.

En uno o más puntos en el procesamiento de paso de mensajes, después de la lectura de la memoria, los Z mensajes están sujetos a una operación de permutación, es decir una operación de reordenamiento. La operación de reordenamiento puede ser una operación de rotación, o una rotación para acortar. Estas operaciones de rotación corresponden con las rotaciones asociadas con las aristas de vectores que interconectan las Z copias del gráfico proyectado para formar el único gran gráfico. Esta rotación puede aplicarse, por ejemplo antes de suministrar los mensajes al procesador de nodos de vectores correspondiente (restricción o variable). Como alternativa, la rotación puede aplicarse posteriormente al procesamiento por el procesador de nodos de vectores.

La rotación puede implementarse usando un simple dispositivo de conmutación que conecta, por ejemplo, la memoria de mensajes con la unidad de procesamiento de nodos de vectores y reordena esos mensajes a medida que pasan desde la memoria a la unidad de procesamiento de los nodos de vectores. En tal realización de ejemplo, uno de los mensajes en cada uno de los mensajes de vectores leído desde la memoria se suministra a la unidad correspondiente de las Z unidades de procesamiento de nodos en paralelo, dentro de un procesador de nodos de vectores, como se determina por la rotación aplicada al mensaje de vectores por el dispositivo de conmutación. Una operación de rotación como se implementa por el dispositivo de conmutación puede aplicarse también o alternativamente al mensaje de vectores antes de su escritura en la memoria y posterior procesamiento de nodos.

La descripción almacenada o calculada del gráfico proyectado puede incluir, por ejemplo, información sobre el orden en el cual los mensajes en una fila correspondientes al gráfico proyectado se leerán y/o se escribirán en memoria durante el procesamiento de nodos de restricciones y/o variables. Los mensajes de todo el gran gráfico se almacenan en múltiples filas, correspondiendo cada una de las filas a una copia diferente del gráfico pequeño, disponiéndose las filas para formar columnas de mensajes. Cada una de las columnas de mensajes representa un mensaje de vectores, que puede accederse como una única unidad. De este modo, la información sobre cómo acceder a los mensajes en una fila de un gráfico proyectado puede usarse para determinar el orden en el cual los mensajes de vectores correspondientes a las múltiples copias del gráfico proyectado se acceden de acuerdo con la presente invención.

La variación del orden en el cual se leen y/o se escriben los mensajes de vectores en la memoria de acuerdo a si la operación de lectura/escritura corresponde un procesamiento del lado de los nodos de variables o del lado de los nodos de restricciones puede describirse como una primera permutación realizada sobre los mensajes. Esta permutación corresponde al intercalador asociado con el gráfico proyectado. Para representar el gran gráfico del decodificador a partir del gráfico del decodificador proyectado, un segundo conjunto de información de permutación, por ejemplo, la información de rotación, se almacena además de la información del orden de acceso a los mensajes de vectores (por ejemplo, la columna). La segunda información de permutación (por ejemplo, la información de rotación), que representa la información de control de conmutación, indica cómo deberían reordenarse los mensajes en cada uno de los mensajes de vectores, por ejemplo, las columnas de mensajes, cuando, por ejemplo se leen y/o se escriben en la memoria. Esta permutación de dos etapas divide la permutación más grande que describe el gráfico LDPC completo en dos partes implementadas a través de diferentes mecanismos.

En una realización particular, se usa una permutación cíclica como el segundo nivel de permutación debido a la facilidad con la que puede implementarse tal permutación y la compacidad de la descripción. Este caso motiva el uso de la rotación de términos para describir esta permutación de segundo nivel para fines de explicación. Sin embargo, se entenderá que el segundo nivel de permutación no se necesita limitarse a las rotaciones y puede implementarse usando otros esquemas de reordenación.

En diversas realizaciones de la presente invención, el decodificador genera salidas de múltiples bits software con un bit, por ejemplo, como bit de signo o bit de paridad de cada una de las salidas software, correspondiendo a una salida de decisión hardware del decodificador, por ejemplo, la palabra de código original en el caso en el que se hayan corregido todos los errores o no haya ningún error presente en la palabra recibida. La salida del decodificador, por ejemplo la palabra de código recuperada, puede procesarse a continuación para recuperar los datos originales que se usaron en el instante de la codificación para producir la palabra de código transmitida.

De acuerdo con una característica de la presente invención las salidas hardware y/o software producidas después de cada una de las iteraciones completas del procesamiento de nodos de variables se examinan para determinar si las restricciones de comprobación de paridad indicativas de una palabra de código se satisfacen por las decisiones hardware actual. Este proceso de comprobación también disfruta de los beneficios de la estructura de permutación dividida en las dos etapas del gráfico. El proceso de decodificación iterativo (paso de mensajes) puede pararse una vez que se detecta la recuperación de la palabra de código de este modo. Por consiguiente, en el caso de señales relativamente libres de error, la decodificación puede completarse y detectarse inmediatamente, por ejemplo, después de dos o tres iteraciones del proceso de decodificación de paso de mensajes. Sin embargo, en el caso de palabras recibidas que incluyen más errores, pueden ocurrir numerosas iteraciones del proceso de decodificación antes de que la decodificación sea satisfactoria o el proceso se pare debido a una restricción de temporización.

La detección de la indicación de decodificación satisfactoria, de acuerdo con la presente invención, permite un uso más eficaz de recursos comparado con los sistemas que asignan un número fijo de iteraciones de decodificación para cada una de las palabras recibidas.

Como las técnicas de decodificación de la presente invención permiten un gran número de operaciones de decodificación, por ejemplo operaciones de procesamiento del decodificador de nodos de restricciones y/o variables, a realizar en paralelo, los decodificadores de la presente invención pueden usarse para decodificar palabras recibidas a altas velocidades. Además, dada la técnica novedosa de la presente invención utilizada para representar grandes gráficos y/o el control del paso de mensajes para operaciones de decodificación asociadas con tales gráficos, las dificultades de almacenamiento de las descripciones de grandes gráficos y el control de encaminamiento de sus mensajes se reducen y/o se superan.

Ciertas generalizaciones de los códigos LDPC y técnicas de decodificación de la invención incluyen la codificación/decodificación sobre mayores alfabetos, no simplemente bits, los cuales tienen dos posibles valores, pero mayor número de posibilidades. Los códigos donde los nodos de restricciones representan restricciones distintas que las restricciones de la comprobación de paridad también pueden decodificarse usando los procedimientos y el aparato de la presente invención. Otras generalizaciones relevantes a las que puede aplicarse la invención incluyen situaciones en las que un algoritmo de paso de mensajes se implementará sobre un gráfico y se tiene la opción de diseñar el gráfico. Resultará evidente para los especialistas en la técnica, a la vista de la presente

solicitud de patente, cómo para aplicar las técnicas de la presente invención a estas situaciones más generales.

Numerosas ventajas adicionales, características y aspectos de las técnicas de decodificación y los decodificadores de la presente invención serán evidentes a partir de la descripción detallada que sigue a continuación.

Descripción detallada de la invención

Como se ha tratado anteriormente, los procedimientos de decodificación y el aparato de la presente invención se describirán para propósitos de explicación en el contexto de una realización del decodificador de LDPC. En primer lugar se describirán las etapas involucradas en la decodificación de un código LDPC con referencia a las Fig. 4 y 5 seguido por una descripción más detallada de diversas características de la presente invención.

La Figura 4 ilustra un código LDPC irregular de ejemplo que usa un gráfico bipartito 400. El gráfico incluye m nodos de comprobación 402, n nodos de variables 406, y una pluralidad de aristas 404. Los mensajes entre los nodos de comprobación y los nodos de variables se intercambien sobre las aristas 404. Los bits de entrada software y_1 hasta y_n, corresponden a la palabra recibida Y, y las salidas software (o hardware) x_1 hasta x_n se indican por la referencia numérica 408. El nodo de comprobación de orden m se identifica usando la referencia numérica 402', el nodo de variable de orden n se identifica usando la referencia numérica 406' mientras que la entrada software de orden n y_n y las salida software de orden n x_n se indican en la Fig. 4 usando los números de referencia 410 y 409 respectivamente.

Los nodos de variables 406 procesan los mensajes de los nodos de restricciones 402 junto con los valores de entrada software desde la palabra recibida y_1, …y_n para actualizar el valor de las variables de salida x_1, …x_n correspondientes a los nodos de variables y para generar mensajes para los nodos de restricciones. Un mensaje se genera por un nodo de variable para cada una de las aristas conectadas al nodo de variable. El mensaje generado se transmite a lo largo de la arista desde el nodo de variable al nodo de restricción conectado a la arista. Para fines de explicación, los mensajes de los nodos de variables a los nodos de restricciones se indicarán, de vez en cuando, en la presente solicitud usando la abreviatura V2C mientras que los mensajes desde los nodos de variables a los nodos de restricciones se indicarán usando la abreviatura C2V. Los índices pueden añadirse a las componentes V y C de esta abreviatura para indicar el nodo particular de los nodos de variables y los nodos de restricciones que sirve como la fuente/destino de un mensaje particular. Cada uno de los nodos de restricciones 402 es responsable del procesamiento de los mensajes recibidos desde los nodos variables a través de las aristas conectadas al nodo de restricción particular. Los mensajes V2C recibidos desde los nodos de variables se procesan por los nodos de restricciones 402 para generar los mensajes C2V que se transmiten a continuación de vuelta a lo largo de las aristas conectadas a cada uno de los nodos de restricciones. Los nodos de variables 406 procesan a continuación los mensajes C2V, junto con los valores de entrada software, para generar y transmitir nuevos mensajes V2C, y generar las salidas software x_i. La secuencia de realización del procesamiento en los nodos de variables 406 comprende: la transmisión de los mensajes generados a los nodos de comprobación 402, generando en los nodos de variables salidas software x_i y recibiendo los mensajes desde los nodos de comprobación, puede realizarse repetidamente, es decir, iterativamente, hasta que las salidas x_i desde los nodos de variables 406 indican que la palabra de código se ha decodificado satisfactoriamente o algún otro criterio de parada, por ejemplo, se ha satisfecho la terminación de un número fijo de iteraciones de paso de mensajes. Debería apreciarse que la secuencia de operaciones descrita anteriormente no es necesario que ocurra estrictamente en el orden descrito. El procesamiento de nodos puede continuar de forma asíncrona y el procesamiento de nodos de variables y restricciones puede ocurrir simultáneamente. Sin embargo la lógica del proceso iterativo es como se ha descrito.

Los mensajes, V2C y C2V, pueden ser de uno o más bits, por ejemplo de K bits cada uno, donde K es un valor entero positivo distinto de cero. De forma similar las salidas software x_i pueden ser de uno o de múltiples bits. Los mensajes y salidas de múltiples bits proporcionan la oportunidad de pasar la información de confianza o de fiabilidad en el mensaje o salida. En el caso de una salida multi-bit, (software), el signo del valor de la salida software puede usarse para proporcionar la salida hardware de un único bit del proceso de decodificación correspondiente a un nodo de variable, por ejemplo los bits de la palabra de código decodificada. Los valores de salida software pueden corresponder a valores software decodificados o, como alternativa, a una información llamada extrínseca (excluyendo la información de entrada correspondiente) que puede usarse en otros procesos iterativos más largos dentro de los cuales el decodificador LDPC es sólo un módulo.

El proceso de paso de mensajes iterativo asociado con la decodificación de un código LDPC se tratará ahora adicionalmente con respecto a las Fig. 5a hasta 5d.

Cuando se decodifica un código LDPC, el procesamiento en cada uno de los nodos de restricciones y nodos de variables puede realizarse independientemente. Por consiguiente, el procesamiento de nodos de variables y/o de restricciones puede realizarse un nodo cada vez, por ejemplo, en secuencia, hasta que algunos o todos los procesamientos de los nodos de variables y restricciones se han completado para una iteración particular del proceso de decodificación. Esto permite proporcionar y reutilizar una única unidad de procesamiento hardware, si se desea, realizar el procesamiento asociado con cada uno de los nodos de variables y/o de restricciones. Otra

característica significativa de la decodificación LDPC es que los mensajes V2C y C2V utilizados durante una iteración del procesamiento particular no necesitan generarse al mismo tiempo, por ejemplo durante la misma iteración del proceso. Esto permite implementaciones donde el procesamiento de nodos de restricciones y variables puede realizarse en paralelo sin tener en cuenta cuándo se actualizaron por última vez los mensajes actualizados. Siguiendo un número suficiente de actualizaciones de mensajes y de iteraciones en donde todos los nodos de variables y restricciones procesan los mensajes recibidos y generan mensajes actualizados, la salida (hardware) de los nodos de variables convergerá asumiendo que el gráfico se diseño adecuadamente y que no quedan errores sin corregir en la palabra recibida que se está procesando.

Dado que el procesamiento de cada uno de los nodos de comprobación y nodos de variable puede verse como una operación independiente, el procesamiento iterativo realizado en un nodo de comprobación de ejemplo único C_n 502' y el nodo de variable V_n 506' se tratarán con más detalle con referencia a las Fig. 5a – 5d. Para fines de descripción pensaremos que los valores del mensaje, la entrada software y los valores de salida son números. Un número positivo corresponde a una decisión hardware del bit 0 y un número negativo corresponde a una decisión hardware de bit 1. Mayores magnitudes indican una mayor fiabilidad. De este modo, el número cero indica una total falta de fiabilidad y el signo (positivo o negativo) es irrelevante. Esta convención es consistente con la práctica normalizada en la cual los valores software (mensajes, recibidos y valores de salida) representan log de probabilidades de bits asociados, es decir, los valores software toman la forma

log (probabilidad de que el bit sea 0 / probabilidad de que el bit sea 1)

en la que la probabilidad está condicionada sobre una variable aleatoria, por ejemplo, la observación física del bit del canal de comunicaciones en el caso de un valor recibido.

La Fig. 5a ilustra la etapa inicial en un proceso de decodificación LDPC. Inicialmente, el nodo de variable V_n 506' se suministra con la entrada software, por ejemplo los valores recibidos (1 ó más bits) y_n desde la palabra recibida a procesar. Los mensajes C2V al comienzo de la operación de decodificación y la salida software X_n 509 se fijan inicialmente a cero. En base a las entradas recibidas, por ejemplo, los mensajes C2V de valor cero y la entrada y_n, el nodo de variable V_n 506' genera un mensaje V2C para cada uno de los nodos de comprobación al cual está conectado. Típicamente, en la etapa inicial, cada uno de estos mensajes será igual a y_n.

En la figura 5b se muestran los mensajes V2C generados que se transmiten a lo largo de cada uno de las aristas conectadas al nodo de variable V_n 506'. De este modo, los mensajes V2C actualizados se transmiten a cada uno de los nodos de comprobación 502 acoplados al nodo de variable V_n506' incluyendo el nodo de comprobación C_m 502'.

Además de generar los mensajes V2C, el procesamiento de los nodos de variables da como resultado la actualización de la salida software X_n 509' correspondiente al nodo de variable que hace el procesamiento. La salida software X_n se muestra actualizándose en la Fig. 5c. Aunque se muestran como etapas diferentes, la salida software puede ser una salida en el mismo tiempo que se sacan los mensajes V2C.

Como se tratará adicionalmente más adelante, de acuerdo con algunas realizaciones de la presente invención, las salidas software (o sus decisiones hardware asociadas) pueden usarse para determinar cuándo se ha recuperado una palabra de código a partir de la palabra recibida, es decir, cuando se han satisfecho las restricciones de paridad por los valores de salida. Esto indica una decodificación satisfactoria (aunque la palabra de código encontrada puede ser incorrecta, es decir, no la que se transmitió) por lo tanto permitiendo que el proceso de decodificación iterativo se pare en el momento oportuno, por ejemplo antes de que se complete el número máximo fijado permitido de iteraciones de paso de mensajes.

El procesamiento de los nodos de comprobación puede realizarse una vez que el nodo de comprobación, por ejemplo el nodo de comprobación C_m 502', recibe mensajes V2C a los largo de las aristas a los que está conectado. Los mensajes V2C recibidos se procesan en el nodo de comprobación para generar los mensajes C2V actualizados, uno por cada una de las aristas conectadas al nodo de comprobación particular. Como resultado del procesamiento del nodo de comprobación, el mensaje C2V transmitido de vuelta al nodo de variable a lo largo de una arista dependerá del valor de cada uno de los mensajes V2C recibidos sobre las otras aristas conectadas al nodo de comprobación pero no (usualmente y preferentemente pero no necesariamente) una vez recibido el mensaje V2C desde el nodo de variable particular al cual se está transmitiendo el mensaje C2V. De este modo, los mensajes C2V se usan para transmitir información generada a partir de mensajes recibidos desde los nodos de variables distintos que el nodo al cual se está transmitiendo el mensaje.

La Fig. 5d ilustra el paso de los mensajes C2V actualizados a los nodos de variables incluyendo el nodo 506'. En particular, en la Fig. 5d el nodo de restricción C_m 502' se muestra sacando dos mensajes C2V actualizados suministrándose el mensaje C_m2V_n actualizado al nodo de variable V_n 506'. V_n 506' también recibe mensajes C_2V_n actualizados adicionales de otros nodos de restricciones a los que está conectado.

Con la recepción de mensajes actualizados C2V, el procesamiento de nodos de variables puede repetirse para generar los mensajes V2C actualizados y las salidas software. A continuación puede repetirse la actualización de los

mensajes C2V y así sucesivamente hasta que se satisface el criterio de parada del decodificador.

De este modo, el procesamiento mostrado en las Fig. 5a – 5d se repetirá después de la primera iteración, usando valores del mensaje actualizados en lugar de los valores iniciales, hasta que se para el proceso de decodificación.

La naturaleza iterativa del proceso de decodificación LDPC, y el hecho de que el procesamiento en los nodos individuales pueda realizarse independientemente del procesamiento de los otros nodos proporciona un alto grado de flexibilidad cuando se implementa un decodificador LDPC. Sin embargo, como se trató anteriormente, la mera complejidad de las relaciones entre las aristas y los nodos puede dificultar el almacenamiento de la información de las relaciones de las aristas, es decir, la descripción del gráfico. Incluso de forma más importante, la complejidad del gráfico puede hacer difícil de implementar el paso de mensajes en implementaciones en paralelo donde múltiples mensajes se pasan al mismo tiempo.

Las implementaciones prácticas del decodificador LDPC a menudo incluyen una memoria de aristas para almacenar los mensajes que pasan a lo largo de las aristas entre los nodos de restricciones y/o los nodos de variables. Además incluyen un descriptor de gráficos a veces denominado como un mapa de permutaciones lo cual incluye información de especificación de las conexiones de las aristas, o emparejamiento de encajes, definiendo por lo tanto el gráfico de decodificación. Este mapa de permutaciones puede implementarse como datos almacenados o como un circuito que calcula o implica la permutación. Además de la memoria de aristas, son necesarias una o más unidades de procesamiento de nodos para realizar el procesamiento real asociado con un nodo.

Las implementaciones del decodificador LDPC software son posibles, en las que se usa el software para controlar una CPU para que funcione como una unidad de procesamiento de vectores y para controlar el paso de mensajes usando una memoria acoplada a la CPU. En implementaciones software, puede usarse también una única memoria para almacenar la descripción del gráfico del decodificador, los mensajes de aristas, así como las rutinas del decodificador utilizadas para controlar la CPU.

Como se tratará más adelante, en diversas implementaciones de la invención pueden usarse uno o más memorias de aristas. En una realización de ejemplo de múltiples memorias de arista, se usa una primera memoria de aristas para el almacenamiento y paso de los mensajes C2V y se usa una segunda memoria de aristas para el almacenamiento y el paso de los mensajes V2C. En tales realizaciones, pueden emplearse, y a menudo se emplean, múltiples unidades de procesamiento de nodos, por ejemplo, una para realizar el procesamiento de nodos de restricciones y otra para realizar el procesamiento de nodos de variables. Como se tratará más adelante, tales realizaciones permiten efectuar operaciones de procesamiento de variables y restricciones en paralelo escribiéndose los mensajes resultantes dentro de cada una de las dos memorias de mensajes para usar durante la siguiente iteración del proceso de decodificación.

Ahora presentaremos un simple ejemplo de un pequeño gráfico de LDPC y su representación que se utilizará posteriormente en la explicación de la invención. La discusión del gráfico de LDPC se seguirá por una descripción de un decodificador LDPC que puede usarse para decodificar el gráfico pequeño.

La Fig. 6 ilustra un código LDPC simple irregular en la forma de un gráfico 600. El código es de longitud 5 como se indica por los 5 nodos de variables de V_1 hasta V_5 602. Cuatro nodos de comprobación de C_1 a C_4 606 están acoplados a los nodos de variables 602 por un total de 12 aristas 604 sobre las cuales pueden pasarse los mensajes.

La Fig. 7 ilustra el código LDPC mostrado en la Fig. 6, usando las matrices 702, 704, en la forma de matrices de comprobación de paridad. Como se ha tratado anteriormente, las aristas se representan en la matriz de permutación H 702 usando unos. El bit x_i está asociado con los nodos de variables V_i. Las matrices 706 y 708 muestran los unos en la matriz H, correspondientes a las aristas en el gráfico, indexadas de acuerdo con el orden de encajes de variables y el orden de encajes de restricciones, respectivamente.

Con fines de explicación, las 12 aristas se numerarán desde el lado de los nodos de variables, es decir, de acuerdo con sus encajes de variables, Las conexiones establecidas por las aristas entre los nodos de variables 602 y los nodos de comprobación 606 puede verse en la Fig. 6. Para fines de la discusión, las aristas conectadas a la variable V_1 que la conectan con sus nodos de comprobación C_1, C_2 y C_3 se asignan las etiquetas 1, 2, 3, correspondientes a la numeración de encajes de variables. El nodo de variable V_2 está conectado a los nodos de comprobación C_1, C_3 y C_4 por las aristas 4, 5, y 6 respectivamente. El nodo de variable V_3 está acoplado a los nodos de comprobación C_1 y C_4 por las aristas 7 y 8, respectivamente. Además, el nodo de variable V_4 está acoplado a los nodos de comprobación C_2 y C_4 por las aristas 9 y 10, respectivamente, mientras que el nodo de variable V_5 está acoplado a los nodos de comprobación C_2 y C_3 por las aristas 11 y 12, respectivamente. Esta indexación corresponde con la matriz 706 de la Figura 7, es decir el orden de encajes de variables.

La Figura 8 ilustra las relaciones entre las 12 aristas de la Fig. 6, como se numeran desde el lado de los nodos de variables, en relación con los nodos de variables y de comprobación a los cuales están conectados. La fila 802 muestra los 5 nodos de variables V_1 hasta V_5. Debajo de las variables 802 se muestran las aristas 1 hasta 12, 804

correspondientes a los encajes asociados que están conectados a un nodo de variable particular. Obsérvese que como las aristas se ordenan desde el lado de los nodos de variables, en la fila 804 aparecen en el orden de 1 a 12. Asumamos que los mensajes se almacenan en memoria en el orden indicado en la fila 804.

Durante el procesamiento de nodos de variables, los 12 mensajes de arista en la memoria se acceden en secuencia, por ejemplo, en el orden mostrado en 804. De este modo, durante el procesamiento de los nodos de variables, los mensajes pueden leerse de forma simple en orden y suministrarse a una unidad de procesamiento.

La fila 806 ilustra los cuatro nodos de restricciones de C1 hasta C4 presentes en el código de las Fig. 6 y 7. Obsérvese que las aristas están reordenadas en la fila 804' para reflejar el orden en el cual están conectados a los nodos de restricciones, pero la indexación indicada es la inducida desde el lado de los nodos de variables. Por consiguiente, asumiendo que los mensajes de las aristas están almacenados en orden desde el lado de los nodos de variables, cuando se realiza el procesamiento de nodos de restricciones los mensajes se leerían en el orden ilustrado en la fila 804'. Esto es, durante el procesamiento de nodos de restricciones los mensajes se leerían de la memoria en el orden 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10. Puede usarse un módulo de ordenamiento de mensajes para sacar la secuencia correcta de la información de acceso de los mensajes de arista, por ejemplo, las localizaciones de memoria, para leer datos de la memoria o escribir datos a la memoria durante las operaciones de procesamiento de nodos de variables y de comprobación.

Un decodificador serie LDPC 900 que realiza operaciones de procesamiento de mensajes de forma secuencial, una arista cada vez, se tratará ahora en relación con la Fig. 9 y se tratará la decodificación usando el código de ejemplo mostrado en la Fig. 6. El decodificador LDPC 900 comprende un módulo de control del decodificador 902, un módulo de ordenamiento de mensajes (memoria de permutación de encajes) 904, una memoria de grados de nodos 910, una memoria de aristas 906, un procesador de nodos 908, una memoria intermedia de salida 916, una memoria de decisiones hardware 912 y un verificador de comprobación de paridad 914.

La memoria de aristas 906 incluye las localizaciones de la memoria de los bits L K correspondiendo cada una de las localizaciones de bit K a una arista y donde L es el número total de aristas en el gráfico de LDPC que se están usando y K es el número de bits por mensaje intercambiados a lo largo de una arista. Para concretar, asumimos que los mensajes se almacenan en orden de acuerdo con el ordenamiento de aristas inducido por los encajes de variables. De este modo, para el gráfico de ejemplo 600 los mensajes correspondientes a las aristas 1, 2, …, 12 se almacenan en el orden indicado. La memoria de decisiones hardware 912 incluye L localizaciones de memoria de 1 bit, correspondiendo cada una de las localizaciones de 1 bit a una arista. Esta memoria almacena decisiones hardware transmitidas por los nodos de variables a lo largo de cada una de sus aristas de modo que las restricciones de comprobación de paridad pueden verificarse. El verificador de comprobación de paridad 914 recibe las decisiones de bit hardware a medida que el procesador de los nodos de comprobación recibe mensajes. Las comprobaciones de paridad se verifican en el verificador de comprobación de paridad y, en el caso de que se satisfagan todas las comprobaciones, transmite una señal de convergencia al módulo de control del decodificador 902.

El módulo de ordenamiento de mensajes 904 puede implementarse como un mapa de permutaciones o una tabla de búsqueda que incluye información que describe el ordenamiento de mensajes en la memoria de aristas como se ve desde el lado de los nodos de variables o como se ve desde el lado de los nodos de restricciones. De este modo, para nuestro gráfico de ejemplo 600, la secuencia 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10 que especifica el orden de las aristas como se ve desde el lado de las restricciones se almacenaría, efectivamente en el módulo de ordenamiento de mensajes. Esta secuencia se usa para ordenar los mensajes para el procesamiento de los nodos de restricciones y para ordenar las decisiones hardware leídas de la Memoria de Decisiones Hardware 912 para el procesamiento por el verificador de comprobación de paridad 914.

En el decodificador de la Fig. 9, los mensajes correspondientes a una arista se sobrescriben después de que se procesan por un procesador de nodos. De este modo, la memoria de aristas alternará entre el almacenamiento de mensajes V2C y el almacenamiento de mensajes C2V. La verificación de decisión hardware se produce durante el procesamiento de los nodos de restricciones, por ejemplo a medida que se leen los mensajes V2C de la memoria de mensajes de aristas 906.

La unidad de control del decodificador 902 es responsable de alternar la operación del decodificador entre los modos de operación del procesamiento de nodos de variables y de comprobación, para determinar cuándo debería pararse el proceso de decodificación iterativa, por ejemplo debido a la recepción de una señal de convergencia o búsqueda de una cuenta de iteración permitida máxima, para suministrar y controlar el suministro de la información de grado a la unidad de procesamiento de nodos y el verificador de comprobación de paridad, y para controlar el suministro de un índice de arista al Módulo de Ordenamiento de Mensajes 904. Durante el funcionamiento, el módulo de control del decodificador 902 transmite un índice de aristas al módulo de ordenamiento de mensajes 904. El valor del índice de arista, se aumenta en el tiempo para una secuencia a través de todas las aristas en el gráfico. Un índice de arista diferente, por ejemplo único se usa para cada una de las aristas en un gráfico que se está implementando. En respuesta a cada uno de los índices de arista recibidos, el módulo de ordenamiento de mensajes sacará un

identificador de arista, por ejemplo, la información de dirección de la memoria de aristas, seleccionando de este modo la localización de la memoria de arista que se accederá, por ejemplo, leyendo o escribiendo en cualquier instante determinado. Asumiendo el ordenamiento de los encajes de variables, el módulo de ordenamiento de mensajes 904 causará que los mensajes se lean o se escriban de nuevo en orden secuencial durante el procesamiento de los nodos de variables y causará que los mensajes se lean y se escriban de nuevo en el orden correspondiente al ordenamiento de encajes de restricciones durante el procesamiento de los nodos de restricciones. De este modo, en nuestro ejemplo anterior, los mensajes se leerán y se escribirán de nuevo en el orden 1, 2, 3, …, 12 durante el procesamiento de los nodos de variables y, al mismo tiempo, se escribirán las decisiones hardware dentro de la memoria de decisiones hardware 912 en el orden 1, 2, 3, …, 12. Durante el procesamiento de nodos de restricciones los mensajes se leerán y escribirán de nuevo en el orden 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10 y, al mismo tiempo, el módulo de ordenamiento de mensajes 904 causará que los bits de decisión hardware se lean de la memoria de decisión hardware 912 en el orden 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10.

A medida que se leen los mensajes de la memoria de aristas en respuesta al identificador de arista recibido del módulo de control del decodificador 902, se suministran al procesador de nodos 908. El procesador de nodos 908 realiza la operación de procesamiento de nodos de restricciones o de variables apropiada, dependiendo del modo de operación, usando por lo tanto los mensajes recibidos para generar mensajes actualizados correspondientes al nodo particular que se está implementando en cualquier instante determinado. Los mensajes actualizados resultantes se escriben a continuación de nuevo en la memoria de aristas sobrescribiendo los mensajes que se acaban de leer de memoria. Los mensajes enviados a un nodo particular llegan al procesador de nodos como bloques contiguos, es decir, uno tras otro. El módulo de control del decodificador 902 señala la delineación de nodos al procesador de nodos, por ejemplo indicando el último mensaje correspondiente a un nodo proporcionando por lo tanto la información de grado del nodo. En el caso del gráfico de ejemplo 600, los grados de los nodos de variables se especificarían, por ejemplo como la secuencia (3, 3, 2, 2, 2) y se especificarían los grados de los nodos de restricciones, por ejemplo, como la secuencia (3, 3, 3, 3). Esta información puede almacenarse en una memoria de grados de nodos 910 que a continuación se leería por el módulo de control del decodificador 902 a medida que itera sobre los índices de arista. Como alternativa, la información de grado puede programarse dentro de cada una de las unidades de procesamiento de nodos. Esto puede ser preferible, por ejemplo cuando se sabe con antelación que los grados de los nodos serán uniformes, es decir el gráfico será regular.

El verificador de comprobación de la paridad 914 funciona en gran parte del mismo modo que un procesador de nodos de comprobación excepto que los mensajes entrantes son simples bits, no se calcula ningún mensaje saliente, y el cálculo interno es más simple.

Durante el funcionamiento del modo de nodos de variables, los cálculos de los nodos de variables, se realizarán un nodo cada vez por la unidad de procesamiento de nodos hasta que se haya completado el procesamiento, por ejemplo la actualización del mensajes y las operaciones de generación de valor de la salida software asociados con cada uno de los nodos de variables. Los mensajes se suministran al procesador de nodos 908 en el orden del lado de los nodos de variables de modo que todos los mensajes correspondientes a un nodo llegan en secuencia al procesador de nodos 908. Una vez completada la iteración del procesamiento de los nodos de variables, el módulo de control del decodificador 902 causa que el decodificador 900 conmute al modo de nodos de restricciones de la operación de procesamiento. En respuesta al cambio de la señal de control C/V (Restricción/Variable), la unidad de procesamiento de nodos 908 conmuta desde el modo de procesamiento de nodos variables al modo de procesamiento de nodos de restricciones. Además el módulo de ordenamiento de mensajes 904 conmuta a un modo de procesamiento en el que los identificadores de mensajes se suministrarán a la memoria de aristas en el orden de los encajes de las restricciones. Una o más señales de control enviadas sobre la línea de control C/V pueden usarse para controlar el conmutador entre los modos de operación del procesamiento de nodos de restricciones y de variables.

A medida que el circuito de control del decodificador 902 controla el decodificador para realizar el procesamiento de nodos de restricciones en una secuencia de nodos de restricciones, un nodo cada vez, los mensajes almacenados en la memoria de aristas se actualizarán de nuevo una vez, esta vez por los mensajes C2V generados por el procesamiento de nodos de restricciones. Cuando el procesamiento asociado con el conjunto total de los nodos de restricciones se ha completado, el circuito de control del decodificador 902 conmutará de nuevo al modo de nodos de variables de la operación de procesamiento. De este modo, el decodificador 900 alterna entre el procesamiento de nodos de variables y de nodos de restricciones. Como se ha descrito, el procesamiento se realiza de forma secuencial, un nodo cada vez, hasta que el circuito de control del decodificador 902 determina que la operación de decodificación se ha completado.

El sistema de decodificación LDPC escalar o secuencial ilustrado en la Fig. 9 puede implementarse usando relativamente poco hardware. Además tiende por si mismo bien a la implementación de software. Desafortunadamente, la naturaleza secuencial del procesamiento realizado tiende a dar como resultado una implementación del decodificador relativamente lenta. Por consiguiente, aunque la arquitectura escalar mostrada en la Fig. 9 tiene algunos atributos notables, tiende a ser inadecuado para aplicaciones de ancho de banda elevado tales como las comunicaciones ópticas o almacenamiento de datos donde es deseable una decodificación de alta

velocidad y el uso de largas palabras de código.

Antes de presentar los decodificadores para la decodificación de grandes gráficos LDPC convertidos a vectores, trataremos conceptos generales y técnicas relativas a las características de la conversión de los gráficos a vectores de la presente invención. La discusión de la conversión a vectores se seguirá por una presentación de decodificadores LDPC de ejemplo, convertidos a vectores que realiza la presente invención.

Con el propósito de obtener el entendimiento de gráficos LDPC convertidos a vectores, consideramos un código LDPC 'pequeño' con la matriz de comprobación de paridad *H*. El pequeño gráfico en el contexto de un mayor gráfico convertido a vectores, se denominará como *gráficos proyectados*. Sea Ψ un subconjunto de matrices de permutación de *Z* x *Z*. Asumimos que las inversas de las permutaciones en Ψ también están en Ψ. Dado el pequeño gráfico proyectado, podemos formar un gráfico LDPC Z veces mayor reemplazando cada uno de los elementos de *H* con una matriz de *Z* x *Z*. Los elementos 0 de H se reemplazan con la matriz cero, denotada por 0. Los elementos 1 de *H* se reemplazan cada uno con una matriz de Ψ. De este modo 'elevamos' el grado de LDPC a otro Z veces mayor. La complejidad de la representación comprende aproximadamente, el número de bits requerido para especificar las matrices de permutación, $|E_H|$ log |Ψ| más la complejidad requerida para representar *H*, donde $|E_H|$ denota el número de unos en *H* y |Ψ| denota el número de permutaciones distintas en Ψ. Por ejemplo, si Ψ es el espacio de permutaciones cíclicas entonces |Ψ| = *Z*. En la práctica podríamos tener, por ejemplo *Z* = 16 para n = 1000.

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} \sigma_1 & \mathbf{0} & \sigma_7 & \sigma_9 & \sigma_{11} & \mathbf{0} & \mathbf{0} \\ \sigma_2 & \sigma_4 & \sigma_8 & \mathbf{0} & \mathbf{0} & \sigma_{13} & \mathbf{0} \\ \sigma_3 & \sigma_5 & \mathbf{0} & \sigma_{10} & \mathbf{0} & \mathbf{0} & \sigma_{15} \\ \mathbf{0} & \sigma_6 & \mathbf{0} & \mathbf{0} & \sigma_{12} & \sigma_{14} & \sigma_{16} \end{bmatrix}$$

Ejemplo: Elevando una pequeña matriz de comprobación de paridad, σ_i para *i* = 1, …16 son elementos de Ψ mostrados en este punto indexados en el orden de encajes de variables proyectados.

El subconjunto Ψ puede elegirse en general usando diversos criterios. Una de las motivaciones principales para la estructura anterior es simplificar la implementación hardware de los decodificadores. Por lo tanto, puede ser beneficioso restringir Ψ a permutaciones que pueden implementarse de forma eficaz en hardware, por ejemplo en una red de conmutación.

Las topologías de las redes de conmutación en paralelo es un tema bien estudiado en conexión con las arquitecturas de multiprocesadores y conmutadores de conmutación de alta velocidad. Un ejemplo práctico de arquitectura adecuada para el subconjunto de permutaciones Ψ es una clase de redes de conmutación multicapa incluyendo, por ejemplo, las redes omega (barajado perfecto) / delta, redes de desplazamiento de registros, etc. Estas redes ofrecen una complejidad de implementación razonable y riqueza suficiente para el subconjunto Ψ. Adicionalmente las redes de conmutación multicapa se escalan bien, por ejemplo, su complejidad se presenta como N log N donde N es el número de entradas a la red, lo que les hace especialmente adecuadas para los decodificadores LDPC en paralelo de forma masiva. Como alternativa, en los decodificadores de la presente invención con niveles relativamente bajos de paralelismo y pequeñas Z pueden implementar el subconjunto Ψ de permutaciones en una única capa.

Un gráfico LDPC se dice que tiene "aristas múltiples" si cualquier par de nodos está conectado por más de una arista. Una arista *múltiple* es el conjunto de aristas que conectan un par de nodos que están conectados por más de una arista. Aunque generalmente no es deseable para un gráfico LDPC tener aristas múltiples, en muchas ocasiones puede ser necesario en la construcción de gráficos convertidos a vectores, en los que el gráfico proyectado posee múltiples aristas. Puede extenderse la notación de una matriz de comprobación de paridad para permitir entradas de matrices para denotar el *número* de aristas que conectan el par de nodos asociados. La definición de la palabra de código es aún la misma: el código es el conjunto de 0,1 1 vectores x que satisfacen que Hx = 0 en módulo 2. Cuando se convierte a vectores un gráfico proyectado con aristas múltiples, de acuerdo con la invención, cada una de las aristas dentro de una arista múltiple se reemplaza con una matriz de permutación desde T y estas matrices se añaden para obtener la matriz de comprobación de paridad extendida del código total. De este modo, un valor de j>1 en la matriz de comprobación de paridad H del gráfico proyectado se 'elevará' a una suma $\sigma_k + \sigma_{k+1} + \ldots + \sigma_{k+j-1}$, de las matrices de permutación desde Ψ. Usualmente, se elegirán los elementos de la suma de modo que cada una de las entradas $\sigma_k + \sigma_{k+i} + \ldots + \sigma_{k+j-1}$ sea o cero o uno, es decir el grafico total no tiene aristas múltiples.

La elevación descrita anteriormente parece tener una limitación. Bajo la construcción anterior tanto la longitud de código como la longitud de la unidad de datos codificados deben ser múltiplos de Z. Sin embargo esta limitación aparente es fácilmente superable. Supongamos que la unidad de datos a codificar tiene una longitud AZ + B donde A es un número entero positivo y B está entre 1 y Z inclusive, y la longitud de código deseada es CZ + D donde C es

un número entero positivo y D está entre 0 y Z-1 inclusive. Sea E el número entero positivo más pequeño tal que EZ ≥ CZ + D + (Z – B). Se puede diseñar un gráfico elevado que codifica la unidad de datos de longitud (A + 1) Z para producir una palabra de código de longitud EZ de modo que la unidad de datos aparece como parte de la palabra de código, y usa esto para producir los parámetros de código deseados como sigue. Dada una unidad de datos de longitud AZ + B se concatenan Z – B ceros para producir una unidad de datos de longitud (A + 1) Z. Esa unidad de datos se codifica para producir una palabra de código de longitud EZ. Los Z – B ceros no se transmiten. De los otros EZ – (Z – B) bits en la palabra de código se seleccionan EZ – CZ – D – (Z – B) bits y se perforan, obsérvese que el número de bits de perforación está entre 0 y Z-1 inclusive. Estos bits no se transmitirán, de modo que el número real de bits transmitidos es de EZ – (Z – B) – (EZ – CZ – D – (Z – B)) = CZ +D, que es la longitud de código deseada. El receptor que tiene conocimiento de antemano acerca de los ceros adicionales y los bits perforados sustituye los bits software por los bits perforados indicando borrado, y sustituye los bits software por los bits cero conocidos indicando un valor de cero con la mayor fiabilidad posible. La palabra extendida recibida, de longitud EZ puede ahora decodificarse para recuperar la unidad de datos original. En la práctica usualmente se hacen estos ajustes perforando bits de sólo un nodo de vector y declarando bits conocidos de sólo un nodo de vector.

Ahora nos dirigimos a diversas implementaciones de decodificadores que usan la técnica tratada anteriormente de conversión a vectores de los gráficos LDPC.

Como se ha tratado anteriormente la decodificación de paso de mensajes de los códigos LDPC involucra el paso de mensajes a lo largo de las aristas del gráfico que representa el código y la realización de los cálculos en base a esos mensajes en los nodos del gráfico, es decir, los nodos de variables y de restricciones.

Dado un gráfico LDPC convertido a vectores se pueden convertir a vectores los procesos de decodificación como sigue. El decodificador funciona como si se estuviesen decodificando Z copias del código LDPC proyectado de forma síncrona y en paralelo. El control del proceso de decodificación corresponde al gráfico LDPC proyectado y puede compartirse a través de las Z copias. De este modo, describimos el decodificador como si operase sobre mensajes de vectores atravesando las aristas de vectores y recibiéndose por los nodos de vectores, teniendo cada uno de los vectores Z elementos. Los encajes también se convierten en vectores. En particular, un procesador de nodos de vectores podría comprender Z procesadores de Z nodos en paralelo y, cuando se suministra un vector de mensajes, (m_1, …, m_z) al procesador de nodos de vectores, se suministra el mensaje m_i al procesador de orden i. De este modo, no se produce ningún encaminamiento ni reordenamiento de mensajes dentro del procesador de nodos de vectores, es decir, el mensaje de vectores se alinea con el vector de los procesadores de un modo fijo.

Una desviación de la ejecución en paralelo puramente disjunta de los Z gráficos proyectados es que los mensajes se reordenan dentro de un mensaje de vector durante el proceso de paso de mensajes. Nos referimos a esta operación de reordenamiento como una *rotación*. La rotación implementa las operaciones de permutación definidas por Ψ. Debido a las rotaciones, las trayectorias de procesamiento de las Z copias de la mezcla de gráficos proyectados enlazan los mismos por lo tanto para formar un gráfico único más grande. La información de control que especifica las rotaciones se necesita además de la información de control requerida para el gráfico proyectado. Afortunadamente, la información de control de la rotación puede especificarse usando relativamente poca memoria.

Aunque pueden usarse diversas permutaciones para las rotaciones de acuerdo con la presente invención, el uso de permutaciones cíclicas es particularmente interesante debido a la facilidad con la que tales permutaciones pueden implementarse. Por simplicidad asumiremos que Ψ comprende el grupo de permutaciones cíclicas. En este caso, nuestros grandes gráficos LDPC están restringidos a tener una estructura casi cíclica. Con fines de este ejemplo, sea *N* el número de nodos de variables en el gráfico y sea *M* el número de nodos de restricciones en el gráfico. En primer lugar asumimos que tanto *N* como *M* son múltiplos de *Z, N = nZ* y *M = mZ* donde *Z* denota el orden del ciclo.

Veamos que los nodos están doblemente indexados. De este modo, el nodo de variable $v_{i,j}$ es el nodo de variable de orden *j* de la copia de orden *i* del gráfico proyectado. Como Ψ es el grupo de permutaciones cíclicas, el nodo de variable $v_{i,j}$ está conectado a un nodo de restricción $C_{a,b}$ si y sólo si el nodo de variable $v_{i+k \bmod z,\ j}$ está conectado al nodo de restricción $c_{a+k \bmod Z,\ b}$ para k = 1, …, Z.

Las técnicas de la presente invención para la representación de un gran gráfico que usa una representación de gráfico mucho más pequeña y la información de rotación se explicará ahora además en referencia a las Fig. 10 hasta 16 que se refieren a la conversión a vectores del gráfico 600. Las técnicas descritas con referencia a estas figuras pueden aplicarse a gráficos LDPC mucho más grandes.

De acuerdo con la presente invención, puede generarse un gráfico mayor replicando, es decir, implementado múltiples copias, de un gráfico pequeño mostrado en la Fig. 6 y realizando a continuación operaciones de rotación para interconectar las diversas copias del gráfico replicado. Nos referimos al gráfico pequeño dentro de la estructura del gráfico más grande como el gráfico proyectado.

La Fig. 10 es un gráfico 1000 que ilustra el resultado de realización de 3 copias en paralelo del pequeño gráfico ilustrado en la Fig. 6. Los nodos de variables 602', 602'' y 602''' corresponden a los gráficos primero hasta el tercero, respectivamente, resultando de la realización de las tres copias del gráfico de la Fig. 6. Además los nodos de

comprobaciones 606', 606'' y 606''' corresponden a los gráficos primero hasta el tercero, respectivamente, resultando de la realización de las tres copias. Obsérvese que no hay aristas de nodos que conectan nodos de uno de los tres gráficos con los nodos de cualquier otro de los tres gráficos. Por consiguiente, este proceso de copias, que "eleva" el gráfico básico por un factor de 3, da como resultado tres gráficos idénticos disjuntos.

La Fig. 11 ilustra el resultado del proceso de copiado tratado anteriormente usando matrices 1102 y 1104. Obsérvese que para hacer las tres copias del gráfico original cada uno de los elementos distintos de cero en la matriz 702 se reemplazan con una matriz de identidad de 3x3. De este modo, cada uno en la matriz 702 se reemplaza por una matriz de 3x3 que tiene unos a lo largo de la diagonal y ceros en el resto para producir la matriz 1102. Obsérvese que la matriz 1102 tiene 3 veces el número de aristas que tiene la matriz 702, 12 aristas para cada una de las tres copias del gráfico básico mostrado en la Fig. 6. En este punto, la variable x_{ij} corresponde con el nodo de variables V_{ij}.

La Fig. 12 muestra las relaciones entre las 36 aristas (3 x 12), los 15 nodos de variables (3 x 5) y los 12 nodos de restricciones (3 x 4) que constituyen el gráfico 1000. Como en el caso de la Fig. 8, las aristas se numeran desde el lado de los nodos de variables.

Para propósitos de anotación, el primer número usado para identificar un nodo, restricción o arista indica la copia del gráfico a la cual pertenece la arista, por ejemplo, la primera, segunda o tercera copia del gráfico. El segundo número se usa para identificar el número de elemento dentro de la copia especificada particular del gráfico básico.

Por ejemplo, en la fila 1202' se usa el valor (1, 2) para indicar la arista 2 de la primera copia del gráfico mientras que en la fila 1202'' se usa (2,2) para indicar la arista 2 de la segunda copia del gráfico.

Obsérvese que las filas de aristas 1202', 1202'', 1202''' son simplemente copias de la fila 804 que representan tres copias de la fila de las aristas 804, mostradas en la Fig. 8 que se refieren a los nodos de variables. De forma similar, las filas de aristas 1204', 1204'' y 124''' representan tres copias de la fila de aristas 804' mostrada en la Fig. 8 que se refieren a los nodos de restricciones.

Tratemos brevemente cómo modificar el decodificador 900 de la Fig. 9 para decodificar los Z=3 gráficos paralelos ahora definidos. El procesador de nodos 908 se convertirá en un procesador de nodos de vectores, capaz de procesar 3 nodos idénticos simultáneamente en paralelo. Todas las salidas desde el procesador de nodos 908 se convertirán a vectores, transportando por lo tanto 3 veces los datos transportados anteriormente. La memoria de decisión hardware 912 y la memoria de mensajes de aristas 906 se harán 3 veces más anchos, cada uno capaz de escribir o leer 3 unidades (bits o mensajes de K bits respectivamente) en paralelo usando la dirección de una única instrucción SIMD. Las salidas desde estas memorias serán ahora vectores, 3 veces más anchos que antes. El verificador de comprobación de paridad 914 y la memoria intermedia de salida 916 también se convertirán adecuadamente a vectores con todos los procesamientos puestos en paralelo de forma adecuada.

Consideremos ahora la introducción de las rotaciones en nuestro ejemplo. Esto puede ilustrarse reemplazando cada una de las matrices identidad de 3x3 mostradas en la Fig. 11 con matrices de permutación cíclica de 3x3 como se muestra en la Fig. 13. Obsérvese que hay tres posibilidades para la matriz de permutación cíclica utilizada en la Fig. 13. Es posible indicar la matriz de permutación particular a sustituir por una matriz identidad indicando si la matriz de permutación tiene un "1" localizado en la primera, segunda o tercera posición en la primera fila de la matriz de permutación. Por ejemplo, en el caso de la matriz 1302, comenzando en la parte superior izquierda y procediendo a la esquina inferior derecha (orden de encajes de las restricciones de vector) las rotaciones podrían especificarse por la secuencia (2, 2, 3, 3 1, 1, 1, 3, 2, 1, 2, 3).

La Fig. 14 ilustra el efecto de la realización de la permutación cíclica (rotación) sobre el lado de nodos de restricciones. Como la permutación se realiza desde el lado de los nodos de restricciones, la relación entre las aristas, por ejemplo, el ordenamiento desde el lado de los nodos de variables permanece sin cambiar como se muestra en las filas 1402', 1402'' y 1402'''. Desde el lado de las restricciones, sin embargo, la permutación da como resultado encajes dentro de una columna, por ejemplo las aristas dentro de una arista de vector específico, reordenándose como se muestra en las filas 1404', 1404'', 1404'''. Esto produce interconexiones entre nodos correspondientes a diferentes copias del gráfico proyectado.

Consideramos, por ejemplo, la columna 1 de filas 1404 en relación con la columna 1 de filas 1104 de la Fig. 11. Obsérvese que como resultado de operación de permutación de aristas de vectores, el nodo de restricción $C_{1,1}$ está ahora conectado a la arista (2, 1) en lugar de la arista (1, 1), el nodo de restricción $C_{2,1}$ está acoplado a la arista (3,1) en lugar de a la arista (2,1) y el nodo de restricción $C_{3,1}$ está acoplado a la arista (1,1) en lugar de la arista (3,1).

Tratamos anteriormente cómo convertir a vectores el decodificador 900, o decodificar Z copias en paralelo del gráfico proyectado. Introduciendo conmutaciones dentro de las trayectorias del mensaje para realizar rotaciones, decodificamos el código LDPC definido en la Fig. 13.

La Figura 15 ilustra un decodificador que incorpora diversas características de la presente invención. El

decodificador 1500 convierte a vectores totalmente, con rotaciones, el decodificador 600. Obsérvese que la figura indica Z=4 mientras que nuestro ejemplo tiene Z=3, en general podemos tener cualquier Z>1 pero en la práctica a menudo serán preferible Z valores en la forma de 2^k siendo k un número entero. Las similitudes con el decodificador 600 son evidentes. En particular el módulo de control del decodificador 1502 y la memoria de grados de los nodos 1510 funcionan de la misma forma o similar que sus homólogos respectivos 902 y 910 en el decodificador 900. Por ejemplo, para decodificar el código LDPC definido en las Fig. 13 y 14 el funcionamiento de estos componentes sería exactamente el mismo que el de sus homólogos en el decodificador 900 cuando se decodifica el gráfico de ejemplo 600. La memoria de mensajes de aristas 1506 y la memoria de decisión hardware 1512 son versiones convertidas a vectores de sus homólogas 906 y 912 en el decodificador 900. Mientras que en el decodificador 900 las memorias almacenaban unidades únicas (bits o mensajes de k bits) las memorias correspondientes en el decodificador 1500 almacenan conjuntos, es decir vectores, mensajes, dando como resultado por ejemplo mensajes de ZxK bits almacenados. Estos vectores se escriben o se leen como unidades únicas usando instrucciones SIMD. De este modo los identificadores de mensajes enviados a estos módulos desde el módulo de ordenamiento de mensajes 1504 son equivalentes o similares a los del decodificador 900. El módulo de ordenamiento de mensajes 1504 tiene el papel adicional, más allá del que tenía su homólogo 904 en el decodificador 900, de almacenar y proporcionar la información de permutación, por ejemplo, de rotación. Recordemos que en el ejemplo de decodificación 600, el decodificador 900 almacenaba en su módulo de ordenamiento de mensajes 904 la secuencia de aristas (1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10). Consideramos usar el decodificador 1500, para decodificar el código de las Fig. 13 y 14. El módulo de ordenamiento de mensajes 1504 almacenaría la misma secuencia anterior para el acceso de vectores de mensajes durante el procesamiento de los nodos de restricción, y también almacenaría la secuencia (2, 2, 3, 3, 1, 1, 1, 3, 2, 1, 2, 3) que describe las rotaciones asociadas con la misma secuencia de mensajes de vectores. Esta secuencia sirve como base para generar la señal de decadencia que se usa por el módulo de ordenamiento de mensajes 1504 para causar que los conmutadores 1520 y 1522 giren los mensajes de vectores y los bits de decisión hardware de vector respectivamente. (Obsérvese que los bits de decisión hardware se proporcionan sólo durante el modo de procesamiento de nodos de variables). El verificador de comprobación de paridad de vectores 1514 es una versión de vectores de su homólogo 914 en el decodificador 900. Obsérvese que la señal de convergencia es un escalar, como antes. La memoria intermedia de salida 1516 sirve para el mismo propósito que la memoria intermedia 916, pero los datos de salida se escriben como vectores. El procesador de nodos de vectores 1508, es por ejemplo como Z procesadores de nodos, cada uno como en 908, en paralelo. Estos nodos compartirían las señales de grado y la señal de control C/V del módulo de control del decodificador 1502.

Para facilitar la capacidad de sacar decisiones del decodificador bien software o hardware generadas por la unidad de procesamiento de variables se suministran a una entrada de decisión software de la memoria intermedia 1516. De este modo, en cualquier momento anterior a la terminación de la decodificación, pueden obtenerse decisiones software a partir de la salida de la memoria intermedia 1516.

Consideramos adicionalmente cómo funcionaría el decodificador 1500 decodificando el ejemplo de las Fig. 13 y 14. Inicialmente la memoria de mensajes de aristas 1506 se puebla con ceros. El módulo de control del decodificador 1502 en primer lugar bascula al modo de procesamiento de nodos de variables. Se leen los vectores de la memoria de mensajes de aristas 1506 (todos ceros en este punto) en orden y se suministran al procesador de nodos de vectores 1508 para el procesamiento de los nodos de variables. El procesador de nodos de vectores 1508 a continuación saca los valores recibidos solos, a lo largo de cada arista desde un nodo de variable, usaremos y para denotar estos primeros mensajes para indicar esto. De este modo, los vectores salientes serían ($y_{1,i}$, $y_{2,i}$, $y_{3,i}$) para i= 1, …, 12 en orden creciente. La señal de decadencia se usa para controlar el reordenamiento de los mensajes de control realizado por los circuitos de conmutación 1520, 1522. La señal de decadencia del módulo de ordenamiento de mensajes 1504 causará que los mensajes en los vectores roten para producir vectores procesados como sigue: ($y_{2,1}$, $y_{3,1}$, $y_{1,1}$), ($y_{3,2}$, $y_{1,2}$, $y_{2,2}$), ($y_{1,3}$, $y_{2,3}$, $y_{3,3}$), ($y_{2,4}$, $y_{3,4}$, $y_{1,4}$), ($y_{3,5}$, $y_{1,5}$, $y_{2,5}$), ($y_{1,6}$, $y_{2,6}$, $y_{3,6}$), ($y_{3,7}$, $y_{1,7}$, $y_{2,7}$), ($y_{2,8}$, $y_{3,8}$, $y_{1,8}$), ($y_{1,9}$,$y_{2,9}$, $y_{3,9}$), ($y_{3,10}$, $y_{1,10}$, $y_{2,10}$), ($y_{1,11}$, $y_{2,11}$, $y_{3,11}$), ($y_{2,12}$, $y_{3,12}$, $y_{1,12}$). Una vez que se han escrito los vectores procesados dentro de la memoria de aristas 1506, en el orden indicado, el módulo de control del decodificador 1502 basculará al modo de restricciones. Los mensajes del vector almacenados a continuación se leerán en orden (1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10). De este modo se presentarán al procesador de nodos de vectores 1508 en el orden ($y_{2,1}$, $y_{3,1}$, $y_{1,1}$), ($y_{2,4}$, $y_{3,4}$, $y_{1,4}$), ($y_{3,7}$, $y_{1,7}$, $y_{2,7}$), ($y_{3,2}$, $y_{1,2}$, $y_{2,2}$), ($y_{1,9}$,$y_{2,9}$, $y_{3,9}$), ($y_{1,11}$, $y_{2,11}$, $y_{3,11}$), ($y_{1,3}$, $y_{2,3}$, $y_{3,3}$), ($y_{3,5}$, $y_{1,5}$, $y_{2,5}$), ($y_{2,12}$, $y_{3,12}$, $y_{1,12}$) ($y_{1,6}$, $y_{2,6}$, $y_{3,6}$), ($y_{2,8}$, $y_{3,8}$, $_{1,8}$), ($y_{3,10}$, $y_{1,10}$, $y_{2,10}$). El procesador de nodos de vectores 1508 se implementa como tres procesadores de nodos (Z=3) en paralelo. El primer elemento (mensaje) de cada vector de mensajes (conjunto de mensajes) se suministra al primer procesador de nodos, el segundo mensaje se suministra al segundo procesador, y el tercer mensaje se suministra al tercer procesador, respectivamente. La señal de grado, que indica el grado del nodo actual que se está procesando, se suministra por la memoria de grados 1510 a los tres procesadores en paralelo del procesador de nodos de vectores 1508. En este punto la señal de grado indica que las restricciones son todas de grado 3 de modo que el primer procesador procesaría $y_{2,1}$, $y_{2,4}$, e $y_{3,7}$ para su primer nodo de restricción e $y_{3,2}$, $y_{1,9}$, e $y_{1,11}$ para su segundo nodo de restricción. De forma similar, el segundo procesador procesaría $y_{3,1}$, $y_{3,4}$, e $y_{1,7}$ para su primer nodo de restricción e $y_{1,2}$, $y_{2,9}$ e $y_{2,11}$ para su segundo.

Sea $m_{i,j}$ el mensaje saliente correspondiente al entrante $y_{i,j}$. A medida que los vectores emergen del procesador de nodos de vectores 1508, la señal de decadencia para el conmutador 1520 causará que los vectores se reordenen de modo que se invierte la rotación anterior, por lo tanto llegan a la memoria de aristas como ($m_{1,j}$, $m_{2,j}$, $m_{3,j}$), en el

orden j= 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10. Los mensajes se escriben de nuevo en la memoria de acuerdo con el orden del identificador de mensajes con los cuales se leyeron, de modo que después de escribirse aparecen en la memoria como ($m_{1,j}$, $m_{2,j}$, $m_{3,j}$) en el orden j= 1, …, 12. El módulo de reordenamiento de mensajes 1504 bascula ahora dentro del modo de procesamiento de nodos de variables en respuesta a la señal C/V suministrada por el módulo de control del decodificador 1502. Los vectores de mensajes se leen a continuación en orden j= 1, …, 12 y se suministran al procesador de nodos de vectores 1508 para el procesamiento de los nodos de variables. Esto completa una iteración.

Durante el procesamiento de los nodos de variables, el procesador de nodos de vectores 1508 también saca vectores decodificados software que se almacenan en la memoria intermedia de salida 1516. También saca decisiones hardware que se suministran al circuito de conmutación 1522. Los vectores de decisiones hardware de un bit sufren la misma operación de rotación que los vectores de mensajes en los instantes correspondientes. Los vectores de decisiones hardware rotados producidos por el circuito de conmutación 1522 se disponen a continuación en la memoria de decisiones hardware 1512 donde se almacenan. Como resultado de la aplicación de la misma rotación aplicada a los vectores de mensajes, las decisiones hardware pueden leerse en el mismo orden que se leen los mensajes de vectores durante el procesamiento de los nodos de restricciones. Durante el procesamiento de los nodos de restricciones, las decisiones hardware se suministran al verificador de comprobación de la paridad de vectores 1514 que realiza Z comprobaciones de paridad en paralelo. Si todas las comprobaciones de paridad se satisfacen, a continuación se genera la señal de convergencia, CON y se emite. En respuesta a la recepción de la señal de convergencia indicando la decodificación satisfactoria, el módulo de control del decodificador 1502 para el proceso de decodificación.

Resultará evidente que hay muchas variaciones para el decodificador 1500 que cada una de las realizaciones de la invención actual. Por ejemplo, el conmutador 1520 podría haberse situado en cambio a lo largo de la trayectoria de datos entre la memoria de mensajes de aristas 1506 y el procesador de nodos de vectores 1508. De forma similar, el conmutador 1522 podría haberse situado en cambio a lo largo de la trayectoria de datos entre la memoria de decisión hardware 1512 y el verificador de comprobación de paridad de vectores 1514. Tal sustitución involucraría también el ajuste apropiado de la temporización de la señal de decadencia. La memoria de decisión hardware 1512, el verificador de comprobación de paridad de vectores 1514 y las trayectorias de datos acompañantes no necesitan usarse y se eliminan para las realizaciones del decodificador que realizan un número fijo de iteraciones y por lo tanto no requieren detección de convergencia. Muchas variaciones adicionales serán evidentes para los especialistas en la técnica a la vista de la presente invención.

La Fig. 16 ilustra un decodificador 1600 que se implementa de acuerdo con otra realización de la presente invención. El decodificador 1600 incluye muchos elementos que son los mismos, o similares a los elementos del decodificador 1500. Por consiguiente, con el fin de la brevedad, tales elementos se identificarán usando los mismos números de referencia que los utilizados en la Fig. 15 y no se tratarán de nuevo en detalle. El decodificador 1600 es capaz de realizar tanto las operaciones de procesamiento de nodos de variables como de nodos de restricciones, por ejemplo, operaciones de actualización de mensajes, al mismo tiempo, es decir, simultáneamente e independientemente. En contraste con la Fig. 15, la implementación del decodificador que puede describirse como un decodificador de lado a lado debido al modo en que alterna entre las iteraciones de procesamiento de los nodos de variables y los nodos de restricciones, el decodificador 1600 puede describirse como un decodificador de iteración asíncrona ya que las operaciones de procesamiento de nodos de variables y nodos de restricciones puede realizarse independientemente, por ejemplo, simultáneamente.

El circuito decodificador 1600 incluye un módulo de control del decodificador 1602, un módulo de ordenamiento de mensajes 1604, un primer circuito de conmutación 1621, una memoria de mensajes de arista V2C 1606, un procesador de vectores de nodos de restricciones (por ejemplo, Z procesadores de nodos de restricciones en paralelo) 1609, un segundo circuito de conmutación 1620, la memoria de mensajes de aristas C2V 1607, un procesador de vectores de nodos de variables 1608 (por ejemplo, Z procesadores de nodos de variables en paralelo), una memoria de decisión hardware 1612, y un tercer conmutador 1622 acoplados juntos como se ilustra en la Fig. 16.

Diversas realizaciones de procesadores de nodos de restricciones individuales y procesadores de nodos de variables individuales, Z de los cuales pueden usarse en paralelo para implementar el procesador de vectores de nodos de restricciones 1609 y el procesador de nodos de variables 1608, respectivamente, se describen en la Solicitud de Patente Provisional de los Estados Unidos 60/328.469, titulada "Procesadores de Nodos Para Uso en los Decodificadores de Comprobación de Paridad", que se presento el 10 de Octubre de 2001. Los inventores de la presente solicitud de patente son también los inventores nombrados de la solicitud de patente provisional incorporada.

Para soportar la actualización independiente y/o en paralelo de mensajes de restricciones y variables en la realización de la Fig. 16 se usan memorias de mensajes de aristas 1606, 1607 y circuitos de conmutación 1620, 1621 separados para soportar las operaciones de procesamiento de nodos de restricciones y nodos de variables, respectivamente. Como en la realización de la Fig. 15, cada una de las memorias de mensajes 1606, 1607 son

capaces de almacenar L (ZxK bits) mensajes de vectores. Cada uno de los mensajes de vectores, por ejemplo, las columnas de Z mensajes de K bits, en las memorias 1606, 1607 pueden leerse o escribirse en una simple operación de lectura o escritura.

La memoria de mensajes de aristas V2C 1606 se usa para almacenar mensajes V2C y por lo tanto tiene una entrada de escritura acoplada a la salida del circuito de conmutación 1621 que recibe datos desde el procesador de vectores de nodos de variables 1608. La memoria de mensajes C2V 1607 se usa para almacenar mensajes de aristas C2V y por lo tanto tiene una entrada de escritura acoplada a la salida del procesador de vectores de nodos de restricciones 1609.

Los conmutadores 1620 y 1621 se usan para acoplar el procesador de vectores de nodos de variables 1608 a la entrada de la memoria de mensajes de aristas V2C y la salida de la memoria de mensajes de aristas C2V, respectivamente. En una realización particular se almacenan vectores de mensajes en el orden de los encajes de restricciones de vectores. Los vectores de mensajes se escriben dentro de la memoria de mensajes de aristas C2V 1607 y se leen de la memoria de mensajes de aristas V2C 1606 en el orden de los encajes de restricciones de vectores, es decir, linealmente, de este modo no se requiere ningún control externo (la salida de índices de aristas desde el módulo de control del decodificador 1602 pasa a través del módulo de ordenación de mensajes 1604 sin cambios a su salida de índice de aristas de restricciones). Los vectores de mensajes se leen de la memoria de mensajes de aristas C2V 1607 y se escriben dentro de la memoria de mensajes de aristas V2C 1606 en el orden de encajes de variables de vectores. El módulo de ordenamiento de mensajes 1604 genera la señal de índices de aristas de variables que indica este ordenamiento. Obsérvese que esta señal controla la lectura de la memoria de mensajes de aristas de C2V 1607 y se suministra a la memoria de mensajes de aristas V2C 106 después de retardarse. El retardo da cuenta del tiempo requerido para el procesamiento realizado por los conmutadores 1620 y 1621 y el procesador de nodos de variables de vectores 1608. Este retardo puede ser una función del grado de nodo que se está procesando, como se indica en la Fig. 16 por la señal de grado de los nodos de variables.

Para evitar los retardos de ejecución simultánea del procesamiento debidos a retardos variables se ordenan ambos nodos de restricciones y variables de modo que los nodos del mismo grado se procesan de una forma contigua. Puede conseguirse una reducción adicional de los retardos de ejecución simultánea que se producen en la frontera de grupos de nodos con diferentes grados clasificando los grupos de nodos por el grado de una forma monótona, por ejemplo, aumentando o disminuyendo el orden del grado. Para la simplicidad de implementación, las realizaciones 900, 1500 y 1600 asumen aumentar el orden de grado.

En la realización particular ilustrada en la Fig. 16 los vectores se almacenan en el orden de rotación de las restricciones de vectores. El conmutador 1620 gira los mensajes en cada uno de los vectores en una rotación variable a medida que cada uno de los mensajes de vectores C2V procede al procesador de vectores de nodos de variables 1608 y a continuación el conmutador 1621 aplica la rotación inversa al mensaje de vectores V2C saliente correspondiente a la misma arista de vectores. La señal de decadencia suministrada al conmutador 1620 se suministra al conmutador 1621 a través del circuito de inversión de rotación 1624 después de un retardo igualado al tiempo de procesamiento en el procesador de nodos de restricciones de vectores. Este retardo puede depender del grado de los nodos de restricciones, como se indica por la salida de la señal de grado de los nodos de restricciones por la memoria de grado 1610.

El decodificador 1600 incluye un módulo de control del decodificador 1602. El módulo de control del decodificador opera de forma similar al módulo de control tratado anteriormente 1502. Sin embargo, en 1602 no se genera ninguna señal de control de C/V. La función de generación del índice de aristas puede proporcionarse por un contador que circula a través de todo el conjunto de aristas de vectores antes de comenzar de nuevo.

Además de sacar las decisiones software, se generan decisiones hardware, una por arista, por cada una de las Z unidades de procesamiento de nodos de variables en el procesador de nodos de variables de vectores 1608 cada vez que se genera un mensaje V2C de vectores. Mientras que los mensajes de vectores se escriben dentro de la memoria de mensajes de aristas V2C 1606, las salidas de decisiones hardware de Z x 1 bit se escriben dentro de la memoria de salida de decisiones hardware 1612 después de girarse por el conmutador 1622. El conmutador 1622 y la memoria de decisión hardware 1612 operan bajo las mismas señales de control que el conmutador 1621 y la memoria de mensajes de aristas V2C 1606, respectivamente.

Los vectores girados resultantes de Z x 1 se suministran al verificador de comprobación de paridad de vectores 1614 que incluye Z verificadores de comprobación de paridad conectados en paralelo. El verificador 1614 determina si se satisfacen las comprobaciones de paridad y si todas se satisfacen entones se genera una señal de convergencia y se envía al módulo de control del decodificador. En respuesta a la recepción de la señal indicando la convergencia, el módulo de control del decodificador para la decodificación de la palabra de código recibida. En la realización 1600 la señal de detección de convergencia está disponible una iteración después de que se ha escrito una palabra de código en la memoria intermedia de salida ya que la verificación de restricciones para los datos desde la iteración N se hace durante la iteración N +1 y la señal de convergencia está disponible después de la terminación de la iteración N + 1.

En el decodificador 1600 que emplea diferente circuitería de detección de convergencia, la Fig. 17 ilustra una realización similar para el decodificador 1700, la verificación de restricciones se cumple "sobre la marcha" a medida que los valores de salida X de ZxK bits se escriben en la memoria intermedia de salida 1716. En este caso el bloque de memoria 1712 sigue el estado de las restricciones según se actualizan las restricciones de vectores por la salida de decisión hardware desde el procesador de nodos de variables 1708. Cada una de las localizaciones de memoria del estado de restricción corresponde a una restricción de comprobación de la paridad de la palabra de código. En la última actualización de cualquier localización de estado de restricción se verifican estos valores de comprobación de paridad. Si se satisfacen todas las verificaciones durante la iteración N a continuación se generará una señal de convergencia y la salida por el verificador de comprobación de paridad de vectores 1714 inmediatamente después de la iteración N. Esto calificará para módulo de control del decodificador 1702 que los datos en la memoria intermedia de salida 1716 son válidos. En la realización de la Fig. 17, el módulo de reordenamiento de mensajes 1704 genera una señal adicional que define el índice de nodos de restricciones (en oposición al índice de aristas) que no se genera en la realización de la Fig. 16. El índice de los nodos de restricciones define el destino de los nodos de restricciones del mensaje V2C actual. Este campo sirve como un índice para la memoria de estado de restricciones 1712 a la cual se suministra.

Aunque requieren un poco más de circuitería que la realización de la Fig. 15, las realizaciones de las Fig. 16 y la Fig. 17 tienen la ventaja de un uso más eficaz de los procesadores de los nodos de vectores de restricciones y de variables 1609/1709, 1608/1708 ya que ambos procesadores de nodos de vectores se utilizan totalmente durante cada una de las iteraciones del procesamiento. Además, el tiempo de decodificación se reduce en comparación con la realización de la Fig. 15 ya que el procesamiento de los nodos de restricciones y variables se realiza en paralelo, por ejemplo simultáneamente.

Los procedimientos de decodificación descritos anteriormente permiten que se realice la decodificación del paso de mensajes, por ejemplo la decodificación LDPC usando software y ordenadores de propósito general capaces de soportar las operaciones de SIMD. En tales realizaciones, uno o más procesadores en paralelo sirven como unidades de procesamiento de vectores, o puede usarse hardware dentro de un procesador único para realizar operaciones de procesamiento de múltiples vectores en paralelo. En tales realizaciones, la memoria de aristas, el mapa de permutaciones y la información sobre el número de mensajes por nodo pueden todos almacenarse en una memoria común, por ejemplo la memoria principal de los ordenadores. La lógica del control de paso de mensajes y la lógica de control del decodificador pueden implementarse como rutinas software ejecutadas sobre la unidad de procesamiento del ordenador. Además, el dispositivo de conmutación puede implementarse usando software y una o más instrucciones de procesamiento SIMD.

Los procedimientos de decodificación LDPC descritos anteriormente permiten la realización de la decodificación de LDPC sobre diversas plataformas de hardware tales como las Disposiciones de Puertas Programables en Campo o en un Circuito Integrado de Aplicación Específica. La presente invención es especialmente útil en estas configuraciones donde puede explotarse explícitamente el paralelismo simple.

Numerosas variaciones adicionales sobre los procedimientos de decodificación y el aparato de la presente invención serán evidentes para los especialistas en la técnica a la vista de la descripción anterior de la invención. Tales variaciones se considerarán dentro del alcance de la invención como se define por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un aparato (1500, 1600, 1700) para la realización de operaciones de decodificación de paso de mensajes usando gráficos LDPC convertidos a vectores que representan matrices de comprobación de paridad elevadas por lo que, en una matriz de comprobación de paridad elevada los elementos cero de una matriz de comprobación de paridad H de un código LDPC proyectado se reemplazan con matrices de Z x Z ceros y los elementos 1 de la matriz de comprobación de paridad H se reemplazan con matrices de permutación de Z x Z comprendiendo el aparato:

una memoria (1506) incluyendo un conjunto de localizaciones de memoria para almacenar L conjuntos de Z mensajes de K bits, donde Z es un número entero positivo mayor de uno y K y L son números enteros positivos distintos de cero;
un procesador de vectores de nodos (1508) incluyendo Z unidades de procesamiento de nodos en paralelo, cada una de las unidades de procesamiento de nodos (1508) para realizar al menos una de las operaciones de procesamiento de nodos de restricciones y una operación de procesamiento de nodos de variables; y
un dispositivo de conmutación (1520) acoplado a la memoria (1506) y al procesador de vectores de nodos (1508), el dispositivo de conmutación (1520) para pasar conjuntos de Z mensajes de K bits, pasado cada conjunto de Z mensajes de K bits en paralelo entre dicha memoria y dicho procesador de vectores de nodos y para reordenar los mensajes en al menos uno de dichos conjuntos de mensajes en respuesta a la información de control de conmutación,
un módulo de ordenamiento de mensajes (1504) acoplado a dicho dispositivo de conmutación para generar dicha información de control de conmutación usada para controlar el reordenamiento de mensajes en dicho, al menos, un conjunto de mensajes, en el que el módulo de reordenamiento de mansajes (1504) está además acoplado a dicha memoria y genera secuencialmente conjuntos de indicadores, controlando cada uno de los identificadores de conjunto la memoria para acceder a las localizaciones de memoria correspondientes a uno de dichos conjuntos de mensajes en una operación de lectura o escritura única por la que, uno de dichos conjuntos de Z mensajes de K bits se escribe o se lee como una unidad única accediendo a todos los Z mensajes de dicho conjunto usando una instrucción SIMD.

2. El aparato (1500) de la reivindicación 1, en el que el dispositivo de conmutación (1520) incluye circuitería para realizar una operación de rotación de mensajes para reordenar los mensajes incluidos en un conjunto de mensajes.

3. El aparato (1500) de la reivindicación 1, en el que el módulo de reordenamiento de mensajes (1504) almacena información sobre el orden de los conjuntos de los mensajes a leer de la memoria y la información que indica qué reordenamiento de mensajes se realizará por dicho conmutador sobre los conjuntos individuales de mensajes leídos de la memoria.

4. El aparato (1500) de la reivindicación 1, en el que dicho identificador de conjunto es una simple dirección de memoria.

5. El aparato (1500) de la reivindicación 1, en el que dicha pluralidad de unidades de procesamiento de nodos incluye Z unidades de procesamiento de nodos dispuestas en paralelo, funcionando cada una de las Z unidades de procesamiento de nodos en paralelo para procesar un mensaje diferente en cada uno de los conjuntos de Z mensajes pasados entre dicha memoria y dicho procesador de nodos.

6. El aparato (1500) de la reivindicación 5, en el que dicha memoria (1506) incluye una entrada de dirección que permite dirigirse a cada uno de los conjuntos de mensajes como una unidad posibilitando por lo tanto la lectura de un conjunto de mensajes desde dicha memoria en una única operación de lectura de SMID.

7. El aparato de la reivindicación 5, en el que dicha memoria (1506) incluye una entrada de dirección que permite dirigirse a cada uno de los conjuntos de mensajes como dicha unidad posibilitando por lo tanto que un conjunto de mensajes se escriban dentro de dicha memoria en una única operación de escritura de SMID.

8. El aparato (1500) de la reivindicación 5,

en el que el dispositivo de control del decodificador está además acoplado a un dispositivo de control del paso de mensajes; y
en el que el dispositivo de control del paso de mensajes especifica un orden diferente en el que se leerán cada uno de los L conjuntos de Z mensajes de la memoria durante el modo de nodos de variables de la operación de procesamiento que durante el modo de nodos de restricciones de la operación de procesamiento.

9. El aparato (1500) de la reivindicación 1, que comprende además un módulo de control del decodificador (1502) acoplado al módulo de ordenamiento de mensajes (1504), incluyendo el módulo de control del decodificador medios para el suministro de información al módulo de ordenamiento de mensajes utilizado para controlar el orden en el que se leerá cada uno de los L conjuntos de Z mensajes de dicha memoria.

10. El aparato (1500) de la reivindicación 9, en el que el módulo de control del decodificador (1502) incluye además

medios para el suministro de un índice de aristas al módulo de ordenamiento de mensajes (1504) que controla la generación de los identificadores de conjuntos suministrados a dicha memoria.

11. El aparato (1500) de la reivindicación 10, que comprende además una memoria de grados (1510) acoplada con el procesador de vectores de nodos (1508) para el almacenamiento de un conjunto de información del grado de los nodos.

12. El aparato (1500) de la reivindicación 11, en el que el módulo de control (1502) genera además un índice de nodos usado para determinar qué información del grado de los nodos en el conjunto almacenado de la información del grado de los nodos se aplicará al procesador de vectores de nodos en cualquier momento determinado.

13. El aparato (1600) de la reivindicación 1, que comprende además:

un segundo procesador de vectores de nodos (1609, 1608) acoplado a dicha memoria, incluyendo el segundo procesador de vectores de nodos unidades de procesamiento de nodos, cada una de las unidades de procesamiento de nodos para realizar una operación de procesamiento de nodos de restricciones,
en el que dicho procesador de vectores de nodos es un procesador de nodos de variables para la realización de las operaciones de procesamiento de comprobación de paridad del decodificador de nodos de variables;
en el que dicho segundo procesador de nodos es un procesador de nodos de restricciones (1609) para la realización de las operaciones de procesamiento del decodificador de comprobación de la paridad de nodos de restricciones.

14. El aparato (1600) de la reivindicación 13, que comprende además:

la memoria adicional (1607, 1606) que acopla dicho procesador de vectores de nodos a dicho segundo procesador de vectores de nodos, incluyendo la memoria adicional un conjunto de localizaciones de memoria para el almacenamiento de L conjuntos de Z mensajes de K bits.

15. El aparato (1600) de la reivindicación 14, que comprende además:

un segundo dispositivo de conmutación (1620, 1621) que acopla dicho procesador de vectores de nodos a dicha memoria adicional, un segundo dispositivo de conmutación para pasar conjuntos de Z mensajes de K bits entre dicho procesador de vectores de nodos y dicha memoria adicional y para reordenar los mensajes en al menos uno de los conjuntos de mensajes pasado por el segundo conmutador.

16. El aparato (1600) de la reivindicación 13, que comprende además:

un verificador de comprobación de paridad (1614), acoplado a dicho procesador de vectores de nodos, para determinar a partir de una salida de cada una de las unidades de procesamiento incluidas en el mismo, cuándo se ha completado satisfactoriamente la operación de decodificación de comprobación de paridad.

17. Un procedimiento para la realización del procesamiento de decodificación del paso de mensajes para la decodificación de gráficos de LDPC convertidos a vectores que representan matrices de comprobación de paridad elevadas por lo que en una matriz de comprobación de paridad elevada, los elementos cero de una matriz de comprobación de paridad H de un código LDPC proyectado se reemplazan con matrices de Z x Z ceros y los elementos 1 de la matriz de comprobación de paridad H se reemplazan con matrices de permutación de Z x Z que comprenden las etapas de:

almacenar L conjuntos de mensajes de K bits en una memoria (1506), incluyendo cada uno de los conjuntos de mensajes de K bits en primer lugar Z mensajes, donde Z es un número entero positivo mayor que uno y K y L son números enteros positivos distintos de cero;
leer uno de dichos conjuntos de mensajes de K bits de la memoria (1506);
realizar una operación de reordenamiento de mensajes en dicho conjunto de mensajes de K bits leídos para producir un conjunto reordenado de Z mensajes de K bits;
suministrar, en paralelo, los Z mensajes en el conjunto reordenado de mensajes a un procesador de vectores de nodos (1508); y que opera el procesador de vectores de nodos (1508) para realizar las operaciones del decodificador de paso de mensajes usando Z mensajes suministrados como entrada, incluyendo dicho procesador de vectores de nodos (1508) Z unidades de procesamiento en paralelo, por lo que cada una de las unidades de procesamiento de nodos realiza al menos una operación de procesamiento de nodos de restricciones y una operación de procesamiento de nodos de variables, comprendiendo el procedimiento además:

generar un identificador de conjunto indicando el conjunto de Z mensajes a leer de la memoria (1506), en el que la etapa de leer uno de dichos conjuntos de mensajes de K bits incluye:

realizar una única operación de lectura usando dicho identificador de conjunto para acceder con dicha operación de lectura única a todos los Z mensajes de K bits de dicho conjunto como una

única unidad usando una instrucción SMID.

18. El procedimiento de la reivindicación 17, en el que dichas operaciones del decodificador de paso de mensaje generan un conjunto de Z mensajes de decodificador a partir de los Z mensajes en el conjunto de mensajes reordenados suministrado.

19. El procedimiento de la reivindicación 17, que comprende además:

realizar una segunda operación de reordenamiento de mensajes, realizándose la segunda operación de reordenamiento de mensajes sobre el conjunto generado de Z mensajes del decodificador para producir un conjunto reordenado de mensajes del decodificador generados.

20. El procedimiento de la reivindicación 19, que comprende además:

almacenar el conjunto reordenado de mensajes del decodificador generados en dicha memoria.

21. El procedimiento de la reivindicación 20, en el que la etapa de almacenar el conjunto reordenado de mensajes del decodificador generados incluye realizar una operación de escritura SMID para escribir dicho conjunto reordenado de mensajes del decodificador generados en la memoria.

22. El procedimiento de la reivindicación 19, en el que la etapa de realizar una segunda operación de reordenamiento de mensajes incluye realizar la inversa de la operación de reordenamiento de mensajes realizado sobre dicho conjunto de mensajes de K bits leídos desde la memoria.

23. El procedimiento de la reivindicación 17, que comprende además:

acceder a la información de permutación del conjunto de mensajes almacenados; y
en el que la etapa de realización de una operación de reordenamiento de mensajes incluye la etapa de:
realizar dicho reordenamiento como una función de la información de permutación del conjunto de mensajes almacenados accedidos.

24. El procedimiento de la reivindicación 22, en el que dicha información de permutación del conjunto de mensajes incluye información de rotación cíclica.

25. El procedimiento de la reivindicación 17, en el que dichas operaciones del decodificador de paso de mensajes son operaciones de procesamiento de nodos de variables, incluyendo cada una de las operaciones de procesamiento de nodos de variables un valor de decisión, y en el que el procedimiento comprende además:

examinar los valores de decisión generados operando el procesador de vectores para determinar si se ha satisfecho una condición de decodificación.

26. Un procedimiento de realización del procesamiento de decodificador de paso de mensajes para decodificar los gráficos LDPC convertidos a vectores que representan las matrices de comprobación de paridad elevadas por lo que en una matriz de comprobación de paridad elevada, los elementos cero de una matriz de comprobación de paridad H de un código LDPC proyectado se reemplazan con matrices de Z x Z ceros y los elementos 1 de la matriz de comprobación de paridad H se reemplazan con matrices de permutación de Z x Z, comprendiendo el procedimiento las etapas de:

operar un procesador de vectores de nodos (1508) para generar un conjunto de Z mensajes de K bits, a almacenar en un dispositivo de memoria (1506) para almacenar L conjuntos de Z mensajes de K bits, donde Z es un número entero positivo mayor de uno y K y L son números enteros positivos distintos de cero, incluyendo dicho procesador de vectores de nodos (1508) Z unidades de procesamiento en paralelo, por lo que cada una de las unidades de procesamiento de nodos realiza al menos una de las operaciones de procesamiento de nodos de restricciones y una operación de procesamiento de nodos de variables;
realizar una operación de reordenamiento de mensajes en el conjunto generado de Z mensajes de K bits para producir un conjunto reordenado de Z mensajes de K bits;
realizar una única operación de escritura para almacenar el conjunto reordenado de Z mensajes de K bits en dicho dispositivo de memoria (1506), en el que la etapa de realizar una operación de escritura única para almacenar el conjunto de Z mensajes de K bits reordenados comprende realizar una única operación de escritura usando un identificador de conjunto para escribir con dicha única operación de escritura todos los Z mensajes de K bits de dicho conjunto reordenado como una única usando una instrucción SIMD.

27. El procedimiento de la reivindicación 26, en el que la etapa de operar el procesador de vectores de nodos para generar un conjunto de Z mensajes de K bits, incluye la etapa de:

realizar, en paralelo, las Z operaciones de procesamiento de nodos, generando cada una de las operaciones de procesamiento de nodos un mensaje en dicho conjunto de Z mensajes de K bits.

28. El procedimiento de la reivindicación 26, en el que las Z operaciones de procesamiento de nodos son operaciones de procesamiento de nodos de variables.

29. El procedimiento de la reivindicación 28, en el que las Z operaciones de procesamiento de nodos son operaciones de procesamiento de nodos de restricciones.

30. El procedimiento de la reivindicación 28, en el que realizar una operación de reordenamiento de mensajes en el conjunto generado de Z mensajes de K bits incluye:

girar los mensajes en el conjunto de Z mensajes de K bits realizando una operación de conmutación para reordenar los mensajes en el conjunto de mensajes.

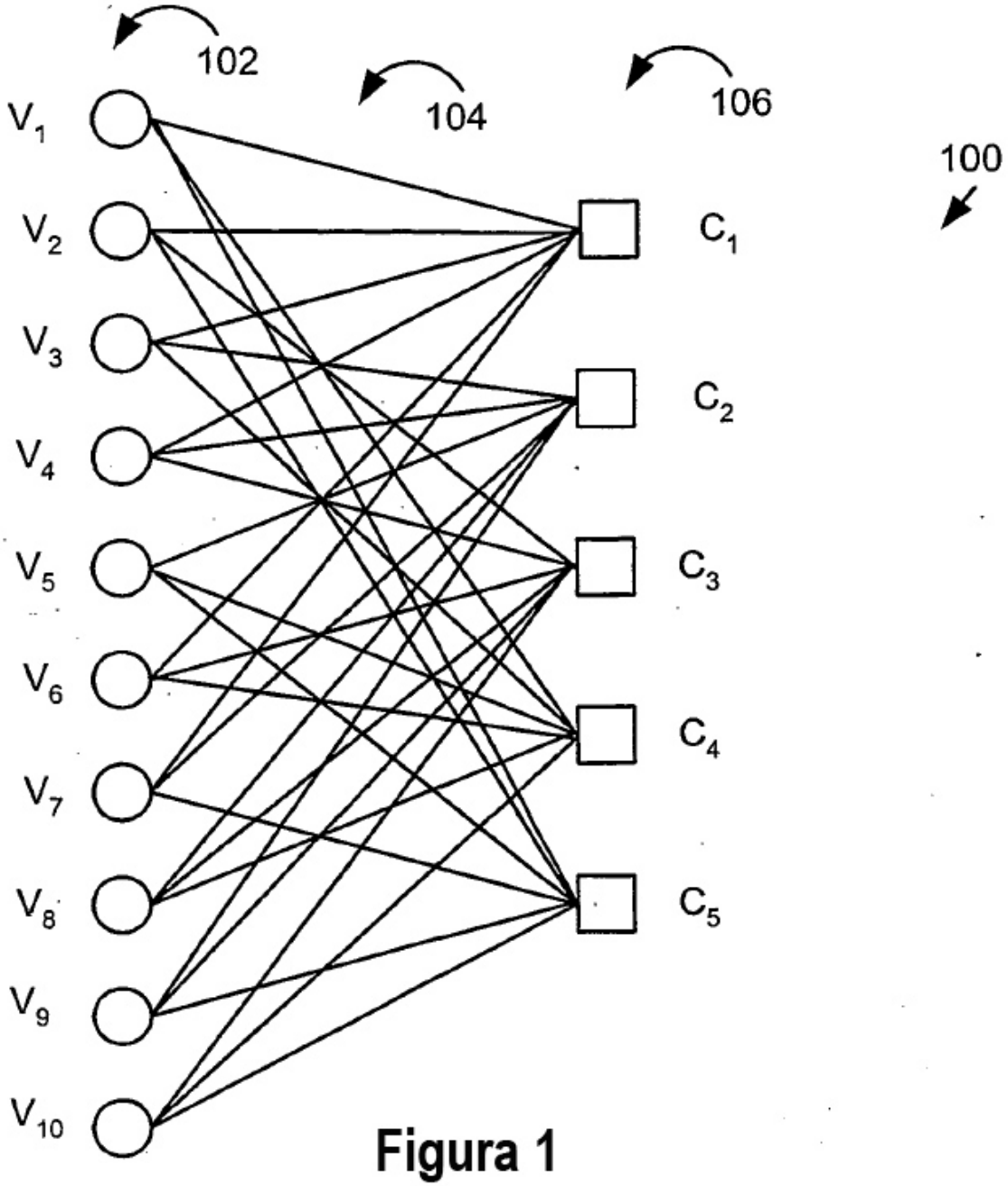

Figura 1

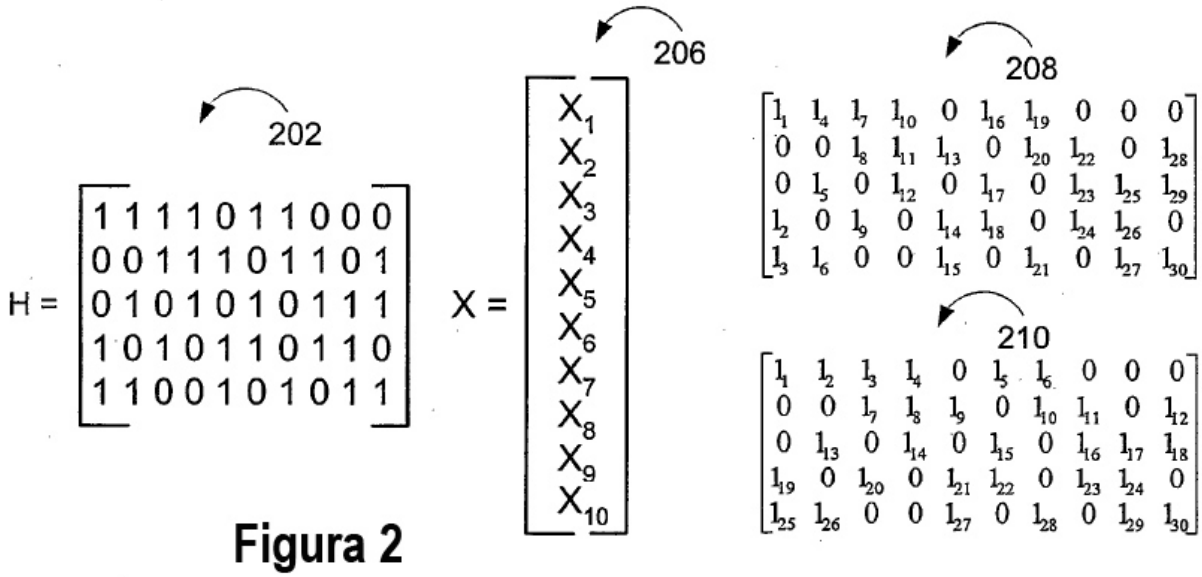
202
206
208
210
H =
X =
1 1 1 1 0 1 1 0 0 0
0 0 1 1 1 0 1 1 0 1
0 1 0 1 0 1 0 1 1 1
1 0 1 0 1 1 0 1 1 0
1 1 0 0 1 0 1 0 1 1
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
l1 l4 l7 l10 0 l16 l19 0 0 0
0 0 l8 l11 l13 0 l20 l22 0 l28
0 l5 0 l12 0 l17 0 l23 l25 l29
l2 0 l9 0 l14 l18 0 l24 l26 0
l3 l6 0 0 l15 0 l21 0 l27 l30
l1 l2 l3 l4 0 l5 l6 0 0 0
0 0 l7 l8 l9 0 l10 l11 0 l12
0 l13 0 l14 0 l15 0 l16 l17 l18
l19 0 l20 0 l21 l22 0 l23 l24 0
l25 l26 0 0 l27 0 l28 0 l29 l30

Figura 2

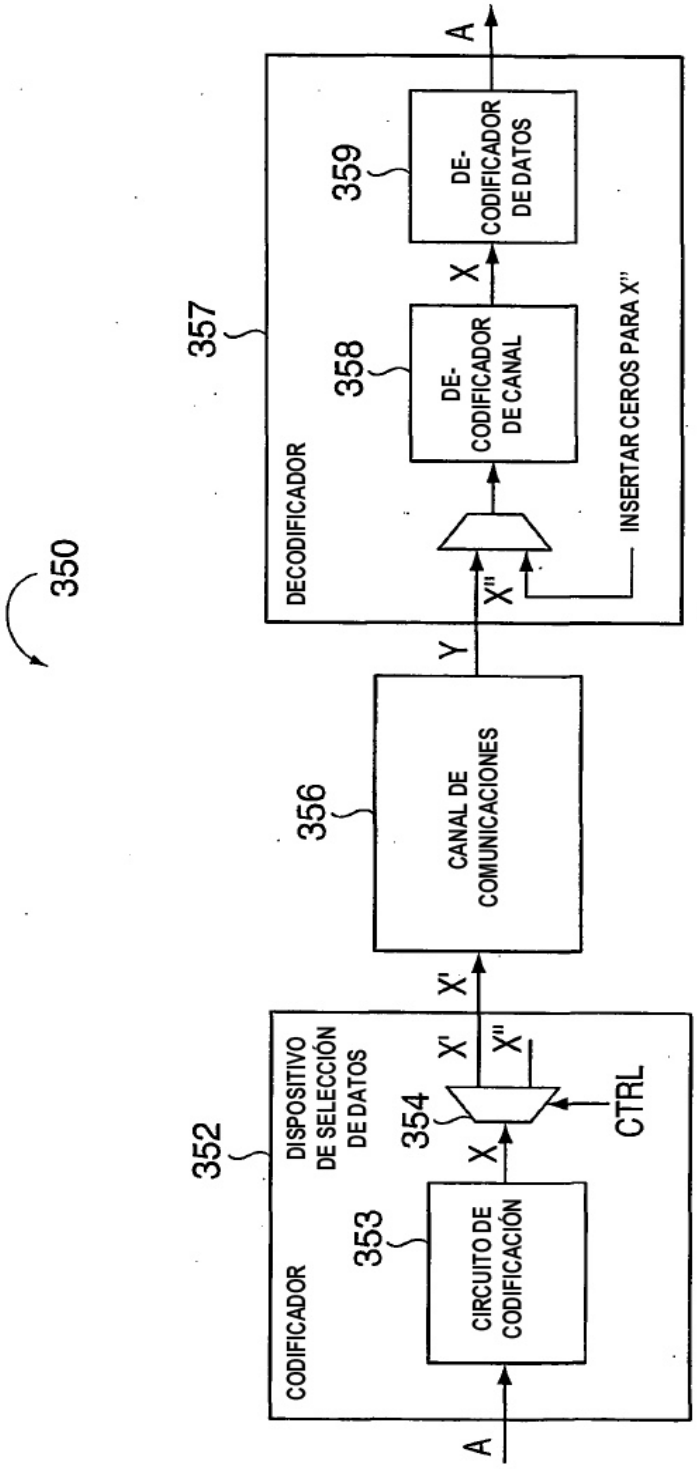
350
CODIFICADOR
352
DISPOSITIVO DE SELECCIÓN DE DATOS
353
CIRCUITO DE CODIFICACIÓN
354
CTRL
A
X
X'
X''
356
CANAL DE COMUNICACIONES
Y
DECODIFICADOR
357
358
DE-CODIFICADOR DE CANAL
359
DE-CODIFICADOR DE DATOS
INSERTAR CEROS PARA X''
A

FIG. 3

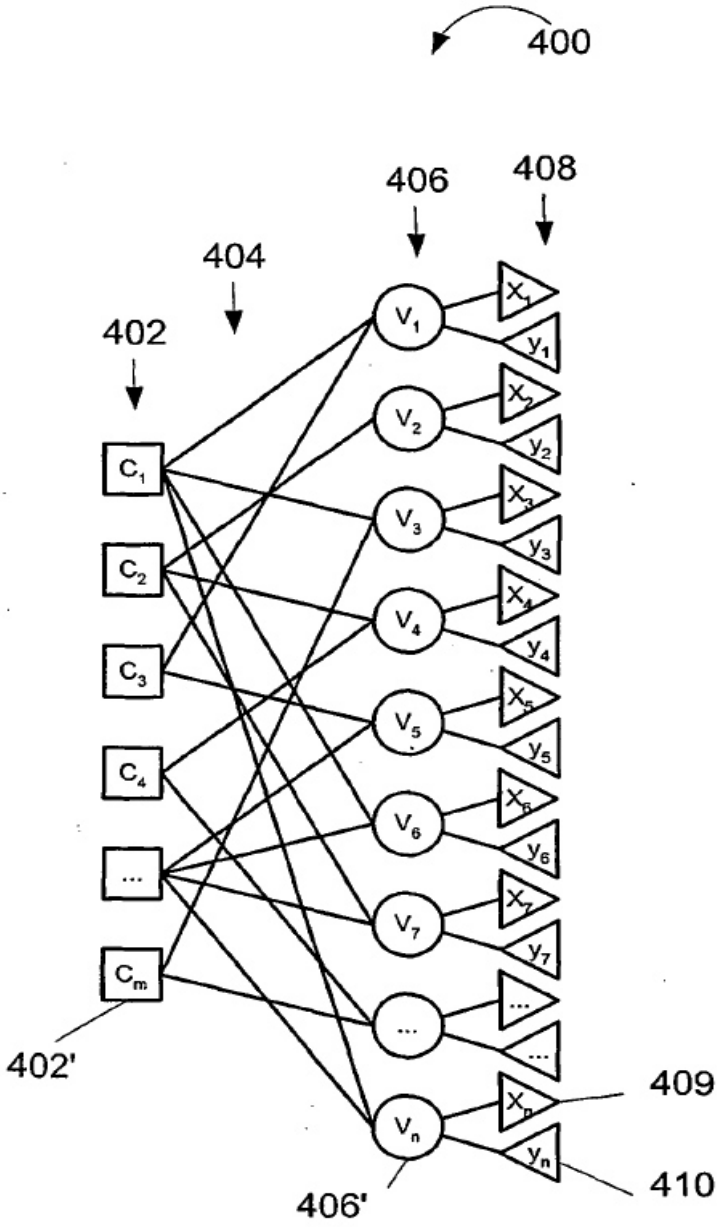
400
406
408
404
402
C_1
C_2
C_3
C_4
...
C_m
V_1
V_2
V_3
V_4
V_5
V_6
V_7
...
V_n
x_1
y_1
x_2
y_2
x_3
y_3
x_4
y_4
x_5
y_5
x_6
y_6
x_7
y_7
...
...
x_n
y_n
402'
406'
409
410

Figura 4

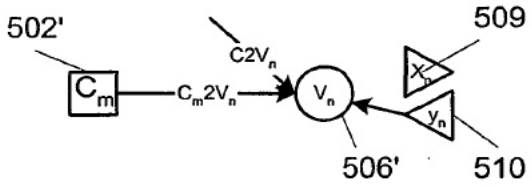

Figura 5a

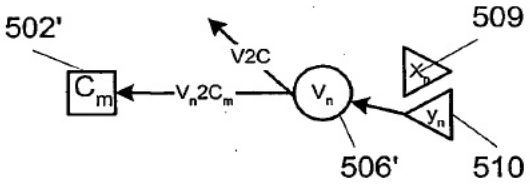

Figura 5b

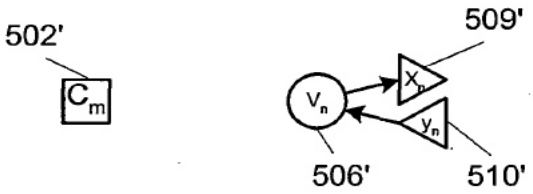

Figura 5c

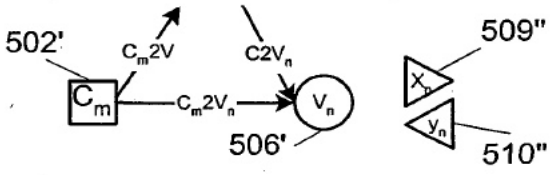

Figura 5d

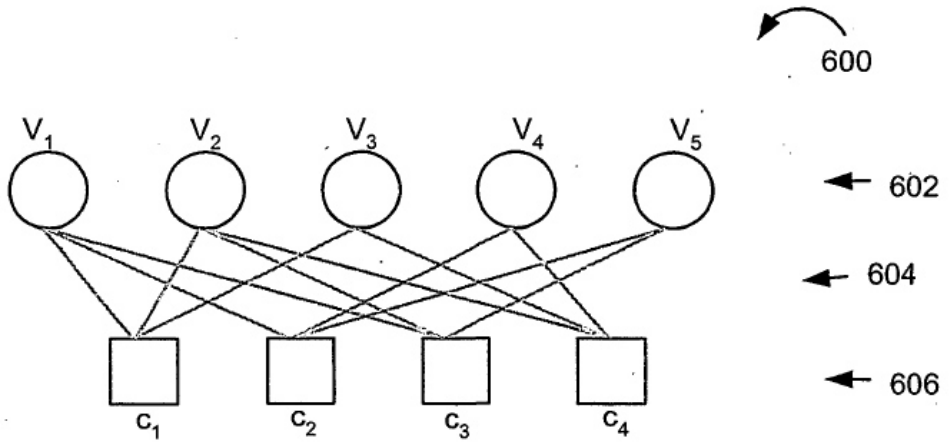

Figura 6

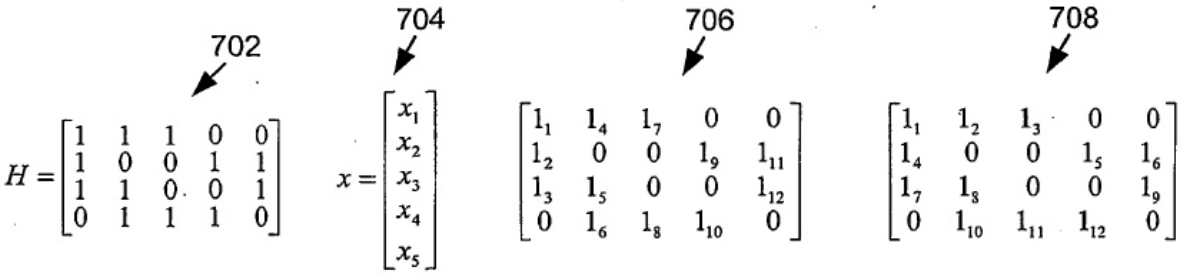

Figura 7

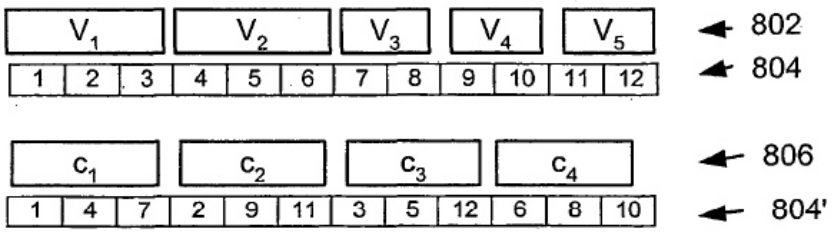

Figura 8

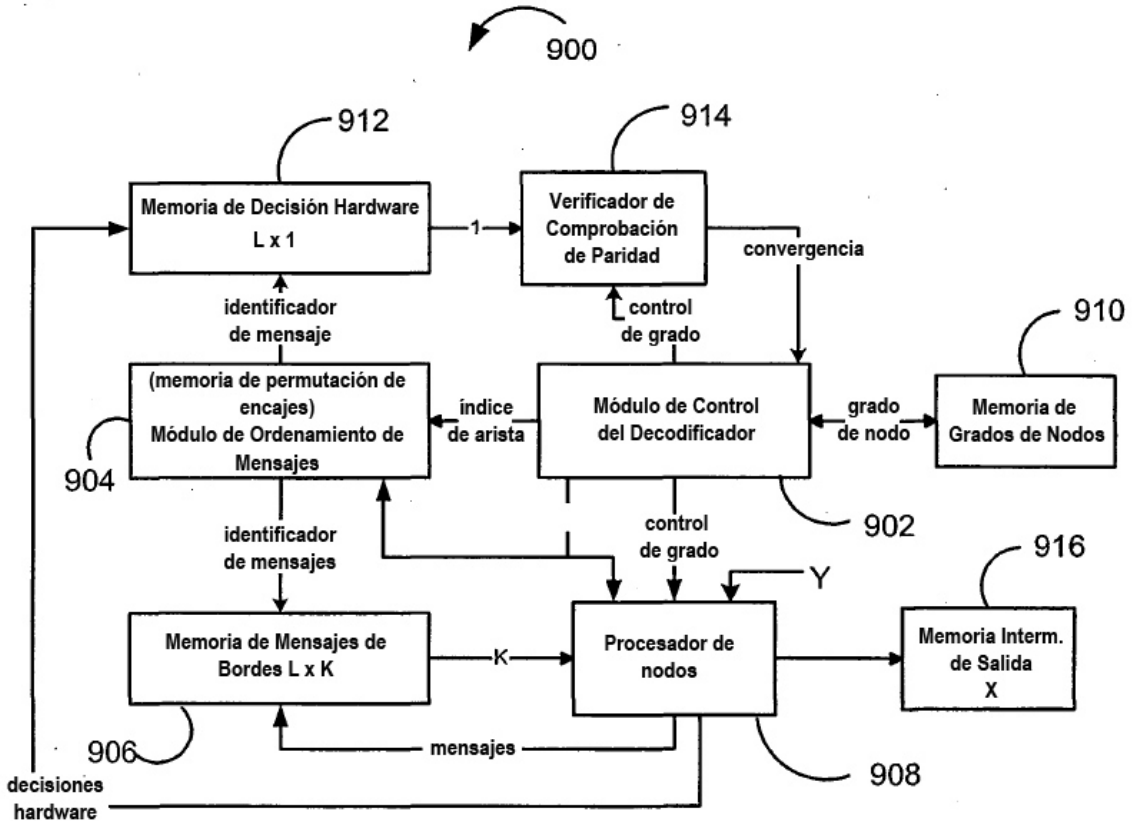
900
912
914
Memoria de Decisión Hardware
L x 1
1
Verificador de
Comprobación
de Paridad
convergencia
identificador
de mensaje
control
de grado
910
(memoria de permutación de
encajes)
Módulo de Ordenamiento de
Mensajes
índice
de arista
Módulo de Control
del Decodificador
grado
de nodo
Memoria de
Grados de Nodos
904
902
identificador
de mensajes
control
de grado
Y
916
Memoria de Mensajes de
Bordes L x K
K
Procesador de
nodos
Memoria Interm.
de Salida
X
906
mensajes
908
decisiones
hardware

Figura 9

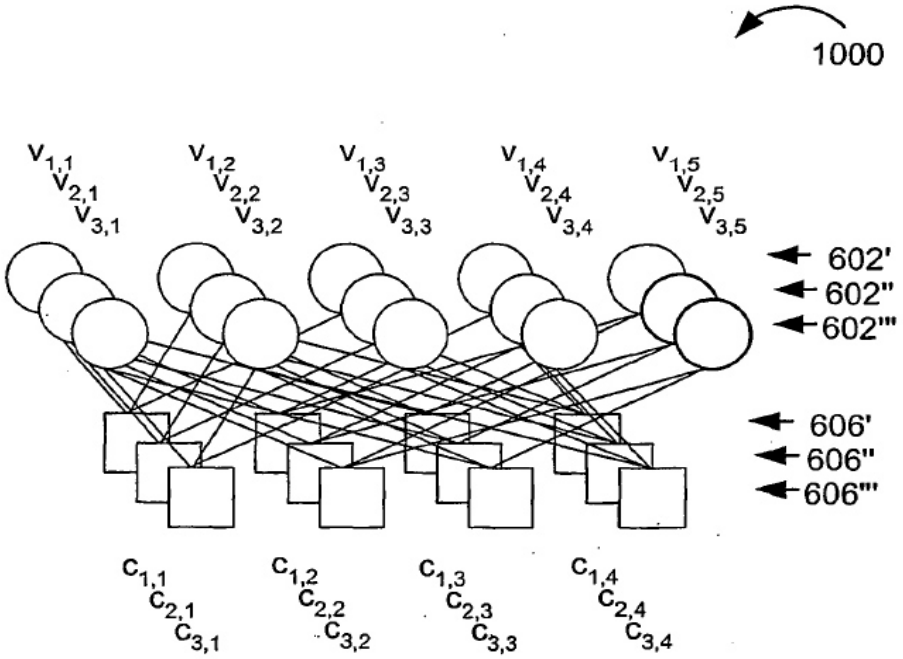

Figura 10

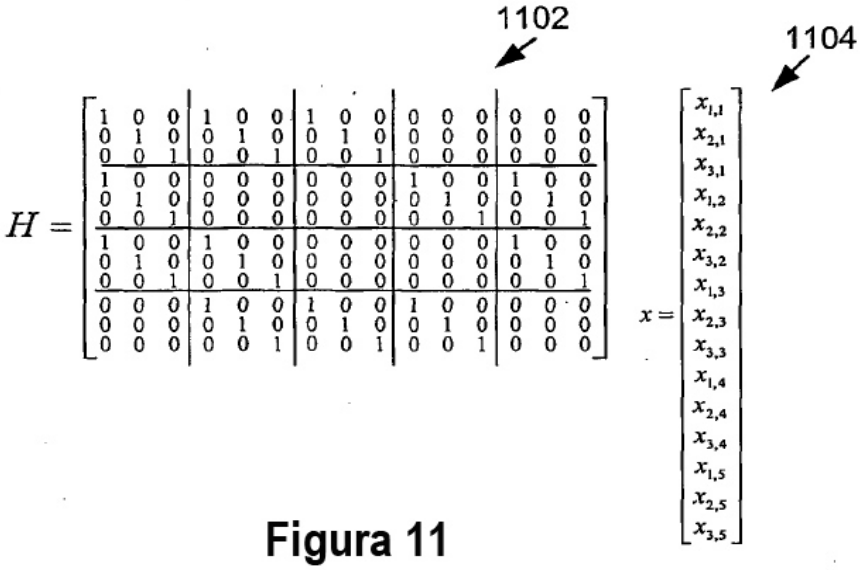

Figura 11

	$v_{,1}$			$v_{,2}$			$v_{,3}$		$v_{,4}$		$v_{,5}$		
v_1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11	1,12	← 1202'
v_2	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10	2,11	2,12	← 1202"
v_3	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9	3,10	3,11	3,12	← 1202'''

	$c_{,1}$			$c_{,2}$			$c_{,3}$			$c_{,4}$			
c_1	1,1	1,4	1,7	1,2	1,9	1,11	1,3	1,5	1,12	1,6	1,8	1,10	← 1204'
c_2	2,1	2,4	2,7	2,2	2,9	2,11	2,3	2,5	2,12	2,6	2,8	2,10	← 1204"
c_3	3,1	3,4	3,7	3,2	3,9	3,11	3,3	3,5	3,12	3,6	3,8	3,10	← 1204'''

Figura 12

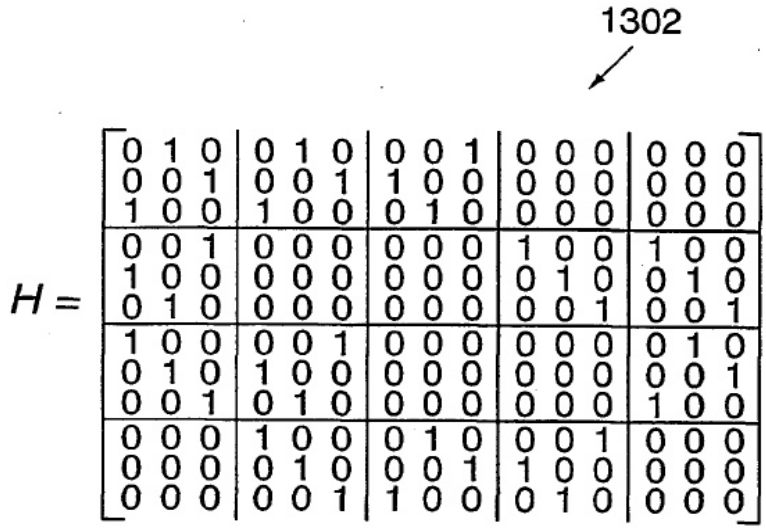

FIGURA 13

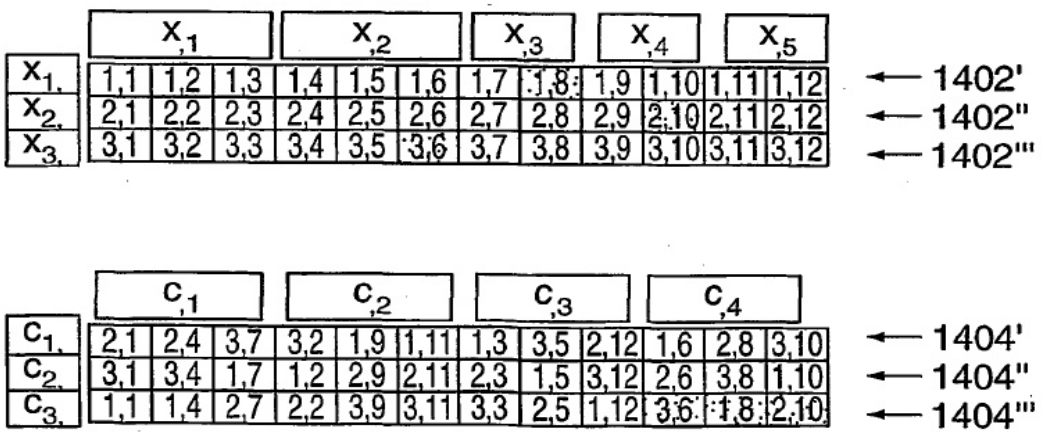

FIGURA 14

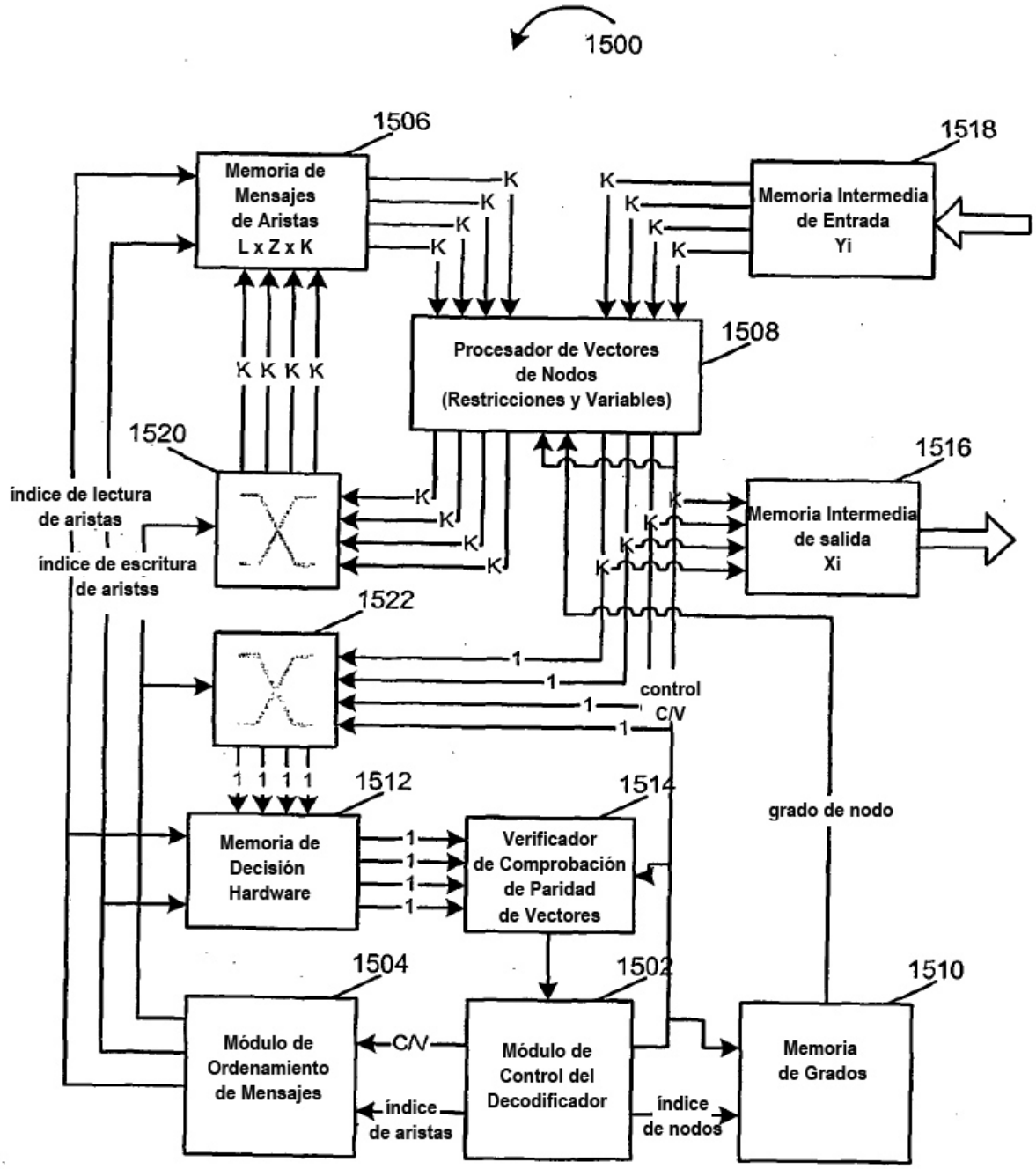
1500
1506
Memoria de Mensajes de Aristas L x Z x K
1518
Memoria Intermedia de Entrada Yi
1508
Procesador de Vectores de Nodos (Restricciones y Variables)
1520
índice de lectura de aristas
índice de escritura de aristss
1516
Memoria Intermedia de salida Xi
1522
control C/V
1512
Memoria de Decisión Hardware
1514
Verificador de Comprobación de Paridad de Vectores
grado de nodo
1504
Módulo de Ordenamiento de Mensajes
C/V
índice de aristas
1502
Módulo de Control del Decodificador
índice de nodos
1510
Memoria de Grados
K
1

Figura 15

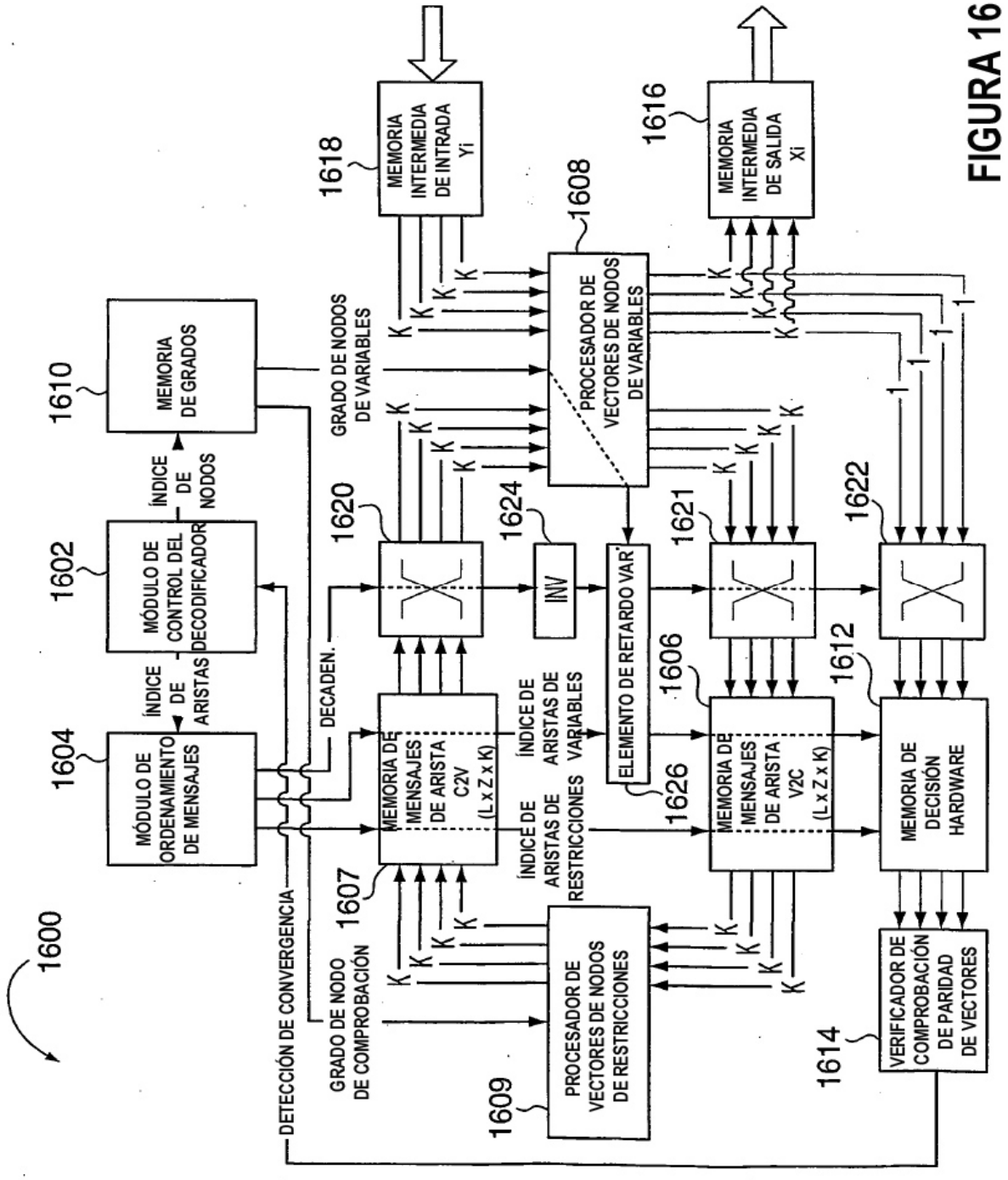
1600
1602
MÓDULO DE CONTROL DEL DECODIFICADOR
1604
MÓDULO DE ORDENAMIENTO DE MENSAJES
1606
MEMORIA DE MENSAJES DE ARISTA V2C (L x Z x K)
1607
MEMORIA DE MENSAJES DE ARISTA C2V (L x Z x K)
1608
PROCESADOR DE VECTORES DE NODOS DE VARIABLES
1609
PROCESADOR DE VECTORES DE NODOS DE RESTRICCIONES
1610
MEMORIA DE GRADOS
1612
MEMORIA DE DECISIÓN HARDWARE
1614
VERIFICADOR DE COMPROBACIÓN DE PARIDAD DE VECTORES
1616
MEMORIA INTERMEDIA DE SALIDA Xi
1618
MEMORIA INTERMEDIA DE INTRADA Yi
1620
1621
1622
1624
INV
1626
ELEMENTO DE RETARDO VAR
ÍNDICE DE NODOS
ÍNDICE DE ARISTAS
DECADEN.
GRADO DE NODOS DE VARIABLES
ÍNDICE DE ARISTAS DE VARIABLES
ÍNDICE DE ARISTAS DE RESTRICCIONES
GRADO DE NODO DE COMPROBACIÓN
DETECCIÓN DE CONVERGENCIA
K
1

FIGURA 16

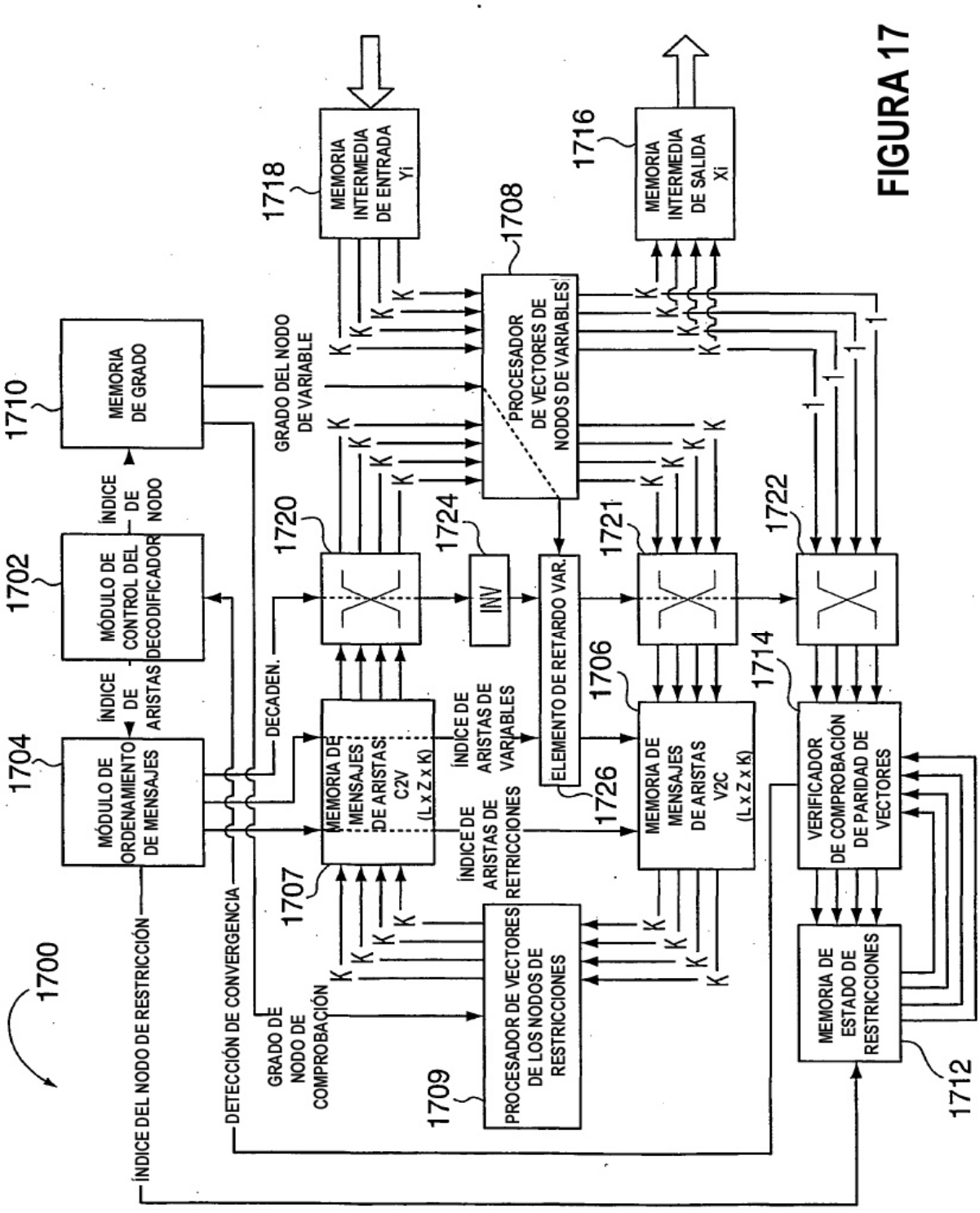
1700
1702
MÓDULO DE CONTROL DEL DECODIFICADOR
ÍNDICE DE NODO
ÍNDICE DE ARISTAS
1704
MÓDULO DE ORDENAMIENTO DE MENSAJES
1706
MEMORIA DE MENSAJES DE ARISTAS V2C (L x Z x K)
1707
MEMORIA DE MENSAJES DE ARISTAS C2V (L x Z x K)
1708
PROCESADOR DE VECTORES DE NODOS DE VARIABLES
1709
PROCESADOR DE VECTORES DE LOS NODOS DE RESTRICCIONES
1710
MEMORIA DE GRADO
1712
MEMORIA DE ESTADO DE RESTRICCIONES
1714
VERIFICADOR DE COMPROBACIÓN DE PARIDAD DE VECTORES
1716
MEMORIA INTERMEDIA DE SALIDA Xi
1718
MEMORIA INTERMEDIA DE ENTRADA Yi
1720
1721
1722
1724
INV
1726
ELEMENTO DE RETARDO VAR.
GRADO DEL NODO DE VARIABLE
DECADEN.
ÍNDICE DE ARISTAS DE VARIABLES
ÍNDICE DE ARISTAS DE RETRICCIONES
GRADO DE NODO DE COMPROBACIÓN
DETECCIÓN DE CONVERGENCIA
ÍNDICE DEL NODO DE RESTRICCIÓN
K
1

FIGURA 17