(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 516 765**

(51) Int. CI.:

G06N 3/02 (2006.01)
H03M 13/11 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **31.05.2002 E 10009716 (1)**

(97) Fecha y número de publicación de la concesión europea: **13.08.2014 EP 2264907**

(54) Título: **Procedimientos y aparatos para descodificar códigos LDPC**

(30) Prioridad:

15.06.2001 US 298480 P
10.10.2001 US 975331

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
31.10.2014

(73) Titular/es:

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

(72) Inventor/es:

RICHARDSON, TOM y
NOVICHKOV, VLADIMIR

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Procedimientos y aparatos para descodificar códigos LDPC

Campo de la invención

La presente invención está orientada a procedimientos y aparatos para detectar y / o corregir errores en datos binarios, p. ej., mediante el uso de códigos de control de paridad tales como los códigos de control de paridad de baja densidad (LDPC).

Antecedentes

En la moderna era de la información, los valores binarios, p. ej., unos y ceros, son usados para representar y comunicar diversos tipos de información, p. ej., de vídeo, de audio, información estadística, etc. Lamentablemente, durante el almacenamiento, la transmisión y / o el procesamiento de datos binarios, pueden introducirse errores no intencionalmente, p. ej., un uno puede ser cambiado por un cero o viceversa.

En general, en el caso de la transmisión de datos, un receptor observa cada bit recibido en presencia de ruido o distorsión, y solamente se obtiene una indicación del valor del bit. En estas circunstancias, se interpretan los valores observados como una fuente de bits “blandos”. Un bit blando indica una estimación preferida del valor del bit, es decir, un uno o un cero, junto con alguna indicación de la fiabilidad de esa estimación. Si bien el número de errores puede ser relativamente bajo, incluso un pequeño número de errores, o un pequeño nivel de distorsión, puede dar como resultado que los datos sean inutilizables o, en el caso de errores de transmisión, puede requerir la retransmisión de los datos.

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

Cuando las señales surgidas de las palabras de código transmitidas son recibidas o procesadas, la información redundante incluida en la palabra de código, según lo observado en la señal, puede ser usada para identificar y / o corregir errores en, o eliminar la distorsión de, la señal recibida, a fin de recuperar la unidad original de datos. Tal comprobación y / o corrección de errores puede ser implementada como parte de un proceso de descodificación. En ausencia de errores, o en el caso de errores o distorsiones corregibles, la descodificación puede ser usada para recuperar, de los datos de origen que se están procesando, la unidad original de datos que fue codificada. En el caso de errores irrecuperables, el proceso de descodificación puede producir alguna indicación de que los datos originales no pueden ser recuperados completamente. Tales indicaciones de fallo de descodificación pueden ser usadas para iniciar la retransmisión de los datos.

Si bien la redundancia de datos puede aumentar la fiabilidad de los datos a almacenar o transmitir, es a cambio del coste de espacio de almacenamiento y / o del uso del valioso ancho de banda de las comunicaciones. En consecuencia, es deseable añadir redundancia de manera eficaz, maximizando la magnitud de la capacidad ganada de corrección / detección de errores, para una magnitud dada de redundancia introducida en los datos.

Con el creciente uso de líneas de fibra óptica para la comunicación de datos, y los aumentos en la velocidad a la cual los datos pueden ser leídos de, y almacenados en, dispositivos de almacenamiento, p. ej., unidades de disco, cintas, etc., hay una creciente necesidad, no solamente de un uso eficaz de la capacidad de almacenamiento y transmisión de datos, sino también de la capacidad de codificar y descodificar datos a altas tasas de velocidad.

Si bien son importantes la eficacia de codificación y las altas velocidades de datos, para que un sistema de codificación y / o descodificación sea práctico para su uso en una amplia gama de dispositivos, p. ej., dispositivos de consumo, es importante que los codificadores y / o descodificadores sean capaces de ser implementados a un coste razonable. En consecuencia, la capacidad de implementar eficazmente los esquemas de codificación / descodificación usados con fines de corrección y / o detección de errores, p. ej., en términos de costes de hardware, puede ser importante.

Diversos tipos de esquemas de codificación han sido usados a lo largo de los años con fines de corrección de errores. Una clase de códigos, generalmente mencionados como “turbo-códigos”, fueron inventados recientemente (1993). Los turbo-códigos ofrecen significativos beneficios sobre técnicas más antiguas de codificación, tales como los códigos convolutivos, y han tenido numerosas aplicaciones.

Conjuntamente con la llegada de los turbo-códigos, ha habido creciente interés en otra clase de códigos relacionados, aparentemente más sencillos, usualmente denominados códigos de control de paridad de baja densidad (LDPC). Los códigos de LDPC fueron efectivamente inventados por Gallager alrededor de 40 años atrás (1961), pero solo recientemente han llegado a destacarse. Los turbo-códigos y los códigos de LDPC son esquemas de codificación que son usados en el contexto de los denominados sistemas de codificación iterativa, es decir, son descodificados usando

descodificadores iterativos. Recientemente, se ha mostrado que los códigos de LDPC pueden proporcionar muy buenas prestaciones de detección y corrección de errores, sobrepasando o igualando las de los turbo-códigos para grandes palabras de código, p. ej., con tamaños de palabra de código que superan aproximadamente 1.000 bits, dada una selección adecuada de parámetros de codificación de LDPC. Además, los códigos de LDPC, en potencia, pueden ser descodificados a velocidades mucho mayores que los turbo-códigos.

En muchos esquemas de codificación, las palabras de código más largas son a menudo más flexibles para fines de detección y corrección de errores, debido a la interacción de la codificación sobre un mayor número de bits. De tal modo, el uso de palabras de código largas puede ser beneficioso en términos del aumento de la capacidad de detectar y corregir errores. Esto es especialmente cierto para los turbo-códigos y los códigos de LDPC. Así, en muchas aplicaciones es deseable el uso de palabras de código largas, p. ej., palabras de código que superan los mil bits de longitud.

La principal dificultad hallada en la adopción de la codificación de LDPC y la Turbo-codificación en el contexto de las palabras de código largas, donde el uso de tales códigos ofrece la mejor promesa, es la complejidad de implementar estos sistemas de codificación. En un sentido práctico, la complejidad se traduce directamente en el coste de implementación. Ambos sistemas de codificación son significativamente más complejos que los sistemas de codificación usados tradicionalmente, tales como los códigos convolutivos y los códigos de Reed-Solomon.

El análisis de complejidad de los algoritmos de procesamiento de señales se centra usualmente en totales de operaciones. Al intentar explotar el paralelismo de hardware en sistemas de codificación iterativa, especialmente en el caso de los códigos de LDPC, una significativa complejidad surge, no de los requisitos de cálculo, sino más bien de los requisitos de encaminamiento. La raíz del problema está en la construcción de los códigos en sí.

Los códigos de LDPC y los turbo-códigos se apoyan en la intercalación de mensajes dentro de un proceso iterativo. A fin de que el código rinda bien, la intercalación debe tener buenas propiedades de mezcla. Esto hace necesaria la implementación de un complejo proceso de intercalación.

Los códigos de LDPC están bien representados por gráficos bipartitos, a menudo llamado gráficos de Tanner, en los cuales un conjunto de nodos, los nodos *variables*, corresponde a los bits de la palabra de código, y el otro conjunto de nodos, los nodos de *restricción*, a menudo llamados nodos de *verificación*, corresponden al conjunto de restricciones de control de paridad que definen el código. Los bordes en el gráfico conectan nodos variables con nodos de restricción. Un nodo variable y un nodo de restricción se dicen *vecinos* si están conectados por un borde en el gráfico. Para simplificar, suponemos en general que un par de nodos está conectado por un borde, a lo sumo. A cada nodo variable está asociado un bit de la palabra de código. En algunos casos, algunos de estos bits podrían ser *punzados* o *conocidos*, según se expone adicionalmente más adelante.

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

Los descodificadores y algoritmos de descodificación usados para descodificar palabras de código de LDPC funcionan intercambiando mensajes dentro del gráfico a lo largo de los bordes, y actualizando estos mensajes realizando cálculos en los nodos, en base a los mensajes entrantes. Tales algoritmos serán generalmente denominados algoritmos de paso de mensajes. Cada nodo variable en el gráfico está inicialmente provisto de un bit blando, denominado un *valor recibido*, que indica una estimación del valor del bit asociado, según lo determinado por observaciones provenientes, p. ej., del canal de comunicaciones. Idealmente, las estimaciones para bits individuales son estadísticamente independientes. Este ideal puede ser, y a menudo es, violado en la práctica. Una colección de valores recibidos constituye una *palabra recibida*. Para los fines de esta solicitud, podemos identificar la señal observada, p. ej., por el receptor en un sistema de comunicaciones, con la palabra recibida.

El número de bordes adosados a un nodo, es decir, un nodo variable o un nodo de restricción, se denomina el *grado* del nodo. Un gráfico o código *regular* es uno para el cual todos los nodos 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 originalmente considerados por Gallager (1961). A diferencia de un código “regular”, un código irregular tiene nodos de restricción y / o nodos variables de distintos grados. Por ejemplo, algunos nodos variables pueden ser de grado 4, otros de grado 3 y otros más de grado 2.

Si bien los códigos irregulares pueden ser más complicados de representar y / o de implementar, se ha mostrado que los códigos irregulares de LDPC pueden proporcionar prestaciones superiores de corrección / detección de errores, en comparación con los códigos regulares de LDPC.

A fin de describir más precisamente el proceso de descodificación, introducimos la noción de un *receptáculo* al describir gráficos de LDPC. Un receptáculo puede ser visto como una asociación de un borde en el gráfico con un nodo

en el gráfico. Cada nodo tiene un receptáculo para cada borde adosado al mismo, y los bordes están “enchufados en” los receptáculos. De tal modo, un nodo de grado d tiene d receptáculos adosados al mismo. Si el gráfico tiene L bordes, entonces hay L receptáculos en el sector de nodos variables del gráfico, llamados los receptáculos variables, y L receptáculos en el sector de nodos de restricción del gráfico, llamados los receptáculos de restricción. Con fines de identificación y ordenamiento, los receptáculos variables pueden ser enumerados como 1, ..., L, de modo que todos los receptáculos variables adosados a un nodo variable aparezcan contiguamente. En tal caso, si los tres primeros nodos variables tienen grados d_1, d_2 y d_3, respectivamente, entonces los receptáculos variables 1, ..., d_1 están adosados al primer nodo variable, los receptáculos variables d_1+1, ..., d_1+d_2 están adosados al segundo nodo variable y los receptáculos variables d_1+d_2+1, ..., d_1+d_2+d_3 están adosados al tercer nodo variable. Los receptáculos de nodos de restricción pueden ser enumerados, de manera similar, 1, ..., L, apareciendo contiguamente todos los receptáculos de restricción adosados a un nodo de restricción. Un borde puede ser visto como un apareo de receptáculos, con uno de cada par saliendo de cada lado del gráfico. De tal modo, los bordes del gráfico representan un intercalador o permutación sobre los receptáculos, desde un lado del gráfico, p. ej., el lado de nodos variables, al otro, p. ej., el lado de nodos de restricción. Las permutaciones asociadas a estos sistemas son a menudo complejas, reflejando la complejidad del intercalador según lo indicado anteriormente, y requiriendo un encaminamiento complejo del paso de mensajes para su implementación.

La noción de algoritmos de paso de mensajes implementados en gráficos es más general que la descodificación de LDPC. La visión general es un gráfico con nodos intercambiando mensajes a lo largo de los bordes en el gráfico, y realizando cálculos basados en mensajes entrantes, a fin de producir mensajes salientes.

Un gráfico ejemplar bipartito 100, que determina un código LDPC (3,6) regular de longitud diez y velocidad de un medio se muestra en la Fig. 1. La longitud diez indica que hay diez nodos variables V_1 a V_{10}, cada uno identificado con un bit de la palabra de código X_1 a X_{10} (y ninguna punción en este caso), identificados en general con el número 102 de referencia. La velocidad de un medio indica que los nodos de verificación son la mitad de los nodos variables, es decir, hay cinco nodos de verificación C_1 a C_5, identificados por el número 106 de referencia. La velocidad de un medio indica además que las cinco restricciones son linealmente independientes, según se expone más adelante. Cada una de las líneas 104 representa un borde, p. ej., un trayecto o conexión de comunicación, entre los nodos de verificación y los nodos variables con los cuales está conectada la línea. Cada borde identifica dos receptáculos, un receptáculo variable y un receptáculo de restricción. Los bordes pueden ser enumerados de acuerdo a sus receptáculos variables o a sus receptáculos de restricción. La enumeración de receptáculos variables corresponde al ordenamiento de bordes (de arriba hacia abajo) según aparece en el sector de nodos variables, en el punto en donde están conectados con los nodos variables. La enumeración de receptáculos de restricción corresponde al ordenamiento de bordes (de arriba hacia abajo) según aparece en el sector de nodos de restricción, en el punto donde están conectados con los nodos de restricción. Durante la descodificación, los mensajes se pasan en ambas direcciones a lo largo de los bordes. Así, como parte del proceso de descodificación, los mensajes son pasados a lo largo de un borde, desde un nodo de restricción a un nodo variable, y viceversa.

Si bien la Fig. 1 ilustra el gráfico asociado a un código de longitud 10, puede apreciarse que representar el gráfico para una palabra de código de longitud 1000 sería 100 veces más complicado.

Una alternativa al uso de un gráfico para representar códigos es usar una representación matricial, tal como la mostrada en la Fig. 2. En la representación matricial de un código, la matriz H 202, usualmente denominada la *matriz de control de paridad*, incluye la conexión de bordes relevantes, información de nodos variables e información de nodos de restricción. En la matriz H, cada columna corresponde a uno de los nodos variables, mientras que cada fila corresponde a uno de los nodos columna. Dado que hay 10 nodos variables y 5 nodos de restricción en el código ejemplar, la matriz H incluye 10 columnas y 5 filas. La entrada de la matriz correspondiente a un nodo variable específico y a un nodo de restricción específico se fija en 1 si está presente un borde en el gráfico, es decir, si los dos nodos son vecinos; en caso contrario, se fija en 0. Por ejemplo, dado que el nodo variable V_1 está conectado con el nodo de restricción C_1 por un borde, se coloca un uno en el extremo superior izquierdo de la matriz 202. Sin embargo, el nodo variable V_4 no está conectado con el nodo de restricción C_1, por lo que se coloca un 0 en la cuarta posición de la primera fila de la matriz 202, indicando que los correspondientes nodos, variable y de restricción, 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 enumeración de los bordes por receptáculos, variables o de restricción, corresponde a la enumeración de los unos en H. La enumeración de receptáculos variables corresponde a la enumeración de arriba hacia abajo dentro de las columnas y al avance de izquierda a derecha entre columna y columna, según se muestra en la matriz 208. La enumeración de receptáculos de restricción corresponde a la enumeración de izquierda a derecha entre las filas y al avance de arriba hacia abajo entre fila y fila, según se muestra en la matriz 210.

En el caso de una representación matricial, la palabra de código X que ha de ser transmitida puede ser representada como un vector 206 que incluye los bits X_1 a X_n de la palabra de código a procesar. Una secuencia de bits X_1 a X_n es una palabra de código si y solo si el producto de las matrices 206 y 202 es igual a cero, es decir: $Hx = 0$.

En el contexto de la exposición de palabras de código asociadas a gráficos de LDPC, debería apreciarse que, en algunos casos, la palabra de código puede estar *punzada*. La punción es el acto de eliminar bits de una palabra de código para producir, en efecto, una palabra de código más corta. En el caso de gráficos de LDPC, esto significa que algunos de los nodos variables en el gráfico corresponden a bits que no son efectivamente transmitidos. Estos nodos variables y los bits asociados a ellos son denominados a menudo variables de estado. Cuando se usa la punción, el descodificador puede ser usado para reconstruir la parte de la palabra de código que no es físicamente comunicada por un canal de comunicaciones. Allí donde una palabra de código punzada es transmitida, el dispositivo receptor puede poblar inicialmente los valores (bits) faltantes de la palabra recibida con unos o ceros asignados, p. ej., de forma arbitraria, junto con una indicación (bit blando) de que estos valores no son fiables en absoluto, es decir, que estos valores están *borrados*. Con fines de explicación de la invención, supondremos que, cuando se usan, estos valores poblados por el receptor son parte de la palabra recibida que ha de ser procesada.

Considérese, por ejemplo, el sistema 350 mostrado en la Fig. 3. El sistema 350 incluye un codificador 352, un descodificador 357 y un canal 356 de comunicación. El codificador 352 incluye un circuito 353 de codificación que procesa los datos de entrada A para producir una palabra de código X. La palabra de código X incluye, con fines de detección y / o corrección de errores, alguna redundancia. La palabra de código X puede ser transmitida por el canal de comunicaciones. Alternativamente, la palabra de código X puede ser dividida mediante un dispositivo 354 de selección de datos en las partes X', X'' primera y segunda, respectivamente, por medio de alguna técnica de selección de datos. Una de las partes de la palabra de código, p. ej., la primera parte X', puede luego ser transmitida por el canal de comunicaciones a un receptor que incluye el descodificador 357, mientras la segunda parte X'' es punzada. Como resultado de las distorsiones producidas por el canal 356 de comunicaciones, partes de la palabra de código transmitida pueden perderse o corromperse. Desde la perspectiva del descodificador, los bits punzados pueden ser interpretados como perdidos.

En el receptor, los bits blandos son insertados en la palabra recibida para ocupar el lugar de bits perdidos o punzados. El borrado indicador insertado de X'' bits blandos indican y / o bits perdidos en la transmisión.

El descodificador 357 intentará reconstruir la palabra de código X completa a partir de la palabra Y recibida y de bits blandos insertados cualesquiera, y luego realizará una operación de descodificación de datos para producir A a partir de la palabra de código X reconstruida.

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

Se apreciará que las palabras recibidas generadas conjuntamente con la codificación de LDPC pueden ser procesadas realizando operaciones de descodificación de LDPC sobre las mismas, p. ej., operaciones de corrección y detección de errores, para generar una versión reconstruida de la palabra de código original. La palabra de código reconstruida puede luego ser sometida a la descodificación de datos para recuperar los datos originales que fueron codificados. El proceso de descodificación de datos, p. ej., puede ser seleccionar sencillamente un subconjunto específico de los bits a partir de la palabra de código reconstruida.

Las operaciones de descodificación de LDPC comprenden generalmente 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 descodificación de LDPC. La presente invención puede ser aplicada en el contexto de virtualmente cualquier algoritmo de paso de mensajes de ese tipo y, por lo tanto, puede ser usada en diversos sistemas de paso de mensajes, de los cuales los descodificadores de LDPC no son más que 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 propagación de creencia.

La propagación de creencia para códigos de LDPC (binarios) puede ser expresada de la siguiente manera.

Los mensajes transmitidos a lo largo de los bordes del gráfico son interpretados como logaritmos de probabilidades $\log \frac{p_0}{p_1}$ para el bit asociado al nodo variable. Aquí, (p_0, p_1) representa una distribución de probabilidad condicional sobre el bit asociado. Los bits blandos proporcionados al descodificador por el receptor también están dados en forma de un logaritmo de probabilidad. Así, los valores recibidos, es decir, los elementos de la palabra recibida, son logaritmos de probabilidad de los bits asociados, condicionados por la observación de los bits proporcionados por el canal de comunicación. En general, un mensaje m representa el logaritmo de probabilidad m, y un valor recibido y representa el logaritmo de probabilidad y. Para bits punzados, el valor y recibido se fija en 0, indicando $p_0 = p_1 = 1/2$.

Consideremos las reglas de paso de mensajes de la propagación de creencia. Los mensajes son indicados como m^{C2V} para mensajes desde nodos de verificación a nodos variables, y como m^{V2C} para mensajes desde nodos variables a nodos de verificación. Considérese un nodo variable con d bordes. Para cada borde j = 1, ..., d, sea $m^{C2V}(i)$ el mensaje entrante en el borde i. En el mismo comienzo del proceso de descodificación fijamos $m^{C2V} = 0$ para cada borde. Entonces, los mensajes salientes están dados por

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

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

$$m_p^{C2V}(j) = (\Sigma_{i=1}^{d} m_p^{V2C}(i)) - m_p^{V2C}(j),$$

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

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

donde definimos F(x) := log coth (x/2). (En ambas ecuaciones precedentes, el superíndice V2C indica los mensajes entrantes en el nodo de verificación). Observemos que F es su propia inversa, es decir, $F^{-1}(x) = F(x)$.

La mayoría de los algoritmos de paso de mensajes pueden ser vistos como aproximaciones a la propagación de creencia. Se apreciará que en cualquier implementación digital práctica, los mensajes comprenderán un número finito de bits y las reglas de actualización de mensajes, adecuadamente adaptadas.

Debería ser evidente que la complejidad asociada a la representación de códigos de LDPC para grandes palabras de código es intimidatoria, al menos para implementaciones de hardware que intenten explotar el paralelismo. Además, puede ser difícil implementar el paso de mensajes de una manera que pueda prestar soporte al procesamiento a altas velocidades.

A fin de hacer más práctico el uso de códigos de LDPC, existe la necesidad de procedimientos de representación de códigos de LDPC correspondientes a grandes palabras de código, de una manera eficaz y compacta, reduciendo por ello la cantidad de información requerida para representar el código, es decir, para describir el gráfico asociado. Además, existe la necesidad de técnicas que permitan que el paso de mensajes asociado a múltiples nodos y múltiples bordes, p. ej., cuatro o más nodos o bordes, sea realizado en paralelo de una manera fácilmente controlable, permitiendo por ello que incluso las grandes palabras de código sean eficazmente descodificadas en una cantidad razonable de tiempo. Hay necesidad adicional de una arquitectura de descodificador que sea bastante flexible para descodificar varios códigos distintos de LDPC. Esto es porque muchas aplicaciones requieren códigos de distintas longitudes y velocidades. Aún más deseable es una arquitectura que permita que la especificación del código específico de LDPC sea programable.

También se reclama atención a un artículo de BOUTILLION Emanuel et al., titulado "Descodificador – primer diseño de código" y publicado en los ANALES DEL SIMPOSIO INTERNACIONAL SOBRE TURBO-CÓDIGOS Y TEMAS AFINES, BREST, FRANCIA, 4 de septiembre de 2000, páginas 459 a 462, XP008011934. El artículo describe una arquitectura de descodificador para un código de Control de Paridad de Baja Densidad. En particular, se describe una arquitectura que usa N memorias distintas que son direccionadas por separado. En el sistema, N elementos son seleccionados aleatoriamente, cada uno para cada una de N memorias direccionadas por separado y aleatoriamente.

También se reclama atención a un artículo de Tong Zhang et al., titulado "Códigos de control de paridad de baja densidad (3,k)-regulares, orientados a la implementación en VLSI" y publicado en los ANALES DEL TALLER DEL IEEE SOBRE SISTEMAS DE PROCESAMIENTO DE SEÑALES, 26 de septiembre de 2001 al 28 de septiembre de 2001, páginas 25 a 36, XP010562749, ISBN: 0-7803-7145-3. El artículo describe que, en los pocos años recientes, los códigos de control de paridad de baja densidad (LDPC) de Gallager recibieron un montón de atención y que han sido dedicados muchos esfuerzos a analizar y mejorar sus prestaciones de corrección de errores. Sin embargo, poca consideración se ha dado a la implementación en VLSI del descodificador de LDPC. La arquitectura sencilla del descodificador, totalmente paralela, incurre usualmente en complejidad demasiado alta para muchos fines prácticos, y debería ser transformada en una realización parcialmente paralela. Lamentablemente, debido a la aleatoriedad de los códigos de LDPC, es casi imposible desarrollar una transformación efectiva para un código de LDPC arbitrariamente dado. Los autores del artículo proponen un enfoque de diseño conjunto de código y descodificador para construir una

clase de códigos de LDPC (3,*k*)-regulares que corresponda exactamente a una implementación de descodificador parcialmente paralelo, y que tenga muy buenas prestaciones. Además, para tales códigos de LDPC, los autores del artículo proponen un esquema de codificación sistemático y eficaz, explotando efectivamente la poca densidad de su matriz de control de paridad.

Breve descripción de las Figuras

La Figura 1 ilustra una representación gráfica bipartita de un código ejemplar de LDPC regular de longitud diez.

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

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

La Figura 4 es una representación gráfica bipartita de un código ejemplar de LDPC irregular.

La Figura 5, que comprende la combinación de las Figs. 5a a 5d, ilustra las etapas realizadas como parte de una operación de descodificación de LDPC de acuerdo al código de LDPC ilustrado en la Fig. 4.

La Figura 6 es una representación gráfica de un código pequeño de LDPC que es usado como la base de un código de LDPC mucho más grande, para presentar un ejemplo de acuerdo a la presente invención.

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

La Figura 8 ilustra cómo pueden ser dispuestos los bordes en el código mostrado en la Fig. 6, p. ej., enumerados en orden desde el sector de nodos variables, y cómo aparecerían los mismos bordes desde el sector de nodos de restricción.

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

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

La Figura 11 ilustra la representación matricial del control de paridad del gráfico de LDPC ilustrado en la Fig. 10.

La Figura 12 ilustra cómo pueden ser dispuestos los bordes en el código mostrado en la Fig. 11, p. ej., enumerados en orden desde el sector de nodos variables, y cómo aparecerán los mismos bordes desde el sector de nodos de restricción.

La Figura 13 ilustra el efecto de reemplazar las matrices identidad de dimensión 3x3 mostradas en la Fig. 11 por matrices de permutación cíclica, de acuerdo a una realización ejemplar de la presente invención.

La Figura 14 ilustra cómo pueden ser enumerados los bordes en el código mostrado en la Fig. 13, en orden desde el sector de nodos variables, y cómo aparecerán los mismos bordes desde el sector de nodos de restricción, después de haber sido sometidos a una permutación cíclica, de acuerdo a la invención.

La Figura 15 ilustra un descodificador de LDPC implementado de acuerdo a la presente invención, que vectoriza el descodificador de la Fig. 9.

Las Figuras 16 y 17 ilustran otros descodificadores de LDPC implementados de acuerdo a la presente invención.

Sumario de la invención

De acuerdo a la presente invención, se proporcionan un procedimiento y un aparato, según lo expuesto, respectivamente, en las reivindicaciones independientes. Las realizaciones preferidas de la invención se describen en las reivindicaciones dependientes.

La presente invención está orientada a procedimientos y aparatos para realizar operaciones de descodificación sobre palabras, usando técnicas de descodificación de paso de mensajes. Las técnicas de la presente invención están particularmente bien adaptadas para su uso con grandes códigos de LDPC, p. ej., palabras de código de longitudes mayores que 750 bits, pero pueden ser usadas también para longitudes más cortas. Las técnicas y aparatos de la presente invención también pueden ser usados para el diseño de gráficos y la descodificación allí donde se usan otros tipos de algoritmos de paso de mensajes. Con fines de explicación de la invención, sin embargo, se describirán descodificadores de LDPC y técnicas de descodificación ejemplares.

Las técnicas de la presente invención permiten la descodificación de gráficos de LDPC que poseen una cierta estructura jerárquica en la cual un gráfico completo de LDPC parece estar, en gran parte, compuesto por múltiples copias, digamos, Z, de un gráfico Z veces más pequeño. Las Z copias del gráfico pueden ser idénticas. Para ser

precisos, nos referiremos al gráfico más pequeño como el gráfico *proyectado*. La técnica puede ser óptimamente apreciada considerando primero un descodificador que descodifica Z pequeños gráficos idénticos de LDPC, síncronamente y en paralelo. Considérese un descodificador de paso de mensajes para un pequeño gráfico único de LDPC. El descodificador implementa una secuencia de operaciones correspondientes a un algoritmo de paso de mensajes. Considérese ahora aumentar el mismo descodificador de modo que descodifique Z gráficos idénticos de LDPC de ese tipo, síncronamente y en paralelo. Cada operación en el algoritmo de paso de mensajes es replicado Z veces. Obsérvese que la eficacia del proceso de descodificación ha mejorado porque, en total, la descodificación avanza Z veces más rápido y porque los mecanismos de control requeridos para controlar el proceso de paso de mensajes no necesitan ser replicados para las Z copias, sino que, en cambio, pueden ser compartidos por las Z copias. También podemos ver el anterior descodificador Z-paralelo como un descodificador *vectorial*. Podemos ver el proceso de hacer Z copias del gráfico más pequeño como la vectorización del gráfico más pequeño (proyectado): cada nodo del gráfico más pequeño se convierte en un *nodo vectorial*, que comprende Z nodos, cada borde del gráfico más pequeño se convierte en un *borde vectorial*, consistente en Z bordes, y cada mensaje intercambiado en la descodificación del gráfico más pequeño se convierte en un *mensaje vectorial*, que comprende Z mensajes.

La presente invención obtiene la eficacia de la vectorización anteriormente descrita, al modificarla de modo que el descodificador vectorial esté, de hecho, descodificando un gran gráfico, Z veces mayor que el gráfico proyectado. Esto se logra interconectando las Z copias del gráfico proyectado de forma controlada. Específicamente, dejamos que los Z bordes dentro de un borde vectorial se sometan a una permutación, o intercambio, entre copias del gráfico proyectado mientras van, p. ej., desde el sector de nodos variables al sector de nodos de restricción. En el proceso de paso de mensajes vectorizados correspondiente a los Z gráficos proyectados paralelos, este intercambio es implementado permutando mensajes dentro de un mensaje vectorial según es pasado desde un sector del gráfico vectorizado al otro.

Considérese la indización de los gráficos de LDPC proyectados con 1, j, ..., Z. En el descodificador estrictamente paralelo, los nodos variables en el gráfico j están conectados solamente con los nodos de restricción en el gráfico j. De acuerdo a la presente invención, tomamos un borde vectorial, incluyendo un correspondiente borde, cada uno de cada copia del gráfico, y permitimos una permutación dentro de los Z bordes, p. ej., permitimos que los receptáculos de restricción correspondientes a los bordes dentro del borde vectorial sean permutados, p. ej., reordenados. En adelante nos referiremos a menudo a las permutaciones, p. ej., reordenamientos, dentro de los bordes vectoriales como *rotaciones*.

De tal modo, de acuerdo a la presente invención, un gráfico relativamente grande puede ser representado, p. ej., descrito, usando relativamente poca memoria. Por ejemplo, un gráfico puede ser representado almacenando información que describe el gráfico proyectado e información que describe las rotaciones. Alternativamente, la descripción del gráfico puede ser realizada como un circuito que implementa una función que describe la conectividad del gráfico.

En consecuencia, la técnica de representación de gráficos de la presente invención facilita implementaciones de gráficos paralelas, p. ej., vectorizadas. Además, las técnicas de representación de gráficos de la presente invención pueden ser usadas para dar soporte a la descodificación 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 ser almacenada y proporcionada a un elemento de procesamiento de nodos vectoriales. Obsérvese que todos los nodos pertenecientes a un nodo vectorial tendrán el mismo grado, por lo que la información de grados se requiere solamente para un gráfico proyectado.

En diversas realizaciones, el descodificador se hace programable, permitiendo por ello que sea programado con múltiples descripciones de gráficos, p. ej., según lo expresado en términos de gráfico proyectado almacenado e información de rotación almacenada, o en términos de una función implementada. En consecuencia, los descodificadores de la presente invención pueden ser programados para descodificar un gran número de códigos distintos, p. ej., tanto regulares como irregulares. En algunas realizaciones específicas, el descodificador es usado para un gráfico fijo, o para grados fijos y esta información. En tales realizaciones la información de descripción de gráficos puede ser pre-programada o implícita. En tales casos, el descodificador puede ser menos flexible que las realizaciones programables, pero se ahorran los recursos requeridos para dar soporte a la programabilidad.

De acuerdo a una realización de la presente invención, se proporciona una memoria de mensajes que incluye filas de ubicaciones de memoria, correspondiendo cada fila a los mensajes asociados a una copia del gráfico proyectado.

Los mensajes correspondientes a los Z gráficos múltiples proyectados son apilados para formar columnas de Z mensajes por columna, y una columna de ese tipo corresponde a un mensaje vectorial. Esta disposición de memoria permite que los mensajes vectoriales, p. ej., el conjunto de Z mensajes, correspondiente a un borde vectorial, sean leídos de, o escritos en, la memoria como una unidad, p. ej., usando una instrucción SIMD para acceder a todos los Z mensajes en una columna en una operación. De tal modo, la memoria da soporte a la lectura y escritura de mensajes vectoriales como unidades. En consecuencia, la presente invención evita la necesidad de proporcionar una dirección distinta de lectura / escritura para cada mensaje individual en un conjunto de Z mensajes.

En uno o más puntos en el procesamiento del paso de mensajes, después de ser leídos en la memoria, los Z mensajes son sometidos a una operación de permutación, p. ej., una operación de reordenamiento. La operación de reordenamiento puede ser una operación de rotación, o una rotación, para abreviar. Estas operaciones de rotación corresponden a las rotaciones asociadas a los bordes vectoriales que interconectan las Z copias del gráfico proyectado para formar el gran gráfico único. Esta rotación puede ser aplicada, p. ej., antes de que los mensajes sean suministrados a un correspondiente procesador de nodos vectoriales (de restricción o variables). Alternativamente, la rotación puede ser aplicada a continuación del procesamiento por un procesador de nodos vectoriales.

La rotación puede ser implementada usando un sencillo dispositivo conmutador que conecta, p. ej., la memoria de mensajes con la unidad procesadora de nodos vectoriales y reordena esos mensajes según pasan desde la memoria a la unidad procesadora de nodos vectoriales. En una tal realización ejemplar, uno de los mensajes en cada mensaje vectorial leído en la memoria es suministrado a una unidad correspondiente entre las Z unidades paralelas de procesamiento de nodos, dentro de un procesador de nodos vectoriales, según lo determinado por la rotación aplicada al mensaje vectorial por el dispositivo conmutador. Una operación de rotación, según lo implementado por el dispositivo conmutador, puede también, o alternativamente, ser aplicada al mensaje vectorial antes de que sea escrito en la memoria y después del procesamiento del nodo.

La descripción almacenada o calculada del gráfico proyectado puede incluir, p. ej., información sobre el orden en el cual los mensajes en una fila correspondiente a un gráfico proyectado han de ser leídos de, y / o escritos en, la memoria durante el procesamiento de nodos de restricción y / o variables.

Los mensajes del gráfico grande entero son almacenados en múltiples filas, correspondiendo cada fila a una copia distinta del gráfico pequeño, estando las filas dispuestas para formar columnas de mensajes. Cada columna de mensajes representa un mensaje vectorial, al que se puede acceder como una única unidad. De ese modo, la información sobre cómo acceder a mensajes en una fila de un gráfico proyectado puede ser usada para determinar el orden en el cual se accede a los mensajes vectoriales correspondientes a múltiples copias del gráfico proyectado, de acuerdo a la presente invención.

La variación del orden en el cual los mensajes vectoriales son leídos de, y / o escritos en, la memoria de acuerdo a si la operación de lectura / escritura corresponde al procesamiento del sector de nodos variables, o al del sector de nodos de restricción, puede ser descrita como una primera permutación realizada sobre los mensajes. Esta permutación corresponde al intercalador asociado al gráfico proyectado. A fin de representar el gran gráfico del descodificador a partir del gráfico proyectado del descodificador, un segundo conjunto de información de permutación, p. ej., la información de rotación, es almacenado además de la información del orden de acceso a mensajes vectoriales (p. ej., columnas). La información de la segunda permutación (p. ej., la información de rotación), que representa información de control de conmutación, indica cómo deberían ser reordenados los mensajes en cada mensaje vectorial, p. ej., cada columna de mensajes, cuando, p. ej., son leídos de, y / o escritos en, la memoria. Esta permutación en dos etapas factoriza la permutación mayor, que describe el gráfico completo de LDPC, en dos partes implementadas mediante mecanismos distintos.

En una realización específica, se usa una permutación cíclica como la permutación de segundo nivel, debido a la facilidad con que una permutación de ese tipo puede ser implementada, y a la compacidad de su descripción. Este caso motiva el uso del término rotación para describir esta permutación de segundo nivel con fines de explicación. Sin embargo, ha de entenderse que la permutación de segundo nivel no necesariamente debe estar limitada a rotaciones, y que puede ser implementada usando otros esquemas de reordenamiento.

En diversas realizaciones de la presente invención, el descodificador genera salidas blandas de múltiples bits con un bit, p. ej., el bit de signo o paridad de cada salida blanda, correspondiente a una salida de decisión dura del descodificador, p. ej., la palabra de código original en el caso en que todos los errores han sido corregidos, o bien no está presente ningún error en la palabra recibida. La salida del descodificador, p. ej., la palabra de código recuperada, puede luego ser adicionalmente procesada para recuperar los datos originales que fueron usados en el tiempo de codificación para producir la palabra de código transmitida.

De acuerdo a una característica de la presente invención, las salidas blandas y / o duras producidas después de cada iteración completa del procesamiento de nodos variables son examinadas para determinar si las restricciones del control de paridad, indicativas de una palabra de código, son satisfechas por las actuales decisiones duras. Este proceso de verificación también goza de los beneficios de la estructura de permutación factorizada en dos etapas del gráfico. El proceso de descodificación iterativa (paso de mensajes) puede ser detenido una vez que se detecta la recuperación de una palabra de código de esta manera. En consecuencia, en el caso de señales relativamente libres de errores, la descodificación puede ser completada y detectada prontamente, p. ej., después de dos o tres iteraciones del proceso de descodificación del paso de mensajes. Sin embargo, en el caso de palabras recibidas que incluyen más errores, pueden ocurrir numerosas iteraciones del proceso de descodificación antes de que la descodificación tenga éxito o que el proceso se detenga debido a una restricción de expiración temporal.

La pronta detención de la descodificación exitosa, de acuerdo a la presente invención, admite un uso más eficaz de recursos en comparación con sistemas que adjudican un número fijo de iteraciones de descodificación a cada palabra recibida.

Dado que las técnicas de descodificación de la presente invención admiten que un gran número de operaciones de descodificación, p. ej., operaciones de procesamiento del descodificador de nodos de restricción y / o variables, sean realizadas en paralelo, los descodificadores de la presente invención pueden ser usados para descodificar palabras recibidas a altas velocidades. Además, dada la novedosa técnica de la presente invención, usada para representar grandes gráficos y / o para controlar el paso de mensajes para las operaciones de descodificación asociadas a tales gráficos, las dificultades del almacenamiento de las descripciones de grandes gráficos y del control del encaminamiento de sus mensajes son reducidas y / o superadas.

Ciertas generalizaciones de los códigos de LDPC y de las técnicas de descodificación de la invención incluyen la codificación / descodificación sobre alfabetos más grandes, no sencillamente bits, que tienen dos posibles valores, sino algún número mayor de posibilidades. Los códigos donde los nodos de restricción representan restricciones distintas a las restricciones del control de paridad también pueden ser descodificados usando los procedimientos y aparatos de la presente invención. Otras generalizaciones relevantes a las cuales puede ser aplicada la invención incluyen situaciones donde un algoritmo de paso de mensajes ha de ser implementado en un gráfico y se tiene la opción de diseñar el gráfico. Será evidente para los expertos en la técnica, a la vista de la presente solicitud de patente, cómo aplicar las técnicas de la presente invención a esas situaciones más generales.

Numerosas ventajas, características y aspectos adicionales de las técnicas de descodificación y los descodificadores de la presente invención serán evidentes a partir de la siguiente descripción detallada.

Descripción detallada de la invención:

Como se ha expuesto anteriormente, los procedimientos y aparatos de descodificación de la presente invención serán descritos con fines de explicación en el contexto de una realización de un descodificador de LDPC. Las etapas implicadas en la descodificación de un código de LDPC serán primero descritas con referencia a las Figs. 4 y 5, seguidas por una exposición más detallada de diversas características de la presente invención.

La Figura 4 ilustra un código ejemplar de LDPC irregular, usando un gráfico bipartito 400. El gráfico incluye m nodos 402 de verificación, n nodos variables 406 y una pluralidad de bordes 404. Los mensajes entre los nodos de verificación y los nodos variables son intercambiados sobre los bordes 404. Los bits blandos y_1 a y_n de entrada, correspondientes a la palabra recibida Y, y las salidas blandas (o duras) x_1 a x_n están indicados por el número de referencia 408. El m-ésimo nodo de verificación está identificado usando el número de referencia 402', y el n-ésimo nodo variable está identificado usando el número de referencia 406', mientras que la n-ésima entrada blanda y_n y la n-ésima salida blanda x_n están indicadas en la Fig. 4 usando, respectivamente, los números de referencia 410 y 409.

Los nodos variables 406 procesan mensajes provenientes de los nodos 402 de restricción, junto con los valores blandos de entrada provenientes de la palabra recibida y_1, ..., y_n, para actualizar el valor de las variables x_1, ..., x_n de salida, correspondientes a los nodos variables, y para generar mensajes para los nodos de restricción. Un mensaje es generado por un nodo variable para cada borde conectado con el nodo variable. El mensaje generado es transmitido a lo largo del borde desde el nodo variable hasta el nodo de restricción adosado el borde. Con fines de explicación, los mensajes desde nodos variables a nodos de restricción, cada tanto en la presente solicitud, serán indicados usando la abreviatura V2C, mientras que los mensajes desde nodos variables a nodos de restricción serán indicados usando la abreviatura C2V. Pueden ser añadidos índices a los componentes V y C de esta abreviatura, para indicar el nodo específico, entre los nodos variables y los nodos de restricción, que sirve como el origen / destino de un mensaje específico. Cada nodo 402 de restricción es responsable de procesar los mensajes recibidos desde los nodos variables mediante los bordes adosados al nodo de restricción específico. Los mensajes V2C recibidos desde los nodos variables son procesados por los nodos 402 de restricción para generar mensajes C2V, que son luego transmitidos de vuelta a lo largo de los bordes adosados a cada nodo de restricción. Los nodos variables 406 procesan luego los mensajes C2V, junto con los valores blandos de entrada, para generar y transmitir nuevos mensajes V2C, y generar salidas blandas, x_i. La secuencia de realización del procesamiento en los nodos variables 406 que comprende: transmitir mensajes generados a los nodos 402 de verificación, generar en los nodos variables salidas blandas x_i y recibir mensajes desde los nodos de verificación, puede realizarse repetidamente, es decir, iterativamente, hasta que las salidas x_i desde los nodos variables 406 indiquen que la palabra de código ha sido descodificada con éxito, o que algún otro criterio de detención, p. ej., la cumplimentación de un número fijo de iteraciones de paso de mensajes, haya sido satisfecho. Debería apreciarse que la secuencia de operaciones descritas anteriormente no necesariamente debe ocurrir estrictamente en el orden descrito. El procesamiento de nodos puede proceder asíncronamente y el procesamiento de nodos variables y de restricción puede ocurrir simultáneamente. No obstante, la lógica del proceso iterativo es como se ha descrito.

Los mensajes, V2C y C2V, pueden ser de uno o más bits, p. ej., de K bits cada uno, donde K es un valor entero

positivo distinto de cero. De manera similar, las salidas blandas x_i pueden tener uno o múltiples bits. Los mensajes y salidas de múltiples bits proporcionan la oportunidad de retransmitir información de confianza o fiabilidad en el mensaje o salida. En el caso de una salida (blanda) de múltiples bits, el signo del valor de salida blanda puede ser usado para proporcionar la salida dura de bit único del proceso de descodificación correspondiente a un nodo variable, p. ej., los bits de la palabra de código descodificada. Los valores blandos de salida pueden corresponder a valores blandos descodificados o, alternativamente, a la llamada información extrínseca (excluyendo la correspondiente información de entrada), que puede ser usada en otro proceso iterativo más amplio dentro del cual el descodificador de LDPC no es más que un módulo.

El proceso iterativo de paso de mensajes asociado a la descodificación de un código de LDPC será ahora expuesto adicionalmente con respecto a las Figs. 5a a 5d.

Al descodificar un código de LDPC, el procesamiento en cada nodo de restricción y variable puede ser realizado independientemente. En consecuencia, el procesamiento de nodos variables y / o de restricción puede ser realizado un nodo por vez, p. ej., en secuencia, hasta que alguno entre el procesamiento de nodos variables y de restricción, o ambos, haya(n) sido completado(s) para una iteración específica del proceso de descodificación. Esto permite que se proporcione y reutilice una única unidad de hardware de procesamiento, si se desea, para realizar el procesamiento asociado a cada uno de los nodos variables y / o de restricción. Otra característica significativa de la descodificación de LDPC es que los mensajes V2C y C2V usados durante una iteración específica de procesamiento no necesariamente deben haber sido generados al mismo tiempo, p. ej., durante la misma iteración de procesamiento. Esto admite implementaciones donde el procesamiento de nodos de restricción y variables puede ser realizado en paralelo sin importar cuándo fueron actualizados por última vez los mensajes utilizados. A continuación de un número suficiente de actualizaciones de mensajes e iteraciones, en donde todos los nodos variables y de restricción procesan los mensajes recibidos y generan mensajes actualizados, la salida (dura) de los nodos variables convergerá, suponiendo que el gráfico fuera debidamente diseñado y que no hay errores restantes no corregidos en la palabra recibida que está siendo procesada.

Dado que el procesamiento en cada nodo de verificación y nodo variable puede ser visto como una operación independiente, el procesamiento iterativo realizado en un único nodo C_n 502' ejemplar de verificación y un nodo variable V_n 506' será ahora expuesto en más detalle con referencia a las Figs. 5a a 5d. Con fines de descripción, pensaremos en los valores de mensajes, y los valores de entrada y salida blandas, como números. Un número positivo corresponde a una decisión de bit duro de 0 y un número negativo corresponde a una decisión de bit duro de 1. Las magnitudes más grandes indican una mayor fiabilidad. De tal modo, el número cero indica total falta de fiabilidad y el signo (positivo o negativo) es irrelevante. Esta convención es congruente con la práctica estándar, en la cual los valores blandos (mensajes, valores recibidos y de salida) representan logaritmos de probabilidad de los bits asociados, es decir, los valores blandos adoptan la forma

$$\log \frac{\textit{el bit de probabilidad es un } 0}{\textit{el bit de probabilidad es un } 1}$$

donde la probabilidad está condicionada por alguna variable aleatoria, p. ej., la observación física del bit proveniente del canal de comunicaciones en el caso de un valor recibido.

La Fig. 5a ilustra la etapa inicial en un proceso de descodificación de LDPC. Inicialmente, se suministra al nodo variable V_n 506' la entrada blanda, p. ej., los valores recibidos (1 o más bits) y_n desde una palabra recibida para ser procesada. Los mensajes C2V al comienzo de una operación de descodificación y la salida blanda X_n 509 son inicialmente fijados en cero. En base a las entradas recibidas, p. ej., los mensajes C2V de valor cero y la entrada y_n, el nodo variable V_n 506' genera un mensaje V2C para cada nodo de verificación con el cual está conectado. Habitualmente, en la etapa inicial, cada uno de estos mensajes será igual a y_n.

En la Fig. 5b, los mensajes V2C generados se muestran transmitidos a lo largo de cada uno de los bordes conectados con el nodo variable V_n 506'. De ese modo, los mensajes V2C actualizados son transmitidos a cada uno de los nodos 502 de verificación acoplados con el nodo variable V_n 506', incluyendo el nodo C_m 502' de verificación.

Además de generar los mensajes V2C, el procesamiento de nodos variables da como resultado la actualización de la salida blanda X_n 509' correspondiente al nodo variable que efectúa el procesamiento. La salida blanda X_n se muestra siendo actualizada en la Fig. 5c. Si bien se muestran como etapas distintas, la salida blanda puede ser emitida al mismo tiempo que son emitidos los mensajes V2C.

Como se expondrá adicionalmente más adelante, de acuerdo a algunas realizaciones de la presente invención, las salidas blandas (o sus decisiones duras asociadas) pueden ser usadas para determinar cuándo una palabra de código ha sido recuperada de la palabra recibida, es decir, cuándo las restricciones de paridad han sido satisfechas por los valores de salida. Esto indica una descodificación exitosa (aunque la palabra de código hallada puede ser incorrecta, es decir, no ser aquella que fue transmitida), permitiendo por ello que el proceso descodificador iterativo sea detenido

de manera oportuna, p. ej., antes de que se complete algún número máximo permitido y fijado de iteraciones de paso de mensajes.

El procesamiento de nodos de verificación puede ser efectuado una vez que un nodo de verificación, p. ej., el nodo C_m 502' de verificación recibe mensajes V2C a lo largo de los bordes con los cuales está conectado. Los mensajes V2C recibidos son procesados en el nodo de verificación para generar mensajes C2V actualizados, uno para cada borde conectado con el nodo de verificación específico. Como resultado del procesamiento de nodos de verificación, el mensaje C2V transmitido de vuelta a un nodo variable a lo largo de un borde dependerá del valor de cada uno de los mensajes V2C recibidos en los otros bordes conectados con el nodo de verificación, pero (usualmente y preferiblemente, pero no necesariamente) no del mensaje V2C recibido desde el nodo variable específico al cual está siendo transmitido el mensaje C2V. Así, los mensajes C2V son usados para transmitir información generada a partir de mensajes recibidos desde nodos variables, distintos al nodo al cual está siendo transmitido el mensaje.

La Fig. 5d ilustra el paso de mensajes C2V actualizados a nodos variables, incluyendo el nodo 506'. En particular, en la Fig. 5d, el nodo C_m 502' de restricción se muestra emitiendo dos mensajes C2V actualizados, con el mensaje C_m2V_n actualizado suministrado al nodo variable V_n 506'. El V_n 506' también recibe uno o más mensajes $C2V_n$ adicionales actualizados desde otro(s) nodo(s) de restricción con el cual, o con los cuales, está conectado.

Con la recepción de mensajes C2V actualizados, el procesamiento de nodos variables puede ser repetido para generar mensajes V2C actualizados y salidas blandas. Luego la actualización de mensajes C2V puede ser repetida, y así hasta que se satisfaga el criterio de detención del descodificador.

De tal modo, el procesamiento mostrado en las Figs. 5a a 5d será repetido después de la primera iteración, usando valores de mensajes actualizados, a diferencia de los valores iniciales, hasta que el proceso de descodificación se detenga.

La naturaleza iterativa del proceso de descodificación de LDPC, y el hecho de que el procesamiento en nodos individuales puede ser realizado independientemente del procesamiento en otros nodos, provee una buena medida de flexibilidad al implementar un descodificador de LDPC. Sin embargo, como se ha expuesto anteriormente, la mera complejidad de las relaciones entre los bordes y los nodos puede dificultar el almacenamiento de información de relaciones de bordes, p. ej., la descripción del gráfico. Lo que es incluso más importante, la complejidad del gráfico puede hacer que el paso de mensajes sea difícil de implementar en implementaciones paralelas, donde múltiples mensajes han de ser pasados al mismo tiempo.

Las implementaciones prácticas del descodificador de LDPC a menudo incluyen una memoria de bordes para almacenar mensajes pasados a lo largo de los bordes, entre nodos de restricción y / o variables. Además, incluyen un descriptor de gráficos denominado a veces un mapa de permutación, que incluye información que especifica conexiones de bordes, o apareo de receptáculos, definiendo por ello el gráfico de descodificación. Este mapa de permutación puede ser implementado como datos almacenados o como un circuito que calcula o implica la permutación. Además de la memoria de bordes, se necesitan una o más unidades de procesamiento de nodos para realizar el procesamiento efectivo asociado a un nodo.

Son posibles implementaciones en software del descodificador de LDPC, en donde el software es usado para controlar una CPU para funcionar como una unidad de procesamiento vectorial, y para controlar el paso de mensajes usando una memoria acoplada a la CPU. En implementaciones en software, una única memoria también puede ser usada para almacenar la descripción del gráfico del descodificador, y los mensajes de bordes, así como las rutinas del descodificador usadas para controlar la CPU.

Como se expondrá más adelante, en diversas realizaciones de la presente invención, pueden ser usadas una o más memorias de bordes. En una realización ejemplar de memoria de múltiples bordes, una primera memoria de bordes se usa para el almacenamiento y paso de mensajes C2V, y una segunda memoria de bordes se usa para el almacenamiento y paso de mensajes V2C. En tales realizaciones, múltiples unidades procesadoras de nodos, p. ej., una para realizar el procesamiento de nodos de restricción y otra para realizar el procesamiento de nodos variables, pueden ser, y a menudo son, empleadas. Como se expondrá más adelante, tales realizaciones admiten que las operaciones de procesamiento de nodos variables y de restricción sean realizadas en paralelo con la escritura de los mensajes resultantes en cada una de las dos memorias de mensajes, para su uso durante la próxima iteración del proceso de descodificación.

Presentaremos ahora un ejemplo sencillo de un pequeño gráfico de LDPC y su representación, que será usada posteriormente al explicar la invención. La exposición del gráfico de LDPC será seguida por una descripción de un descodificador de LDPC que puede ser usado para descodificar el gráfico pequeño.

La Fig. 6 ilustra un sencillo código irregular de LDPC en forma de un gráfico 600. El código tiene longitud cinco, según lo indicado por los 5 nodos variables V_1 a V_5 602. Cuatro nodos C_1 a C_4 606 de verificación están acoplados con los

nodos variables 602, por un total de 12 bordes 604, sobre los cuales pueden ser pasados los mensajes.

La Fig. 7 ilustra, usando las matrices 702, 702, el código de LDPC mostrado en la Fig. 6, en forma de matriz de control de paridad. Como se ha expuesto anteriormente, los bordes están representados en la matriz H 702 de permutación usando unos. El bit X_i está asociado al nodo variable V_i. Las matrices 706 y 706 muestran los unos en H, correspondientes a bordes en el gráfico, indizados, respectivamente, de acuerdo al orden de receptáculos variables y al orden de receptáculos de restricción.

Con fines de explicación, los 12 bordes serán enumerados a partir del sector de nodos variables, es decir, de acuerdo a sus receptáculos variables. Las conexiones establecidas por los bordes entre los nodos variables 602 y los nodos 606 de verificación pueden ser vistas en la Fig. 6. Con fines de exposición, los bordes adosados a la variable V_1, que la conectan con los nodos C_1, C_2 y C_3 de verificación, tienen asignadas las etiquetas 1, 2, 3, correspondientes a la enumeración de receptáculos variables. El nodo variable V_2 está conectado con los nodos C_1, C_3 y C_4 de verificación, por los bordes 4, 5 y 6, respectivamente. El nodo variable V_3 está acoplado con los nodos C_1 y C_4 de verificación por los bordes 7 y 8, respectivamente. Además, el nodo variable V_4 está acoplado con los nodos C_2 y C_4 de verificación por los bordes 9 y 10, respectivamente, mientras que el nodo variable V_5 está acoplado con los nodos C_2 y C_3 de verificación por los bordes 11 y 12, respectivamente. Esta indización corresponde a la matriz 706 de la Figura 7, es decir, al orden de receptáculos variables.

La Fig. 8 ilustra la relación entre los 12 bordes de la Fig. 6, según la enumeración desde el sector de nodos variables, en relación con los nodos variables y de verificación con los cuales están conectados. La fila 802 muestra los 5 nodos variables V_1 a V_5. Por debajo de las variables 802 se muestran los bordes 1 a 12 804, correspondientes a los receptáculos asociados, que están conectados con el nodo variable específico. Obsérvese que, dado que los bordes están ordenados desde el sector de nodos variables, en la fila 804 aparecen en orden desde 1 a 12. Supongamos que los mensajes están almacenados en memoria en el orden indicado en la fila 804.

Durante el procesamiento de nodos variables, se accede a los 12 mensajes de bordes en la memoria en secuencia, p. ej., en el orden mostrado en 804. Así, durante el procesamiento de nodos variables, los mensajes pueden ser sencillamente leídos en orden y suministrados a una unidad de procesamiento.

La fila 806 ilustra los cuatro nodos C1 a C4 de restricción presentes en el código de las Figs. 6 y 7. Obsérvese que los bordes están reordenados en la fila 804' para reflejar el orden en el cual están conectados con los nodos de restricción, pero la indización indicada es la inducida a partir del sector de nodos variables. En consecuencia, suponiendo que los mensajes de bordes están almacenados en orden a partir del sector de nodos variables, al realizar el procesamiento de nodos de restricción, los mensajes serían leídos en el orden ilustrado en la fila 804'. Es decir, durante el procesamiento de nodos de restricción, los mensajes serían leídos en la memoria en el orden 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10. Un módulo de ordenamiento de mensajes puede ser usado para emitir la secuencia correcta de información de acceso a mensajes de bordes, p. ej., ubicaciones de memoria, para leer datos de la memoria o escribir datos en la memoria durante las operaciones de procesamiento de nodos variables y de verificación.

Un descodificador 900 de LDPC en serie que realiza secuencialmente operaciones de procesamiento de mensajes, un borde a la vez, será ahora expuesto con respecto a la Fig. 9, y se expondrá la descodificación que usa el código ejemplar mostrado en la Fig. 6. El descodificador 900 de LDPC comprende un módulo 902 de control de descodificador, un módulo de ordenamiento de mensajes (memoria de permutación de receptáculos) 904, una memoria 910 de grados de nodos, una memoria 906 de bordes, un procesador 908 de nodos, un almacén temporal 916 de salida, una memoria 912 de decisión dura y un verificador 914 de control de paridad.

La memoria 906 de bordes incluye L ubicaciones de memoria de K bits, correspondiendo cada ubicación de K bits a un borde, y donde L es el número total de bordes en el gráfico de LDPC que se está usando, y K es el número de bits por mensaje intercambiado a lo largo de un borde. Para concretar, suponemos que los mensajes están almacenados en orden de acuerdo al ordenamiento de bordes inducido por los receptáculos variables. Así, para el gráfico ejemplar 600, los mensajes correspondientes a los bordes 1, 2, ..., 12 son almacenados en el orden indicado. La memoria 912 de decisión dura incluye L ubicaciones de memoria de 1 bit, correspondiendo cada ubicación de 1 bit a un borde. Esta memoria almacena decisiones duras transmitidas por los nodos variables a lo largo de cada uno de sus bordes, de modo que puedan ser verificadas las restricciones del control de paridad. El verificador 914 de control de paridad recibe las decisiones de bits duras según el procesador de nodos de verificación recibe mensajes. Los controles de paridad son verificados en el verificador de controles de paridad y, en el caso de que todas las verificaciones estén satisfechas, transmite una señal de convergencia al módulo 902 de control del descodificador.

El módulo 904 de ordenamiento de mensajes puede ser implementado como un mapa de permutación o tabla de consulta que incluye información que describe el ordenamiento de mensajes en la memoria de bordes, según se ve desde el sector de nodos variables, o según se ve desde el sector de nodos de restricción. Así, para nuestro gráfico ejemplar 600, la secuencia 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10, que especifica el orden de bordes, según se ve desde el sector de restricciones, sería efectivamente almacenada en el módulo de ordenamiento de mensajes. Esta secuencia

es usada para ordenar mensajes para el procesamiento de nodos de restricción y para ordenar decisiones duras leídas en la Memoria 912 de Decisiones Duras para el procesamiento por el verificador 914 del control de paridad.

En el descodificador de la Fig. 9, los mensajes correspondientes a un borde son tachados después de que son procesados por un procesador de nodos. De esta manera, la memoria de bordes alternará entre el almacenamiento de mensajes V2C y el almacenamiento de mensajes C2V. La verificación de decisiones duras ocurre durante el procesamiento de nodos de restricción, p. ej., según los mensajes V2C son leídos en la memoria 906 de mensajes de bordes.

La unidad 902 de control de descodificador es responsable de alternar el funcionamiento del descodificador entre las modalidades de funcionamiento del procesamiento de nodos variables y de verificación, para determinar cuándo debería detenerse el proceso de descodificación iterativa, p. ej., debido a la recepción de una señal de convergencia o al llegar a un total máximo permitido de iteraciones, para suministrar o controlar el suministro de información de grados a la unidad de procesamiento de nodos y al verificador del control de paridad, y para controlar el suministro de un índice de bordes al Módulo 904 de Ordenamiento de Mensajes. Durante el funcionamiento, el módulo 902 de control de descodificador transmite un índice de borde al módulo 904 de ordenamiento de mensajes. El valor, el índice de borde, es incrementado a lo largo del tiempo para que recorra en secuencia todos los bordes en el gráfico. Un índice de borde distinto, p. ej., único es usado para cada borde en un gráfico que está siendo implementado. En respuesta a cada índice de borde recibido, el módulo de ordenamiento de mensajes emitirá un identificador de borde, p. ej., información de dirección de memoria de bordes, seleccionando así la ubicación de la memoria de bordes a la que se accederá, p. ej., que será leída o escrita, en cualquier momento dado. Suponiendo un ordenamiento de receptáculos variables, el módulo 904 de ordenamiento de mensajes provocará que los mensajes sean leídos y escritos de vuelta en orden secuencial durante el procesamiento de nodos variables, y provocará que los mensajes sean leídos y escritos de vuelta en el orden correspondiente al ordenamiento de receptáculos de restricción durante el procesamiento de nodos de restricción. Así, en nuestro ejemplo anterior, los mensajes serán leídos y escritos de vuelta en el orden 1, 2, 3, ..., 12 durante el procesamiento de nodos variables y, simultáneamente, las decisiones duras serán escritas en la memoria 912 de decisiones duras en el orden 1, 2, 3, ..., 12. Durante el procesamiento de nodos de restricción, los mensajes serán leídos y escritos de vuelta en el orden 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10 y, simultáneamente, el módulo 904 de ordenamiento de mensajes hará que los bits de decisión dura sean leídos de la memoria 912 de decisión dura en el orden 1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10.

Según los mensajes son leídos de la memoria de bordes en respuesta al identificador de borde recibido desde el módulo 902 de control de descodificador, son suministrados al procesador 908 de nodos. El procesador 908 de nodos realiza la operación adecuada de procesamiento de nodos de restricción o variables, según la modalidad de operación, usando por ello los mensajes recibidos para generar mensajes actualizados correspondientes al nodo específico que está siendo implementado en cualquier momento dado. Los mensajes actualizados resultantes son luego escritos de vuelta en la memoria de bordes, sobrescribiendo los mensajes que hubieran sido recién leídos de la memoria. Los mensajes enviados a un nodo específico llegan al procesador de nodos como un bloque contiguo, es decir, uno detrás del otro. El módulo 902 de control de descodificador señaliza la delineación del nodo al procesador de nodos, p. ej., indicando el último mensaje correspondiente a un nodo, proporcionando por ello información de grado del nodo. En el caso del gráfico ejemplar 600, los grados de nodos variables serían especificados, p. ej., como la secuencia (3, 3, 2, 2, 2) y los grados de nodos de restricción serían especificados, p. ej., como la secuencia (3, 3, 3, 3). Esta información puede ser almacenada en la memoria 910 de grados de nodos, que sería luego leída por el módulo 902 de control de descodificador según itera sobre los índices de borde. Alternativamente, la información de grados puede ser pre-programada en cada una de las unidades de procesamiento de nodos. Esto puede ser preferible, p. ej., cuando se sabe de antemano que los grados de nodos serán uniformes, es decir, que el gráfico será regular.

El verificador 914 del control de paridad funciona de prácticamente la misma manera que un procesador de nodos de verificación, excepto en que los mensajes entrantes son bits individuales, no se calcula ningún mensaje saliente y el cálculo interno es más sencillo.

Durante el funcionamiento de la modalidad de nodos variables, los cálculos de nodos variables serán realizados un nodo a la vez por la unidad de procesamiento de nodos, hasta que el procesamiento, p. ej., la actualización de mensajes y las operaciones de generación de valores blandos de salida, asociadas a cada uno de los nodos variables, haya sido completado. Los mensajes son entregados al procesador 908 de nodos en el orden del sector de nodos variables, de modo que todos los mensajes correspondientes a un nodo lleguen en secuencia al procesador 908 de nodos. Con una iteración completada del procesamiento de nodos variables, el módulo 902 de control de descodificador hace que el descodificador 900 conmute a la modalidad de nodo de restricción de la operación de procesamiento. En respuesta al cambio en la señal de control C / V, la unidad 908 de procesamiento de nodos conmuta desde una modalidad de procesamiento de nodos variables a una modalidad de procesamiento de nodos de restricción. Además, el módulo 904 de ordenamiento de mensajes conmuta a una modalidad en la cual los identificadores de mensajes serán suministrados a la memoria de bordes en el orden de los receptáculos de restricción. Una o más señales de control, enviadas por la línea de control C / V, pueden ser usadas para controlar la conmutación

entre las modalidades de funcionamiento del procesamiento de nodos de restricción y variables.

Según el circuito 902 de control de descodificador controla al descodificador para realizar el procesamiento de nodos de restricciones, en la secuencia de los nodos de restricciones, un nodo a la vez, los mensajes almacenados en la memoria de bordes serán actualizados una vez más, esta vez por los mensajes C2V generados por el procesamiento de los nodos de restricciones. Cuando el procesamiento asociado al conjunto completo de nodos de restricción haya sido completado, el circuito 902 de control de descodificador conmutará de nuevo a la operación de la modalidad de procesamiento de nodos variables. De esta manera, el descodificador 900 alterna entre el procesamiento de nodos variables y nodos de restricción. Según lo descrito, el procesamiento se realiza secuencialmente, un nodo a la vez, hasta que el circuito 902 de control de descodificador determine que la operación de descodificación ha sido completada.

El sistema de descodificación de LDPC, escalar o secuencial, ilustrado en la Fig. 9 puede ser implementado usando relativamente poco hardware. Además, se presta bien a la implementación de software. Lamentablemente, la naturaleza secuencial del procesamiento realizado tiende a dar como resultado una implementación del descodificador relativamente lenta. En consecuencia, si bien la arquitectura escalar mostrada en la Fig. 9 tiene algunos atributos dignos de mención, tiende a ser inadecuada para aplicaciones de alto ancho de banda, tales como las comunicaciones ópticas o el almacenamiento de datos, donde se desea alta velocidad de descodificación y el uso de grandes palabras de código.

Antes de presentar descodificadores para descodificar grandes gráficos de LDPC vectorizados, expondremos conceptos y técnicas generales referidas a características de vectorización de gráficos de la presente invención. La exposición de la vectorización será seguida por una presentación de descodificadores ejemplares de LDPC vectorizado que realizan la presente invención.

Con fines de obtener la comprensión de la vectorización de gráficos de LDPC, consideremos un código 'pequeño' de LDPC con matriz H de control de paridad. El gráfico pequeño, en el contexto de un gráfico vectorizado más grande, será mencionado como el *gráfico proyectado*. Sea Ψ un subconjunto de matrices de permutación de dimensiones Z x Z. Suponemos que las inversas de las permutaciones en Ψ también están en Ψ. Dado el gráfico pequeño, proyectado, podemos formar un gráfico de LDPC Z veces más grande, reemplazando cada elemento de H por una matriz de dimensiones Z x Z. Los elementos 0 de H son reemplazados por la matriz cero, indicada como 0. Cada uno de los elementos 1 de H es reemplazado por una matriz de Ψ. De esta manera, 'elevamos' un gráfico de LDPC a uno Z veces más grande. La complejidad de la representación comprende, a grandes rasgos, el número de bits requeridos para especificar las matrices de permutación $|E_H| \log|\Psi|$ más la complejidad requerida para representar H, donde $|E_H|$ indica los números 1 en H y $|\Psi|$ indica el número de permutaciones distintas en Ψ. P. ej., si Ψ es el espacio de permutaciones cíclicas, entonces $|\Psi| = Z$. En la práctica, podríamos tener, p. ej., $Z = 16$ para $n \approx 1.000$.

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

Ejemplo: Elevando una pequeña matriz de control de paridad, los σ_i, i = 1, ..., 16, son elementos de Ψ mostrados aquí indizados en orden de receptáculo variable proyectado.

El subconjunto Ψ, en general, puede ser escogido usando diversos criterios. Una de las motivaciones principales para la estructura anterior es simplificar la implementación en hardware de los descodificadores. Por lo tanto, puede ser ventajoso restringir Ψ a permutaciones que puedan ser eficazmente implementadas en hardware, p. ej., en una red de conmutación.

Las topologías de redes de conmutación paralela es un tema bien estudiado con relación a las arquitecturas de multiprocesadores y conmutadores de comunicación de alta velocidad. Un ejemplo práctico de una arquitectura adecuada para el subconjunto Ψ de permutaciones es una clase de redes de conmutación de múltiples capas, que incluyen, p. ej., redes omega (mezcla perfecta) / delta, redes desplazadoras logarítmicas, etc. Estas redes ofrecen una razonable complejidad de implementación y una frondosidad suficiente para el subconjunto Ψ. Adicionalmente, las redes conmutadoras de múltiples capas se ajustan bien a escala, p. ej., su complejidad aumenta según N log N, donde N es el número de entradas a la red, lo que las hace especialmente adecuadas para descodificadores de LDPC masivamente paralelos. Alternativamente, en descodificadores de la presente invención con niveles relativamente bajos de paralelismo y un Z pequeño, el subconjunto Ψ de permutaciones puede ser implementado en una única capa.

Se dice que un gráfico de LDPC tiene "múltiples bordes" si cualquier par de nodos está conectado por más de un borde. Un *borde múltiple* es el conjunto de bordes que conectan un par de nodos que están conectados por más de un

borde. Aunque es generalmente indeseable que un gráfico de LDPC tenga múltiples bordes, en muchos casos puede ser necesario, en la construcción de gráficos vectorizados, que el gráfico proyectado posea múltiples bordes. Se puede extender la noción de una matriz de control de paridad para permitir que las entradas de la matriz indiquen el *número* de bordes que conectan el par de nodos asociado. La definición de palabra de código es todavía la misma: el código es el conjunto de vectores x de 0 y 1 que satisface Hx = 0 módulo 2. Al vectorizar un gráfico proyectado con múltiples bordes, de acuerdo a la invención, cada borde dentro del borde múltiple es reemplazado por una matriz de permutación de Ψ y estas matrices son añadidas para producir la matriz extendida de control de paridad del código completo. De tal modo, un j > 1 en la matriz H de control de paridad del gráfico proyectado será 'elevado' a una suma $\sigma_k + \sigma_{k+1} + ... + \sigma_{k+j-1}$ de matrices de permutación de Ψ. Habitualmente, se escogerán los elementos de la suma de modo que cada entrada de $\sigma_k + \sigma_{k+1} + ... + \sigma_{k+j-1}$ sea 0 o 1, es decir, el gráfico completo no tenga ningún borde múltiple.

La elevación descrita anteriormente parece tener una limitación. Según la construcción anterior, tanto la longitud del código como la longitud de la unidad de datos codificados deben ser múltiplos de Z. Esta aparente limitación es superada fácilmente, sin embargo. Supongamos que la unidad de datos a codificar tiene longitud A Z + B, donde A es un número entero positivo y B está entre 1 y Z inclusive, y la longitud de código deseada es C Z + 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 E Z >= C Z + D + (Z-B). Se puede diseñar un gráfico elevado que codifique una unidad de datos de longitud (A+1)Z, para producir una palabra de código de longitud E Z, de modo que la unidad de datos aparezca como parte de la palabra de código, y usar esto para producir los parámetros de código deseados según lo siguiente. 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 es codificada para producir una palabra de código de longitud EZ. Los Z-B ceros no son transmitidos. Entre los otros EZ – (Z-B) bits en la palabra de código se seleccionan EZ – CZ – D – (Z-B) bits y se punzan; obsérvese que el número de bits de punción está entre 0 y Z-1 inclusive. Estos bits no serán transmitidos, por lo que el número efectivo de bits transmitidos es EZ – (Z-B) – (EZ – CZ – D – (Z-B)) = CZ + D, que es la longitud de código deseada. El receptor, sabiendo de antemano sobre los ceros adicionales y los bits punzados, sustituye bits blandos en lugar de los bits punzados, indicando el borrado, y sustituye bits blandos en lugar de los bits cero conocidos, indicando un valor cero con la mayor fiabilidad posible. La palabra extendida recibida, de longitud EZ, puede ahora ser descodificada para recuperar la unidad de datos originales. En la práctica, habitualmente se hacen estos ajustes punzando bits de solamente un nodo vectorial y declarando los bits conocidos de solamente un nodo vectorial.

Diversas implicaciones del descodificador, que usan la técnica expuesta anteriormente de vectorización de gráficos de LDPC, serán abordadas ahora.

Según lo expuesto anteriormente, la descodificación del paso de mensajes de los códigos de LDPC implica pasar mensajes a lo largo de los bordes del gráfico que representa el código, y realizar cálculos basados en esos mensajes en los nodos del gráfico, p. ej., los nodos variables y de restricción.

Dado un gráfico de LDPC vectorizado, se puede vectorizar el proceso de descodificación de la siguiente manera. El descodificador opera como si estuviera descodificando Z copias del código de LDPC proyectado, síncronamente y en paralelo. El control del proceso de descodificación corresponde al gráfico de LDPC proyectado y puede ser compartido entre las Z copias. Así, describimos el descodificador como operando sobre mensajes vectoriales que recorren bordes vectoriales y que son recibidos por nodos vectoriales, teniendo cada vector Z elementos. Los receptáculos también devienen vectorizados. En particular, un procesador de nodos vectoriales podría comprender Z procesadores de nodos en paralelo y, cuando un vector de mensajes $(m_1, ..., m_Z)$ es entregado al procesador de nodos vectoriales, el mensaje m_i es entregado al i-ésimo procesador. De tal modo, no ocurre ningún encaminamiento ni reordenamiento dentro de un procesador de nodos vectoriales, es decir, el mensaje vectorial es alineado con el vector de procesadores de forma fija.

Una desviación de la ejecución paralela puramente disjunta de los Z gráficos proyectados es que los mensajes son reordenados dentro de un mensaje vectorial 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, los trayectos de procesamiento de las Z copias del gráfico proyectado se mezclan, enlazándolas por ello para formar un único gráfico 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 rotación puede ser especificada usando relativamente poca memoria.

Si bien diversas permutaciones pueden ser usadas para las rotaciones de acuerdo a la presente invención, el uso de permutaciones cíclicas es especialmente interesante, debido a la facilidad con la cual pueden ser implementadas tales permutaciones. Para simplificar, supondremos ahora que Ψ comprende el grupo de permutaciones cíclicas. En este caso, nuestros grandes gráficos de LDPC están restringidos a tener una estructura cuasi-cíclica. Para los fines de este ejemplo, sea *N* el número de nodos variables en el gráfico y sea *M* el número de nodos de restricción en el gráfico. Primero, suponemos que tanto *N* como *M* son múltiplos de *Z*, $N = nZ$ y $M = mZ$, donde Z indicará el orden del ciclo.

Veamos que los nodos están doblemente indizados. Así, el nodo variable $v_{i,j}$ es el j-ésimo nodo variable de la i-ésima copia del gráfico proyectado. Dado que Ψ es el grupo de permutaciones cíclicas, el nodo variable $v_{i,j}$ está conectado a

un nodo $c_{a,b}$ de restricción si y solo si el nodo variable $v_{i+k \bmod Z,j}$ está conectado con un nodo $c_{a+k \bmod Z,\ b}$ de restricción, para k = 1, ..., Z.

Las técnicas de la presente invención para representar un gráfico grande usando una representación de un gráfico mucho más pequeño, e información de rotación, serán ahora explicadas adicionalmente con referencia a las Figs. 10 a 16, que se refieren a la vectorización del gráfico 600. Las técnicas descritas con referencia a estas figuras pueden ser aplicadas a gráficos de LDPC mucho más grandes.

De acuerdo a la presente invención, un gráfico más grande puede ser generado replicando, es decir, implementando múltiples copias, del gráfico pequeño mostrado en la Fig. 6, y realizando luego 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 hacer 3 copias paralelas del gráfico pequeño ilustrado en la Fig. 6. Los nodos variables 602', 602'' y 602''' corresponden, respectivamente, a los gráficos primero a tercero, resultantes de hacer tres copias del gráfico de la Fig. 6. Además, los nodos 606', 606'' y 606''' de verificación corresponden, respectivamente, a los gráficos primero a tercero, resultantes de hacer las tres copias. Obsérvese que no hay ningún borde que conecte nodos de uno de los tres gráficos con nodos de otro de los tres gráficos. En consecuencia, este proceso de copia, que "eleva" el gráfico básico en un factor de 3, da como resultado tres gráficos idénticos disjuntos.

La Fig. 11 ilustra el resultado del proceso de copia expuesto anteriormente usando las matrices 1102 y 1104. Obsérvese que para hacer tres copias del gráfico original, cada elemento distinto de cero en la matriz 702 es reemplazado por una matriz identidad de dimensiones 3x3. Así, cada uno en la matriz 702 es reemplazado por una matriz de dimensiones 3x3 que tiene unos a lo largo de la diagonal y ceros en todos los demás lugares, para producir la matriz 1102. Obsérvese que la matriz 1102 tiene 3 veces el número de bordes que tenía la matriz 702, 12 bordes para cada una de las 3 copias del gráfico básico mostrado en la Fig. 6. Aquí, la variable x_{ij} corresponde al nodo variable v_{ij}.

La Fig. 12 muestra la relación entre los (3x12) 36 bordes, los (3x5) 15 nodos variables y los (3x4) 12 nodos de restricción, que componen el gráfico 1000. Como en el caso de la Fig. 8, los bordes están enumerados desde el sector de nodos variables.

Con fines de notación, el primer número usado para identificar un nodo, restricción o borde indica la copia del gráfico a la cual pertenece el borde, p. ej., la primera, segunda o tercera copia del gráfico. El segundo número es usado para identificar el número de elemento dentro de la copia particular especificada del gráfico básico.

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

Obsérvese que las filas 1202', 1202'', 1202''' de bordes son sencillamente copias de la fila 804 que representa tres copias de la fila de bordes 804, mostrada en la Fig. 8, según se refieren a los nodos variables. De manera similar, las filas 1204', 1204'' y 1204''' de bordes representan tres copias de la fila de bordes 804' mostrada en la Fig. 8, según se refieren a los nodos de restricción.

Expongamos ahora brevemente cómo modificar el descodificador 900 de la Fig. 9 para descodificar los Z = 3 gráficos paralelos ahora definidos. El procesador 908 de nodos se convertirá en un procesador de nodos vectoriales, capaz de procesar 3 nodos idénticos simultáneamente en paralelo. Todas las salidas desde el procesador 908 de nodos serán vectorizadas, llevando por ello 3 veces el tiempo que llevaba anteriormente. La memoria 912 de decisiones duras y la memoria 906 de mensajes de bordes pasará a ser 3 veces más amplia, cada una de ellas capaz de escribir o leer 3 unidades (bits o mensajes de K bits, respectivamente) en paralelo, usando la indicación de una única instrucción SIMD. Las salidas de estas memorias serán ahora vectores, 3 veces más amplios que antes. El verificador 914 del control de paridad y el almacén temporal 916 de salida también serán adecuadamente vectorizados, con todo el procesamiento paralelizado adecuadamente.

Consideremos ahora la introducción de rotaciones en nuestro ejemplo. Esto puede ser ilustrado reemplazando cada una de las tres matrices identidad de dimensiones 3x3 en la Fig. 11 por matrices de permutación cíclica de dimensiones 3x3, según se muestra en la Fig. 13. Obsérvese que hay tres posibilidades para la matriz de permutación cíclica usada en la Fig. 13. Es posible indicar la matriz específica de permutación a ser sustituida en lugar de una matriz identidad, indicando si la matriz de permutación tiene o no un "1" situado 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 el extremo superior izquierdo y avanzando hacia la esquina inferior derecha (orden de receptáculos vectoriales de restricción), las rotaciones podrían ser especificadas por la secuencia (2, 2, 3, 3, 1, 1, 1, 3, 2, 1, 3, 2).

La Fig. 14 ilustra el efecto de realizar la permutación cíclica (rotación) en el sector de nodos de restricción. Dado que la permutación es realizada desde el sector de nodos de restricción, la relación entre los bordes, p. ej., el ordenamiento,

desde el sector de nodos variables permanece sin cambios, según se muestra en las filas 1402', 1402'' y 1402'''. Desde el sector de restricciones, sin embargo, la permutación da como resultado que los bordes dentro de una columna, p. ej., los bordes dentro de un borde vectorial específico, sean reordenados según se muestra en las filas 1404', 1404'' y 1404'''. Esto produce interconexiones entre nodos correspondientes a distintas copias del gráfico proyectado.

Consideremos, por ejemplo, la columna 1 de las filas 1404 en relación con la columna 1 de las filas 1104 de la Fig. 11. Obsérvese que, como resultado de la operación de permutación vectorial de bordes, el nodo $C_{1,1}$ de restricción está ahora conectado con el borde (2,1), en lugar del borde (1,1), el nodo $C_{2,1}$ de restricción está acoplado con el borde (3,1), en lugar del borde (2,1), y el nodo $C_{3,1}$ de restricción está acoplado con el borde (1,1), en lugar del borde (3,1).

Hemos expuesto anteriormente cómo vectorizar el descodificador 900 para descodificar Z copias paralelas del gráfico proyectado. Introduciendo conmutadores en los trayectos de mensajes para realizar rotaciones, descodificamos el código de LDPC definido en la Fig. 13.

La Figura 15 ilustra un descodificador que incorpora diversas características de la presente invención. El descodificador 1500 vectoriza completamente, con rotaciones, el descodificador 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, los valores de Z de la forma 2^k, para k entero, son a menudo preferibles. Las similitudes con el descodificador 600 son evidentes. En particular, el módulo 1502 de control de descodificador y la memoria 1501 de grados de nodos funcionan de la misma manera, o de manera similar, que sus respectivas contrapartidas 902 y 910 en el descodificador 900. Por ejemplo, para descodificar el código de LDPC definido en las Figs. 13 y 14, el funcionamiento de estos componentes sería exactamente el mismo que el de sus contrapartidas en el descodificador 900 al descodificar el gráfico ejemplar 600. La memoria 1506 de mensajes de bordes y la memoria 1512 de decisiones duras son versiones vectorizadas de sus contrapartidas 906 y 912 en el descodificador 900. Mientras que en el descodificador 900 las memorias almacenaban unidades individuales (mensajes de K bits, o bits), las correspondientes memorias en el descodificador 1500 almacenan conjuntos, es decir, vectores y mensajes, dando como resultado, p. ej., que sean almacenados mensajes de Z x K bits. Estos vectores son escritos o leídos como unidades individuales usando instrucciones SIMD. De tal modo, los identificadores de mensajes enviados a estos módulos desde el módulo 1504 de ordenamiento de mensajes son equivalentes o similares a los del descodificador 900. El módulo 1504 de ordenamiento de mensajes tiene el papel adicional, más allá de lo que su contrapartida 904 tenía en el descodificador 900, de almacenar y proporcionar la información de permutación, p. ej., de rotación. Recuérdese que en el ejemplo 600 de descodificación, el descodificador 900 almacenaba en su módulo 904 de ordenamiento de mensajes la secuencia de bordes (1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10). Consideremos usar el descodificador 1500 para descodificar el código de la Fig. 13 y 14. El módulo 1504 de ordenamiento de mensajes almacenaría la misma secuencia anterior para acceder a vectores de mensajes durante el procesamiento de 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 a la misma secuencia de mensajes vectoriales. Esta secuencia sirve como base para generar la señal de rotación que es usada por el módulo 1504 de ordenamiento de mensajes para hacer que los conmutadores 1520 y 1522 roten, respectivamente, mensajes vectoriales y bits vectoriales de decisión dura. (Obsérvese que los bits de decisión dura son proporcionados solamente durante la modalidad de procesamiento de nodos variables). El verificador vectorial 1514 del control de paridad es una versión vectorial de su contrapartida 914 en el descodificador 900. Obsérvese que la señal de convergencia de salida es un escalar, como antes. El almacén temporal 1516 de salida sirve para el mismo fin que el almacén temporal 916, pero los datos de salida son escritos como vectores. El procesador 1508 de nodos vectoriales es, p. ej., Z procesadores de nodos, cada uno como 908, en paralelo. Estos nodos compartirían las señales de grados y la señal de control C / V desde el módulo 1502 de control de descodificador.

A fin de facilitar la capacidad de emitir decisiones del descodificador, ya sean blandas o duras, las decisiones blandas generadas por la unidad de procesamiento variable son suministradas a una entrada de decisiones blandas del almacén temporal 1516. Así, en cualquier momento antes de completar la descodificación, las decisiones blandas pueden ser obtenidas desde la salida del almacén temporal 1516.

Consideremos además ahora cómo el descodificador 1500 funcionaría descodificando el ejemplo de las Figs. 13 y 14. Inicialmente, la memoria 1506 de mensajes de bordes está poblada con ceros. El módulo 1502 de control de descodificador alterna primero a la modalidad de procesamiento de nodos variables. Los vectores de la memoria 1506 de mensajes de bordes (todos ceros en este momento) son leídos en orden y entregados al procesador 1508 de nodos vectoriales para el procesamiento de nodos vectoriales. El procesador 1508 de nodos vectoriales emite luego los valores recibidos solo a lo largo de cada borde desde un nodo variable; usaremos y para indicar estos primeros mensajes, para indicar esto. Así, los vectores salientes serían $(y_{1,i}, y_{2,i}, y_{3,i})$ para i = 1, ..., 12 en orden creciente. La señal de rotación es usada para controlar el reordenamiento de mensajes realizado por los circuitos 1520, 1522 de conmutación. La señal de rotación desde el módulo 1502 de ordenamiento de mensajes hará que los mensajes en los vectores sean rotados para producir vectores procesados de la siguiente manera: $(y_{2,1}, y_{3,1}, y_{1,1})$, $(y_{3,2}, y_{1,2}, y_{2,2})$, $(y_{1,3}, y_{2,3}, y_{3,3})$, $(y_{2,4}, y_{3,4}, y_{1,4})$, $(y_{3,5}, y_{1,5}, y_{2,5})$, $(y_{1,6}, y_{2,6}, y_{3,6})$, $(y_{3,7}, y_{1,7}, y_{2,7})$, $(y_{2,8}, y_{3,8}, y_{1,8})$, $(y_{1,9}, y_{2,9}, y_{3,9})$, $(y_{3,10}, y_{1,10}, y_{2,10})$,

($y_{1,11}$, $y_{2,11}$, $y_{3,11}$), ($y_{2,12}$, $y_{3,12}$, $y_{1,12}$). Una vez que los vectores procesados son escritos en la memoria 1506 de bordes, en el orden indicado, el módulo 1502 de control de descodificador alternará a la modalidad de restricción. Los mensajes vectoriales almacenados serán entonces leídos en el orden (1, 4, 7, 2, 9, 11, 3, 5, 12, 6, 8, 10). Así, serán presentados al procesador 1508 de nodos vectoriales en el orden ($y_{2,1}$, $y_{3,1}$, $y_{1,1}$), ($y_{2,4}$, $y_{3,4}$, $y_{1,4}$), ($y_{3,7}$, $y_{1,7}$, $y_{2,7}$), ($y_{3,2}$, $y_{1,2}$, $y_{2,2}$), ($y_{1,9}$, $y_{2,9}$, $y_{3,9}$), ($y_{1,11}$, $y_{2,11}$, $y_{3,11}$), ($y_{1,3}$, $y_{2,3}$, $y_{3,3}$), ($y_{3,5}$, $y_{1,5}$, $y_{2,5}$), ($y_{2,12}$, $y_{3,12}$, $y_{1,12}$), ($y_{1,6}$, $y_{2,6}$, $y_{3,6}$), ($y_{2,8}$, $y_{3,8}$, $y_{1,8}$), ($y_{3,10}$, $y_{1,10}$, $y_{2,10}$). El procesador 1508 de nodos vectoriales está implementado como tres (Z = 3) procesadores de nodos en paralelo. El primer elemento (mensaje) de cada vector de mensajes (conjunto de mensajes) es entregado al primer procesador de nodos; el segundo mensaje es entregado al segundo procesador; y el tercer mensaje es entregado al tercer procesador, respectivamente. La señal de grado, que indica el grado del nodo actual que está siendo procesado, es suministrada por la memoria 1510 de grados a los tres procesadores paralelos del procesador 1508 de nodos vectoriales. En este momento, la señal de grado indica que las restricciones son (todas) de grado 3, por lo 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. De manera 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 $y_{i,j}$ entrante. Según los vectores están emergiendo del procesador 1508 de nodos vectoriales, la señal de rotación hacia el conmutador 1520 hará que los vectores sean reordenados de modo que se invierta la rotación previa, y por tanto lleguen a la memoria de bordes 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 son reescritos en la memoria de acuerdo al orden de los identificadores de mensajes en el cual fueron leídos, por lo que, después de la escritura, aparecen en la memoria como ($m_{1,j}$, $m_{2,j}$, $m_{3,j}$), en orden j = 1, ..., 12. El módulo 1504 de ordenamiento de mensajes alterna ahora a la modalidad de procesamiento de nodos variables, en respuesta a una señal C / V suministrada por el módulo 1502 de control de descodificador. Los vectores de mensajes son luego leídos en el orden j = 1, ..., 12, y entregados al procesador 1508 de nodos vectoriales para el procesamiento de nodos variables. Esto completa una iteración.

Durante el procesamiento de nodos variables, el procesador 1508 de nodos vectoriales también emite vectores descodificados blandos, que son almacenados en el almacén temporal 1516 de salida. También emite decisiones duras que son suministradas al circuito 1522 de conmutación. Los vectores de decisión dura de un bit se someten a la misma operación de rotación que los vectores de mensajes en los momentos correspondientes. Los vectores rotados de decisión dura, producidos por el circuito 1522 de conmutación, son luego dispuestos en la memoria 1512 de decisiones duras, donde son almacenados. Como resultado de aplicar la misma rotación aplicada a los vectores de mensajes, las decisiones duras pueden ser leídas en el mismo orden en que son leídos los mensajes vectoriales durante el procesamiento de nodos de restricción. Durante el procesamiento de nodos de restricción, las decisiones duras son entregadas al verificador vectorial 1512 de control de paridad, que realiza Z controles de paridad en paralelo. Si todos los controles de paridad son satisfechos, entonces la señal de convergencia, CON, es generada y emitida. En respuesta a la recepción de la señal de convergencia que indica una descodificación exitosa, el módulo 1502 de control de descodificador detiene el proceso de descodificación.

Debe ser evidente que hay muchas variaciones para el descodificador 1500 que realizan, cada una, la invención actual. Por ejemplo, el conmutador 1520 podría haber sido colocado, en cambio, a lo largo del trayecto de datos entre la memoria 1506 de mensajes de bordes y el procesador 1508 de nodos vectoriales. De manera similar, el conmutador 1522 podría haber sido colocado, en cambio, a lo largo del trayecto de datos entre la memoria 1512 de decisiones duras y el verificador vectorial 1514 de control de paridad. Tal reemplazo también implicaría el ajuste adecuado de la temporización de la señal de rotación. La memoria 1512 de decisiones duras, el verificador vectorial 1514 de control de paridad y los trayectos asistentes de datos no necesariamente deben ser usados, y son eliminados para realizaciones del descodificador que realicen un número fijo de iteraciones y, por lo tanto, no requieran detección de convergencia. Muchas variaciones adicionales serán evidentes para los expertos en la técnica, a la vista de la presente invención.

La Fig. 16 ilustra un descodificador 1600 que está implementado de acuerdo a otra realización de la presente invención. El descodificador 1600 incluye muchos elementos que son los mismos que, o similares a, los elementos del descodificador 1500. En consecuencia, con fines de brevedad, tales elementos serán identificados usando los mismos números de referencia usados en la Fig. 15, y no serán expuestos nuevamente en detalle. El descodificador 1600 es capaz de realizar operaciones de procesamiento, tanto de nodos variables como de nodos de restricción, p. ej., operaciones de actualización de mensajes, al mismo tiempo, p. ej., simultáneamente e independientemente. A diferencia de la implementación del descodificador de la Fig. 15, que puede ser descrito como un descodificador de sector a sector, debido a la manera en que alterna entre las iteraciones de procesamiento de nodos variables y de nodos de restricción, el descodificador 1600 puede ser descrito como un descodificador de iteración asíncrona, dado que las operaciones de procesamiento de nodos variables y de nodos de restricción pueden ser realizadas independientemente, p. ej., simultáneamente.

El circuito descodificador 1600 incluye un módulo 1602 de control de descodificador, un módulo 1604 de ordenamiento de mensajes, un primer circuito conmutador 1621, una memoria 1606 de mensajes de bordes V2C, un procesador vectorial de nodos de restricción (p. ej., Z procesadores de nodos de restricción en paralelo) 1609, un segundo circuito

conmutador 1620, la memoria 1607 de mensajes de bordes C2V, un procesador vectorial 1608 de nodos variables (p. ej., Z procesadores de nodos variables en paralelo), una memoria 1612 de decisiones duras y un tercer conmutador 1622, acoplados entre sí según lo ilustrado en la Fig. 16.

Diversas realizaciones de procesadores de nodos de restricción individuales y de procesadores de nodos variables individuales, Z de los cuales pueden ser usados en paralelo para implementar, respectivamente, el procesador vectorial 1609 de nodos de restricción y el procesador 1608 de nodos variables, están descritas en detalle en la Solicitud Provisoria de Patente Estadounidense 60 / 328.469, titulada “Procesadores de nodos para su uso en descodificadores de control de paridad”, que fue presentada el 10 de octubre de 2001. Los inventores de la presente solicitud de patente son también los inventores nombrados en la solicitud provisoria de patente incorporada.

A fin de dar soporte a la actualización independiente y / o paralela de mensajes de restricción y variables en la realización de la Fig. 16, se usan las memorias 1606, 1607 de mensajes individuales de bordes y los circuitos 1620, 1621 de conmutación para dar soporte, respectivamente, a las operaciones de procesamiento de nodos de restricción y de nodos variables. Como en la realización de la Fig. 15, cada una de las memorias 1606, 1607 de mensajes es capaz de almacenar L mensajes vectoriales (de ZxK bits). Cada mensaje vectorial, p. ej., una columna de Z mensajes de K bits, en las memorias 1606, 1607, puede ser leído, o escrito, en una única operación de lectura o de escritura.

La memoria 1606 de mensajes de bordes V2C es usada para almacenar mensajes V2C y, por lo tanto, tiene una entrada de datos acoplada con la salida del circuito 1621 de conmutación, que recibe datos desde el procesador vectorial 1608 de nodos variables. La memoria 1607 de mensajes C2V es usada para almacenar mensajes de bordes C2V y, por lo tanto, tiene una entrada de escritura acoplada a la salida del procesador vectorial 1609 de nodos de restricción.

Los conmutadores 1620 y 1621 son usados para acoplar el procesador vectorial 1608 de nodos variables con la entrada de la memoria de mensajes de bordes V2C y la salida de la memoria de mensajes de bordes C2V, respectivamente. En una realización específica, los vectores de mensajes son almacenados en el orden de los receptáculos vectoriales de restricción. Los vectores de mensajes son escritos en la memoria 1607 de mensajes de bordes C2V y son leídos de la memoria 1606 de mensajes de bordes V2C en el orden de los receptáculos vectoriales de restricción, es decir, linealmente, y por tanto no se requiere ningún control externo (la salida de índices de bordes desde el módulo 1602 de control de descodificador atraviesa el módulo 1604 de ordenamiento de mensajes sin cambios, hasta su salida de índices de bordes de restricción). Los vectores de mensajes son leídos de la memoria 1607 de mensajes de bordes C2V y escritos en la memoria 1606 de mensajes de bordes V2C, en el orden de los receptáculos vectoriales variables. El módulo 1604 de ordenamiento de mensajes genera la señal de índice de borde variable que indica este ordenamiento. Obsérvese que esta señal controla la lectura de la memoria 1607 de mensajes de bordes C2V y que es entregada a la memoria 1606 de mensajes de bordes V2C después de ser retardada. El retardo da cuenta del tiempo requerido para el procesamiento realizado por los conmutadores 1620 y 1621, y el procesador vectorial 1608 de nodos variables. Este retardo puede ser una función del grado del nodo que está siendo procesado, según lo indicado en la Fig. 16 por la señal de grado de nodos variables.

Para evitar atascos del trayecto de procesamiento, debidos al retardo variable, tanto los nodos de restricción como los variables están ordenados de modo que los nodos del mismo grado sean procesados de forma contigua. Una reducción adicional de los atascos del trayecto que ocurren en el límite de grupos de nodos con distintos grados puede ser lograda clasificando los grupos de nodos por el grado de manera monótona, p. ej., en orden creciente o decreciente del grado. Por simplicidad de implementación, las realizaciones 900, 1500 y 1600 suponen un orden creciente del grado.

En la realización específica ilustrada en la Fig. 16, los vectores están almacenados en orden de rotación de los nodos vectoriales de restricción. El conmutador 1620 rota los mensajes en cada vector en rotación variable, según cada mensaje vectorial C2V avanza hacia el procesador vectorial 1608 de nodos variables, y luego el conmutador 1621 aplica la rotación inversa al mensaje vectorial saliente V2C, correspondiente al mismo borde vectorial. La señal de rotación entregada al conmutador 1620 es entregada al conmutador 1621 mediante el circuito 1624 de inversión de rotación, después de un retardo correlacionado con el tiempo de procesamiento en el procesador vectorial de nodos de restricción. Este retardo puede depender del grado del nodo de restricción, según lo indicado por la señal de grado de nodo de restricción emitida por la memoria 1610 de grados.

El descodificador 1600 incluye el módulo 1602 de control de descodificador. El módulo de control de descodificador funciona de manera similar a la del módulo 1502 de control previamente expuesto.

Sin embargo, en 1602 no se genera ninguna señal de control C / V. La función de generación de índices de borde puede ser proporcionada por un contador que recorre cíclicamente el conjunto entero de bordes vectoriales antes de recomenzar.

Además de emitir las decisiones blandas, las decisiones duras son generadas, una por borde, por cada una de las Z

unidades de procesamiento de nodos variables en el procesador vectorial 1608 de nodos variables, cada vez que se genera un mensaje vectorial V2C. Mientras los mensajes vectoriales son escritos en la memoria 1606 de mensajes de bordes V2C, las salidas de decisiones duras de Zx1 bits son escritas en la memoria 1612 de salida de decisiones duras, después de haber sido rotadas por el conmutador 1622. El conmutador 1622 y la memoria 1612 de decisiones duras funcionan bajo las mismas señales de control, respectivamente, que el conmutador 1621 y la memoria 1606 de mensajes de bordes V2C.

Los Zx1 vectores rotados resultantes son entregados al verificador vectorial 1612 de control de paridad, que incluye Z verificadores de control de paridad conectados en paralelo. El verificador 1614 determina si los controles de paridad están satisfechos y, si todos están satisfechos, entonces se genera una señal de convergencia y se envía al módulo de control de descodificador. En respuesta a la recepción de una señal que indica convergencia, el módulo de control de descodificador detiene la descodificación sobre la palabra de código recibida. En la realización 1600, la señal de detección de convergencia está disponible una iteración después de que haya sido escrita una palabra de código en el almacén temporal de salida, ya que la verificación de restricciones para los datos de la iteración N se hace durante la iteración N+1, y la señal de convergencia está disponible al completar la iteración N+1.

En el descodificador 1600, el cual emplea distintos circuitos de detección de convergencia, la Fig. 17 ilustra una realización similar al descodificador 1700, y la verificación de restricciones se efectúa “sobre la marcha”, según los valores X de salida de ZxK bits son escritos en el almacén temporal 1716 de salida. En este caso, el bloque 1712 de memoria mantiene el rastro del estado de las restricciones, según las restricciones vectoriales son actualizadas por decisiones duras emitidas desde el procesador vectorial 1708 de nodos variables. Cada ubicación de memoria de estados de restricción corresponde a una restricción de control de paridad de palabras de código. En la última actualización de cualquier ubicación de estado de restricción, se verifican estos valores de control de paridad. Si todas las verificaciones durante la iteración N son satisfechas, entonces una señal de convergencia será generada y emitida por el verificador vectorial 1714 de control de paridad, inmediatamente después de la iteración N. Esto calificará, para el módulo 1702 de control de descodificador, como válidos los datos en el almacén temporal 1716 de salida. En la realización de la Fig. 17, el módulo 1704 de ordenamiento de mensajes genera una señal adicional que define el índice de nodo de restricción (distinto al índice de borde), que no es generada en la realización de la Fig. 16. El índice de nodo de restricción identifica el destino del nodo de restricción del mensaje V2C actual. Este campo sirve como índice para la memoria 1712 de estados de restricción a la cual se suministra.

Si bien requieren un poco más de circuitos que la realización de la Fig. 15, las realizaciones de la Fig. 16 y la Fig. 17 tienen la ventaja de un uso más eficaz de los procesadores vectoriales 1609/1709, 1608/1708 de nodos de restricción y variables, dado que ambos procesadores de nodos vectoriales son utilizados por completo durante cada iteración del procesamiento. Además, el tiempo de descodificación se reduce en comparación a la realización de la Fig. 15, ya que el procesamiento de nodos de restricción y variables se realiza en paralelo, p. ej., simultáneamente.

Los procedimientos de descodificación descritos en lo que antecede admiten que la descodificación del paso de mensajes, p. ej., la descodificación de LDPC, sea realizada usando software y ordenadores de propósito general, capaces de dar soporte a operaciones SIMD. En tales realizaciones, uno o más procesadores paralelos sirven como unidades de procesamiento vectorial o bien el hardware dentro de un único procesador puede ser usado para realizar múltiples operaciones de procesamiento vectorial en paralelo. En tales realizaciones, la memoria de bordes, el mapa de permutaciones y la información sobre el número de mensajes por nodo pueden ser todos almacenados en una memoria común, p. ej., la memoria principal de los ordenadores. La lógica de control del paso de mensajes y la lógica de control del descodificador pueden ser implementadas como rutinas de software ejecutadas en la unidad de procesamiento del ordenador. Además, el dispositivo de conmutación puede ser implementado usando software y una o más instrucciones de procesamiento SIMD.

Los procedimientos de descodificación de LDPC descritos en lo que antecede permiten que la descodificación de LDPC sea realizada sobre diversas plataformas de hardware, tales como las Formaciones de Compuertas Programables en el Terreno, o en un Circuito Integrado Específico de la Aplicación. La presente invención es especialmente útil en estas configuraciones, donde el paralelismo sencillo puede ser explícitamente explotado.

Numerosas variaciones adicionales de los procedimientos y aparatos de descodificación de la presente invención serán evidentes para los expertos en la técnica, a la vista de la anterior descripción de la invención. Tales variaciones han de ser consideradas dentro del alcance de la invención, según lo definido por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un procedimiento de realización del procesamiento de descodificación del paso de mensajes de control de paridad, usando gráficos vectorizados de LDPC que representan matrices elevadas de control de paridad, por lo cual, en una matriz elevada de control de paridad, los elementos 0 de una matriz H de control de paridad de un código de LDPC proyectado son reemplazados por matrices de ceros de dimensiones ZxZ, y los elementos 1 de la matriz H de control de paridad son reemplazados por matrices de permutación de dimensiones ZxZ, comprendiendo el procedimiento las etapas de:

mantener L conjuntos de mensajes de K bits en un dispositivo (1506, 1607, 1707) de almacenamiento de mensajes, incluyendo cada conjunto de mensajes de K bits los primeros Z mensajes en pasar, donde L y Z son números enteros positivos mayores que uno y K es un número entero positivo distinto de cero, por lo cual cada uno de dichos conjuntos de Z mensajes de K bits es escrito o leído como una unidad individual, usando una instrucción SIMD;

emitir uno de dichos conjuntos leídos de Z mensajes de K bits, desde el dispositivo (1506, 1607, 1707) de almacenamiento de mensajes;

realizar una operación de reordenamiento de mensajes sobre dicho conjunto leído de Z mensajes de K bits, para producir un conjunto reordenado de Z mensajes de K bits;

suministrar, en paralelo, los Z mensajes de K bits en el conjunto reordenado de mensajes a un procesador vectorial (1508, 1608, 1707) de nodos, que incluye Z unidades de procesamiento paralelo de nodos; operar el procesador vectorial (1508, 1608, 1707) de nodos para realizar operaciones de procesamiento de nodos variables, usando como entrada los Z mensajes de K bits suministrados, por lo cual una operación de procesamiento de nodos variables es realizada en cada una de las Z unidades de procesamiento paralelo de nodos, y una operación de procesamiento de nodos variables incluye generar un valor de decisión, y

examinar los valores de decisión generados para determinar si ha sido satisfecha una condición de descodificación.

2. El procedimiento de la reivindicación 1, que comprende adicionalmente: generar un identificador de conjunto de mensajes que indica el conjunto de Z mensajes de K bits a emitir por el dispositivo (1506, 1607, 1707) de almacenamiento de mensajes.

3. El procedimiento de la reivindicación 2, en el cual la etapa de emitir uno de dichos conjuntos de Z mensajes de K bits incluye: realizar la operación SIMD de salida usando dicho identificador de conjunto de mensajes para identificar el conjunto de Z mensajes de K bits a emitir como una unidad individual.

4. El procedimiento de la reivindicación 1, que comprende además: realizar una segunda operación de reordenamiento de mensajes, siendo realizada la segunda operación de reordenamiento de mensajes sobre un conjunto de Z mensajes de K bits emitidos desde el procesador vectorial (1508, 1608, 1708) de nodos, para producir un conjunto reordenado de mensajes de descodificador generados.

5. El procedimiento de la reivindicación 4, en el cual la etapa de realizar una segunda operación de reordenamiento de mensajes incluye realizar la inversa de la operación de reordenamiento de mensajes realizada sobre dicho conjunto de Z mensajes de K bits emitido por el dispositivo (1506, 1607, 1707) de almacenamiento de mensajes.

6. El procedimiento de la reivindicación 5, en el cual dicha operación de reordenamiento de mensajes es realizada como una función de la información de permutación del conjunto de mensajes, que incluye información de rotación cíclica.

7. El procedimiento de la reivindicación 1, que comprende adicionalmente: acceder a información almacenada de permutación de conjuntos de mensajes; y en el cual la etapa de realizar una operación de reordenamiento de mensajes incluye la etapa de: realizar dicho reordenamiento como una función de la información almacenada de permutación de conjuntos de mensajes a la que se ha accedido.

8. Un aparato (1500, 1600, 1700) para realizar operaciones de descodificación del paso de mensajes de control de paridad, usando gráficos de LDPC vectorizados que representan matrices elevadas de control de paridad, por lo cual, en una matriz elevada de control de paridad, los elementos 0 de una matriz H de control de paridad de un código de LDPC proyectado son reemplazados por matrices de ceros de dimensiones ZxZ, y los elementos 1 de la matriz H de control de paridad son reemplazados por matrices de permutación de dimensiones ZxZ, comprendiendo el aparato:

un origen (1506, 1607, 1707) de mensajes para suministrar al menos un conjunto de Z mensajes de K bits desde

uno cualquiera de entre al menos L conjuntos de Z mensajes de K bits, donde Z es un número entero positivo mayor que uno y K y L son números enteros positivos distintos de cero, por lo cual cada uno de dichos conjuntos de Z mensajes de K bits es escrito o leído como una unidad individual usando una instrucción SIMD;

un procesador vectorial (1508, 1608, 1708) de nodos que incluye Z unidades de procesamiento paralelo de nodos, siendo cada unidad de procesamiento de nodos para realizar una operación de procesamiento de nodos variables que incluye generar un valor de decisión;

un dispositivo (1520, 1620, 1720) de conmutación acoplado con el origen (1506, 1607, 1707) de mensajes y con el procesador vectorial (1508, 1608, 1708) de nodos, el dispositivo (1520, 1620, 1720) de conmutación para pasar conjuntos de Z mensajes de K bits, cada conjunto de Z mensajes de K bits pasado en paralelo, entre dicho origen (1506, 1607, 1707) de mensajes y dicho procesador vectorial (1506, 1608, 1708) de nodos, y para reordenar los mensajes en al menos uno de dichos conjuntos pasados de Z mensajes de K bits, en respuesta a información de control de conmutación;

un verificador vectorial (1514, 1614, 1714) de control de paridad, adaptado para examinar los valores de decisión generados, para determinar si ha sido satisfecha una condición de descodificación.

9. El aparato de la reivindicación 8, en el cual dicho origen (1506, 1607, 1707) de mensajes es un dispositivo de memoria.

10. El aparato de la reivindicación 8, que comprende adicionalmente:

un módulo (1504, 1604, 1704) de control de ordenamiento de mensajes, acoplado con dicho dispositivo (1520, 1620, 1720) de conmutación, para generar dicha información de control de conmutación, usada para controlar el reordenamiento de mensajes en dicho al menos un conjunto de Z mensajes de K bits.

11. El aparato de la reivindicación 10, en el cual el dispositivo (1520, 1620, 1720) de conmutación incluye circuitos para realizar una operación de rotación de mensajes para reordenar los mensajes incluidos en dicho al menos un conjunto de Z mensajes de K bits.

12. El aparato de la reivindicación 10, en el cual el módulo (1504, 1604, 1704) de control de ordenamiento de mensajes almacena información sobre el orden en el cual los conjuntos de Z mensajes de K bits han de ser recibidos desde dicho origen (1506, 1607, 1707) de mensajes, e información que indica qué reordenamiento de mensajes ha de ser realizado por dicho conmutador sobre conjuntos individuales de Z mensajes de K bits recibidos desde dicho origen (1506, 1607, 1707) de mensajes.

13. El aparato de la reivindicación 10, en el cual el módulo (1504, 1604, 1704) de control de ordenamiento de mensajes está adicionalmente acoplado con dicho origen (1506, 1607, 1707) de mensajes y genera secuencialmente identificadores de conjunto, controlando cada identificador de conjunto al origen (1506, 1607, 1707) de mensajes para emitir un conjunto de Z mensajes de K bits.

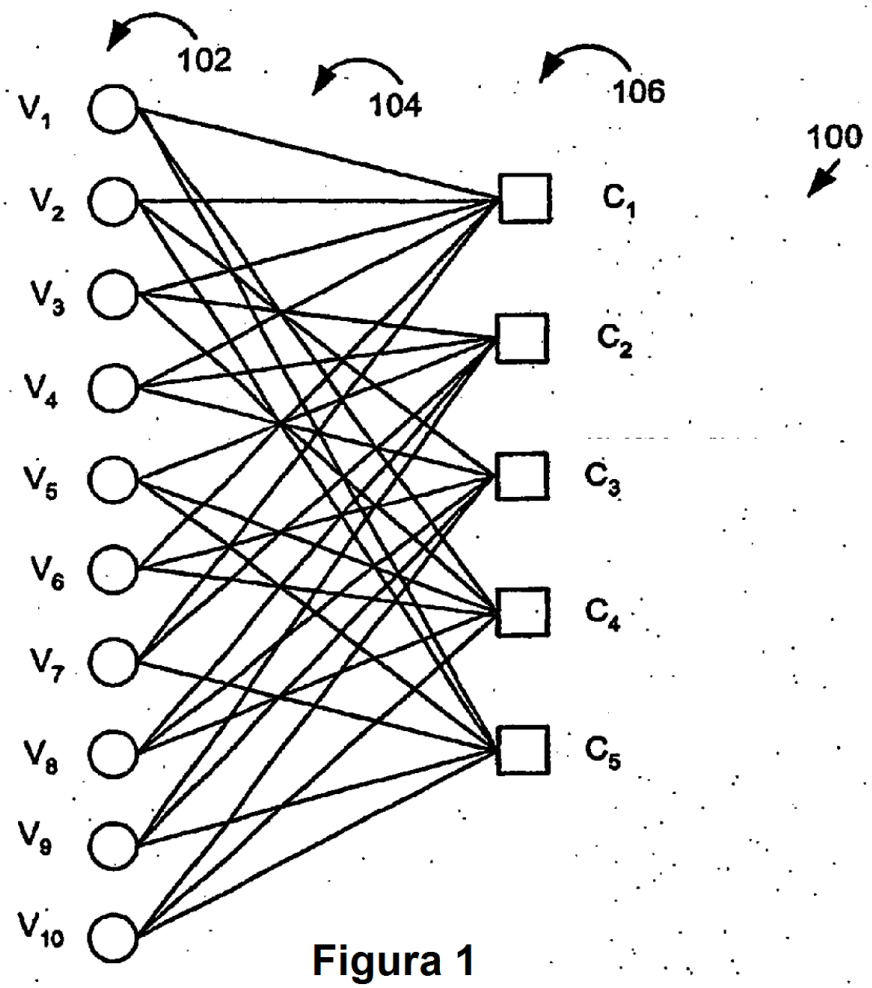


Figura 1

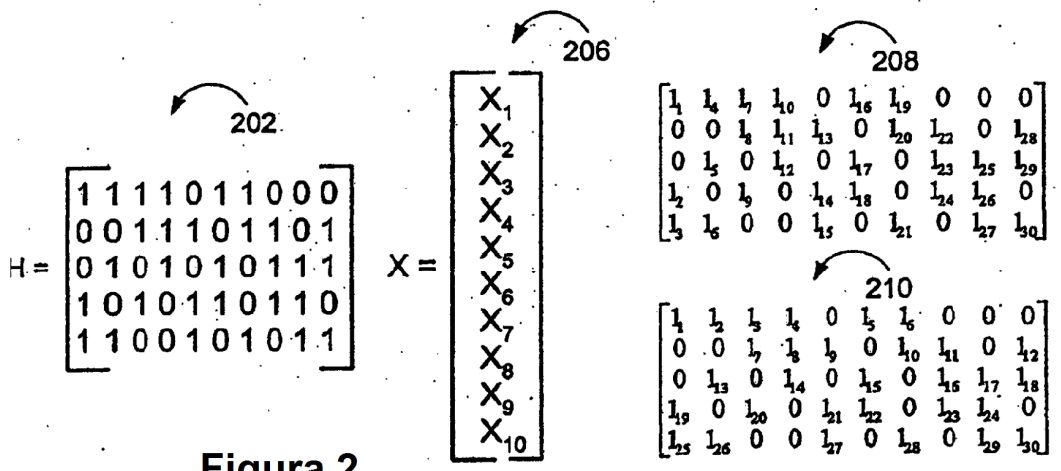

202
206
208
210
H =
X =
1 1 1 1 0 1 1 0 0 0
0 0 1 1 1 0 1 1 0 1
0 1 0 1 0 1 0 1 1 1
1 0 1 0 1 1 0 1 1 0
1 1 0 0 1 0 1 0 1 1
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10

Figura 2

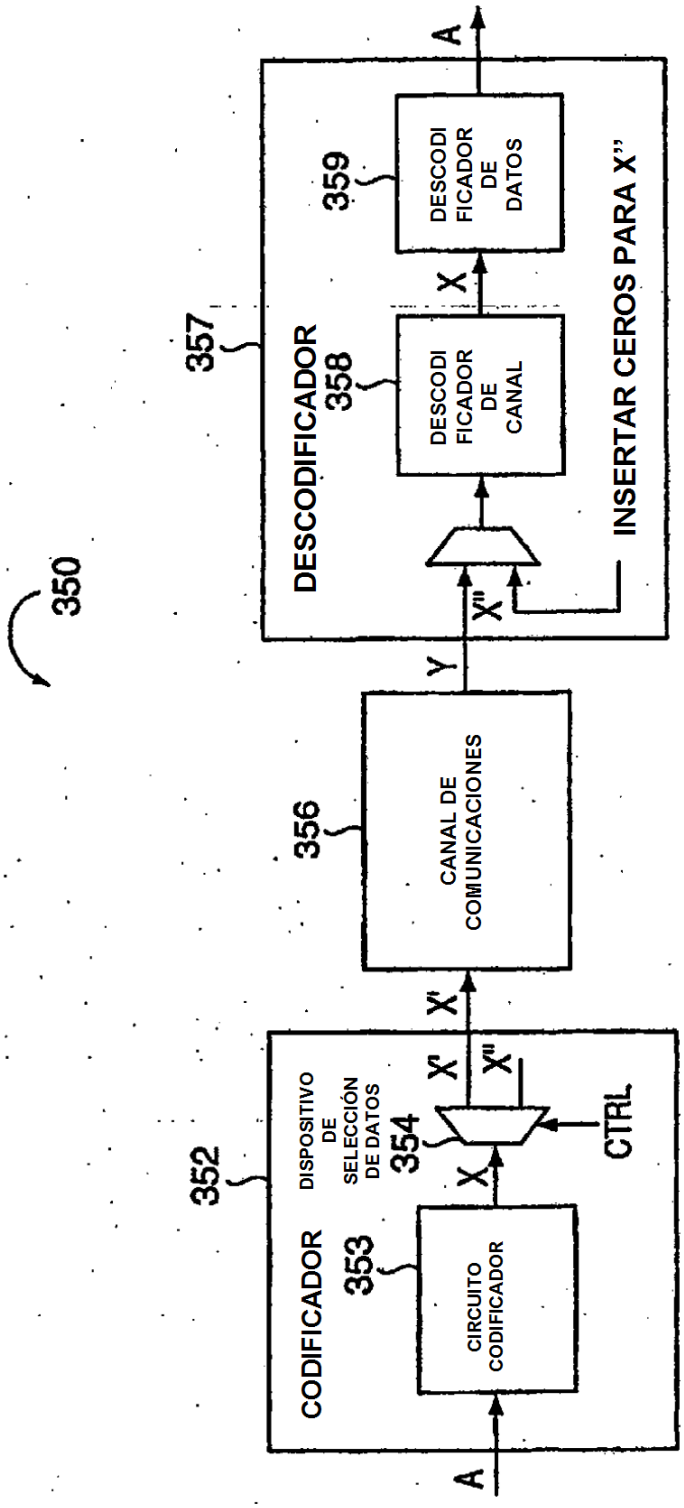

350
CODIFICADOR
352
DISPOSITIVO DE SELECCIÓN DE DATOS
354
CIRCUITO CODIFICADOR
353
A
X
X'
X''
CTRL
X'
CANAL DE COMUNICACIONES
356
Y
DESCODIFICADOR
357
DESCODI FICADOR DE CANAL
358
DESCODI FICADOR DE DATOS
359
X
A
X''
INSERTAR CEROS PARA X''

FIG. 3

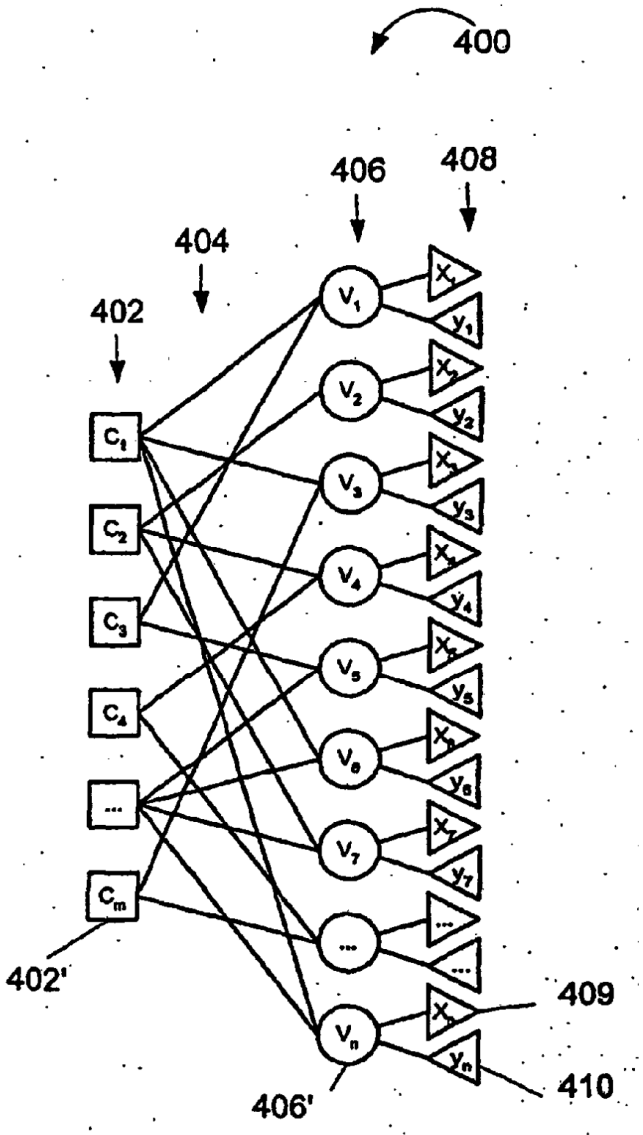

400
406
408
404
402
C1
C2
C3
C4
...
Cm
V1
V2
V3
V4
V5
V6
V7
...
Vn
X1
y1
X2
y2
X3
y3
X4
y4
X5
y5
X6
y6
X7
y7
...
...
Xn
yn
402'
406'
409
410

Figura 4

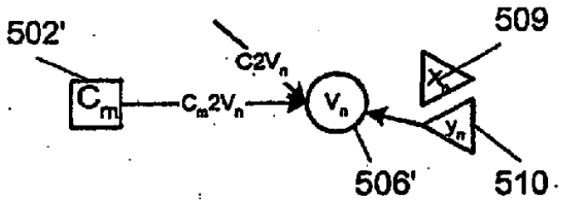


Figura 5a

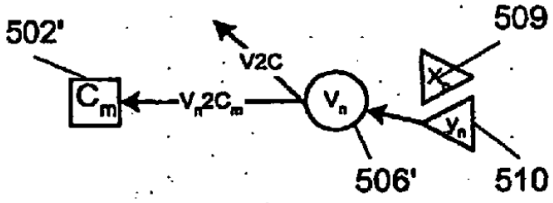


Figura 5b

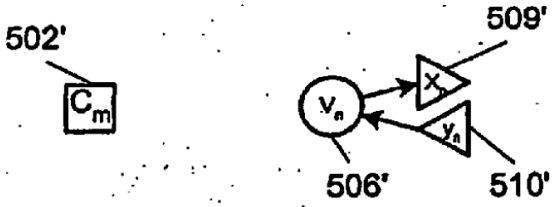


Figura 5c

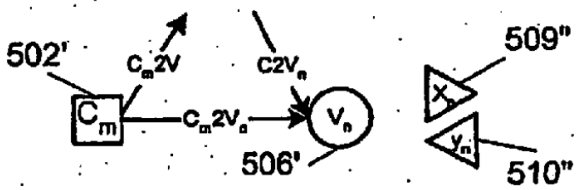


Figura 5d

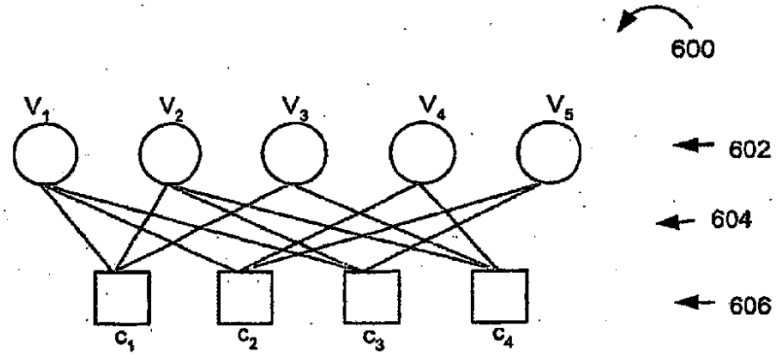


Figura 6

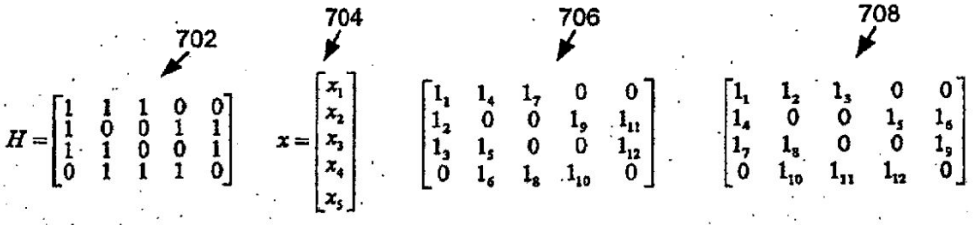


Figura 7

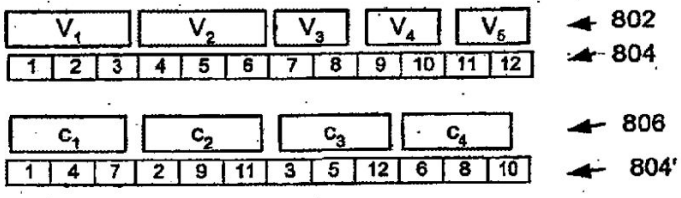


Figura 8

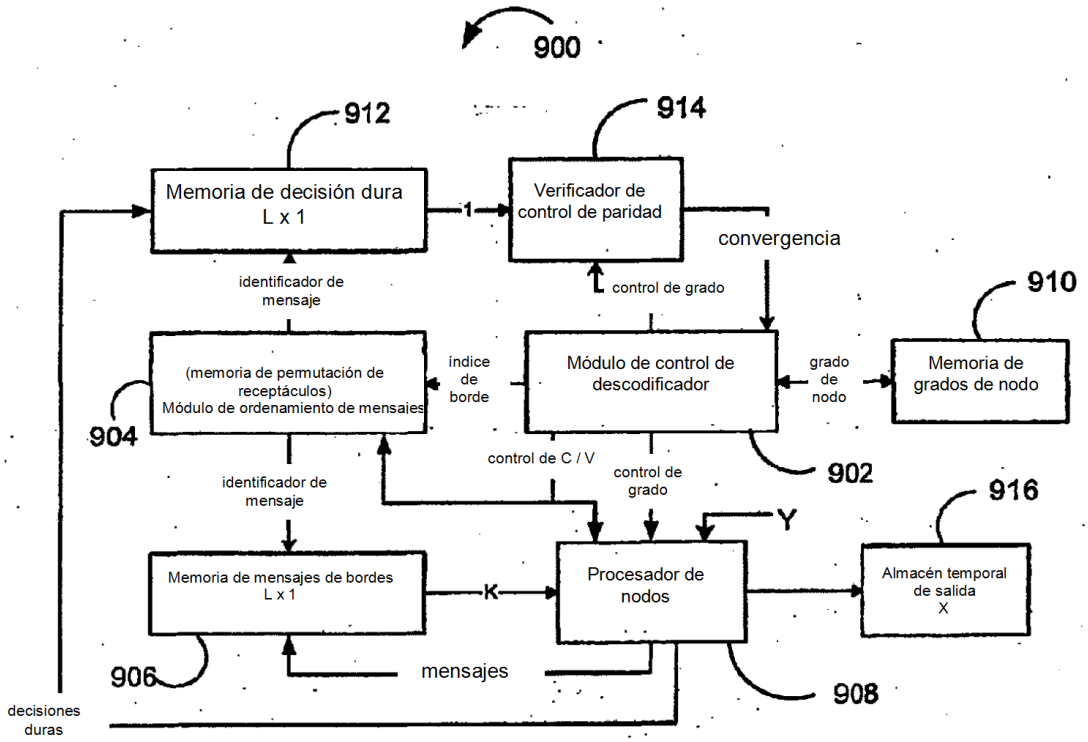

900
912
914
Memoria de decisión dura
L x 1
1
Verificador de
control de paridad
convergencia
identificador de
mensaje
control de grado
910
(memoria de permutación de
receptáculos)
Módulo de ordenamiento de mensajes
índice
de
borde
Módulo de control de
descodificador
grado
de
nodo
Memoria de
grados de nodo
904
control de C / V
control de
grado
902
identificador de
mensaje
916
Y
Memoria de mensajes de bordes
L x 1
K
Procesador de
nodos
Almacén temporal
de salida
X
906
mensajes
908
decisiones
duras

Figura 9

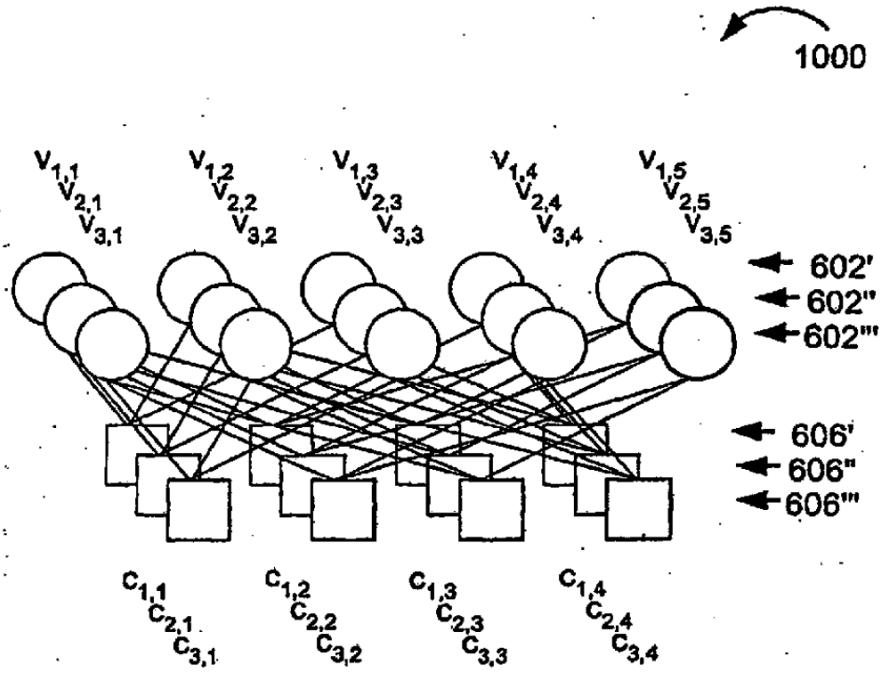


Figura 10

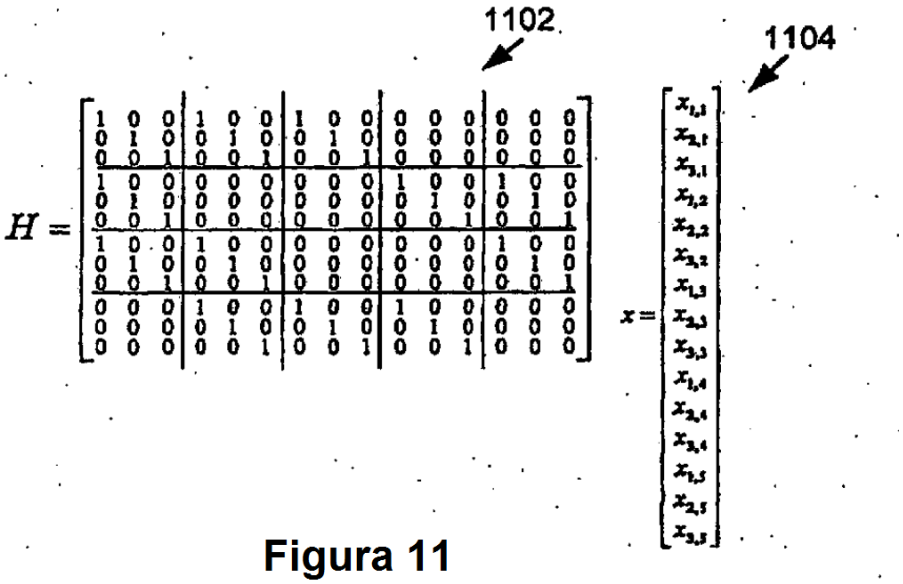
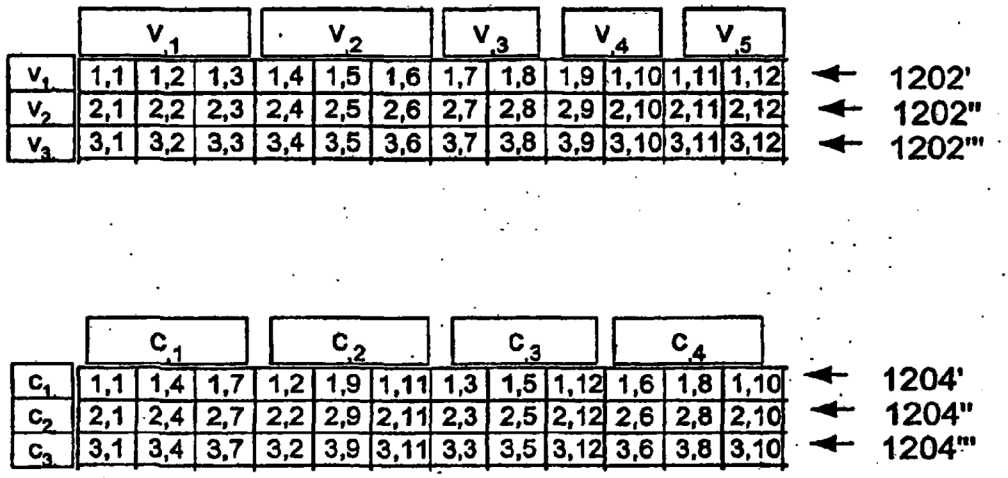


Figura 11



	$V_{,1}$			$V_{,2}$			$V_{,3}$		$V_{,4}$		$V_{,5}$		
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

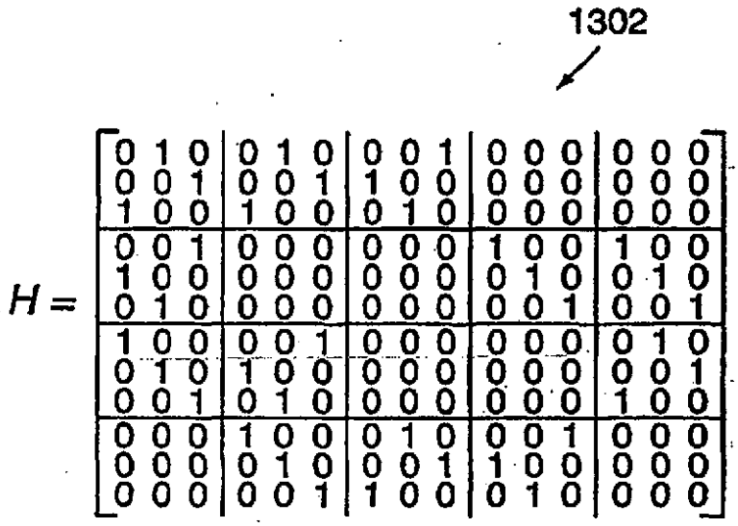


FIGURA 13

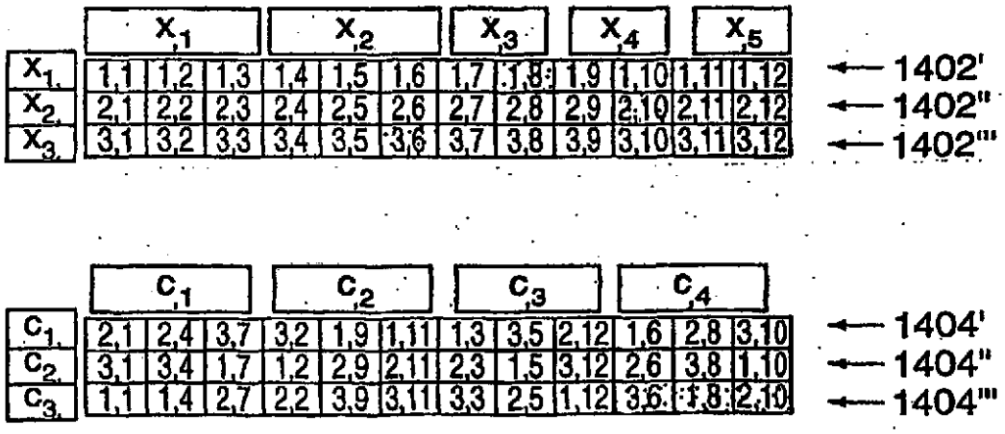


FIGURA 14

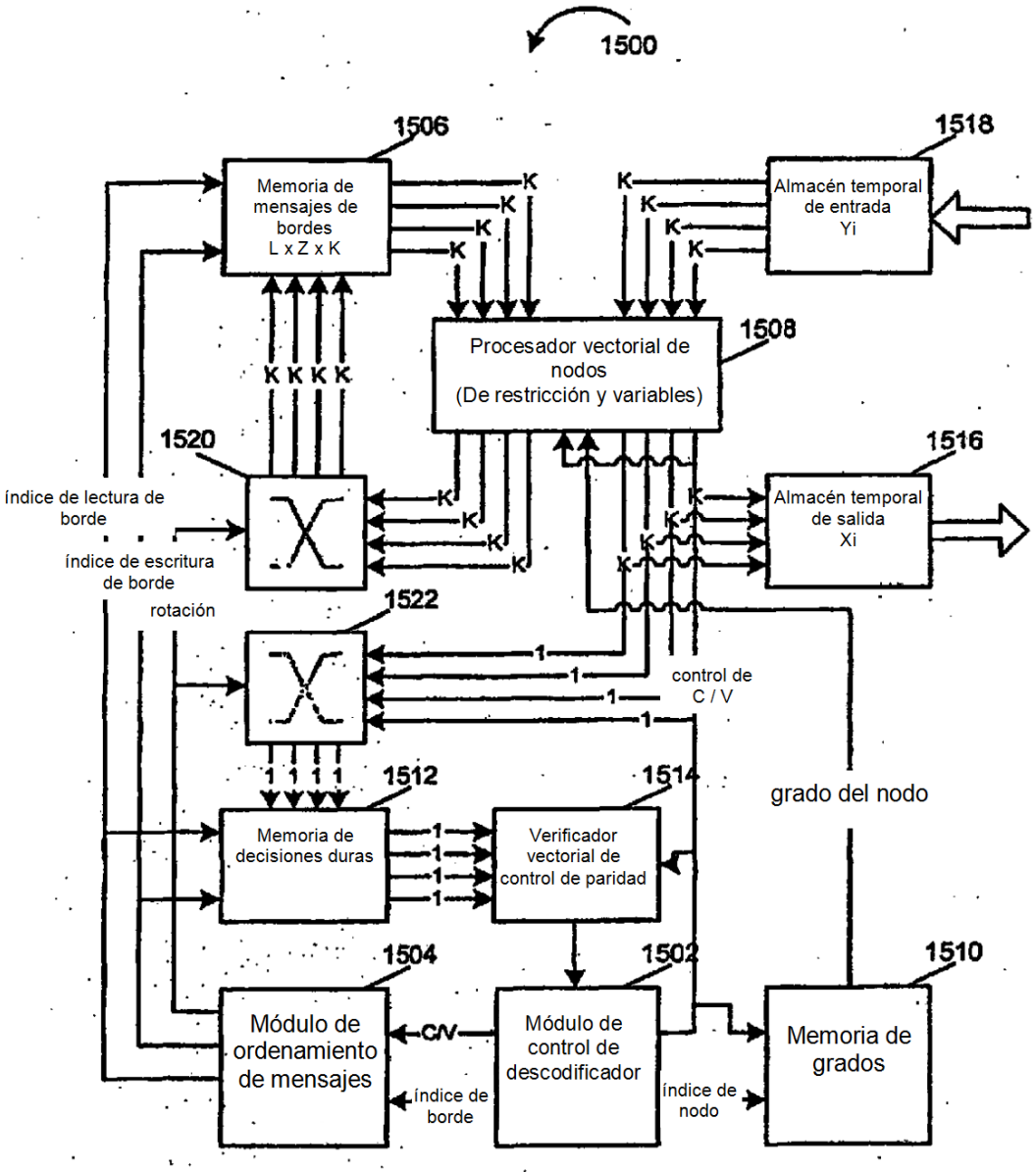

1500
1506
Memoria de mensajes de bordes L x Z x K
1518
Almacén temporal de entrada Yi
1508
Procesador vectorial de nodos (De restricción y variables)
1520
índice de lectura de borde
índice de escritura de borde
rotación
1516
Almacén temporal de salida Xi
1522
control de C / V
grado del nodo
1512
Memoria de decisiones duras
1514
Verificador vectorial de control de paridad
1504
Módulo de ordenamiento de mensajes
C/V
índice de borde
1502
Módulo de control de descodificador
índice de nodo
1510
Memoria de grados
K
1

Figura 15

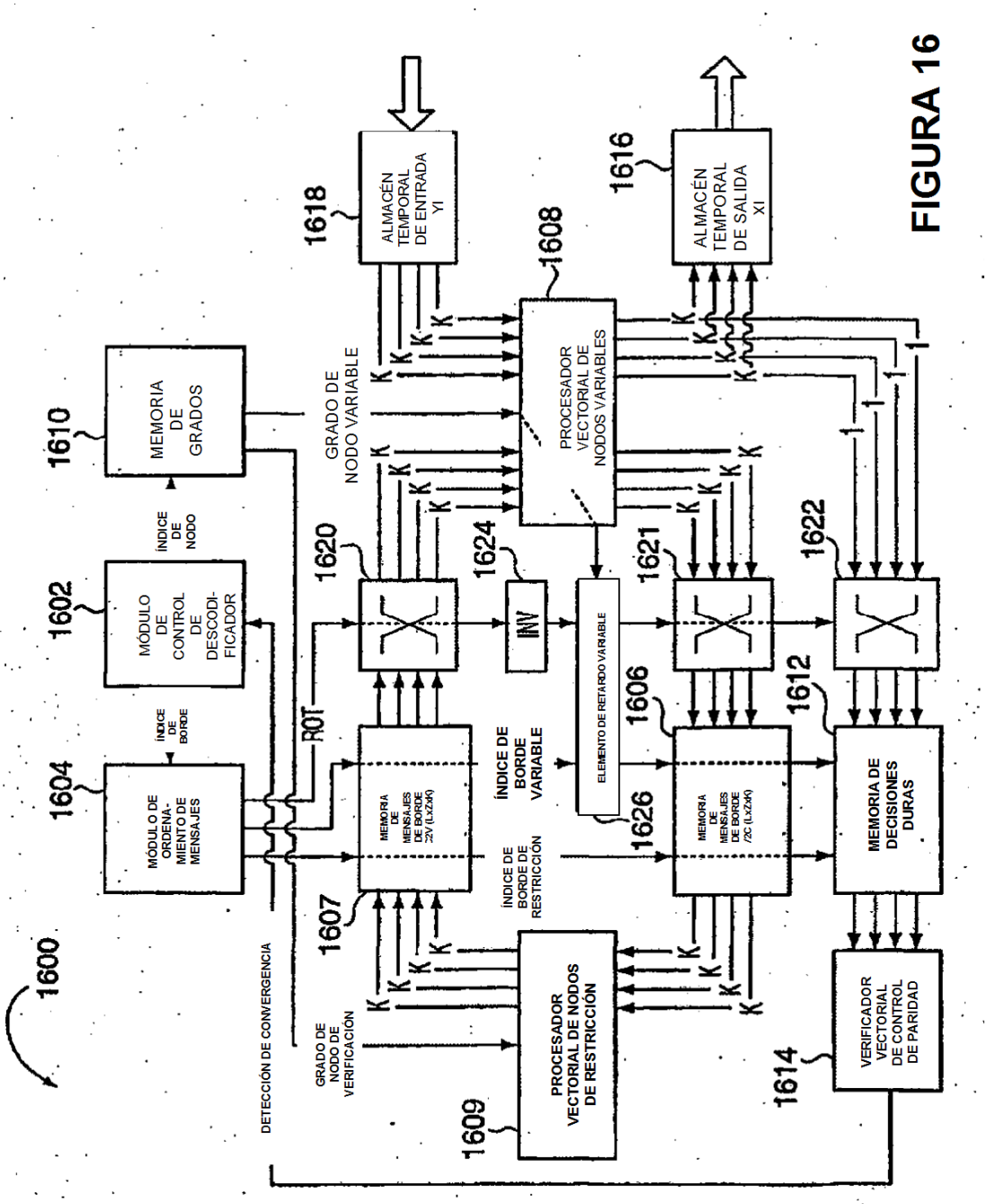

1600
1602
MÓDULO DE CONTROL DE DESCODIFICADOR
ÍNDICE DE NODO
1604
MÓDULO DE ORDENAMIENTO DE MENSAJES
ÍNDICE DE BORDE
1606
MEMORIA DE MENSAJES DE BORDE /2V (LXZ4K)
1607
MEMORIA DE MENSAJES DE BORDE /2C (LXZ4K)
1608
PROCESADOR VECTORIAL DE NODOS VARIABLES
1609
PROCESADOR VECTORIAL DE NODOS DE RESTRICCIÓN
1610
MEMORIA DE GRADOS
1612
MEMORIA DE DECISIONES DURAS
1614
VERIFICADOR VECTORIAL DE CONTROL DE PARIDAD
1616
ALMACÉN TEMPORAL DE SALIDA XI
1618
ALMACÉN TEMPORAL DE ENTRADA YI
1620
1621
1622
1624
INV
1626
ELEMENTO DE RETARDO VARIABLE
ROT
ÍNDICE DE BORDE VARIABLE
ÍNDICE DE BORDE DE RESTRICCIÓN
GRADO DE NODO VARIABLE
GRADO DE NODO DE VERIFICACIÓN
DETECCIÓN DE CONVERGENCIA
K
1

FIGURA 16

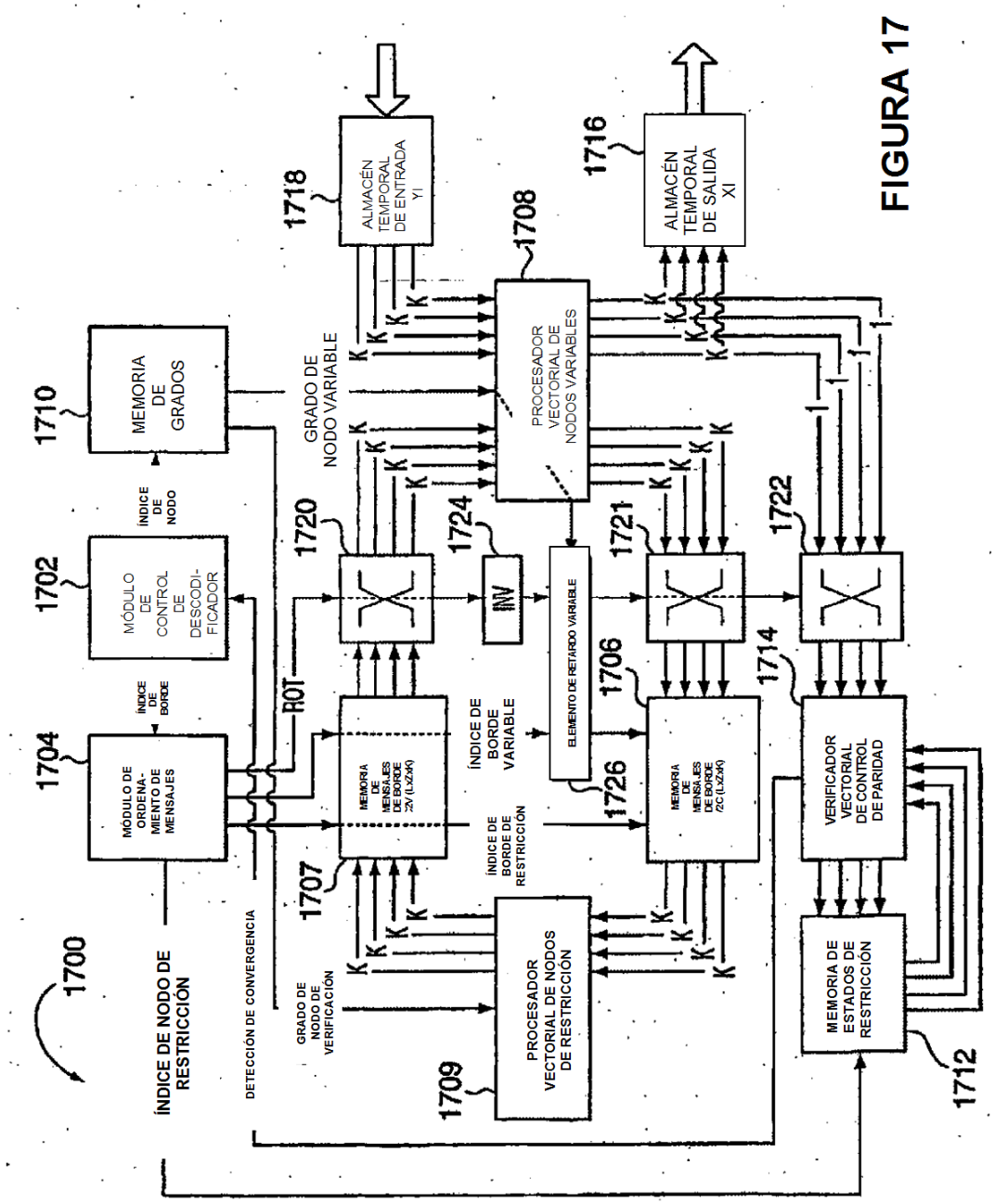

1700
ÍNDICE DE NODO DE RESTRICCIÓN
1702
MÓDULO DE CONTROL DE DESCODI-FICADOR
ÍNDICE DE NODO
1704
MÓDULO DE ORDENA-MIENTO DE MENSAJES
ÍNDICE DE BORDE
1706
MEMORIA DE MENSAJES DE BORDE /2V (LxZM)
1707
MEMORIA DE MENSAJES DE BORDE /2C (LxZM)
1708
PROCESADOR VECTORIAL DE NODOS VARIABLES
1709
PROCESADOR VECTORIAL DE NODOS DE RESTRICCIÓN
1710
MEMORIA DE GRADOS
1712
MEMORIA DE ESTADOS DE RESTRICCIÓN
1714
VERIFICADOR VECTORIAL DE CONTROL DE PARIDAD
1716
ALMACÉN TEMPORAL DE SALIDA XI
1718
ALMACÉN TEMPORAL DE ENTRADA YI
1720
1721
1722
1724
INV
1726
ELEMENTO DE RETARDO VARIABLE
ROT
GRADO DE NODO VARIABLE
ÍNDICE DE BORDE VARIABLE
ÍNDICE DE BORDE DE RESTRICCIÓN
GRADO DE NODO DE VERIFICACIÓN
DETECCIÓN DE CONVERGENCIA
K
1

FIGURA 17