(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 356 767**

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

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **02800936 .3**
(96) Fecha de presentación : **07.10.2002**
(97) Número de publicación de la solicitud: **1442527**
(97) Fecha de publicación de la solicitud: **04.08.2004**

(54) Título: **Procesadores de nodo para su uso en decodificadores de control de paridad.**

(30) Prioridad: **10.10.2001 US 328469 P**
10.10.2001 US 975331
04.04.2002 US 117264

(45) Fecha de publicación de la mención BOPI:
13.04.2011

(45) Fecha de la publicación del folleto de la patente:
13.04.2011

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

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

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

DESCRIPCIÓN

Solicitudes relacionadas

La presente solicitud reivindica el beneficio de la solicitud provisional estadounidense n.º de serie 60/328,469 presentada el 10 de octubre de 2001, y el beneficio de la solicitud provisional estadounidense n.º de serie 60/298,480 presentada el 15 de junio de 2001, y es una continuación en parte de la solicitud de patente estadounidense n.º de serie 09/975,331 presentada el 10 de octubre de 2001.

Campo de la invención

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

Antecedentes

Los códigos de corrección de errores están presentes de manera generalizada en sistemas de comunicaciones y almacenamiento de datos. Recientemente se ha suscitado un considerable interés en una clase de códigos conocidos como códigos de control de paridad de baja densidad (LDPC).

Los códigos LDPC se representan adecuadamente mediante grafos bipartitos, a menudo llamados grafos de Tanner, en los que un conjunto de nodos, los nodos *variables*, corresponden a bits de la palabra de código y el otro conjunto de nodos, los nodos *de restricción*, a veces llamados nodos *de control*, corresponden al conjunto de restricciones de control de paridad que definen el código. Las aristas del grafo conectan nodos variables con nodos de restricción. Un nodo variable y un nodo de restricción se dice que son *vecinos* si están conectados por una arista en el grafo. Para simplificar, en general asumimos que un par de nodos están conectados como mucho por una arista. A cada nodo variable está asociado un bit de la palabra de código. En algunos casos, algunos de estos bits podrían *perforarse*, es decir, eliminarse de la palabra de código. Para simplificar, asumiremos de manera general que no se usa ninguna perforación.

Una secuencia de bits asociada en relación de uno a uno con la secuencia de nodos variables es una palabra de código del código si, y solo si, por cada nodo de restricción, los bits vecinos de la restricción (a través de su asociación con nodos variables) suman cero módulo dos, es decir, comprenden un número par de unos. Los decodificadores y algoritmos de decodificación usados para decodificar palabras de código LDPC funcionan intercambiando mensajes dentro del grafo a lo largo de las aristas y actualizando estos mensajes realizando cálculos en los nodos basándose en los mensajes entrantes. Tales algoritmos se denominarán generalmente algoritmos de paso de mensajes. A cada nodo variable en el grafo se le proporciona inicialmente un bit suave (*soft bit*), denominado *valor recibido*, que indica una estimación del valor del bit asociado según se determina mediante observaciones a partir de, por ejemplo, el canal de comunicaciones. Idealmente, las estimaciones para bits separados son estadísticamente independientes. Este ideal puede ser, y a menudo es, infringido en la práctica. Una colección de valores recibidos constituye una *palabra recibida*. Para los fines de esta solicitud, podemos identificar la señal observada por, por ejemplo, el receptor en un sistema de comunicaciones, con la palabra recibida.

El número de aristas unidas a un nodo, es decir, a un nodo variable o a un nodo de restricción, se denomina *grado* del nodo. Un grafo o código *regular* es uno en 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 se dice que el código es un código regular (j, k). Estos códigos fueron inventados originalmente por Gallager (1961). A diferencia de un código “regular”, un código irregular tiene nodos de restricción y/o nodos variables de diferentes grados. Por ejemplo, algunos nodos variables pueden ser de grado 4, otros de grado 3 y otros más de grado 2.

Aunque los códigos irregulares pueden ser más complicados de representar y/o implementar, se ha demostrado que los códigos LDPC irregulares pueden proporcionar un rendimiento superior en la corrección/detección de errores en comparación con los códigos LDPC regulares.

Con el fin de describir de manera más precisa el proceso de decodificación, presentamos el concepto de *zócalo* en la descripción de grafos LDPC. Un zócalo puede interpretarse como una asociación de una arista en el grafo con un nodo en el grafo. Cada nodo tiene un zócalo para cada arista unida al mismo y las aristas se “insertan en” los zócalos. De esta manera, un nodo de grado d tiene d zócalos unidos al mismo. Si el grafo tiene L aristas entonces hay L zócalos en el lado de nodo variable del grafo, llamados zócalos variables, y L zócalos en el lado de nodo de restricción del grafo, llamados zócalos de restricción. Con fines de identificación y ordenación, los zócalos variables pueden enumerarse 1,…, L de manera que los zócalos variables unidos a un nodo variable aparezcan de manera contigua. En tal caso, si los primeros tres nodos variables tienen grados d_1, d_2 y d_3 respectivamente, entonces los zócalos variables 1,…, d_1 están unidos al primer nodo variable, los zócalos variables d_1+1,..., d_1+d_2 están unidos al segundo nodo variable, y los zócalos variables d_1+d_2+1,…, d_1+d_2+d_3 están unidos al tercer nodo variable. Los zócalos de nodo de restricción pueden enumerarse de manera similar 1,…, L, apareciendo los zócalos de restricción unidos a un nodo de restricción de manera contigua. Una arista puede verse como un emparejamiento de zócalos, procediendo uno de cada par de cada lado del grafo. De esta manera, las aristas del grafo representan un entrelazado o permutación en los zócalos de un lado del grafo, por ejemplo, el lado de nodo variable, al otro, por

ejemplo, el lado de nodo de restricción. Las permutaciones asociadas con estos sistemas son a menudo complejas.

Un grafo 100 bipartito ejemplar que determina un código LDPC regular (3,6) de longitud diez y proporción 1/2 se muestra en la figura 1. Longitud diez indica que hay diez nodos variables V_1-V_{10}, cada uno identificado con un bit de la palabra de código X_1- X_{10} (y sin perforación en este caso), generalmente identificados por el número de referencia 102. Proporción 1/2 indica que hay la mitad de nodos de control que de nodos variables, es decir, hay cinco nodos de control C_1-C_5 identificados por el número de referencia 106. Proporción 1/2 además indica que las cinco restricciones son linealmente independientes, como se comenta más adelante. Cada una de las líneas 104 representa una arista, por ejemplo, un camino de comunicación o conexión, entre los nodos de control y los nodos variables a los que está conectada la línea. Cada arista identifica dos zócalos, un zócalo variable y un zócalo de restricción. Las aristas pueden enumerarse según sus zócalos variables o sus zócalos de restricción. La enumeración de los zócalos variables corresponde a la ordenación de aristas (de arriba hacia abajo) tal como aparece en el lado de nodo variable en el punto en el que se conectan a los nodos variables. La enumeración de los zócalos de restricción corresponde a la ordenación de aristas (de arriba hacia abajo) tal como aparece en el lado de nodo de restricción en el punto en el que se conectan a los nodos de restricción. Durante la decodificación, se pasan mensajes en ambos sentidos a lo largo de las aristas. De esta manera, como parte del proceso de decodificación se pasan mensajes a lo lardo de una arista desde un nodo de restricción hacia un nodo variable y viceversa.

Una alternativa al uso de un grafo para representar códigos es usar una representación matricial tal como se muestra en la figura 2. En la representación matricial de un código, la matriz H 202, comúnmente denominada matriz de control de paridad, incluye la información relevante de conexión de aristas, nodos variables y nodos de restricción. Para simplificar asumimos que una arista conecta como mucho un par de nodos cualquiera. En la matriz H, cada columna corresponde a uno de los nodos variables mientras que cada fila corresponde a uno de los nodos de restricción. Puesto 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 particular y a un nodo de restricción particular se ajusta a 1 si hay una arista presente en el grafo, es decir, si los dos nodos son vecinos, de lo contrario se ajusta a 0. Por ejemplo, puesto que el nodo variable V_1 está conectado al nodo de restricción C_1 por una arista, se coloca un uno en la esquina superior izquierda de la matriz 202. Sin embargo, el nodo variable V_4 no se encuentra conectado al 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.

Se dice que las restricciones son linealmente independientes si las filas de H son vectores linealmente independientes sobre GF [2]. Enumerar las aristas por zócalos, variables o de restricción, corresponde a enumerar los 1 en H. La enumeración de zócalos variables corresponde a enumerar de arriba hacia abajo dentro de las columnas y luego proceder de izquierda a derecha de columna a columna, tal como se muestra en la matriz 208. La enumeración de zócalos de restricción corresponde a enumerar de izquierda a derecha a través de las filas y procediendo de arriba hacia abajo de fila a fila, tal como se muestra en la matriz 210.

En el caso de una representación matricial, la palabra de código X que va a transmitirse puede representarse como un vector 206 que incluye los bits X_1- X_n de la palabra de código que va a procesarse. Una secuencia de bits X_1-X_n es una palabra de código si y sólo si el producto de la matriz 206 y 202 es igual a cero, esto es: Hx=0.

En el contexto de la explicación de palabras de códigos asociadas a grafos LDPC, debe apreciarse que en algunos casos la palabra de código puede *perforarse*. La perforación es el acto de eliminar bits de una palabra de código para dar lugar, como efecto, a una palabra de código más corta. En el caso de los grafos LDPC esto significa que algunos de los nodos variables en el grafo corresponden a bits que en realidad no se transmiten. Estos nodos variables y los bits asociados con estos se denominan a menudo variables de estado. Cuando se usa perforación, el decodificador puede usarse para reconstruir la parte de la palabra de código que no se comunica físicamente a través de un canal de comunicaciones. Cuando se transmite una palabra de código perforada, el dispositivo de recepción puede rellenar inicialmente los valores de palabra recibidos (bits) ausentes con unos y ceros asignados, por ejemplo, de una manera arbitraria, junto con una indicación (bit suave) de que estos valores no son en absoluto fiables, es decir, que estos valores están *borrados*. Para simplificar, asumimos que, cuando se usan, estos valores rellenados por el receptor forman parte de la palabra recibida que va a procesarse.

Considérese, por ejemplo, el sistema 350 mostrado en la figura 3. El sistema 350 incluye un codificador 352, un decodificador 357 y un canal 356 de comunicación. El codificador 350 incluye un circuito 353 codificador que procesa los datos A de entrada para producir una palabra de código X. La palabra de código X incluye, para fines de detección y/o corrección de errores, una cierta redundancia. La palabra de código X puede transmitirse a través del canal de comunicaciones. Como alternativa, la palabra de código X puede dividirse mediante un dispositivo 354 de selección de datos en una primera y una segunda parte X', X'' respectivamente mediante alguna técnica de selección de datos. Una de las partes de la palabra de código, por ejemplo, la primera parte X', puede entonces transmitirse a través del canal de comunicaciones hacia un receptor que incluye el decodificador 357, mientras que la segunda parte X'' se perfora. Como resultado de las distorsiones producidas por el canal 356 de comunicaciones, partes de la palabra de código transmitida pueden perderse o dañarse. Desde la perspectiva del decodificador, los bits perforados pueden interpretarse como perdidos.

En el receptor se insertan bits suaves dentro de la palabra recibida para que ocupen el lugar de los bits

perdidos o perforados. Los bits suaves insertados indican el borrado de los bits X" y/o bits perdidos en la transmisión.

El decodificador 357 intentará reconstruir la palabra de código X completa a partir de la palabra Y recibida y cualquier bit suave insertado, y entonces realizará una operación de decodificación de datos para producir A a partir de la palabra de código X reconstruida.

El decodificador 357 incluye un decodificador 358 de canal, por ejemplo, un decodificador LDPC, para reconstruir la palabra de código X completa a partir de la palabra Y recibida y cualquier bit suave insertado. Además incluye un decodificador 359 de datos para eliminar la información redundante incluida en la palabra de código para producir los datos A de entrada originales a partir de la palabra de código X reconstruida.

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

Como se menciono anteriormente, las operaciones de decodificación LDPC generalmente comprenden algoritmos de paso de mensajes. Hay numerosos algoritmos de paso de mensajes potencialmente útiles y el uso de tales algoritmos no está limitado a los decodificadores LDPC. Como se comentará en detalle más adelante, la presente invención está dirigida a procedimientos y aparatos que proporcionan una implementación simple, por ejemplo, de baja complejidad de hardware, de un algoritmo decodificador que da un rendimiento muy bueno y a menudo casi óptimo en numerosas circunstancias. El algoritmo propuesto puede verse como una aproximación del algoritmo de propagación de creencias ampliamente conocido.

Para facilitar el entendimiento de la invención explicada en las secciones que siguen, se dará ahora una breve descripción matemática de la propagación de creencias.

La propagación de creencias para códigos LDPC (binarios) puede expresarse como sigue. Los mensajes transmitidos a lo largo de las aristas del grafo se interpretan como verosimilitudes logarítmicas log para el bit asociado al nodo variable. Aquí, (p_0, p_1) representa una distribución de probabilidad condicional en el bit asociado donde p_x indica la probabilidad de que el bit adopte el valor *x*. Los bits suaves proporcionados al decodificador por el receptor también vienen dados en forma de una verosimilitud logarítmica. De esta manera, los valores recibidos, es decir, los elementos de la palabra recibida, son verosimilitudes logarítmicas de los bits asociados $\frac{p_0}{p_1}$ condicionadas por la observación de los bits proporcionados por el canal de comunicación. En general, un mensaje *m* representa la verosimilitud logarítmica *m* y un valor recibido *y* representa la verosimilitud logarítmica *y*. Para bits perforados, el valor de verosimilitud logarítmica recibido *y* se ajusta a 0, indicando $p_0=p_1=½$.

Considérense las reglas de paso de mensajes de la propagación de creencias. Los mensajes se indican mediante m^{C2V} para mensajes desde nodos de control hacia nodos variables y mediante m^{V2C} para mensajes desde nodos variables hacia nodos de control. Considérese un nodo variable con *d* aristas. Para cada arista j=1,…, d, m^{C2V} (i) indica el mensaje entrante en la arista i. En la inicialización del proceso de decodificación se ajusta $m^{C2V}=0$ para cada arista. En general, los mensajes salientes desde nodos variables vienen dados por . El valor suave decodificado saliente desde un nodo (no un mensaje de arista) correspondiente a esta operación viene dado por . La decisión dura saliente asociada a esta salida se obtiene a partir del signo de x_{out}.

En los nodos de control es a menudo más conveniente representar los mensajes usando su 'signo' y magnitudes. De esta manera, para un mensaje *m,* $m_p \in$ GF [2] indica la 'paridad' del mensaje, es decir, $m_p = 0$ si m es 0 y $m_p = 1$ si m < 0. Además $m_r \in [0, \infty]$ indica la magnitud de *m*. De esta manera, se obtiene que $m = -1^{mp} m_r$. En el nodo de control las actualizaciones para m_p y m_r son independientes. Se obtiene, para un nodo de control 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 se define F(x):= ln coth(x/2). En las dos ecuaciones anteriores, el superíndice V2C indica los mensajes

entrantes en el nodo de restricción. Obsérvese que F es su propia inversa, es decir, $F^1(x)=F(x)$.

Breve descripción de las figuras

La figura 1 ilustra una representación de grafo bipartito de un código LDPC regular ejemplar de longitud 10.

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

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

La figura 4 es una representación de grafo bipartito de un código LDPC irregular ejemplar.

La figura 5, que comprende la combinación de las figuras 5a a 5d, ilustra las etapas realizadas como parte de una operación de decodificación LDPC según el código LDPC ilustrado en la figura 4.

La figura 6 ilustra un decodificador LDPC serie que incluye procesadores de nodo implementados según la presente invención.

La figura 7 ilustra una implementación de un procesador de nodo variable.

La figura 8 ilustra una implementación de un procesador de nodo de restricción.

Las figuras 9 y 10 ilustran la primera y la segunda operación de transformación, respectivamente, asociadas con el procesamiento de nodo de restricción en el que se utilizan tamaños de paso de cuantificación ln2 con mensajes de 5 bits.

Las figuras 11 y 12 ilustran la primera y la segunda operación de transformación, respectivamente, asociadas con el procesamiento de nodo de restricción en el que se utilizan tamaños de paso de cuantificación ½ ln2 con mensajes de 6 bits.

La figura 13 ilustra la estructura general de un procesador de nodo, implementado según la presente invención, para su uso en un decodificador de paso de mensajes.

La figura 14 ilustra varios mensajes y valores generados durante el procesamiento de mensajes correspondientes a nodos de distintos grados, en secuencia, por el procesador de nodo ilustrado en la figura 13.

La figura 15 ilustra un aparato que puede usarse para generar valores de verosimilitud logarítmica cuantificados a partir de valores detectados por un receptor.

Se llama la atención adicionalmente sobre el documento ENGLING Y ET AL, “VLSI architectures for iterative decoders in magnetic recording channels” IEEE TRANSACTIONS ON MAGNETICS, IEEE SERVICE CENTER, NUEVA YORK, NY, EE.UU., Volumen 37, n.º 2, marzo de 2001, páginas 748-755, XP002958229. Este documento se refiere a las complejidades de la implementación VLSI de decodificadores de entrada suave - salida suave (*soft-input soft-output*, SISO). Estos decodificadores se usan en algoritmos iterativos basados en turbocódigos o códigos de control de paridad de baja densidad (LDPC), y prometen una ventaja significativa en el rendimiento de errores de bit frente a los sistemas de verosimilitud máxima de respuesta parcial (PRML) convencionalmente usados, a costa de una mayor complejidad. Se analizan los requisitos de hardware y memoria computacional, y se analizan sugerencias para una decodificación de menor complejidad y una lógica de control reducido. El documento además se refiere a la concatenación en serie de códigos entrelazados, usando un código de bloque externo con un canal de respuesta parcial actuando como codificador interno para aplicaciones de almacenamiento magnético.

Se llama la atención adicionalmente sobre el documento A-Raw G, et al., “Optimizing iterative decoding of low-density parity check codes on programmable pipelined parallel architectures”, Proc., IEEE Global Telecommunications Conf., Globecom 2001, 25-29 de noviembre de 2001, San Antonio, Texas, páginas 3012-3018, XP10747546. Este documento se refiere a la minimización de la latencia de la decodificación iterativa de códigos de control de paridad de baja densidad usando un algoritmo de suma-producto en una arquitectura paralela segmentada programable de baja complejidad. Se describen técnicas heurísticas para resolver los problemas de optimización combinatoria con NP duro del mapeo y la planificación de las tareas de procesamiento de la decodificación de un código LDPC arbitrario en *n* unidades de procesamiento segmentadas paralelas.

Se llama la atención también al documento HE YU-CHENG ET AL: “Fast decoding of LDPC codes using quantisation” ELECTRONICS LETTERS, IEE STEVENAGE, GB, vol. 38, n.º 4, 14 de febrero de 2002, páginas 189-190, XP006017844 ISNN: 0013-5194. Este documento se refiere a un algoritmo de propagación de creencias cuantificado para la decodificación de códigos de control de paridad de baja densidad. El algoritmo descrito posibilita esquemas de cuantificación flexibles.

SUMARIO DE LA INVENCIÓN

Según la presente invención, se proporcionan un sistema de detección y/o corrección de errores, según lo establecido en la reivindicación 1, y un procedimiento de detección y/o corrección de errores, según lo establecido en la reivindicación 10. Realizaciones adicionales se reivindican en las reivindicaciones dependientes.

La presente invención está dirigida a procedimientos y aparatos para realizar operaciones de decodificación que se usan junto con técnicas de decodificación de paso de mensajes. Las técnicas de la presente invención son muy apropiadas particularmente para su uso con códigos LDPC.

En el material de los antecedentes de esta solicitud se proporciona una descripción matemática del algoritmo de propagación de creencias que puede usarse junto con la decodificación LDPC. Es evidente, según la descripción, que la principal dificultad encontrada en la implementación del algoritmo tiene que ver con la función F y su inversa. Otras operaciones requeridas tienden a ser relativamente simples de implementar.

Para facilitar la implementación en hardware de un decodificador LDPC, en algunas realizaciones de la invención se cuantifican valores de verosimilitud logarítmica a múltiplos enteros de ½ ln 2. Los valores de verosimilitud logarítmica pueden ser, por ejemplo, razones de verosimilitud logarítmica o aproximaciones de las mismas.

Las razones de verosimilitud logarítmica pueden describirse como sigue. Sea *x* un bit. Se asume que *x*, posiblemente junto con otros bits, se transmiten como una señal y que el receptor observa *z* como consecuencia. La razón de verosimilitud logarítmica para *x* se define como , donde p(z|x=i) indica la probabilidad condicional de observar *z* dada la condición x=i. Hay numerosos procedimientos y formas posibles para calcular *y* dependiendo del esquema de señalización, el modelo de canal, etc. Asumiendo que las dos posibilidades para *x* son a priori igualmente probables, la razón de verosimilitudes igual a la razón de las posteriores probabilidades de *x* dada *z*, es decir, De esta manera se indica a menudo la razón de verosimilitud logarítmica como simplemente donde p_i indica una probabilidad condicional de que x=i. Durante diversas realizaciones de decodificación LDPC de la invención se calcula, al menos aproximadamente, tales razones de verosimilitud logarítmica (por ejemplo los mensajes) donde la información condicional aumenta con cada iteración.

$$y = \log \frac{p(z \mid x=0)}{p(z \mid x=1)}$$

$$\frac{p(z \mid x=0)}{p(z \mid x=1)}$$

$$\frac{p(x=0 \mid z)}{p(x=1 \mid z)}$$

En una implementación práctica de decodificador de paso de mensajes digital, los mensajes de decodificador se representan mediante un número finito de bits. En anticipación del uso de la cuantificación según la invención, se introduce un parámetro de ajuste a escala δ. En las implementaciones ejemplares descritas de la invención, los mensajes son enteros m y la interpretación del mensaje es que representa la verosimilitud logarítmica mδ. De esta manera, en las realizaciones ejemplares, los presentes mensajes son enteros que, cuando se ajustan a escala por δ según la invención determinan las verosimilitudes logarítmicas asociadas.

$$y = \log \frac{p_0}{p_1}$$

Según la invención, se aproxima la función F descrita en la sección de los antecedentes de esta solicitud sustituyéndola por otra función que se presta mejor para la implementación. En diversas realizaciones se modifica ligeramente la aproximación inicial mencionada, junto con su inversa, de manera que el decodificador resultante se comporte de manera más aproximada a un verdadero decodificador de propagación de creencias.

La idea de la aproximación puede entenderse como la ampliación de F(δx) en una serie en $e^{-\delta x}$ como sigue:

$$F(\delta x) = \ln(\coth(\delta x/2)) = \ln(1+e^{-\delta x}) - \ln(1-e^{-\delta x}) = \sum_{k=1}^{\infty} \frac{2}{2k-1} e^{-k\delta x}.$$

De esta manera, para valores grandes de δx, la función ln(coth(δx/2)) puede aproximarse adecuadamente por $2e^{-\delta x}$. Teniendo en mente que nuestro objetivo es encontrar una implementación de baja complejidad, la aproximación de la función F(x) por $2e^{-\delta x}$ es bastante atractiva. Si se reemplaza simplemente F(x) por $2e^{-\delta x}$, entonces la parte de magnitud de la actualización del nodo de control adopta la forma

$$m_r^{C2V}(j) = -\left[\delta^{-1} \ln\left(\left(\sum_{i=1}^{d} e^{-\delta m_r^{V2C}(i)}\right) - e^{-\delta m_r^{V2C}(j)}\right)\right],$$

donde [x] indica la parte entera de x.

Obsérvese que si se escoge δ=ln2 entonces el calculo requerido para la actualización de fiabilidad del nodo de control es particularmente simple, permitiendo una implementación que use operaciones de suma y desplazamiento. Para proporciones de código aproximadamente por encima de 1/5, resulta que establecer δ=ln2 proporciona suficiente resolución para alcanzar un comportamiento de propagación de creencias caso completo. Para proporciones más bajas, esta cuantificación es demasiado basta y puede ser preferible establecer

$$\delta = \tfrac{1}{2} \ln 2$$

. La principal ventaja de escoger δ de esta manera es que simplifica en gran medida el cálculo de la operación ln.

Considérese el caso en el que δ=ln2. En este caso los cálculos de actualización del nodo de restricción adoptan la forma

$$m_r^{C2V}(j) = -\left[\log_2\left(\left(\sum_{i=1}^{d} 2^{-m_r^{V2C}(i)}\right) - 2^{-m_r^{V2C}(j)}\right)\right].$$

Puesto que es la parte entera de la función logarítmica la que se usa, la función puede implementarse como un codificador de prioridad, es decir, es posible implementar la función simplemente determinando la ubicación del primer 1 en la representación binaria del argumento.

La aproximación de ln(coth(δx/2)) por $2e^{-\delta x}$ puede dar como resultado errores relativamente grandes en el caso de valores pequeños de δx. Es posible compensar algo el error con un leve ajuste de las funciones directa e inversa. Más específicamente, pueden aplicarse pequeños desfases, y en distintas realizaciones de la invención se aplican, durante la actualización del nodo de restricción. Al hacer esto, los cálculos de actualización de adoptan la forma

$$m^{V2C}(j) = r + \left(\sum_{i=1}^{d} m^{C2V}(i)\right) - m^{C2V}(j),$$

en los nodos variables, y adoptan la forma

$$m_r^{C2V}(j) = C_2 - (\text{Prioridad})\log_2\left(\left(\sum_{i=1}^{d} 2^{C_1 - m_r^{V2C}(i)}\right) - 2^{C_1 - m_r^{V2C}(j)}\right)$$

en los nodos de control, donde C_1 y C_2 son constantes y “Prioridad” se refiere a la operación de hallar el primer ‘1’ en una representación binaria apropiada del argumento. Se profundizará más en el significado y las implementaciones ejemplares del codificador de prioridad en la descripción detallada que sigue.

Se pasa ahora las consideraciones de hardware. Obsérvese que, ignorando por el momento las transformaciones, el cálculo dominante para actualizaciones de mensajes tiene la forma simplificada:

$$m_{out}(j) = \Sigma_{i=1}^{d}\ m_{in}(i) - m_{in}(j).$$

Se propone realizar las operaciones de paso de mensajes en serie en el tiempo. Los mensajes entrantes llegan, por ejemplo, uno por cada ciclo de reloj. Es por tanto deseable tener una estructura segmentada eficaz que pueda producir un mensaje de arista saliente por cada ciclo de reloj. La presente invención también incluye una descripción de una estructura particular, un procesador de nodo, para implementar de esta manera la regla computacional anterior. La implementación particular proporciona un cálculo racional eficaz de las operaciones de paso de mensajes.

Descripción detallada de la invención

Como se mencionó anteriormente, los procedimientos y aparatos de decodificación de la presente invención se describirán, con fines explicativos, en el contexto de una realización de decodificador LDPC. Las etapas implicadas en la decodificación de un código LDPC se describirán en primer lugar con referencia a las figuras 4 y 5 seguido de una explicación más detallada de la presente invención.

La figura 4 ilustra un código LDPC irregular ejemplar usando un grafo 400 bipartito. El grafo incluye *m* nodos 402 de control, n nodos 406 variables, y una pluralidad de aristas 404. Entre los nodos de control y los nodos variables se intercambian mensajes a través de las aristas 404. Bits de entrada suaves y_1 a y_n, correspondientes a la palabra recibida Y, y salidas suaves (o duras) x_1 a x_n, se indican mediante el número de referencia 408. El m-ésimo nodo de control se identifica usando el número de referencia 402’, el enésimo nodo variable se identifica usando el número de referencia 406’ mientras que la enésima entrada suave y_n y la enésima salida suave x_n se indican en la figura 4 usando los números de referencia 410, 409 respectivamente.

Los nodos 406 variables procesan los mensajes desde los nodos 402 de restricción junto con los valores suaves de entrada procedentes de la palabra recibida $y_1,\ldots, y_n$ para actualizar el valor de las variables de salida $x_1,\ldots, x_n$ correspondientes a los nodos variables y para generar mensajes para los nodos de restricción. Se genera un mensaje por un nodo variable por cada arista conectada al mismo. El mensaje generado se transmite a lo largo de la arista desde el nodo variable hacia el nodo de restricción unido a la arista. Con fines explicativos, los mensajes desde nodos variables hacia nodos de restricción se indicarán, de vez en cuando en la presente solicitud, mediante el uso de la abreviatura V2C mientras que los mensajes desde nodos de restricción hacia nodos variables se indicarán mediante el uso de la abreviatura C2V. Pueden añadirse índices a las componentes V y C de esta abreviatura para indicar el nodo variable y el nodo de restricción particular que sirve como origen/destino de un mensaje particular. Cada nodo 402 de restricción es responsable de procesar los mensajes recibidos desde los nodos variables a través de las aristas. Los mensajes V2C recibidos desde los nodos variables se procesan por los nodos 402 de restricción para generar mensajes C2V que se transmiten de nuevo a lo largo de las aristas unidas a cada nodo de restricción. Los nodos 406 variables entonces procesan los mensajes C2V, junto con los valores de entrada suaves, para generar y transmitir nuevos mensajes V2C, y generar salidas suaves, x_i. La secuencia de realización del procesamiento en los nodos 406 variables que comprende: transmitir mensajes generados a los nodos 402 de control, generar en los nodos variables salidas suaves x_i, y recibir mensajes desde los nodos de control, puede realizarse varias veces, es decir, iterativamente, hasta que las salidas x_i de los nodos variables 406 indiquen que la palabra de código se ha decodificado correctamente o que se ha satisfecho algún otro criterio de detención, por ejemplo, la finalización de un número fijo o iteraciones de paso de mensajes. Se debe apreciar que la secuencia de operaciones descrita anteriormente no tiene que ocurrir estrictamente en el orden descrito. El procesamiento de nodo puede avanzar de forma asíncrona y el procesamiento de nodo variable y de restricción puede ocurrir simultáneamente. Sin embargo, la lógica del proceso iterativo es tal como se ha descrito.

Los mensajes, V2C y C2V, pueden ser de uno o más bits, por ejemplo, K bits cada uno, donde K es un valor entero positivo distinto de cero. De forma similar, las salidas suaves x_i pueden ser de uno o más bits. Mensajes y salidas de múltiples bits proporcionan la oportunidad de transmitir información de confianza o fiabilidad en el mensaje o salida. En el caso de una salida de múltiples bits (suave), el signo del valor de salida suave puede usarse para proporcionar la salida dura de un único bit del proceso de decodificación correspondiente a un nodo variable, por ejemplo, los bits de la palabra de código decodificada. Valores de salida suaves pueden corresponder a valores suaves decodificados o, alternativamente, a la denominada información extrínseca (excluyendo la información de entrada correspondiente) que puede usarse en otro proceso iterativo mayor dentro del cual el decodificador LDPC no es más que un módulo.

El proceso iterativo de paso de mensajes asociado con la decodificación de un código LDPC se explicará ahora adicionalmente con referencia a las figuras 5a a 5d.

Cuando se decodifica un código LDPC, el procesamiento en cada nodo de restricción y variable puede realizarse independientemente. Por lo tanto, el procesamiento de nodo variable y/o de restricción puede realizarse en un nodo cada vez, por ejemplo, en secuencia, hasta que se haya completando todo o parte del procesamiento de nodo variable y de restricción para una iteración particular del proceso de decodificación. Esto permite proporcionar una única unidad de hardware de procesamiento y reutilizarla, si se desea, para realizar el procesamiento asociado con cada uno de los nodos variables y/o de restricción. Otra característica significativa de la decodificación LDPC es que los mensajes V2C y C2V usados durante una iteración de procesamiento particular no es necesario que se hayan generado al mismo tiempo, por ejemplo, durante la misma iteración de procesamiento. Esto permite implementaciones en las que el procesamiento de nodo de restricción y variable puede realizarse de forma asíncrona y en paralelo sin considerar el retardo de tiempo desde la última actualización de los mensajes usados. Después de un número suficiente de iteraciones y actualizaciones de mensajes en el que 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 grafo fue diseñado correctamente y no quedan errores sin corregir en la palabra recibida que está procesándose.

Dado que el procesamiento de cada nodo de control y nodo variable puede verse como una operación independiente, el procesamiento iterativo realizado en un solo nodo 502' de control C_n y nodo 506' variable V_n ejemplar se analizará ahora en más detalle con referencia a las figuras 5a-5d. Con fines de descripción se supondrá un algoritmo de propagación de creencias cuantificado. Los valores y mensajes recibidos son por lo tanto números reales. 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. Magnitudes mayores indican mayor fiabilidad. De esta manera, el número cero indica absoluta falta de fiabilidad y el signo (positivo o negativo) es irrelevante. Detalles de los cálculos realizados en el algoritmo se han descrito en la sección anterior.

La figura 5a ilustra la etapa inicial en un proceso de decodificación LDPC. Inicialmente, al nodo 506' variable V_n se le suministra la entrada suave, por ejemplo, los valores recibidos (1 o mas bits que representan una verosimilitud logarítmica) y_n de una palabra recibida que va a procesarse. Los mensajes C2V en el inicio de una operación de decodificación y la salida 509 suave X_n se ajustan inicialmente a cero. Basándose en las entradas recibidas, por ejemplo, los mensajes C2V de valor cero y la entrada y_n, el nodo 506' variable V_n genera un mensaje V2C para cada nodo de control al que está conectado. Normalmente, en la etapa inicial, cada uno de estos mensajes será igual a y_n.

En la figura 5b se muestran los mensajes V2C generados siendo transmitidos a lo largo de cada arista conectada al nodo 506' variable V_n. De esta manera, los mensajes V2C actualizados se transmiten de cada uno de los nodos 502 de control acoplados al nodo 506' variable V_n incluyendo el nodo 502' de control C_m.

Además de generar los mensajes V2C, el procesamiento del nodo variable da como resultado la actualización de la salida 509' suave X_n correspondiente al nodo variable que realiza el procesamiento. La salida suave X_n se muestra siendo actualizada en la figura 5c. Aunque se muestran como distintas etapas, la salida suave puede emitirse al mismo tiempo que se emiten los mensajes V2C.

Como se explicará más adelante, según algunas realizaciones de la presente invención, las salidas suaves (o sus decisiones duras asociadas) pueden usarse para determinar cuándo se ha recuperado una palabra de código a partir de la palabra recibida, es decir, cuándo se han satisfecho las restricciones de paridad por los valores de salida. Esto indica una decodificación satisfactoria (aunque la palabra de código hallada puede ser incorrecta, es decir, no la que fue transmitida) de este modo permitiendo detener el proceso de decodificación iterativo en el momento oportuno, por ejemplo, antes de que se complete algún número máximo fijado permitido de iteraciones de paso de mensajes.

El procesamiento de nodo de control puede realizarse una vez que un nodo de control, por ejemplo, el nodo 502' de control C_m, reciba mensajes V2C a lo largo de las aristas a las que está conectado. Los mensajes V2C recibidos se procesan en el nodo de control para generar mensajes C2V actualizados, uno por cada arista conectada al nodo de control particular. Como resultado del procesamiento de nodo de control, el mensaje C2V transmitido de nuevo a un nodo variable a lo largo de la arista dependerá del valor de cada uno de los mensajes V2C recibidos en las otras aristas conectadas al nodo de control, pero (por lo general y preferiblemente) no del mensaje V2C recibido desde el nodo variable particular al que se está transmitiendo el mensaje C2V. De esta manera, los mensajes C2V se usan para transmitir información generada a partir de mensajes recibidos desde nodos variables distintos del nodo al que se está transmitiendo el mensaje.

La figura 5d ilustra el paso de mensajes C2V actualizados a nodos variables incluyendo el nodo 506'. En particular, en la figura 5d el nodo 502' de restricción C_m se muestra emitiendo dos mensajes C2V actualizados, con el mensaje C_m2V_n actualizado siendo suministrado al nodo 506' variable V_n. El nodo 506' V_n también recibe (un) mensaje(s) $C2V_n$ actualizado(s) adicional(es) de otro(s) nodo(s) de restricción a los que está conectado.

Con la recepción de mensajes C2V actualizados, el procesamiento de nodo variable puede repetirse para generar mensajes V2C actualizados y salidas suaves. Entonces la actualización de mensajes C2V puede repetirse adicionalmente y así sucesivamente hasta que el criterio de detención del decodificador sea satisfecho.

De esta manera, el procesamiento mostrado en las figuras 5a-5d se repetirá después de la primera iteración, usando valores de mensajes actualizados en oposición a los valores iniciales, hasta que el proceso de decodificación se detenga.

La presente invención trata la forma de los mensajes que se pasan, los cálculos realizados sobre los mismos, y las estructuras de hardware que realizan estos cálculos. Para colocar la invención en contexto se describirá brevemente una implementación de decodificador LDPC. Esta implementación es con fines ilustrativos. Arquitecturas de implementación eficaces se comentan en la solicitud de patente estadounidense n.º de serie 09/975,333 presentada el 10 de octubre de 2001 titulada: "Procedimientos y aparatos para decodificar códigos LDPC".

Según una característica de la presente invención, los valores y de entrada de decodificación de paso de mensajes están en forma de valores de verosimilitud logarítmica que se cuantifican para ser múltiplos enteros de 1/2 ln 2. La generación de valores de verosimilitud logarítmica cuantificados puede involucrar el procesamiento de valores recibidos para generar a partir de los mimos valores de verosimilitud logarítmica, por ejemplo razones, o aproximaciones de los mismos. Los valores de verosimilitud logarítmica se cuantifican entonces usando tamaños de paso que son razones enteras de 1/2 ln 2 para producir los valores de verosimilitud logarítmica cuantificados. Los valores de verosimilitud logarítmica son, en varias realizaciones, razones de verosimilitud logarítmica o aproximaciones de las mismas. El uso de tales razones de verosimilitud logarítmica cuantificadas facilita la implementación del decodificador.

Los circuitos para generar valores de verosimilitud logarítmica cuantificados, por ejemplo razones, a partir de valores recibidos pueden estar incorporados directamente en un receptor que forma parte del canal 356 de comunicaciones que precede al decodificador de paso de mensajes de la presente invención, por ejemplo, el decodificador 600 mostrado en la figura 6.

Pueden usarse varios circuitos para generar razones de verosimilitud logarítmica cuantificadas según la invención. Haciendo referencia ahora brevemente a la figura 15, se muestra un circuito 1500 ejemplar para generar una razón de verosimilitud logarítmica cuantificada y_o, a partir de un valor Y_{in} recibido. El circuito de la figura 15 es apropiado en casos de señalización BPSK (+1,-1) con ruido gaussiano aditivo, donde puede asumirse que cada valor recibido es proporcional a la razón de verosimilitud logarítmica para el bit asociado. El tal caso, el valor recibido puede convertirse en una razón de verosimilitud logarítmica simplemente multiplicando por una constante.

El valor y_{in} puede ser un valor recibido por un circuito receptor que se procesa para producir la razón de verosimilitud logarítmica y_o suministrada por un decodificador de paso de mensajes de la presente invención. El circuito 1500 incluye un multiplicador 1502 para generar razones de verosimilitud logarítmica multiplicando el valor Y_{in} de entrada por $(2/s^2)$, donde s^2 es una constante correspondiente al ruido gaussiano aditivo introducido por el canal de comunicaciones en la señal recibida. La razón de verosimilitud logarítmica resultante se cuantifica entonces por el cuantificador 1504 para ser un múltiplo entero de ½ ln 2. El cuantificador 1504 puede implementarse dividiendo la razón de verosimilitud logarítmica generada por el multiplicador 1502 entre ½ ln 2 y entonces saturando el resultado hasta +15 o -15 cuando la magnitud excede de 15 y, en caso contrario, tomando los 5 lsb (bits menos significativos) del valor resultante como la razón de verosimilitud logarítmica Yo que puede suministrarse entonces como una entrada al circuito 600 decodificador.

La figura 6 representa un decodificador 600 en serie simple que realiza secuencialmente operaciones de procesamiento de mensajes, de una arista cada vez. El decodificador 600 LDPC comprende un módulo 610 de control de decodificador, una memoria 630 de aristas V2C, una memoria 650 de aristas C2V, un procesador 620 de nodo variable, un procesador 640 de nodo de restricción, y una memoria 660 intermedia de salida. Para simplificar, con respecto a la explicación de la invención, se supondrá que el decodificador ejecuta para un número fijo de iteraciones, es decir no se realiza detección de convergencia.

Las memorias 630, 650 de aristas V2C y C2V incluyen cada una L ubicaciones de memoria de K bits, correspondiendo cada ubicación de K bits a una arista y donde L es el número total de aristas en el grafo LDPC usado y K es el número de bits por mensaje intercambiado a lo largo de una arista. La memoria 660 intermedia de salida incluye memoria para almacenar valores de salida de nodo variable x, que pueden ser valores duros (1 bit) o suaves (más de 1 bit).

El módulo 610 de control de decodificador incluye información que describe el grafo en forma almacenada. Utiliza esta información para controlar el paso de mensajes como se describe a continuación. Cada valor y mensaje recibido se supone que está comprendido por K bits. El decodificador funciona en serie. En primer lugar realiza operaciones de actualización de nodo variable y entonces operaciones de actualización de nodo de restricción. Repite este ciclo algún número fijo de veces finalizando después de una actualización final de nodo variable. Inicialmente la memoria de mensajes C2V está rellena con ceros. (Obsérvese que el procesamiento de nodo variable y el procesamiento de nodo de restricción pueden ser físicamente simultáneos en el tiempo. El orden descrito anteriormente indica el flujo de información durante el procesamiento).

Ahora se describirá una actualización de nodo variable. El módulo 610 de control de decodificador hace que los mensajes C2V se lean desde la memoria de mensajes C2V en orden de zócalo de nodo variable y se entreguen al procesador 620 de nodo variable. El módulo 610 de control de decodificador señaliza a la memoria 650 de mensajes de aristas C2V un identificador de mensajes (por ejemplo, una ubicación de memoria o puntero) indicando qué mensaje debe ser leído en ese momento. Haciendo referencia a la figura 1, por ejemplo, los primeros tres mensajes entregados al procesador 620 de nodo variable serán los entrantes al nodo variable v_1. Los siguientes tres mensajes entregados al procesador 620 de nodo variable serán los entrantes al nodo variable v_2, y así sucesivamente.

Los mensajes para un nodo dado se procesan por el procesador 620 de nodo variable. El procesador 620 de nodo variable recibe desde el módulo 610 de control de decodificador una señal, la señal de reloj de nodo, que indica los límites de nodo. Esta señal informa al procesador de nodo variable, en efecto, acerca del grado del nodo que está procesándose actualmente. La señal puede enviarse, y en varias realizaciones se envía, por ejemplo, en el momento coincidente con la llegada de los últimos mensajes entrantes correspondientes a un nodo particular.

Los cálculos de actualización de nodo se realizan en el procesador 620 de nodo variable. Los mensajes V2C salientes se emiten en orden de zócalo variable, es decir, correspondiente al orden de aristas de los mensajes entrantes, y estos mensajes se almacenan en la memoria 630 de mensajes de aristas V2C. Valores de salida suaves o duros se almacenan en la memoria 660 intermedia de salida. Una vez completada la actualización de nodo variable, a menos que sea la última actualización, el decodificador procede a realizar una actualización de nodo de restricción.

Ahora se describirá una actualización de nodo de restricción. Es muy similar a una actualización de nodo variable y se hará de manera concisa. El módulo 610 de control de decodificador hace que los mensajes V2C se lean desde la memoria 630 de mensajes V2C en orden de zócalo de restricción y se entreguen al procesador 640 de nodo de restricción. El módulo 610 de control de decodificador señaliza a la memoria 650 de mensajes de aristas V2C un identificador de mensajes (por ejemplo, una ubicación de memoria) indicando qué mensaje debe ser leído en ese momento. Los mensajes para un nodo de restricción dado se procesan por el procesador 640 de nodo de restricción. El procesador 640 de nodo de restricción recibe una señal, la señal de reloj de nodo, desde el módulo 610 de control de decodificador que indica los límites de nodo. Los mensajes C2V salientes se emiten en orden de zócalo de restricción, es decir, correspondiente al orden de aristas de los mensajes entrantes, y estos mensajes se almacenan en la memoria 650 de mensajes de aristas C2V.

La actualización de nodo variable y la actualización de nodo de restricción comprenden juntas una iteración completa. (Como se señaló anteriormente, el procesamiento de nodo variable y de restricción puede ocurrir al mismo tiempo).

A continuación se describirán diversas características del procesador de nodo de la presente invención en el contexto del sistema 600 decodificador LDPC ejemplar mostrado en la figura 6. Se va a describir un conjunto específico de cálculos y formatos de mensaje que pueden usarse para la decodificación LDPC según la presente invención. Debe entenderse que los mismos cálculos, descritos en la implementación de procesadores de nodo en el decodificador en serie de la figura 6, pueden usarse en implementaciones paralelas en las que una pluralidad o todos los mensajes para un nodo dado llegan al procesador simultáneamente y se procede a la vez al procesamiento de nodo de la pluralidad o todos los mensajes. Además, el procesador de nodo puede replicarse para proporcionar procesamiento paralelo de varios nodos a la vez.

Se da una descripción precisa de un algoritmo basado en mensajes de cinco bits ejemplar que materializa la presente invención. Los mensajes y valores suaves de entrada son en forma de razón de verosimilitud logarítmica y están cuantificados en múltiplos de δ=ln2 según la invención. De esta manera los cinco bits de un mensaje representan posibles valores enteros {-15,-14,…,-1, 0,1,…,15}. En la práctica es conveniente representar estos valores suaves como “signo y magnitud”. Un signo adopta el valor de 0 ó 1 indicando el valor preferido, por ejemplo, la decisión dura asociada, para el bit asociado. La magnitud indica la fiabilidad del signo: grandes magnitudes indican alta fiabilidad. (Generalmente es una buena idea limitar las magnitudes de valores de entrada suaves al intervalo {0,1,…, M} para algún M<15. Obsérvese también que en esta representación hay en realidad dos valores de magnitud ‘0’, uno con el signo 0 y otro con el signo 1). Por lo tanto se escribirá un mensaje como un par (m_p,m_r) donde m_p indica el bit de signo y m_r indica la fiabilidad, un valor de cuatro bits no negativo en el ejemplo. Dado un mensaje de este tipo, se usará m para indicar la razón de verosimilitud logarítmica ajustada a escala representativa, es decir, $m=(-1)^{mp}m_r$. La fórmula de actualización para los nodos variables viene dada por:

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

Cuando el mensaje saliente m V^{2C}(j) tiene el valor 0, el signo puede escogerse arbitrariamente. Cuando la magnitud del mensaje saliente excede de 15, se satura, por ejemplo, se ajusta a 15.

Las figuras 7 y 8 ilustran una implementación ejemplar de esta regla para el decodificador en serie descrito anteriormente. La figura 7 ilustra una implementación de un procesador 700 de nodo variable apropiado para el decodificador representando en la figura 6. Los mensajes (m) para un nodo variable particular llegan en secuencia. Cada mensaje, *m*, incluye K bits. Cada mensaje recibido se suma por el sumador 710 con una suma previa, es decir, una suma de mensajes acumulada, emitida por el elemento 712 de retardo en unidades. La suma de mensajes acumulada suministrada al sumador 710 se reinicia a “0” con la recepción de un mensaje correspondiente a un nuevo nodo. El valor de entrada suave recibido y se suma en el sumador 720 a la suma de mensajes acumulada emitida por el retardo 712, para formar una nueva suma acumulada SUM. Cuando todos los mensajes correspondientes al nodo que está procesándose se han sumando para formar una suma total acumulada completa, el valor SUM se enclava y se almacena en un circuito 730 de enclavamiento (*latch*). Este enclavamiento ocurre como resultado de una señal de reloj de nodo externa cuya sincronía se controla para garantizar que la suma completa se enclava. La señal de reloj de nodo se determina en función del grado del nodo.

Cada mensaje recibido se pasa también a través de una línea de retardo que incluye un elemento 760 de retardo variable cuya duración corresponde al grado del nodo que está implementándose.

El primer mensaje asociado al nodo variable particular que está procesándose emerge de la línea de retardo y el valor del mensaje se resta a la suma total enclavada almacenada en el circuito 730 de enclavamiento por el sumador 740. El resultado saliente se pasa a continuación a través de un operador 770 de saturación para garantizar que el valor del mensaje saliente está en el intervalo deseado, por ejemplo, limitado a K bits. Los elementos 742 y 772 de retardo en unidades situados antes y después del operador 770 de saturación se usan para sincronizar las operaciones de procesamiento de nodo variable con las operaciones de procesamiento de nodo de restricción.

La figura 8 ilustra una implementación de un procesador 800 de nodo de restricción apropiado para su uso en el decodificador representando en la figura 6. El procesador 800 de nodo de control incluye un circuito 801 divisor, un circuito 802 de procesamiento de signo, un circuito 804 de procesamiento de magnitud y un circuito 899 combinatorio. Como en el caso del procesador 700 de nodo variable, los mensajes m para un nodo de control particular llegan en secuencia. La actualización de nodo de restricción está separada en procesamiento de actualización de signo y magnitud, utilizándose el divisor 801 para separar el bit de signo de la información de magnitud en el mensaje. El único bit de signo del mensaje recibido se divide por el circuito 801 separador y se suministra a un circuito 802 de procesamiento bits de signo para su procesamiento. Los restantes K-1 bits del mensaje de K bits, m, se suministran a un circuito 804 de procesamiento de mensajes. Además, señales de reloj de nodo y de grado de nodo se suministran entradas correspondientes de ambos circuitos 802, 804 de procesamiento

de signo y magnitud.

Primero se describirá el procesamiento de actualización de signo realizado por el circuito 802 de procesamiento de signo. Los bits de signo entrantes se retardan un ciclo de reloj por el elemento 803 de retardo en unidades antes de suministrarse a una entrada del circuito 810 lógico de O exclusivo (XOR). El circuito 810 XOR realiza una suma o exclusivo o *módulo 2* del bit de signo entrante y un resultado XOR previo que se ha retardado por un segundo elemento 815 de retardo en unidades antes de suministrarse a una segunda entrada del circuito 810 XOR. De esta manera, la aplicación de la operación XOR a todos los bits de signo correspondientes al nodo que está procesándose produce un resultado o producto XOR total, SUM_{sign}, a través de un procesamiento iterativo. El valor SUM_{sign} se almacena en el circuito 830 de enclavamiento bajo el control de la señal 830 de reloj de nodo para la posterior operación XOR con bits de signo retardados correspondientes al nodo. El MUX 813 se usa para emitir el valor 0 al que va a aplicarse una operación XOR con el bit de signo del primer mensaje correspondiente a un nodo. En otros momentos envía el resultado XOR retardado.

Además de pasarse a través de la ruta XOR comenzando con el elemento 803 de retardo, cada bit de signo se pasa también a través de una línea de retardo que incluye el elemento 820 de retardo variable. El retardo del elemento 820 se controla por la señal de grado de nodo de modo que la duración del retardo impuesto al bit de signo corresponderá al grado del nodo. Cuando todos los bits de signo correspondientes a un nodo se hayan combinado, la suma total (o exclusivo), que se indica como SUM_{sign}, se almacena en el circuito 830 de enclavamiento. El enclavamiento del valor SUM_{sign} ocurre bajo la dirección de una señal de reloj de nodo externa como en el proceso de actualización de nodo variable explicado anteriormente. Esta señal también se usa para controlar la salida del MUX 813.

Como en la actualización de nodo variable, los bits de signo emergen de la línea de retardo en secuencia. El valor del bit de signo retardado se resta (operación o exclusivo) a SUM_{sign} en un sumador 840. El resultado constituye el signo del mensaje saliente. Obsérvese que algunos elementos 841, 843 de retardo adicionales se han insertado en el procesador 802, para mantener la actualización de signo sincronizada con la actualización de magnitud realizada por el circuito 804.

Ahora se describirá el procesamiento de actualización de magnitud realizado por el circuito 804 de procesamiento de magnitud. En la primera etapa, las magnitudes entrantes $m_r^{V2C}(i)$, representadas usando (K-1) bits, se convierten en valores en el dominio de restricción. Esto equivale a sustituir el mensaje $m_r^{V2C}(i)$ por el valor usando el circuito 850 de transformación. El objetivo de este proceso de transformación es representar los valores de mensaje en una forma en la que el procesamiento de restricción pueda implementarse a través del uso de simples sumas y restas. En la práctica, esta transformación puede implementarse como una operación de desplazamiento: el valor C se almacena en forma binaria y se desplaza a la derecha para obtener el mensaje transformado. Para el ejemplo del decodificador de 5 bits el valor C que ha resultado ser efectivo viene dado por 6000 en notación hexadecimal.

Véase la figura 9 para un diagrama de flujo detallado que explica el proceso 900 de transformación directa para cambiar de la representación en el dominio de nodo variable más compacta, por ejemplo, 4 bits, de la parte de magnitud de un mensaje a la representación en el dominio de nodo de restricción más larga, por ejemplo, 15 bits, de la parte de magnitud de un mensaje. En la etapa 906, una constante C, usada para convertir entre los dominios de nodo variable y de restricción, se desplaza a la derecha el valor decimal (número) de lugares indicado por la parte de magnitud del mensaje recibido. Además, se insertan ceros en las posiciones de bit a la izquierda (bits de alto orden) desocupadas como resultado de la operación de desplazamiento.

Por ejemplo, supóngase una entrada en la etapa 902 de los 4 bits (0010) que tienen un valor decimal de 2. En la etapa 906, la constante binaria C=110000000000000 se desplazará dos lugares hacia la derecha y los bits más a la izquierda se rellenan con ceros dando como resultado un valor de magnitud en el dominio de nodo de restricción (binario) de 001100000000000.

En la etapa 908, la parte 910 de magnitud (por ejemplo, 15 bits) del mensaje ahora en forma de nodo de restricción se emite para su procesamiento adicional. El valor de salida representa un valor de magnitud transformado apropiado para el proceso de nodo de restricción a través de operaciones de suma.

Refiriéndose una vez más a la figura 8 puede verse que las magnitudes transformadas se pasan a través del elemento 851 de retardo en unidades antes de suministrarse al sumador 860 y al elemento 880 de retardo variable.

Una suma de las magnitudes de mensajes de aristas transformadas se crea mediante el uso del sumador 860 para sumar la salida de mensaje de arista por el elemento 851 de retardo a la salida retardada del sumador 860, creando de este modo una suma acumulada, SUM_{mag}. La suma acumulada, SUM_{mag}, se almacena en el circuito 870 de enclavamiento tras la recepción de la señal de reloj de nodo indicando que todos los mensajes de arista correspondientes al nodo se han sumado entre sí. La señal de reloj de nodo también se usa para controlar el MUX 863 para que emita un “0” cuando el primer mensaje correspondiente a un nodo se suministra al sumador 860. De esta manera, el MUX 860 permite el reinicio de la suma acumulada al inicio del procesamiento de mensajes correspondientes a cada uno de una pluralidad de nodos.

Las magnitudes de mensajes de aristas transformadas retardas emergen posteriormente de la línea 880 de retardo y sus valores se restan a la suma almacenada SUM_{mag} en un sumador 890 antes de suministrarse al elemento 891 de retardo en unidades. Entonces, se realiza una transformación inversa sobre la magnitud saliente retardada obtenida del elemento 891 de retardo por el circuito 895 de transformación.

Este segundo circuito de transformación, que realiza una transformación desde la representación de mensaje de nodo de restricción a la de nodo variable, en una realización ejemplar, funciona como sigue. Un codificador de prioridad determina la ubicación del primer 1 en la representación binaria del mensaje de magnitud en el dominio de restricción. Esta ubicación se expresará desde la derecha. Por lo tanto, v indica una magnitud en el dominio de restricción y 1(v) indica su "ubicación". En general, si el valor entero decimal de v está en el intervalo inclusivo de 2^j y 2^{j+1}-1, entonces 1(v)=j. Si v es 0 entonces 1(v) es 0. Si 1(v) es mayor que o igual a 15 (en el caso de magnitudes de mensaje de nodo variable de 4 bits) entonces la magnitud saliente se ajusta a 0. De lo contrario, la magnitud saliente es 15-1(v).

La figura 10 es un diagrama de flujo que ilustra el proceso 1000 de transformación realizado en la etapa 895 que implica la conversión del valor de magnitud de mensaje desde el dominio de nodo de restricción de vuelta a la representación en el dominio de nodo variable. Obsérvese que en este punto la magnitud del mensaje puede incluir más bits (por ejemplo, el mensaje puede ser de 20 bits de longitud) que a la salida de la etapa 850 (por ejemplo, donde el mensaje era de 15 bits de longitud), debido a las operaciones de suma y resta de mensajes realizadas en las etapas 860 y 891.

En la etapa 1004, se recibe la parte 1002 de magnitud de un mensaje en forma de dominio de nodo de restricción y la posición del primer "1" desde la izquierda se determina contando las posiciones de bit desde la derecha. Por ejemplo, si la parte 1002 de magnitud era (00000010000000000100) un codificador de prioridad puede usarse para determinar que el primer bit 1 desde la izquierda ocurre en la posición de bit 14 medida desde la derecha.

A continuación, en la etapa 1006, la posición de bit determinada se compara con el valor máximo que puede representarse por el número de bits usados para representar las magnitudes de mensaje en forma de dominio de nodo variable. Por ejemplo, si se usaron 14 bits para magnitudes de mensaje en el dominio de nodo variable, el valor máximo será 15. En tal caso, si la posición de bit determinada fue mayor que 15, la operación avanzará a la etapa 1008. De lo contrario, la operación avanzará de la etapa 1006 a la etapa 1010.

En la etapa 1008 todos los bits en la representación de magnitud de mensaje en el dominio de nodo variable, por ejemplo, 4 bits, se ajustan a cero generando de ese modo la forma de dominio de nodo variable de la magnitud 1002 de mensaje de nodo de restricción recibida.

A continuación se describirá el procesamiento en la etapa 1010. En la etapa 1010, la magnitud de mensaje en forma de dominio variable se genera restando el número de la posición de bit identificada obtenida en la etapa 1006 al valor más grande que puede representarse usando el número de bits usado para representar la magnitud de mensaje en forma de dominio de nodo variable. Por ejemplo, suponiendo una representación de magnitud de 4 bits en el dominio de nodo variable, en la etapa 1010 la posición de bit de la etapa 1006, por ejemplo, 14 en el caso del ejemplo, se restará a 15 dando como resultado una magnitud de mensaje de nodo variable de (15-14=1) uno representada de manera binaria como (0001).

En la etapa 1012 la magnitud 1014 de mensaje ahora en la forma de magnitud en el dominio de nodo variable se emite, por ejemplo, al elemento 897 de retardo.

En algunas implementaciones del procesador de nodo de restricción será más conveniente almacenar las fiabilidades en la línea 880 de retardo en forma de dominio de nodo variable en lugar de en forma de dominio de restricción. Esto puede reducir la complejidad puesto que la forma de dominio variable requiere una cantidad significantemente menor de bits que la forma de dominio de restricción. Para modificar la figura 8 para esta realización, el bloque 850 funcional se replicará a la salida de la línea 880 de retardo y la entrada a la línea de retardo se tomará antes del bloque 880 funcional en lugar de después.

Se apreciará que, aunque se usan mensajes de nodo variable de 5 bits en los ejemplos anteriores (1 bit de signo y 4 bits de magnitud), las reglas descritas pueden aplicarse a mensajes que tengan menos o más bits simplemente disminuyendo o aumentando el intervalo usado y ajustando las constantes según sea necesario.

También se apreciará que en algunos casos un espaciado de 2ln2 para la razón de verosimilitud logarítmica puede ser más deseable que el espaciado ln2 ejemplar mencionado. Este caso se usará normalmente con mensajes constituidos por menos bits. El cambio de k2 a 2ln2 cambiará las reglas de actualización explicadas anteriormente en los nodos de restricción pero no en los nodos variables. La modificación en los nodos de restricción equivale a darse cuenta de que todos los mensajes son como en el caso del espaciado ln2 pero sólo se permiten magnitudes de valor par y, en este caso, el último bit de la magnitud será siempre 0 y no es necesario pasarlo, y en varias realizaciones no se pasa, como parte del mensaje.

En algunos otros casos, un espaciado de ½ ln2 puede ser preferible. Las figuras 11 y 12 representan los procesos 1100, 1200 de transformación directo e inverso para las realizaciones con espaciado ½ ln2, que

corresponden a las transformaciones mostradas en las figuras 9 y 10, respectivamente, para la realización con espaciado ln2. Una vez más, las reglas de actualización, por ejemplo, las operaciones 850 y 895 de transformación, cambian en los nodos de restricción pero no en los nodos variables. Cuando el bit menos significativo (bit más a la derecha en este caso dado el orden de bits) de la magnitud es 0, la transformación al dominio de nodo de restricción tiene lugar como en el caso del espaciado ln2 salvo porque el bit menos significativo, por ejemplo, el quinto bit en una representación de magnitud de 5 bits, de la magnitud entrante se elimina, es decir, el valor de magnitud de mensaje se desplaza a la derecha en uno, o, de manera equivalente, se divide por 2. Cuando el bit menos significativo es un 1 se realiza la misma operación pero la constante C usada en la transformación será diferente. En tal caso, la constante C será sustituida por un valor ligeramente más pequeño situado entre C y C/2. El procesamiento de nodo de restricción procede como antes hasta el punto de la transformación inversa usando los bits restantes de la parte de magnitud del mensaje que está procesándose.

La figura 11 ilustra el proceso 1100 de transformar una parte 1102 de mensaje de magnitud en forma de dominio de nodo variable a la forma de dominio de nodo de restricción en el caso en el que se usa el espaciado ½ ln2. Como se ilustra en la etapa 1104, el LSB, por ejemplo, el bit 5, se escinde de la parte de magnitud del mensaje y se suministra a la etapa 1100 mientras los bits restantes, por ejemplo, los 4 bits de orden superior se suministran a la etapa 1106. En la etapa 1100, el LSB se compara con cero para determinar qué valor debe usarse para la constante C en el proceso de transformación. Si el LSB=0, la operación avanza hasta la etapa 1113 donde C se ajusta a un primer valor 110000000000000. Sin embargo, si el LSB=1, la operación avanza desde la etapa 1100 hasta la etapa 1114 donde C se ajusta a un segundo valor 100001000000000. El valor C se suministra desde la etapa 1113 ó 1114 a la etapa 1106 que también recibe los bits restantes de orden superior, por ejemplo, 4 bits, de la parte 1102 de magnitud de mensaje.

En la etapa 1106 la constante C se desplaza a la derecha el numero de lugares especificado por el valor decimal de los bits restantes (por ejemplo, 4 bits) de la magnitud de mensaje y se insertan ceros en las posiciones de bit de la izquierda desocupadas como resultado de la operación de desplazamiento. Entonces en la etapa 1108 la magnitud 1110 de mensaje (15 bits), ahora en la forma de magnitud en el dominio de nodo de restricción, se emiten a, por ejemplo, el elemento 851 de retardo.

En el caso de ½ ln2, la transformación inversa, por ejemplo, representación en el dominio de nodo de restricción a representación en el dominio de nodo variable, implica una etapa adicional más allá de la presente en el caso del espaciado ln2 que se usa para determinar el bit menos significativo del mensaje saliente. El valor obtenido usando la regla de espaciado ln2 proporciona los otros bits, de orden superior, en el mensaje.

Para determinar el valor del bit menos significativo en el caso ½ ln2, se examinan algunos bits a la derecha del primer 1 de la magnitud en el dominio de nodo de restricción. Se usa un umbral para determinar si el bit menos significativo será 0 ó 1.En una realización ½ ln2 particular, el valor en el dominio de nodo de restricción v se compara con el valor umbral $t\,2^{l(v)}$ donde t es alguna constante predeterminada apropiada. Si v es mayor que $t\,2^{l(v)}$ entonces el LSB se ajusta a 0, de lo contrario se ajustará a 1. Se apreciará que hay muchas implementaciones alternativas usando la misma regla de umbral.

La figura 12 ilustra el proceso 1200 de conversión de una parte 1202 de magnitud de un mensaje en forma de dominio de nodo de restricción a la forma de dominio de nodo variable. El proceso comienza en la etapa 1204 en la que se determina la ubicación del primer “1” desde la izquierda medida desde la derecha. Un codificador de prioridad puede usarse para realizar esta operación. Entonces, en la etapa 1206 se realiza una determinación de si la posición de bit identificada es mayor que el número máximo que puede representarse por el número de bits, excluyendo el LSB, usado para representar la magnitud de mensaje en el dominio de nodo variable. Por ejemplo, suponiendo que se utilizan 5 bits para representar la magnitud en el dominio de nodo variable, 5 bits - 1 para el LSB = 4 bits lo que permite un valor decimal máximo de 15. En tal caso, en la etapa 1206, se realiza una determinación de si la posición de bit identificada supera el número máximo de 15. Si la respuesta es afirmativa, la operación avanza a la etapa 1208. De lo contrario, la operación avanza a la etapa 1212.

El procesamiento avanza a la etapa 1208, como un resultado de una determinación de que la magnitud de mensaje debe ajustarse a cero. En la etapa 1208, todos los bits en la representación de magnitud de mensaje en el dominio de nodo variable se ajustan a cero para generar la magnitud de mensaje (por ejemplo, 00000) en forma de dominio de nodo variable. Con la transformación completa, la operación avanza desde la etapa 1208 a la etapa 1214.

En la etapa 1210, que representa una trayectoria de de procesamiento alternativa a la etapa 1208, el número identificado, es decir, número de la posición de bit, se resta al valor más grande que puede representarse por el número de bits no LSB de una magnitud de mensaje en forma de dominio de nodo variable. Por ejemplo, suponiendo una representación de magnitud de 5 bits en forma de dominio de nodo variable, hay 4 bits además del LSB. El número más grande que puede representarse por 4 bits es 15. En consecuencia, en una realización de este tipo en la etapa 1210, el número de la posición de bit identificada se restará a 15 para producir los 4 bits de orden superior del valor de magnitud en el dominio de nodo variable.

En la etapa 1212, el LSB, por ejemplo, el quinto bit del valor de magnitud ejemplar de 5 bits, se determina a partir del valor de uno o más bits a la derecha del “1” más a la izquierda presente en el valor de magnitud en la

representación 1202 en el dominio de nodo de restricción de la magnitud de mensaje.

En la etapa 1214, la magnitud 1216 de mensaje generada, por ejemplo, 5 bits, ahora en la forma de nodo variable se emite, por ejemplo, al elemento 897 de retardo en unidades.

Las figuras 7 y 8 incluyen una y dos instancias, respectivamente, de una estructura básica que se describe adicionalmente en referencia a la figura 13. Ha de recordarse que el cálculo dominante para las actualizaciones de mensaje tiene la forma simplificada:

$$m_{out}(j) = \Sigma_{i=1}^{d} \; m_{in}(i) - m_{in}(j)$$

y que es deseable tener una estructura segmentada eficaz que pueda producir un mensaje de arista saliente por cada ciclo de reloj. El diseño de hardware de la presente invención da una debida consideración a las siguientes observaciones y admite las características enumeradas:

no hay dependencia explícita entre datos de entrada y salida, lo que permite que la segmentación sea lo suficientemente profunda para permitir una tasa de reloj muy alta;

la segmentación es capaz de mantener su eficacia mientras procesa nodos de grado variable; la suma y la resta en la fórmula anterior pueden generalizarse a cualquier operación que tenga una inversa y siga una ley asociativa (módulo suma/resta, multiplicación/división, etc.); la segmentación puede incluir fases adicionales de procesamiento anterior y posterior tales como la transformación de la función, saturación, elementos de retardo, etc.

En la figura 13 se ilustra un sistema 1300 de procesamiento de mensajes, que representa una estructura de segmentación propuesta generalizada. El sistema 1300 de procesamiento recibe secuencialmente mensajes de entrada “A”, uno por cada ciclo de reloj según determina una señal de reloj de arista usada para dirigir los diversos componentes del sistema. También recibe una señal de reloj de nodo y una señal de grado de nodo. La señal de reloj de nodo sirve como señal de entramado de nodo y se activa cuando se suministra un mensaje correspondiente a un nuevo nodo a la entrada de mensaje del módulo 1302 acumulador. Como se explicará a continuación, la señal de reloj de nodo se utiliza para controlar diversas operaciones, incluyendo la inicialización de una suma continua de mensaje y el enclavamiento de una suma total, generada para cada nodo. La señal de reloj de nodo se genera en función del grado, por ejemplo, el número de mensajes, correspondiente al nodo para el que está implementándose el procesamiento. La señal de grado de nodo indica el grado del nodo para el que está realizándose el procesamiento y por tanto el número de mensajes que corresponden al nodo. Como se explicará a continuación, la señal de grado de nodo se utiliza para controlar un elemento 1306 de retardo variable usado para retardar los mensajes recibidos. La operación del sistema 1300 se explicará más adelante con referencia al correspondiente flujo de datos mostrado en la figura 14. Para simplificar, se omite cualquier fase de procesamiento anterior y posterior y se suponen operaciones simples de suma y resta.

El sistema 1300 de procesamiento comprende dos fases computacionales, a veces denominadas módulos, es decir, un módulo 1302 de acumulación “A” y un módulo 1304 de resta “S”. Puesto que el módulo de resta genera mensajes de salida, a veces se denomina también módulo de generación de mensajes. El sistema 1300 también comprende una trayectoria de retardo de mensajes variable que incluye un elemento 1306 de retardo variable que emite mensajes retardados “D”.

El módulo 1302 de acumulación “A” recibe mensajes de entrada en secuencia y genera, para cada conjunto de mensajes correspondientes a un nodo, una suma total. El módulo 1302 de acumulación comprende un sumador 1310, un elemento 1312 de retardo en unidades, un mux 1314 y un circuito 1316 de enclavamiento de suma de nodo. El elemento 1312 de retardo en unidades se utiliza para almacenar una suma continua generada por el sumador 1310. El MUX 1314 suministra o bien un cero o bien la suma continua emitida por el elemento 1312 de retardo, a una de las entradas del sumador 1310. El MUX se controla por la señal de reloj de nodo para emitir el cero cuando se suministra el primer mensaje correspondiente a un nodo a la otra entrada del sumador 1310 y la suma continua B en cualquier otro momento. De esta manera, el sumador sumará los mensajes recibidos correspondientes a un nodo para generar una suma total para el nodo.

La señal de reloj de nodo también se utiliza para almacenar la suma continua en el circuito 1316 de enclavamiento de suma de nodo. En el momento en el que la señal de reloj de nodo se activa y el valor se enclava, la suma continua representa una suma total para el nodo.

El módulo 1304 de resta o de generación de mensajes “S” recibe como su entrada la suma total generada por el módulo 1302 de acumulación y los mensajes de entrada retardados por el elemento 1306 de retardo variable. La fase 1304 “S” resta secuencialmente los mensajes de entrada retardados a la suma total “C” almacenada en el circuito 1316 de enclavamiento de suma de nodo produciendo mensajes de salida para un nodo, por ejemplo, el nodo N, un mensaje por cada ciclo de reloj. Los resultados de la operación de resta, un mensaje de salida “E” se almacena en un registro 1321 de salida antes de emitirse por . La operación de las fases 1302 “A” y 1304 “S” pueden superponerse o solaparse totalmente. Por ejemplo, mientras la fase “A” 1302 está realizando el

procesamiento para el nodo N+1, la fase “S” puede realizar el procesamiento para el nodo N.

La finalidad de la línea de retardo variable que incluye el elemento 1306 de retardo es suministrar mensajes de entrada originales retardados, representados como “D”, para el nodo N a la fase de resta “S” mientras que almacena los mensajes de entrada para el nodo N+1. El retardo, en unidades de ciclos de reloj de procesamiento, del elemento 1308 de retardo, aplicado a los mensajes correspondientes a un nodo, es igual al grado del nodo actual al que corresponden los mensajes. Por ejemplo, un mensaje correspondiente a un nodo de grado 4 se retardará cuatro ciclos de reloj por el elemento 1306 de retardo variable mientras que los mensajes correspondientes a un nodo de grado 2 se retardarán dos ciclos de reloj. Para soportar múltiples grados de nodo, en una realización, el elemento 1306 de retardo variable se implementa con al menos suficiente espacio de almacenamiento para almacenar tantos mensajes como el grado de nodo más alto que debe soportarse.

El uso del elemento 1306 de retardo variable y el mensaje D retardado ahorra ancho de banda de la memoria de mensajes mediante la eliminación de lecturas duplicadas necesarias para la operación segmentada solapada. Obsérvese que la línea de retardo, que incluye el elemento 1306 de retardo, puede implementarse en la práctica ya sea con control de valor de retardo externo (por ejemplo, registro de desplazamientos con derivación de salida variable) o con control de valor de retardo interno (por ejemplo, FIFO autónomo). El primer procedimiento puede ser preferible para decodificadores de vector, ya que la lógica de control de retardo puede compartirse entre múltiples nodos de procesamiento.

La estructura segmentada descrita anteriormente permite cambiar el grado de nodo sobre la marcha alterando tanto la frecuencia de la señal de reloj de nodo usada para la señal de entramado de nodo y cambiando la señal de grado que se utiliza para controlar el retardo de mensaje impuesto por el elemento 1306 de retardo. A través de estas señales de control, se puede cambiar fácilmente la profundidad de la segmentación. En este contexto, la profundidad de la segmentación puede interpretarse como el retardo desde el momento en que el primer mensaje entrante (para el nodo N) se introduce en el sistema 1300 para su procesamiento hasta el momento en que el primer mensaje E saliente (para el nodo N) aparece en la salida de la segmentación. La segmentación de profundidad variable puede ofrecer una ventaja significativa en cuanto al rendimiento cuando deben soportarse códigos LDPC irregulares ya que el número total de ciclos requeridos por iteración es igual al número de aristas en el grafo más la dispersión de grado (la diferencia entre los grados máximo y mínimo).

Por el contrario, un diseño de segmentación fija requeriría (grado máximo de arista) * (número de nodos) ciclos por iteración que puede ser considerablemente mayor que el número de aristas, especialmente cuando la dispersión de grado es grande.

La figura 14 ilustra un gráfico 1400 que incluye un ejemplo de los distintos valores, de A a D, presentes en el sistema 1300 en un momento en el que se produce un cambio de un nodo de grado 3 a un nodo de grado 4. Obsérvese que no hay ningún atasco de segmentación, por ejemplo, retardos con respecto a la llegada de los datos de entrada, debido al cambio en el grado de nodo.

La figura 14 ilustra el procesamiento asociado con mensajes correspondientes a cuatro nodos, el nodo 1 (n1), el nodo 2 (n2), el nodo 3 (n3) y el nodo 4 (n4). Las celdas en el gráfico de la figura 14 que incluyen una sola línea corresponden a n1. Las celdas que incluyen dos líneas corresponden a n2. Las celdas que incluyen tres líneas corresponden a n3. Además, las celdas que incluyen cuatro líneas corresponden a n4. Como se muestra en la primera columna 1401, los nodos n1 y n2 son de grado 3 mientras que los nodos n3 y n4 son de grado 4. Cada fila en el gráfico 1400 corresponde a un ciclo de reloj diferente, mientras que las columnas corresponden a valores diferentes según se indica.

La segunda columna 1402 enumera los mensajes recibidos A, en el orden en que se reciben, uno por cada ciclo de reloj. La tercera columna 1404 enumera la suma continua B, durante cada ciclo de reloj de procesamiento. Obsérvese que en la columna 1404, el indicador de nodo, por ejemplo, n1 del mensaje n1_e0, se omite en la columna 1404 para mayor brevedad. La cuarta columna 1406 enumera la suma total enclavada C generada para cada nodo como parte del procesamiento de mensajes. La quinta columna 1408 enumera el valor retardado D emitido por el elemento de retardo que se resta a una suma en la columna C para producir el mensaje de salida E. Los mensajes de salida generados se enumeran en la sexta columna 1410.

Por supuesto, debido al cambio de la profundidad de la segmentación, se producen ranuras vacías en el flujo de datos de salida. Si los nodos se clasifican por grado en orden monótono (creciente o decreciente), el número total de ranuras vacías por iteración es igual a la dispersión de grado y es muy pequeño en comparación con el número de mensajes procesados, por lo tanto se alcanza una utilización de segmentación muy alta.

Los procedimientos de decodificación LDPC anteriormente descritos y las implementaciones de procesador de nodo anteriormente descritas permiten realizar la decodificación LDPC en diferentes plataformas de hardware tales como disposiciones de puertas programables en campo o en un circuito integrado de aplicación específica. La presente invención es especialmente útil en estas configuraciones en las que los nodos de paralelismo simple y fáciles de implementar pueden aprovecharse explícitamente.

Numerosas variaciones adicionales de los procedimientos y aparatos de decodificación de la presente invención serán evidentes para los expertos en la técnica a la vista de la descripción anterior de la invención. Tales variaciones se considera que entran dentro del alcance de la invención.

REIVINDICACIONES

1. Un sistema de detección y/o corrección de errores, comprendiendo el sistema:

medios (1500) para generar valores de verosimilitud logarítmica cuantificados a múltiplos enteros de 1/2 ln 2 para producir valores y_0 de verosimilitud logarítmica cuantificados a partir de los valores recibidos;

un decodificador (358, 600) de paso de mensajes, acoplado a dichos medios para generar valores de verosimilitud logarítmica, para realizar operaciones de decodificación de paso de mensajes usando dichos valores y_0 de verosimilitud logarítmica cuantificados como valores de entrada;

en el que dicho decodificador (358, 600) de paso de mensajes incluye un dispositivo (1300) para realizar operaciones de procesamiento de nodo que comprende

un módulo (1302) acumulador para procesar, en secuencia, mensajes *m* de entrada correspondientes a una pluralidad de nodos, representando cada mensaje *m* de entrada un valor de verosimilitud logarítmica cuantificado a múltiplos enteros de 1/2 ln 2, recibiéndose un conjunto de mensajes de entrada por nodo, siendo el número de mensajes en un conjunto de mensajes correspondientes a un nodo igual al grado *d* de dicho nodo, donde d es un entero positivo distinto de cero, y para generar una suma de mensajes total a partir de un conjunto de mensajes *m* de entrada recibidos secuencialmente correspondientes a un nodo, generándose una suma de nodo total para cada conjunto recibido de mensajes de entrada, teniendo el módulo (1302) acumulador un elemento (1312) de retardo para almacenar una suma continua y un multiplexor (1314), MUX, adaptado para emitir la suma continua, o un cero cuando un primer mensaje en el conjunto de mensajes *m* de entrada correspondientes al nodo se suministra al módulo acumulador

una línea de retardo de mensajes que incluye una unidad (1306) de retardo controlable para almacenar cada uno de los mensajes *m* de entrada recibidos secuencialmente durante un período de tiempo directamente proporcional al grado d del nodo al que corresponde cada mensaje almacenado; y

un módulo (1304) de generación de mensajes para generar mensajes de salida correspondientes a un nodo a partir de la suma total correspondiente al nodo y los mensajes retardados correspondientes a dicho nodo, generando el módulo (1304) de generación de mensajes un mensaje de salida para cada mensaje de entrada recibido correspondiente al nodo.

2. El sistema según la reivindicación 1, en el que el módulo (1304) de generación de mensajes incluye:

un circuito (1320) de resta acoplado a dicho módulo (1302) acumulador y a la unidad de retardo controlable para restar cada valor de mensaje retardado correspondiente al nodo a la suma total correspondiente a dicho nodo.

3. El sistema según la reivindicación 2, en el que dicha unidad (1306) de retardo controlable incluye una entrada para recibir una señal de grado de nodo que indica el grado del nodo correspondiente a los mensajes recibidos por dicha unidad de retardo controlable.

4. El sistema según la reivindicación 3, en el que dicha unidad de retardo controlable se implementa como un dispositivo de almacenamiento de datos de tipo primero en entrar primero en salir.

5. El sistema según la reivindicación 2, en el que dicho módulo (1302) acumulador incluye:

un sumador (1310) que tiene una primera entrada para recibir mensajes *m* de entrada

un dispositivo (1316) de almacenamiento para almacenar la suma de nodo total generada; y

mediante el cual el elemento (1312) de retardo para almacenar la suma continua generada por dicho sumador (1310) y para emitir dicha suma continua almacenada está acoplado entre dicho sumador (1310) y dicho dispositivo (1316) de almacenamiento.

6. El sistema según la reivindicación 5, en el que dicho (1302) multiplexor (1314), acoplado a dicho elemento (1312) de retardo para recibir desde dicho elemento (1312) de retardo la suma continua almacenada y para emitir uno de la suma continua almacenada y un cero a una segunda entrada de dicho sumador (1310) bajo el control de una señal de reloj de nodo; y

en el que dicho dispositivo (1316) de almacenamiento además incluye una entrada de control para recibir dicha señal de reloj de nodo, almacenando dicho dispositivo (1316) de almacenamiento la suma continua almacenada que va a usarse como la suma de nodo total bajo la dirección de dicha señal de reloj de nodo.

7. El sistema según la reivindicación 6, que comprende además:

medios para activar la señal de reloj de nodo para hacer que el multiplexor emita un cero cuando un primer mensaje correspondiente a un nodo se recibe por dicho sumador (1310).

8. El sistema según la reivindicación 7, en el que la señal de reloj de nodo hace que dicho dispositivo (1316) de almacenamiento almacene la suma continua como la suma de nodo total al mismo tiempo que hace que el multiplexor (1314) emita un cero.

9. El sistema según la reivindicación 1, en el que la unidad (1306) de retardo controlable incluye una pluralidad de ubicaciones de almacenamiento de mensajes, siendo el número de ubicaciones de almacenamiento de mensajes en dicha pluralidad de ubicaciones de almacenamiento de mensajes al menos tan grande como el grado más alto de un nodo para el que dicho dispositivo va a realizar operaciones de procesamiento.

10. Un procedimiento de detección y/o corrección de errores, comprendiendo el procedimiento las etapas de:

generar valores de verosimilitud logarítmica cuantificados a múltiplos enteros de 1/2 ln 2 para producir valores y_0 de verosimilitud logarítmica cuantificados a partir de valores recibidos;

realizar una operación de procesamiento del decodificador de paso de mensajes usando dichos valores y_0 de verosimilitud logarítmica cuantificados como valores de entrada;

en el que dicha operación de procesamiento de decodificación de paso de mensajes incluye operaciones de procesamiento de nodo que comprenden las etapas de:

recibir secuencialmente mensajes *m* de entrada que van a procesarse correspondientes a una pluralidad de nodos, representando cada mensaje *m* de entrada un valor de verosimilitud logarítmica cuantificado a múltiplos enteros de 1/2 ln 2, recibiéndose un conjunto de mensajes de entrada por nodo, siendo el número de mensajes en un conjunto de mensajes correspondientes a un nodo igual al grado d de dicho nodo, donde d es un entero positivo distinto de cero, y generar una suma de mensajes total a partir de un conjunto de mensajes *m* de entrada recibidos secuencialmente correspondientes a un nodo, generándose una suma de nodo total para cada conjunto recibido de mensajes de entrada, almacenar una suma continua y multiplexar la suma continua con un cero cuando se recibe un primer mensaje en el conjunto de mensajes *m* de entrada correspondientes al nodo

retardar, usando una unidad de retardo controlable, cada mensaje *m* de entrada recibido individual durante un periodo de tiempo directamente proporcional al grado *d* del nodo al que corresponde dicho mensaje de entrada recibido individual; y

generar mensajes de salida correspondientes a un nodo a partir de la suma total correspondiente al nodo y los mensajes retardados correspondientes a dicho nodo, mediante el cual se genera un mensaje de salida para cada mensaje de entrada recibido correspondiente al nodo.

11. El procedimiento según la reivindicación 10, en el que dicha etapa de retardar cada mensaje m de entrada recibido individual incluye la etapa de:

recibir una señal de control que indica el grado del nodo al que corresponde un mensaje recibido para su procesamiento.

12. El procedimiento según la reivindicación 11, que comprende además la etapa de:

sumar cada mensaje m de entrada recibido a la suma continua de mensajes recibidos para generar dicha suma de mensajes total para cada conjunto de mensajes de entrada recibidos.

13. El procedimiento según la reivindicación 12, que comprende además la etapa de:

enclavar la suma continua en un dispositivo (1316) de almacenamiento, una vez que el último mensaje en un conjunto de mensajes de entrada recibidos se ha añadido a la suma continua, siendo la suma continua enclavada la suma de mensajes total.

14. El procedimiento según la reivindicación 13, que comprende además:

sumar un valor de entrada recibido a dicha suma continua antes de enclavar la suma continua en el dispositivo (1316) de almacenamiento.

15. El procedimiento según la reivindicación 13, que comprende además la etapa de inicializar la suma continua a cero cada vez que se recibe un mensaje correspondiente a un nodo diferente que el mensaje recibido inmediatamente precedente.

16. El procedimiento según la reivindicación 13, en el que dicho enclavamiento e inicialización se controlan mediante la misma señal de control.

17. El procedimiento según la reivindicación 12, que comprende además la etapa de emitir los mensajes de salida generados en secuencia, uno cada vez.

18. El procedimiento según la reivindicación 10, en el que cada mensaje de entrada recibido incluye un bit de signo y un valor de magnitud que, en combinación, comprende dicho valor cuantificado.

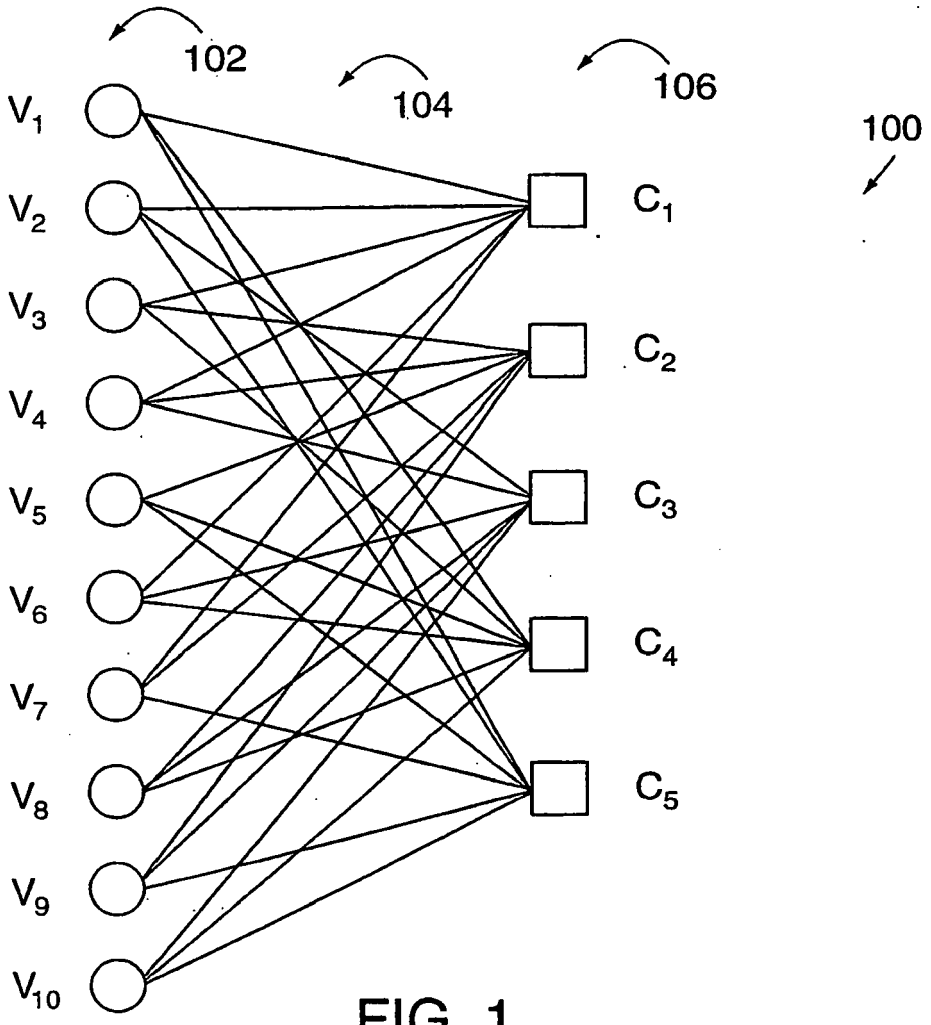

FIG. 1

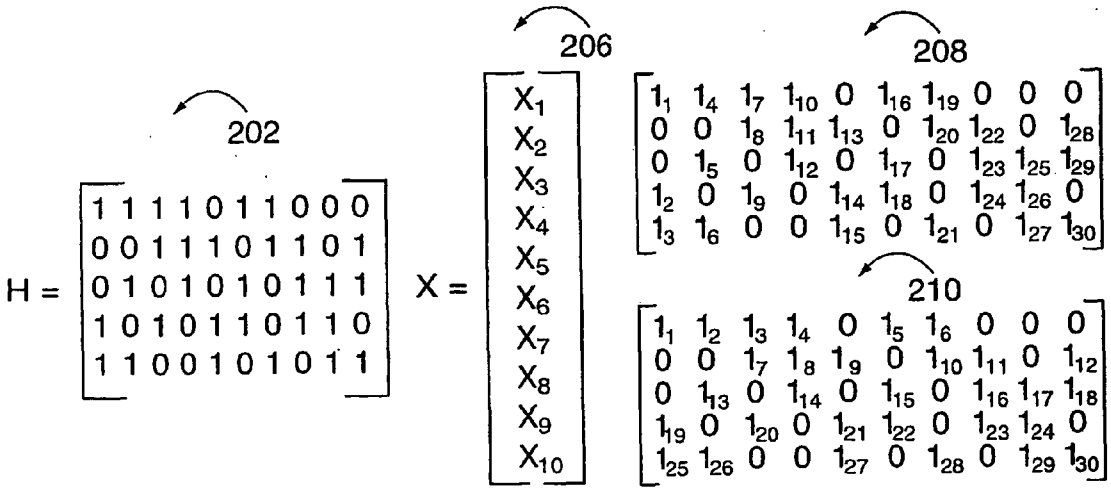
202
206
208
210
H =
X =
1 1 1 1 0 1 1 0 0 0
0 0 1 1 1 0 1 1 0 1
0 1 0 1 0 1 0 1 1 1
1 0 1 0 1 1 0 1 1 0
1 1 0 0 1 0 1 0 1 1
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
1 1 4 1 7 1 10 0 1 16 1 19 0 0 0
0 0 1 8 1 11 1 13 0 1 20 1 22 0 1 28
0 1 5 0 1 12 0 1 17 0 1 23 1 25 1 29
1 2 0 1 9 0 1 14 1 18 0 1 24 1 26 0
1 3 1 6 0 0 1 15 0 1 21 0 1 27 1 30
1 1 1 2 1 3 1 4 0 1 5 1 6 0 0 0
0 0 1 7 1 8 1 9 0 1 10 1 11 0 1 12
0 1 13 0 1 14 0 1 15 0 1 16 1 17 1 18
1 19 0 1 20 0 1 21 1 22 0 1 23 1 24 0
1 25 1 26 0 0 1 27 0 1 28 0 1 29 1 30

FIG. 2

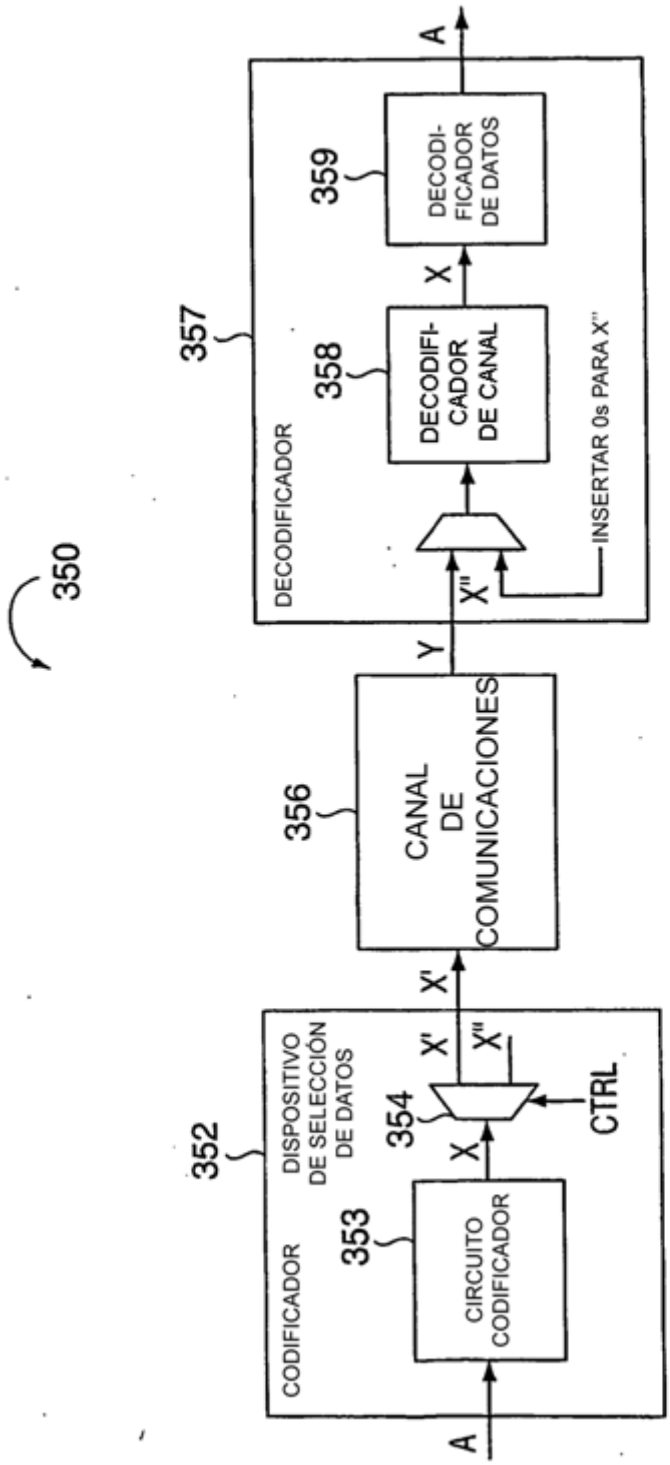
350
357
DECODIFICADOR
358
DECODIFI-
CADOR
DE CANAL
359
DECODI-
FICADOR
DE DATOS
A
X
X''
INSERTAR 0s PARA X''
Y
356
CANAL
DE
COMUNICACIONES
X'
352
CODIFICADOR
DISPOSITIVO
DE SELECCIÓN
DE DATOS
353
CIRCUITO
CODIFICADOR
354
CTRL
A

FIG. 3

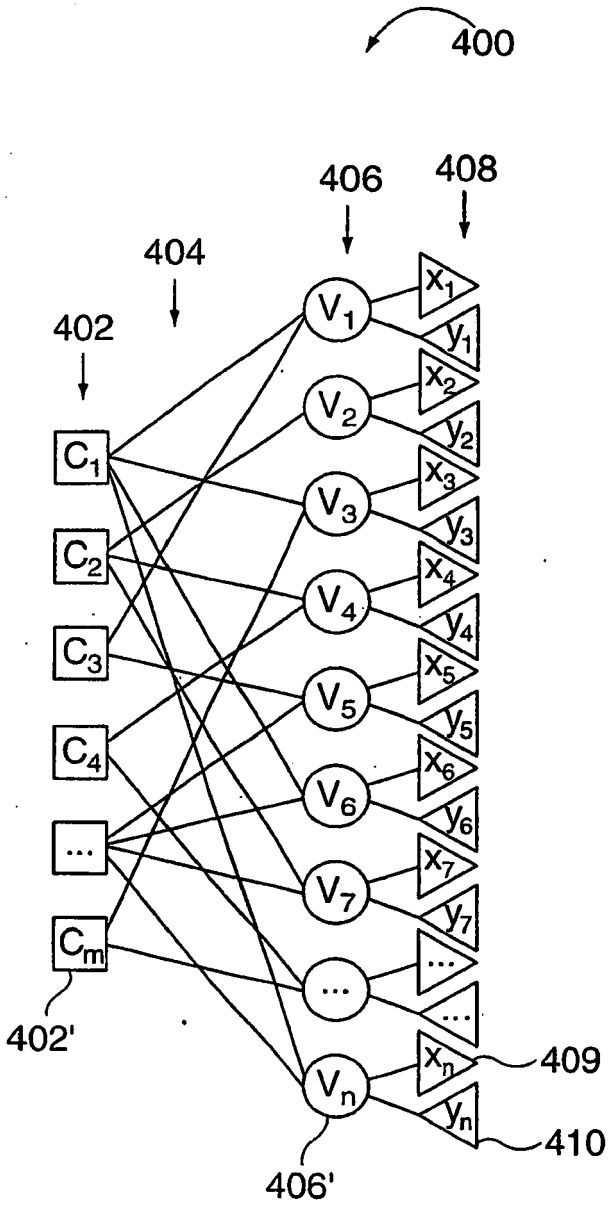
400
406
408
404
402
C1
C2
C3
C4
...
Cm
V1
V2
V3
V4
V5
V6
V7
...
Vn
x1
y1
x2
y2
x3
y3
x4
y4
x5
y5
x6
y6
x7
y7
...
...
xn
yn
402'
406'
409
410

FIG. 4

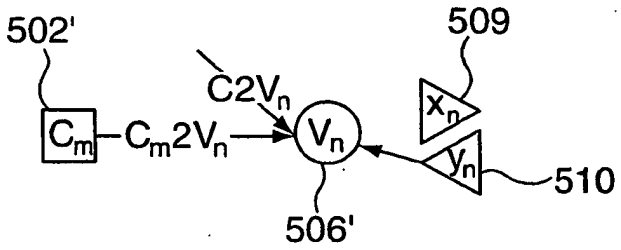

FIG. 5A

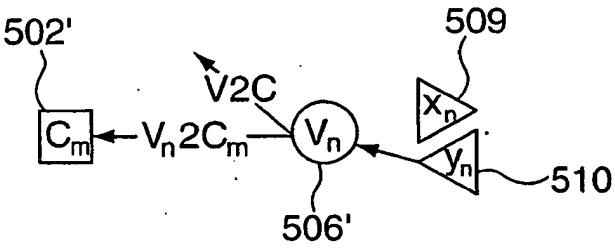

FIG. 5B

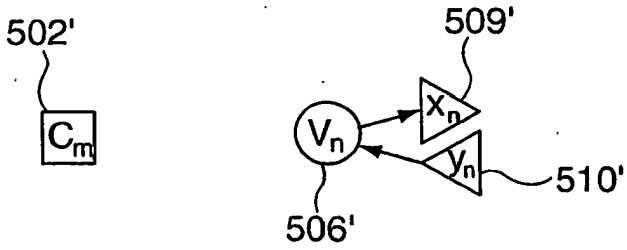

FIG. 5C

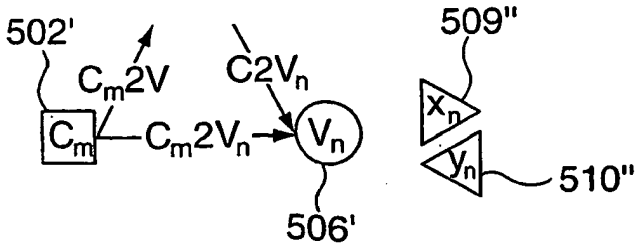

FIG. 5D

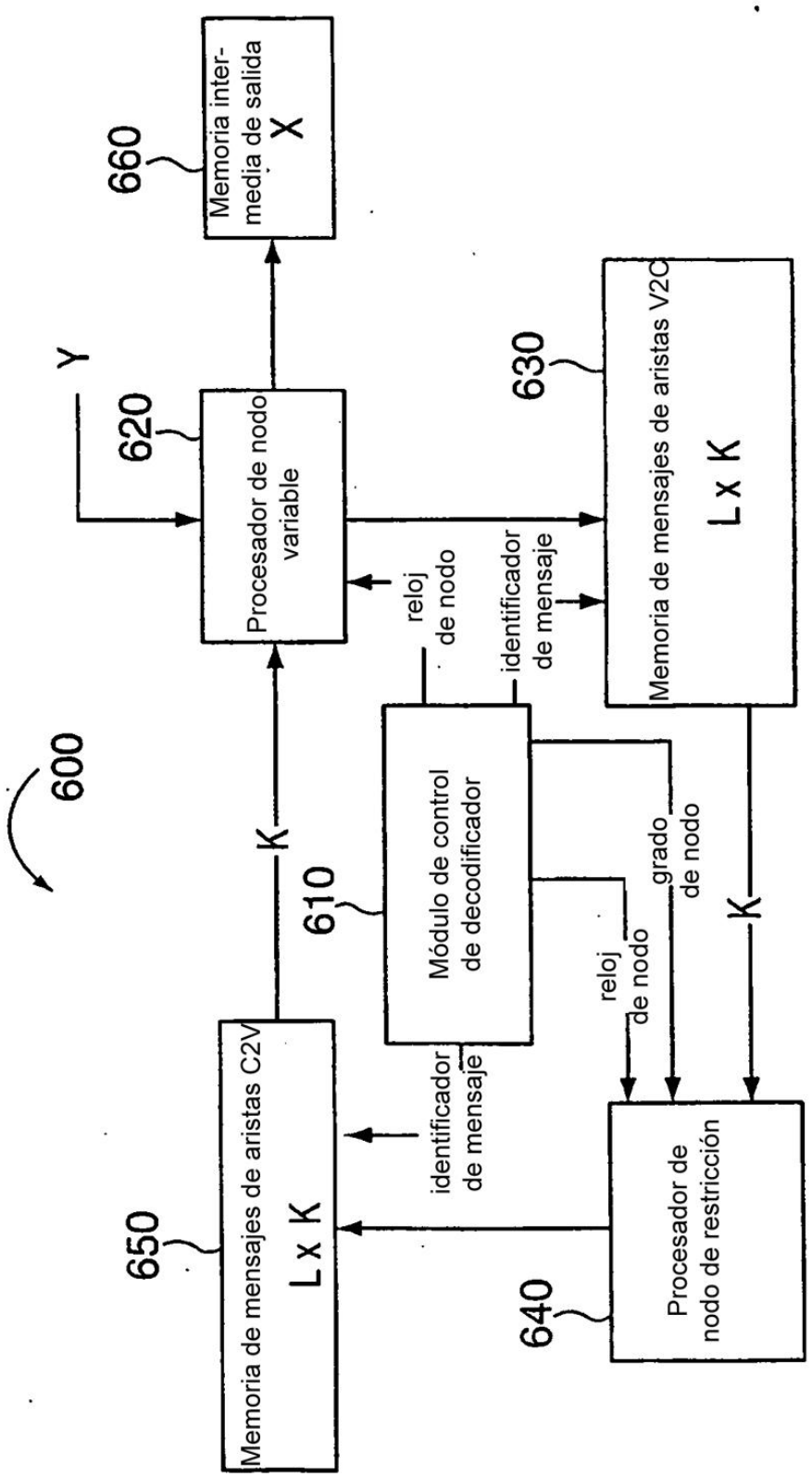
600
660
Memoria intermedia de salida
X
Y
620
Procesador de nodo variable
630
Memoria de mensajes de aristas V2C
L x K
reloj de nodo
identificador de mensaje
K
610
Módulo de control de decodificador
grado de nodo
reloj de nodo
K
identificador de mensaje
650
Memoria de mensajes de aristas C2V
L x K
640
Procesador de nodo de restricción

FIG. 6

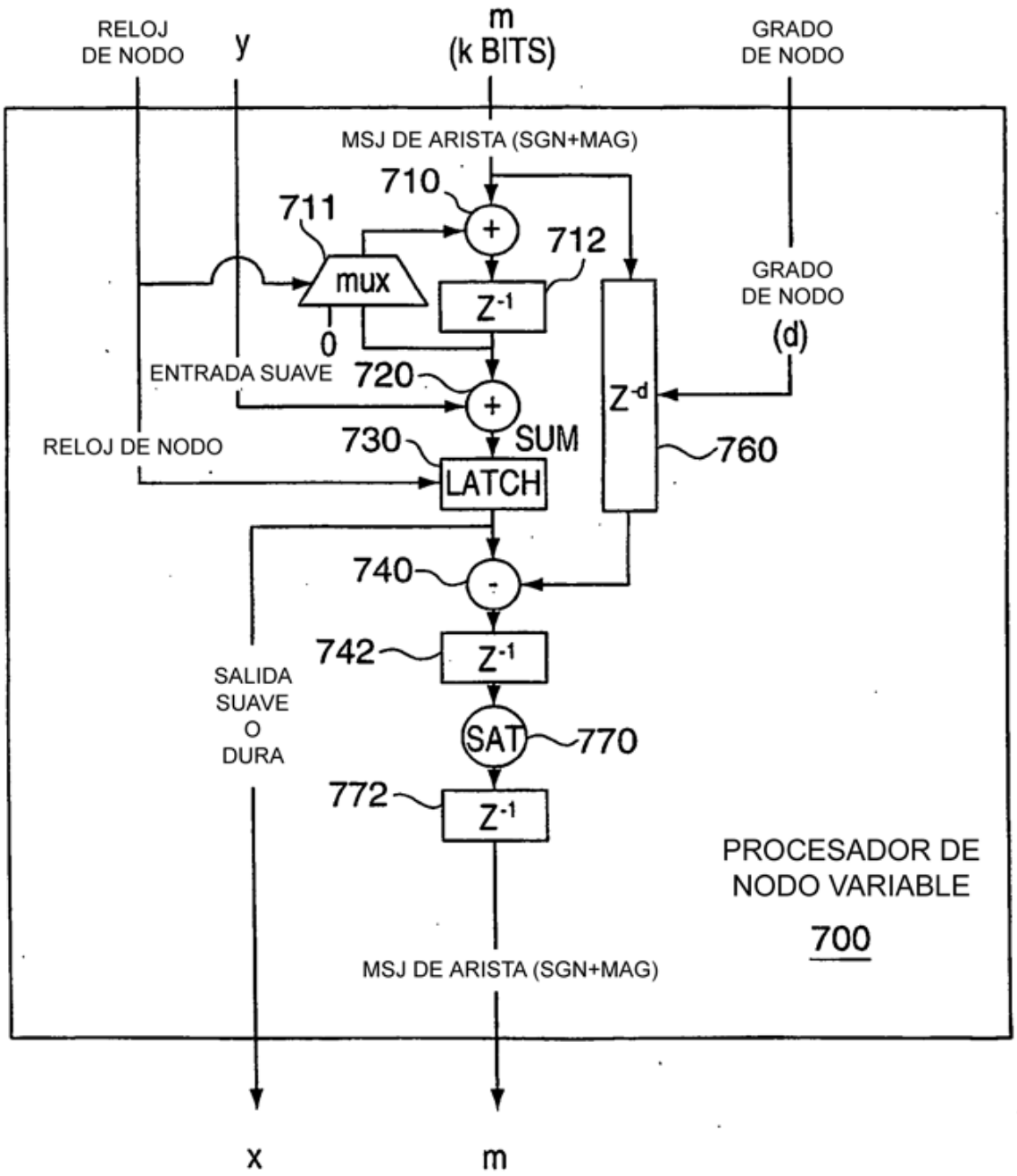
RELOJ DE NODO
y
m (k BITS)
GRADO DE NODO
MSJ DE ARISTA (SGN+MAG)
710
711
712
mux
Z-1
0
GRADO DE NODO
(d)
ENTRADA SUAVE
720
Z-d
RELOJ DE NODO
730
SUM
760
LATCH
740
742
Z-1
SALIDA SUAVE O DURA
SAT
770
772
Z-1
PROCESADOR DE NODO VARIABLE
700
MSJ DE ARISTA (SGN+MAG)
x
m

FIG. 7

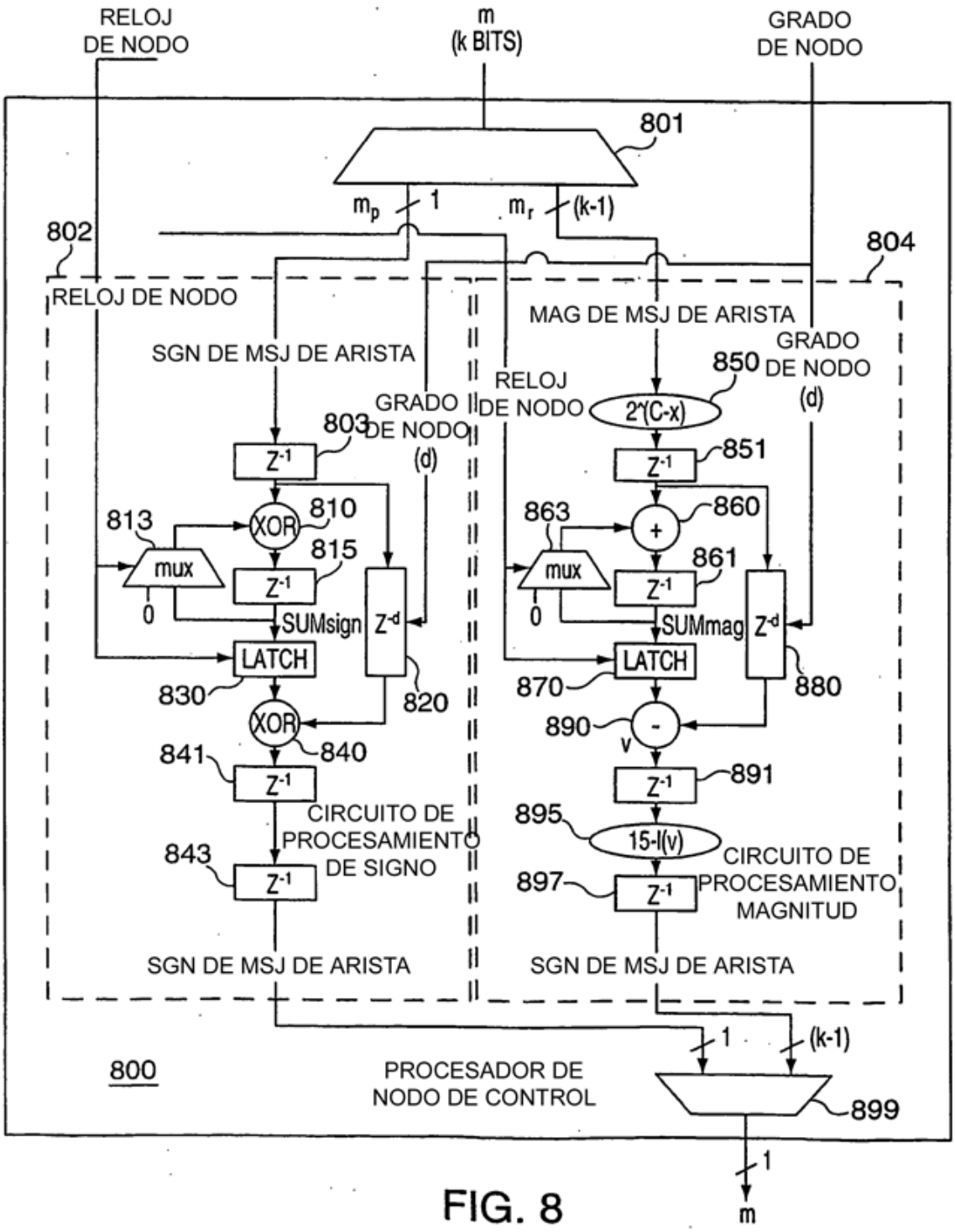
RELOJ DE NODO
m (k BITS)
GRADO DE NODO
801
m_p 1
m_r (k-1)
802
804
RELOJ DE NODO
SGN DE MSJ DE ARISTA
MAG DE MSJ DE ARISTA
GRADO DE NODO (d)
RELOJ DE NODO
GRADO DE NODO (d)
803
$2^{\wedge}(C-x)$
850
Z^{-1}
851
813
XOR
810
mux
815
863
+
860
861
0
SUMsign
Z^{-d}
SUMmag
LATCH
830
820
870
880
XOR
840
890
-
v
841
891
CIRCUITO DE PROCESAMIENTO DE SIGNO
895
15-l(v)
CIRCUITO DE PROCESAMIENTO MAGNITUD
843
897
SGN DE MSJ DE ARISTA
SGN DE MSJ DE ARISTA
1
(k-1)
800
PROCESADOR DE NODO DE CONTROL
899
1
m

FIG. 8

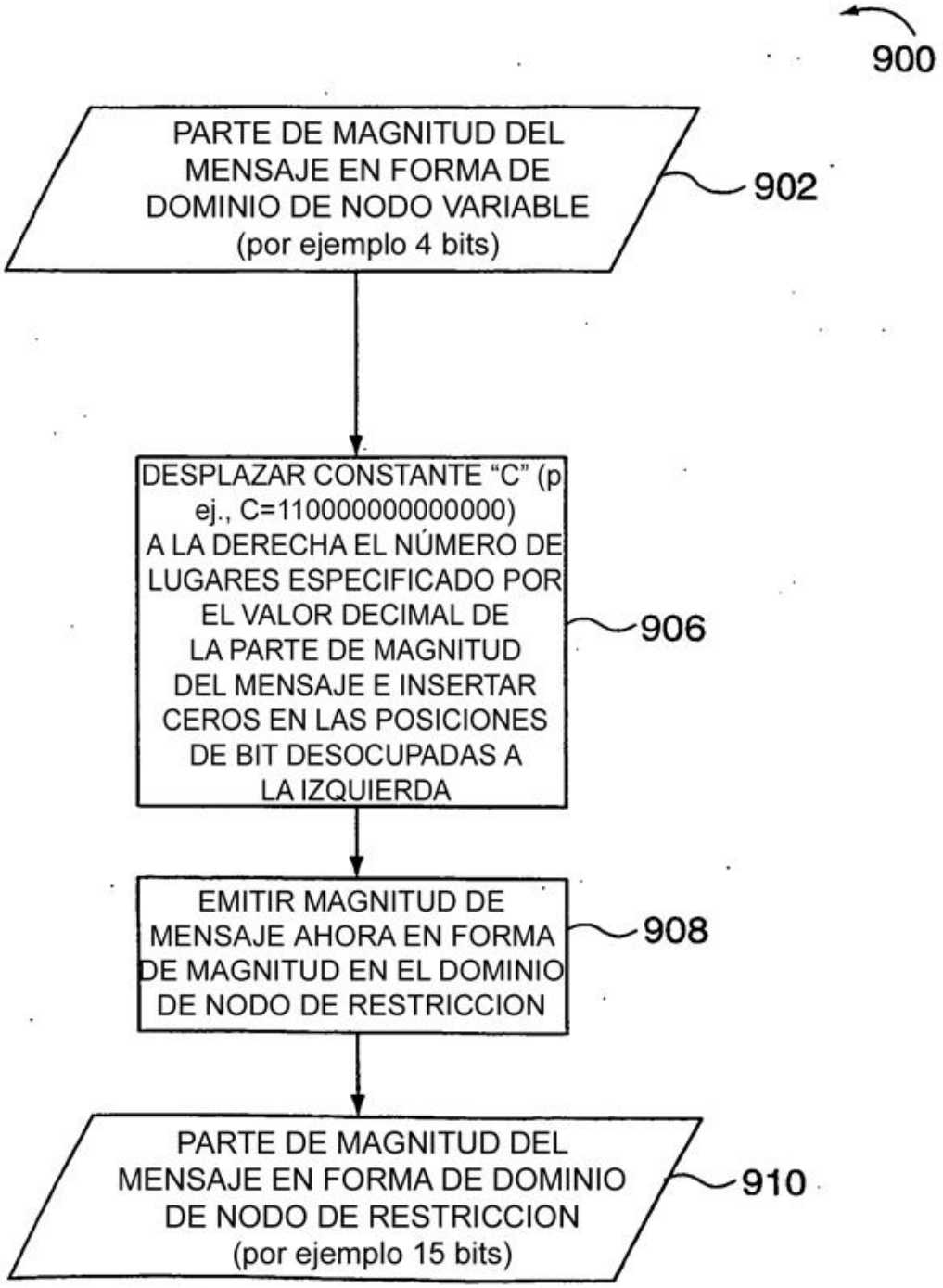
900
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE (por ejemplo 4 bits)
902
DESPLAZAR CONSTANTE "C" (p ej., C=110000000000000) A LA DERECHA EL NÚMERO DE LUGARES ESPECIFICADO POR EL VALOR DECIMAL DE LA PARTE DE MAGNITUD DEL MENSAJE E INSERTAR CEROS EN LAS POSICIONES DE BIT DESOCUPADAS A LA IZQUIERDA
906
EMITIR MAGNITUD DE MENSAJE AHORA EN FORMA DE MAGNITUD EN EL DOMINIO DE NODO DE RESTRICCION
908
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO DE RESTRICCION (por ejemplo 15 bits)
910

FIG. 9

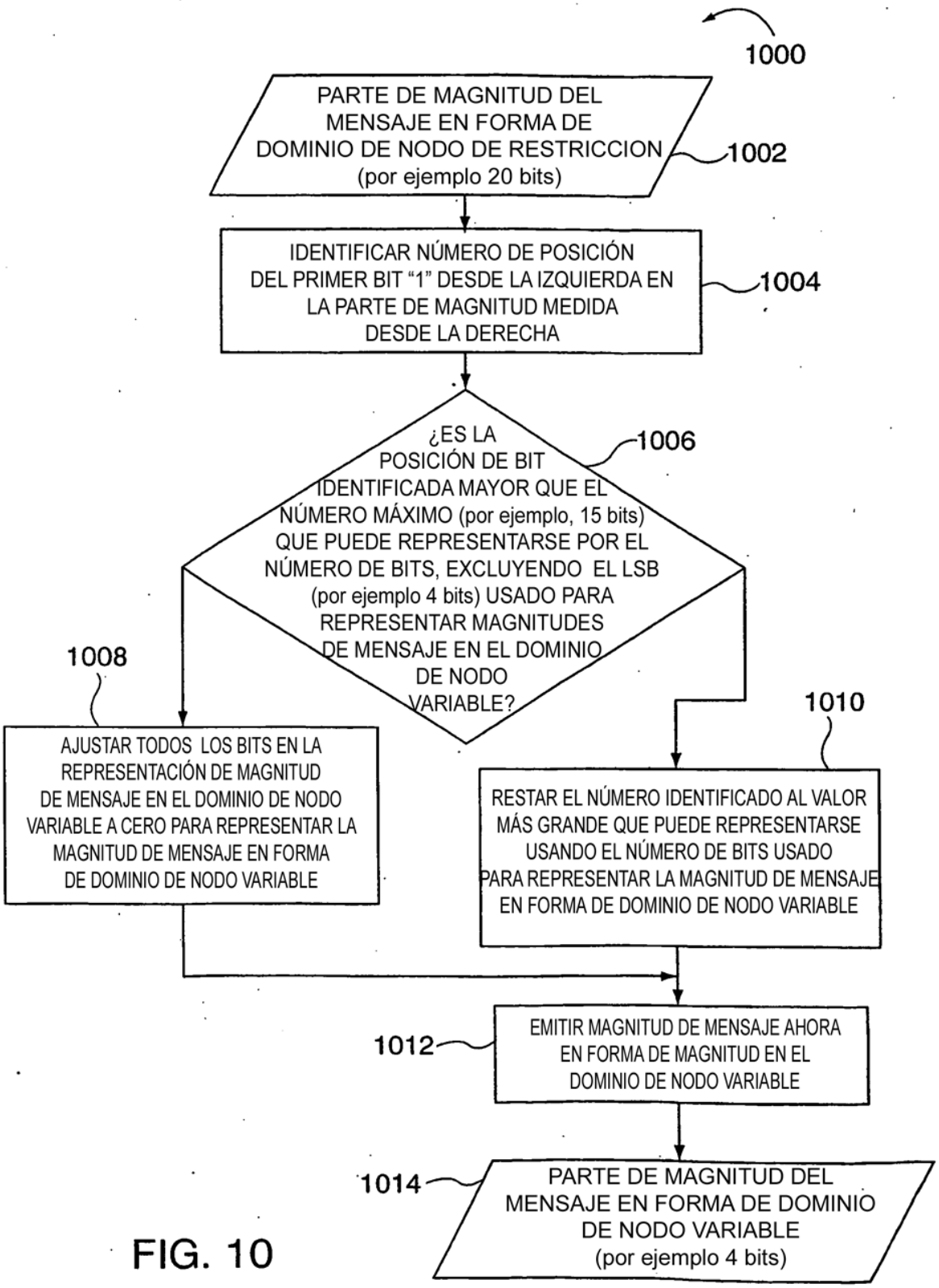
1000
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO DE RESTRICCION (por ejemplo 20 bits)
1002
IDENTIFICAR NÚMERO DE POSICIÓN DEL PRIMER BIT "1" DESDE LA IZQUIERDA EN LA PARTE DE MAGNITUD MEDIDA DESDE LA DERECHA
1004
¿ES LA POSICIÓN DE BIT IDENTIFICADA MAYOR QUE EL NÚMERO MÁXIMO (por ejemplo, 15 bits) QUE PUEDE REPRESENTARSE POR EL NÚMERO DE BITS, EXCLUYENDO EL LSB (por ejemplo 4 bits) USADO PARA REPRESENTAR MAGNITUDES DE MENSAJE EN EL DOMINIO DE NODO VARIABLE?
1006
1008
AJUSTAR TODOS LOS BITS EN LA REPRESENTACIÓN DE MAGNITUD DE MENSAJE EN EL DOMINIO DE NODO VARIABLE A CERO PARA REPRESENTAR LA MAGNITUD DE MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE
1010
RESTAR EL NÚMERO IDENTIFICADO AL VALOR MÁS GRANDE QUE PUEDE REPRESENTARSE USANDO EL NÚMERO DE BITS USADO PARA REPRESENTAR LA MAGNITUD DE MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE
1012
EMITIR MAGNITUD DE MENSAJE AHORA EN FORMA DE MAGNITUD EN EL DOMINIO DE NODO VARIABLE
1014
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE (por ejemplo 4 bits)

FIG. 10

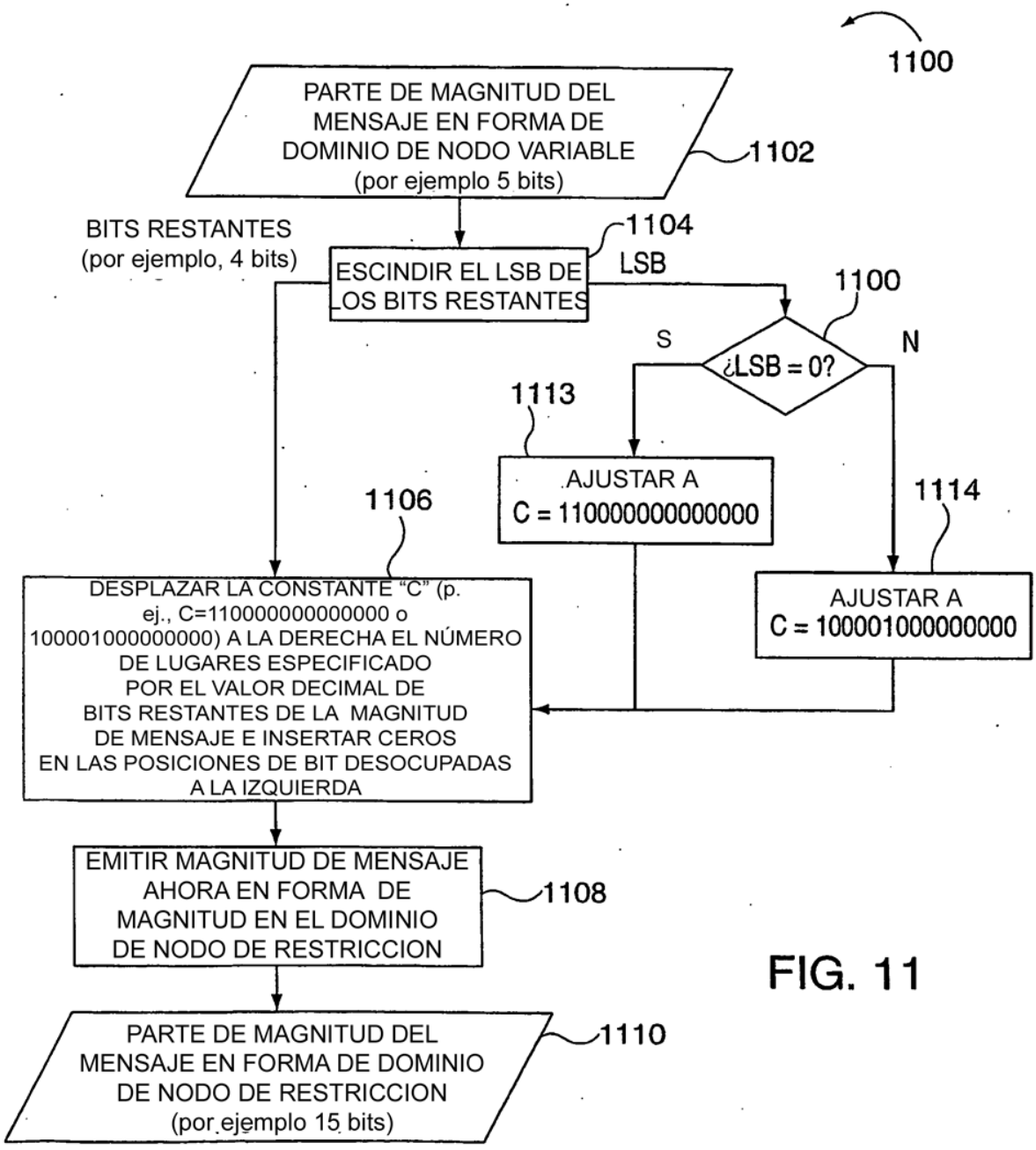
1100
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE (por ejemplo 5 bits)
1102
BITS RESTANTES (por ejemplo, 4 bits)
1104
ESCINDIR EL LSB DE LOS BITS RESTANTES
LSB
1100
S
¿LSB = 0?
N
1113
AJUSTAR A C = 110000000000000
1106
1114
AJUSTAR A C = 100001000000000
DESPLAZAR LA CONSTANTE "C" (p. ej., C=110000000000000 o 100001000000000) A LA DERECHA EL NÚMERO DE LUGARES ESPECIFICADO POR EL VALOR DECIMAL DE BITS RESTANTES DE LA MAGNITUD DE MENSAJE E INSERTAR CEROS EN LAS POSICIONES DE BIT DESOCUPADAS A LA IZQUIERDA
EMITIR MAGNITUD DE MENSAJE AHORA EN FORMA DE MAGNITUD EN EL DOMINIO DE NODO DE RESTRICCION
1108
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO DE RESTRICCION (por ejemplo 15 bits)
1110

FIG. 11

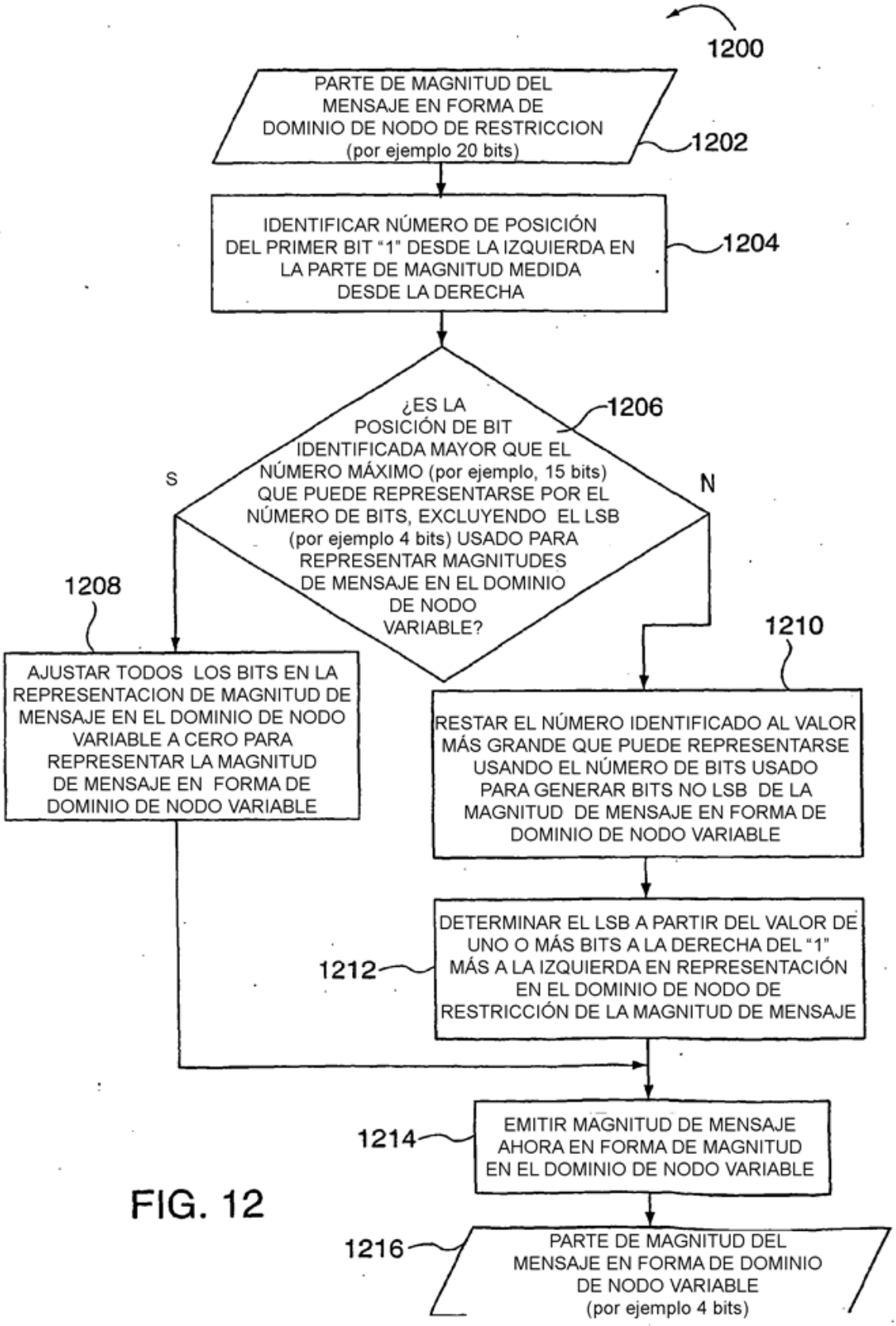
1200
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO DE RESTRICCION (por ejemplo 20 bits)
1202
IDENTIFICAR NÚMERO DE POSICIÓN DEL PRIMER BIT "1" DESDE LA IZQUIERDA EN LA PARTE DE MAGNITUD MEDIDA DESDE LA DERECHA
1204
¿ES LA POSICIÓN DE BIT IDENTIFICADA MAYOR QUE EL NÚMERO MÁXIMO (por ejemplo, 15 bits) QUE PUEDE REPRESENTARSE POR EL NÚMERO DE BITS, EXCLUYENDO EL LSB (por ejemplo 4 bits) USADO PARA REPRESENTAR MAGNITUDES DE MENSAJE EN EL DOMINIO DE NODO VARIABLE?
1206
S
N
1208
AJUSTAR TODOS LOS BITS EN LA REPRESENTACION DE MAGNITUD DE MENSAJE EN EL DOMINIO DE NODO VARIABLE A CERO PARA REPRESENTAR LA MAGNITUD DE MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE
1210
RESTAR EL NÚMERO IDENTIFICADO AL VALOR MÁS GRANDE QUE PUEDE REPRESENTARSE USANDO EL NÚMERO DE BITS USADO PARA GENERAR BITS NO LSB DE LA MAGNITUD DE MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE
1212
DETERMINAR EL LSB A PARTIR DEL VALOR DE UNO O MÁS BITS A LA DERECHA DEL "1" MÁS A LA IZQUIERDA EN REPRESENTACIÓN EN EL DOMINIO DE NODO DE RESTRICCIÓN DE LA MAGNITUD DE MENSAJE
1214
EMITIR MAGNITUD DE MENSAJE AHORA EN FORMA DE MAGNITUD EN EL DOMINIO DE NODO VARIABLE
1216
PARTE DE MAGNITUD DEL MENSAJE EN FORMA DE DOMINIO DE NODO VARIABLE (por ejemplo 4 bits)

FIG. 12

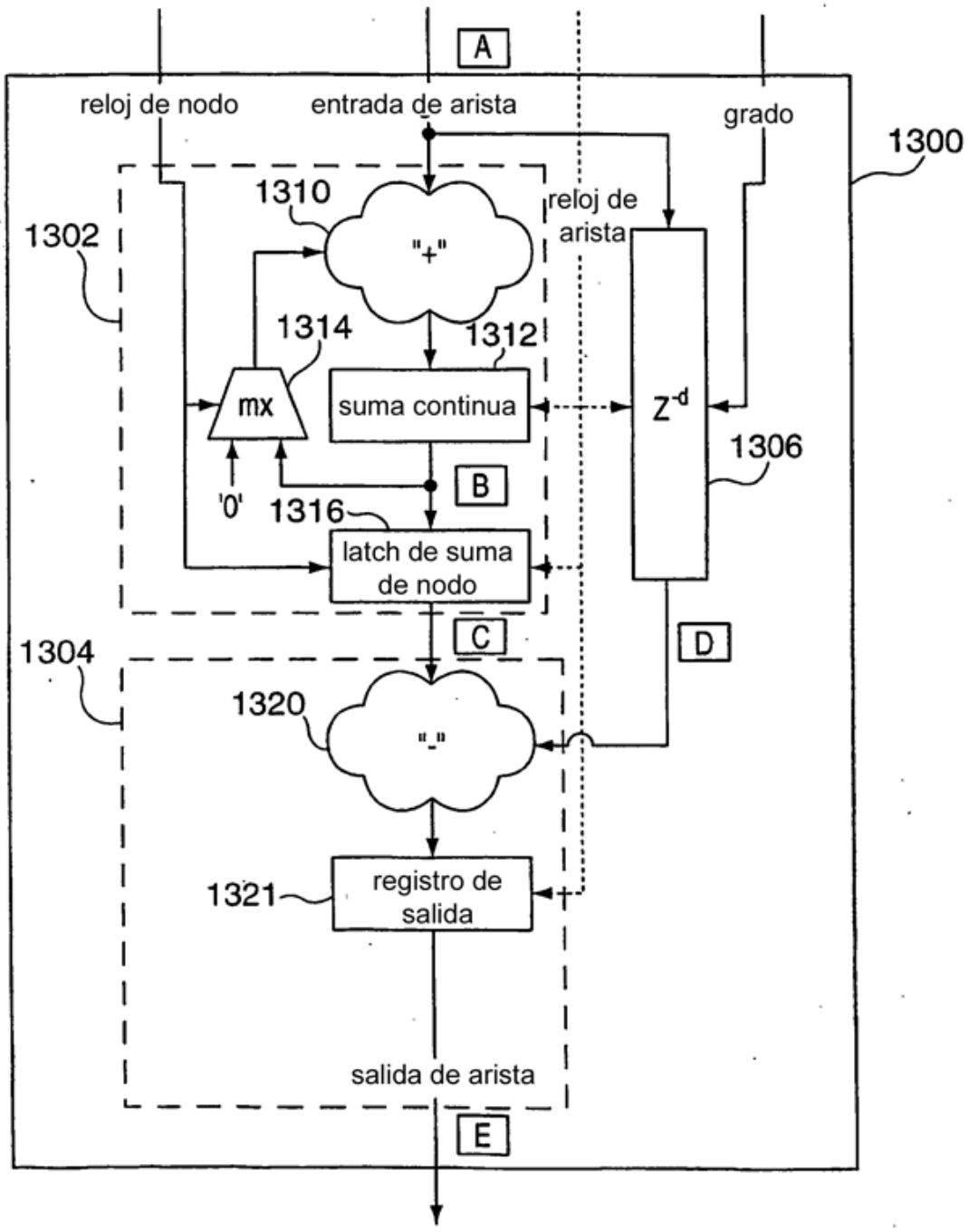
A
reloj de nodo
entrada de arista
grado
1300
1310
reloj de arista
1302
"+"
1314
1312
mx
suma continua
Z^{-d}
1306
'0'
B
1316
latch de suma de nodo
C
D
1304
1320
"-"
1321
registro de salida
salida de arista
E

FIG. 13

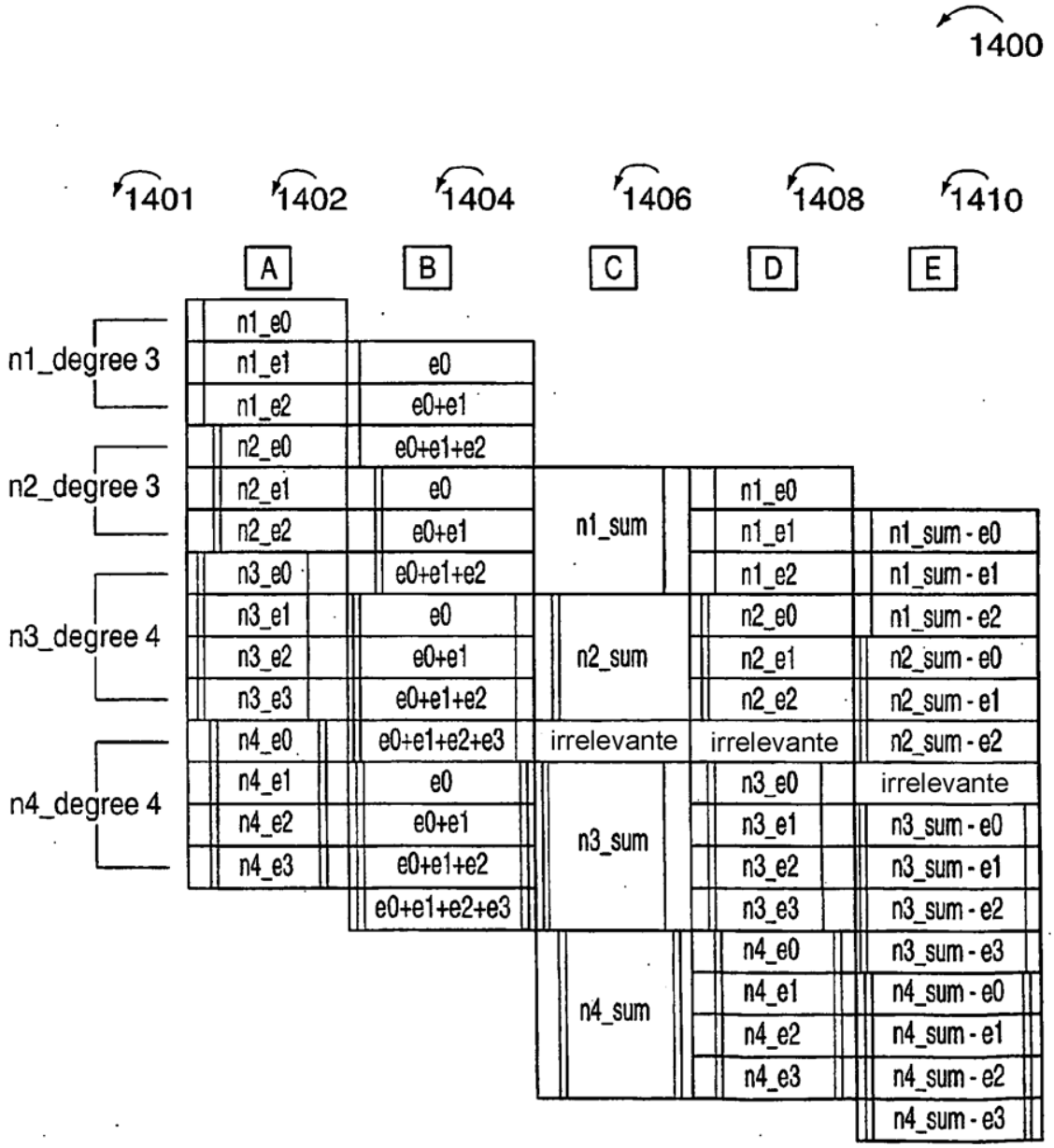
1400
1401
1402
1404
1406
1408
1410
A
B
C
D
E
n1_degree 3
n2_degree 3
n3_degree 4
n4_degree 4
n1_e0
n1_e1
n1_e2
n2_e0
n2_e1
n2_e2
n3_e0
n3_e1
n3_e2
n3_e3
n4_e0
n4_e1
n4_e2
n4_e3
e0
e0+e1
e0+e1+e2
e0
e0+e1
e0+e1+e2
e0
e0+e1
e0+e1+e2
e0+e1+e2+e3
e0
e0+e1
e0+e1+e2
e0+e1+e2+e3
n1_sum
n2_sum
n3_sum
n4_sum
irrelevante
irrelevante
n1_e0
n1_e1
n1_e2
n2_e0
n2_e1
n2_e2
n3_e0
n3_e1
n3_e2
n3_e3
n4_e0
n4_e1
n4_e2
n4_e3
n1_sum - e0
n1_sum - e1
n1_sum - e2
n2_sum - e0
n2_sum - e1
n2_sum - e2
irrelevante
n3_sum - e0
n3_sum - e1
n3_sum - e2
n3_sum - e3
n4_sum - e0
n4_sum - e1
n4_sum - e2
n4_sum - e3

FIG. 14

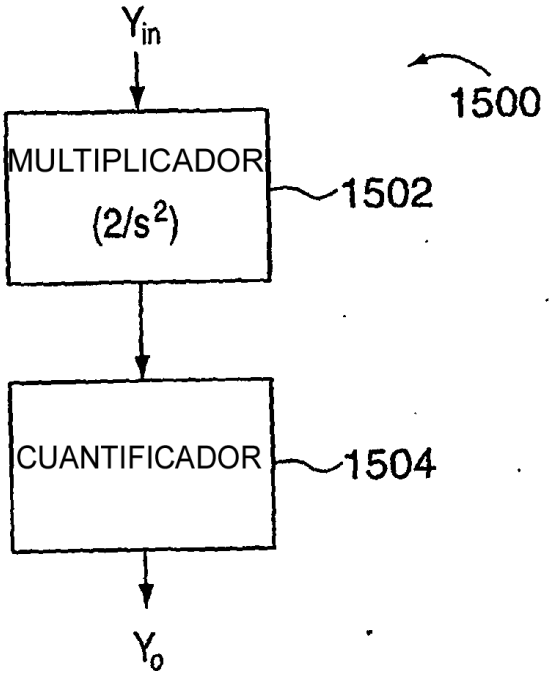

FIG. 15