(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 474 215**

(51) Int. CI.:

H03M 13/29 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **12.12.2002 E 02797323 (9)**

(97) Fecha y número de publicación de la concesión europea: **19.03.2014 EP 1459450**

(54) Título: **Procedimiento y aparato para codificar bits de datos en paralelo**

(30) Prioridad:

14.12.2001 US 20532

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
08.07.2014

(73) Titular/es:

QUALCOMM, INCORPORATED (100.0%)
5775 MOREHOUSE DRIVE
SAN DIEGO, CA 92121, US

(72) Inventor/es:

HURT, JAMES Y.;
HOWARD, MICHAEL A. y
FUCHS, ROBERT J.

(74) Agente/Representante:

FÀBREGA SABATÉ, Xavier

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín europeo de patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre concesión de Patentes Europeas).

DESCRIPCIÓN

Procedimiento y aparato para codificar bits de datos en paralelo

Antecedentes

Campo de la invención

La presente invención se refiere a comunicaciones de datos. Más particularmente, la presente invención se refiere a la codificación de múltiples bits de datos en paralelo (por ejemplo, usando una memoria de múltiples puertos) para reducir de manera significativa los retrasos asociados con la codificación.

Descripción de la técnica relacionada

En un sistema típico de comunicaciones digitales, los datos se procesan, modulan y acondicionan a una unidad transmisora para generar una señal modulada que, a continuación, se transmite a una o más unidades de recepción. El procesamiento de datos puede incluir, por ejemplo, formatear los datos en un formato de trama en particular, codificar los datos formateados con un esquema de codificación particular para proporcionar una detección y/o corrección de errores en las unidades de recepción, canalizar (es decir, cubrir) los datos codificados y difundir los datos canalizados sobre el ancho de banda del sistema. El procesamiento de datos viene definido típicamente por el sistema o estándar que se está aplicando.

En la unidad receptora, se recibe la señal transmitida, acondicionada, demodulada y procesada digitalmente para recuperar los datos transmitidos. El procesamiento en la unidad receptora es complementario al llevado a cabo en la unidad transmisora y puede incluir, por ejemplo, al desensanchar las muestras recibidas, descubrir las muestras desensanchadas, y decodificar los símbolos descubiertos para recuperar los datos transmitidos.

La capacidad de corregir errores de transmisión mejora la fiabilidad de una transmisión de datos. Muchos sistemas de comunicaciones digitales emplean un código convolucional o un código Turbo para proporcionar capacidad de corrección de errores en las unidades receptoras. Los códigos convolucionales operan en los datos en serie, uno o unos pocos bits a la vez. Hay una variedad de códigos convolucionales útiles y una variedad de algoritmos para la decodificación de las secuencias de información codificada recibida para recuperar los datos originales. La codificación Turbo es específicamente un esquema de codificación convolucional concatenada en paralelo. Un código concatenado es una combinación en cascada de dos o más códigos y se utiliza para proporcionar capacidades adicionales de corrección de errores. Para un código concatenado, los bits de código entre las etapas de codificación pueden intercalarse (es decir, reordenarse) para proporcionar diversidad temporal, lo que puede mejorar aún más el rendimiento. Todo paquete o trama de bits de código se almacena normalmente antes de llevar a cabo el reordenamiento. Los bits de código reordenados se recuperan a continuación en serie y son codificados por la siguiente etapa de codificación.

Convencionalmente, la codificación convolucional y Turbo se llevan a cabo en serie en un flujo de bits de entrada. Para cada ciclo de reloj, se proporciona un bit de datos al codificador y se generan dos o más bits de código en función de la tasa de código del codificador. Algunos de los bits de código puede entonces ser perforados (es decir, borrados) para obtener bits de código a otras tasas de código.

Los sistemas de comunicación de acceso múltiple digital normalmente transmiten los datos en paquetes o tramas para posibilitar el intercambio eficiente de los recursos del sistema entre los usuarios activos. Para los servicios que no pueden tolerar largos retrasos (por ejemplo, voz, video), los paquetes se seleccionan para ser de corta duración (por ejemplo, 10 ms) y los códigos de forma tal que se seleccionan para que tenga retrasos en el procesamiento más cortos. Sin embargo, para mejorar la eficacia de la codificación, es deseable procesar y codificar los paquetes de mayor tamaño, lo que puede dar lugar a retrasos de procesamiento más largos utilizando la técnica convencional de codificar datos en serie. Retrasos largos en el procesamiento pueden afectar negativamente al rendimiento del sistema de comunicaciones. Por ejemplo, un tipo de usuario o de datos en particular puede ser seleccionado para una transmisión de datos en particular en base a las condiciones del enlace de comunicaciones. Si los retrasos en el procesamiento son excesivamente largos, las condiciones de enlace pueden haber cambiado en el momento de la transmisión de datos y el rendimiento puede verse comprometido o afectado adversamente.

El documento EP 1 085 660 A1 divulga una implementación paralela de un codificador turbo. Un proceso de sustitución de variables de estado interno se aplica a cada variable de estado interno de la descripción formalizada en general, en donde la representación de las variables de estado internas se analiza en busca de elementos máximos del índice de tiempo que luego son sustituidos a través de la realización de transiciones retroactivas del índice de tiempo utilizando representaciones sustitutas del índice de tiempo determinadas previamente de las variables de estado internas. Los pasos de sustitución se repiten hasta que la representación de cada variable de estado interno sólo depende de los elementos del vector de entrada y variables para las variables de estado internas del código turbo están siendo retardadas de acuerdo con el grado de paralelización.

El documento BENEDETTO, S et al: "A soft-input soft-output máximum a posteriori (MAP) module to decode parallel and serial concatenated codes ", TDA PROGRESS REPORT, volumen 42, número 127, 15 de noviembre de 1996, páginas 1-20, divulga un codificador para códigos concatenados en serie y paralelo.

El documento WO 00/13323 A1 divulga un codificador de códigos turbo que consta de una concatenación en paralelo de dos codificadores convolucionales recursivos constituyentes. Un codificador constituyente emplea una configuración de complementos modulares y registros de desplazamiento acoplados a través de nodos para producir bits de salida.

Resumen de la invención

La invención se define en las reivindicaciones independientes.

A continuación se describen otros aspectos y realizaciones de la invención.

Breve descripción de los dibujos

Las características, naturaleza y ventajas de la presente invención serán más evidentes a partir de la descripción detallada expuesta a continuación cuando se toma en conjunción con los dibujos en los que caracteres de referencia identifican elementos correspondientes y en donde:

La Figura 1 es un diagrama de bloques de un sistema de comunicaciones;

La Figura 2 es un codificador concatenado en serie de acuerdo con una realización;

La Figura 3 es un diagrama de un codificador concatenado que implementa un conjunto particular de matrices polinomio generador;

La Figura 4 es un diagrama de bloques de un codificador convolucional para codificar múltiples bits de datos en paralelo;

Las Figuras 5A y 5B son diagramas esquemáticos de un codificador convolucional que implementa una matriz polinomio generador específica y para la codificación de ocho bits de datos en paralelo;

La Figura 6 es un diagrama esquemático de un codificador convolucional que implementa otra matriz polinomio generador específica y puede codificar cuatro bits de código en paralelo;

La Figura 7A es un diagrama de un intercalador;

Las Figuras 7B y 7C son diagramas de una interfaz entre un codificador convolucional y un intercalador exterior sin y con perforación, respectivamente, de acuerdo con diversas realizaciones;

La Figura 8 es un diagrama de bloques de un codificador de acuerdo con una realización;

La Figura 9 es un diagrama de flujo de un procedimiento para llevar a cabo codificación concatenada de múltiples bits de datos en paralelo de acuerdo con una realización;

La Figura 10 es un sistema de comunicación inalámbrica;

La Figura 11 es un codificador concatenado paralelo;

La Figura 12 es un diagrama funcional de un codificador concatenado en paralelo de acuerdo con una realización;

La Figura 13 es un diagrama funcional de un codificador turbo de acuerdo con una realización;

La Figura 14 es un diagrama funcional de una realización de la invención.

La Figura 15 es un diagrama funcional de circuitos de generación de direcciones para un intercalador para codificadores turbo concatenados en paralelo de acuerdo con una realización;

La Figura 16 es un diagrama funcional de un codificador turbo de acuerdo con una realización; y

La Figura 17 es un diagrama funcional de generación de direcciones para un intercalador de un codificador turbo.

Descripción detallada de varias realizaciones

Codificar múltiples bits en paralelo

La Figura 1 es un diagrama de bloques simplificado de una realización de un sistema de comunicaciones 100 en el que se pueden implementar diversos aspectos de la presente invención. En una unidad transmisora 110, se envía el tráfico de datos, típicamente en paquetes o tramas, desde una fuente de datos 112 a un codificador 114 que formatea y codifica los datos utilizando un esquema de codificación particular. Típicamente el codificador 114 lleva a cabo además intercalación (es decir, reordenamiento) de los bits de código. Un modulador (MOD) 116 recibe, canaliza (es decir, cubre) y ensancha los datos codificados para generar símbolos que se convierten a continuación en una o más señales analógicas. Las señales analógicas son filtradas, (en cuadratura) moduladas, amplificadas y convertidas de forma ascendente por un transmisor (TMTR) 118 para generar una señal modulada, que se transmite entonces a través de una antena 120 a una o más unidades de recepción.

En una unidad receptora 130, la señal transmitida es recibida por una antena 132 y proporcionada a un receptor (RCVR) 134. Dentro del receptor 134, la señal recibida es amplificada, filtrada, convertida bajándola en frecuencia, demodulada en cuadratura y digitalizada para proporcionar muestras. Las muestras son desensanchadas, descubiertas y demoduladas por un demodulador (DEMOD) 136 para generar símbolos demodulados. Un decodificador 138 decodifica entonces los símbolos demodulados y (posiblemente) reordena los datos decodificados para recuperar los datos transmitidos. El procesamiento llevado a cabo por el demodulador 136 y el decodificador 138 es complementario al procesamiento llevado a cabo en la unidad transmisora 110. Los datos recuperados se proporcionan entonces a un receptor de datos 140.

El procesamiento de señales descrito anteriormente da soporte a transmisiones de voz, video, datos por paquetes, mensajería y otros tipos de comunicación en una sola dirección. Un sistema de comunicaciones bidireccional compatible con la transmisión de datos de dos vías. Sin embargo, el procesamiento de la señal para la otra dirección no se muestra en la Figura 1 por simplicidad.

El sistema de comunicaciones 100 puede ser un sistema de Acceso Múltiple por División de Código (CDMA), un sistema de comunicaciones de Acceso Múltiple por División de Tiempo (TDMA) (por ejemplo, un sistema GSM), un sistema de comunicaciones de Acceso Múltiple por División de Frecuencia (FDMA) u otro sistema de comunicaciones de acceso múltiple que de soporte a la comunicación de voz y datos entre usuarios por un enlace terrestre.

El uso de técnicas CDMA en un sistema de comunicaciones de acceso múltiple se divulga en la patente US nº 4.901.307, titulada "SPREAD SPECTRUM MULTIPLE ACCESS SYSTEM USING SATELLITE OR TERRESTRIAL REPEATERS" y en la patente US nº 5.103.459, titulada "SYSTEM AND METHOD FOR GENERATING WAVEFORMS IN A CDMA CELLULAR TELEPHONE SYSTEM". Otro sistema CDMA específico se describe en la solicitud de patente US nº de Serie 08/963.386, titulada "SYSTEM AND METHOD FOR HIGH RATE PACKET DATA TRANSMISSION", presentada el 03 de noviembre 1997 (en lo sucesivo denominada sistema HDR). Estas patentes y solicitud de patente se asignan al cesionario de la presente invención y se incorporan en este documento por referencia.

Los sistemas CDMA están típicamente diseñados para adaptarse a uno o más estándares tales como el TIA/EIA/IS-95-A “Un estándar de Compatibilidad Estación Móvil - Estación Base para un Sistema Celular de Espectro Ensanchado de Banda Ancha de modo dual” (denominado en lo sucesivo el estándar IS-95-A), el “Estándar Mínimo Recomendado TIA/EIA/IS-98 para una estación móvil celular de Espectro Ensanchado de Banda Ancha de modo dual” (en lo sucesivo, el estándar IS-98), el estándar ofrecido por un consorcio denominado " Proyecto de Asociación de 3 ª Generación " (3GPP) y plasmado en un conjunto de documentos que incluyen los documentos número 3G TS 25.211, 3G TS 25.212, 3G TS 25.213, y 3G TS 25.214 (denominado en lo sucesivo como el estándar W - CDMA), y el “TR - 45.5 estándar de Capa Física para Sistemas de Espectro Ensanchando cdma2000 " (en lo sucesivo denominado estándar CDMA - 2000). Se proponen continuamente nuevos estándares CDMA y se adoptan para su uso. Estos estándares CDMA se incorporan aquí por referencia.

La Figura 2 es un diagrama de bloques de un codificador 200 que puede ser diseñado para poner en práctica algunas realizaciones de la presente invención. El codificador 200 puede ser utilizado por el codificador 114 en la Figura 1. En esta forma de realización, el codificador 200 implementa un código concatenado e incluye un codificador convolucional exterior 212, un intercalador 214 y un codificador convolucional interior 216 acoplados en cascada. El codificador convolucional exterior 212 recibe y codifica de forma convolucional los datos de entrada para generar bits de código, que se proporcionan al intercalador 214 para su almacenamiento. Una vez que un paquete entero de bits de código se ha almacenado en el intercalador 214, los bits de código se recuperan y se proporcionan al codificador convolucional interior 216. Para lograr el intercalado, los bits de código se leen en un orden que es diferente del orden en el que los bits se escriben en el intercalador 214. El codificador convolucional exterior 212 recibe y codifica de forma convolucional los bits de código para generar datos codificados, que se proporcionan a continuación a la etapa de procesamiento posterior.

Un codificador convolucional convencional recibe y codifica datos en serie, un bit cada vez (es decir, por cada ciclo de reloj). Para los sistemas de comunicación que transmiten datos en paquetes grandes, la codificación de datos en

serie puede resultar en un retraso de procesamiento largo. Por otra parte, para un codificador concatenado compuesto de múltiples codificadores convolucionales acoplados en cascada, los retrasos de procesamiento pueden ser excesivamente largos, especialmente si los codificadores convolucionales exterior e interior ambos codifican bits en serie.

En un aspecto, un codificador convolucional es capaz de recibir múltiples (M) bits de codificación en paralelo. Esta capacidad permite al codificador convolucional codificar un paquete de datos en aproximadamente (1/M)-ésimo de la cantidad de tiempo requerida por un codificador convolucional convencional. Los beneficios son más pronunciados para un codificador concatenado (por ejemplo, un codificador Turbo) cuando cada uno de los codificadores convolucionales distintos procesa bits en paralelo.

De acuerdo con otro aspecto, un intercalador es capaz de almacenar y proporcionar múltiples bits de datos en paralelo. El intercalador puede implementarse usando, por ejemplo, una memoria multi-puerto. Cuando se utiliza en combinación con los codificadores convolucionales descritos en el presente documento, el intercalador puede reducir aún más los retrasos de procesamiento ya que los datos se pueden escribir en y leer desde el dispositivo de intercalado en una fracción del tiempo.

Para un mayor claridad, un ejemplo se describe ahora a modo de ejemplo para un codificador utilizado para una transmisión de datos de enlace descendente en el sistema de comunicaciones descrito en la antes mencionada solicitud de patente US nº de serie 08/963.386 (es decir, el sistema HDR). El sistema HDR emplea un código concatenado compuesto de un código convolucional exterior, intercalado y un código convolucional interior. El sistema HDR también define dos formatos de paquetes que tienen las propiedades que figuran en la Tabla 1.

Tabla 1

Parámetros	Formato de paquete 1	Formato de paquete 2	Unidades
Bits/paquete total	1024	2048	bits
Codificador convolucional exterior			
Bits/paquete datos entrada	1018	2042	bits
Bits/paquete cola de código	4	4	bits
Tasa de código exterior	1/2	2/3	
Patrón de perforación de código exterior	(1111)	(1011)	
Bits/paquete código salida	2044	3069	bits
Profundidad de intercalador	2048	3072	bits
Codificador convolucional interior			
Bits/paquete código entrada	2044	3069	bits
Bits/paquete cola de código	4	3	bits
Tasa de código interior	1/2	3/4	
Patrón de perforación de código interior	(111111)	(111001)	
Bits/paquete código salida	4096	4096	bits
Tasa de código global	1/4	1/2	

En el sistema HDR, el codificador convolucional exterior implementa un código convolucional con tasa 1/2 definido por la siguiente matriz polinomio generador:

$$G_o(x) = \left[1, \frac{x^4 + x^2 + x + 1}{x^4 + x^3 + 1}\right].$$ Ecuación (1)

El codificador convolucional interior en el sistema HDR implementa una tasa de 1/2 código convolucional definido por la siguiente matriz polinomio generador:

$$G_i(x) = \left[1, \frac{x^2 + x + 1}{x + 1}\right].$$ Ecuación (2)

La Figura 3 es un diagrama de un codificador 300 que implementa los códigos convolucionales exterior e interior definidos por las ecuaciones (1) y (2). Los bits de datos u se proporcionan a un codificador convolucional exterior 310 que implementa la ecuación (1) y genera dos salidas y_{oa} y y_{ob}. Dentro del codificador 310, los bits de datos u se proporcionan a un sumador 312 que se empareja en cascada con los registros 314a hasta 314d (que se utilizan para implementar un conjunto de retardos). Las salidas del sumador 312 y los registros 314A, 314B, y 314D son sumadas por los sumadores 316A, 316B, y 316C para implementar el numerador del segundo elemento en la matriz polinomio generador expresado en la ecuación (1). Las salidas de los registros 314C y 314D son sumadas por un sumador 318 y se proporcionan al sumador 312 para poner en práctica el denominador del segundo elemento en la ecuación (1). Los u bits de datos de entrada se proporcionan como la primera salida y_{oa} y la salida del sumador 316c comprende la segunda salida y_{ob}.

Los bits de código en las salidas y_{oa} y y_{ob} de codificador convolucional exterior 310 se puede perforar (no se muestra en la Figura 3 por simplicidad). Los bits de código sin perforar, se proporcionan al intercalador 330 y se reordenan. Los bits de código reordenados se proporcionan, a continuación, a un codificador convolucional interior 340 que implementa la ecuación (2) y genera dos salidas y_{ia} y y_{ib}. Dentro del codificador 340, los v bits de código se proporcionan a un sumador 342 acopla en cascada con los registros 344A y 344B. Las salidas del sumador 342 y los registros 344A y 344B son sumadas por los sumadores 346A y 346B para implementar el numerador del segundo elemento en la matriz polinomio generador expresada en la ecuación (2). La salida del registro 344A se proporciona al sumador 342 para poner en práctica el denominador del segundo elemento en la ecuación (2). Los v bits de código de entrada se proporcionan como la primera salida y_{ia} y la salida del sumador 346B comprende la segunda salida y_{ib}.

Convencionalmente, los bits de datos en serie u se proporcionan al codificador 310 y los v bits de código también se proporcionan en serie al codificador 340. Para cada bit de datos de entrada, el codificador convolucional exterior 310 genera dos bits de código. El intercalador 330 recibe y almacena los bits de código y proporciona los bits de código en serie al codificador convolucional interior 340. La codificación de los bits en serie da lugar a retrasos largos en el procesamiento.

El codificador convolucional de una forma de realización es capaz de codificar múltiples bits en paralelo para acortar significativamente los retardos de codificación. Para cada ciclo de reloj, múltiples bits de datos (por ejemplo, M) pueden ser recibidos y codificados para generar múltiples bits de código. Para una tasa de codificación media, se generan 2M bits de código para los M bits de datos. M puede seleccionarse para que sea cualquier número tal como, por ejemplo, 4, 8, 16, 32 y así sucesivamente. Varias realizaciones alternativas de un tales codificadores convolucionales se describen a continuación.

Muchos sistemas de comunicaciones digitales, tales como el sistema HDR, transmiten datos en paquetes. El número de bits en un paquete (es decir, el tamaño del paquete) se selecciona basándose en un número de criterios tales como, por ejemplo, la tasa de datos, la cantidad de datos a transmitir, los requisitos de retraso de procesamiento y demás. Para permitir que el decodificador en el receptor inicie en un estado conocido al comienzo de cada paquete, lo que acorta el tiempo de decodificación y mejora el rendimiento, el codificador se inicializa en un estado conocido (por ejemplo, todos ceros) al inicio de cada paquete. La inicialización se consigue mediante la inserción de un conjunto de bits de cola de código en el final del paquete precedente. Los bits de cola de código se seleccionan de tal manera que el codificador se establece en el estado conocido.

El codificador convolucional del ejemplo a modo de ejemplo se implementa con una tabla de consulta. Haciendo referencia a la Figura 3, el codificador convolucional exterior 310 puede ser visto como una máquina de estados con un estado de 4 bits definido por las salidas de los registros 314A hasta 314D. Para generar el contenido de la tabla de consulta, los M bits de datos de entrada en índice temporal n pueden representarse mediante un vector U_n, Los bits de código 2M pueden ser representados por un vector Y_n, y el estado del codificador de corriente puede representarse mediante un vector X_n. El siguiente estado X_{n+1} para el codificador y el vector de salida del codificador Y_n se puede expresar como:

Datos	Cola de código	
$X_{n+1} = f(X_n, U_n)$	$X_{n+1} = 0$	Ecuación (3)
$Y_n = g_1(X_n, U_n)$	$Y_n = g_2(X_n, U_n)$	Ecuación (4)

Cada una de las ecuaciones (3) y (4) establece una ecuación para usar cuando la entrada es de datos y otra ecuación para usar cuando la entrada del codificador incluye bits de cola de código.

Las ecuaciones (3) y (4) pueden calcularse para todas las posibles combinaciones de bits de datos de entrada y estados del codificador. Por ejemplo, para la ecuación (4), los bits de código de salida pueden calcularse para el vector de entrada U_n = 0...00, y un estado del codificador X_n = 0...00, un vector de entrada U_n = 0...01 y el estado del codificador de X_n = 0...00, y así sucesivamente, y un vector de entrada U_n = 1...11 y el estado del codificador X_n = 0...00. Los bits de código de salida se pueden calcular para todas las combinaciones posibles del vector de entrada U_n y un estado de codificador X_n = 0...01. El proceso entonces continúa hasta que se calculan todas las combinaciones de vector de entrada y estado del codificador. La ecuación (3) también puede calcularse de manera similar.

Los resultados de los cálculos de las ecuaciones (3) y (4) se pueden almacenar en una memoria que implementa una tabla de búsqueda. El tamaño de memoria requerido depende del número de bits de datos a codificar en paralelo y la matriz polinomio generador particular que se está implementado. Por ejemplo, si se van a codificar ocho bits de datos en paralelo con el código convolucional expresado en la ecuación (1), se puede utilizar una memoria que tiene un tamaño de 12 bits de dirección y 20 bits de datos (es decir, 4,096 x 20). La dirección de 12 bits se compone de 8 bits de datos de entrada y 4 bits para el estado actual del codificador. La salida de 20 bits incluye 16 bits de código y 4 bits para el estado siguiente del codificador.

Una vez que la memoria se ha definido adecuadamente, el vector de datos de entrada U_n y el estado del codificador actual X_n pueden ser proporcionados a la entrada de dirección de la memoria, que luego proporciona el vector de salida Y_n y el siguiente estado del codificador X_{n+1}. El siguiente estado del codificador X_{n+1} se almacena apropiadamente para su uso con el siguiente vector de datos de entrada U_{n+1}.

En otro ejemplo, el codificador convolucional se lleva a cabo con una máquina de estados. El estado y la salida del codificador se pueden expresar como se muestra en las ecuaciones (3) y (4). Cada una de las ecuaciones (3) y (4) se puede resolver de forma recursiva, y las ecuaciones resultantes se implementan en hardware, software o una combinación de los mismos. Las ecuaciones recursivas para el codificador pueden resolver de la siguiente manera. Sea $X_n^T = [x_4 \quad x_3 \quad x_2 \quad x_1]$ el vector de estado transpuesto y u_0 el bit de datos de entrada en el índice de tiempo 0. El siguiente estado y salida del codificador pueden ser expresados como:

$$X_1 = AX_0 + Bu_0 \,, \qquad \text{Ecuación (5)}$$

$$y_0 = CX_0 + Du_0 \,. \qquad \text{Ecuación (6)}$$

donde A, B, C, y D son escalares, vectores, y la matriz que dependen de la matriz polinomio generador particular que se está implementando. La ecuación de estado del codificador (5) puede resolverse de forma recursiva como sigue:

$$X_2 = A^2X_0 + ABu_0 + Bu_1$$
$$X_3 = A^3X_0 + A^2Bu_0 + ABu_1 + Bu_2$$
$$\vdots$$
$$X_8 = A^8X_0 + A^7Bu_0 + A^6Bu_1 + A^5Bu_2 + A^4Bu_3 + A^3Bu_4 + A^2Bu_5 + ABu_6 + Bu_7$$

La ecuación de salida del codificador (6) también puede resolverse de forma recursiva de manera similar.

Las ecuaciones (5) y (6) se utilizan para codificar un bit de datos u cada vez. Un conjunto similar de ecuaciones se puede derivar para codificar M bits de datos en paralelo. Por ejemplo, para codificar 8 bits de datos en paralelo (es decir, M = 8), el transpuesto del vector de datos de entrada en el índice de tiempo n se puede definir como

$$U_n^T = [u_{n7} \quad u_{n6} \quad u_{n5} \quad u_{n4} \quad u_{n3} \quad u_{n2} \quad u_{n1} \quad u_{n0}]$$

y la transpuesta del vector de código de salida se puede definir como

$$Y_n^T = [y_{n7} \quad y_{n6} \quad y_{n5} \quad y_{n4} \quad y_{n3} \quad y_{n2} \quad y_{n1} \quad y_{n0}]$$

Usando las notaciones vectoriales definidas U_n e Y_n, las ecuaciones (5) y (6) se pueden expresar como:

$$X_{n+1} = FX_n + GU_n \,, \qquad \text{Ecuación (7)}$$

$$Y_n = HX_n + IU_n \,. \qquad \text{Ecuación (8)}$$

donde F, G, H e I son vectores y matrices que son dependientes de la matriz polinomio generador particular implementada, el actual estado del codificador X_n y el vector de datos de entrada U_n. La ecuación (7) se utiliza para generar el siguiente estado codificador X_{n+1} después de que los M bits de datos se hayan codificado, y la ecuación (8) se utiliza para generar las salidas del codificador Y_n para el vector de entrada U_n.

Para determinar F, G, H e I en las ecuaciones (7) y (8), las ecuaciones (5) y (6) se pueden resolver de forma recursiva utilizando diversas técnicas y los resultados de los cálculos recursivos se pueden utilizar para poner en práctica las ecuaciones (7) y (8). Por ejemplo, una tabla puede usarse para tabular el estado y las salidas del codificador para cada bit de datos de entrada. Las entradas en la tabla a continuación, pueden ser utilizadas para implementar las ecuaciones (7) y (8), tal como se describe a continuación.

La Tabla 2 muestra los estados del codificador y salidas después de ocho bits de datos de entrada u_0 hasta u_7 se han proporcionado en serie al codificador convolucional 310 en la Figura 3, que implementa la ecuación (1). Como se muestra en la Figura 3, los registros 314A hasta 314D almacenan inicialmente los valores x_1, x_2, x_3 y x_4, respectivamente. En el primer ciclo de reloj, se proporciona el primer bit de datos u_0 al codificador 310, y la salida del sumador 312 se calcula como $x_4 + x_3 + u_0$, que se almacena en la segunda fila, segunda columna de la Tabla 2. Las salidas del codificador se calculan como $y_{a0} = u_0$ y $y_{b0} = (x_3 + x_4 + u_0) + x_4 + x_2 + x_1 = x_3 + x_2 + x_1 + u_0$. (Cada sumador 316 lleva a cabo adición en módulo 2.) En el siguiente ciclo de reloj, los valores del sumador 312 y los registros 314A hasta 314C se desplazan en los registros 314A hasta 314D, respectivamente. El siguiente bit de datos u_1 se proporciona al codificador, y la salida del sumador 312 se calcula como $u_0 + x_3 + x_2$, que se almacena en la tercera fila, segunda columna de la Tabla 2. Las salidas del codificador se calculan como $y_{a1} = u_1$ y $y_{b2} = (x_3 + x_2 + x_1) + x_3 + x_1 + (x_4 + x_3 + u_0) = x_4 + x_3 + x_2 + x_1 + u_0 + u_1$. El proceso continúa hasta que el octavo bit de datos u_7 es recibido y procesado.

El vector de salida del codificador $Y_b = [y_{b7}\, y_{b6}\, y_{b5}\, y_{b4}\, y_{b3}\, y_{b2}\, y_{b1}\, y_{b0}]$ corresponde al vector de entrada U = [$u_7\, u_6\, u_5\, u_4\, u_3\, u_2\, u_1\, u_0$] y se genera en base a las entradas de la última columna de la Tabla 2. El estado del codificador X_{n+1} después de que el octavo bit de datos u_7 haya sido codificado se genera en base a las entradas de la última fila de la Tabla 2. Como se muestra en la Tabla 2, el vector de salida del codificador Y_b y el siguiente estado del codificador X_{n+1} son cada uno una función del estado actual del codificador $X_n = [x_4\, x_3\, x_2\, x_1$ y el vector de entrada U. Para la fase de datos, la salida del codificador Y_a vector es simplemente una función del vector de entrada U (es decir, Y_a = U).

Tabla 2

u	1	x_1	x_2	x_3	x_4	y_a	y_b
u_0	$x_4 + x_3 + u_0$	x_1	x_2	x_3	x_4	u_0	$x_3 + x_2 + x_1 + u_0$
u_1	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	x_1	x_2	x_3	u_1	$x_4 + x_3 + x_2 + x_1 + u_0 + u_1$
u_2	$x_2 + x_1 + u_2$	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	x_1	x_2	u_2	$x_4 + u_2 + x_1 + u_0 + u_1 + u_2$
u_3	$x_4 + x_3 + x_1 + u_0 + u_3$	$x_4 + x_3 + x_1 + u_0 + u_3$	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	x_1	u_3	$x_4 + x_1 + u_0 + u_1 + u_2 + u_3$
u_4	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_3 + x_1 + u_0 + u_4$	$x_3 + x_2 + x_1$	$x_4 + x_3 + u_0$	u_4	$x_4 + u_0 + u_1 + u_2 + u_3 + u_4$
u_5	$x_4 + x_2 + u_1 + u_2 + u_5$	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_3 + x_1 + u_0 + u_3$	$x_3 + x_2 + x_1$	u_5	$x_3 + u_1 + u_2 + u_3 + u_4 + u_5$
u_6	$x_4 + x_3 + x_2 + u_0 + u_2 + u_3 + u_6$	$x_3 + x_1 + u_1 + u_2 + u_5$	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_3 + u_1 + u_0 + u_3$	u_6	$x_2 + u_2 + u_3 + u_4 + u_6 + u_6$
u_7	$x_3 + x_2 + x_1 + u_1 + u_3 + u_4 + u_7$	$x_4 + x_3 + x_2 + u_0 + u_2 + u_3 + u_6$	$x_4 + x_2 + u_1 + u_2 + u_5$	$x_4 + x_2 + u_0 + u_1 + u_4$	$x_4 + x_2 + u_0 + u_1 + u_4$	u_7	$x_1 + u_3 + u_4 + u_5 + u_6 + u_7$
		$x_3 + x_2 + x_1 + u_1 + u_3 + u_4 + u_7$	$x_4 + x_3 + x_2 + u_0 + u_2 + u_3 + u_6$	$x_3 + x_1 + u_1 + u_2 + u_5$	$x_4 + x_2 + u_0 + u_1 + u_4$		

Haciendo referencia de nuevo a la Tabla 1, el codificador convolucional exterior en el sistema HDR 1018 recibe bits de datos y cuatro bits de cola de código para cada paquete en el formato de paquete 1. Si se codifican ocho bits en paralelo, 128 se utilizan ciclos de reloj para codificar un paquete de datos. Los primeros 127 ciclos de reloj se utilizan para un código de 1016 bits de datos (es decir, 127 x 8 = 1016), y el ciclo de reloj 128a se utiliza para codificar los dos bits de datos restantes y cuatro bits de cola de código. Los primeros 127 ciclos de reloj se les conoce como "fase de datos" y el último ciclo de reloj se denomina "fase de cola de código".

El codificador convolucional exterior recibe 2042 bits de datos y cuatro bits de cola de código para cada paquete en formato de paquete 2. Si se codifican en paralelo ocho bits, se utilizan 256 ciclos de reloj para codificar un paquete de datos. Los primeros 255 ciclos de reloj se utilizan para un código de 2040 bits de datos (es decir, 255 x 8 = 2040), y el ciclo de reloj 256º se utiliza para codificar los dos bits de datos restantes y cuatro bits de cola de código. Los primeros 255 ciclos de reloj se denominan fase de datos y el último ciclo de reloj se denomina fase de cola de código.

La Tabla 3 muestra los estados del codificador y salidas después de que dos bits de datos u_0 y u_1 y cuatro bits de cola de código se hayan proporcionado serie al codificador convolucional 310 en la Figura 3. Una vez más, a través de los registros 314A hasta 314D almacenan inicialmente los valores x_1, x_2, x_3 y x_4, respectivamente. En los dos primeros ciclos de reloj, los dos bits de datos, u_0 y u_1, se proporcionan en serie al codificador. Los estados del codificador x_1 *a* x_4, y las salidas del codificador y_c a y_m se calculan de manera similar a la descrita anteriormente. Por lo tanto, las segunda y tercera filas de la Tabla 3 son idénticas a las segunda y tercera filas de la Tabla 2. En el tercer ciclo de reloj, se proporciona el primer bit de cola de código que tiene un valor $x_2 + x_1$ al codificador. El valor del bit de cola de código se selecciona de tal manera que la salida del sumador 312 es igual a cero, lo que se utiliza para eliminar el codificador convolucional. Las salidas del codificador se calculan como $y_{c2} = x_2 + x_2$ y $y_{d2} = x_4 + u_0 + u_1$. En el siguiente ciclo de reloj, los valores del sumador 312 y los registros 314A hasta 314C se desplazan en los registros 314A hasta 314D, respectivamente. El segundo bit de cola de código se selecciona para que sea $x_4 + x_3 + x_1 + u_o$, de nuevo para fijar la salida del sumador 312 a cero y reiniciar el codificador. El proceso continúa, con los dos últimos bits proporcionados al codificador que tiene valores de cero.

Tal y como se muestra en la Tabla 3, las salidas del codificador Y_c y Y_d son ambas funciones del vector de entrada U y el estado actual del codificador X_n. Para la fase de cola de código, el siguiente estado del codificador X_{n+1} se establece en un estado conocido de todos ceros (es decir, X_8 = [0 0 0 0].

Tabla 3

u	1	x_1	x_2	x_3	x_4	y_c	y_d
u_0	$x_4 + x_3 + u_0$	x_1	x_2	x_3	x_4	u_0	$x_3 + x_2 + x_1 + u_0$
u_1	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	x_1	x_2	x_3	u_1	$x_4 + x_3 + x_2 + x_1 + u_0 + u_1$
$x_2 + x_1$	0	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	x_1	x_2	$x_2 + x_1$	$x_4 + u_0 + u_1$
$x_4 + x_3 + x_1 + u_0$	0	0	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	x_1	$x_4 + x_3 + x_1 + u_0$	$x_3 + x_2 + x_1 + u_1$
$x_4 + x_2 + u_0 + u_1$	0	0	0	$x_3 + x_2 + u_1$	$x_4 + x_3 + u_0$	$x_4 + x_2 + u_0 + u_1$	$x_4 + x_3 + u_0$
$x_3 + x_2 + u_1$	0	0	0	0	$x_3 + x_2 + u_1$	$x_3 + x_2 + u_1$	$x_3 + x_2 + x_1$
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
		0	0	0	0		

La Figura 4 es un diagrama de bloques de un codificador convolucional 400 que puede codificar múltiples bits de datos de entrada en paralelo. El codificador convolucional 400 se puede utilizar para implementar las fases de datos y el código de cola (por ejemplo, tal como se definen en las Tablas 2 y 3, respectivamente). La arquitectura de codificador mostrada en la Figura 4 se puede utilizar para poner en práctica, por ejemplo, el codificador convolucional exterior 310 o el codificador convolucional interior 340 de la Figura 3.

Dentro del codificador convolucional 400, los bits de datos de entrada se proporcionan en paralelo como un vector de datos U a una máquina de estado de codificador 410, un generador de salida de fase de datos 420 y un generador de salida de fase de cola de código 430. La máquina de estados del codificador 410 también recibe el estado actual del codificador X y determina el nuevo estado del codificador en base al vector recibido de entradas U y el estado actual de la máquina de estados del codificador X. El codificador 410 puede implementar, por ejemplo, la última fila en la Tabla 2.

El generador de salida de fase de datos 420 y el generador de salida de fase de cola de código 430 también reciben el actual estado del codificador *X* y determinan las salidas del codificador para la fase de datos y la fase de cola de código, respectivamente, sobre la base de las aportaciones recibidas *X* y *U*. El generador de salida de fase de datos 420 puede implementar, por ejemplo, las dos últimas columnas de la Tabla 2, y el generador de salida de cola de código 430 puede implementar, por ejemplo, las dos últimas columnas de la Tabla 3. La primera y segunda salidas, Y_n e Y_b, desde el generador de salida de datos de la fase 420 se proporcionan a los multiplexores (MUX) 440A y 440B, respectivamente. Del mismo modo, las primera y segunda salidas, Y_c e Y_d de generador de salida de fase de cola de código 430 se proporcionan a los multiplexores 440A y 440B, respectivamente. Los multiplexores 440A y 440B ofrecen las salidas Y_a e Y_b, respectivamente, desde el generador de salida de fase de datos 420 cuando se opera en la fase de datos y las salidas Y_c e Y_d, respectivamente, desde el generador de salida de fase de cola de código 430 cuando se opera en la fase de cola de código.

Para implementar un codificador convolucional que codifica de forma continua bits de datos de entrada tal y como se reciben, sin tener que restablecer el estado del codificador en el inicio de cada paquete, solo se necesitan la máquina de estados del codificador 410 y el generador de salida de fase de datos 420. Para los sistemas de comunicaciones (por ejemplo, el sistema HDR) en el que los datos se envían en paquetes y los bits de cola de código se utilizan para restablecer el codificador convolucional a un estado conocido en el inicio de cada paquete, el generador de salida de fase de cola de código 430 y los multiplexores 440 se utilizan para proporcionar las salidas requeridas del codificador.

El diseño de la máquina de estados del codificador 410 y del generador de salida de fase de datos 420 depende de la matriz polinomio generador particular que se aplica y el número de bits de datos a codificar en paralelo. El diseño del generador de salida de fase de cola de código 430 es dependiente de la matriz polinomio generador, el número de bits de datos a codificar en paralelo y el formato de trama particular (es decir, el número de datos y bits de cola de código a codificar en la fase de cola de código). Un diseño específico del codificador convolucional 400 se describe ahora a continuación.

La Figura 5A es un diagrama esquemático de un codificador convolucional 500 que puede codificar ocho bits de datos de entrada en paralelo y aplica la matriz polinomio generador expresada en la ecuación (1). El codificador convolucional 500 incluye una máquina de estados del codificador 510 que implementa la máquina de estados se define en la Tabla 2 y un generador de salida de fase de datos 520 que genera las salidas del codificador definidas en la Tabla 2. La máquina de estados del codificador 510 y el generador de salida de fase de datos 520 corresponden a la máquina de estados del codificador 410 y al generador de salida de fase de datos 420, respectivamente, en la Figura 4. En este ejemplo, la máquina de estados del codificador 510 se implementa con

puertas AND 512A hasta 512D y registros 514A hasta 514D, y el generador de salida de fase de datos 520 se implementa con puertas AND 522A hasta 522H.

Tal y como se muestra en la Figura 5A, los ocho bits de datos de entrada, u_0 hasta u_7, se proporcionan en paralelo a las entradas a la máquina de estados del codificador 510 y el generador de salida de datos de la fase 520, cada uno de los cuales recibe también el estado actual del codificador definido por x_1 a x_4. Cada puerta AND 512 dentro de la máquina de estados del codificador 510 acopla selectivamente las entradas u_0 - u_7 y x_1 - x_4, según la definición de la última fila de la Tabla 2. Por ejemplo, la puerta AND 512A se acopla a las entradas x_3, x_2, x_1, u_1, u_3, u_4, y u_7, según la definición de la entrada en la última fila, tercera columna (x_1) en la Tabla 2. Las salidas de las puertas 512A hasta 512D se acoplan a las entradas de los registros 514A hasta 514D, respectivamente. Las salidas de los registros 514A hasta 514D comprenden las salidas de la máquina de estados x_1 hasta x_4, respectivamente.

De manera similar, cada puerta AND 522 dentro del generador de salida de fase de datos 520 se acopla selectivamente a las entradas u_0 - u_7 y x_1 - x_4, según se definen en la última columna de la Tabla 2. Por ejemplo, la puerta AND 522a se acopla a las entradas, x_3, x_2, x_1, u_0, según la definición de la entrada en la segunda fila, última columna (y_{b0}) en la Tabla 2. Las entradas u_0 hasta u_7 comprenden las salidas del codificador a través y_{a0} hasta y_{a7}, respectivamente (no se muestra en la Figura 5A para simplificar), y las salidas de las puertas 522A hasta 522H comprenden las salidas del codificador y_{b0} hasta y_{b7}, respectivamente.

La Figura 5B es un diagrama esquemático de un generador de salida de fase de cola de código 530 y multiplexores 540A y 540B que implementan la fase de cola de código de la matriz polinomio generador expresada en la ecuación (1) y para los formatos de paquetes 1 y 2 que se muestran en la Tabla 1. El generador de salida de fase de cola de código 530 y los multiplexores 540A y 540B corresponden al generador de salida de fase de cola de código 430 y los multiplexores 440A y 440B, respectivamente, en la Figura 4. En este ejemplo, el generador de salida de fase de cola de código 530 se implementa con las puertas AND 532A hasta 532J y genera las salidas del codificador Y_c e Y_d para la fase de cola de código definida en la Tabla 3. El multiplexor 540a se implementa con los multiplexores 2x1 542A hasta 542F y proporciona la primera salida del codificador Y_{ob}. Del mismo modo, el multiplexor 540B se implementa con los multiplexores 2x1 544a hasta 544H y proporciona la segunda salida Y_{ob} del codificador.

La máquina de estados del codificador 510, un generador de salida de la fase de datos 520, el generador de salida de fase de cola de código 530 y los multiplexores 540A y 540B en las Figuras 5A y 5B forman una implementación específica del codificador convolucional 400. Esta implementación específica se utiliza para implementar la matriz polinomio generador expresada en la ecuación (1) y para los formatos de paquete descritos en la Tabla 1.

Para el formato de paquetes 1, se proporcionan 1018 bits de datos al codificador convolucional 500 más de 128 ciclos de reloj. Para cada uno de los primeros 127 ciclos de reloj, se proporcionan ocho bits de datos al codificador 500 y los multiplexores 540A y 540B se seleccionan para proporcionar las salidas Y_a e Y_b del generador de salida de fase de datos 520. En el ciclo de reloj 128º, se proporcionan los dos bits de datos restantes, cuatro bits de cola de código y dos ceros al codificador 500. Los registros 514A hasta 514D se ponen a cero (se sincronizan) y los multiplexores 540A y 540B se seleccionan para proporcionar las salidas Y_c e Y_d del generador de salida de fase de cola de código 530. Para el formato de paquetes 2, se proporcionan 2042 bits de datos al codificador convolucional 500 durante 256 ciclos de reloj. Para cada uno de los primeros 255 ciclos de reloj, correspondientes a la fase de datos, ocho bits de datos se codifican en paralelo y los multiplexores 540A y 540B proporcionan las salidas Y_a e Y_b, respectivamente. En el ciclo de reloj 256º, correspondiente a la fase de cola de código, se codifican en paralelo dos bits de datos, cuatro bits de cola de código y dos ceros y los multiplexores 540A y 540B proporcionan las salidas Y_c e Y_d, respectivamente.

La aplicación específica mostrada en las Figuras 5A y 5B se describe para proporcionar una comprensión más clara. Se observará que diferentes implementaciones también pueden contemplarse y están dentro del alcance de la presente invención. Por otra parte, un diseño diferente se utiliza típicamente una matriz polinomio generador diferente, un número diferente de bits de datos de entrada o diferentes formatos de paquete.

De forma similar, otro codificador convolucional puede ser diseñado para implementar la matriz polinomio generador expresada en la ecuación (2). En una forma de realización, el codificador convolucional está diseñado para recibir y codificar cuatro bits de código en paralelo. Las ecuaciones (5) y (6) para el siguiente estado y salidas del codificador, respectivamente, puede resolverse de forma recursiva de la forma descrita anteriormente.

La Tabla 4 muestra los estados del codificador y las salidas después de que cuatro bits de código de entrada v_0 hasta v_3 se han proporcionado en serie al codificador convolucional 340 en la Figura 3. Los registros 344A y 344B almacenan inicialmente los valores de x_1 y x_2, respectivamente. En el primer ciclo de reloj, se proporciona el primer bit de código v_0 a un codificador 340, y la salida del sumador 342 se calcula como $x_1 + v_0$, que se almacena en la segunda fila, segunda columna de la Tabla 4. Las salidas del codificador se calculan como $y_{e0} = v_0$ y $y_{f0} = (x_1 + v_0) + x_2 + x_1 = x_2 + v_0$. En el siguiente ciclo de reloj, los valores del sumador 312 y el registro 344A se desplazan en los registros 344A y 344B, respectivamente. El siguiente bit de código v_1 se proporciona al codificador 340, y la salida del sumador 342 se calcula como $x_1 + v_0 + v_1$, que se almacena en la tercera fila, segunda columna. Las salidas se calculan como $y_{e1} = v_1$ y $y_{f1} = (x_1 + v_0 + v_1) + (x_1 + v_0) + x_1 = x_1 + v_1$. El procesamiento continúa hasta que se recibe y

se procesa el cuarto bit de código v_3.

El vector de salida del codificador Y_f se genera basándose en las entradas de la última columna de la Tabla 4. El estado del codificador X_{n+1}, después de la cuarta v3 bit de código ha sido codificado se genera basándose en las entradas de la última fila de la Tabla 4. Como se muestra en la Tabla 4, el vector de salida del codificador Y_f y el siguiente estado del codificador X_{n+1} son cada uno una función del estado actual del codificador $X_n = [x_1 \ x_2]$ y el vector de entrada V. Para la fase de datos, el vector de salida del codificador Y_e es simplemente una función del vector de entrada V.

Tabla 4

v	1	x_1	x_2	y_e	y_f
v_0	$x_1 + v_0$	x_1	x_2	v_0	$x_2 + v_0$
v_1	$x_1 + v_0 + v_1$	$x_1 + v_0$	x_1	x_1	$x_1 + v_1$
v_2	$x_1 + v_0 + v_1 + v_2$	$x_3 + x_2 + u_1$	$x_1 + v_0$	v_2	$x_1 + v_0 + v_2$
v_3	$x_1 + v_0 + v_1 + v_2 + v_3$	$x_1 + v_0 + v_1 + v_2$	$x_3 + x_2 + u_1$	v_3	$x_1 + v_0 + v_1 + v_3$
		$x_1 + v_0 + v_1 + v_2 + v_3$	$x_1 + v_0 + v_1 + v_2$		

Haciendo referencia de nuevo a la Tabla 1, el codificador convolucional interior en el sistema HDR 2044 recibe los bits de código y cuatro bits de cola de código para cada paquete en el formato de paquete 1. Si se codifican cuatro bits en paralelo, se utilizan 512 ciclos de reloj para codificar un paquete. Los primeros 511 ciclos de reloj se utilizan para codificar 2044 bits de código (es decir, 511 x 4 = 2044), y el ciclo de reloj 512º se utiliza para codificar los cuatro bits de cola de código. El codificador convolucional recibe 3,079 bits de código y codifica tres bits de cola para cada paquete en el formato de paquete 2. Si se codifican en paralelo cuatro bits, se utilizan 768 ciclos de reloj para codificar un paquete de datos. Los primeros 767 ciclos de reloj se utilizan para codificar 3068 bits de código (es decir, 767 x 4 = 3,068) y el ciclo de reloj 768º se utiliza para codificar el último bit de código y tres bits de cola de código.

La Tabla 5 muestra los estados y salidas del codificador convolucional interior para la fase de cola de código para el formato de paquete 1. En el primer ciclo de reloj, se proporciona el primer bit de cola de código con un valor x_1 al codificador. El valor de bit de cola de código se selecciona de tal manera que la salida del sumador 342 es igual a cero. Las salidas del codificador se calculan como $y_{g0} = x_1$ y $y_{h0} = x_2 + x_1$. El procesamiento continúa de manera similar para los tres bits restantes de cola de código.

Tabla 5

v	1	x_1	x_2	y_g	y_h
x_1	0	x_1	x_2	x_1	$x_2 + x_1$
0	0	0	x_1	0	x_1
0	0	0	0	0	0
0	0	0	0	0	0
		0	0		

La Tabla 6 muestra los estados y salidas del codificador convolucional interior para la fase de cola de código para el formato de paquete 2. En el primer ciclo de reloj se proporciona el último bit de código v_0 al codificador, y los estados del codificador x_1 y x_2 y salidas y_{i0} y y_{j0} se calculan de manera similar a la descrita anteriormente. La segunda fila de la Tabla 6 es por lo tanto idéntica a la segunda fila de la Tabla 4. En el segundo ciclo de reloj, se proporciona el primer bit de cola de código que tiene un valor $x_1 + v_0$ al codificador. El valor de bit de cola de código se selecciona de tal manera que la salida del sumador 342 es igual a cero. Las salidas del codificador se calculan como $y_{i1} = x_1 + v_0$ y $y_{j1} = v_0$. El procesamiento continúa de manera similar para los bits de cola de código restantes.

Tabla 6

v	1	x_1	x_2	y_i	y_j
v_0	$x_1 + v_0$	x_1	x_2	v_0	$x_2 + v_0$
$x_1 + v_0$	0	$x_1 + v_0$	x_1	$x_1 + v_0$	v_0
0	0	0	$x_1 + v_0$	0	$x_1 + v_0$
0	0	0	0	0	0
		0	0		

La Figura 6 es un diagrama esquemático de un codificador convolucional 600 que puede codificar cuatro bits de código de entrada en paralelo y aplica la matriz polinomio generador expresada en la ecuación (2). El codificador

convolucional 600 incluye una máquina de estado del codificador 610 que implementa la máquina de estado definida por la Tabla 4, un generador 620 de salida que genera las salidas del codificador definidas en las Tablas 4 a 6 y los multiplexores 640A y 640B que proporcionan las salidas del codificador adecuadas para los datos y fases de cola de código para los formatos de paquete 1 y 2.

Tal y como se muestra en la Figura 6, cuatro bits de código de entrada, v_0 hasta v_3, se proporcionan en paralelo a las entradas de la máquina de estados del codificador 610 y el generador de salida 620, cada uno de los cuales recibe también el estado del codificador actual definido como $X_n = [x_1 \ x_2]$. Cada puerta AND 612 dentro de la máquina de estados del codificador 610 se acopla selectivamente a las entradas $v_0 – v_3$ y $x_1 – x_2$, tal como se define en la última fila en la Tabla 4. Por ejemplo, y la puerta AND 612A se acopla a las entradas x_1, v_0, v_1, v_2, v_3 y v_4, tal como viene definido por el número de la última fila, tercera columna (x_1) en la Tabla 4. Las salidas de las puertas AND 612A y 6128 se acopla a las entradas de los registros G14A y 614B, respectivamente. Las salidas de los registros 614A y 614B comprenden las salidas de la máquina de estados x_1 y x_2, respectivamente.

Del mismo modo, cada puerta AND 622 dentro del generador de salida 620 se acopla de forma selectiva a las entradas $v_0 – v_3$ y $x_1 – x_2$, tal como viene definida por las dos últimas columnas en las Tablas 4 a 6. Por ejemplo, la puerta 622A se acopla a las entradas x_2 y v_0 y genera y_{f0} (segunda fila, última columna de la Tabla 4) y la puerta 622b se acopla a las entradas x_2 y x_1 y genera y_{h0} (segunda fila, última columna en la Tabla 5), y la puerta AND 622C se acopla a las entradas x_2 y v_0 y genera y_{j0} (segunda fila, última columna de la Tabla 6). Las otras salidas del codificador se generan tal y como se indica en las Tablas 4 a 6.

El multiplexor 640A incluye los multiplexores 3x1 642A hasta 642D que proporcionan las primeras salidas del codificador y_{ia0} hasta y_{ia3}, respectivamente, al codificador convolucional interior 600. Durante las fases de datos, y_{e0} hasta y_{e3} se proporcionan a los multiplexores 642a hasta 642D, respectivamente. Durante la fase de cola de código, los multiplexores 642a hasta 642D proporcionan respectivamente y_{g0} hasta y_{g3} para el formato de paquetes 1 y y_{i0} hasta y_{i3} para el formato de paquete 2. Del mismo modo, el multiplexor 640B incluye los multiplexores 3x1 644A hasta 644D que proporcionan las segundas salidas del codificador y_{ib0} hasta y_{ib3},respectivamente, para el codificador convolucional interior 600. Durante las fases de datos, se proporcionan y_{f0} hasta y_{f3} a través de los multiplexores 644A hasta 644D, respectivamente. Durante la fase de cola de código, los multiplexores 644A hasta 644D respectivamente proporcionan y_{h0} hasta y_{h3} para el formato de paquetes 1 y y_{j0} hasta y_{j3} para el formato de paquetes 2.

Otro aspecto de la invención proporciona un intercalador capaz de almacenar múltiples bits de código generados en paralelo por el codificador convolucional exterior y proporcionar múltiples bits de código en paralelo al codificador convolucional interior. Haciendo de nuevo referencia a la Figura 2, un intercalador está acoplado entre los codificadores convolucionales exterior e interior. El intercalador está diseñado para almacenar uno o más paquetes de bits de código. Después de que un paquete completo se haya almacenado, los bits de código se recuperan en un orden de lectura que es diferente al orden de escritura para lograr el intercalado de los bits de código. Si no se desea el intercalado, los bits de código se pueden recuperar desde el intercalador en el mismo orden.

El codificador convolucional exterior de la forma de realización ejemplar puede diseñarse para recibir y codificar bits de datos en paralelo y generar M • R bits de código, donde R está relacionada con la tasa de código del codificador convolucional exterior (por ejemplo, R = 2 para una tasa de código media). Para acelerar el proceso y reducir los retrasos, el intercalador puede estar diseñado para almacenar M • R bits de código desde el codificador convolucional exterior en paralelo al ir siendo generados los bits de código por el codificador. Del mismo modo, el codificador convolucional interior puede estar diseñado para recibir y codificar N bits de código en paralelo. Una vez más, para acelerar el proceso y reducir los retrasos, el intercalador puede ser diseñado para proporcionar por lo menos n bits de código en paralelo al codificador convolucional interior en una sola operación de lectura.

Los bits de código de cada uno de los codificadores convolucionales exterior e interior pueden ser perforados para proporcionar bits de código a otras tasas de código. Por ejemplo, al referirse de nuevo a la Tabla 1, las salidas del codificador convolucional exterior están sin perforar para el formato de paquete 1 para obtener una tasa de código de 1/2 y perforados para el formato de paquete 2 para obtener una tasa de código de 2/3. Del mismo modo, las salidas del codificador convolucional interior son sin perforación para el formato de paquete de 1 para obtener una tasa de código de 1/2 y se punza para el formato de paquete 2 para obtener una tasa de código de 3/4. La interfaz entre el codificador y el intercalador puede estar diseñada para lograr de manera eficiente el símbolo de perforación.

La Figura 7A es un diagrama de una realización de un intercalador 700. En esta forma de realización, el intercalador 700 se implementa con una memoria multi-puerto 710 que tiene P puertos, donde P es mayor que uno. Dependiendo de la unidad de memoria particular usada para implementar el dispositivo de intercalado, cada uno de los P puertos puede ser utilizado como puerto de lectura o escritura o puede ser un puerto dedicado de escritura o lectura. En la realización mostrada en la Figura 7A, la memoria 710 incluye W puertos designados como puertos de escritura D_1 a D_W, y R puertos designados como puertos de lectura Q_1 a Q_R. La memoria 710 incluye, además, P entradas de dirección, A_1 hasta A_P, Una entrada de dirección para cada uno de los P puertos. Cada puerto de escritura y lectura puede transferir los C bits en paralelo.

Un generador de direcciones 720 recibe una dirección de entrada ADDR, genera las direcciones necesarias para cada puerto activo y proporciona las direcciones generadas a las entradas de dirección A_1 hasta A_P de la memoria 710. Aunque no se muestra en la Figura 7A por simplicidad, el generador de direcciones 720 genera además una o más señales de control que dirigen la memoria 710 para llevar a cabo una operación de escritura o lectura.

En una forma de realización, la memoria 710 se configura como una memoria de dos dimensiones que tiene un número de filas y un número de columnas. En una forma de realización, los bits de código se escriben en filas secuenciales en la memoria 710. Por eficiencia, la anchura de cada fila puede corresponder a la anchura de cada puerto (es decir, C bits). Esto permite que las filas de W bits de código se escriban en los puertos W escritura de la memoria 710 en cada operación de escritura. Una vez que los bits de código para un paquete entero se han almacenado en la memoria 710 los bits de código se pueden recuperar de la memoria. En una forma de realización, los bits de código también se leen de la memoria 710 por filas. Para la realización mostrada en la Figura 7A, hasta R filas de bits de código se pueden recuperar de los R puertos de lectura para cada operación de lectura.

Varios diseños se pueden utilizar para proporcionar bits de código desde el intercalador 700 al codificador convolucional interior. El diseño particular a implementar depende de los requisitos del sistema en particular. En un diseño, los R multiplexores 730A través 730R están acoplados a los R puertos de lectura Q_1 hasta Q_R, respectivamente. Para cada operación de lectura, hasta R filas de bits de código se recuperan de la memoria 710 y se proporcionan a los multiplexores 730A 730R, que también recibe las señales de control AD1a través de ADR, respectivamente. Cada multiplexor 730 recibe los C bits de código, selecciona uno de los bits de código en base a la señal de control respectiva ADx, y proporciona el bit de código seleccionado a la salida del multiplexor. Las señales de control AD1 hasta ADR seleccionan un bit de código especial de cada fila recuperada de bits de código. Los R multiplexores 730 por lo tanto pueden ser utilizados para proporcionar hasta R bits de código en paralelo al codificador convolucional interior.

Para una comprensión más clara, un diseño específico del intercalador se describe ahora para su uso con los codificadores convolucionales exteriores e interiores descritos anteriormente en las Figuras 5A, 5B, y 6. En los diseños anteriores del codificador, el codificador convolucional exterior recibe y codifica 8 bits de datos en paralelo en un ciclo de reloj para generar 16 bits de código, y el codificador convolucional interior recibe y codifica 4 bits de código en paralelo. En este diseño específico de intercalador, se emplea una memoria de 8 puertos, con cuatro puertos que se utilizan para la recepción de bits de código en las operaciones de escritura y cuatro puertos que se utilizan para proporcionar bits de código en las operaciones de lectura. En este diseño, cada puerto es capaz de recibir o proporcionar 8 bits en paralelo. Por lo tanto, para este diseño específico, se pueden escribir hasta 32 bits de código en el dispositivo de intercalado en una operación de escritura y se pueden leer hasta 32 bits de código desde el dispositivo de intercalado en una operación de lectura.

La Figura 7B es un diagrama de una realización de la interfaz entre el codificador convolucional exterior y el intercalador sin perforación. En esta forma de realización, los bits de código generados por el codificador convolucional exterior se proporcionan a los cuatro registros 732A hasta 732D. Los registros 732A y 732B reciben los 16 bits de código generados en el primer ciclo de reloj y los registros 732c y 732D reciben los 16 bits de código generados en el segundo ciclo de reloj (por ejemplo, alternado). Cuando no se lleva a cabo ninguna perforación, los 32 bits de código en un registro 732A hasta 732D se proporcionan a los puertos D_1 a D_4, respectivamente, de la memoria en una operación de escritura.

La Figura 7C es un diagrama de una realización de la interfaz entre el codificador convolucional exterior y el intercalador con perforación. Haciendo de nuevo referencia a la Tabla 1, los bits de código para el código externo se perforan con el patrón de perforación (1011) para el formato del paquete 2. Por lo tanto, en un ciclo de reloj, se generan 16 bits de código, se perforan 4 bits de código, y se almacenan 12 bits de código. Inicialmente, los bits de código 16 generados en el primer ciclo de reloj se almacenan en los registros 732A y 732B, y los bits de código 16 generados en el segundo ciclo de reloj se almacenan en los registros 732c y 732D. Después de la perforación, se mantienen 24 bits de código, como se muestra en la Figura 7C y se proporcionan a los tres puertos de escritura (por ejemplo, D_1 a D_3).

El generador de direcciones proporciona las direcciones adecuadas para la escritura de los bits de código sin perforación en filas secuenciales en la memoria. Se genera una dirección para cada puerto activo utilizado para la escritura de los bits de código, De esta manera, el generador de direcciones genera cuatro direcciones para el puerto D_1 a D_4 cuando no se lleva a cabo perforación y genera tras direcciones para el puerto D_1 a D_3 cuando se realiza la perforación.

Para proporcionar cuatro bits de código en paralelo al codificador convolucional interior, se recuperan cuatro filas de bits de código de la memoria y se proporcionan a cuatro multiplexores 8x1. Cada multiplexor también recibe una señal de control respectiva de 3-bits ADx que selecciona un bit en particular en la fila recuperada para proporcionar al codificador convolucional interior. La dirección de cada bit recuperado puede, pues, ser dividida en dos partes, identificando la primera parte una fila en particular en la memoria y la segunda parte la identificando en una localización particular dentro de la fila. La primera parte de la dirección se proporciona a la entrada de dirección apropiada de la memoria y la segunda parte se proporciona como la señal de control ADx. Las partes primera y

segunda de la dirección se generan de acuerdo con el esquema de intercalado particular, definido por el sistema estándar aplicado.

El intercalador de la forma de realización de ejemplo también se puede implementar utilizando otras memorias. Por ejemplo, se pueden utilizar una unidad de memoria de puerto único o múltiples unidades de memoria para almacenar simultáneamente y proporcionar múltiples bits en paralelo. Para una unidad de memoria de un solo puerto, pueden utilizarse múltiples operaciones de escritura para almacenar los bits de código generados y también se pueden usar múltiples operaciones de lectura para recuperar los bits de código requeridos. En los diseños que emplean múltiples unidades de memoria, cada unidad de memoria puede ser operada de forma similar a un puerto (o un par de puertos) de la memoria multi-puerto. Por lo tanto, se pueden utilizar numerosos diseños para implementar el intercalador y están dentro del alcance de la presente invención.

En las realizaciones descritas anteriormente, un intercalador se utiliza entre los codificadores convolucionales exterior e interior. Esta configuración se utiliza para implementar un codificador turbo, que puede proporcionar ciertas ventajas. En otros diseños del codificador, puede no ser necesario el intercalado después del codificador convolucional exterior y puede no ser necesaria una memoria después del codificador convolucional exterior o simplemente puede ser utilizada como una memoria temporal.

El codificador concatenado de la forma de realización de ejemplo puede funcionar de diversas maneras. En un diseño especifico, el codificador se hace funcionar para codificar un paquete de datos cada vez. Haciendo de nuevo referencia a la Figura 2, un paquete en particular de datos puede ser codificado por el codificador convolucional exterior y almacenarse en el intercalador. Una vez que todo un paquete ha sido codificado por el codificador convolucional exterior, los bits de código se recuperan desde el intercalador y son codificados por el codificador convolucional interno. Una vez que todo el paquete ha sido codificado por el codificador convolucional interior, el siguiente paquete es codificado por el codificador convolucional exterior. Este diseño reduce los requisitos de memoria para el intercalador, que puede ser deseable en algunas aplicaciones.

En otro diseño especifico, el intercalador se implementa con la capacidad de almacenar dos más paquetes de bits de código. Por ejemplo, la memoria utilizada para implementar el intercalador puede ser dividida en dos bancos, siendo capaz cada banco de memoria de almacenar un paquete completo de bits de código. Los dos bancos de memoria permiten a los codificadores convolucionales exterior e interior operar en dos paquetes al mismo tiempo. El codificador convolucional exterior codifica un primer paquete y almacena los bits de código para este paquete en un banco de memoria. Una vez que todo el primer paquete se ha almacenado en la memoria, el codificador convolucional exterior codifica un segundo paquete y almacena los bits de código para este paquete en el segundo banco de memoria. Mientras que el codificador convolucional exterior codifica y almacena los bits de código para el paquete actual en un banco de memoria, el codificador convolucional interior puede recuperar y codificar los bits de código para el paquete anterior del otro banco de memoria. Este diseño puede reducir los retardos en el procesamiento.

La Figura 8 es un diagrama de bloques de un diseño específico de un codificador 800 que puede ser utilizado para implementar algunas formas de realización. El codificador 800 puede ser usado para implementar el codificador 114 en la Figura 1. El codificador 800 incluye una unidad de procesamiento 810 acoplada a un generador de direcciones 820 y a una memoria 830. La unidad de procesamiento 810 recibe datos de una memoria temporal 802 y la información de control desde una fuente de control (no mostrada), codifica los datos recibidos de conformidad con la información de control y proporciona los datos codificados a una memoria temporal 850.

En la realización mostrada en la Figura 8, la unidad de procesamiento 810 incluye una interfaz de entrada 812, un codificador de múltiples bits 814, una interfaz de salida 816 y una unidad de control 818. La interfaz de entrada 812 genera direcciones y señales de control para la memoria temporal 802, recibe los datos proporcionados por la memoria temporal 802 en respuesta a las direcciones generadas y señales de control y encamina los datos recibidos al codificador de múltiples bits 814. El codificador de múltiples bits 814 implementa los codificadores convolucionales interior y exterior. Cuando funciona como un codificador convolucional exterior, el codificador de múltiples bits 814 codifica los datos de la interfaz de entrada 812 y proporciona los bits de c6dlgo generados a la memoria 830 Y cuando funciona como un codificador convolucional interno, el codificador de múltiples bits 814 codifica los bits de código de la memoria 830 y proporciona los bits de código generados para la interfaz de salida 816. La interfaz de salida 816 proporciona entonces los datos codificados a la memoria temporal 850.

La unidad de control 818 recibe varias informaciones de control tales como, por ejemplo, el paquete de datos particular a codificar, la ubicación del paquete en la memoria temporal 802, el formato del paquete, el esquema de codificación a utilizar, la ubicación para almacenar el paquete codificado en la memoria temporal 850 y así sucesivamente. La unidad de control 818 dirige entonces la interfaz de entrada 812 para recuperar los bits de datos correspondientes de la memoria temporal 802, dirige el codificador de máquina de estados 814 para utilizar el esquema de codificación adecuado y, además, dirige la interfaz de salida 816 para proporcionar los datos codificados en la ubicación apropiada en la memoria temporal 850.

El generador de direcciones 820 genera las direcciones correspondientes para ambos bits de código que escriben

bits de código en la memoria 830 y leen de la memoria. El generador de direcciones 820 puede ser implementado con lógica, una tabla de consulta o algunos otros diseños.

La memoria 830 almacena los bits de código generados por el codificador de múltiples bits 814 y también proporciona los bits de código almacenados al codificador de múltiples bits 814. Mediante la generación de las direcciones correctamente, la memoria 830 puede ser operada para proporcionar el intercalado de los bits de código. La memoria 830 puede ser implementada con una memoria multi-puerto, tal y como se describió anteriormente con una o más unidades de memoria.

La Figura 9 es un diagrama de flujo de una realización de un procedimiento para llevar a cabo codificación concatenada de múltiples bits de datos en paralelo. Inicialmente, se recibe un número (M) de bits de datos de un paquete de datos en particular, en la etapa 912, y se codifica en paralelo de acuerdo con un primer (por ejemplo, convolucional) esquema de codificación para generar un número (MR) de bits de código, en la etapa 914. El número de bits de código generados por el primer esquema de codificación depende de la tasa de código particular del esquema. Cero o más de los bits de código generados se pueden perforar con un primer esquema de perforación, en la etapa 916, para proporcionar bits de código a una tasa de código diferente. Los bits de código sin perforar son entonces almacenados en una memoria, en la etapa 918.

En la realización mostrada en la Figura 9, un paquete completo es codificado por el primer esquema de codificación y almacenado antes de la posterior codificación por un segundo esquema de codificación. Esto permite el intercalado de los bits de código, tal y como se describió anteriormente. Por lo tanto, se hace una determinación de si todo el paquete ha sido codificado, en la etapa 920. 5i la respuesta es no, el proceso vuelve a la etapa 912 y se reciben otros M (o menos) bits de datos.

De lo contrario, si todo el paquete ha sido codificado, un número (N) de bits de código se recupera de la memoria, en la etapa 922, y se codifica en paralelo de acuerdo con el segundo (por ejemplo, convolucional) esquema de codificación para generar un número (NR) de bits de código, en la etapa 924. Una vez más, el número de bits de código generados por el segundo esquema de codificación depende de la tasa de código particular del esquema. Y de nuevo, cero o más de los bits de código generados se pueden perforar con un segundo esquema de perforación, en la etapa 926, para proporcionar bits de código a otras tasas de código. Los bits de código sin perforar se proporcionan como datos codificados a la siguiente unidad de procesamiento (por ejemplo, el modulador 116 en la Figura 1), en la etapa 928.

Para la eficiencia y la reducción de los retardos, W palabras pueden ser almacenadas en paralelo (por ejemplo, a través de los puertos de escritura W) en la memoria y R palabras pueden ser recuperadas en paralelo (por ejemplo, a través de R puertos de lectura) de la memoria. Las W palabras permiten el almacenamiento en paralelo de los bits de código sin perforar del primer esquema de codificación y las R palabras permiten proporcionar N bits de código en paralelo con el segundo esquema de codificación. La memoria puede ser operada de la manera descrita más arriba para lograr el intercalado de los bits de código. Por ejemplo, las W palabras pueden escribirse en filas secuenciales en las palabras de memoria y se pueden ser leer R palabras de filas permutables en la memoria.

El codificador y el intercalador de la realización a modo de ejemplo se pueden utilizar para acortar considerablemente el tiempo de codificación. Al codificar M bits de datos en paralelo con el codificador convolucional exterior y N bits de código en paralelo con el codificador convolucional interior, los retardos generales de codificación pueden ser reducidos significativamente. El intercalador de la invención da soporte a la codificación en paralelo con su capacidad de recibir múltiples bits de código para una operación de escritura y de proporcionar múltiples bits de código para una operación de lectura. La mejora en los retardos de procesamiento para un diseño especifico, con M = 8 y N = 4 y para los formatos de paquete 1 y 2 en el sistema HDR, tal y como se muestra en la Tabla 7.

Tabla 8

	Formato de paquete 1		Formato de paquete 2	
	paralelo	serie	paralelo	serie
Codificador convolucional exterior				
Bits de entrada	1018		2042	
Bits de cola de código	4		4	
Bits de entrada totales		1022	2046	
Ciclos de reloj necesarios	128	1024	256	2048
Codificador convolucional interior				
Bits de entrada	2044		3069	
	Formato de paquete 1		Formato de paquete 2	
	paralelo	serie	paralelo	serie
Bits de cola de código	4		3	
Bits de entrada totales	2048		3072	

Ciclos de reloj necesarios	512	2048	768	3072
Tiempo de codificación (reloj de 20 MHz)				
Codificador convolucional exterior (μseg)	6,4	51,2	12,8	102,4
Codificador convolucional interior (μseg)	25,6	102,4	38,4	153,6
Tiempo de codificación total (reloj de 20 MHz)	32,0	153,6	51,2	256,0

Para el diseño específico que se muestra en la Tabla 8, los retardos generales de codificación se reducen un factor de 4,8 para retardos proporcionados por el codificador y el intercalador de la presente invención proporciona numerosas ventajas. Algunas de estas ventajas se describen brevemente a continuación.

En primer lugar, los retardos de procesamiento más cortos pueden ser utilizados para dar soporte a ciertos tipos de servicios, tales como voz y video, que tienen requisitos de retardo más estrictos. Los retardos en el procesamiento más cortos pueden, por tanto, permitir el uso de esquemas de codificación más eficientes para aplicaciones sensibles al retardo.

En segundo lugar, los retardos de procesamiento más cortos pueden mejorar el rendimiento del sistema. Por ejemplo, si se selecciona un usuario en particular o una tasa de datos para una transmisión particular, en base a las condiciones del enlace de comunicaciones, que se determinan en un momento determinado, los retardos de procesamiento más cortos aumentan la probabilidad de que las condiciones de enlace no hayan cambiado en el momento de la transmisión de datos. Las condiciones del enlace normalmente varían con el tiempo y retardos de procesamiento más largos aumentan la probabilidad de que las condiciones de enlace hayan cambiado en el momento de la transmisión de datos, lo que puede dar lugar a un rendimiento degradado.

En tercer lugar, los retardos de procesamiento más cortos pueden mejorar la capacidad de algunos sistemas de comunicaciones. Por ejemplo, en el sistema HDR, los datos de control de patencia se multiplexan con los datos de tráfico y se transmiten a los terminales de usuario. Retardos de procesamiento más cortos permiten un control más preciso de la potencia de transmisión de los terminales de usuario, lo que puede aumentar la capacidad del sistema y mejorar el rendimiento.

Cuarto, retardos de procesamiento más cortos permiten el intercambio secuencial de un recurso hardware (es decir, el codificador) en una ranura de tiempo de procesamiento (es decir, la ranura de enlace directo en un sistema HDR) para múltiples entidades de transmisión (es decir, tres usuarios en un sistema de tres sectores) para reducir el área global del diseño hardware.

Para mayor claridad, ciertos aspectos y realizaciones del codificador de la invención se han descrito específicamente para el enlace directo en el sistema HDR. Sin embargo, la invención también se puede utilizar en otros sistemas de comunicaciones que emplean los mismos, similares o diferentes esquemas de codificación. Por ejemplo, el codificador de la invención se puede utilizar para implementar un codificador convolucional capaz de recibir y codificar múltiples bits de datos en paralelo. El codificador de la invención también se puede utilizar para implementar un codificador concatenado, tal como un codificador turbo, que es capaz de recibir y codificar múltiples bits de datos en paralelo. El diseño específico del codificador depende de varios factores tales como, por ejemplo, la matriz polinomio generador particular que se está implementando, el número de bits en el código en paralelo, del formato de paquetes, de la utilización de bits de cola de código y así sucesivamente.

El codificador de la invención se puede utilizar ventajosamente en una estación base o un terminal de usuario (por ejemplo, una unidad móvil, un teléfono y así sucesivamente) de un sistema de comunicaciones. La codificación para el enlace directo (es decir, enlace descendente) y el enlace inverso (es decir, enlace ascendente) puede ser diferente y depende típicamente del sistema estándar CDMA en particular que se esté aplicando. Por lo tanto, el codificador de la invención esta típicamente diseñado especialmente para la aplicación particular para la que se utiliza.

En referencia al diseño especifico que se muestra en las tablas 2 y 3, los próximos estados y salidas para el codificador convolucional exterior se pueden generar con funciones que tienen hasta siete términos. Haciendo referencia al diseño específico que se muestra en las Tablas 4 a 6, se pueden generar los siguientes estados y salidas para el codificador convolucional interno con funciones que tienen hasta cinco términos. Estas funciones se pueden generar fácilmente utilizando puertas lógicas de manera conocida en la técnica. Los otros elementos de los codificadores convolucionales exterior e interior (por ejemplo, registros, multiplexores) también se pueden implementar de una manera conocida en la técnica.

Algunos a todos los elementos descritos anteriormente para el codificador de la presente invención (por ejemplo, codificador de múltiples bits, interfaces de entrada y de salida, unidad de control, máquina de estados del codificador, generador de salida, multiplexor y así sucesivamente) se pueden implementar dentro de uno a mas

circuitos integrados de aplicación específica (ASIC), procesadores de señales digitales (DSP), dispositivo lógico programable (PLD), PLD Complejo (CPLD), controladores, microcontroladores, microprocesadores, otras unidades electr6nicas diseñadas para realizar las funciones descritas en el presente documento o una combinación de los mismos. Algunos o todos los elementos del codificador de la invención también pueden implementarse utilizando software o firmware ejecutado en un procesador.

Las memorias y las unidades de memoria, tales como las que se utilizan para implementar el intercalador de la presente invención pueden ser implementadas con diversas tecnologías de memoria, tales como, por ejemplo, memoria de acceso aleatorio (RAM), RAM dinámica (DRAM), memoria Flash y otros. La unidad de memoria puede implementarse también con elementos de almacenamiento tales como, por ejemplo, un disco duro, una unidad de CD-ROM y otros. Varias otras aplicaciones de las unidades de memoria son posibles y entran dentro del alcance de la presente invención.

Codificación recursiva de múltiples bits en paralelo

De acuerdo con una forma de realización, los codificadores están configurados en paralelo para proporcionar el doble de la cantidad de datos de salida, en donde múltiples bits son procesados por el codificador. El aumento en la salida de datos es particularmente aplicable a un sistema de comunicación de alta velocidad de datos en la que las tramas han de ser codificadas rápidamente. El ejemplo de la realización codifica múltiples bits por ciclo de reloj, cumpliendo así con las limitaciones de tiempo de transmisión de datos. Esla realización evita el uso de un codificador por sector mediante el intercambio de un solo codificador a lo largo de múltiples sectores. Realizaciones alternativas pueden implementar cualquier número de codificadores en paralelo. Al compartir la sección de codificación en todos los sectores, la velocidad del codificador individual puede ser menos estricta.

De acuerdo con un aspecto de la forma de realización de ejemplo memorias temporales de tramas almacenan múltiples copias de cada trama. Se utilizan circuitos de Tabla de Consulta (LUT) paralela y multiplicadores para implementar los generadores de direcciones del intercalador turbo. El diseño utiliza los arboles AND-XOR para implementar codificación paralela. El bit de perforación/reordenación también se hace en paralelo después del proceso de codificación.

La Figura 10 sirve como ejemplo de un sistema de comunicaciones 1000 que da soporte a un número de usuarios y es capaz de implementar al menos algunos aspectos y realizaciones de la invención. Se puede utilizar cualquiera de una variedad de algoritmos y procedimientos para planificar transmisiones en el sistema 1000. El sistema 1000 proporciona comunicación a un número de células 1020A hasta 1020G, cada una de los cuales es atendida, respectivamente, por una estación base correspondiente 1040A hasta 1040G. En la realización de ejemplo, algunas de las estaciones de base 1040 tienen múltiples antenas de recepción y otras tienen sólo una antena de recepción. Del mismo modo, algunas de las estaciones de base 1040 tienen múltiples antenas de transmisión y otras tienen antenas de transmisión individuales. No hay restricciones en las combinaciones de antenas de transmisión y antenas de recepción. Por lo tanto, es posible para una estación base 1040 tener múltiples antenas de transmisión y una única antena de recepción, o tener múltiples antenas de recepción y una única antena de transmisión o tener antenas de recepción y transmisión tanto únicas como múltiples.

Los terminales 1060 en el área de cobertura pueden ser fijos (es decir, estacionarios) o móviles. Como se muestra en la Figura 1, diversos terminales 1060 están dispersos por todo el sistema. Cada terminal 1060 se comunica con al menos una y posiblemente más estaciones base 1040 en el enlace descendente y el enlace ascendente en cualquier momento dado dependiendo de, por ejemplo, si se emplea traspaso continuo o si el terminal está diseñado y se opera para (concurrentemente o secuencialmente) recibir múltiples transmisiones desde múltiples estaciones base. El traspaso continuo en sistemas de comunicaciones CDMA es bien conocido en la técnica y se describe en detalle en la patente estadounidense número 5.101.501, titulada "Method and system for providing a Soft Handoff in a CDMA Celullar Telephone System", asignada al cesionario de la presente invención.

El enlace descendente se refiere a la transmisión desde la estación base al terminal y el enlace ascendente se refiere a la transmisión desde el terminal a la estación base. En la realización de ejemplo, algunos de los terminales 1060 tienen múltiples antenas de recepción y otras tienen sólo una antena de recepción. En la Figura 1, la estación base 1040A transmite datos a los terminales 1060A y 1060J en el enlace descendente, la estación base 1040B transmite datos a los terminales 1060B y 1060J, la estación base 1040G transmite datos al terminal de 1060G y así sucesivamente.

De acuerdo con una realización de ejemplo, un sistema de comunicación inalámbrica está adaptado para codificar información para la transmisión utilizando múltiples codificadores convolucionales configurados en paralelo. Cada uno de los codificadores individuales tiene una estructura similar y están acoplados mediante un intercalador. Los codificadores paralelos proporcionan un número múltiple de salidas, es decir, para dos codificadores en paralelo; la combinación proporciona el doble de valores de salida. Se realiza una selección a continuación, en la salida de los valores de salida que se utilizará en su posterior procesamiento. Se procesan múltiples bits mediante los codificadores paralelos. El proceso dentro de cada codificador se realiza en paralelo.

La forma de realización de ejemplo procesa múltiples bits por ciclo de reloj del sistema, por ejemplo, cuatro bits por ciclo. El codificador de la forma de realización de ejemplo se implementa utilizando una combinación de hardware y software. El hardware se utiliza para almacenar y procesar los bits de entrada de información. El software incluye instrucciones para controlar el hardware y otros cálculos de codificación, por ejemplo, la generación de valores intermedios durante el proceso de codificación.

El bloque codificador turbo 1500 que se ilustra en la Figura 11 emplea dos codificadores convolucionales sistemáticos 1502,1552 conectados en paralelo, con un intercalador 1522 que precede al segundo codificador convolucional 1552. El intercalador 1522 se conoce como intercalador turbo. Los dos códigos convolucionales se denominan códigos que constituyen el código turbo. Las salidas de los codificadores constituyentes son perforadas y repetidas para lograr el número deseado de símbolos de salida del codificador turbo. La función de transferencia del código constituyente de acuerdo con una forma de realización se da como:

$$G(D) = \left[1 \quad \frac{n_0(D)}{d(D)} \quad \frac{n_1(D)}{d(D)}\right] \qquad \text{Ecuación (9)}$$

En la que:

$$d(D) = 1 + D^2 + D^3 \qquad \text{Ecuación (10)}$$

$$n_0(D) = 1 + D + D^3$$

$$n_1(D) = 1 + D + D^2 + D^3 .$$

Cada uno de los codificadores constituyentes 1502, 1552 incluye una pluralidad de registros, en concreto dentro del codificador 1502 los registros 1510, 1512 y 1514, Y dentro de codificador 1552 los registros 1560, 1562, y 1564. Inicialmente, los estados de los registros dentro de los codificadores constituyentes 1502, 1552 se ponen a cero. Cada codificador 1502 a 1552 recibe datos de un conmutador de entrada 1501, 1551, respectivamente. Los bits de información se proporcionan como entrada al primer codificador 1502 mediante el conmutador 1501. Los bits de información de entrada incluyen N_{turbo} bits, que es efectivamente el número de bits en el codificador 1500. Los bits de información de entrada se proporcionan además a un intercalador turbo 1522, en el que los bits se intercalan, es decir, se barajan, para aumentar la exactitud de la transmisión de datos. La salida del intercalador turbo 1522 se proporciona al segundo codificador 1552 a través del conmutador 1551. La operación de cada uno de los codificadores 1502 y 1552 es similar y por lo tanto la siguiente discusión sólo detalla el funcionamiento del codificador 1502. Realizaciones alternativas pueden implementar diferentes tipos de codificadores para cada codificador incluido en el bloque codificador turbo 1500.

La entrada al codificador 1502 se proporciona a un conmutador, en donde el conmutador es controlado por un reloj del sistema (no mostrado). Los bits de información se introducen una vez para cada uno de los N_{turbo} periodos de bits de datos con el conmutador hacia arriba y luego se introducen varias veces para cada uno de los periodos de bits de cola con el conmutador hacia abajo. Según un ejemplo, los bits de Información son Introducidos 6 veces durante el periodo de bit de cola, incluyendo 3 veces para cada codificador 1502, 1552. Los símbolos de salida de datos codificados se generan mediante la conmutación de los codificadores constituyentes 1502, 1552 N_{turbo} veces con los conmutadores en las posiciones hacia arriba y perforando las salidas de acuerdo con un modelo de perforación predeterminado. La salida para el codificador 1500 se genera en la secuencia: X, Y_0, Y_1, X', Y'_0, Y'_1. De acuerdo con el ejemplo de realización, la repetición de símbolos no se implementa en la generación del símbolo de salida. El codificador turbo 1500 genera símbolos de salida de cola que se añaden a los símbolos de salida de datos codificados. Los símbolos de salida de cola se generan después de que los codificadores constituyentes 1502, 1552 se hayan registrado N_{turbo} veces.

El funcionamiento del intercalador turbo 1522 está diseñado para producir un resultado equivalente funcional como si cada uno de la secuencia de bits de entrada se escribiese secuencialmente a una matriz en una secuencia de direcciones, en donde la secuencia se leyó entonces de otra secuencias de direcciones definidas por un procedimiento de intercalado predeterminado o protocolo. El funcionamiento del intercalador se detalla aún más con respecto a la Figura 12.

Continuando con la Figura 11, un nodo del conmutador 1501 esta acoplado a la entrada. Un segundo nodo del conmutador 1501 esta acoplado a una puerta OR exclusiva (XOR). La salida de la puerta XOR 1504 esta acoplada a una serie de registros o elementos de retardo 1510, 1512, 1514. Cada elemento de retardo tiene un estado asociado, en el que la información almacenada en el elemento de retardo 1510 se denomina "estado 0"; la información almacenada en el elemento de retardo 1512 se denomina "estado 1" y la información almacenada en el elemento de retardo 1514 se denomina "estado 2". La salida del elemento de retardo 1510 se identifica como "S0" la salida del elemento de retardo 1512 se identifica como "S1"; la salida del elemento de retardo 1514 se identifica como "S2".

Las salidas de los elementos de retardo 1512 y 1514 están cada una acopladas a las entradas de una puerta XOR 1516. La salida de la puerta XOR 1516 se acopla entonces a un tercer nodo del conmutador de entrada 1501 y a una entrada de la puerta XOR 1504. La salida de la puerta XOR 1504 esta acoplada además a una entrada de una puerta XOR 1508. Otras entradas a XOR 1508 se reciben de cada una de las salidas individuales de elementos de retardo 1510, 1512 y 1514. La salida de la puerta XOR 1504 está acoplada además a una entrada de la puerta XOR 1506. Otras entradas a la puerta XOR 1506 se reciben de las salidas individuales de los elementos de retardo 1510 y 1514.

La salida del codificador 1502 incluye un componente X directamente desde el conmutador de entrada 1501, una salida de bit de paridad Y_0 de la puerta XOR 1506, y un segundo componente de salida de bit de paridad Y, desde la salida de la puerta XOR 1508. Las salidas de X, Y_0 e Y_1, se proporcionan cada una a una unidad de repetición y perforación de símbolos 1520.

Funcionalmente, la configuración del codificador 1520 implementa las siguientes ecuaciones:

$$X = I \qquad \text{Ecuación (13)}$$

$$Y_0 = [I \oplus (S1 \oplus S2)] \oplus S0 \oplus S2 \qquad \text{Ecuación (14)}$$

$$Y_1 = [I \oplus (S1 \oplus S2)] \oplus S0 \oplus S1 \oplus S2 \qquad \text{Ecuación (15)}$$

en las que *I* representa los bits de información de entrada, S0, S1 y S2 representan las salidas de los elementos de retardo 1510, 1512 y 1514, respectivamente, y la operación representa la operación lógica XOR. Aplicando las reglas asociativa y distributiva de la lógica digital, las ecuaciones (10) y (11) se pueden reducir a:

$$Y_0 = I \oplus S1 \oplus S0 \qquad \text{Ecuación (16)}$$

$$Y_1 = I \oplus S0. \qquad \text{Ecuación (17)}$$

De acuerdo con el ejemplo, el codificador turbo tiene dos etapas. Durante la primera etapa, se lee la trama desde una fuente externa. El CRC se calcula también en la primera etapa. Durante la segunda etapa, la trama se codifica, se perfora y se repite. La tasa de código para el codificador turbo puede ser 1/3 ó 1/5.

Durante la segunda etapa, se reciben cuatro bits en el codificador 1500, en el que los cuatro bits se procesan en

paralelo a fin de aumentar el rendimiento del codificador. Efectivamente, a pesar de que los bits de información de entrada I[0]:I[3] se presentan simultáneamente al codificador 1500, los bits de información de entrada se procesan como si se le presentasen al codificador 1500 en serie. Esto se logra aplicando de forma recursiva las ecuaciones (16) y (17) a los datos de entrada. Durante un ciclo de reloj individual del sistema, se determinan los valores de los estados, es decir, S0[0]:S0[4], S1[0]:S1[4],S2[0]:S2[4], respectivamente.

La Figura 12 ilustra el funcionamiento de un intercalador de acuerdo con una realización. Como se ilustra, un intercalador 1300 recibe una entrada de un contador (no mostrado) para incrementar la circuitería de suma 1302. Funcionalmente, la generación de dirección es equivalente a escribir valores de contador en una matriz de filas de almacenamiento de memoria. Las filas se mezclan de acuerdo con una inversión de bits u otra regia, y los elementos en cada fila se permutan de acuerdo a una secuencia de congruencia lineal específica de la fila. Las direcciones de salida se leen per columnas. La secuencia de congruencia lineal se puede dar como:

$$x(i+1) = \left(x(i) + c\right) \bmod 2^n, \qquad \text{Ecuación (18)}$$

en la que x (0) = c y c es un valor específico de fila de una tabla de búsqueda.

Continuando con la Figura 12, el bit de información más significativo (MSB) a partir del valor del contador se proporciona a la unidad de suma 1302. La unidad de suma 1302 incrementa el valor MSB del contador y proporciona el resultado a la unidad de multiplicador 1304. En una forma de realización, el valor resultante está codificado a fin de proporcionar sólo un número predeterminado de bits como salida. El bit menos significativo (LSB) de información del valor del contador se proporciona a la tabla de consulta (LUT) 130B y a la unidad de inversión de bits 1310. La información LSB se utiliza para tratar la LUT 130B, en donde el valor almacenado en esa ubicación también se proporciona a la unidad de multiplicar 1304. Las entradas de la unidad de multiplicar 1304 se multiplican entre sí y el producto se entrega a la unidad de selección 1306. En una forma de realización, la unidad de multiplicar 1304 proporciona sólo una parte del producto como salida a la unidad de selección 1306 tal como la parte LSB del producto. La unidad de inversión de bits 1310 lleva a cabo una operación de inversión de bits, similar a la que se ha descrito anteriormente, en la parte LSB del valor del contador. La salida de la unidad de inversión de bits 1310 se proporciona a la unidad de selección 1306. De acuerdo con la realización de ejemplo, la entrada que la unidad de selección 1306 recibe de la unidad de multiplicar 1304 se utiliza como una sección LSB y la entrada recibida de la unidad de inversión de bits 1310 se utiliza como una parte MSB. La unidad de selección 1306 también determina si la dirección de salida resultante es una dirección válida. Si la dirección no es válida, la unidad de selección descarta el resultado y en el siguiente incremento del contador se genera una nueva dirección. Realizaciones alternativas pueden implementar esquemas de intercalado alternos aplicados entre los codificadores convolucionales paralelos.

La Figura 13 ilustra un codificador de una realización de ejemplo. El codificador de la forma de realización de ejemplo codifica múltiples bits por ciclo de reloj satisfaciendo así los requisitos de velocidad de un sistema de transmisión de datos de alta velocidad. Por ejemplo, como se ilustra en la Figura 13, la operación funcional del codificador 1100 codifica cuatro bits por ciclo de reloj, donde el codificador 1100 está diseñado para codificar una trama de tamaño máximo en aproximadamente 32 ms usando un reloj de 40 MHz. Como se analizó anteriormente en este documento, el codificador 1100 opera en dos etapas. Durante la primera etapa, se lee una trama a partir de una fuente externa y se calcula el CRC. Durante la segunda etapa, la trama se codifica, se perfora y se repite. Tal como se utiliza en la descripción de la forma de realización de ejemplo, una trama es una unidad de transmisión de datos que tiene una parte de cabecera y una parte de carga útil.

La Figura 13 es una ilustración funcional de funcionamiento del codificador 1100 de acuerdo con la forma de realización de ejemplo. El codificador 1100 recibe bits de información como palabras de 16 bits de una fuente externa que de acuerdo con una forma de realización es una pluralidad de memorias de acceso aleatorio (RAM). Se almacenan cinco capias idénticas en la memoria del codificador 1100. Los bits se proporcionan selectivamente a un codificador paralelo 1132 a través de un demultiplexor 1130. De acuerdo con una realización de ejemplo, se proporcionan 16 bits desde la memoria 1128 al demultiplexor 1130, que selecciona 4 bits cada vez para su presentación al codificador paralelo 1132. El codificador paralelo recibe los 4 bits de entrada y genera tres salidas, X, Y_0 e Y_1. Cada una de las salidas es una salida de 4 bits, en donde el codificador paralelo utiliza procesamiento recursivo para producir salidas de 4 bits para cada ciclo de reloj.

Como se analizó anteriormente, el generador de CRC 1134 opera durante el primer estado, en donde se calcula un CRC de 16 bits en el paquete que se está procesando actualmente. Un paquete incluye una carga útil, una parte CRC y una parte de cola. Una realización es compatible con paquetes de longitud variable. Como los datos se leen 16 bits por cada ciclo de reloj, el generador de CRC 1134 calcula el CRC de cada ciclo. Para el final de la primera etapa, el CRC está listo. En este punto, el CRC se escribe en la unidad de almacenamiento de memoria (MEM) 1128 y también en cuatro dispositivos de almacenamiento de memoria MEM 1104 a 1106. También durante la primera etapa, los bits de información se proporcionan a las MEMS 1104 a 1106. Los bits de información se introducen en

las MEMS 1104 a 1106, en donde se introducen 16 bits cada ciclo de reloj. Cabe destacar que las MEMs de la realización de ejemplo 1104 a 1106 incluyen cuatro memorias, sin embargo, realizaciones alternativas pueden incluir números alternativos de memorias. Las MEMs 1104 a 1106 recibirán la información de direccionamiento de control del generador de direcciones 1124 y del contador 1126, las cuales están cada una acoplada a las entradas de un multiplexor 1136. La salida del multiplexor 1136 proporciona la señal de control TNE a las MEMs 1104 a 1106. El generador de direcciones 1124 incrementa la dirección para el almacenamiento de cuatro valores. Durante una operación de escritura en las MEMs 1104 a 1106, cada una de las MEMs 1104 a 1106 recibe la misma dirección. Durante una operación de lectura de las MEM 1104 a 1106, cada una de las MEMs 1104 a 1106 recibe una dirección diferente. Como se ilustra en la Figura 13, la MEM 1128 alimenta uno de los codificadores paralelos 1132, mientras que las MEMs 1104 a 1106 alimentan un segundo codificador paralelo 1144. De los codificadores paralelos 1132 y 1144, cada uno de los cuales proporciona conjuntos de salida X, Y_0 e Y_1 los bits de salida se proporcionan a un bloque de repetición y perforación de símbolos, tales como el bloque 1520 en la Figura 11.

Tal y como se ilustra en la Figura 14, los bits de información *I*[0] : *I*[3] se presentan a un codificador 1200, similar a los codificadores 1500 y 1100. El codificador 1200 incluye el generador de estados futuros 1202 para la aplicación de las Ecuaciones (16) y (17) a los bits de información de entrada *I*[0] : *I*[3]. El generador de estados futuros 1202 genera la información de estado y almacena los estados S0[4], S1[4], S2[4] en un registro o memoria del dispositivo de almacenamiento 1204. La información de estado se actualiza cada ciclo de reloj del sistema. Antes de almacenar los primeros valores, el dispositivo de almacenamiento de memoria 1204 se inicializa con valores de estado predeterminados. Los valores de estado S0[0]:S0[4], S1[0]:S1[4], S2[0]:S2[4] se proporcionan a continuación a los generadores de salida de varios bits 1203, 1208. El bit de información de entrada *I*[0] : *I*[3] se proporciona como las salidas X[0]:X[3]. El generador de salida de varios bits 1206 genera las salidas Y_0[0]:Y_0[3]; mientras que el generador de salida de varios bits 1208 genera las salidas Y_1[0]: Y_1[3]. Los generadores de salida de varios bits 1206 y 1208 calculan de forma recursiva valores en base a las ecuaciones (16) y (17) dadas anteriormente.

Como se analizó anteriormente, la generación de dirección de la realización de ejemplo proporciona cuatro direcciones de lectura a cuatro memorias del intercalador turbo 1104,.... 1106. Las direcciones del intercalador turbo no tienen un patrón discernible y es por lo tanto deseable generar cuatro copias de cada dirección para obtener un rendimiento de lectura de 4 bits por reloj. Cada uno de las memorias del intercalador 1104,..., 1106 proporcionan una palabra de 16 bits como palabra de lectura, se selecciona un bit de cada palabra de16 bits que se lee a través de múltiples multiplexores 16:1. En la realización de ejemplo, cada intercalador 1104,...,1106 esta acoplado a un multiplexor 1140...., 1142, respectivamente. Los 4 bits (es decir, un bit de cada intercalador 1104..... 1106) se pasan al segundo codificador 1144.

El tiempo total de codificación es el tiempo que se tarda en leer los bits en la memoria durante la primera etapa, más el tiempo para codificar durante la segunda etapa. Por ejemplo, considere un tamaño de fotograma de 4096 bits, en donde el número aproximado de ciclos para codificar la trama viene dado por:

$$\frac{4096}{16}+\frac{4096}{4}=1280. \qquad \text{Ecuación (19)}$$

Por lo tanto, para un sistema que tiene un sistema de reloj de 40 MHz, se necesitaran aproximadamente 32μs para codificar una trama de 4096 bits, que está dentro de un periodo de tiempo de codificación objetivo de 40μs.

Como se ha descrito anteriormente en este documenta, el codificador de dos etapas proporciona todo el paquete residente en una estructura de memoria interna. En dicha estructura se proporciona la información de entrada al codificador mediante un puerto de lectura capaz de procesar cuatro bits, es decir, de cuatro puertos de lectura. Una memoria fuente externa de tramas es generalmente un puerto de lectura y por lo tanto se utiliza un procedimiento alternativo para codificar la trama de esta memoria directamente. La realización de ejemplo proporciona un procesamiento recursivo de múltiples bits de información pro ciclo de reloj a fin de proporcionar cuatro bits codificados cada ciclo de reloj.

El generador de CRC 1134 y los codificadores 1132 y 1144 operan sobre datos a velocidades superiores a 1 bit por ciclo de reloj. El ejemplo de realización implementa una estructura de árbol AND-XOR para permitir procesamiento en paralelo. Las realizaciones alternativas pueden implementar cualquier estructura lógica que implemente de forma recursiva las ecuaciones (13), (14) y (15). A cada árbol AND-XOR se le da una matriz bidimensional única de bits que determinan las taps del árbol AND-XOR. Por ejemplo, considere los codificadores 1132, 1144, en donde cada uno incluye un estado de interno 3-bits con diferentes taps XOR para las salidas de bits de paridad, es decir, Y_0, Y_1. Cada codificador 1132,1144 codifica 4 bits por ciclo de reloj en la ejecución en paralelo, en donde un codificador de tasa 1/3 producirá 12 bits de datos por ciclo de reloj, es decir, 4 bits X, 4 bits Y_0, 4 bits Y_1. Cada bit de salida depende de todos los 4 bits de entrada, así como del estado actual. Cada codificador incluye 3 árboles AND-XOR

que generan los dos grupos siguientes de valores de salida de 4 bits, asl como el siguiente estado de 3 bits. La salida X se proporciona directamente a la entrada del codificador y no se proporciona a través de un árbol AND-XOR.

En la realización de ejemplo, se requieren múltiples direcciones válidas por ciclo de reloj. De acuerdo con el ejemplo de realización, las múltiples direcciones incluyen cuatro direcciones. Se utilizan cuatro circuitos independientes para generar las cuatro direcciones de lectura independientes. Para el codificador 1144, se utilizan 4 bits de entrada por ciclo de reloj. Estos 4 bits de entrada provienen de cuatro ubicaciones de dirección de intercalador diferentes en las 4 memorias de tramas y por lo tanto 4 generadores de direcciones proporcionan las 4 direcciones.

Como un ejemplo de la operación recursiva se detalla en la operación especifica de la Figura 14, considere la siguiente aplicación. Los estados generados y almacenados en los elementos 1510, 1512 y 1514 se identifican como los estados S_0, S_1 y S_2 respectivamente. El cálculo para cada estado en una iteración dada se determina mediante el siguiente conjunto de ecuaciones.

$$S_0[n+1] = I[n] \oplus S_1[n] \oplus S_2[n] \quad \text{Ecuación (20)}$$

$$S_1[n+1] = S_0[n] \quad \text{Ecuación (21)}$$

$$S_2[n+1] = S_1[n] \quad \text{Ecuación (22)}$$

en las que n es el índice de iteración. El codificador 1500 ha recibido una entrada I[0], correspondiente a la entrada en la iteración 0. En consecuencia, cada uno de los elementos 1510, 1512, y 1514 se han inicializado a unos valores $S_0[0]$, $S_1[0]$, $S_2[0]$. En este caso, para la iteración n = 1, las ecuaciones se implementan como:

$$S_0[1] = I[0] \oplus S_1[0] \oplus S_2[0] \quad \text{Ecuación (23)}$$

$$S_1[1] = S_0[0]$$

$$S_2[1] = S_1[0] \quad \text{Ecuación (25)}$$

en la que los valores de entrada y los valores de estado para n = 0 (en la inicialización). Del mlsmo modo, en la iteración n = 2, los valores de iteración n = 1 se almacenan en los elementos 1510, 1512, y 1514 y se utilizan para calcular valores de estado como:

$$S_0[2] = I[1] \oplus S_1[1] \oplus S_2[1] \quad \text{Ecuación (26)}$$

$$S_1[2] = S_0[1] \quad \text{Ecuación (27)}$$

$$S_2[2] = S_1[1]. \quad \text{Ecuación (28)}$$

Utilizando los valores y las relaciones generadas previamente, las ecuaciones (26), (27) y (28) resultarán en:

$$S_0[2] = I[1] \oplus S_1[1] \oplus S_2[1] \quad \text{Ecuación (29)}$$

$$S_0[2] = I[1] \oplus S_0[0] \oplus S_1[0] \quad \text{Ecuación (30)}$$

$$S_1[2] = S_0[1] \quad \text{Ecuación (31)}$$

$$S_1[2] = I[0] \oplus S_1[0] \oplus S_2[0] \quad \text{Ecuación (32)}$$

$$S_2[2] = S_1[1] \quad \text{Ecuación (33)}$$

$$S_2[2] = S_0[0]. \quad \text{Ecuación (34)}$$

Los resultados para la iteración n = 3 se dan como:

$$S_0[3] = I[2] \oplus S_1[2] \oplus S_2[2] \quad \text{Ecuación (35)}$$

$$S_0[3] = I[2] \oplus (I[0] \oplus S_1[0] \oplus S_2[0]) \oplus S_0[0] \quad \text{Ecuación (36)}$$

$$S_1[3] = S_0[2] \quad \text{Ecuación (37)}$$

$$S_1[3] = I[1] \oplus S_0[0] \oplus S_1[0]$$

$$S_2[3] = S_1[2]$$

$$S_2[3] = I[0] \oplus S_1[0] \oplus S_2[0]. \quad \text{Ecuación (40)}$$

Del mismo modo, los resultados para la iteración n = 4 se dan como:

Ecuación (41)

$$S_0[4] = I[3] \oplus S_1[3] \oplus S_2[3] \qquad \text{Eq (41)}$$

$$S_0[4] = I[3] \oplus (I[1] \oplus S_0[0] \oplus S_1[0]) \oplus (I[0] \oplus S_1[0] \oplus S_2[0]) \qquad \text{Ecuación (42)}$$

$$S_0[4] = I[0] \oplus (I[1] \oplus I[3] \oplus S_0[0]) \oplus S_2[0] \qquad \text{Ecuación (43)}$$

$$S_1[4] = S_0[3] \qquad \text{Ecuación (44)}$$

$$S_1[4] = I[2] \oplus I[0] \oplus S_1[0] \oplus S_2[0] \oplus S_0[0] \qquad \text{Ecuación (45)}$$

$$S_2[4] = S_1[3] \qquad \text{Ecuación (46)}$$

$$S_2[4] = I[1] \oplus S_0[0] \oplus S_1[0] \qquad \text{Ecuación (47)}$$

Tenga en cuenta que algunas iteraciones tienen como resultado relaciones de cálculo o de estado similares que pueden ser explotadas en el diseño y funcionamiento del codificador.

Continuando con el funcionamiento del codificador 1100, durante la primera etapa, se escriben memorias dentro del codificador con los mismos datos en las mismas direcciones de memoria. Durante la segunda etapa, las memorias se leen de forma independiente en diferentes direcciones.

Volviendo a la Figura 12, el circuito de la Figura 12 produce una dirección por ciclo de reloj. El contador de entrada es una potencia de dos, y por lo tanto, se pueden producir algunas direcciones que están fuera del rango definido por el tamaño de bloque del intercalador turbo. Las direcciones no válidas son detectadas y omitidas por el generador de direcciones. Un registro de segmentación se puede insertar a medio camino a través del circuito de generación de dirección para aumentar la velocidad de funcionamiento.

Un circuito de generación de dirección de intercalador turbo 1400 de acuerdo con una realización se ilustra en la Figura 15. Una señal de habilitación y un indicador de tamaño del paquete se proporcionan a un puntero de dirección 1402. La salida del puntero de dirección 1402 se proporciona a los caminos de circuito paralelas y a las LUT 1404, 1454 que se utilizan para incrementar los valores de dirección. Las unidades de agregación 1406, 1456 añaden 1 bit a los 2 bits recibidos de cada una de las LUT 1404, 1454, respectivamente. Las salidas de las unidades de agregación 1406, 1456 se proporcionan a los sumadores 1408, 1458, respectivamente. El resultado de la operación de adición se proporciona, a continuación, a los multiplexores 1410, 1460. Una señal de habilitación se proporciona a cada uno de los multiplexores 1410, 1460, que cada uno produce 12 bits. Las salidas de los multiplexores 1410, 1460 se proporcionan a elementos de retardo 1412, 1462, cuya salida se alimenta de nuevo a los sumadores 1408, 1458. Las salidas de los elementos de retardo 1412, 1462 se proporcionan a una red de elementos de retardo incluyendo la LUT codificador turbo 1414, 1464. Los 7 bits más significativos de la salida de elementos de retardo 1412, 1462 se proporcionan a los elementos de retardo 1416, 1466. Los 5 bits menos significativos se proporcionan tanto a las LUTs 1414, 1464 como a los elementos de retardo 1420, 1470. Las salidas de los elementos de retardo 1416 y 1418 se acoplan a las entradas de un multiplicador acoplado al elemento de retardo 1422. Las salidas de los elementos de retardo 1466, 1468 están acopladas a las entradas de una puerta multiplicadora acoplada al elemento de retardo 1472. La salida del elemento de retardo 1420 esta acoplada a una unidad inversora de bits 1424. La salida del elemento de retardo 1470 esta acoplada a una unidad inversora de bits 1474. Cada camino se proporciona finalmente a un elemento de retardo 1426, 1476, respectivamente.

De acuerdo con una forma de realización, las direcciones válidas se dividen en cuatro grupos. Se determina cada uno de los valores del contador resulta en una dirección válida, así como los valores de los contadores que se traducirán en una dirección no válida. El mapeo de los valores de contador de direcciones se almacena en la LUT

1404, 1454. Para cada LUT individual, cuando se incrementa el valor del contador a un valor correspondiente a una dirección no válida, la LUT entrega un valor de desplazamiento conveniente para establecer el siguiente valor de contador que corresponde a una dirección válida. De esta manera, el generador de direcciones sólo genera direcciones válidas. El proceso evita cálculos de direcciones innecesarias, es decir, el cálculo de direcciones no válidas que son posteriormente descartadas.

El circuito de generación de dirección de la Figura 15 puede aplicarse a una configuración de codificador tal como el codificador 1600 de la Figura 16. El codificador 1600 es sensible a una memoria externa que proporciona la trama a codificar. Se hacen cinco copias de los datos de entrada y se almacenan en cada una de las memorias 1622, 1624, 1626, 1628 y 1630, denominadas respectivamente MEM 1, MEM 2, MEM 3, MEM 4 y MEM 5. MEM 1 ofrece 4 bits secuenciales al codificador 1650. A las direcciones dentro de MEM 1 se accede secuencialmente. El codificador 1650 proporciona 4 salidas de bits para cada uno de X, Y_0 e Y_1.

Los generadores de direcciones 1632, 1634, 1636 Y 1638 están acoplados a MEM 2, MEM 3, MEM 4 y MEM 5, respectivamente. MEM 2, MEM 3, MEM 4 y MEM 5 proporcionan cada una un bit al codificador 1652. El codificador 1652 también proporciona salidas de 4 bits para cada uno de X, Y_0 e Y_1.

Los generadores de direcciones 1632, 1634, 1636 y 1638 producen series únicas de las direcciones de cada una de las memorias asociadas. Por ejemplo, en un esquema, el generador de direcciones 1632 produce las ubicaciones de dirección 0, 4, 8, etc.; el generador de direcciones 1634 produce las ubicaciones de dirección 1, 5, 9, etc.; el generador de direcciones 1636 produce las ubicaciones de dirección 2, 6, 10, etc.; la dirección 1638 produce las ubicaciones de dirección 3, 7, 11, etc. Cuando la dirección generada supera el tamaño de bloque del intercalador, el generador de direcciones salta esta dirección.

La Figura 17 ilustra un procedimiento 2000 para generar direcciones para intercalado tal como se utiliza en un codificador turbo paralelo. El procedimiento incrementa un contador en la etapa 2002. El valor del contador se utiliza para generar una dirección para intercalar datos de entrada en un codificador turbo, tales como la forma de realización ilustrada en la Figura 16. En el rombo de decisión 2004 se comprueba el valor del contador para determinar si el valor del contador se traducirá en una dirección no válida, Las direcciones no válidas están predeterminadas, en donde los valores de los contadores correspondientes a las direcciones no válidas se almacenan en un dispositivo de almacenamiento de memoria. Las direcciones no válidas son direcciones mayores que el tamaño del paquete de datos. Si el valor del contador se traduce en una dirección no válida, el proceso continúa a la etapa 2006 para ajustar el contador a una siguiente dirección válida. El procesamiento continua a la etapa 2008 para generar una dirección basándose en el valor del contador, ya sea originalmente generada o ajustada. Si el paquete esta completo el procesamiento termina en el rombo de decisión 2010. Si no, el procesamiento vuelve a la etapa 2002.

La presente invención proporciona un procedimiento de codificación de múltiples bits en paralelo, utilizando un procedimiento iterativo de procesar las diversas salidas. Durante cada ciclo de reloj, el codificador procesa múltiples bits y genera salidas consistentes con los que se generan secuencialmente en un codificador convolucional convencional. En una forma de realización, los datos de entrada se almacena en múltiples unidades de almacenamiento de memoria, que son entonces cada dirección única para proporcionar datos a los dos codificadores en paralelo, por ejemplo, que incorpora un codificador turbo.

Por lo tanto, se ha presentado un nuevo y mejorado procedimiento y aparato para la codificación de múltiples bits en paralelo, utilizando un procedimiento iterativo de procesar las diversas salidas. Las direcciones se generan para la operación de intercalado mediante el uso de múltiples dispositivos de almacenamiento de memoria, en donde se utiliza un contador para la generación de direcciones del intercalador y se proporciona un mapeo para identificar direcciones no válidas. Los expertos en la técnica entenderán que los datos, instrucciones, comandos, información, señales, bits, símbolos y chips que pueden referenciarse a lo largo de la descripción anterior se representan ventajosamente mediante voltajes, corrientes, ondas electromagnéticas. partículas o campos magnéticos, partículas o campos ópticos o cualquier combinación de los mismos. Los expertos apreciaran además que los diversos bloques lógicos ilustrativos, módulos, circuitos y etapas de algoritmo descritos en conexión con las realizaciones divulgadas en el presente documento pueden implementarse como hardware electrónico, software informático o combinaciones de ambos. Los diversos componentes ilustrativos, bloques, módulos, circuitos y etapas se han descrito generalmente en términos de su funcionalidad. Si la funcionalidad se implementa como hardware o software depende de la aplicación particular y las restricciones de diseño impuestas sobre el sistema global. Los expertos reconocen la intercambiabilidad de hardware y software bajo estas circunstancias y la mejor manera de implementar la funcionalidad descrita para cada aplicación particular. Como ejemplos, las etapas en los diversos bloques lógicos ilustrativos, módulos, circuitos y algoritmos descritos en conexión con las realizaciones divulgadas en el presente documento pueden implementarse o llevarse a cabo en un procesador digital de señal (DSP), un circuito integrado de aplicación específica (ASIC), un matriz de puertas programable (FPGA) u otro dispositivo lógico programable, puerta discreta o lógica de transistor, componentes hardware discretos tales como, por ejemplo, registros y FIFO, un procesador que ejecuta un conjunto de instrucciones firmware, cualquier módulo de software programable convencional, un procesador o cualquier combinación los mismos diseñada para llevar a cabo las realizaciones descritas en el presente documento. El procesador puede ser ventajosamente un microprocesador, pero como

alternativa. El procesador puede ser cualquier procesador convencional, controlador, microcontrolador, dispositivo lógico programable, matriz de elementos lógicos o máquina de estados. El módulo de software podría residir en memoria RAM, memoria flash, memoria ROM, memoria EPROM, memoria EEPROM, registros, disco duro, un disco extraíble, un CD-ROM o cualquier otra forma de medio de almacenamiento conocida en la técnica. Un procesador de ejemplo se acopla ventajosamente al medio de almacenamiento a fin de leer información de, y escribir información en, el medio de almacenamiento. En la alternativa, el medio de almacenamiento puede estar integrado en el procesador. El procesador y el medio de almacenamiento pueden residir en un ASIC. El ASIC puede residir en un teléfono u otro terminal de usuario. En la alternativa, el procesador y el medio de almacenamiento pueden residir en un teléfono u otro terminal de usuario. El procesador puede implementarse como una combinación de un DSP y un microprocesador o como dos microprocesadores junto con un núcleo DSP, etc. Se han mostrado y descrito, por lo tanto, las realizaciones preferidas de la presente invención. Seria evidente para un experto normal en la técnica, sin embargo, que las numerosas alteraciones pueden ser hechas a las realizaciones aquí descritas sin alejarse del espíritu o alcance de la invención. Por lo tanto, la presente invención no está limitada excepto de acuerdo con las siguientes reivindicaciones.

REIVINDICACIONES

1. Un procedimiento de codificación de datos con un codificador (1200) de un sistema de comunicación inalámbrica, en el que el codificador es síncrono, comprendiendo el procedimiento:

recibir una pluralidad de bits de información de entrada I[0]:I[3] en el codificador, en donde el procedimiento procesa la pluralidad de bits de información de entrada, I, y realiza los siguientes pasos durante un solo ciclo de reloj:

- calcular los valores de estado $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$, para el codificador, en base a la pluralidad de bits de información de entrada I[0]:I[3] y los valores de estado iniciales o almacenados $S_0[0]$ y $S_1[0]$; y

- generar un conjunto de valores codificados de salida $X[0]:X[3]$, $Y_0[0]:Y_0[3]$, $Y_1[0]:Y_1[3]$ usando los valores de estado calculados $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$, y la pluralidad de bits de información de entrada I[0]:I[3] y los valores de estado iniciales o almacenados $S_0[0]$ y $S_1[0]$ aplicando de forma recursiva:

$$X[n] = I[n];$$

$$Y_0[n] = I[n] \oplus S_1[n] \oplus S_0[n]; \text{ y}$$

$$Y_1[n] = I[n] \oplus S_0[n];$$

en las que I[n] representa un elemento n-ésimo de bits de dicha pluralidad de bits de información de entrada I[0]:I[3], X[n] representa una salida del codificador igual al n-ésimo bit de información de entrada;

$S_0[n]$ y $S_1[n]$ representan elementos n-ésimos de bit de un primer y un segundo valor de estado de los valores de estado $S_0[0]:S_03]$, $S_1[0]:S_1[3]$,

$Y_0[n]$ representa el primer elemento de bit de paridad enésimo de dicho conjunto de valores codificados de salida $X[0]:X[3]$, $Y_0[0]: Y_0[3]$, $Y_1[0]:Y_1[3]$, $Y_1[n]$ representa el segundo elemento de bit de paridad enésimo de dicho conjunto de valores codificados de salida $X[0]:X[3]$, $Y_0[0]:Y_0[3]$, $Y_1[0]:Y_1[3]$, $\oplus$ representa la operación lógica digital XOR y n representa un índice de iteración de la operación recursiva,

en el que dicha etapa de cálculo de dichos valores de estado $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$ utiliza:

$$S_0[n + 1] = I[n] \oplus S_1[n] \oplus S_2[n];$$

$$S_1[n + 1] = S_0[n]; \text{ y}$$

$$S_2[n + 1] = S_1[n],$$

en las que $S_2[n]$ representa un elemento enésimo de bit de un tercer valor de estado $S_2[0]:S_2[3]$.

2. El procedimiento según la reivindicación 1, en el que el conjunto de valores de salida codificados comprende:

I,

Y_0, y

Y_1,

en donde Y_0 representa una primera salida de bit de paridad de múltiples bits e Y_1 representa una segunda salida de bit de paridad de múltiples bits.

3. El procedimiento según la reivindicación 1, que comprende además:

almacenar los valores de estado $S_0[4]$, $S_1[4]$, $S_2[4]$ en un dispositivo de almacenamiento de memoria.

4. Un procedimiento de codificación turbo que comprende primera y segunda codificación según el procedimiento según la reivindicación 1,

en el que el procedimiento comprende además antes de la segunda codificación:

intercalar bits de información de entrada de un paquete de datos, en donde el intercalado comprende además producir el equivalente funcional de: escribir los bits de información de entrada como una matriz en una secuencia de direcciones y generar otra secuencia de direcciones de salida para la lectura de los bits de información de entrada del paquete de datos.

5. El procedimiento según la reivindicación 4, en el que el intercalado comprende además:

incrementar (2002) un contador, en donde el valor del contador se utiliza para generar una dirección para intercalar datos de entrada;

determinar (2004) si el valor del contador se traducirá en una dirección no válida que es mayor que el paquete de datos, en donde las direcciones no válidas se almacenan en un dispositivo de almacenamiento de memoria, si el valor del contador se traduce en una dirección no válida,

ajustar (2006) el contador a una siguiente dirección válida, y

generar (2008) una dirección en base al valor del contador.

6. El procedimiento según la reivindicación 5, en el que un mapeo de los valores de contador de direcciones válidas se almacena en una tabla de consulta (LUT).

7. El procedimiento según la reivindicación 4, en el que el intercalado comprende además:

recibir un valor de contador de un contador;

determinar información de bit más significativo (MSB) a partir del valor del contador,

incrementar la información MSB;

determinar información de bit menos significativo (LSB) a partir del valor del contador;

utilizar la información LSB para direccionar una tabla de consulta (LUT), almacenando la LUT un mapeo de los valores de contador y direcciones, y

multiplicar la información MSB incrementada y un valor almacenado en la ubicación de la LUT direccionada por la información LSB para generar un producto;

invertir los bits de la información LSB;

generar una dirección de salida en base a la información LSB Invertida y el producto.

8. El procedimiento según las reivindicaciones 6 ó 7, que comprende además:

determinar si la dirección de salida generada es válida, y

si la dirección de salida generada no es válida, desechar la dirección de salida generada.

9. Un aparato codificador (1200) de un sistema de comunicación inalámbrica, en el que el aparato codificador es síncrono, que comprende:

medios para recibir una pluralidad de bits de información de entrada I[0]:I[3] en el codificador, en donde el aparato codificador procesa la pluralidad de bits de información de entrada, I, y realiza los siguientes pasos durante un solo ciclo de reloj:

medios para calcular los valores de estado $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$, para el codificador, en base a la pluralidad de bits de información de entrada I[0]:I[3] y los valores de estado iniciales o almacenados $S_0[0]$ y $S_1[0]$; y

medios para generar un conjunto de valores codificados de salida $X[0]:X[3]$, $Y_0[0]:Y_0[3]$, $Y_1[0]:Y_1[3]$ usando los valores de estado calculados $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$ y la pluralidad de bits de información de entrada I[0]:I[3] y los valores de estado iniciales o almacenados $S_0[0]$ y $S_1[0]$ aplicando de forma recursiva:

$$X[n] = I[n];$$

$Y_0[n] = I[n] \oplus S_1[n] \oplus S_0[n]$; y

$Y_1[n] = I[n] \oplus S_0[n]$;

en las que I[n] representa un elemento n-ésimo de bits de dicha pluralidad de bits de información de entrada I[0]:I[3], X[n] representa una salida del codificador igual al n-ésimo bit de información de entrada;

$S_0[n]$ y $S_1[n]$ representan elementos n-ésimos de bit de un primer y un segundo valor de estado de los valores de estado $S_0[0]:S_03]$, $S_1[0]:S_1[3]$, $S_2[0]:S_2[3]$

$Y_0[n]$ representa el primer elemento de bit de paridad enésimo de dicho conjunto de valores codificados de salida X[0]:X[3], $Y_0[0]$: $Y_0[3]$, $Y_1[0]:Y_1[3]$, $Y_1[n]$ representa el segundo elemento de bit de paridad enésimo de dicho conjunto de valores codificados de salida X[0]:X[3], $Y_0[0]:Y_0[3]$, $Y_1[0]:Y_1[3]$, $\oplus$ representa la operación lógica digital XOR y n representa un índice de iteración de la operación recursiva,

en el que dichos valores de estado $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$ se calculan utilizando:

$S_0[n + 1] = I[n] \oplus S_1[n] \oplus S_2[n]$;

$S_1[n + 1] = S_0[n]$; y

$S_2[n + 1] = S_1[n]$,

en las que $S_2[n]$ representa un elemento enésimo de bit de un tercer valor de estado $S_2[1]:S_2[3]$;

en el que los medios para calcular un conjunto de valores de estado $S_0[1]:S_0[3]$, $S_1[1]:S_1[3]$, $S_2[1]:S_2[3]$ y los medios para generar un conjunto de valores de salida codificados X[0]:X[3], $Y_0[0]:Y_0[3]$, $Y_1[0]:Y_1[3]$ están habilitados durante un único ciclo de reloj.

10. El aparato según la reivindicación 9, en el que el conjunto de valores de salida codificados comprende:

I,

Y_0, y

Y_1,

en donde Y_0 representa una primera salida de bit de paridad de múltiples bits e Y_1 representa una segunda salida de bit de paridad de múltiples bits.

11. El aparato según la reivindicación 9, que comprende además:

medios para almacenar los valores de estado $S_0[4]$, $S_1[4]$ y $S_2[4]$.

12. Un codificador turbo que comprende un primer y un segundo aparato codificador según la reivindicación 9 que comprende además

un intercalador anterior al segundo codificador, adaptado para intercalar los bits de información de entrada de un paquete de datos, produciendo el intercalador el equivalente funcional de: escribir los bits de información de entrada como una matriz en una secuencia de direcciones, y generar otra secuencia de direcciones de salida para leer los bits de información de entrada del paquete de datos.

13. El codificador turbo según la reivindicación 12, en el que el intercalador comprende además:

medios para incrementar (2002) un contador, en donde el valor del contador se utiliza para generar una dirección para intercalar datos de entrada;

medios para determinar (2004) si el valor del contador se traducirá en una dirección no válida que es mayor que el paquete de datos, en el que las direcciones no válidas se almacenan en un dispositivo de almacenamiento de memoria; si el valor del contador se traduce en una dirección no válida, ajustar (2006) el contador a una siguiente dirección válida, y

medios para generar una dirección basándose en el valor del contador.

14. El codificador turbo según la reivindicación 13, en el que un mapeo de los valores de contador a las direcciones válidas se almacena en una tabla de consulta (LUT).

15. El codificador turbo según la reivindicación 14, en el que el intercalador comprende además:

medias para recibir un valor de contador de un contador;

medios para determinar información de bit más significativo (MSB) a partir del valor del contador;

medios para incrementar la información MSB;

medios para determinar información de bit menos significativo (LSB) a partir del valor del contador,

medios para utilizar la información LSB para direccionar una tabla de consulta (LUT), almacenando la LUT un mapeo de los valores de contador y direcciones, y

medios para multiplicar la información MSB incrementada y un valor almacenado en la ubicación de la LUT direccionada por la información LSB para generar un producto;

medios para invertir los bits de la información LSB;

medios para generar una dirección de salida en base a la información LSB invertida y el producto.

16. El aparato según las reivindicaciones 14 ó 15, que comprende además:

medios para determinar si la dirección de salida generada es válida; y

medios para desechar la dirección de salida generada si la dirección de salida generada no es válida.

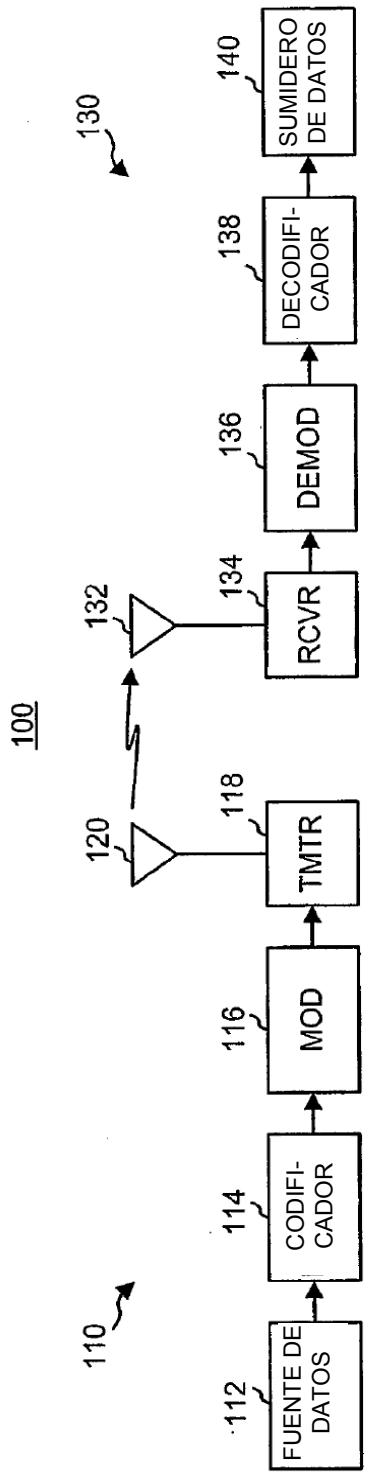
100
110
112
FUENTE DE DATOS
114
CODIFI-CADOR
116
MOD
118
TMTR
120
132
134
RCVR
136
DEMOD
138
DECODIFI-CADOR
130
140
SUMIDERO DE DATOS

FIG. 1

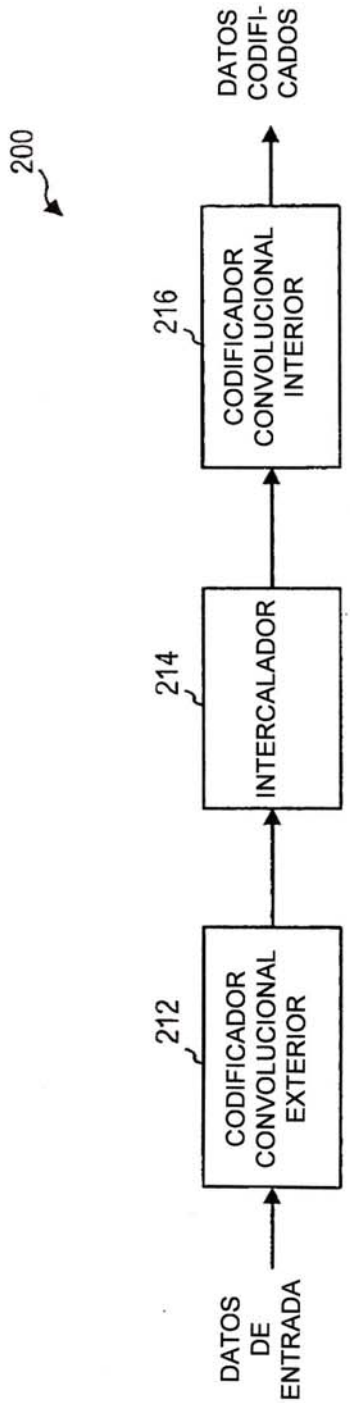
200
DATOS DE ENTRADA
212
CODIFICADOR CONVOLUCIONAL EXTERIOR
214
INTERCALADOR
216
CODIFICADOR CONVOLUCIONAL INTERIOR
DATOS CODIFI-CADOS

FIG. 2

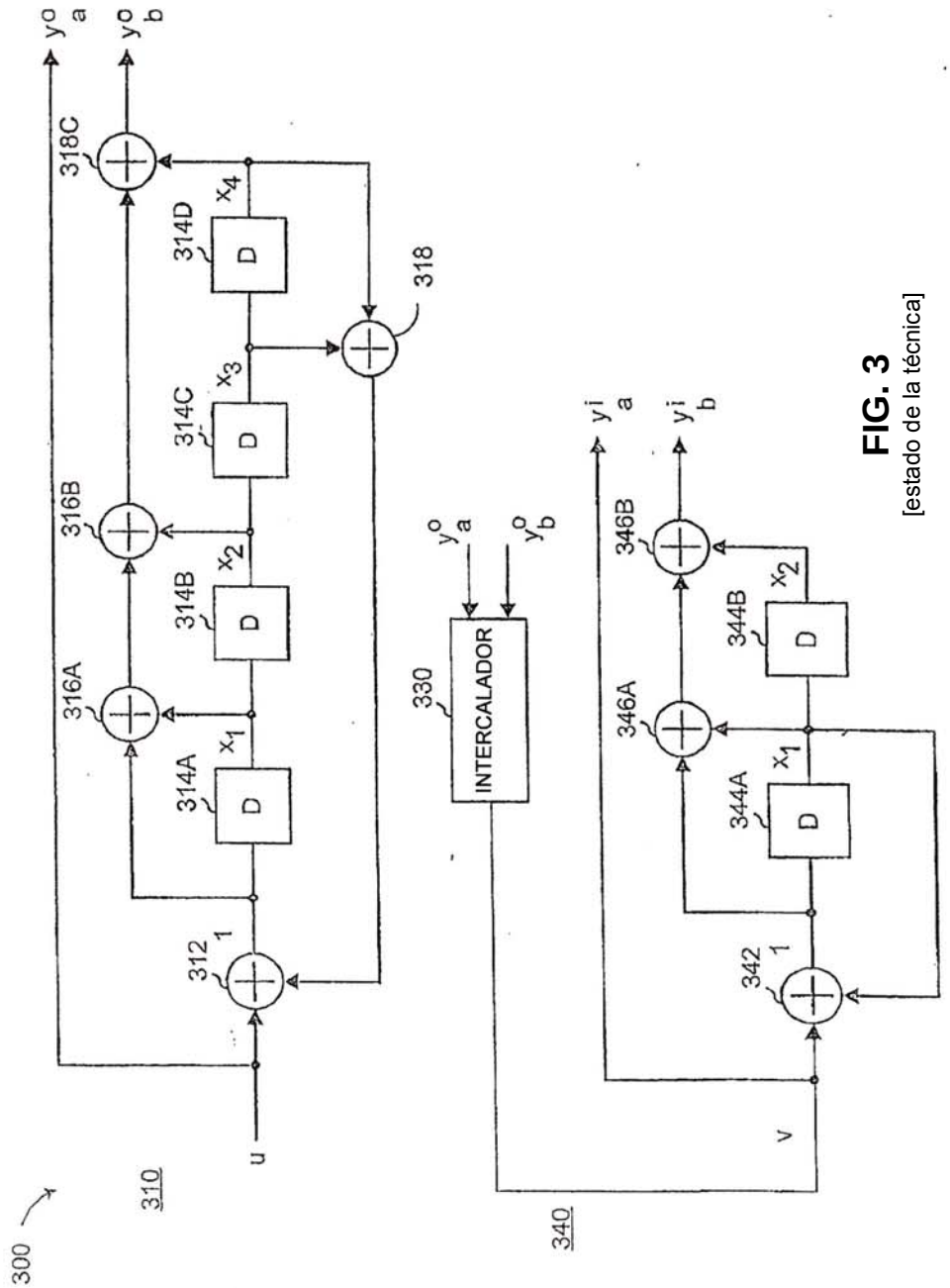

FIG. 3
[estado de la técnica]

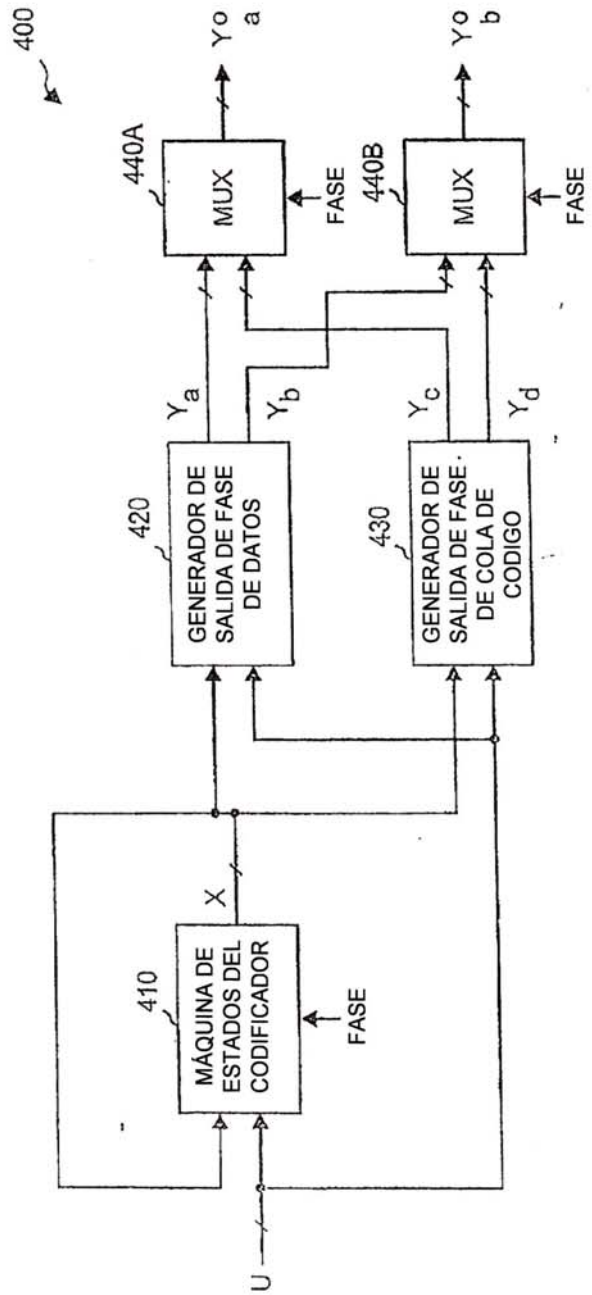

FIG. 4
[estado de la técnica]

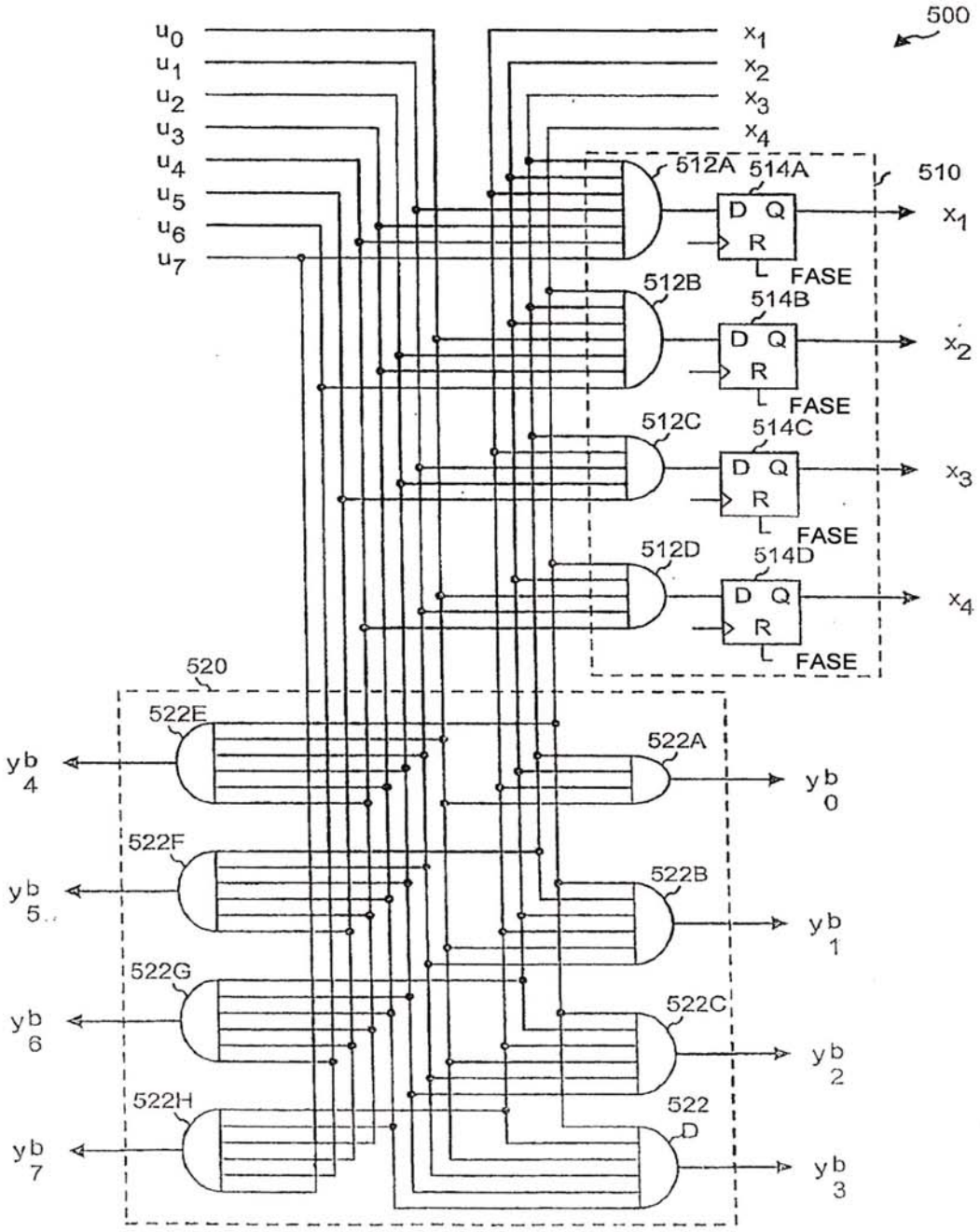

FIG. 5A
[estado de la técnica]

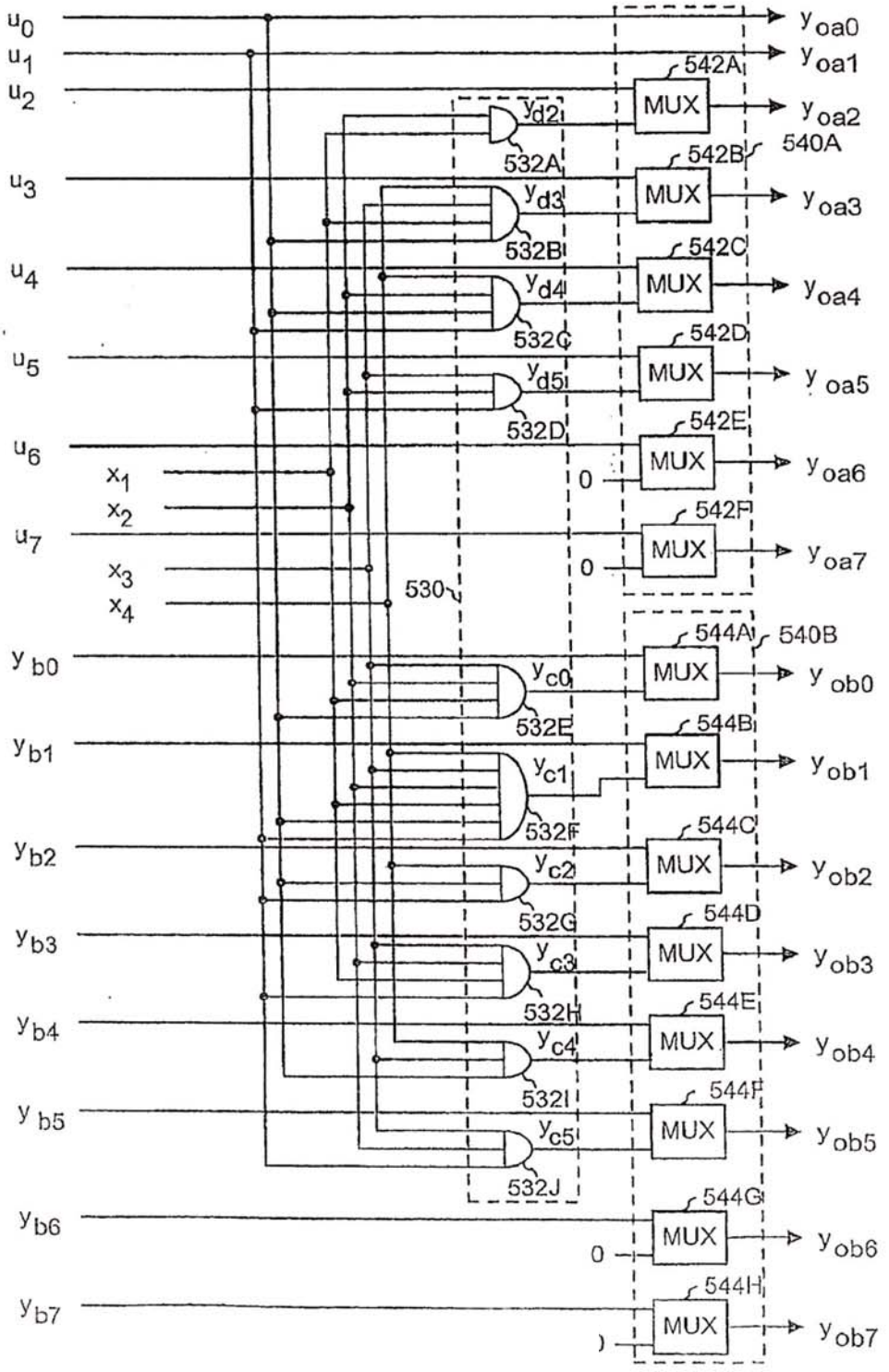

FIG. 5B
[estado de la técnica]

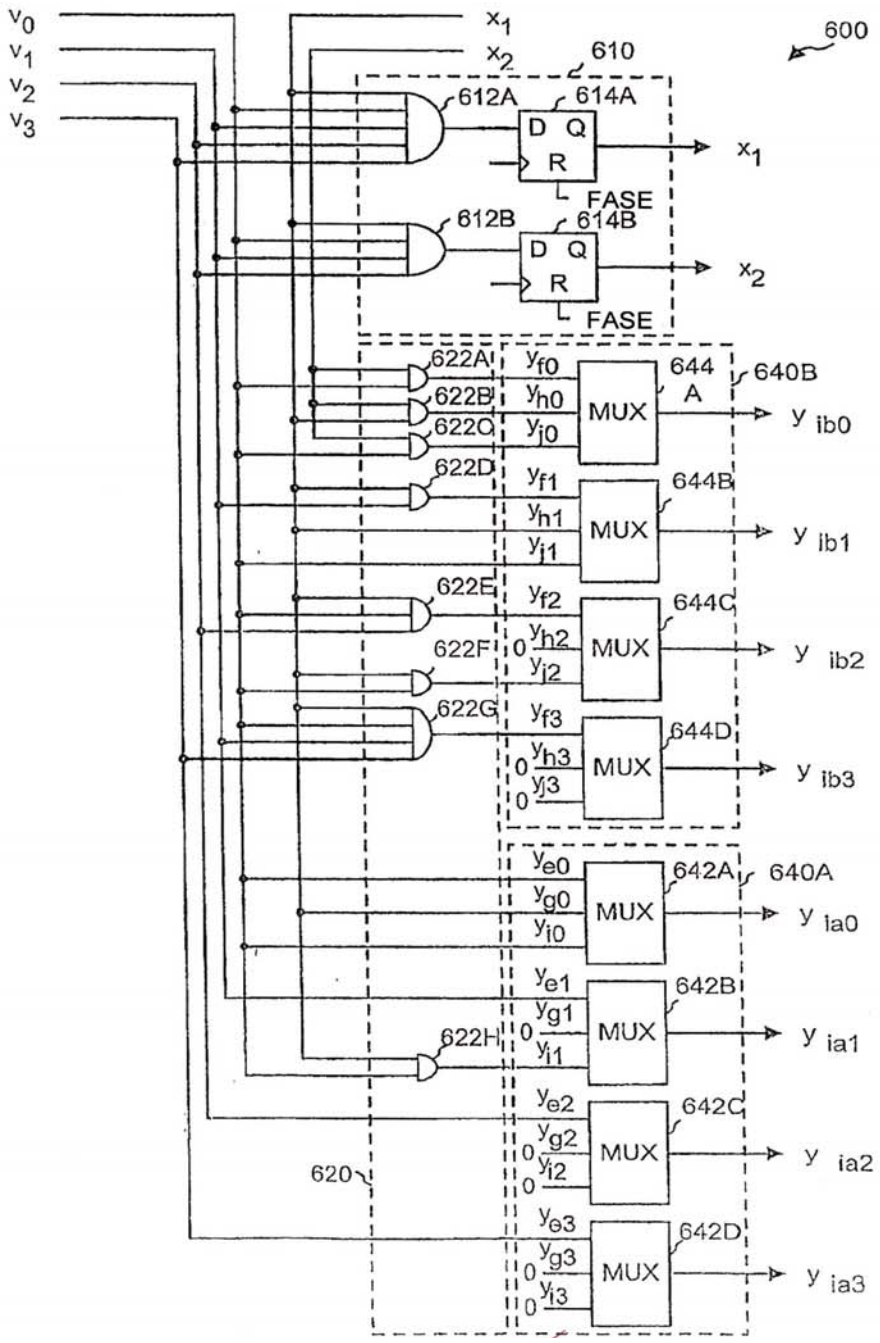

FIG. 6
[estado de la técnica]

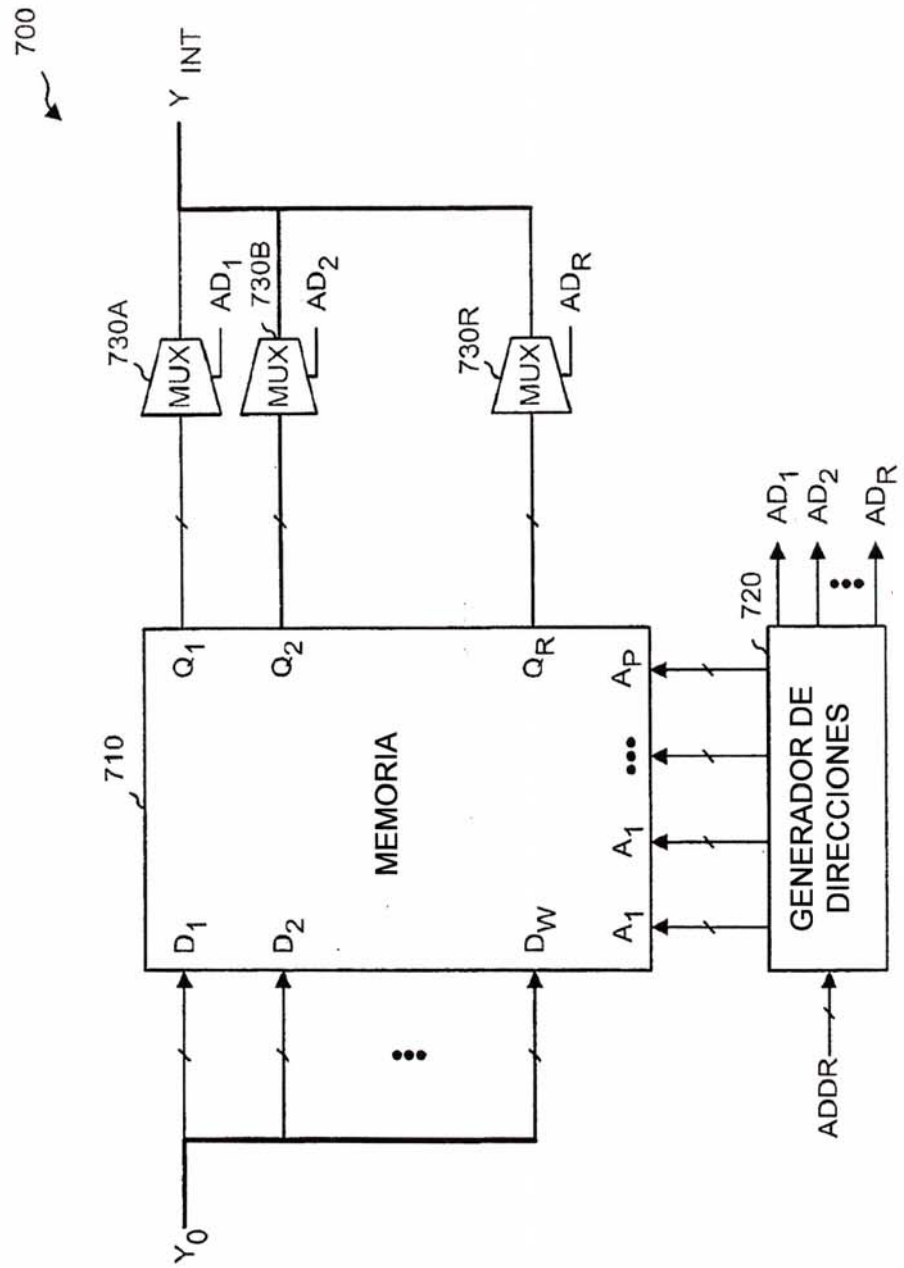
700
Y INT
730A
MUX
AD1
730B
MUX
AD2
730R
MUX
ADR
710
MEMORIA
Q1
Q2
QR
AP
A1
A1
D1
D2
DW
720
GENERADOR DE DIRECCIONES
AD1
AD2
ADR
ADDR
Y0

FIG. 7A

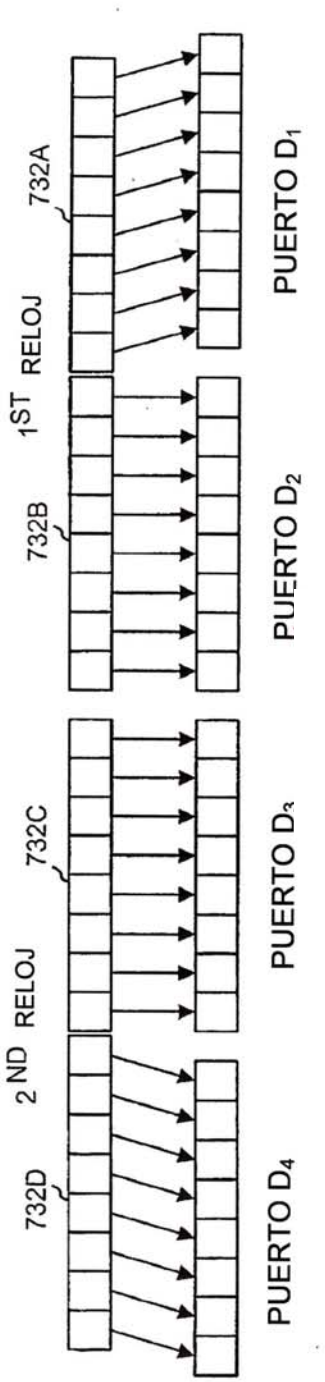
732A
1ST RELOJ
732B
PUERTO D1
PUERTO D2
732C
2ND RELOJ
732D
PUERTO D3
PUERTO D4

FIG. 7B

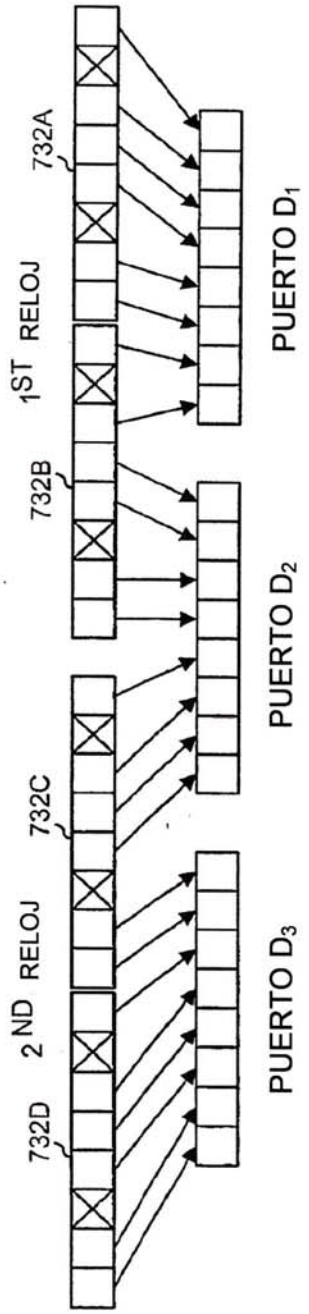
732A
1ST RELOJ
732B
PUERTO D1
PUERTO D2
732C
2ND RELOJ
732D
PUERTO D3

FIG. 7C

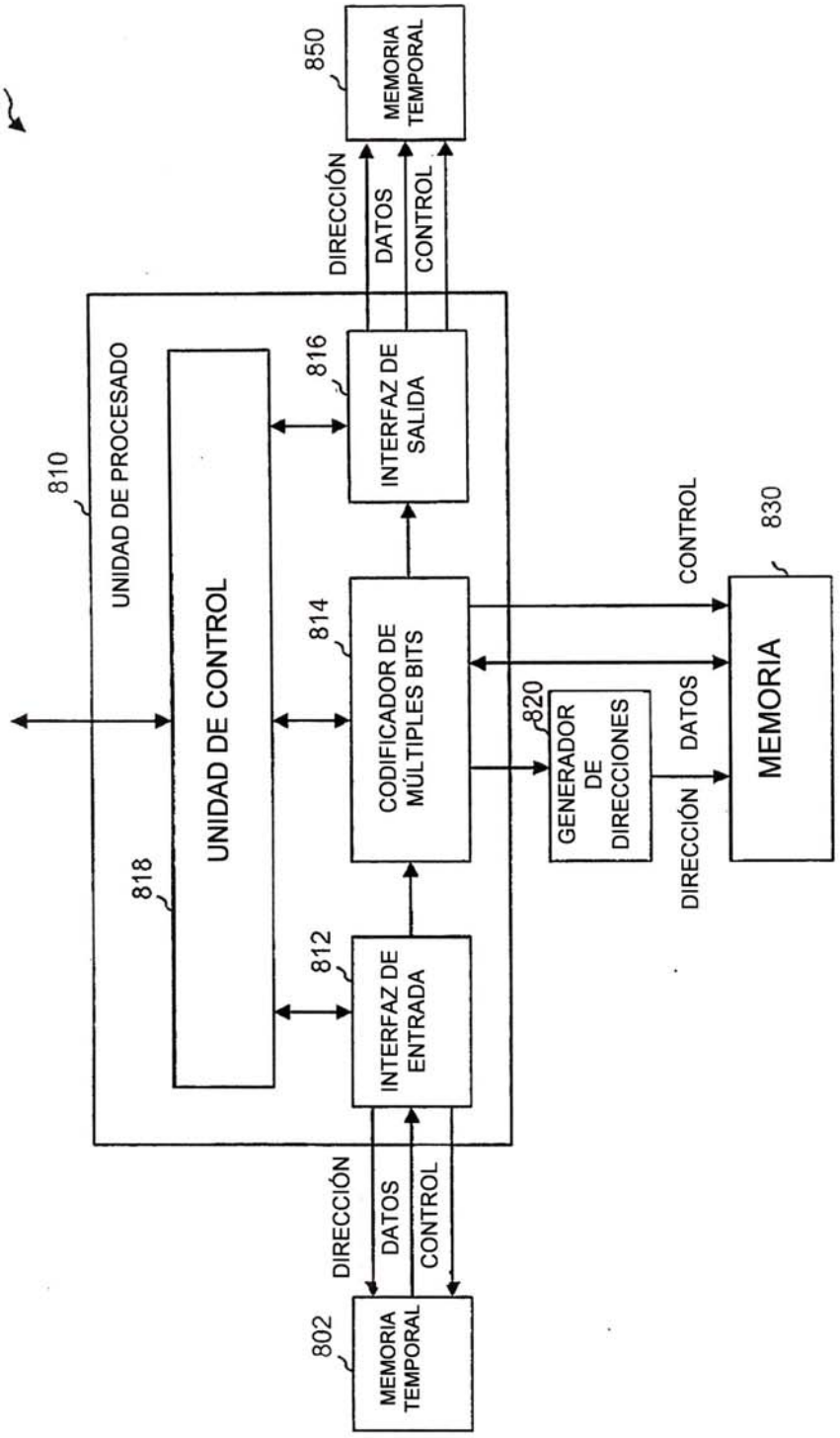
800
850
MEMORIA TEMPORAL
DIRECCIÓN
DATOS
CONTROL
810
UNIDAD DE PROCESADO
816
INTERFAZ DE SALIDA
UNIDAD DE CONTROL
814
CODIFICADOR DE MÚLTIPLES BITS
820
GENERADOR DE DIRECCIONES
CONTROL
830
MEMORIA
DATOS
DIRECCIÓN
818
812
INTERFAZ DE ENTRADA
DIRECCIÓN
DATOS
CONTROL
802
MEMORIA TEMPORAL

FIG. 8

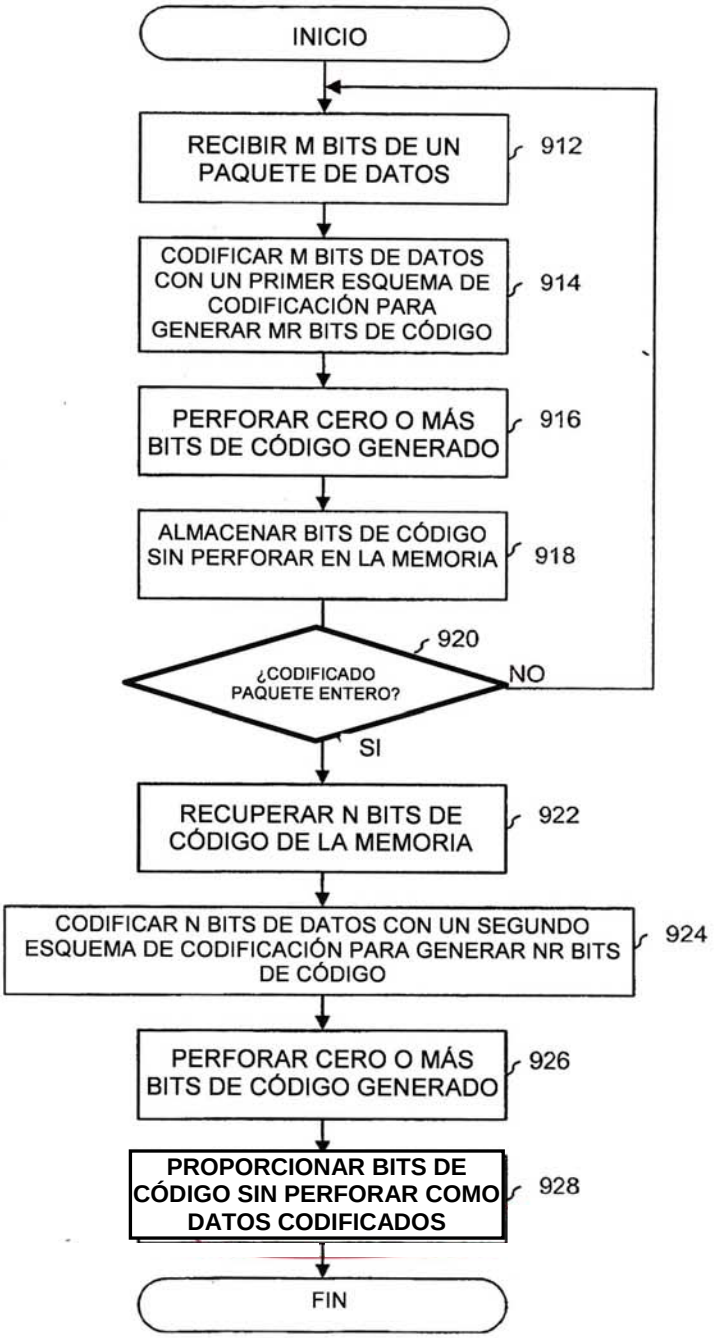
INICIO
RECIBIR M BITS DE UN PAQUETE DE DATOS
912
CODIFICAR M BITS DE DATOS CON UN PRIMER ESQUEMA DE CODIFICACIÓN PARA GENERAR MR BITS DE CÓDIGO
914
PERFORAR CERO O MÁS BITS DE CÓDIGO GENERADO
916
ALMACENAR BITS DE CÓDIGO SIN PERFORAR EN LA MEMORIA
918
920
¿CODIFICADO PAQUETE ENTERO?
NO
SI
RECUPERAR N BITS DE CÓDIGO DE LA MEMORIA
922
CODIFICAR N BITS DE DATOS CON UN SEGUNDO ESQUEMA DE CODIFICACIÓN PARA GENERAR NR BITS DE CÓDIGO
924
PERFORAR CERO O MÁS BITS DE CÓDIGO GENERADO
926
PROPORCIONAR BITS DE CÓDIGO SIN PERFORAR COMO DATOS CODIFICADOS
928
FIN

FIG. 9

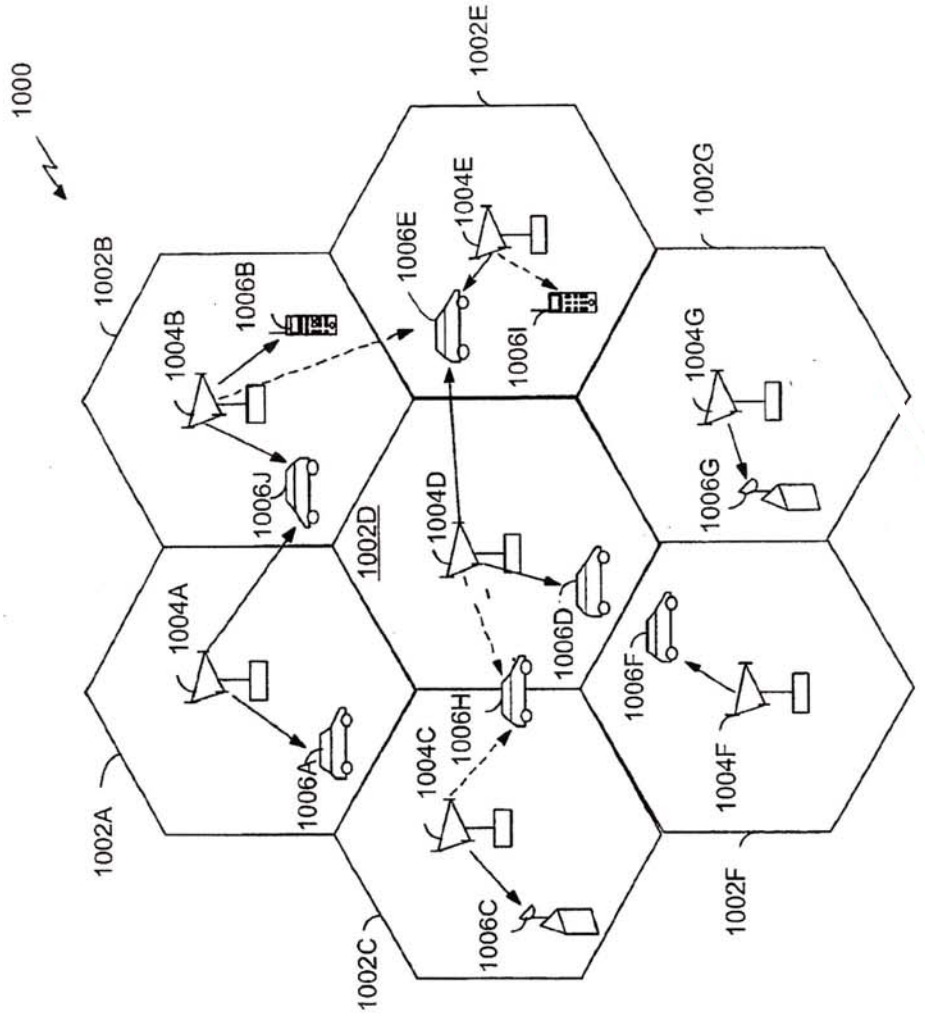
1000
1002A
1002B
1002C
1002D
1002E
1002F
1002G
1004A
1004B
1004C
1004D
1004E
1004F
1004G
1006A
1006B
1006C
1006D
1006E
1006F
1006G
1006H
1006I
1006J

FIG. 10

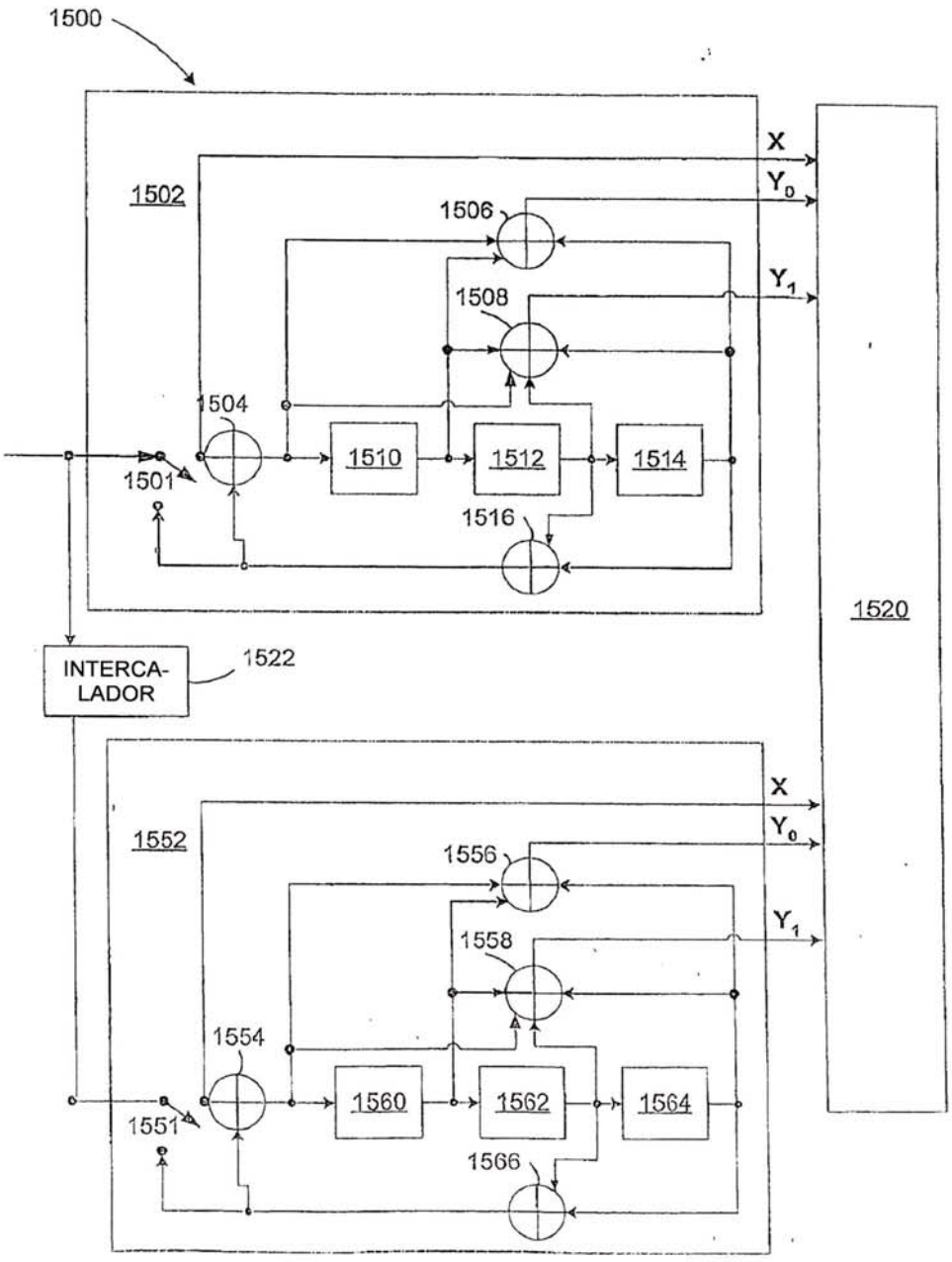

FIG. 11
[estado de la técnica]

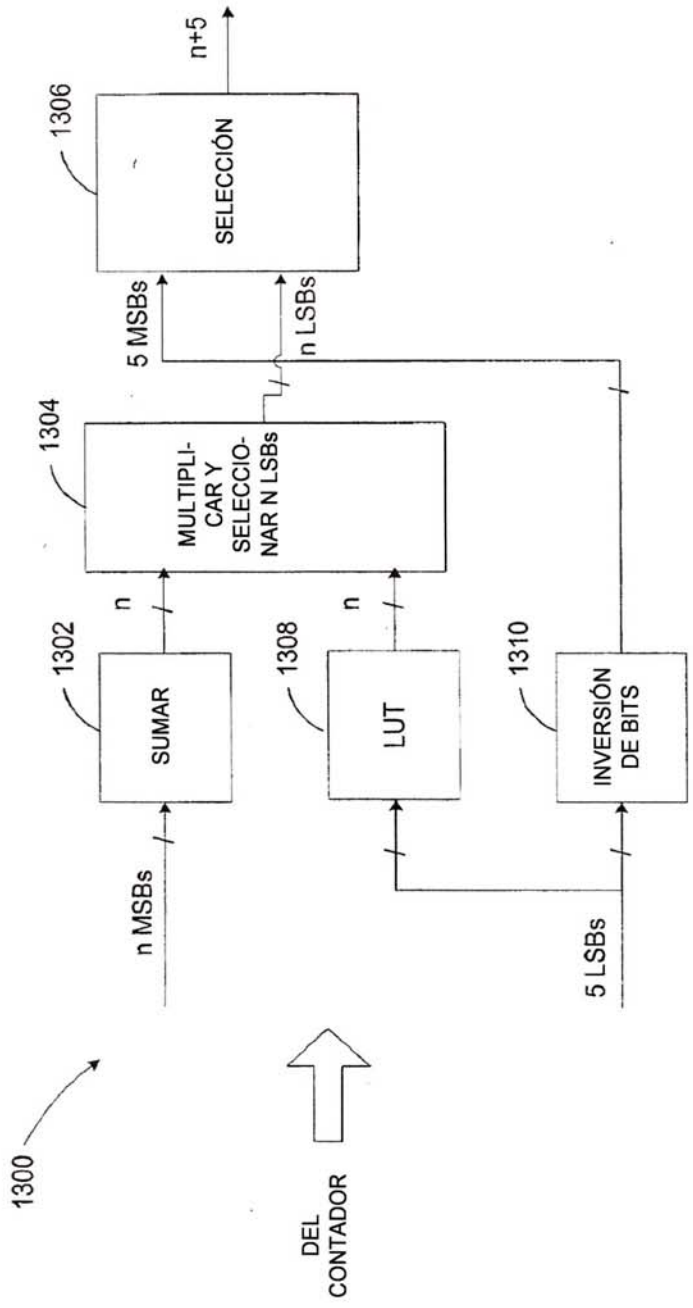
1300
DEL CONTADOR
n MSBs
5 LSBs
1302
SUMAR
1308
LUT
1310
INVERSIÓN DE BITS
n
n
1304
MULTIPLI-CAR Y SELECCIO-NAR N LSBs
5 MSBs
n LSBs
1306
SELECCIÓN
n+5

FIG. 12

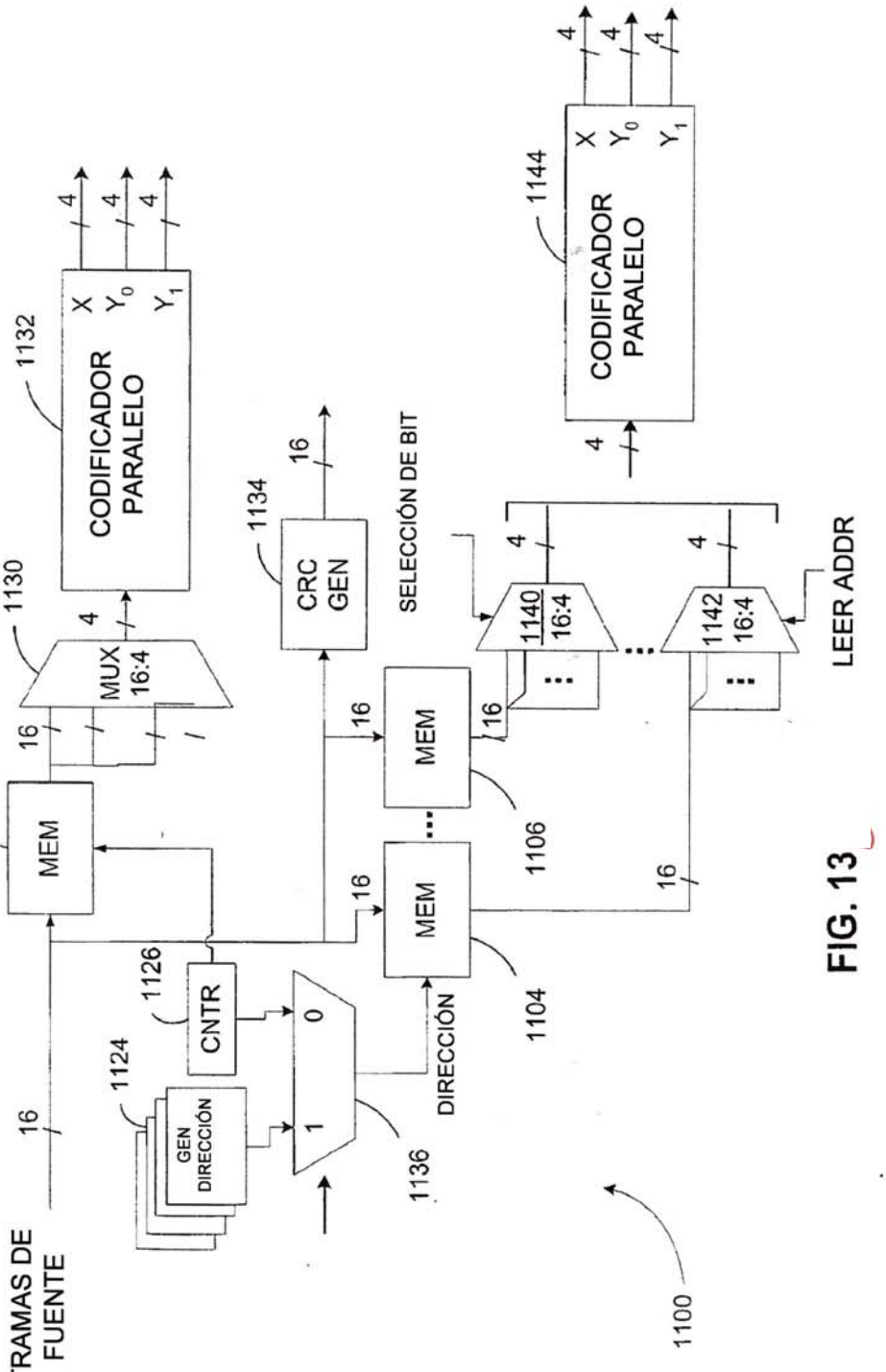
TRAMAS DE FUENTE
16
1128
MEM
1124
GEN DIRECCIÓN
1126
CNTR
1136
0
1
DIRECCIÓN
1104
MEM
1106
MEM
16
1130
MUX 16:4
4
1132
CODIFICADOR PARALELO
X
Y0
Y1
1134
CRC GEN
SELECCIÓN DE BIT
1140
16:4
1142
16:4
1144
CODIFICADOR PARALELO
LEER ADDR
1100

FIG. 13

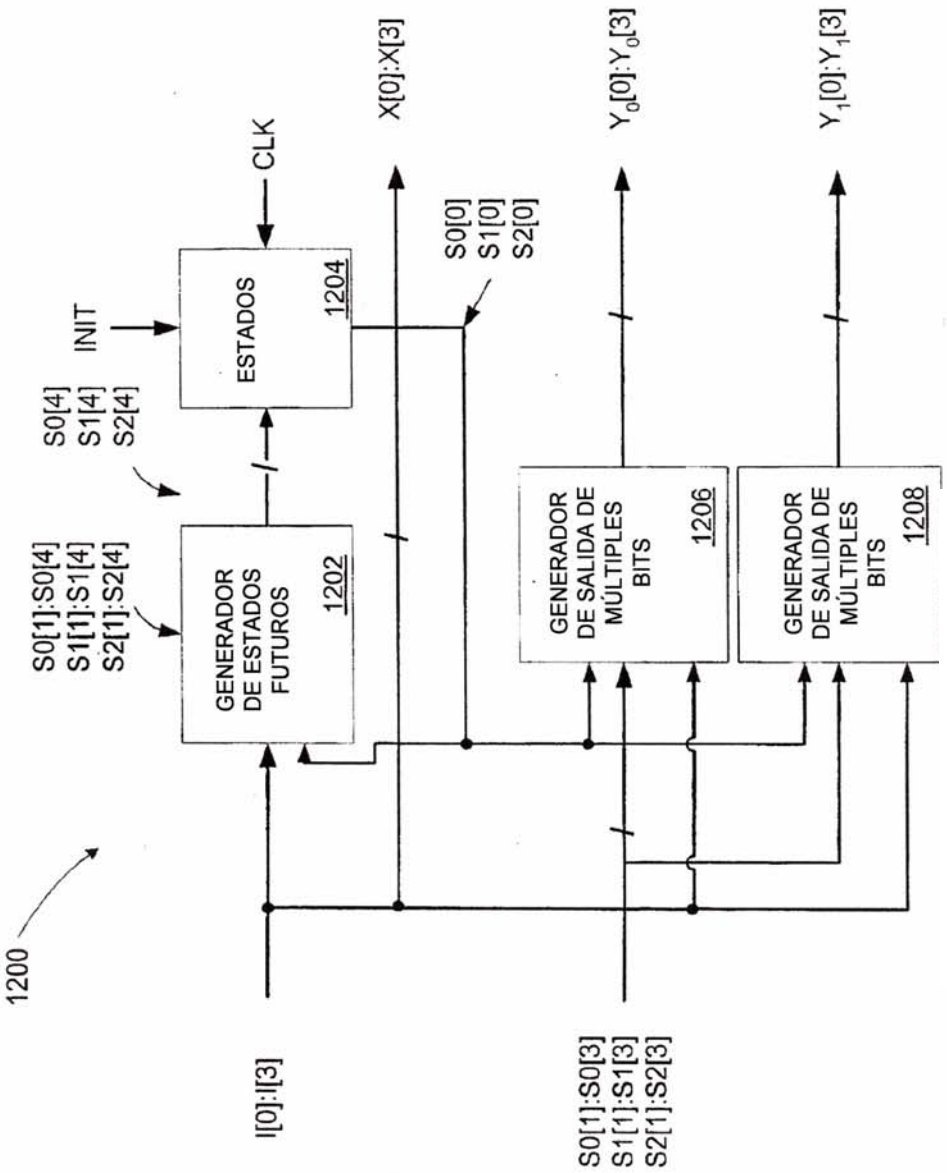
1200
I[0]:I[3]
S0[1]:S0[4]
S1[1]:S1[4]
S2[1]:S2[4]
GENERADOR DE ESTADOS FUTUROS
1202
S0[4]
S1[4]
S2[4]
INIT
CLK
ESTADOS
1204
X[0]:X[3]
S0[0]
S1[0]
S2[0]
S0[1]:S0[3]
S1[1]:S1[3]
S2[1]:S2[3]
GENERADOR DE SALIDA DE MÚLTIPLES BITS
1206
$Y_0[0]:Y_0[3]$
GENERADOR DE SALIDA DE MÚLTIPLES BITS
1208
$Y_1[0]:Y_1[3]$

FIG. 14

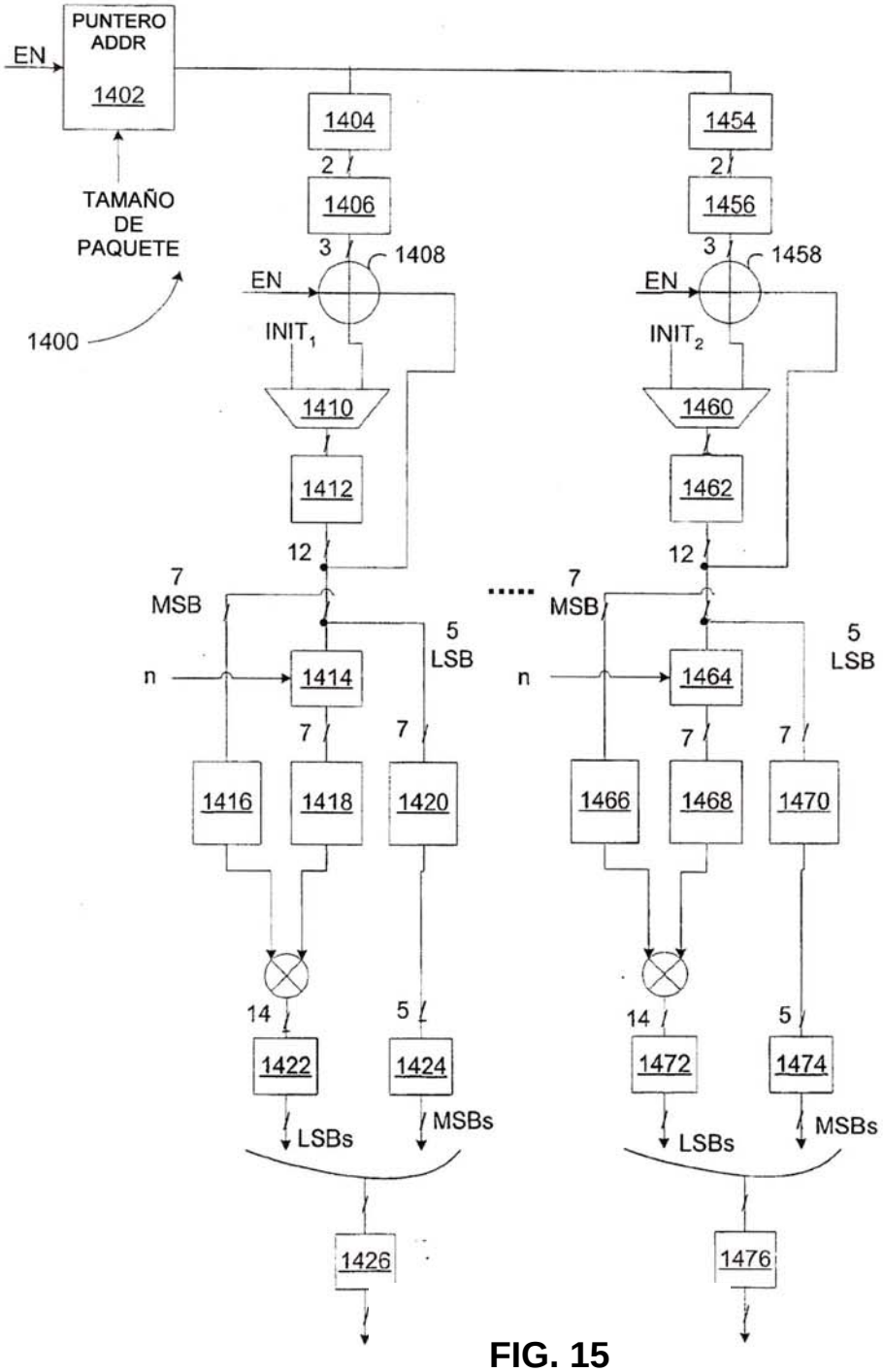
PUNTERO ADDR
1402
EN
TAMAÑO DE PAQUETE
1400
1404
2
1406
3
1408
EN
INIT1
1410
1412
12
7 MSB
5 LSB
n
1414
7
7
1416
1418
1420
14
5
1422
1424
LSBs
MSBs
1426
1454
2
1456
3
1458
EN
INIT2
1460
1462
12
7 MSB
5 LSB
n
1464
7
7
1466
1468
1470
14
5
1472
1474
LSBs
MSBs
1476

FIG. 15

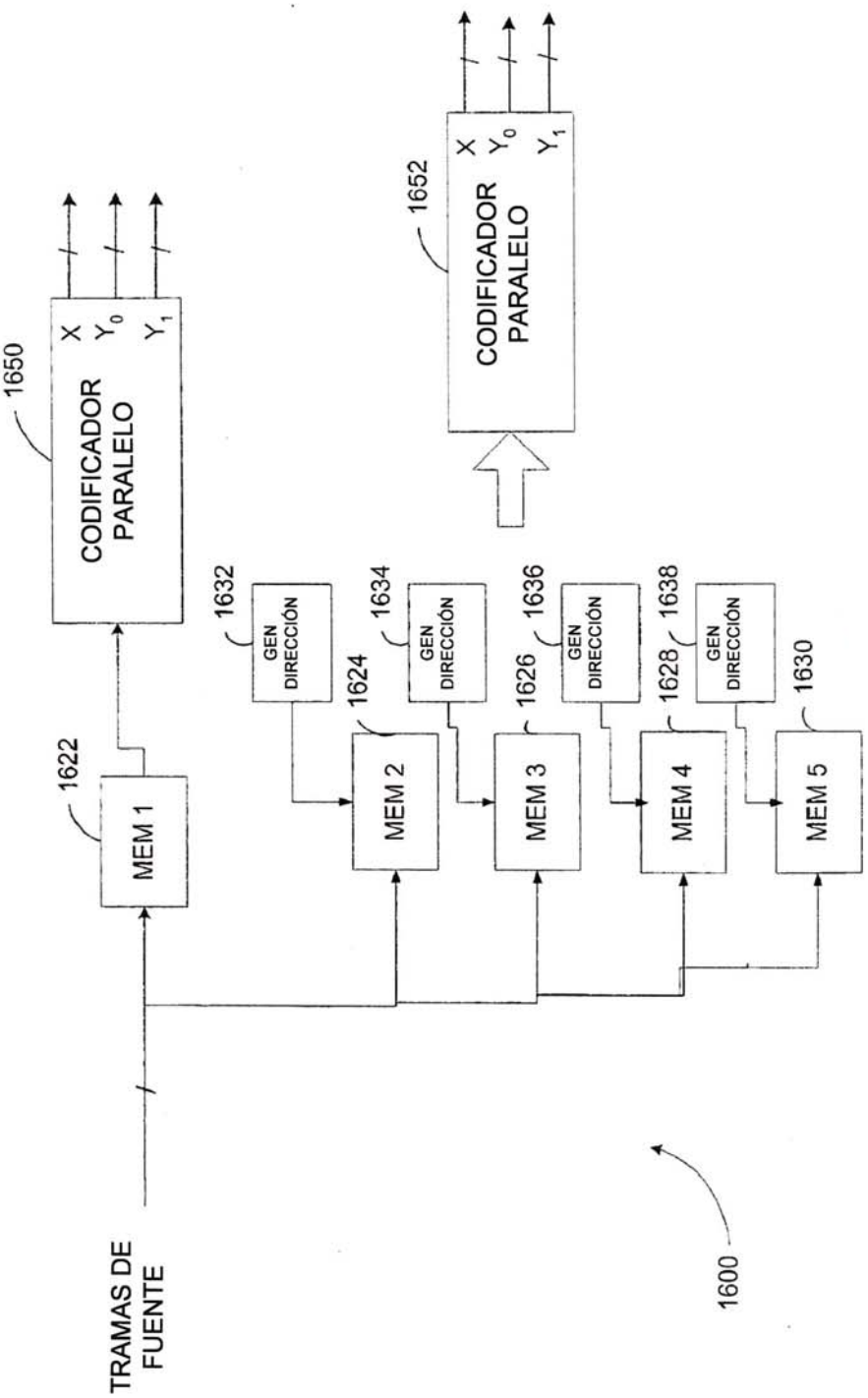
TRAMAS DE FUENTE
1600
MEM 1
1622
MEM 2
1624
MEM 3
1626
MEM 4
1628
MEM 5
1630
GEN DIRECCIÓN
1632
GEN DIRECCIÓN
1634
GEN DIRECCIÓN
1636
GEN DIRECCIÓN
1638
CODIFICADOR PARALELO
1650
X
Y0
Y1
CODIFICADOR PARALELO
1652
X
Y0
Y1

FIG. 16

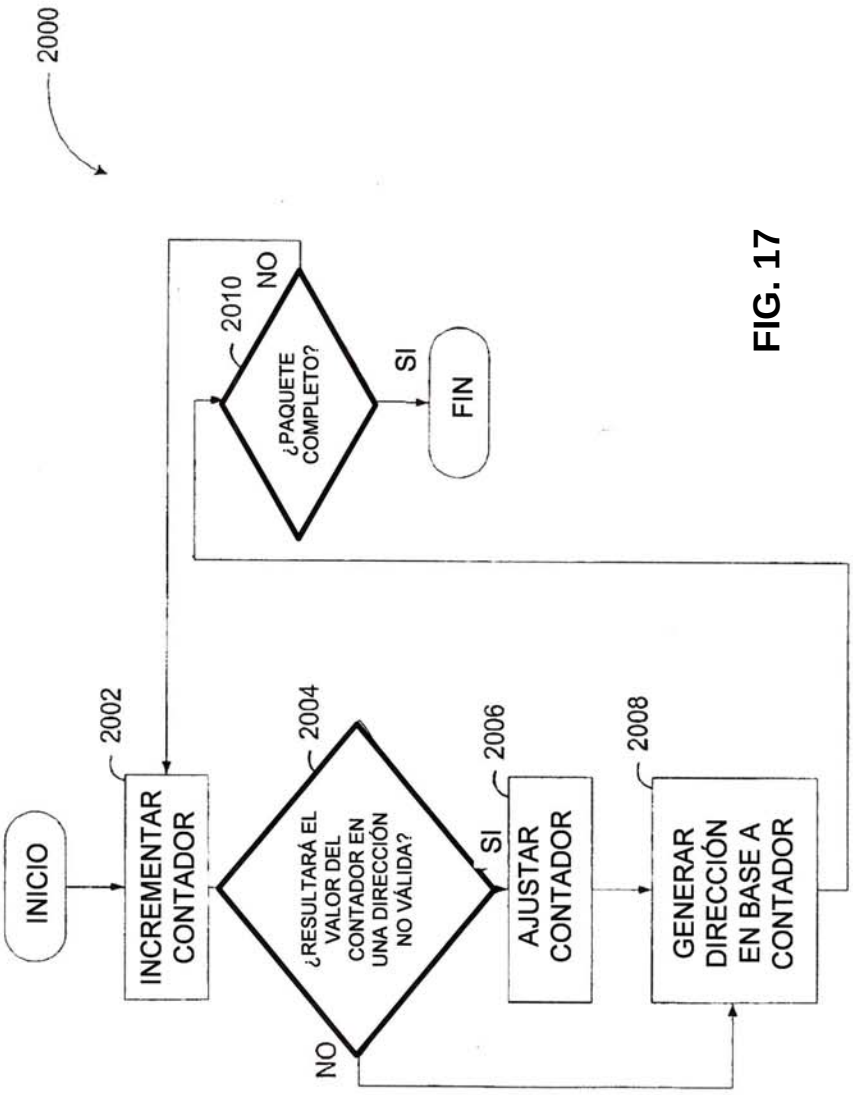
2000
INICIO
2002
INCREMENTAR CONTADOR
2004
¿RESULTARÁ EL VALOR DEL CONTADOR EN UNA DIRECCIÓN NO VÁLIDA?
SI
NO
2006
AJUSTAR CONTADOR
2008
GENERAR DIRECCIÓN EN BASE A CONTADOR
2010
¿PAQUETE COMPLETO?
NO
SI
FIN

FIG. 17