(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 610 818**

(51) Int. CI.:

H04L 1/18 (2006.01)
H04L 1/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **17.06.2008 PCT/US2008/067175**

(87) Fecha y número de publicación internacional: **24.12.2008 WO08157523**

(96) Fecha de presentación y número de la solicitud europea: **17.06.2008 E 08771231 (1)**

(97) Fecha y número de publicación de la concesión europea: **19.10.2016 EP 2171908**

(54) Título: **Un aparato que comprende un almacenador temporal circular y método para asignar versiones de redundancia a un almacenador temporal circular**

(30) Prioridad:

20.06.2007 US 765921

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
03.05.2017

(73) Titular/es:

GOOGLE TECHNOLOGY HOLDINGS LLC (100.0%)
1600 Amphitheatre Parkway
Mountain View, CA 94043, US

(72) Inventor/es:

BLANKENSHIP, YUFEI W.;
BLANKENSHIP, T. KEITH;
CLASSON, BRIAN K. y
NIMBALKER, AJIT

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCIÓN

Un aparato que comprende un almacenador temporal circular y método para asignar versiones de redundancia a un almacenador temporal circular

Campo de la invención

La presente invención se refiere de manera general a sistemas de comunicación y, en particular, a un método y aparato para asignar versiones de redundancia a un almacenador temporal circular dentro de un sistema de comunicación.

Antecedentes de la invención

En un sistema de comunicación, se usan técnicas de control de error para proteger una señal contra deterioros durante la transmisión sobre un canal. Como parte de tales técnicas de control de error, se produce una palabra de código para cada bloque de información. A fin de adaptar una tasa de transmisión por el aire, la palabra de código se redimensiona a través de un algoritmo de adaptación de tasa antes de ser enviada sobre el canal físico. El algoritmo de adaptación de tasa proporciona que un número deseado de bits sea enviado sobre el canal a través de perforación o repetición de la palabra de código. El algoritmo de adaptación de tasa es importante en que puede impactar significativamente al rendimiento del sistema y la eficiencia de implementación.

Si el sistema usa corrección de error sin canal de retorno (FEC) solamente, entonces solamente se necesita una transmisión para un bloque de información dado y el algoritmo de adaptación de tasa solamente proporciona una versión de la palabra de código. Si el sistema usa una operación de Petición de Repetición Automática Híbrida (HARQ), entonces se pueden necesitar múltiples transmisiones para un bloque de información dado y se puede requerir al algoritmo de adaptación de tasa que proporcione diferentes versiones de la palabra de código (por ejemplo, para HARQ de redundancia incremental). HARQ degenera a FEC si se usa solamente una transmisión para un bloque de información.

Una forma de realizar la función de adaptación de tasa es definir patrones de perforación / repetición para cada tamaño de bloque de información y tasa de código que el sistema puede encontrar. No obstante, este método es inflexible. Por otra parte, llega a ser poco práctico si el sistema implica miles de combinaciones de tamaño de bloque de información y tasa de código, tal como en un sistema de comunicación del 3GPP.

Otra forma de realizar adaptación de tasa es definir una regla que dé pasos a través de los bits de palabra de código uno por uno y determine si un bit de palabra de código se debería perforar (eliminar) o repetir. Tal método tiene la ventaja de la flexibilidad y se define en la Rel-99 del 3GPP. No obstante, tal método sufre de ineficiencia de implementación. Además, no hay forma de garantizar que la transmisión asociada con un bloque de información dado no se solape (es decir, ortogonal) para HARQ de IR. De esta manera el rendimiento del sistema puede sufrir.

Alternativamente, un algoritmo de adaptación de tasa basado en almacenador temporal circular se puede diseñar para proporcionar buen rendimiento con complejidad de implementación baja. En este método, los bits de la palabra de código se disponen para formar un almacenador temporal circular. Si se necesitan para la transmisión $N_{deseados}$ bits codificados, un bloque de longitud $N_{deseados}$ de bits consecutivos se toma del almacenador temporal circular (envuelto alrededor del principio si se alcanza el último bit del almacenador temporal circular). Por lo tanto, se puede lograr perforación y repetición usando un único método. La técnica de almacenamiento temporal circular tiene ventajas en flexibilidad, rendimiento y facilidad de implementación.

Para realizar las diferentes versiones de palabra de código en una operación HARQ, un parámetro tal como versión de redundancia (RV) se puede usar como una entrada al algoritmo de adaptación de tasa basado en almacenador temporal circular. Las versiones de redundancia pueden definir posiciones de partida dentro del almacenador temporal circular para seleccionar el segmento de la palabra de código para la transmisión.

En la técnica anterior, las posiciones de partida permisibles de las RV se distribuyen por igual sobre el almacenador temporal circular que contiene la secuencia de bits de palabra de código después de la reordenación, lo cual hace difícil implementar cada transmisión de HARQ. Por lo tanto, existe una necesidad de un método y aparato para asignar versiones de redundancia a un almacenador temporal circular dentro de un sistema de comunicación que mejora la eficiencia del algoritmo de adaptación de tasa basado en almacenador temporal circular.

Los documentos del 3GPP “Circular buffer rate matching for LTE” (R1-072245) y “Redundancy version definition for circular buffer rate matching” (R1-072138) describe técnicas de la técnica anterior relacionadas con adaptación de tasa HARQ por medio de almacenadores temporales circulares.

El documento del 3GPP “Rate matching and interleaver for HARQ” (R1-071578), 03-04-2007, describe acortamiento de almacenadores temporales circulares mediante procedimientos de adaptación de tasa que eliminan algunos de los bits de paridad en una primera etapa.

Breve descripción de los dibujos

La FIG. 1 es un diagrama de bloques de un transmisor de la técnica anterior.

La FIG. 2 ilustra intercalado y entrelazado de subbloques.

La FIG. 3 es un diagrama de bloques de un transmisor.

La FIG. 4 y la FIG. 5 ilustran intercalado y entrelazado de subbloques.

La FIG. 6 ilustra un almacenador temporal circular de eliminado ficticio que se puede presentar en formato de matriz.

La FIG. 7 ilustra un almacenador temporal circular de versiones de redundancia que tiene bits ficticios.

La FIG. 8 ilustra el uso de un almacenador temporal circular virtual.

La FIG. 9 ilustra una primera etapa de adaptación de tasa con almacenamiento temporal circular.

La FIG. 10 ilustra perforación de bits sistemáticos.

La FIG. 11 ilustra el salto de una parte pequeña de bits sistemáticos dentro de un almacenador temporal circular.

La FIG. 12 es un diagrama de flujo que muestra la operación del transmisor de la FIG. 3.

La FIG. 13 es un diagrama de flujo que muestra la operación del transmisor de la FIG. 3.

Descripción detallada de los dibujos

A fin de abordar la necesidad mencionada anteriormente, se proporciona en la presente memoria un método y aparato para asignar versiones de redundancia a un almacenador temporal circular dentro de un sistema de comunicación. Durante la operación se crea un almacenador temporal circular en el que solamente se definen versiones de redundancia que comienzan en las partes superiores (es decir, en la primera fila) de los intercaladores de subbloques constituyentes. Por ejemplo, con un total de ocho versiones de redundancia, las versiones de redundancia se colocarían en las posiciones $\lceil Kflujo/32 \rceil$(12xi+σ), i = 0, 1, …, 7 donde σ indica el índice de columna de intercalador de subbloques de la posición de la primera RV (RV_0).

Definir las posiciones de RV como se describió anteriormente provoca una implementación de HARQ más fácil. Más particularmente, esta técnica permite una definición de RV sin memoria que es muy útil en el soporte de un número grande de tamaños de bloque de entrada con adaptación de tasa (RM) de almacenador temporal circular. En particular, la colocación de RV anterior permite la implementación de un almacenador temporal circular virtual (es decir, no se implementa un almacenador temporal circular físico) que permite que un número de bits deseado sea seleccionado directamente de un flujo de salida de codificador comenzando desde cualquier RV en el almacenador temporal circular.

La presente invención abarca un método para asignar versiones de redundancia a un almacenador temporal circular. El método comprende los pasos de recibir bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad. Los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad todos comprenden bits ficticios. Los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad son bloques intercalados individualmente y el primer bloque de bits de paridad se entrelaza con el segundo bloque de bits de paridad para crear bits de paridad entrelazados. Los bits sistemáticos intercalados se anteponen a los bits de paridad entrelazados para crear un almacenador temporal circular y las versiones de redundancia se definen para comenzar en una fila particular del almacenador temporal circular. Cuando se reciben una versión de redundancia (RV) y un número de bits deseados, el número de bits deseados se saca comenzando en la posición de bit de RV.

La presente invención comprende un aparato que comprende un codificador que saca bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad. Se proporciona circuitería de adaptación de tasa que recibe los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad y que crea un almacenador temporal circular que tiene bits ficticios. Una fila particular del almacenador temporal circular se usa como versiones de redundancia (RV) y la circuitería de adaptación de tasa saca bits no ficticios desde el almacenador temporal circular a un transmisor comenzando en una RV particular. Finalmente, se proporciona circuitería de transmisión que recibe los bits no ficticios y que transmite los bits no ficticios.

Finalmente, la presente invención comprende un aparato que comprende un codificador que saca bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad, cada uno de longitud K_{flujo}. El aparato comprende adicionalmente circuitería de adaptación de tasa para recibir los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad e intercalar los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad. Se proporciona circuitería lógica para definir versiones de

redundancia (RV) que corresponden a posiciones dentro de un flujo de datos al cual comenzar a sacar datos. Cuando se usan ocho RV, las RV están en posiciones $\lceil Kflujo/32 \rceil(12xi+\sigma)$, i = 0, 1, ..., 7, donde σ indica el índice de columna del intercalador de subbloques de la posición de partida de la primera RV (RV_0). El flujo comprende los bits sistemáticos intercalados y un primer y segundo bloques intercalados y entrelazados de bits de paridad. Finalmente, se proporciona circuitería de transmisión para recibir bits no ficticios comenzando en una RV particular y transmitiendo los bits no ficticios.

Volviendo ahora a los dibujos, en los que números iguales designan componentes iguales, la FIG. 1 y la FIG. 2 ilustran la funcionalidad de un método de adaptación de tasa basado en almacenador temporal circular. Más particularmente, la FIG. 1 es un diagrama de bloques de un transmisor y la FIG. 2 ilustra el intercalado y entrelazado de subbloques. Durante la operación del transmisor 100, el codificador 101 saca un flujo de bits sistemáticos y al menos dos flujos de paridad. Para este ejemplo particular, el codificador 101 comprende un turbo codificador de tasa 1/3 adoptado en la estandarización del 3GPP. Se señala que los siguientes conceptos se pueden adaptar a otros tipos de códigos de corrección de error, tales como turbo códigos con otras tasas de código, códigos de comprobación de paridad de baja densidad (LDPC), códigos convolucionales, etc.

El codificador 101 saca tres flujos que corresponden al flujo de bits sistemáticos y los dos flujos de paridad. En ciertos casos, el flujo sistemático puede contener varios (por ejemplo, 4) bits que no son sistemáticos, por ejemplo, debido a bits de cola en cuanto al turbo código del 3GPP. (Los bits de cola están ausentes cuando se usa codificación de mordedura de cola.) Los flujos también pueden contener bits de relleno insertados anterior a la turbo codificación. Por simplicidad, todos los bits en el flujo sistemático se refieren como bits sistemáticos y todos los bits en los flujos de paridad respectivos se refieren como bits de paridad 0 y bits de paridad 1, respectivamente.

La circuitería de adaptación de tasa 102 recibe los flujos sacados del codificador 101 y realiza intercalado de subbloques en cada flujo individualmente. Esto se ilustra en la FIG. 2 donde S', P_0' y P_1' son flujo de bits sistemáticos, flujo de bits de paridad 0 y flujo de bits de paridad 1, respectivamente. Cada flujo se reordena (intercala) con su propio intercalador de subbloques (no mostrado en la FIG. 1 ó 2) para producir S, P_0 y P_1. P_0 y P_1 luego se entrelazan uno con otro para producir una parte de P_0 y P_1 entrelazados 201. Un almacenador temporal de salida único 105 (también llamado almacenador temporal circular) se forma almacenando los bits sistemáticos reordenados S al principio seguidos por la parte de P_0 y P_1 entrelazados 201.

Para una tasa de código de operación deseada, el número de bits codificados $N_{deseado}$ que se selecciona para transmisión se calcula y pasa a la circuitería lógica 104 como entrada. La circuitería lógica 104 simplemente lee un bloque de longitud $N_{deseado}$ de bits consecutivos de la secuencia, el almacenador temporal circular (envuelto alrededor del principio si se excede el último bit del almacenador temporal circular) desde un cierto punto de partida. Por lo tanto, la perforación y la repetición se pueden lograr usando un único método. La técnica de almacenamiento temporal circular tiene una ventaja en la flexibilidad (en las tasas de código logradas) y la granularidad (en tamaños de flujos). La adaptación de tasa de almacenador temporal circular selecciona los bits de paridad distribuidos aproximadamente por igual sobre el enrejado de código con independencia de la tasa de código de operación deseada si los intercaladores de subbloques están diseñados adecuadamente.

La FIG. 3 es un diagrama de bloques del transmisor 300. El transmisor 300 comprende el codificador 301 y la circuitería de adaptación de tasa 303. La circuitería de adaptación de tasa 303 comprende adicionalmente una circuitería lógica 305 y una memoria 307. La circuitería lógica 305 es preferiblemente un microprocesador, microcontrolador, procesadores de señal digital (DSP) u otros dispositivos tales conocidos por los expertos en la técnica. Las operaciones/funciones particulares de la circuitería lógica 305, de esta manera del transmisor 300, se determinan mediante una ejecución de instrucciones y rutinas software. La memoria 307 comprende una memoria de acceso aleatorio (RAM), memoria de acceso aleatorio dinámica (DRAM) y/o memoria de sólo lectura (ROM) o equivalentes de las mismas, que se usan como un almacenador temporal circular. El codificador 301 es preferiblemente un turbo codificador de velocidad 1/3 del 3GPP, no obstante, las técnicas descritas en la presente memoria para operar el transmisor 300 se pueden aplicar a otros codificadores, incluyendo, pero no limitados a, turbo codificadores que realizan turbo codificación con otras tasas de código, con bits de cola o sin bits de cola, turbo codificadores de mordedura de cola, binarios o dúo binarios, con o sin inserción de bits de relleno, ..., etc.

Durante la operación el codificador 301 saca tres flujos que corresponden al flujo de bits sistemáticos y a los dos flujos de paridad. La circuitería lógica 305 recibe los flujos sacados del codificador 301 y realiza intercalado de subbloques en cada flujo individualmente. La circuitería lógica 305 entonces entrelaza P_0' y P_1'. Un almacenador temporal de salida único 307 (*almacenador temporal circular*) se forma almacenando los bits sistemáticos reordenados S al principio seguidos por P_0 y P_1 entrelazados.

Aunque el intercalador de subbloques puede ser cualquier permutador, se usa por simplicidad normalmente un intercalador de rectángulo (también llamado intercalador de bloques) de N_r filas y N_c columnas. Las operaciones son directas si el tamaño de flujo K_{flujo} de cada flujo es igual al tamaño del intercalador de rectángulo $N_r \times N_c$ (es decir, el bloque está completo). No obstante, a menudo el tamaño del flujo es menor que $N_r \times N_c$, de esta manera se necesitan ($N_r \times N_c - K_{flujo}$) bits ficticios para llenar el bloque.

Este proceso de formación del almacenador temporal circular se ilustra en la FIG. 4 y la FIG. 5 con 4 bits ficticios insertados en cada flujo. Con referencia a la FIG. 4, los bits sistemáticos 401, que tienen añadidos bits ficticios 407, son bloques intercalados para producir bits sistemáticos permutados S 402. De una manera similar, los bits de paridad P_0' y P_1' 403 (que tienen añadidos bits ficticios) se intercalan para producir los bits de paridad P_0 and P_1 404. Los bits ficticios se añaden a los flujos de manera que el intercalador de bloques está lleno para cada flujo. P_0 y P_1 se entrelazan para producir la parte entrelazada P_0 y P_1 405 de la parte entrelazada. Las dos matrices S y P_{0-1} se combinan como se muestra en la FIG. 5 y almacenan en el almacenador temporal 307. El almacenador temporal 307 se lee en modo columna, comenzando en la parte superior de la matriz. Aunque en la descripción anterior los bits ficticios se insertan después de un flujo, en ciertas realizaciones, también se pueden insertar antes del flujo.

Durante la operación del transmisor 300, se calcula un número de bits codificados $N_{deseado}$ que se selecciona para la transmisión y se pasa a la circuitería lógica 305. La circuitería lógica 305 simplemente lee un bloque de longitud $N_{deseado}$ de bits consecutivos desde el almacenador temporal de salida 307 (envueltos alrededor del principio si se excede el último bit del almacenador temporal de salida) desde un cierto punto de partida. Para la operación de Petición de Repetición Automática Híbrida (HARQ), se proporciona un parámetro (versión de redundancia (RV)) a la circuitería lógica 305 para definir el punto de partida dentro del almacenador temporal 307 de manera que se puedan seleccionar diferentes secciones del almacenador temporal para la transmisión. Dado que FEC es equivalente a HARQ con una transmisión solamente, FEC también se puede definir con un valor de RV. De esta manera, la circuitería lógica 305 recibe un valor de RV y el número de bits codificados $N_{deseado}$ que se selecciona para la transmisión. Se leen $N_{deseado}$ bits del almacenador temporal 307 comenzando en la posición definida por RV. Estos bits se sacan típicamente al transmisor 311 para su modulación y transmisión posterior.

Se debería señalar que el formato particular para S y los P_o y P_1 entrelazados se dan anteriormente con propósitos de ilustración y se pueden formatear de varias formas. Por ejemplo, aunque el almacenador temporal circular se representa usando una formación de una única dimensión, a menudo es útil usar un formato de matriz bidimensional con propósitos matemáticos.

Si los bits ficticios se mantienen en el almacenador temporal circular, se llama un almacenador temporal circular de relleno ficticio. El tamaño del almacenador temporal circular de relleno ficticio es igual a la suma de los tamaños de flujos de entrada y el número total de bits ficticios insertados. Como estará claro a partir del contexto, el almacenador temporal circular de relleno ficticio puede estar en formato de matriz como se ilustra en la FIG. 5 o en formato de secuencia (es decir, una disposición o flujo de datos de una única dimensión) como se ilustra en la FIG. 2. Si se eliminan los bits ficticios, se llama un almacenador temporal circular de eliminado ficticio. El tamaño del almacenador temporal circular de eliminado ficticio es igual a la suma de los tamaños de flujos de entrada. Del mismo modo, el almacenador temporal circular de eliminado ficticio se puede presentar en formato de matriz como en la FIG. 6 en la que existen discontinuidades donde quiera que fueran eliminados los bits ficticios o en formato de secuencia. Las RV de la técnica anterior se distribuyen aproximadamente por igual sobre el almacenador temporal circular de eliminado ficticio, lo que hace difícil implementar cada transmisión de HARQ, ya que la posición de partida de RV precisa en el flujo de salida del codificador necesita ser calculada usando operaciones no triviales para cada caso. A fin de abordar este problema, se puede usar un almacenador temporal circular de relleno ficticio 307 para definir una posición de partida para una versión de redundancia.

El siguiente texto puede ser útil en descripciones posteriores de técnicas de adaptación de tasa.

- K_{info} se refiere al tamaño del bloque de información (o la longitud del paquete de mensaje)
- K_{FEC} se refiere a un tamaño de bloque de entrada soportado por el codificador FEC (K_{FEC} es igual al tamaño del intercalador para el turbo código binario y es igual a dos veces el tamaño del intercalador para el turbo código dúo binario.)
- $K_{relleno}$ se refiere al número de bits de relleno añadidos al bloque de Información para obtener un tamaño de entrada que se soporta por el codificador FEC. $K_{relleno} = K_{FEC} - K_{info}$.
- $K_{flujo} = K$ se conoce como longitud del flujo, que es igual o ligeramente mayor que el K_{FEC} dependiendo del método de terminación de enrejado usado en el turbo código. Para el turbo código del 3GPP, hay tres flujos, cada uno de longitud $K_{flujo} = K_{FEC} + 4$, que consta de los bits sistemáticos, bits de paridad de entre el primer y segundo codificadores, respectivamente y 12 bits de cola se distribuyen uniformemente en los tres flujos.
- N_r se refiere al número de filas en el intercalador de subbloques usado en el almacenador temporal circular
- Nc se refiere al número de columnas en el intercalador de subbloques usado en el almacenador temporal circular
- $K_{rect} = N_r \times N_c$ es la dimensión del intercalador de subbloques usado en el almacenador temporal circular. En general K_{rect} se elige que sea mayor o igual que K_{flujo} aunque también se puede elegir que sea mayor o igual que K_{FEC} si los bits de relleno se descartan como parte del algoritmo de adaptación de tasa.

Definición de RV

Para ilustración, consideremos el caso cuando el intercalador de subbloques tiene 30 columnas y necesitan ser definidas 8 RV. En este caso, el almacenador temporal en la FIG. 7 puede tener 90 columnas que no se pueden dividir por igual en las ocho RV. Es difícil averiguar el punto de partida de cada RV en el flujo de salida del codificador (excepto RV_0 si RV_0 siempre comienza desde el principio del almacenador temporal circular). Suponiendo que se necesitan 8 RV distribuidas por igual, entonces los puntos de partida de RV_i son K_{flujo}x3/8xi (cuando sea necesario, llevar a un valor entero a través de operaciones tales como redondeo, suelo o techo), si no incluir los bits ficticios (es decir, almacenador temporal circular de eliminado ficticio), i = 0, 1, ... 7. Alternativamente, si se cuentan los bits ficticios (es decir, almacenador temporal circular de relleno de bits), entonces el punto de partida de RV_i es N_r x N_c x3/8xi = N_r x90/8xi (cuando sea necesario, llevar a un valor entero a través de operaciones tales como redondeo, suelo o techo). En cualquier caso, los puntos de partida de 7 RV se sitúan casi siempre en el medio del rectángulo N_r x90 si se desean posiciones de partida de RV distribuidas por igual. (Una solución es dejar que todos los puntos de partida de 7 RV estén en la fila superior del rectángulo N_r x30, conduciendo por ello a una distribución ligeramente desigual de las RV en el almacenador temporal). Señalar lo siguiente si los intercaladores de subbloques de 30 columnas se aplican al estándar de Evolución a Largo Plazo (LTE) del 3GPP.

- Si se almacenan los puntos de partida de RV, entonces necesitan ser almacenados un total de 188x7 puntos de partida, gastando memoria. Señalar que hay 188 tamaños de intercalador QPP y 7 RV que necesitan ser almacenadas para cada tamaño.

- Si los puntos de partida de RV se calculan sobre la marcha, entonces necesitan ser programadas en el hardware dos operaciones difíciles para averiguar los índices de fila y columna de un punto dentro del rectángulo de los bits sistemáticos o del rectángulo de la paridad 0 / paridad 1. Por ejemplo, si una RV comienza en el índice de orden L dentro del flujo sistemático, entonces las dos operaciones son:

    - División para averiguar el índice de columna = suelo(L /_N_r) suponiendo que el índice de columna comienza en 0. Dado que N_r cambia con K_{flujo} si N_c se fija a 30 y N_r es muy probable que no sea una potencia de 2, la división no es trivial en hardware.

    - Módulo para averiguar el índice de fila = mod(L, N_c) = mod(L,30). Dado que 30 no es potencia de 2, la operación de módulo tampoco es trivial en hardware.

Usar N_c = 30 contribuyó a las dificultades de localizar las RV en una gran medida. En su lugar, es preferible tener que el número total de columnas en el almacenador temporal 307 sea un múltiplo del número total de las RV soportadas para tener las RV distribuidas por igual (aunque las RV todavía se pueden definir de manera desigual). Por ejemplo, el intercalador de subbloques puede usar 32 columnas cuando van a ser definidas 8 RV. A menudo, necesitan ser definidas 2^c RV, donde c es un entero. De esta manera, es conveniente usar $N_c = 2^d$ donde d es un entero mayor o igual que c. Para reducir aún más la complejidad y minimizar la cantidad de bits ficticios, es preferible usar el mismo valor d para todos los tamaños de bloques de información del sistema. Por ejemplo, se puede usar la constante d=5 (de esta manera N_c = 32 columnas) para todos los tamaños de bloques de información de LTE del 3GPP.

Adicionalmente, las RV distribuidas por igual se pueden definir que comiencen siempre en las partes superiores (es decir, la primera fila) de las columnas de los intercaladores de subbloques constituyentes si los bits ficticios no se descartan anterior a definir la posición de partida de las RV, evitando de esta manera operaciones complejas tales como división y módulo tratadas anteriormente.

Si N_c = 32 se usa el intercalador de subbloques con un patrón de permutación entre columnas = {0, 16, 8, 24, 4, 20, 12, 28, 2, 18, 10, 26, 6, 22, 14, 30, 1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31}, entonces se usa el mismo número de columnas para todos los K_{flujo}, con el número de filas que cambia con K_{flujo}, N_r = $\lceil Kflujo / N_c \rceil = \lceil Kflujo / 32 \rceil$. Dado que, normalmente se requieren las RV para dividir el almacenador temporal 307 en secciones aproximadamente uniformes, se requiere una sobrecarga computacional adicional para modelar la desigualdad en la definición de RV para cada tamaño de flujo si se usa un almacenador temporal circular de eliminado ficticio.

Para mayor claridad, consideremos el caso cuando el número de RV = 8, el número de columnas en un intercalador de subbloques es N_c = 32 y el tamaño del flujo de entrada es K_{flujo} y se usa una permutación de longitud 32 columnas en el intercalado de subbloques de S, P_0 y P_1.

La FIG. 6 muestra un almacenador temporal circular de eliminado ficticio (es decir, después de descartar bits ficticios) con 8 RV definidas. Señalar que hay discontinuidades donde quiera que fueron descartados bits ficticios; se obtiene un almacenador temporal circular de eliminado ficticio en formato de secuencia (se puede usar en el almacenador temporal 307) leyendo los elementos de matriz en modo columna desde la esquina superior izquierda hacia adelante. La falta de regularidad en la distribución de RV en los rectángulos causa dificultad en la operación HARQ donde cualquier valor de RV se puede introducir al bloque de RM. Los índices de fila y columna para el punto

de partida de la RV tienen que ser calculados en base al tamaño del flujo (y el número de bits ficticios), conduciendo de esta manera a un hardware complicado.

Se describe a continuación una forma alternativa para facilitar una operación HARQ con un almacenador temporal circular. En este caso, los bits ficticios se descartan después de que las RV se definen como se muestra en la FIG. 7. De hecho, los bits ficticios se dejan en el almacenador temporal circular (almacenador temporal circular de relleno ficticio) y dado que el tamaño de almacenador temporal circular de relleno ficticio es un múltiplo de 8, es posible definir ocho RV que están separadas por igual para cualquier tamaño de flujo K_{flujo}. En particular, las ocho RV se pueden definir que comiencen en la parte superior de una columna del intercalador de subbloques. Es conveniente, en general, asignar las RV a cualquier posición en la primera fila (es decir, en la parte superior de una columna) del intercalador de subbloques. Por ejemplo, las ocho RV pueden comenzar en la parte superior de las ocho columnas siguientes 2, 14, 26, 38, 50, 62, 74, 86 en la FIG. 7 (con $N_c = 32$). Cuando se describe en términos de posición en el almacenador temporal circular de relleno ficticio de formato de secuencia, las versiones de redundancia comienzan en las posiciones $\lceil Kflujo/32 \rceil(12xi+\sigma)$, i = 0, 1, ..., 7 donde σ indica el índice de la columna de la posición de partida de la primera RV (RV_0).

Señalar que esta definición de RV es sin memoria para un almacenador temporal circular de relleno ficticio, en el sentido de que es fácil comenzar a sacar los bits de código deseados desde cualquier RV sin conocer la posición de partida de las RV anteriores. Los bits ficticios se descartan cuando los bits se están leyendo del almacenador temporal circular de relleno ficticio. Incluso si el número total de columnas NO es un múltiplo del número total de RV soportadas, por ejemplo, N_c=30 con 8 RV, las RV aún se pueden definir que comiencen en la parte superior de 8 cualesquiera de las 90 columnas.

Aunque la parte superior de la columna se usa en la discusión anterior como una forma simple para definir las posiciones de partida de las RV, también se pueden usar otras posiciones convenientes. Por ejemplo, las posiciones de partida de las RV están todas situadas en una fila particular *r* del almacenador temporal circular de relleno ficticio, donde $0 \leq r < Nr$.

La justificación para definir las RV sobre el almacenador temporal circular de relleno ficticio se proporciona a continuación.

- La permutación de cada flujo (o intercalador de subbloques) se describe como la formación rectangular $K_{rect} = N_r \times N_c$ con bits ficticios.
    - Cuando $K_{flujo} = N_r \times N_c$, los bits de cada flujo se escriben en el rectángulo en modo fila y se leen en modo columna (después de la permutación de columna).
    - Cuando $K_{flujo} < N_r \times N_c$ los bits de cada flujo se escriben en el rectángulo en modo fila y se leen en modo columna y se rellenan ($N_r \times N_c - K_{flujo}$) bits ficticios para llenar completamente el rectángulo. Los bits ficticios son borran o “cortan” antes de la transmisión sobre el canal.
- Se debería señalar que cuando el rectángulo está lleno (bits ficticios usados o no), *la permutación de cada flujo tiene una expresión en forma cerrada*, lo que significa que es fácil determinar la permutación inversa. Si es fácil determinar la permutación inversa, por lo tanto también es fácil determinar la posición en el flujo de salida del codificador que corresponde a una RV particular. Si el rectángulo está parcialmente lleno (sin bits ficticios y $K_{flujo} < N_r \times N_c$), *la permutación de cada flujo no tiene una expresión en forma cerrada*. En este caso, es difícil determinar la permutación inversa y por lo tanto también es difícil determinar la posición en el flujo de salida del codificador que corresponde a una RV particular.

Por facilidad de explicación, en las discusiones anteriores, se supone que el almacenador temporal circular 307 (de eliminado ficticio o de relleno ficticio) se forma físicamente, por ejemplo, almacenando los bits del almacenador temporal circular en una memoria. Para implementación de hardware, es mejor utilizar el concepto de almacenador temporal circular *virtual* que genera directamente los bits de salida deseados sin formar el almacenador temporal circular físico 307. En otras palabras, la funcionalidad de adaptación de tasa se cumple sin el almacenamiento intermedio. Este concepto se muestra en la FIG. 8.

El paso intermedio de formar el almacenador temporal circular físico se puede evitar encontrando una regla algebraica para generar fácilmente una parte deseada del almacenador temporal circular físico sin tener que pasar a través de todo el proceso de formación de almacenador temporal circular físico.

En el almacenador temporal circular virtual, el algoritmo de adaptación de tasa inicializa su generador de direcciones a un valor de partida adecuado en base a los parámetros de entrada (número de RV y tamaño de entrada). Entonces comienza directamente sacando los bits siguiendo la regla de generación de direcciones que se describe por las permutaciones de subbloques de los flujos de entrada. La regla de generación de direcciones define la secuencia de posiciones en el flujo de salida del codificador desde el cual se sacan los bits.

El almacenador temporal circular virtual se puede operar tanto como un almacenador temporal circular de relleno ficticio como un almacenador temporal circular de eliminado ficticio. No obstante, si se usa un almacenador temporal circular de eliminado ficticio, entonces las RV pueden comenzar en cualquier parte en el almacenador temporal circular (en el formato de matriz) como se ilustra en la FIG. 6. Por lo tanto, en tal escenario, los puntos de partida de RV pueden necesitar ser almacenados explícitamente, gastando memoria. Alternativamente, los puntos de partida de RV se pueden derivar para cada caso, pero la inicialización de dirección en la operación de almacenador temporal circular virtual llega a ser complicada de implementar en hardware.

Por el contrario, para operar el almacenador temporal circular virtual cuando se utiliza un almacenador temporal circular de relleno ficticio, las RV comienzan en la parte superior de las columnas del intercalador de subbloques y por lo tanto la inicialización de dirección en la operación del almacenador temporal circular virtual llega a ser más eficiente. Siempre que el generador de direcciones apunte a un bit ficticio, el generador de direcciones se avanza automáticamente sin sacar un bit.

Definición de RV con adaptación de tasa de 1ª Etapa

Se puede necesitar una técnica de adaptación de tasa de 1ª etapa cuando el receptor tiene una cantidad limitada de tamaño de almacenador temporal flexible. En la adaptación de tasa de 1ª etapa, el transmisor puede tener conocimiento de la capacidad del almacenador temporal flexible del receptor y por lo tanto se le permite transmitir solamente no más bits de código que se puedan almacenar en el almacenador temporal flexible del receptor.

El tamaño del almacenador temporal flexible máximo puede imponer una cierta restricción en la adaptación de tasa con el almacenador temporal circular. En particular, el almacenador temporal circular para cada bloque de código se puede limitar además como se muestra en la FIG. 9. En este caso, la envoltura alrededor del almacenador temporal circular puede ocurrir en un punto anterior al final del almacenador temporal circular.

En general, si la entrada (bloque de Transporte o bloque de Transporte concatenado) al turbo codificador es mayor que el tamaño máximo soportado por el turbo intercalador, entonces el TB se segmenta en múltiples segmentos de bloques de código, cada uno de los cuales se turbo codifica individualmente y adapta en tasa, permitiendo de esta manera una operación canalizada. Por lo tanto, cada segmento del TB puede tener su propio almacenador temporal circular (o almacenador temporal circular virtual) (Señalar que no es necesario crear almacenadores temporales circulares físicos para cada segmento.). Supongamos que N_{IR} es el tamaño total del almacenador temporal flexible por proceso HARQ (o tamaño de almacenador temporal máximo que corresponde a un TB).

Si la adaptación de tasa se realiza sobre una base de palabra de código por palabra de código (es decir, segmento por segmento), es preferible que la adaptación de tasa de 1ª etapa, si se incluye, se realice sobre una base de palabra de código por palabra de código (segmento por segmento) también. El tamaño del almacenador temporal circular para cada palabra de código (o segmento) se limita a N_{cw}, donde $N_{cw} < 3 \times K_{flujo}$ y $3 \times K_{flujo} = 3 \times K_{info} + 12$ es el tamaño completo del almacenador temporal circular (por segmento) antes de la adaptación de tasa de 1ª etapa. Señalar que cuando $N_{cw} \geq 3 \times K_{flujo}$, la RM de primera etapa (por segmento) es transparente. La suma del tamaño N_{cw} para todas las palabras de código (de todos los bloques de transporte) no puede exceder N_{IR}, que es el tamaño del almacenador temporal flexible de todos los TB. En lugar de tomar N_{IR} como entrada al algoritmo de RM de 1ª etapa, la RM de 1ª etapa de la RM por palabra de código en LTE puede tomar N_{cw} como entrada.

Las opciones para la adaptación de tasa de 1ª etapa pueden incluir:

1. Limitar el tamaño del almacenador temporal a un valor máximo fijado para la clase de UE (en todos los TB, todas las palabras de código de un TTI dado. Una clase de Equipo de Usuario (UE) define un conjunto de capacidades de transmisor/receptor en 3GPP y LTE del 3GPP). Determinar el límite por TB y por palabra de código a partir de N_{IR} a través de escalado. Por ejemplo, N_{cw} = suelo (N_{IR}/C), donde C es el número de turbo palabras de código para los TB concatenados, determinado por la regla de segmentación de bloques de código. Es posible determinar el límite por palabra de código de una manera similar a la regla de segmentación de bloque de código.

2. Limitar el almacenador temporal a un valor máximo fijado N'_{IR} para la clase de UE (fijar el límite por TB). Determinar el límite por palabra de código a partir de este máximo a través de escalado.

3. Limitar el almacenador temporal a un valor máximo fijado por palabra de código para la clase de UE.

Con cualquiera de estas tres opciones, se puede definir el límite por palabra de código borrando columnas enteras del almacenador temporal circular virtual (en segmentos individuales).

Para las definiciones de RV, las opciones incluyen:

Esquema 1. Permitir solamente un subconjunto de RV posibles. Si se definen Y RV sin la adaptación de tasa de 1ª etapa (es decir, la adaptación de tasa de 1ª es transparente), entonces Y' RV de entre las Y RV disponibles se pueden mantener cuando la adaptación de tasa de la 1ª etapa no es transparente, Y' <= Y. Por ejemplo, se usan RV-i, i = 0, 1, ..., Y'-1, donde el punto de partida de RV-Y' es menor que N_{cw}.

Esquema 2. Redefinir las RV de manera que un conjunto completo de Y RV esté disponible dentro del tamaño de almacenador temporal circular máximo permitido. Las RV se redefinen de manera que comienzan en la parte superior de las columnas del almacenador temporal circular con bits ficticios insertados. La cantidad de perforación de bits sistemáticos no se modifica, de manera que RV 0 comienza en la misma posición para todas las clases de UE. Por ejemplo, si N_{col} columnas del almacenador temporal circular de relleno ficticio (que contiene 3 x N_c columnas antes de la adaptación de tasa de 1ª etapa) permanecen después de la adaptación de tasa de 1ª etapa, entonces las Y RV pueden comenzar en las partes superiores de columnas de índice de columna suelo((N_{col}-2)/Y)xi+2, i = 0, 1, ..., Y-1.

Los puntos de partida de RV también se pueden definir sobre una base por palabra de código, aunque es preferible señalar una única RV por TB para una operación de HARQ eficiente.

Perforación de bits sistemáticos

La definición de RV tratada anteriormente se puede refinar para acomodar una técnica de mejora de rendimiento llamada perforación sistemática.

Es bien conocido en la bibliografía que los turbo códigos parcialmente sistemáticos pueden superar a los turbo códigos sistemáticos, una explicación simple que es que estos últimos son un subconjunto de los primeros. No obstante, la perforación de bits sistemáticos no se usa en los algoritmos de adaptación de tasa en los estándares inalámbricos existentes tales como turbo codificación WCDMA o CDMA2000. Una razón principal para esto que es que se prefiere que la transmisión (en RV_0) sea decodificable automáticamente cuando la tasa de codificación efectiva es alta.

La perforación de bits sistemáticos se usa para evitar perforación excesiva de los bits de paridad, lo que de otro modo conduciría a distancias mínimas escasas a tasas de codificación más altas y por lo tanto a un rendimiento inferior.

En sistemas de comunicación tales como 3GPP, las tasas de codificación quizás pueden ser tan altas como 5/6 (o incluso más altas) y por lo tanto tales sistemas deberían ser capaces de manejar una pequeña fracción de perforación de bits sistemáticos. Señalar que tales códigos parcialmente sistemáticos todavía son decodificables automáticamente. Para un turbo código con un tamaño de bloque K_{info}, permitamos que $\Delta K(R)$ indique la fracción de bits sistemáticos que se perforan a la tasa de código R.

Un reto en el diseño de la adaptación de tasa es entonces encontrar valores óptimos de $\Delta K(R)$ para todas las combinaciones válidas de K_{info} y R. Después de que se determina $\Delta K(R)$, el almacenador temporal circular se puede reconfigurar para incluir perforación de bits sistemáticos. Hay muchas opciones para la reconfiguración como se muestra en la FIG. 10. En la discusión de la perforación sistemática, se usa el almacenador temporal circular en formato de secuencia para ilustrar las técnicas y el almacenador temporal circular puede ser de relleno ficticio o de eliminado ficticio. En los siguientes ejemplos, se supone que la tasa de código madre es 1/3 y los bits se toman siempre secuencialmente comenzando desde el primer bit en el almacenador temporal circular. Como se mencionó, K_{info} indica el tamaño de bloque de información, K_{flujo} indica la longitud de un flujo.

1. La opción 1 en la FIG. 10 describe el almacenador temporal circular normal en formato de secuencia en el que se transmiten primero todos los bits sistemáticos, seguidos por algunos o todos los bits de paridad.

2. La opción 2 en la FIG. 10 contiene los mismos bits que el almacenador temporal circular normal, pero se modifica el almacenador temporal circular; algunos de los bits sistemáticos (S_A) se toman de la parte sistemática del almacenador temporal circular y se colocan al final del almacenador temporal. Por lo tanto, en la primera transmisión, los bits sistemáticos en S_A se perforan a muchas de las tasas de código. La opción 2 también se puede lograr simplemente colocando RV_0 al comienzo en una posición con un desplazamiento con respecto al principio del almacenador temporal circular. Este desplazamiento se puede determinar en base a la cantidad de perforación de bits sistemáticos requerida. Para la tasa de código $R>=K_{info}/(3xL-S_A)$, los bits sistemáticos en S_A se perforan siempre.

3. La opción 3 es la misma que la opción 2 en que se modifica el almacenador temporal circular; No obstante, los bits sistemáticos (S_A) tomados de la parte sistemática del almacenador temporal circular se colocan en el almacenador temporal después de X bits de los flujos de paridad. Por lo tanto, en la primera transmisión, algunos de los bits sistemáticos se perforan a muchas de las tasas de código. Para la tasa de código $R>=K_{info}/(S_B+X)$, los bits sistemáticos en S_A se perforan siempre.

    a. A partir de la tasa de código $K/(S_B+X) > R > K_{info}$ (S_B+X+S_A), se perforan parte de los bits sistemáticos en S_A.

    b. Para tasas de código $K_{info}/(S_B+X+S_A)>=R$, se transmiten todos los bits sistemáticos.

4. La opción 4 es la misma que la opción 3 en que se modifica el almacenador temporal circular. No obstante, los bits sistemáticos (S_A) tomados de la parte sistemática del almacenador temporal circular se distribuyen

en el almacenador temporal en un cierto intervalo. Por lo tanto, en la primera transmisión, algunos de los bits sistemáticos se perforan a muchas de las tasas de código. La opción 4 permite más flexibilidad en el diseño de un almacenador temporal circular, optimizando la relación de perforación sistemática $\Delta K(R)$. Por ejemplo, los bits sistemáticos en S_A se pueden distribuir por igual dentro de un intervalo; o se distribuyen en densidad creciente de bits sistemáticos dentro de un intervalo; o se distribuyen en densidad decreciente de bits sistemáticos dentro de un intervalo.

Existen otras variaciones a las opciones enumeradas anteriormente. Por ejemplo, los bits sistemáticos en S_A pueden no estar situados en la parte delantera del flujo sistemático intercalado en subbloques S', en lugar de en el medio o al final de S'. También se pueden tomar de ubicaciones discontinuas de S', tales como un bit por cada x bits de S', x>1.

Aunque la nueva modificación se muestra como un paso adicional realizado después de formar el almacenador temporal circular, este paso se podría combinar directamente en la definición del almacenador temporal circular.

Las diferentes opciones tienen diferentes soluciones de compromiso de complejidad/rendimiento. La opción 2 puede no ser preferible ya que la perforación de bits sistemáticos es fija para todas las tasas casi cercanas a 1/3. Las simulaciones sugieren que, para el LTE, se puede preferir perforación de bits sistemáticos de alrededor del 5% para tasas altas tales como 3/4 o 5/6, mientras que se prefiere 0% para tasas de codificación más bajas tales como 2/3 o 1/2. Por lo tanto, puede ser preferible colocar los bits sistemáticos en partes del almacenador temporal como se dicta por tales razones. Alternativamente, se ha sugerido usar RV_7 a tasas de codificación bajas para desactivar la perforación de bits sistemáticos.

Aunque la discusión se centra en el diseño de adaptación de tasa para la primera transmisión (de esta manera, FEC), el mismo almacenador temporal circular reconfigurado se puede usar en el contexto de HARQ con múltiples transmisiones por bloque de información. Se pueden obtener diferentes versiones de redundancia para transmisión HARQ tomando diferentes secciones del almacenador temporal circular reconfigurado, en el que el tamaño de la sección es igual a la longitud requerida para la transmisión del canal físico actual. En particular, si se necesitan Y versiones de redundancia (RV), una forma simple de definir Y diferentes puntos de partida (A_0, A_1, ..., A_{Y-1}) dentro del almacenador temporal circular reconfigurado, uno para una versión de redundancia. Para RV_i, los bits se toman de A_i a mod(A_i + N_{tx}, N_{CB}), donde i = 0, 1, ..., Y-1 y N_{tx} es el número de bits requeridos para la transmisión, N_{CB} es la longitud del almacenador temporal circular. En otras palabras, se leen N_{tx} bits comenzando con la posición A_i, envueltos alrededor del principio del almacenador temporal circular si se alcanza el final del almacenador temporal. Para un bloque de información dado, N_{tx} puede ser diferente para cada transmisión.

Definición de RV con perforación sistemática

Combinando (a), la definición de RV en base a un almacenador temporal circular de relleno ficticio y (b), perforación sistemática, el diseño de adaptación de tasa mejorado tiene las siguientes ventajas, usando N_c = 32 y la tasa de código madre 1/3 (es decir, tres flujos de entre el turbo codificador 101) por ilustración:

- Bits ficticios incluidos en el almacenador temporal circular para facilitar la definición de RV (es decir, almacenador temporal circular de relleno ficticio).
- Facilidad de inicio de cada RV para retransmisiones HARQ. Por ejemplo, las RV se definen para comenzar en la parte superior (es decir, la primera fila) de las columnas del intercalador de subbloques en el almacenador temporal circular de formato de matriz. Más específicamente,
  - Si necesitan ser definidas 8 RV, RV_i comenzaría con la posición N_r x (N_c x 3/8xi+σ) = N_r x (12xi+σ) = $\lceil Kflujo/32 \rceil$(12xi+σ), i = 0, 1, ..., 7, en el almacenador temporal circular de relleno ficticio de formato de secuencia. En formato matriz, RV_i comienza en la parte superior de la columna de orden (12xi+σ).
  - Si solamente se definen 4 RV, cada RV comienza en la parte superior de la columna de orden (24xi+σ) (con σ = 2). De manera equivalente, en formato de secuencia, RV_i comienza con la posición de bit N_r x (N_c x 3/4 x i + σ) = N_r x (24xi+σ) = $\lceil Kflujo/32 \rceil$(24xi+σ), i = 0, 1, ... , 3.

Cuando σ = 0 (módulo 3xN_c), RV_0 (es decir, i = 0) comenzaría al principio del almacenador temporal circular, de esta manera sin perforación de bits sistemáticos. La perforación de bits sistemáticos se puede activar ajustando que σ sea un valor mayor que 0 (módulo 3xN_c) para 3 flujos (es decir, tasa de código madre 1/3), saltando de esta manera una pequeña fracción de bits sistemáticos cuando se leen los bits para RV_0. En otras palabras, RV_0 (la primera versión de redundancia) se define para comenzar en la parte superior de la columna de orden σ, donde σ es el número de columnas desplazadas. Esto se muestra en la FIG. 11. Por ejemplo, RV_0 se puede definir que comience desde la columna #2 en lugar de la columna #0 ajustando σ = 2. También se pueden considerar otros valores de σ.

El diseño mejorado tratado anteriormente puede tener distintas variaciones. Algunas de las cuales siguen:

- Aunque se usan 32 columnas en el diseño propuesto, se pueden usar como N_c otros valores, especialmente potencias de 2, tales como 8, 16, 64 ó 128.
- Aunque se usa una permutación de columnas particular en el ejemplo de diseño, se puede usar otra permutación para permutar las columnas también.
- Si se usan dos o más N_c en el diseño de adaptación de tasa, entonces cada N_c se puede usar para numerosos tamaños de intercalador.
- El intercalador de subbloques puede no ser el mismo para todos los flujos. Por ejemplo, el intercalador de subbloques puede ser el mismo para los flujos tanto sistemáticos como de paridad 1, pero el intercalador de subbloques del flujo de paridad 2 puede ser diferente.
- Aunque por facilidad de descripción, las RV se definen que comienzan en las partes superiores de las columnas del intercalador de subbloques, también se pueden definir que comiencen en otras ubicaciones convenientes en el almacenador temporal circular de relleno ficticio. Por ejemplo, todas las RV pueden comenzar en la misma fila *r* de los intercaladores de subbloques constituyentes, donde $0 \leq r < N_r$.

Definición de RV con adaptación de tasa de 1ª etapa y perforación sistemática

Cuando se usan tanto perforación sistemática como adaptación de tasa de 1ª etapa, puede necesitar ser modificada la definición de RV para cumplir con los siguientes requisitos:

- Facilidad de definición de RV incluyendo bits ficticios en el almacenador temporal circular (es decir, almacenador temporal circular de relleno ficticio).
- Facilidad de inicio de la generación de direcciones para cada RV para retransmisiones HARQ. Por ejemplo, se podrían definir las RV para comenzar en las partes superiores de las columnas del intercalador de subbloques (es decir, la primera fila de una columna de intercalador de subbloques) en el almacenador temporal circular de formato de matriz.
- Mantener las posiciones de partida de las RV dentro del intervalo de los límites del almacenador temporal circular después de la adaptación de tasa de 1ª etapa.

Cuando se usan tanto perforación sistemática como adaptación de tasa de 1ª etapa, la implementación se puede ilustrar como sigue. En primer lugar, la RM de 1ª etapa se logra borrando una o más columnas (preferiblemente de la última columna del almacenador temporal circular virtual) del almacenador temporal circular para formar un almacenador temporal circular acortado. De esta manera, los bits sistemáticos se pueden perforar definiendo la primera versión de redundancia para comenzar en una posición con un desplazamiento con respecto al principio del almacenador temporal circular acortado. En otras palabras, la perforación de bits sistemáticos se puede lograr definiendo RV_0 en un cierto número de columna distinta de cero en el intercalador de subbloques del intercalador de subbloques sistemático. Señalar que los pasos anteriores también se pueden combinar directamente. De nuevo, se puede considerar la adaptación de tasa de 1ª etapa que se aplica sobre una base por palabra de código (o por segmento). Por comodidad, el almacenador temporal circular acortado también se refiere como almacenador temporal circular después de que se aplique la adaptación de tasa de 1ª etapa. Esta distinción debería estar clara a partir del contexto.

Por ejemplo, si se usan 8 RV como la línea de base (cuando está ausente la RM de 1ª etapa), entonces las siguientes opciones están disponibles para definir las RV con la RM de 1ª etapa.

- Con el Esquema 1 de definición de RV, las posiciones de partida de 8 RV se pueden mantener pero no necesariamente usar siempre. Se pueden permitir solamente las posiciones de partida de RV originales que se encuentran dentro del intervalo ($0 \leq j \leq N_{cw} - 1$) en el almacenador temporal circular. Esto también se puede expresar usando la siguiente condición: solamente se usan RV-i con $N_r x\sigma <= N_r (12xi+\sigma) <= N_{cw} - 1$, provocando posiblemente menos de 8 RV en algunos casos. El último índice de RV i permitido es, de esta manera, suelo(($(N_{cw} - 1)/ N_r - \sigma)/12$).
- Con el Esquema 2 de definición de RV, todavía se necesitan (en todos los casos) 8 RV, entonces las posiciones de partida de RV pueden ser $N_r(Cxi+\sigma)$, i = 0, 1, ..., 7, donde C es el entero máximo de manera que $N_r (Cxi+\sigma) <= N_{cw} - 1$. En otras palabras, C = suelo(($(N_{cw} - 1)/ N_r - \sigma)/7$).

En la discusión anterior, C indica el número de columnas entre dos RV adyacentes, Y es el número de RV definidas.

Se puede aplicar una modificación de RV similar si otro número (por ejemplo, 4) de RV se define antes de la adaptación de tasa de 1ª etapa. Por ejemplo, si se usan 4 RV como la línea de base (cuando está ausente la RM de 1ª etapa), entonces están disponibles las siguientes opciones para definir las RV con la RM de 1ª etapa.

Sin la adaptación de tasa de 1ª etapa, RV-i comienza con la posición N_r x (N_cx3/4xi+σ) = N_r x (24xi+σ), i = 0, 1, 2, 3, en el almacenador temporal circular de relleno ficticio de formato de secuencia. Después de que se use la adaptación de tasa de 1ª etapa, el almacenador temporal circular contiene bits de posición j, donde $0 \leq j \leq N_{cw} - 1$.

- Con el Esquema 1 de definición de RV, se mantienen las posiciones de partida de 4 RV, entonces solamente se usan RV-i con N_r x2 <= N_rx (24xi+σ)<= N_{cw} -1, provocando posiblemente menos de 4 RV.

- Con el Esquema 2 de definición de RV, todavía se necesitan 4 RV, entonces las posiciones de partida de RV pueden ser N_rx (Cxi+2), i = 0, 1, 2, 3, donde C es el entero máximo de manera que N_r x (Cxi+σ)<= N_{cw} - 1.

La FIG. 12 es un diagrama de flujo que muestra la operación del transmisor de la FIG. 3. En particular, el flujo lógico de la FIG. 12 muestra los pasos para asignar versiones de redundancia al almacenador temporal circular 307 y la transmisión de bits posterior desde el almacenador temporal circular 307. El flujo lógico comienza como el paso 1201 en el que la circuitería lógica 305 recibe bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad. Estos bits se sacan desde el codificador 301. Como se trató anteriormente, los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad todos comprenden los bits ficticios. La circuitería lógica 305 intercala en bloques individualmente los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad usando el intercalador 309 (paso 1203). En el paso 1205, la circuitería lógica 305 entrelaza el primer bloque de bits de paridad con el segundo bloque de bits de paridad y antepone los bits sistemáticos intercalados a los bits de paridad entrelazados para crear el almacenador temporal circular 307. En el paso 1207, la circuitería lógica 305 define entonces las versiones de redundancia dentro de una fila particular del almacenador temporal circular. En una realización de la presente invención, la fila particular comprende la fila superior del almacenador temporal circular, por ejemplo, 8 ó 4 partes superiores de columnas del intercalador de subbloques. La circuitería lógica 305 recibe una versión de redundancia y un número de bits deseados (paso 1209) y saca a un transmisor el número de bits deseados comenzando en la posición de bit de RV (paso 1211).

La FIG. 13 es un diagrama de flujo que muestra la operación del transmisor de la FIG. 3. En particular, el flujo lógico de la FIG. 13 muestra los pasos para asignar versiones de redundancia a un almacenador temporal circular virtual y la transmisión de bits posterior desde el almacenador temporal circular virtual. El flujo lógico comienza como el paso 1301 en el que la circuitería lógica 305 recibe bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad. Como se trató anteriormente, los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad todos comprenden bits ficticios. La circuitería lógica 305 intercala en bloques individualmente los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad usando el intercalador 309 (paso 1303). En el paso 1305, la circuitería lógica 305 entrelaza el primer bloque de bits de paridad con el segundo bloque de bits de paridad y antepone los bits sistemáticos intercalados a los bits de paridad entrelazados. En el paso 1307, la circuitería lógica 305 define entonces las versiones de redundancia que corresponden a las posiciones de bits dentro de un flujo de datos y en el que todas las RV posibles están en posiciones $\lceil Kflujo/32 \rceil$(13xi+σ), i = 0, 1, ..., 7 y donde σ indica el índice de columna de la posición de partida de la primera RV (RV_0). El flujo de datos comprende los bits sistemáticos intercalados y un primer bloque entrelazado de bits de paridad intercalados y el segundo bloque de bits de paridad intercalados. La circuitería lógica 305 recibe una versión de redundancia y un número de bits deseados (paso 1309) y saca a un transmisor el número de bits deseados comenzando en la posición de bit de RV (paso 1311).

Aunque la invención se ha mostrado particularmente y descrito con referencia a una realización particular, se entenderá por los expertos en la técnica que se pueden hacer dentro de la misma diversos cambios en la forma y en los detalles sin apartarse del alcance de la invención.

En un ejemplo, aunque se explica el procedimiento, los bits ficticios pueden no ser añadidos a los flujos y más tarde eliminados del almacenador temporal circular, sino que el mismo efecto se realiza por otras vías tales como generar las direcciones adecuadamente.

En otro ejemplo, comenzar las RV desde una fila particular (por ejemplo, las partes superiores de las columnas) del almacenador temporal circular puede ser equivalente a comenzar las RV desde una fila particular en los intercaladores de subbloques para cada flujo mientras están presentes los bits ficticios. La asignación de los puntos de partida a su ubicación dentro del intercalador de subbloques puede ser más conveniente para cierta implementación, mientras que la definición de los puntos de partida de RV en el almacenador temporal circular puede ser más conveniente para otras implementaciones.

Se pretende que tales cambios queden dentro del alcance de las siguientes reivindicaciones.

REIVINDICACIONES

1. Un método para asignar versiones de redundancia a un almacenador temporal circular, el método que comprende los pasos de:

recibir bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad, en el que los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad todos comprenden bits ficticios;

intercalar en bloques individualmente los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad;

entrelazar el primer bloque de bits de paridad con el segundo bloque de bits de paridad para crear bits de paridad entrelazados;

anteponer los bits sistemáticos intercalados a los bits de paridad entrelazados para crear un almacenador temporal circular, caracterizado por:

borrar una o más de las columnas del almacenador temporal circular para formar un almacenador temporal circular acortado, cada columna que comprende primeros y segundos bits de paridad entrelazados;

definir versiones de redundancia para comenzar en una fila particular del almacenador temporal circular acortado;

recibir una versión de redundancia, RV y un número de bits deseado; y

sacar el número de bits deseado comenzando en la posición de bit de RV.

2. El método de la reivindicación 1 que además comprende el paso de definir versiones de redundancia para comenzar en las partes superiores de las columnas del almacenador temporal circular acortado.

3. El método de la reivindicación 2 en el que el paso de definición de versiones de redundancia para comenzar en las partes superiores de las columnas del almacenador temporal circular acortado comprende el paso de definición de versiones de redundancia para comenzar en las partes superiores de 8 columnas del almacenador temporal circular acortado.

4. El método de la reivindicación 2 en el que el paso de definición de versiones de redundancia para comenzar en las partes superiores de las columnas del almacenador temporal circular acortado comprende el paso de definición de las versiones de redundancia para comenzar en las partes superiores de 4 columnas del almacenador temporal circular acortado.

5. El método de la reivindicación 2 que además comprende el paso de definir una primera versión de redundancia para comenzar en la parte superior de una columna de orden σ, donde σ es el número de columnas de desplazamiento.

6. El método de la reivindicación 2 que además comprende el paso de perforación de bits sistemáticos definiendo la primera versión de redundancia para comenzar en una posición con un desplazamiento con respecto al principio del almacenador temporal circular acortado.

7. Un aparato que comprende:

un codificador (101) que saca bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad;

circuitería de adaptación de tasa (102) que recibe los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad, que intercala en bloques individualmente los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad, que entrelaza el primer bloque de bits de paridad con el segundo bloque de bits de paridad para crear bits de paridad entrelazados, que antepone los bits sistemáticos intercalados a los bits de paridad entrelazados para crear un almacenador temporal circular y que crea un almacenador temporal circular que tiene bits ficticios y en el que una fila particular del almacenador temporal circular contiene puntos de partida de las versiones de redundancia, RV, caracterizado por que la circuitería de adaptación de tasa borra una o más de las columnas del almacenador temporal circular para formar un almacenador temporal circular acortado, en el que cada columna comprende unos primeros y segundos bits de paridad entrelazados y en donde la circuitería de adaptación de tasa saca bits no ficticios del almacenador temporal circular acortado a un transmisor que comienza en una RV particular;

circuitería de transmisión (103) que recibe los bits no ficticios y que transmite los bits no ficticios.

8. El aparato de la reivindicación 7 en el que las partes superiores de 8 columnas del almacenador temporal circular acortado se definen como una posición de partida para una versión de redundancia (RV).

9. El aparato de la reivindicación 7 en el que las partes superiores de 4 columnas del almacenador temporal circular acortado se definen como una posición de partida para una versión de redundancia (RV).

10. El aparato de la reivindicación 7 en el que el almacenador temporal circular acortado comprende bits sistemáticos intercalados, un primer y segundo bloque de bits de paridad intercalados y entrelazados.

11. El aparato de la reivindicación 7 en el que la circuitería de adaptación de tasa inicia una primera versión de redundancia en una parte superior de la columna de orden σ, donde σ es un número de columnas de desplazamiento.

12. El aparato de la reivindicación 7 en el que la circuitería de adaptación de tasa define versiones de redundancia que comienza en las partes superiores de columnas del almacenador temporal circular acortado.

13. El aparato de la reivindicación 12 en el que la circuitería de adaptación de tasa perfora bits sistemáticos definiendo la primera versión de redundancia que comienza en una posición con un desplazamiento con respecto al principio del almacenador temporal circular acortado.

14. El método de la reivindicación 1, en el que el paso de sacar los bits que comienzan en la posición de RV incluye una envoltura alrededor del principio del almacenador temporal circular en un punto anterior que el final del almacenador temporal circular.

15. El método de la reivindicación 1, en el que el paso de borrar la una o más columnas comprende borrar columnas de la última columna del almacenador temporal circular.

16. El método de la reivindicación 1, en el que el paso de borrar la una o más columnas comprende un paso de determinación de un tamaño de almacenador temporal por segmento en base al suelo (N_{IR}/C), donde C es el número de segmentos para el bloque de Transporte, determinado por la regla de segmentación de bloque de código y en el que N_{IR} es el tamaño de almacenador temporal flexible total por proceso de Petición de Repetición Automática Híbrida.

17. El aparato de la reivindicación 7, en el que la circuitería de adaptación de tasa saca los bits que comienzan en una posición RV que incluye una envoltura alrededor del principio del almacenador temporal circular en un punto anterior que el final del almacenador temporal circular.

18. El aparato de la reivindicación 7, en el que la circuitería de adaptación de tasa que borra la una o más columnas comprende borrar columnas desde la última columna del almacenador temporal circular.

19. El aparato de la reivindicación 7, en el que la circuitería de adaptación de tasa determina un tamaño de almacenador temporal por segmento en base al suelo(N_{IR}/C), donde C es el número de segmentos para el bloque de Transporte, determinado por la regla de segmentación de bloque de código y en el que N_{IR} es el tamaño de almacenador temporal flexible total por proceso de Petición de Repetición Automática Híbrida.

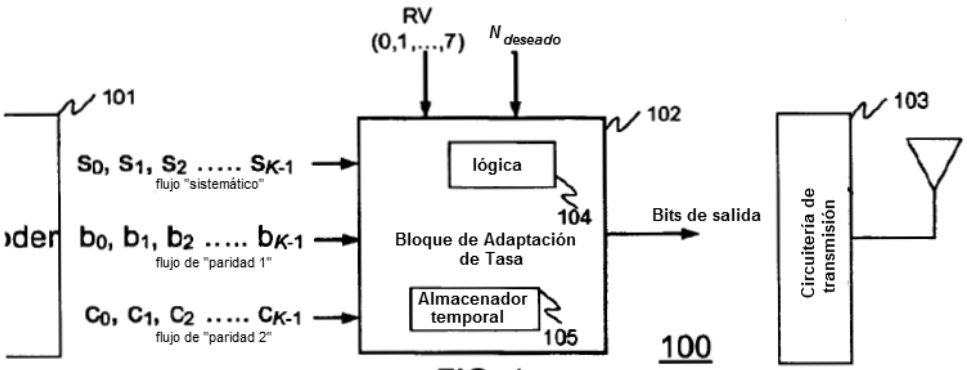


FIG. 1

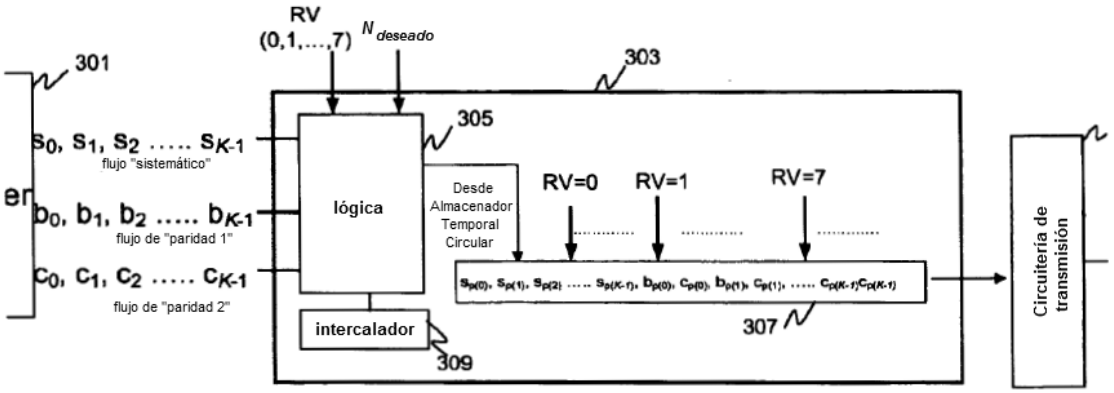


FIG. 3
300

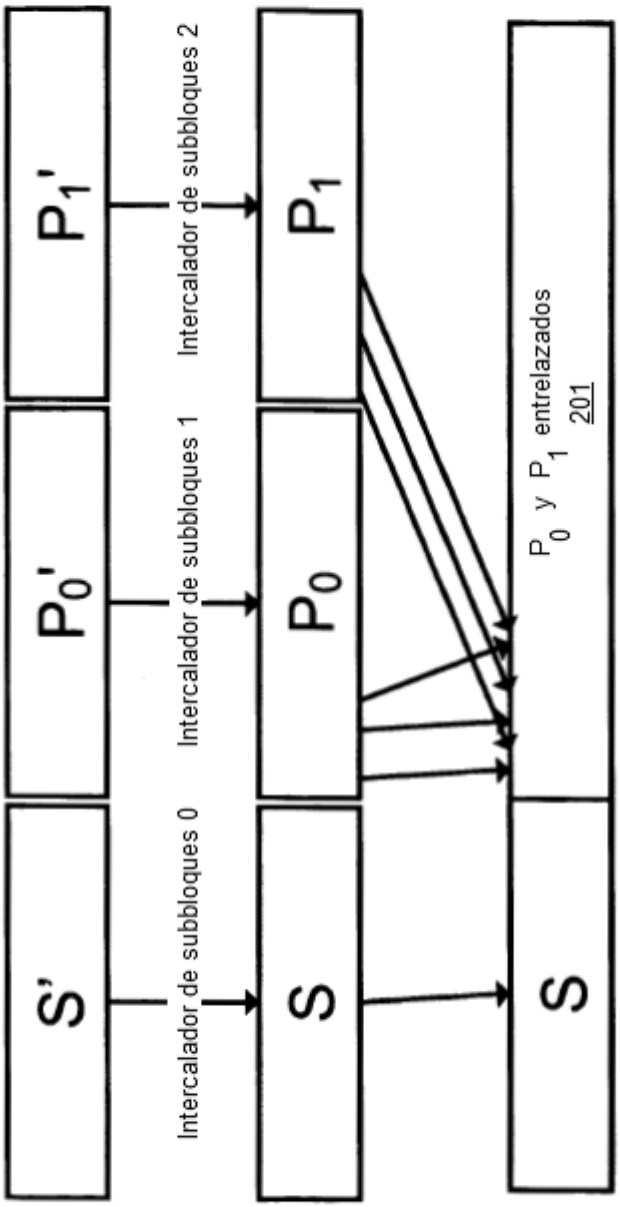

S'
P0'
P1'
Intercalador de subbloques 0
Intercalador de subbloques 1
Intercalador de subbloques 2
S
P0
P1
S
P0 y P1 entrelazados
201

FIG. 2

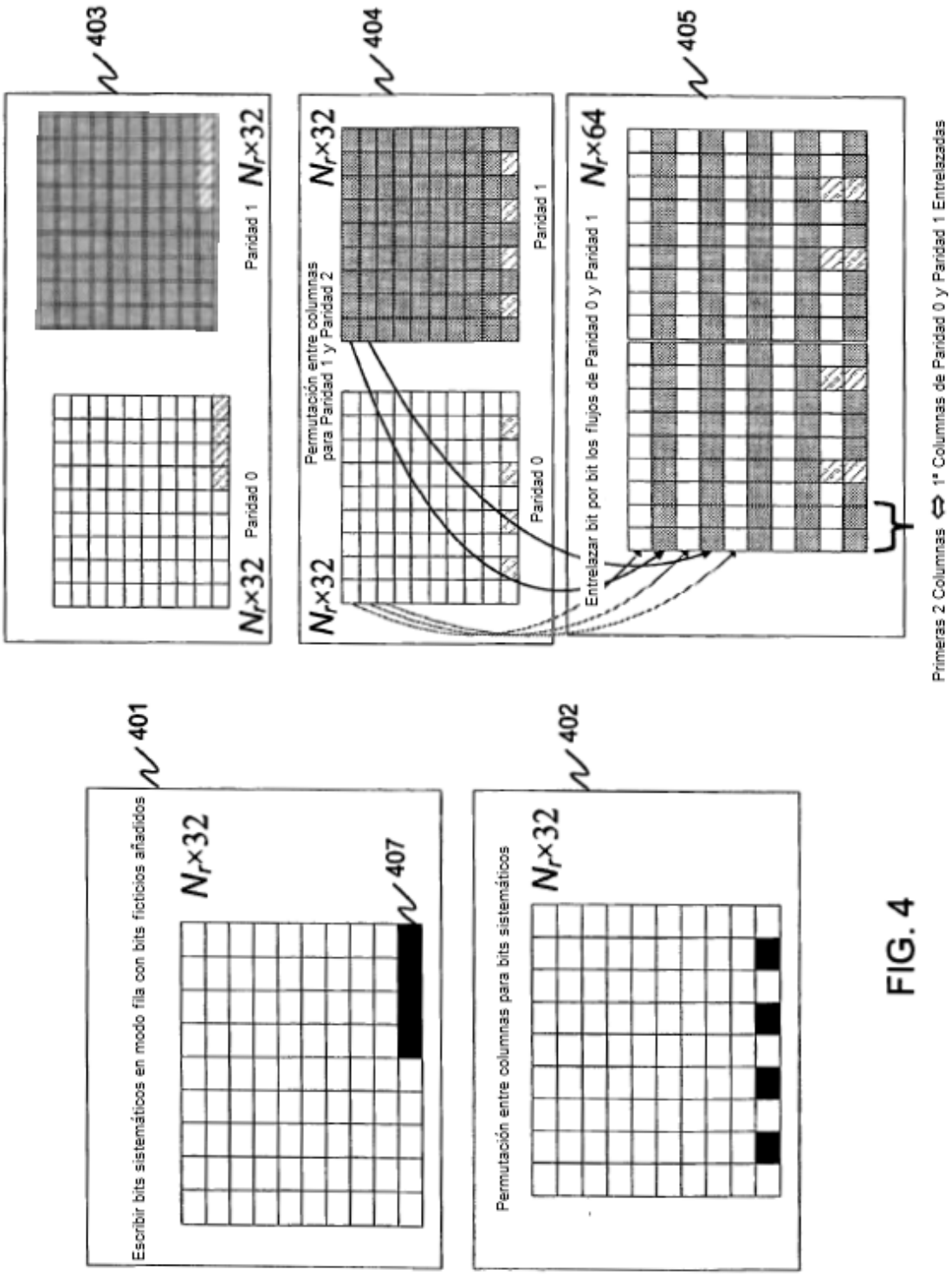

401
Escribir bits sistemáticos en modo fila con bits ficticios añadidos
N_r×32
407
402
Permutación entre columnas para bits sistemáticos
N_r×32
403
Paridad 1
N_r×32
Paridad 0
N_r×32
404
Permutación entre columnas para Paridad 1 y Paridad 2
N_r×32
Paridad 1
Paridad 0
N_r×32
405
Entrelazar bit por bit los flujos de Paridad 0 y Paridad 1
N_r×64
Primeras 2 Columnas ⇔ 1ª Columnas de Paridad 0 y Paridad 1 Entrelazadas

FIG. 4

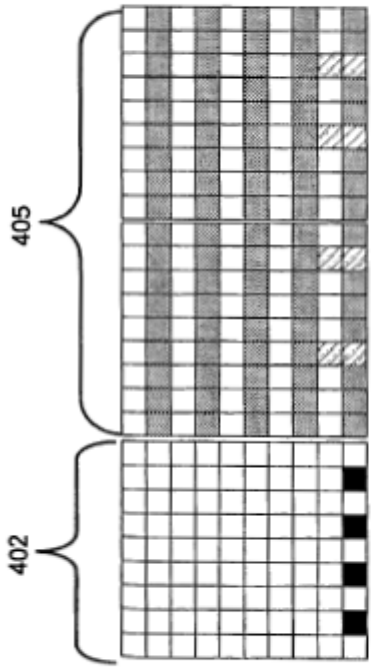


FIG. 5

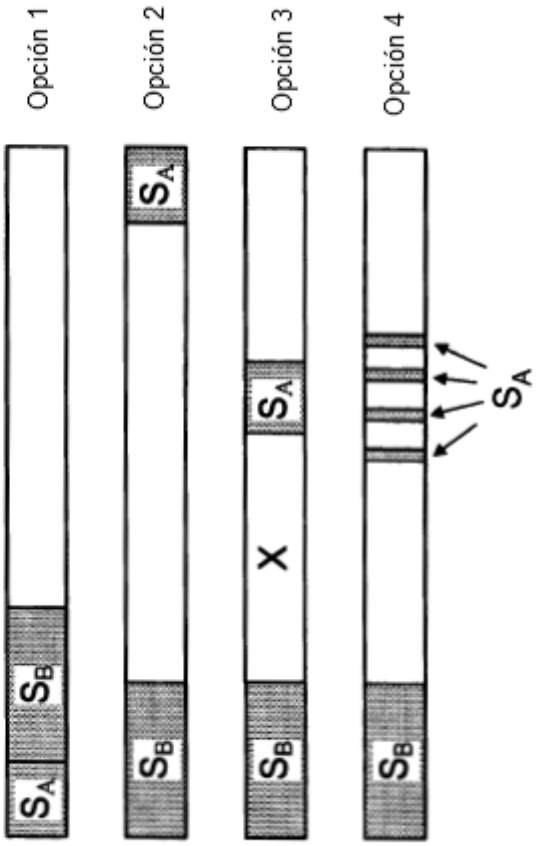
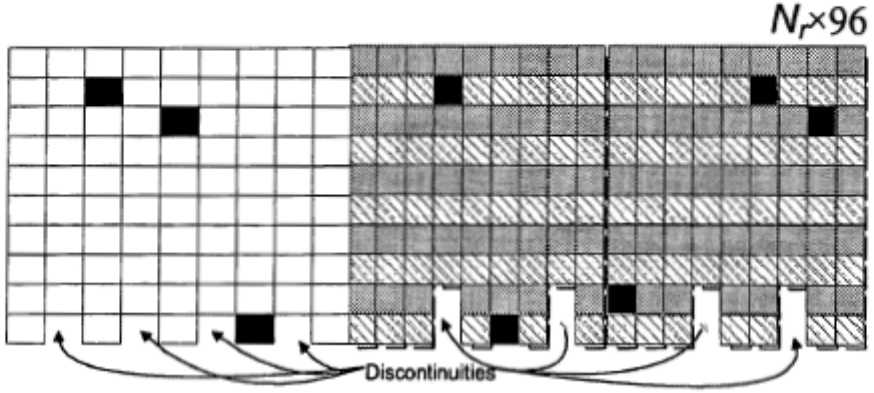


FIG. 10

FIG. 6

Definición de RV con CB que no tiene bits ficticios



Las RV pueden comenzar en cualquier sitio

- representa el comienzo de una RV con RV distribuidas por igual
- representa bits del segundo flujo de paridad
- representa bits del primer flujo de paridad
- representa bits del flujo sistemático

- CB formado leyendo bits columna por columna
- Las discontinuidades aparecen siempre que fueron descartados bits ficticios
  - Falta de expresión de forma cerrada
  - Las discontinuidades varían con tamaño de bloque de Código K, tamaño de Rectángulo K_{rec} y Permutación de Columna

ES 2 610 818 T3

Definición de RV con CB que tiene bits ficticios

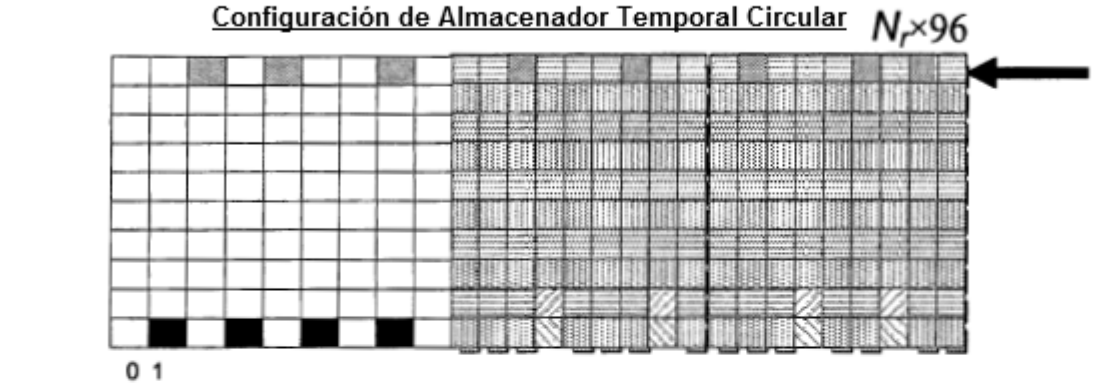


FIG. 7

- Definición de RV sin memoria es clave para comenzar eficientemente desde cualquier RV
- CB se puede formar leyendo bits columna por columna
- Bits ficticios descargados mientras que se lee
- Cada RV comienza en la parte superior de una columna {2, 14, 26, 38, 50, 62, 74, 86}
  - RV(i) comienza en una posición de bit $pos_{vCB}(i)=\lceil K/32 \rceil \times (12i+2) = \lfloor (6 \times i+1) \times K_{rect}/16 \rfloor$

ES 2 610 818 T3

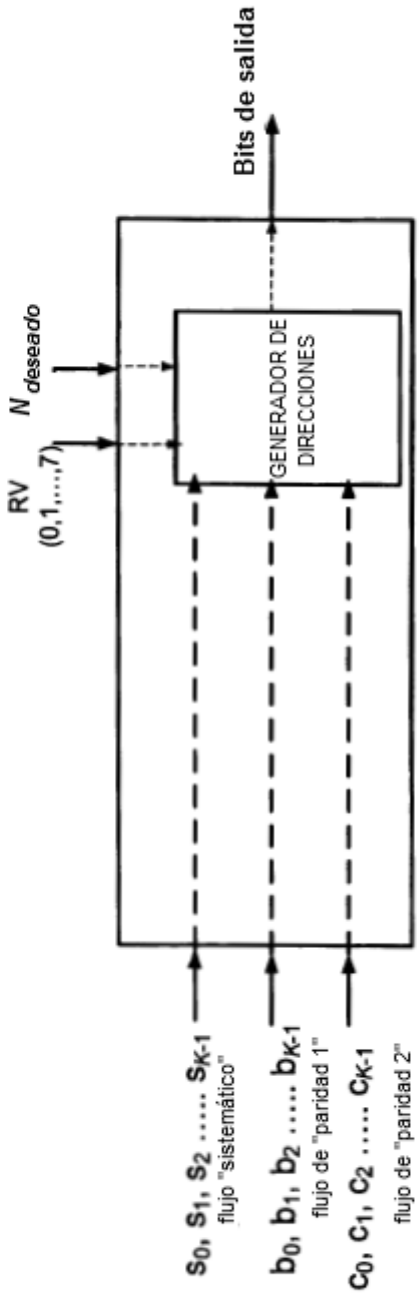

Bits de salida
GENERADOR DE DIRECCIONES
N deseado
RV (0,1,...,7)
s0, s1, s2 ..... sK-1
flujo "sistemático"
b0, b1, b2 ..... bK-1
flujo de "paridad 1"
c0, c1, c2 ..... cK-1
flujo de "paridad 2"

FIG. 8

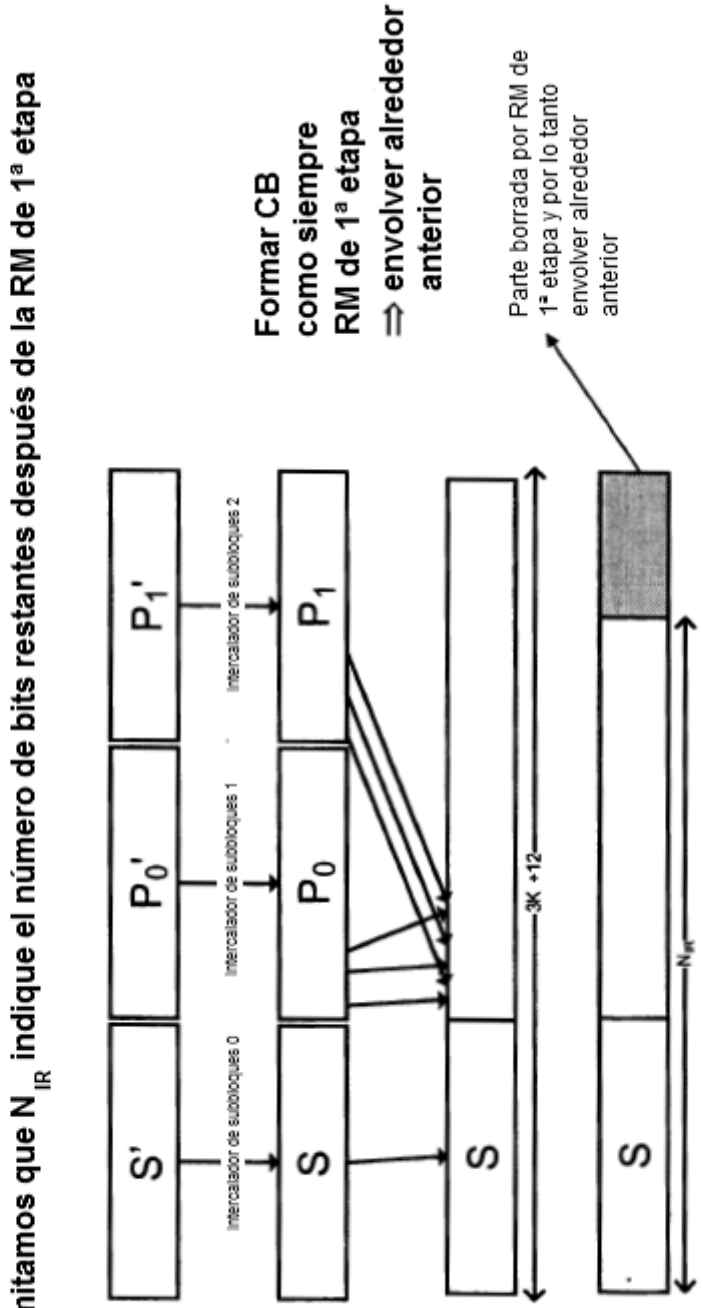

RM de 1ª Etapa con CB
Permitamos que N_{IR} indique el número de bits restantes después de la RM de 1ª etapa
S'
P_0'
P_1'
Intercalador de subbloques 0
Intercalador de subbloques 1
Intercalador de subbloques 2
S
P_0
P_1
S
S
3K +12
N_{IR}
Formar CB
como siempre
RM de 1ª etapa
⇒ envolver alrededor
anterior
Parte borrada por RM de
1ª etapa y por lo tanto
envolver alrededor
anterior

FIG. 9

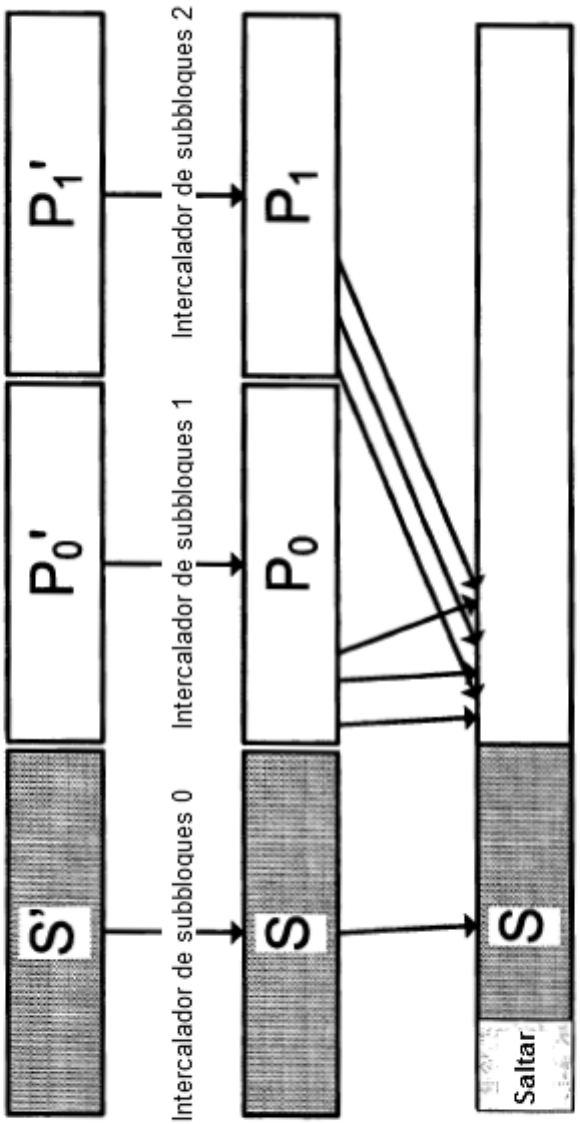

S'
P0'
P1'
Intercalador de subbloques 0
Intercalador de subbloques 1
Intercalador de subbloques 2
S
P0
P1
S
Saltar

FIG. 11

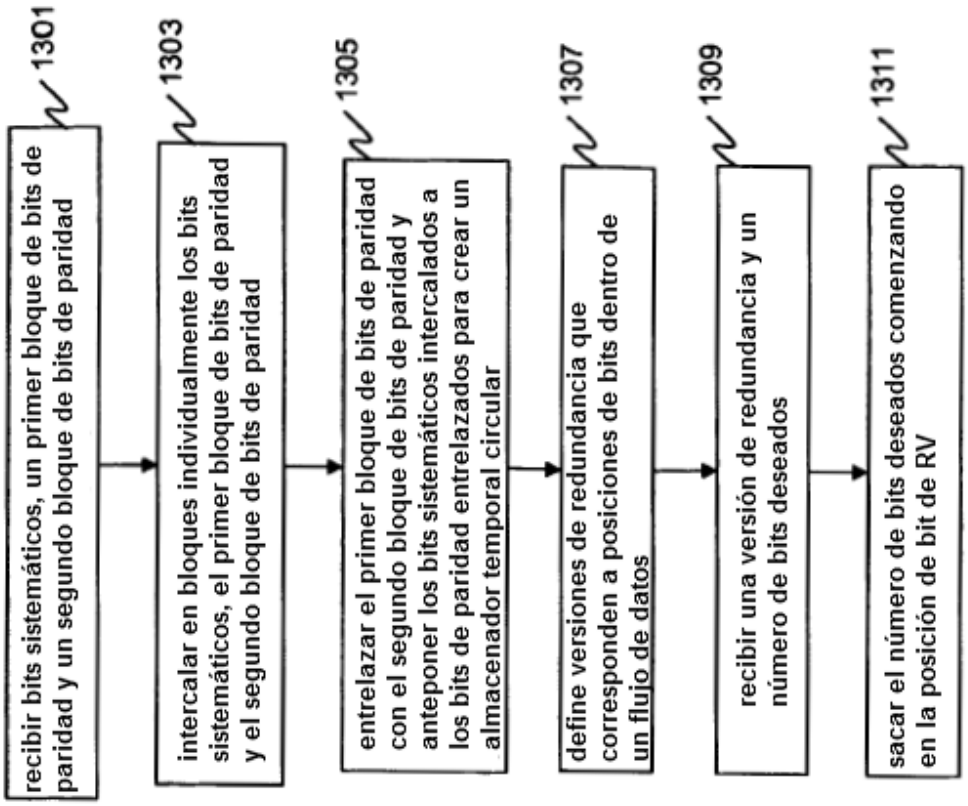

recibir bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad
1301
intercalar en bloques individualmente los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad
1303
entrelazar el primer bloque de bits de paridad con el segundo bloque de bits de paridad y anteponer los bits sistemáticos intercalados a los bits de paridad entrelazados para crear un almacenador temporal circular
1305
define versiones de redundancia que corresponden a posiciones de bits dentro de un flujo de datos
1307
recibir una versión de redundancia y un número de bits deseados
1309
sacar el número de bits deseados comenzando en la posición de bit de RV
1311

FIG. 13

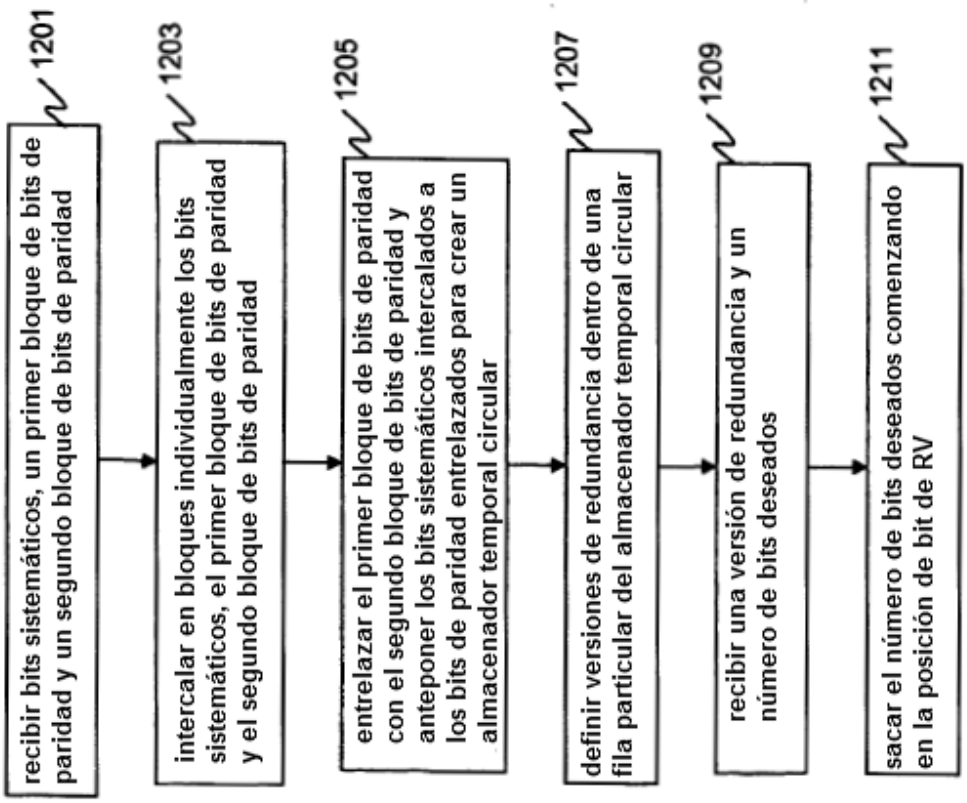

recibir bits sistemáticos, un primer bloque de bits de paridad y un segundo bloque de bits de paridad
1201
intercalar en bloques individualmente los bits sistemáticos, el primer bloque de bits de paridad y el segundo bloque de bits de paridad
1203
entrelazar el primer bloque de bits de paridad con el segundo bloque de bits de paridad y anteponer los bits sistemáticos intercalados a los bits de paridad entrelazados para crear un almacenador temporal circular
1205
definir versiones de redundancia dentro de una fila particular del almacenador temporal circular
1207
recibir una versión de redundancia y un número de bits deseados
1209
sacar el número de bits deseados comenzando en la posición de bit de RV
1211

FIG. 12