(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 566 917**

(51) Int. CI.:

H03M 7/40 (2006.01)
H04N 19/13 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **12.01.2012 E 14160511 (3)**

(97) Fecha y número de publicación de la concesión europea: **23.03.2016 EP 2768145**

(54) Título: **Esquema de codificación y descodificación por entropía**

(30) Prioridad:

14.01.2011 US 201161432884 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.04.2016

(73) Titular/es:

GE VIDEO COMPRESSION, LLC (100.0%)
8 Southwoods Boulevard
Albany, NY 12211, US

(72) Inventor/es:

MARPE, DETLEV;
NGUYEN, TUNG;
SCHWARZ, HEIKO y
WIEGAND, THOMAS

(74) Agente/Representante:

ARIZTI ACHA, Monica

Esquema de codificación y descodificación por entropía

DESCRIPCIÓN

La presente invención se refiere a una codificación y descodificación por entropía y se puede usar en aplicaciones tales como, por ejemplo, compresión de vídeo y de audio.

La codificación por entropía, en general, se puede considerar como la forma más genérica de compresión de datos sin pérdida. La compresión sin pérdida tiene por objeto representar unos datos discretos con menos bits de los necesarios para la representación de datos original pero sin pérdida alguna de información. Los datos discretos se pueden dar en forma de texto, gráficos, imágenes, vídeo, audio, habla, fax, datos médicos, datos meteorológicos, datos financieros, o cualquier otra forma de datos digitales.

En la codificación por entropía, se descuidan las características de alto nivel específicas del origen de datos discretos subyacente a menudo. En consecuencia, se considera que cualquier origen de datos se da como una secuencia de símbolos de origen que adopta valores en un alfabeto *m*-ario dado y que se caracteriza por una distribución de probabilidad (discreta) correspondiente $\{p_1, \ldots, p_m\}$. En estos escenarios abstractos, el límite inferior de cualquier método de codificación por entropía en términos de la longitud de palabra de código esperada en bits por símbolos viene dado por la entropía

$$H = -\sum_{i=1}^{m} p_i \log_2 p_i \,. \qquad \text{(A1)}$$

Los códigos de Huffman y los códigos aritméticos son ejemplos bien conocidos de códigos prácticos capaces de aproximarse al límite de entropía (en un cierto sentido). Para una distribución de probabilidad fija, los códigos de Huffman son relativamente sencillos de construir. La propiedad más atractiva de los códigos de Huffman es que su implementación puede ser realizada de forma eficiente mediante el uso de tablas de códigos de longitud variable (VLC, *variable length code*). No obstante, cuando se abordan unas estadísticas de origen variables en el tiempo, es decir, unas probabilidades de símbolo cambiantes, la adaptación del código de Huffman y sus tablas de VLC correspondientes es bastante exigente, tanto en términos de la complejidad algorítmica así como en términos de los costes de implementación. Asimismo, en el caso de tener un valor de alfabeto dominante con $p_k > 0{,}5$, la redundancia del código de Huffman correspondiente (sin usar ampliación de alfabeto alguna tal como la codificación de longitud de series) puede ser bastante sustancial. Otra deficiencia de los códigos de Huffman viene dada por el hecho de que en el caso de tratar con un modelado de probabilidad de orden superior, se pueden requerir múltiples conjuntos de tablas de VLC. La codificación aritmética, por otro lado, a pesar de ser sustancialmente más compleja que VLC, ofrece la ventaja de una manipulación más consistente y adecuada cuando se hace frente a un modelado de probabilidad adaptativo y de orden superior así como con el caso de unas distribuciones de probabilidad muy sesgadas. En la práctica, esta característica básicamente resulta del hecho de que la codificación aritmética proporciona un mecanismo, por lo menos conceptualmente, para poner en correspondencia cualquier valor dado de estimada de probabilidad de una forma más o menos directa con una porción de la palabra de código resultante. Estando provista con una interconexión de este tipo, la codificación aritmética permite una separación limpia entre las tareas de modelado de probabilidad y de estimación de probabilidad, por un lado, y la codificación por entropía real, es decir, la puesta en correspondencia de unos símbolos con palabras de código, por otro lado.

Una alternativa a la codificación aritmética y la codificación de VLC es la codificación de PIPE. Para ser más precisos, en la codificación de PIPE, el intervalo unitario se subdivide en particiones para dar un pequeño conjunto de intervalos de probabilidad disjuntos para canalizar el procesamiento de codificación a lo largo de las estimadas de probabilidad de variables de símbolo aleatorias. De acuerdo con esta subdivisión en particiones, una secuencia de entrada de símbolos de origen discretos con tamaños de alfabeto arbitrarios se puede poner en correspondencia con una secuencia de símbolos alfabéticos y cada uno de los símbolos alfabéticos se asigna a un intervalo de probabilidad particular que es, a su vez, codificado por un proceso de codificación por entropía especialmente dedicado. Con cada uno de los intervalos estando representado por una probabilidad fija, el proceso de codificación de entropía de subdivisión en particiones de intervalos de probabilidad (PIPE, *probability interval partitioning entropy*) se puede basar en el diseño y la aplicación de códigos de longitud de variable a variable. El modelado de probabilidad puede o bien ser fijo o bien ser adaptativo. No obstante, a pesar de que la codificación de PIPE es significativamente menos compleja que la codificación aritmética, esta sigue teniendo una complejidad más alta que la codificación de VLC.

Por lo tanto, sería favorable tener a mano un esquema de codificación por entropía el cual posibilite lograr una mejor compensación recíproca entre la complejidad de codificación por un lado y la eficiencia de compresión por otro lado, incluso cuando se compara con la codificación de PIPE la cual ya combina ventajas tanto de la codificación aritmética como de la codificación de VLC.

Además, en general, sería favorable tener a mano un esquema de codificación por entropía el cual posibilite lograr una mejor eficiencia de compresión de por sí, con una complejidad de codificación moderada.
El documento WO 2008/129021 A2 se refiere a la compresión escalonable de secuencias de estructura reticular en 3D temporalmente consistentes. En lo que respecta a la cuantificación y la codificación por entropía, el documento describe que los errores de predicción de los vectores de estructura reticular se comprimen componente a componente. En particular, las componentes se ponen en correspondencia con la cantidad de números enteros, es decir con signo, y un máximo para la cantidad, es decir imax, se usa para definir un intervalo dentro de la cantidad de números enteros para el cual las componentes que caen dentro de este intervalo se codifican por entropía. La cantidad residual, es decir la distancia al extremo más cercano del intervalo, se codifica mediante el uso de códigos de Golomb.

Un objeto de la presente invención es la provisión de un concepto de codificación por entropía el cual satisface la demanda que se ha identificado en lo que antecede, es decir permite conseguir una mejor compensación recíproca entre la complejidad de codificación por un lado y la eficiencia de compresión por otro lado.

Este objeto se logra mediante la materia objeto de las reivindicaciones independientes.

La presente invención se basa en la idea de que la descomposición de un intervalo de valores de los elementos de sintaxis respectivos en una secuencia de n particiones con codificación de las componentes de los valores de elemento de sintaxis z encontrándose dentro de las particiones respectivas por separado con por lo menos una mediante codificación de VLC y con por lo menos una mediante codificación aritmética. Por consiguiente, de acuerdo con realizaciones de la presente invención, elementos de sintaxis se descomponen en un número respectivo n de símbolos de origen s_i con i = 1 ... n, el número respectivo n de símbolos de origen dependiendo de en cual de una secuencia de n particiones (140_{1-3}) en las cuales está subdividido un intervalo de valores de los elementos de sintaxis respectivos, cae un valor z de los elementos de sintaxis respectivos, de tal modo que una suma de valores del número respectivo de símbolos de origen s_i da z, y, si n > 1, para todo i = 1 ... n-1, el valor de s_i se corresponde con un intervalo de la i-ésima partición.

Aspectos preferidos de la presente invención son el objeto de las reivindicaciones dependientes incluidas.

Realizaciones preferidas de la presente invención se describen en lo sucesivo con respecto a las figuras. Estas realizaciones representan ejemplos, en la medida en la que estas no usen codificación aritmética junto a la codificación de VLC. Entre las figuras,

la figura 1a muestra un diagrama de bloques de un aparato de codificación por entropía;

la figura 1b muestra un diagrama esquemático que ilustra una posible descomposición de elementos de sintaxis en símbolos de origen;

la figura 1c muestra un diagrama de flujo que ilustra un posible modo de funcionamiento de la unidad de descomposición de la figura 1a en la descomposición de elementos de sintaxis en símbolos de origen;

la figura 2a muestra un diagrama de bloques de un aparato de descodificación por entropía;

la figura 2b muestra un diagrama de flujo que ilustra un posible modo de funcionamiento de la unidad de composición de la figura 2a en la composición de elementos de sintaxis a partir de símbolos de origen;

la figura 3 muestra un diagrama de bloques de un codificador de PIPE de acuerdo con una realización de comparación el cual se puede usar en la figura 1;

la figura 4 muestra un diagrama de bloques de un descodificador de PIPE adecuado para descodificar un tren de bits generado por el codificador de PIPE de la figura 3, de acuerdo con una realización de comparación, el cual se puede usar en la figura 2;

la figura 5 muestra un diagrama esquemático que ilustra un paquete de datos con trenes de bits parciales multiplexados;

la figura 6 muestra un diagrama esquemático que ilustra un paquete de datos con una segmentación alternativa que usa unos segmentos de tamaño fijo;

la figura 7 muestra un diagrama de bloques de un codificador de PIPE usando una intercalación de trenes de bits parciales;

la figura 8 muestra un diagrama esquemático que ilustra ejemplos para el estatus de una memoria intermedia de

palabras de código en el lado de codificador de la figura 7;

la figura 9 muestra un diagrama de bloques de un descodificador de PIPE que usa una intercalación de trenes de bits parciales;

la figura 10 muestra un diagrama de bloques de un descodificador de PIPE usando una intercalación de palabras de código que usa un único conjunto de palabras de código;

la figura 11 muestra un diagrama de bloques de un codificador de PIPE usando una intercalación de secuencias de bits de longitud fija;

la figura 12 muestra un diagrama esquemático que ilustra ejemplos para el estatus de una memoria intermedia de bits global en el lado de codificador de la figura 11;

la figura 13 muestra un diagrama de bloques de un descodificador de PIPE usando una intercalación de secuencias de bits de longitud fija;

la figura 14 muestra una gráfica para ilustrar una discretización de intervalo de probabilidad óptima en K = 4 intervalos suponiendo una distribución de probabilidad uniforme en (0, 0,5];

la figura 15 muestra un diagrama esquemático que ilustra un árbol de eventos binarios para una probabilidad de LPB de p = 0,38 y un código de longitud variable asociado obtenido mediante el algoritmo de Huffman;

la figura 16 muestra una gráfica a partir de la cual se puede reunir el aumento relativo de tasa de bits ρ(p, C) para unos códigos óptimos C dado un número máximo de entradas de tabla L_m;

la figura 17 muestra una gráfica que ilustra el aumento de tasa para la subdivisión en particiones de intervalos de probabilidad teóricamente óptima en K = 12 intervalos y un diseño real con códigos de V2V con un número máximo de L_m = 65 entradas de tabla;

la figura 18 muestra un diagrama esquemático que ilustra un ejemplo para la conversión de un árbol de elección ternario en un árbol de elección binario completo;

la figura 19 muestra un diagrama de bloques de un sistema que comprende un codificador (parte izquierda) y un descodificador (parte derecha);

la figura 20 muestra un diagrama de bloques de un aparato de codificación por entropía;

la figura 21 muestra un diagrama de bloques de un aparato de descodificación por entropía;

la figura 22 muestra un diagrama de bloques de un aparato de codificación por entropía;

la figura 23 muestra un diagrama esquemático que ilustra ejemplos para el estatus de una memoria intermedia de bits global en el lado de codificador de la figura 22;

la figura 24 muestra un diagrama de bloques de un aparato de descodificación por entropía.

Antes de que se describan varias realizaciones de la presente solicitud en lo sucesivo con respecto a las figuras, se hace notar que se usan signos de referencia iguales por la totalidad de las figuras con el fin de indicar unos elementos iguales o equivalentes en estas figuras, y la descripción de estos elementos presentada con cualquiera de las figuras previas también será de aplicación a cualquiera de las siguientes figuras siempre que la descripción anterior no entre en conflicto con la descripción de las figuras actuales.

La figura 1a muestra un aparato de codificación por entropía. El aparato comprende un subdivisor 100, un codificador de VLC 102 y un codificador de PIPE 104.

El subdivisor 100 está configurado para subdividir una secuencia de símbolos de origen 106 en una primera subsecuencia 108 de símbolos de origen y una segunda subsecuencia 110 de símbolos de origen. El codificador de VLC 102 tiene una entrada del mismo que está conectada con una primera salida del subdivisor 100 y está configurado para convertir por símbolos los símbolos de origen de la primera subsecuencia 108 en unas palabras de código formando un primer tren de bits 112. El codificador de VLC 102 puede comprender una tabla de consulta y usar, de forma individual, los símbolos de origen como un índice con el fin de consulta, por símbolos de origen, una palabra de código respectiva en la tabla de consulta. El codificador de VLC emite esta última palabra de código, y prosigue con el siguiente símbolo de origen en la subsecuencia 110 con el fin de emitir una secuencia de palabras de código en la que cada palabra de código está asociada con exactamente uno de los símbolos de origen dentro de

la subsecuencia 110. Las palabras de código pueden tener diferentes longitudes y se pueden definir de tal modo que ninguna palabra de código forma un prefijo con cualquiera de las otras palabras de código. Adicionalmente, la tabla de consulta puede ser estática.

El codificador de PIPE 104 tiene una entrada del mismo que está conectada con una segunda salida del subdivisor 100 y está configurado para codificar la segunda subsecuencia 110 de símbolos de origen, que se representa en la forma de una secuencia de símbolos alfabéticos, y comprende una unidad de asignación 114 que está configurada para asignar una medida para una estimada de una distribución de probabilidad entre los posibles valores que pueden asumir los símbolos alfabéticos respectivos, a cada símbolo alfabético de la secuencia de símbolos alfabéticos sobre la base de la información contenida dentro de símbolos alfabéticos previos de la secuencia de símbolos alfabéticos, una pluralidad de codificadores por entropía 116 cada uno de los cuales está configurado para convertir los símbolos alfabéticos reenviados al codificador por entropía respectivo en un segundo tren de bits 118 respectivo, y un selector 120 que está configurado para reenviar cada símbolo alfabético de la segunda subsecuencia 110 a uno seleccionado de la pluralidad de codificadores por entropía 116, dependiendo la selección de la medida que se ha mencionado en lo que antecede para la estimada de la distribución de probabilidad asignada al símbolo alfabético respectivo. La asociación entre símbolos de origen y símbolos alfabéticos puede ser de tal modo que cada símbolo alfabético se asocia de forma única con exactamente un símbolo de origen de la subsecuencia 110 con el fin de representar, junto con símbolos alfabéticos posiblemente adicionales de la secuencia de símbolos alfabéticos los cuales pueden seguir inmediatamente unos a otros, este mismo símbolo de origen.

Tal como se describe con más detalle en lo sucesivo, la secuencia 106 de símbolos de origen puede ser una secuencia de elementos de sintaxis de un tren de bits que se puede analizar sintácticamente. El tren de bits que se puede analizar sintácticamente puede representar, por ejemplo, contenido de vídeo y / o de audio de una forma escalonable o no escalonable con los elementos de sintaxis representando, por ejemplo, niveles de coeficiente de transformada, vectores de movimiento, índices de referencia de imágenes de movimiento, factores de escala, valores de energía de envolvente de audio o similares. Los elementos de sintaxis pueden, en particular, ser de diferente tipo o categoría con elementos de sintaxis del mismo tipo, por ejemplo, teniendo el mismo significado dentro del tren de bits que se puede analizar sintácticamente pero con respecto a diferentes porciones del mismo, tal como diferentes imágenes, diferentes macrobloques, diferentes componentes espectrales o similares, mientras que elementos de sintaxis de diferente tipo pueden tener un significado diferente dentro del tren de bits, tal como un vector de movimiento tiene un significado diferente al de un elemento de sintaxis que representa un nivel de coeficiente de transformada que representa el residuo de predicción de movimiento.

El subdivisor 100 se puede configurar para realizar la subdivisión dependiendo del tipo de los elementos de sintaxis. Es decir, el subdivisor 100 puede reenviar elementos de sintaxis de un primer grupo de tipos a la primera subsecuencia 108 y reenviar elementos de sintaxis de un segundo grupo de tipos distinto del primer grupo, a la segunda subsecuencia 110. La subdivisión que es realizada por el subdivisor 100 se puede diseñar de tal modo que las estadísticas de símbolos de los elementos de sintaxis dentro de la subsecuencia 108 son adecuadas para someterse a codificación de VLC por el codificador de VLC 102, es decir da como resultado, de hecho, casi una mínima entropía posible a pesar del uso de la codificación de VLC y su restricción con respecto a su idoneidad para determinadas estadísticas de símbolos tal como se bosqueja en la porción introductoria de la memoria descriptiva de la presente solicitud. Por otro lado, el subdivisor 100 puede reenviar todos los otros elementos de sintaxis a la segunda subsecuencia 110 de tal modo que estos elementos de sintaxis que tienen unas estadísticas de símbolos que no son adecuadas para la codificación de VLC, son codificados por el codificador de PIPE más complejo, pero más eficiente - en términos de la relación de compresión - 104.

Tal como es también el caso con el mayor detalle con respecto a las siguientes figuras, el codificador de PIPE 104 puede comprender un simbolizador 122 que está configurado para poner en correspondencia de forma individual cada elemento de sintaxis de la segunda subsecuencia 110 con una secuencia parcial respectiva de símbolos alfabéticos, formando en conjunto la secuencia 124 que se ha mencionado en lo que antecede de símbolos alfabéticos. Dicho de otra forma, el simbolizador 122 puede no estar presente si, por ejemplo, los símbolos de origen de la subsecuencia 110 ya están representados como secuencias parciales respectivas de símbolos alfabéticos. El simbolizador 122 es, por ejemplo, ventajoso en el caso de que los símbolos de origen dentro de la subsecuencia 110 sean de diferentes alfabetos y, en especial, alfabetos que tienen diferentes números de símbolos alfabéticos posibles. En concreto, en este caso, el simbolizador 122 puede armonizar los alfabetos de los símbolos que llegan al interior del subtren 110. El simbolizador 122 se puede realizar, por ejemplo, como una unidad de binarización que está configurada para binarizar los símbolos que llegan dentro de la subsecuencia 110.

Tal como se ha mencionado antes, los elementos de sintaxis pueden ser de diferente tipo. Esto también puede ser cierto para los elementos de sintaxis en el interior del subtren 110. Entonces, el simbolizador 122 se puede configurar para realizar la puesta en correspondencia individual de los elementos de sintaxis de la subsecuencia 110 usando un esquema de puesta en correspondencia de simbolización, tal como un esquema de binarización, diferente para elementos de sintaxis de diferente tipo. En la siguiente descripción se presentan ejemplos para esquemas de binarización específicos, tal como un esquema de binarización unaria, un esquema de binarización de

Golomb Exponencial de orden 0 o de orden 1, por ejemplo, o un esquema de binarización unaria truncada, un esquema de binarización de orden 0 de Golomb Exponencial truncada y reordenada o un esquema de binarización no sistemática.

Por consiguiente, los codificadores por entropía 116 se podrían configurar para operar sobre un alfabeto binario. Por último, se debería hacer notar que el simbolizador 122 se puede considerar como que es parte del codificador de PIPE 104 en sí tal como se muestra en la figura 1a. Como alternativa, no obstante, la unidad de binarización se puede considerar como que es externa con respecto al codificador de PIPE.
De forma similar a lo que se ha hecho indicado anteriormente, se debería hacer notar que la unidad de asignación 114, a pesar de que se muestra que está conectado en serie entre el simbolizador 122 y el selector 120, como alternativa se puede considerar como que está conectada entre una salida del simbolizador 124 y una primera entrada del selector 120, con una salida de la unidad de asignación 114 estando conectada con otra entrada del selector 120 tal como se describe posteriormente con respecto a la figura 3. En efecto, la unidad de asignación 114 acompaña cada símbolo alfabético con la medida que se ha mencionado en lo que antecede para una estimación de la distribución de probabilidad.

En lo que respecta a la salida del aparato de codificación por entropía de la figura 1a, la misma está compuesta por el primer tren de bits 112 que es emitido por el codificador de VLC 102 y la pluralidad de segundos trenes de bits 118 que son emitidos por la pluralidad de codificadores por entropía 116. Tal como se describe adicionalmente en lo sucesivo, la totalidad de estos trenes de bits se pueden transmitir en paralelo. Como alternativa, los mismos se pueden intercalar para dar un tren de bits común 126 mediante el uso de un intercalador 128. Las figuras 22 a 24 muestran ejemplos con tal intercalación de trenes de bits. Tal como se muestra adicionalmente en la figura 1, el codificador de PIPE 104 en sí puede comprender su propio intercalador 130 con el fin de intercalar la pluralidad de segundos trenes de bits 118 en un tren de bits codificado de PIPE común 132. A partir de la descripción de las figuras 5 a 13 se pueden obtener posibilidades para tal intercalador 130. El tren de bits 132 y el tren de bits 112 pueden representar, en una configuración en paralelo, la salida del aparato de codificación por entropía de la figura 1a. Como alternativa, otro intercalador 134 puede intercalar ambos trenes de bits, caso en el cual, el intercalador 130 y 134 formaría dos fases de un intercalador de dos fases 128.

Tal como se ha descrito en lo que antecede, el subdivisor 100 puede realizar la subdivisión por elementos de sintaxis, es decir los símbolos de origen sobre los que opera el subdivisor 100 pueden ser elementos de sintaxis completos o, hablando de forma alternativa, el subdivisor 100 puede operar en unidades de elementos de sintaxis.

No obstante, el aparato de codificación por entropía de la figura 1a puede comprender la unidad de descomposición 136 con el fin de descomponer elementos de sintaxis dentro de un tren de bits que se puede analizar sintácticamente 138 de forma individual en uno o más de los símbolos de origen de la secuencia de símbolos de origen 106 que entran en el subdivisor 100. En particular, la unidad de descomposición 136 se puede configurar para convertir la secuencia 138 de elementos de sintaxis en la secuencia 106 de símbolos de origen mediante la descomposición de forma individual de cada elemento de sintaxis en un número entero respectivo de símbolos de origen. El número entero puede variar entre los elementos de sintaxis. En particular, algunos de los elementos de sintaxis incluso pueden ser dejados sin cambios por la unidad de descomposición 136, mientras que otros elementos de sintaxis se descomponen en exactamente dos, o por lo menos dos, símbolos de origen. El subdivisor 100 se puede configurar para reenviar uno de los símbolos de origen de tales elementos de sintaxis descompuestos a la primera subsecuencia 108 de símbolos de origen y otro de los símbolos de origen del mismo elemento de sintaxis descompuesto a la segunda subsecuencia 110 de símbolos de origen. Tal como se ha mencionado en lo que antecede, los elementos de sintaxis en el interior del tren de bits 138 pueden ser de diferente tipo, y la unidad de descomposición 136 se puede configurar para realizar la descomposición individual dependiendo del tipo del elemento de sintaxis. La unidad de descomposición 136 preferiblemente realiza la descomposición individual de los elementos de sintaxis de tal modo que existe una puesta en correspondencia inversa única previamente determinada que se usa más adelante en el lado de descodificación, a partir del número entero de símbolos de origen hasta el elemento de sintaxis respectivo, común para todos los elementos de sintaxis.

Por ejemplo, la unidad de descomposición 136 se puede configurar para descomponer los elementos de sintaxis z en el tren de bits que se puede analizar sintácticamente 138, en dos símbolos de origen x e y de tal modo que $z = x + y$, $z = x \cdot y$, o $z = x : y$. Mediante esta medida, el subdivisor 100 puede descomponer los elementos de sintaxis en dos componentes, en concreto los símbolos de origen del tren de símbolos de origen 106, uno de los cuales es adecuado para someterse a codificación de VLC en términos de la eficiencia de compresión, tal como x, y el otro de los cuales no es adecuado para la codificación de VLC y se pasa, por lo tanto, al segundo subtren 110 en lugar del primer subtren 108, tal como y. No es necesario que la descomposición usada por la unidad de descomposición 136 sea biyectiva. No obstante, tal como se ha mencionado antes, debería existir una puesta en correspondencia inversa que permita una única recuperación de los elementos de sintaxis de las posibles descomposiciones entre las cuales puede elegir la unidad de descomposición 136 si la descomposición no es biyectiva.

Hasta la fecha, se han descrito diferentes posibilidades para la manipulación de diferentes elementos de sintaxis. En

lo que respecta a si existen tales elementos de sintaxis o casos, es opcional. La descripción adicional, no obstante, se centra en los elementos de sintaxis que se descomponen por la unidad de descomposición 136 de acuerdo con el siguiente principio.

Tal como se muestra en la figura 1b, la unidad de descomposición 136 está configurada para descomponer determinados elementos de sintaxis z en el tren de bits que se puede analizar sintácticamente 138 en fases. Pueden existir dos o más fases. Las fases son para dividir el intervalo de valores de elemento de sintaxis z en dos o más subintervalos o subrangos adyacentes tal como se muestra en la figura 1c. El intervalo de valores de elemento de sintaxis puede tener dos puntos de extremo infinitos, meramente uno o puede tener unos puntos de extremo definidos. En la figura 1c, el intervalo de valores de elemento de sintaxis se subdivide a modo de ejemplo en tres particiones 140_{1-3}. Tal como se muestra en la figura 1b, si el elemento de sintaxis es mayor o igual que el límite 142 de la primera partición 140_1, es decir el límite superior que separa las particiones 140_1 y 140_2, entonces al elemento de sintaxis se le sustrae el límite límite1 de la primera partición 140_1 y z se comprueba de nuevo en lo que respecta a si la misma es incluso mayor o igual que el límite 144 de la segunda partición 140_2, es decir el límite superior que separa las particiones 140_2 y 140_3. Si z' es mayor o igual que el límite 144, entonces a z' se le sustrae el límite límite2 de la segunda partición 140_2 lo que da como resultado z". En el primer caso en el que z es más pequeño que límite1, el elemento de sintaxis z se envía al subdivisor 100 en lenguaje claro. En el caso de que z se encuentre entre el límite1 y el límite2, el elemento de sintaxis z se envía al subdivisor 100 en como una tupla (límite1, z') con z = límite1 + z', y en el caso de que z se encuentre por encima de límite2, el elemento de sintaxis z se envía al subdivisor 100 en como una tripleta (límite1, límite2 - límite1, z') con z = límite1 + límite2 + z'. La primera (o la única) componente, es decir z o límite1, forma un primer símbolo de origen que va a codificarse por el subdivisor 100, la segunda componente, es decir z' o límite2 - límite1 forma un segundo símbolo de origen que va a codificarse por el subdivisor 100, de encontrarse presente, y la tercera componente, es decir z", forma un tercer símbolo de origen que va a codificarse por el subdivisor 100, de encontrarse presente. Por lo tanto, de acuerdo con la figura 1b y 1c, el elemento de sintaxis se pone en correspondencia con cualquiera de 1 a 3 símbolos de origen, pero se pueden obtener fácilmente generalizaciones acerca de un número más o menos máximo de símbolos de origen a partir de la descripción anterior, y tales alternativas también se describirán en lo sucesivo.

En cualquier caso, la totalidad de estas diferentes componentes o símbolos de origen resultantes son de acuerdo con las realizaciones posteriores, codificados con alternativas de codificación de entre las mismas. Por lo menos uno de estos se reenvía por el subdivisor al codificador de PIPE 104, y por último otro de los mismos se envía al codificador de VLC 102.

En lo sucesivo se esbozan con más detalle realizaciones ventajosas particulares.

Después de haber descrito en lo que antecede un aparato de codificación por entropía, un aparato de descodificación por entropía se describe con respecto a la figura 2a. El aparato de descodificación por entropía de la figura 2a comprende un descodificador de VLC 200 y un descodificador de PIPE 202. El descodificador de VLC 200 está configurado para reconstruir por códigos los símbolos de origen de una primera subsecuencia 204 a partir de palabras de código de un primer tren de bits 206. El primer tren de bits 206 es igual a un tren de bits 112 de la figura 1, y lo mismo es de aplicación a la subsecuencia 204 en lo que respecta a la subsecuencia 108 de la figura 1a. El descodificador de PIPE 202 está configurado para reconstruir una segunda subsecuencia 208 de símbolos de origen, que se representa en la forma de una secuencia de símbolos alfabéticos, y comprende una pluralidad de descodificadores por entropía 210, una unidad de asignación 212 y un selector 214. La pluralidad de descodificadores por entropía 210 están configurados para convertir uno respectivo de los segundos trenes de bits 216 en símbolos alfabéticos de la secuencia de símbolos alfabéticos. La unidad de asignación 212 está configurada para asignar una medida de una estimada de una distribución de probabilidad entre los posibles valores que pueden asumir los símbolos alfabéticos respectivos, a cada símbolo alfabético de la secuencia de símbolos alfabéticos que representa la segunda subsecuencia 208 de símbolos de origen que se va a reconstruir, sobre la base de la información que está contenida dentro de símbolos alfabéticos previamente reconstruidos de la secuencia de símbolos alfabéticos. Con este fin, la unidad de asignación 212 se puede conectar en serie entre una salida del selector 214 y una entrada del mismo, mientras que entradas adicionales del selector 214 tienen salidas de los descodificadores por entropía 210 conectadas a los mismos de forma respectiva. El selector 214 está configurado para recuperar cada símbolo alfabético de la secuencia de símbolos alfabéticos a partir de uno seleccionado de la pluralidad de descodificadores por entropía 210, dependiendo la selección de la medida asignada al símbolo alfabético respectivo. Dicho de otra forma, el selector 214 junto con la unidad de asignación 212 es operativo para recuperar los símbolos alfabéticos obtenidos mediante los descodificadores por entropía 210 en un orden entre los descodificadores por entropía 210 obtenidos mediante información de prospección contenida dentro de símbolos alfabéticos previos de la secuencia de símbolos alfabéticos. Dicho incluso de otra forma, la unidad de asignación 212 y el selector 214 son capaces de reconstruir el orden original de los símbolos alfabéticos de símbolo alfabético a símbolo alfabético. Junto con el pronóstico del siguiente símbolo alfabético, la unidad de asignación 212 es capaz de determinar la medida que se ha mencionado en lo que antecede de la estimada de la distribución de probabilidad para el símbolo alfabético respectivo mediante el uso del cual el selector 214 selecciona entre los descodificadores por entropía 210 para recuperar el valor real de este símbolo alfabético. Para ser incluso más precisos, tal como se

describirá con más detalle en lo sucesivo, el descodificador de PIPE 202 se puede configurar para reconstruir la subsecuencia 208 de símbolos de origen, que se representa en la forma de la secuencia de símbolos alfabéticos, sensible a solicitudes de símbolo alfabético que solicitan de forma secuencial los símbolos alfabéticos, y la unidad de asignación 212 se puede configurar para asignar a cada solicitud para un símbolo alfabético de la secuencia de símbolos alfabéticos que representa la segunda subsecuencia (208) de símbolos de origen que se va a reconstruir, la medida que se ha mencionado en lo que antecede de una estimada de una distribución de probabilidad entre los posibles valores que puede asumir el símbolo alfabético respectivo. Por consiguiente, el selector 214 se puede configurar para recuperar, para cada solicitud para un símbolo alfabético de la secuencia de símbolos alfabéticos que representa la segunda subsecuencia (208) de símbolos de origen que se va a reconstruir, el símbolo alfabético respectivo de la secuencia de símbolos alfabéticos a partir de uno seleccionado de la pluralidad de descodificadores por entropía 210, dependiendo la selección de la medida asignada a la solicitud respectiva del símbolo alfabético respectivo. La concordancia entre solicitudes en el lado de descodificación por un lado, y la codificación o flujo de datos en el lado de codificación por otro lado se bosquejará con más detalle con respecto a la figura 4.

Como la primera subsecuencia 214 de símbolos de origen y la segunda subsecuencia 208 de símbolos de origen comúnmente a partir de una secuencia común 210 de símbolos de origen, el aparato de descodificación por entropía de la figura 2a puede comprender de forma opcional un recombinador 220 que está configurado para recombinar la primera subsecuencia 204 y la segunda subsecuencia 208 para obtener la secuencia común 218 de símbolos de origen. Esta secuencia común 208 de símbolos de origen da una reconstrucción de la secuencia 106 de la figura 1a.

De acuerdo con la descripción que se ha presentado en lo que antecede con respecto a la figura 1, los símbolos de origen de la primera y la segunda subsecuencias 204 y 208 pueden ser elementos de sintaxis de un tren de bits que se puede analizar sintácticamente. En este caso, el recombinador 220 se podría configurar para reconstruir este tren de bits que se puede analizar sintácticamente de la secuencia 218 de elementos de sintaxis mediante la intercalación de los símbolos de origen que llegan por medio de la primera y la segunda subsecuencias 204 y 208 en un orden que es dictado por una cierta regla de análisis sintáctico que define un orden entre los elementos de sintaxis. En particular, los elementos de sintaxis pueden ser, tal como se ha descrito en lo que antecede, de diferente tipo y el recombinador 220 se puede configurar para recuperar o solicitar elementos de sintaxis de un primer grupo de tipos a partir del descodificador de VLC 200 por medio del subtren 204, y elementos de sintaxis de un segundo tipo a partir del descodificador de PIPE 202 por medio del subtren 208. Por consiguiente, siempre que la regla de análisis sintáctico recién mencionada indique que un elemento de sintaxis de un tipo dentro del primer grupo es el siguiente en línea, el recombinador 202 inserta un símbolo de origen real de la subsecuencia 204 en la secuencia común 218, y a partir de la subsecuencia 208 de lo contrario.

De forma similar, el descodificador de PIPE 202 podría comprender un desimbolizador 222 conectado entre la salida del selector 214 y una entrada del recombinador 220. De forma similar a la descripción en lo que antecede con respecto a la figura 1, el desimbolizador 222 se podría considerar como que es externo con respecto al descodificador de PIPE 202 e incluso se podría disponer por detrás del recombinador 202, es decir en el lado de salida del recombinador 220, como alternativa. El desimbolizador 222 se podría configurar para volver a establecer una correspondencia, en unidades de secuencias parciales de símbolos alfabéticos, de la secuencia de símbolos alfabéticos 224 que es emitida por el selector 214 con los símbolos de origen, es decir elementos de sintaxis de la subsecuencia 208. De forma similar al recombinador 220, el desimbolizador 222 tiene conocimiento sobre la construcción de secuencias parciales posibles de símbolos alfabéticos. En particular, el desimbolizador 222 puede analizar símbolos alfabéticos recientemente recibidos a partir del selector 214 con el fin de determinar en lo que respecta a si estos símbolos alfabéticos recientemente recibidos dan una secuencia parcial válida de símbolos alfabéticos asociados con un valor respectivo del elemento de sintaxis respectivo, o en lo que respecta a si este no es el caso, y qué símbolo alfabético es el siguiente que está ausente. Dicho incluso de otra forma, el simbolizador 222 tiene conocimiento, en cualquier instante, en lo que respecta a si se han de recibir símbolos alfabéticos adicionales a partir del selector 214 con el fin de acabar la recepción de un elemento de sintaxis respectivo o no, y por consiguiente, a qué elemento de sintaxis pertenece uno respectivo de los símbolos alfabéticos que son emitidos por el selector 214. Con este fin, el desimbolizador 222 puede usar un esquema de puesta en correspondencia de (de)simbolización que difiere para elementos de sintaxis de diferente tipo. De forma similar, la unidad de asignación 212 tiene conocimiento sobre la asociación de un símbolo alfabético actual que va a recuperarse a partir de cualquiera de los descodificadores por entropía 210 por el selector 214, hasta uno respectivo de los elementos de sintaxis y puede establecer la medida que se ha mencionado en lo que antecede de estimación de una distribución de probabilidad de este símbolo alfabético en consecuencia, es decir dependiendo del tipo de elemento de sintaxis asociado. Lo que es más, la unidad de asignación 212 puede diferenciar entre diferentes símbolos alfabéticos que pertenecen a la misma secuencia parcial de un símbolo alfabético actual y puede establecer la medida de la estimada de distribución de probabilidad de forma diferente para estos símbolos alfabéticos. En lo sucesivo se describen con más detalle detalles a este respecto. Tal como se ha descrito en los mismos, la unidad de asignación 212 se puede configurar para asignar contextos a los símbolos alfabéticos. La asignación puede ser dependiente del tipo de elemento de sintaxis y / o la posición dentro de la secuencia parcial de símbolos alfabéticos del elemento de sintaxis actual. Tan pronto como la unidad de asignación 212 ha asignado un contexto a un símbolo alfabético actual que va a recuperarse a partir de cualquiera de los descodificadores por entropía 210 por el selector 214, el símbolo

alfabético puede tener de forma inherente la medida de la estimada de distribución de probabilidad asociada al mismo debido a que cada contexto tiene su medida de estimada asociada al mismo. Además, el contexto - y su medida asociada de la estimada de distribución de probabilidad – puede adaptarse de acuerdo con las estadísticas reales de los símbolos alfabéticos del contexto respectivo que se ha recuperado a partir de los descodificadores por entropía 210 hasta el momento. En lo sucesivo se presentan con más detalle detalles a este respecto.

De forma similar al análisis anterior de la figura 1, puede ser posible que la correspondencia entre los símbolos de origen que se han mencionado en lo que antecede de las subsecuencias 204 y 208 en los elementos de sintaxis no sea una correspondencia biunívoca. En su lugar, los elementos de sintaxis se pueden haber descompuesto en un número entero de símbolos de origen con el número variando, con el tiempo, entre los elementos de sintaxis, pero que es, en cualquier caso, mayor que uno por lo menos para un elemento de sintaxis. Tal como se ha hecho notar en lo que antecede, la siguiente descripción se centra en la manipulación de estos tipos de elementos de sintaxis, e incluso puede que no se encuentren presentes elementos de sintaxis de otros tipos.

Para manipular los elementos de sintaxis recién mencionados, el aparato de descodificación por entropía de la figura 2a puede comprender una unidad de composición 224 que está configurada para volver a hacer la descomposición que es realizada por la unidad de descomposición 136 de la figura 1a. En particular, la unidad de composición 224 se puede configurar para componer la secuencia 226 de elementos de sintaxis a partir de los símbolos de origen de la secuencia 218 o, si el recombinador 220 está ausente, las subsecuencias 204 y 208, mediante la composición de forma individual de cada elemento de sintaxis a partir de un número entero respectivo de símbolos de origen con uno de los símbolos de origen del número entero de símbolos de origen que pertenecen a la primera subsecuencia 204 y otro de los símbolos de origen del número entero de símbolos de origen del mismo elemento de sintaxis que pertenece a la segunda subsecuencia 208. Mediante esta medida, determinados elementos de sintaxis se pueden haber descompuesto en el lado de codificador con el fin de separar componentes adecuadas para la descodificación de VLC a partir de una componente restante que tienen que pasarse a través de la trayectoria de descodificación de PIPE. De forma similar al análisis anterior, el elemento de sintaxis puede ser de diferente tipo y la unidad de composición 224 se puede configurar para realizar la composición individual dependiendo del tipo de los elementos de sintaxis. En particular, la unidad de composición 224 se puede configurar para obtener los elementos de sintaxis respectivos al combinar de forma lógica o matemática el número entero de símbolos de origen del elemento de sintaxis respectivo. Por ejemplo, la unidad de composición 224 puede configurarse, para cada elemento de sintaxis, para aplicar +, -, : o · a un primer y un segundo símbolos de origen de un elemento de sintaxis.

Tal como se ha descrito en lo que antecede, las realizaciones que se describen en lo sucesivo en el presente documento, no obstante, se centran en elementos de sintaxis que se descomponen por la unidad de descomposición 136 de acuerdo con la figura 1b y 1c y las alternativas que se describen en lo que respecta a las mismas. La figura 2a muestra lo que respecta a cómo puede funcionar la unidad de composición 224 para reconstruir estos elementos de sintaxis a partir de sus símbolos de origen 218.

Tal como se muestra en la figura 2b, la unidad de composición 224 está configurada para componer tales elementos de sintaxis z en fases a partir de unos símbolos de origen entrantes s_1 a s_x con x siendo cualquiera de 1 a 3 en el presente ejemplo. Pueden existir dos o más fases. Tal como se muestra en la figura 2b, la unidad de composición 224 establece preliminarmente que z es el primer símbolo s_1 y comprueba en lo que respecta a si z es igual al primer límite1. Si este no es el caso, se ha hallado z. De lo contrario, la unidad de composición 224 añade el siguiente símbolo de origen s_2 del tren de símbolos de origen 218 a z y comprueba de nuevo en lo que respecta a si esta z es igual al límite2. De no ser así, se ha hallado z. De no ser así, la unidad de composición 224 añade el siguiente símbolo de origen s_3 del tren de símbolos de origen 218 a z, con el fin de obtener z en su forma final. Se pueden obtener fácilmente generalizaciones acerca de un número más o menos máximo de símbolos de origen a partir de la descripción anterior, y tales alternativas también se describirán en lo sucesivo.

En cualquier caso, la totalidad de estas diferentes componentes o símbolos de origen resultantes son de acuerdo con la descripción posterior, codificados con alternativas de codificación de entre las mismas. Por lo menos uno de estos se reenvía por el subdivisor al codificador de PIPE 104, y por último otro de los mismos se envía al codificador de VLC 102.

En lo sucesivo se esbozan con más detalle detalles ventajosos particulares. Estos detalles se centran en posibilidades favorables de dividir el intervalo de valores de los elementos de sintaxis y los esquemas de codificación de VLC y de PIPE por entropía los cuales se pueden usar para codificar los símbolos de origen.

Además, tal como se ha descrito también en lo que antecede con respecto a la figura 1, el aparato de descodificación por entropía de la figura 2a se puede configurar para recibir el primer tren de bits 206 así como la pluralidad de segundos trenes de bits 216 por separado o en una forma intercalada por medio de un tren de bits intercalado 228. En este último caso, el aparato de descodificación por entropía de la figura 2a puede comprender un desintercalador 230 que está configurado para desintercalar el tren de bits intercalado 228 para obtener el primer tren de bits 206 por un lado y la pluralidad de segundos trenes de bits 216 por otro lado. De forma similar al análisis

anterior de la figura 1, el desintercalador 230 se puede subdividir en dos fases, en concreto un desintercalador 232 para desintercalar el tren de bits intercalado 228 en dos partes, en concreto el tren de bits 206 por un lado y una forma intercalada 234 del segundo tren de bits 216 por otro lado, y un desintercalador 236 para desintercalar este último tren de bits 234 para obtener los trenes de bits individuales 216.

Por lo tanto, la figura 1a y la figura 2a mostraron un aparato de codificación por entropía por un lado y un aparato de descodificación por entropía adecuado para descodificar el resultado de codificación que se obtiene por el aparato de codificación por entropía de la figura 1, por otro lado. Detalles en lo que respecta a muchos de los elementos que se muestran en la figura 1a y 2 se describen con más detalle con respecto a las figuras adicionales. Por consiguiente, se hace referencia a estos detalles en la siguiente descripción y estos detalles se deberían considerar como que también son de aplicación a la figura 1a y 2 de forma individual, siempre que estos detalles se puedan implementar por separado en los codificadores y los descodificadores que se han descrito en lo que antecede. Meramente con respecto a los intercaladores y desintercaladores 132 y 234, se realiza en el presente caso alguna notificación adicional. En particular, una intercalación de los trenes de bits 112 y 118 puede ser favorable en el caso de que los trenes de bits hayan de multiplexarse en un canal con el fin de transmitirse. En este caso, puede ser favorable intercalar el tren de bits de VLC 112 por un lado y los trenes de bits de codificación de PIPE 118 por otro lado con el fin de obedecer determinadas condiciones que se han de satisfacer tales como obedecer un cierto retardo de descodificación máximo. Dicho incluso de otra forma, puede ser necesario que el desplazamiento de tiempo relativo entre los instantes en los que los elementos de sintaxis y los símbolos de origen, de forma respectiva, se pueda recuperar en el lado de descodificación por un lado y el desplazamiento relativo en el tiempo de acuerdo con su posición en el tren de bits que se puede analizar sintácticamente por otro lado, no supere un cierto retardo máximo. Muchas alternativas para resolver este problema se describen en lo sucesivo. Una de estas posibilidades comporta que los codificadores por entropía 116 sean de un tipo de codificador de longitud variable que está configurado para poner en correspondencia secuencias de símbolos alfabéticos con palabras de código, y los descodificadores por entropía 210 para realizar la puesta en correspondencia inversa. Las palabras de código del tren de bits de VLC 112 y los trenes de bits de PIPE 118 pueden estar, pero no tienen que estar, seleccionadas de tal modo que ninguna palabra de código de alguno de estos trenes de bits es un prefijo de cualquier palabra de código de cualquiera de los otros trenes de bits, de tal modo que las fronteras de palabra de código siguen pudiendo determinarse de forma única en el lado de descodificador. En cualquier caso, el intercalador 128 puede configurarse para reservar y almacenar en memoria intermedia una secuencia de entradas de palabra de código para la palabra de código dentro del primer tren de bits 112 y el segundo tren de bits 118 en un orden secuencial dependiendo de un orden en el cual los símbolos alfabéticos de la secuencia 124 de símbolos alfabéticos reenviados por el selector 120 a la pluralidad de codificadores por entropía 116 dan como resultado un comienzo de una nueva secuencia de símbolos alfabéticos que van a ponerse en correspondencia con una palabra de código respectiva en el codificador por entropía respectivo 116 y un nuevo símbolo de origen del segundo subtren 108 se pone en correspondencia por el codificador de VLC 102, de forma respectiva. Dicho de otra forma, el intercalador 128 inserta las palabras de código del tren de bits 112 en el tren de bits común 126 en el orden de los símbolos de origen del que se han obtenido mediante codificación de VLC, en su orden en el interior del subtren 108 y el tren de símbolos de origen 106, de forma respectiva. Las palabras de código que son emitidas por los codificadores por entropía 116 se insertan en el tren de bits común 126 entre unas consecutivas de las palabras de código del tren de bits de VLC 112. Debido a la categorización de la codificación de PIPE de los símbolos alfabéticos por la unidad de asignación 114 y el selector 120, de forma respectiva, cada una de las palabras de código de los codificadores por entropía 116 tiene símbolos alfabéticos de diferentes símbolos de origen del subtren 110 codificados en las mismas. La posición de las palabras de código de los trenes de bits codificados de PIPE 118 dentro del tren de bits común 126 entre estos y en relación con la palabra de código de VLC del tren de bits 112 se determina mediante el primer símbolo alfabético codificado en cada palabra de código, de forma respectiva, es decir el más antiguo en el tiempo. El orden de estos símbolos alfabéticos primarios codificados en las palabras de código de los trenes de bits 118 en el tren de símbolos alfabéticos 124 determina el orden de las palabras de código de los trenes de bits 118 dentro del tren de bits común 126 entre estos; en relación con las palabras de código de VLC del tren de bits 112, el símbolo de origen al cual pertenecen estos símbolos alfabéticos primarios codificados en las palabras de código de los trenes de bits 118, determina entre qué palabras de código consecutivas del tren de bits 112 ha de situarse la palabra de código respectiva de cualquiera de los trenes de bits 118. En particular, las palabras de código de VLC consecutivas entre las cuales se ha de situar la palabra de código respectiva de cualquiera de los trenes de bits 118, son aquellas entre las cuales se sitúa el símbolo de origen del subtren 110 de acuerdo con el orden original del tren de símbolos de origen sin subdividir 106, al cual pertenece el símbolo alfabético primario respectivo codificado en la palabra de código respectiva de los trenes de bits 118. El intercalador 128 se puede configurar para retirar palabras de código introducidas en las entradas de palabra de código que se han mencionado en lo que antecede en orden secuencial para obtener el tren de bits común 126 de palabras de código intercaladas. Tal como ya se ha descrito en lo que antecede, los codificadores por entropía 116 se pueden configurar para introducir de forma secuencial sus palabras de código en las entradas de palabra de código que se han reservado para el codificador por entropía respectivo 116 y el selector 120 se puede configurar para reenviar los símbolos alfabéticos que representan los símbolos de origen del segundo subtren 110 en un orden manteniendo un orden en el cual los símbolos de origen del primer subtren 108 y el segundo subtren 110 se intercalaron dentro de la secuencia 106 de símbolos de origen.

Se pueden proporcionar medidas adicionales con el fin de hacer frente a situaciones en las que unos determinados de los codificadores por entropía 116 se seleccionan tan raramente que lleva demasiado tiempo obtener una palabra de código válida dentro de ese codificador por entropía 116 muy raramente usado. En lo sucesivo se describen con más detalle ejemplos para tales medidas. En particular, el intercalador 128 junto con el codificador por entropía 116 puede configurarse, en este caso, para evacuar sus símbolos alfabéticos recogidos hasta el momento y las palabras de código que se han introducido en las entradas de palabra de código que se han mencionado en lo que antecede, de forma respectiva, de una forma de tal modo que el tiempo de este procedimiento de evacuación se puede pronosticar o emular en el lado de descodificación.

En el lado de descodificación, el desintercalador 230 puede actuar en el sentido inverso: siempre que, de acuerdo con el esquema de análisis sintáctico que se ha mencionado en lo que antecede, el siguiente símbolo de origen que va a ser descodificado, sea un símbolo codificado de VLC, una palabra de código actual dentro del tren de bits común 228 se considera como una palabra de código VLC y se reenvía en el interior del tren de bits 206 al descodificador de VLC 200. Por otro lado, siempre que cualquiera de los símbolos alfabéticos que pertenecen a cualquiera de los símbolos codificados de PIPE del subtren 208 sea un símbolo alfabético primario, es decir necesite una nueva puesta en correspondencia de una palabra de código de uno respectivo de los trenes de bits 216 con una secuencia de símbolos alfabéticos respectiva por el descodificador por entropía respectivo 210, la palabra de código actual del tren de bits común 228 se considera como una palabra de código codificada de PIPE y se reenvía al descodificador por entropía respectivo 210. La detección de la siguiente frontera de palabra de código, es decir la detección de la extensión de la siguiente palabra de código desde el final de la palabra de código que acaba de reenviarse a cualquiera de los descodificadores 200 y 202, de forma respectiva, hasta su extremo dentro del tren de bits intercalado entrante 228 se puede aplazar, y realizarse con el conocimiento del descodificador 200 y 202 siendo el destinatario dedicado de esta siguiente palabra de código de acuerdo con la regla que se ha bosquejado en lo que antecede: sobre la base de este conocimiento, el libro de códigos que es usado por el descodificador destinatario es conocido y la palabra de código respectiva es detectable. Si, por otro lado, los libros de códigos se diseñaran de tal modo que las fronteras de palabra de código fueran detectables sin el conocimiento a priori acerca del descodificador destinatario entre 200 y 202, entonces la separación de palabras de código se podría realizar en paralelo. En cualquier caso, debido a la intercalación, los símbolos de origen se encuentran disponibles en el descodificador en una forma descodificada por entropía, es decir como símbolos de origen, en su orden correcto con un retardo razonable.

Después de haber descrito en lo que antecede realizaciones para un aparato de codificación por entropía y un aparato de descodificación por entropía respectivo, a continuación se describen más detalles para los codificadores de PIPE y los descodificadores de PIPE que se han mencionado en lo que antecede.

Un codificador de PIPE se ilustra en la figura 3. El mismo se puede usar como codificador de PIPE en la figura 1a. El codificador de PIPE convierte de una forma sin pérdida un tren de símbolos de origen 1 en un conjunto de dos o más trenes de bits parciales 12. Cada símbolo de origen 1 puede estar asociado con una categoría o tipo de un conjunto de una o más categorías o tipos. Como un ejemplo, las categorías pueden especificar el tipo del símbolo de origen. En el contexto de la codificación de vídeo híbrida, una categoría separada puede estar asociada con modos de codificación de macrobloques, modos de codificación de bloques, índices de imágenes de referencia, diferencias de vector de movimiento, indicadores de subdivisión, indicadores de bloques codificados, parámetros de cuantificación, niveles de coeficiente de transformada, etc. En otras áreas de aplicación tales como la codificación de audio, de habla, de textos, de documentos o de datos generales, son posibles diferentes categorizaciones de símbolos de origen. En general, cada símbolo de origen puede adoptar un valor de un conjunto finito o infinito contable de valores, en donde el conjunto de posibles valores de símbolo de origen puede diferir para diferentes categorías de símbolos de origen. Para reducir la complejidad del algoritmo de codificación y de descodificación y para permitir un diseño de codificación y de descodificación general para diferentes símbolos de origen y categorías de símbolos de origen, los símbolos de origen 1 se convierten en unos conjuntos ordenados de decisiones binarias y estas decisiones binarias se procesan entonces mediante unos algoritmos de codificación binarios simples. Por lo tanto, la unidad de binarización 2 pone en correspondencia de forma biyectiva el valor de cada símbolo de origen 1 con una secuencia (o cadena) de contenedores 3. La secuencia de contenedores 3 representa un conjunto de decisiones binarias ordenadas. Cada contenedor 3 o decisión binaria puede adoptar un valor de un conjunto de dos valores, por ejemplo uno de los valores 0 y 1. El esquema de binarización puede ser diferente para diferentes categorías de símbolos de origen. El esquema de binarización para una categoría de símbolos de origen particular puede depender del conjunto de posibles valores de símbolo de origen y / u otras propiedades de los símbolos de origen para la categoría particular. La tabla 1 ilustra tres esquemas de binarización a modo de ejemplo para conjuntos infinitos contables. Los esquemas de binarización para conjuntos infinitos contables también se pueden aplicar para conjuntos finitos grandes de valores de símbolo. En particular para conjuntos finitos grandes de valores de símbolo, la falta de eficiencia (resultante de secuencias sin usar de contenedores) puede ser despreciable, pero la universalidad de tales esquemas de binarización proporciona una ventaja en términos de la complejidad y de los requisitos de memoria. Para conjuntos finitos pequeños de valores de símbolo, a menudo es preferible (en términos de la eficiencia de codificación) adaptar el esquema de binarización al número de valores de símbolo posibles. La tabla 2 ilustra tres esquemas de binarización a modo de ejemplo para conjuntos finitos de 8 valores. Los esquemas

de binarización para conjuntos finitos se pueden obtener a partir de los esquemas de binarización universales para conjuntos infinitos contables modificando algunas secuencias de contenedores de una forma tal que los conjuntos finitos de secuencias de contenedores representan un código libre de redundancia (y potencialmente mediante la reordenación de las secuencias de contenedores). Como un ejemplo, el esquema de binarización unaria truncada en la tabla 2 se creó modificando la secuencia de contenedores para el símbolo de origen 7 de la binarización unaria universal (véase la tabla 1). La binarización de Golomb Exponencial truncada y reordenada de orden 0 en la tabla 2 se creó modificando la secuencia de contenedores para el símbolo de origen 7 de la binarización de orden 0 de Golomb Exponencial universal (véase la tabla 1) y mediante la reordenación de las secuencias de contenedores (la secuencia de contenedores truncada para el símbolo 7 se asignó al símbolo 1). Para conjuntos finitos de símbolos, también es posible usar unos esquemas de binarización no sistemática / no universal, tal como se ejemplifica en la última columna de la tabla 2.

Tabla 1: Ejemplos de binarización para conjuntos infinitos contables (o conjuntos finitos grandes).

valor de símbolo	binarización unaria	binarización de orden 0 de Golomb Exponencial	binarización de orden 1 de Golomb Exponencial
0	1	1	10
1	01	010	11
2	001	011	0100
3	0001	0010 0	0101
4	0000 1	0010 1	0110
5	0000 01	0011 0	0111
6	0000 001	0011 1	0010 00
7	0000 0001	0001 000	0010 01
...	...	...	...

Tabla 2: Ejemplos de binarización para conjuntos finitos.

valor de símbolo	binarización unaria truncada	binarización de orden 0 de Golomb Exponencial truncada y reordenada	binarización no sistemática
0	1	1	000
1	01	000	001
2	001	010	01
3	0001	011	1000
4	0000 1	0010 0	1001
5	0000 01	0010 1	1010
6	0000 001	0011 0	1011 0
7	0000 000	0011 1	1011 1

Cada contenedor 3 de la secuencia de contenedores creado por la unidad de binarización 2 se alimenta a la unidad de asignación de parámetros 4 en orden secuencial. La unidad de asignación de parámetros asigna un conjunto de uno o más parámetros a cada contenedor 3 y emite el contenedor con el conjunto asociado de parámetros 5. El conjunto de parámetros se determina exactamente de la misma forma en el codificador y en el descodificador. El conjunto de parámetros puede consistir en uno o más de los siguientes parámetros:

- una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual,
- una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual,
- un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor actual,
- la categoría del símbolo de origen asociado,
- una medida para la importancia del símbolo de origen asociado,
- una medida para la ubicación del símbolo asociado (por ejemplo, en conjuntos de datos temporales, espaciales o volumétricos),
- un identificador que especifica la protección de código de canal para el contenedor o el símbolo de origen asociado,
- un identificador que especifica el esquema de cifrado para el contenedor o el símbolo de origen asociado,
- un identificador que especifica una clase para el símbolo asociado,
- el número de contenedor en la secuencia de contenedores para el símbolo de origen asociado.

La unidad de asignación de parámetros 4 puede asociar cada contenedor 3, 5 con una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual. La unidad de asignación de parámetros 4 asocia cada contenedor 3, 5 con una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual y un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor actual. Se debería hacer notar que la probabilidad para el valor de contenedor menos probable o más probable y el identificador que especifica cuál de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable son unas medidas equivalentes para la probabilidad de uno de los dos valores de contenedor posibles.

La unidad de asignación de parámetros 4 puede asociar cada contenedor 3, 5 con una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual y uno o más parámetros adicionales (que puede ser uno o más de los parámetros que se han enumerado en lo que antecede). Además, la unidad de asignación de parámetros 4 puede asociar cada contenedor 3, 5 con una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual, un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor actual, y uno o más parámetros adicionales (que puede ser uno o más de los parámetros que se han enumerado en lo que antecede).
La unidad de asignación de parámetros 4 puede determinar una o más de las medidas de probabilidad que se han mencionado en lo que antecede (una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual, una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual, un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor actual) sobre la base de un conjunto de uno o más símbolos ya codificados. Los símbolos codificados que se usan para determinar las medidas de probabilidad pueden incluir uno o más símbolos ya codificados de la misma categoría de símbolos, uno o más símbolos ya codificados de la misma categoría de símbolos que se corresponden con conjuntos de datos (tales como bloques o grupos de muestras) de unas ubicaciones espaciales y / o temporales cercanas (en relación con el conjunto de datos que está asociado con el símbolo de origen actual), o uno o más símbolos ya codificados de diferentes categorías de símbolos que se corresponden con conjuntos de datos de las mismas ubicaciones espaciales y / o temporales y / o unas cercanas (en relación con el conjunto de datos que está asociado con el símbolo de origen actual).

Cada contenedor con un conjunto asociado de parámetros 5 que es una salida de la unidad de asignación de parámetros 4 se alimenta en un selector de memoria intermedia de contenedores 6. El selector de memoria intermedia de contenedores 6 potencialmente modifica el valor del contenedor de entrada 5 sobre la base del valor de contenedor de entrada y los parámetros asociados 5 y alimenta el contenedor de salida 7 - con un valor potencialmente modificado – en una de dos o más memorias intermedias de contenedores 8. La memoria intermedia de contenedores 8 a la que se envía el contenedor de salida 7 se determinar sobre la base del valor del contenedor de entrada 5 y / o el valor de los parámetros asociados 5.

El selector de memoria intermedia de contenedores 6 puede no modificar el valor del contenedor, es decir, el contenedor de salida 7 tiene siempre el mismo valor que el contenedor de entrada 5.

El selector de memoria intermedia de contenedores 6 puede determinar el valor de contenedor de salida 7 sobre la base del valor de contenedor de entrada 5 y la medida asociada para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual. El valor de contenedor de salida 7 puede establecerse igual al valor de contenedor de entrada 5 si la medida para la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual es menor que (o menor que o igual a) un umbral particular; si la medida para la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual es mayor que o igual a (o mayor que) un umbral particular, el valor de contenedor de salida 7 se modifica (es decir, se establece en el opuesto del valor de contenedor de entrada). El valor de contenedor de salida 7 puede establecerse igual al valor de contenedor de entrada 5 si la medida para la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual es mayor que (o mayor que o igual a) un umbral particular; si la medida para la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual es menor que o igual a (o menor que) un umbral particular, el valor de contenedor de salida 7 se modifica (es decir, se establece en el opuesto del valor de contenedor de entrada). El valor del umbral se puede corresponder con un valor de 0,5 para la probabilidad estimada para ambos valores de contenedor posibles.

El selector de memoria intermedia de contenedores 6 puede determinar el valor de contenedor de salida 7 sobre la base del valor de contenedor de entrada 5 y el identificador asociado que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor actual. El valor de contenedor de salida 7 puede establecerse igual al valor de contenedor de entrada 5 si el identificador especifica que el primero de los dos valores de contenedor posibles representa el valor de contenedor menos probable (o más probable) para el contenedor actual, y el valor de contenedor de salida 7 se

modifica (es decir, se establece en el opuesto del valor de contenedor de entrada) si el identificador especifica que el segundo de los dos valores de contenedor posibles representa el valor de contenedor menos probable (o más probable) para el contenedor actual.

El selector de memoria intermedia de contenedores 6 puede determinar la memoria intermedia de contenedores 8 a la que se envía el contenedor de salida 7 sobre la base de la medida asociada para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual. El conjunto de valores posibles para la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles puede ser finito y el selector de memoria intermedia de contenedores 6 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 8 con cada valor posible para la estimada de la probabilidad para uno de los dos valores de contenedor posibles, en donde diferentes valores para la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles pueden asociarse con la misma memoria intermedia de contenedores 8. Además, el intervalo de valores posibles para la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles se puede subdividir en particiones para dar un número de intervalos, el selector de memoria intermedia de contenedores 6 determina el índice de intervalo para la medida actual para una estimada de la probabilidad para uno de los dos valores de contenedor posibles, y el selector de memoria intermedia de contenedores 6 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 8 con cada valor posible para el índice de intervalo, en donde diferentes valores para el índice de intervalo pueden asociarse con la misma memoria intermedia de contenedores 8. Los contenedores de entrada 5 con medidas opuestas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles (son medidas opuestas aquellas que representan unas estimadas de probabilidad P y 1 - P) pueden alimentarse a la misma memoria intermedia de contenedores 8. Además, la asociación de la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor actual con una memoria intermedia de contenedores particular se adapta con el tiempo, por ejemplo con el fin de asegurar que los trenes de bits parciales creados tienen unas tasas de bits similares.

El selector de memoria intermedia de contenedores 6 puede determinar la memoria intermedia de contenedores 8 a la que se envía el contenedor de salida 7 sobre la base de la medida asociada para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual. El conjunto de valores posibles para la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable puede ser finito y el selector de memoria intermedia de contenedores 6 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 8 con cada valor posible de la estimada de la probabilidad para el valor de contenedor menos probable o más probable, en donde diferentes valores para la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable pueden asociarse con la misma memoria intermedia de contenedores 8. Además, el intervalo de valores posibles para la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable se puede subdividir en particiones para dar un número de intervalos, el selector de memoria intermedia de contenedores 6 determina el índice de intervalo para la medida actual para una estimada de la probabilidad para el valor de contenedor menos probable o más probable, y el selector de memoria intermedia de contenedores 6 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 8 con cada valor posible para el índice de intervalo, en donde diferentes valores para el índice de intervalo pueden asociarse con la misma memoria intermedia de contenedores 8. La asociación de la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual con una memoria intermedia de contenedores particular puede adaptarse con el tiempo, por ejemplo con el fin de asegurar que los trenes de bits parciales creados tienen unas tasas de bits similares.

Cada una de las dos o más memorias intermedias de contenedores 8 se conecta con exactamente un codificador de contenedores 10 y cada codificador de contenedores solo se conecta con una memoria intermedia de contenedores 8. Cada codificador de contenedores 10 lee contenedores a partir de la memoria intermedia de contenedores asociada 8 y convierte una secuencia de contenedores 9 en una palabra de código 11, la cual representa una secuencia de bits. Las memorias intermedias de contenedores 8 representan unas memorias intermedias de tipo primero en entrar primero en salir; los contenedores que se alimentan posteriormente (en orden secuencial) a una memoria intermedia de contenedores 8 no se codifican antes que los contenedores que se alimentan con anterioridad (en orden secuencial) a la memoria intermedia de contenedores. Las palabras de código 11 que son una salida de un codificador de contenedores particular 10 se escriben en un tren de bits parcial particular 12. El algoritmo de codificación global convierte los símbolos de origen 1 en dos o más trenes de bits parciales 12, en donde el número de trenes de bits parciales es igual al número de memorias intermedias de contenedores y codificadores de contenedores. Un codificador de contenedores 10 puede convertir un número variable de contenedores 9 en una palabra de código 11 de un número variable de bits. Una ventaja de la codificación de PIPE que se ha bosquejado en lo que antecede y que se bosqueja en lo sucesivo es que la codificación de contenedores se puede realizar en paralelo (por ejemplo, para diferentes grupos de medidas de probabilidad), que reduce el tiempo de procesamiento para varias implementaciones.

Otra ventaja de la codificación de PIPE es que la codificación de contenedores, la cual es realizada por los

codificadores de contenedores 10, se puede diseñar de forma específica para diferentes conjuntos de parámetros 5. En particular, la codificación y la codificación de contenedores se puede optimizar (en términos de la complejidad y / o la eficiencia de codificación) para diferentes grupos de probabilidades estimadas. Por un lado, esto permite una reducción de la complejidad de codificación / descodificación en relación con algoritmos de codificación aritmética con una eficiencia de codificación similar. Por otro lado, esto permite una mejora de la eficiencia de codificación en relación con los algoritmos de codificación de VLC con una complejidad de codificación / descodificación similar. Los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación (es decir una puesta en correspondencia de secuencias de contenedores con palabras de código) para diferentes grupos de medidas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles 5 para el contenedor actual. Los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación para diferentes grupos de medidas para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor actual. Como alternativa, los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación para diferentes códigos de protección de canal. Los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación para diferentes esquemas de cifrado. Los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles 5 para el contenedor actual. Los codificadores de contenedores 10 implementan diferentes algoritmos de codificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas para una estimada de la probabilidad para el valor de contenedor menos probable o más probable 5 para el contenedor actual. Los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles 5 para el contenedor actual. Los codificadores de contenedores 10 pueden implementar diferentes algoritmos de codificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas para una estimada de la probabilidad para el valor de contenedor menos probable o más probable 5 para el contenedor actual.

Los codificadores de contenedores 10 - o uno o más de los codificadores de contenedores - pueden representar motores de codificación aritmética binaria. Uno o más de los codificadores de contenedores pueden representar un motor de codificación aritmética binaria, en el que la puesta en correspondencia a partir de la probabilidad de LPS / LPB representativa p_{LPS} de una memoria intermedia de contenedores dada con una anchura de intervalo de código correspondiente R_{LPS} - es decir la subdivisión en intervalos del estado interno del motor de codificación aritmética binaria, la cual se define mediante la anchura de intervalo actual R y el desplazamiento de intervalo actual L, que identifica, por ejemplo, el límite inferior del intervalo de código - se realiza mediante el uso de una consulta de tabla. Para cada motor de codificación aritmética binaria basado en tabla que está asociado con una memoria intermedia de contenedores dada, se pueden usar K valores de anchura de intervalo representativos $\{Q_0, \ldots, Q_{K-1}\}$ para representar R_{LPS} con la elección de K y los valores de anchura de intervalo representativos $\{Q_0, \ldots, Q_{K-1}\}$ siendo dependientes de la memoria intermedia de contenedores. Para una elección de $K > 1$, la codificación aritmética de un contenedor puede implicar las subetapas de puesta en correspondencia de la anchura de intervalo actual R con un índice de cuantificación q con valores en $\{0, \ldots, K-1\}$ y de realización de la subdivisión en intervalos accediendo al valor de anchura de intervalo parcial correspondiente Q_q a partir una tabla de consulta usando q como un índice. Para una elección de $K = 1$, es decir, para el caso en el que solo se da un valor de anchura de intervalo representativo Q_0, este valor Q_0 se puede elegir como una potencia de dos con el fin de permitir la descodificación de múltiples valores de MPS / MPB que entran en la memoria intermedia de contenedores correspondiente dentro de un único ciclo de renormalización. Las palabras de código resultantes de cada motor de codificación aritmética se pueden transmitir, paquetizar o almacenar por separado, o estas se pueden intercalar para el fin de transmisión o almacenamiento tal como se describe en lo sucesivo en el presente documento.

Es decir, un motor de codificación aritmética binaria 10 podría realizar las siguientes etapas en la codificación de los contenedores en su memoria intermedia de contenedores 8:

1. Recibir el contenedor valLPS a partir de una memoria intermedia de contenedores (recuérdese: el motor de codificación aritmética binaria respectivo 10 considerado en el presente caso se había elegido para recibir “contenedor” debido a que (o, dicho de otra forma, “contenedor” estaba asociado con el motor de codificación aritmética binaria respectivo 10) la estimada de distribución de probabilidad, tal como p_state[contenedor], estaba asociada con ese motor de codificación aritmética binaria 10)
2. Cuantificación de R:
   q_index = Qtab[R>>q] (o alguna otra forma de cuantificación)
3. Determinación de R_{LPS} y R:
   R_{LPS} = Rtab [q_index] (obsérvese que p_state no se menciona en el presente caso, debido a que este es fijo para el motor de codificación aritmética binaria 10 considerado, es decir p_state[codificador], y Rtab tiene almacenado en la misma unos valores previamente calculados para p[p_state[codificador]] · Q[q_index]
   R = R - R_{LPS} [es decir, R se pre-actualiza preliminarmente como si “contenedor” fuera MPS]
4. Cálculo del nuevo intervalo parcial:

si (contenedor = 1 - valMPS) entonces

$L \leftarrow L + R$
$R \leftarrow R_{LPS}$

5. Renormalización de L y R, escribiendo bits,
en donde
q_index describe el índice de un valor de cuantificación leído de Qtab,
p_state describe el estado actual (fijo para el motor de codificación aritmética binaria 10),
R_{LPS} describe la anchura de intervalo que se corresponde con el LPS y
valMPS describe el valor del bit que se corresponde con el MPS.

Por consiguiente, un motor de descodificación aritmética binaria 22 podría realizar las siguientes etapas en la descodificación de los contenedores que se emiten a la memoria intermedia de contenedores 20:

1. Recibir la solicitud para un contenedor (recuérdese: el motor de descodificación aritmética binaria respectivo 22 considerado en el presente caso se había elegido para descodificar "contenedor" debido a que (o, dicho de otra forma, "contenedor" estaba asociado con el motor de descodificación aritmética binaria respectivo 22) la estimada de distribución de probabilidad, tal como p_state[contenedor], estaba asociada con ese motor de descodificación aritmética binaria 22)
2. Cuantificación de R:
q_index = Qtab[R>>q] (o alguna otra forma de cuantificación)
3. Determinación de R_{LPS} y R:
R_{LPS} = Rtab [q_index] (obsérvese que p_state no se menciona en el presente caso, debido a que este es fijo para el motor de descodificación aritmética binaria 22 considerado, es decir p_state[codificador], y Rtab tiene almacenado en la misma unos valores previamente calculados para p[p_state[codificador]] · Q[q_index]
$R = R - R_{LPS}$ [es decir, R se re-actualiza preliminarmente como si "contenedor" fuera MPS]
4. Determinación de contenedor dependiendo de la posición del intervalo parcial:
si ($V \geq R$) entonces
contenedor ← 1 - valMPS (contenedor se descodifica como LPS; el selector de memoria intermedia de contenedores 18 obtendrá el valor de contenedor real mediante el uso de esta información de contenedor y valMPS)
$V \leftarrow V - R$
$R \leftarrow R_{LPS}$
o bien
contenedor ← valMPS (contenedor se descodifica como MPS; el selector de memoria intermedia de contenedores 18 obtendrá el valor de contenedor real mediante el uso de esta información de contenedor y valMPS)
5. Renormalización de R, lectura de un bit y actualización de V,
en donde
q_index describe el índice de un valor de cuantificación leído de Qtab,
p_state describe el estado actual (fijo para el motor de descodificación aritmética binaria 22),
R_{LPS} describe la anchura de intervalo que se corresponde con el LPS,
valMPS describe el valor del bit que se corresponde con el MPS, y V describe un valor procedente del interior del intervalo parcial actual.

Los codificadores de contenedores 10 - o uno o más de los codificadores de contenedores - pueden representar unos codificadores por entropía que ponen en correspondencia directamente las secuencias de contenedores de entrada 9 con las palabras de código 10. Tales puestas en correspondencia se pueden implementar de forma eficiente y no requieren un motor de codificación aritmética compleja. La puesta en correspondencia inversa de palabras de código con unas secuencias de contenedores (al igual que se hace en el descodificador) debería ser única con el fin de garantizar una descodificación perfecta de la secuencia de entrada, pero no se necesita que la puesta en correspondencia de secuencias de contenedores 9 con las palabras de código 10 sea necesariamente única, es decir, es posible que una secuencia de contenedores particular se pueda poner en correspondencia con más de una secuencia de palabras de código. La puesta en correspondencia de las secuencias de contenedores de entrada 9 con las palabras de código 10 también puede ser biyectiva. Preferiblemente, los codificadores de contenedores 10 - o uno o más de los codificadores de contenedores - pueden representar unos codificadores por entropía que ponen en correspondencia directamente unas secuencias de longitud variable de los contenedores de entrada 9 con unas palabras de código de longitud variable 10. Las palabras de código de salida pueden representar unos códigos libres de redundancia tales como códigos de Huffman generales o códigos de Huffman canónicos.

En la tabla 3 se ilustran dos ejemplos para la puesta en correspondencia biyectiva de secuencias de contenedores con unos códigos libres de redundancia. Las palabras de código de salida pueden representar unos códigos redundantes adecuados para la detección de errores y la recuperación de errores. Las palabras de código de salida pueden representar unos códigos de cifrado adecuados para cifrar los símbolos de origen.

Tabla 3: Ejemplos para puestas en correspondencia entre secuencias de contenedores y palabras de código.

secuencia de contenedores (el orden de los contenedores es de izquierda a derecha)	palabras de código (el orden de los bits es de izquierda a derecha)
0000 0000	1
0000 0001	0000
0000 001	0001
0000 01	0010
0000 1	0011
0001	0100
001	0101
01	0110
1	0111

secuencia de contenedores (el orden de los contenedores es de izquierda a derecha)	palabras de código (el orden de los bits es de izquierda a derecha)
000	10
01	11
001	010
11	011
1000 0	0001
1001	0010
1010	0011
1000 1	0000 0
1011	0000 1

Los codificadores de contenedores 10 - o uno o más de los codificadores de contenedores - pueden representar unos codificadores por entropía que ponen en correspondencia directamente unas secuencias de longitud variable de los contenedores de entrada 9 con unas palabras de código de longitud fija 10. Los codificadores de contenedores 10 - o uno o más de los codificadores de contenedores - representan unos codificadores por entropía que ponen en correspondencia directamente unas secuencias de longitud fija de los contenedores de entrada 9 con las palabras de código de longitud variable 10.

Un descodificador de PIPE se ilustra en la figura 4. El descodificador realiza básicamente las operaciones inversas del codificador de la figura 3, de tal modo que la secuencia (previamente codificada) de símbolos de origen 27 se descodifica a partir de un conjunto de dos o más trenes de bits parciales 24. El descodificador incluye dos flujos de proceso diferentes: Un flujo para solicitudes de datos, el cual reproduce el flujo de datos del codificador, y un flujo de datos, el cual representa la inversa del flujo de datos de codificador. En la ilustración en la figura 4, las flechas de trazo discontinuo representan el flujo de solicitudes de datos, mientras que las flechas de trazo continuo representan el flujo de datos. Los bloques básicos del descodificador básicamente reproducen los bloques básicos del codificador, pero implementan las operaciones inversas.

La descodificación de un símbolo de origen es desencadenada por una solicitud para un nuevo símbolo de origen descodificado 13 que se envía a la unidad de binarización 14. Cada solicitud para un nuevo símbolo de origen descodificado 13 puede asociarse con una categoría de un conjunto de una o más categorías. La categoría que está asociada con una solicitud para un símbolo de origen es la misma que la categoría que estaba asociada con el símbolo de origen correspondiente durante la codificación.

La unidad de binarización 14 pone en correspondencia la solicitud para un símbolo de origen 13 con una o más solicitudes para un contenedor que se envían a la unidad de asignación de parámetros 16. Como respuesta final a una solicitud para un contenedor que se envía a la unidad de asignación de parámetros 16 por la unidad de binarización 14, la unidad de binarización 14 recibe un contenedor descodificado 26 a partir del selector de memoria intermedia de contenedores 18. La unidad de binarización 14 compara la secuencia recibida de contenedores descodificados 26 con las secuencias de contenedores de un esquema de binarización particular para el símbolo de origen solicitado y, si la secuencia recibida de contenedores descodificados 26 coincide con la binarización de un símbolo de origen, la unidad de binarización vacía su memoria intermedia de contenedores y emite el símbolo de origen descodificado como respuesta final a la solicitud para un nuevo símbolo descodificado. Si la secuencia ya recibida de contenedores descodificados no coincide con cualquiera de las secuencias de contenedores para el esquema de binarización para el símbolo de origen solicitado, la unidad de binarización envía otra solicitud para un contenedor a la unidad de asignación de parámetros hasta que la secuencia de contenedores descodificados

coincide con una de las secuencias de contenedores del esquema de binarización para el símbolo de origen solicitado. Para cada solicitud para un símbolo de origen, el descodificador usa el mismo esquema de binarización que se usó para codificar el símbolo de origen correspondiente. El esquema de binarización puede ser diferente para diferentes categorías de símbolos de origen. El esquema de binarización para una categoría de símbolos de origen particular puede depender del conjunto de posibles valores de símbolo de origen y / u otras propiedades de los símbolos de origen para la categoría particular.

La unidad de asignación de parámetros asigna un conjunto de uno o más parámetros a cada solicitud para un contenedor y envía la solicitud para un contenedor con el conjunto asociado de parámetros al selector de memoria intermedia de contenedores. El conjunto de parámetros que se asignan a un contenedor solicitado por la unidad de asignación de parámetros es el mismo que se asignó al contenedor correspondiente durante la codificación. El conjunto de parámetros puede consistir en uno o más de los parámetros que se mencionan en la descripción de codificador.

La unidad de asignación de parámetros 16 puede asociar cada solicitud para un contenedor con una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor solicitado actual. En particular, la unidad de asignación de parámetros 16 puede asociar cada solicitud para un contenedor con una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor solicitado actual y un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor solicitado actual.

La unidad de asignación de parámetros 16 puede asociar cada solicitud para un contenedor 15, 17 con una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor solicitado actual y uno o más parámetros adicionales. La unidad de asignación de parámetros 16 puede asociar cada solicitud para un contenedor 15, 17 con una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor solicitado actual, un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor solicitado actual, y uno o más parámetros adicionales (que puede uno o más de los parámetros que se han enumerado en lo que antecede).

La unidad de asignación de parámetros 16 puede determinar una o más de las medidas de probabilidad que se han mencionado en lo que antecede (una medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor solicitado actual, una medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor solicitado actual, un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable para el contenedor solicitado actual) sobre la base de un conjunto de uno o más símbolos ya descodificados. La determinación de las medidas de probabilidad para a particular solicitud para un contenedor reproduce el proceso en el codificador para el contenedor correspondiente. Los símbolos descodificados que se usan para determinar las medidas de probabilidad pueden incluir uno o más símbolos ya descodificados de la misma categoría de símbolos, uno o más símbolos ya descodificados de la misma categoría de símbolos que se corresponden con conjuntos de datos (tal como bloques o grupos de muestras) de unas ubicaciones espaciales y / o temporales cercanas (en relación con el conjunto de datos que está asociado con la solicitud actual para un símbolo de origen), o uno o más símbolos ya descodificados de diferentes categorías de símbolos que se corresponden con conjuntos de datos de las mismas y / o unas ubicaciones espaciales y / o temporales cercanas (en relación con el conjunto de datos que está asociado con la solicitud actual para un símbolo de origen).

Cada solicitud para un contenedor con un conjunto asociado de parámetros 17 que es una salida de la unidad de asignación de parámetros 16 se alimenta a un selector de memoria intermedia de contenedores 18. Sobre la base del conjunto asociado de parámetros 17, el selector de memoria intermedia de contenedores 18 envía una solicitud para un contenedor 19 a una de dos o más memorias intermedias de contenedores 20 y recibe un contenedor descodificado 25 a partir de la memoria intermedia de contenedores seleccionada 20. El contenedor de entrada descodificado 25 se modifica potencialmente y el contenedor de salida descodificado 26 - con un valor potencialmente modificado - se envía a la unidad de binarización 14 como respuesta final a la solicitud para un contenedor con un conjunto asociado de parámetros 17.

La memoria intermedia de contenedores 20 a la cual se reenvía la solicitud para un contenedor se selecciona de la misma forma que la memoria intermedia de contenedores a la que se envió el contenedor de salida del selector de memoria intermedia de contenedores en el lado de codificador.

El selector de memoria intermedia de contenedores 18 puede determinar la memoria intermedia de contenedores 20 a la que se envía la solicitud para un contenedor 19 sobre la base de la medida asociada para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor solicitado actual. El conjunto de valores posibles para la medida para una estimada de la probabilidad para uno de los dos valores de contenedor

posibles puede ser finito y el selector de memoria intermedia de contenedores 18 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 20 con cada valor posible de la estimada de la probabilidad para uno de los dos valores de contenedor posibles, en donde diferentes valores para la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles pueden asociarse con la misma memoria intermedia de contenedores 20. El intervalo de valores posibles para la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles se puede subdividir en particiones para dar un número de intervalos, el selector de memoria intermedia de contenedores 18 determina el índice de intervalo para la medida actual para una estimada de la probabilidad para uno de los dos valores de contenedor posibles, y el selector de memoria intermedia de contenedores 18 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 20 con cada valor posible para el índice de intervalo, en donde diferentes valores para el índice de intervalo pueden asociarse con la misma memoria intermedia de contenedores 20. Solicitudes para unos contenedores 17 con medidas opuestas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles (son medidas opuestas aquellas que representan unas estimadas de probabilidad P y 1 - P) puede reenviarse a la misma memoria intermedia de contenedores 20. Además, la asociación de la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para la solicitud de contenedor actual con una memoria intermedia de contenedores particular puede adaptarse con el tiempo.

El selector de memoria intermedia de contenedores 18 puede determinar la memoria intermedia de contenedores 20 a la que se envía la solicitud para un contenedor 19 sobre la base de la medida asociada para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor solicitado actual. El conjunto de valores posibles para la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable puede ser finito y el selector de memoria intermedia de contenedores 18 puede contener una tabla que asocia exactamente una memoria intermedia de contenedores 20 con cada valor posible de la estimada de la probabilidad para el valor de contenedor menos probable o más probable, en donde diferentes valores para la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable pueden asociarse con la misma memoria intermedia de contenedores 20. El intervalo de valores posibles para la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable se puede subdividir en particiones para dar un número de intervalos, el selector de memoria intermedia de contenedores 18 determina el índice de intervalo para la medida actual para una estimada de la probabilidad para el valor de contenedor menos probable o más probable, y el selector de memoria intermedia de contenedores 18 contiene una tabla que asocia exactamente una memoria intermedia de contenedores 20 con cada valor posible para el índice de intervalo, en donde diferentes valores para el índice de intervalo pueden asociarse con la misma memoria intermedia de contenedores 20. En la asociación de la medida para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para la solicitud de contenedor actual con una memoria intermedia de contenedores particular se adapta con el tiempo.

Después de recibir un contenedor descodificado 25 a partir de la memoria intermedia de contenedores seleccionada 20, el selector de memoria intermedia de contenedores 18 potencialmente modifica el contenedor de entrada 25 y envía el contenedor de salida 26 - con un valor potencialmente modificado - a la unidad de binarización 14. La puesta en correspondencia de contenedores de entrada / salida del selector de memoria intermedia de contenedores 18 es la inversa de la puesta en correspondencia de contenedores de entrada / salida del selector de memoria intermedia de contenedores en el lado de codificador.

El selector de memoria intermedia de contenedores 18 se puede configurar para no modificar el valor del contenedor, es decir, el contenedor de salida 26 tiene siempre el mismo valor que el contenedor de entrada 25.

El selector de memoria intermedia de contenedores 18 puede determinar el valor de contenedor de salida 26 sobre la base del valor de contenedor de entrada 25 y la medida para una estimada de la probabilidad para uno de los dos valores de contenedor posibles para el contenedor solicitado actual que se asocia con la solicitud para un contenedor 17. El valor de contenedor de salida 26 puede establecerse igual al valor de contenedor de entrada 25 si la medida para la probabilidad para uno de los dos valores de contenedor posibles para la solicitud de contenedor actual es menor que (o menor que o igual a) un umbral particular; si la medida para la probabilidad para uno de los dos valores de contenedor posibles para la solicitud de contenedor actual es mayor que o igual a (o mayor que) un umbral particular, el valor de contenedor de salida 26 se modifica (es decir, se establece en el opuesto del valor de contenedor de entrada). El valor de contenedor de salida 26 puede establecerse igual al valor de contenedor de entrada 25 si la medida para la probabilidad para uno de los dos valores de contenedor posibles para la solicitud de contenedor actual es mayor que (o mayor que o igual a) un umbral particular; si la medida para la probabilidad para uno de los dos valores de contenedor posibles para la solicitud de contenedor actual es menor que o igual a (o menor que) un umbral particular, el valor de contenedor de salida 26 se modifica (es decir, se establece en el opuesto del valor de contenedor de entrada). El valor del umbral se puede corresponder con un valor de 0,5 para la probabilidad estimada para ambos valores de contenedor posibles.

El selector de memoria intermedia de contenedores 18 puede determinar el valor de contenedor de salida 26 sobre la base del valor de contenedor de entrada 25 y el identificador, que especifica una estimada para cual de los dos

valores de contenedor posibles representa el valor de contenedor menos probable o más probable para la solicitud de contenedor actual, que se asocia con la solicitud para un contenedor 17. El valor de contenedor de salida 26 puede establecerse igual al valor de contenedor de entrada 25 si el identificador especifica que el primero de los dos valores de contenedor posibles representa el valor de contenedor menos probable (o más probable) para la solicitud de contenedor actual, y el valor de contenedor de salida 26 se modifica (es decir, se establece en el opuesto del valor de contenedor de entrada) si el identificador especifica que el segundo de los dos valores de contenedor posibles representa el valor de contenedor menos probable (o más probable) para la solicitud de contenedor actual.

Tal como se ha descrito en lo que antecede, el selector de memoria intermedia de contenedores envía una solicitud para un contenedor 19 a una de las dos o más memorias intermedias de contenedores 20. Las memorias intermedias de contenedores 20 representan unas memorias intermedias de tipo primero en entrar primero en salir, que se alimentan con unas secuencias de contenedores descodificados 21 a partir de los descodificadores de contenedores conectados 22. Como respuesta a una solicitud para un contenedor 19 que se envía a una memoria intermedia de contenedores 20 a partir del selector de memoria intermedia de contenedores 18, la memoria intermedia de contenedores 20 retira el contenedor de su contenido que en primer lugar se alimentó a la memoria intermedia de contenedores 20 y lo envía al selector de memoria intermedia de contenedores 18. Los contenedores que se envían con anterioridad a la memoria intermedia de contenedores 20 se retiran con anterioridad y se envían al selector de memoria intermedia de contenedores 18.

Cada una de las dos o más memorias intermedias de contenedores 20 se conecta con exactamente un descodificador de contenedores 22 y cada descodificador de contenedores solo se conecta con una memoria intermedia de contenedores 20. Cada descodificador de contenedores 22 lee las palabras de código 23, las cuales representan unas secuencias de bits, a partir de un tren de bits parcial separado 24. El descodificador de contenedores convierte una palabra de código 23 en una secuencia de contenedores 21 que se envía a la memoria intermedia de contenedores conectada 20. El algoritmo de descodificación global convierte dos o más trenes de bits parciales 24 en un número de símbolos de origen descodificados, en donde el número de trenes de bits parciales es igual al número de memorias intermedias de contenedores y de descodificadores de contenedores y la descodificación de símbolos de origen es desencadenada por solicitudes para nuevos símbolos de origen. Un descodificador de contenedores 22 puede convertir las palabras de código 23 de un número variable de bits en una secuencia de un número variable de contenedores 21. Una ventaja de la configuración de PIPE anterior es que la descodificación de contenedores a partir de los dos o más trenes de bits parciales se puede realizar en paralelo (por ejemplo, para diferentes grupos de medidas de probabilidad), que reduce el tiempo de procesamiento para varias implementaciones.

Otra ventaja de la descodificación de PIPE anterior es que la descodificación de contenedores, la cual es realizada por los descodificadores de contenedores 22, se puede diseñar de forma específica para diferentes conjuntos de parámetros 17. En particular, la codificación y la descodificación de contenedores se puede optimizar (en términos de la complejidad y / o la eficiencia de codificación) para diferentes grupos de probabilidades estimadas. Por un lado, esto permite una reducción de la complejidad de codificación / descodificación en relación con algoritmos de codificación aritmética con una eficiencia de codificación similar. Por otro lado, esto permite una mejora de la eficiencia de codificación en relación con los algoritmos de codificación de VLC con una complejidad de codificación / descodificación similar. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación (es decir la puesta en correspondencia de secuencias de contenedores con palabras de código) para diferentes grupos de medidas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles 17 para la solicitud de contenedor actual. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes grupos de medidas para una estimada de la probabilidad para el valor de contenedor menos probable o más probable para el contenedor solicitado actual. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes códigos de protección de canal. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes esquemas de cifrado. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles 17 para el contenedor solicitado actual. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas para una estimada de la probabilidad para el valor de contenedor menos probable o más probable 17 para el contenedor solicitado actual. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas para una estimada de la probabilidad para uno de los dos valores de contenedor posibles 17 para el contenedor solicitado actual. Los descodificadores de contenedores 22 pueden implementar diferentes algoritmos de descodificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas para una estimada de la probabilidad para el valor de contenedor menos probable o más probable 17 para el contenedor solicitado actual.

Los descodificadores de contenedores 22 do la puesta en correspondencia inversa de los codificadores de contenedores correspondientes en el lado de codificador.

Los descodificadores de contenedores 22 - o uno o más de los descodificadores de contenedores - pueden representar motores de descodificación aritmética binaria.

Los descodificadores de contenedores 22 - o uno o más de los descodificadores de contenedores - pueden representar unos descodificadores por entropía que ponen en correspondencia directamente las palabras de código 23 con las secuencias de contenedores 21. Tales puestas en correspondencia se pueden implementar de forma eficiente y no requieren un motor de codificación aritmética compleja. La puesta en correspondencia de palabras de código con secuencias de contenedores ha de ser única. La puesta en correspondencia de las palabras de código 23 con las secuencias de contenedores 21 puede ser biyectiva. Los descodificadores de contenedores 10 - o uno o más de los descodificadores de contenedores - pueden representar unos descodificadores por entropía que ponen en correspondencia directamente unas palabras de código de longitud variable 23 con unas secuencias de longitud variable de contenedores 21. Las palabras de código de entrada pueden representar unos códigos libres de redundancia tales como códigos de Huffman generales o códigos de Huffman canónicos. En la tabla 3 se ilustran dos ejemplos para la puesta en correspondencia biyectiva de unos códigos libres de redundancia con unas secuencias de contenedores. Las palabras de código de entrada pueden representar unos códigos redundantes adecuados para la detección de errores y la recuperación de errores. Las palabras de código de entrada pueden representar códigos de cifrado.

Los descodificadores de contenedores 22 - o uno o más de los descodificadores de contenedores - pueden representar unos descodificadores por entropía que ponen en correspondencia directamente unas palabras de código de longitud fija 23 con unas secuencias de longitud variable de contenedores 21. Como alternativa, los descodificadores de contenedores 22 - o uno o más de los descodificadores de contenedores - representan unos descodificadores por entropía que ponen en correspondencia directamente las palabras de código de longitud variable 23 con unas secuencias de longitud fija de contenedores 21.

Por lo tanto, la figura 3 y 4 mostraron un codificador de PIPE para codificar una secuencia de símbolos de origen 1 y un descodificador de PIPE para reconstruir la misma. Es decir, el codificador de PIPE de la figura 3 se puede usar como el codificador de PIPE 104 en la figura 1a con la unidad de binarización 2 que actúa como el simbolizador 122, la unidad de asignación de parámetros 4 que actúa como la unidad de asignación 114, el selector de memoria intermedia de contenedores 6 que actúa como selector 120, y el par de la memoria intermedia de contenedores 8 y el codificador de contenedores 10 conectado en serie que actúa como uno respectivo de los codificadores por entropía 116 cada uno de los cuales emite unos trenes de bits 12 que se corresponden con los trenes de bits 118 en la figura 1a. Como queda claro a partir de la comparación de la figura 3 y la figura 1, la unidad de asignación 114 de la figura 1a puede tener su entrada como alternativa conectada con el lado de entrada del simbolizador 122 en lugar del lado de salida de este último. De forma similar, el descodificador de PIPE de la figura 4 se puede usar como el descodificador de PIPE 202 en la figura 2a con unos trenes de bits parciales 24 que se corresponden con los trenes de bits 216 en la figura 2, los pares de la memoria intermedia 20 y el descodificador de contenedores 22 conectado en serie que se corresponden con los descodificadores por entropía individuales 210, el selector de memoria intermedia de contenedores 18 que actúa como selector 214, la unidad de asignación de parámetros 16 que actúa como la unidad de asignación 212 y la unidad de binarización 14 que actúa como el desimbolizador 222. De nuevo, una comparación entre la figura 2a y la figura 4 deja claro que la interconexión entre el desimbolizador 222, la unidad de asignación 212 y el selector 214 puede configurarse de forma diferente, de tal modo que como alternativa, las conexiones de la figura 2a se modifican para corresponderse con las que se muestran en la figura 4.

El codificador de PIPE de la figura 3 comprende una unidad de asignación 4 configurada para asignar un número de parámetros 5 a cada símbolo alfabético de la secuencia de símbolos alfabéticos 3. La asignación se basa en información contenida dentro de símbolos alfabéticos previos de la secuencia de símbolos alfabéticos tales como la categoría del elemento de sintaxis 1 a la cual pertenece la representación - tal como binarización - del símbolo alfabético actual y la cual, de acuerdo con la estructura sintáctica de los elementos de sintaxis 1, es de esperar en la actualidad, expectativa que, a su vez, es deducible a partir de la historia de los elementos de sintaxis 1 y los símbolos alfabéticos 3 previos. Además, el codificador comprende una pluralidad de codificadores por entropía 10 cada uno de los cuales está configurado para convertir los símbolos alfabéticos 3 reenviados al codificador por entropía respectivo en un tren de bits respectivo 12, y un selector 6 configurado para reenviar cada símbolo alfabético 3 a uno seleccionado de la pluralidad de codificadores por entropía 10, dependiendo la selección del número de parámetros 5 asignado al símbolo alfabético respectivo 3. El descodificador de PIPE de la figura 4 comprende una pluralidad de descodificadores por entropía 22, cada uno de los cuales está configurado para convertir un tren de bits respectivo 23 en símbolos alfabéticos 21; una unidad de asignación 16 configurada para asignar un número de parámetros 17 a cada símbolo alfabético 15 de una secuencia de símbolos alfabéticos que se va a reconstruir sobre la base de información contenida dentro de símbolos alfabéticos previamente reconstruidos de la secuencia de símbolos alfabéticos (véanse 26 y 27 en la figura 4); y un selector 18 que está configurado para recuperar cada símbolo alfabético de la secuencia de símbolos alfabéticos que se va a reconstruir a partir de uno seleccionado de la pluralidad de descodificadores por entropía 22, dependiendo la selección del número de parámetros que se definen para el símbolo alfabético respectivo. La unidad de asignación 16 puede configurarse de

tal modo que el número de parámetros asignado a cada símbolo alfabético comprende, o es, una medida para una estimada de una probabilidad de distribución entre los posibles valores de símbolo alfabético que puede asumir un símbolo alfabético respectivo. La secuencia de símbolos alfabéticos que se va a reconstruir puede ser de un alfabeto binario y la unidad de asignación 16 puede configurarse de tal modo que la estimada de la distribución de probabilidad consiste en una medida para una estimada de una probabilidad de un valor de contenedor menos probable o más probable de los dos valores de contenedor posibles del alfabeto binario y un identificador que especifica una estimada para cual de los dos valores de contenedor posibles representa el valor de contenedor menos probable o más probable. La unidad de asignación 16 puede configurarse adicionalmente para asignar de forma interna un contexto a cada símbolo alfabético de la secuencia de símbolos alfabéticos 15 que se va a reconstruir sobre la base de la información contenida dentro de símbolos alfabéticos previamente reconstruidos de la secuencia de símbolos alfabéticos que se va a reconstruir con cada contexto teniendo una estimada de distribución de probabilidad respectiva asociado a ello, y adaptar la estimada de distribución de probabilidad para cada contexto a una estadística de símbolos real sobre la base de los valores de símbolo de símbolos alfabéticos previamente reconstruidos a los que se asigna el contexto respectivo. El contexto puede tener en cuenta una relación espacial o proximidades de posiciones a las cuales pertenecen los elementos de sintaxis tal como en la codificación de vídeo o de imágenes, o incluso en tablas en el caso de las aplicaciones financieras. Entonces, la medida para la estimada de la distribución de probabilidad para cada símbolo alfabético puede determinarse sobre la base de la estimada de distribución de probabilidad asociada con el contexto asignado al símbolo alfabético respectivo tal como mediante la cuantificación de la estimada de distribución de probabilidad asociada con el contexto asignado con el símbolo alfabético respectivo a una de una pluralidad de representantes de estimada de distribución de probabilidad con el fin de obtener la medida para la estimada de la distribución de probabilidad. El selector puede configurarse de tal modo que se define una asociación sobreyectiva entre la pluralidad de codificadores por entropía y la pluralidad de representantes de estimada de distribución de probabilidad, es decir cada codificador por entropía tiene por lo menos una representante de estimada de distribución de probabilidad asociado con el mismo, pero más de una representante de estimada de distribución de probabilidad se pueden asociar con un codificador por entropía. La asociación incluso puede ser biyectiva. El selector 18 se puede configurar para cambiar una puesta en correspondencia de cuantificación de un intervalo de las estimadas de distribución de probabilidad a la pluralidad de representantes de estimada de distribución de probabilidad de una forma determinista previamente determinada dependiendo de símbolos alfabéticos previamente reconstruidos de la secuencia de símbolos alfabéticos, con el tiempo. Es decir, el selector 18 puede cambiar los tamaños de paso de cuantificación, es decir los intervalos de distribuciones de probabilidad puestos en correspondencia con los índices de probabilidad individuales los cuales, a su vez, pueden asociarse de forma sobreyectiva con los descodificadores por entropía individuales. La pluralidad de descodificadores por entropía 22, a su vez, se pueden configurar para adaptar su forma de convertir símbolos alfabéticos en unos trenes de bits sensibles a un cambio en la puesta en correspondencia de cuantificación. Por ejemplo, cada descodificador por entropía 22 se puede optimizar para, es decir puede tener una tasa de compresión óptima para, una determinada estimada de distribución de probabilidad dentro del intervalo de cuantificación de estimadas de distribución de probabilidad respectivo, y puede cambiar su correspondencia de secuencia de símbolos / palabras de código con el fin de adaptar la posición de esta determinada estimada de distribución de probabilidad dentro del intervalo de cuantificación de estimadas de distribución de probabilidad respectivo con un cambio de este último con el fin de optimizarse. El selector se puede configurar para cambiar la puesta en correspondencia de cuantificación de tal modo que las tasas mediante las cuales se recuperan los símbolos alfabéticos a partir de la pluralidad de descodificadores por entropía, se hacen menos dispersas. En lo que respecta a la unidad de binarización 14 se hace notar que el mismo puede no usarse si los elementos de sintaxis ya son binarios. Además, dependiendo del tipo de descodificador 22, no es necesaria la existencia de las memorias intermedias 20. Además, las memorias intermedias pueden estar integradas dentro de los descodificadores.

Hasta el momento, se han descrito en lo que antecede más detalles para el codificador de PIPE 104 y el descodificador de PIPE 202 en la figura 1a y 2, con respecto a la figura 3 y 4, que, si se implementa fácilmente en los aparatos de la figura 1a y 2, conducen a una salida de tren de bits en paralelo en la cual los trenes de bits parciales de VLC y de PIPE se transportan en paralelo. En lo sucesivo, se describen posibilidades, cómo combinar los trenes de bits parciales de PIPE para transmitirse, entonces, a lo largo del tren de bits de VLC en paralelo, o con una intercalación en segundo lugar de ambos trenes de bits, es decir el tren de bits de VLC y el tren de bits de PIPE intercalado.

Terminación de secuencias de símbolos de origen finitas

En los codificadores y descodificadores de PIPE, la codificación y la descodificación se pueden realizar para un conjunto finito de símbolos de origen. A menudo una cierta cantidad de datos tal como una imagen fija, una trama o campo de una secuencia de vídeo, una rebanada de una imagen, una rebanada de una trama o un campo de una secuencia de vídeo, o un conjunto de muestras de audio sucesivas, etc. se codifica. Para conjuntos finitos de símbolos de origen, en general, han de determinarse los trenes de bits parciales que se crean en el lado de codificador, es decir, se ha de asegurar que todos los símbolos de origen se pueden descodificar a partir de los trenes de bits parciales transmitidos o almacenados. Después de que el último contenedor se haya insertado en la memoria intermedia de contenedores 8 correspondiente, el codificador de contenedores 10 ha de asegurar que una

palabra de código completa se escribe en el tren de bits parcial 12. Si el codificador de contenedores 10 representa un motor de codificación aritmética binaria, la palabra de código aritmética ha de determinarse. Si el codificador de contenedores 10 representa un codificador por entropía que implementa una puesta en correspondencia directa de secuencias de contenedores con palabras de código, la secuencia de contenedores que se almacena en la memoria intermedia de contenedores después de la escritura del último contenedor en la memoria intermedia de contenedores podría no representar una secuencia de contenedores que se asocia con una palabra de código (es decir, esta podría representar un prefijo de dos o más secuencias de contenedores que están asociadas con palabras de código). En caso de este tipo, cualquiera de las palabras de código que están asociadas con una secuencia de contenedores que contiene la secuencia de contenedores en la memoria intermedia de contenedores como prefijo ha de escribirse en el tren de bits parcial (se ha de evacuar la memoria intermedia de contenedores). Esto se podría realizar mediante la inserción de contenedores con un valor particular o uno arbitrario en la memoria intermedia de contenedores hasta que se escribe una palabra de código. El codificador de contenedores puede seleccionar una de las palabras de código con una longitud mínima (además de la propiedad de que la secuencia de contenedores asociada debe contener la secuencia de contenedores en la memoria intermedia de contenedores como prefijo). En el lado de descodificador, el descodificador de contenedores 22 puede descodificar más contenedores de los que se requieren para la última palabra de código en un tren de bits parcial; estos contenedores no son solicitados por el selector de memoria intermedia de contenedores 18 y se descartan y se ignoran. La descodificación del conjunto finito de símbolos se controla mediante solicitudes para símbolos de origen descodificados; si no se solicita símbolo de origen adicional alguno para una cantidad de datos, la descodificación se termina.

Transmisión y multiplexación de los trenes de bits parciales

Los trenes de bits parciales 12 que se crean por el codificador de PIPE se pueden transmitir por separado, o estos se pueden multiplexar para dar un único tren de bits, o las palabras de código de los trenes de bits parciales se pueden intercalar en un único tren de bits.

Cada tren de bits parcial para una cantidad de datos se puede escribir en un paquete de datos. La cantidad de datos puede ser un conjunto arbitrario de símbolos de origen tales como una imagen fija, un campo o trama de una secuencia de vídeo, una rebanada de una imagen fija, una rebanada de un campo o trama de una secuencia de vídeo, o una trama de muestras de audio, etc.

Dos o más de los trenes de bits parciales para una cantidad de datos o todos los trenes de bits parciales para una cantidad de datos se pueden multiplexar para dar un paquete de datos. La estructura de un paquete de datos que contiene unos trenes de bits parciales multiplexados se ilustra en la figura 5. Es decir, el paquete de datos que se muestra en la figura 5 podría ser parte del tren intercalado intermedio 132 y 234, de forma respectiva.
El paquete de datos 300 consiste en un encabezamiento y una partición para los datos de cada tren de bits parcial (para la cantidad considerada de datos). El encabezamiento 300 del paquete de datos contiene indicaciones para la subdivisión en particiones de (el resto de) el paquete de datos en segmentos de los datos de tren de bits 302. Además de las indicaciones para la subdivisión en particiones, el encabezamiento puede contener una información adicional. Las indicaciones para la subdivisión en particiones del paquete de datos pueden ser las ubicaciones del comienzo de los segmentos de datos en unidades de bits o bytes o múltiples de bits o múltiples de bytes. Las ubicaciones del comienzo de los segmentos de datos se pueden codificar como valores absolutos en el encabezamiento del paquete de datos, o bien en relación con el comienzo del paquete de datos o bien en relación con el final del encabezamiento o en relación con el comienzo del paquete de datos previo. Las ubicaciones del comienzo de los segmentos de datos se pueden codificar de forma diferencial, es decir, solo la diferencia entre el comienzo real de un segmento de datos y una predicción para el comienzo del segmento de datos se codifica. La predicción se puede obtener sobre la base de una información ya conocida o transmitida tal como el tamaño global del paquete de datos, el tamaño del encabezamiento, el número de segmentos de datos en el paquete de datos, la ubicación del comienzo de segmentos de datos precedentes. La ubicación del comienzo del primer paquete de datos puede no codificarse, sino inferirse sobre la base del tamaño del encabezamiento de paquete de datos. En el lado de descodificador, las indicaciones de partición transmitidas se usan para obtener el comienzo de los segmentos de datos. Los segmentos de datos se usan entonces como trenes de bits parciales y los datos que están contenidos en los segmentos de datos se alimentan a los descodificadores de contenedores correspondientes en orden secuencial.

Hay varias alternativas para multiplexar los trenes de bits parciales 12 en un paquete de datos. Una alternativa, la cual puede reducir la información conexa requerida, en particular para los casos en los que los tamaños de los trenes de bits parciales son muy similares, se ilustra en la figura 6. La cabida útil del paquete de datos, es decir, el paquete de datos 310 sin su encabezamiento 311, se subdivide en particiones para dar unos segmentos 312 de una forma previamente definida. Como un ejemplo, la cabida útil de paquete de datos se puede subdividir en particiones para dar unos segmentos del mismo tamaño. Entonces cada segmento se asocia con un tren de bits parcial o con la primera parte de un tren de bits parcial 313. Si un tren de bits parcial es mayor que el segmento de datos asociado, su resto 314 se coloca en el espacio sin usar al final de otros segmentos de datos. Esto se puede realizar de una forma tal que la parte restante de un tren de bits se inserta en el orden inverso (comenzando a partir del final del

segmento de datos), que reduce la información conexa. La asociación de los restos de los trenes de bits parciales con unos segmentos de datos y, cuando más de un resto se añade a un segmento de datos, el punto de inicio para uno o más de los restos se ha de señalizar en el interior del tren de bits, por ejemplo en el encabezamiento de paquete de datos.

Intercalación de palabras de código de longitud variable

Para algunas aplicaciones, la multiplexación que se ha descrito en lo que antecede de los trenes de bits parciales 12 (para una cantidad de símbolos de origen) en un paquete de datos puede tener las siguientes desventajas: Por un lado, para unos paquetes de datos pequeños, el número de bits para la información conexa que se requiere para señalizar la subdivisión en particiones se puede volver significativa en relación con los datos reales en los trenes de bits parciales, lo que en última instancia reduce la eficiencia de codificación. Por otro lado, la multiplexación puede no ser adecuada para las aplicaciones que requieren un bajo retardo (por ejemplo, para aplicaciones de videoconferencia). Con la multiplexación descrita, el codificador de PIPE no puede iniciar la transmisión de un paquete de datos antes de que los trenes de bits parciales se hayan creado completamente, debido a que las ubicaciones del comienzo de las particiones no se conocían con anterioridad. Además, en general, el descodificador de PIPE ha de esperar hasta que este recibe el comienzo del último segmento de datos antes de que este pueda iniciar la descodificación de un paquete de datos. Para aplicaciones como sistemas de videoconferencia, estos retardos pueden sumarse a un retardo global adicional del sistema de varias imágenes de vídeo (en particular para unas tasas de bits que se encuentran cerca de la tasa de bits de transmisión y para codificadores / descodificadores que requieren casi el intervalo de tiempo entre dos imágenes para codificar / descodificar una imagen), que es crítico para tales aplicaciones. Con el fin de superar las desventajas para determinadas aplicaciones, el codificador de PIPE puede configurarse de una forma tal que las palabras de código que se generan mediante los dos o más codificadores de contenedores se intercalan para dar un único tren de bits. El tren de bits con las palabras de código intercaladas puede enviarse directamente al descodificador (cuando se desprecia un pequeño retardo de memoria intermedia, véase en lo sucesivo). En el lado de descodificador de PIPE, los dos o más descodificadores de contenedores leen las palabras de código directamente a partir del tren de bits en el orden de descodificación; la descodificación se puede iniciar con el primer bit recibido. Además, no se requiere información conexa alguna para señalizar la multiplexación (o intercalación) de los trenes de bits parciales.

La estructura básica de un codificador de PIPE con una intercalación de palabras de código se muestra en la figura 7. Los codificadores de contenedores 10 no escriben las palabras de código directamente en los trenes de bits parciales, sino que se conectan con una única memoria intermedia de palabras de código 29, a partir de las cual se escriben palabras de código en el tren de bits 34 en el orden de codificación. Los codificadores de contenedores 10 envían solicitudes para una o más nuevas entradas de memoria intermedia de palabras de código 28 a la memoria intermedia de palabras de código 29 y posteriormente envían las palabras de código 30 a la memoria intermedia de palabras de código 29, que se almacenan en las entradas de memoria intermedia reservadas. Las palabras de código (en general de longitud variable) 31 de la memoria intermedia de palabras de código 29 se acceden por una unidad de escritura de palabras de código 32, que escribe los bits 33 correspondientes en el tren de bits producido 34. La memoria intermedia de palabras de código 29 opera como una memoria intermedia de tipo primero en entrar primero en salir; las entradas de palabra de código que se reservan con anterioridad se escriben con anterioridad en el tren de bits.

En una generalización adicional, son posibles múltiples memorias intermedias de palabras de código y trenes de bits parciales 12, en donde el número de memorias intermedias de palabras de código es menor que el número de codificadores de contenedores. Un codificador de contenedores 10 reserva una o más palabras de código en la memoria intermedia de palabras de código 29, mediante lo cual la reserva de las una o más palabras de código en la memoria intermedia de palabras de código es desencadenada por determinados eventos en la memoria intermedia de contenedores conectada 8. La memoria intermedia de palabras de código 29 se puede operar de una forma tal que el descodificador de PIPE puede descodificar de forma instantánea el tren de bits 34 el cual se corresponde con 132 en la figura 1a y 134 en la figura 2, de forma respectiva. El orden de codificación en el cual las palabras de código se escriben en el tren de bits es el mismo que el orden en el cual las palabras de código correspondientes se reservan en la memoria intermedia de palabras de código. Cada codificador de contenedores 10 puede reservar una palabra de código, con la reserva siendo desencadenado por un determinado evento en la memoria intermedia de contenedores conectada. Cada codificador de contenedores 10 puede reservar más de una palabra de código, con la reserva siendo desencadenado por un determinado evento en la memoria intermedia de contenedores conectada. Los codificadores de contenedores 10 pueden reservar una cantidad diferente de palabras de código, en donde la cantidad de palabras de código que son reservadas por un codificador de contenedores particular puede ser dependiente del codificador de contenedores particular y / u otras propiedades del codificador de contenedores / memoria intermedia de contenedores particular (tal como la medida de probabilidad asociada, el número de bits ya escritos, etc.).

La memoria intermedia de palabras de código puede operar tal como sigue. Si un nuevo contenedor 7 se envía a una memoria intermedia de contenedores particular 8 y el número de contenedores ya almacenados en la memoria

intermedia de contenedores es cero y en la actualidad no hay palabra de código alguna reservada en la memoria intermedia de palabras de código para el codificador de contenedores que se conecta con la memoria intermedia de contenedores particular, el codificador de contenedores conectado 10 envía una solicitud a la memoria intermedia de palabras de código, mediante lo cual una o más entradas de palabra de código se reservan en la memoria intermedia de palabras de código 29 para el codificador de contenedores particular. Las entradas de palabra de código pueden tener un número variable de bits; un umbral superior para el número de bits en una entrada de memoria intermedia viene dado por lo general por el tamaño de palabra de código máximo para el codificador de contenedores correspondiente. La siguiente palabra de código o las siguientes palabras de código que se producen por el codificador de contenedores (para lo cual la entrada de palabra de código o entradas de palabra de código se han reservado) se almacenan en la entrada o entradas reservadas de la memoria intermedia de palabras de código. Si todas las entradas de memoria intermedia reservadas en la memoria intermedia de palabras de código para un codificador de contenedores particular se llenan con palabras de código y el siguiente contenedor se envía a la memoria intermedia de contenedores que se conecta con el codificador de contenedores particular, una o más nuevas palabras de código se reservan en la memoria intermedia de palabras de código para el codificador de contenedores particular, etc. La memoria intermedia de palabras de código 29 representa una memoria intermedia de tipo primero en entrar primero en salir de una cierta forma. Las entradas de memoria intermedia se reservan en orden secuencial. Las palabras de código para las que las entradas de memoria intermedia correspondientes se han reservado con anterioridad se escriben con anterioridad en el tren de bits. La unidad de escritura de palabras de código 32 comprueba el estatus de la memoria intermedia de palabras de código 29, o bien de forma continua o bien después de que una palabra de código 30 se escriba en la memoria intermedia de palabras de código 29. Si la primera entrada de memoria intermedia contiene una palabra de código completa (es decir, la entrada de memoria intermedia no está reservada, sino que incluye una palabra de código), la palabra de código 31 correspondiente y la entrada de memoria intermedia correspondiente se retiran de la memoria intermedia de palabras de código 20 y los bits de la palabra de código 33 se escriben en el tren de bits. Este proceso se repite hasta que la primera entrada de memoria intermedia no contiene una palabra de código (es decir, esta está reservada o es libre). Al final del proceso de descodificación, es decir, si se han procesado todos los símbolos de origen de la cantidad considerada de datos, se ha de evacuar la memoria intermedia de palabras de código. Para ese proceso de evacuación, lo sucesivo es de aplicación para cada memoria intermedia de contenedores / codificador de contenedores como primera etapa: Si la memoria intermedia de contenedores contiene, de hecho, unos contenedores, un contenedor con un valor particular o uno arbitrario se añade hasta que la secuencia de contenedores resultante representa una secuencia de contenedores que se asocia con una palabra de código (tal como se ha hecho notar en lo que antecede, una forma preferida de adición de contenedores es añadir aquellos valores de contenedor que producen la palabra de código posible más corta - o una de ellas - que se asocia con una secuencia de contenedores que contiene el para el contenido original de la memoria intermedia de contenedores como prefijo), entonces la palabra de código se escribe en la siguiente entrada de memoria intermedia reservada para el codificador de contenedores correspondiente (y la correspondiente) memoria intermedia de contenedores se vacía. Si se ha reservado más de una entrada de memoria intermedia para uno o más codificadores de contenedores, la memoria intermedia de palabras de código puede seguir conteniendo entradas de palabra de código reservadas. En ese caso, estas entradas de palabra de código se llenan con unas palabras de código arbitrarias pero válidas para los codificadores de contenedores correspondientes. Preferiblemente, se inserta la palabra de código válida más corta o una de las palabras de código válidas más cortas (de haber una multitud). Por último, todas las palabras de código restantes en la memoria intermedia de palabras de código se escriben en el tren de bits.

En la figura 8 se ilustran dos ejemplos para el estatus de la memoria intermedia de palabras de código. En el ejemplo (a), la memoria intermedia de palabras de código contiene 2 entradas que se llenan con una palabra de código y 5 entradas reservadas. Además, se marca la siguiente entrada de memoria intermedia libre. La primera entrada se llena con una palabra de código (es decir, el codificador de contenedores 2 acababa de escribir una palabra de código en una entrada previamente reservada). En la siguiente etapa, esta palabra de código se retirará de la memoria intermedia de palabras de código y se escribirá en el tren de bits. Entonces, la primera palabra de código reservada para el codificador de contenedores 3 es la primera entrada de memoria intermedia, pero esta entrada no se puede retirar de la memoria intermedia de palabras de código, debido a que esta solo está reservada, pero no se ha escrito palabra de código alguna en esta entrada. En el ejemplo (b), la memoria intermedia de palabras de código contiene 3 entradas que se llenan con una palabra de código y 4 entradas reservadas. La primera entrada se marca como reservada y, por lo tanto, la unidad de escritura de palabras de código no puede escribir una palabra de código en el tren de bits. A pesar de que 3 palabras de código están contenidas en la memoria intermedia de palabras de código, la unidad de escritura de palabras de código ha de esperar hasta que se escriba una palabra de código en la primera entrada de memoria intermedia reservada para el codificador de contenedores 3. Obsérvese que las palabras de código deben escribirse en el orden en el cual fueron reservadas, con el fin de ser capaces de invertir el proceso en el lado de descodificador (véase en lo sucesivo).

La estructura básica de un descodificador de PIPE con una intercalación de palabras de código se muestra en la figura 9. Los descodificadores de contenedores 10 no leen las palabras de código directamente a partir de trenes de bits parciales separados, sino que están conectados con una memoria intermedia de bits 38, a partir de la que las palabras de código 37 se leen en el orden de codificación. Se debería hacer notar que la memoria intermedia de bits

38 no se requiere necesariamente, debido a que las palabras de código también se podrían leer directamente a partir del tren de bits. La memoria intermedia de bits 38 se incluye principalmente en la ilustración para separar claramente diferentes aspectos de la cadena de procesamiento. Los bits 39 del tren de bits 40 con palabras de código intercaladas, lo que por lo tanto se corresponde con el tren de bits 234 en la figura 2, se insertan de forma secuencial en la memoria intermedia de bits 38, la cual representa una memoria intermedia de tipo primero en entrar primero en salir. Si un descodificador de contenedores particular 22 recibe una solicitud para una o más secuencias de contenedores 35, el descodificador de contenedores 22 lee una o más palabras de código 37 a partir de la memoria intermedia de bits 38 por medio de solicitudes para bits 36. El descodificador de PIPE puede descodificar de forma instantánea los símbolos de origen. Obsérvese que el codificador de PIPE (tal como se ha descrito en lo que antecede) ha de asegurar, al operar de forma conveniente la memoria intermedia de palabras de código, que las palabras de código se escriben en el mismo orden en el tren de bits en el que estas son solicitadas por los descodificadores de contenedores. En el descodificador de PIPE, la totalidad del proceso de descodificación es desencadenado por solicitudes para símbolos de origen. Parámetros como el número de palabras de código que son reservados en el lado de codificador por un codificador de contenedores particular y el número de palabras de código que se leen por el descodificador de contenedores correspondiente han de ser los mismos.

En una generalización adicional, son posibles múltiples memorias intermedias de palabras de código y trenes de bits parciales, en donde el número de memorias intermedias de bits es menor que el número de descodificadores de contenedores. Un descodificador de contenedores 22 lee una o más palabras de código a partir de la memoria intermedia de bits 38 en un instante en el tiempo, mediante lo cual la lectura de las una o más palabras de código a partir de la memoria intermedia de bits es desencadenada por determinados eventos en la memoria intermedia de contenedores conectada 20. El descodificador se puede operar de una forma tal que una o más palabras de código se leen cuando una solicitud para un contenedor 19 se envía a una memoria intermedia de contenedores particular 20 y la memoria intermedia de contenedores no contiene contenedor alguno. Pero también es posible desencadenar la lectura de palabras de código mediante otros eventos, por ejemplo si el número de contenedores en la memoria intermedia de contenedores se encuentra por debajo de un umbral previamente definido. Cada descodificador de contenedores 22 puede leer una palabra de código, con la lectura siendo desencadenado por un determinado evento en la memoria intermedia de contenedores conectada. Como alternativa, cada descodificador de contenedores 22 puede leer más de una palabra de código, con la lectura siendo desencadenado por un determinado evento en la memoria intermedia de contenedores conectada. Los descodificadores de contenedores 22 pueden ser leídos una cantidad diferente de palabras de código, en donde la cantidad de palabras de código que se leen por un descodificador de contenedores particular puede ser dependiente del descodificador de contenedores particular y / u otras propiedades del descodificador de contenedores / memoria intermedia de contenedores particular (tal como la medida de probabilidad asociada, el número de bits ya leídos, etc.).

La lectura de palabras de código a partir de la memoria intermedia de bits se puede operar tal como sigue. Si una nueva solicitud de contenedor 19 se envía a partir del selector de memoria intermedia de contenedores 18 a una memoria intermedia de contenedores particular 20 y el número de contenedores en la memoria intermedia de contenedores es cero, el descodificador de contenedores conectado 22 lee una o más palabras de código 37 a partir de la memoria intermedia de bits 38, por medio de solicitud de bit 36 en la memoria intermedia de bits 38. El descodificador de contenedores 22 convierte las palabras de código 37 leídas en las secuencias de contenedores 21 y almacena estas secuencias de contenedores en la memoria intermedia de contenedores conectada 20. Como respuesta final a la solicitud para un contenedor 19, el contenedor insertado en primer lugar se retira de la memoria intermedia de contenedores 20 y se envía al selector de memoria intermedia de contenedores 18. Como respuesta a las solicitudes de contenedor adicionales, los contenedores restantes en la memoria intermedia de contenedores son removed hasta que la memoria intermedia de contenedores está vacía. Una solicitud de contenedor adicional desencadena que el descodificador de contenedores lea una o más nuevas palabras de código a partir de la memoria intermedia de bits, etc. La memoria intermedia de bits 38 representa una memoria intermedia de tipo primero en entrar primero en salir de un tamaño previamente definido y se llena continuamente con los bits 39 a partir del tren de bits 40. Con el fin de asegurar que las palabras de código se escriben en el tren de bits de la misma forma que estas son solicitadas por el proceso de descodificación, la memoria intermedia de palabras de código en el lado de codificador se puede operar de la forma que se ha descrito en lo que antecede.

Por lo tanto, cada uno de la pluralidad de descodificadores por entropía puede ser un descodificador de longitud variable que está configurado para poner en correspondencia palabras de código de longitudes fijas con unas secuencias de símbolos de longitudes variables, y una entrada de palabra de código tal como la salida de la memoria intermedia de palabras de código 43 puede proporcionarse para recibir un único tren de palabras de código intercaladas. La pluralidad de descodificadores por entropía 22 se pueden configurar para recuperar las palabras de código de la entrada de palabra de código en un orden secuencial dependiendo de un orden en el cual los símbolos de la secuencia de símbolos que se va a reconstruir como recuperada por el selector 18 a partir de la pluralidad de descodificadores por entropía dan como resultado una nueva secuencia de símbolos que va a ponerse en correspondencia a partir de una nueva palabra de código en los descodificadores por entropía respectivos.

Intercalación de palabras de código de longitud variable con una restricción de retardo baja

La intercalación de palabras de código descrita para la codificación de PIPE no requiere que se envíe información alguna de subdivisión en particiones como información conexa. Y debido a que las palabras de código se intercalan en el tren de bits, el retardo es en general pequeño. No obstante, no se garantiza que se obedezca una restricción de retardo particular (por ejemplo, especificada por un número máximo de bits que se almacenan en la memoria intermedia de palabras de código). Además, el tamaño de memoria intermedia requerido para la memoria intermedia de palabras de código se puede volver, en teoría, muy grande. Cuando se considera el ejemplo en la figura 8(b), podría ser posible que no se enviara contenedor adicional alguno a la memoria intermedia de contenedores 3 y, por lo tanto, el codificador de contenedores 3 no enviará palabra de código nueva alguna a la memoria intermedia de palabras de código hasta que se aplique el proceso de evacuación al final del paquete de datos. Entonces todas las palabras de código para los codificadores de contenedores 1 y 2 tendrían que esperar hasta el final del paquete de datos, antes de que estas se puedan escribir en el tren de bits. Este inconveniente puede sortearse mediante la adición de un mecanismo adicional al proceso de codificación de PIPE (y también al proceso de descodificación de PIPE tal como se describe posteriormente). El concepto básico de ese mecanismo adicional es que si una medida en relación con el retardo o un límite superior del retardo (véase en lo sucesivo) supera un umbral especificado, la primera entrada de memoria intermedia reservada se llena mediante la evacuación de la memoria intermedia de contenedores correspondiente (usando un mecanismo similar al del final de un paquete de datos). Mediante un mecanismo de este tipo, el número de entradas de memoria intermedia en espera se reduce hasta que la medida de retardo asociada es menor que el umbral especificado. En el lado de descodificador, se han de descartar los contenedores que se han insertado en el lado de codificador con el fin de obedecer la restricción de retardo. Para este descarte de contenedores se puede usar básicamente el mismo mecanismo que en el lado de codificador. En lo sucesivo se describen dos posibilidades para un control de retardo de este tipo.

En una posibilidad, la medida para el retardo (o un límite superior del retardo) es el número de entradas de memoria intermedia activas en la memoria intermedia de palabras de código, en donde el número de entradas de memoria intermedia activas es el número de entradas de memoria intermedia reservadas más el número de entradas de memoria intermedia que contienen palabras de código. Obsérvese que la primera entrada de memoria intermedia es siempre una entrada de memoria intermedia reservada o una entrada de memoria intermedia libre, debido a que si la primera entrada de memoria intermedia contiene una palabra de código, esta palabra de código se escribe en el tren de bits. Si por ejemplo, el máximo retardo de memoria intermedia admitido (según sea determinado por la aplicación) es D bits y el tamaño de palabra de código máximo para todos los codificadores de contenedores es L, un límite inferior para el número máximo de palabras de código que puede estar contenido en la memoria intermedia de palabras de código sin violar la restricción de retardo se puede calcular por $N = D / L$. La medida de retardo D en bits no se requiere por el sistema, pero el número máximo de palabras de código N ha de ser conocido tanto por el codificador como por el descodificador. El número máximo de entradas de memoria intermedia de palabras de código N puede estar fijado por la aplicación. Como alternativa, el número máximo de entradas de memoria intermedia de palabras de código N se puede señalizar en el interior del tren de bits, por ejemplo, en el encabezamiento del paquete de datos (o el encabezamiento de rebanada) o en un conjunto de parámetros, el cual se incluye en el tren de bits. Si un codificador de contenedores 10 envía una solicitud para la reserva de una o más nuevas entradas de memoria intermedia a la memoria intermedia de palabras de código 29, el siguiente proceso se ejecuta antes de que se reserve una nueva entrada de memoria intermedia de palabras de código (es decir, se ejecuta múltiples veces si múltiples entradas de memoria intermedia de palabras de código son reservadas por una solicitud): Si el número de entradas de memoria intermedia activas en la actualidad más 1 (teniendo en cuenta la entrada de memoria intermedia que se reservará a continuación) es mayor que el número máximo de entradas de memoria intermedia de palabras de código N, la primera entrada de memoria intermedia (la cual está reservada) se evacúa por el proceso que se describe en lo sucesivo hasta que el número de entradas de memoria intermedia activas en la actualidad más 1 sea menor que o igual al número máximo de entradas de memoria intermedia de palabras de código N. La evacuación de una entrada de memoria intermedia reservada es similar a la evacuación al final de un paquete de datos: El codificador de contenedores 10 que ha reservado la primera entrada de memoria intermedia correspondiente se evacúa mediante la adición de contenedores con unos valores particulares o arbitrarios a la memoria intermedia de contenedores conectada 8 hasta que la secuencia de contenedores resultante representa una secuencia de contenedores que se asocia con una palabra de código, entonces la palabra de código se escribe en la entrada de memoria intermedia reservada y, en última instancia, se añade al tren de bits (al tiempo que se vacía la memoria intermedia de contenedores y se retira la entrada de memoria intermedia previamente reservada). Tal como se ha mencionado en lo que antecede, una forma preferida para añadir contenedores a la memoria intermedia de contenedores es añadir aquellos contenedores que producen la palabra de código posible más corta. En el lado de descodificador, se ejecuta un proceso similar para descartar los contenedores que se han añadido para obedecer la restricción de retardo. Por lo tanto, el descodificador mantiene un contador C que recuenta las palabras de código que se han leído a partir de la memoria intermedia de bits (este contador puede mantenerse en la memoria intermedia de bits). Este contador C se inicializa (por ejemplo, con cero) en el comienzo de la descodificación de un paquete de datos y se incrementa en uno después de que se lea una palabra de código. Además, cada descodificador de contenedores 22 contiene un contador Cx, que almacena el valor del contador de palabras de código C antes de que la última palabra de código sea leída por el descodificador de contenedores 22 correspondiente. Es decir, cuando un descodificador de contenedores particular 22 lee una nueva palabra de código,

su contador *Cx* se establece igual a *C* como una primera etapa y entonces la palabra de código se lee a partir de la memoria intermedia de bits. Cuando una solicitud para un contenedor 19 se envía a una memoria intermedia de contenedores particular 20 y la diferencia (*C - Cx) entre* el contador de palabras de código global *C* y el contador *Cx del* descodificador de contenedores conectado 22 es mayor que el número máximo de entradas de memoria intermedia de palabras de código N, todos los contenedores que se almacenan en la actualidad en la memoria intermedia de contenedores particular 20 se descartan y se ignoran. Además de esa etapa adicional, la descodificación se opera tal como se ha descrito en lo que antecede. Si la memoria intermedia de contenedores 20 a la cual se envía una solicitud para un contenedor 19 está vacía (o bien debido a que ya se han retirado todos los contenedores o bien debido a que el mecanismo de bajo retardo descartó todos los contenedores en la primera etapa después de que se haya recibido la solicitud de contenedor), el descodificador de contenedores conectado 22 lee una o más nuevas palabras de código a partir de la memoria intermedia de bits 38 etc.

La medida para el retardo (o un límite superior del retardo) puede ser la suma de las longitudes de palabra de código máximas para las entradas de memoria intermedia activas en la memoria intermedia de palabras de código, en donde la longitud de palabra de código máxima para una entrada de memoria intermedia particular depende del contenedor descodificado que se asocia con esa entrada de memoria intermedia. Como ilustración, las longitudes de palabra de código máximas para las entradas de memoria intermedia están indicadas en los ejemplos en 6. Obsérvese de nuevo que la primera entrada de memoria intermedia es siempre una entrada de memoria intermedia reservada o una entrada de memoria intermedia libre, debido a que si la primera entrada de memoria intermedia contiene una palabra de código, esta palabra de código se escribe en el tren de bits. Sea el máximo retardo de memoria intermedia admitido (según sea determinado por la aplicación) de *D* bits. Este máximo retardo de memoria intermedia *D* ha de ser conocido tanto por el codificador como por el descodificador. El máximo retardo de memoria intermedia *D* puede estar fijado por la aplicación. El máximo retardo de memoria intermedia *D* se puede señalizar en el interior del tren de bits, por ejemplo, en el encabezamiento del paquete de datos (o el encabezamiento de rebanada) o en un conjunto de parámetros, el cual se incluye en el tren de bits. Este se puede señalizar en unidades de bits, o bytes, o un múltiplo de bits, o un múltiplo de bytes. Si un codificador de contenedores 10 envía una solicitud para la reserva de una o más nuevas entradas de memoria intermedia a la memoria intermedia de palabras de código 29, el siguiente proceso se ejecuta antes de que se reserve una nueva entrada de memoria intermedia de palabras de código (es decir, se ejecuta múltiples veces si múltiples entradas de memoria intermedia de palabras de código son reservadas por una solicitud).

Si la suma de las longitudes de palabra de código máximas para todas las entradas de memoria intermedia activas en la actualidad más la longitud de palabra de código máxima para la entrada de memoria intermedia que se reservará es mayor que el máximo retardo de memoria intermedia *D*, la primera entrada de memoria intermedia (la cual está reservada) se evacúa por el proceso que se ha descrito en lo que antecede hasta que la suma de las longitudes de palabra de código máximas para todas las entradas de memoria intermedia activas más la longitud de palabra de código máxima para la entrada de memoria intermedia que se reservará es menor que o igual al máximo retardo de memoria intermedia *D*. Como un ejemplo, considérese el ejemplo en la figura 8(b). La suma de las longitudes de palabra de código máximas para todas las entradas de memoria intermedia activas en la actualidad es 29. Supóngase que el máximo retardo de memoria intermedia *D* se establece igual a 32. Si la siguiente entrada de memoria intermedia es reservada por el codificador de contenedores 2 para el que la longitud de palabra de código máxima es igual a 3, la primera entrada de memoria intermedia no se evacúa, debido a que 29 + 3 no es mayor que 32. Pero si la siguiente entrada de memoria intermedia es reservada por el codificador de contenedores 1 para el que la longitud de palabra de código máxima es igual a 7, la primera entrada de memoria intermedia se evacúa, debido a que 29 + 7 es mayor que 32. La evacuación de la entrada de memoria intermedia reservada se realiza tal como se ha descrito en lo que antecede (mediante la adición de un contenedor con unos valores particulares o arbitrarios a la memoria intermedia de contenedores correspondiente).

En el lado de descodificador, se ejecuta un proceso similar para descartar los contenedores que se han añadido para obedecer la restricción de retardo. Por lo tanto, el descodificador mantiene un contador C que recuenta la longitud de palabra de código máxima para las palabras de código que se han leído a partir de la memoria intermedia de bits (este contador puede mantenerse en la memoria intermedia de bits). Obsérvese que las longitudes de palabra de código máximas que están asociadas con diferentes descodificadores de contenedores pueden ser diferentes. El contador C se inicializa (por ejemplo, con cero) en el comienzo de la descodificación de un paquete de datos y este se incrementa después de que se lea una palabra de código. Este contador no se incrementa en la longitud real de las palabras de código leídas, sino en su longitud máxima. Es decir, si una palabra de código se lee por un descodificador de contenedores particular y la longitud de palabra de código máxima que se asocia con la tabla de palabras de código que es usada por el descodificador de contenedores particular es *Lx* (un descodificador de contenedores diferente puede asociarse con una longitud de palabra de código máxima diferente), el contador *C* se incrementa en *Lx*. Además del contador global *C*, cada descodificador de contenedores 22 contiene un contador *Cx*, que almacena el valor del contador de palabras de código *C* antes de que la última palabra de código sea leída por el descodificador de contenedores 22 correspondiente. Es decir, cuando un descodificador de contenedores particular 22 lee una nueva palabra de código, su contador *Cx* se establece igual a *C* como una primera etapa y entonces la palabra de código se lee a partir de la memoria intermedia de bits. Cuando una solicitud

para un contenedor 19 se envía a una memoria intermedia de contenedores particular 20 y la diferencia (*C* - *Cx*) entre el contador global *C* y el contador *Cx del* descodificador de contenedores conectado 22 es mayor que el máximo retardo de memoria intermedia *D*, todos los contenedores que están almacenados en la actualidad en la memoria intermedia de contenedores particular 20 se descartan y se ignoran. Además de esa etapa adicional, la descodificación se opera tal como se ha descrito en lo que antecede. Si la memoria intermedia de contenedores 20 a la cual se envía una solicitud para un contenedor 19 está vacía (o bien debido a que ya se han retirado todos los contenedores o debido a que el mecanismo de bajo retardo descartó todos los contenedores en la primera etapa después de que se haya recibido la solicitud de contenedor), el descodificador de contenedores conectado 22 lee una o más nuevas palabras de código a partir de la memoria intermedia de bits 38 etc.

Por lo tanto, la pluralidad de descodificadores por entropía 22 y el selector 18 se pueden configurar para descartar de forma intermitente sufijos de secuencias de símbolos con el fin de no participar en la formación de la secuencia de símbolos que se va a reconstruir 29. El descarte de forma intermitente se puede realizar en unos eventos en los que un número de palabras de código que se han recuperado a partir de la entrada de palabra de código por la pluralidad de descodificadores por entropía entre dos recuperaciones de palabra de código consecutivas de un descodificador por entropía respectivo a partir de la entrada de palabra de código, satisface un criterio previamente determinado. La pluralidad de codificadores por entropía y la memoria intermedia de palabras de código, a su vez, puede configurarse para ampliar de forma intermitente unos símbolos reenviados en la actualidad pero aún no puestos en correspondencia a unas secuencias de símbolos válidas mediante unos símbolos de “no importa” que tienen los símbolos reenviados en la actualidad pero aún no puestos en correspondencia como prefijo, poner en correspondencia las secuencias de símbolos así ampliadas con palabras de código, introducir las palabras de código así obtenidas en las entradas de palabra de código reservadas y evacuar las entradas de palabra de código. La ampliación de forma intermitente, la entrada y la evacuación pueden tener lugar en unos eventos en los que un número de entradas de palabra de código reservadas más un número de entradas de palabra de código que tienen palabras de código introducidas en las mismas satisface un criterio previamente determinado. Los criterios previamente determinados pueden tener en cuenta las longitudes máximas de palabras de código de la pluralidad de pares de codificador / descodificador.

Para algunas arquitecturas, la forma que se ha descrito en lo que antecede de intercalación de palabras de código podría dar como resultado un inconveniente en términos de la complejidad de descodificación. Tal como se ilustra en la figura 9, todos los descodificadores de contenedores 22 leen palabras de código (en el caso general, palabras de código de longitud variable) a partir de una única memoria intermedia de bits 38. La lectura de las palabras de código no se puede realizar en paralelo, debido a que la palabra de código debe leerse en el orden correcto. Eso quiere decir que un descodificador de contenedores particular ha de esperar hasta que otros descodificadores de contenedores acaban la lectura de palabras de código. Y cuando la complejidad de la lectura de las palabras de código de longitud variable es significativa en relación con el resto del proceso de descodificación (parcialmente paralelizado), este acceso de las palabras de código de longitud variable puede ser un cuello de botella para la totalidad del proceso de descodificación. Hay algunas variaciones que pueden emplearse para reducir la complejidad del acceso a partir de la única memoria intermedia de bits, unas pocas de estas se describirán en lo sucesivo. Existe, por ejemplo, un único conjunto de palabras de código (que representa por ejemplo un código de prefijo libre de redundancia) y el conjunto de palabras de código que se usa para cada descodificador de contenedores 22 es un subconjunto del único conjunto de palabras de código. Obsérvese que diferentes descodificadores de contenedores 22 pueden usar diferentes subconjuntos del único conjunto de palabras de código. Incluso si los conjuntos de palabras de código que se usan por algunos de los descodificadores de contenedores 22 son los mismos, su asociación con secuencias de contenedores es diferente para diferentes descodificadores de contenedores 22. El mismo conjunto de palabras de código se puede usar para todos los descodificadores de contenedores 22. Si se tiene un único conjunto de palabras de código que incluye los conjuntos de palabras de código para todos los descodificadores de contenedores como subconjuntos, el análisis sintáctico de las palabras de código se puede realizar fuera de los descodificadores de contenedores, lo que puede reducir la complejidad del acceso a palabras de código. El proceso de codificación de PIPE no se modifica en relación con el proceso que se ha descrito en lo que antecede. El proceso de descodificación de PIPE modificado se ilustra en la figura 10. Un único lector de palabras de código se alimenta con los bits 46 a partir del tren de bits 40 y analiza sintácticamente las palabras de código - en general de longitud variable. Las palabras de código leídas 44 se insertan en una memoria intermedia de palabras de código 43, la cual representa una memoria intermedia de tipo primero en entrar primero en salir. Un descodificador de contenedores 22 envía una solicitud para una o más palabras de código 41 a la memoria intermedia de palabras de código 43 y como respuesta a esta solicitud, las una o más palabras de código se retiran de la memoria intermedia de palabras de código (en orden secuencial) y se envían al descodificador de contenedores 22 correspondiente. Obsérvese que en este caso, el potencialmente complejo análisis sintáctico de palabras de código se puede realizar en un proceso de segundo plano y no es necesario que este espere a los descodificadores de contenedores. Los descodificadores de contenedores acceden a unas palabras de código ya analizadas sintácticamente, el potencialmente complejo análisis sintáctico de palabras de código ha dejado de ser parte de una solicitud a la memoria intermedia global. En su lugar, unas palabras de código ya analizadas sintácticamente se envían a los descodificadores de contenedores, que también pueden implementarse de una forma tal que solo se envían índices de palabras de código a los descodificadores de

contenedores.

Intercalación de secuencias de bits de longitud fija

Una forma adicional de reducir la complejidad del descodificador de PIPE puede conseguirse cuando los descodificadores de contenedores 22 no leen palabras de código de longitud variable a partir de la memoria intermedia de bits global 38 sino que, en su lugar, siempre leen unas secuencias de longitud fija de bits a partir de la memoria intermedia de bits global 38 y añaden estas secuencias de longitud fija de bits a una memoria intermedia de bits local, en donde cada descodificador de contenedores 22 está conectado con una memoria intermedia de bits local separada. Las palabras de código de longitud variable se leen entonces a partir de la memoria intermedia de bits local. Por lo tanto, el análisis sintáctico de palabras de código de longitud variable se puede realizar en paralelo, solo el acceso de secuencias de longitud fija de bits se ha de realizar de una forma sincronizada, pero un acceso de este tipo de secuencias de longitud fija de bits es por lo general muy rápido, de tal modo que la complejidad de descodificación global puede reducirse para algunas arquitecturas. El número fijado de contenedores que se envían a una memoria intermedia de bits local particular puede ser diferente para diferentes memorias intermedias de bits locales y también puede variar con el tiempo, dependiendo de determinados parámetros como eventos en el descodificador de contenedores, la memoria intermedia de contenedores o la memoria intermedia de bits. No obstante, el número de bits que se leen por un proceso particular no depende de los bits reales que se leen durante el acceso particular, que es la diferencia importante para la lectura de palabras de código de longitud variable. La lectura de las secuencias de longitud fija de bits es desencadenada por determinados eventos en las memorias intermedias de contenedores, los descodificadores de contenedores o las memorias intermedias de bits locales. Como un ejemplo, es posible solicitar la lectura de una nueva secuencia de longitud fija de bits cuando el número de bits que se encuentran presentes en una memoria intermedia de bits conectada cae por debajo de un umbral previamente definido, en donde se pueden usar diferentes valores de umbral para diferentes memorias intermedias de bits. En el codificador, se ha de asegurar que las secuencias de longitud fija de contenedores se insertan en el mismo orden en el tren de bits, en el que se leen a partir del tren de bits en el lado de descodificador. También es posible combinar esta intercalación de secuencias de longitud fija con un control de retardo bajo similar a los que se han explicado en lo que antecede. En lo sucesivo, se describe una forma para la intercalación de secuencias de longitud fija de bits.

La figura 11 muestra una ilustración de una estructura de codificador de PIPE que intercala secuencias de longitud fija de bits para dos o más codificadores de contenedores. En contraste con la figura 7, los codificadores de contenedores 10 no están conectados con una única memoria intermedia de palabras de código. En su lugar, cada codificador de contenedores 10 está conectado con una memoria intermedia de bits 48 separada, que almacena bits para el tren de bits parcial correspondiente. Todas las memorias intermedias de bits 48 están conectadas con una memoria intermedia de bits global 51. La memoria intermedia de bits global 51 está conectada con una unidad de escritura de bits 53, la cual retira los bits 52 en el orden de codificación / descodificación de la memoria intermedia de bits global y escribe los bits retirados 54 en el tren de bits 55. Tras un determinado evento en una memoria intermedia de bits particular 48 o el codificador de contenedores conectado 10 o la memoria intermedia de contenedores 8, la memoria intermedia de bits 48 envía una solicitud 49 a la memoria intermedia de bits global 51 mediante la cual se reserva un determinado número de bits en la memoria intermedia de bits global 51. Las solicitudes para la reserva de las secuencias de bits de longitud fija 49 se procesan en orden secuencial. La memoria intermedia de bits global 51 representa una memoria intermedia de tipo primero en entrar primero en salir de una cierta forma; los bits que se reservan con anterioridad se escriben con anterioridad en el tren de bits. Se debería hacer notar que diferentes memorias intermedias de bits 48 pueden reservar una cantidad diferente de bits, que también puede variar con el tiempo sobre la base de unos símbolos ya codificados; pero el número de bits que son reservados por una solicitud particular se conoce en el momento en el que se envía la solicitud a la memoria intermedia de bits global.

En particular, las memorias intermedias de bits 48 y la memoria intermedia de bits global 51 se operan tal como se describe en lo sucesivo. La cantidad de bits que es reservada por una memoria intermedia de bits particular 48 se indica como *Nx*. Este número de bits *Nx* puede ser diferente para diferentes memorias intermedias de bits 48 y también puede variar con el tiempo. El número de bits *Nx* que son reservados por una memoria intermedia de bits particular 48 puede ser fijo con el tiempo. Las reservas para un número fijado *Nx* de bits 49 se desencadenan sobre la base del número de bits *Mx* en las memorias intermedias de bits 48, el número de bits *Nx* para las solicitudes de reserva, y la longitud de palabra de código máxima *Lx* asociada. Obsérvese que cada codificador de contenedores 10 puede asociarse con una longitud de palabra de código máxima *Lx* diferente. Si un contenedor 7 se envía a una memoria intermedia de contenedores particular 8, y la memoria intermedia de contenedores particular 8 está vacía, y se reserva no más de una secuencia de *Nx* bits en la memoria intermedia de bits global para la memoria intermedia de bits 48 que está conectada con la memoria intermedia de contenedores particular (por medio de un codificador de contenedores), y la diferencia *Nx* - *Mx* entre el número *Nx* de bits que son reservados por una solicitud de reserva de la memoria intermedia de bits 48 que está conectada (por medio de un codificador de contenedores) con la memoria intermedia de contenedores particular 8 y el número de bits *Mx* que se encuentran presentes en la actualidad en esta memoria intermedia de bits 48 es menor que la longitud de palabra de código máxima *Lx* que se asocia con el

codificador de contenedores 10 correspondiente, la memoria intermedia de bits conectada 49 envía una solicitud 49 para la reserva de *Nx* bits a la memoria intermedia de bits global 51. La memoria intermedia de bits global 51 reserva *Nx* bits para la memoria intermedia de bits particular 48 e incrementa su puntero para la siguiente reserva. Después de que se hayan reservado los *Nx* bits en la memoria intermedia de bits global, el contenedor 7 se almacena en la memoria intermedia de contenedores 8. Si este único contenedor ya representa, de hecho, una secuencia de contenedores que se asocia con una palabra de código, el codificador de contenedores 10 retira este contenedor de la memoria intermedia de contenedores 8 y escribe la palabra de código 47 correspondiente en la memoria intermedia de bits conectada 48. De lo contrario (este único contenedor ya representa, de hecho, una secuencia de contenedores que se asocia con una palabra de código), unos contenedores adicionales 7 se aceptan por la memoria intermedia de contenedores particular 8 hasta que la memoria intermedia de contenedores 8 contiene una secuencia de contenedores que se asocia con una palabra de código. En este caso, el codificador de contenedores conectado 10 retira la secuencia de contenedores 9 de la memoria intermedia de contenedores 8 y escribe la palabra de código 47 correspondiente en la memoria intermedia de bits conectada 48. Si el número resultante de bits *Mx* en la memoria intermedia de bits 48 es mayor que o igual al número de bits reservados *Nx*, los *Nx* bits que se escribieron en primer lugar en la memoria intermedia de bits 48 se insertan en el espacio previamente reservado en la memoria intermedia de bits global 51. Para el siguiente contenedor 7 que se envía a la memoria intermedia de contenedores particular 8, se ejecuta el mismo proceso que se ha especificado en lo que antecede; es decir, se comprueba en primer lugar si se ha de reservar un nuevo número de *Nx* bits en la memoria intermedia de bits global *(si Nx - Mx* es menor que *Lx*) y entonces el contenedor se inserta en la memoria intermedia de contenedores 8, etc. La unidad de escritura de bits escribe las secuencias de bits de longitud fija de la memoria intermedia de bits global en el orden en el cual se han reservado estas. Si la primera entrada de longitud fija en la memoria intermedia de bits global 51 contiene una secuencia de bits de longitud fija que se ha insertado en la práctica en la memoria intermedia de bits global (es decir, no solo se reserva), la unidad de escritura de bits 53 retira los bits para esta secuencia de bits 52 de la memoria intermedia de bits global 51 y escribe los bits 54 en el tren de bits. Este proceso se repite hasta que la primera entrada de longitud fija en la memoria intermedia de bits global representa una entrada reservada o una libre. Si la primera entrada de longitud fija en la memoria intermedia de bits global representa una entrada reservada, la unidad de escritura de bits 53 espera hasta que esta entrada se ha llenado con bits reales antes de que esta escriba unos bits adicionales 54 en el tren de bits 55.

Al final de un paquete de datos, las memorias intermedias de contenedores se evacúan tal como se ha descrito en lo que antecede. Además, las memorias intermedias de bits se han de evacuar mediante la adición de bits con un valor particular o uno arbitrario hasta que todas las entradas de memoria intermedia reservadas en la memoria intermedia de bits global se han llenado y escrito en el tren de bits.

En la figura 12 se ilustran dos ejemplos para el posible estatus de la memoria intermedia de bits global 51. En el ejemplo (a), se ilustra un caso en el que diferentes memorias intermedias de bits / codificadores de contenedores reservan un número diferente de bits. La memoria intermedia de bits global contiene 3 entradas con unas secuencias de bits de longitud fija escritas en la práctica y 4 entradas con unas secuencias de bits de longitud fija reservadas. La primera entrada de longitud fija ya contiene bits reales (los cuales han de estar recién insertados por la memoria intermedia de bits / codificador de contenedores 2); esta entrada (es decir, los 8 bits correspondientes) puede retirarse y escribirse en el tren de bits. La siguiente entrada reserva 10 bits para el codificador de contenedores 3, pero aún no se han insertado bits reales. Esta entrada no se puede escribir en el tren de bits; ha de esperarse hasta que se han insertado los bits reales. En el segundo ejemplo (b), todas las memorias intermedias de bits / codificadores de contenedores reservaron el mismo número de bits (8 bits). La memoria intermedia de bits global contiene 4 reservas para secuencias de 8 bits y 3 secuencias de 8 bits escritas en la práctica. La primera entrada contiene una reserva para 8 bits para el codificador de contenedores 3. Antes de que se pueda escribir algún bit nuevo en el tren de bits, la unidad de escritura de bits ha de esperar hasta que la memoria intermedia de bits / codificador de contenedores 3 escriba los valores reales de los 8 bits en esta entrada reservada.

La figura 13 muestra una ilustración de una estructura de descodificador de PIPE que intercala secuencias de longitud fija de bits. En contraste con la figura 9, los descodificadores de contenedores 22 no están conectados con una única memoria intermedia de bits. En su lugar, cada descodificador de contenedores 22 está conectado con una memoria intermedia de bits 58 separada, que almacena bits a partir del tren de bits parcial correspondiente. Todas las memorias intermedias de bits 58 están conectadas con una memoria intermedia de bits global 61. Los bits 62 a partir del tren de bits 63 se insertan en la memoria intermedia de bits global 61. Tras un determinado evento en una memoria intermedia de bits particular 58 o el descodificador de contenedores conectado 22 o la memoria intermedia de contenedores 20, la memoria intermedia de bits 58 envía una solicitud 59 a la memoria intermedia de bits global 61 mediante lo cual una secuencia de longitud fija de bits 60 se retira de la memoria intermedia de bits global 61 y se inserta en la memoria intermedia de bits particular 58. Las solicitudes para las secuencias de bits de longitud fija 59 se procesan en orden secuencial. La memoria intermedia de bits global 61 representa una memoria intermedia de tipo primero en entrar primero en salir; los bits que se insertan con anterioridad en la memoria intermedia de bits global se retiran con anterioridad. Se debería hacer notar que diferentes memorias intermedias de bits 58 pueden solicitar una cantidad diferente de bits, que también puede variar con el tiempo sobre la base de unos símbolos ya descodificados; pero el número de bits que son solicitados por una solicitud particular se conoce en el momento en

el que se envía la solicitud a la memoria intermedia de bits global. Se debería hacer notar que la memoria intermedia de bits global 61 no se requiere necesariamente, debido a que las palabras de código también se podrían leer directamente a partir del tren de bits. La memoria intermedia de bits global 61 se incluye principalmente en la ilustración para separar claramente diferentes aspectos de la cadena de procesamiento.

Las memorias intermedias de bits 58 y la memoria intermedia de bits global 61 se pueden operar tal como se describe en lo sucesivo. La cantidad de bits que son solicitados y leídos por una memoria intermedia de bits particular 58 se indica como *Nx*, esta es igual a la cantidad de bits que se escribe en la memoria intermedia de bits global por la memoria intermedia de bits correspondiente en el lado de codificador. Este número de bits *Nx* puede ser diferente para diferentes memorias intermedias de bits 58 y también puede variar con el tiempo. Preferiblemente, el número de bits *Nx* que son solicitados y leídos por una memoria intermedia de bits particular 58 puede ser fijo con el tiempo. La lectura de un número fijado *Nx* de bits 60 se desencadena sobre la base del número de bits *Mx* en la memoria intermedia de bits 58 y la longitud de palabra de código máxima *Lx* asociada. Obsérvese que cada descodificador de contenedores 22 puede asociarse con una longitud de palabra de código máxima *Lx* diferente. Si una solicitud para un contenedor 19 se envía a una memoria intermedia de contenedores particular 20, y la memoria intermedia de contenedores particular 20 está vacía, y el número *Mx* de bits en la memoria intermedia de bits 58 que está conectada (por medio de un descodificador de contenedores) con la memoria intermedia de contenedores particular 20 es menor que la longitud de palabra de código máxima Lx que se asocia con el descodificador de contenedores 22 correspondiente, la memoria intermedia de bits conectada 58 envía una solicitud 59 para una nueva secuencia *de Nx* bits a la memoria intermedia de bits global 61. Como respuesta a esta solicitud, los primeros *Nx* bits se retiran de la memoria intermedia de bits global 61 y esta secuencia de *Nx* bits 60 se envía a la memoria intermedia de bits 58 a partir de la cual se envió la solicitud. Por último, esta secuencia de *Nx* bits se añade a la memoria intermedia de bits 58 correspondiente. Entonces la siguiente palabra de código 57 se lee a partir de esta memoria intermedia de bits, y el descodificador de contenedores conectado 22 inserta la secuencia de contenedores asociada 21 en la memoria intermedia de contenedores conectada 20. Como respuesta final a la solicitud original para un contenedor 19, el primer contenedor se retira de la memoria intermedia de contenedores 20 y este contenedor descodificado 25 se envía al selector de memoria intermedia de contenedores 18. Cuando la siguiente solicitud de contenedor 19 se envía a la memoria intermedia de contenedores particular 20 y la memoria intermedia de contenedores no está vacía, el siguiente bit se retira de la memoria intermedia de contenedores 20. Si la memoria intermedia de contenedores está vacía pero el número *Mx* de bits en la memoria intermedia de bits conectada 58 es mayor que o igual a la longitud de palabra de código máxima *Lx* asociada, la siguiente palabra de código se lee a partir de la memoria intermedia de bits y una nueva secuencia de contenedores se inserta en la memoria intermedia de contenedores, a partir de la cual se retira el primer bit y se envía al selector de memoria intermedia de contenedores. Si la memoria intermedia de contenedores está vacía y el número Mx de bits en la memoria intermedia de bits conectada 58 es menor que la longitud de palabra de código máxima *Lx* asociada, la siguiente secuencia de *Nx* bits se lee a partir de la memoria intermedia de bits global 61 y se inserta en la memoria intermedia de bits local conectada 58, la siguiente palabra de código se lee a partir de la memoria intermedia de bits, una nueva secuencia de contenedores se inserta en la memoria intermedia de contenedores, y el primer contenedor de la secuencia es removed y se envía al selector de memoria intermedia de contenedores. Este proceso se repite hasta que se han descodificado todos los símbolos de origen.

Al final de un paquete de datos, más contenedores y / o bits de los que se requieren para descodificar los símbolos de origen solicitados se podrían insertar en la memoria intermedia de contenedores y / o la memoria intermedia de bits. Los contenedores restantes en la memoria intermedia de contenedores y los bits restantes en la memoria intermedia de bits se descartan y se ignoran.

Intercalación de secuencias de bits de longitud fija con una restricción de retardo baja

El codificador y descodificador de PIPE con una intercalación de secuencias de bits de longitud fija también se pueden combinar con el esquema para controlar el retardo de memoria intermedia de codificador, el cual se ha descrito en lo que antecede. El concepto de codificación de PIPE es el mismo que en el caso de comportar el control de retardo que se ha descrito en lo que antecede. Si una medida en relación con el retardo o un límite superior del retardo (véase en lo sucesivo) supera un umbral especificado, la primera entrada de memoria intermedia reservada se llena mediante la evacuación de la memoria intermedia de contenedores correspondiente (usando un mecanismo similar al del final de un paquete de datos) y potencialmente mediante la escritura de bits adicionales para llenar todos los bits de la entrada de memoria intermedia de longitud fija reservada. Mediante un mecanismo de este tipo, el número de entradas de memoria intermedia en espera se reduce hasta que la medida de retardo asociada es menor que el umbral especificado. En el lado de descodificador, se han de descartar los contenedores y los bits que se han insertado en el lado de codificador con el fin de obedecer la restricción de retardo. Para este descarte de contenedores y bits se puede usar básicamente el mismo mecanismo que en el lado de codificador.

La medida para el retardo (o un límite superior del retardo) puede ser el número de bits en las entradas de memoria intermedia activas en la memoria intermedia de bits global, en donde el número de entradas de memoria intermedia activas es el número de entradas de memoria intermedia de longitud fija reservadas más el número de entradas de

memoria intermedia de longitud fija que contienen unos bits ya escritos. Obsérvese que la primera entrada de memoria intermedia es siempre una entrada de memoria intermedia de longitud fija reservada o una entrada de memoria intermedia libre, debido a que si la primera entrada de memoria intermedia contiene bits escritos, estos bits se escriben en el tren de bits. Sea el máximo retardo de memoria intermedia admitido (según sea determinado por la aplicación) de *D* bits. Este máximo retardo de memoria intermedia *D* ha de ser conocido tanto por el codificador como por el descodificador. El máximo retardo de memoria intermedia *D* puede estar fijado por la aplicación. El máximo retardo de memoria intermedia *D* se puede señalizar en el interior del tren de bits, por ejemplo, en el encabezamiento del paquete de datos (o el encabezamiento de rebanada) o en un conjunto de parámetros, el cual se incluye en el tren de bits. Este se puede señalizar en unidades de bits, o bytes, o un múltiplo de bits, o un múltiplo de bytes. Si un codificador de contenedores 10 envía una solicitud para la reserva de una nueva secuencia de bits de longitud fija a la memoria intermedia de bits global 51, el siguiente proceso se ejecuta antes de que se reserve una nueva entrada de memoria intermedia de longitud fija.

Si el número de bits en las entradas de memoria intermedia activas en la memoria intermedia de bits global más el número de bits que serán reservados por la solicitud de reserva actual es mayor que el máximo retardo de memoria intermedia *D*, la primera entrada de memoria intermedia (la cual está reservada) se evacúa por el proceso que se describe en lo sucesivo hasta que el número de bits en las entradas de memoria intermedia activas en la memoria intermedia de bits global más el número de bits que serán reservados por la solicitud de reserva actual sea menor que o igual al máximo retardo de memoria intermedia *D*. La evacuación de una entrada de memoria intermedia de longitud fija reservada es similar a la evacuación al final de un paquete de datos: El codificador de contenedores 10 que está conectado con la memoria intermedia de bits 48 que ha reservado la primera entrada de memoria intermedia correspondiente se evacúa mediante la adición de contenedores con unos valores particulares o arbitrarios a la memoria intermedia de contenedores conectada 8 hasta que la secuencia de contenedores resultante representa una secuencia de contenedores que se asocia con una palabra de código, entonces la palabra de código se inserta en la memoria intermedia de bits 48 correspondiente. Tal como se ha mencionado en lo que antecede, una forma preferida para añadir contenedores a la memoria intermedia de contenedores es añadir aquellos contenedores que producen la palabra de código posible más corta. Si, después de la escritura de la palabra de código en la memoria intermedia de bits conectada y una inserción potencial de una secuencia de bits de longitud fija en la memoria intermedia de bits global, sigue habiendo bits en la memoria intermedia de bits (es decir, la palabra de código escrita no llenó por completo la secuencia de longitud fija reservada de bits), unos bits adicionales con unos valores particulares o arbitrarios se añaden a la memoria intermedia de bits hasta que todos los bits se han retirado de la memoria intermedia de bits y se han escrito en la entrada de memoria intermedia reservada. Por último, al final de este proceso, la entrada de memoria intermedia completada (la primera entrada de longitud fija en la memoria intermedia de bits global) se retira de la memoria intermedia de bits global y se escribe en el tren de bits.

En el lado de descodificador, se ejecuta un proceso similar para descartar los contenedores y los bits que se han añadido para obedecer la restricción de retardo. Por lo tanto, el descodificador mantiene un contador *C* que recuenta los bits que se han leído a partir de la memoria intermedia de bits global (este contador puede mantenerse en la memoria intermedia de bits global). El contador *C* se inicializa (por ejemplo, con cero) en el comienzo de la descodificación de un paquete de datos y este se incrementa después de que se lea una secuencia de longitud fija. Si una secuencia de longitud fija de *Nx* bits se lee a partir de la memoria intermedia de bits global 61, el contador *C* se incrementa en *Nx*. Además del contador global *C*, cada memoria intermedia de bits 58 contiene un contador *Cx*, que almacena el valor del contador de bits *C* antes de que la última secuencia de bits de longitud fija se inserte en la memoria intermedia de bits 58 correspondiente. Cuando una memoria intermedia de bits particular 58 lee una nueva secuencia de bits de longitud fija, su contador *Cx* se establece igual a *C* como una primera etapa y entonces la secuencia de bits de longitud fija se lee a partir de la memoria intermedia de bits global 61. Cuando una solicitud para un contenedor 19 se envía a una memoria intermedia de contenedores particular 20 y la diferencia (*C* - *Cx*) entre el contador global *C* y el contador *Cx* de la memoria intermedia de bits conectada 58 es mayor que el máximo retardo de memoria intermedia *D*, todos los contenedores que se almacenan en la actualidad en la memoria intermedia de contenedores particular 20 y todos los bits que se almacenan en la memoria intermedia de bits conectada 58 se descartan y se ignoran. Además de esa etapa adicional, la descodificación se opera tal como se ha descrito en lo que antecede. Si la memoria intermedia de contenedores 20 a la cual se envía una solicitud para un contenedor 19 está vacía (o bien debido a que ya se han retirado todos los contenedores o bien debido a que el mecanismo de bajo retardo descartó todos los contenedores en la primera etapa después de que se haya recibido la solicitud de contenedor), el descodificador de contenedores conectado 22 intenta leer una nueva palabra de código a partir de la memoria intermedia de bits conectada 58. Si el número de bits en la memoria intermedia de bits 58 es menor que la longitud de palabra de código máxima, una nueva secuencia de bits de longitud fija se lee a partir de la memoria intermedia de bits global 61, antes de que se lea la palabra de código, etc.

Las figuras 7 a 13 se refieren a posibilidades para conseguir una trayectoria de trenes de bits intercalados entre el codificador de PIPE 104 por un lado y el descodificador de PIPE 202 por otro lado. Tal como se ha descrito en lo que antecede con respecto a las figuras 1 y 2, un aparato de codificación y descodificación por entropía puede conectarse entre sí por dos canales separados, uno de los cuales transporta el tren de bits de VLC 112 y el otro de los cuales transporta el tren de bits codificado de PIPE intercalado. No obstante, hay también posibilidades de

intercalar incluso tanto el tren de bits de VLC 112 así como los trenes de bits codificados de PIPE 118, y tales posibilidades se describirán más adelante con respecto a las figuras 20 a 24. No obstante, antes de eso, se proporciona trasfondo matemático con respecto al esquema de codificación de PIPE así como detalles en lo que respecta a cómo subdividir de forma óptima el intervalo de probabilidad con la asignación de los intervalos parciales individuales resultantes a los codificadores por entropía 116 y los descodificadores por entropía 210 individuales, de forma respectiva.

Tal como ya se ha hecho notar, en la codificación de PIPE el espacio de eventos de la secuencia de entrada de símbolos discretos se pone en correspondencia con un pequeño conjunto de intervalos de probabilidad binaria. Los modelos de probabilidad para los símbolos de origen pueden ser fijos o adaptativos al tiempo la realización de una codificación por entropía usando los intervalos de probabilidad permanece fijada y está desacoplada de la fase de modelado. Cada uno de los intervalos de probabilidad se puede codificar usando un código por entropía muy simple que tiene el nivel de complejidad de los códigos de Huffman. El exceso de tasa del código de entropía de subdivisión en particiones de intervalos de probabilidad (PIPE, *probability interval partitioning entropy*) es similar a la de la codificación aritmética pura.

La codificación por entropía, en general, se puede considerar como la forma más genérica de compresión de datos sin pérdida. La compresión sin pérdida tiene por objeto representar unos datos discretos con menos bits de los necesarios para la representación de datos original pero sin pérdida alguna de información. Los datos discretos se pueden dar en forma de texto, gráficos, imágenes, vídeo, audio, habla, fax, datos médicos, datos meteorológicos, datos financieros, o cualquier otra forma de datos digitales. En muchas aplicaciones de codificación, los datos de origen originales se ponen en correspondencia en primer lugar con unos así denominados símbolos de codificación y estos símbolos de codificación a continuación se codifican por entropía. La puesta en correspondencia con símbolos de codificación puede incluir cuantificación, caso en el cual, el esquema de codificación global tiene pérdidas. Un símbolo de codificación s puede adoptar cualquier valor de un alfabeto *M-ario* ($M \geq 2$) $A = \{a_0, \ldots, a_{M-1}\}$. Para el fin de codificar el símbolo s, el alfabeto se asocia con una función de masa de probabilidad (pmf, *probability mass function*) estimada $\{p_s(a_0), \ldots, p_s(a_{M-1})\}$ y todas las dependencias entre símbolos de codificación que no se consideran en esta pmf se desprecian. Para estos escenarios abstractos, la entropía

$$H_s = -\sum_{i=0}^{M-1} p_s(a_i)\log_2 p_s(a_i) \qquad \text{(B1)}$$

es el más grande límite inferior para la longitud de palabra de código esperada en bits por símbolos, para codificar los símbolos s, que pueden conseguirse con técnicas de codificación por entropía. Durante décadas, la codificación de Huffman y la codificación aritmética han dominado la codificación por entropía práctica. Estos son ejemplos bien conocidos de códigos prácticos capaces de aproximarse al límite de entropía (en un cierto sentido).

Para una distribución de probabilidad fija, los códigos de Huffman son relativamente sencillos de construir. La propiedad más atractiva de los códigos de Huffman es que su implementación puede realizarse de forma eficiente mediante el uso de tablas de códigos de longitud variable (VLC, *variable length code*). No obstante, cuando se abordan unas estadísticas de origen variables en el tiempo, es decir, unas probabilidades de símbolo cambiantes, la adaptación del código de Huffman y sus tablas de VLC correspondientes es bastante exigente, tanto en términos de la complejidad algorítmica como en términos de los costes de implementación. Asimismo, en el caso de tener un valor de alfabeto dominante con $p_s(a_i) > 0{,}5$, la redundancia del código de Huffman correspondiente (sin usar ampliación de alfabeto alguna tal como la codificación de longitud de series) puede ser bastante sustancial. Otra deficiencia de los códigos de Huffman viene dada por el hecho de que en el caso de tratar con un modelado de probabilidad de orden superior, se pueden requerir múltiples conjuntos de tablas de VLC.

La codificación aritmética, por otro lado, a pesar de ser sustancialmente más compleja que VLC, ofrece la ventaja de una manipulación más consistente y adecuada cuando se hace frente a un modelado de probabilidad adaptativo y de orden superior así como con el caso de unas distribuciones de probabilidad muy sesgadas. En la práctica, esta característica básicamente resulta del hecho de que la codificación aritmética proporciona un mecanismo, por lo menos conceptualmente, para poner en correspondencia cualquier valor dado de estimada de probabilidad de una forma más o menos directa con una porción de la palabra de código resultante. Estando provista con una interconexión de este tipo, la codificación aritmética permite una separación limpia entre las tareas de modelado de probabilidad y de estimación de probabilidad, por un lado, y la codificación por entropía real, es decir, la puesta en correspondencia de símbolos con palabras de código, por otro lado.

Diferente de los esquemas de codificación por entropía convencionales recién analizados, la codificación de PIPE usa subdivisión en particiones de intervalos de probabilidad, el trasfondo matemático de lo cual se describe con más detalle en lo sucesivo.

Considérese la secuencia de símbolos de codificación $\{s_0, \ldots, s_{N-1}\}$. Cada símbolo se extrae de un alfabeto $s_i \in A_i$.

Los alfabetos $A_i = \{a_0^i, a_1^i, \ldots\}$ contienen dos o más letras estando cada una asociada con una estimada de probabilidad $p_s(a_m^i)$. Las estimadas de probabilidad $p_s(a_m^i)$ son conocidas por el codificador y el descodificador y pueden ser fijas o variables. Se supone que se estiman probabilidades variables de forma simultánea en el codificador y en el descodificador. O bien los alfabetos A_i pueden ser idénticos para la secuencia de símbolos o bien diferentes tipos de símbolo están asociados con diferentes alfabetos. En este último caso, se supone que el descodificador conoce el alfabeto de cada símbolo en la secuencia. Esta suposición se justifica debido a que las descripciones de códec de origen prácticas contienen una sintaxis que estipula el orden de los símbolos y sus alfabetos.

La secuencia de símbolos {s_0, ... , s_{N-1}} se convierte a una secuencia de símbolos binarios, a los que también se hace referencia como contenedores. Para cada símbolo s_i, la binarización

$$\boldsymbol{b}^i = \{b_0^i \ldots\} = \gamma_b^i(s_i) \qquad \text{(B2)}$$

representa una puesta en correspondencia biyectiva de las letras del alfabeto a_m^i con unos conjuntos ordenados de contenedores $\boldsymbol{b}_m^i$. La puesta en correspondencia de binarización γ_b^i puede ser diferente para diferente símbolos s_i o categorías de símbolos. Cada secuencia de contenedores i para un símbolo particular s_i consiste en uno o más contenedores b_k^i. En el lado de descodificador, un símbolo s_i se puede reconstruir por la puesta en correspondencia inversa $s_i = (\gamma_b^i)^{-1}(\mathbf{b}^i)$ dada la secuencia de contenedores $\boldsymbol{b}^i$. Como resultado de la binarización, se obtiene una secuencia de contenedores {b_0, ... , b_{B-1}} que representa la secuencia de símbolos de origen {s_0, ... , s_{N-1}}.

Todos los contenedores b_j están asociados con el mismo alfabeto binario B = {0, 1}, pero las pmf binarias correspondientes $\{p_0^j, p_1^j\}$, con $p_1^j = p_0^j$, son por lo general diferentes. Una pmf binaria $\{p_0^j, p_1^j\}$ se puede describir mediante el valor del contenedor menos probable (LPB, *less probable bin*) $b^j{}_{LPB}$ y su probabilidad $p^j{}_{LPB}$ (con $p^j{}_{LPB} \leq 0{,}5$). Esta descripción de probabilidad binaria {$b^j{}_{LPB}$, $p^j{}_{LPB}$} se puede obtener directamente a partir de las estimadas de probabilidad $p_s(a_m^i)$ para los alfabetos de símbolos dadas las puestas en correspondencia de binarización γ_b^i. También es posible (y a menudo preferible) estimar directamente {$b^j{}_{LPB}$, $p^j{}_{LPB}$} de forma simultánea en el codificador y en el lado de descodificador. Por lo tanto, los contenedores pueden asociarse con un modelo de probabilidad (al que también se hace referencia como contexto) sobre la base de la sintaxis y los contenedores o símbolos previamente codificados. Y para cada modelo de probabilidad, la descripción de probabilidad {$b^j{}_{LPB}$, $p^j{}_{LPB}$} puede estimarse sobre la base de los valores de los contenedores que se codifican con el modelo de probabilidad. Un ejemplo para un modelado de probabilidad binaria de este tipo se describe con respecto a CABAC de la norma H.264.

Debido a que la función de entropía binaria

$$H(p) = -p\log_2(p) - (1-p)\log_2(1-p) \qquad \text{(B3)}$$

es simétrica en torno a p = 0,5, el mismo codificador binario se puede usar para codificar todos los contenedores que están asociados con la misma probabilidad de LPB $p^j{}_{LPB}$, independiente del valor de $b^j{}_{LPB}$. Por lo tanto, la secuencia de contenedores {b_0, ... , b_{B-1}} se convierte a una secuencia de contenedores de codificación $\{b_0^c, \ldots, b_{B-1}^c\}$. Para cada contenedor b_j, la puesta en correspondencia biyectiva correspondiente γ_c^j se especifica por

$$b_j^c = \gamma_c^j(b_j) = b_j \oplus b^j{}_{LPB} \qquad \text{(B4)}$$

en la que $\oplus$ indica el operador o exclusivo. En el lado de descodificador, los contenedores b_j se pueden reconstruir dados los contenedores de codificación b_j^c y el valor de LPB correspondiente $b^j{}_{LPB}$ por la puesta en correspondencia inversa $b_j = (\gamma_c^j)^{-1}(b_j^c) = b_j^c \oplus b^j{}_{LPB}$. Un contenedor de codificación $b_j^c = 0$ especifica que el valor del contenedor correspondiente b_j es igual al valor de LPB $b^j{}_{LPB}$, y un contenedor de codificación $b_j^c = 1$ especifica que el valor del contenedor correspondiente b_j es igual al valor del contenedor más probable (MPB, *more*

probable bin) 1 - b^j_{LPB}.

La secuencia de contenedores de codificación $\{b_0^c,\ldots,b_{B-1}^c\}$ representa, de hecho, de forma única la secuencia de símbolos de origen {s_0, ... , s_{N-1}} y las estimadas de probabilidad correspondientes, que puede emplearse para la codificación por entropía, se describen completamente mediante las probabilidades de LPB p^j_{LPB} (con $p^j_{LPB} \leq 0,5$). Por lo tanto, solo es necesario considerar probabilidades en el intervalo semiabierto (0, 0,5] para diseñar el codificador por entropía binario para los contenedores de codificación b_i^c .

Para la codificación por entropía binaria real, la secuencia de contenedores de codificación $\{b_0^c,\ldots,b_{B-1}^c\}$ se proyecta sobre un pequeño número de intervalos de probabilidad I_k. El intervalo de probabilidad de LPB (0, 0,5] se subdivide en particiones para dar K intervalos $I_k = (p_k, p_{k+1}]$

$$\bigcup_{k=0}^{K-1} I_k = (0,0,5] \qquad y \qquad I_k \bigcap I_j = \varnothing \ \ para \ \ k \neq j \quad \text{(B5)}$$

El conjunto de K intervalos se caracteriza por K-1 fronteras de intervalos p_k con k = 1, ... , K-1. Sin pérdida de generalidad, se supone $p_k < p_{k+1}$ para k = 0, ... , K. Las fronteras de intervalo exteriores son fijas y vienen dadas por $p_0 = 0$ y $p_K = 0,5$. Se diseña un codificador por entropía binario no adaptativo simple para cada intervalo I_k. Todos los contenedores de codificación b_j^c con las probabilidades de LPB asociadas $p^j_{LPB} \in I_k$ están asignados al intervalo I_k y se codifican con el codificador por entropía fijo correspondiente.

En la siguiente descripción, todos los contenedores representan unos contenedores de codificación b_j^c y todas las probabilidades p son unas probabilidades de LPB p^j_{LPB}.

Para investigar el impacto de la discretización de intervalo de probabilidad sobre la eficiencia de codificación, se supone que se puede diseñar un codificador por entropía óptimo para una probabilidad fija que alcanza el límite de entropía. Cada intervalo de probabilidad $I_k = (p_k, p_{k+1}]$ se asocia con una probabilidad representativa $p_{I_k} \in I_k$ y el codificador por entropía óptimo correspondiente alcanzará el límite de entropía para esta probabilidad representativa. Bajo esta suposición, la tasa para codificar un contenedor con una probabilidad p usando el codificador por entropía óptimo para la representante de intervalo pI_k viene dada por

$$\begin{aligned} R(p,p_{I_k}) &= -p\log_2(p_{I_k}) - (1-p)\log_2(1-p_{I_k}) \\ &= H(p_{I_k}) + (p-p_{I_k})\,H'(p_{I_k}) \end{aligned} \qquad \text{(B6)}$$

en la que $H(p)$ representa la función de entropía binaria 3 y

$$H'(p) = \log_2\left(\frac{1-p}{p}\right) \qquad \text{(B7)}$$

es su primera derivada. Se supone adicionalmente que la distribución de las probabilidades en el intervalo (0, 0,5] viene dada por $f(p)$, con $\int_0^{0,5} f(p)\,dp = 1$. Entonces, la tasa esperada, en bits por contenedor, para un conjunto dado de K intervalos $\{I_k\}$ con unas probabilidades representativas correspondientes $\{p_{I_k}\}$ puede escribirse como

$$R = R(\{I_k\},\{p_{I_k}\}) = \sum_{k=0}^{K-1}\left(\int_{p_k}^{p_{k+1}} R(p,p_{I_k})\,f(p)\,dp\right) \qquad \text{(B8)}$$

La primera derivada parcial con respecto a cualquier probabilidad representativa p_{I_k}, con k = 0, ... , K-1, viene dada por

$$\frac{\partial}{\partial p_{I_k}} R = \frac{p_{I_k} \int_{p_k}^{p_{k+1}} f(p)\, dp - \int_{p_k}^{p_{k+1}} p\, f(p)\, dp}{p_{I_k}(1-p_{I_k}) \ln 2} \qquad \text{(B9)}$$

La ecuación $\frac{\partial}{\partial p_{I_k}} R = 0$ tiene una única solución

$$p_{I_k}^* = \frac{\int_{p_k}^{p_{k+1}} p\, f(p)\, dp}{\int_{p_k}^{p_{k+1}} f(p)\, dp} \qquad \text{(B10)}$$

para la probabilidad representativa p_{I_k} en el interior del dominio de definición I_k. La segunda derivada parcial para esta solución

$$\frac{\partial^2}{\partial p_{I_k}^2} R(p_{I_k}^*) = \frac{\int_{p_k}^{p_{k+1}} f(p)\, dp}{p_{I_k}^*(1-p_{I_k}^*) \ln 2} \qquad \text{(B11)}$$

es siempre mayor que cero si

$$\int_{p_k}^{p_{k+1}} f(p)\, dp > 0 \qquad \text{(B12)}$$

Por lo tanto, si se satisface la condición B12, el valor dado $p_{I_k}^*$ en la eq. B10 es la probabilidad representativa para un intervalo I_k que reduce al mínimo la tasa global esperada R dados los límites de intervalo p_k y p_{k+1}. De lo contrario, no se proyecta contenedor alguno en el intervalo I_k y la probabilidad representativa $p_{I_k} \in I_k$ se puede elegir de forma arbitraria sin impacto alguno sobre la tasa global R; pero una configuración de este tipo debería evitarse, debido a que el intervalo I_k no se emplearía para la codificación por entropía.

Para hallar una condición para unas fronteras de intervalo óptimas, se investigan las primeras derivadas de la tasa global esperada R con respecto a las fronteras de intervalo p_k con $k = 1, \ldots, K-1$. Si $f(p) > 0$ para todo $p \in [p_{I_{k-1}}, p_{I_k})$, la ecuación $\frac{\partial}{\partial p_k} R = 0$ tiene una única solución

$$p_k^* = \frac{H(p_{I_k}) - p_{I_k} H'(p_{I_k}) - H(p_{I_{k-1}}) + p_{I_{k-1}} H'(p_{I_{k-1}})}{H'(p_{I_{k-1}}) - H'(p_{I_k})} \qquad \text{(B13)}$$

para la frontera de intervalo p_k en el interior del dominio de definición $[p_{I_{k-1}}, p_{I_k})$ y la segunda derivada parcial para esta solución

$$\frac{\partial^2}{\partial p_k^2} R(p_k^*) = f(p_k^*)\left(H'(p_{I_{k-1}}) - H'(p_{I_k})\right) \qquad \text{(B14)}$$

es siempre mayor que cero, de tal modo que p_k^* es la frontera de intervalo $p_k \in [p_{I_{k-1}}, p_{I_k})$ que reduce al mínimo la tasa global esperada R dadas las representantes de intervalo $p_{I_{k-1}}$ y p_{I_k}. Si existen unas probabilidades $p \in [p_{I_{k-1}}, p_{I_k})$ con $f(p) = 0$, la ecuación $\frac{\partial}{\partial p_k} R = 0$ tiene múltiples soluciones, pero p_k^* tal como se da en la eq. B13 sigue siendo óptima incluso a pesar de que pueden existir soluciones óptimas adicionales.

Dado el número de intervalos K y la distribución de probabilidad $f(p)$, las fronteras de intervalo p_k, con $k = 1, \ldots, K-1$,

y las representantes de intervalo p_{lk}, con k = 0, ... , K-1, que reducen al mínimo la tasa global esperada R pueden obtenerse resolviendo el sistema de ecuaciones dado por las ecuaciones B10 y B 13 sujeto a las condiciones B12 para k = 0, ... , K-1. Esto puede conseguirse con el siguiente algoritmo iterativo.

Algoritmo 1:

1) Subdividir en particiones el intervalo (0, 0,5] para dar K intervalos arbitrarios $I_k = (p_k, p_{k+1}]$ con $p_0 = 0$, $p_K = 0{,}5$, y $p_k < p_{k+1}$ para todo k = 0, ... , K-1 de una forma tal que las condiciones B 12 se obedecen para todo k = 0, ... , K-1.
2) Actualizar las representantes p_{l_k} con k = 0, ... , K-1 de acuerdo con la eq. B10
3) Actualizar las fronteras de intervalo p_k con k = 1, ... , K-1 de acuerdo con la eq. B13
4) Repetir las dos etapas previas hasta la convergencia

La figura 14 muestra un ejemplo para la discretización de intervalo óptima usando el algoritmo descrito. Para el presente ejemplo, se supuso una distribución de probabilidad uniforme $f(p) = 2$ para $0 < p \leq 0{,}5$ y se subdividió en particiones el intervalo de probabilidad (0, 0,5] para dar K = 4 intervalos. Puede verse que la discretización de intervalo de probabilidad conduce a una aproximación lineal por trozos $A(p)$ de la función de entropía binaria $H(p)$ con $A(p) \geq H(p)$ para todo $p \in (0, 0{,}5]$.

Como medida para el impacto de la discretización de intervalo sobre la eficiencia de codificación, se puede usar el aumento de tasa global esperada en relación con el límite de entropía

$$\bar{\rho} = \frac{R}{\int_0^{0,5} H(p)\, f(p)\, dp} - 1 \qquad \text{(B15)}$$

Para el ejemplo particular de la figura 14, el valor esperado de la entropía $\bar{H} = \int_0^{0,5} H(p)\, f(p)\, dp$ es igual a 1 / (2 ln 2) bit por contenedor y la tara de tasa $\bar{\rho}$ es igual a un 1,01 %. La tabla 4 enumera unas taras de tasa $\bar{\rho}_{uni}$ y $\bar{\rho}_{lin}$ para la distribución de probabilidad uniforme y una distribución de probabilidad creciente lineal $f(p) = 8p$ con $p \in (0, 0{,}5]$, de forma respectiva, para números seleccionados de intervalos K.

Tabla 4: Tara de tasa frente al número de intervalos de probabilidad para la distribución de probabilidad uniforme y para una distribución de probabilidad creciente lineal

K	1	2	4	8	12	16
$\bar{\rho}_{uni}$ [%]	12,47	3,67	1,01	0,27	0,12	0,07
$\bar{\rho}_{lin}$ [%]	5,68	1,77	0,50	0,14	0,06	0,04

Las investigaciones en esta sección mostraron que la discretización del intervalo de probabilidad de LPB (0, 0,5] en un pequeño número de intervalos con una probabilidad fija (por ejemplo, de 8 a 10 intervalos) tiene un impacto muy pequeño sobre la eficiencia de codificación.

La codificación por entropía que se ha analizado en lo que antecede para intervalos de probabilidad, permite por lo tanto codificadores individuales usando unas probabilidades fijas.

En lo sucesivo, en primer lugar se muestra cómo se puede diseñar un código simple para unas probabilidades fijas. Dados estos resultados, se desarrolla un algoritmo que optimiza de forma conjunta el diseño de código y la subdivisión en particiones del intervalo de probabilidad de LPB (0, 0,5].

La codificación por entropía para unas probabilidades fijas $p = p_{l_k}$ se puede realizar usando codificación aritmética o codificación de longitud variable. Para este último caso, el siguiente enfoque parece ser simple y muy eficiente.

Se considera un esquema de codificación por entropía binaria mediante el cual un número variable de contenedores se pone en correspondencia con palabras de código de longitud variable. Para la capacidad de descodificación única, la puesta en correspondencia inversa de una palabra de código con una secuencia de contenedores ha de ser única. Y debido a que se desea diseñar un código que se aproxime al límite de entropía tanto como sea posible, las consideraciones realizadas por los inventores de la presente invención se limitan a las puestas en correspondencias biyectivas. Una puesta en correspondencia biyectiva de este tipo se puede representar por un árbol binario en el que todos los nodos hoja están asociados con palabras de código, tal como se ilustra en la figura 15. Las aristas del árbol representan eventos binarios. En el ejemplo de la figura 15, las aristas inferiores representan el valor de contenedor de LPB y las aristas superiores representan el valor de contenedor de MPB. El árbol binario representa

un código de prefijo para los contenedores si este es un árbol binario completo, es decir, si cada nodo es o bien una hoja o bien tiene dos descendientes. Cada nodo hoja se asocia con una probabilidad sobre la base de la probabilidad de LPB dada p. El nodo raíz tiene la probabilidad $p_{raíz} = 1$. La probabilidad para todos los otros nodos se obtiene al multiplicar la probabilidad del predecesor correspondiente con p para los descendientes de LPB y $q = 1 - p$ para los descendientes de MPB. Cada nodo hoja L_l se caracteriza por el número de aristas de LPB a_l y el número de aristas de MPB b_l desde el nodo raíz hasta el nodo hoja. Para una probabilidad de LPB particular p, la probabilidad p_l para un nodo hoja $L_l = \{a_l, b_l\}$ es igual a

$$p_l = p^{a_l} (1-p)^{b_l} \qquad \text{(B16)}$$

El árbol binario T está completamente caracterizado por el número de nodos hoja L y los pares asociados $\{a_l, b_l\}$ con $l = 0, \ldots, L-1$.

Dado un árbol binario completo T y una probabilidad de LPB p, la asignación óptima de palabras de código a los nodos hoja puede obtenerse mediante el algoritmo de Huffman. La puesta en correspondencia del número variable resultante de bits con las palabras de código de longitud variable (V2V) C se caracteriza por el número de palabras de código L, el cual es idéntico al número de nodos hoja, y las tuplas $\{a_l, b_l, l_l\}$ para $l = 0, \ldots, L-1$, en las que l_l representa la longitud de palabra de código que se asocia con el nodo hoja correspondiente $L_t = \{a_l, b_l\}$. Se debería hacer notar que hay múltiples posibilidades para la asignación de palabras de código dadas las longitudes de palabra de código $\{l_l\}$ y la asignación de palabras de código real no es importante siempre que las palabras de código representen un código de prefijo que se puede descodificar de forma única. La tasa esperada $R(p, C)$ en bits por contenedor para un código C dado y una probabilidad de LPB p es la relación de la longitud de palabra de código esperada y el número esperado de contenedores por palabra de código

$$R(p,C) = \frac{\sum_{l=0}^{L-1} p_l \, l_l}{\sum_{l=0}^{L-1} p_l (a_l + b_l)} = \frac{\sum_{l=0}^{L-1} p^{a_l} (1-p)^{b_l} \, l_l}{\sum_{l=0}^{L-1} p^{a_l} (1-p)^{b_l} (a_l + b_l)} \qquad \text{(B17)}$$

El diseño de código a menudo está limitado por factores como el número máximo de palabras de código L, el número máximo de contenedores por palabra de código, o la longitud de palabra de código máxima, o este se restringe a códigos de estructuras particulares (por ejemplo, para permitir un análisis sintáctico optimizado). Si se supone que se da el conjunto S_C de códigos utilizables para una aplicación particular, el código óptimo $C^* \in S_C$ para una probabilidad de LPB particular p se puede hallar mediante la reducción al mínimo de la tasa esperada $R(p, C)$

$$C^*(p) = \arg \min_{\forall C \in S_C} R(p,C) \qquad \text{(B18)}$$

Como una alternativa más rápida, se considera que la reducción al mínimo también puede proseguir sobre un conjunto dado de árboles binarios S_T y para cada árbol solo un código de V2V C que se obtiene mediante el algoritmo de Huffman. Como un ejemplo, se diseñaron códigos de V2V para diversas probabilidades de LPB p al considerar todos los árboles binarios T para los que el número de nodos hoja L es menor que o igual a un máximo L_m dado. En la figura 16, el aumento relativo de tasa $\rho(p, C^*(p)) = R(p, C^*(p)) / H(p)$ se representa gráficamente sobre la probabilidad de LPB p para unos tamaños de tabla máximos seleccionados L_m. El aumento de tasa $\rho(p)$ puede reducirse por lo general permitiendo unos tamaños de tabla más grandes. Para unas probabilidades de LPB más grandes, un pequeño tamaño de tabla L de 8 a 16 palabras de código es por lo general suficiente para mantener el aumento de tasa $\rho(p)$ razonablemente pequeño, pero para unas probabilidades de LPB más pequeñas (por ejemplo, $p < 0{,}1$), se requieren unos tamaños de tabla más grandes L.

En las secciones previas, se consideró la discretización de probabilidad óptima suponiendo unos códigos óptimos y el diseño de código para unas probabilidades de LPB fijas. Pero debido a que, en general, no se puede lograr el límite de entropía con códigos de V2V reales de tamaños de tabla limitados, el diseño de código y la subdivisión en particiones del intervalo de probabilidad de LPB (0, 0,5] se han de considerar de forma conjunta para obtener un diseño optimizado de codificación por entropía.

Para un intervalo $I_k = (p_k, p_{k+1}]$ dado, un código C_k de un conjunto dado S_C es un código óptimo C_k^* si este reduce al mínimo la tasa esperada $R = \int_{p_k}^{p_{k+1}} R(p, C_k) \, f(p) \, d\,p$ para el intervalo dado.

$$C_k^* = \arg \min_{\forall C_k \in S_C} \int_{p_k}^{p_{k+1}} R(p, C_k)\, f(p)\, dp \qquad \text{(B19)}$$

Para los diseños prácticos, la reducción al mínimo de la integral en la eq. B19 puede simplificarse, con un impacto mínimo sobre la eficiencia de codificación, determinando en primer lugar una probabilidad representativa óptima $p_{I_k}^*$ para el intervalo I_k de acuerdo con la eq. B10 y entonces elegir el código óptimo C_k^* del conjunto dado S_C para la probabilidad representativa $p_{I_k}^*$ de acuerdo con la eq. B18.

Unas fronteras de intervalo óptimas p_k, con k = 1, ... , *K-1*, dado el conjunto de códigos C_k, con k = 0, ... , *K-1*, se pueden obtener mediante la reducción al mínimo de la tasa global esperada

$$R = R(\{p_k\}, \{C_k\}) = \sum_{k=0}^{K-1} \left(\int_{p_k}^{p_{k+1}} R(p, C_k)\, f(p)\, dp \right) \qquad \text{(B20)}$$

Establecer las primeras derivadas con respecto a las fronteras de intervalo iguales a cero, $\frac{\partial}{\partial p_k} R = 0$, para k = 1, ... , *K-1*, da

$$p_k^* = p_k \quad \text{con} \quad R(p_k, C_{k-1}) = R(p_k, C_k) \qquad \text{(B21)}$$

De forma similar a la eq. B 13, se puede mostrar que p_k^* es siempre una solución óptima, pero dependiendo de la distribución de probabilidad $f(p)$ podrían existir soluciones óptimas adicionales. Por lo tanto, una frontera de intervalo óptima p_k^* entre dos intervalos I_{k-1} e I_k con códigos asociados C_{k-1} y C_k dados, de forma respectiva, es el punto de intersección de las funciones $R(p, C_{k-1})$ y $R(p, C_k)$.

En consecuencia, el siguiente algoritmo interactivo se puede usar para obtener de forma conjunta la subdivisión en particiones de intervalos de probabilidad y los códigos asociados dado el número K de intervalos de probabilidad, el conjunto de posibles códigos S_C, y la distribución de probabilidad $f(p)$, con $p \in (0, 0{,}5]$.

Algoritmo 2:

1) Obtener los límites de intervalo de probabilidad inicial p_k, con k = 0, ... , K, usando el algoritmo 1 que se especifica en la sec. 3
2) Obtener unas representantes p_{lk} para los intervalos de probabilidad I_k, con k = 0, ... , K -1, de acuerdo con la eq. B10
3) Obtener códigos $C_k \in S_C$ para las representantes de intervalo p_{lk}, con k = 0, ... , K -1, de acuerdo con la eq. B18
4) Actualizar fronteras de intervalo p_k, con k = 1, ... , *K-1*, de acuerdo con la eq. B21
5) Repetir las tres etapas previas hasta la convergencia

Las etapas 2 y 3 en el algoritmo 2 también se podrían sustituir por una obtención directa de los códigos $C_k \in S_C$, con k = 0, ... , K-1, sobre la base de las fronteras de intervalo p_k, con k = 0, ... , K, de acuerdo con la eq. B19. Y, tal como se ha mencionado en la sec. 4.1, la reducción al mínimo en la etapa 3 también puede proseguir sobre un conjunto dado de árboles binarios S_T en los que para cada árbol binario T solo se considera un código de V2V C_k obtenido mediante el algoritmo de Huffman.

Como un ejemplo, se obtuvo de forma conjunta la subdivisión en particiones para dar K = 12 intervalos de probabilidad y códigos de V2V correspondientes usando el algoritmo 2. En este, la reducción al mínimo en la etapa 3 del algoritmo se sustituyó con una reducción al mínimo equivalente sobre un conjunto dado de árboles binarios S_T en los que el código evaluado C para cada árbol T se obtuvo mediante el algoritmo de Huffman. Se consideraron unos árboles T con un número máximo de L_m = 65 nodos hoja y, por lo tanto, códigos C con hasta 65 entradas de tabla. Todos los árboles binarios T con hasta 16 nodos hoja se han evaluado en la reducción al mínimo; para unos árboles con más de 16 nodos hoja, se empleó una búsqueda subóptima dados los mejores resultados para unos árboles con un número más pequeño de nodos hoja.

En la figura 17 el aumento de tasa esperada en relación con el límite de entropía $\Delta R(p) = R(p) - H(p)$ para el ejemplo

de diseño de código se representa gráficamente sobre la probabilidad de LPB *p*. Como comparación, también se representó gráficamente el aumento de tasa esperada Δ*R* para la discretización de intervalo de probabilidad teóricamente óptima (tal como se desarrolla en la sec. 3) y la discretización de probabilidad teóricamente óptima con la restricción adicional $p_{lk\text{-}1}$ = 0,5 en el interior del diagrama. Puede verse que la discretización de intervalo de probabilidad conjunta y el diseño de código de V2V conduce a un desplazamiento de las fronteras de intervalo (las fronteras de intervalo p_k, con *k* = 1, ... , *K-1*, vienen dadas por los máximos locales de las curvas de Δ*R*(*p*)). El aumento relativo de tasa global esperada en relación con el límite de entropía para el ejemplo de diseño con códigos de V2V reales es *p* = 0,24 %, cuando se supone una distribución de probabilidad uniforme *f*(*p*). Los aumentos relativos de tasa correspondientes para la discretización de intervalo de probabilidad teóricamente óptima y la discretización de probabilidad teóricamente óptima con la restricción adicional $p_{lk\text{-}1}$ = 0,5 son ρ = 0,12 % y ρ = 0,13 %, de forma respectiva.

La terminación de palabras de código se puede realizar tal como sigue. Cuando se codifica una secuencia finita de símbolos {s_0, ... , $s_{N\text{-}1}$}, cada uno de los *K* codificadores binarios procesa una secuencia finita de contenedores de codificación $\boldsymbol{b}_k^c = \{b_0^c, \ldots, b_{B_k-1}^c\}_k$, con *k* = 0, ... , *K-1*. Y se ha de haber asegurado que, para cada uno de los *K* codificadores binarios, todos los contenedores de codificación de la secuencia $\boldsymbol{b}_k^c = \{b_0^c, \ldots, b_{B_k-1}^c\}_k$ se pueden reconstruir dada la palabra de código o la secuencia de palabras de código $\mathbf{c}_k(\mathbf{b}_k^c)$.

Cuando se emplea la codificación aritmética, la palabra de código aritmética para la secuencia de contenedores de codificación ha de determinarse de una forma tal que todos los contenedores de codificación se puedan descodificar dada la palabra de código. Para los códigos de V2V que se han descrito en lo que antecede, los contenedores al final de la secuencia $\mathbf{b}_k^c$ pueden no representar una secuencia de contenedores que se asocia con una palabra de código. En caso de este tipo, se puede escribir cualquier palabra de código que contenga la secuencia de contenedores restante como prefijo. La tara se puede reducir al mínimo si se elige la palabra de código correspondiente que tiene la longitud mínima (o una de estas palabras de código). En el lado de descodificador, se descartan los contenedores leídos adicionalmente al final de la secuencia de contenedores, los cuales se pueden identificar dados los esquemas de binarización y de sintaxis de trenes de bits.

Un ejemplo de diseño de código simple se presenta en lo sucesivo. Por razones de ilustración, se considera el ejemplo simple de un origen {*s*} con tres letras y unas probabilidades asociadas fijas de $p_s(a_0)$ = 0,7, $p_s(a_1)$ = 0,18, y $p_s(a_2)$ = 0,12. El árbol de elección ternario correspondiente puede convertirse en un árbol binario completo tal como se muestra en la figura 18.

Una binarización para el árbol binario completo en la figura 18 se da en la tabla 5. El símbolo ternario pmf p_s se convierte a dos pmf binarias p_{b0} = (0,7, 0,3) y p_{b1} = (0,6, 0,4). Para cada símbolo *s* en el tren de bits, el contenedor b_0 se encuentra presente. Cuando b_0 es igual a 0, también b_1 se encuentra presente. Obsérvese que la binarización que se da en la tabla 2 es idéntica a un código de Huffman de una sola letra óptimo para el origen *s*.

Tabla 5: Binarización de un origen de tres letras. Las probabilidades de LPB p_{LPB} son 0,3 para el primer contenedor y 0,4 para el segundo contenedor

Símbolo a_i	Probabilidad $p(a_i)$	Contenedor b_0	Contenedor b_1
a0	0,7	1	
*a*1	0,18	0	1
a2	0,12	0	0
LPB Prob.	*p LPB* = *p*(*bj* = 0)	0,3	0,4

La entropía para el origen *s* es

$$H(0{,}7, 0{,}18, 0{,}12) = H(0{,}7, 0{,}3) + 0{,}3\, H(0{,}6, 0{,}4)$$

$$= 1{,}1726 \text{ bits / símbolo} \qquad \text{(B22)}$$

La longitud de palabra de código promedio del código de Huffman de una sola letra se da como

$$\bar{\ell}_{HC} = \sum_{i=0}^{M-1} p_i \ell_i^{HC} = 1{,}3 \; bits \,/\, símbolo \qquad \text{(B23)}$$

que se corresponde con una redundancia de ρ_{HC} = 0,1274 bits / símbolo o un 10,87 % de la tara de tasa esperada.

Para el ejemplo de binarización particular con unas pmf fijas, los contenedores b_0 y b_1 ya representan unos contenedores de codificación, debido a que para ambos contenedores, el valor de LPB b'_{LPB} es igual a 0. La distribución $f(s)$ de las probabilidades de LPB es discreta, con $f(p) = 0$ excepto por p = 0,3 y p = 0,4. En consecuencia, la discretización de probabilidad óptima con duce a K = 2 intervalos con las representantes p_{l_0} = 0,3 y p_{l_1}, = 0,4. La frontera de intervalo p_1 entre estos intervalos se puede elegir de forma arbitraria en [0,3, 0,4).

Para codificar el origen, la secuencia de símbolos de origen se binariza para dar una secuencia de contenedores. El contenedor b_0 se transmite para cada símbolo de origen. El contenedor b_1 solo se transmite cuando b_0 = 0. Los contenedores b_0 y b_1 se codifican por separado con unas probabilidades de LPB constantes de p_{l_0} = 0,3 y p_{l_1} = 0,4, de forma respectiva.

Una codificación eficiente de un alfabeto binario con probabilidad fija puede conseguirse mediante una puesta en correspondencia de V2V simple. En la tabla 6 y la tabla 7 se dan, de forma respectiva, ejemplos para puestas en correspondencia de V2V con unas tablas de codificación pequeñas para las probabilidades de LPB p_{LPB} = 0,3 y p_{LPB} = 0,4. La puesta en correspondencia de V2V para p_{LPB} = 0,3 da una redundancia de 0,0069 *bits /* contenedor o un 0,788 %. Para la probabilidad de LPB de p_{LPB} = 0,4, la redundancia es 0,0053 *bits /* contenedor o un 0,548 %.

Tabla 6: Árbol de contenedores y códigos para una probabilidad de LPB de p_{LPB} = 0,3. La redundancia de este código es de un 0,788 %

Árbol de contenedores	Probabilidad	Códigos
‘11’	0,72 = 0,4 9	‘1’
‘01’	0,7 · 0,3 = 0,21	‘01’
‘0’	0,3	‘00’

Tabla 7: Árbol de contenedores y códigos para una probabilidad de LPB de p_{LPB} = 0,4. La redundancia de este código es de un 0,548 %

Árbol de contenedores	Probabilidad	Árbol de códigos
‘111’	0,63 = 0,216	‘11’
‘110’		‘001’
	0,62 · 0,4 = 0,144	
‘10’	0,6 · 0,4 = 0,24	‘11’
‘01’	0,4 · 0,6 = 0,24	‘01’
‘00’	0,42 = 0,16	‘000’

La tasa esperada global provocada por el nuevo método de codificación es

$$\bar{\ell}_{NC} = \bar{\ell}_{b_0} + 0{,}3 \cdot \bar{\ell}_{b_1} = 1{,}181 \; bits \,/\, símbolo \qquad \text{(B24)}$$

La redundancia global es de un 0,73 % en relación con el límite de entropía, lo cual representa una mejora significativa en comparación con el código de Huffman de una sola letra.

Se podría argumentar que se podría obtener una mejora en la eficiencia de codificación similar creando un código de longitud de series. Para el ejemplo anterior, se podría construir un código de longitud de series para el símbolo más probable al considerar unas series de hasta dos símbolos. Cada uno de los eventos {a_0a_0, a_0a_1, a_0a_2, a_1, a_2} se asociaría con una palabra de código separada. Un código de este tipo da una redundancia de un 1,34 % en relación con el límite de entropía. En la práctica, los códigos de V2V se pueden considerar como una generalización de los códigos de longitud de series para símbolos binarios (el código de V2V en la tabla 3 representa, de hecho, de forma

efectiva un código de longitud de series). Para un único alfabeto de símbolos con unas probabilidades fijas, una eficiencia de codificación similar como para el enfoque presentado también puede conseguirse creando un código que pone en correspondencia un número variable de símbolos de origen con palabras de código de longitud variable. La ventaja principal del enfoque presentado es su flexibilidad en la puesta en correspondencia de unas secuencias de símbolos de origen arbitrarias con unas estimadas de probabilidad fijas o adaptativas con un pequeño número de codificadores binarios simples que se operan con unas probabilidades de LPB fijas.

Se considera a continuación cómo lograr una capacidad de descodificación única.

Con el esquema de codificación por entropía presentado, la codificación de una secuencia de símbolos de origen $\boldsymbol{s} = \{s_0, \ldots, s_{N-1}\}$ consiste en las siguientes tres etapas básicas:

- binarización de símbolos $\boldsymbol{b} = \{b_0, \ldots, b_{B-1}\} = \gamma_b(\mathbf{s})$ que da una secuencia de contenedores $\boldsymbol{b} = \{b_0, \ldots, b_{B-1}\}$
- conversión de la secuencia de contenedores en una secuencia de contenedores de codificación $\boldsymbol{b}^c = \{b_0^c, \ldots, b_{B-1}^c\} = \gamma_c(\mathbf{b})$
- codificación por entropía binaria de la secuencia de contenedores de codificación $\boldsymbol{b}^c = \{b_0^c, \ldots, b_{B-1}^c\}$ usando una discretización de intervalo de probabilidad y K codificadores binarios fijos

La secuencia de símbolos $s = \{s_0, \ldots, s_{N-1}\}$ se puede descodificar de forma única, si la secuencia de contenedores de codificación $\boldsymbol{b}^c = \{b_0^c, \ldots, b_{B-1}^c\}$ se puede descodificar de forma única y las puestas en correspondencia γ_b y γ_c son invertibles.

Permítase que γ_e notifique la puesta en correspondencia de codificador de una secuencia de uno o más contenedores de codificación $\boldsymbol{b}^c = \{b_0^c, \ldots\}$ con una secuencia de una o más palabras de código $\boldsymbol{c}(\boldsymbol{b}^c) = \{c_0, \ldots\}$

$$\boldsymbol{c}(\boldsymbol{b}^c) = \gamma_e(\mathbf{b}^c) \qquad \text{(B25)}$$

Para una capacidad de descodificación única de una secuencia de contenedores de codificación $\boldsymbol{b}^c$ dada la secuencia de palabras de código $\boldsymbol{c}(\boldsymbol{b}^c)$, la puesta en correspondencia de codificador γ_e debe tener la propiedad de que una palabra de código única $\boldsymbol{c}(\boldsymbol{b}^c)$ se asigna a cada posible secuencia de contenedores de codificación $\boldsymbol{b}^c$:

$$\forall\, \boldsymbol{b}_i^c, \boldsymbol{b}_j^c \quad \boldsymbol{b}_i^c \neq \boldsymbol{b}_j^c \;\rightarrow\; \boldsymbol{c}(\boldsymbol{b}_i^c) \neq \boldsymbol{c}(\boldsymbol{b}_j^c) \qquad \text{(B26)}$$

Esta propiedad se satisface siempre cuando se usan códigos aritméticos o códigos de prefijo. Esta se satisface en particular para los códigos de V2V que se describen en la sec. 4.1 (incluyendo la terminación de palabras de código que se describe en la sec. 4.3), debido a que los códigos de V2V representan códigos de prefijo para números variables de contenedores.

No obstante, en el enfoque de codificación por entropía binaria presentado, la secuencia de contenedores de codificación $\boldsymbol{b}^c$ se subdivide en particiones para dar K subsecuencias $\boldsymbol{b}_k^c$, con $k = 0, \ldots, K-1$,

$$\{\boldsymbol{b}_0^c, \ldots, \boldsymbol{b}_{K-1}^c\} = \gamma_p(\mathbf{b}^c) \qquad \text{(B27)}$$

y a cada una de las subsecuencias $\boldsymbol{b}_k^c$, se asigna una secuencia de palabras de código $\boldsymbol{c}_k(\boldsymbol{b}_k^c)$ usando una puesta en correspondencia de codificador particular γ_e^k. En consecuencia, se ha de ampliar la condición sobre la capacidad de descodificación única. Una secuencia de contenedores de codificación $\boldsymbol{b}^c$ se puede descodificar de forma única dadas K secuencias de palabras de código $\boldsymbol{c}_k(\boldsymbol{b}_k^c)$, con $k = 0, \ldots, K-1$, si cada subsecuencia de contenedores de codificación $\boldsymbol{b}_k^c$ se puede descodificar de forma única dada la palabra de código correspondiente $\boldsymbol{c}_k(\boldsymbol{b}_k^c)$ y la regla de subdivisión en particiones γ_p es conocida por el descodificador. La regla de subdivisión en particiones γ_p viene dada por la discretización de intervalo de probabilidad de LPB $\{l_k\}$ y las probabilidades de LPB p_{LPB}^j que están asociadas con los contenedores de codificación b_j^c, con $j = 0, \ldots, B-1$. Por lo tanto, la discretización de intervalo de probabilidad de LPB $\{l_k\}$ ha de ser conocida en el lado de descodificador y la probabilidad de LPB p_{LPB}^j para cada contenedor de codificación b_j^c, con $j = 0, \ldots, B-1$, se ha de obtener de la misma forma en el lado de codificador y de descodificador.

Para la puesta en correspondencia γ_c de una secuencia de contenedores con una secuencia de contenedores de codificación *c*, cada único b_j, con *j* = 0, ... , *B*-1, se convierte por la puesta en correspondencia binaria $b_j^c = \gamma_c^j(b_j) = b_j \oplus b_{LPB}^j$. En el lado de descodificador, la secuencia de contenedores se puede obtener mediante las puestas en correspondencia binarias

$$b_j = (\gamma_c^j)^{-1}(b_j^c) = b_j^c \oplus b_{LPB}^j \qquad \text{(B28)}$$

con *j* = 0, ... , *B*-1. Si el valor de LPB b^j_{LPB} para cada contenedor b_j se obtiene de la misma forma en el lado de codificador y de descodificador, estas puestas en correspondencia $(\gamma_c^j)^{-1}$ representan las inversas de las puestas en correspondencia de codificador correspondientes γ_c^j , debido a que

$$b_j^c \oplus b_{LPB}^j = b_j \oplus b_{LPB}^j \oplus b_{LPB}^j = b_j \oplus 0 = b_j \qquad \text{(B29)}$$

y por lo tanto, la conversión γ_b de una secuencia de contenedores ***b*** en una secuencia de contenedores de codificación $\boldsymbol{b}^c$ es invertible.

Por último, se investiga la invertibilidad de la binarización ***b*** = γ_b(***s***) mediante la cual cada símbolo s_i, con *i* = 0, ... , *N*-1, se pone en correspondencia con una secuencia de contenedores $\boldsymbol{b}^i = \gamma_b^i(s_i)$. Un símbolo s_i se puede descodificar de forma única dada la secuencia de contenedores correspondiente $\boldsymbol{b}^i$ si la puesta en correspondencia de binarización γ_b^i asigna una secuencia de contenedores diferente $\boldsymbol{b}_m^j$ a cada letra a_m^i del alfabeto A_i para el símbolo s_i. No obstante, esta condición no es suficiente, debido a que la subdivisión en particiones de la secuencia de contenedores $b = \{b_0, \ldots, b_{B-1}\}$ en unas secuencias de contenedores $\boldsymbol{b}^i$ que se corresponden con los símbolos s_i, con *i* = 0, ... , *N*-1, no es conocida por el descodificador. Se da una condición suficiente, cuando para cada símbolo s_i, las secuencias de contenedores $\boldsymbol{b}_m^j$ que están asociadas con las letras a_m^i del alfabeto correspondiente A_i forman un código de prefijo y las puestas en correspondencia de binarización γ_b^i para cada símbolo s_i, con *i* = 0, ... , *N*-1, se conocen en el lado de descodificador.

Las condiciones de la capacidad de descodificación única para el enfoque de codificación por entropía binaria presentado se pueden resumir tal como sigue:

- las puestas en correspondencia de binarización γ_b^i representan códigos de prefijo y son conocidas por el descodificador (en el orden de codificación de símbolos)
- los modelos de probabilidad (b^j_{LPB}, p^j_{LPB}) para todos los contenedores b_j se obtienen de la misma forma en el lado de codificador y de descodificador
- la subdivisión en particiones del intervalo de probabilidad de LPB (0, 0,5] en *K* intervalos I_k, con *k* = 0, ... , *K*-1, es conocida por el descodificador
- la puesta en correspondencia γ_e^k para cada intervalo de probabilidad I_k, con *k* = 0, ... , *K*-1, representa un código que se puede descodificar de forma única

En lo sucesivo, se describen con más detalle ejemplos para el diseño global de codificador y de descodificador. Los inventores de la presente invención se centran en los esquemas de codificación en los que los modelos de probabilidad {b_{LPB}, p_{LPB}} para los contenedores son estimados directamente en el lado de codificador y de descodificador y los *K* codificadores binarios usan unas puestas en correspondencia de V2V que se han descrito en lo que antecede. Cada símbolo de origen *s* se asociará con una categoría de símbolos c_s, la cual determina el tipo del símbolo incluyendo su intervalo de valores. El orden de los símbolos y las categorías de símbolos asociadas debería venir dado por la sintaxis, que se presupone que se conoce en el lado de codificador y de descodificador.

El diagrama de bloques para un ejemplo diseño de codificador de PIPE y de descodificador de PIPE se ilustra en la figura 19. En el lado de codificador, los símbolos *s* con unas categorías de símbolos asociadas c_s se alimentan a la unidad de binarización, que convierte cada símbolo *s* en una secuencia de contenedores $\boldsymbol{b}_s = \gamma_b^{c_s}(s)$.

El esquema de binarización $\gamma_b^{c_s}$ usado se determina sobre la base de la categoría de símbolos c_s. Además, la unidad de binarización asocia cada contenedor *b* de una secuencia de contenedores *s* con una indicación de modelo de probabilidad c_b, la cual especifica el modelo de probabilidad que se usa para codificar el contenedor *b*. La indicación de modelo de probabilidad c_b se puede obtener sobre la base de la categoría de símbolos c_s, el número

de contenedor del contenedor actual en el interior de la secuencia de contenedores *s*, y / o los valores de contenedores y símbolos ya codificados.

El *estimador de probabilidad y unidad de asignación* mantiene múltiples modelos de probabilidad, que se caracterizan por unos pares de valores $\{b_{LPB}, p_{LPB}\}$. Este recibió unos contenedores *b* y unas indicaciones de modelo de probabilidad asociadas c_b a partir de la unidad de binarización, y reenvía el valor de LPB b_{LPB} y la probabilidad de LPB p_{LPB} del modelo de probabilidad indicado a la unidad de obtención de contenedores de codificación y la unidad de cuantificación de probabilidad, de forma respectiva. A continuación de lo anterior, el modelo de probabilidad correspondiente $\{b_{LPB}, p_{LPB}\}$ se actualiza usando el valor del contenedor recibido *b*.

La *unidad de obtención de contenedores de codificación* recibe los contenedores *b* y los *valores* de LPB asociados b_{LPB} a partir de la unidad de binarización y el estimador de probabilidad y la unidad de asignación, de forma respectiva, y envía unos contenedores de codificación b^c, los cuales se obtienen mediante $b^c = b \oplus b_{LPB}$, a la unidad de cuantificación de probabilidad. La *unidad de cuantificación de probabilidad* reenvía cada contenedor de codificación b^c a uno de los *K* codificadores binarios. Esta contiene información sobre la cuantificación de intervalo de probabilidad de LPB $\{l_k\}$. La probabilidad de LPB p_{LPB}, que se asocia con un contenedor de codificación b^c y se recibe a partir del estimador de probabilidad y la unidad de asignación, se compara con las fronteras de intervalo $\{p_k\}$ y el índice de intervalo de probabilidad *k*, para el cual se obtiene $p_{LPB} \in l_k$. Entonces, el contenedor de codificación b^c se reenvía al codificador binario asociado.

Cada uno de los *K* codificadores binarios consiste en una *memoria intermedia de contenedores* y un *codificador de contenedores*. La memoria intermedia de contenedores recibe unos contenedores de codificación b^c a partir de la unidad de cuantificación de probabilidad y almacena estos en el orden de codificación. El codificador de contenedores implementa una puesta en correspondencia de V2V particular y compara la secuencia de contenedores en la memoria intermedia de contenedores con las secuencias de contenedores que están asociadas con palabras de código. Si la secuencia de contenedores en la memoria intermedia de contenedores es igual a una de esas secuencias de contenedores, el codificador de contenedores retira la secuencia de contenedores $\{b^c\}$ de la memoria intermedia de contenedores y escribe la palabra de código asociada ($\{b^c\}$) en el tren de palabras de código correspondiente. Al final del proceso de codificación para una secuencia de símbolos, para todos los codificadores binarios para los cuales las memorias intermedias de contenedores no están vacías, una palabra de código de terminación se escribe tal como se describe en la sec. 4.3.

Los *K* trenes de palabras de código resultantes se pueden transmitir, paquetizar o almacenar por separado, o estos se pueden intercalar (compárese con la sec. 6.2) para el fin de transmisión o almacenamiento.

En el lado de descodificador, cada uno de los *K* descodificadores binarios que consisten en un descodificador de contenedores y una memoria intermedia de contenedores recibe un tren de palabras de código. El descodificador de contenedores lee las palabras de código ($\{b^c\}$) a partir del tren de palabras de código e inserta la secuencia de contenedores asociada $\{b^c\}$, en el orden de codificación, en la memoria intermedia de contenedores.

La descodificación de la secuencia de símbolos es impulsada por la sintaxis subyacente. Solicitudes para un símbolo *s* se envían junto con la categoría de símbolos c_s a la unidad de binarización. La unidad de binarización convierte estas solicitudes de símbolo en una solicitud para contenedores. Una solicitud para un contenedor se asocia con una indicación de modelo de probabilidad c_b, la cual se obtiene de la misma forma que en el codificador, y se envía al estimador de probabilidad y la unidad de asignación. El *estimador de probabilidad* y la *unidad de asignación* se operan de forma similar a su homólogo en el lado de codificador. Sobre la base de la indicación de modelo de probabilidad c_b, esta identifica un modelo de probabilidad y reenvía su valor de LPB b_{LPB} y la *probabilidad* de LPB p_{LPB} a la unidad de obtención de contenedores y la unidad de cuantificación de probabilidad, de forma respectiva.

La *unidad de cuantificación de probabilidad* determina uno de los *K* descodificadores binarios sobre la base de la probabilidad de LPB p_{LPB}, de la misma forma que el codificador binario se determina en el lado de codificador, retira el primer contenedor de codificación b^c, en el orden de codificación, de la memoria intermedia de contenedores correspondiente, y lo reenvía a la unidad de obtención de contenedores. La *unidad de obtención de contenedores* recibe unos contenedores de codificación b^c y los valores de LPB asociados b_{LPB} a partir de la unidad de cuantificación de probabilidad y el estimador de probabilidad y la unidad de asignación, de forma respectiva, y determina los valores de contenedor $b = b^c \oplus b_{LPB}$. Como respuesta final a una solicitud de contenedor enviada por la unidad de binarización, la unidad de obtención de contenedores envía el valor de contenedor descodificado b a la unidad de binarización y el estimador de probabilidad y la unidad de asignación.

En el estimador de probabilidad y la unidad de asignación, el valor del contenedor descodificado *b* se usa para actualizar el modelo de probabilidad $\{b_{LPB}, p_{LPB}\}$, el cual fue elegido por el valor asociado c_b, de la misma forma que en el lado de codificador. Por último, la unidad de binarización añade el contenedor recibido *b* a la secuencia de contenedores *s* la cual ya se ha recibido para una solicitud de símbolo y compara esta secuencia de contenedores *s*

con las secuencias de contenedores que están asociadas con unos valores de símbolo por el esquema de binarización $\gamma_b^{c_s}$. Si la secuencia de contenedores *s* coincide con una de esas secuencias de contenedores, el símbolo descodificado correspondiente *s* se emite como respuesta final a la solicitud de símbolo. De lo contrario, la unidad de binarización envía solicitudes de contenedor adicionales hasta que el símbolo *s* se ha descodificado.

La descodificación de una secuencia de símbolos se termina si no se reciben solicitudes de símbolo adicionales, las cuales son impulsadas por la sintaxis. Se descartan los contenedores de codificación b^c que pueden estar contenidos en las memorias intermedias de contenedores al final del proceso de descodificación por entropía (como resultado de las palabras de código de terminación).

Después de haber descrito determinados codificadores de PIPE y descodificadores de PIPE con respecto a las figuras 3 a 13 y de haber proporcionado un trasfondo matemático en lo que respecta a la codificación de PIPE en general con respecto a las figuras 14 a 19, con respecto a las figuras 20 a 24, se describen más detalles para aparatos de codificación y descodificación por entropía. Los siguientes ejemplos de las figuras 22 a 24 no solo intercalan los trenes de bits codificados de PIPE entre sí, sino que intercalan el tren de bits de VLC y los trenes de bits codificados de PIPE completamente. En comparación con esto, los codificadores de PIPE y los descodificadores de PIPE con una intercalación para los trenes de bits codificados de PIPE, que se muestran en las figuras 7 a 13, proporcionados meramente para una intercalación separada de los trenes de bits codificados de PIPE. Tal como ya se ha mencionado en lo que antecede, incluso estos mismos pueden servir como base para conseguir un tren de bits completamente intercalado mediante el uso de otro par de intercalador / desintercalador 134 y 228, de forma respectiva (véanse las figuras 1 y 2), tal como, por ejemplo, mediante el uso de una intercalación de trenes de bits tal como se muestra en las figuras 5 y 6. No obstante, las posibilidades que se describen a continuación realizan la intercalación al mismo tiempo tanto en el tren de bits de VLC como en los trenes de bits codificados de PIPE con el uso, dicho de otra forma, el intercalador / desintercalador de una fase 128 y 230, de forma respectiva.

Antes de describir con detalle casos en los que unos símbolos codificados de VLC y de PIPE se intercalan en el interior de un tren de bits, con el fin de conseguir una compensación recíproca más adecuada entre la complejidad y la eficiencia de codificación, una estructura básica del mismo sin una intercalación se describe con respecto a la figura 20 y 21.

La estructura del aparato de codificación por entropía de la figura 1a se muestra en la figura 20. El aparato de codificación por entropía convierte un tren de símbolos de origen 1a, que se corresponden con la combinación de símbolos de origen codificados de VLC y codificados de PIPE de las figuras 1 y 2, en concreto 106 y 218, de forma respectiva, en un conjunto de dos o más trenes de bits parciales 12, 12a, con el tren de bits 12a correspondiéndose con los trenes de bits 112 y 206 de las figuras 1 y 2.

Tal como ya se ha hecho notar en lo que antecede, cada símbolo de origen 1a puede tener asociado al mismo una indicación que especifica si el símbolo de origen se codifica usando códigos de VLC convencionales dentro del codificador de VLC 22a, el cual se corresponde con el codificador de VLC 102 en la figura 1, o en lo que respecta a si el símbolo de origen se va a codificar con un concepto de codificación de PIPE. Tal como ya se ha descrito en lo que antecede con respecto a las figuras 1 y 2, esta indicación puede no transmitirse de forma explícita al lado de descodificación. En su lugar, la indicación asociada puede resultar del tipo o la categoría del propio símbolo de origen.

Los símbolos codificados de VLC 1b se codifican con códigos de VLC convencionales, que a su vez, pueden depender de la categoría de símbolos o el tipo de símbolo recién mencionado usando un codificador de VLC 22a. Las palabras de código 11a correspondientes se escriben en un tren de bits parcial distinto 12a. Los símbolos no de código VLC 1 se codifican usando codificación de PIPE tal como se ha descrito en lo que antecede con respecto a las figuras 1 y 3, por ejemplo, con lo cual se obtienen múltiples trenes de bits parciales 12. Algunos de los símbolos de origen 1a ya se pueden haber binarizado de una forma tal que la binarización consiste en dos partes tal como ya se ha mencionado en lo que antecede con respecto a la figura 1a. Una de estas partes se puede codificar con el enfoque de PIPE y escribirse en los trenes de bits parciales 12 correspondientes. La otra parte de la secuencia de contenedores se puede codificar con los códigos de VLC convencionales y escribirse en el tren de bits parcial 12a correspondiente.

El aparato de descodificación por entropía básico adecuado para la figura 20 se muestra en la figura 21.

El descodificador realiza básicamente las operaciones inversas del codificador de la figura 20, de tal modo que la secuencia previamente codificada de símbolos de origen 27, 27a se descodifica a partir de un conjunto de dos o más trenes de bits parciales (24, 24a). El descodificador incluye dos flujos de proceso diferentes: Un flujo para solicitudes de datos, el cual reproduce el flujo de datos del codificador, y un flujo de datos, el cual representa la inversa del flujo de datos de codificador. En la ilustración en la figura 21 las flechas de trazo discontinuo representan el flujo de solicitudes de datos, mientras que las flechas de trazo continuo representan el flujo de datos. Los bloques básicos del descodificador básicamente reproducen los bloques básicos del codificador, pero implementan las operaciones

inversas.

Cada solicitud de símbolo 13a puede asociarse con una indicación que especifica si el símbolo de origen se codifica usando códigos de VLC convencionales o con el concepto de codificación de PIPE. Tal como ya se ha mencionado en lo que antecede con respecto a la figura 20, esta indicación puede resultar de las reglas de análisis sintáctico o la sintaxis de los elementos de sintaxis representados por el propio símbolo de origen. Por ejemplo, con respecto a las figuras 1 y 2, se ha descrito que diferentes tipos de elemento de sintaxis pueden asociarse con los diferentes esquemas de codificación, en concreto la codificación de VLC o la codificación de PIPE. Lo mismo puede ser de aplicación para diferentes porciones de binarizaciones, o más en general, otras simbolizaciones de los elementos de sintaxis. Si un símbolo se somete a codificación de VLC, la solicitud se pasa al descodificador de VLC 22a y una palabra de código de VCL 23a se lee a partir de un tren de bits parcial distinto 24a. El símbolo de descodificación 27a correspondiente se emite. Si un símbolo se codifica con PIPE, el símbolo 27 se descodifica a partir de un conjunto de trenes de bits parciales 24 tal como se ha descrito en lo que antecede con respecto a la figura 4, por ejemplo.

Algunos de los símbolos de origen se pueden binarizar de una forma tal que la binarización consiste en dos partes. Una de estas partes se codifica con el enfoque de PIPE se descodifica de forma correspondiente a partir de los trenes de bits parciales asociados 24. Y la otra parte de la secuencia de contenedores se codifica con códigos de VLC convencionales y se descodifica con un descodificador de VLC 22a que lee las palabras de código 23a correspondientes a partir de un tren de bits parcial distinto 24a.

Transmisión y multiplexación de los trenes de bits parciales (sometidos a codificación de VLC y sometidos a codificación de PIPE)

Los trenes de bits parciales 12, 12a que se crean por el codificador se pueden transmitir por separado, o estos se pueden multiplexar para dar un único tren de bits, o las palabras de código de los trenes de bits parciales se pueden intercalar en un único tren de bits.

Cada tren de bits parcial para una cantidad de datos se puede escribir en un paquete de datos. La cantidad de datos puede ser un conjunto arbitrario de símbolos de origen tales como una imagen fija, un campo o trama de una secuencia de vídeo, una rebanada de una imagen fija, una rebanada de un campo o trama de una secuencia de vídeo, o una trama de muestras de audio, etc.

Dos o más de los trenes de bits parciales 12, 12a para una cantidad de datos o todos los trenes de bits parciales para una cantidad de datos se pueden multiplexar para dar un paquete de datos. La estructura de un paquete de datos que contiene unos trenes de bits parciales multiplexados puede ser tal como se ilustra en la figura 5.

El paquete de datos 300 consiste en un encabezamiento y una partición para los datos de cada tren de bits parcial (para la cantidad considerada de datos). El encabezamiento 301 del paquete de datos contiene indicaciones para la subdivisión en particiones de (el resto de) el paquete de datos en segmentos de los datos de tren de bits 302. Además de las indicaciones para la subdivisión en particiones, el encabezamiento puede contener una información adicional. Las indicaciones para la subdivisión en particiones del paquete de datos pueden ser las ubicaciones del comienzo de los segmentos de datos en unidades de bits o bytes o múltiples de bits o múltiples de bytes. Las ubicaciones del comienzo de los segmentos de datos se pueden codificar como valores absolutos en el encabezamiento del paquete de datos, o bien en relación con el comienzo del paquete de datos o en relación con el final del encabezamiento o bien en relación con el comienzo del paquete de datos previo. Las ubicaciones del comienzo de los segmentos de datos se pueden codificar de forma diferencial, es decir, solo se codifica la diferencia entre el comienzo real de un segmento de datos y una predicción para el comienzo del segmento de datos. La predicción se puede obtener sobre la base de una información ya conocida o transmitida tal como el tamaño global del paquete de datos, el tamaño del encabezamiento, el número de segmentos de datos en el paquete de datos, la ubicación del comienzo de segmentos de datos precedentes. La ubicación del comienzo del primer paquete de datos puede no codificarse, sino inferirse sobre la base del tamaño del encabezamiento de paquete de datos. En el lado de descodificador, las indicaciones de partición transmitidas se usan para obtener el comienzo de los segmentos de datos. Los segmentos de datos se usan entonces como los trenes de bits parciales 12, 12a y los datos que están contenidos en los segmentos de datos se alimentan a los descodificadores de contenedores y los descodificadores de VLC correspondientes en orden secuencial.

Intercalación de palabras de código (palabras de código de VLC y de PIPE)

Para algunas aplicaciones, la multiplexación que se ha descrito en lo que antecede de los trenes de bits parciales (para una cantidad de símbolos de origen) en un paquete de datos puede tener las siguientes desventajas: Por un lado, para unos paquetes de datos pequeños, el número de bits para la información conexa que se requiere para señalizar la subdivisión en particiones se puede volver significativa en relación con los datos reales en los trenes de bits parciales, lo que en última instancia reduce la eficiencia de codificación. Por otro lado, la multiplexación puede

no ser adecuada para las aplicaciones que requieren un bajo retardo (por ejemplo, para aplicaciones de videoconferencia). Con la multiplexación descrita, el codificador no puede iniciar la transmisión de un paquete de datos antes de que los trenes de bits parciales se hayan creado completamente, debido a que las ubicaciones del comienzo de las particiones no se conocían con anterioridad. Además, en general, el descodificador ha de esperar hasta que este recibe el comienzo del último segmento de datos antes de que este pueda iniciar la descodificación de un paquete de datos. Para aplicaciones como sistemas de videoconferencia, estos retardos pueden sumarse a un retardo global adicional del sistema de varias imágenes de vídeo (en particular para unas tasas de bits que se encuentran cerca de la tasa de bits de transmisión y para codificadores / descodificadores que requieren casi el intervalo de tiempo entre dos imágenes para codificar / descodificar una imagen), que es crítico para tales aplicaciones. Con el fin de superar las desventajas para determinadas aplicaciones, el codificador o puede configurarse de una forma tal que las palabras de código que se generan mediante los dos o más codificadores de contenedores y el codificador de VLC se intercalan para dar un único tren de bits. El tren de bits con las palabras de código intercaladas puede enviarse directamente al descodificador (cuando se desprecia un pequeño retardo de memoria intermedia, véase en lo sucesivo). En el lado de descodificador, los dos o más descodificadores de contenedores y el descodificador de VLC leen las palabras de código directamente a partir del tren de bits en el orden de descodificación; la descodificación se puede iniciar con el primer bit recibido. Además, no se requiere información conexa alguna para señalizar la multiplexación (o intercalación) de los trenes de bits parciales.

La estructura básica de un codificador con una intercalación de palabras de código se muestra en la figura 22. Los codificadores de contenedores 10 y el codificador de VLC 10a no escriben las palabras de código directamente en los trenes de bits parciales, sino que se conectan con una única memoria intermedia de palabras de código 29, a partir de las cual se escriben palabras de código en el tren de bits 34 en el orden de codificación. Los codificadores de contenedores 10 envían solicitudes para una o más nuevas entradas de memoria intermedia de palabras de código 28 a la memoria intermedia de palabras de código 29 y posteriormente envían las palabras de código 30 a la memoria intermedia de palabras de código 29, que se almacenan en las entradas de memoria intermedia reservadas. El codificador de VLC 10a escribe directamente las palabras de código de VLC 30a en la memoria intermedia de palabras de código 29. Las palabras de código (en general de longitud variable) 31 de la memoria intermedia de palabras de código 29 se acceden por una unidad de escritura de palabras de código 32, que escribe los bits 33 correspondientes en el tren de bits producido 34. La memoria intermedia de palabras de código 29 opera como una memoria intermedia de tipo primero en entrar primero en salir; las entradas de palabra de código que se reservan con anterioridad se escriben con anterioridad en el tren de bits.

La memoria intermedia de palabras de código se puede operar tal como sigue. Si un nuevo contenedor 7 se envía a una memoria intermedia de contenedores particular 8 y el número de contenedores ya almacenados en la memoria intermedia de contenedores es cero y en la actualidad no hay palabra de código alguna reservada en la memoria intermedia de palabras de código para el codificador de contenedores que está conectado con la memoria intermedia de contenedores particular, el codificador de contenedores conectado 10 envía una solicitud a la memoria intermedia de palabras de código, mediante la cual una o más entradas de palabra de código se reservan en la memoria intermedia de palabras de código 29 para el codificador de contenedores particular 10. Las entradas de palabra de código pueden tener un número variable de bits; un umbral superior para el número de bits en una entrada de memoria intermedia viene dado por lo general por el tamaño de palabra de código máximo para el codificador de contenedores correspondiente. La siguiente palabra de código o las siguientes palabras de código que se producen por el codificador de contenedores (para el que se han reservado la entrada de palabra de código o entradas de palabra de código) se almacenan en la entrada o entradas reservadas de la memoria intermedia de palabras de código. Si todas las entradas de memoria intermedia reservadas en la memoria intermedia de palabras de código para un codificador de contenedores particular se llenan con palabras de código y el siguiente contenedor se envía a la memoria intermedia de contenedores que está conectada con el codificador de contenedores particular, una o más nuevas palabras de código se reservan en la memoria intermedia de palabras de código para el codificador de contenedores particular, etc. El codificador de VLC 10a escribe directamente las palabras de código de VLC 30a en la siguiente entrada libre de la memoria intermedia de palabras de código 29, es decir, para el codificador de VLC, la reserva de palabras de código y la escritura de la palabra de código se realizan al mismo tiempo. La memoria intermedia de palabras de código 29 representa una memoria intermedia de tipo primero en entrar primero en salir de una cierta forma. Las entradas de memoria intermedia se reservan en orden secuencial. Las palabras de código para las que las entradas de memoria intermedia correspondientes se han reservado con anterioridad se escriben con anterioridad en el tren de bits. La unidad de escritura de palabras de código 32 comprueba el estatus de la memoria intermedia de palabras de código 29, o bien de forma continua o bien después de que una palabra de código 30 se escriba en la memoria intermedia de palabras de código 29. Si la primera entrada de memoria intermedia contiene una palabra de código completa (es decir, la entrada de memoria intermedia no está reservada, sino que incluye una palabra de código), la palabra de código 31 correspondiente y la entrada de memoria intermedia correspondiente se retiran de la memoria intermedia de palabras de código 20 y los bits de la palabra de código 33 se escriben en el tren de bits. Este proceso se repite hasta que la primera entrada de memoria intermedia no contiene una palabra de código (es decir, esta está reservada o es libre). Al final del proceso de descodificación, es decir, si se han procesado todos los símbolos de origen de la cantidad considerada de datos, se ha de evacuar la memoria intermedia de palabras de código. Para ese proceso de evacuación, lo sucesivo es de

aplicación para cada memoria intermedia de contenedores / codificador de contenedores como una primera etapa: Si la memoria intermedia de contenedores contiene, de hecho, unos contenedores, se añade un contenedor con un valor particular o uno arbitrario hasta que la secuencia de contenedores resultante representa una secuencia de contenedores que se asocia con una palabra de código (tal como se ha hecho notar en lo que antecede, una forma preferida de adición de contenedores es añadir aquellos valores de contenedor que producen la palabra de código posible más corta - o una de ellas - que se asocia con una secuencia de contenedores que contiene el para contenido original de la memoria intermedia de contenedores como prefijo), entonces la palabra de código se escribe en la siguiente entrada de memoria intermedia reservada para el codificador de contenedores (y la correspondiente) memoria intermedia de contenedores se vacía. Si se ha reservado más de una entrada de memoria intermedia para uno o más codificadores de contenedores, la memoria intermedia de palabras de código puede seguir conteniendo entradas de palabra de código reservadas. En ese caso, estas entradas de palabra de código se llenan con unas palabras de código arbitrarias pero válidas para los codificadores de contenedores correspondientes. Se puede insertar la palabra de código válida más corta o una de las palabras de código válidas más cortas (de haber una multitud). El codificador de VLC no requiere terminación alguna. Por último, todas las palabras de código restantes en la memoria intermedia de palabras de código se escriben en el tren de bits.

En la figura 23 se ilustran dos ejemplos para el estatus de la memoria intermedia de palabras de código. En el ejemplo (a), la memoria intermedia de palabras de código contiene 4 entradas que se llenan con una palabra de código (dos de estas son entradas de VLC) y 3 entradas reservadas. Además, se marca la siguiente entrada de memoria intermedia libre. La primera entrada se llena con una palabra de código (es decir, el codificador de contenedores 2 acababa de escribir una palabra de código en una entrada previamente reservada). En la siguiente etapa, esta palabra de código se retirará de la memoria intermedia de palabras de código y se escribirá en el tren de bits. Entonces, la primera palabra de código reservada para el codificador de contenedores 3 es la primera entrada de memoria intermedia, pero esta entrada no se puede retirar de la memoria intermedia de palabras de código, debido a que esta solo está reservada, pero no se ha escrito palabra de código alguna en esta entrada. En el ejemplo (b), la memoria intermedia de palabras de código contiene 4 entradas que se llenan con una palabra de código (una de estas es una entrada de memoria intermedia de VLC) y 4 entradas reservadas. La primera entrada se marca como reservada y, por lo tanto, la unidad de escritura de palabras de código no puede escribir una palabra de código en el tren de bits. A pesar de que 4 palabras de código están contenidas en la memoria intermedia de palabras de código, la unidad de escritura de palabras de código ha de esperar hasta que se escriba una palabra de código en la primera entrada de memoria intermedia reservada para el codificador de contenedores 3. Obsérvese que las palabras de código deben escribirse en el orden en el cual se reservaron, con el fin de ser capaces de invertir el proceso en el lado de descodificador (véase en lo sucesivo). Y obsérvese adicionalmente que una entrada de memoria intermedia de VLC siempre está completada, debido a que la reserva y la escritura de la palabra de código se realizan al mismo tiempo.

La estructura básica de un descodificador con una intercalación de palabras de código se muestra en la figura 24. Los descodificadores de contenedores 22 y el descodificador de VLC 2a no leen las palabras de código directamente a partir de trenes de bits parciales separados, sino que están conectados con una memoria intermedia de bits 38, a partir de la que las palabras de código 37, 37a se leen en el orden de codificación. Se debería hacer notar que la memoria intermedia de bits 38 no se requiere necesariamente, debido a que las palabras de código también se podrían leer directamente a partir del tren de bits. La memoria intermedia de bits 38 se incluye principalmente en la ilustración para separar claramente diferentes aspectos de la cadena de procesamiento. Los bits 39 del tren de bits 40 con palabras de código intercaladas se insertan de forma secuencial en la memoria intermedia de bits 38, la cual representa una memoria intermedia de tipo primero en entrar primero en salir. Si un descodificador de contenedores particular 22 recibe una solicitud para una o más secuencias de contenedores 35, el descodificador de contenedores 22 lee una o más palabras de código 37 a partir de la memoria intermedia de bits 38 por medio de solicitudes para bits 36. El descodificador puede descodificar de forma instantánea los símbolos de origen. De forma similar, si el descodificador de VLC 22a recibe una solicitud para un nuevo símbolo 19a, lee la palabra de código de VLC 37a correspondiente a partir de la memoria intermedia de bits 38 y devuelve el símbolo descodificado 27a. Obsérvese que el codificador (tal como se ha descrito en lo que antecede) ha de asegurar, al operar de forma conveniente la memoria intermedia de palabras de código, que las palabras de código se escriban en el mismo orden en el tren de bits en el que estas son solicitadas por los descodificadores de contenedores. En el descodificador, la totalidad del proceso de descodificación es desencadenado por solicitudes para símbolos de origen. Parámetros como el número de palabras de código que son reservados en el lado de codificador por un codificador de contenedores particular y el número de palabras de código que se leen por el descodificador de contenedores correspondiente han de ser los mismos.

Intercalación de palabras de código de longitud variable con una restricción de retardo baja

La intercalación de palabras de código descrita no requiere que se envíe información alguna de subdivisión en particiones como información conexa. Y debido a que las palabras de código se intercalan en el tren de bits, el retardo es en general pequeño. No obstante, no se garantiza que se obedezca una restricción de retardo particular (por ejemplo, especificada por un número máximo de bits que se almacenan en la memoria intermedia de palabras

de código). Además, el tamaño de memoria intermedia requerido para la memoria intermedia de palabras de código se puede volver, en teoría, muy grande. Cuando se considera el ejemplo en la figura 23(b), podría ser posible que no se enviara contenedor adicional alguno a la memoria intermedia de contenedores 3 y, por lo tanto, el codificador de contenedores 3 no enviará palabra de código nueva alguna a la memoria intermedia de palabras de código hasta que se aplique el proceso de evacuación al final del paquete de datos. Entonces todas las palabras de código para los codificadores de contenedores 1 y 2 tendrían que esperar hasta el final del paquete de datos, antes de que estas se puedan escribir en el tren de bits. Este inconveniente puede sortearse mediante la adición de un mecanismo adicional al proceso de codificación (y también al proceso de descodificación tal como se describe posteriormente). El concepto básico de ese mecanismo adicional es que si una medida en relación con el retardo o un límite superior del retardo (véase en lo sucesivo) supera un umbral especificado, la primera entrada de memoria intermedia reservada se llena mediante la evacuación de la memoria intermedia de contenedores correspondiente (usando un mecanismo similar al del final de un paquete de datos). Mediante un mecanismo de este tipo, el número de entradas de memoria intermedia en espera se reduce hasta que la medida de retardo asociada es menor que el umbral especificado. En el lado de descodificador, se han de descartar los contenedores que se han insertado en el lado de codificador con el fin de obedecer la restricción de retardo. Para este descarte de contenedores se puede usar básicamente el mismo mecanismo que en el lado de codificador.

Después de haber descrito con detalle posibilidades para intercalar los trenes de bits de VLC y la codificación de PIPE, en lo sucesivo, la descripción de nuevo se centra en los elementos de sintaxis ya mencionados en lo que antecede descompuestos en símbolos de origen tal como se menciona con respecto a la figura 1b, 1c y 2b. Por razones de ilustración, la siguiente descripción supone que los elementos de sintaxis así descompuestos son un nivel de coeficiente de transformada absoluto. No obstante, esto es solo un ejemplo, y otros tipos de elementos de sintaxis se pueden manipular de forma similar. *En particular*, *en lo sucesivo se describe la codificación de niveles absolutos mediante la subdivisión en particiones y el uso de diferentes códigos por entropía en codificadores de vídeo y de imágenes basados en bloques*.

Por ejemplo, las imágenes de una secuencia de vídeo se descomponen por lo general en bloques. Los bloques o las componentes de color de los bloques se predicen mediante o bien predicción compensada por movimiento o bien intrapredicción. Los bloques pueden tener diferentes tamaños y pueden ser o bien cuadrados o bien rectangulares. Todas las muestras de un bloque o una componente de color de un bloque se predicen usando el mismo conjunto de parámetros de predicción, tal como índices de referencia (que identifican una imagen de referencia en el conjunto de imágenes ya codificado), parámetros de movimiento (que especifican una medida para el movimiento de un bloque entre una imagen de referencia y la imagen actual), parámetros para especificar el filtro de interpolación, modos de intrapredicción, etc. Los parámetros de movimiento se pueden representar por unos vectores de desplazamiento con una componente horizontal y vertical o por unos parámetros de movimiento de orden superior tales como unos parámetros de movimiento afín que consisten en 6 componentes. También es posible que más de un conjunto de parámetros de predicción (tal como índices de referencia y parámetros de movimiento) estén asociados con un único bloque. En ese caso, para cada conjunto de parámetros de predicción, se genera una única señal de predicción intermedia para el bloque o la componente de color de un bloque, y la señal de predicción final se construye mediante una suma ponderada de las señales de predicción intermedias. Los parámetros de ponderación y potencialmente también una desviación constante (que se añade a la suma ponderada) pueden o bien ser fijos para una imagen, o una imagen de referencia o un conjunto de imágenes de referencia, o bien estos se pueden incluir en el conjunto de parámetros de predicción para el bloque correspondiente. De forma similar, las imágenes fijas también se descomponen a menudo en bloques, y los bloques se predicen mediante un método de intrapredicción (que puede ser un método de intrapredicción espacial o un método de intrapredicción simple que predice la componente de CC del bloque). En un caso límite, la señal de predicción también puede ser cero.

La diferencia entre los bloques originales o las componentes de color de los bloques originales y las señales de predicción correspondientes, a la que también se hace referencia como señal residual, por lo general se transforma y se cuantifica. Una transformada bidimensional se aplica a la señal residual y los coeficientes de transformada resultantes se cuantifican. Para esta codificación de transformada, los bloques o las componentes de color de los bloques, para los cuales se ha usado un conjunto particular de parámetros de predicción, se pueden dividir adicionalmente antes de la aplicación de la transformada. Los bloques de transformada pueden ser iguales a o más pequeños que los bloques que se usan para la predicción. También es posible que un bloque de transformada incluya más de uno de los bloques que se usan para la predicción. Diferentes bloques de transformada en una imagen fija o una imagen de una secuencia de vídeo pueden tener diferentes tamaños y los bloques de transformada pueden representar unos bloques cuadrados o rectangulares.

La totalidad de estos parámetros de predicción y residuales pueden formar el tren de los elementos de sintaxis 138 y 226, de forma respectiva.

Los coeficientes de transformada cuantificados resultantes, a los que también se hace referencia como niveles de coeficiente de transformada, se pueden transmitir entonces usando una codificación por entropía mediante cualquiera de los esquemas de codificación anteriores. Con este fin, un bloque de niveles de coeficientes de

transformada se puede poner en correspondencia con un vector (es decir, un conjunto ordenado) de valores de coeficiente de transformada usando una exploración, en donde se pueden usar diferentes exploraciones para diferentes bloques. A menudo se usa una exploración en zig-zag. Para los bloques que contienen solo muestras de un campo de una trama intercalada (estos bloques pueden ser bloques en campos codificados o bloques de campos en tramas codificadas), también es común el uso de una exploración diferente que está diseñada de forma específica para los bloques de campos. Un posible esquema de codificación para codificar la secuencia ordenada resultante de coeficientes de transformada es la codificación de nivel de series. Por lo general, un gran número de los niveles de coeficiente de transformada son cero, y un conjunto de niveles de coeficiente de transformada sucesivos que son iguales a cero se puede representar de forma eficiente mediante la codificación del número de niveles de coeficiente de transformada sucesivos que son iguales a cero (la serie) por un elemento de sintaxis respectivo. Para los coeficientes de transformada (no nulos) restantes, el nivel real se codifica en la forma de unos elementos de sintaxis respectivos. Hay diversas alternativas de códigos de nivel de series. La serie antes de un coeficiente no nulo y el nivel del coeficiente de transformada no nulo se pueden codificar de forma conjunta usando un único elemento de sintaxis. A menudo, se incluyen elementos de sintaxis especiales para el fin de bloque, el cual se envía después del último coeficiente de transformada no nulo. O es posible codificar en primer lugar el número de niveles de coeficiente de transformada no nulo, y dependiendo de este número, se codifican los niveles y las series.

Un enfoque algo diferente se usa en la muy eficiente codificación por entropía de CABAC en la norma H.264 / AVC. En el presente caso, la codificación de los niveles de coeficiente de transformada se divide en tres etapas. En la primera etapa, un elemento de sintaxis binario encoded_block_flag se transmite para cada bloque de transformada, el cual señaliza si el bloque de transformada contiene niveles de coeficiente de transformada significativos (es decir, coeficientes de transformada que son no nulos). Si este elemento de sintaxis indica que se encuentran presentes unos niveles de coeficiente de transformada significativos, se codifica una correspondencia de significación de valores binarios, la cual especifica cuál de los niveles de coeficiente de transformada tiene unos valores no nulos. Y entonces, en un orden inverso al de exploración, se codifican los valores de los niveles de coeficiente de transformada no nulo. La correspondencia de significación se codifica para dar el tren de elementos de sintaxis 138 tal como sigue. Para cada coeficiente en el orden de exploración, se codifica un elemento de sintaxis binario significant_coeff_flag, el cual especifica si el nivel de coeficiente de transformada correspondiente no es igual a cero. Si el contenedor significant_coeff_flag es igual a uno, es decir, si existe un nivel de coeficiente de transformada no nulo en la posición de exploración, se codifica un elemento de sintaxis binario adicional last_significant_coeff_flag. Este contenedor indica si el nivel de coeficiente de transformada significativo actual es el último nivel de coeficiente de transformada significativo en el interior de un bloque o si niveles de coeficiente de transformada significativos adicionales siguen en el orden de exploración. Si last_significant_coeff_flag indica que no sigue coeficiente de transformada significativo adicional alguno, no se codifica elemento de sintaxis adicional alguno para especificar la correspondencia de significación para el bloque. En la siguiente etapa, se codifican los valores de los niveles de coeficiente de transformada significativos, cuyas ubicaciones en el interior de un bloque ya están determinadas por la correspondencia de significación. Los valores de los niveles de coeficiente de transformada significativos se codifican en el orden inverso al de exploración mediante el uso de los siguientes tres elementos de sintaxis. El elemento de sintaxis binario coeff_abs_greater_one indica si el valor absoluto del nivel de coeficiente de transformada significativo es mayor que uno. Si el elemento de sintaxis binario coeff_abs_greater_one indica que el valor absoluto es mayor que uno, se envía un elemento de sintaxis adicional coeff_abs_level_minus_two, el cual especifica el valor absoluto del nivel de coeficiente de transformada menos dos. Este es el tipo de elemento de sintaxis el procesamiento el cual se puede realizar de acuerdo con la figura 1b, 1c y 2b. Por último, el elemento de sintaxis binario coeff_sign_flag, el cual especifica el signo del valor de coeficiente de transformada, se codifica para cada nivel de coeficiente de transformada significativo. Se debería hacer notar de nuevo que los elementos de sintaxis que se refieren a la correspondencia de significación se codifican en el orden de exploración, mientras que los elementos de sintaxis que se refieren a los valores reales de los niveles de coeficientes de transformada se codifican en el orden inverso al de exploración permitiendo el uso de más modelos de contexto adecuados. También es posible que se use un patrón de exploración adaptativo para la correspondencia de significación como en el primer modelo de prueba de la norma H.265 / HEVC. Otro concepto se usa para la codificación de los niveles de coeficiente de transformada absolutos para unos bloques de transformada más grandes que 4 x 4 en el primer modelo de prueba de la norma H.265 / HEVC. En el caso de unos bloques de transformada más grandes que 4 x 4, el bloque de transformada más grande se subdivide en particiones para dar bloques de 4 x 4 y los bloques de 4 x 4 se codifican en el orden de exploración mientras que para cada bloque de 4 x 4, se usa el orden inverso al de exploración.

En la codificación por entropía de CABAC en la norma H.264 / AVC, todos los elementos de sintaxis para los niveles de coeficiente de transformada se codifican usando un modelado de probabilidad binaria. El elemento de sintaxis no binario coeff_abs_level_minus_two, por ejemplo, se binariza en primer lugar, es decir, se pone en correspondencia con una secuencia de decisiones binarias (contenedores), y estos contenedores se codifican de forma secuencial. Los elementos de sintaxis binarios significant_coeff_flag, last_significant_coeff_flag, coeff_abs_greater_one, y coeff_sign_flag se codifican directamente. Cada contenedor codificado (incluyendo los elementos de sintaxis binarios) se asocia con un contexto. Un contexto representa un modelo de probabilidad para una clase de contenedores codificados. Una medida en relación con la probabilidad para uno de los dos valores de contenedor

posibles se estima para cada contexto sobre la base de los valores de los contenedores que ya se han codificado con el contexto correspondiente. Para varios contenedores en relación con la codificación de transformada, el contexto que se usa para la codificación se selecciona sobre la base de elementos de sintaxis ya transmitidos o sobre la base de la posición en el interior de un bloque.

Después de la codificación de la correspondencia de significación, el bloque se procesa en el orden inverso al de exploración. Tal como se ha mencionado antes, otro concepto se usa en el primer modelo de prueba de la norma H.265 / HEVC. Los bloques de transformada más grandes que 4 x 4 se subdividen en particiones para dar bloques de 4 x 4 y los bloques de 4 x 4 resultantes se procesan en el orden de exploración, mientras que los coeficientes de los bloques de 4 x 4 se codifican en el orden inverso al de exploración. La siguiente descripción es válida para todos los bloques de 4 x 4 en el primer modelo de prueba de la norma H.265 / HEVC y H.264 / AVC y también para los bloques de 8 x 8 en la norma H.264 / AVC y esta descripción también se puede aplicar a la construcción del tren de los elementos de sintaxis 138 y 226, de forma respectiva.

Si una posición de exploración es significativa, es decir, el coeficiente es diferente de cero, el elemento de sintaxis binario coeff_abs_greater_one se transmite dentro del tren 138. Inicialmente (en el interior de un bloque), el segundo modelo de contexto del conjunto de modelo de contexto correspondiente se selecciona para el elemento de sintaxis coeff_abs_greater_one. Si el valor codificado de cualquier elemento de sintaxis coeff_abs_greater_one en el interior de un bloque es igual a uno (es decir, el coeficiente absoluto es mayor que 2), el modelado de contexto conmuta de vuelta al primer modelo de contexto del conjunto y usa este modelo de contexto hasta el final del bloque. De lo contrario (todos los valores codificados de coeff_abs_greater_one en el interior de un bloque son cero y los niveles de coeficiente absolutos correspondientes son iguales a uno), el modelo de contexto se elige dependiendo del número de los elementos de sintaxis coeff_abs_greater_one igual a cero que ya se han procesado en la exploración inversa del bloque considerado. La selección de modelo de contexto para el elemento de sintaxis coeff_abs_greater_one se puede resumir por la siguiente ecuación, en la que el índice de modelo de contexto actual C_{t+1} se selecciona sobre la base del índice de modelo de contexto previo C_t y el valor del elemento de sintaxis previamente codificado coeff_abs_greater_one, el cual se representa por contenedor$_t$ en la ecuación. Para el primer elemento de sintaxis coeff_abs_greater_one en el interior de un bloque, el índice de modelo de contexto se establece igual a $C_1 = 1$.

$$C_{t+1}(C_t, bin_t) = \begin{cases} 0 & bin_t = 1 \\ \min(C_t + 1, 4) & bin_t = 0 \end{cases}$$

El segundo elemento de sintaxis para codificar los niveles de coeficiente de transformada absolutos, coeff_abs_level_minus_two solo se codifica cuando el elemento de sintaxis coeff_abs_greater_one para la misma posición de exploración es igual a uno. El elemento de sintaxis no binario coeff_abs_level_minus_two se binariza en una secuencia de contenedores y para el primer contenedor de esta binarización; un índice de modelo de contexto se selecciona tal como se describe en lo sucesivo. Los contenedores restantes de la binarización se codifican con unos contextos fijos. El contexto para el primer contenedor de la binarización se selecciona tal como sigue. Para el primer elemento de sintaxis coeff_abs_level_minus_two, se selecciona el primer modelo de contexto del conjunto de modelos de contexto para el primer contenedor del elemento de sintaxis coeff_abs_level_minus_two, el índice de modelo de contexto correspondiente se establece igual a $C_1 = 0$. Para cada primer contenedor adicional del elemento de sintaxis coeff_abs_level_minus_two, el modelado de contexto conmuta al siguiente modelo de contexto en el conjunto, en donde el número de modelos de contexto en el conjunto está limitado a 5. La selección de modelo de contexto puede expresarse por la siguiente fórmula, en la que el índice de modelo de contexto actual C_{t+1} se selecciona sobre la base del índice de modelo de contexto previo C_t.

$$C_{t+1}\,(C_t) = \text{mín}\,(C_t+1,\, 4)$$

Tal como se ha mencionado antes, para el primer elemento de sintaxis coeff_abs_remain_minus_two en el interior de un bloque, el índice de modelo de contexto se establece igual a $C_t = 0$. Obsérvese que se pueden definir diferentes conjuntos de modelos de contexto para los elementos de sintaxis coeff_abs_greater_one y coeff_abs_remain_minus_two. También obsérvese que para el primer modelo de prueba de la norma H.265 / HEVC, los bloques de transformada más grandes que 4 x 4 se pueden subdividir en particiones para dar bloques de 4 x 4. Los bloques de 4 x 4 subdivididos en particiones se pueden procesar en el orden de exploración y para cada bloque de 4 x 4 subdividido en particiones, un conjunto de contexto se puede obtener sobre la base del número de coeficientes mayor que uno en el bloque de 4 x 4 previo. Para el primer bloque de 4 x 4 de un bloque de transformada más grande que 4 x 4 y para los bloques de transformada de 4 x 4 de origen se pueden usar un conjunto de contexto separado.

Es decir, siempre que en la siguiente descripción se use una codificación basada en contexto para cualquiera de los símbolos de origen en la que coeff_abs_greater_one y coeff_abs_remain_minus_two puede descomponerse tal como sigue, entonces esta obtención de contexto puede ser usada por la unidad de asignación 114 y 212 y el

codificador / descodificador de VLC 102 y 202, por ejemplo.

Con el fin de reducir la complejidad en términos del número de contenedores que son procesados por CABAC o PIPE en comparación con las técnicas del estado de la técnica y también en términos de la complejidad de computación o también para incrementar la eficiencia de codificación, la explicación posterior describe un enfoque para codificar niveles absolutos mediante el uso de diferentes códigos de longitud variable para diferentes particiones 140_1 a 140_3 en codificadores y descodificadores de imagen y de vídeo. La posibilidad que se bosqueja en lo sucesivo se puede aplicar, no obstante, a todo tipo de niveles absolutos para codificadores de vídeo y de imagen, como diferencias de vector de movimiento o coeficientes del filtro de lazo adaptativo. Mientras que la codificación de niveles de coeficientes de transformada se realiza tal como se bosqueja en lo sucesivo, la codificación de la correspondencia de significación puede permanecer como en el primer modelo de prueba de la norma H.265 / HEVC o tal como se ha descrito en lo que antecede, o la codificación de la correspondencia de significación también se puede realizar como en la norma H.264 / AVC o de otro modo.

Tal como se ha descrito en lo que antecede, la codificación de los niveles de transformada absolutos se realiza en múltiples particiones 140_{1-3}. El esquema de codificación se ha ilustrado a modo de ejemplo en la figura 1b con tres particiones 140_{1-3}. Los límites 142 y 144 del esquema son variables, lo que da como resultado unos tamaños de partición variables. El esquema de codificación se realiza tal como sigue.
Un primer código por entropía se usa para codificar la primera componente o símbolo de origen, es decir un nivel de coeficiente de transformada absoluto (z) en el caso de que el mismo sea más pequeño que límite1, o límite1 en el caso de que el mismo no lo sea. Si el nivel de coeficiente de transformada absoluto es mayor que o igual al límite límite1 de la primera partición 140_1, entonces el límite límite1 (142) de la primera partición (140_1) se resta del nivel de coeficiente de transformada absoluto y el valor resultante z' se codifica con un segundo código por entropía. Si el nivel de coeficiente de transformada absoluto restante z' es mayor que o igual a un límite límite2 - límite1 para la segunda partición 140_2, entonces el límite límite2 - límite1 de la segunda partición se resta de nuevo del nivel de coeficiente de transformada absoluto z' y el valor resultante se codifica con un tercer código por entropía. Hablando en términos generales, cuando se alcanza el límite de una partición, el código por entropía para la siguiente partición hasta el límite se usa para codificar el valor resultante de los niveles de coeficiente de transformada absolutos menos el límite de la partición correspondiente.

Los códigos por entropía pueden ser códigos de longitud variable simples como códigos de longitud de series o tablas de consulta (por ejemplo, el código de Huffman) o códigos por entropía más complejos que emplean modelos de probabilidad como CABAC o PIPE. El número de particiones y el límite de las particiones pueden ser variables o dependientes del elemento de sintaxis real. El concepto de subdivisión en particiones que se muestra en la figura 1b tiene los siguientes beneficios. En lo sucesivo, se usan los coeficientes de transformada absolutos como un ejemplo, pero se debería entender que estos se pueden sustituir con cualquier otro elemento de sintaxis. Como un ejemplo, la distribución de probabilidad de los niveles de coeficiente de transformada absolutos puede ser aproximadamente una distribución geométrica. Por lo tanto, los códigos por entropía optimizados para distribuciones geométricas se pueden emplear para codificar los niveles de coeficiente de transformada absolutos. Pero un modelo de este tipo no es siempre localmente óptimo, incluso si se emplean modelado de contexto y selección de modelos de probabilidad. Por ejemplo, para un bloque de transformada, los niveles de coeficiente de transformada absolutos locales siguen una distribución la cual no es en absoluto geométrica si esta contiene la misma cantidad de niveles de coeficiente de transformada de rango muy bajo y medio, mientras que el modelo puede ser cierto (con una cierta precisión) para una cantidad específica de bloques en imágenes o vídeo debido a la ley de los grandes números. Para un caso de este tipo, la distribución geométrica no es un modelo adecuado. Asimismo, cuando se consideran unos niveles de coeficiente de transformada absolutos con valores grandes, la distribución de estos valores grandes es a menudo uniforme. El concepto de subdivisión en particiones permite diferentes modelos de probabilidad para diferentes niveles de coeficiente de transformada absolutos. Para unos valores absolutos más bajos, se pueden aplicar códigos por entropía más complejos para una eficiencia más alta, mientras que para niveles absolutos grandes, se pueden emplear códigos por entropía menos complejos para reducir la complejidad.

Tal como se ha mencionado antes, se usan códigos por entropía adecuados para las diferentes particiones. Se pueden emplear tres tipos de códigos por entropía. El primer código por entropía usa PIPE. No obstante, se debería hacer notar que, de acuerdo con una alternativa, un método de codificación por entropía como CABAC o, como alternativa, se puede usar cualquier otro codificador aritmético. Es decir, los primeros símbolos s_1 (véase la figura 2b) se pueden codificar por medio de la trayectoria de codificación de PIPE. El subdivisor 100 y el recombinador 220 actúan en consecuencia.

Para el segundo tipo, se pueden usar códigos de Golomb y algunas variantes truncadas incluyendo subconjuntos (por ejemplo, códigos de Golomb-Rice). Es decir, los segundos símbolos s_2 (véase la figura 2b) se pueden codificar por medio de tales códigos de VLC en el codificador / descodificador de VLC 102 / 202. El subdivisor 100 y el recombinador 220 actúan en consecuencia.

Se usan códigos de Golomb Exponencial como el tercer tipo. Es decir, los terceros símbolos s_3 (véase la figura 2b)

se pueden codificar por medio de tales códigos de VLC en el codificador / descodificador de VLC 102 / 202. El subdivisor 100 y el recombinador 220 actúan en consecuencia. Son posibles diferentes códigos de VLC y diferentes combinaciones de códigos de VLC y de PIPE o de VLC y aritméticos.

Mientras que el primer código es más complejo pero da un mejor rendimiento de compresión, los segundos códigos por entropía representan una compensación recíproca razonable entre la complejidad y el rendimiento. Los últimos códigos por entropía, por ejemplo los códigos de Golomb Exponencial, son de una complejidad muy baja. En lo sucesivo, se describe la codificación de las diferentes particiones.

Si una partición tal como la partición 140_1, tal como el símbolo s_1, se va a codificar por entropía usando un codificador por entropía empleando modelos de probabilidad como el codificador de PIPE 104 (de nuevo, se puede usar CABAC o cualquier otro codificador aritmético en una alternativa que no se describe adicionalmente en el presente caso), el subdivisor 120 dirige la misma hacia el codificador de PIPE 104. En primer lugar, los niveles de coeficiente de transformada absolutos de valor no binario se pueden binarizar en el simbolizador 122 usando un método de binarización. La binarización pone en correspondencia los niveles de coeficiente de transformada absolutos de valor no binario con una secuencia de contenedores binarios. Cada contenedor de la cadena de contenedores se codifica con un contexto elegido por la unidad de asignación 114. El modelado de contexto se puede realizar para el primer contenedor y ser fijo para los siguientes contenedores de la secuencia de contenedores como coeff_abs_level_minus_two en la norma H.264 / AVC o se puede usar un modelado de contexto diferente para cada contenedor de la cadena de contenedores. Obsérvese que la binarización puede ser un código de longitud variable como los códigos de Golomb o los códigos de Golomb Exponencial u otros códigos de longitud variable.

A continuación, una partición tal como la partición 140_2 o los símbolos s_2 se pueden codificar con un código de Golomb, en el presente caso en el codificador de VLC 102 y se pueden descodificar de forma respectiva en el descodificador de VLC. Los códigos de Golomb son un conjunto de códigos por entropía que están diseñados para un origen distribuido geométrico. Si el orden del código de Golomb es cero, el código de Golomb también se conoce como código unario. El código unario se refiere a la binarización de coeff_abs_level_minus_two en la norma H.264 / AVC. Los códigos de Golomb se construyen tal como sigue. Para un parámetro de Golomb *k* específico, el valor *n* se divide por el parámetro de Golomb *k* usando división de números enteros y se calcula el resto *r*.

$$p = \left\lfloor \frac{n}{k} \right\rfloor$$

$$r = n - pk$$

Después de obtener los parámetros especificados por las fórmulas en lo que antecede, el valor n se puede codificar con dos partes. La primera parte, a la que también se hace referencia como parte de prefijo, es un código unario. El valor resultante *p + 1* especifica el número de unos y un cero de terminación o viceversa. El valor del resto, al que también se hace referencia como parte de resto y se indica por r, se representa con un código binario truncado. Los códigos de Golomb-Rice se emplean para codificar los símbolos de origen tales como los símbolos de origen s_2, siendo los códigos de Golomb-Rice un subconjunto de los códigos de Golomb. Asimismo, cuando se usan tales códigos por entropía para las particiones $140_{1\text{-}3}$ que contienen un límite, tal como la partición 140_2 el alfabeto de los símbolos de origen respectivos (tales como los símbolos de origen s_2) está limitado y el código de Golomb-Rice se puede modificar de tal modo que se puede mejorar la eficiencia de codificación. El parámetro del código de Golomb-Rice puede ser fijo o variable. Si el parámetro es variable, el parámetro se puede estimar como una parte de la fase de modelado de contexto. Por ejemplo, si un símbolo de origen s_2 entra en el codificador de VLC 102, este último puede determinar el parámetro del código de Golomb-Rice a partir de un contexto de s_2. Los códigos de Golomb-Rice son códigos de Golomb con unos parámetros elevados a la potencia de dos. Por lo tanto, estos se basan en la división y multiplicación por dos y, por lo tanto, estos se pueden implementar de forma eficiente en una arquitectura binaria con unas operaciones de desplazamiento y de adición. La relación entre el parámetro de Golomb-Rice y el parámetro de Golomb es, por lo tanto, $k_{GOLOMB} = 2^{kRICE}$. En el caso del código de Golomb-Rice, la parte de resto es exactamente la representación binaria del valor del resto. Para el parámetro de Golomb-Rice de cero, el código resultante es idéntico al código unario y no tiene una parte de resto. Para el parámetro igual a uno, la parte de resto consiste en un contenedor con dos símbolos de entrada que comparten el mismo prefijo unario. En lo sucesivo, se ilustran algunas tablas a modo de ejemplo para unos parámetros de Golomb-Rice seleccionados.

Valor	**Prefijo**	**Resto**	**Prefijo**	**Resto**	**Prefijo**	**Resto**	**Prefijo**	**Resto**
	k = 0		k = 1		k = 2		k = 3	
0	0		0	0	0	00	0	000

1	10		0	1	0	01	0	001
2	110		10	0	0	10	0	010
3	1110		10	1	0	11	0	011
4	11110		110	0	10	00	0	100
5	111110		110	1	10	01	0	101
6	1111110		1110	0	10	10	0	110
7	11111110		1110	1	10	11	0	111
8	111111110		11110	0	110	00	10	000
9	1111111110		11110	1	110	01	10	001
10	11111111110		111110	0	110	10	10	010
11	111111111110		111110	1	110	11	10	011
12	1111111111110		1111110	0	1110	00	10	100
13	11111111111110		1111110	1	1110	01	10	101

Dado el rango de la partición tal como por ejemplo límite2 - límite1 en el caso de la partición 140_2 y el parámetro del código de Golomb-Rice, el truncamiento se puede realizar tal como sigue. El parámetro de Golomb-Rice describe el número de contenedores que se requieren para representar la parte de resto y el dos elevado a la potencia del valor de parámetro describe el número de valores, el cual se puede representar con el mismo prefijo. Estos valores forman un grupo de prefijo. Por ejemplo, para el parámetro cero, solo un prefijo puede representar un valor específico, mientras que para el parámetro tres, ocho valores de entrada comparten el mismo prefijo y, por lo tanto, un grupo de prefijo contiene ocho valores para el parámetro tres. Para un alfabeto de origen limitado y un código de Golomb-Rice dado, el último contenedor del prefijo se puede omitir para los valores en el último grupo de prefijo lo que da como resultado un código de prefijo con una longitud fija.

Por ejemplo, el rango puede ser nueve y el parámetro de Golomb-Rice es dos. Para el presente caso a modo de ejemplo, el número de valores que se puede representar con un mismo prefijo es cuatro. El valor máximo es nueve, lo que también indica que se supera la frontera y que se ha de usar el siguiente código por entropía de la siguiente partición. En el presente caso a modo de ejemplo, los valores de 0 a 3 tienen el prefijo 0, los valores de 4 a 7 tienen el prefijo 10 y de 8 a 9 el prefijo 110. Debido a que los valores 8 - 9 formaron el último grupo de prefijo, su cero final se puede omitir y los valores 8 - 9 se pueden representar por 11. Dicho de otra forma, imagínese que un símbolo de origen s_2 entró en el codificador de VLC 102 con el número de valores posibles de s_2 siendo 9 (= límite2 - límite1 + 1) y el parámetro de Golomb-Rice para ese símbolo de origen siendo dos. Entonces, una palabra de código de Golomb-Rice respectiva sería emitida por el codificador de VLC 102 para ese símbolo de origen, el cual tiene un prefijo tal como se acaba de describir. Para la parte de resto de la palabra de código, el código truncado se puede obtener tal como sigue por el codificador de VLC 102. Normalmente, el parámetro de los códigos de Golomb-Rice indica el número de contenedores de la parte de resto. En un caso truncado, no es necesario que se codifiquen todos los contenedores del resto. Para un caso truncado, se recuentan todos los valores con un prefijo fijo (por ejemplo, se omite un contenedor del prefijo). Obsérvese que el valor contado es siempre más pequeño que o igual al número máximo de valores para un prefijo debido a que el código está truncado. Si es posible un truncamiento de la parte de resto, la obtención de la parte de resto truncado para el último grupo de prefijo puede proseguir en las siguientes etapas. En primer lugar, se obtiene el número más grande *l* elevado a la potencia de dos menor o igual que el número contado. Entonces, en la segunda etapa, se obtiene el más pequeño número *h* elevado a la potencia de dos mayor que el número contado. El primer valor *l* describe el número de valores en el grupo de prefijo con un resto de *h* contenedores. Todos los restos de estos valores comienzan con un 0 seguido por la representación binaria del resto limitada al número de valores en el grupo de restos. Para los valores restantes del último grupo de prefijo, que se tratan ahora como un nuevo grupo de prefijo, se realiza el mismo procedimiento excepto por los valores resultantes que formaban el primer grupo de restos, el resto comienza con un 1. Este procedimiento se realiza hasta que se obtienen todos los restos. Como un ejemplo, el rango es 14 y el parámetro es tres. El primer grupo de prefijo contiene unos valores de 0 a 7 y el segundo grupo de prefijo contiene unos valores de 8 a 13. El segundo grupo de prefijo contiene seis valores. Los parámetros son *l* = 2 y *h* = 3. Por lo tanto, los primeros cuatro valores de los grupos de prefijo se representan por un resto con tres contenedores (un cero final y la representación binaria para distinguir los cuatro valores). Para los últimos dos valores, el mismo procedimiento se realiza de nuevo. Los parámetros son *l* = y *h* = 2. El resto de los últimos dos valores se pueden representar entonces como 10 y 11. Otro ejemplo para mostrar el método es un parámetro de Golomb-Rice de cuatro y un rango de diez. Para el presente ejemplo, los parámetros son *l* = 3 y *h* = 4. Con estos parámetros, la parte de resto truncado para los primeros ocho valores se representa por cuatro contenedores. Los dos valores restantes tienen la misma parte de resto que en el ejemplo previo en lo que antecede. Si el rango es nueve para el ejemplo previo, el parámetro para la segunda serie es *l* = 0 y *h* = 1. La parte de resto del único valor que queda es 1.

El tercer tipo de códigos por entropía pueden ser códigos de Golomb Exponencial. Estos se pueden emplear para

unas distribuciones equiprobables (por ejemplo, con parámetro cero) tales como los símbolos de origen s_3. Es decir, el par de codificador / descodificador de VLC puede ser responsable de su codificación. Tal como se ha mencionado antes, los niveles de coeficiente de transformada absolutos más grandes a menudo se distribuyen de manera uniforme. Más precisamente, el código de Golomb Exponencial de orden cero se puede usar para codificar la última partición 140_3. El comienzo y, por lo tanto, la frontera 144 para la partición previa 140_2 pueden ser variables. La posición de la frontera 144 puede ser controlada por el codificador / descodificador de VLC 102 / 200 con dependencia de los símbolos de origen 106, 108 y / o 110 previamente codificados / descodificados o los elementos de sintaxis 138 (o 218, 204 y / o 208 o los elementos de sintaxis 226).

El número de particiones puede ser tres tal como se muestra en la figura 1b y los límites 142 y 144 pueden ser variables. Para la primera partición 140_1, la codificación de PIPE se puede emplear tal como se ha analizado en lo que antecede. No obstante, como alternativa también se puede usar CABAC. En ese caso, el par de codificador / descodificador de PIPE se sustituiría por un par de codificador / descodificador de codificación aritmética binaria. Los códigos de Golomb-Rice truncados se pueden usar para la segunda partición 140_2 y el código de Golomb Exponencial de orden cero se puede usar para la última partición 140_3.

El número de particiones puede ser igual a dos. La primera partición 140_1 podría usar CABAC o PIPE. La segunda partición podría usar códigos de Golomb-Rice tales como 140_2.

El número de particiones puede ser igual a tres mientras que ambas fronteras 142 y 144 son variables. Por ejemplo, para la primera partición 140_1, se emplea CABAC o PIPE, mientras que la segunda partición 140_2 puede usar el código de Golomb-Rice truncado y la tercera partición 140_3 usa el código de Golomb Exponencial de orden cero.

El límite 142 de la primera partición 140_1 usando CABAC o PIPE, que emplea modelos de probabilidad adaptativos, puede ser dos. El modelado de contexto para el primer contenedor se puede realizar tal como se describe para coeff_abs_greater_one tal como se ha descrito en lo que antecede y el modelado de contexto para el segundo contenedor se puede realizar tal como se describe para coeff_abs_level_minus_two en la norma H.264 / AVC tal como se ha descrito también en lo que antecede. Esta última determinación de contexto sería determinada por las unidades de asignación 114 y 212, de forma respectiva. El límite 142 de la primera partición 140_1 usando una codificación por entropía que emplea modelado de probabilidad (por ejemplo, PIPE o CABAC) puede ser dos y el modelado de contexto tanto para el primero como para el segundo contenedor se puede realizar tal como se describe para coeff_abs_greater_one en la norma H.264 / AVC tal como se ha descrito en lo que antecede. La evaluación de conjuntos de contexto tal como se describe para coeff_abs_greater_one se puede realizar por separado para el segundo contenedor.

Se puede usar un conjunto de códigos de Golomb-Rice truncados como códigos por entropía de la segunda partición 140_2. El límite 144 de la segunda partición que especifica el comienzo de la tercera partición 140_3 dependiendo del parámetro de código por entropía puede ser variable. Asimismo, el parámetro de Golomb-Rice puede estar limitado por tres y la selección de parámetros se puede realizar como el modelado de contexto para coeff_abs_level_minus_two en la norma H.264 / AVC. El rango límite - límite2 puede ser variable y puede depender del parámetro de Golomb-Rice. Si el parámetro es cero, el rango es 8. Para el parámetro uno, el rango es 10. En el caso del parámetro dos, el rango es 12 y para el parámetro tres, el rango es igual a 16. En el presente ejemplo, el parámetro de Golomb-Rice puede establecerse a cero en el comienzo de un bloque de coeficientes de transformada. Para cada nivel de coeficiente de transformada codificado en el bloque más grande o igual que el primer límite, se usa el código de Golomb-Rice correspondiente. Después de la codificación (o la descodificación) de un nivel, se hace la siguiente evaluación para actualizar el parámetro de Golomb-Rice para la codificación (o la descodificación) del siguiente nivel mayor o igual que el primer límite. Obsérvese que el parámetro de Golomb-Rice no se puede disminuir mediante el uso de esta forma de adaptación.

La regla de adaptación de parámetros se puede resumir tal como sigue, en donde k_{t+1} indica el parámetro de Golomb-Rice que se va a usar para la codificación del siguiente valor de nivel y $valor_t$ indica el valor previamente codificado con el parámetro de Golomb-Rice correspondiente k_t

$$k_{t+1} = \begin{cases} 0 & valor_t \in [0,1] \wedge k_t < 1 \\ 1 & valor_t \in [2,3] \wedge k_t < 2 \\ 2 & valor_t \in [4,5] \wedge k_t < 3 \\ 3 & valor_t > 5 \wedge k_t < 4 \\ k_t & en\ otro\ caso \end{cases} \qquad (QQ)$$

En una realización preferida, se puede usar un conjunto de códigos de Golomb-Rice truncados como códigos por entropía de la segunda partición 140_2. El límite 144 de la segunda partición 140_2 que especifica el comienzo de la

tercera partición 140_3 dependiendo del parámetro de código por entropía puede ser variable. También en la presente realización preferida, el parámetro de Golomb-Rice puede estar limitado por tres y la selección de parámetros se puede realizar como modelado de contexto para coeff_abs_level_minus_two en la norma H.264 / AVC. El rango puede ser variable y depender del parámetro de Golomb-Rice. Si el parámetro es cero, el rango es 8. Para el parámetro uno, el rango es 10. En el caso del parámetro dos, el rango es 12 y para el parámetro tres, el rango es igual a 16. En la presente realización preferida, el parámetro de Golomb-Rice se establece a cero en el comienzo del bloque. La adaptación de parámetros de Golomb-Rice se realiza tal como se describe mediante la ecuación (QQ). Obsérvese que el parámetro no se puede disminuir mediante el uso de esta forma de adaptación.

En otra realización preferida, se puede usar un conjunto de códigos de Golomb-Rice truncados como códigos por entropía de la segunda partición 140_2. El límite 144 de la segunda partición 140_2 que especifica el comienzo de la tercera partición 140_3 dependiendo del parámetro de código por entropía puede ser fijo. También en la presente realización preferida, el parámetro de Golomb-Rice puede estar limitado a tres y la selección de parámetros se puede realizar como modelado de contexto para coeff_abs_level_minus_two en la norma H.264 / AVC. El rango de la segunda partición 140_2 se puede fijar a 14. En la presente realización preferida, el parámetro de Golomb-Rice puede establecerse a cero en el comienzo del bloque. La adaptación de parámetros de Golomb-Rice se realiza tal como se describe mediante la ecuación (QQ). Obsérvese que el parámetro no se puede disminuir mediante el uso de esta forma de adaptación.

En otra realización preferida, se puede usar un conjunto de códigos de Golomb-Rice truncados como códigos por entropía de la segunda partición 140_2. El límite 144 de la segunda partición 140_2 que especifica el comienzo de la tercera partición 140_3 dependiendo del parámetro de código por entropía puede ser variable. También en la presente realización preferida, el parámetro de Golomb-Rice puede estar limitado a tres y la selección de parámetros se puede realizar como modelado de contexto para coeff_abs_level_minus_two en la norma H.264 / AVC. El rango puede ser variable y depender del parámetro de Golomb-Rice. Si el parámetro puede ser cero, el rango puede ser 8. Para el parámetro uno, el rango puede ser 10. En el caso del parámetro dos, el rango puede ser 12 y para el parámetro tres, el rango puede ser igual a 16. En la presente realización preferida, el parámetro de Golomb-Rice puede establecerse a cero en el comienzo del bloque. La adaptación de parámetros de Golomb-Rice se realiza tal como se describe mediante la ecuación (QQ). Obsérvese que el parámetro no se puede disminuir mediante el uso de esta forma de adaptación. Y obsérvese también que es posible una conmutación directa, por ejemplo de cero a tres. En la presente realización preferida, la parte de prefijo de los códigos de Golomb-Rice se codifica con códigos por entropía empleando modelos de probabilidad. El modelado de contexto se puede realizar como para coeff_abs_level_minus_two en la norma H.264 / AVC.

En otra realización preferida, un parámetro de Golomb-Rice fijo se puede usar para codificar todos los niveles de coeficiente de transformada en el bloque de transformada actual. En la presente realización, el mejor parámetro del bloque previo se puede calcular y usar para el bloque de transformada actual. Para la presente realización, el rango se puede fijar a 14.

En otra realización preferida, un parámetro de Golomb-Rice fijo se puede usar para codificar todos los niveles de coeficiente de transformada en el bloque de transformada actual. En la presente realización, el mejor parámetro del bloque previo se puede calcular y usar para el bloque de transformada actual. Para la presente realización, el rango puede ser variable tal como se ha descrito anteriormente.

En otra realización preferida, se evalúa si las proximidades ya codificadas (o descodificadas) del índice de exploración actual contienen unos niveles de coeficiente de transformada absolutos más grandes que el límite previo. Para la presente realización preferida, el mejor parámetro se puede obtener mediante el uso de vecinos en una plantilla causal local.

Por lo tanto, las realizaciones que se han mencionado en lo que antecede describieron entre otras cosas, un aparato de codificación por entropía que comprende una unidad de descomposición 136 que está configurada para convertir una secuencia 138 de elementos de sintaxis en una secuencia 106 de símbolos de origen 106 mediante la descomposición de forma individual de por lo menos un subgrupo de los elementos de sintaxis en un número respectivo n de símbolos de origen s_i con i = 1 ... n, el número respectivo n de símbolos de origen dependiendo de en cual de una secuencia de n particiones 140_{1-3} en las cuales está subdividido un intervalo de valores de los elementos de sintaxis respectivos, cae un valor z de los elementos de sintaxis respectivos, de tal modo que una suma de valores del número respectivo de símbolos de origen S_i da z, y, si n > 1, para todo i = 1 ... n-1, el valor de s_i se corresponde con un intervalo de la i-ésima partición; un subdivisor 100 que está configurado para subdividir la secuencia 106 de símbolos de origen en una primera subsecuencia 108 de símbolos de origen y una segunda subsecuencia 110 de símbolos de origen de tal modo que todos los símbolos de origen s_x con x siendo miembro de un primer subconjunto de {1 ... n} están contenidos dentro de la primera subsecuencia 108 y todos los símbolos de origen s_y con y siendo miembro de un segundo subconjunto de {1 ... n} que es disjunto con respecto al primer subconjunto, están contenidos dentro de la segunda subsecuencia 110; un codificador de VLC 102 que está configurado para codificar por símbolos los símbolos de origen de la primera subsecuencia 108; y un codificador de

PIPE o aritmético 104 que está codificado para codificar la segunda subsecuencia 110 de símbolos de origen.

Los valores z del subgrupo de los elementos de sintaxis pueden ser valores absolutos. El segundo subconjunto puede ser {1} con la secuencia de n particiones estando dispuesta de tal modo que una p-ésima partición cubre valores más altos del intervalo de valores que una q-ésima partición para todo p, q ∈ {1 .. n} con p > q. n puede ser 3. El primer subconjunto puede ser {2, 3} con el un codificador de VLC (102) configurado para usar un código de Golomb-Rice para codificar por símbolos los símbolos de origen s_2, y un código de Golomb Exponencial para codificar por símbolos los símbolos de origen s_3. Más en general, 2 puede ser un elemento del primer subconjunto con el un codificador de VLC (102) estando configurado para usar un código de Golomb-Rice para codificar por símbolos los símbolos de origen s_2 y adaptar un parámetro de Golomb-Rice, es decir k, del código de Golomb-Rice de acuerdo con unos símbolos de origen previamente codificados. La unidad de descomposición se puede configurar para adaptar uno o más de los límites entre las particiones de acuerdo con unos símbolos de origen previamente codificados. Ambas adaptaciones se pueden combinar. Es decir, unas posiciones de los límites que limitan la segunda partición se pueden adaptar de tal modo que estas están separadas entre sí de tal modo que la longitud del código de Golomb-Rice, es decir el número de palabras de código del mismo, se corresponde con (o dicta) la longitud de la anchura de la segunda partición. El límite entre la separación de la primera y la segunda partición se puede adaptar de acuerdo con otra dependencia del contexto, caso en el cual, la adaptación de k puede definir la posición del límite que separa la segunda y la tercera partición por medio de la longitud del código de Golomb-Rice y la anchura de la segunda partición, de forma respectiva. Mediante el acoplamiento de la adaptación de k de tal modo que la anchura de la segunda partición se corresponde con la longitud del código de Golomb-Rice, la eficiencia de código se usa de forma óptima. Adaptar k a las estadísticas del elemento de sintaxis posibilita adaptar la anchura de la segunda partición de tal modo que la tercera partición puede cubrir tanto como sea posible con el fin de reducir la complejidad de codificación global debido a que se puede haber usado un código menos complejo para la tercera partición tal como el código de Golomb Exponencial. Además, la longitud de la primera partición se puede restringir a j ∈ {1, 2, 3} valores de elemento de sintaxis posibles, tal como los tres niveles más bajos. Los elementos de sintaxis que se están considerando se pueden codificar de forma diferencial o representan un residuo de predicción, este es el caso con los niveles de coeficiente de transformada ejemplificados en lo que antecede los cuales representan un residuo de predicción. Los primeros símbolos de origen s_1 se pueden simbolizar / desimbolizar mediante el uso de un código unario truncado con los j contenedores resultantes codificándose - parcialmente o la totalidad de estos - de forma adaptativa con el contexto o no tal como se ha mencionado en lo que antecede.

El subgrupo de los elementos de sintaxis puede abarcar unos niveles de coeficiente de transformada absolutos de coeficientes de transformada absolutos de los bloques de transformada de una imagen con los niveles de coeficiente de transformada absolutos de un bloque de transformada respectivo estando dispuestos dentro de la secuencia (138) de elementos de sintaxis de acuerdo con una trayectoria de exploración que conduce a través de los coeficientes de transformada absolutos de los bloques de transformada respectivos, en donde la unidad de descomposición se puede configurar para adaptar uno o más de los límites entre las particiones durante la descomposición de los niveles de coeficiente de transformada absolutos de los coeficientes de transformada absolutos de un bloque de transformada respectivo dependiendo de unos niveles de coeficiente de transformada absolutos ya codificados de coeficientes de transformada absolutos de los bloques de transformada respectivos precedentes en el orden de exploración o dependiendo de una posición del nivel de coeficiente de transformada absoluto que se va a descomponer en la actualidad en el orden de exploración, o sobre la base de una evaluación de los niveles de coeficiente de transformada absolutos ya reconstruidos de coeficientes de transformada en las proximidades - o bien espacialmente o bien en el orden de exploración - de la posición del nivel de coeficiente de transformada absoluto que se va a descomponer en la actualidad.

Además, las realizaciones que se han mencionado en lo que antecede describen entre otras cosas un aparato de descodificación por entropía que comprende un descodificador de VLC 200 que está configurado para reconstruir por palabras de código unos símbolos de origen de una primera subsecuencia 204 de símbolos de origen a partir de palabras de código de un primer tren de bits 206; un descodificador de PIPE o aritmético 202 que está configurado para reconstruir una segunda subsecuencia 208 de símbolos de origen; una unidad de composición 224 que está configurada para componer una secuencia 226 de elementos de sintaxis a partir de la primera subsecuencia 204 de símbolos de origen y la segunda subsecuencia 208 de símbolos de origen mediante la composición de forma individual de cada elemento de sintaxis a partir de un número respectivo de símbolos de origen, en donde la unidad de composición está configurada para determinar, para por lo menos un subgrupo de los elementos de sintaxis, el número respectivo n de símbolos de origen s_i con i = 1 ... n dependiendo de en cual de una secuencia de n particiones 140_{1-3} en las cuales está subdividido un intervalo de valores de los elementos de sintaxis respectivos, cae un valor z de los elementos de sintaxis respectivos, mediante la adición de los valores del número respectivo de símbolos de origen s_i de 1 a n siempre que el valor de s_i se corresponda con un intervalo de la i-ésima partición con el fin de obtener el valor del elemento de sintaxis z, en donde la unidad de composición 224 está configurada para recuperar todos los símbolos de origen s_x con x siendo miembro de un primer subconjunto de {1 ... n} a partir de la primera subsecuencia (204) y todos los símbolos de origen s_y con y siendo miembro de un segundo subconjunto de {1 ... n} que es disjunto con respecto al primer subconjunto, a partir de la segunda subsecuencia 208. Los valores z

del subgrupo de los elementos de sintaxis pueden ser valores absolutos. El segundo subconjunto puede ser {1} con la secuencia de n particiones estando dispuesta de tal modo que una p-ésima partición cubre valores más altos del intervalo de valores que una q-ésima partición para todo p, q ∈ {1 .. n} con p > q. n puede ser 3. El primer subconjunto puede ser {2, 3} con el un descodificador de VLC 200 estando configurado para usar un código de Golomb-Rice para reconstruir por palabras de código los símbolos de origen s_2, y un código de Golomb Exponencial para reconstruir por palabras de código los símbolos de origen s_3. Más en general, 2 puede ser un elemento del primer subconjunto con el un descodificador de VLC 102 estando configurado para usar un código de Golomb-Rice para reconstruir por palabras de código los símbolos de origen s_2 y adaptar un parámetro de Golomb-Rice del código de Golomb-Rice de acuerdo con unos símbolos de origen previamente reconstruidos. El aparato de descodificación por entropía puede comprender adicionalmente un recombinador 220 que está configurado para recombinar la primera subsecuencia 204 de símbolos de origen y la segunda subsecuencia de símbolos de origen para obtener la secuencia 218 de símbolos de origen. Los elementos de sintaxis pueden ser de diferente tipo y la unidad de composición se puede configurar para realizar la composición individual dependiendo del tipo de los elementos de sintaxis. El subgrupo de los elementos de sintaxis puede abarcar unos niveles de coeficiente de transformada absolutos de coeficientes de transformada absolutos de los bloques de transformada de una imagen con los niveles de coeficiente de transformada absolutos de un bloque de transformada respectivo estando dispuestos dentro de la secuencia 138 de elementos de sintaxis de acuerdo con una trayectoria de exploración que conduce a través de los coeficientes de transformada absolutos de los bloques de transformada respectivos, en donde la unidad de composición se puede configurar para adaptar uno o más de los límites entre las particiones durante la composición de los niveles de coeficiente de transformada absolutos de los coeficientes de transformada absolutos de un bloque de transformada respectivo dependiendo de unos niveles de coeficiente de transformada absolutos ya reconstruidos de coeficientes de transformada absolutos de los bloques de transformada respectivos precedentes en el orden de exploración o dependiendo de una posición del nivel de coeficiente de transformada absoluto que se va a componer en la actualidad en el orden de exploración, o sobre la base de una evaluación de los niveles de coeficiente de transformada absolutos ya reconstruidos de coeficientes de transformada en las proximidades - o bien espacialmente o bien en el orden de exploración - de la posición del nivel de coeficiente de transformada absoluto que se va a componer en la actualidad.

En lo que respecta a la combinación de la codificación de PIPE con la codificación de VLC usando la descomposición de acuerdo con la figura 1b, lo sucesivo se hace notar con el fin de repetir algunos aspectos de la misma con otras palabras.

Se ha descrito la puesta en correspondencia de una secuencia de símbolos con un tren de bits y la puesta en correspondencia inversa. Cada símbolo porta un parámetro o parámetros asociados con este que se conocen de forma simultánea en el codificador y en el descodificador. El códec por entropía contiene múltiples memorias intermedias de tipo primero en entrar primero en salir (FIFO, *first-in first-out*) con cada una de ellas asignada a subconjuntos de un parámetro o parámetros que están asociados con los símbolos. Para un parámetro o parámetros dados de un símbolo, el codificador asigna el símbolo a la memoria intermedia de tipo FIFO correspondiente. Debido a que la regla de asignación de codificadores se conoce en el lado de descodificador, el descodificador lee a partir de la memoria intermedia de tipo FIFO a cuál el codificador ha asignado el símbolo.

Algunos elementos de sintaxis se codifican usando códigos de longitud variable convencionales y se escriben en una memoria intermedia particular. Otros elementos de sintaxis se codifican usando el concepto de codificación de entropía de subdivisión en particiones de intervalos de probabilidad (PIPE, *probability interval partitioning entropy*). En este, los símbolos se binarizan en primer lugar y los contenedores resultantes se clasifican sobre la base de estimadas de probabilidad asociadas. La estimada de probabilidad se puede dar u obtener a partir de una medición que se puede llevar a cabo de forma simultánea en el codificador y en el descodificador. Una memoria intermedia de tipo FIFO particular contiene símbolos con unos valores de probabilidad estimada que caen dentro de un subconjunto de probabilidades que se elige de tal modo que se puede mejorar la codificación por entropía. La mejora lograda por una combinación del concepto de PIPE con VLC es una reducción en la complejidad al tiempo que se sigue proporcionando una eficiencia de codificación alta. Unos símbolos para los cuales un código de VLC convencional es adecuado se codifican con el enfoque de VLC simple y de baja complejidad, mientras que otros símbolos para los cuales la tasa de bits se incrementaría de forma significativa mediante la codificación de estos con un código de VLC se codifican con el más sofisticado concepto de PIPE.

Por lo tanto, con el fin de reducir adicionalmente la complejidad de la codificación por entropía, los símbolos se han dividido en dos categorías. Los símbolos de una primera categoría se pueden representar bien con códigos de VLC y no requieren la más compleja codificación de PIPE, mientras que los símbolos de una segunda categoría no se pueden representar de forma eficiente con códigos de VLC y una codificación de PIPE para estos los símbolos redujo de forma significativa la tasa de bits requerida.

A pesar de que algunos aspectos se han descrito en el contexto de un aparato, es evidente que estos aspectos también representan una descripción del método correspondiente, en donde un bloque o dispositivo se corresponde con una etapa de método o una característica de una etapa de método. De forma análoga, los aspectos que se

describen en el contexto de una etapa de método también representan una descripción de un bloque o artículo o característica correspondiente de un aparato correspondiente. Algunas o la totalidad de las etapas de método pueden ser ejecutadas por (o usando) un aparato de soporte físico, como por ejemplo, un procesador, un ordenador programable o un circuito electrónico. En algunas realizaciones, algunas una o más de las etapas de método más importantes pueden ser ejecutadas por un aparato de este tipo.

Las señales codificadas / comprimidas de la invención se pueden almacenar en un medio de almacenamiento digital o se pueden transmitir por un medio de transmisión tal como un medio de transmisión inalámbrica o un medio de transmisión por cable tal como Internet.

Dependiendo de determinados requisitos de implementación, se pueden implementar realizaciones de la invención en soporte físico o en soporte lógico. La implementación se puede realizar usando un medio de almacenamiento digital, por ejemplo un disco flexible, un DVD, un Blue-Ray, un CD, una ROM, una PROM, una EPROM, una EEPROM o una memoria FLASH, que tiene unas señales de control legibles por medios electrónicos almacenadas en el mismo, las cuales cooperan (o son capaces de cooperar) con un sistema informático programable de tal modo que se realiza el método respectivo. Por lo tanto, el medio de almacenamiento digital puede ser legible por ordenador.

Algunas realizaciones de acuerdo con la invención comprenden un medio de soporte de datos que tiene unas señales de control legibles por medios electrónicos, las cuales son capaces de cooperar con un sistema informático programable, de tal modo que se realiza uno de los métodos que se describen en el presente documento.

En general, se pueden implementar realizaciones de la presente invención como un producto de programa informático con un código de programa, siendo el código de programa operativo para realizar uno de los métodos cuando el producto de programa informático se ejecuta en un ordenador. El código de programa se puede almacenar por ejemplo en un medio de soporte legible por máquina.

Otras realizaciones comprenden el programa informático para realizar uno de los métodos que se describen en el presente documento, almacenado en un medio de soporte legible por máquina.

Dicho de otra forma, una realización del método de la invención es, por lo tanto, un programa informático que tiene un código de programa para realizar uno de los métodos que se describen en el presente documento, cuando el programa informático se ejecuta en un ordenador.

Una realización adicional de los métodos de la invención es, por lo tanto, un medio de soporte de datos (o un medio de almacenamiento digital, o un medio legible por ordenador) que comprende, registrado en el mismo, el programa informático para realizar uno de los métodos que se describen en el presente documento.

Una realización adicional del método de la invención es, por lo tanto, un tren de datos o una secuencia de señales que representan el programa informático para realizar uno de los métodos que se describen en el presente documento. El tren de datos o la secuencia de señales se puede configurar por ejemplo para transferirse por medio de una conexión de comunicación de datos, por ejemplo por medio de Internet.

Una realización adicional comprende unos medios de procesamiento, por ejemplo un ordenador, o un dispositivo lógico programable, configurado para o adaptado para realizar uno de los métodos que se describen en el presente documento.

Una realización adicional comprende un ordenador que tiene instalado en el mismo el programa informático para realizar uno de los métodos que se describen en el presente documento.

En algunas realizaciones, un dispositivo lógico programable (por ejemplo una matriz de puertas programable en campo) se puede usar para realizar algunas o la totalidad de las funcionalidades de los métodos que se describen en el presente documento. En algunas realizaciones, una matriz de puertas programable en campo puede cooperar con un procesador con el fin de realizar uno de los métodos que se describen en el presente documento. En general, preferiblemente los métodos son realizados por cualquier aparato de soporte físico.

Las realizaciones que se han descrito en lo que antecede son meramente ilustrativas de los principios de la presente invención. Se entiende que modificaciones y variaciones de las disposiciones y los detalles que se describen en el presente documento serán evidentes a otros expertos en la materia. La intención es, por lo tanto, estar limitado solo por el alcance de las inminentes reivindicaciones de patente y no por los detalles específicos que se presentan a modo de descripción y explicación de las realizaciones en el presente documento.

REIVINDICACIONES

1. Aparato de codificación por entropía que comprende
una unidad de descomposición (136) que está configurada para convertir una secuencia (138) de elementos de sintaxis que tienen un intervalo de valores el cual está subdividido en una secuencia de N particiones (140_{1-3}) en una secuencia (106) de símbolos de origen (106) mediante la descomposición de forma individual de por lo menos un subgrupo de los elementos de sintaxis en un número respectivo n de símbolos de origen s_i con i = 1 ... n, el número respectivo n de símbolos de origen dependiendo de en cual de la secuencia de N particiones (140_{1-3}) cae un valor z de los elementos de sintaxis respectivos, de tal modo que una suma de valores del número respectivo de símbolos de origen s_i da z, y, si n > 1, para todo i = 1 ... n-1, el valor de s_i se corresponde con un intervalo de la i-ésima partición;
un subdivisor (100) que está configurado para subdividir la secuencia (106) de símbolos de origen en una primera subsecuencia (108) de símbolos de origen y una segunda subsecuencia (110) de símbolos de origen de tal modo que todos los símbolos de origen s_x con x siendo miembro de un primer subconjunto de {1 ... N} están contenidos dentro de la primera subsecuencia (108) y todos los símbolos de origen s_y con y siendo miembro de un segundo subconjunto de {1 ... N} que es disjunto con respecto al primer subconjunto, están contenidos dentro de la segunda subsecuencia (110);
un codificador de VLC (102) que está configurado para codificar por símbolos los símbolos de origen de la primera subsecuencia (108); y
un codificador aritmético (104) que está codificado para codificar la segunda subsecuencia (110) de símbolos de origen, **caracterizado por que**
los valores z del subgrupo de los elementos de sintaxis son valores absolutos, y
en el que la unidad de descomposición está configurada para adaptar uno o más de los límites entre las particiones de acuerdo con unos símbolos de origen previamente codificados.

2. Aparato de descodificación por entropía que comprende
un descodificador de VLC (200) que está configurado para reconstruir por palabras de código los símbolos de origen de una primera subsecuencia (204) de símbolos de origen a partir de palabras de código de un primer tren de bits (206);
un descodificador aritmético (202) que está configurado para reconstruir una segunda subsecuencia (208) de símbolos de origen;
una unidad de composición (224) que está configurada para componer una secuencia (226) de elementos de sintaxis que tienen un intervalo de valores el cual está subdividido en una secuencia de N particiones (140_{1-3}) a partir de la primera subsecuencia (204) de símbolos de origen y la segunda subsecuencia (208) de símbolos de origen mediante la composición de forma individual de cada elemento de sintaxis a partir de un número respectivo n de símbolos de origen mediante, para por lo menos un subgrupo de los elementos de sintaxis, la determinación del número respectivo n de símbolos de origen s_i con i = 1 ... n dependiendo de en cual de la secuencia de N particiones (140_{1-3}) en las cuales está subdividido un intervalo de valores de los elementos de sintaxis respectivos, cae un valor z de los elementos de sintaxis respectivos, mediante la adición de los valores del número respectivo de símbolos de origen s_i de 1 a n siempre que el valor de s_i se corresponda con un intervalo de la i-ésima partición con el fin de obtener el valor del elemento de sintaxis z, en el que la unidad de composición (224) está configurada para recuperar todos los símbolos de origen s_x con x siendo miembro de un primer subconjunto de {1 ... N} a partir de la primera subsecuencia (204) y todos los símbolos de origen s_y con y siendo miembro de un segundo subconjunto de {1 ... N} que es disjunto con respecto al primer subconjunto, a partir de la segunda subsecuencia (208), **caracterizado por que**
los valores z del subgrupo de los elementos de sintaxis son valores absolutos, y
en el que la unidad de composición está configurada para adaptar uno o más de los límites entre las particiones de acuerdo con unos símbolos de origen previamente reconstruidos.

3. Aparato de descodificación por entropía de acuerdo con la reivindicación 2, en el que el segundo subconjunto es {1} con la secuencia de N particiones estando dispuesta de tal modo que una p-ésima partición cubre valores más altos del intervalo de valores que una q-ésima partición para todo p, q ∈ {1 .. N} con p > q.

4. Aparato de descodificación por entropía de acuerdo con la reivindicación 3, en el que N = 3.

5. Aparato de descodificación por entropía de acuerdo con la reivindicación 2, en el que 2 es un elemento del primer subconjunto con el descodificador de VLC (102) estando configurado para usar un código de Golomb-Rice para reconstruir por palabras de código los símbolos de origen s_2 y adaptar un parámetro de Golomb-Rice del código de Golomb-Rice de acuerdo con unos símbolos de origen previamente reconstruidos.

6. Aparato de descodificación por entropía de acuerdo con cualquiera de las reivindicaciones 2 a 4 que comprende adicionalmente un recombinador (220) que está configurado para recombinar la primera subsecuencia (204) de símbolos de origen y la segunda subsecuencia de símbolos de origen para obtener la secuencia (218) de símbolos de origen.

7. Método de codificación por entropía que comprende
convertir una secuencia (138) de elementos de sintaxis que tienen un intervalo de valores el cual está subdividido en una secuencia de N particiones (140_{1-3}) en una secuencia (106) de símbolos de origen (106) mediante la descomposición de forma individual de por lo menos un subgrupo de los elementos de sintaxis en un número respectivo n de símbolos de origen s_i con i = 1 ... n, el número respectivo n de símbolos de origen dependiendo de en cual de la secuencia de N particiones (140_{1-3}) cae un valor z de los elementos de sintaxis respectivos, de tal modo que una suma de valores del número respectivo de símbolos de origen s_i da z, y, si n > 1, para todo i = 1 ... n-1, el valor de s_i se corresponde con un intervalo de la i-ésima partición;
subdividir la secuencia (106) de símbolos de origen en una primera subsecuencia (108) de símbolos de origen y una segunda subsecuencia (110) de símbolos de origen de tal modo que todos los símbolos de origen s_x con x siendo miembro de un primer subconjunto de {1 ... N} están contenidos dentro de la primera subsecuencia (108) y todos los símbolos de origen s_y con y siendo miembro de un segundo subconjunto de {1 ... N} que es disjunto con respecto al primer subconjunto, están contenidos dentro de la segunda subsecuencia (110);
mediante codificación de VLC, codificar por símbolos los símbolos de origen de la primera subsecuencia (108); y
mediante codificación aritmética, codificar la segunda subsecuencia (110) de símbolos de origen,
caracterizado por que los valores z del subgrupo de los elementos de sintaxis son valores absolutos,
en el que la conversión mediante una descomposición individual comprende adaptar uno o más de los límites entre las particiones de acuerdo con unos símbolos de origen previamente codificados.

8. Método de descodificación por entropía que comprende
mediante descodificación de VLC, reconstruir por palabras de código los símbolos de origen de una primera subsecuencia (204) de símbolos de origen a partir de palabras de código de un primer tren de bits (206);
mediante descodificación aritmética, reconstruir una segunda subsecuencia (208) de símbolos de origen;
componer una secuencia (226) de elementos de sintaxis que tienen un intervalo de valores el cual está subdividido en una secuencia de N particiones (140_{1-3}) a partir de la primera subsecuencia (204) de símbolos de origen y la segunda subsecuencia (208) de símbolos de origen mediante la composición de forma individual de cada elemento de sintaxis a partir de un número respectivo n de símbolos de origen mediante, para por lo menos un subgrupo de los elementos de sintaxis, la determinación del número respectivo n de símbolos de origen s_i con i = 1 ... n dependiendo de en cual de la secuencia de N particiones (140_{1-3}) en las cuales está subdividido un intervalo de valores de los elementos de sintaxis respectivos, cae un valor z de los elementos de sintaxis respectivos, mediante la adición de los valores del número respectivo de símbolos de origen s_i de 1 a n siempre que el valor de s_i se corresponda con un intervalo de la i-ésima partición con el fin de obtener el valor del elemento de sintaxis z, en el que la composición (224) comprende recuperar todos los símbolos de origen s_x con x siendo miembro de un primer subconjunto de {1 ... N} a partir de la primera subsecuencia (204) y todos los símbolos de origen s_y con y siendo miembro de un segundo subconjunto de {1 ... N} que es disjunto con respecto al primer subconjunto, a partir de la segunda subsecuencia (208), **caracterizado por que**
los valores z del subgrupo de los elementos de sintaxis son valores absolutos, en el que la composición comprende adaptar uno o más de los límites entre las particiones de acuerdo con unos símbolos de origen previamente reconstruidos.

9. Un programa informático que tiene un código de programa para realizar, cuando se ejecuta en un ordenador, un método de acuerdo con la reivindicación 7 u 8.

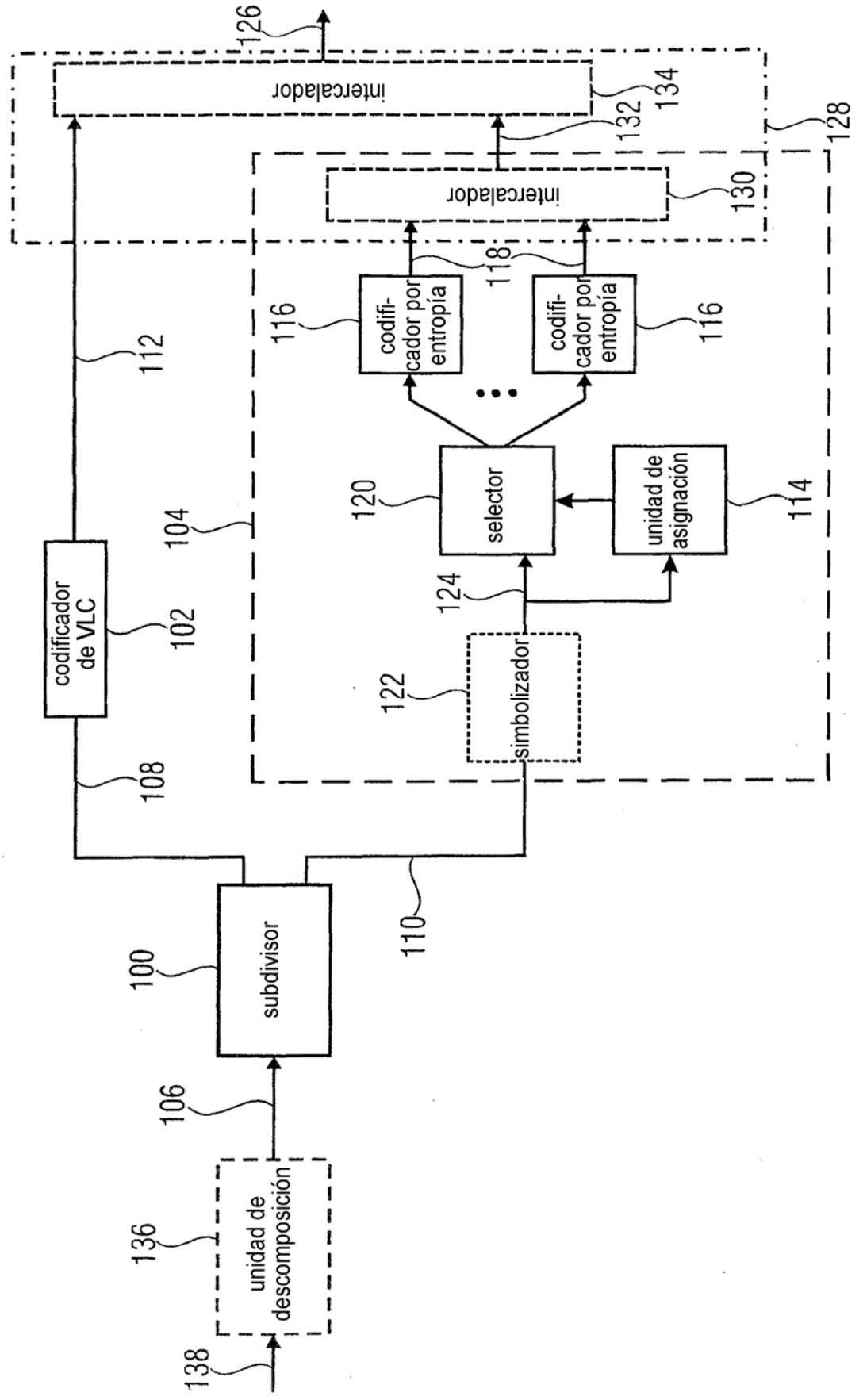

126
intercalador
134
132
128
intercalador
130
118
codifi-cador por entropía
116
codifi-cador por entropía
116
112
104
selector
120
unidad de asignación
114
codificador de VLC
102
124
simbolizador
122
108
110
subdivisor
100
106
unidad de descomposición
136
138

FIGURA 1A

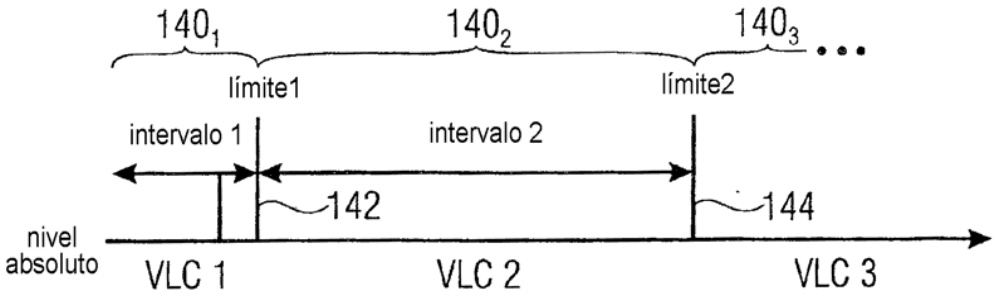


FIGURA 1B

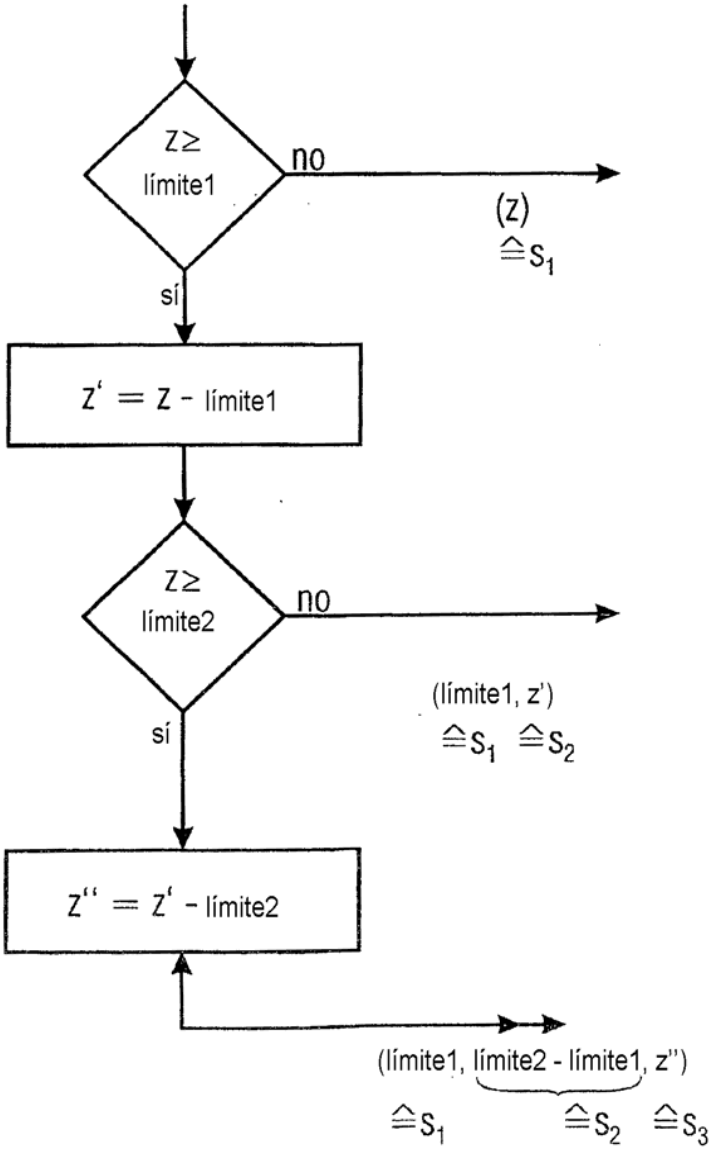


FIGURA 1C

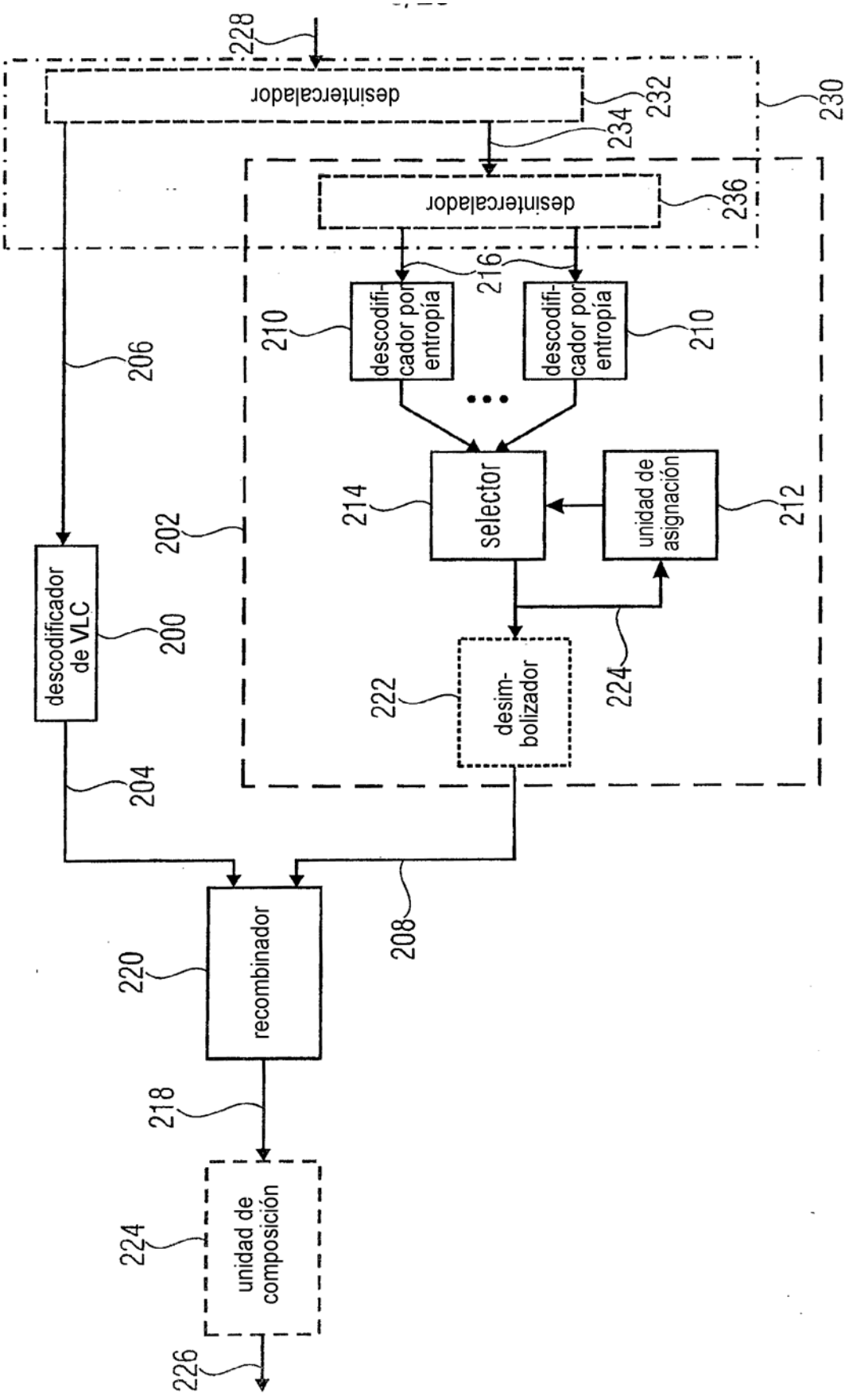

228
desintercalador
232
234
230
desintercalador
236
216
descodificador por entropía
210
descodificador por entropía
210
206
selector
214
unidad de asignación
212
202
descodificador de VLC
200
222
desimbolizador
224
204
208
recombinador
220
218
224
unidad de composición
226

FIGURA 2A

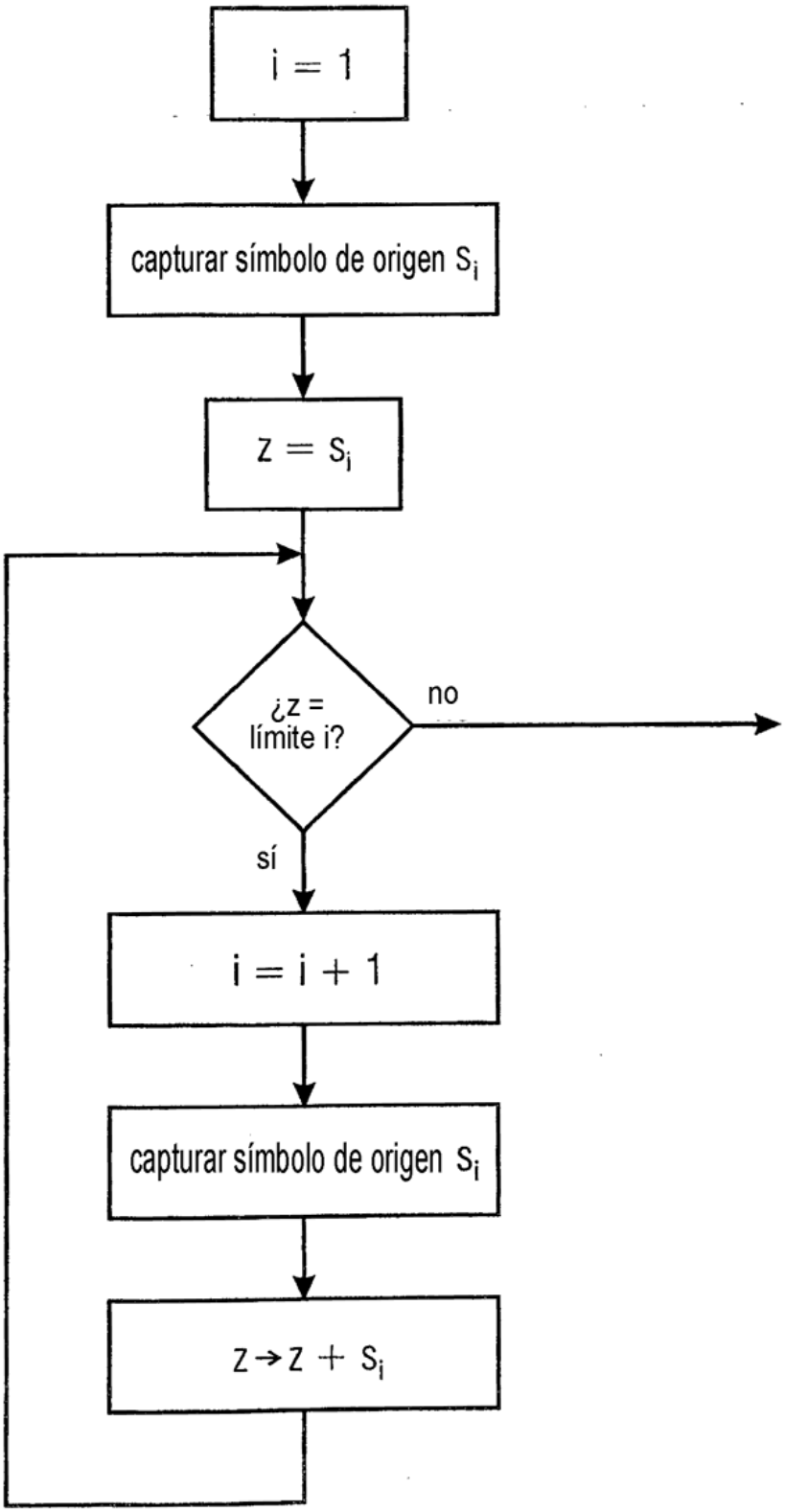

i = 1
capturar símbolo de origen S_i
$Z = S_i$
¿Z = límite i?
no
sí
i = i + 1
capturar símbolo de origen S_i
$Z \rightarrow Z + S_i$

FIGURA 2B

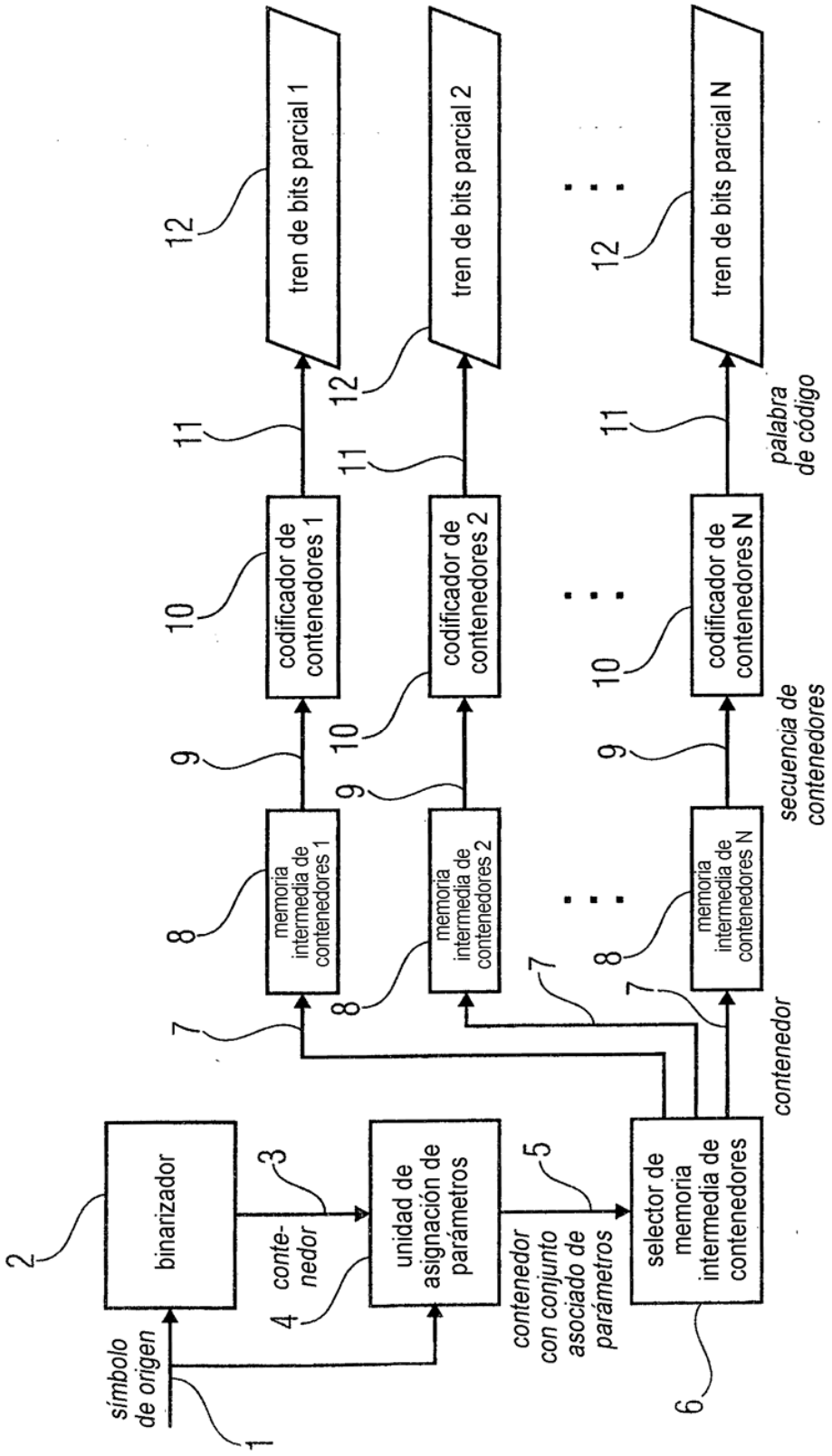

símbolo de origen
1
2
binarizador
3
conte-nedor
4
unidad de asignación de parámetros
5
contenedor con conjunto asociado de parámetros
6
selector de memoria intermedia de contenedores
7
contenedor
8
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
9
10
codificador de contenedores 1
codificador de contenedores 2
codificador de contenedores N
11
12
tren de bits parcial 1
tren de bits parcial 2
tren de bits parcial N
secuencia de contenedores
palabra de código

FIGURA 3

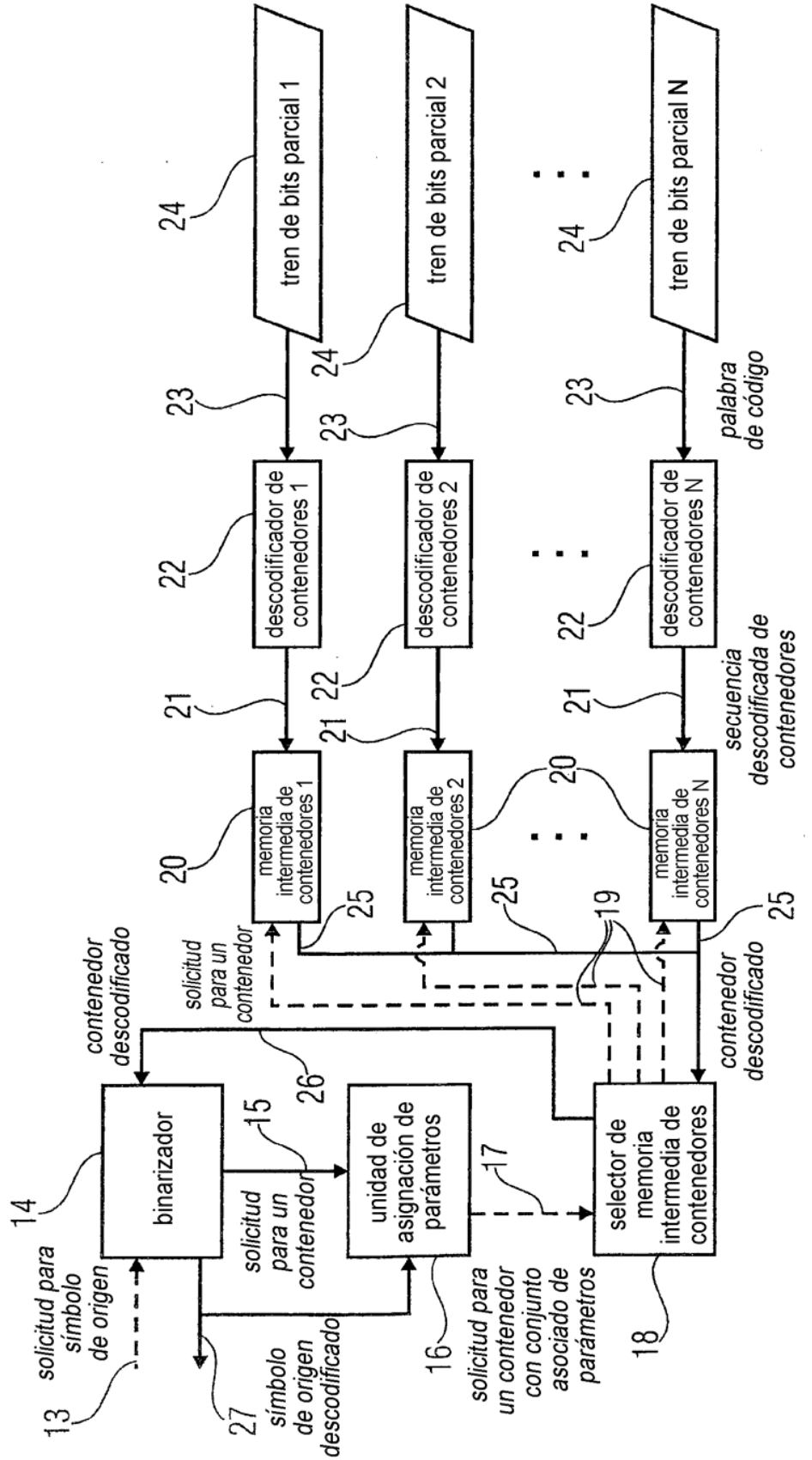

tren de bits parcial 1
tren de bits parcial 2
tren de bits parcial N
descodificador de contenedores 1
descodificador de contenedores 2
descodificador de contenedores N
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
palabra de código
secuencia descodificada de contenedores
contenedor descodificado
solicitud para un contenedor
binarizador
solicitud para símbolo de origen
solicitud para un contenedor
unidad de asignación de parámetros
selector de memoria intermedia de contenedores
símbolo de origen descodificado
solicitud para un contenedor con conjunto asociado de parámetros
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

FIGURA 4

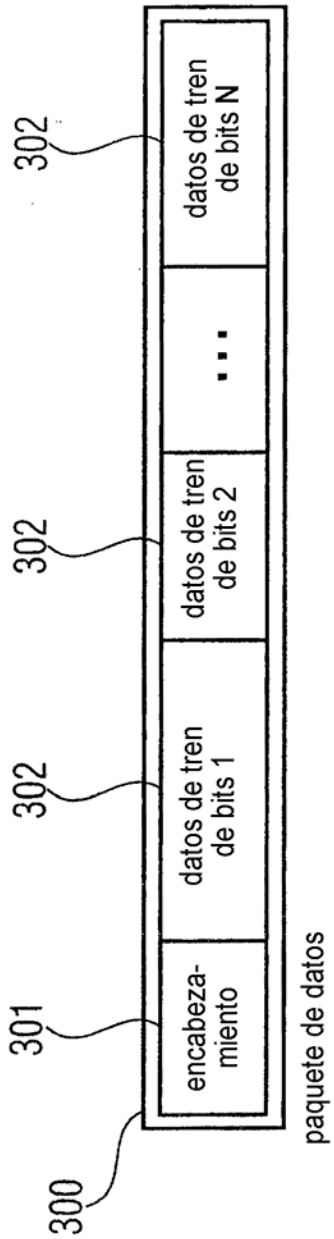


FIGURA 5

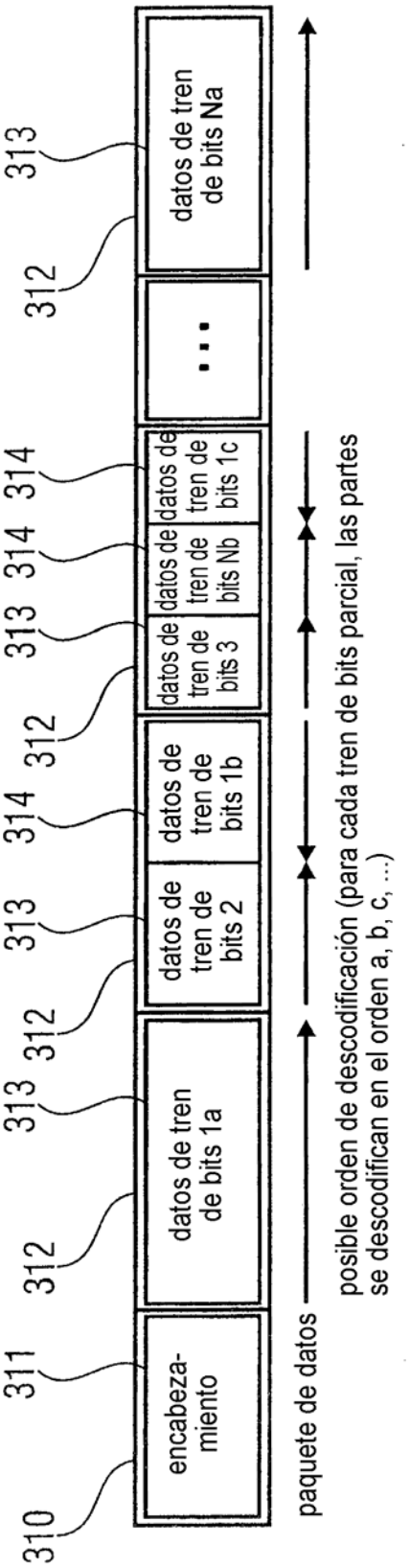


FIGURA 6

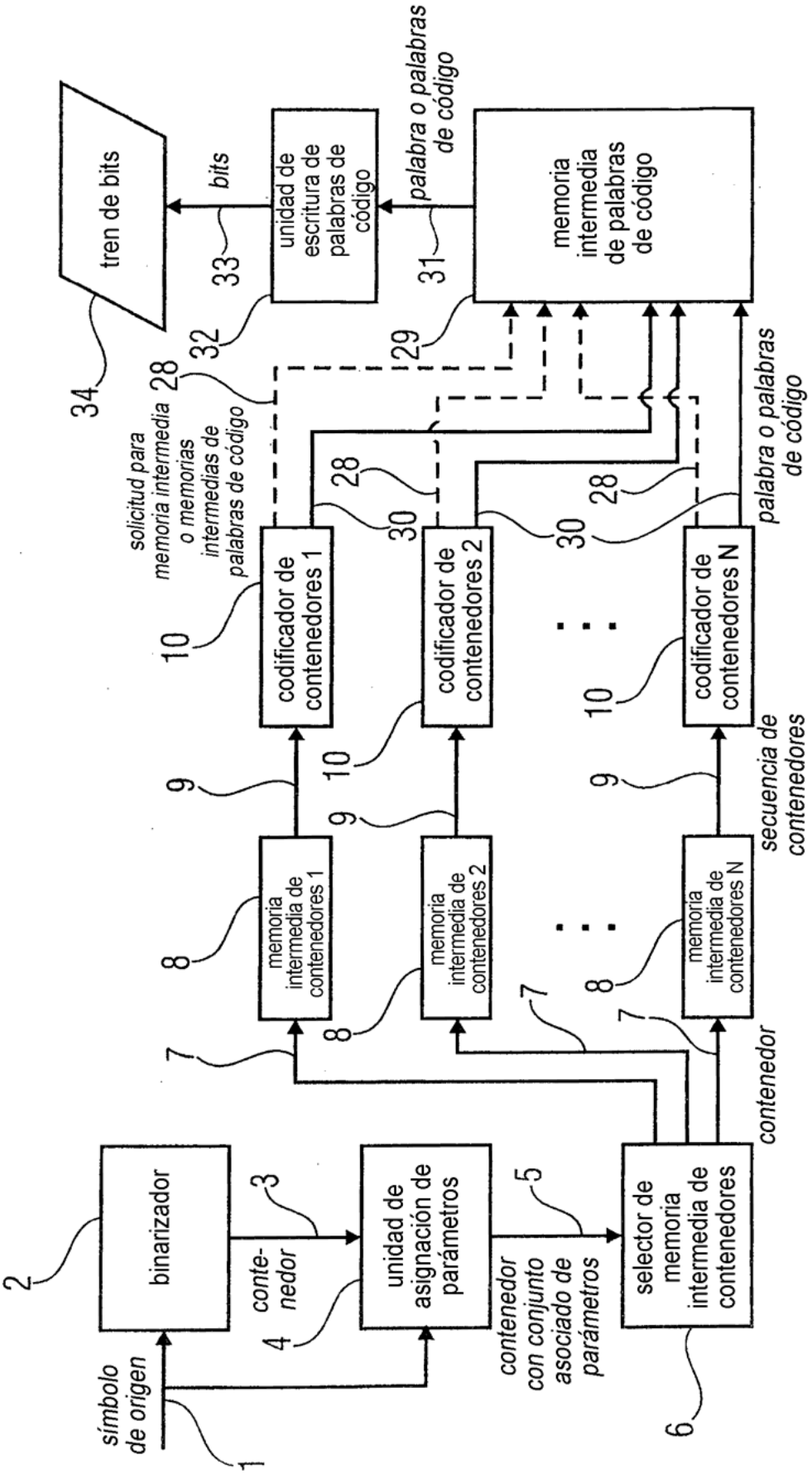

símbolo de origen
1
2
binarizador
3
conte-nedor
4
unidad de asignación de parámetros
5
contenedor con conjunto asociado de parámetros
6
selector de memoria intermedia de contenedores
7
contenedor
8
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
9
secuencia de contenedores
10
codificador de contenedores 1
codificador de contenedores 2
codificador de contenedores N
28
solicitud para memoria intermedia o memorias intermedias de palabras de código
29
30
palabra o palabras de código
31
memoria intermedia de palabras de código
32
unidad de escritura de palabras de código
33
bits
34
tren de bits

FIGURA 7

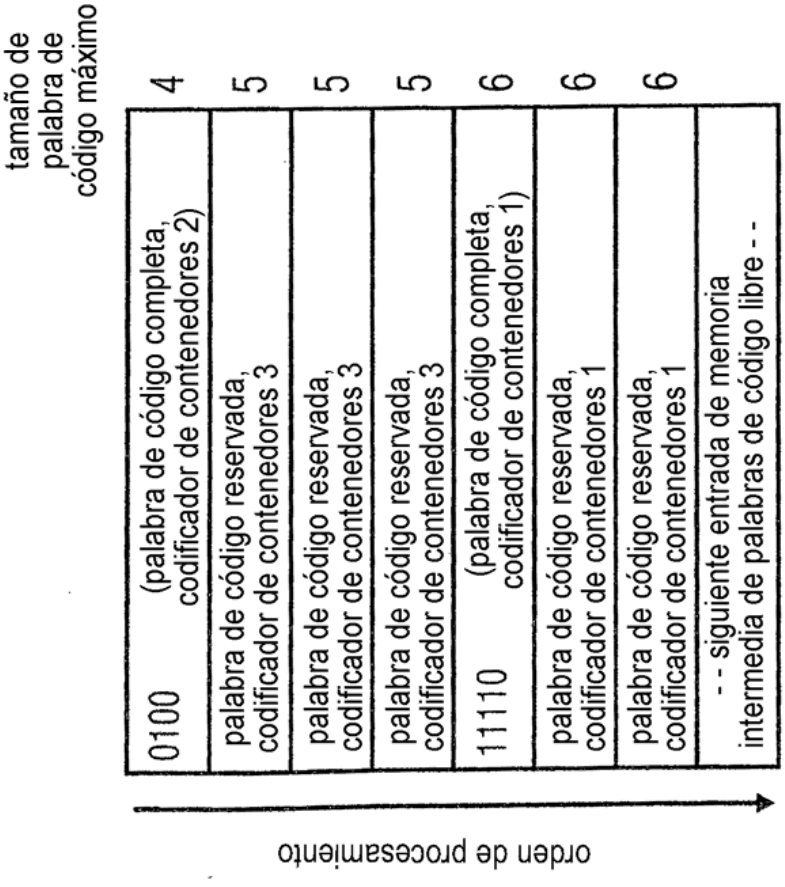


FIGURA 8A

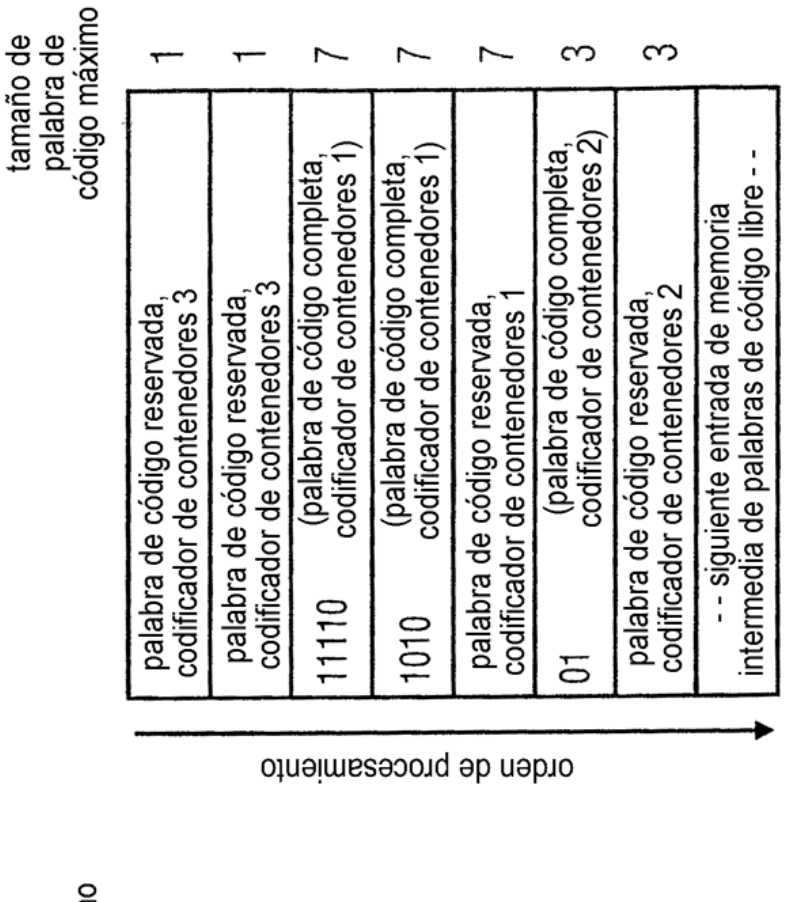


FIGURA 8B

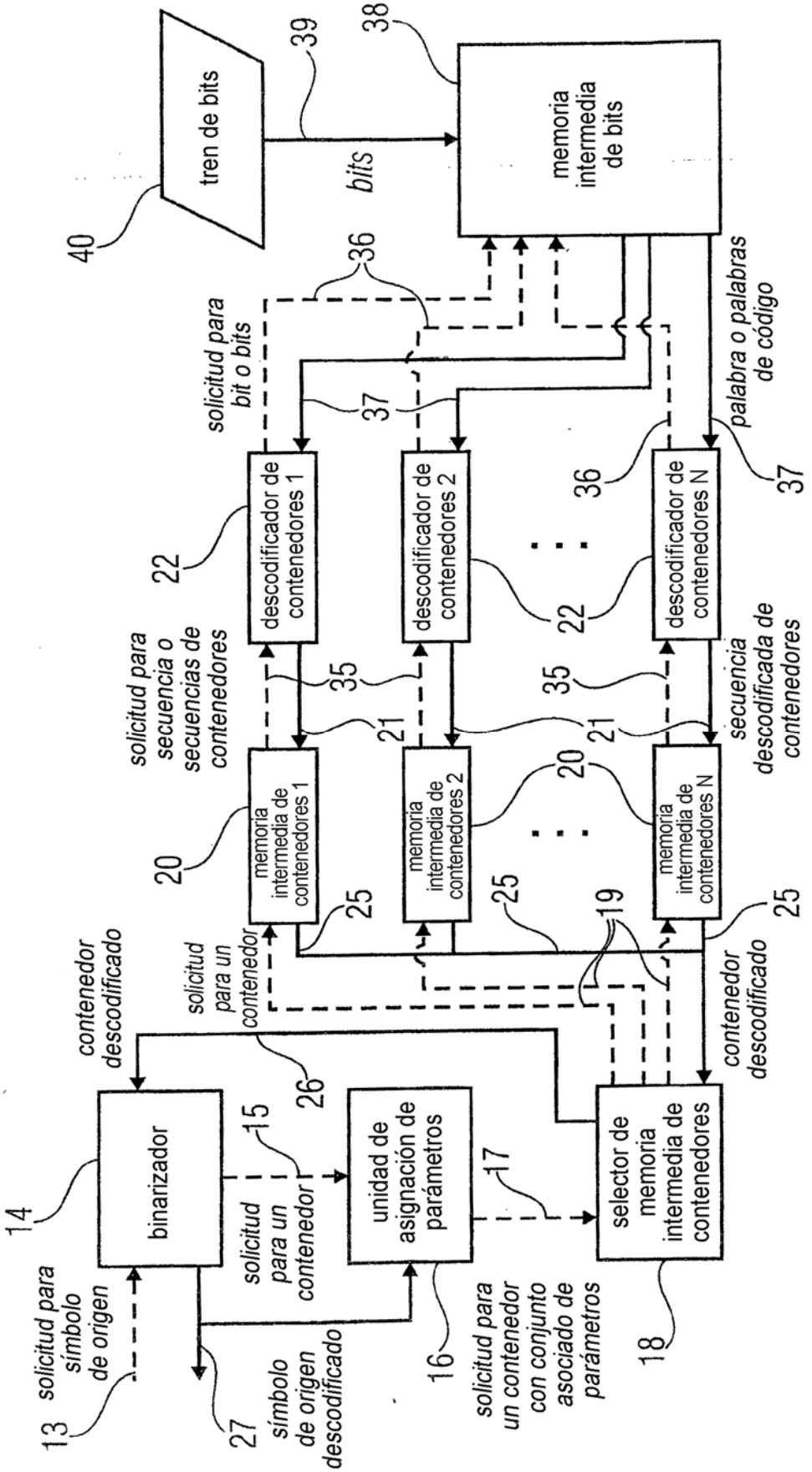

tren de bits
40
39
bits
38
memoria intermedia de bits
36
solicitud para bit o bits
37
palabra o palabras de código
descodificador de contenedores 1
descodificador de contenedores 2
descodificador de contenedores N
22
solicitud para secuencia o secuencias de contenedores
35
21
secuencia descodificada de contenedores
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
20
25
19
solicitud para un contenedor
contenedor descodificado
26
binarizador
14
unidad de asignación de parámetros
15
17
selector de memoria intermedia de contenedores
18
16
solicitud para un contenedor
solicitud para un conjunto asociado de parámetros
13
solicitud para símbolo de origen
27
símbolo de origen descodificado

FIGURA 9

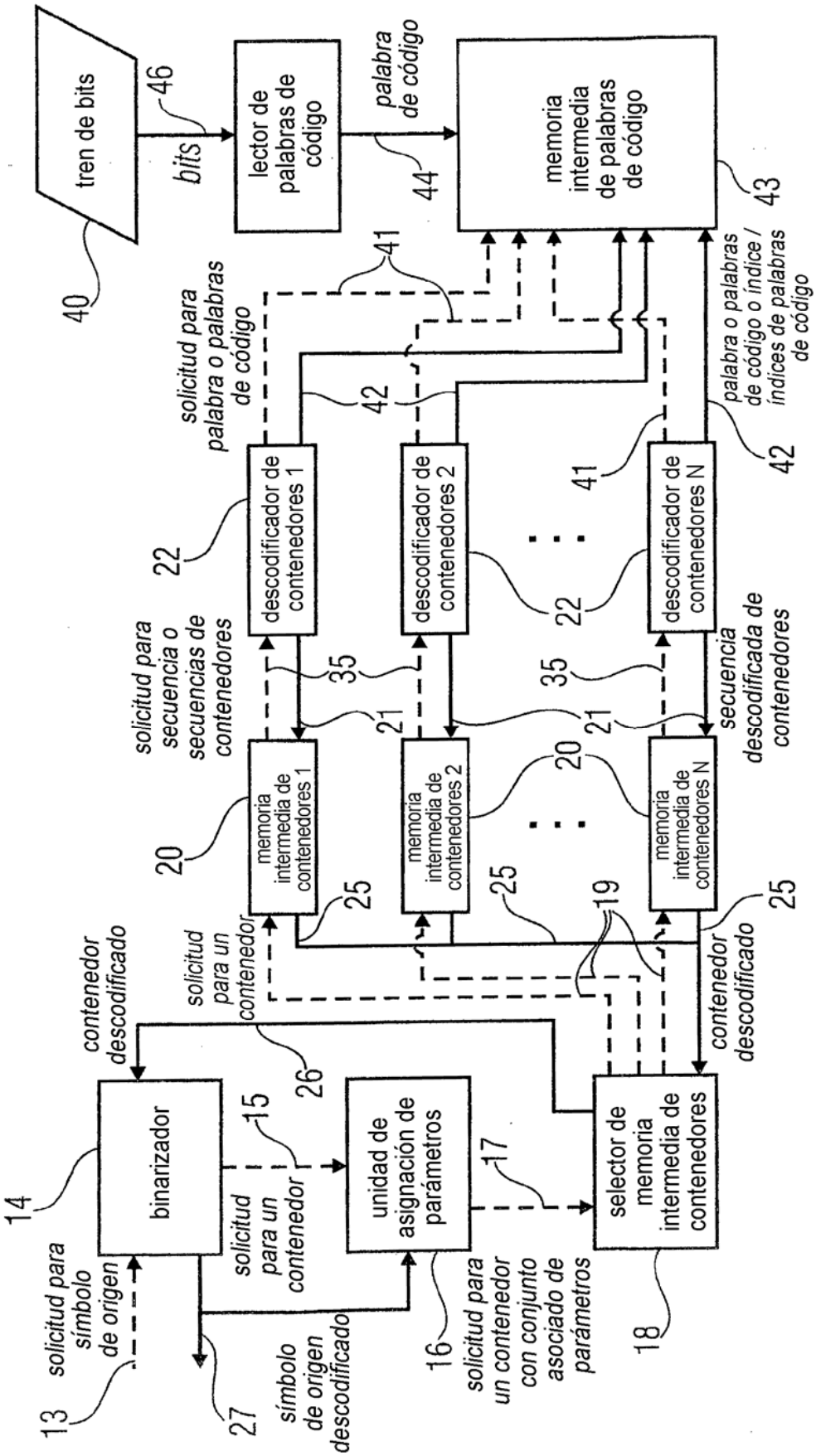

tren de bits
40
46
bits
lector de palabras de código
palabra de código
44
memoria intermedia de palabras de código
43
41
solicitud para palabra o palabras de código
42
palabra o palabras de código o índice / índices de palabras de código
descodificador de contenedores 1
descodificador de contenedores 2
descodificador de contenedores N
22
solicitud para secuencia o secuencias de contenedores
35
21
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
20
25
19
solicitud para un contenedor
26
contenedor descodificado
secuencia descodificada de contenedores
binarizador
14
15
unidad de asignación de parámetros
17
selector de memoria intermedia de contenedores
18
solicitud para símbolo de origen
13
solicitud para un contenedor
27
símbolo de origen descodificado
16
solicitud para un contenedor con conjunto asociado de parámetros

FIGURA 10

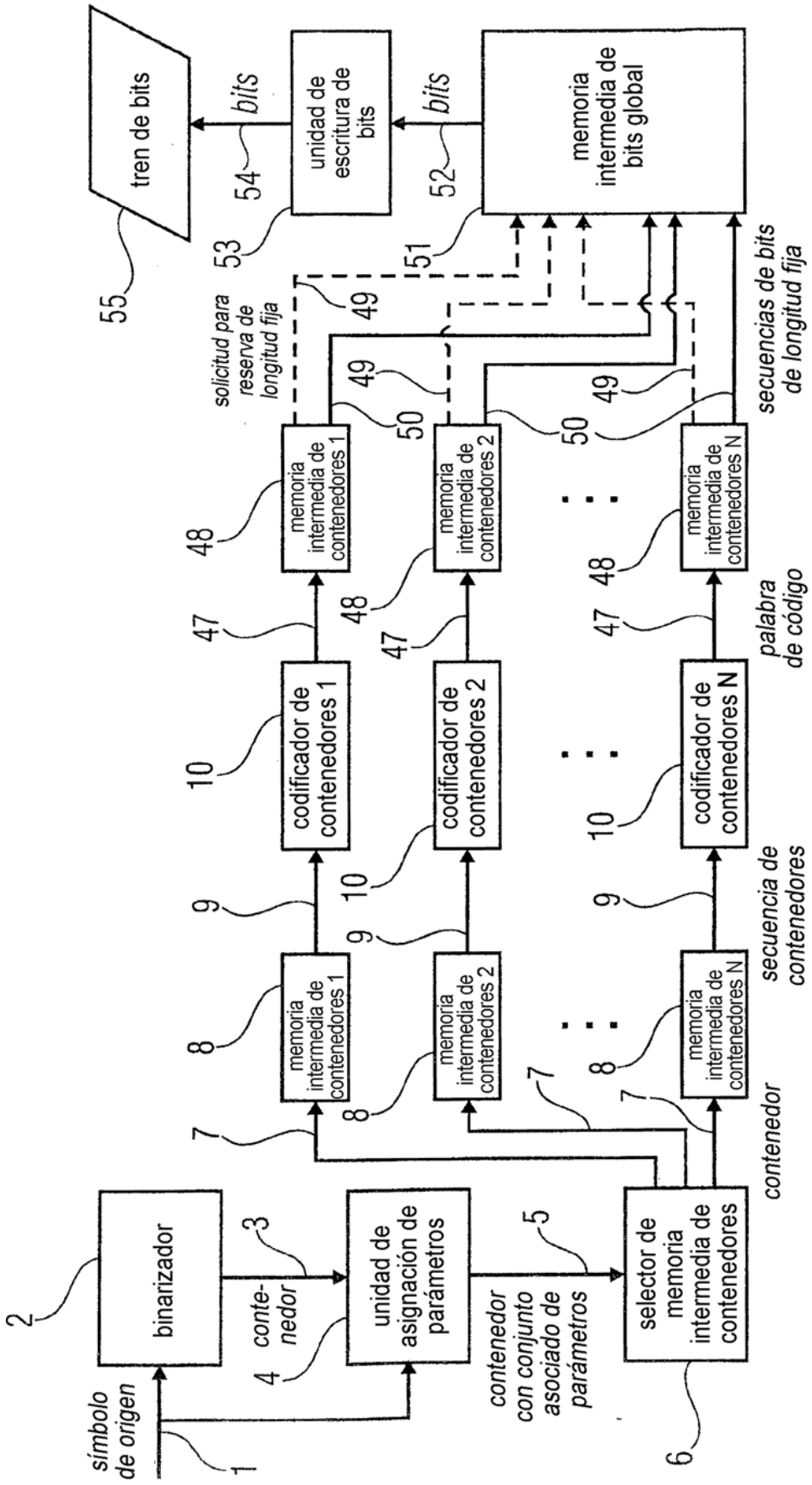

símbolo de origen
1
binarizador
2
conte-nedor
3
4
unidad de asignación de parámetros
5
contenedor con conjunto asociado de parámetros
selector de memoria intermedia de contenedores
6
contenedor
7
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
8
secuencia de contenedores
9
codificador de contenedores 1
codificador de contenedores 2
codificador de contenedores N
10
palabra de código
47
48
49
50
solicitud para reserva de longitud fija
51
memoria intermedia de bits global
52
bits
unidad de escritura de bits
53
54
bits
tren de bits
55
secuencias de bits de longitud fija

FIGURA 11

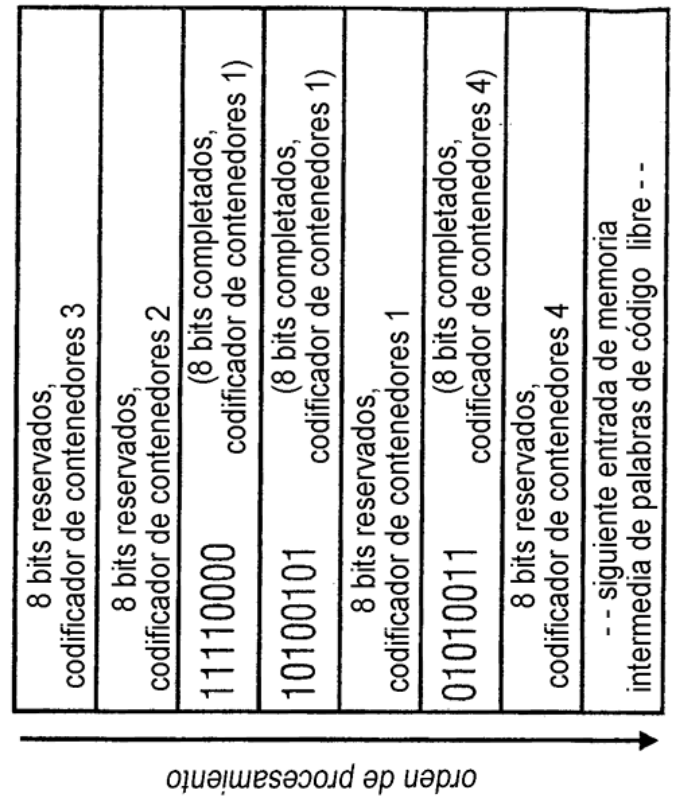


FIGURA 12B

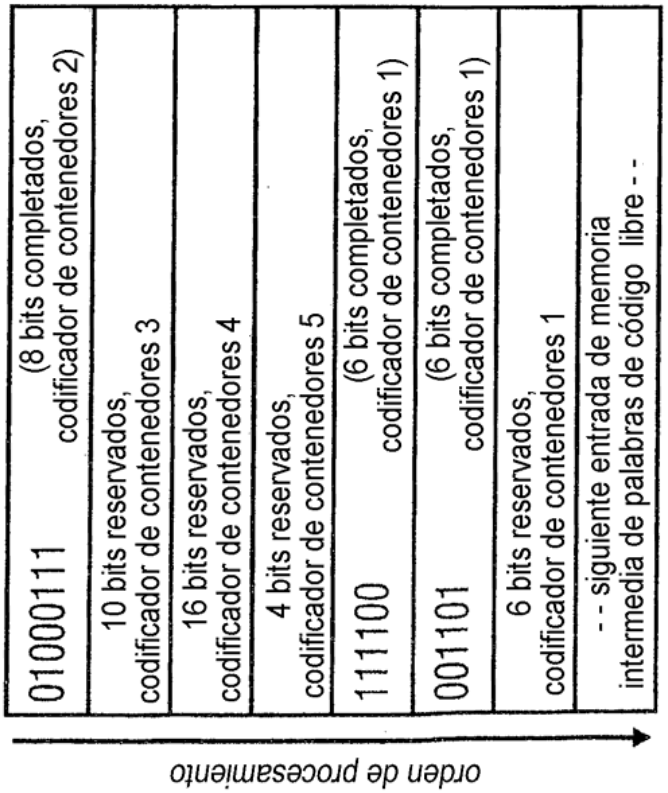


FIGURA 12A

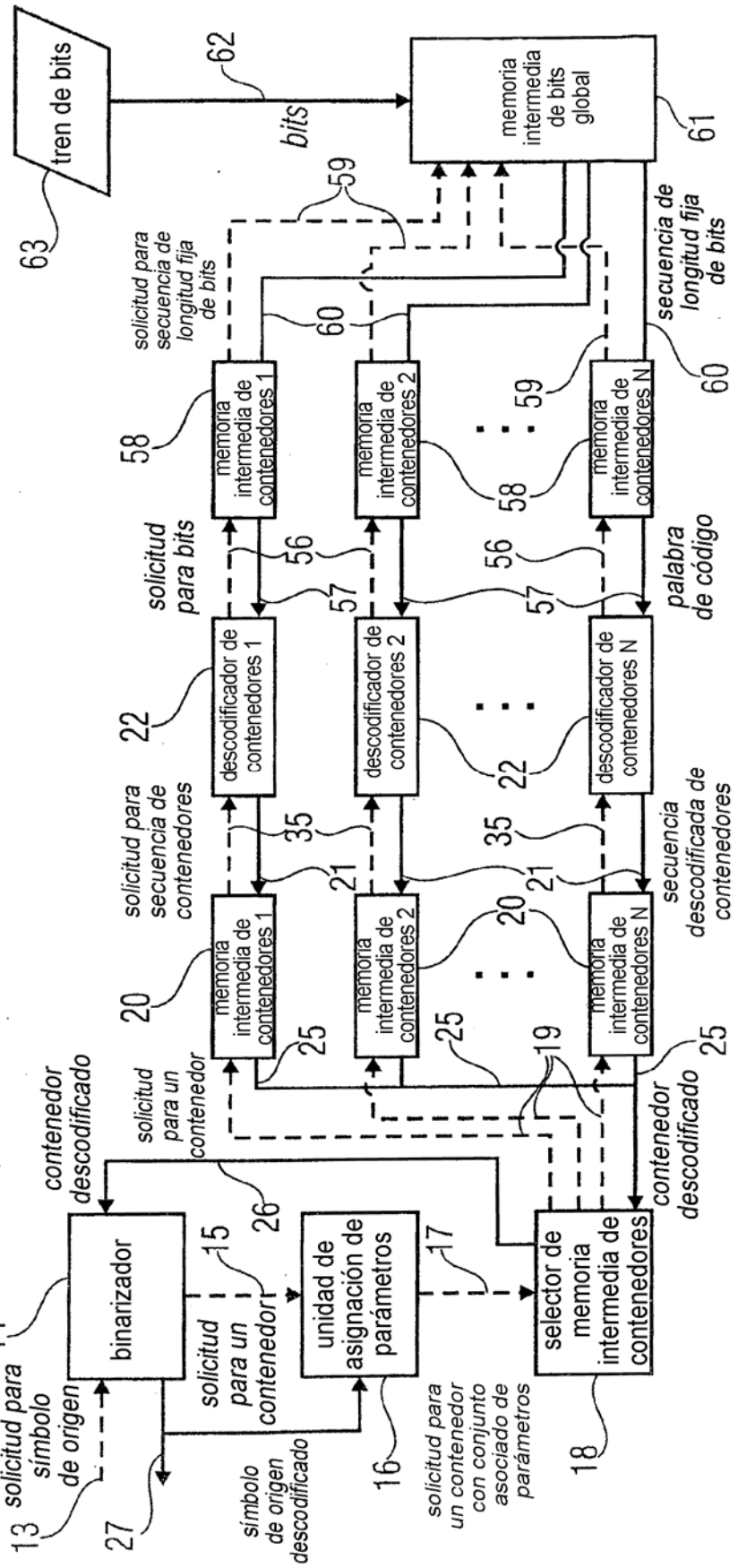

tren de bits
63
bits
62
memoria intermedia de bits global
61
solicitud para secuencia de longitud fija de bits
59
secuencia de longitud fija de bits
60
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
58
solicitud para bits
56
57
palabra de código
descodificador de contenedores 1
descodificador de contenedores 2
descodificador de contenedores N
22
solicitud para secuencia de contenedores
35
21
secuencia descodificada de contenedores
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
20
25
19
solicitud para un contenedor
26
contenedor descodificado
binarizador
14
15
solicitud para un contenedor
unidad de asignación de parámetros
17
selector de memoria intermedia de contenedores
18
solicitud para un contenedor con conjunto asociado de parámetros
16
solicitud para símbolo de origen
13
27
símbolo de origen descodificado

FIGURA 13

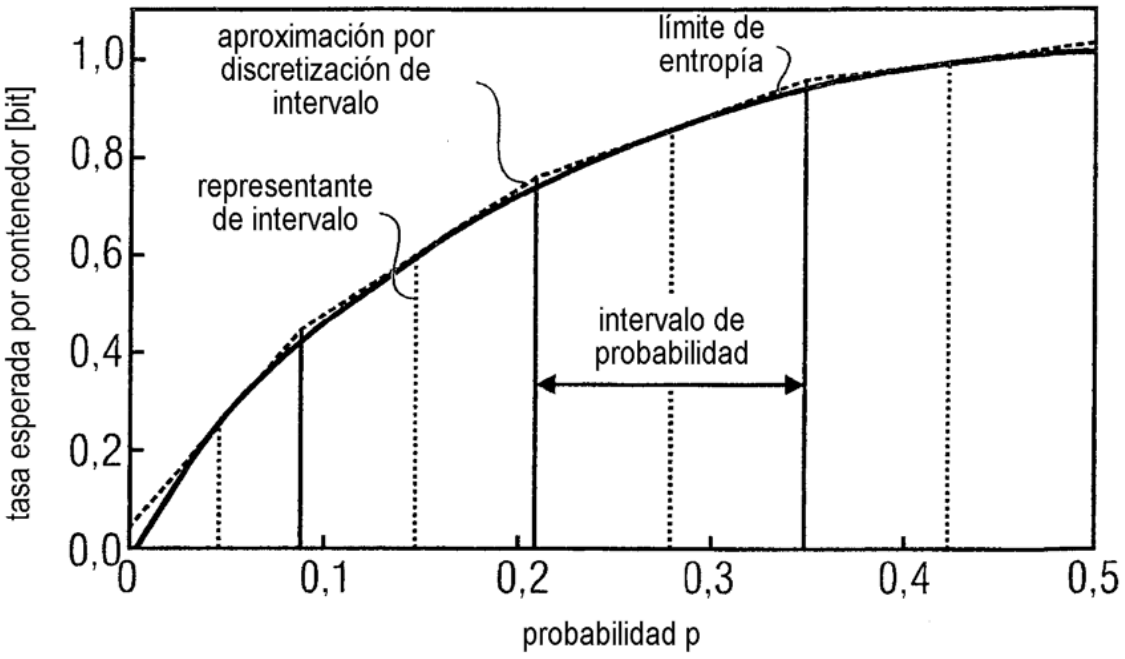


FIGURA 14

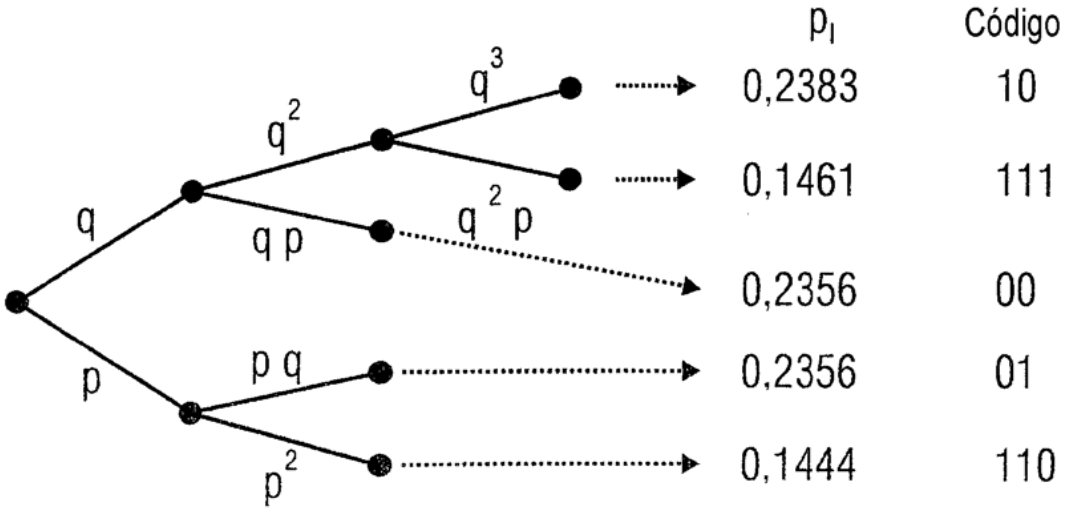


FIGURA 15

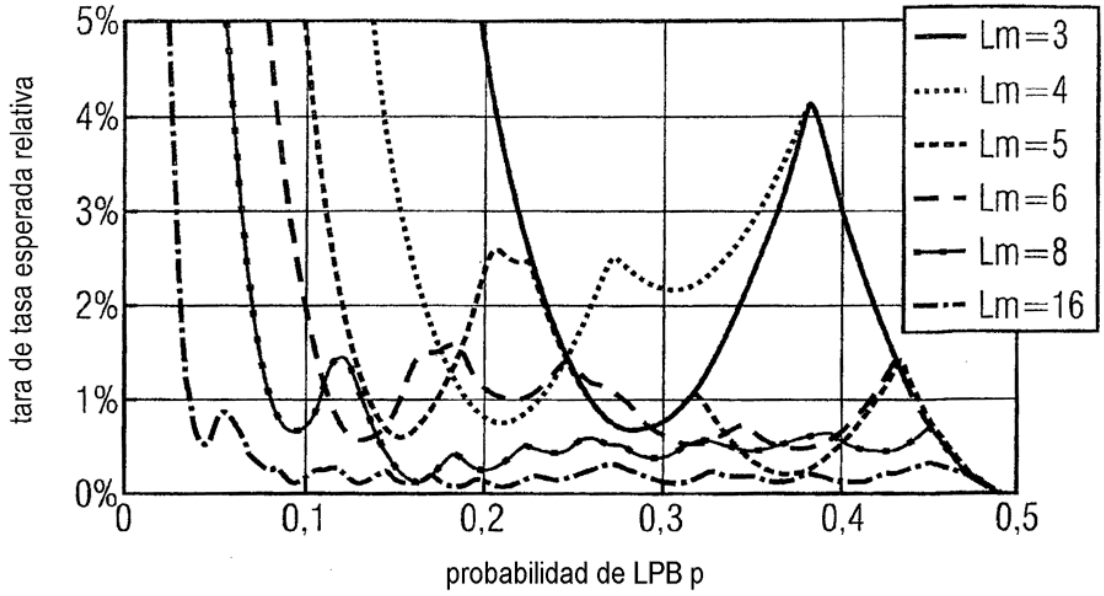


FIGURA 16

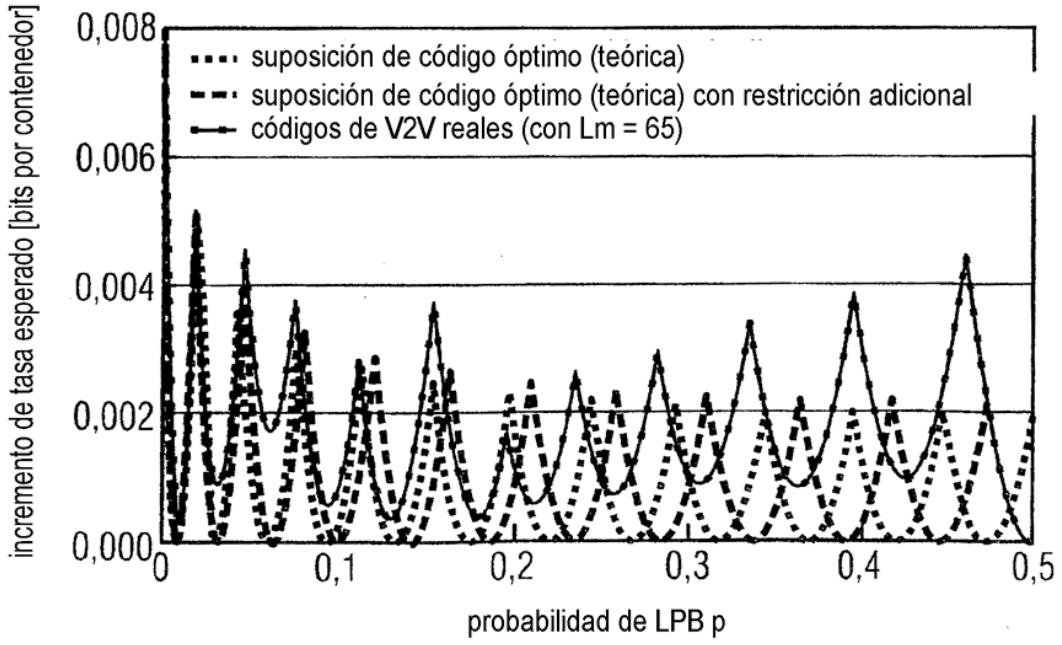
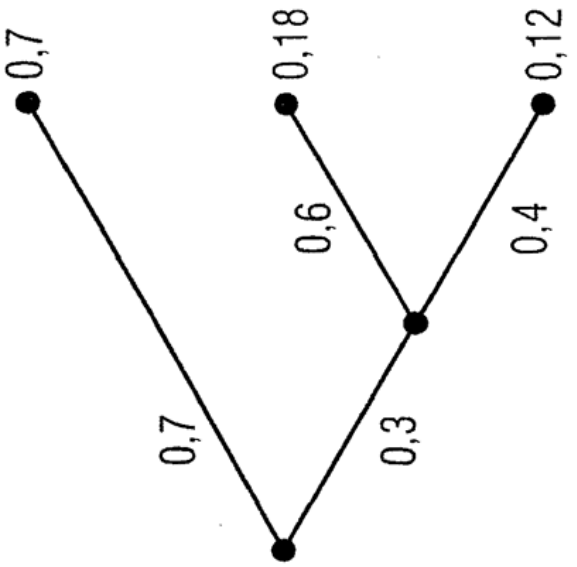


FIGURA 17



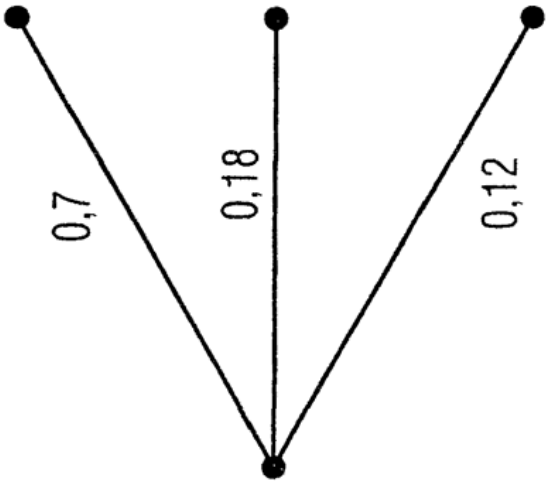


FIGURA 18

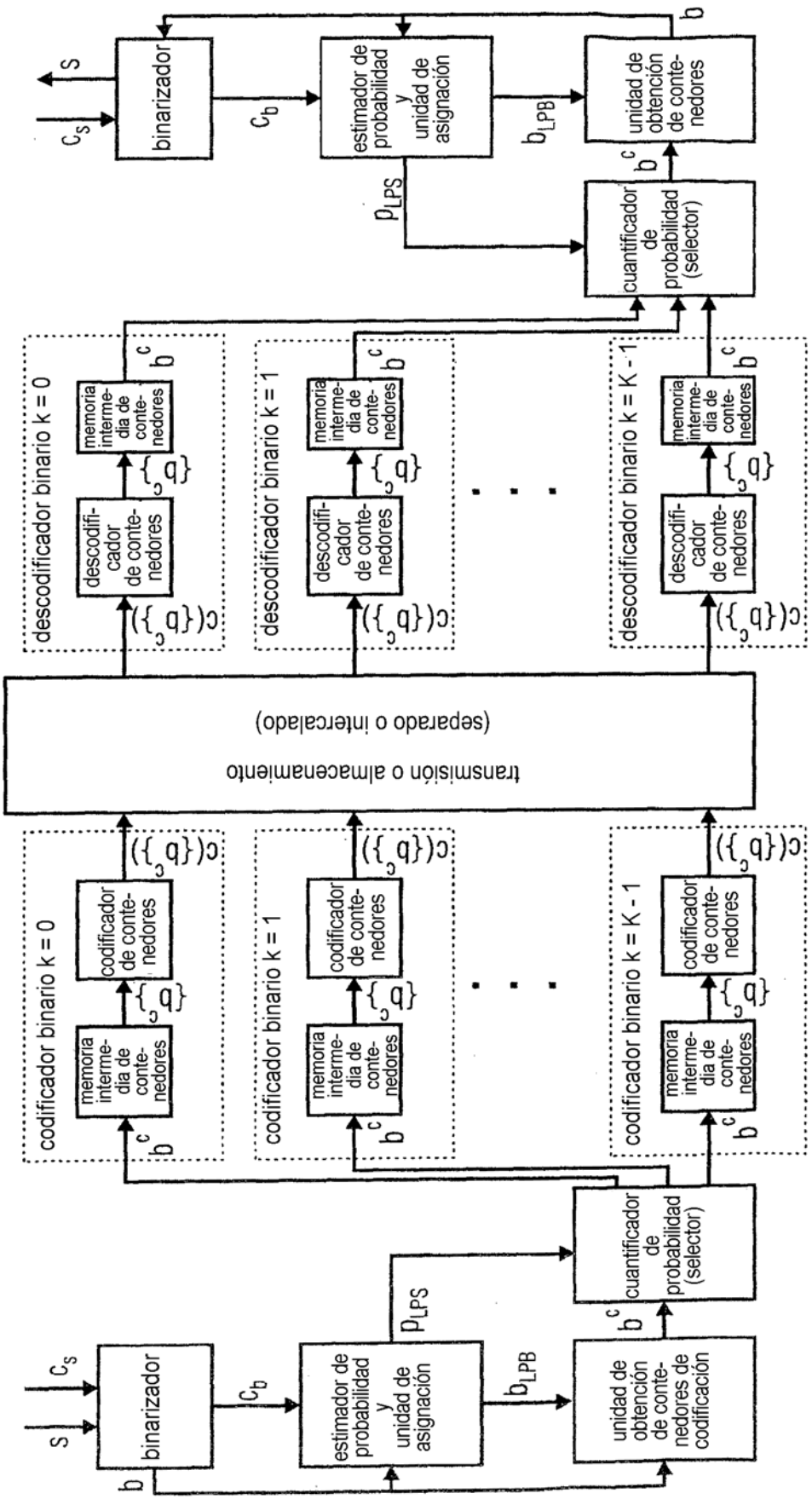

s
c_s
binarizador
b
c_b
estimador de probabilidad y unidad de asignación
p_{LPS}
b_{LPB}
unidad de obtención de contenedores de codificación
b^c
cuantificador de probabilidad (selector)
codificador binario k = 0
codificador binario k = 1
codificador binario k = K - 1
memoria intermedia de contenedores
codificador de contenedores
$\{_cq\}$
$c(\{_cq\})$
transmisión o almacenamiento
(separado o intercalado)
descodificador binario k = 0
descodificador binario k = 1
descodificador binario k = K - 1
descodificador de contenedores
unidad de obtención de contenedores

FIGURA 19

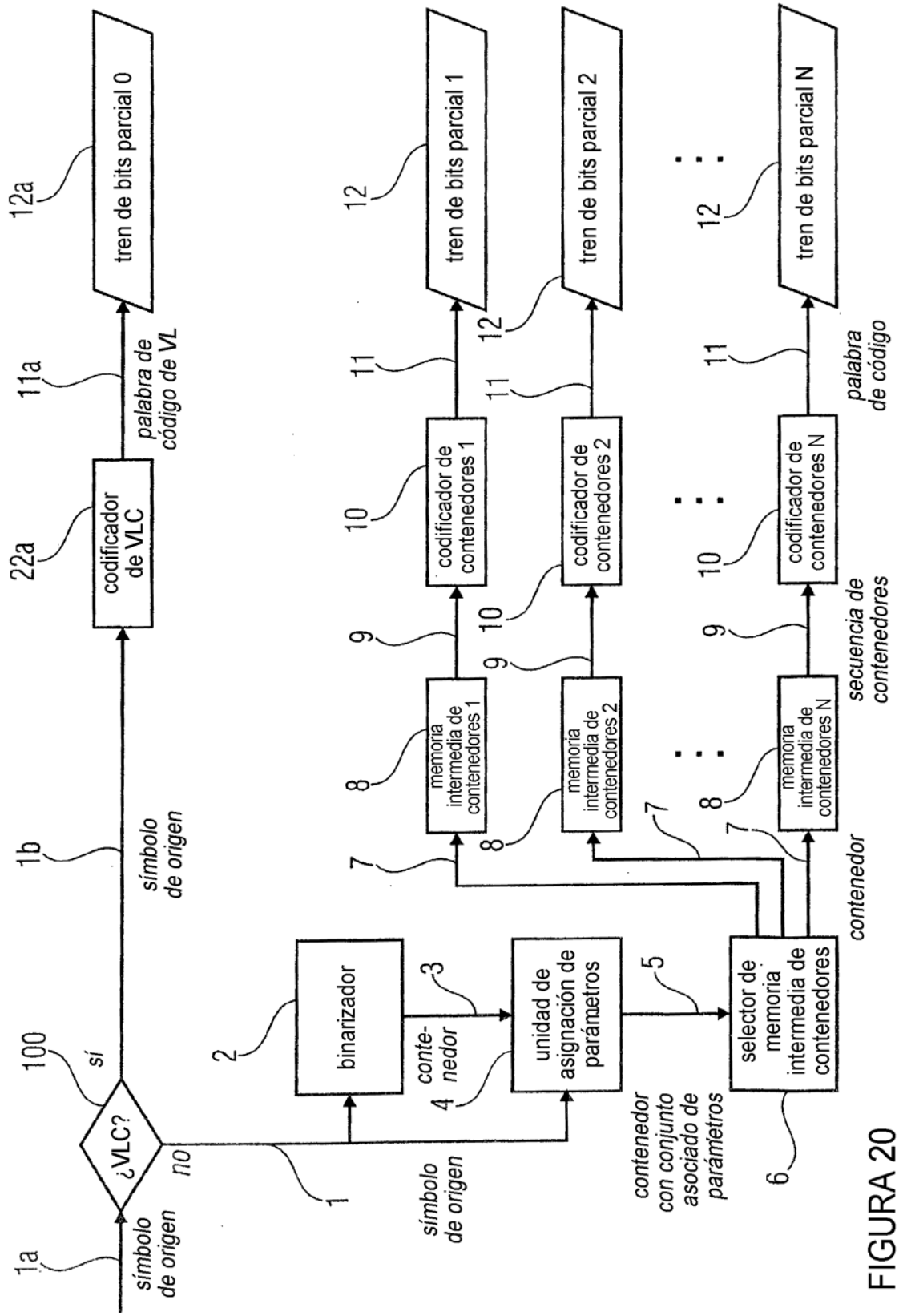

1a
símbolo de origen
¿VLC?
100
sí
no
1b
símbolo de origen
22a
codificador de VLC
11a
palabra de código de VL
12a
tren de bits parcial 0
1
símbolo de origen
2
binarizador
3
conte-nedor
4
5
unidad de asignación de parámetros
contenedor con conjunto asociado de parámetros
6
selector de memoria intermedia de contenedores
contenedor
7
8
memoria intermedia de contenedores 1
9
10
codificador de contenedores 1
11
12
tren de bits parcial 1
memoria intermedia de contenedores 2
codificador de contenedores 2
tren de bits parcial 2
memoria intermedia de contenedores N
codificador de contenedores N
tren de bits parcial N
secuencia de contenedores
palabra de código

FIGURA 20

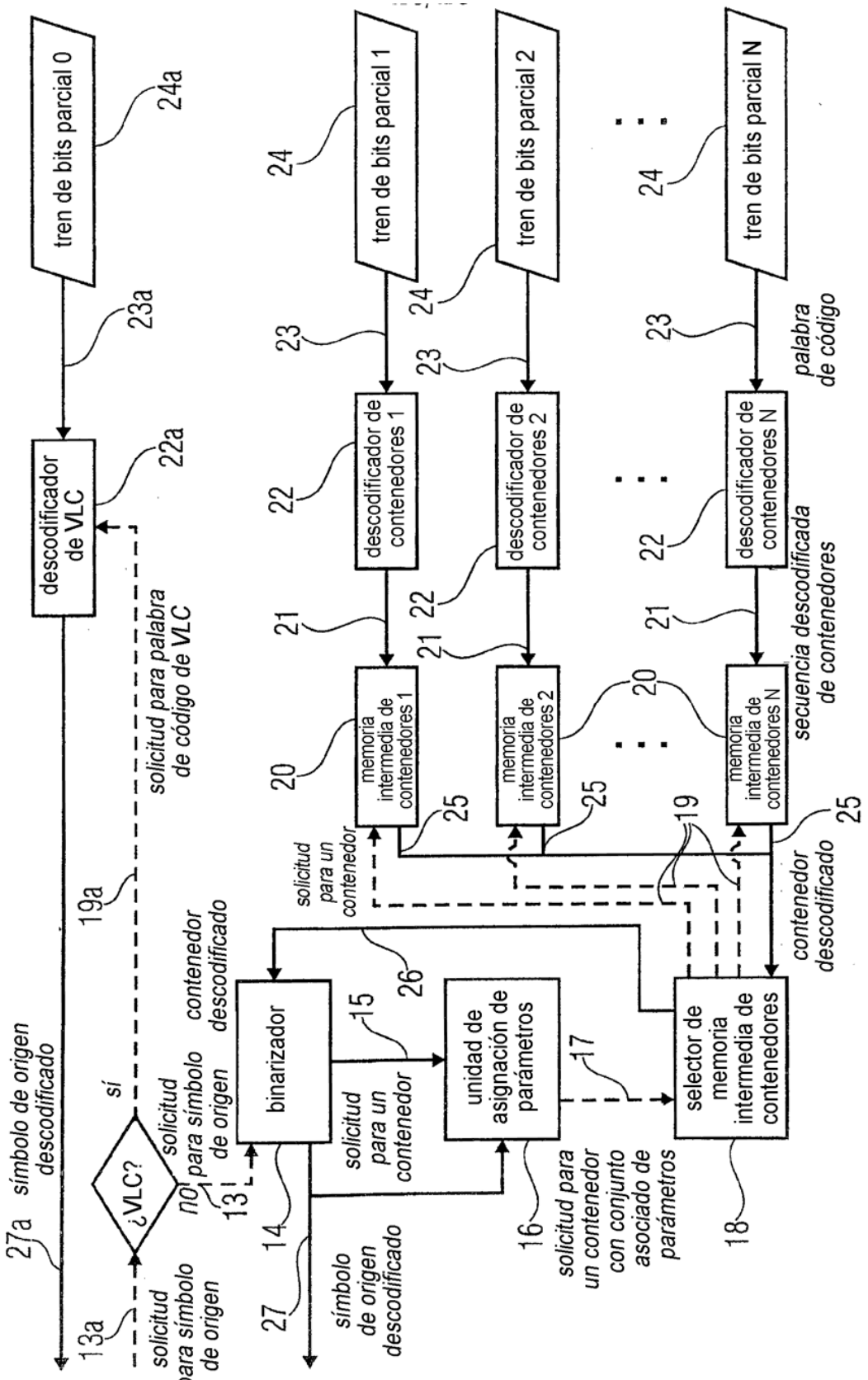

tren de bits parcial 0
tren de bits parcial 1
tren de bits parcial 2
tren de bits parcial N
descodificador de VLC
descodificador de contenedores 1
descodificador de contenedores 2
descodificador de contenedores N
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
binarizador
unidad de asignación de parámetros
selector de memoria intermedia de contenedores
¿VLC?
sí
no
solicitud para símbolo de origen
símbolo de origen descodificado
contenedor descodificado
solicitud para palabra de código de VLC
solicitud para un contenedor
solicitud para un contenedor con conjunto asociado de parámetros
secuencia descodificada de contenedores
palabra de código
24a
23a
22a
19a
27a
13a
27
14
13
15
26
16
17
18
19
20
21
22
23
24
25

FIGURA 21

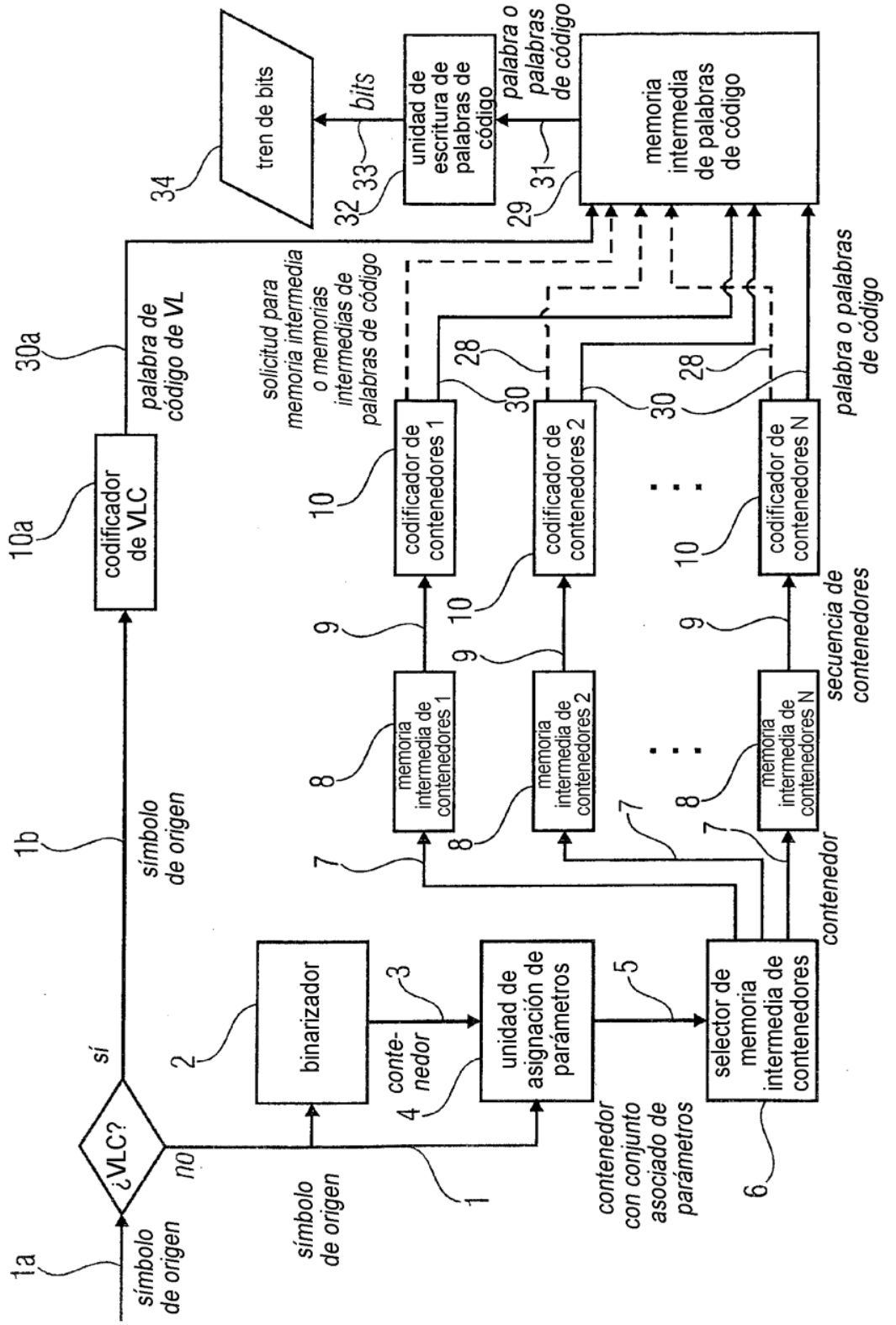

1a
símbolo de origen
¿VLC?
sí
no
1b
símbolo de origen
10a
codificador de VLC
30a
palabra de código de VL
2
binarizador
3
conte-nedor
4
unidad de asignación de parámetros
5
contenedor con conjunto asociado de parámetros
6
selector de memoria intermedia de contenedores
7
8
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
9
10
codificador de contenedores 1
codificador de contenedores 2
codificador de contenedores N
solicitud para memoria intermedia o memorias intermedias de palabras de código
28
30
palabra o palabras de código
secuencia de contenedores
contenedor
29
31
memoria intermedia de palabras de código
32
33
unidad de escritura de palabras de código
bits
34
tren de bits

FIGURA 22

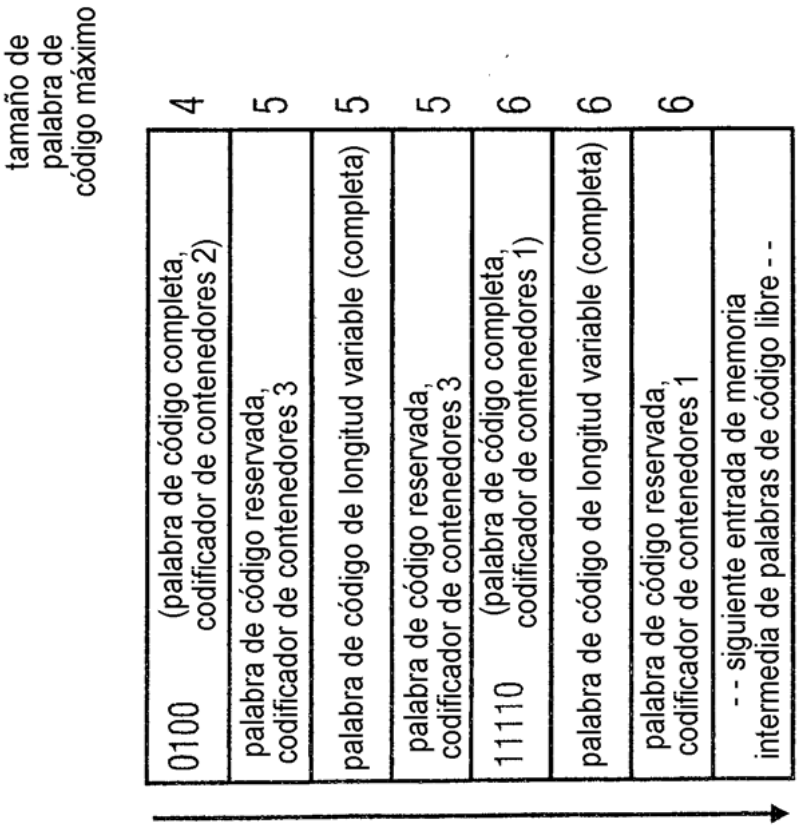


FIGURA 23A

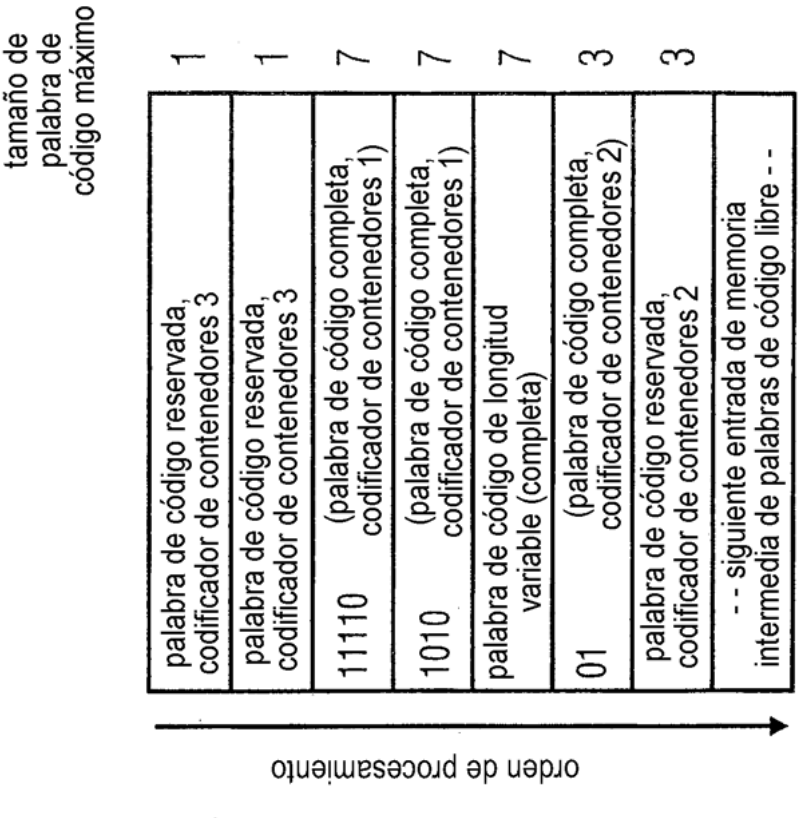


FIGURA 23B

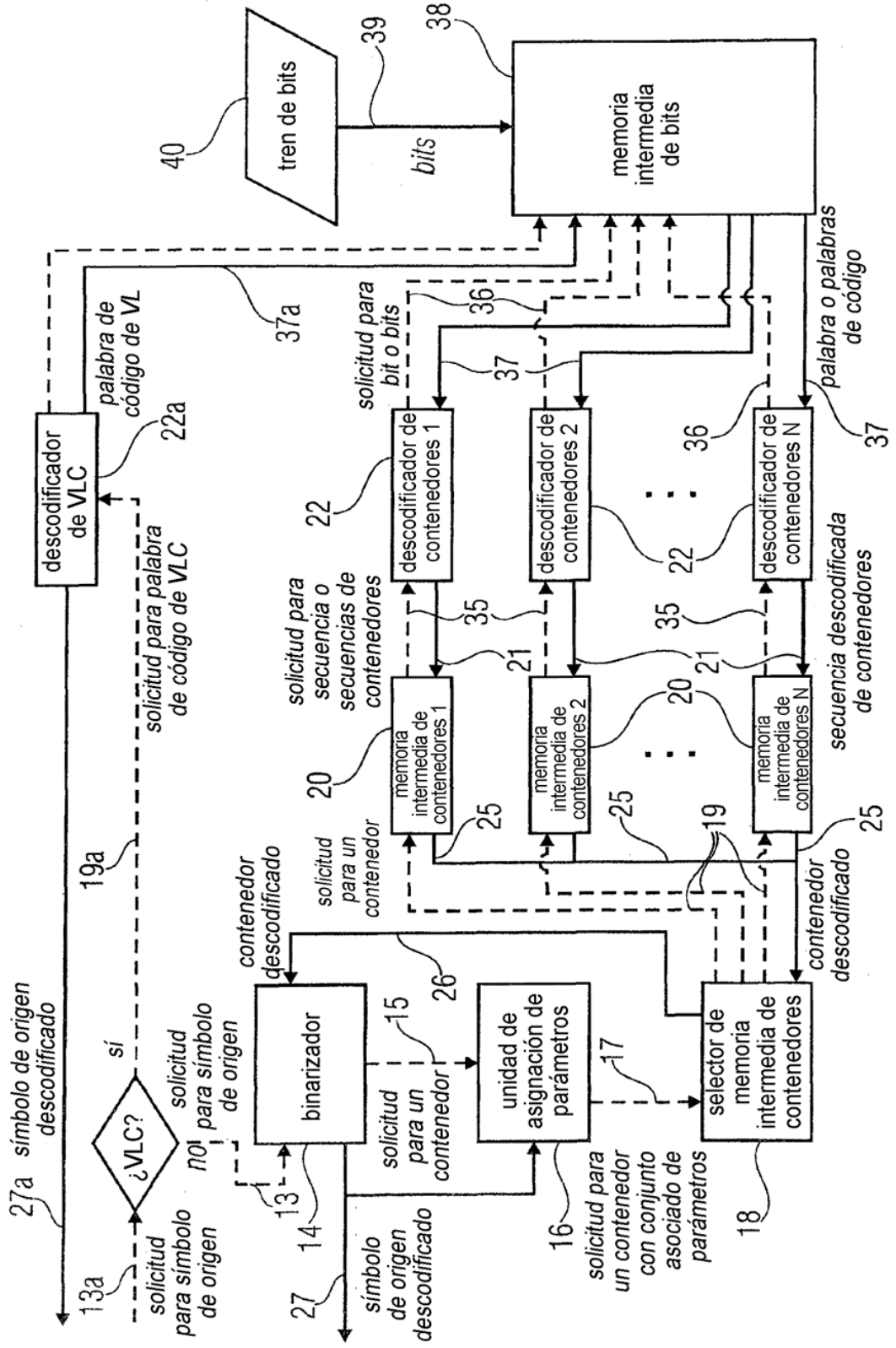

tren de bits
bits
memoria intermedia de bits
descodificador de VLC
palabra de código de VL
solicitud para palabra de código de VLC
solicitud para bit o bits
palabra o palabras de código
descodificador de contenedores 1
descodificador de contenedores 2
descodificador de contenedores N
solicitud para secuencia o secuencias de contenedores
secuencia descodificada de contenedores
memoria intermedia de contenedores 1
memoria intermedia de contenedores 2
memoria intermedia de contenedores N
solicitud para un contenedor
contenedor descodificado
símbolo de origen descodificado
¿VLC?
sí
no
solicitud para símbolo de origen
binarizador
unidad de asignación de parámetros
selector de memoria intermedia de contenedores
solicitud para un contenedor con conjunto asociado de parámetros
símbolo de origen descodificado

FIGURA 24