

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 607 982**

51 Int. Cl.:

H04N 19/13 (2014.01)

H03M 7/40 (2006.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

86 Fecha de presentación y número de la solicitud internacional: **12.01.2012 PCT/EP2012/050431**

87 Fecha y número de publicación internacional: **19.07.2012 WO12095488**

96 Fecha de presentación y número de la solicitud europea: **12.01.2012 E 12700329 (1)**

97 Fecha y número de publicación de la concesión europea: **02.11.2016 EP 2664070**

54 Título: **Esquema de codificación y decodificación entrópica**

30 Prioridad:

14.01.2011 US 201161432884 P

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

05.04.2017

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

ES 2 607 982 T3

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

Esquema de codificación y decodificación entrópica
DESCRIPCIÓN

5 La presente invención se refiere a codificación y decodificación entrópica y puede usarse en aplicaciones tales como, por ejemplo, compresión de video y audio.

10 La codificación entrópica, en general, puede considerarse como la forma más genérica de compresión de datos sin pérdida. La compresión sin pérdida se dirige a representar datos discretos con menos bits que los necesarios para la representación de los datos originales pero sin ninguna pérdida de información. Los datos discretos pueden darse en la forma de texto, gráficos, imágenes, video, audio, voz, facsímil, datos médicos, datos meteorológicos, datos financieros o cualquier otra forma de datos digitales.

15 En la codificación entrópica, las características de alto nivel específicas de la fuente de datos discreta subyacente frecuentemente se desprecia. Consecuentemente, cualquier fuente de datos se considera que se da como una secuencia de símbolos fuente que toman valores en un alfabeto *m-ario* dado y que se caracteriza por una distribución de probabilidad (discreta) correspondiente $\{p_1, \dots, p_m\}$. En estos ajustes abstractos, la delimitación inferior de cualquier método de codificación entrópica en términos de longitud de palabra de código esperada en bits por símbolo es dado por la entropía

$$20 \quad H = - \sum_{i=1}^n p_i \log_2 p_i \quad (A1)$$

25 Los códigos de Huffman y 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 fáciles de construir. La propiedad más atractiva de los códigos de Huffman es que su implementación puede realizarse eficientemente mediante el uso de tablas de código de longitud variable (VLC, del inglés "Variable Length Code"). Sin embargo, cuando se trabaja con estadísticas de origen variable en el tiempo, es decir, probabilidades de símbolos cambiantes, la adaptación del código de Huffman y sus tablas de VLC correspondientes es bastante exigente, tanto en términos de complejidad algorítmica como en términos de costes de implementación. También, en el caso de tener un valor de alfabeto dominante con $p_k > 0,5$, la redundancia del código de Huffman correspondiente (sin el uso de ninguna extensión de alfabeto tal como codificación de tirada) puede ser bastante sustancial. Otros inconvenientes de los códigos de Huffman vienen dados por el hecho de que en caso de tratar con modelizados de probabilidad de orden más alto, pueden requerirse múltiples conjuntos de tablas de VLC. La codificación aritmética, por otro lado, aunque es sustancialmente más compleja que la VLC, ofrece la ventaja de un manejo más consistente y adecuado cuando se trata con modelizados de probabilidad adaptativos y de orden más alto así como con el caso de distribuciones de probabilidad altamente sesgadas. Realmente, esta característica básicamente es el resultado del hecho de que la codificación aritmética proporciona un mecanismo, al menos conceptualmente, para mapear cualquier valor dado de probabilidad estimado en una forma más o menos directa a una parte de la palabra de código resultante. Provista con una interfaz de ese tipo, la codificación aritmética permite una separación limpia entre las tareas de modelizado de probabilidad y estimación de probabilidad, por un lado, y la codificación entrópica real, es decir, mapeados de unos símbolos a palabras de código, por otro lado.

45 Una alternativa a la codificación aritmética y a la codificación VLC es la codificación PIPE. Para ser más preciso, en la codificación PIPE, el intervalo unitario se particiona en un pequeño conjunto de intervalos de probabilidad disjuntos para canalizar el proceso de codificación junto con las estimaciones de probabilidad de variables de símbolos aleatorios. De acuerdo con esta partición, una secuencia de entrada de símbolos fuente discretos con tamaños de alfabeto arbitrarios puede mapearse a una secuencia de símbolos de alfabeto y cada uno de los símbolos de alfabeto se asigna a un intervalo de probabilidad particular que se codifica, a su vez, mediante un proceso de codificación entrópico especialmente dedicado. Estando representado cada uno de los intervalos por una probabilidad fija, el proceso de codificación de entropía de probabilidad intervalo de partición (PIPE, del inglés "Probability Interval Partitioning Entropy") puede basarse en el diseño y aplicación de códigos de longitud simples de variable a variable. El modelizado de probabilidad puede ser fijo o adaptativo. Sin embargo, aunque la codificación PIPE es significativamente menos compleja que la codificación aritmética, tiene aún una complejidad más alta que la codificación VLC.

55 Por lo tanto, sería favorable tener disponible un esquema de codificación entrópico que permita conseguir un compromiso mejor entre la complejidad de codificación por un lado y la eficiencia de compresión por otro lado, incluso cuando se compara con la codificación PIPE que ya combina las ventajas tanto de la codificación aritmética como de la codificación VLC.

60 El documento WO 2008/129021 A2 se refiere a la compresión escalable de secuencias de entramado en 3D consistentes en el tiempo. En relación a la cuantificación y codificación entrópica, el documento describe que los errores de previsión de los vectores del entramado se comprimen componente a componente. En particular, los

componentes se mapean a la cantidad de números enteros, es decir con signo, y se usa un máximo para la cantidad, es decir i_{max} , para definir un intervalo dentro de la cantidad de números enteros para los que los componentes que caen dentro de este intervalo son codificados entrópicamente. La cantidad residual, es decir la distancia al extremo más próximo del intervalo, se codifica usando códigos Golomb.

5 Adicionalmente, en general, sería favorable tener disponible un esquema de codificación entrópico que permita conseguir una mejor eficiencia de compresión *per se*, con una complejidad de codificación moderada.

10 Es un objeto de la presente invención proporcionar un concepto de codificación entrópico que satisfaga la demanda anteriormente identificada, es decir permita conseguir un mejor compromiso entre la complejidad de codificación por un lado y la eficiencia de compresión por otro lado.

Este objeto se consigue por la materia objeto de las reivindicaciones independientes.

15 La presente invención se basa en la idea de la descomposición de un intervalo de valores de los elementos de sintaxis respectivos en una secuencia de n particiones con codificación de los componentes de los z valores del elemento de sintaxis que caen dentro de las particiones respectivas por separado con al menos una por codificación VLC y con al menos una por codificación aritmética. En consecuencia, de acuerdo con las realizaciones de la presente invención, los elementos de sintaxis se descomponen en un número respectivo de n símbolos fuente s_i ,
 20 siendo $i=1\dots n$, dependiendo el número n respectivo de símbolos fuente de en cuál de una secuencia de n particiones (140_{1-3}) dentro de la que se subdivide un intervalo de valores de los elementos de sintaxis respectivos, cae dentro un valor z de los elementos de sintaxis respectivos, de modo que una suma de valores del número respectivo de símbolos fuente s_i conduce a z , y, si $n>1$, para todo $i=1\dots n-1$, el valor de s_i corresponde a un intervalo de la partición i -ésima.

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

30 Se describen a continuación realizaciones preferidas de la presente invención con respecto a las figuras. Esta realización representa ejemplos, en la medida en que no usan codificación aritmética próxima a la codificación VLC. Entre las figuras,

La Fig. 1a muestra un diagrama de bloques de un aparato de codificación entrópica;

35 La Fig. 1b muestra un diagrama esquemático que ilustra una descomposición posible de elementos de sintaxis en símbolos fuente;

La Fig. 1c muestra un diagrama de flujo que ilustra un modo posible de funcionamiento del descomponedor de la Fig. 1a en la descomposición de elementos de sintaxis en símbolos fuente;

40 La Fig. 2a muestra un diagrama de bloques de un aparato de decodificación entrópica;

La Fig. 2b muestra un diagrama de flujo que ilustra un modo posible de funcionamiento del componedor de la Fig. 2a en la composición de elementos de sintaxis a partir de símbolos fuente;

45 La Fig. 3 muestra un diagrama de bloques de un codificador PIPE de acuerdo con una realización de comparación que puede usarse en la Fig. 1;

50 La Fig. 4 muestra un diagrama de bloques de un decodificador PIPE adecuado para la decodificación de un flujo de bits generado por el codificador PIPE de la Fig. 3, de acuerdo con una realización de comparación, que puede usarse en la Fig. 2;

La Fig. 5 muestra un diagrama esquemático que ilustra un paquete de datos con flujos de bits parciales multiplexados;

55 La Fig. 6 muestra un diagrama esquemático que ilustra un paquete de datos con una segmentación alternativa usando segmentos de tamaño fijo;

La Fig. 7 muestra un diagrama de bloques de un codificador PIPE que usa intercalado de flujos de bits parcial;

60 La Fig. 8 muestra ejemplos de ilustración esquemática para el estado de una memoria intermedia de palabras de código en el lado del codificador de la Fig. 7;

La Fig. 9 muestra un diagrama de bloques de un decodificador PIPE que usa un intercalado de flujos de bits parcial;

- La Fig. 10 muestra un diagrama de bloques de un decodificador PIPE que usa un intercalado de palabras de código usando un conjunto simple de palabras de código;
- 5 La Fig. 11 muestra un diagrama de bloques de un codificador PIPE que usa intercalado de secuencias de bits de longitud fija;
- La Fig. 12 muestra un esquema que ilustra ejemplos para el estado de una memoria intermedia de bits global en el lado del codificador de la Fig. 11;
- 10 La Fig. 13 muestra un diagrama de bloques de un decodificador PIPE que usa intercalado de secuencias de bits de longitud fija;
- La Fig. 14 muestra un gráfico para la ilustración de un intervalo de probabilidad óptimo esquematizado en $K = 4$ intervalos suponiendo una distribución de probabilidad uniforme en $(0, 0,5]$;
- 15 La Fig. 15 muestra un diagrama esquemático que ilustra un árbol de eventos binarios para una probabilidad LPB de $P = 0,38$ y un código de longitud variable asociado obtenido mediante el algoritmo de Huffman;
- 20 La Fig. 16 muestra un gráfico a partir del que pueden recogerse el incremento relativo en la tasa de bits $\rho(p, C)$ para códigos C óptimos dado un número máximo de entradas de tabla L_m ;
- La Fig. 17 muestra un gráfico que ilustra el incremento de la tasa para la partición del intervalo de probabilidad teóricamente óptimo en $K = 12$ intervalos y un diseño real con $V2V$ códigos con un número máximo de entradas de tabla $L_m = 65$;
- 25 La Fig. 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;
- 30 La Fig. 19 muestra un diagrama de bloques de un sistema que comprende un codificador (parte izquierda) y decodificador (parte derecha);
- La Fig. 20 muestra un diagrama de bloques de un aparato de codificación entrópica;
- 35 La Fig. 21 muestra un diagrama de bloques de un aparato de decodificación entrópica;
- La Fig. 22 muestra un diagrama de bloques de un aparato de codificación entrópica;
- La Fig. 23 muestra un diagrama esquemático que ilustra ejemplos del estado de una memoria intermedia de bits global en el lado del codificador de la Fig. 22;
- 40 La Fig. 24 muestra un diagrama de bloques de un aparato de decodificación entrópica.

45 Antes de que se describan varias realizaciones de la presente solicitud en lo que sigue con respeto a las figuras, se observa que se usan iguales signos de referencia a todo lo largo de las figuras para indicar elementos iguales o equivalentes en estas figuras, y la descripción de estos elementos presentada en cualquiera de las figuras previas puede aplicarse también a cualquiera de las figuras siguientes siempre que la descripción previa no entre en conflicto con la descripción de las figuras actuales.

50 La Fig. 1a muestra un aparato de codificación entrópica. El aparato comprende un subdivisor 100, un codificador VLC 102 y un codificador PIPE 104.

El subdivisor 100 se configura para subdividir una secuencia de símbolos fuente 106 en una primera subsecuencia 108 de símbolos fuente y una segunda subsecuencia 110 de símbolos fuente. El codificador VLC 102 tiene una entrada del mismo conectada a una primera salida del subdivisor 100 y se configura para convertir acertadamente con relación a símbolos, los símbolos fuente de la primera subsecuencia 108 en palabras de código que forman un primer flujo de bits 112. El codificador VLC 102 puede comprender una tabla de búsqueda y usar, individualmente, los símbolos fuente como un índice para la búsqueda, por símbolo fuente, de una palabra de código respectiva en la tabla de búsqueda. El codificador VLC produce la última palabra de código, y prosigue con el siguiente símbolo fuente en la subsecuencia 110 para producir una secuencia de palabras de código en la que cada palabra de código se asocia con exactamente uno de los símbolos fuente dentro de la subsecuencia 110. Las palabras de código pueden tener longitudes diferentes y pueden definirse de modo que ninguna palabra de código forme un prefijo con cualquiera de las otras palabras de código. Adicionalmente, la tabla de búsqueda puede ser estática.

55

60

El codificador PIPE 104 tiene una entrada del mismo conectada a una segunda salida del subdivisor 100 y se configura para codificar la segunda subsecuencia 110 de símbolos fuente, representados en la forma de una secuencia de símbolos de alfabeto, y comprende un asignador 114 configurado para asignar una medida para una estimación de una distribución de probabilidad entre valores posibles que pueden asumir los símbolos de alfabeto respectivos, a cada símbolo de alfabeto de la secuencia de símbolos de alfabeto basándose en la información contenida dentro de los símbolos de alfabeto previos de la secuencia de símbolos de alfabeto, una pluralidad de codificadores entrópicos 116 cada uno de los cuales se configura para convertir los símbolos de alfabeto dirigidos al codificador entrópico respectivo en un segundo flujo de bits 118 respectivo, y un selector 120 configurado para dirigir cada símbolo de alfabeto de la segunda subsecuencia 110 a uno seleccionado de la pluralidad de codificadores entrópicos 116, dependiendo la selección de la medición mencionada anteriormente para la estimación de la distribución de probabilidad asignada al símbolo de alfabeto respectivo. La asociación entre símbolos fuente y símbolos de alfabeto puede ser tal que cada símbolo de alfabeto se asocie únicamente con exactamente un símbolo fuente de la subsecuencia 110 para representar, posiblemente junto con símbolos de alfabeto adicionales de la secuencia de símbolos de alfabeto que pueden seguirse inmediatamente entre sí, este símbolo fuente.

Como se describe más en detalle a continuación, la secuencia 106 de símbolos fuente puede ser una secuencia de elementos de sintaxis de un flujo de bits que puede analizarse. El flujo de bits que puede analizarse puede, por ejemplo, representar contenido de video y/o audio en una forma escalable o no escalable representando los elementos de sintaxis, por ejemplo, niveles del coeficiente de transformada, vectores de movimiento, índices de referencia de imagen en movimiento, factores de escala, valores de energía de la envolvente de audio o similares. Los elementos de sintaxis pueden, en particular, ser de tipo o categoría diferente siendo los elementos de sintaxis del mismo tipo, por ejemplo, teniendo el mismo significado dentro del flujo de bits que puede analizarse pero con respecto a diferentes partes del mismo, tal como diferentes imágenes, diferentes macrobloques, diferentes componentes espectrales o similares, mientras que los elementos de sintaxis de tipo diferente pueden tener un significado diferente dentro del flujo de bits, tal como un vector de movimiento tiene un significado diferente que un elemento de sintaxis que represente un nivel de coeficiente de transformada que represente la predicción de movimiento residual.

El subdivisor 100 puede configurarse para realizar la subdivisión dependiendo del tipo de elementos de sintaxis. Esto es, el subdivisor 100 puede enviar elementos de sintaxis de un primer grupo de tipos a la primera subsecuencia 108 y enviar elementos de sintaxis de un segundo grupo de tipos distintos del primer grupo, a la segunda subsecuencia 110. La subdivisión realizada por el subdivisor 100 puede diseñarse de modo que las estadísticas de símbolos de los elementos de sintaxis dentro de la subsecuencia 108 sea adecuada para ser codificada por VLC por el codificador VLC 102, es decir resulte en casi una mínima entropía posible a pesar del uso de la codificación VLC y su restricción con relación a su adecuación para ciertas estadísticas de símbolo tal como se describe en la parte introductoria de la memoria de la presente solicitud. Por otro lado, el subdivisor 100 puede enviar todos los otros elementos de sintaxis en la segunda subsecuencia 110 de modo que estos elementos de sintaxis que tienen estadísticas de símbolos que no son adecuadas para la codificación VLC, se codifican mediante un codificador PIPE 104 más complejo, pero más eficiente —en términos de la relación de compresión—.

Como es también el caso con mayor detalle con respecto a las figuras siguientes, el codificador PIPE 104 puede comprender un simbolizador 122 configurado para mapear individualmente cada elemento de sintaxis de la segunda subsecuencia 110 en una secuencia parcial respectiva de símbolos de alfabeto, junto con la formación de la secuencia 124 anteriormente mencionada de símbolos de alfabeto. En otras palabras, el simbolizador 122 puede no estar presente si, por ejemplo, el símbolo fuente de la subsecuencia 110 está representado como secuencias parciales respectivas de símbolos de alfabeto. El simbolizador 122 es, por ejemplo, ventajoso en caso de que los símbolos fuente dentro de la subsecuencia 110 sean de diferentes alfabetos, y especialmente, alfabetos que tengan diferentes números de símbolos de alfabeto posibles. Concretamente, en este caso, el simbolizador 122 es capaz de armonizar los alfabetos de los símbolos que llegan dentro del subflujo 110. El simbolizador 122 puede, por ejemplo, realizarse como un binarizador configurado para binarizar los símbolos que llegan dentro de la subsecuencia 110.

Como se ha mencionado anteriormente, los elementos de sintaxis pueden ser de tipo diferente. Esto puede ser verdad también para los elementos de sintaxis dentro del subflujo 110. El simbolizador 122 puede configurarse entonces para realizar el mapeado individual de los elementos de sintaxis de la subsecuencia 110 usando un esquema de mapeado de simbolización, tal como un esquema de binarización, diferente para elementos de sintaxis de tipo diferente. Ejemplos para esquemas de binarización específicos se presentan en la descripción siguiente, tal como un esquema de binarización unario, un esquema de binarización exp-Golomb de orden 0 u orden 1, por ejemplo, o un esquema de binarización unario truncado, un esquema de binarización truncado y reordenado exp-Golomb de orden 0 o un esquema de binarización no sistemático.

En consecuencia, los codificadores entrópicos 116 podrían configurarse para funcionar sobre un alfabeto binario. Finalmente, debería tomarse nota que el simbolizador 122 puede considerarse como parte del codificador PIPE 104 en sí tal como se muestra en la Fig. 1a. Alternativamente, sin embargo, el binarizador puede considerarse como externo al codificador PIPE.

De modo similar a la última nota, debería tomarse nota que el asignador 114, aunque se muestra conectado en serie entre el simbolizador 122 y el selector 120, puede considerarse alternativamente como conectado entre una salida del simbolizador 124 y una primera entrada del selector 120, estando conectada una salida del asignador 114 a otra entrada del selector 120 como se describe posteriormente con respecto a la Fig. 3. En efecto, el asignador 114 acompaña cada símbolo de alfabeto con la medida anteriormente mencionada para una estimación de la distribución de probabilidad.

En lo que se refiere a la salida del aparato de codificación entrópica de la Fig. 1a, la misma se compone de un primer flujo de bits 112 producido por el codificador VLC 102 y la pluralidad de segundos flujos de bits 118 producidos por la pluralidad de codificadores entrópicos 116. Como se describe adicionalmente a continuación, todos estos flujos de bits pueden transmitirse en paralelo. Alternativamente, los mismos pueden intercalarse en un flujo de bits 126 común mediante el uso de un intercalador 128. Las Figs. 22 a 24 muestran ejemplos de un tal intercalado de flujos de bits. Como se muestra adicionalmente en la Fig. 1, el codificador PIPE 104 en sí puede incluir su propio intercalador 130 para intercalar la pluralidad de segundos flujos de bits 118 en un flujo de bits 132 codificado PIPE común. Las posibilidades para dicho intercalador 130 pueden deducirse de la descripción de las Figs. 5 a 13. El flujo de bits 132 y el flujo de bits 112 pueden, en una configuración paralela, representar la salida del aparato de codificación entrópica de la Fig. 1a. Alternativamente, otro intercalador 134 puede intercalar ambos flujos de bits en cuyo caso el intercalador 130 y 134 formarían dos etapas de un intercalador 128 en dos etapas.

Como se ha descrito anteriormente, el subdivisor 100 puede realizar la subdivisión en elementos de sintaxis, es decir, los símbolos fuente sobre los que opera el subdivisor 100 pueden ser elementos de sintaxis completos, o dicho de otro modo, el subdivisor 100 puede operar en unidades de elementos de sintaxis.

Sin embargo, el aparato de codificación entrópica de la Fig. 1a puede incluir el descomponedor 136 para descomponer elementos de sintaxis dentro de un flujo de bits 138 que puede analizarse individualmente en uno o más símbolos fuente de la secuencia de símbolos fuente 106 que entra en el subdivisor 100.

En particular, el descomponedor 136 puede configurarse para convertir la secuencia 138 de elementos de sintaxis en la secuencia 106 de símbolos fuente mediante la descomposición individualmente de cada elemento de sintaxis en un número entero respectivo de símbolos fuente. El número entero puede variar entre los elementos de sintaxis. En particular, alguno de los elementos de sintaxis puede incluso dejarse sin cambiar por el descomponedor 136, mientras que otros elementos de sintaxis se descomponen exactamente en dos, o al menos dos, símbolos fuente. El subdivisor 100 puede configurarse para enviar uno de los símbolos fuente de dichos elementos de sintaxis descompuestos a la primera subsecuencia 108 de los símbolos fuente y otro de los símbolos fuente del mismo elemento de sintaxis descompuesto a la segunda subsecuencia 110 de símbolos fuente. Como se ha mencionado anteriormente, los elementos de sintaxis dentro del flujo de bits 138 pueden ser de tipo diferente, y el descomponedor 136 puede configurarse para realizar la descomposición individual dependiendo del tipo del elemento de sintaxis. El descomponedor 136 realiza preferentemente la descomposición individual de los elementos de sintaxis de modo que existe un mapeado inverso único predeterminado usado posteriormente en el lado de la decodificación, desde el número entero de símbolos fuente al elemento de sintaxis respectivo, común para todos los elementos de sintaxis.

Por ejemplo, el descomponedor 136 puede configurarse para descomponer elementos de sintaxis z en el flujo de bits 138 que puede analizarse, en dos símbolos fuente x e y de modo que $z = x + y$, $z = x - y$, $z = x \cdot y$ o $z = x : y$. Con esta medida, el subdivisor 100 puede descomponer los elementos de sintaxis en dos componentes, concretamente símbolos fuente del flujo de símbolos fuente 106, uno de los cuales es adecuado para ser codificado por VLC en términos de eficiencia de compresión, tal como x , y el otro de los cuales no es adecuado para la codificación por VLC y se pasa, por lo tanto, al segundo subflujo 110 en lugar de al primer subflujo 108, tal como y . La descomposición usada por el descomponedor 136 no necesita ser biyectiva. Sin embargo, como se ha mencionado anteriormente existiría un mapeado inverso que permita una recuperación única de los elementos de sintaxis de las posibles descomposiciones entre las que el descomponedor 136 puede elegir si la descomposición no es biyectiva.

Hasta ahora, se han descrito diferentes posibilidades para el manejo de diferentes elementos de sintaxis. Si dichos elementos de sintaxis o casos existen, es opcional. La descripción adicional, sin embargo, se concentra en elementos de sintaxis que se descomponen por el descomponedor 136 de acuerdo con el siguiente principio.

Como se muestra en la Fig. 1b, el descomponedor 136 se configura para descomponer ciertos elementos de sintaxis z en los flujos de bits 138 que puede analizarse en etapas. Pueden existir dos o más etapas. Las etapas son para la división del intervalo de valores de los elementos de sintaxis z en dos o más subintervalos adyacentes o subrangos tal como se muestra en la Fig. 1c. El rango de valores del elemento de sintaxis puede tener dos puntos extremos infinitos, meramente uno o puede tener puntos extremos definidos. En la Fig. 1c, el rango de valores del elemento de sintaxis se subdivide ejemplarmente en tres particiones 140_{1-3} . Tal como se muestra en la Fig. 1b, si el elemento de sintaxis es mayor que o igual que la delimitación 142 de la primera partición 140_1 , es decir el límite superior de

separación de particiones 140_1 y 140_2 , entonces el elemento de sintaxis se resta de la delimitación límite1 de la primera partición 140_1 y z se comprueba de nuevo para comprobar si el mismo es incluso mayor o igual que la delimitación 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 que o igual que la delimitación 144, entonces z' se resta la delimitación límite2 de la segunda partición 140_2 dando 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 tal cual, en caso de que z esté entre límite1 y límite2, el elemento de sintaxis z se envía al subdivisor 100 en una dupla (límite1, z') siendo $z = \text{límite1} + z'$, y en caso de que z esté por encima de límite2, el elemento de sintaxis z se envía al subdivisor 100 como un triplete (límite1, límite2-límite1, z') siendo $z = \text{límite1} + \text{límite2} + z'$. El primer (o único) componente, es decir z o límite1, forma un primer símbolo fuente a ser codificado por el subdivisor 100, el segundo componente, es decir z' o límite2 - límite1, forman un segundo símbolo fuente a ser codificado por el subdivisor 100, si está presente, y el tercer componente, es decir z'' , forma un tercer símbolo fuente a ser codificado por el subdivisor 100, si está presente. Por ello, de acuerdo con la Fig. 1b y 1c, el elemento de sintaxis se mapea a cualquiera de 1 a 3 símbolos fuente, pero las generalizaciones sobre un número máximo menor o mayor de símbolos fuente es fácilmente deducible de la descripción anterior, y dichas alternativas también se describirán en lo que sigue.

En cualquier caso, todos estos diferentes componentes o símbolos fuente resultantes están de acuerdo con las realizaciones a continuación, codificadas con codificación entre las alternativas. Al menos una de ellas se envía por el subdivisor al codificador PIPE 104, y al menos otra de las mismas se envía al codificador VLC 102.

Se describen con más detalle a continuación realizaciones ventajosas particulares.

Después de haber descrito anteriormente un aparato de codificación entrópica, se describe con respecto a la Fig. 2a un aparato de decodificación entrópica. El aparato de decodificación entrópica de la Fig. 2a comprende un decodificador VLC 200 y un decodificador PIPE 202. El decodificador VLC 200 se configura para reconstruir acertadamente con relación a código los símbolos fuente de una primera subsecuencia 204 de palabras de código de un primer flujo de bits 206. El primer flujo de bits 206 es igual al flujo de bits 112 de la Fig. 1, y lo mismo se aplica a la subsecuencia 204 en lo que se refiere a la subsecuencia 108 de la Fig. 1a. El decodificador PIPE 202 se configura para reconstruir una segunda subsecuencia 208 de símbolos fuente, representados en la forma de una secuencia de símbolos de alfabeto, que comprende una pluralidad de decodificadores entrópicos 210, un asignador 212 y un selector 214. La pluralidad de decodificadores entrópicos 210 se configura para convertir uno respectivo de los segundos flujos de bits 216 en símbolos de alfabeto de la secuencia de símbolos de alfabeto. El asignador 212 se configura para asignar una medida de una estimación de una distribución de probabilidad entre valores posibles que pueden asumir los símbolos de alfabeto respectivos, a cada símbolo de alfabeto de la secuencia de símbolos de alfabeto que representan la segunda subsecuencia 208 de símbolos fuente a ser reconstruidos, basándose en la información contenida dentro de los símbolos de alfabeto previamente reconstruidos de la secuencia de símbolos de alfabeto. Con este fin, el asignador 212 puede conectarse en serie entre una salida del selector 214 y una entrada del mismo, en tanto que entradas adicionales del selector 214 tienen salidas de los decodificadores entrópicos 210 respectivamente conectadas al mismo. El selector 214 se configura para recuperar cada símbolo de alfabeto de la secuencia de símbolos de alfabeto desde uno seleccionado de la pluralidad de decodificadores entrópicos 210, dependiendo la selección de la medida asignada al símbolo de alfabeto respectivo. En otras palabras, el selector 214 junto con el asignador 212 es operativo para recuperar los símbolos de alfabeto obtenidos por los decodificadores entrópicos 210 en un orden entre los decodificadores entrópicos 210 obtenidos mediante la revisión de la información contenida dentro de símbolos de alfabeto previos de la secuencia de símbolos de alfabeto. Incluso en otras palabras, el asignador 212 y el selector 214 son capaces de reconstruir el orden original de los símbolos de alfabeto de símbolo de alfabeto a símbolo de alfabeto. Junto con la predicción del siguiente símbolo de alfabeto, el asignador 212 es capaz de determinar la medida anteriormente mencionada de la estimación de la distribución de probabilidad para el símbolo de alfabeto respectivo mediante el uso del cual el selector 214 selecciona entre los decodificadores entrópicos 210 a recuperar el valor actual de su símbolo de alfabeto. Para ser incluso más preciso, y tal como se describirá con más detalle a continuación, el decodificador PIPE 202 puede configurarse para reconstruir la secuencia 208 de símbolos fuente, representada en la forma de la secuencia de símbolos de alfabeto, reaccionando a las solicitudes de símbolos de alfabeto que solicitan secuencialmente los símbolos de alfabeto, y el asignador 212 puede configurarse para asignar cada solicitud de un símbolo de alfabeto de la secuencia de símbolos de alfabeto que representan la segunda subsecuencia (208) de símbolos fuente a ser reconstruidos, la medida anteriormente mencionada de una estimación de una distribución de probabilidad entre los valores posibles que pueden asumir los símbolos de alfabeto respectivos. En consecuencia, el selector 214 puede configurarse para recuperar, para cada solicitud de un símbolo de alfabeto de la secuencia de símbolos de alfabeto que representa la segunda subsecuencia (208) de símbolos fuente a ser reconstruida, el símbolo de alfabeto respectivo de la secuencia de símbolos de alfabeto de uno seleccionada de la pluralidad de decodificadores entrópicos 210, dependiendo la selección de la medida asignada a la solicitud respectiva del símbolo de alfabeto respectivo. La concordancia entre solicitudes en el lado de la decodificación por un lado, y el flujo de datos o codificación en el lado de la codificación por otro lado, se describirá con más detalle con respecto a la Fig. 4.

Como la primera subsecuencia 214 de símbolos fuente y la segunda subsecuencia 208 de símbolos fuente forman

comúnmente una secuencia común 210 de símbolos fuente, el aparato de decodificación entrópica de la Fig. 2a puede, opcionalmente, comprender un recombinaor 220 configurado para recombinar la primera subsecuencia 204 y la segunda subsecuencia 208 para obtener la secuencia común 218 de símbolos fuente. Esta secuencia común 208 de símbolos fuente conduce a una reconstrucción de la secuencia 106 de la Fig. 1a.

5 De acuerdo con la descripción presentada anteriormente con respecto a la Fig. 1, los símbolos fuente de la primera y segunda subsecuencias 204 y 208 pueden ser elementos de sintaxis de un flujo de bits que puede analizarse. En este caso, el recombinaor 220 podría configurarse para reconstruir este flujo de bits que puede analizarse de la secuencia 218 de los elementos de sintaxis mediante el intercalado de los símbolos fuente que llegan a través de la primera y segunda subsecuencias 204 y 208 en un orden preestablecido mediante alguna regla de análisis que define un orden entre los elementos de sintaxis. En particular, los elementos de sintaxis pueden ser, como se ha descrito anteriormente, de tipo diferente y el recombinaor 220 puede configurarse para recuperar o solicitar elementos de sintaxis de un primer grupo de tipos desde el decodificador VLC 200 a través del subflujo 204, y elementos de sintaxis de un segundo tipo desde el decodificador PIPE 202 a través del subflujo 208. 10 15

consecuencia, siempre que la regla de análisis recién mencionada indique que un elemento de sintaxis de un tipo dentro del primer grupo es el siguiente en línea, el recombinaor 202 inserta un símbolo fuente actual de la subsecuencia 204 en la secuencia común 218 y en caso contrario de la subsecuencia 208.

20 De la misma manera, el decodificador PIPE 202 podría incluir un desimbolizador 222 conectado entre la salida del selector 214 y una entrada del recombinaor 220. De modo similar a la descripción anterior con respecto a la Fig. 1, el desimbolizador 222 podría considerarse como externo al decodificador PIPE 202 y podría disponerse incluso detrás del recombinaor 202, es decir en el lado de salida del recombinaor 220, alternativamente. El desimbolizador 222 podría configurarse para volver a mapear, en unidades de secuencias parciales de símbolos de alfabeto, la secuencia de símbolos de alfabeto 224 producida por el selector 214 en la forma de símbolos fuente, es decir elementos de sintaxis de la subsecuencia 208. De modo similar al recombinaor 220, el desimbolizador 222 sabe acerca de la construcción de posibles secuencias parciales de símbolos de alfabeto. En particular, el desimbolizador 222 puede analizar símbolos de alfabeto recientemente recibidos desde el selector 214 para asegurarse de si estos símbolos de alfabeto recientemente recibidos conducen a una secuencia parcial válida de símbolos de alfabeto asociada con un valor respectivo del elemento de sintaxis respectivo, o si este no es el caso, y 25 30 qué símbolo de alfabeto falta a continuación. En aún otras palabras, el simbolizador 222 sabe, en cualquier momento, si han de recibirse símbolos de alfabeto adicionales desde el selector 214 para finalizar la recepción de un elemento de sintaxis respectivo o no, y en consecuencia, a qué elemento de sintaxis pertenece uno respectivo de los símbolos de alfabeto producidos por el selector 214. Con este fin, el desimbolizador 222 puede usar un esquema de (des)mapeado del simbolizado que diferencie los elementos de sintaxis de tipo diferente. De modo similar, el asignador 212 sabe acerca de la asociación de un símbolo de alfabeto actual a ser recuperado desde cualquiera de los decodificadores entrópicos 210 por el selector 214, a uno respectivo de los elementos de sintaxis y puede fijar la medida anteriormente mencionada de estimación de una distribución de probabilidad de este símbolo de alfabeto en consecuencia, es decir depende del tipo de elemento de sintaxis asociado. Aún más, el asignador 212 puede diferenciar entre diferentes símbolos de alfabeto que pertenezcan a la misma secuencia parcial de un símbolo de alfabeto actual y puede fijar la medida de la estimación de distribución de probabilidad de modo diferente para estos símbolos de alfabeto. Se describen con más detalle a continuación detalles en este sentido. Como se describe en la presente memoria, el asignador 212 puede configurarse para asignar contextos a los símbolos de alfabeto. La asignación puede depender del tipo de elemento de sintaxis y/o de la posición dentro de la secuencia parcial de los símbolos de alfabeto del elemento de sintaxis actual. Tan pronto como el asignador 212 haya asignado un contexto al símbolo de alfabeto actual a ser recuperado desde cualquiera de los decodificadores entrópicos 210 por el selector 214, el símbolo de alfabeto puede tener inherentemente la media de la estimación de la distribución de probabilidad asociada con el mismo dado que cada contexto tiene su medida de estimación asociada con él. Adicionalmente, el contexto —y su medida asociada de estimación de la distribución de probabilidad— pueden adaptarse de acuerdo con las estadísticas actuales de los símbolos de alfabeto del contexto respectivo que se han recuperado desde los decodificadores entrópicos 210 hasta el momento. Se presentan con más detalle a continuación detalles en este sentido. 35 40 45 50

De modo similar a la explicación anterior de la Fig. 1, puede ser posible que la correspondencia entre los símbolos fuente anteriormente mencionados de las subsecuencias 204 y 208 en elementos de sintaxis no sea una correspondencia uno a uno. Por el contrario, los elementos de sintaxis pueden haberse descompuesto en un número entero de símbolos fuente variando el número, eventualmente, entre los elementos de sintaxis pero siendo, en cualquier caso, mayor que uno al menos para un elemento de sintaxis. Como se ha hecho notar anteriormente, la descripción a continuación se enfoca sobre el manejo de esta clase de elementos de sintaxis, y elementos de sintaxis de otras clases pueden incluso no estar presentes. 55 60

Para el manejo de los elementos de sintaxis recién mencionados, el aparato de decodificación entrópica de la Fig. 2a puede comprender un componedor 224 configurado para rehacer la descomposición realizada por el descomponedor 136 de la Fig. 1a. En particular, el componedor 224 puede configurarse para componer la secuencia 226 de elementos de sintaxis desde los símbolos fuente de la secuencia 218 o, si el recombinaor 220 falta,

subsecuencias 204 y 208, mediante la composición individualmente de cada elemento de sintaxis a partir de un número entero respectivo de símbolos fuente perteneciendo uno de los símbolos fuente del número entero de símbolos fuente a la primera subsecuencia 204 y perteneciendo otro de los símbolos fuente del número entero de símbolos fuente del mismo elemento de sintaxis a la segunda subsecuencia 208. Con esta medida, ciertos elementos de sintaxis pueden haberse descompuesto en el lado del codificador de modo que separen componentes adecuados para la decodificación VLC de un componente restante que haya de ser pasado a través de una trayectoria de decodificación PIPE. De modo similar a la explicación anterior, el elemento de sintaxis puede ser un tipo diferente y el componedor 224 puede configurarse para realizar la composición individual dependiendo del tipo de elementos de sintaxis. En particular, el componedor 224 puede configurarse para obtener los elementos de sintaxis respectivos mediante la combinación lógica o matemáticamente del número entero de símbolos fuente del elemento de sintaxis respectivo. Por ejemplo, el componedor 224 puede configurarse, para aplicar, a cada elemento de sintaxis, +, -, : o · al primer y segundo símbolos fuente de un elemento de sintaxis.

Como se ha descrito anteriormente, las realizaciones descritas en el presente documento a continuación, sin embargo, se concentran sobre elementos de sintaxis que se descomponen por el descomponedor 136 de acuerdo con las Figs. 1b y 1c y las alternativas descritas en relación a las mismas. La Fig. 2a muestra cómo puede funcionar el componedor 224 para reconstruir estos elementos de sintaxis a partir de sus símbolos fuente 218.

Como se muestra en la Fig. 2b, el componedor 224 se configura para componer dichos elementos de sintaxis z en etapas a partir de los símbolos fuente incidentes s_1 a s_x , siendo x cualquiera de 1 a 3 en el ejemplo presente. Pueden existir dos o más etapas. Como se muestra en la Fig. 2b, el componedor 224 fija preliminarmente z como el primer símbolo s_1 y comprueba si z es igual al primer límite límite1. Si no es este el caso, se ha encontrado z . En caso contrario, el componedor 224 añade el siguiente símbolo fuente s_2 del flujo de símbolos fuente 218 a z y comprueba de nuevo si esta z es igual a límite2. Si no, se ha encontrado z . Si no, el componedor 224 añade el siguiente símbolo fuente s_3 del flujo de símbolos fuente 218 a z , para obtener z en su forma final. Generalizaciones sobre un número máximo mayor o menor de símbolos fuente es fácilmente deducible de la descripción anterior, y dichas alternativas también se describirán en lo que sigue.

En cualquier caso, todos estos diferentes componentes o símbolos fuente o resultantes están de acuerdo con la descripción a continuación, codificados con entre las alternativas de codificación. Al menos uno de ellos se envía por el subdivisor al codificador PIPE 104, y al menos otro de los mismos se envía al codificador VLC 102.

Detalles ventajosos particulares se describen con más detalle a continuación. Estos detalles se concentran sobre las posibilidades favorables de dividir el intervalo de valores de los elementos de sintaxis y los esquemas de codificación VLC y PIPE entrópicos que pueden usarse para codificar los símbolos fuente.

Adicionalmente, como se ha descrito también anteriormente con respecto a la Fig. 1, el aparato decodificador entrópico de la Fig. 2a puede configurarse para recibir el primer flujo de bits 206 así como la pluralidad de segundos flujos de bits 216 por separado o en una forma intercalada por medio de un flujo de bits 228 intercalado. En este último caso, el aparato de decodificación entrópica de la Fig. 2a puede comprender un desintercalador 230 configurado para desintercalar el flujo de bits 228 intercalado para obtener el primer flujo de bits 206 por un lado y la pluralidad de segundos flujos de bits 216 por otro lado. De modo similar a la explicación anterior de la Fig. 1, el desintercalador 230 puede subdividirse en dos etapas, concretamente un desintercalador 232 para el desintercalado del flujo de bits 228 intercalado en dos partes, concretamente el flujo de bits 206 por un lado y una forma intercalada 234 del segundo flujos de bits 216 por otro lado, y un desintercalador 236 para el desintercalado de este último flujo de bits 234 para obtener los flujos de bits 216 individuales.

Así, la Fig. 1a y la Fig. 2a mostraron un aparato de codificación entrópica por un lado y un aparato de decodificación entrópica adecuado para la decodificación del resultado de codificación obtenido por el aparato de codificación entrópica de la Fig. 1, por otro lado. Detalles con relación a muchos de los elementos mostrados en la Fig. 1a y la Fig. 2 se describen con más detalle con relación a las figuras adicionales. En consecuencia, se hace referencia a estos detalles en la descripción que sigue y estos detalles deberán considerarse aplicables también a la Fig. 1a y la Fig. 2 individualmente, siempre que estos detalles puedan implementarse por separado en los codificadores y decodificadores anteriormente descritos. Meramente con respecto a los intercaladores y desintercaladores 132 y 234, se realiza aquí una indicación adicional. En particular, el intercalado de los flujos de bits 112 y 118 puede ser favorable en caso de que los flujos de bits hayan de ser multiplexados en un canal para ser transmitidos. En este caso, puede ser favorable intercalar el flujo de bits VLC 112 por un lado y los flujos de bits de codificación PIPE 118 por otro lado de modo que obedezcan a ciertas condiciones a ser satisfechas tal como obedecer a algún retardo máximo de decodificación. Dicho en otras palabras, puede ser necesario que el desplazamiento de tiempo relativo entre los tiempos de los elementos de sintaxis y símbolos fuente, respectivamente, sean recuperables en el lado de decodificación por un lado y el desplazamiento relativo en el tiempo de acuerdo con su posición en el flujo de bits que puede realizarse por otro lado, no supere un cierto retardo máximo. Se describen a continuación muchas alternativas para resolver este problema. Una de estas posibilidades implica que los codificadores entrópicos 116 sean un tipo de codificador de longitud variable configurado para mapear secuencias de símbolos de alfabeto a palabras de código, y los decodificadores entrópicos 210 hacer el mapeado inverso. Las palabras de código del flujo

de bits VLC 112 y los flujos de bits PIPE 118 pueden ser, pero no han de ser, seleccionadas de modo que ninguna palabra de código de ninguno de estos flujos de bits sea el prefijo de ninguna palabra de código de ninguno de los otros flujos de bits, de modo que los bordes de las palabras de código permanecen únicamente determinables en el lado del decodificador. En cualquier caso, el intercalador 128 puede configurarse para reservar y almacenar en memoria intermedia una secuencia de entradas de palabras de código para la palabra de código dentro del primer flujo de bits 112 y del segundo flujo de bits 118 en un orden secuencial dependiendo de un orden en el que los símbolos de alfabeto de la secuencia 124 de símbolos de alfabeto enviados por el selector 120 a la pluralidad de codificadores entrópicos 116 dan como resultado un inicio de una nueva secuencia de símbolos de alfabeto a ser mapeada a una palabra de código respectiva en el codificador entrópico 116 respectivo y se mapee un nuevo símbolo fuente del segundo subflujo 108 por el codificador VLC 102, respectivamente. En otras palabras, el intercalador 128 inserta las palabras de código del flujo de bits 112 dentro del flujo de bits 126 común en el orden de los símbolos fuente a partir de los que se han obtenido mediante la codificación VLC, en su orden dentro del subflujo 108 y la corriente de símbolos fuente 106, respectivamente. Las palabras de código producidas por los codificadores entrópicos 116 se insertan dentro del flujo de bits 126 común entre las palabras de código consecutivas del flujo de bits VLC 112. Debido a la categorización de la codificación PIPE de los símbolos de alfabeto por el asignador 114 y selector 120, respectivamente, cada una de las palabras de código de los codificadores entrópicos 116 tiene símbolos de alfabeto de diferentes símbolos fuente del subflujo 110 codificado en ella. La posición de las palabras de código de los flujos de bits 118 codificados PIPE dentro del flujo de bits 126 común entre ellos y con relación a la palabra de código VLC del flujo de bits 112 se determina por el primer símbolo de alfabeto codificado en cada palabra de código, respectivamente, es decir el más antiguo en el tiempo. El orden de estos símbolos de alfabeto primarios codificados dentro de las palabras de código de los flujos de bits 118 en el flujo de los símbolos de alfabeto 124 determina al orden de las palabras de código de los flujos de bits 118 dentro del flujo de bits 126 común entre ellos, con relación a las palabras de código VLC del flujo de bits 112 el símbolo fuente al cual pertenece este símbolo de alfabeto primario codificado dentro de las palabras de código del flujo de bits 118, determinan entre qué palabras de código consecutivas del flujo de bits 112 se posiciona la palabra de código respectiva de cualquiera de los flujos de bits 118. En particular, las palabras de código VLC consecutivas entre las que se han de posicionar las palabras de código respectivas de cualquiera de los flujos de bits 118, son aquellas entre las que el símbolo fuente del subflujo 110 se posiciona de acuerdo con el orden original de un flujo de símbolos fuente no subdividido 106, al que pertenece el símbolo de alfabeto primero respectivo codificado en las palabras de código respectiva del flujo de bits 118. El intercalador 128 puede configurarse para eliminar las palabras de código introducidas dentro de las entradas de palabras de código anteriormente mencionadas en el orden secuencial para obtener el flujo de bits 126 común de palabras de código intercaladas. Como ya se ha descrito anteriormente, los codificadores entrópicos 116 pueden configurarse para introducir secuencialmente sus palabras de código en las entradas de palabras de código reservadas para el codificador entrópico 116 respectivo y el selector 120 puede configurarse para enviar los símbolos de alfabeto que representan los símbolos fuente del segundo subflujo 110 en un orden que mantiene un orden en el que los símbolos fuente del primer subflujo 108 y el segundo subflujo 110 se intercalaron dentro de la secuencia 106 de símbolos fuente.

Pueden proporcionarse medidas adicionales para hacer frente a situaciones en las que ciertos de los codificadores entrópicos 116 se seleccionan tan aleatoriamente que lleva mucho tiempo obtener una palabra de código válida dentro de ese codificador entrópico 116 raramente usado. Ejemplos de dichas medidas se describen con más detalle a continuación. En particular, el intercalador 128 junto con el codificador entrópico 116 pueden, en este caso, configurarse para purgar sus símbolos de alfabeto recogidos hasta el momento y las palabras de código que se han introducido dentro de las entradas de palabras de código anteriormente mencionadas, respectivamente, de tal manera que el tiempo del procedimiento de purga puede predecirse o emularse en el lado de decodificación.

En el lado de decodificación, el desintercalador 230 pueda actuar en el sentido inverso: siempre que, de acuerdo con el esquema de análisis anteriormente mencionado, el siguiente símbolo fuente a ser decodificado, sea un símbolo codificado en VLC, una palabra de código actual dentro del flujo de bits 228 común se considera como una palabra de código VLC y se envía dentro del flujo de bits 206 al decodificador VLC 200. Por otro lado, siempre que cualquiera de los símbolos de alfabeto que pertenecen a cualquiera de los símbolos codificados PIPE del subflujo 208 sea un símbolo de alfabeto primario, es decir necesite un nuevo mapeado de una palabra de código de uno respectivo de los flujos de bits 216 a una secuencia de símbolos de alfabeto respectiva por el decodificador entrópico 210 respectivo, la palabra de código actual del flujo de bits 228 común se considera como una palabra de código codificada PIPE y se envía al decodificador entrópico 210 respectivo. La detección del siguiente borde de palabra de código, es decir la detección de la extensión de la siguiente palabra de código desde el extremo de la palabra de código que se acaba de enviar a cualquiera de los codificadores 200 y 202, respectivamente, hasta su fin dentro del flujo de bits 228 intercalado entrante puede aplazarse, y realizarse bajo el conocimiento de, el decodificador 200 y 202 siendo el receptor dedicado de esta siguiente palabra de código de acuerdo con la regla anteriormente descrita: basándose en este conocimiento, el libro de códigos usado por el decodificador receptor es conocido y la palabra de código respectiva detectable. Si, por otro lado, los libros de códigos se diseñaran de modo que los bordes de la palabra de código fueran detectables sin un conocimiento a priori acerca del decodificador receptor entre 200 y 202, entonces la separación de palabras de código serie se realizaría en paralelo. En cualquier caso, debido al intercalado, los símbolos fuente están disponibles en el decodificador en una forma entrópica

decodificada, es decir como símbolos fuente, en su orden correcto con un retardo razonable.

Después de haber descrito las realizaciones anteriores para un aparato de codificación entrópica y el aparato de decodificación entrópica respectivo, a continuación se describen más detalles de los codificadores PIPE y decodificadores PIPE anteriormente mencionados.

Un codificador PIPE se ilustra en la Fig. 3. El mismo puede usarse como codificador PIPE en la Fig. 1a. El codificador PIPE convierte sin pérdidas un flujo de símbolos fuente 1 en un conjunto de dos o más flujos de bits 12 parciales. Cada símbolo fuente 1 puede asociarse con una categoría o tipo entre un conjunto de una o más categorías o tipos. Como un ejemplo, las categorías pueden especificar el tipo de símbolo fuente. En el contexto de la codificación de video híbrida, una categoría separada puede asociarse con modos de codificación de macrobloque, modos de codificación de bloque, índices de imagen de referencia, diferencias del vector de movimiento, indicadores de subdivisión, indicadores de bloque codificado, parámetros de cuantificación, niveles de coeficiente de transformación, etc. En otras áreas de aplicación tales como audio, voz, texto, documentos, o codificación de datos en general, son posibles diferentes categorizaciones de símbolos fuente.

En general, cada símbolo fuente puede tomar un valor de un conjunto finito o contable infinito de valores, en donde el conjunto de posibles valores del símbolo fuente puede diferir para diferentes categorías de símbolos fuente. Para reducir la complejidad del algoritmo de codificación y decodificación y para permitir un diseño de codificación y decodificación general para diferentes símbolos fuente y categorías de símbolos fuente, los símbolos fuente 1 se convierten en conjuntos ordenados de decisiones binarias y estas decisiones binarias se procesan a continuación mediante algoritmos de codificación binaria simples. Por lo tanto, el binarizador 2 mapea de modo biyectivo el valor de cada símbolo fuente 1 en una secuencia (o cadena) de bins 3. La secuencia de bins 3 representa un conjunto de decisiones binarias ordenadas. Cada bin 3 o decisión binaria puede tomar 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 fuente. El esquema de binarización para una categoría de símbolos fuente particular puede depender del conjunto de posibles valores del símbolo fuente y/u otras propiedades de los símbolos fuente para la categoría particular. La Tabla 1 ilustra tres ejemplos de esquemas de binarización para conjuntos infinitos contables. Los esquemas de binarización para conjuntos infinitos contables pueden aplicarse también para conjuntos finitos de valores de símbolos. En particular para conjuntos finitos grandes de valores de símbolos, la ineficiencia (resultante de secuencias de bins no usadas), puede ser despreciable, pero la universalidad de dichos esquemas de binarización proporciona una ventaja en términos de complejidad y requisitos de memoria. Para conjuntos finitos pequeños de valores de símbolos, es frecuentemente preferible (en términos de eficiencia de codificación) adaptar el esquema de binarización al número de posibles valores de símbolos. La Tabla 2 ilustra tres esquemas de binarización de ejemplo para conjuntos finitos de 8 valores. Los esquemas de binarización para conjuntos finitos pueden deducirse de los esquemas de binarización universales para conjuntos infinitos contables mediante la modificación de algunas secuencias de bins en una forma en que los conjuntos finitos de secuencias de bins representan un código libre de redundancias (y potencialmente reordenación de las secuencias de bins). Como un ejemplo, el esquema de binarización unario truncado en la Tabla 2 se creó mediante la modificación de la secuencia de bins para el símbolo fuente 7 de la binarización unaria universal (véase la Tabla 1). La binarización Exp-Golomb truncada y reordenada de orden 0 en la Tabla 2 se creó mediante la modificación de la secuencia de bins para el símbolo fuente 7 de la binarización unaria universal Exp-Golomb de orden 0 (véase Tabla 1) y mediante la reordenación de las secuencias de bins (la secuencia de bins truncada para el símbolo 7 se asignó al símbolo 1). Para conjuntos finitos de símbolos, es posible también usar esquemas de binarización no sistemáticos / no universales, 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 del símbolo	binarización unaria	binarización Exp-Golomb de orden 0	binarización Exp-Golomb de orden 1
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 del símbolo	binarización unaria truncada	binarización Exp-Golomb de orden 0 truncada y reordenada	binarización no sistemática
0	1	1	000

Valor del símbolo	binarización unaria truncada	binarización Exp-Golomb de orden 0 truncada y reordenada	binarización no sistemática
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 bin 3 de la secuencia de bins creada por el binarizador 2 se suministra al interior de un asignador de parámetros 4 en orden secuencial. El asignador de parámetros asigna un conjunto de uno o más parámetros a cada bin 3 y produce la salida del bin con el conjunto de parámetros 5 asociado. El conjunto de parámetros se determina en exactamente la misma forma en el codificador y decodificador. El conjunto de parámetros puede consistir en uno o más de los siguientes parámetros:

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

El asignador de parámetros 4 puede asociar cada bin 3, 5 con una medida de una estimación de la probabilidad para uno de los dos posibles valores de bin para el bin actual. El asignador de parámetros 4 asocia cada bin 3, 5 con una medida de una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin actual y un identificador que especifica la estimación de cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin actual. Debería tomarse nota de que la probabilidad para el valor de bin menos probable o más probable y el indicador que especifica cuál de los dos valores de bin posibles representa el valor de bin menos probable o más probable son medidas equivalentes de la probabilidad de uno de los dos valores de bin posibles.

El asignador de parámetros 4 puede asociar cada bin 3, 5 con una medida de una estimación de la probabilidad para uno de los dos posibles valores de bin para el bin actual y uno o más parámetros adicionales (que pueden ser uno o más de los parámetros listados anteriormente). Adicionalmente, el asignador de parámetros 4 puede asociar cada bin 3, 5 con una medida de la estimación de la probabilidad del valor de bin menos probable o más probable para el bin actual, un identificador que especifica la estimación para cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin actual, y uno o más parámetros adicionales (que pueden ser uno o más de los parámetros listados anteriormente).

El asignador de parámetros 4 puede determinar una o más de las medidas de probabilidad mencionadas anteriormente (medida de una estimación de la probabilidad de uno o más valores de bin posibles para el bin actual, medida de la estimación de la probabilidad del valor de bin menos probable o más probable para el bin actual, identificador que especifica la estimación de cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin actual) basándose en 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 corresponden al conjunto de datos (tales como bloques o grupos de muestras) de localizaciones espaciales y/o temporales (en relación al conjunto de datos asociado con el símbolo fuente actual), o uno o más símbolos ya codificados de diferentes categorías de símbolos que corresponden a conjuntos de datos de la misma y/o contiguas localizaciones espaciales y/o temporales (en relación al conjunto de datos asociado con el símbolo fuente actual).

Cada bin con un conjunto asociado de parámetros 5 que sale del asignador de parámetros 4 se alimenta al interior de un selector 6 de memoria intermedia de bin. El selector 6 de memoria intermedia de bin modifica potencialmente el valor del bin de entrada 5 basándose en el valor del bin de entrada y los parámetros asociados 5 y suministra el bin 7 de salida —con un valor potencialmente modificado— a una de dos o más memorias intermedias de bin 8. La

memoria intermedia de bin 8 a la que se envía el bin de salida 7 se determina basándose en el valor del bin de entrada 5 y/o el valor de los parámetros asociados 5.

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

10 El selector 6 de memoria intermedia de bin puede determinar el valor del bin de salida 7 basándose en el valor de bin de entrada 5 y la medida asociada para una estimación de la probabilidad para uno de los dos valores de bin posibles para el bin actual. El valor del bin de salida 7 puede fijarse igual al valor del bin de entrada 5 si la medida de la probabilidad para uno de los dos valores posibles de bin para el bin actual es menor que (o menor que o igual a) un umbral particular; si la medida de la probabilidad para uno de los dos posibles valores de bin para el bin actual es mayor que o igual a (o mayor que) un umbral particular, el valor del bin de salida 7 se modifica (es decir, se fija opuesto al valor del bin de entrada). El valor del bin de salida 7 puede fijarse igual al valor del bin de entrada 5 si la medida de la probabilidad para uno de los dos valores de bin posible para el bin actual es mayor que (o mayor que o igual a) un umbral particular; si la medida de la probabilidad de uno de los dos valores de bin posibles para el bin actual es menor que o igual a (o menor que) un umbral particular, el valor del bin de salida 7 se modifica (es decir, se fija opuesto al valor del bin de entrada). El valor del umbral puede corresponder a un valor de 0,5 para la probabilidad estimada de ambos posibles valores de bin.

20 El selector 6 de memoria intermedia de bin puede determinar el valor del bin de salida 7 basándose en el valor del bin de entrada 5 y el identificador asociado que especifica la estimación para cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin actual. El valor de bin de salida 7 puede fijarse igual al valor de bin de entrada 5 si el identificador especifica que el primero de los dos posibles valores de bin representa el valor de bin menos probable (o más probable) para el bin actual, y el valor de bin de salida 7 se modifica (es decir se fija en oposición al valor del bin de entrada) si el identificador especifica que el segundo de los dos posibles valores de bin representa el valor de bin menos probable (o más probable) para el bin actual.

30 El selector 6 de la memoria intermedia de bin puede determinar la memoria intermedia de bin 8 a la que se envía el bin de salida 7 basándose en la medida asociada para una estimación de la probabilidad para uno de los dos posibles valores de bin para el bin actual. El conjunto de valores posibles para la medida de una estimación de la probabilidad para uno de los dos posibles valores de bin puede ser finita y el selector 6 de memoria intermedia de bin contener una tabla que asocia exactamente una memoria intermedia de bin 8 con cada valor posible para la estimación de la probabilidad de uno de los dos posibles valores de bin, en el que pueden asociarse diferentes valores para la medida de la estimación de la probabilidad de uno de los dos posibles valores de bin con la misma memoria intermedia de bin 8. Adicionalmente, el intervalo de posibles valores para la medición de una estimación de la probabilidad para uno de los dos posibles valores de bin puede particionarse en un número de intervalos, el selector 6 de memoria intermedia de bin determina el índice del intervalo para la medición actual para una estimación de la probabilidad para uno de los dos posibles valores de bin, y el selector 6 de la memoria intermedia de bin contiene una tabla que asocia exactamente una memoria intermedia de bin 8 con cada valor posible para el índice del intervalo, en el que pueden asociarse diferentes valores para el índice de intervalo con la misma memoria intermedia de bin 8. Los bins de entrada 5 con medidas opuestas para una estimación de la probabilidad para uno de los dos posibles valores de bin (medición opuesta a aquella que representa las estimaciones de probabilidad P y 1 - P), pueden suministrarse a la misma memoria intermedia de bin 8. Adicionalmente, la asociación de la medida para la estimación de la probabilidad para uno de los dos posibles valores de bin para el bin actual con una memoria intermedia de bin particular se adapta a lo largo del tiempo, por ejemplo para asegurar que los flujos de bits parciales creados tienen similares tasas de bits.

50 El selector 6 de la memoria intermedia de bin puede determinar la memoria intermedia de bin 8 a la que se envía el bin de salida 7 basándose en la medida asociada para una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin actual. El conjunto de valores posibles para la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable de ser finita y el selector 6 de memoria intermedia de bin contener una tabla que asocia exactamente una memoria intermedia de bin 8 con cada valor posible de la estimación de la probabilidad para el valor de bin menos probable o más probable, en el que valores diferentes para la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable puede asociarse con la misma memoria intermedia de bin 8. Adicionalmente, el rango de posibles valores para la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable puede particionarse en un número de intervalos, el selector 6 de memoria intermedia de bin determina el índice del intervalo para la medición actual de una estimación de la probabilidad para el valor de bin menos probable o más probable, y el selector 6 de la memoria intermedia de bin contiene una tabla que asocia exactamente una memoria intermedia de bin 8 con cada posible valor para el índice del intervalo, en donde valores diferentes para el índice del intervalo pueden asociarse con la misma memoria intermedia de bin 8. La asociación de la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin actual con una memoria intermedia de bin particular puede adaptarse a lo largo del tiempo, por ejemplo para asegurar que los flujos de bits parciales creados tienen similares tasas de bits.

Cada una de las dos o más memorias intermedias de bin 8 se conecta con exactamente un codificador de bin 10 y cada codificador de bin se conecta solamente con una memoria intermedia de bin 8. Cada codificador de bin 10 lee bins desde la memoria intermedia de bin 8 asociada y convierte una secuencia de bins 9 en una palabra de código 11, que representa una secuencia de bits. La memoria intermedia de bin 8 representa memorias intermedias primero en entrar primero en salir; los bins que se suministran más tarde (en el orden secuencial) al interior de la memoria intermedia de bin 8 no se codifican antes de los bin que se suministran antes (en orden secuencial) dentro de la memoria intermedia de bin. Las palabras de código 11 que salen de un codificador de bin 10 particular se escriben en un flujo de bits 12 parcial particular. El algoritmo de codificación global convierte los símbolos fuente 1 en dos o más flujos de bits parciales 12, en donde el número de flujos de bits parciales es igual al número de memorias intermedias de bin y codificadores de bin. Un codificador de bin 10 puede convertir un número variable de bins 9 en una palabra de código 11 de un número de bits variable. Una ventaja de la codificación PIPE descrita anteriormente y a continuación es que la codificación de bins puede realizarse en paralelo (por ejemplo, para diferentes grupos de medidas de probabilidad), lo que reduce el tiempo de procesamiento para varias implementaciones.

Otra ventaja de la codificación PIPE es que la codificación bin, que se realiza por los codificadores de bin 10, puede diseñarse específicamente para diferentes conjuntos de parámetros 5. En particular, la codificación y decodificación de bin puede optimizarse (en términos de eficiencia de codificación y/o complejidad) para diferentes grupos de probabilidades estimadas. Por otro lado, esto permite una reducción de la complejidad de codificación/decodificación con relación a los algoritmos de codificación aritmética con similar eficiencia de codificación. Por otro lado, permite una mejora de la eficiencia de codificación con relación a los algoritmos de codificación VLC con complejidad de codificación/decodificación similar. Los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación (es decir mapeado de secuencias de bin en palabras de código) para diferentes grupos de medidas para la estimación de la probabilidad para uno de los dos posibles valores de bin 5 para el bin actual. Los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación para diferentes grupos de medidas para una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin actual. Alternativamente, los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación para diferentes códigos de protección de canal. Los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación para diferentes esquemas de cifrado. Los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas para una estimación de la probabilidad para uno de los dos posibles valores de bin 5 para el bin actual. Los codificadores de bin 10 implementan diferentes algoritmos de codificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas para una estimación de la probabilidad para el valor de bin 5 menos probable o más probable para el bin actual. Los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas para una estimación de la probabilidad para uno de los dos posibles valores de bin 5 para el bin actual. Los codificadores de bin 10 pueden implementar diferentes algoritmos de codificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas para una estimación de la probabilidad para el valor de bin 5 menos probable o más probable para el bin actual.

Los codificadores de bin 10 —o uno más de los codificadores de bin— pueden representar motores de codificación aritmética binaria. Uno o más de los codificadores de bin puede representar un motor de codificación aritmético binario, en el que el mapeado desde la probabilidad LPS/LPB representativa p_{LPS} de una memoria intermedia de bin dada a un ancho de intervalo de código correspondiente R_{LPS} —es decir, la subdivisión de intervalos del estado de intervalos del motor de codificación aritmética binaria, que se define por el ancho del intervalo actual R y el desplazamiento del intervalo actual L , que identifica, por ejemplo, la delimitación inferior del intervalo de código— se realiza mediante el uso de una tabla de búsqueda. Para cada motor de codificación aritmética binaria basado en tabla asociado con una memoria intermedia de bin dada, pueden usarse K valores representativos del ancho de intervalo $\{Q_0, \dots, Q_{K-1}\}$ para la representación de R_{LPS} siendo la elección de K y los valores representativos del ancho de intervalo $\{Q_0, \dots, Q_{K-1}\}$ dependientes de la memoria intermedia de bin. Para una elección de $K > 1$, la codificación aritmética de un bin puede implicar las subetapas de mapeado del ancho de intervalo actual R a un índice de cuantificación q con valores en $\{0, \dots, K-1\}$ y realizando la subdivisión de intervalos accediendo al valor del ancho de intervalo parcial Q_q correspondiente desde una tabla de búsqueda usando q como un índice. Para una elección de $K=1$, es decir, para el caso en donde solo se da un valor del ancho de intervalo representativo Q_0 , este valor Q_0 puede elegirse como una potencia de dos para permitir la decodificación de múltiples valores de MPS/MPB que entran en la memoria intermedia de bin correspondiente dentro de un ciclo de normalización único. Las palabras de código resultantes para cada motor de codificación aritmética pueden transmitirse por separado, en paquetes, o almacenarse, o pueden intercalarse con la finalidad de transmisión o almacenamiento tal como se describe en el presente documento a continuación.

Esto es, un motor de codificación aritmética binaria 10 podría realizar las siguientes etapas en la codificación de los bins en su memoria intermedia de bin 8:

1. Recibir $vallPS$, bins desde la memoria intermedia de bin (recordatorio: el motor de codificación aritmética

binaria 10 respectivo considerado aquí se ha elegido para recibir "bin" (o, en otras palabras, se asocia "bin" con el motor de codificación aritmética binaria 10 respectivo) debido a que la estimación de la distribución de probabilidad, tal como $p_estado[bin]$, se asocia con ese motor de codificación aritmética binaria 10)

2. Cuantificación de R:

5

$$q_índice = Qtab[R \gg q]$$

(o alguna otra forma de cuantificación)

10

3. Determinación de R_{LPS} y R: $R_{LPS} = Rtab [q_índice]$ (nótese que p_estado no se ha mencionado aquí, dado que se fija para el motor de codificación de aritmética binaria 10 considerado, es decir $p_estado[codificador]$, y $Rtab$ ha almacenado en él los valores calculados para $p[p_estado[codificador]] \cdot Q[q_índice]$ R = R - R_{LPS} [esto es, R se preactualiza preliminarmente como si "bin" fuese MPS])

4. Cálculo del nuevo intervalo parcial:

si ($bin = 1 - valMPS$) entonces

15

$$L \leftarrow L + R$$

$$R \leftarrow RLPS$$

5. Renormalización de L y R, escribiendo bits,

en el que

20

$q_índice$ describe el índice de un valor de cuantificación leído de $Qtab$,
 p_estado describe el estado actual (fijado por el motor de codificación de aritmética binaria 10),
 R_{LPS} describe el ancho de intervalo correspondiente al LPS y
 $valMPS$ describe el valor del bit que corresponde al MPS.

25

En consecuencia, un motor de decodificación aritmética binaria 22 podría realizar las siguientes etapas en la decodificación de los bins de salida de la memoria intermedia 20 de bin:

1. Recibir la solicitud de un bin (recordatorio: el motor de decodificación aritmética binaria 22 respectivo considerado aquí se ha elegido para decodificar "bin" (o, en otras palabras, se asocia "bin" con el motor de decodificación aritmética binaria 22 respectivo) debido a que la estimación de la distribución de probabilidad, tal como $p_estado[bin]$, se asocia con ese motor de decodificación aritmética binaria 22)

30

2. Cuantificación de R:

35

$$q_índice = Qtab[R \gg q]$$

(o alguna otra forma de cuantificación)

3. Determinación de R_{LPS} y R:

40

$$R_{LPS} = Rtab [q_índice]$$

(nótese que p_estado no se ha mencionado aquí, dado que se fija para el motor de decodificación de aritmética binaria 22 considerado, es decir $p_estado[codificador]$, y $Rtab$ ha almacenado en él los valores precalculados para $p[p_estado[codificador]] \cdot Q[q_índice]$ R = R - R_{LPS} [esto es, R se preactualiza preliminarmente como si "bin" fuese MPS])

45

4. Determinación de bin dependiendo de la posición del intervalo parcial:

si ($V \geq R$) entonces

$bin \leftarrow 1 - valMPS$ (bin se decodifica como LPS; el selector 18 de memoria intermedia de bin obtendrá el valor de bin real mediante el uso de esta información de bin y $valMPS$)

50

$$V \leftarrow V - R$$

$$R \leftarrow RLPS$$

en caso contrario

$bin \leftarrow valMPS$ (bin se decodifica como MPS; el selector 18 de memoria intermedia de bin obtendrá el valor de bin real mediante el uso de esta información de bin y $valMPS$)

55

5. Renormalización de R, leyendo un bit y actualizando V,

en el que

60

$q_índice$ describe el índice de un valor de cuantificación leído de $Qtab$,
 p_estado describe el estado actual (fijado por el motor de decodificación de aritmética binaria 22),
 R_{LPS} describe el ancho de intervalo correspondiente al LPS,
 $valMPS$ describe el valor del bit que corresponde al MPS, y
 V describe un valor desde el interior del intervalo parcial actual.

Los codificadores de bin 10 —o uno o más de los codificadores de bin— puedan representar codificadores entrópicos que directamente mapean secuencias de bins de entrada 9 sobre palabras de código 10. Dichos mapeados pueden implementarse eficientemente y no requieren un motor de codificación aritmética complejo. El mapeado inverso de palabras de código sobre secuencias de bins (como se realiza en el decodificador) debería ser

5 único para garantizar una decodificación perfecta de la secuencia de entrada, pero al mapeado de las secuencias de bin 9 sobre palabras de código 10 no precisa necesariamente ser único, es decir, es posible que una secuencia particular de bins pueda mapearse sobre más de una secuencia de palabras de código. El mapeado de secuencias de bins de entrada 9 sobre palabras de código 10 puede ser también biyectiva. Preferentemente, los codificadores de bin 10 —o uno o más de los codificadores de bin— pueden representar codificadores entrópicos que

10 directamente mapean secuencias de longitud variable de bins de entrada 9 sobre palabras de código 10 de longitud variable. Las palabras de código de salida pueden representar códigos libres de redundancia tal como los códigos Huffman general o códigos Huffman canónicos.

Dos ejemplos de mapeado biyectivo de secuencias de bin a códigos libres de redundancia se ilustran en la Tabla 3. Las palabras de código de salida pueden representar códigos redundantes adecuados para detección de error y recuperación de error. Las palabras de código de salida pueden representar códigos de cifrado adecuados para el cifrado de los símbolos fuente.

15

Tabla 3: Ejemplos de mapeados entre secuencias de bin y palabras de código.

secuencias de bins (el orden del bin es de izquierda a derecha)	palabras de código (el orden de bits es de izquierda a derecha)
00000000	1
0000 0001	0000
0000 001	0001
0000 01	0010
00001	0011
0001	0100
001	0101
01	0110
1	0111
secuencias de bins (el orden del bin es de izquierda a derecha)	palabras de código (el orden de 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 bin 10 —o uno o más de los codificadores de bin— pueden representar codificadores entrópicos que mapean directamente secuencias de longitud variable de bins de entrada 9 en palabras de código 10 de longitud fija. Los codificadores de bin 10 —o uno o más de los codificadores de bin— representan codificadores entrópicos que mapean directamente secuencias de longitud fija de bins de entrada 9 en palabras de código 10 de longitud variable.

20

25

Se ilustra un decodificador PIPE en la Figura 4. El decodificador realiza básicamente las operaciones inversas del codificador de la Fig. 3, de modo que la secuencia de símbolos fuente 27 (previamente codificada) se decodifica a partir de un conjunto de dos o más flujos de bits 24 parciales. El decodificador incluye dos flujos de proceso diferentes: un flujo para las solicitudes de datos, que replica el flujo de datos del codificador, y un flujo de datos que representa la inversa del flujo de datos del codificador. En la ilustración de la Fig. 4, las flechas discontinuas representan el flujo de solicitud de datos, mientras que las flechas continuas representan el flujo de datos. Los bloques de construcción del decodificador replican básicamente los bloques de construcción del codificador, pero implementan las operaciones inversas.

30

35

La decodificación de un símbolo fuente se activa mediante una solicitud de un nuevo símbolo fuente 13 decodificado que se envía al binarizador 14. Cada solicitud de un nuevo símbolo fuente 13 decodificado puede asociarse con una categoría de un conjunto de una o más categorías. La categoría que se asocia con una solicitud de un símbolo fuente es la misma que la categoría que se asoció con el símbolo fuente correspondiente durante la codificación.

5 El binarizador 14 mapea la solicitud de un símbolo fuente 13 en una o más solicitudes para un bin que se envía al asignador de parámetros 16. Como la respuesta final a una solicitud para un bin que se envía al asignador de parámetros 16 por el binarizador 14, el binarizador 14 recibe un bin decodificado 26 desde el selector 18 de memoria intermedia de bin. El binarizador 14 compara la secuencia recibida de los bins decodificados 26 con las secuencias de bins de un esquema de binarización particular para el símbolo fuente solicitado y, si la secuencia recibida de bins decodificados 26 coincide con la binarización de un símbolo fuente, el binarizador vacía su memoria intermedia de bins y produce la salida de símbolos fuente decodificados como respuesta final a la solicitud de un nuevo símbolo decodificado. Si la secuencia ya recibida de bins decodificados no coincide con cualquiera de las secuencias de bins del esquema de binarización para el símbolo fuente solicitado, el binarizador envía otra solicitud de un bin al asignador de parámetros hasta que la secuencia de bins decodificados coincida con una de las secuencias de bins del esquema de binarización para el símbolo fuente solicitado. Para cada solicitud de un símbolo fuente, el decodificador usa el mismo esquema de binarización que se usó para la codificación del símbolo fuente correspondiente. El esquema de binarización puede ser diferente para diferentes categorías de símbolos fuente. El esquema de binarización para una categoría de símbolos fuente particular puede depender del conjunto de posibles valores de símbolos fuente y/u otras propiedades de los símbolos fuente para la categoría particular.

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

El asignador de parámetros 16 puede asociar cada solicitud de un bin con una medida de una estimación de la probabilidad de uno o más de los posibles valores de bin para el bin solicitado actual. En particular, el asignador de parámetros 16 puede asociar cada solicitud de un bin con una medida de una estimación de la probabilidad del valor de bin menos probable o más probable para el bin solicitado actual y un identificador que especifica la estimación para cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin solicitado actual.

El asignador de parámetros 16 puede asociar cada solicitud de un bin 15, 17 con una medida de una estimación de la probabilidad para uno de los dos posibles valores de bin para el bin solicitado actual y uno o más parámetros adicionales. El asignador de parámetros 16 puede asociar cada solicitud de un bin 15, 17 con una medida de una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin solicitado actual, un identificador que especifica la estimación para cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin solicitado actual y uno o más parámetros adicionales (que pueden ser uno o más de los parámetros anteriormente listados).

El asignador de parámetros 16 puede determinar una o más de las medidas de probabilidad anteriormente mencionadas (medida de una estimación de la probabilidad para uno de los dos posibles valores de bin para el bin solicitado actual, medida para una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin solicitado actual, identificador que especifica una estimación para cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para el bin solicitado actual) basándose en un conjunto de uno o más símbolos ya decodificados. La determinación de las medidas de probabilidad para una solicitud particular de un bin replica el proceso en el codificador para el bin correspondiente. Los símbolos decodificados que se usan para la determinación de las medidas de probabilidad pueden incluir uno o más símbolos ya decodificados de la misma categoría de símbolos, uno o más símbolos ya decodificados de la misma categoría de símbolos que corresponden a conjuntos de datos (tales como bloques o grupos de muestras) de localizaciones vecinas espaciales y/o temporales (en relación al conjunto de datos asociado con la solicitud actual de un símbolo fuente), o uno o más símbolos ya decodificados de diferentes categorías de símbolos que corresponden a conjuntos de datos de las mismas y/o vecinas localizaciones espaciales y/o temporales (en relación al conjunto de datos asociado con la solicitud actual de un símbolo fuente).

Cada solicitud de un bin con un conjunto asociado de parámetros 17 que es la salida del asignador de parámetros 16 se proporciona al interior de un selector 18 de memoria intermedia de bins. Basándose en el conjunto de parámetros 17 asociado, el selector 18 de memoria intermedia de bin envía una solicitud de un bin 19 a una de dos o más memorias intermedias 20 de bins y recibe un bin decodificado 25 desde la memoria intermedia 20 de bins seleccionada. El bin de entrada 25 decodificado se modifica potencialmente y el bin de salida 26 decodificado — potencialmente con un valor modificado— se envía al binarizador 14 como respuesta final a la solicitud de un bin con un conjunto de parámetros 17 asociado.

La memoria intermedia 20 de bin a la que se envía la solicitud de un bin se selecciona de la misma manera que la memoria intermedia de bin a la que se envió el bin de salida del selector de memoria intermedia de bin en el lado del codificador.

5 El selector 18 de memoria intermedia de bin puede determinar la memoria intermedia 20 de bin a la que se envía la solicitud de un bin 19 basándose en la medida asociada para una estimación de la probabilidad para uno de los dos posibles valores de bin para el bin solicitado actual. El conjunto de posibles valores para la medida de una estimación de la probabilidad para uno de los dos posibles valores de bin puede ser finita y el selector 18 de memoria intermedia de bin puede contener una tabla que asocia exactamente una memoria intermedia de bin 20 con cada posible valor de la estimación de la probabilidad para uno de los dos posibles valores de bin, en la que diferentes valores de la medida de una estimación de la probabilidad para uno de los dos posibles valores de bin puede asociarse con la misma memoria intermedia 20 de bin. El rango de valores posibles para la medida de una estimación de la probabilidad para uno de los dos posibles valores de bin puede particionarse en un número de intervalos, el selector 18 de memoria intermedia de bin determina el índice del intervalo para la medida actual de una estimación de la probabilidad para uno de los dos posibles valores de bin, y el selector 18 de memoria intermedia de bin contiene una tabla que asocia exactamente una memoria intermedia 20 de bin con cada posible valor para el índice de intervalo, en donde valores diferentes para el índice de intervalo pueden asociarse con la misma memoria intermedia 20 de bin. Solicitudes para bins 17 con medidas opuestas de una estimación de la probabilidad para uno de los dos posibles valores de bin (medidas opuestas son aquellas que representan estimaciones de probabilidad P y $1 - P$) pueden enviarse a la misma memoria intermedia 20 de bin. Adicionalmente, puede adaptarse a lo largo del tiempo la asociación de la medida de una estimación de la probabilidad para uno de los dos posibles valores de bin para la solicitud de bin actual con una memoria intermedia de bin particular.

25 El selector 18 de memoria intermedia de bin puede determinar la memoria intermedia 20 de bin a la que se envía la solicitud de un bin 19 basándose en la medida asociada de una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin solicitado actual. El conjunto de posibles valores para la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable puede ser finita y el selector 18 de memoria intermedia de bin puede contener una tabla que asocia exactamente una memoria intermedia 20 de bin con cada posible valor de la estimación de la probabilidad para el valor de bin menos probable o más probable, en donde pueden asociarse diferentes valores para la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable con la misma memoria intermedia 20 de bin. El rango de posibles valores para la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable puede particionarse en un número de intervalos, el selector 18 de memoria intermedia de bin determina el índice del intervalo para la medida actual de una estimación de la probabilidad para el valor de bin menos probable o más probable, y el selector 18 de memoria intermedia de bin contiene una tabla que asocia exactamente una memoria intermedia 20 de bin con cada valor posible para el índice de intervalo, en donde pueden asociarse diferentes valores para el índice de intervalo con la misma memoria intermedia 20 de bin. La asociación de la medida de una estimación de la probabilidad para el valor de bin menos probable o más probable para la solicitud de bin actual con una memoria intermedia de bin particular se adapta a lo largo del tiempo.

Después de recibir un bin decodificado 25 desde la memoria intermedia 20 de bin seleccionada, el selector 18 de memoria intermedia de bin modifica potencialmente el bin de entrada 25 y envía el bin de salida 26 —con un valor potencialmente modificado— al binarizador 14. El mapeado de entrada/salida de bin del selector 18 de memoria intermedia de bin en el inverso que el mapeado de entrada/salida de bin del selector de memoria intermedia de bin en el lado del codificador.

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

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

El selector 18 de memoria intermedia de bin puede determinar el valor del bin de salida 26 basándose en el valor del bin de entrada 25 y el identificador, que especifica la estimación de cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable para la solicitud de bin actual, que se asocia con la solicitud de un bin 17. El valor del bin de salida 26 puede fijarse igual al valor de bin de entrada 25 si el identificador

5 especifica que el primero de dos posibles valores de bin representa el valor de bin menos probable (o más probable) para la solicitud de bin actual, y el valor de bin de salida 26 se modifica (es decir, se fija opuesto al valor del bin de entrada) si el identificador especifica que el segundo de los dos valores de bin posibles representa el valor de bin menos probable (o más probable) para la solicitud de bin actual.

10 Como se ha descrito anteriormente, el selector de memoria intermedia de bin envía una solicitud un bin 19 a una de las dos o más memorias intermedias de bin 20. Las memorias intermedias de bin 20 representan memorias intermedias primero en entrar primero en salir, que se alimentan con secuencias de bin decodificadas 21 desde los decodificadores de bin 22 conectados. Como respuesta a una solicitud de un bin 19 que se envía a una memoria intermedia 20 de bin desde el selector 18 de memoria intermedia de bin, la memoria intermedia 20 de bin elimina de su contenido el bin que se suministró primero al interior de la memoria intermedia 20 de bin y lo envía al selector 18 de memoria intermedia de bin. Los bins que se enviaron antes a la memoria intermedia 20 de bin se retiran antes y se envían al selector 18 de memoria intermedia de bin.

20 Cada una de las dos o más memorias intermedias de bin 20 se conecta con exactamente un decodificador de bin 22 y cada decodificador de bin se conecta solamente con una memoria intermedia 20 de bin. Cada decodificador de bin 22 lee palabras de código 23, que representan secuencias de bits, desde un flujo de bits 24 parcial separado. El decodificador de bin convierte una palabra de código 23 en una secuencia de bins 21 que se envía a la memoria intermedia 20 de bin conectada. El algoritmo de decodificación global convierte dos o más flujos de bits 24 parciales en un número de símbolos fuente decodificados, en donde el número de flujos de bits parciales es igual al número de memorias intermedias de bin y decodificadores de bin y la decodificación de símbolos fuente se activa por las solicitudes de nuevos símbolos fuente. Un decodificador de bin 22 puede convertir palabras de código 23 de un número variable de bits en una secuencia de un número variable de bins 21. Una ventaja de la configuración PIPE anterior es que la decodificación de bins desde dos o más flujos de bits parciales puede realizarse en paralelo (por ejemplo, para diferentes grupos de medidas de probabilidad), lo que reduce el tiempo de procesamiento para varias implementaciones.

Otra ventaja de la decodificación PIPE anterior es que la decodificación de bin, que se realiza por los decodificadores de bin 22, puede diseñarse específicamente para diferentes conjuntos de parámetros 17. En particular, la codificación y decodificación de bin puede optimizarse (en términos de eficiencia de codificación y/o complejidad) para diferentes grupos de probabilidades estimadas. Por un lado, esto permite una reducción de la complejidad de codificación/decodificación con relación a los algoritmos de codificación aritmética con similar eficiencia de codificación. Por otro lado, permite una mejora de la eficiencia de codificación con relación a los algoritmos de codificación VLC con complejidad de codificación/decodificación similar. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación (es decir mapeados de secuencias de bin sobre palabras de código) para diferentes grupos de medidas de una estimación de la probabilidad para uno de los dos posibles valores de bin 17 para la solicitud de bin actual. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes grupos de medidas de una estimación de la probabilidad para el valor de bin menos probable o más probable para el bin solicitado actual. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes códigos de protección de canal. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes esquemas de cifrado. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas de la estimación de la probabilidad para uno de los dos posibles valores de bin 17 para el bin solicitado actual. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes combinaciones de códigos de protección de canal y grupos de medidas de una estimación de la probabilidad para el valor de bin 17 menos probable o más probable para el bin solicitado actual. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas de una estimación de la probabilidad para uno de los dos posibles valores de bin 17 para el bin solicitado actual. Los decodificadores de bin 22 pueden implementar diferentes algoritmos de decodificación para diferentes combinaciones de esquemas de cifrado y grupos de medidas de una estimación de la probabilidad para el valor de bin 17 menos probable o más probable para el bin solicitado actual.

Los decodificadores de bin 22 realizan el mapeado inverso de los codificadores de bin correspondientes en el lado del codificador.

60 Los decodificadores de bin 22 —o uno o más de los decodificadores de bin— pueden representar motores de decodificación aritmética binaria.

Los decodificadores de bin 22 —o uno o más de los decodificadores de bin— pueden representar decodificadores

entrópicos que mapean directamente palabras de código 23 en secuencias de bins 21. Dichos mapeados pueden implementarse eficientemente y no requieren un motor de codificación aritmética complejo. El mapeado de palabras de código en secuencias de bin ha de ser único. El mapeado de palabras de código 23 en secuencias de bin 21 puede ser biyectivo. Los decodificadores de bin 10 —o uno o más de los decodificadores de bin— pueden
 5 representar decodificadores entrópicos que mapean directamente palabras de código 23 de longitud variable en secuencias de bins 21 de longitud variable. Las palabras de código de entrada pueden representar códigos libres de redundancia tales como los códigos Huffman generales o los códigos Huffman canónicos. Se ilustran en la Tabla 3 dos ejemplos de mapeado biyectivo de códigos libres de redundancia en secuencias de bins. Las palabras de código de entrada pueden representar códigos redundantes adecuados para detección de error y recuperación de error. Las
 10 palabras de código de entrada pueden representar códigos cifrados.

Los decodificadores de bin 22 —o uno o más de los decodificadores de bin— pueden representar decodificadores entrópicos que mapean directamente palabras de código 23 de longitud fija en secuencias de bins 21 de longitud variable. Alternativamente los decodificadores de bin 22 —o uno o más de los decodificadores de bin— representan
 15 decodificadores entrópicos que mapean directamente palabras de código 23 de longitud variable en secuencias de bins 21 de longitud fija.

Así, la Fig. 3 y la Fig. 4 muestran un codificador PIPE para la codificación de una secuencia de símbolos fuente 1 y un decodificador PIPE para la reconstrucción de la misma. Esto es, el codificador PIPE de la Fig. 3 puede usarse
 20 como el codificador PIPE 104 en la Fig. 1a actuando un binarizador 2 como el simbolizador 122, actuando el asignador de parámetros 4 como el asignador 114, actuando el selector de memoria intermedia de bin 6 como el selector 120, y actuando el par de memorias intermedias de bin 8 y el codificador de bin 10 conectados en serie como uno respectivo de los codificadores entrópicos 116 cada uno de los cuales produce flujos de bits 12 correspondientes a los flujos de bits 118 en la Fig. 1a. Como queda claro a partir de la comparación de la Fig. 3 y la
 25 Fig. 1, el asignador 114 de la Fig. 1a puede tener sus entradas conectadas alternativamente al lado de entrada del simbolizador 122 en lugar de al lado de salida de este último. De modo similar, el decodificador PIPE de la Fig. 4 puede usarse como el decodificador PIPE 202 de la Fig. 2a correspondiendo los flujos de bits 24 parciales a los flujos de bits 216 en la Fig. 2, correspondiendo los pares de memorias intermedias 20 y decodificador de bin 22 conectados en serie a los decodificadores entrópicos 210 individuales, actuando el selector 18 de memoria
 30 intermedia de bin como el selector 214, actuando el asignador de parámetros 16 como el asignador 212 y actuando el binarizador 14 como el desimbolizador 222. De nuevo, una comparación entre la Fig. 2a y la Fig. 4 deja claro que la interconexión entre el desimbolizador 222, asignador 212 y selector 214 puede configurarse de modo diferente, de modo que, alternativamente, las conexiones de la Fig. 2a se añaden a las correspondientes mostradas en la Fig. 4. El codificador PIPE de la Fig. 3 comprende un asignador 4 configurado para asignar un número de parámetros 5 a cada símbolo de alfabeto de la secuencia de símbolos de alfabeto 3. La asignación se basa en información contenida dentro de los símbolos de alfabeto previos de la secuencia de símbolos de alfabeto de modo que la categoría del elemento de sintaxis 1 a cuya representación —tal como binarización— pertenece el símbolo de alfabeto actual y que, de acuerdo con la estructura de sintaxis de los elementos de sintaxis 1, se espera actualmente con expectación, a su vez, es deducible de la historia de los elementos de sintaxis 1 previos y símbolos de alfabeto
 35 3. Adicionalmente, el codificador comprende una pluralidad de codificadores entrópicos 10 cada uno de los cuales se configura para convertir los símbolos de alfabeto 3 enviados al codificador entrópico respectivo en un flujo de bits 12 respectivo, y un selector 6 configurado para enviar cada símbolo de alfabeto 3 a uno seleccionado de la pluralidad de codificadores entrópicos 10, dependiendo la selección del número de parámetros 5 asignado al símbolo de alfabeto respectivo 3. El decodificador PIPE de la Fig. 4 comprende una pluralidad de decodificadores entrópicos 22, cada uno de los cuales se configura para convertir un flujo de bits 23 respectivo en símbolos de alfabeto 21; un
 40 asignador 16 configurado para asignar un cierto número de parámetros 17 a cada símbolo 15 del alfabeto de una secuencia de símbolos de alfabeto a ser reconstruida basándose en información contenida dentro de símbolos de alfabeto previamente reconstruidos de la secuencia de símbolos de alfabeto (véase 26 y 27 en la Fig. 4); y un selector 18 configurado para recuperar cada símbolo de alfabeto de la secuencia de símbolos de alfabeto a ser reconstruida a partir de uno seleccionado de la pluralidad de decodificadores entrópicos 22, dependiendo la selección del número de parámetros definidos para el símbolo de alfabeto respectivo. El asignador 16 puede configurarse de modo que el número de parámetros asignados a cada símbolo de alfabeto comprende, o es, una medida de una estimación de una probabilidad de distribución entre los posibles valores de símbolos de alfabeto que puede asumir un símbolo de alfabeto respectivo. La secuencia de símbolos de alfabeto a ser reconstruida puede ser
 45 de un alfabeto binario y el asignador 16 puede configurarse de modo que la estimación de la distribución de probabilidad consiste en una medición de una estimación de una probabilidad de un valor de bin menos probable o más probable de los dos posibles valores de bin del alfabeto binario y un identificador que especifica la estimación de cuál de los dos posibles valores de bin representa el valor de bin menos probable o más probable. El asignador 16 puede configurarse adicionalmente para asignar internamente un contexto a cada símbolo de alfabeto de la
 50 secuencia de símbolos de alfabeto 15 a ser reconstruida basándose en la información contenida dentro de los símbolos de alfabeto previamente reconstruidos de la secuencia de símbolos de alfabeto a ser reconstruida teniendo cada contexto una estimación de distribución de probabilidad respectiva asociada con él, y adaptar la estimación de distribución de probabilidad para cada contexto a una estadística de símbolos actuales basándose en valores de símbolos de símbolos de alfabeto previamente reconstruidos a los que se asigna el contexto respectivo. El contexto

puede tener en cuenta una relación espacial o vecindad de posiciones a las que los elementos de sintaxis pertenecen tal como en codificación de video o imagen, o incluso en tablas en caso de aplicaciones financieras. A continuación, la medición de la estimación de la distribución de probabilidad para cada símbolo de alfabeto puede determinarse basándose en la estimación de distribución de probabilidad asociada con el contexto asignado al símbolo de alfabeto respectivo tal como mediante cuantificación de la estimación de distribución de probabilidad asociada con el contexto asignado con el símbolo de alfabeto respectivo a uno de una pluralidad de estimaciones de distribución de probabilidad representativas para obtener la medición para la estimación de la distribución de probabilidad. El selector puede configurarse de modo que se define una asociación sobreyectiva entre la pluralidad de codificadores entrópicos y la pluralidad de estimaciones de distribución de probabilidades representativas, es decir cada codificador entrópico tiene al menos una estimación de distribución de probabilidad representativa asociada con él, pero puede asociarse más de una estimación de distribución de probabilidad representativa con un codificador entrópico. La asociación puede ser incluso biyectiva. El selector 18 puede configurarse para cambiar un mapeado de cuantificación desde un rango en las estimaciones de distribución de probabilidad a la pluralidad de estimaciones de distribución de probabilidad representativas en una forma determinista predeterminada dependiendo de símbolos de alfabeto previamente reconstruidos de la secuencia de símbolos de alfabeto, a lo largo del tiempo. Esto es, el selector 18 puede cambiar los tamaños de la etapa de cuantificación, es decir los intervalos de distribuciones de probabilidad mapeados en índices de probabilidad individuales que, a su vez, pueden asociarse sobreyectivamente con los decodificadores entrópicos individuales. La pluralidad de decodificadores entrópicos 22, a su vez, puede configurarse para adaptar su forma de conversión de símbolos de alfabeto en flujos de bits en respuesta a un cambio en el mapeado de cuantificación. Por ejemplo, cada decodificador entrópico 22 puede optimizarse para, es decir puede tener una tasa de compresión óptima para, una cierta estimación de distribución de probabilidad dentro del intervalo de cuantificación de la estimación de distribución de probabilidad, y puede cambiar su mapeado de palabras de código/ secuencias de símbolos de modo que adapte la posición de esta cierta estimación de distribución de probabilidad dentro del intervalo de cuantificación de estimación de distribución de la probabilidad respectivo tras un cambio de este último, de modo que se optimice. El selector puede configurarse para cambiar el mapeado de cuantificación de modo que las tasas a las que se recuperan los símbolos de alfabeto desde la pluralidad de decodificadores entrópicos, se hace menos dispersa. Como con el binarizador 14 se hace notar que el mismo puede suprimirse si los elementos de sintaxis ya son binarios. Adicionalmente, dependiendo del tipo del decodificador 22, la existencia de las memorias intermedias 20 no es necesaria. Adicionalmente, las memorias intermedias pueden integrarse dentro de los decodificadores.

Hasta el momento, se han descrito más detalles para el codificador PIPE 104 y el decodificador PIPE 202 en las Figs. 1a y 2 anteriores, con respecto a las Figs. 3 y 4, lo que, si se implementa claramente en los aparatos de las Figs. 1a y 2, conduce a una producción de un flujo de bits paralelo en el que los flujos de bits parciales VLC y PIPE se transmiten en paralelo. En lo que sigue, se describen posibilidades sobre cómo combinar los flujos de bits parciales PIPE para que se transmitan, a continuación, junto con flujos de bits VLC en paralelo, o intercalando en segundo lugar ambos flujos de bits, es decir el flujo de bits VLC y el flujo de bits PIPE intercalado.

Finalización de secuencias de símbolos fuente finitos

En los codificadores y decodificadores PIPE, la codificación y decodificación puede realizarse para un conjunto finito de símbolos fuente. Frecuentemente se codifica una cierta cantidad de datos tal como una imagen fija, un cuadro o campo de una secuencia de video, un fragmento de imagen, un fragmento de un cuadro o un campo de una secuencia de video, un conjunto de muestras de audio sucesivas, etc. Para conjuntos finitos de símbolos fuente, en general, los flujos de bits parciales que se crean en el lado del codificador han de finalizarse, es decir, ha de asegurarse que todos los símbolos fuente pueden decodificarse a partir de los flujos de bits parciales transmitidos o almacenados. Después de que se inserte el último bin dentro de la memoria intermedia de bin 8 correspondiente, el codificador de bin 10 tiene que asegurar que se escribe una palabra de código completa en el flujo de bits 12 parcial. Si el codificador de bin 10 representa un motor de codificación aritmético binario, la palabra de código aritmética ha de finalizarse. El codificador de bin 10 representa un codificador entrópico que implementa un mapeado directo de secuencias de bin sobre palabras de código, la secuencia de bins que se almacena en la memoria intermedia de bin después de escribir el último bin en la memoria intermedia de bin podría no representar una secuencia de bins que se asocia con una palabra de código (es decir, podría representar un prefijo de dos o más secuencias de bin que se asocian con palabras de código). En dicho caso, cualquiera de las palabras de código asociadas con una secuencia de bins que contenga una secuencia de bins en la memoria intermedia de bin como prefijo ha de escribirse en el flujo de bits parcial (la memoria intermedia de bin ha de purgarse). Esto podría realizarse mediante la inserción de bins con un valor particular o uno arbitrario dentro de la memoria intermedia de bin hasta que se escriba una palabra de código. El codificador de bin puede seleccionar una de las palabras de código con longitud mínima (además de la propiedad de que la secuencia de bin asociada debe contener la secuencia de bins en la memoria intermedia de bin como prefijo). En el lado del decodificador, el decodificador de bin 22 puede decodificar más bins que los requeridos para la última palabra de código en un flujo de bits parcial; estos bins no son solicitados por el selector 18 de memoria intermedia de bin y se descartan e ignoran. La decodificación del conjunto finito de símbolos se controla mediante solicitudes de símbolos fuente decodificados; si no se solicitan símbolos fuente adicionales para la cantidad de datos, la decodificación se finaliza.

Transmisión y multiplexado de flujos de bits parciales

5 Los flujos de bits 12 parciales que se crean por el codificador PIPE pueden transmitirse por separado, o pueden multiplexarse en un único flujo de bits, o las palabras de código de los flujos de bits parciales pueden intercalarse en un único flujo de bits.

10 Cada flujo de bits parcial para una cantidad de datos puede escribirse en un paquete de datos. La cantidad de datos puede ser un conjunto arbitrario de símbolos fuente tal como una imagen fija, un campo o cuadro de una secuencia de vídeo, un fragmento de una imagen fija, un fragmento de un campo o cuadro de una secuencia de vídeo, o un cuadro de muestras de audio, etc.

15 Dos o más de los flujos de bits parciales para la cantidad de datos o todos los flujos de bits parciales para una cantidad de datos pueden multiplexarse en un paquete de datos. La estructura de un paquete de datos que contenga flujos de bits parciales multiplexados se ilustra en la Figura 5. Esto es, el paquete de datos mostrado en la Fig. 5 podría ser parte del flujo intercalado 132 y 234 intermedio, respectivamente.

20 El paquete de datos 300 consiste en una cabecera y una partición de los datos para cada flujo de bits parcial (para la cantidad de datos considerada). La cabecera 300 del paquete de datos contiene indicaciones para la partición del (resto del) paquete de datos en segmentos de datos de flujo de bits 302. Junto a las indicaciones para la partición, la cabecera puede contener información adicional. Los indicadores para la partición del paquete de datos pueden ser localizaciones del inicio de los segmentos de datos en unidades de bits o bytes o múltiplos de bits o múltiplos de bytes. Las localizaciones del inicio de los segmentos de datos pueden codificarse como valores absolutos en la cabecera del paquete de datos. O bien con relación al inicio del paquete de datos o en relación al final de la cabecera o en relación al inicio del paquete de datos previo. Las localizaciones del inicio de los segmentos de datos pueden codificarse diferencialmente, es decir, solo se codifica la diferencia entre el inicio real del paquete de un segmento de datos y una predicción para el inicio del segmento de datos. La predicción pueden deducirse basándose en información ya conocida o transmitida tal como el tamaño global del paquete de datos, el tamaño de la cabecera, el número de segmentos de datos en el paquete de datos, la localización del inicio de los segmentos de datos precedentes. La localización del inicio del primer paquete de datos puede no estar codificada, sino deducirse basándose en el tamaño de la cabecera del paquete de datos. En el lado del decodificador, las indicaciones de partición transmitidas se usan para deducir el inicio de los segmentos de datos. Los segmentos de datos se usan entonces como flujos de bits parciales y los datos contenidos en los segmentos de datos se proporcionan al interior de los decodificadores de bin correspondientes en orden secuencial.

35 Hay varias alternativas para multiplexado de los flujos de bits 12 parciales en un paquete de datos. Una alternativa, que puede reducir el requisito de información lateral, en particular para casos en los que los tamaños de los flujos de bits parciales son muy similares, se ilustra en la Fig. 6. La carga útil del paquete de datos, es decir, el paquete de datos 310 sin su cabecera 311, se particiona en segmentos 312 en una forma predefinida. Como un ejemplo, la carga útil del paquete de datos puede particionarse en segmentos del mismo tamaño. Entonces se asocia cada segmento con un flujo de bits parcial o con la primera parte de un flujo de bits parcial 313. Si un flujo de bits parcial es mayor que el segmento de datos asociado, su resto 314 se coloca dentro del espacio no usado en el extremo de los otros elementos de datos. Esto puede realizarse en una forma en la que la parte restante de un flujo de bits se inserta en orden inverso (comenzando desde el final del segmento de datos), lo que reduce la información lateral. La asociación de los restos de los flujos de bits parciales a segmentos de datos y, cuando se añade más de un resto al segmento de datos, el punto de inicio para uno más de los restos han de señalizarse dentro del flujo de bits, por ejemplo en la cabecera del paquete de datos.

Intercalado de palabras de código de longitud variable

50 Para algunas aplicaciones, el multiplexado anteriormente descrito de flujos de bits 12 parciales (para una cantidad de símbolos fuente) en un paquete de datos puede tener las siguientes desventajas: por un lado, para pequeños paquetes de datos, el número de bits de la información lateral que se requiere para señalización de la partición puede convertirse en significativo en relación a los datos reales en el flujo de bits parcial, lo que finalmente reduce la eficiencia de codificación. Por otro lado, el multiplexado puede no ser adecuado para aplicaciones que requieran un bajo retardo (por ejemplo, para aplicaciones de videoconferencia). Con el multiplexado descrito, el codificador PIPE no puede iniciar la transmisión de un paquete de datos antes de que los flujos de bits parciales se hayan creado completamente, dado que las localizaciones de inicio de las particiones no son conocidas antes. Adicionalmente, en general, el decodificador PIPE ha de esperar hasta que reciba el inicio del último segmento de datos antes de que pueda iniciar la decodificación de un paquete de datos. Para aplicaciones tales como sistemas de videoconferencia, estos retardos pueden añadirse a un retardo global adicional del sistema de varias imágenes de vídeo (en particular para tasas de bits que estén próximas a la tasa de bits de transmisión y para codificadores/decodificadores que requieran prácticamente el intervalo de tiempo entre dos imágenes para codificación/decodificación de una imagen), lo que es crítico para dichas aplicaciones. Para superar las desventajas para ciertas aplicaciones, el codificador

PIPE puede configurarse en una forma en la que las palabras de código que se generen por los dos o más codificadores de bin se intercalan en un único flujo de bits. El flujo de bits con las palabras de código intercaladas pueden enviarse directamente al decodificador (cuando se desprecia un pequeño retardo de la memoria intermedia, véase a continuación). En el lado del decodificador PIPE, los dos o más decodificadores de bin leen las palabras de código directamente desde el flujo de bits en orden de decodificación; la decodificación puede iniciarse con el primer bit recibido. Además, no se requiere ninguna información lateral para señalización del multiplexado (o intercalado) de los flujos de bits parciales.

La estructura básica de un codificador PIPE con intercalado de palabras de código se muestra en la Fig. 7. Los codificadores de bin 10 no escriben las palabras de código directamente en los flujos de bits parciales, sino que se conectan con una única memoria intermedia 29 de palabras de código, desde la que se escriben las palabras de código en el flujo de bits 34 en orden de codificación. Los codificadores de bin 10 envían solicitudes de una o más entradas 28 en la memoria intermedia de palabras de código a la memoria intermedia 29 de palabras de código y posteriormente envían las palabras de código 30 a la memoria intermedia 29 de palabras de código, que se almacenan en entradas de la memoria intermedia reservadas. Se accede a las palabras de código 31 (en general de longitud variable) de la memoria intermedia 29 de palabras de código por un escritor 32 de palabras de código, que escribe los bits correspondientes 33 en el flujo de bits 34 producido. La memoria intermedia 29 de palabras de código ópera como una memoria intermedia primero en entrar primero en salir; las entradas de palabras de código que se reservan antes se escriben antes en el flujo de bits.

En una generalización adicional, son posibles múltiples memorias intermedias de palabras de código y flujos de bits 12 parciales, en donde el número de memorias intermedias de palabras de código es menor que el número de codificadores de bin. Un codificador de bin 10 reserva una o más palabras de código en la memoria intermedia 29 de palabras de código, mediante lo que la reserva de una o más palabras de código en la memoria intermedia de palabras de código es activada por ciertos eventos en la memoria intermedia de bin 8 conectada. La memoria intermedia 29 de palabras de código puede funcionar en una manera en la que el decodificador PIPE puede decodificar instantáneamente el flujo de bits 34 que corresponde a 132 en la Fig. 1a y 134 en la Fig. 2, respectivamente. El orden de codificación en el que se escriben las palabras de código en el flujo de bits es el mismo que el orden en el que se reservan las palabras de código correspondientes en la memoria intermedia de palabras de código. Cada codificador de bin 10 puede reservar una palabra de código, siendo activada la reserva por un cierto evento en la memoria intermedia de bin conectada. Cada codificador de bin 10 puede reservar más de una palabra de código, siendo activada la reserva por un cierto evento en la memoria intermedia de bin conectada. Los codificadores de bin 10 pueden reservar una cantidad diferente de palabras de código, en donde la cantidad de palabras de código que se reserva por un codificador de bin particular puede depender del codificador de bin particular y/u otras propiedades del codificador de bin/memoria intermedia de bin particular (tal como la medida de probabilidad asociada, el número de bits ya escritos, etc.).

La memoria intermedia de palabras de código puede funcionar como sigue. Si se envía un nuevo bin 7 a una memoria intermedia de bin 8 particular y el número de bins ya almacenado en la memoria intermedia de bin es cero y no hay actualmente reservada ninguna palabra de código en la memoria intermedia de palabras de código para el codificador de bin que está conectado con la memoria intermedia de bin particular, el codificador de bin 10 conectado envía una solicitud a la memoria intermedia de palabras de código, mediante la que se reservan una o más entradas de palabras de código en la memoria intermedia 29 de palabras de código para el codificador de bin particular. Las entradas de palabras 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 normalmente por el tamaño máximo de la palabra de código para el codificador de bin correspondiente. La siguiente palabra de código o las siguientes palabras de código que se producen por el codificador de bin (para el que se ha reservado la entrada de palabras de código o las entradas de palabras 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 bin particular se llenan con palabras de código y se envía el siguiente bin a la memoria intermedia de bin que está conectada con el codificador de bin particular, se reservan una o más palabras de código nuevas en la memoria intermedia de palabras de código para el codificador de bin particular, etc. La memoria intermedia 29 de palabras de código representa una memoria intermedia primero en entrar primero en salir en una cierta forma. Las entradas a la memoria intermedia se reservan en orden secuencial. Las palabras de código para las que se han reservado antes las entradas de memoria intermedia correspondientes se escriben antes en el flujo de bits. El escritor 32 de palabras de código comprueba el estado de la memoria intermedia 29 de palabras de código, tanto continuamente como después de que se escriba una palabra de código 30 en la memoria intermedia 29 de palabras de código. Si la primera entrada a la memoria intermedia contiene una palabra de código completa (es decir, la entrada de la memoria intermedia no está reservada, pero incluye una palabra de código), la correspondiente palabra de código 31 y la correspondiente entrada de la memoria intermedia se eliminan de la memoria intermedia 20 de palabras de código y los bits de la palabra de código 33 se escriben en el flujo de bits. Este proceso se repite hasta que la primera entrada de la memoria intermedia no contiene una palabra de código (es decir, está reservada o libre). Al final del proceso de decodificación, es decir, si se han procesado todos los símbolos fuente de la cantidad de datos considerada, la memoria intermedia de palabras de código debe purgarse. Para ese

proceso de purga, se aplica lo siguiente para cada memoria intermedia de bin/codificador de bin como una primera etapa: si la memoria intermedia de bin no contienen bins, se añade un bin con un valor particular o un valor arbitrario hasta que la secuencia de bins resultante representa una secuencia de bins que está asociada con una palabra de código (como se hecho notar anteriormente, una forma preferida de añadir bins es añadir dichos valores de bin que producen la palabra de código más corta posible —o una de ellas— que se asocia con una secuencia de bins que contiene el contenido original de la memoria intermedia de bin como prefijo), entonces se escribe la palabra de código en la siguiente entrada de la memoria intermedia reservada para el codificador de bin correspondiente y se vacía la (correspondiente) memoria intermedia de bin. Si se ha reservado más de una entrada de memoria intermedia para uno o más codificadores de bin, la memoria intermedia de palabras de código puede contener aún entradas de palabras de código reservadas. En ese caso, estas entradas de palabras de código se rellenan con palabras de código arbitrarias, pero válidas, para los codificadores de bin correspondientes. Preferentemente, se inserta la palabra de código válida más corta o una de las palabras de código válidas más cortas (si hay múltiples). Finalmente, todas las palabras de código restantes en la memoria intermedia de palabras de código se escriben en el flujo de bits.

Se ilustran en la Fig. 8 dos ejemplos para el estado 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 la memoria intermedia libre. La primera entrada se llena con una palabra de código (es decir, el codificador de bin 2 solo escribe una palabra de código en una entrada previamente reservada). En la siguiente etapa, esta palabra de código se eliminará de la memoria intermedia de palabras de código y se escribe en el flujo de bits. A continuación, la primera palabra de código reservada para el codificador de bin 3 es la primera entrada de memoria intermedia, pero esta entrada no puede eliminarse de la memoria intermedia de palabras de código, dado que solo está reservada, pero no se ha escrito ninguna palabra de código 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 ello el escritor de palabras de código no puede escribir una palabra de código al flujo de bits. Aunque están contenidas 3 palabras de código en la memoria intermedia de palabras de código, el escritor de palabras de código ha de esperar hasta que se escribe la palabra de código en la primera entrada de memoria intermedia reservada para el codificador de bin 3. Obsérvese que las palabras de código deben escribirse en el orden en que se reservaron. Para ser capaz de invertir el proceso en el lado del decodificador (véase a continuación).

La estructura básica de un decodificador PIPE con intercalado de palabras de código se muestra en la Fig. 9. Los decodificadores bin 10 no leen las palabras de código directamente de flujos de bits parciales separados, sino que se conectan a una memoria intermedia 38 de bits, desde la que se leen las palabras de código 37 en el orden de codificación. Debería tomarse nota que no se requiere necesariamente la memoria intermedia 38 de bits, dado que las palabras de código podrían leerse directamente desde el flujo de bits. La memoria intermedia 38 de bits se incluye principalmente en la ilustración para separar claramente aspectos diferentes de la cadena de procesamiento. Los bits 39 del flujo de bits 40 con palabras de código intercaladas, que por ello corresponden al flujo de bits 234 en la Fig. 2, se insertan secuencialmente en la memoria intermedia 38 de bits, que representa una memoria intermedia primero en entrar primero en salir. Si un decodificador de bin 22 particular recibe una solicitud de una o más secuencias de bin 35, el decodificador de bin 22 lee una o más palabras de código 37 de la memoria intermedia 38 de bits a través de la solicitud de bits 36. El decodificador PIPE puede decodificar instantáneamente los símbolos fuente. Obsérvese que el codificador PIPE (tal como se ha descrito anteriormente) debe asegurar mediante la operación en forma adecuada de la memoria intermedia de palabras de código que las palabras de código se escriben en el mismo orden al flujo de bits en el que se solicitan por los decodificadores de bin. En el decodificador PIPE, todo el proceso de decodificación se activa mediante solicitudes de símbolos fuente. Parámetros tales como el número de palabras de código que se reservan en el lado del codificador por un codificador de bin particular y el número de palabras de código que se leen por el decodificador de bin correspondiente deben ser los mismos.

En una generalización adicional, son posibles múltiples memorias intermedias de palabras de código y flujos de bits parciales, en donde el número de memorias intermedias de bit es menor que el número de decodificadores de bin. Un decodificador de bin 22 lee una o más palabras de código para la memoria intermedia 38 de bits en un instante de tiempo, mediante lo que la lectura de las una o más palabras de código de la memoria intermedia de bits es activada por ciertos eventos en la memoria intermedia 20 de bin conectada. El decodificador puede hacerse funcionar en una forma en la que una o más palabras de código se leen cuando se envía una solicitud de un bin 19 a una memoria intermedia 20 de bin particular y la memoria intermedia de bin no contiene ningún bin. Pero es posible también activar la lectura de las palabras de código por otros eventos, por ejemplo si el número de bins en la memoria intermedia de bin está por debajo de un umbral predefinido. Cada decodificador de bin 22 puede leer una palabra de código, siendo activada la lectura por un cierto evento en la memoria intermedia de bin conectada. Alternativamente, cada decodificador de bin 22 debe leer más de una palabra de código, siendo activada la lectura por un cierto evento en la memoria intermedia de bin conectada. Los decodificadores de bin 22 puede leer una cantidad diferente de palabras de código, en donde la cantidad de palabras de código que se lee por un decodificador de bin particular puede depender del decodificador de bin particular y/u otras propiedades del decodificador de bin / memoria intermedia de bin particular (tal como la medida de probabilidad asociada, el número

de bits ya leídos, etc.).

La lectura de palabras de código desde la memoria intermedia de bits puede funcionar como sigue. Si se envía una nueva solicitud de bin 19 desde el selector 18 de memoria intermedia de bin a una memoria intermedia 20 de bin particular y el número de bin en la memoria intermedia de bin es cero, el decodificador de bin 22 conectado lee una o más palabras de código 37 de la memoria intermedia 38 de bits, a través de la solicitud de bits 36 a la memoria intermedia 38 de bits. El decodificador de bin 22 convierte las palabras de código leídas 37 en secuencias de bins 21 y almacena estas secuencias de bin en la memoria intermedia 20 de bin conectada. Como respuesta final a la solicitud de un bin 19, el primer bin insertado se elimina de la memoria intermedia 20 de bin y se envía al selector 18 de memoria intermedia de bin. Como respuesta a solicitudes de bin adicionales, los bin restantes en la memoria intermedia de bin se eliminan hasta que se vacía la memoria intermedia de bin. Una solicitud de bin adicional activa el decodificador de bin para leer una o más nuevas palabras de código desde la memoria intermedia de bits, etc. La memoria intermedia 38 de bits representa una memoria intermedia primero en entrar primero en salir de un tamaño predefinido y se llena continuamente con bits 39 del flujo de bits 40. Para asegurar que las palabras de código se escriben en el flujo de bits en la misma forma en la que se solicitan por el proceso de decodificación, la memoria intermedia de palabras de código en el lado del codificador puede funcionar en la forma descrita anteriormente.

De ese modo, cada uno de la pluralidad de decodificadores entrópicos puede ser un decodificador de longitud variable configurado para mapear palabras de código de longitudes fijas en secuencias de símbolos de longitudes variables, y puede proporcionarse una entrada de palabra de código tal como la salida de la memoria intermedia de palabras de código 43 para la recepción de un único flujo de palabras de código intercaladas. La pluralidad de decodificadores 22 entrópicos puede configurarse para recuperar las palabras de código de la entrada de palabras de código en un orden secuencial que depende de un orden en el que se recuperan los símbolos de la secuencia de símbolos a ser reconstruidos por el selector 18 de la pluralidad de decodificadores entrópicos dando como resultado una nueva secuencia de símbolos a ser mapeada a partir de una palabra de código nueva en los decodificadores entrópicos respectivos.

Intercalado de palabras de código de longitud variable con restricción de bajo retardo

El intercalado de palabras de código descrito para la codificación PIPE no requiere que se envíe ninguna información de partición como información lateral. Y dado que las palabras de código se intercalan en el flujo de bits, el retardo es en general pequeño. Sin embargo, no se garantiza que se obedezca a 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). Adicionalmente, el tamaño de la memoria intermedia requerido para memoria intermedia de palabras de código puede hacerse teóricamente muy grande. Cuando se considera el ejemplo de la Fig. 8(b), sería posible que no se envíen bins adicionales a la memoria intermedia 3 de bin y por ello el codificador de bin 3 no enviará ninguna nueva palabra de código a la memoria intermedia de palabras de código hasta que se aplique el proceso de purgado al final del paquete de datos. Entonces todas las palabras de código para los codificadores de bin 1 y 2 tendrían que esperar hasta el final del paquete de datos, antes de que puedan escribirse en el flujo de bits. Este inconveniente puede sortearse añadiendo un mecanismo adicional al proceso de codificación PIPE (y también al proceso de decodificación PIPE tal como se describe más adelante). El concepto básico de ese mecanismo adicional es que si una medida en relación al retardo o a una delimitación superior del retardo (véase a continuación) supera un umbral especificado, la primera entrada en la memoria intermedia reservada se llena mediante el purgado de la memoria intermedia de bin correspondiente (usando un mecanismo similar al del final del paquete de datos). Mediante dicho mecanismo, el número de entradas de memoria intermedia en espera se reduce hasta que la medida del retardo asociada sea menor que el umbral especificado. En el lado del decodificador, los bin que se han insertado en el lado del codificador para obedecer a la recepción de retardo deben descartarse. Para este descarte de los bins puede usarse básicamente el mismo mecanismo que en el lado del codificador. En lo que sigue se describen dos posibilidades para dicho control del retardo.

En una posibilidad, la medida para el retardo (o una delimitación 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, dado que si la primera entrada de memoria intermedia contiene una palabra de código, esta palabra de código se escribe en el flujo de bits. Si, por ejemplo, el retardo máximo de memoria intermedia permitido (tal como se determina por la aplicación) es D bits y el tamaño de palabra de código máximo para todos los codificadores bin es L , una delimitación inferior para el número máximo de palabras de código que puede contenerse en la memoria intermedia de palabras de código sin violar la restricción de retardo puede calcularse por $N = D/L$. La media del retardo D en bits no es requerida por el sistema, pero el número máximo de palabras de código N debe ser conocido tanto para el codificador como para el decodificador. El número máximo de entradas de memoria intermedia de palabras de código N puede fijarse con la aplicación. Alternativamente, el número máximo de entradas de memoria intermedia de palabras de código N puede señalizarse en el interior del flujo de bits, por ejemplo, en la cabecera del

paquete de datos (o cabecera del fragmento) o en un conjunto de parámetros, que se incluye en un flujo de bits. Si un codificador de bin 10 envía una solicitud para la reserva de una o más nuevas entradas de memoria intermedia a la memoria intermedia 29 de palabras de código, se ejecuta el proceso siguiente antes de que se reserve una nueva entrada de memoria intermedia de palabras de código (es decir, se ejecutan múltiples veces si se reservan múltiples entradas de memoria intermedia de palabras de código mediante una solicitud): si el número de entradas de memoria intermedia actualmente activas 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 (que se reserva) se purga mediante los procesos descritos a continuación hasta que el número de entradas de memoria intermedia actualmente activas más 1 sea menor que o igual a el número máximo de entradas de memoria intermedia de palabras de código N . El purgado de una entrada de memoria intermedia de palabras de código es similar al purgado al final de un paquete de datos. El codificador de bin 10 que ha reservado la primera entrada de memoria intermedia correspondiente es purgado mediante la adición de bins con valores particulares o arbitrarios a la memoria intermedia 8 de bin conectada hasta que la secuencia de bins resultante represente una secuencia de bin que está asociada con una palabra de código, la palabra de código se escribe a continuación a la entrada de memoria intermedia reservada y se añade finalmente al flujo de bits (mientras se vacía la memoria intermedia de bins y se eliminan las entradas de memoria intermedia previamente reservadas). Como se ha mencionado anteriormente, una forma preferida para añadir bins a la memoria intermedia de bin es añadir aquellos bins que producen la palabra de código más corta posible. En el lado del decodificador, se ejecuta un proceso similar para descartar los bin que se han añadido para obedecer a la restricción de retardo. Por lo tanto, el decodificador mantiene un contador C que cuenta las palabras de código que se han leído desde la memoria intermedia de bits (este contador puede mantenerse en la memoria intermedia de bits). El contador C se inicializa (por ejemplo, a cero) al inicio de la decodificación de un paquete de datos se incrementa en uno después de que se lea una palabra de código. Además, cada decodificador de bin 22 contiene un contador C_x , que almacena el valor del contador de palabras de código C antes de que se lea la última palabra de código por el decodificador de bin 22 correspondiente. Es decir, cuando un decodificador de bin 22 particular lee una nueva palabra de código, su contador C_x se fija igual a C como una primera etapa y a continuación la palabra de código se lee desde la memoria intermedia de bits. Cuando se envía una solicitud de un bin 19 a una memoria intermedia 20 de bin particular y la diferencia ($C - C_x$) entre el contador de palabras de código global C y el contador C_x del decodificador de bin 22 conectado es mayor que el número máximo de entradas de memoria intermedia de palabras de código N , todos los bin que se almacenan actualmente en la memoria intermedia 20 de bin particular se descartan e ignoran. Junto a esa etapa adicional, la decodificación se hace funcionar como se ha descrito anteriormente. Si la memoria intermedia 20 de bin a la que se envía una solicitud de un bin 19 está vacía (bien porque todos los bin que se han eliminado o porque un mecanismo de bajo retardo descartó todos los bin en la primera etapa después de que se haya recibido la solicitud del bin), el decodificador de bin 22 conectado lee una o más palabras de código desde la memoria intermedia 38 de bits, etc.

La medida del retardo (o una delimitación superior del retardo) puede ser la suma de las longitudes máximas de palabras de código para las entradas de memoria intermedia activas en la memoria intermedia de palabras de código, en donde una longitud máxima de palabras de código para una entrada de memoria intermedia particular depende del bin decodificado que se asocia con esa entrada de memoria intermedia. Como ilustración, las longitudes máximas de palabra de código para las entradas de memoria intermedia se indican 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, dado que si la primera entrada de memoria intermedia contiene una palabra de código, esta palabra de código se escribe al flujo de bits. Sea el retardo máximo de memoria intermedia permitido (tal como se determina por la aplicación) de D bits. El retardo máximo de memoria intermedia D debe ser conocido tanto para el codificador como para el decodificador. El retardo máximo de memoria intermedia D puede fijarse por la aplicación. El retardo máximo de memoria intermedia D puede señalizarse en el interior del flujo de bits, por ejemplo, en la cabecera del paquete de datos (o cabecera del fragmento) o en un conjunto de parámetros, que se incluye en el flujo de bits. Puede señalizarse en unidades de bits, o bytes, o un múltiplo de bits, o un múltiplo de bytes. Si un codificador de bin 10 envía una solicitud para la reserva de una o más nuevas entradas de memoria intermedia a la memoria intermedia 29 de palabras de código, se ejecuta el proceso siguiente antes de que se reserve una nueva entrada de memoria intermedia de palabras de código (es decir, se ejecuta múltiples veces si se reservan múltiples entradas de memoria intermedia de palabras de código mediante una solicitud).

Si la suma de las longitudes máximas de palabras de código para todas las entradas de memoria intermedia actualmente activas más la longitud máxima de palabra de código para la entrada de memoria intermedia que se reservará es mayor que el retardo máximo de memoria intermedia D , la primera entrada de memoria intermedia (que se reserva) es purgada por el proceso descrito anteriormente hasta que la suma de las longitudes máximas de palabras de código para todas las entradas de memoria intermedia activas más la longitud máxima de palabra de código para la entrada de memoria intermedia que se reservará sea menor que o igual al retardo máximo de memoria intermedia D . Como un ejemplo, considérese el ejemplo en la Fig. 8(b). La suma de las longitudes máximas de palabras de código para todas las entradas de memoria intermedia actualmente activas es 29. Supóngase que el retardo máximo de memoria intermedia D se fija igual a 32. Si la siguiente entrada de memoria intermedia se reserva

por el codificador de bin 2 para el que la longitud máxima de palabra de código es igual a 3, la primera entrada de la memoria intermedia no se purga, dado que $29 + 3$ no es mayor que 32. Pero si la siguiente entrada de memoria intermedia es reservada por el codificador de bin 1 para el que la longitud máxima de palabra de código es igual a 7, la primera entrada de memoria intermedia se purga, dado que $29 + 7$ es mayor que 32. El purgado de la entrada de memoria intermedia reservada se realiza tal como se ha descrito anteriormente (mediante la adición de bin con valores particulares o arbitrarios desde la memoria intermedia de bin correspondiente).

En el lado del decodificador, se ejecuta un proceso similar para descartar los bin que ya se han añadido para obedecer a la restricción de retardo. Por lo tanto, el decodificador mantiene un contador C que cuenta la longitud máxima de palabras de código para las palabras de código que se han leído desde la memoria intermedia de bits (este contador puede mantenerse en la memoria intermedia de bits). Obsérvese que las longitudes máximas de palabra de código que se asocian con decodificadores de bin diferentes pueden ser diferentes. El contador C se inicializa (por ejemplo con cero) al inicio de la decodificación de un paquete de datos y 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 por su longitud máxima. Es decir, si una palabra de código es leída por un decodificador de bin particular y la longitud máxima de palabra de código que se asocia con la tabla de palabras de código usada por el decodificador de bin particular es Lx (un decodificador de bin diferente puede asociarse con una longitud máxima de palabra de código diferente), el contador C se incrementa por Lx . Además del contador global C , cada decodificador de bin 22 contiene un contador Cx , que almacena al valor del contador de palabras de código C antes de que se lea la última palabra de código por el decodificador de bin 22 correspondiente. Es decir, cuando un decodificador de bin 22 particular lee una nueva palabra de código, su contador Cx se fija igual a C como una primera etapa y a continuación la palabra de código se lee desde la memoria intermedia de bits. Cuando se envía una solicitud para un bin 19 a una memoria intermedia de bin 20 particular y la diferencia ($C - Cx$) entre el contador global C y el contador Cx del decodificador de bin 22 conectados es mayor que el retardo de memoria intermedia máximo D , todos los bins que están almacenados actualmente en la memoria intermedia 20 de bin particular se descartan e ignoran. Junto a esa etapa adicional, la decodificación funciona como se ha descrito anteriormente. Si la memoria intermedia 20 de bin a la que se envía la solicitud de un bin 19 está vacía (porque todos los bin ya se han eliminado o porque el mecanismo de bajo retardo descartó todos los bin en la primera etapa después de que se haya recibido la solicitud de bin), el decodificador de bin 22 conectado lee una o más nuevas palabras de código desde la memoria intermedia 38 de bits, etc.

Además, la pluralidad de decodificadores entrópicos 22 y el selector 18 pueden configurarse para descartar intermitentemente sufijos de secuencias de símbolos de modo que no participen en la formación de la secuencia de símbolos a ser reconstruida 29. El descarte de modo intermitente puede realizarse en eventos en donde un número de palabras de código que se hayan recuperado desde la entrada de palabras de código por la pluralidad de decodificadores entrópicos entre dos recuperaciones de palabras de código consecutivas de un decodificador entrópico respectivo desde la entrada de palabras de código, cumple con un criterio predeterminado. La pluralidad de codificadores entrópicos y la memoria intermedia de palabras de código pueden, a su vez, configurarse para extender de modo intermitente símbolos actualmente enviados pero aún no mapeados a secuencias de símbolos válidas mediante no tener en cuenta los símbolos actualmente enviados pero aún no mapeados como prefijo, mapear las secuencias de símbolos así extendidas en palabras de código, introducir las palabras de código así obtenidas en las entradas de palabras de código reservadas y purgar las entradas de palabras de código. La extensión, introducción y purgado en forma intermitente puede tener lugar con eventos en donde un número de entradas de palabras de código reservadas más un número de entradas de palabras de código que tengan palabras de código introducidas en él cumpla con un criterio predeterminado. El criterio predeterminado puede tener en cuenta las longitudes máximas de palabras de código de la pluralidad de pares codificador/decodificador.

Para algunas arquitecturas, la forma anteriormente descrita de intercalado de palabras de código podría dar como resultado un inconveniente en términos de complejidad de decodificación. Como se ilustra en la Fig. 9, todos los decodificadores de bin 22 leen palabras de código (en el caso general, palabras de código de longitud variable) desde una única memoria intermedia 38 de bits. La lectura de las palabras de código no puede realizarse en paralelo, dado que las palabras de código deben leerse en el orden correcto. Esto significa que, un decodificador de bin particular debe esperar hasta que otros decodificadores de bin acaben 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 al resto de los procesos de decodificación (parcialmente en paralelo), este acceso de las palabras de código de longitud variable puede ser un cuello de botella para todo el proceso de decodificación. Hay algunas variaciones que pueden emplearse para la reducción de la complejidad del acceso desde la única memoria intermedia de bits, unas pocas de las cuales se describirán en lo que sigue. En este caso, por ejemplo, existe un único conjunto de palabras de código (que representa por ejemplo un código de prefijo no redundante) y el conjunto de palabras de código que se usa para cada decodificador de bin 22 es un subconjunto de un único conjunto de palabras de código. Obsérvese que diferentes decodificadores de bin 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 decodificadores de bin 22 son el mismo, su asociación con las secuencias de bin es diferente para diferentes decodificadores de bin 22. El mismo conjunto de palabras de código puede usarse para todos los decodificadores de bin 22. Si se tiene un único conjunto

de palabras de código que incluye los conjuntos de palabras de código para todos los decodificadores de bin como subconjuntos, el análisis de las palabras de código puede realizarse fuera de los decodificadores de bin, lo que puede reducir la complejidad del acceso a la palabra de código. El proceso de codificación PIPE no se cambia en relación con el proceso anteriormente descrito. El proceso de decodificación PIPE modificado se ilustra en la Fig. 10.

5 Un único lector de palabras de código es alimentado con los bits 46 desde el flujo de bits 40 y analiza las palabras de código —en general de longitud variable—. Las palabras de código 44 leídas se insertan dentro de una memoria intermedia 43 de palabras de código, que representa una memoria intermedia primero en entrar primero en salir. Un decodificador de bin 22 envía una solicitud para una o más palabras de código 41 a la memoria intermedia 43 de palabras de código y como respuesta a esta solicitud, las una o más palabras de código se eliminan de la memoria intermedia de palabras de código (en orden secuencial) y se envían al decodificador de bin 22 correspondiente. Obsérvese que en este caso, el análisis de palabras de código potencialmente complejo puede realizarse en un proceso en segundo plano y no necesita esperar a los decodificadores de bin. Los decodificadores de bin acceden a palabras de código ya analizadas, el análisis de palabras de código potencialmente complejo no es parte de una solicitud a la memoria intermedia global. En su lugar las palabras de código ya analizadas se envían a los

10

15 decodificadores de bin, que pueden implementarse también en una forma en que solo se envían los índices de las palabras de código a los decodificadores de bin.

Intercalado de secuencias de bits de longitud fija

20 Una forma adicional de reducir la complejidad del decodificador PIPE puede conseguirse cuando los decodificadores de bin 22 no leen palabras de código de longitud variable desde la memoria intermedia de bits 38 global, sino en su lugar siempre leen secuencias de bits de longitud fija desde la memoria intermedia de bits 38 global y añaden estas secuencias de bits de longitud fija a una memoria intermedia de bits local, en donde cada decodificador de bin 22 se conecta con una memoria intermedia de bits local separada. Las palabras de código de longitud variable se leen a

25 continuación desde la memoria intermedia de bits local. Por ello, el análisis de las palabras de código de longitud variable puede realizarse en paralelo, solo el acceso de secuencias de bits de longitud fija ha de realizarse en una forma sincronizada, pero dicho acceso de secuencias de bits de longitud fija es normalmente muy rápido, de modo que la complejidad de decodificación global puede reducirse para algunas arquitecturas. El número fijo de bins que se envían a una memoria intermedia de bits local particular puede ser diferente para diferentes memorias intermedias de bits locales y también pueden variar a lo largo del tiempo, dependiendo de ciertos parámetros como eventos en el decodificador de bin, memoria intermedia de bin, o memoria intermedia de bits. Sin embargo, el número de bits que se leen en un acceso particular no depende de los bits reales que se hayan leído durante el acceso particular, lo que es la diferencia importante respecto a la lectura de palabras de código de longitud variable. La lectura de secuencias de bits de longitud fija se activa por ciertos eventos en las memorias intermedias de bin, decodificadores de bin, o memorias intermedias de bits locales. Como un ejemplo, es posible solicitar la lectura de una nueva secuencia de bits de longitud fija cuando el número de bits que están presentes en una memoria intermedia de bits conectada cae por debajo de un umbral predefinido, en donde pueden usarse diferentes valores de umbral para diferentes memorias intermedias de bits. En el codificador, ha de asegurarse que las secuencias de longitud fija de bin se insertan en el mismo orden dentro del flujo de bits, en el que se leen desde el flujo de bits en el

30

35 lado del decodificador. Es posible también combinar este intercalado de secuencias de longitud fija con un control de bajo retardo similar al explicado anteriormente. En lo que sigue, se describe una forma para intercalado de secuencias de bits de longitud fija.

45 La Fig. 11 muestra una realización de una estructura de codificador PIPE que intercala secuencias de bits de longitud fija para dos o más codificadores de bin. A diferencia de la Fig. 7, los codificadores de bin 10 no se conectan con una única memoria intermedia de palabras de código. En su lugar, cada codificador de bin 10 se conecta con una memoria intermedia de bits 48 separada, que almacena bits para el flujo de bits parcial correspondiente. Todas las memorias intermedias 48 de bits se conectan a una memoria intermedia de bit 51 global. La memoria intermedia de bits 51 global se conecta a un escritor de bits 53, que elimina los bits 52 en el orden de

50 codificación/decodificación desde la memoria intermedia de bits global y escribe los bits eliminados 54 en el flujo de bits 55. En un cierto evento en una memoria intermedia de bits 48 particular o el codificador de bin 10 conectado o la memoria intermedia de bin 8, la memoria intermedia de bits 48 envía una solicitud 49 a la memoria intermedia de bits 51 global mediante la que se reservan un cierto número de bits en la memoria intermedia de bit 51 global. La solicitud para la reserva de secuencias de bits 49 de longitud fija se procesa en orden secuencial. La memoria intermedia de bit 51 global representa una memoria intermedia primero en entrar primero en salir en una cierta forma; los bits que se reservan antes se escriben antes en el flujo de bits. Debería tomarse nota que diferentes memorias intermedias de bit 48 pueden reservar una cantidad diferente de bits, lo que también puede variar a lo largo del tiempo basándose en símbolos ya codificados; pero el número de bits que se reservan para una solicitud particular es conocido en el momento en el que se envía la solicitud a la memoria intermedia de bits global.

60 En una particular, las memorias intermedias de bit 48 y la memoria intermedia de bit 51 global se hacen funcionar como se describe en lo que sigue. La cantidad de bits que se reserva por una memoria intermedia de bits 48 particular se indica por N_x . Este número de bits N_x puede ser diferente para diferentes memorias intermedias de bits 48 y puede también variar a lo largo del tiempo. El número de bits N_x que se reserva para una memoria intermedia

de bits particular 48 puede ser fijo a lo largo del tiempo. Las reservas para un número fijo N_x de bits 49 se activan basándose en el número de bits M_x en la memoria intermedia de bits 48, el número de bits N_x para las solicitudes de reserva, y la longitud máxima de palabras de código L_x asociada. Obsérvese que cada codificador de bin 10 puede asociarse con una longitud máxima de palabra de código L_x diferente. Si un bin 7 se envía a una memoria intermedia de bin 8 particular, y la memoria intermedia de bin 8 particular está vacía, y no se reserva más de una secuencia de N_x bits en la memoria intermedia de bits global para la memoria intermedia de bits 48 que se conecta con la memoria intermedia de bin particular (a través de un codificador de bin), y la diferencia $N_x - M_x$ entre el número N_x de bits que se reservan mediante una solicitud de reserva de la memoria intermedia de bits 48 que está conectada (a través de un codificador de bin) con la memoria intermedia de bin 8 particular y el número de bits M_x que está actualmente presente en esta memoria intermedia de bits 48 es menor que la longitud máxima de palabra de código L_x que está asociada con el codificador de bin 10 correspondiente, la memoria intermedia de bits 49 conectada envía una solicitud 49 para la reserva de N_x bits a la memoria intermedia de bit 51 global. La memoria intermedia de bits 51 global reserva N_x bits para la memoria intermedia de bits 48 particular e incrementa su puntero para la siguiente reserva. Después de que se hayan reservado los N_x bits en la memoria intermedia de bits global, el bin 7 se almacena en la memoria intermedia de bin 8. Si este único bin no representa ya una secuencia de bins que está asociada con una palabra de código, el codificador de bin 10 elimina este bin de la memoria intermedia de bin 8 y escribe la palabra de código 47 correspondiente en la memoria intermedia de bits 48 conectada. En caso contrario (este único documento de bin ya presente en una secuencia de bins que está asociada con una palabra de código), se aceptan bins 7 adicionales por la memoria intermedia de bin 8 particular hasta que la memoria intermedia de bin 8 contenga una secuencia de bin que está asociada con una palabra de código. En este caso, el codificador de bin 10 conectado elimina la secuencia de bins 9 de la memoria intermedia de bin 8 y escribe la palabra de código 47 correspondiente en la memoria intermedia de bits 48 conectada. Si el número resultante de bits M_x en la memoria intermedia de bits 48 es mayor que o igual a el número de bits N_x reservado, los N_x bits que se escribieron primero en la memoria intermedia de bits 48 se insertan en el espacio previamente reservado en la memoria intermedia de bits 51 global. Para el siguiente bin 7 que se envía a la memoria intermedia de bin 8 particular, se ejecuta el mismo proceso especificado anteriormente; es decir, se comprueba primero si debe reservarse un nuevo número de N_x bits en la memoria intermedia de bits global (si $N_x - M_x$ es menor que L_x) y a continuación se inserta el bin en la memoria intermedia de bin 8, etc. El escritor de bits escribe las secuencias de bits de longitud fija de la memoria intermedia de bits global en el orden en que se han reservado. Si la primera entrada de longitud fija en la memoria intermedia de bit 51 global contiene una secuencia de bits de longitud fija que ya se ha insertado realmente en la memoria intermedia de bits global (es decir, no está solo reservada), el escritor de bits 53 elimina los bits de esta secuencia de bits 52 de la memoria intermedia de bits 51 global y escribe los bits 54 en el flujo 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, el escritor de bits 53 espera hasta que esta entrada está llena con los bits reales antes de escribir bits adicionales 54 en el flujo de bits 55.

Al final de un paquete de datos, las memorias intermedias de bin se purgan tal como se ha descrito anteriormente. Además, las memorias intermedias de bits deben purgarse 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 llenan y escriben hacia el flujo de bits.

Se ilustran en la Fig. 12 dos ejemplos de estados posibles de la memoria intermedia de bits 51 global. En el ejemplo (a), se ilustra un caso en el que diferentes memorias intermedias de bits/codificadores de bin reservan un número diferente de bits. Las memorias intermedias de bit globales contienen 3 entradas con secuencias de bits de longitud fija realmente escritas y 4 entradas con secuencias de bits de longitud fija reservadas. La primera entrada de longitud fija ya contiene bits reales (que se deben haber insertado recientemente por la memoria intermedia de bit/codificador de bin 2); esta entrada (es decir los 8 bits correspondientes) pueden eliminarse y escribirse en el flujo de bits. La siguiente entrada reserva 10 bits para el codificador de bin 3, pero los bits reales no se han insertado aun. Esta entrada no puede escribirse en el flujo de bits; debe esperarse hasta que se inserten los bits reales; en el segundo ejemplo (b), todas las memorias intermedias de bits/codificadores de bin reservaron el mismo número de bits (8 bits). La memoria intermedia de bits global contiene 4 reservas para secuencias de bits 8 y 3 secuencias de bits 8 realmente escritas. La primera entrada contiene una reserva para 8 bits para el codificador de bin 3. Antes de que puedan escribirse nuevos bits al flujo de bits, el escritor de bits ha de esperar hasta que la memoria intermedia de bits/codificador de bin 3 escriba los valores reales de los 8 bits en la entrada reservada.

La Fig. 13 muestra una ilustración de una estructura de decodificador PIPE que intercala secuencias de bits de longitud fija. A diferencia de la Fig. 9, los decodificadores de bin 22 no están conectados con una única memoria intermedia de bits. En su lugar, cada decodificador de bin 22 se conecta con una memoria intermedia 58 de bits separada, que almacena los bits desde el flujo de bits parcial correspondiente. Todas las memorias intermedias de bits 58 se conectan a una memoria intermedia 61 de bits global. Los bits 62 desde el flujo de bits 63 se insertan en la memoria intermedia 61 de bits global. En ciertos eventos, en una memoria intermedia 58 de bits particular o codificador de bin 22 conectado o memoria intermedia 20 de bin, la memoria intermedia 58 de bits envía una solicitud 59 a la memoria intermedia 61 de bits global mediante la que se elimina una secuencia de bits 60 de

longitud fija de la memoria intermedia 61 de bits global y se inserta dentro de la memoria intermedia 58 de bits particular. La solicitud para las secuencias de bits 59 de longitud fija se procesa en orden secuencial. La memoria intermedia 61 de bits global representa una memoria intermedia primero en entrar primero en salir; los bits que se insertan antes dentro de la memoria intermedia de bits global se eliminan antes. Debería tomarse nota que diferentes memorias intermedias de bits 58 pueden solicitar una cantidad diferente de bits, lo que también puede variar a lo largo del tiempo basándose en símbolos ya decodificados; pero el número de bits que se solicita mediante una solicitud particular es conocido en el momento en el que la solicitud se envía a la memoria intermedia de bits global. Debería tomarse nota que la memoria intermedia 61 de bits global no se requiere realmente, dado que las palabras de código podrían leerse también directamente desde el flujo de bits. La memoria intermedia 61 de bits global 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 61 de bits global pueden hacerse funcionar como se describe en lo que sigue. La cantidad de bits que se solicitan y leen mediante una memoria intermedia 58 de bits particular se indica como Nx , es igual a la cantidad de bits que se escriben en la memoria intermedia de bits global por la memoria intermedia de bits correspondiente en el lado del codificador. Este número de bits Nx puede ser diferente para diferentes memorias intermedias 58 de bits y puede variar también a lo largo del tiempo. Preferentemente, el número de bits Nx que se solicitan y leen por una memoria intermedia 58 de bits particular puede ser fijo a lo largo del tiempo. La lectura de un número fijo Nx de bits 60 se activa basándose en el número de bits Mx en la memoria intermedia 58 de bits y la longitud máxima de la palabra de código Lx asociada. Obsérvese que cada decodificador de bin 22 puede asociarse con una longitud máxima de palabra de código Lx diferente. Si se envía una solicitud para un bin 19 a una memoria intermedia 20 de bin particular, y la memoria intermedia 20 de bin particular está vacía, y el número Mx de bits de la memoria intermedia 58 de bits que se conecta (a través de un decodificador de bin) con la memoria intermedia 20 de bin particular es menor que la longitud máxima de palabra de código Lx que se asocia con el decodificador de bin 22 correspondiente, la memoria intermedia 58 de bits conectada envía una solicitud 59 para unas nuevas secuencias de Nx bits a la memoria intermedia 61 de bits global. Como respuesta a esta solicitud, se eliminan los primeros Nx bits de la memoria intermedia de bits global 61 y esta secuencia de Nx bits 60 se envía a la memoria intermedia 58 de bits desde la que se envió la solicitud. Finalmente, esta secuencia de Nx bits se añade a la memoria intermedia 58 de bits correspondiente. A continuación se lee la siguiente palabra de código 57 desde esta memoria intermedia de bits, y el decodificador de bin 22 conectado inserta la secuencia de bins 21 asociada dentro de la memoria intermedia 20 de bin conectada. Como respuesta final a la solicitud original de un bin 19, el primer bin se elimina de la memoria intermedia 20 de bin y este bin 25 decodificado se envía al selector 18 de memoria intermedia de bin. Cuando se envía la siguiente solicitud de bin 19 a la memoria intermedia 20 de bin particular y la memoria intermedia de bin no está vacía, se elimina el siguiente bit de la memoria intermedia 20 de bin. Si la memoria intermedia de bin está vacía pero el número Mx de bits de la memoria intermedia 58 de bits conectada es mayor que o igual a la longitud máxima de palabra de código Lx asociada, la siguiente palabra de código se lee desde la memoria intermedia de bits y se inserta una nueva secuencia de bins en la memoria intermedia de bin, a partir de la que se elimina el primer bits y se envía al selector de memoria intermedia de bin. Si la memoria intermedia de bin está vacía y el número Mx de bits en la memoria intermedia 58 de bits conectada es menor que la longitud máxima de la palabra de código Lx conectada, la siguiente secuencia de Nx bits se lee desde la memoria intermedia 61 de bits global y se inserta dentro de la memoria intermedia 58 de bits local conectada, se lee la siguiente palabra de código de la memoria intermedia de bits, se inserta una nueva secuencia de bin en la memoria intermedia de bin, el primer bin de la secuencia se elimina y envía al selector de memoria intermedia de bin. Este proceso se repite hasta que todos los símbolos fuente se decodifican.

Al final de un paquete de datos, podría insertarse más bin y/o bits que los requeridos para la decodificación de los símbolos fuente requeridos dentro de la memoria intermedia de bin y/o memoria intermedia de bits. Los bin restantes en la memoria intermedia de bin y los bits restantes en la memoria intermedia de bits se descartan e ignoran.

50 Intercalado de secuencias de bits de longitud fija con restricción de bajo retardo

El codificador y decodificador PIPE con intercalado de secuencias de bits de longitud fija puede combinarse también con el esquema para el control del retardo de memoria intermedia del codificador, que se ha descrito anteriormente, el concepto de codificación PIPE es el mismo en el caso que implica el control del retardo descrito anteriormente. Si una medida en relación con el retardo o una delimitación superior del retardo (véase a continuación) excede un umbral especificado, la primera entrada de memoria intermedia reservada se llena mediante la purga de la memoria intermedia de bin correspondiente (usando un mecanismo similar como al final de un paquete de datos) y escribiendo potencialmente bits adicionales para el llenado de todos los bits de la entrada de memoria intermedia de longitud fija reservada. Mediante dicho mecanismo, el número de entradas de memoria intermedia en espera se reduce hasta que la medida del retardo asociado es menor que el umbral especificado. En el lado del decodificador, los bin y bits que se han insertado en el lado del codificador en orden para obedecer a la restricción de retardo deben descartarse. Para este descarte de bin y bits puede usarse básicamente el mismo mecanismo que en el lado del codificador.

La medición para el retardo (o una delimitación superior del retardo) puede ser el número de bits en las entradas de memoria intermedia activa en la memoria intermedia de bits global, en donde el número de entradas de memoria intermedia activa es el número de entradas de memoria intermedia de longitud fija reservada más el número de entradas de memoria intermedia de longitud fija que contienen 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, dado que si la primera entrada de memoria intermedia contiene bits escritos, estos bits se escriben al flujo de bits. Sea el retardo máximo de memoria intermedia permitida (tal como se determina por la aplicación) de D bits. Este retardo máximo de memoria intermedia D debe ser conocido tanto para el codificador como para el decodificador. El retardo máximo de memoria intermedia D puede fijarse por la aplicación. El retardo máximo de memoria intermedia D puede señalizarse en el interior del flujo de bits, por ejemplo en la cabecera del paquete de datos (o cabecera del fragmento) o en un conjunto de parámetros, que se incluye en el flujo de bits. Puede señalizarse en unidades de bits, o bytes, o un múltiplo de bits, o un múltiplo de bytes. Si un codificador de bin 10 envía una solicitud para la reserva de una nueva secuencia de bits de longitud fija a la memoria intermedia de bits global, se ejecuta el siguiente proceso antes de que se reserve una nueva entrada en la memoria intermedia de longitud fija.

Si el número de bits en las entradas de memoria intermedia activa en la memoria intermedia de bits global más el número de bits que se reservarán por la solicitud de reserva actual es mayor que el retardo máximo de memoria intermedia D , la primera entrada de memoria intermedia (que se reserva) se purga por el proceso descrito en lo que sigue hasta que el número de bits en las entradas de memoria intermedia activa en la memoria intermedia de bits global más el número de bits que se reservarán por la solicitud de reserva actual sea menor que o igual al retardo máximo de memoria intermedia D . La purga de una entrada de memoria intermedia de longitud fija reservada es similar a la purga al final de un paquete de datos: El codificador de bin 10 que se conecta con la memoria intermedia de bits 48 que ha reservado la primera entrada de memoria intermedia correspondiente es purgada mediante la adición de bins con valores particulares o arbitrarios a la memoria intermedia de bin 8 conectada hasta que la secuencias de bin resultantes representen una secuencia de bin que se asocia con una palabra de código, la palabra de código se inserta entonces en la memoria intermedia de bits 48 correspondiente. Como se ha mencionado anteriormente, una forma preferida para añadir bins a la memoria intermedia de bins es añadir aquellos bins que producen la palabra de código más corta posible. Si, después de la escritura de la palabra de código a la memoria intermedia de bits conectada y la inserción potencial de una secuencia de bits de longitud fija dentro de la memoria intermedia de bits global, hay aún bits en la memoria intermedia de bits (es decir, la escritura de palabras de código no llena completamente la secuencia de bits de longitud fija reservada), se añaden bits adicionales con valores particulares o arbitrarios a la memoria intermedia de bits hasta que se eliminen todos los bits de la memoria intermedia de bits y se escriban en la entrada de memoria intermedia reservada. Finalmente, 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 elimina de la memoria intermedia de bits global y se escribe en el flujo de bits.

En el lado del decodificador, se ejecuta un proceso similar para descartar los bin y bits que se han añadido para obedecer a la restricción de retardo. Por lo tanto, el decodificador mantiene un contador C que cuenta los bits que se han leído desde 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) al comienzo de la decodificación de un paquete de datos y se incrementa después de que se lea una secuencia de longitud fija. Si se lee una secuencia de Nx bits de longitud fija desde la memoria intermedia 61 de bits global, el contador C se incrementa en Nx . Además del contador global C , cada memoria intermedia 58 de bits contiene un contador Cx , que almacena al valor del contador de bits C antes de que se lea la última secuencia de bits de longitud fija dentro de la memoria intermedia 58 de bits correspondiente. Cuando una memoria intermedia 58 de bits particular lee una nueva secuencia de bits de longitud fija, su contador Cx se fija igual a C como una primera etapa y a continuación la secuencia de bits de longitud fija se lee desde la memoria intermedia 61 de bits global. Cuando se envía una solicitud de un bin 19 a una memoria intermedia 20 de bin particular y la diferencia ($C - Cx$) entre el contador global C y el contador Cx de la memoria intermedia 58 de bits conectada es mayor que el retardo máximo de memoria intermedia D , todos los bin que están actualmente almacenados en la memoria intermedia 20 de bin particular y todos los bits que están almacenados en la memoria intermedia 58 de bits conectada se descartan e ignoran. Junto a esa etapa adicional, la decodificación se hace funcionar como se ha descrito anteriormente. Si la memoria intermedia 20 de bin a la que se envía una solicitud de un bin 19 está vacía (o bien porque todos los bin ya se han eliminado o porque un mecanismo de bajo retardo descartó todos los bin en la primera etapa después de que se haya recibido la solicitud de bin), el decodificador de bin 22 conectado intenta leer una nueva palabra de código desde la memoria intermedia 58 de bits conectada. Si el número de bits en la memoria intermedia 58 de bits es mayor que la longitud de palabra de código máxima, se lee una nueva secuencia de bits de longitud fija desde la memoria intermedia 61 de bits global, antes de que se lea la palabra de código, etc.

Las Figs. 7 a 13 se refieren a posibilidades para conseguir una trayectoria de flujo de bits intercalada entre el codificador PIPE 104 por un lado y el decodificador PIPE 202 por otro lado. Como se ha descrito anteriormente con respecto a las Figs. 1 y 2, los aparatos de codificación y decodificación entrópica pueden conectarse entre sí mediante dos canales separados, uno de los cuales transporta el flujo de bits VLC 112 y el otro de los cuales

transporta el flujo de bits codificado según PIPE intercalado. Sin embargo, hay también posibilidades de intercalar incluso ambos flujos de bits VLC 112 así como los flujos de bits codificados PIPE 118, y dichas posibilidades se describirán a continuación con respecto a las Figs. 20 a 24. Sin embargo, antes de eso, se proporciona una base matemática con relación al esquema de codificación PIPE así como detalles con relación a cómo subdividir de modo
 5 óptimo el intervalo de probabilidades con la asignación de los resultados parciales individuales resultantes a los codificadores entrópicos 116 y decodificadores entrópicos 210 individuales, respectivamente.

Como ya se ha hecho notar, en la codificación PIPE el espacio uniforme de la secuencia de entrada de símbolos discretos se mapea sobre un pequeño conjunto de intervalos de probabilidad binaria. Los modelos de probabilidad
 10 para los símbolos fuente pueden ser fijos o adaptativos mientras que la codificación entrópica que usa los intervalos de probabilidad permanece fija y se desacoplan de la etapa de modelizado. Cada uno de los intervalos de probabilidad puede codificarse usando un código de entropía muy simple que tiene el nivel de complejidad de los códigos Huffman. La tasa en exceso del código de entropía de probabilidad del intervalo de partición (PIPE, del inglés "Probability Interval Partitioning Entropy") es similar al de la codificación aritmética pura.

La codificación entrópica, en general, puede considerarse como la forma más genérica de compresión de datos sin pérdida. La compresión de datos sin pérdida se dirige a representar datos discretos con menos bits que los necesarios para la representación de datos original pero sin ninguna pérdida de información. Los datos discretos pueden darse en la forma de texto, gráficos, imágenes, video, audio, voz, facsímil, datos médicos, datos
 20 meteorológicos, datos financieros o cualquier otra forma de datos digitales. En muchas aplicaciones de codificación, los datos fuente originales se mapean primero en los denominados símbolos de codificación y estos símbolos de codificación se codifican a continuación de modo entrópico. El mapeado en símbolos de codificación puede incluir la cuantificación, en cuyo caso el esquema de codificación global es con pérdidas. Un símbolo de codificación s puede tomar cualquier valor desde un alfabeto M -ario ($M \geq 2$) $A = \{a_0, \dots, a_{M-1}\}$. Para la finalidad de codificación del símbolo s , el alfabeto se asocia con una función de masa de probabilidad (pmf) estimada $\{p_s(a_0), \dots, p_s(a_{M-1})\}$ y todas las dependencias entre símbolos de codificación que no se consideran en esta pmf se desprecian. Para estos ajustes abstractos, la entropía

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

es la delimitación inferior más grande para la longitud de palabra esperada en bits por símbolo, para la codificación de los símbolos s , que puede conseguirse con técnicas de codificación entrópica. Durante décadas, la codificación Huffman, y la codificación aritmética han dominado la codificación entrópica práctica. Hay ejemplos bien conocidos de códigos prácticos capaces de aproximarse al límite entrópico (en un cierto sentido).
 35

Para una distribución de probabilidad fija, los códigos de Huffman son relativamente fáciles de construir. La propiedad más atractiva de los códigos de Huffman es que su implementación puede realizarse de modo eficiente mediante el uso de tablas de códigos de longitud variable (VLC). Sin embargo, cuando se trata con estadísticas fuente variables en el tiempo, es decir, probabilidades de símbolos cambiantes, la adaptación del código Huffman y sus tablas VLC correspondientes es bastante exigente, tanto en términos de complejidad algorítmica así como en términos de costes de implementación. También, 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 el uso de ninguna extensión de alfabeto tal como la ejecución de codificación de tirada) puede ser bastante sustancial. Otros inconvenientes de los códigos Huffman viene dado por el hecho de que en caso de tratar con modelizados de probabilidad de orden más alto, pueden requerirse múltiples conjuntos de tablas VLC.
 45

La codificación aritmética, por otro lado, aunque es sustancialmente más compleja que la VLC, ofrece la ventaja de un manejo más consistente y adecuado cuando se maneja un modelizado de probabilidad adaptativo y de orden más alto así como con el caso de distribuciones de probabilidad altamente sesgadas. Realmente, esta característica es el resultado básicamente del hecho de que la codificación aritmética proporciona un mecanismo, al menos conceptualmente, para mapear cualquier valor dado de estimación de probabilidad en una forma más o menos directa a una parte de la palabra de código resultante. Estando provista con dicha interfaz, la codificación aritmética permite una clara separación entre las tareas de modelizado de probabilidad y estimación de probabilidad, por un lado, y la codificación entrópica real, es decir el mapeado de símbolos a palabras de código, por otro lado.
 55

A diferencia de los esquemas de codificación entrópica convencional recién explicados, la codificación PIPE usa particionado de los intervalos de probabilidad, cuya base matemática se describe con más detalle a continuación.

Considérese la secuencia de símbolos de codificación $\{s_0, \dots, s_{N-1}\}$. Cada símbolo se extrae de un alfabeto $s_i \in A_i$. Los alfabetos $A_i = \{a_0^i, a_1^i, \dots\}$ contienen dos o más letras estando cada una asociada con una estimación de probabilidad $p_s(a_m^i)$. Las estimaciones de probabilidad $p_s(a_m^i)$ son conocidas para codificador y decodificador y pueden ser fijas
 60

o variables. Se supone que las probabilidades variables se estiman simultáneamente en el codificador y decodificador. Los alfabetos A_i pueden ser tanto idénticos para la secuencia de símbolos como asociarse tipos de símbolos diferentes con diferentes alfabetos. En este último caso, se supone que el decodificador conoce el alfabeto de cada símbolo en la secuencia. Esta suposición se justifica dado que las descripciones de códec fuente prácticas
5 contienen una sintaxis que estipula el orden de los símbolos y sus alfabetos.

La secuencia de símbolos $\{s_0, \dots, s_{N-1}\}$ se convierte en una secuencia de símbolos binarios, que también se denominan como bins. Para cada símbolo s_i la binarización

$$10 \quad b_i = \{b_0^i \dots\} = \gamma_b^i(s_i) \quad (B2)$$

representa un mapeado biyectivo de letras del alfabeto a_m^i en conjuntos de bins ordenados b_m^i . El mapeado de binarización γ_b^i puede ser diferente para diferentes símbolos s_i o categorías de símbolos. Cada secuencia de bins i para un símbolo particular s_i consiste en uno o más bins b_k^i . En el lado del decodificador, puede reconstruirse un
15 símbolo s_i mediante el mapeado inverso $s_i = (\gamma_b^i)^{-1}(b^i)$ dada la secuencia de bins b^i . Como resultado de la binarización, se obtiene una secuencia de bins $\{b_0, \dots, b_{S-1}\}$ que representa la secuencia de símbolos fuente $\{s_0, \dots, s_{N-1}\}$.

Todos los bins b_j se asocian con el mismo alfabeto binario $B = \{0,1\}$, pero las pmfs binarias correspondientes $\{p_0^j, p_1^j\}$ siendo $p_1^j = p_0^j$, son normalmente diferentes. Una pmf binaria $\{p_0^j, p_1^j\}$ puede describirse por el valor del bin menos probable (LPB) b_{LPB}^j y su probabilidad p_{LPB}^j (siendo $p_{LPB}^j \leq 0,5$). Esta descripción de probabilidad binaria $\{b_{LPB}^j, p_{LPB}^j\}$ puede deducirse directamente de estimaciones de probabilidad $p_s(a_m^i)$ para los alfabetos de símbolos dados los mapeados de binarización γ_b^i . Es posible también (y frecuentemente preferible) estimar directamente
20 $\{b_{LPB}^j, p_{LPB}^j\}$ simultáneamente en el lado del codificador y del decodificador. Por lo tanto, los bins pueden asociarse con un modelo de probabilidad (al que también se hace referencia como un contexto) basándose en la sintaxis y símbolos o bins realmente codificados. Y para cada modelo de probabilidad, la descripción de probabilidad $\{b_{LPB}^j, p_{LPB}^j\}$ puede estimarse basándose en los valores de los bins que se codifican con el modelo de probabilidad. Un ejemplo de dicho modelizado de probabilidad binario se describe con respecto a CABAC de H.264.

30 Dado que la función entrópica binaria

$$H(p) = -p \log_2(p) - (1 - p) \log_2(1 - p) \quad (B3)$$

es simétrica alrededor de $p=0,5$, el mismo codificador binario puede usarse para la codificación de todos los bin que
35 se asocian con la misma probabilidad LPB p_{LPB}^j , independientemente del valor de b_{LPB}^j . Por lo tanto la secuencia de bins $\{b_0, \dots, b_{B-1}\}$ se convierte en una secuencia de bins de codificación $\{b_0^c, \dots, b_{B-1}^c\}$. Para cada bin b_j , se especifica el mapeado biyectivo correspondiente γ_c^j mediante

$$40 \quad b_j^c = \gamma_c^j(b_j) = b_j \oplus b_{LPB}^j \quad (B4)$$

en la que \oplus indica el operador O exclusiva. En el lado del decodificador, los bins b_j pueden reconstruirse dados los bins de codificación b_j^c y el valor LPB correspondiente b_{LPB}^j mediante el mapeado inverso $b_j = (\gamma_c^j)^{-1}(b_j^c) = b_j^c \oplus b_{LPB}^j$. Un bin codificado $b_j^c = 0$ especifica que el valor de bin correspondiente b_j es igual al valor LPB b_{LPB}^j y un bin codificado $b_j^c = 1$ especifica que el valor del bin correspondiente b_j es igual al valor del bin más probable (MPB)

$$45 \quad 1 - b_{LPB}^j.$$

La secuencia de bins codificados $\{b_0^c, \dots, b_{B-1}^c\}$ únicamente representa la secuencia de símbolos fuente $\{s_0, \dots, s_{N-1}\}$ y las estimaciones de probabilidad correspondientes, que pueden emplearse para codificación entrópica, se describen completamente por las probabilidades LPB p_{LPB}^j (siendo $p_{LPB}^j \leq 0,5$). Por ello, solo necesitan considerarse
50 las probabilidades en el intervalo semiabierto $(0, 0,5]$ para el diseño del codificador entrópico binario para la codificación de bins b_j^c .

Para la codificación entrópica binaria real, la secuencia de bins de codificación $\{b_0^c, \dots, b_{B-1}^c\}$ se proyecta sobre un número pequeño de intervalos de probabilidad I_k . El intervalo de probabilidad LPB $(0, 0.5]$ se particiona en K intervalos $I_k = (p_k, p_{k+1}]$

5

$$\bigcup_{k=0}^{K-1} I_k = (0, 0.5] \quad \text{y} \quad I_k \cap I_j = \emptyset \quad \text{para} \quad k \neq j \quad (B5)$$

El conjunto de K se caracteriza por $K-1$ bordes de intervalo p_k siendo $k = 1, \dots, K-1$. Sin pérdida de generalidad suponemos que $p_k < p_{k+1}$ para $k = 0, \dots, K$. Los bordes de intervalo exteriores se fijan y dan por $p_0 = 0$ y $p_K = 0.5$. Un codificador entrópico binario no adaptativo simple se designa para cada intervalo I_k . Todos los bins de codificación b_j^c con probabilidades LPB asociadas $p_{LPB}^j \in I_k$ se asignan al intervalo I_k y se codifican con el codificador entrópico fijo correspondiente.

10

En la descripción que sigue, todos los bins b_j^c representan bins de codificación y todas las probabilidades p son probabilidades LPB p_{LPB}^j .

15

Para investigar el impacto del esquematizado en el intervalo de probabilidad sobre la eficiencia de codificación, suponemos que podemos diseñar un codificador entrópico óptimo para una probabilidad fija que alcanza la delimitación entrópica. 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 entrópico óptimo correspondiente debería conseguir el límite de entropía para su probabilidad representativa. Bajo esta suposición, la tasa de codificación de un bin con probabilidad p que use el codificador entrópico óptimo para el intervalo representativo p_{I_k} viene dado por

20

$$R(p, p_{I_k}) = -p \log_2(p_{I_k}) - (1-p) \log_2(1-p_{I_k}) = H(p_{I_k}) + (p-p_{I_k}) H'(p_{I_k}) \quad (B6)$$

25

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

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

30

En su primera derivada. Suponemos adicionalmente que la distribución de probabilidades en el intervalo $(0, 0.5]$ viene dada por $f(p)$, siendo $\int_0^{0.5} f(p) dp = 1$. A continuación, la tasa esperada, en bits por bin, para un conjunto dado de K intervalos $\{I_k\}$ con probabilidades representativas correspondientes $\{p_{I_k}\}$ puede describirse como

35

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

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

40

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

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

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

45

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

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

es siempre mayor que cero si

$$5 \quad \int_{p_k}^{p_{k+1}} p f(p) dp > 0 \quad (B12)$$

Por ello, si se cumple la condición B12, el valor p_k^* dado por la ecuación B10 es la probabilidad representativa para un intervalo I_k que minimiza la tasa global R esperada dadas las fronteras del intervalo p_k y p_{k+1} . En caso contrario, no se proyecta ningún bin en el intervalo I_k y la probabilidad representativa $p_k \in I_k$ puede elegirse arbitrariamente sin ningún impacto en la tasa global R; pero dicha configuración debería evitarse, dado que el intervalo I_k no debería emplearse para codificación entrópica.

Para hallar una condición para los bordes de intervalo óptimos, investigamos las primeras derivadas de la tasa global R esperada con respecto a los bordes de intervalo p_k con $k = 1, \dots, K-1$. Si $f(p) > 0$ para todo $p \in [p_{k-1}, p_k]$, la ecuación $\frac{\partial}{\partial p_k} R = 0$ tiene una única solución.

$$p_k^* = \frac{H(p_k) - p_k H'(p_k) - H(p_{k-1}) + p_{k-1} H'(p_{k-1})}{H'(p_{k-1}) - H'(p_k)} \quad (B13)$$

para el borde de intervalo p_k dentro del dominio de definición $[p_{k-1}, p_k]$ y la segunda derivada parcial para esta solución

$$\frac{\partial^2}{\partial p_k^2} R(p_k^*) = f(p_k^*) (H'(p_{k-1}) - H'(p_k)) \quad (B14)$$

es siempre mayor que cero, de modo que p_k^* es el borde del intervalo $p_k \in [p_{k-1}, p_k]$ que minimiza la tasa global R esperada dados los intervalos representativos p_{k-1} y p_k . Si existen probabilidades $p \in [p_{k-1}, p_k]$ con $f(p)=0$, la ecuación $\frac{\partial}{\partial p_k} R = 0$ tiene múltiples soluciones, pero p_k^* tal como se da por la ecuación B13 es aún óptimo incluso aunque puedan existir soluciones óptimas adicionales.

Dado el número de intervalos K y la probabilidad de distribución $f(p)$, los bordes de intervalo p_k , con $k=1, \dots, K-1$, y las representaciones del intervalo p_k , con $k = 0, \dots, K-1$, que minimizan la tasa global R esperada pueden obtenerse mediante la resolución del sistema de ecuaciones dado por las ecuaciones B10 y B13 sometido a las condiciones B12 para $k = 0, \dots, K-1$. Esto puede conseguirse con el siguiente algoritmo iterativo.

Algoritmo 1:

- 1) Particionar el intervalo (0, 0,5] en 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 $k = 0, \dots, K-1$ de tal forma que las condiciones B12 se obedezcan para todo $k = 0, \dots, K-1$.
- 2) Actualizar los representantes p_k con $k = 0, \dots, K-1$ de acuerdo con la ecuación B10
- 3) Actualizar los bordes de intervalo p_k con $k = 1, \dots, K-1$ de acuerdo con la ecuación B13
- 4) Repetir las dos etapas previas hasta convergencia

La Figura 14 muestra un ejemplo para el discretizado del intervalo óptimo usando el algoritmo descrito. Para este ejemplo, suponemos una distribución de probabilidad uniforme $f(p) = 2$ para $0 < p \leq 0,5$ y se particiona el intervalo de probabilidad (0, 0,5] en $K = 4$ intervalos. Puede verse que el discretizado del intervalo de probabilidad conduce a una aproximación lineal en forma de piezas $A(p)$ de la función de entropía binaria $H(p)$ siendo $A(p) \geq H(p)$ para todo $p \in (0, 0,5]$.

Como medida para el impacto del discretizado del intervalo sobre la eficiencia de codificación puede usarse la tasa global esperada de incremento con relación al límite de entropía

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

Para el ejemplo de la Figura 14, el valor esperado de la entropía es igual a $1/(2\ln 2)$ bit por bin y la sobrecarga de la tasa ρ es igual al 1,01 %. La Tabla 4 lista las sobrecargas de tasa $\bar{\rho}_{uni}$ y $\bar{\rho}_{lin}$ para la distribución de probabilidad uniforme y una distribución de probabilidad de incremento lineal $f(p)=8\rho$ con $p \in (0, 0,5]$, respectivamente, para números seleccionados de intervalos K.

Tabla 4: Sobrecarga de tasa respecto al número de intervalos de probabilidad para la distribución de probabilidad uniforme y de incremento 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 el discretizado del intervalo de probabilidad LPB (0, 0,5] en un número pequeño de intervalos con una probabilidad fija (por ejemplo, 8 a 10 intervalos) tiene muy poco impacto sobre la eficiencia de codificación.

La codificación de entropía anteriormente explicada para intervalos de probabilidad, permite así codificadores individuales que usen probabilidades fijas.

En lo que sigue, primero se muestra cómo puede diseñarse un código simple para probabilidades fijas. Dados estos resultados, se desarrollan unos algoritmos que optimizan conjuntamente el diseño del código y la partición del intervalo de probabilidad LPB (0, 0,5].

Una codificación entrópica para probabilidades fijas $p = p_k$ puede realizarse 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 entrópica binaria para el que se mapea un número variable de bins sobre unas palabras de código de longitud variable. Para una capacidad de decodificación única, el mapeado inverso de una palabra de código a una secuencia de bins debe ser único. Y dado que se desea diseñar un código que enfoque el límite de entropía tan próximo como sea posible, restringimos las consideraciones a mapeados biyectivos. Dicho mapeado biyectivo puede representarse por un árbol binario en donde todos los nodos de hoja se asocian con palabras de código, tal como se representa en la Figura 15. Los tres bordes representan eventos binarios. En el ejemplo de la Figura 15, los límites inferiores representan el valor de bin LPB y los límites superiores representan el valor de bin MPB. El árbol binario representa un código de prefijo para los bins si es un árbol binario completo, es decir, si cada nodo es o bien una hoja o tiene dos descendientes. Cada nodo de hoja se asocia con una probabilidad basándose en una probabilidad LPB p dada. El nodo raíz tienen la probabilidad $p_{raiz} = 1$. La probabilidad para todos los otros nodos se obtiene multiplicando la probabilidad del ascendiente correspondiente por p para los descendientes LPB y $q = 1 - p$ para los descendientes MPB. Cada nodo de hoja L_1 se caracteriza por el número de límites LPB a_1 y el número de límites MPB b_1 desde el nodo raíz al nodo hoja. Para una probabilidad LPB p particular, la probabilidad p_1 para el nodo hoja $L_1 = \{a_1, b_1\}$ es igual a

$$p_1 = p^{a_1} (1 - p)^{b_1} \tag{B16}$$

El árbol binario T se caracteriza totalmente por el número de nodos hoja L y los pares $\{a_l, b_l\}$ asociados siendo $l = 0, \dots, L-1$.

Dado el árbol binario T completo y una probabilidad LPB p , la asignación óptima de palabras de código a nodos hoja puede obtenerse mediante el algoritmo de Huffman. El número variable resultante de mapeado C de bits a palabras de código de longitud variable (V2V) se caracteriza por el número de palabras de código L , que es idéntico al número de nodos hoja, y de tuplas $\{a_l, b_l, l_1\}$ para $l = 0, \dots, L-1$, en donde l_1 representa la longitud de la palabra de código que se asocia con el nodo hoja $L_1 = \{a_1, b_1\}$ correspondiente. Debería tomarse nota que hay múltiples posibilidades para la asignación de palabras de código dadas por las longitudes de palabra de código $\{l_1\}$ y las asignación de palabras de código real no es importante siempre que las palabras de código representen un código de prefijo decodificable únicamente. La tasa esperada $R(p, C)$ en bits por bin para un código C dado y una probabilidad LPB p es la relación de longitud de palabra de código esperada y el número esperado de bins por palabra de código

$$R(p, C) = \frac{\sum_{i=0}^{L-1} p_i l_i}{\sum_{i=0}^{L-1} p_i (a_i + b_i)} = \frac{\sum_{i=0}^{L-1} p^{a_i} (1-p)^{b_i} l_i}{\sum_{i=0}^{L-1} p^{a_i} (1-p)^{b_i} l_i (a_i + b_i)} \quad (B17)$$

El diseño del código está frecuentemente limitado por factores como el número máximo de palabras de código L , el número máximo de bins por palabra de código, o la longitud máxima de la palabra de código, o está restringido a 5 códigos de estructuras particulares (por ejemplo, para permitir un análisis optimizado). Si suponemos que se da el conjunto S_c de códigos utilizable para una aplicación particular, el código óptimo $C^* \in S_c$ para una probabilidad LPB p particular puede hallarse minimizando la tasa esperada $R(p, C)$

$$C^*(p) = \arg \min_{\forall C \in S_c} R(p, C) \quad (B18)$$

10 Como alternativa más rápida, el minimizado también puede proceder sobre un conjunto dado de árboles binarios S_T y para cada árbol solo se considera un código V2V C que se obtiene mediante el algoritmo Huffman. Como un ejemplo, se diseñan códigos V2V para varias probabilidades LPB p mediante la consideración de todos los árboles binarios T para los que el número de nodos hoja L es menor que o igual a un máximo dado L_m . En la Figura 16, el 15 incremento de tasa relativo $\rho(p, C^*(p)) = R(p, C^*(p))/N(p)$ se trata sobre la probabilidad LPB p para tamaños de tabla máximo L_m seleccionados. El incremento de tasa $\rho(p)$ puede reducirse normalmente permitiendo tamaños de tabla mayores. Para probabilidades LPB mayores, es normalmente suficiente un tamaño de tabla L pequeño de 8 a 16 palabras de código para mantener el incremento de tasa $\rho(p)$ razonablemente pequeño, pero para probabilidades LPB más pequeñas (por ejemplo, $p < 0,1$), se requieren tamaños de tabla L mayores.

20 En las secciones previas, se consideró el discretizado de probabilidad óptimo suponiendo códigos óptimos y el diseño de código para probabilidades LPB fijas. Pero dado que, en general, no se puede conseguir el límite de entropía con códigos V2V reales de tamaños de tabla limitados, debe considerarse el diseño de código y la partición del intervalo de probabilidad LPB $(0, 0,5]$ conjuntamente para la obtención de un diseño de código entrópico optimizado. 25

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

$$30 \quad C_k^* = \arg \min_{\forall C \in S_c} \int_{p_k}^{p_{k+1}} R(p, C_k) f(p) dp \quad (B19)$$

Para diseños prácticos, el minimizado de la integral en la ecuación B19 puede simplificarse, con un impacto mínimo en la eficiencia de codificación, determinando primero una probabilidad representativa óptima p_k^* para el intervalo I_k de acuerdo con la ecuación B10 y eligiendo a continuación el código C_k^* óptimo del conjunto dado S_c para la 35 probabilidad representativa p_k^* de acuerdo con la ecuación B18.

Bordes de intervalo óptimos p_k , con $k = 1, \dots, K-1$, dado el conjunto de códigos C_k , con $k = 0, \dots, K-1$, pueden deducirse minimizando la tasa global esperada

$$40 \quad R = R(\{p_k\}, \{C_k\}) = \sum_{k=0}^{K-1} \left(\int_{p_k}^{p_{k+1}} R(p, C_k) f(p) dp \right) \quad (B20)$$

Establecer las primeras derivadas con respecto a los bordes de intervalos iguales a cero, $\frac{\partial}{\partial p_k} R = 0$, para $k = 1, \dots, K-1$, conduce a

$$45 \quad p_k^* = p_k \quad \text{con} \quad R(p_k, C_{k-1}) = R(p_k, C_k) \quad (B21)$$

De modo similar que para la conexión B13, se puede mostrar que p_k^* es siempre una solución óptima, pero dependiendo de la distribución de probabilidad $f(p)$ pudieran existir soluciones óptimas adicionales. Por ello, un borde de intervalo óptimo p_k^* entre dos intervalos I_{k-1} e I_k con códigos asociados C_{k-1} y C_k dados, respectivamente,

es el punto de intersección de las funciones $R(p, C_{k-1})$ y $R(p, C_k)$.

En consecuencia, puede usarse el siguiente algoritmo interactivo para deducir conjuntamente la partición del intervalo de probabilidad y los códigos asociados dado el número K de intervalos de probabilidad, el conjunto de códigos S_c posibles y la distribución de probabilidad $f(p)$, con $p \in (0, 0,5]$.

Algoritmo 2:

- 1) Deducir las fronteras iniciales de intervalo de probabilidad p_k , con $k = 0, \dots, K$, usando el algoritmo 1 especificado en la sección 3
- 2) Deducir las representaciones p_{I_k} para los intervalos de probabilidad I_k , con $k = 0, \dots, K-1$, de acuerdo con la ecuación B10
- 3) Deducir los códigos $C_k \in S_c$ para las representaciones de intervalo p_{I_k} , con $k = 0, \dots, K-1$, de acuerdo con la ecuación B18
- 4) Actualizar los bordes de intervalo p_k , con $k = 1, \dots, K-1$, de acuerdo con la ecuación B21
- 5) Repetir las tres etapas previas hasta convergencia

Las etapas 2 y 3 en el algoritmo 2 podrían sustituirse también por una deducción directa de los códigos $C_k \in S_c$, con $k = 0, \dots, K-1$, basándose en los bordes del intervalo p_k , con $k = 0, \dots, K$, de acuerdo con la ecuación B19. Y, como se menciona en la sección 4.1, el minimizado en la etapa 3 puede proseguir también sobre un conjunto dado de árboles binarios S_T en donde para cada árbol binario T solo se considera un código V2V C_k obtenido mediante el algoritmo de Huffman.

Como un ejemplo, se deduce conjuntamente la partición en $K = 12$ intervalos de probabilidad y códigos V2V correspondientes usando el algoritmo 2. Para ello, el minimizado en la etapa 3 del algoritmo se sustituyó con un minimizado equivalente sobre un conjunto dado de árboles binarios S_T en donde el código evaluado C para cada árbol T se obtuvo mediante el algoritmo de Hoffman. W_c árboles T considerados con un número máximo de $L_m = 65$ nodos de hoja y por ello C códigos con hasta 65 entradas de tabla. Todos los árboles binarios T con hasta 16 nodos de hoja se han evaluado en el minimizado; para árboles con más de 16 nodos de hoja, se empleó una búsqueda subóptima dados los mejores resultados para los árboles con un número más pequeño de nodos de hoja.

En la Figura 17, el incremento esperado de tasa con relación al límite de entropía $\Delta R(p) = R(p) - H(p)$ para el diseño de código de ejemplo se traza sobre la probabilidad LPB p . Como comparación, también se trazó dentro del diagrama el incremento de tasa ΔR esperado para el discretizado del intervalo de probabilidad teóricamente óptimo (tal como se desarrolla en la sección 3) y el discretizado de probabilidad teóricamente óptimo con la restricción adicional $p_{I_{k-1}} = 0,5$. Puede verse que el discretizado del intervalo de probabilidad conjunto y el diseño del código V2V conduce a un desplazamiento de los bordes del intervalo (los bordes del intervalo p_k , con $k = 1, \dots, K-1$, vienen dados por el máximo local de las curvas $\Delta R(p)$). El incremento de tasa global esperado relativo con relación al límite de entropía para el ejemplo del diseño con códigos V2V reales es $\bar{\rho} = 0,24 \%$, cuando se asume una distribución de probabilidad uniforme $f(p)$. Los incrementos de tasa relativos correspondientes para el discretizado de intervalo de probabilidad teóricamente óptimo y el discretizado de probabilidad teóricamente óptimo con la restricción adicional $p_{I_k} = 0,5$ son $\bar{\rho} = 0,12 \%$ y $\bar{\rho} = 0,13 \%$, respectivamente.

La finalización de la palabra de código puede realizarse como sigue. Cuando se codifica una secuencia finita de símbolos $\{s_0, \dots, s_{N-1}\}$, cada uno de los K codificadores binarios procesa una secuencia finita de bins de codificación $b_k^c = \{b_0^c, \dots, b_{B-1}^c\}_k$, con $k = 0, \dots, K-1$. Y se ha asegurado que, para cada uno de los K codificadores binarios, todos los bin de codificación de la secuencia $b_k^c = \{b_0^c, \dots, b_{B-1}^c\}_k$ puedan reconstruirse dada la palabra de código o secuencia de palabras de código $c_k(b_k^c)$.

Cuando se emplea codificación aritmética, la palabra de código aritmética para la secuencia de bins de codificación ha de finalizarse en una forma, en la que todos los bin de codificación puedan decodificarse dada la palabra de código. Para los códigos V2V descritos anteriormente, los bins al final de la secuencia b_k^c pueden no representar una secuencia de bins que esté asociada con una palabra de código. En dicho caso, puede escribirse cualquier palabra de código que contenga el resto de la secuencia de bins como prefijo. La sobrecarga puede minimizarse, si se elige la palabra de código correspondiente (o una de esas palabras de código) que tenga la longitud mínima. En el lado del decodificador, los bins leídos adicionalmente al final de la secuencia de bins, que pueden identificarse dada la sintaxis del flujo de bits y esquemas de binarización, se descartan.

Se presenta a continuación un diseño de código simple de ejemplo. Con finalidades de ilustración, se considera el ejemplo simple de una fuente $\{s\}$ con tres letras y probabilidades asociadas fijas de $ps(a_0) = 0,7$, $ps(a_1) = 0,18$, y $ps(a_2) = 0,12$. El árbol de elección ternaria correspondiente puede convertirse en un árbol binario completo como se

muestra en la Fig. 18.

Una binarización para el árbol binario completo en la Fig. 18 se da en la Tabla 5. El símbolo ternario pmf p_s se convierte en dos pmfs binarios $p_{b_0} = (0,7, 0,3)$ y $p_{b_1} = (0,6, 0,4)$. Para cada símbolo s en el flujo de bits, está presente el bin b_0 . Cuando b_0 es igual a 0, también está presente b_1 . Obsérvese que la binarización dada en la Tabla 2 es idéntica a un código de Huffman de letra simple óptimo para la fuente s .

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

Símbolo a_i	Probabilidad $p(a_i)$	Bin b_0	Bin b_1
a_0	0,7	1	
a_1	0,18	0	1
a_2	0,12	0	0
LPB Prob.	$P_{LPB} = p(b_j = 0)$	0,3	0,4

La entropía para la fuente s es

$$H(0,7, 0,18, 0,12) = H(0,7, 0,3) + 0,3 H(0,6, 0,4) = 1,1726 \text{ bit/símbolo} \quad (B22)$$

La longitud media de la palabra de código del código Huffman de letra única viene dado como

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

Correspondiente a una redundancia de $\rho_{HC} = 0,1274$ bit/símbolo o 10,87 % de sobrecarga de tasa esperado.

Para el ejemplo de binarización particular con pmfs fijas, los bins b_0 y b_1 ya representan codificación de bins, dado que para ambos bins el valor LPB p_{LPB}^j es igual a 0. La distribución $f(s)$ de las probabilidades LPB es discreta, con $f(p) = 0$ excepto para $p = 0,3$ y $p = 0,4$. En consecuencia, el discretizado de probabilidad óptimo conduce a $K = 2$ intervalos con representantes $p_{i_0} = 0,3$ y $p_{i_1} = 0,4$. El borde del intervalo p_1 ente estos intervalos puede elegirse arbitrariamente en $[0,3, 0,4)$.

Para la codificación de la fuente, la secuencia de símbolos fuente se binariza en una secuencia de bins. El bin b_0 se transmite para cada símbolo fuente. El bin b_1 solo se transmite cuando $b_0 = 0$. Los bins b_0 y b_1 se codifican por separado con probabilidades LPB constantes de $p_{i_0} = 0,3$ y $p_{i_1} = 0,4$, respectivamente.

Una codificación eficiente de un alfabeto binario con probabilidad fija puede conseguirse simplemente mapeando un V2V. Ejemplos de mapeados para V2V con pequeñas tablas de codificación para las probabilidades LPB $p_{LPB} = 0,3$ y $p_{LPB} = 0,4$ se dan en la Tabla 6 y la Tabla 7, respectivamente. El mapeado V2V para $p_{LPB} = 0,3$ conduce a una redundancia de 0,0069 bit/bin o 0,788 %. Para la probabilidad LPB de $p_{LPB} = 0,4$, la redundancia es 0,0053 bit/bin o 0,548 %.

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

Árbol de bin	Probabilidad	Códigos
'11'	$0,7^2 = 0,49$	'1'
'01'	$0,7 \cdot 0,3 = 0,21$	'01'
'0'	0,3	'00'

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

Árbol de bin	Probabilidad	Árbol de código
'111'	$0,6^3 = 0,216$	'11'
'110'		'001'
	$0,6^2 \cdot 0,4 = 0,144$	
'10'	$0,6 \cdot 0,4 = 0,24$	'11'
'01'	$0,4 \cdot 0,6 = 0,24$	'01'
'00'	$0,4^2 = 0,16$	'000'

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

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

5 La redundancia global es del 0,73 % con relación al límite de entropía, lo que representa una mejora significativa en comparación con el código de Huffman de letra única.

10 Se podría argumentar que podría obtenerse una mejora de eficiencia de codificación similar mediante la creación de un código de longitud de tirada. Para el ejemplo anterior, podemos construir un código de longitud de tirada para el símbolo más probable considerando las tiradas 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 un la palabra de código separada. Dicho código conduce a la redundancia de 1,34 % con relación al límite de entropía. Realmente, los códigos V2V pueden considerarse como una generalización de los códigos de longitud de tirada para símbolos binarios (el código V2V en la Tabla 3 representa efectivamente un código de longitud de tirada). Para un alfabeto de símbolos único con probabilidades fijas, puede conseguirse una eficiencia de codificación similar que para el enfoque presentado creando un código que mapee un número variable de símbolos fuente a palabras de código de longitud variable. La ventaja principal del enfoque presentado es su flexibilidad en el mapeado de secuencias de símbolos fuente arbitrarias con estimaciones de probabilidad fija o adaptativa a un número pequeño de codificadores binarios simples que funcionan con probabilidades LPB fijas.

20 Se considera a continuación cómo conseguir una capacidad de decodificación única.

Con el esquema de codificación entrópico presentado, la codificación de una secuencia de símbolos fuente $s = \{s_0, \dots, s_{N-1}\}$ consiste en las siguientes tres etapas básicas.

- binarización del símbolo $b = \{b_0, \dots, b_{S-1}\} = \gamma_b(s)$ que conducen la secuencia de bins $b = \{b_0, \dots, b_{S-1}\}$
- 25 • conversión de la secuencia de bins en una secuencia de bins de codificación $b^c = \{b_0^c, \dots, b_{B-1}^c\} = \gamma_c(b)$
- codificación entrópica binaria de la secuencia de bins de codificación $b^c = \{b_0^c, \dots, b_{B-1}^c\}$ usando discretizado del intervalo de probabilidad y K codificadores binarios fijos

30 La secuencia de símbolos $s = \{s_0, \dots, s_{N-1}\}$ puede decodificarse únicamente, si la secuencia de bins de codificación $b^c = \{b_0^c, \dots, b_{B-1}^c\}$ puede decodificarse únicamente y los mapeados γ_b y γ_c pueden invertirse.

Dejemos que γ_e notifique el mapeado del codificador de una secuencia de uno o más bins de codificación $b^c = \{b_0^c, \dots\}$ sobre una secuencia de una o más palabras de código $c(b^c) = \{c_0, \dots\}$

$$35 \quad c(b^c) = \gamma_e(b^c) \quad (\text{B25})$$

Para una capacidad de decodificación única de una secuencia de bins de codificación b^c dada la secuencia de palabras de código $c(b^c)$, el mapeado del codificador γ_e debe tener la propiedad de que se asigna una única palabra de código $c(b^c)$ a cada posible secuencia de bins de codificación b^c :

$$40 \quad \forall b_i^c, b_j^c \quad b_i^c \neq b_j^c \rightarrow c(b_i^c) \neq c(b_j^c) \quad (\text{B26})$$

45 Esta propiedad se satisface siempre que se usan códigos aritméticos o códigos de prefijo. Se satisface particularmente para los códigos V2V descritos en la sección 4.1 (incluyendo la palabra de código, finalización descrita en la sección 4.3), dado que los códigos V2V representan códigos de prefijo para números variables de bins.

Sin embargo, en el enfoque de codificación entrópica presentado, la secuencia de bins de codificación b^c se particiona en K subsecuencias b_k^c , con $k = 0, \dots, K-1$,

$$50 \quad \{b_0^c, \dots, b_{K-1}^c\} = \gamma_\rho(b^c) \quad (\text{B27})$$

y se asigna a cada una de las subsecuencias b_k^c , una secuencia de palabras de código $c_k(b_k^c)$ que usa un mapeado de codificador γ_c^k particular. En consecuencia, la condición de capacidad de decodificación única ha de extenderse.

55 Una secuencia de bins de codificación b^c es decodificable únicamente dadas K secuencias de palabras de código $c_k(b_k^c)$, con $k = 0, \dots, K-1$, si cada subsecuencia de bins de codificación b_k^c es únicamente decodificable dada la palabra de código $c_k(b_k^c)$ correspondiente y la regla de partición γ_e es conocida para el decodificador. La regla de

partición γ_e viene dada por el discretizado del intervalo de probabilidad LPB $\{I_k\}$ y las probabilidades LPB p_{LPB}^j que se asocian con los bins de codificación b_j^c , con $j = 0, \dots, B-1$. Por ello, el discretizado del intervalo de probabilidad LPB $\{I_k\}$ ha de ser conocido en el lado del decodificador y la probabilidad LPB p_{LPB}^j para cada bin de codificación b_j^c , con $j = 0, \dots, B-1$, ha de deducirse de la misma manera en el lado del codificador y del decodificador.

5 Para el mapeado γ_c de una secuencia de bins en una secuencia de bins de codificación b^c , cada único b_j , con $j = 0, \dots, B-1$, se convierte mediante el mapeado binario $b_j^c = \gamma_c^j(b_j) = b_j \oplus b_{LPB}^j$. En el lado del decodificador, la secuencia de bins puede deducirse mediante los mapeados binarios

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

10 con $j = 0, \dots, B-1$. Si el valor LPB b_{LPB}^j para cada bin b_j se deduce de la misma manera en el lado del codificador y del decodificador, estos mapeados $(\gamma_c^j)^{-1}$ representan los inversos de los mapeados del codificador γ_c^j correspondientes y por ello

$$15 \quad b_j^c \oplus b_{LPB}^j = b_j \oplus b_{LPB}^j \oplus b_{LPB}^j = b_j \oplus 0 = b_j \quad (B29)$$

y por ello, la conversión γ_b de una secuencia de bins b en una secuencia de bins de codificación b^c es invertible.

Finalmente, se investiga la capacidad de inversión de la binarización $b = \gamma_b(s)$ mediante la que cada símbolo s_i , con $i = 0, \dots, N-1$, se mapea sobre una secuencia de bin $b^i = \gamma_b^i(s_i)$. Un símbolo s_i puede decodificarse únicamente dada la correspondiente secuencia de bin b^i si el mapeado de binarización γ_b^i asigna una secuencia de bin b_m^i diferente a cada letra a_m^i del alfabeto A_i para el símbolo s_i . Sin embargo, esta condición no es suficiente, dado que la partición de la secuencia de bins $b = \{b_0, \dots, b_{S-1}\}$ en secuencias de bin b^i que corresponden a los símbolos s_i , con $i = 0, \dots, N-1$, no es conocida para el decodificador. Una condición suficiente se da cuando para cada símbolo s_i , las secuencias de bin b_m^i que se asocian con las letras a_m^i del alfabeto A correspondiente forman un código de prefijo y los mapeados de binarización γ_b^i para cada símbolo s_i , con $i = 0, \dots, N-1$, son conocidos en el lado del decodificador.

Las condiciones de capacidad de decodificación única para el enfoque de codificación entrópico presentado pueden resumirse como sigue:

- los mapeados de binarización γ_b^i representan códigos de prefijo y son conocidos para el decodificador (en el orden de codificación de símbolos)
- los modelos de probabilidad (b_{LPB}^j, p_{LPB}^j) para todos los bins b_j se deducen en la misma forma en el lado del codificador y del decodificador
- la partición del intervalo de probabilidad LPB $(0, 0.5]$ en K intervalos I_k , con $k = 0, \dots, K-1$, es conocida para el decodificador
- el mapeado γ_c^k para cada intervalo de probabilidad I_k , con $k = 0, \dots, K-1$, representa un código únicamente decodificable

40 En lo que sigue, se describen ejemplos del diseño global del codificador y decodificador con más detalle. Se concentrará sobre esquemas de codificación, en los que los modelos de probabilidad $\{b_{LPB}, p_{LPB}\}$ para los bins se estiman directamente en el lado del codificador y del decodificador y los K codificadores binarios usan mapeados V2V descritos anteriormente. Cada símbolo fuente s se deberá asociar con una categoría de símbolos c_s , que determina el tipo de símbolo incluyendo su rango de valores. El orden de los símbolos y las categorías de símbolos asociadas deberán darse por la sintaxis, que se presume es conocida en el lado del codificador y del decodificador.

El diagrama de bloques para un ejemplo de diseño de codificador PIPE y de decodificador PIPE se ilustra en la Figura 19. En el lado del codificador, los símbolos s con categorías de símbolos c_s asociadas se suministran al interior del *binarizador*, que convierte cada símbolo s en una secuencia de bins $b_s = \gamma_b^{c_s}(s)$.

El esquema de binarización $\gamma_b^{c_s}$ usado se determina basándose en la categoría de símbolos c_s . Además, el binarizador asocia cada bin b a una secuencia de bins, con una indicación c_b del modelo de probabilidad, que

específica el modelo de probabilidad que se usa para la codificación del bin b . La indicación c_b del modelo de probabilidad puede deducirse basándose en la categoría del símbolo c_s , el número de bin del bin actual en el interior de la secuencia s de bins, y/o los valores de bins y símbolos ya codificados.

5 El *estimador de probabilidad y asignador* mantiene múltiples modelos de probabilidad, que se caracterizan por pares de valores $\{b_{LPB}, p_{LPB}\}$. Recibe bins b e indicaciones c_b del modelo de probabilidad del binarizador, y envía el valor LPB b_{LPB} y la probabilidad LPB p_{LPB} del modelo de probabilidad indicado a la parte que deduce el bin de codificación y al cuantificador de probabilidad, respectivamente. Posteriormente, se actualiza el modelo de probabilidad $\{b_{LPB}, p_{LPB}\}$ correspondiente usando el valor del bin b recibido.

10 La *parte que deduce el bin* recibe bins b y valores de LPB b_{LPB} asociados del binarizador y el estimador de probabilidad y asignador, respectivamente, y envía bins de codificación b_c , que se deducen mediante $b_c = b \oplus b_{LPB}$, al cuantificador de probabilidad. El *cuantificador de probabilidad* envía cada bin de codificación b_c a uno de los K codificadores binarios. Contiene información acerca de la cuantificación $\{I_k\}$ del intervalo de probabilidad LPB. La probabilidad LPB p_{LPB} , que se asocia con un bin de codificación b_c y se recibe desde el estimador de probabilidad y asignador, se compara con los bordes del intervalo $\{p_k\}$ y el índice del intervalo de probabilidad k , para el que se deduce $p_{LPB} \in I_k$. A continuación, se envía el bin de codificación b_c al codificador binario asociado.

20 Cada uno de los K *codificadores binarios* consiste en una *memoria intermedia de bin* y un *codificador de bin*. Cada memoria intermedia de bin recibe bins de codificación b_c desde el cuantificador de probabilidad y los almacena en el orden de codificación. El codificador de bin implementa un mapeado V2V particular y compara la secuencia de bins en la memoria intermedia de bin con las secuencias de bin que se asocian con palabras de código. Si la secuencia de bin en la memoria intermedia de bin es igual a una de aquellas secuencias de bin, el codificador de bin elimina la secuencia de bin $\{b^c\}$ de la memoria intermedia de bin y escribe la palabra de código asociada ($\{b^c\}$) en el flujo de palabras de código correspondiente. Al final del proceso de codificación para una secuencia de símbolos, para todos los codificadores binarios para los que las memorias intermedias de bin están vacías, se escribe una palabra de código de finalización tal como se describe en la sección 4.3.

30 Los K flujos de palabras de código resultantes pueden transmitirse por separado, empaquetarse o almacenarse, o pueden intercalarse (compárese con la sección 6.2) con la finalidad de transmisión o almacenamiento.

En el lado del decodificador, cada uno de los K decodificadores binarios que consiste en un decodificador de bin y una memoria intermedia de bin recibe un flujo de palabras de código. El decodificador de bin lee las palabras de código ($\{b^c\}$) desde el flujo de palabras de código e inserta la secuencia de bin asociada $\{b^c\}$ en el orden de codificación, en la memoria intermedia de bin.

40 La decodificación de la secuencia de símbolos es impulsada por la sintaxis subyacente. Se envían solicitudes para un símbolo s junto con la categoría del símbolo c_s al binarizador. El *binarizador* convierte esta solicitud de este símbolo en solicitudes de bins. Una solicitud de un bin se asocia con una indicación c_b del modelo de probabilidad, que se deduce de la misma manera que en el codificador, y se envía al estimador de probabilidad y asignador. El *estimador de probabilidad y asignador* funciona de manera similar a su contraparte en el lado del codificador. Basándose en la indicación c_b del modelo de probabilidad, identifica un modelo de probabilidad y envía su valor LPB b_{LPB} y probabilidad LPB p_{LPB} a la parte que deduce el bin y al cuantificador de probabilidad, respectivamente.

45 El *cuantificador de probabilidad* determina uno de los K decodificadores binarios basándose en la probabilidad LPB p_{LPB} , en la misma manera que el codificador binario lo determina en el lado del codificador, elimina el primer bin de codificación b^c en el orden de codificación, de la memoria intermedia de bin correspondiente, y lo envía a la parte que deduce el bin. La *parte que deduce el bin* recibe los bin b^c de codificación y valores LPB b_{LPB} asociados del cuantificador de probabilidad y estimador de probabilidad y asignador, respectivamente, y determina los valores de bin $b = b^c \oplus b_{LPB}$. Como respuesta final a la solicitud de bin enviada por el binarizador, la parte que deduce el bin envía el valor de bin decodificado b al binarizador y al estimador de probabilidad y asignador.

55 En el estimador de probabilidad y asignador, el valor del bin decodificado b se usa para actualizar el modelo de probabilidad $\{b_{LPB}, p_{LPB}\}$, que se eligió mediante el valor c_b asociado, en la misma manera que en el lado del codificador. Finalmente, el binarizador añade el bin b recibido a la secuencia de bins s que ya se ha recibido para una solicitud de símbolo y compara esta secuencia de bin s con las secuencias de bin que se asocian con valores del símbolo por el esquema de binarización $\gamma_b^{c_s}$. Si la secuencia de bin s coincide con una de esas secuencias de bin, el símbolo decodificado s correspondiente se produce como salida como respuesta final a la solicitud de símbolo. En caso contrario, el binarizador envía solicitudes de bin adicionales hasta que se decodifica el símbolo s .

60 La decodificación de una secuencia de símbolos se finaliza si no se reciben más solicitudes de símbolos adicionales, lo que es condicionado por la sintaxis. Los bins de codificación b^c que puedan estar contenidos en las memorias intermedias de bin al final del proceso de decodificación entrópica (como resultado de las palabras de código de

finalización) se descartan.

Después de haber descrito ciertos codificadores PIPE y decodificadores PIPE con relación a las Figs. 3 a 13 y habiendo proporcionado una base matemática con relación a la codificación PIPE en general con relación a las Figs. 14 a 19, se describen más detalles para los aparatos de codificación y decodificación entrópica, con relación a las Figs. 20 a 24. Los siguientes ejemplos de las Figs. 22 a 24 intercalan entre sí no solo flujos de bits codificados PIPE, sino que intercalan el flujo de bits VLC y los flujos de bits codificados PIPE conjuntamente. En comparación con ello, los codificadores PIPE y decodificadores PIPE con intercalado de flujos de bits codificados PIPE, mostrados en las Figs. 7 a 13, meramente proporcionan un intercalado separado de los flujos de bits codificados PIPE. Como ya se ha mencionado anteriormente, incluso esto mismo puede servir como base para conseguir un flujo de bits completamente intercalado mediante el uso de otro par de intercalador/desintercalador 134 y 228, respectivamente (véanse las Figs. 1 y 2), tal como, por ejemplo, mediante el uso del intercalador de flujos de bits tal como se muestra en las Figs. 5 y 6. Sin embargo, las posibilidades descritas a continuación realizan el intercalado de una vez tanto en el flujo de bits VLC como en los flujos de bits codificados PIPE con el uso, en otras palabras, de un intercalador/desintercalador en una etapa 128 y 230, respectivamente.

Antes de describir en detalle casos en donde se intercalan símbolos codificados VLC y PIPE dentro de un flujo de bits, para conseguir un compromiso más adecuado entre la complejidad y la eficiencia de codificación, se describe una estructura básica de los mismos sin intercalado con respecto a las Figs. 20 y 21.

La estructura del aparato de codificación entrópica de la Fig. 1a se muestra en la Fig. 20. El aparato de codificación entrópica convierte un flujo de símbolos fuente 1a, que corresponden a la combinación de símbolos fuente codificados VLC y codificados PIPE de las Figs. 1 y 2, concretamente 106 y 218, respectivamente, en un conjunto de dos o más flujos de bits parciales 12, 12a, correspondiendo el flujo de bits 12a a los flujos de bits 112 y 206 de las Figs. 1 y 2.

Como ya se hecho notar anteriormente, cada símbolo fuente 1a puede tener asociado con él una indicación que especifica si el símbolo fuente está codificando usando códigos VLC estándar dentro del codificador VLC 22a, que se corresponde con el codificador VLC 102 en la Fig. 1, o si el símbolo fuente se ha de codificar con un concepto de codificación PIPE. Como ya se ha descrito anteriormente con respecto a las Figs. 1 y 2, esta indicación no puede transmitirse explícitamente al lado de decodificación. Por el contrario, la indicación de asociado puede proceder del tipo o categoría del símbolo fuente en sí.

Los símbolos 1b codificados VLC se codifican con códigos VLC estándar, que a su vez, pueden depender de la categoría de símbolos recientemente mencionada o tipo de símbolo usando un codificador VLC 22a. Las palabras de código 11a correspondientes se escriben en un flujo de bits parcial 12a diferente. Los símbolos 1 de código no VLC se codifican usando codificación PIPE como se ha descrito anteriormente con respecto a las Figs. 1 y 3, por ejemplo, con lo que se obtienen múltiples flujos de bits 12 parciales. Algunos de los símbolos fuente 1a pueden haberse binarizado ya en una forma en la que consiste la binarización en dos partes ya anteriormente mencionada con respecto a la Fig. 1a. Otras de estas partes pueden codificarse con el enfoque PIPE y escribirse en los flujos de bits 12 parciales correspondientes. La otra parte de la secuencia de bin puede codificarse con los códigos VLC estándar y escribirse en el flujo de bits parcial 12a correspondiente.

El aparato de decodificación entrópica básico que se ajusta a la Fig. 20 se muestra en la Fig. 21.

El decodificador realiza básicamente las operaciones inversas del codificador de la Fig. 20, de modo que la secuencia previamente codificada de símbolos fuente 27, 27a se decodifica a partir de un conjunto de dos o más flujos de bits parciales (24, 24a). El decodificador incluye dos flujos de proceso diferentes: un flujo para las solicitudes de datos, que replica el flujo de datos del codificador, y un flujo de datos, que representa la inversa del flujo de datos del codificador. En la ilustración de la Fig. 21, las flechas discontinuas representan el flujo de solicitudes de datos, mientras que las flechas continuas representan el flujo de datos. Los bloques de construcción del decodificador replican básicamente los bloques de construcción del codificador, pero implementan las operaciones inversas.

Cada solicitud de símbolo 13a puede asociarse con una indicación que especifica si el símbolo fuente se codifica usando códigos VLC estándar o con el concepto de codificación PIPE. Como ya se ha mencionado anteriormente con respecto a la Fig. 20, Este indicación puede proceder de la regla de análisis o la sintaxis de los elementos de sintaxis representados por el símbolo fuente en sí. Por ejemplo, con respecto a las Figs. 1 y 2, se ha descrito que diferentes tipos de elementos de sintaxis pueden asociarse con diferentes esquemas de codificación, concretamente codificación VLC o codificación PIPE. Lo mismo se puede aplicar para diferentes partes de binarizaciones, o más generalmente, otras simbolizaciones de los elementos de sintaxis. Si un símbolo se codifica VLC, la solicitud se pasa al decodificador VLC 22a y se lee una palabra de código VLC 23a desde un flujo de bits parcial 24a distinto. Se produce la salida del símbolo de decodificación 27a correspondiente. Si el símbolo se codifica con PIPE, el símbolo 27 se decodifica a partir de un conjunto de flujos de bits 24 parciales tal como se ha descrito anteriormente con

respecto a la Fig. 4, por ejemplo.

Algunos de los símbolos fuente pueden binarizarse en una forma tal que la binarización consiste en dos partes. Una de estas partes se codifica con el enfoque PIPE y se decodifica correspondientemente a partir de los flujos de bits parciales 24 asociados. Y la otra parte de la secuencia de bins se codifica con códigos VLC estándar y se decodifica con un decodificador VLC 22a que lee las palabras de código 23a correspondientes a partir de un flujo de bits parcial 24a distinto.

Transmisión y multiplexado de flujos de bits parciales (codificados VLC y codificados PIPE)

Los flujos de bits parciales 12, 12a que se crean por el codificador pueden transmitirse por separado, o pueden multiplexarse en un único flujo de bits, o las palabras de código de los flujos de bits parciales pueden intercalarse en un único flujo de bits.

Cada flujo de bits parcial para una cantidad de datos puede escribirse en un paquete de datos. La cantidad de datos puede ser un conjunto arbitrario de símbolos fuente tal como una imagen fija, un campo o un cuadro de una secuencia de video, un fragmento de una imagen fija, un fragmento de un campo o cuadro de una secuencia de video, o una trama de muestras de audio, etc.

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

El paquete de datos 300 consiste en una cabecera y una partición para los datos de cada flujo de bits parcial (para la cantidad de datos considerada). La cabecera 301 del paquete de datos contiene indicaciones para la partición del (resto del) paquete de datos en segmentos de datos 302 del flujo de bits. Junto a las indicaciones para la partición, la cabecera puede contener información adicional. Las indicaciones para la partición del paquete de datos pueden ser las localizaciones del comienzo de los segmentos de datos en unidades de bits o bytes o múltiplos de bits o múltiplos de bytes. Las localizaciones del comienzo de los segmentos de datos pueden codificarse como valores absolutos en la cabecera del paquete de datos, tanto con relación al comienzo del paquete de datos como con relación al final de la cabecera o con relación al comienzo del paquete de datos previo. Las localizaciones del comienzo de los segmentos de datos pueden codificarse diferencialmente, es decir, solamente 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 pueden deducirse basándose en información ya conocida o transmitida tal como el tamaño global del paquete de datos, el tamaño de la cabecera, el número de segmentos de datos en el paquete de datos, la localización del comienzo de los segmentos de datos precedentes. La localización del comienzo del primer paquete de datos puede no estar codificada, sino deducirse basándose en el tamaño de la cabecera del paquete de datos. En el lado del decodificador, las indicaciones de partición transmitidas se usan para deducir el comienzo de los segmentos de datos. Los segmentos de datos se usan entonces como flujos de bits parciales 12, 12a y los datos contenidos en los segmentos de datos se suministran al interior de los decodificadores de bin correspondientes y decodificadores VLC en orden secuencial.

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

Para algunas aplicaciones, el multiplexado anteriormente descrito de los flujos de bits parciales (para una cantidad de símbolos fuente) en un paquete de datos puede tener las siguientes desventajas: por un lado, para pequeños paquetes de datos, el número de bits para la información lateral que se requiere para señalización de la partición puede convertirse en significativo con relación a los datos reales en los flujos de bits parciales, lo que finalmente reduce la eficiencia de codificación. Por otro lado, el multiplexado puede no ser adecuado para aplicaciones que requieren un bajo retardo (por ejemplo, para aplicaciones de videoconferencia). Con el multiplexado descrito, el codificador no puede iniciar la transmisión de un paquete de datos antes de que se hayan creado completamente los flujos de bits parciales, dado que las localizaciones del inicio de las particiones no son conocidas antes. Adicionalmente, en general, el decodificador ha de esperar hasta que recibe el comienzo del último segmento de datos antes de que pueda comenzar la decodificación de un paquete de datos. Para aplicaciones tales como sistemas de videoconferencia, estos retardos pueden añadirse a un retardo global adicional del sistema de varias imágenes de video (en particular, para tasas de bits que se aproximan a la tasa de bits de transmisión y para codificadores/decodificadores que requieren prácticamente el intervalo de tiempo entre dos imágenes para codificación/decodificación de una imagen), lo que es crítico para dichas aplicaciones. Para superar las desventajas para ciertas aplicaciones, el codificador puede configurarse en una forma tal que las palabras de código que se generan por los dos o más codificadores de bin y el codificador VLC se intercalan en un único flujo de bits. El flujo de bits con las palabras de código intercaladas, puede enviarse directamente al decodificador (cuando se desprecia un pequeño retardo de la memoria intermedia, véase a continuación). En el lado del decodificador, los dos o más decodificadores de bin y el decodificador VLC leen las palabras de código directamente del flujo de bits en el orden de decodificación; la decodificación puede iniciarse con el primer bit recibido. Además, no se requiere ninguna

información lateral para señalización del multiplexado (o intercalado) de los flujos de bits parciales.

- La estructura básica de un codificador con intercalado de palabras de código se muestra en la Fig. 22. Los codificadores de bin 10 y el codificador VLC 10a no escriben las palabras de código directamente en los flujos de bits parciales, sino que se conectan con una única memoria intermedia 29 de palabras de código, a partir de la que se escriben las palabras de código al flujo de bits 34 en orden de codificación. Los codificadores de bin 10 envían solicitudes de una o más entradas de palabras de código 28 a la memoria intermedia 29 de palabras de código y posteriormente envía las palabras de código 30 a la memoria intermedia 29, que se almacenan en las entradas de memoria intermedia reservadas. El codificador VLC 10a escribe directamente las palabras de código VLC 30a en la memoria intermedia 29 de palabras de código. Se accede a palabras de código 31 (en general de longitud variable) de la memoria intermedia 29 de palabras de código mediante un escritor 32 de palabras de código, que escribe los bits 33 correspondientes en el flujo de bits 34 producido. La memoria intermedia 29 de palabras de código funciona como una memoria primero en entrar primero en salir; las entradas de palabras de código que se reservan antes se escriben antes en el flujo de bits.
- La memoria intermedia de palabras de código puede funcionar como sigue. Si se envía un nuevo bin 7 a una memoria intermedia de bin 8 particular y el número de bins ya almacenados en la memoria intermedia de bin es cero y no hay actualmente reservada ninguna palabra de código en la memoria intermedia de palabras de código para el codificador de bin que está conectado con la memoria intermedia de bin particular, el codificador de bin 10 conectado envía una solicitud a la memoria intermedia de palabras de código, mediante la que se reservan una o más entradas de palabras de código en la memoria intermedia 29 de palabras de código para el codificador de bin 10 particular. Las entradas de palabras 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 normalmente por el tamaño máximo de la palabra de código para el codificador de bin correspondiente. La siguiente palabra de código o las siguientes palabras de código que se producen por el codificador de bin (para el que se ha reservado la entrada de palabras de código o las entradas de palabras 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 bin particular se llenan con palabras de código y se envía el siguiente bin a la memoria intermedia de bin que está conectada con el codificador de bin particular, se reservan una o más palabras de código nuevas en la memoria intermedia de palabras de código para el codificador de bin particular, etc. El codificador VLC 10a escribe directamente las palabras de código VLC 30a en la siguiente entrada libre de la memoria intermedia 29 de palabras de código, es decir para el codificador VLC la reserva de palabras de código y la escritura de las palabras de código se realiza de una vez. La memoria intermedia 29 de palabras de código representa una memoria intermedia primero en entrar primero en salir en una cierta forma. Las entradas a la memoria intermedia se reservan en orden secuencial. Las palabras de código para las que se han reservado antes las entradas de memoria intermedia correspondientes se escriben antes en el flujo de bits. El escritor 32 de palabras de código comprueba el estado de la memoria intermedia 29 de palabras de código, tanto continuamente como después de que se escriba una palabra de código 30 en la memoria intermedia 29 de palabras de código. Si la primera entrada a la memoria intermedia contiene una palabra de código completa (es decir, la entrada de la memoria intermedia no está reservada, pero incluye una palabra de código), la correspondiente palabra de código 31 y la correspondiente entrada de la memoria intermedia se eliminan de la memoria intermedia 29 de palabras de código y los bits de la palabra de código 33 se escriben en el flujo de bits. Este proceso se repite hasta que la primera entrada de la memoria intermedia no contiene una palabra de código (es decir, está reservada o libre). Al final del proceso de decodificación, es decir, si se han procesado todos los símbolos fuente de la cantidad de datos considerada, la memoria intermedia de palabras de código debe purgarse. Para ese proceso de purga, se aplica lo siguiente para cada memoria intermedia de bin/codificador de bin como una primera etapa: si la memoria intermedia de bin no contienen bins, se añade un bin con un valor particular o un valor arbitrario hasta que la secuencia de bin resultante representa una secuencia de bins que está asociada con una palabra de código (como se hecho notar anteriormente, una forma preferida de añadir bins es añadir dichos valores de bin que producen la palabra de código más corta posible —o una de ellas— que se asocia con una secuencia de bins que contiene el contenido original de la memoria intermedia de bin como prefijo), entonces se escribe la palabra de código en la siguiente entrada de la memoria intermedia reservada para el codificador de bin correspondiente y se vacía (la correspondiente) memoria intermedia de bin. Si se ha reservado más de una entrada de memoria intermedia para uno o más codificadores de bin, la memoria intermedia de palabras de código puede contener aún entradas de palabras de código reservadas. En ese caso, estas entradas de palabras de código se rellenan con palabras de código arbitrarias, pero válidas, para los codificadores de bin correspondientes. Puede insertarse la palabra de código válida más corta o una de las palabras de código válidas más cortas (si hay múltiples). El codificador VLC no requiere ninguna finalización. Finalmente, todas las palabras de código restantes en la memoria intermedia de palabras de código se escriben en el flujo de bits.
- Se ilustran en la Fig. 23 dos ejemplos para el estado 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 ellas son entradas VLC) y 3 entradas reservadas. Además, se marca la siguiente entrada de la memoria intermedia libre. La primera entrada se llena con una palabra de código (es decir, el codificador de bin 2 justamente escribe una palabra de código en una entrada previamente reservada). En la siguiente etapa, esta palabra de código

se eliminará de la memoria intermedia de palabras de código y se escribe en el flujo de bits. A continuación, la primera palabra de código reservada para el codificador de bin 3 es la primera entrada de memoria intermedia, pero esta entrada no puede eliminarse de la memoria intermedia de palabras de código, dado que solo está reservada, pero no se ha escrito ninguna palabra de código 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 ellas es una entrada de memoria intermedia VLC) y 4 entradas reservadas. La primera entrada se marca como reservada y por ello el escritor de palabras de código no puede escribir una palabra de código al flujo de bits. Aunque están contenidas 4 palabras de código en la memoria intermedia de palabras de código, el escritor de palabras de código ha de esperar hasta que se escribe la palabra de código en la primera entrada de memoria intermedia reservada para el codificador de bin 3. Obsérvese que las palabras de código deben escribirse en el orden en que se reservaron, para ser capaz de invertir el proceso en el lado del decodificador (véase a continuación). Y adicionalmente obsérvese que una entrada de memoria intermedia VLC está siempre completa, dado que la reserva y escritura de la palabra de código se realiza de una vez.

La estructura básica de un decodificador con intercalado de palabras de código se muestra en la Fig. 24. Los decodificadores bin 22 y el decodificador VLC 2a no leen las palabras de código directamente de flujos de bits parciales separados, sino que se conectan a una memoria intermedia 38 de bits, desde la que se leen las palabras de código 37, 37a en el orden de codificación. Debería tomarse nota que no se requiere necesariamente la memoria intermedia 38 de bits, dado que las palabras de código podrían leerse también directamente desde el flujo de bits. La memoria intermedia 38 de bits se incluye principalmente en la ilustración para separar claramente aspectos diferentes de la cadena de procesamiento. Los bits 39 del flujo de bits 40 con palabras de código intercaladas se insertan secuencialmente en la memoria intermedia 38 de bits, que representa una memoria intermedia primero en entrar primero en salir. Si un decodificador de bin 22 particular recibe una solicitud de una o más secuencias de bin 35, el decodificador de bin 22 lee una o más palabras de código 37 de la memoria intermedia 38 de bits a través de la solicitud de bits 36. El decodificador puede decodificar instantáneamente los símbolos fuente. De modo similar, si el decodificador VLC 22a recibe una solicitud de un nuevo símbolo 19a lee la palabra de código VLC 37a correspondiente desde la memoria intermedia 38 de bits y devuelve el símbolo decodificado 27a. Obsérvese que el codificador (como se ha descrito anteriormente) debe asegurar mediante la operación en forma adecuada de la memoria intermedia de palabras de código que las palabras de código se escriben en el mismo orden al flujo de bits en el que se solicitan por los decodificadores de bin. En el decodificador, todo el proceso de decodificación se activa mediante solicitudes de símbolos fuente. Parámetros tales como el número de palabras de código que se reservan en el lado del codificador por un codificador de bin particular y el número de palabras de código que se leen por el decodificador de bin correspondiente deben ser los mismos.

35 Intercalado de palabras de código de longitud variable con restricciones de bajo retardo

El intercalado de palabras de código descrito no requiere que se envíe ninguna información de partición como información lateral. Y dado que las palabras de código se intercalan en el flujo de bits, el retardo es en general pequeño. Sin embargo, no se garantiza que se obedezca a 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). Adicionalmente, el tamaño de la memoria intermedia requerido para memoria intermedia de palabras de código puede hacerse teóricamente muy grande. Cuando se considera el ejemplo de la Fig. 23(b), sería posible que no se envíen bins adicionales a la memoria intermedia 3 de bin y por ello el codificador de bin 3 no enviará ninguna nueva palabra de código a la memoria intermedia de palabras de código hasta que se aplique el proceso de purgado al final del paquete de datos. Entonces todas las palabras de código para los codificadores de bin 1 y 2 tendrían que esperar hasta el final del paquete de datos, antes de puedan escribirse en el flujo de bits. Este inconveniente puede sortearse añadiendo un mecanismo adicional al proceso de codificación (y también al proceso de decodificación tal como se describe más adelante). El concepto básico de ese mecanismo adicional es que si una medida en relación al retardo o a una delimitación superior del retardo (véase a continuación) supera un umbral especificado, la primera entrada en la memoria intermedia reservada se llena mediante el purgado de la memoria intermedia de bin correspondiente (usando un mecanismo similar al del final del paquete de datos). Mediante dicho mecanismo, el número de entradas de memoria intermedia en espera se reduce hasta que la medida del retardo asociada sea menor que el umbral especificado. En el lado del decodificador, los bin que se han insertado en el lado del codificador para obedecer a la restricción de retardo deben descartarse. Para este descarte de los bins puede usarse básicamente el mismo mecanismo que en el lado del codificador.

Después de haber descrito en detalle posibilidades para intercalado de los flujos de bits de codificación VLC y PIPE, en lo que sigue, la descripción se enfoca de nuevo en los elementos de sintaxis ya mencionados anteriormente descompuestos en símbolos fuente tal como se ha mencionado con respecto a las Figs. 1b, 1c y 2b. Con finalidades de ilustración, la siguiente descripción asume que los elementos de sintaxis así descompuestos son niveles de coeficiente de transformada absoluta. Sin embargo, esto es solamente un ejemplo, y pueden manejarse de la misma manera otros tipos de elementos de sintaxis. *En particular, en lo que sigue se describe la codificación de niveles absolutos mediante la partición y el uso de diferentes códigos de entropía en codificadores de imagen y video basados en bloques.*

Por ejemplo, las imágenes de la secuencia de video se descomponen normalmente en bloques. Los bloques o los componentes de color de los bloques se predicen mediante o bien predicción de movimiento compensado o intra predicción. Los bloques pueden tener diferentes tamaños y pueden ser o bien cuadráticos o bien rectangulares.

5 Todas las muestras de un bloque o un componente de color de bloque se predicen usando el mismo conjunto de parámetros de predicción, tales como índices de referencia (que identifican una imagen de referencia en el conjunto de imágenes ya codificadas), parámetros de movimiento (que especifican una medición para el movimiento de los bloques entre una imagen de referencia y la imagen actual), parámetros para la especificación del filtro de interpolación, modos de predicción intra, etc. Los parámetros de movimiento pueden representarse por vectores de desplazamiento con un componente horizontal y vertical o mediante parámetros de movimiento de orden más alto tales como parámetros de movimiento afinados que consisten en 6 componentes. Es posible también que se asocie más de un conjunto de parámetros de predicción (tales como índices de referencia y parámetros de movimiento) 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 el 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 un desplazamiento constante (que se añade a la suma ponderada) pueden ser o bien fijos para una imagen, o una imagen de referencia, o un conjunto de imágenes de referencia, o pueden incluirse en el conjunto de parámetros de predicción para el bloque correspondiente. De modo similar, las imágenes fijas también se descomponen frecuentemente en bloques, y los bloques se predicen por un método de predicción intra (que puede ser un método de predicción intra espacial o un método de predicción intra simple que predice el componente continuo de bloque). En el caso de una esquina, la señal de predicción puede ser también cero.

La diferencia entre los bloques originales o los componentes de color de los bloques originales y las señales de predicción correspondientes, que también se denominan como señales residuales, se transforman y cuantifican usualmente. Se aplica una transformada bidimensional a la señal residual y se cuantifican los coeficientes de transformada resultantes. Para esta codificación de transformada, los bloques o los componentes de color de los bloques, para los que se ha usado un conjunto particular de parámetros de predicción, pueden repartirse adicionalmente antes de la aplicación de la transformada. Los bloques de transformada pueden ser iguales o más pequeños que los bloques que se usan para predicción. Es posible también que un bloque de transformada incluya más de uno de los bloques que se usan para predicción. Diferentes bloques de transformada en una imagen fija o un cuadro de una secuencia de vídeo pueden tener diferentes tamaños y los bloques de transformada pueden representar bloques cuadráticos o rectangulares.

35 Todos estos parámetros de predicción y residuales pueden formar el flujo de elementos de sintaxis 138 y 226, respectivamente.

Los coeficientes de transformada cuantificados resultantes, también denominados como niveles de coeficiente de transformada, pueden transmitirse a continuación usando codificación entrópica mediante uno de los esquemas de codificación anteriores. Con este fin, puede mapearse un bloque de niveles de coeficiente de transformada sobre un vector (es decir, un conjunto ordenado) de valores de coeficiente de transformada usando un escaneado, en donde pueden usarse diferentes escaneados para diferentes bloques. Frecuentemente se usa un escaneado en zigzag. Para bloques que contienen solo muestras de un campo de un cuadro entrelazado (estos bloques pueden ser bloques en campos codificados o bloques de campos en cuadros codificados), es común también usar un escaneado diferente diseñado específicamente para bloques de campos. Un posible esquema de codificación para la codificación de la secuencia ordenada resultante de coeficientes de transformada es codificación a nivel de tirada. Normalmente, un gran número de niveles de coeficiente de transformada son cero, y un conjunto de niveles de coeficiente de transformada sucesivos que sean iguales a cero puede representarse eficientemente mediante la codificación del número de niveles de coeficiente de transformada sucesivos que son iguales a cero (la tirada) por un elemento de sintaxis respectivo. Para los coeficientes de transformada restantes (no cero), el nivel real se codifica en la forma de elementos de sintaxis respectivos. Hay varias alternativas de códigos a nivel de tirada. La tirada antes de un coeficiente no cero y el nivel del coeficiente de transformada no cero pueden codificarse juntos usando un elemento de sintaxis único. Frecuentemente, se incluyen elementos de sintaxis especiales para el final del bloque, que se envía después del último coeficiente de transformada no cero. O es posible codificar primero el número de niveles de coeficiente de transformada no cero, y dependiendo de este número, se codifican los niveles y tiradas.

55 Se usa un enfoque en alguna forma diferente para la codificación entrópica CABAC altamente eficiente en H.264/AVC. En este caso, la codificación de los niveles de coeficiente de transformada se reparte en tres etapas. En la primera etapa, se transmite para cada bloque de transformada un elemento de sintaxis binario indicador_coficiente_codificado, lo que señala si el bloque de transformada contiene niveles de coeficiente de transformada significativos (es decir, coeficientes de transformada que sean no cero). Si este elemento de sintaxis indica que están presentes elementos de coeficiente de transformada significativos, se codifica un mapa de significación evaluado en binario, que especifica cuál de los niveles de coeficiente de transformada tiene valores no cero. Y a continuación, en un orden de escaneado inverso, se codifican los valores de los niveles de coeficiente de transformada no cero. El mapeado de significación se codifica en un flujo de elemento de sintaxis 138 como sigue.

Para cada coeficiente en el orden de escaneado, se codifica un elemento de sintaxis binario indicador_coficiente_significativo, que especifica si el nivel de coeficiente de transformada correspondiente no es igual a cero. Si el bin del indicador_coficiente_significativo es igual a uno, es decir, si existe un nivel de coeficiente de transformada no cero en esta posición de escaneado, se codifica un elemento de sintaxis binario adicional último_indicador_coficiente_significativo. Este bin indica si el nivel de coeficiente de transformada significativo es el último nivel de coeficiente de transformada significativo dentro del bloque o si niveles de coeficiente de transformada significativos adicionales siguen en el orden de escaneado. Si último_indicador_coficiente_significativo indica que no hay ningún coeficiente de transformada significativo adicional que siga, no se codifican elementos de sintaxis adicionales para la especificación del mapa de significación para el bloque. En la siguiente etapa, se codifican los valores de los niveles de coeficiente de transformada significativos, cuyas localizaciones dentro del bloque ya se han determinado por el mapa de significación. Los valores de los niveles de coeficiente de transformada significativo se codifican en orden de escaneado inverso mediante el uso de los siguientes tres elementos de sintaxis. El elemento de sintaxis binario coeficiente_absoluto_mayor_uno indica si el valor absoluto del nivel de coeficiente de transformada significativo es mayor que uno. Si el elemento de sintaxis binario coeficiente_absoluto_mayor_uno indica que el valor absoluto es mayor que uno, se envía un elemento de sintaxis adicional coeficiente_absoluto_nivel_menos_dos que especifica el valor absoluto del nivel de coeficiente de transformada menos dos. Esta es la clase de elementos de sintaxis cuyo procesamiento puede realizarse de acuerdo con la Fig. 1b, 1c y 2b. Finalmente, se codifica el elemento de sintaxis binario indicador_signo_coficiente, que especifica el signo del valor del coeficiente de transformada, para cada nivel de coeficiente de transformada significativo. Debería tomarse nota de nuevo que los elementos de sintaxis que se refieren al mapa de significación se codifican en orden de escaneado, mientras que los elementos de sintaxis que se refieren a los valores actuales de los niveles de coeficientes de transformada se codifican en orden de escaneado inverso permitiendo el uso de modelos de contexto más adecuados. Es posible también que se use un patrón de escaneado adaptativo para el mapa de significación como en el primer modelo de prueba de H.265/HEVC. Otro concepto se usa para la codificación de los niveles de coeficiente de transformada absoluta para la transformada de bloques mayores de 4x4 en el primer modelo de prueba de H.265/HEVC. En caso de bloques de transformada mayores de 4x4, el bloque de transformada mayor se particiona en bloques de 4x4 y los bloques de 4x4 se codifican en el orden de escaneado mientras que se usa el orden de escaneado inverso para cada uno de los bloques 4x4.

En la codificación entrópica CABAC en H.264/AVC, todos los elementos de sintaxis para los niveles de coeficiente de transformada se codifican usando un modelizado de probabilidad binaria. El elemento de sintaxis no binario coeficiente_absoluto_nivel_menos_dos, por ejemplo, se binariza primero, es decir, se mapea sobre una secuencia de decisiones binarias (bins), y estos bins se codifican secuencialmente. El elemento de sintaxis binaria indicador_coficiente_significativo, último_indicador_coficiente_significativo, coeficiente_absoluto_mayor_uno, e indicador_signo_coficiente se codifican directamente. Cada bin codificado (que incluye los elementos de sintaxis binaria) se asocia con un contexto. Un contexto representa un modelo de probabilidad para una clase de bins codificados. Una medida con relación a la probabilidad para uno de los dos posibles valores de bin se estima para cada contexto basándose en los valores de los bins que ya se han codificado con el contexto correspondiente. Para varios bins relativos a la codificación de transformada, el contexto que se usa para la codificación se selecciona basándose en elementos de sintaxis ya transmitidos o basándose en la posición dentro de un bloque.

Después de la codificación del mapa de significación, los bloques se procesan en orden de escaneado inverso. Como se ha mencionado anteriormente, otro concepto se usa en el primer modelo de prueba de H.265/HEVC. Los bloques de transformada mayores de 4x4 se particionan en bloques de 4x4 y los bloques de 4x4 resultantes se procesan en orden de escaneado, mientras que los coeficientes de los bloques de 4x4 se codifican en orden de escaneado inverso. La siguiente descripción es válida para todos los bloques de 4x4 en el primer modelo de prueba de H.265/HEVC y en H.264/AVC y también para bloques 8x8 en H.264/AVC y esta descripción puede aplicarse también a la construcción del flujo de elementos de sintaxis 138 y 226, respectivamente.

Si una posición de escaneado significativa, es decir, el coeficiente es diferente de cero, se transmite el elemento de sintaxis binaria coeficiente_absoluto_mayor_uno dentro del flujo 138. Inicialmente (dentro de un bloque), se selecciona el segundo modelo de contexto del conjunto de modelos de contexto correspondientes para el elemento de sintaxis coeficiente_absoluto_mayor_uno. Si el valor codificado de cualquier elemento de sintaxis coeficiente_absoluto_mayor_uno dentro del bloque es igual a uno (es decir, el coeficiente absoluto es mayor de 2), el modelizado de contexto conmuta de vuelta al primer modelo de contexto del conjunto y usa este modelo de contexto hasta el final del bloque. En caso contrario (todos los valores codificados de coeficiente_absoluto_mayor_uno dentro del bloque son cero y los niveles de coeficiente absoluto correspondientes son iguales a uno), el modelo de contexto se elige dependiendo del número de elementos de sintaxis coeficiente_absoluto_mayor_uno iguales a cero que ya se han procesado en el escaneado inverso del bloque considerado. La selección del modelo de contexto para el elemento de sintaxis coeficiente_absoluto_mayor_uno puede resumirse por la ecuación siguiente, en donde el índice modelo de contexto actual C_{i+1} , se selecciona basándose en el índice del modelo de contexto previo C , y el valor del elemento de sintaxis previamente codificado coeficiente_absoluto_mayor_uno, que se representa por bin_i en la ecuación. Para el primer elemento de sintaxis coeficiente_absoluto_mayor_uno dentro del bloque, el índice del modelo de contexto se fija igual a $C_1 = 1$.

$$C_{i+1}(C_i, bin_i) = \begin{cases} 0 & bin_i = 1 \\ \min(C_i + 1, 4) & bin_i = 0 \end{cases}$$

El segundo elemento de sintaxis para codificación de los niveles de coeficiente de transformada absoluta, coeficiente_absoluto_nivel_menos_dos solo se codifica, cuando el elemento de sintaxis coeficiente_absoluto_mayor_uno para la misma posición de escaneado es igual a uno. El elemento de sintaxis no binaria coeficiente_absoluto_nivel_menos_dos se binariza en una secuencia de bins y para el primer bin de esta binarización; se selecciona un índice de modelo de contexto tal como se describe en lo que sigue. Los bin restantes de la binarización se codifican con contextos fijos. El contexto para el primer bin de la binarización se selecciona como sigue. Para el primer elemento de sintaxis coeficiente_absoluto_nivel_menos_dos, se selecciona el primer modelo de contexto del conjunto de modelo de contextos para el primer bin del elemento de sintaxis coeficiente_absoluto_nivel_menos_dos, el índice del modelo de contexto correspondientes se fija igual a $C_1 = 0$. Para cada primer bin adicional del elemento de sintaxis coeficiente_absoluto_nivel_menos_dos, el modelizado de contexto conmuta al siguiente modelo de contexto en el conjunto, en donde el número de modelos de contexto en el conjunto se limita 5. La selección del modelo de contexto puede expresarse por la fórmula siguiente, en donde el índice de modelo de contexto actual C_{r+1} se selecciona basándose en el índice de modelo de contexto previo C_r .

$$C_{r+1}(C_r) = \min(C_r + 1, 4)$$

Como se ha mencionado anteriormente, para el primer elemento de sintaxis coeficiente_absoluto_permanece_menos_dos dentro de un bloque, el índice de modelo de contexto se fija igual a $C_r = 0$. Obsérvese que pueden definirse diferentes conjuntos de modelos de contexto para los elementos de sintaxis coeficiente_absoluto_mayor_uno y coeficiente_absoluto_permanece_menos_dos. Obsérvese también que para el primer modelo de prueba de H.265/HEVC, los bloques de transformada mayores de 4x4 pueden particionarse en bloques de 4x4. Los bloques de 4x4 particionados pueden procesarse en orden de escaneado y para cada bloque de 4x4 particionado, puede deducirse un conjunto de contexto basándose en el número de coeficientes mayores que uno en el bloque de 4x4 previo. Para el primer bloque de 4x4 de un bloque de transformada mayor de 4x4 y para bloques de transformada de 4x4 de origen puede usarse un conjunto de contexto separado.

Esto es, en cualquier caso que se use en la siguiente descripción codificación basada en contexto para cualquiera de los símbolos fuente en los que coeficiente_absoluto_mayor_uno y coeficiente_absoluto_permanece_menos_dos pueden descomponerse como sigue, entonces esta deducción de contexto puede usarse por el asignador 114 y 212 y el codificador/decodificador VLC 102 y 202, por ejemplo.

Para reducir la complejidad en términos de número de bins procesados por CABAC o PIPE en comparación con las técnicas del estado de la técnica y también en términos de complejidad de cálculo o también para incrementar la eficiencia de codificación, la explicación a continuación describe un enfoque para la codificación de niveles absolutos mediante el uso de códigos de longitud variable diferentes para diferentes particiones 140₁ a 140₃ en codificadores y decodificadores de imagen y video. La posibilidad descrita a continuación puede ser, sin embargo, aplicada a cualquier clase de niveles absolutos para codificadores de imagen y video, como diferencias del vector de movimiento o coeficientes del filtro de bucle adaptativo. Aunque la codificación de niveles de coeficiente de transformada se realiza como se describe a continuación, la codificación del mapa de significación puede permanecer como en el primer modelo de prueba de H.265/HEVC o como se ha descrito anteriormente, o la codificación del mapa de significación puede realizarse como en H.264/AVC o en otra forma.

Como se ha descrito anteriormente, la codificación de los niveles de transformada absoluta se realizan en múltiples particiones 140₁₋₃. El esquema de codificación se ha ilustrado como ejemplo la Figura 1b con tres particiones 140₁₋₃. Las delimitaciones 142 y 144 del esquema son variables dando como resultado tamaños de partición variables. El esquema de codificación se realiza como sigue.

Se usa un primer código de entropía para codificar el primer componente o símbolo fuente, es decir el nivel de coeficiente de transformada absoluta (z) en caso igual es más pequeño que límite1, o límite1 si no es igual. Si el nivel de coeficiente de transformada absoluta es mayor que o igual a la delimitación límite1 de la primera partición 140₁, que la delimitación límite1 (142) de la primera partición (140₁) se resta del nivel de coeficiente de transformada absoluta y el valor resultante z' se codifica con un segundo código de entropía. Si el nivel del coeficiente de transformada absoluta restante z' es mayor que o igual a una delimitación límite2-límite1 para la segunda partición 140₂, entonces la delimitación límite2-límite1 de la segunda partición se resta de nuevo del nivel de coeficiente de transformada absoluta z' y el valor resultante se codifica con un tercer código de entropía. Hablando en general, cuando se alcanza la delimitación de una partición, se usa el código de entropía para la siguiente partición en la delimitación para codificar el valor resultante de los niveles de coeficiente de transformada absoluta menos la delimitación de la partición correspondiente.

Los códigos de entropía pueden ser códigos de longitud variable simples tales como códigos de longitud de tirada o

tablas de búsqueda (por ejemplo código Huffman) o códigos de entropía más complejos que empleen modelos de probabilidad como CABAC o PIPE. El número de particiones y la delimitación de las particiones pueden ser variables o dependientes del elemento de sintaxis real. El concepto de partición mostrado en la Fig. 1b tiene los siguientes beneficios. En lo que sigue, se usan los coeficientes de transformada absoluta como un ejemplo, pero se debería entender que pueden sustituirse con cualquier otro elemento de sintaxis. Como un ejemplo, la distribución de probabilidad de los niveles de coeficiente de transformada absoluta puede tener aproximadamente una distribución geométrica. Por lo tanto, los códigos de entropía optimizados para distribuciones geométricas pueden emplearse para codificación de los niveles de coeficiente de transformada absoluta. Pero dicho modelo no es siempre óptimo localmente, incluso si se emplean el modelizado del contexto y la selección del modelo de probabilidad. Por ejemplo, para un bloque de transformada, los niveles de coeficiente de transformada absoluta locales siguen una distribución que no es geométrica en absoluto si contiene la misma cantidad de niveles de coeficiente de transformada de rango bajo y medio, en tanto que el modelo puede mantenerse cierto (con una cierta precisión) para una cantidad específica de bloques en imagen o video debido a la ley de los grandes números. Para dicho caso, la distribución geométrica no es un modelo adecuado. También, cuando se consideran los niveles de coeficiente de transformada absoluta con grandes valores, la distribución de estos grandes valores es frecuentemente uniforme. El concepto de partición permite diferentes modelos de probabilidad para diferentes niveles del coeficiente de transformada absoluta. Para valores absolutos más bajos, pueden aplicarse códigos de entropía más complejos para una eficiencia mayor, mientras para niveles absolutos mayores pueden emplearse códigos de entropía menos complejos para reducir la complejidad.

Como se ha mencionado anteriormente, se usan códigos de entropía adecuados para diferentes particiones. Pueden emplearse tres tipos de códigos de entropía. El primer código de entropía usa PIPE. Sin embargo, debería tomarse nota que, de acuerdo con una alternativa, puede usarse alternativamente un método de codificación entrópica como CABAC o cualquier otro codificador aritmético. Esto es, los primeros símbolos s_1 (véase la Fig. 2b) pueden codificarse a través de la trayectoria de codificación PIPE. El subdivisor 100 y el recombinador 220 actúan en consecuencia.

Para el segundo tipo, pueden usarse los códigos Golomb y algunas variantes truncadas que incluyen subconjuntos (por ejemplo, códigos Golomb-Rice). Esto es, los segundos símbolos s_2 (véase la Fig. 2b) pueden codificarse a través de dichos códigos VLC en el codificador/decodificador VLC 102/202. El subdivisor 100 y el recombinador 220 actúan en consecuencia.

Los códigos exponencial-Golomb se usan como el tercer tipo. Esto es, los terceros símbolos s_3 (véase la Fig. 2b) pueden codificarse a través de dichos códigos VLC en el codificador/decodificador VLC 102/202. El subdivisor 100 y el recombinador 220 actúan en consecuencia. Son posibles diferentes códigos VLC y diferentes combinaciones de códigos VLC y PIPE o VLC y aritméticos.

Mientras que el primer código es más complejo pero conduce a un mejor rendimiento de compresión, el segundo código entrópico representa un compromiso razonable entre complejidad y rendimiento. Los últimos códigos de entropía, por ejemplo códigos Exponencial-Golomb, tienen una complejidad muy baja. En lo que sigue, se describe la codificación de diferentes particiones.

En una partición tal como la partición 140_1 , tal como el símbolo s_1 , ha de ser codificado por entropía usando un codificador entrópico que emplee modelos de probabilidad como el codificador PIPE 104 (de nuevo, puede usarse CABAC o cualquier codificador aritmético en un alternativa no descrita adicionalmente en el presente documento), el subdivisor 120 dirige la misma hacia el codificador PIPE 104. En primer lugar, los niveles de coeficiente de transformada absoluta evaluados no binarios pueden binarizarse en el simbolizador 122 usando un método de binarización. La binarización mapea los niveles de coeficiente de transformada absoluta evaluados no binarios en una secuencia de bins binarios. Cada bin de la cadena de bins se codifica con un contexto elegido por un asignador 114. El modelizado de contexto puede realizarse por el primer bin y fijarse para los bins siguientes de la secuencia de bin dado que puede usarse coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC o un modelizado de contexto diferente para cada bin de la cadena de bins. Obsérvese que la binarización puede ser un código de longitud variable como los códigos Golomb o códigos exponencial-Golomb u otros códigos de longitud variable.

A continuación, una partición tal como la partición 140_2 o símbolos s_2 puede codificarse con un código Golomb, en este caso en el codificador VLC 102 y decodificarse respectivamente en el decodificador VLC. Los códigos Golomb son un conjunto de códigos de entropía diseñados para una fuente distribuida geométricamente. Si el orden del código Golomb es cero, el código Golomb es conocido también como código unario. El código unario se refiere a la binarización de coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC. Los códigos Golomb se construyen como sigue. Para un parámetro Golomb específico k , el valor n se divide por el parámetro Golomb k usando división entera y se calcula el resto r .

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

$$r = n - pk$$

Después de deducir los parámetros especificados por las fórmulas anteriores, puede codificarse el valor n con dos partes. La primera parte, también denominada como parte prefijo, es un código unario. El valor resultante $p+1$ especifica el número de unos y un cero de finalización o viceversa. El valor de resto, también denominado como parte resto e indicada por r , se representa con un código binario truncado. Los códigos Golomb-Rice se emplean para codificación de símbolos fuente tales como los símbolos fuente s_2 siendo los códigos Golomb-Rice un subconjunto de los códigos Golomb. También, cuando se usan dichos códigos de entropía para particiones 140_{1-3} que contienen una delimitación, tal como la partición 140_2 , el alfabeto de los símbolos fuente respectivos (tal como los símbolos fuente s_2) está limitado y el código Golomb-Rice puede modificarse de modo que pueda mejorarse la eficiencia de codificación. El parámetro del código Golomb-Rice puede ser fijo o variable. Si el parámetro es variable, el parámetro puede estimarse como una parte de la etapa de modelizado de contexto. Por ejemplo, si un símbolo fuente s_2 entra en el decodificador VLC 102, este último puede determinar el parámetro del código Golomb-Rice a partir de un contexto de s_2 . Los códigos Golomb-Rice son códigos Golomb con parámetros a la potencia de dos. De modo que se basan en división y multiplicación por dos y por lo tanto pueden implementarse eficientemente en una arquitectura binaria con operaciones de desplazamiento y adición. La relación entre el parámetro Golomb-Rice y el parámetro Golomb es por lo tanto $K_{GOLOMB} = 2^{KRICE}$. En caso del código Golomb-Rice, la parte de resto es exactamente la representación binaria del valor del resto. Para el parámetro 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 bin con dos símbolos de entrada que comparten el mismo prefijo unario. En lo que sigue, se ilustran algunas tablas de ejemplo para parámetros 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 caso de la partición 140_2 y el parámetro del código Golomb-Rice, el truncado puede realizarse como sigue. El parámetro Golomb-Rice describe el número de bins requeridos para representar la parte de resto y el dos a la potencia del valor de parámetro describe el número de valores, que puede representarse con el mismo prefijo. Estos valores forman un grupo de prefijo. Por ejemplo, para el parámetro cero, un prefijo solo 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 fuente limitado y un código Golomb-Rice dado, el último bin del prefijo puede dejarse fuera de los valores en el último grupo de prefijo dando como resultado un código de prefijo con longitud fija.

En el caso de que el rango sea nueve y el parámetro Golomb-Rice sea dos, el número de valores que pueden representarse con el mismo prefijo es cuatro. El valor máximo es nueve, lo que también indica que el borde se ha superado y ha de usarse el siguiente código de entropía de la siguiente partición. En este caso de ejemplo, los valores desde 0 - 3 tienen el prefijo 0, los valores desde 4 - 7 tienen el prefijo 10 y 8 - 9 el prefijo 110. Debido a que los valores 8 - 9 forman el último grupo de prefijo, su cero de la izquierda puede quitarse y los valores 8 - 9 pueden representarse por 11. En otras palabras, se puede imaginar un símbolo fuente s_2 introducido en el codificador VLC 102 siendo el número de valores posibles de s_2 de 9 (=límite2-límite1) y siendo el parámetro Golomb-Rice para ese símbolo fuente dos. A continuación, se produciría una palabra de código Golomb-Rice respectiva por el codificador VLC 102 para ese símbolo fuente, que tiene un prefijo según se acaba de describir. Para la parte de resto de la palabra de código, el código truncado puede deducirse como sigue por el codificador VLC 102. Normalmente, el

parámetro de los códigos Golomb-Rice indica el número de bins de la parte de resto. En un caso truncado, no todos los bins del resto necesitan codificarse. Para un caso truncado, se cuentan todos los valores con prefijo fijo (por ejemplo un bin del prefijo se extrae). 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 se trunca. Si es posible un truncado de la parte de resto, la deducción de la parte de resto truncada para el último grupo de prefijo puede proseguir en las siguientes etapas. Primero, se deduce el mayor número l a la potencia de dos más pequeño o igual que el número encontrado. A continuación, en la segunda etapa, se deduce el número más pequeño h a la potencia de dos mayor que el número encontrado. El primer valor l escribe el número de valores en el grupo de prefijo con un resto de h bins. 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 del resto. Para los valores restantes del último grupo de prefijo, ahora tratados como un nuevo grupo de prefijo, se realiza el mismo procedimiento excepto para los valores resultantes formados por el primer grupo de resto, el resto comienza con un 1. Este procedimiento se realiza hasta que se deducen todos los restos. Como un ejemplo, el rango es 14 y el parámetro es tres. El primer grupo de prefijo contiene valores desde 0 - 7 y el segundo grupo de prefijo valores desde 8 - 13. El segundo grupo de prefijo contiene seis valores. Los parámetros son $l = 2$ y $h = 3$. De ese modo, los primeros cuatro valores de los grupos de prefijo se representan por un resto con tres bins (un cero a la izquierda y la representación binaria para distinguir los cuatro valores). Para los últimos dos valores, se realiza de nuevo el mismo procedimiento. Los parámetros son $l = 1$ y $h = 2$. El resto de los dos últimos valores puede representarse ahora como 10 y 11. Otro ejemplo para demostrar el método es un parámetro de Golomb-Rice de cuatro y un rango de diez. Para este ejemplo, los parámetros son $l = 3$ y $h = 4$. Con estos parámetros, la parte de resto truncada para los primeros ocho valores se representa mediante cuatro bins. Los dos valores restantes tienen la misma parte de resto que en el ejemplo anterior. Si el rango es nueve para el ejemplo previo, el parámetro para la segunda tirada es $l = 0$ y $h = 1$. La parte de resto del único valor que permanece es 1.

El tercer tipo de códigos de entropía pueden ser códigos Exponencial-Golomb. Pueden emplearse para distribuciones igualmente probables (por ejemplo con parámetro cero) tales como los símbolos fuente s_3 . Esto es, el par de codificador/decodificador VLC puede ser responsable de su codificación. Como se ha mencionado anteriormente, los niveles de coeficiente de transformada absoluta mayores frecuentemente se distribuyen uniformemente. Más precisamente, el código Exponencial-Golomb de orden cero puede usarse para codificar la última partición 140_3 . El comienzo y por lo tanto el borde 144 de la partición previa 140_2 puede ser variable. La posición del borde 144 puede controlarse por el codificador/decodificador VLC 102-200 dependiendo de los símbolos fuente codificados/decodificados 106, 108 precisamente y/o 110 o los elementos de sintaxis 138 (o 218, 204 y/o 208 o elementos de sintaxis 226).

En una realización preferida, el número de particiones es tres como se muestra en la Fig. 1b y las delimitaciones 142 y 144 pueden ser variables. Para la primera partición 140_1 , puede emplearse codificación PIPE como se ha explicado anteriormente. Sin embargo, puede usarse alternativamente asimismo CABAC. En ese caso, el par codificador/decodificador PIPE se sustituiría por un par codificador/decodificador de codificación aritmética. Los códigos Golomb-Rice truncados pueden usarse para la segunda partición 140_2 y el código Exponencial-Golomb de orden cero puede usarse para la última partición 140_3 .

En otra realización preferida, el número de particiones 140_{1-3} es tres y la primera delimitación 142 es fija, mientras que la segunda delimitación 144 es variable. Para la primera partición 140_1 , se emplea CABAC o PIPE. Los códigos Golomb-Rice truncados pueden usarse para la segunda partición 140_2 , y el código Exponencial-Golomb de orden cero puede usarse para la última partición 140_3 .

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

La delimitación 142 de la primera partición 140_1 que usa CABAC o PIPE, que emplea modelos de probabilidad adaptativos puede ser dos. El modelizado de contexto para el primer bin puede realizarse como se ha descrito para coeficiente_ absoluto_mayor_uno tal como se ha descrito anteriormente y el modelizado de contexto para el segundo bin puede realizarse como se ha descrito para coeficiente_ absoluto_nivel_menos_dos en H.264/AVC, como también se ha descrito anteriormente. La última determinación de contexto se determinaría por los asignadores 114 y 212, respectivamente. La delimitación 142 de la primera partición 140_1 que usa codificación de entropía que emplea modelizado de probabilidad (por ejemplo PIPE o CABAC) puede ser dos y el modelizado de contexto para tanto el primero como el segundo bin puede realizarse como se ha descrito para coeficiente_ absoluto_mayor_uno en H.264/AVC como se ha descrito anteriormente. La valoración del conjunto de contexto como se ha descrito para coeficiente_ absoluto_mayor_uno puede realizarse por separado para el segundo bin.

La delimitación 142 para la primera partición 140_1 que usa codificación entrópica que emplea modelizado de probabilidad (por ejemplo CABAC o PIPE) puede ser igual a uno. Para el único bin (o símbolo de alfabeto) de la cadena de bins del símbolo fuente respectivo, el modelizado de contexto puede realizarse como se ha descrito para

coeficiente_ absoluto_ mayor_ uno en H.264/AVC previamente descrito.

5 La delimitación 142 de la primera partición 140₁ que usa codificación entrópica que emplea modelizado de probabilidad (por ejemplo CABAC o PIPE) puede ser igual a tres. El modelizado de contexto del primer y segundo bins de la cadena de bins del símbolo fuente respectivo puede realizarse como coeficiente_ absoluto_ mayor_ uno en H.264/AVC. El modelizado de contexto para el tercer bin puede realizarse como coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC. La evaluación del conjunto de contexto como se ha descrito para coeficiente_ absoluto_ mayor_ uno puede realizarse por separado para el segundo bin.

10 Un conjunto de códigos Golomb-Rice truncados puede usarse como códigos de entropía de la segunda partición 140₂. La delimitación 144 de la segunda partición que especifica el comienzo de la tercera partición 140₃ que depende del parámetro del código de entropía puede ser variable. También, el parámetro Golomb-Rice puede limitarse por tres y la selección del parámetro puede realizarse como el modelizado de contexto para coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC. El rango límite-límite2 puede ser variable y puede depender del parámetro Golomb-Rice. Si el parámetro es cero, el rango es 8. Para el parámetro uno, el rango es 10. En caso de parámetro dos, el rango es 12 y para el parámetro tres, el rango es igual a 16. En este ejemplo, el parámetro Golomb-Rice puede fijarse a cero al comienzo de un bloque de coeficientes de transformada. Para cada nivel de coeficiente de transformada de código en el bloque mayor o igual que la primera delimitación, se usa el correspondiente código Golomb-Rice. Después de la codificación (o decodificación) del nivel, se realiza la siguiente evaluación para actualizar el parámetro Golomb-Rice para codificación (o decodificación) del siguiente nivel mayor que o igual a la primera delimitación. Obsérvese que el parámetro Golomb-Rice no puede disminuirse mediante el uso de esta forma de adaptación.

25 La regla de adaptación del parámetro puede resumirse como sigue, en donde k_{i+1} indica el parámetro de Golomb-Rice a ser usado para codificación del siguiente valor de nivel y $valor_i$ indica el valor previamente codificado con el parámetro Golomb-Rice k_i correspondiente

$$k_{i+1} = \begin{cases} 0 & valor_i \in [0, 1] \wedge k_i < 1 \\ 1 & valor_i \in [2, 3] \wedge k_i < 2 \\ 2 & valor_i \in [4, 5] \wedge k_i < 3 \\ 3 & valor_i > 5 \wedge k_i < 4 \\ k_i & \text{en caso contrario} \end{cases} \quad (QQ)$$

30 En una realización preferida, puede usarse un conjunto de códigos Golomb-Rice truncados como códigos de entropía de una segunda partición 140₂. La delimitación 144 de la segunda partición 140₂ que especifica el comienzo de la tercera partición 140₃ dependiendo del parámetro del código de entropía puede ser variable. También en esta realización preferida, el parámetro Golomb-Rice puede estar limitado a tres y la selección del parámetro puede realizarse como modelizado de contexto para coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC. El rango puede ser variable y depende del parámetro Golomb-Rice. Si el parámetro es cero, el rango es 8. Para el parámetro uno, el rango es 10. En caso de parámetro dos, el rango es 12 y para el parámetro tres, el rango es igual a 16. En esta realización preferida, el parámetro Golomb-Rice se fija en cero al comienzo del bloque. La adaptación del parámetro Golomb-Rice se realiza como se describe por la ecuación (QQ). Obsérvese que el parámetro no puede disminuirse mediante el uso de esta forma de adaptación.

40 En otra realización preferida, puede usarse un conjunto de códigos Golomb-Rice truncado como códigos de entropía de la segunda partición 140₂. La delimitación 144 de la segunda partición 140₂ que especifica el comienzo de la tercera partición 140₃ dependiendo del parámetro del código de entropía puede ser fija. También en esta realización preferida, el parámetro Golomb-Rice puede estar limitado a tres y la selección del parámetro puede realizarse como modelizado de contexto para coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC. El rango de la segunda partición 140₂ puede fijarse en 14. En esta realización preferida, el parámetro Golomb-Rice puede fijarse a cero al comienzo del bloque. La adaptación del parámetro Golomb-Rice se realiza como se describe por la ecuación (QQ). Obsérvese que el parámetro no puede disminuirse mediante el uso de esta forma de adaptación.

50 En otra realización preferida, puede usarse un conjunto de códigos Golomb-Rice truncados como códigos de entropía de una segunda partición 140₂. La delimitación 144 de la segunda partición 140₂ que especifica el comienzo de la tercera partición 140₃ dependiendo del parámetro del código de entropía puede ser variable. También en esta realización preferida, el parámetro Golomb-Rice puede estar limitado a tres y la selección del parámetro puede realizarse como modelizado de contexto para coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC. El rango puede ser variable y depende del parámetro Golomb-Rice. Si el parámetro puede ser cero, el rango puede ser 8. Para el parámetro uno, el rango puede ser 10. En caso de parámetro dos el rango puede ser 12 y para el parámetro tres, el rango puede ser igual a 16. En esta realización preferida, el parámetro Golomb-Rice puede fijarse a cero al comienzo del bloque. La adaptación del parámetro Golomb-Rice se realiza como se describe por la ecuación (QQ).

Obsérvese que el parámetro no puede disminuirse mediante el uso de esta forma de adaptación. Y tómesese también nota que es posible una conmutación directa, por ejemplo desde cero a tres. En esta realización preferida, la parte de prefijo de los códigos Golomb-Rice se codifica con códigos de entropía que emplean modelos de probabilidad. El modelizado de contexto puede realizarse como para coeficiente_ absoluto_ nivel_ menos_ dos en H.264/AVC.

5 En otra realización preferida, se evalúa si la vecindad ya codificada (o decodificada) del índice de escaneado actual contiene niveles de coeficiente de transformada absoluta mayores que la delimitación previa. Para esta realización preferida, el mejor parámetro puede deducirse mediante el uso de vecinos en una plantilla causal local.

10 De ese modo, las realizaciones anteriormente mencionadas describen entre otros, un aparato de codificación entrópica que comprende un descomponedor 136 configurado para convertir una secuencia 138 de elementos de sintaxis en una secuencia 106 de símbolos fuente 106 mediante la descomposición individualmente de al menos un subgrupo de los elementos de sintaxis en un número respectivo n de símbolos fuente s_i , con $i = 1, \dots, n$, dependiendo el número respectivo n de símbolos fuente de cuál de entre una secuencia de n particiones 140_{1-3} dentro de las que se subdivide un rango de valores de los elementos de sintaxis respectivos, cae un valor z de los elementos de sintaxis respectivos, de modo que una suma de valores del número respectivo de símbolos fuente s_i conduce a z y, si $n > 1$, para todo $i = 1, \dots, n-1$, el valor de s_i corresponde a un rango de la partición i^a ; un divisor 100 configurado para subdividir la secuencia 106 de símbolos fuente en una primera subsecuencia 108 de símbolos fuente y una segunda subsecuencia 110 de símbolos fuente de modo que todos los símbolos fuente s_x siendo x miembro de un primer subconjunto de $\{1 \dots n\}$ están contenidos dentro de la primera subsecuencia 108 y todos los símbolos fuente s_y siendo y un miembro del segundo subconjunto de $\{1 \dots n\}$ que es disjuncto respecto al primer subconjunto, están contenidos dentro de la segunda subsecuencia 110; un codificador VLC 102 configurado para codificar acertadamente con relación a símbolos los símbolos fuente de la primera subsecuencia 108, y un codificador PIPE o aritmético 104 configurado para codificar la segunda subsecuencia 110 de símbolos fuente.

25 Los valores z del subgrupo de los elementos de sintaxis pueden ser valores absolutos. El segundo subconjunto puede ser $\{1\}$ estando dispuesta la secuencia de n particiones de modo que una partición p^a cubre valores más altos del rango de valores que una partición q^a para todo $p, q \in \{1 \dots n\}$ siendo $p > q$. n puede ser 3. El primer subconjunto puede ser $\{2, 3\}$ configurado el codificador VLC (102) para usar un código Golomb-Rice para codificar acertadamente con relación a símbolos los símbolos fuente s_2 , y un código Exp-Golomb para codificar acertadamente con relación a símbolos los símbolos fuente s_3 . Más generalmente, 2 pueden ser elementos del primer subconjunto con un codificador VLC (102) configurado para usar un código Golomb-Rice para codificar acertadamente con relación a símbolos los símbolos fuente s_2 y adaptar un parámetro Golomb-Rice, es decir k , del código Golomb-Rice de acuerdo con los símbolos fuente previamente codificados. El descomponedor puede configurarse para adaptar uno o más de los límites entre las particiones de acuerdo con los símbolos fuente previamente codificados. Ambas adaptaciones pueden combinarse. Esto es, las posiciones de los límites que limitan la segunda partición pueden adaptarse de modo que estén separados entre sí de modo que la longitud del código Golomb-Rice, es decir el número de palabras de código del mismo, corresponda a (o prescriba) la longitud del ancho de la segunda partición. El límite para la separación de la primera de la segunda partición puede definirse en otra forma tal como ser fijo por definición o adaptado de acuerdo con otra dependencia de contexto, en cuyo caso la adaptación de k puede definir la posición de la separación límite de la segunda y tercera partición a través de la longitud del código de Golomb-Rice y el ancho de la segunda partición, respectivamente. Al acoplar la adaptación de k de modo que el ancho de la segunda partición corresponda a la longitud del código Golomb-Rice, la eficiencia del código se usa de modo óptimo. Adaptar k a las estadísticas del elemento de sintaxis permite adaptar el ancho de la segunda partición de modo que la tercera partición pueda cubrir tanto como sea posible de modo que reduzca la complejidad de codificación global dado que puede haberse usado un código menos complejo para la tercera partición tal como el código Exp-Golomb. Adicionalmente, la longitud de la primera partición puede restringirse a $j \in \{1, 2, 3\}$ posibles valores de elementos de sintaxis, tal como los tres niveles más bajos. Los elementos de sintaxis bajo consideración pueden codificarse diferencialmente o representar una predicción residual tal como es el caso con los niveles de coeficiente de transformada ejemplificados anteriormente que representan una predicción residual. Los primeros símbolos fuente s_1 pueden simbolizarse/desimbolizarse mediante el uso de un código unario truncado siendo codificados los j bins resultantes —parcialmente o todos ellos— adaptativamente al contexto o no como se ha mencionado anteriormente.

55 El subgrupo de los elementos de sintaxis puede englobar niveles de coeficiente de transformada absoluta de los coeficientes de transformada absoluta de bloques de transformada de la imagen con los niveles de coeficiente de transformada absoluta de un bloque de transformada respectivo que se dispone dentro de la secuencia (138) de elementos de sintaxis de acuerdo con una trayectoria de barrido que conduce a través de los coeficientes de transformada absoluta de los bloques de transformada respectivos, en el que el descomponedor puede configurarse para adaptar uno o más límites entre las particiones durante la descomposición de los niveles de coeficiente de transformada absoluta de los coeficientes de transformada absoluta de un bloque de transformada respectivo dependiendo de los niveles de coeficiente de transformada absoluta ya codificados de los coeficientes de transformada absoluta de los bloques de transformada respectivos que preceden en el orden de escaneado o dependen de una posición del nivel de coeficiente de transformada absoluta actualmente a ser descompuesto en el

orden de escaneado, o basado en una evaluación de los niveles de coeficiente de transformada absoluta a ser reconstruidos de coeficientes de transformada vecinos —tanto espacialmente como en orden de escaneado— la posición del nivel del coeficiente de transformada absoluta a ser descompuesta actualmente.

5 Adicionalmente, las realizaciones mencionadas anteriormente describen entre otros un aparato de decodificación entrópica que comprende un decodificador VLC 200 configurado para reconstruir acertadamente con relación a las palabras de código símbolos fuente de una primera subsecuencia 204 de símbolos fuente a partir de palabras de código de un primer flujo de bits 206; un decodificador PIPE o aritmético 202 configurado para reconstruir una
 10 segunda subsecuencia 208 de símbolos fuente; un componedor 224 configurado para componer una secuencia 226 de elementos de sintaxis a partir de la primera subsecuencia 204 de símbolos fuente y la segunda subsecuencia 208 de símbolos fuente mediante la composición individualmente de cada elemento de sintaxis a partir de un número respectivo de símbolos fuente, en el que el componedor se configura para, en al menos un subgrupo de los elementos de sintaxis, determinar el número respectivo n de símbolos fuente s_i con $i = 1 \dots n$ dependiendo de en cuál de una secuencia de n particiones 140₁₋₃ dentro de la que se subdivide un rango de valores de los elementos de sintaxis respectivo, cae dentro un valor z de los elementos de sintaxis respectivos, mediante la suma de los valores del número respectivo de símbolos fuente s_i desde 1 a n siempre que el valor de s_i corresponda a un rango en la partición i -ésima de modo que obtenga el valor del elemento de sintaxis z , en el que el componedor 224 se configura para recuperar todos los símbolos fuente s_x siendo x miembro de un primer subconjunto de $\{1 \dots n\}$ de la primera subsecuencia (204) y todos los símbolos fuente s_y siendo y miembro de un segundo subconjunto de $\{1 \dots n\}$ que es
 20 disjuncto 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 $\{1\}$ estando dispuesta la secuencia de n particiones de modo que una partición p^a cubre valores más altos del rango de valores que una partición q^a para todo $p, q \in \{1 \dots n\}$ siendo $p > q$. n puede ser 3. El primer subconjunto puede ser $\{2, 3\}$ estando configurado el decodificador VLC 200 para usar un código Golomb-Rice para reconstruir acertadamente con relación a palabras de código los símbolos fuente s_2 , y un código Exp-Golomb para reconstruir acertadamente con relación a palabras de código los símbolos fuente s_3 . Más generalmente, 2 pueden ser elemento del primer subconjunto con el decodificador VLC 102 configurado para usar un código Golomb-Rice para reconstruir acertadamente con relación a palabras de código los símbolos fuente s_2 y adaptar un parámetro Golomb-Rice del código Golomb-Rice de acuerdo con los símbolos fuente previamente reconstruidos. El aparato de decodificación entrópica puede comprender
 30 adicionalmente un recombinador 220 configurado para recombinar la primera subsecuencia 204 de símbolos fuente y la segunda subsecuencia de símbolos fuente para obtener la secuencia 218 de símbolos fuente. Los elementos de sintaxis pueden ser de diferente tipo y el componedor puede configurarse para realizar la composición individual dependiendo del tipo de elementos de sintaxis. El subgrupo de los elementos de sintaxis pueden englobar niveles de coeficiente de transformada absoluta de coeficientes de transformada absoluta de los bloques de transformada de una imagen con los niveles de coeficiente de transformada absoluta de un bloque de transformada respectivo que se dispone dentro de la secuencia 138 de elementos de sintaxis de acuerdo con una trayectoria de escaneado que conduce a través de los coeficientes de transformada absoluta de los bloques de transformada respectivos, en el que el componedor puede configurarse para adaptar uno o más de los límites entre las particiones durante la composición de los niveles de coeficiente de transformada absoluta de los coeficientes de transformada absoluta de un bloque de transformada respectivo dependiendo de niveles de coeficiente de transformada absoluta ya reconstruidos de coeficientes de transformada absoluta de los bloques de transformada respectivos que preceden en el orden de escaneado o dependen de una posición del nivel de coeficiente de transformada absoluta actualmente a ser compuesto en el orden de escaneado, o basándose en una evaluación de niveles de coeficientes de transformada absoluta ya reconstruidos de los coeficientes de transformada vecinos —tanto espacialmente como
 45 en orden de escaneado— la posición del nivel de coeficiente de transformada absoluta a ser compuesto actualmente.

En relación a la combinación de codificación PIPE con codificación VLC usando la descomposición de acuerdo con la Fig. 1b, se observa lo siguiente para repetir algunos aspectos de los mismos en otras palabras.

50 Se ha descrito el mapeado de una secuencia de símbolos en un flujo de bits y el mapeado inverso. Cada símbolo lleva un(os) parámetro(s) asociado(s) con él que son conocidos simultáneamente en codificador y decodificador. El códec de entropía contiene múltiples memorias intermedias primero en entrar primero en salir (FIFO) cada una de ellas asignada a subconjuntos de parámetro(s) que se asocian a los símbolos. Para parámetro(s) dado(s) de un símbolo, el codificador asigna el símbolo a la memoria intermedia FIFO correspondiente. Dado que la regla de asignación del codificador es conocida en el lado del decodificador, el decodificador lee de la memoria intermedia FIFO a la que el codificador ha asignado al símbolo.

60 Algunos elementos de sintaxis se codifican usando códigos de longitud variable estándar y se escriben en una memoria intermedia particular. Otros elementos de sintaxis se codifican usando el concepto de codificación de entropía de probabilidad del intervalo de partición (PIPE). En esto, los símbolos se binarizan primero y los bins resultantes se clasifican basándose en estimaciones de probabilidad asociadas. La estimación de probabilidad puede darse o deducirse a partir de la medición que puede realizarse simultáneamente en codificador y decodificador. Una memoria intermedia FIFO particular contiene símbolos con valores de probabilidad estimados

que caen dentro del subconjunto de probabilidades que se eligen, de modo que pueda mejorarse la codificación entrópica. La mejora conseguida mediante la combinación del concepto PIPE con VLC es una reducción de complejidad en tanto aún se proporciona una alta eficiencia de codificación. Los símbolos para los es adecuado un código VLC estándar se codifican con un enfoque VLC simple y baja complejidad, mientras otros símbolos para los que la tasa de bits se incrementaría significativamente mediante su codificación con un código VLC se codifican con el concepto PIPE más sofisticado.

Así, para reducir adicionalmente la complejidad de la codificación entrópica, los símbolos se han repartido en dos categorías. Símbolos de una primera categoría pueden representarse bien con códigos VLC y no requieren la codificación PIPE más compleja, mientras que los símbolos de una segunda categoría no pueden representarse eficientemente con códigos VLC y una codificación PIPE para estos símbolos reduce significativamente la tasa de bits requerida.

Aunque algunos aspectos se han descrito en el contexto de un aparato, es claro que estos aspectos también representan una descripción de un método correspondiente, en donde un bloque o dispositivo corresponde a una etapa del método o una característica de una etapa del método. Análogamente, aspectos descritos en el contexto de una etapa del método también representan una descripción de un bloque o apartado o característica correspondiente de un aparato correspondiente. Algunas o todas de las etapas del método pueden ejecutarse mediante (o usando) un aparato de hardware, como por ejemplo un microprocesador, un ordenador programable o un circuito electrónico. En algunas realizaciones, algunas o más de las etapas del método más importantes pueden ejecutarse mediante un aparato de ese tipo.

Las señales codificadas/comprimidas inventivas pueden almacenarse en un medio de almacenamiento digital o pueden transmitirse sobre un medio de transmisión tal como un medio de transmisión inalámbrico o un medio de transmisión por cable tal como Internet.

Dependiendo de ciertos requisitos de implementación, las realizaciones de la invención pueden implementarse en hardware o software. La implementación puede realizarse usando un medio de almacenamiento digital, por ejemplo un disco flexible, un DVD, un Blue-Ray, un CD, un ROM, un PROM, un EPROM, un EEPROM o una memoria FLASH, que tenga señales de control legibles electrónicamente almacenadas en él, que coopere (o sea capaz de cooperar) con un sistema informático programable de modo que se realice 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 portador de datos que tiene señales de control legibles electrónicamente, que es capaz de cooperar con un sistema informático programable, de modo se realice uno de los métodos descritos en el presente documento.

Generalmente, realizaciones de la presente invención pueden implementarse como un producto de programa informático con un código de programa, siendo operativo el código de programa para la realización de uno de los métodos cuando el producto de programa informático se ejecuta en un ordenador. El código de programa puede almacenarse por ejemplo sobre un portador legible por máquina.

Otras realizaciones comprenden el programa informático para la realización de uno de los métodos descritos en el presente documento, almacenado en un portador legible por máquina.

En otras palabras, una realización del método inventivo es, por lo tanto, un programa informático que tiene un código de programa para la realización de uno de los métodos descritos en el presente documento, cuando el programa informático se ejecuta en un ordenador.

Una realización adicional de los métodos inventivos es, por lo tanto, un portador de datos (o un medio de almacenamiento digital, o un medio legible por ordenador) que comprende, registrado en él, el programa informático para la realización de uno de los métodos descritos en el presente documento.

Una realización adicional del método inventivo es, por lo tanto, un flujo de datos o una secuencia de señales que representan el programa informático para la realización de uno de los métodos descritos en el presente documento. El flujo de datos o la secuencia de señales pueden configurarse por ejemplo para transferirse a través de una conexión de comunicación de datos, por ejemplo a través de la Internet.

Una realización adicional comprende un medio de procesamiento, por ejemplo un ordenador, o un dispositivo lógico programable, configurado para o adaptado para realizar uno de los métodos descritos en el presente documento.

Una realización adicional comprende un ordenador que tiene instalado en él el programa informático para la realización de uno de los métodos descritos en el presente documento.

En algunas realizaciones, puede usarse un dispositivo lógico programable (por ejemplo una matriz de puertas

programable en campo) para realizar algunas o todas las funcionalidades de los métodos descritos en el presente documento. En algunas realizaciones, una matriz de puertas programable en campo puede cooperar con un microprocesador para realizar uno de los métodos descritos en el presente documento. Generalmente, los métodos se realizan preferentemente mediante un aparato de hardware.

5 Las realizaciones descritas anteriormente son meramente ilustrativas de los principios de la presente invención. Se entiende que serán evidentes para los expertos en la materia modificaciones y variaciones de las disposiciones y los detalles descritos en el presente documento. Es la intención, por lo tanto, limitarse solamente por el alcance de las reivindicaciones de patente inminentes y no por los detalles específicos presentados a modo de descripción y
10 explicación de las realizaciones del presente documento.

REIVINDICACIONES

1. Aparato de codificación entrópica que comprende

5 un descomponedor (136) configurado para convertir una secuencia (138) de elementos de sintaxis que tienen un rango de valores que se subdivide en una secuencia de N particiones (140₁₋₃) en una secuencia (106) de símbolos fuente (106) mediante la descomposición individualmente de al menos un subgrupo de los elementos de sintaxis en un número respectivo n de símbolos fuente s_i con $i = 1 \dots n$, dependiendo del número respectivo n de símbolos fuente de en cuál de las secuencias de N particiones (140₁₋₃) cae un valor z de los elementos de sintaxis respectivos, de modo que una suma de valores del número respectivo de símbolos fuente s_i conduzca a z y, si $n > 1$, para todo $i = 1 \dots n-1$, el valor de s_i corresponda a un rango de la partición i-ésima;

10 un subdivisor (100) configurado para subdividir la secuencia (106) de símbolos fuente en una primera subsecuencia (108) de símbolos fuente y una segunda subsecuencia (110) de símbolos fuente de modo que todos los símbolos fuente s_x , siendo x un miembro de un primer subconjunto de $\{1 \dots N\}$, estén contenidos dentro de la primera subsecuencia (108) y todos los símbolos fuente s_y , siendo y un miembro de un segundo subconjunto de $\{1 \dots N\}$ que es disyunto respecto al primer subconjunto, estén contenidos dentro de la segunda subsecuencia (110);

15 un codificador VLC (102) configurado para codificar acertadamente con relación a símbolos los símbolos fuente de la primera subsecuencia (108); y

un codificador aritmético (104) configurado para codificar la segunda subsecuencia (110) de símbolos fuente, en el que el segundo subconjunto es $\{1\}$ estando dispuesta la secuencia de N particiones de modo que una p-ésima partición cubra valores más altos del rango de valores que una q-ésima partición para todo $p, q \in \{1 \dots N\}$ con $p > q$,

caracterizado porque

$N = 3$, y

25 en el que el primer subconjunto es $\{2, 3\}$ con el codificador VLC (102) configurado para usar un código Golomb-Rice para codificar acertadamente con relación a símbolos los símbolos fuente s_2 , y un código Exp-Golomb para codificar acertadamente con relación a símbolos los símbolos fuente s_3 ,

en el que los valores z del subgrupo de los elementos de sintaxis son valores absolutos, y

en el que el codificador VLC (102) se configura para adaptar un parámetro Golomb-Rice del código Golomb-Rice de acuerdo con símbolos previamente codificados.

30 2. Aparato de decodificación entrópica que comprende

un decodificador VLC (200) configurado para reconstruir acertadamente con relación a palabras de código símbolos fuente de una primera subsecuencia (204) de símbolos fuente a partir de palabras de código de un primer flujo de bits (206);

35 un decodificador aritmético (202) configurado para reconstruir una segunda subsecuencia (208) de símbolos fuente;

un componedor (224) configurado para componer una secuencia (226) de elementos de sintaxis que tienen un rango de valores que se subdivide en una secuencia de N particiones (140₁₋₃) a partir de la primera subsecuencia (204) de símbolos fuente y la segunda subsecuencia (208) de símbolos fuente mediante la composición individualmente de cada elemento de sintaxis a partir de un número respectivo de símbolos fuente mediante, para al menos un subgrupo de elementos de sintaxis, la determinación del número respectivo n de símbolos fuente s_i con $i = 1 \dots n$ dependiendo de en cuál de la secuencia de N particiones (140₁₋₃) en la que se subdivide un rango de valores de los elementos de sintaxis respectivo, cae dentro un valor z de los elementos de sintaxis respectivos, mediante la suma de los valores del número respectivo de símbolos fuente s_i desde 1 a n siempre que el valor de s_i corresponde a un rango de la partición i-ésima de modo que obtenga el valor del elemento de sintaxis z, en el que el componedor (224) se configura para recuperar todos los símbolos fuente s_x siendo x un miembro de un primer subconjunto de $\{1 \dots N\}$ a partir de la primera subsecuencia (204) y todos los símbolos fuente s_y siendo y un miembro de un segundo subconjunto de $\{1 \dots N\}$ que es disyunto respecto al primer subconjunto, a partir de la segunda subsecuencia (208);

45 en el que el segundo subconjunto es $\{1\}$ estando dispuesta la secuencia de N particiones de modo que una p-ésima partición cubra valores más altos del rango de valores que una q-ésima partición para todo $p, q \in \{1 \dots N\}$ con $p > q$,

caracterizado porque

$N = 3$,

50 en el que el primer subconjunto es $\{2, 3\}$ con el decodificador VLC (200) configurado para usar un código Golomb-Rice para reconstruir acertadamente con relación a las palabras de código los símbolos fuente s_2 , y un código Exp-Golomb para reconstruir acertadamente con relación a las palabras de código los símbolos fuente s_3 ,

55 en el que los valores z del subgrupo de los elementos de sintaxis son valores absolutos, y

en el que el decodificador VLC (102) se configura para adaptar un parámetro Golomb-Rice del código Golomb-Rice de acuerdo con símbolos fuente previamente reconstruidos.

60 3. Aparato de decodificación entrópica de acuerdo con la reivindicación 2 que comprende adicionalmente un recombinador (220) configurado para recombinar la primera subsecuencia (204) de símbolos fuente y la segunda subsecuencia de símbolos fuente para obtener la secuencia (218) de símbolos fuente.

4. Método de codificación entrópica que comprende

convertir una secuencia (138) de elementos de sintaxis que tienen un rango de valores que se subdivide en una secuencia de N particiones (140₁₋₃) en una secuencia (106) de símbolos fuente (106) mediante la descomposición individualmente de al menos un subgrupo de los elementos de sintaxis en un número respectivo n de símbolos fuente s_i con $i = 1 \dots n$, dependiendo el número respectivo n de símbolos fuente de en cuál de las secuencias de N particiones (140₁₋₃) cae un valor z de los elementos de sintaxis respectivos, de modo que una suma de valores del número respectivo de símbolos fuente s_i conduzca a z y, si $n > 1$, para todo $i = 1 \dots n-1$, el valor de s_i corresponda a un rango de la partición i-ésima; subdividir la secuencia (106) de símbolos fuente en una primera subsecuencia (108) de símbolos fuente y una segunda subsecuencia (110) de símbolos fuente de modo que todos los símbolos fuente s_x , siendo x un miembro de un primer subconjunto de $\{1 \dots N\}$, estén contenidos dentro de la primera subsecuencia (108) y todos los símbolos fuente s_y , siendo y un miembro de un segundo subconjunto de $\{1 \dots N\}$ que es disyunto respecto al primer subconjunto, estén contenidos dentro de la segunda subsecuencia (110); mediante codificación VLC, la codificación acertadamente con relación a símbolos de los símbolos fuente de la primera subsecuencia (108); y mediante configuración aritmética, codificar la segunda subsecuencia (110) de símbolos fuente, en el que el segundo subconjunto es $\{1\}$ estando dispuesta la secuencia de N particiones de modo que una p-ésima partición cubra valores más altos del rango de valores que una q-ésima partición para todo $p, q \in \{1 \dots N\}$ con $p > q$, **caracterizado porque** $N = 3$, y en el que el primer subconjunto es $\{2, 3\}$ usando la codificación VLC (102) un código Golomb-Rice para codificar acertadamente con relación a símbolos los símbolos fuente s_2 , y una codificación Exp-Golomb para codificar acertadamente con relación a símbolos los símbolos fuente s_3 , en el que los valores z del subgrupo de los elementos de sintaxis son valores absolutos, y en el que el codificador VLC (102) se configura para adaptar un parámetro Golomb-Rice del código Golomb-Rice de acuerdo con símbolos previamente codificados.

5. Método de decodificación entrópica que comprende

mediante decodificación VLC, reconstruir acertadamente con relación a palabras de código símbolos fuente de una primera subsecuencia (204) de símbolos fuente a partir de palabras de código de un primer flujo de bits (206); mediante decodificación aritmética, reconstruir una segunda subsecuencia (208) de símbolos fuente; componer una secuencia (226) de elementos de sintaxis a partir de la primera subsecuencia (204) de símbolos fuente y la segunda subsecuencia (208) de símbolos fuente mediante la composición individualmente de cada elemento de sintaxis a partir de un número respectivo n de símbolos fuente mediante, para al menos un subgrupo de elementos de sintaxis, la determinación del número respectivo n de símbolos fuente s_i con $i = 1 \dots n$ dependiendo de en cuál de la secuencia de N particiones (140₁₋₃) en la que se subdivide un rango de valores de los elementos de sintaxis respectivo, cae dentro un valor z de los elementos de sintaxis respectivos, mediante la suma de los valores del número respectivo de símbolos fuente s_i desde 1 a n siempre que el valor de s_i corresponde a un rango de la partición i-ésima de modo que obtenga el valor del elemento de sintaxis z, en el que la composición comprende la recuperación de todos los símbolos fuente s_x , siendo x un miembro de un primer subconjunto de $\{1 \dots N\}$ a partir de la primera subsecuencia (204), y de todos los símbolos fuente s_y , siendo y un miembro de un segundo subconjunto de $\{1 \dots N\}$ que es disyunto respecto al primer subconjunto, a partir de la segunda subsecuencia (208); en el que el segundo subconjunto es $\{1\}$ estando dispuesta la secuencia de N particiones de modo que una p-ésima partición cubra valores más altos del rango de valores que una q-ésima partición para todo $p, q \in \{1 \dots N\}$ con $p > q$, **caracterizado porque** $N = 3$, en el que el primer subconjunto es $\{2, 3\}$ usando la decodificación VLC (200) un código Golomb-Rice para reconstruir acertadamente con relación a las palabras de código los símbolos fuente s_2 , y un código Exp-Golomb para reconstruir acertadamente con relación a las palabras de código los símbolos fuente s_3 , en el que los valores z del subgrupo de los elementos de sintaxis son valores absolutos, y en el que el decodificador VLC (102) se configura para adaptar un parámetro Golomb-Rice del código Golomb-Rice de acuerdo con símbolos fuente previamente reconstruidos.

6. Un programa informático que tiene un código de programa para la realización, cuando se ejecuta en un ordenador, de un método de acuerdo con la reivindicación 4 o 5.

55

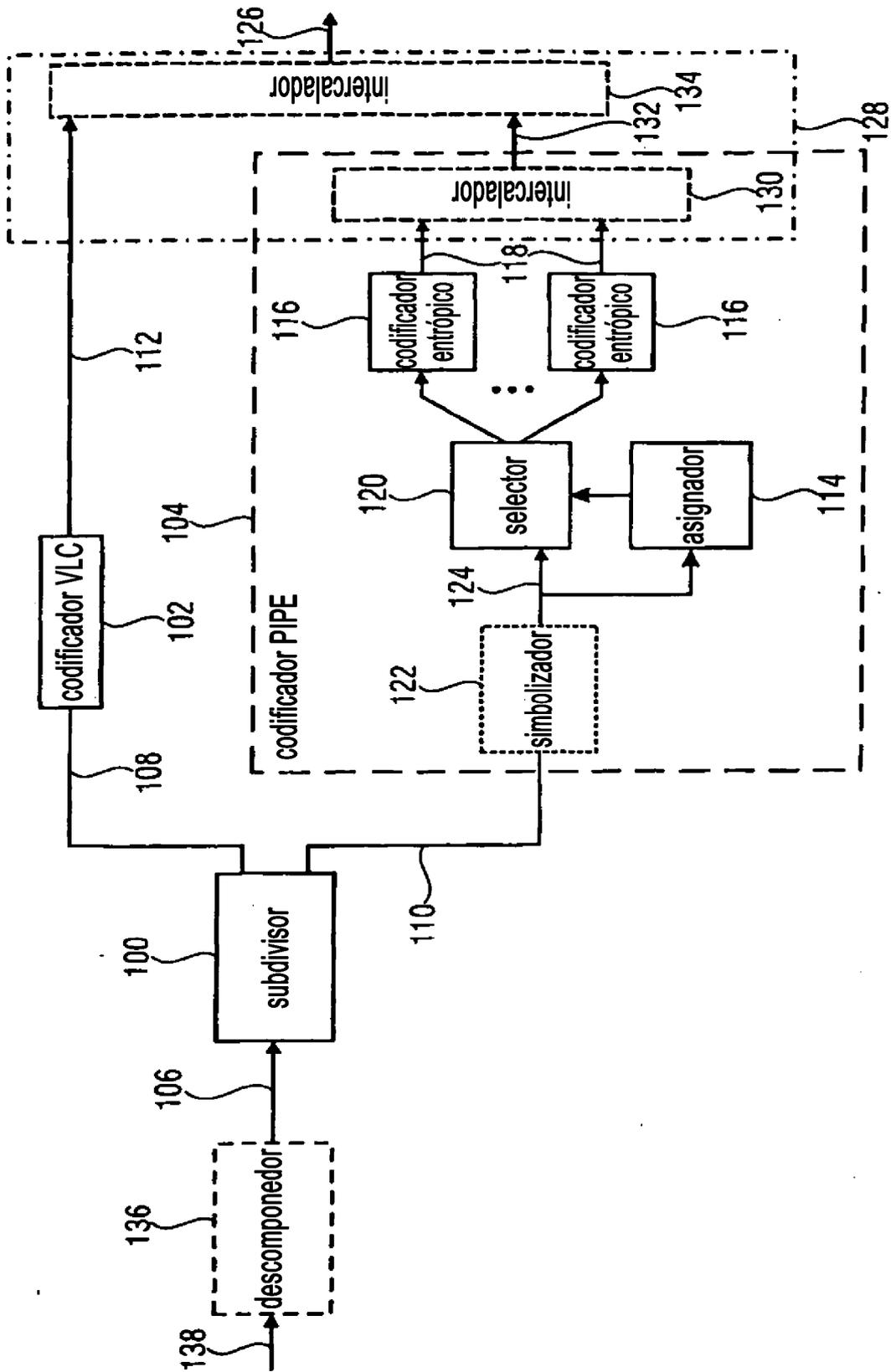


FIGURA 1A

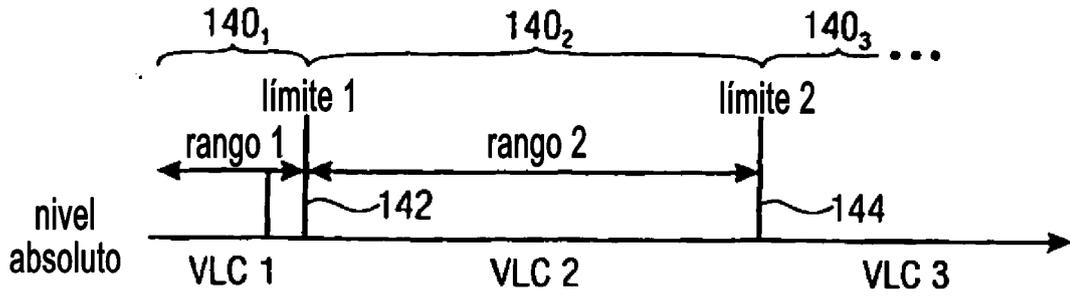


FIGURA 1B

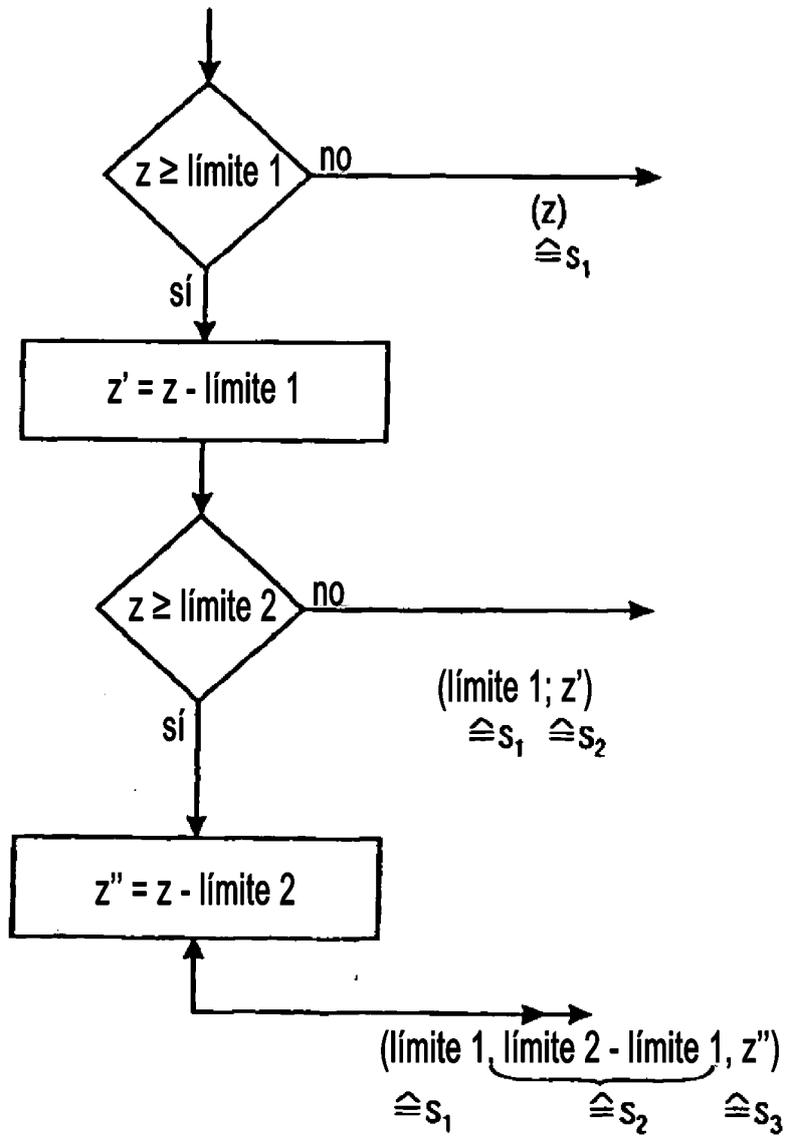


FIGURA 1C

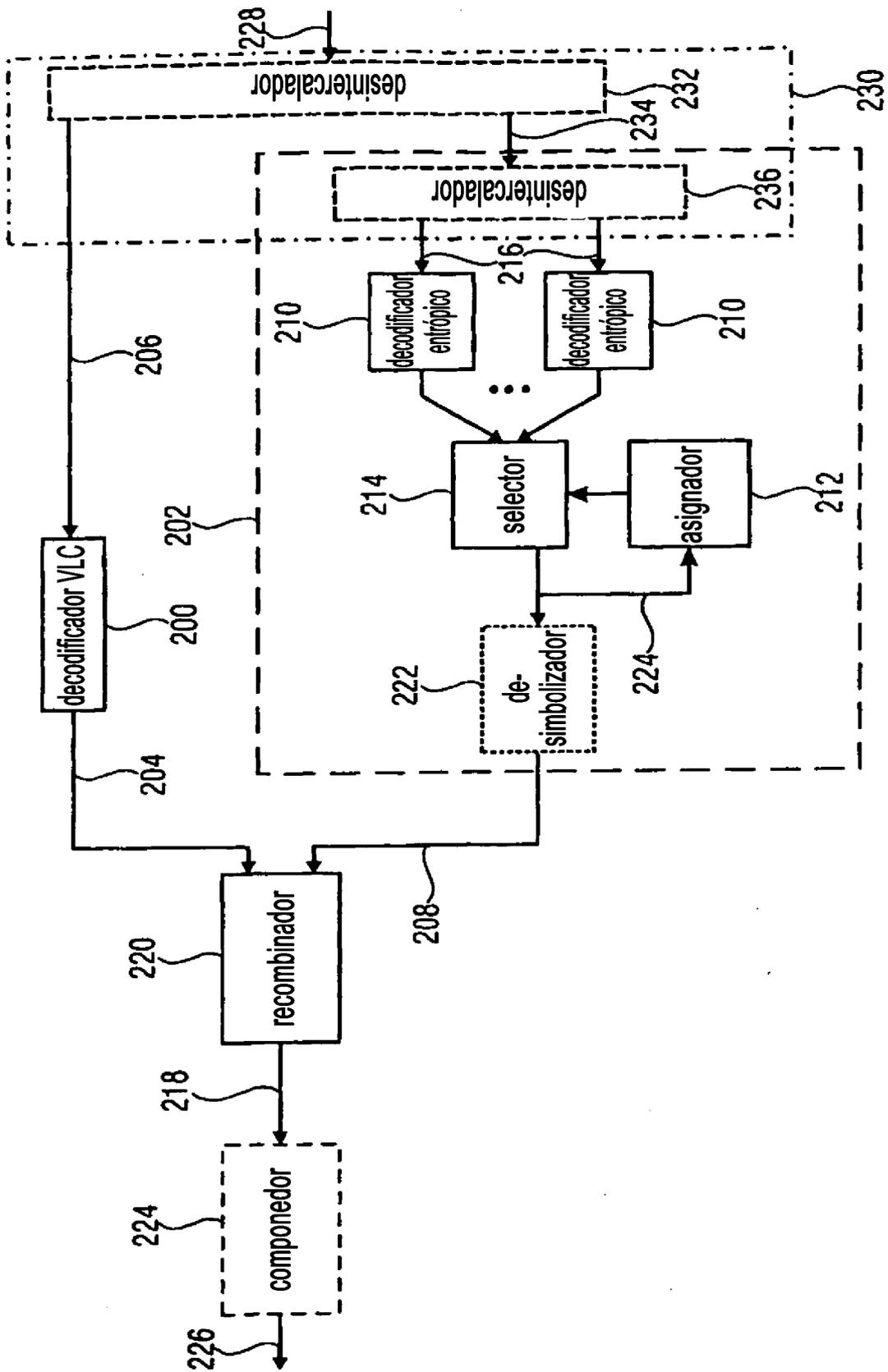


FIGURA 2A

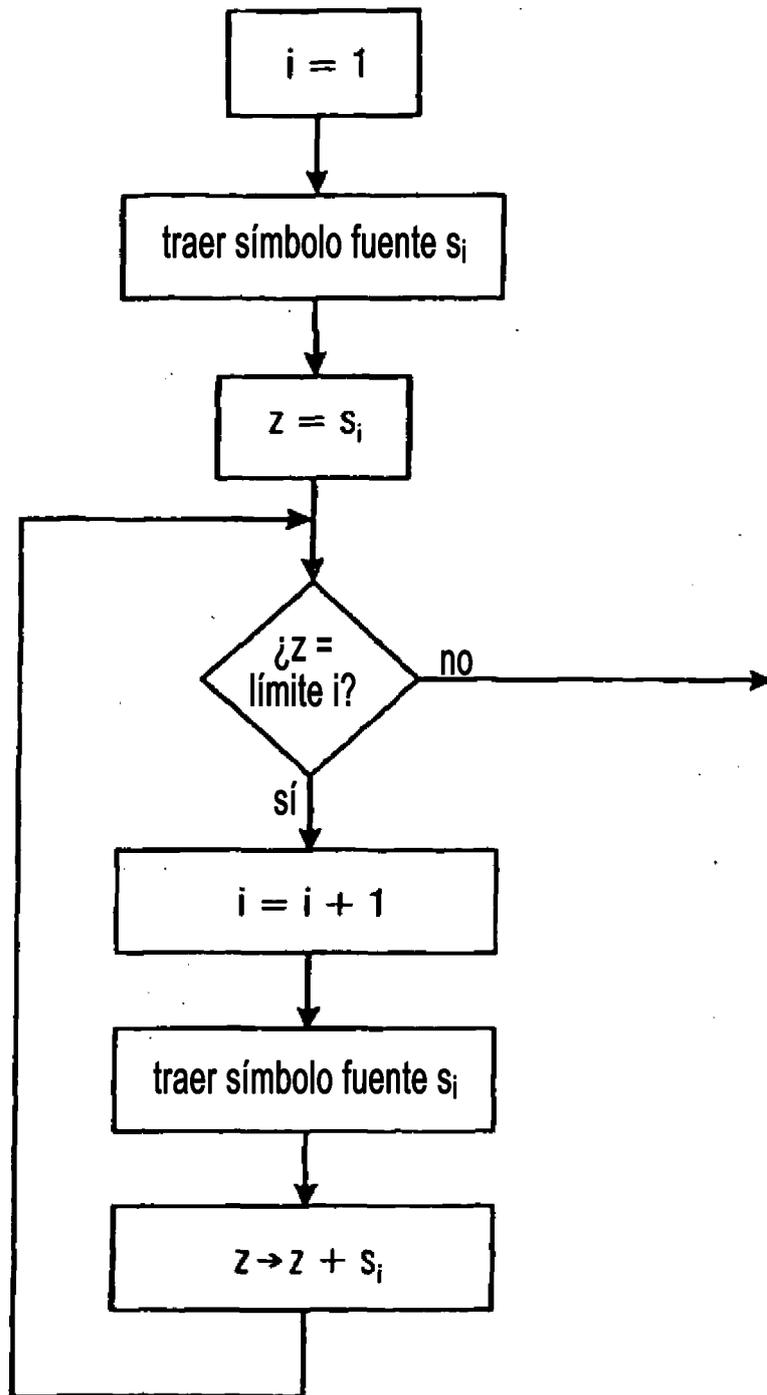


FIGURA 2B

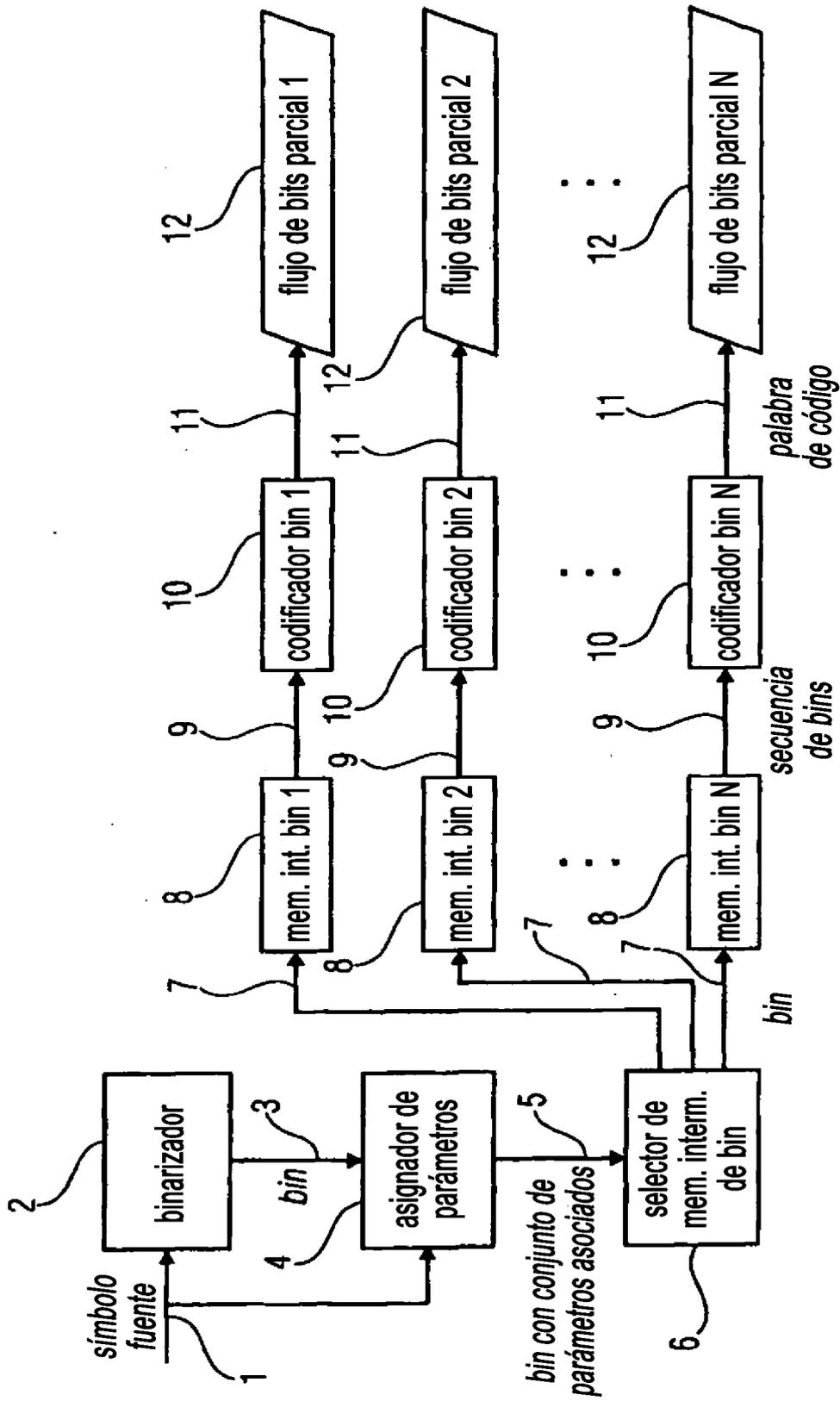


FIGURA 3

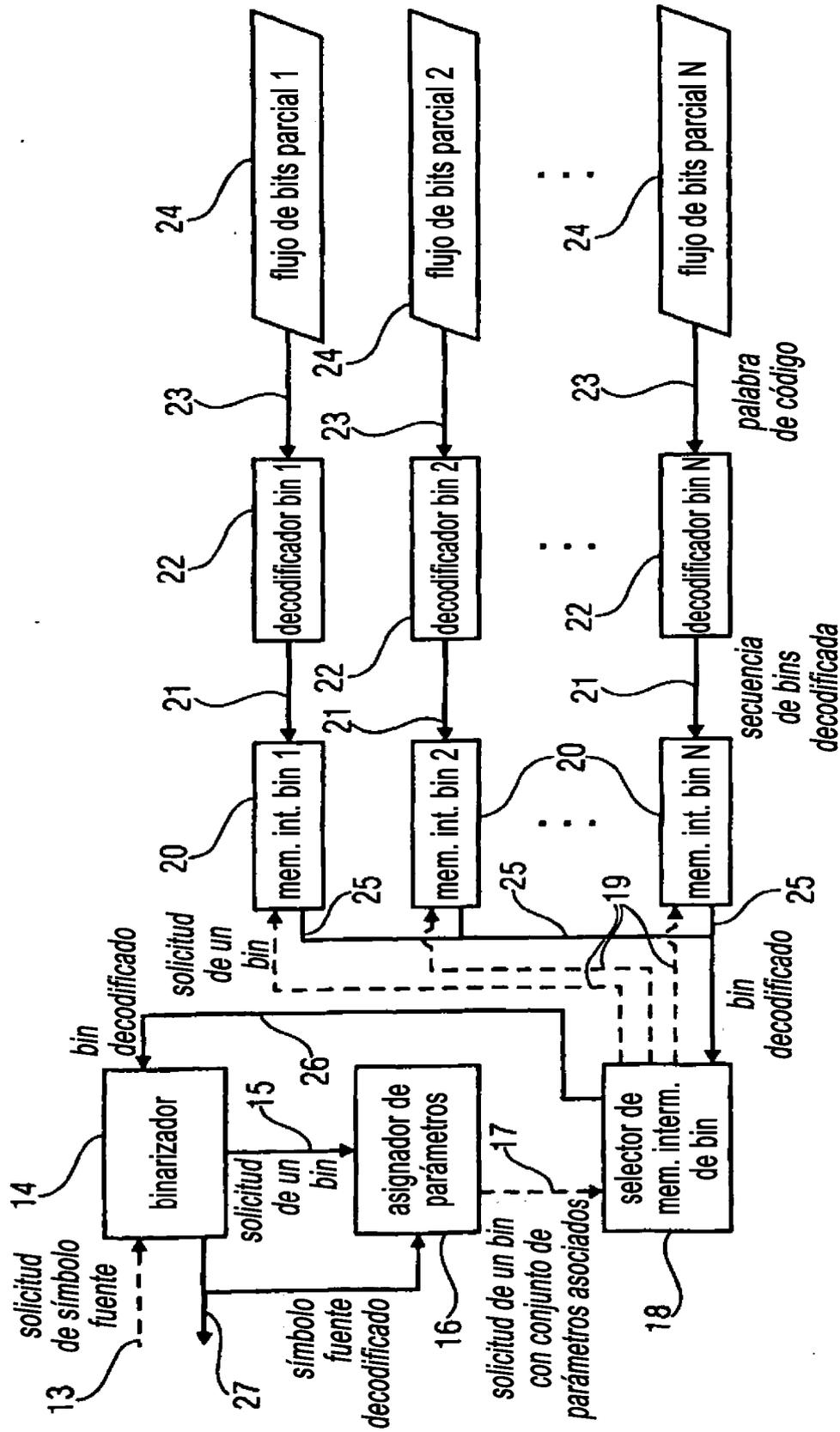


FIGURA 4

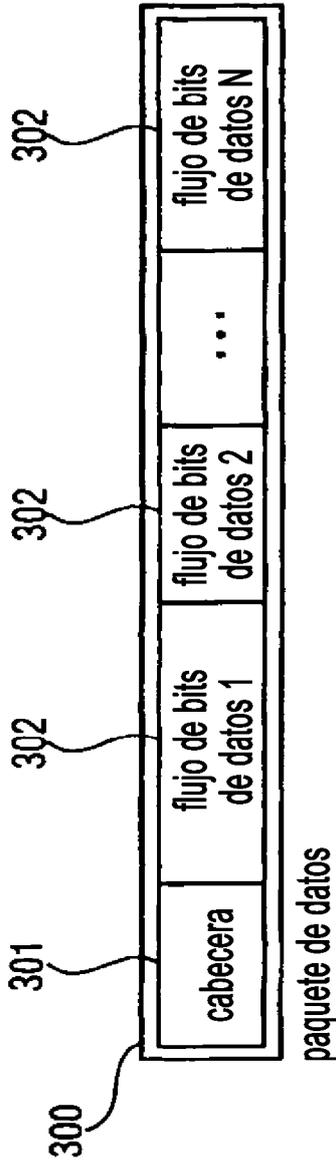


FIGURA 5

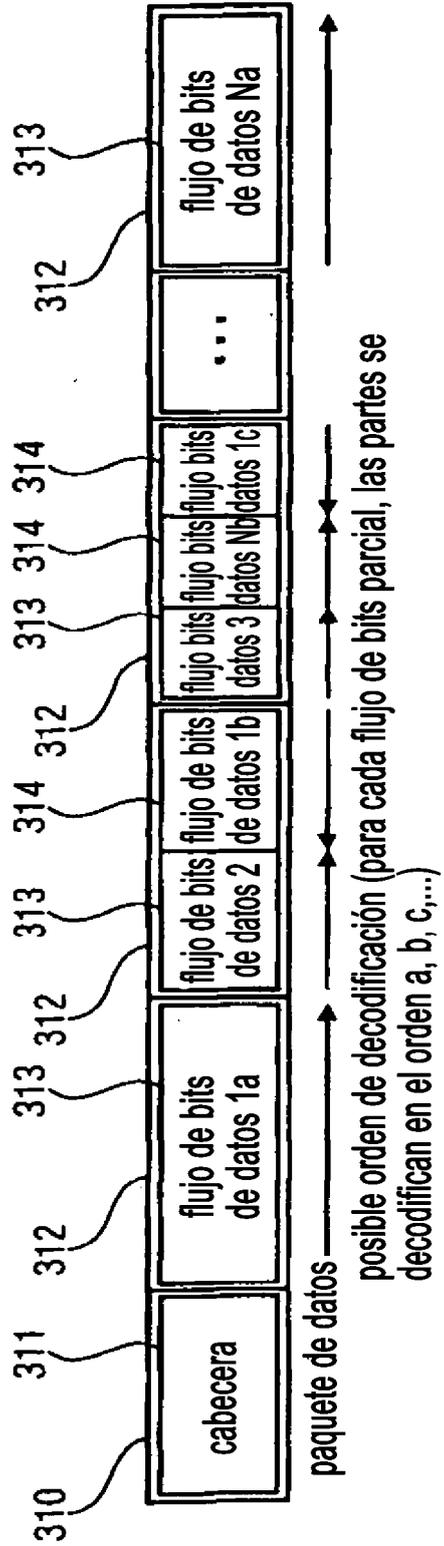


FIGURA 6

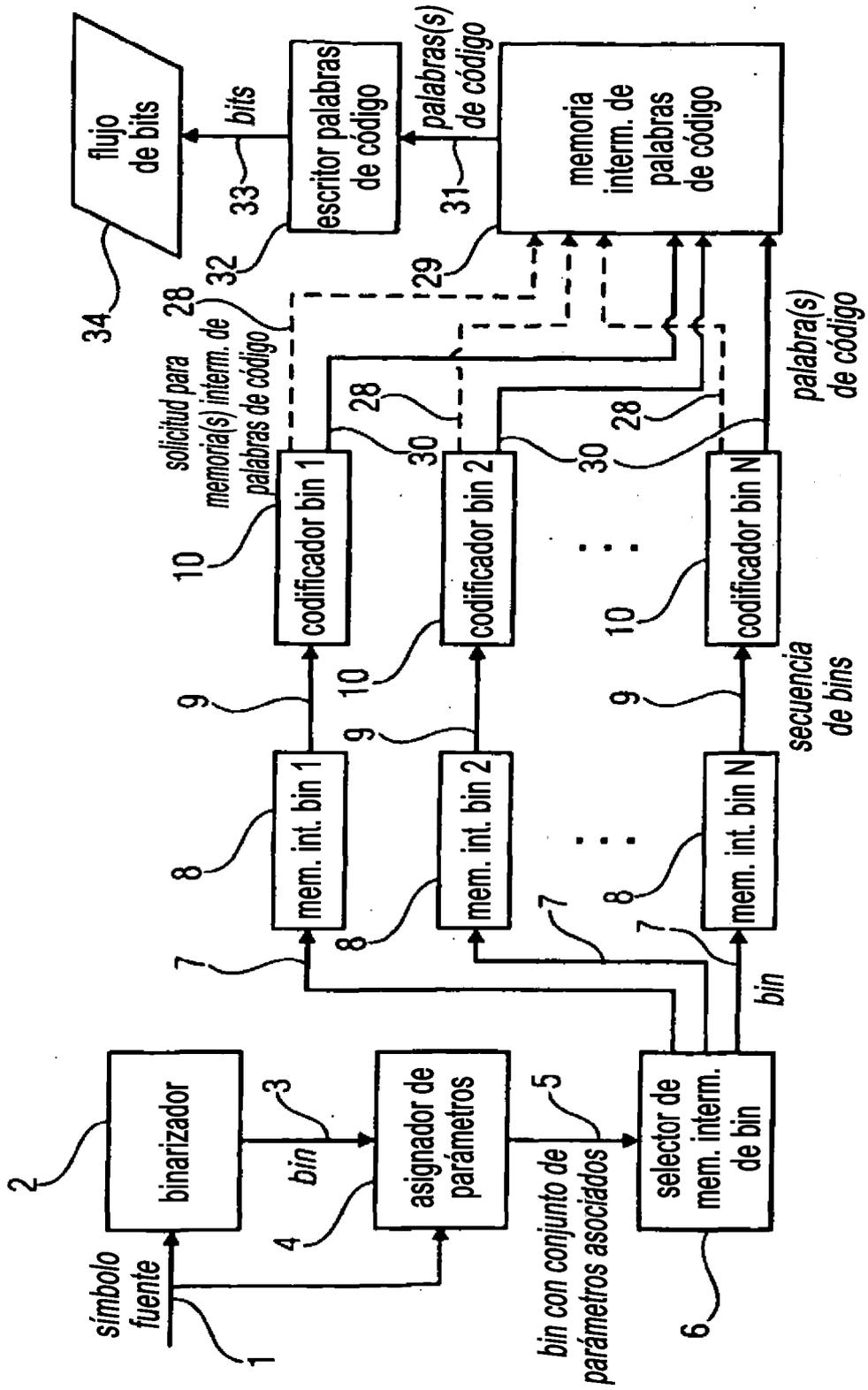


FIGURA 7

tamaño máximo palabra de código	
1	pal. código reservada, codificador bin 3
1	pal. código reservada, codificador bin 3
7	1110 (pal. código completa, codificador bin 1)
7	1010 (pal. código completa, codificador bin 1)
7	pal. código reservada, codificador bin 1
3	01 (pal. código completa, codificador bin 2)
3	pal. código reservada, codificador bin 2
	-- sig. entrada libre mem. interm. pal.código --

orden de procesamiento

tamaño máximo palabra de código	
4	0100 (pal. código completa, codificador bin 2)
5	pal. código reservada, codificador bin 3
5	pal. código reservada, codificador bin 3
5	pal. código reservada, codificador bin 3
6	1110 (pal. código completa, codificador bin 1)
6	pal. código reservada, codificador bin 1
6	pal. código reservada, codificador bin 1
	-- sig. entrada libre mem. interm. pal.código --

orden de procesamiento

FIGURA 8B

FIGURA 8A

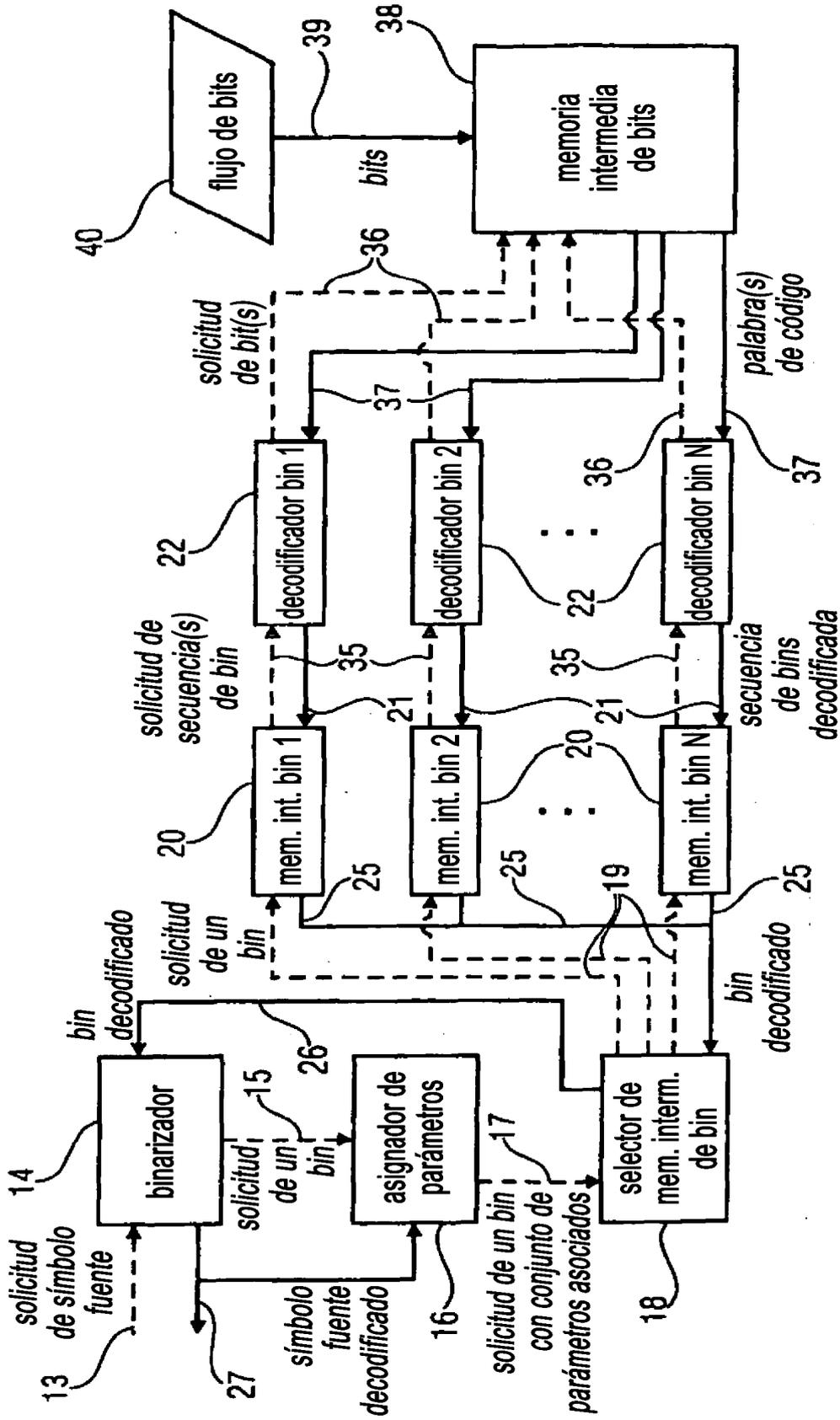


FIGURA 9

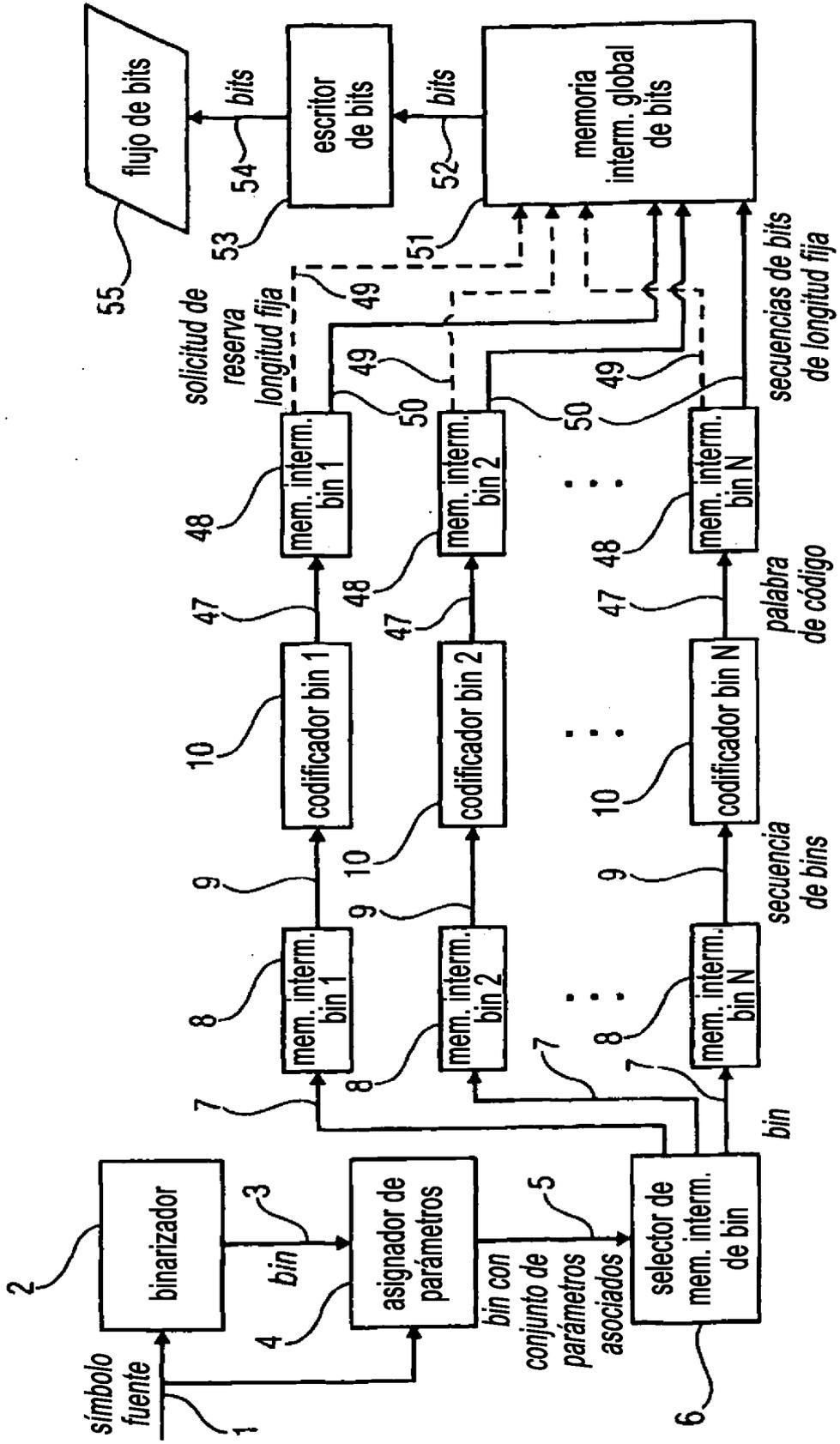


FIGURA 11

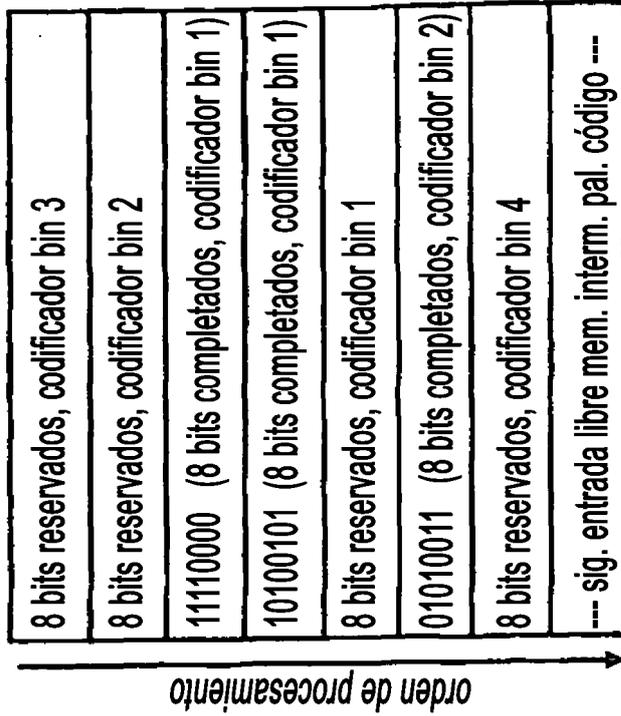


FIGURA 12B

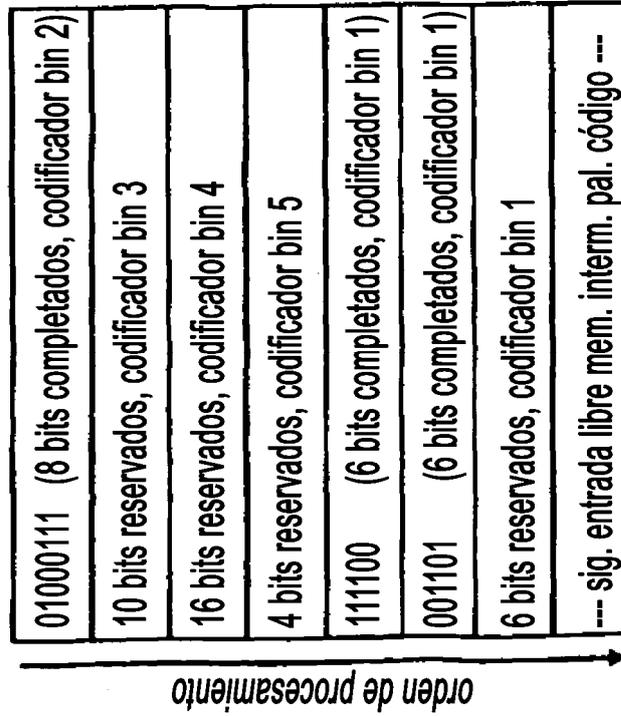


FIGURA 12A

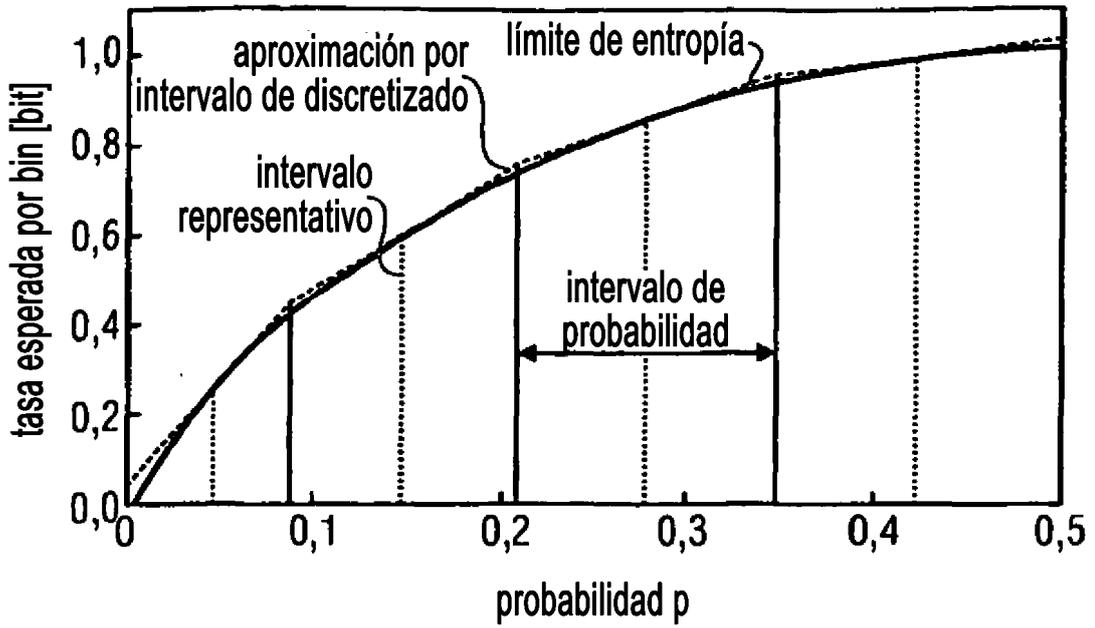


FIGURA 14

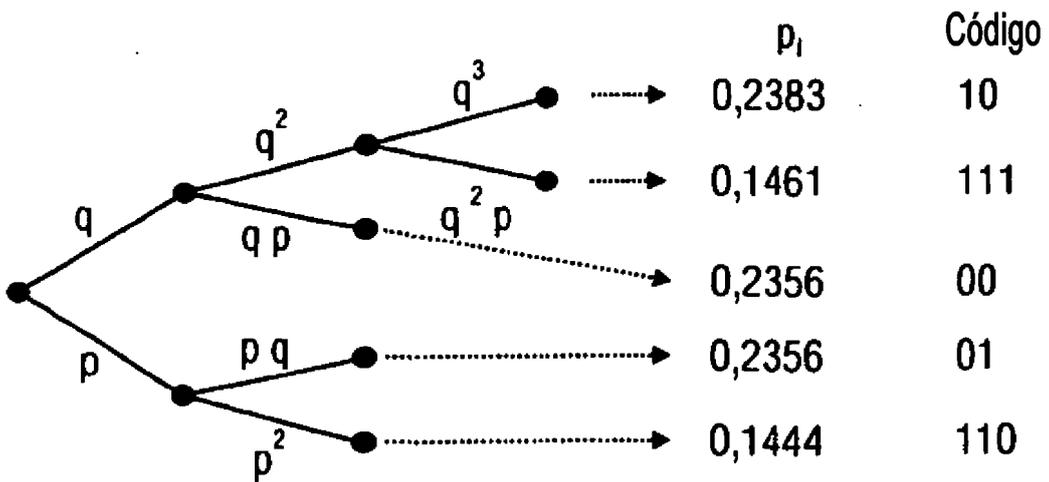


FIGURA 15

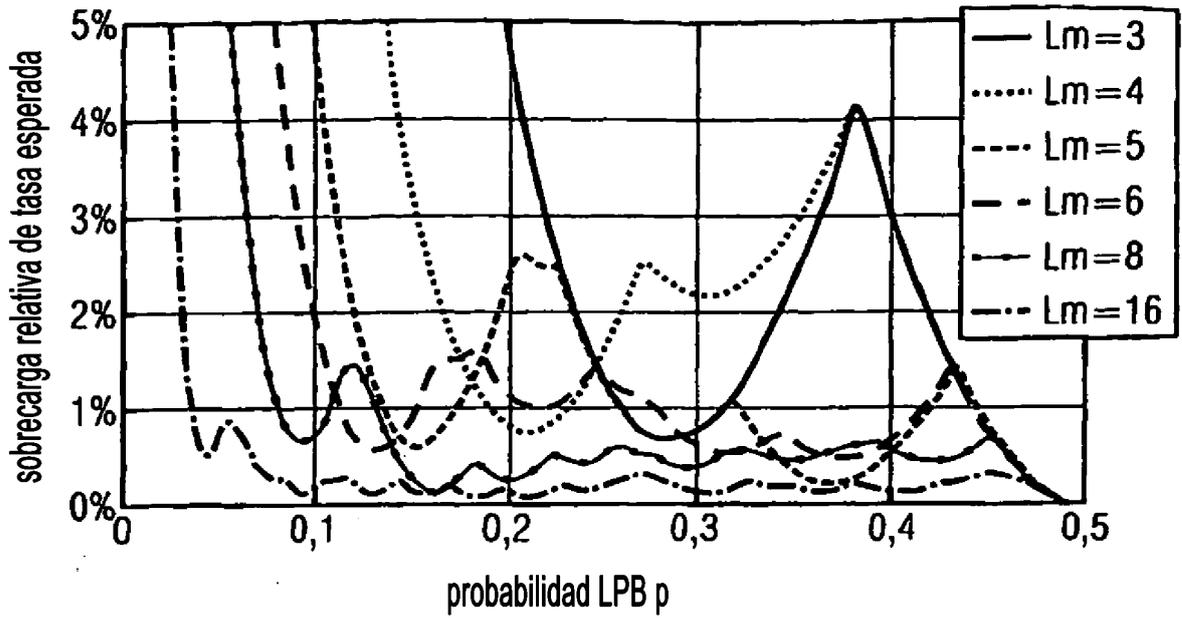


FIGURA 16

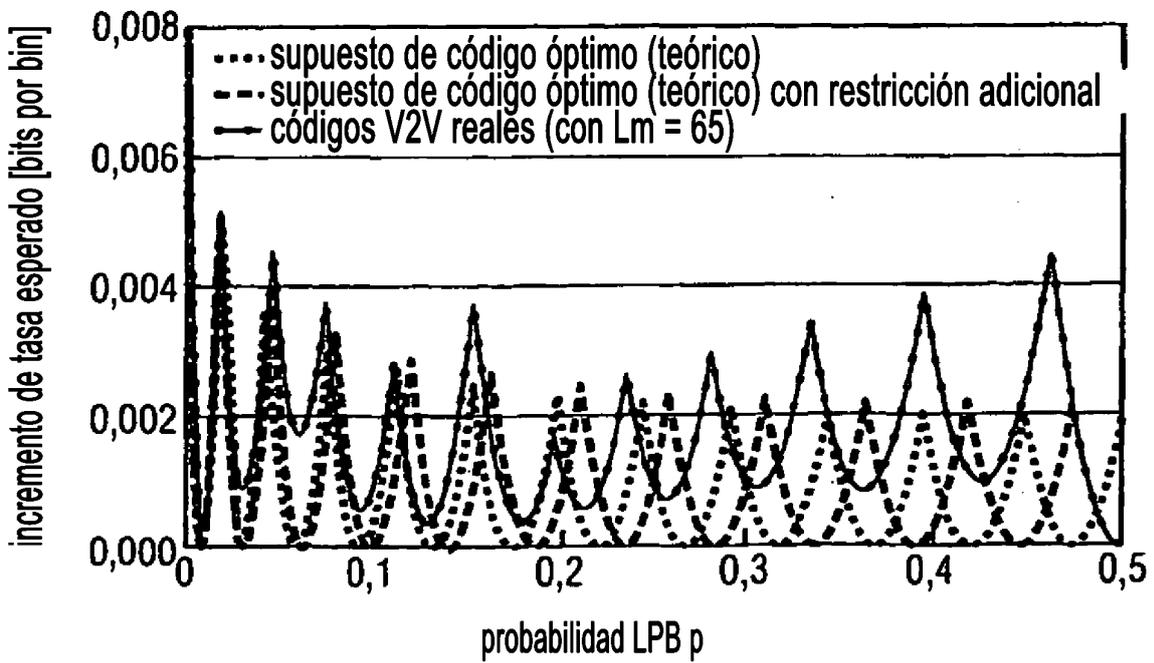


FIGURA 17

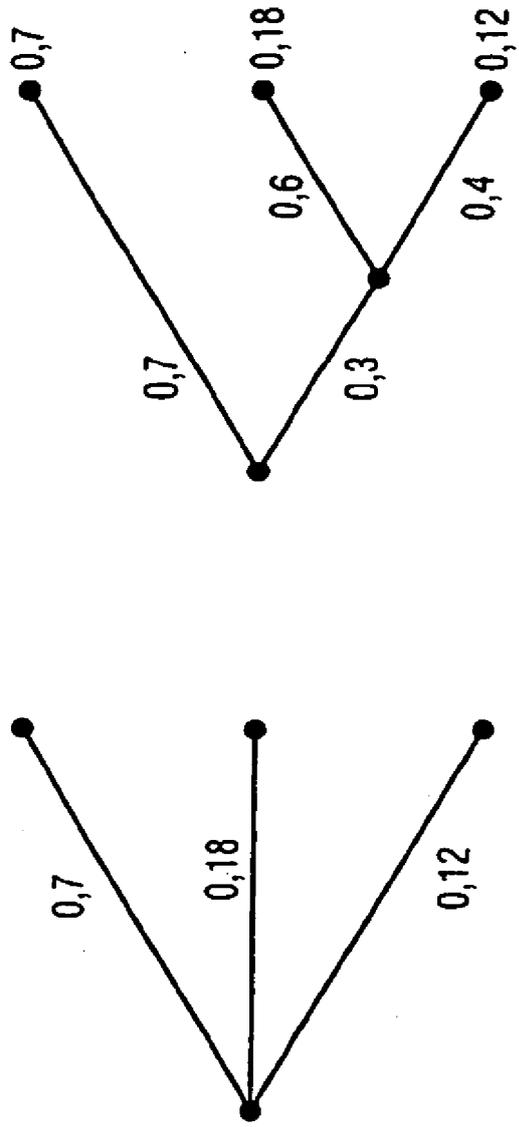


FIGURA 18

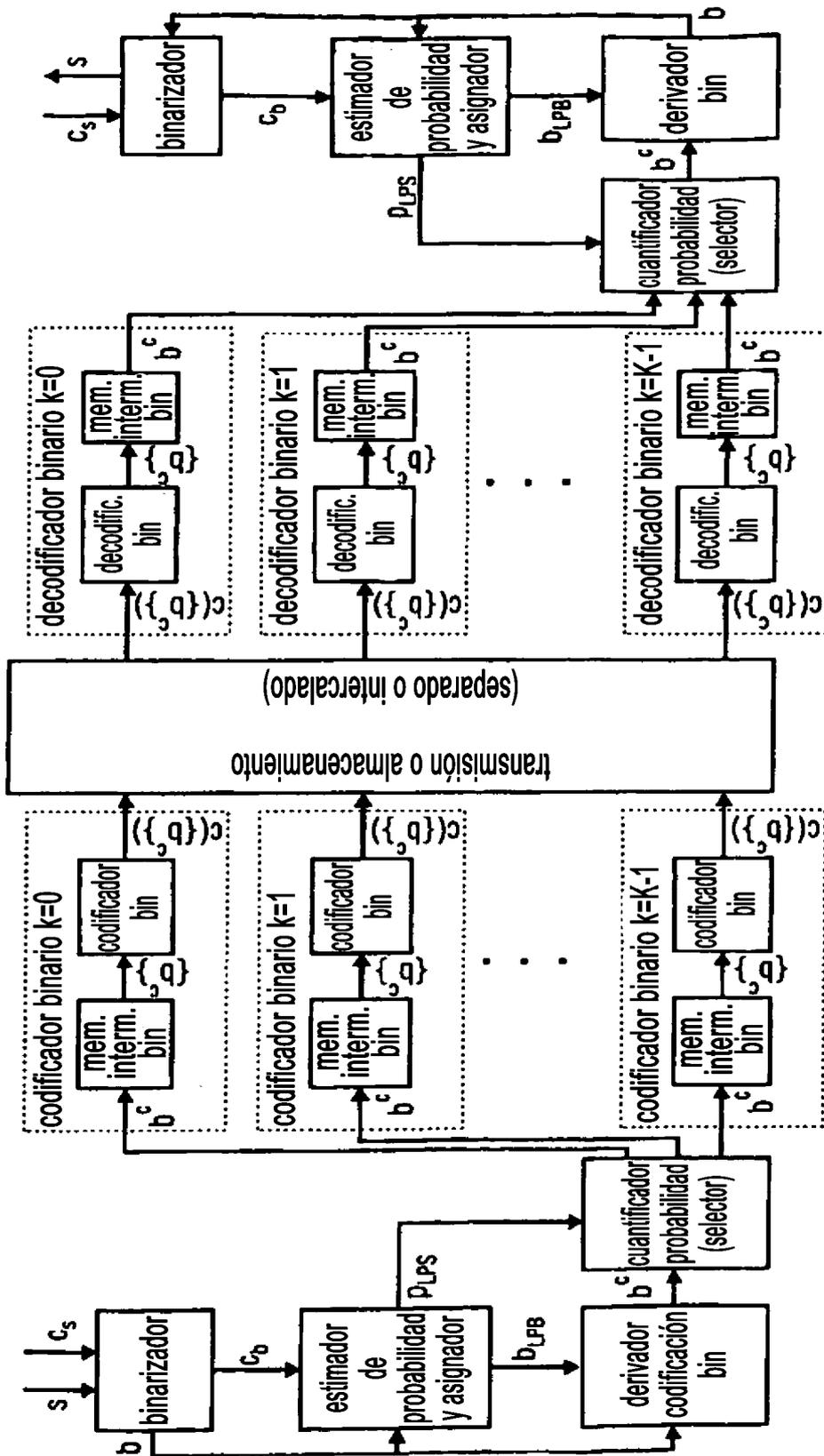


FIGURA 19

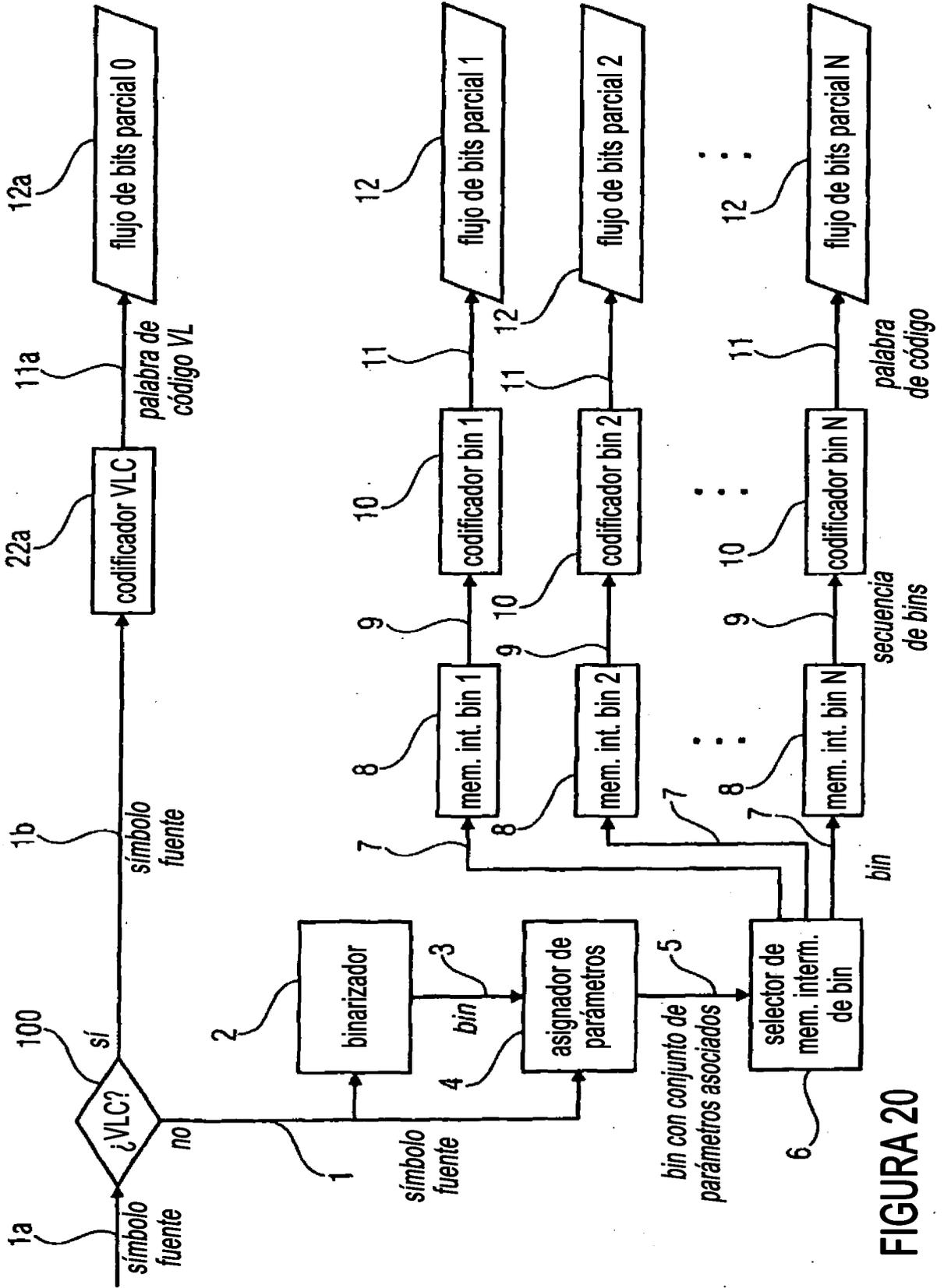


FIGURA 20

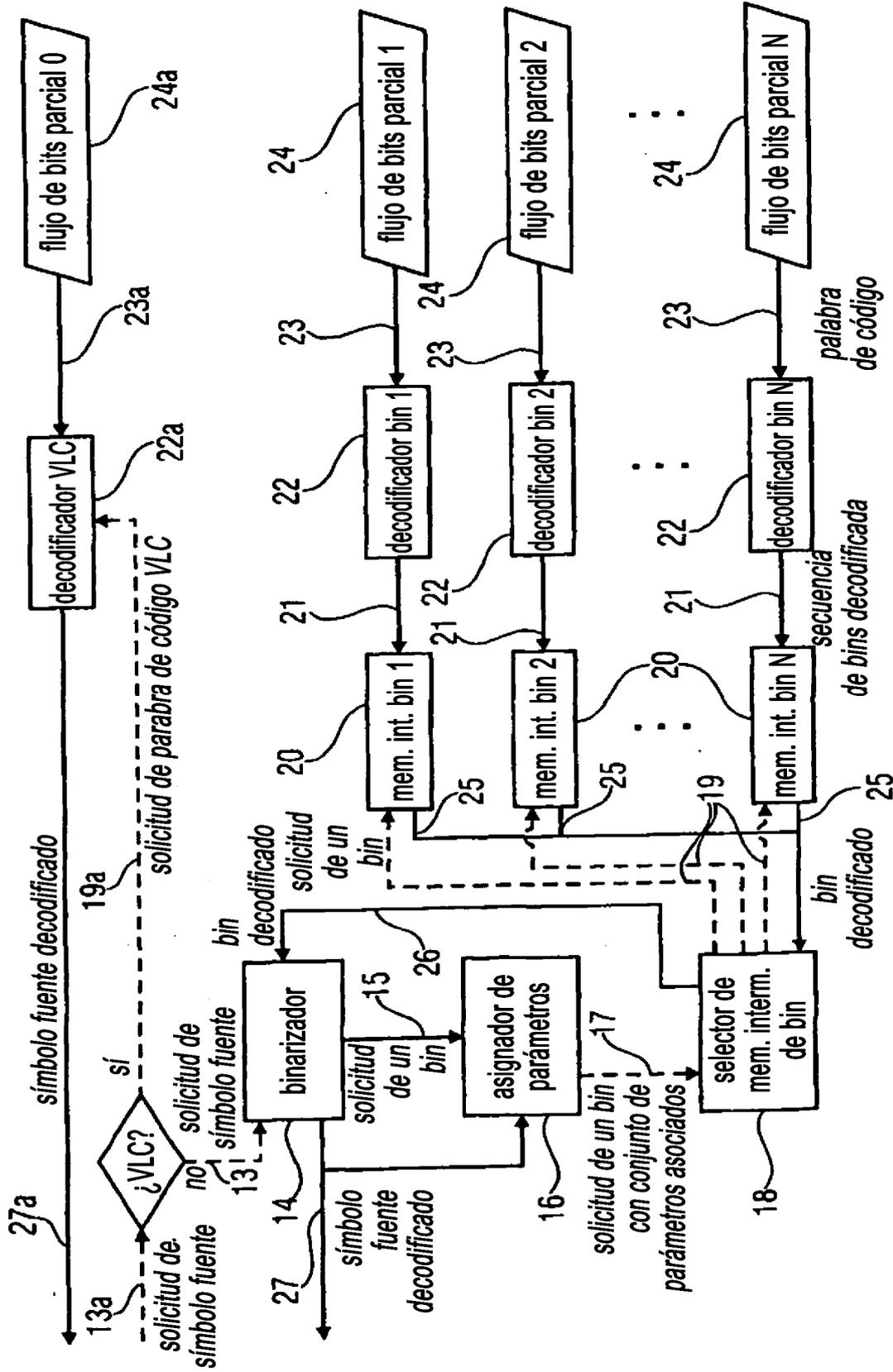


FIGURA 21

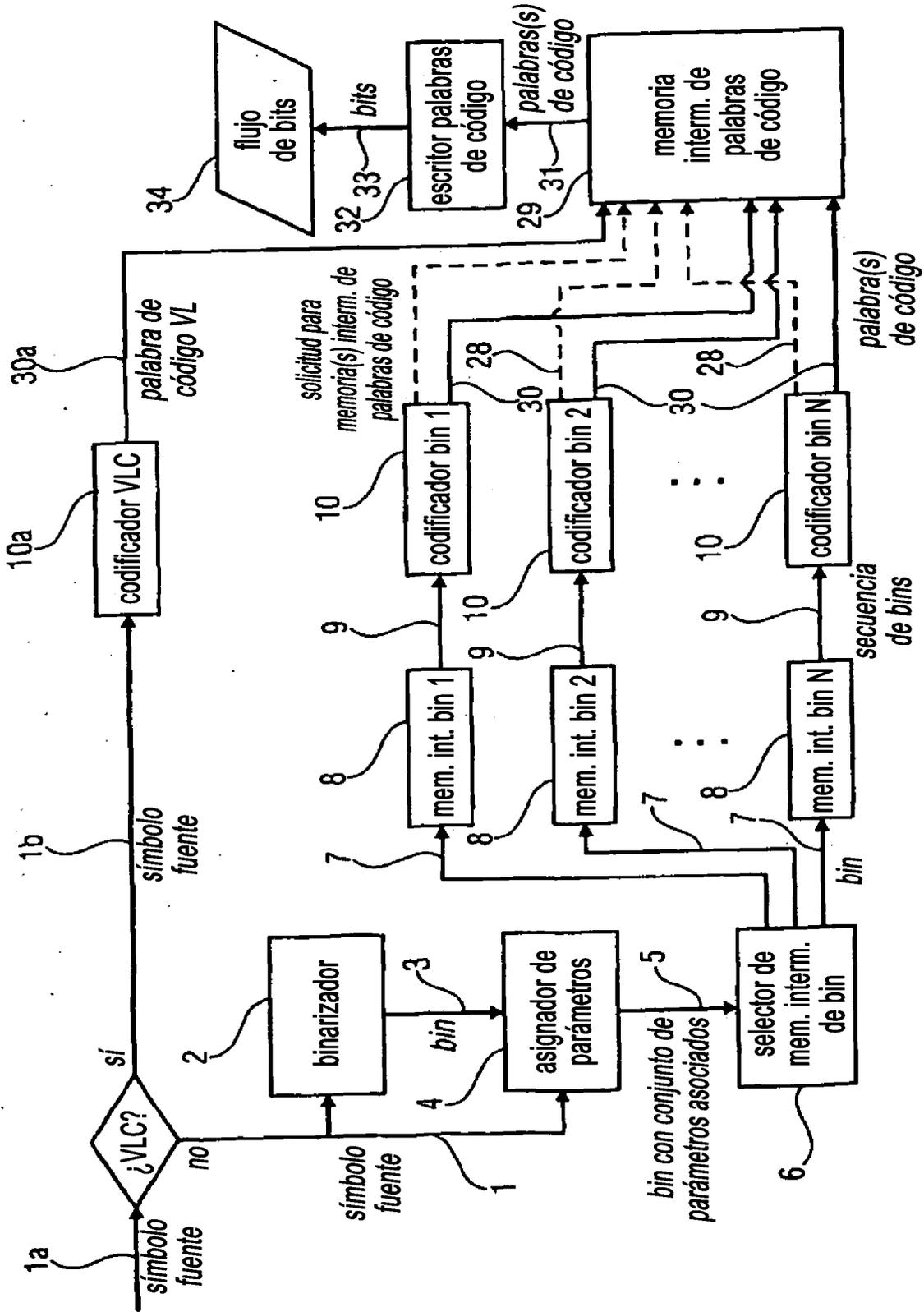


FIGURA 22

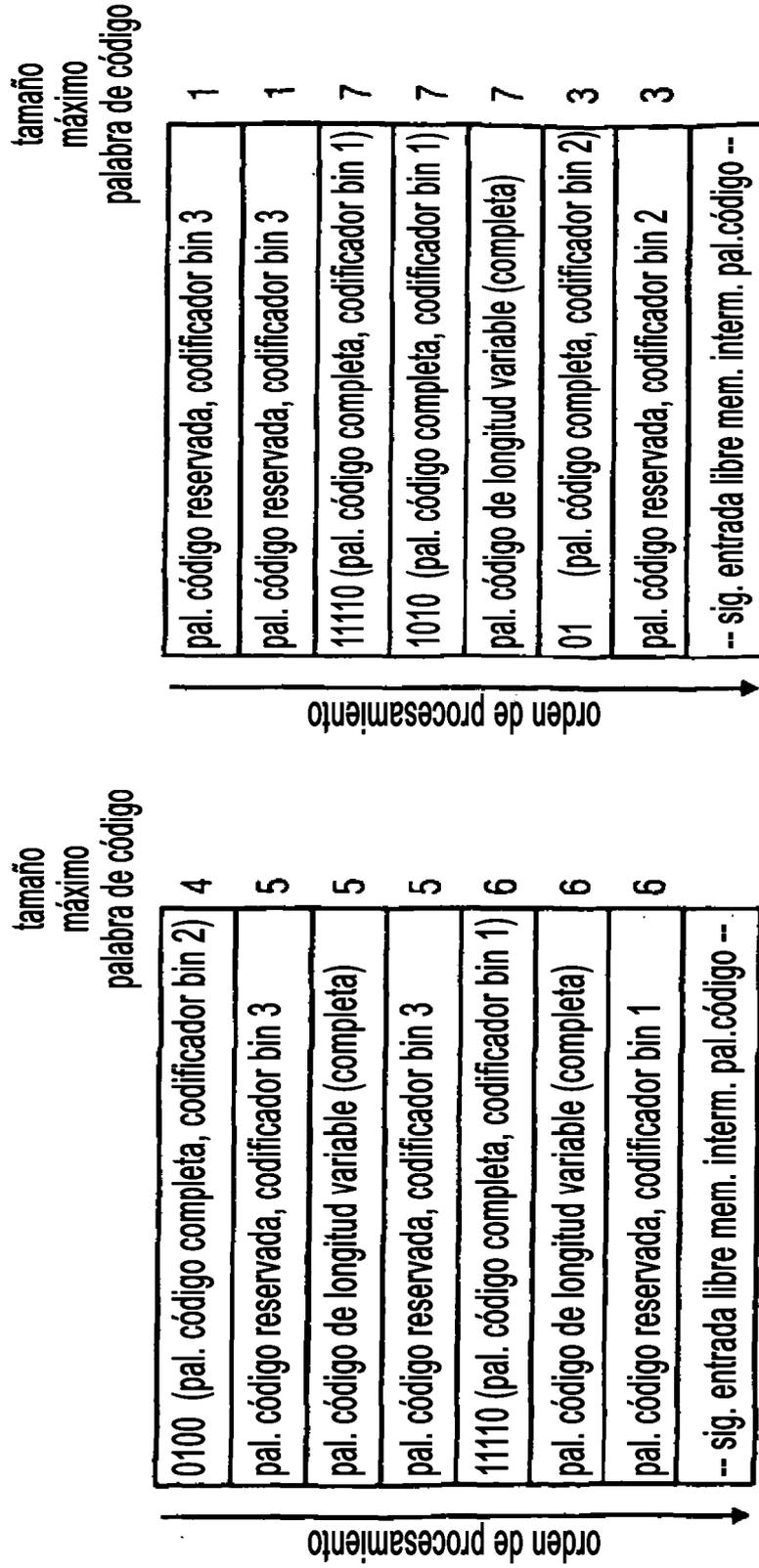


FIGURA 23A

FIGURA 23B

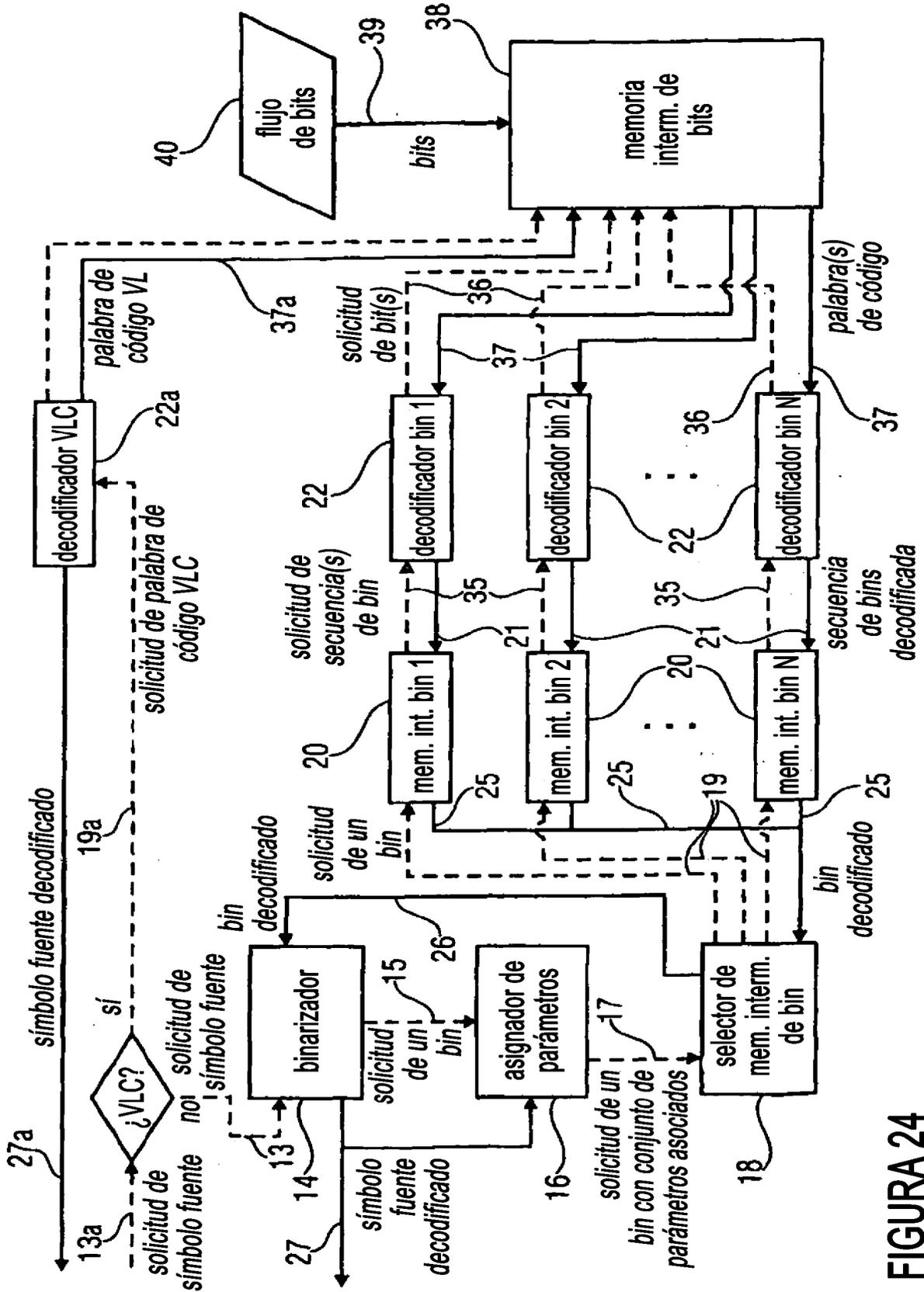


FIGURA 24