

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **3 002 107**

51 Int. Cl.:

G10L 19/02 (2013.01)

G10L 19/022 (2013.01)

H03M 7/30 (2006.01)

H03M 7/40 (2006.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **18.05.2009** **E 24166206 (3)**

97 Fecha y número de publicación de la concesión europea: **27.11.2024** **EP 4376305**

54 Título: **Codificador de audio y decodificador de audio**

30 Prioridad:

11.07.2008 US 7984208 P

08.10.2008 US 10382008 P

45 Fecha de publicación y mención en BOPI de la
traducción de la patente:
06.03.2025

73 Titular/es:

**FRAUNHOFER-GESELLSCHAFT ZUR
FÖRDERUNG DER ANGEWANDTEN
FORSCHUNG E.V. (100.00%)
Hansastraße 27c
80686 München, DE**

72 Inventor/es:

**MULTRUS, MARKUS;
GRILL, BERNHARD;
FUCHS, GUILLAUME;
GEYERSBERGER, STEFAN;
RETTTELBACH, NIKOLAUS y
BACIGALUPO, VIRGILIO**

74 Agente/Representante:

RUÍZ GALLEGOS, Natalia

ES 3 002 107 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).

DESCRIPCIÓN

Codificador de audio y decodificador de audio

- 5 La presente invención se encuentra en el campo de la codificación de audio, especialmente en el campo de la codificación por entropía basada en contexto.

Los conceptos de codificación de audio tradicionales incluyen un esquema de codificación por entropía para reducción de redundancia. Típicamente, la codificación por entropía se aplica a coeficientes espectrales cuantificados para esquemas de codificación basados en el dominio de frecuencia o muestras del dominio del tiempo cuantificadas para esquemas de codificación basados en el dominio del tiempo. Estos esquemas de codificación por entropía típicamente hacen uso de la transmisión de una palabra de código en combinación con un índice de libro de códigos acorde, que posibilita que un decodificador busque una cierta página de libro de códigos para decodificar una palabra de información codificada que corresponde a la palabra de código transmitida en dicha página. En algunos conceptos de codificación, sin embargo, la transmisión del índice de libro de códigos puede no ser aleatoria, por ejemplo, para casos en los que el índice de libro de códigos puede determinarse a partir del contexto de un símbolo, que está, por ejemplo, codificado por entropía como se describe en Meine, Edler, "Improved Quantization and Lossless Coding for Subband Audio Coding" y Meine, "Vektorquantisierung und kontextabhängige arithmetische Codierung für MPEG-4 AAC", Dissertation, Gottfried Wilhelm Universität Hannover, Hanover 2007.

Para codificación de audio basada en el dominio de la frecuencia o espectral, un contexto puede describir símbolos o propiedades estadísticas, como por ejemplo, coeficientes espectrales cuantificados, que son anteriores en tiempo y/o frecuencia. En algunos de los conceptos convencionales, estos símbolos pueden estar disponibles tanto en el lado del codificador como del decodificador, y basándose en estos símbolos, puede determinarse un libro de códigos o contexto de manera síncrona tanto en el lado del codificador como del decodificador.

La Figura 9 ilustrará un ejemplo de un contexto y sus dependencias. La Figura 9 muestra un plano de frecuencia de tiempo, en el que se indica un número de símbolos. El símbolo $S_{n,m}$ indica un símbolo en el tiempo n y frecuencia m . La Figura 9 ilustra que para codificar un cierto símbolo, se usa su contexto para determinar el libro de códigos asociado. Por ejemplo, para un símbolo S_{n_0,m_0} esto sería todos los símbolos con $n < n_0$ y cualquier m , o con $n = n_0$ y $m < m_0$.

En implementaciones prácticas, un contexto no es infinito, pero está limitado. En el ejemplo representado en la Figura 9, el contexto para el símbolo $S_{0,3}$ podría ser, por ejemplo,

35 $S_{0,2}, S_{0,1}, S_{-1,5}, S_{-1,4}, S_{-1,3}, S_{-1,2}, S_{-1,1}, S_{-2,5}, S_{-2,4}, S_{-2,3}, S_{-2,2}, S_{-2,1}$.

Para codificación de audio basada en frecuencia, pueden usarse bancos de filtros adaptativos de señal variable en el tiempo o las denominadas transformaciones de bloque, como se describe, por ejemplo, en Edler, B., "Codierung von Audiosignalen mit überlappender Transformation und adaptiven Fensterfunktionen", Frequenz, Edición 43, septiembre de 1989.

En otras palabras, los cambios de resolución de frecuencia/tiempo pueden tener lugar con el tiempo dentro de estos conceptos de codificación de audio. Un concepto de audio conocido es la denominada AAC (AAC = Codificación de Audio Avanzada), caso en el que se usan longitudes de dos bloques, para los que se codifican por ejemplo 128 o 1024 coeficientes transformados que representan los componentes de frecuencia de 256 o 2048 muestras del dominio de tiempo en ventanas respectivamente.

Estos conceptos permiten la conmutación entre diferentes resoluciones, dependiendo de ciertas características de señal, como por ejemplo aparición de transitorios o tonalidad o si la señal es similar a música o similar a voz, etc. En el caso de la conmutación entre diferentes resoluciones de tiempo/frecuencia, como por ejemplo, entre diferentes tipos de bloque de AAC, el contexto no es consistente. Los conceptos convencionales o implementaciones del estado de la técnica pueden utilizar el reseteo del contexto, es decir, se conmuta básicamente a un estado, en el que no hay contexto disponible, en el que se crea un contexto desde cero. Este enfoque puede funcionar suficientemente bien, por ejemplo, en AAC, puesto que garantiza al menos dos bloques largos u ocho bloques cortos en una fila, donde puede suponerse que la conmutación tiene lugar únicamente raras veces.

Sin embargo, los conceptos convencionales que resetean el contexto son en general subóptimos en términos de eficacia de codificación, puesto que cada vez que se resetea el contexto, la siguiente selección de libro de códigos está basada en valores, que están diseñados como soluciones de repliegue para contexto desconocido. En general, se seleccionan entonces libros de códigos subóptimos. La desventaja en la eficacia de codificación puede ser despreciable para casos en los que tiene lugar la conmutación únicamente raras veces. Sin embargo, para un escenario con conmutación más frecuente, esto conduce a una pérdida significativa en eficacia de codificación. Por una parte, se desea de manera más intensa una conmutación más frecuente para tasas de datos/tasas de muestreo inferiores, puesto que especialmente en este punto, se desea una adaptación óptima de la longitud de transformada

a la característica de la señal. Por otra parte, la eficacia de codificación se reduce significativamente cuando se conmuta con frecuencia.

5 Es el objeto de la presente invención proporcionar un concepto para conmutar entre diferentes longitudes de transformada en codificación de audio que proporcione una eficacia de codificación mejorada.

El objeto se consigue mediante un codificador de audio de acuerdo con la reivindicación 1, y un decodificador de audio de acuerdo con la reivindicación 6.

10 La presente invención está basada en el hallazgo de que en codificación basada en contexto tal como, por ejemplo, codificación por entropía basada en contexto, que puede aplicarse a diferentes resoluciones de tiempo/frecuencia, puede usarse un mecanismo de mapeo de contexto, en el caso de que la resolución de tiempo/frecuencia cambie con el tiempo, consiguiendo de esta manera una eficacia de codificación mejorada. Es un hallazgo de la presente invención que cuando se conmuta entre diferentes resoluciones de tiempo o frecuencia, los contextos para los
15 coeficientes que tienen las nuevas resoluciones pueden derivarse de coeficientes que tienen las resoluciones antiguas. Es un hallazgo de la presente invención que puede usarse, por ejemplo, interpolación, extrapolación, sub-muestreo, muestreo descendente, muestreo ascendente, etc., para adaptación y/o derivación de contexto cuando se conmutan resoluciones de tiempo/frecuencia en codificación de audio.

20 Las realizaciones de la presente invención proporcionan un método de mapeo, que mapea los coeficientes de frecuencia o espectrales de un contexto almacenado, que hace referencia a una resolución antigua, a la resolución de frecuencia de un contexto actual o una trama actual. En otras palabras, la información de contexto anterior puede usarse para la determinación del libro de códigos, es decir para derivar nueva información de contexto. Las realizaciones pueden posibilitar con ello una conmutación más frecuente de longitud de bloque y por lo tanto una
25 mejor adaptación a características de señal sin perder eficacia de codificación.

Las realizaciones de la presente invención se detallarán usando las figuras adjuntas, en las que

30 La Figura 1 muestra una realización de un codificador de audio;

La Figura 2 muestra una realización de un decodificador de audio;

La Figura 3 muestra una realización para un muestreo ascendente de contexto;

35 La Figura 4 muestra una realización para un muestreo descendente de contexto;

La Figura 5 ilustra resoluciones de tiempo y frecuencia de conmutación de audio;

40 La Figura 6 ilustra una implementación de una realización;

La Figura 7a muestra un diagrama de flujo de una realización de un método para codificación;

La Figura 7b ilustra el procedimiento de actualización de contexto general de una realización;

45 La Figura 7c ilustra el procedimiento de actualización de contexto de una realización para cambios de resolución;

La Figura 8 muestra un diagrama de flujo de una realización de un método para decodificación; y

50 La Figura 9 muestra un esquema de codificación de tiempo frecuencia del estado de la técnica.

La Figura 1 muestra una realización de un codificador de audio 100 para codificar segmentos de coeficientes, representando los segmentos de coeficientes diferentes resoluciones de tiempo o frecuencia de una señal de audio muestreada. El codificador de audio 100 comprende un procesador 110 para derivar un contexto de codificación para un coeficiente actualmente codificado de un segmento actual basándose en un coeficiente previamente
55 codificado de un segmento anterior, representando el coeficiente previamente codificado una resolución de tiempo o frecuencia diferente de la del coeficiente codificado actualmente. La realización del codificador de audio comprende adicionalmente un codificador por entropía 120 para codificación por entropía del coeficiente actual basándose en el contexto de codificación para obtener un flujo de audio codificado.

60 En las realizaciones los coeficientes pueden corresponder a muestras de audio, muestras de audio cuantificadas, coeficientes espectrales o de frecuencia, coeficientes escalados, coeficientes transformados o filtrados, etc., o cualquier combinación de los mismos.

En las realizaciones el codificador de audio 100 puede comprender adicionalmente un medio para proporcionar los

segmentos de coeficientes de un flujo de audio, formando los coeficientes una representación espectral de una señal de audio a una resolución espectral que varía entre los coeficientes. Los medios para proporcionar los segmentos pueden adaptarse para determinar los segmentos basándose en diferentes longitudes de ventana de dominio de tiempo o diferentes tramas de audio, es decir, tramas de audio que tienen diferentes longitudes o diferentes números de coeficientes por ancho de banda, es decir una resolución espectral o de frecuencia diferente. Los medios para proporcionar pueden adaptarse para determinar segmentos de 1024 y 128 coeficientes de tiempo, frecuencia o espectrales.

El procesador 110 puede adaptarse en las realizaciones para derivar el contexto de codificación basándose en las representaciones de dominio de la frecuencia o espectral de los coeficientes o segmentos actuales y anteriores. En otras palabras, en las realizaciones los segmentos sucesivos pueden representarse en diferentes dominios de tiempo y/o frecuencia o espectrales. El procesador 110 puede adaptarse para derivar el contexto de codificación por banda de frecuencia o espectral del segmento actual, por ejemplo basándose en coeficientes espectrales vecinos de segmentos anteriores y/o el segmento actual. En las realizaciones, los segmentos pueden determinarse inicialmente en el dominio del tiempo, por ejemplo, poniendo en ventanas un flujo de audio de entrada. Basándose en estos segmentos o coeficientes de dominio de tiempo, pueden determinarse segmentos o coeficientes del dominio de la frecuencia o espectral por medio de transformación. Los segmentos pueden representarse en el dominio de la frecuencia o espectral en términos de energía, una amplitud y fase, una amplitud y signo, etc., por banda de frecuencia o espectral, es decir, los segmentos pueden subdividirse en diferentes bandas de frecuencia o espectrales. El procesador 110 puede a continuación derivar contextos de codificación por banda de frecuencia o espectral en algunas realizaciones.

El procesador 110 y el codificador por entropía 120 pueden configurarse para operar basándose en un muestreo descendente de coeficientes de frecuencia o espectrales de un segmento anterior cuando un segmento anterior que pertenece al contexto comprende una resolución espectral o de frecuencia más precisa que la del segmento actual. En las realizaciones el procesador 110 y el codificador por entropía 120 pueden configurarse para operar basándose en un muestreo ascendente de los coeficientes de frecuencia o espectrales de un segmento anterior, cuando un segmento anterior del contexto comprende una resolución espectral o de frecuencia más baja que la del segmento actual.

Las realizaciones pueden proporcionar un método para codificar segmentos de coeficientes que representan diferentes resoluciones de tiempo o frecuencia de una señal de audio muestreada. El método puede comprender una etapa de derivación de un contexto de codificación para un coeficiente codificado actualmente o actual de un segmento actual basándose en un coeficiente codificado previamente o anterior de un segmento anterior y opcionalmente también basándose en el coeficiente actualmente codificado o actual, en el que el coeficiente codificado previamente o anterior representa una resolución de tiempo o frecuencia diferente de la del coeficiente actualmente codificado o actual. El método puede comprender adicionalmente una etapa de codificación por entropía del coeficiente actual basándose en el contexto de codificación para obtener un flujo de audio codificado.

En correspondencia, las realizaciones pueden comprender un decodificador de audio 200 del cual se representa una realización en la Figura 2. El decodificador de audio 200 está adaptado para decodificar un flujo de audio codificado para obtener segmentos de coeficientes que representan diferentes resoluciones de tiempo o frecuencia de una señal de audio muestreada, el decodificador de audio 200 comprende un procesador 210 para derivar un contexto de codificación para un coeficiente decodificado actualmente o actual basándose en un coeficiente decodificado previamente o anterior, representando el coeficiente decodificado previamente o anterior una resolución de tiempo o frecuencia diferente que la del coeficiente actualmente decodificado. Adicionalmente, el decodificador de audio 200 comprende un decodificador por entropía 220 para la decodificación por entropía del coeficiente actual basándose en el contexto de codificación y el flujo de audio codificado.

En las realizaciones, el decodificador de audio 200 puede comprender un decodificador por entropía 220, que está adaptado para determinar los segmentos de coeficientes decodificados basándose en diferentes longitudes de ventana de dominio de tiempo o diferentes longitudes de trama de audio. El decodificador por entropía 220 puede adaptarse para determinar segmentos de, por ejemplo, 1024 y 128, muestras de dominio de tiempo o frecuencia o coeficientes espectrales. En correspondencia, el procesador 210 puede adaptarse para derivar el contexto de codificación basándose en una representación del dominio de la frecuencia o espectral de coeficientes de segmentos anteriores y/o el segmento actual.

En las realizaciones, el procesador 210 puede adaptarse para derivar el contexto de codificación por banda de frecuencia o espectral del segmento actual, por ejemplo basándose en coeficientes espectrales vecinos del segmento anterior o segmentos y opcionalmente a partir del segmento actual. En otras palabras, los segmentos pueden procesarse en el dominio de la frecuencia o espectral, que puede llevarse a cabo por banda de frecuencia o espectral. En correspondencia, el procesador 210 puede a continuación adaptarse para derivar un contexto específico de banda de frecuencia o espectral.

El decodificador por entropía 200 puede adaptarse para decodificación por entropía del coeficiente actual basándose en una regla de codificación por entropía o de longitud variable.

5 El procesador 210 puede adaptarse para derivar el contexto de codificación basándose en un muestreo descendente de coeficientes de frecuencia o espectrales de un segmento anterior cuando el segmento anterior comprende más coeficientes por ancho de banda (es decir una resolución espectral o de frecuencia más precisa) que la del segmento actual. En realizaciones adicionales, el procesador 210 y el codificador por entropía 220 pueden configurarse para operar basándose en un muestreo ascendente de coeficientes espectrales de un segmento anterior, cuando el segmento anterior comprende menos coeficientes por ancho de banda (es decir una resolución espectral o de frecuencia más basta) que la del segmento actual.

15 En consecuencia, las realizaciones pueden proporcionar un método para decodificar un flujo de audio codificado para obtener segmentos de coeficientes que representan muestras de audio decodificadas. El método para decodificar puede comprender una etapa de derivación de un contexto de codificación para un coeficiente decodificado actualmente o actual de un segmento actual basándose en un coeficiente decodificado previamente o anterior de un segmento anterior, representando el coeficiente decodificado previamente o anterior una resolución de tiempo o frecuencia diferente que la del coeficiente actualmente decodificado. Adicionalmente, el método puede comprender una etapa de decodificación por entropía el coeficiente actual basándose en el contexto de codificación y el flujo de audio codificado. Opcionalmente, el método puede comprender una etapa de determinación de los segmentos de coeficientes de audio codificados desde el flujo de audio codificado, representando los segmentos diferentes números de coeficientes de audio.

25 La Figura 3 ilustra cómo un procesador 110; 210 puede derivar un contexto de codificación para un segmento actual de $M_{c,nuevo}$ coeficientes basándose en un segmento anterior de $M_{c,antiguo}$ coeficientes, en el que el segmento anterior comprende un número diferente de coeficientes de audio que el del segmento actual. En la realización representada en la Figura 3, el número de coeficientes del segmento M determina la resolución de frecuencia o espectral del segmento. La realización puede comprender un método de mapeo, que mapea los $M_{c,antiguo}$ coeficientes de un segmento anterior a $M_{c,nuevo}$ coeficientes que tienen la misma resolución de frecuencia o espectral del contexto que el segmento actual. La Figura 3 muestra dos conjuntos de coeficientes en dos segmentos, es decir el segmento anterior original 310 que representa $M_{c,antiguo}$ coeficientes, $S_{n,0}$, $S_{n,1}$, $S_{n,2}$, etc., y en correspondencia, el segmento anterior mapeado 320, que tiene una resolución superior, es decir, $M_{c,nuevo}$ es mayor que $M_{c,antiguo}$, que representa $M_{c,nuevo}$ coeficientes, $S_{n,0}$, $S_{n,1}$, $S_{n,2}$, $S_{n,3}$, etc.

35 En general, pueden distinguirse dos realizaciones, dependiendo de si la resolución del contexto del segmento actual es superior o inferior que la resolución del contexto del segmento anterior. La Figura 3 ilustra una realización, en la que la resolución del segmento anterior de $M_{c,antiguo}$ coeficientes es inferior que la resolución del segmento actual de $M_{c,nuevo}$ coeficientes. La Figura 3 muestra los coeficientes del segmento anterior 310 y los símbolos para el segmento anterior mapeado 320. A partir de la Figura 3 puede observarse que la resolución del segmento actual de $M_{c,nuevo}$ coeficientes es superior que la resolución del segmento anterior 310 que tiene únicamente $M_{c,antiguo}$ coeficientes. En una realización el segmento anterior 310 se muestrea ascendentemente a un segmento 320 de $M_{c,nuevo}$ coeficientes para hacer coincidir la resolución de frecuencia o espectral del segmento actual. Esto puede incluir puro muestreo ascendente con mecanismos de duplicación de símbolo y decimación por ejemplo, repetir cada valor $M_{c,nuevo}$ veces antes de decimar el segmento muestreado ascendentemente resultante manteniendo únicamente 1 coeficiente cada $M_{c,antiguo}$. Otros mecanismos de interpolación o extrapolación pueden usarse también.

45 En las realizaciones, el mapeo puede llevarse a cabo para todos los segmentos anteriores 310, que son necesarios para determinar los contextos para el segmento actual, por ejemplo en el tiempo n, en otras palabras, puede tenerse en cuenta múltiples segmentos anteriores, es decir, segmentos anteriores en los tiempos n-1, n-2, etc. En general, las realizaciones pueden tener en cuenta múltiples intervalos de tiempo o segmentos anteriores, el número de intervalos de tiempo necesarios para definir un contexto completo puede ser diferente para diferentes implementaciones o realizaciones.

50 La Figura 4 ilustra otra realización, en la que los coeficientes de un segmento anterior 410 se muestrean descendentemente a un segmento 420 usado para calcular los contextos del segmento actual, es decir, en el que el número de coeficientes $M_{c,antiguo}$ del segmento anterior 410 es superior al número de coeficientes $M_{c,nuevo}$ del segmento actual. La Figura 4 usa una ilustración similar que la Figura 3, por consiguiente se muestran múltiples coeficientes en cada segmento 410 y 420. Como se ilustra en la Figura 4, $M_{c,antiguo}$ es mayor que $M_{c,nuevo}$. Por lo tanto, los $M_{c,antiguo}$ coeficientes se sub-muestrean, para adaptar la resolución de frecuencia o espectral del segmento actual de $M_{c,nuevo}$ coeficientes, es decir en las realizaciones los segmentos anteriores que tienen una resolución superior pueden sub-muestrearse para adaptar la resolución del segmento actual que tiene una resolución inferior. En las realizaciones esto puede incluir muestreo descendente puro con mecanismos de duplicación y decimación de coeficientes como por ejemplo, repetir cada valor $M_{c,nuevo}$ veces antes de decimar el segmento muestreado ascendentemente resultante manteniendo únicamente 1 coeficiente cada $M_{c,antiguo}$. En otras realizaciones, pueden tenerse en cuenta operaciones de filtro, como por ejemplo promedio de dos o múltiples valores adyacentes.

La Figura 5 ilustra otra realización, en la que se lleva a cabo conmutación entre diferentes resoluciones. La Figura 5 muestra un plano de tiempo/frecuencia, en el que se muestran tres segmentos posteriores de coeficientes de audio, en concreto 510, 520 y 530. Cada uno de los segmentos 510, 520 y 530 corresponde a un único conjunto de coeficientes. En la realización ilustrada en la Figura 5, se supone que el segundo segmento 520 es el doble de largo que el primer y tercer segmentos 510 y 530. Esto puede conseguirse usando diferentes ventanas cuando se segmenta en el dominio del tiempo, como se hace por ejemplo en AAC. En la realización ilustrada en la Figura 5, se supone que la tasa de muestreo permanece constante, en otras palabras, el segundo segmento más largo 520 comprende el doble de coeficientes de audio por ancho de banda que el primer o tercer segmento 510 o 530.

La Figura 5 muestra que en este caso la resolución en el dominio de la frecuencia o espectral escala con la extensión del segmento en el dominio del tiempo. En otras palabras, cuanto más corta es la ventana en el dominio del tiempo, menor es la resolución en el dominio de la frecuencia o espectral. Cuando se evalúan contextos para codificar las muestras en el dominio de la frecuencia o espectral, la Figura 5 muestra que la codificación necesita tener una versión de resolución superior del segmento 510 cuando se codifica el segundo segmento 520 como en el ejemplo considerado, tiene que derivarse una resolución doble del segmento 510. En otras realizaciones, especialmente cuando se usan otras transformaciones de tiempo-frecuencia o bancos de filtros, pueden resultar otras relaciones entre las resoluciones del dominio del tiempo y dominio de frecuencia.

De acuerdo con una realización, los coeficientes codificados durante el primer segmento 510 proporcionan una base para determinar el contexto para el segundo segmento 520, por ejemplo, por medio de un muestreo ascendente intermedio. En otras palabras, el contenido de contexto que proviene desde el primer segmento 510 puede obtenerse mediante un muestreo ascendente del primer segmento 510, por ejemplo, en términos de interpolación o extrapolación, para derivar el contexto para el segundo segmento 520, que tiene una resolución superior.

Como se muestra en la Figura 5, cuando se conmuta desde el segundo segmento 520 al tercer segmento 530, el elemento constituyente de contexto tiene que cambiar también, a medida que la resolución se ha reducido ahora. De acuerdo con una realización, los coeficientes codificados durante el segundo segmento 520 pueden usarse para derivar el contexto para el tercer segmento, por medio de un muestreo descendente intermedio. Esto puede llevarse a cabo, por ejemplo, en términos de promedio o simplemente usando únicamente cada segundo valor y otras medidas para muestreo descendente.

Las realizaciones consiguen la ventaja de una eficacia de codificación aumentada, teniendo en cuenta el contexto pasado derivado de segmentos anteriores incluso cuando tienen lugar cambios en la resolución o longitud de ventana. Los elementos de contexto constituyentes pueden adaptarse a nuevas resoluciones, en términos de muestreo ascendente o descendente, por ejemplo, con inter- o extrapolación, filtración o promedio, etc.

A continuación se presentará una realización más específica en términos de codificación sin ruido espectral. La codificación sin ruido espectral puede usarse para reducir adicionalmente la redundancia de un espectro cuantificado en codificación de audio. La codificación sin ruido espectral puede estar basada en una codificación aritmética en conjunto con adaptación de contexto de manera dinámica.

La codificación sin ruido puede estar basada en valores espectrales cuantificados y puede usar las tablas de frecuencia acumulativas dependientes del contexto derivadas desde, por ejemplo, cuatro tuplas vecinas previamente decodificadas. La Figura 6 ilustra otra realización. La Figura 6 muestra un plano de tiempo frecuencia, en el que a lo largo del eje del tiempo se indexan tres intervalos de tiempo, n , $n-1$ y $n-2$. Adicionalmente, la Figura 6 ilustra cuatro bandas de frecuencia o espectrales que se etiquetan por $m-2$, $m-1$, m y $m+1$. La Figura 6 muestra dentro de cada intervalo de tiempo-frecuencia recuadros, que representan tuplas de muestras a codificarse o decodificarse. Se ilustran tres diferentes tipos de tuplas en la Figura 6, en la que los recuadros que tienen un borde discontinuo de líneas o de puntos indican tuplas restantes a codificarse o decodificarse, recuadros blancos que tienen un borde continuo indican tuplas previamente codificadas o decodificadas y recuadros grises con un borde continuo indican tuplas previamente codificadas/decodificadas, que se usan para determinar el contexto para la tupla actual a codificarse o decodificarse.

Obsérvese que los segmentos anteriores y actuales referenciados en las realizaciones anteriormente descritas pueden corresponder a una tupla en la presente realización, en otras palabras, los segmentos pueden procesarse a nivel de bandas en el dominio de la frecuencia o espectral. Como se ilustra en la Figura 6, las tuplas o segmentos en las cercanías de una tupla actual (es decir en el dominio de tiempo y de la frecuencia o espectral) pueden tenerse en cuenta para derivar un contexto. Las tablas de frecuencia acumulativa pueden a continuación usarse por el codificador aritmético para generar un código binario de longitud variable. El codificador aritmético puede producir un código binario para un conjunto dado de símbolos y sus respectivas probabilidades. El código binario puede generarse mapeando un intervalo de probabilidad, donde radica el conjunto de símbolos, a una palabra de código. El codificador aritmético puede corresponder al codificador por entropía 120, respectivamente el decodificador por entropía 220 en las realizaciones anteriormente descritas.

En la presente realización la codificación aritmética basada en contexto puede llevarse a cabo basándose en 4-tuplas (es decir en cuatro índices de coeficientes espectrales), que también se etiquetan $q(n,m)$, que representan los coeficientes espectrales después de la cuantificación, que son vecinos en el dominio de la frecuencia o espectral y que se codifican por entropía en una etapa. De acuerdo con la descripción anterior, la codificación puede llevarse a cabo basándose en el contexto de codificación. Como se indica en la Figura 6, adicionalmente a la 4-tupla, que se codifica (es decir el segmento actual) se tienen en cuenta 4-tuplas previamente codificadas para derivar el contexto. Estas cuatro 4-tuplas determinan el contexto y son anteriores en el dominio de la frecuencia y/o anteriores en el del tiempo.

La Figura 7a muestra un diagrama de flujo de un codificador aritmético dependiente del contexto de USAC (USAC = Codificador de Voz y Audio Universal) para el esquema de codificación de coeficientes espectrales. El proceso de codificación depende de la 4-tupla actual más el contexto, donde el contexto se usa para seleccionar la distribución de probabilidad del codificador aritmético y para predecir la amplitud de los coeficientes espectrales. En la Figura 7a el recuadro 705 representa determinación de contexto, que es basándose en t_0 , t_1 , t_2 y t_3 que corresponde a $q(n-1, m)$, $q(n, m-1)$, $q(n-1, m-1)$ y $q(n-1, m+1)$, que son los recuadros grises con bordes continuos en la Figura 6.

En general, en las realizaciones el codificador por entropía puede adaptarse para codificar el segmento actual en unidades de una 4-tupla de coeficientes espectrales y para predecir un intervalo de amplitud de la 4-tupla basándose en el contexto de codificación.

En la presente realización el esquema de codificación comprende varias etapas. En primer lugar, la palabra de código literal se codifica usando un codificador aritmético y una distribución de probabilidad específica. La palabra de código representa cuatro coeficientes espectrales vecinos (a,b,c,d), sin embargo, cada uno de a, b, c, d está limitado en el intervalo:

$$-5 < a,b,c,d < 4.$$

En general, en las realizaciones el codificador por entropía 120 puede adaptarse para dividir la 4-tupla en un factor predeterminado tan a menudo como sea necesario para adaptarse a un resultado de la división en el intervalo previsto o en un intervalo predeterminado y para codificar un número de divisiones necesarias, un resto de división y el resultado de la división cuando la 4-tupla no radica en el intervalo predeterminado, y para codificar un resto de división y el resultado de la división de otra manera.

A continuación, si el término (a,b,c,d), es decir cualquier coeficiente a, b, c, d, supera el intervalo dado en esta realización, esto puede considerarse en general dividiendo (a,b,c,d) tan a menudo por un factor (por ejemplo 2 o 4) como sea necesario, para ajustar la palabra de código resultante en el intervalo dado. La división por un factor de 2 corresponde a un desplazamiento binario al lado a la derecha, es decir $(a,b,c,d) \gg 1$. Esta disminución se hace por una representación de número entero, es decir la información puede perderse. Los bits menos significativos, que pueden perderse por el desplazamiento a la derecha, se almacenan y más tarde se codifican usando el codificador aritmético y una distribución de probabilidad uniforme. El proceso de desplazamiento a la derecha se lleva a cabo por todos los cuatro coeficientes espectrales (a,b,c,d).

En realizaciones generales, el codificador por entropía 120 puede adaptarse para codificar el resultado de la división o la 4-tupla usando un índice de grupo ng , el índice de grupo ng que hace referencia a un grupo de una o más palabras de código para las que una distribución de probabilidad está basada en el contexto de codificación, y un índice de elemento ne en el caso de que el grupo comprenda más de una palabra de código, el índice de elemento ne hace referencia a una palabra de código dentro del grupo y el índice de elemento puede suponerse que está distribuido uniformemente, y para codificar el número de divisiones por un número de símbolos de escape, siendo un símbolo de escape un índice de grupo específico ng únicamente usado para indicar una división y para codificar los restos de las divisiones basándose en una distribución uniforme usando una regla de codificación aritmética. El codificador por entropía 120 puede adaptarse para codificar una secuencia de símbolos en el flujo de audio codificado usando un alfabeto de símbolos que comprende el símbolo de escape, y símbolos de grupo que corresponden a un conjunto de índices de grupo disponibles, un alfabeto de símbolos que comprende los correspondientes índices de elemento, y un alfabeto de símbolos que comprende los diferentes valores de los restos.

En la realización de la Figura 7a, la distribución de probabilidad para codificar la palabra de código literal y también una estimación del número de etapas de intervalo-reducción puede derivarse a partir del contexto. Por ejemplo, todas las palabras de código, en un total $8^4 = 4096$, abarcan en total 544 grupos, que consisten en uno o más elementos. La palabra de código puede representarse en el flujo de bits como el índice de grupo ng y el elemento de grupo ne . Ambos valores pueden codificarse usando el codificador aritmético, usando ciertas distribuciones de probabilidad. En una realización la distribución de probabilidad para ng puede derivarse a partir del contexto, mientras que la distribución de probabilidad para ne puede suponerse que es uniforme. Una combinación de ng y ne

puede identificar de manera inequívoca una palabra de código. El resto de la división, es decir los planos de bits desplazados, puede suponerse que están distribuidos uniformemente también.

5 En la Figura 7a, en la etapa 710, se proporciona la 4-tupla $q(n,m)$, que es (a,b,c,d) o el segmento actual y se inicia un parámetro lev estableciéndolo a 0.

En la etapa 715 desde el contexto, se estima el intervalo de (a,b,c,d) . De acuerdo con esta estimación, (a,b,c,d) puede reducirse por $lev0$ niveles, es decir dividirse por un factor de 2^{lev0} . Los $lev0$ planos de bits menos significativos se almacenan para uso posterior en la etapa 750.

10 En la etapa 720 se comprueba si (a,b,c,d) supera el intervalo dado y en caso afirmativo, el intervalo de (a,b,c,d) se reduce en un factor de 4 en la etapa 725. En otras palabras, en la etapa 725 (a,b,c,d) se desplazan en 2 a la derecha y los planos de bits eliminados se almacenan para uso posterior en la etapa 750.

15 Para indicar esta etapa de reducción, ng se establece a 544 en la etapa 730, es decir $ng = 544$ sirve como una palabra de código de escape. Esta palabra de código se escribe a continuación en el flujo de bits en la etapa 755, donde para derivar la palabra de código en la etapa 730 se usa un codificador aritmético con una distribución de probabilidad derivada del contexto. En caso de que esta etapa de reducción se aplicara la primera vez, es decir si $lev=lev0$, el contexto se adapta ligeramente. En caso de que la etapa de reducción se aplique más de una vez, el contexto se descarta y se usa una distribución por defecto más adelante. El proceso a continuación continúa con la etapa 720.

25 Si en la etapa 720 se detecta una coincidencia para el intervalo, más específicamente si (a,b,c,d) coincide con la condición de intervalo, (a,b,c,d) se mapea a un grupo ng , y, si es aplicable, el índice de elemento de grupo ne . Este mapeo es de manera inequívoca, es decir (a,b,c,d) puede derivarse a partir de ng y ne . El índice de grupo ng se codifica a continuación por el codificador aritmético, usando una distribución de probabilidad llegada para el contexto adaptado/descartado en la etapa 735. El índice de grupo ng se inserta a continuación en el flujo de bits en la etapa 755. En una siguiente etapa 740, se comprueba si el número de elementos en el grupo es mayor que 1. Si es necesario, es decir si el grupo indexado por ng consiste en más de un elemento, el índice de elemento de grupo ne se codifica por el codificador aritmético en la etapa 745, suponiendo una distribución de probabilidad uniforme en la presente realización.

30 Siguiendo la etapa 745, el índice de grupo de elemento ne se inserta en el flujo de bits en la etapa 755. Finalmente, en la etapa 750, todos los planos de bits almacenados se codifican usando el codificador aritmético, suponiendo una distribución de probabilidad uniforme. Los planos de bits almacenados codificados se insertan a continuación en el flujo de bits en la etapa 755.

40 En las realizaciones el decodificador por entropía 220 puede adaptarse para decodificar un índice de grupo ng a partir del flujo de audio codificado basándose en una distribución de probabilidad derivada del contexto de codificación, en la que el índice de grupo ng representa un grupo de una o más palabras de código, y para, basándose en una distribución de probabilidad uniforme, decodificar un índice de elemento ne a partir del flujo de audio codificado si el índice de grupo ng indica un grupo que comprende más de una palabra de código, y para derivar una 4-tupla de coeficientes espectrales del segmento actual basándose en el índice de grupo ng y el índice de elemento ne , obteniendo de esta manera la representación de dominio espectral en tuplas de coeficientes espectrales.

45 En las realizaciones el decodificador por entropía 220 puede adaptarse para decodificar una secuencia de símbolos desde el flujo de audio codificado basándose en la distribución de probabilidad derivada desde el contexto de codificación usando un alfabeto de símbolo que comprende un símbolo de escape y símbolos de grupo que corresponden a un conjunto de índices de grupo disponibles ng , para derivar una 4-tupla preliminar de coeficientes espectrales basándose en un índice de grupo disponible ng al que corresponde un símbolo de grupo de la secuencia de símbolos, y basándose en el índice de elemento ne , y para multiplicar la 4-tupla preliminar con un factor que depende de un número de símbolos de escape en la secuencia de símbolos para obtener la tupla de coeficientes espectrales.

50 El decodificador por entropía 220 puede adaptarse adicionalmente para decodificar un resto del flujo de audio codificado basándose en una distribución de probabilidad uniforme usando una regla de codificación aritmética y para añadir el resto a la 4-tupla preliminar multiplicada para obtener la 4-tupla de coeficientes espectrales.

55 El decodificador por entropía 220 puede adaptarse para multiplicar la 4-tupla por un factor predeterminado tan a menudo como un símbolo de escape se decodifica desde el flujo de audio codificado, siendo un símbolo de escape un índice de grupo específico ng únicamente usado para indicar una multiplicación, y para decodificar un resto del flujo de audio codificado basándose en una distribución de probabilidad uniforme usando una regla de codificación aritmética, el decodificador por entropía 220 puede adaptarse adicionalmente para añadir el resto a la 4-tupla

multiplicada para obtener el segmento actual.

A continuación se describirá una realización de un esquema de decodificación de codificador aritmético dependiente del contexto USAC. Correspondiente a la realización anterior del esquema de codificación, se consideran 4-tuplas que corresponden a coeficientes espectrales cuantificados, que se codifican de manera sin ruido. Adicionalmente, se supone que las 4-tuplas se transmiten empezando desde la frecuencia o coeficiente espectral más inferior y progresando a la frecuencia o coeficiente espectral más superior. Los coeficientes pueden corresponder, por ejemplo, a coeficientes de AAC, que se almacenan en una serie, y el orden de transmisión de las palabras de codificación sin ruido se supone que es de manera que cuando se decodifican en el orden recibido y se almacenan en la serie, bin es el índice de incremento más rápido y g es el índice de incremento más lento. En una palabra de código, el orden de decodificación es a,b,c,d.

La Figura 7b ilustra el procedimiento de actualización de contexto general de una realización. Los detalles sobre la adaptación de contexto de acuerdo con un mecanismo de predicción de profundidad de bit se considerarán en la presente realización. La Figura 7b muestra un plano 760 que presenta el posible intervalo de una 4-tupla (a,b,c,d) en términos de planos de bits. La profundidad de bits, es decir el número de planos de bits necesarios para representar una 4-tupla, puede predecirse por el contexto de la 4-tupla actual por medio del cálculo de la variable denominada lev0, que también se indica en la Figura 7b. La 4-tupla se divide a continuación por $2^{\text{lev}0}$, es decir se eliminan lev=lev0 bit planos y se almacenan para uso posterior de acuerdo con la etapa anteriormente descrita 715.

Si la 4-tupla está en el intervalo $-5 < a,b,c,d < 4$, la profundidad de bits prevista lev0 se predijo correctamente o se sobreestimó. La 4-tupla puede a continuación decodificarse por el índice de grupo ng, el índice de elemento ne y los lev planos de bits restantes, en relación con la descripción anterior. La codificación de la 4-tupla actual se finaliza a continuación. La codificación del índice de elemento ne se indica en la Figura 7b por la distribución de probabilidad uniforme 762, que se usará a continuación siempre para codificar índices de elemento, donde en la Figura 7b el parámetro r representa el resto de la 4-tupla después de la división y p(r) representa la función de densidad de probabilidad uniforme correspondiente.

Si la 4-tupla no está en el intervalo $-5 < a,b,c,d < 4$ la predicción basada en el contexto de codificación 764 es demasiado baja, se codifica 766 un símbolo de escape (ng=544) y la 4-tupla se divide por 4 y lev se incrementa por 2, de acuerdo con la etapa 730 en la Figura 7a. El contexto está adaptado como sigue. Si lev==lev0+2 el contexto está ligeramente adaptado, que corresponde a 768 en la Figura 7b. Puede establecerse una bandera en la representación de contexto, t, y se usará a continuación un nuevo modelo de distribución de probabilidad para codificar los símbolos futuros ng.

Si lev>lev0+2 se codifica otro símbolo de escape de acuerdo con la etapa 770 en la Figura 7b el contexto se resetea completamente, consúltase 772, descartado como en la etapa 730 en la Figura 7a, respectivamente. No se usa adaptación de contexto adicional puesto que se considera que no es relevante para la codificación de 4-tupla actual. El modelo de probabilidad por defecto, el usado cuando no hay disponible contexto, se usa a continuación para los futuros ng símbolos, que se indica por las etapas 774 y 776 en la Figura 7b. El proceso se repite a continuación para las otras tuplas.

Para resumir, la adaptación de contexto es un mecanismo para reducir el significado de contexto en la codificación adaptativa de contexto. La adaptación de contexto puede activarse cuando el lev0 previsto y el lev real no coinciden. Se detecta fácilmente por el número de símbolos de escape codificados (ng=544), consúltase 766 y 770 en la Figura 7b, y puede llevarse a cabo por lo tanto en el decodificador de una manera similar también.

La adaptación de contexto puede realizarse activando una bandera en la representación de estado de contexto t. El valor t se calcula por la función get_state(), como lev0, usando el contexto derivado de la trama o segmento anterior y/o actual de la 4-tupla actual, que se almacena en una tabla q[[]]. El estado del contexto puede representarse, por ejemplo, por 24 bits. Existen 1905800 posibles estados en una realización. Estos estados pueden representarse por únicamente 21 bits. Los bits de orden 23 y 24 de t se reservan para adaptar el estado de contexto. De acuerdo con los valores de los bits de orden 23 y 24, get_pk() emitirá diferentes modelos de distribución de probabilidad. En una realización, el bit de orden 23 de t puede establecerse a uno cuando la 4-tupla se divide por 4 después de dividirse previamente por lev0, es decir lev==lev0+2.

En consecuencia, el mapeo entre el estado de contexto t y el modelo de distribución de probabilidad pki, es diferente para lev==lev0+2 que para lev==lev0. El mapeo entre el estado de contexto t y el modelo pki se pre-define durante una fase de entrenamiento realizando optimizaciones en las estadísticas globales de la secuencia de entrenamiento. Cuando lev>lev0+2, el contexto y t pueden establecerse a cero. Get_pk() emite a continuación el modelo de distribución de probabilidad por defecto pki, que corresponde a t=0.

A continuación se describirán los detalles de un mapeo de contexto en una realización. El mapeo de contexto es la primera operación realizada en codificación adaptativa de contexto después del reseteo eventual del contexto en la

presente realización. Se hace en dos etapas.

En primer lugar, antes de la codificación, la tabla de contexto `qs[]` de tamaño `previous_lg/4`, grabada en la trama anterior, se mapea en una tabla de contexto `q[0][]` de tamaño `lg/4` que corresponde al tamaño de la trama actual. El mapeo se realiza en la función `arith_map_context`, que se ejemplifica por el siguiente pseudo-código:

```

/*variable de entrada*/
lg/4 /*número de 4-tuplas*/
arith_map_context()
10 {
    v=w=0
    if(core_mode==1){
        q[0][v++]=qs[w++];
    }
15    ratio= ((float)previous_lg)/((float)lg);
    for(j=0; j<lg/4; j++){
        k = (int) ((float)) ((j)*ratio);
        q[0][v++] = qs[w+k];
    }
20    if(core_mode==0) {
        q[0][lg/4]=qs[previous_lg/4];
    }
    q[0][lg/4+1]=qs[previous_lg/4+1];
    previous_lg=lg;
25 }

```

Como puede observarse a partir del pseudo-código, el mapeo puede no ser exactamente el mismo para todas las estrategias de codificación. En la presente realización, el mapeo se diferencia cuando se usa (`core_mode==0`) AAC (Codificación de Audio Avanzada) para un coeficiente de cuando se usa (`core_mode==1`) TCX (Codificación basada en Transformada). Una de las diferencias proviene de cómo se manejan los límites de las tablas. En AAC, el mapeo puede iniciarse desde el índice 0 (primer valor en la tabla), mientras que para TCX puede iniciarse desde el índice 1 (segundo valor en la tabla) conociendo que el primer valor siempre se establece a 'desconocido' (estado específico usado para resetear el contexto). La relación de `previous_lg` sobre `lg` determinará el orden del muestreo ascendente (`ratio<1`) o el muestreo descendente (`ratio>1`), que se realizará en la presente realización. La Figura 7c ilustra el caso de TCX cuando se convierte desde una tabla de contexto grabada de tamaño 1024/4, consúltese el lado a la izquierda 780 de la Figura 7c, a un tamaño de 512/4, consúltese el lado a la derecha 782 de la Figura 7c. Puede observarse, que mientras para la tabla de contexto actual 782 se usan incrementos graduales de 1, para la tabla de contexto grabada 780 se usan incrementos graduales de la relación anteriormente descrita.

La Figura 7c ilustra el procedimiento de actualización de contexto de una realización para cambios de resolución. Una vez que se realiza el mapeo, se realiza la codificación adaptativa de contexto. Al final de la codificación, los elementos de la trama actual se graban en la tabla `qs[]` para la siguiente trama. Puede realizarse en `arith_update_context()`, que se ejemplifica por el siguiente pseudo-código:

```

/*variables de entrada*/
a,b,c,d /* valor de la 4-tupla decodificada*/
i /*el índice de la 4-tupla para decodificar en el vector*/
lg/4 /*número de 4-tuplas*/
arith_update_context()
50 {
    q[1][1+i].a=a;
    q[1][1+i].b=b;
    q[1][1+i].c=c;
    q[1][1+i].d=d;
55    if ((a<-4) || (a>=4) || (b<-4) || (b>=4) || (c<-4) || (c>=4) || (d<-4) || (d>=4)) {
        q[1][1+i].v =1024;
    }
    else q[1][1+i].v=egroups[4+a][4+b][4+c][4+d];
    if(i==lg/4 && core_mode==1) {
60        qs[0]=q[1][0];
        ratio= ((float) lg)/((float)1024);
        for(j=0; j<256; j++) {
            k = (int) ((float) j*ratio);
            qs[1+k] = q[1][1+j] ;
65        }
    }
}

```

```

        qs[previous_lg/4+1] = q[1][lg/4+1];
previous_lg = 1024;
    }
    if(i==lg/4 && core_mode==0) {
5      for(j=0; j<258; j++) {
        qs[j] = q[1][k];
      }
      previous_lg = min(1024,lg);
10    }
  }

```

En la presente realización, la grabación se hace de manera diferente de acuerdo con el codificador principal (AAC o TCX). En TCX el contexto siempre se graba en una tabla qs [] de 1024/4 valores. Este mapeo adicional puede hacerse debido a la decisión de bucle cerrado del AMR-WB+ (Código de Banda Ancha de Tasa Múltiple Adaptativa).

15 En la decisión de bucle cerrado son necesarios varios procedimientos de copia de los estados del codificador para probar cada posible combinación de TCX y ACELP (Predicción Lineal con Excitación Aritmética por Código). La copia de estado es más fácil de implementar cuando todos los modos de TCX comparten el mismo tamaño para la tabla qs []. A continuación se usa un mapeo para convertir sistemáticamente de lg/4 a 1024/4. Por otra parte, AAC graba únicamente el contexto y no realiza mapeo durante esta fase.

20 La Figura 8 ilustra un diagrama de flujo de la realización del esquema de decodificación. En la etapa 805, que corresponde a la etapa 705, el contexto se deriva basándose en t0, t1, t2 y t3. En la etapa 810, el primer nivel de reducción lev0 se estima a partir del contexto, y la variable lev se establece a lev0. En la siguiente etapa 815, el grupo ng se lee a partir del flujo de bits y la distribución de probabilidad para decodificar ng se deriva a partir del contexto. En la etapa 815, el grupo ng puede decodificarse a continuación a partir del flujo de bits.

25 En la etapa 820 se determina si ng equivale 544, que corresponde al valor de escape. En caso afirmativo, la variable lev puede aumentarse en 2 antes de volver a la etapa 815. En caso de que esta rama se use la primera vez, es decir, si lev==lev0, el contexto puede adaptarse en consecuencia a la distribución de probabilidad respectivamente, descartarse respectivamente si la rama no se usa la primera vez, en relación con el mecanismo de adaptación de contexto anteriormente descrito, consúltase las Figuras 7b y 7c. En caso de que el índice de grupo ng no sea igual a 544 en la etapa 820, en una siguiente etapa 825 se determina si el número de elementos en un grupo es mayor que 1, y en caso afirmativo, en la etapa 830, se lee el elemento de grupo ne y se decodifica a partir del flujo de bits suponiendo una distribución de probabilidad uniforme. El índice de elemento ne se deriva a partir del flujo de bits usando codificación aritmética y una distribución de probabilidad uniforme.

En la etapa 835 la palabra de código literal (a,b,c,d) se deriva a partir de ng y ne, por ejemplo, por un proceso de búsqueda en las tablas, por ejemplo, haciendo referencia a dgroups [ng] y acod_ne[ne].

40 En la etapa 840 para todos los lev planos de bits faltantes, los planos se leen a partir del flujo de bits usando codificación aritmética y suponiendo una distribución de probabilidad uniforme. Los planos de bits pueden a continuación anexarse a (a,b,c,d) desplazando (a,b,c,d) a la izquierda y añadiendo el plano de bits bp: ((a,b,c,d)<<=1)1=bp. Este proceso puede repetirse lev veces.

45 Finalmente en la etapa 845 puede proporcionarse la 4-tupla q(n,m), es decir(a,b,c,d).

A continuación se proporcionan pseudo códigos y detalles de implementación de acuerdo con una realización. Se usarán las siguientes definiciones.

50 (a,b,c,d) 4-tupla para decodificar

ng índice de grupo del plano de 2 bits más significativos de la 4-tupla, donde $0 \leq ng \leq 544$. El último valor 544 corresponde al símbolo de escape, ARITH_ESCAPE.

55 ne índice de elemento en un grupo. ne radica entre 0 y el cardinal de cada grupo mm. El número máximo de elementos dentro de un grupo es 73.

lev nivel de los planos de bits restantes. Corresponde al número de planos de bits menos significativos que el plano de 2 bits más significativos.

60 egroups [a][b][c][d] tabla de índice de grupo. Permite mapear el plano de 2 bits más significativos de la 4-tupla (a,b,c,d) en los 544 grupos.

mm cardinal del grupo

og compensación del grupo

5 dggroups[] mapea el índice de grupo ng al cardinal de cada grupo mm (primeros 8 bits) y la compensación del grupo op en dgvectors[] (últimos 8 bits).

dgvectors[] mapea la compensación del grupo og y el índice del elemento ne al plano de 2 bits más significativos de la 4-tupla (a,b,c,d).

10 arith_cf_ng_hash[] tabla de función de troceo que mapea el estado de contexto a un índice de tabla de frecuencias acumulativas pki.

arith_cf_ng[pki][545] Modelos de las frecuencias acumulativas para el índice de símbolo de grupo ng.

15 arith_cf_ne[] frecuencias acumulativas para el símbolo de índice de elemento ne.

r plano de bits de la 4-tupla menos significativo que el plano de 2 bits más significativos.

20 arith_cf_r [] frecuencias acumulativas para el símbolo r de planos de bits menos significativos

A continuación se considera en primer lugar el proceso de decodificación. Se codifican sin ruido coeficientes espectrales cuantificados de cuatro tuplas y se transmiten empezando desde el coeficiente espectral o de frecuencia más baja y progresando al coeficiente espectral o de frecuencia más alta. Los coeficientes a partir de AAC se almacenan en la serie x_ac_quant[g][win][sfb][bin], y el orden de transmisión de las palabras de codificación sin ruido es de manera que cuando se decodifican en el orden recibido y se almacenan en la serie, bin es el índice de incremento más rápido y g es el índice de incremento más lento. En una palabra de código el orden de decodificación es a, b, c, d. El coeficiente desde la TCX se almacena directamente en la serie x_tcx_invquant[win][bin], y el orden de la transmisión de las palabras de codificación sin ruido es de manera que cuando se decodifican en el orden recibido y se almacenan en la serie, bin es el índice de incremento más rápido y win es el índice de incremento más lento. En una palabra de código el orden de decodificación es a, b, c, d. En primer lugar, la bandera arith_reset_flag determina si el contexto debe resetearse. Si la bandera es VERDADERO se solicita la siguiente función:

```

35  *variables globales*/
    q[2][290] /*contexto actual */
    qs[258] /*contexto pasado*/
    previous_lg /*número de 4-tuplas del contexto pasado*/
    arith_reset_context()
    {
40      for(i=0;i<258;i++){
          qs[i].a=0; qs [i].b=0; qs[i].c=0;qs [i].d=0
          qs[i].v=-1;
        }
        for(i=0;i<290;i++){
45          q[0][i].a=0; q[0][i].b=0; q[0][i].c=0;
          q[0][i].d=0
          q[0][i].v=-1;
          q[1][i].a=0; q[1][i].b=0; q[1][i].c=0;
          q[1][i].d=0
50          q[1][i].v=-1;
        }
        previous_lg=256;
    }

```

55 De otra manera, cuando arith_reset_flag es FALSA, se realiza un mapeo entre el contexto pasado y el contexto actual:

```

/*variable de entrada*/
lg /*número de 4-tuplas*/
60 arith_map_context(lg)
    {
        v=w=0
        if(core_mode==1){
            q[0][v++]=qs[w++];
65     }
    }

```

```

ratio= ((float)previous_lg)/((float)lg);
for(j=0; j<lg; j++){
    k = (int) ((float)) ((j)*ratio);
    q[0][v++] = qs[w+k];
5  }
    if(core_mode==0){
        q[0][lg]=qs[previous_lg];
    }
    q[0][lg+1]=qs[previous_lg+1];
10  previous_lg=lg;
    }

```

El decodificador sin ruido emite 4-tuplas de coeficientes espectrales cuantificados con signo. En primer lugar se calcula el estado del contexto basándose en los cuatro grupos previamente decodificados que rodean la 4-tupla para decodificar. El estado del contexto se proporciona por la función arith_get_context():

```

/* variables de entrada*/
i /*el índice de la 4-tupla para decodificar en el vector*/
arith_get_context(i,)
20  {
    t0=q[0][1+i].v+1;
    t1=q[1][1+i-1].v+1;
    t2=q[0][1+i-1].v+1;
    T3=q[0][1+i+1].v+1;
25  if ( (t0<10) && (t1<10) && (t2<10) && (t3<10) ){
        if (t2>1) t2=2;
        if (t3>1) t3=2;
        return 3*(3*(3*(3*(10*(10*t0+t1))+t2)+t3)));
    }
30  if ((t0<34) && (t1<34) && (t2<34) && (t3<34)){
        if ((t2>1) && (t2<10)) t2=2; else if (t2>=10) t2=3;
        if ((t3>1) && (t3<10)) t3=2; else if (t3>=10) t3=3;
        return 252000+4*(4*(34*(34*t0+t1))+t2)+t3;
    }
35  if ((t0<90) && (t1<90)) return 880864+90*(90*t0+t1);
    if ((t0<544) && (t1<544)) return 1609864 + 544*t0+t1;
    if (t0>1)
    {
        a0=q[0][i].a;
        b0=q[0][i].b;
        c0=q[0][i].c;
        d0=q[0][i].d;
40    }
    else a0=b0=c0=d0=0;
45  if (t1>1)
    {
        a1=q[1][i-1].a;
        b1=q[1][i-1].b;
        c1=q[1][i-1].c;
        d1=q[1][i-1].d;
50    }
    else a1=b1=c1=d1=0;
    l=0;
    do
55  {
        a0>>=1;
        b0>>=1;
        c0>>=1;
        d0>>=1;
60    a1>>=1;
        b1>>=1;
        c1>>=1;
        d1>>=1;
        l++;
65  }
    }

```

```

while ((a0<=-4) || (a0>=4) || (b0<=-4) || (b0>=4) || (c0<=-4) || (c0>=4) || (d0<=-4) ||
(d0>=4) || (a1<=-4) || (a1>=4) || (b1<=-4) || (b1>=4) || (c1<=-4) || (c1>=4) || (d1<=-
4) || (d1>=4) );
if (t0>1) t0=1+(egroups[4+a0][4+b0][4+c0][4+d0] >> 16);
5 if (t1>1) t1=1+(egroups[4+a1][4+b1][4+c1][4+d1] >> 16);
return 1609864 + ((1<<24) | (544*t0+t1));
}

```

Una vez que se conoce el estado el grupo al que pertenece el plano de 2 bits más significativos de 4-tupla se decodifica usando arith_decode() alimentada con la tabla de frecuencias acumulativas apropiada que corresponde al estado de contexto. La correspondencia se realiza mediante la función arith_get_pk():

```

/*variable de entrada*/
s /* Estado del contexto*/
15 arith_get_pk(s)
{
    psci[28] = {
        247,248,249,250,251,252,253,254,254,0,254,254,254,255,250,215,
        215,70,70,123,123,123,123,3,67,78,82,152
20    };
    register unsigned char *p;
    register unsigned long i,j;
    i=123*s;
    for (;;)
25    {
        j=arith_cf_nq_hash[i&32767];
        if (j==0xFFFFFFFFul) break;
        if ((j>>8)==s) return j&255;
        i++;
30    }
    p=psci+7*(s>>22);
    j= s & 4194303;
    if (j<436961)
    {
        if (j<252001) return p[(j<243001)?0:1]; else return
        p[(j<288993)?2:3];
    }
    else
    {
40        if (j<1609865) return p[(j<880865)?4:5]; else return p[6];
    }
}

```

A continuación se solicita la función arith_decode() con la tabla de frecuencias acumulativa que corresponde al índice devuelto por arith_get_pk(). El codificador aritmético es una implementación de número entero que genera la etiqueta con escalado. El siguiente pseudo-código de C describe el algoritmo usado.

```

/*funciones de ayuda*/
bool arith_first_symbol(void);
50 /* Devolver VERDADERO si es el primer símbolo de la secuencia, FALSO de lo
    contrario*/
Ushort arith_get_next_bit(void);
/* Obtener el siguiente bit del flujo de bits*/
/* variables globales */
55 low
high
value
/* variables de entrada */
cum_freq[];
60 cfl;
arith_decode()
{
    if(arith_first_symbol())
    {
65        value = 0;
    }
}

```

```

    for (i=1; i<=20; i++)
    {
        value = (val<<1) | arith_get_next_bit();
    }
5    low=0;
    high=1048575;
}
range = high-low+1;
cum = (((int64) (value-low+1))<<16)-((int64) 1))/((int64) range);
10    p = cum_freq-1;
    do
    {
        q=p+(cfl>>1);
        if ( *q > cum ) { p=q; cfl++; }
15    cfl>=1;
    }
    while ( cfl>1 );
    symbol = p-cum_freq+1;
    if(symbol)
20    high = low + (((int64)range)*((int64)cum_freq[symbol-1]))>>16 - 1;
    low += (((int64) range)* ((int64) cum_freq[symbol]))>>16;
    for (; ;)
    {
        if ( high<524286) { }
25    else if ( low>=524286)
        {
            value -=524286;
            low -=524286;
            high -=524286;
30    }
        else if ( low>=262143 && high<786429)
        {
            value -= 262143;
            low -= 262143;
35    high -= 262143;
        }
        else break;
        low += low;
        high += high+1;
40    value = (value<<1) | arith_get_next_bit();
    }
    return symbol;
}

```

45 Aunque el índice de grupo decodificado ng es el símbolo de escape, ARITH_ESCAPE, se decodifica un índice de grupo adicional ng y la variable lev se incrementa en dos. Una vez que el índice de grupo decodificado no es el símbolo de escape, ARITH_ESCAPE, el número de elementos, mm, en el grupo y la compensación de grupo, og, se deducen buscando la tabla dgroups[]:

```

50    mm=dgroups[nq]&255

    og = dgroups[nq]>>8

```

55 El índice de elemento ne se decodifica a continuación solicitando arith_decode() con la tabla de frecuencias acumulativa (arith_cf_ne+((mm*(mm-1))>>1)[]. Una vez que se decodifica el elemento puede derivarse el plano de 2 bits más significativos de la 4-tupla con la tabla dgvector[]:

```

    a=dgvector[4*og+ne]
60    b=dgvector[4*og+ne]+1

    c=dgvector[4*og+ne]+2

    d=dgvector[4*og+ne]+3
65

```

Los restantes planos de bits se decodifican a continuación a partir del nivel más significativo al menos significativo solicitando lev veces arith_decode() con la tabla de frecuencias acumulativa arith_cf_r []. El plano de bits decodificado r permite perfeccionar la 4-tupla de decodificación mediante la siguiente manera:

```

5          a = (a<<1) | (r&1)

          b = (b<<1) | ((r>>1)&1)

          c = (c<<1) | ((r>>2)&1)
10         d = (d<<1) | ((r>>3)&1)

```

Una vez que la 4-tupla (a,b,c,d) está completamente decodificada, se actualizan las tablas de contexto q y qs solicitando la función arith_update_context().

```

15 arith_update_context(a,b,c,d,i,lg)
{
    q[1][1+i].a=a;
    q[1][1+i].b=b;
20    q[1][1+i].c=c;
    q[1][1+i].d=d;
    if ( (a<-4) || (a>=4) || (b<-4) || (b>=4) || (c<-4)

        || (c>=4) || (d<-4) || (d>=4) )
25    q[1][1+i].v =1024;
    else q[1][1+i].v=egroups[4+a][4+b][4+c][4+d];
    if(i==lg && core_mode==1){
        qs[0]=q[1][0];
        ratio= ((float) lg)/((float)256);
30        for(j=0; j<256; j++){
            k = (int) ((float)) ((j)*ratio);
            qs[1+k] = q[1][1+j] ;
        }
        qs[previous_lg+1]=q[1][lg+1];
35        previous_lg=256;
    }
    if(i==lg && core_mode==0){
        for(j=0; j<258; j++){
            qs[j] = q[1][k];
40        }
        previous_lg=min(1024,lg);
    }
}

```

Dependiendo de ciertos requisitos de implementación de los métodos inventivos, los métodos inventivos pueden implementarse en hardware o en software. La implementación puede formarse usando un medio de almacenamiento digital, en particular, un disco, un DVD, o un CD, que tiene señales de control electrónicamente legibles en el mismo, que cooperan con el ordenador programable, de manera que se realizan los métodos inventivos. En general, la presente invención es por lo tanto un producto de programa informático con un código de programa para un soporte legible por máquina, siendo el código de programa operativo para realizar los métodos inventivos cuando el programa informático se ejecuta en un ordenador. En otras palabras, los métodos inventivos son, por lo tanto, un programa informático que tiene un código de programa para realizar al menos uno de los métodos inventivos cuando el programa informático se ejecuta en un ordenador.

REIVINDICACIONES

1. Un codificador de audio (100) para codificar una secuencia de segmentos de coeficientes, siendo los segmentos subsiguientes entre sí en el tiempo, comprendiendo el codificador de audio (100)
5 un medio para proporcionar la secuencia de segmentos de coeficientes a partir de un flujo de audio que representa una señal de audio muestreada mediante el uso de diferentes longitudes de transformación de manera que los segmentos de coeficientes para los que se utilizan diferentes longitudes de transformación, representan espectralmente la señal de audio muestreada a diferentes resoluciones de frecuencia y de manera que los
10 segmentos sean de 1024 y 128 coeficientes espectrales, respectivamente;

un procesador (110) para derivar un contexto de codificación para un coeficiente codificado actualmente de un segmento actual en base a un coeficiente codificado previamente de un segmento anterior, correspondiendo los
15 segmentos anterior y actual a diferentes resoluciones de frecuencia y diferentes longitudes de transformación, respectivamente; y

un codificador por entropía (120) para codificar por entropía el coeficiente actual en función del contexto de codificación para obtener un flujo de audio codificado,

20 en donde el procesador (110) y el codificador por entropía (120) están configurados para funcionar en función de un muestreo ascendente de los coeficientes espectrales del segmento anterior, cuando el segmento anterior muestra una resolución espectral más gruesa que el segmento actual.
2. El codificador de audio (100) de la reivindicación 1, en el que el codificador por entropía (120) está adaptado para
25 codificar el segmento actual en unidades de una tupla de coeficientes espectrales y para prever un intervalo de la tupla basándose en el contexto de codificación.
3. El codificador de audio (100) de la reivindicación 2, en el que el codificador por entropía (120) está adaptado para
30 dividir la tupla por un factor predeterminado tan a menudo como sea necesario para adaptarse a un resultado de la división en un intervalo predeterminado y para codificar un número de divisiones necesarias, un resto de división y el resultado de la división cuando la tupla no radica en el intervalo previsto.
4. El codificador de audio (100) de la reivindicación 3, en el que el codificador por entropía (120) está adaptado para
35 codificar el resultado de la división o la tupla usando un índice de grupo, haciendo referencia el índice de grupo a un grupo de una o más palabras de código para el que una distribución de probabilidad está basada en el contexto de codificación, y, basándose en una distribución de probabilidad uniforme, un índice de elemento en el caso de que el grupo comprenda más de una palabra de código, haciendo referencia el índice de elemento a una palabra de código dentro del grupo, y para codificar el número de divisiones por un número de símbolos de escape, siendo un símbolo de escape un índice de grupo específico únicamente usado para indicar una división, y para codificar los restos de
40 las divisiones basándose en una distribución de probabilidad uniforme usando una regla de codificación aritmética.
5. El codificador de audio (100) de la reivindicación 4, en el que el codificador por entropía (120) está adaptado para
45 codificar una secuencia de símbolos en el flujo de audio codificado usando un alfabeto de símbolos que comprende el símbolo de escape y símbolos de grupo correspondientes a un conjunto de índices de grupo disponibles, un alfabeto de símbolos que comprende los índices de elementos correspondientes y un alfabeto de símbolos que comprende los diferentes valores de los restos.
6. Un decodificador de audio (200) para decodificar un flujo de audio codificado que representa una señal de audio
50 muestreada para obtener una secuencia de segmentos de coeficientes siendo anteriores entre sí en el tiempo y que representan la señal de audio muestreada mediante el uso de diferentes longitudes de transformada, de modo que los segmentos de coeficientes para los que se usan diferentes longitudes de transformada, representan espectralmente la señal de audio muestreada a diferentes resoluciones de frecuencia, que comprende

un procesador (210) para derivar un contexto de codificación para un coeficiente decodificado actualmente de un
55 segmento actual en base a un coeficiente decodificado previamente de un segmento anterior, correspondiendo los segmentos anterior y actual a diferentes resoluciones de frecuencia y diferentes longitudes de transformación, respectivamente, y de manera que los segmentos sean de 1024 y 128 coeficientes espectrales, respectivamente; y

un decodificador por entropía (220) para decodificar por entropía el coeficiente actual basándose en el contexto de
60 codificación y el flujo de audio codificado,

en donde el procesador (210) y el decodificador por entropía (220) están configurados para funcionar basándose en un muestreo ascendente de los coeficientes espectrales del segmento anterior, cuando el segmento anterior muestra una resolución espectral más gruesa que el segmento actual.

7. El decodificador de audio (200) de la reivindicación 6, en el que el procesador está adaptado para derivar el contexto de codificación por banda espectral para el coeficiente actual, basándose en coeficientes espectrales vecinos previamente decodificados en el segmento anterior y opcionalmente en el segmento actual.

5

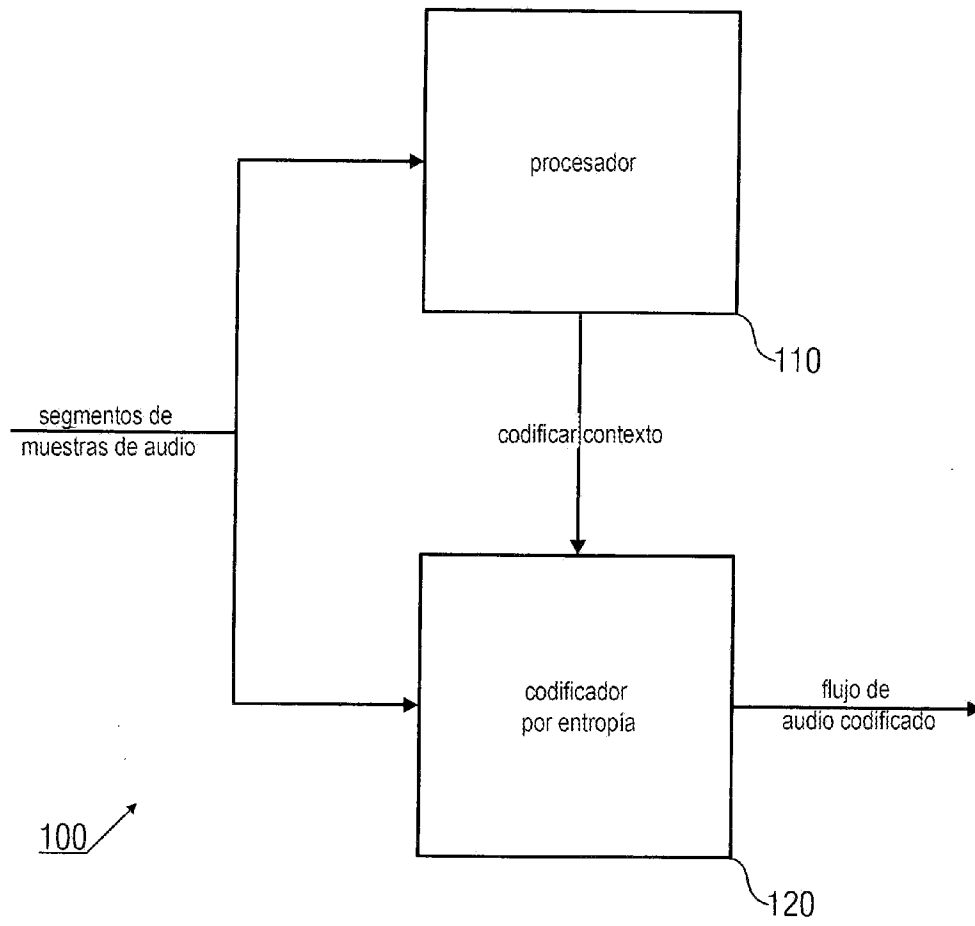


FIG 1

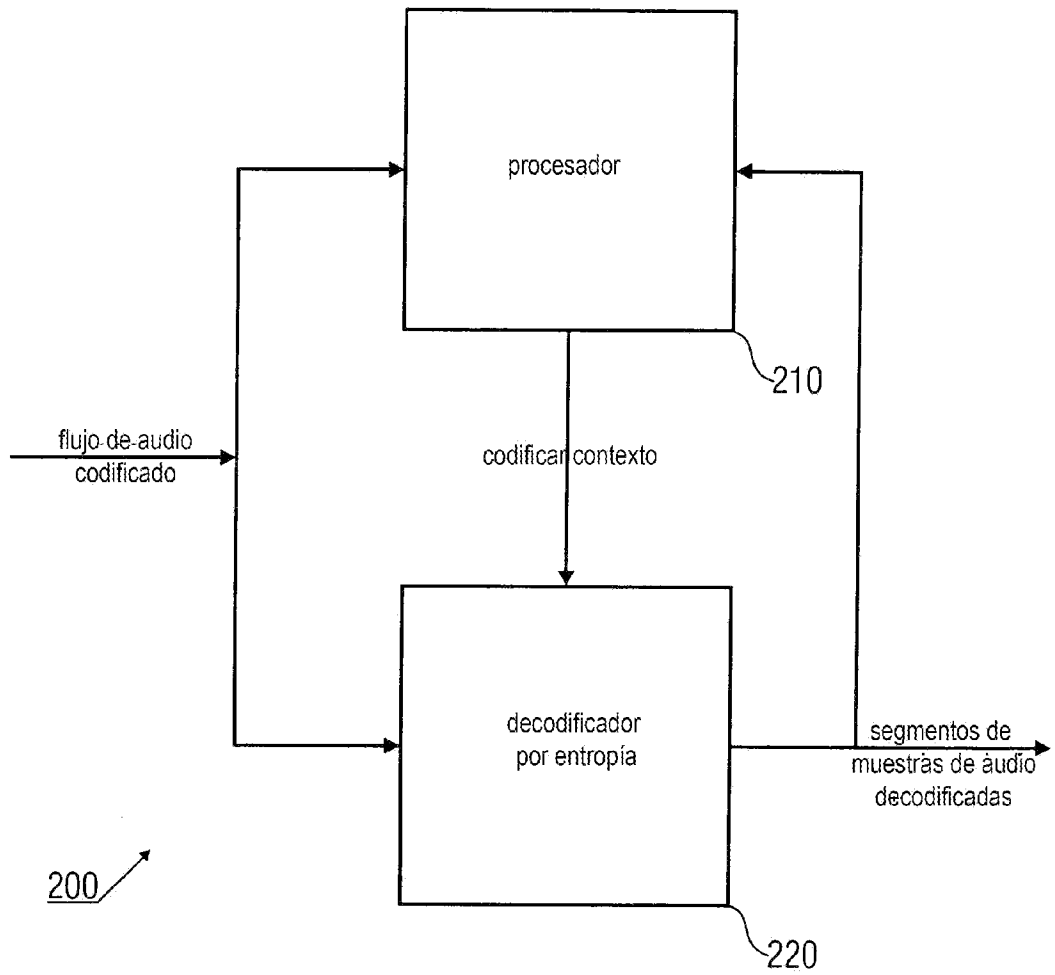


FIG 2

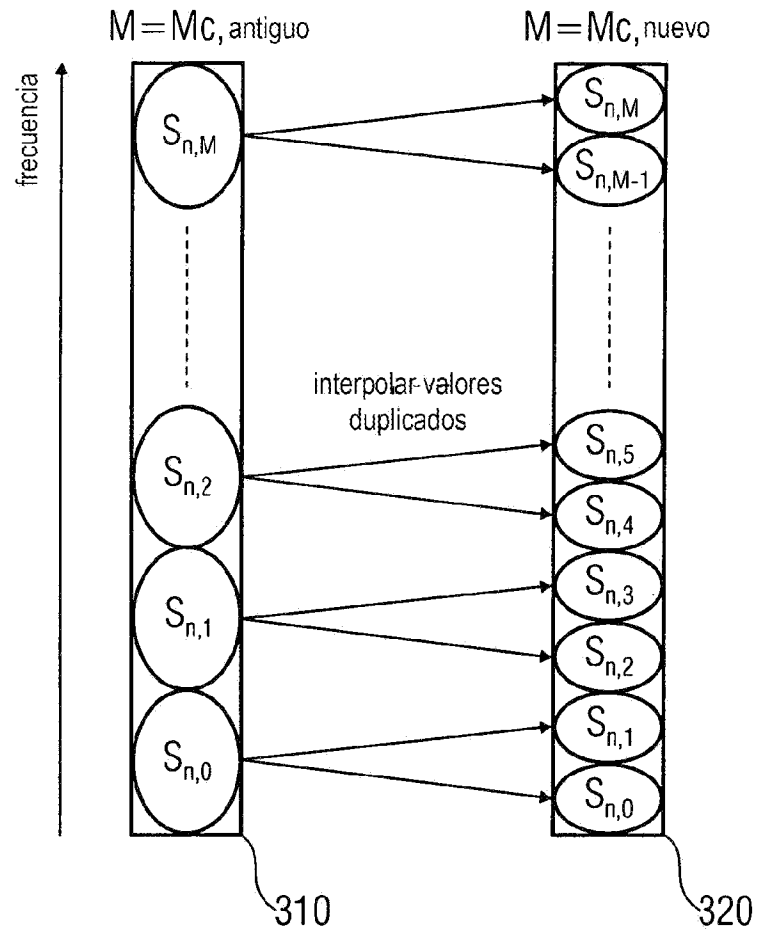


FIG 3

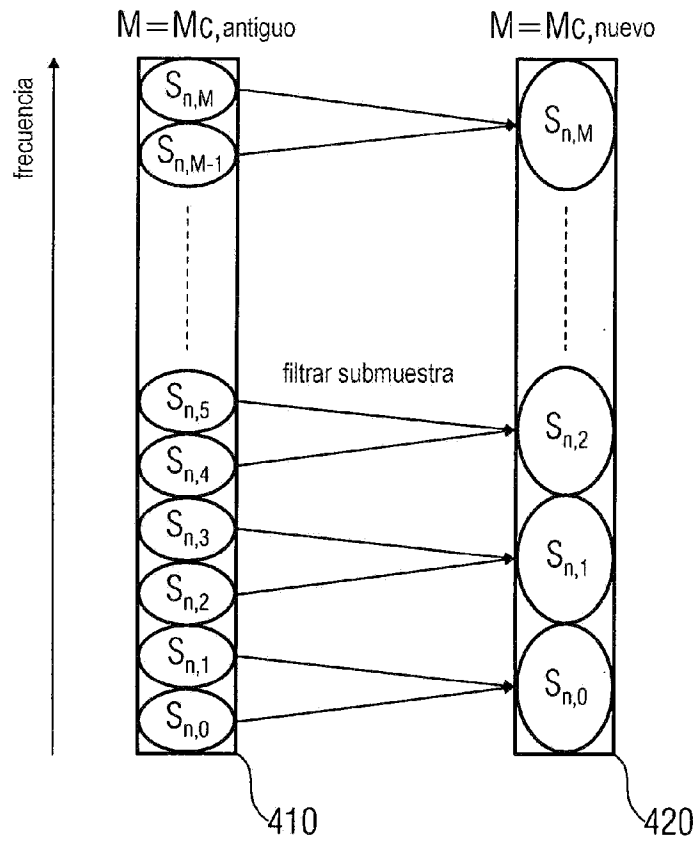


FIG 4

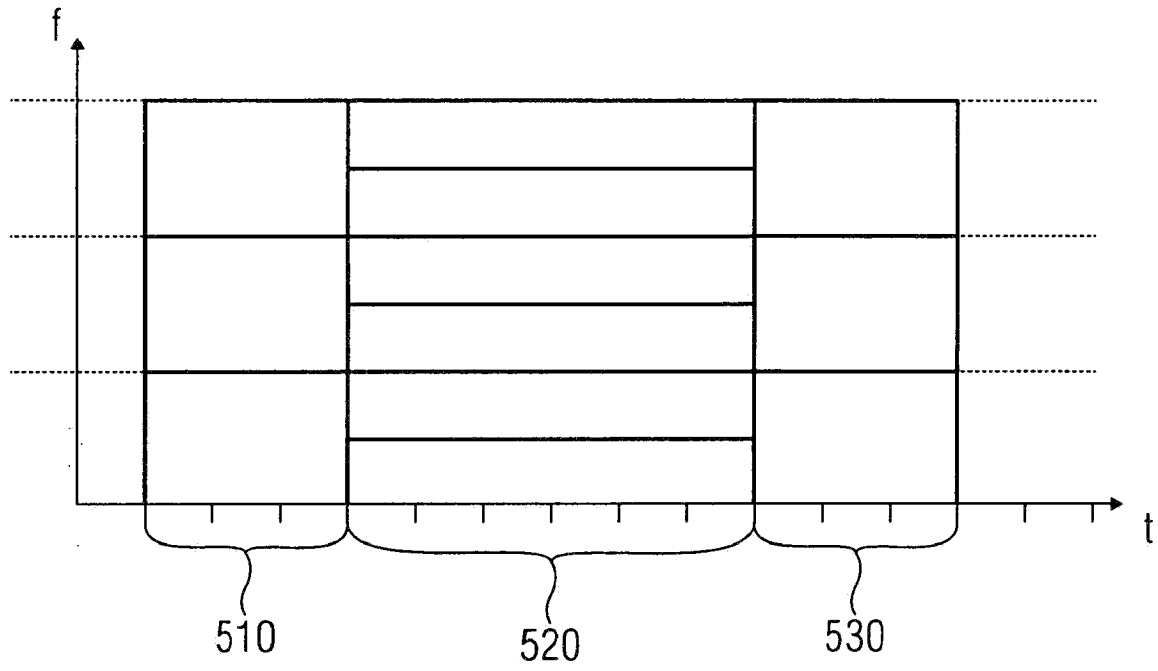


FIG 5

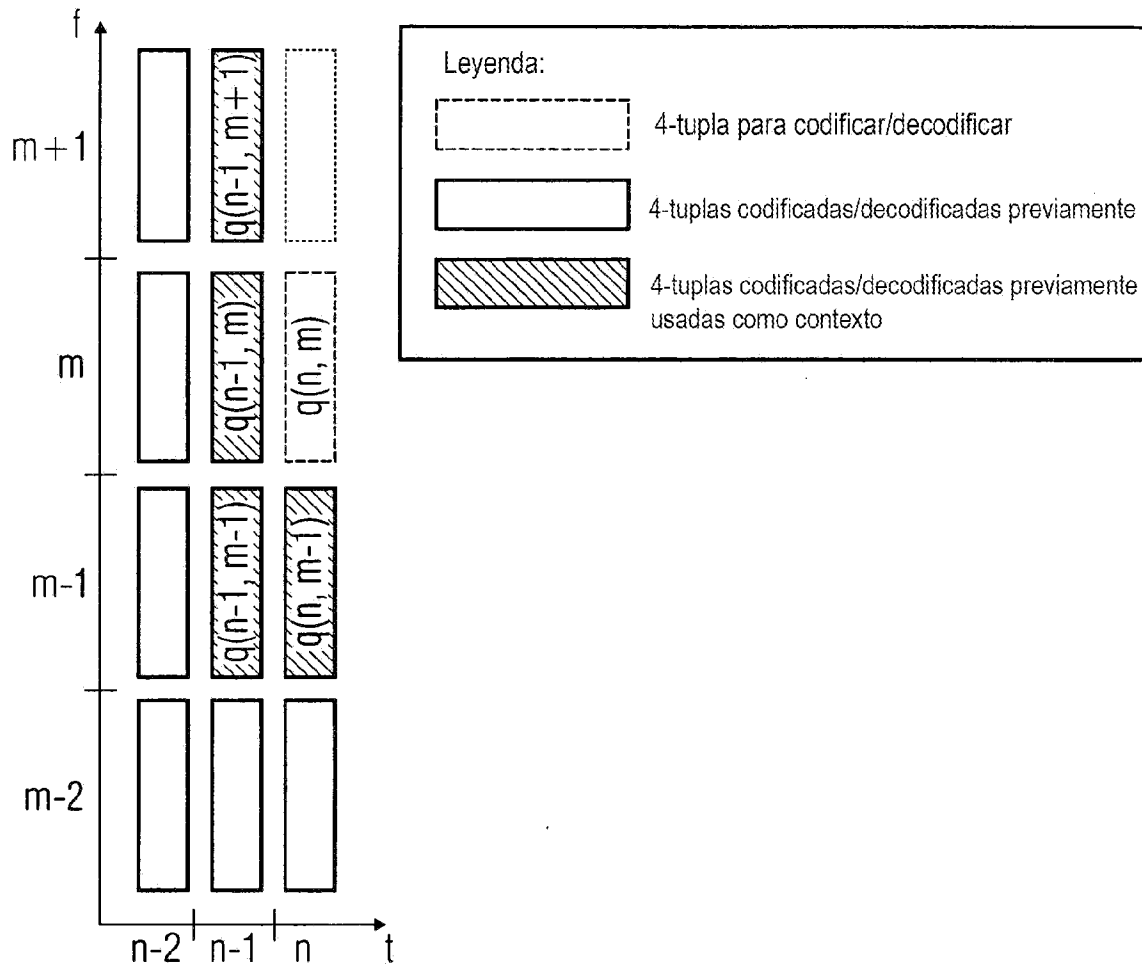


FIG 6

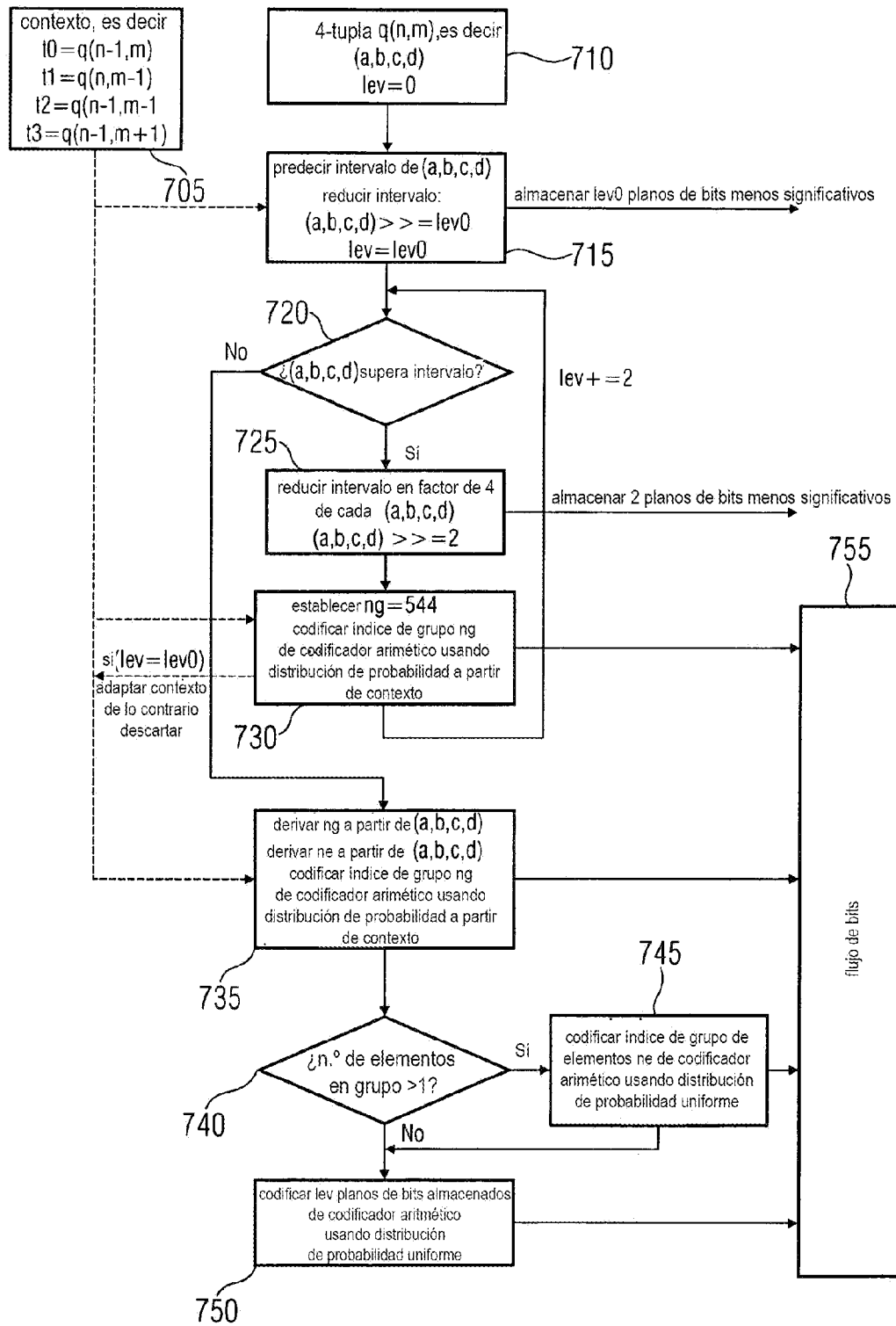


FIG 7A

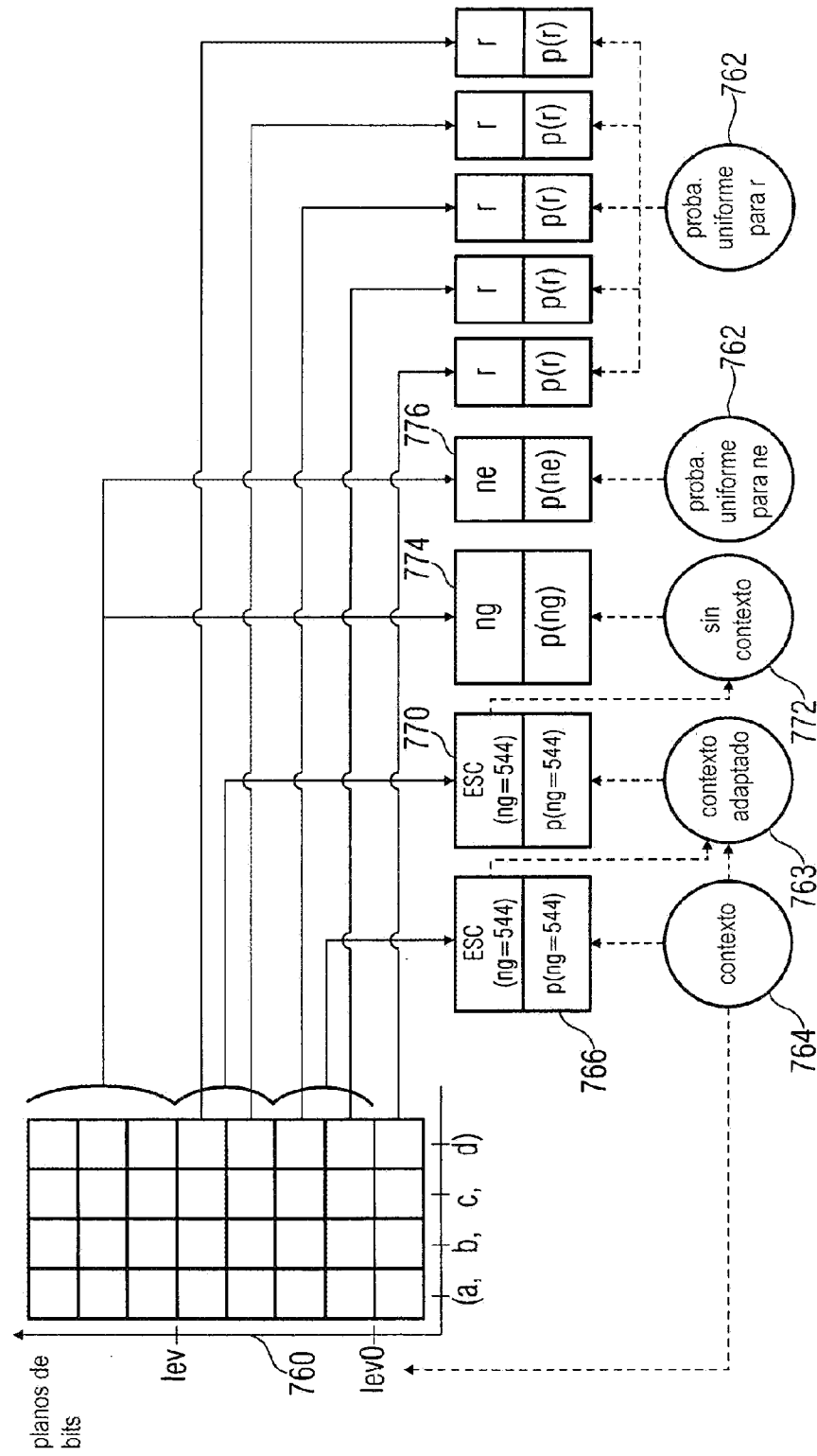


FIG 7B

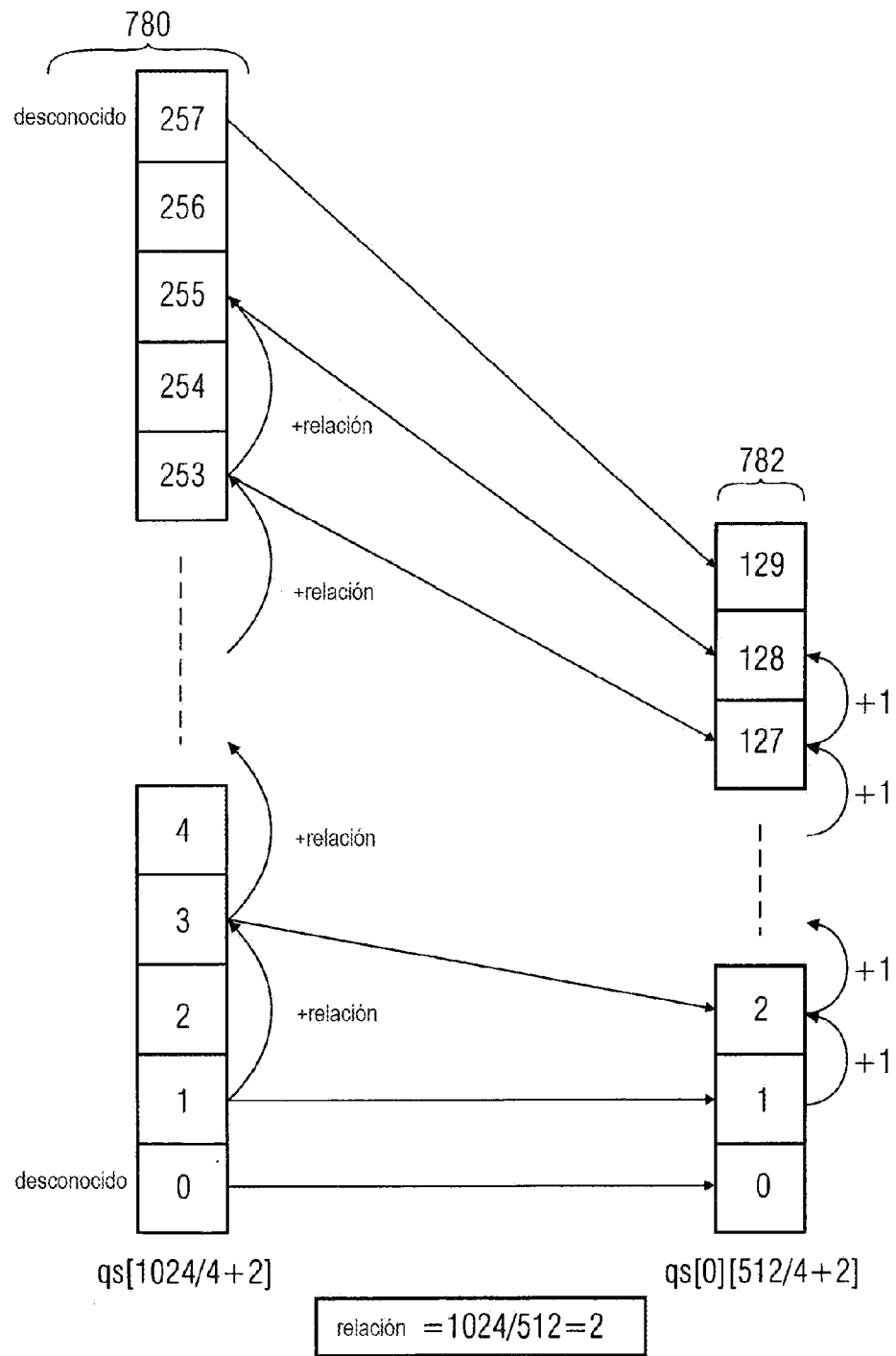


FIG 7C

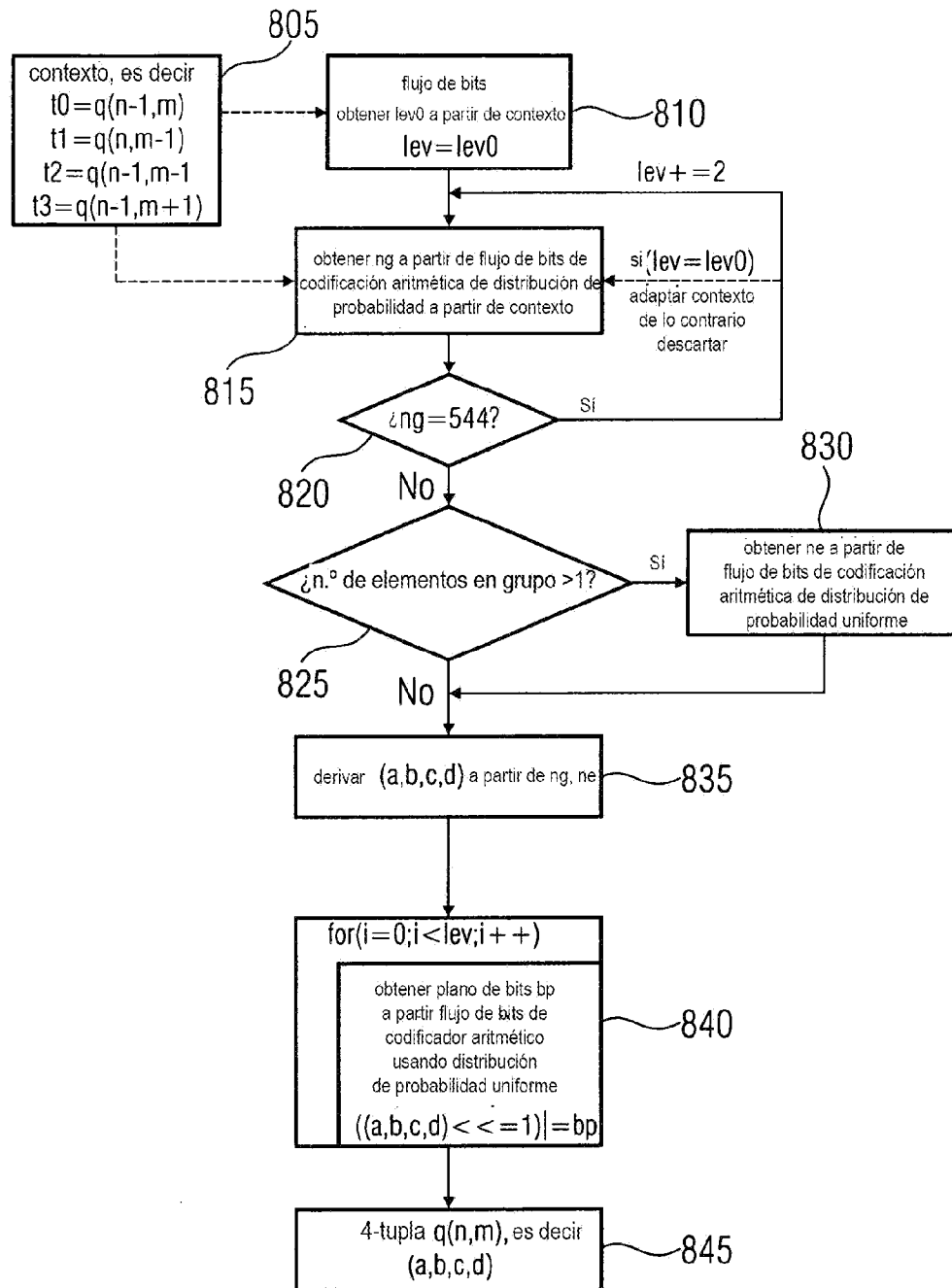


FIG 8

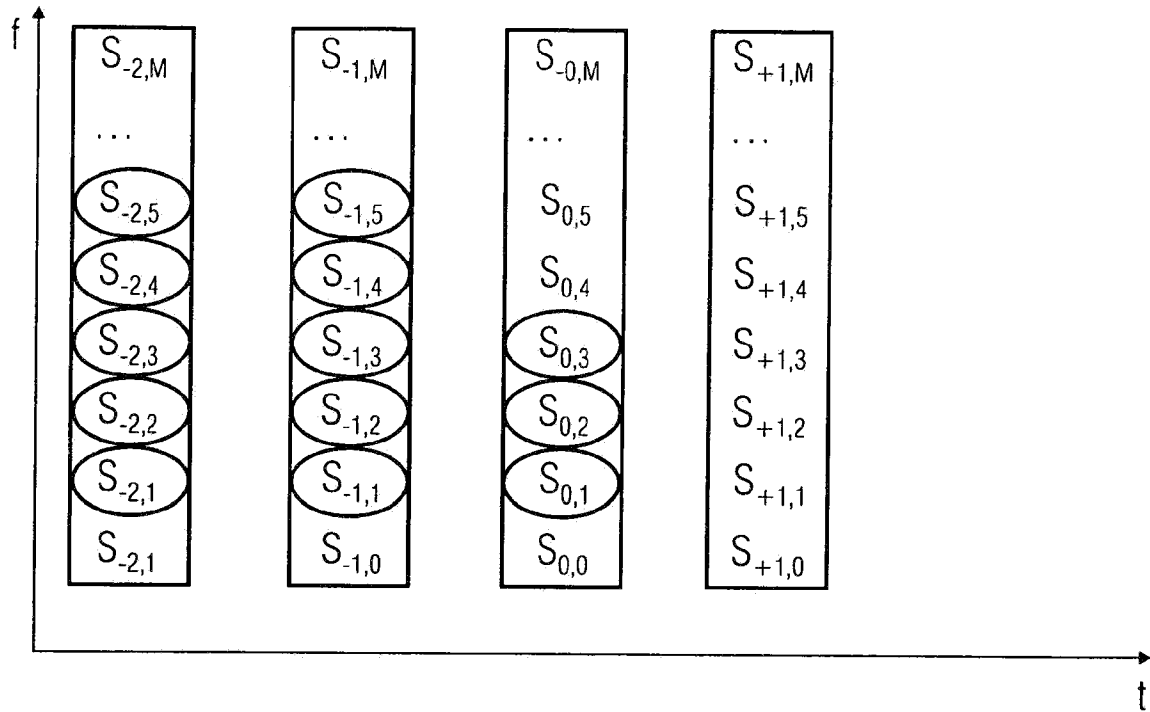


FIG 9
(ESTADO DE LA TÉCNICA)