(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 369 214**

(51) Int. CI.:
H04N 7/30 (2006.01)
H03M 7/40 (2006.01)
H04N 1/41 (2006.01)
H04N 7/26 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **08721576 .0**
(96) Fecha de presentación: **07.03.2008**
(97) Número de publicación de la solicitud: **2120461**
(97) Fecha de publicación de la solicitud: **18.11.2009**

(54) Título: **MÉTODO Y DISPOSITIVO DE ESTIMACIÓN DE LA CANTIDAD DE CÓDIGO, SU PROGRAMA Y MEDIO DE ALMACENAMIENTO.**

(30) Prioridad:
14.03.2007 JP 2007064283

(45) Fecha de publicación de la mención BOPI:
28.11.2011

(45) Fecha de la publicación del folleto de la patente:
28.11.2011

(73) Titular/es:
NIPPON TELEGRAPH AND TELEPHONE CORPORATION
3-1, OTEMACHI 2-CHOME
CHIYODA-KU TOKYO 100-8116, JP

(72) Inventor/es:
TANIDA, Ryuichi y
SHIMIZU, Atsushi

(74) Agente: **Pérez Barquín, Eliana**

DESCRIPCIÓN

Método y dispositivo de estimación de la cantidad de código, su programa y medio de almacenamiento

Campo de la invención

La presente invención se refiere a un método de estimación de la cantidad de código y a un dispositivo, programa, y medio de almacenamiento correspondientes en la codificación de vídeo en el que se somete una imagen de vídeo a una transformación ortogonal y codificación usando código de longitud variable.

Se reivindica prioridad de la solicitud de patente japonesa n.º 2007-064283, presentada el 14 de marzo de 2007, cuyo contenido se incorpora al presente documento como referencia.

Técnica anterior

En la mayoría de los métodos de codificación de vídeo recientes, cada trama se divide en pequeñas áreas, y se somete una imagen diferencial basada en una imagen predicha a una transformación ortogonal, cuantificación, y después codificación de entropía, comprimiendo así los datos de vídeo.

En la norma de codificación de vídeo H.264 (véase el documento no de patente 1) como formato de codificación de vídeo dominante actual, puede seleccionarse no sólo un método de codificación de longitud variable adaptativa según el contexto ("CAVLC") para realizar una codificación de entropía haciendo referencia a una tabla, sino también un método de codificación aritmética binaria adaptativa según el contexto ("CABAC") que puede mejorar adicionalmente la eficacia de codificación.

La CABAC anterior es un método de codificación que puede comprimir una señal estacionaria hasta un límite lógico, y por tanto es una técnica esencial para la codificación altamente eficaz. Sin embargo, en comparación con la CAVLC, el coste computacional de CABAC es muy alto (véase el documento no de patente 2).

Cuando se codifica una imagen de vídeo y se genera un flujo que puede distribuirse en una red que tiene una banda de transmisión limitada, es necesario producir una cantidad constante de código generado por unidad de tiempo para no superar la banda limitada. Generalmente, se ejecuta un control de tasa de transmisión para controlar la cantidad de código generado variando el tamaño de la etapa de cuantificación ("Q_{etapa}").

Por ejemplo, se codifica cada bloque objetivo de codificación; se calcula la cantidad de código generado correspondiente; y se ajusta la Q_{etapa} del siguiente bloque basándose en el resultado calculado, manteniendo así una cantidad constante de código generado.

Cuando se usa CABAC, se requiere una cantidad considerable de tiempo para obtener la cantidad de código generado, lo que aumenta un retraso en la codificación. En un método propuesto conocido para reducir el retraso, se aproxima la relación entre Q_{etapa} y la cantidad de código generado usando una función, de modo que se estima la cantidad de código generado (véase el documento de patente 1).

Sin embargo, usar una función aproximada produce una variación en la precisión de medición dependiendo de cada imagen de vídeo. Con el fin de realizar la estimación con una precisión mejorada, puede usarse CAVLC que tiene un coste computacional menor que CABAC para estimar la cantidad de código (es decir, estimación de la cantidad de código). En tal caso, se usa un resultado obtenido realizando la codificación de longitud variable, y por tanto puede ejecutarse una estimación de la cantidad de código superior.

Las figuras 7A y 7B muestran un diagrama de flujo de una operación de codificación mediante el cual puede usarse CAVLC para la estimación de la cantidad de código de CABAC. En este caso, la figura 7A muestra una rutina principal, y la figura B muestra un proceso de CABAC.

En primer lugar, se explicará la rutina principal (etapas de S101 a S111) en la figura 7A.

En primer lugar se determina el modo de inter-predicción y el modo de intra-predicción (véanse las etapas S101 y S102).

A continuación, se determina el modo de predicción realizando una intra/inter-determinación (véase la etapa S103), y se calcula una predicción residual para el modo determinado (véase la etapa S104) y se somete a DCT (véase la etapa S105).

Se aplica la cuantificación a coeficientes de transformada de DCT usando una Q_{etapa} suministrada (véase la etapa S106).

Se ordenan los coeficientes de transformada cuantificados en una forma unidimensional, y se suministra información

de coeficientes a una unidad de cálculo de CABAC. Simultáneamente, se realiza la estimación de la cantidad de código basándose en la información de coeficientes (proceso de precodificación) (véase la etapa S107).

Los coeficientes cuantificados también se someten a una cuantificación inversa (véase la etapa S108) e IDCT (véase la etapa S109), y entonces se añaden a una imagen predicha, generando así una imagen decodificada (véase la etapa S 110).

Finalmente, se somete la imagen decodificada a un proceso de filtrado (véase la etapa S111).

A continuación, se explicará el proceso de CABAC (véanse las etapas de S121 a S125) en la figura 7B.

En primer lugar, se espera (véanse las etapas de S121 a S122), la recepción de información de coeficientes generada en el proceso (S107) de precodificación. Cuando se reciben los datos relevantes, se realiza una etapa de CABAC (véase la etapa S123), y se transmite un flujo generado (véase la etapa S124). Finalmente, se envía la cantidad de código generado a un controlador de cantidad de código (véase la etapa S125).

La figura 8 muestra un ejemplo de la estructura para implementar la operación anterior.

El dispositivo mostrado tiene una unidad 101 de determinación de modo de inter-predicción, una unidad 102 de determinación de modo de intra-predicción, un selector 103 de modo de predicción, un conmutador 104, un substractor 105, una unidad 106 DCT, un cuantificador 107, un controlador 108 de cantidad de código, un procesador 109 de precodificación, un codificador 110 de entropía, un cuantificador inverso 111, una unidad 112 IDCT, un sumador 113, una memoria intermedia 114 de almacenamiento de imagen decodificada, un filtro 115, y una memoria intermedia 116 de almacenamiento de imagen de referencia.

La unidad 101 de determinación de modo de inter-predicción realiza la predicción compensada de movimiento usando una imagen de referencia en la memoria intermedia 116 de almacenamiento de imagen de referencia, determina el modo de inter-predicción, envía información del modo de predicción al selector 103 de modo de predicción, y también envía una imagen predicha al conmutador 104.

La unidad 102 de determinación de modo de intra-predicción determina el modo de intra-predicción usando una imagen decodificada en la memoria intermedia 114 de almacenamiento de imagen decodificada, envía información de modo de predicción al selector 103 de modo de predicción, y también envía una imagen predicha al conmutador 104.

El selector 103 de modo de predicción determina el modo de predicción, y selecciona uno del modo de intra-predicción y el modo de inter-predicción enviando una señal de control al conmutador 104.

Basándose en la señal de control del selector 103 de modo de predicción, el conmutador 104 selecciona una de una imagen inter-predicha enviada desde la unidad 101 de determinación de modo de inter-predicción y una imagen intra-predicha enviada desde la unidad 102 de determinación de modo de intra-predicción.

El substractor 105 genera una imagen residual predicha calculando la diferencia entre una imagen original y una imagen predicha, y envía la imagen generada a la unidad 106 DCT.

La unidad 106 DCT aplica una transformada DCT a la imagen residual predicha enviada, y envía la imagen al cuantificador 107.

El cuantificador 107 realiza la cuantificación de los coeficientes de transformada de DCT usando el tamaño de etapa de cuantificación Q_{etapa} enviado desde el controlador 108 de cantidad de código, y envía el resultado cuantificado al procesador 109 de precodificación y al cuantificador inverso 111.

Basándose en una cantidad de código estimada (cantidad de código estimada) enviada desde el procesador 109 de precodificación, el controlador 108 de cantidad de código calcula Q_{etapa} del siguiente macrobloque, y envía la Q_{etapa} calculada al cuantificador 107 y al cuantificador inverso 111. El controlador 108 de cantidad de código también recibe la cantidad de código generado enviada desde el codificador 110 de entropía, y corrige la diferencia con respeto a la cantidad de código estimada.

El procesador 109 de precodificación calcula la cantidad de código estimada basándose en los coeficientes de DCT cuantificados enviados desde el cuantificador 107, y envía el valor calculado al controlador 108 de cantidad de código. El procesador 109 de precodificación también genera información de coeficientes ordenando los coeficientes de DCT cuantificados (datos bidimensionales) en una forma unidimensional, y envía la información generada al codificador 110 de entropía.

El codificador 110 de entropía codifica la información de coeficientes, que se envía desde el procesador 109 de precodificación, por medio de CABAC, y emite los datos codificados como un flujo codificado.

El cuantificador inverso 111 realiza la cuantificación inversa multiplicando el valor cuantificado relevante por Q_{etapa}, y envía el resultado a la unidad 112 IDCT.

La unidad 112 IDCT aplica IDCT a los datos recibidos, y envía el resultado al sumador 113.

El sumador 113 añade la imagen residual predicha enviada desde la unidad 112 IDCT a la imagen predicha enviada desde el conmutador 104, y envía el resultado como una imagen decodificada a la memoria intermedia 114 de almacenamiento de imagen decodificada.

La memoria intermedia 114 de almacenamiento de imagen decodificada almacena la imagen decodificada enviada desde el sumador 113, y envía la imagen al filtro 115. La memoria intermedia 114 de almacenamiento de imagen decodificada también envía información de píxeles adyacentes a la unidad 102 de determinación de modo de intra-predicción.

El filtro 115 aplica un proceso de filtrado a la imagen decodificada almacenada en la memoria intermedia 114 de almacenamiento de imagen decodificada, y envía la imagen filtrada a la memoria intermedia 116 de almacenamiento de imagen de referencia.

La memoria intermedia 116 de almacenamiento de imagen de referencia almacena la imagen decodificada filtrada, y envía la imagen como una imagen de referencia a la unidad 101 de determinación de modo de inter-predicción.

Según las funciones anteriores, se implementa la operación mostrada en las figuras 7A y 7B.

A continuación, se explicará el procesador 109 de precodificación, al que puede aplicarse la presente invención.

El procesador 109 de precodificación ordena los datos bidimensionales de los coeficientes de DCT cuantificados en una forma unidimensional, genera información de coeficientes, envía la información al codificador 110 de entropía, y estima la cantidad de código haciendo referencia a una tabla.

En primer lugar, se explicará el método de generar información de coeficientes a partir de datos bidimensionales.

En un ejemplo en el que los coeficientes de DCT tienen una forma de bloque 4x4, los coeficientes se ordenan en una forma unidimensional en el orden mostrado en la figura 9, y los valores de coeficientes se examinan secuencialmente desde el coeficiente de orden 0 de modo que se almacena como un conjunto el número de coeficientes sucesivos que tienen un valor de 0 y el coeficiente (coeficiente distinto de cero) que sigue a los coeficientes y tiene un valor distinto de 0. En este caso, el número de coeficientes “0” sucesivos se denomina Ejecución y el coeficiente distinto de 0 se denomina Nivel. Una operación de barrido de este tipo de los valores de coeficientes en forma de zigzag para ordenarlos en una forma unidimensional y convertirlos en datos de Ejecución-Nivel se denomina “barrido en zigzag”.

En la figura 10 se muestra un ejemplo específico, en el que no existe ningún “0” antes de los coeficientes “5” y “3”, y se les asigna 0 (como Ejecución).

Adicionalmente, en la tabla de referencia en H.264, se necesita no sólo Ejecución y Nivel, sino también (i) el número de los coeficientes distintos de cero y (ii) el número de la sucesión final de coeficientes “1” o “-1” y el signo relevante. Basándose en los datos necesarios, se estima la cantidad de código haciendo referencia a una tabla. Además, la información de Ejecución-Nivel se codifica por medio de codificación aritmética.

La figura 11 muestra un ejemplo de un diagrama de flujo de la operación anterior.

En primer lugar, se realiza el barrido en zigzag del bloque 4x4 relevante, y se obtienen los conjuntos Ejecución-Nivel (véase la etapa S151). Los resultados se envían al codificador 110 de entropía (véase la etapa S 152).

Para los conjuntos Ejecución-Nivel obtenidos, se determina el número de coeficientes distintos de cero, el número de sucesión final de coeficientes “1” o “-1”, y el signo positivo o negativo de los mismos (véase la etapa S153), y se calcula la cantidad de código relevante usando una tabla de codificación de longitud variable (denominada “tabla VLC”) (véase la etapa S154).

La cantidad de código calculada se envía como cantidad de código estimada (cantidad de código estimada) al controlador 108 de cantidad de código (véase la etapa S 155).

La figura 12 muestra un diagrama de flujo de barrido en zigzag.

En primer lugar, se inicializan contadores i y n cada uno a 0 (véase la etapa S201). Adicionalmente, también se inicializa la variable “ejecución” a 0 (véase la etapa S202).

A continuación, se obtienen las coordenadas S_i(x, y) del coeficiente de orden i en el barrido haciendo referencia a una tabla, y se almacena el valor de coeficiente en las coordenadas obtenidas en k[i] (véase la etapa S204). En un ejemplo de procesamiento de un bloque 4x4, posteriormente se introducen los coeficientes en k [i] en el orden mostrado en la figura 9.

Si k[i]=0 (véase la etapa S205), la ejecución se incrementa en 1 (véase la etapa S206), e i también se incrementa en 1 (véase la etapa S209).

Si k[i] no es cero (véase la etapa S205), el valor de ejecución se almacena en Ejecución[n] para almacenar información de Ejecución, y el coeficiente distinto de cero k[i] se almacena en Nivel[n] para almacenar información de Nivel (véase la etapa S207). Entonces se incrementa i en 1 (véase la etapa S209).

Cuando el barrido ha llegado al último coeficiente, se completa la operación (véase la etapa S210). Cuando el barrido aún no ha llegado al último coeficiente, se repite el proceso anterior desde la etapa S203 hasta la S210.

Según la operación anterior, los conjuntos Ejecución-Nivel pueden obtenerse por medio de barrido en zigzag.

La figura 13 muestra un ejemplo de la estructura del procesador 109 de precodificación en la figura 8.

La estructura incluye una memoria intermedia 201 de almacenamiento de valor cuantificado, un contador 202 de ejecución, un controlador 203 de proceso de precodificación, un contador 204 de barrido 4x4, una tabla 205 de referencia de orden de barrido 4x4, una memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel, un controlador 207 de estimación de la cantidad de código, una unidad 208 de estimación de la cantidad de código, y una memoria 209 de almacenamiento de tabla de VLC.

La memoria intermedia 201 de almacenamiento de valor cuantificado almacena los (valores de) coeficientes de DCT cuantificados. Cuando se recibe información de coordenadas de la tabla 205 de referencia de orden de barrido 4x4, la memoria intermedia 201 de almacenamiento de valor cuantificado envía el valor cuantificado correspondiente a las coordenadas relevantes al contador 202 de ejecución. Cuando se recibe el valor cuantificado, la memoria intermedia 201 de almacenamiento de valor cuantificado envía una señal de inicio de operación al controlador 203 de proceso de precodificación.

El contador 202 de ejecución almacena una variable “ejecución” y recibe el valor cuantificado de la memoria intermedia 201 de almacenamiento de valor cuantificado. Cuando el valor cuantificado recibido es 0, el contador 202 de ejecución incrementa la ejecución en 1. Cuando el valor cuantificado recibido no es 0, el contador 202 de ejecución envía el coeficiente relevante y la Ejecución actualmente almacenada a la memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel como información de Ejecución-Nivel, y reinicia la ejecución a 0. El contador 202 de ejecución también reinicia la ejecución a 0 cuando se recibe una señal de reinicio del controlador 203 de proceso de precodificación.

Cuando el controlador 203 de proceso de precodificación recibe una señal de inicio de la memoria intermedia 201 de almacenamiento de valor cuantificado, el controlador 203 de proceso de precodificación envía una señal de reinicio al contador 202 de ejecución y la memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel para reiniciarlos, y entonces envía una señal de inicio de operación al contador 204 de barrido 4x4. Además, cuando se recibe una señal de terminación del contador 204 de barrido 4x4, el controlador 203 de proceso de precodificación envía una señal de inicio de estimación al controlador 207 de estimación de la cantidad de código.

Cuando se recibe la señal de inicio de operación del controlador 203 del proceso de precodificación, el contador 204 de barrido 4x4 envía secuencialmente valores numéricos desde 0 hasta 15 a la tabla 205 de referencia de orden de barrido 4x4. Cuando se ha enviado el último “15”, el contador 204 de barrido 4x4 envía una señal de terminación al controlador 203 de proceso de precodificación.

La tabla 205 de referencia de orden de barrido 4x4 recibe coordenadas correspondientes a los valores numéricos del contador 204 de barrido 4x4, y envía las coordenadas a la memoria intermedia 201 de almacenamiento de valor cuantificado.

Cuando se recibe información de Ejecución-Nivel del contador 202 de ejecución, la memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel almacena la información, y la envía a la unidad 208 de estimación de la cantidad de código según una señal de control del controlador 207 de estimación de la cantidad de código. La memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel también envía la información de Ejecución-Nivel al codificador 110 de entropía. Adicionalmente, cuando se recibe una señal de reinicio del controlador 203 del proceso de precodificación, la memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel limpia el contenido de la memoria intermedia.

Cuando el controlador 207 de estimación de la cantidad de código recibe una señal de inicio de estimación del

controlador 203 de proceso de precodificación, el controlador 207 de estimación de la cantidad de código envía una señal de inicio de estimación a la unidad 208 de estimación de la cantidad de código, y también envía una señal de control a la memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel para enviar información de Ejecución-Nivel a la unidad 208 de estimación de la cantidad de código.

Cuando se recibe la señal de inicio de estimación del controlador 207 de estimación de la cantidad de código, la unidad 208 de estimación de la cantidad de código recibe información de VLC de la memoria 209 de almacenamiento de tabla de VLC basándose en la información de Ejecución-Nivel enviada desde la memoria intermedia 206 de almacenamiento de información de Ejecución-Nivel, y estima y emite una cantidad de código.

La memoria 209 de almacenamiento de tabla de VLC almacena una tabla de VLC, y la envía como la información de VLC a la unidad 208 de estimación de la cantidad de código.

Según la estructura anterior, puede implementarse la operación tal como se muestra en la figura 11.

Documento no de patente 1: Sakae Okubo, Shinya Kadono, Yoshihiro Kikuchi, y Teruhiko Suzuki, "H.264/AVC TEXTBOOK", Impress, págs. 144-146, 2004.

Documento no de patente 2: CABAC: Detlev Marpe, Heiko Schwarz, Thomas Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, Vol. 13, n.º 7, págs. 620-636, julio de 2003.

Documento de patente 1: solicitud de patente japonesa no examinada, primera publicación n.º H07-264579.

El documento US 2006/0176953 A1 da a conocer un controlador de tasa de transmisión en un sistema de vídeo compuesto por un estimador de recuento de bits y un selector de cuantificación. El estimador de recuento de bits recibe una entrada para un codificador y genera una estimación de recuento de bits. La estimación de recuento de bits es una aproximación de un recuento de bits en una salida del codificador. El selector de cuantificación fija un valor de cuantificación basándose en la estimación de recuento de bits.

El documento XP-002339408 da a conocer un método para armonizar la decodificación de entropía de CAVLC con la decodificación de VLC dedicada de un proceso de decodificación de ABT.

Descripción de la invención

Problema que va a solucionar la invención

Incluso cuando puede seleccionarse una pluralidad de tamaños para la transformación ortogonal, no siempre se preparan tablas de codificación de longitud variable correspondientes a todos los tamaños. Es decir, puede no prepararse ninguna tabla de codificación de longitud variable correspondiente a una transformación ortogonal de gran tamaño, y una tabla de codificación de longitud variable correspondiente a una transformación ortogonal de pequeño tamaño puede también asignarse a la transformación ortogonal de gran tamaño.

Por consiguiente, en un sistema de codificación de vídeo que tiene una pluralidad de métodos de codificación de longitud variable seleccionables, el método de generación de Ejecución-Nivel puede cambiarse dependiendo del método de codificación de longitud variable incluso con el mismo tamaño de transformación ortogonal. En un caso de este tipo, la cantidad de código requerida en una codificación de longitud variable que tiene un alto coste computacional puede estimarse usando una codificación de longitud variable que tiene un bajo coste computacional, y tal estimación puede aumentar el coste computacional.

Por ejemplo, en H.264, puede usarse no sólo DCT 4x4 sino también DCT 8x8. La figura 14 muestra un orden de barrido empleado cuando se codifican coeficientes de cuantificación de DCT 8x8 por medio de CABAC. Tal como se entiende por la figura 14, se realiza un barrido en zigzag desde 0 hasta 63 de modo que se determina la Ejecución y el Nivel.

En cambio, cuando se codifican los coeficientes de cuantificación de DCT 8x8 por medio de CAVLC, no hay ninguna tabla de codificación de longitud variable exclusiva para DCT 8x8, y el objetivo de codificación se divide en cuatro partes para usar una tabla de codificación de longitud variable preparada para DCT 4x4. Por tanto, cuando se realiza la codificación de DCT 8x8 usando CAVLC, el barrido, cuyo orden difiere completamente del de CABAC, se ejecuta cuatro veces, y por tanto deben fijarse cuatro pseudo-coeficientes de DCT 4x4 divididos.

La figura 15 muestra un orden de barrido de CAVLC. Los 64 (8x8) coeficientes se clasifican en cuatro grupos (de A0 a A15; de B0 a B15; de C0 a C15; y de D0 a D15) y se procesan.

En la figura 15, se barren secuencialmente un primer bloque desde A0 hasta A15, un segundo bloque desde B0 hasta B15, un tercer bloque desde C0 hasta C15, y un cuarto bloque desde D0 hasta D15, es decir, se ejecuta el

barrido cuatro veces. Tras fijar cuatro bloques divididos, se calcula la cantidad de código requerida para cada conjunto de coeficientes de DCT 4x4 de cuatro bloques haciendo referencia a una tabla de VLC para DCT 4x4, y entonces se calcula el coste total.

Por tanto, cuando se usa CAVLC para estimar la cantidad de código requerida en la codificación de CABAC, debe ejecutarse un barrido cuatro veces independiente de CABAC, lo que aumenta el coste computacional.

La figura 16 muestra un ejemplo de un flujo de operación de un método convencional ejecutado por el procesador 109 de precodificación para DCT 8x8.

En primer lugar, se inicializa la cantidad estimada Tasa de transmisión de código a 0 (véase la etapa S301). Después, se realiza el barrido en zigzag para la codificación real (véase la etapa S302). Esta etapa se realiza de manera similar al flujo en la figura 12, y la tabla de referencia de orden de barrido devuelve coordenadas en el orden mostrado en la figura 14. La información de Ejecución-Nivel obtenida se envía al codificador 110 de entropía (véase la etapa S303).

A continuación, se realiza el proceso de estimar la cantidad de código.

En primer lugar, se inicializa un contador de bucle i a 0 (véase la etapa S304), y se realiza el barrido del primer bloque (de orden 0) (véase la etapa S305). Esta etapa se realiza de manera similar al flujo en la figura 12, y la tabla de referencia de orden de barrido devuelve las coordenadas en de A0 a A15 en la figura 15.

Basándose en la información de Ejecución-Nivel obtenida, se calcula el número de coeficientes distintos de cero, el número de sucesión final de coeficientes “1” o “-1”, y el signo positivo o negativo de la sucesión (véase la etapa S306), y se calcula la cantidad de código usando una tabla de VLC (véase la etapa S307).

Se añade la cantidad de código calculada a la Tasa de transmisión (véase la etapa S308), se incrementa i en 1 (véase la etapa S310), y se somete el segundo bloque a una operación similar a la anterior (véanse las etapas de S305 a S310), en la que la tabla de referencia de orden de barrido devuelve las coordenadas en de B0 a B 15 en la figura 15.

Entonces, de C0 a C15 y de D0 a D15 se someten cada uno a una operación similar a la anterior (véanse las etapas de S305 a S310), y finalmente, se transmite el valor de la cantidad estimada Tasa de transmisión de código (véase la etapa S311).

La figura 17 muestra un ejemplo de la estructura que implementa la operación anterior.

La estructura incluye una memoria intermedia 301 de almacenamiento de valor cuantificado 8x8, un contador 302 de ejecución, un controlador 303 de proceso de precodificación, un contador 304 de barrido 8x8, una tabla 305 de referencia de orden de barrido 8x8, un contador 306 de barrido 4x4, un conmutador 307 “A”, una tabla 308 de referencia de orden de barrido 4x4 “a”, una 309 tabla de referencia de orden de barrido 4x4 “b”, una tabla 310 de referencia de orden de barrido 4x4 “c”, una tabla 311 de referencia de orden de barrido 4x4 “d”, una memoria intermedia 312 de almacenamiento de información de Ejecución-Nivel, un controlador 313 de estimación de la cantidad de código 8x8, una unidad 314 de estimación de la cantidad de código, una memoria 315 de almacenamiento de tabla de VLC, y una unidad 316 de cálculo de cantidad de código estimada.

Entre los elementos estructurales anteriores, el contador 302 de ejecución, el controlador 303 de proceso de precodificación, el contador 306 de barrido 4x4, la memoria intermedia 312 de almacenamiento de información de Ejecución-Nivel, la unidad 314 de estimación de la cantidad de código, y la memoria 315 de almacenamiento de tabla de VLC tienen las mismas funciones que las de los elementos estructurales descritos anteriormente que tienen los mismos nombres.

La memoria intermedia 301 de almacenamiento de valor cuantificado 8x8 almacena los valores cuantificados de los coeficientes de DCT 8x8. Cuando se recibe información de coordenadas de la tabla 305 de referencia de orden de barrido 8x8 y las tablas de referencia de orden de barrido 4x4 308 “a”, 309 “b”, 310 “c”, y 311 “d”, la memoria intermedia 301 de almacenamiento de valor cuantificado 8x8 envía los valores cuantificados almacenados en las coordenadas correspondientes al contador 302 de ejecución.

Cuando el contador 304 de barrido 8x8 recibe una señal de inicio de operación del controlador 303 de proceso de precodificación, el contador 304 de barrido 8x8 envía secuencialmente valores numéricos de 0 a 63 a la tabla 305 de referencia de orden de barrido 8x8.

La tabla 305 de referencia de orden de barrido 8x8 envía coordenadas correspondientes a cada valor numérico (enviado desde el contador 304 de barrido 8x8) a la memoria intermedia 301 de almacenamiento de valor cuantificado 8x8.

El conmutador 307 “A” realiza la conmutación entre terminales a a d basándose en una señal de control enviada desde el contador 306 de barrido 4x4.

La tabla 308 de referencia de orden de barrido 4x4 “a” envía coordenadas correspondientes a cada valor numérico (enviado desde el contador 306 de barrido 4x4) a la memoria intermedia 301 de almacenamiento de valor cuantificado 8x8. Las coordenadas corresponden a de A0 a A15 en la figura 15.

La tabla 309 de referencia de orden de barrido 4x4 “b” envía coordenadas correspondientes a cada valor numérico (enviado desde el contador 306 de barrido 4x4) a la memoria intermedia 301 de almacenamiento de valor cuantificado 8x8. Las coordenadas corresponden a de B0 a B15 en la figura 15.

La tabla 310 de referencia de orden de barrido 4x4 “c” envía coordenadas correspondientes a cada valor numérico (enviado desde el contador 306 de barrido 4x4) a la memoria intermedia 301 de almacenamiento de valor cuantificado 8x8. Las coordenadas corresponden a de C0 a C15 en la figura 15.

La tabla 311 de referencia de orden de barrido 4x4 “d” envía coordenadas correspondientes a cada valor numérico (enviado desde el contador 306 de barrido 4x4) a la memoria intermedia 301 de almacenamiento de valor cuantificado 8x8. Las coordenadas corresponden a de D0 a D15 en la figura 15.

Cuando el controlador 313 de estimación de la cantidad de código 8x8 recibe una señal de inicio de estimación del controlador 303 de proceso de precodificación, el controlador 313 de estimación de la cantidad de código 8x8 envía una señal de control al conmutador 307 “A” para seleccionar el terminal a del conmutador, y entonces envía una señal de inicio al contador 306 de barrido 4x4.

Cuando se recibe una señal de terminación del contador 306 de barrido 4x4, el controlador 313 de estimación de la cantidad de código 8x8 envía una señal de inicio de estimación a la unidad 314 de estimación de la cantidad de código, y envía una señal de control a la memoria intermedia 312 de almacenamiento de información de Ejecución-Nivel para suministrar información de Ejecución-Nivel a la unidad 314 de estimación de la cantidad de código.

El controlador 313 de estimación de la cantidad de código 8x8 envía entonces una señal de control al conmutador 307 “A” para seleccionar el terminal b del conmutador, y entonces envía una señal de inicio al contador 306 de barrido 4x4.

Cuando se recibe una señal de terminación del contador 306 de barrido 4x4, el controlador 313 de estimación de la cantidad de código 8x8 envía una señal de inicio de estimación a la unidad 314 de estimación de la cantidad de código, y también envía una señal de control a la memoria intermedia 312 de almacenamiento de información de Ejecución-Nivel para suministrar información de Ejecución-Nivel a la unidad 314 de estimación de la cantidad de código.

Después de eso, el terminal del conmutador 307 “A” se conmuta a c, y después a d, de modo que se realizan operaciones similares a las explicadas anteriormente.

Cuando la unidad 316 de cálculo de cantidad de código estimada recibe una cantidad de código estimada de la unidad 314 de estimación de la cantidad de código, la unidad 316 de cálculo de cantidad de código estimada realiza la adición del valor recibido. Es decir, tras añadir las cuatro cantidades de código relevantes, la unidad 316 de cálculo de cantidad de código estimada transmite la suma de las mismas como una cantidad de código estimada. Adicionalmente, cuando se recibe una señal de reinicio del controlador 303 de proceso de precodificación, la unidad 316 de cálculo de cantidad de código estimada reinicia el valor almacenado a 0.

Según la estructura anterior, puede implementarse la operación mostrada en la figura 16.

Sin embargo, el barrido debe ejecutarse cinco veces tal como se explicó anteriormente, la cantidad de procesamiento aumenta y la estructura es compleja.

Las técnicas referentes a la presente invención y los problemas de las técnicas convencionales, que se han descrito, volverán a describirse de manera resumida según un ejemplo concreto de estimación de la cantidad de código por medio de un método H.264 de codificación.

Técnicas relacionadas con la presente invención

En comparación con la codificación de longitud variable (CAVLC), la codificación aritmética (CABAC) tiene un grado superior de eficacia de codificación, pero requiere un coste computacional superior. Sin embargo, con el fin de controlar la cantidad de código, es necesario detectar rápidamente la cantidad de código generado. Por tanto, la cantidad de código se estima usando la codificación de longitud variable (CAVLC) que realiza una operación de alta velocidad y requiere un pequeño coste computacional, y la codificación real se ejecuta realizando CABAC en un proceso separado. Es decir, en una técnica básica relacionada con la presente invención, la codificación real se

realiza usando CABAC altamente eficaz (que produce un gran retraso), y la estimación de la cantidad de código se realiza usando CAVLC de alta velocidad (que produce un pequeño retraso).

Cuando se estima la cantidad de código basándose en la técnica básica anterior, deben considerarse las siguientes especificaciones de CABAC y CAVLC.

Especificación de CAVLC en H.264

En H.264, pueden seleccionarse DCT 4x4 y DCT 8x8. Sin embargo, no hay ninguna tabla de codificación de longitud variable (tabla de VLC) para DCT 8x8.

Por tanto, con el fin de usar la tabla de VLC para el DCT 8x8, en vez de eso, se dividen los 64 (8x8) elementos en cuatro partes (no una división en cuatro sencilla), para producir conjuntos de 16 elementos y para realizar el barrido cuatro veces. Es decir, se procesan cuatro pseudo-elementos de DCT 4x4.

Por consiguiente, puede hacerse referencia a una tabla de VLC para DCT 4x4 (en el que se necesita hacer referencia cuatro veces).

Especificación de CABAC en H.264

En CABAC en H.264, se reordenan coeficientes de DCT 8x8 usando un barrido en zigzag similar al empleado en MREG-2, para realizar la codificación.

Es decir, en el barrido de CABAC, se alinean 64 coeficientes en una línea recta. En cambio, en CAVLC que realiza el barrido cuatro veces, se producen cuatro líneas de coeficientes, en cada una de las cuales se alinean 16 coeficientes en una línea recta.

Problema que se produce cuando se usa CAVLC en la estimación de la cantidad de código de CABAC.

Cuando se codifican coeficientes de DCT 8x8 en H.264 por medio de VLC, no hay ninguna tabla de VLC para DCT 8x8 tal como se describió anteriormente, y por tanto se reordenan 64 (8x8) elementos en cuatro matrices unidimensionales. Cada una de las cuatro matrices se considera como un resultado de barrido en zigzag, y se codifica haciendo referencia a una tabla de VLC para DCT 4x4.

Por otro lado, cuando se codifican coeficientes de DCT 8x8 en H.264 por medio de CABAC, se reordenan 64 coeficientes en una forma unidimensional, y se codifican. Cuando se estima la cantidad de código correspondiente (requerida cuando se aplica CABAC a la codificación) usando una tabla de VLC para CAVLC, la ordenación de datos unidimensionales, es decir, la forma de barrido, es diferente entre CABAC y CAVLC.

En CABAC, simplemente se barren en zigzag secuencialmente 64 coeficientes. En cambio, en CAVLC, se realizan cuatro operaciones de barrido para 16 coeficientes para producir cuatro elementos de datos unidimensionales.

Por tanto, en la técnica convencional que usa CAVLC para la estimación de la cantidad de código de CABAC, no sólo debe ejecutarse el barrido para CABAC, sino también el barrido para CAVLC sólo para la estimación de la cantidad de código, lo que aumenta el coste computacional.

A la vista de las circunstancias anteriores, un objeto de la presente invención es proporcionar una técnica de codificación novedosa para estimar con precisión la cantidad de código generado con una cantidad de operación menor que la técnica convencional descrita anteriormente.

Medios para solucionar el problema

Con el fin de lograr el objeto anterior, la presente invención se refiere a un sistema de codificación de vídeo que tiene dos métodos de implementación para codificar coeficientes de una transformación ortogonal de gran tamaño, tales como: un primer método de codificación de fuente de información en el que se realiza la codificación tras reordenar los coeficientes para tener una forma unidimensional; y un segundo método de codificación de fuente de información en el que se dividen los coeficientes en una pluralidad de grupos de ordenación unidimensionales y se realiza la codificación usando una tabla de codificación de longitud variable para una transformación ortogonal de menor tamaño. La presente invención tiene:

(i) un dispositivo que realiza una transformación ortogonal de mayor tamaño, reordena los coeficientes cuantificados para tener una forma unidimensional en el orden definido en el primer método de codificación de longitud variable, y almacena conjuntos Ejecución-Nivel del número Ejecución de coeficientes “0” sucesivos y un coeficiente significativo Nivel que los sigue;

(ii) un dispositivo que calcula el número de grupos basándose en una proporción de área entre el tamaño de

transformación ortogonal que tiene la tabla de codificación de longitud variable y un tamaño objetivo de transformación ortogonal;

(iii) un dispositivo que clasifica los conjuntos Ejecución-Nivel en grupos que tienen el número de grupos;

(iv) un dispositivo que divide cada Ejecución entre el número de grupos, y fija el cociente obtenido como Ejecución;

(v) un dispositivo que determina una longitud de código de cada conjunto Ejecución-Nivel en cada grupo haciendo referencia a la tabla de codificación de longitud variable del segundo método de codificación de fuente de información; y

(vi) un dispositivo que calcula la suma total de las longitudes de código determinadas,

en el que se estima que la cantidad de código generado en el primer método de codificación de fuente de información es la suma total de las longitudes de código de todos los grupos.

Lo que sigue son ejemplos para clasificar los conjuntos Ejecución-Nivel en los grupos:

(i) un primer ejemplo incluye asignar un número de índice a cada conjunto Ejecución-Nivel en el orden de detección de los conjuntos Ejecución-Nivel, dividir cada número de índice entre el número de grupos, y clasificar los conjuntos Ejecución-Nivel que tienen el mismo resto para la división en el mismo grupo; y

(ii) un segundo ejemplo incluye acumular un valor obtenido añadiendo 1 a cada Ejecución en el orden de detección de los conjuntos Ejecución-Nivel, dividir cada valor acumulado entre el número de grupos, y clasificar los conjuntos Ejecución-Nivel que tienen el mismo resto para la división en el mismo grupo.

Efecto de la invención

Según la presente invención, cuando se realiza la estimación de la cantidad de código usando una tabla de codificación para una transformación ortogonal de menor tamaño que la transformación ortogonal realmente realizada, la estimación de la cantidad de código puede ejecutarse de manera altamente precisa al tiempo que se reduce la cantidad de cálculo.

Breve descripción de los dibujos

La figura 1 es un diagrama que muestra un ejemplo de clasificación en grupos de conjuntos Ejecución-Nivel en DCT 8x8 en cuatro grupos.

La figura 2 es un diagrama que muestra otro ejemplo de clasificación en grupos de conjuntos Ejecución-Nivel en DCT 8x8 en cuatro grupos.

La figura 3 es un gráfico que muestra resultados de estimación de la cantidad de código cuando se aplica la presente invención a H.264.

La figura 4 es un diagrama de flujo de un procesador de precodificación en una realización de la presente invención.

La figura 5 es un diagrama de flujo que muestra un ejemplo de proceso de clasificación de coeficientes de DCT 8x8 en cuatro modos.

La figura 6 es un diagrama que muestra un ejemplo de la estructura de un dispositivo según la presente invención.

La figura 7A es un diagrama de flujo de una rutina principal en una operación de codificación mediante la cual puede usarse CAVLC para la estimación de la cantidad de código de CABAC.

La figura 7B es un diagrama de flujo de un proceso de CABAC en la operación de codificación.

La figura 8 es un diagrama que muestra un ejemplo de un dispositivo de codificación para implementar la operación de las figuras 7A y 7B.

La figura 9 es un diagrama que muestra un orden de barrido para cada bloque 4x4.

La figura 10 es un diagrama que muestra un ejemplo de barrido en zigzag.

La figura 11 es un diagrama que muestra un ejemplo de un flujo de operación ejecutado por un procesador de precodificación para un bloque 4x4.

La figura 12 es un diagrama que muestra un ejemplo de un flujo de operación para barrido en zigzag.

La figura 13 es un diagrama que muestra un ejemplo de la estructura del procesador de precodificación en la figura 8.

La figura 14 es un diagrama que muestra un orden de barrido de CABAC.

La figura 15 es un diagrama que muestra un orden de barrido de CAVLC.

La figura 16 es un diagrama de flujo de un procesador de precodificación para DCT 8x8 según una técnica convencional.

La figura 17 es un diagrama que muestra un ejemplo de estructura del procesador de precodificación para DCT 8x8 según la técnica convencional.

1 memoria intermedia de almacenamiento de valor cuantificado 8x8

2 controlador de proceso de precodificación

3 contador de barrido 8x8

4 tabla de referencia de orden de barrido 8x8

5 contador de ejecución

6 memoria intermedia de almacenamiento de información de ejecución-nivel “B”

7 gestor de número de modo

8 contador

9 ordenador de número de conjunto

10 controlador comparativo

11 unidad de estimación de la cantidad de código

12 operador de desplazamiento

13 memoria intermedia de almacenamiento de información de ejecución-nivel

14 controlador de estimación de la cantidad de código

15 memoria de almacenamiento de tabla de VLC

16 unidad de cálculo de cantidad de código estimada

Mejor modo de llevar a cabo la invención

Según la presente invención, 64 coeficientes de CABAC pueden producir resultados de barrido de CAVLC. Es decir, se omite un proceso para realizar independientemente un barrido de CAVLC, y se producen resultados de barrido de CAVLC usando resultados de barrido de CABAC.

En un ejemplo de método de cálculo del número de grupos en la presente invención, el área de una transformación ortogonal de gran tamaño se divide entre el área de una transformación ortogonal de pequeño tamaño, y el cociente obtenido se fija como el número de grupos.

Por ejemplo, si se dividen coeficientes de DCT 8x8 en elementos de DCT 4x4, 64/16=4 y el número de grupos es 4.

Adicionalmente, en un ejemplo de método de división de conjuntos Ejecución-Nivel en una pluralidad de grupos, se asignan secuencialmente números de índice a los conjuntos Ejecución-Nivel en el orden de detección en el barrido de los conjuntos, y cada número de índice se divide entre el número de grupos, para usar el resto obtenido para la clasificación en grupos de los conjuntos Ejecución-Nivel.

En H.264, deben clasificarse coeficientes de DCT 8x8 en cuatro grupos. En el método anterior, tras el barrido de CABAC para DCT 8x8, se clasifican los conjuntos Ejecución-Nivel relevantes en cuatro grupos que incluyen: un

primer grupo generado extrayendo los conjuntos Ejecución-Nivel de orden 1, 5,..., (4i+1) (i=0, 1, ...); un segundo grupo generado extrayendo los conjuntos Ejecución-Nivel de orden 2, 6, ..., (4i+2) (i=0, 1, ...); un tercer grupo generado extrayendo los conjuntos Ejecución-Nivel de orden, 7, ..., (4i+3) (i=0, 1, ...); y un cuarto grupo generado extrayendo los conjuntos Ejecución-Nivel de orden 4, 8, ..., (4i+4) (i=0, 1,...).

Es decir, el ejemplo de clasificación de conjuntos Ejecución-Nivel en cuatro grupos incluye (i) un proceso de asignación de números a los conjuntos Ejecución-Nivel en el orden de detección de los resultados (es decir, los conjuntos Ejecución-Nivel) del barrido de CABAC, y asignar igualmente números de grupo 1, 2, 3, 4, 1, 2, 3, 4, ... a los números asignados anteriores desde el más pequeño hasta el más grande, y (ii) un proceso de disminución de la longitud de cada Ejecución a un cuarto de la misma.

Según la presente invención, se clasifican conjuntos Ejecución-Nivel obtenidos para una DCT de gran tamaño en grupos correspondientes a un tamaño de transformación ortogonal, que tiene una tabla de codificación, y cada Ejecución de todos los conjuntos Ejecución-Nivel se divide entre el número de grupos, de modo que el cociente obtenido se fija como Ejecución. Por consiguiente, los conjuntos Ejecución-Nivel obtenidos para una DCT de gran tamaño se pseudo-clasifican en grupos de conjuntos Ejecución-Nivel correspondientes a un tamaño de transformación ortogonal, que tiene una tabla de codificación.

La figura 1 muestra un ejemplo de clasificación en grupos de coeficientes de DCT 8x8 en cuatro grupos de DCT 4x4 usando el resto para 4.

En cada grupo obtenido, los conjuntos Ejecución-Nivel se consideran información de Ejecución-Nivel para un tamaño de DCT 4x4. La cantidad de código requerida para ellos se calcula haciendo referencia a una tabla de CAVLC, y se emite la suma de cuatro cantidades como cantidad de código estimada.

En otro ejemplo de clasificación de los conjuntos Ejecución-Nivel en una pluralidad de grupos, cada valor calculado añadiendo 1 a la Ejecución de cada conjunto Ejecución-Nivel se acumula en el orden de detección de barrido, y los conjuntos se clasifican basándose en el resto obtenido dividiendo el valor acumulado de cada conjunto (incluyendo el valor acumulado el valor calculado para el conjunto relevante) entre el número de grupos.

La figura 2 muestra un ejemplo de clasificación en cuatro grupos usando el resto para 4. Cada valor acumulado se obtiene acumulando secuencialmente un valor calculado añadiendo 1 a cada Ejecución. Cada valor acumulado se divide entre 4 para obtener el resto. Los conjuntos Ejecución-Nivel que tienen un resto de 1 se asignan al grupo 1; los conjuntos Ejecución-Nivel que tienen un resto de 2 se asignan al grupo 2; los conjuntos Ejecución-Nivel que tienen un resto de 3 se asignan al grupo 3; y los conjuntos Ejecución-Nivel que tienen un resto de 0 se asignan al grupo 4.

Tal como se describió anteriormente, en la presente invención, los resultados (es decir, conjuntos Ejecución-Nivel) del barrido de CABAC pueden clasificarse en cuatro grupos, para producir cuatro pseudo-elementos de datos unidimensionales. Por consiguiente, puede usarse CAVLC sin realizar un nuevo barrido para estimar la cantidad de código.

Por tanto, la presente invención puede reducir el coste computacional y estimar la cantidad de código con una alta precisión.

A continuación, se explicará una realización específica de la presente invención con referencia a los dibujos.

En este caso, en un dispositivo de codificación que usa la estimación de la cantidad de código según la presente invención, se explican principalmente elementos estructurales distintivos de la presente invención, mientras que se omiten explicaciones detalladas de los demás elementos, que son iguales a elementos correspondientes de la técnica convencional y por tanto ya se han explicado.

En la siguiente realización, la cantidad de código requerida cuando se codifican coeficientes de DCT 8x8 por medio de CABAC se estima usando una tabla para DCT 4x4.

El número de grupos se fija a “4” lo que se obtiene dividiendo el área de DCT 8x8 entre el área de DCT 4x4, y la información de Ejecución-Nivel se clasifica en cuatro grupos usando el resto para 4 de cada número de índice.

La figura 4 es un diagrama de flujo de un procesador de precodificación en la presente realización.

En primer lugar, se inicializa la cantidad de Tasa de transmisión de código estimada a 0 (véase la etapa S1).

A continuación, se somete un bloque de 8x8 a barrido en zigzag (véase la etapa S2), de modo que se genera información de Ejecución-Nivel. Este proceso se muestra en la figura 12, y una tabla de referencia de orden de barrido devuelve coordenadas en el orden mostrado en la figura 14. La información de Ejecución-Nivel del bloque de 8x8 se envía como información de codificación a un codificador de entropía (véase la etapa S3).

A continuación, se clasifica la información de Ejecución-Nivel obtenida para el bloque de 8x8 en cuatro modos (de modo 1 a modo 4) (véase la etapa S4). En la figura 5 se muestra un ejemplo de este proceso.

Tal como se muestra en la figura 5, en primer lugar, se fija la variable m que indica el número de modo a 1 (véase la etapa S21), y se genera información de Ejecución-Nivel de modo 1 mediante la siguiente operación.

Después se inicializa la variable i a 0 (véase la etapa S22), y se fija la variable n a “4*i + m”, donde “*” indica la multiplicación (véase la etapa S23).

Si n es inferior al número N de conjuntos de información de Ejecución-Nivel obtenido mediante DCT 8x8 (es decir, el número de coeficientes significativos) (véase la etapa S24), entonces para la información de Ejecución-Nivel de orden n (Ejecución[n] y Nivel[n]), se desplaza Ejecución[n] a la derecha 2 bits, y se almacena el resultado en Ejecución_t[m][i], mientras que Nivel[n] se almacena en Nivel_t[m][i]. Adicionalmente, se incrementa i en 1 (véase la etapa S25).

La operación anterior (etapas de S23 a S25) se realiza de manera repetida.

Cuando n supera a N (véase la etapa S24), se conmuta el modo al siguiente modo.

Tras confirmar que el número de modo m es menor que 4 (véase la etapa S26), se incrementa m en 1 (véase la etapa S27), y se repite la operación descrita anteriormente. Cuando se ha procesado el modo final 4, se completa el proceso de clasificación.

Por consiguiente, el conjunto de orden i de información de Ejecución-Nivel del número de modo m se almacena en Ejecución_t[m][i] y Nivel_t[m][i].

Volviendo ahora a la figura 4, se fija de nuevo la variable m a 1 (véase la etapa S5). Para el modo 1, se obtiene información de codificación (es decir, el número de coeficientes distintos de cero, el número de sucesión final de coeficiente “1” o “-1”, y el signo positivo o negativo para la misma) distinta de los conjuntos Ejecución-Nivel usando cada conjunto de Ejecución_t[m][i] y Nivel_t[mj [i] (i=0, 1,...) (véase la etapa S6), y se calcula la cantidad r_tmp de código usando una tabla de VLC (véase la etapa S7).

La cantidad r_tmp de código calculada se añade a la cantidad Tasa de transmisión de código estimada (véase la etapa S8). Entonces, si el número de modo m es inferior a 4 (véase la etapa S9), se incrementa m en 1 (véase la etapa S10), y se repite la operación descrita anteriormente para el siguiente modo (véanse las etapas de S6 a S10).

Finalmente, se envía la cantidad Tasa de transmisión de código estimada a un controlador de cantidad de código (véase la etapa S11).

Puede implementarse una operación según la presente invención usando un diagrama de flujo tal como se explicó anteriormente.

A continuación, se mostrará un ejemplo de la estructura para ejecutar el presente diagrama de flujo.

La estructura del dispositivo de codificación relevante puede mostrarse similar a la figura 8 para la técnica convencional. La presente realización se aplica al procesador 109 de precodificación indicado mediante un bloque en negrita.

Por tanto, la figura 6 muestra un ejemplo de la estructura del procesador 109 de precodificación. La presente invención se aplica a la parte rodeada por una línea discontinua en negrita.

El procesador de precodificación de la presente realización tiene una memoria intermedia 1 de almacenamiento de valor cuantificado 8x8, un controlador 2 de proceso de precodificación, un contador 3 de barrido 8x8, una tabla 4 de referencia de orden de barrido 8x8, un contador 5 de ejecución, una memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B”, un gestor 7 de número de modo, un contador 8, un ordenador 9 de número de conjunto, un controlador comparativo 10, una unidad 11 de estimación de la cantidad de código, un operador 12 de desplazamiento, una memoria 13 intermedia de almacenamiento de información de Ejecución-Nivel, un controlador 14 de estimación de la cantidad de código, una memoria 15 de almacenamiento de tabla de VLC, y una unidad 16 de cálculo de cantidad de código estimada.

Entre los elementos estructurales anteriores, la memoria intermedia 1 de almacenamiento de valor cuantificado 8x8, el controlador 2 de proceso de precodificación, el contador 3 de barrido 8x8, la tabla 4 de referencia de orden de barrido 8x8, el contador 5 de ejecución, la unidad 11 de estimación de la cantidad de código, el controlador 14 de estimación de la cantidad de código, la memoria 15 de almacenamiento de tabla de VLC, la memoria 13 intermedia de almacenamiento de información de Ejecución-Nivel, y la unidad 16 de cálculo de cantidad de código estimada

tienen funciones iguales a las de los elementos estructurales descritos anteriormente que tienen los mismos nombres.

Cuando la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B” recibe una señal de reinicio del controlador 2 de proceso de precodificación, la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B” inicializa la información almacenada.

A continuación, cuando se recibe información de Ejecución-Nivel del contador 5 de ejecución, la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B” almacena la información, y envía el número de los conjuntos Ejecución-Nivel, como información de conjunto N, al controlador comparativo 10.

También cuando se recibe el número de conjunto “n” del controlador comparativo 10, la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B” envía el conjunto de información de Ejecución-Nivel de orden n al operador 12 de desplazamiento.

Cuando el gestor 7 de número de modo recibe una señal de inicio de estimación del controlador 2 de proceso de precodificación, el gestor 7 de número de modo fija el número de modo “m” a 1, y lo envía al contador 8 y al ordenador 9 de número de conjunto.

Además, cuando se recibe una señal de control del controlador comparativo 10, el gestor 7 de número de modo incrementa el número de modo m en 1, y envía el número incrementado al contador 8 y al ordenador 9 de número de conjunto.

Además, cuando se recibe la señal de control tras enviar “4” como el número de modo m, el gestor 7 de número de modo envía el número de modo “0” al contador 8 y al ordenador 9 de número de conjunto, para detener la operación relevante hasta que se recibe de nuevo una señal de inicio de estimación.

Cuando el contador 8 recibe el número de modo m (es decir, de 1 a 4), el contador 8 reinicia “i” a 0, y envía i al ordenador 9 de número de conjunto al tiempo que incrementa i (desde 0) en 1. Cuando se recibe 0 como número de modo, el contador 8 detiene su operación.

Cuando el ordenador 9 de número de conjunto recibe el número de modo m del gestor 7 de número de modo y el valor i del contador 8, el ordenador 9 de número de conjunto calcula el número de conjunto n mediante “n=4*i + m”, y envía el número de conjunto n al controlador comparativo 10.

El controlador comparativo 10 compara el número de conjunto n enviado desde el ordenador 9 de número de conjunto con el número N enviado desde la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B”. Si n es inferior o igual a N, el controlador comparativo 10 envía el número de conjunto n a la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B”. Si n es superior a N, el controlador comparativo 10 envía una señal de control al gestor 7 de número de modo, y también envía una señal de inicio de estimación al controlador 14 de estimación de la cantidad de código.

Cuando el operador 12 de desplazamiento recibe la información de Ejecución- Nivel de la memoria intermedia 6 de almacenamiento de información de Ejecución-Nivel “B”, el operador 12 de desplazamiento desplaza el valor de Ejecución a la derecha 2 bits, y entonces envía los valores de Ejecución y Nivel como conjunto a la memoria 13 intermedia de almacenamiento de información de Ejecución-Nivel.

Empleando la estructura anterior, puede implementarse la operación mostrada en las figuras 4 y 5.

La operación de estimación de la cantidad de código descrita anteriormente también puede implementarse mediante un ordenador y un programa de software. Un programa informático de este tipo puede proporcionarse almacenándolo en un medio de almacenamiento legible por ordenador apropiado, o por medio de una red.

El efecto de reducir la cantidad de cálculo según la presente invención es considerable especialmente cuando el número de coeficientes significativos es pequeño.

Por ejemplo, en H.264, incluso cuando sólo se obtiene un conjunto de información de Ejecución- Nivel mediante DCT 8x8, una operación de barrido en zigzag (véase la figura 14) para la estimación de la cantidad de código debe ejecutarse cuatro veces en la técnica convencional. En cambio, en la presente invención, la cantidad de código puede estimarse sólo sometiendo el único conjunto de información de Ejecución-Nivel a un desplazamiento de 2 bits y haciendo referencia a una tabla.

La figura 3 muestra resultados de estimación de la cantidad de código cuando se aplica la presente invención a la codificación en H. 264.

Específicamente, la figura 3 muestra la representación gráfica para todos los macrobloques cuando se realiza DCT

8x8, en la que el eje horizontal indica cada cantidad de código estimada según la presente invención, y el eje vertical indica cada cantidad de código generada cuando se usa CABAC.

Se realizó una clasificación en grupos usando el resto de 4.

Los resultados mostrados indican que el valor estimado por la presente invención es proporcional a la cantidad de código generado real mediante CABAC, y la estimación de la cantidad de código puede realizarse con precisión mediante un método según la presente invención.

A continuación se muestra un motivo para realizar con precisión la estimación de la cantidad de código mediante un método de la presente invención.

En la especificación de CAVLC en H.264, el método de dividir 64 (8x8) elementos en cuatro grupos produce un resultado en el que se incluyen componentes de baja frecuencia y de alta frecuencia como que son igual de posibles en los cuatro grupos de datos unidimensionales divididos.

También en la presente invención, un método de clasificar resultados de barrido de CABAC (es decir, conjuntos Ejecución-Nivel) en cuatro grupos produce un resultado en el que se incluyen componentes de baja frecuencia y de alta frecuencia de los coeficientes relevantes como que son igual de posibles en los cuatro grupos.

Por tanto, tal como se muestra en los resultados de estimación de la cantidad de código en la figura 3, pueden obtenerse resultados de estimación de la cantidad de código altamente precisos mediante CAVLC.

Aplicabilidad industrial

Según la presente invención, cuando se realiza la estimación de la cantidad de código usando una tabla de codificación para una transformación ortogonal de menor tamaño que la transformación ortogonal realmente realizada, la estimación de la cantidad de código puede ejecutarse de manera altamente precisa al tiempo que se reduce la cantidad de cálculo.

REIVINDICACIONES

1. Método de estimación de la cantidad de código, usado en la codificación de vídeo, para estimar una cantidad de código generada en un primer método de codificación de fuente de información en el que puede seleccionarse una pluralidad de tamaños de transformación ortogonal y se realiza una codificación reordenando coeficientes de transformación ortogonal bidimensionales cuantificados para tener una forma unidimensional, en el que la cantidad de código se estima usando un segundo método de codificación de fuente de información que tiene un coste computacional inferior al primer método de codificación de fuente de información y realiza la codificación usando una tabla de codificación de longitud variable, y el método de estimación de la cantidad de código comprende las etapas de:

realizar una reordenación, cuando se codifica, según dicho primer método de codificación de fuente de información, de valores cuantificados de coeficientes de una transformación ortogonal de mayor tamaño que un tamaño de transformación ortogonal asignado a dicha tabla de codificación de longitud variable según dicho segundo método de codificación de fuente de información, reordenando los valores cuantificados para tener una forma unidimensional en el orden definido en el primer método de codificación de fuente de información, en el que en la forma unidimensional, se obtienen conjuntos de Ejecución-Nivel del número Ejecución de coeficientes “0” sucesivos y un coeficiente significativo Nivel que los sigue, y se almacenan los conjuntos Ejecución-Nivel obtenidos;

con el fin de dividir los conjuntos Ejecución-Nivel en una pluralidad de grupos de conjuntos Ejecución-Nivel, calcular el número de grupos basándose en una proporción entre un área de transformación ortogonal correspondiente al tamaño de transformación ortogonal asignado a dicha tabla de codificación de longitud variable según dicho segundo método de codificación de fuente de información y un área de transformación ortogonal seleccionada mediante dicho primer método de codificación de fuente de información;

clasificar los conjuntos Ejecución-Nivel en grupos cuyo número es dicho número de grupos;

dividir la Ejecución de cada conjunto Ejecución-Nivel entre el número de grupos, y fijar el cociente obtenido como Ejecución del conjunto Ejecución-Nivel;

determinar una longitud de código de cada conjunto Ejecución-Nivel en cada grupo haciendo referencia a dicha tabla de codificación de longitud variable según dicho segundo método de codificación de fuente de información; y

calcular la suma total de las longitudes de código determinadas,

en el que se estima que la cantidad de código generada en el primer método de codificación de fuente de información es la suma total de las longitudes de código de todos los grupos.

2. Método de estimación de la cantidad de código según la reivindicación 1, en el que la etapa de clasificar los conjuntos Ejecución-Nivel en los grupos incluye asignar un número de índice a cada conjunto Ejecución-Nivel en el orden de detección de los conjuntos Ejecución-Nivel, dividir cada número de índice entre el número de grupos, y clasificar los conjuntos Ejecución-Nivel que tienen el mismo resto para la división en el mismo grupo.

3. Método de estimación de la cantidad de código según la reivindicación 1, en el que la etapa de clasificar los conjuntos Ejecución-Nivel en los grupos incluye acumular un valor obtenido añadiendo 1 a cada Ejecución en el orden de detección de los conjuntos Ejecución-Nivel, dividir cada valor acumulado entre el número de grupos, y clasificar los conjuntos Ejecución-Nivel que tienen el mismo resto para la división en el mismo grupo.

4. Dispositivo de estimación de la cantidad de código, usado en la codificación de vídeo, para estimar una cantidad de código generada en un primer método de codificación de fuente de información en el que puede seleccionarse una pluralidad de tamaños de transformación ortogonal y se realiza la codificación reordenando coeficientes de transformación ortogonal bidimensionales cuantificados para tener una forma unidimensional, en el que se estima la cantidad de código usando un segundo método de codificación de fuente de información que tiene un coste computacional inferior al del primer método de codificación de fuente de información y realiza la codificación usando una tabla de codificación de longitud variable, y el dispositivo de estimación de la cantidad de código comprende:

un dispositivo que realiza la reordenación, cuando se codifica, según dicho primer método de codificación de fuente de información, de valores cuantificados de coeficientes de una transformación ortogonal de mayor tamaño que un tamaño de transformación ortogonal asignado a dicha tabla de codificación de longitud variable según dicho segundo método de codificación de fuente de información, reordenando los

valores cuantificados para tener una forma unidimensional en el orden definido en el primer método de codificación de fuente de información, en el que para la forma unidimensional, el dispositivo obtiene conjuntos Ejecución-Nivel del número Ejecución de coeficientes “0” sucesivos y un coeficiente significativo Nivel que los sigue, y almacena los conjuntos Ejecución-Nivel obtenidos;

un dispositivo que calcula, con el fin de dividir los conjuntos Ejecución-Nivel en una pluralidad de grupos de conjuntos Ejecución-Nivel, el número de grupos basándose en una proporción entre un área de transformación ortogonal correspondiente al tamaño de transformación ortogonal asignado a dicha tabla de codificación de longitud variable según dicho segundo método de codificación de fuente de información y un área de transformación ortogonal seleccionada mediante dicho primer método de codificación de fuente de información;

un dispositivo que clasifica los conjuntos Ejecución-Nivel en grupos cuyo número es dicho número de grupos;

un dispositivo que divide la Ejecución de cada conjunto Ejecución-Nivel entre el número de grupos, y fija el cociente obtenido como Ejecución del conjunto Ejecución-Nivel;

un dispositivo que determina una longitud de código de cada conjunto Ejecución-Nivel en cada grupo haciendo referencia a dicha tabla de codificación de longitud variable según dicho segundo método de codificación de fuente de información; y

un dispositivo que calcula la suma total de las longitudes de código determinadas,

en el que se estima que la cantidad de código generada en el primer método de codificación de fuente de información es la suma total de las longitudes de código de todos los grupos.

5. Programa informático que comprende código de software adaptado para realizar el método según la reivindicación 1 cuando se ejecuta en un ordenador.

6. Medio de almacenamiento legible por ordenador que almacena un programa informático que comprende código de software adaptado para realizar el método según la reivindicación 1 cuando se ejecuta en un ordenador.

FIG. 1

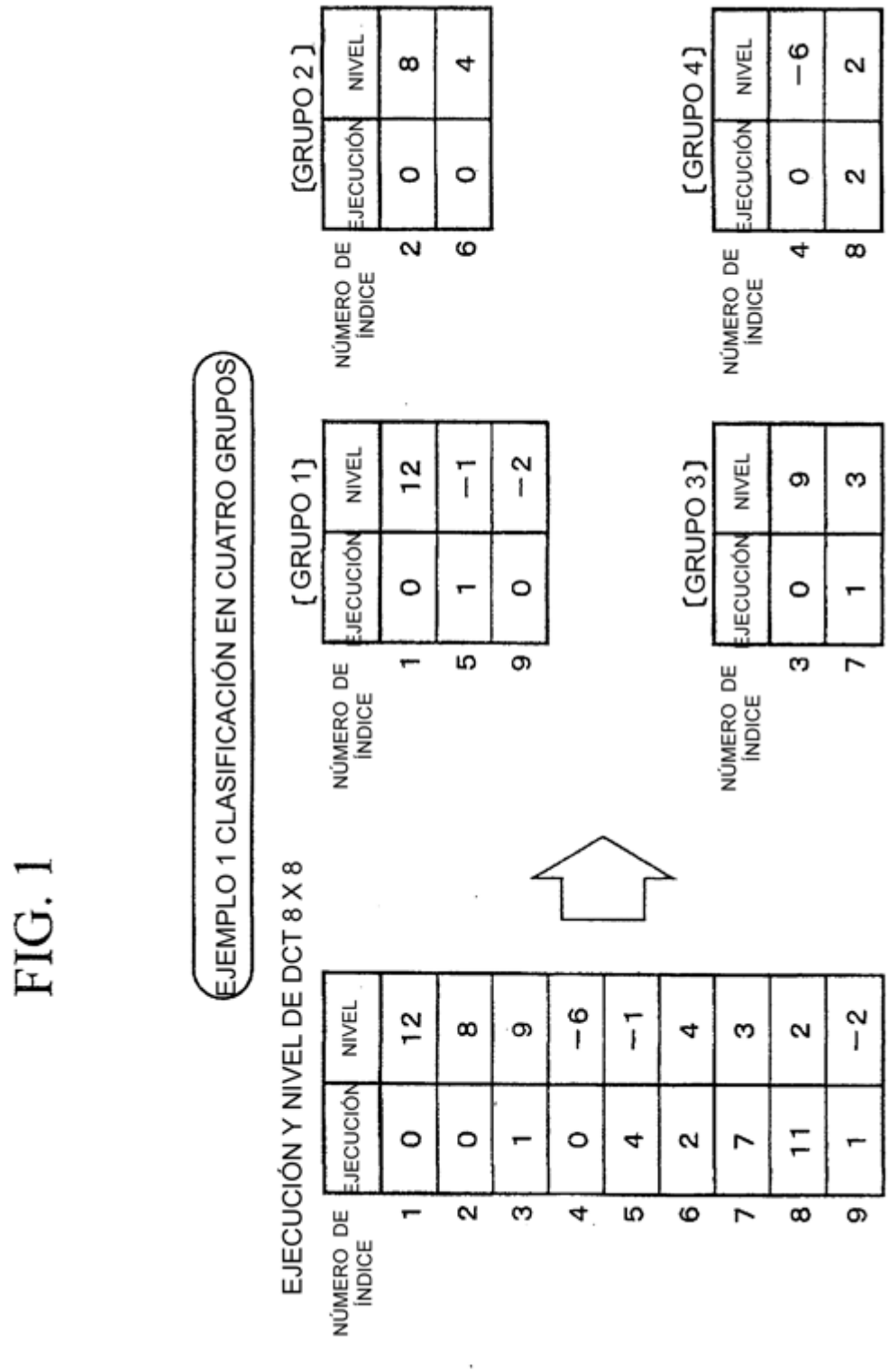
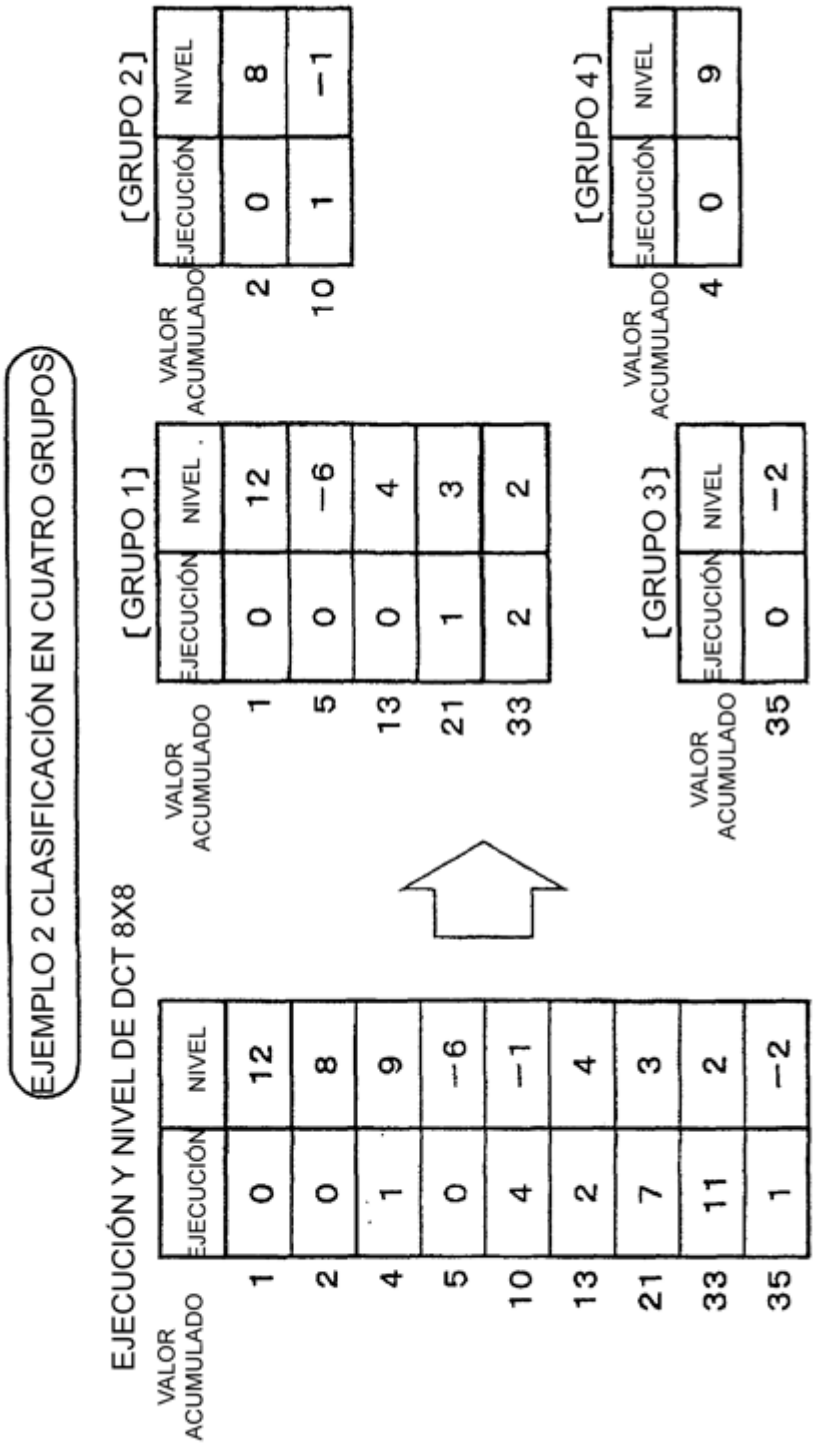

EJEMPLO 1 CLASIFICACIÓN EN CUATRO GRUPOS
EJECUCIÓN Y NIVEL DE DCT 8 X 8
NÚMERO DE ÍNDICE
EJECUCIÓN
NIVEL
1 0 12
2 0 8
3 1 9
4 0 −6
5 4 −1
6 2 4
7 7 3
8 11 2
9 1 −2
[GRUPO 1]
NÚMERO DE ÍNDICE
EJECUCIÓN
NIVEL
1 0 12
5 1 −1
9 0 −2
[GRUPO 2]
NÚMERO DE ÍNDICE
EJECUCIÓN
NIVEL
2 0 8
6 0 4
[GRUPO 3]
NÚMERO DE ÍNDICE
EJECUCIÓN
NIVEL
3 0 9
7 1 3
[GRUPO 4]
NÚMERO DE ÍNDICE
EJECUCIÓN
NIVEL
4 0 −6
8 2 2

FIG. 2


EJEMPLO 2 CLASIFICACIÓN EN CUATRO GRUPOS
EJECUCIÓN Y NIVEL DE DCT 8X8
VALOR ACUMULADO
EJECUCIÓN
NIVEL
1 0 12
2 0 8
4 1 9
5 0 −6
10 4 −1
13 2 4
21 7 3
33 11 2
35 1 −2
[GRUPO 1]
VALOR ACUMULADO
EJECUCIÓN
NIVEL
1 0 12
5 0 −6
13 0 4
21 1 3
33 2 2
[GRUPO 2]
VALOR ACUMULADO
EJECUCIÓN
NIVEL
2 0 8
10 1 −1
[GRUPO 3]
VALOR ACUMULADO
EJECUCIÓN
NIVEL
35 0 −2
[GRUPO 4]
VALOR ACUMULADO
EJECUCIÓN
NIVEL
4 0 9

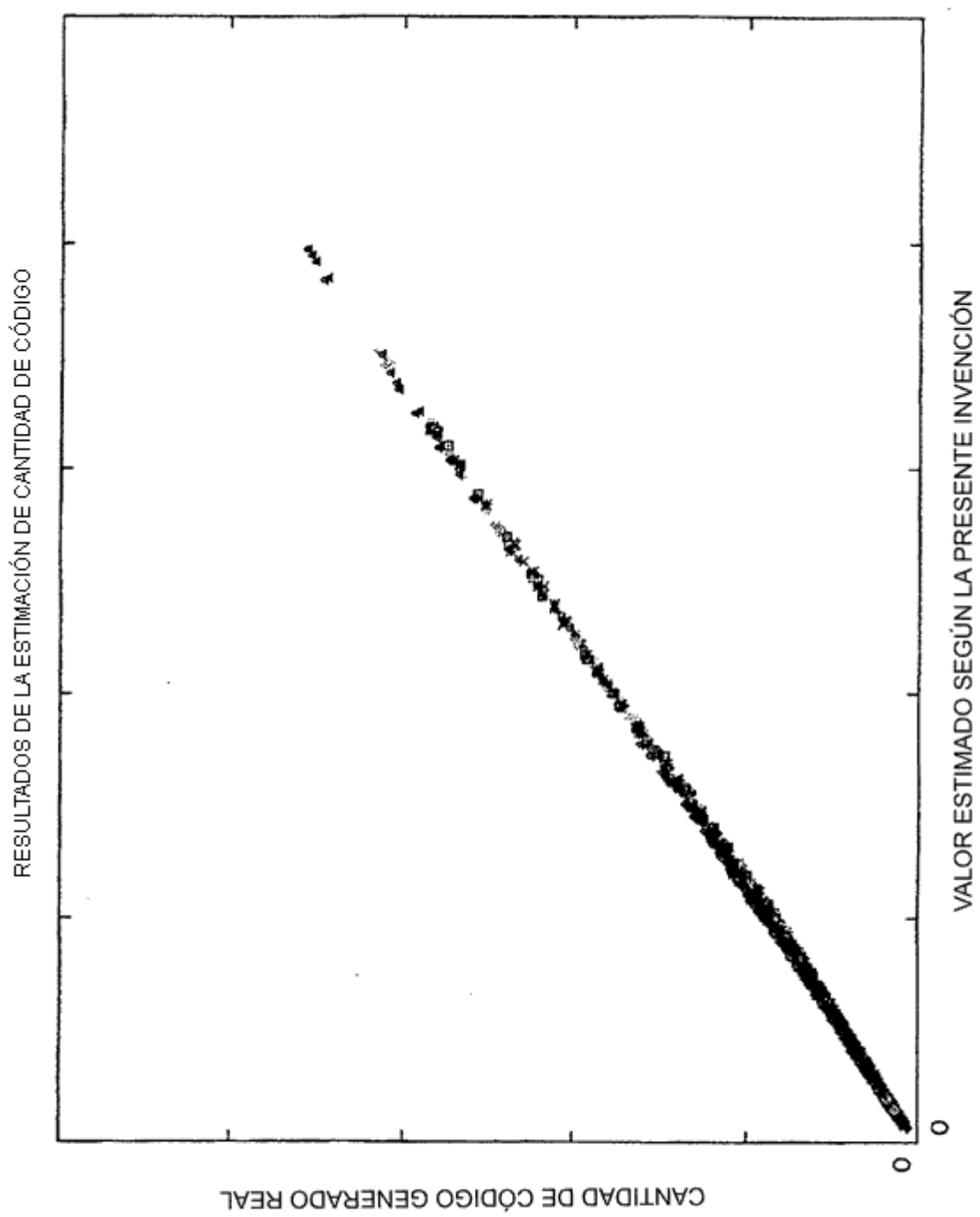

RESULTADOS DE LA ESTIMACIÓN DE CANTIDAD DE CÓDIGO
CANTIDAD DE CÓDIGO GENERADO REAL
VALOR ESTIMADO SEGÚN LA PRESENTE INVENCIÓN
0
0

FIG. 3

FIG. 4

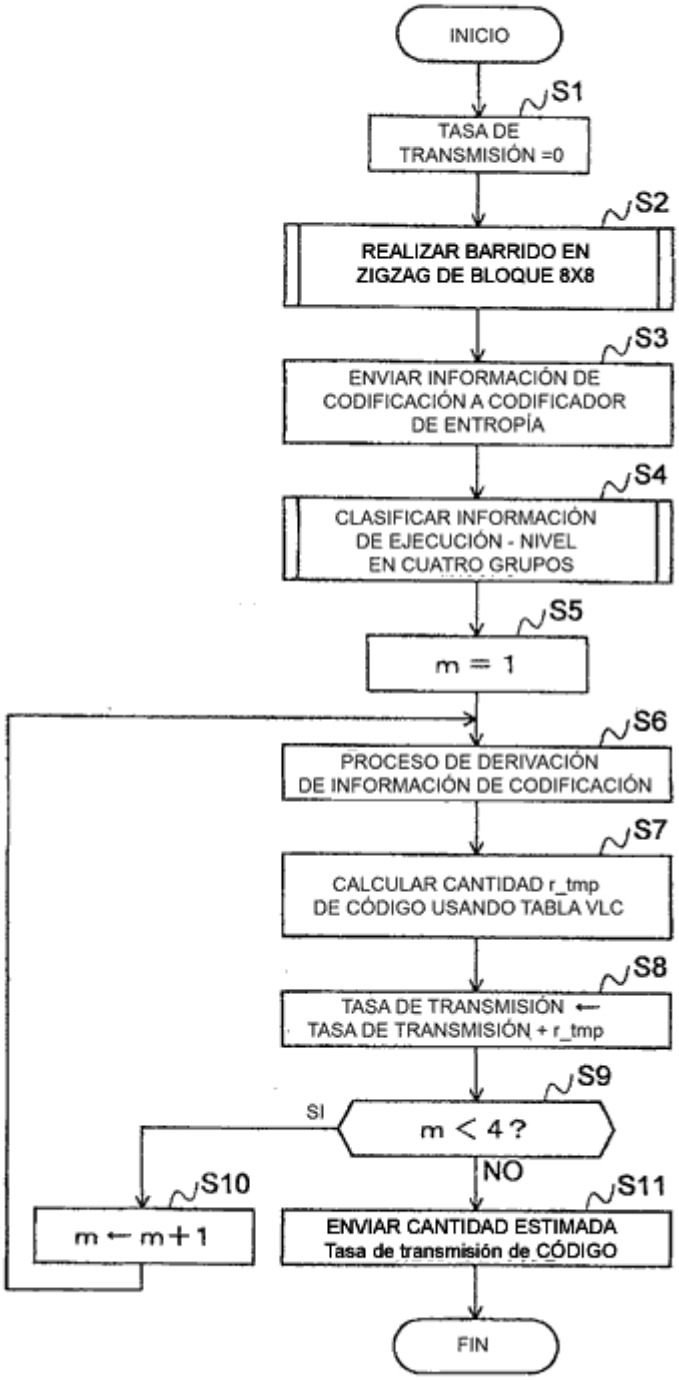

INICIO
S1
TASA DE TRANSMISIÓN =0
S2
REALIZAR BARRIDO EN ZIGZAG DE BLOQUE 8X8
S3
ENVIAR INFORMACIÓN DE CODIFICACIÓN A CODIFICADOR DE ENTROPÍA
S4
CLASIFICAR INFORMACIÓN DE EJECUCIÓN - NIVEL EN CUATRO GRUPOS
S5
m = 1
S6
PROCESO DE DERIVACIÓN DE INFORMACIÓN DE CODIFICACIÓN
S7
CALCULAR CANTIDAD r_tmp DE CÓDIGO USANDO TABLA VLC
S8
TASA DE TRANSMISIÓN ← TASA DE TRANSMISIÓN + r_tmp
S9
SI
m < 4 ?
NO
S10
m ← m+1
S11
ENVIAR CANTIDAD ESTIMADA Tasa de transmisión de CÓDIGO
FIN

FIG. 5

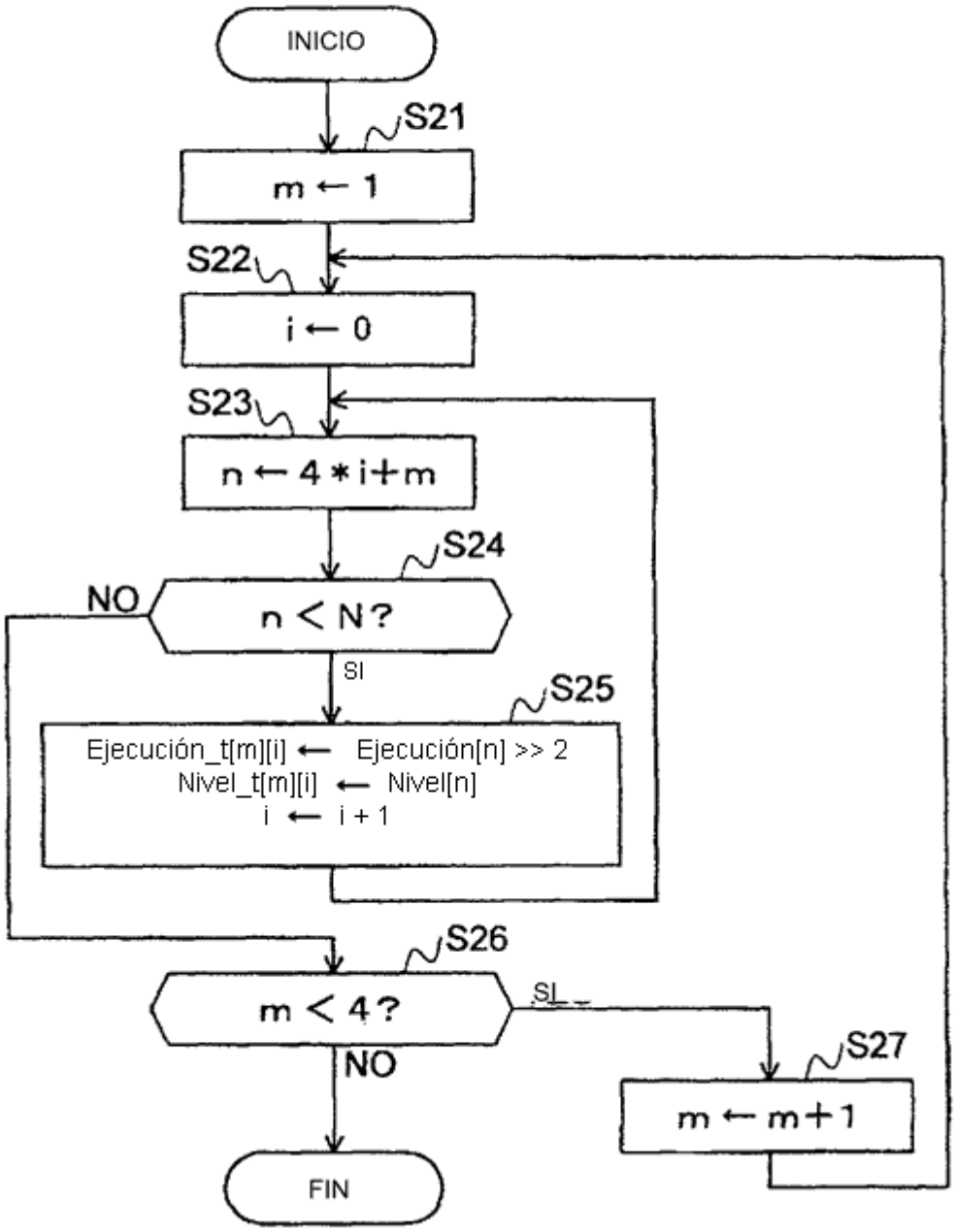

INICIO
S21
m ← 1
S22
i ← 0
S23
n ← 4 * i + m
S24
NO
n < N?
SI
S25
Ejecución_t[m][i] ← Ejecución[n] >> 2
Nivel_t[m][i] ← Nivel[n]
i ← i + 1
S26
m < 4?
SI
NO
S27
m ← m + 1
FIN

FIG. 6

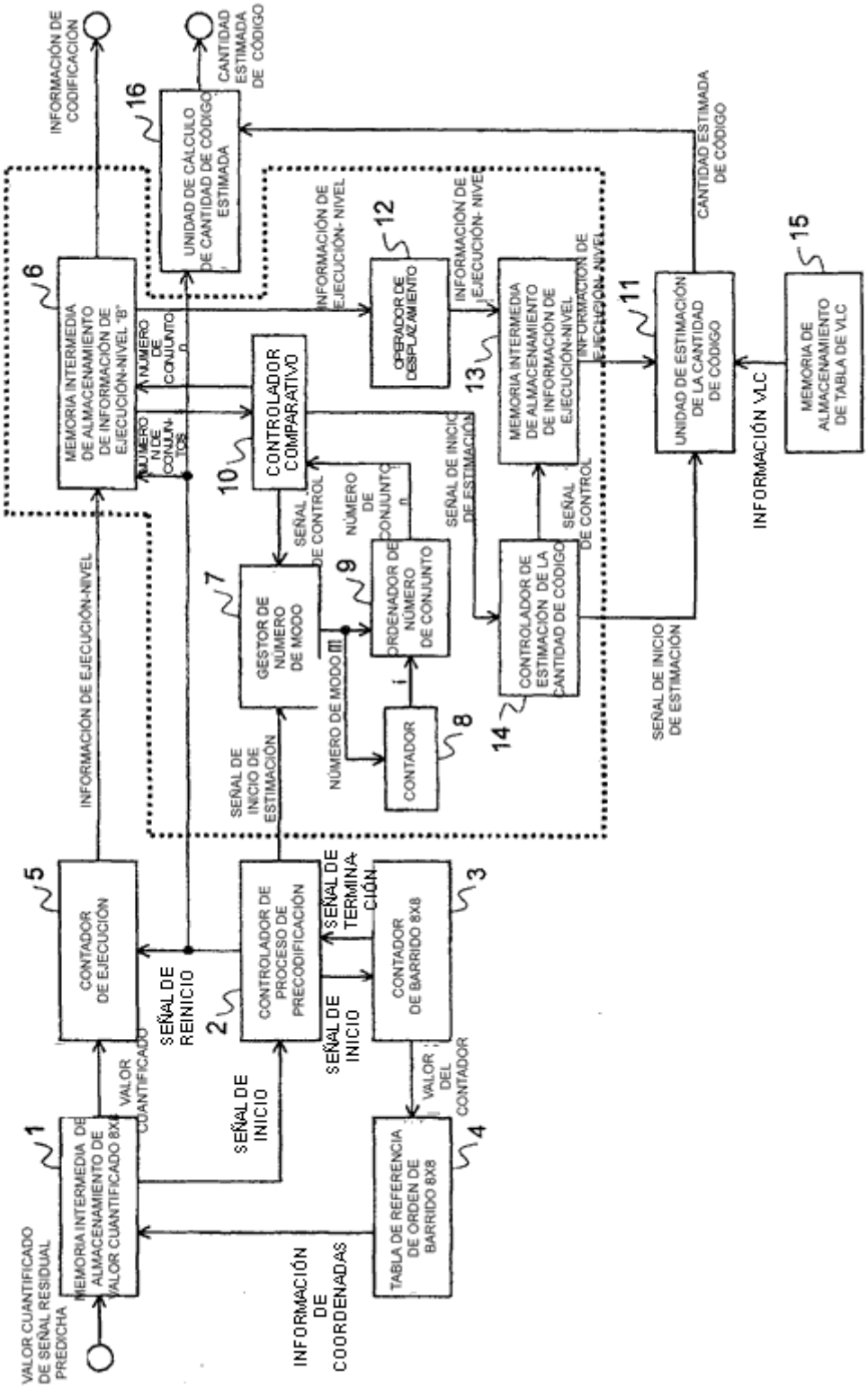

VALOR CUANTIFICADO DE SEÑAL RESIDUAL PREDICHA
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE VALOR CUANTIFICADO 8X8
1
VALOR CUANTIFICADO
CONTADOR DE EJECUCIÓN
5
SEÑAL DE REINICIO
CONTROLADOR DE PROCESO DE PRECODIFICACIÓN
2
SEÑAL DE INICIO
SEÑAL DE TERMINACIÓN
SEÑAL DE INICIO
CONTADOR DE BARRIDO 8X8
3
VALOR DEL CONTADOR
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 8X8
4
INFORMACIÓN DE COORDENADAS
INFORMACIÓN DE CODIFICACIÓN
INFORMACIÓN DE EJECUCIÓN-NIVEL
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE INFORMACIÓN DE EJECUCIÓN-NIVEL "B"
6
NÚMERO DE CONJUNTOS
NÚMERO DE CONJUNTO n
UNIDAD DE CÁLCULO DE CANTIDAD DE CÓDIGO ESTIMADA
16
CANTIDAD ESTIMADA DE CÓDIGO
CONTROLADOR COMPARATIVO
10
SEÑAL DE CONTROL
GESTOR DE NÚMERO DE MODO
7
NÚMERO DE MODO m
ORDENADOR DE NÚMERO DE CONJUNTO
9
NÚMERO DE CONJUNTO n
CONTADOR
i
8
SEÑAL DE INICIO DE ESTIMACIÓN
SEÑAL DE INICIO DE ESTIMACIÓN
INFORMACIÓN DE EJECUCIÓN- NIVEL
OPERADOR DE DESPLAZAMIENTO
12
INFORMACIÓN DE EJECUCIÓN- NIVEL
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE INFORMACIÓN DE EJECUCIÓN-NIVEL
13
INFORMACIÓN DE EJECUCIÓN-NIVEL
CONTROLADOR DE ESTIMACIÓN DE LA CANTIDAD DE CÓDIGO
14
SEÑAL DE CONTROL
SEÑAL DE INICIO DE ESTIMACIÓN
UNIDAD DE ESTIMACIÓN DE LA CANTIDAD DE CÓDIGO
11
CANTIDAD ESTIMADA DE CÓDIGO
INFORMACIÓN VLC
MEMORIA DE ALMACENAMIENTO DE TABLA DE VLC
15

FIG. 7A

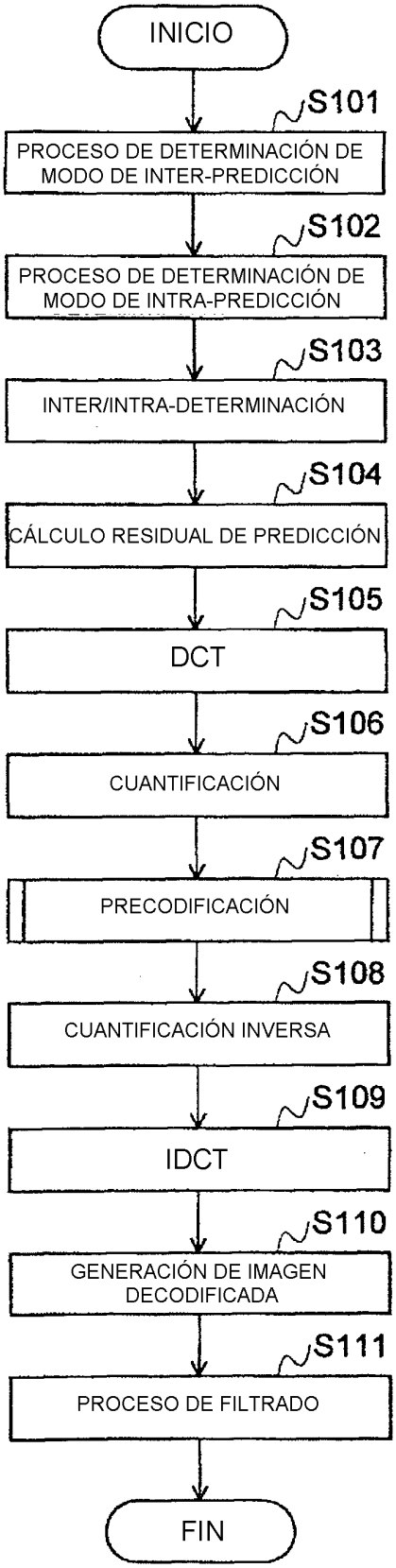

INICIO
S101
PROCESO DE DETERMINACIÓN DE MODO DE INTER-PREDICCIÓN
S102
PROCESO DE DETERMINACIÓN DE MODO DE INTRA-PREDICCIÓN
S103
INTER/INTRA-DETERMINACIÓN
S104
CÁLCULO RESIDUAL DE PREDICCIÓN
S105
DCT
S106
CUANTIFICACIÓN
S107
PRECODIFICACIÓN
S108
CUANTIFICACIÓN INVERSA
S109
IDCT
S110
GENERACIÓN DE IMAGEN DECODIFICADA
S111
PROCESO DE FILTRADO
FIN

FIG. 7B

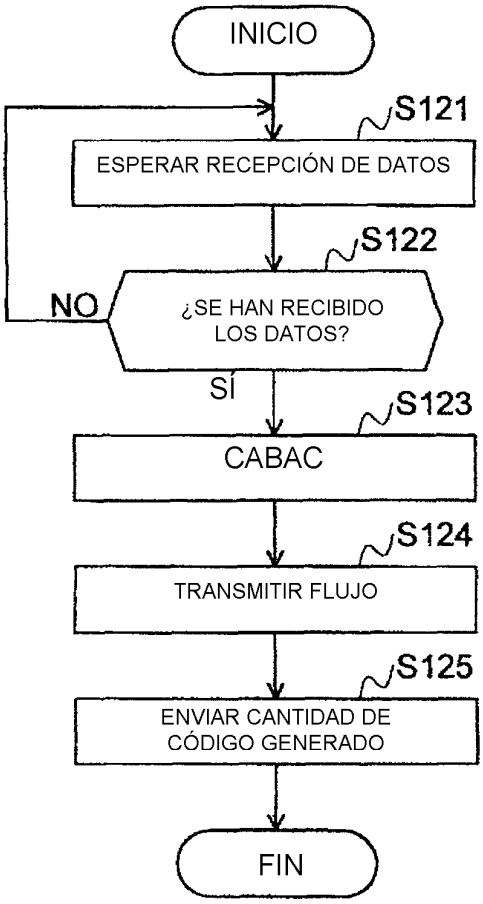

INICIO
S121
ESPERAR RECEPCIÓN DE DATOS
S122
NO
¿SE HAN RECIBIDO LOS DATOS?
SÍ
S123
CABAC
S124
TRANSMITIR FLUJO
S125
ENVIAR CANTIDAD DE CÓDIGO GENERADO
FIN

FIG. 8

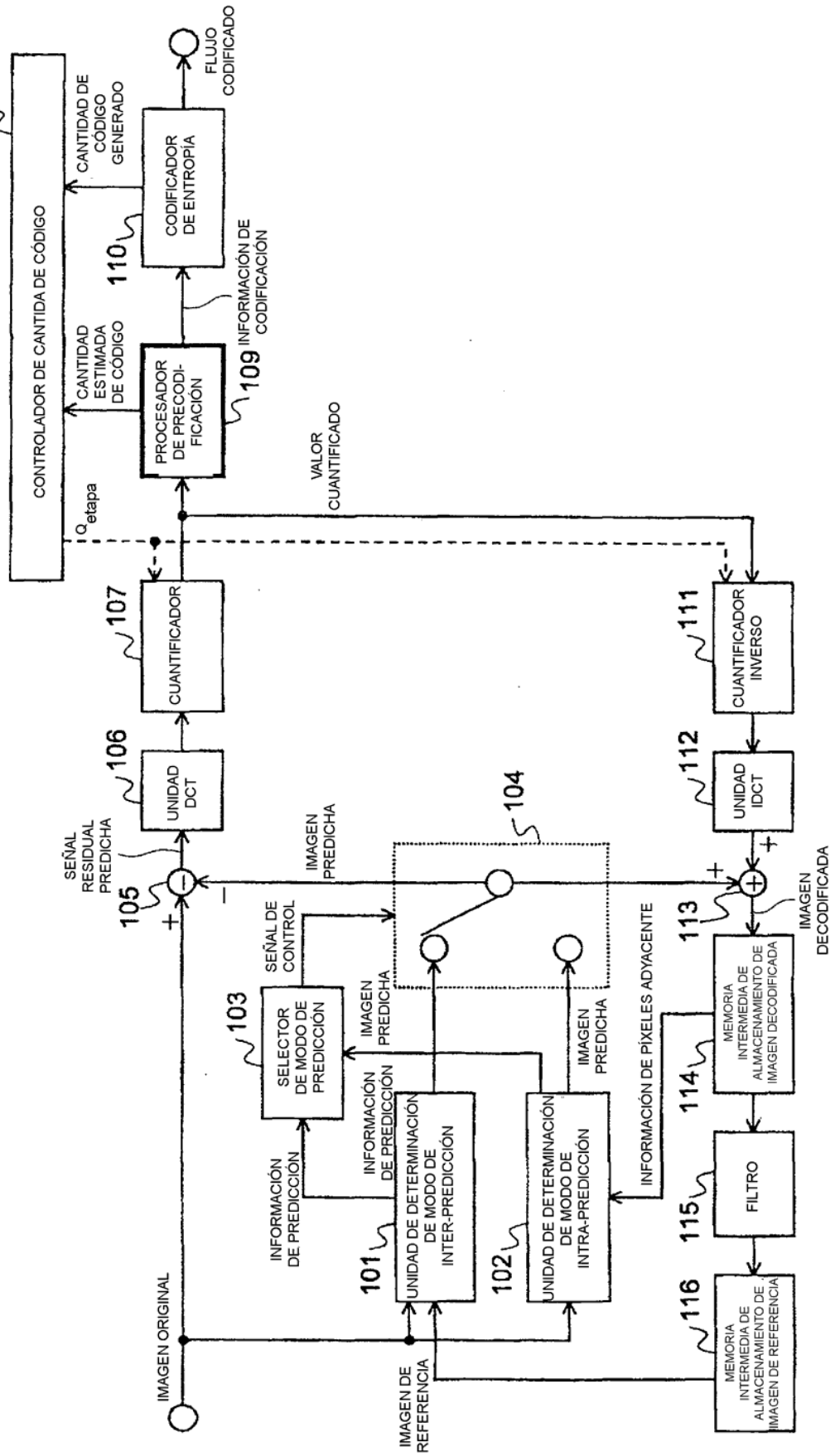

108
CONTROLADOR DE CANTIDA DE CÓDIGO
CANTIDAD DE CÓDIGO GENERADO
FLUJO CODIFICADO
110
CODIFICADOR DE ENTROPÍA
INFORMACIÓN DE CODIFICACIÓN
CANTIDAD ESTIMADA DE CÓDIGO
109
PROCESADOR DE PRECODIFICACIÓN
VALOR CUANTIFICADO
Qetapa
107
CUANTIFICADOR
111
CUANTIFICADOR INVERSO
106
UNIDAD DCT
112
UNIDAD IDCT
SEÑAL RESIDUAL PREDICHA
105
IMAGEN PREDICHA
104
IMAGEN DECODIFICADA
113
SEÑAL DE CONTROL
103
SELECTOR DE MODO DE PREDICCIÓN
IMAGEN PREDICHA
INFORMACIÓN DE PREDICCIÓN
INFORMACIÓN DE PÍXELES ADYACENTE
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE IMAGEN DECODIFICADA
114
INFORMACIÓN DE PREDICCIÓN
101
UNIDAD DE DETERMINACIÓN DE MODO DE INTER-PREDICCIÓN
102
UNIDAD DE DETERMINACIÓN DE MODO DE INTRA-PREDICCIÓN
115
FILTRO
IMAGEN ORIGINAL
IMAGEN DE REFERENCIA
116
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE IMAGEN DE REFERENCIA

FIG. 9

ORDEN DE BARRIDO DE BLOQUE 4X4

0	1	5	6
2	4	7	12
3	8	11	13
9	10	14	15

FIG. 10

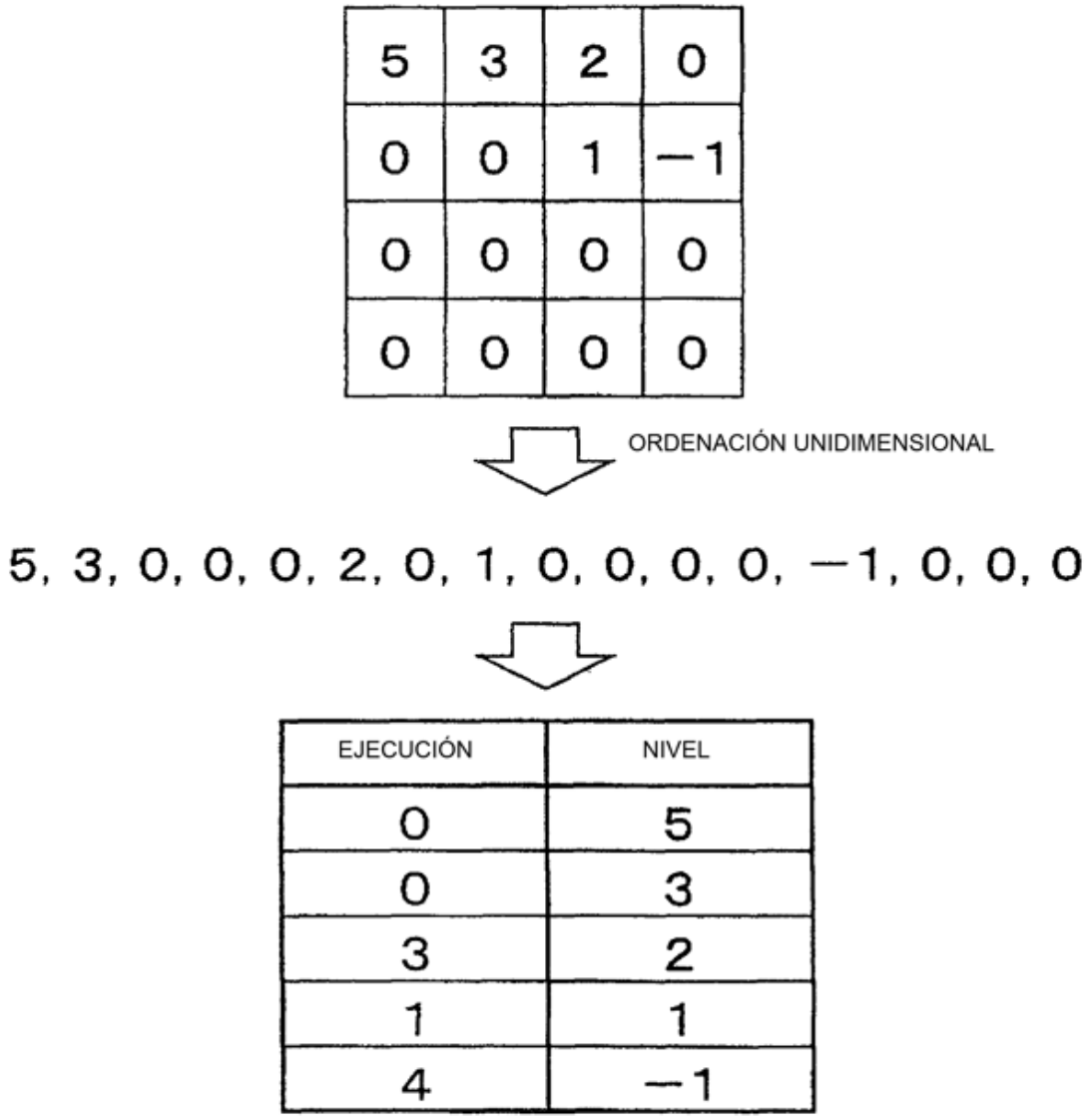

5 3 2 0
0 0 1 −1
0 0 0 0
0 0 0 0
ORDENACIÓN UNIDIMENSIONAL
5, 3, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, −1, 0, 0, 0
EJECUCIÓN
NIVEL
0 5
0 3
3 2
1 1
4 −1

FIG. 11

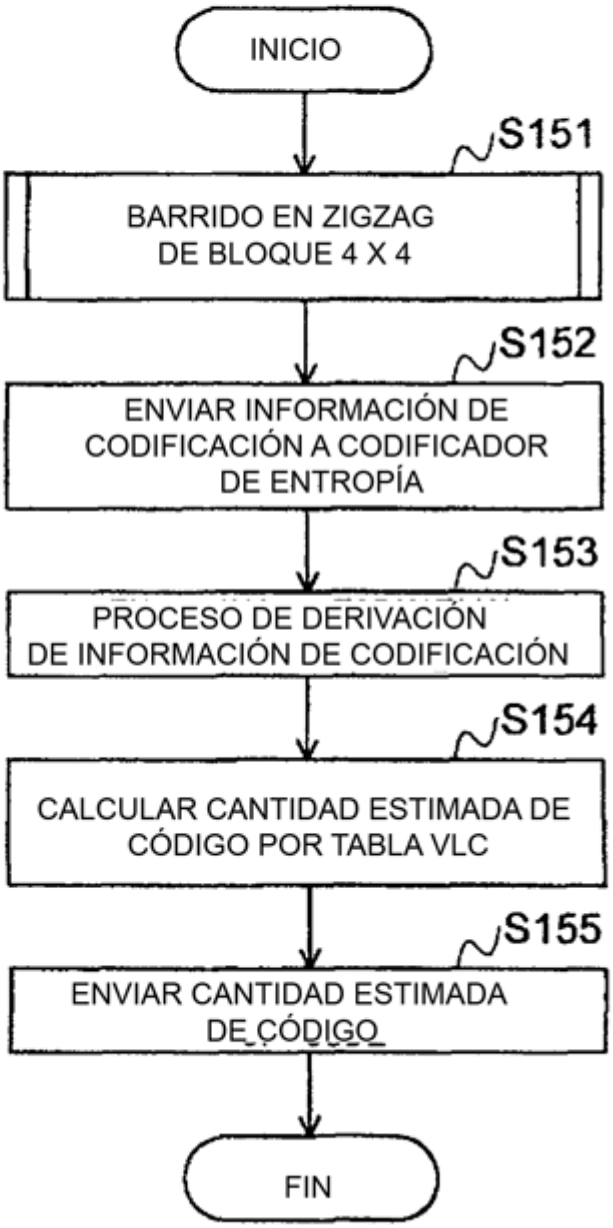

INICIO
S151
BARRIDO EN ZIGZAG DE BLOQUE 4 X 4
S152
ENVIAR INFORMACIÓN DE CODIFICACIÓN A CODIFICADOR DE ENTROPÍA
S153
PROCESO DE DERIVACIÓN DE INFORMACIÓN DE CODIFICACIÓN
S154
CALCULAR CANTIDAD ESTIMADA DE CÓDIGO POR TABLA VLC
S155
ENVIAR CANTIDAD ESTIMADA DE CÓDIGO
FIN

FIG. 12

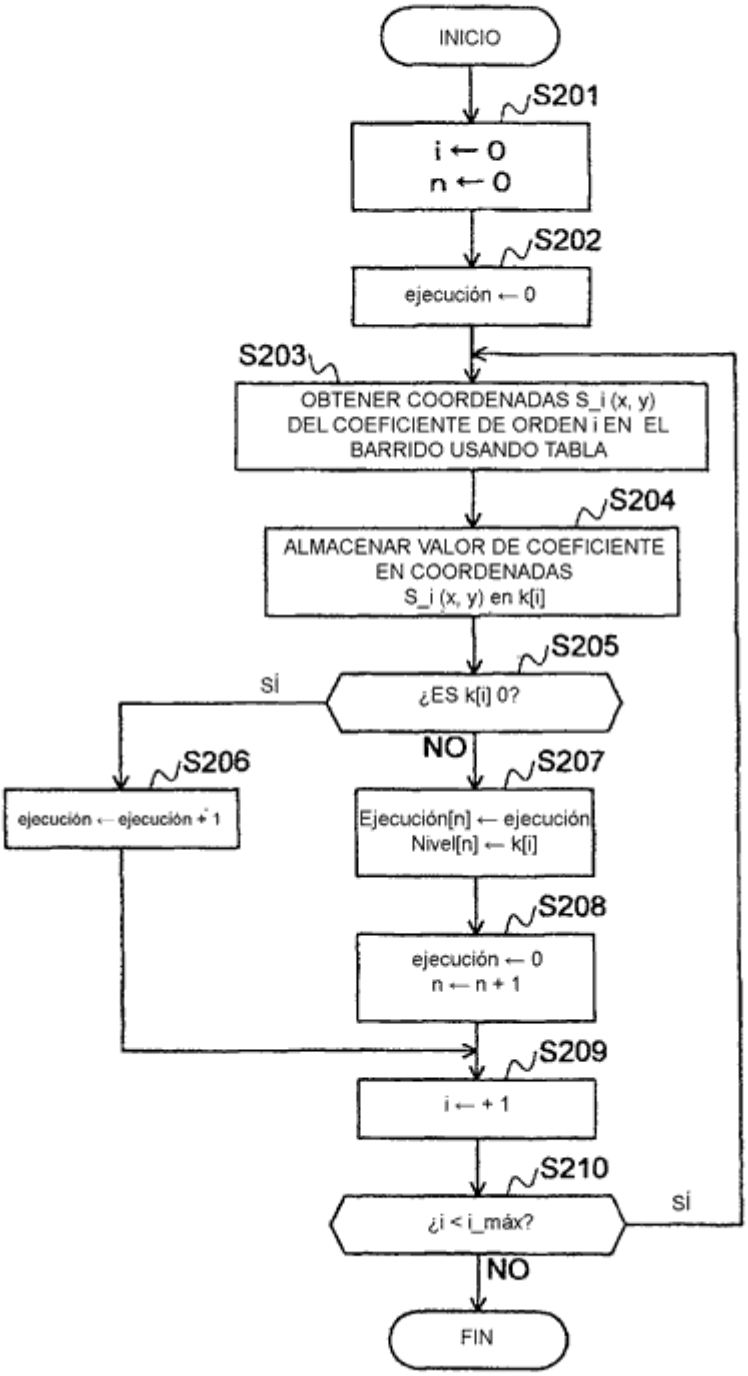

INICIO
S201
i ← 0
n ← 0
S202
ejecución ← 0
S203
OBTENER COORDENADAS S_i (x, y) DEL COEFICIENTE DE ORDEN i EN EL BARRIDO USANDO TABLA
S204
ALMACENAR VALOR DE COEFICIENTE EN COORDENADAS S_i (x, y) en k[i]
S205
¿ES k[i] 0?
SÍ
NO
S206
ejecución ← ejecución + 1
S207
Ejecución[n] ← ejecución
Nivel[n] ← k[i]
S208
ejecución ← 0
n ← n + 1
S209
i ← + 1
S210
¿i < i_máx?
SÍ
NO
FIN

FIG. 13

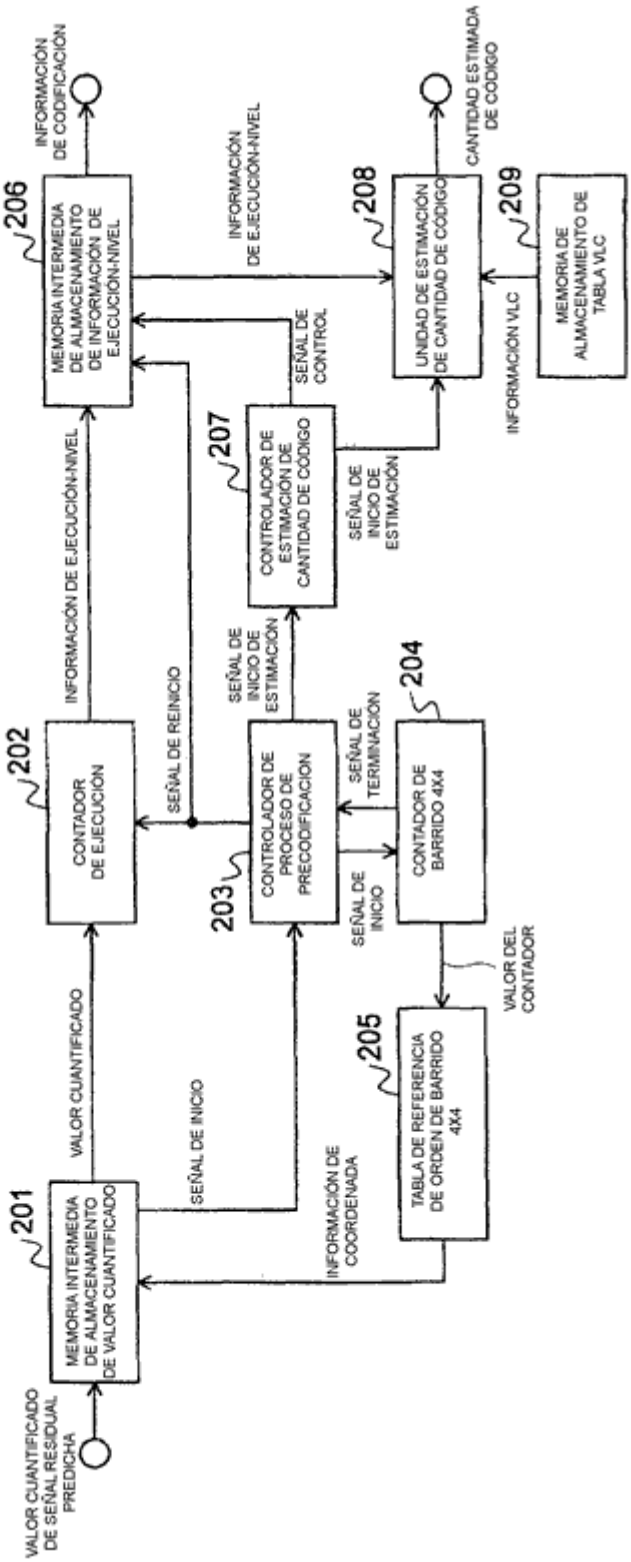

VALOR CUANTIFICADO DE SEÑAL RESIDUAL PREDICHA
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE VALOR CUANTIFICADO
201
VALOR CUANTIFICADO
SEÑAL DE INICIO
INFORMACIÓN DE COORDENADA
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 4X4
205
VALOR DEL CONTADOR
CONTADOR DE BARRIDO 4X4
204
SEÑAL DE INICIO
SEÑAL DE TERMINACIÓN
CONTROLADOR DE PROCESO DE PRECODIFICACIÓN
203
CONTADOR DE EJECUCIÓN
202
SEÑAL DE REINICIO
INFORMACIÓN DE EJECUCIÓN-NIVEL
SEÑAL DE INICIO DE ESTIMACIÓN
CONTROLADOR DE ESTIMACIÓN DE CANTIDAD DE CÓDIGO
207
SEÑAL DE CONTROL
SEÑAL DE INICIO DE ESTIMACIÓN
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE INFORMACIÓN DE EJECUCIÓN-NIVEL
206
INFORMACIÓN DE CODIFICACIÓN
INFORMACIÓN DE EJECUCIÓN-NIVEL
UNIDAD DE ESTIMACIÓN DE CANTIDAD DE CÓDIGO
208
CANTIDAD ESTIMADA DE CÓDIGO
INFORMACIÓN VLC
MEMORIA DE ALMACENAMIENTO DE TABLA VLC
209

FIG. 14

ORDEN DE BARRIDO DE CABAC

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

FIG. 15

ORDEN DE BARRIDO DE CAVLC

A0	B0	B1	C1	C3	D3	D6	A7
CO	A1	D1	B3	A4	C6	B7	C10
D0	A2	A3	B4	B6	C7	B10	D10
B2	D2	C4	A6	D7	A10	A11	B13
C2	D4	D5	A8	D9	B11	A13	C13
A5	C5	B8	C9	C11	D12	D13	A15
B5	C8	B9	D11	C12	A14	D14	B15
D8	A9	A12	B12	B14	C14	C15	D15

FIG. 16

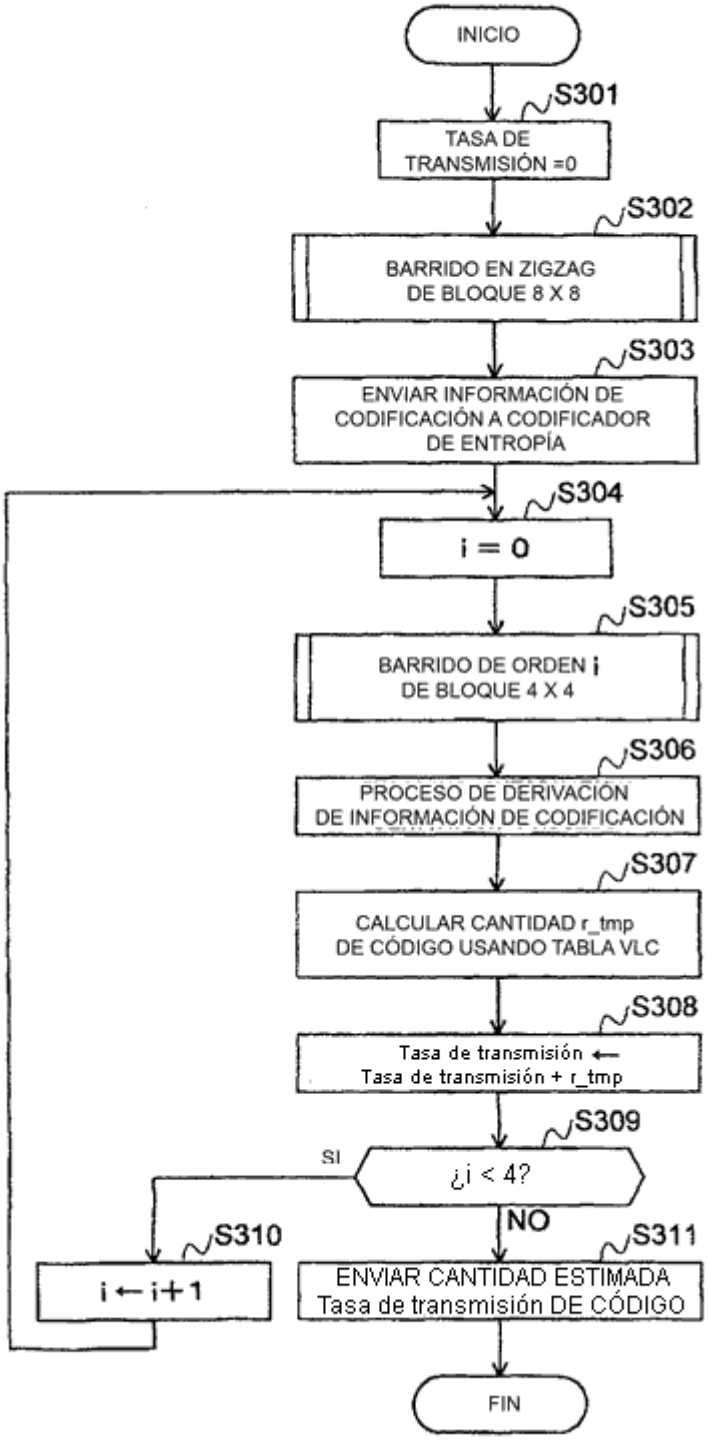
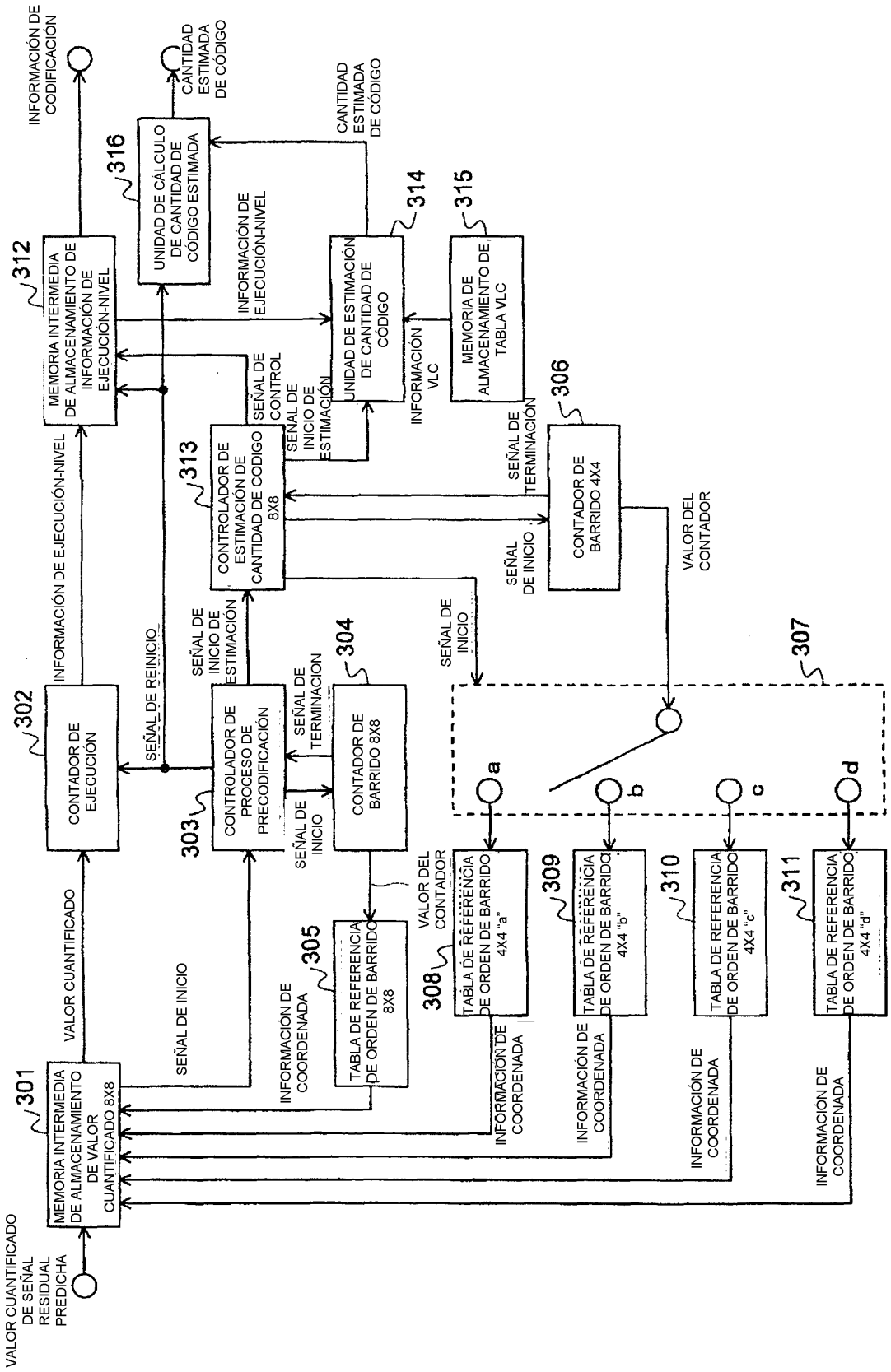

INICIO
S301
TASA DE TRANSMISIÓN =0
S302
BARRIDO EN ZIGZAG DE BLOQUE 8 X 8
S303
ENVIAR INFORMACIÓN DE CODIFICACIÓN A CODIFICADOR DE ENTROPÍA
S304
i = 0
S305
BARRIDO DE ORDEN i DE BLOQUE 4 X 4
S306
PROCESO DE DERIVACIÓN DE INFORMACIÓN DE CODIFICACIÓN
S307
CALCULAR CANTIDAD r_tmp DE CÓDIGO USANDO TABLA VLC
S308
Tasa de transmisión ← Tasa de transmisión + r_tmp
S309
SI
¿i < 4?
NO
S310
i ← i+1
S311
ENVIAR CANTIDAD ESTIMADA Tasa de transmisión DE CÓDIGO
FIN

FIG. 17


VALOR CUANTIFICADO DE SEÑAL RESIDUAL PREDICHA
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE VALOR CUANTIFICADO 8X8
301
VALOR CUANTIFICADO
SEÑAL DE INICIO
CONTADOR DE EJECUCIÓN
302
INFORMACIÓN DE EJECUCIÓN-NIVEL
SEÑAL DE REINICIO
CONTROLADOR DE PROCESO DE PRECODIFICACIÓN
303
SEÑAL DE INICIO
SEÑAL DE TERMINACION
CONTADOR DE BARRIDO 8X8
304
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 8X8
305
INFORMACIÓN DE COORDENADA
VALOR DEL CONTADOR
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 4X4 "a"
308
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 4X4 "b"
309
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 4X4 "c"
310
TABLA DE REFERENCIA DE ORDEN DE BARRIDO 4X4 "d"
311
INFORMACIÓN DE CODIFICACIÓN
MEMORIA INTERMEDIA DE ALMACENAMIENTO DE INFORMACIÓN DE EJECUCIÓN-NIVEL
312
SEÑAL DE INICIO DE ESTIMACIÓN
CONTROLADOR DE ESTIMACIÓN DE CANTIDAD DE CODIGO 8X8
313
SEÑAL DE CONTROL
SEÑAL DE INICIO DE ESTIMACIÓN
UNIDAD DE ESTIMACIÓN DE CANTIDAD DE CODIGO
314
INFORMACIÓN VLC
MEMORIA DE ALMACENAMIENTO DE TABLA VLC
315
UNIDAD DE CÁLCULO DE CANTIDAD DE CÓDIGO ESTIMADA
316
CANTIDAD ESTIMADA DE CÓDIGO
INFORMACIÓN DE EJECUCIÓN-NIVEL
CANTIDAD ESTIMADA DE CÓDIGO
SEÑAL DE TERMINACIÓN
CONTADOR DE BARRIDO 4X4
306
SEÑAL DE INICIO
VALOR DEL CONTADOR
SEÑAL DE INICIO
307
a
b
c
d
INFORMACIÓN DE COORDENADA
INFORMACIÓN DE COORDENADA
INFORMACIÓN DE COORDENADA
INFORMACIÓN DE COORDENADA