(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 930 257**

(51) Int. CI.:

H04N 19/11 (2014.01)
H04N 19/13 (2014.01)
H04N 19/176 (2014.01)
H04N 19/46 (2014.01)
H04N 19/91 (2014.01)
H04N 19/70 (2014.01)
H04N 19/96 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **18.01.2013 E 20178124 (2)**

(97) Fecha y número de publicación de la concesión europea: **12.10.2022 EP 3737094**

(54) Título: **Procedimiento, aparato y programa para codificar datos de imagen en un flujo de bits**

(30) Prioridad:

20.01.2012 AU 2012200345

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.12.2022

(73) Titular/es:

CANON KABUSHIKI KAISHA (100.0%)
30-2 Shimomaruko 3-chome, Ohta-ku
Tokyo 146-8501, JP

(72) Inventor/es:

ROSEWARNE, CHRISTOPHER JAMES

(74) Agente/Representante:

DURAN-CORRETJER, S.L.P

DESCRIPCIÓN

Procedimiento, aparato y programa para codificar datos de imagen en un flujo de bits

SECTOR TÉCNICO

La presente invención se refiere, en general, al procesamiento de señales de video digitales y, en concreto, a un procedimiento, aparato y sistema para codificar y descodificar los coeficientes residuales de una unidad de transformación (TU, Transform Unit), en los que la unidad de transformación (TU) puede tener una forma cuadrada o una forma no cuadrada.

ANTECEDENTES

Actualmente existen muchas aplicaciones para la codificación de video, que incluyen aplicaciones para la transmisión y el almacenamiento de datos de video. También se han desarrollado muchos estándares de codificación de video, y otros están actualmente en desarrollo. Los recientes desarrollos en la estandarización de la codificación de video han llevado a la formación de un grupo llamado “Equipo de colaboración conjunta sobre codificación de video” (JCT-VC, Joint Collaborative Team on Video Coding). El Equipo de colaboración conjunta sobre codificación de video (JCT-VC) incluye miembros del Grupo de estudio 16, Cuestión 6 (SG16/Q6, Study Group 16, Question 6) del Sector de estandarización de las Telecomunicaciones (ITU-T) de la Unión internacional de las Telecomunicaciones (ITU, International Telecommunications Union), conocido como el Grupo de Expertos en Codificación de video (VCEG, Video Coding Experts Group), y miembros del Grupo de trabajo 11, Subcomité 29, Comité técnico conjunto 1 de las Organizaciones internacionales para la estandarización/Comisión electrotécnica internacional (ISO/IEC JTC1/SC29/WG11, International Organisations for Standardisation/International Electrotechnical Commission Joint Technical Committee 1/Subcommittee 29/Working Group 11), también conocido como el Grupo de expertos en imágenes en movimiento (MPEG, Moving Picture Experts Group).

El equipo de colaboración conjunta en codificación de video (JCT-VC) tiene el objetivo de generar un nuevo estándar de codificación de video para mejorar significativamente un estándar de codificación de video actualmente existente, conocido como “H.264/MPEG-4 AVC”. El estándar H.264/MPEG-4 AVC es en sí mismo una gran mejora con respecto a los estándares de codificación de video anteriores, tales como el MPEG-4 y el H.263 de la ITU-T. El nuevo estándar de codificación de video en desarrollo ha sido denominado “codificación de video de alta eficiencia (HEVC, High Efficiency Video Coding)”. El equipo de colaboración conjunta sobre codificación de video, JCT-VC, también está considerando las dificultades de implementación que surgen de la tecnología propuesta para la codificación de video de alta eficiencia (HEVC), que crea dificultades al escalar implementaciones de la norma para funcionar a altas resoluciones o altas velocidades de fotogramas.

Un área del estándar de codificación de video H.264/MPEG-4 AVC que presenta dificultades para lograr una alta eficiencia de compresión es la codificación de los coeficientes residuales utilizados para representar los datos de video. Los datos de video están formados por una secuencia de fotogramas, y cada fotograma tiene una matriz bidimensional de muestras. Habitualmente, los fotogramas incluyen un canal de luminancia y dos canales de crominancia. Cada fotograma se descompone en uno o varios fragmentos. Cada fragmento contiene una o varias unidades de codificación más grandes (LCU, Largest Coding Units). Las unidades de codificación más grandes (LCU) tienen un tamaño fijo, siendo las dimensiones entre bordes una potencia de dos, y tienen la misma anchura y altura, tal como 64 muestras de luma. Una característica del estándar de codificación de video de alta eficiencia (HEVC) en desarrollo es “fragmentos de granularidad fina”. Cuando la función de fragmentos de granularidad fina está habilitada, los límites de los fragmentos no están limitados a los límites de la unidad de codificación más grande (LCU). Los fragmentos de granularidad fina pueden estar habilitados al nivel del flujo de bits.

Un árbol de codificación permite la subdivisión de cada unidad de codificación más grande (LCU) en cuatro zonas del mismo tamaño, teniendo cada una la mitad de la anchura y la altura de la unidad de codificación más grande (LCU) padre. Cada una de las zonas puede ser subdividida en cuatro zonas del mismo tamaño. Cuando una zona no se subdivide más, existe una unidad de codificación, que ocupa la totalidad de la zona. Dicho proceso de subdivisión se puede aplicar de manera recursiva hasta que el tamaño de una zona sea la unidad de codificación más pequeña (SCU, Smallest Coding Unit), y se deduzca una unidad de codificación (CU, Codification Unit) del tamaño de la unidad de codificación más pequeña (SCU). La subdivisión recursiva de una unidad de codificación más grande en una jerarquía de unidades de codificación tiene una estructura de árbol cuaternario y se conoce como árbol de codificación. Las unidades de codificación (CU) o zonas tienen una propiedad conocida como su “profundidad”, que hace referencia a su posición en el árbol de codificación en términos del nivel en la jerarquía de subdivisiones. Este proceso de subdivisión se codifica en el flujo de bits como una secuencia de indicadores codificados aritméticamente. Cuando se habilitan fragmentos de granularidad fina, se especifica un umbral que determina el tamaño más pequeño de la unidad de codificación en la que puede existir un límite de fragmento.

Existen un conjunto de unidades de codificación en el árbol de codificación que no se subdividen más, siendo esas unidades de codificación las que ocupan los nodos hoja del árbol de codificación. Existen árboles de transformación en estas unidades de codificación. Un árbol de transformación puede descomponer aún más una unidad de codificación

utilizando una estructura de árbol cuaternario tal como se utiliza para el árbol de codificación. En los nodos hoja del árbol de transformación, los datos residuales se codifican utilizando unidades de transformación (TU). A diferencia del árbol de codificación, el árbol de transformación puede subdividir las unidades de codificación en unidades de transformación que tienen una forma no cuadrada. Además, la estructura del árbol de transformación no requiere que las unidades de transformación (TU) ocupen toda el área proporcionada por la unidad de codificación padre.

Cada unidad de codificación en los nodos hoja de los árboles de codificación está subdividida en una o varias matrices de muestras de datos predichas, cada una conocida como unidad de predicción (PU, Prediction Unit). Cada unidad de predicción (PU) contiene una predicción de una porción de los datos del fotograma de entrada, obtenidos a partir de la aplicación de un proceso de intrapredicción o un proceso de interpredicción. Se pueden utilizar varios procedimientos para codificar unidades de predicción (PU) dentro de una unidad de codificación (CU). Una sola unidad de predicción (PU) puede ocupar un área completa de la unidad de codificación (CU), o la unidad de codificación (CU) puede estar dividida en dos unidades de predicción (PU) rectangulares del mismo tamaño, ya sea horizontal o verticalmente. Además, las unidades de codificación (CU) pueden estar divididas en cuatro unidades de predicción (PU) cuadradas del mismo tamaño.

Un codificador de video comprime los datos de video en un flujo de bits mediante la conversión de los datos de video en una secuencia de elementos de sintaxis. Un esquema de codificación aritmética binaria adaptable al contexto (CABAC, Context Adaptive Binary Arithmetic Coding) está definido en el estándar de codificación de video de alta eficiencia (HEVC), en desarrollo, utilizando un esquema de codificación aritmética idéntico al definido en el estándar de compresión de video MPEG4-AVC/H.264. En el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo, cuando se utiliza la codificación aritmética binaria adaptable al contexto (CABAC), cada elemento de sintaxis se expresa como una secuencia de bins, en la que los bins se seleccionan de un conjunto de bins disponibles. El conjunto de bins disponibles se obtiene de un modelo de contexto, con un contexto por bin. Cada contexto contiene un valor probable de bin (el 'valMPS') y un estado de probabilidad para la operación de codificación aritmética o la operación de descodificación aritmética. Se debe tener en cuenta que los bins pueden ser codificados asimismo por derivación, donde no hay asociación con un contexto. Los bins codificados por derivación consumen un bit en el flujo de bits y, por lo tanto, son adecuados para los bins con la misma probabilidad de ser de un valor uno o de un valor cero. La creación de dicha secuencia de bins a partir de un elemento de sintaxis se conoce como "binarizar" los elementos de sintaxis.

En un codificador de video o descodificador de video, puesto que existe información de contexto independiente para cada bin, la selección de contexto para los bins proporciona un medio para mejorar la eficiencia de la codificación. En concreto, la eficiencia de la codificación se puede mejorar seleccionando un bin particular, de tal manera que las propiedades estadísticas de los casos anteriores del bin, donde se utilizó la información de contexto asociada, se correlacionen con las propiedades estadísticas de un caso actual del bin. Dicha selección de contexto utiliza con frecuencia información espacialmente local para determinar el contexto óptimo.

En el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo y en el estándar H.264/MPEG-4 AVC, se obtiene una predicción para un bloque actual, en base a datos de muestra de referencia de otros fotogramas o de zonas vecinas dentro del bloque actual que han sido descodificadas previamente. La diferencia entre la predicción y los datos de muestra deseados se conoce como residuo. Una representación del residuo en el dominio de la frecuencia es una matriz bidimensional de coeficientes residuales. Por convención, la esquina superior izquierda de la matriz bidimensional contiene coeficientes residuales que representan información de baja frecuencia.

Un aspecto del rendimiento del estándar de codificación de video de alta eficiencia (HEVC) en desarrollo hace referencia a la capacidad de codificar o descodificar datos de video a altas velocidades de transferencia de bits. El esquema de codificación aritmética binaria adaptable al contexto (CABAC) empleado en el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo es compatible con un modo de operación de "igual probabilidad" denominado "codificación por derivación". En este modo, el bin no está asociado con un contexto del modelo de contexto y, por lo tanto, no hay ninguna etapa de actualización del modelo de contexto. En este modo, es posible leer múltiples bins adyacentes del flujo de bits en paralelo, siempre que cada bin esté codificado por derivación, lo que aumenta el rendimiento. Por ejemplo, las implementaciones de hardware pueden escribir/leer grupos de datos codificados por derivación adyacentes en paralelo para aumentar el rendimiento de la codificación/descodificación del flujo de bits.

Un documento de V. Seregin y otros, "Utilización del modo CABAC de probabilidad igual para la codificación de modos intra" ("Utilisation of CABAC equal probability mode for intra modes coding"), 6ª reunión del JCT-VC, reunión del MPEG, 14-7-2011 al 22-7-2011, Turín, Equipo de colaboración conjunta sobre codificación de video (JCT-VC) del ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, el documento XP030009399, da a conocer la utilización del módulo de codificación CABAC de igual probabilidad (derivación) además de la modelización por contexto actual para la intracodificación de luma y croma. En el procedimiento propuesto, la mayoría de los bins de modo intra se codifican con modo de derivación, sin modelización de contexto, con la excepción del primer bin de modo intra de croma, y el modo intra de luma para las unidades de predicción 8x8, 16x16 y 32x32.

Un documento de C. Yeo y otros, "No CE6: sobre codificación en modo de intrapredicción" ("Non-CE6: On

intra-prediction mode coding"), 7ª reunión de JCT-VC, 21-11-2011 al 30-11-2011, Ginebra, Equipo de colaboración conjunta sobre codificación de video (JCT-VC) de ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, el documento XP030110137, da a conocer una propuesta de dos modificaciones a la codificación actual del modo de intrapredicción en HM. La primera modificación propuesta es eliminar el contexto para codificar el índice de modo más probable cuando CABAC es el codificador por entropía. La segunda es una modificación de la codificación de modo restante.

Un documento de K. Misra y otros, "Utilización del modo CABAC de derivación para codificación del modo de intrapredicción" ("Using CABAC bypass mode for coding intra-prediction mode"), 7ª reunión del JCT-VC, 21-11-2011 al 30-11-2011, Ginebra, Equipo de colaboración conjunto sobre codificación de video (JCT-VC) del ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, el documento XP030110691, da a conocer una propuesta para la utilización del modo CABAC de derivación para codificar los elementos de sintaxis del modo de intrapredicción mpm_idx y rem_intra_luma_pred_mode.

Un documento de H. Sasai y otros, "Codificación de probabilidad fija para modo Intra" ("Fixed probability coding for Intra mode"), 6ª reunión del JCT-VC, 14-7-2011 al 22-7-2011, Turín, Equipo de colaboración conjunta sobre codificación de video (JCT-VC) de ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, el documento XP030009449, da a conocer una técnica para la reducción de la complejidad para el proceso de análisis de los parámetros de modo intra.

Un documento de B. Bross y otros, "Borrador de trabajo 5 de la especificación textual de codificación de video de alta eficiencia (HEVC)" ("High Efficiency Video Coding (HEVC) text specification Working Draft 5"), 7ª reunión del JCT-VC, 21-11-2011 al 30-11-2011, Ginebra, Equipo de colaboración conjunta sobre Codificación de video (JCT-VC) de ITU-T SG16 WP3 e ISO/IEC JTC1/SC29/WG11, el documento XP030111032, da a conocer el borrador de trabajo 5 de la codificación de video de alta eficiencia.

Un documento de V. Sze y otros, "Procesamiento de contexto en paralelo al nivel de coeficientes" ("Parallel Context Processing of Coefficient Level"), 6ª reunión del JCT-VC, 14-7-2011 al 22-7-2011, Turín, Equipo de colaboración conjunta sobre Codificación de video (JCT-VC) de ISO/IEC JTC1/SC29/WG11 e ITU-T SG16, XP030009153, da a conocer el agrupamiento de bins de derivación en elementos de sintaxis al nivel de coeficientes.

Un documento de H. Sasai y otros "Codificación MVD modificada para CABAC" ("Modified MVD coding for CABAC"), 97ª reunión del MPEG; 18-7-2011 al 22-7-2011, Turín, Grupo de expertos de imágenes en movimiento de ISO/IEC JTC1/SC29/WG11, XP030049414, da a conocer una técnica para la reducción de la complejidad en el proceso de análisis sintáctico del parámetro diferencia en el vector de movimiento.

Un documento de E. Francois y otros, "CE6b: Codificación intramodo con 4 MPM y clasificación de modos" ("CE6b: Intra mode coding with 4 MPMs and mode ranking"), 98ª reunión del MPEG; 28-11-2011 al 2-12-2011, Ginebra, Grupo de expertos de imágenes en movimiento de ISO/IEC JTC1/SC29/WG11), número m21801, 21 de noviembre de 2011, XP0300500364, describe una solución de codificación intramodo que utiliza 4 MPM (2 MPM normales y 2 MPRM) y clasificación de modos para codificar los bins de modos restantes.

Un documento de D. Marpe y otros, "Codificación aritmética binaria adaptativa basada en contexto en el estándar de compresión de video H.264/AVC" ("Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard"), IEEE Transacciones en circuitos y sistemas para tecnología de video, vol. 13, número 7, 1 de julio de 2003, páginas 620-636, XP055120073, describe la combinación de una técnica de codificación aritmética binaria adaptativa con modelización de contexto para conseguir una adaptación de alto grado y reducción de la redundancia.

CARACTERÍSTICAS

Un objetivo de la presente invención es superar o, por lo menos, mejorar, sustancialmente uno o varios inconvenientes de las disposiciones existentes. Según un aspecto de la presente invención, se da a conocer un procedimiento de codificación según cualquiera de las reivindicaciones 1 a 6 adjuntas. Según otro aspecto de la presente invención, existe un aparato para codificar según la reivindicación 7. Según otro aspecto de la presente invención, se da a conocer un programa según la reivindicación 8. Según otro aspecto de la presente invención, se da a conocer un procedimiento de descodificación según cualquiera de las reivindicaciones 9 a 14. En otro aspecto de la presente invención, se da a conocer un aparato según la reivindicación 15. En otro aspecto más de la presente invención, se da a conocer un programa según la reivindicación 16.

También se dan a conocer otros aspectos.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Por lo menos una realización de la presente invención se describirá a continuación haciendo referencia a los siguientes dibujos, en los que:

la figura 1 es un diagrama de bloques esquemático que muestra módulos funcionales de un codificador de video;

la figura 2 es un diagrama de bloques esquemático que muestra módulos funcionales de un descodificador de video;

las figuras 3A y 3B forman un diagrama de bloques esquemático de un sistema informático de propósito general sobre el que el codificador y el descodificador de las figuras 1 y 2, respectivamente, se pueden poner en práctica;

la figura 4 es un diagrama de bloques esquemático que muestra módulos funcionales de un codificador por entropía;

la figura 5 es un diagrama de bloques esquemático que muestra módulos funcionales de un descodificador por entropía;

la figura 6 es un diagrama de bloques esquemático que muestra una unidad de codificación más grande (LCU) a modo de ejemplo;

la figura 7 es un diagrama de bloques esquemático que muestra un flujo de bits convencional que representa la unidad de codificación más grande (LCU) a modo de ejemplo;

la figura 8 es un diagrama de bloques esquemático que muestra un flujo de bits según la presente invención que representa la unidad de codificación más grande (LCU) a modo de ejemplo;

la figura 9 es un diagrama de flujo esquemático que muestra un procedimiento según la presente invención para descodificar los elementos de sintaxis de una unidad de codificación más grande (LCU) de un flujo de bits similar a la de la figura 8;

la figura 10 es un diagrama de flujo esquemático que muestra un procedimiento resumido según la presente invención para descodificar los elementos de sintaxis de una unidad de codificación más grande (LCU) de un flujo de bits;

la figura 11 es un diagrama de flujo esquemático que muestra un procedimiento según la presente invención para codificar los elementos de sintaxis de una unidad de codificación más grande (LCU);

la figura 12 es un diagrama de flujo esquemático que muestra un procedimiento resumido según la presente invención para codificar los elementos de sintaxis de una unidad de codificación más grande (LCU) de un flujo de bits;

la figura 13 es un diagrama de bloques esquemático que muestra un flujo de bits que representa la unidad de codificación más grande (LCU) a modo de ejemplo generada por el procedimiento alternativo según la presente invención para su análisis sintáctico;

la figura 14 es un diagrama de flujo esquemático que muestra un procedimiento alternativo según la presente invención para descodificar los elementos de sintaxis de una unidad de codificación más grande (LCU);

la figura 15 es un diagrama de bloques esquemático que muestra una unidad de codificación más grande (LCU) a modo de ejemplo con un límite de fragmento dentro de la unidad de codificación más grande (LCU);

la figura 16 es un diagrama de bloques esquemático que muestra un flujo de bits que representa la unidad de codificación más grande (LCU) a modo de ejemplo generada mediante el procedimiento para análisis sintáctico con fragmentos de granularidad fina (FGS, Fine Granularity Slices) habilitado; y

la figura 17 es un diagrama de flujo esquemático que muestra otro procedimiento alternativo según la presente invención para descodificar los elementos de sintaxis de una unidad de codificación más grande (LCU).

DESCRIPCIÓN DETALLADA INCLUYENDO EL MEJOR MODO

Cuando se hace referencia en uno o varios de los dibujos adjuntos a etapas y/o características que tienen los mismos números de referencia, esas etapas y/o características tienen, para los propósitos de esta descripción, la misma función o funciones u operación u operaciones, a menos que aparezca la intención contraria.

La figura 1 es un diagrama de bloques esquemático que muestra módulos funcionales de un codificador de video 100. La figura 2 es un diagrama de bloques esquemático que muestra los módulos funcionales de un descodificador de video 200 correspondiente. El codificador de video 100 y el descodificador de video 200 pueden ser implementados utilizando un sistema informático de propósito general 300, tal como el mostrado en las figuras 3A y 3B, en el que los diversos módulos funcionales pueden ser implementados mediante hardware exclusivo dentro del sistema informático 300, mediante software ejecutable en el sistema informático 300 o, de manera alternativa, mediante una combinación de hardware y software exclusivo, ejecutables en el sistema informático 300.

Tal como se ve en la figura 3A, el sistema informático 300 incluye: un módulo de ordenador 301; dispositivos de entrada tales como un teclado 302, un dispositivo 303 para señalar del tipo de ratón, un escáner 326, una cámara

327 y un micrófono 380; y dispositivos de salida que incluyen una impresora 315, un dispositivo de visualización 314 y altavoces 317. Un dispositivo transceptor modulador-demodulador externo (módem) 316 puede ser utilizado por el módulo de ordenador 301 para comunicarse hacia y desde una red de comunicaciones 320 a través de una conexión 321. La red de comunicaciones 320 puede ser una red de área extensa (WAN, Wide Area Network), tal como Internet, una red de telecomunicaciones celulares, o una WAN privada. Cuando la conexión 321 es una línea telefónica, el módem 316 puede ser un módem tradicional de "marcación". Alternativamente, cuando la conexión 321 es una conexión de alta capacidad (por ejemplo, un cable), el módem 316 puede ser un módem de banda ancha. Asimismo, se puede utilizar un módem inalámbrico para la conexión inalámbrica a la red de comunicaciones 320.

El módulo de ordenador 301 incluye habitualmente, por lo menos, una unidad de procesador 305, y una unidad de memoria 306. Por ejemplo, la unidad de memoria 306 puede tener una memoria de acceso aleatorio (RAM, Random Access Memory) de semiconductores y una memoria de solo lectura (ROM, Read Only Memory) de semiconductores. El módulo de ordenador 301 incluye asimismo varias interfaces de entrada/salida (I/O, Input/Output) que incluyen: una interfaz de audio y video 307 que se acopla a la pantalla de video 314, altavoces 317 y un micrófono 380; una interfaz I/O 313 que se acopla al teclado 302, un ratón 303, un escáner 326, una cámara 327 y, opcionalmente, un mando de palanca u otro dispositivo de interfaz humana (no ilustrado); y una interfaz 308 para el módem externo 316 y la impresora 315. En algunas implementaciones, el módem 316 puede estar incorporado dentro del módulo de ordenador 301, por ejemplo, dentro de la interfaz 308. El módulo de ordenador 301 tiene asimismo una interfaz de red local 311, que permite el acoplamiento del sistema informático 300 a través de una conexión 323 a una red de comunicaciones de área local 322, conocida como Red de área local (LAN, Local Area Network). Tal como se ilustra en la figura 3A, la red de comunicaciones local 322 puede ser acoplada asimismo a la red extensa 320 a través de una conexión 324, que, habitualmente, incluiría un dispositivo llamado "cortafuegos" o un dispositivo de funcionalidad similar. La interfaz 311 de red local puede comprender una tarjeta de circuito Ethernet™, una disposición inalámbrica Bluetooth™ o una disposición inalámbrica IEEE 802.11; sin embargo, se pueden poner en práctica muchos otros tipos de interfaces para la interfaz 311.

Las interfaces I/O 308 y 313 pueden proporcionar conectividad en serie y en paralelo, o ambas, la primera, habitualmente se está implementando según los estándares del bus de serie universal (USB, Universal Serial Bus) y tiene conectores USB correspondientes (no ilustrados). Se proporcionan dispositivos de almacenamiento 309, e incluyen habitualmente una unidad de disco duro (HDD, Hard Disk Drive) 310. Asimismo, se pueden utilizar otros dispositivos de almacenamiento, tales como una unidad de disco flexible y una unidad de cinta magnética (no ilustrados). Una unidad de disco óptica 312 está proporcionada habitualmente para actuar como una fuente de datos no volátil. Los dispositivos de memoria portátiles, tales como discos ópticos (por ejemplo, CD-ROM, DVD, Blu-ray Disc™), USB-RAM, discos duros externos portátiles y discos flexibles, por ejemplo, se pueden utilizar como fuentes de datos apropiadas para el sistema 300. Habitualmente, cualquiera de las unidades HDD 310, la unidad óptica 312, las redes 320 y 322 o la cámara 327 pueden formar una fuente para codificar datos de video o, con la pantalla 314, un destino para que los datos de video descodificados sean almacenados o reproducidos.

Los componentes 305 a 313 del módulo de ordenador 301 se comunican habitualmente a través de un bus 304 interconectado y de una manera que resulta en un modo de operación convencional del sistema informático 300 conocido por los expertos en la técnica. Por ejemplo, el procesador 305 se acopla al bus del sistema 304 mediante una conexión 318. Del mismo modo, la memoria 306 y la unidad de disco óptico 312 se acoplan al bus del sistema 304 mediante conexiones 319. Ejemplos de ordenadores en los que se pueden poner en práctica las disposiciones descritas incluyen PC comercializados por la firma IBM y compatibles, Sparcstations comercializados por la firma Sun, Mac™ comercializados por la firma Apple o sistemas informáticos similares.

Cuando sea apropiado o deseado, el codificador 100 y el descodificador 200, así como los procedimientos descritos a continuación, pueden ser implementados utilizando el sistema informático 300 en el que el codificador 100, el descodificador 200 y los procesos de las figuras 10 y 11, que se describirán, pueden ser implementados como uno o varios programas de aplicación de software 333 ejecutables en el sistema informático 300. En concreto, el codificador 100, el descodificador 200 y las etapas de los procedimientos descritos son realizados mediante instrucciones 331 (ver figura 3B) en el software 333 que se llevan a cabo dentro del sistema informático 300. Las instrucciones del software 331 pueden estar formadas como uno o varios módulos de código, cada uno para realizar una o varias tareas. El software se puede dividir asimismo en dos partes independientes, en las que una primera parte y los módulos de código correspondientes realizan los procedimientos descritos, y una segunda parte y los módulos de código correspondientes administran una interfaz de usuario entre la primera parte y el usuario.

El software puede estar almacenado en un medio legible por ordenador, incluidos los dispositivos de almacenamiento que se describen a continuación, por ejemplo. El software es cargado en el sistema informático 300 desde el medio legible por ordenador, y, a continuación, ejecutado por el sistema informático 300. Un medio legible por ordenador que tiene dicho software o programa informático grabado en el medio legible por ordenador es un producto de programa informático. La utilización del producto de programa informático en el sistema informático 300 efectúa preferentemente un aparato ventajoso para implementar el codificador 100, el descodificador 200 y los procedimientos descritos.

El software 333 normalmente se almacena en el HDD 310 o la memoria 306. El software se carga en el sistema

informático 300 desde un medio legible por ordenador, y es ejecutado por el sistema informático 300. De este modo, por ejemplo, el software 333 puede ser almacenado en un medio de almacenamiento de disco legible de manera óptica (por ejemplo, un CD-ROM) 325 que se lee mediante la unidad de disco óptico 312.

En algunos casos, los programas de aplicación 333 pueden ser suministrados al usuario codificados en uno o varios CD-ROM 325 y ser leídos a través de la unidad correspondiente 312, o, alternativamente, pueden ser leídos por el usuario desde las redes 320 o 322. Aún más, el software también puede ser cargado en el sistema informático 300 desde otros medios legibles por ordenador. Los medios de almacenamiento legibles por ordenador se refieren a cualquier medio de almacenamiento tangible no transitorio que proporciona instrucciones y/o datos grabados al sistema informático 300 para su ejecución y/o procesamiento. Ejemplos de dichos medios de almacenamiento incluyen discos flexibles, una cinta magnética, un CD-ROM, DVD, un disco Blu-ray, una unidad de disco duro, una ROM o un circuito integrado, una memoria USB, un disco magnetoóptico o una tarjeta legible por ordenador, tal como una tarjeta PCMCIA y similares, tanto si dichos dispositivos son internos como externos al módulo de ordenador 301. Ejemplos de medios de transmisión legibles por ordenador transitorios o no tangibles que también pueden participar en la provisión del software, programas de aplicación, instrucciones y/o datos de video o datos de video codificados al módulo de ordenador 301 incluyen canales de transmisión por radio o infrarrojos, así como una conexión de red a otro ordenador o dispositivo en la red, e Internet o Intranets, incluidas las transmisiones de correo electrónico e información registrada en sitios web y similares.

La segunda parte de los programas de aplicación 333 y los módulos de código correspondientes mencionados anteriormente puede ser ejecutada para implementar una o varias interfaces gráficas de usuario (GUI, Graphical User Interfaces) que serán renderizadas o representadas de otra manera en la pantalla 314. Mediante la manipulación, habitualmente, del teclado 302 y el ratón 303, un usuario del sistema informático 300 y la aplicación puede manipular la interfaz de una manera funcionalmente adaptable para proporcionar comandos de control y/o entrada a las aplicaciones asociadas con la GUI o las GUI. Asimismo, se pueden implementar otras formas de interfaces de usuario funcionalmente adaptables, tales como una interfaz de audio que utiliza la salida de indicaciones de voz a través de los altavoces 317 y la entrada de comandos de voz del usuario a través del micrófono 380.

La figura 3B es un diagrama de bloques esquemático detallado del procesador 305 y una “memoria” 334. La memoria 334 representa una agregación lógica de todos los módulos de memoria (incluidos el HDD 309 y la memoria 306 de semiconductores) a los que puede acceder el módulo de ordenador 301 en la figura 3A.

Cuando el módulo de ordenador 301 es encendido inicialmente, se ejecuta un programa 350 de autocomprobación de encendido (POST, Power-On Self-Test). El programa POST 350 se almacena habitualmente en una ROM 349 de la memoria de semiconductores 306 de la figura 3A. Un dispositivo de hardware tal como el software de almacenamiento ROM 349 a veces se denomina firmware. El programa POST 350 examina el hardware dentro del módulo de ordenador 301 para garantizar un funcionamiento adecuado y, habitualmente, comprueba el procesador 305, la memoria 334 (309, 306) y un módulo de software básico de los sistemas de entrada-salida (BIOS, Basic Input-Output Software) 351, asimismo, habitualmente almacenado en la ROM 349, para comprobar su correcto funcionamiento. Una vez que el programa POST 350 ha sido ejecutado con éxito, el BIOS 351 activa la unidad de disco duro 310 de la figura 3A. La activación de la unidad de disco duro 310 hace que se ejecute un programa de cargador de rutina de arranque 352 que reside en la unidad de disco duro 310 a través del procesador 305. Esto carga un sistema operativo 353 en la memoria RAM 306, tras lo cual el sistema operativo 353 comienza a funcionar. El sistema operativo 353 es una aplicación al nivel del sistema, ejecutable por el procesador 305, para cumplir con varias funciones de alto nivel, incluida la gestión del procesador, la gestión de la memoria, la gestión de dispositivos, la gestión del almacenamiento, la interfaz de aplicación de software y la interfaz de usuario genérica.

El sistema operativo 353 gestiona la memoria 334 (309, 306) para garantizar que cada proceso o aplicación que se ejecuta en el módulo de ordenador 301 tenga suficiente memoria para ejecutar sin chocar con la memoria asignada a otro proceso. Además, los diferentes tipos de memoria disponibles en el sistema 300 de la figura 3A deben ser utilizados correctamente para que cada proceso pueda ser ejecutado de manera efectiva. Por consiguiente, la memoria 334 agregada no pretende ilustrar cómo se asignan fragmentos particulares de memoria (a menos que se indique lo contrario), sino más bien proporcionar una vista general de la memoria accesible por el sistema informático 300 y cómo se utiliza.

Tal como se muestra en la figura 3B, el procesador 305 incluye una serie de módulos funcionales que incluyen una unidad de control 339, una unidad lógica aritmética (ALU, Arithmetic Logic Unit) 340 y una memoria local o interna 348, a veces llamada memoria caché. La memoria caché 348 incluye habitualmente una serie de registros de almacenamiento 344 a 346 en una sección de registro. Uno o varios buses internos 341 interconectan funcionalmente estos módulos funcionales. El procesador 305 tiene asimismo habitualmente una o varias interfaces 342 para comunicarse con dispositivos externos a través del bus del sistema 304, utilizando una conexión 318. La memoria 334 está acoplada al bus 304 utilizando una conexión 319.

El programa de aplicación 333 incluye una secuencia de instrucciones 331 que puede incluir instrucciones condicionales de rama y bucle. El programa 333 puede incluir asimismo datos 332 que se utilizan en la ejecución del programa 333. Las instrucciones 331 y los datos 332 se almacenan en las ubicaciones de memoria 328, 329, 330 y

335, 336, 337, respectivamente. Dependiendo del tamaño relativo de las instrucciones 331 y las ubicaciones de memoria 328 a 330, una instrucción particular puede ser almacenada en una única ubicación de memoria, tal como se muestra en la instrucción mostrada en la ubicación de memoria 330. Alternativamente, una instrucción puede estar segmentada en una cantidad de partes, cada una de las cuales se almacena en una ubicación de memoria separada, tal como se muestra en los fragmentos de instrucciones que se muestran en las ubicaciones de memoria 328 y 329.

En general, al procesador 305 se le dan un conjunto de instrucciones que se ejecutan en el mismo. El procesador 305 espera una entrada posterior, a la que reacciona el procesador 305 ejecutando otro conjunto de instrucciones. Cada entrada puede ser proporcionada desde una o varias de diversas fuentes, incluidos los datos generados por uno o varios de los dispositivos de entrada 302, 303, datos recibidos de una fuente externa a través de una de las redes 320, 302, datos recuperados de uno de los dispositivos de almacenamiento 306, 309 o datos recuperados de un medio de almacenamiento 325 introducidos en el lector correspondiente 312, todos representados en la figura 3A. La ejecución de un conjunto de instrucciones puede resultar, en algunos casos, en la emisión de datos. La ejecución puede implicar asimismo almacenar datos o variables en la memoria 334.

El codificador 100, el descodificador 200 y los procedimientos descritos utilizan variables de entrada 354, que se almacenan en la memoria 334 en las ubicaciones de memoria correspondientes 355, 356, 357. El codificador 100, el descodificador 200 y los procedimientos descritos generan variables de salida 361, que se almacenan en la memoria 334 en las ubicaciones de memoria 362, 363, 364 correspondientes. Las variables temporales 358 pueden ser almacenadas en las ubicaciones de memoria 359, 360, 366 y 367.

Con referencia al procesador 305 de la figura 3B, los registros 344, 345, 346, la unidad lógica aritmética (ALU) 340 y la unidad de control 339 trabajan juntos para realizar secuencias de microoperaciones necesarias para realizar la “recuperación, descodificación y ejecución” de ciclos para cada instrucción en el conjunto de instrucciones que conforman el programa 333. Cada ciclo de recuperación, descodificación y ejecución comprende:

(a) una operación de recuperación, que recupera o lee una instrucción 331 desde una ubicación de memoria 328, 329, 330;
(b) una operación de descodificación, en la que la unidad de control 339 determina qué instrucción se ha recuperado; y
(c) una operación de ejecución, en la que la unidad de control 339 y/o la ALU 340 ejecutan la instrucción.

A partir de entonces, se puede ejecutar un ciclo adicional de recuperación, descodificación y ejecución para la siguiente instrucción. De manera similar, se puede realizar un ciclo de almacenamiento mediante el cual la unidad de control 339 almacena o escribe un valor en una ubicación de memoria 332.

Cada etapa o subproceso en los procesos de las figuras 1 a 17 que se describirá está asociado con uno o varios fragmentos del programa 333 y es realizado habitualmente por la sección de registro 344, 345, 347, la ALU 340 y la unidad de control 339 en el procesador 305 trabajando juntos para realizar la recuperación, descodificación y ejecución de ciclos para cada instrucción del conjunto de instrucciones para los fragmentos anotados del programa 333.

El codificador 100, el descodificador 200 y los procedimientos descritos pueden ser implementados alternativamente mediante hardware exclusivo, tal como uno o varios circuitos integrados que realizan las funciones o subfunciones de los procedimientos descritos. Dicho hardware exclusivo puede incluir procesadores gráficos, procesadores de señales digitales, circuitos integrados específicos para aplicaciones (ASIC, Application Specific Integrated Circuits), matrices de puertas programables in situ (FPGA, Field Programmable Gate Array) o uno o varios microprocesadores y memorias asociadas. El efecto neto de los sistemas descritos es un aparato computarizado configurado para procesar unidades de codificación asociadas con un flujo de bits de datos de video.

Tal como se describió anteriormente, el codificador de video 100 puede ser implementado como uno o varios módulos de código de software del programa de aplicación de software 333 que reside en la unidad de disco duro 305 y que está controlado en su ejecución por el procesador 305. En concreto, el codificador de video 100 comprende los módulos 102 a 112, 114 y 115, que pueden ser implementados cada uno como uno o varios módulos de código de software del programa de aplicación de software 333.

Aunque el codificador de video 100 de la figura 1 es un ejemplo de un canal de descodificación de video de alta eficiencia de codificación de video (HEVC), las etapas de procesamiento realizadas por los módulos 102 a 112, 114 y 115 son comunes a otros códecs de video tales como VC-1 o H.264/MPEG-4 AVC. El codificador de video 100 recibe datos 101 de fotogramas no codificados como una serie de fotogramas que incluyen muestras de luminancia y crominancia. El codificador de video 100 divide cada fotograma de los datos 101 de fotograma en conjuntos jerárquicos de unidades de codificación (CU), que se pueden representar, por ejemplo, como un árbol de unidades de codificación (CU).

El codificador de video 100 funciona emitiendo, desde un módulo de multiplexación 110, una serie de muestras de datos predichas conocida como unidad de predicción (PU) 120. Un módulo de diferencia 115 genera la diferencia entre la unidad de predicción (PU) 120 y una matriz correspondiente de muestras de datos recibidas de los datos

101 de fotograma, conociéndose la diferencia como muestras de datos residuales 122.

Las muestras de datos residuales 122 del módulo de diferencia 115 son recibidas por un módulo de transformación 102, que convierte la diferencia de una representación espacial en una representación del dominio de la frecuencia para crear los coeficientes de transformación 124 para cada unidad de transformación (TU) en el árbol de transformación. Para el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo, la conversión a la representación en el dominio de la frecuencia se implementa utilizando una transformada del coseno discreta modificada (DCT, Discrete Cosine Transform), en la cual se modifica una DCT tradicional para ser implementada utilizando desplazamientos y adiciones. Los coeficientes de transformación 124 son introducidos a continuación en un módulo de escala y cuantificación 103, y son escalados y cuantificados para generar los coeficientes residuales 126. El proceso de escala y cuantificación produce una pérdida de precisión. Los coeficientes residuales 126 se toman como entrada a un módulo de escalado inverso 105 que invierte la escala realizada por el módulo de escalado y cuantificación 103 para generar coeficientes de transformación nuevamente escalados 128, que son versiones nuevamente escaladas de los coeficientes residuales 126. Asimismo, se toman los coeficientes residuales 126 como entrada a un módulo de codificación por entropía 104, que codifica los coeficientes residuales en un flujo de bits codificado 113. Debido a la pérdida de precisión resultante del módulo de escalado y cuantificación 103, los coeficientes de transformación 128 nuevamente escalados no son idénticos a los coeficientes de transformación 124 originales. Los coeficientes de transformación 128 nuevamente escalados del módulo 105 de escalado inverso se emiten a continuación a un módulo 106 de transformación inversa. El módulo 106 de transformación inversa realiza una transformación inversa del dominio de frecuencia al dominio espacial para generar una representación en el dominio del espacio 130 de los coeficientes de transformación 128 nuevamente escalados idéntica a una representación en el dominio del espacio que es generada en un descodificador.

Un módulo de estimación de movimiento 107 genera vectores de movimiento 132 comparando los datos 101 de fotograma con los datos de fotograma anteriores almacenados en un módulo de memoria temporal 112 de fotogramas configurado dentro de la memoria 306. Los vectores de movimiento 132 son introducidos a continuación en un módulo de compensación de movimiento 108, que genera muestras de referencia interpredichas 134 filtrando muestras almacenadas en el módulo de memoria temporal 112 de fotogramas, teniendo en cuenta un desplazamiento espacial obtenido de los vectores de movimiento 132. No ilustrados en la figura 1, los vectores de movimiento 132 también son pasados como elementos de sintaxis al módulo de codificación por entropía 104 para codificación en el flujo de bits codificado 113. Un módulo de intrapredicción 109 de fotogramas genera muestras de referencia intrapredichas 136 utilizando muestras 138 obtenidas de un módulo de suma 114, que suma la salida 120 del módulo de multiplexación 110 y la salida 130 del módulo de transformación inversa 106.

Las unidades de predicción (PU) pueden ser codificadas utilizando procedimientos de intrapredicción o interpredicción. La decisión sobre si utilizar intrapredicción o interpredicción se toma según un compromiso entre la velocidad de transferencia y la distorsión entre la velocidad de transferencia de bits deseada del flujo de bits codificado 113 resultante y la magnitud de la distorsión de la calidad de la imagen introducida por cualquiera de los procedimientos de intrapredicción o interpredicción. Si se utiliza la intrapredicción, se selecciona un modo de intrapredicción de entre un conjunto de modos posibles, también según un compromiso entre la velocidad de transferencia y la distorsión. Se selecciona un modo de intrapredicción para cada unidad de predicción. El modelo 5.0 de prueba de codificación de video de alta eficiencia (HEVC) (HM-5.0) soporta 35 modos de intrapredicción, sin embargo, no todos los modos de intrapredicción pueden ser utilizados para todos los tamaños de unidades de predicción. Por ejemplo, una unidad de predicción de 8x8 puede tener 35 modos de intrapredicción disponibles para su selección, y una unidad de predicción de 4x4 puede tener 18 modos de intrapredicción disponibles para su selección en algunas implementaciones y 19 modos disponibles para su selección en otras implementaciones. El módulo de multiplexación 110 selecciona cualquiera de las muestras de referencia intrapredichas 136 del módulo de intrapredicción 109 de fotogramas o las muestras de referencia interpredichas 134 del bloque de compensación de movimiento 108, dependiendo del modo de predicción 142 actual, determinado por la lógica de control no ilustrada pero bien conocida en la técnica. El modo de predicción 142 es proporcionado asimismo al codificador por entropía 104 y, por lo tanto, se utiliza para determinar o establecer de otro modo el orden de escaneo de las unidades de transformación, tal como se describirá. La interpredicción de fotogramas utiliza solo un orden de escaneo diagonal, mientras que la intrapredicción de fotogramas puede utilizar un orden de escaneo diagonal, escaneo horizontal o escaneo vertical.

El módulo de suma 114 genera una suma 138 que es introducida en un módulo de filtro de desbloqueo 111. El módulo de filtro de desbloqueo 111 realiza el filtrado a lo largo de los límites del bloque, generando muestras desbloqueadas 140 que son escritas en el módulo de memoria temporal 112 de fotogramas configurado dentro de la memoria 306. El módulo de memoria temporal 112 de fotogramas es una memoria temporal con capacidad suficiente para almacenar datos de varios fotogramas anteriores para referencia en el futuro.

En el codificador de video 100, las muestras de datos residuales 122 dentro de una unidad de transformación (TU) son determinadas encontrando la diferencia entre las muestras de datos de los datos 101 de fotograma de entrada y la predicción 120 de las muestras de datos de los datos 101 de fotograma de entrada. La diferencia proporciona una representación espacial de los coeficientes residuales de la unidad de transformación (TU).

Los coeficientes residuales de una unidad de transformación (TU) son convertidos al mapa de relevancia bidimensional.

El mapa de relevancia de los coeficientes residuales en la unidad de transformación (TU) se escanea a continuación en un orden concreto, conocido como orden de escaneo, para formar una lista unidimensional de valores de indicador, llamada lista de indicadores de coeficiente significativo. El orden de escaneo puede ser descrito o especificado de otra manera mediante un patrón de escaneo, tal como el recibido con el modo de predicción 142 desde el módulo de intrapredicción 109. El patrón de escaneo puede ser horizontal, vertical, diagonal o en zigzag. La versión 5 del modelo de prueba de codificación de video de alta eficiencia (HEVC) realiza el escaneo hacia atrás, sin embargo, el escaneo hacia adelante también es posible. Para unidades de transformación (TU) de 16x16, 32x32, 4x16, 16x4, 8x32 y 32x8, se define un escaneo de dos niveles, en el que la unidad de transformación (TU) se divide en un conjunto de bloques secundarios, teniendo cada bloque secundario una forma cuadrada. En un nivel superior, el escaneo se realiza escaneando cada nivel inferior utilizando un escaneo tal como el escaneo diagonal hacia abajo, hacia la izquierda y hacia atrás. En el nivel inferior, también conocido como nivel de bloque secundario, el escaneo también se realiza utilizando un escaneo, tal como el escaneo diagonal hacia abajo, hacia la izquierda y hacia atrás. En la versión 5.0 del modelo de referencia HEVC, la operación de escaneo inicia un coeficiente residual después de un último coeficiente significativo (donde 'después' corresponde a la dirección de un escaneo hacia atrás de los coeficientes residuales) y avanza hasta que se alcanza una ubicación superior izquierda del mapa de relevancia. Las operaciones de escaneo que tienen esta propiedad y que cumplen con la versión 5.0 del modelo de referencia de HEVC se conocen como 'escaneados hacia atrás'. En el software de referencia de la versión 5.0 de HEVC, la ubicación del último coeficiente significativo se señala mediante la codificación de las coordenadas del coeficiente en la unidad de transformación (TU). Resultará evidente para las personas que estén familiarizadas con la técnica que la utilización del adjetivo "último" en este contexto depende del orden concreto de escaneo. El que puede ser el "último" coeficiente residual distinto de cero o el correspondiente indicador de coeficiente significativo de valor uno según un patrón de escaneo puede no ser el "último" según otro patrón de escaneo. La lista de indicadores de coeficientes significativos, que indica la relevancia de cada coeficiente residual antes del último coeficiente significativo, es codificada en el flujo de bits. No es necesario que el valor del indicador del último coeficiente significativo esté codificado explícitamente en el flujo de bits, porque la codificación anterior de la ubicación del indicador del último coeficiente significativo indicó implícitamente que este coeficiente residual era significativo.

La agrupación de coeficientes residuales de mayor valor hacia la parte superior izquierda de la unidad de transformación (TU) da como resultado que la mayoría de los indicadores de relevancia que aparecen al principio de la lista sean significativos, mientras que los indicadores de menor relevancia se encuentran más adelante en la lista.

Tal como se describió anteriormente, el codificador de video 100 comprende asimismo un módulo de codificación por entropía 104 que implementa un procedimiento de codificación por entropía. El módulo de codificación por entropía 104 genera elementos de sintaxis a partir de los datos del coeficiente residual (o coeficientes residuales) 126 de entrada recibidos del módulo de escalado y cuantificación 103. El módulo de codificación por entropía 104 emite el flujo de bits codificado 113 y se describirá con más detalle a continuación. Para el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo, el flujo de bits codificado 113 es delineado en unidades de capa de abstracción de la red (NAL, Network Abstraction Layer). Cada fragmento de un fotograma está contenido en una unidad NAL.

Existen varias alternativas para el procedimiento de codificación por entropía implementado en el módulo de codificación por entropía 104. El estándar de codificación de video de alta eficiencia (HEVC) en desarrollo es compatible con la codificación aritmética binaria adaptable al contexto (CABAC), una variante de la codificación aritmética binaria adaptativa al contexto (CABAC) que se encuentra en H.264/MPEG-4 AVC. Un esquema alternativo de codificación por entropía es el codificador por entropía en particiones de intervalos de probabilidad (PIPE, Probability Interval Partitioning Entropy), que es bien conocido en la técnica.

Para un codificador de video 100 que soporta múltiples procedimientos de codificación de video, uno de los procedimientos de codificación por entropía soportados se selecciona según la configuración del codificador 100. Además, en la codificación de las unidades de codificación de cada fotograma, el módulo de codificación por entropía 104 escribe el flujo de bits codificado 113, de tal manera que cada fotograma tiene uno o varios fragmentos por fotograma, y cada fragmento contiene datos de imagen para parte del fotograma. Generar un fragmento por fotograma reduce la sobrecarga asociada con la delimitación de cada límite de fragmento. Sin embargo, también es posible dividir el fotograma en varios fragmentos.

El descodificador de video 200 de la figura 2 puede ser implementado como uno o varios módulos de código de software del programa de aplicación de software 333 que se encuentran en la unidad de disco duro 305 y que están controlados en su ejecución por el procesador 305. En concreto, el descodificador de video 200 comprende los módulos 202 a 208 y 210 que pueden ser implementados, cada uno, como uno o varios módulos de código de software del programa de aplicación de software 333. Aunque el descodificador de video 200 se describe con referencia a un canal de descodificación de video de codificación de video de alta eficiencia (HEVC), las etapas del procesamiento realizadas por los módulos 202 a 208 y 209 son comunes a otros códecs de video que emplean codificación por entropía, tales como H.264/MPEG-4 AVC, MPEG-2 y VC-1.

El descodificador de video 200 recibe un flujo de bits codificado, tal como el flujo de bits codificado 113. El flujo de bits codificado 113 se puede leer desde la memoria 306, la unidad de disco duro 310, un CD-ROM, un disco

Blu-ray™ u otro medio de almacenamiento legible por ordenador. Alternativamente, el flujo de bits codificado 113 puede ser recibido desde una fuente externa tal como un servidor conectado a la red de comunicaciones 320 o un receptor de radiofrecuencia. El flujo de bits codificado 113 contiene elementos de sintaxis codificados que representan datos de fotograma para ser descodificados.

El flujo de bits codificado 113 es introducido en un módulo de descodificación por entropía 202 que extrae los elementos de sintaxis del flujo de bits codificado 113 y pasa los valores de los elementos de sintaxis a otros bloques en el descodificador de video 200. Se pueden implementar múltiples procedimientos de descodificación por entropía en el módulo de descodificación por entropía 202, tales como los descritos con referencia al módulo de codificación por entropía 104. Los datos del elemento de sintaxis 220 que representan los datos del coeficiente residual son pasados a un módulo de escalado inverso y transformación 203 y los datos 222 del elemento de sintaxis, que representan información del vector de movimiento, son pasados a un módulo de compensación de movimiento 204. El módulo de módulo de escalado inverso y transformación 203 realiza un escalado inverso sobre los datos del coeficiente residual para crear coeficientes de transformación reconstruidos. A continuación, el módulo 203 realiza una transformación inversa para convertir los coeficientes de transformación reconstruidos de una representación en el dominio de la frecuencia a una representación en el dominio del espacio, generando muestras residuales 224, tal como la transformada inversa descrita con referencia al módulo de transformación inversa 106.

El módulo de compensación de movimiento 204 utiliza los datos 222 del vector de movimiento del módulo 202 de descodificación por entropía, combinados con los datos 226 de fotogramas anteriores de un bloque 208 de memoria temporal de fotogramas, configurado en la memoria 306, para generar muestras de referencia interpredichas 228 para una unidad de predicción (PU), que es una predicción de datos de fotograma descodificados de salida. Cuando un elemento de sintaxis indica que la unidad de codificación actual fue codificada utilizando intrapredicción, el módulo 205 de intrapredicción de fotogramas genera muestras 230 de referencia intrapredichas para la unidad de predicción (PU) utilizando muestras espacialmente próximas a la unidad de predicción (PU). Las muestras espacialmente próximas se obtienen a partir de una suma 232 emitida por un módulo de suma 210. El módulo de multiplexación 206 selecciona muestras de referencia intrapredichas o muestras de referencia interpredichas para la unidad de predicción (PU) en función del modo de predicción actual, lo que se indica mediante un elemento de sintaxis en el flujo de bits codificado 113. La matriz de muestras 234 emitida por el módulo de multiplexación 206 es sumada a las muestras residuales 224 del módulo de escalado inverso y transformación 203 mediante el módulo de suma 210 para generar la suma 232 que, a continuación, es introducida en cada uno del módulo 207 de filtro de desbloqueo y el módulo 205 de intrapredicción de fotogramas. En contraste con el codificador 100, el módulo 205 de intrapredicción de fotogramas recibe un modo de predicción 236 desde el descodificador por entropía 202. El multiplexador 206 recibe una señal de selección de intrapredicción de fotogramas/interpredicción de fotogramas desde el descodificador por entropía 202. El módulo 207 de filtro de desbloqueo realiza el filtrado a lo largo de los límites del bloque de datos, para suavizar los artefactos visibles a lo largo de los límites del bloque de datos. La salida del módulo 207 de filtro de desbloqueo se escribe en el módulo 208 de memoria temporal de fotogramas configurado en la memoria 306. El módulo 208 de memoria temporal de fotogramas proporciona almacenamiento suficiente para contener múltiples fotogramas descodificados para futuras referencias. Los fotogramas descodificados 209 también se envían desde el módulo 208 de memoria temporal de fotogramas.

El codificador por entropía 104 se describirá con referencia a la figura 4. Los elementos de sintaxis, tales como los coeficientes residuales 401, son introducidos en un módulo de binarización 404. El tamaño 402 de una unidad de transformación (TU) es introducido en el módulo de binarización 404. El tamaño de la unidad de transformación (TU) indica el tamaño de la unidad de transformación (TU) a codificar. Un patrón de escaneo 403 es introducido en el módulo de binarización 404. El módulo de binarización 404 binariza cada elemento de sintaxis en una secuencia de bins. Cada bin comprende un valor de bin 406 y un índice de contexto 405. Un modelo de contexto 407 recibe el valor de bin 406 y el índice de contexto 405, lo que genera un contexto 408, seleccionado según el índice de contexto 405. El contexto 408 es actualizado según el valor del bin 405. El procedimiento para actualizar el contexto 408 concuerda con el utilizado por la codificación aritmética binaria adaptable al contexto (CABAC) en H.264/MPEG-4 AVC. Un codificador aritmético binario 409 utiliza el contexto 408 y el valor del bin 406 para codificar el bin en el flujo de bits codificado 113.

El descodificador por entropía 202 se describirá con referencia a la figura 5. Un módulo de binarización inversa 503 recibe el tamaño 502 de la unidad de transformación (TU) y un patrón de escaneo 501. El módulo de binarización inversa 503 emite los coeficientes residuales 509 realizando la operación inversa del módulo de binarización 404. Un índice de contexto 504 es emitido desde el módulo de binarización inversa 503 para cada bin a ser descodificado. Un modelo de contexto 505 genera un contexto 506 seleccionado por el índice de contexto 504. Un descodificador aritmético binario 507 descodifica un valor de bin 508 del flujo de bits codificado 113 utilizando el contexto 506. El modelo de contexto 505 recibe el valor del bin 508 y lo utiliza para actualizar el contexto 506. El módulo de binarización inversa 503 también recibe el valor del bin 508.

Una unidad de codificación más grande (LCU) 600 a modo de ejemplo se describirá con referencia a la figura 6A. La unidad de codificación más grande (LCU) 600 tiene una forma cuadrada de 64x64 muestras de luma. La unidad de codificación más grande 600 se subdivide recursivamente en una unidad de codificación 1 601 a la unidad de codificación 10 608. La división de la unidad de codificación más grande (LCU) 600 hace uso de niveles jerárquicos,

lo que permite la división recursiva de una zona que contiene la unidad de codificación más grande (LCU) en cuatro zonas del mismo tamaño, de forma cuadrada, que no se superponen, cada una de ellas con la mitad de las dimensiones vertical y horizontal de la zona de partida, y que ocupan conjuntamente toda el área de la zona de partida. Una vez que una zona ya no se subdivide en zonas más pequeñas, existe una unidad de codificación que ocupa completamente la zona. En un nivel concreto de subdivisión, el tamaño de la zona resulta igual a un tamaño conocido como unidad de codificación más pequeña (SCU), momento en el que ya no es posible una subdivisión adicional o puede estar impedida de otra manera, por convención o por razones prácticas. Para el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo, el tamaño de la unidad de codificación más pequeña (SCU) está configurado como muestras de luma de 8x8. Cada unidad de codificación tiene uno de varios tamaños posibles, tal como la unidad de codificación 1 601, que tiene un tamaño de 32x32, una unidad de codificación 2 602, que tiene un tamaño de 16x16 y una unidad de codificación 4 603, que tiene un tamaño de 8x8. También son posibles otros tamaños de unidad de codificación, dependiendo del tamaño de la unidad de codificación más grande (LCU) seleccionada y del tamaño de la unidad de codificación más pequeña (SCU) utilizada en el estándar de codificación de video de alta eficiencia (HEVC) en desarrollo.

La subdivisión de la unidad de codificación más grande (LCU) 600 se describirá más detalladamente con referencia a la figura 6B. En este caso, en una zona de la unidad de codificación más grande (LCU) 604, se produce una división, que divide la zona de la unidad de codificación más grande (LCU) en cuatro zonas del mismo tamaño, tal como una zona 605. Se utiliza una nueva división para obtener otras cuatro zonas más pequeñas, tales como la zona 607. Una vez que el tamaño de la zona alcanza las dimensiones de la unidad de codificación más pequeña (SCU), tal como una zona 606, no es posible realizar ninguna otra división. En cada zona en la que no se produce ninguna división adicional, una unidad de codificación ocupa por completo la zona.

La unidad de codificación más grande (LCU) 604 de la figura 6B se puede representar asimismo como un árbol de codificación 630 jerárquico tal como el mostrado en la figura 6C. Cuando se utiliza un árbol jerárquico para representar la unidad de codificación más grande (LCU), cada una de las unidades de codificación formará nodos hoja, mientras que las zonas que contienen zonas subdivididas adicionales formarán nodos que no son de hoja. El nodo raíz 632 del árbol 630 se basa en la zona 604 de la figura 6B y está en un nivel que representa muestras de 64x64. Por debajo del nodo raíz está dispuesta una segunda capa que representa zonas de muestras de 32x32, tal como la zona 605. La unidad de codificación 1 de la figura 6A está representada como el nodo hoja 634, mientras que la zona que contiene las unidades de codificación 2 a 8 está representada por el nodo que no es de hoja 640. Las zonas de tamaño 16x16 se muestran en un tercer nivel del árbol 630, representando el nodo hoja 636 la unidad de codificación 2, y la zona 607 de la figura 6B se representa como un nodo que no es de hoja 642. Las cuarta y última capa del árbol 630 representa zonas de tamaño 8x8, tal como la zona 606, que contiene la unidad de codificación 4 603, que está representada por el nodo hoja 638. Resulta claro a partir de lo anterior, que el tamaño de las unidades de codificación en el árbol disminuye a medida que aumenta la profundidad del árbol.

Tal como se describirá más detalladamente a continuación, se utiliza un indicador de división para indicar que una zona es un nodo hoja en la unidad de codificación más grande (LCU). El árbol de codificación 630 se puede considerar como una forma de representar una estructura de codificación de la unidad de codificación más grande (LCU).

Con referencia a las figuras 6 y 7, se describirá un flujo de bits 700 que codifica la unidad de codificación más grande (LCU) 600 de manera convencional. Como un fotograma de una imagen de video puede tener muchas unidades de codificación más grandes (LCU) por fragmento, un flujo de bits codificado, tal como el flujo de bits codificado 113, puede comprender muchos casos del flujo de bits 700 que se muestra en la figura 7. La figura 7 adopta una convención para representar elementos de sintaxis binarizados codificados de tal manera que los fragmentos marcados con “S” contienen un indicador de división codificado aritméticamente, los fragmentos marcados con ‘A’ contienen uno o varios elementos de sintaxis binarizados codificados aritméticamente o una porción o porciones de los mismos, los fragmentos marcados con ‘B’ contienen uno o varios elementos de sintaxis binarizados codificados por derivación o una porción o porciones de los mismos, y los fragmentos marcados con ‘A, B’ contienen uno o varios elementos de sintaxis binarizados codificados utilizando una mezcla de codificación aritmética y codificación por derivación. El flujo de bits 700 representa una porción del flujo de bits codificado 113, ya que los fragmentos consisten habitualmente en múltiples unidades de codificación más grandes (LCU) concatenadas entre sí. Para los tamaños de fotograma que no son múltiplos enteros de las dimensiones de la LCU, la inferencia de los indicadores de división evita que el límite del fotograma pase a través de una unidad de codificación. Las unidades de codificación que quedarían fuera del límite del fotograma no están codificadas en un flujo de bits. La unidad de codificación 1 601 está codificada en el flujo de bits 700 en una componente de flujo de bits 1 701. La unidad de codificación 2 a la unidad de codificación 10 están codificadas asimismo en el flujo de bits 700 en una componente de flujo de bits 2 a un componente de flujo de bits 10.

Se utiliza un indicador de división para indicar que una zona está dividida, indicando un valor de indicador de 1 que la zona está dividida, mientras que un valor de indicador de 0 indica que la zona no está dividida. Las zonas que están divididas se subdividen en cuatro zonas más pequeñas superpuestas de igual tamaño, que, conjuntamente, ocupan la totalidad de la zona padre. Cualquier zona que tenga el mismo tamaño que la unidad de codificación más pequeña (SCU) predeterminada tendrá un valor 0 inferido para el indicador de división, para indicar que la zona no está subdividida. Cualquier zona que sea más grande que el tamaño de las unidades de codificación más pequeñas

requiere la codificación de un indicador de división.

Un indicador de división 709 indica que la zona 604 de la unidad de codificación más grande (LCU) 600 está dividida en cuatro zonas de 32x32, tal como la zona 605. Un indicador de división 710 indica que la zona 605 no se divide más. La unidad de codificación 4 603 es una unidad de codificación más pequeña (SCU), por lo que no es posible realizar más divisiones. Por lo tanto, los indicadores de división no están codificados para cada una de las unidades de codificación 4 a 7. Sin embargo, hay un indicador de división de valor uno para indicar que una zona 607 está subdividida. El indicador de división 711 para la zona 607 está situado antes de la unidad de codificación 4 603.

El componente del flujo de bits 1 701 contiene elementos de sintaxis binarizados que utilizan una mezcla de codificación aritmética y codificación por derivación. Un modo de predicción codificado aritméticamente 703 determina si la unidad de codificación 1 601 utiliza interpredicción o intrapredicción. Si la unidad de codificación utiliza intrapredicción, un indicador 704 de modo más probable codificado aritméticamente codifica si se utiliza el modo más probable para la intrapredicción o si se utiliza un esquema alternativo para codificar el modo de intrapredicción. Si se está utilizando un modo más probable, un código 705 de modo de intrapredicción por derivación codifica un índice de modo más probable con una longitud de un bit. El índice de modo más probable determina cuál de los dos modos predeterminados más probables de intrapredicción se utiliza para la unidad de codificación. Si no se utiliza un modo más probable, el código 705 de modo de intrapredicción codifica un modo restante que especifica un modo de intrapredicción para la unidad de codificación. El código 705 de modo de intrapredicción puede tener una longitud de 5 bits o 6 bits para el modo restante. Un bloque de datos 706 utiliza la codificación aritmética y por derivación para una o varias unidades de transformación dentro de la unidad de codificación 601. El componente del flujo de bits 1 701 contiene todos los elementos de sintaxis necesarios para descodificar la unidad de codificación 1. De manera similar, los componentes del flujo de bits 2 a 10 contienen los elementos de sintaxis necesarios para descodificar las unidades de codificación 2 a 10 respectivamente.

Un flujo de bits 800 según la presente invención que codifica la unidad de codificación más grande (LCU) 600 se describirá con referencia a las figuras 6 y 8. La figura 8 adopta la convención de la figura 7 para representar elementos de sintaxis binarizados codificados. El flujo de bits 800 representa una porción del flujo de bits codificado 113 que codifica la unidad de codificación más grande (LCU) 600. El flujo de bits 800 tiene tres porciones que se pueden ver en un primer nivel de detalle 820 que son un primer bloque de datos 801 codificado aritméticamente que, agrupa información sobre la estructura de la unidad de codificación de las unidades de codificación 1 a 10, un segundo bloque de datos 802 codificado por derivación, que agrupa información sobre los modos de intrapredicción para las unidades de codificación 1 a 10, y un tercer bloque de datos 803, que contiene datos codificados de manera aritmética y por derivación y grupos de información para datos residuales para las unidades de codificación 1 a 10. A diferencia del flujo de bits 700, cada una de las tres porciones del flujo de bits 800 puede contener información acerca de las unidades de codificación 1 a 10.

El primer bloque de datos codificado aritméticamente se utiliza preferentemente para almacenar indicadores de división, modo de predicción, y cuando se está utilizando la interpredicción, la información de modo más probable para las unidades de codificación 1 a 10 según sea necesario. El primer bloque de datos se ilustra con más detalle en un segundo nivel de detalle 830 del flujo de bits 800 en la figura 8. Tal como se muestra en el segundo nivel de detalle 830, un primer indicador de división 813 tiene un valor de 1, para indicar que la zona 604 de la unidad de codificación más grande (LCU) 600 está dividida en cuatro zonas de 32x32, tal como la zona 605. Un indicador de división 807 tiene un valor de 0, para indicar que la zona 605 no tiene más divisiones. Un modo de predicción 808 codifica un valor para indicar si la unidad de codificación 1 utiliza la interpredicción o la intrapredicción. Cuando la unidad de codificación 1 utiliza la intrapredicción, el indicador de modo más probable 809 indica si se utilizó el modo más probable o el modo restante para la intrapredicción de la unidad de codificación. Otros casos de indicadores de división, valores del modo de predicción e indicadores de modo más probable se codifican en una porción del flujo de bits 804 para representar las unidades de codificación 2 a 10 de la unidad de codificación más grande (LCU) 600. En primer lugar, el indicador de división 813, el indicador de división 807, el modo de predicción 808, el indicador de modo más probable 809 y la porción del flujo de bits 804 forman todos parte de la porción del flujo de bits 801, que puede consistir exclusivamente en elementos de sintaxis codificados aritméticamente.

El segundo bloque de datos 802 contiene datos de derivación 810 que están presentes en el flujo de bits 800 cuando la unidad de codificación 1 601 utiliza la intrapredicción. Cuando se está utilizando la intrapredicción y el indicador de modo más probable 809 indica que se está utilizando el modo más probable, los datos de derivación 810 son un índice que codifica la utilización de uno de los dos modos más probables. El índice ocupa una longitud fija de un bit. Alternativamente, cuando se utiliza la intrapredicción y el indicador de modo más probable 809 indica que se está utilizando el modo restante, los datos de derivación 810 son un índice que codifica la utilización de uno de los 33 modos restantes de intrapredicción (de los 35 modos posibles de intrapredicción, se excluyen los dos modos más probables, quedando 33 modos restantes). En este caso, los datos de derivación 810 tienen una longitud de 5 bits o 6 bits, dependiendo del modo de intrapredicción codificado. La longitud o el tamaño de los datos de derivación 810 pueden ser determinados a partir de los primeros 5 bits de los datos de derivación 810, establecidos por la estructura de la unidad de codificación. Es posible determinar si el sexto bit del flujo de bits es necesario, después de inspeccionar los primeros 5 bits. Cuando se utiliza la interpredicción para la unidad de codificación 1 601, se omiten los datos de derivación 810 del flujo de bits 800. Existen otros casos de datos de derivación 810 para las

unidades de codificación 2 a 10 en un bloque de datos de derivación 805 si, por lo menos, una de las unidades de codificación 2 a 10 utiliza la intrapredicción. Un bloque de datos de derivación 802 codifica los datos de derivación 810 y el bloque de datos de derivación 805 cuando sea necesario.

El tercer bloque de datos 803 se muestra con más detalle como bloque de datos codificados aritméticamente y por derivación 811. El bloque de datos 811 codifica una o varias unidades de transformación dentro de la unidad de codificación 1 601 que contiene coeficientes residuales para las unidades de transformación que pueden ser utilizados, con la información del modo de predicción para generar datos de video. Un indicador 812 de final de fragmento codificado aritméticamente está presente en las mismas condiciones que se describen con referencia a la figura 7.

Un procedimiento 900 para descodificar el flujo de bits 800 se describirá con referencia a las figuras 6, 8 y 9. El procedimiento 900 recibe el flujo de bits 800 y procesa los tres bloques de datos para permitir la descodificación de las unidades de codificación en el flujo de bits 800. El procedimiento 900 comienza con una etapa 901 de determinación del valor del indicador de división, en la que se determina el valor de un indicador de división, tal como el indicador de división 807. Cuando la unidad de codificación es más grande que la unidad de codificación más pequeña (SCU), el valor del indicador de división se determina mediante la descodificación de un indicador de división del flujo de bits 800. Cuando la unidad de codificación tiene el mismo tamaño que la unidad de codificación más pequeña (SCU), tal como la unidad de codificación 4 606, entonces se infiere que el valor del indicador de división es cero.

El valor del indicador de división se utiliza a continuación para determinar si la estructura de la unidad de codificación está actualmente en un nodo hoja. Si el valor del indicador de división es cero, una etapa 902 de prueba del nodo hoja pasa el control a la etapa 903 del modo de predicción de la unidad de codificación. De lo contrario, la etapa 902 de prueba del nodo hoja pasa el control de nuevo a la etapa 901 de determinación del valor del indicador de división, aumentado la profundidad del árbol de codificación para indicar una zona en un nivel por debajo del nivel actual en el árbol de codificación, como el árbol 630 descrito anteriormente en relación con la figura 6B. Las zonas son procesadas en un orden de escaneo de ráster procesando el árbol de codificación en una manera de primero la profundidad. La utilización del orden de escaneo de ráster garantiza que las unidades de codificación 1 a 10 en la figura 6A son procesadas en orden.

La etapa 903 de modo de predicción de la unidad de codificación determina un valor del modo de predicción. El valor del modo de predicción se determina mediante la descodificación de un modo de predicción, tal como el modo de predicción 808. El modo de predicción especifica tanto el modo de predicción utilizado para la unidad de codificación como el modo de partición utilizado para dividir la unidad de codificación en una o varias unidades de predicción. Los modos de partición posibles son NxN o 2Nx2N. Si el modo de partición es NxN, entonces la unidad de codificación se divide en 4 unidades de predicción, cada una con un modo de predicción. Si el modo de partición es 2Nx2N, entonces la unidad de codificación solo contiene una unidad de predicción. Los modos de partición NxN y 2Nx2N dan como resultado unidades de predicción que tienen una forma cuadrada. También son posibles otros modos de partición, por ejemplo, 2NxN y Nx2N, dando como resultado unidades de predicción de forma rectangular. Se debe tener en cuenta que la intrapredicción o la interpredicción se especifican al nivel de la unidad de codificación, por lo que para NxN, las cuatro unidades de predicción serán de intrapredicción; sin embargo, cada unidad de predicción puede tener un modo de intrapredicción diferente, por lo que cada unidad de predicción tiene indicadores de modo más probable (MPM, Most Probable Mode) y modo de predicción independientes. Aunque el procedimiento 900 se describe, en general, en relación con cada unidad de codificación que tiene una sola unidad de predicción, el procedimiento 900 puede ser extendido para abarcar las unidades de codificación que contienen múltiples unidades de predicción.

Cuando el valor del indicador de división es cero y el valor del modo de predicción para la unidad de codificación especifica la intrapredicción, una etapa 904 de indicador de MPM determina el valor del indicador de modo más probable. El valor del indicador de modo más probable se determina mediante la descodificación de un indicador de modo más probable, tal como el indicador de modo más probable 804 de la figura 8. Una etapa 905 de prueba de si hay más nodos determina si se ha encontrado la última unidad de codificación en la unidad de codificación más grande (LCU). Si es así, el control pasa a una etapa 906 de determinación del modo de intrapredicción. En caso contrario, el control vuelve a la etapa 901 de determinar el valor del indicador de división.

Para una unidad de codificación de intrapredicción de 32x32, tal como la unidad de codificación 1 601 de la figura 6, la unidad de codificación puede contener una, dos o cuatro unidades de predicción, dependiendo del modo de partición de la unidad de codificación. Las etapas 906 y 907 son iteradas sobre la estructura de la unidad de codificación que fue determinada en las etapas 901 a 905. La etapa 906 de determinar el modo de intrapredicción determina como sigue el modo de intrapredicción para una unidad de predicción. Si el valor de indicador de modo más probable para una unidad de predicción indica que se ha utilizado un modo más probable, entonces se descodifica un valor del índice de modo más probable de un bit del flujo de bits 800 utilizando la descodificación por derivación. Un valor de índice de modo más probable de un bit indica cuál de los dos modos más probables posibles se está utilizando. De lo contrario, el valor del indicador de modo más probable indica la utilización de un modo restante y un valor del modo restante es descodificado del flujo de bits 800 utilizando la descodificación por derivación. El número de valores válidos del modo de intrapredicción y el intervalo del código de longitud variable dependen del tamaño de la unidad de predicción. De los modos de intrapredicción disponibles para un tamaño de unidad de predicción determinado, el número de modos restantes es igual al número de modos más probables

restados del número de modos disponibles. Cuando el número de modos restantes es una potencia de dos, el modo restante puede utilizar un código de longitud fija, de lo contrario, se utiliza un código de longitud variable. Por ejemplo, una unidad de predicción de 4x4 intrapredicha con 18 modos de intrapredicción disponibles y dos modos más probables tiene 16 modos restantes y, por lo tanto, puede utilizar un código de cuatro bits para codificar el modo restante. Alternativamente, una unidad de predicción de 4x4 intrapredicha con 19 modos de intrapredicción disponibles y dos modos más probables tiene 17 modos restantes y, por lo tanto, puede utilizar un código de cuatro bits o cinco bits para codificar el modo restante. Una unidad de predicción de 8x8 intrapredicha con dos modos más probables tiene 33 modos restantes y, por lo tanto, puede utilizar un código de longitud variable de cinco o seis bits. En una implementación, el código de longitud variable es descodificado leyendo, por lo menos, un número suficiente de bins para determinar la longitud del código de longitud variable utilizado para el modo restante. Para dichas unidades de predicción, es posible descodificar cinco bits para determinar si se debe descodificar un sexto bit. Como resultado, se puede realizar una segunda lectura para descodificar la porción siguiente del modo restante en base a los bits suficientes descodificados. Una implementación alternativa introduce un indicador de modo restante codificado aritméticamente, codificado después del indicador de modo más probable, lo que indica que la unidad de predicción utiliza un modo restante predeterminado. Si el modo restante predeterminado, por ejemplo, la 'intrapredicción plana' no se está utilizando, uno de los otros modos restantes es codificado utilizando el elemento de sintaxis de modo restante codificado por derivación. Por ejemplo, si una unidad de predicción de 4x4 intrapredicha tiene 19 modos disponibles, con dos modos más probables y uno restante predeterminado, existen otros 16 modos restantes, que pueden ser codificados utilizando un elemento de sintaxis de modo restante de longitud fija de cuatro bits. Además, si una unidad de predicción de 8x8 intrapredicha tiene 35 modos disponibles, con dos modos más probables y un modo restante predeterminado, existen otros 32 modos restantes, que pueden ser codificados utilizando un elemento de sintaxis de modo restante de longitud fija de cinco bits. Cuando el número de modos restantes u otros modos restantes es una potencia de dos, un código de longitud fija es suficiente para codificar el modo restante u otro modo restante utilizado. El modo de intrapredicción para la unidad de predicción se determina por lo tanto utilizando el valor del indicador de modo más probable y uno del valor del índice de modo más probable o del valor de modo restante. Alternativamente, el indicador de modo restante predeterminado y, opcionalmente, el otro modo restante, se utilizan para determinar el modo de intrapredicción para la unidad de predicción. Cuando múltiples códigos de longitud variable están concatenados, es posible realizar una lectura de la longitud mínima de los códigos combinados para determinar si son necesarias lecturas adicionales para completar la descodificación de los códigos. El flujo de bits 800 puede codificar cada una de las porciones de longitud mínima de los modos restantes de longitud variable adyacentes en el segundo bloque de datos 802 y, a continuación, codificar los datos restantes de los modos restantes de longitud variable en el segundo bloque de datos 802. Utilizando esta codificación, es posible que las implementaciones lean todas las porciones de longitud mínima en una lectura y determinen la longitud de los datos restantes para completar la lectura de los modos restantes de longitud variable.

Una etapa 907 de prueba de si hay más nodos determina si hay más nodos del árbol de codificación que necesitan que su modo de intrapredicción sea determinado. El resultado de ejecutar la etapa 907 es que la etapa 906 de determinar el modo de intrapredicción es iterada sobre todos los nodos de la unidad de codificación más grande (LCU).

Una etapa 908 de descodificar datos residuales descodifica el tercer bloque de datos 803. La etapa 908 de descodificar datos residuales descodifica cada una de las unidades de transformación para las unidades de codificación 1 a 10 en la unidad de codificación más grande (LCU) 600. A medida que se descodifica cada unidad de transformación, el módulo de escalado inverso y transformación 203 convierte los datos residuales del dominio de la frecuencia al dominio del espacio, para generar las muestras residuales 224. Utilizando el modo intrapredicción, el módulo 205 de intrapredicción de fotogramas determina la predicción 234 para cada unidad de predicción. Otras etapas para descodificar la unidad de codificación más grande (LCU) 600 se corresponden con la operación descrita en la figura 2.

Se describirá un procedimiento 1000 para descodificar el flujo de bits 800 con referencia a la figura 10. El procedimiento 1000 comienza con una etapa 1001 de determinar la estructura de la unidad de codificación que construye una estructura de la unidad de codificación para representar la división de una unidad de codificación más grande (LCU) en múltiples unidades de codificación en base a la información del indicador de división en el primer bloque de datos 801 codificado aritméticamente. Otra información acerca de las unidades de codificación se determina asimismo a partir del primer bloque de datos 801. La información incluye un valor del modo de predicción para la unidad de codificación e indicadores de MPM para cualquier unidad de predicción de la unidad de codificación. Más detalles de cómo se hace esto se han descrito anteriormente en la figura 9 en la etapa 901 de determinar el valor del indicador de división, la etapa 902 de nodo hoja, la etapa 903 de determinar el valor del modo de predicción de la unidad de codificación, la etapa 905 de determinar el valor del indicador de MPM de la unidad de predicción, y la etapa 906 de si hay más nodos.

A continuación, una etapa 1002 de descodificar datos codificados por derivación descodifica el segundo bloque de datos 802 codificado por derivación. El segundo bloque de datos 802 codificado por derivación proporciona información acerca de los modos de intrapredicción utilizados para cada una de las unidades de codificación intrapredichas de la mayor unidad de codificación (LCU). La etapa 1002 de descodificar los datos codificados por derivación se describe con más detalle en la etapa 906 de determinar el modo de intrapredicción y la etapa 907 de si hay más nodos de la figura 9, descrita anteriormente.

A continuación, el procedimiento 1000 continúa hacia una etapa 1003 de descodificar datos residuales, en la que los datos residuales son descodificados del tercer bloque de datos 803. Tal como se describió anteriormente, el tercer bloque de datos 803 contiene datos codificados aritméticamente y por derivación. La etapa 1003 de descodificar datos residuales se describe con más detalle en la etapa 908 de descodificar datos residuales de la figura 9 anterior.

Finalmente, una etapa 1004 de formar unidades de codificación combina el modo de intrapredicción de la etapa 1002 de descodificar datos codificados por derivación y los datos residuales de la etapa 1003 de descodificar los datos residuales para formar unidades de codificación descodificadas, tal como se describe en relación con la figura 2. Una vez que una unidad de codificación descodificada ha sido formada, el modo de intrapredicción y los datos residuales pueden ser combinados para formar parte de un fotograma de video descodificado.

Aunque el procedimiento 1000 se ha descrito en relación con el procedimiento 900 de la figura 9, el procedimiento también puede abarcar otros procedimientos, tales como el procedimiento 1400 de la figura 14 y el procedimiento 1700 de la figura 17, que se describirán a continuación.

Un procedimiento 1100 para codificar el flujo de bits 800 se describirá ahora con referencia a la figura 11. El procedimiento 1100 codifica el flujo de bits 800 y genera los tres bloques de datos para habilitar la descodificación de las unidades de codificación en el flujo de bits 800. El procedimiento 1100 se inicia con una etapa 1101 de codificar el valor del indicador de división, en la que el valor de un indicador de división, tal como el indicador de división 807, es codificado. Las reglas que rigen la ubicación de los indicadores de división han sido explicadas con más detalle anteriormente, en relación con las figuras 6A y 6B. Cuando una unidad de codificación es más grande que la unidad de codificación más pequeña (SCU), un indicador de división codifica el valor del indicador de división apropiado en el flujo de bits 800. Sin embargo, el indicador de división no se codifica cuando la unidad de codificación tiene el mismo tamaño que la unidad de codificación más pequeña (SCU), tal como la unidad de codificación 4 606 de la figura 6B.

Si el valor del indicador de división es cero, una etapa 1102 de prueba de nodo hoja pasa el control a una etapa 1103 de codificar el valor del modo de predicción de la unidad de codificación, ya que el cero del indicador de división indica que la unidad de codificación actual es un nodo hoja del árbol de codificación. Si el nodo actual del árbol de codificación es un nodo que no es de hoja, la etapa 1102 de prueba del nodo hoja devuelve el control a la etapa 1101 de codificar el valor del indicador de división, incrementándose la profundidad del árbol de codificación hasta una zona un nivel por debajo del nivel actual en el árbol de codificación, tal como el árbol 630 descrito anteriormente en relación con la figura 6C. Al igual que con el procedimiento 900 de la figura 9, las zonas son procesadas en un orden de escaneo de ráster, procesando el árbol de codificación de una manera de primero la profundidad. La utilización del orden de escaneo de ráster garantiza que las unidades de codificación 1 a 10 en la figura 6A son procesadas en orden.

Una etapa 1103 de codificar el modo de predicción de la unidad codifica un valor del modo de predicción. Para fragmentos que contienen tanto unidades de predicción interpredichas como unidades de predicción intrapredichas, el modo de predicción especifica el tipo de predicción utilizado. Para los fragmentos que contienen solo unidades de predicción intrapredichas, el modo de predicción no está codificado en el flujo de bits codificado 113. De manera similar al procedimiento 900 de la figura 9, el modo de predicción especifica tanto el modo de predicción utilizado para la unidad de codificación como el modo de partición. Mientras que el procedimiento 1100 se describe en relación con una unidad de codificación con una sola unidad de predicción, el procedimiento puede ser extendido para abarcar unidades de codificación que contienen múltiples unidades de predicción.

Cuando el valor del indicador de división es cero y el valor del modo de predicción para la unidad de codificación especifica intrapredicción, una etapa 1104 de codificar el indicador de MPM codifica un valor del indicador de modo más probable. El módulo de intrapredicción 109 de fotogramas de la figura 1 determina el modo de intrapredicción para una unidad de predicción. El módulo de intrapredicción 109 de fotogramas determina asimismo los dos modos más probables para la intrapredicción. Si el modo de intrapredicción determinado es igual a uno de los modos más probables, el valor del indicador de modo más probable se establece en 1, lo que indica la utilización del modo más probable. De lo contrario, el valor del indicador de modo más probable se establece en 0, lo que indica la utilización de un modo restante. Un indicador de modo más probable, tal como el indicador de modo más probable 804 de la figura 8, es codificado como el valor del indicador de modo más probable. Una etapa 1105 de prueba de si hay más nodos determina si se ha encontrado la última unidad de codificación en la unidad de codificación más grande (LCU). Si es así, el control pasa a una etapa 1106 de codificar datos por derivación. Si no, el control vuelve a ejecutar la etapa 1101 de codificar el valor del indicador de división.

Para una unidad de codificación de intra-predicción de 32x32, tal como la unidad de codificación 1 601 de la figura 6, la unidad de codificación puede contener una, dos o cuatro unidades de predicción, dependiendo del modo de partición de la unidad de codificación. La etapa 1106 de codificar datos de derivación codifica el modo de intrapredicción para una unidad de predicción de la siguiente manera. Si el valor del indicador de modo más probable para una unidad de predicción indica que se ha utilizado un modo más probable, entonces un valor del índice de modo más probable de un bit, que indica cuál de los dos modos más probables disponibles fue seleccionado, es codificado en el flujo de bits 800 utilizando descodificación por derivación. De lo contrario, el valor del indicador de modo más

probable indica la utilización de un modo restante, y un valor de modo restante es codificado en el flujo de bits 800 utilizando codificación por derivación. Cuando se concatenan varios valores de índice de modo más probable o valores de modo restante, es posible realizar una escritura de los códigos combinados en una sola operación, en lugar de escribir el código para cada unidad de predicción de manera separada.

Una etapa 1107 de prueba de si hay más nodos determina si hay más nodos del árbol de codificación que necesitan que su modo de intrapredicción sea determinado. El resultado es que se ejecuta la etapa 1106 de codificar datos de derivación para iterar sobre todos los nodos de la unidad de codificación más grande (LCU). La iteración sobre la etapa 1106 de codificar datos de derivación y la etapa 1107 de si hay más nodos pueden tener lugar antes de escribir los datos de derivación en el flujo de bits codificado 113 para predeterminar la longitud de los datos que se escribirán.

Una etapa 1108 de codificar los datos residuales codifica el tercer bloque de datos 803. La etapa 1108 de codificar los datos residuales codifica cada una de las unidades de transformación para las unidades de codificación 1 a 10 en la unidad de codificación más grande (LCU) 600 en el flujo de bits codificado 113. Para codificar cada unidad de transformación, las muestras residuales 122 son transformadas mediante el bloque de transformación 102 en los coeficientes de transformación 124. El bloque de escalado y cuantificación 103 convierte a continuación los coeficientes de transformación 124 en coeficientes residuales 126. Los coeficientes residuales 126 son codificados mediante el codificador por entropía 104 al flujo de bits codificado 113. Otras etapas para codificar la unidad de codificación más grande (LCU) 600 se corresponden con la operación descrita en el codificador de video 100 de la figura 1.

Se describirá un procedimiento 1200 para codificar el flujo de bits 800 con referencia a la figura 12. El procedimiento 1200 se inicia con una etapa 1201 de codificar la estructura de la unidad de codificación, que codifica una estructura de la unidad de codificación para representar la división de una unidad de codificación más grande (LCU) en múltiples unidades de codificación mediante la codificación de la información del indicador de división en el primer bloque de datos codificado aritméticamente 801. Otra información acerca de las unidades de codificación es codificada asimismo en el primer bloque de datos 801. La información incluye un valor del modo de predicción para la unidad de codificación y los indicadores MPM para cualquier unidad de predicción de la unidad de codificación. El detalle de cómo se hace esto se ha descrito anteriormente en la figura 11 en la etapa 1101 de codificar el valor del indicador de división, la etapa 1102 de nodo hoja, la etapa 1103 de codificar el valor del modo de predicción de la unidad de codificación, la etapa 1105 de codificar el valor del indicador de MPM de la unidad de predicción y la etapa 1106 de si hay más nodos.

A continuación, una etapa 1202 de codificar datos codificados por derivación codifica el segundo bloque 802 de datos codificados por derivación. El segundo bloque 802 de datos codificados por derivación codifica información acerca de los modos de intrapredicción utilizados para cada una de las unidades de codificación intrapredichas de la unidad de codificación más grande (LCU). La etapa 1202 de codificar datos codificados por derivación se describe con más detalle en la etapa 1106 de codificar datos de derivación y la etapa 1107 de si hay más nodos de la figura 11, descritas anteriormente.

El procedimiento 1200 continúa después hacia una etapa 1203 de codificar datos residuales, en la que los datos residuales son codificados en el tercer bloque de datos 803. Tal como se describió anteriormente, el tercer bloque de datos 803 contiene datos que están codificados aritméticamente y por derivación. La etapa 1203 de codificar datos residuales se describe con más detalle en la etapa 1108 de codificar datos residuales de la figura 11 anterior.

Una etapa 1204 de almacenar bloques de datos almacena datos codificados aritméticamente en el bloque de datos 801, datos codificados por derivación en el bloque de datos 802 y una mezcla de datos codificados aritméticamente y codificados por derivación en el bloque de datos 803 en el flujo de bits codificado 113. La etapa 1204 de almacenar bloques de datos puede ser implementada como una sola etapa para almacenar los bloques de datos, o como almacenamiento temporal de datos codificados a medida que los bloques de datos son generados mediante sus respectivas etapas en el procedimiento 1200.

Aunque el procedimiento 1200 se ha descrito en relación con el procedimiento 1100 de la figura 11, el procedimiento también puede abarcar otros procedimientos de codificación relacionados con la descodificación, tal como el procedimiento 1400 de la figura 14 y el procedimiento 1700 de la figura 17 que se describirán a continuación.

Un flujo de bits 1300 alternativo para codificar la unidad de codificación más grande (LCU) 600 se describirá con referencia a la figura 13. La figura 13 adopta la convención de la figura 7 para representar elementos de sintaxis binarizados codificados. El flujo de bits 1300 representa una porción del flujo de bits codificado 113 que codifica la unidad de codificación más grande (LCU) 600. Un primer bloque de datos 1301 tiene una estructura similar al primer bloque de datos 801, y codifica elementos de sintaxis utilizando exclusivamente codificación aritmética. El primer bloque de datos 1301 es similar al primer bloque de datos 801, ya que el primer bloque de datos 1301 codifica aritméticamente un valor del modo de predicción para una unidad de codificación utilizando un modo de predicción, tal como el modo de predicción 1308. A diferencia del primer bloque de datos 801, el primer bloque de datos 1301 no codifica un indicador de modo más probable, tal como el indicador de modo más probable 809 del primer bloque de datos 801. Por el contrario, el indicador de modo más probable 1309 es codificado en un segundo bloque de datos 1302 utilizando codificación por derivación. El segundo bloque de datos 1302 utiliza codificación por derivación exclusivamente para

codificar elementos de sintaxis, tal como se ha descrito para el segundo bloque de datos 802. La codificación del indicador de modo más probable con codificación por derivación puede permitir la descodificación con un mayor rendimiento, al leer grupos más grandes de bins de derivación en una sola operación de lectura. De manera similar a los datos de derivación 810, cuando un modo de predicción 1308 indica la utilización de la intrapredicción, el flujo de bits 1300 incluye los datos de derivación 1310 que representan un índice de modo más probable o un modo restante.

A continuación, se describirá una implementación alternativa en relación con el procedimiento 1400 de la figura 14, para descodificar el flujo de bits alternativo 1300. Una etapa 1401 de determinar el valor del indicador de división, una etapa 1402 de nodo hoja, una etapa 1403 de determinar el valor del modo de predicción de la unidad de codificación y una etapa 1404 de si hay más nodos operan de manera similar a la etapa 901 de determinar el valor del indicador de división, la etapa 902 de nodo hoja, la etapa 903 de determinar el valor del modo de predicción de la unidad de codificación, y la etapa 905 de si hay más nodos de la figura 9. En contraste con el procedimiento 900, una etapa correspondiente a la etapa 904 de determinar el valor del indicador de MPM de la unidad de predicción de la figura 9 no se incluye en el conjunto anterior de etapas de la figura 14. Por el contrario, la etapa correspondiente, que es la etapa 1405, tiene lugar más adelante en el procedimiento 1400. La etapa 1405 de determinar el valor del indicador de MPM de la unidad de predicción determina el valor del indicador de MPM de la unidad de predicción de manera similar a la correspondiente etapa 904 de la figura 9, excepto por que un indicador 1309 de modo más probable codificado por derivación es descodificado a partir del flujo de bits 1300. Una etapa 1406 de determinar el modo de intrapredicción, una etapa 1407 de si hay más nodos y una etapa 1408 de descodificar los datos residuales funcionan tal como se ha descrito con referencia a la etapa 906 de determinar el modo de intrapredicción, la etapa 907 de si hay más nodos y la etapa 908 de descodificar los datos residuales de la figura 9.

A continuación, se describirá una unidad de codificación más grande (LCU) 1500 a modo de ejemplo, de la figura 15. La unidad de codificación más grande (LCU) 1500 tiene una composición idéntica de las unidades de codificación 1 a 10 como la unidad de codificación más grande (LCU) 600 de la figura 6. Sin embargo, a diferencia de la unidad de codificación más grande (LCU) 600, la unidad de codificación más grande (LCU) 1500 incluye un límite de fragmento entre una unidad de codificación 9 1503 y una unidad de codificación 10 1505, ya que se han habilitado fragmentos de granularidad fina. Por consiguiente, las unidades de codificación 1 a 9 de la figura 15 están situadas en un primer fragmento, mientras que la unidad de codificación 10 1505 está situada en un segundo fragmento.

A continuación, se describirá un flujo de bits 1600, que se muestra en la figura 16, que codifica la unidad de codificación más grande (LCU) 1500. El flujo de bits 1600 está codificado con fragmentos de granularidad fina habilitados y el umbral del fragmento de granularidad fina configurado para limitar los límites del fragmento a los límites de la unidad de codificación de 32x32. Cuando se habilitan fragmentos de granularidad fina, la unidad de codificación más grande (LCU) 1500 se puede dividir en fragmentos independientes en cualquier zona de un tamaño igual o superior al umbral del fragmento de granularidad fina. Un elemento de sintaxis de final de fragmento indica la terminación de un fragmento. El elemento de sintaxis de final de fragmento es codificado después de la última unidad de codificación en cada zona cuyo tamaño es igual al umbral del fragmento de granularidad fina. En la figura 16 hay cuatro elementos de sintaxis de final de fragmento, puesto que la unidad de codificación más grande (LCU) de 64x64 tiene un tamaño límite de 32x32. El elemento de indicador de sintaxis de final de fragmento estará situado después de las unidades de codificación 1, 8, 9 y 10. Es un requisito que los elementos de sintaxis en un fragmento describan completamente las unidades de codificación en ese fragmento. Cuando se habilitan fragmentos de granularidad fina, la decisión de dividir la unidad de codificación más grande (LCU) 1500 en dos fragmentos se puede tomar durante el procedimiento de codificación del flujo de bits 1600. Por consiguiente, cuando la información procedente de una serie de unidades de codificación es agrupada en un primero, segundo y tercer bloques de datos, las unidades de codificación del grupo pueden no extenderse más allá de un indicador de final. Un componente del flujo de bits 1601 comprende elementos de sintaxis para la unidad de codificación 1. Un componente del flujo de bits 1602 comprende un primer bloque de datos 1615, un segundo bloque de datos 1616 y un tercer bloque de datos 1607, que codifica las unidades de codificación 2 a 8 de la figura 15 contenidas en la zona 607. El primer bloque de datos 1615, el segundo bloque de datos 1616 y el tercer bloque de datos 1607 que codifican las unidades de codificación 2 a 8 de la figura 15 son similares al primer bloque de datos 801, al segundo bloque de datos 802 y el tercer bloque de datos 803 de la figura 8. En contraste con el flujo de bits 800, la agrupación de elementos de sintaxis en el primer, segundo y tercer bloque de datos en el flujo de bits 1600 está limitada al umbral de fragmento de granularidad fina. Dado que el umbral de fragmento de granularidad fina se establece en 32x32, las unidades de codificación 1, 9 y 10 no se agrupan con otras unidades de codificación, mientras que las unidades de codificación 2 a 8 están agrupadas. Un indicador 1614 de final de fragmento indica que el primer fragmento termina después de la unidad de codificación 9 1503, y el segundo fragmento comienza en la unidad de codificación 10 1505 de la figura 15.

En una implementación, el flujo de bits 1600 codifica un indicador de habilitación de fragmentos de granularidad fina al inicio de cada unidad de codificación más grande (LCU), tal como la unidad de codificación más grande (LCU) 1500. Cuando los fragmentos de granularidad fina no están habilitados para una unidad de codificación más grande (LCU), el procedimiento 900 se aplica a la unidad de codificación más grande (LCU). Cuando los fragmentos de granularidad fina están habilitados para una unidad de codificación más grande (LCU), el procedimiento 900 se aplica a cada unidad de codificación subdividida de igual tamaño que el umbral del fragmento de granularidad fina.

A continuación, se describirá otra implementación alternativa en relación con el procedimiento 1700 de la figura 17, para

descodificar el flujo de bits 800. Una etapa 1701 de determinar el valor del indicador de división, una etapa 1702 de nodo hoja, una etapa 1703 de determinar el valor del modo de predicción de la unidad de codificación, una etapa 1704 de determinar el valor del indicador de MPM de la unidad de predicción y una etapa 1705 de si hay más nodos funcionan de manera similar a las etapas correspondientes de la figura 9, que son la etapa 901 de determinar el valor del indicador de división, la etapa 902 de nodo hoja, la etapa 903 de determinar el valor del modo de predicción de la unidad de codificación, la etapa 904 de determinar el valor del indicador de MPM de la unidad de predicción, y la etapa 905 de si hay más nodos. La estructura de la unidad de codificación resultante y la información de modo más probable son utilizadas por una etapa 1706 de leer datos de derivación para leer el bloque de datos de derivación 802. La longitud del bloque de datos de derivación 802 está determinada por la estructura de la unidad de codificación y la información del modo más probable mediante la suma de las longitudes de los índices de modo más probable y de los modos restantes. La etapa 1706 de leer datos de derivación puede leer el bloque de datos de derivación 802 en una sola operación, o en múltiples operaciones, pero no está limitada a leer información para una unidad de predicción a la vez. La cantidad de datos a leer es la longitud del bloque de datos de derivación 802 que ya se ha determinado.

A continuación, una etapa 1707 de asignar modos de intrapredicción a la unidad de predicción divide los datos de derivación de la etapa 1706 de leer los datos de derivación y determina el modo de intrapredicción para cada unidad de predicción. Una etapa 1708 de descodificar los datos residuales funciona tal como se describe con referencia a la etapa 908 de descodificar los datos residuales de la figura 9.

Cuando el modo restante es codificado utilizando un código de longitud variable, tal como el código de cinco o seis bits descrito con referencia a la figura 9, la longitud de los datos codificados por derivación 802 no se puede determinar antes de la etapa 1706 de leer los datos de derivación. Por el contrario, se puede calcular una longitud mínima en base al conocimiento de la estructura de la unidad de codificación y de los valores de indicador de modo más probable y de los tamaños de la unidad de predicción. La longitud mínima puede ser leída del flujo de bits codificado 113 y analizada para determinar el modo de intrapredicción, por lo menos, de una de las unidades de predicción. El análisis sintáctico puede ser aplicado repetidamente hasta conocer la longitud de los datos de derivación. Se pueden realizar una o varias lecturas de datos de derivación consecutivas para leer la totalidad de los datos codificados por derivación 802 del flujo de bits codificado 113. Aunque el código de longitud variable para la técnica de datos de derivación de la implementación alternativa descrita anteriormente se describe en relación con el procedimiento 1700 de la figura 17, la técnica se puede aplicar durante otros procedimientos de descodificación descritos anteriormente, tal como el procedimiento 900 de la figura 9.

Cuando el indicador de modo más probable está codificado por derivación, una variación del procedimiento 1700 puede funcionar para eliminar la etapa 1704 de determinar el valor del indicador de MPM de la unidad de predicción e incorpora la funcionalidad de la etapa 1704 de determinar el valor del indicador de MPM de la unidad de predicción en la etapa 1706 de leer los datos de derivación.

Una implementación alternativa para consumir datos codificados por derivación de longitud desconocida, funciona para acceder al flujo de bits codificado 113 para determinar un segmento de datos, que contiene, por lo menos, algunos datos codificados por derivación. Sin embargo, a diferencia de los planteamientos descritos anteriormente, los datos no son consumidos del flujo de bits. Los índices de modo más probable y de modos restantes son descodificados del segmento de datos y se mantiene un total acumulado para una longitud de los datos descodificados. Una vez que todos los datos codificados por derivación son descodificados del segmento de datos, la longitud total acumulada es consumida a continuación del flujo de bits. El resultado es que el segmento de datos accede a los datos del flujo de bits más allá del segundo bloque de datos 802 codificado por derivación, pero los datos no se consumen y, por lo tanto, el tercer bloque de datos 803 aritmético y de derivación está disponible para la descodificación mediante la etapa de descodificar los datos residuales. Aunque la técnica de descodificación de código de longitud variable de la implementación alternativa descrita anteriormente se ha descrito en relación con el procedimiento 1700 de la figura 17, la técnica se puede aplicar durante otros procedimientos de descodificación descritos anteriormente, tales como el procedimiento 900 de la figura 9.

Los procedimientos 900, 1000, 1400 y 1700, cuando son aplicados al descodificador de video 200, permiten que las implementaciones realicen un aumento en el rendimiento del análisis de un flujo de bits codificado, tal como el flujo de bits codificado 113. Esto ocurre cuando se leen mayores cantidades de datos codificados por derivación en una sola operación, debido a la concatenación de datos codificados por derivación. El mayor rendimiento es más notable para las implementaciones de hardware en las que la lectura o la escritura de datos codificados por derivación se puede realizar en paralelo para aumentar el rendimiento del sistema. Se obtiene un beneficio similar para el codificador de video 100 cuando los procedimientos 1100, 1200 y los procedimientos 1400 y 1700, variados en consecuencia para realizar la codificación, son aplicados para generar un flujo de bits codificado.

El Apéndice A que sigue a esta descripción detallada representa modificaciones que se pueden hacer al modelo 5.0 de prueba de codificación de video de alta eficiencia (HEVC) (HM-5.0) para especificar el flujo de bits 800 de la figura 8 que puede ser descodificado por el procedimiento 900 de la figura 9 descrito anteriormente.

APLICABILIDAD INDUSTRIAL

Las disposiciones descritas son aplicables a las industrias de procesamiento de datos e informática y, en concreto, al procesamiento de señales digitales para la codificación y la descodificación de señales tales como señales de video.

APÉNDICE A

A continuación, se representan modificaciones que pueden ser realizadas al modelo 5.0 de prueba de codificación de video de alta eficiencia (HEVC) (HM-5.0) para especificar el flujo de bits 800 de la figura 8, que puede ser descodificado mediante el procedimiento 900 de la figura 9 descrito anteriormente.

SINTAXIS DEL ÁRBOL DE CODIFICACIÓN

coding_tree(x0, y0, log2CUSize, cuDepth) {	**Descriptor**
if (slice_type == I &&SliceGranularity == 0) {	
if (cuDepth == 0) {	
coding_tree_split_and_mpm_flags(x0, y0, log2CUSize)	
coding_tree_luma_intra_mode(x0, y0, log2CUSize)	
}	
} else {	
if(x0 + (1 << log2CUSize) <= $PicWidthInSamples_L$ && y0 + (1 << log2CUSize) <= $PicHeightInSamples_L$ && cuAddress(x0, y0) >= SliceAddress && log2CUSize > Log2MinCUSize) {	
split_coding_unit_flag[x0 **][** y0 **]**	ae(v)
}	
}	
if(adaptive loop filter flag && alf_cu_control_flag) {	
if(cuDepth <= alf_cu_control_max_depth)	
if(cuDepth == alf_cu_control_max_depth ||	
split_coding_unit_flag[x0][y0] == 0)	
AlfCuFlagIdx++	
}	
if(cu_qp_delta_enabled_flag && log2CUSize >= log2MinCUDQPSize)	
IsCuQpDeltaCoded = 0	
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
if(cuAddress(x1, y0) > SliceAddress)	
moreDataFlag = coding_tree(x0, y0, log2CUSize - 1, cuDepth + 1)	
if(cuAddress(x0, y1) > SliceAddress && moreDataFlag && x1 < $PicWidthInSamples_L$)	
moreDataFlag = coding_tree(x1, y0, log2CUSize - 1, cuDepth + 1)	
if(cuAddress(x1, y1) > SliceAddress && moreDataFlag && y1 < $PicHeightInSamples_L$)	
moreDataFlag = coding_tree(x0, y1, log2CUSize - 1, cuDepth + 1)	
if(moreDataFlag && x1 < $PicWidthInSamples_L$ && y1 < $PicHeightInSamples_L$)	
moreDataFlag = coding_tree(x1, y1, log2CUSize - 1, cuDepth + 1)	
} else {	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	

(continuación)

AlfCuFlag[x0][y0] = alf_cu_flag[AlfCuFlagIdx]	
coding_unit(x0, y0, log2CUSize)	
if((slice_type != I || SliceGranularity != 0)&& granularity_block_boundary(x0, y0, log2CUSize)) {	
end_of_slice_flag	ae(v)
moreDataFlag = !end_of_slice_flag	
} else	
moreDataFlag = 1	
}	
return moreDataFlag	
}	

Sintaxis del árbol de codificación para indicadores de división y de MPM

coding_tree_split_and_mpm_flags(x0, y0, log2CUSize) {	**Descriptor**
if(x0 + (1 << log2CUSize) <= $PicWidthInSamples_L$ && y0 + (1 << log2CUSize) <= $PicHeightInSamples_L$ && cuAddress(x0, y0) >= SliceAddress&& log2CUSize > Log2MinCUSize) {	
split_coding_unit_flag[x0 **][** y0 **]**	ae(v)
}	
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 =y0 + ((1 << log2CUSize) >> 1)	
if(cuAddress(x1, y0) > SliceAddress)	
coding_tree_split_and_mpm_flags (x0, y0, log2CUSize - 1)	
if(cuAddress(x0, y1) >SliceAddress && x1 < $PicWidthInSamples_L$)	
coding_tree_split_and_mpm_flags (x1, y0, log2CUSize - 1)	
if(cuAddress(x1, y1) > SliceAddress && y1 < $PicHeightInSamples_L$)	
coding_tree_split_and_mpm_flags (x0, y1, log2CUSize - 1)	
if(x1 < $PicWidthInSamples_L$ && y1 < $PicHeightInSamples_L$)	
coding_tree_split_and_mpm_flags (x1, y1, log2CUSize - 1)	
} else {	
coding_unit_part_mode_and_mpm_flag (x0, y0, log2CUSize)	
}	
}	

Sintaxis del árbol de codificación para el modo intra luma

coding_tree_luma_intra_mode(x0, y0, log2CUSize) {	**Descriptor**
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
if(cuAddress(x1, y0) > SliceAddress)	
coding_tree_luma_intra_mode(x0, y0, log2CUSize - 1)	
if(cuAddress(x0, y1) >SliceAddress && x1 < $PicWidthInSamples_L$)	
coding_tree_luma_intra_mode (x1, y0, log2CUSize - 1)	
if(cuAddress(x1, y1) > SliceAddress && y1 < $PicHeightInSamples_L$)	
coding_tree_luma_intra_mode (x0, y1, log2CUSize - 1)	
if(x1 < $PicWidthInSamples_L$ && y1 < $PicHeightInSamples_L$)	
coding_tree_luma_intra_mode (x1, y1, log2CUSize - 1)	
} else {	
coding_unit_luma_intra_mode(x0, y0, log2CUSize)	
}	
}	

SINTAXIS DE LA UNIDAD DE CODIFICACIÓN

coding_unit(x0, y0, log2CUSize) {	**Descriptor**
if(slice_type != I)	
skip_flag[x0 **][** y0 **]**	ae(v)
if(skip_flag[x0][y0])	
prediction_unit(x0, y0 , log2CUSize)	
else {	
if (slice_type != I || SliceGranularity != 0) {	
pred_type	ae(v)
}	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
x2 = x1 - ((1 << log2CUSize) >> 2)	
y2 = y1 - ((1 << log2CUSize) >> 2)	
x3 = x1 + ((1 << log2CUSize) >> 2)	
y3 =y1 + ((1 << log2CUSize) >> 2)	
if(PartMode == PART_2Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0 , log2CUSize)	

(continuación)

prediction_unit(x0, y1 , log2CUSize)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
} else if(PartMode == PART_2NxnU) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y2 , log2CUSize)	
} else if(PartMode == PART_2NxnD) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y3 , log2CUSize)	
} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x2, y0 , log2CUSize)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x3, y0 , log2CUSize)	
} else { /* PART_NxN */	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
prediction_unit(x1, y1 , log2CUSize)	
}	
if(!pcm_flag) {	
transform_tree(x0, y0, log2CUSize, log2CUSize, 0, 0)	
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 0)	
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 1)	
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 2)	
}	
}	
}	

Sintaxis de la unidad de codificación para el modo de partición e indicador de MPM

coding_unit_part_mode_and_mpm_flag (x0, y0, log2CUSize) {	**Descriptor**
if (log2CUSize == Log2MinCUSize) {	
pred_type	ae(v)
}	
x1 = x0 + ((1 << log2CUSize) >> 1)	

(continuación)

y1 = y0 + ((1 << log2CUSize) >> 1)	
x2 = x1 - ((1 << log2CUSize) >> 2)	
y2 = y1 - ((1 << log2CUSize) >> 2)	
x3 = x1 + ((1 << log2CUSize) >> 2)	
y3 =y1 + ((1 << log2CUSize) >> 2)	
if(PartMode == PART_2Nx2N) {	
prediction_unit_mpm_flag(x0, y0 , log2CUSize)	
} else { /* PART NxN */	
prediction_unit_mpm_flag(x0, y0 , log2CUSize)	
prediction_unit_mpm_flag(x1, y0 , log2CUSize)	
prediction_unit_mpm_flag(x0, y1 , log2CUSize)	
prediction_unit_mpm_flag(x1, y1 , log2CUSize)	
}	
}	

Sintaxis de la unidad de codificación para el modo intra luma

coding_unit_luma_intra_mode(x0, y0, log2CUSize) {	**Descriptor**
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
x2 = x1 - ((1 << log2CUSize) >> 2)	
y2 = y1 - ((1 << log2CUSize) >> 2)	
x3 = x1 + ((1 << log2CUSize) >> 2)	
y3 =y1 + ((1 << log2CUSize) >> 2)	
if(PartMode == PART_2Nx2N) {	
prediction_unit_luma_intra_mode(x0, y0 , log2CUSize)	
} else { /* PART NxN */	
prediction_unit_luma_intra_mode(x0, y0 , log2CUSize)	
prediction_unit_luma_intra_mode(x1, y0 , log2CUSize)	
prediction_unit_luma_intra_mode(x0, y1 , log2CUSize)	
prediction_unit_luma_intra_mode(x1, y1 , log2CUSize)	
}	
}	

SINTAXIS DE LA UNIDAD DE PREDICCIÓN

prediction_unit(x0, y0, log2CUSize) {	**Descriptor**
if(skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else if(PredMode = = MODE_INTRA) {	
if (SliceGranularity == 0) {	
if(!pcm_flag) {	
intra_chroma_pred_mode[x0][y0]	ae(v)
SignaledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[x0] [y0] == 3 : intra_chroma_pred_mode[x0][y0] == 2)	
}	
} else {	
if(PartMode == PART _2Nx2N && log2CUSize >= Log2MinIPCMCUSize)	
pcm_flag	ae(v)
if(pcm_flag) {	
while (!byte_aligned( ))	
pcm_alignment_zero_bit	u(v)
for(i = 0; i < 1 << (log2CUSize << 1); i++)	
pcm_sample_luma[i]	u(v)
for(i = 0; i < (1 << (log2CUSize << 1)) >> 1; i++)	
pcm_sample_chroma[i]	u(v)
} else {	
prev_intra_luma_pred_flag[x0 **][** y0 **]**	ae(v)
if(prev_intra_luma_pred_flag[x0][y0])	
mpm_idx[x0 **][** y0 **]**	ae(v)
else	
rem_intra_luma_pred_mode[x0 **][** y0 **]**	ae(v)
intra_chroma_pred_mode[x0][y0]	ae(v)
SignaledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[x0] [y0] == 3 : intra_chroma_pred_mode[x0][y0] == 2)	
}	
}	
} else {/* MODE_INTER */	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0 **][** y0 **]**	ae(v)
} else {	
if(slice_type = = B)	
inter_pred_flag[x0 **][** y0 **]**	ae(v)

(continuación)

if(inter_pred_flag[x0][y0] = = Pred_LC) {	
if(num_ref_idx_lc_active_minus1 > 0)	
ref_idx_lc[x0 **][** y0 **]**	ae(v)
mvd_coding(mvd_lc[x0][y0][0], mvd_lc[x0][y0][1])	
mvp_idx_lc[x0] [y0]	ae(v)
}	
else {/* Pred_L0 or Pred_BI */	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(mvd_10[x0][y0][0], mvd_l0[x0][y0][1])	
mvp_idx_l0[x0] [y0]	ae(v)
}	
if(inter_pred_flag[x0][y0] = = Pred_BI) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0 **][** y0 **]**	ae(v)
mvd_coding(mvd_l1[x0][y0][0], mvd_l1[x0][y0][1])	
mvp_idx_l1[x0][y0]	ae(v)
}	
}	
}	
}	

Sintaxis de la unidad de predicción para el indicador de MPM

prediction_unit_mpm__flag(x0, y0, log2CUSize) {	**Descriptor**
if(PredMode = = MODE_INTRA) {	
if(PartMode == PART _2Nx2N && log2CUSize >= Log2MinIPCMCUSize)	
pcm_flag	ae(v)
if(pcm_flag) {	
while (!byte_aligned( ))	
pcm_alignment_zero_bit	u(v)
for(i = 0; i < 1 << (log2CUSize << 1); i++)	
pcm_sample_luma[i]	u(v)
for(i = 0; i < (1 << (log2CUSize << 1)) >> 1; i++)	
pcm_sample_chroma[i]	u(v)
} else {	
prev_intra_luma_pred_flag[x0 **][** y0 **]**	ae(v)
}	
}	
}	

Sintaxis de la unidad de predicción para el modo intra luma

prediction_unit_luma_intra_mode(x0, y0, log2CUSize) {	**Descriptor**
if(PredMode = = MODE INTRA) {	
if(!pcm_flag[x0][y0]) {	
if(prev_intra_luma_pred_flag[x0][y0])	
mpm_idx[x0][y0]	ae(v)
else	
rem_intra_luma_pred_mode[x0][y0]	ae(v)
}	
}	

La siguiente sintaxis del árbol de codificación muestra la asignación de elementos sintácticos a bloques de datos según tres categorías etiquetadas con 1, 2 y 3.

SINTAXIS DEL ÁRBOL DE CODIFICACIÓN

coding_tree(x0, y0, log2CUSize, cuDepth) {	**Categoría**	**Descriptor**
if(x0 + (1 << log2CUSize) <= $PicWidthInSamples_L$ && y0 + (1 << log2CUSize) <= $PicHeightInSamples_L$ && cuAddress(x0, y0) >= SliceAddress && log2CUSize > Log2MinCUSize) {		
split _coding _unit flag[x0][y0]	1	ae(v)
}		
if(adaptive loop filter flag && alf cu_control_flag) {		
if(cuDepth <= alf_cu_control_max_depth)		
if(cuDepth == alf_cu_control_max_depth ||		
split_coding_unit_flag[x0][y0] == 0)		
AlfCuFlagIdx++		
}		
if(cu_qp_delta_enabled_flag && log2CUSize >= log2MinCUDQPSize)		
IsCuQpDeltaCoded = 0		
if(split_coding_unit_flag[x0][y0]) {		
x1 = x0 + ((1 << log2CUSize) >> 1)		
y1 = y0 + ((1 << log2CUSize) >> 1)		
if(cuAddress(x1, y0) > SliceAddress)		
moreDataFlag = coding_tree(x0, y0, log2CUSize - 1, cuDepth + 1)		

(continuación)

if(cuAddress(x0, y1) > SliceAddress && moreDataFlag && x1 < PicWidthInSamples$_L$)		
moreDataFlag = coding_tree(x1, y0, log2CUSize - 1, cuDepth + 1)		
if(cuAddress(x1, y1) > SliceAddress && moreDataFlag && y1 < PicHeightInSamples$_L$)		
moreDataFlag = coding_tree(x0, y1, log2CUSize - 1, cuDepth + 1)		
if(moreDataFlag && x1 < PicWidthInSamples$_L$ && y1 < PicHeightInSamples$_L$)		
moreDataFlag = coding_tree(x1, y1, log2CUSize - 1, cuDepth + 1)		
} else {		
if(adaptive_loop_filter_flag && alf_cu_control_flag)		
AlfCuFlag[x0][y0] = alf_cu_flag[AlfCuFlagIdx]		
coding_unit(x0, y0, log2CUSize)		
if(granularity_block_boundary(x0, y0, log2CUSize)) {		
end of slice_flag	3	ae(v)
moreDataFlag = !end_of_slice_flag		
} else		
moreDataFlag = 1		
}		
return moreDataFlag		
}		

SINTAXIS DE LA UNIDAD DE CODIFICACIÓN

coding_unit(x0, y0, log2CUSize) {	**Categoría**	**Descriptor**
if(slice_type != I)		
skip_flag[x0][y0]	n/a	ae(v)
if(skip flag[x0][y0])		
prediction_unit(x0, y0 , log2CUSize)		
else if(slice_type != I || log2CUSize = = Log2MinCUSize) {		
pred_type	1	ae(v)
x1 = x0 + ((1 << log2CUSize) >> 1)		
y1 = y0 + ((1 << log2CUSize) >> 1)		
x2 = x1 - ((1 << log2CUSize) >> 2)		
y2 = y1 - ((1 << log2CUSize) >> 2)		
x3 = x1 + ((1 << log2CUSize) >> 2)		
y3 = y1 + ((1 << log2CUSize) >> 2)		
if(PartMode == PART_2Nx2N) {		
prediction_unit(x0, y0 , log2CUSize)		
} else if(PartMode == PART_2NxN) {		
prediction unit(x0, y0 , log2CUSize)		

(continuación)

prediction unit(x0, y1 , log2CUSize)		
} else if(PartMode == PART_N×2N) {		
prediction unit(x0, y0 , log2CUSize)		
prediction_unit(x1, y0 , log2CUSize)		
} else if(PartMode == PART_2NxnU) {		
prediction unit(x0, y0 , log2CUSize)		
prediction unit(x0, y2 , log2CUSize)		
} else if(PartMode == PART_2NxnD) {		
prediction unit(x0, y0 , log2CUSize)		
prediction unit(x0, y3 , log2CUSize)		
} else if(PartMode == PART_nLx2N) {		
prediction unit(x0, y0 , log2CUSize)		
prediction unit(x2, y0 , log2CUSize)		
} else if(PartMode == PART_nRx2N) {		
prediction unit(x0, y0 , log2CUSize)		
prediction unit(x3, y0 , log2CUSize)		
} else { /* PART NxN */		
prediction unit(x0, y0 , log2CUSize)		
prediction_unit(x1, y0 , log2CUSize)		
prediction unit(x0, y1 , log2CUSize)		
prediction_unit(x1, y1 , log2CUSize)		
}		
if(!pcm_flag) {		
transform_tree(x0, y0, log2CUSize, log2CUSize, 0, 0)		
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 0)		
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 1)		
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 2)		
}		
}		
}		

SINTAXIS DE LA UNIDAD DE PREDICCIÓN

prediction unit(x0, y0, log2CUSize) {	**Categoría**	**Descriptor**
if(skip_flag[x0][y0]) {		
if(MaxNumMergeCand > 1)		
merge_idx[x0][y0]	n/a	ae(v)
} else if(PredMode = = MODE INTRA) {		

(continuación)

if(PartMode == PART_2Nx2N && log2CUSize >= Log2MinIPCMCUSize)		
pcm flag	n/a	ae(v)
if(pcm_flag) {		
while (!byte_aligned())		
pcm_alignment_zero_bit	n/a	u(v)
for(i = 0; i < 1 « (log2CUSize << 1); i++)		
pcm_sample_luma[i]	n/a	u(v)
for(i = 0; i < (1 << (log2CUSize << 1)) >> 1; i++)		
pcm_sample_chroma[i]	n/a	u(v)
} else {		
prev_intra_luma_pred_flag[x0][y0]	1 (2 if EP)	ae(v)
if(prev_intra_luma_pred_flag[x0][y0])		
mpm_idx[x0][y0]	2	ae(v)
else		
rem_intra_luma_pred_mode[x0][y0]	2	ae(v)
intra_chroma_pred_mode[x0][y0]	3	ae(v)
SignaledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[x0][y0] == 3 : intra_chroma_pred_ mode[x0][y0] == 2)		
}		
} else { /* MODE INTER */		
merge_flag[x0][y0]	n/a	ae(v)
if(merge_flag[x0][y0]) {		
if(MaxNumMergeCand > 1)		
merge_idx[x0][y0]	n/a	ae(v)
} else {		
if(slice_type = = B)		
inter_pred_flag[x0][y0]	n/a	ae(v)
if(inter_pred_flag[x0][y0] = = Pred_LC) {		
if(num_ref_idx_lc_active_minus1 > 0)		
ref_idx_lc[x0][y0]	n/a	ae(v)
mvd_coding(mvd_lc[x0][y0][0], mvd_lc[x0][y0][1])		
mvp_idx_lc[x0][y0]	n/a	ae(v)
}		
else {/* Pred_L0 or Pred_BI */		
if(num_ref_idx_10_active_minus1 > 0)		
ref_idx_10[x0][y0]	n/a	ae(v)
mvd_coding(mvd_l0[x0][y0][0], mvd_l0[x0][y0][1])		
mvp_idx_l0[x0][y0]	n/a	ae(v)

(continuación)

}		
if(inter_pred_flag[x0][y0] = = Pred_BI) {		
if(num_ref_idx_l1_active_minus1 > 0)		
ref_idx_l1[x0][y0]	n/a	ae(v)
mvd_coding(mvd_l1[x0][y0][0], mvd_l1[x0][y0][1])		
mvp_idx_l1[x0][y0]	n/a	ae(v)
}		
}		
}		
}		

(fin del Apéndice A)

REIVINDICACIONES

1. Procedimiento para codificar datos de imagen en un flujo de bits, comprendiendo el procedimiento:

codificar, en el flujo de bits, un indicador de división para indicar si una unidad de codificación está dividida en unidades de codificación más pequeñas, y datos para indicar información de un modo de predicción (808) que indica si se utiliza interpredicción o intrapredicción para una unidad de codificación objetivo y para indicar un modo de partición que indica si la unidad de codificación objetivo contiene una unidad de predicción o está dividida en una pluralidad de unidades de predicción (1101, 1103), en el que el modo de partición puede ser un modo de partición 2NxN en el que una unidad de codificación contiene dos unidades de predicción rectangulares; y
cuando la unidad de codificación objetivo no está dividida en unidades de codificación más pequeñas, se utiliza la intrapredicción para la unidad de codificación objetivo, y la unidad de codificación objetivo se divide en una pluralidad de unidades de predicción:
codificar aritméticamente los indicadores, siendo cada uno de los cuales para una unidad de predicción diferente de la pluralidad de unidades de predicción en la unidad de codificación objetivo, para formar un bloque de datos (801) de datos codificados adyacentes en el flujo de bits, indicando el indicador (809) si se utiliza un modo más probable o un modo restante para la intrapredicción de una unidad de predicción (1104, 1201);
codificar por derivación la información, siendo cada fragmento de la misma para una unidad de predicción diferente de la pluralidad de unidades de predicción en la unidad de codificación objetivo, para formar otro bloque de datos (802) de datos codificados adyacentes en el flujo de bits, indicando cada fragmento de la información un valor de índice del modo más probable o un valor de modo restante utilizado para codificar la unidad de predicción (1106, 1202); y
codificar la unidad de codificación objetivo en el flujo de bits utilizando los modos de intrapredicción de la pluralidad de unidades de predicción tal como indican los indicadores y la información (1204),
en el que, en el flujo de bits, el otro bloque de datos (802) de información codificada por derivación está situado posteriormente al bloque de datos (801) de indicadores codificados aritméticamente.

2. Procedimiento, según la reivindicación 1, en el que el valor de modo restante es para designar uno de los modos de intrapredicción restantes.

3. Procedimiento, según la reivindicación 2, en el que los modos de intrapredicción restantes son modos de intrapredicción distintos de los modos más probables.

4. Procedimiento, según cualquiera de las reivindicaciones anteriores, en el que el valor del índice del modo más probable designa uno de una pluralidad de los modos más probables.

5. Procedimiento, según la reivindicación 1, en el que los indicadores se codifican consecutivamente en el flujo de bits.

6. Procedimiento, según la reivindicación 1, en el que la información se codifica consecutivamente en el flujo de bits.

7. Aparato configurado para codificar datos de imagen en un flujo de bits, según el procedimiento de cualquiera de las reivindicaciones 1 a 6.

8. Programa que, cuando es ejecutado en un ordenador, hace que el ordenador realice el procedimiento de cualquiera de las reivindicaciones 1 a 6.

9. Procedimiento para descodificar datos de imagen de un flujo de bits, comprendiendo el procedimiento:

descodificar, del flujo de bits, un indicador de división (813) para indicar si una unidad de codificación está dividida en unidades de codificación más pequeñas (901, 1001); y
determinar, mediante la descodificación de los datos del flujo de bits, una información de modo de predicción (808) para indicar si se utiliza interpredicción o intrapredicción para una unidad de codificación objetivo y un modo de partición para indicar que la unidad de codificación objetivo contiene una unidad de predicción o está dividida en una pluralidad de unidades de predicción, en el que el modo de partición puede ser un modo de partición 2NxN en el que una unidad de codificación contiene dos unidades de predicción rectangulares; y
cuando la unidad de codificación objetivo no está dividida en unidades de codificación más pequeñas, se utiliza la intrapredicción para la unidad de codificación objetivo, y la unidad de codificación objetivo se divide en una pluralidad de unidades de predicción:
descodificar aritméticamente los indicadores codificados, siendo cada uno de los cuales para una diferente de la pluralidad de unidades de predicción en la unidad de codificación objetivo, a partir de un bloque de datos (801) de datos codificados adyacentes en el flujo de bits, indicando el indicador (809) si se utiliza un modo más probable o un modo restante para una intrapredicción de una unidad de predicción (904, 1001);
descodificar por derivación la información codificada, siendo cada fragmento de la misma para una diferente de la pluralidad de unidades de predicción en la unidad de codificación objetivo, de otro bloque de datos (802) de datos

codificados adyacentes en el flujo de bits, indicando cada fragmento de información un valor de índice del modo más probable o un valor de modo restante utilizado para codificar la unidad de predicción (906, 907, 1002); y
descodificar la unidad de codificación objetivo utilizando los modos de intrapredicción de la pluralidad de unidades de predicción tal como indican los indicadores y la información (1003, 1004),
en el que, en el flujo de bits, el bloque de datos (802) de información codificada por derivación está situado posteriormente al bloque de datos (801) de indicadores codificados aritméticamente.

10. Procedimiento, según la reivindicación 9, en el que el valor del modo restante es para designar uno de los modos de intrapredicción restantes.

11. Procedimiento, según la reivindicación 10, en el que los modos de intrapredicción restantes son modos de intrapredicción distintos de los modos más probables.

12. Procedimiento, según cualquiera de las reivindicaciones anteriores, en el que el valor del índice del modo más probable designa uno de una pluralidad de los modos más probables.

13. Procedimiento, según la reivindicación 9, en el que los indicadores se codifican consecutivamente en el flujo de bits.

14. Procedimiento, según la reivindicación 9, en el que la información se codifica consecutivamente en el flujo de bits.

15. Aparato configurado para descodificar datos de imagen de un flujo de bits, según el procedimiento de la reivindicación 9 a 14.

16. Programa que, cuando es ejecutado en un ordenador, hace que el ordenador realice el procedimiento de las reivindicaciones 9 a 14.

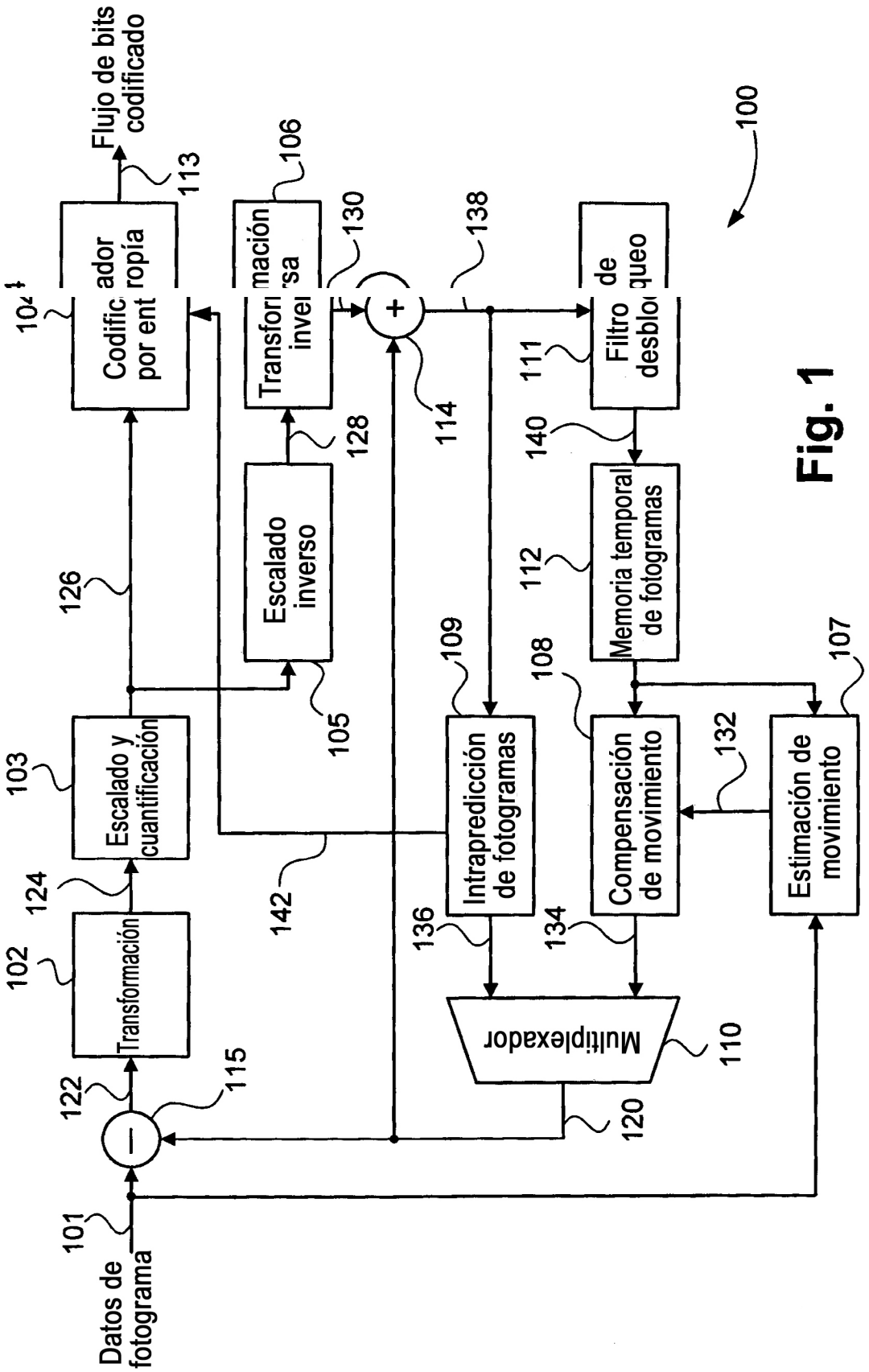
Datos de fotograma
101
Flujo de bits codificado
113
100
Codific ador por ent ropía
104
Transformación
102
Escalado y cuantificación
103
Escalado inverso
105
Transfori mación inve rsa
106
Filtro de desblo queo
111
Memoria temporal de fotogramas
112
Intrapredicción de fotogramas
109
Compensación de movimiento
108
Estimación de movimiento
107
Multiplexador
110
124
122
115
126
128
130
138
114
140
142
136
134
132
120

Fig. 1

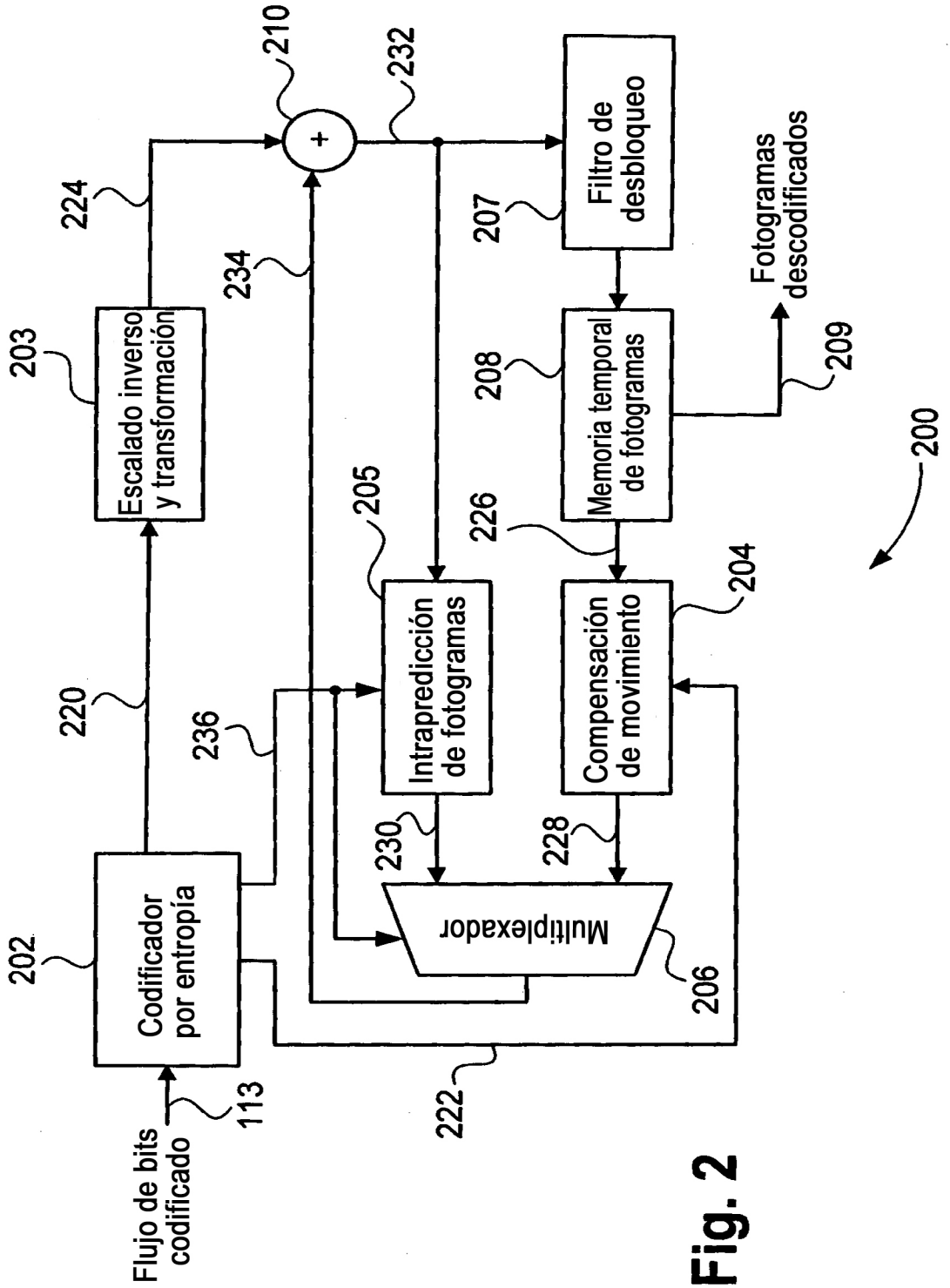
Flujo de bits codificado
113
Codificador por entropía
202
220
Escalado inverso y transformación
203
224
210
+
232
Filtro de desbloqueo
207
Fotogramas descodificados
209
Memoria temporal de fotogramas
208
234
205
Intrapredicción de fotogramas
236
226
Compensación de movimiento
204
230
228
Multiplexador
206
222
200

Fig. 2

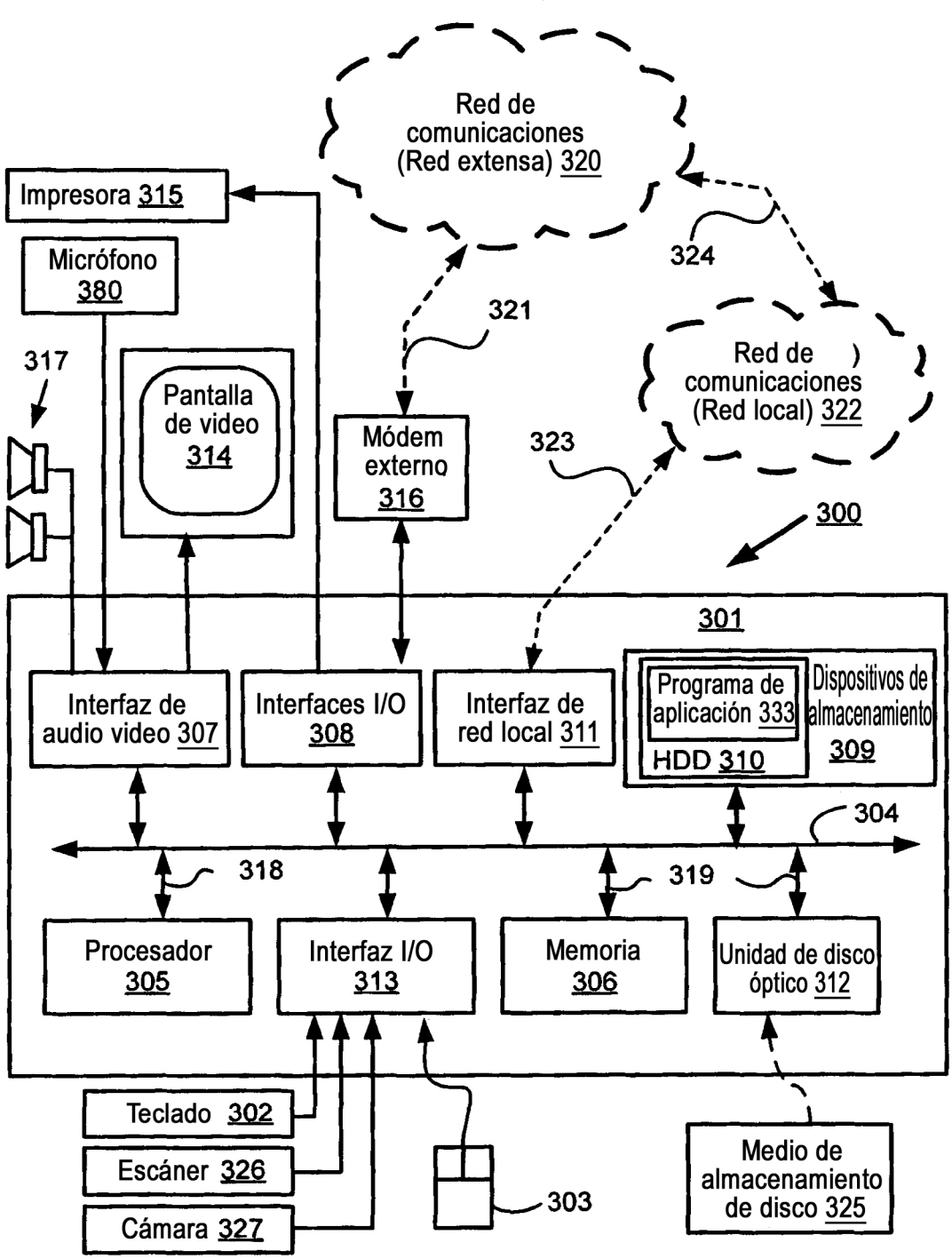
Red de comunicaciones (Red extensa) 320
Impresora 315
Micrófono 380
317
Pantalla de video 314
Módem externo 316
321
324
Red de comunicaciones (Red local) 322
323
300
301
Interfaz de audio video 307
Interfaces I/O 308
Interfaz de red local 311
Programa de aplicación 333
HDD 310
Dispositivos de almacenamiento 309
304
318
319
Procesador 305
Interfaz I/O 313
Memoria 306
Unidad de disco óptico 312
Teclado 302
Escáner 326
Cámara 327
303
Medio de almacenamiento de disco 325

Fig. 3A

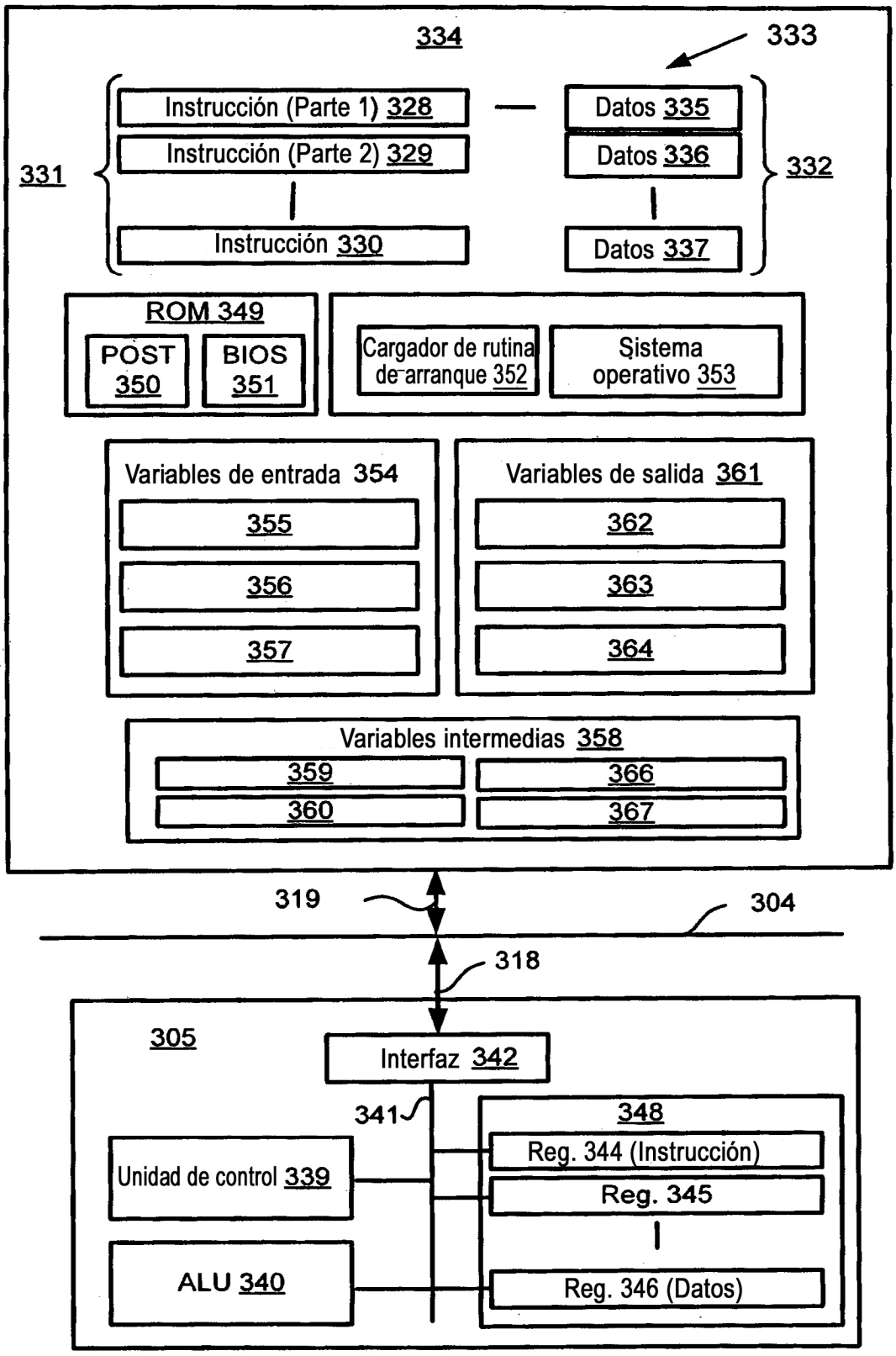
334
333
Instrucción (Parte 1) 328
Instrucción (Parte 2) 329
Instrucción 330
331
Datos 335
Datos 336
Datos 337
332
ROM 349
POST 350
BIOS 351
Cargador de rutina de arranque 352
Sistema operativo 353
Variables de entrada 354
355
356
357
Variables de salida 361
362
363
364
Variables intermedias 358
359
360
366
367
319
304
318
305
Interfaz 342
341
348
Unidad de control 339
Reg. 344 (Instrucción)
Reg. 345
ALU 340
Reg. 346 (Datos)

Fig. 3B

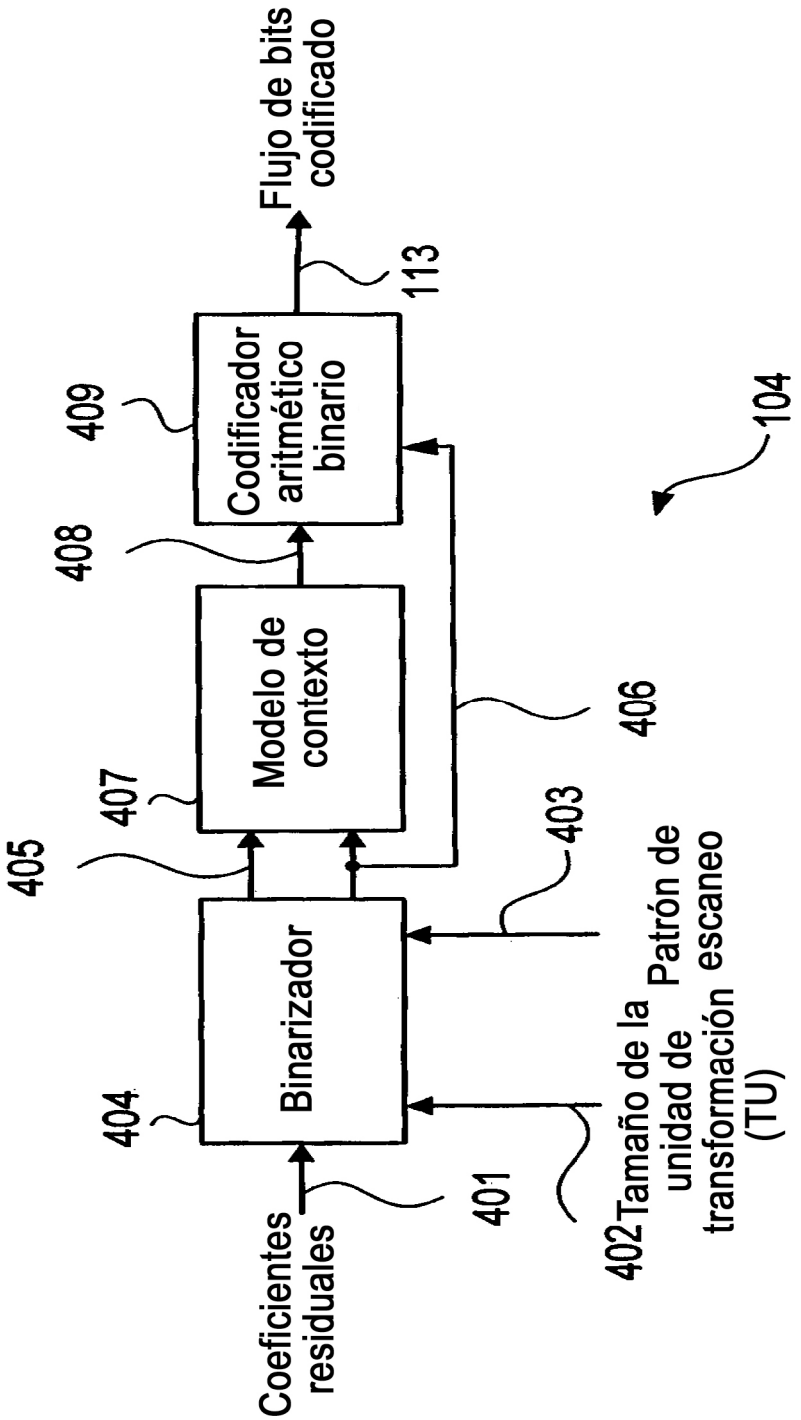
Flujo de bits codificado
113
Codificador aritmético binario
409
408
Modelo de contexto
407
405
Binarizador
404
Coeficientes residuales
401
Tamaño de la unidad de transformación (TU)
402
Patrón de escaneo
403
406
104
142

Fig. 4

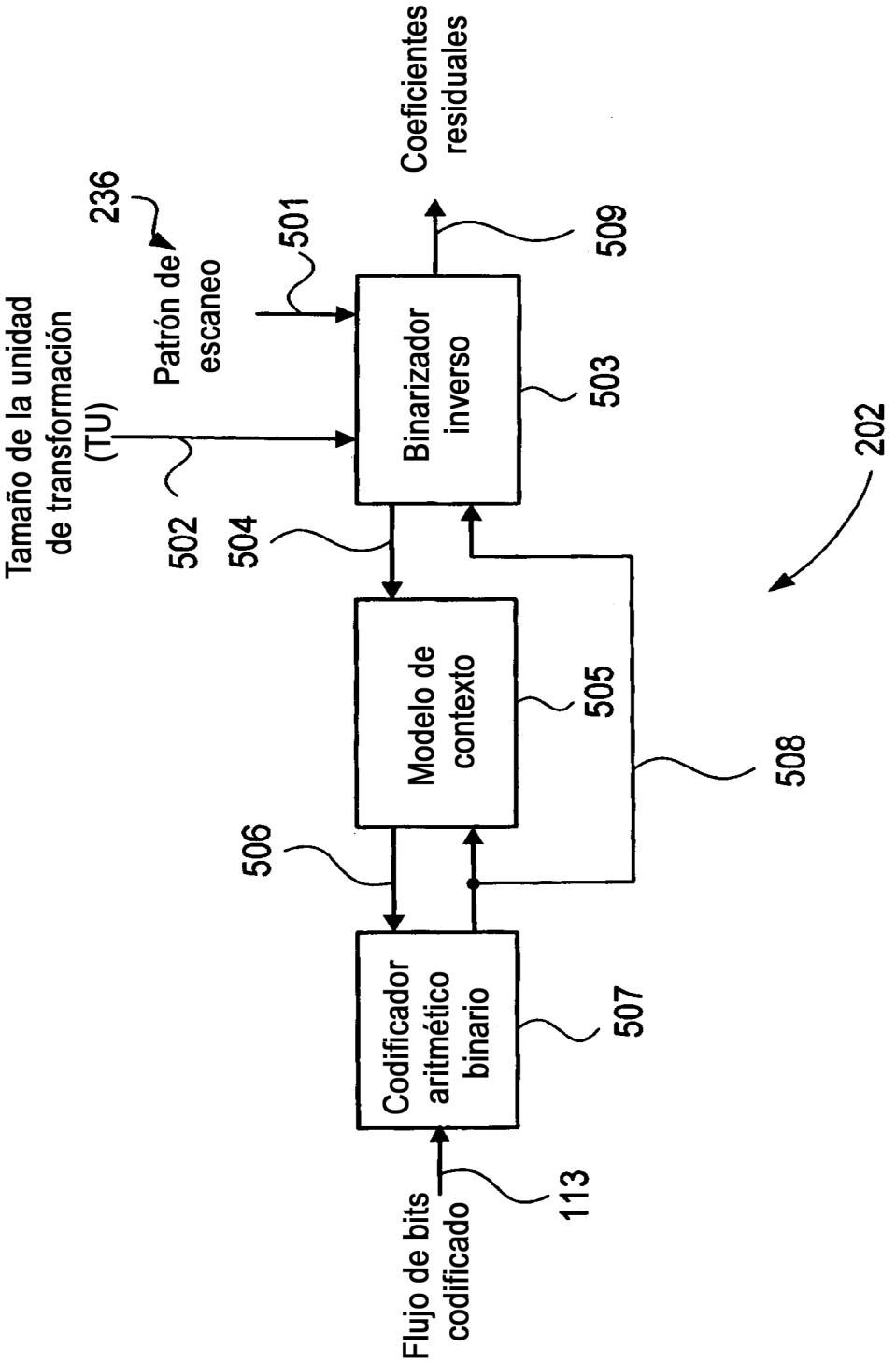
Tamaño de la unidad de transformación (TU)
Patrón de escaneo
236
501
Coeficientes residuales
509
Binarizador inverso
503
202
502
504
Modelo de contexto
505
508
506
Codificador aritmético binario
507
Flujo de bits codificado
113

Fig. 5

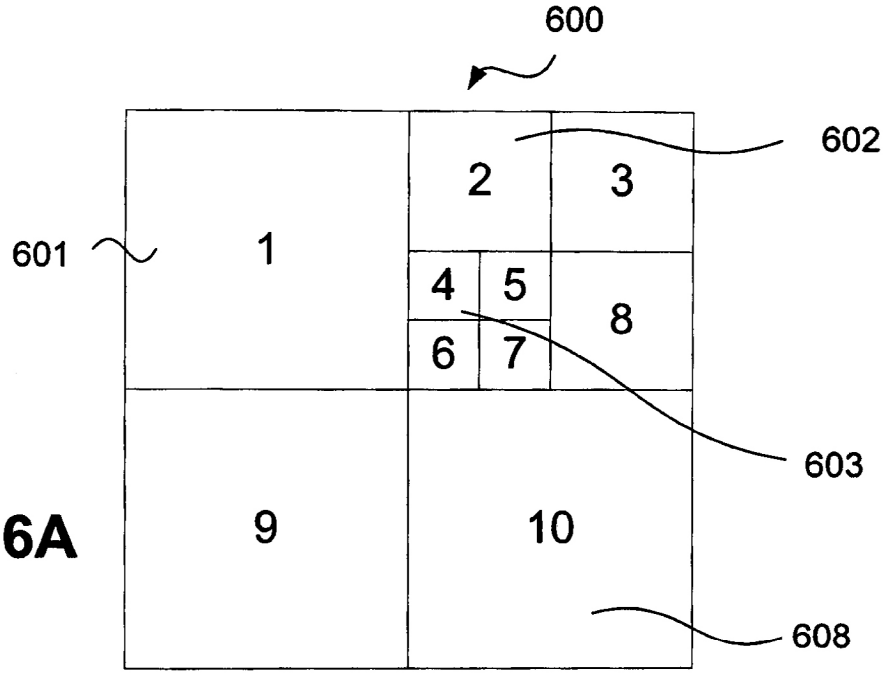
600
602
601
1
2
3
4
5
6
7
8
603
9
10
608

Fig. 6A

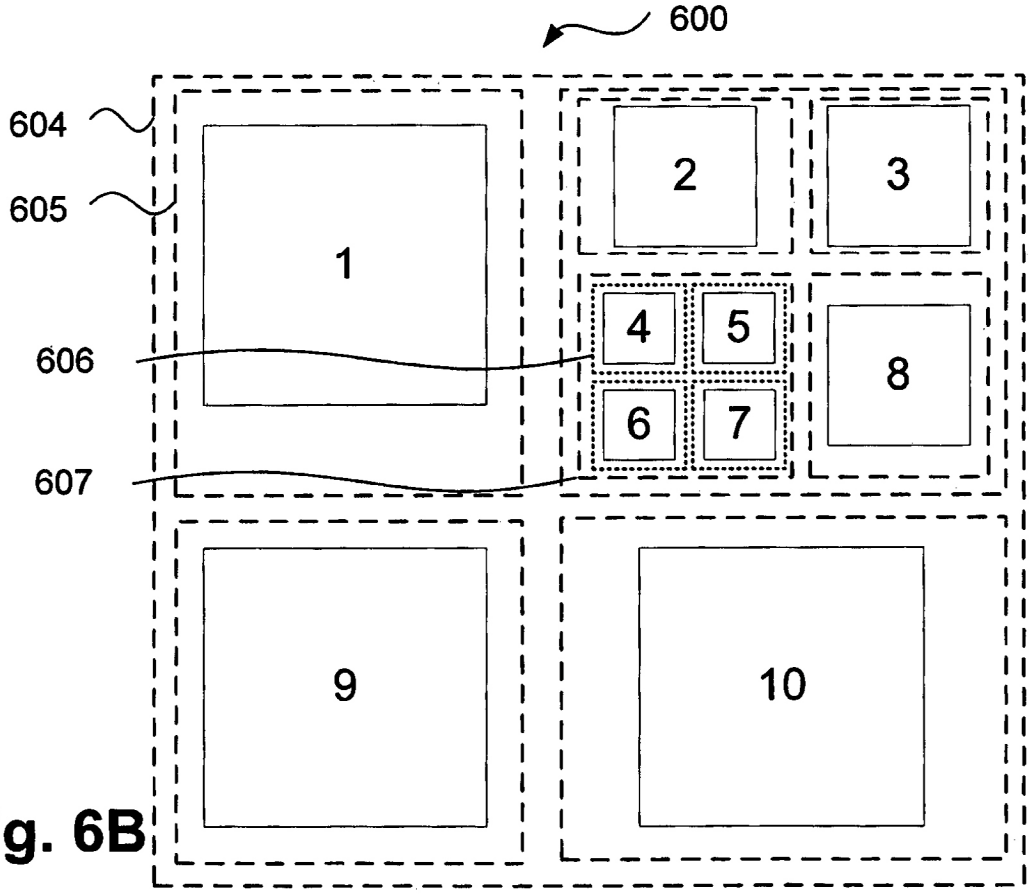
600
604
605
1
2
3
606
4
5
6
7
8
607
9
10

Fig. 6B

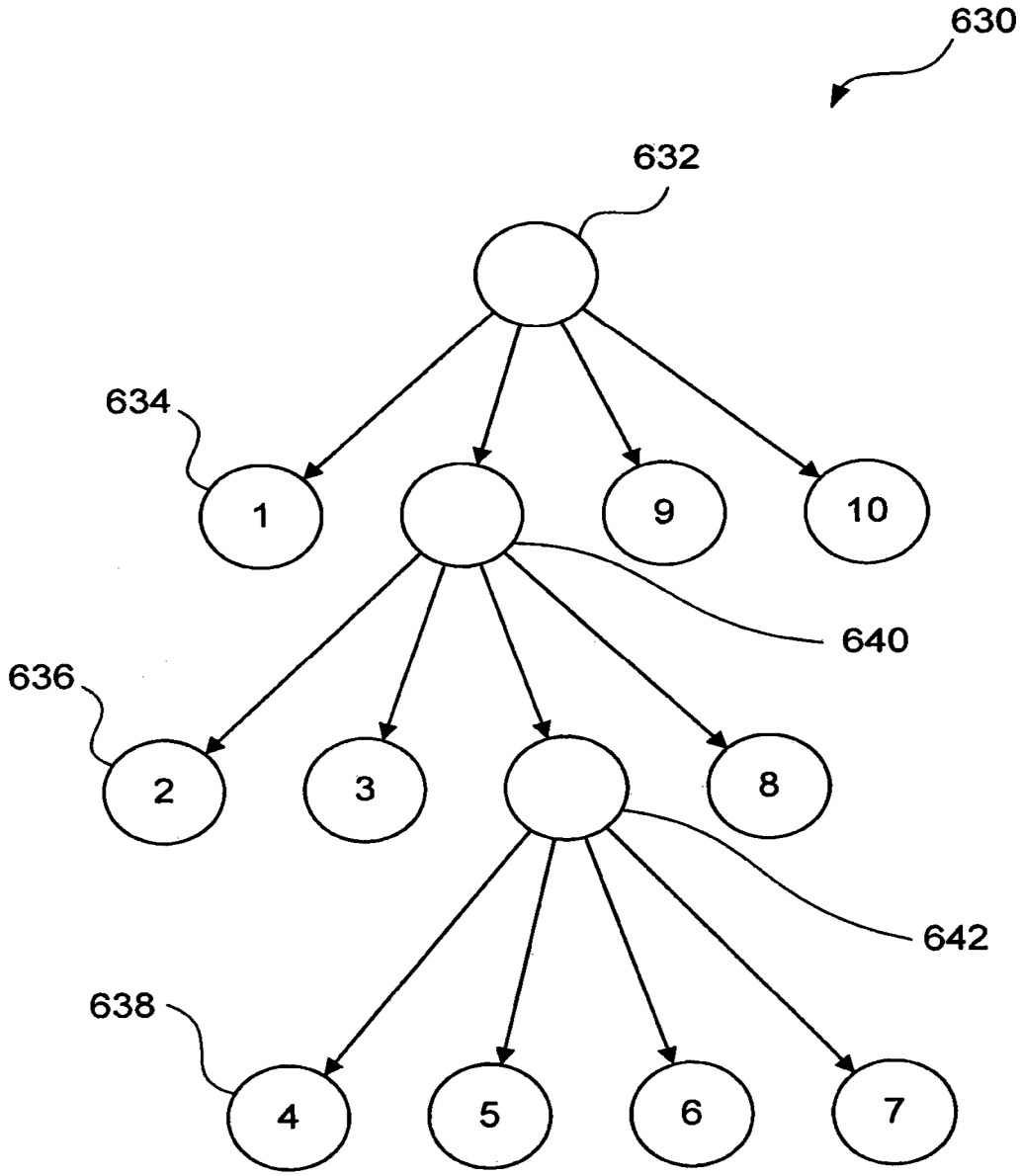

Fig. 6C

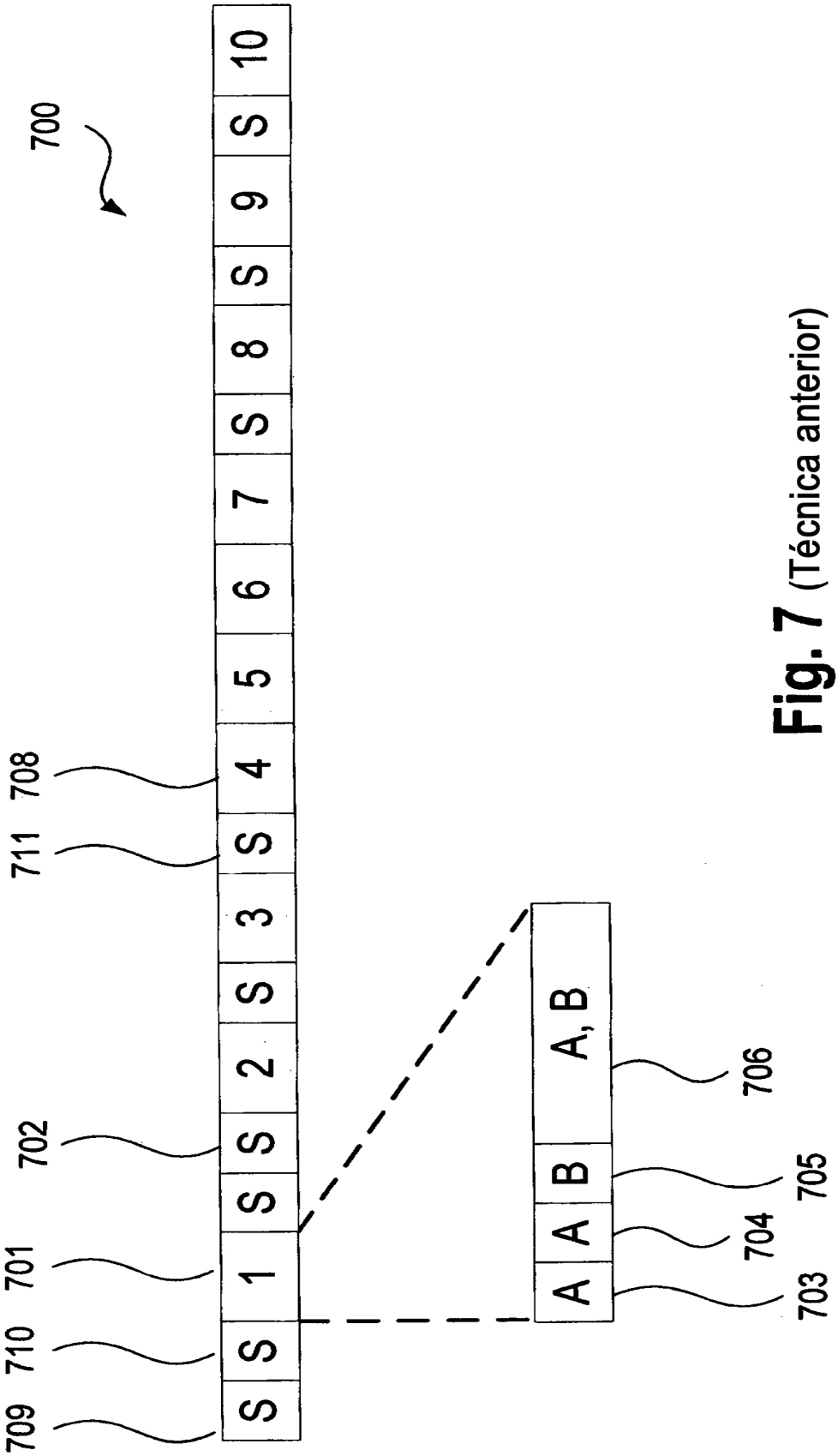

Fig. 7 (Técnica anterior)

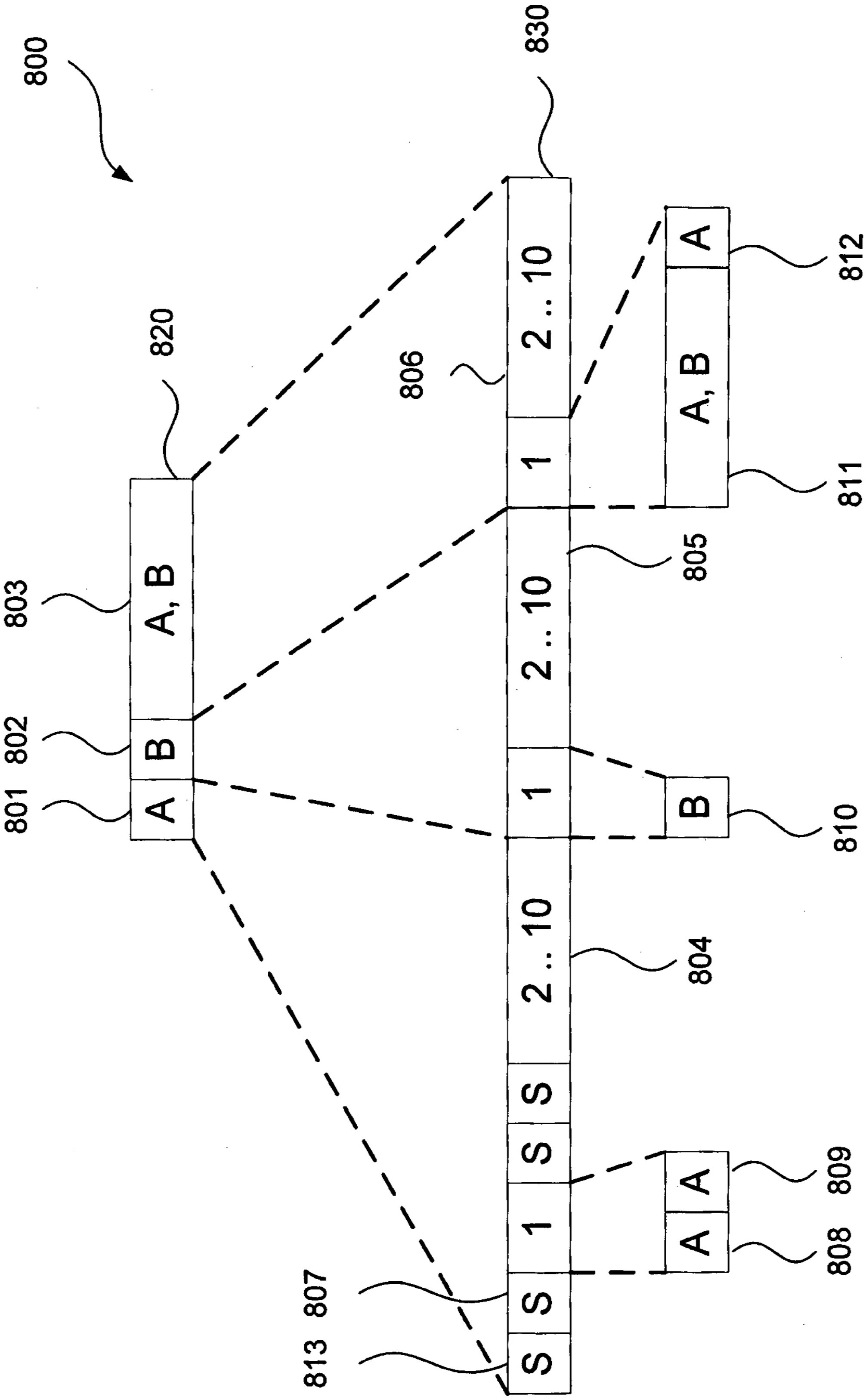
800
801
802
803
820
A
B
A, B
830
806
2 .. 10
1
805
804
S
S
1
S
S
2 .. 10
813
807
A
A
B
A, B
A
808
809
810
811
812

Fig. 8

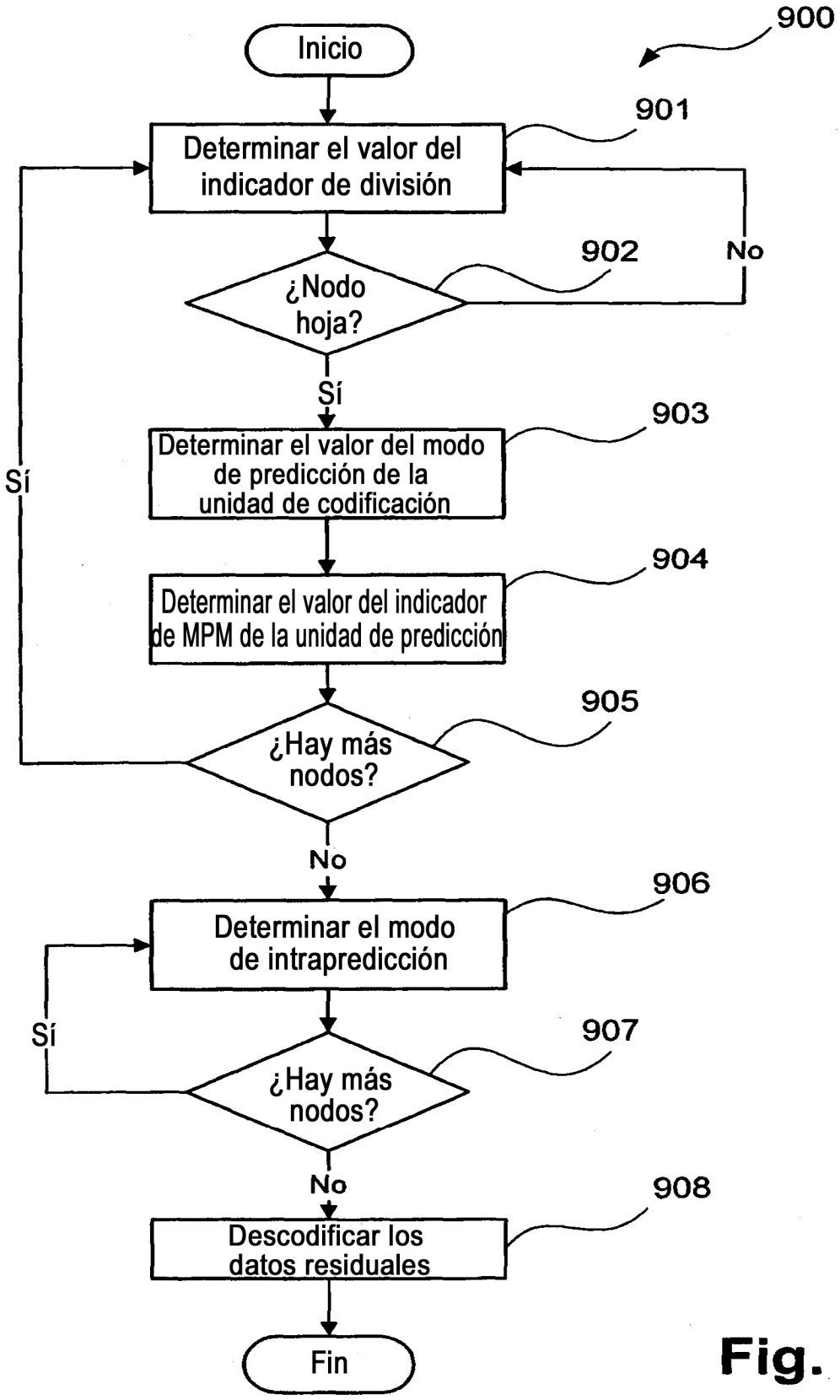
900
Inicio
901
Determinar el valor del indicador de división
902
¿Nodo hoja?
No
Sí
903
Determinar el valor del modo de predicción de la unidad de codificación
904
Determinar el valor del indicador de MPM de la unidad de predicción
905
¿Hay más nodos?
Sí
No
906
Determinar el modo de intrapredicción
907
¿Hay más nodos?
Sí
No
908
Descodificar los datos residuales
Fin

Fig. 9

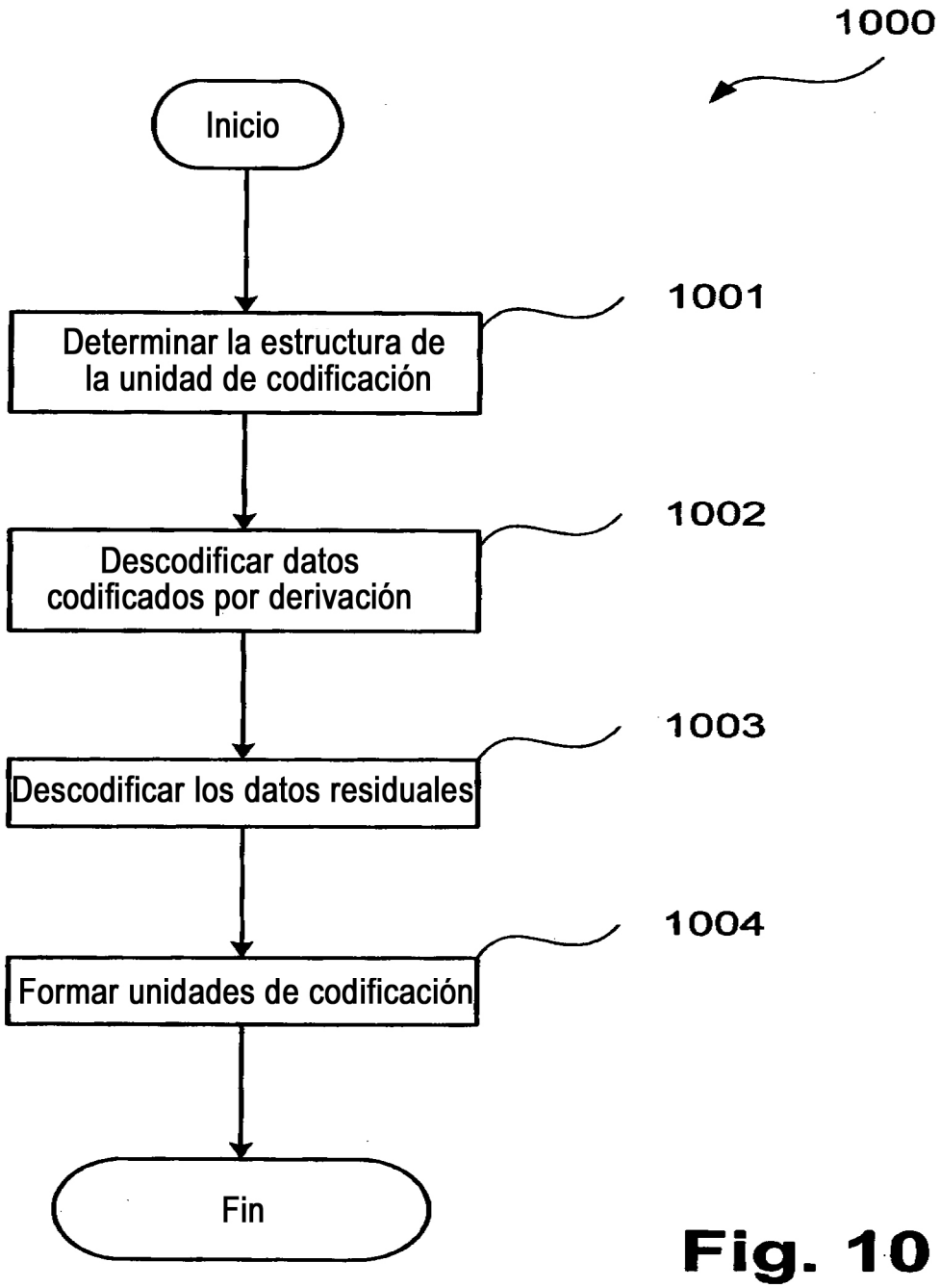
1000
Inicio
1001
Determinar la estructura de la unidad de codificación
1002
Descodificar datos codificados por derivación
1003
Descodificar los datos residuales
1004
Formar unidades de codificación
Fin

Fig. 10

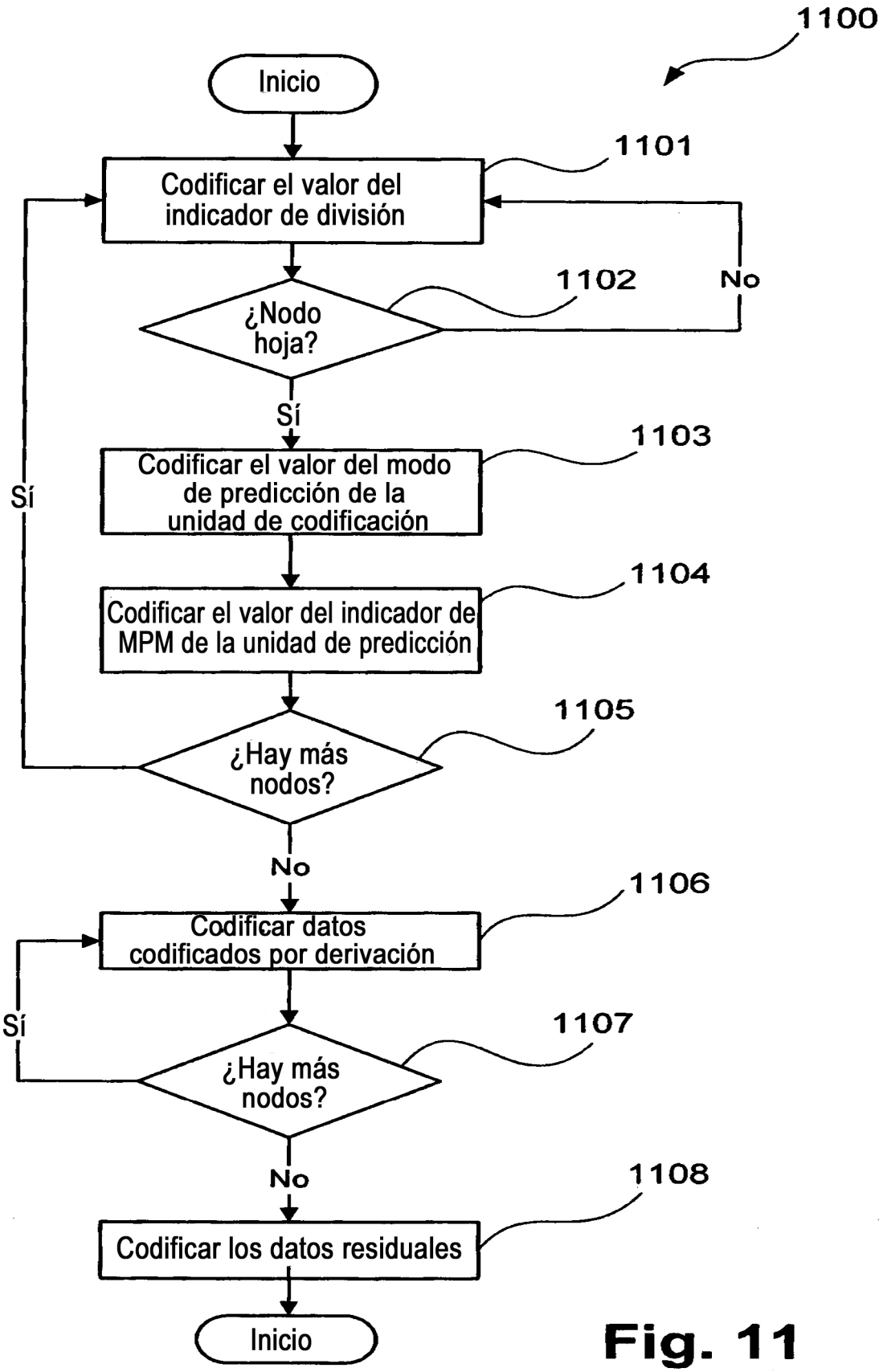
1100
Inicio
1101
Codificar el valor del indicador de división
1102
¿Nodo hoja?
No
Sí
1103
Codificar el valor del modo de predicción de la unidad de codificación
1104
Codificar el valor del indicador de MPM de la unidad de predicción
1105
¿Hay más nodos?
Sí
No
1106
Codificar datos codificados por derivación
1107
¿Hay más nodos?
Sí
No
1108
Codificar los datos residuales
Inicio

Fig. 11

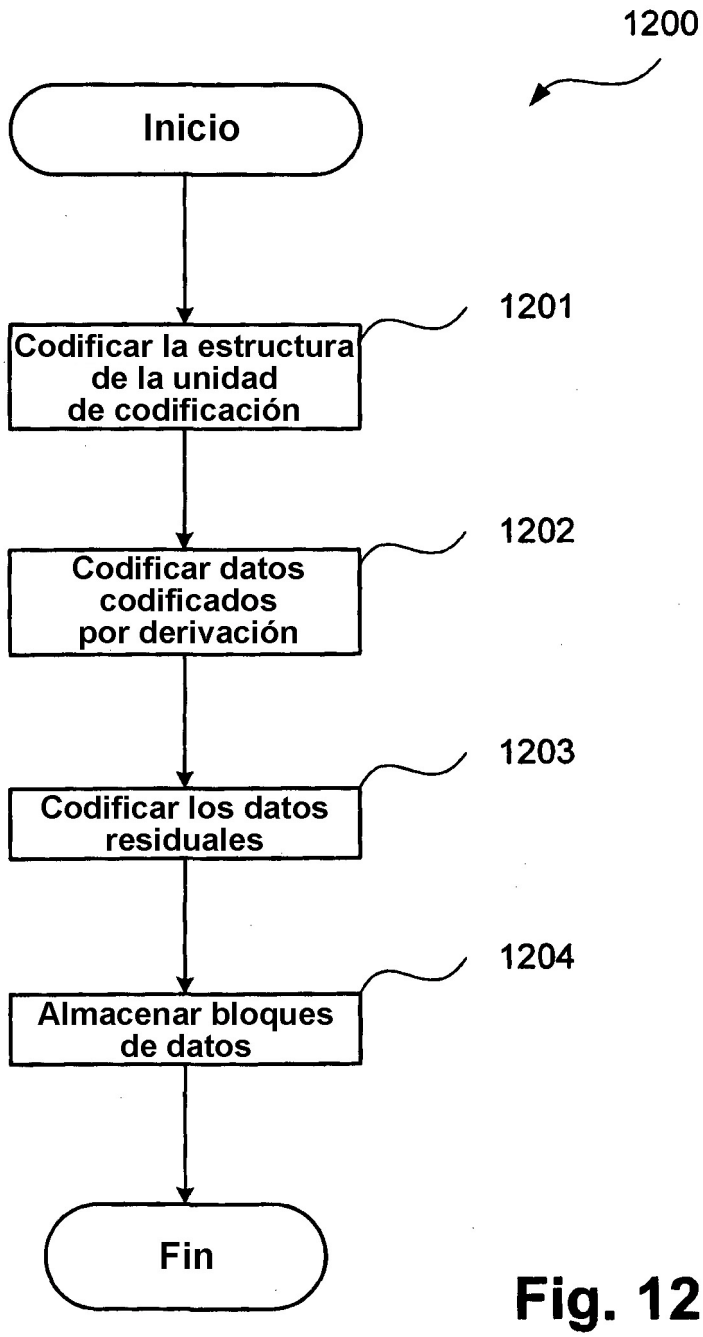
1200
Inicio
1201
Codificar la estructura de la unidad de codificación
1202
Codificar datos codificados por derivación
1203
Codificar los datos residuales
1204
Almacenar bloques de datos
Fin

Fig. 12

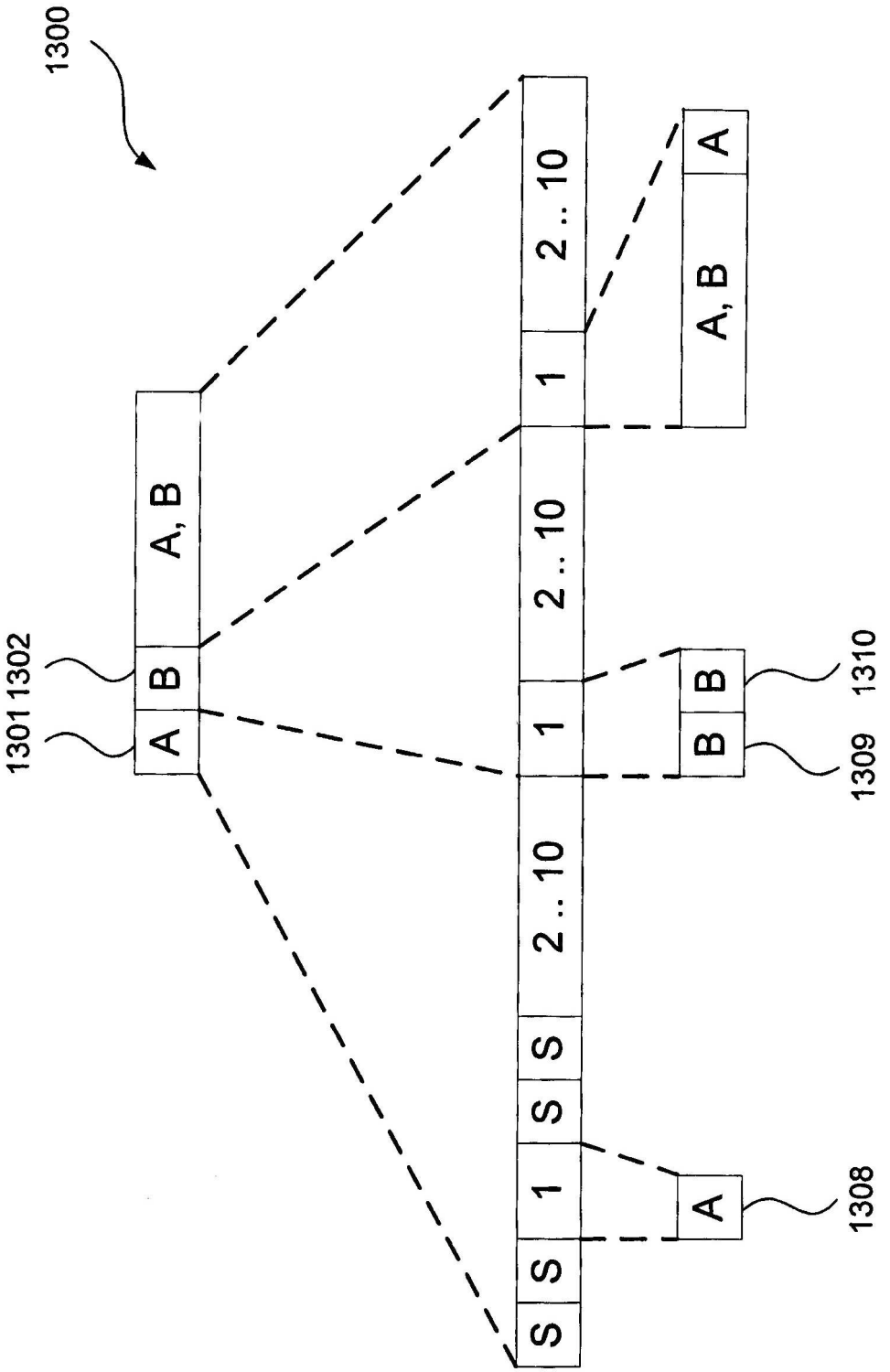

Fig. 13

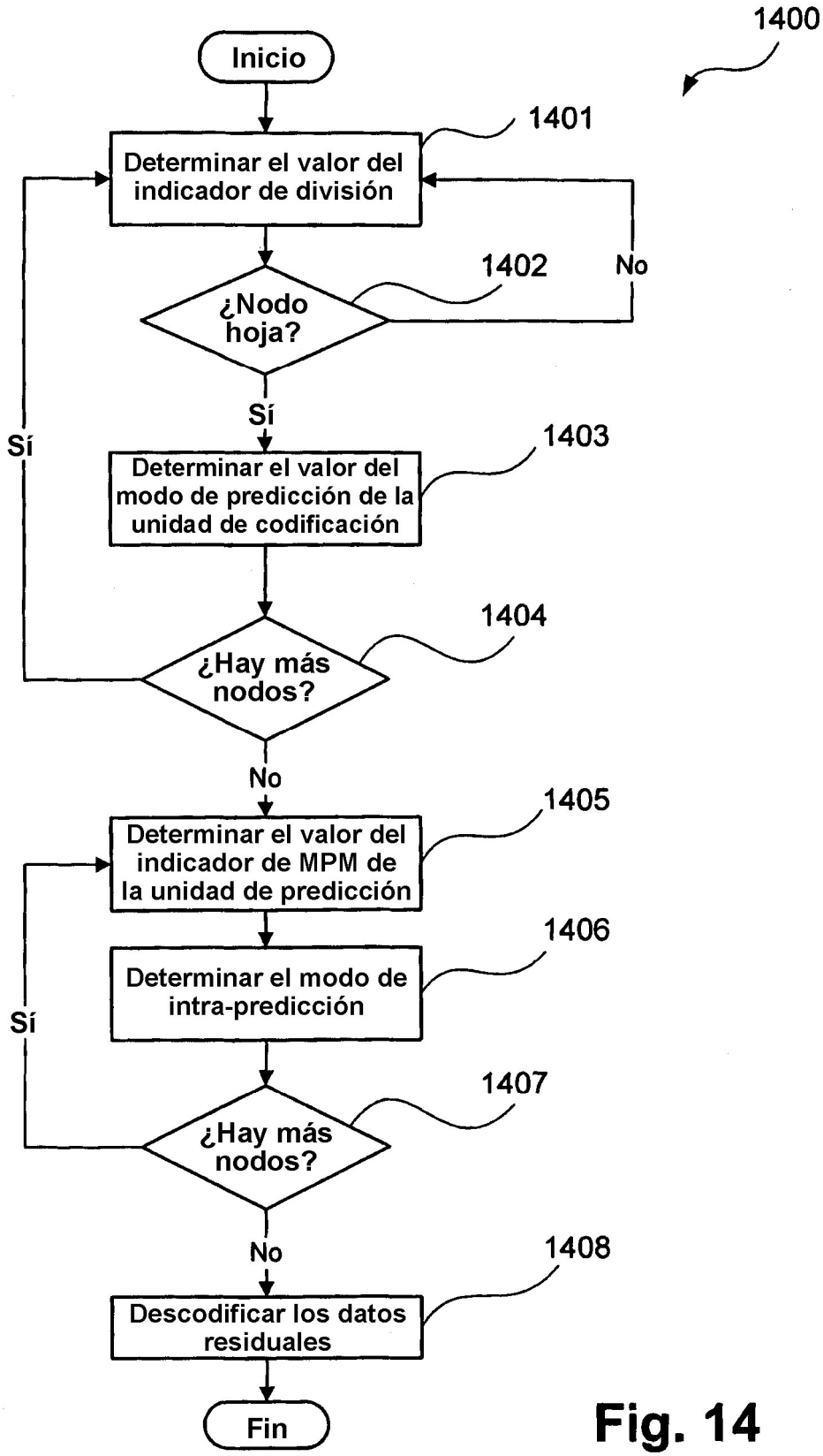
1400
Inicio
1401
Determinar el valor del indicador de división
1402
¿Nodo hoja?
No
Sí
Sí
1403
Determinar el valor del modo de predicción de la unidad de codificación
1404
¿Hay más nodos?
No
Sí
1405
Determinar el valor del indicador de MPM de la unidad de predicción
1406
Determinar el modo de intra-predicción
1407
¿Hay más nodos?
No
1408
Descodificar los datos residuales
Fin

Fig. 14

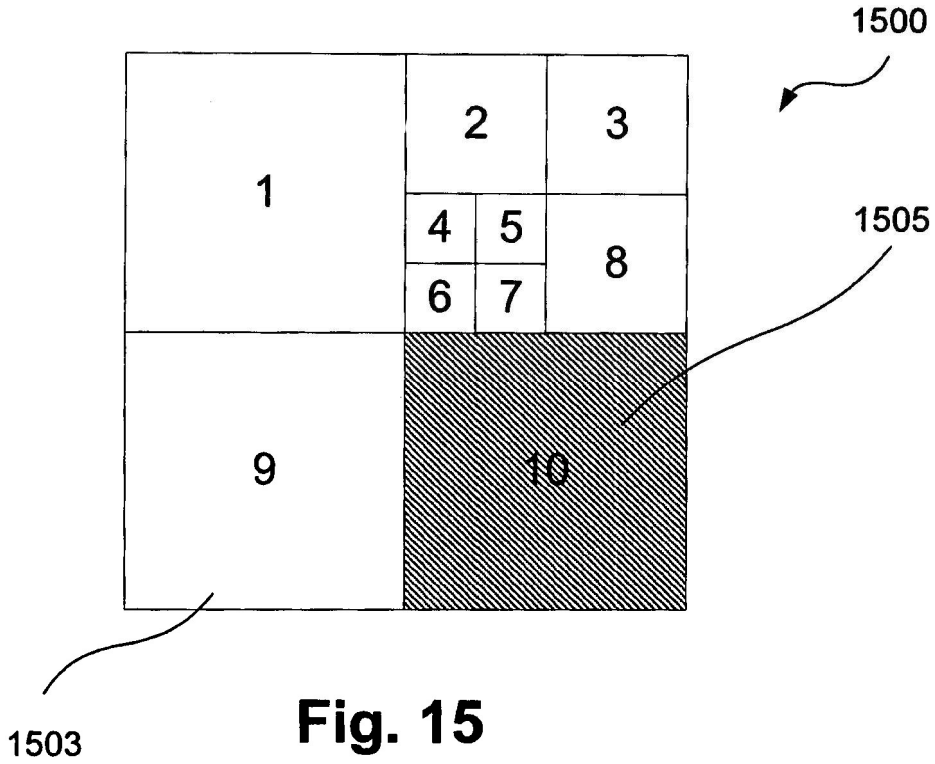

Fig. 15

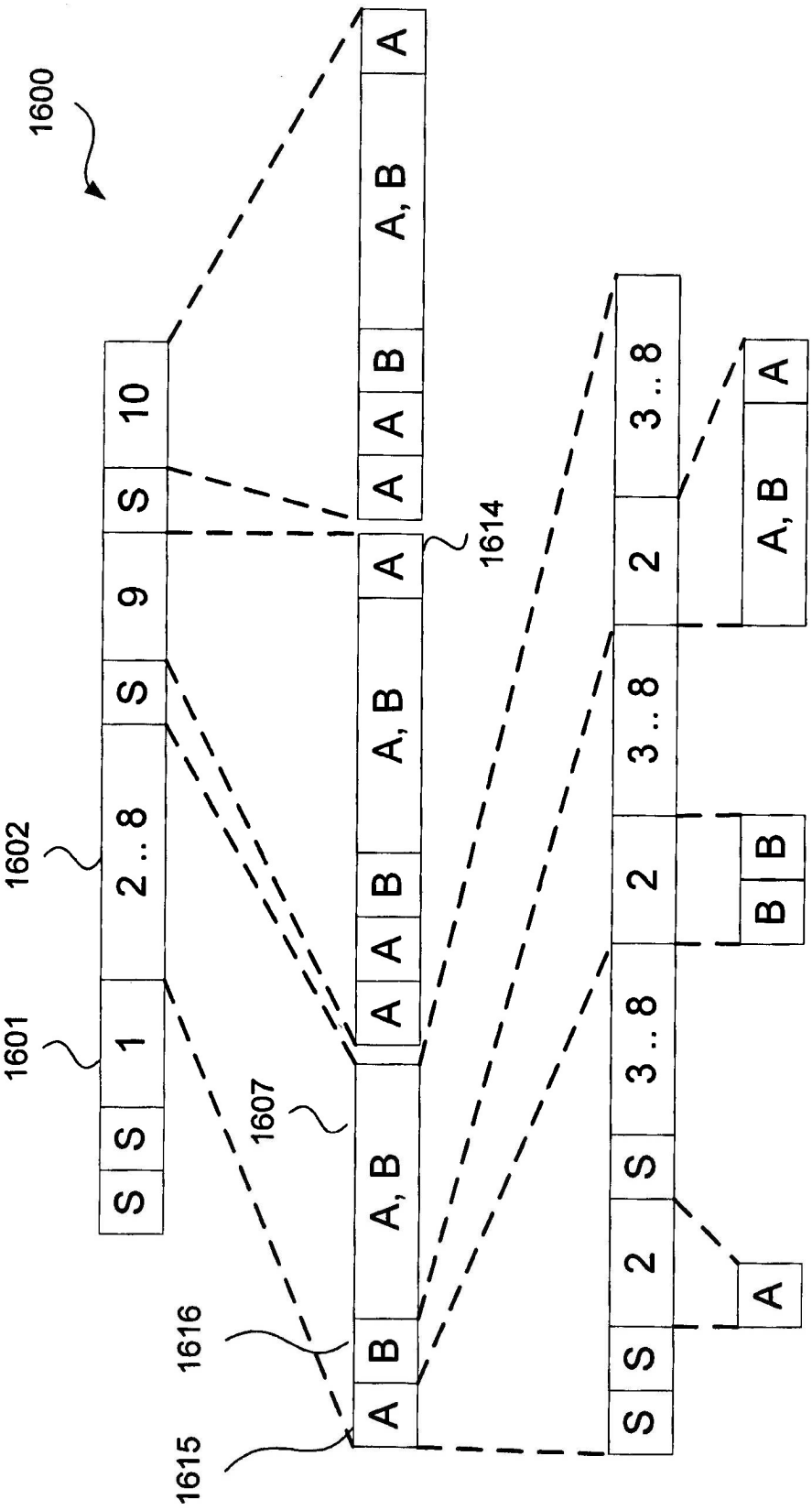
1600
1601
1602
1607
1614
1615
1616
S
S
1
2 .. 8
S
9
S
10
A
B
A
A
A, B
A
A
B
A
A, B
A
2
3 .. 8
S
2
3 .. 8
S
2
3 .. 8
2
A, B
A
B
B
A

Fig. 16

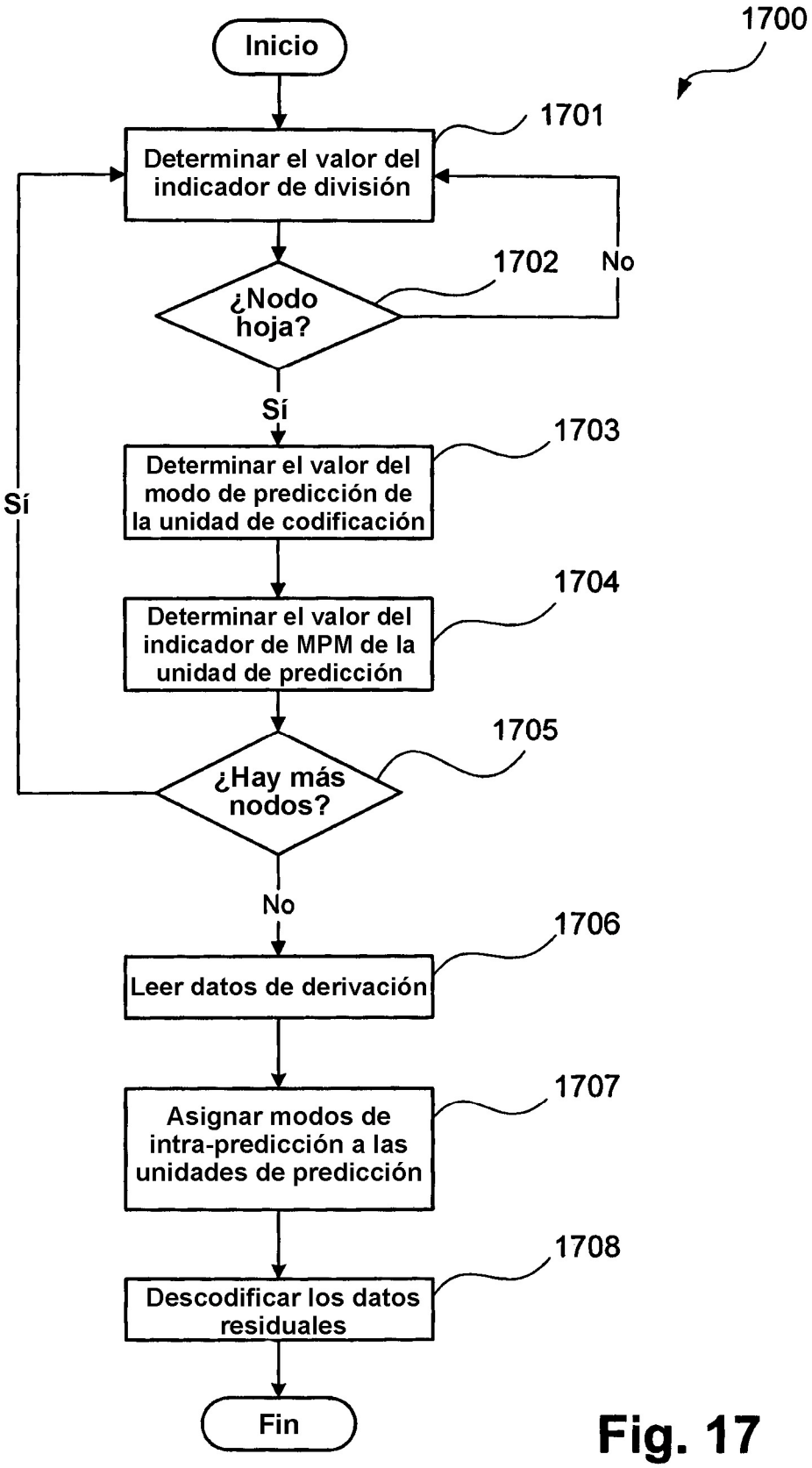
1700
Inicio
1701
Determinar el valor del indicador de división
1702
¿Nodo hoja?
No
Sí
1703
Determinar el valor del modo de predicción de la unidad de codificación
Sí
1704
Determinar el valor del indicador de MPM de la unidad de predicción
1705
¿Hay más nodos?
No
1706
Leer datos de derivación
1707
Asignar modos de intra-predicción a las unidades de predicción
1708
Descodificar los datos residuales
Fin

Fig. 17

REFERENCIAS CITADAS EN LA DESCRIPCIÓN

Esta lista de referencias citada por el solicitante es únicamente para mayor comodidad del lector. No forman parte del documento de la Patente Europea. Incluso teniendo en cuenta que la compilación de las referencias se ha efectuado con gran cuidado, los errores u omisiones no pueden descartarse; la EPO se exime de toda responsabilidad al respecto.

Literatura no patente citada en la descripción

- **SEREGIN V et al.** Utilisation of CABAC equal probability mode for intra modes coding. *6th JCT-VC meeting, MPEG meeting,* 14 July 2011
- **YEO C et al.** Non-CE6: On intra prediction mode coding. *7th JCT-VC meeting,* 21 November 2011
- **MISRA K et al.** Using CABAC bypass mode for coding intra prediction mode. *7th JCT-VC meeting,* 21 November 2011
- **SASAI H et al.** Fixed probability coding for Intra mode. *6th JCT-VC meeting,* 14 July 2011
- **BROSS B et al.** High Efficiency Video Coding (HEVC) text specification Working Draft 5. *7th JCT-VC meeting,* 21 November 2011
- **SZE V et al.** Parallel Context Processing of Coefficient Level. *6th JCT-VC meeting,* 14 July 2011
- **SASAI H et al.** Modified MVD coding for CABAC. *97 MPEG Meeting,* 18 July 2011
- **FRANCOIS E et al.** CE6b: Intra mode coding with 4 MPMs and mode ranking. *98 MPEG MEETING,* 21 November 2011, (m21801
- **D. MARPE et al.** Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on circuits and systems for video technology,* 01 July 2003, vol. 13 (7), 620-636